Arduino

Communication en Python avec le Moby-CREA

L'Arduino Leonardo assure les mesures et le pilotage du MobyCREA, il est connecté au PC par un câble USB, la liaison PC-Arduino est vue par le PC comme une liaison série. Vous devez déterminer le port affecté à cette liaison série sur le PC dans le panneau de configuration.

Arduino Leonardo
(COM13)

Ce programme en Python peut vous aider, il liste les ports du PC.

```
from serial import *
import serial.tools.list_ports
import time

ports = serial.tools.list_ports.comports()
for port, desc, hwid in sorted(ports):
    print (format(desc))
```

Vous pourrez alors faire toutes les mesures et piloter votre Moby-CREA avec un programme écrit en Python ou dans un autre langage.

Protocole de communication

La communication avec le Moby-CREA se fait avec des chaines de caractères. Toutes les commandes se terminent par le caractère #. La communication se fait à 115200 bauds sur 8 bits, sans parité avec 1 bit de stop. Le premier test à faire consiste à envoyer ?#, le MobyCREA doit répondre ok.

```
from serial import *    #bibliotheque qui gère les accès au port série
import time             # bibliothèque pour la gstion du temps

portSerie=Serial(port="COM15",baudrate=115200)  # ouverture du port série "COM7" (à adapter)
time.sleep(2)           # pause pour laisser le temps à la liaison série de s'établir

portSerie.write(b'?#')        # écriture des caractères "?#", convertis en octects (byte)
retour=portSerie.readline()   # attendre que Arduino retourne "OK"
print(retour)

portSerie.close()             # fermeture de la liaison série
```

Message d'accueil
? : retourne ok

Version du programme Arduino
V : retourne la version du logiciel : portSerie.write(b'V#')

Fin de séquence
F : retourne une fin de chaine logiciel : portSerie.write(b'F#')

Mise à 1 d'une sortie logique
H suivi du code de la sortie sur deux chiffres : portSerie.write(b'H08#')

Mise à 0 d'une sortie logique
L suivi du code de la sortie sur deux chiffres : portSerie.write(b'L10#')

Sortie PWM
P suivi du code de la sortie sur deux chiffres et de la valeur entre 0 et 255 sur 3 chiffres : portSerie.write(b'P06128#')

Lecture d'une entrée logique
I suivi du code de l'entrée sur deux chiffres, retourne 1 ou 0 : portSerie.write(b'I13#')

Lecture d'une entrée analogique
A suivi du code de l'entrée sur 2 chiffres, retourne la valeur numérique sur 10bits : portSerie.write(b'A06#')

Mesure de la largeur d'une impulsion sur une entrée
T suivi du code l'entrée sur 1 octet (2 ou 3), retourne la valeur en microsecondes : portSerie.write(b'T2#')

Lecture des impulsions comptées sur une entrée
C suivi du numéro de l'interruption, 0 pour pin 2 ou 1 pour pin 3 : portSerie.write(b'C0#')

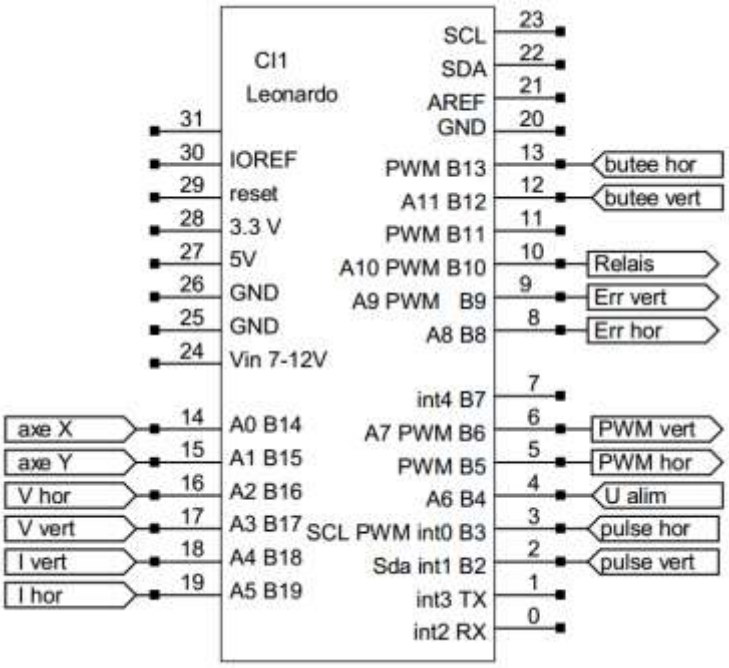
Mise à zéro d'un compteur d'impulsions
R suivi du numéro de l'interruption, 0 pour pin 2 ou 1 pour pin 3 : portSerie.write(b'R0#')

