RESSOURCES - PROGRAMMATION - CARTE STEAMI

PRISE EN MAIN DE MICROPYTHON

DÉCOUVERTE DE **THONNY** POUR LA STEAMI

De quoi parle-t-on ?

Cette fiche met en place tout l'environnement nécessaire pour programmer la STeaMi : installer le logiciel Thonny, installer MicroPython sur la carte, et régler la communication entre les deux. Une fois cet environnement prêt, vous pourrez suivre toutes les autres fiches (LED, buzzer, capteurs, écran...), qui supposent cette mise en route déjà faite.

Éditeur

Thonny (version 4.x ou supérieure), l'éditeur MicroPython dont cette fiche détaille l'installation et la configuration : https://thonny.org/

Durée

40 min

Niveau de difficulté

Débutant

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
- Le programme MicroPython STeaMi .hex disponible ici : https://github.com/steamicc/micropython-steami-lib/releases

STeaMi

OBJECTIFS D'APPRENTISSAGE

- Comprendre le principe de la programmation embarquée : un programme écrit sur l'ordinateur est envoyé à la carte, qui l'exécute ensuite par elle-même
- Distinguer le rôle de l'éditeur (Thonny), du langage (MicroPython) et de la carte (STeaMi)
- Savoir préparer un environnement de programmation pour la STeaMi

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.

PRISE EN MAIN DE MICROPYTHON SUR THONNY

ÉTAPE 1 - CONSTRUIRE

Thonny est un logiciel **conçu pour l'apprentissage** qui simplifie l'accès à la programmation en Python dans l'environnement scolaire. Associé à MicroPython, une version de Python adaptée aux cartes électroniques, il offre un espace de travail **gratuit, utilisable sans connexion internet**, où l'on peut écrire du code, le tester en direct dans le REPL (une fenêtre de dialogue où l'on tape une instruction et la carte y répond immédiatement), et déboguer pas-à-pas.

Thonny n'est qu'un choix parmi d'autres : tout éditeur compatible MicroPython (Mu, VS Code, Vittascience, mpremote, etc.) permet de programmer la STeaMi de la même façon. Cette fiche s'appuie sur Thonny pour fixer les idées, mais la démarche reste valable sur d'autres outils.

Installer Thonny

Sous **Windows** ou **macOS**, téléchargez le programme d'installation sur thonny.org, lancez le fichier, suivez les étapes, puis ouvrez Thonny depuis le menu Démarrer ou le Launchpad.

Sous **Linux** (Debian / Ubuntu), il faudra saisir l'une de ces commandes ci-contre dans un terminal :

1

2

```
# Via les dépôts de la distribution (recommandé pour un public débutant)
sudo apt install thonny

# Ou via pipx pour avoir la dernière version
# (pipx doit être installé au préalable : 
#  sudo apt install pipx && pipx ensurepath)
pipx install thonny
```

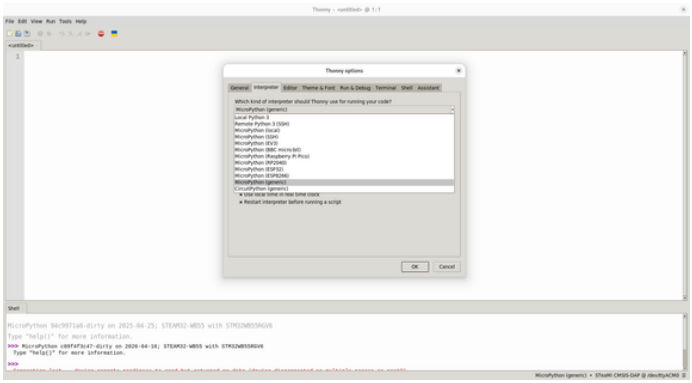
Commandes sous Linux

Installer MicroPython sur la STeaMi

Une STeaMi sortie d'usine est en général livrée avec **MicroPython déjà installé**. Si après l'étape 3 "**Configurer Thonny pour la STeaMi**" l'invite >>> apparaît dans le Shell de Thonny (le signe en début de ligne qui indique que MicroPython attend une commande, équivalent d'un « prompt »), cette étape d'installation est déjà faite : **passez directement à la section suivante**.

