NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF MATERIALS SCIENCE

MARIA MOUTINHO ALVAREZ

BSc in Micro and Nanotechnologies Engineering

OPTIMIZATION OF A CONTROLLER FOR MARINE POLLUTANT MONITORING USING FIELD EFFECT DEVICES

MASTER IN MICRO AND NANOTECHNOLOGIES ENGINEERING

NOVA University Lisbon
September, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF MATERIALS SCIENCE

OPTIMIZATION OF A CONTROLLER FOR MARINE POLLUTANT MONITORING USING FIELD EFFECT DEVICES

MARIA MOUTINHO ALVAREZ

BSc in Micro and Nanotechnologies Engineering

Adviser: Dr. Joana Dória Vaz Pinto
Assistant Professor, NOVA University Lisbon

Co-adviser: Dr. Bruno Miguel Ribeiro Veigas
Senior Researcher, Almascience

Examination Committee

Chair: Dr. Pedro Miguel Cândido Barquinha
Associate Professor, FCT-NOVA

Rapporteur: Dr. Rui Alberto Garção Barreira do Nascimento Igreja
Associate Professor, Another University

MASTER IN MICRO AND NANOTECHNOLOGIES ENGINEERING

NOVA University Lisbon
September, 2023

Optimization of a Controller for Marine Pollutant Monitoring Using Field Effect Devices

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.10.18) [1].

Acknowledgements

I want to express my deepest thanks to everyone who played a part in shaping this thesis, which represents the culmination of my journey in the field of Micro and Nanotechnologies Engineering.

I'm grateful to my thesis advisors, Professor Joana Pinto and Bruno Veigas, for their guidance, mentorship and expertise throughout this project. I'd also like to thank all the members of the CEMOP team who helped me with problems beyond my expertise, Ricardo and Ana Santa. I'm also thankful to the entire faculty at NOVA SST, especially the professors in the Department of Materials Science, and the research teams at CEMOP and CENIMAT, for sharing their knowledge and supporting me. I'm grateful for the collaboration of the SeaSenseX team and University of Macao, specially to the contribution of Man Kay Law, since their contribution was instrumental in shaping my research.

I'm deeply grateful for my friends Ana, André, Mia, Margarida, Sofia, and Rui, whose presence and encouragement were always present in our conversations, discussions, lunches and breaks. Your friendship was a source of strength, laughter and companionship much needed during this adventure.

The deepest gratitude is reserved for my family. Mãe, Pai, Mano and Ju, thank you for always believing in me. Your support and understanding during tough times have always been my rock. Your faith in me has motivated me to pursue my goals. Your love has given me strength, and I'm forever grateful for that.

”

“The only thing predictable about life is its unpredictability.”

— Remy

Abstract

The growing threat of heavy metal pollution in marine ecosystems demands reliable monitoring solutions due to its growing threats to aquatic ecosystems and human health. The SeaSenseX project addresses this urgency by creating a fully portable device equipped with innovative sensors, custom-made circuits and hardware components for reliable, real-time data collection and analysis. This work aims to overcome two critical challenges: simplifying the operation of an advanced technology and eliminating the dependence on laboratory-based tests, which in turn allows for the operation of users with minimal technical knowledge. A custom-made software allows for seamless integration of essential components, such as a Printed circuit board, a Charge-modulated ion sensitive field-effect transistor sensor, and an Arduino microcontroller. The software enables data collection, analysis and visualisation directly on-site. The versatility of the programming language Python was leveraged in the development of a reliable and intuitive guided user interface. The user can effortlessly engage with the technology through the user-friendly interface, thus bridging the accessibility gap. The results of this work have significant implications for marine pollution monitoring, offering a reliable, user-friendly tool for real-time data collection and analysis in various field scenarios. By eliminating the need to transport samples to a lab, this approach streamlines the monitoring process and empowers researchers to make informed decisions promptly. In conclusion, this work is based on the development of a the custom-made software for the control and analysis of a data acquisition system using Charge-modulated ion sensitive field-effect transistor sensors. Additionally the development and testing of the sensors was overseen. Finally, the custom-made Printed circuit boards were analysed discovering discrepancies and inefficiencies, while necessary modifications and optimizations were proposed to enhance the device's functionality. The software allows for the integration of these components while maintaining the complexity of the device's usage low.

Keywords: User-friendly software, Guided User Interface, Printed Circuit Board, Data Acquisition System, CMISFET

Resumo

A progressiva ameaça de poluição de metais pesados nos ecosistemas marinhos exige soluções de monitorização fiáveis devido ao crescente impacto nestes ecossistemas e na saúde humana. O projeto SeaSenseX aborda este problema através da criação de um dispositivo completamente portátil equipado com sensores inovadores, circuitos e componentes de hardware feitos à medida para a recolha e análise de dados em tempo real. Este trabalho tem como objetivo superar dois desafios: simplificar a operação de uma tecnologia avançada e eliminar a dependência nos laboratórios para a realização de testes, permitindo a utilização de utilizadores com conhecimentos mínimos. Um software feito à medida permite a integração de componentes essenciais, como a PCB, um sensor CMISFET e um microcontrolador Arduino. O software permite a recolha, análise e visualização de dados diretamento no local de análise. A versatilidade da linguagem de programação Python foi utilizada no desenvolvimento de uma interface de utilizador intuitiva. Esta interface amigável permite que os utilizadores interajam com a tecnologia facilmente, eliminando a lacuna de acessibilidade. Os resultados deste trabalho têm implicações significativas para a monitorização de poluição marinha, proporcionando uma ferramenta fiavél para a recolha e análise de dados em tempo real em diversos cenários. Eliminando a necessidade de transporte das amostras para laboratório, o processo de monitorizaçãoesta é simplificado e habilita investigadores a tomar decisões informadas rapidamente. Em conclusão, este trabalho baseia-se no desenvolvimento de um software feito à medida para o controlo e analise de um sistema de acquisição de dados usando um sensor CMISFET. Para além disso, o desenvolvimento e teste dos sensores for acompanhado e replicado. Por fim, as PCBs feitas à medida foram analisadas, sendo identificadas discrepâncias e ineficiências, para as quais são então propostas as modificações e optimizações necessárias por forma a melhorar a funcionalidade do dispositivo. Desta forma o software permite a integração destes components, mantedndo a complexidade de utilização do dispositivo baixa.

Palavras-chave: Software Amigável, Interface de Utilizador Intuitiva, Placa de Circuito Impresso, Sistema de Aquisição de Dados, CMISFET

Contents

List of Figures

List of Tables

Acronyms

ADC	Analog-to-digital converter
CMISFET	Charge-modulated ion sensitive field-effect transistor
DAQ	Data acquisition
GUI	Graphical user interface
MOSFET	Metal-oxide-semiconductor field-effect transistor
PCB	Printed circuit board
PEN	Polyethylene naphthalate
ZIF	Zero insertion force

Symbols

C_f	Feedback capacitor
I_d	Drain current of MOSFET
R_f	Feedback resistor
V_a	Input voltage on the amplifier non-inverting node
V_{adc}	Circuit output voltage inserted in the ADC
V_b	Bulk voltage of MOSFET
V_{cg}	CMSIFET control-gate voltage
V_d	Drain voltage of MOSFET
V_{ds}	Drain-to-source voltage
V_g	Gate voltage of MOSFET
V_s	Source voltage of MOSFET

1 Introduction

Marine pollution, including of heavy metals, has become a worldwide concern, impacting both marine ecosystems and human health [2]. The World Health Organization estimates that around 1 million people die from lead poisoning every year and millions more, many of whom are children, are exposed to this heavy metal, causing lifelong health problems [3].

Heavy metals have high density and atomic weight and, nowadays, stem from anthropogenic sources rather than natural sources. These metals can be essential, such as lead , which are required for the correct functioning of living organisms, or non-essential, like mercury, cadmium, arsenic and nickel, which do not present a biologic role in living organisms [2]. These pollutants can easily come into contact with the marine environment, through landfills, waste dumps, shipwrecks, oil spills, and the usage of pesticides, insecticides, and fertilizers [4]. Then, sewage and runoff transport them, which in turn will disperse them into rivers, seas and oceans. Exposure to heavy metals can occur through the ingestion of contaminated food and water, the inhalation of tainted air, or skin contact with these elements, and its impact varies based on factors such as the type of metal, exposure route and duration, and individual susceptibility [2, 5].

The presence of non-essential heavy metals, or higher concentrations of essential heavy metals, can lead to toxic accumulation in organs and tissues, causing irreversible health issues. Lead, commonly found in volcanic eruptions and industrial applications such as mining, paints and plastics, has been linked to hypertension and cardiovascular issues. Mercury, present in gold mining, coal combustion, and thermometers, can cause insomnia, neurological damage and pneumonia. Cadmium, used in batteries and cement production, may lead to lung cancer and immune system malfunctions. Arsenic, common in fossil combustion and pesticides, can result in skin cancer and neurological complications. Nickel, found in wildfires, battery factories and coin making, is associated with asthma, ulcers and lung cancer [2, 5]. In addition to affecting humans, organisms living in or near contaminated waters are also heavily impacted by these pollutants. Besides the previously mentioned consequences, heavy metals can restrict the growth of various aquatic life forms, including algae, mollusks, and fish, thus disrupting the food chain. Additionally,

the reproductive patterns of certain species can be affected, leading to population shifts that can, in turn, disturb the balance of ecosystems, either by species absence or changes in dominance [6].

In order to minimize health risks associated with the exposure of heavy metals, there is a growing demand for reliable and efficient tools to monitor and measure these types of pollutants in the marine environment [7].

This work, part of the ongoing SeaSenseX project, is a collaborative effort between the SeaTox Lab, CENIMAT and I3N research units and the University of Macao, aiming to develop a novel biosensor for marine pollution monitoring [8]. The project aims to create a fully portable Data acquisition (DAQ) system for on-site real-time monitoring empowering researchers to make informed decisions promptly.

1.1 Data Acquisition Systems

A DAQ system acquires data from a physical phenomenon, in this project the presence of heavy metals in a sample varies the response of the sensor, which is then converted from the analog value to a digital one to be manipulated by a computer. The proposed DAQ is made of a sensor, a Charge-modulated ion sensitive field-effect transistor (CMISFET), electric network and signal conditioning, both part of a Printed circuit board (PCB), DAQ hardware, an Arduino microcontroller which contains an Analog-to-digital converter (ADC), DAQ software and an operating system. This thesis contributes to the SeaSenseX project in the development of an user-friendly software that enables the interaction of all hardware components and facilitates seamless data collection and analysis.

1.2 CMISFET Sensor

Marine pollution monitoring relies on a variety of sensors capable of detecting and measuring different types of pollutants, so in recent years, this field has made significant advancements, leading to highly sensitive and efficient devices. During this work, the development and testing of the Charge-modulated ion sensitive field-effect transistor (CMISFET) sensors was overseen.

The Charge-modulated ion sensitive field-effect transistor (CMISFET), Figure 1.1 is a potentiometric sensor that uses an ion-sensitive field-effect transistor structure, combined with a capacitive structure for the detection of variations in potential in its surface. This variation in potential is caused by the interaction between the sensitive layer and an analyte or test sample. The sensor operates on the principle of potentiometric measurement, where changes in pH result in voltage variations and consequently in shifts in the transistor's characteristic curves. This allows the CMISFET to convert the chemical information of the sample into an electrical signal that can be easily measured and processed. This makes the CMISFET an ideal choice for applications requiring high sensitivity and accuracy in the detection of various chemical species. The main advantage of CMISFET sensors

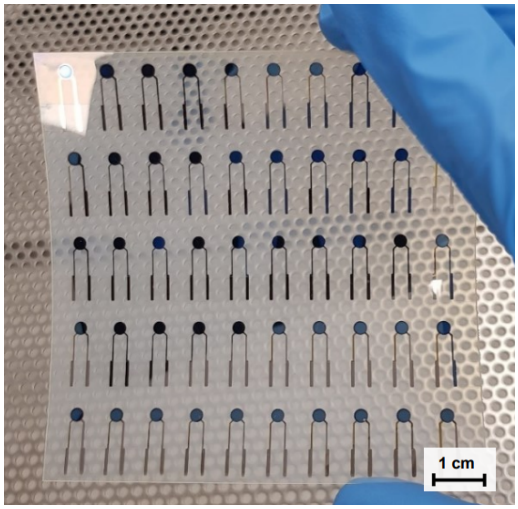


Figure 1.1: CMISFET sensors developed on PEN substrate. Reproduced from a Master's thesis in Micro and Nanotechnology Engineering in 2022 [9]

lies in the ability to provide real-time, label-free, and non-destructive measurements. Additionally, CMISFET sensors offer several other advantages such as small size, low power consumption, and compatibility with microfabrication techniques, allowing for the development of miniaturized and integrated sensing systems [10].

1.3 Hardware Components

For this marine pollution monitoring device, a PCB acts as a bridge between the CMISFET sensor and an Arduino microcontroller. This circuit was designed to meet the specific demands of the project by creating a completely portable device. The reliance on laboratory equipment is given by the need for high impedance instruments in order to analyse small variations of current and voltage of the CMISFET. By designing a PCB that emulates these types of interactions, eliminates the need to transfer samples back to a laboratory for CMISFET sensor usage. With the PCB's interaction with the Arduino, the device becomes transportable, enabling on-site measurements.

The PCB needs to be designed to meet the specific demands of the project, ensuring reliability, signal integrity and ease of assembly. In order to simplify the assembly process, the PCB was designed to be directly placed on top of the Arduino and with an easy access to the placement of the sensor. The ability to deliver a specific voltage that could be precisely changed to the CMISFET sensor, allowed for different types of analysis. The sensor's response is then routed through a transistor, which results in a variable drain current. This current is then filtered to eliminate unwanted noise ensuring signal clarity, which in turn is amplified and converted to voltage. This conversion is crucial since it allows for the Arduino to interpret the data.

The development of the PCB was done in a Master's thesis of a student in 2019 [11]. In this work, an initial version, which had already been constructed, was tested in order

to understand the circuit. An upgraded version, also previously designed by the same student, intended to solve problems with the previous version, was also tested.

In parallel to the testing phase of these versions, a custom made MOSFET, part of the circuit used to control and evaluate the response of the sensor, was developed by researchers from the University of Macao. Additionally, a new PCB was also developed so that it could house either MOSFET, the newly fabricated one and the commercially available one which was being used in the previous versions.

1.4 Software

In parallel with the need for efficient pollution monitoring tools, the SeaSenseX project tackles these environmental concerns from multiple angles. One of the crucial parts of this project is the ease of use of the device for all users, which requires a simplification of the data acquisition and processing that is retrieved from the hardware components, the CMISFET sensor, the PCB and the Arduino.

1.4.1 Existing Software

In the realm of data acquisition, software plays a crucial role in simplifying complex tasks, automating processes, and enabling real-time data analysis and visualization. Several software tools and platforms have been developed to meet the diverse needs of researches, each with specific advantages, disadvantages and specifications [7, 12].

