



# Rapport de Projet Thématique

Système d'acquisition pour textile intelligent

**Caio H Barz Molinari  
Youssra Makmar**

Enseirb-Matmeca  
Département d'Électronique  
PR214 - Projet thématique.  
Mai 2023

# Contenu

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Matériaux et Méthodologie</b>	<b>4</b>
2.1	Bas Niveau . . . . .	5
2.1.1	Matériaux . . . . .	5
2.1.2	Structure du programme embarqué . . . . .	7
2.2	Haut Niveau . . . . .	8
2.2.1	Matériaux . . . . .	8
2.2.2	Structure du logiciel . . . . .	8
<b>3</b>	<b>Résultats</b>	<b>10</b>
3.1	Tests Unitaires . . . . .	10
3.1.1	Test du Multiplexeur ADG726 . . . . .	10
3.1.2	Test de l'ADC ADS1115 et de sa bibliothèque . . . . .	10
3.1.3	Test de sauvegarde dans la carte SD . . . . .	11
3.1.4	Test de réception des données en Python . . . . .	11
3.1.5	Test de sauvegarde des données dans Influx db . . . . .	12
3.1.6	Test d'affichage local et sur Grafana . . . . .	13
3.2	Tests intégrés . . . . .	14
<b>4</b>	<b>Conclusion</b>	<b>17</b>

# Liste de Figures

2.1	Schéma de l'architecture du projet . . . . .	5
2.2	Schéma électrique simplifié de la carte d'acquisition . . . . .	6
2.3	Schéma simplifié du module ADS1115 . . . . .	6
2.4	Flux du code en bas niveau . . . . .	7
2.5	Timing du programme embarqué . . . . .	8
2.6	Flux du code en haut niveau . . . . .	9
3.1	Montage utilisé pour les tests unitaires . . . . .	10
3.2	Lecture de l'ADC . . . . .	11
3.3	Test de sauvegarde dans la carte SD . . . . .	11
3.4	Test de réception de données . . . . .	12
3.5	Test de sauvegarde dans la base de données en cloud . . . . .	13
3.6	Test de mesures avec jambe en plastique . . . . .	14
3.7	Affichage des mesures sur Grafana . . . . .	14
3.8	Histogramme généré à partir de fichier local .csv . . . . .	15
3.9	Tableau de bord avec mesure de température . . . . .	15
3.10	Montage avec Raspberry Pi . . . . .	16
3.11	Test avec Raspberry Pi . . . . .	16

# Chapitre 1

## Introduction

Depuis quelques années, l'ENSEIRB-MATMECA et la société Thingva, une entreprise active dans le domaine pharmaceutique, la recherche académique et l'impression 3D, collaborent étroitement pour développer un système de suivi de différents paramètres physiologiques. Ce système, dont le principe repose sur l'utilisation d'un certain nombre de voies de mesures équipées d'un filament résistif chacune, fera l'objet de notre projet. Ces filaments peuvent être étirés ou allongés en fonction des déformations qui leur sont appliquées.

Thingva souhaite intégrer ces capteurs au textile d'une veste dans le cadre d'un essai clinique. Le but principal serait de pouvoir reconstruire les déformations de la cage thoracique des patients portant ladite veste et d'exploiter les résultats obtenus à des fins de surveillance médicale, de diagnostic ou encore de recherche. Les avantages liés à l'implémentation d'un tel dispositif sont multiples. Tout d'abord, ces capteurs offriront une surveillance et un suivi constant et précis des paramètres physiologiques liés à la santé respiratoire et aux mouvements du corps du patient, permettant ainsi le diagnostic précoce et le suivi clinique de certaines anomalies respiratoires. En outre, une telle technologie représente une avancée considérable dans le biomédical, ouvrant ainsi des horizons prometteurs dans le domaine du textile et des accessoires intelligents intégrés au secteur de la santé, et contribuant à l'amélioration du quotidien du patient.

Jusqu'à présent, la conception électronique a conduit à la conception d'un prototype fonctionnel d'un frontal d'acquisition simple voie. Le concept de celui-ci repose sur l'injection d'un courant programmable et la mesure différentielle de tension. Par la suite, le circuit a été multiplexé permettant ainsi la mesure de plusieurs voies de calcul simultanément avec un seul et même circuit d'acquisition. À long terme, l'objectif final serait de pouvoir traiter plusieurs centaines de voies en même temps. Pour chaque filament résistif que l'on souhaite mesurer, il est nécessaire d'utiliser deux interrupteurs distincts : l'un pour l'excitation, c'est-à-dire pour appliquer un courant à travers le filament, et l'autre pour établir la connexion avec l'amplificateur d'instrumentation afin de pouvoir effectuer la mesure. En parallèle de ces éléments analogiques, un micro-contrôleur Arduino Uno permet d'assurer la communication entre les différents composants électroniques et la synchronisation des mesures, et un programme Python est spécialement conçu pour l'enregistrement et la visualisation des données. Les deux se complètent et facilitent ainsi l'utilisation du dispositif.

L'objectif de ce projet est de compléter le développement de ce système d'acquisition afin de soutenir la mise en place de l'essai clinique mentionné précédemment avant tout, mais aussi d'évaluer si la solution actuelle peut être adaptée et peut répondre à des plus grands besoins si elle est étendue à plus grande échelle.

Ce projet nous permettra d'explorer des fonctionnalités diverses de l'électronique analogique et numérique en instrumentation, particulièrement les applications embarquées, notamment dans les secteurs de la santé et de l'environnement. Pour notre part, nous nous concentrerons principalement sur la partie numérique de la conception.

## Chapitre 2

# Matériaux et Méthodologie

Pour mener à bien notre projet, ce dernier a été réparti entre deux binômes afin de diviser le travail de manière équilibrée et efficace. Le premier binôme s'est concentré sur la partie analogique, tandis que le deuxième s'est occupé de la partie numérique. Ce rapport est consacré à la partie numérique du projet.

Le but principal du projet est **d'améliorer la base de code existante, visualiser et analyser les données issues du prototype existant**. Concrètement, ceci correspond à :

1. Écrire les routines élémentaires permettant au binôme n°1 de réaliser la campagne de mesures désirées.
2. Récupérer les données produites lors de la campagne de mesures du binôme n° 1.
3. Traiter, analyser et visualiser ces données.
4. Améliorer les éléments logiciels déjà existants :
  - (a) Uniformisation, refactorisation et nettoyage du code ;
  - (b) Amélioration de l'interface pour un usage par une personne non experte.

La méthodologie employée consiste à utiliser le micro-contrôleur Arduino Uno comme interface d'acquisition et transmission de données. Tout d'abord, on s'intéresse à la mesure de la température de l'environnement et chacune des 16 voies de capteurs. Le capteur de température ds18b20, qui utilise le protocole de communication One Wire, a été choisi pour effectuer les mesures de température. Comme le convertisseur analogique numérique externe (ADC) proposé dans le cadre de ce projet (ADS1115 - utilisation du protocole de communication I2C) n'a qu'une seule entrée, deux multiplexeurs 16-1, conjointement avec la carte d'acquisition, ont été utilisés pour mesurer chacune des voies résistives de façon quasi-parallèle. À ce stade, un premier enregistrement de données est réalisé en utilisant un module de carte SD (utilisation du protocole de communication SPI) sous format .csv. Ensuite, les données sont envoyées à travers la communication série UART vers un ordinateur ou Raspberry-Pi (le dernier n'étant utilisé qu'à la fin du projet) équipé de Python 3.9, qui, à son tour, a pour rôle de traiter la réception des données, de les sauvegarder localement sous format .csv et de les envoyer vers la base de donnée InfluxDb en *cloud*. Une fois les données bien enregistrées sur la base de données, le logiciel Grafana, en ligne, est employé pour l'affichage et l'analyse des résultats.

