André Gomes de Sá

Bachelor in Micro and Nanotechnologies Engineering

Development of a controller for gene-expression analysis using field-effect devices

Dissertation submitted in partial fulfillment of the

Requirements for the degree of

Master of Science in

Micro and Nanotechnologies Engineering

Adviser: Dr. Bruno Veigas, Post-Doc Researcher,
Faculty of Sciences and Technology, NOVA University of Lisbon

Co-adviser: Dr. Joana Vaz Pinto, Assistant Professor,
Faculty of Sciences and Technology, NOVA University of Lisbon

Examination committee:

Chairperson: Dr. Rodrigo Martins

Rapporteurs: Dr. João Pedro Oliveira

Dr. Bruno Miguel Veigas

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

October 2019

Development of a controller for gene-expression analysis using field-effect devices

“When something is important enough, you do it even if the odds are not in your favor.”

— Elon Musk

Acknowledgments

This work is representative of the knowledge that was acquired throughout my entire academic path and thus, marks its end. This section is dedicated to all the people that were part of it, direct or indirectly.

First of all, I want to thank Professor Rodrigo Martins and Professor Elvira Fortunato for creating the course that allowed me to know and to get in touch with the nanotechnology world. I especially appreciate the fact that I had access to excellent equipment and the opportunity to meet the researchers and teachers that constitute CENIMAT that produce world-renowned content.

A special thanks to my advisers for allowing me to work with this project, and for their constant availability and unconditional support. To Professor Joana Pinto, for showing me how creativity leads to fast solutions and how a different point of view can be so helpful when there's no apparent solution. To Post-Doc researcher Bruno Veigas, for introducing me to the biotechnology world and showing me what's the dynamics of a genetic laboratory, alongside with its techniques and processes to perform genetic-expression analysis. To both, thank you for everything.

A great appreciation goes for Man Kay Law (Mathew) for showing me how to look at an electronic circuit in the application point-of-view, and how to dimension and build it, based on requirements. I could never thank you enough for all the knowledge that you were able to transmit during our meetings. Thank you for your patience and your help.

Another big thanks goes to Beatriz Oliveira, for always making sure that the solutions were properly produced to achieve the best results possible, interrupting her own work to do so. Her contribution to this project is greatly appraised and not forgotten. Muito obrigado pela disponibilidade.

I also want to express my gratitude to Ph.D. student Emanuel Carlos and Professor Rita Branquinho for being my supervisors during the research internship program. There are many things that can't be learned in class and during that time I took lessons for life. To Manu, for showing me how to be organized, and how to manage time in the laboratory. To Rita Branquinho, for showing how rigor and attention to detail are important in the lab environment, as well as the importance of doing things by ourselves. I learned a lot from working personally with you both. Thank you for your guidance.

Quero agradecer exclusivamente à minha família, que sem ela nada do que vivi e aprendi seria possível. Ao meu pai, Carlos Sá, por sempre garantir e me relembrar que o que eu faço é para o meu futuro e que o meu progresso está dependente de mim. Obrigado por me questionares e pela confiança. Deu-me garantias e motivação para continuar, e mesmo no caso de dúvida, senti que teria sempre o teu apoio. À minha mãe, Ana Gomes, quero agradecer pela paciência e esforço que puseste para garantir que tudo corria bem. Obrigado por esperares acordada às tantas da manhã só para jantarmos juntos. Obrigado por todos os pequenos, grandes gestos e por acreditares. Aos meus avós, quero agradecer por todo o apoio que nunca faltou. Ao meu irmão, José Miguel Gomes de Sá, não sei como agradecer o suficiente. Tenho que te agradecer por tudo o que fizeste para garantir o meu sucesso. Obrigado pelos conselhos, pelas longas conversas, por me mostrares alternativas, por quereres ensinar, por quereres aprender também, por estarmos lá um para o outro. Este trabalho é uma reflexão daquilo em que depositaram tanta confiança e portanto dedico esta tese a vocês.

Não podia deixar de parte aqueles que foram a minha família durante este percurso. Em primeiro lugar aos grossos: Catela, Loureiro, Chains, Opinião, Pires, Marco António, Breno, Tomás, Gonçalo, Paulo, Raúl, Torres, Vivas, Campôa e William. Obrigado por preencherem estes anos de memórias. Quero agradecer ao Sérgio por ter sido um grande companheiro durante o curso, e à Maria por todos os trabalhos que fizemos, e por todos os testes para que estudamos juntos. Foi excelente estudar com vocês e serão amigos para a vida.

A last big thanks to all the people that I've met that was not referenced above but helped in the completion of this life achievement. Thank you all.

Abstract

Field-effect-based devices have become a basic structural element in a new generation of biosensors and are today the foundation of radical new approaches for specific detection and characterization of DNA. Reliable molecular detection and characterization of genetic biomarkers are vital for disease diagnostics and therapeutic follow-up. In this work, an analog front-end circuit (AFE) was developed for an ion-sensitive FET- based real-time monitorization of isothermal DNA amplification. For this, a microcontroller/circuit integration was developed allowing for precise control of applied voltage, enabling real-time output measurements. This prototype was built over an Arduino microcontroller and was developed as a standalone platform with dedicated software for control and data handling, for a sample in result out- like sensing platform.

This prototype as shown to be able to detect changes in charge/pH comparable to previously attained results measured in a standard apparatus (22± 6 mV/pH – 58mv/pH theoretical maximum via Nernst Equation). The platform was tested with standard pH solutions (ranging from 4 to 12) and was able to convert, amplify and monitor the current variation in the order of 0.3µA/pH generated by the surface potential difference. This approach was integrated with isothermal amplification techniques (LAMP) for a relevant cancer biomarker (c-Myc proto-oncogene). Real-time LAMP amplification reactions were monitored using the standard fluorescence-based approach and compared to those attained via the developed sensing platform. Results show that the proposed platform allows for the real-time monitorization of LAMP amplification. At this stage, the error associated with the measurement increases over time, with decreasing signal to noise ratios. Also, a noticeable change in reaction kinetics was observed, leading to an increase in reaction times. Subsequently, the proposed circuit was implemented in a printed circuit board in order to reduce noise and improve overall stability.

Keywords: Transimpedance amplifier, analog-front-end circuit, LAMP, pH

Resumo

Dispositivos baseados em efeito de campo tornaram-se no elemento estrutural básico da nova geração de biossensores e hoje são a base de novas abordagens radicais para detecção específica e caracterização de DNA. Detecção molecular confiável e caracterização de biomarcadores genéticos são vitais para o diagnóstico de doenças e acompanhamento terapêutico. Neste trabalho, um circuito analógico *front-end* (AFE) foi desenvolvido para a monitorização em tempo real da amplificação isotérmica de DNA, baseado em FETs sensíveis a iões. Para isso, foi desenvolvida uma integração de microcontrolador / circuito, permitindo um controle preciso da tensão aplicada, possibilitando medições do sinal de saída em tempo real. Este protótipo foi construído sobre um microcontrolador Arduino e foi desenvolvido como uma plataforma autónoma com software dedicado para controlo e manipulação de dados, para uma amostra na plataforma de detecção de resultados.

Este protótipo demonstrou ser capaz de detectar alterações na carga / pH comparáveis aos resultados obtidos anteriormente, medidos num equipamenti padrão (22 ± 6 mV/pH - 58mv/pH máximo teórico segundo a equção de Nernst). A plataforma foi testada com soluções de pH padrão (variando de 4 a 12) e foi capaz de converter, amplificar e monitorizar a variação de corrente na ordem de 0,3μA / pH gerado pela diferença de potencial de superfície nos sensores. Essa abordagem foi integrada na técnica de amplificação isotérmica (LAMP) para um biomarcador de cancro relevante (proto-oncogene c-Myc). As reações de amplificação de LAMP em tempo real foram monitorizadas usando a abordagem baseada em fluorescência padrão e comparadas com as obtidas pela plataforma de detecção desenvolvida. Os resultados mostram que a plataforma proposta permite o monitoramento em tempo real da amplificação LAMP. Nesta fase, o erro associado à medição aumenta com o tempo, com a diminuição da razão sinal / ruído. Além disso, foi observada uma mudança notável na cinética da reação, levando a um aumento no tempo de reação. Posteriormente, o circuito proposto foi implementado em uma placa de circuito impresso para reduzir o ruído e melhorar a estabilidade geral.

Palavras-chave: Amplificador de transimpedância, circuito *front-end* analógico, LAMP, pH

Acronyms

ADC	Analog-to-Digital Converter
AFE	Analog Front-End
CAGR	Compound Annual Growth Rate
CENIMAT | i3N	Centro de investigação de materiais/ Instituto de nanoestruturas, nanomodelação e nanofabricação
CMISFET	Charge-Modulated Ion-Sensitive Field Effect Transistor
FED	Field-effect device
FET	Field-Effect Transistor
FPC	Flexible Printed Circuit
GBW	Gain Bandwidth Product
IC	Integrated Circuit
ISFET	Ion-Sensitive Field Effect Transistor
LAMP	Loop-mediated isothermal amplification
LDO	Low-Dropout voltage regulator
MISO	Master-In Slave-Out
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MOSI	Master-Out Slave-In
nMOSFET	n-channel Metal Oxide Semiconductor Field Effect Transistor
OA	Operational amplifier
PCB	Printed Ciruit Board
PCR	Polymerase Chain reaction
PEN	Polyethylene Naphtalate
PWM	Pulse Width Modulation
SDI	Serial Data In
SDO	Serial Data Out
SPI	Serial Peripheral Interface
SNR	Signal-to-Noise Ratio
TIA	Transimpedance Amplifier
USD	United states dollars

List of symbols

α	Oxide sensivity Parameter
β_{int}	Intrinsic buffer capacity
$\Delta\Psi$	Surface potential variation
μ	Population mean
σ	Population standard deviation
C_{dif}	Electrolyte diferential capacitance
C_f	Feedback capacitor
C_{in}	Input capacitance
dB	Decibel
f_p	Pole frequency
I_{ds}	Drain to source current
I_{in}	Input current
I_{max}	Maximum current
I_{min}	Minimum current
k_B	Boltzmann constant
LSB	Least significant bit
n	Bit resolution
pH_{pzc}	Point of zero charge
q	Electron charge
R_f	Feedback resistor
V_a	Input voltage on the amplifier non-inverting node
V_{adc}	Circuit output voltage that is inserted in the ADC
V_{cg}	Control-gate voltage
V_{cgs}	Control-gate to source potential difference
V_{ds}	Drain-to-source potential difference
V_g	Gate voltage
V_{HI}	The highest readable voltage of the ADC
V_{in}	Input Voltage
V_{LOW}	The lowest readable voltage of the ADC
V_{offset}	The voltage applied to the nMOSFET drain controlled by V_a
V_{out}	Output Voltage
T	Temperature

Table of contents

List of figures

List of tables

Motivation and objectives

There's a growing interest in spending less with healthcare as it's one of the greatest economic burdens in the business world. Cancer alone has affected 18.1 million people worldwide in 2018 and it's estimated that in a 5-year period, 43.8 million will be diagnosed with the pathology that caused 9.6 million deaths during 2018 [1]. Considering such statistics, this disorder has been a hot topic on medical research for the most recent decades turning to be one of the world problems to be solved.

In the European Union it was estimated that during the year of 2009, €126 billion were spent on cancer which 60% of the cost was due to productivity loss as an effect of early death [2]. As a result, companies and research centers are adopting analytic solutions in clinical diagnosis while there's an increasing prevalence of chronic and genetic diseases. Also, as the data market gets involved in the life science industry as an important asset, this market is evaluated as $19.3 billion on the present year of 2019 and expected to grow at an interesting compound annual growth rate (CAGR) of 11,5% and predicted to reach $33.2 billion by 2024 worldwide as depicted in Figure 0.1 [3].

Fig. 0.1 – Life science analytics global market by region and forecast from 2017 to 2024. Adapted from [3]

On the other hand, the growing trend of this market may be attenuated due to the high implementation costs of emerging technologies. Considering this, DNA based diagnosis based on fast, easy-to-use and cost-effective platforms is highly desirable. As the research on biosensors is growing incredibly fast with 90334 entries on science direct concerning biosensors in 2019, the signal processing electronics needs to be able to acquire the signal from the newly developed sensors.

Currently, microelectronic progress took us to the moment where a microcontroller that fits in a pocket and has more computing power than the Apollo mission computer, costs less than 5$ [4]. Nowadays, Arduino alone has sold 700,000 official boards by 2014, meaning that at least 700000 users are relying on open-source hardware and software to develop their own projects and share their developments. Building an expanding network of idea exchange brings progress as everyone can learn from each other due to its user-friendliness.

The main goal of this master thesis is to develop a simple, stand-alone, open-source platform for gene expression analysis replacing the use of complex experimental setups. The platform will be used to control and process the data signal from CMISFET biosensors produced at CENIMAT.

In order to achieve this, communication with the sensor must be performed to assure sensor data acquisition and processing in order to transmit an output signal to the final user. The following tasks were performed:

- Proof of concept: electronic circuit simulation using LtSpice;
- Breadboard prototyping;

- Circuit electrical characterization;
- Breadboard prototype testing of the sensor using different pH buffer solutions;
- Breadboard prototype testing with LAMP;
- Printed circuit board (PCB) design using EagleCad;
- PCB performance tests.

Concerning the development of architectures of control and data acquisition:

- Programming the Arduino;
- Arduino-Python communication;
- Data acquisition and treatment software development;

With the rising need for simple solutions for huge problems comes the topic that covers this master thesis.

1. Introduction

Modern new approaches for specific detection and characterization of DNA relies on field-effect devices as a basic structural element that thus, marks the new generation of biosensors. Reliable molecular detection and characterization of genetic biomarkers are vital for disease diagnostics and therapeutic follow-up. Following the development of label-free detection techniques for biomolecules, and the design of field-effect sensors for gene-expression analysis, comes the necessity to introduce the electronics for signal conditioning and data acquisition, completing the fundamental stages for a biosensing platform [5][6].

It's paramount that user-friendly, open-source, low-cost technology is developed in order to progress faster. With this aspect in mind, the work that follows is concerning the design of an analog front-end (AFE) circuit and implementation into a stand-alone, portable platform for genetic expression reactions monitoring that, despite the increasing use of electrochemical DNA detection approaches, only a few have been directed towards gene expression analysis.

1.1. DNA amplification: PCR and LAMP

Genetic expression is the process of converting genetic data present in the DNA and converting it into a genetic product, which is often a protein. Genes are constant in all cells, however, its expression varies according to the type of cell, from the time transcription occurs or during different environmental conditions. In order to analyze the gene expression according to many different variables, it is required a large amount of available DNA. In the field of life science research, nucleic acid amplification techniques are used as a tool in order to obtain enough material to perform such tests [7] [8].

Polymerase chain reaction (PCR) is a widely used method since it's development in 1983 by Kary Mullis and Michael Smith that gave the team the honor of winning the Nobel Prize in Chemistry in 1993 for such remarkable invention. It allows copying DNA sequences according to the progression $2^n - n - 1$, where n is the cycle number. After 20 cycles, over 1 million DNA copies have been produced [9][10]. However, PCR requires three very specific temperatures for each of its three-step processes in order to successfully replicate genetic data: 95ºC for denaturation, 60ºC for annealing and 72ºC for elongation phase[11].

The necessity of having three distinct temperatures for DNA amplification certainly limits the process. In order to solve this issue, the scientific research community developed several isothermal nucleic acid amplification techniques. The most commonly used due to its properties is the loop-mediated isothermal amplification (LAMP) [8]. In this method, the reaction reacts under isothermal conditions of 60ºC to 65ºC with high specificity, efficiency, and rapidity[12]. Besides being an isothermal method, DNA is amplified 10^9 to 10^{10} times in 15 to 60 min, meaning that it can save 1 hour when compared to PCR reaction. Furthermore, if the target gene is absent, it forms a white magnesium pyrophosphate precipitate visible to the naked-eye allowing easy inspection [7]. However, in order to detect the amplification, a nucleic acid dye is used to follow the replication procedure through fluorescence analysis.

In 2017, thin-film ion-sensitive sensors, that rely on H^+ ions as the LAMP reaction product to detect DNA amplification in real-time were designed and produced as an innovative detection method that uses a FET as a transducer component[6][13].

1.2. DNA sensors

Considering all of the types of sensors (such as optical, electrochemical, piezoelectric, potentiometric) that are used in recognition of non-visible at naked eye species like ions or biomolecules, using field-effect device (FED) biosensors present two main advantages: label-free detection and the

possibility of easy integration due to its small size, even though FET integration it's out of this thesis scope [14][15].

Label-free detection is obtained by combining FEDs as the transducer and a functional sensitive surface. It can be sensitive to ions, molecules or even DNA strands. When these two are combined, they exhibit tunable electrical conduction which is used in favor of molecule detection.

In this category, the ion-sensitive field-effect transistor (ISFET) is one of the most popular electrical biosensors, having been adopted by the various lab-on-chip and health care market applications[16]. This particular implementation, uses the accumulation of reaction by-products (e.g. H+; OH-), in the oxide surface (e.g.v SiO2, Si_3N_4, Al_2O_3, Ta_2O_5) resulting in depletion or accumulation of carriers caused by the change of electric charges on the gate terminal [17].

So far the implementation of ISFETs as been made using external reference electrodes that are maintained at a constant voltage, inside the electrolytic solution. As the electrons flow to the electrode, it attracts positively charged ions and thus, negatively charged ions are attracted to the oxide surface, leading to current control that flows in the channel [18]. Several architectures have been pursued concerning reference electrodes such as differential electrodes, pseudo-reference electrodes, and solid-state back electrodes known as charge-modulated architecture. It reduces the cost and complexity of the sensing mechanism and based on this fact, charge-modulated FET sensors were developed.

1.2.1. Charge-modulated FET sensors

Based on the concept introduced by Barbaro et all in 2006, Veigas and his team developed electrodes with an extended gate contact coupled with a sensitive layer, allowing for much simpler device fabrication. The post-processing steps of the sensitive area are kept away from the transducer and separate the wet from the dry area where the transistor is activated. Since the control gate is not in contact with the solution, a reference counter-electrode is not needed [19][20][21].

Charge-modulated ion-sensitive field-effect transistor (CMISFET) sensing electrodes were produced on flexible and transparent polyethylene naphtalate (PEN) substrates in a stacked configuration. Electrodes were fabricated using the following thin film deposition sequence: molybdenum back contact; parylene dielectric film; molybdenum top contact that is used as the extended gate contact of a commercial MOSFET (CD4007UB) and Ta_2O_5 sensitive layer as shown in figure 1.1 [19].

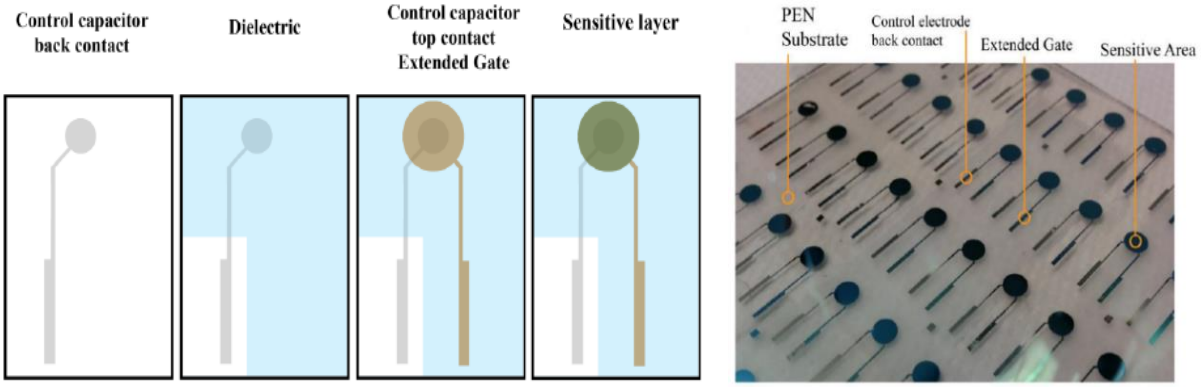

Fig. 1.1 – Sensor structural layers and design. Adapted from [6]

The sensitivity of the sensor depends mostly on the intrinsic properties of the tantalum oxide and of the ionic concentration that is related to the Debye length. This parameter correlates the distance of the ions to the surface with the surface potential (ψ). The closer the ions are to the sensitive layer, the greater the surface potential, reaching its maximum when ionic bonds are formed. In order to characterize an oxide's sensitivity, there's the sensitivity parameter, α. For a perfect sensitive oxide, α=1. As so, theoretical maximum sensitivity can be calculated according to the following equations 1, 2 and 3.

$$\alpha = \left(\frac{2.3\ k_B T C_{dif}}{q^2 \beta_{int}} + 1\right)^{-1} \quad (1) \qquad \frac{\delta \Psi_0}{\delta pH_B} = \frac{-2.3 k_B T}{q} \alpha \quad (2) \qquad \Delta\Psi = \frac{2.3 k_B T}{q} \Delta pH \quad (3)$$

Where C_{dif} is the differential capacitance that translates as the ability of the electrolyte to adjust the amount of charge as a result of a small change in the electrostatic potential. β_{int} is the oxide's intrinsic buffer capacity, which is related to the change in the total number of surface charge groups as a function of proton concentration variation. This expression relates the surface potential variation due to a pH variation, and for an ideal sensitive oxide, a maximum sensitivity of 59.2 mV/pH can be achieved. This is known as the Nernstian sensitivity since it is derived from the Nernst Equation [5].

The surface potential variation of an ISFED (CMISFET, ISFET) is usually represented by an extra contribution to the transistor gate potential that will lead to a shift in the transfer curves of the device. For such, its measurement relies on the ability to detect low changes as low as few mV (in V_T) or nA in I_{DS}.

1.3. Analog front-end circuit

Preceding signal digitation, the transducer outputs are amplified and conditioned by an analog front-end circuit (AFE) in order to improve central aspects of the data such as noise, linearity, input impedance, bandwidth and power consumption[22].

As an example, active ground shielding is critical for improving signal gain while using high impedance transducers such as in capacitive sensors like in the current case. Operational amplifiers (OA) have high input impedance and therefore, they can be used as feedback amplifiers to reduce voltage difference across parasitic capacitances. Such a feature is extremely important to improve the linearity of inherently non-linear transducers and its signal gain[22].

The transducer mechanism used in this work induces an I_{DS} shift in the FED. However, since analog to digital converters (ADC) rely on voltage readings from 0V to 5V, the design for the AFE circuit must have a transimpedance amplifier (TIA) that converts and amplifies current to voltage[23].

1.4. Transimpedance amplifiers

The main block of a TIA is an operational amplifier. Since 1967 this integrated circuit (IC) has been produced and now presents the basic block of most current technology. Such devices rely on the feedback concept and are composed of the basic amplifier and the feedback network that have voltage or current as inputs and voltage or current as output resulting in four types of feedback configurations.

Using negative feedback brings four major benefits for designers. The most important one is the ability to stabilize the gain of the amplifier when some changes occur in the performance of active devices such as temperature change or supply voltage fluctuation. Negative feedback also allows to design and modify the input and output impedances of the circuit as desired to fit the requirements. When dealing with waveforms, using negative feedback reduces the signal distortion and also the possibility to broaden the bandwidth of the circuits. There are also disadvantages associated with this configuration. Usually, the overall gain of the circuit gets reduced proportionally to the benefits that it brings. In many situations it's required to add gain stages to compensate for this loss, increasing hardware cost and complexity.[24]

In this case, in particular, the input variable is current and voltage as output is desired so that the signal can be read by the ADC. More details about this device are presented later.

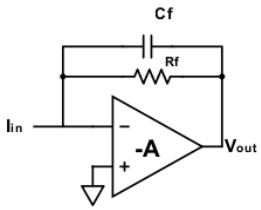

Fig. 1.2 – Transimpedance amplifier

For a better understanding of the circuit, an ideal OA with differential input, infinite voltage gain, infinite input resistance, and zero output resistance is assumed to characterize its working principle. As the output is connected to the inverting node, it is an inverting configuration. So the gain can be dimensioned according to the circuit necessity as follows [25]:

$$\frac{V_{out}}{-I_{in}} = R_f \leftrightarrow R_f = -\frac{V_{Max}-V_{Min}}{I_{Max}-I_{Min}} \quad (4)$$

Where R_f and C_f are the feedback resistor and capacitor, V_{Max} and V_{Min} is the desired maximum and minimum voltage output, and Imax and Imin, the maximum and minimum current flowing into the inverting node.

