

FEUX LUMINEUX POUR PIÉTONS.

Contexte.

La traversée d'une rue en toute sécurité nécessite des feux lumineux pour les piétons et les voitures.

Ta mission est de concevoir et de réaliser le contrôle d'un feu pour piétons (rouge ou vert) et d'un feu pour voitures (rouge, orange ouvert). Tu disposes des deux feux réalisés avec des LEDs, tu devras programmer la séquence d'allumage des feux. Une séquence démarre en appuyant sur le bouton A du Micro:bit.

Les feux utilisés.



Feu piétons.

Ils sont composés de deux LEDs (rouge et verte) contrôlées par les broches P2 et P1 du Micro:bit.



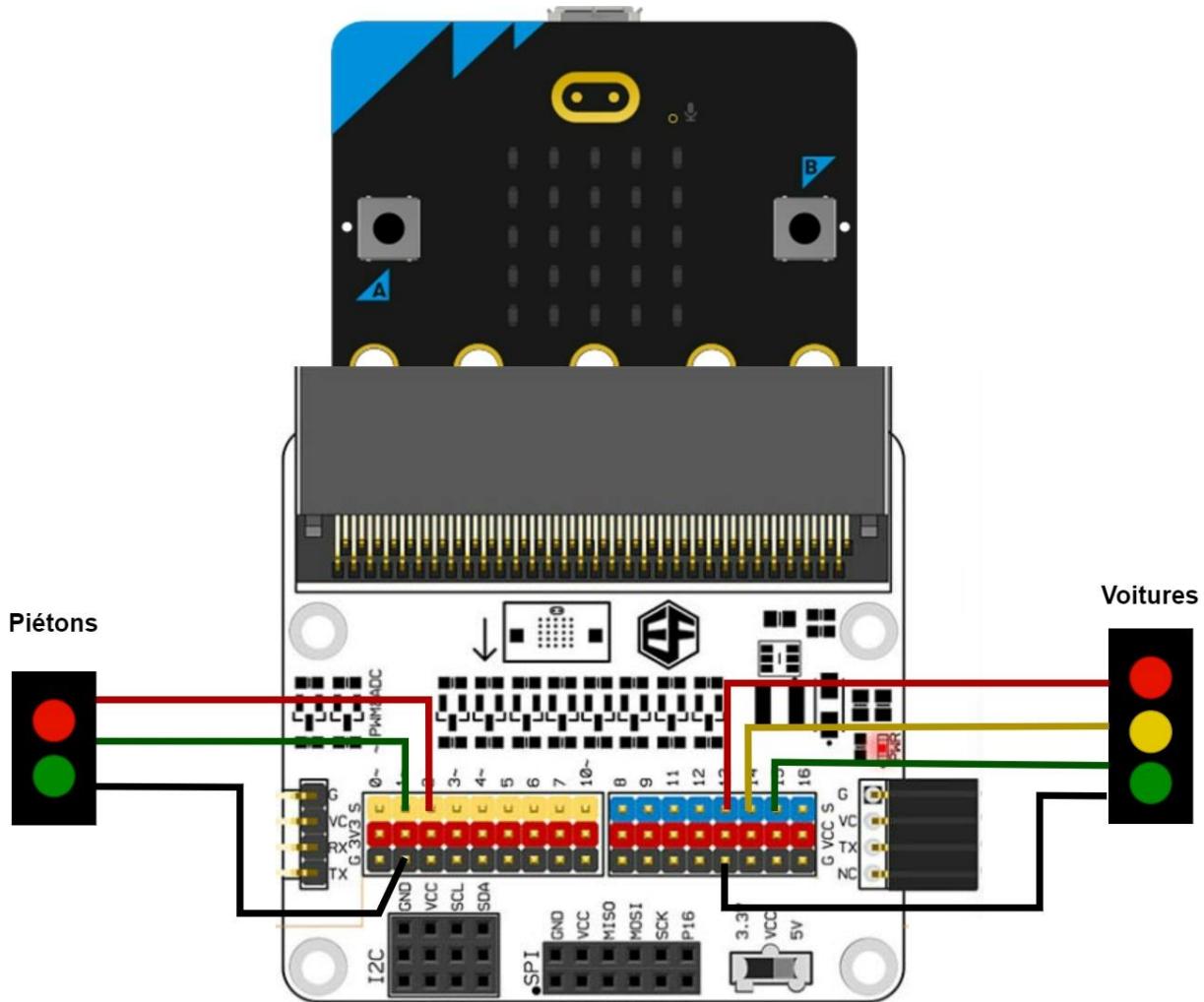
Feu voitures.