Labview and LabWindows/CVI, both develop by National Instruments, offer comprehensive environments for measurement, control and data analysis. These graphical programming languages allow users to visually design systems for input control and output visualization, offering compatibility with various National Instruments' hardware and other hardware technologies [12–14]. However, being a proprietary software, both Labview and LabWindows/CVI can result significant licensing costs, particularly for extended functionalities. Additionally, modifying or extending LabVIEW or LabWindows/CVI-based applications may require additional licenses, limiting the software's reach compared to open-source applications [13, 14].

The Data Acquisition Toolbox is a MATLAB add-on that extends its data acquisition and control capabilities. It offers a user-friendly hardware interface with a seamless integration into MATLAB, simplifying data acquisition and analysis processes. Just as for the National Instruments' programs, acquiring a MATLAB and toolbox licenses can be expensive, limiting its accessibility. Furthermore, due to licensing constraints, sharing MATLAB-based applications can be challenging, impacting collaboration. Additionally, this option does not cover all specialized hardware needs [15, 16].

1.4.2 Custom-made Software

Given the limitations and difficulties of existing software tools, the development of custom-made solutions has become necessary. A custom-made software solution arises as a key asset, adapting to specific project requirements and simplifying complex tasks, making it more accessible to a wider audience. These types of solutions aim to offer user-friendly interfaces, seamless hardware integration, and tailored data analysis and visualization capabilities. By addressing specific project needs, custom-made software can optimize workflows, reduce costs and empower users with minimal technical knowledge to engage efficiently with advanced technology [12].

This thesis contributes to the SeaSenseX project in the development of a custom-made software that enables the interaction of all hardware components, the CMISFET sensor, the PCB and the Arduino, facilitating seamless data collection and analysis

The software, designed with users in mind, allows for the generation of valuable insights developed from raw data acquired through the Arduino. This makes the interaction with the multiple hardware components (CMISFET sensor, PCB and Arduino) much simpler and, by offering the user a multitude of analytical tools, facilitates data manipulation and visualisation. The software allows for multiple analysis options tailoring it to the specific needs of the research, aiding in decision making.

The GUI of the software, developed through the Python programming language, offers an intuitive means for users to input parameters, initiate measurements and visualize data. By taking advantage of advanced Python libraries, such as Tkinter, Pandas, Numpy and Matplolib, the software allows for complex data transformations and visualization with efficiency always in mind [17–21]. Moreover, the software addresses user input errors by identifying and notifying the user, which alongside with multiple features of the software, highlights its reliability in working alongside the advanced hardware, ensuring accurate and useful environmental insights.

This thesis oversees the production of a new batch of CMISFET sensors, made both in paper and PET substrates, along with the testing of the sensors. The existing PCBs were analysed and tested, along with a new version of a MOSFET custom-made for the project. Along with the findings of the previously PCBs and the need to house the newly developed MOSFET, the requirements for a new optimized version of a PCB were developed. Based on these findings, the software, aiming for the integration of all components and on-site analysis, was developed to mimic the tests made in laboratory. This software and its functionalities are a pivotal step in the SeaSenseX project allowing for informed decisions based on real-time and precise data.

2

Materials and Methods

2.1 Hardware

2.1.1 CMISFET Fabrication

The CMISFET sensor consists of four layers which are deposited sequentially on a substrate, which can be paper, PEN, or corning glass. The back and front contacts were made of molybdenum thin films that were sputtered in a AJA ATC system, using a 3" molybdenum target, a RF power of 175 W, in argon atmosphere, with a deposition pressure of 0.24 Pa, while spinning the substrate for uniform deposition. The deposition time was 12 minutes for a thickness of 65 nm, except for paper substrates, which required 24 minutes for 130 nm thickness due to increased substrate roughness. The parylene-C was deposited by chemical vapor deposition using a Parylene Deposition System 2010 LabcoterTM 2 by Specialty Coating Systems, with two consecutive depositions to create a 400 nm dielectric film, with silane used as an adhesion promoter. Another molybdenum contact layer was deposited using a mechanical mask, similar to the previous contact layer. The tantalum oxide sensitive layer was sputtered using a mask similar to that used for the molybdenum layer, with specific areas covered by kapton tape.

2.1.2 CMISFET Testing

The sensors underwent a dry capacitance test to confirm functionality, with a defined working range of capacitance between 0.120 nF and 0.350 nF, established from previous works.

The sensors then underwent a dry electrical testing coupled with either of the MOSFET options to ensure proper signal transfer, using the Agilent 4155C semiconductor parameter analyser and Cascade Microtech M150 microprobe station, altogether with Keysight EasyEXPERT group+ software. The back contact of the sensor, the control-gate, is connected directly to one of the SMU ports and the front contact connected to the gate of the MOSFET. The remaining pins of the MOSFET, drain, source and bulk are directly connected to the SMU ports. Here, voltage sweeps are performed, monitoring the drain

current of the MOSFET, I_d, applying a drain voltage of V_d = 500 mV, a source voltage of V_s = 0 V, and bulk voltage of V_b = 0 mV, and varying the control-gate voltage,V_{cg}, the voltage being applied on the contact of the sensor, from 800 mV to 2 V, with voltage steps of 30 mV. For each MOSFET, ten analysis were performed sequentially with a hold of 20 s between sweeps.

Subsequently, a time analysis was performed on the sensors, where V_{cg} = 1.7 V, V_d = 500 mV, V_s = 0 V and V_b = 0 V, and monitoring I_d with sampling intervals of 1 s. For this test pH4 and pH10 buffer solutions were applied on the sensors instead of sample solutions with heavy metals, since the former are easier to acquire and pose much less of a threat for this testing phase and, due to their charges, will still cause a change in the sensor's response.

2.1.3 MOSFET Testing

This testing evaluated the MOSFETs through their transfer characteristics and response with the sensor and samples being applied, comparing between the two options. The electrical characterization was again performed with Agilent 4155C semiconductor parameter analyser and Cascade Microtech M150 microprobe station, altogether with Keysight EasyEXPERT group+ software.

A voltage sweep is performed by monitoring the drain current, I_d, while applying a V_d = 500 mV, V_s = 0 V, and V_b = 0 mV, and varying V_g from 800 mV to 2 V, with voltage steps of 30 mV. For each MOSFET, ten analysis were performed sequentially with a hold of 20 s between sweeps.

2.1.4 PCB Testing

Thorough testing of the circuits is essential to ensure proper functionality of the PCB as a whole, ensuring circuit correctness, reliability and effectiveness. The testing process begins with an examination of the connections and comparing them to the circuit diagram to ensure accurate wiring. Subsequently, the PCB is powered, enabling the measurement of voltage values along the components.

Once the testing of the PCB is completed, the connections between the PCB and the Arduino are made, and the testing of the micro-controller along with the PCB can be made. In order to test the acquisition process of the Arduino, it is possible to compare the readings of the Arduino with those made with a voltmeter. This can be achieved by running a sample script, included in appendix B, through the Serial Monitor of the Arduino IDE, which measures the readings from the analog pins. The script provides a convenient means of cross-referencing the acquired data, ensuring the integrity of the Arduino's data acquisition capabilities.

2.2 Software Development

In the initial phase of the process, the software's requirements and tasks were gathered through a thorough examination of existing systems, to understand their functionalities and user needs. Given that the device aims to emulate the types of analysis conducted in a laboratory setting, an in-depth study of those laboratory instruments served as a foundation for the software's blueprint. With those requirements in hand, the subsequent phase focuses on software analysis. During this stage, a deep understanding of each requirement is done, focusing on determining how the software would function to fulfill these needs. Detailed flowcharts and diagrams are created to define the software's architecture. The implementation phase followed, where the design specifications are translated into code and software modules. Finally, testing was conducted systematically, covering every aspect to identify and rectify potential bugs or issues. In cases where issues or requirements emerged during the testing phase, the analysis phase was revisited. This iterative process allows for adjustments to the design and implementation, ensuring the software meets all objectives [12].

The development of the software was done with the Python programming language, through Spyder IDE, which offers a wide range of versatile tools and libraries [17]. Additionally, the scripts to communicate between the computer and the Arduino were written in the C++ programming language through the Arduino IDE [22].

3

Results and Discussion

This chapter presents a review of the experimental data and its implications for the research's objectives. Focused on functionality, integration and performance of the hardware and software components, this analysis evaluates the overall performance the device. This assessment provides insights into the device's strengths, limitations and potential improvements. Here, the hardware components are presented first, followed by the software, even though the testing and development procedures were made at the same time.

3.1 Hardware Performance

As previously mentioned, the PCB is used for the control and response analysis of a CMISFET. From previous works, the response of the CMISFET is passed through a MOSFET and the drain current is analysed [9]. Two types of analysis could be done, a voltage sweep and real-time monitoring, Figure 3.1.

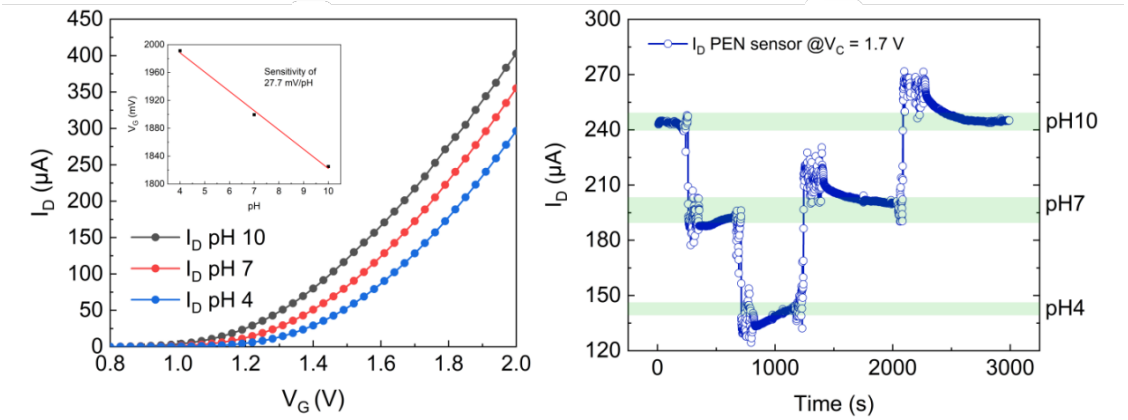


Figure 3.1: CMISFET pH sensing performance methodologies. Left: transfer characteristics obtained at a constant $V_d = 0.5$ V for a complete pH cycle, with curves expressed as variation of I_d at fixed V_g. Right: continuous real time monitoring of I_d at a control voltage of 1.7 V for a complete cycle of pH. Reproduced from a Master's thesis in Micro and Nanotechnology Engineering in 2022 [9].

However, in order to make this type of analysis, specific laboratory equipment was needed, such as the Agilent 4155C semiconductor parameter analyser and Cascade Microtech M150 microprobe station, altogether with Keysight EasyEXPERT group+ software. This meant that for a portable device,the circuit needed to emulate the characteristics of this component and overcome noise reduction and amplification restraints. Two PCB versions, 1.1 and 1.2, were developed in a Master's thesis, and this project started with the evaluation of these two [11].

3.1.1 PCB Evaluation and Development

This section is focused on the evaluation of the performance of the PCBs. The goal is to evaluate the components' functionality, potential challenges, a comparison between actual performance with the initial design, and the identification of potential improvements based on the outcomes of the tests. This analysis aims to shed light on the effectiveness and reliability of this hardware component, providing information on how well they correspond with the desired goals.

The PCB was designed as a means of transferring data from the sensor to the Arduino. For this, the signal from the sensor needed to be processed since the high impedance characteristics of the laboratory's instruments were no longer in place. This meant that the PCB both needed to filter the signal and amplify it before it is processed by the ADC. A diagram explaining the transformation applied on the data can be seen through Figure 3.2.

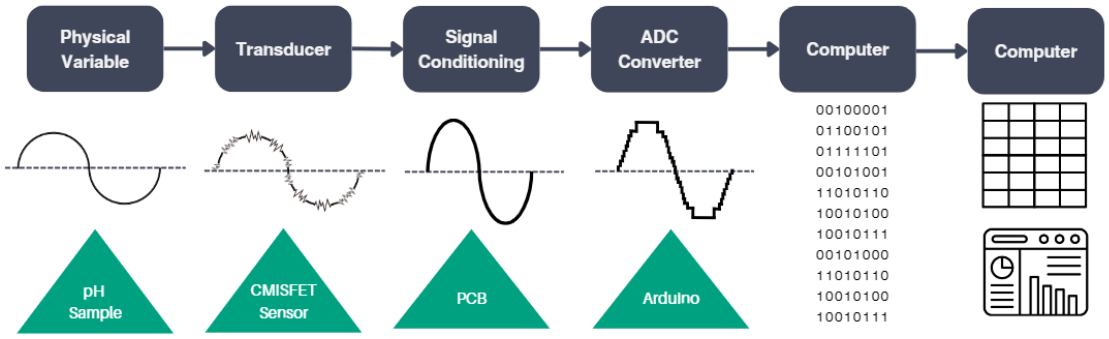


Figure 3.2: Schematic illustration of signal transfer and transformation through hardware (sensor, PCB, Arduino and computer) and software components.

3.1.1.1 PCB Version 1.1

The first version of the PCB, had been previously developed to replicate the circuit shown in Figure 3.3, aimed to establish the connection between the microcontroller and one sensor, while filtering and amplifying its response. The proposed circuit would allow for the control of the voltage being applied to the sensor, V_{cg}, by controlling the X9C103P component, while monitoring V_s and V_a to evaluate the performance of the circuit. The

V_{adc}, highlighted in yellow, is the output of the circuit, which is what should translate the response of the sensor and then be transferred to the Arduino.

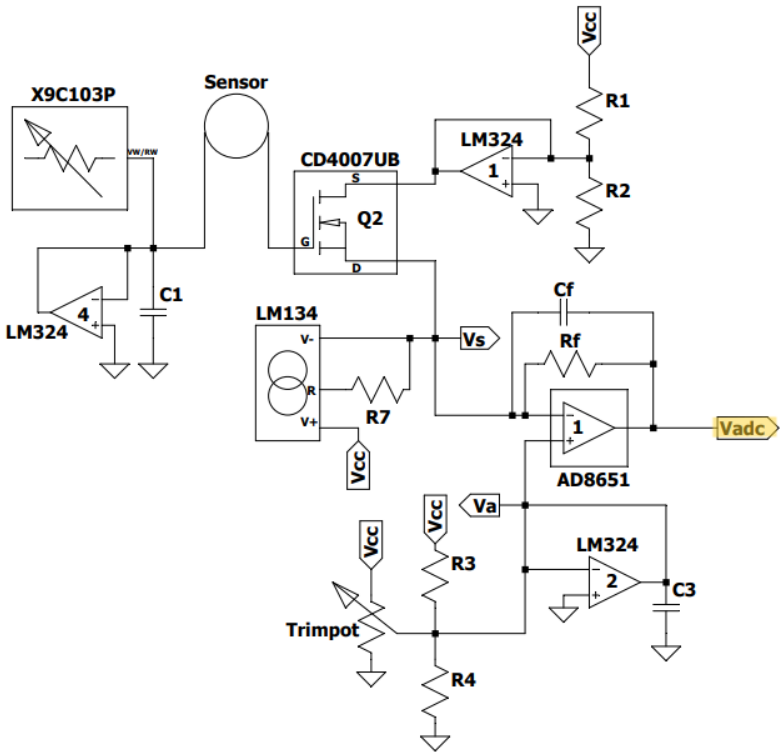


Figure 3.3: Schematic of theoretical circuit design of the PCB portraying the connections between components and highlighting output voltage (V_{adc}). Reproduced from a Master's thesis in Micro and Nanotechnology Engineering in 2019 [11].