Le diagramme 2.1 illustre le flux du projet décrit ci-dessous.

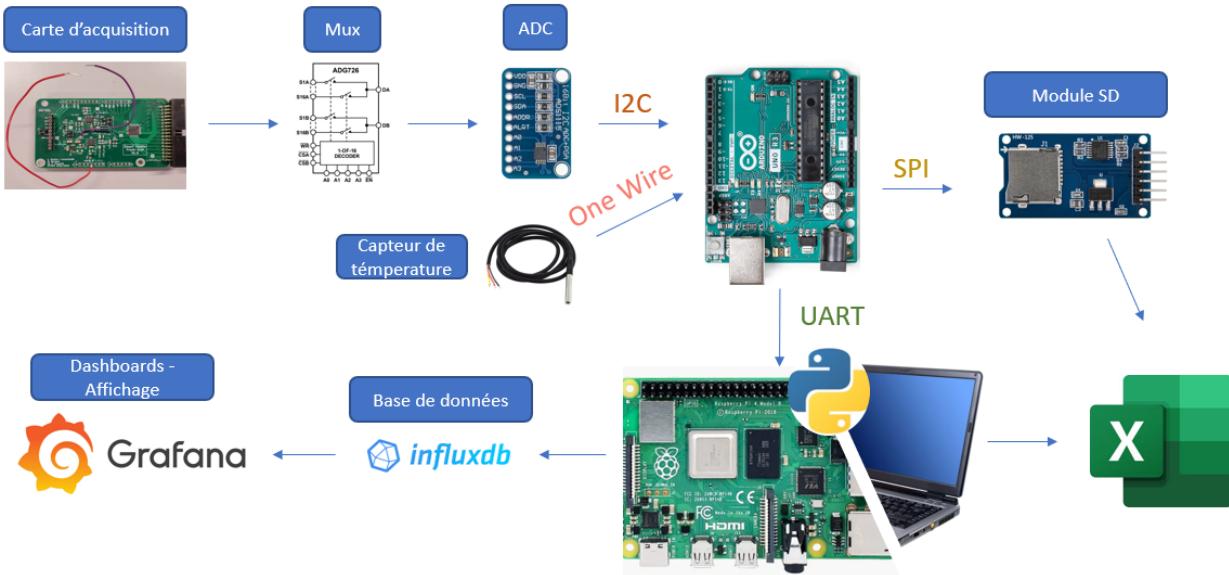


Figure 2.1: Schéma de l'architecture du projet

Maintenant que l'architecture générale a été décrite, le travail achevé sera détaillé en deux parties: une partie bas niveau et une partie haut niveau du code.

## 2.1 Bas Niveau

### 2.1.1 Matériaux

Les outils utilisés, ainsi que les démarches employées pour les piloter, seront décrits ensuite dans l'ordre présenté dans la figure 2.1.

- **Carte d'acquisition:** Ce prototype est le produit des projets SATI précédents et il permet de déduire la résistance d'un filament résistif à travers une source de courant interne dont la valeur est connue (environ 1 mA). En observant la variation de résistance, il est possible d'en déduire la distorsion mécanique responsable de ce changement. L'objectif de ce projet est de mesurer 16 voies résistives. Pour établir l'interface de chaque voie avec l'ADC (convertisseur analogique-numérique) et, par la suite, avec le microcontrôleur, il était nécessaire d'utiliser un **multiplexeur 16-1**. Ce multiplexeur permet de basculer rapidement entre les différentes voies afin d'obtenir des mesures quasi-simultanées.
- **Multiplexeur ADG726** Comme mentionné précédemment, le module de multiplexage est utilisé pour basculer entre les voies des filaments résistifs. La particularité de cette technique est que, en réalité, il faut employer deux multiplexeurs 16-1 en parallèle pour y aboutir. Cela est dû au fait qu'il y a deux interrupteurs à commuter afin de transmettre à la chaîne d'acquisition la valeur courante de tension d'une certaine voie: l'un pour activer la source de courante, et l'autre pour connecter la voie à l'entrée d'un amplificateur opérationnel (AOP). La figure 2.2 démontre plus nettement cette dynamique. Le module choisi pour cette tâche a été le *dual-multiplexer* ADG726. Son pilotage est assez simple, il suffit de suivre la table de vérité figurant sur son *datasheet* [1].

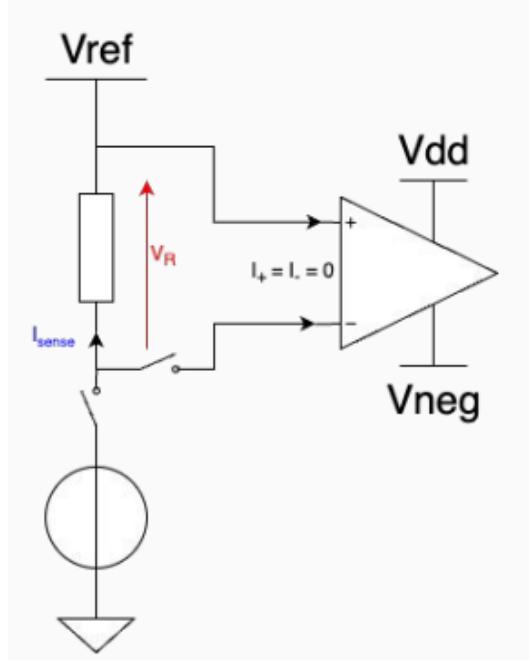


Figure 2.2: Schéma électrique simplifié de la carte d'acquisition

- **Convertisseur Analogique-Numérique (ADC) ADS1115:** Une fois le pilotage du module de multiplexage réussi, il faut convertir les données analogiques obtenues sous forme numérique afin de les traiter. Pour ce faire, l'ADC ADS1115 de chez Texas Instruments [2] a été utilisé. Il s'agit d'un convertisseur externe à 16 bits qui utilise le protocole de communication I<sup>2</sup>C et dont l'opération est configurable - mesure simple (*single-ended*) ou différentielle, acquisition en continu ou *single shot*, fréquence d'échantillonage et plage des valeurs de tension d'entrée - selon les besoins de l'utilisateur. Dans le cadre de ce projet, le mode de mesure simple et la plage de tension de 2,048 V ont été choisis. Le choix des paramètres d'acquisition sera exposé dans la section 2.1.2. Afin de communiquer avec celui-ci, une bibliothèque propre a été développée, encapsulant des commandes de la bibliothèque standard de l'Arduino Uno pour la communication I<sup>2</sup>C, Wire. Bien qu'il existe des bibliothèques permettant d'utiliser le ADS1115, la bibliothèque développée est plus légère en mémoire et plus efficiente en termes de structuration du code. Nous pouvons observer le diagramme de blocs simplifié de ce module sur la figure 2.3

