Aabu

Configurer PyCharm pour MicroPython sous Linux

22 mai 2021

Table des matières

Pour le développement Python, j'apprécie le confort de PyCharm, mon IDE Python habituel. Il est également possible de l'utiliser pour des projets MicroPython, mais dans sa configuration de base, les aides habituelles sont absentes voire induisent en erreur. Heureusement, il est possible de le configurer pour rendre le développement avec MicroPython confortable.

J'explique dans ce billet comment configurer PyCharm afin de travailler confortablement sur un projet MicroPython.

Ce billet a été réalisé avec la configuration suivante:
- Ubuntu 20.04.1,
- PyCharm Community 2020.2.3,
- Plugin MicroPython 1.1.2,
- Pyboard v1.1.

Si vous avez une autre configuration, des différences peuvent apparaître. En particulier, le plugin pourrait temporairement ne pas être compatible avec les toutes dernières versions de PyCharm.

1. Installation

1.1. Installer le plugin Micropython pour PyCharm

La configuration de PyCharm est très simple puisqu'il suffit essentiellement d'installer un *plugin*. Le *plugin* en question est nommé tout simplement *MicroPython*.

Pour l'installer, procédez ainsi:

- allez dans le menu File | Settings | Plugins;
- allez sur l'onglet Marketplace;
- saisir «*MicroPython*» dans le champ de recherche;
- cliquez sur le gros bouton vert Install sur l'entrée «MicroPython».