Upon receiving the developed PCB, the analysis uncovered a significant inconsistency from the intended design, which was due to the accommodation of two CMISFET sensors, instead of the intended one. This lead to further analysis of the circuit's connections, using a ohmmeter and a voltmeter, to systemically retrace the schematic. The reconstructed circuit, represented in Figure 3.4, reveals the differences between the intended design and the real implementation.

Further analysis revealed other inconsistencies in the connections, specifically with the Q1 and Q2 MOSFETs, part of the CD4007UB component. The source and drain pins of these transistors were incorrectly connected, contrary to the operational principle of a MOSFET, wherein the drain should maintain a higher voltage than the source, so that V_{ds} is bigger than zero, to ensure proper functioning [23]. Since the voltages in these pins did not align with the requirement, the transistors were not being correctly biased to perform their intended functions. Additionally, the LM134 components, the current sources, were also incorrectly connected. The $V+$ pin, which should be connected to V_{cc}, was incorrectly connected to the rest of the circuit, while the $V-$ pin was the one connected to V_{cc} [24]. These misconnections meant that the current sources were also not behaving as intended. Furthermore, the feedback capacitors (C_{f1} and C_{f2}) for the inverting low-pass

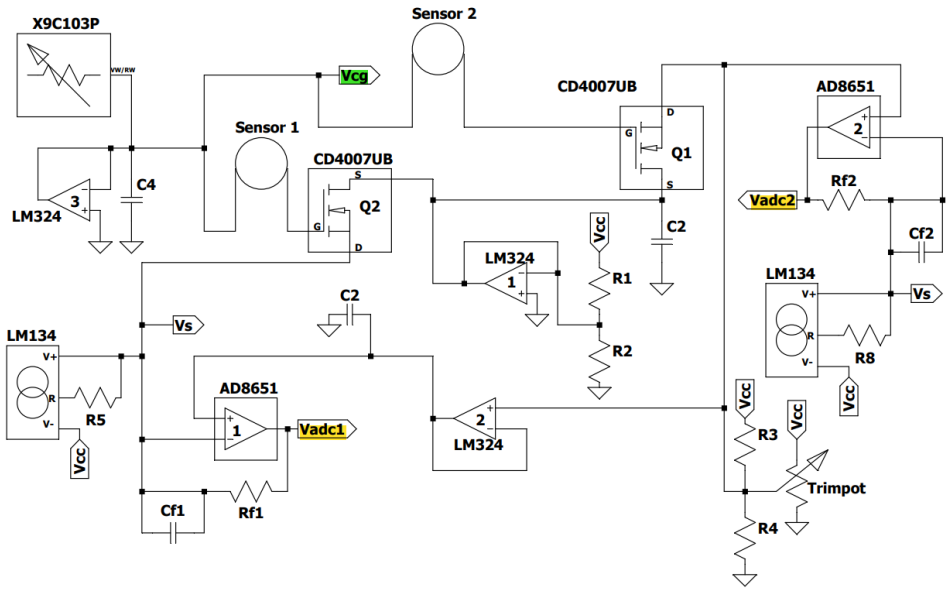
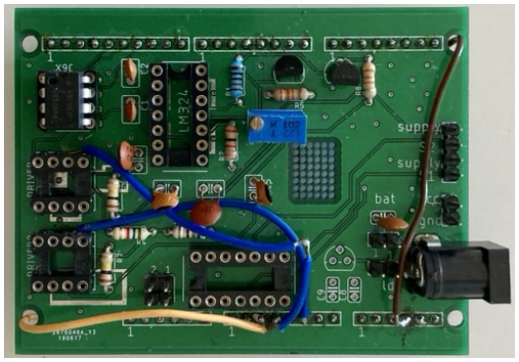


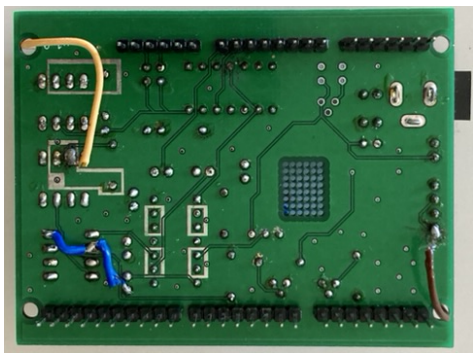
Figure 3.4: Schematic representing the actual circuit reconstructed from PCB version 1.1 portraying the actual connections between components and highlighting output voltages (V_{adc}) and sensor input voltage (V_{cg}).

filters were not connected in parallel to their corresponding feedback resistors (R_{f1} and R_{f2}) as required [25].

From the PCB's front and back pictures in Figure 3.5, it is evident that other errors had previously been identified by the wires added between some components. These wires were used to solve some problems and some were added to ensure that all needed values were accessible by the microcontroller from the pins on the periphery of the PCB. This improvisation, while temporarily ensuring the resolution of some of the issues, meant that the fabrication of a functional device that relied on this version, would not be efficient or reliable.



(a) Front view of PCB version 1.1 portraying the spatial arrangement of components on the top side of the board.



(b) Back view of PCB version 1.1 portraying the spatial arrangement of components on the bottom side of the board.

Figure 3.5: Front and back views of PCB version 1.1 portraying the spatial arrangement of components on the board.

3.1.1.2 PCB Version 1.2

An upgraded version of the version 1.1 had also been developed to address the issues discovered in the previous one. Version 1.2 was developed in order to remove the biggest issue encountered, the double sensor configuration, while maintaining the initial schematic from Figure 3.3. As before, the circuit would allow for the control of the voltage being applied to the sensor, V_{cg}, by controlling the X9C103P component, while monitoring V_s and V_a to evaluate the performance of the circuit. The V_{adc}, highlighted in yellow, is the output of the circuit, which is what should translate the response of the sensor and then be transferred to the Arduino. For this version, the necessary components were soldered following the instructions left by the previous work and, by applying the same strategy as before, it was possible to acquire the reconstructed circuit seen in Figure 3.6.

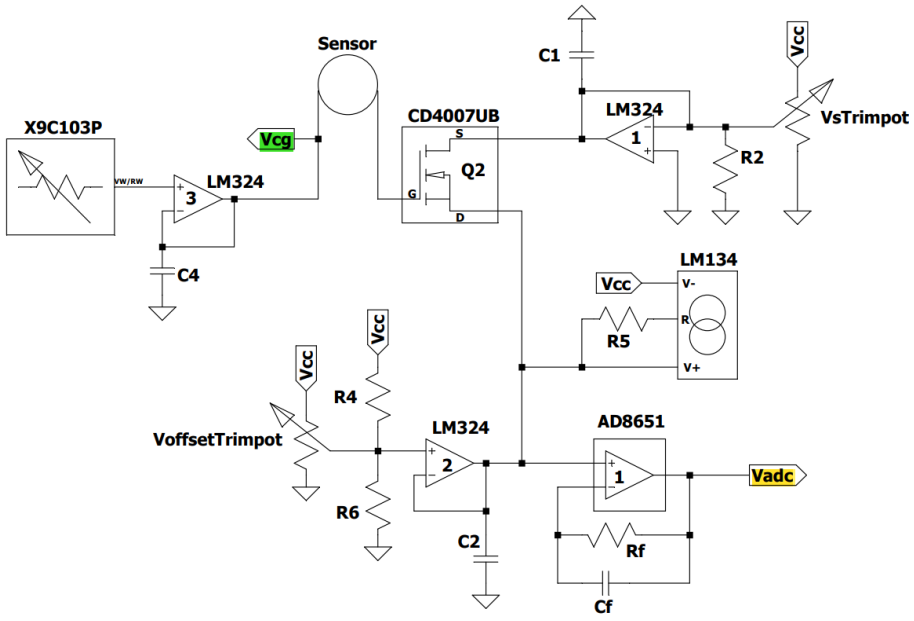


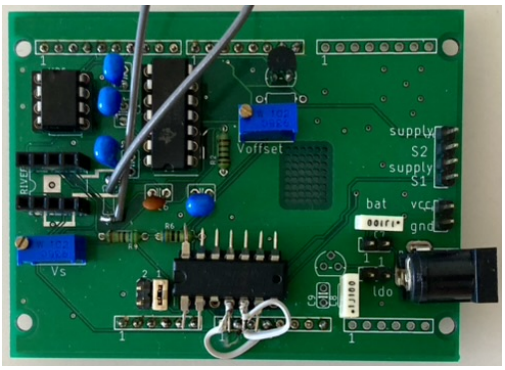
Figure 3.6: Schematic representing the actual circuit reconstructed from PCB version 1.2 portraying the actual connections between components and highlighting output voltage V_{adc} and sensor input voltage V_{cg}.

From the reconstructed circuit, it is possible to see that the biggest issue was addressed and the circuit only houses one sensor. Additionally, the feedback capacitor, C_f, was also connected properly in parallel to the feedback resistor, R_f. Another analog potentiometer was introduced to regulate the voltages applied to transistor Q2.

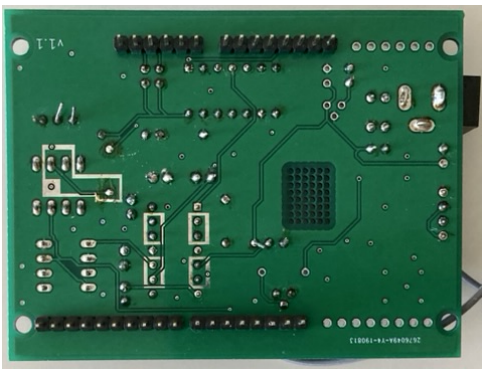
However certain issues persisted. The voltages at the source and drain pins of the MOSFET remained incorrect, as were the $V+$ and $V-$ pins of the current source. It was also noted that some pins of the MOSFET component were incorrectly connected to other components. Furthermore, there was a switch of the + and - pins of the operational amplifier, part of LM324 component, denominated 3 in Figure 3.6. A similar discrepancy can be found with the operational amplifier part of the LM324 component, labeled as 2 in Figure 3.6. Moreover the operational amplifier 2 was now connected directly to the drain

pin of the MOSFET and positive pin of the AD8651 component, contrary to the original circuit. Lastly, a capacitor C_2 was also added to the output and negative pins of amplifier 2 of the LM324.

In an attempt to rectify the connection errors, a few modifications were made to the PCB as shown in Figure 3.7. The most notable was the switching of the source and drain pins and the severing of the unnecessary connections, leaving only the needed intact [26]. However, this alone did not make the circuit functional.



(a) Front view of PCB Version 1.2 portraying the spatial arrangement of components on the top side of the board.



(b) Back view of PCB Version 1.2 portraying the spatial arrangement of components on the bottom side of the board.

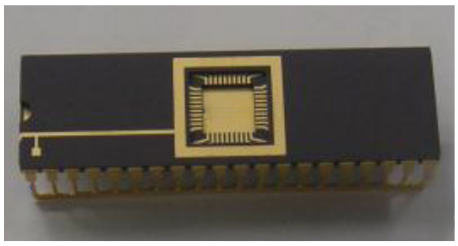
Figure 3.7: Front and back views of PCB version 1.2 portraying the spatial arrangement of components of the board.

Additionally, it was found that the MOSFET had a very limited dynamic range, given by the specifications of the component. This meant that the transistor only responded to a voltage range between 3.6 V and 3.8 V, whereas sensor's input voltage could vary from 0 V to a maximum of 5 V. In order to increase the dynamic range, the value of R_f could be significantly reduced. However, this reduction would proportionally decrease the amplification of the signal when it is converted from current to voltage by passing this resistor. Consequently, the microcontroller might not be able to detect the necessary variations in V_{adc}.

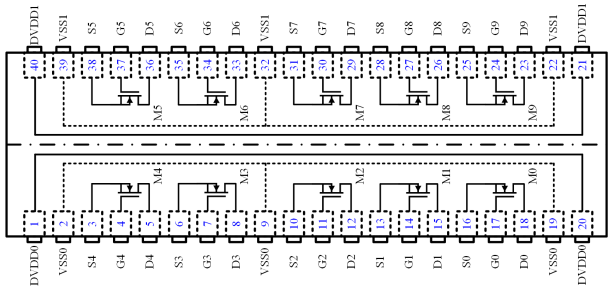
3.1.2 Custom MOSFET

Since the analysis of the sensor depended heavily on the transistor used, a custom MOSFET was developed by researchers of the University of Macao. This custom MOSFET was tuned to align with the the specific requirements of the project, ensuring optimal performance. For the testing of this newly developed component, here referred as the MKProt seen Figure 3.8, comparisons were made with the commercially available MOSFET CD4007UB, which was being used in the testing of the sensor up until this point.

The focus of this analysis is their transfer characteristics, specifically the relationship between V_g and I_d, as each can be seen in Figure 3.9a and Figure 3.9b. The CD4007UB



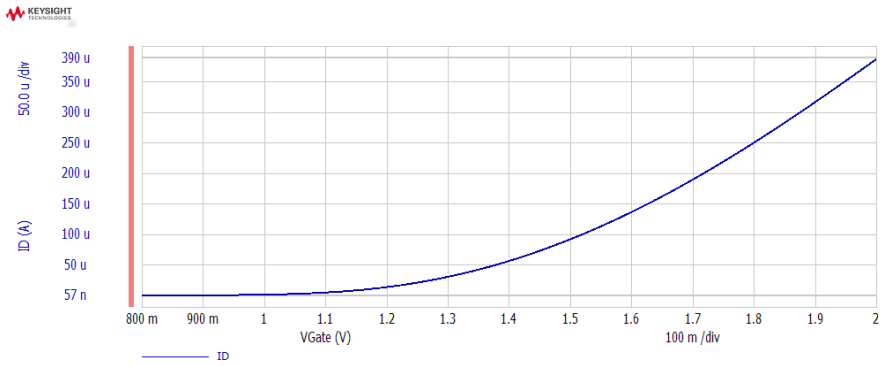
(a) Packaging of the custom-made MOSFET, MKProt.



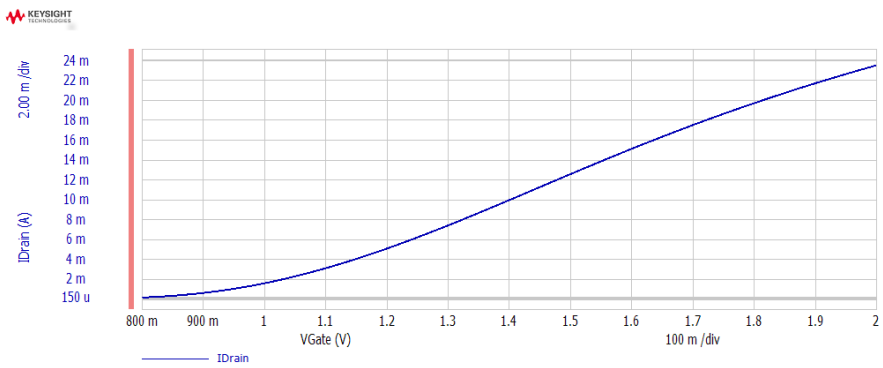
(b) Schematic diagram of the internal arrangement and interconnections of transistors within the the custom-made MOSFET, MKProt.

