Projet Solarius

TRACKER SOLAIRE

Nom et adresse du client :
Nicolas Flandrois | 傅子洋
Ecam Lyon

SOLARENERGY

Procés verbal de livraison

Fait à Lyon, le 18 juin 2021

Objet : TP Solarius tracker solaire – **PROFILI Emilio**

Livrable	Dépôt	Version
Code source : • Dossier Github **Langage de programmation :** Python ; C++	https://github.com/emilio-profili/TP211-SOLARIUS	N°1 : 25/05/21 N°2 : 26/05/21 N°2 : 01/06/21 N°2 : 12/06/21 N°2 : 13/06/21
Documentation : https://arduino103.blogspot.com/2018/12/un-trackeur-solaire-2-axes-pour-arduino.html http://www.slg-instruments.com/PDF/E-Log-datalogger.pdf https://circuitpython.readthedocs.io/en/latest/shared-bindings/pwmio/index.html https://lecluseo.scenari-community.org/CircuitPython/co/g_lightmeter.html	Les codes Arduino déposés sur github proviennent du premier site en lien ci-contre. Le TP solarius étant issu du projet Brown Dog Gadgets, il est primordial de citer les sources de ce TP.	3 versions du code arduino. La première version est le code de base dépourvu de l'horodatage.

Détail de la livraison

Dossier organisé sur Github comprenant :

- 3 codes arduinos
- 3 codes python
- 3 flowcharts
- 1 dossier READ.ME (explication du dossier)

Anomalies non corrigées, problèmes connus

- Le bon fonctionnement du datalogger n'a pas été vérifié.

Actions à réaliser, mode opératoire

Ouvrir dans l'interface Arduino le 3ème code Arduino en lien sur github puis téléverser le programme dans la plaque Arduino pour faire fonctionner le tracker solaire.

Date de la livraison : 19/06/2021

Projet Solarius

TRACKER SOLAIRE

Nom et adresse du client :
Nicolas Flandrois | 傅子洋
Ecam Lyon

Commentaires du fabricant :

Description fonctionnel et technique :

Dans un premier temps, le scenario 1 permet de faire fonctionner les 2 moteurs pour décrire un arc de cercle à votre panneau solaire. En termes plus simples, votre plateau bougera simultanément de gauche à droite, et de bas en haut, avec un écart de 0,5 s entre chaque changements de positions.

Explication du programme : scénario1

Tous d'abord nous pouvons expliquer le module pwmio importé dans ce programme. Ce module contient des classes permettant d'accéder aux E/S d'impulsion de base.

Le programme initialise le tracker solaire avec **pwnio.PWMOut()**, et définit **duty_cycle**

Il définit ensuite **servo1** et **servo2**,

puis se met en veille pendant 0,5 seconde grâce à **time.sleep()**.

Pour plus d'informations, le lien vers mes sources : https://circuitpython.readthedocs.io/en/latest/shared-bindings/pwmio/index.html

```
import board
import time
import pwmio
from adafruit_motor import servo

x1 = pwmio.PWMOut(board.A1, duty_cycle=2 ** 15, frequency=50)
x2 = pwmio.PWMOut(board.A3, duty_cycle=2 ** 15, frequency=50)

# Create a servo object, servo.
servo1 = servo.Servo(x1)
servo2 = servo.Servo(x2)

while True:
    for angle in range(0, 180, 1):  # 0 - 180 degrees, 1 degre
        servo1.angle = angle
        servo2.angle = angle
        time.sleep(0.5)
    for angle in range(180, 0, -1):  # 180 - 0 degrees, 1 degr
        servo1.angle = angle
        servo2.angle = angle
        time.sleep(0.5)
```

J'ai choisit ensuite de faire une While True, c'est-à-dire une boucle répétée à l'infini, répétant continuellement un ordre donné au tracker solaire. Cet ordre est de pivoter puis de se mettre en veille pendant 0,5s. Ce qui permet au tracker solaire de pivoter est donc **la fonction** range() **dans la** while True

. **La fonction range()** renvoie une séquence de nombres, commençant à 0 par défaut, d'incrément 1 et s'arrêtant avant un nombre spécifié, ici 180 car l'angle maximal est de 180°. Cette séquence de nombres n'est rien d'autre que la variation de l'angle des servomoteurs 1 et 2 par rapport à leur position initiale.

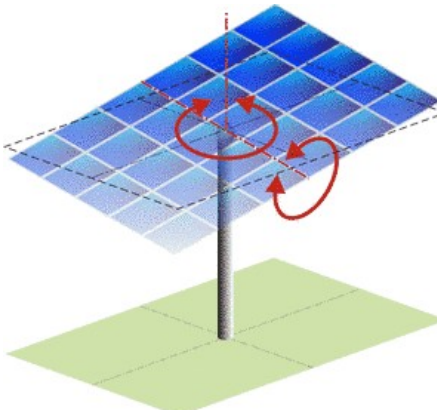


Schéma des deux axes de rotation du tracker solaire :

Le programme fait donc varier l'angle des servomoteurs par rapport à leur position initiale de 0° et puis quand ils arrivent en bout de course (avec un angle de 180°), les deux servomoteurs doivent pouvoir tourner dans l'autre sens. C'est pourquoi il y a deux **for** dans la **While True**. Le second **for** permet de faire varier l'angle des servomoteurs dans l'autre sens, avec pour position initiale l'angle de 180° et en utilisant donc une incrémentation de -1. Faire tourner les servomoteurs dans les deux sens est impératif pour pouvoir suivre la course du soleil d'un jour à l'autre.

