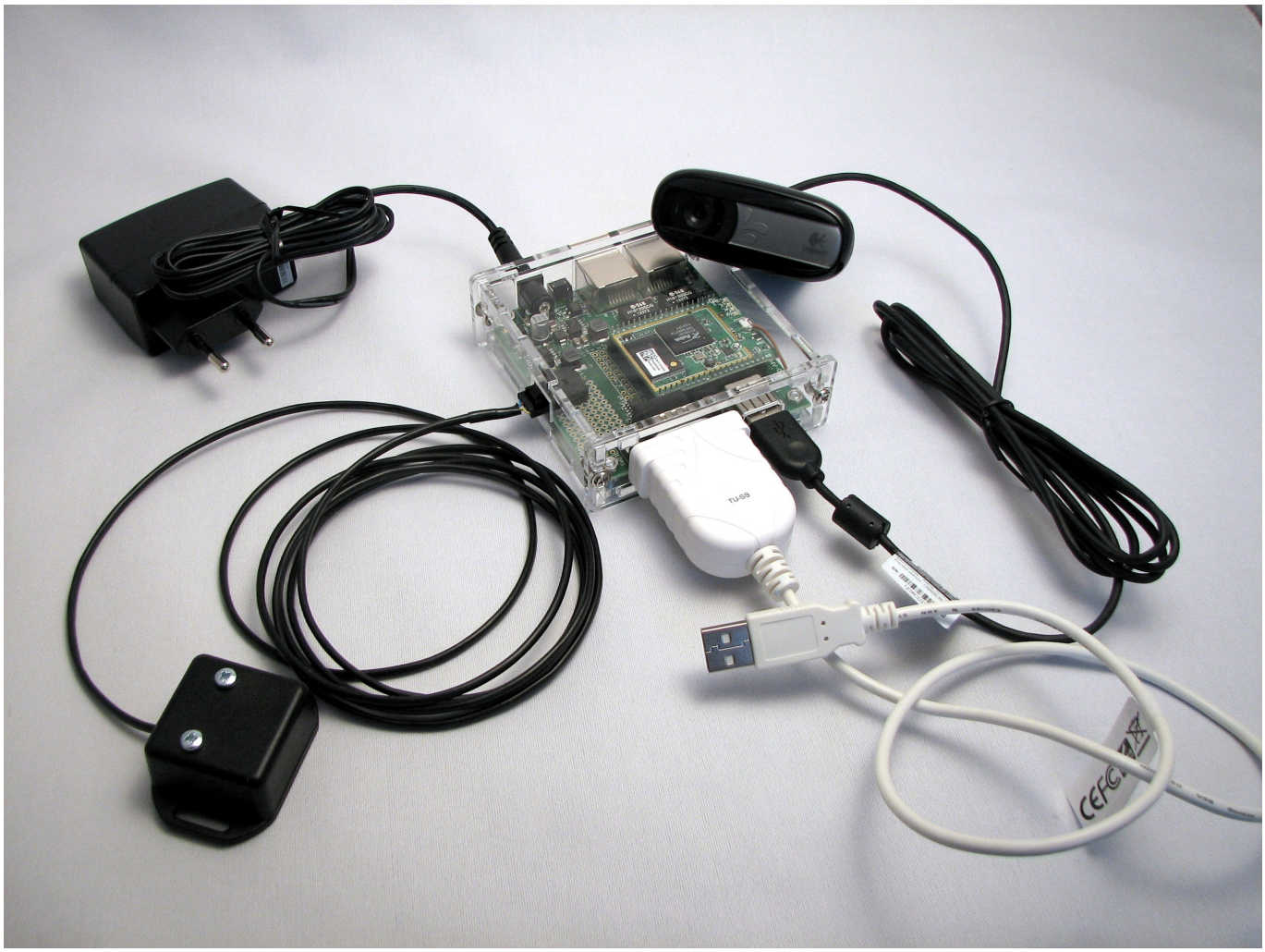
CARAMBOLA-BOX

DOCUMENTATION COMPLÈTE



Date de dernière mise à jour : 18/12/2013

Table des matières

1 - Présentation générale de la Carambola-Box

La Carambola-Box est une plate-forme informatique embarquée open-source avec connectivité Wifi et i2c native permettant de faire de nombreuses expériences, en connectant par exemple une Webcam USB ou une mini centrale inertielle 6 axes sur le port i2c.

Elle est constituée d'un boîtier en plexiglas transparent contenant une carte Carambola. Tournant sous Linux OpenWrt, cette carte est le cœur du système, offrant nativement différents types de connectivité :

- Wifi
- Ethernet
- Console série
- USB
- broches GPIO (General Purpose Input Output) ajoutant entre autres des possibilités de communication avec des périphériques i2c

Dans sa version la plus complète, la Carambola-Box est livrée avec les éléments suivants :

- une mini centrale inertielle (baptisée IMU dans la suite de cette documentation), basée sur le composant MPU6050, apportant 6 axes de mesure (3 accélérations et 3 vitesses de rotation) et se connectant sur la prise i2c de la Carambola-Box
- une Webcam qui peut se connecter sur le port USB du boîtier pour faire de la transmission vidéo temps-réel via les liaisons Wifi ou Ethernet
- un câble d'interface USB / RS232 permettant de connecter un ordinateur ne possédant pas de port série sur la console série de la carte
- une alimentation

Enfin, de nombreux logiciels applicatifs principalement écrits en langage Python sont fournis en code source. Les scripts devant s'exécuter sur le système sont préchargés sur celui-ci à la livraison ; les logiciels destinés à l'ordinateur hôte sont téléchargeables à l'adresse suivante :

http://www.3sigma.fr/Telechargements.html

Ce système permet de réaliser de nombreux travaux pratiques, par exemple :

- connexion au système via la console série ou un terminal SSH
- exécution de commandes sur un système d'exploitation embarqué
- affichage des données de l'IMU dans la console série ou le terminal SSH
- stockage des données de l'IMU sur une clé USB branchée sur la carte. Si le système est alimenté par une batterie (comme celle-ci : http://boutique.3sigma.fr/14-batterie-lipo-74-v-2200-mah-connecteur-jack-arduino.html, non fournie) la Carambola-Box peut se transformer en enregistreur longue durée de données accélérométriques sur n'importe quel type de matériel en mouvement
- stockage des données de l'IMU dans une base SQL embarquée dans le système
- manipulation sur ces données en langage SQL directement sur le système

- télémétrie des mesures de l'IMU sur un ordinateur connecté (en Ethernet ou en Wifi) au système
- affichage de l'orientation de la plate-forme en 3D sur un ordinateur connecté (en Ethernet ou en Wifi) au système
- envoi des mesures de l'IMU sur le port série du système et récupération sur n'importe matériel équipé d'un port RS232
- transmission en temps-réel (en Ethernet ou en Wifi) de la vidéo prise par la Webcam connectée sur le port USB
- téléchargement de pages HTML sur le serveur Web intégré et exécution des applications Web correspondantes

2 - La carte Carambola

Cette carte constitue le cœur du système. Elle est basée sur un SoC (System On Chip) RT3050 possédant une connectivité Wifi native. C'est un véritable mini-ordinateur embarqué tournant sous la distribution Linux OpenWrt (http://wiki.openwrt.org/about/start), dédiée aux systèmes embarqués.

Les SoC Wifi sous OpenWrt sont très utilisés comme routeurs dans le grand public, mais l'ouverture matérielle de la carte Carambola lui permet de réaliser des applications beaucoup plus passionnantes et originales.

Voyons tout d'abord comment s'y connecter pour accéder au système Linux.

2.1 - Connexion avec la carte de communication

Il existe 3 possibilités de connexion:

- Via une liaison série
- Via une liaison Ethernet filaire
- En Wifi

2.1.1 - Connexion via une liaison série

Ce type de connexion est très utile quand les autres modes ne fonctionnent plus, suite à une erreur de configuration, par exemple. Les étapes à suivre pour y parvenir sont les suivantes:

- Brancher un câble série classique entre l'ordinateur hôte et le boîtier. Si l'ordinateur ne possède pas de port série, ce qui est plus que probable, utiliser le câble de conversion USB ↔ RS232 normalement fourni avec le système (installer au préalable le pilote fourni sur le CDROM accompagnant le câble de conversion).
- Identifier le port série sur lequel est branché ce câble (par exemple, COM2 sur Windows, /dev/ttyS0 sous Linux,...)
- Lancer sur l'ordinateur le logiciel adéquat (par exemple: RealTerm sur Windows, minicom sur Linux,...) et se connecter au port identifié à l'étape précédente, avec les propriétés suivantes:
  115200 bit/s
  bits de données: 8
  parité: aucune
  bits d'arrêt: 1
  contrôle de flux: aucun
- Mettre le système sous tension (il suffit de brancher l'alimentation sur le connecteur jack adéquat)
- Appuyer sur la touche « Entrée » dans le logiciel de communication: les messages de démarrage défilent alors jusqu'à obtenir la ligne de commande permettant d'interagir avec le système Linux

2.1.2 - Connexion via une liaison Ethernet filaire

Ce mode permet d'avoir une interface un peu plus confortable qu'avec une liaison série mais nécessite que la configuration réseau soit correcte (ce qui est le cas lorsque le système est livré). Le principal avantage de ce type de communication est qu'il permet d'ajouter la Carambola-Box à un réseau local connecté à Internet. C'est très pratique si pour installer un package d'OpenWrt (par exemple) téléchargé directement sur Internet.

Les étapes à suivre sont les suivantes:

- Brancher un câble Ethernet entre l'ordinateur hôte et le premier port RJ45 de la carte Carambola, celui situé juste à côté du connecteur d'alimentation
- Brancher l'alimentation. Au bout d'environ 1 minute, le système est lancé, 3 diodes vertes doivent être allumées.
- L'accès au système Linux se fait ensuite en SSH sur l'adresse 192.168.0.199 (voir section 2.2)

Noter les points suivants:

- La Carambola-Box est initialement configurée avec une adresse IP fixe (192.168.0.199). Si vous connectez la Carambola-Box à votre réseau local et que celui-ci utilise une autre plage d'adresse que 192.168.0.xxx, vous n'aurez pas accès au système. C'est malgré tout faisable à condition de modifier le paramétrage de sa partie « réseau » (voir section 2.3)
- Pour que votre appareil puisse accéder à la Carambola-Box, il doit accepter l'adressage automatique (adresse fournie par serveur DHCP). Si votre appareil n'accepte pas l'adressage automatique, cela signifie que vous lui avez donné une adresse IP fixe. Dans ce cas, celle-ci doit commencer par 192.168.0. (elle doit donc être de la forme 192.168.0.xxx). Si ce n'est pas le cas, vous ne pourrez pas communiquer avec la Carambola-Box.

2.1.3 - Connexion via une liaison Wifi

Ce mode est le plus souple puisqu'il ne nécessite pas de câble, ce qui est très pratique si le système doit être déporté. La configuration réseau Wifi doit cependant être correcte (ce qui est le cas lorsque le système est livré).

Les étapes à suivre sont les suivantes:

- Brancher l'alimentation de la Carambola-Box
- Attendre environ une minute que le système ait démarré (2 diodes s'allument si un câble Ethernet n'est pas branché, 3 si un câble Ethernet est branché)
- Un réseau Wifi avec un SSID de la forme « Carambola-Box-N » est alors disponible. Dans le nom du SSID, N est un ou plusieurs chiffres
- Le mot de passe est pass-carambola-box-N (le même N qu'au point précédent). Par exemple, si le SSID est Carambola-Box-5, le mot de passe est pass-carambola-box-5
- Une fois connecté en Wifi, l'accès au système Linux se fait alors en SSH sur l'adresse 192.168.0.199 (voir section 2.2)

Noter les points suivants:

- Un appareil donné ne peut pas en général se connecter en même temps à deux réseaux Wifi différents: la connexion d'un ordinateur à la Carambola-Box entraînera la déconnexion du réseau auquel vous étiez éventuellement connecté au préalable. Vous n'aurez alors plus accès à Internet sur votre appareil, sauf si celui-ci est :
  - un ordinateur connecté en parallèle en filaire (liaison Ethernet). Attention cependant: si ce réseau Ethernet utilise la même plage d'adresse (192.168.0.xxx), vous devrez peut-être le désactiver pour vous connecter à l'adresse 192.168.0.199 de la Carambola-Box en Wifi
  - un smartphone ou une tablette connecté à un réseau 3G en parallèle
- La Carambola-Box fonctionne comme un point d'accès « maître ». Il ne peut pas, dans sa configuration de base, se connecter comme client du réseau Wifi de votre organisation. C'est malgré tout faisable à condition de modifier le paramétrage de sa partie « réseau » (voir section 2.3)
- Pour que votre appareil puisse accéder au système, il doit accepter l'adressage automatique (adresse fournie par serveur DHCP). Si votre appareil n'accepte pas l'adressage automatique, cela signifie que vous lui avez donné une adresse IP fixe. Dans ce cas, celle-ci doit commencer par 192.168.0. (elle doit donc être de la forme 192.168.0.xxx). Si ce n'est pas le cas, vous ne pourrez pas communiquer avec la Carambola-Box.

2.2 - Accès au système Linux en SSH

Lorsque la communication se fait en Ethernet filaire ou en Wifi, vous pouvez accéder au système Linux embarqué en ligne de commande suivant le protocole SSH. Les étapes à suivre sont les suivantes:

- Lancez un logiciel de communication supportant le SSH (PuTTY, WinSCP,... sur Windows, OpenSSH,... sur Linux,...)
- Connectez-vous à l'adresse IP de la Carambola-Box (192.168.0.199 est l'adresse par défaut) avec les informations suivantes:
  - login: root
  - mot de passe: admin

Vous accédez alors à la ligne de commande du système Linux embarqué.

ATTENTION !
Avec WinSCP, choisir le protocole de fichier SCP.

2.3 - Configuration réseau

Les propriétés réseau (adresse IP, nom du SSID, mots de passe,...) de la Carambola-Box peuvent être facilement modifiées, de deux façons différentes, exposées ci-après.

