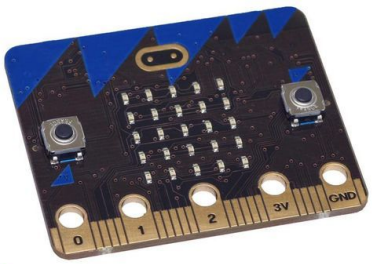
MicroPython sur carte micro:bit

Rendre les objets intelligents grâce à Python



Olivier Lécluse

Mars 2019
1.0

Table des matières

Introduction

La carte micro:bit est un ordinateur à carte unique à processeur ARM conçu par la BBC pour l'éducation informatique au Royaume-Uni. En 2015, ce projet prévoit d'offrir gratuitement un exemplaire à chaque écolier britannique de douze ans. Plus récemment

la carte micro::bit peut se programmer en utilisant plusieurs langages : javascript (blocs ou texte), scratch3, LISP, C++ avec l'environnement Arduino mais aussi MicroPython. Nous nous intéresserons ici uniquement à la programmation de la carte sous MicroPython.

Pour la programmation sous MicroPython, nous utiliserons l'environnement *mu* que nous avons déjà rencontré pour les cartes Adafruit ou ESP8266/32 précédemment dans ce tutoriel. Cet environnement et simple à utiliser et s'adapte à plusieurs types de cartes différents. Il est libre, gratuit et multi-plateforme.

Activités de prise en main de la micro:bit

I

Nous allons voir ici des petits exercices permettant de se familiariser avec la carte micro:bit. Tous ces exercices pourront être faits avec la carte seule mais aussi avec l'*émulateur en ligne*

1. Activité 1

Analyse

Analysez ce code. Que fait-i l ?

```
1 from microbit import *
2
3 while True:
4     display.show("1")
5     sleep(500)
6     display.show(" ")
7     sleep(500)
```

Faîtes-le fonctionner sur la carte ou le simulateur en ligne.

Exercice 1 :

modifiez le programme ci-dessus afin qu'il compte en boucle de 0 jusqu'à 9

Indication : on pourra utiliser la commande `str(i)` qui transforme le nombre **i** en texte. Ex : **str(5)** renvoie **"5"**

2. Activité 2 : jouer avec les pixels

Analyse

Analysez ce code. Que fait-i l ?

```
1 from microbit import *
2
3 for x in range(5):
4     display.set_pixel(x,0,9)
5     sleep(500)
```

Explications : la fonction set_pixel allume un point sur l'écran. Elle prend 3 paramètres :

- les deux premiers sont l'abscisse et l'ordonnée du point (le point de coordonnées 0,0 étant en haut à gauch de l'écran)
- le dernier paramètre est la luminosité du point entre 0 et 9 : 0 signifie que le point est éteint et 9 est la luminosité maximale.

Faîtes-le fonctionner sur la carte ou le simulateur en ligne.

Exercice 2 :

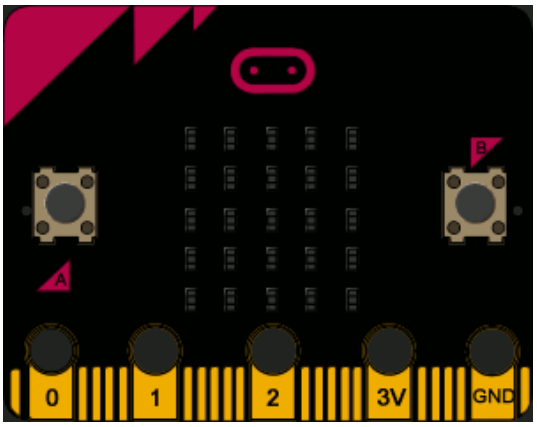
Modifiez ce programme afin qu'il allume la colonne centrale

Exercice 3 :

Modifiez ce programme afin qu'il allume successivement tous les pixels de l'écran.

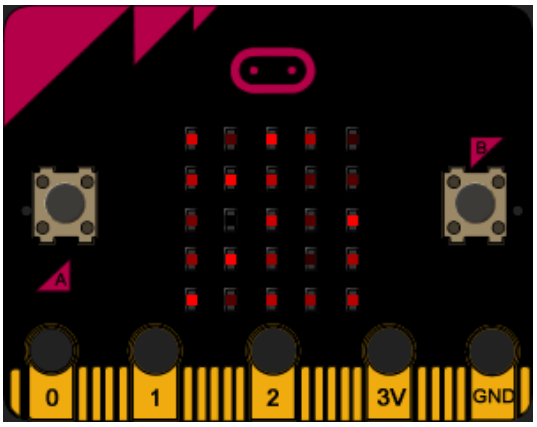
Indication : On pourra utiliser 2 boucles for imbriquées l'une dans l'autre. Pensez à changer le nom de la variable de la seconde boucle !

Exercice 4 :



Modifiez le programme précédent afin qu'il allume tous les pixels colonne par colonne, donc en 5 étapes.

Exercice 5 : le ciel étoilé



Modifiez ce programme afin d'obtenir un affichage avec des pixels dont l'illumination est aléatoire.

Indice : pour obtenir un nombre aléatoire entre 0 et 9 :

- importez la fonction **randint** depuis la librairie **random** : `from random import randint`
- utilisez `randint(0,9)` pour choisir un nombre aléatoire entre 0 et 9

Exercice 6

Analysez la fonction suivante :

```
def carre(l, color):
    for x in range(5-l,l):
        for y in range(5-l,l):
            display.set_pixel(x,y,color)
```

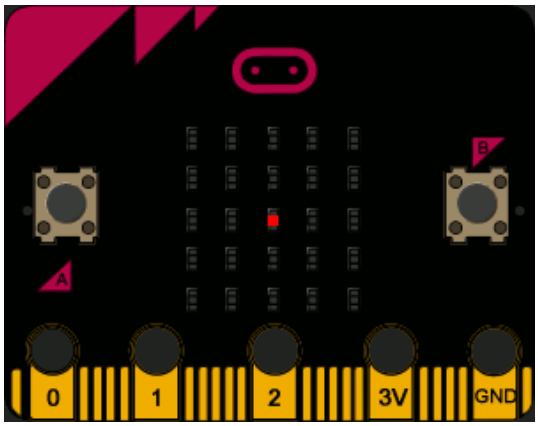
Dessinez sur votre feuille (1 carreau / pixel) l'écran de la carte lorsque l'on invoque

- `carre(3,9)`
- `carre(4,9)`
- `carre(5,9)`

Vous avez donc 3 écrans à représenter.

Une fois le travail réalisé sur papier, vérifiez-le à l'aide d'un programme sur simulateur ou sur la carte.

Exercice 7 :



En utilisant la fonction **carre()** ci-dessus, écrire un programme le plus concis possible réalisant l'animation ci-contre.

3. Activité 3 : jouer avec les boutons

Exercice 8 :

Étudiez le code suivant :

```
from microbit import *

compteur = 0
while True:
    if button_a.was_pressed():
        display.show(str(compteur))
        sleep(1000)
        display.clear()
```

Que fait-il ? vérifiez votre réponse en le testant sur la carte ou le simulateur en ligne.

Exercice 9 :

Modifiez le programme ci-dessus pour

- que le nombre augmente de 1 à chaque appui sur le bouton A.
- que le nombre revienne à 0 lorsqu'on appuie sur le bouton B.
- que le nombre reste affiché en permanence

Exercice 10 :

Étudiez le code suivant :

```
from microbit import *

x = 0
y = 0

while True:
    display.set_pixel(x,y,0)
    if button_a.was_pressed():
        x = x - 1
    if button_b.was_pressed():
        x = x + 1
    x = max(0, min(x, 4))
    display.set_pixel(x,y,9)
    sleep(20)
```

Que fait-il ? vérifiez votre réponse en le testant sur la carte ou le simulateur en ligne.

Expliquez le rôle de la ligne `x = max(0, min(x, 4))`

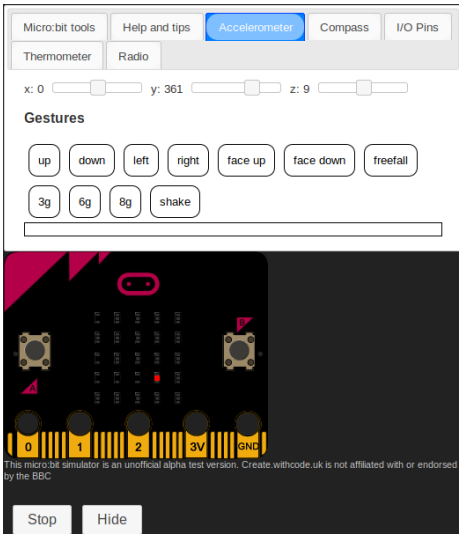
Exercice 11 :

Adaptez le programme ci-dessus afin que le point puisse se déplacer verticalement. On va pour cela utiliser l'accéléromètre de la carte :

- Pour récupérer l'information d'orientation de la carte selon l'axe y, utilisez la ligne `dy = accelerometer.get_y()`
- Si la variable **dy** est plus grande que 600, le point sera sur la 5eme ligne
- Si la variable **dy** est plus grande que 300, le point sera sur la 4eme ligne
- Si la variable **dy** est plus petit que -600, le point sera sur la 1ère ligne
- Si la variable **dy** est plus grande que -300, le point sera sur la 2ème ligne
- sinon, la carte est à peu près horizontale et le point sera su la ligne centrale.

