PROJET TUTEURE MCPC

2023-2024

IUT DE LILLE

2023-2024

Delcour Adrien / Cache Josse / Leroy Thibaut / Calligaro Bastien

Sommaire :

Introduction :

Les batteries jouent un rôle prépondérant dans notre société moderne, alimentant une vaste gamme de dispositifs, des smartphones aux véhicules électriques. La performance d'une batterie dépend étroitement de son état de charge, un paramètre fondamental qui influe sur son efficacité, sa durée de vie et sa sécurité d'utilisation. Notre projet se concentre sur l'analyse approfondie de l'influence de plusieurs grandeurs physiques cruciales.

Ce projet est une suite de celui de l'année dernière qui était inachevée en raison de soucis liés à la programmation du spectromètre fibré. Les objectifs et les rôles restent ainsi les mêmes que l'année passée.

En premier lieu, nous allons discuter d'un nouvel aspect électrochimique qui a été étudié lors du S5. Le transport de matière qui sert à faire fonctionner l'électrolyte.

Dans un second temps, nous présenterons la méthode du cyclage avec la batterie en flux.

Nous parlerons ensuite des résultats des premières manipulations lancées. Ces manipulations nous permettent de déterminer à quel débit fixé le rendement de la batterie est le meilleur. Ce débit sera ainsi celui utilisé pour les manipulations avec le spectromètre fibré.

Enfin, nous continuerons sur l'avancée de la programmation servant à piloter tous nos capteurs extérieurs à la batterie tels que le spectromètre fibré ou encore des sondes PT100 à l'aide d'une interface graphique nommé « TechVolt »

I / Transport de matière de l'électrolyte :

Notre batterie fonctionne à l'aide de deux solutions : une solution de ferricyanure de potassium et une solution de ferrocyanure de potassium toutes deux à 0,1 mol.L

Analyse des couples Red/Ox :

Premier couple : $Fe(CN)_6^{3-}/Fe(CN)_6^{4-}$

Demi-équation : $Fe^{3+} + 6(CN)^- + e^- \leftrightarrow Fe^{2+} + 6(CN)^-$

Potentiel standard : $E_1 = E^0 + 0{,}06 \log\left(\frac{[Fe^{3+}][CN^-]^6}{[Fe^{2+}][CN^-]^6}\right) \leftrightarrow E_1 = E^0 = 0{,}41V$

Second couple : O_2/H_2O

Demi-équation : $O_2 + 4H^+ + 4e^- \leftrightarrow 2H_2O$

Potentiel standard : $E_2 = E^0 + 0{,}015 \log([H^+]^4) \leftrightarrow E_2 = E^0 - 0{,}06pH \leftrightarrow E_2 = 1{,}23 - 0{,}06pH$

Troisième couple : H^+/H_2

Demi-équation : $2H^+ + 2e^- \leftrightarrow H_2$

Potentiel standard : Par défaut, $E_3 = -0{,}06pH$

Attributions :

Nous avons 2 réductions qui se font à la cathode :

$$Fe(CN)_6^{3-} \rightarrow Fe(CN)_6^{4-}$$

$$O_2 \rightarrow H_2O$$

Nous avons 2 oxydations qui se font à la cathode :

$$Fe(CN)_6^{4-} \rightarrow Fe(CN)_6^{3-}$$

$$H_2 \rightarrow H^+$$

Or, $O_2, H^+ et\ H_2O$ n'interviennent pas dans les réactions électrochimiques. Les espèces qui vont intervenir sont :

$$Fe(CN)_6^{3-}, Fe(CN)_6^{4-} et\ K^+$$

Structures chimiques des couples Red/Ox :

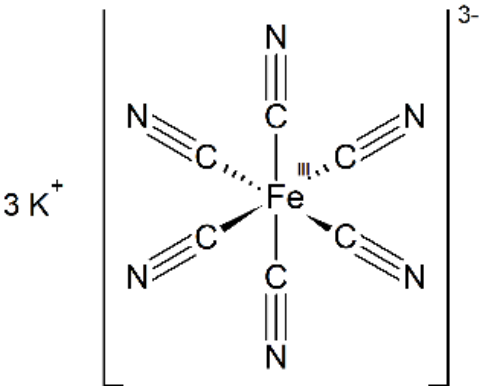


Figure 1 : Ferricyanure de potassium

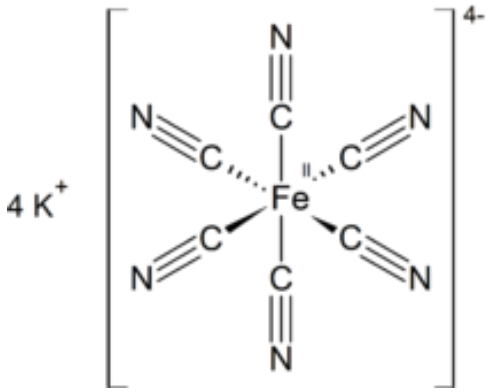


Figure 2 : Ferrocyanure de potassium

Lors de la dissolution, les ions K^+ des complexes ferricyanure et ferrocyanure se séparent de la molécule pour assurer en partie la conduction ionique tandis que les ligands cyanure gardent leur liaison avec l'ion métallique. Ils assurent également la conduction malgré leur taille importante qui ne facilite pas le transport car ils sont fortement chargés (3^- et 4^-).

Nombre de transport :

$$[Fe(CN)_6^{3-}] = [Fe(CN)_6^{4-}] = 0{,}1\ mol.L$$

Il faut 4 moles de K+ pour avoir une mol de ferrocyanure de potassium et 3 mols de K+ pour avoir une mol de ferricyanure de potassium.

Ainsi, $[K^+] = 0{,}7\ mol.L$

$$\lambda_{Fe(CN)_6^{3-}} = \frac{302{,}7 * 0{,}1}{302{,}7 * 0{,}1 + 441{,}6 * 0{,}1 + 73{,}5 * 0{,}7} = 0{,}24$$

$$\lambda_{Fe(CN)_6^{4-}} = 0{,}35$$

λ_{K^+}= 0,41

Ces nombres de transport déterminent le nombre d'électron que chaque espèce transporte.

Schéma de l'électrolyse :

Le courant est fixé à 10 électrons.

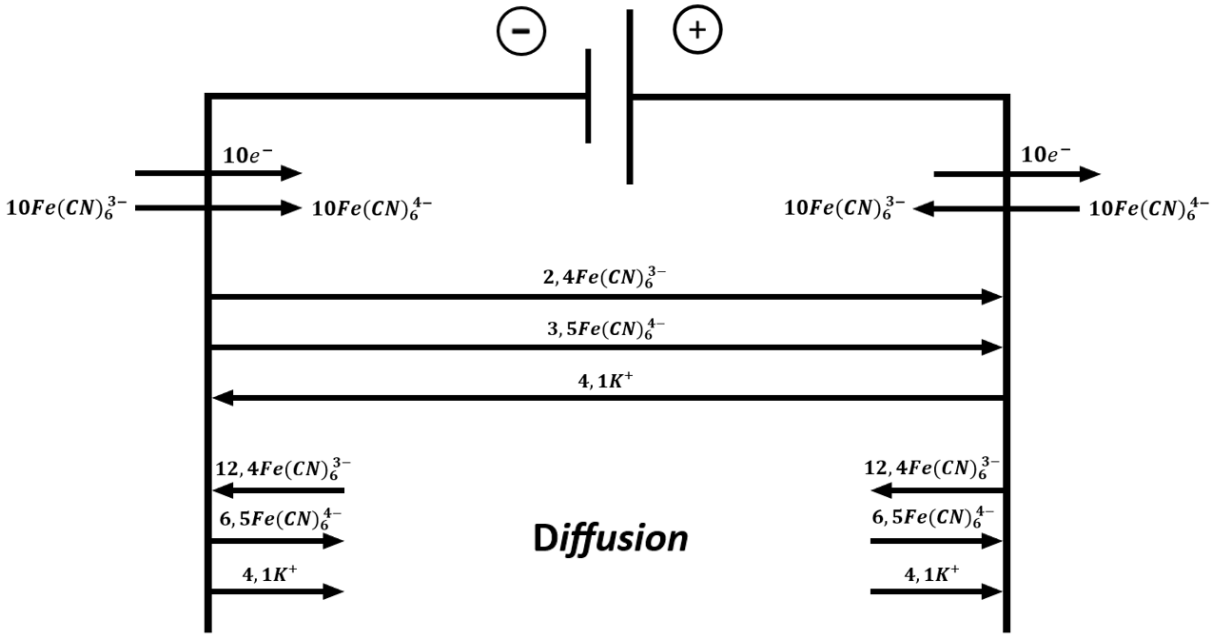


Figure 3 : Schéma du transfert de matière de l'électrolyse

En pratique néanmoins, il n'y a que les ions K^+qui aussure le transport de courant. En cause la membrane de notre batterie qui est une membrane cationique : seuls les cations peuvent circuler d'une électrolyte à une autre.

