WOSPi
– a Weather Observation System for the Raspberry Pi –

© Torkel M. Jodalen
annoyingdesigns.com

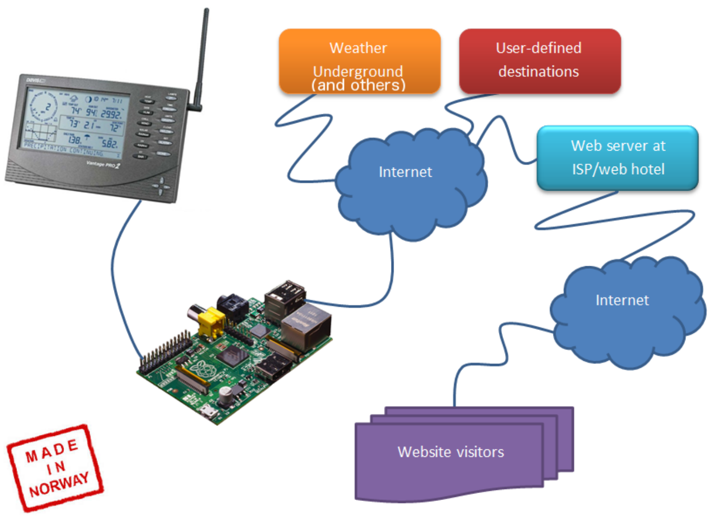
December 25, 2016

Abstract

This document describes a weather observation system for the Raspberry Pi, utilizing the Davis Vantage Pro2 Plus/Vantage Pro2/Vantage Vue weather station as observation platform. The weather station is connected directly to the Raspberry Pi without using a Davis Instruments data logger. All necessary details, part numbers and relevant source code samples are provided. The *WOSPi* software is implemented using the *Python* programming language.

Weather data output is provided in various formats (Weather Underground, weathercloud.net, WindGURU, WindFinder, APRS, XML, plain text file, data plots) and can be customised according to user requirements.

An example of live weather data from a Vantage Pro2 Plus and a Raspberry Pi running the *WOSPi* software can be viewed at `http://meteo.annoyingdesigns.com`.


Weather Underground (and others)
User-defined destinations
Internet
Web server at ISP/web hotel
Internet
Website visitors
MADE IN NORWAY

Contents

Page

thinkbeforeprinting.org

Don't print it unless you *have* to!

1 Project background

The Davis Vantage Pro2 Plus (VP2P) weather station is capable of interfacing to external devices through its rear extension connector. Davis Instruments Corp. provides various *WeatherLink* software products along with their data loggers[1], which are capable of uploadig weather data to a proprietary, subscription-based web server operated by Davis Instruments Corp. The *WeatherLink IP* data logger lacks flexibility and doesn't really leave the user with any real control of the device. The *WeatherLink* software itself is not pretty and it requires a power-consuming personal computer running 24/7 to upload weather data to the web. With the advent of small, embedded Linux-based systems, time has certainly come for better solutions than the dated *WeatherLink* software.

As the serial protocol employed by the VP2P weather station is thoroughly documented by Davis Instruments Corp.[2] and the connector pinout has been documented by unofficial sources, creating the missing link between the weather station and the internet is a rather easy task to accomplish — without using a ridiciously expensive data logger unit. The Raspberry Pi has proven itself as an excellent hardware platform for the weather observation system.

1.1 Disclaimer

The author assumes no responsibility for your use of information contained in this document. Experiment at your own risk. The author does not represent Davis Instruments Corp. or the Raspberry Pi foundation. Likewise, the author does not have any commercial interests in these organisations. All trademarks remain the property of their respective holders, and are used only to identify the products mentioned. Their use in no way indicates any affiliation between the author and the holders of the said trademarks.

Note: the *Raspbian Jessie LITE* image (build 2015-11-21) was used when preparing this document. Other *Raspbian* images may require additional customization.

1.2 Main goal

It is my hope that the *WOSPi* software will be useful and inspire others to experiment with their VP2P and the Raspberry Pi. You can use the *WOSPi* software as a standalone weather observation system or import the desired functions from the `wospi` module for use with your own software projects. This PDF file along with the relevant *Python* documentation strings provide all the information you need to get going and is also the author's own *aide-mémoire*. Have fun!

[1]The *WOSPi* software may be used with a data logger unit, but it will probably be disastrous to connect the RS-232 data logger directly to the GPIO (P1) header of the Raspberry Pi.

[2]`http://www.davisnet.com/support/weather/downloads/software_dllsdk.asp`

2 The first few steps

2.1 A word of warning — console firmware version 3.xx and later

The Davis Instruments line of weather stations, including the Vantage Pro2/Vue series, comes with different firmware versions. It has come to public attention that Davis Instruments Corp. upgraded their consoles at some time in late 2012. New consoles are shipping with firmware versions 3.00, 3.12, 3.15 or later. These versions have a severe drawback compared to earlier versions: they introduce *no* new functionality, they address *no* reliability issues in the v. 1.90 firmware — *but* they prevent the end-user from accessing the serial line at the rear extension connector of the console without purchasing an expensive, original Davis Instruments data logger.

It is a sad situation — Davis Instruments Corp. painting themselves into a corner when the world is on a move towards *open source* hardware and *open data*. Please let Davis Instruments Corp. know what you think of their move to force users into purchasing expensive data loggers in order to retrieve their very own weather data from their very own weather station consoles. Davis Instruments Corp. maintains a Facebook page and they can also be reached by email to `support@davisnet.com` and telephone (800) 678–3669. Firmware version 3.xx is nothing but a move to force end-users into purchasing proprietary hardware solutions. There are no other reasons why Davis Instruments Corp. would introduce the "original logger requirement" for users to access data from the weather station consoles.

Hopefully, Davis Instruments Corp. will listen to customer feedback and take note of the problems they introduced in FW version 3.xx (including versions 3.00, 3.12 and 3.15). Until that happens, users have to deal with the situation themselves — one solution can be found here: `http://meteo.annoyingdesigns.com/DavisSPI.pdf` (yes, a "free my console" modchip is available for US$ 25 or you can easily program one yourself).

The *WOSPi* software has been successfully tested with FW version 3.12 on a model #6312 console, mfg. code A111201xxxx and a chip-modified console running FW v. 3.00, mfg. code AA13020xxxx.

2.2 Weather station and ISS installation

Install the weather station according to the instructions provided. Please pay close attention to setting the station time, UTC offset, automatic DST adjustment, altitude, latitude and longitude correctly. The barometer reading should be verified against a nearby airport with an up-to-date METAR. Rain season should be set to start on the 1st of January. Perform the *Clear All* command on the console after installing the integrated sensor suite to clear collected weather data which may contain erroneous readings after the installation. Refer to the console documentation for a description of the proper procedure.

2.3 A brief quick-start guide

Please note that the *WOSPi* system is *not* ready to run out of the box. A little effort is required to get the system up running — hopefully with the added benefit of learning the basics of the Raspberry Pi.

The suggested procedure for putting *WOSPi* to work is as follows:

- Gather the required parts, refer to section 2.4 for suggestions.
- Configure the Raspberry Pi as described in section 4.
- Carefully wire the VP2P/VP2/Vue console to the Raspberry Pi as detailed in section 5.
- Download and install *WOSPi* according to instructions provided in sections 4.13, 8.4 and 8.5.
- If required, ask for help. Section 11 lists the URL of the *WOSPi* support forum hosted by Google Groups.

This document contains numerous references to the VP2P console. *WOSPi* has been successfully installed and used with VP2 and Vue consoles as well. The console rear-panel connections remain identical but data recording capabilities differ between the various console models and firmware versions. Please refer to section 4.13 for details.

The *WOSPi* software was originally developed for the VP2P, FW versions 1.90, 3.00, 3.12 and 3.15. Using *WOSPi* with firmware versions prior to v. 1.90 may require additional customisation. In particular, the baud rate of older consoles is 9600 bps and the `config.py` file should be modified accordingly.

Users upgrading from an older version of the *WOSPi* software should refer to the revision history (`changelog.txt`). Upgrades may require changes to the `config.py` configuration file as well as other configuration files.

2.4 Shopping for parts

Company names, prices and part numbers referred to here are valid as of November, 2013.

Davis Vantage Pro2 (Plus) weather station The "Plus" model includes solar and UV sensors, but if these are of little or no interest you can just as well start out with a less expensive model. The solar and UV sensors can be added later, if so desired. In the US, AmbientWeather (`http://www.ambientweather.com`) seems like a good place to start shopping for a weather station. With console FW version 3.xx, be prepared to obtain a modchip in order to unlock the console serial line.

ATtiny85 modchip (OPTIONAL) A modchip is required to unlock the console serial line for new VP2/VP2P/Vue consoles shipped with firmware versions $\geq$ 3.00. Available from `http://www.annoyingdesigns.com`. Cost: US$ 25. Figure 1 refers.

Figure 1: The modchip which shouldn't really have been necessary.

FT232RL breakout, 3.3V version (OPTIONAL) While not a *required* part, the FT232RL breakout from SparkFun Electronics (`http://www.sparkfun.com`) implements the USB 2.0 protocol and contains a USB to UART IC. This little breakout board lets you hook up the VP2P console to a PC via a USB port. Also a very useful part if you intend to update the console firmware. *Do not use* the 5V version. Part #BOB-00718. Cost: US$ 15. Figure 2 refers.

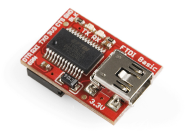


Figure 2: The SparkFun FT232RL breakout, 3.3V version.