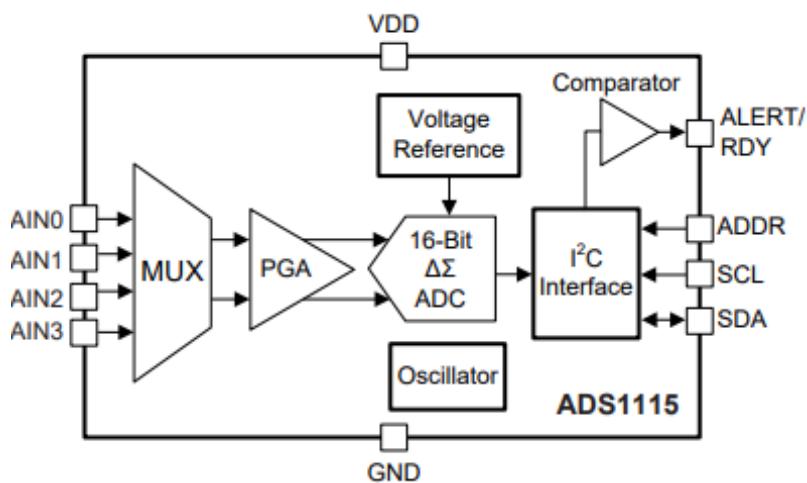


Figure 2.3: Schéma simplifié du module ADS1115

- **Capteur de température DS18B20:** Pour réaliser une analyse plus complète des mesures des voies résistives, la température de l'environnement a été mesurée dans le but de voir si les mesures sont faussées en cas de changement de température. Pour ce faire, le capteur DS18B20 a été employé; il mesure des températures de -10 °C à +85 °C avec une précision de ±0,5 °C. Il utilise le protocole One Wire, en

envoyant 12 bits de données vers le master [3].

- **Module carte SD:** Un adaptateur de carte SD utilisant le protocole SPI a été utilisé pour faire une sauvegarde préliminaire des données mensurées (température et tension des voies) sous format .csv.

### 2.1.2 Structure du programme embarqué

Une fois que l'ensemble des composants dont on a besoin ont été étudiés et pilotés individuellement par des programmes d'essai, un code intégrant tous les composants de la chaîne bas niveau a été mis en place. Ce *firmware* avait pour objectif, et ce, **chaque seconde**, de:

- Mesurer les 16 voies résistives de la carte d'acquisition ainsi que la température ambiante;
- Sauvegarder les données dans la carte SD;
- Transmettre les données vers l'ordinateur à travers le protocole UART.

Pour répondre à la contrainte de temporisation imposée, une interruption de *timer* a été mise en place pour contrôler le basculement des multiplexeurs ainsi que les mesures. L'ADC, à son tour, a été configuré avec une fréquence d'échantillonage de 64 Hz (même fréquence d'interruption du *timer*) dans le mode single-shot (c'est-à-dire, l'ADC réalise des conversions sur demande).

Dans un premier temps, le système exploitable embarqué *FreeRtos* a été utilisé dans le but de diviser l'exécution des tâches et de rendre possible l'expansion simplifiée du *firmware* si nécessaire. Bien que les tâches soient bien mises en place, un problème est survenu lors de l'utilisation du module SD. Comme l'écriture dans la carte SD est faite par blocs de 512 KB, allouant ainsi 512 kB de mémoire RAM à chaque cycle d'écriture/lecture, l'arduino UNO, ayant 2 GB de mémoire RAM, a manqué de mémoire pour supporter à la fois le module SD et l'*overhead* du système opérationnel.

Par conséquent, nous avons opté pour le paradigme plus conventionnel de programmation des systèmes d'exploitation à 8 bits. Dans un premier temps, les mesures des 16 voies résistives ainsi que la température sont effectuées. Ensuite, les données obtenues sont sauvegardées dans la carte SD et transmises vers l'ordinateur. Le diagramme 2.4 illustre le flux décrit.

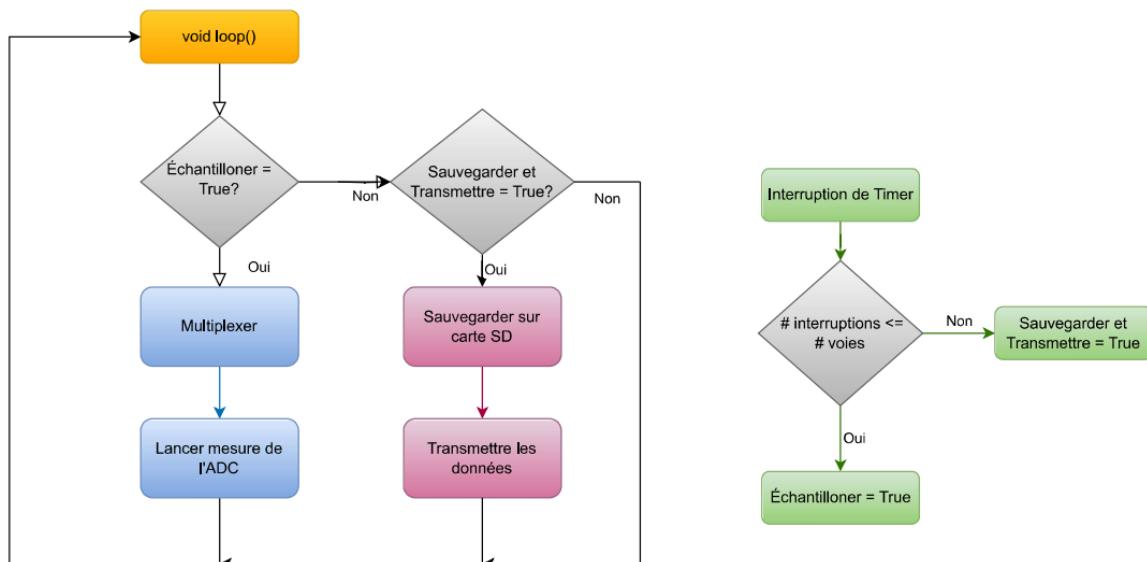


Figure 2.4: Flux du code en bas niveau

Dans le but de garantir la bonne temporisation du programme, un test a été exécuté à l'aide du hardware du micro-contrôleur: Lors du début d'une tâche (mesure, sauvegarde ou transmission), une broche était activée à un état "haut" (5V), tandis qu'à la fin de cette tâche, elle était ramenée à un état "bas" (0V). De ce fait, le temps d'exécution de chaque tâche, ainsi que le temps disponible pour l'ajout d'autres éventuelles tâches, a été mesuré comme démontré par la figure 2.5.