II / Présentation de la méthode du cyclage avec la batterie en flux

Pour cette partie du projet, nous allons évaluer l'impact de différents paramètres sur le rendement de la batterie et nous pourrons également décrire certains phénomènes au cours du temps.

Les paramètres pouvant être étudiés lors du cyclage de la batterie au cours du temps sont :

- La concentration
- Le volume des électrolytes
- Le débit
- Le courant
- Le pH
- La conductivité
- …

On sait que la concentration et le volume des électrolytes feront varier un autre paramètre qui est la charge Q en Coulomb ou en mA.h.

La concentration et le volume des électrolytes resteront identiques à l'année précédente c'est-à-dire :

- 100 ml de solution de chaque électrolyte (ferro et ferri de potassium dissous)
- 0,1 mol/L

Les paramètres qui seront donc modifiés sont le courant et le débit. Pour ce qui est du pH et de la conductivité, ils seront évalués en fonction de l'avancement du projet.

1°) Paramètre du courant

Afin de couvrir une large gamme de courant on introduit une autre grandeur qui est « C » qui est défini comme la puissance qui fournirait la capacité d'une batterie complète sur une heure de temps. Cela signifie qu'à courant constant, la batterie peut-être complètement chargée ou déchargée en une heure.

On sait que la charge Q lorsque l'on charge à 1C est de 268,02 mA.h (cf. Rapport 2022-2023 / I. 4°). Nous allons varier les acquisitions de cycle charges/décharges de 0,2C jusque 2C lors d'une période de 17 jour théorique, c'est-à-dire 408 heures de cyclage de la batterie.

Avec les temps associés, on obtient :

C	Charge (mA.h)	Temps d'un cycle (h)	14 cycles charge/décharge (h)
0,2	53,60	5,00	180
0,4	107,21	2,50	90
0,6	160,81	1,67	60
1,0	268,02	1,00	36
1,5	402,03	0,67	24
2,0	536,04	0,50	18

Figure 4 : Tableau des paramètres de charge

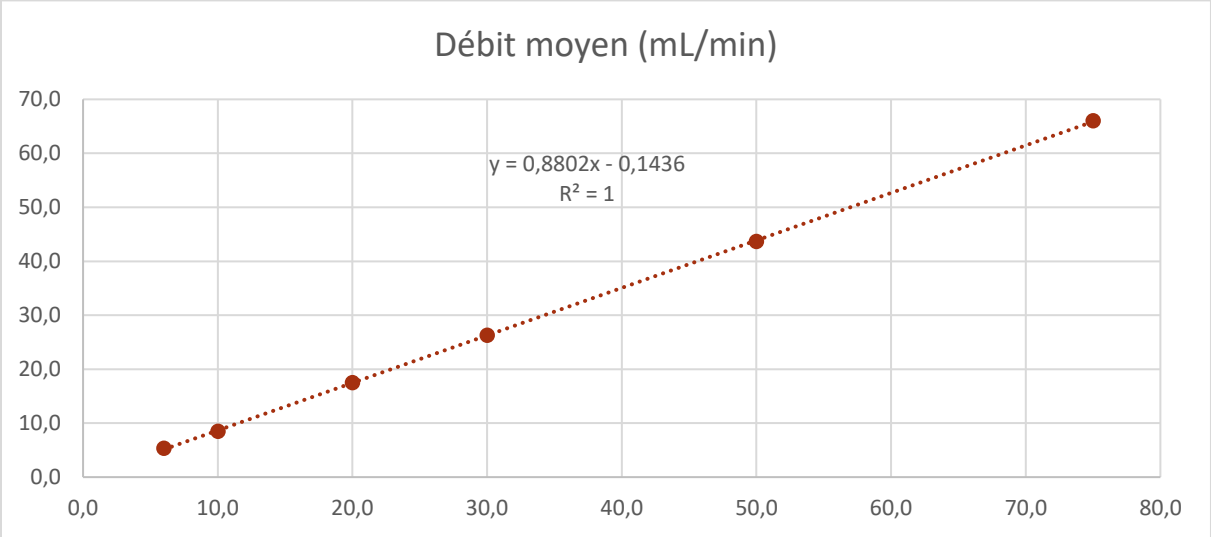
2°) Paramètre du débit

Ensuite, l'autre paramètre étudié sera le débit, nous gardons les paramètres précédents pour le courant qui sont des paramètres « programmables » et allons reproduire la même manipulation à différents débits.

En effet, l'inverse aurait été plus contraignant (courant constant) car pour changer la vitesse de rotation de la pompe péristaltique, il faut le faire manuellement avec la commande numérique ([voir I. 2°- Présentation du matériel : pompe rotative multicanaux] du rapport BUT2).

Les manipulations se dérouleront donc à débit constant.

La vitesse de rotation de la pompe n'ayant pas d'unité, la conversion en débit est effectuée en amont (manipulation et données extraites par Monsieur Beaucamp).



Grâce à l'équation de la régression linéaire, nous pouvons en déduire le débit pour une vitesse de rotation choisie. Afin de couvrir une large gamme de débit et d'évaluer rapidement et efficacement l'impact de celui-ci, nous allons travailler avec des débits d'environ 10, 40 et 73 mL/min qui correspondent respectivement à des vitesses de rotation (en tours/min) de 12, 46 et 84.

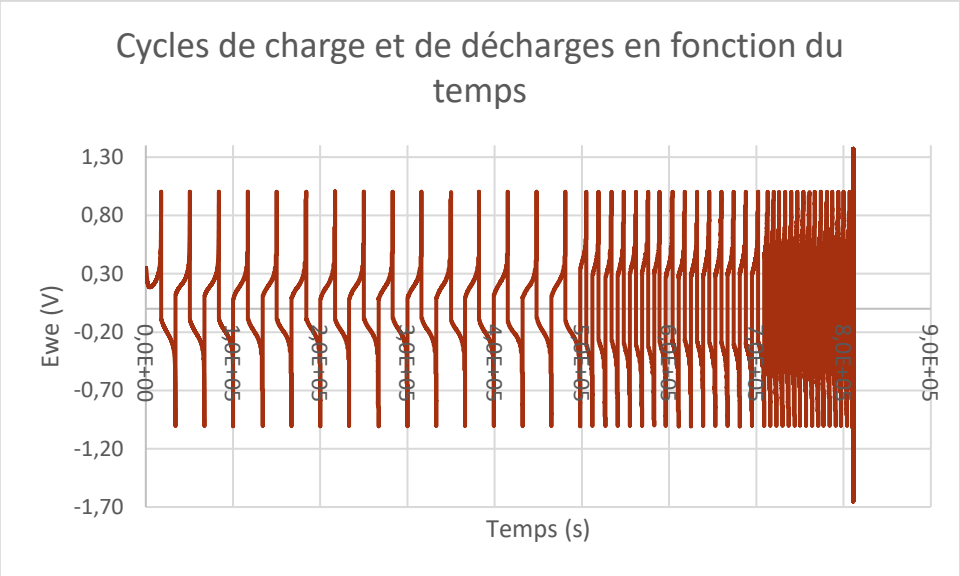
III / Résultats des premières manipulations

Comme rappeler dans plus tôt, nous cherchons à étudier la façon dont réagis la batterie lorsque que nous sommes menées à modifier certains paramètres. Aussi lors des premières manipulations, nous avons modifier l'état de charge ainsi que le débit. Ces modifications nous mènerons à définir les paramètres idéals de fonction de la batterie afin d'obtenir le rendement le plus élevé.

