OBJETS CONNECTÉS

La ville connectée avec micro :bit

Ruban de LED, communication radio, capteur à ultrasons, barrière d'accès

Nom : ..... Prénom : ..... Classe : .....

Objectif. Programmer des objets de la ville connectée : un ruban de LED, deux cartes qui communiquent par radio, un capteur de distance à ultrasons et une barrière d'accès à une zone piétonne.
Prérequis. La fiche « Prise en main de la carte micro :bit » : affichage, boucles `for` et `while`, tests `if`, boutons, broches.
Matériel. Deux cartes micro :bit, un ruban de 30 LED, une LED Grove, un capteur à ultrasons Grove, un capteur infrarouge, un buzzer et une barrière motorisée (servomoteur). Toutes les activités peuvent aussi être faites sur le **simulateur micro :bit · ville connectée** de `https://www.webclasse.fr`.
Méthode. Pour chaque activité « Découvrir », lire le programme, prédire ce qu'il va faire, le tester, puis répondre aux questions. Relever ensuite les défis.

1. Le ruban de LED

Découvrir Allumer une LED du ruban

Le ruban comporte 30 LED. Chacune peut prendre n'importe quelle couleur. Il est branché sur la broche `pin0`.

Tester le programme ci-contre : la première LED s'allume en rouge pendant 2 secondes, puis le ruban s'éteint.

Programme

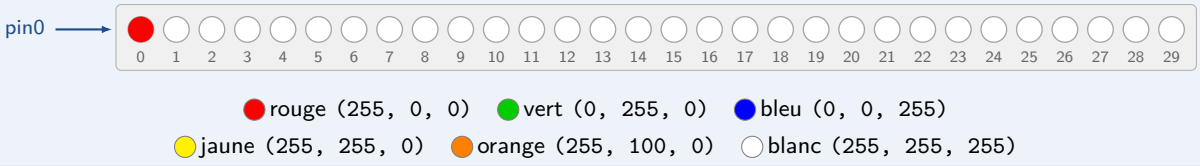
```
from microbit import *
from neopixel import *

np = NeoPixel(pin0, 30)

np[0] = (164, 0, 0)
np.show()
sleep(2000)
np.clear()
```

À retenir — Le ruban de LED

- `from neopixel import *` importe les instructions du ruban ; `np = NeoPixel(pin0, 30)` déclare un ruban de 30 LED branché sur `pin0`.
- Les LED sont numérotées **à partir de 0** : `np[0]` est la première, `np[29]` la dernière.
- Une couleur s'écrit **(rouge, vert, bleu)**, chaque composante allant de 0 à 255.
- `np[4] = (0, 255, 0)` **mémorise** une couleur ; `np.show()` l'**envoie** au ruban ; `np.clear()` éteint tout le ruban.



Savoir-faire : *allumer une LED du ruban avec une couleur choisie.*

Questions de compréhension

1) Que représentent les trois nombres de `(164, 0, 0)` ?

.....

2) Comment obtenir du vert ? du bleu ? du blanc ?

3) Que se passe-t-il si on supprime la ligne `np.show()` ? Pourquoi ?

4) Quel est le numéro de la première LED ? de la dernière ?

★ Défi n° R1

Écrire un programme qui allume la **cinquième** LED du ruban en vert.

À retenir — Les fonctions

Une **fonction** regroupe des instructions sous un nom. Ses **paramètres** sont des valeurs qu'on lui donne à chaque **appel**.

- `def nom(a, b):` définit la fonction ; les instructions qui suivent sont décalées.
- `nom(3, 5)` l'appelle avec `a = 3` et `b = 5`.

```
def allume_en_rouge(position):
    np[position] = (255, 0, 0)
    np.show()

allume_en_rouge(2)    # appel
```

Savoir-faire : *définir et appeler une fonction avec des paramètres.*

★ Défi n° R2

Écrire une fonction `RubanLed_allumer_une(position, couleur)` qui allume la LED numéro `position` avec la couleur `couleur`. La tester, par exemple avec `RubanLed_allumer_une(7, (0, 0, 255))`.

★ Défi n° R3

Écrire une fonction `RubanLed_successif_bleu()` qui allume en bleu toutes les LED l'une après l'autre, avec 200 ms entre deux LED, en utilisant **une boucle**. À la fin, tout le ruban est bleu. *Indice : utiliser une boucle `for` qui parcourt toutes les LED, `for i in range(len(np)):` (`len(np)` donne le nombre de LED). À chaque tour, colorer la LED `i`, l'envoyer au ruban avec `np.show()`, puis faire une pause.*

★ Défi n° R4

a) Écrire une fonction `RubanLed_allumer_tout_successif(couleur, intervalle_temps)` qui allume l'une après l'autre toutes les LED avec la couleur `couleur`, en attendant `intervalle_temps` millisecondes entre deux LED.

b) La tester avec `RubanLed_allumer_tout_successif((125, 125, 0), 200)` et observer la couleur des LED au-delà de la huitième. Comment l'expliquer ?