PowerBASIC Console Compiler (OPTIONAL) While not a *required* software product, this tool (`http://www.powerbasic.com`) lets you experiment with console communication from your Windows 7/8/10 PC, and you can even write useful software. All without the trouble of writing/setting up a GUI. Cost: US$ 89–169.

SD card The Raspberry Pi requires a SD card to store the operating system, software and user files. An 8 GB, class 4 SD card is sufficient, and you can easily prepare an existing card using the procedure described on the Raspberry Pi website[3]. Download and install the *Raspbian Jessie LITE* image (build 2015-11-21 was used when preparing this document). Cost: US$ 5–10 or so.

Raspberry Pi It does not matter whether you use the ”old” or the ”new” model of the Raspberry Pi. Read up on the different models/revisions and their associated power requirements — the ”low end” models are better suited for projects running on a tight electrical power budget.

The Raspberry Pi is available from Farnell/Newark and numerous other vendors (`http://www.farnell.com` and `http://www.newark.com`). Cost: US$ 35. Figure 3 refers.

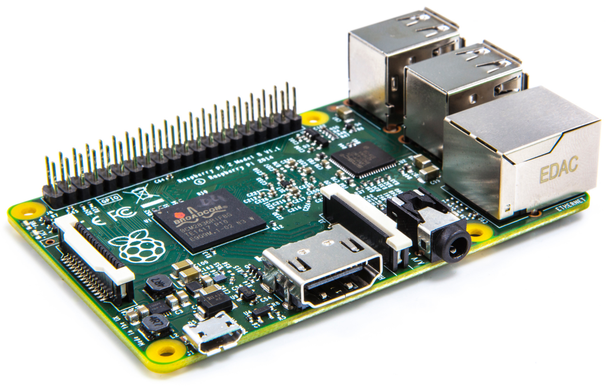


Figure 3: Raspberry Pi 2, model B.

Male/female jumper wires These jumper wires are useful when connecting the VP2P console to the FT232RL breakout. You may as well throw in a pack of female/female jumper wires which will come in handy when hooking up the VP2P console directly to the Raspberry Pi. SparkFun Electronis. Part #PRT-09140. Cost: US$ 3.95. Figure 4 refers.

3.5mm (or 6.35mm) *stereo* phono jacks Using these stereo phono jacks, you'll get a handy connection between the VP2P console and the Raspberry Pi for the serial-line signals (TX, RX and GND). The jacks are available from any electronics store, as is the extension wire that goes ”in between”. *Do not use mono plugs.* Cost: dirt cheap. Figure 5 refers. Note: just about *any* suitable connectors featuring 3 pins can be used.

USB power supply A 5V micro-USB power supply capable of delivering at least 1A is required to run the Raspberry Pi. Cost: not very expensive.

[3]http://www.raspberrypi.org/quick-start-guide

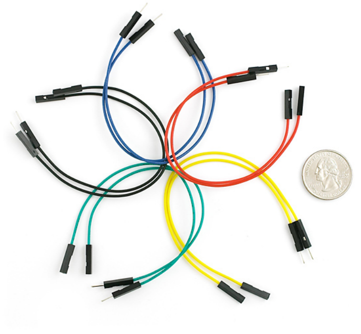


Figure 4: Male/female jumper wires.

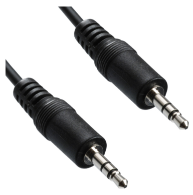


Figure 5: The 3.5mm stereo phono jacks — only the male version shown here.

Pi Holder (OPTIONAL) Sooner or later, you'll want a proper case for the Raspberry Pi. While the Raspberry Pi does not generate a lot of heat, it *does* get warm. Go for a proper aluminum case which allows the Raspberry Pi to run cool by acting as an effective thermal heat sink. In a typcal office environment (room temperature of 20°C), the aluminum case temperature will typically remain at 26–32°C. The *Pi Holder* is expensive but well worth it. Also refer to section 9.20. (`http://www.piholder.com`). Cost: US$ 74.95.

Extension connector, 2mm/20-pos. After a minor modification, this part is extremely useful to "elevate" the expansion connector at the rear of the console. Use a sharp wire cutter to cut it just below the thin plastic spacer. It will then fit perfectly between the console's expansion connector and the receptable (see below). Digi-Key Corp. (`http://www.digi-key.com`). Part #ESQT-110-02-G-D-760-ND. Cost: US$ 6.28. Figure 6 refers.

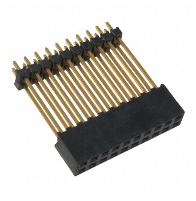


Figure 6: Extension connector, 20mm/20-pos.

USB memory stick/SD card/SD card reader The USB storage device will be used to store weather observations — somewhat equivalent to the ”data logger” functionality. The SD card containing the operating system and the *WOSPi* software should not be used for this purpose.

Connector, receptable, 2mm/20-pos. Farnell/Newark. Part #FCI-89947-720LF, Farnell order code 2112423, Newark part #63K1291. Cost: US$ 2.03. Figure 7 refers.

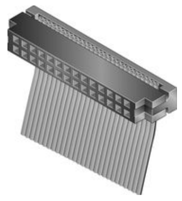


Figure 7: Connector, receptable, 20mm/20-pos.

Ribbon cable, 20 cond. multi, 5’ Digi-Key Corp. Part #AE20B-5-ND. Cost: US$ 8.65. Figure 8 refers.

Figure 8: Ribbon cable.

WiFi USB dongle (OPTIONAL) Pick a variant which is known to work with the Raspberry Pi. Adafruit.com and SparkFun.com are both known to sell add-ons which work well with the Raspberry Pi. Please note that a WiFi USB dongle may significantly increase the power requirements of the Raspberry Pi. Cost: US$ 10-15. Figure 9 refers.

Figure 9: WiFi USB dongle.

3 Putting things together

3.1 Connecting the VP2P console to a PC — without a data logger

If you have no intention to experiment using a traditional personal computer, simply skip this section and move on to section 4. Section 3.2 may still contain relevant information, though.

Connecting the VP2P console to a PC is by no means *required*, but it will allow for experimenting with console communications using familiar terminal emulation software such as Realterm[4]. This setup may also be used to update the console firmware. Also, it will serve as an important first step in programatically decoding the contents of the LOOP and LOOP2 data packages returned by the console. A few *PowerBASIC* source code examples are included here, but this could just as well be implemented using any other language supporting serial communications and decoding of binary data.

Using the SparkFun FT232RL breakout (3.3V version — *do not use* the 5V version), the weather station can easily be interfaced to a PC running the Windows 7/8/10 operating system. Drivers are also available for MacOS X and Linux.

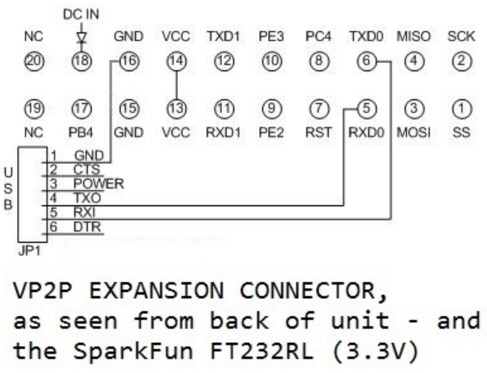


Figure 10: Wiring diagram between the VP2P console and the SparkFun FT232RL breakout. The same wiring may also be utilized to update the console firmware. Based on schematics by *DeKay*.

Wiring between the VP2P console and the SparkFun USB-to-serial converter requires three wires:

- `TXD0` from the VP2P console goes to `RX` on the FT232RL
- `RXD0` from the VP2P console goes to `TX` on the FT232RL
- `GND` from the VP2P console goes to `GND` on the FT232RL

Using a terminal emulation program such as Realterm, you can now start communicating with the VP2P console. Please refer to the *Serial Communication Reference Manual*, available from the Davis Instruments Corp. website for details.

[4] `http://realterm.sourceforge.net`

3.2 Extending the console's rear expansion connector

Accessing the rear expansion port requires a steady hand. For not-so-steady hands, adding an extension connector as shown below will make the exercise a bit easier.

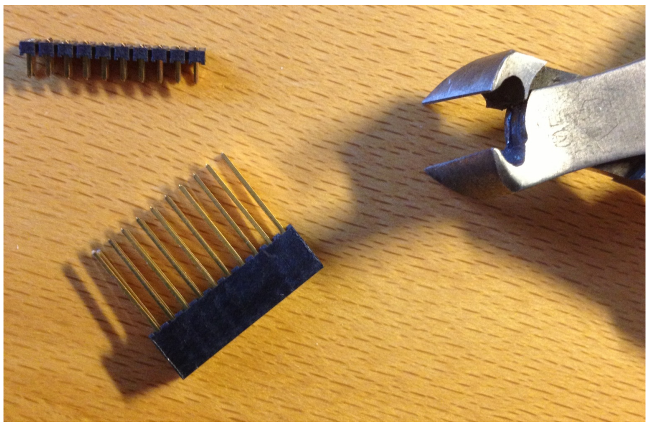


Figure 11: Preparing the extension connector.

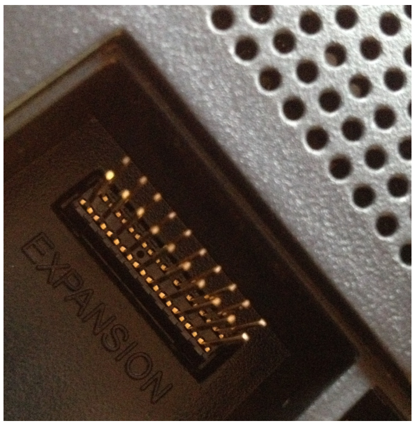


Figure 12: "Elevating" the rear expansion port makes it easier to work with. Don't let the pins get in close contact with each other.