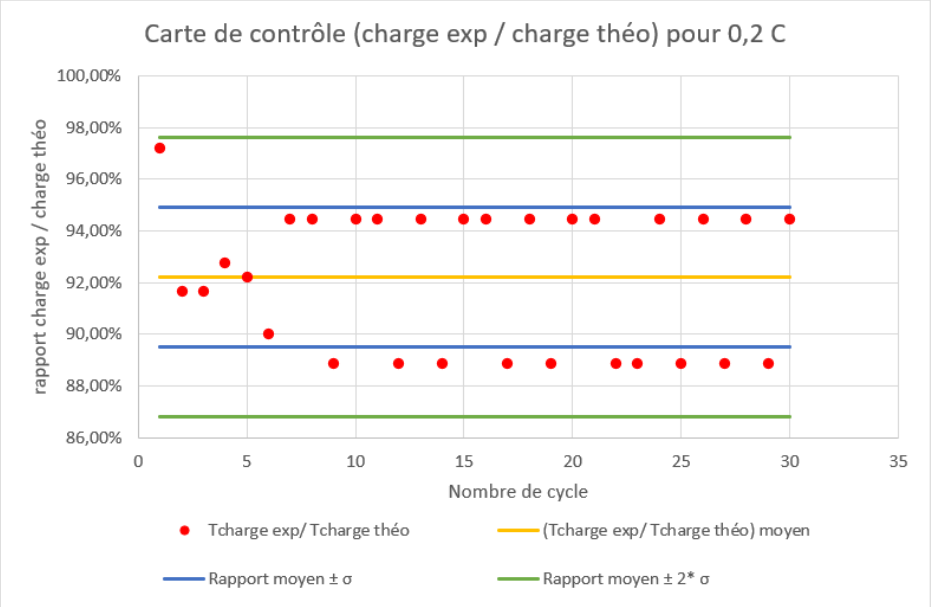
Dans un premier temps nous avons lancé une manipulation à un débit fixé de 10 mL/min et nous avons effectué des cycles à des états de charge différents. Pour ce qui est des états de charge et les courants associés, nous avons balayer la plage suivante :

C	Charge (mA.h)	Temps d'un cycle (h)
0,2	53,60	5,00
0,4	107,21	2,50
0,6	160,81	1,67
1,0	268,02	1,00
1,5	402,03	0,67
2,0	536,04	0,50

Nous avons effectué des cycles de charge et décharge. Nous obtenons à la suite de cette manipulation le graphique suivant :

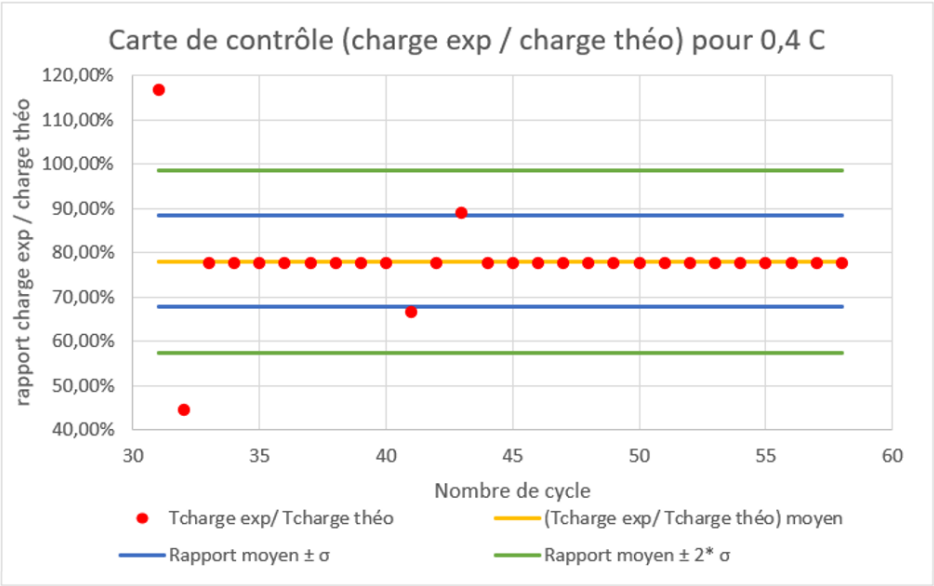


Nous pouvons bien identifier la succession de la charge et de la décharge de la batterie. Aussi nous pouvons identifier clairement les différents états de charges, quoique l'arrêt soudain de cette succession et de la manipulation nous éveille sur un possible problème durant cette dernière.
Afin de vérifier le bon déroulement de la manipulation, nous avons tiré les cartes de contrôle de chacun des états de charges. Pour cela nous pouvons dans un premier temps comparer le temps de charge + décharge théorique à celui observé expérimentalement.



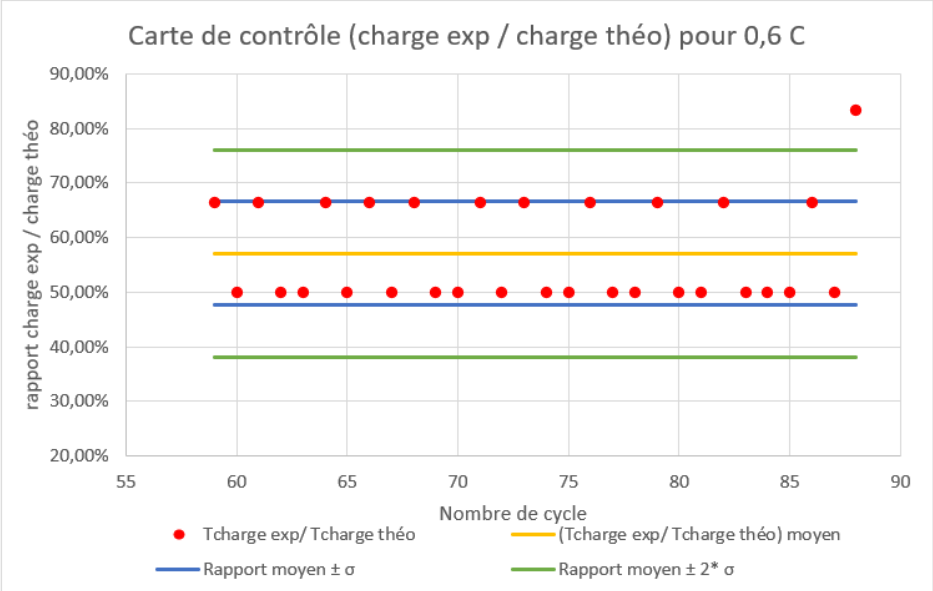
A un état de charge de 0,2 C, le temps de charge/décharge théorique est de 5h. Ici, on observe le rapport Tcharge expérimentale /Tcharge Théorique moyen à 92%, le rapport idéal étant normalement de 100%, nous avons une efficacité correcte. Aussi la dispersion des points n'est pas aberrante ce qui nous rassure quant à l'utilisation de batterie dans ces paramètres.

