

# MicroPython sur carte micro:bit

*Rendre les objets intelligents grâce à Python*



Olivier Lécluse

# Table des matières



<b>Introduction</b>	3
<b>I - Activités de prise en main de la micro:bit</b>	4
1. Activité 1 .....	4
2. Activité 2 : jouer avec les pixels .....	4
3. Activité 3 : jouer avec les boutons .....	6
4. Activité 4 : Afficher des images .....	8
<b>II - Mini-projets sur Microbit</b>	10
1. Réaliser un compteur - niveau facile .....	10
2. Jeu de Pierre Feuille Ciseaux .....	12
3. Simulation d'un feu de chantier pour la circulation automobile .....	15
3.1. Version 0 .....	15
3.2. Version 1 : Passage à 2 feux .....	17
3.3. Version 2 : Le véhicule prioritaire .....	19
3.4. Version 3 : Améliorer la robustesse du dispositif .....	21
3.5. Version finale .....	22
3.6. Pour aller plus loin .....	22
4. le jeu de bataille navale - Niveau intermédiaire .....	22
5. Jeu de tir : scudFighter - Niveau intermédiaire .....	26
5.1. Les variables globales utilisées .....	26
5.2. Fonctions simples .....	27
5.3. Traitement des événements .....	28
5.4. La boucle principale .....	30
<b>Contenus annexes</b>	31

# Introduction



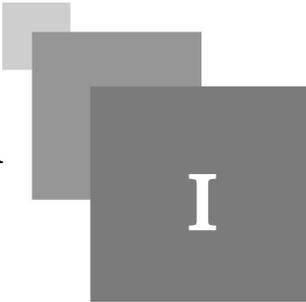
La carte micro:bit est un ordinateur à carte unique à processeur ARM conçu par la BBC pour l'éducation informatique au Royaume-Uni. En 2015, ce projet prévoit d'offrir gratuitement un exemplaire à chaque écolier britannique de douze ans. Plus récemment

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 est 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



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-il ?

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

Faites-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-il ?

```
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 gauche 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.

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

### Exercice 2 :

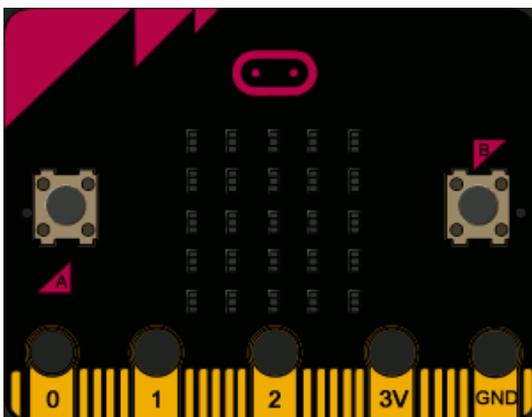
Modifiez ce programme afin qu'il allume la colonne centrale

### Exercice 3 :

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

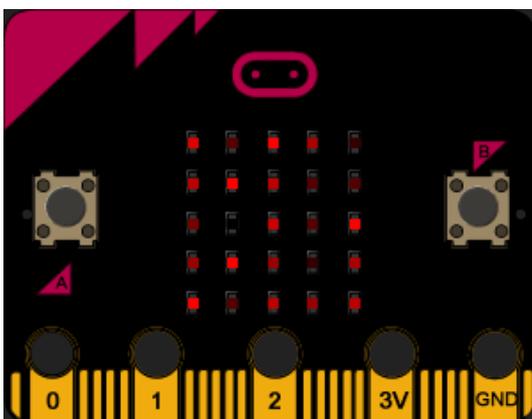
*Indication* : On pourra utiliser 2 boucles for imbriquées l'une dans l'autre. Pensez à changer le nom de la variable de la seconde boucle !

### 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 :

```
1 def carre(l, color):
2     for x in range(5-l,1):
3         for y in range(5-l,1):
4             display.set_pixel(x,y,color)
```

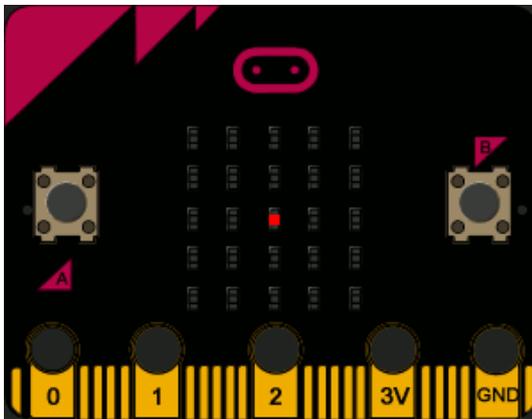
Dessinez sur votre feuille (1 carreau / pixel) l'écran de la carte lorsque l'on invoque

- `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 :

```
1 from microbit import *
2
3 compteur = 0
4 while True:
5     if button_a.was_pressed():
6         display.show(str(compteur))
7         sleep(1000)
8         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 :

```
1 from microbit import *
2
3 x = 0
4 y = 0
5
6 while True:
7     display.set_pixel(x,y,0)
8     if button_a.was_pressed():
9         x = x - 1
10    if button_b.was_pressed():
11        x = x + 1
12    x = max(0, min(x, 4))
13    display.set_pixel(x,y,9)
14    sleep(20)
```

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

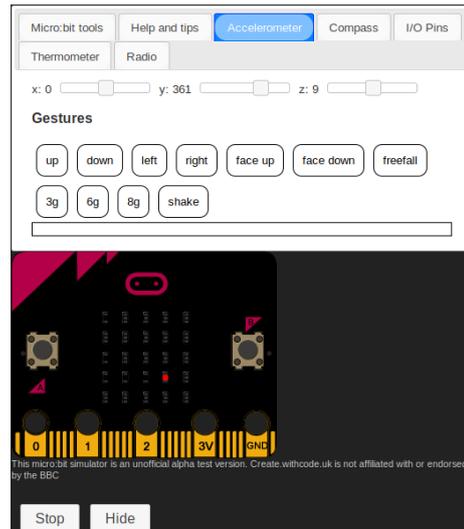
Expliquez le rôle de la ligne `x = max(0, min(x, 4))`

### 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.

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 votre 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

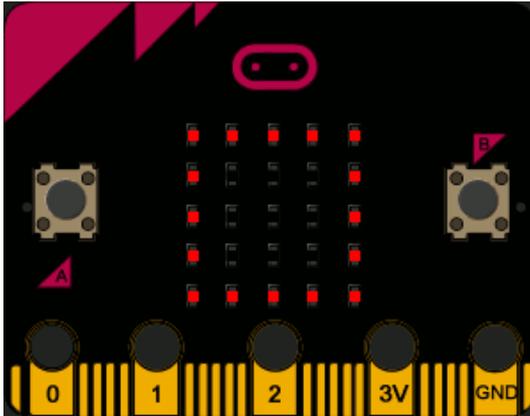
*Exercice 13 :*

Observez le code suivant. Il réalise une animation.

```
1 from microbit import *
2
3 imgs = [
4     Image('90000:00000:00000:00000:00000:'),
5     Image('90000:09000:00000:00000:00000:'),
6     Image('90000:09000:00900:00000:00000:'),
7     Image('90000:09000:00900:00090:00000:'),
8     Image('90000:09000:00900:00090:00009:')
9 ]
10 display.show(imgs, delay=500, loop=True)
11
```

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 :

```
1 from microbit import *
2 from random import choice
3
4 mesImages = [Image('00000:00000:00900:00000:00000:'),
5               Image('00009:00000:00000:00000:90000:'),
6               Image('00009:00000:00900:00000:90000:'),
7               Image('90009:00000:00000:00000:90009:'),
8               Image('90009:00000:00900:00000:90009:'),
9               Image('90009:00000:90009:00000:90009:')]
10 rolled = choice(mesImages)
11 display.show(rolled)
12
```

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 créez 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...

```

1 from microbit import *
2
3 c=0
4 while True:
5     if button_a.was_pressed():
6         c += 1
7
8     if button_b.was_pressed():
9         c=0
10
11     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 !

