# Acquisition de cartes spectrales en tension

**Corentin Morin (corentinmpro@outlook.com), Hiver 2022**

**Résumé**

Ce document sert d'appui aux éléments de codes et aux installations électroniques permettant de réaliser des acquisitions de cartes spectrales résolues en tension.

## Table des matières



# 1 Prérequis

## 1.1 Matériel

Le matériel utilisé pour mettre en place l'acquisition de cartes spectrale en tension est présenté en figure 1.1.1.

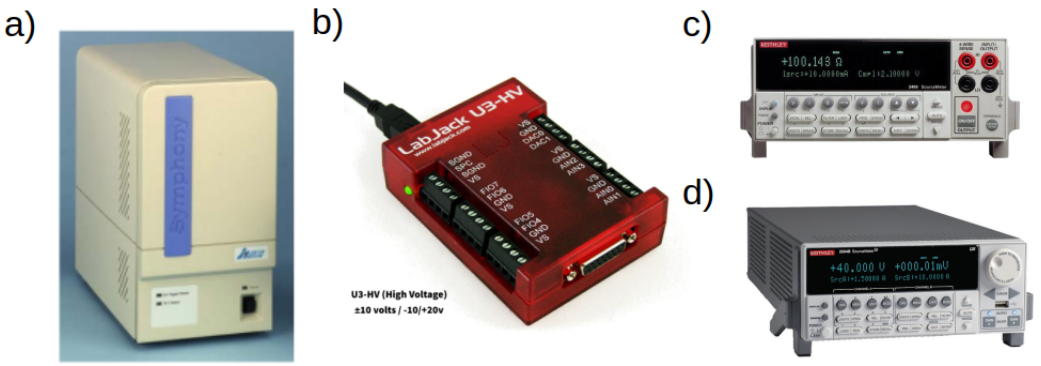


FIGURE 1.1.1 – a) Controlleur CCD Symphony, b) Module électronique LabJack, c)Keithley 2401, d) Keithley 2604

L'utilisateur aura le choix d'utiliser au choix le Keithley modèle 2604 ou le modèle 2401.

## 1.2 Connexions et électronique

Le LabJack est utilisé afin d'envoyer et de lire des signaux TTL au et venant du contrôleur Symphony.
Le port *DAC0* du LabJack est relié à l'entrée *TRIGGER INPUT* du Symphony. Le port *AIN0* du LabJack est relié à la sortie *TTL OUTPUT 1* du Symphony. (Voir schéma de la figure 1.2.1)

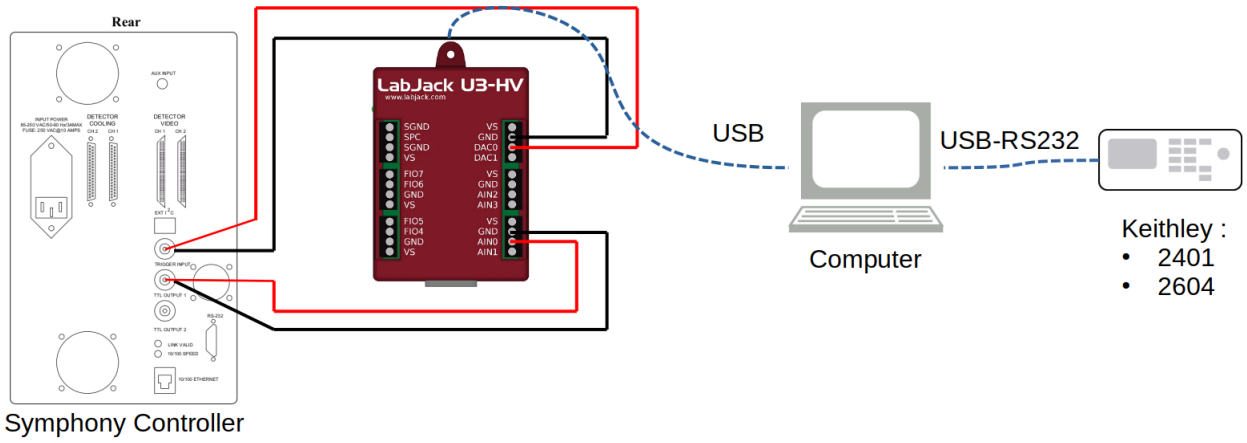


FIGURE 1.2.1 – Schéma des connexions entre les différents éléments de la manipulation

Le LabJack est connecté à l'ordinateur via un cable USB Type B - USB Type A. Le SourceMeter Keithley est relié à l'ordinateur pour un *Remote Control* via un RS232-USB Type A (Faire attention au fonctionnement du câble et au drive associé en allant vérifier la détection du cable dans l'outil *Gestionnaire de périphériques* de Windows, dans la section *Ports (COM & LPT)*.

