Politecnico di Torino

Politecnico di Torino

Master's Degree in Electronic Engineering

A.a. 2025/2026

Graduation Session March 2026

Firmware Development for a Neuroprosthetic Device

Supervisors:
Prof. Danilo Demarchi
Ph.D. Paolo Motto Ros
M.Sc Sara Lo Vecchio
M.Sc Letizia Cantore
M.Sc Stefano Concadoro

Candidate:
Marika Di Martino

Abstract

Lower Urinary Tract (LUT) dysfunctions, such as urinary incontinence or urinary retention, represent a clinically relevant group of conditions that significantly impact patients' quality of life. Neuroprosthetic devices used for the treatment of the LUT dysfunctions are typically based on continuous open-loop stimulation paradigms, which may lead to voiding dysfunctions due to neural adaptation. In contrast, closed-loop stimulation strategies have demonstrated improved bladder capacity and voiding efficiency compared to continuous stimulation.

This work has been developed within the framework of the Bladder Control project, which aims to overcome the limitations of open-loop approaches by introducing closed-loop neuroprosthetic systems. Such systems rely on the acquisition of electroneurographic (ENG) signals and on decoding strategies based on machine learning (ML) techniques to estimate bladder-filling states and apply nerve stimulation accordingly.

The objective of this work is the development of an acquisition platform for ENG signals from the pudendal nerve using implantable electrodes. Since signal quality depends on the electrode-tissue electrical interface, the integrity of the connection was assessed through an impedance check.

Neural activity is acquired using the fully integrated Intan RHD2132 electrophysiology interface chip, controlled by an nRF5340 microcontroller via Serial Peripheral Interface (SPI). Bladder-filling states are decoded through a classification algorithm executed on the microcontroller, and the results are transmitted to a personal computer via Bluetooth Low Energy (BLE) using an nRF52840 dongle. Electrode impedance was measured using the module provided by the Intan chip.

A Graphical User Interface (GUI) has been developed to enable user interaction with the system, allowing selection among recording and impedance check modes.

Neural signal acquisition was performed simultaneously on four channels with a 16-bit resolution and a sampling rate of 10 kHz per channel, resulting in a required data throughput of at least 640 kbit/s. Electrode impedance measurements were also experimentally validated.

Acknowledgements

Ringrazio Paolo Motto Ros per avermi dato l'opportunità di far parte del progetto "Bladder Control", e i miei supervisori: Sara Lo Vecchio per tutto ciò che mi ha insegnato, Letizia Cantore e Stefano Concadoro per il prezioso supporto durante la fase finale del lavoro.
Un sentito grazie agli amici di sempre, al gruppo di Torino che dal primo anno è stato un punto di riferimento, e ai compagni del corso di elettronica.
Ringrazio inoltre i medici del reparto di Chirurgia Generale e Oncologica dell'Ospedale Mauriziano di Torino, sarò sempre immensamente grata per il loro straordinario lavoro e per l'affetto ricevuto da tutto il reparto.
Un grazie speciale va alla mia famiglia, per il continuo supporto e tutto l'amore che mi ha dato, riuscendo a colmare la distanza. Infine, ringrazio i miei pelosi preferiti: Romeo e Leo, per il loro incondizionato amore.

Table of Contents

List of Figures

Chapter 1

Introduction

Ancient Egyptian medical texts described vertebral fractures as "an ailment not to be treated." For many centuries, and even until the period of the World War II, individuals with spinal cord injuries often died from the initial trauma or experienced severe complications related to secondary conditions such as respiratory or circulatory failure. Significant progress in the management of spinal cord injury began to emerge during the 1970s, when advances in emergency care and rehabilitation strategies substantially improved patient outcomes.

The spinal cord plays a crucial role in transmitting information between the brain and the rest of the body. In addition to acting as a communication pathway, it is also capable of generating coordinated motor activity through neural networks known as central pattern generators. When the spinal cord is damaged due to trauma or disease, these mechanisms are disrupted, leading to impairments in motor, sensory, and autonomic functions.

Today, improvements in emergency medicine and rehabilitation have significantly increased the life expectancy and functional independence of individuals with spinal cord injury (SCI). Modern therapeutic approaches aim to maximize functional capacity and improve quality of life through targeted interventions.

Neuroprostheses (NPs) are electronic systems designed to stimulate neural structures in order to restore or improve bodily functions that have been lost because of damage to the central or peripheral nervous system. These devices include both surface stimulators, which deliver electrical current through the skin, and implanted stimulators that directly target specific nerves. Neuroprosthetic technologies range from relatively simple devices used to enhance muscle activity to more sophisticated systems implanted in the brain or spinal cord. Currently, NPs are used to assist a wide range of functions, including hand movement, postural control, standing,

walking, respiration, bladder control, and pain management [1].

The lower urinary tract (LUT) is responsible for storing and eliminating urine through the coordinated activity of the bladder, bladder neck, and urethral rabdomiosphinter. The proper functioning of these smooth and striated muscles depends on complex neural control mechanisms involving both the central and peripheral nervous systems. Neurological disorders such as spinal cord injury, multiple sclerosis, or brain lesions (for example Parkinson's disease or traumatic brain injury) may lead to neurogenic lower urinary tract dysfunction (LUTD).

These dysfunctions can manifest as urinary incontinence and/or incomplete bladder emptying, significantly affecting the social and emotional wellbeing of affected individuals. In recent years, neuroprosthetic devices have been increasingly adopted to restore bladder function or alleviate LUT symptoms. Such devices typically use electrodes positioned near specific nerves to deliver continuous or intermittent electrical stimulation.

However, the long-term effectiveness of these treatments remains variable. For instance, the success rate of sacral nerve stimulation (SNS), defined as a reduction of symptoms greater than 50%, ranges approximately between 29% and 76%, while tibial nerve stimulation (TNS) shows success rates of around 54–59%.

The Food and Drug Administration has approved neuroprosthetic systems that operate using a continuous open-loop stimulation paradigm, in which electrical stimulation is delivered without considering the bladder filling state. This approach may lead to neural adaptation, potentially reducing the effectiveness of bladder emptying over time.

For this reason, researchers are investigating conditional or closed-loop stimulation strategies, which adapt stimulation according to the physiological state of the bladder. Experimental studies in both animals and humans suggest that state-dependent stimulation, such as pudendal nerve stimulation, may improve bladder capacity and voiding efficiency compared with continuous stimulation. Nevertheless, several challenges remain, including accurately detecting bladder filling and determining optimal stimulation parameters, and standardized clinical protocols for LUTD treatment are still under development [2].

1.1 Goal

The goal of this thesis is to develop the Firmware to allow data acquisition (Recording), impedance test and the Software for the graphical user interface (GUI) that allows the user to interact with the neuroprosthetic system.

The GUI provides access to four main functionalities: Acquisition of ElectroNeuroGraphic (ENG) signals, Stimulation, Electrodes Impedance Check, and Decoding of bladder-filling states from pudendal nerve ENG signals.

Regarding data acquisition, the firmware implements the communication between the electrophysiological interface chip RHD2132 and the nRF5340 microcontroller via Serial Peripheral Interface (SPI), and the data transfer between the microcontroller and the PC via Universal Asynchronous Receiver-Transmitter (UART). The firmware has been optimized to maximize the UART transmission speed in order to handle the continuous data stream from the chip. Given the maximum baud rate of 921600 bit/s, the system supports the recording of up to four channels with a sampling rate of 10 $kHz/channel$.

For the interaction with the Graphical User Interface (GUI), the firmware enables Bluetooth Low Energy (BLE) communication between the microcontroller and the GUI, supported by the nRF52840 dongle. This allows the user to send the desired commands.

The firmware for the Impedance Check allows the user to verify the quality of the electrode before recording or stimulation, ensuring reliability and safety of the acquired data and delivered stimuli. This functionality exploits the internal impedance measurement module of the Intan chip.

1.2 Chapters organization

The thesis is organized into five chapters:

- `Chapter 1:` Introduction to neuroprosthetic devices, with a focus on bladder-related applications, closed-loop neuroprosthetic systems, and the objectives of this thesis.

- `Chapter 2:` State of the Art. This chapter describes the working principles of the communication protocols (UART, BLE, and SPI) and discusses the motivation for transitioning from open-loop to closed-loop systems.

- `Chapter 3:` Description of the system setup, with a detailed focus on each system component.
- `Chapter 4:` This chapter explains the firmware and software development process.
- `Chapter 5:` Results. This chapter presents and discusses the results obtained.
- `Chapter 6:` Conclusions and Future Perspectives. This chapter discusses the goals reached and the future goals.

Chapter 2

State of Art

2.1 Basics of UART

The Universal Asynchronous Receiver/Transmitter (UART) is a hardware communication protocol that enables asynchronous serial data exchange between electronic devices. The term asynchronous indicates that data transmission does not rely on a shared clock signal between the transmitter and the receiver. Instead, each device operates using its own internal timing while maintaining synchronization through predefined communication parameters such as the baud rate.

UART communication is widely used in embedded systems, microcontrollers, and computers as a simple and reliable method for device-to-device data transfer. Compared to other communication protocols, UART requires only two signal lines to operate, making it a relatively straightforward interface for serial communication [3]. Each UART interface includes two main signals:

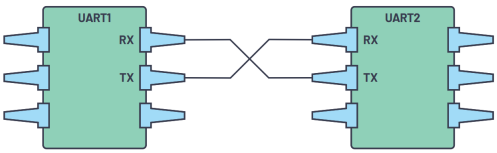


Figure 2.1: Two UARTs directly communicate with each other. [3]

- Transmitter (Tx)
- Receiver (Rx)

These lines allow devices to exchange serial data. The transmitter is responsible for sending information, while the receiver collects incoming data from another device

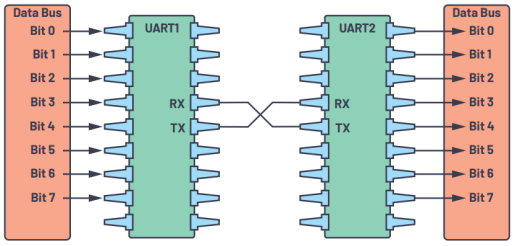


Figure 2.2: UART with data bus. [3]

[3]. In a typical configuration, the transmitting UART receives data from a system data bus in parallel form. This data is then converted into a serial bit stream and transmitted over a communication line to another UART module. At the receiving end, the incoming serial data is converted back into parallel format so that it can be processed by the receiving system. In this way, UART acts as an interface that enables communication between systems operating with parallel data structures.

Each UART module includes dedicated pins for transmission and reception. Because communication occurs asynchronously, both the transmitting and receiving devices must be configured with the same baud rate, which represents the speed at which data bits are transmitted over the communication channel. In serial communication, the baud rate typically corresponds to the maximum number of bits transmitted per second.

Unlike synchronous protocols, UART does not rely on an external clock signal shared between devices. The transmitter generates the serial data stream using its own internal clock, while the receiver independently samples the incoming signal using its internal timing mechanism. Proper synchronization between the two devices is therefore achieved by configuring identical baud rate on both devices.

If the baud rate differs significantly between the transmitting and receiving devices, timing errors may occur during data sampling, potentially leading to corrupted or lost information. In practice, UART communication can tolerate a small mismatch in timing, typically up to approximately 10%, before synchronization problems begin to affect data integrity [3].

2.1.1 Data Transmission

In UART communication, information is transmitted in the form of structured data packets. The interface between the transmitter and receiver manages the

creation of these serial packets and controls the electrical signals on the communication lines. A typical UART packet includes several elements: a start bit, a data frame, an optional parity bit, and one or more stop bits [3]. When no data

Figure 2.3: UART packet. [3]

is being transmitted, the UART transmission line remains at a high voltage level. Communication begins when the transmitting device pulls the line from high to low, generating the start bit. This voltage transition signals the receiving UART that a new data packet is beginning. Once this transition is detected, the receiver starts to read the incoming signal according to the configured baud rate [3].

The data frame contains the information being transmitted. Depending on the communication configuration, it typically includes between five and eight data bits when parity is enabled. If parity checking is not used, the frame may contain up to nine bits. In most implementations, the least significant bit (LSB) is transmitted first [3].

To provide a basic form of error detection, UART communication may include a parity bit. Parity indicates whether the number of logical high bits (value 1) within the data frame is even or odd. During transmission, disturbances such as electromagnetic interference, incorrect baud rate configuration, or long transmission distances may alter one or more bits [3].

When the receiver obtains the data frame, it counts the number of bits with value 1 and compares this result with the expected parity condition. In even parity, the total number of logical high bits in the frame must be even. Conversely, in odd parity, the total number must be odd. If the calculated parity matches the received parity bit, the data is assumed to have been transmitted correctly. Otherwise, the receiver detects that an error has likely occurred during transmission [3].

The stop bit indicates the end of the packet. After sending the data and the optional parity bit, the transmitting UART drives the communication line back to a high voltage level for one or two bit periods, marking the completion of the transmission [3].

2.1.2 Steps of UART Transmission

- The transmitting UART first receives data from the system data bus in parallel format.

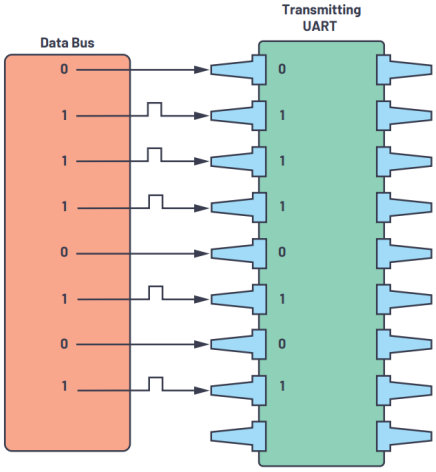


Figure 2.4: Data bus to the transmitting UART. [3]

- Before transmission, the UART module prepares the communication frame by adding the necessary control bits, including the start bit, an optional parity bit, and one or more stop bits.
- The complete frame is then transmitted over the communication line as a serial bit stream, beginning with the start bit and ending with the stop bit(s). During this process, the receiving UART samples the incoming signal according to the configured baud rate.
- After detecting the packet, the receiving UART removes the control bits (start, parity, and stop bits) in order to isolate the useful data.
- Finally, the received serial information is reconstructed into parallel format and delivered to the data bus of the receiving system.

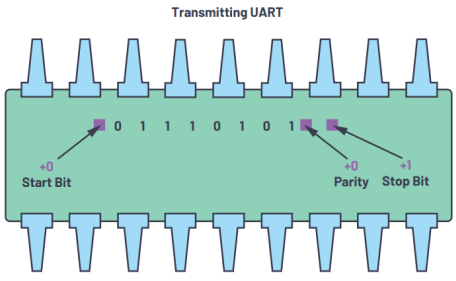


Figure 2.5: UART data frame at the Tx side. [3]

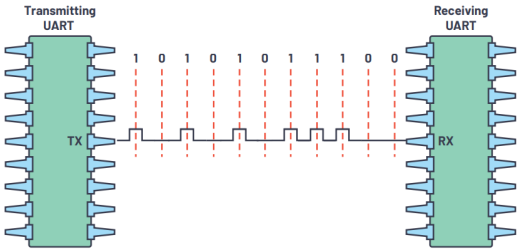


Figure 2.6: UART transmission. [3]

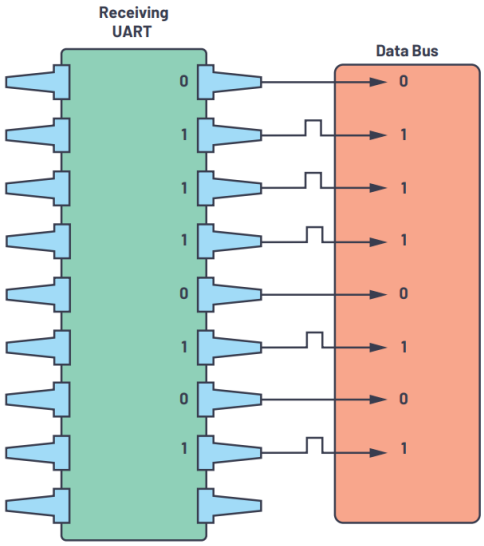


Figure 2.7: Receiving UART to data bus. [3]

2.2 Bluetooth Low Energy Fundamentals

Traditional Bluetooth technology is well suited for applications that require continuous data transmission, such as wireless audio streaming. In these scenarios, the available data throughput is sufficient to ensure stable communication without significant packet loss. However, devices such as wireless speakers must be recharged periodically due to their relatively high energy consumption.

In contrast, many modern applications—including wearable devices and large-scale Internet of Things (IoT) systems—require extremely low power consumption, since frequent battery charging or replacement is often impractical. To address these requirements, the *Bluetooth Special Interest Group (SIG)* introduced Bluetooth Low Energy (BLE) starting with the Bluetooth Core Specification version

4.0. The objective was to provide a wireless communication technology specifically optimized for low-power IoT devices.

As suggested by its name, Bluetooth LE prioritizes energy efficiency, even if this comes at the cost of reduced data throughput. This reduction is mainly achieved through two mechanisms. First, the size of transmitted packets is limited, typically ranging between 27 and 251 bytes. Second, data transmissions occur only when necessary. As a result, Bluetooth LE is particularly suitable for battery-powered devices that transmit small amounts of data intermittently.

Bluetooth LE also differs from Bluetooth Classic in several architectural aspects, including the supported device roles and network topologies. These differences arise from the distinct application domains for which the two technologies were designed, with Bluetooth LE specifically tailored for low-power and highly scalable IoT environments.

Operating band	2.4 GHz
Channel bandwidth	2 MHz
Number of RF channels	40
Maximum transmit power	0.1 W
Maximum application data throughput	1.4 Mbps
Maximum range at reduced data rates (125-500 kbps)	1000 m

Table 2.1: BLE characteristics

2.2.1 Central and Peripheral

In a BLE network, communication occurs between two types of devices: the *Central* and the *Peripheral*.

The peripheral is the device that broadcasts its presence and availability for connection, while the central is the device that actively scans for these advertisement packets. When the central detects a peripheral's advertisement, it can initiate a connection by sending a request to the peripheral. Once the connection is accepted, the two devices are considered connected. A central device has the capability to manage connections with multiple peripherals at the same time, assuming the role of the host in each connection. Peripherals can also accept connections from different centrals by restarting their advertising process after an existing connection is established.

Because the central acts as the host, it handles tasks such as connection management and a significant portion of data processing. Consequently, peripherals generally consume less power compared to centrals. This characteristic makes resource-constrained IoT devices well suited to act as peripherals, while devices with greater power resources, such as smartphones, typically take on the central role.

BLE communication occurs over a total of 40 frequency channels. These channels are divided into three primary advertisement channels and 37 secondary channels. The primary channels are primarily used for broadcasting advertisements, whereas the secondary channels are mainly utilized for data transfer once a connection has been established, although they may also be used for advertising in certain cases. To provide a degree of redundancy, BLE advertising packets are transmitted across

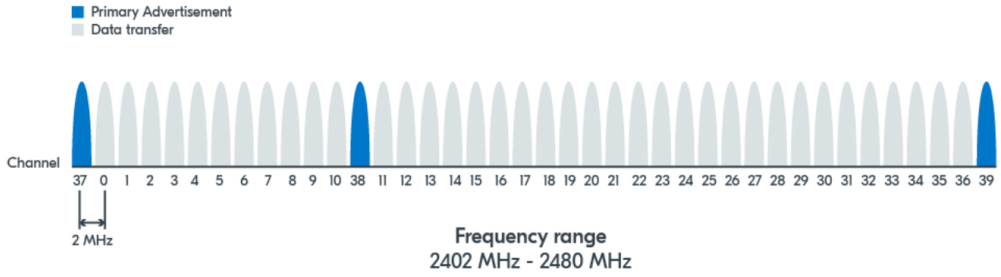


Figure 2.8: BLE channels

all three primary advertising channels: 37, 38, and 39. At the same time, scanning devices monitor these channels to detect nearby advertising peripherals.

Because advertising packets play a critical role in establishing connections, the selection of the primary advertising channels is carefully optimized. Although channels 37, 38, and 39 appear to be consecutive, they are not physically adjacent, as illustrated in Figure 2.8. This separation helps minimize interference from neighboring channels. Moreover, these specific channels experience relatively low levels of interference from other technologies operating in the ISM band, such as Wi-Fi, further improving the reliability of BLE advertising.

2.2.2 ATT & GATT: Data representation and exchange

Once a BLE connection has been established, devices must be able to exchange data in both directions. This requires well-defined data structures and communication protocols designed specifically for bidirectional transfer.

In Bluetooth Low Energy, the *Attribute Protocol (ATT)* together with the *Generic Attribute Profile (GATT)* specify how data is structured, accessed, and exchanged between connected devices, ensuring interoperability and efficient communication.

The Attribute Protocol

The Attribute Protocol (ATT) serves as the foundation for data handling during the connection phase between BLE devices. It operates using a client-server model, where the server maintains the data and can either provide it to the client proactively or respond to requests initiated by the client.

Both central and peripheral devices can assume the role of client or server depending on the application requirements and the type of data being transmitted. Typically, the peripheral acts as the server because it is the device that collects and stores the data, whereas the central generally acts as the client, requesting and receiving data from the server.

- **GATT server:** A device responsible for storing data and offering mechanisms through which a GATT client can access this information.
- **GATT client:** A device that retrieves and interacts with data stored on a GATT server using defined GATT operations.

Within the ATT layer, data is organized using a structure known as an *attribute*, which serves as the fundamental unit for storing information on the GATT server. A single GATT server can maintain multiple attributes simultaneously, allowing it to manage diverse types of data and provide them to clients as needed.

The Generic Attribute Profile

The Generic Attribute Profile (GATT) extends the functionality of ATT by arranging attributes into a hierarchical framework consisting of *profiles*, *services*, and *characteristics*. This organization provides a standardized approach for representing complex data in BLE applications.

A *profile* specifies a complete application scenario, such as the Heart Rate Profile, and comprises one or more *services*. Each service contains multiple *characteristics*, with each characteristic representing an individual piece of data (for example, a heart rate measurement) and potentially including additional descriptors, such as units or valid ranges.

By utilizing this hierarchical structure, GATT enables consistent and interoperable data transfer between BLE devices, allowing applications to efficiently exchange information while maintaining flexibility.

2.2.3 Advertisement packet

The structure of a BLE packet is illustrated in Figure 2.9, with the central component referred to as the *Protocol Data Unit (PDU)*. Depending on the purpose of the packet, the PDU can be either an *advertising PDU* (also called an advertising channel PDU) or a *data PDU* (also referred to as a data channel PDU). Advertising PDUs are used for broadcasting device availability, while data PDUs carry information once a connection has been established.

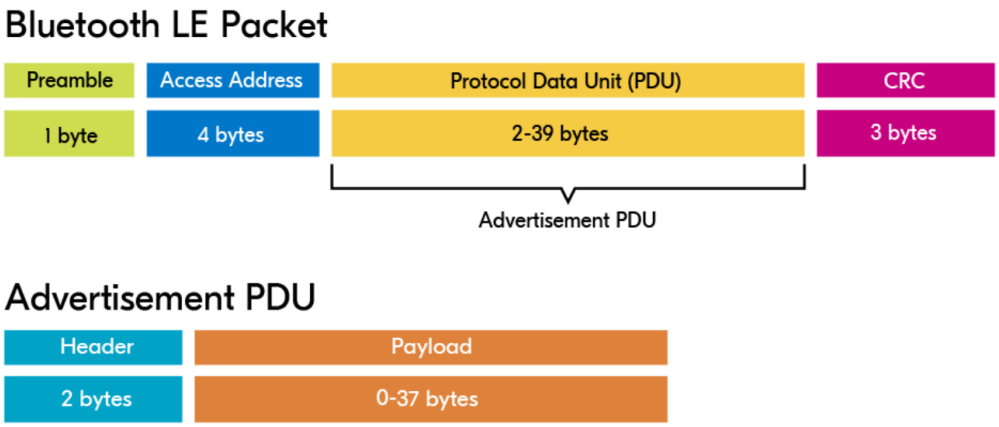


Figure 2.9: Bluetooth LE packet structure

The advertising PDU is composed of two main components: a header and a payload. The header contains several fields that define the properties and behavior of the advertising packet, including:

- **PDU Type:** Specifies the type of advertisement, such as ADV_IND, as described in the section on advertising types.
- **RFU:** Reserved for future use.
- **ChSel:** Indicates support for the LE Channel Selection Algorithm #2 when set to 1.
- **TxAdd (Transmitter Address):** Set to 0 or 1 depending on whether the transmitter's address is public or random.

- **RxAdd (Receiver Address):** Set to 0 or 1 depending on whether the receiver's address is public or random.

- **Length:** Specifies the size of the payload in bytes.

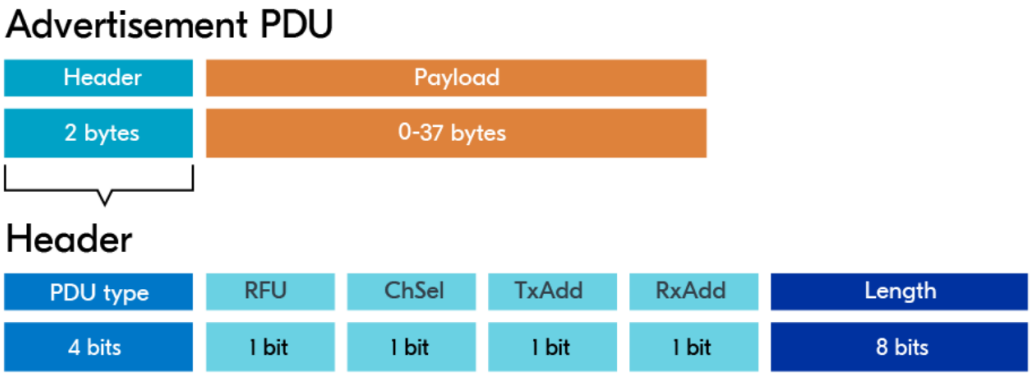


Figure 2.10: Advertisement PDU header

The payload of an advertising PDU is structured into two main sections. The first six bytes are reserved for the advertiser's address, known as *AdvA*, while the remaining bytes carry the actual advertisement information, referred to as *AdvData*.

- **AdvA:** The unique Bluetooth address of the device sending the advertisement.

- **AdvData:** Contains the data associated with the advertisement, such as device capabilities or service information.

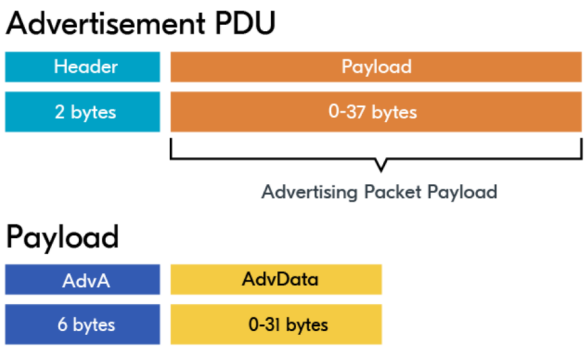


Figure 2.11: Advertisement PDU payload

The advertisement data section is represented as shown in the Figure 2.12.

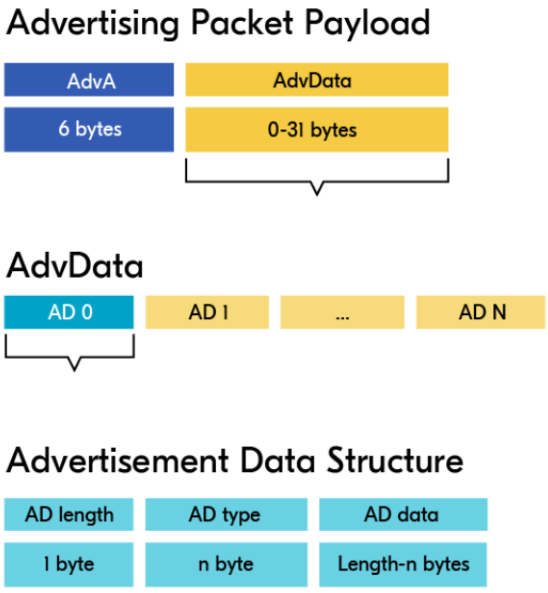


Figure 2.12: Advertising packet payload

The advertisement data packet is organized into multiple elements known as *advertisement data structures* (AD structures). Each AD structure includes a length field, a type field (*AD Type*) to specify the kind of data, and a field containing the actual data (*AD Data*). It is worth noting that the most frequently used AD Type

occupies one byte.

2.3 Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is a short-range serial communication protocol designed for high-speed data transfer, typically faster than I^2C, making it suitable for applications requiring higher throughput. SPI supports several configurations, but the four-wire variant is the most commonly used. The four primary signals in this interface are:

- SCLK (serial clock)
- MOSI (master-out-slave-in)
- MISO (master-in-slave-out)
- CS (chip select)

These SPI signals may be referred to by alternative names in different sources, but their functions remain consistent. SPI communication follows a master-slave architecture, allowing a single master to interface with multiple slave devices.

The **SCLK** line carries the serial clock generated by the master, which is used for synchronizing data transmission and sampling. As a result, the master must continuously provide clock pulses whenever communication occurs on the SPI bus.

The **MOSI** (Master-Out-Slave-In) line transmits serial data from the master to the connected slaves. Conversely, the **MISO** (Master-In-Slave-Out) line carries serial data from the slaves back to the master.

The **CS** (Chip Select) signal, also known as **Slave Select (SS)**, is an active-low control line from the master that activates a specific slave device. Because multiple slaves can share the same SPI bus, only the slave with its CS line asserted can participate in communication at any given time.

Figure 2.13 illustrates a configuration in which three slave devices are connected to a single SPI master. In this setup, the SCLK, MOSI, and MISO lines are shared among all slaves, while each slave has an individual CS line. The master uses these CS signals to select one device at a time, ensuring that only the selected slave can read from or write to the bus.

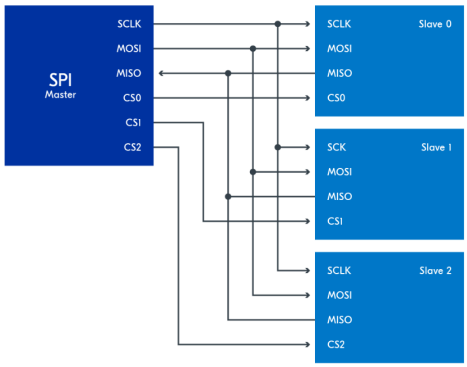


Figure 2.13: SPI configuration: 3 slave devices connected to single master

The SPI slave device utilizes separate transmit (TXD) and receive (RXD) buffers, which are typically double-buffered to allow continuous data flow in both directions without interruption. Additionally, the SPI master does not inherently manage the chip select (CS) signals; therefore, the system CPU must control the appropriate GPIOs to select and manage each slave independently of the SPI module.

SPI communication supports four modes, labeled 0 through 3, which are determined by the clock polarity (CPOL) and clock phase (CPHA). The clock polarity indicates the idle state of the clock signal when the CS line is active: a low level (CPOL = 0) or a high level (CPOL = 1). The clock phase specifies at which edge of the clock the data bits are sampled. For CPOL = 0, CPHA = 0 corresponds to sampling on the rising edge (low-to-high transition), while CPHA = 1 samples on the falling edge (high-to-low transition). For CPOL = 1, CPHA = 0 samples on the falling edge, and CPHA = 1 samples on the rising edge.

2.4 Bladder Control: from an open-loop system versus a closed-loop system

Currently approved neuroprosthetic devices operate using a continuous, open-loop stimulation paradigm, in which the delivered electrical signals do not take into account the current state of bladder filling. This approach may lead to neural adaptation, resulting in a gradual reduction in voiding effectiveness. To address this limitation, researchers are investigating conditional, closed-loop stimulation strategies, comparing their outcomes with open-loop systems in both animal models and human studies. These investigations highlight the potential benefits of transitioning from neuroprostheses with fixed activation patterns to devices that

provide stimulation on an as-needed basis [2].

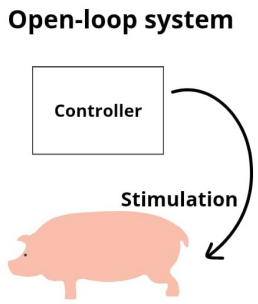


Figure 2.14: Open-loop system

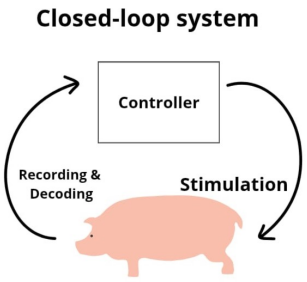


Figure 2.15: Closed-loop system

An illustrative example of the advantages of closed-loop systems can be found in recent research on the restoration of upper limb function through brain–machine interfaces (BMIs). In this study, a noninvasive, real-time magnetoencephalographic (MEG)-based BMI was developed to control a prosthetic hand. The system was able to detect movement onset and classify the type of hand movement (grasping or opening) in six subjects. The decoded information was then used online to trigger the prosthetic hand's movements at the intended time. This experiment demonstrates how closed-loop paradigms, by integrating neural decoding and real-time feedback, can transform neuroprosthetic control into a more natural and adaptive process [4].