★ Défi n° R5 (expert)

Écrire une fonction `RubanLed_double_boucle()` qui allume tout le ruban en `(0, 0, 25)` pendant une demi-seconde, puis en `(0, 0, 50)` pendant une demi-seconde, et ainsi de suite jusqu'à `(0, 0, 250)`. Utiliser deux boucles `for` imbriquées : l'une pour l'intensité du bleu, l'autre pour les LED. *Indice : `range(25, 251, 25)` donne 25, 50, …, 250.*

2. La communication radio

Découvrir Envoyer un message d'une carte à l'autre

Il faut **deux cartes**. Téléverser le premier programme sur la carte 1 et le second sur la carte 2, puis maintenir le bouton A de la carte 1 enfoncé : la carte 2 sourit. Relâcher : elle devient triste.

Carte 1 · émettrice

```
from microbit import *
import radio

radio.config(address=1, group=1)
radio.on()

while True:
    if button_a.is_pressed():
        radio.send("A")
    else:
        radio.send("-")
```

Carte 2 · réceptrice

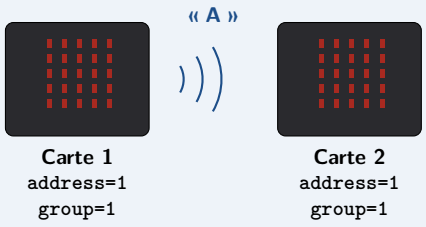
```
from microbit import *
import radio

radio.config(address=1, group=1)
radio.on()

while True:
    donnees_reception = radio.receive()
    if donnees_reception == "A":
        display.show(Image.SMILE)
    elif donnees_reception == "-":
        display.show(Image.SAD)
```

À retenir — La radio

- `import radio` importe les instructions de la radio.
- `radio.config(address=1, group=1)` règle l'adresse et le groupe. Deux cartes ne se « comprennent » que si elles ont la **même adresse** et le **même groupe**.
- `radio.on()` allume la radio.
- `radio.send("A")` envoie un texte.
- `message = radio.receive()` récupère le texte reçu, ou `None` si rien n'est arrivé.
- Pour envoyer un nombre, on le convertit en texte : `radio.send(str(d))`.



Savoir-faire : *faire communiquer deux cartes par radio.*

Questions de compréhension

1) Remplacer `group=1` par `group=2` sur la carte 2 seulement. Que se passe-t-il ? Pourquoi ?

. . . . . .

. . . . . .

2) Que renvoie `radio.receive()` quand aucun message n'est arrivé ? Que fait alors la carte 2 ?

. . . . . .

. . . . . .

3) Pourquoi faut-il une boucle `while True` sur chacune des deux cartes ?

.....

.....

★ Défi n° RA1

Une LED Grove est branchée sur la broche `pin1` de la carte 2. Modifier les deux programmes pour **allumer** cette LED avec le bouton A de la carte 1, et l'**éteindre** avec le bouton B. *Indice : choisir deux messages différents, par exemple « ON » et « OFF ».*

3. Le capteur à ultrasons

Découvrir Mesurer une distance

Le capteur à ultrasons est branché sur `pin1`. La fonction `distance_cm()` est **fournie** : il suffit de la recopier en haut du programme puis de l'utiliser.

Fonction fournie et programme de test

```
from microbit import *
import utime
import machine

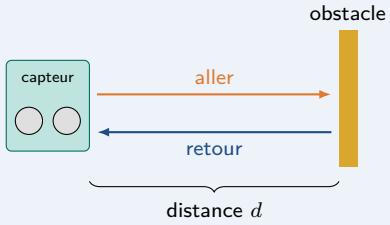
def distance_cm():
    v = 0.0340  # vitesse du son : 0,0340 cm par microseconde (340 m/s)
    pin1.write_digital(0)
    utime.sleep_us(2)
    pin1.write_digital(1)
    utime.sleep_us(5)
    pin1.write_digital(0)
    pin1.read_digital()
    temps = machine.time_pulse_us(pin1, 1, 1000000)
    return temps * v / 2 + 0.55

while True:
    print(distance_cm())
    sleep(500)
```

À retenir — La mesure d'une distance par ultrasons

Le capteur émet un ultrason, qui se réfléchit sur l'obstacle et revient. La carte mesure la durée t de cet **aller-retour**, en microsecondes, puis calcule la distance :

$$d = \frac{v \times t}{2} \quad \text{avec } v = 0{,}0340 \text{ cm/µs.}$$



— `distance_cm()` renvoie la distance en centimètres.