Ils sont composés de trois LEDs (rouge, jaune et verte) contrôlées par les broches P13, P14 et P15 du Micro:bit.
.

Les feux se connectent sur une carte d'extension **Octopus:bit** sur laquelle un **Micro:bit v2** doit être placé. Le Micro:bit est raccordé à ton PC par un câble USB.

Les LEDs sont des petites lampes comportant deux fils : un fil + et un fil -. Les fils - des LEDs de chaque signal sont raccordées sur un même fil noir. Les fils + des LEDs ont la même couleur que la LED à laquelle ils sont raccordés.

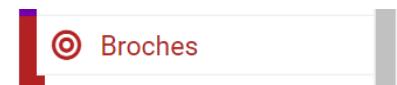
Câblage.



Programme.

Créer un nouveau projet et lui donner un nom, par exemple [Feux Piétons](#)

Pour contrôler les LEDs des feux, tu vas utiliser le bloc [écrire sur la broche](#) de la famille :



Il se trouve dans les familles de bloc [extensions](#).



Il y a deux choses à choisir dans le bloc : la broche (ici P0) et sa valeur (ici 0).

La broche, c'est la connexion de sortie sur laquelle tu as connecté un des fils. Par exemple P0 c'est la broche sur laquelle tu as connecté le fil rouge du signal pour piétons. En cliquant sur la petite flèche, tu obtiens une liste des broches disponibles.



La valeur peut être 0 et 1 :

- 0 éteint la LED.
- 1 allume la LED.



Il ne faut pas utiliser le bloc avec la valeur 1023 :



Ce bloc sert à fournir une valeur analogique, par exemple pour régler l'intensité d'une LED, tu l'utiliseras dans d'autres projets.

Sous-projet 1.

Pour ce sous-projet, tu vas faire clignoter la LED jaune du signal pour voiture : allumée une seconde, éteinte une seconde.

Pour cela tu vas utiliser un bloc **toujours** dans lequel tu vas allumer la LED, puis faire une pause d'une seconde, ensuite éteindre la LED et faire une nouvelle pause d'une seconde.



La valeur de la pause est en **ms**, ce qui signifie millisecondes, c'est-à-dire en millième de seconde. $1000 \text{ ms} = 1 \text{ seconde}$.

Pour tester (ta carte doit être connectée par le câble USB et les feux raccordés comme indiqué plus haut) tu cliques sur le bouton :



Et tu observes le résultat. Tu peux aussi avoir une idée de ce qui se passe sans télécharger en regardant les broches du Micro:bit représenté à gauche de ton code. La broche qui correspond à P14 clignote, lorsqu'elle est à 1 elle apparaît en rouge.

Petite variante : modifie ton programme pour allumer la LED deux secondes et puis l'éteindre une demi-seconde.

Sous-projet 2.

Pour ce sous-projet, il faudra prendre le contrôle de ton feu pour piétons. Pour cela, tu vas allumer le feu rouge (P2) 3 secondes puis le feu vert (P1) 5 secondes et recommencer la séquence indéfiniment.

Tu ne dois pas oublier, quand tu allumes une des LEDs, d'éteindre l'autre.