Closed-loop systems can be found also in systems for the treatment of psychiatric disorders.
Psychiatric disorders are particularly challenging to treat due to their heterogeneity and fluctuating symptom profiles. Since many conditions are associated with abnormal neural activity that normalizes after effective treatment, neurostimulation therapies offer a promising way to both target pathological activity and monitor treatment effects. Neurostimulation can be delivered invasively or non-invasively through techniques such as TMS, tES, tFUS, VNS, SCS, and DBS, traditionally using open-loop paradigms with fixed stimulation parameters. However, the episodic and dynamic nature of psychiatric symptoms makes closed-loop approaches more suitable, as they can adjust stimulation in real time based on neural biomarkers or symptom states.
In a closed-loop system, algorithms act as a biomarker decoder (detecting neural signatures linked to symptoms) and a biomarker actuator (deciding when, where, and

how stimulation should be delivered). This paradigm has the potential to provide more effective, personalized, and state-targeted therapies, with early applications in conditions such as major depressive disorder (MDD) and obsessive-compulsive disorder (OCD). [5]

Closed-loop paradigms have also been applied to cardiovascular diseases (CVDs), particularly via vagus nerve stimulation (VNS). Open-loop VNS delivers pre-set stimulation without accounting for dynamic changes in cardiovascular biomarkers, while closed-loop VNS adapts stimulation in real-time based on neural or physiological signals, such as heart rate, heart rate variability, or baroreflex sensitivity. By decoding neural signals traveling through the vagus nerve, closed-loop systems can provide temporally and spatially precise stimulation, maintaining efficacy despite the plasticity of the cardiac neuraxis, which is often altered in chronic CVDs. Preclinical studies are exploring advanced closed-loop VNS protocols using neural decoding strategies to optimize therapy [6].

2.5 Aim of the Study

The primary objective of this thesis is the development of the firmware and software components required to support a closed-loop system for the lower urinary tract. A central element of this work is the design and implementation of a Graphical User Interface (GUI), which enables direct interaction between the user and the system. In addition to the user interface, the firmware has been designed to perform the following core functionalities:

- **Recording**: acquisition of neural signals from the pudendal nerve;
- **Electrodes Impedance check**: measurement of the electrodes impedance to ensure reliable recordings and stimulation.

Chapter 3

Description of the system

This chapter provides an overview of the hardware and software components used in the implementation of the system.

3.1 Setup

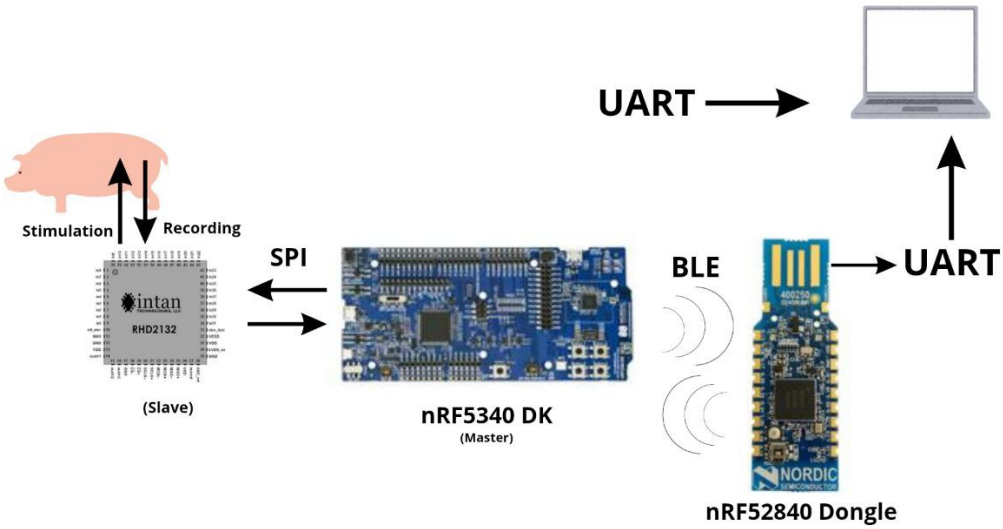


Figure 3.1: Setup

As shown in Figure 3.1, the `nRF5340` acts as the master device and controls the `RHD2132` slave, enabling signal recording. Stimulation cannot be performed using

the RHD2132 Intan chip, but requires the RHS2116 Intan chip. When the recording task is selected, neural signals acquired from the pudendal nerve are transmitted to the PC via UART. The decoding stage, on the other hand, is handled through Bluetooth Low Energy (BLE) communication, supported by the nRF52840 Dongle.

3.1.1 nRF5340 Development Kit

The nRF5340 DK, produced by Nordic Semiconductor, is the main development board used in this work. It integrates the nRF5340 SoC, which consists of two Arm Cortex-M33 processors:

- an application processor running at 128 MHz, with 1 MB of Flash and 512 kB of RAM;
- a network processor running at 64 MHz, with 256 kB of Flash and 64 kB of RAM.

The DK supports multiple wired and wireless communication interfaces, including SPI, I2C, UART, PWM, and Bluetooth Low Energy (BLE). The presence of a SEGGER J-Link debugger facilitates programming and real-time debugging, which was essential for firmware development. Additional peripherals, such as user-programmable buttons and LEDs, were used for rapid testing during the implementation phase.

3.1.2 nRF52840 Dongle

The nRF52840 Dongle is a low-cost USB device based on the nRF52840 SoC. It provides BLE connectivity between the development kit and the PC, supporting both data transmission and control commands. The SoC features a 32-bit Arm Cortex-M4 processor with floating-point unit, operating at 64 MHz, with 1 MB of Flash memory and 256 kB of RAM.

Within the closed-loop system developed in this work, the dongle served as the BLE interface during the decoding stage. Specifically, it enabled bidirectional communication between the PC and the microcontroller: the PC could send *START* and *STOP* commands to trigger the decoding algorithm, while the microcontroller returned the classification results back to the PC through the same interface.

3.1.3 RHD2132

The RHD2132 microchip, developed by Intan Technologies, is a bidirectional electrophysiology interface designed for neural signal recording. It provides 32 independent

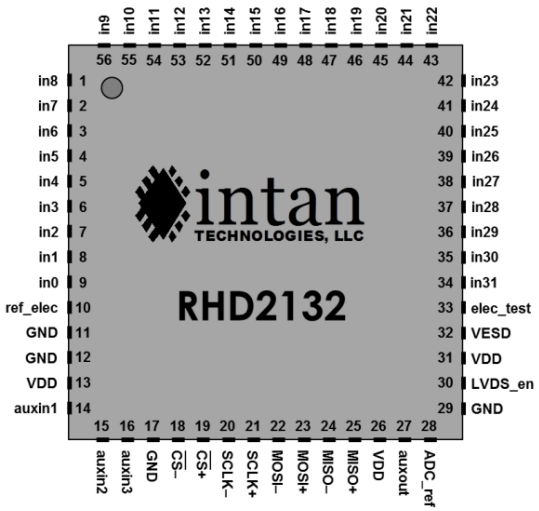


Figure 3.2: RHD2132 chip

recording channels, each featuring two low-noise amplifiers with programmable bandwidth. Sixteen channels are referenced to ground.

The chip architecture integrates low-noise amplifiers, analog and digital filtering stages, a 16-bit multiplexed analog-to-digital converter (ADC), and an electrode impedance measurement module. This integrated design allows the direct connection of electrodes to the chip, simplifying the overall system architecture.

The ADC supports sampling rates of up to 30 kSamples/s per channel, ensuring high-resolution acquisition of electrophysiological signals.

At the core of the RHD2000 series is an array of low-noise amplifiers with configurable analog filters, which can be tuned to isolate frequency bands of interest and reduce aliasing by attenuating frequency components above the Nyquist rate, defined as half of the per-channel ADC sampling rate. Each amplifier exhibits a passband extending from a low-frequency cutoff f_L to a high-frequency cutoff f_H.

Thanks to its compact footprint (8 mm × 8 mm QFN package) and low power consumption, the `RHD2132` is well suited for portable and miniaturized electrophysiological systems.

In this work, the `RHD2132` was employed as the recording interface of the proposed system, enabling the acquisition of neural signals from the pudendal nerve.

A simplified block diagram of the chip is shown in Figure 3.3, highlighting the main functional units, including the amplifiers, the 16 bit ADC and the SPI interface.

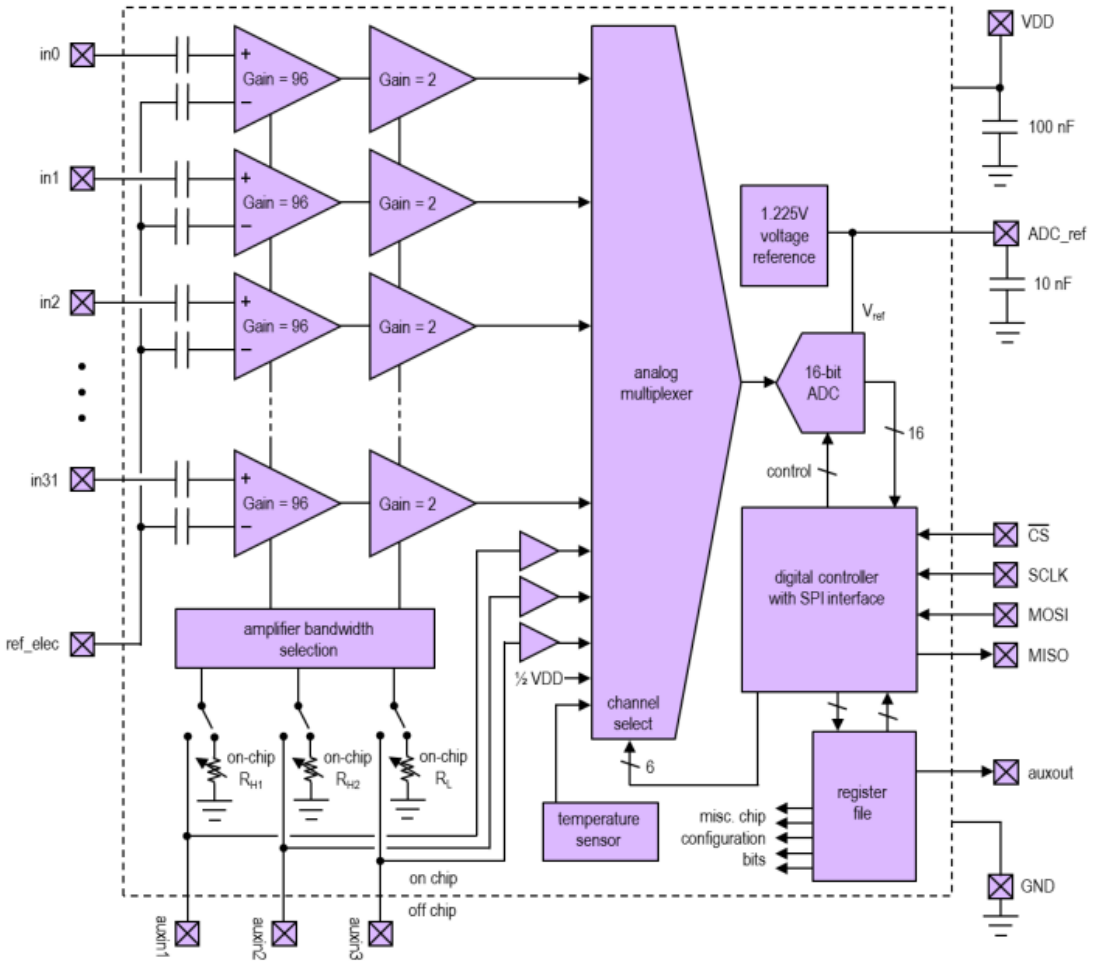


Figure 3.3: RHD2132 Simplified Diagram

SPI Communication with RHD2132

The `RHD2132` communicates with external devices through a standard Serial Peripheral Interface (SPI) bus. The chip always acts as the *slave*, while the microcontroller acts as the *master*. The interface consists of four signals:

- **CS (Chip Select)**: active-low signal used to enable communication with the chip.

- **SCLK (Serial Clock)**: clock generated by the master, with idle state low (CPOL=0) and data sampled on the rising edge (CPHA=0).
- **MOSI (Master Out, Slave In)**: line used by the master to send commands to the chip.
- **MISO (Master In, Slave Out)**: line used by the chip to return data to the master.

Communication occurs in 16-bit data words, transmitted *MSB first*. During each chip select cycle, the master sends a 16-bit command through MOSI, while simultaneously receiving the response to the *previous* command on MISO (pipelined communication). To ensure proper operation, the `CS` line must return high between successive 16-bit transfers.

This protocol allows efficient, synchronized bidirectional communication, and is critical for configuring registers, reading digitized neural signals, and controlling stimulation commands in real time.

SPI Command Words

Communication with the `RHD2132` device is structured around 16-bit *command words* sent by the master, in this case the nRF5340 DK, via the MOSI line. Each command instructs the chip to execute a specific operation and generates a 16-bit response, which is returned over MISO after two communication cycles due to the pipelined nature of the protocol.

The primary command types include:

- **CONVERT(C):** Initiates an analog-to-digital conversion (ADC) on a designated channel.
- **CLEAR:** Resets on-chip calibration parameters previously set by the `CALIBRATE` command.
- **CALIBRATE:** Starts the ADC self-calibration routine, which is typically performed after power-up and configuration of the chip registers.
- **WRITE(R, D):** Writes an 8-bit data value D into the specified register R. The lower byte of the response echoes back the written data to confirm correct reception, while the upper byte is filled with ones.
- **READ(R):** Retrieves the contents of register R. The data byte is returned in the lower byte of the result, with the upper byte consisting of zeros.

Command words may also include additional single-bit flags to modify the chip's behavior, such as enabling or disabling specific amplifiers or selecting particular sampling modes. This pipelined approach allows for continuous data acquisition, as the results of prior conversions are streamed back while new commands are being executed, ensuring efficient utilization of the SPI bus.

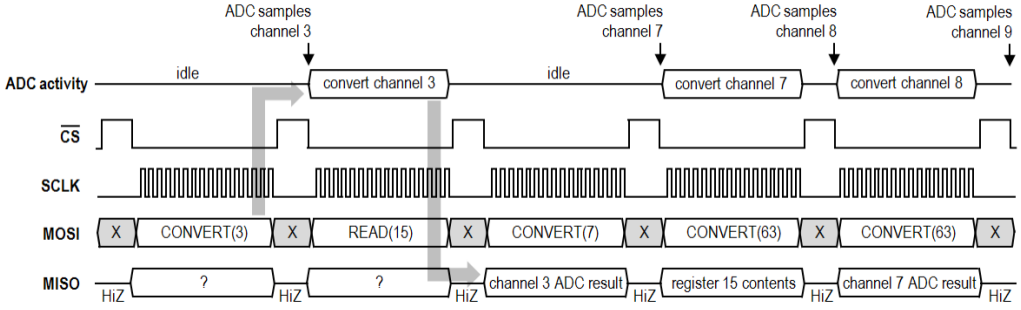


Figure 3.4: Timing diagram (CONVERT command)

3.2 Software Tools

This section presents the software tools employed during the development of the system. Specialized IDEs were used for coding and debugging on the Nordic Semiconductor platforms, while Python tools were adopted for implementing the graphical user interface that interacts with the hardware.

3.2.1 SEGGER Embedded Studio

SEGGER Embedded Studio (SES) is an Integrated Development Environment (IDE) utilized for writing, compiling, and debugging software on the Nordic Semiconductor boards used in this project. The IDE provides extensive support for ARM Cortex-based devices, making it fully compatible with both the nRF5340 DK and the nRF52840 Dongle.

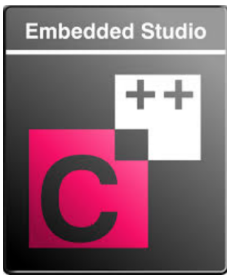


Figure 3.5: Segger embedded studio

3.2.2 nRF Connect for Desktop

nRF Connect for Desktop is a cross-platform software suite intended to support the development of applications for nRF devices. It encompasses a range of tools for tasks such as testing, monitoring, measuring, optimizing, and programming. In this work, the software was used to flash the Dongle with firmware implementing the BLE protocol.

Figure 3.6: nRF Connect for Desktop

3.2.3 Pycharm

PyCharm is an Integrated Development Environment (IDE) tailored for Python programming.
It offers advanced features such as code analysis, integrated debugging, version control support, and user interface design tools.
In this thesis, PyCharm was used to implement the Graphical User Interface (GUI), which allows the user to interact with the system by selecting among the available functions (Recording, Stimulation, Impedance Check, and Data Classification).

Figure 3.7: Pycharm

Chapter 4

Development and Testing of the code

In this chapter, both the Firmware and the Software developed will be explained. The firmware is flashed on the nrf5340 dk micro-controller.

4.1 Background

The firmware developed in this work is organized around a multithreaded architecture provided by the Zephyr RTOS. The central component is the `user_interface()` function, which runs in the *main thread* with the highest priority and manages the interaction between the user and the Bladder System Control.

Through the interface, the user can select among four main operations:

- **Recording**: neural signals are acquired from the RHD2132 chip. The user selects the channels (from channel 8 to channel 23), the lower and upper cutoff frequencies of the amplifiers, and the destination of the recorded data (PC or SD card). Once configured, the control is handed over to the `spi_control_thread()`, which manages SPI communication and to the `file_write_thread()`, which manages the data flow through UART.
- **Impedance Check**: the user specifies the channel, the DAC waveform frequency and the series capacitor. The system then executes impedance measurements using the dedicated module of the RHD2132.
- **Stimulation**: the final project includes the stimulation task, however, it is not within the scope of this thesis. The Intan chip RHD2132 does not include the hardware needed for the stimulation. The final project will replace

the RHD2132 chip with the RHS2116 chip that allows to implement the stimulation task.

- **Data Analysis**: the user can start the decoding stage, where the recorded neural signals are processed by a machine learning algorithm to extract the bladder state (empty, full, micturition) information.

The main menu window of the GUI is shown in Figure 4.1.

Figure 4.1: GUI: main menu

4.2 Recording

The `Recording` task is responsible for acquiring the neural signals from the RHD2132 microchip. It involves transmitting `CONVERT` commands through the SPI interface and subsequently sending the acquired samples to the host PC via UART communication. The overall operational flow of the recording process is depicted in Figure 4.2.

Selection of the Number of Channels

The user can select the number of channels to be recorded through the graphical user interface (Figure 4.3). A maximum of four channels can be acquired simultaneously. This limitation arises from maximum available UART baud-rate, as discussed in detail in the following sections. Restricting the number of active channels ensures

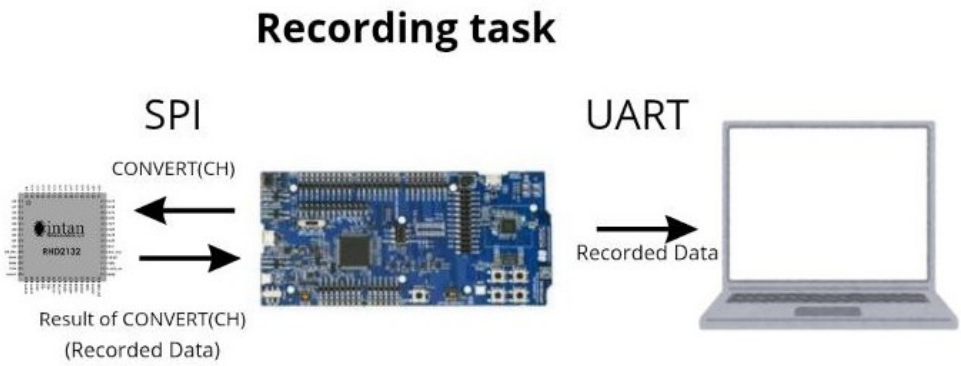


Figure 4.2: Schematic of the Recording task

that no conversion results are lost while maintaining the target sampling frequency of 10 kHz per channel.

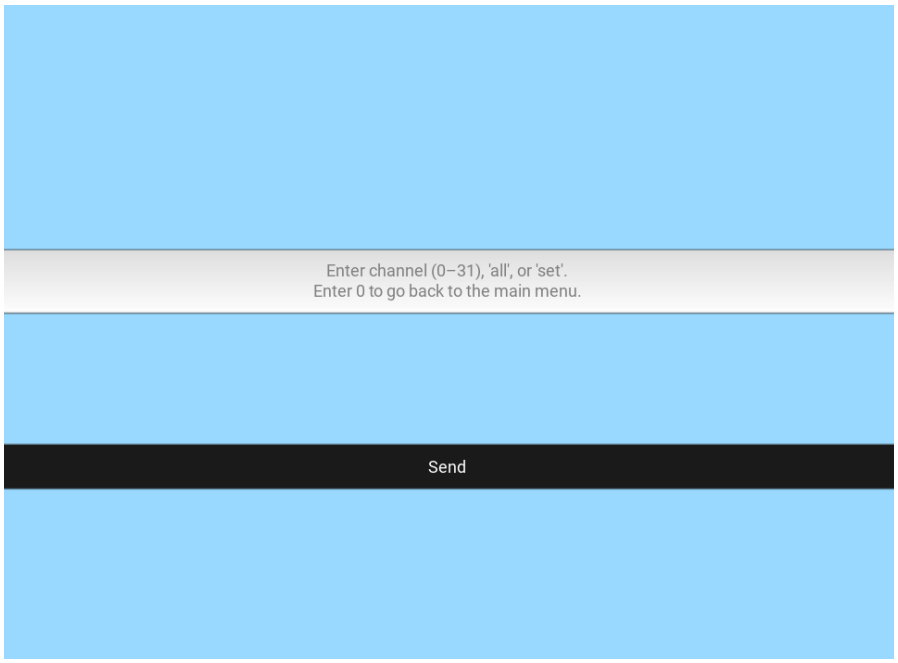


Figure 4.3: GUI: Channels selection

Selection of Frequency Bandwidth

The user can also configure the lower and upper cutoff frequencies of the amplifiers (Figures 4.4 and 4.5). Electrophysiological signals are sensed by AC-coupled low-noise amplifiers integrated in the `RHD2132`, each of which includes programmable analog filters that define a passband from a low-frequency cutoff f_L to a high-frequency cutoff f_H. This configuration allows to get two main purposes:

- Isolation of the neural signals of interest.
- Minimization of aliasing by attenuating spectral components above the Nyquist rate (half the ADC sampling frequency).

Figure 4.4: GUI: low frequency selection

Selection of Data Destination

The recorded data can be stored in two possible destinations (Figure 4.6):

- On an SD card, for local storage.
- On a PC, transmitted in real time through UART.

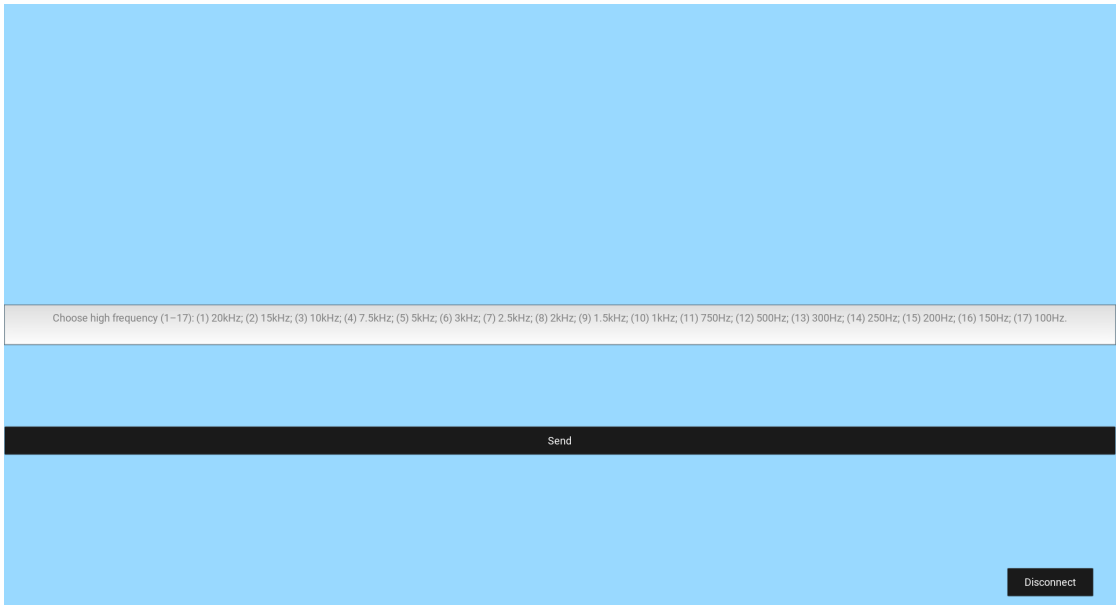


Figure 4.5: GUI: high frequency selection

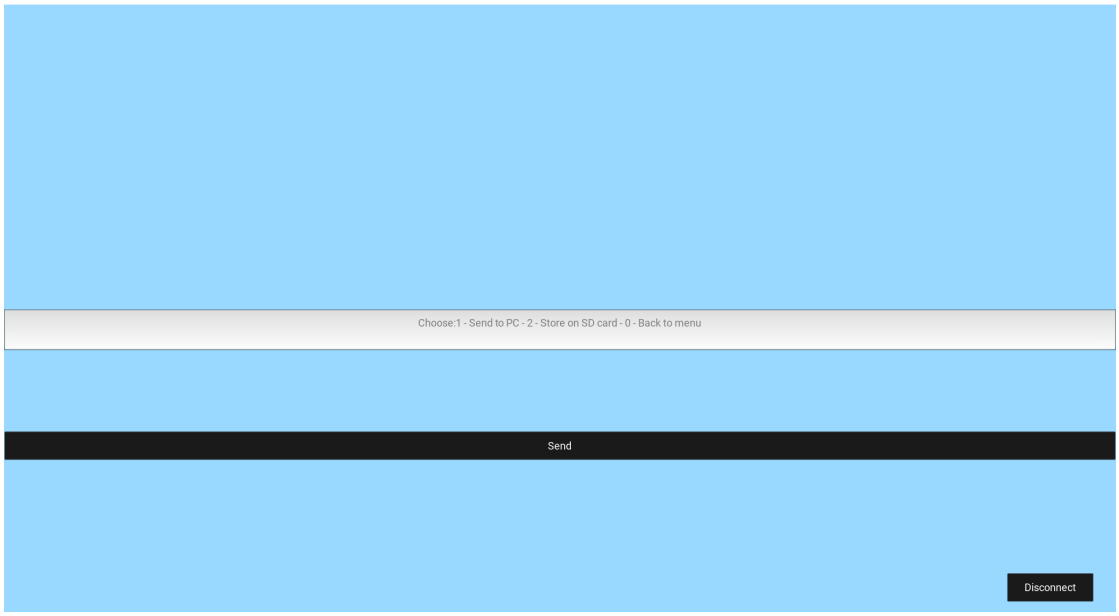


Figure 4.6: GUI: Data destination

4.2.1 SPI communication

The Serial Peripheral Interface (SPI) manages the communication between the nRF5340 microcontroller and the Intan chip RHD2132. Once all recording parameters have been configured, the `spi_control_thread()` function is activated. This thread handles the SPI communication for the Recording task. The ENG signals acquisition is performed by transmitting `CONVERT` commands to the RHD2132, that executes an analog-to-digital conversion of the selected channel, and sending the corresponding conversion results to the PC via UART.

Before the SPI communication is established, a series of initialization steps are required:

- **Initialization of the CONVERT command array:** definition of an array containing the sequence of `CONVERT(CH)` commands.

- **Definition of the sampling frequency:** through the `ppi_configuration()` function, the acquisition timing is set by linking a timer event to the SPI task.

After these preparatory steps, the `spi_control_thread()` function is executed. This thread coordinates the SPI transmission cycle, ensuring continuous data acquisition and transfer to the Graphical User Interface (GUI).

CONVERT Command Initialization

The `CONVERT` command array defines the sequence of conversion operations transmitted to the `RHD2132`, with a maximum length of 128 instructions. When all 16 channels are enabled for acquisition, the initialization routine generates the following command sequence:

```
CONVERT(8)
CONVERT(9)
CONVERT(10)
...
CONVERT(23)
```

The response of a `CONVERT(`CH_i`)` command is available at the command `CONVERT(`CH_{i+2}`)`. To avoid losing any data, additional dummy commands are inserted, as shown in the Table 4.1:

N	COMMAND
0	CONVERT(8)
1	CONVERT(9)
2	CONVERT(10)
3	CONVERT(11)
4	CONVERT(12)
...	...
15	CONVERT(23)
16	CONVERT(8)
17	CONVERT(9)
...	...
124	READ(40) (dummy)
125	READ(41) (dummy)
126	READ(42) (dummy)
127	READ(43) (dummy)

Table 4.1: CONVERT command array

On the other hand, the lenght of Result CONVERT command array is 2N = 256, as shown in Fig.4.9. In Table 4.1, an example of the CONVERT command array initialization is shown for the case in which all 16 channels are enabled. However, the maximum number of channels that can be effectively recorded is constrained by the **UART baud-rate**. The maximum available baud rate is 921600 bit/s. Given a sampling frequency of 10 kHz per channel and a 16-bit ADC resolution, the resulting data rate limits the maximum number of recordable channels to four. Moreover, the spim transmission register and the spim receiving register are also initialized.

```
1  void spim_convert_trx_setup()
2  {
3      spim_inst.p_reg->TXD.PTR = (uint32_t)convert_commands[0];
4      spim_inst.p_reg->RXD.PTR = (uint32_t)spim_trx_result_conv[0];
5      spim_inst.p_reg->TXD.LIST = 1;
6      spim_inst.p_reg->RXD.LIST = 1;
7      spim_inst.p_reg->TXD.MAXCNT = BUFFER_SIZE;
8      spim_inst.p_reg->RXD.MAXCNT = BUFFER_SIZE;
9
10     nrfx_spim_xfer_desc_t xfer_desc =
11         NRFX_SPIM_XFER_TRX(convert_commands[0], BUFFER_SIZE,
12                            spim_trx_result_conv[0], BUFFER_SIZE);
13
14     uint32_t spim_flags = NRFX_SPIM_FLAG_TX_POSTINC
15                         | NRFX_SPIM_FLAG_RX_POSTINC
16                         | NRFX_SPIM_FLAG_NO_XFER_EVT_HANDLER
```

```
17                              | NRFX_SPIM_FLAG_REPEATED_XFER
18                              | NRFX_SPIM_FLAG_HOLD_XFER;
19
20      nrfx_spim_xfer(&spim_inst, &xfer_desc, spim_flags);
21  }
```

The SPI transfer is configured using several flags that define how the transmission and reception buffers are managed and how the transfer is executed:

- **TX_POSTINC / RX_POSTINC**: Flag indicating that TX-RX buffer address will be incremented after each transfer.
- **NO_XFER_EVT_HANDLER**: Flag indicating that the interrupt after each transfer will be suppressed, and the event handler will not be called.
- **REPEATED_XFER**: Flag indicating that the transfer will be executed multiple times.
- **HOLD_XFER**: Flag indicating that the transfer will be set up, but not started.

PPI configuration

To ensure a precise sampling frequency, the firmware exploits the PPI (Programmable Peripheral Interconnect) system of the nRF5340. The PPI allows the direct connection of hardware events (e.g., a timer compare event) to hardware tasks (e.g., start an SPI communication), without the intervention of the CPU. The configuration proceeds as follows:

1. Two PPI channels are allocated.
2. The first channel connects the *timer compare event* (Timer0, CC0) to the `SPIM START` task. In this way, every timer expiration triggers an SPI transmission.
3. The same channel is also "forked" to increment the first hardware counter.
4. The second channel connects a compare event on the first counter to the task `COUNT` of a second counter. Every time the first counter reaches the target value N=128, the second counter is incremented.
5. The second counter target value is M=2.
6. Finally, the configured PPI channels are enabled.

Thanks to this configuration, the entire acquisition chain runs in hardware:

- When the timer expires, a new SPI transfer is triggered and the first counter is incremented.
- After N `SPIM TASKs`, the second counter is incremented and the index of the CONVERT commands array is addressed to 0. In this way the `SPIM TASK` will be a circular transfer.
- After 2N `SPIM TASKs`, the second counter reaches M=2 and the index of the Results of the CONVERT commands are addressed to 0.

```
1     void ppi_configuration()
2 {
3   nrfx_err_t err_code;
4   uint8_t ppi_channel1,ppi_channel2;
5
6   err_code = nrfx_gppi_channel_alloc(&ppi_channel1);
7     if (err_code != NRFX_SUCCESS)
8     {
9         //printk("PPI channel allocation error for channel 1");
10    }
11
12    err_code = nrfx_gppi_channel_alloc(&ppi_channel2);
13    if (err_code != NRFX_SUCCESS)
14    {
15        //printk("PPI channel allocation error for channel 2");
16    }
17
18    uint32_t timer_event_addr =
     nrfx_timer_compare_event_address_get(&timer,
     NRF_TIMER_CC_CHANNEL0);
19    spim_start_task_addr = nrf_spim_task_address_get(spim_inst.
     p_reg, NRF_SPIM_TASK_START);
20
21  nrfx_gppi_channel_endpoints_setup(ppi_channel1, timer_event_addr
     , spim_start_task_addr);
22
23  spim_end_event_addr = nrf_spim_event_address_get(spim_inst.p_reg
     , NRF_SPIM_EVENT_END);
24  uint32_t counter_task_addr = nrfx_timer_task_address_get(&
     counter, NRF_TIMER_TASK_COUNT);
25
26  nrfx_gppi_fork_endpoint_setup(ppi_channel1, counter_task_addr);
27
28  uint32_t counter_event_addr =
     nrfx_timer_compare_event_address_get(&counter,
     NRF_TIMER_CC_CHANNEL4);
29  uint32_t counter_seq_task_addr = nrfx_timer_task_address_get(&
     counter_seq, NRF_TIMER_TASK_COUNT);
```

```
30
31    nrfx_gppi_channel_endpoints_setup(ppi_channel2,
        counter_event_addr, counter_seq_task_addr);
32
33    nrfx_gppi_channels_enable(1 << ppi_channel1 | 1 << ppi_channel2)
        ;
34  }
```

The logic flow is shown in Figure 4.7.