— On la range dans une variable pour l'utiliser : `d = distance_cm()`.

Savoir-faire : *mesurer une distance avec le capteur à ultrasons.*

Questions de compréhension

1) Pourquoi divise-t-on par 2 dans le calcul de la distance ?

.....

.....

2) Le son parcourt 340 m par seconde. Vérifier que cela fait 0,0340 cm par microseconde.

.....

.....

3) Déplacer un obstacle devant le capteur. Comment évoluent les valeurs affichées ?

.....

★ Défi n° U1

Un ruban de 30 LED est branché sur `pin0`. Allumer **tout le ruban** en vert lorsque la distance est supérieure à 30 cm, en orange lorsqu'elle est comprise entre 20 et 30 cm, et en rouge lorsqu'elle est inférieure à 20 cm. *Couleurs conseillées : vert* `(0, 255, 0)`, *orange* `(255, 100, 0)`, *rouge* `(255, 0, 0)`.

★ Défi n° U2

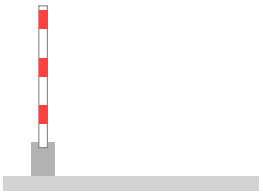
La carte 1, munie du capteur, mesure la distance. Si un obstacle est à moins d'un mètre, la carte 2 affiche une alerte (par exemple `Image.SKULL`) ; sinon, l'écran de la carte 2 reste éteint. *Indice : la carte 1 envoie un message différent selon la distance.*

4. La barrière d'accès à une zone piétonne

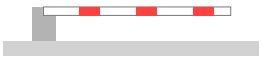
Une ville rend son centre piéton : les voies d'accès sont fermées par des barrières. Les véhicules de secours doivent pourtant pouvoir entrer. Pour simplifier, le « code » du conducteur est le bouton A pour ouvrir et le bouton B pour fermer.

Découvrir Tester la barrière

La barrière est actionnée par un servomoteur branché sur `pin0`. Le fabricant fournit les fonctions `ouvre_barriere()` et `ferme_barriere()` : il suffit de les recopier et de les utiliser. Le contenu de `pilotage()` n'est **pas à connaître**.



`ouvre_barriere()`



`ferme_barriere()`

Fonctions fournies et test

```
from microbit import *

def pilotage(angle):
    impulsion_minimale = 46
    coefficient = 0.5125
    return round(coefficient * angle
                 + impulsion_minimale)

def ouvre_barriere():
    pin0.set_analog_period(20)
    pin0.write_analog(pilotage(0))

def ferme_barriere():
    pin0.set_analog_period(20)
    pin0.write_analog(pilotage(90))

while True:
    ouvre_barriere()
    sleep(2000)
    ferme_barriere()
    sleep(2000)
```

Questions de compréhension

1) Que fait ce programme ?

.....

2) Que se passe-t-il si on supprime les deux `sleep(2000)` ?

.....

.....

★ Défi n° B1

Compléter le programme pour **ouvrir** la barrière lorsqu'on appuie sur le bouton A et la **fermer** lorsqu'on appuie sur le bouton B.

★ Défi n° B2

Par sécurité, un avertissement lumineux (une image sur la matrice) dure **une seconde** avant que la barrière se lève. Avant qu'elle se baisse, la matrice **clignote trois fois**, chaque clignotement durant une demi-seconde. La barrière se commande toujours avec les boutons A et B.

Découvrir Détecter un véhicule

Un capteur infrarouge placé devant la barrière détecte la présence d'un véhicule. Il est relié par un fil à la broche `pin1` de la carte « ordinateur central ». Un buzzer est branché sur `pin0` de cette même carte.

Programme

```
from microbit import *

while True:
    if pin1.read_digital() == 0:
        display.show("O")  # détecté
    else:
        display.show("X")
    sleep(500)
```

À retenir — Le capteur infrarouge et le buzzer

- `pin1.read_digital()` vaut **0** quand un véhicule est détecté, **1** sinon. La petite LED rouge du capteur s'allume en même temps.
- `import music` puis `music.pitch(440, 200, pin0)` émet un son de 440 Hz (un la) pendant 200 ms sur `pin0`.

Savoir-faire : *lire un capteur numérique et commander un buzzer.*

Questions de compréhension

1) Que renvoie `pin1.read_digital()` quand un véhicule est devant le capteur ?

.....

2) À quoi sert la petite LED rouge du capteur ?

.....

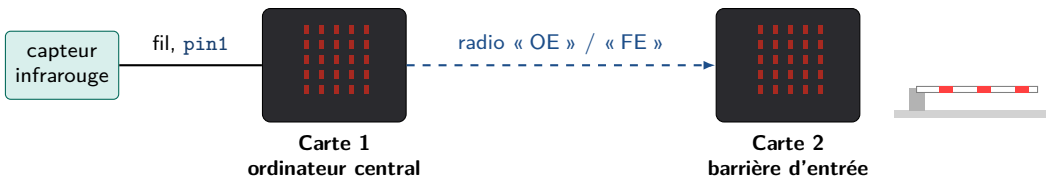
.....

