Potentiostat

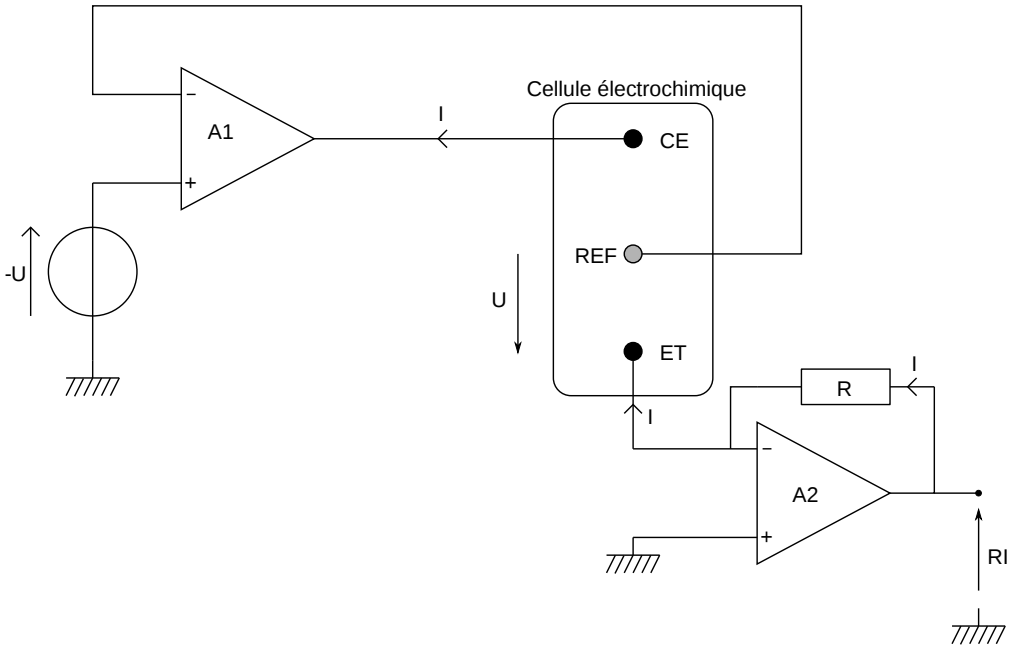
1. Introduction

Ce document présente la réalisation d'un potentiostat pour l'utilisation d'un montage électrochimique à trois électrodes. Le potentiel entre l'électrode de travail et l'électrode de référence est imposé par une boucle de rétroaction analogique. L'interface numérique est réalisée avec une carte d'acquisition SysamSP5 et le module d'interface pour python pycanum. La commande du potentiostat et l'obtention des courbes courant-potentiel se fait avec un programme écrit en python.

Le potentiostat possède deux plages de mesure de courant, la première allant jusqu'à 10 mA, la seconde jusqu'à 100 mA.

2. Principe

Le potentiostat analogique est constitué d'un amplificateur différentiel A1 qui fournit le courant à la contre-électrode (CE), l'électrode de travail (ERT) étant maintenue à la masse. Une rétroaction négative via l'électrode de référence (EREF) permet d'appliquer le potentiel U de consigne entre l'électrode de travail et l'électrode de référence (égal à E-Eref).



L'entrée inverseuse de l'amplificateur doit prélever un courant très faible à l'électrode de référence. On utilise pour cela un amplificateur à transistor FET, dont la résistance d'entrée est de l'ordre de 10^{12} Ω, ce qui conduit à un courant de l'ordre du pico-ampère.