2.3.1 - En ligne de commande

Vous pouvez modifier directement les fichiers de configuration suivants (attention: réservé uniquement aux experts !):

/etc/config/network

/etc/config/wireless

/etc/config/dhcp

/etc/config/firewall

2.3.2 - Avec l'interface Web LuCI

LuCI est une interface graphique permettant de configurer facilement les propriétés « réseau » du système. L'accès à cette interface se fait via l'adresse suivante:

192.168.0.199:81

Si vous avez configuré une autre adresse IP, veuillez bien sûr l'utiliser à la place de l'adresse par défaut donnée dans cet exemple. Les informations de connexion sont les suivantes:

Login: root

Mot de passe: admin

2.4 - Installation de Python

Python est pré-installé sur la Carambola-Box. Notez que pour limiter la place utilisée en mémoire flash, c'est le package « python-mini » (et non « python-full ») de la distribution OpenWrt qui est installé. En effet, l'immense majorité des modules de « python-full » a peu de chances d'être utilisé dans une utilisation pratique de la Carambola-Box.

Cependant, si un module de « python-full » s'avérait nécessaire (son absence générant une erreur lors de l'exécution d'un script), il est possible de l'ajouter grâce à la procédure suivante :

- télécharger PythonFullCarambola.zip dans la section « Carambola-Box » de notre page de téléchargements (http://www.3sigma.fr/Telechargements.html)
- après avoir décompressé l'archive sur votre ordinateur, rechercher le module qui vous intéresse dans le répertoire python2.7 (fichiers.py) ou le répertoire python2.7/lib-dynload (fichiers .so)
- grâce à WinSCP sous Windows ou aux outils nativement intégrés sous Linux, transférer le(s) fichier(s) correspondant(s) à ce module sur la Carambola-Box, en respectant la même arborescence dans /usr/lib/python2.7
- relancer le script Python qui générait une erreur pour vérifier que le problème est résolu ou bien s'il est nécessaire d'ajouter un autre module

Par exemple, les scripts Python disponibles sur la Carambola-Box à sa livraison (voir paragraphe 4) nécessitent entre autres les modules « sched » (permettant d'exécuter des fonctions à cadence fixe) et « sqlite3 » (pour la gestion de base de données SQL). Le support de ces deux modules par la Carambola-Box a simplement nécessité l'ajout des éléments suivants dans /usr/lib/python2.7 :

- pour « sched » : fichier sched.py
- pour « sqite3 » : fichier lib-dynload/_sqlite3.so

Pour toute question, n'hésitez pas à nous contacter à l'adresse support@3sigma.fr.

2.5 - Processus de travail pour le développement de scripts Python

Une cible embarquée comme la Carambola-Box n'est pas adaptée pour y exécuter un éditeur ou un environnement de développement. Par conséquent, le processus classique de développement est le suivant :

- Un script est développé sur l'ordinateur hôte, en utilisant n'importe quel éditeur ou IDE Python (PyCharm, Aptana Studio, un simple éditeur de texte avec coloration syntaxique,...)
- Si le script n'utilise pas les ressources matérielles de la Carambola-Box, il est possible de le tester sur l'ordinateur hôte. Attention cependant : le fonctionnement sera le même que sur la Carambola-Box si les mêmes bibliothèques additionnelles éventuellement utilisées par le script y sont installées (la Carambola-Box utilise une version allégée de Python)
- Si le script utilise des ressources matérielles de la Carambola-Box, il n'est pas possible de le tester sur l'ordinateur hôte
- Il faut ensuite télécharger le script sur la Carambola-Box, en utilisant WinSCP par exemple.
- Il ne reste plus qu'à tester le script sur la cible en se plaçant dans le répertoire contenant le script et en exécutant la commande suivante :
  python nom_du_script.py

L'exécution d'un script sur la Carambola-Box peut générer des erreurs s'il comporte des bugs. Le débuggage d'un script qui effectue des opérations en temps-réel (comme ceux permettant d'acquérir les données de l'IMU) peut être délicat car y placer des points d'arrêt n'a pas beaucoup de sens.

Dans ce cas, la meilleure solution est d'ajouter des instructions « print » pour afficher des valeurs de variables importantes.

Dans le cas où le débuggage avec point d'arrêt a un sens, il est possible de débugger facilement un script, ou bien en ligne de commande ou avec un environnement de développement. C'est ce qui est présenté dans les deux paragraphes suivants.

2.6 - Débuggage d'un programme Python en ligne de commande

Le débuggage d'un script Python se fait aisément en ligne de commande en invoquant le module « pdb ».

Prenons l'exemple du programme de test très simple suivant, situé dans /root/DebugPython et nommé test_script.py :

```
a = 1
b = 2
c = a + b
print c
```

Pour lancer l'exécution de ce script en mode debug, il faut l'exécuter avec la commande suivante :

```
python -m pdb test_script.py
```

On se retrouve sous le prompt de pdb :

```
> /root/DebugPython/test_script.py(4)<module>()
→ a = 1
(Pdb)
```

Attention : la ligne qui est affichée n'a pas encore été exécutée, c'est celle qui va être exécutée.

Une autre possibilité consiste à charger pdb directement dans le script et à y placer des points d'arrêt (avec la commande pdb.set_trace()) comme ceci :

```
import pdb
a = 1
b = 2
pdb.set_trace()
c = a + b
print c
```

L'exécution se fait alors simplement comme ceci

```
python test_script.py
```

L'exécution s'arrête alors au point d'arrêt :

```
> /root/DebugPython/test_script.py(7)<module>()
→ c = a + b
(Pdb)
```

La première méthode a l'avantage d'être non intrusive, puisqu'il n'est pas nécessaire de modifier le script.

La seconde est utile pour débugger de gros programmes, en plaçant un point d'arrêt juste avant le passage qui crée une erreur, sans avoir besoin d'exécuter pas à pas le programme jusqu'à ce point.

Les commandes utiles sous le prompt de pdb sont les suivantes :

- **n** (next) : passe à l'instruction suivante
- **s** (step into) : passe à l'instruction suivante à l'intérieur d'un sous-programme
- **r** (return) : passe à l'instruction suivante du programme appelant (typiquement exécuté pour sortir d'un sous-programme dans lequel on est entré avec **s**)
- **Entrée** : répète la dernière commande de débuggage
- **p <nom_de_variable>** : affiche la valeur de la variable
- **c** (continue) : continue l'exécution jusqu'au prochain point d'arrêt ou jusqu'à la fin du programme
- **l** (list) : permet de voir où on se trouve dans le code
- **q** (quit) : termine pdb et l'exécution du script. L'utilisation de **c** est plus propre

2.7 - Exécution et débuggage interactif directement sur la cible

A la date d'écriture de cette documentation et à notre connaissance, un seul environnement de développement permettant de réaliser une exécution et un débuggage directement sur la cible : JetBrains PyCharm.

La procédure à suivre scrupuleusement est détaillée ci-dessous.

2.7.1 - Installation et configuration

Les étapes suivantes ne sont à réaliser qu'une seule fois.

Téléchargement et installation de PyCharm

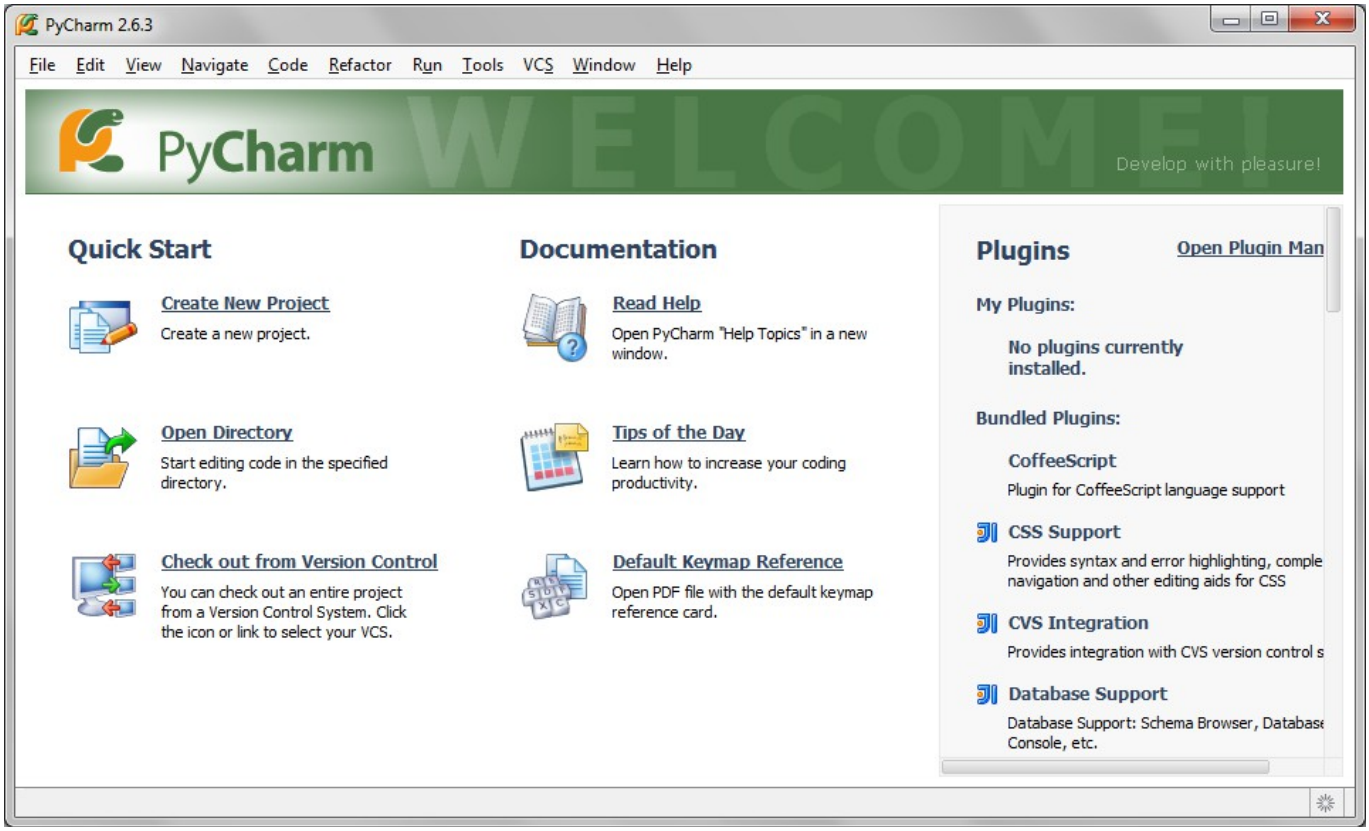
Télécharger la version 2.6.3 (utiliser impérativement cette version, bien qu'il en existe des plus récentes) à l'adresse suivante :

http://confluence.jetbrains.com/display/PYH/Previous+PyCharm+Releases

L'installation se fait sans difficulté en utilisant les options par défaut.

Configuration de PyCharm

Une fois PyCharm complètement lancé (après avoir passé l'accord de licence, le choix des thèmes,... en laissant tout aux valeurs par défaut), on se retrouve face à l'interface d'accueil suivante :



Nous allons maintenant configurer PyCharm pour qu'il utilise l'interpréteur Python de la Carambola-Box, ce qui va être fondamentalement différent de ce que l'on fait d'habitude avec d'autres environnement de développement, qui utilisent l'interpréteur Python de la machine hôte.

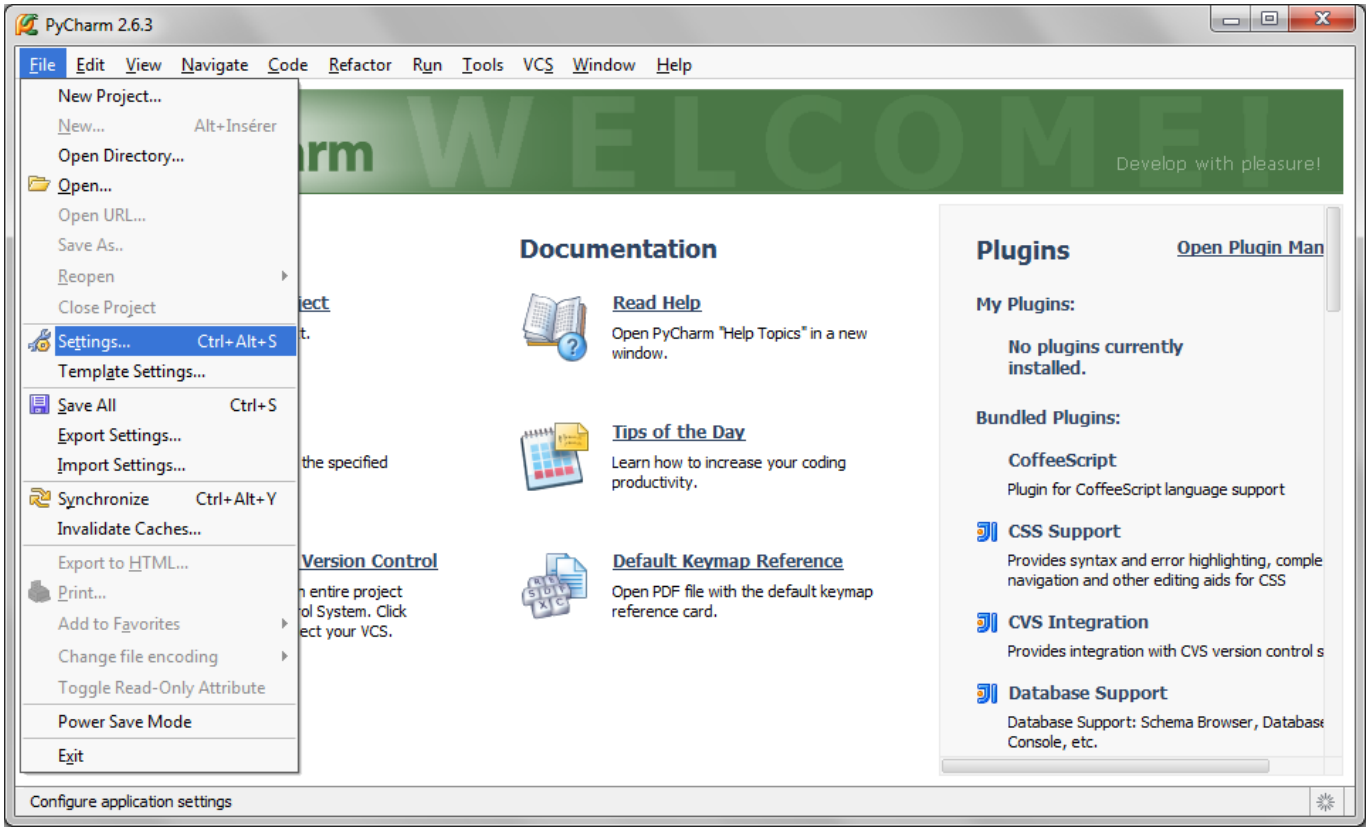
Cette configuration va conduire PyCharm à copier un certain nombre de fichiers sur la Carambola-Box. Ces fichiers n'ayant dans l'absolu rien à faire de façon permanente sur la cible (et celle-ci ne possédant pas assez de mémoire pour les accueillir), nous allons les copier sur une clé USB connectée sur la Carambola-Box.

