UTILISER DES BOUTONS-POUSSOIRS

PROGRAMMATION EN MICROPYTHON

De quoi parle-t-on ?

Cette fiche permet de lire **l'état d'un bouton** de la STeaMi en MicroPython et de déclencher des actions selon qu'il est **appuyé** ou **relâché**. Il s'agit de la brique de base pour construire de **l'interaction utilisateur** sur des objets électroniques : sonnette, jeux, télécommandes.

Disponible sur

MicroPython

Durée

35 min

Matériel

- 1 carte **STeaMi**
- 1 câble USB de données (micro-USB pour la STeaMi V1, USB-C pour la STeaMi V2)
- 1 ordinateur sous Windows, macOS ou Linux

Niveau de difficulté

Intermédiaire

Un éditeur MicroPython installé et configuré pour la STeaMi : Thonny (voir fiche de prise en main) ou tout autre éditeur compatible MicroPython (Mu, VS Code, Vittascience, mpremote...)

STeaMi

OBJECTIFS D'APPRENTISSAGE

- Lire l'état d'un bouton intégré à la STeaMi en MicroPython
- Comprendre la logique inverse des boutons et le principe de la résistance de tirage
- Détecter une transition d'état (relâché ➜ appuyé)
- Combiner lecture de bouton et pilotage de LED
- Découvrir ce qu'est une « machine à états » avec un mini-jeu à deux joueurs

Cette fiche d'activité a été produite par le Laboratoire d'Aix-périmentation et de Bidouille dans le cadre de l'action EXAO du projet I-Novmicro 2, une opération soutenue par l'État dans le cadre de l'AMI Compétences et Métiers d'Avenir du Programme France 2030, opéré par la Caisse des Dépôts, avec le soutien de la Région Sud.

UTILISER DES BOUTONS-POUSSOIRS

ÉTAPE 1 - CONSTRUIRE

Le bouton-poussoir est un composant simple pour faire le lien **entre une action physique** (un doigt qui appuie dessus) **et une action numérique** (un programme qui réagit). On en trouve partout : clavier d'ordinateur, interrupteur de chevet, manette de console, bouton d'arrêt d'urgence. Tous fonctionnent sur le même principe : **un appui ferme un circuit** (laisse le courant électrique passer), **un relâchement le rouvre** (empêche le courant de passer). La STeaMi intègre **trois boutons-poussoirs simples** directement utilisables : les boutons **A**, **B** et **Menu**. La carte possède aussi un **bouton multidirectionnel** que nous n'utiliserons pas dans cette activité.

Localiser les boutons

Sur la STeaMi, trois boutons-poussoirs sont accessibles dans le code via des noms parlants :

- Bouton A = **A_BUTTON**
- Bouton B = **B_BUTTON**
- Bouton Menu = **MENU_BUTTON**

Logique inverse : value() renvoie 0 quand on APPUIE

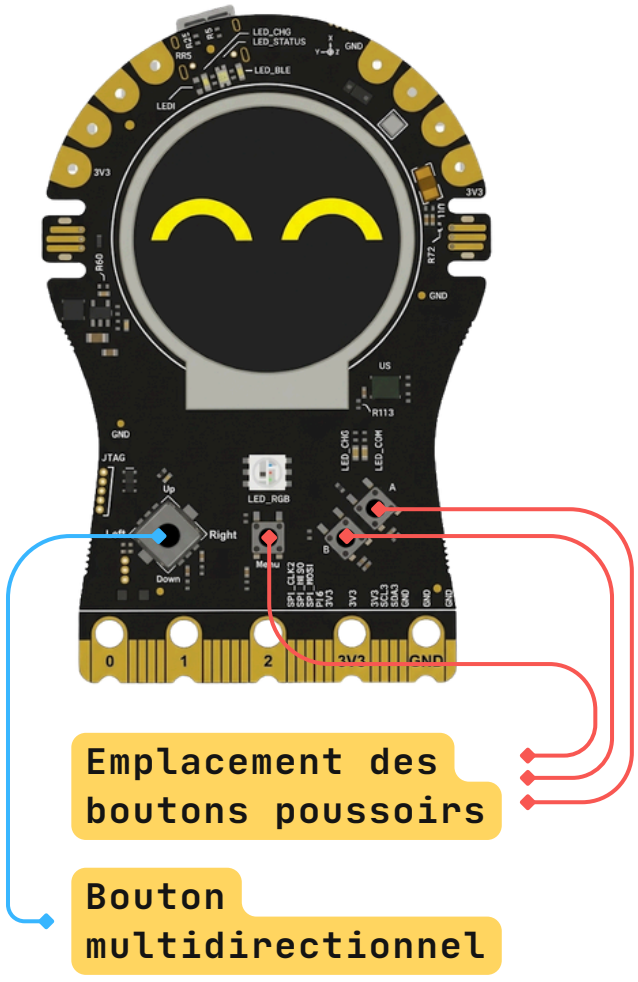
Spontanément, on pourrait penser que « **bouton appuyé = 1** » et « **bouton relâché = 0** ». Sur la STeaMi (comme sur la plupart des cartes microcontrôleurs), c'est l'**inverse** :