Les cycles suivants ont été réalisés à un état de charge de 0,4 C la carte de contrôle est la suivante :

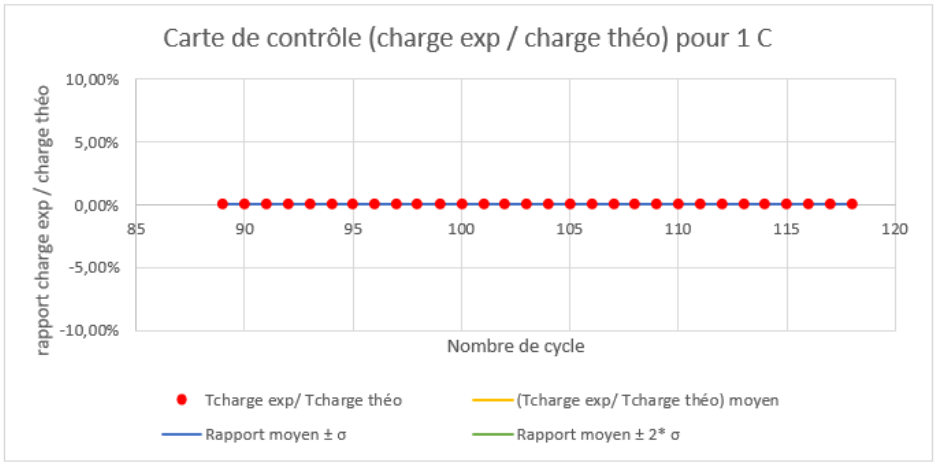


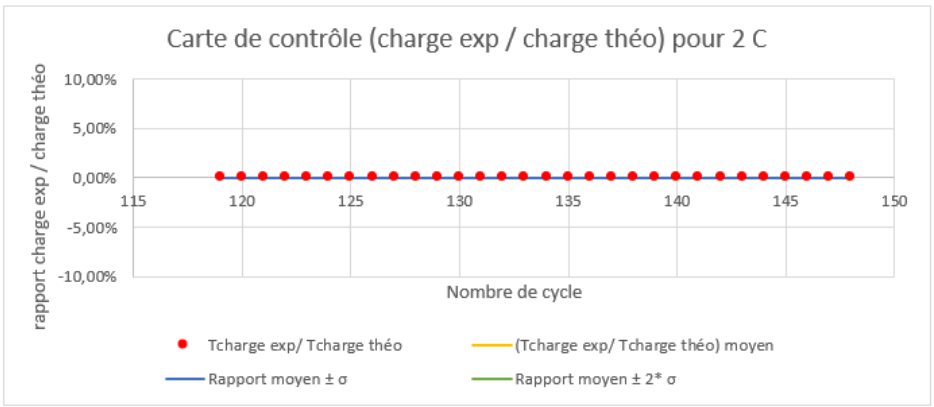
Sur cette carte de contrôle, nous observons un rapport Tcharge expérimentale /Tcharge Théorique moyen à environ 78%. Cela représente une efficacité moindre que lors des cycles à un état de charge de 0,2 C. Cependant la dispersion est plus faible ici. Seuls les deux premiers points ont un comportement différent des autres, on peut l'expliquer par la transition entre les cycles à 0,2 C et les cycles 0,4 C.

Pour ce qui est de l'efficacité, l'allure que à l'air de prendre celle-ci est que plus l'état de charge augmente est plus l'efficacité diminue.



Cette carte de contrôle vient certifier la supposition que nous avons émis plus tôt, en effet plus l'état de charge augmentent plus l'efficacité diminue.





Les carte de contrôle des cycles aux états de charge de 1 C et de 2 C permettent de conclure que le débit choisit lors de cette première manipulation est trop faible pour permettre la batterie de fonctionner efficacement, en effet, la convection n'est pas assez importante pour amener assez de

matière (et donc de réactifs) aux sièges des réactions électrochimiques. De plus lors de ces derniers cycles, la batterie atteint son *Cut off* avant même d'avoir pu commencer à se charger ou à se décharger. En effet la batterie atteint la tension limite inférieure prescrite à laquelle la décharge de la batterie est considérée comme complète ou inversement pour la charge.

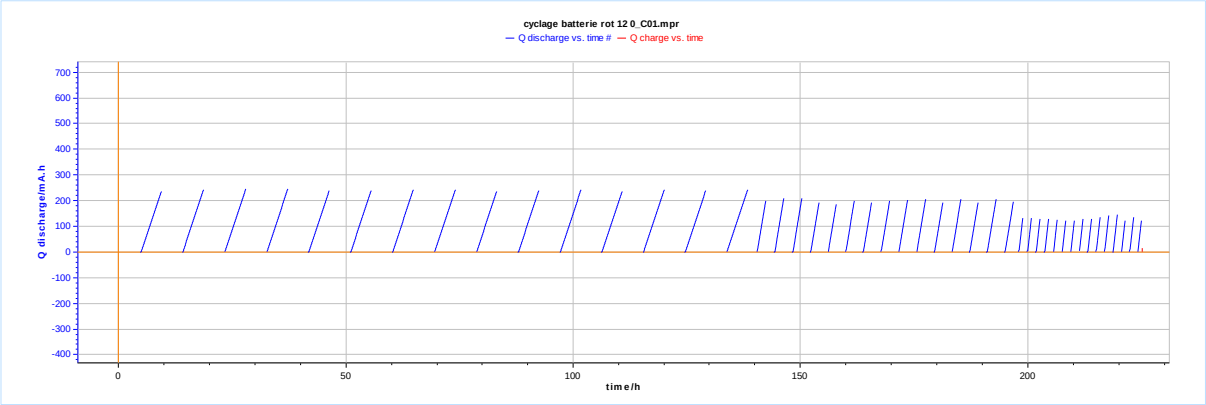


Figure 5 : Graphique des cycles de charges/décharge (mA.h) en fonction du temps

Ce graphique nous permet de vérifier l'hypothèse que nous avons émis ci-dessus, Une fois les cycles à un été de charge de 0,6 terminés, on ne peut plus observé de cycles dû au *Cut off* de la batterie. On remarque bien que plus l'état de charge augmente et plus l'efficacité diminue, jusqu'à être égale à 0.

Suite à cette première manipulation, nous avons effectué deux nouvelles manipulations mais à des débits différents. Cependant nous n'avons pas encore étudier les résultats de ces manipulations.

IV / Programmation

Aujourd'hui nous avons une interface graphique regroupant tous les capteurs qui relève des données lorsque l'on lance une manipulation à partir d'une fenêtre de dialogue qui permet de rentrer différents paramètres, qui synchronise les capteurs et écrit les données dans un fichier sous format « .txt » ou « .csv ».
Nous avons maintenant deux capteurs de température étalonné et un spectromètre qui sont opérationnels avec des options tels que :

- Pouvoir modifier l'intervalle de l'axe des abscisses (Température (°C) ou Absorbance)
- Pouvoir modifier la vue de l'axe des ordonnées (Temps (seconde))
- Pouvoir créer des seuils modifiables qui envoie une notification d'alertes lorsqu'elles sont dépassées
- Pouvoir modifier les paramètres ci-dessus en mode « Grouper » ou « Séparer » (Uniquement pour la partie température)

Nous allons voir dans un premier temps tout ce qui est en rapport avec la température, après cela nous verrons la partie absorbance, suite à ceci nous verrons les différentes options citer en détails puis nous finirons par voir comment lancer une manipulation.