La procédure à suivre est la suivante :

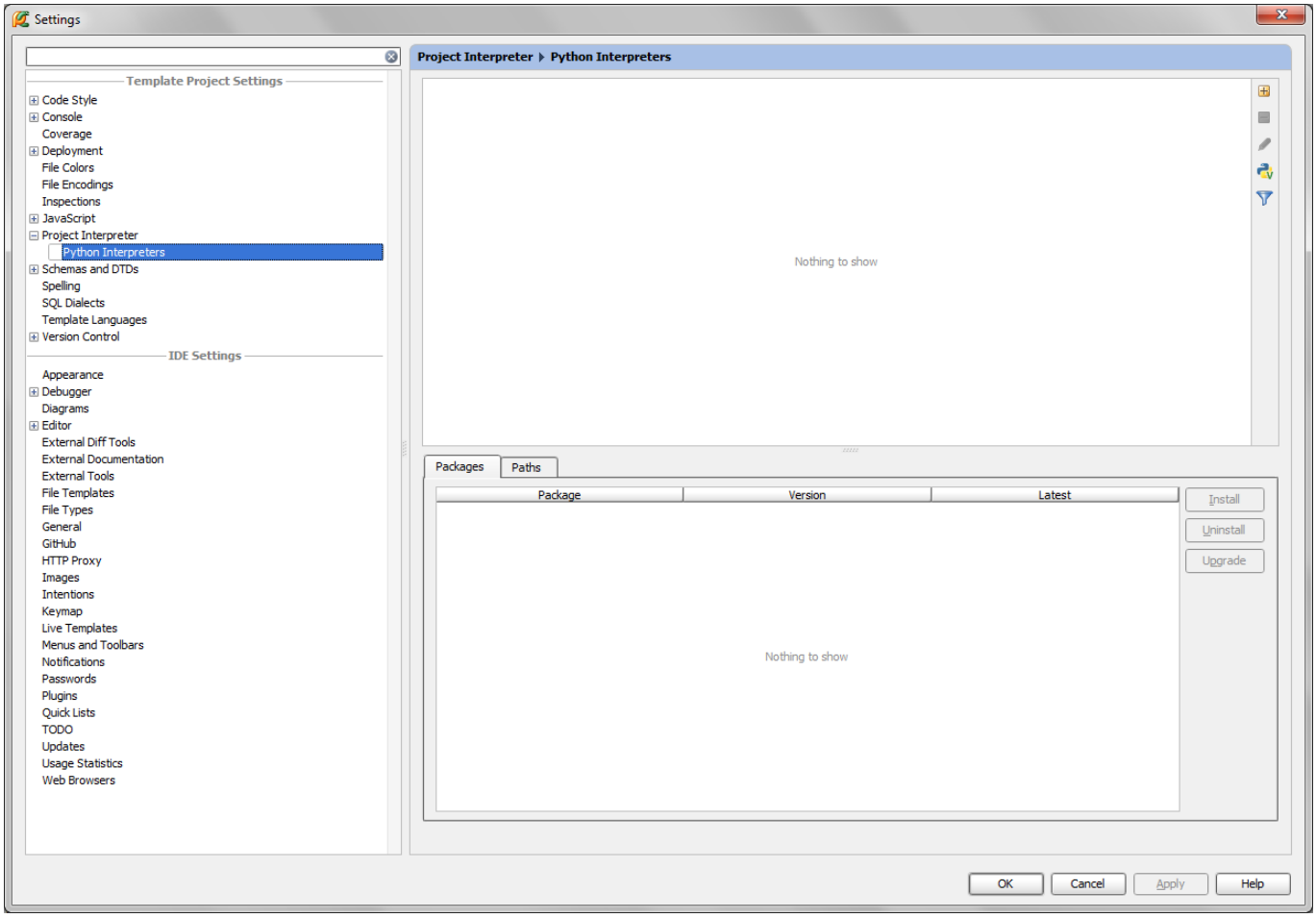
- formater sur l'ordinateur hôte une clé USB avec le système de fichier FAT
- démarrer la Carambola-Box, s'y connecter et ouvrir un terminal permettant d'y exécuter des commandes
- connecter la clé sur le port USB de la Carambola-Box
- dans le terminal de la Carambola-Box, exécuter la commande :
  ```
  mount -t vfat /dev/sda1 /mnt
  ```

La clé est maintenant accessible sous le répertoire /mnt.

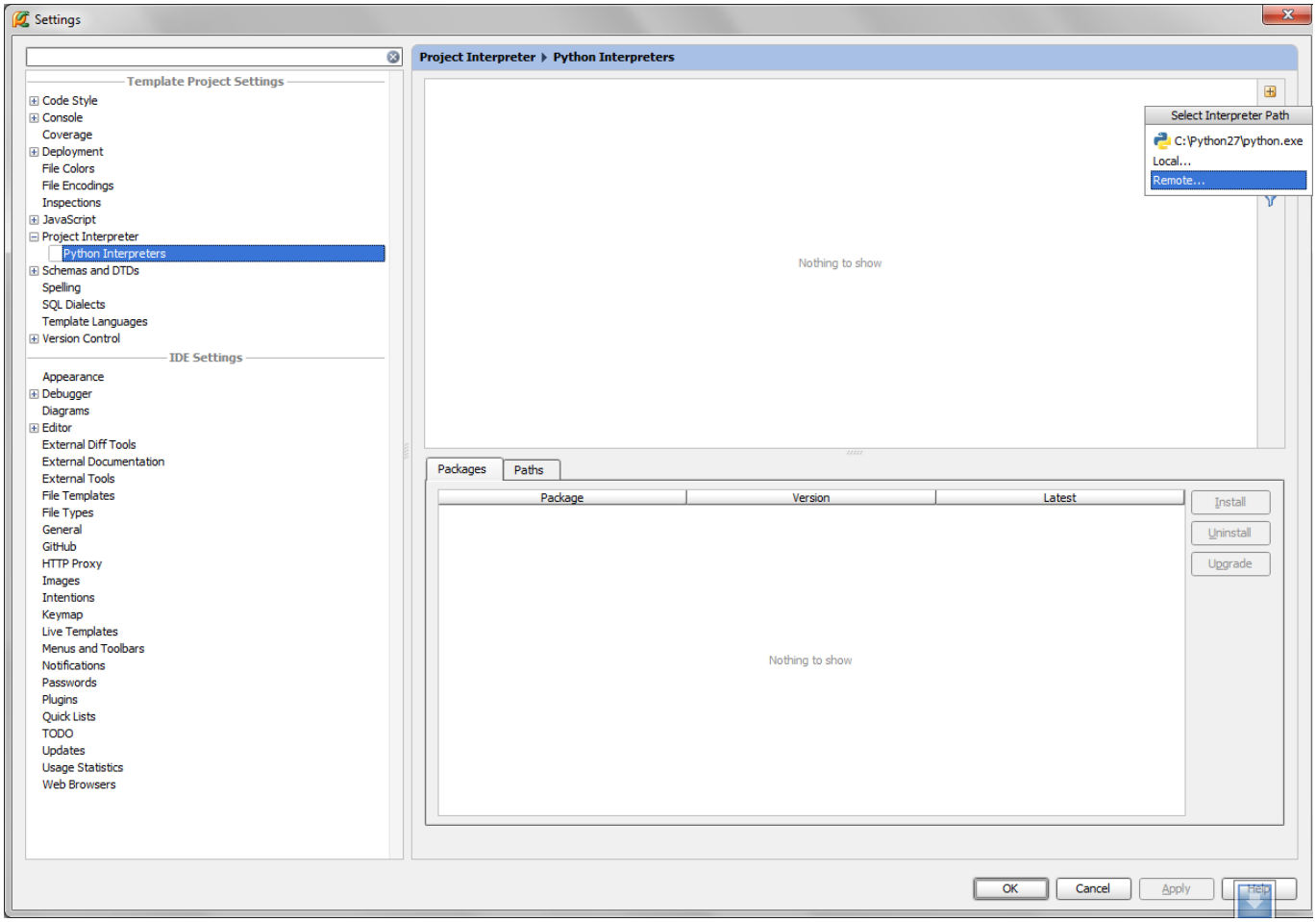
Nous pouvons ensuite configurer l'interpréteur Python. Sélectionner dans PyCharm le menu File → Settings :



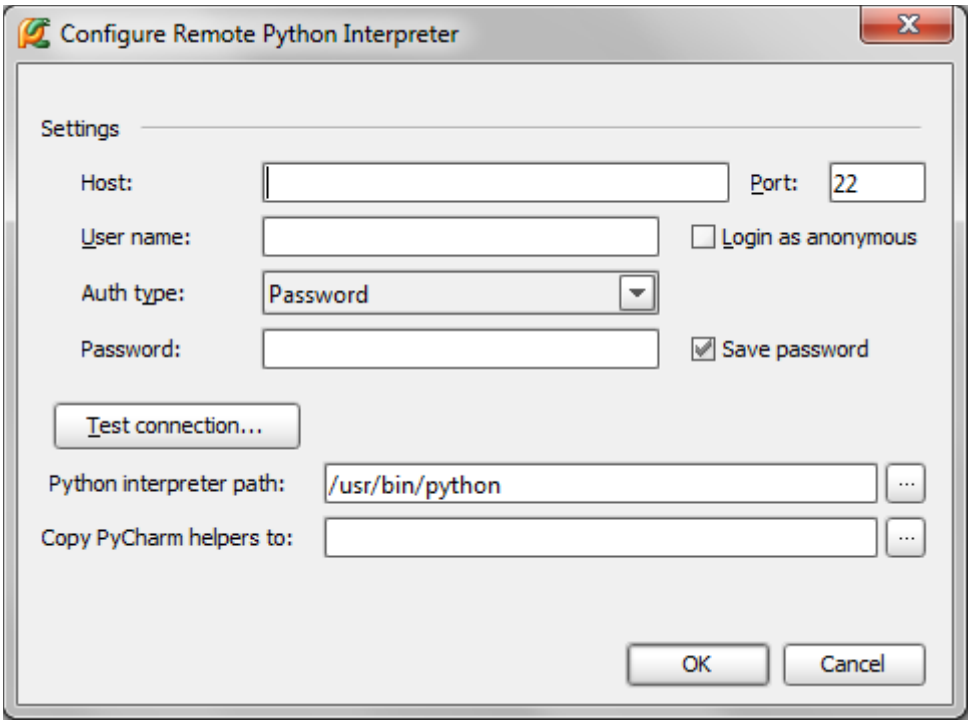
Il s'ouvre alors la fenêtre suivante, dans laquelle on sélectionnera Project Interpreters → Python Interpreters :



Cliquer ensuite sur le + en haut à droite de cette fenêtre, permettant de créer un lien vers un nouvel interpréteur, et choisir « Remote » :



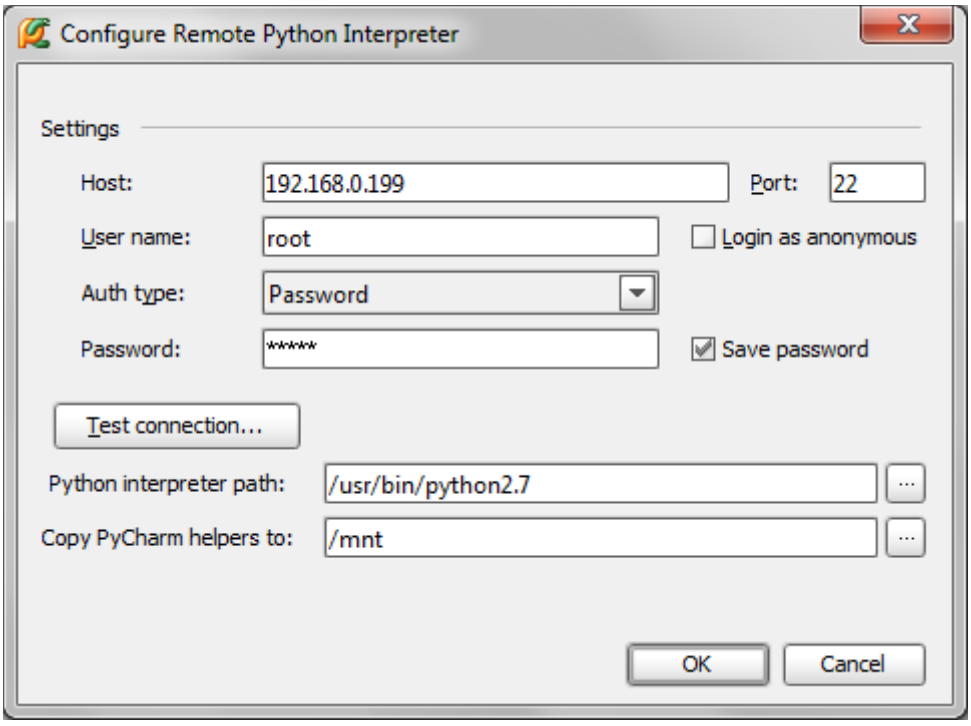
Il s'ouvre alors la fenêtre suivante :