Remarque : Tester le code avec le simulateur

Vous pouvez simuler l'orientation de la carte microbit avec le simulateur en allant dans l'onglet *Accelerometer* puis en déplaçant le curseur *y*. Si vote exercice est correctement réalisé, vous devriez alors voir le point rouge se déplacer verticalement.



Exercice 12 :

Modifiez le programme précédent afin de déplacer le pixel exclusivement avec l'accéléromètre. Selon que l'on penche la carte de droite à gauche, ou d'avant en arrière, le point se déplace sur la matrice LED sans avoir recours aux boutons A et B qui deviennent obsolètes.

4. Activité 4 : Afficher des images

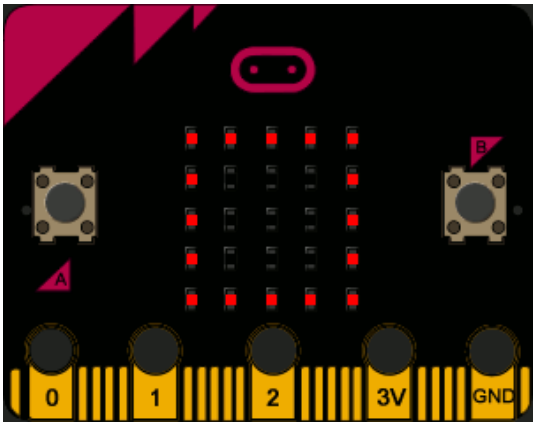
Exercice 13 :

Observez le code suivant. Il réalise une animation.

```
from microbit import *

imgs = [
    Image('90000:00000:00000:00000:00000:'),
    Image('90000:09000:00000:00000:00000:'),
    Image('90000:09000:00900:00000:00000:'),
    Image('90000:09000:00900:00090:00000:'),
    Image('90000:09000:00900:00090:00009:')
]
display.show(imgs, delay=500,loop=True)

```

Expliquez comment sont codées les images ?

Exercice 14 : Réalisez vous même votre propre animation



Recréez l'animation ci-contre sur votre carte. Vous remarquerez que les pixels ne sont pas toujours allumés à pleine intensité !

Exercice 15 :

Observez le fonctionnement du code suivant :

```
from microbit import *
from random import choice

mesImages = [Image('00000:00000:00900:00000:00000:'),
        Image('00009:00000:00000:00000:90000:'),
        Image('00009:00000:00900:00000:90000:'),
        Image('90009:00000:00000:00000:90009:'),
        Image('90009:00000:00900:00000:90009:'),
        Image('90009:00000:90009:00000:90009:')]
rolled = choice(mesImages)
display.show(rolled)

```

Remarquez l'emploi de la fonction **choice()** que l'on a importé de la librairie **random**. Elle permet d'extraire une image au hasard depuis la liste *mesImages*.

Modifiez ce programme afin qu'un appui sur le bouton A face rouler le dé.

Exercice 16 : Pierre feuille ciseaux

En utilisant la technique de l'exercice précédant, réalisez un jeu de Pierre-Feuille-Ciseaux. Un appui sur le bouton A affichera une de ces 3 figures au hasard. Vous créerez les images en utilisant la fonction **Image()** comme dans l'exercice précédent.

Mini-projets sur Microbit

II

1. Réaliser un compteur - niveau facile

L'idée dans ce mini projet est de réaliser un simple compteur. Chaque appui sur le bouton A incrémentera le compteur. Un appui sur le bouton B décrémentera le compteur. La problématique réside dans l'affichage du compteur de manière à ce qu'il tienne sur un seul écran de 25 LEDs. Jusqu'à combien peut-on compter ?

Ce mini projet est tout à fait accessible au niveau SNT.

Méthode : Première version

Dans cette version on affiche le compteur sous forme de chiffres. On peut donc compter jusqu'à 9, au delà, l'affichage n'est plus très exploitable...

```
from microbit import *

c=0
while True:
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    display.show(str(c))
```

Méthode : Seconde version

Nous avons 25 LEDs donc il ne doit pas être très difficile de compter jusqu'à 25 ! Voici donc une seconde version un peu plus intéressante car elle introduit deux boucles pour imbriquées. On peut aussi imaginer une variante avec un modulo et une division entière. Il y a donc une variété de solutions et des discussions intéressantes même pour un sujet aussi trivial !

```
from microbit import *

c=0

def affiche(c):
    k=0
    for y in range(5):
        for x in range(5):
            k+=1
            display.set_pixel(x, y, 9 if k<=c else 0)

while True:
```

```
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    affiche(c)
```

Méthode : Troisième version

Chaque LED possède 9 niveaux de couleur. Il n'est pas aisé de distinguer deux niveaux consécutifs mais on peut facilement distinguer entre rien, à moitié allumé et complètement allumé, ce qui fait deux niveaux d'illumination pour 25 LEDs donc un compteur jusqu'à 50 par écran ! Le projet commence à se compliquer...

```
from microbit import *

c=0

def affiche(c):
    k=-1
    for y in range(5):
        for x in range(5):
            k+=2
            if k<c:
                color=9
            elif k==c:
                color=4
            else:
                color=0
            display.set_pixel(x, y, color)

while True:
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    affiche(c)
```

Méthode : Dernière version

Une LED est allumée ou éteinde, donc 1 ou 0... on arrive à la représentation binaire des nombres avec 2^25 soit plus de 33 millions de possibilités ! !

C'est l'occasion d'introduite aux élèves le principe de comptage en binaire. L'algorithme de conversion décimal-binaire ne sera pas exploité ici : on utilisera la fonction **bin**() de MicroPython à cet effet, mais cela peut être une évolution possible pour les plus forts...

```
from microbit import *

c=0

def affiche(c):
    x=0
    y=0
    binaire=bin(c)[2:]
```

```
    nbChiffres = len(binaire)
    display.clear()
    for i in range(nbChiffres):
        digit = binaire[nbChiffres-i-1]
        display.set_pixel(x,y,int(digit)*9)
        x+=1
        if x==5:
            x=0
            y+=1

while True:
    if button_a.was_pressed():
        c += 1

    if button_b.was_pressed():
        c=0

    affiche(c)
```

2. Jeu de Pierre Feuille Ciseaux

Dans ce mini projet réalisable au niveau SNT, on met en oeuvre un jeu de Pierre Feuille Ciseaux sur deux cartes : chaque joueur doit secouer sa carte 3 fois avant que la carte fasse un choix et l'affiche.

Méthode : Première version

Cette première version est assez simple et met en avant les capacités graphiques de la matrice de LEDs.

```
# Olivier Lecluse
# 8 mars 2019

from microbit import *
from random import randint

pierre = Image("00900:"
               "09990:"
               "99599:"
               "09990:"
               "00900")
feuille = Image("99900:"
          "90090:"
          "90009:"
          "90009:"
          "99999")
ciseaux = Image("96009:"
                "69090:"
                "00900:"
                "69090:"
                "96009")

nbSecousses = 0
while True:
    if accelerometer.was_gesture("shake"):
        nbSecousses +=1
        display.show(str(nbSecousses))
    if nbSecousses == 3:
        choix = randint(1,3)
```

```
        if choix == 1:
            display.show(pierre)
        elif choix == 2:
            display.show(feuille)
        elif choix == 3:
            display.show(ciseaux)
        nbSecousses = 0
```

Méthode : Seconde version

Dans cette seconde version, nous allons tricher ! la carte de la victime enverra silencieusement par radio son choix à la carte tricheur qui sera donc assuré de gagner si celui-ci appuie sur les deux boutons A et B à partir de la seconde secousse. La pauvre victime n'y verra que du feu.

La morale de l'histoire est qu'il est toujours intéressant d'avoir accès au code source et qu'il ne faut pas croire aveuglément les boites noires !

Voici la version victime

```
# Olivier Lecluse
# 8 mars 2019
# Shifumi version victime

from microbit import *
from random import randint
import radio
radio.on()
radio.config(group=2)

pierre = Image("00900:"
               "09990:"
               "99599:"
               "09990:"
               "00900")
feuille = Image("99900:"
           "90090:"
           "90009:"
           "90009:"
           "99999")
ciseaux = Image("96009:"
                "69090:"
                "00900:"
                "69090:"
                "96009")

nbSecousses = 0
triche = 1
while True:
    if accelerometer.was_gesture("shake"):
        nbSecousses +=1
        if nbSecousses == 1:
            choix = randint(1,3)
            radio.send(str(choix))
        display.show(str(nbSecousses))
    if nbSecousses == 3:
        if choix == 1:
            display.show(pierre)
        elif choix == 2:
```

```
            display.show(feuille)
        elif choix == 3:
            display.show(ciseaux)
        nbSecousses = 0
```

Et voici la version à flasher sur la carte du tricheur !

```
# Olivier Lecluse
# 8 mars 2019
# shifumi version tricheur

from microbit import *
from random import randint
import radio
radio.on()
radio.config(group=2)

pierre = Image("00900:"
               "09990:"
               "99599:"
               "09990:"
               "00900")
feuille = Image("99900:"
           "90090:"
           "90009:"
           "90009:"
           "99999")
ciseaux = Image("96009:"
                "69090:"
                "00900:"
                "69090:"
                "96009")

nbSecousses = 0
triche = 1
while True:
    incoming = radio.receive()
    if incoming:
        triche = int(incoming)
    if button_a.was_pressed() and button_b.was_pressed():
        choix = triche%3 + 1
    if accelerometer.was_gesture("shake"):
        nbSecousses +=1
        if nbSecousses == 1:
            choix = randint(1,3)
        display.show(str(nbSecousses))
    if nbSecousses == 3:
        if choix == 1:
            display.show(pierre)
        elif choix == 2:
            display.show(feuille)
        elif choix == 3:
            display.show(ciseaux)
        nbSecousses = 0
```

3. Simulation d'un feu de chantier pour la circulation automobile

Fondamental : Matériel nécessaire

Pour cette activité, nous utiliserons

- 2 cartes micro:bit avec cables croco ou shield pour connecter des composants extérieurs (ici des LED)
- 4 LEDs (2 rouges et 2 oranges)
- 2 résistances de protection de 220Ohms
- une plaquette de prototypage (breadboard) ou bien un modèle de feu imprimé en 3D.

