

COFFRE A L'ANNEAU.

Contexte.



Gandalf doit cacher un anneau magique dans un coffre et souhaite que tu réalises un dispositif pour en contrôler l'accès.

Le principe est le suivant : celui qui désire accéder au coffre reçoit des indications sur le lieu approximatif où il se situe. Lorsqu'il approche du coffre, sa présence est détectée et une petite musique le guide vers le coffre. Ensuite c'est la magie qui va décider si le coffre s'ouvre ou non.

Ta mission est de concevoir et de réaliser un système utilisant un capteur de type sonar et un servo pour commander l'ouverture du coffre.



Pour contrôler l'ouverture du coffre un nombre compris entre 1 et 3 est tiré au sort et le visiteur tente sa chance en appuyant les boutons A (1), B (2) ou A+B (3) du Micro:bit. Si son choix correspond au nombre tiré au sort, le coffre s'ouvre grâce au servo.

Un afficheur OLED sera utilisé pour dialoguer avec le visiteur.

Projet préparatoire : mesurer une distance.

Pour ce sous-projet, tu vas faire utiliser le sonar et l'afficheur OLED pour mesurer la distance entre le capteur et un obstacle. La mesure se fera lorsque tu appuieras sur le bouton A du Micro:bit.

Matériel utilisé.

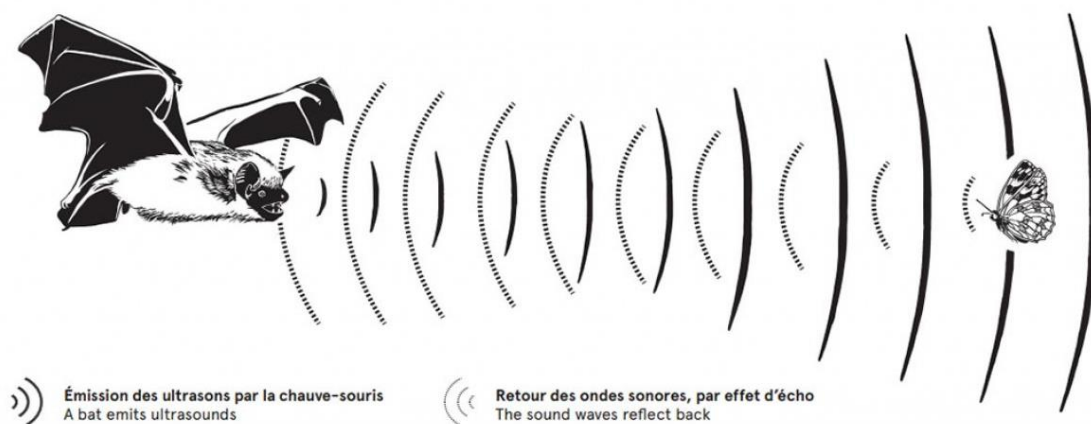
 <p>Un capteur de distance de type Sonar <i>Ce capteur utilise un émetteur et un récepteur d'ultrason.</i></p>	 <p>Un afficheur OLED. <i>Ce petit écran permet d'afficher du texte et des données. C'est le type d'écran qui se trouve dans les smartphones.</i></p>
---	---

Ces éléments se connectent sur une carte d'extension iot:bit (ou Octopus:bit) sur laquelle un Micro:bit v2 doit être placé. Un câble de connexion avec connecteurs à 3 fiches doit être utilisé pour le capteur de distance. Le Micro:bit est raccordé à ton PC par un câble USB.