1°) Température

En reprenant le travail de l'année dernière nous avons premièrement intégrer le graphique dans la nouvelle interface qui est permis avec un GraphicsScene intégrer dans un GraphicsView.
La classe QGraphicsScene fournit une surface permettant de gérer un grand nombre d'éléments graphiques 2D et la classe QGraphicsView fournit un widget pour afficher le contenu d'un QGraphicsScene (Figure 6).

```
self.scene_1 = QGraphicsScene()
self.scene_1.addWidget(self.airGraphCanvas.figure.canvas)

self.scene_2 = QGraphicsScene()
self.scene_2.addWidget(self.tempeGraphCanvas.figure.canvas)

# Graphique Air
self.graphicsView_4 = self.ui.graphicsView_4
self.graphicsView_4.setScene(self.scene_1)
self.graphicsView_4.viewport().installEventFilter(self)
self.graphicsView_4.setContentsMargins(0, 0, 0, 0)
self.graphicsView_4.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
self.graphicsView_4.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOff)

# Graphique Electrolyte
self.tempeGraph = self.ui.tempeGraph
self.tempeGraph.setScene(self.scene_2)
self.tempeGraph.viewport().installEventFilter(self)
self.tempeGraph.setContentsMargins(0, 0, 0, 0)
self.tempeGraph.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
self.tempeGraph.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
```

Figure 6 : Programme permettant l'intégration des graphiques Matplotlib dans l'interface TechVolt

Nous avons rajouté un deuxième capteur de température car cela permet d'avoir un pont de comparaison entre la température de l'air et des électrolytes. Pour intégrer tous ces changements nous avons placer les différents éléments à l'aide du logiciel QtDesigner, voici le résultat à l'heure actuelle :

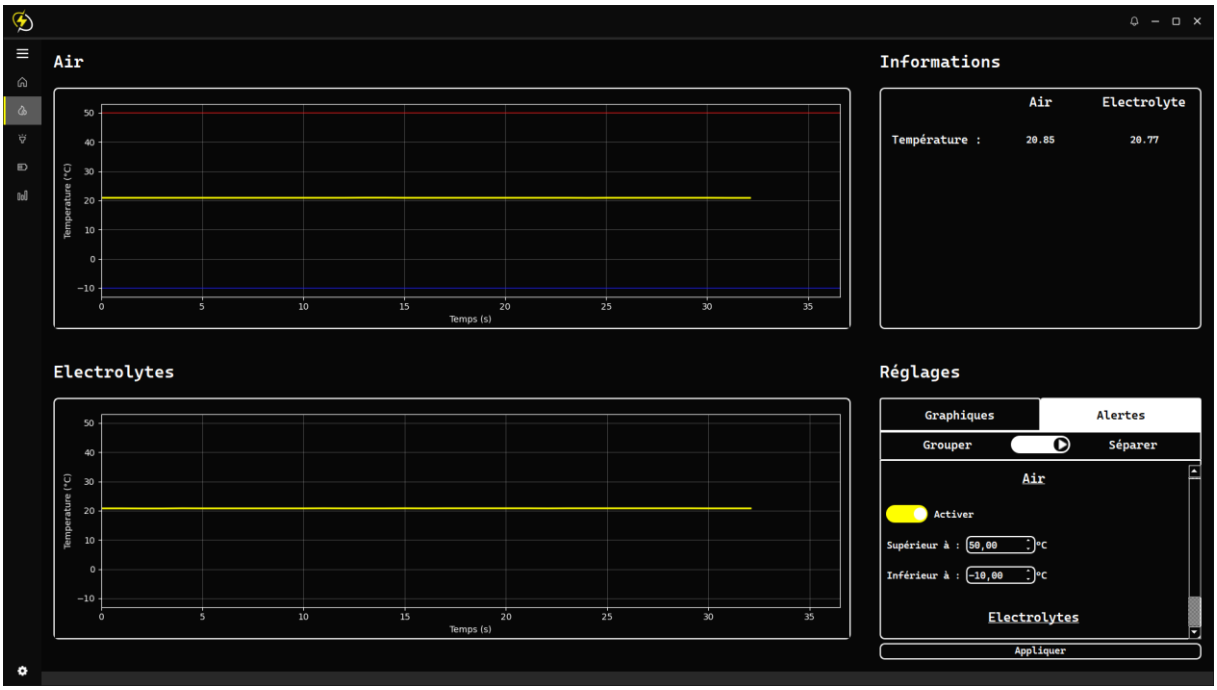


Figure 7 : Vue de l'interface graphique TechVolt pour la température

Comme nous pouvons le constater il y a maintenant deux graphiques Matplotlib, un pour la température de l'air et un autre pour la température des électrolytes qui trace la température en temps réel. Nous avons aussi directement la valeur des températures affichés dans l'onglet « Informations » et nous réfléchissons actuellement aux différentes données que nous pourrions rajouter tels que la moyenne, le maximum, le minimum et encore d'autres qui pourraient être utiles à l'utilisateur. Nous avons aussi un onglet « Réglages » avec un bouton « Appliquer » que nous détaillerons dans la partie dédiée à cela.

Pour envoyer les données des deux capteurs simultanément nous avons dû revoir le câblage du système et le programme Arduino. En effet le câblage ne permettait que d'envoyer des données pour un capteur et nous avons dû revoir le programme Arduino car ce dernier stocker les valeurs sur la carte Arduino et donc au bout d'un certain temps (qui dépends de l'intervalle de la prise de mesure) la mémoire saturée et plus aucune valeur n'était affichée.

Pour la partie câblage nous avons donc d'abord dédoubler l'Adafruit MAX31865 avec sonde Pt100 (Voir rapport 1ère année, page 25, figure 30) afin d'avoir un deuxième capteur.
Ensuite nous avons effectuée des recherches pour comprendre comment la communication d'informations entre la carte Arduino vers l'ordinateur s'effectue. Nous avons trouvé que nous devions utiliser le protocole dit SPI (Serial Peripheral Interface) qui est un protocole de communication série synchrone utilisé pour la communication entre plusieurs périphériques sur le même bus.

Voici le programme Arduino résultant de ces recherches :

```
#include <Adafruit_MAX31865.h>
Adafruit_MAX31865 mon_capteur1 = Adafruit_MAX31865(10);
Adafruit_MAX31865 mon_capteur2 = Adafruit_MAX31865(9);

void setup() {
  Serial.begin(9600);
  mon_capteur1.begin(MAX31865_2WIRE);
  mon_capteur2.begin(MAX31865_2WIRE);
}
void loop() {
  float R1 = mon_capteur1.lecture_resistance();
  float T1=(R1-100.31)/0.3795;

  float R2 = mon_capteur2.lecture_resistance();
  float T2=(R2-100.80)/0.3618;

  Serial.print ("Température Capteur 1: ");
  Serial.print(T1);
  Serial.println("°C");
  Serial.print ("Température Capteur 2: ");
  Serial.print(T2);
  Serial.println("°C");

  delay (1000);
}
```

Figure 8 : Programme Arduino permettant le fonctionnement des capteurs de température

On utilise d'abord la bibliothèque Adafruit_MAX31865 pour interagir avec deux capteurs MAX31865, qui mesurent la température en lisant la résistance de thermocouples.
Le protocole SPI utilise plusieurs lignes de communication, dont certaines sont partagées entre les périphériques connectés. Ces lignes comprennent généralement :

- MOSI (Master Out Slave In) : C'est la ligne par laquelle le maître envoie des données aux esclaves.
- MISO (Master In Slave Out) : C'est la ligne par laquelle les esclaves renvoient des données au maître.
- SCLK (Serial Clock) : C'est la ligne qui génère l'horloge pour synchroniser la communication.
- CS (Chip Select) : Cette ligne est utilisée pour sélectionner le périphérique avec lequel le maître souhaite communiquer.

Dans votre programme, chaque capteur est connecté à un port CS différent.
Nous avons :

```
Adafruit_MAX31865 mon_capteur1 = Adafruit_MAX31865(10);
Adafruit_MAX31865 mon_capteur2 = Adafruit_MAX31865(9);
```

La ligne CS est utilisée pour activer ou désactiver un périphérique spécifique sur le bus SPI. Lorsqu'un capteur est sélectionné (CS est actif), il est prêt à recevoir des commandes et à transmettre des données. Lorsqu'il n'est pas sélectionné, il est en mode veille et n'interfère pas avec les autres

périphériques sur le bus. En résumé, la raison pour laquelle chaque capteur a une ligne CS distincte est de permettre au microcontrôleur de choisir quel capteur il souhaite interroger à un moment donné. Chaque capteur est indépendant dans la communication, mais ils partagent les autres lignes SPI (MOSI, MISO, SCLK) pour économiser des broches et simplifier le câblage.