Fondamental : Méthode alternative sans matériel supplémentaire

Si vous ne disposez que des cartes sans possibilités de raccorder des LED sur les broches d'entrée/sortie, pas de problèmes : vous pourrez réaliser cette activité en utilisant la matrice LED de la carte pour afficher l'état des feux en remplacement des LED rouges et orange.

Je donnerai pour chaque étape de l'activité les codes correspondants aux deux situations : avec et sans feux LEDs.

3.1. Version 0

Consigne

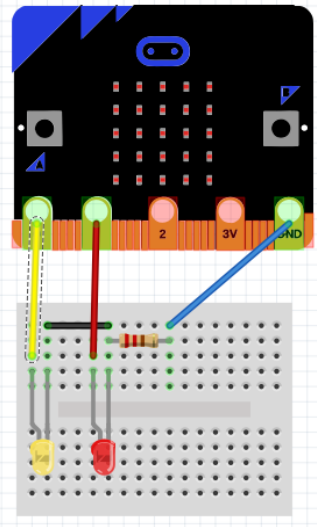
Créer une simulation de feu dans laquelle

- un feu comporte 2 LEDs (orange et rouge) connectées sur les broches 0 et 1 de la carte. En l'absence de LEDs, afficher le feu sous forme d'un carré de 4 pixels en haut à droite pour rouge et en bas à droite pour orange (une valeur de pixels égale à 4 simulera une couleur différente).
- le feu change toutes les 10 secondes
- un compte à rebours s'affiche sur l'écran

Simulation : Utilisation du simulateur en ligne

Cette version 0 peut être réalisée sur le *simulateur en ligne* en utilisant les broches 5 et 6 pour simuler les 2 LED du feu ou bien en utilisant uniquement la matrice de LED.

Méthode : Le montage en cas d'utilisation des LEDs



Pour cette activité, nous utiliserons

- 2 LEDs (une rouge et une orange)
- une résistance de protection de 220Ohms
- une plaquette de prototypage (breadboard) ou bien un modèle de feu imprimé en 3D.

Complément : Indication pour la version avec LED

On pourra utiliser les fonctions ci-dessous permettant de simuler un feu rouge, un feu orange et l'affichage du temps restant

```
def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)

def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)

def afficheAttente(attente):
    display.show(str(attente))
```

Complément : Indication pour la version sans LED

On pourra utiliser les fonctions ci-dessous permettant de simuler un feu rouge, un feu orange et l'affichage du temps restant

```
def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)

def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
```

```
15            display.set_pixel(x,y,4)
16
17 def afficheAttente(attente):
18     for x in range(2):
19         for y in range(5):
20             display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)
```

Corrigé

Voici le corrigé pour la version *avec LEDs* *(cf. p.31)* et *sans LEDs* *(cf. p.31)* .

3.2. Version 1 : Passage à 2 feux

Il y a deux façons d'aborder la suite de cette activité selon le niveau des élèves et la progression Python dans l'année :

- on demande aux élèves de construire le projet eux même
- on fournit un code opérationnel et on pose des questions aux élèves pour les faire réfléchir en mode débranché

Dans la suite de l'activité, je me placerai dans le second scénario : le code version 1 est fourni aux élèves et une démonstration leur est faite du dispositif fonctionnel d'un bout à l'autre de la salle de classe. Il est alors possible de mener la suite de cette activité en mode partiellement débranché en demandant aux élèves de répondre aux questions sur feuille puis passer à la phase de réalisation si vous avez accès à une salle info avec le matériel adéquat.

Objectif de l'activité

Dans cette partie, nous allons étudier un système de 2 feux connectés dans lequel

- les feux sont en opposition !
- Les 2 feux affichent le même compte à rebours

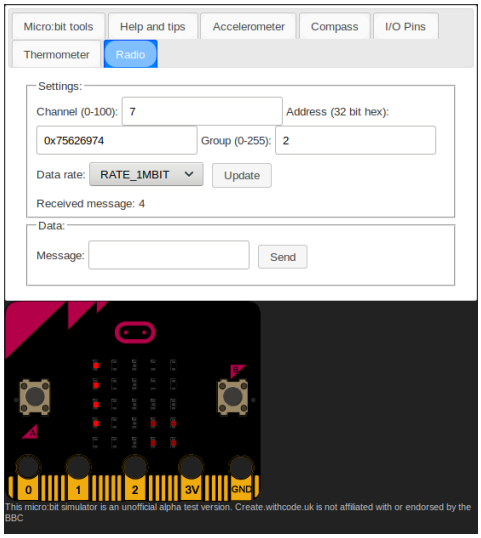
Consigne

Nous allons travailler sur les programmes suivants : étudiez la version *avec LED* *(cf. p.32)* ou *sans LED* *(cf. p. 33)* selon votre configuration (la version sans LED fonctionne sur la carte seule en utilisant l'affichage de la matrice LED pour simuler les feux) puis répondez sur feuille aux questions suivantes :

- Les programmes sur les 2 cartes sont-ils identiques ? Pourquoi ?
- Identifiez le programme maître et le programme esclave.
- Décrire le protocole de communication entre les cartes
- Est-on certain de la robustesse du système ?
    - les feux peuvent-ils être tous les deux au rouge ? Est-ce gênant ?
    - les feux peuvent-ils être tous les deux au vert ? Est-ce gênant ?
    - quelles sont les conséquences du plantage d'un des feux ?

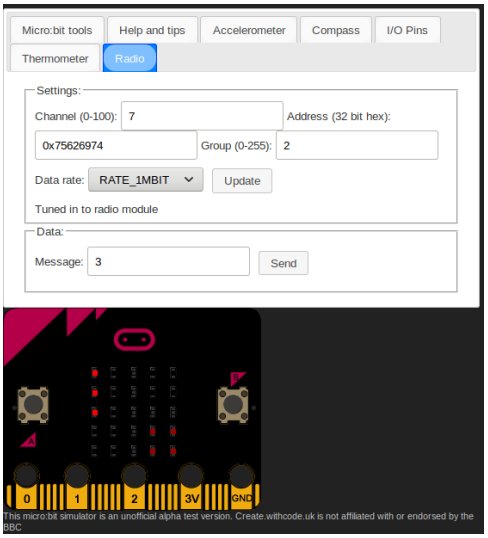
Simulation : Utilisation du simulateur en ligne

Ces deux programmes peuvent être simulés dans *le simulateur en ligne*, mais l'un après l'autre indépendamment.



Pour voir les messages envoyés par le programme maître, réglez les paramètres de l'onglet radio comme indiqué sur la capture d'écran ci-contre.

Sur le programme esclave, vous pourrez utiliser le champ *Message* pour simuler l'envoi d'une commande par le maître (essayez d'envoyer 3 ou VERT et observez le comportement de la carte).



Méthode : Eléments de réponse

Les deux feux peuvent être au rouge simultanément le temps que l'esclave reçoive une consigne du maître mais ce n'est pas gênant. Il n'est pas possible cependant du fait des valeurs d'initialisation de la variable *feuRouge* que les deux feux se retrouvent orange en même temps ce qui est plutôt rassurant.

Pour le moment le protocole de communication est simple : Il va toujours du maître vers l'esclave et utilise les mots suivants :

- **VERT** : demande à l'esclave de passer à l'orange
- **ROUGE** : demande à l'esclave de passer au rouge
- **1** à **9** : Valeur du compte à rebours

Il y a tout de même une faille si un feu est défaillant alors que l'autre laisse passer les voitures. Une absence de feu peut laisser croire aux usager que la voie est libre et ils vont rencontrer des véhicules venant en face. Le dispositif n'est donc pas très robuste.

Complément : Version 1 bis : identique à la version 1 mais le feu orange clignote !

Cette déclinaison de la version 1 n'est pas si simple à mettre en œuvre car elle requiert d'éviter d'avoir recours à la fonction **sleep()** qui bloque l'exécution du programme. Il faut donc envisager la gestion du temps autrement. A réserver aux élèves les plus expérimentés !

La méthode utilise une variable *clignotant* qui mémorise toutes les secondes la valeur de l'horloge donnée par **ticks_ms()**. Ensuite, le clignotement est réalisé dans la fonction **afficheOrange()** par le code suivant ou on utilise le temps écoulé depuis la dernière mémorisation de la valeur *clignotant* pour déterminer la luminosité du pixel. Le maximum de luminosité est donné à l'instant *clignotant* + 500ms.

```
# orange clignotant
    couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,couleur)
```

Voir le code complet en *version LED* *(cf. p.35)* et *sans LED* *(cf. p.37)* .

3.3. Version 2 : Le véhicule prioritaire

Consigne

Un véhicule prioritaire arrive à un feu ce qui doit le faire basculer au vert immédiatement.