3.3 PC-side programming — for those interested

NOTE: jump directly to section 4 if all you care about is getting the *WOSPi* software up running.

PC-side programming can be employed to gain a better knowledge and understanding of the communication protocol. The *PowerBASIC* Console Compiler (PB/CC) contains all required functions to facilitate further testing and/or write a complete weather data retrieval/presentation system for the Win32 platform. Feel free to skip this section, as it is by no means required for the Raspberry Pi implementation.

The following communication parameters apply:

- 19,200 bps
- 8 data bits
- No parity
- 1 stop bit
- No flow control (XON/XOFF)

Allow the console some time to wake up and respond to commands. Experimenting will reveal the required delays in program execution after issuing a command to the console. Note that some commands require slightly longer time to execute on the console.

The `LOOP` and/or `LPS` commands are essential when it comes to retrieving weather station data. Both the LOOP and the LOOP2 packages each consist of 100 bytes, made up of binary-encoded data provided by the console.

PowerBASIC functions `CVWRD` (double-byte/word values) and `CVBYT` (byte values) are essential for decoding the contents of the LOOP and LOOP2 packages.

Listing 1: Complete *PowerBASIC* source code listing: sending the `TEST` command to the console and reading the answer.

```
$COMPORT = "COM3" ' refer to the Windows device manager for COM port number
%DELAY = 100      ' communications delay - 100ms may not be the optimum value
%WKUPDELAY = 1200 ' wakeup delay - according to Davis documentation
DIM F AS LONG     ' file number for COM port access
DIM Q AS LONG     ' number of bytes waiting in RX queue
DIM S AS STRING   ' text from RX queue

F = FREEFILE
COMM SET #F, BAUD = 19200
COMM SET #F, BYTE = 8
COMM SET #F, PARITY = 0
COMM SET #F, STOP = 1
COMM SET #F, XINPFLOW = 0
COMM SET #F, XOUTFLOW = 0

COMM OPEN $COMPORT AS #F CHR=ANSI
COMM RESET #F, FLOW

COMM SEND #F, CHR$(10) + CHR$(10)    ' wake up the console with 2 x LF
SLEEP(%DELAY)
' check for console response to wakeup call
' if no response, SLEEP(%WKUPDELAY) then perform up to two more
' wakeup calls - ref. Davis documentation
COMM SEND #F, "TEST" + CHR$(10)      ' send the TEST command + LF
SLEEP(%DELAY)
Q = COMM(#F, RXQUE)
COMM RECV #F, Q, S
PRINT S
CLOSE #F
CON.WAITKEY$
```

Listing 2: Incomplete *PowerBASIC* source code example reading the *barometric trend* field from the LOOP package using the `CVBYT` function.

```
S$ = COMM_SEND(#F, "LOOP 1" + CHR$(10) )
IF LEN(S$) <> 100 THEN
   PRINT "ERROR: ONLY " + STR$(LEN(S$)) + " BYTES RECEIVED FROM THE CONSOLE."
   BEEP
ELSE
   PRINT "RECEIVED 100 BYTES FROM THE CONSOLE. CRC NOT VERIFIED."

   LOCAL Q AS LONG
   LOCAL BAROTREND AS BYTE
   Q = INSTR(S$, "LOO")
   BAROTREND = CVBYT(S$, Q+3) ' barometric trend byte is at offset 3
   PRINT "BAROMETER TREND : ";
   SELECT CASE BAROTREND
      CASE  -60:  PRINT "falling rapidly"
      CASE  -20:  PRINT "falling slowly"
      CASE    0:  PRINT "steady"
      CASE   20:  PRINT "rising slowly"
      CASE   60:  PRINT "rising rapidly"
      CASE  196:  PRINT "falling rapidly"
      CASE  236:  PRINT "falling slowly"
      CASE ELSE:  PRINT "Rev. A or 3hr BARO DATA not available."
   END SELECT
END IF
```

Listing 3: Incomplete *PowerBASIC* source code example reading the *barometric pressure* (inHg units) field from the LOOP package using the `CVWRD` function. A conversion to hPa/mb is also shown.

```
' Assuming that S$ still contains the LOOP package from the previous
' code listing and that Q contains the INSTR result from the same listing.

LOCAL BARO AS DOUBLE
BARO = CVWRD(S$, Q+7) / 1000  ' barometric pressure word is at offset 7
PRINT "BAROMETER : " + STR$(BARO) + " inHg"
PRINT "BAROMETER : " + FORMAT$(33.8639 * BARO, 5) + " hPa (mb)"
```

You may want to carry out a few experiments using the below commands which should yield easy-to-understand results. Note that the HILOWS, LOOP and LOOP2 packets are binary-encoded — as such, on-screen output will look like a string of random characters/garbage prefixed by the "LOO" characters. All commands should be issued in UPPERCASE LETTERS only. You may want to turn on *local echo* in the terminal emulation program. The console first needs a few LF characters to wake up, and all commands must be terminated with a single LF *or* CR — not both.

- `LAMPS 1` — LCD background illumination ON.
- `LAMPS 0` — LCD background illumination OFF.
- `LOOP 1` — retrieve one LOOP packet.
- `LPS 1 1` — retrieve one LOOP packet, requires console FW version $\geq$ 1.90.
- `LPS 2 1` — retrieve one LOOP2 packet, requires console FW version $\geq$ 1.90.
- `VER` — retrieve console firmware date.
- `NVER` — retrieve console firmware version.
- `BARDATA` — retrieve barometric calibration data.
- `STRMON` and `STRMOFF` — refer to documentation and various web resources.

The *Serial Communication Reference Manual* as provided by Davis Instruments Corp. provides further details on the protocol employed by the console. Sooner or later you will encounter unreasonable readings, communication delays and CRC errors. Make sure that your code is able to handle these exceptions in a reasonable way. It is highly recommended to implement the CCITT-16 CRC routine to verify that data returned by the console is indeed correctly transmitted/received.

When sending data to the console (such as when setting the console date/time based on the correct time obtained from an NTP server), a valid CRC checksum is required by the console. Section 6.6 describes a *Python*-based implementation of the CCITT-16 CRC algorithm.

4 Configuring the Raspberry Pi

The *Raspbian Jessie LITE* image can be downloaded from the *Raspberry Pi* web site[5]. If a GUI framework is desired, download the "full" version instead.

Copying the *Raspbian Jessie LITE* image onto a SD card is covered in the *Quick start guide* on the Raspberry Pi website (keyword for Windows users: *Win32DiskImager*[6]). A working SD card with the *Raspbian Jessie LITE* Linux distribution is assumed from here on.

The default password for the *pi* user is *raspberry.* The default password for the *root* user is not specified. Section 4.6 describes how to set/change the *root* password.

Proceed by plugging the Raspberry Pi into your router, allowing it to acquire an IP address via DHCP — or configure the WLAN adapter as detailed in section 4.4. If so desired, a static IP address may be configured at a later stage.

4.1 The *raspi-config* tool

When first logging in to the Raspberry Pi, you'll be greeted by the *raspi-config* tool. You can move around using the arrow keys and the `TAB` key. Activate your choice by hitting the `ENTER` key. The *raspi-config* tool can be started at any time by issuing the `sudo raspi-config` command.The following options should be configured:

- Internationalisation options
  - Keyboard layout — set according to your desired keyboard layout.
  - Locale information — preferably an option containing "UTF-8".
  - Your timezone.
- Advanced options
  - Enable the SSH server[7].
  - Disable the *shell/login and kernel messages over serial line* option.
- Do *not* start the desktop GUI (if at all installed) on boot.
- Require the user to log in.
- Expand the file system to fill the entire SD card.

A reboot will be required to apply the above settings.

Expanding the file system to fill the entire SD card seems to be required when using the *Raspbian Jessie* images, as these come with a rather small root partition. Refer to section 9.14.4 for details on the impact of expanding the file system.

[5]`http://www.raspberrypi.org`

[6]`http://sourceforge.net/projects/win32diskimager/`

[7]You may want to download the free *PuTTY* SSH client to remotely log in to the Raspberry Pi: `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html` contains what you need

4.2 Manually specifying keyboard layout and locale info

NOTE: this step can now be performed using the *raspi-config* tool. There is really no reason why you should change these settings manually.

You may need the `console-data` package — in that case, follow the general install procedure detailed in section 4.7.

a) `sudo nano /etc/default/keyboard` — set keyboard layout according to your preferences.

b) `sudo setupcon` — select Norwegian/UTF-8, US/UTF-8 or any other setting — with a strong preference for the UTF-8 variants.

c) `sudo dpkg-reconfigure tzdata`

d) `sudo dpkg-reconfigure locales`

At this point it is suitable to restart the Raspberry Pi, using the `shutdown -r now` command.