Dans le cas contraire, grâce à un mode "clé USB" préinstallé, la STeaMi se présente comme une clé USB : installer MicroPython revient à un simple glisser-déposer.

- **Branchez** la STeaMi en USB (câble de données, pas un câble de charge seul). La carte apparaît comme un disque amovible nommé STEAMI.



Outils → Options → Interpréteur : choisir « MicroPython (generic) » et le port de la STeaMi

PRISE EN MAIN DE MICROPYTHON SUR THONNY

ÉTAPE 1 - CONSTRUIRE

- **Téléchargez** le fichier steami-micropython-firmware-vX.Y.Z.hex depuis les releases (https://github.com/steamicc/micropython-steami-lib/releases). *Attention : ne pas confondre avec steami-daplink-firmware-...hex, qui est un autre fichier sans rapport avec MicroPython.*
- **Glissez-déposez** le .hex sur le disque STEAMI.

La LED de statut clignote pendant l'écriture (~5 à 15 s), puis la carte redémarre avec MicroPython. Ne débranchez pas la carte pendant le clignotement et attendez la fin du redémarrage.

Si le disque STEAMI n'apparaît pas, le premier réflexe est de changer de câble : un câble qui ne transporte que l'alimentation ne suffit pas, il faut un câble de données.

Configurer Thonny pour la STeaMi

Si Thonny démarre en anglais, allez dans Tools → Options → onglet General → Language: Français, puis redémarrez Thonny :

- Ouvrez Thonny.
- Outils → Options... → onglet ***Interpréteur*** (le terme employé par Thonny pour désigner l'appareil sur lequel le code va s'exécuter).
- Dans la liste "***Quel interpréteur ou appareil utiliser***", choisissez **MicroPython** (generic).
- Dans ***Port*** (la prise par laquelle l'ordinateur dialogue avec la carte), sélectionnez celui de la STeaMi : **/dev/ttyACM0** sous Linux, **/dev/cu.usbmodemXXXX** sous macOS, **COM3 / COM4...** sous Windows.
- Cliquez OK.

Le panneau Shell affiche alors :

```
MicroPython v1.XX.X on YYYY-MM-DD; STeaMi
with STM32WB55RG
Type "help()" for more information.
>>>
```

Le prompt >>> confirme que Thonny dialogue avec MicroPython sur la STeaMi.

3

Identifier le bon port

- **Windows** : si plusieurs ports COM apparaissent, ouvrir le Gestionnaire de périphériques (clic droit sur le menu Démarrer), section Ports (COM et LPT). Brancher/débrancher la STeaMi pour repérer celui qui apparaît et disparaît.
- **Linux** : en cas d'erreur Permission denied sur le port série, ajouter l'utilisateur au groupe dialout avec la commande ci-dessous, puis se déconnecter / se reconnecter (ou redémarrer la session) pour que le changement prenne effet.

```
sudo usermod -aG dialout $USER
```

PRISE EN MAIN DE MICROPYTHON SUR THONNY

ÉTAPE 1 - CONSTRUIRE

Programmer, exécuter, jouer

Pour programmer votre carte, vous pouvez utilisez les programmes fournis dans les sections “Programmer” de nos fiches ou jouez par vous-même avec l’éditeur. Un programme-exemple simple est disponible ci-dessous.

Pour chaque programme créé :

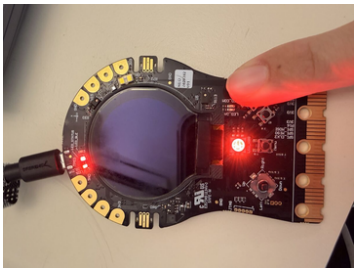
Test rapide. Cliquez sur Run ▶️ (ou F5). Le code s'exécute aussitôt sur la carte, sans être enregistré : idéal pour essayer rapidement.

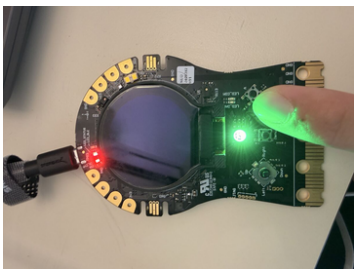
Pour garder le programme sur la carte, ouvrez Fichier ➜ Enregistrer sous.... Thonny demande alors où enregistrer : choisissez MicroPython device (la carte) et non l'ordinateur, puis nommez le fichier main.py. C'est ce nom précis qui fait que le programme se relance tout seul à chaque démarrage de la carte.

