PROGRAMMATION - FICHE D'ACTIVITÉ MICROPYTHON

MAGNETICS

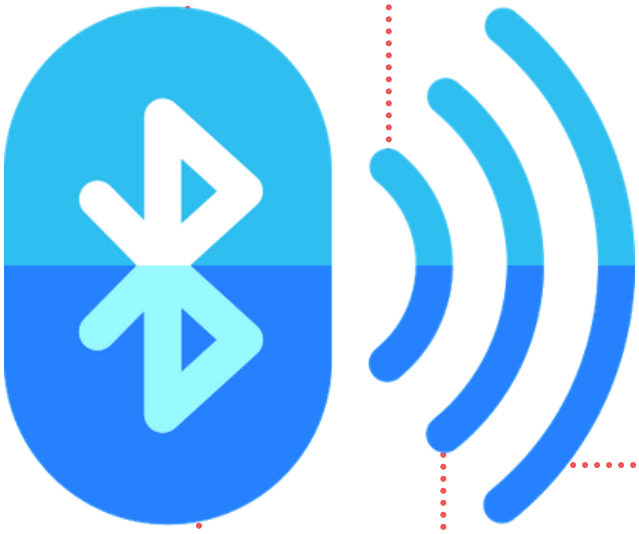
Créer des projets multicartes

De quoi parle-t-on ?

Dans cette activité, nous allons programmer plusieurs cartes électroniques et échanger des données de capteurs à l'aide de l'extension Magnetics, qui permet une communication maillée sans fil.

MicroPython

Disponible sur



Durée

50 minutes

Coopération

Let's STEAM

Matériel

- 4 cartes programmables "***STM32 IoT Node Board***"
- 1 câble USB Micro-B
- 1 écran OLED Monochrome 1.3" 128x64 OLED de Adafruit
- 1 câble QT pour connecter l'écran à la carte

Niveau de difficulté

Avancé

OBJECTIFS D'APPRENTISSAGE

- Utiliser l'extension magnetics pour constuire des projets multicartes
- Appréhender le fonctionnement des modules bluetooth BLE Mesh

Cette fiche d'activité complète le travail réalisé au sein du projet Let's STEAM financé avec le soutien de la Commission européenne par le biais du programme de partenariat stratégique Erasmus +. Son contenu n'engage que son auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qu'elle contient. Ce travail est sous licence Creative Commons Attribution-ShareAlike 4.0 International License.

Cofinancé par le programme Erasmus+ de l'Union européenne

MAGNETICS - CRÉER DES PROJETS MULTICARTES

Cette fiche d'activité propose de créer des projets plus complexes en utilisant plusieurs cartes électroniques non connectées entre elles. Une fois les capteurs maîtrisés, nous pouvons en effet mettre en place des expériences nécessitant l'utilisation de plusieurs cartes. Afin de réaliser la collecte des données, il faut pouvoir faire communiquer les cartes entre elles par les airs. Dans cette activité nous allons programmer plusieurs cartes électroniques et échanger des données de capteurs à l'aide de l'extension **Magnetics** que permet de mettre en œuvre une communication sans fil maillée. Le projet magnetics prend la forme d'une brique technique logicielle implantée directement dans MakeCode. Ce développement est basé sur l'utilisation de la technologie de réseau maillé **Bluetooth Low Energy Mesh** (BLE Mesh) compatible avec toutes les cartes programmables disposant d'un module **Bluetooth Low Energy**.

Ressources : https://www.magnetics.edu-up.fr/
https://blog.rtone.fr/bluetooth-mesh
https://fr.wikipedia.org/wiki/Bluetooth_%C3%A0_basse_consommation

ÉTAPE 1 - CONSTRUIRE

Pour réaliser cette activité nous avons besoin de **quatre cartes STM32 IoT Nodes**. Trois d'entre elles seront **émettrices** de données de capteurs (température, humidité, pression), et la dernière sera **collectrice** des données qu'elle affichera sur un écran OLED. Mis à part l'écran de la dernière, il n'y a pas de câblage car nous utiliserons uniquement les capteurs internes. Nous allons donc vous donner la marche à suivre pour câbler et programmer en premier lieu la carte collectrice puis dans un second temps, programmer individuellement chaque carte émettrice afin de pouvoir construire votre projet.