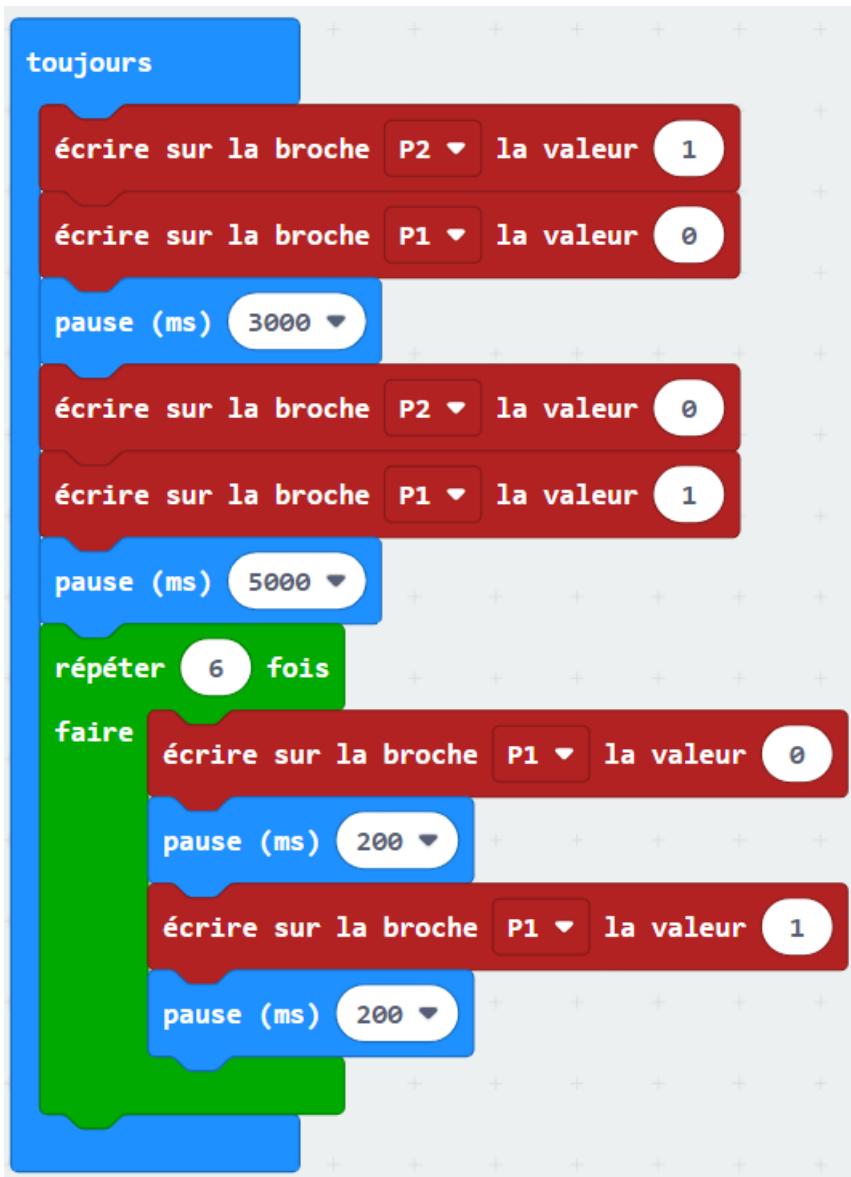
Observe ce qui se passe en simulation puis télécharge ton code sur ta carte.

Variante : ajoute à ton code des blocs pour faire clignoter la LED verte 6 fois après la période au vert. Le clignotement devra être rapide (pause de 200 ms).

Pour cela il faut utiliser un bloc répéter de la famille **Boucles** :



Tu remplaces **4** par **6** pour le nombre de répétitions et tu insères dans l'encoche **faire** le code pour le clignotement que tu as réalisé précédemment. Tu remplaces les temps de pause par 3 et 5 secondes.



Observe ce qui se passe en simulation puis télécharge ton code sur ta carte.

Sous-projet 3.

Pour ce sous-projet tu vas reprendre le fonctionnement précédent du signal pour piétons, mais plutôt que de le faire tourner indéfiniment, tu vas utiliser le bouton A de la carte Micro:bit pour commander le passage au vert. Après chaque passage au vert tu retourneras dans l'état rouge et attendras que l'on appuie sur le bouton A pour recommencer la séquence.