```

1 from microbit import *
2
3 c=0
4
5 def affiche(c):
6     k=0
7     for y in range(5):
8         for x in range(5):
9             k+=1
10            display.set_pixel(x, y, 9 if k<=c else 0)
11
12 while True:

```

```

13     if bouton_a.was_pressed():
14         c += 1
15
16     if bouton_b.was_pressed():
17         c=0
18
19     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...

```

1 from microbit import *
2
3 c=0
4
5 def affiche(c):
6     k=-1
7     for y in range(5):
8         for x in range(5):
9             k+=2
10            if k<c:
11                color=9
12            elif k==c:
13                color=4
14            else:
15                color=0
16            display.set_pixel(x, y, color)
17
18 while True:
19     if bouton_a.was_pressed():
20         c += 1
21
22     if bouton_b.was_pressed():
23         c=0
24
25     affiche(c)

```

### Méthode : Dernière version

Une LED est allumée ou éteinte, 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'introduire 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...

```

1 from microbit import *
2
3 c=0
4
5 def affiche(c):
6     x=0
7     y=0
8     binaire=bin(c)[2:]

```

```

9     nbChiffres = len(binaire)
10    display.clear()
11    for i in range(nbChiffres):
12        digit = binaire[nbChiffres-i-1]
13        display.set_pixel(x,y,int(digit)*9)
14        x+=1
15        if x==5:
16            x=0
17            y+=1
18
19    while True:
20        if button_a.was_pressed():
21            c += 1
22
23        if button_b.was_pressed():
24            c=0
25
26    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.

```

1 # Olivier Lecluse
2 # 8 mars 2019
3
4 from microbit import *
5 from random import randint
6
7 pierre = Image("00900:"
8               "09990:"
9               "99599:"
10              "09990:"
11              "00900")
12 feuille = Image("99900:"
13                 "90090:"
14                 "90009:"
15                 "90009:"
16                 "99999")
17 ciseaux = Image("96009:"
18                 "69090:"
19                 "00900:"
20                 "69090:"
21                 "96009")
22
23 nbSecousses = 0
24 while True:
25     if accelerometer.was_gesture("shake"):
26         nbSecousses +=1
27         display.show(str(nbSecousses))
28     if nbSecousses == 3:
29         choix = randint(1,3)

```

```

30     if choix == 1:
31         display.show(pierre)
32     elif choix == 2:
33         display.show(feuille)
34     elif choix == 3:
35         display.show(ciseaux)
36     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 boîtes noires !

Voici la version victime

```

1 # Olivier Lecluse
2 # 8 mars 2019
3 # Shifumi version victime
4
5 from microbit import *
6 from random import randint
7 import radio
8 radio.on()
9 radio.config(group=2)
10
11 pierre = Image("00900:"
12               "09990:"
13               "99599:"
14               "09990:"
15               "00900")
16 feuille = Image("99900:"
17                "90090:"
18                "90009:"
19                "90009:"
20                "99999")
21 ciseaux = Image("96009:"
22                "69090:"
23                "00900:"
24                "69090:"
25                "96009")
26
27 nbSecousses = 0
28 triche = 1
29 while True:
30     if accelerometer.was_gesture("shake"):
31         nbSecousses +=1
32         if nbSecousses == 1:
33             choix = randint(1,3)
34             radio.send(str(choix))
35             display.show(str(nbSecousses))
36         if nbSecousses == 3:
37             if choix == 1:
38                 display.show(pierre)
39             elif choix == 2:

```

```

40     display.show(feuille)
41     elif choix == 3:
42         display.show(ciseaux)
43     nbSecousses = 0

```

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

```

1 # Olivier Lecluse
2 # 8 mars 2019
3 # shifumi version tricheur
4
5 from microbit import *
6 from random import randint
7 import radio
8 radio.on()
9 radio.config(group=2)
10
11 pierre = Image("00900:"
12               "09990:"
13               "99599:"
14               "09990:"
15               "00900")
16 feuille = Image("99900:"
17                "90090:"
18                "90009:"
19                "90009:"
20                "99999")
21 ciseaux = Image("96009:"
22                "69090:"
23                "00900:"
24                "69090:"
25                "96009")
26
27 nbSecousses = 0
28 triche = 1
29 while True:
30     incoming = radio.receive()
31     if incoming:
32         triche = int(incoming)
33     if button_a.was_pressed() and button_b.was_pressed():
34         choix = triche%3 + 1
35     if accelerometer.was_gesture("shake"):
36         nbSecousses +=1
37         if nbSecousses == 1:
38             choix = randint(1,3)
39         display.show(str(nbSecousses))
40     if nbSecousses == 3:
41         if choix == 1:
42             display.show(pierre)
43         elif choix == 2:
44             display.show(feuille)
45         elif choix == 3:
46             display.show(ciseaux)
47     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*

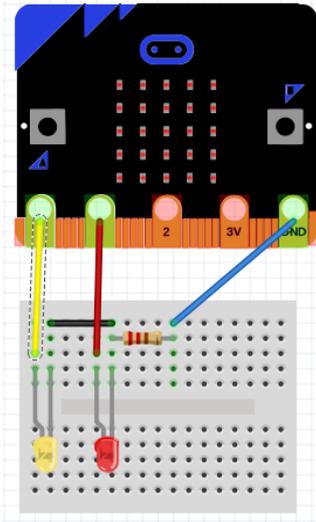
Créer une simulation de feu dans laquelle

- 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.



Pour cette activité, nous utiliserons

- 2 LEDs (une rouge et une orange)
- une résistance de protection de 220 Ohms
- 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