★ Défi n° B3

Quand un véhicule est détecté, le buzzer de l'ordinateur central émet un son. Sans véhicule, il reste silencieux.

Découvrir Commander la barrière à distance

La ville veut gérer la barrière depuis un seul **ordinateur central**. Le capteur infrarouge est relié par un fil à l'ordinateur central ; la barrière d'entrée a sa propre carte, commandée par radio (`group=12`).



Tester la liaison : l'ordinateur central envoie « BE » tant que le bouton A est enfoncé, « RIEN » sinon.

Carte 1 · ordinateur central

```
from microbit import *
import radio

radio.config(address=1, group=12)
radio.on()

while True:
    if button_a.is_pressed():
        radio.send("BE")  # Barrière Entrée
    else:
        radio.send("RIEN")
```

Carte 2 · barrière d'entrée

```
from microbit import *
import radio

radio.config(address=1, group=12)
radio.on()

while True:
    donnees_reception = radio.receive()
    if donnees_reception == "BE":
        display.show("O")
    else:
        display.show("X")
```

Questions de compréhension

1) Sur quels critères peut-on dire que la liaison radio fonctionne ?

.....

.....

.....

★ Défi n° B4

L'ordinateur central envoie par radio une commande d'**ouverture** (un court texte, par exemple « OE ») quand on appuie sur A, et une commande de **fermeture** (« FE ») quand on appuie sur B. La carte de la barrière d'entrée traite les messages reçus pour s'ouvrir ou se fermer.

★ Défi n° B5

Quand un véhicule se présente devant le capteur infrarouge, l'ordinateur central affiche P, sinon X. Si l'opérateur appuie alors sur A, la barrière d'entrée s'ouvre, puis se referme dès qu'il n'y a plus de véhicule détecté. *Indice : c'est l'ordinateur central qui connaît le capteur ; c'est donc lui qui décide quand envoyer la commande de fermeture.*

5. Bilan

Aide-mémoire

Instruction	Rôle
`from neopixel import *`	importer les instructions du ruban de LED
`np = NeoPixel(pin0, 30)`	ruban de 30 LED branché sur `pin0`
`np[4] = (0, 255, 0)`	LED n° 4 (la 5e) en vert : (rouge, vert, bleu), de 0 à 255
`np.show()` `np.clear()`	envoyer les couleurs au ruban tout éteindre
`len(np)`	nombre de LED du ruban
`def nom(a, b):`	définir une fonction à deux paramètres
`nom(3, 5)`	appeler la fonction
`import radio`	importer les instructions de la radio
`radio.config(address=1, group=12)`	régler l'adresse et le groupe (identiques sur les deux cartes)
`radio.on()`	allumer la radio
`radio.send("A")`	envoyer un texte ; un nombre se convertit avec `str()`
`message = radio.receive()`	texte reçu, ou `None` si rien n'est arrivé
`d = distance_cm()`	distance mesurée par le capteur à ultrasons (fonction fournie)
`pin1.read_digital()`	capteur infrarouge : 0 si un véhicule est détecté
`import music`	importer les instructions de son
`music.pitch(440, 200, pin0)`	son de 440 Hz pendant 200 ms sur `pin0`
`ouvre_barriere()` `ferme_barriere()`	lever, baisser la barrière (fonctions fournies)

Erreurs fréquentes

- oubli de `np.show()` : les couleurs sont mémorisées mais le ruban ne change pas ;
- numérotation des LED : la cinquième LED est `np[4]` ;
- cartes radio avec des `group` différents, ou oubli de `radio.on()` ;
- envoi d'un nombre par radio sans le convertir en texte avec `str()` ;
- un `else` qui annule aussitôt la commande donnée par un bouton.

Grille d'auto-évaluation

Je sais…	oui – en cours – non	Défis réussis
allumer une LED du ruban avec une couleur choisie	❑ ❑ ❑	R1 ❑
définir et appeler une fonction avec des paramètres	❑ ❑ ❑	R2 ❑ R3 ❑ R4 ❑ R5 ❑
faire communiquer deux cartes par radio	❑ ❑ ❑	RA1 ❑
mesurer une distance avec le capteur à ultrasons	❑ ❑ ❑	U1 ❑ U2 ❑
commander la barrière avec des boutons	❑ ❑ ❑	B1 ❑ B2 ❑
lire un capteur numérique et commander un buzzer	❑ ❑ ❑	B3 ❑
commander la barrière à distance	❑ ❑ ❑	B4 ❑ B5 ❑