4.3 Assigning a static IP address to the Raspberry Pi — cabled ethernet

While this step is not *required*, it makes life a bit easier when you want to log in to your Raspberry Pi via a network connection.

a) `sudo nano /etc/dhcpcd.conf` — edit as below:

```
hostname
clientid
persistent
option ntp_servers

interface eth0
static ip_address=10.0.0.100/24
static routers=10.0.0.1
static domain_name_servers=10.0.0.1
```

...assuming the that you want to assign the fixed IP address 10.0.0.100 to your Raspberry Pi and that the router/gateway can be found at IP address 10.0.0.1. Also, in the above example a netmask of 255.255.255.0 is specified (/24). Other settings may apply in your network.

b) `sudo nano /etc/resolv.conf` — edit as below:

```
domain MyDomain
search MyDomain
nameserver 10.0.0.1
```

...provided that your router/gateway at IP address 10.0.0.1 also works as a name server. This may or may not apply in your network. Replace *MyDomain* with your local domain name/search list for hostname lookups.

Restart the Raspberry Pi to apply the new settings. If you apply settings which render the Raspberry Pi without a network connection, you can always log in directly from a locally attached console[8] and change the settings back to their original values.

NOTE: also consider using your internet router's DHCP server for repeatedly assigning the same "dynamic" IP address to the Raspberry Pi, based on the unit's MAC address. In this case, simply leave the Raspberry Pi to obtain a DHCP-issued IP address — leaving it to the router to always assign the same IP address to the Raspberry Pi.

[8] Here referring to a keyboard and a monitor.

4.4 Assigning a static IP address to the Raspberry Pi — WLAN

NOTE: an easier configuration option is now provided by the *WiFi Config* tool found in the GUI (type *startx* to launch the GUI, if installed) Also refer to the NOTE in section 4.3.

For WLAN setup, follow the step-by-step instructions as provided in section 4.3, but edit the `/etc/dhcpcd.conf` file as follows:

```
hostname
clientid
persistent
option ntp_servers

interface wlan0
static ip_address=10.0.0.100/24
static routers=10.0.0.1
static domain_name_servers=10.0.0.1
```

The `/etc/wpa_supplicant/wpa_supplicant.conf` file should be updated with your network-specific settings (replace `your_SSID` and `your_PSK` as appropriate). This example will work with with old-style WPA-secured WLAN networks:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
 ssid="your_SSID"
 psk="your_PSK"
 proto=RSN
 key_mgmt=WPA-PSK
 pairwise=CCMP TKIP
 group=CCMP TKIP
}
```

The following example will work with WPA2/Personal-secured WLAN networks (AES):

```
network={
 ssid="your_SSID"
 psk="your_PSK"
 proto=RSN
 key_mgmt=WPA-PSK
 pairwise=CCMP
 auth_alg=OPEN
}
```

There are numerous online guides detailing various WLAN setup options for the Raspberry Pi. The above listings have proved to work, even though they may produce three `ioctl` error messages when bringing up the interface. As it seems, these error messages can be safely ignored.

To activate the WLAN interface after applying the above configuration settings, simply run `sudo ifup wlan0` or restart the Raspberry Pi using `sudo shutdown -r now`.

4.5 Setting a proper host name

Typically, you'll want to set a nice host name (such as "wx") for the Raspberry Pi.

a) `sudo nano /etc/hostname` — this file should only contain the actual host name.

b) `sudo nano /etc/hosts` — add *127.0.0.1 MyNewHostName* where *MyNewHostname* is the host name you specified in the `/etc/hostname` file above. Other "shorthand" host names may also be specified here.

c) `sudo shutdown -r now` — to restart the system.

4.6 Adding/modifying user accounts

You can change the *root* password by logging in as the *pi* user and then running the `sudo passwd root` command. Your *WOSPi* system, when fully configured, will typically have three active user accounts:

root for system setup and system maintenance requiring superuser privileges.

wospi the user account associated with the *WOSPi* software. The *wospi* account requires access to the system's *ttyAMA0* device (UART/serial port).

wx the user account which will automatically be logged in on *tty6* for output of updated weather data to a locally attached display (if present).

The `passwd` command can be used to change password(s). Passwords for the *root* and *wospi* user accounts should be changed at regular intervals.

4.6.1 Adding the *wospi* and *wx* user accounts

The *wospi* user account will be the account running the *WOSPi* software. Invoke the following commands:

a) `sudo adduser wospi` — adding new user *wospi*. Note the new user ID (UID) for use with section 4.10.

b) `sudo adduser wx` — adding new user *wx* [9].

c) `sudo usermod -a -G dialout wospi` — adding the *wospi* user to the *dialout* group.

d) `sudo visudo` — add the *wospi* user to the `/etc/sudoers` file, allowing the *wospi* user to run `sudo`. Simply change the old entry for the *pi* user, replacing *pi* with *wospi* before saving the file.

NOTE: for group affiliation changes to take effect, the *wospi* user will have to log out and then log in again. Use the `groups` command to check which group(s) a user belongs to.

[9] Not really required, but if omitted — also skip the instructions provided in sections 4.14 and 4.15.

4.6.2 Deleting the *pi* user account

The *pi* user account serves no purpose in the *WOSPi* system and should be deleted. Make sure that you are logged in as the *wospi* user and invoke the following command:

a) `sudo deluser pi --remove-home`

4.7 Updating and installing software packages

Once the Raspberry Pi has proper internet access, additional software packages can be installed. The LITE version of the Raspbian distribution comes with a "bare bones" minimum of software packages preinstalled.

First update the locally stored APT[10] package index by running `sudo apt-get update`. Also upgrade existing software packages by running `sudo apt-get upgrade`.

Then install the following packages by running `sudo apt-get install` followed by the package name(s) (some of these packages may already have been installed):

- `mingetty`
- `screen`
- `htop`
- `python-serial`
- `python-dateutil`
- `coreutils`
- `emacs`[11]
- `ftp`
- `gnuplot`
- `minicom`[12]
- `nmap` — not *required*, but useful to have around.
- `zip` — not *required*, but useful to have around.
- `ssmtp` — not *required*, but useful for sending email (section 9.12 refers).
- `mutt` — not *required*, but useful for sending email and may be used for backup purposes as detailed in section 9.12.

The above software packages will require some 430 MB of available space on the SD card (adding to the temporary storage space required during installation).

[10]Advanced Packaging Tool — simplifies the process of managing software on Unix-like computer systems by automating the retrieval, configuration and installation of software packages, either from binary files or by compiling source code.

[11]Some users will probably prefer other editors. Pick your choice. *Nano* is referred to throughout this document only because it is available in the *Raspbian Jessie LITE* image by default. Please use whatever editor which suits your preferences. *Windows' Notepad* is not an option, though.

[12]Inside *minicom*, your best friend will be the `Ctrl-A` keyboard command.

4.8 Setting up a *Python*-based FTP server

A *Python*-based FTP server is not *required*, but it will allow for easy and convenient file transfers between the Raspberry Pi and your desktop/laptop PC — highly recommended for backup purposes. The FTP server will only be running when explicitly invoked by the user.

The latest version of *pyftpdlib* can be found at `https://github.com/giampaolo/pyftpdlib/archive/master.zip`. As of December 2014, the current version number is 1.4.0. Here assuming *Python* version 2.7.x installed on the Raspberry Pi (default on the SD card image available from the Raspberry Pi website). Issue the following commands:

a) `cd ~`

b) `wget https://github.com/giampaolo/pyftpdlib/archive/master.zip`

c) `unzip master.zip`

d) `sudo mv pyftpdlib-master/pyftpdlib /usr/lib/python2.7/pyftpdlib`

e) `rm -Rf pyftpdlib-master`

f) `rm master.zip`

g) Finally log in as the *wospi* user and create a *Python* program in the `wospi` home folder containing the FTP server program. A typical implementation is shown below. NOTE: this file is included in the *WOSPi* distribution archive.

Listing 4: A *Python*-based FTP server using the *pyftpdlib* module.

```python
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer

authorizer = DummyAuthorizer()
authorizer.add_user("Name", "Password", "Path", perm="elradfmw")
handler = FTPHandler
handler.authorizer = authorizer
server = FTPServer(("", 21), handler)
server.serve_forever()
```

Replace *Name*, *Password* and *Path* with suitable values as appropriate (*wospi*, *topSecret* and */home/wospi* could be typical values). Please DO NOT use *topSecret* as password — it is shown here as an example only. Also, for security reasons, the password should not be the same as the login password for the *wospi* user account.

Whenever FTP access to the system is required, simply start the FTP server by running `sudo python ftpServer.py`. Hit `Ctrl-C` to quit the FTP server program when it is no longer needed.

4.9 Editing the system login message

The system login message is defined in the `/etc/motd` file. Running `sudo nano /etc/motd` as *root* allows you to edit the file. A sample file is shown below.