- Imaginer par quel(s) moyen(s) on peut interagir avec la carte pour simuler cette situation ? décrire l'interaction homme-machine.
- la manière de prendre en compte l'arrivée du véhicule prioritaire est-elle identique selon le feu qui reçoit le signal ?
- situation A : le véhicule prioritaire arrive au feu maître. Modifier le programme maître afin qu'un appui sur le bouton A passe le feu maître à l'orange et l'autre feu au rouge.
- situation B : le véhicule prioritaire arrive à l'autre feu. Que faut-il modifier pour prendre en compte cette situation ?
- Décrire les modifications apportées au protocole de communication

Remarque

Dans ce scénario, les élèves n'ont pas à écrire plus de 4 lignes de python pour les deux dernières questions, ce qui est tout à fait abordable pour un élève de seconde en *SNT*, même si au final, le projet global est assez ambitieux.

Éléments de réponse

Il y a plusieurs moyens d'interagir avec la carte pour signaler la présence d'un véhicule prioritaire. Le plus simple est d'utiliser le bouton A de la carte, mais on peut imaginer d'autres scénarios :

- utiliser l'accéléromètre pour détecter une secousse
- communication radio avec une autre carte microbit
- souffler sur la carte qui détectera une élévation soudaine de la température
- approcher un aiment qui sera détecter par le magnétomètre

Pour la prise en charge de l'arrivée du véhicule prioritaire, les deux feux n'ont pas le même rôle : l'un envoie des ordres (maître) et le second obéi (esclave). Il n'y a pas pour le moment de communication de l'esclave vers le maître. Il faudra ajouter cette fonctionnalité pour permettre à un véhicule prioritaire de déclencher le feu esclave.

Concernant le protocole de communication, il n'y a pas de changements pour la situation A : en effet le maître contrôle le feu esclave et il peut choisir de le passer au rouge quand bon lui semble.

Par contre, lorsque le véhicule prioritaire arrive au feu esclave il est nécessaire de modifier le protocole de communication afin de permettre à l'esclave de faire un signalement au maître. On utilise ici le message *PRIO* que l'esclave envoie au maître pour lui demander le passage.

Méthode : Corrigé situation A : un appui sur le bouton A passe le feu maître à l'orange et l'autre au rouge.

Ici seul le code maître doit être actualisé. L'esclave n'est pas concerné par l'arrivée du véhicule prioritaire au feu maître car de toute manières, c'est le feu maître qui commande l'autre feu. Le code maître est modifié en ajoutant ces lignes au début de la boucle principale :

```
# Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")
```

Voir le code complet en *version LED (cf. p.39)* et *sans LED (cf. p.41)* .

Méthode : Corrigé situation B : le véhicule prioritaire arrive au feu esclave.

Ici, nous devons ajuster les deux codes afin d'établir une communication dans le sens esclave vers maître.

Code pour le feu maître. 5 lignes ont été ajoutées pour écouter les communications en provenance de l'esclave.

```
# Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
        feuRouge = True
        attente = DUREE
        radio.send("VERT")
```

Code pour le feu esclave. Deux lignes ont été ajoutées pour envoyer le signal **PRIO** au maître. Il enverra alors l'ordre à l'esclave de passer au vert.

```
# Arrivee vehicule prioritaire
    if button_a.was_pressed():
        radio.send("PRIO")
```

Voir le code complet en *version LED (cf. p.43)* et *sans LED (cf. p.44)* .

3.4. Version 3 : Améliorer la robustesse du dispositif

Consigne

- Quel scénario proposeriez-vous afin de renforcer la robustesse (tolérance aux pannes) du dispositif et renforcer la sécurité ?
- Proposez une modification des programmes en ce sens

Méthode : Corrigé scénario A

Afin de renforcer la fiabilité du dispositif, une communication régulière peut être établie du maître vers l'esclave. Si l'esclave ne reçoit pas de nouvelles du maître, il passe au rouge et affiche une figure triste. Pour ce faire, dans le code du feu maître, 4 lignes ont été ajoutées :

```
# Robustesse du dispositif :
    # Le maitre envoie l'etat du feu a l'esclave
    # toutes les secondes
    if feuRouge :
        radio.send("VERT")
    else:
        radio.send("ROUGE")
```

Pour le feu esclave, voici les lignes impactées. On utilise ici la fonction **ticks_ms()** afin de savoir combien de millisecondes se sont écoulées depuis la dernière transmission du feu maître. Si ce temps dépasse 2 secondes, l'esclave se met en sécurité.

```
# Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
            lastTransm = ticks_ms()  # message recu
        elif incoming == "ROUGE" :
            feuRouge = True
            lastTransm = ticks_ms()  # message recu
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Mise en securite si le maitre plante
    if ticks_ms() - lastTransm > 2000 :
        # Plus de deux secondes sans nouvelles du maitre
        feuRouge = True
        display.show(Image.SAD)
```

Voir le code complet en *version LED (cf. p.46)* et *sans LED (cf. p.48)* .

Méthode : Corrigé scénario B

Pour améliorer encore la fiabilité du dispositif, une communication régulière (envoi du message ALIVE) est établie de l'esclave vers le maître. Si le maître ne reçoit pas de nouvelles de son esclave, il passe au rouge et affiche une figure triste.

Voici les impacts sur le code maître de cette dernière modification :

On commence par ajouter une variable qui mémorise l'instant de la dernière transmission de l'esclave

```
lastTransm = ticks_ms() # derniere transmission de l'esclave
```

Ensuite, dans la boucle principale, on met le maître en sécurité (feu rouge et image triste) si l'esclave n'a pas donné signe de vie depuis plus de 2 secondes.

```
    # Mise en securite si l'esclave plante
    if ticks_ms() - lastTransm > 2000:
        # Plus de deux secondes sans nouvelles de l'esclave
        feuRouge = True
        display.show(Image.SAD)
```

Pour le feu esclave, on ajoute ue nouvelle variable qui mémorise le dernier envoi du message *ALIVE* :

```
lastAlive = ticks_ms()  # derniere transmission ALIVE de l'esclave
```

Ensuite, on envoie le message ALIVE toutes les deux secondes dans la boucle principale :

```
    # L'esclave envoie un message au maitre toutes les secondes
    if ticks_ms() - lastAlive > 1000:
        radio.send("ALIVE")
        lastAlive = ticks_ms()
```

Voir le code complet en *version LED (cf. p.51)* et *sans LED (cf. p.53)* .

3.5. Version finale

L'usage de la fonction **sleep()** n'est vraiment pas recommandable sur le feu maître car pendant qu'il est en sommeil il n'est pas à l'écoute des événements extérieurs et ne peut donc pas réagir en temps réel. La version proposée ci-dessous banni le **sleep()** au profit de la fonction **ticks_ms()** déjà rencontré. Il y a une meilleure réactivité du dispositif mais au prix d'un code légèrement plus compliqué, ce qui peut rebuter des élèves débutants en programmation.

Voir le code complet en *version LED (cf. p.56)* et *sans LED (cf. p.58)* .

3.6. Pour aller plus loin

Quelques idées afin de prolonger ce projet pour ceux qui le souhaitent

- Ajouter un délai de sécurité durant lequel les 2 feux sont rouges pour laisser le temps aux véhicules engagés de libérer la voie
- Réaliser la télécommande du chef de chantier qui
    - affiche quel feu est passant
    - permet de commander les 2 feux avec les boutons A et B
- Ajouter un 3ème feu. Quelles modifications cela implique t-il ? Proposer un protocole de communication gérant une configuration à 3 (voire plus) feux.

4. le jeu de bataille navale - Niveau intermédiaire

Dans cette simulation de jeu de bataille navale, nous utiliserons deux cartes. La première sera l'ordinateur et contiendra deux bateaux placés aléatoirement et la seconde sera pour le joueur qui essayera de couler les bateaux. La partie se termine quand les bateaux sont coulés.

La liaison radio est mise à contribution pour la communication du début de la partie, de la fin de partie, les tirs et les résultats des tirs.

Ce projet commence à être ambitieux pour une réalisation dans le cadre de SNT.

Méthode : Version ordinateur

- Le bouton A permet de décider un autre placement
- Le bouton B permet de démarrer la partie

Des bruitages sont joués lorsqu'un haut-parleur est connecté sur pin0.

```
# Olivier Lecluse
# 8 mars 2019
# version ordi

from random import randint
from microbit import *
import radio,music

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

boats = []
gameOn = False

def placer():
    global boats
    boats=[]
    h=randint(0,1)
    if h==0: # bateau de 3 horizontal
        x3=randint(0,2)
        y3=randint(0,4)
        boats.append((x3,y3))
        boats.append((x3+1,y3))
        boats.append((x3+2,y3))
    else:    # bateau de 3 vertical
        x3=randint(0,4)
        y3=randint(0,2)
        boats.append((x3,y3))
        boats.append((x3,y3+1))
        boats.append((x3,y3+2))

    cont = True
    while cont :
        if h==0: # bateau de 2 verrtical
            x2=randint(0,4)
            y2=randint(0,3)
            if (x2,y2) not in boats and (x2, y2+1) not in boats:
                boats.append((x2,y2))
                boats.append((x2,y2+1))
                cont=False
        else: # bateau de 2 horizontal
            x2=randint(0,3)
            y2=randint(0,4)
            if (x2,y2) not in boats and (x2+1, y2) not in boats:
                boats.append((x2,y2))
```

```
                boats.append((x2+1,y2))
                cont=False

def afficher():
    display.clear()
    for b in boats:
        display.set_pixel(b[0],b[1],9)

placer()
afficher()
while True:
    if button_a.was_pressed() and not gameOn:
        placer()
        afficher()
    if button_b.was_pressed() and not gameOn:
        gameOn = True
        radio.send("GO")
        music.play(music.POWER_UP)
    if gameOn:
        # la partie est en cours

        # Gestion de l'arrivee d'un coup
        incoming = radio.receive()
        if incoming:
            try:
                hx,hy = incoming.split(',')
                hit=(int(hx),int(hy))
            except:
                continue
            if (hit) in  boats:
                boats.remove(hit)
                radio.send("HIT")
                music.pitch(1000, 500)
                color=0
            else:
                radio.send("MISS")
                music.pitch(200, 500)
                color=2
            display.set_pixel(hit[0], hit[1], color)

        # Fin de la partie
        if len(boats)==0:
            radio.send("GAME OVER")
            music.play(music.FUNERAL)
            gameOn=False