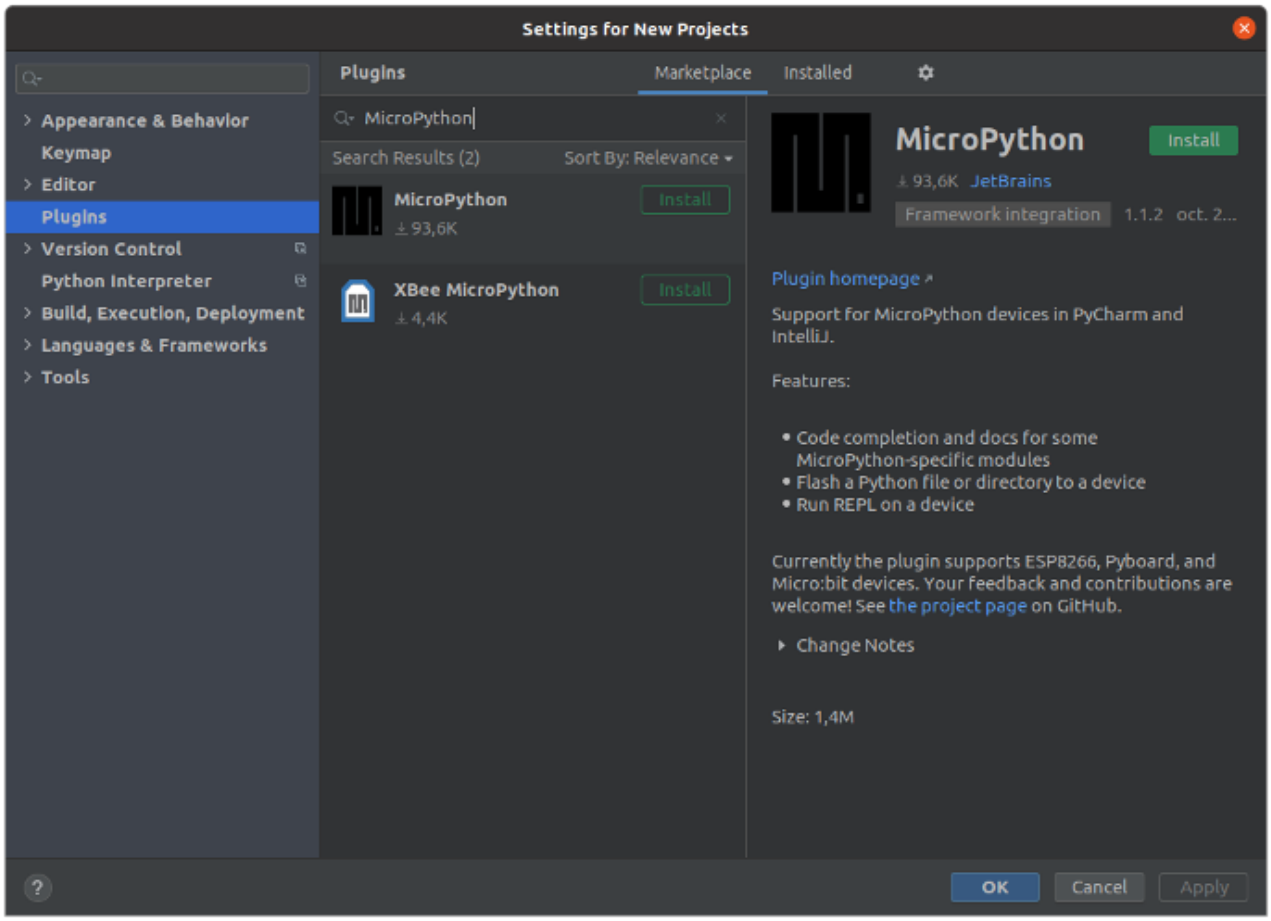


FIGURE 1.1. – Le *plugin* MicroPython juste avant l'installation.

Après l'installation, constatez la bonne installation dans l'onglet Installed.

