MicroPython

CARTES Pyboard --->V1.1 PYCOM ---> Wipy, Lopy, SiPy, etc MikroElektronika ---> Flip & Click

Lycee Pierre EMILE MARTIN – 1 Av de Gionne – 18000 Bourges

**Table des matières**

[1 La carte Pyboard v1.1 4](#_Toc513476693)

[1.1 Caractéristiques techniques 4](#_Toc513476694)

[1.2 Première mise en service 4](#_Toc513476695)

[1.2.1 Connexion à un port USB 4](#_Toc513476696)

[1.2.2 Lecteur PYBFLASH 4](#_Toc513476697)

[1.2.3 Le fichier « Readme.txt » 5](#_Toc513476698)

[1.3 Un premier programme 5](#_Toc513476699)

[1.4 L’afficheur LCD « LCD32MKv1.0 » associé à la carte Pyboard v1.1 5](#_Toc513476700)

[2 Les modules Pycom pour l'internet des objets (IoT) 6](#_Toc513476701)

[2.1 Généralités 6](#_Toc513476702)

[2.2 Configuration du matériel (1) 6](#_Toc513476703)

[2.2.1 Identification 6](#_Toc513476704)

[2.3 Configuration de l'ordinateur (2) 6](#_Toc513476705)

[2.3.1 Drivers 6](#_Toc513476706)

[2.3.2 Mise à jour du firmware du module 7](#_Toc513476707)

[2.3.3 Connexion au réseau Wifi de la carte 7](#_Toc513476708)

[2.3.4 L'environnement de développement : Pymakr (3) 7](#_Toc513476709)

[2.3.5 Première connexion à la carte avec VSCode 7](#_Toc513476710)

[2.4 Programmer les modules Pycom en MicroPython (3) 8](#_Toc513476711)

[2.4.1 Introduction à MicroPython 8](#_Toc513476712)

[2.4.2 Exemples écrits en MicroPython 8](#_Toc513476713)

[2.4.3 Premier projet avec Pymakr 9](#_Toc513476714)

[2.4.4 REPL (Read Evaluate Print Loop) 11](#_Toc513476715)

[2.4.5 FTP 11](#_Toc513476716)

[2.4.6 Démarrage sécurisée (Safe Boot) 12](#_Toc513476717)

[2.4.7 Programmer les cartes avec l'IDE ATOM et l'extension Pymakr 13](#_Toc513476718)

[Annexe : Créer un fichier boot.py de connexion au Wifi 14](#_Toc513476719)

[2.5 Les modules Pycom 15](#_Toc513476720)

[2.5.1 Le module "WiPy 3.0" de Pycom 15](#_Toc513476721)

[2.5.2 Le module "LoPy" de Pycom 16](#_Toc513476722)

[2.5.3 Le module "SiPy" de Pycom 16](#_Toc513476723)

[3 La carte "Flip & Click" de MikroElektronika 17](#_Toc513476724)

[3.1 Caractéristiques 17](#_Toc513476725)

[3.2 IDE : Zerinth Studio 17](#_Toc513476726)

[4 Annexes 18](#_Toc513476727)

[4.1 Exemples avec les modules Pycom 18](#_Toc513476728)

**Dernière mise à jour le 9/5/2018**

# La carte Pyboard v1.1

## Caractéristiques techniques

- alimentation à prévoir: 5 Vcc via port micro-USB ou 3,6 à 10 Vcc sur pastilles à souder  
​- microcontrôleur: STM32F405RG  
- microprocesseur: Cortex M4 168 MHz  
- 1024 kB de mémoire flash  
- 192 kB de mémoire ram  
- connecteur micro-USB (alimentation et communication série)  
- lecteur de carte micro-SD (carte SD non incluse)  
- accéléromètre 3 axes (MMA7660)  
- module horloge temps réel  
- 24 broches d'entrées/sorties  
- 4 LED (rouge, verte, jaune, bleue)  
- 2 boutons poussoirs  
- régulateur de tension 3,3 Vcc/250 mA  
- compatible Windows, Mac et Linux



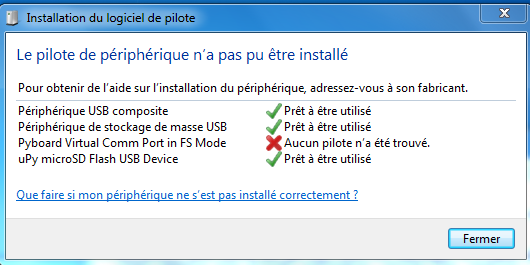
## Première mise en service

### Connexion à un port USB

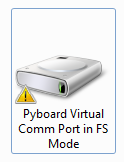


- Sous W10

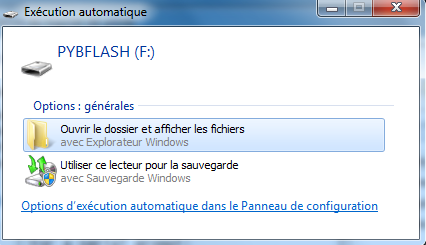
- Sous W7

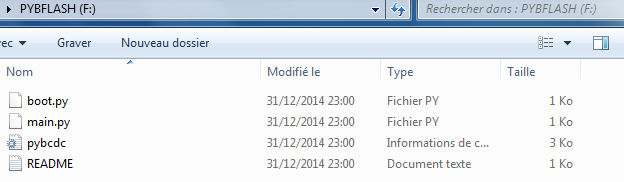


### Lecteur PYBFLASH



Installation terminée => Port virtuel non installé, ouverture du lecteur PYBFLASH





### Le fichier « Readme.txt »

This is a MicroPython board

**You can get started right away by writing your Python code in 'main.py'.**

For a serial prompt:

**- Windows: you need to go to 'Device manager', right click on the unknown device,**

**then update the driver software, using the 'pybcdc.inf' file found on this drive.**

**Then use a terminal program like Hyperterminal or putty.**

- Mac OS X: use the command: screen /dev/tty.usbmodem\*

- Linux: use the command: screen /dev/ttyACM0

Please visit http://micropython.org/help/ for further help.