```

Méthode : Version Joueur

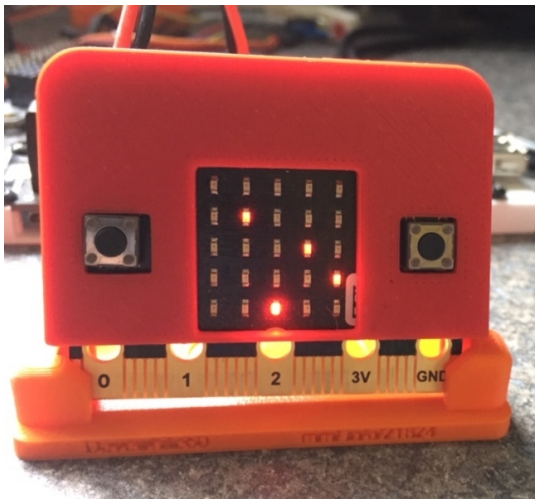
- Le bouton A déplace la cible verticalement
- Le bouton B déplace la cible horizontalement
- Une secousse déclenche le tir

```
# Olivier Lecluse
# 8 mars 2019
# version joueur

```

```
from microbit import *
import radio

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

nbHits = 0
gameOn = False
movePixColor = 5
missPixColor = 2
hitPixColor = 9

def deplacement():
    global x, y, lastX, lastY, lastPix
    # On rétablit l'ancien pixel
    display.set_pixel(lastX, lastY, lastPix)

    lastPix = display.get_pixel(x,y)
    lastX, lastY=x, y
    display.set_pixel(x,y,movePixColor)


while True:
    if not gameOn:
        # en atttente du placement des bateaux
        incoming = radio.receive()
        if incoming == "GO":
            gameOn = True
            display.show("Go")
            sleep(500)
            display.clear()
            nbHits=0
            x,y = 0,0 # position du tir
            lastPix = 0
            lastX, lastY=0,0
            display.set_pixel(0,0,movePixColor)
    else :
        # jouer la partie
        if button_b.was_pressed():
            x = (x+1)%5
            deplacement()
        if button_a.was_pressed():
            y = (y+1)%5
            deplacement()
        if accelerometer.was_gesture("shake"):
            message = "{},{}".format(x, y)
            radio.send(message)
            response = False
            while not response:
                incoming = radio.receive()
                if incoming == "HIT":
                    display.set_pixel(x,y,hitPixColor)
                    lastPix = hitPixColor
                    response = True
                    nbHits += 1
                elif incoming == "MISS":
                    display.set_pixel(x,y,missPixColor)
                    lastPix = missPixColor
```

```
                    response = True
                elif incoming == "GAME OVER":
                    display.set_pixel(x,y,hitPixColor)
                    lastPix = hitPixColor
                    display.show("WIN")
                    gameOn = False
                    response = True
                elif incoming:
                    response = True
        if nbHits == 5:
            gameOn = False
            display.show("WIN")

```

5. Jeu de tir : scudFighter - Niveau intermédiaire



scudFighter est un jeu de tir dans lequel on doit détruire son adversaire en lui lançant des missiles. Le jeu se joue sur 2 cartes micro:bit qui communiquent entre-elles via le module radio.

La manipulation est très simple :

- le bouton A permet de se déplacer vers la droite
- le bouton B permet de se déplacer vers la gauche
- un appui simultané sur A + B permet de lancer un missile

Complément : Programme complet

Vous trouverez ci-dessous le détail de fonctionnement du programme avec des explications sur les extraits de code les plus intéressants. Vous trouverez le *programme complet en suivant ce lien* *(cf. p.61)* .

5.1. Les variables globales utilisées

```
speed=1
accel=0.2
score=0
position=2
scuds = []
lastLoop = ticks_us()
```

speed gère la vitesse du jeu. Le jeu fait **speed** mouvements par secondes. A chaque coup gagnant ou perdant, speed augmente de la valeur **accel**. Le jeu va donc globalement de plus en plus vite.

position est l'abscisse du joueur (entre 0 et 4).

scuds est une liste de missiles. Chaque missile est un tuple de la forme **(x,y,d)** avec

- x : abscisse du missile
- y : ordonnée du missile
- d : direction du missile.
    - 1 = le missile descend
    - -1, le missile monte

lastLoop : gestion du temps. Afin d'avoir une bonne réactivité du jeu et un rythme régulier, on utilise pas (ou presque...) la commande **sleep()**, mais **ticks_us()** qui donne un repère temporel en microsecondes. `ticks_us()-lastLoop` permet donc de savoir le temps écoulé depuis la dernière action.

5.2. Fonctions simples

position_move

Cette fonction permet le déplacement du joueur. Elle agit sur la variable globale **position**.

```
1 def position_move(p):
2     global position
3     display.set_pixel(position,4,0)
4     position += p
5     position = max(min(4,position),0)
6     display.set_pixel(position,4,9)
```

fire

Cette fonction permet de tirer un missile. Il suffit pour cela d'ajouter une entrée à la liste **scuds**. Le mouvement est géré ailleurs, dans la fonction **process_fire** qui est la plus compliquée du projet.

```
1 def fire():
2     scuds.append((position,3,-1))
3     display.set_pixel(position,3,SCUDFRIENDCOLOR)
```

refresh

Lors de l'effacement de l'écran (affichage du score, animation explosion), il est nécessaire de réafficher le plateau de jeu : position joueur et missiles. C'est ce que fait la fonction **refresh**.

```
1 def refresh():
2     # On redessine l'affichage
3     display.clear()
4     display.set_pixel(position,4,9)
5     for s in scuds:
6         display.set_pixel(s[0],max(0,s[1]),SCUDFRIENDCOLOR if s[2]<0 else
  SCUDENMYCOLOR)
```

perdu

Lorsque l'on est touché par un missile, on accélère le jeu, on envoie l'information à l'autre carte par radio afin qu'elle actualise son score et on joue l'animation d'explosion. Il faut ensuite redessiner la scène de jeu.

```
def perdu():
    global speed
    speed += accel # Le jeu s'accelere !

    # envoi radio "HIT"
    radio.send("HIT")

    # animation
    display.clear()
    anim = [
    Image("99999:90009:90009:90009:99999"),
    Image("00000:06660:06060:06660:00000"),
    Image("00000:00000:00300:00000:00000"),
    Image("00000:06660:06060:06660:00000"),
    Image("99999:90009:90009:90009:99999")]
    display.show(anim, delay=100, loop=False, wait=True)
    refresh()
```

gagne

Lorsque l'on gagne, on accélère le jeu, on actualise le score et on l'affiche. Après un petit temps d'attente, on reprend le cours du jeu.

```
def gagne():
    global speed,score
    speed += accel # Le jeu s'accelere !
    score += 1
    # animation
    display.clear()
    display.scroll(score)
    sleep(500)
    refresh()
```

5.3. Traitement des événements

Le traitement des événements se fait dans trois fonctions nommées **process_*** . Ces fonctions sont au coeur du projet.

process_button

On gère ici l'appui sur les boutons. L'appel à la méthode **was_pressed()** vide la mémoire tampon qui mémorise si un bouton a été préssé. On on doit tester les boutons deux fois : pour l'appui simple et pour la détection de l'appui simultané A+B. On va donc mémoriser l'état des boutons dans des variables.

Le reste est simple puisqu'on invoque le tir ou le déplacement selon les boutons actionnés.

```
def process_button():
    # Gesion des boutons
    ba = button_a.was_pressed()
    bb = button_b.was_pressed()
    if ba and bb :
        fire()
    else:
        if ba:
            position_move(-1)
        if bb:
```

```
11              position_move(1)
```

process_radio

Il s'agit ici d'écouter les messages radio qui parviennent à la carte. Ils sont de 2 type :

- **HIT** : signifie que l'adversaire a été touché. On a alors gagné.
- **0->4** : Arrivée d'un missile. On l'ajoute à la liste **scuds** avec la bonne direction (1)

```
1  def process_radio():
2      global lastLoop
3      # gestion des messages radio
4      incoming = radio.receive()
5      if incoming:
6          if incoming == "HIT":
7              gagne()
8          elif len(incoming)==1:
9              # arrivee d'un missile
10             lastLoop=0
11             scuds.append((int(incoming),-1,1))
12             display.set_pixel(int(incoming),0,SCUDENMYCOLOR)
```

process_fire

C'est la fonction la plus complexe du projet. Elle doit faire avancer chaque missile, détecter les éventuelles collisions de ces derniers ainsi que si on a été touché.

On utilise deux variables de type **set** (ensembles) :

- setpos : les positions à venir de tous les missiles afin de détecter les colllisions de ces derniers : en effet, s lors du mouvement d'un missile, on se retrouve dans une position qui existe déjà, on détruit les missiles.
- removeScuds : l'ensemble des missiles qu'il faudra éliminer car ils se seront percutés ou qu'ils auront quitté le plateau de jeu. Une boucle en fin de fonction se chargera de ce nettoyage.