```
1 def afficheRouge():
2     pin1.write_digital(1)
3     pin0.write_digital(0)
4
5 def afficheOrange():
6     pin1.write_digital(0)
7     pin0.write_digital(1)
8
9 def afficheAttente(attente):
10    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

```
1 def afficheRouge():
2     for x in range(3,5):
3         for y in range(2):
4             display.set_pixel(x,y,9)
5     for x in range(3,5):
6         for y in range(3,5):
7             display.set_pixel(x,y,0)
8
9 def afficheOrange():
10    for x in range(3,5):
11        for y in range(2):
12            display.set_pixel(x,y,0)
13    for x in range(3,5):
14        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

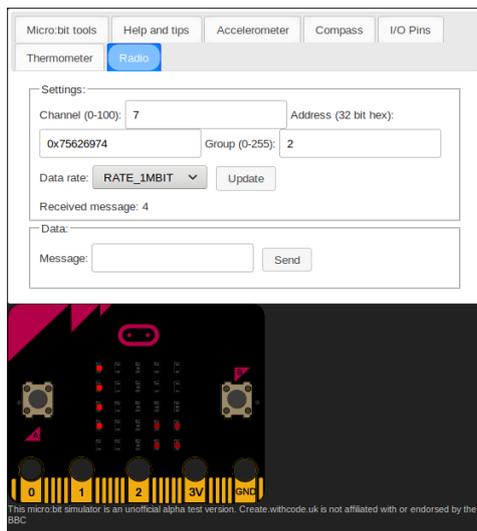
Nous allons travailler sur les programmes suivants : étudiez la version *avec LED* (cf. p.32) ou *sans LED* (cf. p. 33) selon votre configuration (la version sans LED fonctionne sur la carte seule en utilisant l'affichage de la matrice LED pour simuler les feux) puis répondez sur feuille aux questions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



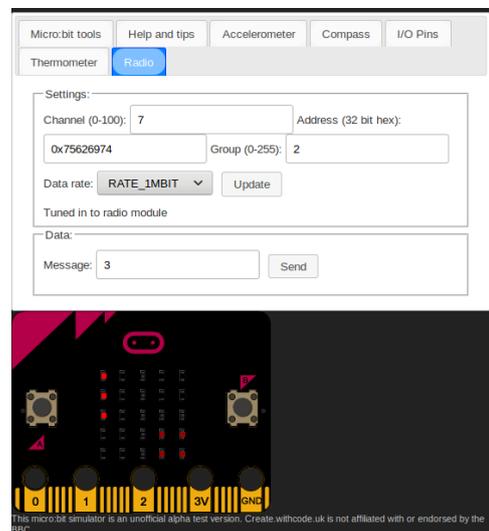
### Simulation : Utilisation du simulateur en ligne

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`.

```
1 # orange clignotant
2 couleur = max(0, 4 - abs(ticks_ms() - clignotant - 500) // 100)
3 for x in range(3, 5):
4     for y in range(3, 5):
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 aimant qui sera détecté 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 manière, 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 :

```
1 # Arrivee du vehicule prioritaire au feu maitre
2     if button_a.was_pressed():
3         feuRouge = False
4         attente = DUREE
5         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.

```
1 # Traitement des messages en provenance de l'esclave
2     incoming = radio.receive()
3     # Arrivee du vehicule priopritaire au feu esclave
4     if incoming == "PRIO" :
5         feuRouge = True
6         attente = DUREE
7         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.

```
1 # Arrivee vehicule prioritaire
2     if button_a.was_pressed():
3         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 :

```
1 # Robustesse du dispositif :
2   # Le maître envoie l'état du feu à l'esclave
3   # toutes les secondes
4   if feuRouge :
5       radio.send("VERT")
6   else:
7       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.

```
1 # Traitement des messages du maitre
2   incoming = radio.receive()
3   if incoming:
4       if incoming == "VERT" :
5           feuRouge = False
6           lastTransm = ticks_ms() # message reçu
7       elif incoming == "ROUGE" :
8           feuRouge = True
9           lastTransm = ticks_ms() # message reçu
10          elif len(incoming) ==1:
11              afficheAttente(int(incoming))
12
13          # Mise en securite si le maitre plante
14          if ticks_ms() - lastTransm > 2000 :
15              # Plus de deux secondes sans nouvelles du maitre
16              feuRouge = True
17              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

---

```
1 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.

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

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

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

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

```
1 # L'esclave envoie un message au maitre toutes les secondes
2 if ticks_ms() - lastAlive > 1000:
3     radio.send("ALIVE")
4     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.

```

1 # Olivier Lecluse
2 # 8 mars 2019
3 # version ordi
4
5 from random import randint
6 from microbit import *
7 import radio,music
8
9 radio.config(group=1)
10 radio.on()
11
12 boats = []
13 gameOn = False
14
15 def placer():
16     global boats
17     boats=[]
18     h=randint(0,1)
19     if h==0: # bateau de 3 horizontal
20         x3=randint(0,2)
21         y3=randint(0,4)
22         boats.append((x3,y3))
23         boats.append((x3+1,y3))
24         boats.append((x3+2,y3))
25     else: # bateau de 3 vertical
26         x3=randint(0,4)
27         y3=randint(0,2)
28         boats.append((x3,y3))
29         boats.append((x3,y3+1))
30         boats.append((x3,y3+2))
31
32     cont = True
33     while cont :
34         if h==0: # bateau de 2 vertical
35             x2=randint(0,4)
36             y2=randint(0,3)
37             if (x2,y2) not in boats and (x2, y2+1) not in boats:
38                 boats.append((x2,y2))
39                 boats.append((x2,y2+1))
40                 cont=False
41         else: # bateau de 2 horizontal
42             x2=randint(0,3)
43             y2=randint(0,4)
44             if (x2,y2) not in boats and (x2+1, y2) not in boats:
45                 boats.append((x2,y2))

```

```

46         boats.append((x2+1,y2))
47         cont=False
48
49 def afficher():
50     display.clear()
51     for b in boats:
52         display.set_pixel(b[0],b[1],9)
53
54 placer()
55 afficher()
56 while True:
57     if button_a.was_pressed() and not gameOn:
58         placer()
59         afficher()
60     if button_b.was_pressed() and not gameOn:
61         gameOn = True
62         radio.send("GO")
63         music.play(music.POWER_UP)
64     if gameOn:
65         # la partie est en cours
66
67         # Gestion de l'arrivee d'un coup
68         incoming = radio.receive()
69         if incoming:
70             try:
71                 hx,hy = incoming.split(',')
72                 hit=(int(hx),int(hy))
73             except:
74                 continue
75             if (hit) in boats:
76                 boats.remove(hit)
77                 radio.send("HIT")
78                 music.pitch(1000, 500)
79                 color=0
80             else:
81                 radio.send("MISS")
82                 music.pitch(200, 500)
83                 color=2
84             display.set_pixel(hit[0], hit[1], color)
85
86         # Fin de la partie
87         if len(boats)==0:
88             radio.send("GAME OVER")
89             music.play(music.FUNERAL)
90             gameOn=False
91

```

## Méthode : Version Joueur

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

```

1 # Olivier Lecluse
2 # 8 mars 2019
3 # version joueur
4

```

```

5 from microbit import *
6 import radio
7
8 radio.config(group=1)
9 radio.on()
10
11 nbHits = 0
12 gameOn = False
13 movePixColor = 5
14 missPixColor = 2
15 hitPixColor = 9
16
17 def deplacement():
18     global x, y, lastX, lastY, lastPix
19     # On rétablit l'ancien pixel
20     display.set_pixel(lastX, lastY, lastPix)
21
22     lastPix = display.get_pixel(x,y)
23     lastX, lastY=x, y
24     display.set_pixel(x,y,movePixColor)
25
26
27 while True:
28     if not gameOn:
29         # en attente du placement des bateaux
30         incoming = radio.receive()
31         if incoming == "GO":
32             gameOn = True
33             display.show("Go")
34             sleep(500)
35             display.clear()
36             nbHits=0
37             x,y = 0,0 # position du tir
38             lastPix = 0
39             lastX, lastY=0,0
40             display.set_pixel(0,0,movePixColor)
41         else :
42             # jouer la partie
43             if button_b.was_pressed():
44                 x = (x+1)%5
45                 deplacement()
46             if button_a.was_pressed():
47                 y = (y+1)%5
48                 deplacement()
49             if accelerometer.was_gesture("shake"):
50                 message = "{},{}".format(x, y)
51                 radio.send(message)
52                 response = False
53                 while not response:
54                     incoming = radio.receive()
55                     if incoming == "HIT":
56                         display.set_pixel(x,y,hitPixColor)
57                         lastPix = hitPixColor
58                         response = True
59                         nbHits += 1
60                     elif incoming == "MISS":
61                         display.set_pixel(x,y,missPixColor)
62                         lastPix = missPixColor

