RESSOURCES - PROGRAMMATION - CARTE STEAMI

ENVOYER DES MESSAGES

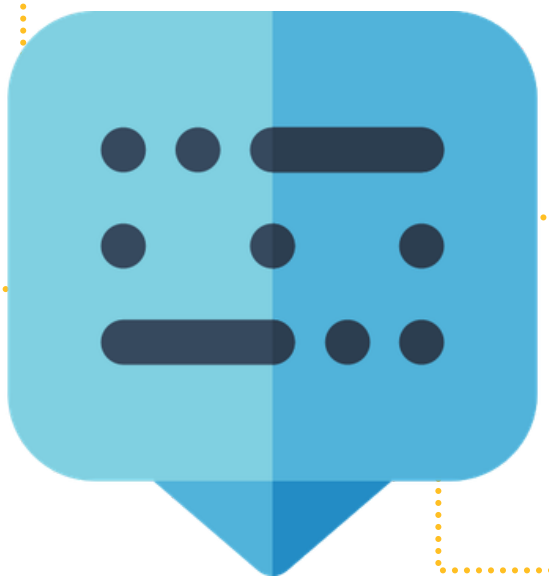
EN **CODE MORSE** AVEC LA STEAMI EN MICROPYTHON

De quoi parle-t-on ?

Cette fiche permet d'envoyer des messages en **code Morse** depuis la STeaMi, à l'aide du buzzer et des boutons intégrés. Le code Morse est l'une des premières méthodes de télécommunication numérique : il code chaque lettre de l'alphabet par une suite de points (signaux courts) et de tirets (signaux longs).

Disponible sur

MicroPython



Durée

30 min

Niveau de difficulté

Avancé

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

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

- Comprendre comment une carte génère un son en faisant vibrer un buzzer
- Découvrir comment générer une fréquence en alternant rapidement un signal numérique
- Identifier la structure temporelle du code Morse
- Lire et coder une séquence de caractères en Morse

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.

ENVOYER DES MESSAGES EN CODE MORSE

ÉTAPE 1 - CONSTRUIRE

Inventé dans les années 1830 pour le télégraphe, le code Morse a été utilisé pendant plus d'un siècle pour la marine, l'aviation et la radio amateur. Aujourd'hui encore, il reste enseigné comme exemple de codage à longueur variable : les lettres les plus fréquentes en anglais (E, T) sont les plus courtes à transmettre, ce qui inspire des techniques de compression modernes. La STeaMi intègre tout le matériel nécessaire pour transmettre du Morse : un **buzzer piézo pour le signal sonore, et les boutons A et B pour saisir les points et les tirets**.

Comprendre le buzzer de la STeaMi

La carte STeaMi est équipée d'un **transducteur piézoélectrique** (*https://fr.wikipedia.org/wiki/Capteur_pi%C3%A9zo%C3%A9lectrique*) soudé sur la face arrière. Quand on lui applique une tension qui varie rapidement, une fine plaque de céramique à l'intérieur **se déforme à la même vitesse**, et ces vibrations produisent une **onde sonore**. La fréquence de la tension détermine la hauteur du son : **440 Hz** donne la note La, **880 Hz** donne la note La de l'octave au-dessus, et ainsi de suite. Les fréquences audibles vont d'environ **20 Hz à 20 000 Hz**.

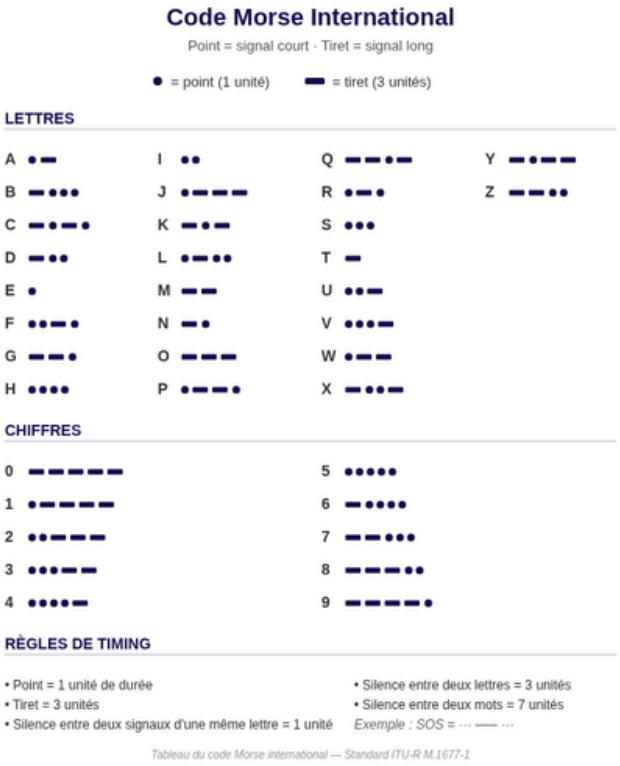
Distinguer buzzer actif et buzzer passif