## 1.3 Informatique

Les besoins informatiques sont les suivants :
- Une installation fonctionnelle de Windows
- Une installation fonctionnelle de python, de préférence, une installation dans un environnement (miniconda) dédié
- Installer les drivers pour le LabJack via ce lien
- Le module python *numpy*
- Le module python *matplotib*
- Le module python *pyvisa*, installation via :

  ```
  pip install -U pyvisa
  ```

- Le module python *LabJackPython* for UD, installation via :

  ```
  pip install LabJackPython
  ```

La communication avec le Keithley se fait grâce au module *pyvisa*, en langage *SCPI* pour le modèle 2401 et en langage *TSP* (voir partie 3).

# 2 Fonctionnement

Tous les codes sont disponibles ici : https://github.com/CorentinMorinSorbonne/spin_init/tree/main/Controle_Horiba/Spectras_Tension_Map_Spectro_final

## 2.1 Principe général

Dans l'application *Synergy*, dans l'onglet *Experiment Setup*, il faut charger un fichier d'expérience nommé *IV_Curve_Labjack.xml*. Ce fichier permet de configurer les triggers Hardware du contrôleur Symphony comme suit :
- L'entrée *TRIGGER INPUT* permet de déclencher une acquisition seule lors de la détection d'un front montant,
- La sortie *TTL OUTPUT 1* correspond à l'information *Experiment Status*, actif sur niveau logique haut. Dette donnée vaut toujours 1 sauf quand la caméra CCD est en train d'être lue. Ainsi, on sait que l'on peut déclencher une nouvelle acquition lorsqu'un front montant est détecté après un front descendant (processus total de lecture de la CCD).

De plus, dans les paramètres avancés, il a fallu décocher le contrôle total par le contrôleur Symphony.

Ensuite, il faut choisir le nombre d'acquisition que l'utilisateur souhaite prendre dans l'option *Accumulation Number* et conserver l'option *Stacking Spectras*. L'utilisateur peut ensuite régler toutes les informations importantes comme le temps d'exposition, la longueur d'onde centrale, la largeur de la fente du spectromètre, etc...

Suite à cela, il faut appuyer sur le bouton *Run*. Cela initialisera l'expérience et le contrôleur Symphony attendra un signa TTL extérieur pour commencer une mesure.

Ce signal est fourni grâce au LabJack contrôlé en Python. A chaque prise de spectre, le Keitlhey change de tension. L'attente d'un régime permanent peut être effectuée entre l'application d'une tension à l'échantillon et la prise d'un

spectre. A la fin, l'utilisateur obtient grâce à Python une courbe I-V dont les données sont enregistrées dans un fichier *.csv* et il obtient les données des spectres dans *Origin*.

## 2.2 Fonctionnement du code (Voir le code 3.1)

- Lignes 1 à 10 : Import des modules python
- Lignes 11 à 24 : Communication avec l'utilisateur. Il sera demandé à l'utilisateur : un chemin Windows où enregistrer le fichier *.csv* où seront stockées les données de la courbe I-V, un nom pour ce fichier, la tention minimale à appliqué, la tension maximale, le nombre de points à prendre, le temps entre l'application de la tension et la prise de mesure, le modèle du Keithley et si besoin le channel sur lequel est connecté l'échantillon.
- Lignes 25 à 28 : Comme le Keithley 2604 a 2 channels, dans les instructions, il faut préciser de quel channel il s'agit. Cela se fait en changeant de lettre à la fin du mot clé *smu*, d'où soit *smua* soit *smub*
- Lignes 40 à 75 : Ouverture des appareils. L'ouverture du LabJack est automatique avec la ligne **d=u3.U3()**. Cependant pour le Keithley, il faut préciser au module pyvisa avec quel appareil nous souhaitons communiquer. La communication avec le Keithley se fait via un port COM. Ce port comme dans le cas de la manipulation de photoluminescence est déterminé par le cable USB-RS232. Pour connaître le port COM, il suffit de se rendre dans l'utilitaire *Gestionnaire de périphériques* de Windows, dans la section *Ports (COM & LPT)* et trouver le cable. Si ce port vient à changer, il suffit de remplacer à chaque apparition *COM3* par *COM...* et *ASRL3 : :INSTR* par *ASRL... :INSTR*.
- Lignes 66 à 73 : Ces lignes permettent de savoir si la communication avec les appareils fonctionnent, si non, le programme s'arrêtera. Il faudra alors débugger le programme (alimentation et connexion du LabJack, port COM du Keithley, etc...)
- Lignes 83 à 123 : Ces lignes correspondent aux commandent pour le système avec le Keithley 2401. Tout d'abord, on s'assure qu'au début le Keitlhey est bien initialisé pour une source à 0 Volts, puis nous allumons la sortie. Ensuite, une tension souhaitée est appliquée à l'échantillon. Un cerain temps défini par l'utilisateur est attendu. Ensuite, une mesure de courant est réalisée. Puis, le LabJack envoie un front montant au contrôleur Symphony. Puis, le LabJack attend que la caméra CCD soit lue, ce qui signifie que l'exposition est finie. Une nouvelle consigne de tension est envoyée au Keitlhey et ainsi de suite....
- Lignes 124 à 170 : Mêmes consignes que précédemment mais pour le Keithley 2604 .
- Ligne 171 : Quand toutes les tensions ont été appliquée, la sortie du Keithley est désactivée.
- Lignes 175 à 194 : Affichage et enregistrement de la courbe I-V
- Lignes 196 à 204 : Un exemple de code python à utiliser pour traiter le fichier généré juste avant.