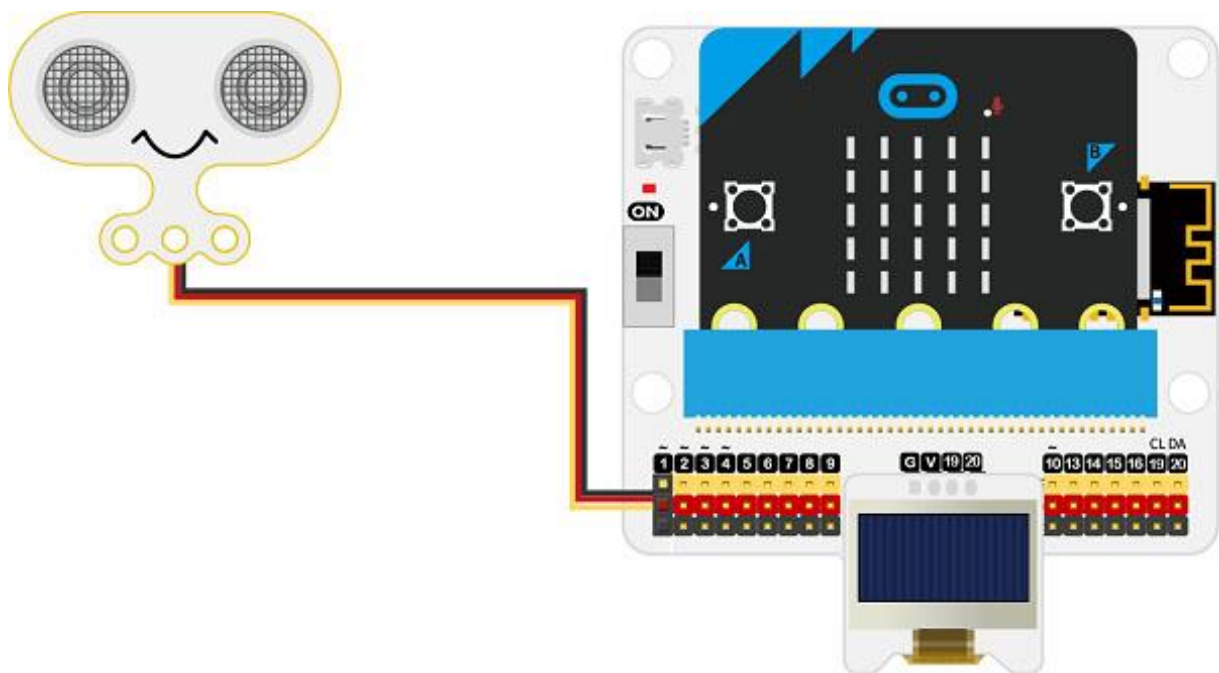
Sonar.

Dans un sonar, une onde ultrasonore est envoyée, lorsque qu'elle touche un obstacle elle revient vers l'émetteur qui la détecte. En mesurant le temps d'aller-retour de cette onde, il est possible de déterminer la distance.

Les chauves-souris utilisent ce principe pour se localier dans l'espace et trouver leur nourriture en vol.



Câblage.



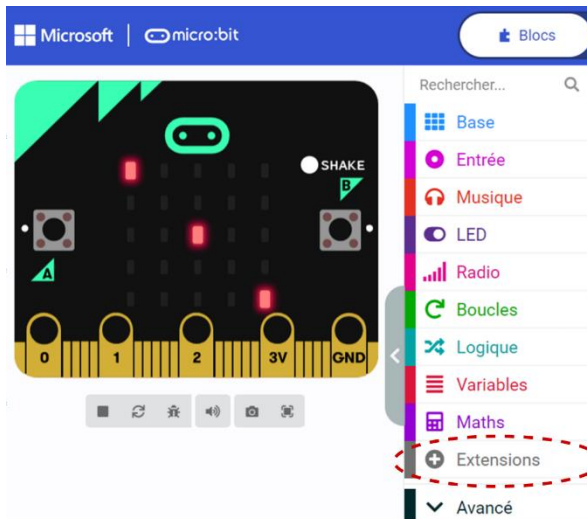
Programme.

Créer un nouveau projet et lui donner un nom, par exemple [Mesure de distance](#)

Le programme va utiliser des familles de blocs qui ne sont pas présentes par défaut, si cela n'a pas déjà été fait, il faut les installer.