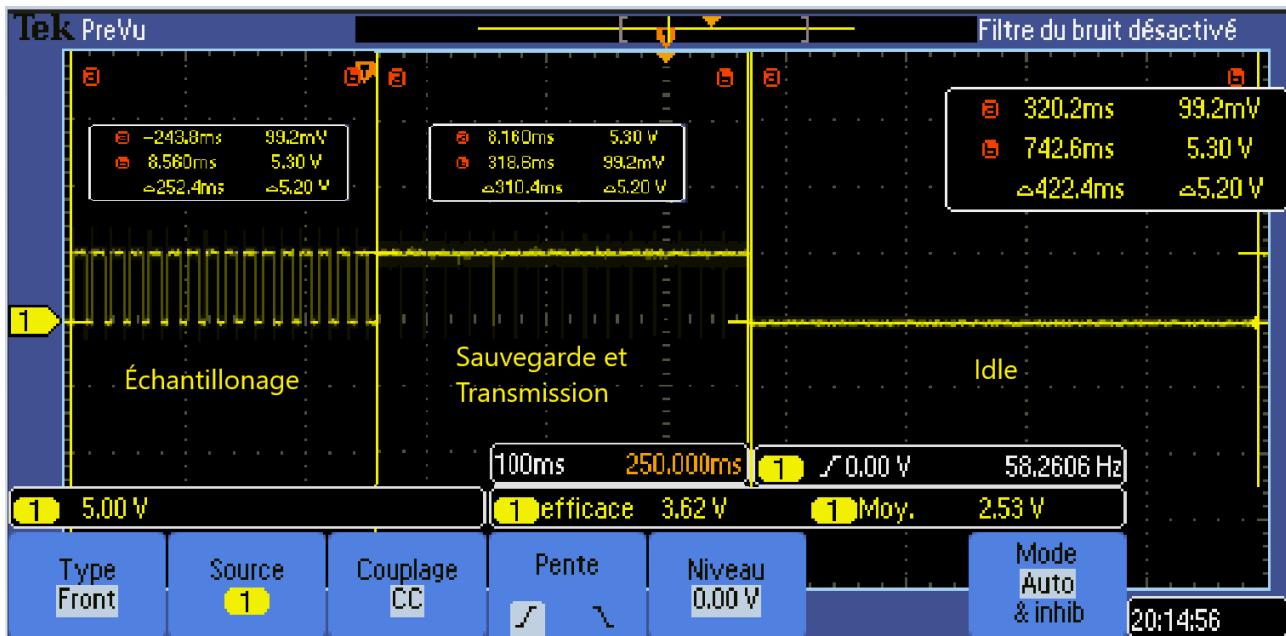


Figure 2.5: Timing du programme embarqué

Comme attendu, le cycle complet des tâches dure 1 seconde et il reste encore 423 ms qui peuvent être occupées par des éventuelles tâches supplémentaires.

Le code en bas niveau ayant réussi, nous avons entamé le développement du code haut niveau.

## 2.2 Haut Niveau

### 2.2.1 Matériaux

- **Python 3.9:** Une fois les données envoyées par le microcontrôleur, le choix du langage de programmation pour leur traitement s'est porté sur Python en raison de sa flexibilité et de la disponibilité de ses bibliothèques. Les mesures reçues sont traitées et envoyées vers la base de données Influx db.
- **Influx db cloud:** Il s'agit d'une base de données en *cloud* dédiée à stocker toutes les données provenant du code Python.
- **Grafana:** Il s'agit d'une interface de visualisation de données en *dashboards*. Dans le cadre de ce projet, la base de données créée sur Influx dB a été intégrée avec un tableau de bord Grafana. Pour interagir avec la base de données (*réaliser des queries*), le langage *flux* (similaire à SQL mais orienté vers les scripts) a été utilisé.

### 2.2.2 Structure du logiciel

Le logiciel a été développé sous forme de script, ce qui signifie que des arguments sont passés au programme (notamment la quantité de mesures à effectuer), il est exécuté et une fois que toutes les tâches sont achevées, le programme se termine. Il ne s'agit donc pas d'une interface utilisateur (GUI) locale.

L'exécution du programme se divise en quatre principales étapes:

1. **Réception et traitement:** La bibliothèque pySerial a été utilisée pour recevoir les données à partir des portes USB au *baud rate* de 115200 à travers le protocole de communication UART. Bien que fiable, la transmission de données peut être légèrement bruitée, en ajoutant des fois des caractères indésirables. Pour résoudre ce problème, la bibliothèque des expressions régulières, Regex, a été utilisée. Le Regex rend possible le filtrage et la séparation des données en groupes selon le motif de caractères attendu. Ainsi, bien qu'il y ait des caractères "intrus", cela ne posera pas de problème à la réception tant que la transmission n'est pas brutalement modifiée.
2. **Sauvegarde:** Une fois les données reçues et traitées, elles sont sauvegardées localement sous format .csv (il faut souligner que ce fichier est rempli tout au long de l'exécution du programme pour qu'on ne perde

pas les données en cas de panne d'une partie de la chaîne de transmission/réception) et dans la base de données en *cloud* Influx dB.

3. **Affichage Locale:** Les données étant sauvegardées dans le fichier .csv, on peut choisir d'afficher des histogrammes correspondant aux mesures de chaque voie résistive. Pour ce faire, la bibliothèque matplotlib a été utilisée.
4. **Affichage en ligne:** Cette partie n'est pas directement liée à l'exécution du programme, mais elle en découle. Une fois la sauvegarde dans Influx dB réussie, il est possible d'utiliser l'outil Grafana pour visualiser les données sous forme de tableaux de bord.

Le diagramme de la figure 2.6 illustre le flux d'exécution du programme.

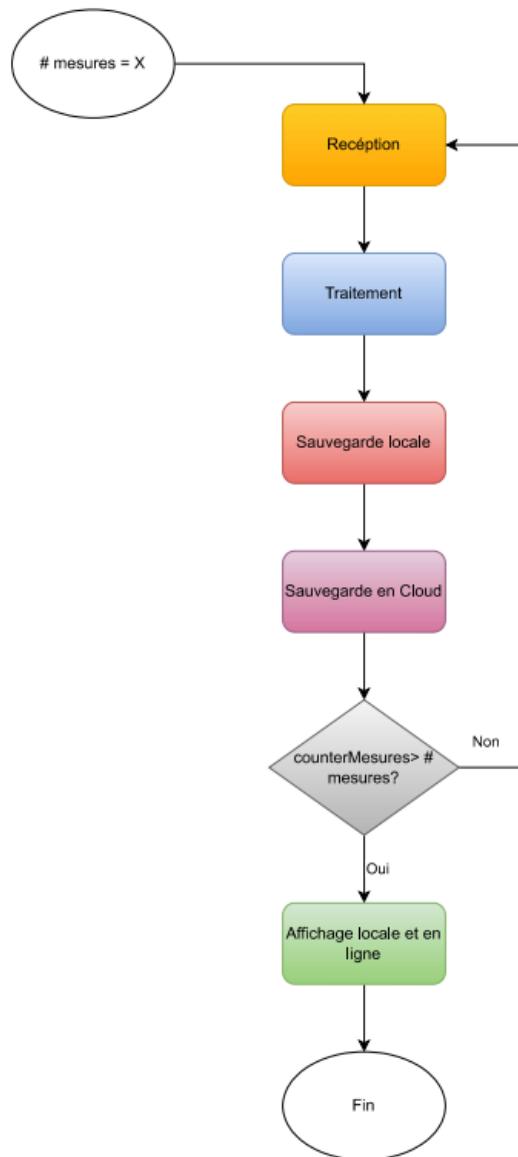


Figure 2.6: Flux du code en haut niveau

La méthodologie étant présentée, les résultats de ce projet seront discutés dans la prochaine section.

