pywws Documentation

Version 16.12.0.dev1367

Jim Easterbrook

13 December 2016

Table des matières

Logiciel Python pour stations météo USB sans-fil

Pywws est une collection de modules Python pour lire, enregistrer et traiter les données provenant de stations météorologiques sans fil USB populaires comme AstroTouch Elecsa 6975, Watson W-8681, WH-1080PC, WH1080, WH1081, WH3080 etc. Je suppose que n'importe quel modèle qui est fourni avec le logiciel Windows EasyWeather est compatible, mais ne peut le garantir.

Le logiciel a été développé pour être exécuté dans un environnement de basse puissance, avec peu de mémoire tel un routeur ou un Raspberry Pi. Il peut être utilisé pour créer des graphiques et les pages web affichant les relevés météorologiques récents, généralement mis à jour toutes les heures. Il peut également envoyer des données 'live' pour des services tels que Weather Underground et poster des messages sur Twitter.

La version de développement de pywws est hébergé sur GitHub.

— https://github.com/jim-easterbrook/pywws

Les versions «Snapshot» de pywws sont disponibles à partir du Python Package Index (PyPI).

— https://pypi.python.org/pypi/pywws

La documentation est hébergé sur Read the Docs.

— http://pywws.readthedocs.org/

La documentation est disponible dans les langues suivantes (les versions non anglaises peuvent ne pas être complètes ou à jour)

— English

— Français – traduit par Jacques Desroches
— Italiano – translated by Edoardo

J'ai écrit ce logiciel pour répondre à mes besoins, mais ai essayé de le rendre adaptable aux besoins des autres. Vous voudrez peut-être modifier certains ou tous les modules, ou en écrire de nouveaux, pour lui faire faire exactement ce que vous souhaitez. L'une des raisons pour lesquelles Python est utilisé est qu'il rend de telles modifications si facile. N'ayez pas peur, essayez-le vous verrez..

CHAPITRE 1

Exigences

Les logiciels dont vous aurez besoin pour exécuter pywws dépend de ce que vous comptez en faire. Vous aurez besoin de Python 2.5 ou plus récent – Python 3 est partiellement supporté, certaine fonctionnalité dépendent de librairies qui n'ont pas été portées sur Python3.

Pour plus de détails, voir Pré-requis.

CHAPITRE 2

Installation et mise à niveau de pywws

Pywws peut être installé directement à partir du Python Package Index (PyPI) en utilisant la commande PIP. Voir Comment démarrer avec pywws pour obtenir des instructions complètes.

Certaines nouvelles versions de pywws modifient ce qui est stocké dans des fichiers de données de synthèse horaire, quotidien ou mensuel. Ces nouvelles versions sont incompatibles avec les données traitées à partir de versions antérieures. Le script `pywws.Reprocess` régénère toutes les données résumé. Il doit être exécuté après toute mise à niveau majeure.

CHAPITRE 3

Documentation

La documentation est inclue avec le téléchargement de pywws, et est également disponible en ligne. Un bon point de départ est le Guide de démarrage qui décrit plus en détail comment installer pywws.

Si vous avez des questions dont vous ne trouvez pas réponse dans la documentation, s'il vous plaît joindre la liste / groupe de discussion pywws Google et y poser votre question. Notez que votre premier message n'apparaît pas immédiatement – les nouveaux membres doivent être approuvés par un modérateur, pour empêcher les pourriels.

3.1 Contenu

3.1.1 Licence Publique Générale GNU

```
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and


    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.
  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.
You may charge a fee for the physical act of transferring a copy, and
```

```
you may at your option offer warranty protection in exchange for a fee.
  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:
    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.
    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.
    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.
In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.
  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:
    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,
    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,
    c) Accompany it with the information you received as to the offer
```

```
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.
If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.
  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.
  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.
  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.
  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
```

```
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.
It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.
This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.
  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.
  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.
Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.
  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.



    NO WARRANTY
  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
```

```
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.
  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.


     END OF TERMS AND CONDITIONS

    How to Apply These Terms to Your New Programs
  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.
  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:
    Gnomovision version 69, Copyright (C) year  name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.
You should also get your employer (if you work as a programmer) or your
```

```
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.
  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice
This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
modification follow.
```

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

3.1.2 Contributeurs de pywws

Le copywright de pywws et sa documentation est détenu conjointement par les contributeurs suivants.

```
Developers
----------

Jim Easterbrook                 jim@jim-easterbrook.me.uk
x2q
3v1n0
Robin Kearney                   robin@kearney.co.uk
Rod Persky
Morten Høybye Frederiksen       morten@mfd-consult.dk
Simon Josefsson                 simon@josefsson.org
Matthew Hilton                  matthilton2005@gmail.com
Sabine Tobolka                        oe1yvw@gmail.com
Markus Birth                    markus@birth-online.de

Translators
-----------

Edoardo                         edoardo69@hotmail.it
Jacques Desroches               metelsto@gmail.com
Sunshades                       joacim@ahlstrand.info
Johabu                          johabu96@yahoo.de
                                karte2@gmail.com
Kyle Gordon                     kyle@lodge.glasgownet.com>
Πτρο                           nouvakis@sch.gr
Ramiro                          ramiro.sanchez@telefonica.net
Rick Sulman                     rick@sulman.org
Pyttsen                         weather@spacelab.se
Tech2304                        tech2304@gmail.com
Pablo Vera                      pablo.vera82@gmail.com
```

Contribuant à pywws

Si vous souhaitez ajouter une fonctionnalité à pywws (ou résoudre un problème) s'il vous plaît faites le. Les logiciel Open Source se développent lorsque ses utilisateurs deviennent des contributeurs actifs. Le processus est assez simple :

1. Join GitHub - c'est gratuit.
2. Fork le projet pywws - voir Fork un dépôt pour l'aide.
3. Clonez votre fork sur un ordinateur que vous pouvez utiliser pour développer votre nouvelle fonction.
4. Utilisez git pour valider les modifications que vous produisez et poussez les modifications à votre fork de pywws.
   Veuillez ajouter un signed-off-by à votre envoi qui certifie votre certificat de développeur d'origine (voir ci-dessous). Par exemple, si votre nom est " John Smith ", et votre adresse e-mail est 'Jsmith@example.com', il suffit d'inclure la ligne suivante au bas de vos messages de commit :

   Signed-off-by : John Smith <jsmith@example.com>

   Vous devriez être capable de faire cela automatiquement en utilisant l'option `-s` de l'option sur votre commande ''git commit''.
5. Ajoutez votre nom et email au fichier `src/contributors/contributors.txt`. N'oubliez pas l'option ''-s" lorsque vous validez ce changement.
6. Testez vos modifications !
7. Quand tout fonctionne comme prévu, soumettez une requête Pull.

Certificat d'origine de développeur

Incluez un Signed-off-by dans votre envoi indique que vous certifiez ce qui suit :

```
Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
660 York Street, Suite 102,
San Francisco, CA 94110 USA

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.


Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
```

```
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

Les clauses (a), (b) et (c) rassurent les utilisateurs de pywws que le projet restera bien en Open Source à l'avenir. La division (d) vous rappelle que vos contributions seront disponibles publiquement, et vous n'avez pas le droit de les retirer à l'avenir.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

3.1.3 Pré-requis

La liste des logiciel requis par pywws peut être effrayante au premier abord. Cependant, plusieurs de ces paquets ne sont pas nécessaires pour la plupart des utilisateurs. Ce que vous avez besoin dépend de ce que vous voulez faire avec pywws. Rappelez-vous que pywws est "un ensemble de pièces" plutôt qu'une application monolithique.

Certains paquets Python sont nécessaires, ils peuvent être téléchargés à partir de Python Package Index (PyPI). Je recommande l'utilisation de pip pour en faire l'installation.

Vous devriez être en mesure d'installer les autres dépendances en utilisant le gestionnaire de paquets de votre système d'exploitation. C'est beaucoup plus simple que de télécharger et de compiler les fichiers source depuis les sites Web des projets. Notez que certaines distributions de Linux utilisent des noms différents pour certains paquets, par exemple dans Ubuntu, pyusb est nommé python-usb.

Remarque : certaines de ces bibliothèques peuvent avoir leurs propres dépendances que vous devrez peut-être installer. Suivez les liens pour en savoir plus sur les exigences de chaque bibliothèque.

Essentiel

— Python version 2.5 2.5 ou supérieur

La version 3 de python est supportée, mais certains modules pourraient ne pas fonctionner correctement. Si vous trouvez un problème avec python 3, svp envoyez un message sur la **'liste pywws < http ://groups.google.com/group/pywws >'_** ou soumettez un rapport de bogue sur GitHub.

— pip

Vous serez probablement en mesure d'installer pip avec le gestionnaire de paquets de votre système, où il peut être nommé python-pip, python3-pip ou quelque chose de similaire. Sinon, téléchargez et exécutez ''get-pip.py" à partir du site web de pip. Dans les deux cas, vous devez immédiatement utiliser pip pour installer la dernière version :

```
sudo pip install --upgrade pip
```

Assurez-vous d'installer la bonne version Python de pip. Si vous souhaitez installer pywws tant pour Python 2 que Python 3 vous aurez besoin de pip2 et pip3.

— tzlocal

C'est un petit module qui fournit des informations sur votre fuseau horaire local. Il vaut mieux l'installer avec "pip"' :

```
sudo pip install tzlocal
```

Librairie USB

Pour lire les données d'une station météorologique, pywws a besoin d'une bibliothèque python qui lui permet de communiquer par l'intermédiaire d'un port USB. Il y a un grand choix de bibliothèques USB qui peuvent être employées. Elles ne sont pas toutes disponibles sur toutes les plates-formes, ce qui peut limiter votre choix.

Mac OS X Sur MacOS X, le pilote hid générique du système d'exploitation "réclame" la station météo, ce qui rend très difficile l'utilisation de tout autre interface USB. Malheureusement, vous aurez besoin de télécharger et compiler hidapi vous-même.

— hidapi
— ctypes (Votre gestionnaire de paquet peut le connaître sous le nom python-ctypes)

Si vous ne pouvez pas installer ctypes alors, vous pouvez essayer l'interface à hidapi avec Cython à la place :

— cython-hidapi
— cython (Votre gestionnaire de paquet peut le connaître sous le nom python-Cython)

Autre systèmes D'autres systèmes utilisent une interface Python pour la librairie système libusb. Il y a un choix d'interfaces et version de le bibliothèque - installez la dernière qui est disponible pour votre ordinateur.

— libusb version 1.x (Devraient être disponibles à partir du gestionnaire de paquets)
— python-libusb1 version 1.3

```
pip install libusb1
```

or

— libusb version 1.x or version 0.1 (Devraient être disponibles à partir du gestionnaire de paquets)
— PyUSB version 1.0

```
pip install pyusb --pre
```

L'indicateur ''–pre" permet l'installation de versions "pré-production", comme l'actuelle version bêta (version 1.0.0b2) de pyusb.

Si aucune de ces options ne fonctionne pour vous, alors vous pouvez utiliser hidapi – voir les instructions pour le Mac OS X ci-dessus.

Modifié dans la version 15.01.0.dev1265 : Ajout de la possibilité d'utiliser l'interface python-libusb1.

Tâches chronométrées flexibles

Le module `pywws.Tasks` peut accomplir des tâches à des moments et / ou des dates particulières. Cela nécessite la bibliothèque croniter. (les tâches simple horaire, quotidienne ou les tâches 'Live' n'ont pas besoin de cette bibliothèque.)

— croniter

```
pip install croniter
```

S'exécute comme daemon

Le programme `pywws.livelogdaemon` exécute l'historisation Live de pywws comme un processus démon UNIX appropriée. Exige la bibliothèque python-daemon :

— python-daemon

```
pip install python-daemon
```

Graphes

Le module de tracé `pywws.Plot` utilise gnuplot pour générer les graphiques. Si vous voulez produire des graphiques de données météorologiques, par exemple, inclus dans une page web, vous devez installer l'application gnuplot :

— gnuplot v4.2 or supérieur (Devraient être disponibles à partir du gestionnaire de paquets)

Après l'installation de gnuplot, Vous devez éditer weather.ini (voir weather.ini - format du fichier de configuration) et définissez l' élément de configuration "gnuplot version". Trouver la version de gnuplot installé est facile :

```
gnuplot -V
```

Téléversement web sécurisé (sftp)

Le module *pywws.Upload* peut utiliser "FTP sur ssh" (sftp) pour envoyer les fichiers sur votre site web. L'envoi normal emploi simplement les modules standard Python, mais si vous souhaitez utiliser sftp vous devez installer ces deux modules :

— paramiko

— pycrypto

```
sudo pip install pycrypto paramiko
```

Mise à jour Twitter

Le module *pywws.ToTwitter* peut être utilisé pour envoyer les messages météorologiques sur Twitter. La publication sur Twitter nécessite ces modules :

— python-twitter v2.1 or higher

— python-oauth2

```
sudo pip install python-twitter oauth2
```

or

— tweepy v2.0 ou supérieur

— python-oauth2

```
sudo pip install tweepy oauth2
```

A noter que `tweepy` semble être le moins fiable des deux. Si vous avez des problèmes, e. g. avec le codage de caractères, essayez plutôt d'installer `python-twitter`.

Modifié dans la version 13.10_r1086 : Réactivé l'utilisation de la bibliothèque `tweepy` comme alternative à `python-twitter`. `python-oauth2` est toujours requis par *pywws.TwitterAuth*.

Modifié dans la version 13.06_r1023 : pywws a auparavant utilisé la bibliothèque `tweepy` au lieu de `python-twitter` et `python-oauth2`.

MQTT

Nouveau dans la version 14.12.0.dev1260.

Le module `pywws.toservice` peut être utilisé pour envoyer des données météorologiques à un courtier MQTT. Cela exige le module paho-mqtt :

— paho-mqtt

```
sudo pip install paho-mqtt
```

Pour créer une nouvelle traduction

pywws peut être configuré pour utiliser d'autres langues que l'anglais, comme décrit dans Comment utiliser pywws dans une autre langue. Le paquet Babel est nécessaire pour extraire les chaînes à traduire et à compiler les fichiers de traduction.

— babel

```
sudo pip install babel
```

La copie de fichiers vers ou depuis Transifex nécessite le paquet transifex-client.

— transifex-client

```
sudo pip install transifex-client
```

Traduire la documentation à l'aide de fichiers locaux nécessite le paquet sphinx-intl.

— sphinx-intl

```
sudo pip install sphinx-intl
```

Modifié dans la version 14.05.dev1209 : pywws a auparavant utilisé le paquet gettext.

Pour 'compiler' la documentation

La documentation de pywws est écrite en "texte restructuré". Un programme appelé Sphinx est utilisée pour convertir ce format facile à écrire en code HTML pour l'utiliser avec un navigateur web. Si vous souhaitez créer une copie locale de la documentation (pour ne pas avoir à compter sur la version en ligne, ou pour tester une traduction sur laquelle vous travaillez), vous devez installer Sphinx, version 1.3 ou ultérieure.

— Sphinx

```
sudo pip install sphinx
```

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

3.1.4 Historique

```
pywws - Python software for USB Wireless Weather Stations
http://github.com/jim-easterbrook/pywws
Copyright (C) 2008-16  pywws contributors

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.

Changes in v16.12.0:
    1/ Added "candlestick" plot type.
```

```
    2/ Added cloud base calculation function.
    3/ Various other bug fixes and minor improvements.

Changes in v16.08.0:
    1/ Fix Python 2.5 incompatibilities.
    2/ Fix python-twitter v3 tweet length problem.

Changes in v16.07.1:
    1/ Further changes to handle UK Met Office server quirks.

Changes in v16.07.0:
    1/ Fix bug with UK Met Office uploads server change.
    2/ Allow user commands in wind roses.
    3/ Various other bug fixes and minor improvements.

Changes in v15.12.0:
    1/ Fix bug with Twitter messages being excessively truncated.
    2/ Improve handling of utf-8 encoded templates.
    3/ Improved plots and wind roses with 'pngcairo' "terminal".
    4/ Various bug fixes and minor improvements.

Changes in v15.11.0:
    1/ Add Russian translation of program text.
    2/ Improved documentation.
    3/ Various bug fixes and minor improvements.

Changes in v15.07.0:
    1/ Can include multiple media in Twitter messages.
    2/ Attempt to fix bug in wind rose axes labels.
    3/ Enable inclusion of time & date in wind rose title.
    4/ Various bug fixes and minor improvements.

Changes in v15.01.0:
    1/ Added 'MQTT' service.
    2/ Added another USB library option.
    3/ Improved Python 3 compatibility.
    4/ Various bug fixes and minor improvements.

Changes in v14.12.0:
    1/ Updated temperatur.nu and wetterarchiv.de service details to suit new
       APIs.

Changes in v14.06.1:
    1/ Revised version numbering scheme.
    2/ Compiled documentation no longer included in releases.
    3/ Can partially specify start & stop date/time in graphs, e.g. to start a
       plot at midnight, no matter when it is plotted.

Changes in v14.06:
    1/ Can now send images to Twitter.
    2/ Periodic tasks can be specified with a cron style syntax.
    3/ Added wind direction filter for use in graphs or user calibration
       modules.
    4/ Wind direction is now stored as a float. Old templates that use the
       wind_dir_text array will need updating, probably to use the
       winddir_text() function.
    5/ Started using "Transifex" to host translations. Changed tools and
       procedures to create new translations.
```

```
    6/ Improved USB hangup avoidance strategy for stations with large clock
       drift figures.
    7/ Various bug fixes and minor improvements.

Changes in v14.05:
    1/ Rearranged package layout, moving examples and documentation.
    2/ Added 'entry point' auto-generated commands for some modules.
    3/ Added verbose output option to pywws-version command.
    4/ Various bug fixes and minor improvements.

Changes in v14.03:
    1/ Extracts additional status from 'wind_dir' byte. You must run
       pywws-reprocess.py with the -u option after upgrading from any previous
       version.
    2/ Added Citizen Weather Observer Program to available 'services'.
    3/ Improved asynchronous upload task queuing.
    4/ Various bug fixes and minor improvements.

Changes in v14.02:
    1/ Improved time zone handling, including non whole hour time zones.
    2/ New 'frequent writes' config option.
    3/ Improved 'live log' sync, particularly with 3080 type stations.
    4/ Record recent memory pointer to improve detection of gaps in data.
    5/ Various bug fixes and minor improvements.