- Bouton relâché (repos) = Tension sur la broche de 3,3 V et value(1)
- Bouton appuyé = Tension sur la broche de 0 V et value(0)

Un bouton-poussoir, ce n'est rien d'autre qu'un interrupteur qui ferme un fil quand on appuie. Sans précaution, quand le bouton est relâché, la broche du microcontrôleur n'est connectée à rien : elle « flotte » et lit n'importe quoi (du bruit électrique). Pour éviter ça, il faut **forcer une valeur par défaut**, et c'est précisément le rôle d'une **résistance de tirage**.

Sur la STeaMi, cette résistance (4,7 kΩ) relie chaque broche de bouton à l'alimentation (3,3 V). Tant que personne n'appuie, elle « tire » doucement la broche vers le haut : on lit 1. Appuyer sur le bouton crée un chemin direct vers la masse (0 V), beaucoup plus « fort » que la résistance : la broche tombe à 0.

1



UTILISER DES BOUTONS-POUSSOIRS

ÉTAPE 1 - CONSTRUIRE

Ce câblage « **tirage vers le haut** » (pull-up en anglais) est la convention la plus courante en électronique embarquée. C'est pour cela que la quasi-totalité des cartes éducatives (STeaMi, micro:bit, Arduino, Raspberry Pi Pico...) renvoient 0 quand on appuie.

Conséquence pratique : dans le code, on teste btn_a.value() == 0 pour savoir si A est appuyé, pas == 1.

Connecter la carte à l'ordinateur

2

Branchez la carte STeaMi à l'ordinateur grâce au câble USB et ouvrez votre éditeur. Il faudra s'assurer que l'éditeur a été configuré au préalable (par exemple pour Thonny, voir la fiche de prise en main).

Si votre éditeur est déjà configuré, une fois lancé, la console MicroPython doit afficher le prompt '>>>'.

Programmer, exécuter, jouer

3

Afin de programmer votre carte, vous pouvez utiliser le programme fourni dans la section "Programmer". Ce programme propose un **mini-jeu à deux joueurs** : **la première personne qui appuie sur son bouton allume sa LED**. Le bouton A allume la LED rouge, le bouton B allume la LED bleue. Tant qu'un bouton est maintenu enfoncé, l'autre joueur ne peut plus gagner.

Une fois le code en place, vous avez deux manières de le lancer :

Test rapide. Lancez le programme depuis l'éditeur (typiquement bouton Run ▶️ ou F5).

Programme persistant. Enregistrez le fichier sous le nom main.py sur la carte : il sera relancé à chaque démarrage, sans avoir à rouvrir l'éditeur.

Une fois le programme lancé, les actions suivantes sont réalisables :

- Appuyer sur A : la LED rouge s'allume.
- Appuyer sur B : la LED bleue s'allume.

Chaque joueur choisit son bouton et l'objectif est de savoir qui sera le plus rapide. Le premier joueur qui fait éclairer sa LED a gagné le tour. C'est l'arbitre qui donnera le signal au joueur pour réagir.

Tant qu'un bouton reste enfoncé, l'autre n'a plus aucun effet. Relâcher les deux boutons permettra aux LED de s'éteindre, le tour suivant peut commencer.

À deux personnes, c'est un mini-jeu de réflexes. Seul, c'est l'occasion d'observer en temps réel comment chaque appui modifie l'état du programme.

UTILISER DES BOUTONS-POUSSOIRS

ÉTAPE 2 - PROGRAMMER

```
# Testée avec firmware STeaMi 0.23.1
from machine import Pin
from time import sleep_ms

# LED RGB de la STeaMi (sortie push-pull)
led_r = Pin('LED_RED', Pin.OUT)
led_b = Pin('LED_BLUE', Pin.OUT)

# Boutons A et B (résistance de tirage externe : 1 au repos, 0 quand on appuie)
btn_a = Pin('A_BUTTON', Pin.IN)
btn_b = Pin('B_BUTTON', Pin.IN)

# Drapeau : un appui est-il possible pour le tour en cours ?
# Tant que `tour_libre` est False, plus aucune LED ne réagit
# avant que les deux boutons soient relâchés.
tour_libre = True

while True:
    a_actif = btn_a.value() == 0
    b_actif = btn_b.value() == 0

    if not a_actif and not b_actif:
        # Les deux boutons sont relâchés : on éteint les LED et
        # on rouvre le tour suivant.
        led_r.off()
        led_b.off()
        tour_libre = True
    elif tour_libre and a_actif:
        # Premier appui du tour, sur A : LED rouge.
        led_r.on()
        tour_libre = False
    elif tour_libre and b_actif:
        # Premier appui du tour, sur B : LED bleue.
        led_b.on()
        tour_libre = False

    # Petite pause pour ne pas saturer le processeur et atténuer
    # les rebonds mécaniques du bouton (cf. la fiche concept
    # "Anti-rebond" en Aller plus loin).
    sleep_ms(20)
```