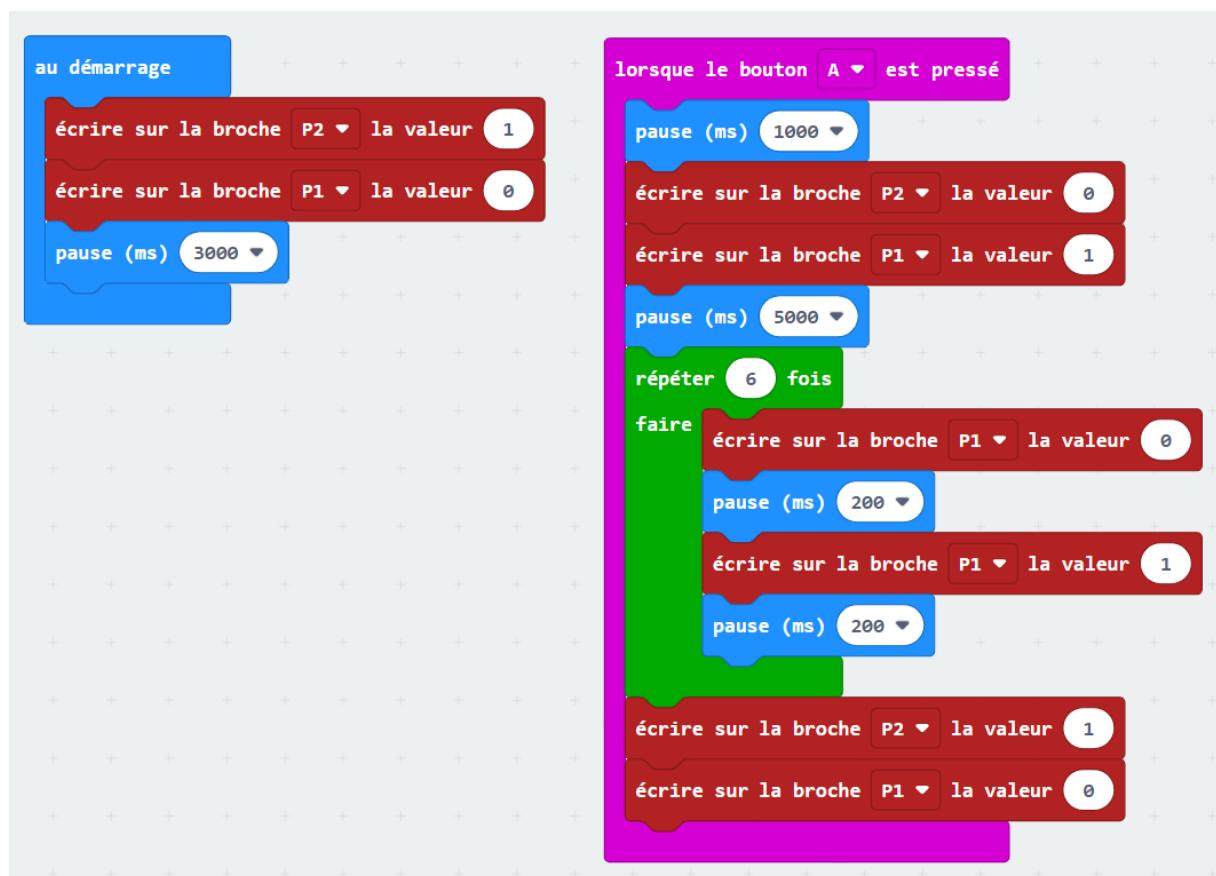
Pour cela il faudra un peu réorganiser le programme.

Remplacer le bloc **toujours** par un bloc :



de la famille Entrée.

Déplacer l'allumage du feu rouge vers la fin de ce bloc, ajouter un délai en début de bloc (pourquoi ?) et ajouter un bloc **au démarrage** pour mettre le feu au rouge lors du démarrage du programme.



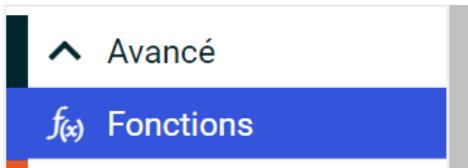
Observe ce qui se passe en simulation (pour simuler l'appui sur le bouton A, clique dessus avec la souris) puis télécharge ton code sur ta carte.



Micro:bit easy.

Dans le programme précédent, tu as dû écrire deux blocs identiques de deux instructions à deux endroits différents. Lorsque cette situation se reproduit souvent il y a une astuce que les informaticiens utilisent : ils créent une espèce de super instruction qui reprend le code dans ce qu'ils appellent une fonction.

Pour cela tu vas dans la famille Fonctions :

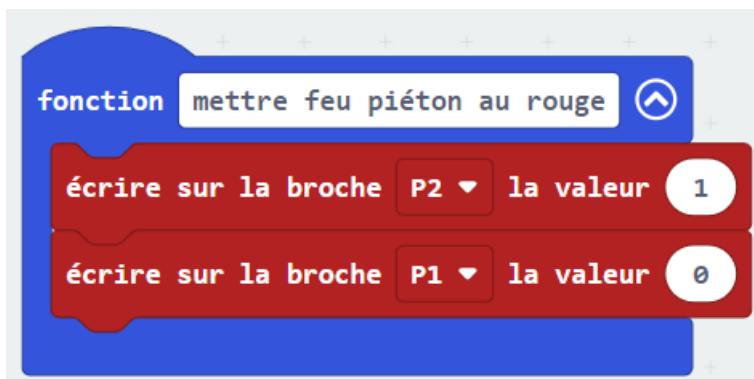


Puis tu cliques sur **Créer une fonction** puis sur terminer et tu obtiens le bloc suivant :



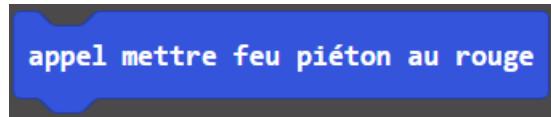
Dans l'encoche tu insères les blocs que tu souhaites regrouper en une fonction.