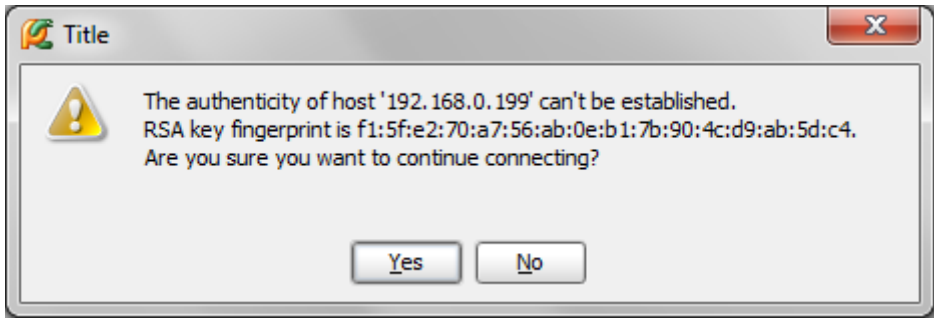
Nous allons remplir les différents champs avec les éléments suivants (nous donnons ci-dessous les valeurs par défaut pour un système juste livré) :

- Host : 192.168.0.199
- Port : 22
- User name : root
- Auth type : Password
- Password : admin
- Python interpreter path : /usr/bin/python2.7
- Copy PyCharm helpers to : /mnt

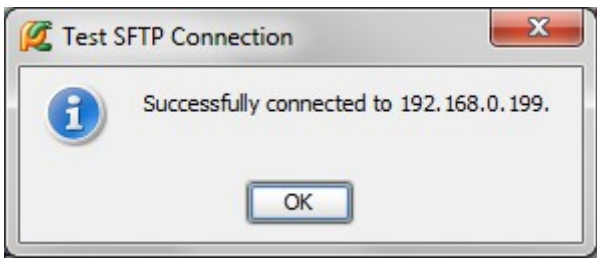
Voici la fenêtre une fois remplie correctement :



Un clic sur le bouton « Test Connection » fait apparaître le message suivant :

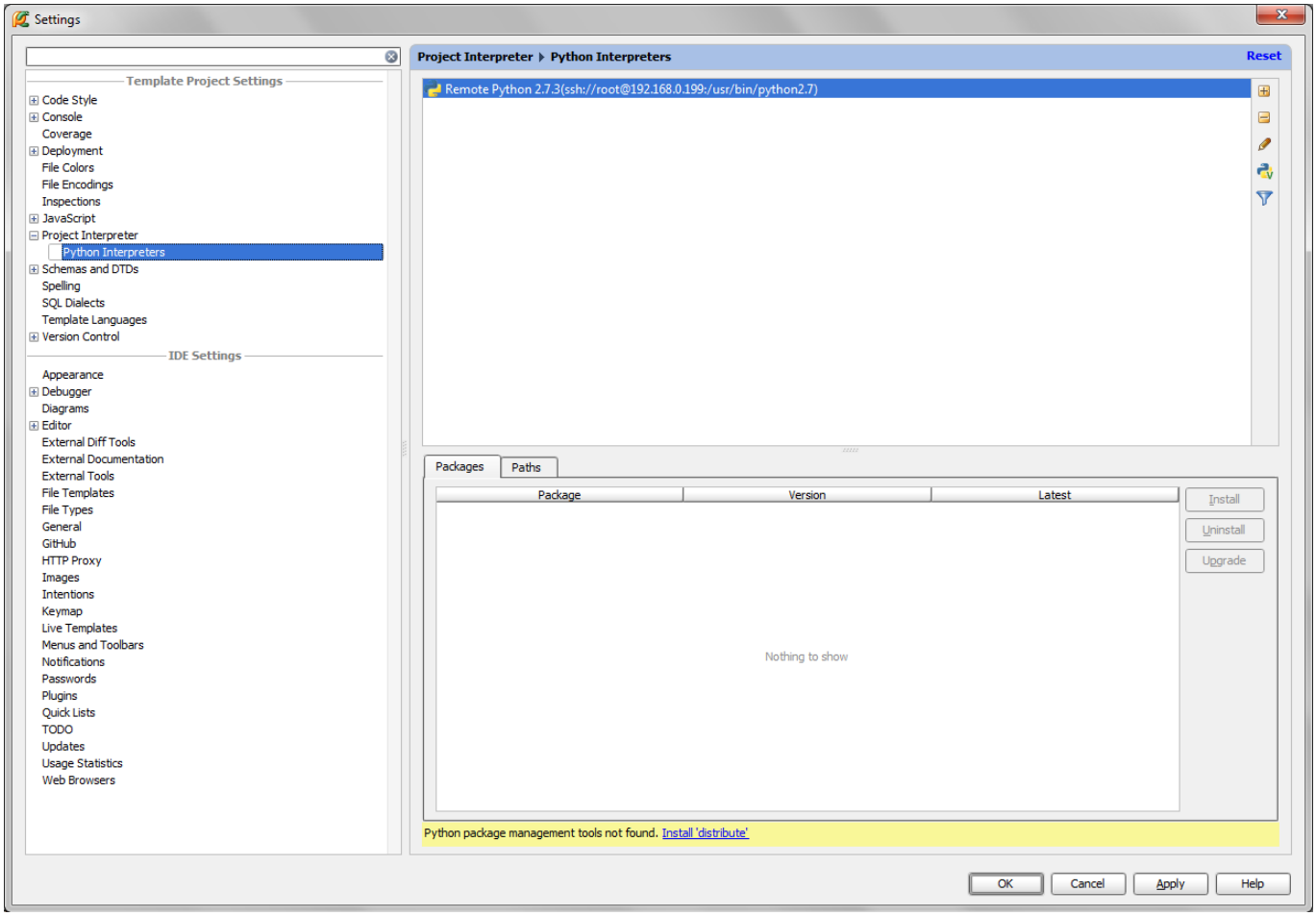


Cliquer sur « Yes », ce qui fait apparaître ce message :



Cliquer sur « OK », puis sur le bouton « OK » de la fenêtre de configuration de l'interpréteur Python.

Les fichiers utilitaires de PyCharm sont alors téléchargés sur la clé USB. On retrouve enfin la fenêtre des « Settings » avec ce nouvel interpréteur dans la liste :



Fermer cette fenêtre en cliquant sur « OK ».

Attention !
Cette configuration de l'interpréteur est maintenant permanente dans PyCharm, (sauf si vous la supprimez, bien entendu).

Par conséquent, si PyCharm doit être démarré alors que l'ordinateur hôte est connecté à la Carambola-Box, il est impératif d'y avoir branché et monté une clé USB avec la commande :

```
mount -t vfat /dev/sda1 /mnt
```

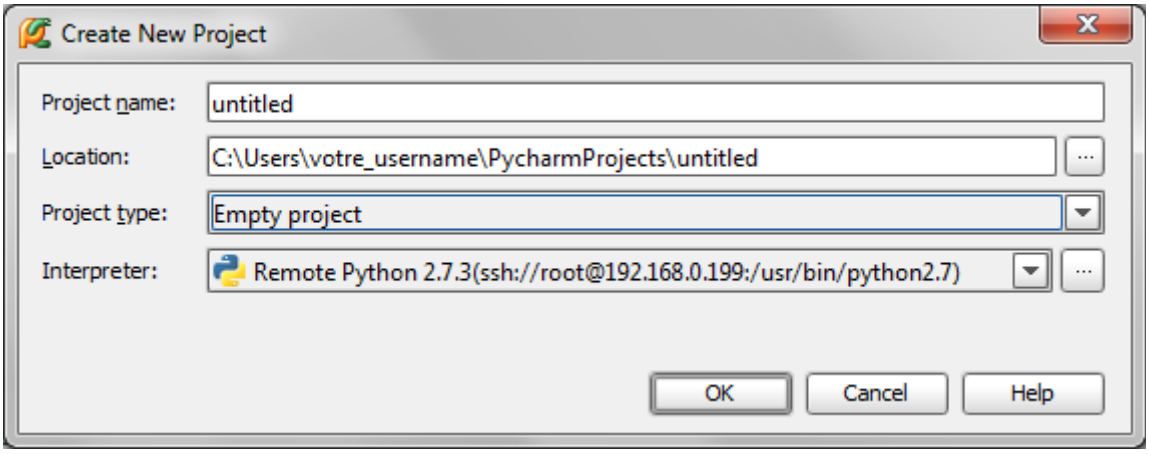
Si cela n'a pas été fait au préalable, PyCharm va copier ses fichiers de configuration sur la Carambola-Box et saturer sa mémoire. Pour rétablir la situation, il faudra :
- tuer la tâche « pycharm » dans le gestionnaire des tâches de l'ordinateur hôte
- effacer le contenu du répertoire /mnt sur la Carambola-Box

2.7.2 - Création d'un nouveau projet

La création d'un nouveau projet se fait sans difficulté :

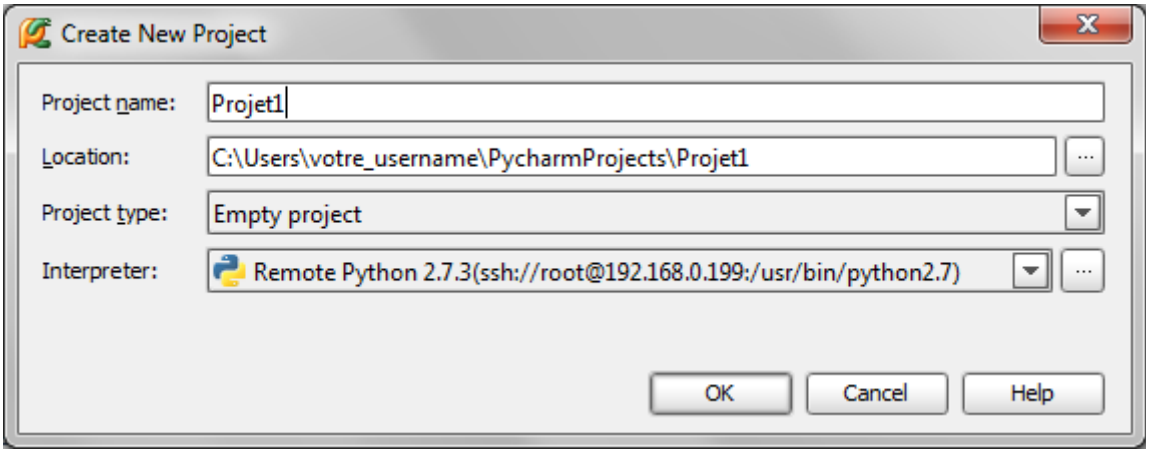
- soit en cliquant sur « Create New Project » dans la zone « Quick Start » de l'écran d'accueil
- soit en sélectionnant File → New Project

Il s'ouvre alors la fenêtre suivante :

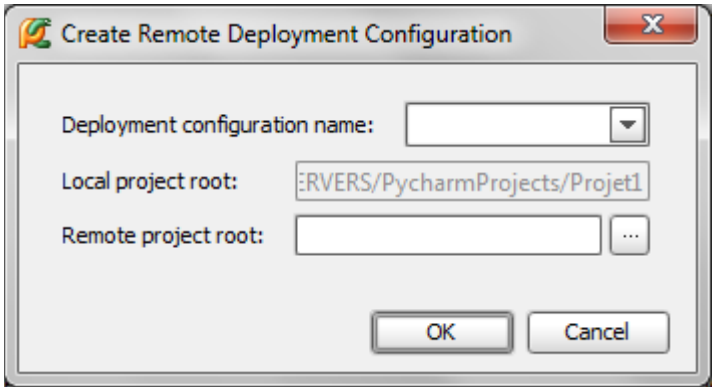


Dans la zone « Location », « votre_username » sera le nom d'utilisateur de votre machine, rempli par défaut.

Donner un nom au projet, par exemple Projet1, et ne rien modifier d'autre :



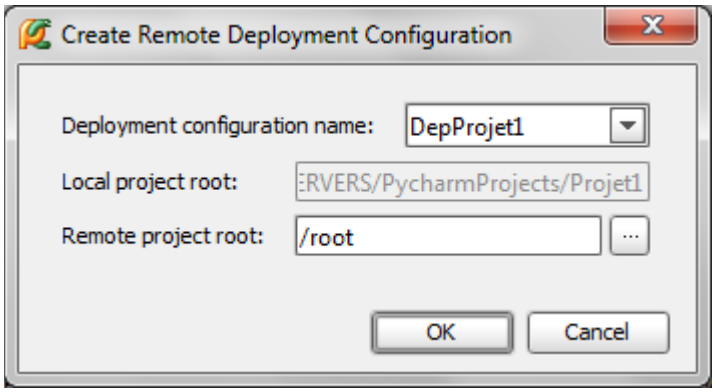
Puis cliquer sur « OK » . Il s'ouvre alors la fenêtre suivante :



Cette interface va nous permettre de spécifier une configuration de déploiement. En effet, n'oublions pas que tout ce que nous allons créer dans le projet sur notre ordinateur sera ensuite déployé et exécuter sur la cible (la Carambola-Box).