## Un premier programme

1. **Ouvrir** le fichier main.py situé dans le lecteur **PYBFLASH** avec un éditeur de texte (ex notepad) ou le lecteur PYBFLASH dans Visual Studio Code
2. **Modifier** le code

Exemple : Commande de la LED rouge

*# main.py -- put your code here!*

**import** pyb

pyb**.**LED(4)**.**on()

1. **Sauvegarder** (Ctrl-S) => la LED rouge s’allume pendant le transfert
2. **Attendre** que la LED rouge s’éteigne et faire un **reset** de la carte (BP rst)

## L’afficheur LCD « LCD32MKv1.0 » associé à la carte Pyboard v1.1

* Caractéristiques : Afficheur graphique 128x32pixels
* Classe : LCD (<http://docs.micropython.org/en/latest/pyboard/library/pyb.LCD.html>)

# Les modules Pycom pour l'internet des objets (IoT)



## Généralités

*Source :* [*https://docs.pycom.io/chapter/gettingstarted/*](https://docs.pycom.io/chapter/gettingstarted/)

Les modules Pycom actuels (2017) sont organisés autour d'un **ESP32**, ils offrent au minimum :

- 512 Kb available for the user as internal storage, (external SD card support available)

- Up to 96 Kb of RAM available for python code.

- Hardware floating point unit

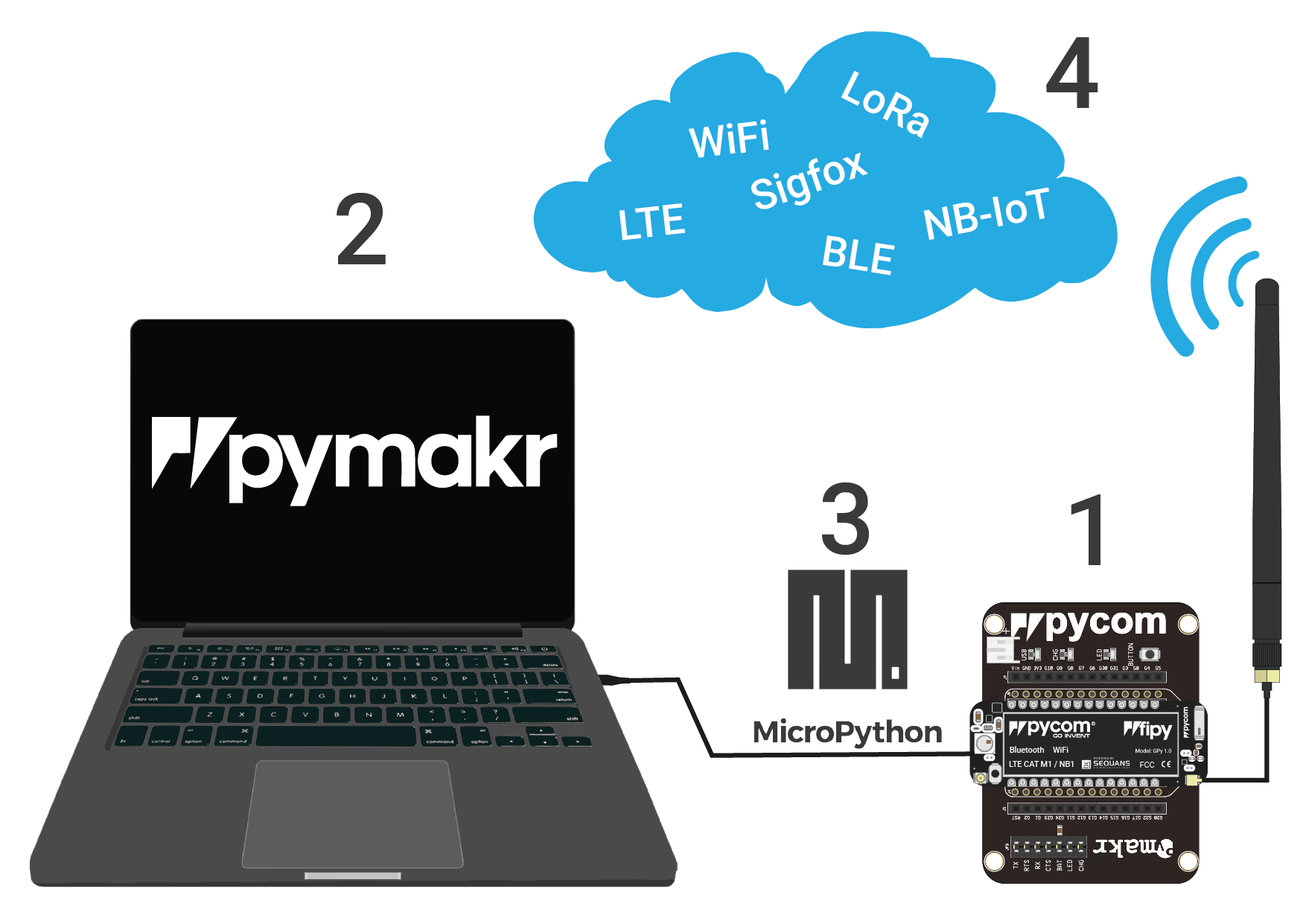


Figure 1 : Ecosystème Pycom

- Up to 24 GPIO [Pins](https://docs.pycom.io/pycom_esp32/library/machine.Pin.html#machine.Pin)

- 2x [UARTs](https://docs.pycom.io/pycom_esp32/library/machine.UART.html" \l "machine.UART" \o "machine.UART) - [Bluetooth](https://docs.pycom.io/pycom_esp32/library/network.Bluetooth.html#network.Bluetooth)

- [SPIs](https://docs.pycom.io/pycom_esp32/library/machine.SPI.html#machine.SPI) - [LoRa](https://docs.pycom.io/pycom_esp32/library/network.LoRa.html#network.LoRa) (only available in the LoPy)

- [Timers](https://docs.pycom.io/pycom_esp32/library/machine.Timer.html#machine.Timer) [hashlib](https://docs.pycom.io/pycom_esp32/library/uhashlib.html#module-uhashlib) MD5, SHA1, SHA256, SHA384 and SHA512 hash algorithms

- [RTC](https://docs.pycom.io/pycom_esp32/library/machine.RTC.html#machine.RTC) - [AES encryption](https://docs.pycom.io/pycom_esp32/library/ucrypto.AES.html#ucrypto.AES)

- [PWM](https://docs.pycom.io/pycom_esp32/library/machine.PWM.html#machine.PWM) - [SSL/TLS support](https://docs.pycom.io/pycom_esp32/library/ussl.html#module-ussl)

- [ADC](https://docs.pycom.io/pycom_esp32/library/machine.ADC.html#machine.ADC)

- [DAC](https://docs.pycom.io/pycom_esp32/library/machine.DAC.html#machine.DAC)

- [I2C](https://docs.pycom.io/pycom_esp32/library/machine.I2C.html#machine.I2C)

- [SD](https://docs.pycom.io/pycom_esp32/library/machine.SD.html#machine.SD)

La description des modules Pycom est accessible ici : <https://docs.pycom.io/chapter/products.html>

## Configuration du matériel (1)

L'installation d'un module Pycom sur une carte d'extension, une carte Pysense, etc. facilite le développement de l'application. Dans l'exemple ci-dessous, un module WiPy est inséré sur une carte d'extension.