## 2.3 Code sans contrôle du spectromètre

En supprimant les lignes correspondant au LabJack, il est possible d'avoir un programme ne réalisant que des courbes I-V. (Voir le code 3.2)

# 3 Codes

## 3.1 Tension map with spectra

```
import pyvisa as visa
import numpy as np
import time
import matplotlib.pyplot as plt
import sys
import csv
from os.path import exists
import u3

#%% /!\ A remplir:

input("Welcome to the program allowing you to take IV curve and spectral datas. Please open synergy, then in the Experiment Setup panel load the IV_Curve_Labjack.xml experiment file. Choose your experimental details (dont forget to modify the accumulation number to the number of samples you want to take) then press Run in Synergy. When done press enter here.")

my_path=input("Enter a path (without the \ at the end) where to record the datas of IV Curve: ")
info_datas=input("Enter a name for the actual experiment: ")
V_deb=float(input("Enter the minimum tension to apply (in Volts): ")) # Tension de d2but de mesure
V_fin=float(input("Enter the maximum tension to apply (in Volts): ")) #Tension de fin de mesure
N_ech=int(input("Enter the number of samples to get (make it match with the Synergy accumulation number): ")) # Nomnre de mesures a prendre
my_time=float(input("Time to wait between the application of the tension and the data recording (in seconds): ")) # Nomnre de mesures a prendre
keithley_model=float(input("Please enter 1 for the Keithley model 2401 or 2 for the Keithley model 2604 then press enter: ")) #1=2401 et 2=2604

if keithley_model==2 :
    channel_meas=input("Enter the letter of the channel you are using A or B (in capital letter) then press enter: ") #A ou B

    if (channel_meas=='A'):
        source_str='smua'
    elif (channel_meas=='B'):
        source_str='smub'
    else:
        print("Unknown channel")
        input("Program will abort, please press enter")
        sys.exit()
elif keithley_model==1 :
    pass
else:
    print("Unknown Keithley")
    input("Program will abort, please press enter")
    sys.exit()

print("Initializing the program")

#%% Ouverture du Keithley et du LabJack
try :
    #Ouverture du LABJACK pour le spectro
    #Le DAC0 envoie l'impulsion pour commencer l'acquisition d'une donnée
    #AIN0 sert a checker le moment actuel de l'acquisiton
    d = u3.U3() #Ouverture du LabJack U3-HV


    rm = visa.ResourceManager()
    # print (rm.list_resources())
```

```
52
53      if keithley_model==1:
54          with rm. open_resource('COM3') as Keithley:
55              Keithley.port = 'COM3'
56              Keithley.baudrate = 9600
57              Keithley.timeout = 25000
58              Keithley.open()
59              Keithley.read_termination = '\r'
60              Keithley.write_termination = '\r'
61      elif  keithley_model==2:
62          Keithley = rm.open_resource('ASRL3::INSTR')
63
64
65
66      Keithley.write("*RST")
67      Keithley.write("*IDN?")
68      Q=Keithley.read()
69      print("Devices opened.")
70  except:
71      print("Couldn't open the LabJAck or the Keithley chexk the COM port and go to the python file
        ↪  to check the working of the program")
72      input("Press enter to end the program")
73      sys.exit()
74
75  #%% Boucle pour prendre les mesures de tensions
76
77  print("Beginnig data taking")
78
79  volt=np.linspace(V_deb,V_fin,N_ech) #Definition de toutes les tensions a explorer
80  res_volt=[] #Vecteur qui contiendra la tension appliquee
81  res_amp=[] #Vecteur qui contiendra le courant mesure
82
83  if keithley_model==1:
84      Keithley.write(":SOUR:VOLT:LEV:IMM:AMPL 0") #Mise a OV de la tension pour etre sur
85      Keithley.write(":OUTP ON")    #Allumage de loutput
86
87      for i in range(len(volt)): #Boucle pour les mesures
88          print("Taking datas for "+str(volt[i])+"V")
89
90          Keithley.write(":SOUR:VOLT:LEV:IMM:AMPL "+str(volt[i]))  #Application de la tension
91          time.sleep(my_time) #Waiting for a permanent regime
92          Keithley.write(":MEAS:CURR:DC?") #Commande de mesure
93          Q=Keithley.read() #La mesure est un tableau, la premiere case est la tension, la seconde
            ↪  le courant
94          A=np.array(Q.split(','))
95          A=A.astype('float64')
96          cur_volt=A[0] #Stockage des donnees
97          cur_amp=A[1]
98          res_volt.append(cur_volt)#Stockage des donnees
99          res_amp.append(cur_amp)
100     #### Code pour gérer acquisition spectro
101
102         DAC0_VALUE = d.voltageToDACBits(4.5, dacNumber = 0, is16Bits = False)
