RESSOURCES - PROGRAMMATION - CARTE STEAMI

CAPTER LA LUMIÈRE

PROGRAMMATION EN MICROPYTHON

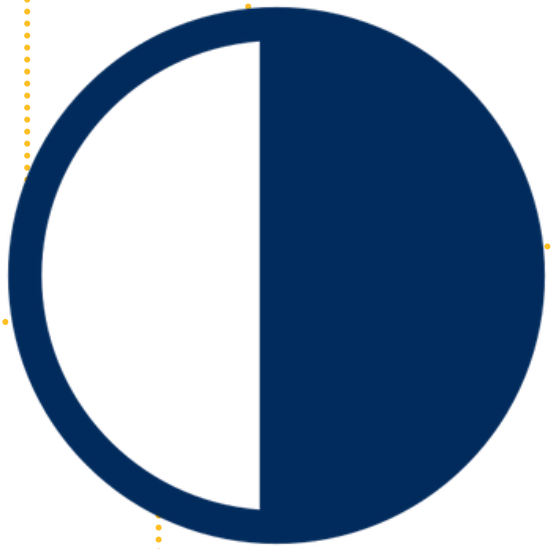
De quoi parle-t-on ?

Cette fiche explore la possibilité de **mesurer une grandeur physique avec un capteur**, et de représenter graphiquement la façon dont cette grandeur varie dans le temps.

Disponible sur

MicroPython

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...)



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

Débutant

STeaMi

OBJECTIFS D'APPRENTISSAGE

- Lire la valeur d'un capteur de lumière intégré à une carte microcontrôleur
- Comprendre la notion de capteur numérique communiquant via un bus I2C
- Afficher périodiquement une mesure dans la console et observer ses variations
- Utiliser une LED RGB comme indicateur visuel d'un seuil de mesure
- Découvrir les autres mesures accessibles avec le capteur APDS-9960 (couleurs RGB)

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.

CAPTER LA LUMIÈRE

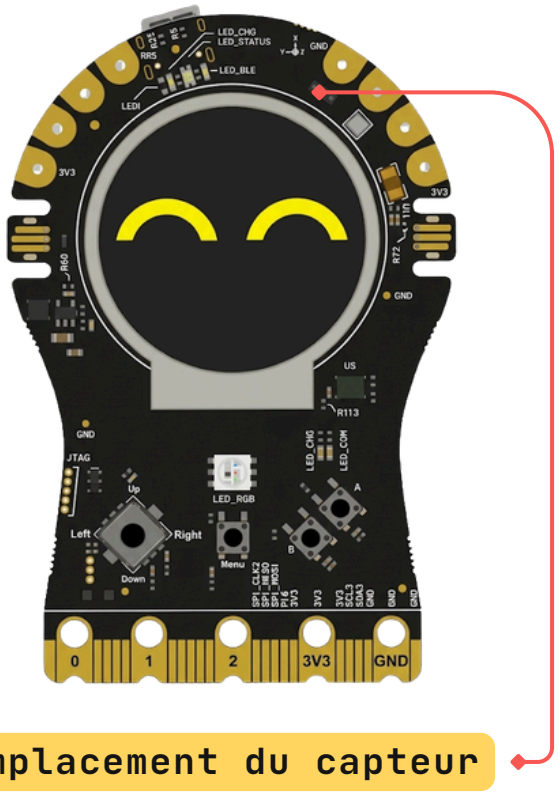
ÉTAPE 1 - CONSTRUIRE

La STeaMi intègre un capteur **APDS-9960** capable de **mesurer la lumière ambiante**, mais aussi les **composantes rouge, verte et bleue de cette lumière**, ainsi que **la proximité d'un objet et certains gestes**. Dans cet exemple de programme, nous nous concentrons sur la mesure de la lumière ambiante. L'avantage du capteur intégré est d'éviter une étape de breadboarding et de câblage. Tout passe par le **bus I2C interne** de la carte, ce qui permet de se concentrer sur la programmation et l'analyse des mesures. *https://fr.wikipedia.org/wiki/I2C*

Localiser le capteur de lumière

1

Le capteur **APDS-9960** est soudé sur la face avant de la STeaMi, près de l'écran OLED. Il n'a pas besoin d'être éclairé directement par une source : il mesure la **lumière ambiante** qui atteint sa fenêtre transparente.