### Identification

*Source :* [*https://docs.pycom.io/chapter/gettingstarted/connection/wipy.html*](https://docs.pycom.io/chapter/gettingstarted/connection/wipy.html)

* Recherchez **le bouton de réinitialisation** sur le module (situé dans un coin de la carte, à côté de la **LED**).
* Localisez le connecteur USB sur la carte d'extension.
* Insérez le module WiPy sur la carte d'extension avec le bouton de réinitialisation pointant vers le connecteur USB.

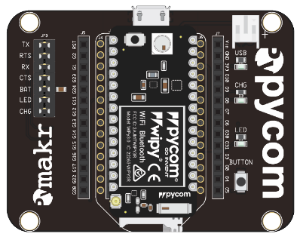
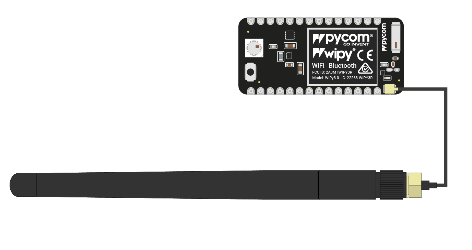


Figure 2 : carte d'extension V2.1



* **Antennes WiFi / Bluetooth (en option)**

Tous les modules Pycom, y compris le WiPy, sont équipés d'une antenne WiFi intégrée ainsi que d'un connecteur U.FL pour une antenne externe. L'antenne externe est facultative et n'est nécessaire que si vous avez besoin de meilleures performances ou si vous installez le WiPy de manière à bloquer le signal WiFi. La commutation entre les antennes se fait via un logiciel, les instructions sont disponibles ici. <https://docs.pycom.io/chapter/firmwareapi/pycom/network/wlan.html>

## Configuration de l'ordinateur (2)

### Drivers

Les modules sont automatiquement reconnus sous Windows 8/10/+. Voir le lien ci-dessous pour Windows 7.

[*https://docs.pycom.io/chapter/gettingstarted/installation/drivers.html*](https://docs.pycom.io/chapter/gettingstarted/installation/drivers.html)

### Mise à jour du firmware du module

*Sources :* [*https://docs.pycom.io/chapter/gettingstarted/hardwaresetup.html*](https://docs.pycom.io/chapter/gettingstarted/hardwaresetup.html)

*"Nous vous recommandons vivement de mettre à jour votre microprogramme vers la dernière version, car nous améliorons constamment et ajoutons de nouvelles fonctionnalités aux périphériques." Source Pycom*.

**AVERTISSEMENT**

Assurez-vous que le cavalier **TX** est présent sur la carte d'extension. Sans ce cavalier, la mise à jour échouera.

Suivre la procédure accessible à partir du lien ci-dessous :

[*https://docs.pycom.io/chapter/gettingstarted/installation/firmwaretool.html*](https://docs.pycom.io/chapter/gettingstarted/installation/firmwaretool.html)

### Connexion au réseau Wifi de la carte

*"Une fois que le microprogramme a été mis à jour, faire un RESET (appuyez sur le bouton à côté de la LED) pour que la carte se positionne en mode diffusion WiFi (****point d'accès****). Cela vous permet de vous connecter au* ***serveur interne du périphérique*** *pour télécharger des fichiers / scripts et modifier les paramètres de configuration.*



*Ouvrez les paramètres réseau sur votre ordinateur. Vous verrez apparaître un* ***SSID*** *avec un nom semblable à* ***wipy-wlan-xxxx****,* ***lopy-wlan-xxxx****, etc. Connectez-vous au réseau du module et entrez le mot de passe* ***www.pycom.io*** *. Une fois que vous êtes connecté au réseau de l'appareil, vous pouvez ouvrir le programme de démarrage dans MicroPython ou télécharger des scripts sur votre appareil !" source Pycom.*

**Note**

*Lorsqu'il est connecté au réseau de la carte, à moins de disposer de plusieurs cartes réseau, vous ne pourrez plus accéder à Internet depuis votre ordinateur !*

### L'environnement de développement : Pymakr (3)

Pycom a développé un plug-in appelé Pymakr pour deux éditeurs de texte.

Suivre la procédure accessible à partir du lien ci-dessous pour installer un de ces plug-ins :

<https://docs.pycom.io/chapter/gettingstarted/installation/pymakr.html>

### Première connexion à la carte avec VSCode

* **Par un port série USB**

Après l'installation du plug-in Pymakr, vous devez prendre quelques secondes pour le configurer lors de la première utilisation.

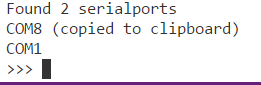
1. Connectez le module Pycom à l'ordinateur avec un câble USB. Si vous utilisez une carte d'extension et que vous venez d'effectuer une mise à niveau du microprogramme, veillez à retirer le câble entre GND et G23 et réinitialisez votre appareil en appuyant sur le bouton RESET.
2. Ouvrez Visual Studio Code et assurez-vous que le plug-in Pymakr est correctement installé.



1. Cliquez sur ***All Commands***



1. Dans la liste, sélectionnez **Pymakr > Extra > List Serial Ports**
2. Cela liste les ports série disponibles. Si Pymakr est capable de détecter automatiquement ce qui doit être utilisé, il sera copié dans votre presse-papier. Si ce n'est pas le cas, copiez manuellement le bon port série.



1. Cliquez à nouveau sur ***All Commands***, puis sur **Pymakr > Global Settings**. Cela va ouvrir un fichier JSON. Collez l'adresse du port série que vous avez copiée précédemment dans le champ "adresse" et enregistrez le fichier.

{

"address": "COM8",

"username": "micro",

"password": "python",

"sync\_folder": "",

"open\_on\_start": true,

"sync\_file\_types": "py,txt,log,json,xml",

"ctrl\_c\_on\_connect": false

}

1. Pour finir, fermez le fichier JSON, cliquez sur ***All Commands***, puis sur Pymakr > Connect pour connecter votre appareil. La console Pymakr devrait montrer trois flèches '>>>', indiquant que vous êtes connecté !

* **Avec Telnet (**[**PuTTY**](https://www.putty.org/) **sous Windows)**

Après l'installation du plug-in Pymakr, un appareil peut être connecté via l'interface Telnet.