Il existe deux familles de buzzers. Un buzzer **actif** contient déjà l'électronique pour générer une fréquence fixe : il suffit de l'alimenter et il sonne, toujours à la même note. Un buzzer **passif**, **comme celui de la STeaMi**, ne sonne que si on lui envoie un signal qui varie. Plus complexe à piloter, il permet cependant de jouer n'importe quelle note.

Générer une fréquence à la main

Le buzzer de la STeaMi se pilote en alternant très rapidement la broche entre allumé **(3,3 V)** et éteint **(0 V).** Si on alterne **440 fois par seconde**, le buzzer vibre à **440 Hz** et on entend la **note La**. C'est le programme lui-même qui se charge de ce va-et-vient, d'où l'utilité d'une **fonction tone()** qui gère l'alternance.

1



Code Morse International

Point = signal court · Tiret = signal long

● = point (1 unité) ▬ = tiret (3 unités)

LETTRES

A ●▬	I ●●	Q ▬▬●▬	Y ▬●▬▬
B ▬●●●	J ●▬▬▬	R ●▬●	Z ▬▬●●
C ▬●▬●	K ▬●▬	S ●●●	
D ▬●●	L ●▬●●	T ▬	
E ●	M ▬▬	U ●●▬	
F ●●▬●	N ▬●	V ●●●▬	
G ▬▬●	O ▬▬▬	W ●▬▬	
H ●●●●	P ●▬▬●	X ▬●●▬	

CHIFFRES

0 ▬▬▬▬▬	5 ●●●●●
1 ●▬▬▬▬	6 ▬●●●●
2 ●●▬▬▬	7 ▬▬●●●
3 ●●●▬▬	8 ▬▬▬●●
4 ●●●●▬	9 ▬▬▬▬●

RÈGLES DE TIMING

- Point = 1 unité de durée
- Tiret = 3 unités
- Silence entre deux signaux d'une même lettre = 1 unité
- Silence entre deux lettres = 3 unités
- Silence entre deux mots = 7 unités

Exemple : SOS = ··· ▬▬ ···

Tableau du code Morse international — Standard ITU-R M.1677-1

2

CC BY SA

ENVOYER DES MESSAGES EN CODE MORSE

ÉTAPE 1 - CONSTRUIRE

Initialiser les composants

Le firmware STeaMi expose les composants de la carte avec des noms parlants qu'on peut utiliser directement dans **Pin(...)**. Pour le programme, il faudra trois composants :

- **Buzzer** = **“SPEAKER”** : buzzer en sortie push-pull
- **Bouton A** = **“A_BUTTON”** : bouton A en entrée
- **Bouton B** = **“B_BUTTON”** : bouton B en entrée

Connecter la carte à l’ordinateur

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 ‘>>>’.

Tester le buzzer dans la console

Avant d'écrire le programme principal, on peut vérifier que le buzzer répond en tapant directement dans la console MicroPython, en **mode interactif** (appelé REPL pour Read-Eval-Print Loop : on tape une commande, elle s'exécute, on voit le résultat, et la console attend la suivante) le programme ci-contre :

3

```
from machine import Pin

SPEAKER = Pin("SPEAKER", Pin.OUT_PP)   #
buzzer en sortie push-pull
A_BUTTON = Pin("A_BUTTON", Pin.IN)     #
bouton A en entrée
B_BUTTON = Pin("B_BUTTON", Pin.IN)     #
bouton B en entrée
```

4

- **Pin.OUT_PP** signifie output push-pull : la broche peut activement imposer 0 V ou 3,3 V. C'est ce qu'il faut pour piloter un buzzer.
- **Pin.IN** signifie input : on lit l'état du bouton sans rien lui imposer. Les boutons A et B de la STeaMi sont câblés avec une résistance externe (4,7 kΩ) qui maintient la broche à 3,3 V quand le bouton est relâché. Quand on appuie, le bouton tire la broche à 0 V. C'est pour cela qu'un bouton appuyé renvoie 0 et un bouton relâché 1 (logique inverse).