Chargement de l'extension nécessaire au projet.

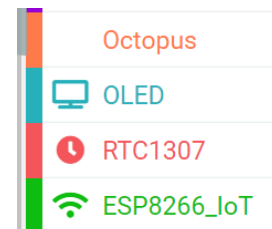
Cliquer sur Extensions dans la liste des blocs.



Puis faire une recherche avec le terme [iot-environment-kit](#) et choisir :



Après cette opération la liste des familles de blocs devrait comporter les nouveaux membres suivants :



Tu peux maintenant commencer le programme.

Il se composera d'un bloc au démarrage dans lequel nous allons initialiser l'afficheur OLED. Et d'un bloc lorsque le bouton A est pressé dans lequel nous allons insérer les blocs du programme qui seront exécutés chaque fois que tu appuieras sur le bouton A..

Tu vas également créer une variable que tu appelleras [distance](#) et insérer le bloc définir pour contrôler cette variable. La valeur sera la mesure de distance obtenue par le bloc ultrasonic distance in unit :



Ce bloc se trouve dans la famille [Octopus](#).

Tu choisiras cm pour obtenir une distance en centimètres et la broche P1 sur laquelle est connecté le sonar.

Ensuite tu afficheras un titre souligné, une ligne blanche et le contenu de la variable distance entourée de texte.

Toutes les fonctions d’affichage se trouvent dans la famille de bloc **OLED**.



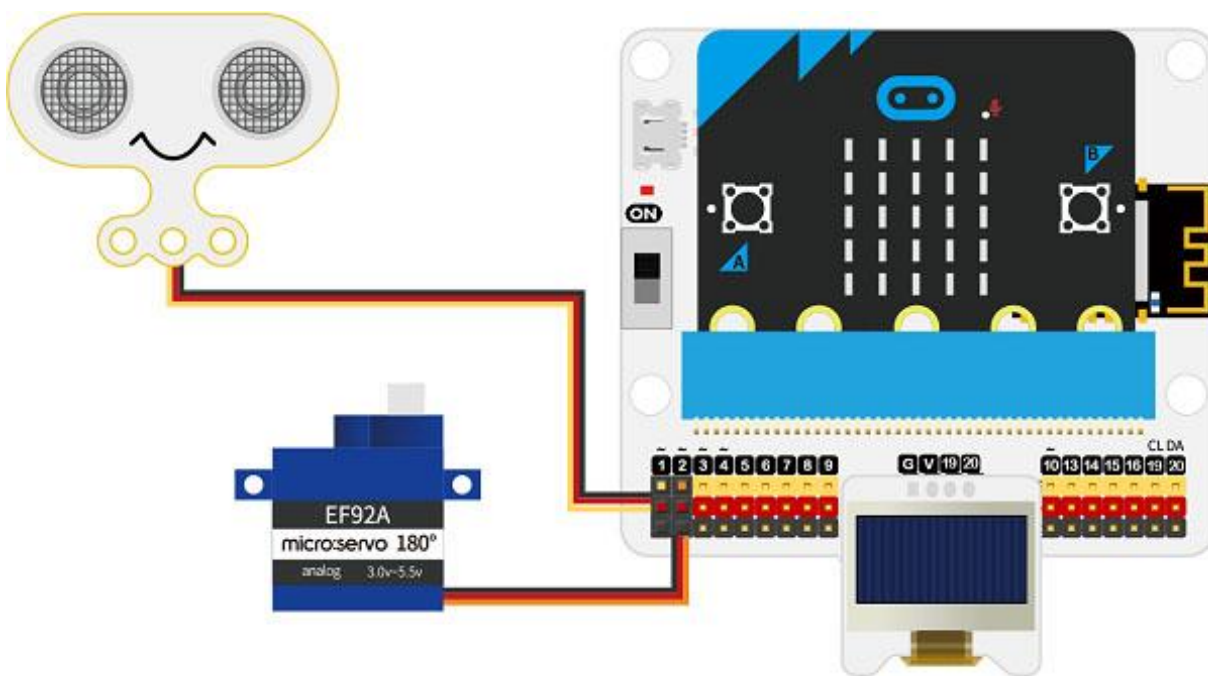
Pour tester (ta carte doit être connectée par le câble USB) tu cliques sur le bouton :