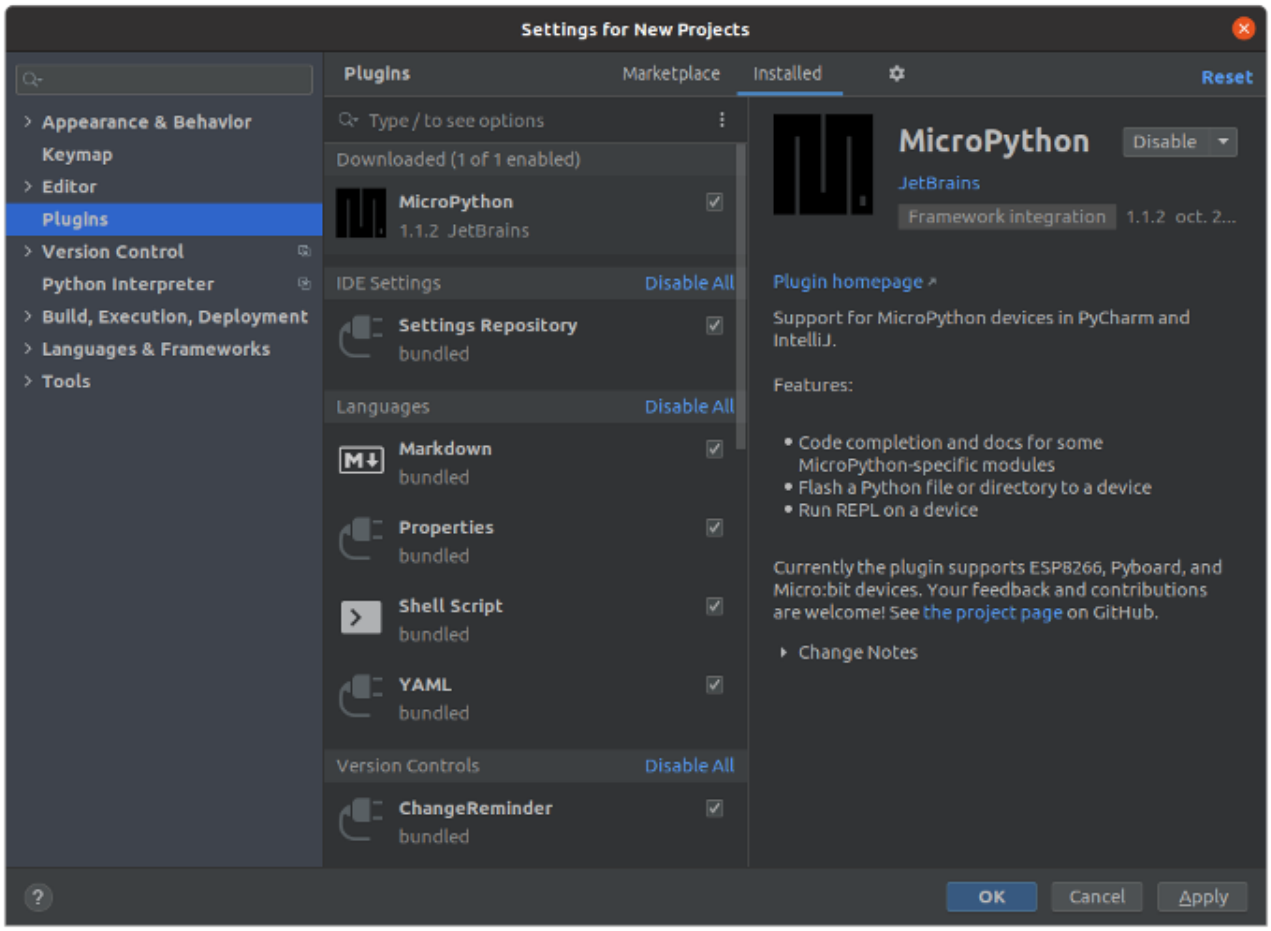


FIGURE 1.2. – Le *plugin* MicroPython après installation.

En cas de doute sur la compatibilité du *plugin* avec votre version de PyCharm, regardez les notes de version. Les développeurs y indiquent en général les modifications liées à la compatibilité avec une version récente. Par exemple, pour la version 1.1.2, on y lit « *Compatibility with 2020.2–2020.3* ».

1.2. Ajouter votre utilisateur au bon groupe

Pour une des fonctionnalités du *plugin*, il faut ajouter l'utilisateur au groupe `dialout`:

```
1 sudo usermod -a -G dialout <username>
```

en remplaçant `<username>` par votre nom d'utilisateur.

2. Configurer un projet MicroPython

À la création d'un nouveau projet, il faut choisir d'activer le *plugin* pour ce projet:

— aller dans le File | Settings | Languages & Frameworks | MicroPython;
— cocher sur Enable MicroPython support;
— choisissez votre carte (Pyboard dans mon cas);
— cocher Auto-detect device path.