- Assurez-vous que le périphérique Pycom est activé

- Connectez l'ordinateur hôte au point d'accès WiFi nommé d'après votre carte (le SSID sera comme suit, par exemple **lopy-wlan-xxxx, wipy-wlan-xxxx**, etc.). Le mot de passe est www.pycom.io.

- Suivez les étapes décrites ci-dessus dans la section "Connexion via USB série", mais entrez **192.168.4.1** comme adresse.

- Le nom d'utilisateur et le mot de passe par défaut sont respectivement **micro** et **python**.

- Enfin fermez le fichier JSON, cliquez sur All commands, puis Pymakr> Connect, Pymakr se connectera maintenant via Telnet.

## Programmer les modules Pycom en MicroPython (3)

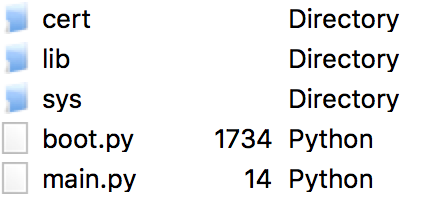


### Introduction à MicroPython

Les modules fonctionnent avec MicroPython; une implémentation de **Python 3.5** optimisée pour fonctionner sur des microcontrôleurs.

Lors du démarrage, deux fichiers sont exécutés automatiquement : d'abord **boot.py** puis **main.py**. Ils sont placés dans le dossier **/flash** de la carte. Tous les autres fichiers ou bibliothèques peuvent également être placés ici, et peuvent être inclus ou utilisés depuis boot.py ou main.py.

La structure des dossiers dans /flash ressemble à l'image ci-dessous. Les fichiers peuvent être gérés soit en utilisant **FTP** ou en utilisant le plug-in **Pymakr**.

****

### Exemples écrits en MicroPython

* **Affectation de variable**

variable = "Hello World"

**print**(variable)

* **Expressions conditionnelles**

temperature = 15

target = 10

**if** temperature > target:

print("Too High!")

**elif** temperature < target:

print("Too Low!")

**else:**

print("Just right!")

* **Boucles (for et while)**

Les boucles sont une autre caractéristique importante de tout langage de programmation. Cela vous permet de faire défiler votre code et de répéter les fonctions / affectations / etc.

Les boucles **for** permettent de contrôler le nombre de fois qu'un bloc de code s'exécute dans une plage

x = 0

**for** y **in range**(0,9):

x += 1

**print**(x)

Les boucles **while** sont similaires aux boucles For, elles vous permettent cependant d'exécuter une boucle jusqu'à ce qu'une condition soit vrai / faux. Dans le cas ci-dessous, la boucle se répète si x est inférieur à 9.

x = 0

**while** x < 9:

x += 1

**print**(x)

* **Les fonctions**

Les fonctions sont des blocs de code référencés par leur nom. Les données peuvent y être passées pour être exploitées (c'est-à-dire, les paramètres) et peuvent éventuellement retourner des données (la valeur de retour). Toutes les données transmises à une fonction sont explicitement transmises.

La fonction ci-dessous prend deux nombres et les ajoute ensemble, produisant le résultat.

**def** add(number1, number2):

**return** number1 + number2

add(1,2) **#** expect a result of 3

La fonction suivante prend un nom en entrée et renvoie une chaîne contenant une phrase de bienvenue.

**def** welcome(name):

welcome\_phrase = "Hello, " + name + "!"

print(welcome\_phrase)

welcome("Alex") **#** expect "Hello, Alex!"

* **Structures de données**
* **Listes**

Structure de données contenant une collection ordonnée (séquence) d'éléments.

networks = ['lora', 'sigfox', 'wifi', 'bluetooth', 'lte-m']

print(networks[2]) # expect 'wifi'

* **Dictionnaires**

Un dictionnaire est comme un carnet d'adresses où vous pouvez trouver l'adresse ou les coordonnées d'une personne en ne connaissant que son nom, c'est-à-dire que les **clés** (noms) sont associées aux **valeurs** (détails).

address\_book = {'Alex':'2604 Crosswind Drive','Joe':'1301 Hillview Drive','Chris':'3236 Goldleaf Lane'}

print(address\_book['Alex']) # expect '2604 Crosswind Drive'

* **Tuple**

Similaire aux listes, mais immuable, c'est-à-dire que vous ne pouvez pas modifier les tuples après l'instanciation.

pycom\_devices = ('wipy', 'lopy', 'sipy', 'gpy', 'fipy')

print(pycom\_devices[0]) # expect 'wipy'

### Premier projet avec Pymakr

* **Créer un projet dans Pymakr**

1. Créez un nouveau répertoire vide sur votre ordinateur. Pour cet exemple, nous allons en créer un appelé **RGB-Blink**.



1. Ensuite, **Visual Studio Code** selon ce que vous avez configuré. (Voir le chapitre Pymakr dans le document sur Visual Studio Code)
2. Une fois l'éditeur de texte chargé, cliquer sur **Fichier> Ouvrir** et ouvrir le répertoire que vous avez créé à l'étape 1.
3. Maintenant que vous avez créé un projet, nous devons y ajouter des fichiers. Un projet MicroPython standard a la structure suivante :

**RGB-Blink**

**|-lib**

**|- some\_library.py**

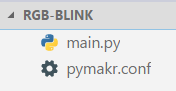
**|-boot.py**

**|-main.py**

* **boot.py** est le premier script exécuté sur le module. Il est souvent utilisé pour le connecter à un réseau WiFi afin que l'on puisse utiliser **Telnet** et **FTP** sans se connecter au point d'accès Wifi (WiFi AP) ni encombrer le fichier main.py. Si vous débutez, vous n'avez pas besoin d'utiliser un fichier boot.py.
* **main.py** s'exécute directement après boot.py et devrait contenir le code principal que vous souhaitez exécuter sur votre appareil.
* Le répertoire **lib.** C'est souvent une bonne idée de séparer le code réutilisable en bibliothèques. Si vous voulez créer ou utiliser des bibliothèques créées par d'autres, vous devrez créer un **répertoire lib** pour les placer. Il est important de placer les fichiers .py directement dans la bibliothèque plutôt que de créer une arborescence de répertoires. **Par défaut, MicroPython ne détectera aucune bibliothèque dans les sous-répertoires**.

Maintenant que la structure du projet est configurée, vous pouvez définir les paramètres spécifiques au projet dans Pymakr, par exemple le port série à utiliser.