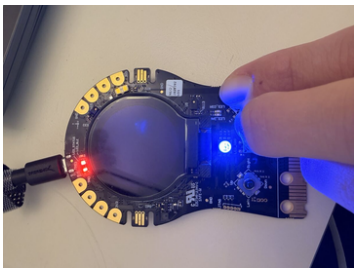
Si un programme est déjà en cours sur la carte, Thonny refusera d'en lancer un autre : arrêtez-le d'abord avec Ctrl+C dans la console, ou cliquez sur le bouton Stop (■).

3

4







PRISE EN MAIN DE MICROPYTHON SUR THONNY

ÉTAPE 2 - PROGRAMMER

Voici un premier programme pour tester votre configuration : **changer la couleur de la LED RGB selon le bouton enfoncé**. Sur la STeaMi, la LED RGB s'allume en écrivant 1 sur la broche, et s'éteint avec 0. Les boutons A et B, eux, fonctionnent à l'envers : leur valeur vaut 1 au repos et passe à 0 quand on appuie (une résistance présente sur la carte impose ce comportement, on n'a rien à faire dans le code).

Brochage utilisé

Composant	Nom dans le programme	Comportement
LED RGB Rouge	LED_RED	1 = allumée, 0 = éteinte
LED RGB Verte	LED_GREEN	1 = allumée, 0 = éteinte
LED RGB Bleue	LED_BLUE	1 = allumée, 0 = éteinte
Bouton A	A_BUTTON	0 = appuyé, 1 = relâché
Bouton B	B_BUTTON	0 = appuyé, 1 = relâché

```
# Testée avec firmware STeaMi 0.23.1

# Premier programme STeaMi avec Thonny, LED RGB + boutons A/B
# - Bouton A     -> LED rouge
# - Bouton B     -> LED verte
# - A + B        -> LED bleue
# - Aucun bouton -> LED éteinte

from machine import Pin
from time import sleep_ms

# LED RGB : on() allume, off() éteint
led_r = Pin('LED_RED', Pin.OUT)
led_g = Pin('LED_GREEN', Pin.OUT)
led_b = Pin('LED_BLUE', Pin.OUT)

# Boutons A et B (résistance pull-up sur la carte : 1 au repos, 0 quand on appuie)
btn_a = Pin('A_BUTTON', Pin.IN)
btn_b = Pin('B_BUTTON', Pin.IN)


def set_rgb(r, g, b):
    """Allume chaque LED selon la composante (1 = on, 0 = off)."""
    led_r.value(r)
    led_g.value(g)
    led_b.value(b)
```

PRISE EN MAIN DE MICROPYTHON SUR THONNY

ÉTAPE 2 - PROGRAMMER

```
print("Programme démarré. Appuyez sur A, B ou les deux.")

while True:
    a_pressed = btn_a.value() == 0
    b_pressed = btn_b.value() == 0

    if a_pressed and b_pressed:
        set_rgb(0, 0, 1)   # bleu
    elif a_pressed:
        set_rgb(1, 0, 0)   # rouge
    elif b_pressed:
        set_rgb(0, 1, 0)   # vert
    else:
        set_rgb(0, 0, 0)   # éteint

    sleep_ms(20)
```

Quand vous le lancez (voir « Programmer, exécuter, jouer » dans l'étape Construire), appuyez sur A, B ou les deux : la LED change de couleur, et le message s'affiche dans la console. La communication entre l'ordinateur et la carte fonctionne

Fonctionnement du programme

- **Préparation des composants**. Les premières lignes donnent un nom à chaque élément et précisent comment on l'utilise. **Pin('LED_RED', Pin.OUT)** déclare la LED rouge en sortie (la carte lui envoie une valeur pour l'allumer ou l'éteindre), tandis que **Pin('A_BUTTON', Pin.IN)** déclare le bouton A en entrée (la carte lit sa valeur). On répète l'opération pour les trois LED et les deux boutons.
- **La fonction set_rgb**. Plutôt que de réécrire trois lignes à chaque fois, on regroupe l'allumage des trois LED dans une petite fonction : **set_rgb(1, 0, 0)** allume le rouge et éteint les deux autres.
- **La boucle. while True** répète sans fin. À chaque tour, le programme lit l'état des deux boutons, allume la couleur correspondante, puis fait une courte pause **sleep_ms(20)**. La boucle tourne ainsi environ 50 fois par seconde : assez vite pour que la réponse paraisse instantanée, assez doucement pour ne pas surcharger la carte.