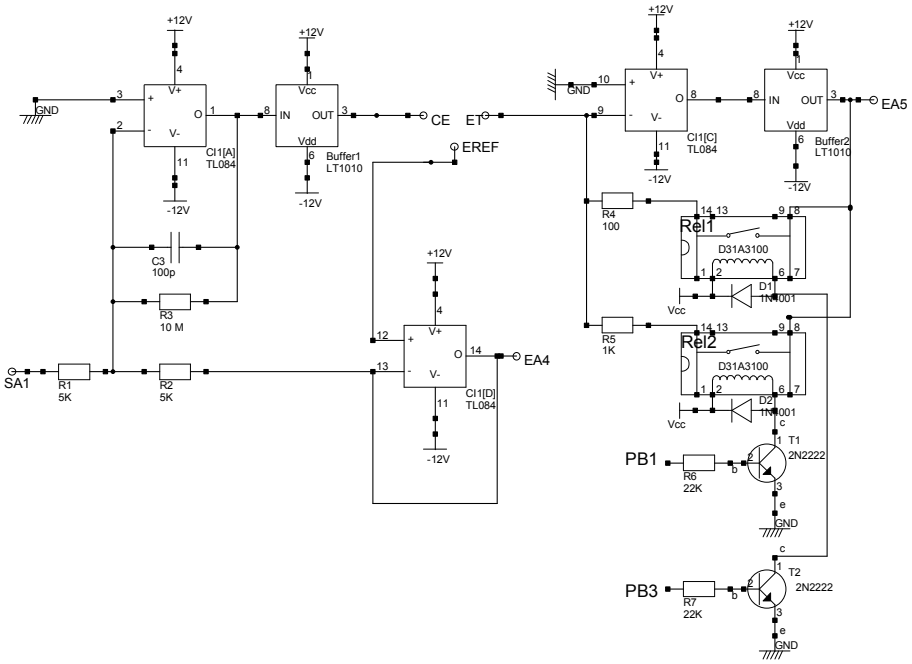
La mesure du courant se fait derrière l'électrode de travail, avec un convertisseur courant-tension A2 qui permet de maintenir le potentiel de l'électrode à la masse.

3. Réalisation

Le courant I doit atteindre 100 mA. Un amplificateur à FET de type TL081 ne permet pas de dépasser 20 mA. On doit donc ajouter un étage de puissance à la sortie des deux amplificateurs du circuit. Nous avons opté pour un tampon de puissance LT1010, qui permet d'atteindre 150 mA. Un tampon est ajouté dans la boucle de rétroaction de l'amplificateur de commande et dans celle du convertisseur courant-tension.

Les amplificateurs et les tampons sont alimentés par l'alimentation double $-12/0/12\ V$ de la carte Sysam SP5.

Le convertisseur courant-tension a deux résistances R possibles. La première est $R = 1\ k\Omega$ et permet d'atteindre un courant de 10 mA. La seconde est $R = 100\ \Omega$ et permet d'atteindre un courant de 100 mA. Celle-ci doit pouvoir dissiper 1 W. La mise en circuit de chacune de ces deux résistances se fait avec un relai Celduc D31A3100.

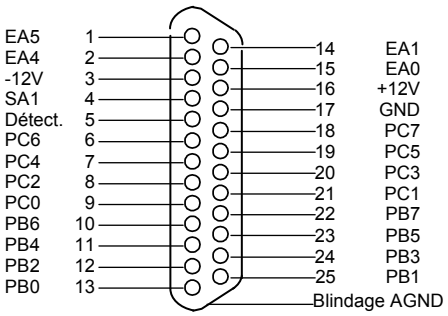


Le relai qui met la résistance $R = 1\ k\Omega$ en circuit est piloté par la sortie numérique PB1 de la carte SysamSP5, via un transistor NPN 2N2222. La résistance $R = 100\ \Omega$ est mise en circuit par la sortie PB3. En position de repos, les deux relais sont ouverts et aucun courant ne circule dans la cellule.

La tension de commande est appliquée par la sortie SA1 de la carte SysamSP5. L'amplificateur d'entrée est un amplificateur inverseur de gain unité. La tension de commande appliquée sur SA1 est donc le potentiel U=E-Eref de consigne, que l'on retrouve en principe entre l'électrode de travail et l'électrode de référence.

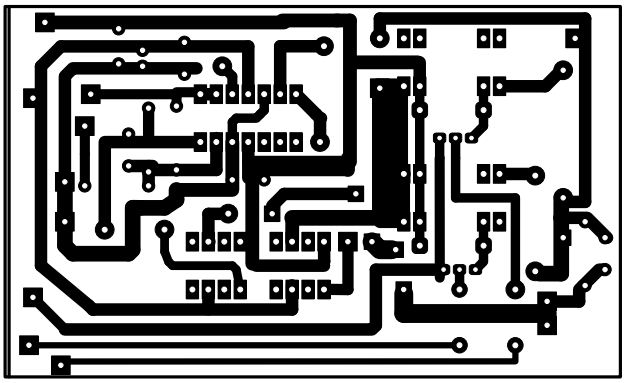
L'entrée analogique EA4 est utilisée pour lire le potentiel de l'électrode de référence par rapport à la masse, c'est-à-dire l'opposé du potentiel E-Eref (car l'électrode de travail est à la masse). L'entrée analogique EA5 est utilisée pour lire la tension RI, dont on déduit le courant I circulant dans la cellule.