# Chapitre 3

## Résultats

Dans cette section, des résultats de tests seront présentés et discutés. Comme mentionné précédemment, le travail a été réparti entre deux binômes: le premier était responsable de la partie analogique et le deuxième (les auteurs de ce rapport), de la partie numérique. Par conséquent, les résultats seront également divisés en deux parties; la première partie concerne des tests "unitaires" de la partie numérique et la deuxième, des tests intégrés entre les deux binômes.

### 3.1 Tests Unitaires

Pour les tests unitaires, le montage montré dans la figure 3.1 a été utilisé où le potentiomètre émule les voies résistives. D'ailleurs, le test unitaire du multiplexeur a été réalisé avec la carte d'acquisition.

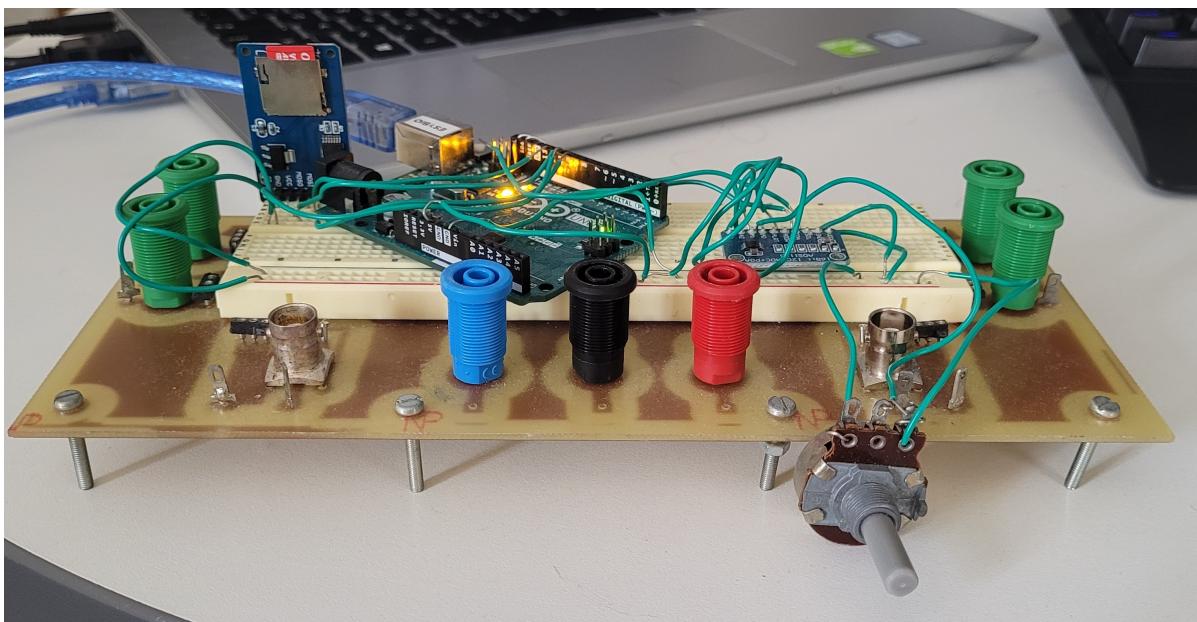


Figure 3.1: Montage utilisé pour les tests unitaires

#### 3.1.1 Test du Multiplexeur ADG726

En suivant la table de vérité du datasheet du multiplexeur ADG726, le multiplexeur a pu être piloté avec succès. Pour vérifier son bon fonctionnement, la continuité entre les entrées et les sorties (A et B) a été testée.

#### 3.1.2 Test de l'ADC ADS1115 et de sa bibliothèque

Comme mentionné dans la section 2.1.1, une bibliothèque propre a été développée pour piloter le convertisseur analogique-numérique ADS1115. Pour vérifier son fonctionnement, des mesures ont été réalisées et l'erreur de mesure respective moyenne en a été déduite. La figure 3.2 montre la lecture de l'ADC, em mV avec 1460 mV en entrée.



Figure 3.2: Lecture de l'ADC

En réalisant quelques mesures, l'erreur moyenne de mesure a été déterminée comme étant de  $\pm 0.9\%$

### 3.1.3 Test de sauvegarde dans la carte SD

En utilisant le montage de la figure 3.1, des mesures ont été sauvegardées avec succès comme le montre la figure 3.3, où chaque ligne correspond aux 16 voies résistives.

The screenshot shows a Notepad window titled '\*SAMPLES2.CSV - Bloco de Notas'. The content of the file is a CSV file with 16 columns, each containing the value '0223' repeated 16 times, representing 16 resistive channels.

Figure 3.3: Test de sauvegarde dans la carte SD

### 3.1.4 Test de réception des données en Python

Ce test a eu pour but de vérifier la bonne réception et le traitement des données issues du micro-contrôleur. Le programme reçoit, à travers le protocole UART, les données brutes comme montré dans la figure 3.4a et les réorganise en groupes à travers le regex comme le montre la figure 3.4b. Il faut souligner qu'à ce stade le capteur de température n'était pas intégré au projet, donc les mesures de température ont été écrites en dur.

0000&Sensor 0:434;ÿSensor 1:434;ÿSensor 2:434;ÿSensor 3:434;ÿSensor 4:434;ÿSensor 5:434;ÿSensor 6:434;ÿSensor 7:434;  
;ÿSensor 8:434;ÿSensor 9:434;ÿSensor 10:434;ÿSensor 11:434;ÿSensor 12:434;ÿSensor 13:434;ÿSensor 14:434;ÿSensor 15:  
434;Temp: 25.00

(a) Chaîne de caractères bruitée reçue par l'ordinateur

```
Sensor: 6
Reading :434
-----
Sensor: 7
Reading :434
-----
Sensor: 8
Reading :434
-----
Sensor: 9
Reading :434
-----
Sensor: 10
Reading :434
-----
Sensor: 11
Reading :434
-----
Sensor: 12
Reading :434
-----
Sensor: 13
Reading :434
-----
Sensor: 14
Reading :434
-----
Sensor: 15
Reading :434
-----
Temperature:25.00
```

(b) Données reçues réorganisées

Figure 3.4: Test de réception de données

### 3.1.5 Test de sauvegarde des données dans Influx db

Comme détaillé dans la section 2, une fois les données traitées par le regex, elles seront sauvegardées dans la base de données en *cloud* Influx db. La figure 3.5 montre que les données ont été bien enregistrées.

```
1 SELECT *
2 FROM "Sensors"
3 WHERE
4 time >= now() - interval '30 days'
5
```

Ready (525ms)