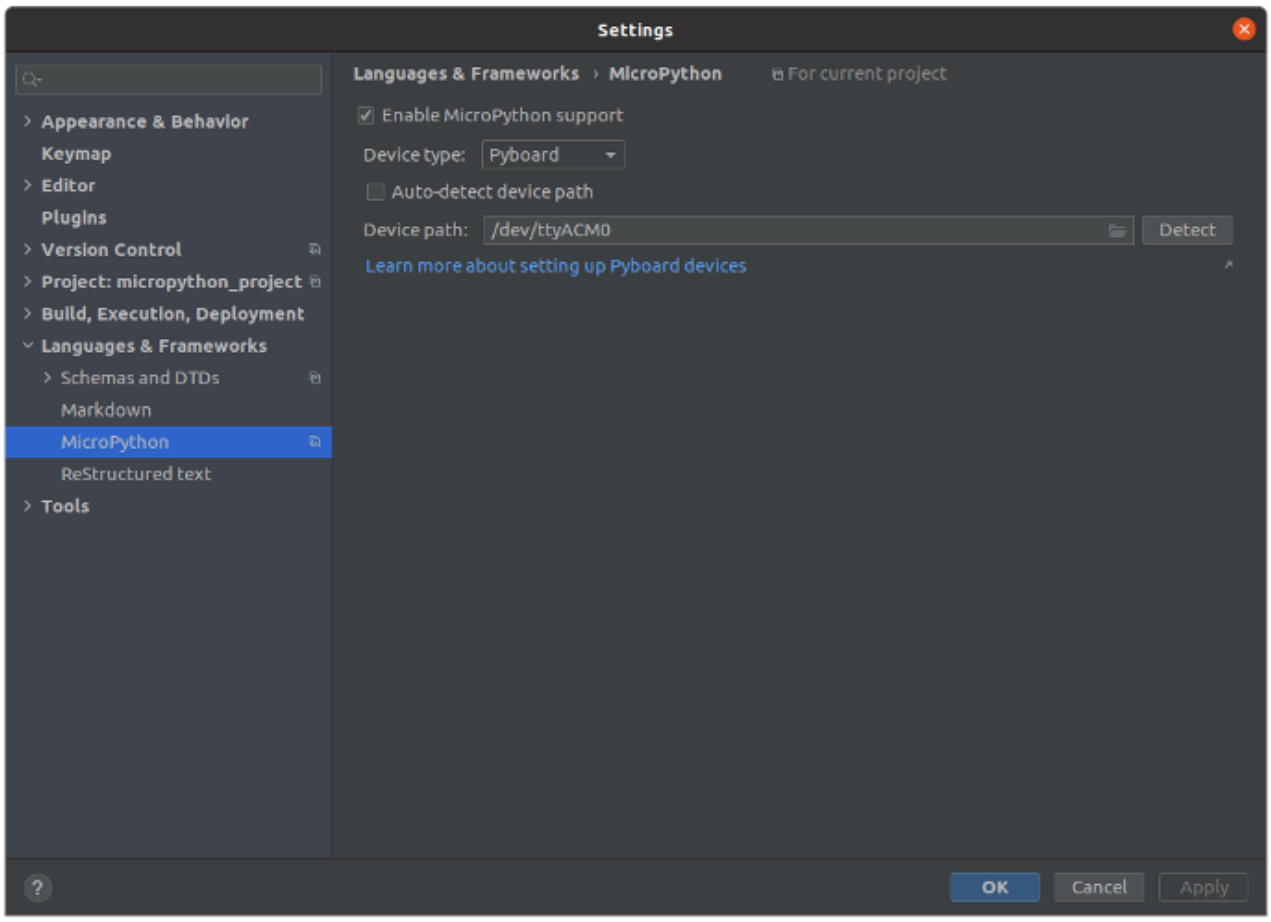


FIGURE 2.3. – image.png

L'auto-détection de la carte (et donc la communication avec elle) ne marchera pas pour le moment, il faut installer des paquets supplémentaires pour cela:

— aller dans n'importe quel fichier Python
— un bandeau jaune devrait apparaître vous demandant d'installer trois paquets manquants (`pyserial`, `docopt` et `adafruit-ampy`).
— cliquer sur Missing required MicroPython packages et attendez que l'installation se termine.