```
-                         WOSPi                         -
- a Weather Observation System for the Raspberry Pi     -
-      by Torkel M. Jodalen  --  tmj@bitwrap.no         -
-                                                       -
-  http://meteo.annoyingdesigns.com                     -
-                                                       -
-  Running on hostname 'wx' with static IP 10.0.0.100.  -
```

4.10 Auto-mounting USB-attached storage devices

In order to reduce the number of write operations to the SD card containing the operating system and the *WOSPi* software, a USB memory stick or other kind of external storage device should be utilized for periodically storing weather data. A USB-connected hard drive could also be used, but would significantly increase the power consumption of the Raspberry Pi unless powered from an external power source. The standard *WOSPi* system will store all weather parameters every ten minutes.

a) `sudo mkdir /media/sd` — this will be the access path of the device.

b) `cd /etc`

c) `sudo cp fstab fstab.backup` — creates a backup copy of the `fstab` file.

d) `ls -l /dev/disk/by-uuid`

e) Insert the USB storage device.

f) `ls -l /dev/disk/by-uuid` — take note of the UUID belonging to the new device.

g) `sudo nano /etc/fstab` — add the following line:

```
UUID=xxx /media/sd vfat defaults,auto,noatime,uid=1000 0 0
```

...where *xxx* will represent the actual UUID from (f) above. The *uid* (user ID) parameter should match the numeric user ID for the *wospi* user (shown as *1000*, as an example), as listed in the `/etc/passwd` file (to list the contents of the file, run the `cat /etc/passwd` command) and/or noted in section 4.6.1.

h) `sudo mount -a` — attempt auto-mount of the newly defined device entry.

i) `ls -l /media/sd` — should list any existing files and folders on the "new" device.

4.11 Setting file system properties

The SD card containing the operating system and the *WOSPi* software should be optimized for running 24/7 by reducing the number of write operations to the card. The following procedure is suggested:

a) `sudo dphys-swapfile swapoff` — to disable the swap file.

b) Modify the `fstab` file with the *noatime* parameter for the *ext4* file system — preventing updates to the "access time" whenever files are accessed: `sudo nano /etc/fstab`. Also, make sure that `/var/tmp`, `/tmp` and `/var/log` are modified as follows:

```
tmpfs           /var/tmp   tmpfs  nodev,nosuid,noatime,size=50M                0   0
tmpfs           /tmp       tmpfs  defaults,noatime,nosuid                      0   0
tmpfs           /var/log   tmpfs  defaults,noatime,nosuid                      0   0
proc            /proc      proc   defaults                                     0   0
/dev/mmcblk0p1  /boot      vfat   defaults                                     0   2
/dev/mmcblk0p2  /          ext4   defaults,discard                             0   1
UUID=xxx        /media/sd  vfat   defaults,auto,noatime,uid=1000               0   0
```

The *UUID=xxx* refers to the UUID of the USB device as described in section 4.10. Replace *xxx* with an appropriate UUID reference. The *uid* parameter should match the user ID of the *wospi* user — refer to sections 4.6.1 and 4.10 for details.

c) `sudo /etc/init.d/rsyslog stop` — stop the *rsyslog* service.

d) `sudo rm -Rf /tmp/*` — remove everything from the */tmp* folder.

e) `sudo rm -Rf /var/log/*` — remove everything from the */var/log* folder.

f) `sudo rm /var/tmp/*` — remove everything from the */var/tmp* folder.

g) `sudo mount -a` — auto-mount attached devices.

h) `sudo /etc/init.d/rsyslog start` — restart the *rsyslog* service.

4.12 Disabling kernel serial line output and serial line login

NOTE: this step can now be performed using the *raspi-config* tool.

By default, the system will output kernel status messages to the serial line (at *ttyAMA0*). Also, a *getty* login prompt is normally available on the serial line. As the serial line will be used for communications with the VP2P console, no other input/output should take place here.

a) `sudo systemctl stop serial-getty@ttyAMA0.service`

b) `sudo systemctl disable serial-getty@ttyAMA0.service`

c) To save some time, also refer to section 4.15 now. If you proceed with section 4.15, *do not restart* the Raspberry Pi until section 4.14 is also completed.

d) ...otherwise restart the Raspberry Pi for the new settings to take effect.

4.13 Installing the *WOSPi* software

Download and unpack the *WOSPi* distribution archive as follows:

a) `wget http://www.annoyingdesigns.com/wospi/wospi.zip`

b) `unzip wospi.zip`

c) Refer to the `readme.txt` file for installation instructions.

d) Refer to the `config.py` file for configuration options.

NOTE: the *WOSPi* software comes preconfigured for use with VP2P consoles, expecting solar radiation and UV radiation readouts.

Refer to the `config.py` configuration file for details on how to configure *WOSPi* for use with consoles which do not supply these radiation values. The `plot24.input` file should also be updated.

4.14 The *wxview.sh* shell script

If the *wx* user was added as described in section 4.6.1, the default shell for this user should be altered to provide weather data output only.

The `wxview.sh` shell script will be the default shell for the *wx* user, providing weather data output on *tty6*, accessible by pressing `Alt-F6` on a locally attached keyboard. The `wxview.sh` file is included in the *wospi* distribution archive. If you move it to the home folder of the *wx* user, make sure to update file ownership and group association accordingly (using `chown` and `chgrp`).

a) Log in as the *wx* user and store the this script as */home/wx/wxview.sh* by running `nano /home/wx/wxview.sh` :

```
# Display weather data output on tty6 (ref. inittab)
# User 'wx' will have the default shell set to this script
# sed removes HTML formatting and special &xxx; characters

#!/bin/bash

clear
echo 'Weather data will appear here shortly. Please be patient.'

while :
do
    if [ -f /var/tmp/wxdata.txt ]
    then
        tail -f -n 50 /var/tmp/wxdata.txt |
        sed -e 's/&amp;/and/g;s/<[^>]*>//g;s/&[^;]*;/ /g'
    fi
    sleep 1
done
```

NOTE: for layout reasons, a line break was inserted above, causing ”`sed -e (...)`” to appear on a line by itself. The actual `wxview.sh` file should be without this line break. The apostrophe character above (`'`) is ASCII character no. 39. Be careful to type in the script *exactly* as shown above.

b) The `wxview.sh` shell script should be made *executable* by running the `chmod 700 wxview.sh` command.

Entering the contents of the `wxview.sh` file by hand is not recommended. Consider ”copy and paste” instead. Most errors are likely to result in the `init: id (x) respawning too fast` error message, effectively denying login for the *wx* user. Again, the file is also included in the *WOSPi* distribution archive — but it needs to be moved to the home folder of the *wx* user.

4.15 Adding auto-login and changing the default shell for the *wx* user

The *wx* user account will provide continous weather data output on *tty6*. Make sure that the `wxview.sh` file has been made executable, as detailed in section 4.14.

a) `sudo -i`

b) `mkdir -pv /etc/systemd/system/getty@tty6.service.d`

c) `nano /etc/systemd/system/getty@tty6.service.d/autologin.conf` — edit as follows:

```
[Service]
ExecStart=
ExecStart=-/sbin/mingetty --autologin=wx tty6
```

d) `chsh -s /home/wx/wxview.sh wx` — change the defaut shell for the *wx* user.

e) `shutdown -r now` for a system restart.

4.16 Setting up RSA keypairs for "passwordless" SCP

The SCP command will normally ask for a password for the remote system. Your system can be set up to allow for "passwordless" SCP file transfers. Replace *myUserName*, *myServer* and *myHomePath* with appropriate values for your system/web hotel.

a) Log in as the *wospi* user.

b) `ssh-keygen` — Save the keys in the default location. Do *not* use a passphrase.

c) `cd ~/.ssh`

d) `scp id_rsa.pub myUserName@myServer:myHomePath/.ssh/authorized_keys2`[13]. You *should* be asked for your password to the remote system[14].

e) `ssh-agent sh -c "ssh-add < /dev/null && bash"`

f) `echo test > t.txt` — for testing.

g) `scp t.txt myUserName@myServer:/myHomePath` — for testing. You should *not* be asked for a password. The file should be copied to the specified server/path.

h) `rm t.txt` — clean up.

i) Hit `Ctrl-D`.

[13]Some systems (incl. MacOS X) use `authorized_keys` instead of `authorized_keys2`.

[14]This step will *overwrite* any existing `authorized_keys2` file on the remote system. *Appending* the file may be a better idea, especially if other units or users also require "passwordless" SCP/SSH access to the same user account at the remote system.

4.17 Enabling auto-run of the *wospi.pyc* program

In order for the *WOSPi* software to restart after a power failure or an intentional system reboot, the `/etc/rc.local` file should be altered slightly:

a) At this stage the *wospi* user has to be a member of the *dialout* group, as described in section 4.6.1.

b) `sudo nano /etc/rc.local` — add the following line *before* the line containing `exit 0` :

```
su - wospi -c
'screen -m -d -S wxscreen /usr/bin/python /home/wospi/wospi.pyc'
```

NOTE: for layout reasons, a line break was inserted above. The actual `rc.local` file should be without this line break. The apostrophe character above (`'`) is ASCII character no. 39.

4.18 Verifying *screen* output

As output from the *WOSPi* software is "encapsulated" by the *screen* utility, it cannot be examined without re-attaching to the *screen* session. Log in as the *wospi* user, then issue the following command(s):

a) `screen -R wxscreen` — to bring up the last output from the *wospi.pyc* program. Depending on conditions, it may take up to 30 seconds until any output appears. Don't worry if you notice an error message or two — they occur periodically.

b) To exit *screen* and leave the *wospi.pyc* program running, hit `Ctrl-A, d` to *detach* from the present *screen* session. The *wospi.pyc* program will keep running in the background.

c) To *exit* the *wospi.pyc* program, hit `Ctrl-A, k` instead.

The *screen* utility supports a number of useful command-line options and arguments. To list active *screen* sessions, simply issue the `screen -ls` command.

4.19 Logging *screen* output to a file

A log file containing output captured by the `screen` utility can be created by editing the `.screenrc` file located in the home folder of the *wospi* user. If the file does not already exist, simply create it using a text editor. This can be a useful tool for debugging your *WOSPi* installation — but will normally not be required. To conserve memory usage, consider deleting the accumulated log file from time to time.