CSV Past 30d RUN

	Sensor 06	Sensor 07	Sensor 08	Sensor 09	Sensor 10	Sensor 11	Sensor 12	Sensor 13	Sensor 14	Sensor 15	Temperature	time
	no group long	date:2023-04-24T14:11:44.894Z										
126	127	128	129	130	131	132	133	134	135	135	25	2023-04-24T14:11:44.894Z
126	127	128	129	130	131	132	133	134	135	135	25	2023-04-24T14:19:47.585Z
819	818	819	818	819	818	819	818	819	818	818	25	2023-04-24T14:29:47.411Z
819	818	819	818	819	818	819	818	819	818	818	25	2023-04-24T14:29:48.302Z
819	818	819	818	819	818	819	818	819	818	818	25	2023-04-24T14:29:49.284Z
819	818	819	817	819	818	819	818	818	818	818	25	2023-04-24T14:29:50.252Z
818	818	819	818	818	818	819	818	819	818	818	25	2023-04-24T14:29:51.236Z
818	818	819	818	819	817	819	818	819	818	818	25	2023-04-24T14:29:52.243Z
819	817	819	818	819	818	819	818	819	818	818	25	2023-04-24T14:29:53.226Z

1 2 3 4 5 ... 213 >

Figure 3.5: Test de sauvegarde dans la base de données en cloud

### 3.1.6 Test d'affichage local et sur Grafana

Les tests d'affichage depuis le fichier .csv local, ainsi que ceux réalisés sur Grafana, ont été concluants. Pour être concis, les résultats d'affichage seront présentés dans la section 3.2, car les mesures qui y sont présentées ont été réalisées avec la carte d'acquisition, ce qui leur confère une importance plus élevée par rapport aux mesures effectuées avec le potentiomètre.

### 3.2 Tests intégrés

Dans cette section, les résultats des tests qui intègrent le travail des deux binomés impliqués dans ce projet sont présentés. Cette fois-ci, le montage utilise non seulement la carte d'acquisition mais aussi une jambe en plastique visant à émuler, évidemment, une vraie jambe humaine. Le but était de mesurer les voies résistives en observant des variations de résistance. La figure 3.6 illustre le montage employé.

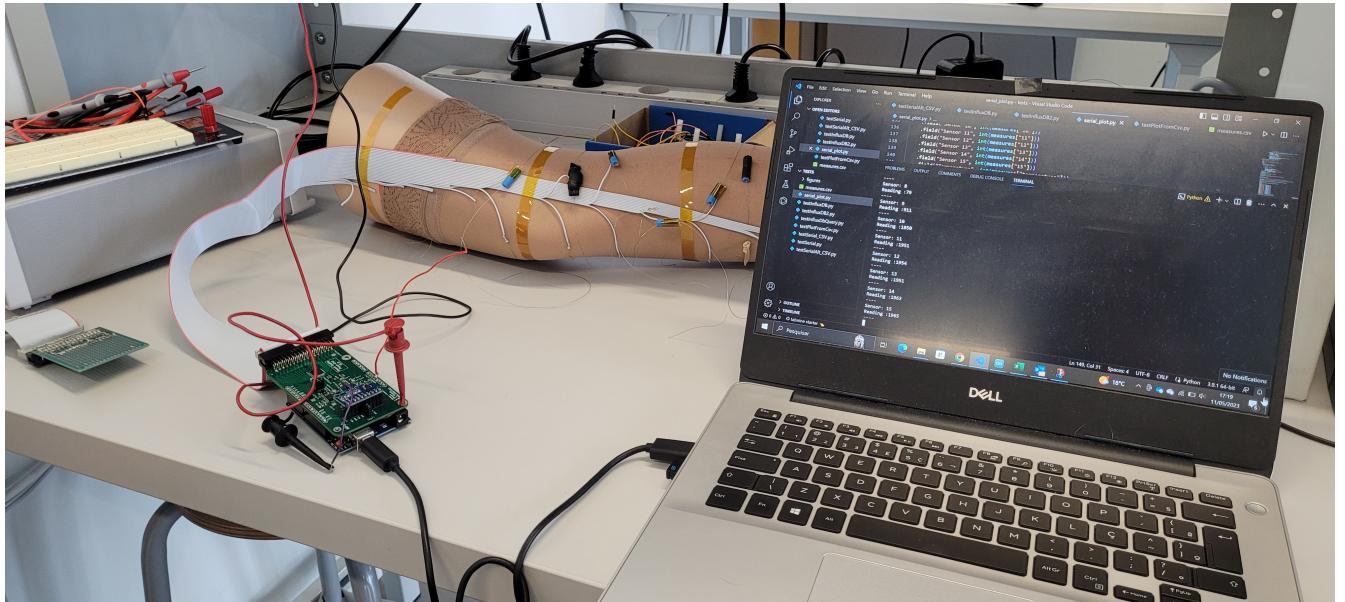


Figure 3.6: Test de mesures avec jambe en plastique

Un test court, d'environ une minute, a été conduit. Les mesures ont pu être visualisées à travers un histogramme généré à partir du fichier .csv local et aussi sur Grafana, où un tableau de bord qui comprend les voies résistives et la température (qui à ce stade n'avait pas encore été intégrée) sous forme de graphiques en séries temporelles ainsi qu'un histogramme a été créé. Les résultats sont montrés dans les figures 3.7 et 3.8.