Commercial OAs are not ideal and there are key factors that directly influence the overall performance of the circuit. An OA can be chosen by analyzing crucial parameters provided by the IC manufacturer, such as:

- **Gain**, the ratio between the output and input power or amplitude;
- **Bandwidth** determines the range of frequencies for which the proper functioning of the amplifier;
- **Noise**, as an unwanted electrical disturbance in the amplification process[23].

1.5. Stability

Generally, TIA circuits use C_f in parallel to ensure stability by compensating for parasitic capacitances at the inverting node of the amplifier. One approach to guarantee circuit stability is to determine the largest value for C_f that could be implemented and then, choose an OA with a sufficient gain bandwidth product (GBW) that fits the requirements. This parameter is very important because it shows to which frequency the gain falls to unity. So, with an OA with e.g 5MHz GBW, meaning that the circuit will remain stable up until 5 MHz as long as the gain remains at 1[26].

C_f together with R_f forms a pole in the frequency response of the amplifier, meaning that above this frequency (f_p) the closed-loop gain of the circuit will decrease. The maximum capacity value for C_f can now be calculated using the value of the feedback resistor and the desired bandwidth [25][26]:

$$f_p = \frac{1}{2\pi C_f R_f} \quad (5) \qquad C_f \leq \frac{1}{2\pi R_f f_p} \quad (6)$$

To ensure that the circuit performs well for the desired bandwidth, the designer must choose a capacitor with a capacity value below C_f.

GBW can be estimated according to the following equation, where C_{in} is the OA input capacitance [25][26]:

$$GBW > \frac{C_{in}+C_f}{2\pi R_f C_f^2} \quad (7)$$

The objective is to choose an OA that would fit the circuit requirements and so, at first instance, the value of C_{in} is unknown. Most OA has input capacitance around 10pF, and so this value is used as a reference value, allowing to estimate the GBW parameter and then choose the right OA accordingly.

Besides stability, dealing with noise is of extreme importance. The next topic is dedicated to what kinds of noise are expected in a circuit and how to deal with this issue.

1.6. Noise

Noise is a critical aspect since it compromises the reliability of the processed signal. To evaluate the signal accuracy, the ratio between the signal power and the noise power is calculated. This is known as the signal-to-noise ratio (SNR) and is often expressed in dB. Alternatively, the ratio between the mean value and its standard deviation can also be used[27].

As an indicator of the circuit performance, the higher the SNR the better although a ratio above 60 dB is considered a good value[27].

In a circuit, noise sources are common and generally, every component can be considered as a source of noise as well as the surroundings that also induce instability. Some examples of the most common types of noise are thermal noise, electromagnetic interference and impedance coupling[28].

Thermal noise is inevitable since it's generated by the thermal agitation of electrons inside a conductor. Their increasing random behavior with temperature and frequency manifests as noise-induced in the circuit. Also, electromagnetic interference from the environment may induce disturbances in the circuit performance. Circuit shielding is usually done by encapsulating the circuit in a metal casing to prevent interference from outside sources and obtain a more stable and reliable signal[29]. Impedance coupling comes as a ground connection related interference when multiple ICs share a common ground. Ideally, ground connections have impedance zero. Practically, this reference point has some resistance or capacitance and therefore, creates unwanted currents that produce signal instability[28].

Total noise in a circuit is a result of a root sum squared of all of the noise sources and not its algebraic sum. Meaning that most of the readable noise is due to the noisiest source and not to the algebraic sum of all the disturbance caused by all components[29].

After the analog signal is processed, it needs to be transformed into a series of bits so that the microcontroller can understand and display the output.

1.7. Analog to Digital Converters and Microcontroller

An analog-to-digital converter (ADC) is an IC that converts a magnitude of voltage to a binary number. One key aspect of working with ADC is its resolution. It determines the number of discrete values over the range of analog values, determining the minimum readable voltage by the ADC [30].

An ADC with 10 bits resolution (n) allows to represent 1024 discrete values within the analog range (V_{HI} - V_{LOW}), and this value can be used to express resolution in volts per step (LSB) as:

$$LSB = \frac{V_{HI} - V_{LOW}}{2^n} \quad (8)$$

V_{HI} and V_{LOW} are the maximum and minimum values of the analog range that it's being converted to a digital signal which is often a 5V range.

When designing an analog circuit it’s very important to determine what’s the needed resolution so that the proper ADC can be chosen. This way assuring that information is not lost and get accurate readings. As so the resolution needed for the ADC can be estimated as:

$$\#Resolution\ bits = \log_2\left(\frac{ADC\ dynamic\ range}{Sensor\ resolution}\right) \quad (9) \qquad \#Error\ resolution\ bits = \log_2\left(\frac{\sigma}{Sensor\ resolution}\right) \quad (10)$$

Considering these aspects, concerning the analog circuitry and the signal digitation parameters, all information is gathered to develop a stand-alone portable platform for acquiring data from ion-sensitive sensors.

2. Materials and methods

The chapter that follows presents all the aspects involved in the design of the proposed platform as well as its electrical characterization and testing. It includes the circuit simulation, followed by the platform prototyping aspects including all the components required and the design of the printed circuit board (PCB). Next, a section dedicated to all of the microcontroller and interface programming that include all the libraries used for data acquisition. Then it's presented the procedure for the circuit electrical characterization. Finally, the platform testing using buffer solutions and LAMP reactions procedure is described.

2.1. Simulation

In order to evaluate the needs for the final circuit regarding the bit resolution and the stability, LTspice was used to simulate the CD4007UB nMOSFET response and the TIA configuration response. The same biasing parameters as the work done by Veigas et al were used. A voltage source was used to simulate a constant V_{cg} of 1.7 V on the back contact and a V_{ds} of 0.05 V. An additional voltage source was used (V_{dna}) to simulate the voltage generated by the surface potential change and an ADA4000 OA model was used for the transimpedance amplifier.

2.2. Platform prototyping

2.2.1. Schematic and components

After the simulation results, the components were chosen according to the circuit needs. The used components are presented in table 3.1.

Table 2.1 – Components list and its unity price[1]

	Type/Value	Unity Price (USD)
Power supply	3 batteries	0.37
	Battery holder	1.61
Resistors*	680 Ω, 2 KΩ, 5.6 KΩ, 10 KΩ, 33 KΩ, 100 KΩ	0.1
Capacitors*	22 pF, 0.01µF, 0.1µF, 10µF	0.15
Trimmer Potentiometer	1 KΩ	1.5
	10 kΩ	1.5
Digital potentiometer	X9C103P [31]	5.43
	MCP4141 [32]	0.83
MOSFET	CD4007UB [33]	0.51
Operacional Amplifiers	LM324N [34]	0.52
	AD8651 [35]	3.36
Current source	LM334Z [36]	0.84
PCB	JLPCB [37]	3.6
Microcontroller	Arduino Atmega 2560 [38]	35**
Miscellaneous	DC Jack	0.5
	Header pins	0.11
	SMT breakout	2.57
	IC socket	0.18

[1] All prices were retrieved from DigiKey store, all the exceptions were marked by * Estimated price; ** Price from the official store

2.2.2. PCB design

To design the PCB, Eagle software was used and the models of the components were imported into the Eagle software using SamacSys library loader available for free to download [34]. All the necessary routing was performed and the PCB layout was generated using the software functionalities. The dimensions of the board were set to 60.95 mm x 79.68 mm so that would fit on top of the Arduino microcontroller board. Ground planes were created by assigning polygon function to a ground signal. Guard rings were also added in the inputs of the OA to minimize noise generated by the surrounding components. The width of the copper route lines was adjusted for 10 mils for digital signal and 12 mils for analog signal and finally, the CAM files were generated and sent to a PCB manufacturer.

2.2.3. Software development

The microcontroller programming was developed in two different languages for different purposes. The Arduino board was initially programmed using the Arduino IDE and third-party libraries to use the X9C103 digital potentiometer, both available online [39][40]. 5 analog signals (V_g, V_s, V_a, V_{offset}, and V_{adc}) were defined and attributed to the 5 analog-read pins in the Arduino board. 3 PWM pins were used to communicate with the digital potentiometer assigned to UD, INC, and CS digital potentiometer pins. The connection of the inputs is arbitrary as long as the analog signal is connected to an analog Arduino input and digital inputs are connected to PWM pins assigned in the microcontroller.

Python scripts were also developed to automatically generate an Arduino file containing information on the number of steps the user wants to perform, the number of data points per step and the number of cycles to perform. After the generated file is send to the microcontroller, another Python script communicates with the Arduino serial and acquires the data and generates a .csv file. Finally, the acquired data can be treated and plotted for fast inspection. The following libraries were used: Numpy, Pandas, Scikit-Learn, Matplotlib, Tkinter, math, and Shutil all available for free [41]–[48].

2.3. Platform performance

2.3.1. Electrical characterization

The node voltages were checked using a multimeter and oscilloscope. The procedure for electrical characterization of the circuit is as follows:

1 - Placing all the components and assure its connections as shown in Annex A. Do not connect the sensor and the digital potentiometer. This is done later;

2 – Using a bench power supply, limit the output current to 10 mA and set the supply to 2.6V to set the Vg voltage by wiring it to the LM324Z negative terminal;

3 - Supplying the circuit with Arduino 5V supply or 4.5V using batteries. If the Arduino supply is going to be used, make sure to share the ground with the microcontroller in order to close the loop;

4 - Using a multimeter, verify the voltage at the input and the output nodes of the LM324Z for V_g, V_s, and V_a. The voltage at the input nodes must match the voltage at the output nodes of the LM324Z, assuring that the unity-gain buffer is properly functioning. If they do not match, verify all the connections. Starting by assuring that V_g, V_s, and V_a are connected to the positive input terminal of the LM324Z, then check supply and ground connections;

5 – Compare the values of V_a and V_{offset}. These must match, guaranteeing the proper functioning of the AD8651 OA. If they do not match, the OA is either saturated, meaning that it can't go above the supply limit or there's no signal to amplify, meaning that the nMOS is *off*. Applying V_g = 2.5V is enough to turn on the transistor so, double-check all the connections if the problem continues;

6 – Monitor V_{adc} using an oscilloscope. It should show a constant signal at around 2.5 V;

7 – Gently change V_g in the power supply and follow the V_{adc} output shift as a function of V_g;

Next, the MCP4141 and X9C103P digital potentiometers were connected to the circuit, substituting the bench power supply and making the transition to the stand-alone platform. In order to test its performance, the code necessary to communicate with the ICs was uploaded (Annex B) and the same procedure as described above was performed. Instead of using the multimeter, the Arduino serial is used to monitor the node voltages.

2.3.2. Prototype testing with the CMISFET sensor

The response of a sensor was tested using pH buffer solutions with pH values of 4, 5, 7, 9, 10 and 12 provided by Hannah Instruments. The optimized protocol consists of: Setting the setpoint to approximately half of the output dynamic range by setting Vcg at 2.6 V. Filling a well on the sensitive layer with 20 µL of a 4 pH buffer solution and was let to stabilize for 30 to 45 minutes before measurement. Monitor the output response as it is expected to rise and stabilize. After the measurement, 18 µL of the previous solution is removed and 18 µL of a new pH solution is introduced and let to stabilize for at least 10 minutes. Tests were performed starting with pH 4 until 12 and going back 4 pH, as well as with random pH buffer solutions to evaluate the response of a random stimulus. The tests were done using the breadboard prototype applying constant V_{cg} and also V_{cg} sweeping type of measurements.

2.4. LAMP

To perform LAMP reactions, a list of reagents were used and presented in table 3.2.

Table 2.2 – LAMP reaction reagents, its concentration, and used volume

Reagents	Volume (µL)
Evagreen 208	0.5
PCR H2O	1
Bst Polymerase Buffer	0.8
FIP primer 10 µM	1.6
BIP primer 10 µM	1.6
F3 primer 10 µM	0.2
B3 primer 10µM	0.2
dNTPs 5mM	0.9
Betaine 5M	1.6
$MgCl_2$ 25mM	1.2
Bst Polymerase	0.4
c-Myc DNA Template	1

The biological material specifications used are described in table 3.3.

Table 2.3 – Used primers in LAMP reaction, its sequence, melting temperature, and gene bank code

Name	Sequence (5’ – 3’)	Tm (°C)	Genebank Acc.No
MYCFIP	CTTTTCCTTACGCACAAGAGTTCCG GAAACGACGAGAACAGT	48*	NM_002467
MYCBIP	ACGATTCCTTCTAACAGAAATGTCC CAAGGTTGTGAGGTTGCA	48*	NM_002467
MYCF3	TCTGAAGAGGACTTGTTGC	48.9	NM_002467
MYCB3	TTCAGTCTCAAGACTCAGC	48.9	NM_002467

Each LAMP solution was created for performing relative fluorescence measurements and for monitoring using the proposed platform. The procedure for solution preparation to perform fluorescence measurements is as follows:

1 – All reagents aliquots were allowed to defrost;

2 – A reaction mix was prepared according to Table 3.2;

3 – Reaction mix was briefly mixed and vortexed;

4 – The reaction mix was distributed to 0.2 ml microtube;

5 – 10 ng of template previously produced was added to 0.2 mL microtube;

6 – Reaction was briefly mixed and vortexed and incubated in the Rotor-Gene RG3000 thermocycler at 65 °C for one hour for the amplification reaction.

For FET-based measurements, the biomarker EvaGreen 208 is not needed. The acrylic well was collocated inside a heater at 65ºC for at least 30 minutes to ensure its overall temperature. Next, the setpoint was set to 2 V and 20µL of buffer solution of $MgCl_2$ was inserted in the well and covered to avoid evaporation. Then, applying constant V_{cg} during one hour to activate and check the sensor stability was done. Next, the volume was removed and 20 µL of amplification solution was deposited in the well, covered to avoid evaporation and V_{cg} sweep measurements were performed according to the developed software (Annex C).

3. Results and discussion

This section is dedicated to the AFE circuit design, prototyping, and testing. The proposed circuit was simulated and implemented in a physical prototype. The platform was used to acquire data from the CMISFET and finally, it was used to monitor LAMP reactions. All the results are shown and discussed further.

3.1. Circuit simulation

The main objective is to replicate the reference results using component models, mostly to determine what is the expected input voltage generated by the sensor. In figure 3.1 it is presented the CMISFET working principle.

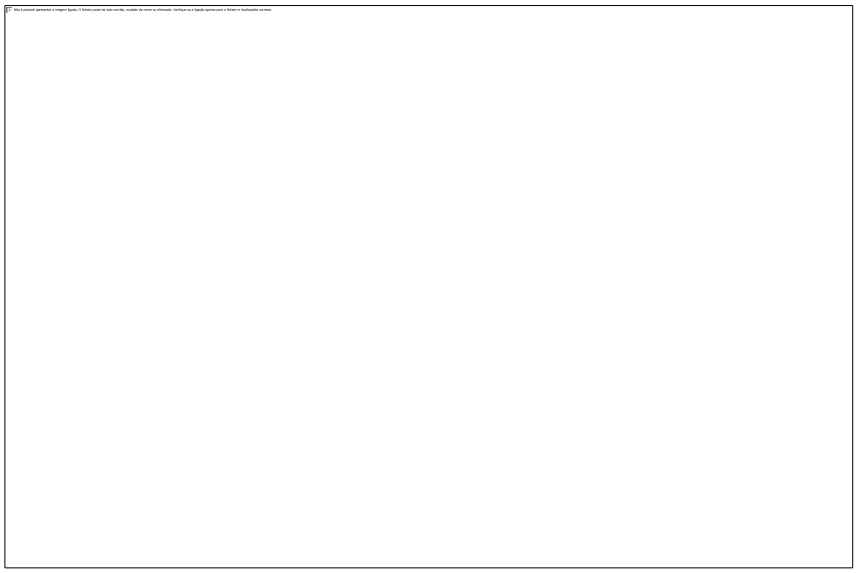

Fig. 3.1 - Charge modulated ISFET biosensor. Sensing electrode comprising a stacked control/sensing electrode and a standard commercial transistor. As the H+ ions get attracted to the oxide sensitive layer by applying a control-gate voltage (V_{cg}), a surface potential shift in induced. The potential difference is transmitted to the gate of the nMOSFET trough an extended gate connection that will drive the drain current to increase or decrease based on charge density on the sensitive surface. Adapted from [19].

To do so, the same MOSFET device was used (CD4007UB). As the CD4007UB library device is not listed in LtSpice, a third-party library was added and tested to check compatibility with the obtained transfer characteristics measured by the Keithley Semiconductor analyzer depicted in figure 3.2[49].

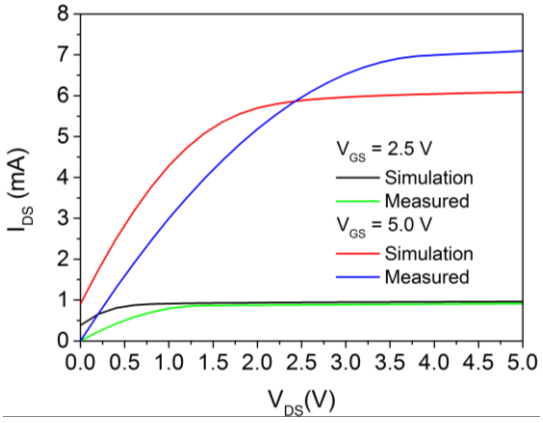

Fig. 3.2 - CD4007UB nMOSFET transfer curve comparison between experimentally measured data and its model

The plotted data shows that I_{DS} current in the linear region for the model is greater than the measured data, showing a difference of 1 mA between the two curves at the origin when applying a gate voltage of 5V. For V_{GS} = 2.5V, in the linear region, a difference of around 0.5 mA between both curves is shown. The transducer tested by Veigas used a constant source to drain voltage V_{DS} = 0.05 V and it's important to take into account this difference. However, the used library fairly reflects its real electrical behavior and it was chosen to perform the following simulations using the model.

The data reported shows the drain current as a function of pH. Based on these values, it's possible to predict how much voltage difference is associated with a surface potential change in the sensor, using the same test conditions (V_{ds} = 0.05 V and constant V_{cg} = 1.7 V) [19].

The reference data presented in figure 3.3, shows that for pH 9 the I_{ds} current is set to approximately 47.6 µA, for pH 7, 48.5 µA and for pH 5, 49.6 µA. From pH 9 to 7, there's a current swing of 0.9 µA and from pH 7 to 5, approximately 1.1 µA. These results show the non-linearity of the response, however, as an estimate, the current varies approximately 0.5 µA/ΔpH, showing how much current the final platform needs to be able to convert and amplify.

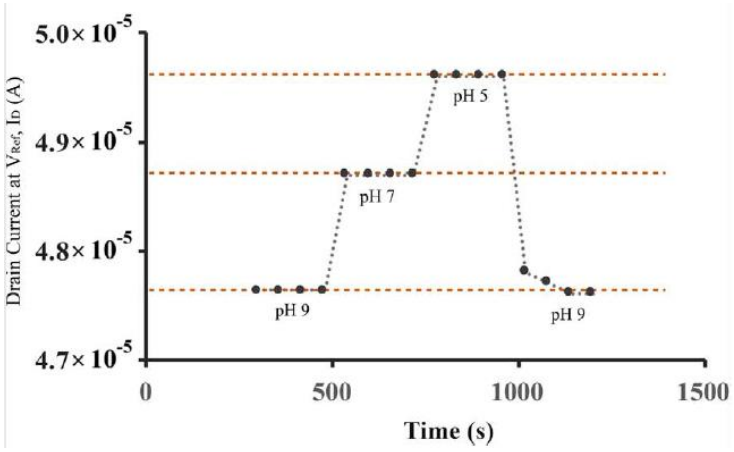

Fig. 3.3 – nMOS drain current shift overtime for pH 9, 7 and 5 adapted from [19].

To predict the surface potential difference due to the anchoring of the ions of the pH buffer solutions on the oxide, an additional voltage source (V_{dna}) was added in series with a control gate voltage (V_{cg}) source in the model. By changing the value of V_{dna}, the I_{DS} current in nMOSFET also changes and therefore can be used to match the experimentally observed current.

V_{dna} was swept from 0 to 1 V with a step size of 1 mV at a constant V_{cg} of 1.7V. The simulation retrieves the same experimentally observed I_{DS} current at pH 9, pH 7 and pH 5, when V_{dna} is respectively 0.522 V, 0.543 V, and 0.568 V. These results reflect a surface potential difference of $\Delta\Psi \approx 11.5$mV/pH consistent with the sensitivity reported for this sensor, of 12 ± 2 mV/pH [6]. V_{dna} simulation schematic and results are present in Annex C.

The circuit that will control the CMISFET sensor was simulated to allow dimensioning the TIA as presented in figure 3.4.

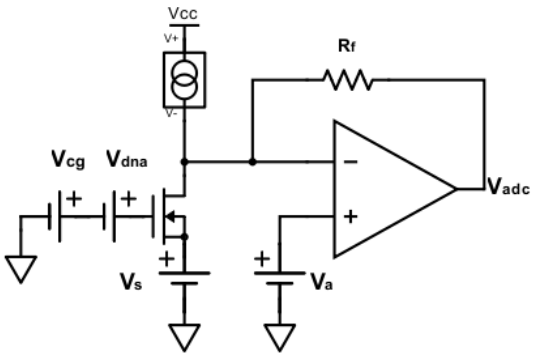

Fig. 3.4 – TIA schematic used for simulation with R_f as feedback resistor, V_a as supply in the non-inverting terminal of the OA, V_s as source potential and the sensor model with a V_{cg} and V_{dna} voltage sources in series.

The circuit was designed considering that with a neutral acidity solution on the sensor (pH 7), the output voltage of the circuit would be equal to 2.5V resulting in output in the middle of the ADC dynamic range.

The current source was dimensioned so that when a more acid solution was used the current would increase and therefore, the output voltage would increase beyond 2.5V. For basic solutions would decrease, going below 2.5V. This feature can be achieved by summing or subtracting a constant current provided by I_s, connected to the inverting node of the OA.

For this project purpose, the LAMP reaction results in an increasing amount of H+ ions, thus increasing output as the reaction occurs. This way, the addition of the current source acts as a versatile component for other future applications or for extra control over the system.

In the circuit presented in figure 3.2, using 0.543V as Vdna doesn't actually result in 48.8 uA due to the addition of a V_s supply to guarantee V_{ds}= 0.05 V. So, the current source supply was tuned to fit the characteristics resulting in a V_{dna} = 0.515 V for a pH = 7.

One of the conditions of the reference results was that V_{ds}=0.05 V. To ensure a 50 mV voltage supply, V_s was set to 2.5V and V_a = 2.55 V. Considering the ideal OA, V_d = V_a and this way assuring V_{ds}=0.05 V and defining the setpoint at approximately 2.5V in the output Vadc.

From the CD4007UB nMOS transfer characteristic, for V_{ds}=0.05 V the maximum output current I_{ds} is 50 µA. This value is crucial to determine the value of the feedback resistor R_f needed to design the TIA.

Taking into account the maximum and minimum current that would flow through R_f, and the dynamic range of the ADC, the transimpedance amplifier was designed according to equation 2 as:

$$Rf = \frac{Vmax - Vmin}{Imax - I\,min} = 100\ \text{k}\Omega$$

Where I_{max}=50 µA, I_{min} = 0 A, V_{max}=5 V and V_{min} = 0 V.

It was shown that the surface potential difference in the sensor already forces the current in the nMOSFET to go higher or lower. For other applications, with the knowledge of $I_{máx}$, I_s can be defined to induce a positive or negative change in the output voltage. By setting I_s to half of the current range that the nMOS provides (that translates to $I_s = \frac{I_{max}}{2}$) and the setpoint to half of the dynamic range of the ADC, current flow in different directions when above or below I_s. As a consequence, V_{adc} decreases or increases depending on the direction that current flows. The following simulations were taken considering this feature.

The values for the components in the simulated TIA are presented in table 3.1.

Table 3.1 – Simulation parameters values

Vcc (V)	Vg (V)	Vs (V)	Va (V)	Is (µA)	Rf (Ω)
5	4.2	2.5	2.55	18	100k

I_s was tuned down to 18 µA so that, around V_{dna} = 0.515 V, the current that flows through the resistor R_f would approximate 0 as possible to meet the previous criteria as shown in figure 3.5 a. The simulation results show that for a neutral acidity there is approximately no current flowing in the resistor R_f and sets V_{adc} at 2.54 V. It is also important to note that we are assuming that a difference in pH would change the sensor surface potential linearly.

The simulated circuit shows the capability of detecting the required signal, however, it doesn't cover the entire dynamic range as can be seen as V_{adc} ranges only between 2.2V and 2.9V.

To assure the coverage of the entire dynamic range, R_f can be increased to 900 kΩ as shown in figure 3.5 b. Also, choosing an OA with a higher open-loop gain is an option to increase the output shift.