Les bornes +12V,-12V,GND,AGND,SA1,PB1,PB3,EA4,EA5 sont prises sur le connecteur subD25 femelle de la carte SysamSP5, dont la figure suivante montre une vue depuis l'extérieur de la carte :



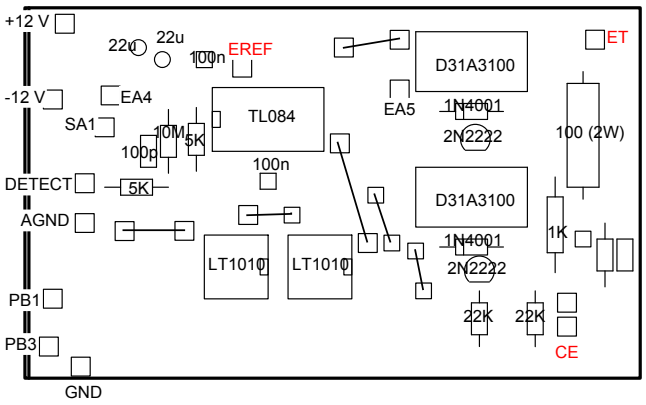
Pour que les entrées-sorties soient accessibles sur ce connecteur, il faut relier la borne DETECT à la masse AGND. Cela a pour effet de fermer un relai qui déconnecte les bornes du dessus de la carte au profit de celles du connecteur subD25.

Voici le typon du circuit imprimé, de taille 98x59 mm :

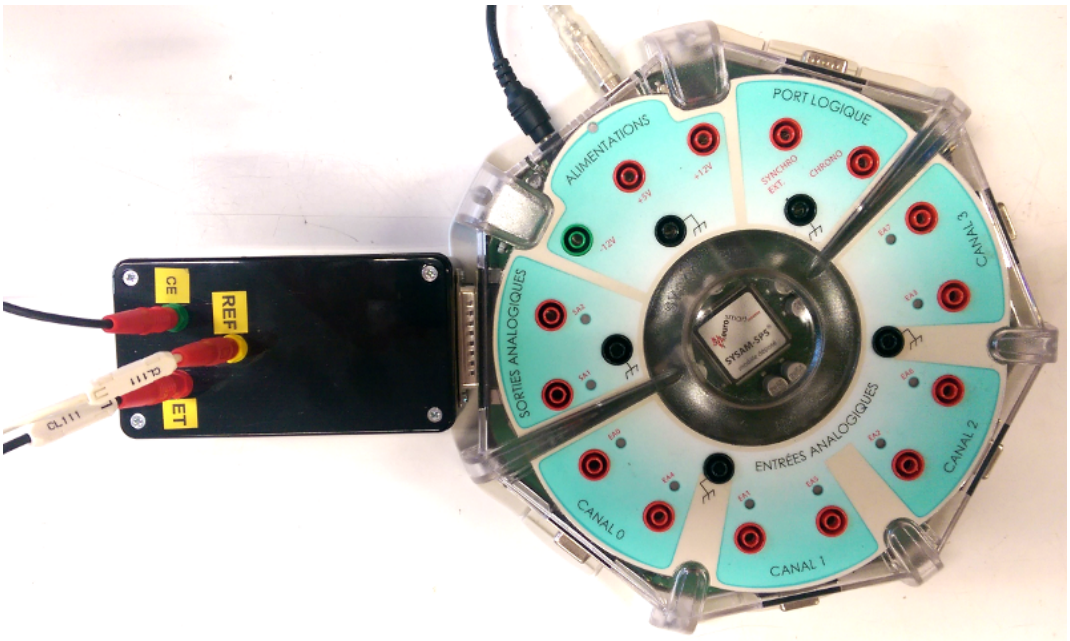


Fichier PDF pour l'impression : potentiostat-cuivre.pdf.

Voici l'implantation des composants et les bornes sur lesquelles les fils doivent être soudés, d'une part vers le connecteur subD25, d'autre part vers les trois bornes CE, ET et EREF :



Le circuit imprimé est logé dans un boitier de taille 120x65x40 mm (Camdenboss BIM2004/14-BLK/BLK), connecté à la carte SysamSP5 par un connecteur subD25 mâle vissé sur le boitier. Trois bornes permettent de brancher les trois électrodes.