On calcule la position suivante du missile en tenant compte de sa direction par cette ligne.

`s1 = (s[0],s[1]+s[2],s[2])`

La détection de collision utilise **setpos** : `if s2 in setpos:`

S'il n'y a pas de collision :

- `if 0<=s1[1]<4:` on procède au déplacement du missile en actualisant la liste scuds et l'affichage
- `elif s1[1]==4:` on a été touché, on élimine le scud et on a perdu
- `else:` le missile quitte l'écran, on communique sa position à l'autre carte par radio.

Enfin on nettoie les scuds à supprimer en parcourant le set **removeScuds**. On supprime les missiles de la liste dans un `try...except` afin d'éviter les erreurs car on a été un peu laxistes dans les vérifications...

```
1  def process_fire():
2      setpos=set()
3      removeScuds=set()
4      for i,s in enumerate(scuds):
5          display.set_pixel(s[0],max(0,s[1]),0)
6          s1 = (s[0],s[1]+s[2],s[2]) # nouvelle position
7          s2 = (s1[0],s1[1],-s1[2])  # missile en collision eventuelle
8          if s2 in setpos:
```

```
            # collision  :autodestruction des 2 scuds
            removeScuds.add(s1)
            removeScuds.add(s2)
            removeScuds.add(s)
            removeScuds.add((s[0],s[1],-s[2]))
        else :
            if 0<=s1[1]<4:
                setpos.add(s1)
                # deplacement du scud
                display.set_pixel(s1[0],s1[1],SCUDFRIENDCOLOR if s1[2]<0 else
SCUDENMYCOLOR)
                scuds[i]=s1
            elif s1[1]==4:
                removeScuds.add(s)
                if s1[0] == position :
                    perdu()
            else :
                removeScuds.add(s)
                # envoi radio
                radio.send(str(s[0]))

    # Nettoyage de la liste des scuds
    for sr in removeScuds:
        display.set_pixel(sr[0],max(0,sr[1]),0)
        try:
            scuds.remove(sr)
        except:
            pass
```

5.4. La boucle principale

Pour finir, la boucle principale. Le code est très court car le gros du traitement se fait dans les fonctions que l'on a décrit précédemment. Il y a néanmoins la gestion du timing qui prend place dans cette partie :

```
delay = 1000000/speed
...
loop = ticks_us()
if(loop-lastLoop < delay):
   # attente 200ms pour la detection tir
   sleep(min((loop-lastLoop)//1000,200))
else :
   lastLoop = loop
   process_fire()
```

L'idée ici est d'attendre que le **delay** soit écoulé pour faire avancer les missiles.

Si le délai n'est pas encore écoulé, on fait une micro-sieste de 200ms afin d'imposer une vitesse maximale au jeu et de rendre l'appui A+B plus facile. Il faut en effet s'assurer lors d'un appui double que le bouton A et B soient testés dans une même boucle. Ce petit délai permet cela.

Contenus annexes

> Corrigé V0

```
from microbit import *

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)

    # Alternance du feu
    if attente == 0:
        feuRouge = not feuRouge
        attente = DUREE
```

> Corrigé V0 sans LED

```
from microbit import *

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
```

```
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)

    # Alternance du feu
    if attente == 0:
        feuRouge = not feuRouge
        attente = DUREE

```

> Version 1 avec LED

Méthode : Première carte

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Traitement des messages
```

```
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()
```

Méthode : Seconde carte

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge
        if feuRouge :
            radio.send("VERT")
        else:
            radio.send("ROUGE")
```

> Version 1 sans LED

Méthode : Première carte

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

```

Méthode : Seconde carte

```
from microbit import *
import radio

radio.config(group=2)
radio.on()
```

```
6
7 DUREE = 10  # 10 secondes
8 attente = DUREE
9 feuRouge = True
10
11 def afficheRouge():
12     for x in range(3,5):
13         for y in range(2):
14             display.set_pixel(x,y,9)
15     for x in range(3,5):
16         for y in range(3,5):
17             display.set_pixel(x,y,0)
18 def afficheOrange():
19     for x in range(3,5):
20         for y in range(2):
21             display.set_pixel(x,y,0)
22     for x in range(3,5):
23         for y in range(3,5):
24             display.set_pixel(x,y,4)
25 def afficheAttente(attente):
26     for x in range(2):
27         for y in range(5):
28             display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)
29
30 while True:
31     # Affichage Rouge / Orange
32     if feuRouge :
33         afficheRouge()
34     else :
35         afficheOrange()
36
37     # Gestion de l'attente
38     sleep(1000)
39     attente -= 1
40     afficheAttente(attente)
41     radio.send(str(attente))
42
43     # Alternance du feu
44     if attente == 0 :
45         attente=DUREE
46         feuRouge = not feuRouge
47         if feuRouge :
48             radio.send("VERT")
49         else:
50             radio.send("ROUGE")
51
```

> Version avec LED, clignotante

Méthode : Code feu Maître

```
1 from microbit import *
2 from time import ticks_ms
3 import radio
4
5 radio.config(group=2)
6 radio.on()
```

```
7
8 DUREE = 10  # 10 secondes
9 attente = DUREE
10 feuRouge = True
11 clignotant = ticks_ms()
12
13 def afficheRouge():
14     pin1.write_digital(1)
15     pin0.write_digital(0)
16
17 def afficheOrange():
18     pin1.write_digital(0)
19     # orange clignotant
20     if ticks_ms()-clignotant < 500:
21         pin0.write_digital(1)
22     else:
23         pin0.write_digital(0)
24
25 def afficheAttente(attente):
26     display.show(str(attente))
27
28 while True:
29     if feuRouge :
30         afficheRouge()
31     else :
32         afficheOrange()
33
34     # Action toutes les secondes
35     if ticks_ms()-clignotant > 1000:
36         clignotant = ticks_ms()
37         attente -= 1
38         afficheAttente(attente)
39         radio.send(str(attente))
40
41     # Alternance du feu
42     if attente == 0 :
43         attente=DUREE
44         feuRouge = not feuRouge
45         if feuRouge :
46             radio.send("VERT")
47         else:
48             radio.send("ROUGE")
```

Méthode : Code feu Esclave

```
1 from microbit import *
2 from time import ticks_ms
3 import radio
4
5 radio.config(group=2)
6 radio.on()
7
8 feuRouge = True
9 clignotant = ticks_ms()
10
11 def afficheRouge():
12     pin1.write_digital(1)
```

```
    pin0.write_digital(0)

def afficheOrange():
    pin1.write_digital(0)
    # orange clignotant
    if ticks_ms()-clignotant < 500:
        pin0.write_digital(1)
    else:
        pin0.write_digital(0)

def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming)==1:
            afficheAttente(int(incoming))

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Compteur temps à 0 toute les secondes
    if ticks_ms()-clignotant > 1000:
        clignotant = ticks_ms()
```

> Version sans LED, clignotante

Méthode : Code feu Maître

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True
clignotant = ticks_ms()

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
```

```
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    # orange clignotant
    couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,couleur)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)


while True:
    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Action toutes les secondes
    if ticks_ms()-clignotant > 1000:
        clignotant = ticks_ms()
        attente -= 1
        afficheAttente(attente)
        radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge
        if feuRouge :
            radio.send("VERT")
        else:
            radio.send("ROUGE")
```

Méthode : Code feu Esclave

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

feuRouge = True
clignotant = ticks_ms()

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
```

```
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    # orange clignotant
    couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,couleur)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)


while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming)==1:
            afficheAttente(int(incoming))

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Compteur temps à 0 toute les secondes
    if ticks_ms()-clignotant > 1000:
        clignotant = ticks_ms()
```

> Corrigé situation A avec LED : un appui sur le bouton A passe le feu maître à l'orange et l'autre feu au rouge.

Méthode : Code feu Maître

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
```

```
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge
        if feuRouge :
            radio.send("VERT")
        else:
            radio.send("ROUGE")
```

Méthode : Code feu Esclave

pas de modification par rapport à la version précédente

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
```

```
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()
```

> Corrigé situation A sans LED : un appui sur le bouton A passe le feu maître à l'orange et l'autre feu au rouge.

Méthode : Code feu Maître

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
```

```
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge
        if feuRouge :
            radio.send("VERT")
        else:
            radio.send("ROUGE")

```

Méthode : Code feu Esclave

pas de modification par rapport à la version précédente

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming) ==1:
```

```
            afficheAttente(int(incoming))

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

```

> Corrigé situation B avec LED : le véhicule prioritaire arrive au feu esclave.

Méthode : Code feu Maître

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
        feuRouge = True
        attente = DUREE
        radio.send("VERT")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)
```

```
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge
        if feuRouge :
            radio.send("VERT")
        else:
            radio.send("ROUGE")
```

Méthode : Code feu Esclave

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Arrivee vehicule prioritaire
    if button_a.was_pressed():
        radio.send("PRIO")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()
```

> Corrigé situation B sans LED : le véhicule prioritaire arrive au feu esclave.

Méthode : Code feu Maître

```
from microbit import *
```

```
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
        feuRouge = True
        attente = DUREE
        radio.send("VERT")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
```

```
        feuRouge = not feuRouge
        if feuRouge :
            radio.send("VERT")
        else:
            radio.send("ROUGE")

```

Méthode : Code feu Esclave

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
        elif incoming == "ROUGE" :
            feuRouge = True
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Arrivee vehicule prioritaire
    if button_a.was_pressed():
        radio.send("PRIO")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