Il est possible de construire une chaine de plusieurs requêtes simultanées

Requête : : portSerie.write(b'I13#I12#C0#C1#A04#A05#A03#A02#T2#T3#A06#F#')

Réponse : 1;0;656;0;598;563;513;463;2004;50000;181;

Affectation des ports



B2 : impulsions en provenance du codeur incrémental du moteur vertical

B3 : impulsions en provenance du codeur incrémental du moteur horizontal

B5 : sortie PWM de commande du moteur horizontal

B6 : sortie PWM de commande du moteur vertical

B8 : sortie de génération d'une fausse butée horizontale (actif à l'état 1)

B9 : sortie de génération d'une fausse butée verticale (actif à l'état 1)

B10 : commande du relais pour passer en pilotage distant (actif à l'état 1)

B12 : butée verticale (actif à l'état 1)

B13 : butée horizontale (actif à l'état 1)

A0 B14 : Broche nommée axe X uniquement sur DB15 (réserve)

A1 B15 : Broche nommée axe X uniquement sur DB15 (réserve)

A2 : V hor, tension d'alimentation du moteur horizontal : Umot = Ualim – (Umesurée x 3,3)

A3 : V vert, tension d'alimentation du moteur vertical : Umoteur = Umesurée x 3,3

A4 : mesure du courant du moteur vertical : I = Umesurée / 0.185V/A)

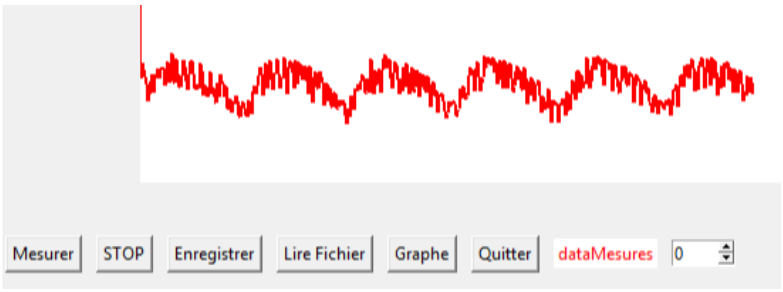
A5 : mesure du courant du moteur horizontal : I = Umesurée / 1.85V/A

A6 : tension d'alimentation mesurée en sortie du bloc secteur : Ualim = Umesurée x 3,3