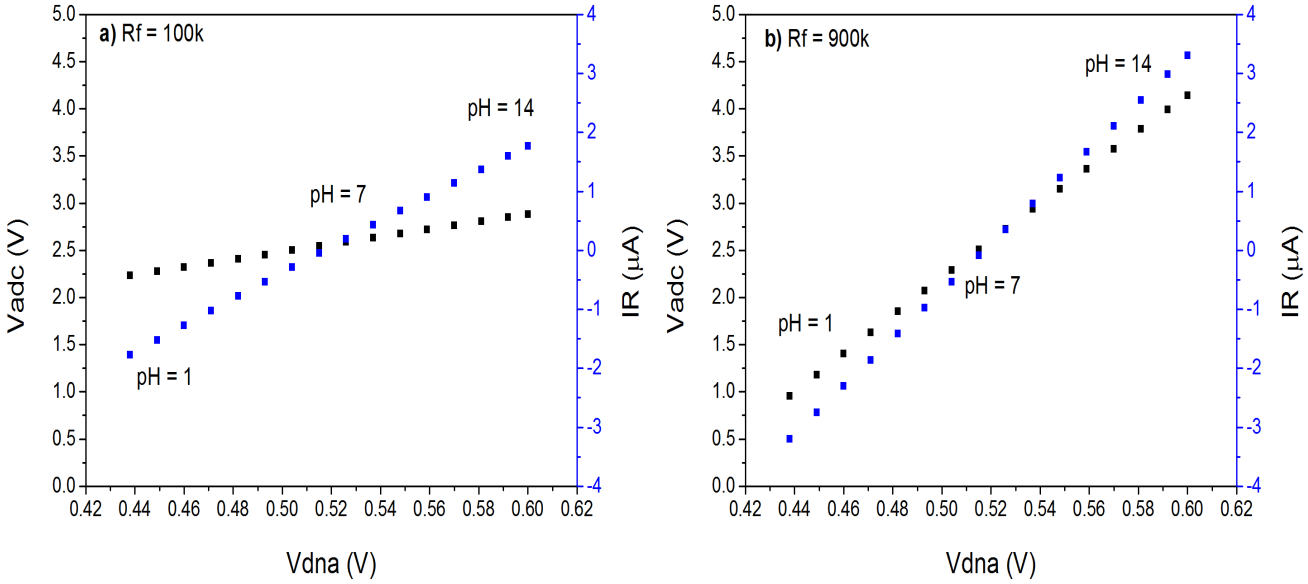

Fig. 3.5 - a) Current flowing in the feedback resistor and Vadc as a function of Vdna using R_f=100kΩ and b) 900kΩ

The digital resolution or bit requirement analysis can be estimated taking into account the linearity of the V_{adc} for both values of R_f.

The number of bits required to convert the analog V_{adc} output to a digital signal is given by equation 9 and it is related to the resolution of the analog signal V_{adc}, expressed as μ_a, and obtained through the linear regression of V_{adc} versus V_{dna}. The number of error bits can also be extracted according to equation 10 where σ_a is the standard deviation of the slope, representing the expected resolution error.

In table 3.2 the results form the linear fit of both circuit outputs and the ADC bit requirements are presented.

Table 3.2 – Output linear fit variables and ADC bit requirements

R_f (kΩ)	μ_a (mV/pH)	σ_a(mV/pH)	SNR	Signal bit requirements	Error bits requirement
100	4.01	0.0027	1485	11	11
900	19.72	0.0678	291	8	9

By increasing the value of the feedback resistor, the overall sensitivity of the circuit improves from 4 mV/pH to 19.7 mV/pH. Nevertheless, the error associated with it also increases, reducing the signal to noise ratio 5 times. Despite the SNR drop, an output with SNR above 60 is considered a good signal and thus, allows the use of a 10-bit ADC instead of a 16 bit, which is more expensive and more susceptible to errors. Also, the Arduino board has a built-in 10-bit ADC, fitting perfectly the needs of the circuit.

As a final remark, the platform will work on DC signals where the bandwidth is not the main issue. It was chosen to make these calculations as a note for other applications. The pole frequency (f_p) was chosen arbitrarily as 5KHz. Using equation 4 with R_f=900K:

$$C_f \leq 35\ pF$$

Using equation 5 and assuming 10 pF as C_{in} and C_f as 30 pF:

$$GBW > 7.8kHz$$

With these values, proper OA can be chosen as the GBW should be bigger than 7.8 kHz.

The simulations showed promises that a TIA can be used to acquire and convert the I_{ds} current from the FET transducer. Next, a physical prototype was dimensioned and implemented.

3.2. Circuit development and implementation

The main idea behind the design of this platform is to be able to make the same measurements done by Veigas[19]. As so, the platform has to be able to perform V_{cg} sweeping and to extract an output voltage as the function of time. To achieve such a feature, the analog front end (AFE) circuit was designed based on the schematic represented in figure 3.6.

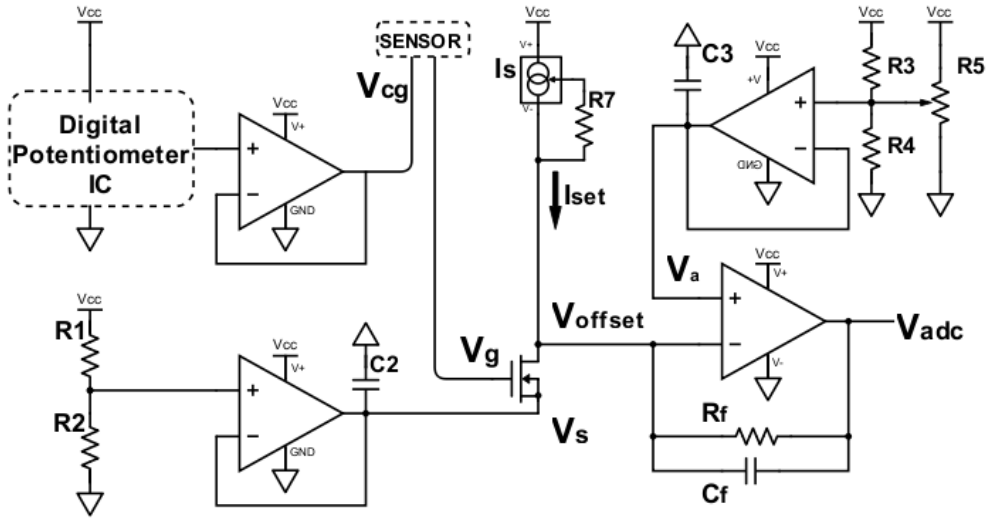

Fig. 3.6 – Proposed AFE circuit where R_1, R_2, R_3, R_4, and R_5 are in a voltage divider configuration. Each voltage divider and the digital potentiometer outputs are inserted in the non-inverting nodes of the unity-gain buffer OAs to boost its output current, setting the values of V_{cg}, V_s, and V_a. R_5 is a trimmer potentiometer that allows to manually tune V_a and therefore, V_{offset}. V_{cg} is connected to the control gate of the sensor. The surface potential difference in the sensor defines V_g that is connected to the nMOSFET gate. I_s is an adjustable current source where R_7 determines the I_{set} current. C_f and R_f are the feedback capacitor and the feedback resistor of the TIA block. Vadc is the final circuit output.

In order to keep the nMOSFET in the linear region, V_s and V_{offset} need to be designed in a way to assure the proper working conditions for the nMOSFET. To do so, V_s and V_a were designed using simple voltage dividers and buffer operational amplifiers (OAs). Since OA have high-impedance, they draw very

little current from the voltage dividers and give the same voltage as output. They function as isolation buffers, isolating the voltage dividers from the adjacent circuitry, minimizing disturbance in the signal.

. The required V_{ds} is 0.05 V and therefore, $V_{offset} – V_s$ should be equal to 0.05V. For an ideal OA, V+ - V- = 0, meaning that setting V_a to the desired voltage, V_{offset} is defined to the same value. Following this idea, by keeping constant V_s and controlling V_a, any value for V_{ds} can be set. For this particular work, the main interest is operating in the linear region but measurements can be performed in the saturation region. To fulfill this feature, a trimmer potentiometer (R_5) was added, giving the possibility to tune V_{offset} and therefore, V_{ds}.

Although the value of V_s is arbitrary, two main aspects were taken into account: the ADC dynamic range and V_{gs}. One way to think about the problem is that V_{offset} could be set to 0.05V and V_s could be grounded. However, this voltage is really close to the lower limit of the dynamic range and so, more susceptible to errors and more difficult to control. As a result, its chosen to set a reasonable value to V_s instead of grounding the nMOS source. Also, V_s shouldn't be too high to easily activate the transistor. As so, a value of V_s ranging from 0.2 V to 1 V was considered.

R_1 to R_5 were dimensioned using the voltage divider so that V_s is approximately equal to 0.8V and V_a approximately from 0.54 to 5V (considering V_{cc} = 5V).

$$V_s = \frac{R_2}{R_1+R_2} Vcc \quad (9)$$

Later analysis revealed that the voltage divider that sets the value of V_a was badly designed since R_3 and R_4 are not necessary when using a potentiometer that allows a voltage variation from 0V to 5V. In future applications, R_3 and R_4 should be removed from the circuit and add a resistor (10Ω) connected between the R_5 negative terminal and ground.

Control over V_{cg} is highly required for this circuit to be able to do different types of measurements: by applying constant V_{cg}, and V_{cg} sweeping type of measurements. Using a digital potentiometer allows control over V_{cg}. These ICs are composed of a series of resistances that can be selected through digital communication and just like an analog trimmer potentiometer, they have a high supply input, a low supply input, and a wiper.

To choose which device fits better the requirements, some important characteristics should be considered: how many resistors the IC has and its values (how many steps can be done), the bit requirements to communicate with the IC, and the supported supply voltage range. Taking these aspects into account, two different digital potentiometers (MCP4141 and X9C103P) were chosen to use in this circuit and its specifications are presented in table 3.3.

Table 3.3 – Digital potentiometer specifications

IC	Supply range (V)	Resistance range (kΩ)	Steps	Bit requirement
MCP4141[32]	2.7 to 5.5	10	129	7
X9C103P[31]	-5 to 5	10	100	7

Considering a voltage supply of 5V, MCP4141 allows to sweep V_{cg} at 38mV per step and the X9C103P 50 mV per step and both need less than 8-bit to communicate with the device. Arduino Mega 2560 provides an 8-bit pulse width modulation (PWM) output, allowing the control of these devices.

For the current source, I_s, LM334 was chosen since it is an adjustable current source and is programmable from 1 µA to 10 mA by setting the value of R_7 as shown in figure 3.7[36].

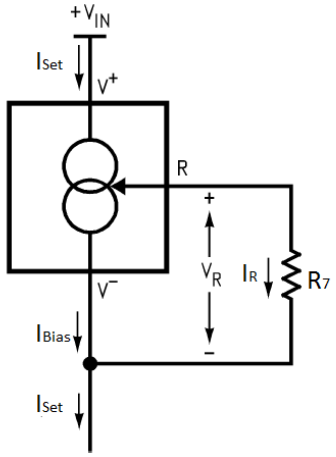

Fig. 3.7 – Adjustable current source schematic showing I_{set} as the sum of I_R and I_{bias} where I_R is controlled by the resistor R_7

The total current flowing through the device is the sum of I_R and I_{Bias}, and the current I_R is controlled by V_R and the value of the resistor R_7. According to the datasheet V_R is temperature dependable and has a value of 214 µV/°K. As so, for 25°C, $V_R \approx 64$ mV. Also, I_{set} is a percentage of the I_{bias} and according to the device datasheet, the ratio (n) I_{set} to I_{bias} is typically 18 for I_{set} currents ranging from 2 µA to 1 mA. So, the value of R_7 is defined as the following[36].

$$I_{set} = \frac{V_R}{R_7}\left(\frac{n}{n-1}\right) \Leftrightarrow R_7 = \frac{0.064}{I_{set}} \times 1.059 \quad (10)$$

As this current source is used as a versatility component, R_7 can be dimensioned to the user needs and can be easily substituted for a trimmer potentiometer or a digital potentiometer. For this project, a very low I_{set} is needed and therefore, a high-value resistor can be used. A 33kΩ resistor was used, meaning that I_{set} was set to approximately 2 µA. This results in a 2 µA current subtraction in the OA inverting node.

C_2 and C_3 were added to the circuit as filtering capacitors and they usually assume the value of 0.01 to 0.1µF capacitors for noise filtering[50]. For DC signal based circuits, these capacitors should be even larger (in the order of dozens of µF) for low-frequency filtering. More than one capacitor is added to filter more frequencies. This was not done and should be taken into account for further circuit development.

Considering that the signal shift to be amplified and measured is on the order of ≈3×10^{-7} A on the CD4007UB nMOSFET drain current, a TIA amplifier must be very precise, low-noise and with high gain since the signal can be easily perceived as noise. As a consequence AD8651 was picked as a TIA amplifier and for the buffer OA, a general-purpose LM324N was chosen. The considered properties for this device are presented in table 3.4.

Table 3.4 – Transimpedance and buffer amplifiers properties

Device	V_{cc} (V)	Gain (dB)	GBW (MHz)	V_{cm} (V)	Offset Voltage (mV)	Slew rate (V/µS)	Voltage output swing (V)	Input bias current (A)	Supply current (mA)
AD8651	2.7 to 5.5	115 dB	50 MHz	-0.1 to 5.1	0.1	41	Rail-to-Rail	1×10^{-12}	8
LM324N	3 to 32	100 dB	1.2 MHz	V^+ - 2	3	0.5	V^+ - 1.5	45×10^{-9}	0.8

Taking into account the OA specifications, both of the devices support the supply voltage of 5V, possess high-bandwidth product and high-gain. For the LM324N, gain and the offset voltage are not concerning parameters since it is going to be used as a unity gain buffer, as opposed to the TIA OA that ideally requires 0V offset voltage. The AD8651 has 0.1 mV offset voltage which can be considered practically 0V and for the 10-bit ADC, it is seen as 0 offset voltage since it doesn't have enough resolution to read 0.1 mV.

The common-mode voltage range gives information on what's the range of voltages that can be fed to the OA inputs and the voltage output swing informs about the maximum voltage that can be expected on the output. As so, the AD8651 supports all the range from 0 to 5V and has a voltage output swing rail-to-rail meaning that its minimum output voltage has the value of the negative supply, and for maximum output voltage has the value of the positive voltage supply. The LM324N properties are more concerning. For the common-mode voltage, using 5V as V^+, the maximum input voltage that can be used is 3V and the maximum output voltage gets restricted to 3.5V which certainly limits the usage of the circuit but provides enough range to set the required voltages V_a, V_s, and V_g.

The input bias current is very important to determine how much error it's introduced on the output relative to the input on the inverting or non-inverting node of the OA. As so, this current should be as low as possible to induce fewer errors on the measurements. For the AD8651 this current its on the order of the pA and for the LM324N on the nA range. There are three orders of magnitude difference between the two components.

Concerning the current needs, both OA can be used since the Arduino board as a V_{cc} absolute maximum output current of 200 mA and batteries deliver as much current as of the circuit needs until its totally discharged.

Comparing the two devices can be easily seen that to have better OA properties, more current is needed and how it's reflected on the device price as shown in table 2.1.

In table 3.5 it's present all the passive devices values used in the breadboard prototype.

Table 3.5 – Passive components values

R1 (kΩ)	R2 (kΩ)	R3 (kΩ)	R4 (Ω)	R5 (kΩ)	Rf (kΩ)	R7 (kΩ)	Cf (pF)	C2 (µF)	C3 (µF)
10	2	5,6	680	1	100	33	22	0.1	0.1

Using a 5v voltage supply, V_s is set to 0.83V and V_a, a variable voltage ranging from 0.54V to 5V. This way V_{ds} can be tuned as required. These values were picked thinking about taking measurements in the nMOSFET linear region. For other applications, another resistor and potentiometer values can be used, taking into account that for higher resistances the thermal noise also increases. A compromise between higher feedback gain and noise must be considered.

3.3. Firmware and software development

To perform the required measurements, the Arduino microcontroller needs to stablish digital communication with the digital potentiometers (MCP4141 and X9C103P). These two devices have different types of communication. The MCP4141 uses the Serial Peripheral Interface (SPI) (the connections can be consulted in Annex D) and the X9C103 communication is controlled with PWM pins. Therefore, these ICs were connected to the Arduino using different available connections.

To program the MCP4141, the SPI transfer function was used to set a user-defined wiper position. A for loop was created to set the wiper position from 0 to 127, covering all the wiper positions and analog-read functions were used to obtain data from the input V_{cg} and output V_{ads}. When the loop ends, the program stops by the introduction of break function.

To perform V_{adc} measurements over time using the MCP4141, a value for a wiper position is fixed and V_{adc} is monitored over a defined time.

Communicating with the X9C103 becomes facilitated by the use of libraries where functions to address wiper positions are already developed. 5 analog signals V_g, V_s, V_a, V_{offset}, and Vads were assigned to 5 analog pins where the signal is acquired by the Arduino board.

To perform measurements using the X9C103p, a for-loop is done containing 2 more for-loops that defines the number of data points per step (wiper position) and the number of steps the digital potentiometer should take. Every time the microcontroller obtains the defined number of data points, the wiper position drops one value and repeats the data acquisition loop. When these 2 for-loops (potentiometer steps and datapoints per step) are finished, one cycle is complete. When the defined number of cycles is complete, the program stops acquiring data.

Python language was used to develop 3 scripts:

ardFile.py – for generating a ready to upload Arduino file the user-defined parameters (number of steps and number of data points per step);

ardtoCsv.py – for downloading the data from the Arduino and write it in a .csv file;

Mod.py – an interface for fast data treatment and visualization.

The detailed firmware and Python software development considerations are presented in Annex E.

3.4. Circuit electrical characterization

After selecting the components required to build the circuit, a breadboard was used to assemble it and to connect the CMISFET sensor to test input and output signals.

The circuit was biased using 3 AAA batteries. Batteries are not an ideal power source. They have internal resistance (often 0.1 Ω to 1 Ω), as well as the battery contacts and wiring, add resistance to the supply circuit. So, the total resistance depends on the chosen components.

The Thevenin theorem establishes that any linear circuit can be presented as a voltage source and a series impedance representing the total impedance as seen from that point. The used supply circuit can be represented with an ideal 4.5V voltage source and a series resistance. According to Ohm's Law, the effective supply of the voltage source is now 4.5 V, minus the voltage drop across the equivalent resistor. So, the voltage supply is dependent on the circuit's current needs. The more current it needs, the less the voltage supply. This represents one disadvantage of using batteries.

Another disadvantage is that part of the dynamic range is lost and also they discharge over time. However, it offers two great advantages when compared to a bench power supply. It is less prone to induce errors as they maintain a steady flow of current, and allows to be a portable platform. For this reason, it was chosen to power the circuit with batteries.

The output performance of the circuit was tested by applying a 10mV step directly on the nMOS gate as shown by generating an output shift of around 185 mV in the linear region. This output curve suggests that the circuit is capable of converting the nMOS current into the output voltage as discrete values for each V_g step and so, showing the possibility to detect the surface potential shift. The output curve can be consulted in Annex F.

To investigate how the digital potentiometers perform, a full-scale sweep to all the wiper positions was done as presented in figure 3.8.

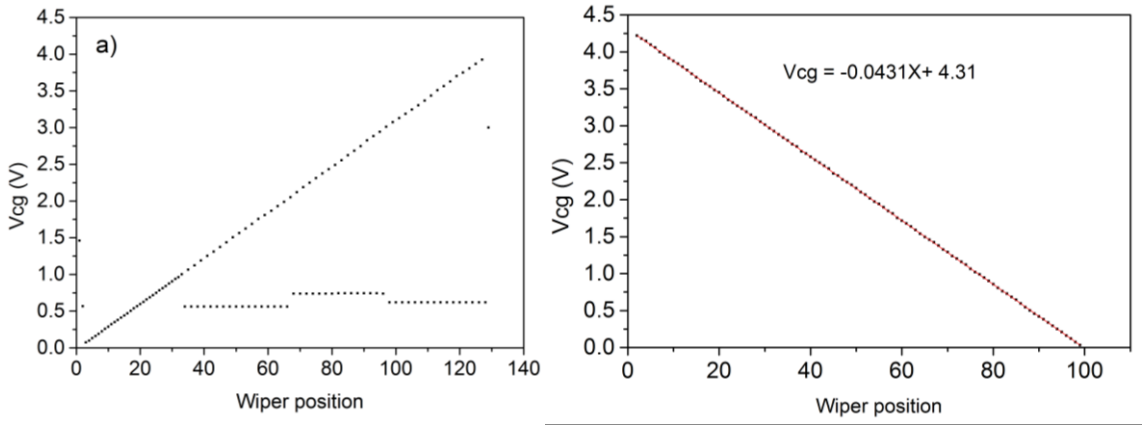

Fig. 3.8 - Digital potentiometer output voltage as a function of its wiper position for a) MCP4141 and b) X9C103P

During the prototype design, the MCP4141 seemed to be more promising since it has more wiper positions allowing better control over V_{cg}. However, results show that for some wiper positions its output doesn't follow the linearity, turning to be less reliable than the X9C103. The X9C103 has fewer wiper positions but it shows a truly linear response where the $r^2 = 0.999$. With a voltage supply of 4.3V volts, the X9C103P is capable of delivering 43 mV per step with a minimal error of approximately 0.02 mV/step. Data shows that only 4.3V were being delivered due to the circuit current needs and the equivalent resistance of the supply circuitry as described above.

The LM334N was also tested to determine if the unity-gain configuration is working and how much voltage is usable after buffering. The registered data is presented in figure 3.9.

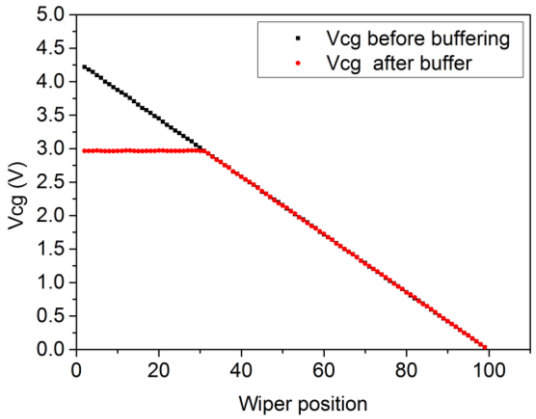

Fig. 3.9 - V_{cg} as a function of the X9C103 wiper sweep, before and after buffering using the LM324N showing the common-mode voltage limitation at approximately 3V.

From 0V to 3V, the data shows that before and after buffering the results are superimposed meaning that the LM324 OA is responding well with the unity-gain configuration. After 3V the output of the buffer is interrupted due to the OA V_{cm} (V^+ - 2). For this reason, the AD8651 (TIA OA) output voltage was not buffered to guarantee that the output would cover the entire dynamic range.

To acquire data from the prototype using a digital potentiometer, the method used involved a certain number of values per each datapoint since there's an error associated with each measure. So, having more data points means a more reliable measure. In figure 3.10 shows the AFE response to a V_g sweep using the digital potentiometer and working as an independent platform.

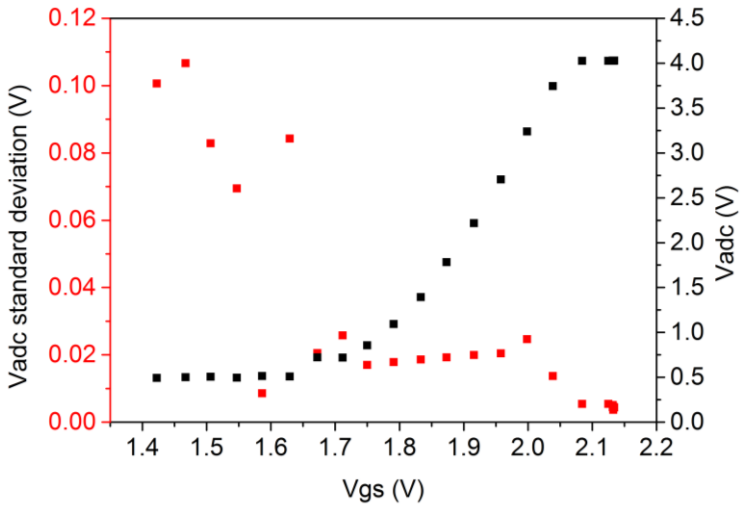

Fig. 3.10 – AFE circuit response and its respective output error of a Vgs sweep using X9C103P ranging from 2V to 2.7V, applying constant Vs=0.655V and Vds=10mV.

For this measurement, 10 data points per step were taken for Vadc and Vg resulting in a more stable and reliable signal showing the biggest error of 110mV and the lowest error of 4 mV. It's noticeable that the biggest error in the region where the nMOS is off. In the linear region, where the measurements are going to be performed, the V_{adc} error may vary between 17 to 25mV which can be a concern when monitoring low output differences.