```

> Corrigé situation A avec LED : Améliorer la robustesse du dispositif

Méthode : Code feu Maître

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
        feuRouge = True
        attente = DUREE
        radio.send("VERT")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge

    # Robustesse du dispositif :
    # Le maitre envoie l'etat du feu a l'esclave
```

```
    # toutes les secondes
    if feuRouge :
        radio.send("VERT")
    else:
        radio.send("ROUGE")
```

Méthode : Code feu Esclave

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

lastTransm = ticks_ms()  # derniere transmission du maitre
feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
            lastTransm = ticks_ms()  # message recu
        elif incoming == "ROUGE" :
            feuRouge = True
            lastTransm = ticks_ms()  # message recu
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Mise en securite si le maitre plante
    if ticks_ms() - lastTransm > 2000 :
        # Plus de deux secondes sans nouvelles du maitre
        feuRouge = True
        display.show(Image.SAD)

    # Arrivee vehicule prioritaire
    if button_a.was_pressed():
        radio.send("PRIO")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()
```

> Corrigé situation A sans LED : Améliorer la robustesse du dispositif

Méthode : Code feu Maître

```
from microbit import *
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
        feuRouge = True
        attente = DUREE
        radio.send("VERT")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
```

```
    attente -= 1
    afficheAttente(attente)
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge

    # Robustesse du dispositif :
    # Le maitre envoie l'etat du feu a l'esclave
    # toutes les secondes
    if feuRouge :
        radio.send("VERT")
    else:
        radio.send("ROUGE")
```

Méthode : Code feu Esclave

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

lastTransm = ticks_ms()  # derniere transmission du maitre
feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    display.set_pixel(2,3,0)
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
            lastTransm = ticks_ms()  # message recu
        elif incoming == "ROUGE" :
            feuRouge = True
```

```
            lastTransm = ticks_ms()  # message recu
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Mise en securite si le maitre plante
    if ticks_ms() - lastTransm > 2000 :
        # Plus de deux secondes sans nouvelles du maitre
        feuRouge = True
        display.clear()
        display.show(Image.SAD)
    else :
        # Arrivee vehicule prioritaire
        if button_a.was_pressed():
            radio.send("PRIO")

        # Affichage Rouge / Orange
        if feuRouge :
            afficheRouge()
        else :
            afficheOrange()
```

> Corrigé situation B avec LED : Améliorer la robustesse du dispositif

Méthode : Code feu Maître

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
lastTransm = ticks_ms()  # derniere transmission de l'esclave
feuRouge = True

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
```

```
        feuRouge = True
        attente = DUREE
        radio.send("VERT")
    # L'esclave est en vie :)
    if incoming == "ALIVE" :
        lastTransm = ticks_ms()

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Gestion de l'attente
    sleep(1000)
    attente -= 1
    afficheAttente(attente)
    radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge

    # Robustesse du dispositif :

    # Le maitre envoie l'etat du feu a l'esclave toutes les secondes
    if feuRouge :
        radio.send("VERT")
    else:
        radio.send("ROUGE")

    # Mise en securite si l'esclave plante
    if ticks_ms() - lastTransm > 2000:
        # Plus de deux secondes sans nouvelles de l'esclave
        feuRouge = True
        display.show(Image.SAD)

```

Méthode : Code feu Esclave

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

lastTransm = ticks_ms()  # derniere transmission du maitre
lastAlive = ticks_ms()  # derniere transmission ALIVE de l'esclave
feuRouge = True
attente=0

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)
def afficheOrange():
    pin1.write_digital(0)
```

```
    pin0.write_digital(1)
def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Traitement des messages du maitre
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
            lastTransm = ticks_ms()  # message recu
        elif incoming == "ROUGE" :
            feuRouge = True
            lastTransm = ticks_ms()  # message recu
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Robustesse du dispositif :
    # Mise en securite si le maitre plante
    if ticks_ms() - lastTransm > 2000 :
        # Plus de deux secondes sans nouvelles du maitre
        feuRouge = True
        display.show(Image.SAD)
    # L'esclave envoie un message au maitre toutes les secondes
    if ticks_ms() - lastAlive > 1000:
        radio.send("ALIVE")
        lastAlive = ticks_ms()

    # Arrivee vehicule prioritaire
    if button_a.was_pressed():
        radio.send("PRIO")

    # Affichage Rouge / Orange
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()
```

> Corrigé situation B sans LED : Améliorer la robustesse du dispositif

Méthode : Code feu Maître

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
lastTransm = ticks_ms()  # derniere transmission de l'esclave
feuRouge = True

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
```

```
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,4)
def afficheAttente(attente):
    display.set_pixel(2,3,0)
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
        feuRouge = True
        attente = DUREE
        radio.send("VERT")
    # L'esclave est en vie :)
    if incoming == "ALIVE" :
        lastTransm = ticks_ms()

    # Mise en securite si l'esclave plante
    if ticks_ms() - lastTransm > 2000:
        display.clear()
        # Plus de deux secondes sans nouvelles de l'esclave
        feuRouge = True
        display.show(Image.SAD)
    else:
        # Affichage Rouge / Orange
        if feuRouge :
            afficheRouge()
        else :
            afficheOrange()

        # Gestion de l'attente
        sleep(1000)
        attente -= 1
        afficheAttente(attente)
        radio.send(str(attente))

    # Alternance du feu
    if attente == 0 :
        attente=DUREE
        feuRouge = not feuRouge
```

```
74
75      # Le maitre envoie l'etat du feu a l'esclave toutes les secondes
76      if feuRouge :
77          radio.send("VERT")
78      else:
79          radio.send("ROUGE")
```

Méthode : Code feu Esclave

```
1  from microbit import *
2  from time import ticks_ms
3  import radio
4
5  radio.config(group=2)
6  radio.on()
7
8  lastTransm = ticks_ms()  # derniere transmission du maitre
9  lastAlive = ticks_ms()  # derniere transmission ALIVE de l'esclave
10 feuRouge = True
11 attente=0
12
13 def afficheRouge():
14     for x in range(3,5):
15         for y in range(2):
16             display.set_pixel(x,y,9)
17     for x in range(3,5):
18         for y in range(3,5):
19             display.set_pixel(x,y,0)
20 def afficheOrange():
21     for x in range(3,5):
22         for y in range(2):
23             display.set_pixel(x,y,0)
24     for x in range(3,5):
25         for y in range(3,5):
26             display.set_pixel(x,y,4)
27 def afficheAttente(attente):
28     display.set_pixel(2,3,0)
29     for x in range(2):
30         for y in range(5):
31             display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)
32
33 while True:
34     # Traitement des messages du maitre
35     incoming = radio.receive()
36     if incoming:
37         if incoming == "VERT" :
38             feuRouge = False
39             lastTransm = ticks_ms()  # message recu
40         elif incoming == "ROUGE" :
41             feuRouge = True
42             lastTransm = ticks_ms()  # message recu
43         elif len(incoming) ==1:
44             afficheAttente(int(incoming))
45
46     # Robustesse du dispositif :
47     # L'esclave envoie un message au maitre toutes les secondes
48     if ticks_ms() - lastAlive > 1000:
```

```
        radio.send("ALIVE")
        lastAlive = ticks_ms()

    # Mise en securite si le maitre plante
    if ticks_ms() - lastTransm > 2000 :
        # Plus de deux secondes sans nouvelles du maitre
        feuRouge = True
        display.show(Image.SAD)
    else:
        # Affichage Rouge / Orange
        if feuRouge :
            afficheRouge()
        else :
            afficheOrange()

    # Arrivee vehicule prioritaire
    if button_a.was_pressed():
        radio.send("PRIO")

```

> Version finale avec LED : ne pas bloquer la boucle avec sleep()

Méthode : Code feu Maître

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True
lastTransm = ticks_ms()  # derniere transmission ALIVE de l'esclave
clignotant = ticks_ms()

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)

def afficheOrange():
    pin1.write_digital(0)
    # orange clignotant
    if ticks_ms()-clignotant < 500:
        pin0.write_digital(1)
    else:
        pin0.write_digital(0)

def afficheAttente(attente):
    display.show(str(attente))

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
```

```
34          radio.send("ROUGE")
35
36      # Traitement des messages en provenance de l'esclave
37      incoming = radio.receive()
38      # Arrivee du vehicule priopritaire au feu esclave
39      if incoming == "PRIO" :
40          feuRouge = True
41          attente = DUREE
42          radio.send("VERT")
43      # L'esclave est en vie :)
44      if incoming == "ALIVE" :
45          lastTransm = ticks_ms()
46
47      # Affichage Rouge / Orange
48      if feuRouge :
49          afficheRouge()
50      else :
51          afficheOrange()
52
53      # Action toutes les secondes
54      if ticks_ms()-clignotant > 1000:
55          clignotant = ticks_ms()
56          attente -= 1
57          afficheAttente(attente)
58          radio.send(str(attente))
59
60          # Alternance du feu
61          if attente == 0 :
62              attente=DUREE
63              feuRouge = not feuRouge
64
65      # Robustesse du dispositif
66
67          # Le maitre envoie l'etat du feu a l'esclave toutes les secondes
68          if feuRouge :
69              radio.send("VERT")
70          else:
71              radio.send("ROUGE")
72
73      # Mise en securite si l'esclave plante
74      if ticks_ms() - lastTransm > 2000:
75          # Plus de deux secondes sans nouvelles de l'esclave
76          feuRouge = True
77          display.show(Image.SAD)
78
```

Méthode : Code feu Esclave