Changes in v13.12:
    1/ Changed API of user calibration module.
    2/ Can use python-twitter *or* tweepy library.
    3/ Added a script to run live logging as a UNIX daemon process.
    4/ Changed data store to use separate read and write caches.
    5/ Various bug fixes and minor improvements.

Changes in v13.10:
    1/ Changed Twitter library from tweepy to python-twitter.
    2/ Added ability to do uploads asynchronously.
    3/ Rearranged and improved documentation.
    4/ Various bug fixes and minor improvements.

Changes in v13.06:
    1/ Substantially rearranged directories, getting rid of 'code' and 'code3'.
    2/ Removed 'makefile' - everything is now done via 'setup.py'.
    3/ Removed 'RunModule.py' - use 'python -m pywws.module' now.
    4/ Separated storage of config (weather.ini) and status (status.ini).
    5/ Replaced toservice.py "rapid fire" mode with a separate config file for
       Weather Underground rapid fire.
    6/ Added 2 more low-level USB access modules.
    7/ Various bug fixes and minor improvements.

Changes in v13.03:
    1/ Added 'rain days' to monthly data. (Reprocess required when upgrading.)
    2/ Extended template syntax to include comments.
    3/ Added 'humidity index' function.
    4/ Added French translation of documentation.
    5/ Reduced frequency of saving data files.
    6/ Various bug fixes.

Changes in v12.12:
    1/ Added support for Python 3.
```

```
    2/ Added French documentation translation.
    3/ Used 'binary search' to speed up data access.
    4/ Various bug fixes.

Changes in v12.11:
    1/ Moved development from Google code to GitHub.
    2/ Made software attempt to avoid USB activity at times when it is assumed
       the weather station might be writing to its memory. This might solve
       the USB lockup problem, but it's too early to tell.

Changes in v12.10:
    1/ Added a 'winddir_text' function for use in templates.
    2/ Added <ytics> and <y2tics> options to graph plots.
    3/ Various bug fixes.

Changes in v12.07:
    1/ Added Open Weather Map to the services.
    2/ Fixed problem with Weather Underground uploads that started on 1st June.
    3/ Various bug fixes and software structure improvements.

Changes in v12.05:
    1/ Made 'fixed block' data available to template calculations.
    2/ Fixed buggy auto-detection of 3080 weather stations.
    3/ Added a function to generate the Zambretti forecast code letter.
    4/ Added a program to test USB communication reliablility.
    5/ Various bug fixes and software structure improvements.

Changes in v12.02:
    1/ Separated out low level USB communications to enable use of different
       libraries. Now works on recent versions of Mac OS.
    2/ Added humidity, pressure & wind data to summary data.
    3/ Merged Weather Underground and UK Met Office uploaders into one combined
       module. Added more 'service' uploaders.
    4/ Various bug fixes and software structure improvements.

Changes in v11.10:
    1/ Complete restructuring of documentation.
    2/ Added a user defined 'calibration' process.
    3/ Sets 'locale' according to language setting.
    4/ Added ability to upload to UK Met Office 'WOW'.
    5/ Various bug fixes and software structure improvements.
    6/ New language files: French, Danish.

Changes in v11.05:
    1/ Added support for '3080' family stations that have illuminance and
       UV sensors.
    2/ Broadened the range of tasks that can be done with 'live' data.
    3/ Various bug fixes and software structure improvements.

Changes in v11.02:
    1/ Various bug fixes and software structure improvements.
    2/ Improved wind direction averaging.
    3/ Added conversion functions for common things such as C to F.
    4/ Added a YoWindow module.
    5/ Improved Zambretti forecaster.

Changes in v10.12:
    1/ Various bug fixes and software structure improvements.
```

```
    2/ Added a 'goto' instruction to Template.py.
    3/ Added a 'Zambretti' forecast function to Template.py. This should
       be treated as an experiment, and not relied upon for accuracy.

Changes in v10.10:
    1/ Added 'catchup' mode to ToUnderground.py.
    2/ Created 'Tasks.py' to handle common tasks.
    3/ Made better use of Python's logger for info and error
       messages.
    4/ Changed over from 'python-twitter' to 'tweepy' for Twitter
       access. Twitter authorisation using OAuth now works.
    5/ Added 'LiveLog.py' live logging program.
    6/ Added 'SetWeatherStation.py' to do some configuration of weather
       station. No longer need EasyWeather to set logging interval!
    7/ Added 'Rapid Fire' ability to ToUnderground.py.
    8/ Added plain text versions of HTML documentation.
    9/ Many bug fixes and minor improvements.

Changes in v10.08:
    1/ Added internal temperature to daily and monthly summaries.
       Run Reprocess.py when upgrading from earlier versions.
    2/ Added 'prevdata' to Template.py. Allows calculations that
       compare values from different times.
    3/ Made 'pressure_offset' available to calculations in Plot.py
       and Template.py. This is only useful when using 'raw' data.
    4/ Improved synchronisation to weather station's clock when
       fetching stored data.

Changes in v10.06:
    1/ Improved localisation code.
    2/ Minor bug fixes.
    3/ Added Y axis label angle control to plots.

Changes in v10.04:
    1/ Changed version numbering to year.month.
    2/ Allowed "upload" to a local directory instead of ftp site.
    3/ Added "calc" option to text templates (Template.py).
    4/ Added -v / --verbose option to Hourly.py to allow silent operation.
    5/ Added internationalisation / localisation of some strings.
    6/ Made 'raw' data available to text templates.
    7/ Added ability to upload to Weather Underground.
    8/ Added dual axis and cumulative graph capability.

Changes in v0.9:
    1/ Added lowest daytime max and highest nighttime min temperatures
       to monthly data.
    2/ Added average temperature to daily and monthly data.
    3/ Added 'terminal' element to Plot.py templates for greater control
       over output appearance.
    4/ Added 'command' element to Plot.py templates for even more
       control, for advanced users.
    5/ Added secure upload option.
    6/ Minor speed improvements.

Changes in v0.8:
    1/ Added meteorological day end hour user preference
    2/ Attempts at Windows compatibility
    3/ Corrected decoding of wind data at speeds over 25.5 m/s
```

```
    4/ Improved speed with new data caching strategy

Changes in v0.7:
    1/ Several bug fixes, mostly around new weather stations with not
       much data
    2/ Added min & max temperature extremes to monthly data
    3/ Added template and workspace directory locations to weather.ini
    4/ Increased versatility of Plot.py with layout and title elements

Changes in v0.6:
    1/ Added monthly data
    2/ Changed 'pressure' to 'abs_pressure' or 'rel_pressure'

Changes in v0.5:
    1/ Small bug fixes.
    2/ Added start time to daily data
    3/ Replaced individual plot programs with XML "recipe" system

Changes in v0.4:
    1/ Can post brief messages to Twitter.
    2/ Now time zone aware. Uses UTC for data indexing and local time
       for graphs and text data files.

Changes in v0.3:
    1/ Now uses templates to generate text data
    2/ Added 28 day plot
    3/ Minor efficiency improvements
    4/ Improved documentation

Changes in v0.2:
    1/ Now uses Python csv library to read and write data
    2/ Creates hourly and daily summary files
    3/ Includes rain data in graphs
```

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

3.1.5 Guides utilisateur

Contenu :

Comment démarrer avec pywws

Installation

Vous devez d'abord installer Python et une bibliothèque USB (pour permettre à Python d'accéder à la station météo). Voir Pré-requis pour plus d'informations.

Créer un dossier pour tous vos fichiers reliés à la météo et vous positionner dans ce dossier. Par exemple (avec un système d'exploitation Linux ou similaire) :

```
mkdir ~/weather
cd ~/weather
```

Installation facile Le plus simple consiste à installer pywws avec la commande pip :

```
sudo pip install pywws
```

La mise à niveau de pywws est également une commande d'une seule ligne :

```
sudo pip install -U pywws
```

Vous êtes maintenant prêt pour *Tester la connexion de la station météo.*.

Téléchargement et extraction If you prefer not to use pip, or you want easy access to the pywws source files (e.g. to translate the documentation – see Comment utiliser pywws dans une autre langue), you can download and extract the files into your weather directory.

Visit http://pypi.python.org/pypi/pywws/ and download one of the .tar.gz or .zip files. Put it in your weather directory, then extract all the files, for example :

```
cd ~/weather
tar zxvf pywws-14.03.dev1178.tar.gz
```

ou :

```
cd ~/weather
unzip pywws-14.03.dev1178.zip
```

This should create a directory (called `pywws-14.03.dev1178` in this example) containing all the pywws source files. It is convenient to create a soft link to this awkwardly named directory :

```
cd ~/weather
ln -s pywws-14.03.dev1178 pywws
```

Upgrading a downloaded snapshot is the same process as the first installation. Download the .tar.gz or .zip file, extract its contents, then delete the soft link pointing to the old download and create one pointing to the new download. Once you are satisfied the new version is working OK you can delete the old download entirely.

Cloner l'entrepôt The PyPI files contain a snapshot release of the software - a new one is issued every few months. If you want to use the very latest version of pywws, e.g. to work on fixing a bug, you can get all the files you need from the GitHub repository. Install git and use it to clone the repos :

```
cd ~/weather
git clone https://github.com/jim-easterbrook/pywws.git
```

To upgrade you use git to pull any changes :

```
cd ~/weather/pywws
git pull
```

Install pywws If you have downloaded or cloned the pywws source files, you need to use setup.py to install it :

```
cd ~/weather/pywws
python setup.py build
sudo python setup.py install
```

Note to Python 3 users : this will generate and use Python 3 versions of the pywws software in `~/weather/pywws/build/lib`.

Compile documentation (optional)

If you'd like to have a local copy of the pywws documentation (and have downloaded the source or cloned the repo) you can "compile" the English documentation. This requires the sphinx package :

```
cd ~/weather/pywws
python setup.py build_sphinx
```

Compiling the documentation in another language requires the additional step of compiling the translation files, which requires the sphinx-intl package. For example, to compile the French documentation :

```
cd ~/weather/pywws
sphinx-intl build --locale-dir src/pywws/lang -l fr
LANG=fr python setup.py build_sphinx
```

The compiled documentation should then be found at `~/weather/pywws/doc/html/index.html`. See Comment utiliser pywws dans une autre langue for more detail.

Tester la connexion de la station météo.

Now you're ready to test your pywws installation. Connect the weather station (if not already connected) then run the *pywws.TestWeatherStation* module :

```
pywws-testweatherstation
```

Si tout fonctionne correctement, vous devriez voir apparaitre un lot de chiffres ressemblant à ceci :

```
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00 0f 00 00 60 55
0020 ea 27 a0 27 00 00 00 00 00 00 00 10 10 12 13 45 41 23 c8 00 32 80 47 2d 2c 01 2c 81 5e 01 1e 80
0040 96 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 03 00 00 00 00 00 00 00
0060 00 00 4e 1c 63 0d 2f 01 73 00 7a 01 47 80 7a 01 47 80 e4 00 00 00 71 28 7f 25 bb 28 bd 25 eb 00
0080 0c 02 84 00 0e 01 e3 01 ab 03 dc 17 00 10 08 21 08 54 10 03 07 22 18 10 08 11 08 30 10 04 21 16
00a0 26 08 07 24 17 17 08 11 01 06 10 09 06 30 14 29 09 01 06 07 46 09 06 30 14 29 09 01 06 07 46 08
00c0 08 31 14 30 10 05 14 15 27 10 01 26 20 47 09 01 23 05 13 10 01 26 20 47 09 01 23 05 13 10 02 22
00e0 11 06 10 02 22 11 06 08 07 07 19 32 08 12 13 22 32 08 09 07 08 48 01 12 05 04 43 10 02 22 14 43
```

Si ce test ne fonctionne pas, plusieurs facteurs peuvent être en cause, mais le plus courant est un problème de 'permissions' ; ce qui peut être vérifié en exécutant la commande suivante avec les droits 'root' :

```
sudo pywws-testweatherstation
```

If this works then you may be able to allow your normal user account to access the weather station by setting up a 'udev' rule. The exact method may depend on your Linux version, but this is typically done by creating a file `/etc/udev/rules.d/39-weather-station.rules` containing the following :

```
ACTION!="add|change", GOTO="weatherstation_end"
SUBSYSTEM=="usb", ATTRS{idVendor}=="1941", ATTRS{idProduct}=="8021", GROUP="weatherstation"
LABEL="weatherstation_end"
```

Unplug and replug the station's USB connection to force `udev` to apply the new rule. This allows any user in the group `weatherstation` to access the weather station. You need to create this group and add your normal user account to it – many Linux systems have a GUI for user and group management.

Pour tout autre problème, n'hésitez pas à demander de l'aide sur la liste de publipostage de pywws : http://groups.google.com/group/pywws

Configurer votre station météo.

If you haven't already done so, you should set your weather station to display the correct relative atmospheric pressure. (See the manual for details of how to do this.) pywws gets the offset between relative and absolute pressure from the station, so this should be set before using pywws.

Vous pouvez obtenir la bonne pression relative de votre emplacement en recherchant sur Internet des rapports météorologiques d'une station à proximité, idéalement une source officielle, tel un aéroport. Il est préférable de le faire durant une période calme lorsque la pression est presque constante sur une longue période.

Configurer l'intervalle d'enregistrement de la station météo.

Votre station météo a probablement quitté l'usine avec un intervalle d'enregistrement de 30 minutes. Ceci permet à la station d'enregistrer environ 11 semaines de données. La plupart des utilisateurs de pywws configurent leur ordinateur pour lire les données de la station à chaque heure, ou plus fréquemment et souhaitent que la station contienne assez de données pour couvrir d'éventuelles panne d'ordinateur. L'intervalle recommandé est de 5 minutes, ce qui représente 2 semaines de données. Utilisez le programme *pywws.SetWeatherStation* pour fixer l'intervalle :

```
pywws-setweatherstation -r 5
```

Note that the weather station will not start using the new interval until the current 30 minute logging period is finished. This may cause "station is not logging data" errors when running pywws logging. If this happens you need to wait until the 30 minute logging period ends.

Enregistrer les données de votre station météo.

First, choose a directory to store all your weather station data. This will be written to quite frequently, so a disk drive is preferable to a flash memory stick or card, as these have a limited number of writes. In most cases your home directory is suitable, for example :

```
mkdir ~/weather/data
```

Ce répertoire est référé dans la documentation pywws en tant que votre 'répertoire de données'.

Assurez-vous que votre ordinateur ait la date et l'heure précise, ainsi que le fuseau horaire approprié, puisqu'ils seront utilisés pour identifier les données de votre station météo. Si vous ne l'avez pas déjà fait, il pourrait être utile d'utiliser la synchronisation horaires par réseau (NTP) pour synchroniser votre ordinateur à un 'serveur de temps'.

La première fois que vous exécutez *pywws.LogData* un fichier de configuration, nommé 'weather.ini', sera créé dans votre répertoire de données, puis s'arrêtera. Vous devez éditer ce fichier de configuration et y modifier la ligne `ws type = Unknown` à `ws type = 1080` ou `ws type = 3080`. (Si la console de votre station météo affiche la luminosité solaire, vous avez une station de type 3080, sinon, vous indiquez le type 1080.) Puis exécutez *pywws.LogData* de nouveau. Cette fois, l'exécution peut durer quelques minutes, puisqu'il copiera l'intégralité des données météo emmagasinées dans la mémoire de votre station météo. Le programme *pywws.LogData* possède une option 'verbose' qui augmente le nombre de messages affichés pendant l'exécution. Cette option est utile lors d'exécution manuelle, par exemple :

```
python -m pywws.LogData -vvv ~/weather/data
```

(Remplacez `~/weather/data` par votre répertoire de données, si différent.)

Vous devriez maintenant avoir quelques fichiers de données à regarder. Par exemple :

```
more ~/weather/data/raw/2012/2012-12/2012-12-16.txt
```

(Remplacez année, mois et jour par la date pour laquelle vous devriez avoir des données.)

Convertir les anciennes données de EasyWeather (optionnel).

Si vous utilisiez EasyWeather avant de décider d'utiliser pywws, vous pouvez convertir les données que EasyWeather a enregistré, vers format de pywws. Localisez votre fichier EasyWeather.dat et convertissez-le ainsi :

```
python -m pywws.EWtoPy EasyWeather.dat ~/weather/data
```

Indiquer certaines options de configuration.

Après avoir exécuté *pywws.LogData*, il devrait y avoir un fichier de configuration nommé 'weather.ini' dans votre répertoire de données. Ouvrez ce fichier avec un éditeur de texte. Vous devriez y trouver des lignes semblables à celle-ci :

```
[config]
ws type = 1080
logdata sync = 1
pressure offset = 9.4
```

Vous devez ajouter une nouvelle entrée nommée `day end hour` dans la section `[config]`. Ceci indique à pywws quelle convention vous souhaitez utiliser pour le calcul du sommaire des données quotidiennes. En Grande-Bretagne, le 'jour météorologique' s'étale habituellement de 09 :00 à 09 :00 GMT (10 :00 à 10 :00 BST durant l'été), j'utilise donc 9 pour l'heure de fin du jour. Dans d'autres pays une valeur de 24 (ou 0) peut être souhaitable. Noter que cette valeur est définie à l'heure hivernale locale. Vous ne devriez pas avoir besoin de changer cette valeur pendant la saison chaude.

Après édition, votre fichier weather.ini devrait resembler à ceci :

```
[config]
ws type = 1080
logdata sync = 1
pressure offset = 9.4
day end hour = 9
```

You can also edit the `pressure offset` value to adjust how pywws calculates the relative (sea level) air pressure from the absolute value that the station measures. If you change the pressure offset or day end hour in future, you must update all your stored data by running `pywws.Reprocess`.

Pour plus de détails sur les options du fichier de configuration, voir weather.ini - format du fichier de configuration.

Modifié dans la version 13.10_r1082 : made `pressure offset` a config item. Previously it was always read from the weather station.

Traitement des données brutes.

pywws.LogData ne fait que copier les données brutes à partir de la station météo. Pour faire quelque chose d'utile avec ces données vous avez probablement besoin des sommaires horaire, quotidien et mensuel. Ces sommaires sont produits à partir du programme `pywws.Process`. Par exemple :

```
python -m pywws.Process ~/weather/data
```

Vous devriez avoir ainsi quelques fichiers traités à consulter :

```
more ~/weather/data/daily/2012/2012-12-16.txt
```

Si vous changez votre période de fin du jour avec l'option de configuration `day end hour`, vous devrez re-traiter toutes vos données météo. Pour ce faire, exécutez le programme `pywws.Reprocess` :

```
python -m pywws.Reprocess ~/weather/data
```

Vous êtes maintenant prêt pour fixer une journalisation régulière ou continue, tel que décrit dans la section Comment configurer la journalisation horaire avec pywws ou Comment configurer le mode 'live' avec pywws.

Lire la documentation.

You're looking at it right now ! The Guides utilisateur section is probably the most useful bit to read first, but the Programmes et modules Python section has a lot more detail on the various pywws modules and commands.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Comment configurer la journalisation horaire avec pywws

Introduction

There are two quite different modes of operation with pywws. Traditionally `pywws.Hourly` would be run at regular intervals (usually an hour) from cron. This is suitable for fairly static websites, but more frequent updates can be useful for sites such as Weather Underground (http://www.wunderground.com/). The newer `pywws.LiveLog` program runs continuously and can upload data every 48 seconds.

Note that although this document (and the program name) refers to 'hourly' logging, you can run `pywws.Hourly` as often or as infrequently as you like, but don't try to run it more often than double your logging interval. For example, if your logging interval is 10 minutes, don't run `pywws.Hourly` more often than every 20 minutes.

Mise en route

Avant tout, vous devez installer pywws et vous assurer qu'il reçoit bien les informations de votre station météo. Voir Comment démarrer avec pywws pour plus de détails.

Try running `pywws.Hourly` from the command line, with a high level of verbosity so you can see what's happening. Use the `pywws-hourly` command to run `pywws.Hourly` :

```
pywws-hourly -vvv ~/weather/data
```

En moins de cinq minutes (assumant que votre intervalle de relevé soit de 5 minutes) vous devriez voir le message 'live_data new ptr', suivi par la recherche et le traitement de nouvelles données de la station météorologique.

Modifié dans la version 14.04.dev1194 : the `pywws-hourly` command replaced `scripts/pywws-hourly.py`.

Configurer l'emplacement des fichiers

Ouvrez votre fichier weather.ini avec un éditeur de texte. Vous devriez avoir une section `[paths]` similaire à ce qui suit (où `xxx` est votre nom d'usager) :

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
local_files = /home/xxx/weather/results/
```

Edit these to suit your installation and preferences. `work` is an existing temporary directory used to store intermediate files, `templates` is the directory where you keep your text template files, `graph_templates` is the directory where you keep your graph template files and `local_files` is a directory where template output that is not uploaded to your web site is put. Don't use the pywws example directories for your templates, as they will get over-written when you upgrade pywws.