Puis tu changes le nom **FaireQuelqueChose** en un nom en rapport avec ce que la fonction fait, par exemple **mettre feu piéton au rouge**.



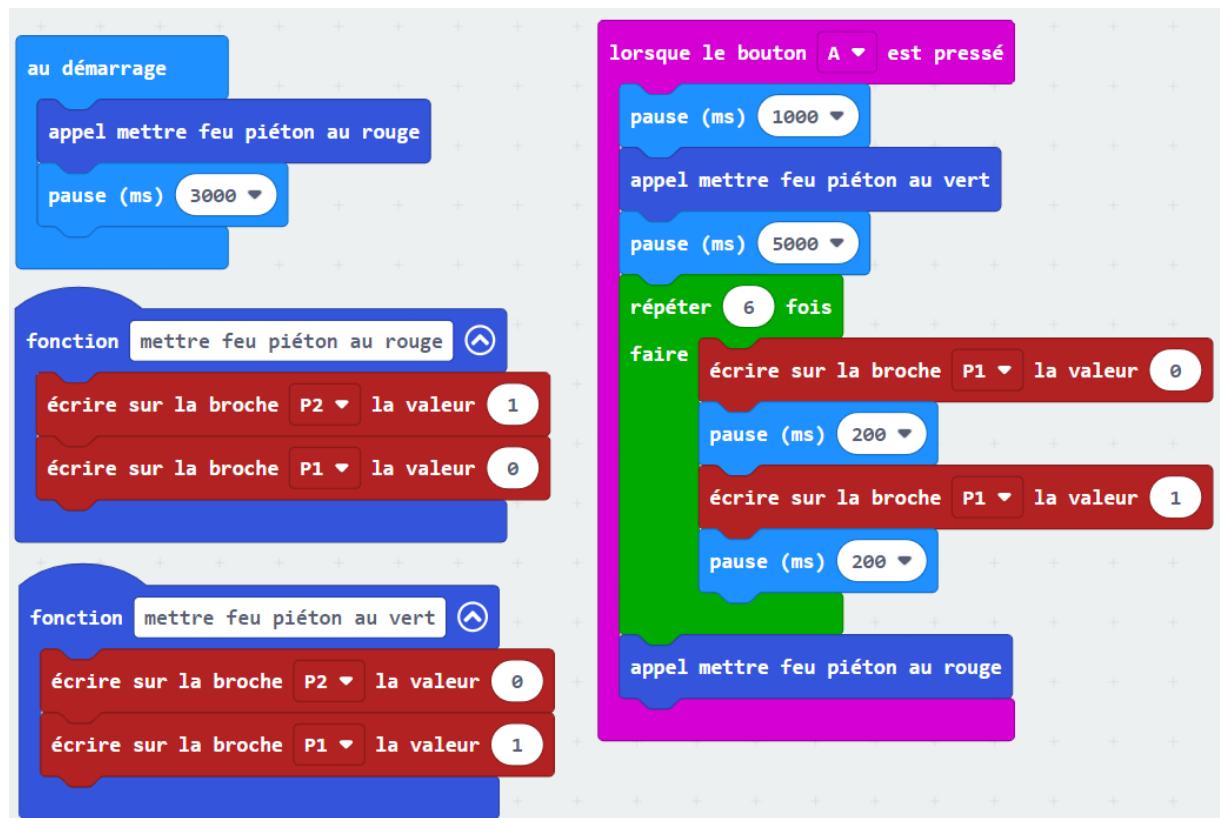
Ça y est, tu viens de créer une nouvelle super instruction !

Pour l'utiliser tu retournes dans la famille **Créer une fonction** et tu vois que tu dispose d'un nouveau bloc avec le mot **appel** devant le nom de ta super instruction.



Il te suffit maintenant de remplacer tes instructions par ce nouveau bloc.

Voici le programme modifié avec en plus une seconde fonction qui a été créée pour le vert.