Et tu observes le résultat en déplaçant un obstacle devant le sonar et en appuyant sur le bouton A chaque fois que tu veux effectuer une mesure.

Projet complet : le contrôle du coffre à l'anneau.

 <p>Un capteur de distance de type Sonar <i>Ce capteur utilise un émetteur et un récepteur d'ultrason.</i></p>	 <p>Un servo 180°. <i>Moteur dont l'angle de rotation est contrôlé par une impulsion électrique dont la largeur détermine l'angle de rotation.</i></p>	 <p>Un afficheur OLED. <i>Ce petit écran permet d'afficher du texte et des données, il sera utilisé en phase de test.</i></p>
---	---	--



Ces éléments se connectent sur une carte d'extension iot:bit (ou Octopus:bit) sur laquelle un Micro:bit v2 doit être placé. Un câble de connexion avec connecteurs à 3 fiches doit être utilisé pour le capteur de distance. Le Micro:bit est raccordé à ton PC par un câble USB.

Le servo servira à ouvrir par une tringlerie ou tout autre dispositif le couvercle d'un coffret.

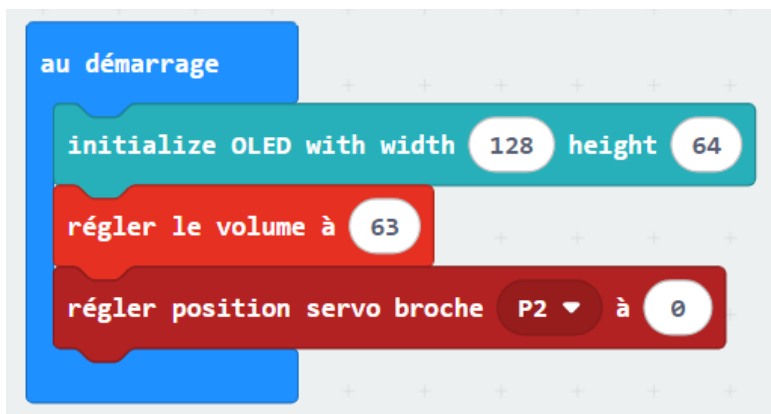
Programme.

Créer un nouveau projet et lui donner un nom, par exemple [contrôle du coffre](#)


Le programme va utiliser des familles de blocs qui ne sont pas présentes par défaut, si cela n'a pas déjà été fait, il faut les installer comme indiqué plus haut.

Le programme va d'abord se composer de deux blocs principaux :

au démarrage : qui va initialiser l'OLED, le servo en position 0 et régler le volume sonore.