Cliquez sur le bouton ***All commands*** dans **Visual Studio Code**, puis sélectionner **F1** suivi de **Pymakr> Project Settings.**





Cela crée un fichier appelé  dans votre projet et le remplit avec les paramètres par défaut copiés à partir de vos paramètres globaux. Une explication détaillée de ces paramètres peut être trouvée [ici](https://docs.pycom.io/chapter/pymakr/settings.html).

* **Contrôler la LED embarquée**

Maintenant que vous avez configuré votre projet, nous pouvons passer à la programmation du module. La première chose à faire est d'importer quelques bibliothèques afin d'interagir avec la LED embarquée. Le firmware Pycom est livré avec une grande quantité de bibliothèques pour les fonctionnalités standard intégrées. Vous pouvez en savoir plus à ce sujet dans la documentation de l'[API](https://docs.pycom.io/chapter/firmwareapi/). Pour cet exemple, vous devrez ouvrir le fichier main.py et ajouter le code suivant :

**import** pycom

**import** time

Cela va importer deux bibliothèques, Pycom qui est responsable des fonctions spécifiques de Pycom, telles que la LED embarquée et time qui est une bibliothèque standard utilisée pour la gestion du temps.

Vous avez peut-être remarqué que lorsque vous allumez votre module, le voyant intégré clignote régulièrement en bleu. Cette "pulsation" permet de savoir que votre module a démarré correctement. Avant de pouvoir changer la couleur de cette LED, nous devons désactiver ce battement. Ajoutez le code suivant après les bibliothèques :

pycom.heartbeat(False)

Il est maintenant temps de tester votre code. Sur le panneau Pymakr en bas de la fenêtre, vous verrez un bouton d'exécution . (Si vous n'êtes pas encore connecté à votre appareil, vous devrez d'abord le faire). Lorsque vous cliquez sur le bouton Run, le code du fichier actuellement ouvert est exécuté sur l'appareil. Le bouton *Run* est brièvement remplacé par un bouton *Stop*. Après avoir exécuté ce code, vous devriez voir que la LED embarquée arrête de clignoter en bleu.

Maintenant que nous pouvons confirmer que le périphérique est connecté et que Pymakr peut exécuter du code, nous pouvons compléter notre script pour faire clignoter la LED comme ceci :

**import** pycom

**import** time

pycom.heartbeat(False)

**while True**:

pycom.rgbled(0xFF0000) # Red

time.sleep(1)

pycom.rgbled(0x00FF00) # Green

time.sleep(1)

pycom.rgbled(0x0000FF) # Blue

time.sleep(1)

Remarque : comme le code du script exécute une boucle while, une fois lancé, il s'exécute indéfiniment. Vous remarquerez que cela vous empêche d'accéder au REPL interactif sur le module (vous ne pouvez pas voir l'invite >>>). Pour arrêter le script, cliquez sur le terminal Pymakr et appuyez sur ctrl-c sur votre clavier ou cliquez sur le bouton  qui a remplacé le bouton RUN.

* **Téléchargement vers votre module**

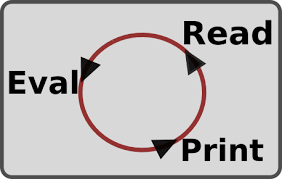
Dans le paragraphe précédent, nous avons fait fonctionner le code sur le module Pycom en utilisant la fonction d'exécution de Pymakr. Ceci est utile pour un test rapide, mais a quelques inconvénients. Premièrement, le code ne reste pas sur l'appareil de façon permanente, si vous le redémarrez, il ne l'exécutera plus. Deuxièmement, cela ne fonctionnera que si vous utilisez des bibliothèques intégrées au firmware. Si vous avez besoin de bibliothèques supplémentaires, celles-ci doivent d'abord être copiées sur l'appareil. C'est là qu'intervient la fonction de téléchargement. Si au lieu de lancer l'envoi, vous cliquez sur télécharger , Pymakr téléchargera tous les fichiers dans le projet (à condition que leur type soit dans le paramètre **sync\_file\_types** dans **pymakr.conf**). Ceux-ci persistent alors sur votre appareil, même entre les redémarrages, et vous permet d'utiliser les bibliothèques du dossier lib dans votre projet.

Si vous avez besoin de supprimer des fichiers de votre appareil, vous avez deux options, soit se connecter via FTP et gérer vos fichiers de cette façon ou formater le ***flash interal*** de l'appareil comme suit :

**import os**

**os.mkfs('/flash')**

### REPL (Read Evaluate Print Loop)



**REPL** signifie **Read Evaluate Print Loop. C'est** le nom donné à **l'invite interactive** MicroPython avec laquelle vous pouvez accéder à un module xPy. L'utilisation de REPL est de loin la manière la plus simple de **tester votre code** python et d'exécuter des commandes.

Vous pouvez utiliser REPL en plus d'écrire des scripts dans main.py.

REPL inclut les fonctionnalités suivantes :

    - **Historique des entrées** : utilisez les flèches vers le haut et vers le bas pour faire défiler l'historique

    - **Tab completion** : appuyez sur la touche Tab pour compléter automatiquement les variables ou les noms de module

    - Arrêtez du code en cours d'exécution avec **Ctrl-C**

    - Copier / coller le code : **Ctrl-C** et **Ctrl-V**

Il existe un certain nombre de raccourcis utiles pour interagir avec MicroPython REPL.

- **Ctrl-A** sur une ligne vide entrera en mode REPL brut. Ceci est similaire au mode collage permanent, sauf que les caractères ne sont pas renvoyés.

- **Ctrl-B** sur un blanc comme va au mode REPL normal.

- **Ctrl-C** annule toute entrée ou interrompt le code en cours d'exécution.

- **Ctrl-D** sur une ligne vide fera une réinitialisation logicielle.

- **Ctrl-E** entre en 'mode collage' qui vous permet de copier et coller des blocs de texte. Quittez ce mode en utilisant Ctrl-D.

- **Ctrl-F** effectue un "démarrage sécurisé" de l'appareil qui empêche boot.py et main.py de s'exécuter

Pour utiliser REPL, vous devez vous connecter au xPy dans le **terminal de VSCode**, via Telnet (avec [PuTTY](https://www.putty.org/) sous Windows), ou avec un convertisseur USB vers série câblé sur l'un des deux UART. Lorsque vous utilisez le REPL sur Telnet, une authentification est nécessaire. Les informations d'identification par défaut sont les suivantes :

**user:** micro

**password:** python

pour en savoir plus : <https://docs.pycom.io/chapter/gettingstarted/programming/repl/serial.html>

### FTP

Il y a un petit système de fichiers interne accessible sur chaque périphérique Pycom, appelé **/flash**. Celui-ci est stocké dans la mémoire flash série externe. Si une **carte micro SD** est connectée et montée, elle sera également disponible. Lorsque l'appareil démarre, il démarre toujours à partir du fichier **boot.py** situé dans le système de fichiers /flash.