Figure 3.8: Packaging and schematic of the custom-made MOSFET, MKProt.

MOSFET exhibited a transfer characteristic ranging from a minimum of 57 nA to a maximum of 390 µA. This represents a variation of approximately 389.943 µA across the entire gate voltage range. In contrast, the transfer characteristic of the MKProt MOSFET revealed a broader operational range. Here, the drain current ranged from a minimum of 150 µA to a maximum of 390 mA, which translates to a variation of 389.85 mA.



(a) Transfer characteristic of the CD4007UB MOSFET.



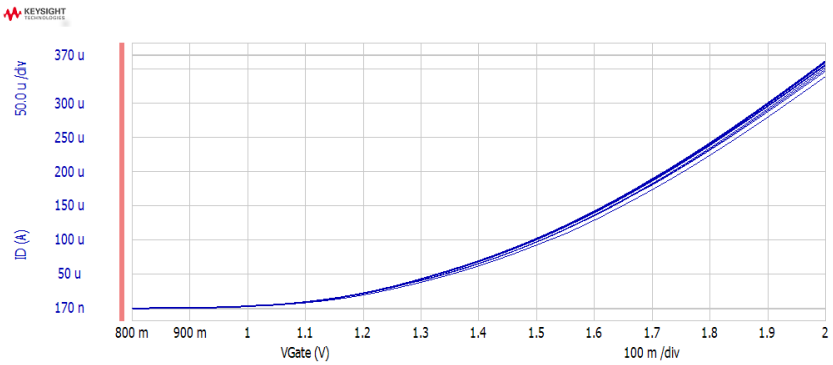
(b) Transfer characteristic of the MKProt MOSFET.

Figure 3.9: Transistor performance of the CD4007UB and MKProt MOSFETs evaluated through the transfer characteristic with 10 curves acquired every 20 seconds.

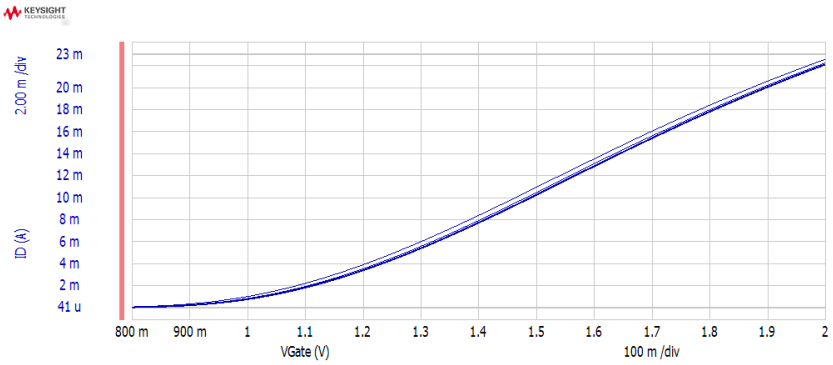
From the transfer characteristics, the MKProt MOSFET stands out significantly. While the CD4007UB MOSFET operates in the microampere range, the MKProt MOSFET operates in the milliampere range. This substantial variation marks the MKProt MOSFET as the optimal choice for this application. Furthermore, due to the subsequent amplification needed in the circuit, adding to the need to convert the current to voltage, a higher MOSFET current is preferred. Since the conversion process is achieved by a resistor, the smaller the value needed, less noise is introduced into the signal. This characteristic further reinforces the suitability of the MKProt MOSFET.

3.1.2.1 MOSFET and CMISFET Testing

Further testing of the MOSFETs was done by coupling them with sensors. Firstly, in order to ensure that a sensor will work properly, it needs to go through a dry capacitance test that has to be performed before hand. Currently, the acceptable capacitance for a working sensor is within a predetermined range of 0.120 nF and 0.350 nF.The need for this pre-analysis process, presents a potential enhancement for the software, which lies in automating the detection and validation of the connected CMISFET sensor.



(a) Transfer characteristic of a tested sensor using the CD4007UB MOSFET.



(b) Transfer characteristic of a tested sensor using the MKProt MOSFET.

Figure 3.10: pH sensing performance of a tested sensor using the CD4007UB MOSFET evaluated through the transfer characteristic with 10 curves acquired every 20 seconds.

This extension could automatically asses the capacitance of the sensor when initializing the software. By employing a real-time validation process the user could be immediately alerted in case of a sensor falling outside the specified capacitance range. Such an advancement would enhance the device's reliability, reduce the risk of inaccurate measurements, elevating the software's usability and efficiency.

When incorporating a sensor into the experimental setup for the electrical analysis, as illustrated in Appendix A, a crucial step involves establishing the connection between the sensor and the rest of the components. This connection was established using a ZIF socket and subsequently tested to ensure its integrity. The resistance measured between the front contact of the CMISFET and the wire within the ZIF socket was approximately 1.8 kΩ, while the corresponding wire connected to the sensor's back contact exhibited a resistance of approximately 2.5 kΩ. These resistance values suggest that the connection between the socket and the sensor is not optimal. In order to enhance data integrity, efforts could be made towards improving this connection, minimizing resistance in these areas to minimize data loss. Additionally, the sensor underwent a dry capacitance test, yielding a capacitance value of 0.350 nF.

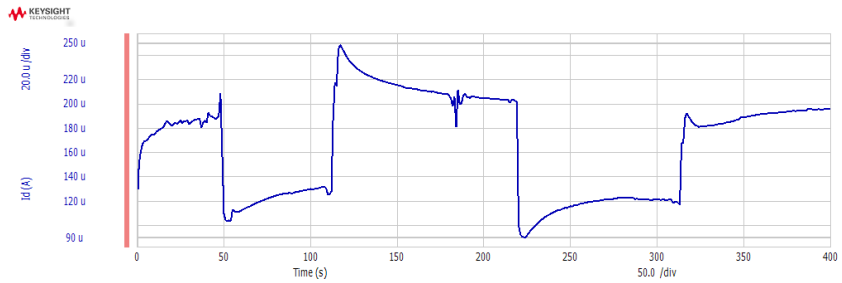
As before, voltage sweeps are conducted with the back contact of the sensor connected to the gate of the MOSFET, as can be seen in Appendix A. Figures 3.10a and 3.10b reaffirm the findings from the transfer characteristic curves of the MOSFETs. It can be seen again that the MKProt MOSFET exhibits a significantly broader range of drain current, measuring approximately 22.959 mA, whereas the CD4007UB MOSFET only exhibits a range of 368.83 µA.

Real-time measurements conducted with each MOSFET, seen in Figures 3.11a and 3.11b, demonstrate that the sensor coupled with both MOSFETs functions as intended. However, notable differences arise when considering the response to pH4 and pH10 buffer solutions. For the CD4007UB set-up, when exposed to the pH4 buffer solution, the drain current stabilizes at approximately 200 µA, whereas the MKProt set-up stabilizes at approximately 18 mA. Similarly, when exposed to the pH10 buffer solution, the CD4007UB set-up stabilizes at approximately 130 µA, while the MKProt set-up stabilizes at approximately 16 mA. This discrepancy reveal that for the CD4007UB set-up, the variation between the two buffer solutions is approximately 70 µA, while for the MKProt set-up, the variation is approximately 2 mA. These results reaffirm the previous findings, highlighting the custom-made MOSFET's capacity to deliver a considerably broader range of response under identical input conditions, solidifying its position as the optimal choice for this application.

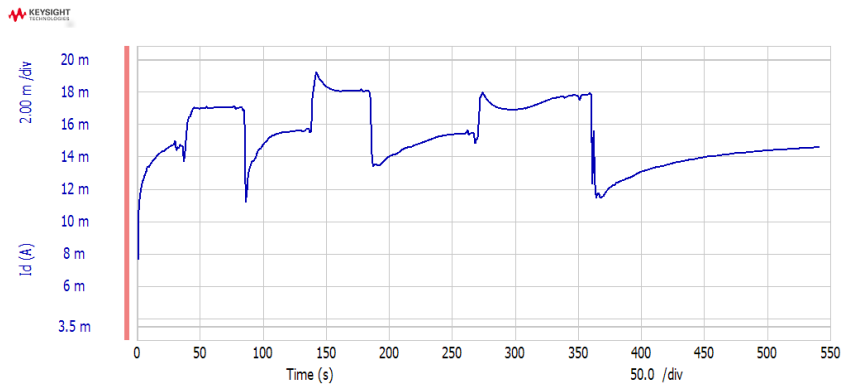
For further analysis of the MOSFETs, and in order to choose the optimal component, testing with each option had to be done with the proposed circuit. However, because of the pin configuration of the MKProt, it was not possible to integrate the component in the previously developed versions, 1.1 and 1.2.

Additionally, the fact that MKProt contains ten MOSFETs, could present an area for future development, where efforts could be made to integrate more than one sensor in the

testing. However, this would entail modifications both in the PCB and in the software.



(a) Continuous real time response of a tested sensor using the CD4007UB MOSFET.



(b) Continuous real time response of a tested sensor using the MKProt MOSFET.

Figure 3.11: pH sensing performance of a tested sensor using CD4007UB and MKProt MOSFETs evaluated through continuous real time response of I_d at fixed V_{cg}.

3.1.3 PCB Version 2

With the need to properly test the newly developed MOSFET and to overcome the complications encountered in the first versions of the PCB, the University of Macao stepped in to create a new version of the circuit. Drawing from the insights gained from the previous versions and their tests, a series of requisites and optimizations were identified for this new version.

Several crucial modifications were drawn to ensure proper functioning of the circuit. The source of the MOSFET was connected to ground to ensure proper functioning of the transistor. Analog potentiometers were replaced with digital ones, allowing for seamless adjustment of the resistance value, eliminating the need for users to make physical circuit modifications. These two separate potentiometers were connected to different pins for independent control. The sensor had to be strategically positioned at the edge of the PCB, facilitating the connection and replacement of the sensor. The PCB design would also allow for the accommodation of both the commercial MOSFET CD4007 or the custom MOSFET, enabling comparisons between the two options using the same assembly. Additionally, a voltage regulator was integrated allowing for two voltage supplies instead of just one, meeting the requirements of certain components within the circuit. By applying these

modifications and considerations, the second version of the PCB was fabricated with the circuit shown in Figure 3.12. As was seen before, the feedback resistor, in this version the R_6, is the component that controls the conversion of the signal from current to voltage. Moreover, the value of this resistor will affect directly the sensibility of the whole device.

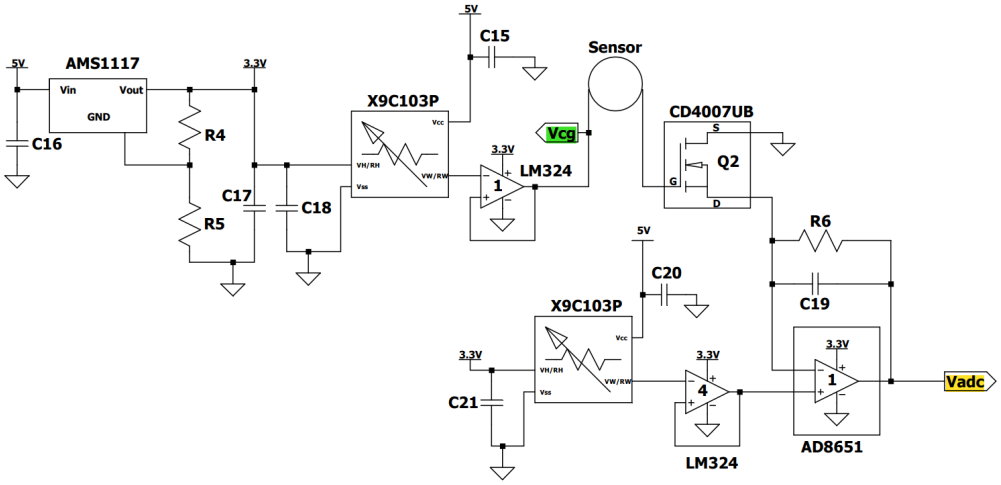


Figure 3.12: Schematic of the theoretical circuit design of the PCB version 2 portraying the connections between components and highlighting output voltage (V_{adc}) and sensor input voltage (V_{cg})

The corresponding developed PCB can be seen in Figure 3.13. As before, the V_{cg} is controlled through the X9C103P component. The V_{adc}, highlighted in yellow, is the output of the circuit, which is what should translate the response of the sensor and then be transferred to the Arduino. It's worth noting that, due to time constraints, this version was not tested and validated within the scope of this thesis. Further testing needs to be done in order to validate the functionality of this PCB, test both MOSFET options and draw conclusions on the optimal component.

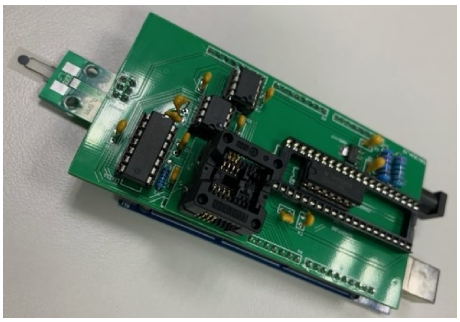


Figure 3.13: Front view of PCB version 2 portraying the spatial arrangement of components on the top side of the board.

3.2 Software Development

In this section, the results from each phase of the software development process are presented and discussed. It starts with requirements gathering, then software analysis, implementation and finally testing. These insights illustrate the development of a custom-made software solution for this project.

3.2.1 Software Requirements

In the initial phase, an understanding of the software's essential functionalities and user needs was achieved. From the point of the supplier, the examination of existing systems, such as the Agilent 4155 C semiconductor parameter analyser and Cascade Microtech M150 microprobe station, altogether with Keysight EasyEXPERT group+ software, provided a list of requirements, guiding the software's development [12]. From the point of the user, the primary objective was to simplify the process of using the sensor and the PCB, ensuring that the software was designed to enhance usability and provide a user-friendly experience [12]. This required the software to simplify the sensor's usage, incorporating clear guidance and an intuitive design throughout the software, explaining all steps and procedures within the GUI.

3.2.2 Software Analysis

The analysis of the requirements were taken into account to ensure it could handle a range of essential tasks. The software had to support real-time monitoring and voltage sweeps. It needed to allow users to input parameters, generate *.ino* files and ensure smooth communication with the Arduino for accurate data acquisition and storage. It needed to perform automated and user-instructed data manipulation and calculations, presenting data through options such as dataframes and plots. Data storage options could include CSV, PNG, or JPEG. Error handling mechanisms in the software could address issues with user input issues, providing informative error messages and minimizing data acquisition problems.