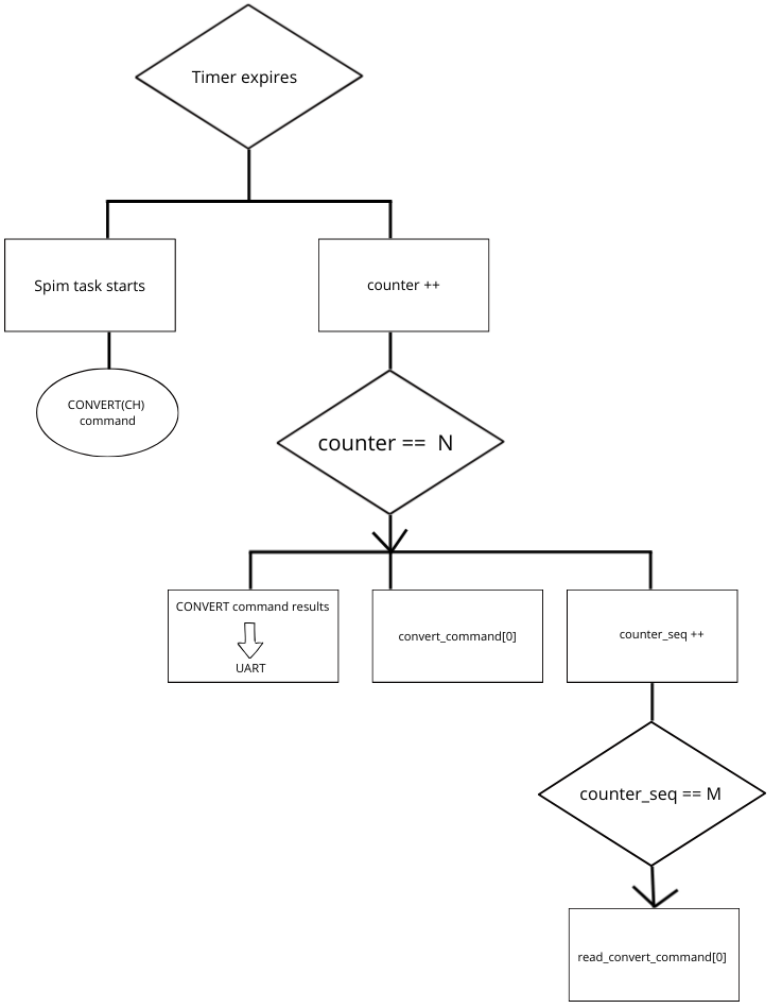


Figure 4.7: PPI configuration

SPI Control Thread

Once the user selects `START`, the `spi_control_thread()` takes control and the SPIM task begins, following the timing configuration described in the `PPI configuration` section. The SPIM task continuously handles the transmission of `CONVERT` commands and can be stopped by the user at any time.

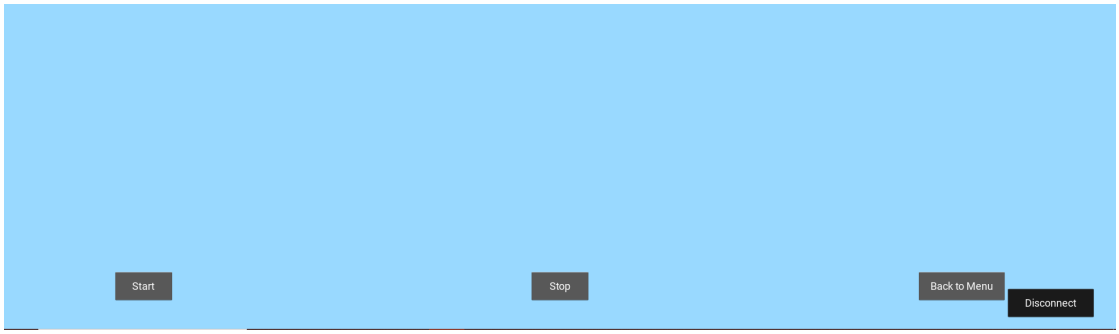


Figure 4.8: GUI: START/STOP

4.2.2 UART Communication

During ENG signal acquisition, the following sequence takes place:

- Every time the timer expires, a new `CONVERT` command to the `RHD2132` is triggered, while the first counter is incremented.
- Every time the first counter reaches $N = 128$, control is handed over to the `file_write_thread()`. This thread handles the UART transmission of the converted data.

File Write Thread

If the user has selected the PC as the storage destination, the results of the `CONVERT` commands are transmitted via UART. The transmission is optimized by exploiting a *double-buffering* scheme:

- When the first counter reaches $N = 128$, the first half of the UART buffer (Result CONVERT command) is transmitted while the second half is being filled.
- When the first counter again reaches $N = 128$, the second half of the buffer (Result CONVERT command) is transmitted while the first half is being refilled.

This mechanism ensures continuous data streaming without loss, while minimizing CPU load. The available UART baud rate is 921600 bit/sec. Each CONVERT result is on 16 bit. Keeping a frequency of 10 kHz/channel, the maximum number of channels that can be recorded is:

$$\frac{N \times 16\ bit}{N \times \frac{100\ \mu s}{Number\ of\ channels}} < 921600\ \frac{bit}{s} \qquad \Rightarrow \qquad Number\ of\ channels < 5.76 \tag{4.1}$$

The maximum number of channels that can be recorded simultaneously is four.

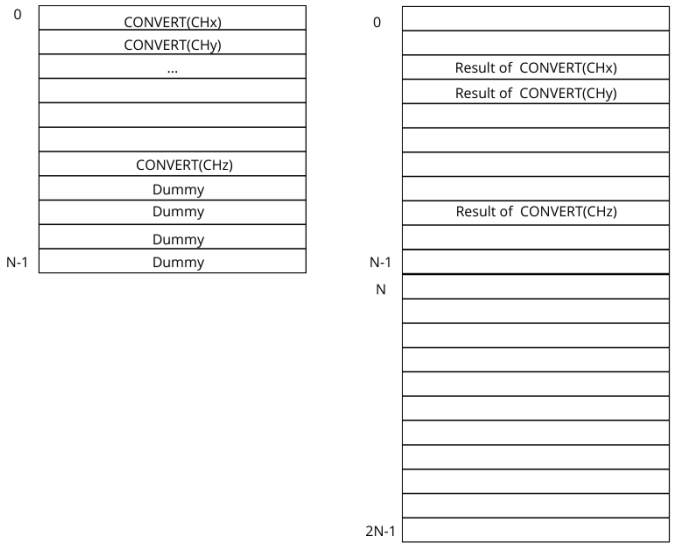


Figure 4.9: CONVERT command array and Result CONVERT command array (UART buffer)

4.3 Impedance Check

Electrode impedance measurement is a fundamental step in neural recording systems, as it allows verification of electrode quality prior to the signal acquisition process. The `RHD2132` chip integrates a dedicated module for performing impedance tests, shown in Fig. 4.10, thus eliminating the need for external measurement hardware.

4.3.1 On-chip Impedance Measurement Principle

Each amplifier input can be individually connected to the on-chip current generator through a set of transistor switches (S0–S31). Under normal operation, when the `Zcheck_en` parameter is set to zero, all switches remain open. When impedance measurement mode is enabled (`Zcheck_en` = 1), the switch corresponding to the electrode selected in the `Zcheck_select` register is closed, connecting the selected electrode to the on-chip current generator, as illustrated in Fig. 4.10. An AC current flows through the selected electrode, producing a voltage drop across it.

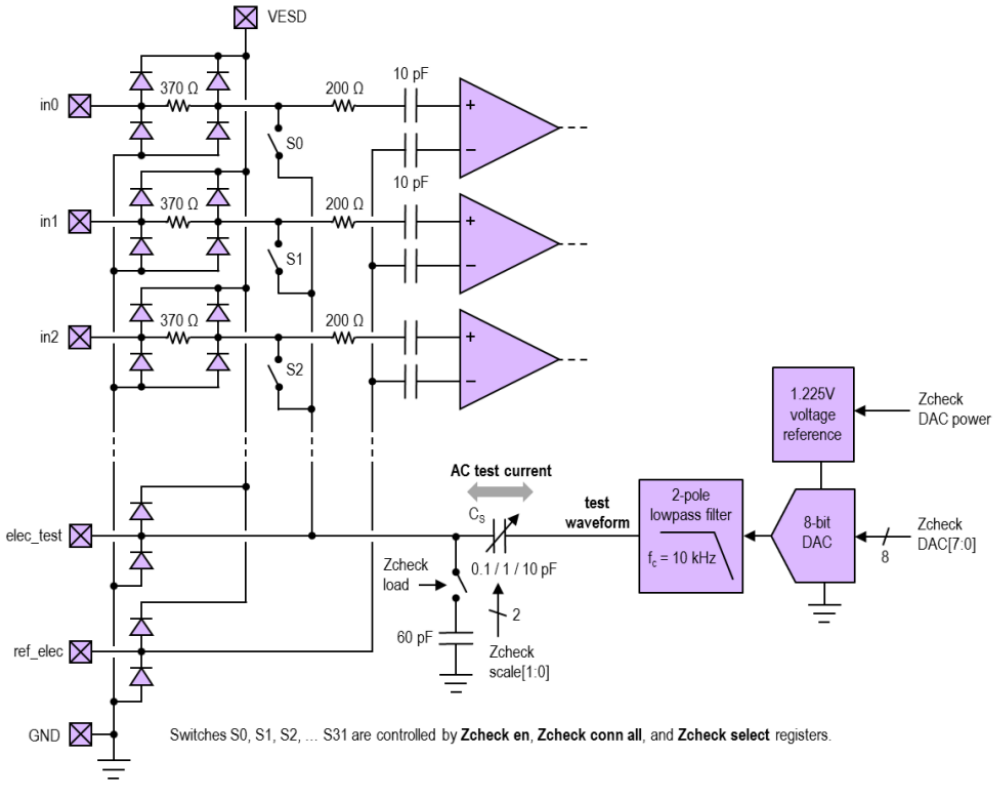


Figure 4.10: Impedance module

The magnitude of the electrode impedance is computed as:

$$|Z| = \frac{V_{peak}}{I_{peak}} \tag{4.2}$$

where V_{peak} and I_{peak} denote the peak amplitude of the measured voltage and the injected current, respectively.

4.3.2 AC Current Generation

The AC current is generated internally by the Intan chip by enabling the appropriate impedance measurement module. This module consists of a programmable 8-bit digital-to-analog converter (DAC), followed by a two-pole 10 kHz low-pass filter that smooths the DAC output waveform, and a series capacitor C_S.
The voltage waveform produced by the DAC, denoted as $v_{\text{DAC}}(t)$, is converted into a current through the series capacitor C_S according to:

$$i_{DAC}(t) = C_S \frac{dv_{DAC}(t)}{dt} \tag{4.3}$$

The value of C_S is programmable and can be set to 0.1 pF, 1 pF, or 10 pF. The DAC is updated periodically to generate a continuous sinusoidal waveform with amplitude V_A and frequency f, resulting in an injected current given by:

$$i_{DAC}(t) = 2\pi f C_S V_A \cos(2\pi f t) \tag{4.4}$$

4.3.3 Firmware Implementation

From a firmware perspective, enabling the impedance measurement module requires configuring three registers:

- Register 5: Impedance Check Control (Fig. 4.11);
- Register 6: Impedance Check DAC (Fig. 4.12);
- Register 7: Impedance Check Amplifier Select (Fig. 4.13).

Register 5: Impedance Check Control

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 5	X	Zcheck DAC power	Zcheck load	Zcheck scale [1:0]		Zcheck conn all	Zcheck sel pol	Zcheck en

Figure 4.11: Register 5 - Impedance Check Control

Register 6: Impedance Check DAC

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 6	Zcheck DAC [7:0]							

Figure 4.12: Register 6 - Impedance Check DAC

Register 7: Impedance Check Amplifier Select

bit	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Register 7	X	X	Zcheck select [5:0]					

Figure 4.13: Register 7 - Impedance Check Amplifier Select

Registers Configuration

Registers 5 and 7 are configured during the initialization phase, whereas Register 6 must be updated periodically to generate the AC current waveform.
The Register 5 is configured as follows:

- `Zcheck_DAC_power`: set to 1 to activate the on-chip digital-to-analog converter (DAC) used to generate waveforms for electrode impedance measurement.
- `Zcheck_scale`: selects the series capacitor value (C_s) among 0.1 pF, 1.0 pF, or 10 pF.
- `Zcheck_en`: set to 1 to enable impedance testing mode.

The Register 7 is configured as follows:

- `Zcheck_select`: selects the channel (electrode) to be connected to the impedance test circuit.

Register 6 is updated periodically by setting the `Zcheck_DAC` field, which defines the DAC output voltage. To this end, the array `spi_commands` of length $Z = 48$ is initialized, containing a sequence of different `Zcheck_DAC` values interleaved with `CONVERT` commands and Dummy commands.

```
1  void Z_CHECK_DAC_INIT(uint8_t C, bool H)
2  {
3    uint8_t REG_ZCheckDAC = 0x06;
4    uint8_t voltage_update;
5
6    // Single channel Test
7    for(int i = 0; i < Z; i=i+3)
8    {
9      voltage_update = voltage_calculation();
10
11     uint16_t command1 = ((C & 0x3F) << 8) | (H ? 0x01 : 0x00);  //
       convert command
12
13         ZcheckDAC_command[i][1] = command1 & 0xFF;         // LSB
14     ZcheckDAC_command[i][0] = (command1 >> 8) & 0xFF; // MSB
15
16     uint16_t command2 = (0x02 << 14) | ((REG_ZCheckDAC & 0x3F) <<
      8) | voltage_update;  // write the reg 6
17
18         ZcheckDAC_command[i+1][1] = command2 & 0xFF;         // LSB
       // WRITE command in the Reg. 6
19     ZcheckDAC_command[i+1][0] = (command2 >> 8) & 0xFF; // MSB
20
21     ZcheckDAC_command[i+2][0] = 0xE8;  // I  // Dummy command
22       ZcheckDAC_command[i+2][1] = 0x00;
23
24   }
25
26 }
```

Waveform Generation and PPI Configuration

In order to periodically update Register 6, the same PPI configuration used for the data acquisition task is reused for the impedance measurement task. The PPI configuration has already been described in the section dedicated to the recording task; therefore, this section focuses on its application to impedance testing. The SPI transmission buffer has a length of Z, while the SPI reception buffer has a length of $2Z$, as shown in Fig.4.14.

- Each time the timer expires, the `SPIM_TASK` is triggered and one command from the SPI transmission buffer array is transmitted from the microcontroller to the Intan chip. This results in the execution of a `CONVERT` command followed by the updating the `Zcheck_DAC` value. At each timer expiration, the first counter is incremented.
- When the first counter reaches the target value $Z = 48$, the responses to the `CONVERT` commands are transmitted to the GUI. The index of the SPI transmission buffer is then reset to zero, enabling circular SPI transmission. Subsequently, the second counter is incremented.
- When the second counter reaches $2Z$, the SPI reception buffer index is reset to zero.

The logic flow of the impedance measurement process is illustrated in Fig. 4.15.

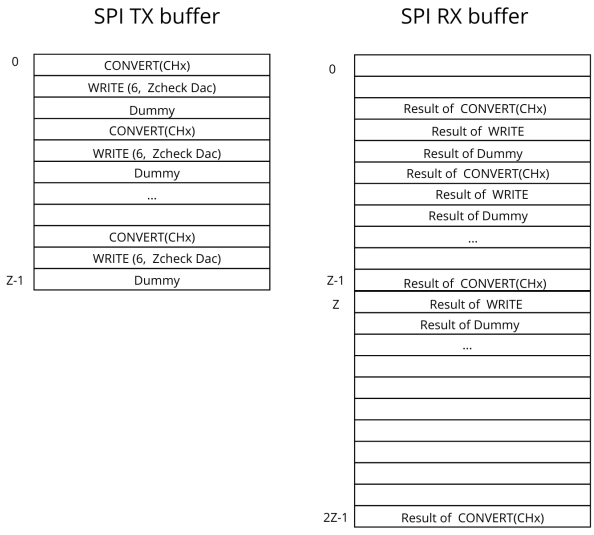


Figure 4.14: SPI buffers for Impedance Check

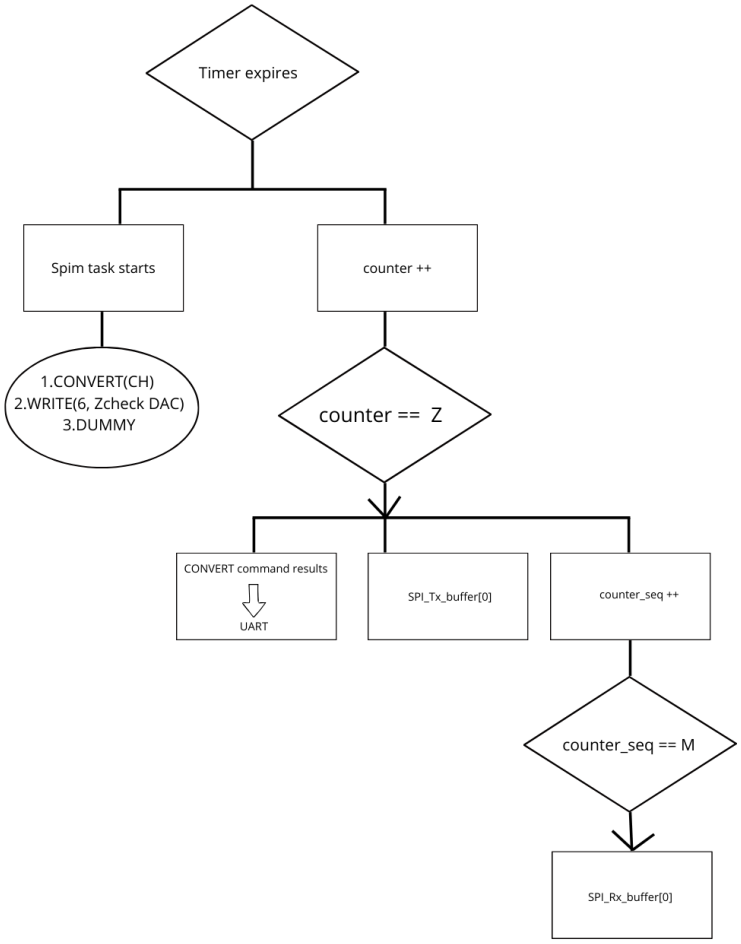


Figure 4.15: Impedance Check: Firmware Flow

Final Impedance Computation

The acquired data are transferred to the GUI via UART in real time. These data represent the voltage drop across the electrode and are therefore processed to compute the electrode impedance as:

$$Z = \frac{V_{peak}}{I_{peak}} \tag{4.5}$$

where V_{peak} is the peak amplitude of the measured voltage and I_{peak} is the peak amplitude of the injected DAC current. The peak DAC injected current is computed,

ideally, as:

$$I_{peak} = 2\pi f C_s V_A \tag{4.6}$$

This amount of DAC current depends on the test waveform frequency, the peak voltage amplitude, and the selected series capacitor. The DAC waveform frequency can be selected by the user.

4.4 BLE Firmware Implementation

The BLE (Bluetooth Low Energy) firmware was implemented to enable wireless communication between the `nRF5340` and the Graphical User Interface (GUI). This communication layer allows the user to interact with the entire Bladder Control system by configuring the desired parameters for both the recording and the impedance check procedures.

The firmware has been developed into three files: `main.c`, `cmdS.c`, and `cmdS.h`. The first one manages the initialization and the application logic, while the last two implement the Bluetooth GATT service, the associated characteristics, and their callback functions.

Custom GATT Service

A custom BLE GATT service, named `BladderControl`, has been defined. It contains several characteristics that allow the communication between the NRF5340 micro-controller and the Graphical User Interface (GUI).

Each service and characteristic is uniquely identified by a Universally Unique Identifier (UUID).
This identifier allows the central device (e.g., a the GUI) to recognize which data or command it is interacting with, even across different devices and implementations. The UUIDs have been defined in the header file `cmdS.h`, as shown in Table 4.2.

Characteristic	**UUID (128-bit)**
BladderControl Service	00001523-1212-efde-1523-785feabcd123
User Interface (Commands)	00001526-1212-efde-1523-785feabcd123

Table 4.2: Custom BLE service and characteristics UUIDs.

The **UI Characteristic** allows the user to send textual commands (e.g., `start`, `stop`, parameters for the Recoding task and for the Impedance check) from the Graphical User Interface (GUI).

Characteristic Callback

The characteristic is associated with a callback function that handle read, write, or notify operations. The callback is implemented in `cmdS.c`. The `received_cmd()` is the *write* callback for the UI characteristic, used to receive user commands.

The BladderControl service and its characteristics are defined as follows:

```
    BT_GATT_CHARACTERISTIC(BT_UUID_LBS_UI, BT_GATT_CHRC_WRITE,
           // BT_GATT_PERM_WRITE_ENCRYPT,
                 BT_GATT_PERM_WRITE, NULL, received_cmd, NULL),
  );
```

Write Callback: `received_cmd()`

The function `received_cmd()` manages textual commands received from the central device. Its workflow is summarized as follows:

- Incoming data is received character by character and stored in a local buffer.
- Once a termination character (`\n` or `\r`) is detected, the full command is pushed into a Zephyr message queue (`k_msgq_put`).
- A semaphore (`k_sem_give`) is released to notify the decoding thread that a new command has been received.

```
static ssize_t received_cmd(struct bt_conn *conn,
                            const struct bt_gatt_attr *attr,
                            const void *buf, uint16_t len,
                            uint16_t offset, uint8_t flags)
{
    const uint8_t *data = buf;

    for (int i = 0; i < len; i++) {
        char c = (char)data[i];
        //printk("Ricevuto: %c\n", c);

        if ((c == '\n' || c == '\r')) {
            if (rx_pos > 0) {
                // String termination
                rx_buf[rx_pos] = '\0';
                //printk("Messaggio completo ricevuto: %s.\n",
   rx_buf);

                char temp_buf[CMD_MAX_LEN];
                strncpy(temp_buf, rx_buf, CMD_MAX_LEN);
                temp_buf[CMD_MAX_LEN - 1] = '\0';
```

```
21
22                  if (k_msgq_put(&uart_msgq, temp_buf, K_NO_WAIT) !=
        0) {
23                      //printk("Errore: messaggio non inserito in
    coda\n");
24
25                  } else {
26                      msg_received = true;
27                      k_sem_give(&msg_received_sem);
28                  }
29                  rx_pos = 0;
30              }
31          } else if (rx_pos < sizeof(rx_buf) - 1) {
32              rx_buf[rx_pos++] = c;
33          } else {
34              //printk("Buffer overflow, reset.\n");
35              rx_pos = 0;
36          }
37      }
38
39      return len;
40  }
```

Listing 4.1: Callback for receiving BLE commands

This mechanism allows the BLE interface to behave like a remote UART, where textual commands control the system operation without direct physical connection.

Once the characteristic callback has been defined, it must be registered and linked to the main application logic.
This linkage is performed during the service initialization phase, which associates the application-level callback functions to the corresponding BLE events (i.e. command reception).
The initialization procedure is handled by the function `bt_lbs_init()`, described in the next section.

Service Initialization

The function `bt_lbs_init()` is responsible for initializing the callback function, `user_interface_cb`.

```
1  int bt_lbs_init(struct bt_lbs_cb *callbacks)
2  {
3    if (callbacks) {
4      lbs_cb.user_interface_cb = callbacks->user_interface_cb;
5    }
6    return 0;
7  }
```

Listing 4.2: Callback for receiving BLE commands

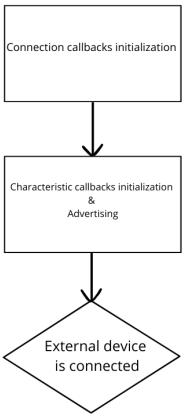


Figure 4.16: BLE initialization

Main Application

The `main.c` file represents the application layer of the firmware. It defines the behavior of the system during BLE connection and disconnection events (via the callbacks `on_connected()` and `on_disconnected()`), and initializes the `BladderControl` GATT service.

BLE Connection Management

The functions `on_connected()` and `on_disconnected()` manage the connection status between the `nRF5340` and the external device (e.g., the GUI). When a connection is established, the LED indicator is turned on, and the flag `is_connected` is set to `true`. On disconnection, the LED is turned off, and the flag is cleared.

User Command Handling

User commands are sent from the BLE central device (PC) to the microcontroller through the UI characteristic. When the central device writes data to this characteristic, the BLE stack invokes the callback function received_cmd() defined in

`cmdS.c`.
Inside `received_cmd()`, the received characters are collected in a buffer. Once a termination character (\n or \r) is detected, the full command is copied into a message queue.

4.5 Graphical User Interface (GUI)

The Graphical User Interface (GUI) has been developed in Python using the `kivy` library, which enables the creation of cross-platform applications running consistently on Windows, macOS, Linux, Android, and iOS. Communication between the GUI and the microcontroller (`nRF5340`) is achieved via Bluetooth Low Energy (BLE), using the `blatann` library to manage the BLE connection, service discovery, and data exchange.

The main purpose of the GUI is to provide an intuitive and user-friendly interface for controlling the overall system and monitoring its operation in real time. Specifically, the GUI allows the user to:

- Connect or disconnect from the BLE peripheral (the `nRF5340`).
- Send the desired commands from the user.
- Visualize the real-time acquired data through graphical plots.
- Visualize the real-time measured electrode impedance through graphical plots.

The GUI acts as the central control unit of the experimental setup, bridging the BLE firmware running on the nRF5340 DK and the user.

4.5.1 Software side

The python code is organized in two python files:

- `ble_manager.py`: implements all the Bluetooth Low Energy (BLE) communication routines, including device scanning, connection and disconnection management, data transmission (write).
- `main.py`: defines the graphical user interface (GUI) and manages the overall BLE application. The GUI runs on the main thread, ensuring a responsive user experience. To avoid blocking the interface during BLE operations, the BLE event loop is executed in a secondary thread, which handles asynchronous communication tasks independently.

Python: ble_manager.py

The `BLEManager` class encapsulates all Bluetooth Low Energy communication logic. It is responsible for scanning, connecting, sending commands, receiving notifications, and managing the connection state with the peripheral device.

The main methods are:

- **`scan_devices(self, timeout`**= 3**)**: Starts the scanning procedure for nearby BLE peripherals using the configured dongle. The function returns a list of available devices that can be connected.
- **`on_connect_dev(peer_address)`**: Establishes a BLE connection with the specified peripheral (the `nRF5340`). Once the connection is established, the function performs service discovery and enables the relevant GATT database.
- **`ble_send(data: bytes)`**: Sends user commands or control messages from the PC to the microcontroller via the BLE link.
- **`disconnect_dev()`**: Gracefully terminates the BLE connection.

Each method is built on top of the `blatann` library, which provides a high-level interface for BLE operations and event handling through asynchronous callbacks.

Python: main.py

This file defines two main classes: `GUI` and `BLEApp`.

The `GUI` class implements the graphical user interface using the widgets provided by the `kivy` library, such as buttons, labels, and layouts. It represents the front-end of the application, through which the user can interact with the microcontroller via BLE.

The main functionalities are organized as follows:

- **start_scanning**: initiates the scanning process by calling the `scan_devices()` function from the `BLEManager` class.
- **connect_to_device**: once the user selects the `BladderControl` peripheral, this function calls `on_connect_dev()` to establish the BLE connection.
- **button_press**: handles user actions related to `Recording`, `Stimulation`, `Impedance Check`, and `Data Classification` tasks.
- Each user command is sent to the microcontroller via BLE through the `ble_send()` function, as shown below:

```
1         self.ble_manager.ble_send("COMMAND")
```

Listing 4.3: Sending commands from GUI to micro-controller via BLE

If the user selects `Recording`, the function `start_recording_flow()` is triggered, guiding the user through the selection of recording parameters such as the number of channels, the bandwidth, and the destination path for the recorded data.

If the user selects `Impedance Check`, the function `start_imped_check_flow()` is executed, allowing the user to set parameters such as the sine wave frequency and the series capacitor value.

The `BLEApp` class represents the main application entry point. It creates and manages the asynchronous event loop used for BLE communication and runs the Kivy interface in the main thread, ensuring smooth user interaction while the BLE operations are handled in a separate thread.

4.5.2 Firmware side

The commands transmitted via BLE are managed by the firmware through the `user_interface()` function. This routine constitutes the core of the command-handling logic and is responsible for decoding the incoming messages from the BLE interface.
It continuously waits for new commands placed in the message queue, interprets them, and triggers the corresponding firmware operation, such as recording, impedance check, stimulation, or data classification.

```
1  void user_interface(){
2
3    while(1){
4
5      //print_uart("Type 'r' for recording, 'i' for impedance check,
       's' for stimulation, 'ML' for data classification:\n\n\r");
6
7      k_sem_take(&msg_received_sem, K_FOREVER);
8
9      if(msg_received)
10     {
11       msg_received = false;
12       if(k_msgq_get(&uart_msgq, &tx_buf, K_FOREVER)==0)
13       {
14         //print_uart(tx_buf);
15         //print_uart("\n\n\r");
16         if (strcmp((char *)tx_buf, "r") == 0){
17                       ...
```

```
18                  } else if (strcmp((char *)tx_buf, "i") == 0){
19                      ...
20                  } else if (strcmp((char *)tx_buf, "s") == 0){
21                  ...
22              } else if (strcmp((char *)tx_buf, "ML") == 0){
23                      ...
24                  }
25              }
26          }
27      }
28  }
```

Once the semaphore is given, the function retrieves the command from the Zephyr message queue (`uart_msgq`) and compares it with the expected strings:

- `"r"`: starts the recording procedure.
- `"i"`: starts the impedance check routine, allowing the user to configure the test parameters (sampling frequency and series capacitor).
- `"s"`: starts the stimulation routine, not implemented in this thesis project.
- `"ML"`: triggers the data classification stage based on the trained machine learning model, not implemented in this thesis project.

Each command activates a specific high-level routine within the firmware, managing the acquisition or the Impedance Check.

The Figure 4.17 shows the link between the Software code and the Firmware code.

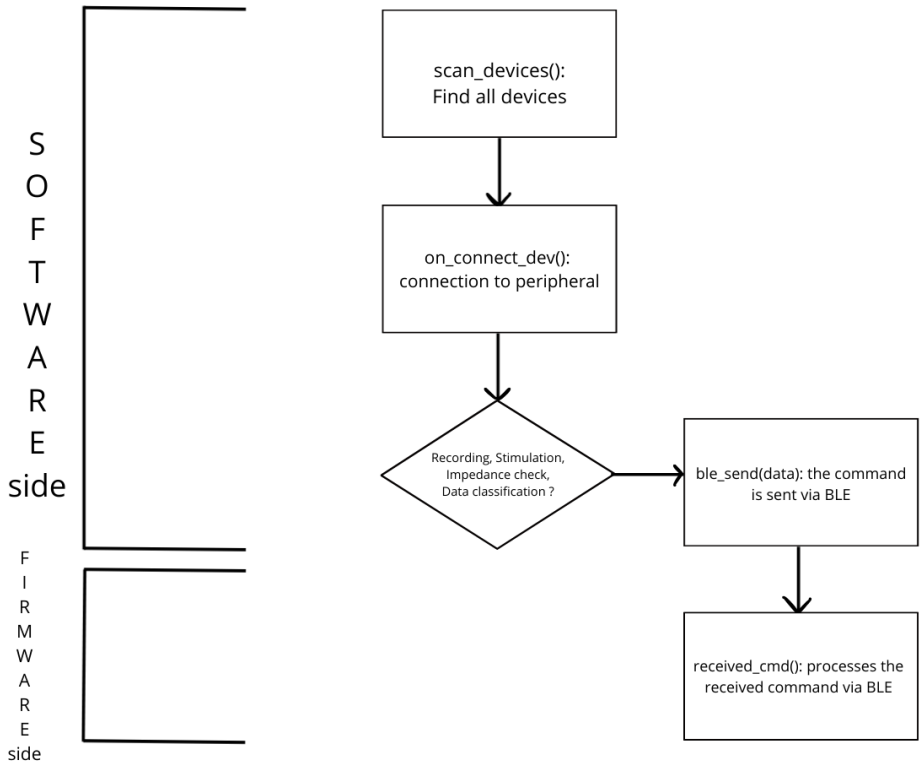


Figure 4.17: SW vs FW

Chapter 5

Results

5.1 Data acquisition

The verification of the data acquisition chain requires a known test waveform. For this purpose, a sinusoidal signal was generated using a function generator.
Since the available generator provides a minimum output amplitude of $100\,mV_{pp}$, while the Intan analog front-end operates with input signals on the order of a few millivolts, a passive resistive voltage divider was introduced to scale down the waveform amplitude.
The divider was implemented using a series resistor R of $220\,k\Omega$ and a resistor $R_{electrode}$ of $2.18\,k\Omega$ (across which the signal is taken and fed to the electrode), yielding an attenuation factor:

$$A = \frac{R_{electrode}}{R + R_{electrode}} \tag{5.1}$$

Thus, for a sinusoidal waveform with amplitude $200\,mV_{pp}$ and frequency $1\,kHz$, the resulting peak-to-peak voltage at the divider output is:

$$V_{in} = 200\,mV_{pp} \cdot \frac{2.18\,k\Omega}{220\,k\Omega + 2.18\,k\Omega} \approx 1.96\,mV_{pp} \tag{5.2}$$

The waveform was acquired in real time and visualized through the graphical user interface (GUI).
The experiments were conducted by varying the following parameters:

- Bandwidth of the Intan chip bandpass filter (low cut-off frequency f_L and high cut-off frequency f_H)
- Amplitude of the test waveform
- Frequency of the test waveform

- Number of acquired channels

Following acquisition, the recorded data were processed offline, and the Power Spectral Density (PSD) was computed and analyzed.