Le système de fichiers est accessible via le **serveur FTP natif** qui s'exécute sur chaque périphérique Pycom. Ouvrez un client FTP et connectez-vous à :

**url** : ftp://192.168.4.1

**username** : micro

**password** : python

Voir [**network.server**](https://docs.pycom.io/chapter/firmwareapi/pycom/network/wlan.html) pour plus d'informations sur la façon de changer les valeurs par défaut. Les clients recommandés sont :

- MacOS / Linux: client FTP par défaut

- Windows: Filezilla et FireFTP.

Par exemple, à partir d'un terminal MacOS / Linux:

$ FTP 192.168.4.1

Le serveur FTP ne supporte pas le mode actif, seul le mode passif. Par conséquent, si vous utilisez le client FTP Unix natif, immédiatement après vous être connecté, exécutez la commande suivante :

FTP> passive

Le serveur FTP ne prend en charge qu'une connexion à la fois. Si vous utilisez d'autres clients FTP, consultez leur documentation pour savoir comment limiter les connexions maximales autorisées à une à la fois.

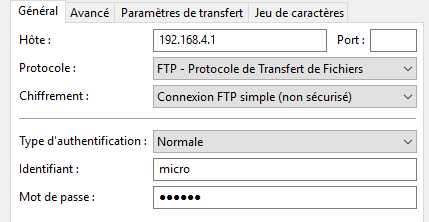
* **FileZilla**



Si vous utilisez FileZilla, il est important de bien configurer les paramètres.

N'utilisez pas le bouton de connexion rapide. Au lieu de cela, ouvrez le gestionnaire de site et créez une nouvelle configuration. Dans l'onglet Général, assurez-vous que le cryptage est défini sur:

Utilisez uniquement le protocole FTP (non sécurisé).



Dans l'onglet Paramètres de transfert, limitez le nombre maximal de connexions à un. Les autres clients FTP peuvent se comporter de manière similaire ; visitez leur documentation pour plus d'informations spécifiques

### Démarrage sécurisé (Safe Boot)

À la mise sous tension ou en appuyant sur le bouton de réinitialisation, un module Pycom démarrera en mode standard ; le fichier **boot.py** sera exécuté en premier, suivi de **main.py**. Il est possible de modifier la procédure de démarrage du module en fixant certaines broches à l'état logique haut ou bas au démarrage du module.

* **bootloader**

Si vous avez mis à jour votre appareil avant de l'utiliser, vous avez déjà mis l'appareil en mode bootloader. Ceci est réalisé en connectant P2 à GND pendant que l'appareil démarre. Si vous avez utilisé un Pysense / Pytrack pour mettre la mise à jour, il l'a fait automatiquement pour vous. Vous avez seulement besoin de mettre votre module Pycom en mode bootloader si vous mettez à jour son firmware, ou si vous programmez votre propre code de bas niveau. Ce n'est pas nécessaire si vous mettez à jour votre code micropython.

* **Safe Boot**

Parfois, le code que vous avez écrit vous empêchera d'accéder au REPL ou vous empêchera de mettre à jour votre code. Par exemple :

- Vous avez désactivé le WiFi / UART

- Votre code est bloqué et vous ne pouvez plus faire fonctionner le REPL

- Vous définissez un socket comme bloquant, mais ne recevez jamais de données

Afin de résoudre ce problème, vous pouvez démarrer votre module en toute sécurité. Cela empêchera l'exécution de boot.py et main.py et vous redirigera directement dans le REPL interactif. Après la réinitialisation, si la broche P12 est maintenue haute (c'est-à-dire si elle est connectée au 3V3), la DEL de pulsation commence à clignoter lentement en orange.

Si au bout de 3 secondes la broche est toujours maintenue haute, la LED commencera à clignoter plus rapidement. Dans ce mode, le module fera comme expliqué précédemment, mais il sélectionnera également l'image OTA précédente si vous avez mis à jour le module via la procédure de mise à jour OTA (les mises à jour effectuées via l'outil de mise à jour du firmware ne comptent pas). Ceci est utile si vous avez flashé une mise à jour OTA qui bloque l'appareil.

Pin P12 relâché pendant :

|  |  |
| --- | --- |
| **1re fenêtre de 3 secondes** | **2e fenêtre de 3 secondes** |
| Désactive boot.py et main.py | Comme précédemment, mais en utilisant le précédent firmware OTA. |

La sélection effectuée lors du démarrage sécurisé n'est pas persistante. Par conséquent, après la réinitialisation normale suivante, le dernier microprogramme sera exécuté à nouveau.

Si des problèmes surviennent dans le système de fichiers ou si vous souhaitez réinitialiser votre module pour supprimer votre code, exécutez le code suivant dans REPL :.

>>> import os

>>> os.mkfs('/flash')

Sachez que la réinitialisation du système de fichiers flash supprimera tous les fichiers se trouvant dans le stockage interne du périphérique (et non la carte SD) et ne pourra pas être récupérée.

* **RESET**

Les périphériques Pycom prennent en charge les réinitialisations logicielles et matérielles. Une réinitialisation logicielle efface l'état de la machine virtuelle MicroPython, mais laisse les périphériques matériels non affectés. Pour effectuer une réinitialisation logicielle, appuyez sur **Ctrl + D** sur le REPL ou depuis un script, lancez :

>>> import sys

>>> sys.exit()

Une réinitialisation matérielle équivaut à exécuter un cycle d'alimentation sur l'appareil. Pour réinitialiser l'appareil, appuyez sur l'interrupteur de réinitialisation ou lancez :

>>> import machine

>>> machine.reset()



### Programmer les cartes avec l'IDE ATOM et l'extension Pymakr

1. **Installer l'IDE ATOM (**[**https://atom.io/**](https://atom.io/) **) et l'extension** [**Pymakr**](https://docs.pycom.io/pycom_esp32/pycom_esp32/toolsandfeatures.html#pymakr-plugin)**.**

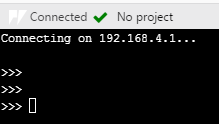


**Pour installer l'extension Pymakr et paramétrer l'IDE :**

<https://docs.pycom.io/>

1. **Programmer dans la console**

- Connecter le module au PC par l'intermédiaire du "Centre de réseau et de partage".