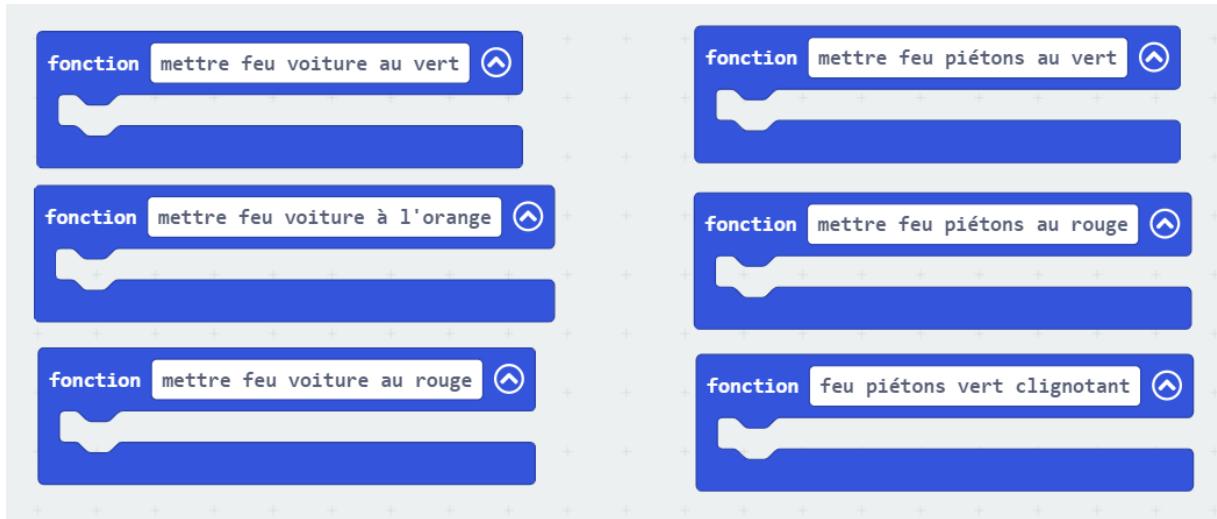


Bien sûr, dans cet exemple tu n'as pas gagné énormément, mais pour le programme final ce sera très utile et surtout cela te permet de créer tes propres blocs ce qui rend l'écriture du programme plus facile.

Projet final.

Tu vas maintenant ajouter le feu tricolore qui va arrêter le trafic des voitures. Il faut bien dire qu'un signal uniquement pour les piétons c'était un peu dangereux !

Pour cela, comme tu sais comment créer de nouveaux blocs grâce aux fonctions, tu vas commencer par créer 6 fonctions :