```
logfile /var/log/screen-%S-%n.log
deflog on
```

5 Wiring the Raspberry Pi to the console

Byrne's Law: In any electrical circuit, appliances and wiring will burn out to protect the fuses. – Robert Byrne

5.1 General information

Wiring between the VP2P console and the Raspberry Pi requires three wires (the TXD[15], RXD[16] and GND[17] signals). Anything *transmitted* from the VP2P console should be *received* by the Raspberry Pi. Likewise, anything *transmitted* from the Raspberry Pi should be *received* by the VP2P console — explaining the "crossing" of the TXD and RXD lines. Both units share the same signal ground reference (GND).

5.2 Now you're warned (again)

You are proceeding entirely at your own risk. Rest assured, though — no VP2P console, no BeagleBone and no Raspberry Pi was hurt during the development of the *WOSPi* software.

- Be extremely careful not to short any wires/connections.
- Always connect/disconnect the wires without an external DC power supply connected and without batteries inserted in the VP2P/VP2/Vue console unit.
- Power down the Raspberry Pi and disconnect its power supply before making the connections to the VP2P/VP2/Vue console.
- Double-check all connections before powering up the devices again.

When attaching the 2mm/20-pos. extension connector to the ribbon cable, make sure to use a vise in order to apply an even pressure to the connector. Using a tool which *does not* apply an even pressure is *guaranteed* to break the connector.

Using standard 3.5mm (or equally standard 6.35mm) *stereo* phono connectors as "interface connectors" between the VP2P console and the Raspberry Pi adds a bit of flexibility — you can easily mount the VP2P console and the Raspberry Pi at slightly different locations. Temporarily disconnecting the units is performed in a matter of seconds. The *stereo* phono connector has three connection points, perfectly suited for the TXD, RXD and GND signals. A cable length of up to 3m has been tested without signs of any problems.

[15] Transmit Data
[16] Receive Data
[17] Ground

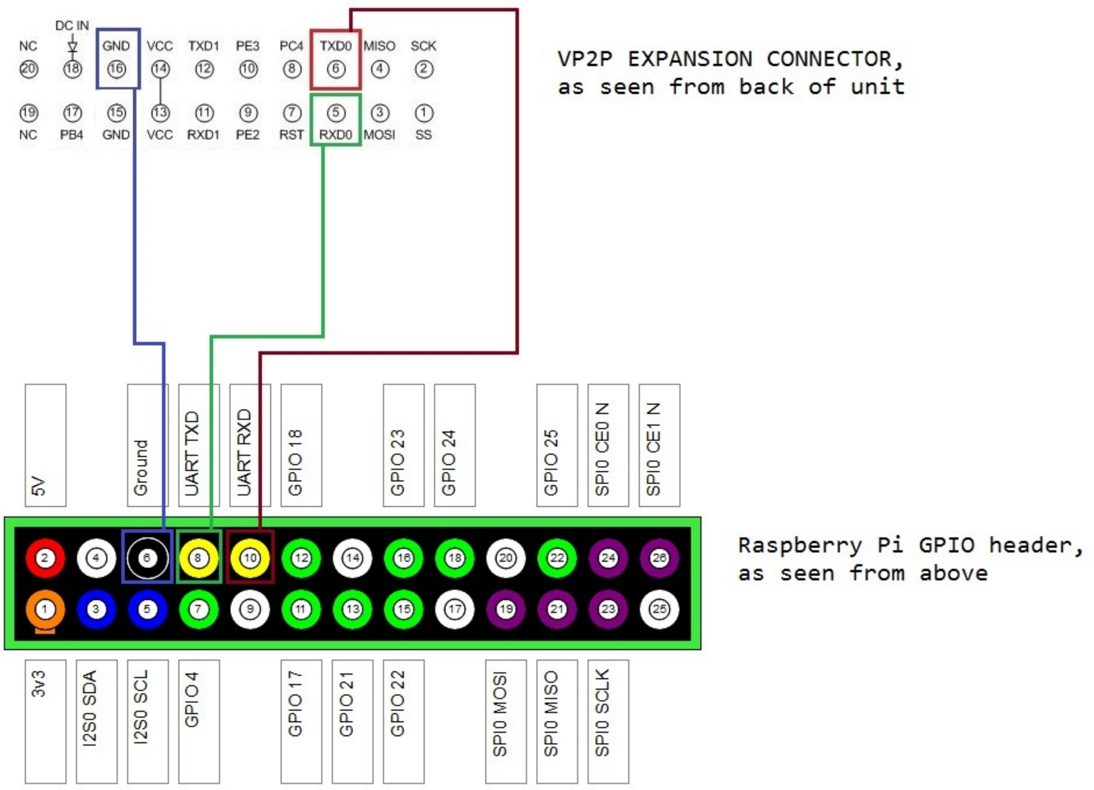


Figure 13: Wiring diagram between the VP2P console and the Raspberry Pi GPIO (P1) header. Based on schematics by *DeKay*. The GPIO headers differ somewhat between the A/B (26 pins, as shown above) and the A+/B+/2B/3 (40 pins, as shown in figure 14) versions of the Raspberry Pi. NOTE: the new Raspberry Pi 3 requires a slightly different connection thanks to the new, on-board Bluetooth modem.

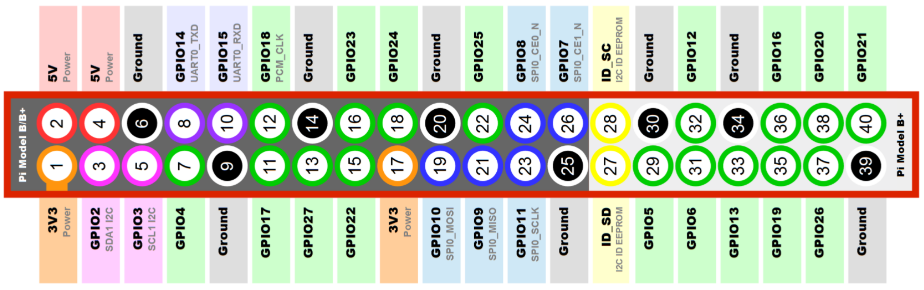


Figure 14: The "new" 40-pin GPIO connector. Signals `GROUND`, `UART0_TXD` and `UART0_RXD` remain easy to locate.

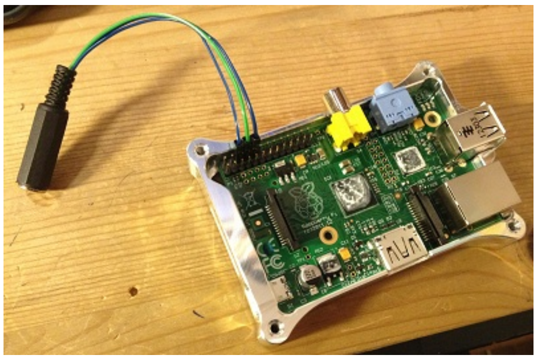


Figure 15: The Raspberry Pi with the *female* 3.5mm stereo phono connector attached. Other connector types may indeed be more suiteable but the phono connectors are fairly common and easy to get hold of.

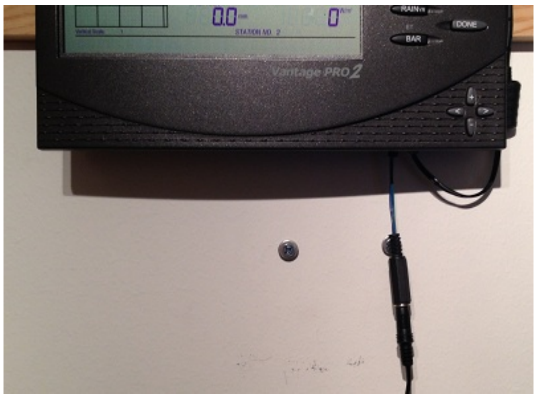


Figure 16: The VP2P console with the *female* 3.5mm stereo phono connector attached, connected to the Raspberry Pi via an off-the-shelves 3.5mm stereo phono jack cable.

6 Sample Python code

Sample *Python* v. 2.7.x code is provided here, to facilitate your experiments with the Raspberry Pi, Python and the VP2P/VP2/Vue console. *WOSPi*'s data extraction is based on the techniques shown here.

6.1 Using the *serial* module

First disable all other use of the `/dev/ttyAMA0` device, as described in section 4.12.

Listing 5: *Python* source code example, showing basic use of the *serial* module. Please refer to the module documentation for full details, including how to *read* data from the serial line.

```
import serial

WXPORT      = '/dev/ttyAMA0'
WXBAUDRATE = 19200
WXBYTESIZE = serial.EIGHTBITS
WXPARITY    = serial.PARITY_NONE
WXSTOPBITS = serial.STOPBITS_ONE
WXXONOFF    = False
WXTIMEOUT   = 3

w = serial.Serial(WXPORT, WXBAUDRATE, WXBYTESIZE, WXPARITY, WXSTOPBITS, WXTIMEOUT, WXXONOFF)

# perform the console wakeup calls here, ref. Davis documentation

w.write('TEST\n')
```

6.2 Using the *struct* module

The *struct* module contains the `unpack_from` function which can be used to decode the binary-encoded data contained in the LOOP and LOOP2 data packages. The `pack` function can be used to encode data going *to* the VP2P console.

Please refer to the *Serial Communication Reference Manual*, available from the Davis Instruments Corp. website for details on the data formats used by the VP2P console.

Listing 6: *Python* source code example, showing basic use of the *struct* module.