Activité 1 - Préparer, câbler et programmer la carte collectrice

Câbler l'écran OLED

Nous devons en premier lieu câbler l'écran OLED directement à la carte collectrice. Il y a deux façons de câbler l'écran **OLED SSD1306** à une carte, soit avec une connexion **I2C** ou **SPI**. Pour notre écran, nous utilisons la connexion **I2C** via le câble **QWIIC/STEMMA** avec la convention suivante :

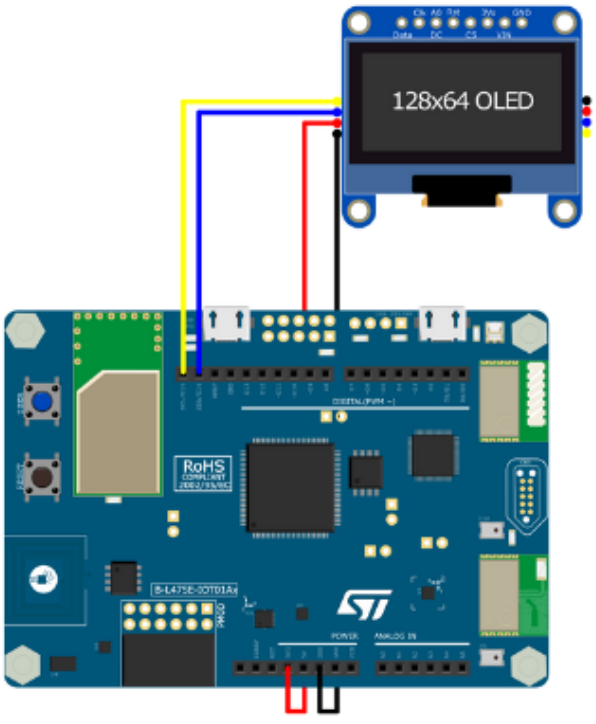
- **Noir** pour **GND**
- **Rouge** pour **V+ (3V3)**
- **Bleu** pour **SDA (D14)**
- **Jaune** pour **SCL (D15)**

Connecter la carte à l'ordinateur

Avec votre câble USB, connectez la carte collectrice à votre ordinateur en utilisant le connecteur micro-USB. Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur.

1

2



Câblage de l'écran OLED sur la carte collectrice

MAGNETICS - CRÉER DES PROJETS MULTICARTES

ÉTAPE 1 - CONSTRUIRE

Ouvrir l'éditeur Micropython

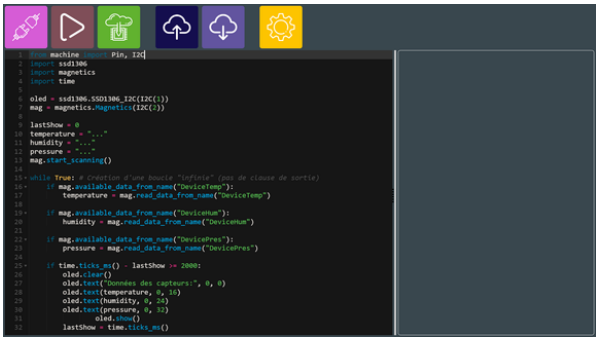
3

Allez dans l'éditeur Micropython (**https://python.lets-steam.eu/**). Depuis cet éditeur, vous pourrez directement copier/coller votre code.

Programmer la carte

4

Dans l'éditeur, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Cliquez sur le bouton "**flash or download**" et attendez que la carte finisse de clignoter.



Activité 2 - Programmer chaque carte émettrice individuellement

Une fois la carte collectrice câblée avec l'écran OLED et programmée, nous pouvons préparer les trois cartes émettrices en suivant les mêmes étapes de programmation que lors de l'étape 1. Pour chaque carte, il faudra donc effectuer les tâches suivantes :

Connecter la carte à l'ordinateur

5

Avec votre câble USB, connectez la carte émettrice que vous souhaitez programmer à votre ordinateur en utilisant le connecteur **micro-USB**. Si tout se passe bien, vous devriez voir apparaître sur votre ordinateur un nouveau lecteur. Ce lecteur est utilisé pour programmer la carte en copiant simplement un fichier binaire.

Ouvrir l'éditeur Micropython

6

Allez dans l'éditeur Micropython (https://python.lets-steam.eu/). Depuis cet éditeur, vous pourrez directement copier/coller votre code.

Programmer la carte

7

Dans l'éditeur, copiez/collez le code disponible dans la section "**Programmer**" ci-dessous. Cliquez sur le bouton "**flash or download**" et attendez que la carte finisse de clignoter.

Activité 3 - Exécuter, modifier, jouer

Une fois votre carte collectrice reliée à votre écran, et vos quatre cartes programmées, votre programme est prêt à être utilisé. Votre programme s'exécutera automatiquement chaque fois que vous les mettrez sous tension ou que vous réinitialiserez votre carte (appuyez sur le bouton intitulé RESET). Essayez de comprendre le fonctionnement de votre code et commencez à le modifier en créant vos propres projets.