5.1.1 Peak-to-Peak Amplitude Estimation from the PSD

The peak-to-peak amplitude of the acquired sinusoidal signal was estimated from its Power Spectral Density (PSD). The PSD was computed in MATLAB using the `pwelch` function, which returns a spectral density expressed in V^2/Hz.
First, the dominant frequency component f_0 was identified as the frequency corresponding to the maximum value of the PSD. The PSD was then integrated over a narrow bandwidth around this frequency in order to estimate the power contained in the spectral peak of interest:

$$P = \int_{f_0-\Delta f}^{f_0+\Delta f} PSD(f)\, df \tag{5.3}$$

where Δf defines the bandwidth around the dominant frequency. Since the integral of the PSD corresponds to the signal power, the RMS amplitude of the sinusoidal component can be obtained as:

$$V_{rms} = \sqrt{P}. \tag{5.4}$$

Then, the peak-to-peak amplitude is computed as

$$V_{pp} = 2\sqrt{2}\, V_{rms}. \tag{5.5}$$

5.1.2 Anti-Aliasing Filter

The test waveform is acquired by the Analog-to-Digital Converter (ADC) integrated in the Intan chip. According to the Nyquist–Shannon sampling theorem, a band-limited signal can be perfectly reconstructed if the sampling frequency f_s satisfies

$$f_s \geq 2f_{max} \tag{5.6}$$

where f_{max} is the highest frequency component present in the signal.
If the analog signal contains spectral components above the Nyquist frequency

$$f_N = \frac{f_s}{2}, \tag{5.7}$$

these components are folded into the baseband during the sampling process, generating aliasing artifacts. A frequency component $f > f_N$ is observed at

$$f_{alias} = |f - kf_s| \tag{5.8}$$

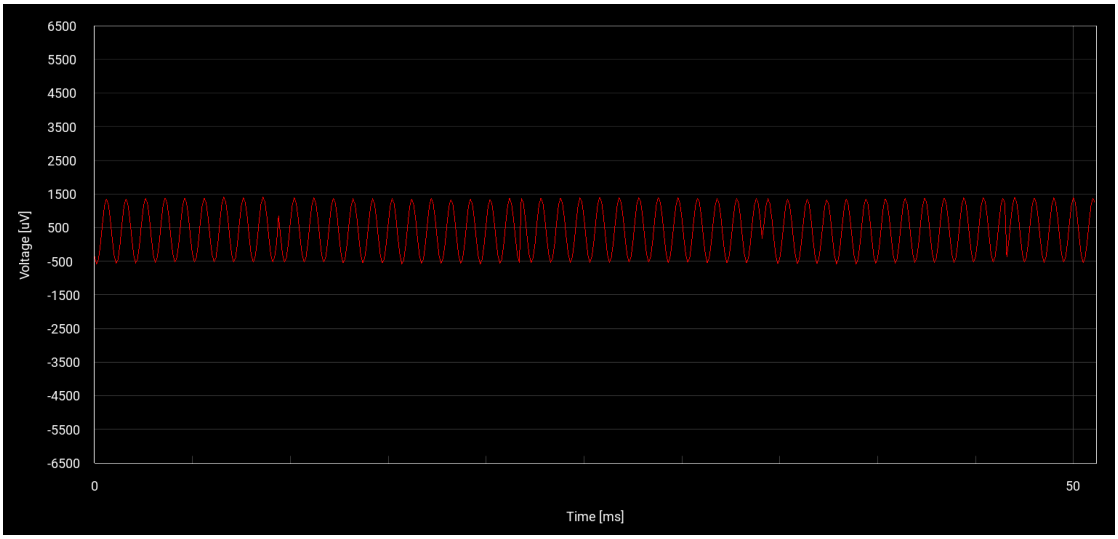
with $k \in \mathbb{Z}$.
Aliasing is an irreversible phenomenon because, once the signal has been sampled, the aliased components are indistinguishable from the original low-frequency components.
To prevent aliasing, the Intan chip includes analog bandpass filters placed before the ADC. In particular, the upper cutoff frequency f_H must be chosen sufficiently below the Nyquist frequency:

$$f_H < f_N \tag{5.9}$$

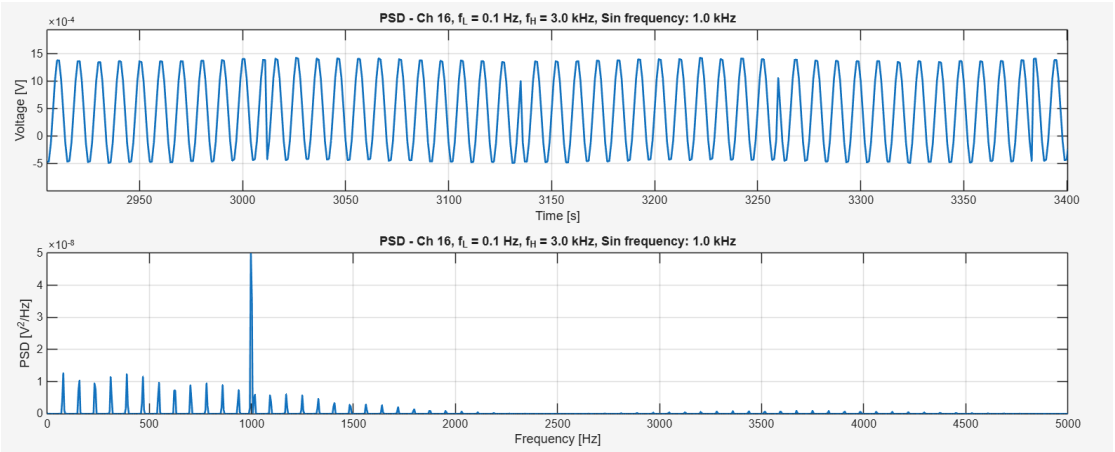
This condition ensures that frequency components near and above f_N are adequately attenuated before sampling, thereby minimizing spectral folding. In addition, the filter allows the frequency band of interest to be isolated.

5.1.3 Sinusoidal wave, 2 mVpp, 1 kHz

The sampling frequency of the ADC is $10\,kHz$ per channel. The acquired waveform is a sinusoidal signal with frequency $1\,kHz$ and amplitude $2\,mV_{pp}$, obtained by attenuating the function generator output through the resistive voltage divider.
The bandpass filter of the Intan chip was configured with low cut-off frequency $f_L = 0.1\,Hz$ and high cut-off frequency $f_H = 3\,kHz$.
The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.1a and Fig. 5.1b, respectively. The dominant spectral component appears at $1.0\,kHz$.



(a) Real-time acquired wave, $f_L = 0.1\ Hz$, $f_H = 3\ kHz$

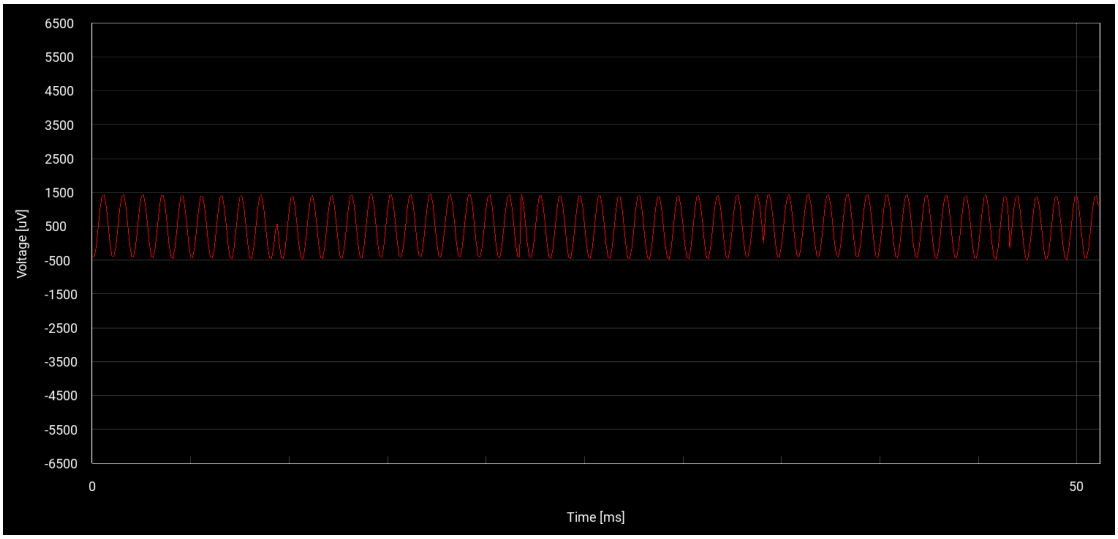


(b) Off-line Acquired wave and PSD: $f_L = 0.1\ Hz$, $f_H = 3kHz$

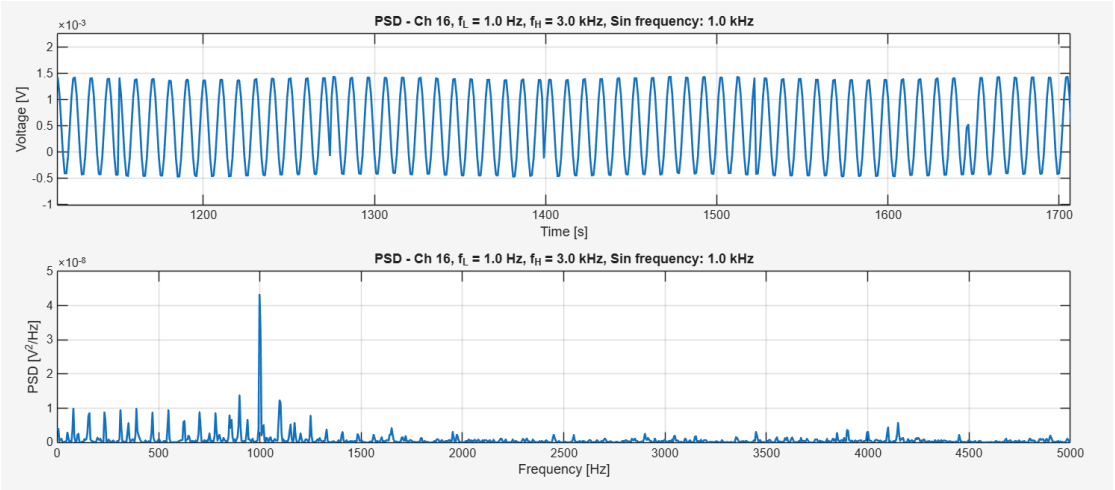
Figure 5.1: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 0.1\ Hz$,$f_H = 3\ kHz$

Bandpass filter: $f_L = 1\ Hz$, $f_H = 3\ kHz$

The Lower Cut-Off frequency is increased from 0.1 Hz to 1 Hz. The amplitude of the generated sinusoidal signal is kept at 2 mV_{pp}. The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.2a and Fig. 5.2b, respectively. The dominant spectral component appears at $1.0\,kHz$.



(a) Real-time acquired wave, $f_L = 1\ Hz$, $f_H = 3\ kHz$

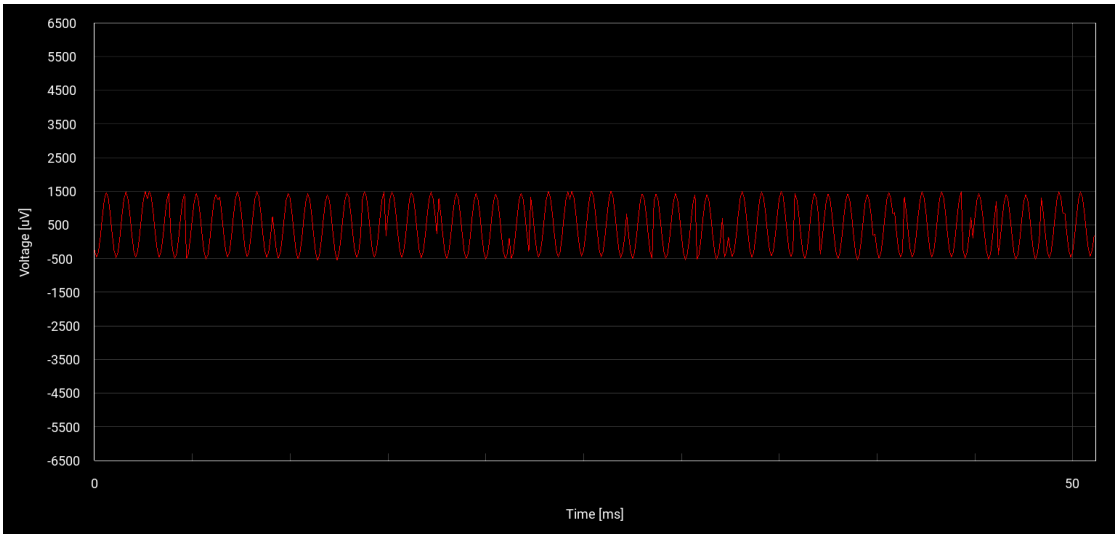


(b) Off-line acquired wave and PSD: $f_L = 1\ Hz$, $f_H = 3\ kHz$

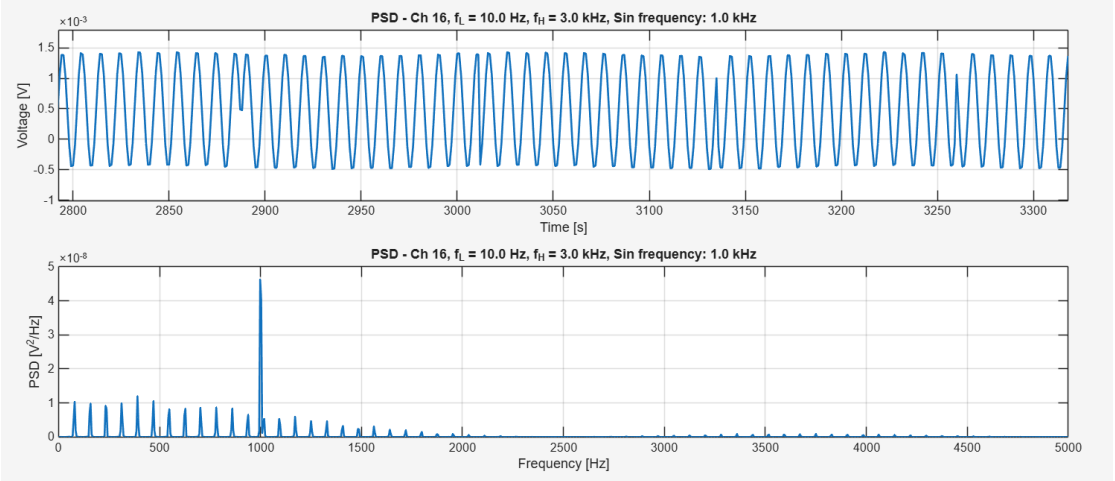
Figure 5.2: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 1\ Hz$, $f_H = 3\ kHz$

Bandpass filter: $f_L = 10\ Hz$, $f_H = 3\ kHz$

The Lower Cut-Off frequency is increased from 1 Hz to 10 Hz. The amplitude of the generated sinusoidal signal is kept at 2 mV_{pp}. The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.3a and Fig. 5.3b, respectively. The dominant spectral component appears at $1.0\,kHz$.



(a) Real-time Acquired wave, $f_L = 10\ Hz$, $f_H = 3\ kHz$

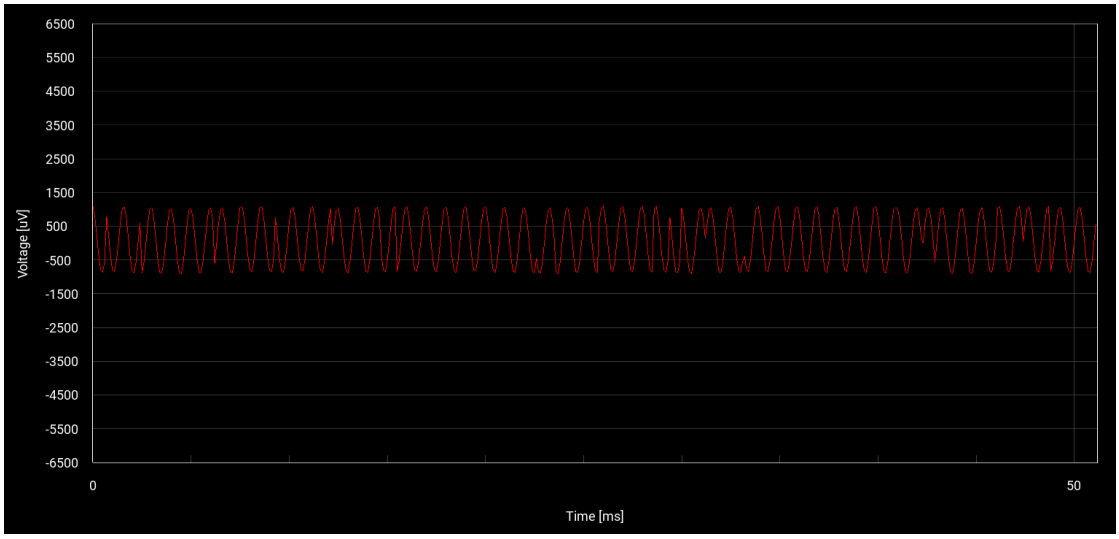


(b) Off-line Acquired wave, $f_L = 10\ Hz$, $f_H = 3\ kHz$

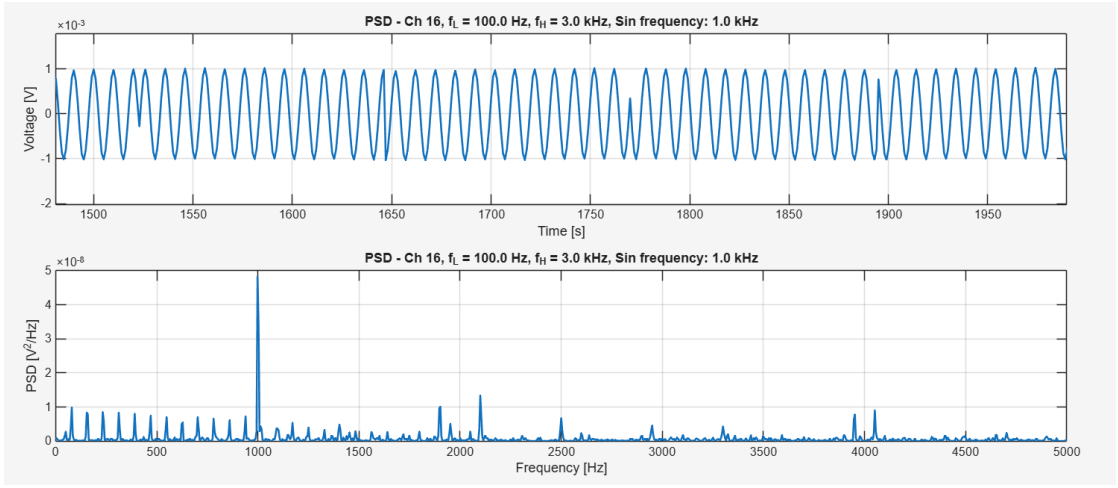
Figure 5.3: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 10\ Hz$, $f_H = 3\ kHz$

Bandpass filter: $f_L = 100\ Hz$, $f_H = 3\ kHz$

The Lower Cut-Off frequency is increased from 10 Hz to 100 Hz. The amplitude of the generated sinusoidal signal is kept at 2 mV_{pp}. The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.4a and Fig. 5.4b, respectively. The dominant spectral component appears at 1.0 kHz.



(a) Real-time Acquired wave, $f_L = 100\ Hz$, $f_H = 3\ kHz$

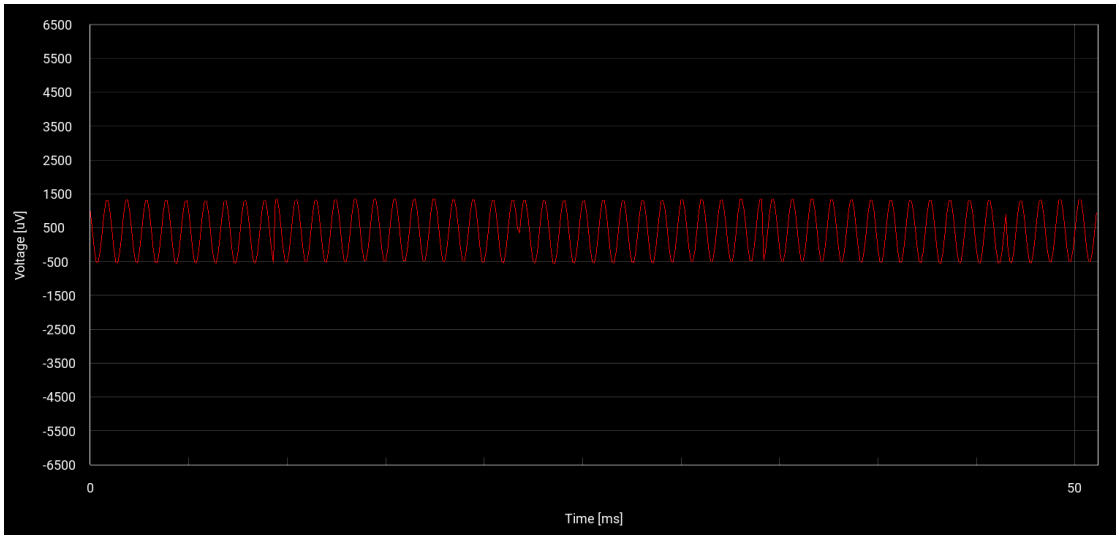


(b) Off-line acquired wave and PSD: $f_L = 100\ Hz$, $f_H = 3\ kHz$

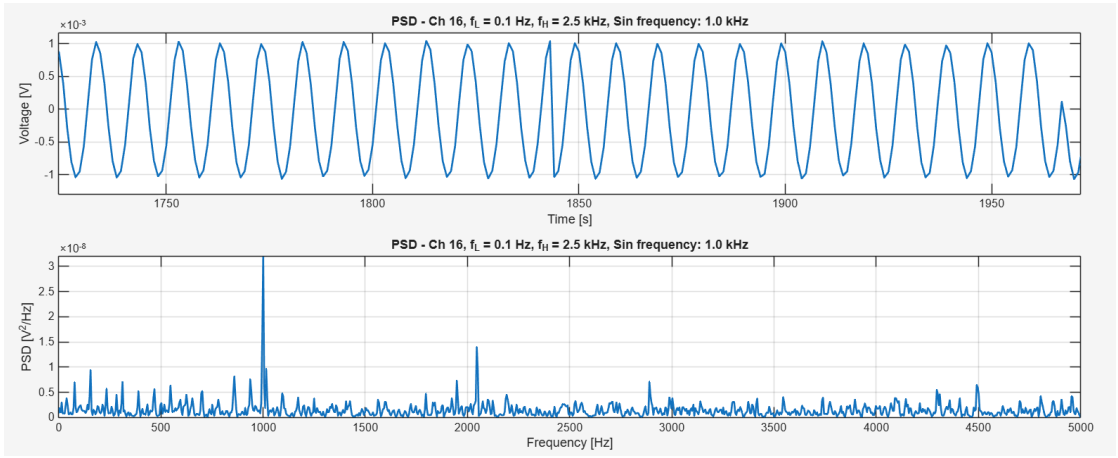
Figure 5.4: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 100\ Hz$, $f_H = 3\ kHz$

Bandpass filter: $f_L = 0.1\ Hz$, $f_H = 2.5\ kHz$

The Upper Cut-Off frequency is decreased from 3 kHz to 2.5 kHz. The amplitude of the generated sinusoidal signal is kept at 2 mV_{pp}. The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.5a and Fig. **??**, respectively. The dominant spectral component appears at 1.0 kHz.



(a) Real-time Acquired wave, $f_L = 0.1\ Hz$, $f_H = 2.5\ kHz$

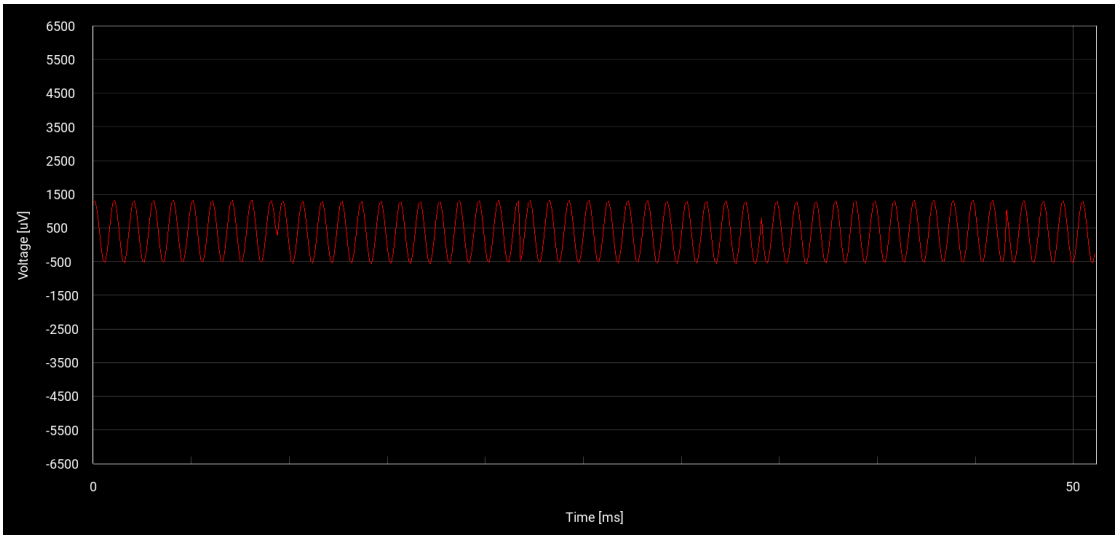


(b) Off-line acquired wave and PSD, $f_L = 0.1\ Hz$, $f_H = 2.5\ kHz$

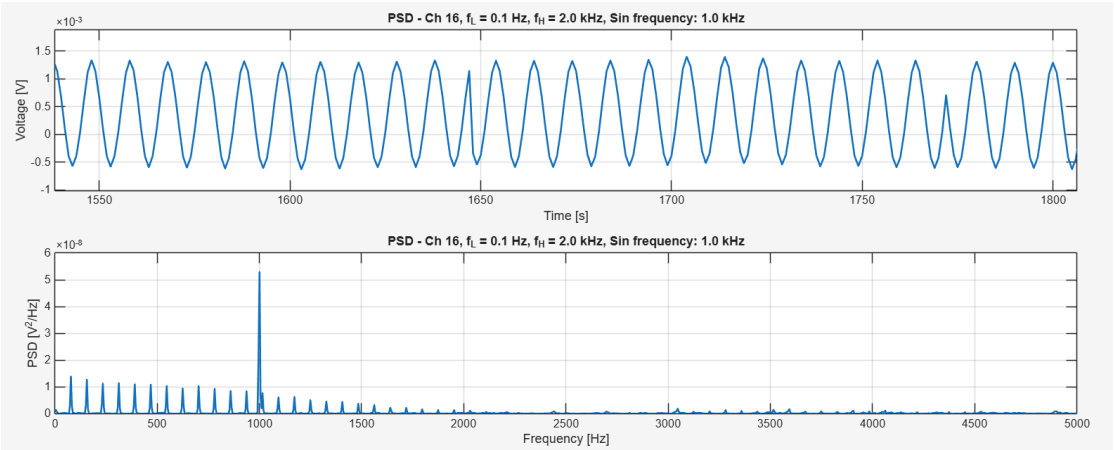
Figure 5.5: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 0.1\ Hz$, $f_H = 2.5\ kHz$

Bandpass filter: $f_L = 0.1\ Hz$, $f_H = 2.0\ kHz$

The Upper Cut-Off frequency is decreased from 2.5 kHz to 2.0 kHz. The amplitude of the generated sinusoidal signal is kept at 2 mV_{pp}. The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.6a and Fig. 5.6b, respectively. The dominant spectral component appears at 1.0 kHz.



(a) Real-time acquired wave, $f_L = 0.1\ Hz$, $f_H = 2.0\ kHz$

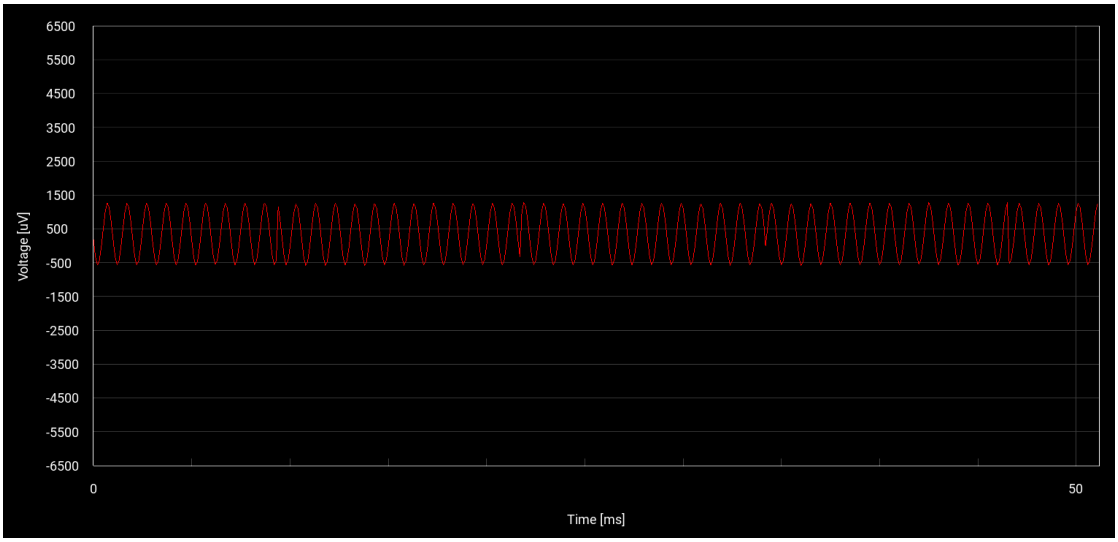


(b) Off-line Acquired wave and PSD: $f_L = 0.1\ Hz$, $f_H = 2.0\ kHz$

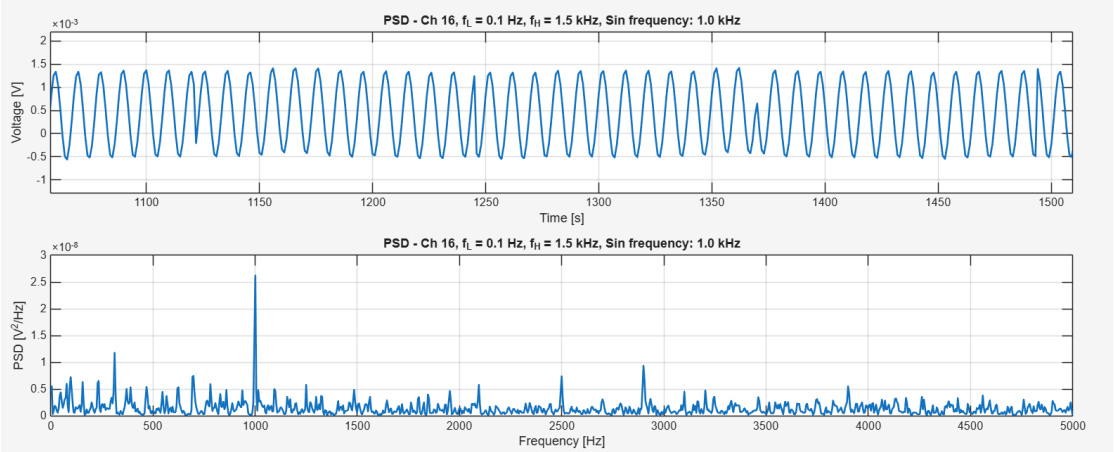
Figure 5.6: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 0.1\ Hz$, $f_H = 2.0\ kHz$

Bandpass filter: $f_L = 0.1\ Hz$, $f_H = 1.5\ kHz$

The Upper Cut-Off frequency is decreased from 2.0 kHz to 1.5 kHz. The amplitude of the generated sinusoidal signal is kept at 2 mV_{pp}. The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.7a and Fig. 5.7b, respectively. The dominant spectral component appears at 1.0 kHz.