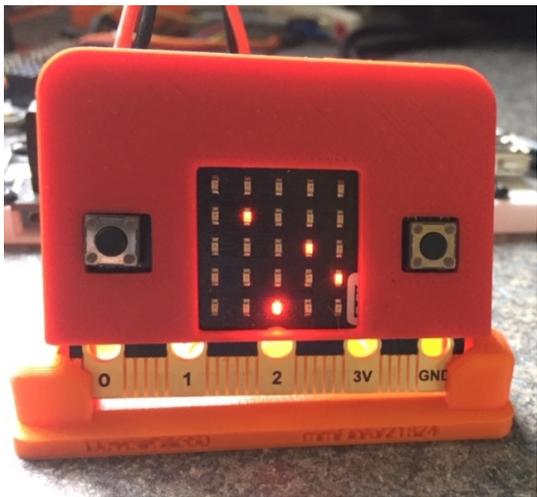
```

```

63         response = True
64         elif incoming == "GAME OVER":
65             display.set_pixel(x,y,hitPixColor)
66             lastPix = hitPixColor
67             display.show("WIN")
68             gameOn = False
69             response = True
70         elif incoming:
71             response = True
72     if nbHits == 5:
73         gameOn = False
74         display.show("WIN")
75

```

## 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

```

1 speed=1
2 accel=0.2
3 score=0
4 position=2
5 scuds = []
6 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.

```

1 def perdu():
2     global speed
3     speed += accel # Le jeu s'accelere !
4
5     # envoi radio "HIT"
6     radio.send("HIT")
7
8     # animation
9     display.clear()
10    anim = [
11        Image("99999:90009:90009:90009:99999"),
12        Image("00000:06660:06060:06660:00000"),
13        Image("00000:00000:00300:00000:00000"),
14        Image("00000:06660:06060:06660:00000"),
15        Image("99999:90009:90009:90009:99999")]
16    display.show(anim, delay=100, loop=False, wait=True)
17    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.

```

1 def gagne():
2     global speed, score
3     speed += accel # Le jeu s'accelere !
4     score += 1
5     # animation
6     display.clear()
7     display.scroll(score)
8     sleep(500)
9     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 pressé. 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.

```

1 def process_button():
2     # Gestion des boutons
3     ba = button_a.was_pressed()
4     bb = button_b.was_pressed()
5     if ba and bb :
6         fire()
7     else:
8         if ba:
9             position_move(-1)
10        if bb:

```

### *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 collisions de ces derniers : en effet, si 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:

```

```

9         # collision : autodestruction des 2 scuds
10        removeScuds.add(s1)
11        removeScuds.add(s2)
12        removeScuds.add(s)
13        removeScuds.add((s[0],s[1],-s[2]))
14    else :
15        if 0<=s1[1]<4:
16            setpos.add(s1)
17            # déplacement du scud
18            display.set_pixel(s1[0],s1[1],SCUDFRIENDCOLOR if s1[2]<0 else
SCUDENMYCOLOR)
19            scuds[i]=s1
20            elif s1[1]==4:
21                removeScuds.add(s)
22                if s1[0] == position :
23                    perdu()
24            else :
25                removeScuds.add(s)
26                # envoi radio
27                radio.send(str(s[0]))
28
29    # Nettoyage de la liste des scuds
30    for sr in removeScuds:
31        display.set_pixel(sr[0],max(0,sr[1]),0)
32        try:
33            scuds.remove(sr)
34        except:
35            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 :

```

1 delay = 1000000/speed
2 ...
3 loop = ticks_us()
4 if(loop-lastLoop < delay):
5     # attente 200ms pour la detection tir
6     sleep(min((loop-lastLoop)//1000,200))
7 else :
8     lastLoop = loop
9     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

```
1 from microbit import *
2
3 DUREE = 10 # 10 secondes
4 attente = DUREE
5 feuRouge = True
6
7 def afficheRouge():
8     pin1.write_digital(1)
9     pin0.write_digital(0)
10 def afficheOrange():
11     pin1.write_digital(0)
12     pin0.write_digital(1)
13 def afficheAttente(attente):
14     display.show(str(attente))
15
16 while True:
17     # Affichage Rouge / Orange
18     if feuRouge :
19         afficheRouge()
20     else :
21         afficheOrange()
22
23     # Gestion de l'attente
24     sleep(1000)
25     attente -= 1
26     afficheAttente(attente)
27
28     # Alternance du feu
29     if attente == 0:
30         feuRouge = not feuRouge
31         attente = DUREE
```

## > Corrigé V0 sans LED

```
1 from microbit import *
2
3 DUREE = 10 # 10 secondes
4 attente = DUREE
5 feuRouge = True
6
7 def afficheRouge():
8     for x in range(3,5):
9         for y in range(2):
```

```

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

```

## > Version 1 avec LED

### Méthode : Première carte

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 feuRouge = True
8
9 def afficheRouge():
10     pin1.write_digital(1)
11     pin0.write_digital(0)
12 def afficheOrange():
13     pin1.write_digital(0)
14     pin0.write_digital(1)
15 def afficheAttente(attente):
16     display.show(str(attente))
17
18 while True:
19     # Traitement des messages

```

```

20     incoming = radio.receive()
21     if incoming:
22         if incoming == "VERT" :
23             feuRouge = False
24         elif incoming == "ROUGE" :
25             feuRouge = True
26         elif len(incoming) ==1:
27             afficheAttente(int(incoming))
28
29     # Affichage Rouge / Orange
30     if feuRouge :
31         afficheRouge()
32     else :
33         afficheOrange()

```

## Méthode : Seconde carte

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 DUREE = 10 # 10 secondes
8 attente = DUREE
9 feuRouge = True
10
11 def afficheRouge():
12     pin1.write_digital(1)
13     pin0.write_digital(0)
14 def afficheOrange():
15     pin1.write_digital(0)
16     pin0.write_digital(1)
17 def afficheAttente(attente):
18     display.show(str(attente))
19
20 while True:
21     # Affichage Rouge / Orange
22     if feuRouge :
23         afficheRouge()
24     else :
25         afficheOrange()
26
27     # Gestion de l'attente
28     sleep(1000)
29     attente -= 1
30     afficheAttente(attente)
31     radio.send(str(attente))
32
33     # Alternance du feu
34     if attente == 0 :
35         attente=DUREE
36         feuRouge = not feuRouge
37         if feuRouge :
38             radio.send("VERT")
39         else:
40             radio.send("ROUGE")

```

**> Version 1 sans LED****✂** *Méthode : Première carte*

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 feuRouge = True
8
9 def afficheRouge():
10     for x in range(3,5):
11         for y in range(2):
12             display.set_pixel(x,y,9)
13     for x in range(3,5):
14         for y in range(3,5):
15             display.set_pixel(x,y,0)
16 def afficheOrange():
17     for x in range(3,5):
18         for y in range(2):
19             display.set_pixel(x,y,0)
20     for x in range(3,5):
21         for y in range(3,5):
22             display.set_pixel(x,y,4)
23 def afficheAttente(attente):
24     for x in range(2):
25         for y in range(5):
26             display.set_pixel(x,y, 0 if 5*x+y >=attente else 9)
27
28 while True:
29     # Traitement des messages du maitre
30     incoming = radio.receive()
31     if incoming:
32         if incoming == "VERT" :
33             feuRouge = False
34         elif incoming == "ROUGE" :
35             feuRouge = True
36         elif len(incoming) ==1:
37             afficheAttente(int(incoming))
38
39     # Affichage Rouge / Orange
40     if feuRouge :
41         afficheRouge()
42     else :
43         afficheOrange()
44