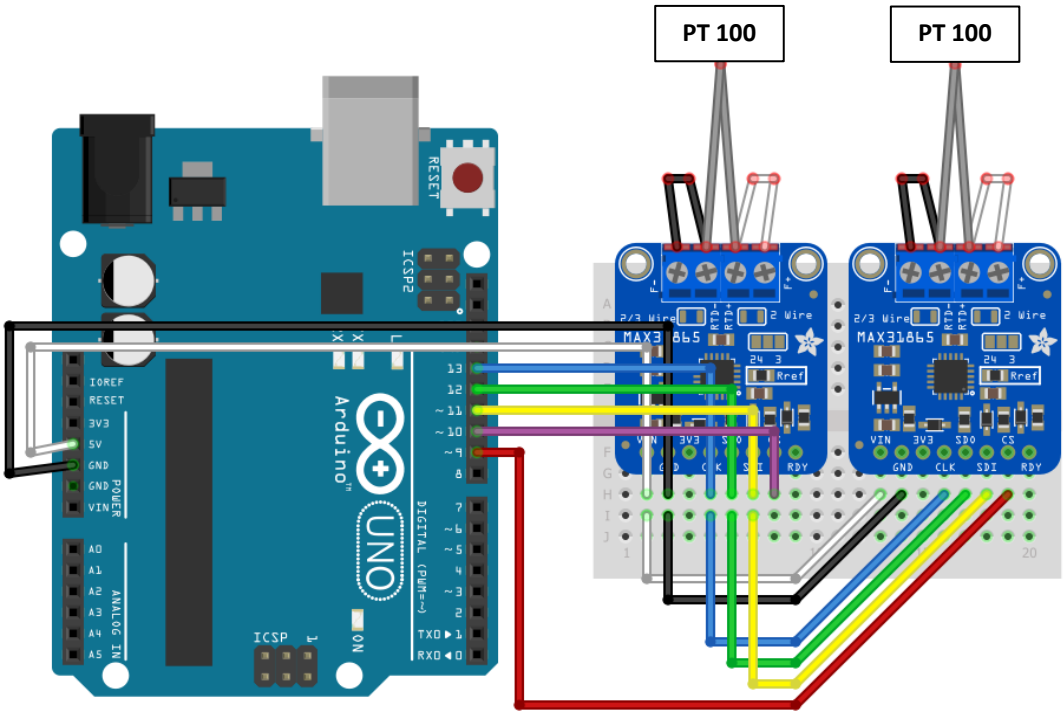


Figure 9 : Câblage des différents composants pour la partie température

Nous avons maintenant deux capteurs de température opérationnels qui envoies les données vers l'ordinateur afin de tracer la température en temps réelle. Pour ce qui est de l'étalonnage on réutilise la même méthode que l'année dernière.

2°) Absorbance

Pour la partie absorbance nous nous étions arrêtés au simple fonctionnement « brut » du spectromètre donc les objectifs étaient les mêmes que pour la température c'est-à-dire avoir un graphique qui trace en temps réel l'absorbance dans l'interface graphique TechVolt et de récupérer les données.

Notre spectromètre qui est un Flame-S-UV-VIS-ES de Ocean Insight (voir rapport 1ère année, page 26), lorsque l'on connecte ce dernier à l'ordinateur il est détecté directement.

Nous avons fait un programme permettant de tout d'abord détecter le spectromètre qui est connecté à l'ordinateur avec la fonction suivante :

```
def initialize_spectrometer(self):
    devices = sb.list_devices()
    if not devices:
        raise ValueError("Aucun spectromètre trouvé.")

    spectrometer = sb.Spectrometer(devices[0])
    spectrometer.integration_time_micros(500000)
    return spectrometer
```

Figure 10 : Programme permettant de détecter le spectromètre

La fonction cherche les spectromètres qui peuvent être connecter à l'ordinateur, s'il y en a un alors on renvois ce dernier à d'autres fonctions et sinon on indique qu'aucun spectromètre n'est détecté.

Avant de faire une manipulation il faut mesurer ce que l'on appelle un « Blanc » et un « Noir ». Voici ce que signifient généralement ces termes :

Blanc :

Définition : Le blanc est une mesure de référence effectuée sans échantillon. Cela implique de mesurer le signal de l'instrument lorsque la source de lumière ou de rayonnement passe à travers un espace vide ou un matériau transparent.
Utilité : Le blanc permet de corriger les effets de fond, tels que la lumière parasite ou le bruit électronique, présents dans le système de mesure. En soustrayant le blanc du spectre total (échantillon + instrument), on peut isoler le signal provenant uniquement de l'échantillon.

Noir :

Définition : Le noir est une mesure de référence effectuée lorsque la source de lumière ou de rayonnement est complètement bloquée, de sorte qu'aucun signal n'atteint le détecteur.

Utilité : Le noir permet de mesurer le bruit de fond maximal de l'instrument. En soustrayant cette mesure du spectre total, on peut identifier et corriger les variations indésirables du signal qui ne proviennent pas de l'échantillon.

Donc en bref le blanc et le noir en spectroscopie servent à éliminer les contributions indésirables au signal, à corriger les variations de l'instrument et à garantir la précision des mesures réalisées sur un échantillon donné.

Pour faire ces deux mesures nous avons ces deux fonctions qui mesurent l'intensité lumineuse et qu'ils les sauvegarde dans un fichier sous format «.pkl» afin de les réutiliser pour le calcul d'absorbance (Figure 11).

```python
def measure_white(self):
    spectrometer = self.initialize_spectrometer()

    self.measure_dialog = MeasureDialog()
    self.measure_dialog.exec()
    time.sleep(0.25)
    white_measurement = self.measure_intensity(spectrometer)

    measurements = {'white': white_measurement, 'black': None}
    self.save_measurement_to_pkl(measurements)

    spectrometer.close()
    return white_measurement

def measure_black(self):
    spectrometer = self.initialize_spectrometer()

    self.measure_dialog = MeasureDialog()
    self.measure_dialog.exec()
    time.sleep(0.25)
    black_measurement = self.measure_intensity(spectr

    measurements = self.load_measurement_from_pkl()
    measurements['black'] = black_measurement
    self.save_measurement_to_pkl(measurements)

    spectrometer.close()
    return black_measurement
```

Figure 11 : Programme permettant la mesure du blanc et du noir

Une fois que nous avons ces deux données nous pouvons lancer la manipulation afin de tracer l'absorbance en temps réel or le spectromètre ne relève pas directement l'absorbance, il relève une intensité lumineuse donc nous devons faire un calcul avant de tracer un point de mesure.

L'absorbance (A) en spectroscopie peut être calculée à l'aide de la formule suivante :

$$A = -log_{10}\left(\frac{I}{I_0}\right)$$

Où :

- A est l'absorbance
- I est l'intensité lumineuse mesurée à travers l'échantillon
- I_0 est l'intensité lumineuse de référence (le blanc)

La formule est basée sur le principe que l'absorbance est liée négativement au logarithme décimal du rapport entre l'intensité lumineuse traversant l'échantillon (I) et l'intensité lumineuse de référence (I_0). Plus l'absorbance est élevée, plus l'échantillon absorbe de lumière.

Mesure de l'intensité lumineuse de référence (I_0) :

Prendre une mesure de l'intensité lumineuse en l'absence d'échantillon, généralement en utilisant un blanc (par exemple, un espace vide ou un matériau transparent). C'est notre intensité de référence (I_0).

Mesure de l'intensité lumineuse à travers l'échantillon (I) :

On place notre échantillon dans le spectromètre et on mesure l'intensité lumineuse à travers lui.

Calcule de l'absorbance (A) :

On utilise la formule $A = -log_{10}\left(\frac{I}{I_0}\right)$ pour calculer l'absorbance en utilisant les valeurs mesurées.