Copy your text and graph templates to the appropriate directories. You may find some of the examples provided with pywws useful to get started. The `pywws-version -v` command should show you where the examples are on your computer.

Nouveau dans la version 14.04.dev1194 : the `pywws-version` command.

Configurer les tâches périodiques

Dans weather.ini vous devriez avoir les sections `[logged]`, `[hourly]`, `[12 hourly]` et `[daily]` similaires à celle-ci :

```
[logged]
services = []
plot = []
text = []

[hourly]
...
```

These specify what `pywws.Hourly` should do when it is run. Tasks in the `[logged]` section are done every time there is new logged data, tasks in the `[hourly]` section are done every hour, tasks in the `[12 hourly]` section are done twice daily and tasks in the `[daily]` section are done once per day.

The `services` entry is a list of online weather services to upload data to. The `plot` and `text` entries are lists of template files for plots and text files to be processed and, optionally, uploaded to your web site. Add the names of your template files and weather services to the appropriate entries, for example :

```
[logged]
services = ['underground', 'metoffice']
plot = []
text = []

[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
plot = []
text = []

[daily]
services = []
```

```
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'allmonths.txt']
```

Note the use of the `'T'` flag – this tells pywws to send the template result to Twitter instead of uploading it to your ftp site.

You can test that all these are working by removing the `[last update]` section from status.ini, then running `pywws.Hourly` again :

```
pywws-hourly -v ~/weather/data
```

Nouveau dans la version 14.05.dev1211 : `[cron name]` sections. If you need more flexibility in when tasks are done you can use `[cron name]` sections. See weather.ini - format du fichier de configuration for more detail.

Modifié dans la version 13.06_r1015 : added the `'T'` flag. Previously Twitter templates were listed separately in `twitter` entries in the `[hourly]` and other sections. The older syntax still works, but is deprecated.

Modifié dans la version 13.05_r1009 : the last update information was previously stored in weather.ini, with `last update` entries in several sections.

Exécuter en tant que tâche 'cron'

Most UNIX/Linux systems have a 'cron' daemon that can run programs at certain times, even if you are not logged in to the computer. You edit a 'crontab' file to specify what to run and when to run it. For example, to run `pywws.Hourly` every hour, at zero minutes past the hour :

```
0 * * * *       pywws-hourly /home/xxx/weather/data
```

This might work, but if it didn't you probably won't get any error messages to tell you what went wrong. It's much better to run a script that runs `pywws.Hourly` and then emails you any output it produces. Here's the script I use :

```
#!/bin/sh
#
# weather station logger calling script

export PATH=$PATH:/usr/local/bin

if [ ! -d ~/weather/data/ ]; then
  exit
  fi

log=/var/log/log-weather

pywws-hourly -v ~/weather/data >$log 2>&1

# mail the log file
/home/jim/scripts/email-log.sh $log "weather log"
```

Vous devrez éditer beaucoup pour adapter à vos emplacements de dossier et ainsi de suite, mais il donne une certaine idée de ce qui peut être fait.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Comment configurer le mode 'live' avec pywws

Introduction

There are two quite different modes of operation with pywws. Traditionally `pywws.Hourly` would be run at regular intervals (usually an hour) from cron. This is suitable for fairly static websites, but more frequent updates can be useful for sites such as Weather Underground (http://www.wunderground.com/). The newer `pywws.LiveLog` program runs continuously and can upload data every 48 seconds.

Mise en route

Avant tout, vous devez installer pywws et vous assurer qu'il reçoit bien les informations de votre station météo. Voir Comment démarrer avec pywws pour plus de détails.

If you have previously been using `pywws.Hourly` then disable your 'cron' job (or whatever else you use to run it) so it no longer runs. You should not run `pywws.Hourly` and `pywws.LiveLog` at the same time.

Try running `pywws.LiveLog` from the command line, with a high level of verbosity so you can see what's happening. Use the `pywws-livelog` command to run `pywws.LiveLog` :

```
pywws-livelog -vvv ~/weather/data
```

Within five minutes (assuming you have set a 5 minute logging interval) you should see a 'live_data new ptr' message, followed by fetching any new data from the weather station and processing it. Let `pywws.LiveLog` run for a minute or two longer, then kill the process by typing '<Ctrl>C'.

Modifié dans la version 14.04.dev1194 : the `pywws-livelog` command replaced `scripts/pywws-livelog.py`.

Configurer l'emplacement des fichiers

Ouvrez votre fichier weather.ini avec un éditeur de texte. Vous devriez avoir une section `[paths]` similaire à ce qui suit (où `xxx` est votre nom d'usager) :

```
[paths]
work = /tmp/weather
templates = /home/xxx/weather/templates/
graph_templates = /home/xxx/weather/graph_templates/
local_files = /home/xxx/weather/results/
```

Edit these to suit your installation and preferences. `work` is an existing temporary directory used to store intermediate files, `templates` is the directory where you keep your text template files, `graph_templates` is the directory where you keep your graph template files and `local_files` is a directory where template output that is not uploaded to your web site is put. Don't use the pywws example directories for your templates, as they will get over-written when you upgrade pywws.

Copy your text and graph templates to the appropriate directories. You may find some of the examples provided with pywws useful to get started. The `pywws-version -v` command should show you where the examples are on your computer.

Nouveau dans la version 14.04.dev1194 : the `pywws-version` command.

Configurer les tâches périodiques

Dans weather.ini vous devriez avoir une section `[live]` similaire à celle-ci :

```
[live]
services = []
plot = []
text = []
```

This section specifies what pywws should do every time it gets a new reading from the weather station, i.e. every 48 seconds. The `services` entry is a list of online weather services to upload data to, e.g. `['underground_rf']`. The `plot` and `text` entries are lists of template files for plots and text files to be processed and, optionally, uploaded to your web site. You should probably leave all of these blank except for `services`.

Si vous utilisez YoWindow (http://yowindow.com/) vous pouvez ajouter l'entrée à la section `[live]` pour spécifier votre fichier YoWindow, ex. :

```
[live]
services = ['underground_rf']
text = [('yowindow.xml', 'L')]
...
```

Note the use of the `'L'` flag – this tells pywws to copy the template result to your "local files" directory instead of uploading it to your ftp site.

Si vous ne les avez pas déjà, créez quatre sections supplémentaires dans votre fichier weather.ini : `[logged]`, `[hourly]`, `[12 hourly]` et `[daily]`. Ces sections doivent avoir des entrées similaires à la section `[live]`, et spécifiez ce qui doit être fait chaque fois qu'une donnée est enregistrée (5 à 30 minutes, dépendant de votre intervalle), chaque heure, deux fois par jour et chaque jour. Ajoutez les noms de vos fichiers de gabarit à l'entrée appropriée, par exemple :

```
[logged]
services = ['underground', 'metoffice']
plot = []
text = []

[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_24hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']

[12 hourly]
services = []
plot = []
text = []

[daily]
services = []
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'allmonths.txt']
```

Note the use of the `'T'` flag – this tells pywws to send the template result to Twitter instead of uploading it to your ftp site.

Nouveau dans la version 14.05.dev1211 : `[cron name]` sections. If you need more flexibility in when tasks are done you can use `[cron name]` sections. See weather.ini - format du fichier de configuration for more detail.

Modifié dans la version 13.06_r1015 : added the `'T'` flag. Previously Twitter templates were listed separately in `twitter` entries in the `[hourly]` and other sections. The older syntax still works, but is deprecated.

Modifié dans la version 13.05_r1013 : added a `'yowindow.xml'` template. Previously yowindow files were generated by a separate module, invoked by a `yowindow` entry in the `[live]` section. This older syntax still works, but is deprecated.

Asynchronous uploads

Nouveau dans la version 13.09_r1057.

Uploading data to web sites or 'services' can sometimes take a long time, particularly if a site has gone off line and the upload times out. In normal operation pywws waits until all uploads have been processed before fetching any more data from the weather station. This can lead to data sometimes being missed.

The `asynchronous` item in the `[config]` section of weather.ini can be set to `True` to tell `pywws.LiveLog` to do these uploads in a separate thread.

Exécuter en arrière-plan

Nouveau dans la version 13.12.dev1118.

In order to have `pywws.LiveLog` carry on running after you finish using your computer it needs to be run as a "background job". On most Linux / UNIX systems you can do this by putting an ampersand ('&') at the end of the command line. Running a job in the background like this doesn't always work as expected : the job may suspend when you log out. It's much better to run as a proper UNIX 'daemon' process.

The `pywws.livelogdaemon` program does this, if you have the python-daemon library installed :

```
pywws-livelog-daemon -v ~/weather/data ~/weather/data/pywws.log start
```

Note that the log file is a required parameter, not an option.

Redémarrage automatique

There are various ways of configuring a Linux system to start a program when the machine boots up. Typically these involve putting a file in `/etc/init.d/`, which requires root privileges. A slightly harder problem is ensuring a program restarts if it crashes. My solution to both problems is to run the following script from cron, several times an hour.

```
#!/bin/sh

export PATH=$PATH:/usr/local/bin

# exit if NTP hasn't set computer clock
[ `ntpdc -c sysinfo | awk '/stratum:/ {print $2}'` -ge 10 ] && exit

pidfile=/var/run/pywws.pid
datadir=/home/jim/weather/data
logfile=$datadir/live_logger.log

# exit if process is running
[ -f $pidfile ] && kill -0 `cat $pidfile` && exit

# email last few lines of the logfile to see why it died
if [ -f $logfile ]; then
  log=/tmp/log-weather
  tail -40 $logfile >$log
```

```
  /home/jim/scripts/email-log.sh $log "weather log"
  rm $log
  fi

# restart process
pywws-livelog-daemon -v -p $pidfile $datadir $logfile start
```

The process id of the daemon is stored in `pidfile`. If the process is running, the script does nothing. If the process has crashed, it emails the last 40 lines of the log file to me (using a script that creates a message and passes it to sendmail) and then restarts `pywws.livelogdaemon`. You'll need to edit this quite a lot to suit your file locations and so on, but it gives some idea of what to do.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Comment intégrer divers services météorologiques à pywws

Ce guide donne de brèves instructions sur comment utiliser pywws avec d'autres services météorologiques et logiciels. Il n'est pas exhaustive, et certains services (comme Twitter) sont couverts en détail ailleurs.

YoWindow

YoWindow est un widget météo qui peut afficher des données provenant d'une source internet ou de votre station météo. Pour afficher les données de votre station, pywws a besoin d'écrire dans un fichier local, en général toutes les 48 secondes lorsque de nouvelles données sont reçues. C'est facile à faire :

1. Arrêter toute instance de pywws
2. Copier le gabarit exemple `yowindow.xml` dans votre répertoire de gabarits text.
3. Si ce n'est déjà fait, éditez le fichier `weather.ini` et configurez la clé `local_files` dans la section `[paths]` pour un répertoire approprié pour votre fichier yowindow.
4. Ajoutez le modèle yowindow dans les tâches `[live]` de `weather.ini`. Mettre son paramètre à `'L'` si le résultat est copié dans votre répertoire local au lieu d'être téléchargés sur un site ftp :

```
    [live]
    text = [('yowindow.xml', 'L')]
```

5. Redémarrez l'enregistrement live de pywws.

Vous pouvez vérifier que le fichier est mis à jour toutes les 48 secondes en utilisant `more` ou `cat` pour afficher à l'écran.

Finallement, configurez yowindow pour l'utilisation de ce fichier. Voir http://yowindow.com/pws_setup.php pour les instructions pour ce faire.

Twitter

Voir Comment configurer pywws pour poster des messages sur Twitter pour les instructions détaillées.

Other "services"

The remaining weather service uploads are handled by the `pywws.toservice` module. See the module's documentation for general configuration options. The following subsections give further information about some of the available services.

Citizen Weather Observer Program Nouveau dans la version 14.02.dev1156.

— Web site : http://www.wxqa.com/
— Create account : http://www.wxqa.com/SIGN-UP.html
— API : http://www.wxqa.com/faq.html
— Example `weather.ini` section :

```
[cwop]
designator = EW9999
latitude = 5130.06N
longitude = 00008.52E
template = default

[logged]
services = ['cwop', 'underground']

[live]
services = ['cwop', 'underground_rf']
```

or, for radio hams :

```
[cwop]
designator = G4XXX
passcode = xxxxxx
latitude = 5130.06N
longitude = 00008.52E
template = default

[logged]
services = ['cwop_ham', 'underground']

[live]
services = ['cwop_ham', 'underground_rf']
```

Note that the latitude and longitude must be in "LORAN" format and leading zeros are required. See question 3 in the CWOP FAQ for more information.

Licensed radio hams use their callsign as the designator and need a passcode. They should use the service name `cwop_ham` instead of `cwop` when running `pywws.toservice` directly and in the `weather.ini services` entries. (The same `[cwop]` config section is used for both.)

CWOP uploads are rate-limited by pywws, so you can safely add it to both the `[live]` and `[logged]` sections in `weather.ini`.

The CWOP/APRS uploader is based on code by Marco Trevisan <mail@3v1n0.net>.

MQTT Nouveau dans la version 14.12.0.dev1260.

MQTT is a "message broker" system, typically running on `localhost` or another computer in your home network. Use of MQTT with pywws requires an additional library. See *Dependencies - MQTT* for details.

— MQTT : http://mqtt.org/
— Mosquitto (a lightweight broker) : http://mosquitto.org/
— Example `weather.ini` section :

```
[mqtt]
topic = /weather/pywws
hostname = localhost
port = 1883
client_id = pywws
retain = True
```

```
auth = False
user = unknown
password = unknown
template = default

[logged]
services = ['mqtt', 'underground']
```

pywws will publish a JSON string of the data specified in the `mqtt_template_1080.txt` file. This data will be published to the broker running on hostname, with the port number specified. (An IP address can be used instead of a host name.) `client_id` is a note of who published the data to the topic. `topic` can be any string value, this needs to be the topic that a subscriber is aware of.

`retain` is a boolean and should be set to `True` or `False` (or left at the default `unknown`). If set to `True` this will flag the message sent to the broker to be retained. Otherwise the broker discards the message if no client is subscribing to this topic. This allows clients to get an immediate response when they subscribe to a topic, without having to wait until the next message is published.

`auth`, `user` and `password` can be used for MQTT authentication.

If these aren't obvious to you it's worth doing a bit of reading around MQTT. It's a great lightweight messaging system from IBM, recently made more popular when Facebook published information on their use of it.

This has been tested with the Mosquitto Open Source MQTT broker, running on a Raspberry Pi (Raspian OS). TLS (mqtt data encryption) is not yet implemented.

Thanks to Matt Thompson for writing the MQTT code and to Robin Kearney for adding the retain and auth options.

UK Met Office

- Web site : http://wow.metoffice.gov.uk/
- Create account : https://register.metoffice.gov.uk/WaveRegistrationClient/public/newaccount.do?service=weatherobservations
- API : http://wow.metoffice.gov.uk/support/dataformats#automatic
- Example `weather.ini` section :

```
[metoffice]
site id = 12345678
aws pin = 987654
template = default

[logged]
services = ['metoffice', 'underground']
```

Open Weather Map

- Web site : http://openweathermap.org/
- Create account : http://home.openweathermap.org/users/sign_up
- API : http://openweathermap.org/stations#trans
- Example `weather.ini` section :

```
[openweathermap]
lat = 51.501
long = -0.142
alt = 10
user = ElizabethWindsor
password = corgi
id = Buck House
template = default
```

```
    [logged]
    services = ['openweathermap', 'underground']
```

When choosing a user name you should avoid spaces (and probably non-ascii characters as well). Having a space in your user name causes strange "internal server error" responses from the server.

The default behaviour is to use your user name to identify the weather station. However, it's possible for a user to have more than one weather station, so there is an optional `name` parameter in the API that can be used to identify the station. This appears as `id` in `weather.ini`. Make sure you choose a name that is not already in use.

PWS Weather

- Web site : http://www.pwsweather.com/
- Create account : http://www.pwsweather.com/register.php
- API based on WU protocol : http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol
- Example `weather.ini` section :

```
    [pwsweather]
    station = ABCDEFGH1
    password = xxxxxxx
    template = default

    [logged]
    services = ['pwsweather', 'underground']
```

temperatur.nu

- Web site : http://www.temperatur.nu/
- Example `weather.ini` section :

```
    [temperaturnu]
    hash = ???
    template = default

    [logged]
    services = ['temperaturnu', 'underground']
```

You receive the hash value from the temperatur.nu admins during sign up. It looks like "d3b07384d113edec49eaa6238ad5ff00".