```
import struct

# Read LOOP and/or LOOP2 packet from the VP2P console, store
# the returned result in string s - and go ahead decoding the
# binary-encoded values using the struct.unpack_from function
# as shown below.

wxDict['BAROTREND'] = struct.unpack_from('B', s, 1)[0]
wxDict['BAROMETER_INHG'] = round(struct.unpack_from('H', s, 5)[0] / 1000.0, 2)
```

6.3 *WOSPi's wxDict* and *wxMinMax* dictionaries

The *wxDict* dictionary will contain the essential values returned by the VP2P console using the LOOP and LOOP2[18] packet formats, provided the data packet CRC checksum is valid. Some additional key-value-pairs are also included. The *wxDict* dictionary is updated twice a minute.

The *wxMinMax* dictionary will contain the essential min/max values returned by the VP2P console using the HILOWS packets. It is updated at the interval specified in the *WOSPi* configuration file.

Listing 7: *Python* source code example, showing basic use of the *wxDict* dictionary.

```
import wospi

wospi.insertTestData() # OR perform actual console readout

print wospi.wxDict     # prints the entire dictionary
print wospi.wxDict['BAROMETER_HPA']
print wospi.wxMinMax   # prints the entire dictionary
```

NOTE: local copies of these dictionaries are also available to the user-defined functions found in the `config.py` file. For further details, please refer to the source code comments.

6.4 The *getRawData* function

The *getRawData* function allows users to extract values from received LOOP 1, LOOP 2 and HILOWS data packets. The function will return the requested value from the last data packet received with a valid checksum. LOOP packets are retrieved twice a minute and HILOWS packets are retrieved at 10-minute intervals[19]. The *Serial Communication Reference Manual* contains descriptions of the data packet format, including offset and data size for each variable.

Listing 8: *Python* source code example, showing use of the *getRawData* function.

```
import wospi

wospi.wx = wospi.openWxComm()
wospi.readWxData()   # read LOOP/LOOP2 packet(s), as determined by the LPS setting in config.py
wospi.hiLows()       # read HILOWS packet

# Note: the wxDict and wxMinMax dictionaries are now populated.
#       "print wxDict" and/or "print wxMinMax" will display the corresponding dictionary contents.

print wospi.getRawData(1,  9, 'H') # LOOP  packet, offset  9, WORD value = inside temperature
print wospi.getRawData(2, 43, 'B') # LOOP2 packet, offset 43, BYTE value = UV index
print wospi.getRawData(1, 18, 'B') # LOOP  packet, offset 18, BYTE value = Extra temperature sensor
# Note: extra temperature is offset by 90 degrees F - ref. Davis documentation
print wospi.getRawData(1, 62, 'B') # LOOP  packet, offset 62, BYTE value = Soil moisture #1 in centibars
```

The *getRawData* function is intended for users who need to extract the less common values from the data packets. Most users will probably be happy to retrieve the values from the provided dictionaries instead.

[18]LOOP2 packets are retrieved only if supported and `LPS=True` is set in the configuration file.
[19]Unless the interval has been changed in the configuration file.

6.5 MSB or LSB first?

The VP2P console expects the MSB[20] first in the CRC[21] checksum. For other values, the LSB[22] is expected as the first bit. Again, refer to the *Serial Communication Reference Manual* for details. The *struct* module can be used with both MSB-first and LSB-first bit-orders.

6.6 The CCITT-16 CRC algorithm — *Python* implementation

The CCITT-16 CRC function is shown here, in case you ever need to implement it for use with your own software. It is available in the *wospi* module as `CRC`. You don't *have to* understand it — it works, it works reasonably well, and it has been around and used extensively in the telecom industry for quite some time.

Listing 9: Python source code, showing the implementation of the CCITT-16 CRC function.

```
def CRC(inputData):
    """CCITT-16 CRC implementation, function should return 0"""
    crcTab = ( \
    0x0000,  0x1021,  0x2042,  0x3063,  0x4084,  0x50a5,  0x60c6,  0x70e7,  \
    0x8108,  0x9129,  0xa14a,  0xb16b,  0xc18c,  0xd1ad,  0xe1ce,  0xf1ef,  \
    0x1231,  0x210,   0x3273,  0x2252,  0x52b5,  0x4294,  0x72f7,  0x62d6,  \
    0x9339,  0x8318,  0xb37b,  0xa35a,  0xd3bd,  0xc39c,  0xf3ff,  0xe3de,  \
    0x2462,  0x3443,  0x420,   0x1401,  0x64e6,  0x74c7,  0x44a4,  0x5485,  \
    0xa56a,  0xb54b,  0x8528,  0x9509,  0xe5ee,  0xf5cf,  0xc5ac,  0xd58d,  \
    0x3653,  0x2672,  0x1611,  0x630,   0x76d7,  0x66f6,  0x5695,  0x46b4,  \
    0xb75b,  0xa77a,  0x9719,  0x8738,  0xf7df,  0xe7fe,  0xd79d,  0xc7bc,  \
    0x48c4,  0x58e5,  0x6886,  0x78a7,  0x840,   0x1861,  0x2802,  0x3823,  \
    0xc9cc,  0xd9ed,  0xe98e,  0xf9af,  0x8948,  0x9969,  0xa90a,  0xb92b,  \
    0x5af5,  0x4ad4,  0x7ab7,  0x6a96,  0x1a71,  0xa50,   0x3a33,  0x2a12,  \
    0xdbfd,  0xcbdc,  0xfbbf,  0xeb9e,  0x9b79,  0x8b58,  0xbb3b,  0xab1a,  \
    0x6ca6,  0x7c87,  0x4ce4,  0x5cc5,  0x2c22,  0x3c03,  0xc60,   0x1c41,  \
    0xedae,  0xfd8f,  0xcdec,  0xddcd,  0xad2a,  0xbd0b,  0x8d68,  0x9d49,  \
    0x7e97,  0x6eb6,  0x5ed5,  0x4ef4,  0x3e13,  0x2e32,  0x1e51,  0xe70,   \
    0xff9f,  0xefbe,  0xdfdd,  0xcffc,  0xbf1b,  0xaf3a,  0x9f59,  0x8f78,  \
    0x9188,  0x81a9,  0xb1ca,  0xa1eb,  0xd10c,  0xc12d,  0xf14e,  0xe16f,  \
    0x1080,  0xa1,    0x30c2,  0x20e3,  0x5004,  0x4025,  0x7046,  0x6067,  \
    0x83b9,  0x9398,  0xa3fb,  0xb3da,  0xc33d,  0xd31c,  0xe37f,  0xf35e,  \
    0x2b1,   0x1290,  0x22f3,  0x32d2,  0x4235,  0x5214,  0x6277,  0x7256,  \
    0xb5ea,  0xa5cb,  0x95a8,  0x8589,  0xf56e,  0xe54f,  0xd52c,  0xc50d,  \
    0x34e2,  0x24c3,  0x14a0,  0x481,   0x7466,  0x6447,  0x5424,  0x4405,  \
    0xa7db,  0xb7fa,  0x8799,  0x97b8,  0xe75f,  0xf77e,  0xc71d,  0xd73c,  \
    0x26d3,  0x36f2,  0x691,   0x16b0,  0x6657,  0x7676,  0x4615,  0x5634,  \
    0xd94c,  0xc96d,  0xf90e,  0xe92f,  0x99c8,  0x89e9,  0xb98a,  0xa9ab,  \
    0x5844,  0x4865,  0x7806,  0x6827,  0x18c0,  0x8e1,   0x3882,  0x28a3,  \
    0xcb7d,  0xdb5c,  0xeb3f,  0xfb1e,  0x8bf9,  0x9bd8,  0xabbb,  0xbb9a,  \
    0x4a75,  0x5a54,  0x6a37,  0x7a16,  0xaf1,   0x1ad0,  0x2ab3,  0x3a92,  \
    0xfd2e,  0xed0f,  0xdd6c,  0xcd4d,  0xbdaa,  0xad8b,  0x9de8,  0x8dc9,  \
    0x7c26,  0x6c07,  0x5c64,  0x4c45,  0x3ca2,  0x2c83,  0x1ce0,  0xcc1,   \
    0xef1f,  0xff3e,  0xcf5d,  0xdf7c,  0xaf9b,  0xbfba,  0x8fd9,  0x9ff8,  \
    0x6e17,  0x7e36,  0x4e55,  0x5e74,  0x2e93,  0x3eb2,  0xed1,   0x1ef0)

    crcAcc = 0

    for byte in (ord(part) for part in inputData):
        ushort = (crcAcc << 8) & 0xff00
        crcAcc = ((ushort) ^ crcTab[((crcAcc >> 8) ^ (0xff & byte))])