4. Programme de commande

4.a. Interface de programmation

La commande du potentiostat se fait depuis un script python au moyen de l'interface de programmation implémentée dans la classe `Potentiostat`. Celle-ci nécessite le module pycanum, qui permet d'accéder à la carte SysamSP5.

potentiostatSysamSP5.py

```
import pycanum.main as pycan
import time
import numpy
```

Le constructeur définit les valeurs des résistances, les ports utilisés et le potentiel de l'électrode de référence. La liaison avec la carte est établie puis les relais sont ouverts.

```
class Potentiostat:
    def __init__(self):
        self.gain_entree = 1.0
        self.sys = pycan.Sysam("SP5")
        self.sys.ecrire(1,0,0,0)
        self.R1 = 1000.0 # calibre 10 mA
        self.R2 = 100.0 # calibre 100 mA
        self.pb1 = 1
        self.pb2 = 3
        self.ui = 5
        self.ue = 4
        self.Eref = 0.25 # electrode ECS
        self.sys.portB_config(0,1)
        self.sys.portB_ecrire(self.pb1,0)
        self.sys.portB_ecrire(self.pb2,0)
        self.sys.config_entrees([self.ue,self.ui],[10,10])
        self.liste_E = []
        self.liste_I = []
```

Lorsque l'utilisation du potentiostat est terminée, on doit fermer l'interface avec la fonction suivante :

```
    def fermer(self):
        self.sys.fermer()
```

La fonction suivante permet de fermer le circuit avec le calibre 1 (10 mA) ou le calibre 2 (100 mA).

```
    def fermer_circuit(self,calibre):
        self.sys.portB_config(0,1)
        if calibre==1:
            self.R = self.R1
```

```
            self.sys.portB_ecrire(self.pb1,1)
            self.sys.portB_ecrire(self.pb2,0)
        elif calibre==2:
            self.R = self.R2
            self.sys.portB_ecrire(self.pb1,0)
            self.sys.portB_ecrire(self.pb2,1)
```

La fonction suivante ouvre le circuit en ouvrant les deux relais :

```
    def ouvrir_circuit(self):
        self.sys.portB_ecrire(self.pb1,0)
        self.sys.portB_ecrire(self.pb2,0)
        self.sys.portB_config(0,0)
```

La fonction `mesure` réalise une mesure pour un calibre donné, un potentiel d'oxydo-réduction E donné, et une durée donnée. Le potentiel E est le potentiel de l'électrode de travail par rapport à l'électrode normale à hydrogène, qui figure en général sur les courbes courant-potentiel. Le potentiel de consigne par rapport à l'électrode de référence est donc E-Eref. Si l'on souhaite travailler avec un potentiel par rapport à l'électrode de référence utilisée, il suffit de poser `self.Eref = 0` dans le constructeur.

Après avoir appliqué la tension de consigne, le programme attend pendant `duree` secondes puis effectue une mesure pendant la même durée. La mesure est constituée de `Ne=10000` échantillons numérisés en 12 bits. La moyenne de ces échantillons est effectuée pour le calcul du potentiel E et du courant I. Le potentiel E mesuré peut être légèrement différent du potentiel demandé, mais la différence n'excède pas un pour cent.

```
    def mesure(self,calibre,E,duree):
        Ne = 10000
        self.sys.config_echantillon(duree*1.0/Ne*1e6,Ne)
        self.sys.ecrire(1,(E-self.Eref)/self.gain_entree,0,0)
        self.fermer_circuit(calibre)
        time.sleep(duree)
        self.sys.acquerir()
        self.ouvrir_circuit()
        tensions=self.sys.entrees()
        temps=self.sys.temps()
        Ue = tensions[0]
        E = self.Eref-Ue
        Ui = tensions[1]
        if (abs(numpy.mean(Ui))>9.9):
            depassement = True
        else:
            depassement = False
        I = Ui/self.R*1000
        Emean = numpy.mean(E)
        Imean = numpy.mean(I)
        if not(depassement):
```

```
            print("E = %f (V), I = %f (mA)"%(Emean,Imean))
        else:
            print("Courant max atteint")
        return (Emean,Imean,E,I,temps[0],depassement)
```