Weather Underground

- Create account : http://www.wunderground.com/members/signup.asp
- API : http://wiki.wunderground.com/index.php/PWS_-_Upload_Protocol
- Example `weather.ini` section :

```
    [underground]
    station = ABCDEFGH1
    password = xxxxxxx
    template = default

    [logged]
    services = ['underground', 'metoffice']
```

Weather Underground "RapidFire" updates Weather Underground a une seconde URL de téléversement pour les mises à jour en temps réel à un interval d'aussi peu que 2.5 secondes. Si vous exécutez pywws en mode 'live logging' (voir Comment configurer le mode 'live' avec pywws) vous pouvez l'utiliser pour envoyez des mises à jour aux 48 secondes, en ajoutant 'underground_rf' à la section des tâches `[live]` du fichier `weather.ini` :

```
[underground]
station = ABCDEFGH1
password = xxxxxxx
template = default

[live]
services = ['underground_rf']

[logged]
services = ['underground', 'metoffice']
```

Make sure you still have an 'underground' service in `[logged]` or `[hourly]`. This will ensure that 'catchup' records are sent to fill in any gaps if your station goes offline for some reason.

wetter.com

— Web site : http://www.wetter.com/wetter_aktuell/wetternetzwerk/
— Register station : http://www.wetter.com/mein_wetter/wetterstation/willkommen/
— Example `weather.ini` section :

```
[wetterarchivde]
user_id = 12345
kennwort = ab1d3456i8
template = default

[logged]
services = ['wetterarchivde', 'underground']

[live]
services = ['wetterarchivde', 'underground_rf']
```

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Comment configurer pywws pour poster des messages sur Twitter

Fichiers prérequis.

Postersur Twitter requière certains logiciels supplémentaires. Voir Pré-requis - *Mise à jour Twitter*.

Créer un compte Twitter.

Vous pouvez poster des mise à jour météo sur votre compte Twitter 'normal', mais je crois qu'il est préférable d'avoir un compte distinct réservé à vos relevés météo. Ce qui pourrait être fort utile à une personne vivant dans votre voisinage, mais ne souhaitant pas savoir ce que vous avez pris au petit déjeuner.

Autoriser pywws à poster sur votre compte Twitter.

Si vous exécutez pywws sur un appareil de faible puissance tel qu'un routeur, il peut plus simple d'exécuter cette étape d'autorisation sur un autre ordinateur en autant que `python-oauth2` y est installé. Utiliser un répertoire 'data' vide – un fichier `weather.ini` sera créé, dont le contenu pourra être copié dans votre fichier `weather.ini` réel en utilisant tout éditeur de texte.

Assurez vous qu'aucune autre instance de pywws ne fonctionne, puis exécutez le module *TwitterAuth* :

```
python -m pywws.TwitterAuth ~/weather/data
```

(Replace `~/weather/data` with your data directory.)

Ceci ouvrira une fenêtre de navigation (ou vous donnera une URL à copier dans votre navigateur) où vous pourrez vous connecter à votre compte Twitter et autoriser pywws à poster. Votre navigateur vous affichera alors un nombre à 7 chiffre que vous aurez besoin de copier dans le programme *TwitterAuth*. Si tout s'est bien déroulé, votre fichier `weather.ini` devrait comporter une nouvelle section `[twitter]` avec les entrées `secret` et `key`. (Ne révéler à quiconque.)

Ajouter vos données de localisation (optionel).

Éditez votre fichier `weather.ini` et ajoutez les entrées `latitude` et `longitude` à votre section `[twitter]`. Par exemple :

```
[twitter]
secret = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
key = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
latitude = 51.501
longitude = -0.142
```

Créer un gabarit.

Les messages pour Twitter sont générés en utilisant un gabarit, tout comme sont créés les fichiers à téléverser sur votre site web. Copiez le gabarit exemple 'tweet.txt' dans votre dossier de gabarit, puis testez-le :

```
python -m pywws.Template ~/weather/data ~/weather/templates/tweet.txt tweet.txt
cat tweet.txt
```

(Replace `~/weather/data` and `~/weather/templates` with your data and template directories.) If you need to change the template (e.g. to change the units or language used) you can edit it now or later.

Poster votre premier Tweet météo.

Maintenant tout est pret pour exécuter *ToTwitter* :

```
python -m pywws.ToTwitter ~/weather/data tweet.txt
```

Si celà fonctionne, Votre nouveau compte Twitter aura posté son premier relevé météo. (Vous devriez supprimer le fichier tweet.txt maintenant.)

Ajouter les envois sur Twitter à vos tâche horaire.

Éditez votre fichier `weather.ini` et modifiez la section `[hourly]`. Par exemple :

```
[hourly]
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']
```

Notez l'usage de l'indicateur `'T'` – ce qui indique à pywws de `'Tweeter'` le résultat du gabarit au lieu de le verser sur votre site par ftp.

Vous pourriez tout aussi bien choisir de plutôt modifier les sections `[logged]`, `[12 hourly]` ou `[daily]`, mais je crois que `[hourly]` est le plus approprié pour les envois sur Twitter.

Modifié dans la version 13.06_r1015 : added the `'T'` flag. Previously Twitter templates were listed separately in `twitter` entries in the `[hourly]` and other sections. The older syntax still works, but is deprecated.

Include an image in your tweet

Nouveau dans la version 14.05.dev1216.

You can add an image to your tweets by specifying an image file location in the tweet template. Make the first line of the tweet `media path` where `path` is the absolute location of the file. The "tweet_media.txt" example template shows how to do this.

The image could be from a web cam, or for a weather forecast it could be an icon representing the forecast. To add a weather graph you need to make sure the graph is drawn before the tweet is sent. I do this by using two `[cron xxx]` sections in weather.ini :

```
[cron prehourly]
format = 59 * * * *
services = []
plot = [('tweet.png.xml', 'L')]
text = []

[cron hourly]
format = 0 * * * *
services = []
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet_media.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt']
```

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Comment utiliser pywws dans une autre langue

Introduction

Some parts of pywws can be configured to use your local language instead of British English. This requires an appropriate language file which contains translations of the various strings used in pywws. The pywws project relies on users to provide these translations.

La documentation de pywws peut aussi être traduite dans d'autres langues. C'est beaucoup de travail, mais pourrait être très utile aux utilisateurs potentiels qui ne lisent pas très bien l'anglais.

Using existing language files

Program strings There may already be a pywws translation for your preferred language. First you need to choose the appropriate two-letter code from the list at http://www.w3schools.com/tags/ref_language_codes.asp. For example, `fr` is the code for French. Now use the *`pywws.Localisation`* module to do a quick test :

```
python -m pywws.Localisation -t fr
```

Cela devrait afficher quelque chose comme ceci :

```
Locale changed from (None, None) to ('fr_FR', 'UTF-8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: lundi, 17 décembre (17/12/2012 16:00:48)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
```

This shows that pywws is already able to generate French output, and that your installation is correctly configured. Now you can edit the `language` entry in your *weather.ini* file to use your language code.

If the above test shows no translations into your language then you need to create a new language file, as described below.

Text encodings The pywws default text encoding is ISO-8859-1, also known as Latin-1. This is suitable for several western European languages but not for some others. If you encounter problems you may need to use a different encoding. See the documentation of `pywws.Template` and `pywws.Plot` for more details.

Documentation If you have downloaded the pywws source files, or cloned the GitHub repository (see *how to get started with pywws*), you can compile a non-English copy of the documentation. This requires the Sphinx package, see *dependencies*.

First delete the old documentation (if it exists) and then recompile using your language :

```
cd ~/weather/pywws
rm -rf doc
LANG=fr python setup.py build_sphinx
```

Note that the `build_sphinx` command doesn't have a `--locale` (or `-l`) option, so the language is set by a temporary environment variable.

You can view the translated documentation by using a web browser to read the file `~/weather/pywws/doc/html/index.html`.

Writing new language files

There are two ways to write new language files (or update existing ones) – use the Transifex online system or use local files. Transifex is preferred as it allows several people to work on the same language, and makes your work instantly available to others.

To test your translation you will need to have downloaded the pywws source files, or cloned the GitHub repository (see *how to get started with pywws*). You will also need to install the `Babel` package, see *dependencies*.

Using Transifex If you'd like to use Transifex, please go to the pywws Transifex project, click on "help translate pywws" and create a free account.

Visit the pywws project page on Transifex and click on your language, then click on the "resource" you want to translate. (`pywws` contains the program strings used when running pywws, the others contain strings from the pywws documentation.) This opens a dialog where you can choose to translate the strings online. Please read *Notes for translators* before you start.

When you have finished translating you should use the `transifex-client` program (see *dependencies*) to download files for testing. For example, this command downloads any updated files for the French language :

```
cd ~/weather/pywws
tx pull -l fr
```

Now you are ready to *Test the pywws translations*.

Using local files If you prefer not to use the Transifex web site you can edit language files on your own computer. This is done in two stages, as follows.

Extract source strings Program messages are extracted using the `Babel` package :

```
cd ~/weather/pywws
mkdir -p build/gettext
python setup.py extract_messages
```

This creates the file `build/gettext/pywws.pot`. This is a "portable object template" file that contains the English language strings to be translated.

The documentation strings are extracted using the `Sphinx` package :

```
cd ~/weather/pywws
python setup.py extract_messages_doc
```

This creates several `.pot` files in the `build/gettext/` directory.

Create language files The `sphinx-intl` command is used to convert the `.pot` files to language specific `.po` files :

```
cd ~/weather/pywws
sphinx-intl update --locale-dir src/pywws/lang -p build/gettext -l fr
```

Now you can open the `.po` files in `src/pywws/lang/fr/LC_MESSAGES/` with your favourite text editor and start filling in the empty `msgstr` strings with your translation of the corresponding `msgid` string. Please read *Notes for translators* before you start.

Test the pywws translations

The `Babel` package is used to compile program strings :

```
python setup.py compile_catalog --locale fr
```

(Replace `fr` with the code for the language you are testing.)

After compilation you can test the translation :

```
python setup.py build
sudo python setup.py install
python -m pywws.Localisation -t fr
```

Sphinx is used to build the translated documentation :

```
cd ~/weather/pywws
rm -rf doc
LANG=fr python setup.py build_sphinx
```

You can view the translated documentation by using a web browser to read the file ~/weather/pywws/doc/html/index.html.

Notes for translators

The pywws program strings (pywws.po) are quite simple. They comprise simple weather forecasts ("Fine weather"), air pressure changes ("rising quickly") and the 16 points of the compass ("NNE"). Leave the "(%Z)" in "Time (%Z)" unchanged and make sure your translation's punctuation matches the original.

The other files contain strings from the pywws documentation. These are in reStructuredText. This is mostly plain text, but uses characters such as backquotes (‘), colons (:) and asterisks (*) for special purposes. You need to take care to preserve this special punctuation. Do not translate program source, computer instructions and cross-references like these :

```
`pip <http://www.pip-installer.org/>`_
:py:class:`datetime.datetime`
:obj:`ParamStore <pywws.DataStore.ParamStore>`\\ (root_dir, file_name)
pywws.Forecast
``pywws-livelog``
```

Translating all of the pywws documentation is a lot of work. However, when the documentation is "compiled" any untranslated strings revert to their English original. This means that a partial translation could still be useful – I suggest starting with the documentation front page, index.po.

Envoyer la traduction à Jim

I'm sure you would like others to benefit from the work you've done in translating pywws. If you've been using Transifex then please send me an email (jim@jim-easterbrook.me.uk) to let me know there's a new translation available. Otherwise, please email me any .po files you create.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

weather.ini - format du fichier de configuration

Presque toute la configuration de pywws est fait via un seul fichier dans le répertoire de donnée : weather.ini. Ce fichier a été structuré de manière similaire aux fichiers INI de Microsoft Windows. Il est divisé en ‘sections’, qui contiennent un nom d'entrée ‘nom = valeur’. L'ordre d'apparition des sections n'a pas d'importance.

Tout éditeur de texte brut peut être utilisé pour éditer le fichier. (Ne pas essayer d'éditer le fichier pendant le fonctionnement de pywws.) Dans plusieurs cas, pywws initialisera les entrées avec des valeurs importantes.

Un autre fichier, status.ini, est utilisé pour emmagasiner certaines informations que pywws utilise à l'interne. Il est décrit à la fin de ce document. En utilisation normale, vous ne devriez pas avoir à éditer ce fichier.

Les sections suivantes sont présentement utilisées :

- config : diverses configuration système.
- paths : dossiers dans lesquels sont emmagasinés les gabarits, etc.
- live : tâches à accomplir à chaque 48 secondes.
- logged : tâches à accomplir chaque fois que la station enregistre une lecture.
- cron : tasks to be done at a particular time or date.
- hourly : tâches à réaliser à chaque heure.
- 12 hourly : tâches à réaliser à chaque 12 heures.
- daily : tâches à faire chaque jour.
- ftp : configuration pour le téléversement du un site web.
- twitter : configuration pour poster sur Twitter.
- underground, metoffice, temperaturnu etc : configuration pour poster sur ces 'services'.

config : diverses configuration système

```
[config]
ws type = 1080
day end hour = 21
pressure offset = 9.4
gnuplot encoding = iso_8859_1
template encoding = iso-8859-1
language = en
logdata sync = 1
rain day threshold = 0.2
asynchronous = False
usb activity margin = 3.0
gnuplot version = 4.2
frequent writes = False
```

`ws type` est le "type" de station. Il devrait être fixé à `1080` pour la plupart des stations météo, ou `3080` si la console de votre station affiche l'éclairement solaire.

`day end hour` is the end of the "meteorological day", in local time without daylight savings time. Typical values are 21, 9, or 24. You must update all your stored data by running `pywws.Reprocess` after you change this value.

`pressure offset` is the difference between absolute and relative (sea level) air pressure. The initial value is copied from the weather station, assuming you have set it up to display the correct relative pressure, but you can adjust the value in weather.ini to calibrate your station. You must update all your stored data by running `pywws.Reprocess` after you change this value.

Modifié dans la version 13.10_r1082 : made `pressure offset` a config item. Previously it was always read from the weather station.

`gnuplot encoding` est l'encodage de texte utilisé lors du tracé de graphes. La valeur par défaut est `iso_8859_1` permettant le symbole des degrés, fort utile dans une application météorologique ! D'autres valeurs peuvent être nécessaire si votre langue comporte des caractères accentués. Les valeurs possible dépendent de votre installation gnuplot, quelques expérimentations peuvent donc être nécessaire.

`template encoding` est l'encodage de text utilisé pour le gabarit. La valeur par défaut est `iso-8859-1`, ce qui est l'encodage utilisé dans les gabarits d'exemple. Si vous créez des gabarits avec un autre jeu de caractère, vous devez changer cette valeur pour celle de vos gabarits.

`language` est utilisé pour traduire pywws. Il est optionnel, puisque pywws utilise habituellement la langue par défaut de l'ordinateur tel que configuré par la variable d'environnement LANG. Vous trouverez les langages disponibles dans

le sous-dossier `translations` de votre installation pywws. Si vous configurez une langue qui n'est pas inclue, pywws affichera en anglais.

`logdata sync` configure la qualité de la synchronisation utilisé par pywws.LogData. Fixer à 0 pour rapide & imprécis ou 1 pour plus lent, mais précis.

`rain day threshold` is the amount of rain (in mm) that has to fall in one day for it to qualify as a rainy day in the monthly summary data. You must update all your stored data by running `pywws.Reprocess` after you change this value.

Nouveau dans la version 13.09_r1057 : `asynchrouous` controls the use of a separate upload thread in `pywws.LiveLog`.

Nouveau dans la version 13.10_r1094 : `usb activity margin` controls the algorithm that avoids the "USB lockup" problem that affects some stations. It sets the number of seconds either side of expected station activity (receiving a reading from outside or logging a reading) that pywws does not get data from the station. If your station is not affected by the USB lockup problem you can set `usb activity margin` to 0.0.

Nouveau dans la version 13.11_r1102 : `gnuplot version` tells `pywws.Plot` and `pywws.WindRose` what version of gnuplot is installed on your computer. This allows them to use version-specific features to give improved plot quality.

Nouveau dans la version 14.01_r1133 : `frequent writes` tells `pywws.Tasks` to save weather data and status to file every time there is new logged data. The default is to save the files every hour, to reduce "wear" on solid state memory such as the SD cards used with Raspberry Pi computers. If your weather data directory is stored on a conventional disc drive you can set `frequent writes` to `True`.

paths : dossiers dans lesquels sont emmagasinés les gabarits, etc.

```
[paths]
templates = /home/$USER/weather/templates/
graph_templates = /home/$USER/weather/graph_templates/
user_calib = /home/$USER/weather/modules/usercalib
work = /tmp/weather
local_files = /home/$USER/weather/results/
```

These entries specify where your text templates and graph templates are stored, where temporary files should be created, where template output (that is not uploaded) should be put, and (if you have one) the location of your calibration module.

live : tâches à accomplir à chaque 48 secondes

```
[live]
services = ['underground_rf']
text = [('yowindow.xml', 'L')]
plot = []
```

This section specifies tasks that are to be carried out for every data sample during 'live logging', i.e. every 48 seconds.

`services` is a list of 'services' to upload data to. Each one listed must have a configuration file in `pywws/services/`. See `../api/pywws.toservice` for more detail. pywws will automatically limit the frequency of service uploads according to each service's specification.

`text` and `plot` are lists of text and plot templates to be processed and, optionally, uploaded to your website.

Modifié dans la version 13.05_r1013 : added a `'yowindow.xml'` template. Previously yowindow files were generated by a separate module, invoked by a `yowindow` entry in the `[live]` section. This older syntax still works, but is deprecated.

logged : tâches à accomplir chaque fois que la station enregistre une lecture

```
[logged]
services = ['underground', 'metoffice']
text = []
plot = []
```

Cette section spécifie les tâche devant être réalisées chaque fois qu'une donnée est lue en mode 'temps réel' ou chaque fois qu'une tâche horaire cron est exécutée.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir `../api/pywws.toservice` pour plus de détails.