```

**✂** *Méthode : Seconde carte*

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 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)

```

```

13 pin0.write_digital(0)
14
15 def afficheOrange():
16     pin1.write_digital(0)
17     # orange clignotant
18     if ticks_ms()-clignotant < 500:
19         pin0.write_digital(1)
20     else:
21         pin0.write_digital(0)
22
23 def afficheAttente(attente):
24     display.show(str(attente))
25
26 while True:
27     # Traitement des messages du maitre
28     incoming = radio.receive()
29     if incoming:
30         if incoming == "VERT" :
31             feuRouge = False
32         elif incoming == "ROUGE" :
33             feuRouge = True
34         elif len(incoming)==1:
35             afficheAttente(int(incoming))
36
37     # Affichage Rouge / Orange
38     if feuRouge :
39         afficheRouge()
40     else :
41         afficheOrange()
42
43     # Compteur temps à 0 toute les secondes
44     if ticks_ms()-clignotant > 1000:
45         clignotant = ticks_ms()

```

## > Version sans 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     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     # orange clignotant
25     couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
26     for x in range(3,5):
27         for y in range(3,5):
28             display.set_pixel(x,y,couleur)
29 def afficheAttente(attente):
30     for x in range(2):
31         for y in range(5):
32             display.set_pixel(x,y, 0 if 5*x+y >=attente else 9)
33
34
35 while True:
36     # Affichage Rouge / Orange
37     if feuRouge :
38         afficheRouge()
39     else :
40         afficheOrange()
41
42     # Action toutes les secondes
43     if ticks_ms()-clignotant > 1000:
44         clignotant = ticks_ms()
45         attente -= 1
46         afficheAttente(attente)
47         radio.send(str(attente))
48
49     # Alternance du feu
50     if attente == 0 :
51         attente=DUREE
52         feuRouge = not feuRouge
53         if feuRouge :
54             radio.send("VERT")
55         else:
56             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     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     # orange clignotant
23     couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
24     for x in range(3,5):
25         for y in range(3,5):
26             display.set_pixel(x,y,couleur)
27 def afficheAttente(attente):
28     for x in range(2):
29         for y in range(5):
30             display.set_pixel(x,y, 0 if 5*x+y >=attente else 9)
31
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         elif incoming == "ROUGE" :
40             feuRouge = True
41         elif len(incoming)==1:
42             afficheAttente(int(incoming))
43
44     # Affichage Rouge / Orange
45     if feuRouge :
46         afficheRouge()
47     else :
48         afficheOrange()
49
50     # Compteur temps à 0 toute les secondes
51     if ticks_ms()-clignotant > 1000:
52         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

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 DUREE = 10 # 10 secondes
8 attente = DUREE
9 feuRouge = True
10
11 def afficheRouge():
12     pin1.write_digital(1)
13     pin0.write_digital(0)
14 def afficheOrange():
15     pin1.write_digital(0)
16     pin0.write_digital(1)

```

```

17 def afficheAttente(attente):
18     display.show(str(attente))
19
20 while True:
21     # Arrivee du vehicule prioritaire au feu maitre
22     if button_a.was_pressed():
23         feuRouge = False
24         attente = DUREE
25         radio.send("ROUGE")
26
27     # Affichage Rouge / Orange
28     if feuRouge :
29         afficheRouge()
30     else :
31         afficheOrange()
32
33     # Gestion de l'attente
34     sleep(1000)
35     attente -= 1
36     afficheAttente(attente)
37     radio.send(str(attente))
38
39     # Alternance du feu
40     if attente == 0 :
41         attente=DUREE
42         feuRouge = not feuRouge
43         if feuRouge :
44             radio.send("VERT")
45         else:
46             radio.send("ROUGE")

```

## Méthode : Code feu Esclave

pas de modification par rapport à la version précédente

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 feuRouge = True
8
9 def afficheRouge():
10     pin1.write_digital(1)
11     pin0.write_digital(0)
12 def afficheOrange():
13     pin1.write_digital(0)
14     pin0.write_digital(1)
15 def afficheAttente(attente):
16     display.show(str(attente))
17
18 while True:
19     # Traitement des messages du maitre
20     incoming = radio.receive()
21     if incoming:
22         if incoming == "VERT" :
23             feuRouge = False

```

```

24     elif incoming == "ROUGE" :
25         feuRouge = True
26     elif len(incoming) ==1:
27         afficheAttente(int(incoming))
28
29     # Affichage Rouge / Orange
30     if feuRouge :
31         afficheRouge()
32     else :
33         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

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 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     # Arrivee du vehicule prioritaire au feu maitre
32     if button_a.was_pressed():
33         feuRouge = False
34         attente = DUREE
35         radio.send("ROUGE")
36
37     # Affichage Rouge / Orange
38     if feuRouge :
39         afficheRouge()
40     else :

```

```

41     afficheOrange()
42
43     # Gestion de l'attente
44     sleep(1000)
45     attente -= 1
46     afficheAttente(attente)
47     radio.send(str(attente))
48
49     # Alternance du feu
50     if attente == 0 :
51         attente=DUREE
52         feuRouge = not feuRouge
53         if feuRouge :
54             radio.send("VERT")
55         else:
56             radio.send("ROUGE")
57

```

## Méthode : Code feu Esclave

pas de modification par rapport à la version précédente

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 feuRouge = True
8
9 def afficheRouge():
10     for x in range(3,5):
11         for y in range(2):
12             display.set_pixel(x,y,9)
13     for x in range(3,5):
14         for y in range(3,5):
15             display.set_pixel(x,y,0)
16 def afficheOrange():
17     for x in range(3,5):
18         for y in range(2):
19             display.set_pixel(x,y,0)
20     for x in range(3,5):
21         for y in range(3,5):
22             display.set_pixel(x,y,4)
23 def afficheAttente(attente):
24     for x in range(2):
25         for y in range(5):
26             display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)
27
28 while True:
29     # Traitement des messages du maitre
30     incoming = radio.receive()
31     if incoming:
32         if incoming == "VERT" :
33             feuRouge = False
34         elif incoming == "ROUGE" :
35             feuRouge = True
36         elif len(incoming) ==1:

```

```

37         afficheAttente(int(incoming))
38
39     # Affichage Rouge / Orange
40     if feuRouge :
41         afficheRouge()
42     else :
43         afficheOrange()
44

```

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

### Méthode : Code feu Maître

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 DUREE = 10 # 10 secondes
8 attente = DUREE
9 feuRouge = True
10
11 def afficheRouge():
12     pin1.write_digital(1)
13     pin0.write_digital(0)
14 def afficheOrange():
15     pin1.write_digital(0)
16     pin0.write_digital(1)
17 def afficheAttente(attente):
18     display.show(str(attente))
19
20 while True:
21     # Arrivee du vehicule prioritaire au feu maitre
22     if button_a.was_pressed():
23         feuRouge = False
24         attente = DUREE
25         radio.send("ROUGE")
26
27     # Traitement des messages en provenance de l'esclave
28     incoming = radio.receive()
29     # Arrivee du vehicule priopritaire au feu esclave
30     if incoming == "PRIO" :
31         feuRouge = True
32         attente = DUREE
33         radio.send("VERT")
34
35     # Affichage Rouge / Orange
36     if feuRouge :
37         afficheRouge()
38     else :
39         afficheOrange()
40
41     # Gestion de l'attente
42     sleep(1000)
43     attente -= 1
44     afficheAttente(attente)

```