103         d.getFeedback(u3.DAC0_8(DAC0_VALUE))
104         time.sleep(0.1)
105
106         DAC0_VALUE = d.voltageToDACBits(0, dacNumber = 0, is16Bits = False)
107         d.getFeedback(u3.DAC0_8(DAC0_VALUE))
108
109         ainValue = d.getAIN(0)#Lecture de la valeur de la sortie du symphony. Quand elle sera à
            ↪  0, cela voudra dire que la CCD est en train detre lue
110
111         if (volt[i]!=volt[-1]):
```

```
            while (ainValue>2): #Attente d'un front descendant (Lecture de la CCD)
                ainValue = d.getAIN(0)
            while (ainValue<2): #Attente d'un front montant (Expérience prete a tourner)
                ainValue = d.getAIN(0)
        else:
            while (ainValue>2): #Attente d'un front descendant (Lecture de la CCD)
                ainValue = d.getAIN(0)
    ######################################################################
        print("Datas for "+str(volt[i])+"V taken")

    Keithley.write(":SOUR:VOLT:LEV:IMM:AMPL 0")

elif  keithley_model==2:

    Keithley.write(source_str+".source.func="+source_str+".OUTPUT_DCVOLTS")
    Keithley.write(source_str+".source.levelv=0")
    Keithley.write(source_str+".source.output ="+ source_str+".OUTPUT_ON")


    for i in range(len(volt)): #Boucle pour les mesures
        print("Taking datas for "+str(volt[i])+"V")

        Keithley.write(source_str+".source.levelv="+str(volt[i]))  #Application de la tension
        time.sleep(my_time) #Waiting for a permanent regime
        Keithley.write('print('+source_str+".measure.i())") #Commande de mesure
        Q=Keithley.read() #La mesure est un tableau, la premiere case est la tension, la seconde
        ↪  le courant
        A=float((Q.replace('\n','')))
        cur_amp=A

        Keithley.write('print('+source_str+".measure.v())") #Commande de mesure
        Q=Keithley.read() #La mesure est un tableau, la premiere case est la tension, la seconde
        ↪  le courant
        A=float((Q.replace('\n','')))
        cur_volt=A

        res_volt.append(cur_volt)#Stockage des donnees
        res_amp.append(cur_amp)
    #### Code pour gérer acquisition spectro

        DAC0_VALUE = d.voltageToDACBits(4.5, dacNumber = 0, is16Bits = False)
        d.getFeedback(u3.DAC0_8(DAC0_VALUE))
        time.sleep(0.1)

        DAC0_VALUE = d.voltageToDACBits(0, dacNumber = 0, is16Bits = False)
        d.getFeedback(u3.DAC0_8(DAC0_VALUE))

        ainValue = d.getAIN(0)#Lecture de la valeur de la sortie du symphony. Quand elle sera à
        ↪  0, cela voudra dire que la CCD est en train detre lue

        if (volt[i]!=volt[-1]):
            while (ainValue>2): #Attente d'un front descendant (Lecture de la CCD)
                ainValue = d.getAIN(0)
            while (ainValue<2): #Attente d'un front montant (Expérience prete a tourner)
                ainValue = d.getAIN(0)
        else:
            while (ainValue>2): #Attente d'un front descendant (Lecture de la CCD)
                ainValue = d.getAIN(0)
    ######################################################################
        print("Datas for "+str(volt[i])+"V taken")


    Keithley.write(source_str+".source.output ="+ source_str+".OUTPUT_OFF")
```

```
172
173
174
175 plt.plot(res_volt,res_amp)
176 plt.xlabel("Tension en V")
177 plt.ylabel("Courant en A")
178 print("An IV curve will be plotted, close it to end the program")
179 plt.show()
180
181 #%% Enregistrement des donnees
182 rows=[res_volt,res_amp] #Donnees a enregistrer
183 file_name=my_path+'\IV_data_'+info_datas #Nom du fichier
184 file_name_new=file_name
185
186 n=1
187 while  exists(file_name_new+'.csv')==True : #Boucle pour ne pas effacer de donnees
188     file_name_new=file_name+'_'+str(n)
189     n+=1
190 file_name=file_name_new+'.csv'
191 with open(file_name, 'w') as f: #Enregistrement
192     write = csv.writer(f, delimiter=' ',
193                                quotechar='|', quoting=csv.QUOTE_MINIMAL)
194     write.writerows(rows)
195
196 #%% Pour lire les donnees
197 # data=[]
198 # with open(file_name, newline='') as csvfile:
199 #     data_read = csv.reader(csvfile, delimiter=',', quotechar='|')
200 #     for row in data_read:
201 #         if len(row)>0:
202 #             data.append(row[0].split(' '))
203 # data=np.array(data).astype('float64') #Donn2es exploitables
204 #%%
205
206 Keithley.close()
207 print("Experiment over, please make sure to reload the DefaultExp.xml experiment file in Synergy
    ↪  the next time you use it")
208 input("Press enter to end the program")
```