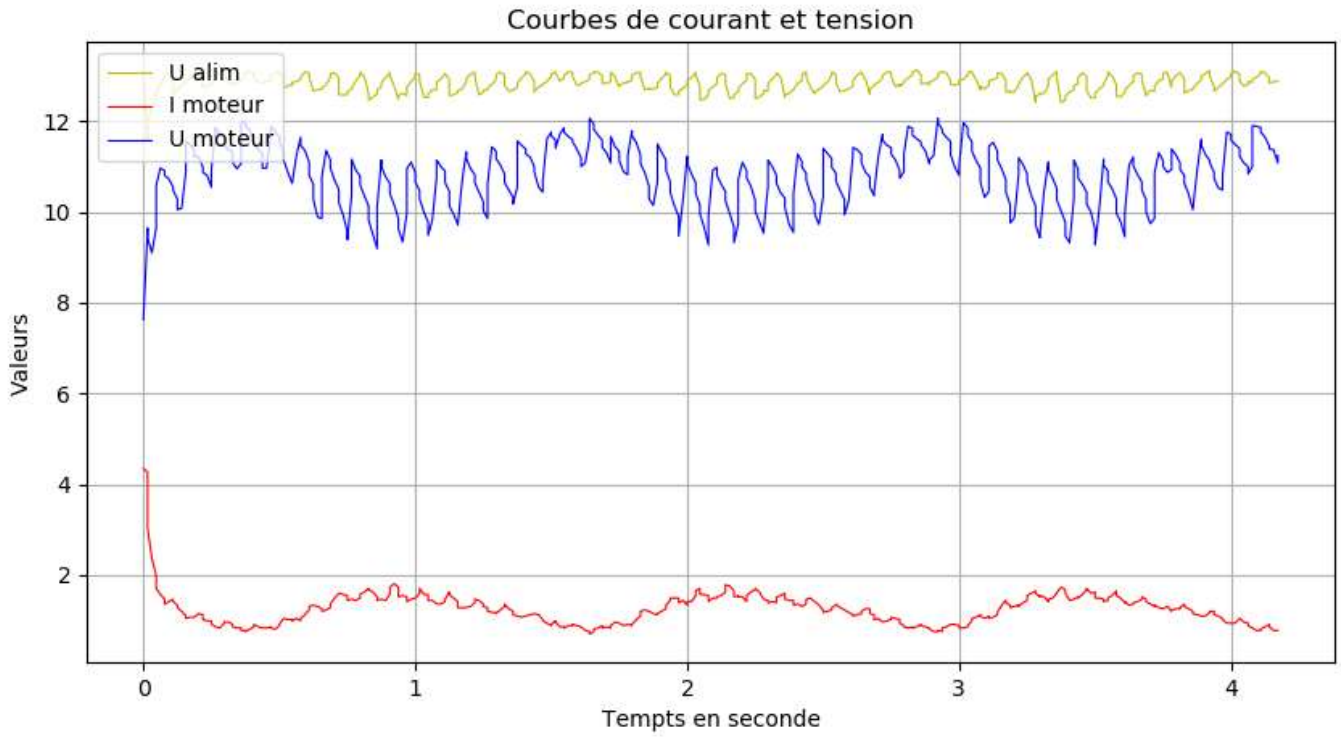
Exemple de programme

Le programme suivant affiche une fenêtre qui permet de lancer des mesures, visualiser les courbes et les enregistrer.

Un aperçu rapide est visible pendant la mesure



Un graphe permet ensuite une lecture détaillée des mesures



Le code Python est téléchargeable en bas de cette page.

```
import matplotlib.pyplot as plt
import time
from tkinter import *
import pickle
from serial import *

def lancerMesure():
    global valIvert,valUvert,valTemps,timeStart,x0,y0
    print ("lancement de la mesure")
    valIvert[:] = [] # vide la liste pour nouvelle mesure
    valUvert[:] = [] # vide la liste
    valTemps[:] = [] # vide la liste
    timeStart = time.time()   # mesure de l'instant initial (t=0)
    x0=0 #abcisse du graphe du canevas
    y0=300 #point de dÃ®part du graphe
    Canevas.delete(ALL) #effacement du graphe (pour nouvelle mesure)
    portSerie.write(b'P06100#') # demarrage axe vertical
    mesurer() #lance la mesure

def mesurer():
    global timer,valIvert,valUvert,valTemps,timeStart,x0,x1,y0,y1
    # mesure des valeur numériques et conversion en volt et Ampere
    portSerie.write(b'A03#F#') # U vertical sur A3
    reponse = int(eval(str(portSerie.readline()).replace(';','''))) #lecture de la réponse
    tension=reponse*(5/1023)*3.3 #conversion en volt
    portSerie.write(b'A04#F#') # I vertical sur A4
    reponse = int(eval(str(portSerie.readline()).replace(';','''))
    courant=reponse*(5/1023)/0.185 #lecture de la réponse
    valIvert.append(courant)
    valUvert.append(tension)
    Tmesure=time.time()-timeStart   # calcul du temps ecoule depuis l'instant initial
    valTemps.append(Tmesure)       # ajout de l'instant T a la liste des valeurs
    x1 = x0+1 # on passe au point suivant en abscisse
    y1 = 300-(valIvert[len(valIvert)-1])*10 #ordonnÃ®e (dÃ®part :0,300 echelle y 50)
    # dessin d'une ligne entre deux points
    Canevas.create_line(x0, y0, x1, y1, width=2, fill='red')
    x0=x1 #le point de d'aarivÃ®e devient le prochain point de dÃ®part
    y0=y1
    timer = fenetre.after(20, mesurer) #le timer relance la mesure

def stop():
    global timer
    if timer: # arret du timer, fin des mesures
        fenetre.after_cancel(timer)
        timer = None
    portSerie.write(b'P06000#') # demarrage axe vertical
    print ("Fin des mesures")

def graphe():
    global valIvert,valUvert,valTemps
    fig = plt.figure(1, figsize=(10, 5)) # taille de la fenetre
    plt.ylabel("Valeurs")   #nom de l'axe des ordonnÃ®es
    plt.xlabel("Tempts en seconde") #nom de l'axe des abcisses
    plt.grid(True) #affichage de la grille
    plt.title("Courbes de courant et tension") #titre de la fenetre
    plt.plot(valTemps,valIvert,"r",linewidth=0.8, label='Courant Vert')  # mise en place du label
    plt.plot(valTemps,valUvert,"b",linewidth=0.8, label='Tension Vert')  # trace la courbe rouge largeur 0.8
    plt.legend(loc="upper left")
    plt.show()
```