```

45     radio.send(str(attente))
46
47     # Alternance du feu
48     ifattente == 0 :
49         attente=DUREE
50         feuRouge = not feuRouge
51         if feuRouge :
52             radio.send("VERT")
53         else:
54             radio.send("ROUGE")

```

### Méthode : Code feu Esclave

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 feuRouge = True
8
9 def afficheRouge():
10     pin1.write_digital(1)
11     pin0.write_digital(0)
12 def afficheOrange():
13     pin1.write_digital(0)
14     pin0.write_digital(1)
15 def afficheAttente(attente):
16     display.show(str(attente))
17
18 while True:
19     # Traitement des messages du maitre
20     incoming = radio.receive()
21     if incoming:
22         if incoming == "VERT" :
23             feuRouge = False
24         elif incoming == "ROUGE" :
25             feuRouge = True
26         elif len(incoming) ==1:
27             afficheAttente(int(incoming))
28
29     # Arrivee vehicule prioritaire
30     if button_a.was_pressed():
31         radio.send("PRIO")
32
33     # Affichage Rouge / Orange
34     if feuRouge :
35         afficheRouge()
36     else :
37         afficheOrange()

```

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

### Méthode : Code feu Maître

```

1 from microbit import *

```

```

2 import radio
3
4 radio.config(group=2)
5 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     # Arrivee du vehicule prioritaire au feu maitre
32     if button_a.was_pressed():
33         feuRouge = False
34         attente = DUREE
35         radio.send("ROUGE")
36
37     # Traitement des messages en provenance de l'esclave
38     incoming = radio.receive()
39     # Arrivee du vehicule priopritaire au feu esclave
40     if incoming == "PRIO" :
41         feuRouge = True
42         attente = DUREE
43         radio.send("VERT")
44
45     # Affichage Rouge / Orange
46     if feuRouge :
47         afficheRouge()
48     else :
49         afficheOrange()
50
51     # Gestion de l'attente
52     sleep(1000)
53     attente -= 1
54     afficheAttente(attente)
55     radio.send(str(attente))
56
57     # Alternance du feu
58     if attente == 0 :
59         attente=DUREE

```

```

60     feuRouge = not feuRouge
61     if feuRouge :
62         radio.send("VERT")
63     else:
64         radio.send("ROUGE")
65

```

## Méthode : Code feu Esclave

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 feuRouge = True
8
9 def afficheRouge():
10     for x in range(3,5):
11         for y in range(2):
12             display.set_pixel(x,y,9)
13     for x in range(3,5):
14         for y in range(3,5):
15             display.set_pixel(x,y,0)
16 def afficheOrange():
17     for x in range(3,5):
18         for y in range(2):
19             display.set_pixel(x,y,0)
20     for x in range(3,5):
21         for y in range(3,5):
22             display.set_pixel(x,y,4)
23 def afficheAttente(attente):
24     for x in range(2):
25         for y in range(5):
26             display.set_pixel(x,y, 0 if 5*x+y >=attente else 9)
27
28 while True:
29     # Traitement des messages du maitre
30     incoming = radio.receive()
31     if incoming:
32         if incoming == "VERT" :
33             feuRouge = False
34         elif incoming == "ROUGE" :
35             feuRouge = True
36         elif len(incoming) ==1:
37             afficheAttente(int(incoming))
38
39     # Arrivee vehicule prioritaire
40     if button_a.was_pressed():
41         radio.send("PRIO")
42
43     # Affichage Rouge / Orange
44     if feuRouge :
45         afficheRouge()
46     else :
47         afficheOrange()
48

```

**> Corrigé situation A avec LED : Améliorer la robustesse du dispositif****X Méthode : Code feu Maître**

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 radio.on()
6
7 DUREE = 10 # 10 secondes
8 attente = DUREE
9 feuRouge = True
10
11 def afficheRouge():
12     pin1.write_digital(1)
13     pin0.write_digital(0)
14 def afficheOrange():
15     pin1.write_digital(0)
16     pin0.write_digital(1)
17 def afficheAttente(attente):
18     display.show(str(attente))
19
20 while True:
21     # Arrivee du vehicule prioritaire au feu maitre
22     if button_a.was_pressed():
23         feuRouge = False
24         attente = DUREE
25         radio.send("ROUGE")
26
27     # Traitement des messages en provenance de l'esclave
28     incoming = radio.receive()
29     # Arrivee du vehicule priopritaire au feu esclave
30     if incoming == "PRIO" :
31         feuRouge = True
32         attente = DUREE
33         radio.send("VERT")
34
35     # Affichage Rouge / Orange
36     if feuRouge :
37         afficheRouge()
38     else :
39         afficheOrange()
40
41     # Gestion de l'attente
42     sleep(1000)
43     attente -= 1
44     afficheAttente(attente)
45     radio.send(str(attente))
46
47     # Alternance du feu
48     if attente == 0 :
49         attente=DUREE
50         feuRouge = not feuRouge
51
52     # Robustesse du dispositif :
53     # Le maitre envoie l'etat du feu a l'esclave

```

```

54 # toutes les secondes
55 if feuRouge :
56     radio.send("VERT")
57 else:
58     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 feuRouge = True
10
11 def afficheRouge():
12     pin1.write_digital(1)
13     pin0.write_digital(0)
14 def afficheOrange():
15     pin1.write_digital(0)
16     pin0.write_digital(1)
17 def afficheAttente(attente):
18     display.show(str(attente))
19
20 while True:
21     # Traitement des messages du maitre
22     incoming = radio.receive()
23     if incoming:
24         if incoming == "VERT" :
25             feuRouge = False
26             lastTransm = ticks_ms() # message reçu
27         elif incoming == "ROUGE" :
28             feuRouge = True
29             lastTransm = ticks_ms() # message reçu
30         elif len(incoming) ==1:
31             afficheAttente(int(incoming))
32
33     # Mise en securite si le maitre plante
34     if ticks_ms() - lastTransm > 2000 :
35         # Plus de deux secondes sans nouvelles du maitre
36         feuRouge = True
37         display.show(Image.SAD)
38
39     # Arrivee vehicule prioritaire
40     if button_a.was_pressed():
41         radio.send("PRIO")
42
43     # Affichage Rouge / Orange
44     if feuRouge :
45         afficheRouge()
46     else :
47         afficheOrange()

```

**> Corrigé situation A sans LED : Améliorer la robustesse du dispositif****X Méthode : Code feu Maître**

```

1 from microbit import *
2 import radio
3
4 radio.config(group=2)
5 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     # Arrivee du vehicule prioritaire au feu maitre
32     if button_a.was_pressed():
33         feuRouge = False
34         attente = DUREE
35         radio.send("ROUGE")
36
37     # Traitement des messages en provenance de l'esclave
38     incoming = radio.receive()
39     # Arrivee du vehicule priopritaire au feu esclave
40     if incoming == "PRIO" :
41         feuRouge = True
42         attente = DUREE
43         radio.send("VERT")
44
45     # Affichage Rouge / Orange
46     if feuRouge :
47         afficheRouge()
48     else :
49         afficheOrange()
50
51     # Gestion de l'attente
52     sleep(1000)