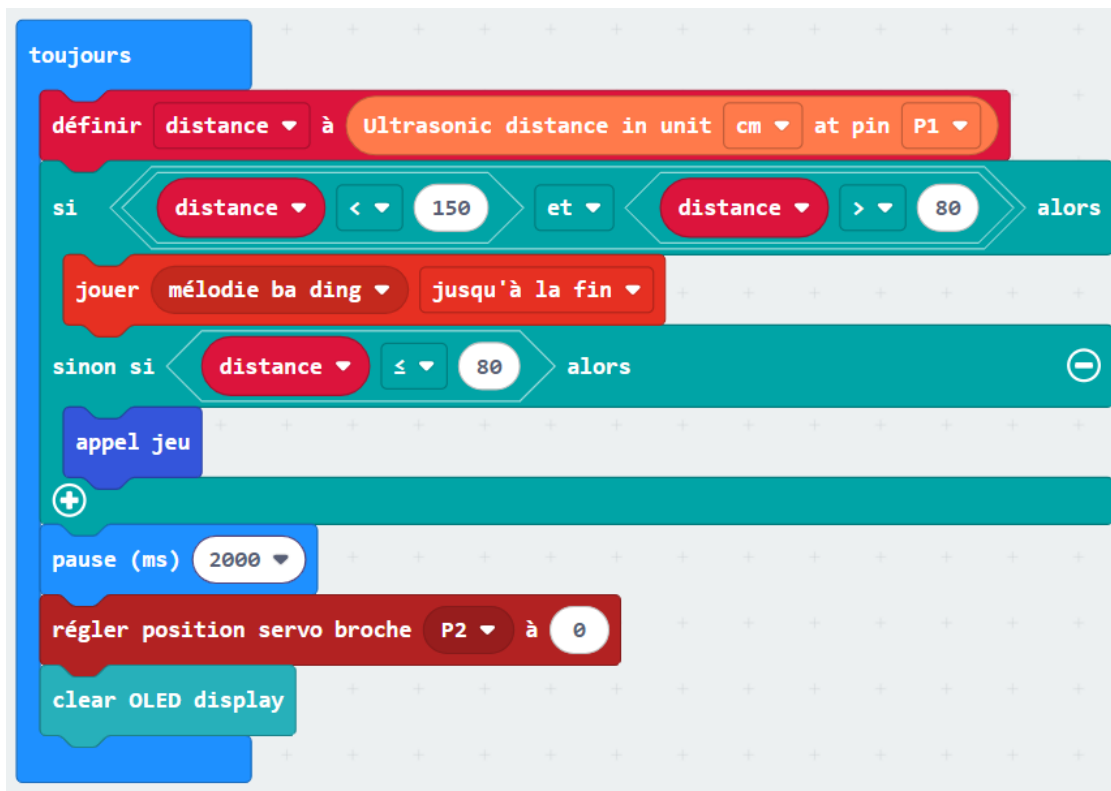
régler le volume à fait partie de la famille :  **Musique**

régler position servo se trouve dans **Avancé** sous rubrique :  **Broches**

toujours : qui répéter indéfiniment

- Lire la distance où se trouve un éventuel visiteur.
- En fonction de cette distance :
 - ne rien faire si elle est supérieure à 150 cm
 - émettre un signal sonore si elle est comprise entre 150 et 80 cm
 - démarrer un jeu pour l'ouverture du coffre si elle est inférieure à 80 cm.
- Attendre un moment
- Réinitialiser le servo et effacer l'écran.

La variable **distance** est définie comme indiqué dans le projet précédent.

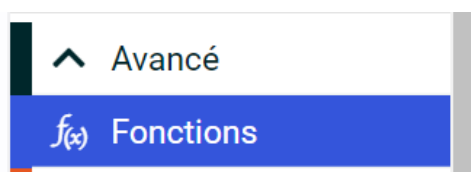


Dans le programme une partie du code se dissimule sous le bloc :



Pour obtenir ce bloc et pouvoir l'insérer dans ton programme, tu vas créer 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 :



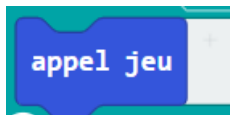
Dans l'encoche tu insères les blocs que tu souhaites regrouper en une fonction.

Puis tu changes le nom `FaireQuelqueChose` en un nom en rapport avec ce que la fonction fait, ici ce sera `jeu`.

Tu obtiens alors l'élément suivant dans lequel tu vas entrer le code du jeu, sous forme de blocs.