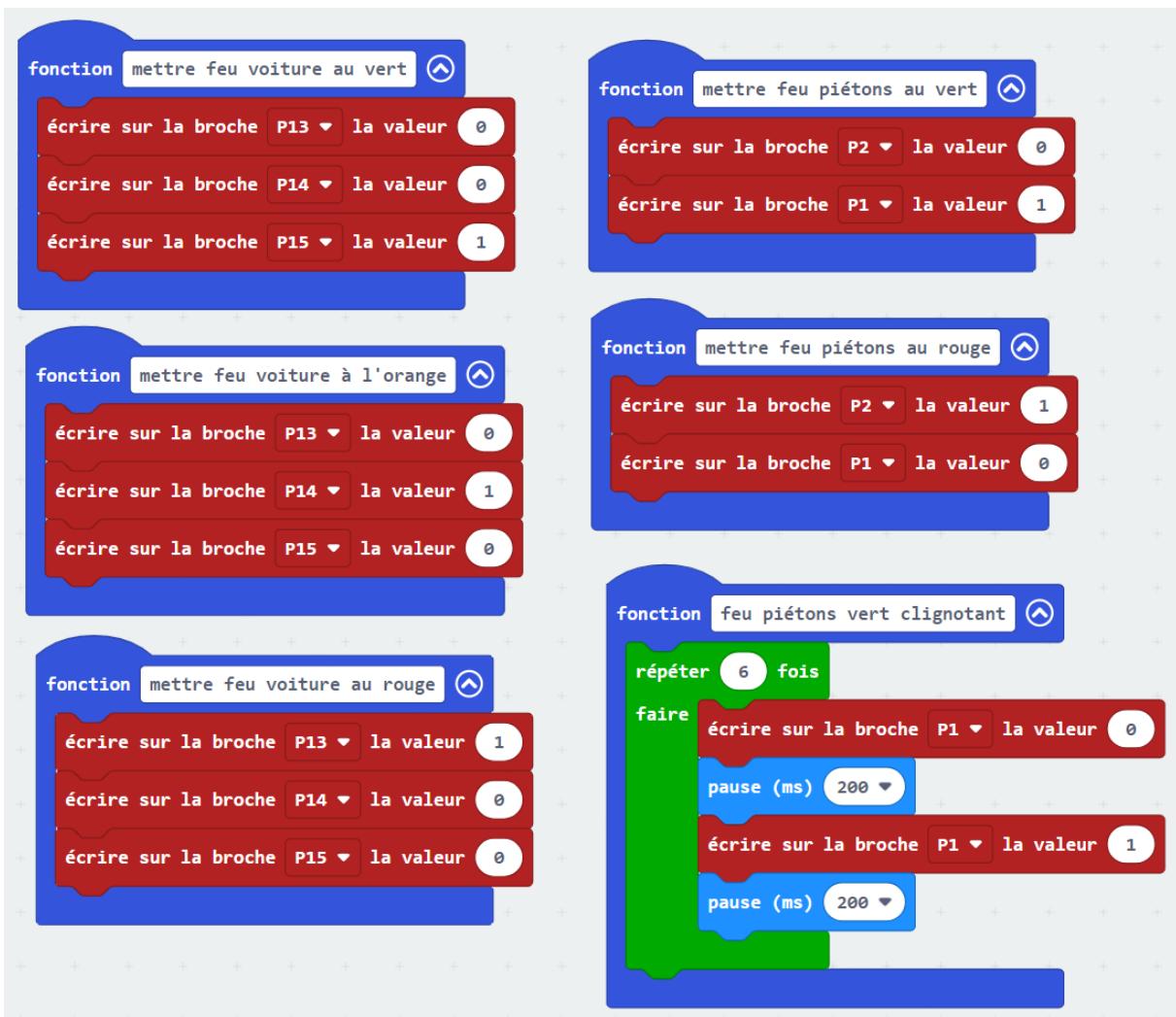
N'oublie pas que pour chaque changement de couleur il faut éteindre (= mettre à 0) les autres couleurs du feu.

Au démarrage il faudra mettre les deux signaux au rouge puis après une pause de une seconde mettre le feu voiture au vert.

Lorsque l'on appuiera sur le bouton A il faudra réaliser une séquence complète :

- Feu voitures à l'orange – pause.
- Feu voitures au rouge – pause.
- Feu piétons au vert – pause.
- Feu piétons vert clignotant 6 fois.
- Feu piétons au rouge – pause.
- Feu voitures au vert.

Les délais seront d'abord mis à des valeurs petites (une et deux secondes) puis dans la version finale, quand le programme sera au point on mettra des valeurs plus longues, car 5 secondes pour traverser c'est un peu court 😊.



```

when green flag is clicked [1]
  [call set pedestrian light to red v]
  [call set car light to red v]
end

when button A is pressed [1]
  [call set pedestrian light to red v]
  [call set car light to red v]
  [set pedestrian light to orange v]
  [set car light to orange v]
  [wait (1000 ms) v]
  [call set pedestrian light to green v]
  [call set car light to green v]
  [set pedestrian light to orange v]
  [set car light to orange v]
  [wait (5000 ms) v]
  [call set pedestrian light to red v]
  [call set car light to red v]
  [wait (1000 ms) v]
  [call set pedestrian light to green v]
  [call set car light to green v]
  [set pedestrian light to orange v]
  [set car light to orange v]
  [wait (2000 ms) v]

```

Voitures :
P13 : rouge
P14 : Orange
P15 : vert

Piétons :
P2 : rouge
P1 : vert

Bouton A : appel piétons

Observe ce qui se passe en simulation (pour simuler l'appui sur le bouton A, clique dessus avec la souris) puis télécharge ton code sur ta carte.

Pour aller plus loin.

Ajoute un mode dérangement qui indique que le signal ne fonctionne plus correctement. Dans ce mode toutes les LEDs sont éteintes sauf la LED orange du signal pour voiture qui clignote. Pour entrer dans ce mode, tu utiliseras le bouton B de la carte Micro:bit. Le code pour faire clignoter la LED sera introduit dans un bloc **tant que** avec la condition logique **vrai**.