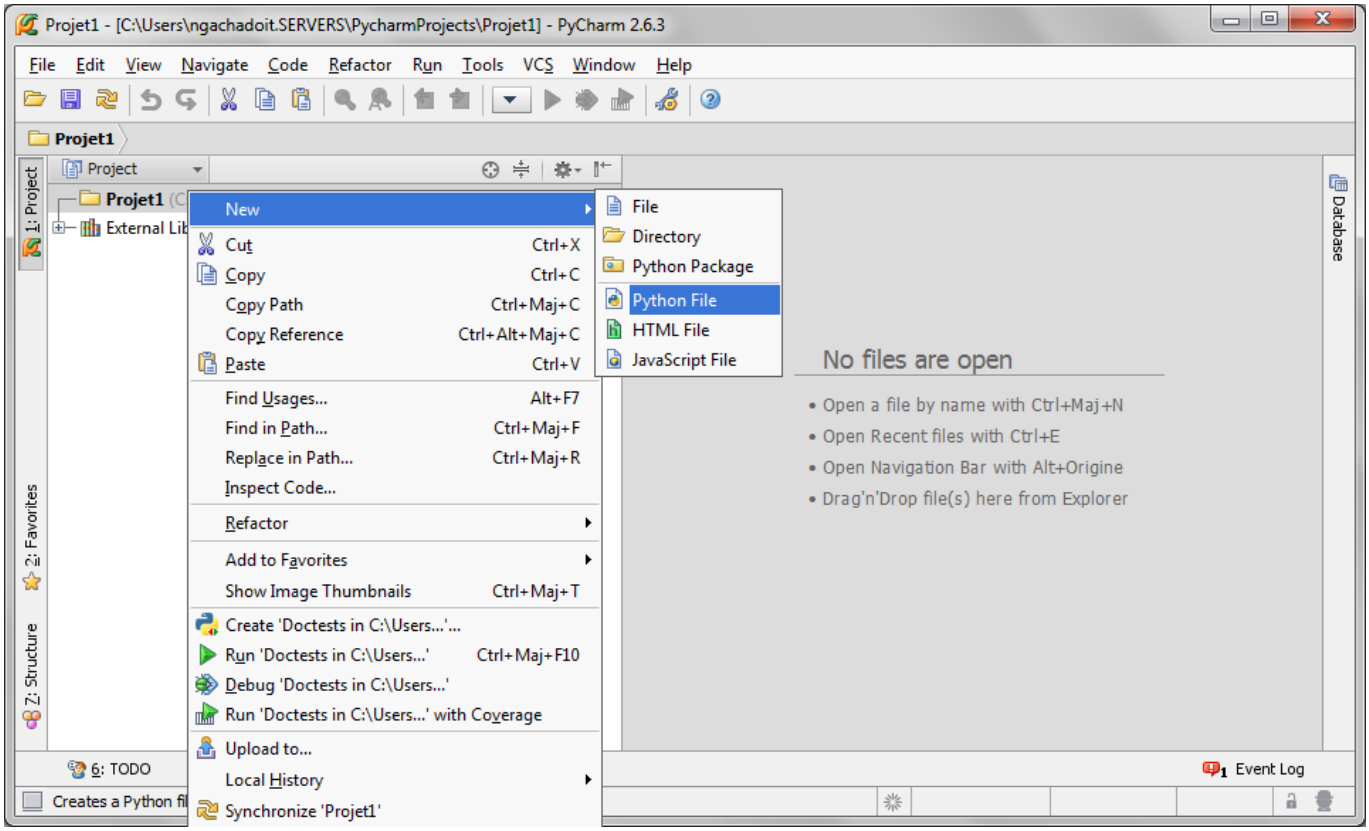
Remplir cette interface comme ceci :

- Deployment configuration name : DepProjet1
- Remote project root : /root

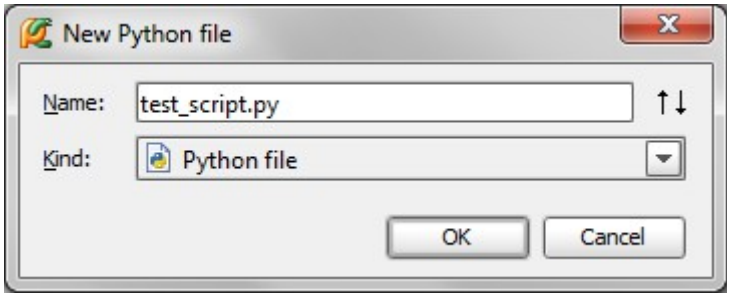


Puis cliquer sur « OK ».

La fenêtre principale de PyCharm s'ouvre alors. Faire un clic bouton droit sur « Projet1 » et choisir New → Python file:

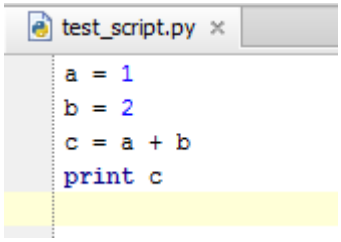


Nommer ce fichier test_script.py puis cliquer sur OK :

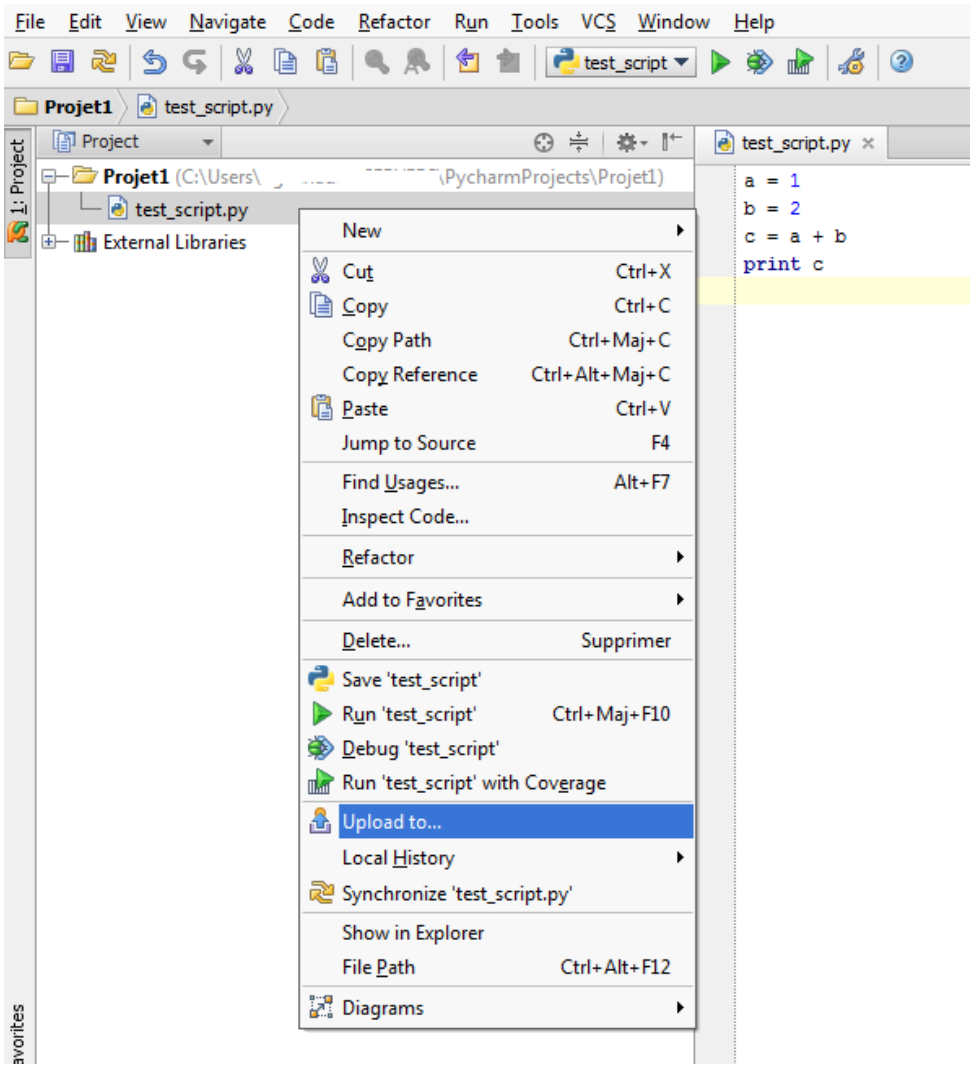


Ecrire dans la zone du fichier ce script très simple :

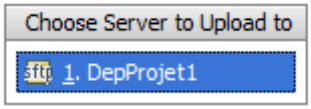
```
a = 1
b = 2
c = a + b
print c
```



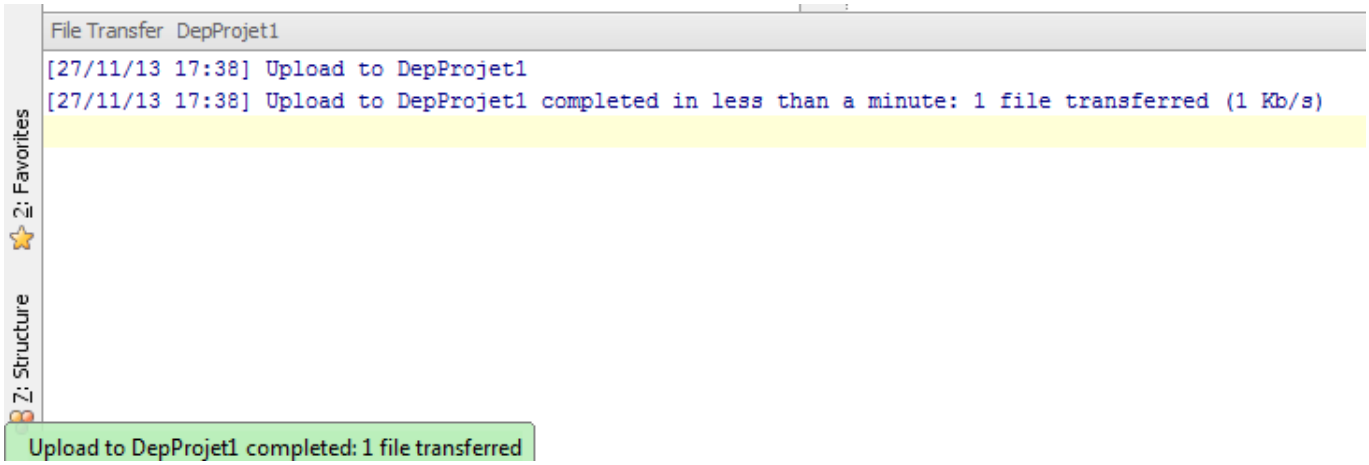
Pour l'instant, ce fichier n'est présent que sur la machine hôte. Avant de pouvoir l'exécuter sur la machine cible, nous devons l'y télécharger. Pour cela, faire un clic bouton droit sur le fichier test_script.py dans la zone du projet et sélectionner « Upload to » :



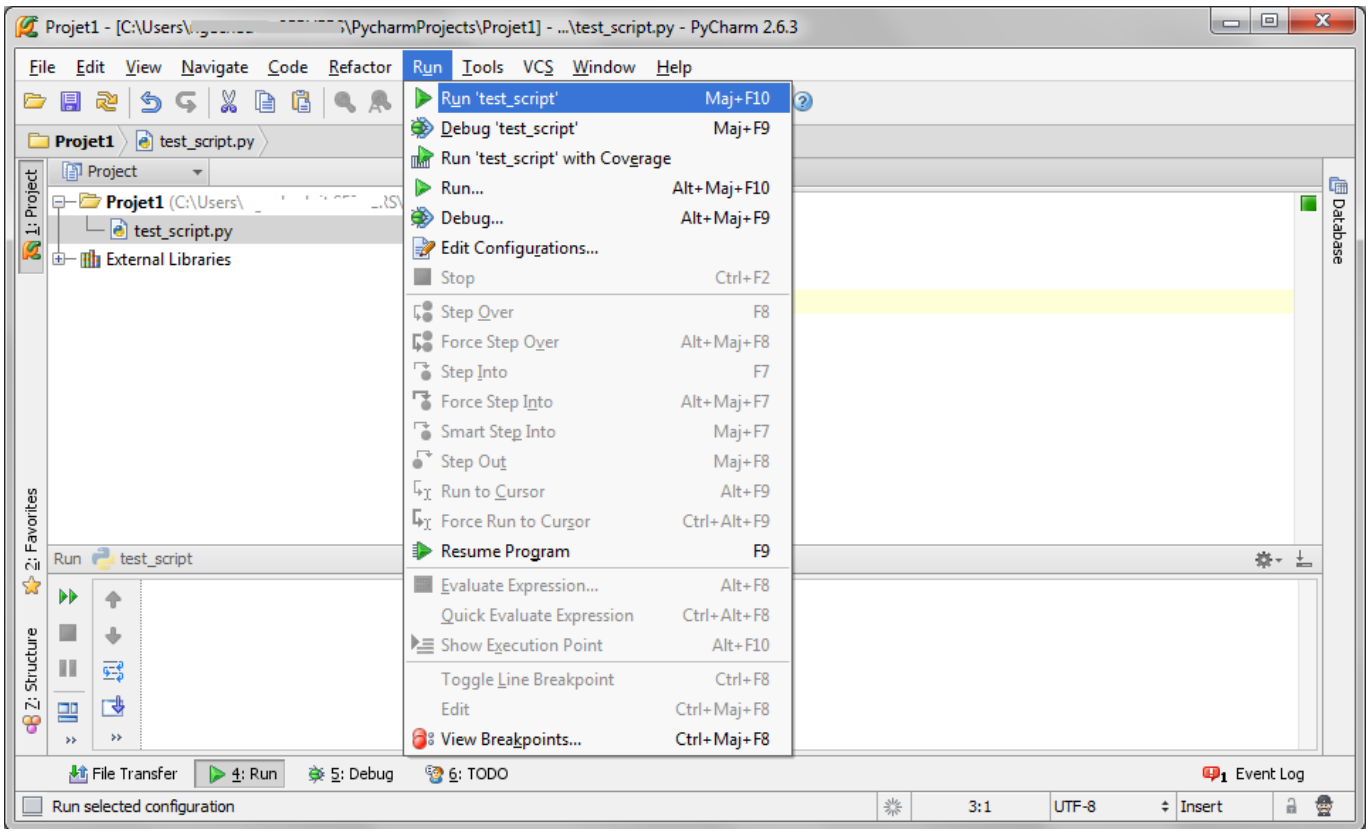
Choisir ensuite le serveur de déploiement en cliquant sur DepProjet1 :