This analysis also allowed to comprehend the intricate details on how the software would interact with the user and all the hardware components. The analysis then resulted in the development of a flowchart, presented in Figure 3.14, which served as a reference point throughout the development process, facilitating a systematic approach to software design. The diagram represents the interaction of every component of the system, from the user, the software, the microcontroller, the PCB and lastly the sensor, and how the data flows through the components. It illustrates the sequential progression from user-defined analysis parameters to software processing, Arduino commands generation, voltage application, sensor response, data acquisition and subsequent data manipulation and presentation options.

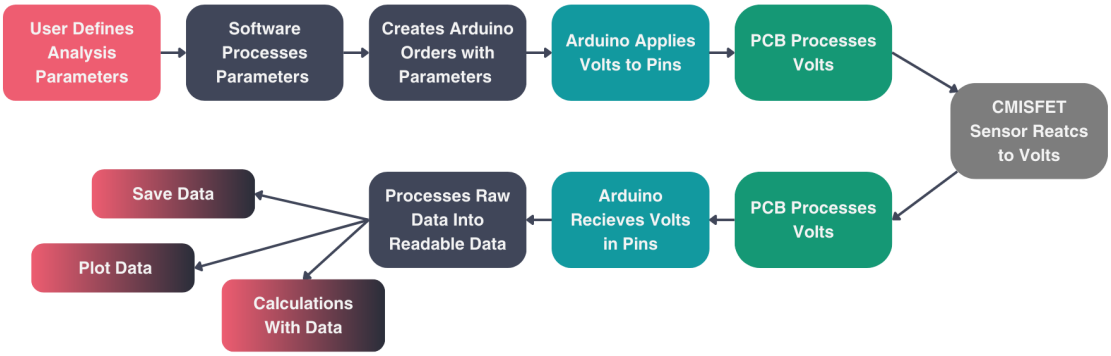


Figure 3.14: Flowchart of data flow and component interactions, from user-defined parameters through sensor reaction to data visualization and manipulation within the DAQ system.

3.2.3 Software Implementation

In this phase, the design specifications outlined during the analysis phase were transformed into functional code and software modules. The software code was divided into modular components which communicate and collaborate with each other. This enables for the software to be developed and tested in stages. These modules encompassed user-device interaction, data communication and data analysis, all contributing to the software's seamless functionality.

For this modular approach, code was divided in five different modules, *main*, *Sensor*, *GUI*, *File* and *Graph*, seen in Figure 3.15. This design offers multiple advantages that contribute to the efficacy and efficiency of the software all the while giving it the flexibility to grow alongside the project's requirements.

Firstly, the modular organization brings a clear organization of the code since it allows each module to handle specific functionalities. This way the complexity of the code is reduced and its readability is increased. The separation of tasks ensures that each module, including the class module, is focused on one job, easing the management of the entire system. This advantage not only benefits the original developer, but also future ones, since the comprehensibility is higher and navigation of the software is much easier. The modular structure also ensure reusability, specially through the class module which includes a well-defined set of attributes and methods. This class acts as a blueprint for the analysis that can be reused across the project, which avoids redundancies and optimizes development time. Furthermore, the segmentation of the modules allows thorough testing and accurate debugging. By isolating the tasks of each module, it is possible to make a more careful assessment, making both bug detection and fixing more efficient. Additionally, this design accommodates enhancements and extensions to be easily added to the software. This would allow for addition of new features and modifications, such as different types of analysis, without major modifications to the entire system.

As can be seen in Figure 3.15, the software is initialized by the *main* module, which in turn will control and interact with the *Sensor* class. Through this class it is possible to store

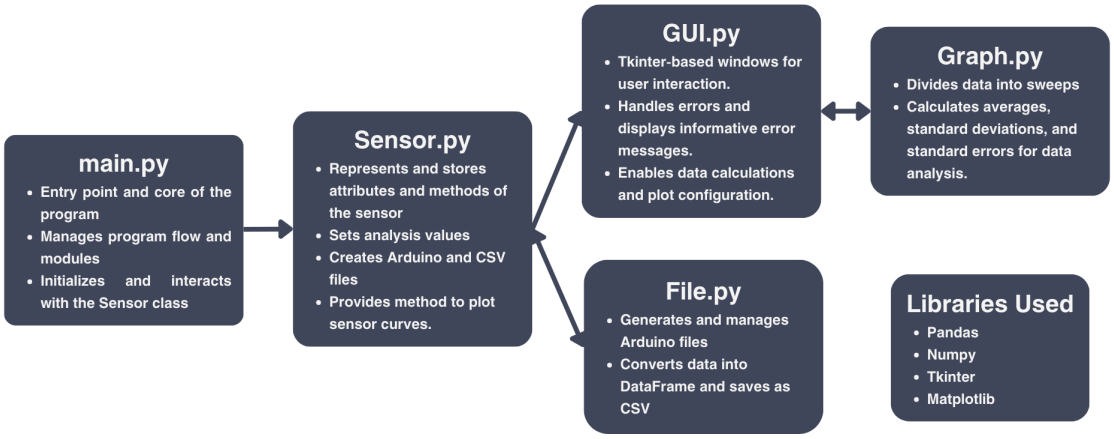


Figure 3.15: Overview of software modules, their functions and interactions among them within the system and main libraries used.

the data of an analysis in its attributes, such as type of analysis and voltage parameters, and apply methods that use the parameters to perform various tasks. These methods can then be called from the *main* module to progressively advance throughout the flow of the software, as previously seen in Figure 3.14.

The *GUI* module enables user interaction by displaying various windows for users to input parameters and displaying information. These windows provide essential instructions for users to comprehend the steps being taken and gather the necessary parameters. Subsequently, other modules take over the processing and management of the retrieved information.

The *File* module is tasked with the management of the different files that need to be generated and handled through the software. Consequently, this module is responsible for generating Arduino files containing instructions for this microcontroller, as well as storing the data retrieved from the Arduino in a CSV file.

Finally, the *Graph* module manages all data processing after retrieval and before presenting it in plots. This module also handles user-requested data manipulation, such as calculating averages or standard deviations and categorizing data for improved visualization.

These modules also make use of a range of Python libraries to enable and enhance their capabilities. The Pandas library allows for powerful data manipulation and analysis tools, which helps in handling structured data [19]. Numpy assists mathematical calculations, including large and multi-dimensional arrays and matrices [20]. Matplotlib aids in the creation of detailed and high-quality plots, enhancing data presentation [21]. Lastly, Tkinter serves as a foundation for the GUI, assisting in the creation of user-friendly interface elements [18]. Additional files containing the Arduino commands are written through the Arduino IDE taking advantage of the X9C10X library that enables the programming of the X9C103P potentiometer [27].

3.2.4 Software Testing

Rigorous testing was conducted to cover every aspect of the software and identify potential issues and bugs. The software's robustness and reliability were assessed and this phase served as a critical checkpoint, ensuring that the software met its intended objectives and functioned smoothly. Importantly, any issues that arose during testing were addressed, either through direct resolution or by revisiting the analysis and implementation phases.

3.3 Software Walk-through

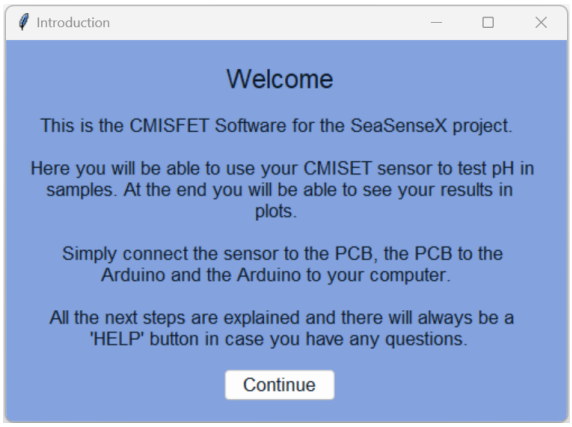
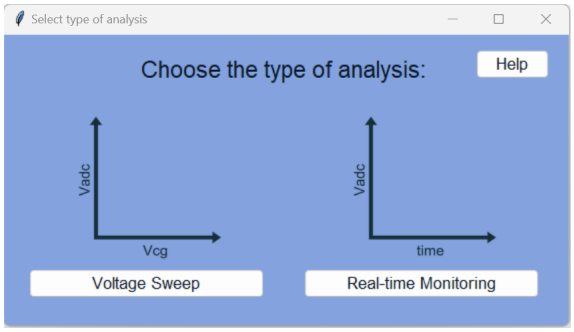
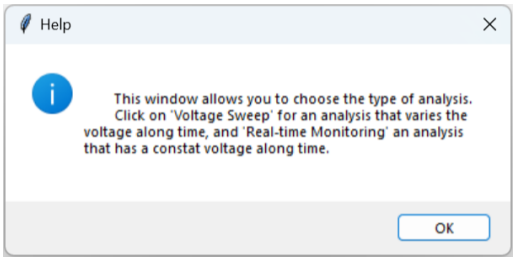


Figure 3.16: Snapshot of the software's graphical user interface displaying the introductory window, providing a brief explanation of the software's functionality.

Once the software is launched, users are presented with the initial page, Figure 3.16, that provides an overview of the functionality of the software and guides the users to ensure the necessary connections are established. These include the connection of the PCB, the Arduino and the CMISFET sensor.



(a) Snapshot of the software's graphical user interface displaying the selection of analysis type, where two choices are shown: Voltage Sweep and Real-Time Monitoring

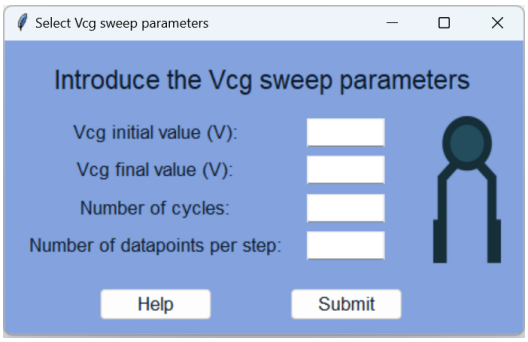


(b) Snapshot of the software's graphical user interface displaying the *Help* message for selection of analysis type.

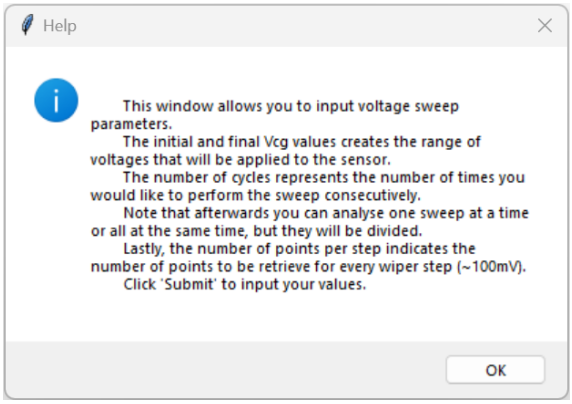
Figure 3.17: Snapshots of the software's graphical user interface displaying the window for selection of analysis type and corresponding *Help* message.

After the introductory page, two options are shown in another window, Figure 3.17a, which lets the user choose the type of analysis being made: voltage sweep analysis or real-time analysis. In addition to the two buttons for the analysis, a *Help* button makes a new window appear that explains the differences between the the two types of analysis presented, Figure 3.17b.

By selecting the voltage-sweep analysis, window in Figure 3.18a appears, and for real-time analysis, window in Figure 3.20a appears. Each window allows the user to specify the parameters for the chosen analysis.



(a) Snapshot of the software's graphical user interface displaying the window for parameter entry for Voltage Sweep Analysis. Users can input specific parameters, such as initial and final voltages, number of cycles and datapoints per step.



(b) Snapshot of the software's graphical user interface displaying the *Help* message for parameter entry in voltage sweep analysis.

Figure 3.18: Snapshots of the software's graphical user interface of the window for parameters entry for voltage sweep analysis and corresponding *Help* message.

For the voltage sweep these parameters include initial and final V_{cg} voltages, number of voltage steps and number of points to be acquired per step. This window also has a *Help* button explaining what each of these parameters represent, which opens a new window that explains these parameters and the format they must have, Figure 3.18b. For this option, the software is designed to handle various user errors when clicking the *Submit* button. It checks if the entered V_{cg} values are valid by ensuring those are either whole or decimal numbers, using a period (".") and not a comma (","). Similarly, it validates the number of cycles and points per step values, confirming those are whole number and not a decimal values. Then it ensures that all values, except for the initial V_{cg}, are positive. The initial V_{cg} value can be either zero or positive. It is then confirmed if the V_{cg} initial value is less than the V_{cg} final value. Lastly, it prevents the final V_{cg} value from exceeding a predefined threshold, determined by the Arduino and components used in the circuit. If any of these checks fail, an informative pop-up window indicates the specific issue to the user, letting them correct the mistake. A snapshot of one of these messages can be seen in Figure 3.19. The remaining error messages can be found in Table 3.1

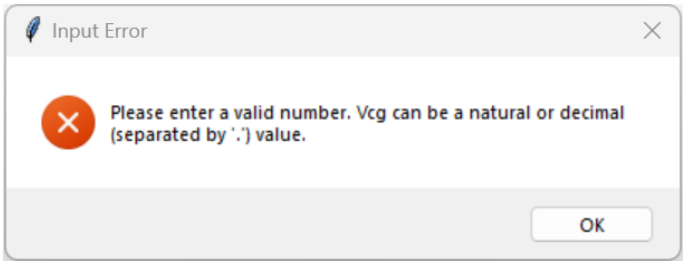
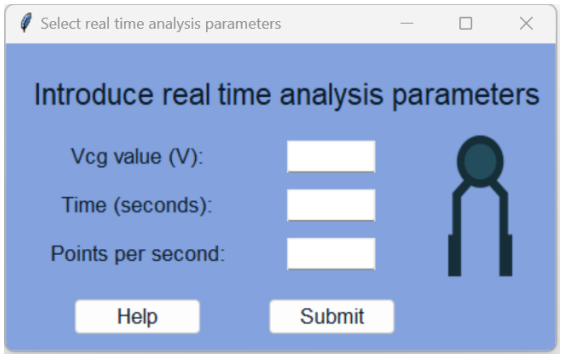


Figure 3.19: Error message for incorrect Vcg value.

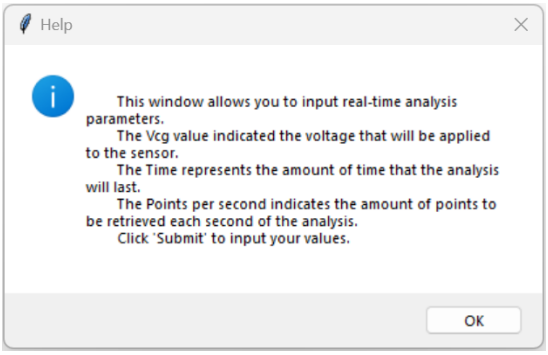
Table 3.1: Error messages for incorrect entry of voltage sweep analysis parameters.