Une fois que tu auras terminé le code tu pourras l'utiliser en insérant dans ton programme le bloc :



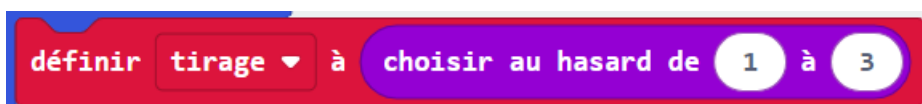
Pour cela, tu retourneras dans la famille `Créer une fonction` et tu vois que tu disposes d'un nouveau bloc avec le mot `appel` devant le nom de ta fonction.

Le but de tout cela est de rendre ton programme plus facile à construire et à relire.

Le jeu (qui se trouvera la fonction `jeu` que tu viens e créer).

Pour contrôler l'ouverture du coffre un nombre compris entre 1 et 3 est tiré au sort et le visiteur tente sa chance en appuyant les boutons A (1), B (2) ou A+B (3) du Micro:bit. Si son choix correspond au nombre tiré au sort le coffre s'ouvre grâce au servo.

Pour réaliser le tirage aléatoire des nombres 1, 2 ou 3 l'instruction `choisir au hasard de...` la famille `Maths` est utilisée, son résultat sera mis dans une variable que tu créeras et appelleras `tirage`.



Pour interagir avec les boutons tu utiliseras les blocs `lorsque le bouton...` de la famille `Entrée`.

Synchronisation : le jeu doit attendre les boutons !

Lorsque tu appuies sur un bouton le bloc **lorsque le bouton...** correspondant est appelé. Mais comment pouvoir :

- 1) Faire attendre la fonction jeu jusqu'au moment où tu appuies sur un bouton.
- 2) Dire à la fonction jeu quel bouton a été appuyé.

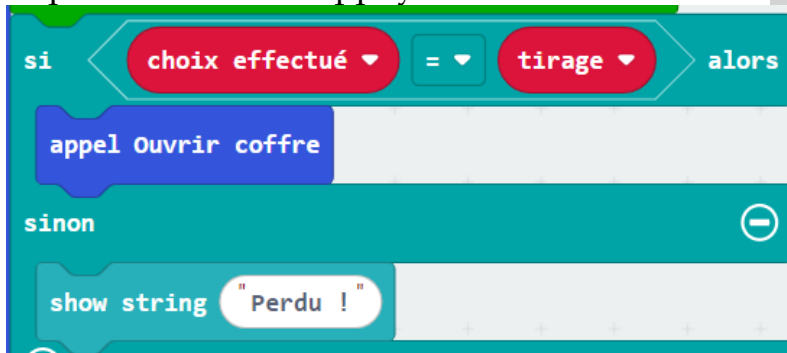
L'astuce est d'utiliser une **variable partagée** entre la fonction jeu et les blocs **lorsque le bouton...** et de convenir que la variable contiendra 0 tant que l'on a pas appuyé sur un bouton, sinon les valeurs 1 pour A, 2 pour B et 3 pour A+B.

Tu devras créer cette variable **choix effectué** et la mettre à 0 au début de la fonction

Pour attendre, tu utiliseras une boucle :

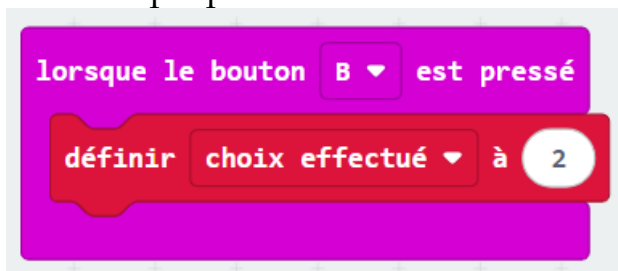


Et pour voir si tu as appuyé sur le bon bouton un **si** :