V_{cg} and V_s were also tested to understand how much error they can introduce in the system. It was found that V_{cg} has a generally 4mV error, V_s a 2 mV error. However, the error may increase due to the susceptibility of the component-breadboard connections. V_a and V_{offset} remain constant when the transistor is in the depletion region.

V_g sweeping measurements with the MCP4141 were not done since the previous analysis showed that this IC could introduce too many outlier datapoints making the data unreliable.

3.5. Prototype response to pH change in the sensor

One of the major challenges was sensor-platform connectivity. All the process to connect the sensor to the prototype induced most of the error due to poor connection. Using crocodile connectors and silver glue was tested revealing to be unpractical solutions. The most effective solution was to solder two wires to an FPC (flexible printed circuit) connector.

Preceding any measurements, the conductivity of the sensor was always checked using two probes and then proceeded to the sensor encasing.

The produced sensors have different properties such as sensitive layer thickness, dielectric layer thickness, sensitive layer area. Thereby, its overall capacity changes according to these characteristics and, as a result, so does the sensor sensitivity. All structural details are present in Annex F [6].

3.5.1. Constant V_{cg} time response

3.5.1.1. Supplying V_{cg} a bench power supply

The first trials were done using a bench power supply to apply constant V_{cg} in the sensor and evaluate V_{adc} over time while changing pH solutions. Tests started with no solution, and then multiple solutions from pH 12 to 4 were used. The results are shown in figure 3.11. A 10 points moving-average was applied to the values, resulting in a cleaner display of the data allowing to easily identify the discrete output values for each pH as shown in figure 3.10 b. Applying V_{cg} = 2.5V and V_s = 0.83V sets the setpoint

to approximately 1.25V. By defining V_a = 0.868V, defines V_{ds} to 38 mV assuring that the measurement was taken in the nMOSFET linear region.

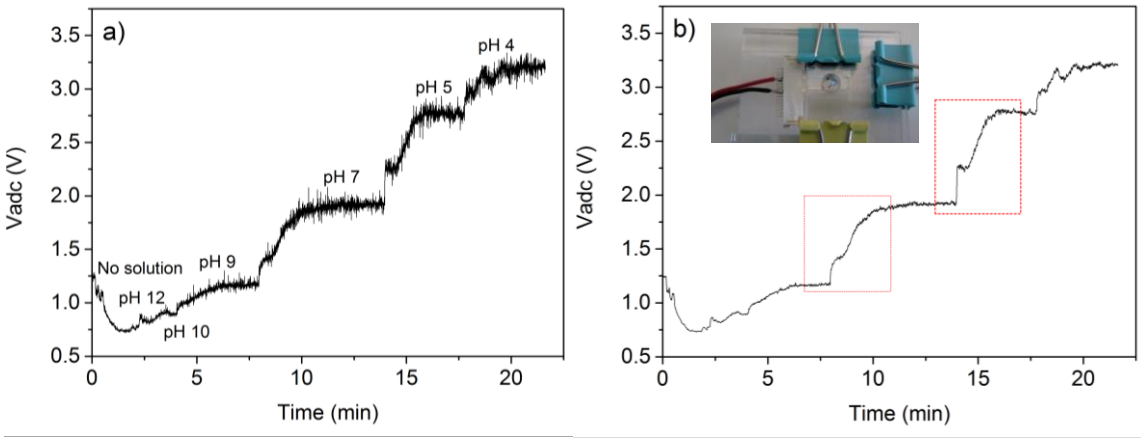

Fig. 3.11 – a) V_{adc} response over time applying constant V_{cg} = 2.5 V, V_s = 0.83 and V_{ds} = 38mV. Initially with no solution, followed by using pH 12,10,9,7,5 and 4 buffer solutions at room temperature and b) the 10 values moving-averaged data showing hyperbolic-like response highlight in red. A photo of the sensor is shown in the inset of the figure

Results show that basic solutions will force the output voltage to decrease as HO^- reduce the surface potential in the sensor. As the pH goes lower, V_{adc} rises, raising the surface potential. At the beginning of the measurement, it was not given enough time for the sensor stabilize and the output voltage for pH 12 and 10 is not reliable for measuring. It is also showing that, on average, the sensor needs approximately 3 to 4 minutes to stabilize every time there's a pH change.

The circuit output as a function of pH values and its corresponding error is presented in figure 3.12.

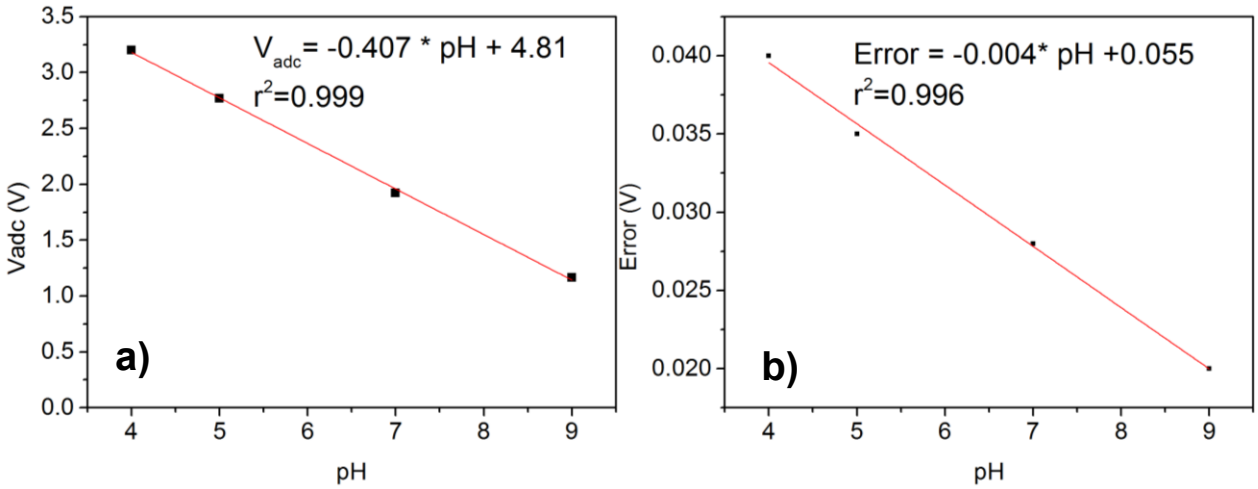

Fig. 3.12 – a) Linear regression applied to V_{adc} as a function of pH showing an r^2 of 0.99 and b) linear regression applied to the error, showing a 4 mV/pH error increase

Also, a mean value for ΔV_{adc}/pH was determined, resulting in a 0.276±0.031 V/pH with high linearity (r^2=0.99). Additionally, it's noticeable that the output error increases linearly with pH at a rate of 0.004V/pH.

Another interesting result is that hyperbolic-like response is observed in the pH 9 to 7 and pH 7 to 5 transitions. This behavior is modeled by the Grahame equation that correlates the surface charge density (σ_d) with surface potential according to:

$$\sigma_d = \sqrt{8c_0\varepsilon_l\varepsilon_0 k_B T}\,\sinh\left(\frac{q\Psi_0}{2k_B T}\right) \quad (11)$$

For lower surface potentials, part of the equation is simplified as sinh(x)≈x making the hyperbolic trend negligible. Since the hyperbolic behavior is observable, it comes as a conclusion that there's a high surface potential shift above 80 mV above the limit determined by the Nernst equation.[5][54]. This limit is calculated based on the Boltzmann model where the ions are assumed to be dimensionless, with no physical size. In 2017, a model was created describing the crowding effect, where a large number of counter-ions are attracted to the sensitive layer as a consequence of the applied voltage. It's reported that the presence and the size of the counter-ions change the profile of the electrolyte diffusion layer. If the surface potential goes above the critical potential (potential when the surface is ion-saturated), the counter-ions start to repel themselves, widening the electrolyte diffusion layer, resulting in a lower ion concentration in the oxide surface and a lower diffusion layer capacitance, contradicting the exponential increase of the double-layer capacitance predicted by Boltzman. So, a modified Boltzmann model distribution was proposed, taking into account the ion size. It's important to note that buffer solutions were used to perform these experiments. Buffer solutions have the property of restoring pH in the solution. They are made of weak acids and their conjugate base salts to counteract small changes in pH. When a pH change occurs, the conjugate base is dissociated and the ion concentration rises in the solution. When applying the modified Boltzmann model, another sensitivity parameter (δ) is introduced based on the ion size. If the ion is large enough, it may induce sensitivity above the Nernst limit. The model equations for sensitivity are as follows[52].

$$\frac{d\Psi_0}{dpH} = -2.3\frac{kT}{q}\left(\frac{1}{\frac{1}{\alpha}-\delta}\right) \quad (12) \qquad \delta = \frac{2a^3 c_B Sinh\left(\frac{1\Psi_0}{kT}\right)}{1+4a^3 c_B Sinh^2\left(\frac{q\,\Psi_0}{2kT}\right)} \quad (13)$$

Where α and δ are the sensitivity parameters that assume values between 0 and 1 and c_B are the counter-ion bulk concentration and a is the counter-ion size. Hannah instruments don't reveal which acids and conjugate bases are used in its buffer solutions so it's challenging to estimate if the sensitivity was influenced by the counter-ion size or not. Also, this model is accepted after a certain potential that is not specified in the literature.

3.5.1.2. Supplying V_{cg} with MCP4141

Previous results showed that the MCP4141 may not be suitable for the V_{cg} sweeping type of measurement. However, for ΔV_{adc} as a function of time is still an alternative. By setting the MCP4141 to wiper position 73, the setpoint is defined at 0.6V. The measurement took 16 minutes and its plotted results are presented in Annex F.

Applying different buffer solutions didn't change the response in the output. The first mistake was that the applied V_{cg} was setting the output at the lowest level for the ADC dynamic range. As so, by using a basic solution that sets the output to decrease, no answer is expected since the setpoint is already at its lowest level. For the pH 4 solution, it was expected for the signal to rise, however, it was not given enough time for ion trapping and the V_{adc} shift was not measured, or the sensor is not sensitive enough. These results showed two main considerations for pH measurements: the setpoint must be set to the middle of the dynamic range or at least to a value to which can be driven to decrease and increase and the sensor needs time to anchor ions. Also, the variable characteristics from sensor-to-sensor may induce a different response.

For a new sensor, after allowing it to stabilize, the setpoint was set to 1.5V and 20 µL of pH4 buffer was added to force the response to rise to the top of the ADC dynamic range. A 20 points moving-average was done. The results are displayed in figure 3.13.

The plot demonstrates that 120 seconds was not enough time for the sensor to stabilize at the current pH so a new measurement in the same conditions was done 5 minutes later. To understand what is the V_{adc} value for a pH4 solution the moving average was calculated for every 10 data-points, followed by the calculation of its average and its standard deviation. For pH4, V_{adc} = 3.494 ± 0.024V.

Annex F is showing the response for pH 12 solution. As the signal drops to the lowest limit of the range and it stabilizes. Applying the same error correction technique using the values from the stable region (V_{adc} from 70 s to 110 s), for a pH 12 V_{adc} = 0.654 ± 0.021 V. Considering theses values it was determined that, on average, there's a 0.234 V change per pH. The error showed to stay relatively constant at 0.022V.

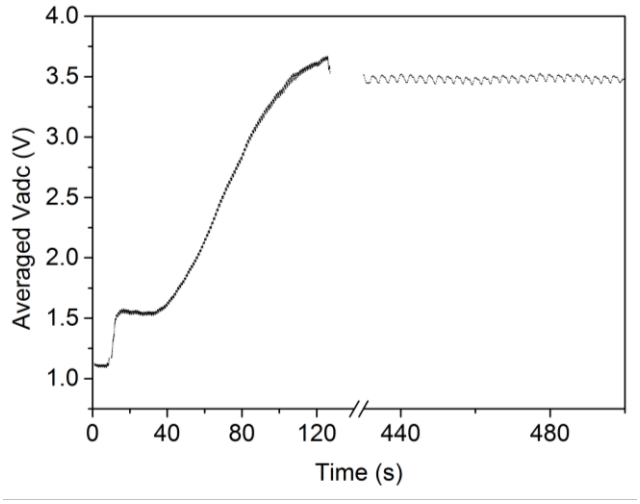

Fig. 3.13 - V_{adc} voltage response from dry-sensor to pH4 trough time with a 5 min break for allowing signal stabilization

3.5.1.3. Supplying V_{cg} with X9C103P

The X9C103P potentiometer was used to supply V_{cg}. Its wiper position was set to 37, placing the setpoint at approximately 2.25V. The buffer solutions were deposited, and the results over time are presented in figure 3.14.

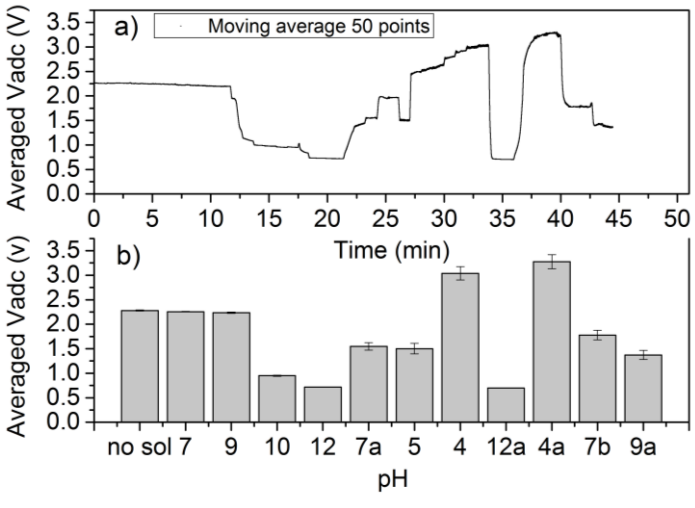

Fig. 3.14 - a) Vadc over time as function of pH in the following order: dry, pH 7, pH 9, pH 10, pH 12, pH7, pH 5, pH 4, pH 12, pH 4, pH7 and pH 9. b) shows the average Vadc value for each pH solution and its standard deviation.

During the first 12 minutes, the sensor was unresponsive to different pH solutions. This effect was seen in the previous measurements. This test response leads to thinking that using strong acid/basic solutions promotes tantalum oxide degradation and therefore, making more available sites for ion anchoring and unlocking the sensor response. So, in order to induce a response, the sensor requires

time or it is necessary to shock the sensor by introducing aggressive solutions that promote decomposition. An example of sensor degradation is presented in Annex F.

For pH 12, the output drops to the lowest readable signal and rises again when introduced to a pH7 solution. When introducing the pH 5 solution, V_{adc} doesn't go up most likely because it was not given enough time to stabilize. Then, when a pH 4 solution is introduced, it quickly responds and reaches the upper limit of the detection range. As a 12 pH is deposited, it sharply responds, dropping to the lowest limit and rises again when changing to an acid solution. When a neutral-acidity solution is contacting with the sensitive layer, the output drops to approximately half of the dynamic range and lowers 400mV more as a pH 9 solution is used. It's demonstrating that the platform responds very well as the input in the sensor changes. However, it's estimated to take approximately 10 to 15 minutes to guarantee a reliable output signal. Considering the Vadc associated with the pH 5 measurement as an outlier, it is shown that a 0.231 V/pH shift is measurable if considering a linear trend. However, it was found that an exponential trend fits better with the acquired results as shown in figure 3.15.

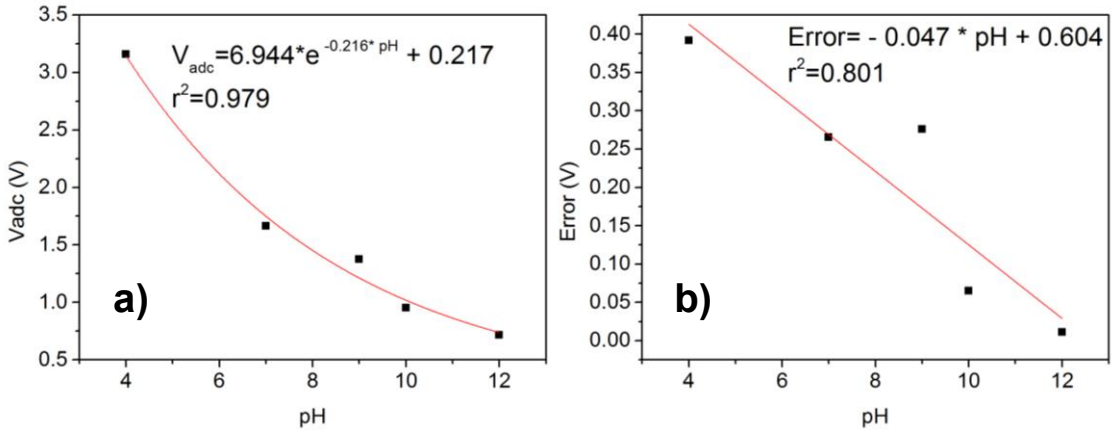

Fig. 3.15 - Exponential fitting in the Vadc as a function of pH for applying constant Vcg and linear fit in the output error

In this experiment, different pH values were introduced in an almost random manner in order to test how the sensor would react to different pH cycles instead of just increasing or decreasing. The exponential fit actually represents a more real response of what is expected. Considering that the ion concentration varies exponentially with the value of pH, it is expected an exponential charge-density accumulation in the oxide surface leading to the obtained results. The error associated with each measure follows an increasing linear trend for more acid solutions with a 47.9mV/pH rate.

3.5.2. V_{cg} Sweep Measurements

3 V_{CG} sweeps from 1.92V to 2.54V (potentiometer wiper position 50 to 25) were done for each pH solution obtaining 50 data-points per step for 25 steps. After placing the solution, it was let to stabilize for 10 minutes prior to performing each measurement. The solution deposition was done in the following order: pH 10, pH 9, pH7, pH5 and pH4. The data-points were averaged and its results are shown in figure 3.16.

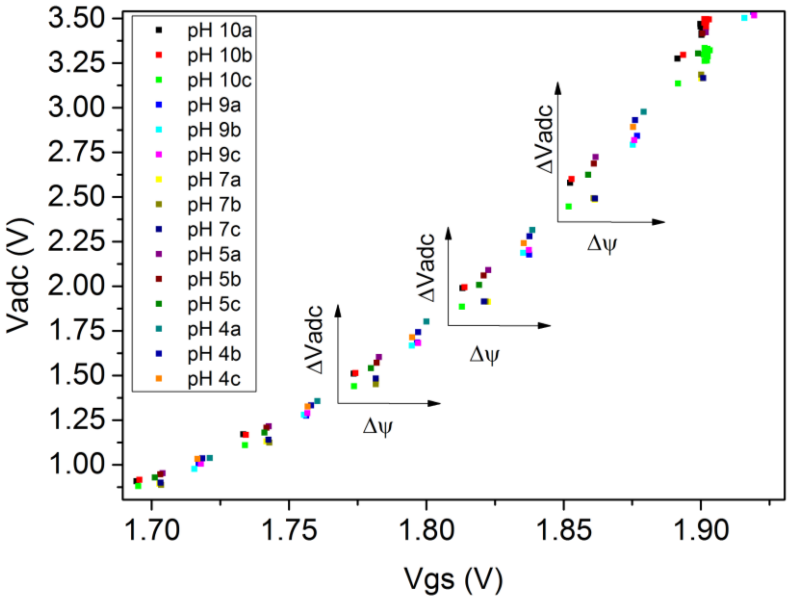

Fig. 3.16 – V_{adc} output as function of pH using constant V_s = 0.567 V and V_{ds} = 0.161 V, in the nMOSFET linear region

The first noticeable aspect of the results is that the CD4007UB nMOS is opening for higher values of V_{gs} than the 1.1V was suggested by the manufacturer. This happens due to the use of the current source that is subtracting current on the OA inverting input. So, from 1.1V the transistor is on but is not getting enough current to amplify until V_{gs} reaches approximately 1.6V. For future measurements, this should be taken into consideration. It is also important to note that the saturation point of the dynamic range is set to approximately 3.8V due to battery discharge. To evaluate the output change according to pH, the data was zoomed in the linear region and presented in figure 3.15 b).

For increasing acidity of the solution, there's a shift on the V_{adc} but also on V_{gs}. At first, it was thought that the V_{gs} shift could be induced from the V_{cg} supply, but the error from the digital potentiometer was determined to be ranging from 2 to 5 mV as opposed to approximately 20 mV that is shown. The V_{gs} signal is composed of the V_{cg} supply plus the surface potential change in the sensor, resulting in an increasing V_{adc} for higher V_{gs}.

The averaging of the 3 sweep measurements for each pH values revealed that the error associated with each measurement is very broad. The maximum obtained error for the V_{gs} was 25mV and for the V_{adc}, 325mV. Nevertheless, the raising V_{adc} trend as a function of pH is observed.

By comparing V_{gs} with V_{cg} it's possible to estimate how much surface potential difference is being caused by the charge density accumulated in the Ta_2O_5 layer. The results are displayed in table 3.6.

Table 3.6 – Surface potential variation estimative by comparing V_{gs} with V_{cg}

	pH 10	**pH 9**	**pH 7**	**pH 5**	**pH 4**
Δψ (V)	-0.079	-0.030	0.009	0.036	0.039
σ (V)	0.005	0.006	0.006	0.008	0.008

Considering the range from pH 4 to pH 10, it's estimated that the sensor sensitivity is 32mV/pH with 7mV error, which is around half of the maximum sensitivity stipulated by Nernst. Analyzing the error it is also demonstrated that for higher acidity, the error increases as previously demonstrated.

Next, V_{adc} was plotted as a function of V_{cgs} (V_{cg} -V_s) and presented in figure 3.17.

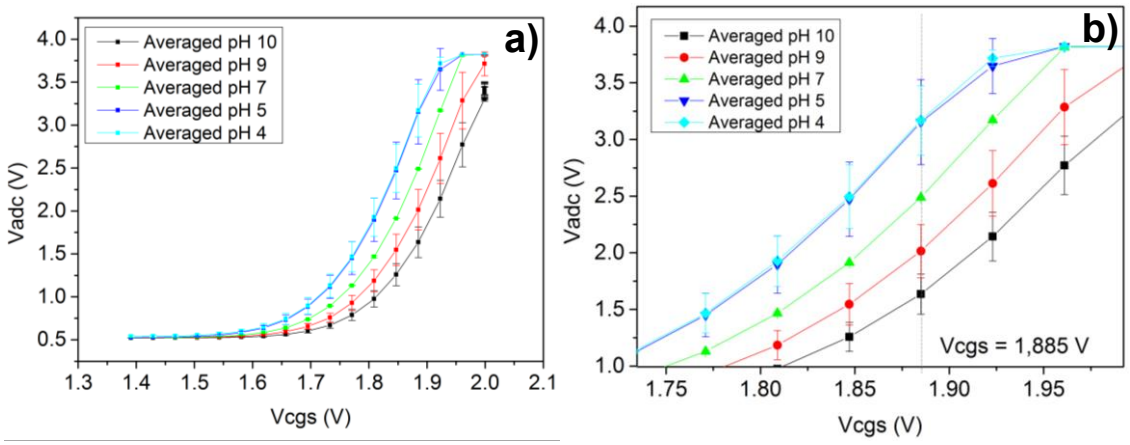

Fig. 3.17 – a) Vadc as a function of Vcgs for the averaged curves for each value of pH and b) a close-up plot int the nMOSFET linear working region

To evaluate how much voltage difference in output is observed by pH change, V_{cgs} was fixed at 1.847V. This value was chosen because it's where the biggest variation is observed and all the values are within the linear region.

The obtained sensibility is around 241mV/pH, proving to be consistent with the previous method of detection Also, the error raises as a function of pH. The data clearly show the threshold voltage modulation due to the electrolyte/oxide interface potential together with the oxide surface potential and the potential drop thought the dielectric as presented in figure 3.18[5].