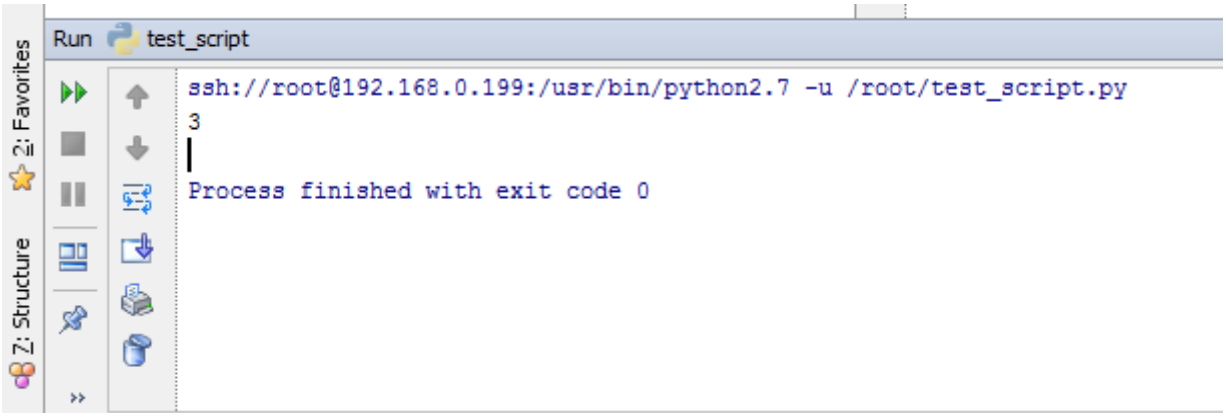
Le fichier est alors téléchargé sur la cible, comme l'indique la zone de log en bas de la fenêtre :



Le script peut maintenant être exécuté en le sélectionnant dans la zone du projet et en sélectionnant le menu Run → Run 'test_script' :



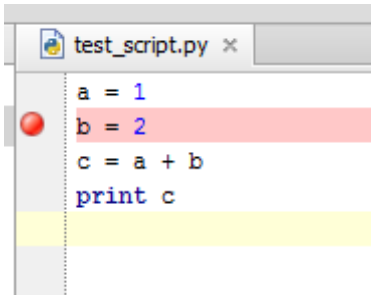
La fenêtre de log montre que le script a bien été exécuté sur la cible et indique que tout s'est bien passé :



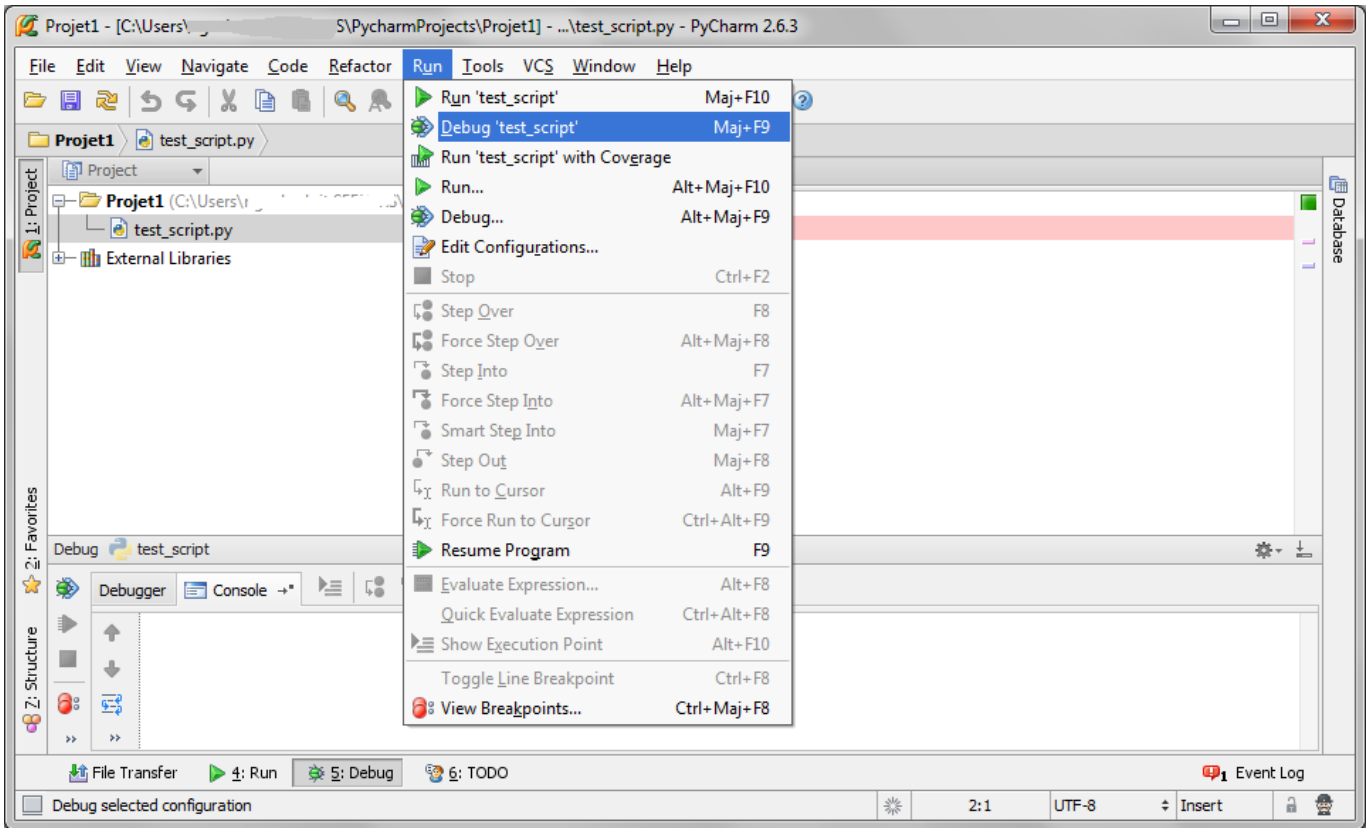
2.7.3 - Débuggage sur la cible

PyCharm permet non seulement l'exécution sur la cible mais également le débuggage.

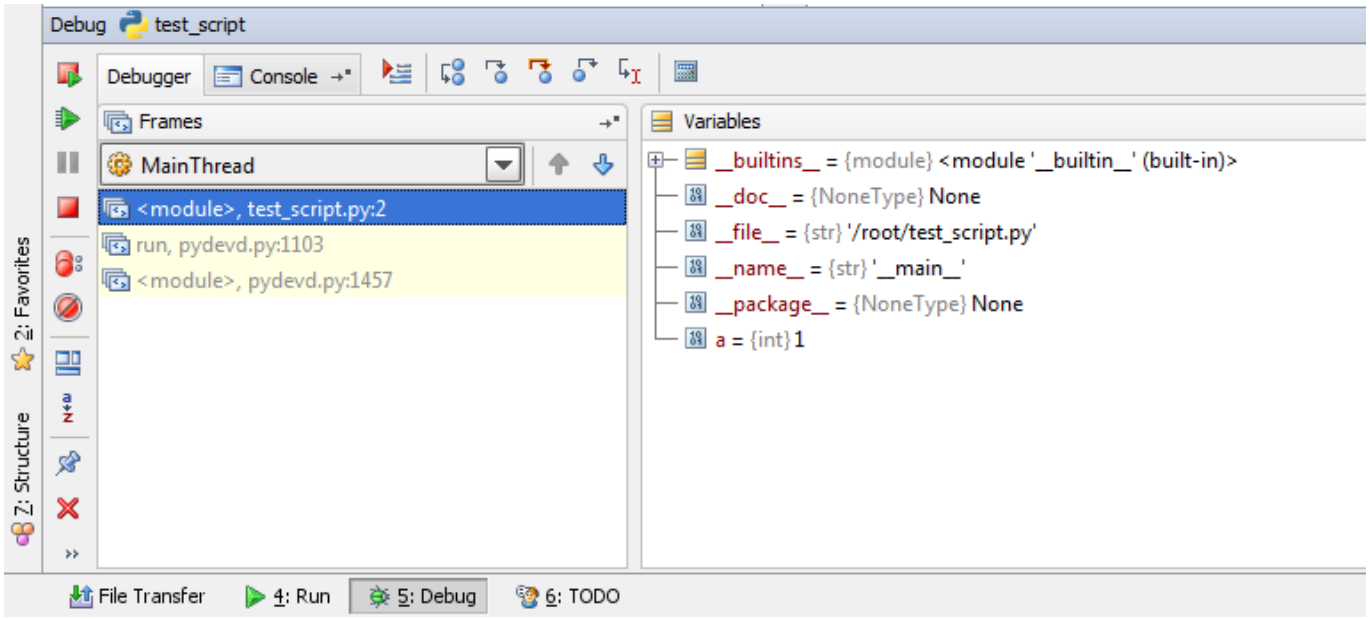
Avant de lancer l'exécution avec le débuggeur, il est nécessaire de placer au moins un point d'arrêt dans le code. Ceci se fait en cliquant dans la colonne grise à gauche de l'éditeur pour faire apparaître un rond rouge (symbole de « stop »), comme ici sur la seconde ligne du script :



Il reste ensuite à lancer le débuggeur en sélectionnant le fichier dans la zone du projet et en exécutant le menu Run → Debug 'test_script' :

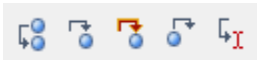


La zone « Debugger » apparaît alors en bas de le fenêtre de PyCharm :



On constate que test_script.py est surligné dans la partie gauche et que le point d'arrêt courant est sur la ligne 2. Dans la partie droite, on peut voir que a est affecté (à 1) puisque le point d'arrêt se trouve sur la ligne suivante.

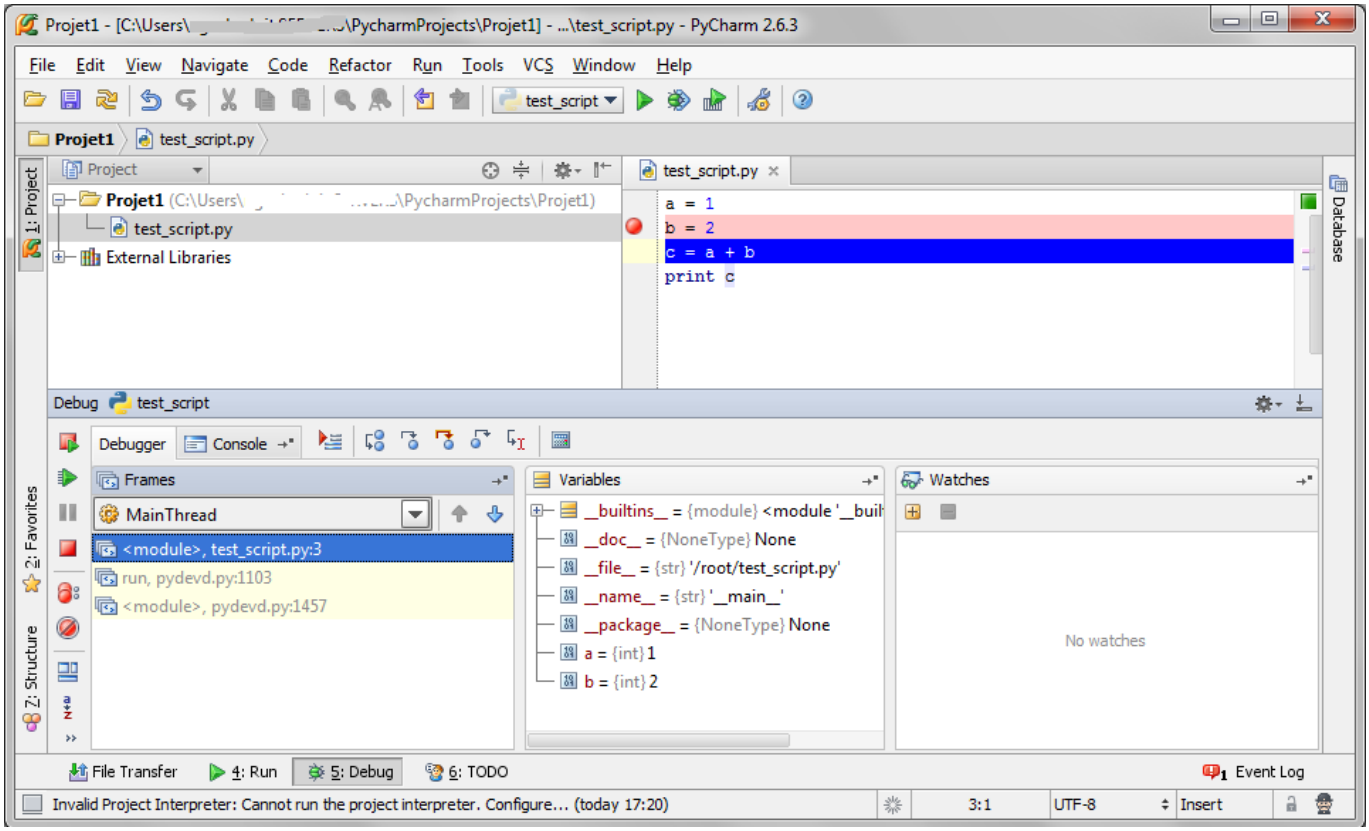
On peut alors utiliser les icônes du debuggeur :



Elles permettent respectivement de :

- passer à l'instruction suivante sans entrer dans un éventuel sous-programme
- passer à l'instruction suivante en entrant dans un éventuel sous-programme
- passer à l'instruction suivante en forçant l'entrée dans un éventuel sous-programme
- passer à l'instruction suivante en sortant d'un éventuel sous-programme
- passer à l'instruction sur laquelle est positionné le curseur dans l'éditeur du script

Pour ce simple script, un clic sur la première icône permet d'exécuter la ligne sur laquelle était positionné le point d'arrêt et de s'arrêter à l'instruction suivante :



On voit que b est maintenant affecté puisqu'il a été exécuté et que la ligne sur laquelle est arrêté le débuggeur est la 3 (surlignée en bleu dans l'éditeur).