`text` and `plot` are lists of text and plot templates to be processed and, optionally, uploaded to your website.

cron : tasks to be done at a particular time or date

Nouveau dans la version 14.05.dev1211.

```
[cron prehourly]
format = 59 * * * *
plot = [('tweet.png.xml', 'L')]
services = []
text = []

[cron hourly]
format = 0 * * * *
plot = ['7days.png.xml', '2014.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt', '2014.txt']
services = []

[cron daily 9]
format = 0 9 * * *
plot = ['28days.png.xml']
text = [('forecast.txt', 'T'), 'forecast_9am.txt', 'forecast_week.txt']
services = []

[cron daily 21]
format = 0 21 * * *
text = ['forecast_9am.txt']
services = []
plot = []

[cron weekly]
format = 0 9 * * 6
plot = ['2008.png.xml', '2009.png.xml', '2010.png.xml', '2011.png.xml',
        '2012.png.xml', '2013.png.xml']
text = ['2008.txt', '2009.txt', '2010.txt', '2011.txt', '2012.txt', '2013.txt']
services = []
```

`[cron name]` sections provide a very flexible way to specify tasks to be done at a particular time and/or date. `name` can be anything you like, but each `[cron name]` section must have a unique name.

To use `[cron name]` sections you need to install the "croniter" package. See Pré-requis for more detail.

`format` specifies when the tasks should be done (in local time), in the usual crontab format. (See `man 5 crontab` on any Linux computer.) Processing is not done exactly on the minute, but when the next live or logged data arrives.

hourly : tâches à réaliser à chaque heure

```
[hourly]
services = []
text = [('tweet.txt', 'T'), '24hrs.txt', '6hrs.txt', '7days.txt', 'feed_hourly.xml']
plot = ['7days.png.xml', '24hrs.png.xml', 'rose_12hrs.png.xml']
```

Cette section spécifie les tâches à réaliser à chaque heure en mode 'temps réel' ou exécutées en tant que tâche horaire cron.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir `../api/pywws.toservice` pour plus de détails.

`text` and `plot` are lists of text and plot templates to be processed and, optionally, uploaded to your website.

Modifié dans la version 13.06_r1015 : added the `'T'` flag. Previously Twitter templates were listed separately in `twitter` entries in the `[hourly]` and other sections. The older syntax still works, but is deprecated.

12 hourly : tâches à réaliser à chaque 12 heures

```
[12 hourly]
services = []
text = []
plot = []
```

This section specifies tasks that are to be carried out every 12 hours when 'live logging' or running an hourly cron job. Use it for things that don't change very often, such as monthly graphs. The tasks are done at your day end hour, and 12 hours later.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir `../api/pywws.toservice` pour plus de détails.

`text` and `plot` are lists of text and plot templates to be processed and, optionally, uploaded to your website.

daily : tâches à faire à chaque 24 heures

```
[daily]
services = []
text = ['feed_daily.xml']
plot = ['2008.png.xml', '2009.png.xml', '2010.png.xml', '28days.png.xml']
```

This section specifies tasks that are to be carried out every day when 'live logging' or running an hourly cron job. Use it for things that don't change very often, such as monthly or yearly graphs. The tasks are done at your day end hour.

`services` est une liste de 'services' auxquels envoyer vos données. Chaque service listé doit avoir un fichier de configuration dans le dossier `pywws/services/`. Voir `../api/pywws.toservice` pour plus de détails.

`text` and `plot` are lists of text and plot templates to be processed and, optionally, uploaded to your website.

ftp : configuration du téléversement à un site web

```
[ftp]
local site = False
secure = False
site = ftp.your_isp.co.uk
user = username
password = userpassword
directory = public_html/weather/data/
port = 21
```

Ces entrées fournissent des détails sur votre site web (ou répertoire local) où doivent être transférés les fichier de texte et de graphe traités.

`local site` spécifie si les fichiers doivent être copiés dans un dossier local ou envoyé sur un site distant. Vous pouvez vouloir fixer ce paramètre si votre serveur web fonctionne sur le même ordinateur qui exécute pywws.

`secure` specifies whether to transfer files using SFTP (secure FTP) instead of the more common FTP. Your web site provider should be able to tell you if you can use SFTP. Note that you may need to change the `port` value when you change to or from secure mode.

`site` est l'adresse web de votre site FTP à laquelle transférer vos fichiers.

`user` et `password` sont les informations de branchement pour votre site FTP. Votre hébergeur doit vous avoir fourni ces informations.

`directory` spécifie où les fichiers doivent être stockés sur le site FTP (ou système de fichiers local). Notez que vous devrez peut-être expérimenter avec cela un peu, vous pourriez avoir besoin d'un caractère '/' au début du chemin.

Nouveau dans la version 13.12.dev1120 : `port` spécifie le numéro de port à utiliser. La valeur par défaut est 21 pour FTP, 22 pour SFTP. Votre fournisseur de site web peut vous dire d'utiliser un numéro de port différent.

twitter : configuration pour poster sur Twitter

```
[twitter]
secret = longstringofrandomcharacters
key = evenlongerstringofrandomcharacters
latitude = 51.365
longitude = -0.251
```

`secret` and `key` sont des données d'authentification fournies par Twitter. Pour les définir, exécuter *`pywws.TwitterAuth`*.

`latitude` et `longitude` sont vos données de localisation et sont optionnelles. Si vous les incluez dans vos 'tweets météo' vos utilisateurs pourront voir où est située votre station. Ce qui pourrait aider les utilisateurs à trouver votre station s'ils cherchent par localisation.

underground, metoffice, temperaturnu etc : configuration pour poster sur ces 'services'

```
[underground]
station = IXYZABA5
password = secret
```

Ces sections contiennent l'information nécessaire pour envoyer vos données sur ces services météo, tel que mot de passe et ID de votre station. Le nom de ces entrées de données dépend du service. L'exemple montré affiche les informations pour Weather Underground.

`station` est le PWS ID (Identificateur de station) alloué à la station météo par Weather Underground.

`password` est votre mot de passe pour Weather Underground.

status.ini - format du fichier de status

Ce fichier est crée par pywws et ne doit pas (habituellement) être édité. Les section suivantes sont actuellement utilisées

— fixed : valeurs copiées à partir des "blocs fixes" de la station météo.
— clock : information de synchronisation.
— last update : date et heure de la plus récente tâche complétée.

fixed : valeurs copiées à partir des "blocs fixes" de la station météo

```
[fixed]
fixed block = {...}
```

`fixed block` est toutes les données stockées dans les 256 octets de mémoire de la station. Cela comprend les valeures minimales et maximum, les paramètres de seuil d'alarme, unités de mesure et ainsi de suite.

clock : information de synchronisation

```
[clock]
station = 1360322930.02
sensor = 1360322743.69
```

Ces valeurs enregistrent le temps calculé lorsque l'horloge de la station a enregistrée certaines données et lorsque les capteurs extérieurs ont transmis un nouveau jeu de données. Elles sont utilisées pour tenter d'empêcher le plantage de l'interface USB si l'ordinateur accède à la station météo en même temps que l'un ou l'autre de ces événements, un problème commun à de nombreuses stations compatibles EasyWeather. Les durées sont mesurées toutes les 24 heures pour permettre la dérive de l'horloge

last update : date et heure de la plus récente tâche complétée

```
[last update]
hourly = 2013-05-30 19:04:15
logged = 2013-05-30 19:04:15
daily = 2013-05-30 09:04:15
openweathermap = 2013-05-30 18:59:15
underground = 2013-05-30 18:58:34
metoffice = 2013-05-30 18:59:15
12 hourly = 2013-05-30 09:04:15
```

Ces enregistrement affichent la date et l'heure de la dernière exécution réussie des diverses tâches. Ils sont utilisés pour permettre aux tâches infructueuses (ex. panne de réseau empêchant les téléversements) d'être réessayé après quelques minutes.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Understanding pywws log files

The pywws software uses Python's logging system to report errors, warnings and information that may be useful in diagnosing problems. When you run the software interactively these messages are sent to your terminal window, when running a daemon process or cron job they should be written to a log file.

This document explains some of the pywws logging messages. It's not a complete guide, and the messages you see will depend on your weather station and configuration, but it should help you understand some of the more common ones.

Many pywws commands have a `-v` or `--verbose` option to increase the verbosity of the logging messages. This option can be repeated for even more verbosity, which may be useful when trying to diagnose a particular fault.

Here are some typical logging outputs. The first shows pywws being run interactively :

```
1  jim@firefly ~ $ pywws-livelog -v ~/weather/data/
2  08:50:43:pywws.Logger:pywws version 15.07.1.dev1308
3  08:50:43:pywws.Logger:Python version 2.7.3 (default, Mar 18 2014, 05:13:23) [GCC 4.6.3]
4  08:50:44:pywws.WeatherStation.CUSBDrive:using pywws.device_libusb1
5  08:50:49:pywws.Calib:Using user calibration
6  08:50:49:pywws.Tasks.RegularTasks:Starting asynchronous thread
7  08:51:05:pywws.ToService(wetterarchivde):1 records sent
8  08:51:06:pywws.ToService(underground_rf):1 records sent
9  08:51:06:pywws.ToService(cwop):1 records sent
10 08:51:52:pywws.ToService(wetterarchivde):1 records sent
11 08:51:52:pywws.ToService(underground_rf):1 records sent
12 08:52:40:pywws.ToService(wetterarchivde):1 records sent
13 08:52:40:pywws.ToService(underground_rf):1 records sent
14 08:53:28:pywws.ToService(wetterarchivde):1 records sent
15 08:53:28:pywws.ToService(underground_rf):1 records sent
```

Note that each line begins with a time stamp, in local time. Line 1 is the command used to start pywws. Line 2 shows the pywws version. Line 3 shows the Python version. Line 4 shows which Python USB library is being used. Line 5 shows that a *"user calibration"* routine is being used. Line 6 shows that a separate thread is being started to handle uploads (see *config : diverses configuration système*). The remaining lines show uploads to various weather "services" (see Comment intégrer divers services météorologiques à pywws). You can see from the time stamps that they happen at 48 second intervals.

When running pywws as a daemon process the logging is less verbose :

```
1  2015-07-20 10:46:00:pywws.Logger:pywws version 15.07.1.dev1308
2  2015-07-20 10:50:06:pywws.weather_station:live_data log extended
3  2015-07-20 16:25:59:pywws.weather_station:setting station clock 59.637
4  2015-07-20 16:25:59:pywws.weather_station:station clock drift -0.461377 -0.296364
5  2015-07-20 16:30:24:pywws.ToService(wetterarchivde):<urlopen error [Errno -2] Name or service not kno
6  2015-07-20 16:30:24:pywws.ToService(underground_rf):<urlopen error [Errno -2] Name or service not kno
7  2015-07-20 23:01:16:pywws.ToService(openweathermap):<urlopen error [Errno -2] Name or service not kno
8  2015-07-21 01:14:18:pywws.weather_station:setting sensor clock 42.6678
9  2015-07-21 01:14:18:pywws.weather_station:sensor clock drift -2.03116 -1.98475
10 2015-07-21 09:00:47:pywws.ToTwitter:('Connection aborted.', gaierror(-2, 'Name or service not known')
11 2015-07-21 09:00:55:pywws.Upload:[Errno -2] Name or service not known
12 2015-07-21 09:01:05:pywws.ToService(cwop):[Errno -3] Temporary failure in name resolution
13 2015-07-21 09:06:05:pywws.ToService(underground):<urlopen error [Errno -2] Name or service not known>
14 2015-07-21 09:06:05:pywws.ToService(metoffice):<urlopen error [Errno -2] Name or service not known>
15 2015-07-21 16:30:59:pywws.weather_station:setting station clock 59.4771
16 2015-07-21 16:30:59:pywws.weather_station:station clock drift -0.159373 -0.262116
```

Each line begins with a date and time stamp, in local time. Line 1 shows the pywws version. The remaining lines show normal status messages that are described below.

Clock drift

```
2015-08-31 20:10:45:pywws.weather_station:setting station clock 45.7137
2015-08-31 20:10:45:pywws.weather_station:station clock drift -0.0171086 -0.313699
2015-09-01 01:54:59:pywws.weather_station:setting sensor clock 35.2755
2015-09-01 01:54:59:pywws.weather_station:sensor clock drift -1.12118 -1.37694
```

These lines report how the weather station's internal ("station") and external ("sensor") clocks are drifting with respect to the computer's clock. These measurements are used to avoid accessing the station's USB port at the same time as it is receiving data or logging data, as this is known to cause some station's USB ports to become inaccessible. The two "drift" figures are the current value (only accurate to about 1 second) and the long term average. You should ensure that the `usb activity margin` value in your *weather.ini file* is at least 0.5 seconds greater than the absolute value of the long term drift of each clock. Note that these drift values change with temperature.

The clock drifts are measured at approximately 24 hour intervals. If pywws loses synchronisation with your station it will measure them again. Doing this measurement increases the risk of causing a USB lockup, so if pywws often loses synchronisation you should try and find out why it's happening.

Network problems

Occasionally one or more of the services and web sites you upload data to may become unavailable. This leads to error messages like these :

```
2015-08-03 04:19:49:pywws.ToService(underground_rf):[Errno 104] Connection reset by peer
2015-08-03 04:49:27:pywws.ToService(underground_rf):<urlopen error [Errno -2] Name or service not kno
2015-08-03 05:19:41:pywws.ToService(wetterarchivde):<urlopen error [Errno 101] Network is unreachable
2015-08-03 05:19:46:pywws.ToService(underground_rf):<urlopen error [Errno 101] Network is unreachable
2015-08-03 05:50:52:pywws.ToService(wetterarchivde):<urlopen error [Errno -2] Name or service not kno
2015-08-03 05:50:52:pywws.ToService(underground_rf):<urlopen error [Errno -2] Name or service not kno
```

To avoid swamping the log files duplicate messages are not logged, so you cannot tell how long the network outage lasted from the log files.

Status

```
2015-09-01 21:50:21:pywws.weather_station:status {'unknown': 0, 'invalid_wind_dir': 2048, 'lost_conne
```

The raw weather station data includes some "status" bits. If any of these bits is non-zero when pywws starts, or the status changes value when pywws is running, the status value is logged. The most common problem is `lost_connection` : the weather station console is not receiving data from the outside sensors. Contact is often restored a few minutes later, but if not you may need to reset your weather station console by taking its batteries out. The `invalid_wind_dir` bit indicates that the wind direction sensor value is missing or invalid. The `rain_overflow` bit is set when the rain gauge counter has reached its maximum value and gone back to zero.

Please let me know if you ever get a non-zero value for `unknown`, particularly if you are able to correlate it with some other event. There are 6 bits of data in the status byte whose function is not yet known.

Log extended

```
2015-08-10 08:25:59:pywws.weather_station:live_data log extended
2015-08-10 08:41:59:pywws.weather_station:live_data log extended
2015-08-10 08:57:59:pywws.weather_station:live_data log extended
```

This shows a curiosity in the weather station's internal processing. As the internal and external sensors drift there comes a time when an external reading is expected at the same time as the station is due to log some data. To avoid a clash the station delays logging by one minute. As the external readings are at 48 second intervals this avoids the problem until 16 minutes later (with the normal 5 minute logging interval) when another one minute delay is needed. Eventually the clocks drift apart and normal operation is resumed.

Rain reset

```
2015-08-25 13:30:51:pywws.Process.HourAcc:2015-08-25 12:30:48 rain reset 1048.4 -> 1047.1
2015-08-25 13:35:51:pywws.Process.HourAcc:2015-08-25 12:30:48 rain reset 1048.4 -> 1047.1
2015-08-25 13:40:51:pywws.Process.HourAcc:2015-08-25 12:30:48 rain reset 1048.4 -> 1047.1
```

The raw rainfall data from the outside sensors is the total number of times the "see saw" has tipped since the external sensors were last reset (by a battery change, unless you do it quickly). This number should only ever increase, so the `pywws.Process` module warns of any decrease in the value as it may indicate corrupted data that needs manually correcting. The logging message includes the UTC time stamp of the problem data to help you find it.

Live data missed

```
1 2015-10-30 04:48:19:pywws.ToService(underground_rf):1 records sent
2 2015-10-30 04:49:07:pywws.ToService(underground_rf):1 records sent
3 2015-10-30 04:49:56:pywws.weather_station:live_data missed
4 2015-10-30 04:50:44:pywws.ToService(underground_rf):1 records sent
5 2015-10-30 04:51:31:pywws.ToService(underground_rf):1 records sent
```

Line 3 indicate that pywws failed to capture live data.

There are two possible causes. One is that a new data record is identical to the previous one so pywws doesn't detect a change. This is unlikely to happen if you are receiving wind data properly.

The more likely reason is that processing the previous record took so long that the next one arrived when pywws wasn't ready for it. "Processing" can include uploading to the Internet which is often prone to delays. A solution to this is to set "asynchronous" to True in weather.ini. This uses a separate thread to do the uploading.

You may run with higher verbosity to get more information. The "pause" values should indicate how soon it's ready for the next data.

Note that this is just an occasional missing "live" record though, so if it does not happen often you shouldn't worry too much about it.