Error	Error Message
Incorrect cycles value.	Please enter a valid number. Cycles needs to be a natural value.
Incorrect data points value.	Please enter a valid number. Data points needs to be a natural value.
Incorrect difference between Vcg values.	Vcg initial value needs to be smaller than Vcg final value.
Vcg value too high.	Vcg final value needs to be smaller or equal to 5 V.
Non-positive values.	Please enter only positive values.

For the real-time monitoring the user only needs to define the voltage at which the analysis is being made, the time that the analysis is supposed to last in seconds and the number of points to be acquired for each second. As before, this window also has a *Help* button explaining what each of these parameters represent and the correct way to input the parameters. Both of these windows can be seen in Figure 3.20.



(a) Snapshot of the software's graphical user interface displaying the window for parameters entry for Real-Time Analysis. Users can input specific parameters, such as voltage value, duration and points per second.



(b) Snapshot of the software's graphical user interface displaying the *Help* message for parameter entry for Real-Time Analysis.

Figure 3.20: Snapshots of the software's graphical user interface of the window for parameters entry for Real-Time Analysis and corresponding *Help* message.

For this step, the software is also designed to handle various user errors when the *Submit* is clicked. It checks if the entered V_{cg} value is valid, ensuring it is either a whole or decimal number, using a period (".") and not a comma (","). It validates the number of cycles and points per step values, confirming they are whole numbers and not decimal. It also ensures all values, except for V_{cg} value, are positive. The V_{cg} can be zero or positive. Finally it is verified if the V_{cg} value is not exceeding the predefined threshold voltage,

which is the same as before. Again, if any of these checks fail, an informative pop-up window indicates the specific issue and how to correct it. These messages follow the format seen in Figure 3.19 and the specific errors and corresponding messages are presented in Table 3.2.

Table 3.2: Error messages for incorrect entry of real-time analysis parameters.

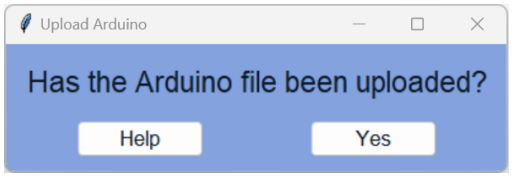
Error	Error Message
Incorrect Vcg value.	Please enter a valid number. Vcgcan be a natural or decimal (separated by ".") value.
Incorrect time value.	Please enter a valid number. Time needs to be a natural value.
Incorrect points per second value.	Please enter a valid number. Points per second needs to be a natural value.
Vcg value too high.	Vcg final value needs to be smaller or equal to 5 V.
Non-positive values.	Please enter only positive values.

By introducing these error handling mechanisms, the software has an enhanced user experience that prevents multiple potential common input mistakes. Since the software not only detects these errors and alerts the user, it also provides detailed error notifications, it allows users to easily identify and rectify issues. This validation process helps ensure accurate and valid data inputs, increasing the reliability of the analysis outcome and enhancing the overall usability and credibility of the software.

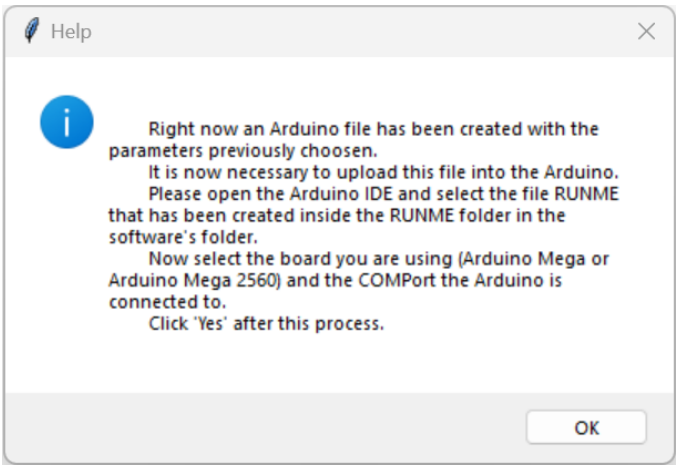
Once the user submits the analysis parameters of either type of analysis, an Arduino sketch file (.ino) is generated automatically. This file is created by customizing a template .ino file, according to the specified information, and stored in a specific folder that can be accessed by the user. The template files for this step, one for the voltage sweep and another for real-time analysis, serve as a foundation for creating the desired file, as they contain the basic instructions for the micro-controller and only a few specific parameters need to be changed. These template files can be found in Appendix D.

Through this flexible approach, the software can easily accommodate the addition of new types of analysis in the future as well as easy modification of the existing analysis. Since the template is an external file, not contained in the Python code, it can be easily accessed to accommodate different pin configurations for different PCBs being used, or other variations to the device's requirements, without needing to change anything in the main code. This advantage allows for ongoing development, customization and adaptation of the Arduino files, ensuring the device remains adaptable to evolving research needs, expansion of the device's capabilities and potential future applications.

The subsequent window, Figure 3.21a, prompts the user to confirm whether the generated .ino file has been successfully uploaded to the Arduino board. While it would be advantageous for the software to have an automated upload process, currently, a manual upload is required. However, in addition to the *Submit* button, a *Help* button is available in the window. By clicking this button, a pop-up window, Figure 3.21b, provides step-by-step instructions for assistance with the upload process. This limitation highlights an area for future development, where efforts could be made in order to implement an automated upload feature to enhance user convenience and minimize potential errors.



(a) Snapshot of the software's graphical user interface displaying window for the confirmation of the Arduino file upload.



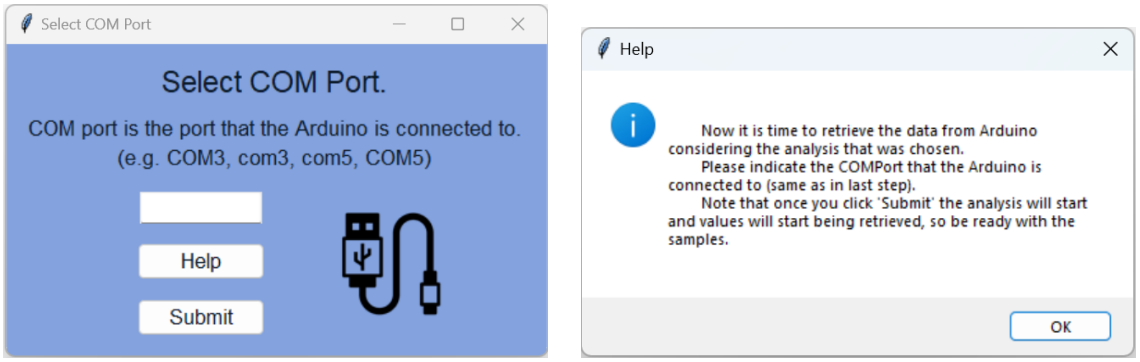
(b) Snapshot of the software's graphical user interface displaying the *Help* message for the confirmations of the Arduino file upload.

Figure 3.21: Snapshots of the software's graphical user interface of the window for confirmation of the Arduino file upload and corresponding *Help* message.

Following the upload of the .ino file and its confirmation, the software prompts the users to select the serial port (COM port) to which the Arduino board is connected, Figure 3.22a, allowing for a bidirectional data transfer. Once again to assist this step, a *Help* button is provided, that creates a pop-up window, Figure 3.22b, offering instructions and insights for this step. In this step, ideally the software would automatically identify the port, however, currently the user needs to provide that information manually. This limitation highlights an area for future development, where efforts could be made in order to implement an automated identification feature to enhance user convenience and minimize potential errors or complications.

Once the connection is established, the Arduino board initiates the instructions contained in the .ino file. This means that the analysis has started and the user can start applying the samples that will be analysed. These values are automatically collected and stored in a dataframe for further manipulation. For data analysis, the software is able to employ multiple calculations tailored to the specific measurement requirements, which enables the extraction of relevant information. An important data conversion that is applied in this step is converting the data read from the Arduino analog pins, that are presented in bits, into voltage values.

Since the Arduino Mega contains a 10 bit analog-to-digital converter [28], then the values read in the analog pins will range from 0 to 1023, since $2^{10} = 1024$. This range



(a) Snapshot of the software's graphical user interface displaying window for communication port entry.

(b) Snapshot of the software's graphical user interface displaying the *Help* message for communication port entry.

Figure 3.22: Snapshots of the software's graphical user interface of the window for communication port entry and corresponding *Help* message.

corresponds to the voltage range, from 0 to the operating voltage (5.0 V or 3.3 V). Since the operating voltage is 5 V, then it is possible to calculate the voltage being read by the analog pin through equation 3.1. This step of the process is crucial as it converts raw numerical readings into meaningful voltage units. This increases interpretability of the data for the user when analysing the data.

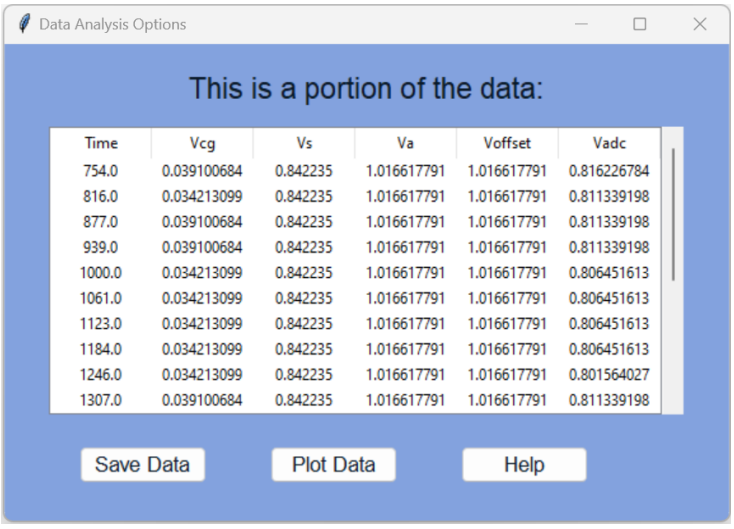
$$Voltage = (AnalogValue/1023) * 5.0\,\text{V} \tag{3.1}$$

After applying the initial calculations, the cleaned data can be seen by the user in a window which shows the first twenty lines of data, as can be seen in Figure 3.23a. This preview of the data shows important details about the data, such as data format, units of measurement, timestamps and any additional information associated with the collected data. The preview lets the user quickly asses the quality and nature of the data before proceeding with the next steps. This feature also enhances efficiency and informed decision-making during the data analysis process. The data presented in Figure 3.23a is a sampled retrieved from results of previous works. This example presents multiple voltage columns that, which is aligned with the voltages being retrieved through Version 1.2 of the PCB. However, thanks to the code's adaptability, the format of the data is not restricted to a limited number of columns or lines. This means that the PCB versions can be continuously improved, and the code can automatically detect the change in format and change its appearance accordingly.

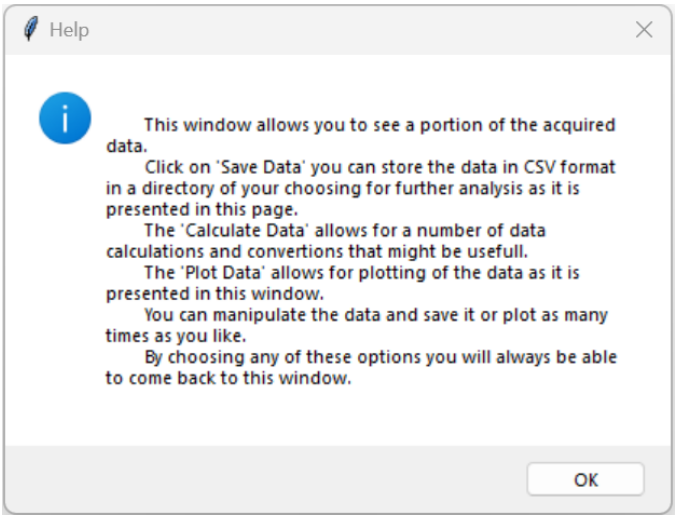
In addition to the table, two options are available at the bottom of the window, enabling the user to take further actions with the data. This window also contains a *Help* button that provides an explanation about the displayed data, along with an explanation of what the different options in the window will allow the users to perform, Figure 3.23b.

The *Save Data* button enables the user to easily store the data in a directory of their choosing in CSV format. This allows for easy extraction of the data, letting the user perform

additional manipulation, calculation and plotting, using their preferred software or tools, such as Excel or Origin. By providing this flexibility in data storage, users can easily incorporate the obtained data into their existing processes and perform more in-depth analysis or create customized reports without limitations. This feature also improves the software's adaptability and usability of the software, facilitating data exploration.



(a) Snapshot of the software's graphical user interface displaying window for data analysis. Window portrays a sample of the data and options for next steps.

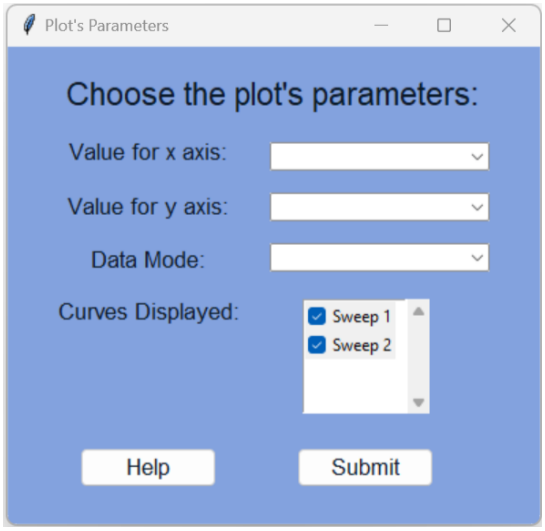


(b) Snapshot of the software's graphical user interface displaying the *Help* message for data analysis.

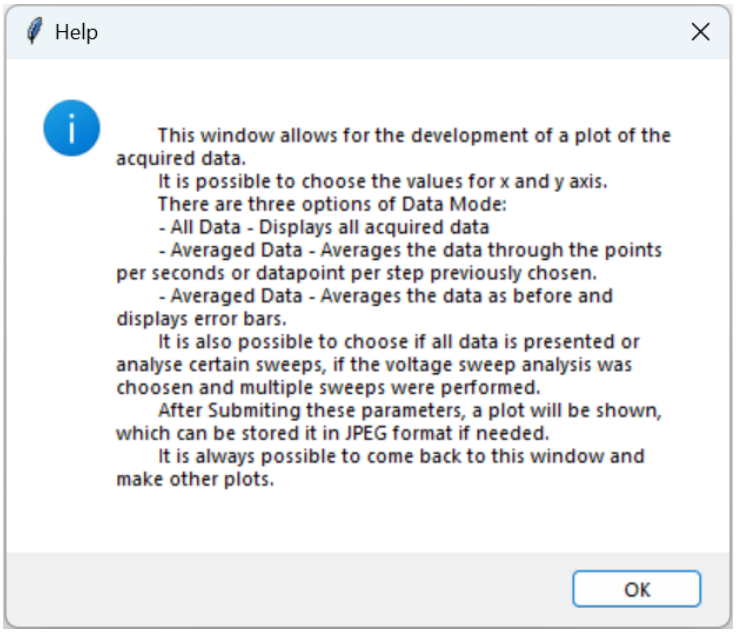
Figure 3.23: Snapshots of the software's graphical user interface of the window for data analysis and corresponding *Help* message.