```

```

53     attente -= 1
54     afficheAttente(attente)
55     radio.send(str(attente))
56
57     # Alternance du feu
58     if attente == 0 :
59         attente=DUREE
60         feuRouge = not feuRouge
61
62     # Robustesse du dispositif :
63     # Le maitre envoie l'etat du feu a l'esclave
64     # toutes les secondes
65     if feuRouge :
66         radio.send("VERT")
67     else:
68         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 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     display.set_pixel(2,3,0)
27     for x in range(2):
28         for y in range(5):
29             display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)
30
31 while True:
32     # Traitement des messages du maitre
33     incoming = radio.receive()
34     if incoming:
35         if incoming == "VERT" :
36             feuRouge = False
37             lastTransm = ticks_ms() # message recu
38         elif incoming == "ROUGE" :
39             feuRouge = True

```

```

40     lastTransm = ticks_ms() # message reçu
41     elif len(incoming) ==1:
42         afficheAttente(int(incoming))
43
44     # Mise en securite si le maitre plante
45     if ticks_ms() - lastTransm > 2000 :
46         # Plus de deux secondes sans nouvelles du maitre
47         feuRouge = True
48         display.clear()
49         display.show(Image.SAD)
50     else :
51         # Arrivee vehicule prioritaire
52         if button_a.was_pressed():
53             radio.send("PRIO")
54
55         # Affichage Rouge / Orange
56         if feuRouge :
57             afficheRouge()
58         else :
59             afficheOrange()

```

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

### 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 lastTransm = ticks_ms() # derniere transmission de l'esclave
11 feuRouge = True
12
13 def afficheRouge():
14     pin1.write_digital(1)
15     pin0.write_digital(0)
16 def afficheOrange():
17     pin1.write_digital(0)
18     pin0.write_digital(1)
19 def afficheAttente(attente):
20     display.show(str(attente))
21
22 while True:
23     # Arrivee du vehicule prioritaire au feu maitre
24     if button_a.was_pressed():
25         feuRouge = False
26         attente = DUREE
27         radio.send("ROUGE")
28
29     # Traitement des messages en provenance de l'esclave
30     incoming = radio.receive()
31     # Arrivee du vehicule priopritaire au feu esclave
32     if incoming == "PRIO" :

```

```

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

```

## 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     pin1.write_digital(1)
15     pin0.write_digital(0)
16 def afficheOrange():
17     pin1.write_digital(0)

```

```

18 pin0.write_digital(1)
19 def afficheAttente(attente):
20     display.show(str(attente))
21
22 while True:
23     # Traitement des messages du maitre
24     incoming = radio.receive()
25     if incoming:
26         if incoming == "VERT" :
27             feuRouge = False
28             lastTransm = ticks_ms() # message reçu
29         elif incoming == "ROUGE" :
30             feuRouge = True
31             lastTransm = ticks_ms() # message reçu
32         elif len(incoming) ==1:
33             afficheAttente(int(incoming))
34
35     # Robustesse du dispositif :
36     # Mise en securite si le maitre plante
37     if ticks_ms() - lastTransm > 2000 :
38         # Plus de deux secondes sans nouvelles du maitre
39         feuRouge = True
40         display.show(Image.SAD)
41     # L'esclave envoie un message au maitre toutes les secondes
42     if ticks_ms() - lastAlive > 1000:
43         radio.send("ALIVE")
44         lastAlive = ticks_ms()
45
46     # Arrivee vehicule prioritaire
47     if button_a.was_pressed():
48         radio.send("PRIO")
49
50     # Affichage Rouge / Orange
51     if feuRouge :
52         afficheRouge()
53     else :
54         afficheOrange()

```

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

### 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 lastTransm = ticks_ms() # derniere transmission de l'esclave
11 feuRouge = True
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     # Arrivee du vehicule prioritaire au feu maitre
35     if button_a.was_pressed():
36         feuRouge = False
37         attente = DUREE
38         radio.send("ROUGE")
39
40     # Traitement des messages en provenance de l'esclave
41     incoming = radio.receive()
42     # Arrivee du vehicule priopritaire au feu esclave
43     if incoming == "PRIO" :
44         feuRouge = True
45         attente = DUREE
46         radio.send("VERT")
47     # L'esclave est en vie :)
48     if incoming == "ALIVE" :
49         lastTransm = ticks_ms()
50
51     # Mise en securite si l'esclave plante
52     if ticks_ms() - lastTransm > 2000:
53         display.clear()
54         # Plus de deux secondes sans nouvelles de l'esclave
55         feuRouge = True
56         display.show(Image.SAD)
57     else:
58         # Affichage Rouge / Orange
59         if feuRouge :
60             afficheRouge()
61         else :
62             afficheOrange()
63
64         # Gestion de l'attente
65         sleep(1000)
66         attente -= 1
67         afficheAttente(attente)
68         radio.send(str(attente))
69
70     # Alternance du feu
71     if attente == 0 :
72         attente=DUREE
73         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:

```

```

49     radio.send("ALIVE")
50     lastAlive = ticks_ms()
51
52     # Mise en securite si le maitre plante
53     if ticks_ms() - lastTransm > 2000 :
54         # Plus de deux secondes sans nouvelles du maitre
55         feuRouge = True
56         display.show(Image.SAD)
57     else:
58         # Affichage Rouge / Orange
59         if feuRouge :
60             afficheRouge()
61         else :
62             afficheOrange()
63
64     # Arrivee vehicule prioritaire
65     if button_a.was_pressed():
66         radio.send("PRIO")
67

```

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

### 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 lastTransm = ticks_ms() # derniere transmission ALIVE de l'esclave
12 clignotant = ticks_ms()
13
14 def afficheRouge():
15     pin1.write_digital(1)
16     pin0.write_digital(0)
17
18 def afficheOrange():
19     pin1.write_digital(0)
20     # orange clignotant
21     if ticks_ms()-clignotant < 500:
22         pin0.write_digital(1)
23     else:
24         pin0.write_digital(0)
25
26 def afficheAttente(attente):
27     display.show(str(attente))
28
29 while True:
30     # Arrivee du vehicule prioritaire au feu maitre
31     if button_a.was_pressed():
32         feuRouge = False
33         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()

```

```

10 lastTransm = ticks_ms() # derniere transmission du maitre
11 lastAlive = ticks_ms() # derniere transmission ALIVE de l'esclave
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     incoming = radio.receive()
30     if incoming:
31         if incoming == "VERT" :
32             feuRouge = False
33             lastTransm = ticks_ms()
34         elif incoming == "ROUGE" :
35             feuRouge = True
36             lastTransm = ticks_ms()
37         elif len(incoming) ==1:
38             display.show(incoming)
39
40     # Robustesse du dispositif :
41     # Mise en securite si le maitre plante
42     if ticks_ms() - lastTransm > 2000 :
43         # Plus de deux secondes sans nouvelles du maitre
44         feuRouge = True
45         display.show(Image.SAD)
46     # L'esclave envoie un message au maitre toutes les secondes
47     if ticks_ms() - lastAlive > 1000:
48         radio.send("ALIVE")
49         lastAlive = ticks_ms()
50
51     # Arrivee vehicule prioritaire
52     if button_a.was_pressed():
53         radio.send("PRIO")
54
55     # Affichage Rouge / Clignotant
56     if feuRouge :
57         afficheRouge()
58     else :
59         afficheOrange()
60
61     # Compteur temps à 0 toute les secondes
62     if ticks_ms()-clignotant > 1000:
63         clignotant = ticks_ms()