## 3.2 I-V curve

```
1 import pyvisa as visa
2 import numpy as np
3 import time
4 import matplotlib.pyplot as plt
5 import sys
6 import csv
7 from os.path import exists
8
9 #%% /!\ A remplir:
10
11 input("Welcome to the program allowing you to take an IV curve. Please, press enter here.")
12
13 my_path=input("Enter a path (without the \ at the end) where to record the datas of IV Curve: ")
14 info_datas=input("Enter a name for the actual experiment: ")
15 V_deb=float(input("Enter the minimum tension to apply (in Volts): ")) # Tension de d2but de
    ↪  mesure
16 V_fin=float(input("Enter the maximum tension to apply (in Volts): ")) #Tension de fin de mesure
17 N_ech=int(input("Enter the number of samples to get (make it match with the Synergy accumulation
    ↪  number): ")) # Nomnre de mesures a prendre
18 my_time=float(input("Time to wait between the application of the tension and the data recording
    ↪  (in seconds): ")) # Nomnre de mesures a prendre
```

```
keithley_model=float(input("Please enter 1 for the Keithley model 2401 or 2 for the Keithley model 2604 then press enter: ")) #1=2401 et 2=2604

if keithley_model==2 :
    channel_meas=input("Enter the letter of the channel you are using A or B (in capital letter) then press enter: ") #A ou B

    if (channel_meas=='A'):
        source_str='smua'
    elif (channel_meas=='B'):
        source_str='smub'
    else:
        print("Unknown channel")
        input("Program will abort, please press enter")
        sys.exit()
elif keithley_model==1 :
    pass
else:
    print("Unknown Keithley")
    input("Program will abort, please press enter")
    sys.exit()

print("Initializing the program")

#%% Ouverture du Keithley
try :

    rm = visa.ResourceManager()

    if keithley_model==1:
        with rm. open_resource('COM3') as Keithley:
            Keithley.port = 'COM3'
            Keithley.baudrate = 9600
            Keithley.timeout = 25000
            Keithley.open()
            Keithley.read_termination = '\r'
            Keithley.write_termination = '\r'
    elif  keithley_model==2:
        Keithley = rm.open_resource('ASRL3::INSTR')



    Keithley.write("*RST")
    Keithley.write("*IDN?")
    Q=Keithley.read()
    print("Devices opened.")
except:
    print("Couldn't open the Keithley chexk the COM port and go to the python file to check the working of the program")
    input("Press enter to end the program")
    sys.exit()

#%% Boucle pour prendre les mesures de tensions

print("Beginnig data taking")

volt=np.linspace(V_deb,V_fin,N_ech) #Definition de toutes les tensions a explorer
res_volt=[] #Vecteur qui contiendra la tension appliquee
res_amp=[] #Vecteur qui contiendra le courant mesure

if keithley_model==1:
    Keithley.write(":SOUR:VOLT:LEV:IMM:AMPL 0") #Mise a 0V de la tension pour etre sur
    Keithley.write(":OUTP ON")    #Allumage de loutput
```

```
79
80      for i in range(len(volt)): #Boucle pour les mesures
81          print("Taking datas for "+str(volt[i])+"V")
82
83          Keithley.write(":SOUR:VOLT:LEV:IMM:AMPL "+str(volt[i]))  #Application de la tension
84          time.sleep(my_time) #Waiting for a permanent regime