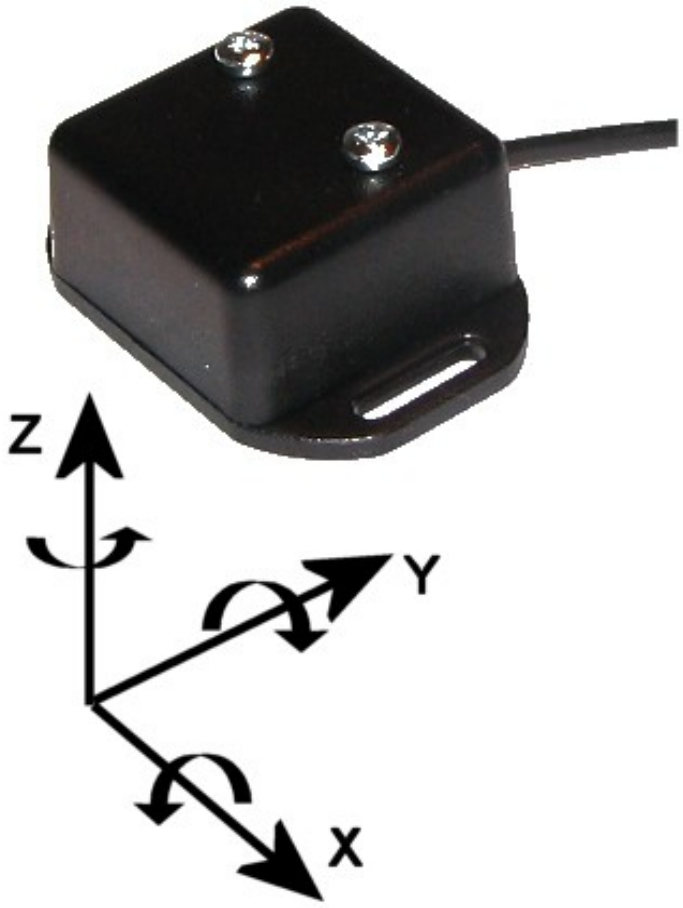
Pour interrompre le débuggage et terminer l'exécution du programme, cliquer sur le triangle vert dans la colonne à gauche de la zone du debuggeur :

Attention ! Ne pas cliquer sur le carré rouge :

En effet, cela arrête le débuggeur côté PyCharm mais pas côté cible, ce qui rend ensuite impossible le lancement d'une nouvelle session de debuggage, à moins de tuer la tâche correspondante sur la Carambola-Box.

3 - Le boîtier IMU connecté en i2c

L'IMU (Inertial Measurement Unit en anglais, soit Unité de Mesure Inertielle) permet de mesurer les accélérations et les vitesses de rotation autour de 3 axes perpendiculaires. Ce boîtier fournit donc 6 mesures via une liaison i2c.



Le schéma ci-dessus montre l'orientation des axes de mesure du boîtier. Pour éviter toute erreur d'interprétation de ce schéma, précisons le sens de mesure des vitesses de rotation :

- autour de x : positif lorsque y tourne vers z
- autour de y : positif lorsque z tourne vers x
- autour de z : positif lorsque x tourne vers y

Ce système de mesure est basé sur le composant MPU6050 d'Invensense, qui intègre tout (y compris un capteur de température et un filtre passe-bas réglable par programme sur les mesures inertielles) sur une puce de 16 mm².

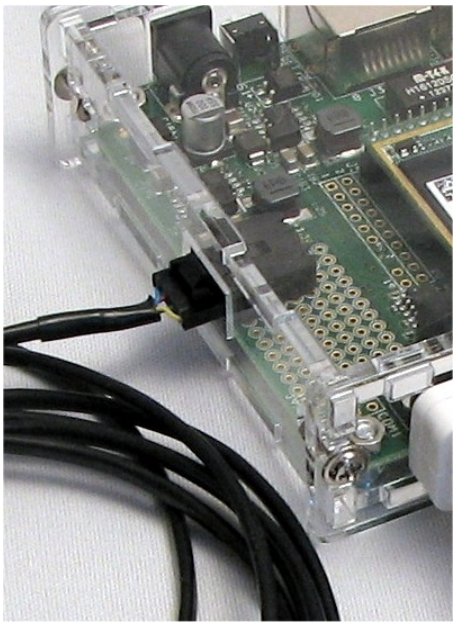
La liaison i2c est un bus série filaire permettant de transporter des signaux sur 2 fils uniquement (4 en pratique si on ajoute l'alimentation et la masse). Il est possible de connecter plusieurs composants sur un seul bus i2c, chaque composant étant référencé par une adresse. Par exemple, l'adresse du MPU6050 est 0x68.

La communication entre la Carambola-Box et le boîtier IMU en i2c est gérée par différentes bibliothèques de fonctions écrites en Python (dans /root/IMU/Python) :

- mpu6050.py : fonctions d'initialisation et de lecture de données (accélérations, vitesses de rotation, température). Normalement, un script Python interagissant avec le MPU6050 ne doit pas utiliser d'autres fonctions que celles présentes dans cette bibliothèque
- i2c.py : fonctions de bas niveau de gestion du bus i2c
- carambol_IO.py : bibliothèque de fonctions intermédiaires entre les deux précitées. Elle apporte un niveau d'abstraction supérieur aux fonctions de i2c.py sans être dédiée spécifiquement au MPU6050
- dechex.py : bibliothèque d'utilitaires de conversion décimal ↔ hexadécimal

Le langage Python a été choisi pour ses aspects didactiques. D'autres langages auraient également pu être utilisés.

Avant toute manipulation nécessitant le boîtier IMU, il est bien sûr nécessaire de connecter celui-ci sur le connecteur adéquat de la Carambola-Box :



4 - Fonctionnalités exécutées par la Carambola-Box

La Carambola-Box est un véritable ordinateur embarqué qui peut exécuter de nombreuses fonctionnalités. Celles décrites ci-dessous sont préchargées sur le système à la livraison mais vous pouvez bien sûr en développer de nouvelles, en fonction de vos besoins et de votre créativité.

4.1 - Script Python : calibration des mesures de l'IMU

Une fois connecté en SSH à la Carambola-Box, un programme Python permet de calibrer l'IMU qui se branche sur le connecteur i2c du système.

La calibration permet de supprimer tous les offsets de mesure en considérant que la position courante de l'IMU est la position à laquelle les 3 accélérations et les 3 vitesses de rotation sont nulles.

Les deux applications principales (mutuellement exclusives) sont les suivantes :

- Suppression des erreurs d'offset
- Initialisation d'une position de référence

ATTENTION !
Avant de réaliser ce qui suit, le boîtier IMU doit être branché à la Carambola-Box.

La procédure de calibration est la suivante :

- Placer le boîtier dans sa position que vous souhaitez choisir comme référence, immobile
- Pour lancer ce programme :
  - se déplacer dans /root/IMU/Python :

```
cd /root/IMU/Python
```

  - exécuter la commande :

```
python CalibrationIMU.py
```

La calibration est très rapide. Lorsqu'elle est terminé, le message « Calibration terminée » apparaît dans le terminal.

4.2 - Script Python : affichage des mesures de l'IMU

Une fois connecté en SSH à la Carambola-Box, un programme Python permet d'afficher les données mesurées par l'IMU suivant ses 6 axes.

ATTENTION !
Avant de réaliser ce qui suit, le boîtier IMU doit être branché à la Carambola-Box.

Pour lancer ce programme :

- se déplacer dans /root/IMU/Python :

```
cd /root/IMU/Python
```

- exécuter la commande :

```
python afficherDataIMU.py
```

Les données mesurées défilent alors à l'écran sur 7 colonnes dans l'ordre suivant :

- temps (s) depuis le lancement de la commande
- accélération suivant l'axe x (m/s²)
- accélération suivant l'axe y (m/s²)
- accélération suivant l'axe z (m/s²)
- vitesse de rotation autour de l'axe x (rad/s)
- vitesse de rotation autour de l'axe y (rad/s)
- vitesse de rotation autour de l'axe z (rad/s)

Pour arrêter l'affichage, la solution la plus simple consiste à interrompre brutalement le programme en tapant CTRL+C dans le terminal.

Les données ne sont pas échantillonnées à cadence fixe. Elles s'affichent aussi rapidement que possible.

Il est possible de les stocker dans un fichier texte en redirigeant la sortie de la commande de cette façon :

```
python afficherDataIMU.py > capture.txt
```

Dans ce cas, les données ne s'affichent plus directement sur le terminal SSH mais sont stockées dans le fichier « capture.txt ». Ce fichier peut ensuite être rapatrié sur l'ordinateur hôte et/ou être affiché dans le terminal SSH avec la commande suivante :

```
cat capture.txt
```

Lorsque le fichier de mesure est très long, il est intéressant de l'afficher page par page :

```
cat capture.txt | more
```

Le passage d'une page à l'autre se fait en pressant la barre espace. Pour quitter le mode d'affichage paginé et revenir au prompt du terminal, taper « q ».

Attention !
Si vous prévoyez d'enregistrer des données sur une longue durée, il est préférable de les stocker directement sur une clé USB. Suivez pour cela la procédure ci-dessous.

Pour accéder au contenu d'une clé USB depuis la Carambola-Box, vous devez tout d'abord monter cette clé sur le système. La commande à exécuter dépend du système de fichier avec lequel a été formatée la clé :

- FAT : exécuter la commande : `mount -t vfat /dev/sda1 /mnt`
- NTFS: exécuter la commande : `mount -t ntfs /dev/sda1 /mnt`
- EXT4: exécuter la commande : `mount -t ext4 /dev/sda1 /mnt`

Pour que la clé soit utilisable sur différents OS, il est préférable de la formater avec le système « FAT ».

Une fois la clé branchée sur la Carambola-Box et montée sur le système, le stockage des données peut se faire avec la commande suivante :

```
python afficherDataIMU.py > /mnt/capture.txt
```

Après extraction de la clé, il est alors possible de lire directement les données sur un ordinateur.

Attention !
Extraire proprement la clé USB avec la commande suivante :
`umount /dev/sda1`

Enfin, il est également possible de rediriger les données sur le port série de la Carambola-Box avec la commande suivante :

```
python afficherDataIMU.py > /dev/ttyS1
```

Ceci permet de récupérer les données sur n'importe quel matériel possédant une interface RS232 et connecté sur le port série de la Carambola-Box.

4.3 - Script Python : stockage des mesures de l'IMU dans une base de données SQL

Une fois connecté en SSH à la Carambola-Box, un programme Python permet de stocker les données mesurées par l'IMU suivant ses 6 axes dans une base de données SQL (sqlite3, pour être précis).

ATTENTION !
Avant de réaliser ce qui suit, le boîtier IMU doit être branché à la Carambola-Box.

Pour lancer ce programme :

- se déplacer dans /root/IMU/Python :

```
cd /root/IMU/Python
```

- exécuter une commande de la forme :

```
python data2DB.py <nom_de_la_base_de_donnees> <duree_d_enregistrement>
```

Par exemple, pour enregistrer les données dans la base « imu_data.db » pendant 10 secondes, il faut exécuter la commande :

```
python data2DB.py imu_data.db 10
```

La base de données peut ensuite être rapatriée sur l'ordinateur hôte, mais il est également possible de visualiser son contenu directement dans le terminal SSH, grâce à la séquence de commandes suivante :

- lancement de l'application de gestion de base de données sur la base « imu_data.db » :

```
sqlite3 imu_data.db
```

- Le terminal fait alors apparaître le prompt sqlite :

```
sqlite>
```

- liste des tables (ne pas oublier le point au début de chaque commande) :

```
.tables
```

- configuration de l'affichage pour que celui-ci soit le plus propre possible :

```
.mode column
.headers on
```

- exécution de la requête SQL permettant d'afficher les données mesurées (stockées dans la table « Data1 »). Cette fois-ci la commande ne commence pas par un point car c'est une requête SQL et non pas une commande de l'utilitaire sqlite. En revanche, ne pas oublier le « ; » à la fin de la requête

```
SELECT * FROM Data1;
```

- pour quitter sqlite3 :

```
.exit
```

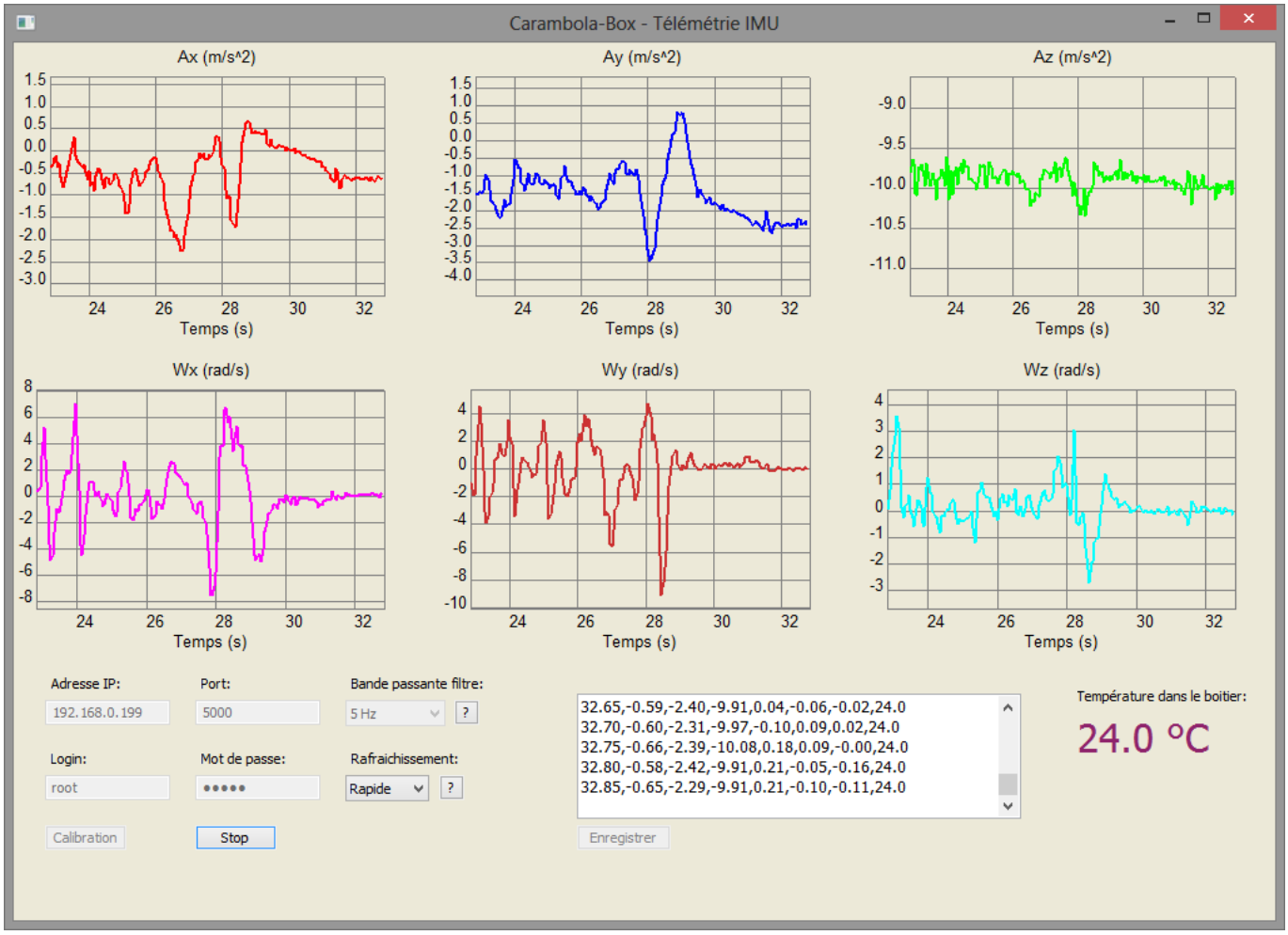
Les données mesurées s'affichent alors proprement à l'écran sur 7 colonnes dans l'ordre suivant :

- temps (s) depuis le lancement de la commande
- accélération suivant l'axe x (m/s²)
- accélération suivant l'axe y (m/s²)
- accélération suivant l'axe z (m/s²)
- vitesse de rotation autour de l'axe x (rad/s)
- vitesse de rotation autour de l'axe y (rad/s)
- vitesse de rotation autour de l'axe z (rad/s)

Les données sont échantillonnées à cadence fixe toutes 0.1 s.