UTILISER DES BOUTONS-POUSSOIRS

ÉTAPE 2 - PROGRAMMER

Fonctionnement du programme

Le programme s'organise en quatre parties :

- **Initialisation** : on déclare les deux LED (**Pin.OUT**) et les deux boutons (**Pin.IN**). Le firmware STeaMi expose les composants sous des noms parlants (**'LED_RED', 'A_BUTTON'**...). Pas besoin de mémoriser un numéro de broche.
- **Lecture des boutons** : **btn_a.value()** renvoie 1 ou 0. À cause de la logique inverse, on compare à 0 pour savoir si le bouton est appuyé. On stocke le résultat dans **a_actif** (booléen) pour rendre la suite plus lisible.
- **Détection de transition** : la variable **tour_libre** joue le rôle de drapeau. Elle prend la valeur **False** dès qu'on allume une LED, ce qui empêche l'autre bouton de prendre la main. Elle ne reprend la valeur **True** que quand les deux boutons sont relâchés : c'est ce qui marque la fin du tour.
- **Boucle principale** : à chaque tour de boucle, on lit les deux boutons et on choisit la branche **if/elif/elif** appropriée. Les LED restent allumées tant qu'un bouton est appuyé. Le **sleep_ms(20)** final laisse souffler le processeur (sans pause, la boucle tournerait à 100 % CPU) et atténue les rebonds mécaniques du contact (le bouton se ferme plusieurs fois en quelques millisecondes au moment de l'appui).

Machine à états

Le drapeau **tour_libre** transforme cette boucle en une machine à états : on est soit dans l'état « tour libre » (aucune LED allumée, on attend un appui), soit dans l'état « tour gagné » (une LED allumée, on attend que les deux boutons se relâchent). Ce principe revient partout en programmation interactive (distributeurs, menus, jeux vidéo).

UTILISER DES BOUTONS-POUSSOIRS

ÉTAPE 3 - AMÉLIORER

Compter les points

Ajoutez deux compteurs **score_a** et **score_b** qui s'incrémentent à chaque tour gagné, puis affichez le score dans la console avec **print()**. Variante : remettez les compteurs à zéro avec le bouton Menu.

1

Faire clignoter la LED du gagnant

Plutôt que d'allumer la LED en continu, faites-la clignoter quelques fois pour bien marquer la victoire. La technique est dans la fiche **Faire clignoter une LED**.

2

Afficher le résultat sur l'écran

Combinez cette fiche avec la fiche **Afficher du texte sur l'écran OLED** pour afficher le nom du gagnant en grand sur l'écran intégré.

3

Jeu de réaction (bonus)

Au lieu de lancer la partie immédiatement, programmez un délai aléatoire (**random.uniform(1, 5) secondes**), puis allumez brièvement les deux LED simultanément comme signal de départ. Le premier qui appuie après le signal gagne ; celui qui appuie avant a un faux départ.

4

ALLER PLUS LOIN

Bouton-poussoir (Wikipedia) - Histoire, types (à fermeture, à ouverture, à accrochage), applications. Le composant le plus simple, mais le pilier de presque toutes les interfaces. *https://fr.wikipedia.org/wiki/Bouton-poussoir*

Le buzzer de jeu télévisé - Exactement le même principe que notre mini-jeu, mais à l'échelle d'un plateau de télévision (Questions pour un champion, Slam, Burger Quiz...). Premier appuyé, premier servi. *https://fr.wikipedia.org/wiki/Buzzer_(jeu_t%C3%A9l%C3%A9vis%C3%A9)*

Borne d'arcade (projet JediTrack) - Fiche du wiki sur la fabrication d'une borne d'arcade complète. Les boutons d'arcade fonctionnent exactement comme ceux de la STeaMi, en plus gros et en plus colorés. *https://wiki.labaixbidouille.com/ressources/jeditrack/borne-arcade*

Manette NES (Nintendo) - 8 boutons-poussoirs sur un PCB, et toute l'histoire du jeu vidéo des années 80. Une bonne occasion de regarder ce qu'il y a à l'intérieur d'un objet familier. *https://fr.wikipedia.org/wiki/Manette_de_Nintendo_Entertainment_System*