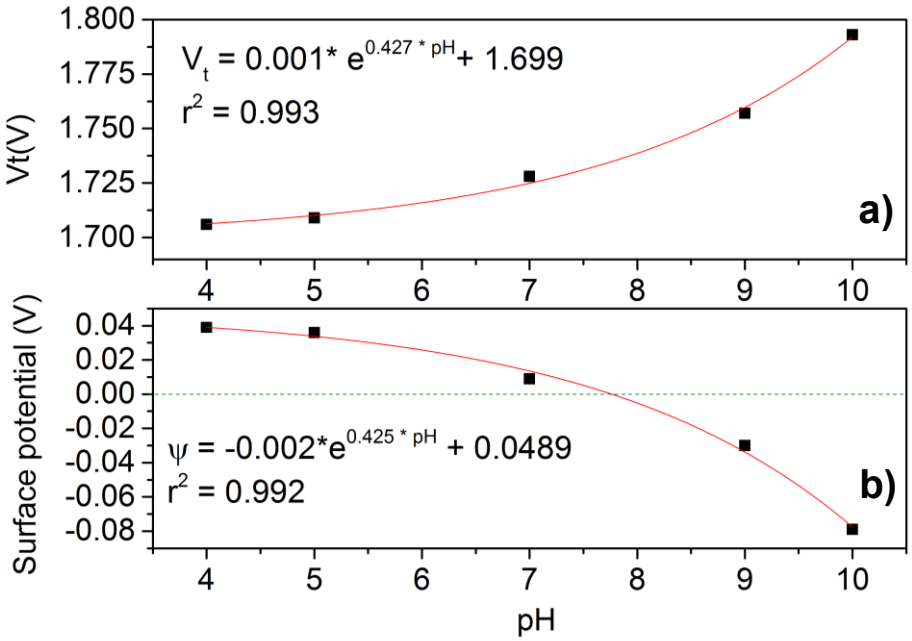

Fig. 3.18 – a) nMOSFET V_T modulation as a function of pH and its exponential fit parameters and b), surface potential variation and its exponential fit parameters with the reference for the point of zero charge in green, where it crosses the exponential trend

It is once again demonstrated that the surface potential, as well as the V_t modulation, varies exponentially according to pH. Another important characteristic that defines ion-sensitive oxides is the point of zero charge (pH_{pzc}). This factor describes the condition where the total electrical charge in the oxide is 0, as a function of pH. As the oxide as an intrinsic surface potential itself, ions counter-balance the oxide potential. As a result, the pH_{pzc} of the sensitive materials is often never pH 7.

According to the literature, the pH_{pzc} for the Ta_2O_5 varies according to the method of obtaining the material. For sputtered tantalum oxide, a study shows that the pH_{pzc} ranges from pH 5.6 to 6.6 in a temperature range from 5°C to 55°C[53]. Also, it is accepted that this value is often between pH 4 and 5[5]. However, the results demonstrate that pH_{pzc} is defined as a value between pH 7 and pH 8, which is different from the consulted literature. Further measurements should be performed to study this matter deeply.

2 types of measurements can be done to characterize pH reaction. By setting a constant value of V_{cg} and analyze the V_{gs} signal, the results give information about the sensor sensitivity. To analyze the output difference, V_{adc} must be swept in the function of V_{cg} only since the surface potential voltage is not being plotted.

Considering that all the above measurements resulted in approximately 235 mV/pH output shifts, the platform shows the capability of monitoring changes of 0.02 pH in the solution.

3.6. Lamp detection using the proposed platform

For this measurement, 285 V_{cg} sweeps were performed ranging from 1.9 to 2.6V, using 25 steps and taking 10 data-points per step. Each cycle took 23 seconds and an interval of 30 seconds between cycles was introduced, approximating the procedure to 1 cycle per minute. This way the input signal approximates to a 0.018Hz quadrangular input waveform that it is presented in Annex F. In total, this measurement took 4.26 hours instead of the usual 1,5 h for standard LAMP reactions. Since the temperature conditions/overall setup is not optimal, it was chosen to make longer measurements. The raw obtained data by the software-hardware interface is shown in figure 3.19.

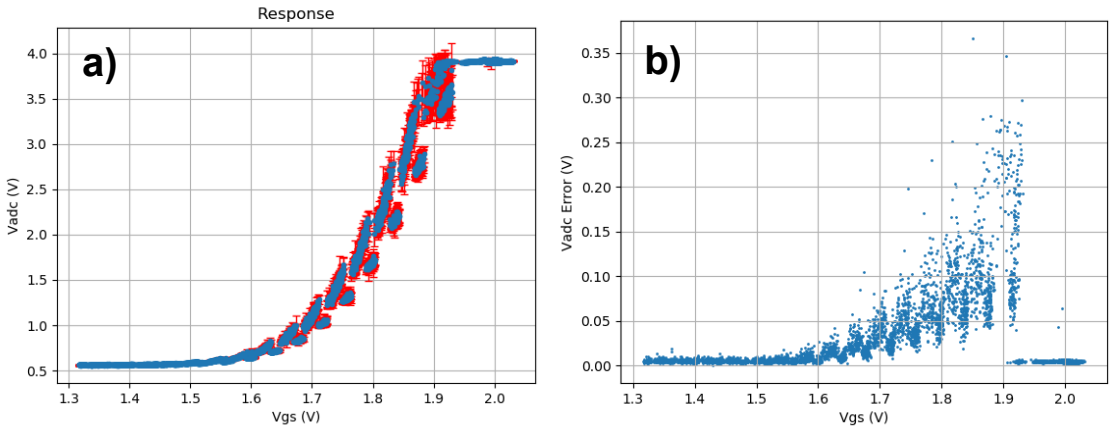

Fig. 3.19 – a) LAMP monitoring data for the 285 V_{cg} sweeping cycles and b) its error plotted as a function of V_g, showing increasing error for an increasing number of cycles.

At first instance, it is clear that for the same V_g, V_{adc} is apparently increasing over-time indicating the possibility of real-time amplification monitoring. Considering the error over-time, and as a function of V_g, it's noticeable that it increases and it's bigger for higher values of V_g. However, it's showing that there's higher error density ranging from 25 to100 mV in the first cycles and that for the last cycles the error increased exponentially. Error plots come as an important indicator of which region the analysis should be done, by choosing the lowest error region in the linear nMOS behavior. The 10^{th} step shows a lower error but also a lower Vadc shift. However, it's out of the depletion region. For this reason, it was chosen to analyze the data in the 13^{th} step where V_{gs} = 1.8V.

By addressing a constant V_{cg}, the V_{adc} and V_{gs} variation can be analyzed as shown in figure 3.20.

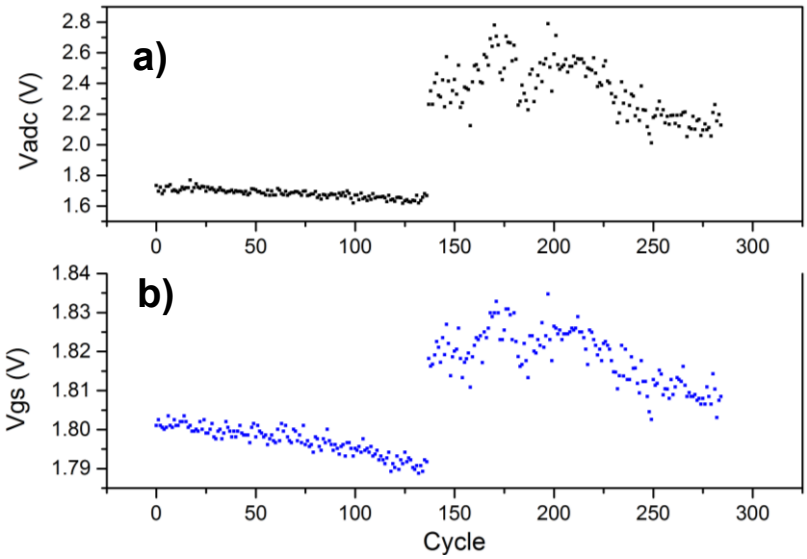

Fig. 3.20 – a) Vadc variation as a function of cycle number showing a 0.5V difference between cycle 137 and 250 and b) the potential difference in the electrode as a function of cycle number showing a peak voltage shift due to a V_s instability

Results show that for the first 136 cycles, Vadc as a tendency to decrease due to the surface potential change in the sensor. A report shows that for a LAMP reaction, its pH change from around 8.8 to around 5 pH, so the output is decreasing for higher values of pH like showed before[54]. After 136 cycles the V_{gs} signal increases 20 mV and, as a consequence, V_{adc} increases by 600mV and stabilizes again after 230 cycles. It was thought that it was due to any circuit disturbance like a touch in a wire or a due to table vibration. The dataset was inspected further and it was found that a variation of 20mV occurred in Vs and as a consequence, the output raised accordingly. Only a 3 mV error was expected on V_s, however, the debility of the breadboard platform is shown to have severe consequences on LAMP reaction monitoring with a risk of inducing misleading results.

Nevertheless, the output showed signs of possible real-time amplification of the c-Myc oncogene, so a new set of data was analyzed using the same criteria. By setting the Vadc constant at 1.78 V, the V_g shift was plotted and compared to a real-time fluorescence data and presented in figure 3.21.

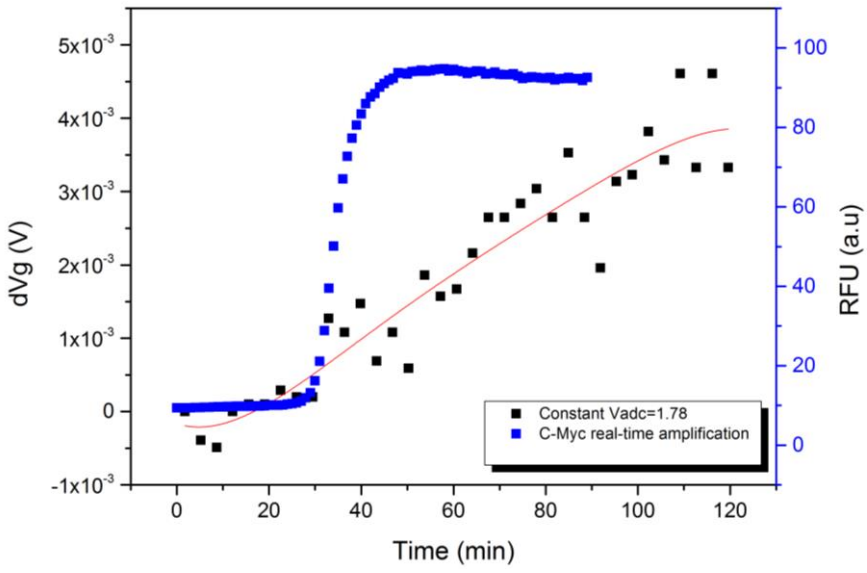

Fig. 3.21 – V_g shift over time and its trend line compared to a real-time amplification curve.

The amplitude of the ΔVg signal is around 4mV meaning that for LAMP monitoring, a larger feedback resistor should be used in order to increase the signal further. However, it is shown that its

response follows the trend of a real-time amplification although it's shown that the reaction occurs at a much slower pace than the fluorescence method. Around the 20-minute mark, it is shown that the Vg voltage is starting to shift and showing signs of the beginning of the amplification reaction, almost the same time when compared to the fluorescence curve.

As the amplification occurs, exponentially generates more H^+ but for the sensor to sense it, they need to anchor to the sensitive oxide and its highly dependent on the available sites. When compared to fluorescence, this process may be much slower even if there's ion availability.

It's also important to note that LAMP reactions are extremely sensitive to contaminants and the setup conditions should be as stable and sterile as possible. For these particular measurements, the solution deposition well was used and reused without proper cleaning and so, very susceptible to introduce false positives in the current tests. For this reason, the results are inconclusive and a stable circuit is highly needed. To do so, 2 PCBs were designed.

3.7. PCB implementation

2 PCBs were designed according to the schematics and layouts present in Annex G and H and sent to the manufacturer. In figure 3.22 is shown the corrected V1.0 PCB stacked with the Arduino and connected to the sensor as the final obtained platform for this master thesis. All the PCB design considerations are presented in Annex I.

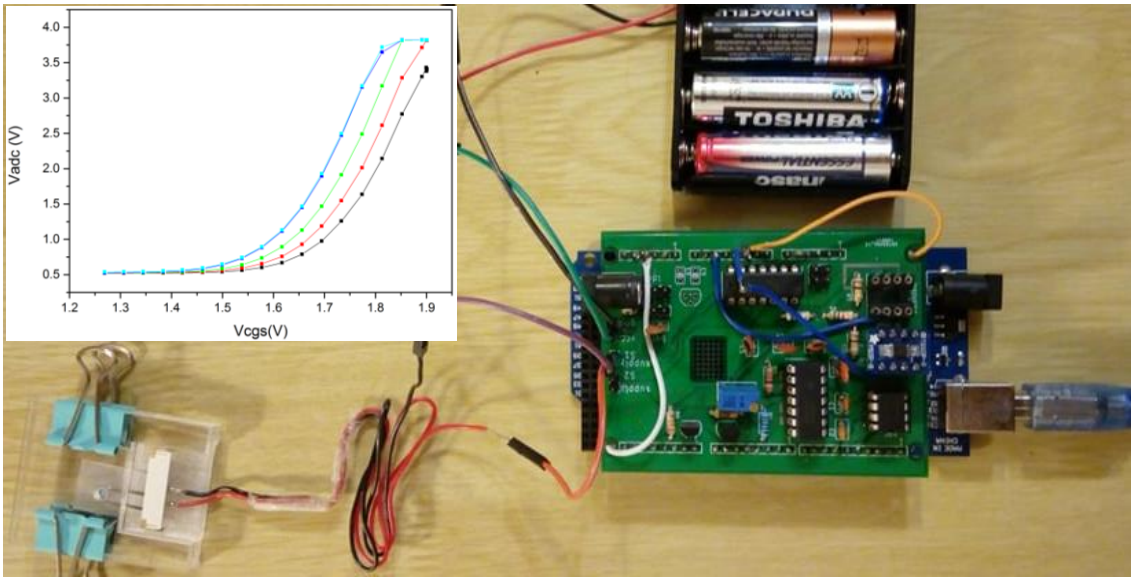

Fig. 3.22 – Stand-alone platform showing the corrected V1.0 PCB, showing the required extra wiring to analog-read pins. The circuit is stacked with the Arduino board and connected to the sensor and the battery voltage supply. An example of the obtained results is embedded in the figure.

4. Conclusions and future perspectives

A working prototype to acquire and amplify a signal caused by a surface potential shift in tantalum oxide ion-sensitive sensors was developed. An electronic study was done recurring to a simulation that allowed to prove the concept of amplifying small current difference from the nMOSFET and converting it to a readable voltage at the microcontroller. A transimpedance amplifier based circuit with a digital potentiometer allowed the control over V_{cg} applied to the sensors. Two different potentiometers were tested and the firmware was developed in order to send and receive data from the AFE circuit. To evaluate its performance, 2 types of tests were performed. Applying constant voltage in the sensor control contact and measure the circuit output over time, and sweeping type of measurements. The prototype was tested using buffer solutions with defined pH that induce surface potential difference. Later, LAMP reaction monitoring was performed using the prototype. As the prototype proved to be working, it was implemented in 2 versions of a PCB in order to improve output stability.

4.1. Final conclusions

Simulations showed that to convert a range of 50 µA to a 5V dynamic range, a 900k resistor was needed. However, by choosing a higher gain OA than the one used in the simulation, a 100k resistor proved to be enough to convert and amplify the signal. It was also found that a 10-bit ADC is enough to process the output voltage difference, meaning that an Arduino microcontroller can be used to acquire data successfully. Since the platform is working on a DC signal basis, bandwidth is not a major concern. The working frequency was set to 5kHz for the calculus of the capacitor resulting that $C_f \leq 35pF$ and with an OA with GBW greater than 7.8 kHz assures the circuit proper functioning.

It was required to have control over the applied voltage in the sensor, V_{ds} and the output of the circuit. This was achieved by dimensioning voltage dividers with unity-gain buffer amplifiers. V_{ds} was designed by keeping V_s constant at 0.83V and V_{offset} (drain voltage) tunable by adding a potentiometer to the voltage divider that allows setting V_{ds} at a maximum of 5V (considering a voltage supply of 5V) and a minimum of 0.54V. In this case, it was of interest to perform measurements in the linear region of the nMOS in order to improve the linear response of the output. The introduction of the potentiometer was done thinking of future applications that may require other setup parameters. At the beginning of this work, it was thought that a current subtractor was needed to control the output. Later results showed that due to the nature of the surface potential difference in the sensor, it is not needed. Nevertheless, it was dimensioned to subtract approximately 2µA. To control the voltage applied in the sensor, 2 digital potentiometers were programmed revealing that the MCP4141 may not be suitable for V_{cg} sweeping since it introduces many outlier values. Poor programming could be the cause and not a defective IC. However, the device was working properly for constant voltage over time measurements. The x9c103P showed perfect step linearity with 43 mV per step with a minimal error of approximately 0.02 mV/step. For this reason, the X9C103P was chosen to integrate the prototype.

Electrical characterization showed that the LM324N buffer OA was not the best choice due to its common-mode voltage that limits its output in 2V. It was showed that 3V was enough to drive the nMOS until saturation and so the measurements were not compromised due to this limitation. However, the output of the circuit wasn't buffered due to this setback. The use of the current source as subtractor showed that the nMOSFET is opening at higher voltages (1.6V instead of the 1V) and also V_s is set too high, forcing the V_g to be also higher. Although the platform worked well with these values, there's room for optimization. Also, to avoid supply induced error, battery-based power supply was chosen as opposed to the Arduino supply. The Arduino supply may always guarantee a 5V supply but it induces noise in the circuit due to the USB connection. By using batteries, part of the dynamic range is lost due to the current needs and also it discharges over time. However, batteries maintain a steady DC flow of current, allow portability and are less prone to induce extra noise.

Prior to the test of the independent platform, It was found that by applying a 10 mV step directly at the nMOS gate, the circuit is able to amplify the FET current to approximately 185mV/step. According to the reference results, the sensor's sensitivity was reported to be around 12.2 mV/pH, proving that the circuit is able to detect small surface potential variations.

The platform was tested using pH buffer solutions in order to effectively tell if the signal can be detected. A constant voltage applied to the sensor's control-gate and recording the circuit output was done. Using a bench power supply, revealed a 276 $\pm$ 31 mV/pH output shift with increasing error for lower pH at a linear rate of 4mV/pH. Then using the MCP4141 as V_{cg} supplier, a 234 $\pm$ 12 mV/pH was determined. Using the X9C103P, the output was 231 $\pm$ 46 mV/pH. Vcg sweep type of measurement using the X9C103P showed 241$\pm$219 mV/pH. By averaging all the output sensitivity it is demonstrated that the circuit is capable of detecting 235 mV/pH output shift. Considering the resolution of the ADC, it is possible to monitor 0.02 pH changes in the solution.

All of the measurements showed higher error for lower pH solutions that lead to the conclusion that, as charge density raises, induces instability in the sensitive oxide surface due to thermal effects together with the intrinsic noise of the digital potentiometer. It also showed that the MCP4141 provides a more stable signal than the X9C103P and so, for constant V_{cg} over-time type of measurement the MCP4141 is proven to be more reliable. V_{cg} sweep type of measurements is necessary to understand in which region these tests should be performed. As so, it was chosen to develop software to obtain data from the V_{cg} sweeping type of results for the final application.

V_{cg} sweeping also allowed to estimate how sensitive the sensors are by comparing V_{gs} with V_{cgs}. For that particular sensor, a sensitivity of 32$\pm$7 mV/pH was estimated, which is around half of the Nernst stipulated limit. Also, the V_t modulation was observable and it was determined that it follows an exponential trend as a result of the charge density accumulation.

LAMP reaction monitoring showed that the amplification can indeed be monitored as there's an increase of the output signal as time elapses. However, it reveals to be a slower detection method compared to fluoresce since it's relying on ion binding in the sensitive layer instead of monitor the actual number of DNA strands. LAMP reaction possesses plenty of different reagents making it more difficult to understand whether the actual readings are coming from the amplification reaction of from ionic binding of other products of the reaction. Nevertheless, it was shown that a precarious amplification-like response showing the denaturation, annealing, and elongation phases. In this measurement, also the error rises as the pH goes lower, remaining a challenge to suppress it.

4.2. Future Perspectives

Some aspects should be investigated to improve circuit performance. The current source is not necessary and so, shouldn't be used. Change the LM324N buffer amplifiers for a rail-to-rail V_{cm} and use it to also buffer the transimpedance amplifier. Dimension and optimize the use of a low-dropout voltage regulator in order to supply the circuit with a noiseless, steady 5V supply. Investigate further on digital potentiometers that allow more steps and as a result a better control over the measurements. Improve sensor-platform connection since the current method revealed unpractical and difficult to handle. Add analog-read signals from V_{gs}, $V_{cg,}$ and V_{cc} for extra control. Future circuits at each IC input 3 filtering capacitors should be added with values (0.01 µF, 0.1 µF, 10µF) and also extra decoupling capacitors (0.1 µF) in the V_{cc} supply. Code optimization is also advisable. Instead of printing the values as they are being acquired, gather all the data and then print it at once. This strategy could be used to reduce the noise induced by the USB connection. Removing completely the USB connection could also be tested, by supplying the Arduino externally and save the acquired data to an SD card. Improve sensor encapsulation and connection in order to minimize as many moving parts as possible and use a metal encasing to place the microcontroller and circuit that connects to a different one where the sensor is inside a heat sink that is maintained at 65ºC. The platform should not be placed in the same space as the sensor since the electronics are temperature sensitive.

5. References

[1] World Health Organisation, “Global cancer data,” Geneva, 2018.

[2] R. Luengo-Fernandez, J. Leal, A. Gray, and R. Sullivan, “Economic burden of cancer across the European Union: A population-based cost analysis,” Lancet Oncol., vol. 14, no. 12, pp. 1165–1174, 2013.

[3] M. and Markets, “Life Science Analytics Market Global Forecast - 2024 | MarketsandMarkets,” 2019. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/pharmaceutical-life-science-analytic-market-174990653.html. [Accessed: 05-Aug-2019].

[4] NASA, “Computers in Spaceflight: The NASA Experience,” 2019. [Online]. Available: https://history.nasa.gov/computers/Ch2-5.html. [Accessed: 15-Sep-2019].

[5] R. M. M. Branquinho, “Label-free detection of biomolecules with Ta2O5-based field effect devices,” Ph.D dissertation, Department of Materials Science, FCT/UNL, 2012. Accessed on: 10-04-2019, Available: https://run.unl.pt/handle/10362/9413

[6] B. M. R. Veigas, “Development of Field Effect sensors for gene expression analysis : Application to cancer diagnosis,” Ph.D dissertation, Chemistry Department, FCT/UNL, 2016. Accessed on: 08-02-2019, Available: https://run.unl.pt/handle/10362/19083

[7] Y. Li, P. Fan, S. Zhou, and L. Zhang, “Loop-mediated isothermal amplification (LAMP): A novel rapid detection platform for pathogens,” Microb. Pathog., vol. 107, pp. 54–61, 2017.

[8] S. H. Lee, S. min Park, B. N. Kim, O. S. Kwon, W. Y. Rho, and B. H. Jun, “Emerging ultrafast nucleic acid amplification technologies for next-generation molecular diagnostics,” Biosens. Bioelectron., vol. 141, no. 111448, 2019.

[9] S. B. Kary Mullis et al., “Process For Amplifying, Detecting, And/Or-Cloning Nucleic Acid,” United States Patent 4683195, Jul. 28, 1987.

[10] The Nobel Foundation, “The Nobel Prize in Chemistry 1993,” 2019. [Online]. Available: https://www.nobelprize.org/prizes/chemistry/1993/summary/. [Accessed: 29-Jul-2019].

[11] H. Zhang et al., “Revealing the Secrets of PCR,” Sensors Actuators B Chem., vol. 298, no. 126924, 2019.

[12] T. Notomi et al., “Loop-mediated isothermal amplification of DNA,” Nucleic Acids Res., vol. 28, no. 12, 2000.

[13] J. M. Rothberg et al., “An integrated semiconductor device enabling non-optical genome sequencing,” Nature, vol. 475, no. 7356, pp. 348–352, 2011.

[14] H. R. Zare and Z. Shekari, "Chapter 6 - Types of monitoring biosensor signals," in Electrochemical Biosensors, Elsevier Inc., 2019.

[15] R. Adzhri et al., “High-performance integrated field-effect transistor-based sensors,” Anal. Chim. Acta, vol. 917, pp. 1–18, 2016.

[16] P. Bergveld, “Thirty years of ISFETOLOGY,” Sensors Actuators B Chem., vol. 88, no. 1, pp. 1–20, 2003.

[17] C. S. Lee, S. Kyu Kim, and M. Kim, “Ion-sensitive field-effect transistor for biological sensing,” Sensors, vol. 9, no. 9, pp. 7111–7131, 2009.

[18] J. F. V. Pérez, M. M. M. Velasco, M. E. M. Rosas, and H. L. M. Reyes, “ISFET sensor characterization,” Procedia Eng., vol. 35, pp. 270–275, 2012.

[19] B. Veigas et al., “Quantitative real-time monitoring of RCA amplification of cancer biomarkers mediated by a flexible ion sensitive platform,” Biosens. Bioelectron., vol. 91, pp. 788–795, 2017.