(a) Real-time acquired wave, $f_L = 0.1\ Hz$, $f_H = 1.5\ kHz$

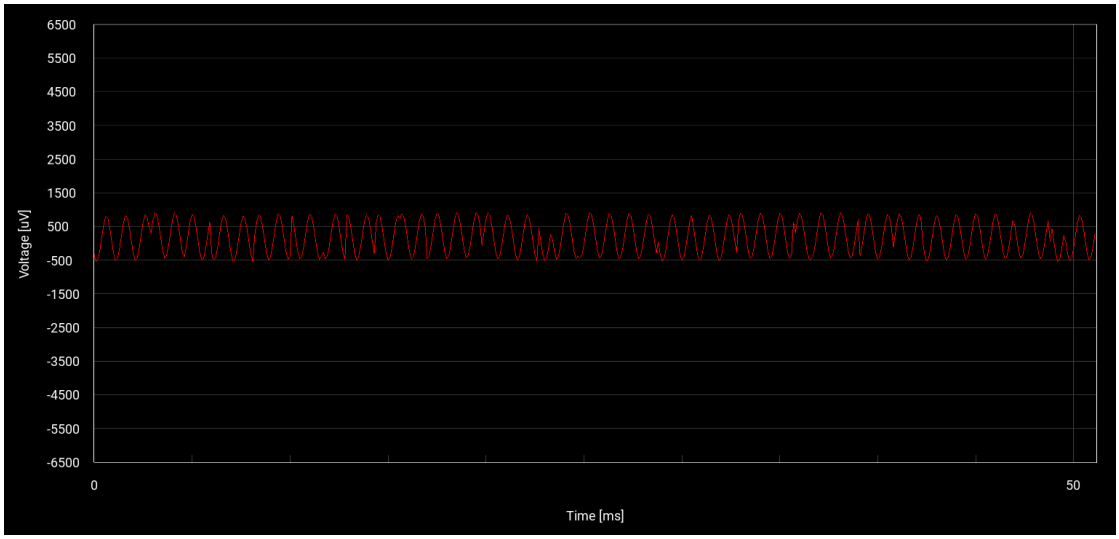


(b) Off-line acquired wave: $f_L = 0.1\ Hz$, $f_H = 1.5\ kHz$

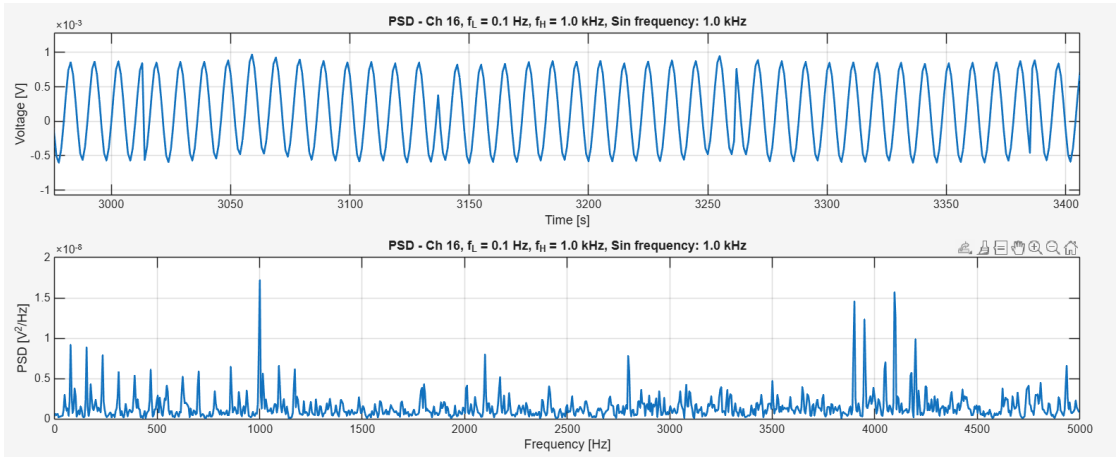
Figure 5.7: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 0.1\ Hz$, $f_H = 1.5\ kHz$

Bandpass filter: $f_L = 0.1\ Hz$, $f_H = 1.0\ kHz$

The Upper Cut-Off frequency is decreased from 1.5 kHz to 1.0 kHz. The amplitude of the generated sinusoidal signal is kept at 2 mV_{pp}. The Real-Time acquired waveform and the Power Spectral Density are shown in Fig. 5.8a and Fig. 5.8b, respectively. The dominant spectral component appears at 1.0 kHz.



(a) Real-time acquired wave, $f_L = 0.1\ Hz$, $f_H = 1.0\ kHz$



(b) Off-line acquired wave and PSD: $f_L = 0.1\ Hz$, $f_H = 1.0\ kHz$

Figure 5.8: Sinusoidal wave: 2 mVpp, 1 kHz, $f_L = 0.1\ Hz$, $f_H = 1.0\ kHz$

5.1.4 Peak-to-Peak Amplitude Estimation from the PSD: Variation of Bandwidth Parameters

The following tables report the peak-to-peak amplitude estimated from the Power Spectral Density, calculated as described in the previous section. Tables 5.1 and 5.2 present the peak-to-peak amplitudes corresponding to variations in the lower and upper cutoff frequencies, respectively.

	V_{pp} $[mV]$
$f_L = 0.1$ Hz, $f_H = 3$ kHz	1.97
$f_L = 1.0$ Hz, $f_H = 3$ kHz	1.96
$f_L = 10.0$ Hz, $f_H = 3$ kHz	1.99
$f_L = 100.0$ Hz, $f_H = 3$ kHz	1.99

Table 5.1: Lower cut-off frequency variation: peak-to-peak amplitudes

	V_{pp} $[mV]$
$f_L = 0.1$ Hz, $f_H = 2.5$ kHz	1.89
$f_L = 0.1$ Hz, $f_H = 2.0$ kHz	1.79
$f_L = 0.1$ Hz, $f_H = 1.5$ kHz	1.72
$f_L = 0.1$ Hz, $f_H = 1.0$ kHz	1.35
$f_L = 0.1$ Hz, $f_H = 750.0$ Hz	1.30

Table 5.2: Upper cut-off frequency variation: peak-to-peak amplitudes

5.1.5 Sinusoidal Wave: $2\,\mathrm{mV_{pp}}$, Frequency Variation

In this test, the frequency of the sinusoidal waveform was varied while keeping the amplitude constant at $2\,\mathrm{mV_{pp}}$. The filter parameters were set to $f_L = 0.1\,\mathrm{Hz}$ and $f_H = 3.0\,\mathrm{kHz}$.

The signal frequency was first set to 500 Hz and then increased to 1.4 kHz. The Power Spectral Density (PSD), shown in Fig. 6.5, exhibits a clear shift of the spectral peak corresponding to the selected frequencies.

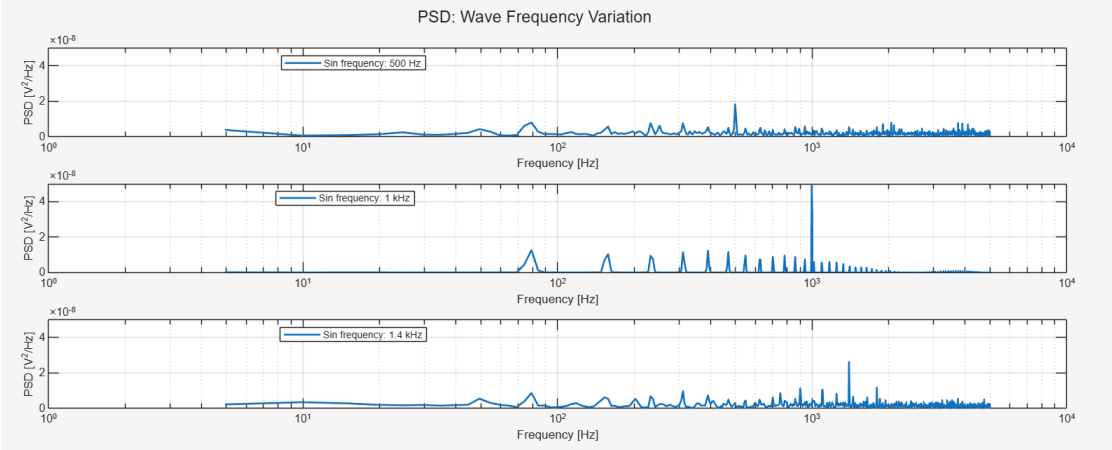


Figure 5.9: Power Spectral Density of the sinusoidal signal with amplitude $2\,\mathrm{mV_{pp}}$, $f_L = 0.1\,\mathrm{Hz}$ and $f_H = 3.0\,\mathrm{kHz}$, for different input frequencies.

5.1.6 Sinusoidal Wave: 1 kHz, Amplitude Variation

In this test, the amplitude of the sinusoidal waveform was varied while keeping the frequency fixed at 1 kHz. The filter parameters were set to $f_L = 0.1\,\mathrm{Hz}$ and $f_H = 3.0\,\mathrm{kHz}$.

The signal amplitude was first set to $0.5\,\mathrm{mV_{pp}}$ and then increased to $1.0\,\mathrm{mV_{pp}}$. The Power Spectral Density (PSD), shown in Fig. 6.6, exhibits a corresponding increase in the spectral peak at the selected frequency.

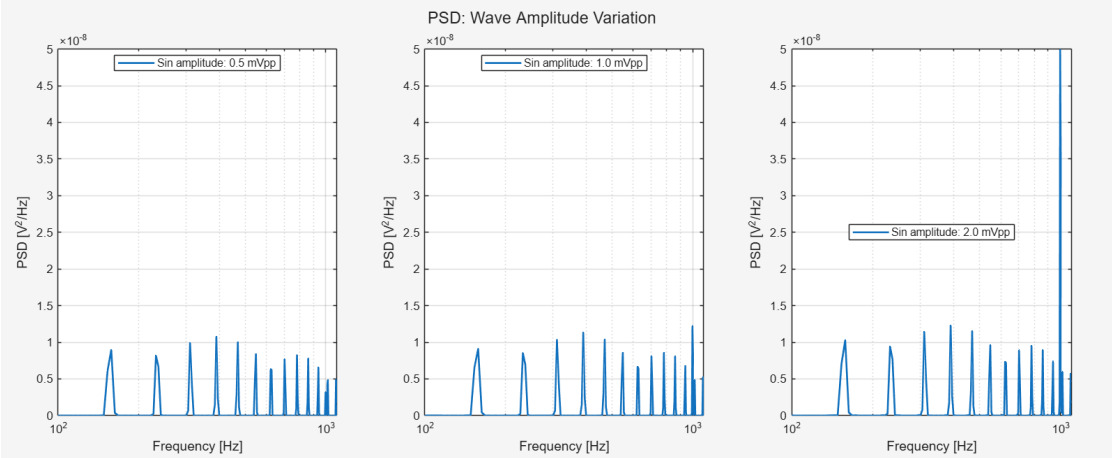


Figure 5.10: Power Spectral Density of the sinusoidal signal with frequency 1 kHz, $f_L = 0.1\,\mathrm{Hz}$ and $f_H = 3.0\,\mathrm{kHz}$, for different input amplitudes.

5.1.7 Multi-Channel Acquisition: Sinusoidal Wave, $2\,\mathrm{mV_{pp}}$, $1\,\mathrm{kHz}$

In this test, four channels were acquired simultaneously. The signal recorded on each channel was a sinusoidal waveform with an amplitude of $2\,\mathrm{mV_{pp}}$ and a frequency of $1\,\mathrm{kHz}$. The Real-Time acquired waves and the Off-Line acquired waves are shown in Fig. 5.11 and in Fig. 6.7, respectively. The Power Spectral Density is shown in Fig. 6.8;

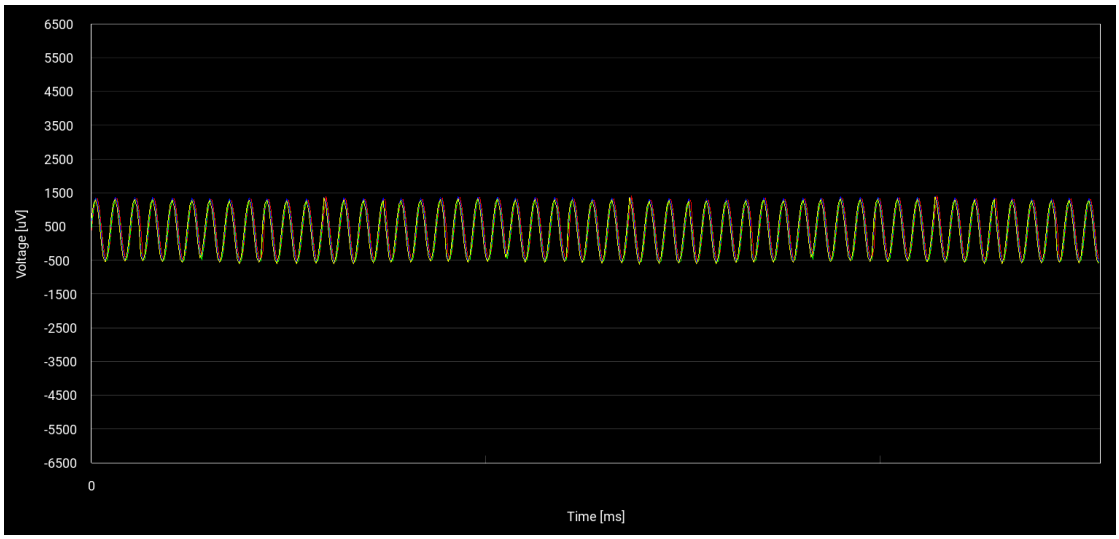


Figure 5.11: GUI: acquired waveforms from channels 16, 17, 18 and 19.

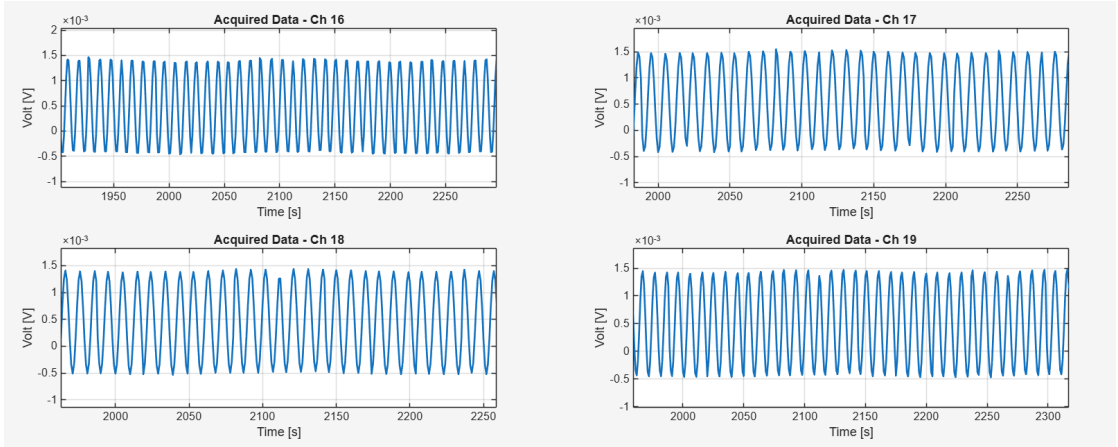


Figure 5.12: Offline acquired waves from channels 16, 17, 18 and 19.

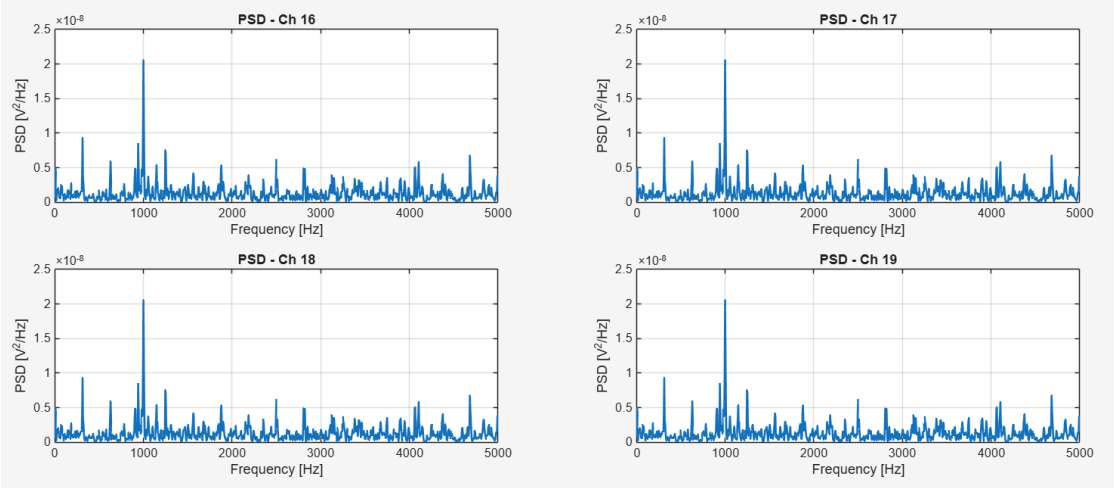


Figure 5.13: Power Spectral Density of the signals acquired from channels 16, 17, 18 and 19.

5.2 Electrode Impedance Test

The impedance of the electrodes was measured using the impedance testing module integrated in the Intan chip, as described in the previous section.
Briefly, the electrode impedance measurement requires enabling the DAC, which generates a sinusoidal voltage waveform $v_{DAC}(t)$. The series capacitor C_s converts this voltage into a current according to:

$$i_{DAC}(t) = C_s \frac{dv_{DAC}(t)}{dt} \tag{5.10}$$

This current is injected into the selected electrode by closing the corresponding switch. The peak injected current is given by:

$$I_p = 2\pi f_{DAC} C_s V_A \tag{5.11}$$

where $V_A = 0.6125\,V$ is the amplitude of the DAC waveform.

The electrode impedance magnitude is then calculated as:

$$Z = \frac{V_p}{I_p} \tag{5.12}$$

where V_p is the peak voltage measured by the ADC.

The developed firmware for the impedance check was validated using precision

resistors with known nominal values. Since good-quality electrodes typically exhibit impedance values in the range 1.0 $k\Omega$–10.0 $k\Omega$, the following resistor values were tested:

- 1.2 $k\Omega$
- 10.0 $k\Omega$
- 110.0 $k\Omega$
- 1.0 $M\Omega$

The tests were performed as follows:

- Single-channel measurements, varying:
  - The resistor value,
  - The series capacitor C_s,
  - The DAC waveform frequency.
- Multi-channel measurements, with different resistor values on each channel.

5.2.1 Estimation of the Actual DAC Current

The ideal DAC current is given by:

$$i_{p_{ideal}} = 2\pi V_A f_{DAC} C_s \tag{5.13}$$

However, the DAC output drives the series combination of the capacitor C_s and the electrode impedance $Z_{\text{electrode}}$. Therefore, a more accurate model has been developed to estimate the actual DAC current. The corresponding electrical model is illustrated in Fig. 5.14.

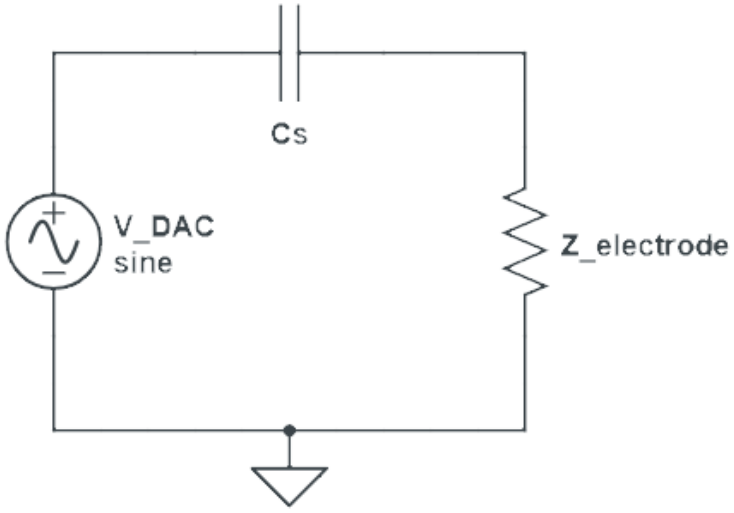


Figure 5.14: Circuit model for the DAC current.

The resulting current can be expressed as:

$$i_{p_{actual}} = \frac{V_{DAC}}{Z_{electrode} + \frac{1}{j\omega C_s}} \tag{5.14}$$

5.2.2 Data Acquisition from the Oscilloscope

In order to verify the correct functionality of the DAC current generator, the voltage drop across the selected electrode was acquired and visualized using an oscilloscope. Since these voltages are very small, ranging from a few microvolts to a few millivolts, they were amplified using the INA126 instrumentation amplifier (Fig. 5.15). The amplifier provides an adjustable gain that can be set through the external resistor R_g. The gain value is computed according to the relationship provided in the device datasheet.

$$G \; = \; 5 \; + \; \frac{80 \; k\Omega}{R_g} \tag{5.15}$$

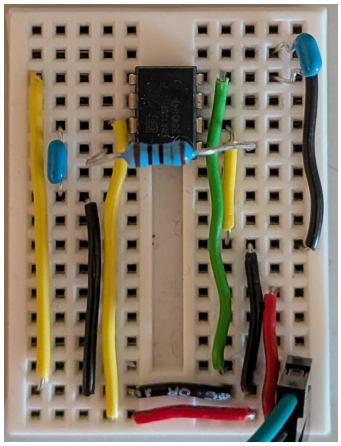


Figure 5.15: Amplification circuit for oscilloscope visualization

5.2.3 Electrode impedance: 1.2 $k\Omega$, DAC frequency 900 Hz

The electrode under test had a nominal impedance of 1.2 $k\Omega$, while the frequency of the generated DAC waveform was set to 900 Hz. The impedance was estimated by varying the series capacitor C_s. The acquired voltage signals were filtered during the post-processing stage.
The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.16 and Fig. 5.17, respectively.

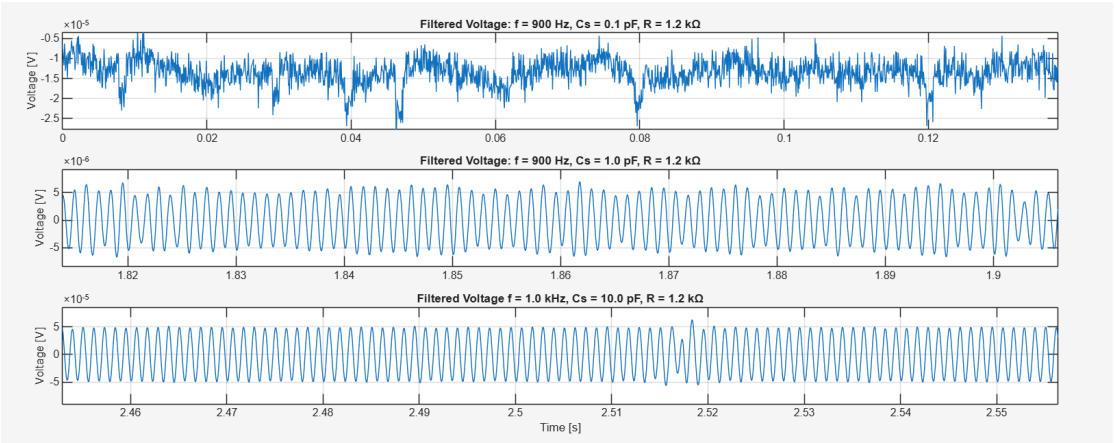


Figure 5.16: Acquired filtered voltage: $f_{DAC} = 900\ Hz$, $R = 1.2\ k\Omega$

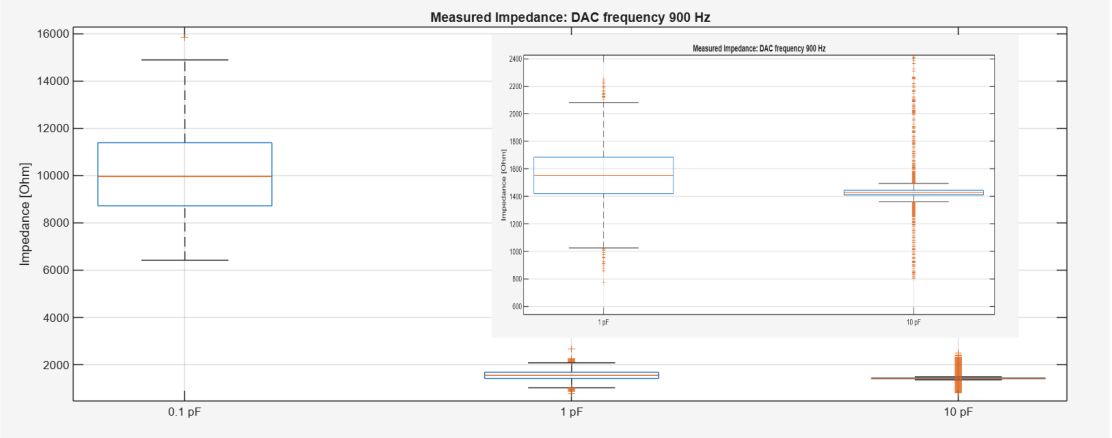


Figure 5.17: Measured Impedance: $f_{DAC} = 900\ Hz$, $R = 1.2\ k\Omega$

Table 5.3 reports the expected voltage drop across the electrode together with the corresponding measured values.

The measured peak voltage for $C_s = 0.1\ pF$ does not match the expected value. This discrepancy is mainly due to the extremely small voltage amplitude generated across the electrode. The amplifier input-referred noise is 2.4 μV_{rms}. Therefore, the Signal-to-Noise Ratio (SNR) for $C_s = 0.1\ pF$ is very low:

$$SNR = \frac{V_{rms}(\text{electrode})}{2.40\ \mu V_{rms}} = \frac{0.293\ \mu V}{2.40\ \mu V} = 0.12 \tag{5.16}$$

The expected signal amplitude is therefore below the noise floor. Consequently, the impedance estimation exhibits a significant error when $C_s = 0.1\,pF$ is used. In this case, the measured peak voltage is 3.45 μV. As a result, the estimated impedance is:

$$Z = \frac{V_p}{I_p} = \frac{3.45\ \mu V}{0.346\ nA} = 9.971\ k\Omega \tag{5.17}$$

In the other cases, i.e., for $C_s = 1.0\ pF$ and $C_s = 10.0\ pF$, the impedance estimation is reliable.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.346 nA	3.46 nA	34.6 nA
Expected DAC voltage V_p	0.415 μV	4.15 μV	41.5 μV
Measured DAC voltage V_p	3.45 μV	5.36 μV	49.33 μV
Impedance Median	9.96 $k\Omega$	1.54 $k\Omega$	1.42 $k\Omega$
Impedance IQR	2.67 $k\Omega$	264.65 Ω	33.56 Ω

Table 5.3: Measured quantities: $R = 1.2\ k\Omega$, $f_{DAC} = 900\ Hz$

The Real-Time acquired impedance is shown in Fig. 5.18. The real-time estimation is close to the expected value.

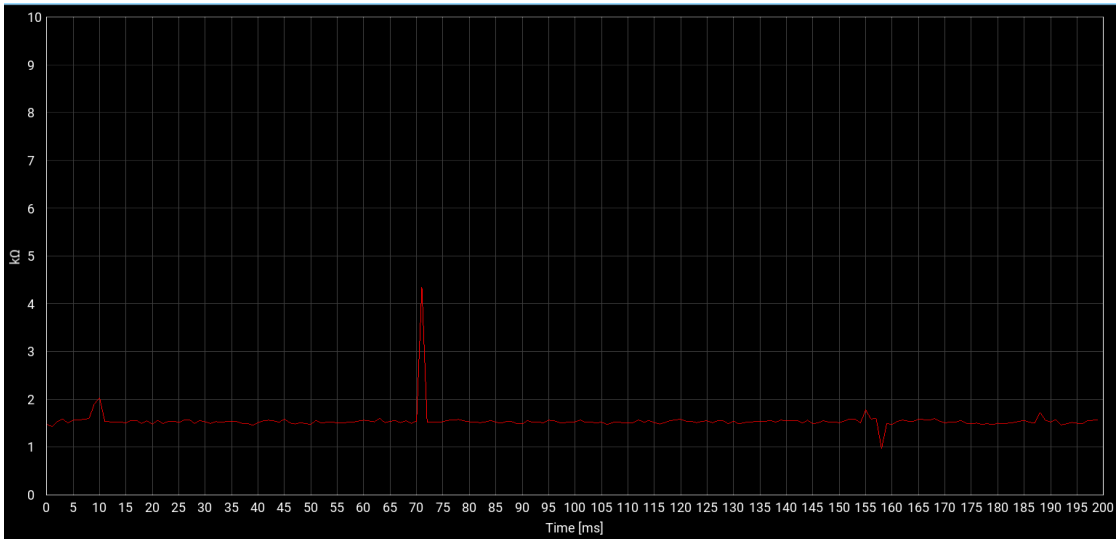


Figure 5.18: Real-Time acquired impedance: $f_{DAC} = 900\ Hz$, $R = 1.2\ k\Omega$, $C_s = 10\ pF$

5.2.4 Electrode impedance: 10.0 $k\Omega$, DAC frequency 900 Hz

The test was repeated by replacing the electrode of 1.2 $k\Omega$ nominal impedance with an electrode of 10.0 $k\Omega$ nominal impedance. The DAC waveform frequency was kept at 900 Hz. The impedance was then estimated by varying the series capacitor C_s.
The acquired voltage signals were filtered during the post-processing stage.
The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.19 and Fig. 5.20, respectively.

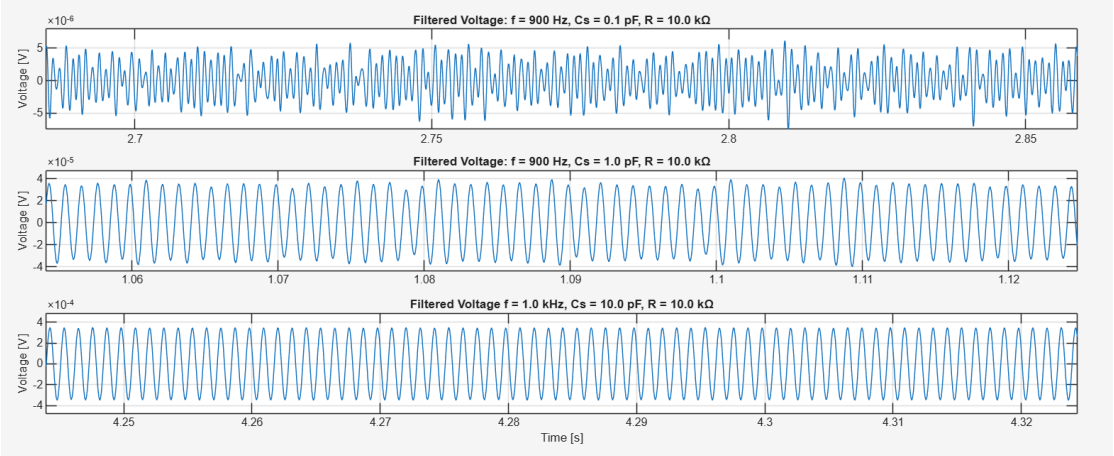


Figure 5.19: Acquired filtered voltage: $f_{DAC} = 900\ Hz$, $R = 10.0\ k\Omega$

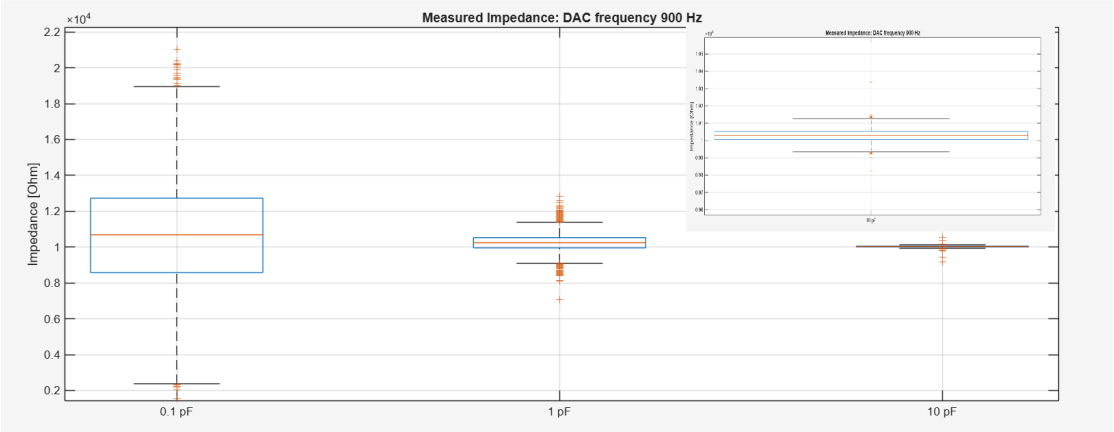


Figure 5.20: Measured Impedance: $f_{DAC} = 900\ Hz$, $R = 10.0\ k\Omega$

Table 5.4 reports the expected voltage drop across the electrode together with the corresponding measured values.
The measured peak voltages match the expected values for each selected C_s.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.346 nA	3.46 nA	34.6 nA
Expected DAC voltage V_p	3.46 μV	34.6 μV	0.346 mV
Measured DAC voltage V_p	3.70 μV	35.47 μV	0.347 mV
Impedance Median	10.68 $k\Omega$	10.24 $k\Omega$	10.03 $k\Omega$
Impedance IQR	4.15 $k\Omega$	573.78 Ω	48.30 Ω

Table 5.4: Measured quantities: $R = 10.0\ k\Omega$, $f_{DAC} = 900\ Hz$

The Real-Time acquired impedance is shown in Fig. 5.21. The real-time estimation is close to the expected value.