```

**> Version finale sans LED : ne pas bloquer la boucle avec sleep()****✂ 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 lastTransm = ticks_ms() # dernière transmission ALIVE de l'esclave
12 clignotant = ticks_ms()
13
14 def afficheRouge():
15     for x in range(3,5):
16         for y in range(2):
17             display.set_pixel(x,y,9)
18     for x in range(3,5):
19         for y in range(3,5):
20             display.set_pixel(x,y,0)
21 def afficheOrange():
22     for x in range(3,5):
23         for y in range(2):
24             display.set_pixel(x,y,0)
25     # orange clignotant
26     couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
27     for x in range(3,5):
28         for y in range(3,5):
29             display.set_pixel(x,y,couleur)
30 def afficheAttente(attente):
31     display.set_pixel(2,3,0)
32     for x in range(2):
33         for y in range(5):
34             display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)
35
36 while True:
37     # Arrivée du véhicule prioritaire au feu maître
38     if button_a.was_pressed():
39         feuRouge = False
40         attente = DUREE
41         radio.send("ROUGE")
42
43     # Traitement des messages en provenance de l'esclave
44     incoming = radio.receive()
45     # Arrivée du véhicule prioritaire au feu esclave
46     if incoming == "PRIO" :
47         feuRouge = True
48         attente = DUREE
49         radio.send("VERT")
50
51     # Robustesse du dispositif
52     # L'esclave est en vie :)

```

```

53  if incoming == "ALIVE" :
54      lastTransm = ticks_ms()
55      # Mise en securite si l'esclave plante
56      if ticks_ms() - lastTransm > 2000:
57          # Plus de deux secondes sans nouvelles de l'esclave
58          feuRouge = True
59          display.show(Image.SAD)
60      else:
61          # Affichage Rouge / Orange
62          if feuRouge :
63              afficheRouge()
64          else :
65              afficheOrange()
66
67          # Action toutes les secondes
68          if ticks_ms()-clignotant > 1000:
69              clignotant = ticks_ms()
70              attente -= 1
71              afficheAttente(attente)
72              radio.send(str(attente))
73
74              # Alternance du feu
75              if attente == 0 :
76                  attente=DUREE
77                  feuRouge = not feuRouge
78
79              # Le maitre envoie l'etat du feu a l'esclave toutes les secondes
80              if feuRouge :
81                  radio.send("VERT")
82              else:
83                  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 lastTransm = ticks_ms() # derniere transmission du maitre
11 lastAlive = ticks_ms() # derniere transmission ALIVE de l'esclave
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     # orange clignotant

```

```

25 couleur = max(0,4 - abs(ticks_ms()-clignotant - 500)//100)
26 for x in range(3,5):
27     for y in range(3,5):
28         display.set_pixel(x,y,couleur)
29 def afficheAttente(attente):
30     display.set_pixel(2,3,0)
31     for x in range(2):
32         for y in range(5):
33             display.set_pixel(x,y, 0 if 5*x+y >= attente else 9)
34
35 while True:
36     incoming = radio.receive()
37     if incoming:
38         if incoming == "VERT" :
39             feuRouge = False
40             lastTransm = ticks_ms()
41         elif incoming == "ROUGE" :
42             feuRouge = True
43             lastTransm = ticks_ms()
44         elif len(incoming) ==1:
45             afficheAttente(int(incoming))
46
47     # Robustesse du dispositif :
48     # L'esclave envoie un message au maitre toutes les secondes
49     if ticks_ms() - lastAlive > 1000:
50         radio.send("ALIVE")
51         lastAlive = ticks_ms()
52     # Mise en securite si le maitre plante
53     if ticks_ms() - lastTransm > 2000 :
54         # Plus de deux secondes sans nouvelles du maitre
55         feuRouge = True
56         display.show(Image.SAD)
57     else:
58         # Arrivee vehicule prioritaire
59         if button_a.was_pressed():
60             radio.send("PRIO")
61
62         # Affichage Rouge / Clignotant
63         if feuRouge :
64             afficheRouge()
65         else :
66             afficheOrange()
67
68         # Compteur temps à 0 toute les secondes
69         if ticks_ms()-clignotant > 1000:
70             clignotant = ticks_ms()

```

## > scudFighter - Listing complet

### *Le programme complet*

```

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

```

```

5 # copy of this software and associated documentation files (the "Software"),
6 # to deal in the Software without restriction, including without limitation
7 # the rights to use, copy, modify, merge, publish, distribute, sublicense,
8 # and/or sell copies of the Software, and to permit persons to whom the
9 # Software is furnished to do so, subject to the following conditions:
10 # The above copyright notice and this permission notice shall be included in
11 # all copies or substantial portions of the Software.
12 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR
13 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
14 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
15 # THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
    OTHER
16 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
17 # FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
18 # DEALINGS IN THE SOFTWARE.
19
20 # Olivier Lecluse
21 # Avril 2019
22
23 from microbit import *
24 from time import ticks_us
25 import radio
26
27 radio.on()
28
29 SCUDFRIENDCOLOR=4
30 SCUDEMYCOLOR=6
31
32 speed=1
33 accel=0.2
34 score=0
35 position=2
36 scuds = []
37 lastLoop = ticks_us()
38
39 def position_move(p) :
40     global position
41     display.set_pixel(position,4,0)
42     position += p
43     position = max(min(4,position),0)
44     display.set_pixel(position,4,9)
45
46 def fire() :
47     scuds.append((position,3,-1))
48     display.set_pixel(position,3,SCUDFRIENDCOLOR)
49
50 def process_fire() :
51     setpos=set()
52     removeScuds=set()
53     for i,s in enumerate(scuds):
54         display.set_pixel(s[0],max(0,s[1]),0)
55         s1 = (s[0],s[1]+s[2],s[2]) # nouvelle position
56         s2 = (s1[0],s1[1],-s1[2]) # missile en collision eventuelle
57         if s2 in setpos:
58             # collision : autodestruction des 2 scuds
59             removeScuds.add(s1)
60             removeScuds.add(s2)
61             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             # déplacement 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)

```

```
119     refresh()
120
121 def process_button():
122     # Gestion des boutons
123     ba = button_a.was_pressed()
124     bb = button_b.was_pressed()
125     if ba and bb :
126         fire()
127     else:
128         if ba:
129             position_move(-1)
130         if bb:
131             position_move(1)
132
133 def process_radio():
134     global lastLoop
135     # gestion des messages radio
136     incoming = radio.receive()
137     if incoming:
138         if incoming == "HIT":
139             gagne()
140         elif len(incoming)==1:
141             # arrivee d'un missile
142             lastLoop=0
143             scuds.append((int(incoming),-1,1))
144             display.set_pixel(int(incoming),0,SCUDENMYCOLOR)
145
146 display.set_pixel(position,4,9)
147 while True:
148     delay = 1000000/speed
149
150     process_button()
151     process_radio()
152
153     # Gestion du timing
154     loop = ticks_us()
155     if(loop-lastLoop < delay):
156         # attente 200ms pour la detection tir
157         sleep(min((loop-lastLoop)//1000,200))
158     else :
159         lastLoop = loop
160         process_fire()
161
```