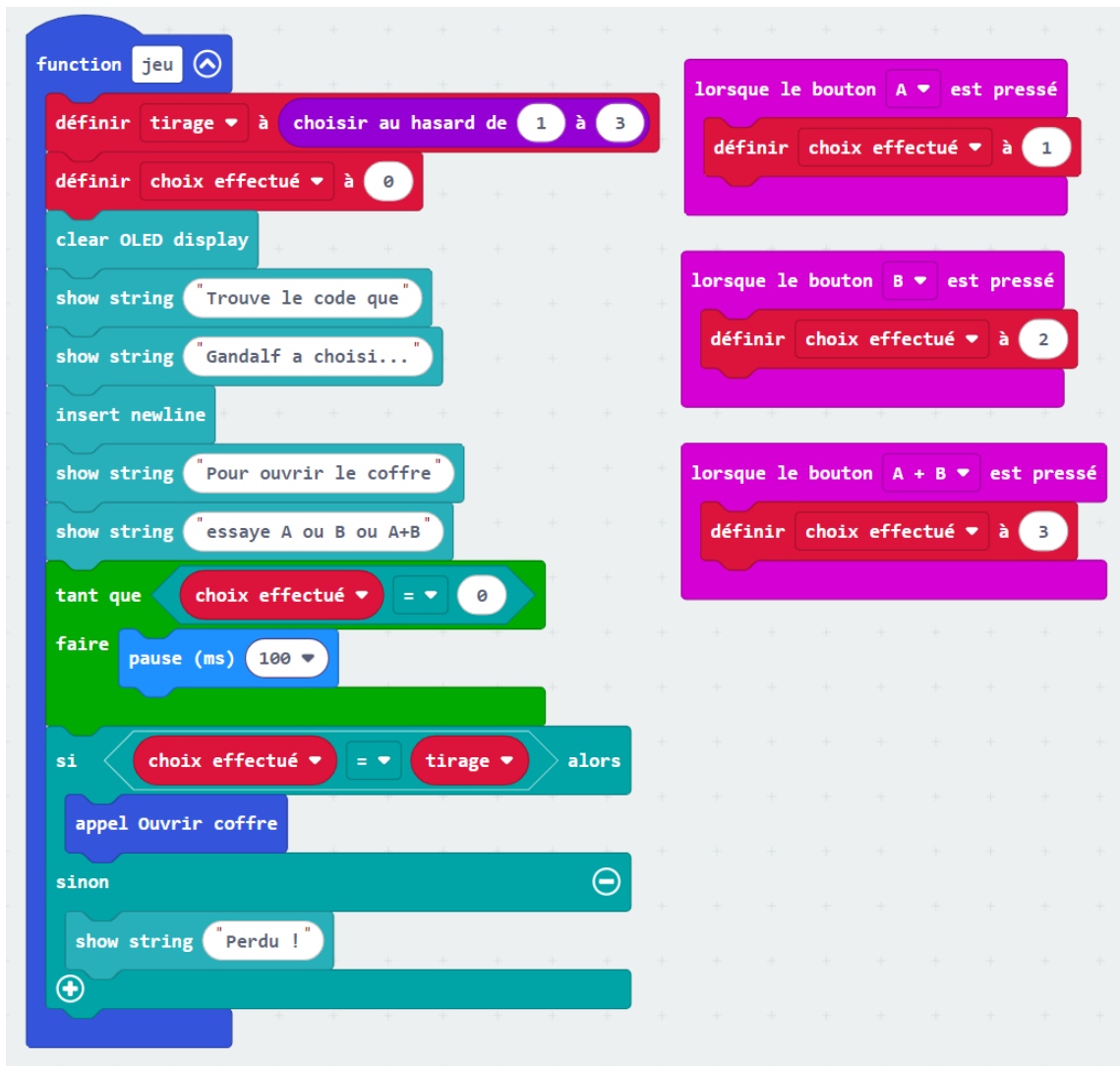


Chaque bloc **lorsque le bouton...** contiendra un **définir** qui fait passer la valeur de **choix effectué** de 0 à la valeur liée au bouton.

Par exemple pour le bouton B :



Ce qui donne en assemblant les morceaux :



Reste la fonction **Ouvrir coffre** qui sera créée comme expliqué pour la fonction **jeu**