Comprendre le capteur numérique I2C

Contrairement à une photorésistance qui fournit une tension variable lue par un convertisseur analogique-numérique, **l'APDS-9960 est un capteur numérique** : il fait lui-même la mesure et la conversion, puis envoie le résultat sous forme de nombre à la carte via une liaison appelée **I2C** (deux fils : SDA pour les données, SCL pour l'horloge).

Cela permet d'avoir un seul composant qui mesure à la fois la lumière, la couleur, la proximité et les gestes, sans câblage supplémentaire.

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 '>>>'.

CAPTER LA LUMIÈRE

ÉTAPE 1 - CONSTRUIRE

Vérifier que le capteur répond

Avant d'écrire le programme principal, on peut vérifier que **le capteur est bien détecté sur le bus I2C**. Pour ça, on utilise 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).

Programmer, exécuter, jouer

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

Test rapide. Lancez le programme depuis l’éditeur (typiquement bouton Run ▶️ ou F5). Les valeurs défilent dans la console MicroPython.

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.

Vous pouvez ensuite faire évoluer le code, ajuster le seuil, ajouter d'autres canaux du capteur ou utiliser la LED RGB de manière plus nuancée. Observer les variations. Une fois le programme lancé, plusieurs choses à essayer :

- **Couvrir le capteur avec la main** : la valeur affichée chute, la LED devient rouge.
- **Approcher la carte d'une fenêtre ou d'une lampe** : la valeur grimpe, la LED passe au vert.
- **Allumer la lampe d'un téléphone et la diriger vers le capteur** : on voit clairement le seuil franchi.

3

```
from machine import I2C
i2c = I2C(1)
[hex(a) for a in i2c.scan()]
['0x1e', '0x29', '0x39', '0x55', '0x5d',
'0x5f', '0x6b']
```

L'adresse 0x39 correspond à l'APDS-9960. Si elle apparaît, le capteur répond, on peut passer à la programmation.

4

```
Shell
>>> %Run -c $EDITOR_CONTENT

MPY: sync filesystems
MPY: soft reboot
Lecture du capteur de lumière. Ctrl+C pour arrêter.
Lumière : 322
Lumière : 324
Lumière : 323
Lumière : 322
Lumière : 323
Lumière : 322
Lumière : 326
Lumière : 327
Lumière : 325
Lumière : 104
Lumière : 58
Lumière : 55
```

Tracer un graphique (spécifique à Thonny)

Si vous utilisez Thonny, l'éditeur propose un traceur de variables : Affichage → Plotter (ou View → Plotter). Une fenêtre s'ouvre à côté de la console et trace en temps réel toutes les valeurs numériques affichées par print(). Cette fonctionnalité est pratique afin de visualiser comment la lumière varie dans le temps quand on bouge un objet devant le capteur.

CAPTER LA LUMIÈRE

ÉTAPE 2 - PROGRAMMER

```
# Testée avec firmware STeaMi 0.23.1
# Capteur de lumière, lecture périodique de la lumière ambiante
# avec retour visuel sur la LED RGB : lumière forte = LED verte et lumière
faible = LED rouge

from machine import I2C, Pin
from apds9960 import uAPDS9960
from time import sleep_ms

# Initialisation du capteur sur le bus I2C interne
i2c = I2C(1)
sensor = uAPDS9960(i2c)

# LED RGB de la STeaMi
led_r = Pin('LED_RED', Pin.OUT)
led_g = Pin('LED_GREEN', Pin.OUT)
led_b = Pin('LED_BLUE', Pin.OUT)

# Seuil au-dessus duquel on considère qu'il fait clair.
# À ajuster selon les conditions d'utilisation.
SEUIL_CLAIR = 100

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)


print("Lecture du capteur de lumière. Ctrl+C pour arrêter.")

while True:
    # Lecture de la lumière ambiante (canal "clear" du capteur, 16 bits)
    lumiere = sensor.ambient_light()
    print("Lumière :", lumiere)

    # Indicateur visuel
    if lumiere > SEUIL_CLAIR:
        set_rgb(0, 1, 0)   # vert : il fait clair
    else:
        set_rgb(1, 0, 0)   # rouge : il fait sombre

    sleep_ms(500)
```