- Ouvrir l'IDE ATOM

Packages -> Pymakr -> Toggle Pycom Console

Le module doit se connecter automatiquement à l'adresse 192.168.4.1 et le **prompt >>>** doit apparaitre.

Entrer le code Python après >>>

1. **Programmer dans un projet**

- **Créer** un projet

File -> Add Project Folder (créer et sélectionner le répertoire)

**Note**

Vous pouvez également utiliser [FTP](https://docs.pycom.io/pycom_esp32/pycom_esp32/toolsandfeatures.html#pycom-filesystem) pour télécharger boot.py et main.py de la carte vers votre dossier de projet. Ceci est couramment utilisé lors de la copie d'un grand nombre de fichiers sur une carte Pycom.

**Pour plus d'informations (connexion avec Filezilla) :**

<https://docs.pycom.io/>

### Annexe : Créer un fichier boot.py de connexion au Wifi

*De nombreux utilisateurs, en particulier les utilisateurs* ***WiPy****, voudront un script WiFi dans le fichier boot.py. Un script WiFi de base, mais aussi des exemples de WLAN plus avancés, comme IP fixe et plusieurs réseaux, se trouvent dans le chapitre "****Exemples de WiFi****". Le script ci-dessous se connecte à votre réseau et imprime l'adresse IP locale de votre appareil.*

**from** machine **import** UART

**import** os

**from** network **import** WLAN

uart **=** UART(0, 115200)

os**.**dupterm(uart)

wlan **=** WLAN(mode**=**WLAN**.**STA)

wlan**.**scan()

wlan**.**connect(ssid**=**'Your Network SSID', auth**=**(WLAN**.**WPA2, 'Your Network Password'))

**while** **not** wlan**.**isconnected():

**pass**

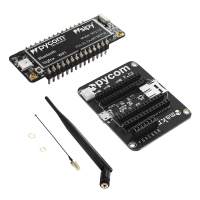
print(wlan**.**ifconfig()) *# prints out local IP to allow for easy connection via Pymakr Plugin or FTP Client*

## Les modules Pycom

Les caractéristiques du module Wipy sont communes à tous les modules (sauf remarques particulières).

### Le module "WiPy 3.0" de Pycom

**Features**



– **Powerful CPU**, **BLE** and state of the art WiFi radio  
– **1KM WiFi** range  
– **MicroPython enabled**   
– Fits in a standard breadboard (with headers)  
– Ultra-low power usage: a fraction compared to other connected micro controllers

**Processing**

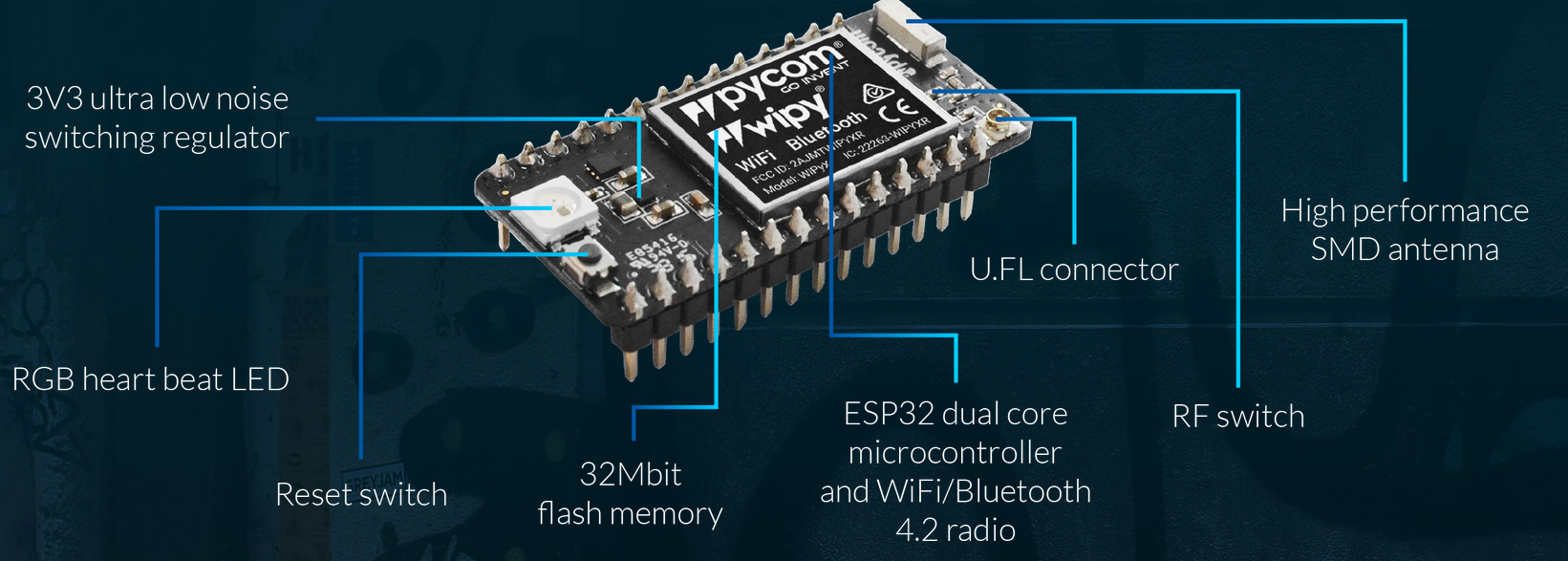
– **Espressif ESP32** chipset  
– Dual processor + WiFi radio system on chip  
– Network processor handles the WiFi connectivity and the IPv6 stack  
– Main processor is entirely free to run the user application  
– An **extra ULP-coprocessor** that can monitor GPIOs, the ADC channels and ; control most of the internal peripherls during

deep-sleep mode while only consuming 25uA

**Interfaces**

– 2 x UART, 2 x SPI, I2C, I2S, micro SD card  
– Analog channels: 8×12 bit ADCs  
– Timers: 4×16 bit with PWM and input capture  
– DMA on all peripherals  
– GPIO: Up to 24

**Hash / Encryption**



SHA, MD5, DES, AES

**WiFi Networking**

802.11b/g/n 16mbps

**Bluetooth**

Low energy and classic

**RTC**

Running at 32KHz

**Power**

– 3.3V to 5.5V  
– 3V3 output capable of sourcing up to 550mA

**Security & Certifications**

– SSL/TLS support  
– WPA Enterprise security  
– FCC – 2AJMTWIPY2R  
– CE 0700

**Memory**

– RAM: 4MB  
– External flash: 8MB  
– Hardware floating point acceleration  
– Python multi-threading