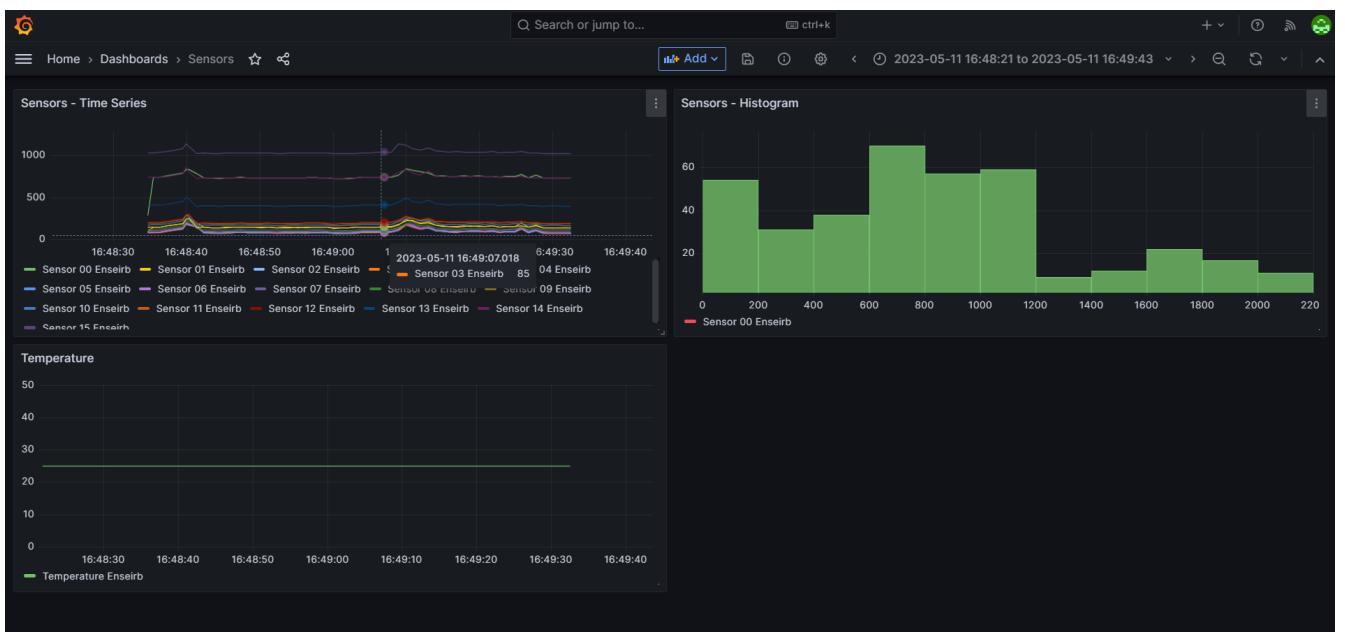


Figure 3.7: Affichage des mesures sur Grafana

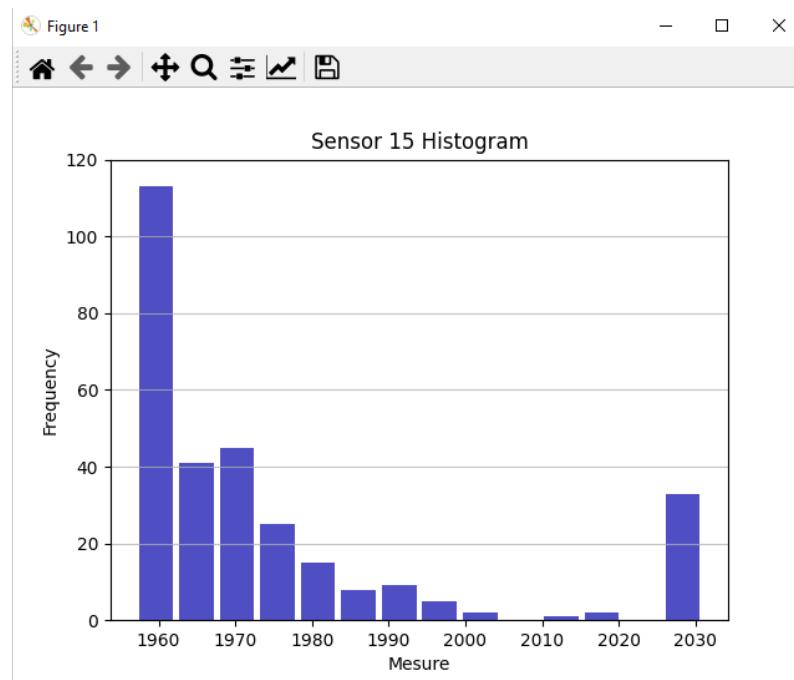


Figure 3.8: Histogramme généré à partir de fichier local .csv

Dans un second temps, la température a été intégrée au tableau de bord comme le montre la figure 3.9.

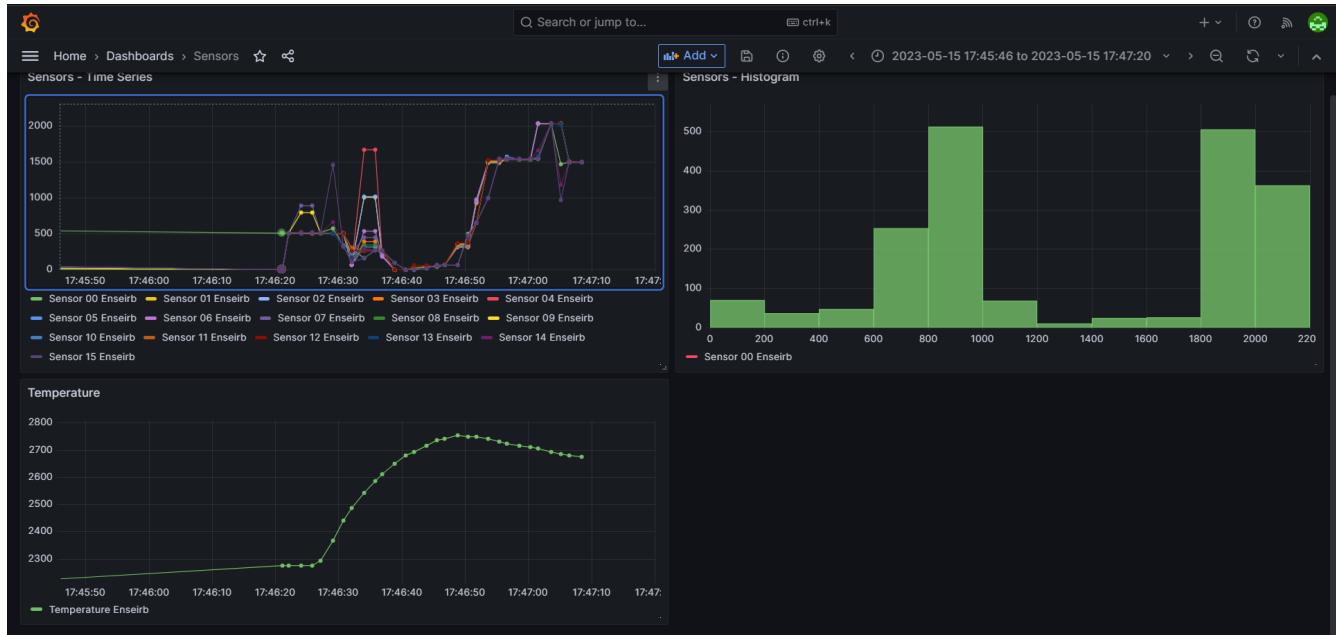


Figure 3.9: Tableau de bord avec mesure de température

L'ajout de la mesure de température est important pour vérifier l'occurrence des dérives des résistances selon la température, comme mentionné dans la section 2.1.1.

Le dernier test à effectuer était un test de longue durée, s'étalant sur plusieurs heures. Pour ce faire, on a utilisé une Raspberry Pi (un micro-ordinateur dont l'OS est une version embarquée de la distribution Linux Debian) pour faire tourner le programme en Python comme le montrent les figures 3.10 et 3.11.

Bien que la configuration de la Raspberry pi et le démarrage du programme aient réussi, un problème inconnu est survenu lors de son exécution, entraînant la corruption de l'OS embarqué.

