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

Pymensura, a module for Python and Mensurasoft-system

Complementarity between Python and Mensurasoft drivers

Python is a programming language widely used in teaching; it is a very versatile interpreted language, with modern structures for data and programming. It allows to make in a few lines a powerful program. Its readability allows these programs to be (quite) easy to maintain and to improve.
Mensurasoft System Device Drivers are small files containing executable instructions (dynamic libraries, .dll for Windows, .so for Linux). Each file contains instructions for making measurements from a given type of measuring device. They must be compiled with a compiled language (C/C++, Pascal, and some Basic), and they are not always easy to implement.
The combination of both programs allows you to take advantage of Python's programming power while avoiding programming details for a particular device.
Th Pymensura module (pymensura.py) is to be set in a directory known by Python. You can put it :

- either in the working directory, where are the main program in Python and the drivers.

- either in the directory of Python /Lib/site-packages, for example c:\Python33\Lib\site-packages for Windows, or in /usr/lib/pythonx.y or usr/local/lib/pythonx.y for Linux.

Contents of pymensura module

Tt has only one class : mensura. This class corresponds to a measurement device. When you want to use a driver or this device, you have to create an instance (an object) of this class by the instruction :

```
mydevice=mensura("bibdyn_syst_pb_ANSI.dll")
```
(if bibdyn_syst_pb_ANSI.dll is the driver of your device)

After that, you can directly use the methods and the properties of this object.

`code` : gives the coding for strings (by default "windows-1252", but you can fix it to "utf-8", or to other values.

`correct` : says if the driver is correctly loaded ; in this way, the file must exist, it must be a file with the correct type (dynamic library), and the file must contain the function cdetail (other functions are not checked at this stage).

You can use the following methods. If initialization is not correct, the functions giving a string give an empty string, except detail() and titre() who give a little error message, and numerical functions give the value -999.

`titre()` : a short title for the driver, with de name of the device.

`detail()` : a string longer than titre(), with often the name of the programmer, and the date of programming.

`ead(n)` is the n-th analog input (with beginning at zero), and nead(n) is the name of this analog input

`sad(n, value)` sets the n-th analog output to value, and nsad(n) is the name of this analog output.

`eb(n)` is the n-th binary input (with beginning at 0), and neb(n) is the name of this binary input.

`sb(n,value)` sets the n-th analog output to value, and nsb(n) is the name of this binary output.

stop() stops this instance, and the device cannot be used by this program.

An example of program, to use these functions

We assume that the name of the driver is « bibdyn_syst_pb_ANSI.dll ».

This program loads the driver, and shows the title (titre()), the detail (detail()), analog input 0, name of analog input 0, binary input 0, name of the binary input 0, sets the output 0 to 1, shows the name of this analog output, shows the name of analog output 0, sets the binary output 0 to s1 (=True), shows the name of this binary output, and stops the driver.

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

Which devices for this module?

1) All devices whose drivers exist, and they are a lot. In the website http://sciencexp.free.fr, you can find drivers for a lot of devices :

- general interfaces for experiments by computer : Jeulin, Orphy, Pierron, Eurosmart, ExpEyes…

- specific devices :pHmeters, conductimeters, scales, multimeters, colorimeters, spectrophotometers, thermometers, luxmeter…

- electronic cards : Arduino, MicroHope, Velleman…

- home-made circuits for game connector, or for serial connector, or for parallel connector etc.

2) And if there is not yet a driver for your device, you can make one with a compiled language (Delphi, FreePascal, C or C++, PureBasic or FreeBasic…). Read the documentation and source-programs at http://sciencexp.free.fr