[20] M. Kaisti, “Detection principles of biological and chemical FET sensors,” Biosens. Bioelectron., vol. 98, pp. 437–448, 2017.

[21] M. Barbaro, A. Bonfiglio, and L. Raffo, “A charge-modulated FET for detection of biomolecular processes: Conception, modeling, and simulation,” IEEE Trans. Electron Devices, vol. 53, no. 1, pp. 158–166, 2006.

[22] M. Kosmulski, “Oxide/electrolyte interface: electric double layer in mixed solvent systems,” Colloids Surfaces A Physicochem. Eng. Asp., vol. 95, no. 2–3, pp. 81–100, 1995.

[23] C. Falconi and S. Mandal, “Interface electronics: State-of-the-art, opportunities and needs,” Sensors Actuators, A Phys., vol. 296, pp. 24–30, 2019.

[24] M. Di and P. Emilio, "Chapter 2 - Data Acquisition Systems: Hardware," in Data Acquisition Systems: From Fundamentals to Applied Design, M.EMilio, Ed. New York: Springer, 2013, pp 11-38

[25] P. R. Gray, "Chapter 8 - Feedback" in Analysis and Design of Analog Integrated Circuits, 5th ed.,V.Vargas, Ed. New York: John Wiley & Sons inc., 2001, pp. 553-633

[26] E. Säckinger, “Transimpedance Amplifier Circuit Examples,” Analysis and Design of Transimpedance Amplifiers for Optical Receivers, no. February 2018. pp. 373–396, 2017.

[27] J. Caldwell, “Transimpedance Amplifiers: What Op Amp Bandwidth do I Need?,” no. 1, pp. 1–8

[28] E. W. John Park, Steve Mackay, Practical Data Communications for Instrumentation and Control, Oxford: Elsevier,2003, pp. 124–145.

[29] C. S. Young, Information Security Science: Measuring the Vulnerability to Data Compromises, New York: Elsevier, 2016, pp. 103-121

[30] John Semmlow, Chapter 15 - Basic Analog Electronics: Operational Amplifiers In Circuits, Signals and Systems for Bioengineers, 3rd ed., J. Semmlow, Academic Press,2018, pp. 681-723.

[31] T. Instruments, “Principles of data acquisition and conversion Part 1,” Microprocessors and Microsystems, vol. 3, no. 10. p. 467, 1979.

[32] Renesas, “X9C103 - Digitally Controlled Potentiometer (XDCPTM),” FN8222 datasheet, Jan 11, 2019, [Rev 4.0]

[33] Microchip, “MCP414X/416X/424X/426X,7/8-Bit Single/Dual SPI Digital POT with Non-Volatile Memory,” DS22059B datasheet, 2008 [Revised Dec. 2008].

[34] Texas Instrument, “CD4007UB Types,” SCHS018C datasheet, 2003 [Revised Sep. 2003].

[35] Texas Instruments, "LMx24-N, LM2902-N Low-Power, Quad-Operational Amplifiers", SNOSC16D datasheet, March 2000 [Revised Jan, 2015]

[36] Analog Devices, “50 MHz, Precision, Low Distortion , Low Noise CMOS Amplifiers,”AD8651/AD8652 datasheet, 2006 [Rev. D]

[37] Texas Instruments, “LM134 / LM234 / LM334 3-Terminal Adjustable Current Sources,” SNVS746E datasheet, March 2000 [Revised May, 2013]

[38] "PCB Prototype & PCB Fabrication Manufacturer - JLCPCB", Jlcpcb.com, 2019. [Online]. Available: https://jlcpcb.com/. [Accessed: 30- Sep- 2019]

[39] "Arduino Mega 2560 Rev3", Store.arduino.cc, 2019. [Online]. Available: https://store.arduino.cc/arduino-mega-2560-rev3. [Accessed: 30- Sep- 2019]

[40] “Arduino Software.” [Online]. Available: https://www.arduino.cc/en/main/software. [Accessed: 08-Sep-2019].

[41] philbowles, “Arduino-X9C.” [Online]. Available: https://github.com/philbowles/Arduino-X9C/.

[42] P. S. Foundation, “Python,” 2019. [Online]. Available: python.org. [Accessed: 10-Sep-2019].

[43] N. Developers, “NumPy,” 2019. [Online]. Available: https://numpy.org/index.html. [Accessed: 10-Sep-2019].

[44] J. V. den B. Tom Augspurger, William Ayd, Chris Bartak, Pietro Battiston, Phillip Cloud, Marc Garcia, Andy Hayden, Masaaki Horikoshi, Stephan Hoyer, Wes McKinney, Brock Mendel, Jeff Reback, Matthew Roeschke, Jeremy Schendel, Chang She, “Pandas,” 2019. [Online]. Available: https://pandas.pydata.org/index.html. [Accessed: 10-Sep-2019].

[45] R. Y. Joris Van den Bossche, Loïc Estève, Thomas J Fan, Alexandre Gramfort, Olivier Grisel, Yaroslav Halchenko, Nicolas Hug, Adrin Jalali, Guillaume Lemaitre, Jan Hendrik Metzen, Andreas Mueller, Vlad Niculae, Joel Nothman, Hanmin Qin, Bertrand Thirion, Tom Dup, “Scikit-learn,” 2019. [Online]. Available: https://scikit-learn.org/stable/index.html. [Accessed: 10-Sep-2019].

[46] J. D. Hunter, “Matplotlib: A 2D graphics environment,” Comput. Sci. \& Eng., vol. 9, no. 3, pp. 90–95, 2007.

[47] P. S. Foundation, “Tkinter.” [Online]. Available: https://docs.python.org/3/library/tkinter.html#module-tkinter. [Accessed: 10-Sep-2019].

[48] P. S. Foundation, “Math.” [Online]. Available: https://docs.python.org/3/library/tkinter.html#module-tkinter. [Accessed: 10-Sep-2019].

[49] P. S. Foundation, “Shutil,” 2019. [Online]. Available: https://docs.python.org/3/library/shutil.html. [Accessed: 10-Sep-2019].

[50] Texas Instruments, “[Resolved] SPICE Model for CD4007UB - Simulation, hardware & system design tools forum - Simulation, hardware & system design tools - TI E2E support forums,” 2019. [Online]. Available: https://e2e.ti.com/support/tools/sim-hw-system-design/f/234/t/323921#. [Accessed: 19-Sep-2019].

[51] Analog Devices, “Decoupling Techniques: WHAT IS PROPER DECOUPLING AND WHY IS IT NECESSARY? ,” Tutorial MT-101, 1996.

[52] Louis E. Frenzel, Chapter Thirty-Five - Serial Peripheral Interface (SPI) in Handbook of Serial Communications Interfaces, Louis E. Frenzel, Oxford: Newnes, 2016, pp. 143-145.

[53] Arduino, “SPI settings.” [Online]. Available: https://www.arduino.cc/en/Reference/SPISettings. [Accessed: 19-Sep-2019].

[54] Jacob N. Israelachvili, Chapter 14 - Electrostatic Forces between Surfaces in Liquids in Intermolecular and Surface Forces, 3rd ed., Jacob N. Israelachvili, Academic Press, 2011, pp. 291-340.

[55] K. B. Parizi, X. Xu, A. Pal, X. Hu, and H. S. Philip Wong, “ISFET pH Sensitivity: Counter-Ions Play a Key Role,” Sci. Rep., vol. 7, no. 3, pp. 1–10, 2017.

[56] J. C. Chou and L. P. Liao, “Study on pH at the point of zero charge of TiO2 pH ion-sensitive field effect transistor made by the sputtering method,” Thin Solid Films, vol. 476, no. 1, pp. 157–161, 2005.

[57] N. A. Tanner, Y. Zhang, and T. C. Evans, “Visual detection of isothermal nucleic acid amplification using pH-sensitive dyes,” Biotechniques, vol. 58, no. 2, pp. 59–68, 2015.

[58] Hank Zumbahlen, CHAPTER 12 - Printed Circuit-Board Design Issues in Linear Circuit Design Handbook, Hank Zumbahlen, Newnes,2008, pp. 821-895.

[59] PCB Trace Width Conversion Calculator | DigiKey", Digikey.com, 2019. [Online]. Available: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width. [Accessed: 30- Sep- 2019]

[60] B. Carter, “Circuit Board Layout Techniques" in Op Amps for Everyone, B. Carter, Texas Instruments, 2002, pp. 349-378.

6. Annexes

Annex A : **Analog circuit schematic**

Fig. 6.1 – Analog circuit schematic showing all the components and the required connections to operate with the Arduino board

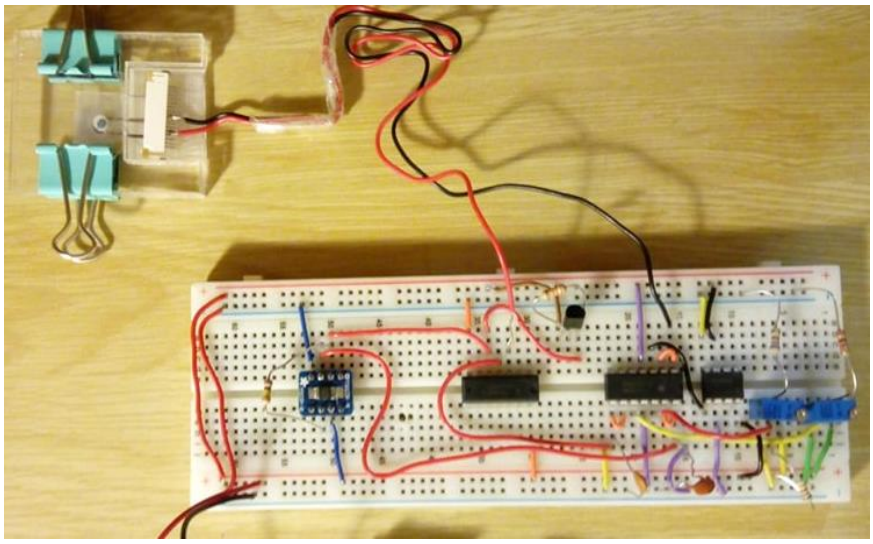

Fig. 6.2 - Analog circuit implementation in a breadboard. Arduino connections were removed to make the figure more clear

Annex B : **Arduino code to communicate with the digital potentiometers**

- MCP4141

```
#include <SPI.h>
const int cspin = 10;
int ads = A4;
int gate = A0;
void setup() {
Serial.begin(9600);
pinMode (cspin, OUTPUT);
SPI.begin();
}
void loop() {
digitalWrite(cspin, LOW);
delay(100);
SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE0));
//-------------MCP4141 WIPER POSITION ADDRESING---------------//
for (int i=0; i<=127; i++){
delay(50);
SPI.transfer(0);
SPI.transfer(i);
float ads = analogRead(ads)*(5.0/1023.0);
float gate =analogRead(gate)*(5.0/1023.0);
Serial.println(ads,5);
if (i>127){
  break;
}
}
 Serial.println("END TRANSACTION");
SPI.endTransaction();
digitalWrite(cspin, HIGH);
exit(0);
}
```

- X9C103P

```
#include <X9C.h>
// ------------- PINS X9C ------------------ //
#define UD  12
#define INC 13
#define CS  11
X9C pot;
// ----------------------------------------------------//
int g = A11;
int s = A1;
int a = A2;
int offset = A0;
int ads = A3;
void setup() {
Serial.begin(9600);
pot.begin (CS, INC, UD);
pot.setPot(50, true); // define o potenciometro para metade do whiper
delay(500);
Serial.println("Time;Vg;Vs;Va;Voffset;Vads;/n"); // /n para nao ficar "," e nao causar incoerencias com o script
delay(1000);
        }
```

```
void loop() {
for (int z=0;z<10;z++){ //number of cycles
pot.setPot(50, true); // define o potenciometro para metade do whiper
delay(500);
        for (int i=0 ; i<25; i++){ //number of steps
        float t=millis();
        pot.trimPot(1, X9C_DOWN, true);
        float R = analogRead(a)*q;
        Serial.println(R,5);
        delay (50);

            for (int x=0; x<25; x++){ //numero de dados por step
            unsigned long t=millis();
            int Vg = analogRead(g);
            int Vs = analogRead(s);
            int Va = analogRead(a);
            int Voffset = analogRead(offset);
            int Vads = analogRead(ads);
            unsigned long myArray[6] = {t, Vg, Vs, Va, Voffset, Vads};

                                    for (int i = 0; i <= 5; i = i+1) {
                                    Serial.print(myArray[i]);
                                    Serial.print(";");
     }
        Serial.println("/n");
        delay(60);
        }
        }
        }
        Serial.println("fim");
        exit(0);}
```

Annex C : **Simulation schematic and results**

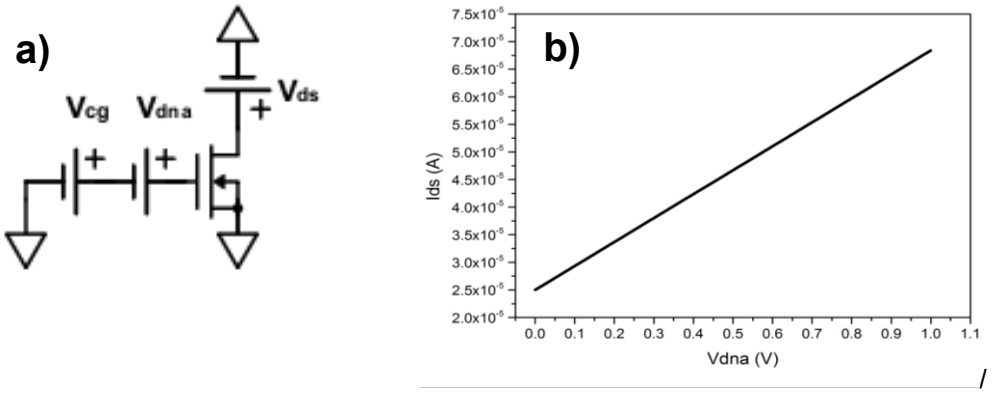

Fig. 6.3 -a) V_{dna} simulation schematic where V_{cg} is maintend at constant 1.7V and Vds at 0.05V and b) shows I_{ds} in function of V_{dna} 1 mV step, from 0 to 1V.

Annex D : **MCP4141 connections to the Arduino**

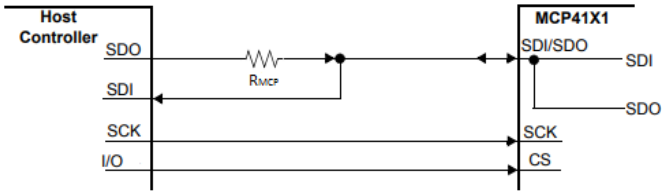

Fig. 6.4 - MCP4141 SPI master-slave connections. Adapted from [32]

Annex E : **Firmware and software considerations**

- Programming MCP4141 measurements

To perform tests using the MCP4141 digital potentiometer the communication is done using the Serial Peripheral Interface (SPI) which is a communication protocol developed by Motorola and now Freescale, which uses a master-slave architecture where the microcontroller modulates the writing and the reading signal [55]. The required connections to communicate with the microcontroller are presented in Annex F.

In the Arduino board, SDO (serial data out) and SDI (serial data in) pins show with a different nomenclature where the SDO is the MOSI (master-out slave-in) and SDI is the MISO (master-in slave-out) pin. A 3kΩ resistor is connected between the two microcontroller outputs. The CS (chip select) pin is assigned to a PWM pin 10 becoming possible to begin and end transactions. To do so, pin PWM 10 is assigned as an output. After starting the SPI communication protocol, CS is set to low and delay is added to ensure the timing requirements for valid data. The device only needs 70 ns to set the CS to low however, 100 ms delay was used.

SPI settings were defined according to SPISettings(speedMaximum, dataOrder, dataMode). Where the speed maximum is SPI chip maximum supported rate of data. The data order tells the chip which bit goes first and has two modes. Most significant bit first (MSBFIRST) or least significant bit first (LSBFIRST). The dataMode sets the clock polarity and phase [56]. THE SPI settings used for communicating with the MCP4141 were: SPISettings(2000000, MSBFIRST, SPI_MODE0).

According to the device datasheet, the maximum SPI input frequency is 10 MHz and there's no minimum limit defined when powering the device from 2.7V to 5.5V. By setting the speedMaximum to 2MHz it's assured it's within the working range of the device. Also, data should be sent the most significant bit first and the SPI mode can be either 0 or 3. SPI mode 0 was chosen.

The MCP is a 7-bit digital potentiometer with 128 wiper positions that are converted to binary. So, for position 0, the microcontroller converts each number of the wiper position to binary. As an example, to address wiper position 50 the MCP receives information as 0110010 and for wiper position 127, it receives 1111111.

Bits are transferred using the function SPI.transfer that converts the decimal to a binary number and sends the data to the digital potentiometer. A for loop was done from 0 to 127 to make sure all the wiper positions were covered and analog-read functions were used to obtain data from the input V_{cg} and output V_{ads}. When the loop ends, the program stops introducing a break.

To perform V_{adc} measurements over time using the MCP4141, a value for a wiper position is fixed and V_{adc} is monitored over a defined time. The Arduino software used to monitor the V_{adc} as a function of all the wiper positions of the MCP4141 is present in Annex C.

- Programming X9C103P measurements

Communicating with the X9C103 becomes facilitated by the use of libraries where functions to address wiper positions are already developed. In Annex D it's represented how to control the potentiometer by controlling the CS, INC, and U/D X9C103P pins.

First, these pins were addressed to PWM pins that allow setting values as high or low (1 or 0) to ensure communication with the device.

Next, the first aspect to be taken into account is that there are key signals that are necessary to register in order to assure that the circuit is working properly and signals that need to be tuned to fit the requirements. First, it is mandatory that the TIA OA is working. To do so, when V_a and V_{offset} have the same value, the OA is working as there is practically no offset between the two voltages as defined by the manufacturer. If these signals do not display the same value, the circuit won't work since it won't be able to amply any input signal. Then, $V_{s,}$ to define the V_{ds} potential and to make sure that V_s is always below V_{offset}. Otherwise, the nMOS won't form a channel and won't be any current flowing from the drain to the source. Finally, V_g and V_{adc} as input and output of the circuit that allows performing the required

measurements. So, 5 analog signals V_g, V_s, V_a, V_{offset}, and Vads were assigned to 5 analog pins where the signal is acquired by the Arduino board.

At the beginning of the program, the wiper position is always set to 50 to activate the potentiometer and a delay of 500 ms is added to make sure the transition is made and also to guarantee that every time the program runs, doesn't start on the last wiper position since the IC stores the last command. Next, a line is printed with the title: "Time;Vg;Vs;Va;Voffset;Vadc;/n" to organize the data. Printing "/n" at the end of each line and add ";" between the variables are factors that were added concerning compatibility with python language and .csv files.

Next, a for-loop is done containing 2 more for-loops that defines how many data points per step and how many steps the digital potentiometer should take. Every time the microcontroller obtains the defined number of data points, the wiper position drops one value and repeats the data acquisition loop. When these two for loops (potentiometer steps and datapoints per step) are finished, one cycle is complete. When the defined number of cycles is complete, the program stops acquiring data.

For constant V_{cg} measurements, the required wiper position is and the time of measurement are defined. After the defined time ends, the program stops.

- ardFile.py

ardFile.py generates an Arduino file based on previously developed Arduino file (copy.ino) created to communicate with X9C103P altering only the user-defined parameters concerning cycles, steps and datapoints per step.

In the current folder where the ardFile.py is, the script checks whether there's a folder called RunMe. If the folder doesn't exist it creates it. If it exists it's deleted in order to create the new Arduino file. There's room for optimization in this function since it always deletes the previous file. In order to save the files from previous data, it's suggested to move the generated file to another folder so it doesn't get lost.

Next, the program opens the reference Arduino file (copy.ino) and alters the values in the for loops so it performs the measurements as the user requires.

After setting the parameters, click submit for every parameter and file is created. Hitting enter doesn't work to set the parameters. It's important to note that the wiper position of the digital potentiometer is set to 50 so, the maximum Vg sweep steps usable in this setup is 25. The initial potentiometer position can be altered in the copy.ino file. Also important to note that this is the reference file and therefore must never be deleted. Full code and user interface can be consulted next.

```
import sys, os, threading, serial, time
import pandas as pd
import math
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np
import tkinter
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
import shutil
path = os.getcwd()+"\\RunMe"
if os.path.isdir(path) is True:
    shutil.rmtree(path)
os.mkdir(path)

main_win = tkinter.Tk()
main_win.title("Select Vg sweep parameters")
main_win.geometry("500x500")

def cyclesnumber():
    global cycles
    cycles = cyclesentry.get()
    return cycles
def stepsnumber():
    global steps
    steps = stepsentry.get()
    return steps
def daterstepnumber():
    global datapstep
    datapstep = dataentry.get()
    return datapstep
        def done():
            main_win.destroy()
        #Labels
        Label (main_win, text='How many cycles').grid(row=1, column=0)
        Label (main_win, text='How many Vg sweep steps').grid(row=2,
column=0)
        Label (main_win, text='How many datapoints per step').grid(row=3,
column=0)
        Label (main_win, text='1- UPLOAD RunMe.ino to the Arduino
board').grid(row=5,column=0)
        Label (main_win, text='2- Run ardtoCsv.py for data
acquisition').grid(row=6,column=0)
        Label (main_win, text='3- mod.py for data
treatment').grid(row=7,column=0)
        #Text entries
        cyclesentry= Entry(main_win, width=5)
        cyclesentry.grid(row=1,column=1)

        stepsentry=Entry(main_win,width=5)
        stepsentry.grid(row=2, column=1)

        dataentry=Entry(main_win, width=5)
        dataentry.grid(row=3,column=1)
        # buttons
        button_cycles=Button(main_win, text='Submit', width= 10,
command=cyclesnumber).grid(row=1, column=2)
        button_steps=Button(main_win, text='Submit', width= 10,
command=stepsnumber).grid(row=2, column=2)
        button_data=Button(main_win, text='Submit', width=10,
command=daterstepnumber).grid(row=3, column=2)
        button_done=Button(main_win,text='Generate Arduino file',width=30,
command=done).grid(row=4,column=0)
```

```
main_win.mainloop()
cyclesStr = str("z<"+cycles)
stepsStr= str("i<"+steps)
dataStr=str("x<"+datapstep)
with open("copy.ino", "r") as f,\
    open(os.getcwd()+"\\RunMe\\RunMe.ino", "w") as outfile:
   lines=f.read()
   lines=lines.replace("z<NC",cyclesStr)
   lines=lines.replace("i<NS",stepsStr)
   lines=lines.replace("x<NDS",dataStr)

   outfile.write(lines)
```

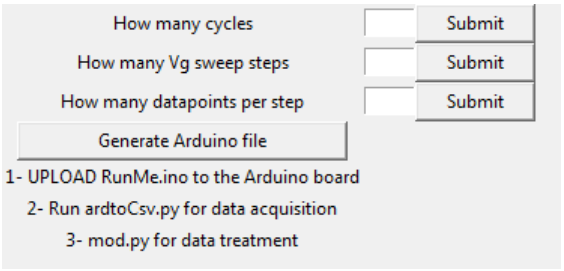

Fig. 6.5 - Software user interface for creating the arduino file

- ArdtoCsv.py

This script retrieves the data that is acquired from the Arduino and creates a .csv file. Prior to run this script there is 1 parameter that changes from user to user: the COM port where the Arduino is connected. For Windows users, this can be consulted in the Device Manager>Ports. In the main application section of the script, there's a variable COMPORT. This variable should be changed to the port that it's being used by the Arduino board. If the selected port is not the correct one, it won't be able to acquire data and the script won't work.