Voici le programme qui en découle :

```python
def calculate_absorbance(self, I, I0, I_black=None):
    if I_black is not None:
        return -np.log10((I - I_black) / I0)
    else:
        return -np.log10(I / I0)

def read_spectrometer_start(self):
    measurements = self.load_measurement_from_pkl()
    white_measurement = measurements['white']
    black_measurement = measurements['black']

    self.spectrometer = self.initialize_spectrometer()
    self.thread = Thread(target=self.background_thread, args=(white_measurement, black_measurement))
    self.thread.start()

    while not self.isReceiving:
        time.sleep(0.1)

    print("Démarrage de la lecture du spectromètre.")

def background_thread(self, white_measurement, black_measurement):
    time.sleep(1.0)
    if self.spectrometer is not None:
        while self.isRun:
            intensity = self.measure_intensity(self.spectrometer)
            absorbance = self.calculate_absorbance(intensity, white_measurement, black_measurement)
            self.data3.append(absorbance)
            time.sleep(1)
            self.isReceiving = True
    else:
        print("Le spectromètre n'est pas correctement initialisé")
```

Figure 12 : Programme permettant la mesure de l'absorbance

Puis comme pour la température, il ne reste plus qu'à placer les éléments (GraphicsScene, GraphicsView, …) à l'aide de QtDesigner et nous avons notre fenêtre pour l'absorbance au complet qui permet de tracer l'absorbance en fonction du temps :

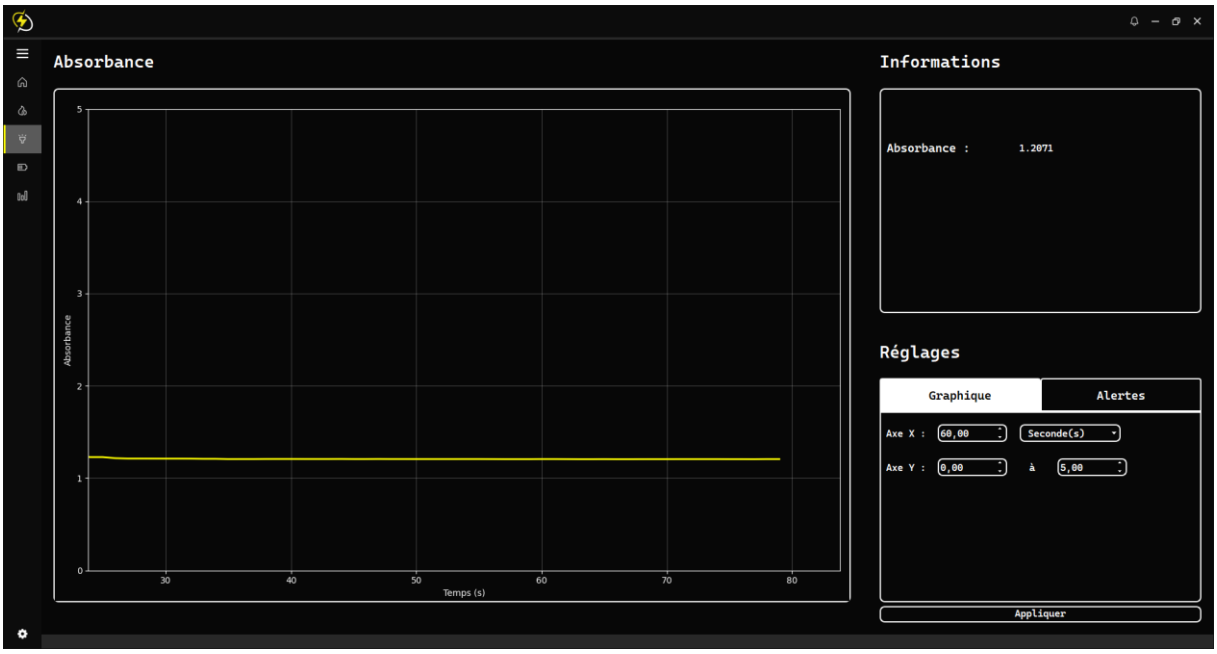


Figure 13 : Vue de l'interface graphique TechVolt pour l'absorbance

Nous avons maintenant un spectromètre opérationnel qui envoie des données vers l'ordinateur afin de tracer l'absorbance en temps réelle.

3°) Les options

Comme on peut le voir sur les figures 7 et 13 on remarque qu'il y a des onglets « Réglages ». L'onglet « Réglages » permet de faire plusieurs modifications et fonctionnalités :

Dans la partie « Graphique » nous avons, comme son nom l'indique toutes les modifications possibles relatives aux graphiques. Nous avons la possibilité de modifier la vue d'intervalle de temps de l'axe X en écrivant le chiffre souhaité (entre 1 et 60) et en choisissant l'ordre de grandeur (Secondes, Minutes, Heures ou Jours). Pour ce qui est de l'axe Y nous pouvons indiquer la vue dont nous souhaitons en indiquant les limites de vue donc par exemple à la figure X on remarque que l'on a mis l'axe Y de -13°C à 53°C et donc cela résulte sur le graphique de la vue de -13°C à 50°C (Figure 7).

Dans la partie « Alertes » nous avons la possibilité d'activer comme son nom l'indique des alertes qui permettent d'être notifier lorsque la valeur de la température et/ou de l'absorbance dépasse un seuil imposé par l'utilisateur représenter par des lignes rouge et bleues sur les graphiques (Figure 16). Nous avons la possibilité de les activer ou de les désactiver afin que dans certains cas d'expérience le canal de notifications ne soient pas trop encombrées avec des alertes inutiles.

Pour le cas de la température il y a un une option en plus qui est « Grouper » ou « Séparer », cette option est déterminée par le côté où se situe le rond blanc. Si le rond blanc est à gauche donc sur l'option « Grouper » alors on peut modifier les paramètres des graphiques Air et Electrolytes et/ou des alertes en même temps afin de ne pas être obliger à écrire deux fois la même chose et donc l'option « Séparer » permet d'avoir des paramètres différents pour chaque graphique que ce soit sur la vue ou sur les alertes.

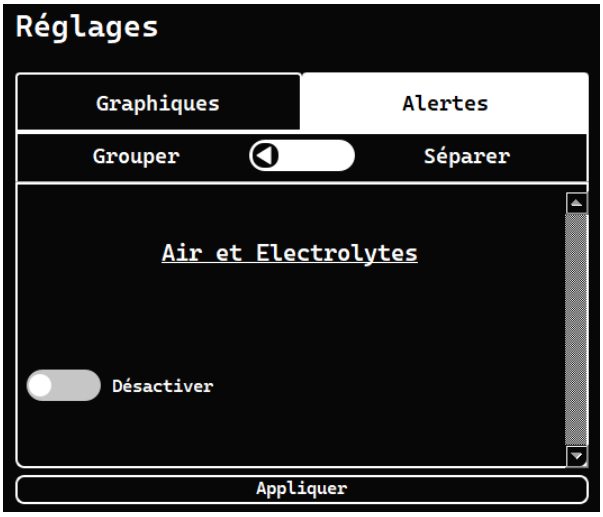


Figure 14 : Alertes de température désactiver

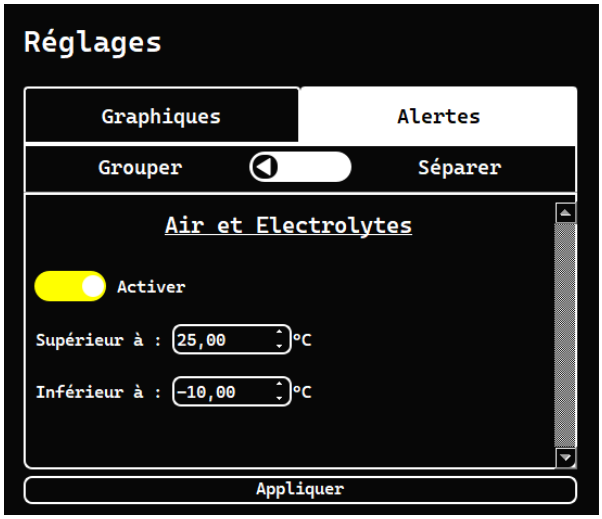


Figure 15 : Alertes de température désactiver

Figure 16 : Vue des graphiques avec les alertes de température activer

4°) Lancement d'une manipulation

Nous avons maintenant tous les capteurs qui sont opérationnels mais il faut que les mesures du blanc et du noir soient effectuées avant la manipulation, il faut également créer un fichier qui permet de sauvegarder toutes les données et enfin d'avoir un bouton qui lance tous les capteurs en même temps afin d'être synchroniser. Voici l'interface créer afin de répondre à toutes ces exigences :