The *Plot Data* button opens another window, depicted in Figure 3.24a, which enables users to customize the visualization of their data through plots. Through this window, users can specify the x and y axis values, select between displaying all data or averaged data. Additionally, users can choose which specific sweeps to display. In case of voltage sweeps, there are multiple choices of sweeps, whereas for real-time analysis, there will

only be one option. For all of these parameters, users are restricted to selecting from a predefined list of options, which is determined from the data that was acquired from the Arduino. This approach not only helps the decision-making process, but also prevents potential errors and increases consistency between analysis. This window also contains a *Help* button that provides an explanation for the choices presented to the user, Figure 3.24b.



(a) Snapshot of the software's graphical user interface displaying the window for parameters entry for plotting options. Users can choose specific parameters, such as x axis and y axis values, data mode and curves displayed.



(b) Snapshot of the software's graphical user interface displaying the *Help* message for parameter choosing for plotting.

Figure 3.24: Snapshots of the software's graphical user interface of the window for parameters entry for plotting and corresponding *Help* message.

When the *Submit* button is clicked the software employs an error checking mechanism. This process involves validation of the selected parameters, such as checking whether any of the parameters have been left unspecified, prompting an error message that guides the user to input the specific parameter. It also informs the user in case the the x and y values are the same, ensuring that the displayed data is meaningful. However, if the user wishes to the plot can be presented with x and y axis for the same value. This process enhances user experience that prevents potential input mistakes. As before, since the error messages are customized to the specific error that occurred, the user can easily rectify the problem. These error messages follow the format of Figure 3.19 and the specific errors and corresponding messages can be found in Table 3.3.

Table 3.3: Error messages for incorrect entry of plotting parameters.

Error	**Error Message**
Empty axis entry.	Please choose values for both axis.
Empty data mode entry.	Please choose a data mode.
Same entries of x and y axis.	Are you sure you want the same values for x-axis and y-axis?

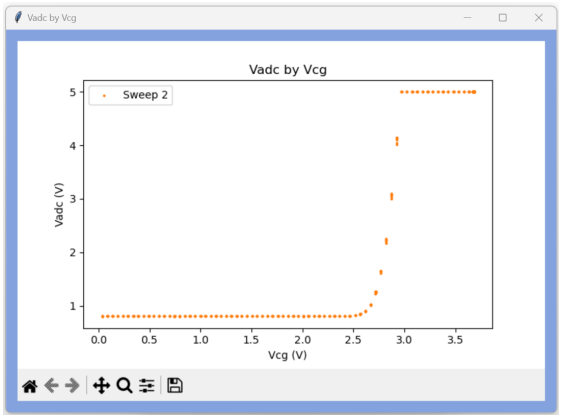
When the parameters have all been selected and verified, the software generates the specified plot that provides a visual representation of the results, aiding in the interpretations of the data. Furthermore, thanks to the implementation of tkinter, this window includes a versatile navigation bar on the bottom left corner, driven by the NavigationToolbar2Tk module. This feature allows users to easily interact with the plotted data, improving the usability of the software. Functionalities include zooming in and out, panning across the plot, switching between multiples aspects of the plot and exporting the plot in various formats (PNG, JPEG) for further analysis.

These functionalities and the different types of plots the software is able to create can be seen in Figure 3.25. It is possible to conclude that by averaging the values, the curves are easier to analyse, and the possibility to analyse multiple sweeps at the same time allows for fast comparison.

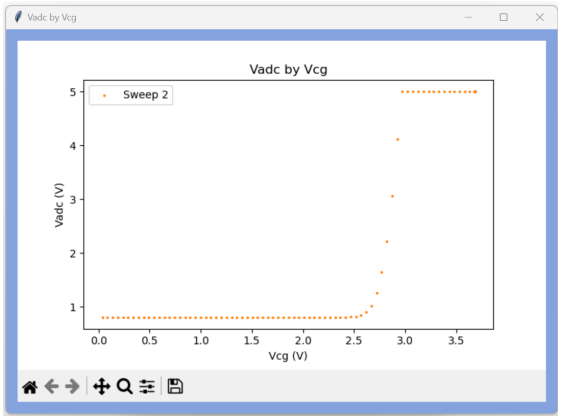
By closing either the *Save Data* or the *Plot Data* windows, the software returns to the data visualization window, Figure 3.23. This allows users to continue their data analysis process and maintain a fluid and uninterrupted workflow.

The developed software was created with the first schematic of the PCB in mind which implies controlling only one digital potentiometer. For future steps in the project, the software needs to be modified in order to accommodate the third version of the PCB. Since the software was developed in modules, as mentioned previously, only the .ino scripts need to be changed in order to be adapted to this newer version.

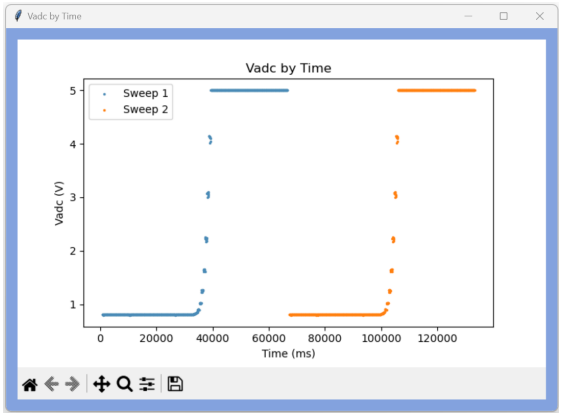
To conclude, the software was successfully developed allowing for efficient control of a DAQ system and analysis of its response. The software successfully gathers, transforms and displays the data given users response, while allowing for different PCB being used. The software can be easily adapted and upgraded in order to accompany the changes and improvements made in the projects and hardware components. The software



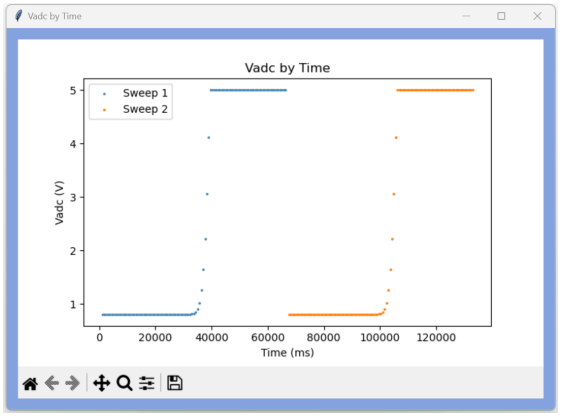
(a) Snapshot of the software's graphical user interface displaying a plot of sample data with one sweep using all data of V_{adc} by V_{cg}.



(b) Snapshot of the software's graphical user interface displaying a plot of sample data with one sweep averaging the data by points per step of V_{adc} by V_{cg}.



(c) Snapshot of the software's graphical user interface displaying a plot of sample data with two sweeps using all data of V_{adc} by Time.



(d) Snapshot of the software's graphical user interface displaying a plot of sample data with two sweeps averaging data of V_{adc} by Time.

Figure 3.25: Snapshots of the software's graphical user interface displaying multiple plotting examples of sample data using some of the software's functionalities.

successfully emulates the laboratory instruments necessary for the measurement of the sensors response, bringing the objective of the project, the development of a fully portable device, one step closer to completion.

3.3.1 Software Distribution

The software can be found in a public GitHub repository available for download [29]. Any user can have access to the code, use it and make requests for improvements and bug corrections. This way it is possible to make a continuous improvement of the platform.

4

Conclusions

In the course of this work, a custom software tailored to the SeaSenseX project's data acquisition device was successfully developed. This custom software offers several advantages over commercially available options, as it aligns its functionalities precisely with the project's unique requirements. Unlike off-the-shelf options with limited customization options, this software grants complete control and adaptability, enabling fine tuning of every aspect to suit the project's needs. This adaptability ensures that the software can easily incorporate modifications and enhancements while seamlessly integrating with all hardware components and their continuous optimizations. The structured approach of the development and software architecture proves instrumental in creating a reliable and robust custom solution.

Simultaneously, critical evaluation of essential hardware components, including MOSFETs, sensors and PCBs, was conducted. The analysis highlighted the significance of MOSFET selection, with the custom-made MKProt providing a milliampere response range, surpassing the CD4007UB's microampere range. Regarding the PCB, the initial version encountered issues with the number of sensors, leading to the analysis of the upgraded version, which addressed most of the inconsistencies. A custom second version, developed by researchers at University of Macao, integrated critical modifications, including MOSFET pin re-connections, digital potentiometers, strategic positioning of the sensor, support for both commercial and custom MOSFETs and a voltage regulator. Due to time constraints, the third version was not tested within this thesis.

4.1 Future Perspectives

As this thesis concludes, several opportunities for improvement and further research come to light:

- Code Enhancements:
  - Implementation of automatic detection and communication of the microcontroller.

  - Implementation of a verification step to ensure correct functioning of all hardware components.
  - Adaptation to accommodate the new PCB design and maintain compatibility.
  - Integration of further data analysis options.

- Testing and Validation:
  - Laboratory testing and validation of Version 2 of the PCB to address any outstanding issues.
  - Testing and comparison both MOSFETs when incorporating them into the new PCB to address their performance characteristics and integration whit the rest of the device.
  - Development of more reliable connections between the sensor and the PCB to enhance device stability.
  - Testing the device with different samples and compare the results to the findings with laboratory instruments.
  - Consideration of in-situ testing of the complete device to assess its performance in real-world conditions.

These future perspectives aim to build on the foundations laid in this thesis, advancing the SeaSenseX project towards its ultimate objectives. Addressing these areas will further refine and optimize the system, making it a valuable asset for real-world applications.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf.

[2] Kiran, R. Bharti, and R. Sharma. "Effect of heavy metals: An overview". In: *Materials Today: Proceedings* 51 (2022). CMAE'21, pp. 880–885. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2021.06.278. URL: https://www.sciencedirect.com/science/article/pii/S221478532104668X.

[3] World Health Organization. *Almost 1 Million People Die Every Year Due to Lead Poisoning, with More Children Suffering Long-term Health Effects*. 2022-10. URL: https://www.who.int/news/item/23-10-2022-almost-1-million-people-die-every-year-due-to-lead-poisoning--with-more-children-suffering-long-term-health-effects.

[4] J. Briffa, E. Sinagra, and R. Blundell. "Heavy metal pollution in the environment and their toxicological effects on humans". In: *Heliyon* 6.9 (2020), e04691. ISSN: 2405-8440. DOI: https://doi.org/10.1016/j.heliyon.2020.e04691. URL: https://www.sciencedirect.com/science/article/pii/S2405844020315346.

[5] M. Kumar, N. Sawhney, and R. Lal. *2 - Chemistry of heavy metals in the environment*. Ed. by V. Kumar, A. Sharma, and A. Cerdà. Elsevier, 2021, pp. 9–37. ISBN: 978-0-12-821656-9. DOI: https://doi.org/10.1016/B978-0-12-821656-9.00002-X. URL: https://www.sciencedirect.com/science/article/pii/B978012821656900002X.

[6] R. Danovaro et al. "Implementing and Innovating Marine Monitoring Approaches for Assessing Marine Environmental Status". In: *Frontiers in Marine Science* 3 (2016). ISSN: 2296-7745. DOI: 10.3389/fmars.2016.00213. URL: https://www.frontiersin.org/articles/10.3389/fmars.2016.00213.

[7] G. Mills and G. Fones. "A review of in situ/IT methods and sensors for monitoring the marine environment". In: *Sensor Review - SENS REV* 32 (2012-01), pp. 17–28. DOI: 10.1108/02602281211197116.

[8] Faculdade de Ciências e Tecnologia Universidade Nova de Lisboa. *SeaTox Lab website*. 2023. URL: https://sites.fct.unl.pt/seatox/.

[9] G. Félix. "Production and electrical characterization of microsensors for marine mutagens and carcinogens monitoring". MA thesis. NOVA School of Science and Technology, NOVA University Lisbon, 2022.

[10] B. Veigas et al. "Quantitative real-time monitoring of RCA amplification of cancer biomarkers mediated by a flexible ion sensitive platform". In: *Biosensors and Bioelectronics* 91 (2017), pp. 788–795. ISSN: 0956-5663. DOI: https://doi.org/10.1016/j.bios.2017.01.052. URL: https://www.sciencedirect.com/science/article/pii/S0956566317300520.

[11] A. Sá. "Development of a controller for gene-expression analysis using field-effect devices". MA thesis. NOVA School of Science and Technology, NOVA University Lisbon, 2019. URL: http://hdl.handle.net/10362/90040.

[12] M. D. P. Emilio. "Software for Data Acquisition Systems". In: *Data Acquisition Systems: From Fundamentals to Applied Design*. New York, NY: Springer New York, 2013, pp. 115–121. ISBN: 978-1-4614-4214-1. DOI: 10.1007/978-1-4614-4214-1_5. URL: https://doi.org/10.1007/978-1-4614-4214-1_5.

[13] National Instruments. *What is LabVIEW?* https://www.ni.com/en/shop/labview.html. Accessed: 2023-07-02.

[14] National Instruments. *What is LabWindows/CVI?* https://www.ni.com/en/shop/electronic-test-instrumentation/programming-environments-for-electronic-test-and-instrumentation/what-is-labwindows-cvi.html. Accessed: 2023-07-02.

[15] The MathWorks, Inc. *MATLAB - Math. Graphics. Programming.* https://www.mathworks.com/products/matlab.html. Accessed: 2023-07-02.

[16] The MathWorks, Inc. *Data Acquisition Toolbox*. https://www.mathworks.com/products/data-acquisition.html. Accessed: 2023-07-02.

[17] G. Van Rossum and F. L. Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[18] G. Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[19] W. McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by S. van der Walt and J. Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[20] C. R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020-09), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[21] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[22] B. Stroustrup. *The C++ programming language*. Pearson Education India, 2000.

[23] C. G. Fonstad. "10 - Field Effect Transistors". In: *Microelectronic Devices and Circuits*. McGraw-Hill, 1994, pp. 265–316. ISBN: 9780070214965.

[24] *LM134/LM234/LM334 3-Terminal Adjustable Current Sources*. Texas Instruments. Dallas, Texas, 2013.

[25] T. Kugelstadt. "Chapter 20 - Active Filter Design Techniques". In: *Op Amps for Everyone (Third Edition)*. Ed. by R. Mancini and B. Carter. Newnes, 2009, pp. 365–438. DOI: https://doi.org/10.1016/B978-1-85617-505-0.00020-X.

[26] *CD4007UB Types, CMOS Dual Complemantary Pair Plus Inverter*. Texas Instruments. Dallas, Texas, 2003.

[27] R. Tillaart. *Arduino Library for X9C10X series digital potentiometer.* 2022. URL: https://github.com/RobTillaart/X9C10X (visited on 2023-08-11).

