Pierre Dieumegard
pierre.dieumegard@ac-orleans-tours.fr

Pymensura, un module Python pour le système Mensurasoft

Complémentarité de Python et des pilotes Mensurasoft

Python est un langage de programmation très utilisé dans l'enseignement ; c'est un langage interprété très souple, ayant toutes les structures de données et de programmation modernes. Il permet de faire en quelques lignes un programme assez puissant, et grâce à sa lisibilité, ces programmes sont (assez) faciles à maintenir et à améliorer.

Les pilotes d'appareils de mesure du système Mensurasoft sont des petits fichiers contenant des instructions exécutables (de type bibliothèques dynamiques, .dll pour Windows, .so pour Linux). Chaque fichier contient les instructions pour réaliser des mesures à partir d'un type donné d'appareil de mesure. Ils doivent obligatoirement être réalisés avec un langage compilé (C/C++, Pascal, et certains Basic), et ne sont pas toujours simples à réaliser.

L'association des deux permet de profiter de la puissance de programmation de Python tout en évitant les détails de la programmation pour un appareil particulier.

Le module Pymensura (pymensura.py) est à placer dans un répertoire accessible à Python. On peut le mettre :

- soit dans le répertoire de travail, là où sont le programme principal et les pilotes.

- soit dans le répertoire /Lib/site-packages, par exemple c:\Python33\Lib\site-packages sous Windows, ou dans /usr/lib/pythonx.y ou usr/local/lib/pythonx.y sous Linux.

Contenu du module pymensura

Il ne contient que la classe mensura, qui correspond à un appareil de mesure. Lorsqu'on veut utiliser un pilote correspondant à un appareil, on crée une instance (un objet) de cette classe par l'instruction :

```
mamachine=mensura("bibdyn_syst_pb_ANSI.dll")
```

Ensuite, on peut directement utiliser les méthodes et propriétés de cet objet.

`codage` : indique le codage des caractères (par défaut "windows-1252", mais on peut le fixer à « utf-8 », ou à d'autres valeurs.

`correct` : indique si le pilote a été correctement chargé ; pour cela, il faut d'une part que le fichier demandé existe, qu'il soit bien un fichier de type bibliothèque dynamique, et enfin qu'il contienne la fonction cdetail (la vérification des autres fonctions n'est pas faite à ce stade).

Les méthodes suivantes sont disponibles. Si l'initialisation n'a pas été bien faite, les fonctions renvoyant une chaîne de caractères renvoient une chaîne vide, sauf detail() et titre() qui renvoient un petit message d'erreur, et les fonctions numériques renvoient une valeur -999.

`titre()` : titre assez court du pilote, avec le nom de l'appareil

`detail()` : chaîne de caractères plus longue que le titre, avec souvent le nom de l'auteur

et la date de réalisation du pilote.

`ead(n)` est la n-ième entrée analogique (à partir de zéro), et nead(n) le nom de cette entrée analogique

`sad(n, valeur)` fixe à valeur la n-ième sortie analogique (à partir de zéro), et nsad(n) est le nom de cette sortie analogique

`eb(n)` est la n-ième entrée binaire (à partir de zéro), et neb(n) est le nom de cette entrée binaire.

`sb(n,valeur)` fixe la n-ième sortie analogique (à partir de zéro), et nsb(n) est le nom de cette sortie binaire.

`stop()` arrête l'utilisation de l'appareil, ce qui le rend disponible pour un autre programme.

Un exemple de programme pour utiliser ces fonctions

On suppose que le nom du pilote est « bibdyn_syst_pb_ANSI.dll ».

Ce programme charge le pilote en question, puis affiche le titre, le détail, l'entrée analogique 0, le nom de cette entrée analogique 0, l'entrée binaire 0, le nom de cette entrée binaire 0, fixe la sortie analogique 0 à la valeur 1, affiche le nom de cette sortie analogique 0, fixe la sortie binaire 0 à la valeur 1 (= « vrai »), affiche le nom de la sortie binaire 0 et finalement arrête l'utilisation de l'appareil (stop()).

```
from pymensura import *
mamachine=mensura("bibdyn_syst_pb_ANSI.dll")
print(mamachine.titre())
print(mamachine.detail())
print(mamachine.ead(0))
print(mamachine.nead(0))
print(mamachine.eb(0))
print(mamachine.neb(0))
print(mamachine.sad(0,1))
print(mamachine.nsad(0))
print(mamachine.sb(0,1))
print(mamachine.nsb(0))
mamachine.stop()
```

Pour quels appareils de mesure ?

1) Pour tous ceux dont des pilotes ont été faits, et ils sont nombreux. Sur le site http://sciencexp.free.fr, on peut en trouver pour un grand nombre d'appareils :

- interfaces généralistes d'EXAO : Jeulin, Orphy, Pierron, Eurosmart, ExpEyes…
- appareils de mesure spécifiques:pHmètres, conductimètres, balances, multimètres, colorimètres, spectrophotomètres, thermomètres, luxmètres…
- petites cartes pour bricoleurs : Arduino, MicroHope, Velleman…
- bricolages pour la prise manettes de jeux, prises série ou parallèle, etc.

2) Et aussi pour tous ceux dont vous pouvez faire le pilote avec un langage compilé (Delphi, FreePascal, C ou C++, PureBasic ou FreeBasic…). Consultez la documentation et les programmes-sources sur http://sciencexp.free.fr