Activité Objets Connectés

O Clignotants pour vélo O

Objectif : Réaliser un dispositif de feux clignotants pour vélo basé sur le fonctionnement particulier des clignotants d'une moto.

1 Conventions

Consignes sur le travail à effectuer Données Ressources

2 Prérequis

- Programmation de base en python : variables, opérateurs,fonctions, boucles,
- Utilisation de la carte micro-bit : gestion des boutons, de la matrice à leds

3 Nouvelles notions à acquérir

- Fonction mémoire
- Fonction logique ET

4 Descriptif du contexte

Sur la plupart des motos, la commande des feux clignotant est un simple commutateur électrique à 3 positions (gauche, neutre, droite)

Sur les motos d'une célèbre marque américaine, le dispositif est différent. Il y a

- un bouton poussoir de chaque côté du guidon qui pemet de mettre en fonctionnement ou d'arrêter les clignotants du côté correspondant

poignée gauche

poignée droite

- un témoin indicateur sur le tableau de bord (flèche orientée) qui clignote au rythme des feux

Fonctionnement :

- Pour mettre en service les feux clignotants d'un côté, il suffit d'appuyer et relacher le bouton poussoir correspondant
- Les feux qui clignotent s'arrêtent dans le 3 cas suivants
  - on appuie de nouveau sur le même bouton poussoir
  - on appuie sur le bouton poussoir de l'autre côté
  - on penche la moto du côté des clignotants et on redresse la moto en position verticale. Ce mode d'arrêt permet d'éteindre automatiquement les clignotants lorsqu'on a terminé son changement de direction

5 Utilisation de la carte micro-bit

- Le Bouton A servira de bouton Gauche ⇦
- Le Bouton B servira de bouton Droite ⇨
- Le témoin indicateur s'affichera sur la matrice à leds  ou 
- Des leds "Gauche" et "Droite" seront respectivement branchées sur les ports P0 et P2

6 Mémorisation d'une information

Dans ce paragraphe on va montrer la différence entre une information non-mémorisée et un information mémorisée.

1. **Information non mémorisée**
   - On met une variable à 1 lorsqu'un événement survient
   - On met la variable à 0 lorsque l'évènement disparaît

   exemple de la vie courante : Quand on appuie sur le bouton "Avertisseur sonore" d'un véhicule, le klaxon émet un son (klaxon=1) et quand on relâche le bouton, le son s'arrête (klaxon =0)

   Application en micro-python

Avec le logiciel mu-editor, **tester** le code suivant sur la carte microbit

```
from microbit import *

while True:
    if button_a.is_pressed():
        mavar = 1
    else:
        mavar = 0
    display.show(mavar)
```

Ressources pour les questions suivantes :
https://microbit-micropython.readthedocs.io/en/latest/tutorials/io.html

- **Connecter** une led sur le port 0 (pin0)
- **Compléter** le programme précédent pour la led s'allume quand la variable ***mavar*** = 1 et s'éteint quand la variable ***mavar*** = 0
- **Enregistrer** ce programme
- **Faire constater** le fonctionnement par le professeur

2. **Information mémorisée**

   La mémorisation consiste à :
   - mettre à 1 variable quand un événement survient
   - maintenir à 1 la variable même lorsque l'évènement disparaît *(mémorisation à 1)*
   - mettre à 0 la variable quand un autre événement survient
   - maintenir à 0 la variable même lorsque l'évènement disparaît *(mémorisation à 0)*

exemple de la vie courante : Pour appeler un ascenseur :

- on appuie sur un bouton poussoir
- un voyant s'allume pour indiquer que la demande a été prise en compte
- quand on relâche le bouton, le voyant reste allumé *(mémorisation à 1)*
- le voyant s'éteint lorsque l'ascenseur est arrivé
- le voyant reste éteint même lorsque l'ascenseur repart *(mémorisation à 0)*

- **Modifier** le programme précédent pour que la led s'allume quand on appuie sur le bouton A, et s'éteigne lorsqu'on appuie sur le bouton B
- **Faire constater** le fonctionnement

7 Application : Clignotants

Etape 1 : *Les clignotants doivent rester en fonctionnement quand on relâche le bouton, il faut donc mémoriser l'appui sur les boutons* ⇦ *et* ⇨