4.4 - Script et interface graphique Python : télémétrie des mesures de l'IMU et affichage des courbes sur l'ordinateur hôte

Cette application permet de visualiser sur l'ordinateur hôte les données mesurées sur les 6 axes de l'IMU, ainsi que sa température :



Ce script est téléchargeable dans la section « Carambola-Box » de notre page de téléchargements :

http://www.3sigma.fr/Telechargements.html

Il nécessite les bibliothèques Python suivantes :

- wxPython (http://www.wxpython.org/download.php)
- Numpy (http://sourceforge.net/projects/numpy/files/)
- Crypto (https://pypi.python.org/pypi/pycrypto)
- Paramiko (https://pypi.python.org/pypi/paramiko/1.10.1)

L'application fonctionne en liaison avec le script TelemetrieIMU.py qui se trouve dans le répertoire /root/IMU/Python de la Carambola-Box.

ATTENTION !
Avant de réaliser ce qui suit, le boîtier IMU doit être branché à la Carambola-Box.

La procédure préliminaire à suivre avant de lancer l'interface est la suivante :

- connecter le boîtier IMU sur le port i2c de la Carambola-Box
- démarrer la Carambola-Box (brancher son alimentation)
- rejoindre le réseau Wifi du système (la mise en réseau peut également se faire via un câble Ethernet)

A ce stade, tout est prêt pour lancer l'interface. Mais avant de pouvoir visualiser les accélérations et vitesses de rotation suivant les 3 axes, il reste à compléter les étapes suivantes :

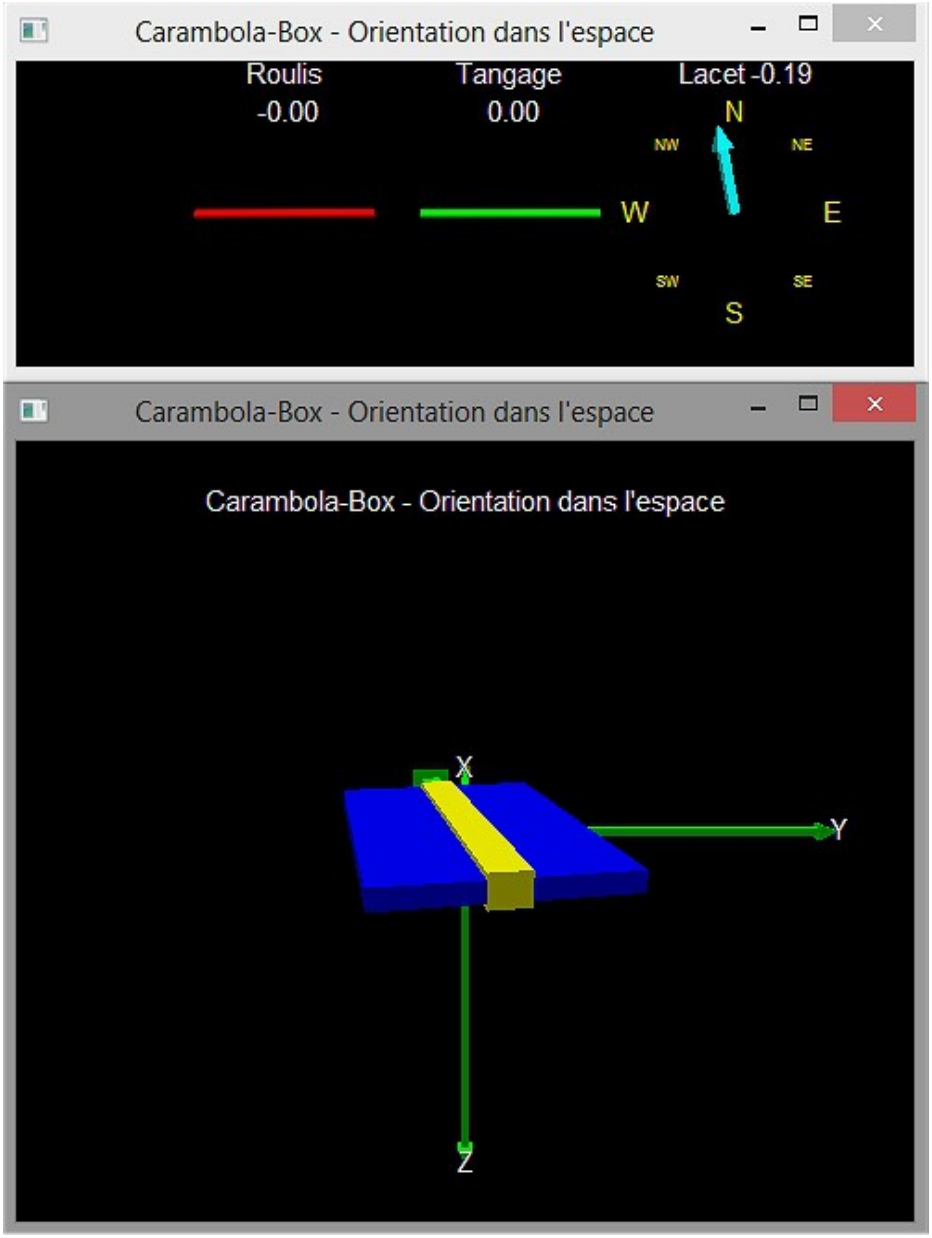
- vérifier que l'adresse IP, le login et le mot de passe correspondent aux informations qui vous permettent de vous connecter en SSH au système (voir paragraphe 2.2)
- le numéro du port ne doit pas être modifié, sauf si vous répercutez la même modification dans le script TelemetrieIMU.py qui se trouve dans le répertoire /root/IMU/Python de la Carambola-Box
- éventuellement, lancer une calibration en cliquant sur le bouton « Calibration ». Cette action exécute à distance le script CalibrationIMU.py qui se trouve dans le répertoire /root/IMU/Python de la Carambola-Box. Attendre le message indiquant que la calibration est terminée dans la fenêtre de log
- éventuellement, spécifier la bande passante du filtre passe-bas intégré sur le MPU6050. Ce filtre permet de réduire le bruit de mesure
- éventuellement, modifier la fréquence de rafraîchissement des courbes. Sur un ordinateur peu performant, il peut être nécessaire d'utiliser un rafraîchissement « Lent » ou « Minimum » pour que les mesures s'affichent en temps-réel
- cliquer sur le bouton « Acquisition » : les mesures s'affichent alors sur les différents graphiques et dans la fenêtre de log

Pour arrêter l'acquisition, cliquer sur le bouton « Stop ». Il est alors possible de cliquer sur le bouton « Enregistrer » pour stocker les données mesurées dans un fichier sur l'ordinateur hôte.

ATTENTION !
Il est possible de vous obteniez un message d'erreur lors du lancement de la première acquisition après le démarrage de la Carambola-Box. Dans ce cas, cliquez de nouveau sur le bouton « Acquisition ».
Si le problème persiste, veuillez vérifier vos paramètres de connexion.

4.5 - Script et interface graphique Python : télémétrie des mesures de l'IMU et affichage 3D sur l'ordinateur hôte

Cette application permet de visualiser sur l'ordinateur hôte l'orientation dans l'espace de la Carambola-Box grâce à un traitement des données mesurées sur les 6 axes de l'IMU pour les convertir en angles d'Euler :



Ce script est téléchargeable dans la section « Carambola-Box » de notre page de téléchargements :

http://www.3sigma.fr/Telechargements.html

Il nécessite les bibliothèques Python suivantes :

- Vpython (http://www.vpython.org/)
- Crypto (https://pypi.python.org/pypi/pycrypto)
- Paramiko (https://pypi.python.org/pypi/paramiko/1.10.1)

L'application fonctionne en liaison avec le script TelemetrieIMU_noTime.py qui se trouve dans le répertoire /root/IMU/Python de la Carambola-Box.

ATTENTION !
Avant de réaliser ce qui suit, le boîtier IMU doit être branché à la Carambola-Box.

La procédure préliminaire à suivre avant de lancer l'interface est la suivante :

- connecter le boîtier IMU sur le port i2c de la Carambola-Box
- démarrer la Carambola-Box (brancher son alimentation)
- rejoindre le réseau Wifi du système (la mise en réseau peut également se faire via un câble Ethernet)

A ce stade, tout est prêt pour lancer l'interface. Mais avant toute chose, il reste à compléter les étapes suivantes :

- vérifier au début du script (juste après les imports) que l'adresse IP, le login et le mot de passe correspondent aux informations qui vous permettent de vous connecter en SSH au système (voir paragraphe 2.2)
- le numéro du port ne doit pas être modifié, sauf si vous répercutez la même modification dans le script TelemetrieIMU_noTime.py qui se trouve dans le répertoire /root/IMU/Python de la Carambola-Box

Vous pouvez maintenant lancer ce script et modifier l'orientation dans l'espace du boîtier IMU pour visualiser cette orientation en temps-réel sur l'écran de l'ordinateur.

Notez que la bande passante du filtre sur les mesures de l'IMU est positionnée au minimum (5 Hz). Le filtrage des bruits de mesure est donc maximum, pour ne pas qu'il perturbe la visualisation.

ATTENTION !
Cette application démarre systématiquement par une procédure de calibration. Il est donc important de laisser le système immobile tant que la visualisation n'apparaît pas sur l'écran de l'ordinateur.

ATTENTION !
L'orientation dans l'espace du boîtier IMU est calculée sur la base de la mesure des vitesses de rotation, recalées avec la mesure de la pesanteur, dans l'hypothèse de l'absence de toute autre accélération. Par conséquent, un choc (exemple typique de perturbation sur la mesure de l'accélération de la pesanteur) conduira à biaiser la correspondance entre l'orientation réelle du boîtier et la visualisation.

ATTENTION !
Il est possible de vous obteniez un message d'erreur lors du lancement de l'application après le démarrage de la Carambola-Box. Dans ce cas, veuillez la lancer de nouveau.
Si le problème persiste, veuillez vérifier vos paramètres de connexion.

4.6 - Transmission vidéo temps-réel

La version la plus complète de la Carambola-Box inclut une Webcam qui se connecte sur le port USB du boîtier.

Les images prises par cette Webcam peuvent être envoyées en temps-réel dans un flux mjpeg à un ordinateur connecté au boîtier en Wifi.

La commande à exécuter pour démarrer le flux vidéo est la suivante:

```
mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f 25" -o "output_http.so -p 8080 -w /www/webcam" &
```

Le dernier caractère de cette ligne de commande (« & ») permet de lancer cette tâche en arrière-plan, sans bloquer la ligne de commande.

La commande mjpg_streamer ci-dessus permet d'envoyer un flux de 25 images par seconde, de résolution 640x480, sur le port 8080 de la Carambola-Box.

ATTENTION !
Il est impératif que la Webcam soit branchée sur le port USB de la Carambola-Box avant d'exécuter la commande ci-dessus.

La connexion sur le port 8080 à l'adresse IP du boîtier (192.168.0.199:8080 à la livraison) permet d'accéder à une application Web permettant différents modes de visualisation des images de la Webcam. Un flux simple est par exemple visible à l'adresse:

http://192.168.0.199:8080/stream_simple.html

Notez qu'il est possible de connecter n'importe quel type de Webcam, à condition que ce soit une Webcam UVC.

4.7 - Serveur Web

La Carambola-Box embarque un serveur Web, dont les fichiers de configuration se trouvent dans le répertoire suivant :

/etc/config/uhttpd

Pour développer une application Web « servie » par la Carambola-Box, il suffit de placer les fichiers de cette application dans le répertoire /www.

ATTENTION !
Ce répertoire contient également les fichiers de l'application Web LuCi (pour la configuration des propriétés réseau) et de gestion de la Webcam.
Ne pas modifier tout fichier ou tout répertoire dont le nom contient « luci » ou « webcam ».

Il est possible de développer des pages Web intégrant le flux vidéo pris par la Webcam. Mais dans ce cas, dans la mesure où ce flux est accessible sur le port 8080 alors que le serveur Web s'accède sur le port 80, l'adresse du flux devra être intégré dans un « iframe » :

```
<iframe src="http://192.168.0.199:8080/stream_simple.html" width="700" height="500" frameborder="0">
</iframe>
```

5 - Important

La Carambola-Box est un produit « vivant » en constant développement pour améliorer ou lui ajouter de nouvelles fonctionnalités. Si vous avez des idées ou des besoins pour des développements spécifiques, n'hésitez pas à nous contacter (support@3sigma.fr).

Ne restez jamais bloqué sans nous contacter !
Pour tout problème ou toute requête, contactez-nous à l'adresse support@3sigma.fr.