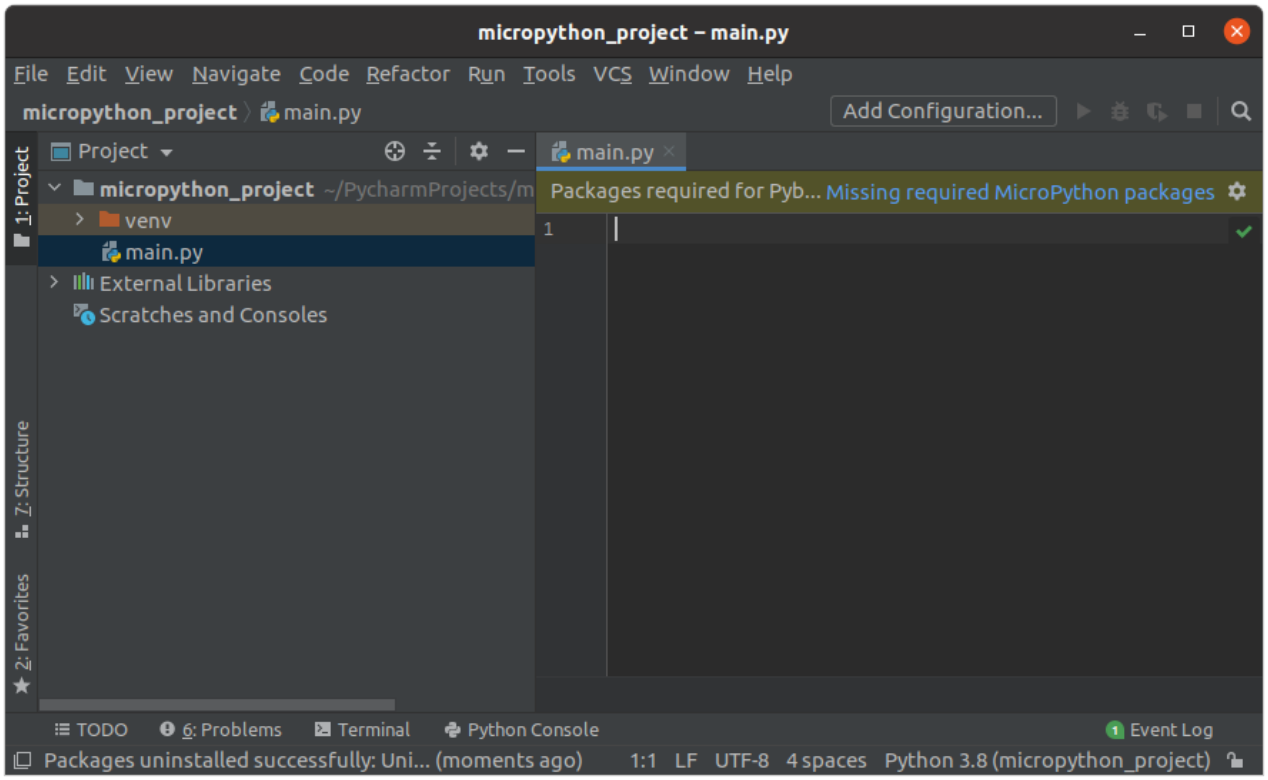


FIGURE 2.4. – Le *plugin* demande d'installer les packages supplémentaires pour communiquer avec la carte.

Normalement tout est bon. Pour tester si la détection et communication est bonne vous pouvez:

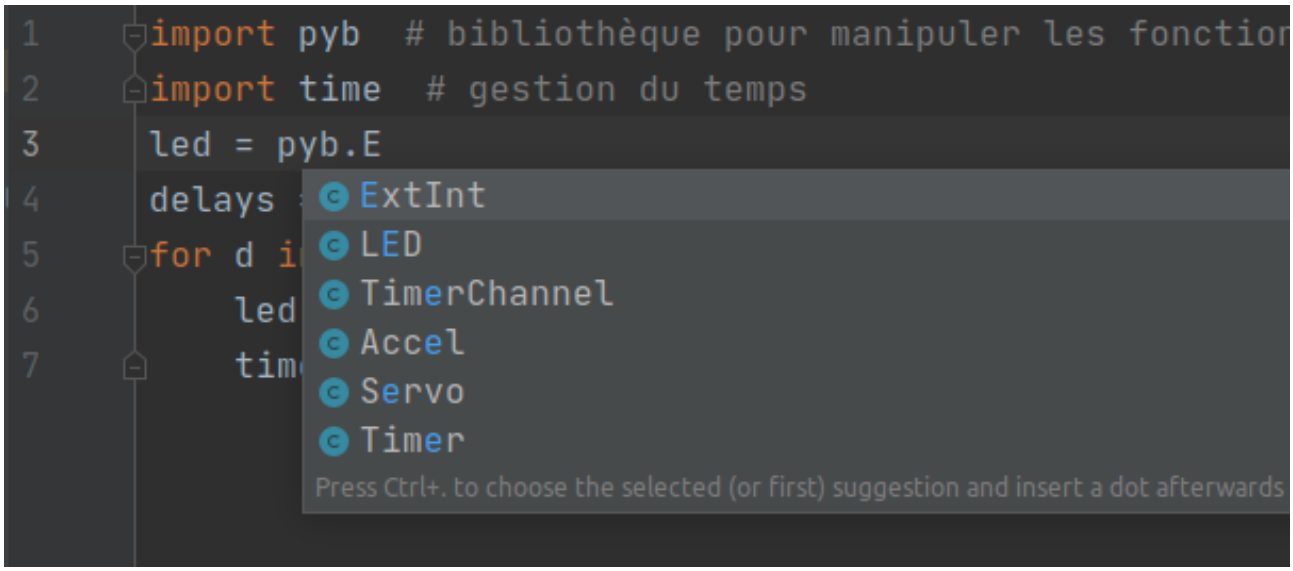
— aller dans le File | Settings | Languages & Frameworks | MicroPython et cliquer sur Detect et constater que ça fonctionne;
— aller dans Tools | MicroPython | MicroPython REPL et constater que la session interactive s'ouvre sans erreur.