**Size**

- 42mm x 20mm x 2.5mm (excluding headers)

### Le module "LoPy" de Pycom

**Features**

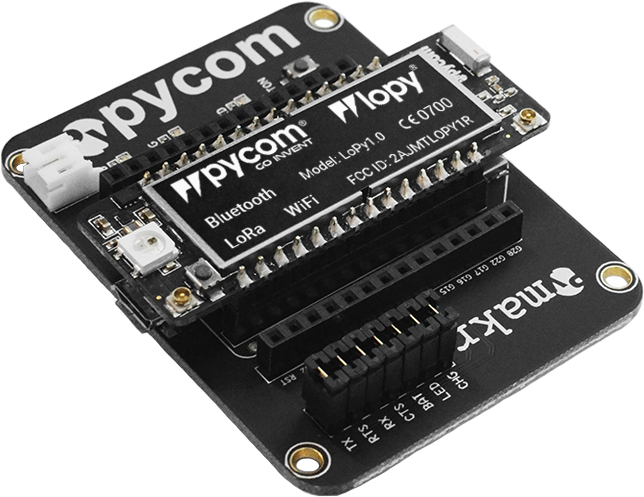
– Can also double up as **Nano LoRa gateway**

**NOTE: Using the LoRa radio without the external antenna can lead to damage of the device and is therefore not recommended.**

**Operating Frequencies**

– 868 MHz (Europe) at +14dBm maximum  
– 915 MHz (North and South America, Australia and New Zealand) at +20dBm maximum

**Range Specification**



– Node range: Up to 40km  
– Nano-gateway: Up to 22km  
– Nano-gateway capacity: Up to 100 nodes

**Memory**

– RAM: 512KB  
– External flash: 4MB

**Power**

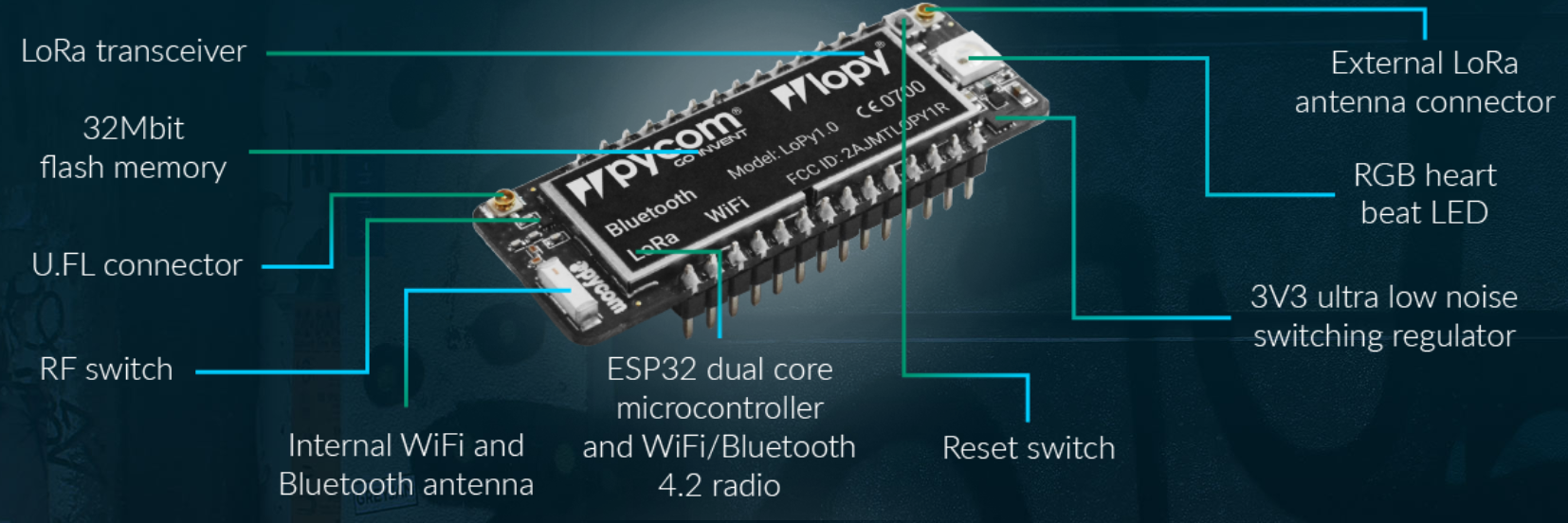
– Input: 3.3V – 5.5V  
– 3v3 output capable of sourcing up to 400mA  
– WiFi: 12mA in active mode, 5uA in standby  
– LoRa: 15mA in active mode, 1-uA in standby

**LoRa Specification**

– Semtech LoRa transceiver SX1272  
– LoRaWAN stack  
– Class A and C devices

**Size**

55mm x 20mm x 3.5mm (Without Headers)



### Le module "SiPy" de Pycom

**Features**

**NOTE: Using the Sgfox radio without the external antenna can lead to damage of the device and is therefore not recommended.**

**Sigfox Operating Frequencies**

– RCZ1 – 868MHz (Europe)  
– RCZ2 – 902MHz (US, Canada and Mexico)  
– RCZ3 – (Japan and Korea)  
– RCZ4 – 920-922MHz (ANZ, Latin America and S-E Asia)

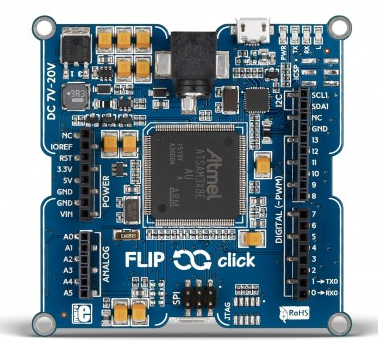


# La carte "Flip & Click" de MikroElektronika

<https://www.mikroe.com/flip-n-click-sam3x>

## Caractéristiques

|  |  |
| --- | --- |
| **Applications** | Beginner-friendly rapid prototyping tool expandable with up to four click boards |
| **Architecture** | ARM (32-bit) |
| **MCU** | AT91SAM3X8E |
| **MCU speed** | 84MHz |
| **MCU Memory** | 512KB of Flash, 100KB of SRAM |
| **Programming** | Via Arduino Bootloader |
| **Expandability** | 4 x mikroBUS sockets, 1 x Arduino UNO socket |
| **Integration** | Mounting Holes |



## IDE : Zerinth Studio



<https://www.zerynth.com/zerynth-studio/>

# Annexes

## Exemples avec les modules Pycom

À venir