"Live log" synchronisation

If you run pywws at a high verbosity you may see messages like the following :

```
1  10:26:05:pywws.Logger:pywws version 15.07.0.dev1307
2  10:26:05:pywws.Logger:Python version 2.7.8 (default, Sep 30 2014, 15:34:38) [GCC]
3  10:26:05:pywws.WeatherStation.CUSBDrive:using pywws.device_libusb1
4  10:26:06:pywws.Calib:Using user calibration
5  10:26:06:pywws.Tasks.RegularTasks:Starting asynchronous thread
6  10:26:06:pywws.weather_station:read period 5
7  10:26:06:pywws.weather_station:delay 2, pause 0.5
8  10:26:07:pywws.weather_station:delay 2, pause 0.5
9  10:26:08:pywws.weather_station:delay 2, pause 0.5
10 10:26:08:pywws.weather_station:delay 2, pause 0.5
```

```
10:26:09:pywws.weather_station:delay 2, pause 0.5
10:26:10:pywws.weather_station:delay 2, pause 0.5
10:26:10:pywws.weather_station:delay 2, pause 0.5
10:26:11:pywws.weather_station:delay 2, pause 0.5
10:26:12:pywws.weather_station:delay 2, pause 0.5
10:26:12:pywws.weather_station:delay 2, pause 0.5
10:26:13:pywws.weather_station:delay 2, pause 0.5
10:26:14:pywws.weather_station:delay 2, pause 0.5
10:26:14:pywws.weather_station:live_data new data
10:26:14:pywws.weather_station:setting sensor clock 38.7398
10:26:14:pywws.weather_station:delay 3, pause 45.4993
10:26:16:pywws.Tasks.RegularTasks:Doing asynchronous tasks
10:27:00:pywws.weather_station:delay 3, pause 0.5
10:27:00:pywws.weather_station:avoid 3.83538614245
10:27:04:pywws.weather_station:live_data new data
10:27:04:pywws.weather_station:delay 3, pause 43.3316
10:27:06:pywws.Tasks.RegularTasks:Doing asynchronous tasks
10:27:48:pywws.weather_station:delay 3, pause 0.5
10:27:48:pywws.weather_station:avoid 3.79589626256
10:27:52:pywws.weather_station:live_data new data
10:27:52:pywws.weather_station:delay 4, pause 0.5
10:27:53:pywws.weather_station:delay 4, pause 0.5
10:27:54:pywws.weather_station:delay 4, pause 0.5
10:27:54:pywws.weather_station:delay 4, pause 0.5
10:27:54:pywws.Tasks.RegularTasks:Doing asynchronous tasks
10:27:55:pywws.weather_station:delay 4, pause 0.5
10:27:56:pywws.weather_station:delay 4, pause 0.5
10:27:56:pywws.weather_station:delay 4, pause 0.5
10:27:57:pywws.weather_station:delay 4, pause 0.5
10:27:58:pywws.weather_station:delay 4, pause 0.5
10:27:58:pywws.weather_station:delay 4, pause 0.5
10:27:59:pywws.weather_station:delay 4, pause 0.5
10:28:00:pywws.weather_station:delay 4, pause 0.5
10:28:00:pywws.weather_station:delay 4, pause 0.5
10:28:01:pywws.weather_station:delay 4, pause 0.5
10:28:02:pywws.weather_station:delay 4, pause 0.5
10:28:02:pywws.weather_station:delay 4, pause 0.5
10:28:03:pywws.weather_station:delay 4, pause 0.5
10:28:04:pywws.weather_station:delay 4, pause 0.5
10:28:04:pywws.weather_station:delay 4, pause 0.5
10:28:05:pywws.weather_station:delay 4, pause 0.5
10:28:06:pywws.weather_station:delay 4, pause 0.5
10:28:06:pywws.weather_station:delay 4, pause 0.5
10:28:07:pywws.weather_station:live_data new ptr: 007320
10:28:07:pywws.weather_station:setting station clock 7.43395
10:28:07:pywws.weather_station:avoid 1.91954708099
10:28:10:pywws.DataLogger:1 catchup records
10:28:10:pywws.Process:Generating summary data
10:28:10:pywws.Process:daily: 2015-08-31 21:00:00
10:28:10:pywws.Process:monthly: 2015-07-31 21:00:00
10:28:10:pywws.Process:monthly: 2015-08-31 21:00:00
10:28:10:pywws.weather_station:delay 0, pause 26.121
10:28:12:pywws.Tasks.RegularTasks:Doing asynchronous tasks
```

Line 6 shows that the weather station has the usual 5 minute logging interval. Lines 7 to 18 show pywws waiting for the station to receive data from the outside sensors. The `delay` value is the number of minutes since the station last logged some data. The `pause` value is how many seconds pywws will wait before fetching data from the station again. Lines 19 & 20 show new data being received and the "sensor" clock being set. Line 21 shows that pywws now

knows when data is next expected, so it can sleep for 43 seconds. Line 22 shows the separate "upload" thread doing its processing while the main thread is sleeping. Line 24 shows pywws avoiding USB activity around the time the station should receive external data. Lines 31 to 53 show pywws waiting for the station to log data. Lines 54 & 55 show the station logging some data and pywws using this to set the "station" clock. The 6 digit number at the end of line 54 is the hexadecimal address where "live" data will now be written to, leaving data at the previous address as a "logged" value. Lines 57 to 61 show pywws fetching logged data from the station and then processing it to produce the various summaries.

Crash with traceback

Sometimes pywws software crashes. When it does, the log file will often contain a traceback like this :

```
1  18:50:57:pywws.LiveLog:error sending control message: Device or resource busy
2  Traceback (most recent call last):
3    File "/usr/local/lib/python2.7/dist-packages/pywws/LiveLog.py", line 80, in LiveLog
4      logged_only=(not tasks.has_live_tasks())):
5    File "/usr/local/lib/python2.7/dist-packages/pywws/LogData.py", line 256, in live_data
6      for data, ptr, logged in self.ws.live_data(logged_only=logged_only):
7    File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 446, in live_data
8      new_ptr = self.current_pos()
9    File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 585, in current_pos
10     self._read_fixed_block(0x0020), self.lo_fix_format['current_pos'])
11   File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 641, in _read_fixed_blo
12     result += self._read_block(mempos)
13   File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 629, in _read_block
14     new_block = self.cusb.read_block(ptr)
15   File "/usr/local/lib/python2.7/dist-packages/pywws/WeatherStation.py", line 265, in read_block
16     if not self.dev.write_data(buf):
17   File "/usr/local/lib/python2.7/dist-packages/pywws/device_pyusb.py", line 152, in write_data
18     usb.REQ_SET_CONFIGURATION, buf, value=0x200, timeout=50)
19 USBError: error sending control message: Device or resource busy
```

Line 1 shows the exception that caused the crash. Lines 3 to 18 show where in the program the problem happened. Usually the last one is of interest, but the other function calls show how we got there. Line 19 shows the full exception. In this case it's a USBError raised by the pyusb library.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Indice Humidité (Humidex)

Auteur de la section : Rodney Persky

Arrière-plan

L'utilisation de votre station météorologique peut être amusant, et les rapports quotidiens des diverses données des sites météo peuvent être très utiles pour vos voisins afin de vérifier les conditions climatiques. Cependant, à un moment donné, vous voudrez peut-être savoir quel sont les effets de la météo sur votre corps, et si il y a une façon de savoir quand il est bon ou non de travailler à l'extérieur.

Nous entrons ici dans un royaume entier de calculs basés sur l'énergie de transfert à travers les murs, et la résistance offerte par ceux-ci. Ce qui peut être une grande aventure de la connaissance, et peut vous faire économiser beaucoup d'argent, et découvrir comment l'énergie se déplace tout autour.

Introduction

L'Humidex est un outil pour déterminer comment un corps humain réagit à la combinaison du vent, de l'humidité et de la température. Fondement de la différence de température entre votre corps et votre peau et est complémentaire à ISO 7243 "Hot Environments - Estimation of the heat stress on working man". Quelques remarques importantes,

- Ces indices sont fondés sur un certain nombre d'hypothèses qui peuvent résulter en une sur- ou sous-estimation de l'état de vos organes internes
- Une station météo personnelle peut ne pas afficher les conditions précises, et donc sur ou sous-estimer l'humidité, le vent ou la température.
- Le choix de vêtements a un effet sur la fatigue et la capacité des organes à rejeter la chaleur, un Indice d'humidité faible ne signifie pas que vous puissiez porter n'importe quoi
- Un entraînement personnel affectera la réaction du corps à une température changeante et l'expérience aidera à savoir quand arrêter de travailler
- La durée des activités qui peuvent être exécutées requiert la connaissance de l'intensité, ce qui ne peut pas être représenté par cet index

Hypothèse

Un certain nombre d'hypothèses ont été faites pour effectuer ce travail qui va affecter directement sa facilité d'utilisation. Ces hypothèses n'ont pas été rendues disponible par Environnement Canada, qui sont les développeurs à l'origine de l'Humidex utilisé dans la fonction PYWWS cadhumidex. Il est suffisamment sûr cependant de dire que la suite aurait été certaines hypothèses

- Type de vêtement, épaisseur
- Surface de la peau exposée à l'air libre
- Exposition au soleil

Cependant, il y a un certain nombre d'hypothèses que pywws doit faire dans le calcul de l'Humidex :

- Les relevés d'humidité, de vent et de température sont exacts

Il y a aussi les hypothèses au sujet du type de corps des individus et de leur 'acclimatation'

- L'entraînement physique d'un individu affectera la réponse de son corps face à des changements de température
- L'expérience aidera à savoir quand arrêter de travailler

Références importantes

Being Prepared for Summer - http://www.ec.gc.ca/meteo-weather/default.asp?lang=En&n=86C0425B-1

Utilisation

La fonction est descriptivement nommé `cadhumidex` et a la température et l'humidité comme paramètres, essentiellement la fonction agit comme une conversion et peut être utilisée d'une manière simple :

```
<ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
```

En l'assemblant, j'ai ajouté des couleurs qui suivent les couleurs d'avertissement de base et les différentes échelles pour produire un graphique décent :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<graph>
  <title>Humidity Index, Bands indicate apparent discomfort in standard on-site working conditions</t
  <size>1820, 1024</size>
  <duration>hours=48</duration>
  <xtics>2</xtics>
  <xformat>%H%M</xformat>
```

```
  <dateformat></dateformat>
  <plot>
    <yrange>29, 55</yrange>
    <y2range>29, 55</y2range>
    <ylabel></ylabel>
    <y2label>Humidex</y2label>
    <source>raw</source>
    <subplot>
      <title>Humidex</title>
      <ycalc>cadhumidex(data['temp_out'],data['hum_out'])</ycalc>
      <colour>4</colour>
      <axes>x1y2</axes>
    </subplot>
    <subplot>
      <title>HI > 54, Heat Stroke Probable</title>
      <ycalc>54</ycalc>
      <axes>x1y2</axes>
      <colour>1</colour>
    </subplot>
    <subplot>
      <title>HI > 45, Dangerous</title>
      <ycalc>45</ycalc>
      <axes>x1y2</axes>
      <colour>8</colour>
    </subplot>
    <subplot>
      <title>HI > 40, Intense</title>
      <ycalc>40</ycalc>
      <axes>x1y2</axes>
      <colour>6</colour>
    </subplot>
    <subplot>
      <title>HI > 35, Evident</title>
      <ycalc>35</ycalc>
      <axes>x1y2</axes>
      <colour>2</colour>
    </subplot>
    <subplot>
      <title>HI > 30, Noticeable</title>
      <ycalc>30</ycalc>
      <axes>x1y2</axes>
      <colour>3</colour>
    </subplot>
  </plot>
</graph>
```

N'exécutez pas la dernière mise à jour ?

Si vous n'utilisez pas ou ne désirez pas utiliser la dernière mise à jour, vous pouvez l'implémenter à l'aide d'un <ycalc> plus long, comme suit :

```
<ycalc>data['temp_out']+0.555*(6.112*10**(7.5*data['temp_out']/(237.7+data['temp_out']))*data['hum_ou
```

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

3.1.6 Programmes et modules Python

Installer et configurer pywws

pywws.TestWeatherStation	Teste la connexion à la station météo.
pywws.SetWeatherStation	Régler certains paramètres de la station météo.
pywws.version	Afficher les informations de version pywws.
pywws.Reprocess	
pywws.TwitterAuth	Autoriser pywws à poster sur votre compte Twitter.
pywws.USBQualityTest	Teste la qualité de la connexion USB avec votre station météo
pywws.EWtoPy	

pywws.TestWeatherStation

Teste la connexion à la station météo.

Le script peut aussi être exécuté avec la commande `pywws-testweatherstation`.

```
usage: python -m pywws.TestWeatherStation [options]
options are:
        --help       display this help
 -c   | --change     display any changes in "fixed block" data
 -d   | --decode     display meaningful values instead of raw data
 -h n | --history n  display the last "n" readings
 -l   | --live       display 'live' data
 -m   | --logged     display 'logged' data
 -u   | --unknown    display unknown fixed block values
 -v   | --verbose    increase amount of reassuring messages
                     (repeat for even more messages e.g. -vvv)
```

This is a simple utility to test communication with the weather station. If this doesn't work, then there's a problem that needs to be sorted out before trying any of the other programs. Likely problems include not properly installing the USB libraries, or a permissions problem. The most unlikely problem is that you forgot to connect the weather station to your computer !

Fonctions

ApplicationLogger(verbose[, logfile])	
main([argv])	
raw_dump(pos, data)	

pywws.TestWeatherStation.**raw_dump**(*pos*, *data*)

pywws.TestWeatherStation.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.SetWeatherStation

Régler certains paramètres de la station météo.

This script can also be run with the `pywws-setweatherstation` command.

```
usage: python -m pywws.SetWeatherStation [options]
options are:
 -h   | --help           display this help
 -c   | --clock          set weather station clock to computer time
                         (unlikely to work)
 -p f | --pressure f     set relative pressure to f hPa
 -r n | --read_period n  set logging interval to n minutes
 -v   | --verbose        increase error message verbosity
 -z   | --zero_memory    clear the weather station logged reading count
```

Fonctions

`ApplicationLogger`(verbose[, logfile])	
`bcd_encode`(value)	
`main`([argv])	

Classes

`datetime`(année, mois, jour[, heure[, minute[, ...)	Les arguments année, mois et jour sont nécessaires.
`timedelta`	Différence entre deux valeurs de datetime.

`pywws.SetWeatherStation.`**`bcd_encode`**(*value*)

`pywws.SetWeatherStation.`**`main`**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.version

Afficher les informations de version pywws.

Le script peut aussi être exécuté avec la commande `pywws-version`.

```
usage: python -m pywws.version [options]
options are:
 -h      or --help      display this help
 -v      or --verbose   show verbose version information
```

Fonctions

main([argv])

pywws.version.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.TwitterAuth

Autorise pywws à publier sur votre compte Twitter :

```
usage: python -m pywws.TwitterAuth [options] data_dir
options are:
 -h or --help       display this help
data_dir is the root directory of the weather data
```

Ce programme autorise *pywws.ToTwitter* à poster sur un compte Twitter. Vous devez créer un compte avant de lancer `TwitterAuth`. Il ouvre une fenêtre de navigateur Web (ou vous donne une URL à copier dans votre navigateur) où vous vous connectez à votre compte Twitter. Si la connexion est réussie, le navigateur affichera un nombre à 7 chiffres, que vous devrez copier dans `TwitterAuth`.

Voir Comment configurer pywws pour poster des messages sur Twitter pour plus de détails sur l'utilisation de Twitter avec pywws.

Fonctions

TwitterAuth(params)
main([argv])

Classes

Twitter

pywws.TwitterAuth.**TwitterAuth**(*params*)

pywws.TwitterAuth.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.USBQualityTest

Teste la qualité de la connexion USB avec votre station météo

```
usage: python -m pywws.USBQualityTest [options]
options are:
```

```
-h | --help           display this help
-v | --verbose        increase amount of reassuring messages
                      (repeat for even more messages e.g. -vvv)
```

La liaison USB à ma station n'est pas fiable à 100%. Les données lues à partir de la station par l'ordinateur sont occasionnellement corrompues, peut-être par une interférence. J'ai essayé de résoudre ce problème en mettant une perle de ferrite autour du câble USB et le déplaçant des sources d'interférences possibles, tels que les disques durs externes. Le tout sans succès jusqu'à présent.

Ce programme teste la connexion USB en lisant de manière continue toute la mémoire de la station météo (sauf les parties qui peuvent être changeantes) à la recherche d'erreurs. Chaque bloc de 32 octets est lu deux fois, et si les deux lectures diffèrent, un message d'avertissement s'affiche. Sont également affichés le nombre de blocs lus, et le nombre d'erreurs rencontrées.

I typically get one or two errors per hour, so the test needs to be run for several hours to produce a useful measurement. Note that other software that accesses the weather station (such as `pywws.Hourly` or `pywws.LiveLog`) must not be run while the test is in progress.

Si vous exécutez ce test et obtenez absolument aucune erreur, s'il vous plaît faites le moi savoir. Il y a quelque chose de spécial dans votre configuration et j'aimerais bien savoir ce que c'est

Fonctions

`ApplicationLogger`(verbose[, logfile])	
`main`([argv])	

`pywws.USBQualityTest.`**`main`**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Obtenir des données et les traiter

`pywws.Hourly`	
`pywws.LiveLog`	
`pywws.livelogdaemon`	

modules "Interne"

`pywws.Tasks`	
`pywws.LogData`	Enregistrer l'historique de station météorologique dans un fichier
`pywws.Process`	
`pywws.calib`	Calibre les données brutes de la station météo
`pywws.Plot`	
`pywws.WindRose`	
`pywws.Template`	
`pywws.Forecast`	
`pywws.ZambrettiCore`	

Suite sur la page suivante

Tableau 3.10 – Suite de la page précédente

pywws.Upload	Téléverser des fichiers sur un serveur web par ftp ou les copier sur un répertoire local
pywws.ToTwitter	Poste un message sur Twitter
pywws.toservice	
pywws.YoWindow	
pywws.WeatherStation	Obtient les données des stations météorologiques WH1080/WH3080 et compatibles.
pywws.device_libusb1	Interface USB de bas niveau de la station météo, à l'aide de python-libusb1.
pywws.device_pyusb1	Interface USB de bas niveau de la station météo, à l'aide de PyUSB v1.0.
pywws.device_pyusb	Interface USB de bas niveau de la station météo, à l'aide de PyUSB v0.4.
pywws.device_ctypes_hidapi	Interface USB de bas niveau de la station météo, utilisant ctype pour accéder hidapi.
pywws.device_cython_hidapi	Interface USB de bas niveau de la station météo, utilisant cython-hidapi.
pywws.DataStore	DataStore.py - enregistre les lectures dans des fichiers facilement accessibles
pywws.TimeZone	
pywws.Localisation	Localisation.py - fournir des traductions des phrases en local
pywws.conversions	
pywws.Logger	Code commun pour l'enregistrement d'informations et d'erreurs.
pywws.constants	Bits de données utilisées dans plusieurs endroits.

pywws.LogData

Enregistre l'historique de la station météo dans un fichier