ÉTAPE 2 - PROGRAMMER

Fonctionnement du programme

Le programme tient en quatre éléments :

- **from apds9960 import uAPDS9960** importe le pilote du capteur fourni avec MicroPython sur la STeaMi. La classe **uAPDS9960** est une version optimisée pour MicroPython.
- **i2c = I2C(1)** ouvre le bus I2C interne de la carte (celui sur lequel se trouvent tous les capteurs intégrés).
- **sensor.ambient_light()** retourne un nombre entre **0** et **65535** qui représente la quantité de lumière atteignant le capteur. Plus la valeur est grande, plus il fait clair. Le capteur est automatiquement activé à la première lecture.
- La boucle **while True** répète la mesure toutes les **500 millisecondes**, allume la **LED** en vert ou en rouge selon la valeur, et affiche le résultat dans la console.

Plage de valeurs

Le capteur retourne une valeur sur **16 bits**, donc **entre 0 (obscurité totale) et 65535 (lumière très forte)**. En conditions normales d'éclairage de salle, on observe typiquement des valeurs de quelques dizaines à quelques milliers, il faudra peut-être ajuster **SEUIL_CLAIR** selon l'environnement. Une lampe pointée directement sur le capteur peut faire saturer la valeur à **65535**.

CAPTER LA LUMIÈRE

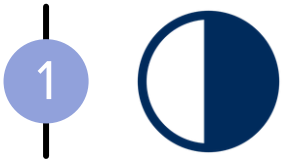
ÉTAPE 3 - AMÉLIORER

Ajuster le seuil et calibrer

La valeur de **SEUIL_CLAIR = 100** est arbitraire. Pour adapter le détecteur à un environnement précis :

- Lancez le programme dans la condition « **sombre** » visée (capteur recouvert par exemple) et notez la valeur lue.
- Faites de même dans la condition « **claire** » (lumière ambiante normale).
- Choisissez comme seuil la **moyenne des deux valeurs**.

Une amélioration plus avancée consiste à transformer la valeur brute en pourcentage par rapport à un minimum et un maximum mesurés, ce qui rend la mesure plus parlante.

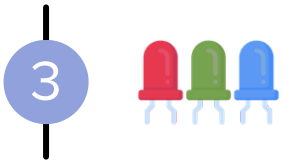


Lire les composantes RGB de la lumière

L'APDS-9960 ne mesure pas seulement la quantité de lumière : il en mesure aussi les **trois composantes rouge, verte et bleue**. C'est une différence importante : alors que **ambient_light()** dit juste « il y a beaucoup ou peu de lumière », les canaux **R, V, B** disent **quelle couleur est dominante**. Le capteur peut donc distinguer la lumière d'une LED rouge de celle d'une LED verte, alors que **ambient_light()** donnerait à peu près la même valeur dans les deux cas. Pour expérimenter, éclairez le capteur avec différentes sources comme la lampe d'un téléphone, l'écran allumé sur un fond rouge ou vert, une LED de couleur, et observez comment les trois canaux varient indépendamment.

Reproduire la couleur sur la LED RGB

En combinant les deux idées précédentes, vous pouvez faire en sorte que la LED RGB de la carte reproduise approximativement la couleur dominante de la lumière reçue : lisez les trois canaux R, V, B, comparez-les entre eux et allumez la composante la plus forte.



ALLER PLUS LOIN

Vision des couleurs : cônes et bâtonnets (ERCO) - Découvrez comment l'œil humain perçoit la lumière grâce à trois types de cônes sensibles au rouge, au vert et au bleu, comme le capteur APDS-9960. *https://www.erco.com/fr/conception-lumiere/savoir-lumiere/l-il-humain/cones-7525/*

Photographier la Voie Lactée - Avec des poses de plusieurs secondes ou minutes, un capteur photo accumule la lumière la plus faible. C'est comme cela que les astronomes amateurs photographient des galaxies invisibles à l'œil nu. *https://www.lemonde.fr/blog/autourduciel/2020/05/22/photographier-la-voie-lactee-en-pause-longue/*