La fonction `balayage` effectue un balayage point par point, c'est-à-dire N mesures en faisant varier le potentiel de `Ei` à `Ef`. Il est possible de faire un balayage par potentiel croissant ou décroissant. Si le courant maximal est atteint, le balayage est stoppé.

```
    def balayage(self,calibre,Ei,Ef,N,duree):
        dE = (Ef-Ei)*1.0/N
        E = Ei
        self.liste_E = []
        self.liste_I = []
        Ne = 10000
        self.balayage_actif = True
        self.sys.config_echantillon(duree*1.0/Ne*1e6,Ne)
        self.sys.ecrire(1,(E-self.Eref)/self.gain_entree,0,0)
        self.fermer_circuit(calibre)
        k = 0
        self.console = ""
        depassement = False
        while (k<N) and (self.balayage_actif):
            self.sys.ecrire(1,(E-self.Eref)/self.gain_entree,0,0)
            time.sleep(duree)
            self.sys.acquerir()
            tensions=self.sys.entrees()
            temps = self.sys.temps()
            Ue = tensions[0]
            Ui = tensions[1]
            if abs(numpy.mean(Ui)) > 9.9:
                print("Courant max atteint")
                depassement = True
                break
            else:
                I = Ui/self.R*1000
                Imean = numpy.mean(I)
                Emean = numpy.mean(self.Eref-Ue)
                self.I = I
                self.t = temps[0]
                self.liste_E.append(Emean)
                self.liste_I.append(Imean)
                print("Ec = %f (V), E = %f (V), I = %f (mA)"%(E,Emean,Imean))
                self.console += "E (V) = %f, I (mA) = %f\n"%(Emean,Imean)
            E += dE
            k += 1
        self.balayage_actif = False
        self.ouvrir_circuit()
```

```
        return (self.liste_E,self.liste_I)
```

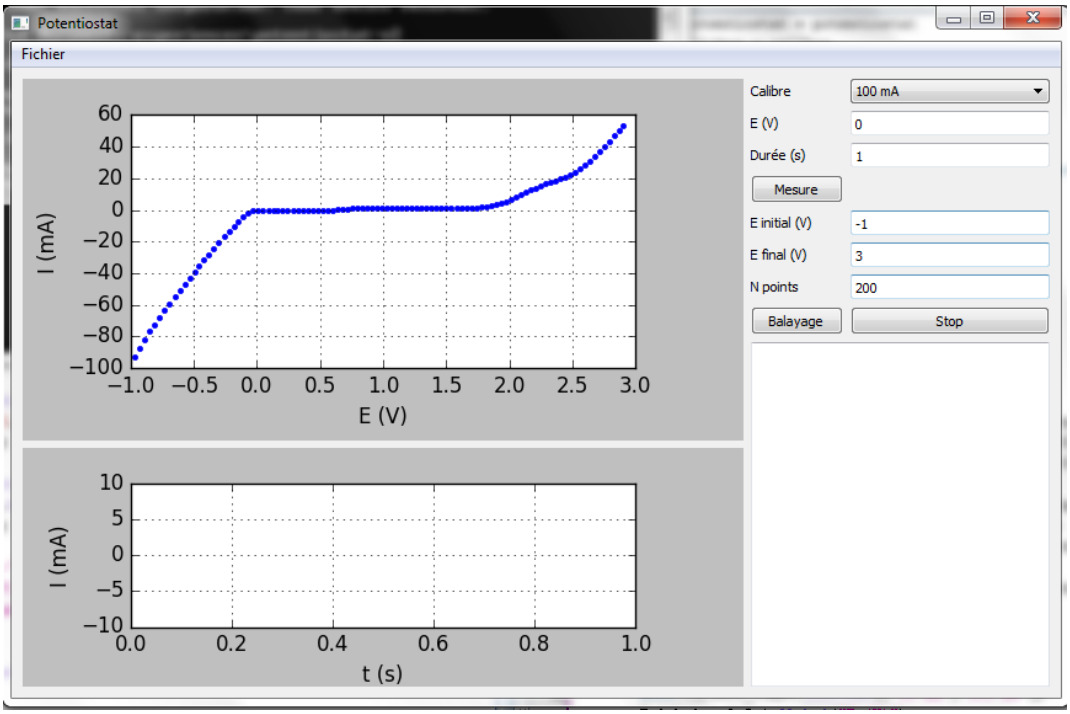
Voici par exemple un script python traçant une courbe courant-potentiel :

```
from potentiostatSysamSP5 import Potentiostat
import numpy
from matplotlib.pyplot import *

pot = Potentiostat()
(E,I) = pot.balayage(2,-0.5,2.5,200,2.0)
numpy.savetxt("test.txt",[E,I])
pot.fermer()
figure()
plot(E,I,"o-")
xlabel("E (V)")
ylabel("I (mA)")
grid()
show()
```

4.b. Interface graphique

Le script potentiostatQT.py offre une interface graphique, qui permet de suivre le tracé de la courbe courant-potentiel pendant le balayage. Il fonctionne sur toute distribution python sur laquelle PyQT est installé (Pythonxy ou Anaconda).



Il permet de faire une mesure à potentiel E et trace dans ce cas le courant en fonction du temps pendant la durée spécifiée. Lorsqu'un balayage est demandé, les points obtenus

sont tracés au fur et à mesure. Un nom de fichier de sauvegarde est demandé au démarrage du balayage. Le programme permet également de lire un fichier enregistré.