Initialiser les composants

5

```
from machine import Pin
import time
buzzer = Pin("SPEAKER", Pin.OUT_PP)
# Faire vibrer le buzzer à 440 Hz pendant
500 ms
for _ in range(440 * 500 // 1000):
...     buzzer.on()
...     time.sleep_us(1136)   # demi-
période de 440 Hz
...     buzzer.off()
...     time.sleep_us(1136)
```

Le buzzer émet la note La pendant une demi-seconde. Ce code fonctionne, mais il est laborieux : on l'emballera dans une fonction **tone()** réutilisable dès dans la section “Programmer”.

Tester le buzzer

CC BY SA

ENVOYER DES MESSAGES EN CODE MORSE

ÉTAPE 1 - CONSTRUIRE

Programmer, exécuter, jouer

Afin de programmer votre carte, vous pouvez utiliser le programme fourni dans la section “Programmer”.

Notre premier programme va émettre des points et des tirets selon le bouton appuyé : le bouton A produit un signal court (point, 100 ms), le bouton B produit un signal long (tiret, 300 ms).

Test rapide. Lancez le programme depuis l’éditeur (typiquement bouton Run ▶️ ou F5). Appuyer sur A fait un bip court, sur B un bip long.

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.

Envoyer un message en Morse. Voici les règles standard pour transmettre un message :

- **La durée d'un point est 1 unité.**
- **La durée d'un tiret est 3 unités.**
- **L'espace entre deux signaux d'une même lettre est 1 unité.**
- **L'espace entre deux lettres est 3 unités.**
- **L'espace entre deux mots est 7 unités.**

Avec une unité de 100 ms (comme dans le programme), essayez d'envoyer SOS : trois points (S), pause, trois tirets (O), pause, trois points (S). Soit, sur les boutons : A A A (pause) B B B (pause) A A A.

CC BY SA

ENVOYER DES MESSAGES EN CODE MORSE

ÉTAPE 2 - PROGRAMMER

Composants utilisés

Composant	Nom dans le programme	Rôle
Buzzer	SPEAKER	Sortie numérique pilotée à la main pour générer un son
Bouton A	A_BUTTON	0 = appuyé, 1 = relâché
Bouton B	B_BUTTON	0 = appuyé, 1 = relâché

```
# Testée avec firmware STeaMi 0.23.1

# Émetteur Morse : bouton A = point, bouton B = tiret
# Le buzzer joue un La (440 Hz) pendant 100 ms (point) ou 300 ms (tiret).

from machine import Pin
import time

# Initialisation des composants
SPEAKER = Pin("SPEAKER", Pin.OUT_PP)
A_BUTTON = Pin("A_BUTTON", Pin.IN)
B_BUTTON = Pin("B_BUTTON", Pin.IN)

# Durées en millisecondes (codage Morse standard)
DUREE_POINT = 100
DUREE_TIRET = 300


def tone(pin, freq, duration_ms):
    """Fait sonner le buzzer à la fréquence demandée pendant duration_ms."""
    if freq == 0:
        time.sleep_ms(duration_ms)
        return
    period_us = int(1_000_000 / freq)
    half_period = period_us // 2
    end_time = time.ticks_add(time.ticks_us(), duration_ms * 1000)
    while time.ticks_diff(end_time, time.ticks_us()) > 0:
        pin.on()
        time.sleep_us(half_period)
        pin.off()
        time.sleep_us(half_period)
```


CC BY SA

ENVOYER DES MESSAGES EN CODE MORSE

ÉTAPE 2 - PROGRAMMER

```
def beep(duree_ms):
 """Joue un La pendant la durée demandée."""
 tone(SPEAKER, 440, duree_ms)

print("Émetteur Morse prêt. A = point, B = tiret. Ctrl+C pour arrêter.")

# Mémorise l'état précédent pour ne déclencher qu'une fois par appui
a_precedent = 1
b_precedent = 1

while True:
    a_actuel = A_BUTTON.value()
    b_actuel = B_BUTTON.value()

    # Détection de transition relâché -> appuyé sur A
    if a_actuel == 0 and a_precedent == 1:
        print(".", end="")
        beep(DUREE_POINT)

    # Détection de transition relâché -> appuyé sur B
    if b_actuel == 0 and b_precedent == 1:
        print("-", end="")
        beep(DUREE_TIRET)

    a_precedent = a_actuel
    b_precedent = b_actuel

    time.sleep_ms(20)
```

