UNIVERSITÉ DE LORRAINE | LORRAINE INP Ensem

Promotion 2022-2025

Systèmes distribué S8

TP2

Réalisé par :

ESSARHIR Tarik

WAIH Mohyi-Eddine

Sous la supervision :

CHEVRIER Vincent

I Introduction

On se propose d'étudier un système électrique composé d'un objet physique nommé armoire, ce composant électrique est conçue pour simuler la mise en charge d'un réseau électrique afin de faire des mesures sur les charges résistives R, capacitives C, et inductives L. À titre indicatif ces charges sont pilotées par un automate Wago 750-8202. Plus de détails sur le contexte matériel sont fourni ici.

L'objectif de ce TP est de programmer en python la partie d'acquisition des données à partir du système physique réel, ainsi qu'une partie qui permet de modifier en temps réel la configuration du système physique pour avoir des données qui évoluent au fil du temps.

II Prise en main du code fourni

Tout d'abord, on commence par exécuter le code fourni ci-dessous. On importe une bibliothèque `Armoire5` [1] qui fournie les fonctions nécessaires pour interagir avec le système électrique, ensuite on définit une fonction `pprint` qui permet d'afficher de manière plus adéquate les données mesurée. Enfin on définit une fonction `test` qui prend en argument une charge et renvoie les données qui lui sont liées, le code et son résultat est le suivant :

```
from Armoire5 import *

def pprint(mesure):
    output="U={:.2f} I={:.2f} PA={:.2f} PR={:.2f}"
    return output.format(mesure[0],mesure[1],mesure[2],mesure[3])

def test(charge):

    a=Armoire5()
    a.resetAll()
    print("charge ",charge)
    a.setCharge(charge)
    s=a.readMesure(charge)
    print(pprint(s), "\nS1" ,pprint(a.readSecteur1()),"\nS2", pprint(a.readSecteur2()),"\
        n")
    a.setSource2(charge)
    s=a.readMesure(charge)
    print(pprint(s), "\nS1" ,pprint(a.readSecteur1()),"\nS2", pprint(a.readSecteur2()),"\
        n")
    a.unsetCharge(charge)
```

```
>>> test(5)
charge  5
U=238.37 I=1.48 PA=-243.00 PR=-256.00
S1 U=238.39 I=3.41 PA=773.00 PR=256.00
S2 U=238.34 I=0.10 PA=-16.00 PR=-11.00

U=237.21 I=1.47 PA=-240.00 PR=-250.00
S1 U=237.20 I=4.52 PA=1073.00 PR=-3.00
S2 U=237.17 I=1.53 PA=-259.00 PR=-254.00
```

Le script nous envoie donc les valeurs de toutes les grandeurs électriques souhaitées *(tension U, courant I, résistances,...)*

1. Cette bibliothèque a été codée par Mr CHEVRIER Vincent, pour plus de détails vous pouvez le contactez via mail

III Modification du système physique

Le but de ce code est de simuler des modifications aléatoires de la charge et de la source d'une armoire électronique nommée `Armoire5` pendant une durée donnée. Cette simulation est réalisée à l'aide d'une fonction nommée `simulation` qui utilise la méthode `time` pour déterminer la durée de la simulation et crée un objet `a` de type `Armoire5` pour simuler les changements de charge et de source de manière aléatoire. À chaque itération de la boucle, la méthode `toggleCharge` et la méthode `toggleSource` de l'objet `a` sont appelées pour modifier de manière aléatoire la charge et la source de l'armoire électronique. Les modifications apportées sont stockées dans les variables `c` et `s` puis affiché dans la console

```
from Armoire5 import *
import time
from random import randint

def simulation(pause ,duree):
    deb = time.time()
    fin = deb + duree
    a = Armoire5()

    while time.time() < fin:
        c = randint(0,6)
        s = randint(1,2)
        a.toggleCharge(c)
        a.togglesource(c)
        print(f"modification de la charge {c} mise sur source {s}")
```

```
>>> simulation(2,30)
modification de la charge 6 mise sur source 2
modification de la charge 2 mise sur source 1
modification de la charge 5 mise sur source 2
modification de la charge 6 mise sur source 2
modification de la charge 2 mise sur source 1
modification de la charge 4 mise sur source 2
modification de la charge 2 mise sur source 2
modification de la charge 4 mise sur source 1
```

On voit donc que pour des paramètres `(0,30)` la fonction a effectué huit changements de charge et de source de manière aléatoire.

IV Récupération et mise à disposition des données

Le but ici est de créer trois processus principaux : simulation, acquisition de données et finalement la lecture des données. Pour cela on crée trois `class` différentes

IV.1 Processus simulation

Ce processus utilise la fonction `simulation` codée dans la partie précédente afin de choisir aléatoirement une charge et modifier sa position :

```
import time
from random import randint
from Armoire5 import *
```

```
from threading import*
import threading

class acquisition(threading.Thread):

    def __init__(self,pause ,duree,liste):
        threading.Thread.__init__(self)
        self.pause=pause
        self.duree=duree
        self.liste=liste

    def run(self):
        L=self.liste
        deb = time.time()
        fin = deb + self.duree
        a = Armoire5()
        while time.time() < fin:
            time.pause=randint(0,9)
            l=a.readSecteur2()
            L.append(l)
            print(L)
```

```
>>> x = acquisition(1,2, []) ; x.run()
[(240.1899871826172, 0.10999999940395355, -18.0, -8.0)]
[(240.1899871826172, 0.10999999940395355, -18.0, -8.0), (240.1899871826172,
    0.10999999940395355, -18.0, -11.0)]
...
```

IV.2 Processus acquisition de donnée

Le but final de ce code est de collecter des données à partir d'une armoire électronique `Armoire5` pendant une certaine durée et de stocker ces données dans une liste. Pour cela, on définit une classe d'acquisition de données appelée `acquisition` qui hérite de la classe `Thread` de la bibliothèque `threading`

La classe `Acquisition` utilise la méthode `run` pour effectuer les actions d'acquisition de données. À chaque itération de la boucle, la méthode `readSecteur2` est utilisée pour lire des données à partir du secteur 2 de l'armoire électronique et ces données sont stockées dans la liste `L`. La méthode `sleep` est utilisée pour attendre un certain nombre de secondes entre chaque acquisition de données.

```
import time
from random import randint
from Armoire5 import *
import threading

class simulation(threading.Thread):

    def __init__(self,pause ,duree):
        threading.Thread.__init__(self)
        self.pause=pause
        self.duree=duree

    def run(self):
        deb = time.time()
        fin = deb + self.duree
        a = Armoire5()

        while time.time() < fin:
            c = randint(0,6)
            s = randint(1,2)
```

```
21              a.toggleCharge(c)
22              a.togglesource(c)
23              print(f"modification de la charge {c} mise sur source {s}")
```

```
1  >>> x = simulation(3,5) ; x.run()
2  modification de la charge 1 mise sur source 2
3  modification de la charge 3 mise sur source 2
```