```
usage: python -m pywws.LogData [options] data_dir
options are:
 -h   | --help     display this help
 -c   | --clear    clear weather station's memory full indicator
 -s n | --sync n   set quality of synchronisation to weather station (0 or 1)
 -v   | --verbose  increase number of informative messages
data_dir is the root directory of the weather data
```

Ce module reçoit les données de la mémoire de la station météorologique et les stocke dans un fichier. Chaque fois qu'il est exécuté, il récupère toutes les données plus récents que les dernières données stockées, de sorte qu'il ne nécessite d'être exécuté qu'environ toutes les heures. Comme la station météo stocke généralement les lectures de deux semaines (en fonction de l'intervalle d'enregistrement), *pywws.LogData* pourrait être exécuté très rarement si vous n'avez pas besoin de données à jour.

Il n'y a aucune information de date ou d'heure dans les données brutes de la station météo, donc *pywws.LogData* crée un horodatage pour chaque lecture. Il utilise l'horloge de l'ordinateur, plutôt que l'horloge de la station météorologique qui ne peut être lu avec précision par l'ordinateur. Un ordinateur en réseau devrait avoir son horloge réglée avec précision par ntp.

La synchronisation avec la station météo est obtenue par l'attente d'un changement dans les données en cours. Il y a deux niveaux de synchronisation, fixés par le fichier de configuration ou une option de ligne de commande. Le niveau 0 est plus rapide, mais n'est exacte qu'à environ douze secondes. Le niveau 1 attend jusqu'à ce que la station météo enregistre un nouvel enregistrement, et obtient un horodatage précis sur quelque secondes. Notez que cela peut prendre un certain temps, si l'intervalle d'enregistrement est supérieur aux cinq minutes recommandées.

API détaillé

Fonctions

ApplicationLogger(verbose[, logfile])

Suite sur la page suivante

Tableau 3.11 – Suite de la page précédente

main([argv])	

Classes

DataLogger(params, status, raw_data)	
datetime(année, mois, jour[, heure[, minute[, ...)	Les arguments année, mois et jour sont nécessaires.
timedelta	Différence entre deux valeurs de datetime.
weather_station([ws_type, status, avoid])	Classe qui représente la station météo pour le programme utilisateur.

class pywws.LogData.**DataLogger**(*params*, *status*, *raw_data*)

check_fixed_block()
catchup(*last_date*, *last_ptr*)
log_data(*sync=None*, *clear=False*)
live_data(*logged_only=False*)

pywws.LogData.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.calib

Calibre les données brutes de la station météo Ce module permet d'ajuster les données brutes de la station météo dans le cadre de l'étape de 'traitement' (voir pywws.Process). Par exemple, si vous avez installé un entonnoir pour doubler votre zone de collecte du pluviomètre, vous pouvez écrire une routine de calibration pour doubler la valeur de pluie.

The default calibration generates the relative atmospheric pressure. Any user calibration you write must also do this.

Écrire votre module de calibration Tout d'abord, décider où vous voulez garder votre module. Comme vos gabarits texte et graphe, il est préférable de le garder séparé du code pywws, de sorte qu'il n'est pas affecté par les mises à jour de pywws. Je suggère la création d'un répertoire modules au même endroit que votre répertoire templates.

You could start by copying one of the example calibration modules, or you can create a plain text file in your modules directory, e.g. calib.py and copy the following text into it :

```
class Calib(object):
    def __init__(self, params, stored_data):
        self.pressure_offset = eval(params.get('config', 'pressure offset'))

    def calib(self, raw):
        result = dict(raw)
        # calculate relative pressure
        result['rel_pressure'] = result['abs_pressure'] + self.pressure_offset
        return result
```

The *Calib* class has two methods. Calib.__init__() is the constructor and is a good place to set any constants you need. It is passed a reference to the raw data storage which can be useful for advanced tasks such as spike removal. Calib.calib() generates a single set of 'calibrated' data from a single set of 'raw' data. There are a few rules to follow when writing this method :

— Assurez-vous d'inclure la ligne `result = dict(raw)`, qui permet de copier toutes les données brutes à votre résultat, au début.
— Ne modifiez pas les données brutes.
— Assurez-vous que vous définissez `result['rel_pressure']`.
— N'oubliez pas de retourner (`return`) le résultat à la fin.

Lorsque vous avez fini d'écrire votre module de calibration vous pouvez demander à pywws de l'utiliser en mettant son emplacement dans votre fichier `weather.ini`. Il va dans les sections `[paths]`, comme le montre l'exemple ci-dessous :

```
[paths]
work = /tmp/weather
templates = /home/jim/weather/templates/
graph_templates = /home/jim/weather/graph_templates/
user_calib = /home/jim/weather/modules/usercalib
```

Notez que la valeur de `user_calib` ne doit pas inclure le `.py` à la fin du nom de fichier.

Classes

Calib(params, stored_data)	Classe qui implémente la calibration par défaut ou la calibration par l'utilisateur.
DefaultCalib(params, stored_data)	Classe de calibration par défaut

class pywws.calib.**DefaultCalib**(*params*, *stored_data*)

Classe de calibration par défaut

This class sets the relative pressure, using a pressure offset originally read from the weather station. This is the bare minimum 'calibration' required.

calib(*raw*)

class pywws.calib.**Calib**(*params*, *stored_data*)

Classe qui implémente la calibration par défaut ou la calibration par l'utilisateur.

D'autres modules pywws utilisent cette méthode pour créer un objet de calibration. Le constructeur crée soit un objet de calibration par défaut ou un objet de calibration utilisateur, en fonction de la valeur `user_calib` dans la section `[paths]` du paramètre `params`. Il adopte alors la méthode de calibration de l'objet :py : meth :*calib* comme sien.

calibrator = None

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.ZambrettiCore

Fonctions

ZambrettiCode(pression, mois, vent, tendance)	Implémentation simple de l'algorithme de prévision de Zambretti.
ZambrettiText(lettre)	
main([argv])	

pywws.ZambrettiCore.**ZambrettiCode**(*pressure*, *month*, *wind*, *trend*, *north=True*, *baro_top=1050.0*, *baro_bottom=950.0*)

Mise en œuvre simple de l'algorithme de prévision Zambretti. Inspiré par algorithme Java de betel-juice.com, telle que convertie en Python par honeysucklecottage.me.uk, et d'autres informations à partir de http://www.meteormetrics.com/zambretti.htm

pywws.ZambrettiCore.**ZambrettiText**(*letter*)

pywws.ZambrettiCore.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.Upload

Téléverse des fichiers sur un serveur web via ftp ou les copies dans un répertoire local

```
usage: python -m pywws.Upload [options] data_dir file [file...]
options are:
 -h or --help    display this help
data_dir is the root directory of the weather data
file is a file to be uploaded
```

Le nom d'usager et les détails du site ftp sont lues à partir du fichier weather.ini dans data_dir.

Introduction Ce module téléverse des fichiers sur (en général) un site Web *via* ftp/sftp ou copie les fichiers dans un répertoire local (par exemple, si vous exécutez pywws sur votre propre serveur web). Les détails de destination de téléversement sont stockés dans le fichier `weather.ini` dans votre répertoire de données. La seule façon de régler ces détails est d'éditer le fichier. Exécuter *`pywws.Upload`* une fois pour définir les valeurs par défaut, que vous pouvez ensuite modifier. Voici ce que vous êtes susceptible de trouver lorsque vous éditez `weather.ini` :

```
[ftp]
secure = False
directory = public_html/weather/data/
local site = False
password = secret
site = ftp.username.your_isp.co.uk
user = username
```

Ce sont, je l'espère, des options assez évidentes. `local site` vous permet de passer de téléversement sur un site distant, à la copie vers un site local. Si vous réglez `local site = True`, vous pouvez supprimer les lignes‘‘secure‘‘, `site`, `user` et `password`.

`directory` est le nom d'un répertoire dans lequel tous les fichiers téléversés seront déposés. Ce qui dépendra de la structure de votre site et du type d'hôte que vous utilisez. Votre fournisseur d'hébergement doit être en mesure de vous dire ce que vous devez utiliser pour `site` et `user`. Vous devriez déjà avoir choisi un `password`.

L'option `secure` vous permet de passer de ftp normal à sftp (ftp sur ssh). Certains fournisseurs d'hébergement l'offre comme mécanisme de chargement plus sûr, alors vous devriez probablement l'utiliser lorsque disponible.

API détaillé

Fonctions

ApplicationLogger(verbose[, logfile])	
main([argv])	

Classes

Upload(params)	

class pywws.Upload.**Upload**(*params*)

connect()
upload_file(*file*)
disconnect()
upload(*files*)

pywws.Upload.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.ToTwitter

Poster un message sur Twitter

```
usage: python -m pywws.ToTwitter [options] data_dir file
options are:
 -h | --help  display this help
data_dir is the root directory of the weather data
file is the text file to be uploaded
```

Ce module poste un bref message sur *Twitter <https ://twitter.com/> _*. Avant de poster sur Twitter, vous devez créer un compte et autoriser pywws en exécutant le programme :py : mod : *TwitterAuth*. Voir :doc : *../Guides/twitter* obtenir les instructions détaillées.

Fonctions

ApplicationLogger(verbose[, logfile])	
main([argv])	

Classes

PythonTwitterHandler(key, secret, latitude, ...)	
ToTwitter(params)	
TweepyHandler(key, secret, latitude, longitude)	
pct	alias de Twitter

class pywws.ToTwitter.**TweepyHandler**(*key*, *secret*, *latitude*, *longitude*)

post(*status*, *media*)

class pywws.ToTwitter.**PythonTwitterHandler**(*key*, *secret*, *latitude*, *longitude*, *timeout*)

post(*status*, *media*)

class pywws.ToTwitter.**ToTwitter**(*params*)

Upload(*tweet*)
UploadFile(*file*)

pywws.ToTwitter.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.WeatherStation

Obtient les données des stations météorologiques WH1080/WH3080 et compatibles.

Dérivé de wwsr.c par Michael Pendec (michael.pendec@gmail.com), wwsrdump.c par Svend Skafte (svend@skafte.net), modifié par Dave Wells, et d'autres sources.

Introduction C'est le module qui parle à l'unité de base de la station météo. Je n'ai pas beaucoup de compréhension de l'USB, donc j'ai copié beaucoup du programme C wwsr de Michael Pendec.

La mémoire de la station météorologique comporte deux parties : un "bloc fixe" de 256 octets et une mémoire tampon circulaire de 65280 octets. Comme chaque lecture météo prend 16 octets, la station peut stocker 4080 lectures, ou 14 jours de lectures intervalle de 5 minutes. (Les stations de type 3080 stockent 20 octets par la lecture, pour un maximum de 3264 lectures) Comme les données sont lues en bloc de 32 octets, mais chaque lecture météorologique est de 16 ou 20 octets, un petit cache est utilisé pour réduire le trafic USB. Le comportement de mise en cache mémoire peut être contourné avec le paramètre `unbuffered` de `get_data` et `get_raw_data`.

Decoding the data is controlled by the static dictionaries `_reading_format`, `lo_fix_format` and `fixed_format`. The keys are names of data items and the values can be an `(offset, type, multiplier)` tuple or another dictionary. So, for example, the _reading_format dictionary entry `'rain' : (13, 'us', 0.3)` means that the rain value is an unsigned short (two bytes), 13 bytes from the start of the block, and should be multiplied by 0.3 to get a useful value.

L'utilisation de dictionnaires imbriqués dans le dictionnaire `fixed_format` permet de décoder des sous-ensembles de données utiles . Par exemple, pour décoder le bloc entier `get_fixed_block` celui-ci est appelée sans paramètres :

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block()
```

Pour obtenir la température extérieure minimale stockées, `get_fixed_block` est appelée avec une séquence de clés :

```
ws = WeatherStation.weather_station()
print ws.get_fixed_block(['min', 'temp_out', 'val'])
```

Souvent, il n'est pas nécessaire de lire et décoder l'ensemble du bloc fixe, comme ses 64 premiers octets contiennent les données les plus utiles : l'intervalle entre les lectures enregistrées, l'adresse du tampon où la lecture en cours est mémorisée, ainsi que la date et l'heure courante. La méthode `get_lo_fix_block` offre un accès facile à ces données.

Pour d'autres exemples de l'utilisation du module WeatherStation, voir le programme TestWeatherStation.

API détaillé

Fonctions

decode_status(status)	

Classes

CUSBDrive()	Interface de bas niveau de la station météo via USB.
DriftingClock(logger, name, status, period, ...)	
USBDevice(idVendor, idProduct)	Low level USB device access via python-libusb1 library.
datetime(année, mois, jour[, heure[, minute[, ...)	Les arguments année, mois et jour sont nécessaires.
weather_station([ws_type, status, avoid])	Classe qui représente la station météo pour le programme utilisateur.

pywws.WeatherStation.**decode_status**(*status*)

class pywws.WeatherStation.**CUSBDrive**

Interface de bas niveau de la station météo via USB.

Vaguement calqué sur une classe C++ obtenue à partir http://site.ambientweatherstore.com/easyweather/ws_1080_2080_protocol
Je n'en sais pas la provenance, mais il semble que cela pourrait provenir du fabricant.

EndMark = 32

ReadCommand = 161

WriteCommand = 160

WriteCommandWord = 162

read_block(*address*)

Lit 32 octets à partir de la station météo.

Si la lecture échoue pour une raison quelconque, None est retourné.

Paramètres **address** (*int*) – address to read from.

Retourne the data from the weather station.

Type retourné list(int)

write_byte(*address*, *data*)

Écrire un seul octet à la station météo.

Paramètres

—**address** (*int*) – address to write to.

—**data** (*int*) – the value to write.

Retourne état succès.

Type retourné bool

class pywws.WeatherStation.**DriftingClock**(*logger*, *name*, *status*, *period*, *margin*)

before(*now*)

avoid()

set_clock(*now*)

invalidate()

class pywws.WeatherStation.**weather_station**(*ws_type='1080'*, *status=None*, *avoid=3.0*)

Classe qui représente la station météo pour le programme utilisateur.

Connecte à la station météo et prépare à lire les données.

min_pause = 0.5

margin = 0.9

live_data(*logged_only=False*)

inc_ptr(*ptr*)

Retourne le pointeur de données suivant du tampon circulaire .

dec_ptr(*ptr*)

Obtient le pointeur de donnée du tampon circulaire précédent.

get_raw_data(*ptr*, *unbuffered=False*)

Obtient les données brutes à partir d'un tampon circulaire.

Si 'unbuffered' est faux, une valeur mise en cache, obtenu précédemment, peut être retourné.

get_data(*ptr*, *unbuffered=False*)

Obtenir les données décodées à partir du tampon circulaire.

Si 'unbuffered' est faux, une valeur mise en cache, obtenu précédemment, peut être retourné.

current_pos()

Obtenir l'emplacement du tampon circulaire lorsque les données actuelles sont en cours d'écriture.

get_raw_fixed_block(*unbuffered=False*)

Obtenir le "bloc fixe" brut des paramètres et des données MIN/MAX.

get_fixed_block(*keys=[]*, *unbuffered=False*)

Obtient le "bloc fixe" décodé des paramètres et des données MIN/MAX.

Un sous-ensemble du bloc complet pouvant être sélectionné par clés.

write_data(*data*)

Écrit un ensemble d'octets unique vers la station météo. Les données doivent être un tableau de (ptr, valeur) paires.

lo_fix_format = {'alarm_1' : (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_lo', 'hum_in_hi', 'hum_out_lo', 'hu

fixed_format = {'alarm_1' : (21, 'bf', ('bit0', 'time', 'wind_dir', 'bit3', 'hum_in_lo', 'hum_in_hi', 'hum_out_lo', 'hum

data_start = 256

reading_len = {'3080' : 20, '1080' : 16}

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.device_libusb1

Interface USB de bas niveau de la station météo, à l'aide de python-libusb1.

Introduction This module handles low level communication with the weather station via the python-libusb1 library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Vérification Run *pywws-testweatherstation* with increased verbosity so it reports which USB device access module is being used :

```
pywws-testweatherstation -vv
11:30:35:pywws.Logger:pywws version 15.01.0
11:30:35:pywws.Logger:Python version 3.3.5 (default, Mar 27 2014, 17:16:46) [GCC]
11:30:35:pywws.WeatherStation.CUSBDrive:using pywws.device_libusb1
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 7f 00 f0 0f 00 50 04
0020 f9 25 74 26 00 00 00 00 00 00 00 15 01 15 11 31 41 23 c8 00 00 00 46 2d 2c 01 64 80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00 00 00 00 00 00
0060 00 00 5a 0a 63 0a 41 01 70 00 dc 01 08 81 dc 01 c5 81 68 01 75 81 95 28 e0 25 24 29 d9 25 fd 02
0080 b9 02 f4 ff fd ff 85 ff 91 ff 6c 09 00 14 10 19 06 29 12 02 01 19 32 11 09 09 05 18 12 03 28 13
00a0 00 13 07 19 18 28 13 01 18 23 21 13 09 24 13 02 13 09 24 13 33 13 09 24 13 02 12 07 28 12 50 13
00c0 09 24 13 02 13 10 14 16 18 12 02 07 19 00 14 02 14 22 39 13 01 04 10 28 15 01 15 03 48 12 03 10
00e0 22 02 13 01 30 21 24 12 07 28 11 59 13 03 06 06 43 12 04 13 00 04 12 04 13 00 04 12 07 31 03 34
```

API

Classes

USBDevice(idVendor, idProduct)	Low level USB device access via python-libusb1 library.

class pywws.device_libusb1.**USBDevice**(*idVendor*, *idProduct*)

Low level USB device access via python-libusb1 library.

Paramètres

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.

—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

read_data(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **size** (*int*) – the number of bytes to read.

Retourne the data received.

Type retourné list(int)

write_data(*buf*)

Envoyer donnée au service

Si l'écriture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **buf** (*list(int)*) – the data to send.

Retourne état succès.

Type retourné bool

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.device_pyusb1

Interface USB de bas niveau de la station météo, à l'aide de PyUSB v1.0.

Introduction This module handles low level communication with the weather station via the PyUSB library (version 1.0). It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Vérification Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used :

```
python -m pywws.TestWeatherStation -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb1
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00 0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64 80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00 00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c 28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05 18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12 02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16 03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57
```

API

Classes

USBDevice(idVendor, idProduct)	Accès de bas niveau au périphérique USB via la bibliothèque PyUSB 1.0.

class pywws.device_pyusb1.**USBDevice**(*idVendor*, *idProduct*)

Accès de bas niveau au périphérique USB via la bibliothèque PyUSB 1.0.

Paramètres

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.

—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

read_data(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **size** (*int*) – the number of bytes to read.

Retourne the data received.

Type retourné list(int)

write_data(*buf*)

Envoyer donnée au service

Si l'écriture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **buf** (*list(int)*) – the data to send.

Retourne état succès.

Type retourné bool

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.device_pyusb

Interface USB de bas niveau de la station météo, à l'aide de PyUSB v0.4.