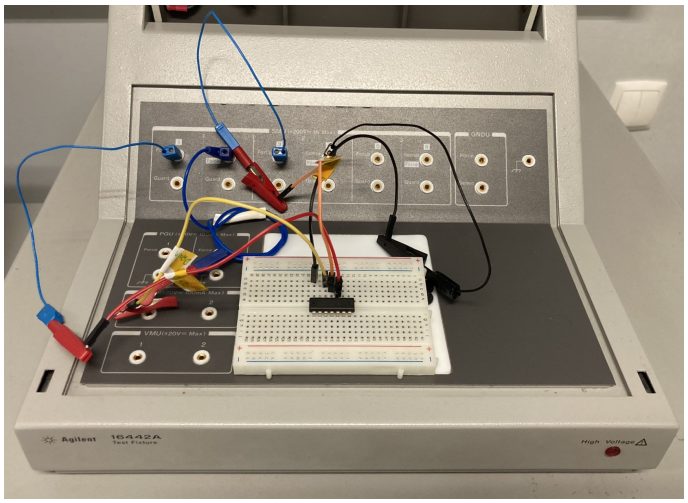
[28] *ADC of Arduino Boards*. 2019. URL: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ (visited on 2023-08-11).

[29] M. Alvarez. *Software for Controller for Marine Pollution*. 2023. URL: https://github.com/MariaMAlvarez/software-for-controller-for-marine-pollution (visited on 2023-09-22).

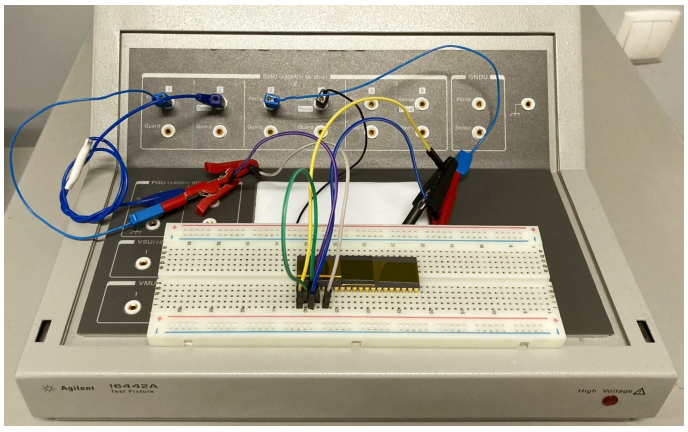
A

Experimental Testing setup

Here is presented the experimental setup for the testing of MOSFETs and CMISFET using the Agilent 4155C semiconductor parameter analyser and the Cascade Microtech M150 microprobe station.

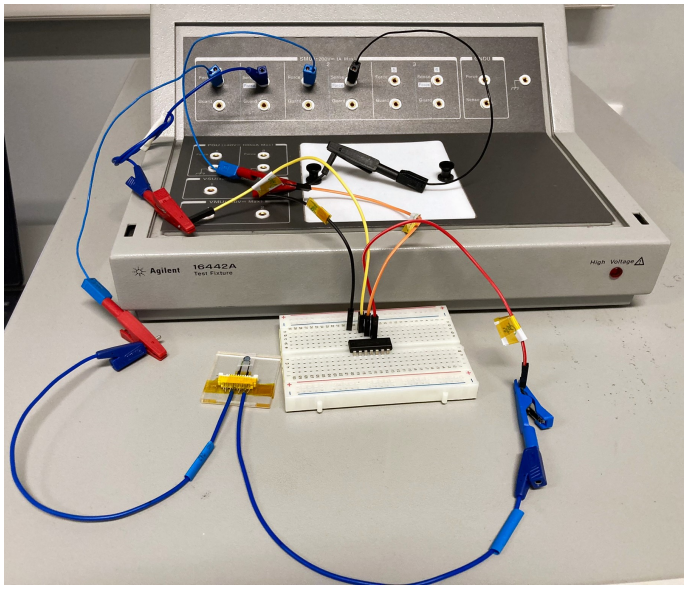


(a) Experimental testing setup for the CD4007UB MOSFET.

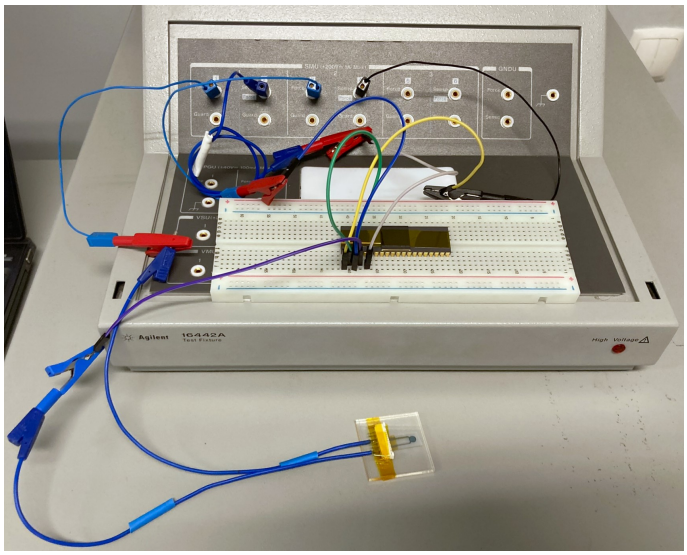


(b) Experimental testing setup for the MKProt MOSFET.

Figure A.1: Experimental testing setups for the CD4007UB and MKProt MOSFETs.



(a) Experimental testing setup for the CD4007UB MOSFET and CMISFET.



(b) Experimental testing setup for the MKProt MOSFET and CMISFET.

Figure A.2: Experimental testing setup for the CD4007UB and MKProt MOSFETs coupled with the CMISFET sensor.

B

Testing Script

To facilitate the testing process, a sample script is provided below, which can be executed using the Arduino IDE. This script offers a way to verify the functionality of the Arduino's data acquisition process.

```
#include <X9C.h>
#define UD 12
#define INC 13
#define CS 11
X9C pot;
int analog_in_vcg = A0, analog_in_vs = A1, analog_in_va = A2,
    analog_in_voffset = A3, analog_in_vadc = A4;
float vcg_volts, vs_volts, va_volts, voffset_volts, vadc_volts;
void setup() {
  Serial.begin(9600);
  pinMode(analog_in_vcg, INPUT);
  pinMode(analog_in_vs, INPUT);
  pinMode(analog_in_va, INPUT);
  pinMode(analog_in_voffset, INPUT);
  pinMode(analog_in_vadc, INPUT);
  pot.begin (CS, INC, UD);
  pot.setPot(50, true);
  delay(100);
  Serial.println("Time;Vcg;Vs;Va;Voffset;Vadc;/n");
  delay(500);
  }
void loop() {
  pot.setPot(28,true);
  unsigned long t = millis();
  Serial.print(t);
  Serial.print(";");
```

```
  volts_vcg = analogRead(analog_in_vcg);
  volts_vcg = volts_vcg * 5/1024;
  Serial.print(volts_vcg);
  Serial.print(";");
  volts_vs = analogRead(analog_in_vs);
  volts_vs = volts_vs * 5/1024;
  Serial.print(volts_vs);
  Serial.print(";");
  volts_va = analogRead(analog_in_va);
  volts_va = volts_va * 5/1024;
  Serial.print(volts_va);
  Serial.print(";");
  volts_voffset = analogRead(analog_in_voffset);
  volts_voffset = volts_voffset *5/1024;
  Serial.print(volts_voffset);
  Serial.print(";");
  volts_vadc = analogRead(analog_in_vadc);
  volts_vadc = volts_vadc * 5/1024;
  Serial.print(volts_vadc);
  Serial.print(";");
  delay(500);
  Serial.println("/n");
}
```

C

PCB V2 Circuit Schematic

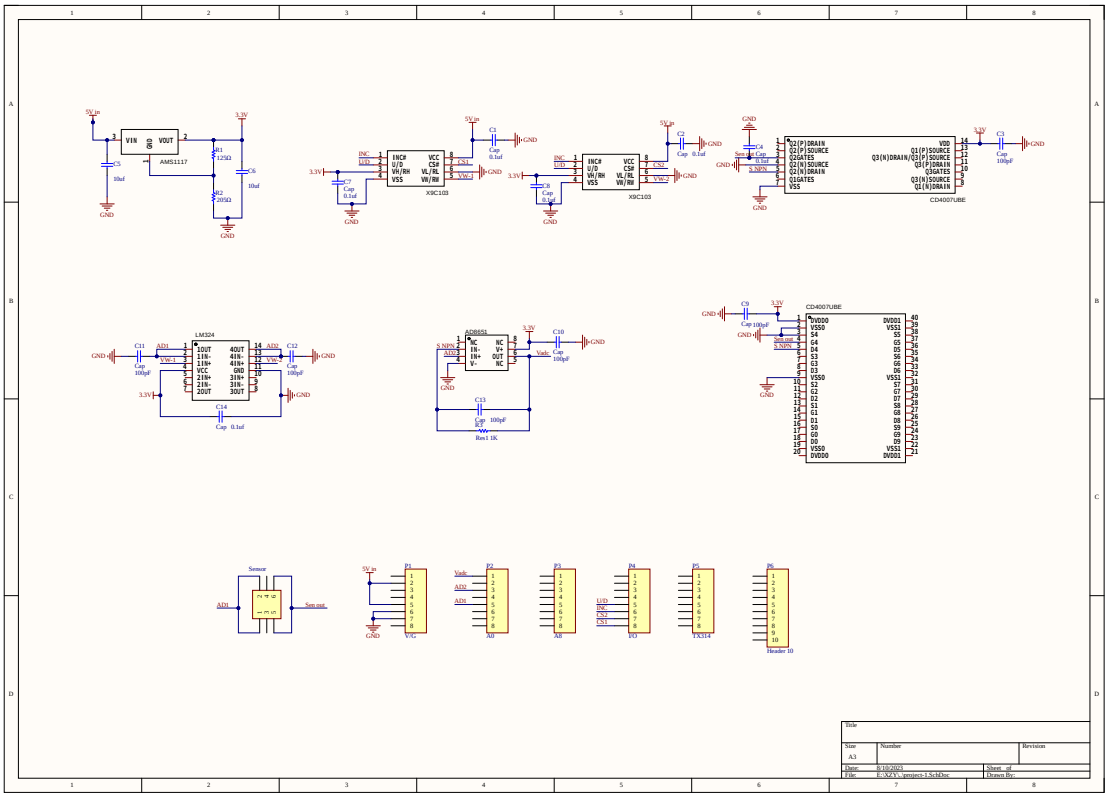


Figure C.1: PCB V2 Circuit Components.

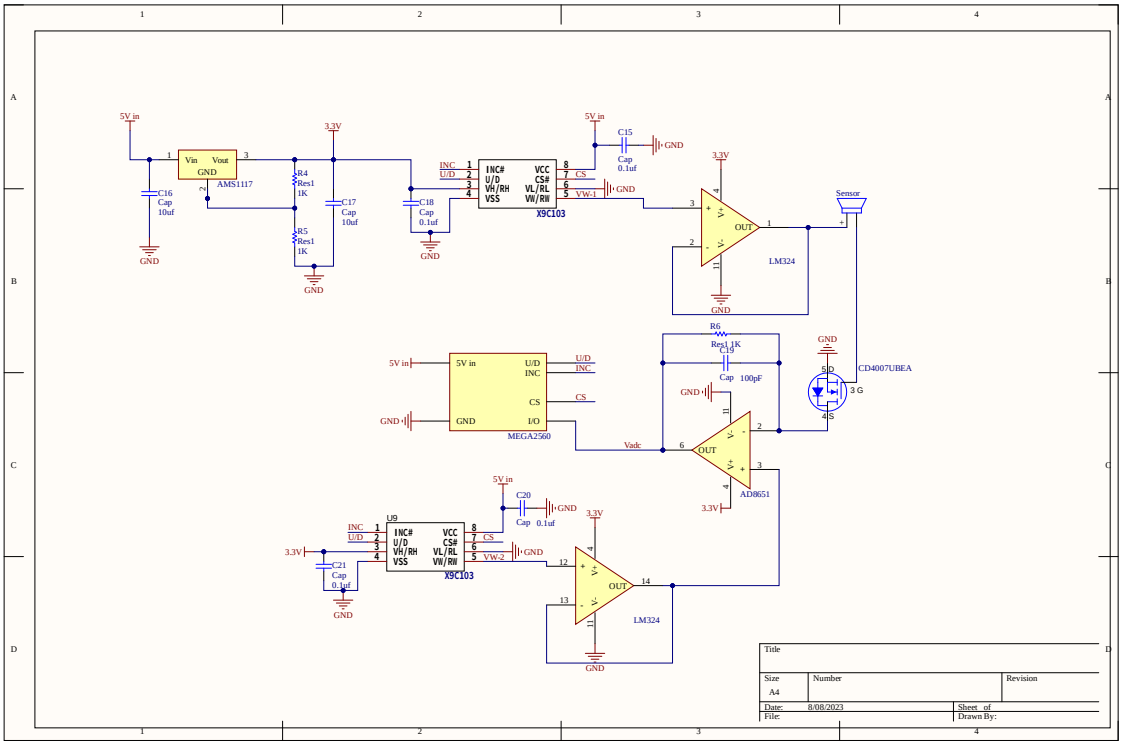


Figure C.2: PCB V2 Circuit Schematic.

D

RUNME Template

The script provided below applies a voltage sweep on the sensor and retrieves the output voltage of the circuit. Other voltages can be analysed if there is an Arduino pin dedicated to it.

```
#include <X9C.h>

// ----- PINS X9C -----//
#define UD  12
#define INC 13
#define CS  11
X9C pot;

// ----- PINS FOR ANALYSIS -----//
int analog_in_vcg = A0, analog_in_vs = A1, analog_in_va = A2,
    analog_in_voffset = A3, analog_in_vadc = A4;

void setup() {
  Serial.begin(9600);
  pinMode(analog_in_vcg, INPUT);
  pinMode(analog_in_vs, INPUT);
  pinMode(analog_in_va, INPUT);
  pinMode(analog_in_voffset, INPUT);
  pinMode(analog_in_vadc, INPUT);
  pot.begin (CS, INC, UD);
  pot.setPot(50, true);
  delay(100);
  Serial.println("Time;Vcg;Vs;Va;Voffset;Vadc;/n");
  delay(1000);
}
```

```
void loop() {
  for (int cycles=0; cycles<num_cycles; cycles++){
    pot.setPot(vcg_initial_value, true);
    delay(500);
    for (int steps=0; steps<wiper_steps; steps++){
      float t=millis();
      pot.trimPot(1, X9C_DOWN, true);
      delay (500);
      for (int data_points=0; data_points<num_data_points;
         data_points++){
        unsigned long t = millis();
        int vcg = analogRead(analog_in_vcg);
        int vs = analogRead(analog_in_vs);
        int va = analogRead(analog_in_va);
        int voffset = analogRead(analog_in_voffset);
        int vadc = analogRead(analog_in_vadc);
        unsigned long myArray[6] = {t, vcg, vs, va, voffset,
           vadc};
        for (int array=0; array<=5; array++){
        Serial.print(myArray[array]);
        Serial.print(";");
      }
      Serial.println("/n");
      delay(600);
    }
  }
  }
  Serial.println("fim");
  delay(10);
  exit(0);
}
```