After this variable is set, the script creates a .txt file writing all the data that is being retrieved by the microcontroller and keeps writing until it finds a line where ''fim" is written. Next, checks if the file has any strange characters in the file that were not supposed to be there. In the first trials, Arduino inserted characters in the first line that altered the organization of the columns and caused inconsistency with the program. As a result, a check() function was developed to delete any character that is not supposed to be in the first line and print the desired one. After it opens the .txt file converts it to .csv and process all the values by converting the discrete values (0 to 1024) to voltage values by multiplying the column by $\frac{5}{1023}$. Finally, the .txt auxiliary file is deleted and a file named based on the time that the measurement was taken plus DATA is created (e.g: 20190730-192612DATA.csv) in the folder where the script is. The commented code is presented next.

```
import sys, os, threading, serial, time
import pandas as pd
import math
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np
def close_file():
  text_file = open(timestr+".txt", "w+")
  text_file.close()

timestr = time.strftime("%Y%m%d-%H%M%S")
def monitor():

  ser = serial.Serial(COMPORT, BAUDRATE, timeout=0)
  valor ="
  timestr = time.strftime("%Y%m%d-%H%M%S")
  text_file = open(timestr+'.txt', "w+")



  while (1):
    tempo_medida = time.perf_counter()
    text_file = open(timestr+'.txt', "a+")
    line = ser.readline()

    if (line != ""):
      valor = str(line).split("b'")

      # write to file
      if(valor[1] and valor[1]!="'"):
        text_file.write(valor[1]+"\n")
        text_file.close()
```

```
        if bytes("fim", 'utf-8') in line: #definir o tempo de mediçao em
segundos - atualizar tb para funçao que se a serial para, o programa tb para
          break

      print ("Stop Monitoring after "+ str (tempo_medida)+" seconds")

    def check(): ## fUNÇAO PARA LIMPAR A PRIMEIRA CASO A
SERIAL DO ARDUINO EMITA CARACTER ESPECIAL QUE IMPOSSIBILITA
A LEITURA DO FICHEIRO
      with open(timestr+".txt", 'r') as fin:
        line = fin.readline()
        data= fin.read()

      if ("Time;Vg;Vs;Va;Voffset;Vads" in line):
       exit
      else:
       data = data.replace(line,"Time;Vg;Vs;Va;Voffset;Vads")

       print("CORRIGIDO")
       with open(timestr+".txt", 'w+') as fout:
        fout.write(data)

    def clean():
      with open(timestr+".txt", "r") as infile,\
          open(timestr+".csv", "w") as outfile:
        data = infile.read()
        data = data.replace("'", "")
        data = data.replace("\\r\\n","")
        data = data.replace(";/n","")
```

```
        data = data.replace("-","")
        outfile.write(data)

    df = pd.read_csv(timestr+".csv",sep=';')
    df.loc[: ,['Vg']] = df.loc[: ,['Vg']]*(5/1023)
    df.loc[: ,['Vs']] = df.loc[: ,['Vs']]*(5/1023)
    df.loc[: ,['Va']] = df.loc[: ,['Va']]*(5/1023)
    df.loc[: ,['Voffset']] = df.loc[: ,['Voffset']]*(5/1023)
    df.loc[: ,['Vads']] = df.loc[: ,['Vads']]*(5/1023)
    df.to_csv(timestr+"DATA.csv", sep=';', index=False)
    os.remove(timestr+".csv")
    os.remove(timestr+".txt")
  def Vads():
    df = pd.read_csv(timestr+".csv")
    df.loc[: ,['Vads']]  \
    =df.loc[: ,['Vads']]*(5/1023)
    df.loc[:,['Time']] = df.loc[:,['Time']]/1000 ### tempo em ms to s
    t = df['Time']
    s = df['Vads']
    return s
""" ---------- MAIN APPLICATION----------"""
print ("Start Serial Monitor")
print ("Uploading data")
COMPORT = 'COM3'          #define port
BAUDRATE = 9600           #define arduino BAUD rate
monitor()
check()
clean()
```

- Mod.py

This script was developed to make the data treatment easier. It has some functionalities like data averaging, plot all the obtained curves, plot only specific curves, plot data at a constant V_{adc}, plot data at a constant V_{gs}, plot output error as a function of V_{gs} and a rudimental threshold voltage calculator.

All the code is presented below.

```
import tkinter
from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
import matplotlib.pyplot as plt
from scipy.fftpack import fft
import numpy as np
import pandas as pd
import time
from sklearn.linear_model import LinearRegression as lr
timestr = time.strftime("%Y%m%d-%H%M%S")
main_win = tkinter.Tk()
main_win.title("Select data")
main_win.geometry("1000x500")
main_win.sourceFile = ''



def get():
    global s
    s = stepsentry.get()
    return s

def getthat():
    global d
    d=datapointsentry.get()
    return d


def chooseFile():
    main_win.sourceFile =
filedialog.askopenfilename(parent=main_win, initialdir= "/", title='Select data
file')

b_chooseFile = tkinter.Button(main_win, text = "Chose File", width =
10, height = 2, command = chooseFile)
b_chooseFile.grid(row=1, column=0)

def done():
    main_win.destroy()

#Labels
Label (main_win, text='How many Vg sweep steps').grid(row=2,
column=0)
Label (main_win, text='How many datapoints per step').grid(row=3,
column=0)

#Text entries
stepsentry= Entry(main_win, width=5)
stepsentry.grid(row=2,column=1)

datapointsentry=Entry(main_win,width=5)
datapointsentry.grid(row=3, column=1)

# buttons
button_1=Button(main_win, text='Submit', width= 10,
command=get).grid(row=2, column=2)
button_2=Button(main_win, text='Submit', width= 10,
command=getthat).grid(row=3, column=2)
done_button=Button(main_win, text='Done', width=10,
command=done).grid(row=4, column=0)

main_win.mainloop()

#-------------------------READ SELECTED DATA----------------------#

df = pd.read_csv(main_win.sourceFile, sep=";")
df.drop(df.tail(1).index,inplace=True)
df['Vads'] = pd.to_numeric(df['Vads'],errors='coerce')
df['Time'] = pd.to_numeric(df['Time'],errors='coerce')

l = len(df)
d=int(d)
s=int(s)
pointspercurve = d*s # number of points per curve
curves=(l/pointspercurve) #number of curves

#---------------------------------CURVE SEPARATOR-------------------------#

a = np.split(df,curves) #splits the dataframe for each cycle

def curva(numerodacurva):
    return a[numerodacurva]

#-------------------------SINGLE CURVE AVERAGING-------------------#

def averagecurve(x, param):
    vadc = curva(x)['Vads']
    vg = curva(x)['Vg']
    time = (curva(x)['Time'])/1000
    vs = curva(x)['Vs']
    vgs= vg - vs
    avgadc = [sum(vadc[i:i+d])/d for i in range(0,len(vadc),d)]
    avgtime = [sum(time[i:i+d])/d for i in range(0,len(time),d)]
    avgVg = [sum(vg[i:i+d])/d for i in range(0,len(vg),d)]
    avgVs = [sum(vs[i:i+d])/d for i in range(0,len(vs),d)]
    avgVgs = [sum(vgs[i:i+d])/d for i in range (0, len(vgs),d)]

    errorAdc = [np.std(vadc[i:i+d]) for i in range(0,len(vadc),d)]
    errorVg = [np.std(vg[i:i+d]) for i in range(0,len(vg),d)]
    errorVs = [np.std(vs[i:i+d]) for i in range (0, len(vs),d)]

    dfaveragetime =
    pd.DataFrame(np.column_stack([avgtime,avgVs,avgVgs,avgadc,
    errorAdc, errorVg, errorVs]),columns=['Time','Average Vs','Average
    Vgs','Average Vadc','Vadc standard deviation','Vg standard
    deviation','Error Vs'])
    dfaverageVg =
    pd.DataFrame(np.column_stack([avgVg,avgVs,avgVgs, avgadc,
    errorAdc, errorVg, errorVs]),columns=['Average Vg','Average
    Vs','Average Vgs','Average Vadc','Vadc standard deviation','Vg
    standard deviation','Error Vs'])

    if (param == 'time'):
        return dfaveragetime
    if (param == 'vg'):
        return dfaverageVg


#-------------All curve averaging and Vadc normalization-----------#
allcurveAvg=[]
for n in range (0, int (curves) ,1):
    ndf=averagecurve(n,'vg') #Vg
    allcurveAvg.append(ndf)
    maxval=allcurveAvg[n].loc[:,['Average Vadc']].max()
    normalize=allcurveAvg[n].loc[:,['Average Vadc']]/maxval
    ndf['Normalized Vadc']=normalize

allcurveAvgtime=[]
for n in range (0, int (curves) ,1):
    ndftime=averagecurve(n,'time') #TIME
    allcurveAvgtime.append(ndftime)
    maxval=allcurveAvgtime[n].loc[:,['Average Vadc']].max()
    normalize=allcurveAvgtime[n].loc[:,['Average Vadc']]/maxval
```

```
        ndftime['Normalized Vadc']=normalize
    #-------------------------------------------------------------------------------#
    def vg(x):
        return allcurveAvg[x].loc[:,['Average Vg']]

    def adc(x):
        return allcurveAvg[x].loc[:,['Average Vadc']]

    def stdErrorVadc(x):
        return allcurveAvg[x].loc[:,['Vadc standard deviation']]

    def stdErrorVg(x):
        return allcurveAvg[x].loc[:,['Vg standard deviation']]

    def normalized(x):
        return allcurveAvg[x].loc[:,['Normalized Vadc']]

    def vgs(x):
        return allcurveAvg[x].loc[:,['Average Vgs']]

    #-------------------------------------PLOT EACH CURVE----------------------#

    def grafico(*nums):
        def func(*nums):
            return nums

        a=func(*nums)

        stringx=[]
        stringy=[]
        stringe=[]
        for x in a:
            txt=vg(x)
            txt1=adc(x)
            txt2=stdErrorVadc(x)
            stringx.append(txt)
            stringy.append(txt1)
            stringe.append(txt2)
            valueX=np.concatenate(stringx)
            valueY=np.concatenate(stringy)
            valueE=np.concatenate(stringe)


        ty=np.hstack((valueX,valueY,valueE))


        fig1, ax = plt.subplots()
        ax.errorbar(ty[:,0],ty[:,1],ty[:,2],linestyle='None',  marker='.',
ecolor='red',elinewidth=1, capsize=3)
        ax.set(xlabel='Vg (V)', ylabel='Vadc (V)', title='Response')
        ax.grid()
        plt.show()

    #-----------------------------------ALL CURVES PLOT--------------------------#
    def plotAllCurves():
        o=np.concatenate(allcurveAvg)
        fig2, ax = plt.subplots()
        ax.errorbar(o[:,2],  o[:,3],  o[:,4],  linestyle='None',  marker='.',
ecolor='red',elinewidth=1, capsize=3)
        ax.set(xlabel='Vgs (V)', ylabel='Vadc (V)', title='Response')
        ax.grid()
        plt.show()
    #---------------------------------------PLOT ERROR---------------------------#
    def plotError():
        o=np.concatenate(allcurveAvg)
        fig, ax = plt.subplots()
        ax.scatter(o[:,2], o[:,4], s=1,cmap='winter')
        ax.set(xlabel='Vgs (V)', ylabel='Vadc Error (V)', title='')
        ax.grid()
        plt.show()

    #-----------------------------------PLOT Vs----------------------------------#
    def plotVs():

        o=np.concatenate(allcurveAvgtime)
        fig2, ax = plt.subplots()
        ax.scatter(o[:,0], o[:,1], s=1,cmap='winter')
        ax.set(xlabel='Time (s)', ylabel='Vs(V)', title='Response')
        ax.grid()
        plt.show()

    #----------------PLOT NORMALIZED CURVES----------------#
    def plotAllnormCurves():
        o=np.concatenate(allcurveAvg)
        fig2, ax = plt.subplots()
        ax.scatter(o[:,2], o[:,7], s=1,cmap='winter')
        ax.set(xlabel='Vgs (V)', ylabel='Normalized values', title='Response')
        ax.grid()
        plt.show()
    #----------------------PLOT EACH NORMALIZED CURVE------------#
    def graficoNORMAL(*nums):
        def func(*nums):
            return nums

        a=func(*nums)

        stringx=[]
        stringy=[]
        for x in a:
            txt=vg(x)
            txt1=normalized(x)
            stringx.append(txt)
            stringy.append(txt1)
            valueX=np.concatenate(stringx)
            valueY=np.concatenate(stringy)

        fig3, ax = plt.subplots()
        ax.scatter(valueX,valueY, s=2)
        ax.set(xlabel='Vg (V)', ylabel='Normalized Value', title='Response')
        ax.grid()

        timef = curva(x)['Time']
        avgtimef = [sum(timef[i:i+d])/d for i in range(0,len(timef),d)]
        label = str (avgtimef[-1])+' ms'
        plt.text(2.5,3,label)
        plt.show()

    #-------------------------------PLOT CONSTANT Vg--------------------#
    def VgCTE(*nums):
        def func(*nums):
            return nums

        a=func(*nums)

        vgctexvalue=[]
        vgcteyvalue=[]
        vgcteevalue=[]
        for curveregion in a:

            for n in range (0, int (curves), 1):
                xvalue = int(n)
                yvalue = float(adc(n).iloc[curveregion])
                evalue = float(stdErrorVadc(n).iloc[curveregion])
                vgctexvalue.append(xvalue)
                vgcteyvalue.append(yvalue)
                vgcteevalue.append(evalue)


vgcteavg=pd.DataFrame(np.column_stack([vgctexvalue,vgcteyvalue,vgcteeval
ue]), columns=['Cycle','Vadc','erro'])

        directory=tkinter.filedialog.askdirectory()
        filestr=("/"+timestr+"AVGcurves.csv")
        savestr=str(directory+filestr)
        print(vgcteavg)
        vgcteavg.to_csv(savestr)

        fig3, ax = plt.subplots()
        ax.errorbar(vgctexvalue,vgcteyvalue,vgcteevalue,  linestyle='None',
marker='.', ecolor='red',elinewidth=1, capsize=3)
        ax.set(xlabel='Cycle', ylabel='Vadc(V)', title='Response')
        ax.grid()

        plt.show()
    #-------------------------------PLOT CONSTANT VADC-----------------------#
    def VadcCTE(*nums):
        def func(*nums):
            return nums

        a=func(*nums)

        vadctexvalue=[]
        vadcteyvalue=[]
        vadcteevalue=[]
        for curveregion in a:

            for n in range (0, int (curves), 1):
                xvalue = int(n)
                yvalue = float(vgs(n).iloc[curveregion])
                evalue = float(stdErrorVg(n).iloc[curveregion])
                vadctexvalue.append(xvalue)
                vadcteyvalue.append(yvalue)
                vadcteevalue.append(evalue)


vadcteavg=pd.DataFrame(np.column_stack([vadctexvalue,vadcteyvalue,vadct
eevalue]), columns=['Cycle','Vg','erro'])

        directory=tkinter.filedialog.askdirectory()
        filestr=("/"+timestr+"AVGcurves.csv")
        savestr=str(directory+filestr)
        print(vadcteavg)
        vadcteavg.to_csv(savestr)

        fig3, ax = plt.subplots()
        ax.errorbar(vadctexvalue,vadcteyvalue,  vadcteevalue,
linestyle='None', marker='.', ecolor='red',elinewidth=1, capsize=3)
        ax.set(xlabel='Cycle', ylabel='Vg(V)', title='Response')
        ax.grid()

        plt.show()

    #---------------------- Linear Regression -------------------------------#
    def linreg(curva,min,max):
        x=vg(curva).loc[min:max,:]
        y=adc(curva).loc[min:max,:]
        model=lr().fit(x,y)

        r_sq = model.score(x, y)
        intercept= float (model.intercept_)
```

```
            slope= float(model.coef_)
            xintercept=float (-intercept/slope)

            xmodel=np.linspace(xintercept+0.1,vg(curva).loc[max,:])
            ymodel=(slope*xmodel)+intercept

            label=                                                    str("Cycle
"+str(curva)+"\n"+"m="+str(slope)+"\n"+"b="+str(intercept)+"\n"+"r2="+str(r_sq)
+"\n"+"x axis intercept = "+str(xintercept))
            plt.plot (xmodel,ymodel, '-r', label = label )
            plt.scatter (vg(curva),adc(curva))
            plt.legend(loc='upper left')
            plt.xlabel('Vg(V)')
            plt.ylabel('Vadc(V)')
            plt.show()

        #-------------------------Save Averaged curves-----------------------------#
        def saveAvgCurves():

            directory=tkinter.filedialog.askdirectory()
            filestr=("/"+timestr+"AVGcurves.csv")
            savestr=str(directory+filestr)
            dfallcurveAvg=pd.DataFrame(allcurveAvg)
            dfallcurveAvg.to_csv(savestr)

        #----------------------------------Plot window---------------------------#

        plot_win = tkinter.Tk()
        plot_win.title("Process the data for me")
        plot_win.geometry("1000x500")

        def get_curveregion():
            global curveregion
            curveregion = regionentry.get()
            return


        def get_curves():
            global response
            response = curvesentry.get()
            return

        def graph():
           global response
           response=str(response)
           response=response.split(',')
           response=list(map(int,response))
           response=tuple(response)
           grafico(*response)
           return

        def VadcCte():
            global curveregion
            curveregion=str(curveregion)
            curveregion=curveregion.split(',')
            curveregion=list(map(int,curveregion))
            curveregion=tuple(curveregion)
            VadcCTE(*curveregion)
            return

        def VgCte():
            global curveregion
            curveregion=str(curveregion)
            curveregion=curveregion.split(',')
            curveregion=list(map(int,curveregion))
            curveregion=tuple(curveregion)
            VgCTE(*curveregion)
            return

        def plotnormalizedCurves():
           global response
           response=str(response)
           response=response.split(',')
           response=list(map(int,response))
           response=tuple(response)
           graficoNORMAL(*response)
           return

        #-----------------Linear Reg functions----------------#
        def get_curva():
            global linregcurva
            linregcurva = linregCurveEntry.get()
            linregcurva=int(linregcurva)
            return

        def get_From():
            global linregfrom
            linregfrom = linregFromEntry.get()
            linregfrom=int(linregfrom)
            return

        def get_To():
            global linregto
            linregto = linregToEntry.get()
            linregto=int(linregto)
            return

        def linregplot():
            linreg(linregcurva,linregfrom,linregto)
            return

            #Labels
            Label (plot_win, text='Which curves you want to plot?').grid(row=2,
column=0)
            Label (plot_win, text='What is the curve region/regions you want to plot
for cte. Vg?').grid(row=3, column=0)
            Label (plot_win, text='Total number of curves =').grid(row=1, column=0)
            Label (plot_win, text=curves).grid(row=1,column=1)

            Label (plot_win, text= "Linear regression").grid(row=8, column=1)
            Label (plot_win, text= "Curve number:").grid(row=9, column=0)
            Label (plot_win, text= "From:").grid(row=10, column=0)
            Label (plot_win, text= "To:").grid(row=11, column=0)
            #Text entries
            curvesentry= Entry(plot_win, width=50)
            curvesentry.grid(row=2,column=1)

            regionentry=Entry(plot_win,width=50)
            regionentry.grid(row=3, column=1)

            linregCurveEntry=Entry(plot_win, width=20)
            linregCurveEntry.grid(row=9,column=1)

            linregFromEntry=Entry(plot_win, width=20)
            linregFromEntry.grid(row=10,column=1)

            linregToEntry=Entry(plot_win, width=20)
            linregToEntry.grid(row=11,column=1)

            # buttons
            button_PlotAll=Button(plot_win,                text='Plot                all',
command=plotAllCurves).grid(row=4, column=0)
            button_PlotAllnormalizedCurves=Button(plot_win,      text='Plot      all
normalized curves', command=plotAllnormCurves).grid(row=5,column=0)
            button_grafico=Button(plot_win,                text='Plot                curves',
command=graph).grid(row=6,column=0)
            button_Plotnormalized=Button(plot_win, text='Plot normalized curves',
command=plotnormalizedCurves).grid(row=7, column=0)

            button_plotError=Button(plot_win,  text  =  'Plot  output  error',
command=plotError).grid(row=5, column=2)
            button_plotVs=Button(plot_win,           text           ='Plot           Vs',
command=plotVs).grid(row=6, column=2)

            button_vadCte=Button(plot_win, text='Plot curves for constant Vadc',
command=VadcCte).grid(row=4, column=1)
            button_vgCte=Button(plot_win,  text='Plot  curves  for  constant  Vg',
command=VgCte).grid(row=5, column=1)

            buttonget_curves=Button(plot_win,                          text="Submit",
command=get_curves).grid(row=2,column=2)
            buttonget_curveregion=Button(plot_win,                     text="Submit",
command=get_curveregion).grid(row=3,column=2)
            button_SaveAvgCurves=Button(plot_win, text='Save averaged curves
data', command=saveAvgCurves).grid(row=4,column=2)

            buttonget_curva=Button(plot_win,                           text="Submit",
command=get_curva).grid(row=9,column=2)
            buttonget_from=Button(plot_win,                            text="Submit",
command=get_From).grid(row=10,column=2)
            buttonget_to=Button(plot_win,                              text="Submit",
command=get_To).grid(row=11,column=2)
            buttonget_linregPlot=Button(plot_win,                        text="Plot",
command=linregplot).grid(row=12,column=1)

            plot_win.mainloop()
```

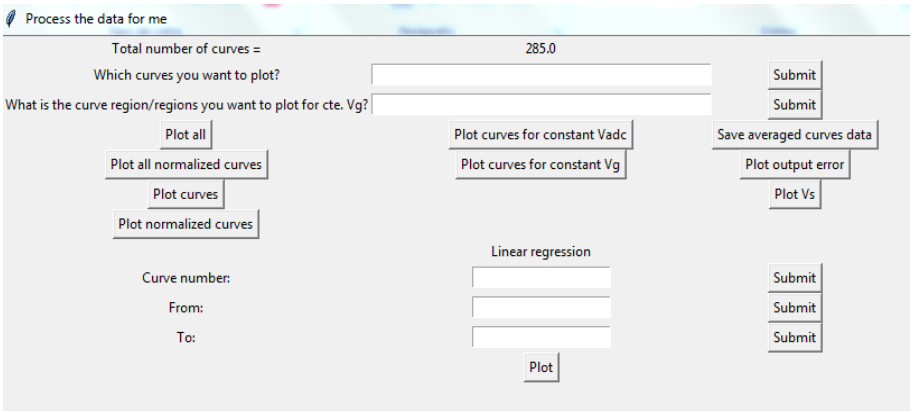

Fig. 6.6 – mod.py data-treatment interface

Annex F : **Circuit characterization, sensor characteristics and input waveform**

This Annex presents some results of the circuit output, sensor characteristics, prototype response to pH, sensor degradation and input waveform for LAMP monitoring.

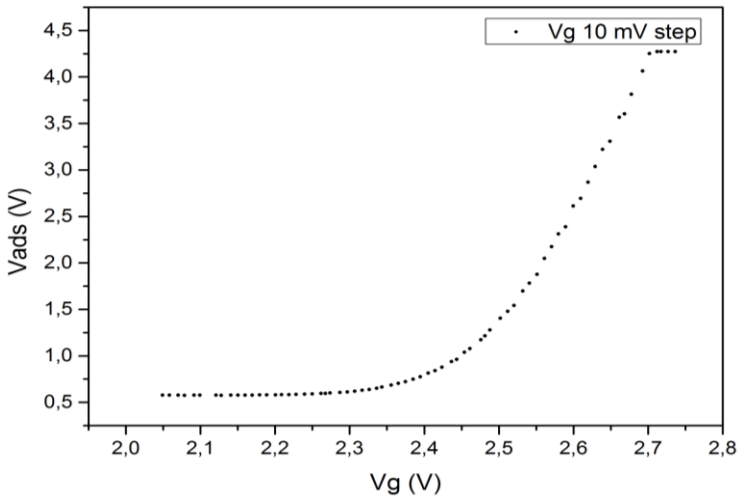

Fig. 6.7 - Vadc output in function of Vg 10 mV sweep showing capabilities of identifying small potential change at the circuit input

Measured Capacity (pF)

	Sensing electrode radius (mm) 1mm						Sensing electrode radius (mm) 2mm						Sensing electrode radius (mm) 4mm					
	Thickness (nm)						Thickness (nm)						Thickness (nm)					
Device thickness	300	σx̄	400	σx̄	500	σx̄	300	σx̄	400	σx̄	500	σx̄	300	σx̄	400	σx̄	500	σx̄
Control electrode overlap (%)																		
5	106.5	21.9	87.3	10.7	55.3	14.6	147.7	14.8	106.5	3.5	61.3	11.7			150.7	14.0	69.3	15.0
25	215.5	6.4	141.5	27.9	121.0	10.0	239.7	11.7	184.0	13.5	141.3	2.9	107.0	99.0	188.3	78.3	194.0	15.0
50	246.5	4.9	156.5	4.9	144.3	20.0	215.5	3.5	214.0	4.2	192.7	27.1			211.0	-	240.0	39.6
75			209.7	7.6	162.7	24.2	214.0	48.1	250.5	2.1	209.3	15.3	140.5	16.3	242.3	82.2	323.3	22.1
100			270.5	6.4	197.7	30.7	130.0	28.3	290.5	14.8	271.0	22.6			408.0	31.8	375.0	36.4

Fig. 6.8 – Reference values of the structural properties of the previously produced sensitive layer of the sensors

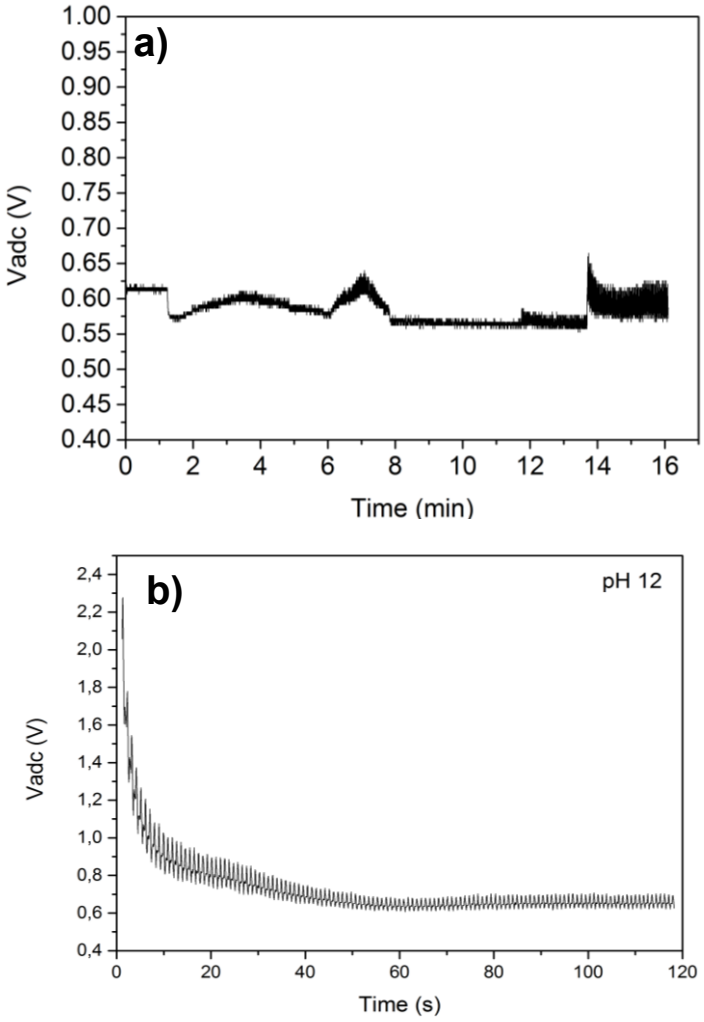

Fig. 6.9 – Constant V_{cg} = 2.25 V with 12, 10, 9 and 4 pH solutions showing no meaningful response and b) another measurement showing the output drop to the lowest limit of the dynamic range using a pH 12 solution. Results obtained using the MCP4141 potentiometer.