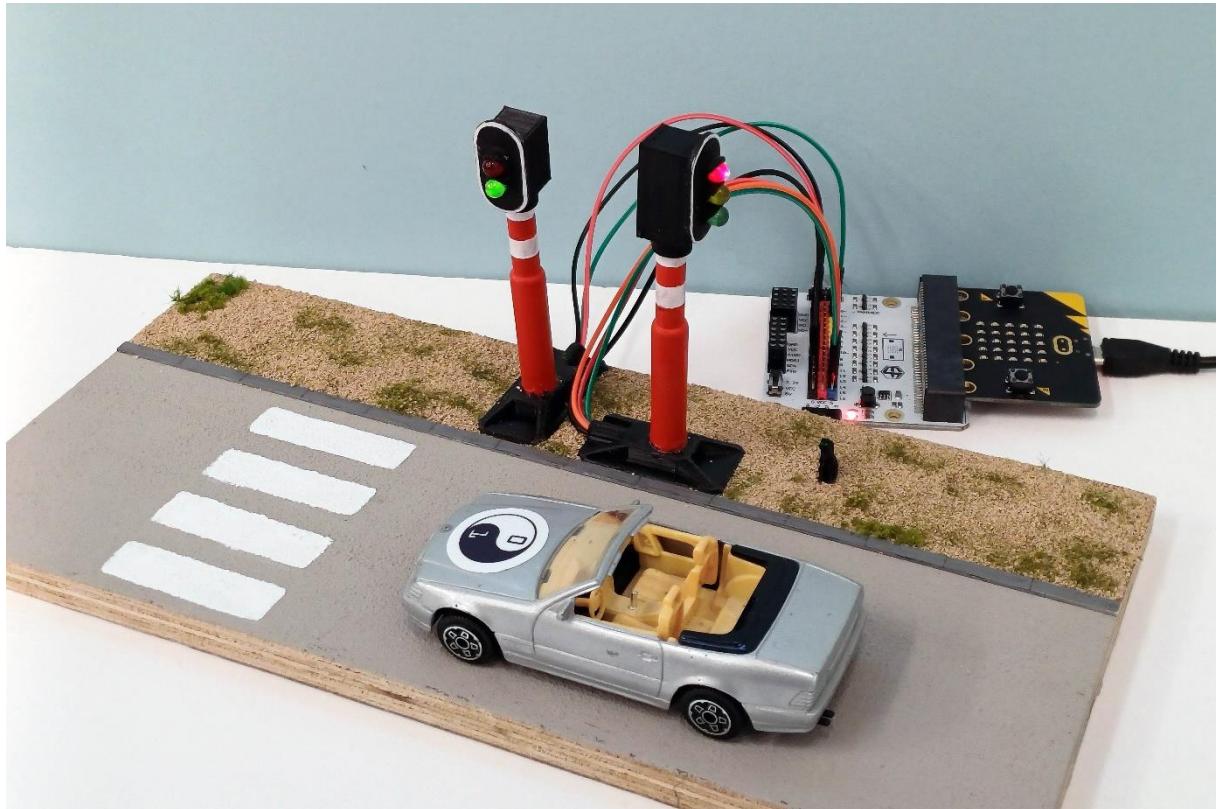


Ce bloc constitue une boucle infinie, ce qui veut dire qu'une fois démarrée elle ne se termine jamais...
Sauf si tu appuies sur le bouton reset de la carte.



Activités et variantes

L'application carrefour peut être l'occasion de pratiquer un peu de bricolage en réalisant un diorama avec une route et des trottoirs, de réaliser un peu d'électronique en construisant les signaux puis en câblant des LEDs...



Sur le même principe que le projet, il est possible de réaliser des feux pour un carrefour entre deux routes, avec détection de l'arrivée de voitures (capteur PIR ou ultrasons), des feux pour les trains miniatures...

Le programme peut être étayé en ajoutant un signal sonore pour les mal voyants, en faisant passer automatiquement les signaux dans la phase piétons si l'on a pas appuyé sur le bouton A endéans un certain temps.

Lexique anglais-français.

Pin : broche (aiguille en traduction littérale) d'où les symboles P0, P1... pour Pin 0, Pin 1...

DIY : (do it yourself) : bricolage.

To Tinker : bricoler, **tinkerer** : bricoleur, bricoleuse.

3D : en trois dimensions.

CAD : conception assistée par ordinateur (Computer Aided Design), par exemple <https://www.tinkercad.com/> est un logiciel simple pour la conception 3D.