Introduction This module handles low level communication with the weather station via the PyUSB library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Vérification Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used :

```
python -m pywws.TestWeatherStation -vv
18:28:09:pywws.WeatherStation.CUSBDrive:using pywws.device_pyusb
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 41 11 00 00 00 81 00 00 0f 05 00 e0 51
0020 03 27 ce 27 00 00 00 00 00 00 00 12 02 14 18 27 41 23 c8 00 00 00 46 2d 2c 01 64 80 c8 00 00 00
0040 64 00 64 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 12 00 00 00 00 00 00 00 00
0060 00 00 49 0a 63 12 05 01 7f 00 36 01 60 80 36 01 60 80 bc 00 7b 80 95 28 12 26 6c 28 25 26 c8 01
0080 1d 02 d8 00 de 00 ff 00 ff 00 ff 00 00 11 10 06 01 29 12 02 01 19 32 11 09 09 05 18 12 01 22 13
00a0 14 11 11 04 15 04 11 12 17 05 12 11 09 02 15 26 12 02 11 07 05 11 09 02 15 26 12 02 11 07 05 11
00c0 09 10 09 12 12 02 02 12 38 12 02 07 19 00 11 12 16 03 27 12 02 03 11 00 11 12 16 03 27 11 12 26
00e0 21 32 11 12 26 21 32 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57 12 02 06 19 57
```

API

Classes

USBDevice(idVendor, idProduct)	Accès de bas niveau au périphérique USB via la bibliothèque PyUSB.

class pywws.device_pyusb.**USBDevice**(*idVendor*, *idProduct*)

Accès de bas niveau au périphérique USB via la bibliothèque PyUSB.

Paramètres

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.

—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

read_data(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **size** (*int*) – the number of bytes to read.

Retourne the data received.

Type retourné list(int)

write_data(*buf*)

Envoyer donnée au service

Si l'écriture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **buf** (*list(int)*) – the data to send.

Retourne état succès.

Type retourné bool

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.device_ctypes_hidapi

Interface USB de bas niveau de la station météo, utilisant ctype pour accéder hidapi.

Introduction This module handles low level communication with the weather station via ctypes and the hidapi library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Vérification Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used :

```
python -m pywws.TestWeatherStation -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_ctypes_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00 07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c 81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00 00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08 30 11 03 07 12
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12 02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13 17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38 10 02 22 14 43
```

API

Fonctions

find_library(name)	

Classes

USBDevice(vendor_id, product_id)	Accès de bas niveau au périphérique USB via la bibliothèque hidapi.

class pywws.device_ctypes_hidapi.**USBDevice**(*vendor_id*, *product_id*)

Accès de bas niveau au périphérique USB via la bibliothèque hidapi.

Paramètres

—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.

—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

read_data(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **size** (*int*) – the number of bytes to read.

Retourne the data received.

Type retourné list(int)

write_data(*buf*)

Envoyer donnée au service

Paramètres **buf** (*list(int)*) – the data to send.

Retourne état succès.

Type retourné bool

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.device_cython_hidapi

Interface USB de bas niveau de la station météo, utilisant cython-hidapi.

Introduction This module handles low level communication with the weather station via the cython-hidapi library. It is one of several USB device modules, each of which uses a different USB library interface. See *Installation - USB library* for details.

Vérification Run *pywws.TestWeatherStation* with increased verbosity so it reports which USB device access module is being used :

```
python -m pywws.TestWeatherStation -vv
18:10:27:pywws.WeatherStation.CUSBDrive:using pywws.device_cython_hidapi
0000 55 aa ff ff ff ff ff ff ff ff ff ff ff ff ff 05 20 01 51 11 00 00 00 81 00 00 07 01 00 d0 56
0020 61 1c 61 1c 00 00 00 00 00 00 00 12 02 14 18 09 41 23 c8 00 32 80 47 2d 2c 01 2c 81 5e 01 1e 80
0040 a0 00 c8 80 a0 28 80 25 a0 28 80 25 03 36 00 05 6b 00 00 0a 00 f4 01 18 00 00 00 00 00 00 00 00
0060 00 00 54 1c 63 0a 2f 01 71 00 7a 01 59 80 7a 01 59 80 e4 00 f5 ff 69 54 00 00 fe ff 00 00 b3 01
0080 0c 02 d0 ff d3 ff 5a 24 d2 24 dc 17 00 11 09 06 15 40 10 03 07 22 18 10 08 11 08 30 11 03 07 12
00a0 36 08 07 24 17 17 11 02 28 10 10 09 06 30 14 29 12 02 11 06 57 09 06 30 14 29 12 02 11 06 57 08
00c0 08 31 14 30 12 02 14 18 04 12 02 01 10 12 11 09 13 17 19 11 08 21 16 53 11 09 13 17 19 12 01 18
00e0 07 17 10 02 22 11 06 11 11 06 13 12 11 11 06 13 12 11 11 10 11 38 11 11 10 11 38 10 02 22 14 43
```

API

Classes

USBDevice(idVendor, idProduct)	Accès de bas niveau au périphérique USB via la bibliothèque cython-hidapi.

class pywws.device_cython_hidapi.**USBDevice**(*idVendor*, *idProduct*)

Accès de bas niveau au périphérique USB via la bibliothèque cython-hidapi.

Paramètres
—**idVendor** (*int*) – the USB "vendor ID" number, for example 0x1941.
—**idProduct** (*int*) – the USB "product ID" number, for example 0x8021.

read_data(*size*)

Reçoit des données de l'appareil.

Si la lecture échoue pour une raison quelconque, une exception IOError est levée.

Paramètres **size** (*int*) – the number of bytes to read.

Retourne the data received.

Type retourné list(int)

write_data(*buf*)

Envoyer donnée au service

Paramètres **buf** (*list (int)*) – the data to send.

Retourne état succès.

Type retourné bool

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.DataStore

DataStore.py - enregistre les lectures dans des fichiers facilement accessibles

Introduction Ce module est au cœur de mon logiciel de station météo. Il stocke les données sur disque, mais sans le coût d'un système de base de données à grande échelle. Je l'ai conçu pour fonctionner sur une machine avec peu de mémoire comme mon routeur Asus. Pour minimiser l'utilisation de la mémoire, il ne charge que l'équivalent d'une journée de données à la fois dans la mémoire.

D'un point de vue "utilisateur", les données sont accédées comme un croisement entre une liste et un dictionnaire. Chaque enregistrement de données est indexée par un objet `datetime.datetime` (comportement dictionnaire), mais les enregistrements sont stockés dans l'ordre et peut être consulté sous forme de tranches (comportement liste).

Par exemple, pour accéder aux données horaires pour le jour de Noël 2009, on peut faire ce qui suit

```
from datetime import datetime
from pywws import DataStore
hourly = DataStore.hourly_store('weather_data')
for data in hourly[datetime(2009, 12, 25):datetime(2009, 12, 26)]:
    print data['idx'], data['temp_out']
```

D'autres exemples d'accès aux données :

```
# get value nearest 9:30 on Christmas day 2008
data[data.nearest(datetime(2008, 12, 25, 9, 30))]
# get entire array, equivalent to data[:]
data[datetime.min:datetime.max]
# get last 12 hours worth of data
data[datetime.utcnow() - timedelta(hours=12):]
```

Notez que l'indice `datetime.datetime` est au format UTC. Vous devrez peut-être appliquer un décalage pour convertir en heure locale.

Le module fournit cinq classes pour stocker des données différentes. *`data_store`* prend les données "brutes" de la station météo ; *`calib_store`*, *`hourly_store`*, *`daily_store`* et *`monthly_store`* stockent les données traitées (voir `pywws.Process`). Les trois sont dérivés de la même classe `core_store`, ils ne diffèrent que par les clés et les types de données stockées dans chaque enregistrement.

API détaillé

Fonctions

`Lock`	allocate_lock() -> lock object
`safestrptime`(date_string[, format])	

Classes

`ParamStore`(root_dir, file_name)	
`RawConfigParser`([defaults, dict_type, ...])	
`calib_store`(root_dir)	Stocke les données "calibrées" de la station météo.
`core_store`(root_dir)	

Suite sur la

Tableau 3.28 – Suite de la page précédente

daily_store(root_dir)	Stocke les données sommaires quotidiens de la stations météo.
data_store(root_dir)	Stocke les données brutes de la station météo.
date	date(année, mois, jour) –> date object
datetime(année, mois, jour[, heure[, minute[, ...)	Les arguments année, mois et jour sont nécessaires.
hourly_store(root_dir)	Stocke les données sommaires horaire de la stations météo.
monthly_store(root_dir)	Stocke les données mensuelles sommaire de la stations météo.
params(root_dir)	Les paramètres sont stockés dans le fichier "weather.ini", dans le répertoire spé
status(root_dir)	Le status est stocké dans le fichier "status.ini", dans le répertoire spécifié par ro
timedelta	Différence entre deux valeurs de datetime.

pywws.DataStore.**safestrptime**(*date_string*, *format=None*)

class pywws.DataStore.**ParamStore**(*root_dir*, *file_name*)

flush()

get(*section*, *option*, *default=None*)
Obtient une valeur de paramètre et renvoie une chaîne.
Si la valeur par défaut est spécifiée et la section ou l'option n'est pas définie dans le fichier, ils sont créés et définit par défaut, qui est alors la valeur retournée.

get_datetime(*section*, *option*, *default=None*)

set(*section*, *option*, *value*)
Définit l'option dans la section, à chaîne.

unset(*section*, *option*)
Supprimer l'option de la section.

class pywws.DataStore.**params**(*root_dir*)
Les paramètres sont stockés dans le fichier "weather.ini", dans le répertoire spécifié par root_dir.

class pywws.DataStore.**status**(*root_dir*)
Le status est stocké dans le fichier "status.ini", dans le répertoire spécifié par root_dir.

class pywws.DataStore.**core_store**(*root_dir*)

before(*idx*)
Retourne datetime (horodate) du plus récent enregistrement de données existant dont datetime est <idx.
Peut même ne pas être dans la même année ! Si aucun enregistrement n'existe, retourne None (Aucun).

after(*idx*)
Retourne datetime (horodate) de la plus ancienne donnée existante dont datetime est> = idx.
Peut même ne pas être dans la même année ! Si aucun enregistrement n'existe, retourne None (Aucun).

nearest(*idx*)
Retourne le datetime (horodate) de l'enregistrement dont le datetime est le plus près de idx.

flush()

class pywws.DataStore.**data_store**(*root_dir*)
Stocke les données brutes de la station météo.
key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'wind_ave', 'wind_gust', 'wind_
conv = {'status' : <type 'int'>, 'wind_ave' : <type 'float'>, 'rain' : <type 'float'>, 'hum_in' : <type 'int'>, 'temp_out' : <ty

class pywws.DataStore.**calib_store**(*root_dir*)
Stocke les données "calibrées" de la station météo.
key_list = ['idx', 'delay', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'rel_pressure', 'wind_ave', 'win
conv = {'status' : <type 'int'>, 'wind_ave' : <type 'float'>, 'rain' : <type 'float'>, 'rel_pressure' : <type 'float'>, 'hum_in'

class pywws.DataStore.**hourly_store**(*root_dir*)
Stocke les données sommaires horaire de la stations météo.
key_list = ['idx', 'hum_in', 'temp_in', 'hum_out', 'temp_out', 'abs_pressure', 'rel_pressure', 'pressure_trend', 'wind_
conv = {'pressure_trend' : <type 'float'>, 'wind_ave' : <type 'float'>, 'rain' : <type 'float'>, 'rel_pressure' : <type 'float'>

class pywws.DataStore.**daily_store**(*root_dir*)
Stocke les données sommaires quotidiens de la stations météo.
key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_max', 'hum_out_max_t', 'temp
conv = {'temp_in_min' : <type 'float'>, 'temp_in_max' : <type 'float'>, 'uv_ave' : <type 'float'>, 'temp_out_max' : <type

class pywws.DataStore.**monthly_store**(*root_dir*)
Stocke les données mensuelles sommaire de la stations météo.
key_list = ['idx', 'start', 'hum_out_ave', 'hum_out_min', 'hum_out_min_t', 'hum_out_max', 'hum_out_max_t', 'temp
conv = {'uv_ave' : <type 'float'>, 'illuminance_max_hi_t' : <function safestrptime at 0x7fd198f29c80>, 'uv_max_lo_t' : <

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.Localisation

Localisation.py - fournit des traductions de phrases en langue locale

```
usage: python -m pywws.Localisation [options]
options are:
 -h       or  --help       display this help
 -t code  or  --test code  test use of a language code
```

Introduction Certains modules pywws, comme WindRose.py, peuvent automatiquement utiliser la langue locale pour des choses telles que les directions du vent. Le module Localisation.py, surtout copié à partir d'exemples dans la documentation Python, permet cela.

Traduction de pywws se fait en deux parties - traduisant les phrases comme 'rising very rapidly', en changeant le "terme local" qui contrôle des choses comme le noms des mois et la représentation des nombres (par exemple le nombre '23, 2' au lieu de '23 .2 '). Sur certains ordinateurs, il peut ne pas être possible de définir les paramètres régionaux, mais les chaînes traduites peuvent encore être utilisées.

En utilisant une langue différente Le langage utilisé par pywws est situé dans la section `[config]` du fichier `weather.ini`. Cela peut être un code de langue à deux lettres, comme `fr` (français), ou pouvez spécifier une variante nationale, telle que `fr_CA` (canadien-français). Il pourrait également s'agir d'un jeu de caractères, par exemple `de_DE.UTF-8`.

Le choix de la langue dépend beaucoup du système, donc Localisation.py peut être exécuté comme un programme autonome pour tester les codes linguistiques. Un bon point de départ pourrait être, par exemple, votre variable d'environnement `LANG` :

```
jim@brains:~/Documents/weather/pywws/code$ echo $LANG
en_GB.UTF-8
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t en_GB.UTF-8
Locale changed from (None, None) to ('en_GB', 'UTF8')
Translation set OK
Locale
  decimal point: 23.2
```

```
  date & time: Friday, 14 October (14/10/11 13:02:00)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Dans la plupart des cas, pas plus que d'un code à deux lettres n'est requis :

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t fr
Locale changed from (None, None) to ('fr_FR', 'UTF8')
Translation set OK
Locale
  decimal point: 23,2
  date & time: vendredi, 14 octobre (14/10/2011 13:04:44)
Translations
  'NNW' => 'NNO'
  'rising very rapidly' => 'en hausse très rapide'
  'Rain at times, very unsettled' => 'Quelques précipitations, très perturbé'
jim@brains:~/Documents/weather/pywws/code$
```

Si vous essayez une langue non prise en charge, pywws reprend l'anglais :

```
jim@brains:~/Documents/weather/pywws/code$ python -m pywws.Localisation -t ja
Failed to set locale: ja
No translation file found for: ja
Locale
  decimal point: 23.2
  date & time: Friday, 14 October (10/14/11 13:08:49)
Translations
  'NNW' => 'NNW'
  'rising very rapidly' => 'rising very rapidly'
  'Rain at times, very unsettled' => 'Rain at times, very unsettled'
jim@brains:~/Documents/weather/pywws/code$
```

Une fois que vous avez trouvé un code de langue approprié qui fonctionne, vous pouvez demander à pywws de l'utiliser en éditant votre fichier `weather.ini` :

```
[config]
language = fr
```

Créer une nouvelle traduction S'il n'y a aucun fichier de traduction pour votre langue préférée, alors vous devez en créer un. Voir Comment utiliser pywws dans une autre langue pour obtenir les instructions détaillées.

Fonctions

SetApplicationLanguage(params)	Défini la langue locale et la traduction pour un programme pywws.
SetLocale(lang)	Régle la langue 'locale' utilisée par un programme.
SetTranslation(lang)	Régle la traduction utilisée par les modules (certains) pywws.
main([argv])	

pywws.Localisation.**SetLocale**(*lang*)

Régle la langue 'locale' utilisée par un programme.

Cela affecte toute l'application, en changeant la façon dont les dates, les devises et les chiffres sont représentés. Il

ne devrait pas être appelée à partir d'une routine de bibliothèque qui peut être utilisée dans un autre programme.

Le paramètre `lang` peut être n'importe quelle chaîne reconnue par `locale.setlocale ()`, par exemple `en`, `fr_FR` ou `fr_FR.UTF-8`.

Paramètres **lang** (*string*) – code de langue.

Retourne état succès.

Type retourné bool

pywws.Localisation.**SetTranslation**(*lang*)

Régle la traduction utilisée par les modules (certains) pywws.

Ceci définit l'objet de la traduction `Localisation.translation` à utiliser une langue particulière.

Le paramètre `lang` peut être une chaîne sous la forme `en`, '' fr_FR'' ou `fr_FR.UTF-8`. Tout ce qui suit un caractère `.` est ignoré. Dans le cas d'une chaîne telle que `fr_FR`, la routine va rechercher un fichier de langue `fr_FR` avant de chercher un `fr`.

Paramètres **lang** (*string*) – code de langue.

Retourne état succès.

Type retourné bool

pywws.Localisation.**SetApplicationLanguage**(*params*)

Défini la langue locale et la traduction pour un programme pywws.

Cette fonction permet de lire la langue à partir du fichier de configuration, puis appel les fonctions *SetLocale()* et *SetTranslation()*.

Paramètres **params** (*object*) – un objet *pywws.DataStore.params*.

pywws.Localisation.**main**(*argv=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.Logger

Code commun pour l'enregistrement d'informations et d'erreurs.

Fonctions

ApplicationLogger(verbose[, logfile])

pywws.Logger.**ApplicationLogger**(*verbose*, *logfile=None*)

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

pywws.constants

Bits de données utilisées dans plusieurs endroits.

This module collects together some 'constants' that are used in other pywws modules.

Classes

Twitter	
timedelta	Différence entre deux valeurs de datetime.

class pywws.constants.**Twitter**

consumer_key = '62moSmU9ERTs0LK0g2xHAg'
consumer_secret = 'ygdXpjr0rDagU3dqULPqXF8GFgUOD6zYDapoHAH9ck'

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

3.2 Index et tables

— genindex
— modindex
— search

CHAPITRE 4

Crédits

Je n'aurais pas été en mesure d'obtenir d'informations de la station météorologique sans accès aux sources du programme "wwsr" de Michael Pendec . Je suis également redevable à Dave Wells pour le décodage du "bloc fixe" de données de la station météorologique.

En dernier lieu, un grand vous remercie à tous les utilisateurs de pywws qui ont aidé avec leur questions et suggestions, et particulièrement à ceux qui ont traduit pywws et sa documentation en d'autres langues.

CHAPITRE 5

Termes

pywws - Logiciel Python pour stations météo USB sans-fil.
http://github.com/jim-easterbrook/pywws

Commentaires et questions ? SVP, abonnez-vous à la liste pywws http://groups.google.com/group/pywws et faites-vous entendre.

p

A

B

C

D

E

F

G

H

I

W

Z