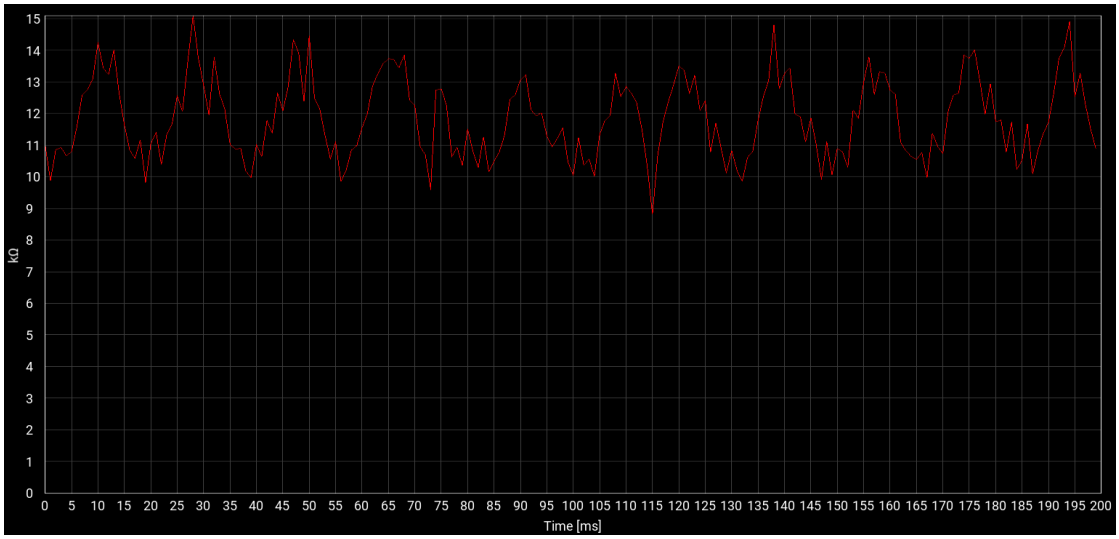


Figure 5.21: Real-Time acquired impedance: $f_{DAC} = 900\ Hz$, $R = 10.0\ k\Omega$, $C_s = 1\ pF$

The voltage drop across the electrode under test was also acquired using an oscilloscope, as described previously, through the instrumentation amplifier INA126. The amplifier gain was set to $G = 541$. The expected peak-to-peak voltage is $0.692\ mV$.
In Fig. 5.22, the amplified peak-to-peak voltage is approximately $0.35\ V$. Therefore, the voltage before amplification can be estimated as:

$$V_{pp} = \frac{V_{pp_{ampl}}}{G} = \frac{0.35\ V}{541} = 0.646\ mV \tag{5.18}$$

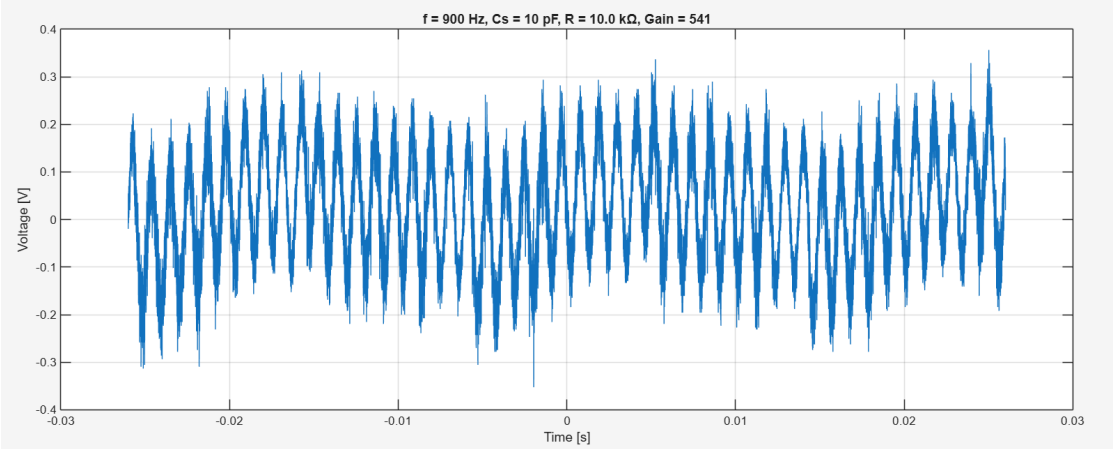


Figure 5.22: Acquired voltage from the oscilloscope: $f_{DAC} = 900\ Hz$, $R = 10.0\ k\Omega$, $C_s = 10\ pF$

The peak-to-peak voltage amplitude corresponds to the expected value.

5.2.5 Electrode impedance: 110.0 $k\Omega$, DAC frequency 900 Hz

The test was repeated by replacing the electrode of 10.0 $k\Omega$ nominal impedance with an electrode of 110.0 $k\Omega$ nominal impedance. The DAC waveform frequency was kept at 900 Hz. The impedance was then estimated by varying the series capacitor C_s.
The acquired voltage signals were filtered during the post-processing stage.
The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.23 and Fig. 5.24, respectively.

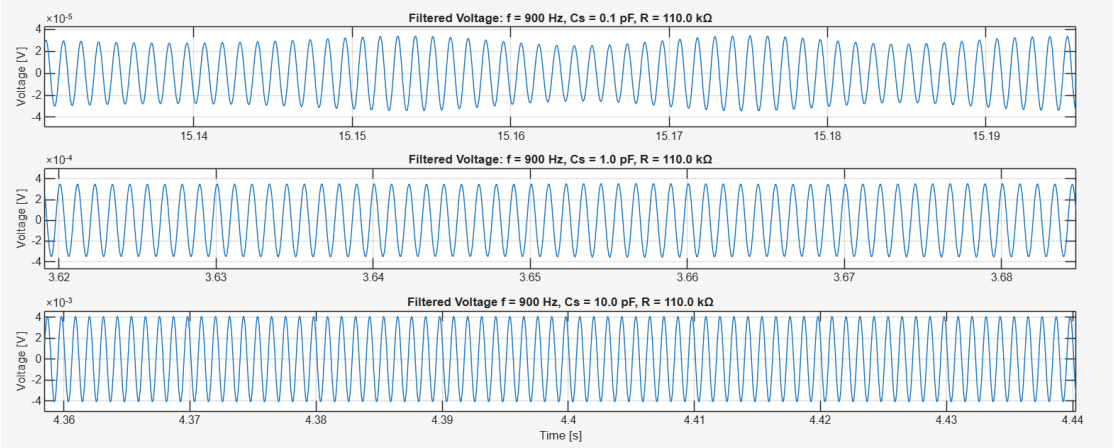


Figure 5.23: Acquired filtered voltage: $f_{DAC} = 900\ Hz$, $R = 110.0\ k\Omega$

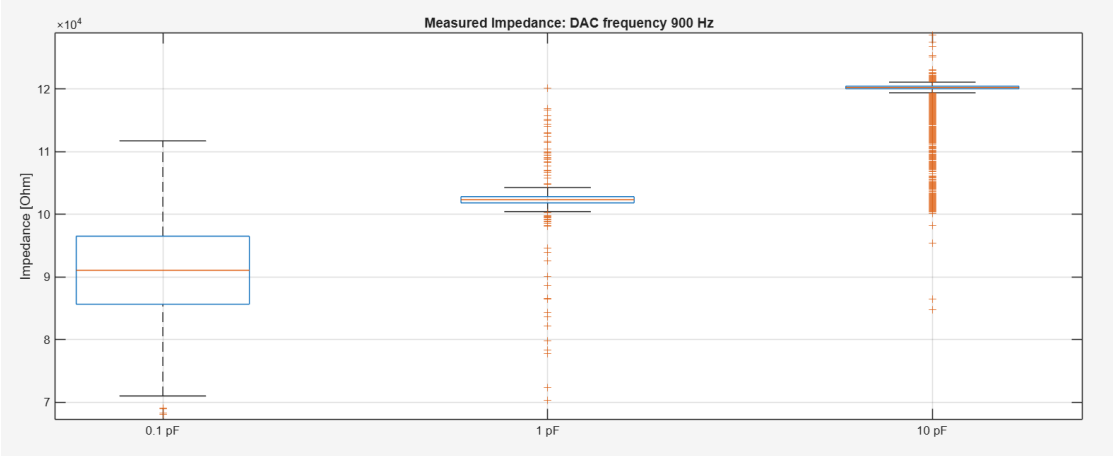


Figure 5.24: Measured Impedance: $f_{DAC} = 900\ Hz$, $R = 110.0\ k\Omega$

Table 5.5 reports the expected voltage drop across the electrode together with the corresponding measured values.
The measured peak voltages match the expected values for each selected C_s. Moreover, both the median impedance and the interquartile range (IQR) indicate excellent measurement reliability for all C_s values tested.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.346 nA	3.46 nA	34.6 nA
Expected DAC voltage V_p	38.06 μV	0.381 mV	3.81 mV
Measured DAC voltage V_p	31.54 μV	0.354 mV	4.20 mV
Impedance Median	91.07 $k\Omega$	102.33 $k\Omega$	118.96 $k\Omega$
Impedance IQR	10.86 $k\Omega$	1.00 $k\Omega$	243.44 Ω

Table 5.5: Measured quantities: $R = 110.0\ k\Omega$, $f_{DAC} = 900\ Hz$

The voltage drop across the electrode under test was also acquired using an oscilloscope, as described previously, through the instrumentation amplifier INA126. The amplifier gain was set to $G = 541$. The expected peak-to-peak voltage is 7.62 mV.
In Fig. 5.25, the amplified peak-to-peak voltage is approximately 3.5 V. Therefore, the voltage before amplification can be estimated as:

$$V_{pp} = \frac{V_{pp_{ampl}}}{G} = \frac{3.5\ V}{541} = 6.46\ mV \tag{5.19}$$

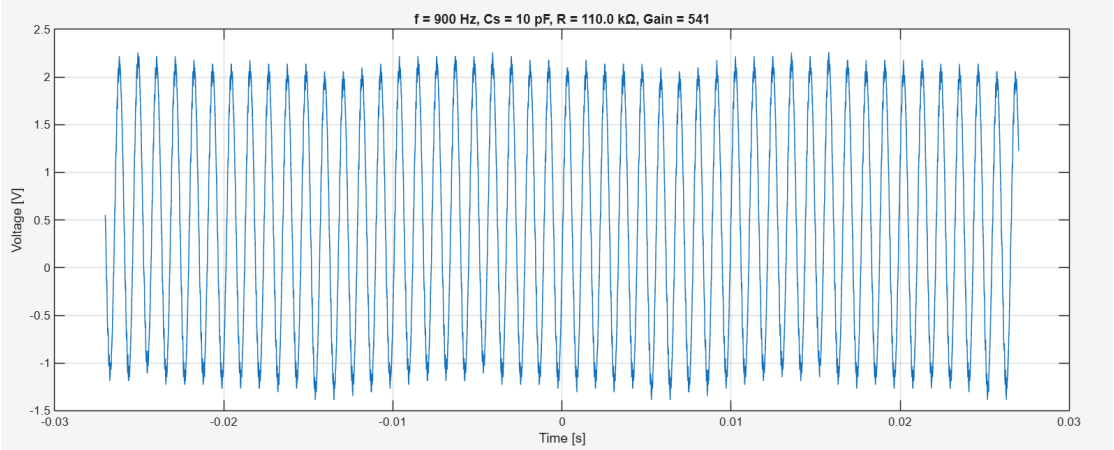


Figure 5.25: Acquired voltage from the oscilloscope: $f_{DAC} = 900\ Hz$, $R = 110.0\ k\Omega$, $C_s = 10\ pF$

The peak-to-peak voltage amplitude is close to the expected value.

5.2.6 Electrode impedance: 1.0 $M\Omega$, DAC frequency 900 Hz

The test was repeated by replacing the electrode with a nominal impedance of 110.0 $k\Omega$ with an electrode having a nominal impedance of 1.0 $M\Omega$. The DAC waveform frequency was kept at 900 Hz. The impedance was then estimated by varying the series capacitor C_s.
In this case, the series capacitor was varied from $C_s = 0.1\ pF$ to $C_s = 1.0\ pF$, since the expected voltage for $C_s = 10.0\ pF$ would exceed the maximum input range of the amplifiers, resulting in an Out Of Range (OOR) condition.

The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.26 and Fig. 5.27, respectively.

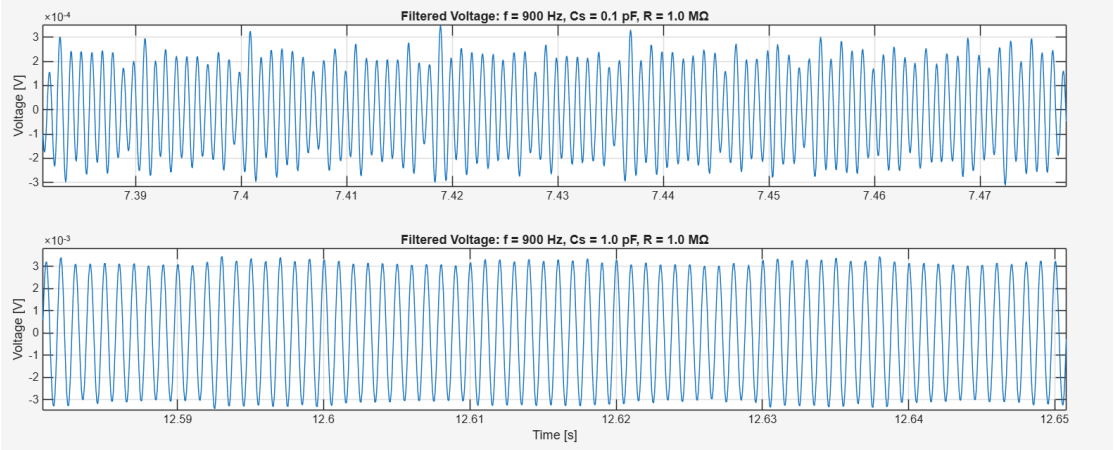


Figure 5.26: Acquired voltage: $f_{DAC} = 900\ Hz$, $R = 1.0\ M\Omega$

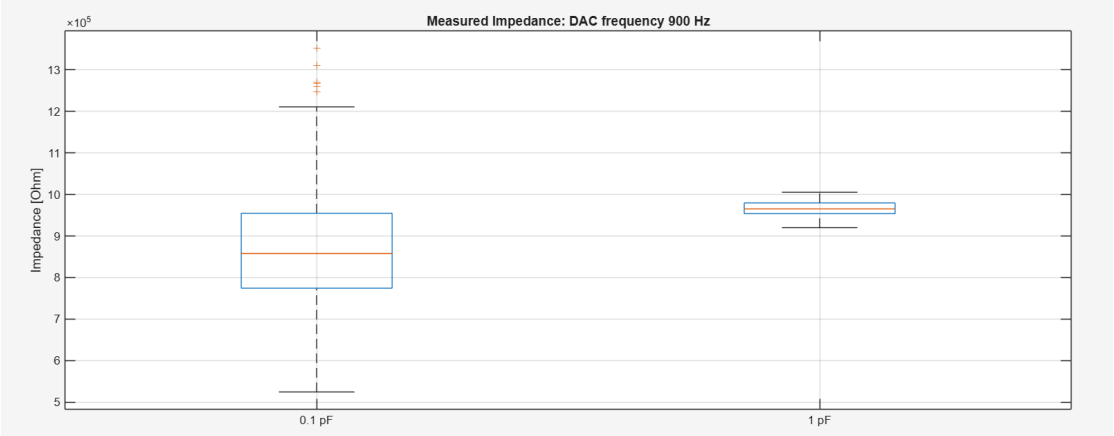


Figure 5.27: Measured Impedance: $f_{DAC} = 900\ Hz$, $R = 1.0\ M\Omega$

Table 5.6 reports the expected voltage drop across the electrode together with the corresponding measured values.

The median impedance values and the corresponding IQR values are reasonably consistent with the expected impedance.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.346 nA	3.46 nA	34.6 nA
Expected DAC voltage V_p	0.346 mV	3.46 mV	34.6 mV
Measured DAC voltage V_p	0.297 mV	3.3 mV	OOR
Impedance Median	858.07 $K\Omega$	965.20 $k\Omega$	OOR
Impedance IQR	180.45 $k\Omega$	25.89 $k\Omega$	OOR

Table 5.6: Measured quantities: $R = 1.0\ M\Omega$, $f_{DAC} = 900\ Hz$

5.2.7 Electrode impedance: DAC frequency 1.0 kHz

The test was repeated for the different electrode impedances by increasing the DAC waveform frequency from 900 Hz to 1.0 kHz. The impedance was estimated by varying the series capacitor C_s. The acquired voltage signals were filtered during the post-processing stage.

First, the electrode with a nominal impedance of 1.2 $k\Omega$ was tested. The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.28 and Fig. 5.29, respectively.

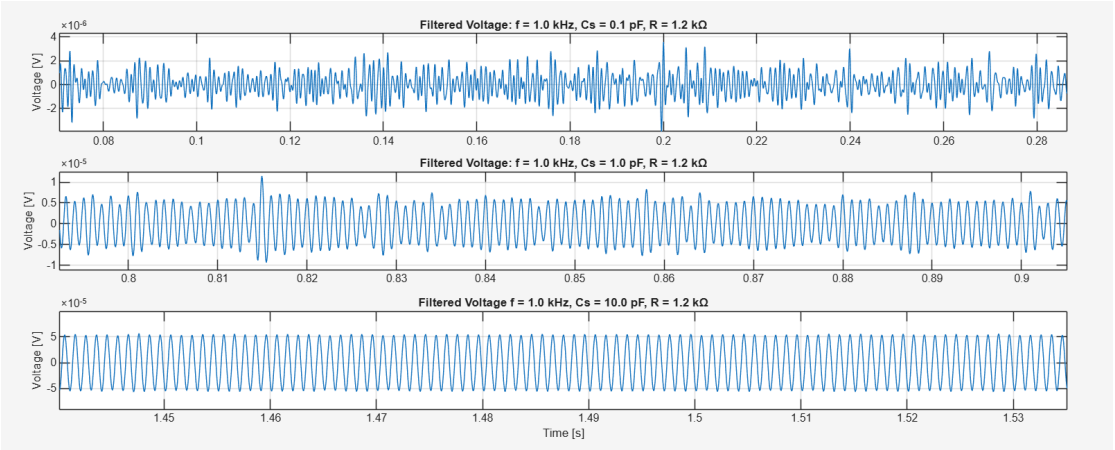


Figure 5.28: Acquired filtered voltage: $f_{DAC} = 1\ kHz$, $R = 1.2\ k\Omega$

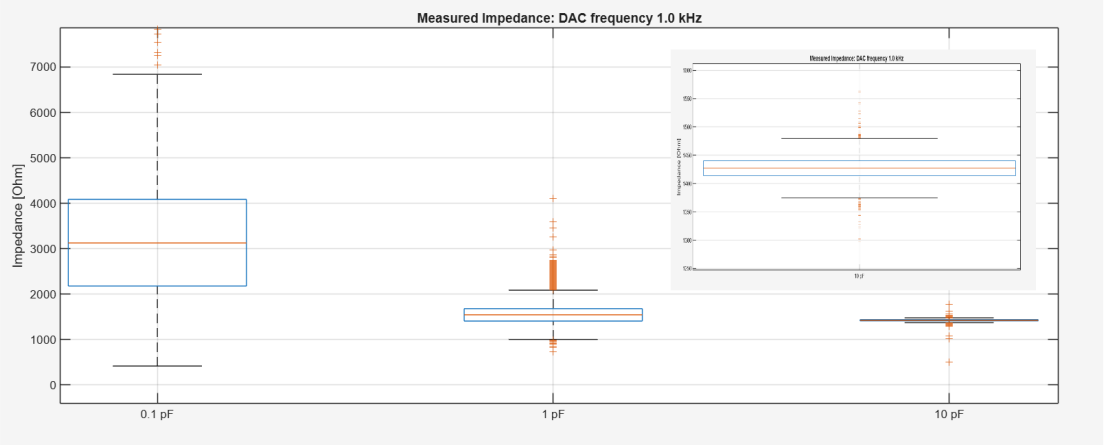


Figure 5.29: Measured Impedance: $f_{DAC} = 1\ kHz$, $R = 1.2\ k\Omega$

Table 5.7 reports the expected voltage drop across the electrode together with the corresponding measured values.

The measured peak voltage for $C_s = 0.1\ pF$ does not match the expected value. This discrepancy is mainly due to the extremely small voltage amplitude generated across the electrode. The amplifier input-referred noise is 2.4 μV_{rms}. Therefore, the Signal-to-Noise Ratio (SNR) for $C_s = 0.1\ pF$ is very low:

$$SNR = \frac{V_{rms}(\text{electrode})}{2.4\ \mu V_{rms}} = \frac{0.327\ \mu V}{2.4\ \mu V} = 0.14 \tag{5.20}$$

The expected signal amplitude is therefore below the noise floor. Consequently, the impedance estimation exhibits a significant error when $C_s = 0.1\,pF$ is used. In this case, the measured peak voltage is 1.20 μV. As a result, the estimated impedance is:

$$Z = \frac{V_p}{I_p} = \frac{1.20\ \mu V}{0.385\ nA} = 3.116\ k\Omega \tag{5.21}$$

In the other cases, i.e., for $C_s = 1.0\ pF$ and $C_s = 10.0\ pF$, the impedance estimation is reliable.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.385 nA	3.85 nA	38.5 nA
Expected DAC voltage V_p	0.462 μV	4.62 μV	46.2 μV
Measured DAC voltage V_p	1.20 μV	5.95 μV	54.91 μV
Impedance Median	3.12 $k\Omega$	1.54 $k\Omega$	1.43 $k\Omega$
Impedance IQR	1.90 $k\Omega$	272.86 Ω	26.40 Ω

Table 5.7: Measured quantities: $R = 1.2\ k\Omega$, $f_{DAC} = 1\ kHz$

The Real-Time acquired impedance is shown in Fig. 5.30. The real-time estimation is close to the expected value.

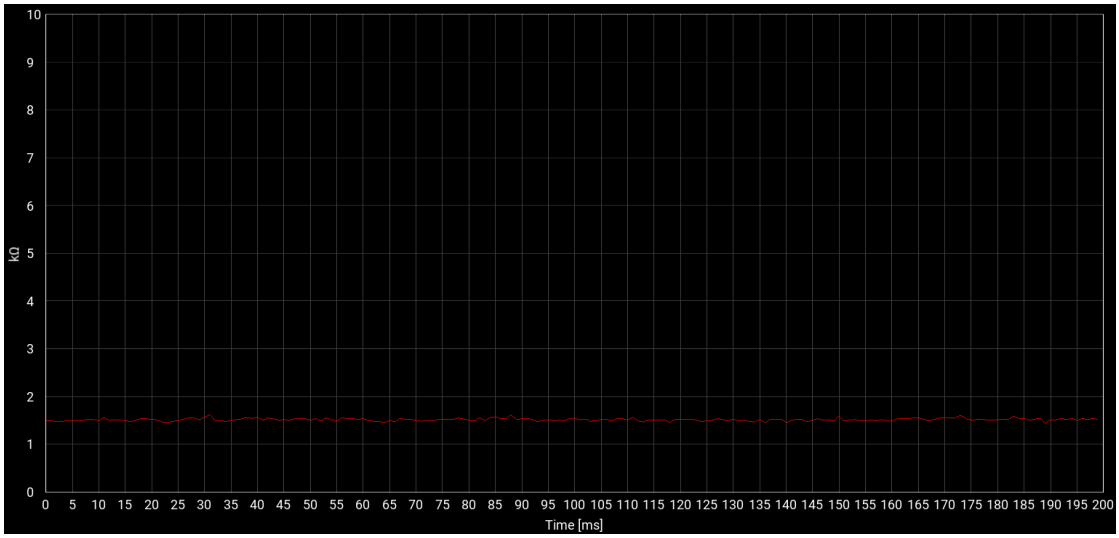


Figure 5.30: Real-Time acquired impedance: $f_{DAC} = 1.0\ kHz$, $R = 1.2\ k\Omega$, $C_s = 10\ pF$

The voltage drop across the electrode under test was also acquired using an oscilloscope, as described previously, through the instrumentation amplifier INA126. The amplifier gain was set to $G = 541$. The expected peak-to-peak voltage is $92.4\ \mu V$.
As shown in Fig. 5.31, the acquired waveform is modulated by a $50\ Hz$ component. This effect is mainly due to the very small amplitude of the acquired signal, which makes it more susceptible to power-line interference. By zooming in on the waveform, the amplified peak-to-peak electrode voltage is approximately $50.0\ mV$. Therefore, the voltage before amplification can be estimated as:

$$V_{pp} = \frac{V_{pp_{ampl}}}{G} = \frac{50.0\ mV}{541} = 92.4\ \mu V \tag{5.22}$$

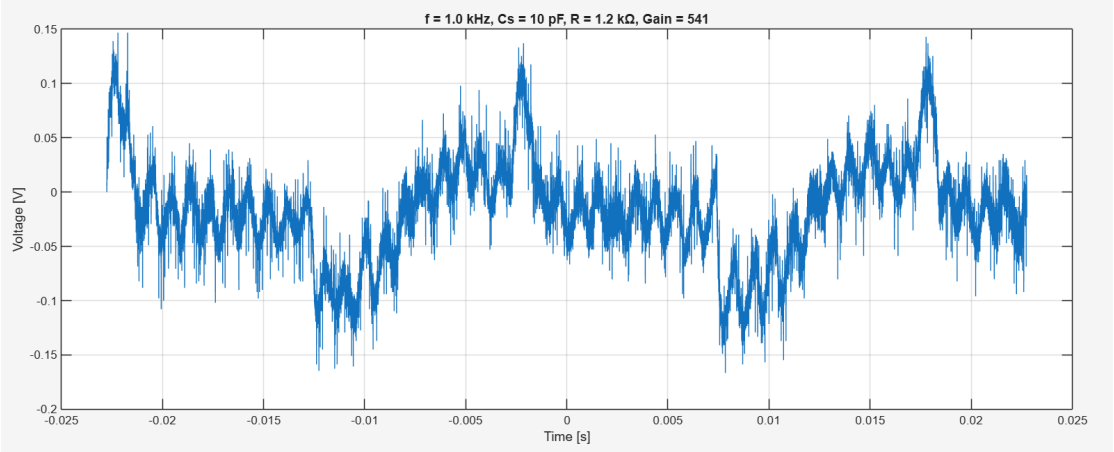


Figure 5.31: Acquired voltage from the oscilloscope: $f_{DAC} = 1.0\ kHz$, $R = 1.2\ k\Omega$, $C_s = 10\ pF$

The peak-to-peak voltage amplitude corresponds to the expected value.

The test was repeated by replacing the electrode of 1.2 $k\Omega$ nominal impedance with an electrode of 10.0 $k\Omega$ nominal impedance. The DAC waveform frequency was kept at 1.0 kHz. The impedance was then estimated by varying the series capacitor C_s.
The acquired voltage signals were filtered during the post-processing stage.
The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.32 and Fig. 5.33, respectively.

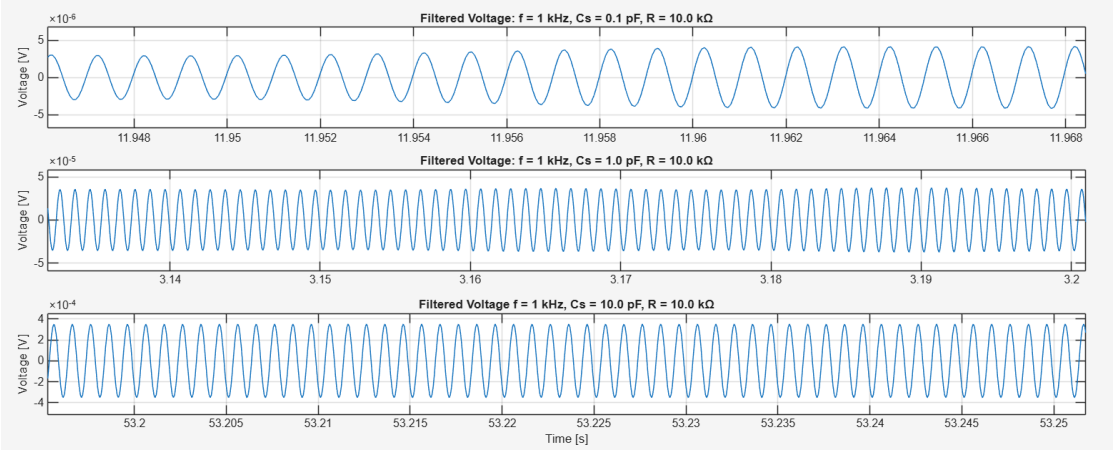


Figure 5.32: Acquired filtered voltage: $f_{DAC} = 1.0\ kHz$, $R = 10.0\ k\Omega$

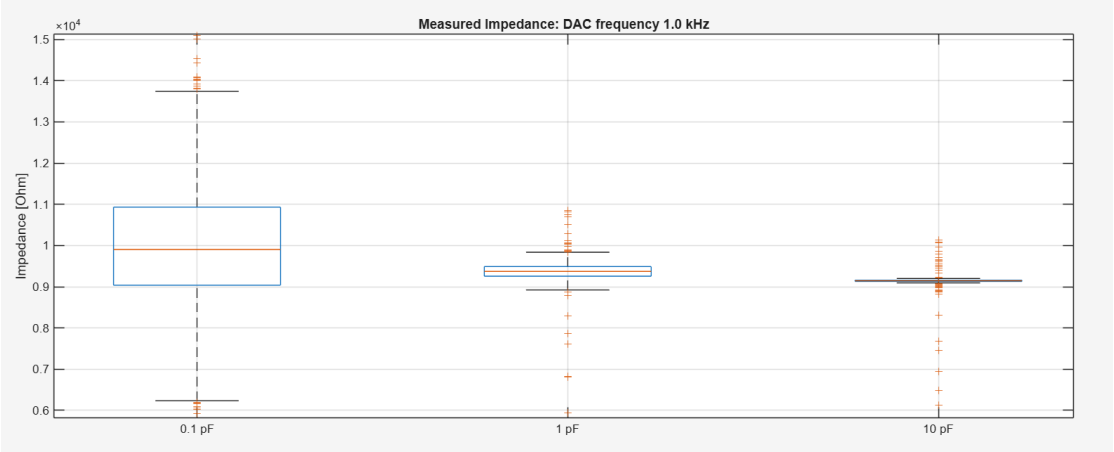


Figure 5.33: Measured Impedance: $f_{DAC} = 1.0\ kHz$, $R = 10.0\ k\Omega$

Table 5.8 reports the expected voltage drop across the electrode together with the corresponding measured values.
The measured peak voltages match the expected values for each selected C_s.
The median impedance values and the corresponding interquartile ranges (IQRs) obtained for the different C_s values closely match the expected impedance.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.385 nA	3.85 nA	38.5 nA
Expected DAC voltage V_p	3.85 μV	38.5 μV	0.385 mV
Measured DAC voltage V_p	3.81 μV	36.07 μV	0.352 mV
Impedance Median	9,90 $k\Omega$	9.37 $k\Omega$	9.14 $k\Omega$
Impedance IQR	1.90 $k\Omega$	233.17 Ω	29.07 Ω

Table 5.8: Measured quantities: $R = 10.0\ k\Omega$, $f_{DAC} = 1.0\ kHz$

The Real-Time acquired impedance is shown in Fig. 5.34. The real-time estimation is close to the expected value.