```
1 from microbit import *
2 from time import ticks_ms
3 import radio
4
5 radio.config(group=2)
6 radio.on()
7
8 feuRouge = True
9 clignotant = ticks_ms()
```

```
lastTransm = ticks_ms()  # derniere transmission du maitre
lastAlive = ticks_ms()  # derniere transmission ALIVE de l'esclave

def afficheRouge():
    pin1.write_digital(1)
    pin0.write_digital(0)

def afficheOrange():
    pin1.write_digital(0)
    # orange clignotant
    if ticks_ms()-clignotant < 500:
        pin0.write_digital(1)
    else:
        pin0.write_digital(0)

def afficheAttente(attente):
    display.show(str(attente))

while True:
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
            lastTransm = ticks_ms()
        elif incoming == "ROUGE" :
            feuRouge = True
            lastTransm = ticks_ms()
        elif len(incoming) ==1:
            display.show(incoming)

    # Robustesse du dispositif :
    # Mise en securite si le maitre plante
    if ticks_ms() - lastTransm > 2000 :
        # Plus de deux secondes sans nouvelles du maitre
        feuRouge = True
        display.show(Image.SAD)
    # L'esclave envoie un message au maitre toutes les secondes
    if ticks_ms() - lastAlive > 1000:
        radio.send("ALIVE")
        lastAlive = ticks_ms()

    # Arrivee vehicule prioritaire
    if button_a.was_pressed():
        radio.send("PRIO")

    # Affichage Rouge / Clignotant
    if feuRouge :
        afficheRouge()
    else :
        afficheOrange()

    # Compteur temps à 0 toute les secondes
    if ticks_ms()-clignotant > 1000:
        clignotant = ticks_ms()
```

> Version finale sans LED : ne pas bloquer la boucle avec sleep()

Méthode : Code feu Maître

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

DUREE = 10  # 10 secondes
attente = DUREE
feuRouge = True
lastTransm = ticks_ms()  # derniere transmission ALIVE de l'esclave
clignotant = ticks_ms()

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    # orange clignotant
    couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,couleur)
def afficheAttente(attente):
    display.set_pixel(2,3,0)
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    # Arrivee du vehicule prioritaire au feu maitre
    if button_a.was_pressed():
        feuRouge = False
        attente = DUREE
        radio.send("ROUGE")

    # Traitement des messages en provenance de l'esclave
    incoming = radio.receive()
    # Arrivee du vehicule priopritaire au feu esclave
    if incoming == "PRIO" :
        feuRouge = True
        attente = DUREE
        radio.send("VERT")

    # Robustesse du dispositif
    # L'esclave est en vie :)
```

```
    if incoming == "ALIVE" :
        lastTransm = ticks_ms()
    # Mise en securite si l'esclave plante
    if ticks_ms() - lastTransm > 2000:
        # Plus de deux secondes sans nouvelles de l'esclave
        feuRouge = True
        display.show(Image.SAD)
    else:
        # Affichage Rouge / Orange
        if feuRouge :
            afficheRouge()
        else :
            afficheOrange()

        # Action toutes les secondes
        if ticks_ms()-clignotant > 1000:
            clignotant = ticks_ms()
            attente -= 1
            afficheAttente(attente)
            radio.send(str(attente))

            # Alternance du feu
            if attente == 0 :
                attente=DUREE
                feuRouge = not feuRouge

            # Le maitre envoie l'etat du feu a l'esclave toutes les secondes
            if feuRouge :
                radio.send("VERT")
            else:
                radio.send("ROUGE")
```

Méthode : Code feu Esclave

```
from microbit import *
from time import ticks_ms
import radio

radio.config(group=2)
radio.on()

feuRouge = True
clignotant = ticks_ms()
lastTransm = ticks_ms()  # derniere transmission du maitre
lastAlive = ticks_ms()  # derniere transmission ALIVE de l'esclave

def afficheRouge():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,9)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,0)
def afficheOrange():
    for x in range(3,5):
        for y in range(2):
            display.set_pixel(x,y,0)
    # orange clignotant
```

```
    couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
    for x in range(3,5):
        for y in range(3,5):
            display.set_pixel(x,y,couleur)
def afficheAttente(attente):
    display.set_pixel(2,3,0)
    for x in range(2):
        for y in range(5):
            display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)

while True:
    incoming = radio.receive()
    if incoming:
        if incoming == "VERT" :
            feuRouge = False
            lastTransm = ticks_ms()
        elif incoming == "ROUGE" :
            feuRouge = True
            lastTransm = ticks_ms()
        elif len(incoming) ==1:
            afficheAttente(int(incoming))

    # Robustesse du dispositif :
    # L'esclave envoie un message au maitre toutes les secondes
    if ticks_ms() - lastAlive > 1000:
        radio.send("ALIVE")
        lastAlive = ticks_ms()
    # Mise en securite si le maitre plante
    if ticks_ms() - lastTransm > 2000 :
        # Plus de deux secondes sans nouvelles du maitre
        feuRouge = True
        display.show(Image.SAD)
    else:
        # Arrivee vehicule prioritaire
        if button_a.was_pressed():
            radio.send("PRIO")

        # Affichage Rouge / Clignotant
        if feuRouge :
            afficheRouge()
        else :
            afficheOrange()

        # Compteur temps à 0 toute les secondes
        if ticks_ms()-clignotant > 1000:
            clignotant = ticks_ms()
```

> scudFighter - Listing complet

Le programme complet

```
# The MIT License (MIT)
# Copyright (c) 2016 British Broadcasting Corporation.
# This software is provided by Lancaster University by arrangement with the
BBC.
# Permission is hereby granted, free of charge, to any person obtaining a
```

```
# copy of this software and associated documentation files (the "Software"),
# to deal in the Software without restriction, including without limitation
# the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following conditions:
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.

# Olivier Lecluse
# Avril 2019

from microbit import *
from time import ticks_us
import radio

radio.on()

SCUDFRIENDCOLOR=4
SCUDENMYCOLOR=6

speed=1
accel=0.2
score=0
position=2
scuds = []
lastLoop = ticks_us()

def position_move(p):
    global position
    display.set_pixel(position,4,0)
    position += p
    position = max(min(4,position),0)
    display.set_pixel(position,4,9)

def fire():
    scuds.append((position,3,-1))
    display.set_pixel(position,3,SCUDFRIENDCOLOR)

def process_fire():
    setpos=set()
    removeScuds=set()
    for i,s in enumerate(scuds):
        display.set_pixel(s[0],max(0,s[1]),0)
        s1 = (s[0],s[1]+s[2],s[2]) # nouvelle position
        s2 = (s1[0],s1[1],-s1[2])  # missile en collision eventuelle
        if s2 in setpos:
            # collision  :autodestruction des 2 scuds
            removeScuds.add(s1)
            removeScuds.add(s2)
            removeScuds.add(s)
```

```
62              removeScuds.add((s[0],s[1],-s[2]))
63          else :
64              if 0<=s1[1]<4:
65                  setpos.add(s1)
66                  # deplacement du scud
67                  display.set_pixel(s1[0],s1[1],SCUDFRIENDCOLOR if s1[2]<0
  else SCUDENMYCOLOR)
68                  scuds[i]=s1
69              elif s1[1]==4:
70                  removeScuds.add(s)
71                  if s1[0] == position :
72                      perdu()
73              else :
74                  removeScuds.add(s)
75                  # envoi radio
76                  radio.send(str(s[0]))
77
78      # Nettoyage de la liste des scuds
79      for sr in removeScuds:
80          display.set_pixel(sr[0],max(0,sr[1]),0)
81          try:
82              scuds.remove(sr)
83          except:
84              pass
85
86  def refresh():
87      # On redessine l'affichage
88      display.clear()
89      display.set_pixel(position,4,9)
90      for s in scuds:
91          display.set_pixel(s[0],max(0,s[1]),SCUDFRIENDCOLOR if s[2]<0 else
  SCUDENMYCOLOR)
92
93  def perdu():
94      global speed
95      speed += accel # Le jeu s'accelere !
96
97      # envoi radio "HIT"
98      radio.send("HIT")
99
100     # animation
101     display.clear()
102     anim = [
103     Image("99999:90009:90009:90009:99999"),
104     Image("00000:06660:06060:06660:00000"),
105     Image("00000:00000:00300:00000:00000"),
106     Image("00000:06660:06060:06660:00000"),
107     Image("99999:90009:90009:90009:99999")]
108     display.show(anim, delay=100, loop=False, wait=True)
109     refresh()
110
111 def gagne():
112     global speed,score
113     speed += accel # Le jeu s'accelere !
114     score += 1
115     # animation
116     display.clear()
117     display.scroll(score)
118     sleep(500)
```

```
    refresh()

def process_button():
    # Gesion des boutons
    ba = button_a.was_pressed()
    bb = button_b.was_pressed()
    if ba and bb :
        fire()
    else:
        if ba:
            position_move(-1)
        if bb:
            position_move(1)

def process_radio():
    global lastLoop
    # gestion des messages radio
    incoming = radio.receive()
    if incoming:
        if incoming == "HIT":
            gagne()
        elif len(incoming)==1:
            # arrivee d'un missile
            lastLoop=0
            scuds.append((int(incoming),-1,1))
            display.set_pixel(int(incoming),0,SCUDENMYCOLOR)

display.set_pixel(position,4,9)
while True:
    delay = 1000000/speed

    process_button()
    process_radio()

    # Gestion du timing
    loop = ticks_us()
    if(loop-lastLoop < delay):
        # attente 200ms pour la detection tir
        sleep(min((loop-lastLoop)//1000,200))
    else :
        lastLoop = loop
        process_fire()

```