Saisir le code suivant dans un nouveau programme nommé clignos-velo.py

```
from microbit import *
cligno_droit = 0
cligno_gauche = 0

while True:
    if button_a.was_pressed():
    ....
```

Les variables cligno_droit et cligno_gauche vont servir à mémoriser l'appui sur les boutons

Compléter ce programme que :

- la variable ***cligno_droit*** change d'état (0 ⟷ 1) à chaque fois qu'on appuie sur le bouton B
- la variable ***cligno_gauche*** change d'état (0 ⟷ 1) à chaque fois qu'on appuie sur le bouton A
- l'état de ***cligno_droit*** s'affiche sur la matrice à leds
- l'état de ***cligno_gauche*** se visualise sur la led branchée sur la broche pin0

Conseil : Pour tester l'appui sur les boutons, utiliser la méthode was-pressed() à la place de is_pressed()

Etape 2 : *Arrêter le clignotant quand on met en marche celui de l'autre côté*

Compléter le programme précédent pour que :
- l'appui sur le bouton A mette la variable ***cligno_droit*** à 0
- l'appui sur le bouton B mette la variable ***cligno_gauche*** à 0

Etape 3 : *Faire clignoter les leds et les flèches sur la matrice à led*

```
from microbit import *
cligno_droit = 0
cligno_gauche = 0

def clignote_a_droite():
    ....

def clignote_a_gauche():
    ....

while True:
```

Ressources pour les questions suivantes :
https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html
https://microbit-micropython.readthedocs.io/en/latest/tutorials/io.html

- **Insérer** les fonctions ***clignote_a_droite()*** et ***clignote_a_gauche()*** comme indiqué ci-dessus
- **compléter** la fonction ***clignote_a_droite()*** pour que :
  - pendant ½ seconde
    - un flèche à droite (ARROW_E) s'allume sur la matrice à leds
    - la led branchée sur la broche P2 (pin2) s'allume
  - pendant ½ seconde
    - la matrice à leds s'éteint
    - la led branchée sur la broche P2 (pin2) s'éteint

idem pour la fonction ***clignote_a_gauche()***
- **ajouter** le code dans la boucle "while" , pour que
  - la fonction ***clignote_a_droite()*** soit exécutée si la variable ***cligno_droit*** est à 1
  - la fonction ***clignote_a_gauche()*** soit exécutée si la variable ***cligno_gauche*** est à 1

Faire constater le fonctionnement

Etape 4 : *Arrêter les clignotants quand on penche puis redresse le vélo.*

Pour mesurer l'inclinaison, on va utiliser l'accéléromètre de la carte micobit
https://microbit-micropython.readthedocs.io/en/latest/tutorials/movement.html

Créer un nouveau programme avec le code ci-dessous et le **tester**
Pour afficher la valeur retournée par print, ouvrir une console REPL puis CTRL+D
Observer, analyser le fonctionnement de ce programme quand on incline la carte microbit

```
from microbit import *

limite = 100

while True:
   reading = accelerometer.get_x()
   print (reading)
   if reading > limite:
      display.show("R")
   elif reading < -limite:
      display.show("L")
   else:
      display.show("-")
```

Revenir au programme clignos-velo.py et ajouter le code pour :

- **initialiser** en début de programme (avant la boucle "while")
  - la variable ***limite*** à 400
  - les variables ***out_limite_d*** et ***out_limite_g*** à 0

Puis en fin de programme **ajouter** le code pour :
- **mettre** dans une variable ***inclinaison*** la valeur retournée par l'accéléomètre x
- **mettre** à 1 la variable ***out_limite_d*** quand la variable ***inclinaison*** dépasse ***limite***
- **mettre** à 1 la variable ***out_limite_g*** quand la variable ***inclinaison*** dépasse ***-limite***

(ça permet de mémoriser le fait qu'on s'est penché)
- **mettre** à 0 la variable ***out_limite_d***, et éteindre les clignotants si :
  - l'inclinaison est comprise entre 30 et -30 (vélo "vertical")
  - **ET** que la variable ***out_limite_d*** est à 1 (on s'est penché à droite)

- **faire** la même chose pour le côté gauche
- f**aire constater** le fonctionnement

Ressources sur les fonctions logiques :

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4