ETAPE 2 - PROGRAMMER

Code de la carte collectrice

```
from machine import Pin, I2C
import ssd1306
import magnetics
import time

oled = ssd1306.SSD1306_I2C(I2C(1))
mag = magnetics.Magnetics(I2C(2))

lastShow = 0
temperature = "..."
humidity = "..."
pressure = "..."
mag.start_scanning()

while True: # Création d'une boucle "infinie" (pas de clause de sortie)
    if mag.available_data_from_name("DeviceTemp"):
        temperature = mag.read_data_from_name("DeviceTemp")

    if mag.available_data_from_name("DeviceHum"):
        humidity = mag.read_data_from_name("DeviceHum")

    if mag.available_data_from_name("DevicePres"):
        pressure = mag.read_data_from_name("DevicePres")

    if time.ticks_ms() - lastShow >= 2000:
        oled.clear()
        oled.text("Données des capteurs:", 0, 0)
        oled.text(temperature, 0, 16)
        oled.text(humidity, 0, 24)
        oled.text(pressure, 0, 32)
        oled.show()
        lastShow = time.ticks_ms()
```

ETAPE 2 - PROGRAMMER

Code de la carte émettrice de la température

```
from machine import Pin, I2C
import magnetics
import time

mag = magnetics.Magnetics(I2C(2))
mag.setLocalName("DeviceTemp")
mag.startEmitting()

while True: # Création d'une boucle "infinie" (pas de clause de sortie)
   magnetics.setAdvertisingKeyValueData("Temp (°C)", input.temperature(TemperatureUnit.Celsius))
pause(500)
```

Code de la carte émettrice de l'humidité

```
from machine import Pin, I2C
import magnetics
import time

mag = magnetics.Magnetics(I2C(2))
mag.setLocalName("DeviceHum")
mag.startEmitting()

while True: # Création d'une boucle "infinie" (pas de clause de sortie)
   magnetics.setAdvertisingKeyValueData("Hum (%)", input.temperature(TemperatureUnit.Celsius))
pause(500)
```

Code de la carte émettrice de la pression

```
from machine import Pin, I2C
import magnetics
import time

mag = magnetics.Magnetics(I2C(2))
mag.setLocalName("DevicePres")
mag.startEmitting()

while True: # Création d'une boucle "infinie" (pas de clause de sortie)
   magnetics.setAdvertisingKeyValueData("Pres (hPa)", input.temperature(TemperatureUnit.Celsius))
pause(500)
```

ETAPE 2 - PROGRAMMER

Comment cela fonctionne-t-il ? Initialiser la collecte de données :

Cela fait beaucoup de code, mais rien de très complexe, puisque le code des **cartes émettrices** est quasiment identique (il n'y a que le capteur qui change) et reste simple à comprendre.

Pour commencer, on donne un nom à notre carte à l'aide de la fonction setLocalName, ce qui permettra au collecteur de reconnaître la donnée envoyée. Pour que la carte puisse émettre des données, il faut lui préciser son rôle, c'est ce que fait la fonction startEmitting. Enfin, via la fonction setAdvertisingKeyValueData, nous envoyons la donnée du capteur toutes les 500 millisecondes (c'est-à-dire deux fois par seconde).

Enfin, intéressons-nous au code du collecteur. Au début, nous déclarons 4 variables:

- temperature, humidity et pressure qui contiendrons les données émises par les émetteurs (respectivement pour la température, l'humidité et le pression atmosphérique)
- lastShow qui permet de savoir à quand remonte le dernier affichage des données (en millisecondes).

Une fois les variables initialisées, nous faisons appel à startScanning, pour rechercher les cartes émettrices qui sont à proximité.

Le reste du code se trouve dans la boucle principale ce qui implique, que nous allons répéter indéfiniment le code suivant. Les trois premiers if(...) permettent de savoir si nous avons reçu des données de la part des émetteurs (grâce à la fonction availableDataFromName), si tel est le cas, alors on enregistre cette donnée dans la variable associée (en utilisant readDataFromName).

Maintenant que nous avons les données, il faut les afficher sur notre écran OLED, c'est le rôle de la dernière condition if, qui nous permet de mettre à jour notre écran avec les nouvelles données, seulement si cela fait plus de deux seconds qu'il n'a pas été rafraîchi.

ETAPE 3 - AMÉLIORER

1

Cette activité ne fait qu'afficher les informations sur un écran, mais il est possible **d'utiliser le datalogger, pour enregistrer nos données et les analyser**.

2

Utiliser une boîte étanche et **faire une station météo**, la connexion étant sans fil, on pourrait avoir un écran en intérieur nous affichant **les données environnementales de l'extérieur.**

ALLER PLUS LOIN

Bluetooth - Découvrir plus d'informations sur cette norme de télécommunication si présente dans notre vie quotidienne
https://fr.wikipedia.org/wiki/Bluetooth

CircuitPython BLE Libraries on Any Computer - Utilisez le code BLE de CircuitPython sur les ordinateurs de bureau, les ordinateurs portables et les Raspberry Pi grâce aux bibliothèques Adafruit
https://learn.adafruit.com/circuitpython-ble-libraries-on-any-computer

Bouton de volume BLE avec CircuitPython - Transformez votre Bluefruit de Circuit Playground en bouton de volume BLE sans fil
https://learn.adafruit.com/bluetooth-le-hid-volume-knob-with-circuitpython