Nom et adresse du client :
Nicolas Flandrois | 傅子洋
Ecam Lyon

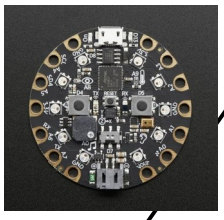
Explication du programme : scénario1

Ensuite, avec les 4 capteurs de luminosité disposés en croix (*cf. photos*) vous pourrez déterminer où se trouve le soleil. Le scripte du scenario 2 permet d'ajuster la position du plateau (inclinaison et rotation), pour suivre le soleil, de façon à ce que le panneau soit toujours en face du soleil.

J'ai appelé la fonction **move_down()** la fonction qui soustrait 5 degrés à l'angle d'un servo. J'ai choisit 5° arbitrairement : le tracker solaire se réoriente donc de 5° maximum pour faire face au rayon lumineux.
Par ailleurs, j'ai définie les paramètres, (variables d'entrée) : **my_servo**

Ensuite, dans une while True j'utilise la fonction **`map_range()`** du module **simpleio**. Cette fonction permet de convertir la luminosité qui est un entier entre 0 et 320 en un entier entre 0 et 9 puisqu'on a 10 LEDs sur la carte CPX.

```
def move_down(servo):

    my_servo = servo
    my_servo.angle -= 5

    """
    move_down() is a fonction that substract 5 deg to the ang
    Parameters (Input Variables):
        servo :
            servo object that you want to move down.
    Output:
        moves the chosen servo.
    Dependencies:
        adafruit_motor
        servo
    """

while True:
    hight = simpleio.map_range(light1.value, 0, 320, 0, 10)
    print(int(hight))

    for i in range(0, 10, 1):
        if i <= hight:
            move_up(servo1)
            move_down(servo2)

    time.sleep(0.01)
```



La carte CPX dispose d'un capteur de lumière sur le côté droit, à côté de l'œil imprimé sur la carte. Il détecte la quantité de lumière ambiante et renvoie le niveau de lumière en fonction de ces données.
Lien de mes sources : https://lecluseo.scenari-community.org/CircuitPython/co/g_lightmeter.html

La variable que j'ai appelé hight est le pic de luminosité, soit la plus haute intensité qui s'obtient lorsque les rayons lumineux sont parfaitement en face aux capteurs et orthogonalement aux cellules photovoltaïques par la même occasion. Par ailleurs, la luminosité est convertit en nombre décimaux mais **hight** reste affiché, grâce à **int,** en nombre entier, ce qui permet d'être compris par la fonction **range().**

Pour **i** allant de 0 à 10 avec une incrémentation de 1 dans la fonction **range()**, si **i** est inférieur ou égale à **hight** alors l'ordre est donné aux servos d'ajuster leur position face au soleil. De plus, les servos doivent être réorientés jusqu'à ce que les capteurs affichent le pic d'intensité. Si à l'inverse, **i** est supérieur ou égale à 1, alors cela signifie que le panneau solaire est déjà idéalement positionné et l'ordre est donné aux servos de suivre le pic de luminosité c'est-à-dire dans ce cas de ne pas bouger.

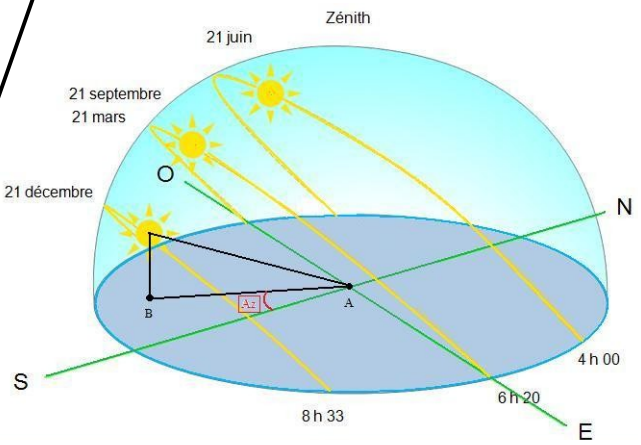


Schéma de la course du soleil par heure et mois :

Le programme du scénario 2 permet donc de suivre la cours du soleil tout au long de la journée car l'ordre donné aux servos de suivre le pic d'intensité est compris dans une **while True**. Cela permet au tracker solaire de positionner son panneau photovoltaïque toujours parfaitement face au soleil. Enfin, j'ai choisi un temps d'attente de 0,01s dans le **time.sleep()** entre chaque ordre pour que le suivit du pic d'intensité **hight** soit précis et d'éviter ainsi de perdre en efficacité de production d'énergie électrique.

Projet Solarius

TRACKER SOLAIRE

Nom et adresse du client :

Nicolas Flandrois | 傅子洋
Ecam Lyon

Date de la réception : _____

La réception est prononcée :

☐ Sans réserve ☐ Avec réserve ☐ Refusée

Fait à _____ Pour {{Client}}

Le _____ *(Faire précéder du nom et prénom)*