PRISE EN MAIN DE MICROPYTHON SUR THONNY

ÉTAPE 3 - AMÉLIORER

Tester du code en direct dans la console

Le REPL (>>> dans le panneau Shell) permet de tester du code directement sur la carte, sans créer de fichier. On écrit une instruction, la carte répond aussitôt. Cette fonctionnalité est pratique pour essayer une idée ou vérifier un détail. Par exemple, tapez **led_g.value(0)** pour éteindre la LED, sans relancer tout le programme. Trois raccourcis utiles : **Ctrl+C** arrête le programme en cours, **Ctrl+D** le relance, et la flèche ↑ rappelle la dernière commande tapée.

Déboguer pas-à-pas

Thonny propose un débogueur intégré, particulièrement adapté à l'enseignement :

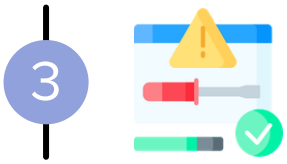
1. Cliquer dans la marge à gauche d'une ligne pour poser un **point d'arrêt**.
2. **Run > Debug current script** (**Ctrl+F5**).
3. Avancer avec Step over (**F6**), Step into (**F7**), Step out (**F8**).
4. Observer les variables dans le panneau Variables.

Le débogueur Thonny est plus efficace en local sur le PC qu'en cible embarquée. Pour des programmes qui interagissent beaucoup avec le matériel, le print() dans le REPL reste souvent plus pratique.

Dépanner les erreurs courantes

La plupart des problèmes rencontrés en classe ne sont pas spécifiques à Thonny mais touchent le matériel ou l'environnement MicroPython (câble, port série, programme bloqué) :

- la carte qui n'apparaît pas comme disque STEAMI (câble) ;
- le port série introuvable ou avec accès refusé (Windows / Linux) ;
- la console vide après connexion (MicroPython pas installé) ;
- Couldn't find the device (plusieurs cartes branchées) ;
- Device is busy (programme déjà en cours) ;
- un main.py qui redémarre en boucle.



PRISE EN MAIN DE MICROPYTHON SUR THONNY

ALLER PLUS LOIN

MOOC FUN, Programmer un objet avec MicroPython : un cours en ligne gratuit (CC-BY-SA) qui couvre les bases de MicroPython sur Pyboard / ESP32 / micro:bit. Idéal en complément pour s'approprier le langage et son écosystème, avant de l'enseigner. *https://www.fun-mooc.fr/fr/cours/programmer-un-objet-avec-micropython/*

L'histoire de MicroPython : lancé en 2013 par Damien George via une campagne Kickstarter, MicroPython est une implémentation de Python pensée pour les systèmes embarqués, un interpréteur compact capable de tourner sur des cartes avec très peu de mémoire, qui a démocratisé Python sur microcontrôleur. *https://fr.wikipedia.org/wiki/MicroPython*

Thonny, pensé pour l'enseignement : conçu par Aivar Annamaa (Université de Tartu) pour rendre Python accessible aux débutant·es. Visualisation pas-à-pas des variables, débogueur didactique, installation sans dépendance. *https://thonny.org/*

Documentation technique pour préparer une séquence ou répondre aux questions des élèves :

- **Présentation matérielle** sur le site de la STeaMi : *https://www.steami.cc/*
- **Wiki STeaMi Thonny** : *https://wiki.steami.cc/docs/software/micropython/thonny*
- **Wiki STeaMi Premiers pas** : *https://wiki.steami.cc/docs/software/getting-started*
- **Wiki STeaMi Hardware** (pinout détaillé) : *https://wiki.steami.cc/docs/hardware/*
- **Drivers MicroPython STeaMi** (code source des modules steami_*) : *https://github.com/steamicc/micropython-steami-lib*
- **Documentation MicroPython** (référence complète du langage et des modules) : *https://docs.micropython.org/*