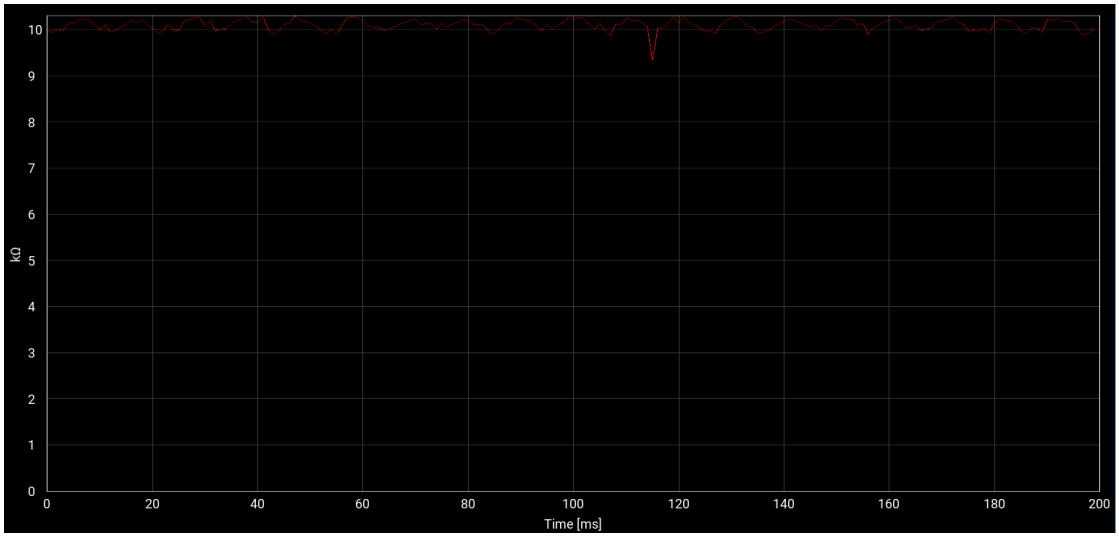


Figure 5.34: Real-Time acquired impedance: $f_{DAC} = 1.0\ kHz$, $R = 10.0\ k\Omega$, $C_s = 10\ pF$

The test was repeated by replacing the electrode of 10.0 $k\Omega$ nominal impedance with an electrode of 110.0 $k\Omega$ nominal impedance. The DAC waveform frequency was kept at 1.0 kHz. The impedance was then estimated by varying the series capacitor C_s.
The acquired voltage signals were filtered during the post-processing stage.
The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.35 and Fig. 5.36, respectively.

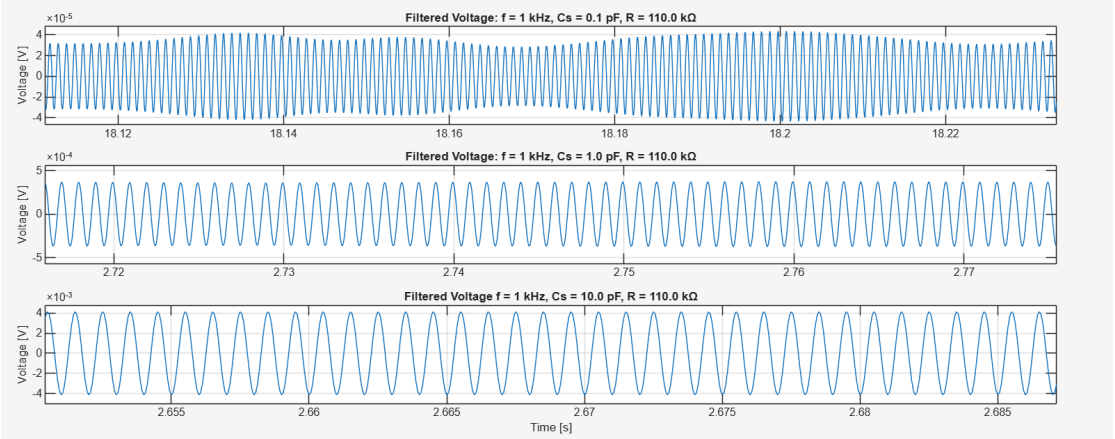


Figure 5.35: Acquired filtered voltage: $f_{DAC} = 1.0\ kHz$, $R = 110.0\ k\Omega$

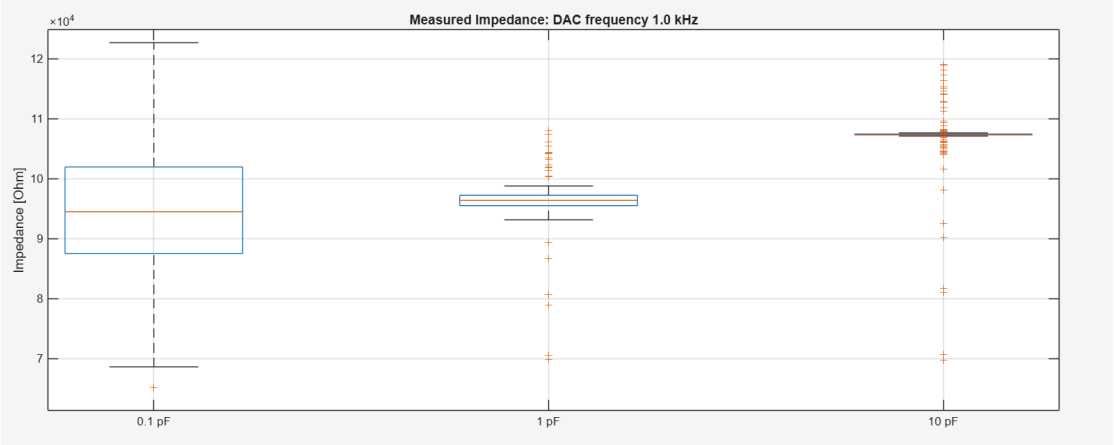


Figure 5.36: Measured Impedance: $f_{DAC} = 1.0\ kHz$, $R = 110.0\ k\Omega$

Table 5.9 reports the expected voltage drop across the electrode together with the corresponding measured values.
The measured peak voltages match the expected values for each selected C_s. Moreover, both the median impedance and the interquartile range (IQR) indicate excellent measurement reliability for all C_s values tested.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.385 nA	3.85 nA	38.5 nA
Expected DAC voltage V_p	42.35 μV	0.423 mV	4.23 mV
Measured DAC voltage V_p	36.36 μV	0.370 mV	4.1 mV
Impedance Median	94.49 $k\Omega$	96.39 $k\Omega$	107.39 $k\Omega$
Impedance IQR	14.39 $k\Omega$	1.72 $k\Omega$	131.54 Ω

Table 5.9: Measured quantities: $R = 110.0\ k\Omega$, $f_{DAC} = 1.0\ kHz$

Finally, the test was repeated by replacing the electrode with a nominal impedance of 110.0 $k\Omega$ with an electrode having a nominal impedance of 1.0 $M\Omega$. The DAC waveform frequency was kept at 1.0 kHz. The impedance was then estimated by varying the series capacitor C_s. In this case, the series capacitor was varied from $C_s = 0.1\ pF$ to $C_s = 1.0\ pF$, since the expected voltage for $C_s = 10.0\ pF$ would exceed the maximum input range of the amplifiers, resulting in an Out Of Range (OOR) condition.
The filtered voltages and the corresponding estimated impedances are shown in Fig. 5.37 and Fig. 5.38, respectively.

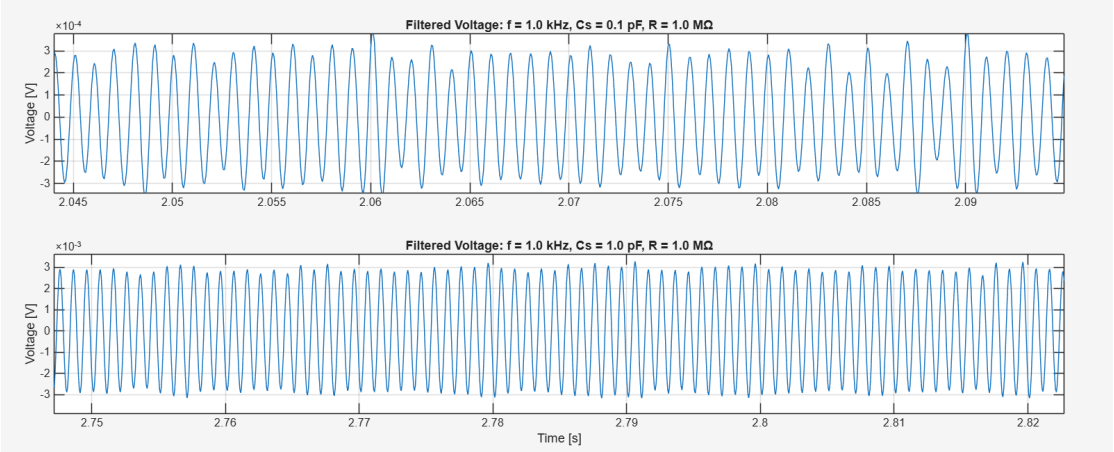


Figure 5.37: Acquired voltage: $f_{DAC} = 1.0\ kHz$, $R = 1.0\ M\Omega$

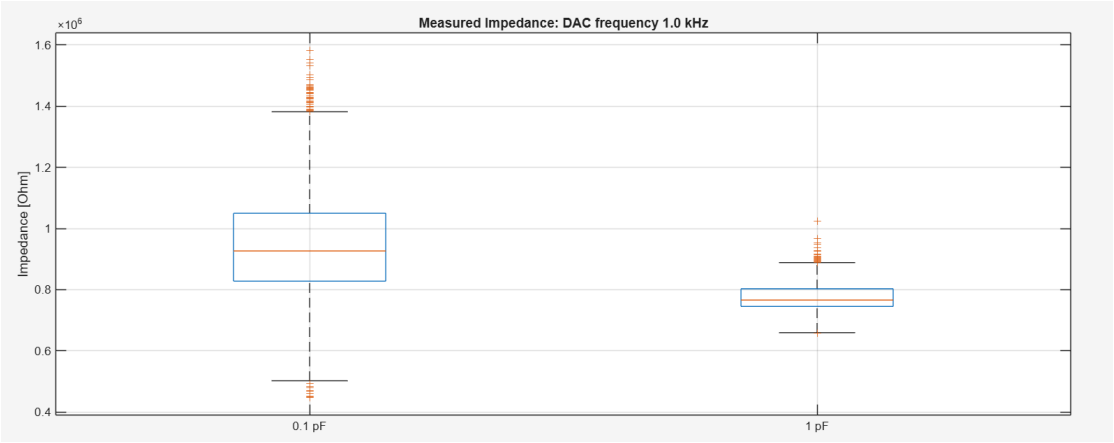


Figure 5.38: Measured Impedance: $f_{DAC} = 1.0\ kHz$, $R = 1.0\ M\Omega$

Table 5.10 reports the expected voltage drop across the electrode together with the corresponding measured values. The median impedance values and the corresponding IQR values are reasonably consistent with the expected impedance.

C_s	0.1 pF	1.0 pF	10 pF
DAC current I_p	0.385 nA	3.85 nA	38.5 nA
Expected DAC voltage V_p	0.385 mV	3.85 mV	38.5 mV
Measured DAC voltage V_p	0.356 mV	2.9 mV	OOR
Impedance Median	926.63 $K\Omega$	766.20 $k\Omega$	OOR
Impedance IQR	221.91 $k\Omega$	57.51 $k\Omega$	OOR

Table 5.10: Measured quantities: $R = 1.0\ M\Omega$, $f_{DAC} = 1.0\ kHz$

5.2.8 Impedance Test on Different Channels

The impedance was measured on different channels using electrodes with different impedance values. The impedance values measured with the multimeter are reported in Table 6.17. The DAC waveform frequency was set to 1.2 kHz.

Channel	16	17	18	19
R	1.202 $k\Omega$	11.19 $k\Omega$	102.4 $k\Omega$	1.010 $M\Omega$

Table 5.11: Impedance values measured with the multimeter for the tested channels

Channel 16

The test has been performed with $C_s = 10.0\ pF$. The filtered voltage and the corresponding estimated impedance for the Channel 16 are shown in Fig. 5.39 and Fig. 5.40, respectively.

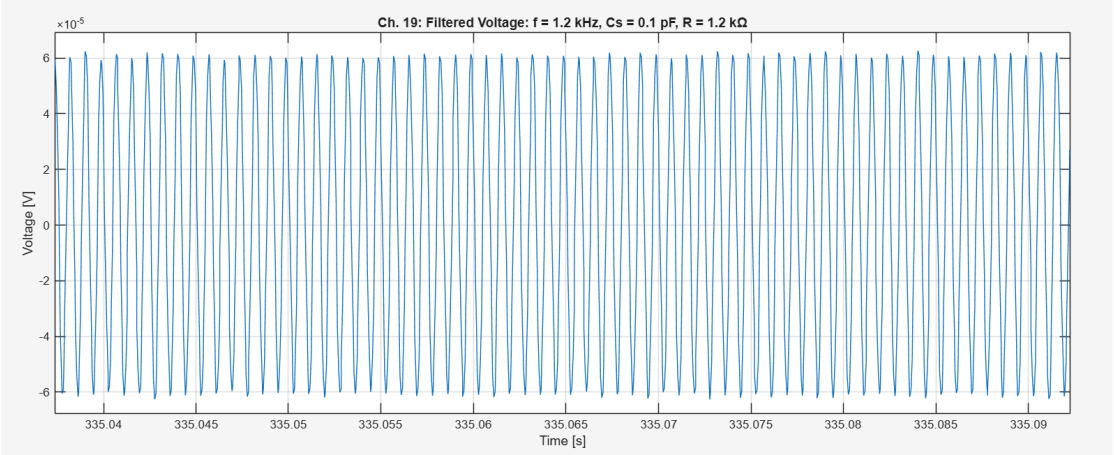


Figure 5.39: Acquired voltage on Channel 16: $f_{DAC} = 1.2\ kHz$, $R = 1.2\ k\Omega$

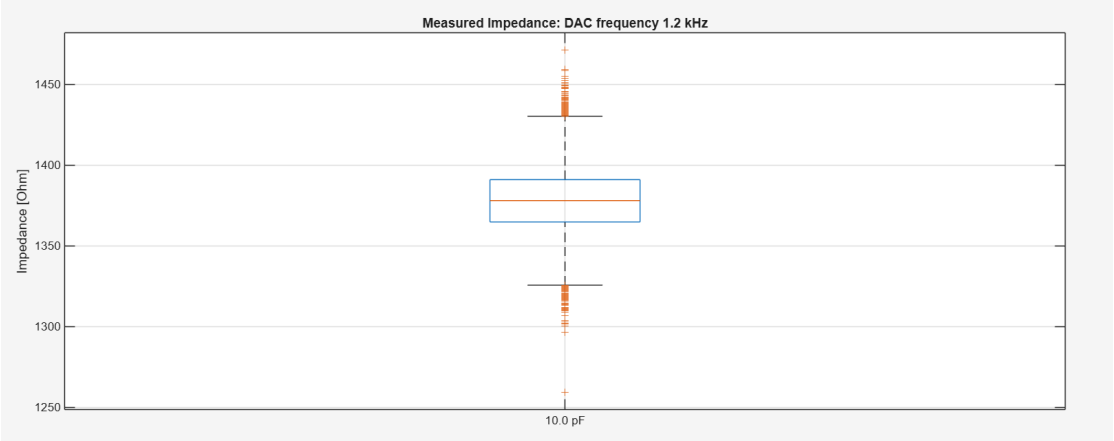


Figure 5.40: Measured Impedance on Channel 16: $f_{DAC} = 1.2\ kHz$, $R = 1.2\ k\Omega$

Table 5.12 reports the expected voltage drop across the electrode together with the corresponding measured values. The median impedance values and the corresponding IQR values are reasonably consistent with the expected impedance.

C_s	10 pF
DAC current I_p	44.25 nA
Expected DAC voltage V_p	53.18 μV
Measured DAC voltage V_p	63.64 μV
Impedance Median	1.36 $k\Omega$
Impedance IQR	26.19 Ω

Table 5.12: Measured quantities: $R = 1.2\ k\Omega$, $f_{DAC} = 1.2\ kHz$

Channel 17

The test has been performed with $C_s = 10.0\ pF$. The filtered voltage and the corresponding estimated impedance for the Channel 17 are shown in Fig. 5.41 and Fig. 5.42, respectively.

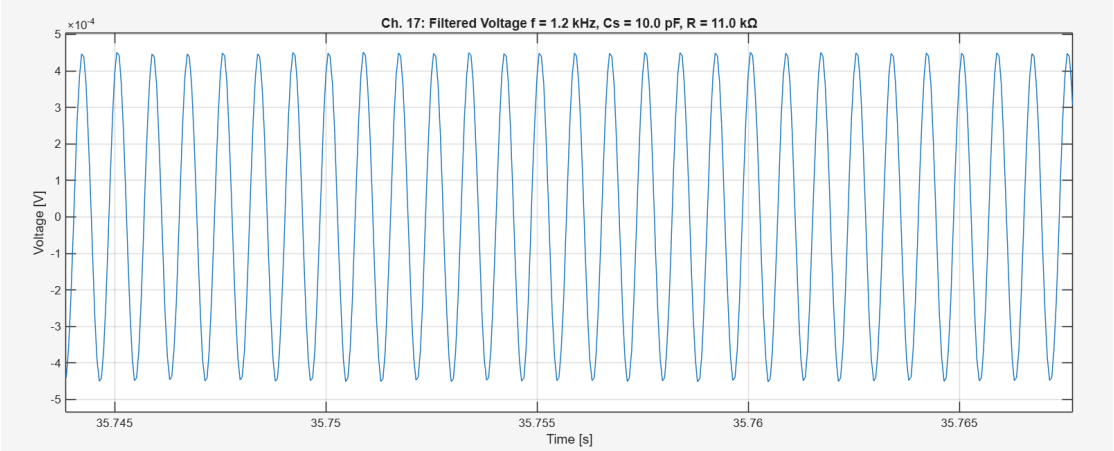


Figure 5.41: Acquired voltage on Channel 17: $f_{DAC} = 1.2\ kHz$, $R = 11.19\ k\Omega$

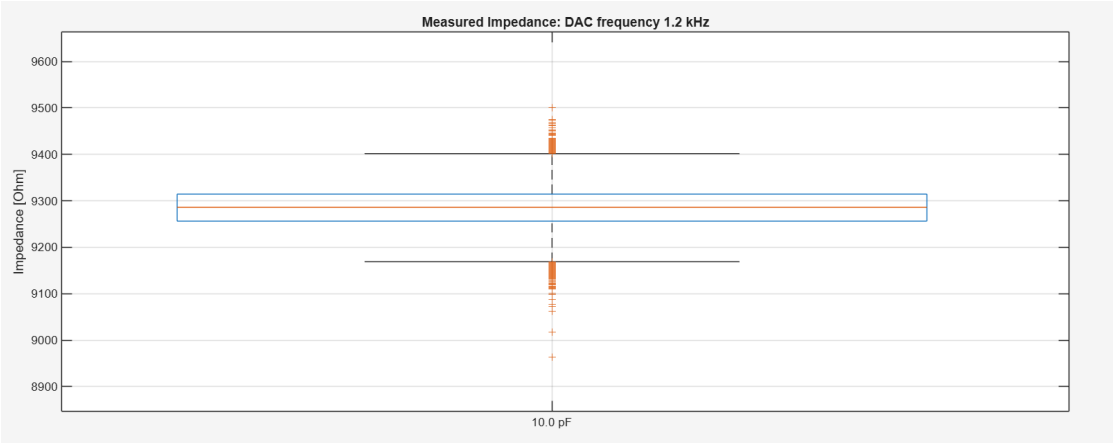


Figure 5.42: Measured Impedance on Channel 17: $f_{DAC} = 1.2\ kHz$, $R = 1.2\ k\Omega$

Table 5.13 reports the expected voltage drop across the electrode together with the corresponding measured values. The median impedance values and the corresponding IQR values are reasonably consistent with the expected impedance.

C_s	10 pF
DAC current I_p	44.25 nA
Expected DAC voltage V_p	0.486 mV
Measured DAC voltage V_p	0.428 mV
Impedance Median	9.29 $K\Omega$
Impedance IQR	58.24 Ω

Table 5.13: Measured quantities: $R = 11.19\ k\Omega$, $f_{DAC} = 1.2\ kHz$

The Real-Time acquired impedance is shown in Fig. 5.43. The real-time estimation is close to the expected value.

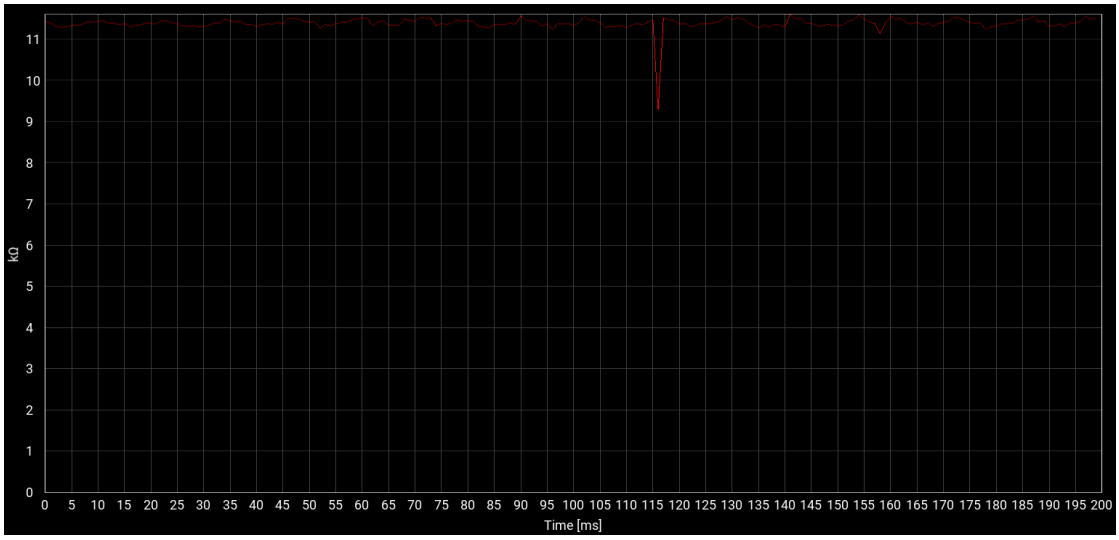


Figure 5.43: Real-Time acquired impedance: $f_{DAC} = 1.2\ kHz$, $R = 11.19\ k\Omega$, $C_s = 10\ pF$

Channel 18

The test has been performed with $C_s = 10.0\ pF$. The filtered voltage and the corresponding estimated impedance for the Channel 18 are shown in Fig. 5.44 and Fig. 5.45, respectively.

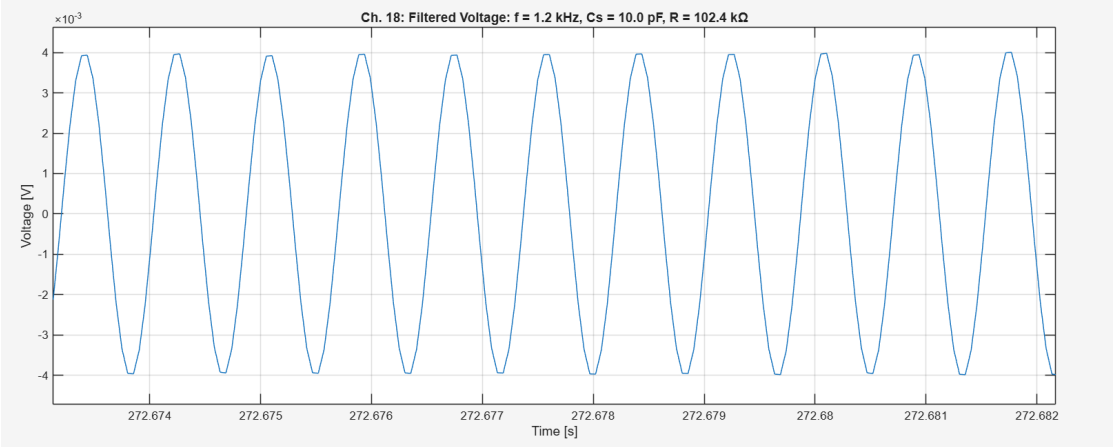


Figure 5.44: Acquired voltage on Channel 18: $f_{DAC} = 1.2\ kHz$, $R = 102.4\ k\Omega$

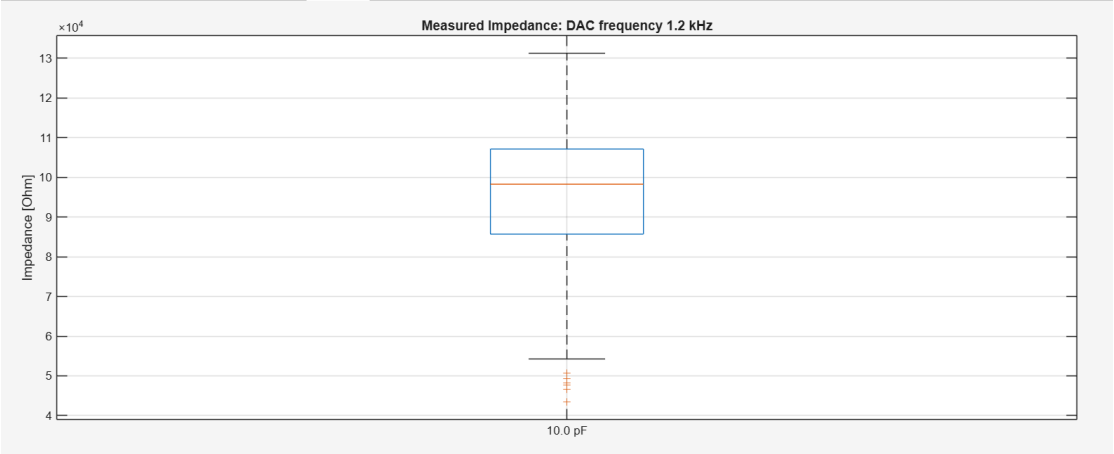


Figure 5.45: Measured Impedance on Channel 18: $f_{DAC} = 1.2\ kHz$, $R = 102.4\ k\Omega$

Table 5.14 reports the expected voltage drop across the electrode together with the corresponding measured values. The median impedance values and the corresponding IQR values are reasonably consistent with the expected impedance.

C_s	10 pF
DAC current I_p	44.25 nA
Expected DAC voltage V_p	4.53 mV
Measured DAC voltage V_p	4.4 mV
Impedance Median	98.29 $K\Omega$
Impedance IQR	21.44 $k\Omega$

Table 5.14: Measured quantities: $R = 102.4\ k\Omega$, $f_{DAC} = 1.2\ kHz$

The Real-Time acquired impedance is shown in Fig. 5.49. The real-time estimation is close to the expected value.

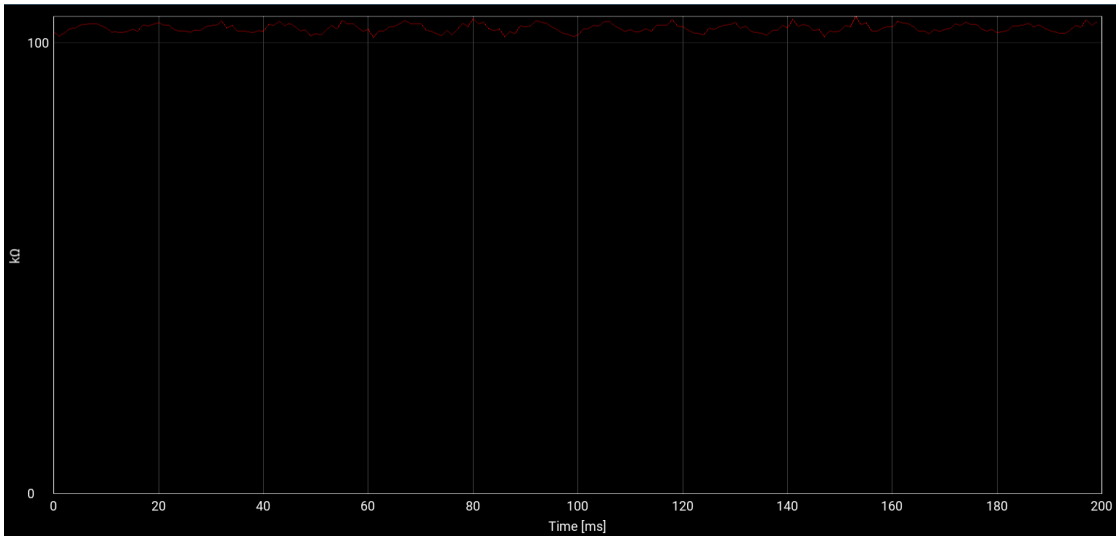


Figure 5.46: Real-Time acquired impedance: $f_{DAC} = 1.2\ kHz$, $R = 102.4\ k\Omega$, $C_s = 10\ pF$

Channel 19

The test has been performed with $C_s = 0.1\ pF$. The filtered voltage and the corresponding estimated impedance for the Channel 19 are shown in Fig. 5.47 and Fig. 5.48, respectively.

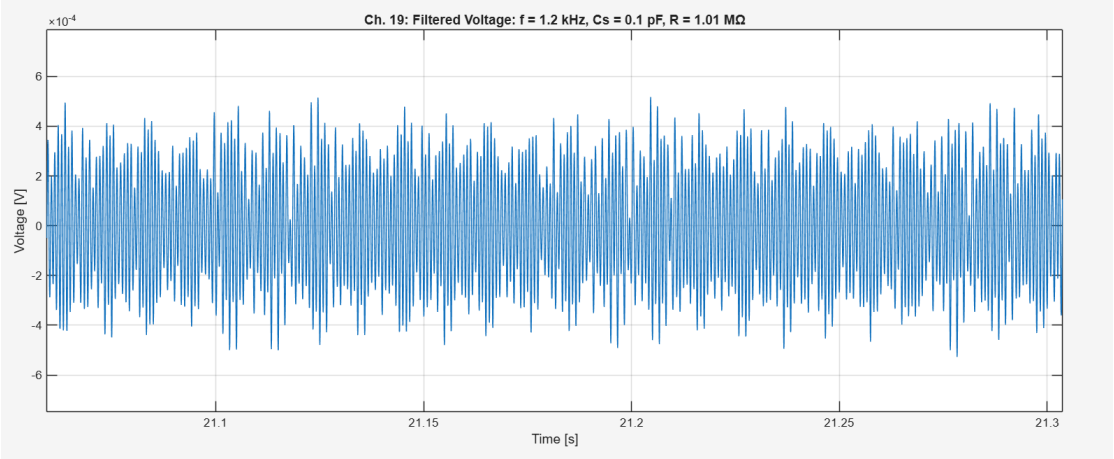


Figure 5.47: Acquired voltage on Channel 19: $f_{DAC} = 1.2\ kHz$, $R = 1.0\ M\Omega$

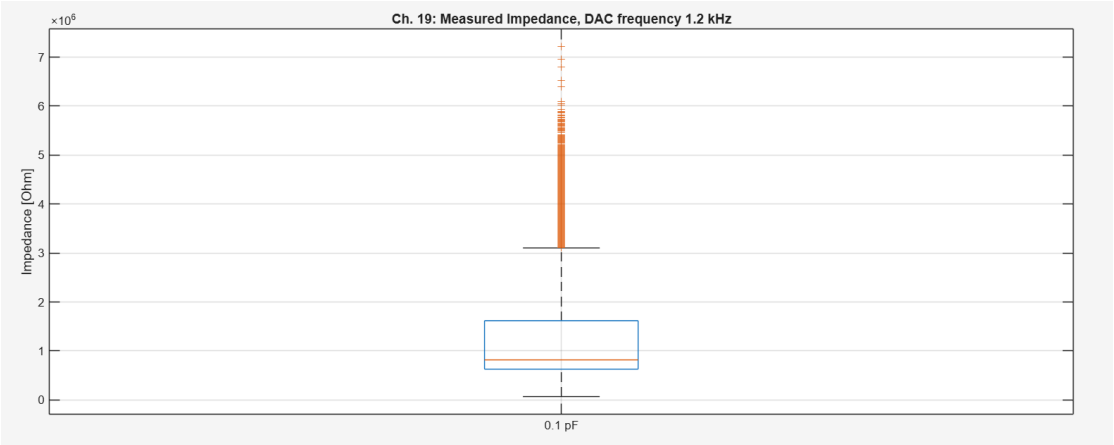


Figure 5.48: Measured Impedance on Channel 19: $f_{DAC} = 1.2\ kHz$, $R = 1.0\ M\Omega$

Table 5.15 reports the expected voltage drop across the electrode together with the corresponding measured values. The median impedance values and the corresponding IQR values are reasonably consistent with the expected impedance.

C_s	0.1 pF
DAC current I_p	0.443 nA
Expected DAC voltage V_p	0.443 mV
Measured DAC voltage V_p	0.378 mV
Impedance Median	818.41 $k\Omega$
Impedance IQR	990.26 $k\Omega$

Table 5.15: Measured quantities: $R = 1.0\ M\Omega$, $f_{DAC} = 1.2\ kHz$

The Real-Time acquired impedance is shown in Fig. 5.49. The real-time estimation is close to the expected value.