Fonctionnement du programme

Le programme s'organise en quatre parties :

- **Initialisation** : on déclare les trois composants en haut du programme. **Pin("SPEAKER", Pin.OUT_PP)** configure la broche du buzzer en sortie ; **Pin("A_BUTTON", Pin.IN)** configure les boutons en entrée (la résistance qui maintient la valeur à **1** au repos est câblée sur la carte, rien à activer côté code).
- **Fonction tone(pin, freq, duration_ms)** : c'est elle qui fait vibrer le buzzer. Elle calcule la demi-période correspondant à la fréquence demandée (à 440 Hz, une période complète dure 1/440 seconde ≈ 2272 µs, donc la demi-période est 1136 µs), puis elle alterne pin.on() / pin.off() pendant la durée totale. C'est ce qu'on appelle du **bit-banging** : le programme génère lui-même le signal, sans utiliser de module hardware dédié.
- **Détection de transition** : on ne veut pas qu'un appui maintenu déclenche une rafale de sons. La technique consiste à mémoriser l'état précédent du bouton et à ne déclencher que quand on passe de **relâché (1) à appuyé (0).**
- **Boucle principale** : elle scrute les deux boutons toutes les **20 ms** et déclenche le bip approprié.

CC BY SA

ENVOYER DES MESSAGES EN CODE MORSE

ÉTAPE 3 - AMÉLIORER

Ajouter un retour visuel pour les malentendants

1

Le buzzer émet un son, mais on peut aussi allumer une **LED RGB** en même temps : le message devient à la fois visible et audible. Les LED RGB de la STeaMi s'utilisent comme n'importe quelle broche, avec les noms parlants **LED_RED, LED_GREEN, LED_BLUE** (1 allume, 0 éteint). Modifiez la fonction **beep()** pour allumer la LED rouge au début du son et l'éteindre à la fin, ou changez de couleur en remplaçant simplement le nom de la broche.

Automatiser l'envoi de SOS

2

Plutôt que d'appuyer manuellement, vous pouvez faire envoyer SOS par la carte dès qu'on appuie sur le bouton A, pratique pour un signal de détresse maritime ou un test prolongé. Définissez deux fonctions utilitaires **point()** et **tiret()** qui jouent un bip court ou long puis attendent une unité de silence inter-signal, puis une fonction **envoyer_sos()** qui les enchaîne en respectant les espaces inter-lettres (3 unités) et inter-mots (7 unités). Branchez l'appel à **envoyer_sos()** sur la détection d'appui A dans la boucle principale.

Encoder un message dans le programme

3

Créez un **dictionnaire** qui associe chaque lettre à sa séquence Morse (**'A': '.-', 'B': '-...', etc.**), puis écrivez une fonction **envoyer_message(texte)** qui parcourt chaque caractère, traduit la séquence, et joue un bip court pour un point ou un bip long pour un tiret, en respectant les silences inter-signaux, inter-lettres et inter-mots. C'est un excellent exercice pour découvrir les structures de données (dictionnaires) et les boucles imbriquées : la boucle externe parcourt les lettres, la boucle interne parcourt les points et tirets de chaque lettre.

ALLER PLUS LOIN

Samuel Morse, l'un des pères fondateurs des télécommunications (Cité des Télécoms) - Découvrez l'histoire de l'inventeur et la genèse du télégraphe électrique sur le site du musée français des télécommunications. *https://www.cite-telecoms.com/accueil/musee-des-telecommunications/les-peres-fondateurs/morse/*

Les « messages personnels » de Radio Londres : pendant la Seconde Guerre mondiale, la BBC ouvrait ses émissions vers la France par les premières notes de la 5e symphonie de Beethoven, qui sont exactement la lettre V en code Morse (. . . -), pour Victoire. Suivaient des phrases codées destinées à la Résistance. *https://fr.wikipedia.org/wiki/Messages_personnels*

CC BY SA