85          Keithley.write(":MEAS:CURR:DC?") #Commande de mesure
86          Q=Keithley.read() #La mesure est un tableau, la premiere case est la tension, la seconde
            ↪  le courant
87          A=np.array(Q.split(','))
88          A=A.astype('float64')
89          cur_volt=A[0] #Stockage des donnees
90          cur_amp=A[1]
91          res_volt.append(cur_volt)#Stockage des donnees
92          res_amp.append(cur_amp)
93
94          print("Datas for "+str(volt[i])+"V taken")
95
96      Keithley.write(":SOUR:VOLT:LEV:IMM:AMPL 0")
97
98  elif  keithley_model==2:
99
100     Keithley.write(source_str+".source.func="+source_str+".OUTPUT_DCVOLTS")
101     Keithley.write(source_str+".source.levelv=0")
102     Keithley.write(source_str+".source.output ="+ source_str+".OUTPUT_ON")
103
104
105     for i in range(len(volt)): #Boucle pour les mesures
106         print("Taking datas for "+str(volt[i])+"V")
107
108         Keithley.write(source_str+".source.levelv="+str(volt[i]))  #Application de la tension
109         time.sleep(my_time) #Waiting for a permanent regime
110         Keithley.write('print('+source_str+".measure.i())") #Commande de mesure
111         Q=Keithley.read() #La mesure est un tableau, la premiere case est la tension, la seconde
            ↪  le courant
112         A=float((Q.replace('\n','')))
113         cur_amp=A
114
115         Keithley.write('print('+source_str+".measure.v())") #Commande de mesure
116         Q=Keithley.read() #La mesure est un tableau, la premiere case est la tension, la seconde
            ↪  le courant
117         A=float((Q.replace('\n','')))
118         cur_volt=A
119
120         res_volt.append(cur_volt)#Stockage des donnees
121         res_amp.append(cur_amp)
122
123         print("Datas for "+str(volt[i])+"V taken")
124
125
126     Keithley.write(source_str+".source.output ="+ source_str+".OUTPUT_OFF")
127
128
129
130 plt.plot(res_volt,res_amp)
131 plt.xlabel("Tension en V")
132 plt.ylabel("Courant en A")
133 print("An IV curve will be plotted, close it to end the program")
134 plt.show()
135
136 #%% Enregistrement des donnees
137 rows=[res_volt,res_amp] #Donnees a enregistrer
138 file_name=my_path+'\IV_data_'+info_datas #Nom du fichier
```

```
file_name_new=file_name

n=1
while  exists(file_name_new+'.csv')==True : #Boucle pour ne pas effacer de donnees
    file_name_new=file_name+'_'+str(n)
    n+=1
file_name=file_name_new+'.csv'
with open(file_name, 'w') as f: #Enregistrement
    write = csv.writer(f, delimiter=' ',
                           quotechar='|', quoting=csv.QUOTE_MINIMAL)
    write.writerows(rows)

#%% Pour lire les donnees
# data=[]
# with open(file_name, newline='') as csvfile:
#     data_read = csv.reader(csvfile, delimiter=',', quotechar='|')
#     for row in data_read:
#         if len(row)>0:
#             data.append(row[0].split(' '))
# data=np.array(data).astype('float64') #Donn2es exploitables
#%%

Keithley.close()
print("Experiment over, please make sure to reload the DefaultExp.xml experiment file in Synergy
↪  the next time you use it")
input("Press enter to end the program")
```