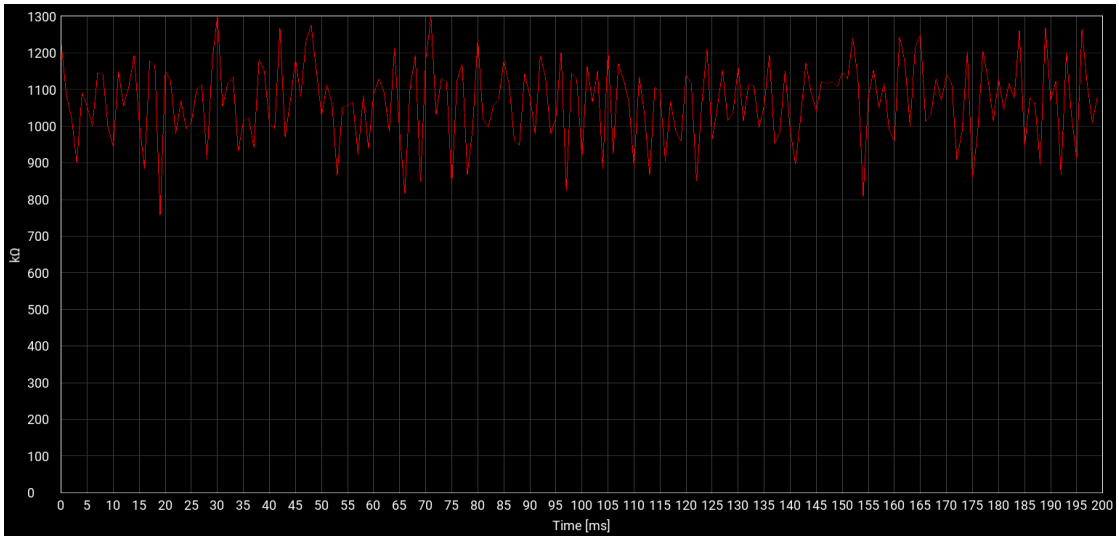


Figure 5.49: Real-Time acquired impedance: $f_{DAC} = 1.2\ kHz$, $R = 1.0\ M\Omega$, $C_s = 0.1\ pF$

Chapter 6

Conclusions and Future Perspectives

6.1 Data Acquisition

6.1.1 Single-Channel Acquisition: Effect of the Band-Pass Filter Lower Cut-Off Frequency

The effect of the lower cutoff frequency (f_L) was analyzed by acquiring a sinusoidal signal with frequency $f_{\sin} = 1\,kHz$ and amplitude $2\,mV_{pp}$. The parameter f_L was varied from $0.1\,Hz$ to $100\,Hz$, while keeping the upper cutoff frequency fixed.

Figure 6.2 shows the corresponding power spectral density (PSD).

Since the fundamental component is located at $1\,kHz$, well above the tested values of f_L, its amplitude remains essentially unchanged. This confirms that the filter effectively suppresses low-frequency components without affecting the fundamental tone.

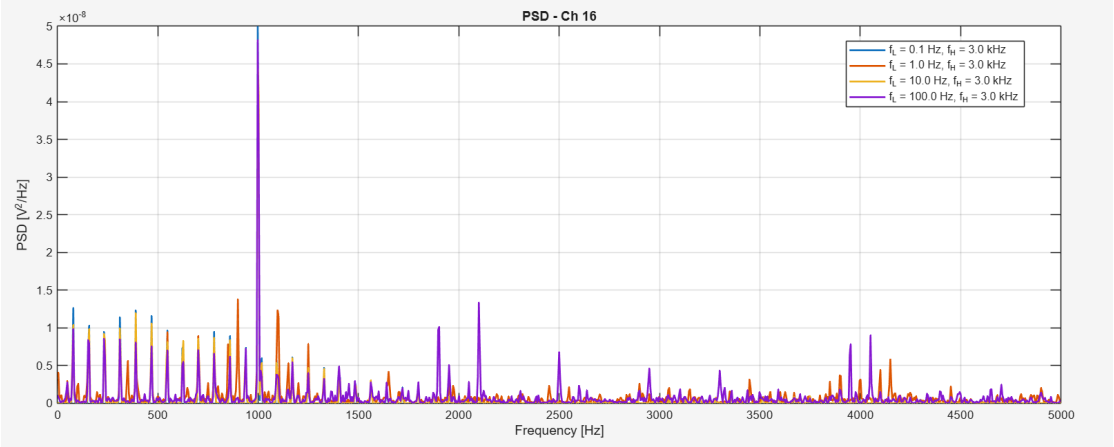


Figure 6.1: Power spectral density (PSD) variation while increasing the lower cut-off frequency f_L.

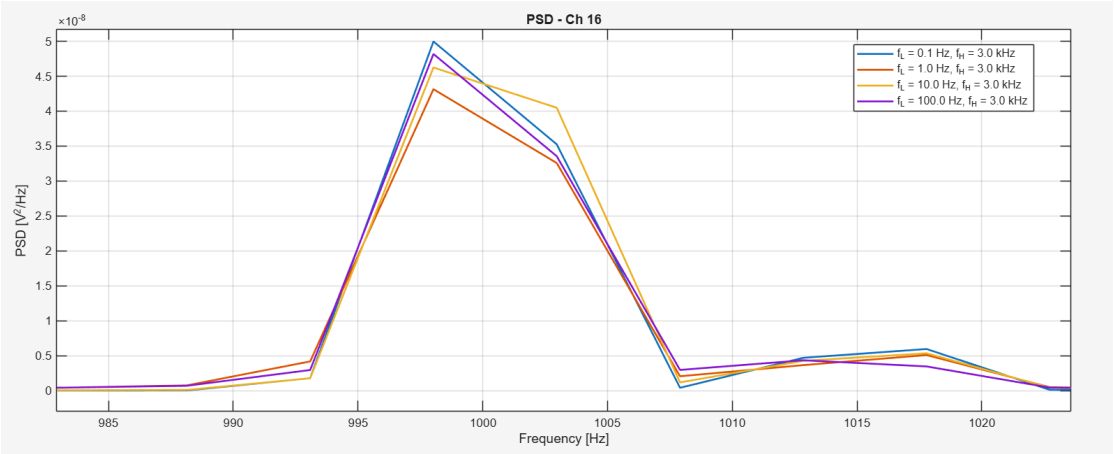


Figure 6.2: Power spectral density (PSD) variation while increasing the lower cut-off frequency f_L.

6.1.2 Single-Channel Acquisition: Effect of the Band-Pass Filter Upper Cut-Off Frequency

The influence of the upper cutoff frequency (f_H) was evaluated by progressively reducing it while keeping the lower cutoff frequency fixed.

Figure 6.4 shows the resulting power spectral density. As f_H decreases from $3\,kHz$ to $750\,Hz$ and approaches the signal frequency ($1\,kHz$), the amplitude of the fundamental component is progressively attenuated due to the low-pass behavior

of the filter.

This result highlights the importance of selecting an upper cutoff frequency sufficiently higher than the signal frequency in order to avoid amplitude attenuation and preserve the correct representation of the signal.

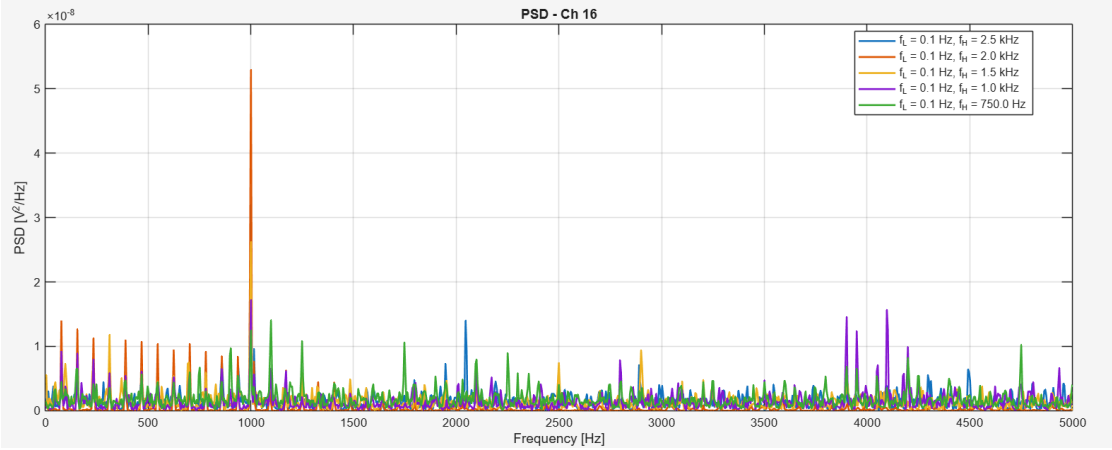


Figure 6.3: Power spectral density (PSD) variation while decreasing the upper cut-off frequency f_H.

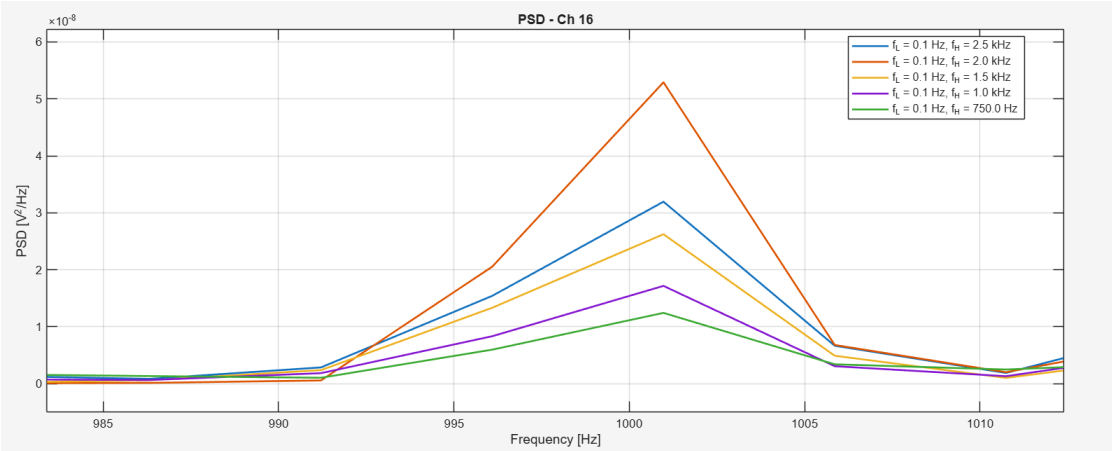


Figure 6.4: Power spectral density (PSD) variation while decreasing the upper cut-off frequency f_H.

6.1.3 Single-Channel Acquisition: Frequency and Amplitude Variation

The performed tests confirm the correct operation of the acquisition chain when varying both the amplitude and the frequency of the input sinusoidal signal. In particular, the system accurately reproduces the spectral characteristics of the input signal.

Fig. 6.5 shows the Power Spectral Density (PSD) of the acquired signal for different input frequencies. As the input frequency varies, the spectral peak shifts accordingly while maintaining the expected spectral shape, demonstrating that the system correctly tracks the frequency variation.

Similarly, Fig. 6.6 reports the PSD obtained for different input amplitudes at a fixed frequency of 1 kHz. The results show that the amplitude of the spectral peak scales consistently with the input signal amplitude, confirming the correct amplitude response of the acquisition system within the considered bandwidth ($f_L = 0.1\,\text{Hz}$, $f_H = 3\,\text{kHz}$).

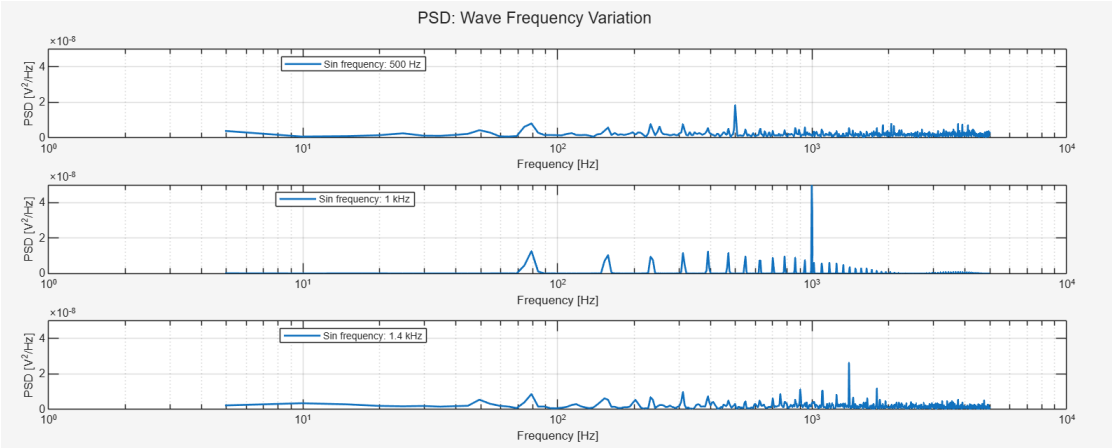


Figure 6.5: Power Spectral Density of the sinusoidal signal with amplitude $2\,\text{mV}_{\text{pp}}$, $f_L = 0.1\,\text{Hz}$ and $f_H = 3.0\,\text{kHz}$, for different input frequencies.

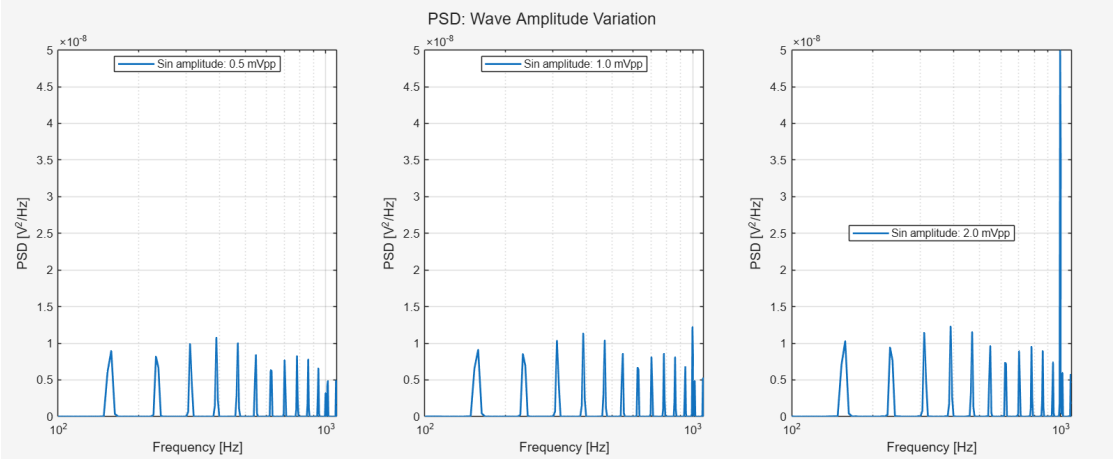


Figure 6.6: Power Spectral Density of the sinusoidal signal with frequency 1 kHz, $f_L = 0.1$ Hz and $f_H = 3.0$ kHz, for different input amplitudes.

6.1.4 Multi-Channel Acquisition

The acquisition chain was successfully tested on multiple channels. In the current implementation, the system allows the simultaneous acquisition of four channels. This limitation is mainly due to the maximum baud rate of the available UART interface, as discussed in the previous sections.

The UART baud rate is 921600 bit/s. Each CONVERT operation produces a 16-bit result. Considering a sampling frequency of 10 kHz per channel, the maximum number of channels that can be transmitted through the UART link can be estimated as:

$$\frac{N \cdot 16 \text{ bit}}{100\ \mu s} < 921600 \text{ bit/s} \tag{6.1}$$

which leads to

$$N < 5.76 \tag{6.2}$$

Therefore, the maximum number of channels that can be reliably acquired and transmitted simultaneously is four.

The acquired waveforms and their corresponding Power Spectral Density (PSD) are shown in Fig. 6.7 and Fig. 6.8, respectively.

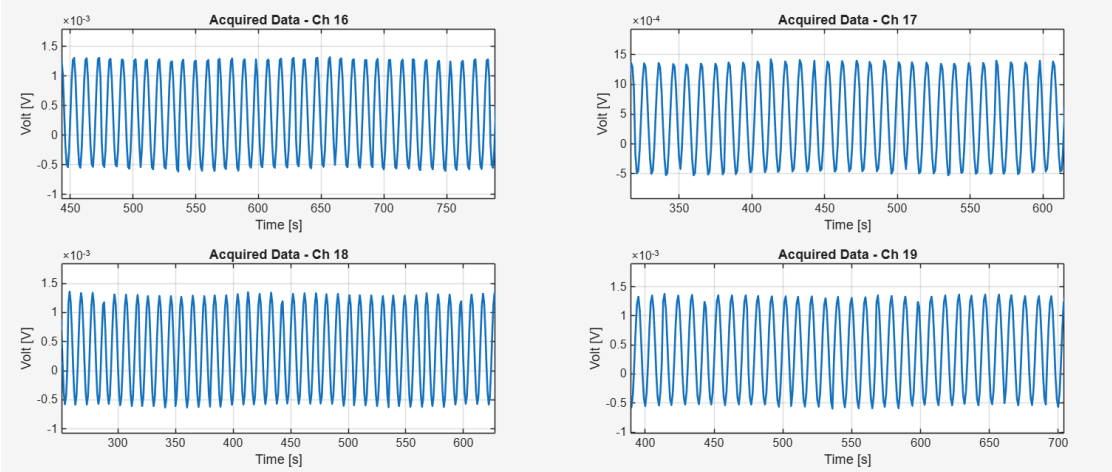


Figure 6.7: Offline acquired waves from channels 16, 17, 18 and 19.

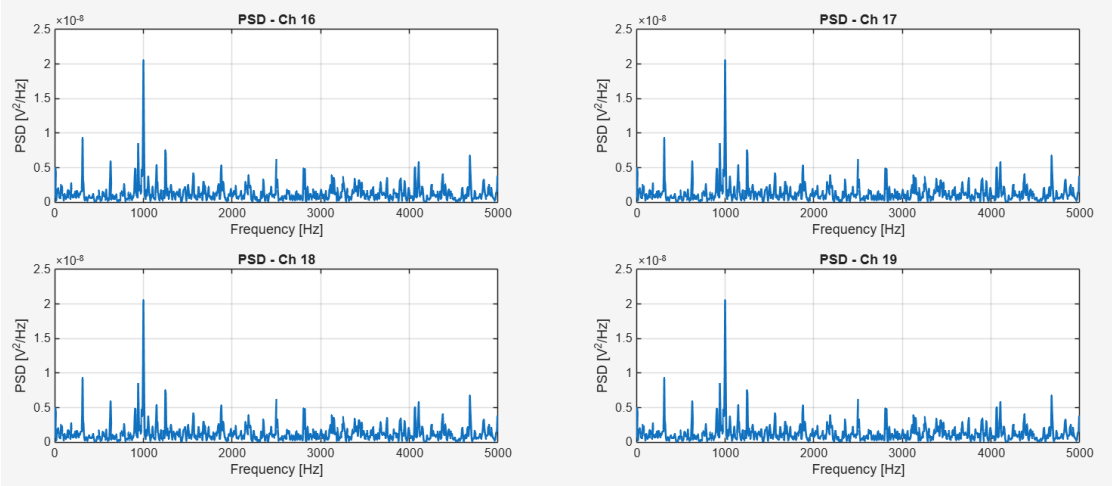


Figure 6.8: Power Spectral Density of the signals acquired from channels 16, 17, 18 and 19.

6.2 Impedance Test

Impedance measurements were performed on multiple channels using electrodes with different nominal impedance values. In this section, the accuracy of the impedance estimation for each tested electrode is discussed.

6.2.1 Electrode impedance: case $R = 1.2\ k\Omega$

The electrode with a nominal impedance of 1.2 $k\Omega$ was tested under different DAC excitation frequencies. Due to the relatively low impedance value, the expected voltage drop across the electrode is very small, as reported in Table 6.1.

C_s	0.1 pF	1.0 pF	10 pF
Expected voltage (900 Hz)	0.415 μV	4.15 μV	41.5 μV
Expected voltage (1 kHz)	0.462 μV	4.62 μV	46.2 μV
Expected voltage (1.2 kHz)	0.532 μV	5.32 μV	53.2 μV

Table 6.1: Expected voltages on the electrode under test: $R = 1.2\ k\Omega$

The amplifier input-referred noise is 2.4 μV_{rms}. Therefore, the Signal-to-Noise Ratio (SNR) for $C_s = 0.1\ pF$ is very low, as reported in the Table 6.2.

C_s	0.1 pF	1.0 pF	10 pF
SNR (900 Hz)	0.12	1.2	12.0
SNR (1 kHz)	0.14	1.4	14.0
SNR (1.2 kHz)	0.16	1.6	16.0

Table 6.2: SNR for the electrode under test: $R = 1.2\ k\Omega$

The expected signal amplitudes are therefore below the noise floor when $C_s = 0.1\ pF$ is selected. Consequently, the impedance estimation exhibits a significant error under this condition.

The acquired voltage drops and the corresponding measured impedances are reported in Table 6.3 and Table 6.4, respectively.

C_s	0.1 pF	1.0 pF	10 pF
Measured voltage (900 Hz)	3.45 μV	5.36 μV	49.33 μV
Measured voltage (1 kHz)	1.20 μV	5.95 μV	54.91 μV
Measured voltage (1.2 kHz)	1.39 μV	7.13 μV	64.36 μV

Table 6.3: Measured voltages on the electrode under test: $R = 1.2\ k\Omega$

C_s	0.1 pF	1.0 pF	10 pF
Z median (900 Hz)	9.96 $k\Omega$	1.54 $k\Omega$	1.42 $k\Omega$
Z median (1 kHz)	3.12 $k\Omega$	1.54 $k\Omega$	1.43 $k\Omega$
Z median (1.2 kHz)	3.00 $k\Omega$	1.54 $k\Omega$	1.39 $k\Omega$

Table 6.4: Measured impedance on the electrode under test: $R = 1.2\ k\Omega$

On the contrary, when $C_s = 1.0\ pF$ and $C_s = 10.0\ pF$ are used, the measured impedance values are much closer to the expected impedance of 1.2 $k\Omega$. In these cases, the larger voltage drop across the electrode leads to a higher SNR and therefore to a more reliable impedance estimation. Overall, the impedance measurement system is able to correctly estimate electrode impedances around 1.2 $k\Omega$ when appropriate values of C_s are selected. This allows the verification of electrode quality, since properly functioning electrodes typically exhibit impedances on the order of 1 $k\Omega$.

6.2.2 Electrode impedance: case $R = 10.0\ k\Omega$

The electrode with a nominal impedance of 10.0 $k\Omega$ was tested under different DAC excitation frequencies. The expected voltage values are reported in Table 6.5.

C_s	0.1 pF	1.0 pF	10 pF
Expected voltage (900 Hz)	3.46 μV	34.6 μV	0.346 mV
Expected voltage (1 kHz)	3.85 μV	38.5 μV	0.385 mV
Expected voltage (1.2 kHz)	4.43 μV	44.3 μV	0.443 mV

Table 6.5: Expected voltages on the electrode under test: $R = 10.0\ k\Omega$

The corresponding Signal-to-Noise Ratio (SNR) values are reported in Table 6.6. In this case, the SNR values are sufficiently high for all the selected C_s values, ensuring reliable impedance estimation.

C_s	0.1 pF	1.0 pF	10 pF
SNR (900 Hz)	1.01	10.1	101.0
SNR (1 kHz)	1.13	11.3	113.0
SNR (1.2 kHz)	1.31	13.1	131.0

Table 6.6: SNR for the electrode under test: $R = 10.0\ k\Omega$

The acquired voltage drops and the corresponding measured impedances are reported in Table 6.7 and Table 6.8, respectively.

C_s	0.1 pF	1.0 pF	10 pF
Measured voltage (900 Hz)	3.70 μV	35.47 μV	0.347 mV
Measured voltage (1 kHz)	3.81 μV	36.07 μV	0.352 mV
Measured voltage (1.2 kHz)	5.76 μV	43.83 μV	0.429 mV

Table 6.7: Measured voltages on the electrode under test: $R = 10.0\ k\Omega$

C_s	0.1 pF	1.0 pF	10 pF
Z median (900 Hz)	10.68 $k\Omega$	10.24 $k\Omega$	10.03 $k\Omega$
Z median (1 kHz)	9.90 $k\Omega$	9.37 $k\Omega$	9.14 $k\Omega$
Z median (1.2 kHz)	12.47 $k\Omega$	9.49 $k\Omega$	9.28 $k\Omega$

Table 6.8: Measured impedance on the electrode under test: $R = 10.0\ k\Omega$

The impedance of the electrode with a nominal value of 10 $k\Omega$ was accurately estimated for all the selected values of C_s. In this case, the voltage drop across the electrode is sufficiently high to guarantee an adequate signal-to-noise ratio. Consequently, the impedance estimation remains stable and consistent across the different measurement configurations.

6.2.3 Electrode impedance: case $R = 110.0\ k\Omega$

The electrode with a nominal impedance of 110.0 $k\Omega$ was tested under different DAC excitation frequencies. The expected voltage values are reported in Table 6.9.

C_s	0.1 pF	1.0 pF	10 pF
Expected voltage (900 Hz)	38.06 μV	0.381 mV	3.81 mV
Expected voltage (1 kHz)	42.35 μV	0.423 mV	4.23 mV
Expected voltage (1.2 kHz)	48.73 μV	0.487 mV	4.87 mV

Table 6.9: Expected voltages on the electrode under test: $R = 110.0\ k\Omega$

The corresponding Signal-to-Noise Ratio (SNR) values are reported in Table 6.10. In this case, the SNR values are high for all the selected C_s values, ensuring reliable impedance estimation.

C_s	0.1 pF	1.0 pF	10 pF
SNR (900 Hz)	11.21	112.1	1121.0
SNR (1 kHz)	12.47	124.7	1247.0
SNR (1.2 kHz)	14.35	143.5	1435.0

Table 6.10: SNR for the electrode under test: $R = 110.0\ k\Omega$

The acquired voltage drops and the corresponding measured impedances are reported in Table 6.11 and Table 6.12, respectively.

C_s	0.1 pF	1.0 pF	10 pF
Measured voltage (900 Hz)	31.54 μV	0.354 mV	4.2 mV
Measured voltage (1 kHz)	36.36 μV	0.370 mV	4.1 mV
Measured voltage (1.2 kHz)	50.29 μV	0.449 mV	4.5 mV

Table 6.11: Measured voltages on the electrode under test: $R = 110.0\ k\Omega$

C_s	0.1 pF	1.0 pF	10 pF
Z median (900 Hz)	91.07 $k\Omega$	102.33 $k\Omega$	118.96 $k\Omega$
Z median (1 kHz)	94.49 $k\Omega$	96.39 $k\Omega$	107.39 $k\Omega$
Z median (1.2 kHz)	108.90 $k\Omega$	97.34 $k\Omega$	96.48 $k\Omega$

Table 6.12: Measured impedance on the electrode under test: $R = 110.0\ k\Omega$

The impedance of the electrode with a nominal value of 110 $k\Omega$ was accurately estimated for all the selected values of C_s. In this impedance range, the voltage drop across the electrode is sufficiently large to ensure a high signal-to-noise ratio. As a result, the impedance estimation remains stable and consistent, as confirmed by the small interquartile range (IQR) values obtained for the different measurement configurations.

6.2.4 Electrode impedance: case $R = 1.0\ M\Omega$

The electrode with a nominal impedance of 1.0 $M\Omega$ was tested under different DAC excitation frequencies. The expected voltage values are reported in Table 6.13.

C_s	0.1 pF	1.0 pF
Expected voltage (900 Hz)	0.346 mV	3.46 mV
Expected voltage (1 kHz)	0.385 mV	3.85 mV
Expected voltage (1.2 kHz)	0.443 mV	4.43 mV

Table 6.13: Expected voltages on the electrode under test: $R = 1.0\ M\Omega$

The corresponding Signal-to-Noise Ratio (SNR) values are reported in Table 6.14. In this case, the SNR values are high for all the selected C_s values, ensuring reliable impedance estimation.

C_s	0.1 pF	1.0 pF
SNR (900 Hz)	102	1019
SNR (1 kHz)	113	1134
SNR (1.2 kHz)	130	1305

Table 6.14: SNR for the electrode under test: $R = 1.0\ M\Omega$

The acquired voltage drops and the corresponding measured impedances are reported in Table 6.15 and Table 6.16, respectively.

C_s	0.1 pF	1.0 pF
Measured voltage (900 Hz)	0.297 mV	3.3 mV
Measured voltage (1 kHz)	0.356 mV	2.9 mV
Measured voltage (1.2 kHz)	0.361 mV	3.5 mV

Table 6.15: Measured voltages on the electrode under test: $R = 1.0\ M\Omega$

C_s	0.1 pF	1.0 pF
Z median (900 Hz)	857.07 $k\Omega$	965.20 $k\Omega$
Z median (1 kHz)	926.63 $k\Omega$	766.20 $k\Omega$
Z median (1.2 kHz)	782.67 $k\Omega$	763.64 $k\Omega$

Table 6.16: Measured impedance on the electrode under test: $R = 1.0\ M\Omega$

In this impedance range, the voltage drop across the electrode is significantly larger, leading to a very high signal-to-noise ratio and consequently to a reliable impedance estimation for the selected values of C_s.

6.2.5 Different channels Impedance Test

The impedance tests on different electrodes have been performed successfully. The electrodes under test had the nominal impedances, measured with a multimeter, as reported in Table 6.17.

Channel	16	17	18	19
Nominal Impedance	1.202 $k\Omega$	11.19 $k\Omega$	102.4 $k\Omega$	1.010 $M\Omega$

Table 6.17: Impedance values measured with the multimeter for the tested channels

All measurements were performed using a DAC waveform frequency of 1.2 kHz. Table 6.18 reports the expected voltage across each electrode, the measured voltage, and the corresponding median impedance values.

Channel	16	17	18	19
Expected voltage	53.18 μV	0.486 mV	4.53 mV	0.443 mV
Measured voltage	63.64 μV	0.428 mV	4.40 mV	0.378 mV
Z median	1.36 $k\Omega$	9.29 $k\Omega$	98.29 $k\Omega$	818.41 $k\Omega$

Table 6.18: Impedance Test on different electrodes

The measured impedances on each channel are generally consistent with the expected values, confirming the effectiveness of the proposed measurement system across a wide range of electrode impedances.

In particular:

- For low-impedance electrodes (channel 16), the measurements are close to the nominal value, although small deviations may be present due to the low voltage amplitude approaching the amplifier noise floor.
- For medium-impedance electrodes (channels 17 and 18), the measured values show high accuracy, as the voltage drop across the electrodes is sufficiently large to ensure a high signal-to-noise ratio.
- For high-impedance electrodes (channel 19), the estimation remains reliable, although slight deviations from the nominal value may be caused by limitations in the excitation current or the amplifier input range.

Overall, these results demonstrate that the system is capable of accurately estimating electrode impedances across a broad range of values, from low- to high-impedance electrodes, highlighting its robustness and versatility for multi-channel

measurements.

6.2.6 Future Perspectives

This thesis project is part of the broader research project "Bladder Control". The final goal of this project is to replace open-loop neuroprosthetic devices for bladder control with a closed-loop system. Such a system requires reliable acquisition of electroneurographic (ENG) signals from the pudendal nerve. In order to ensure a reliable acquisition, it is necessary to verify the quality of the electrodes through impedance measurements. The ENG signals acquired in real time are then processed by a Machine Learning (ML) algorithm, which decodes the bladder state and discriminates among empty, full, and micturition conditions. Based on the detected bladder state, electrical stimulation of the pudendal nerve can be delivered when necessary.
This thesis focused on the development of the data acquisition system and on the validation of the electrode impedance measurement procedure. Future developments of the project include the integration of the already developed ML algorithm with the acquisition system, as well as the implementation of the neural stimulation module to enable the complete closed-loop control strategy.

Bibliography

[1] Edelle C. Field-Fote. *Spinal Cord Injury Rehabilitation*. McGraw-Hill Education, 2009 (cit. on p. 2).

[2] A. Giannotti et al. «Decoding bladder state from pudendal intraneural signals in pigs». In: *APL Bioengineering* 7.4 (Oct. 2023), p. 046101. DOI: 10.1063/5.0156484 (cit. on pp. 2, 18).

[3] Eric Peña and Mary Grace Legaspi. «UART: A hardware communication protocol understanding universal asynchronous receiver/transmitter». In: 54.4 (Dec. 2020) (cit. on pp. 5–9).

[4] R. Fukuma, T. Yanagisawa, S. Yorifuji, R. Kato, H. Yokoi, M. Hirata, et al. «Closed-Loop Control of a Neuroprosthetic Hand by Magnetoencephalographic Signals». In: *PLoS ONE* 10.7 (2015), e0131547. DOI: 10.1371/journal.pone.0131547 (cit. on p. 18).

[5] Kristin K. Sellers, Joshua L. Cohen, Ankit N. Khambhati, Joline M. Fan, A. Moses Lee, Edward F. Chang, and Andrew D. Krystal. «Closed-loop neurostimulation for the treatment of psychiatric disorders». In: *Neuropsychopharmacology Review Article* (2023). © The Author(s), under exclusive licence to American College of Neuropsychopharmacology (cit. on p. 19).

[6] M. M. Ottaviani, F. Vallone, S. Micera, and F. A. Recchia. «Closed-Loop Vagus Nerve Stimulation for the Treatment of Cardiovascular Diseases: State of the Art and Future Directions». In: *Frontiers in Cardiovascular Medicine* 9 (2022). Specialty section: Hypertension; Edited by Deborah Hunt; Reviewed by Ankit Gilani and Charles C. Horn; Correspondence: Fabio A. Recchia (fabio.recchia@santannapisa.it), p. 866957. DOI: 10.3389/fcvm.2022.866957 (cit. on p. 19).