C'est bon.

3. Utilisation des outils

Cette liste n'est pas exhaustive (pour une référence je vous conseille le dépôt officiel du *plugin*).

Avec le *plugin*, on bénéficie de propositions de complétion:



On a aussi, comme mentionné précédemment, un interpréteur interactif *sur la carte*. Pour y accéder, aller dans Tool | MicroPython | MicroPython REPL. *Je conseille d'assigner un raccourci clavier pour y accéder plus facilement.*

Il faut que l'utilisateur soit dans le groupe `dialout` pour que l'interpréteur fonctionne. Dans le cas contraire, vous aurez une erreur de permissions.

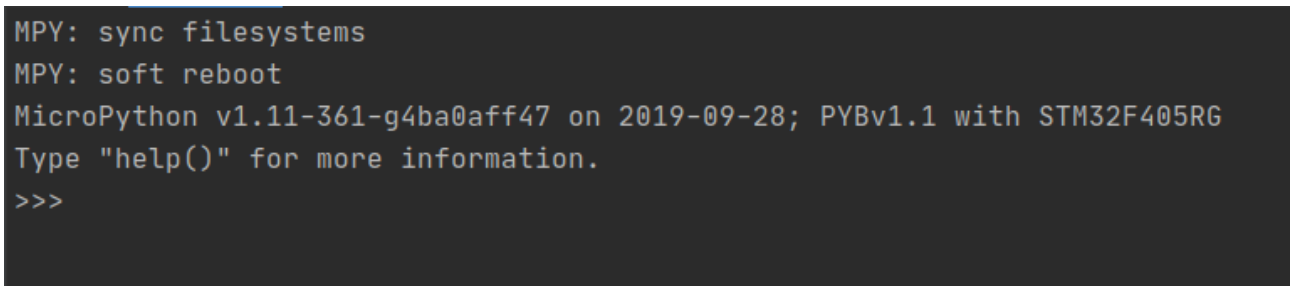


FIGURE 3.5. – REPL.

Avec Tool | MicroPython | Remove All Files from MicroPython Device, on peut vider la carte. Pratique pour partir sur un état connu et effacer des traces d'errances du passé.

On peut aussi flasher des fichiers et des dossiers sur la carte. Dans l'explorateur de projet:

— cliquer droit sur un dossier ou un fichier;
— choisir Run Flash <nom du dossier>.

Notez que cette fonctionnalité ne met pas à jour l'affichage du stockage de masse. Vous pouvez donc avoir une différence entre le contenu réel de la carte et ce que vous voyez dans votre explorateur de fichier favori. Il suffit de relancer la carte pour que ça soit mis à jour.

3.1. Aller plus loin

Une addition intéressante à cette configuration serait d'adapter la configuration de l'environnement virtuel d'un projet MicroPython pour qu'il soit plus en accord avec MicroPython (qui est basé sur Python 3.5, avec de nombreuses particularités sur la bibliothèque standard), mais je n'ai pas creusé la question.

Miniature du billet: logo de MicroPython (source).