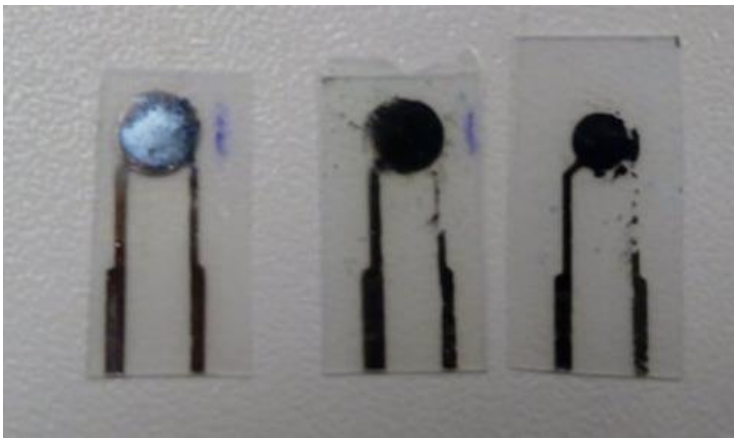

Fig. 6.10 –The sensor's molybdenum contacts and the sensitive layer of Ta_2O_5 show signs of degradation after performing pH measurements.

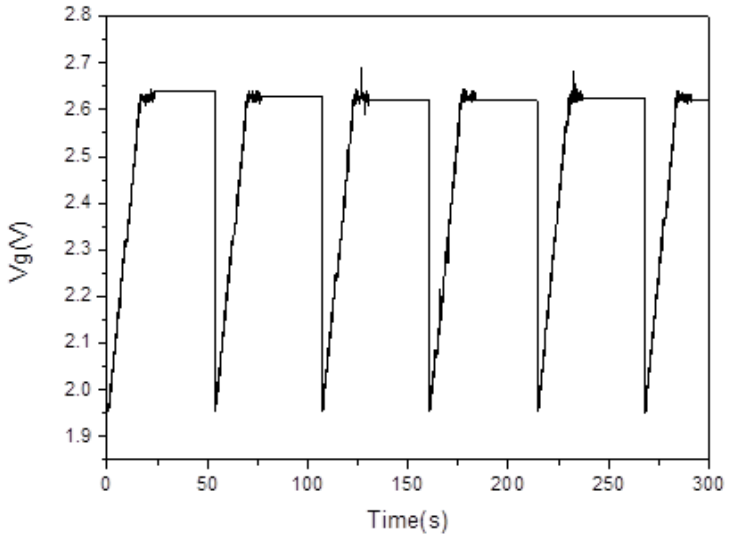

Fig. 6.11 - Input signal on the sensors over time for LAMP monitoring

Annex G **: PCB V1.0 schematic and layout**

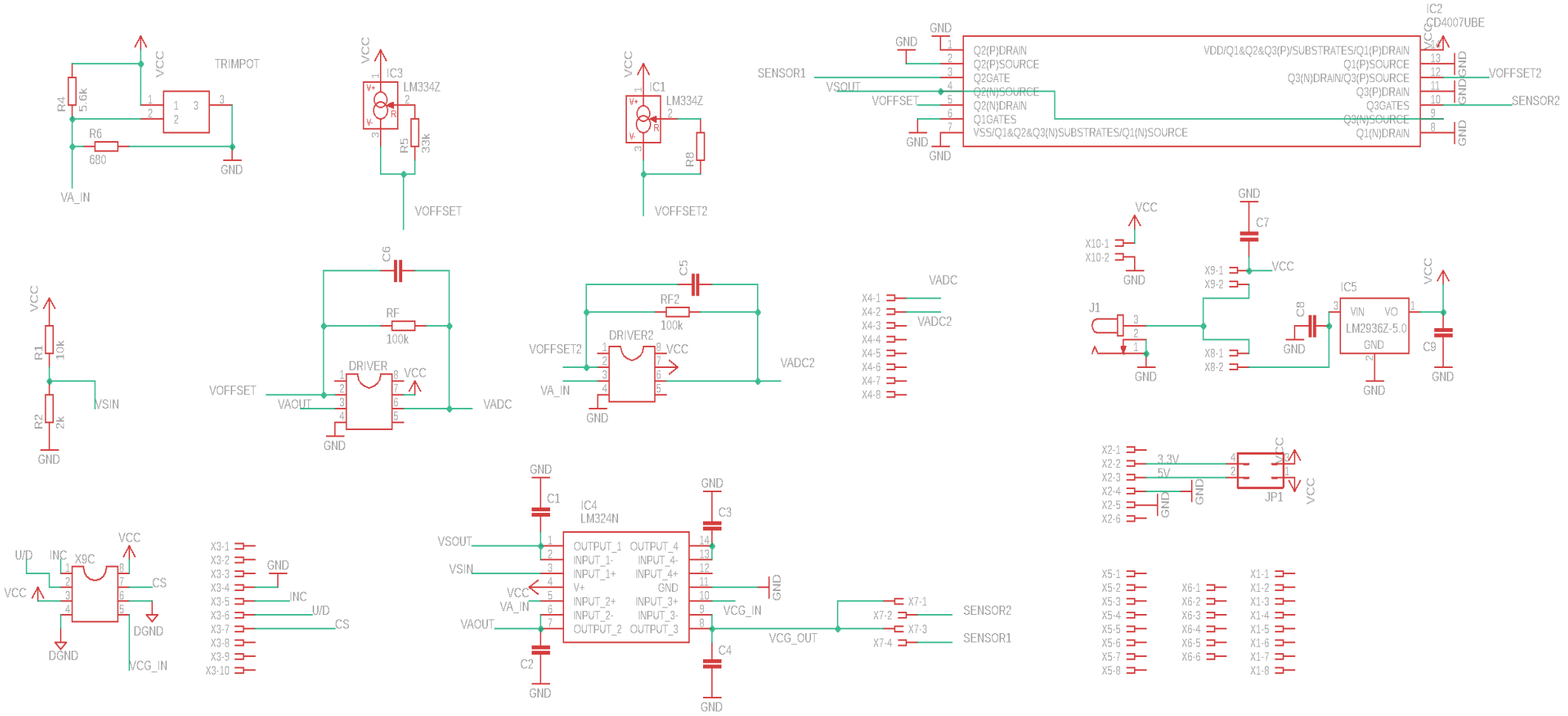

Fig. 6.12 – Corrected schematic used to develop the V1.0 PCB layout, showing all the components and required connections for the AFE circuit

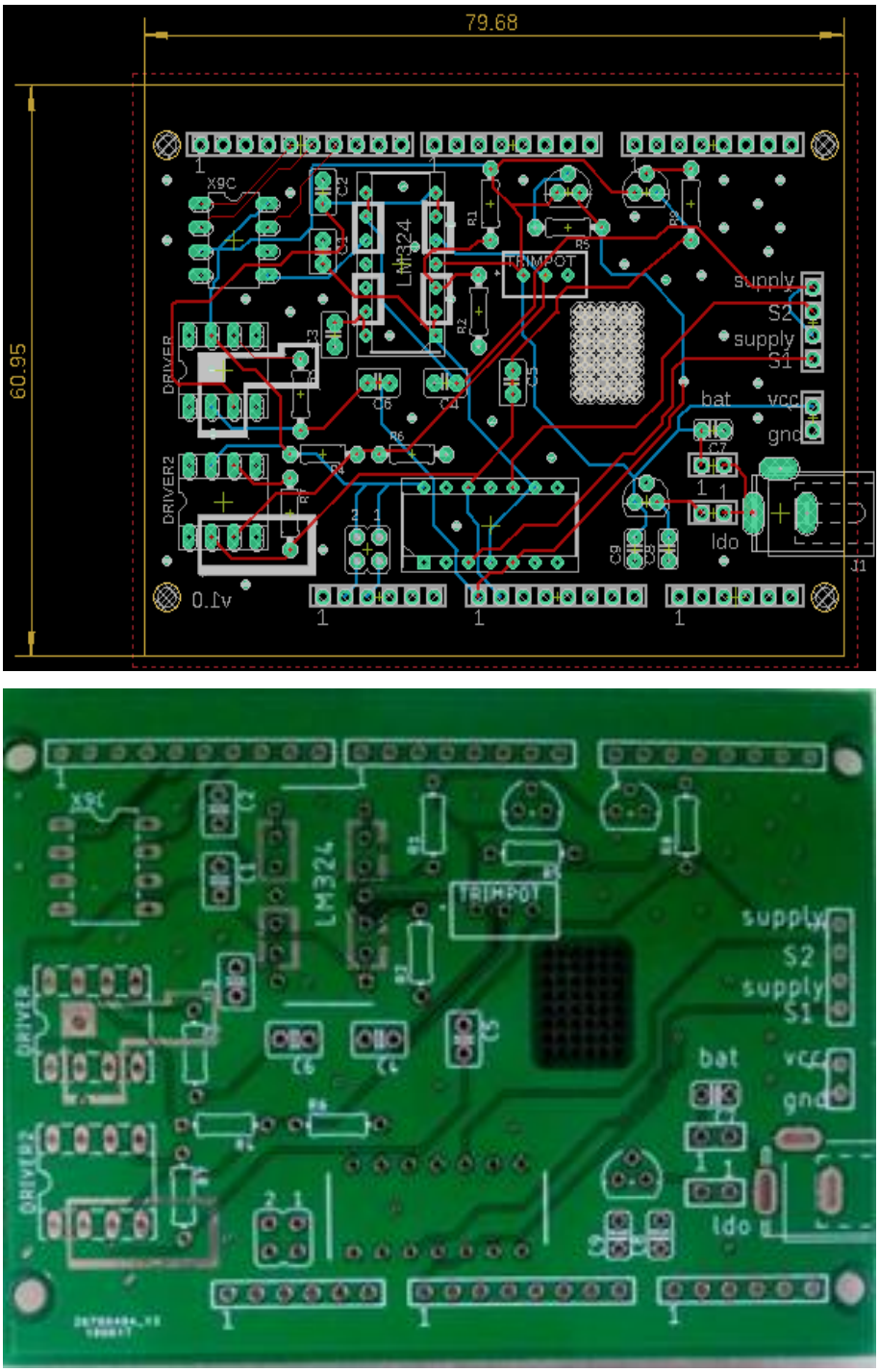

Fig. 6.13 – V1.0 layout and its respective pcb PCB

Annex H : **PCB V1.1 schematic and layout**

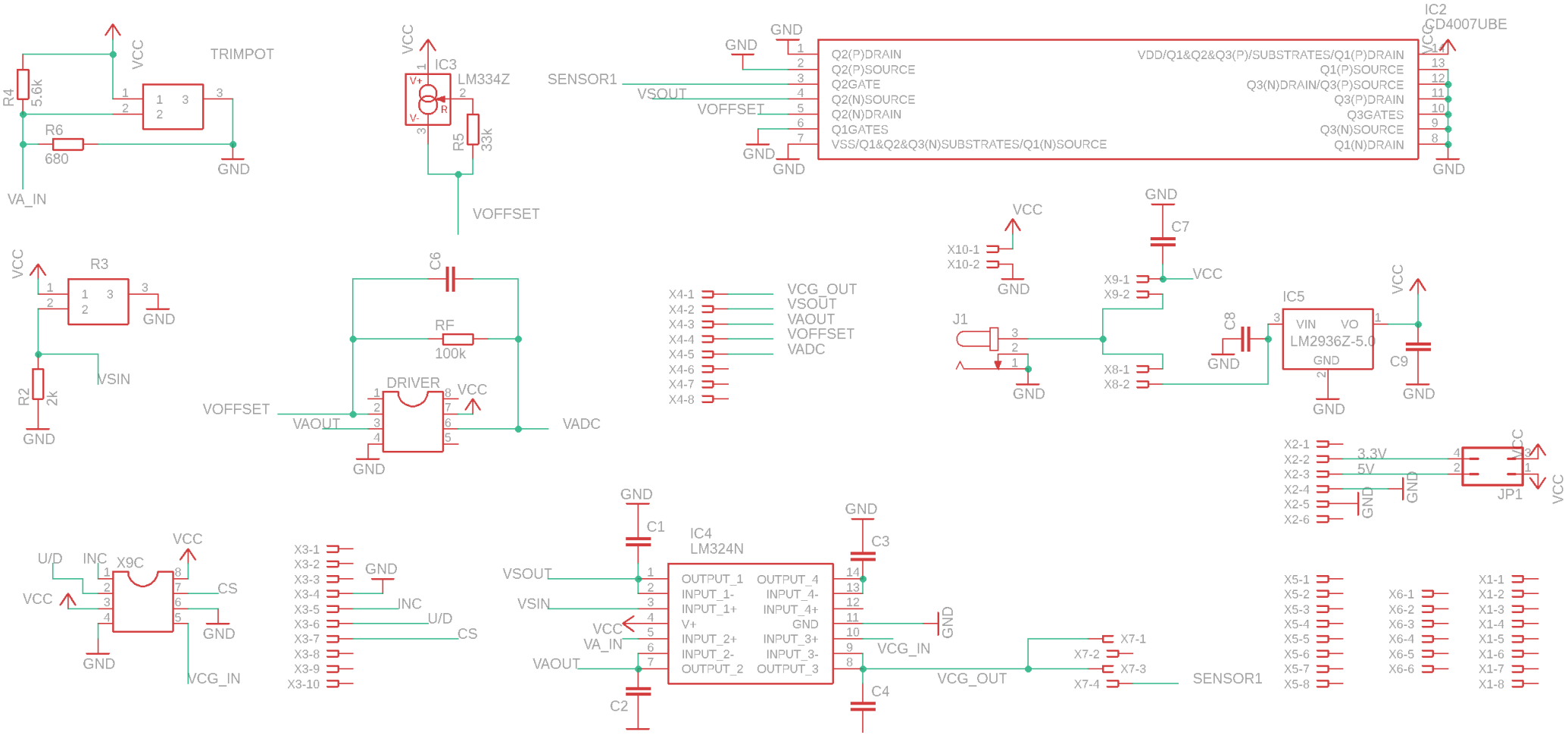

Fig. 6.14 - Corrected schematic used to develop the V1.1 PCB layout, showing all the components and required connections for the AFE circuit

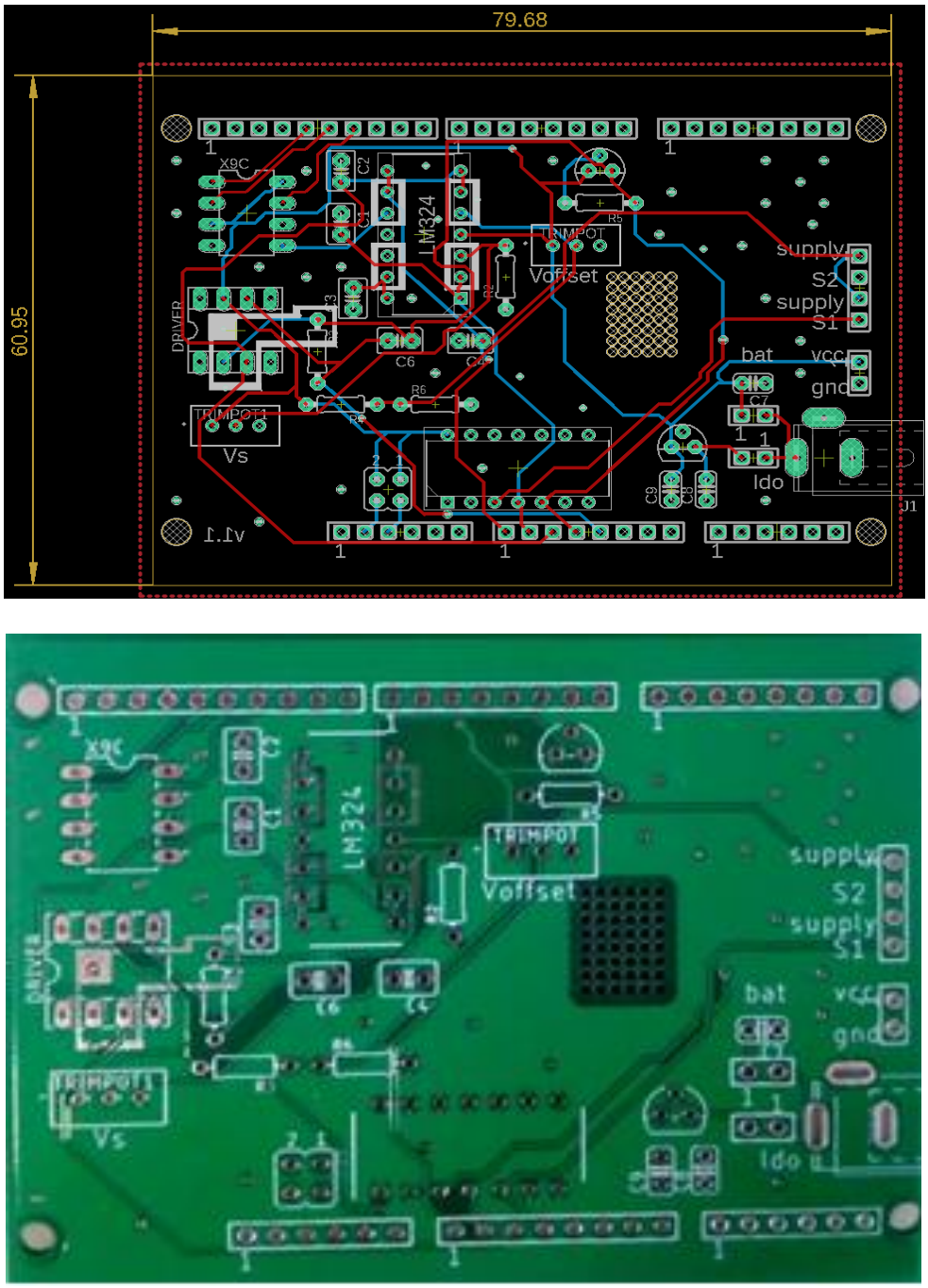

Fig. 6.15 – V1.1 layout and its respective PCB

Annex I : PCB design considerations

The PCB layout was designed thinking of integration with the Arduino board. Stacking the PCB with the microcontroller minimizes the use of noisy jumper wires. To do so, a series of pin headers were added to the schematic that is connected to the ICs and would match analog, PWM, supply and ground pins of the microcontroller. All of the dimension considerations were done using an Arduino Mega 2560 schematic.

Next, voltage supply alternatives to the batteries were added by introducing the DC jack input. It allows 2 configurations: using a jack connector from the battery case instead of connecting 2 wires to the V_{cc} and Gnd pins, or if the voltage supply is above 10V it passes through a voltage regulator assuring

5v supply. Also, the circuit could be powered by the Arduino supply using jumper connectors. Depending on the voltage regulator to be used C8 and C9 should be of values advised in the device datasheet. In this work, the power supply was always batteries and so this feature was not tested.

Due to the use of high-precision OA, using guard-rings in the inputs is advisable. Since these devices present really low input currents, any surrounding signals or contaminants can create a potential difference and therefore unwanted leakage current. To prevent this issue, low impedance guard rings are set up surrounding the inputs from all sides, above and below, connected through a via that sets uniformity[57]. The lowest impedance source available in the PCB is the ground plane and so, the guard rings were connected to it. Besides guarding the AD8651 OA, also the inputs in the LM324 were shielded, however this device as much higher input current and so the leakage current associated with the surroundings won't affect the device performance. Nevertheless, it's considered good practice in PCB design.

The traces thickness is normally arbitrary as long as it corresponds with the supplier manufacturing possibilities. Since the circuit is relying on low power, the traces are not required to be wide. According to Digi-key PCB trace width calculator, for a 10 mA current, with an estimated temperature rise of 10 °C and room temperature of 25 °C, the minimum trace width required is approximately 0.03 mil[58]. For the current design, 10 mil traces were chosen for supply and analog signal and 6 mils for digital signal wiring. The wiring traces width is mostly limited by the usable PCB space and the manufacturer limitations. When wiring needs to cross, it's advisable to cross them at right angles to avoid crosstalk. This detail was considered later and should be taken into account for future designs.

The ground was defined as a ground plane that covers the entire circuit. It's considered a good strategy to lower the overall ground plane impedance and a bad strategy when a high heat dissipation circuit is used. Considering that this is a low power circuit, the heat dissipation is also very low. However, a lot of heat dissipation vias were added in the PCB to minimize thermal-induced noise in the ground plane. Also, 2 different ground planes were defined. An analog and a digital ground plane. According to Texas instruments, it's a good practice for noise suppression and both planes should never be stacked. This particular consideration is more important in high-speed circuitry since the capacitance between the 2 ground planes will couple high-speed noise[59]. In this work, the circuit is dealing with DC and so, it's not so alarming. However, is still good practice to separate the ground planes and should be taken into account in future designs. Also, the ground planes of the PCB should make the connection with Arduino analog and digital ground to close the circuit.

- Version 1.0 troubleshooting

The first main issue with the first version of the PCB was that the digital ground plane was defined but it was not connected with the rest of the circuit. This way, the digital potentiometer doesn't work. The solution for this problem was to solder a wire connecting the ground pins of the digital potentiometer to another shared ground pin.

Next, the variables that were being monitored are Vadc from sensor 1 and Voffset2, from sensor 2, instead of the desired Vadc and Vadc2. This happened due to 2 nets having the same name and resulted in a misplaced connection. This detail was corrected in the V1.0 schematic. Also, only the output was being monitored. It's possible to control the other parameters once using a voltmeter, however, it's more difficult to access the problems. To solve this issue the same strategy can be used. Soldering wires from the correct nodes to not connected pins, sharing now the same voltage becoming possible to acquire the data.

- Version 2

Version 2 was developed to fix the errors listed above. This version was designed to acquire the signal from Vcg, Vs, Va, Voffset, and Vadc and to be ready to communicate with the software. Also, the

digital ground connection was fixed, closing the circuit loop. The wiring traces were also updated where the supply Vcc is 12 mils wide, the analog signal 10 mil wide and the digital signal wiring is 9 mils wide.

Since low-noise is preferable than performing parallel measurements, one of the TIA OA was removed and battery supply is maintained, keeping the supply alternatives.

Given that there were no problems regarding space or manufacturer limitations it was chosen to improve wiring. Finally, a potentiometer to control Vs was added thinking of future applications and additional control. There is still room for improvement as the connections should be updated to right angle crossings when traces from different planes cross.