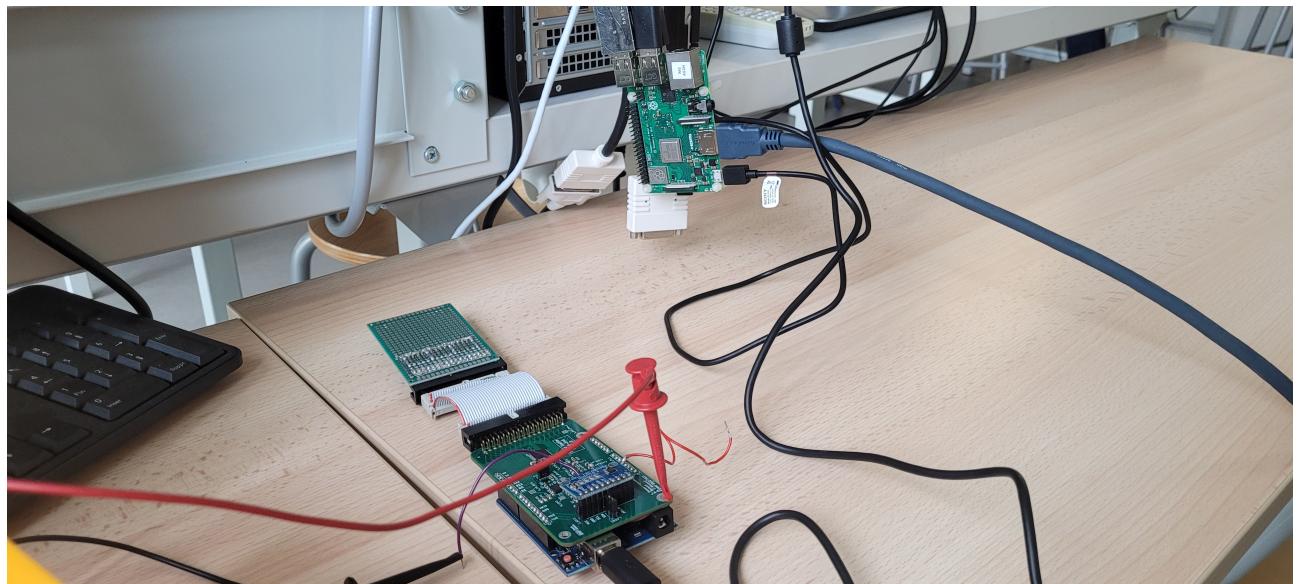


Figure 3.10: Montage avec Raspberry Pi

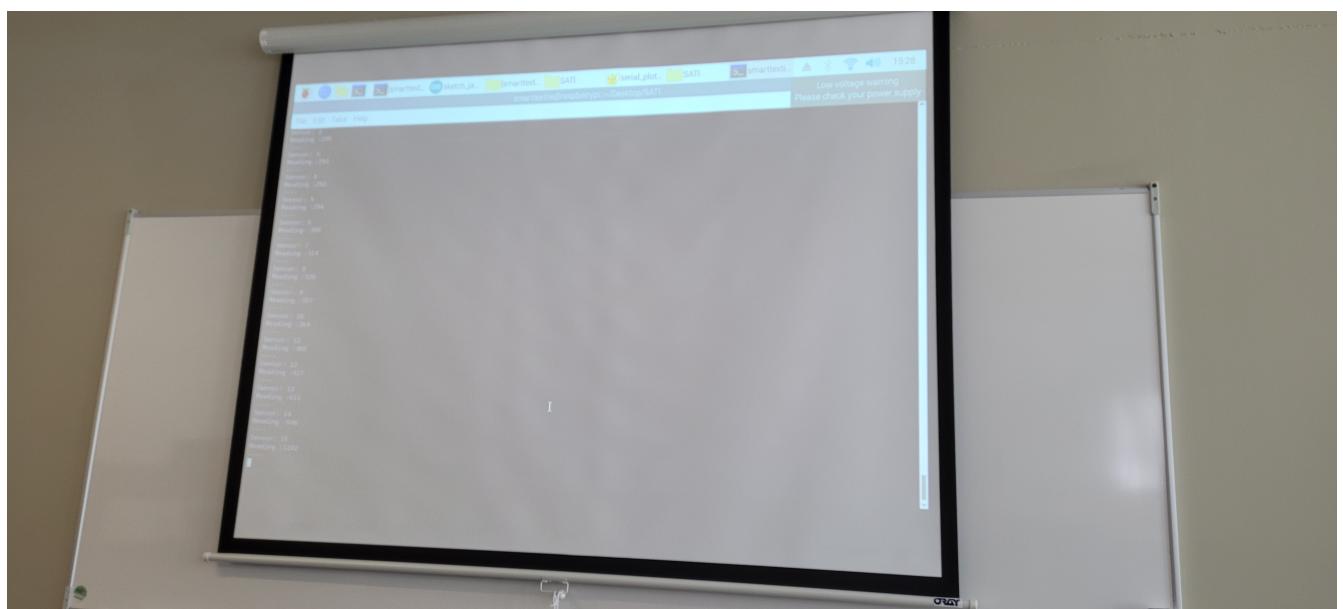


Figure 3.11: Test avec Raspberry Pi

## Chapitre 4

# Conclusion

Dans le cadre de ce projet, nous avons pu valider plusieurs étapes. Tout d'abord, grâce à l'utilisation d'un multiplexeur, nous avons pu effectuer des mesures simultanées sur les 16 voies résistives, ce qui nous a permis d'obtenir des données précises et complètes. De plus, nous avons intégré un capteur de température DS18B20 pour mesurer la température de la pièce, nous permettant ainsi d'observer les variations de résistance en fonction de la température également.

Les données collectées ont été stockées dans une carte SD au format .csv, permettant une sauvegarde facile et une analyse ultérieure. Nous avons également mis en place une transmission des données vers l'ordinateur via le protocole UART.

Dans le processus de traitement des données, nous avons utilisé Python pour analyser et exploiter les mesures obtenues et nous avons ensuite stocké les données traitées dans la base de données Influxdb. Pour la visualisation des données, nous avons utilisé Grafana pour créer un tableau de bord.

Enfin, nous avons réalisé un test supplémentaire en remplaçant une véritable jambe humaine par une jambe en plastique. Cela nous a permis d'observer les variations de résistance et de valider le bon fonctionnement du système dans des conditions similaires à celles d'une utilisation "réelle".

En conclusion, notre projet a été couronné de succès. Nos tests démontrent le potentiel de notre système dans le suivi des paramètres physiologiques et montrent que notre carte peut-être utilisée dans nombreuses applications futures.

# Bibliographie

- [1] Analog Devices. *ADG726 Datasheet (PDF)* - *Analog Devices*. URL: [https://www.analog.com/media/en/technical-documentation/data-sheets/adg726\\_732.pdf](https://www.analog.com/media/en/technical-documentation/data-sheets/adg726_732.pdf). (accessed: 16.05.2023).
- [2] Texas Instruments. *ADS1115 Datasheet*. URL: [https://www.ti.com/product/ADS1115?utm\\_source=google&utm\\_medium=cpc&utm\\_campaign=asc-dc-null-prodfolderdynamic-cpc-pf-google-wwe\\_int&utm\\_content=prodfolddynamic&ds\\_k=DYNAMIC+SEARCH+ADS&DCM=yes&gclid=CjwKCAjw04yjBhApEiwAJcvNoes0Brn8vBwE&gclsrc=aw.ds](https://www.ti.com/product/ADS1115?utm_source=google&utm_medium=cpc&utm_campaign=asc-dc-null-prodfolderdynamic-cpc-pf-google-wwe_int&utm_content=prodfolddynamic&ds_k=DYNAMIC+SEARCH+ADS&DCM=yes&gclid=CjwKCAjw04yjBhApEiwAJcvNoes0Brn8vBwE&gclsrc=aw.ds). (accessed: 16.05.2023).
- [3] Maxim Integrated. *DSB18B20 Datasheet*. URL: <https://www.analog.com/media/en/technical-documentation/data-sheets/DS18B20.pdf>. (accessed: 17.05.2023).