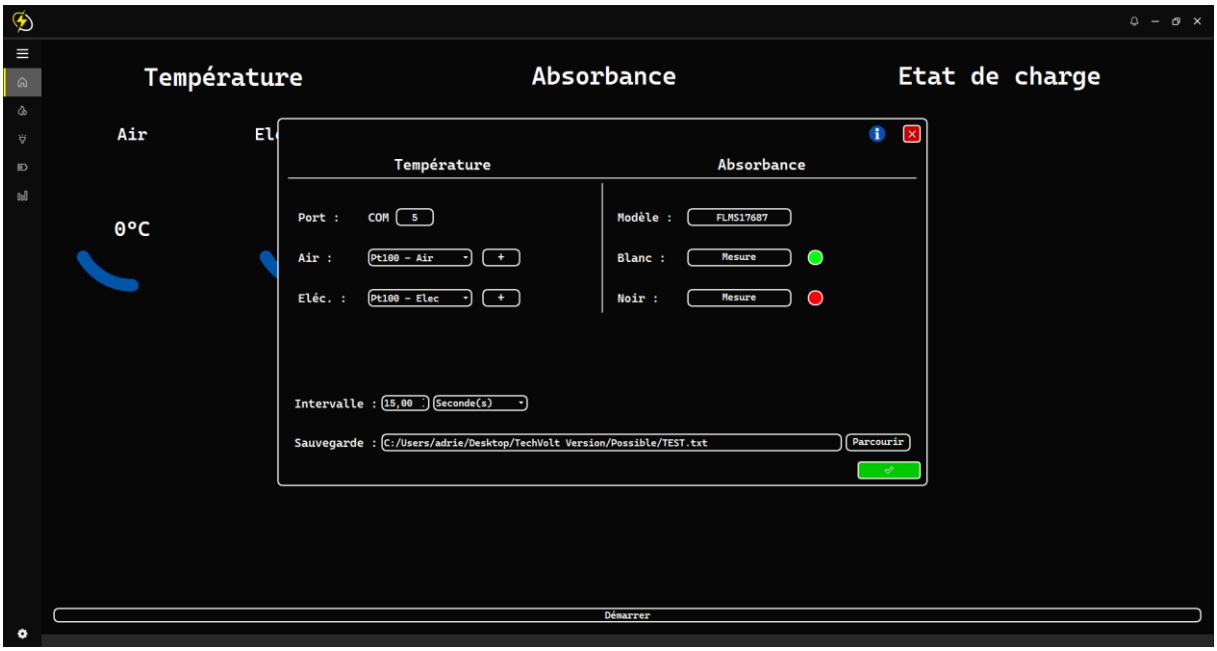


Figure 17 : Vue de la fenêtre permettant de lancer une manipulation

Dans la partie « Température » nous avons l'indication où l'Arduino est détecter (ici Port COM 5), cette détection se fait automatiquement comme la méthode pour le spectromètre. Dans les sections « Air » et « Eléc. » nous avons la possibilité de choisir les différents capteurs que nous pouvons rajouter avec la fenêtre de dialogue (Figure 18).

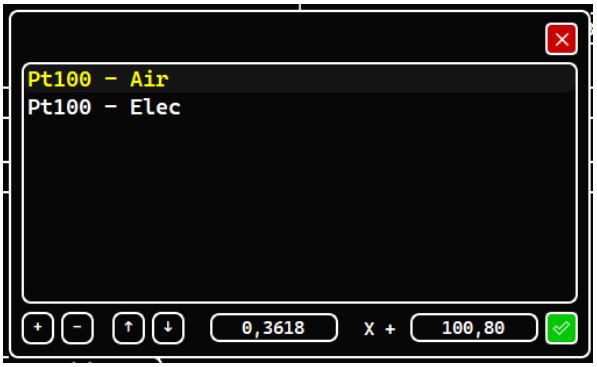


Figure 18 : Vue de la fenêtre permettant de configurer les capteurs

On peut y voir qu'on peut assigner une fonction linéaire à chaque capteur afin de modifier l'étalonnage du capteur si besoin, cela permet d'éviter de modifier directement le programme

Arduino. Malheureusement nous avons quelques difficultés à transmettre l'information vers la carte Arduino donc pour l'instant cette option ne fonctionne pas et les capteurs gardent leur étalonnage d'origine inscrit dans le programme Arduino.

Dans la partie « Absorbance » nous avons le modèle du spectromètre détecter qui est ici FLMS17687 qui est détecter automatiquement avec la méthode Figure X. Nous avons la mesure du blanc et du noir que l'on peut effectuer en cliquant sur les boutons assignés à ces derniers. Il y a des indicateurs à droite des boutons qui permettent de savoir si la mesure du blanc et/ou du noir a été correctement effectué, sur la figure X on remarque que la mesure du blanc a été faites mais pas celle du noir.

Enfin nous avons le chemin de sauvegarde qui s'écrit automatiquement si l'on clique sur le bouton « Parcourir » car cela ouvre une fenêtre d'explorateur de fichier Windows qui permet d'entrer un nom pour notre fichier et le format (.txt ou .csv).

Temps	Temperature Air	Temperature Electrolyte	Absorbance
1.02	19.82	20.61	0.7344
2.02	19.82	20.57	0.7344
3.01	19.82	20.61	0.7283
4.01	19.79	20.57	0.7268
5.03	19.79	20.57	0.7268
6.04	19.82	20.61	0.7205
7.04	19.82	20.57	0.7167
8.04	19.79	20.57	0.7167

Figure 19 : Données enregistrer dans un fichier .txt pendant une manipulation

Ici, les capteurs prennent une mesure toute les secondes mais on remarque qu'il y a de légères fluctuations, nous ne connaissons pas la cause exacte du problème dans le programme mais on peut supposer que le programme doit charger plusieurs fonctions à la fois (comme la fonction pour calculer l'absorbance) et donc il y a un temps de décalage qui se créer au fur et à mesure de la manipulation.

Pour lancer la manipulation il suffit de cliquer sur le bouton « ✓ » si tous les paramètres sont corrects mais si ce n'est pas le cas alors une alerte s'affiche afin d'indiquer à l'utilisateur ce qui ne va pas et ce qu'il doit faire avec une image pour savoir directement ce qui ne va pas, voici les différentes alertes possibles :

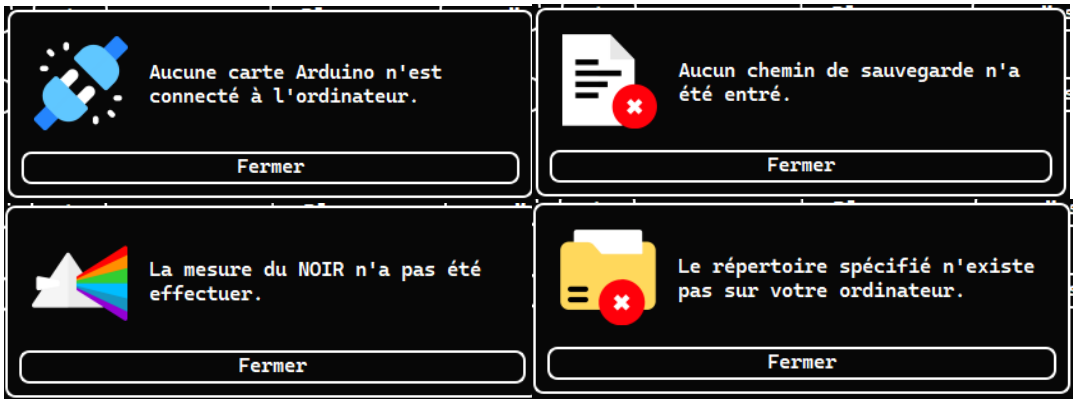


Figure 20 : Types d'erreur possibles lors du démarrage d'une manipulation

Pour résumer nous avons aujourd'hui la possibilité de lancer une manipulation avec deux capteurs de température et un spectromètre où leur donner sont enregistrer dans un fichier txt ou csv et qui sont tracer en temps réel.