    return crcAcc
```

[20] Most Significant Bit

[21] Cyclic Redundancy Check

[22] Least Significant Bit

7 Example output

Sample output from the *WOSPi* system is provided on the next few pages. Output is highly customisable — you have full access to all data retrieved from the VP2P console and you can process the data in any way you like. Observations are stored in plain text files (including CSV[23] and XML[24] files), giving you access to most data collected by the ISS. Only the default VP2P sensors are supported by *WOSPi* but support for additional sensors can easily be added by using the *getRawData* function as described in section 6.4.

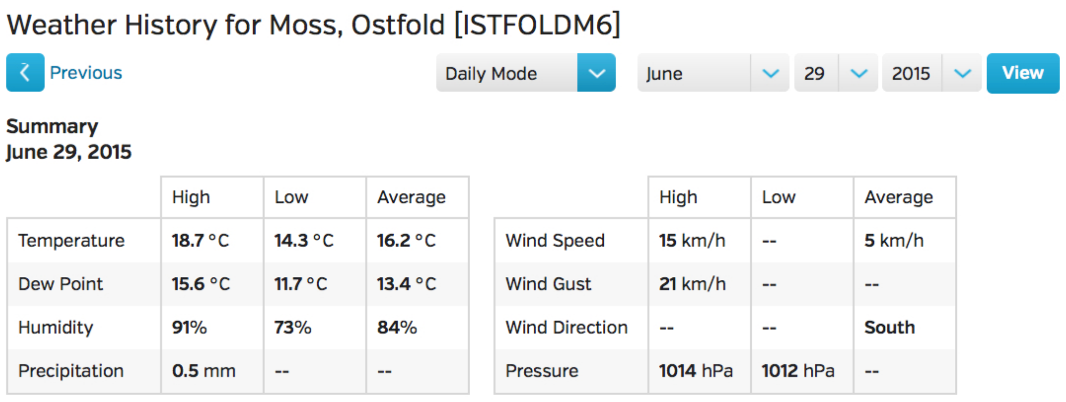
Data plots are processed using *gnuplot*[25]. Additional plots can be created using standard *gnuplot* techniques.

Note that the `DATAERROR` key in the *wxDict* dictionary will inhibit output to the CSV file, Weather Underground, weathercloud.net, WindGURU, WindFinder, APRS, etc. if set to `True`. This will only occur if out-of-range sensor values are detected. The `DATAERROR` key does not presently cover all values, meaning that end-users should still be aware of possible out-of-range sensor readings.

7.1 Weather Underground

Uploads to *Weather Underground* require the LOOP2 packet format, supported by console firmware versions 1.90 and later. Figures 17 and 18 refer.

Data uploads takes place at the interval specified by the `CSVINTERVAL` setting.



Summary
June 29, 2015

	High	Low	Average
Temperature	18.7 °C	14.3 °C	16.2 °C
Dew Point	15.6 °C	11.7 °C	13.4 °C
Humidity	91%	73%	84%
Precipitation	0.5 mm	--	--

	High	Low	Average
Wind Speed	15 km/h	--	5 km/h
Wind Gust	21 km/h	--	--
Wind Direction	--	--	South
Pressure	1014 hPa	1012 hPa	--

Figure 17: Weather Underground output — data is uploaded to the *wunderground.com* website at 10-minute intervals. This feature requires data from the LOOP2 data packets, available in console firmware versions 1.90 and later.

[23] Comma-Separated Values
[24] Extensible Markup Language
[25] `http://www.gnuplot.info`

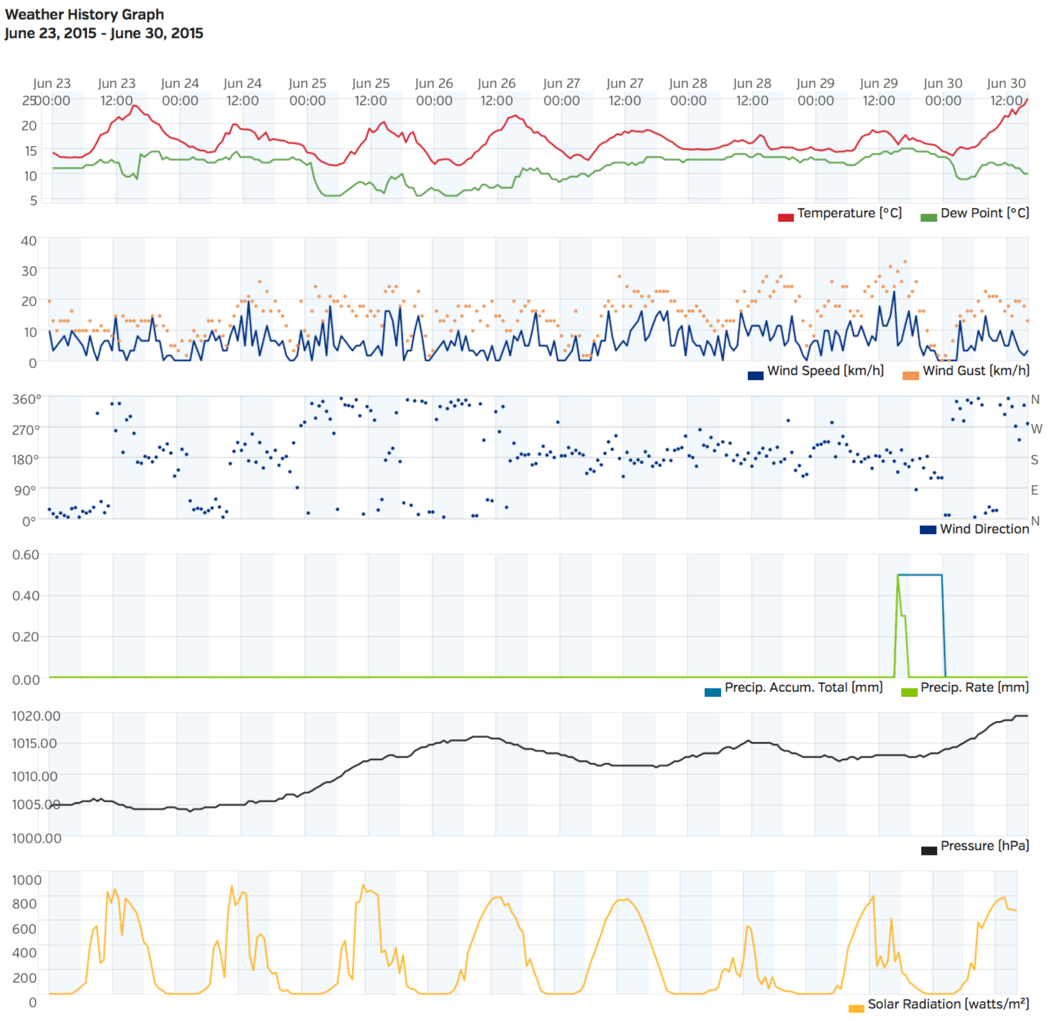
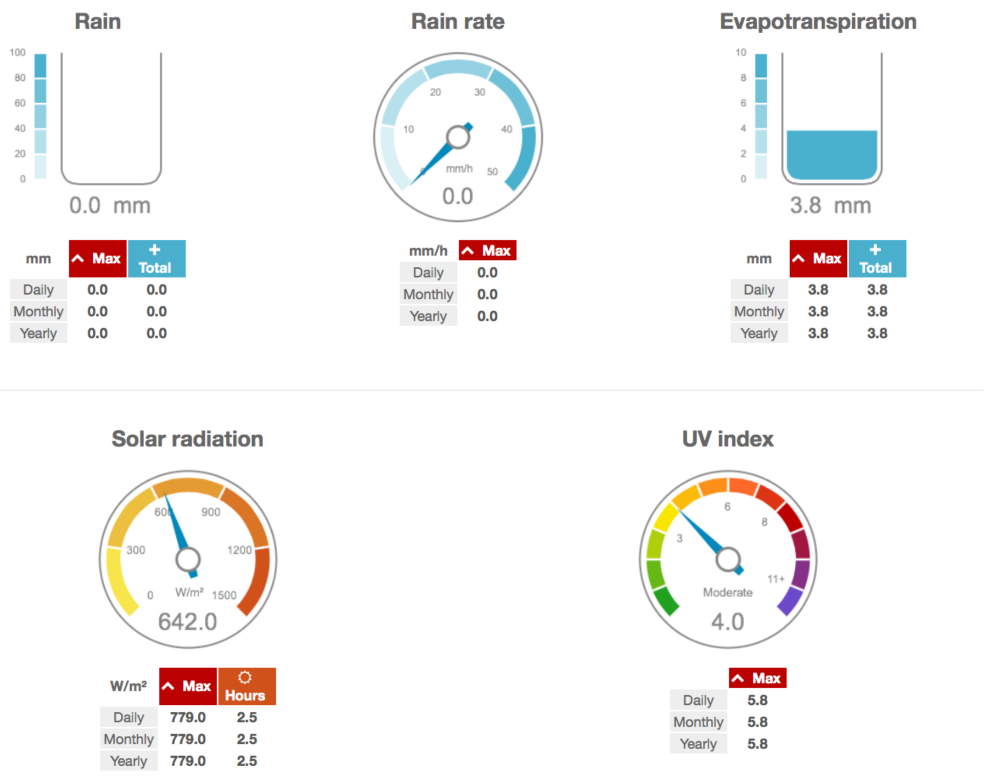


Figure 18: Weather Underground output — WOSPi-provided data processed by Weather Underground, resulting in plots of historical weather data.

7.2 weathercloud.net

Uploads to *weathercloud.net* require the LOOP2 packet format, supported by console firmware versions 1.90 and later. The `WC_ID` and `WC_KEY` options in the `config.py` configuration file should either be left empty or contain the *weathercloud.net* station ID and access key/password. Once the data has been processed by *weathercloud.net*, it will be available for presentation as shown in figure 19. Data uploads take place at the interval specified by the `CSVINTERVAL` setting.



Rain

mm	Max	Total
Daily	0.0	0.0
Monthly	0.0	0.0
Yearly	0.0	0.0

Rain rate

mm/h	Max
Daily	0.0
Monthly	0.0
Yearly	0.0

Evapotranspiration

mm	Max	Total
Daily	3.8	3.8
Monthly	3.8	3.8
Yearly	3.8	3.8

Solar radiation

W/m²	Max	Hours
Daily	779.0	2.5
Monthly	779.0	2.5
Yearly	779.0	2.5

UV index

	Max
Daily	5.8
Monthly	5.8
Yearly	5.8

Figure 19: weathercloud.net presentation of WOSPi-provided weather data.

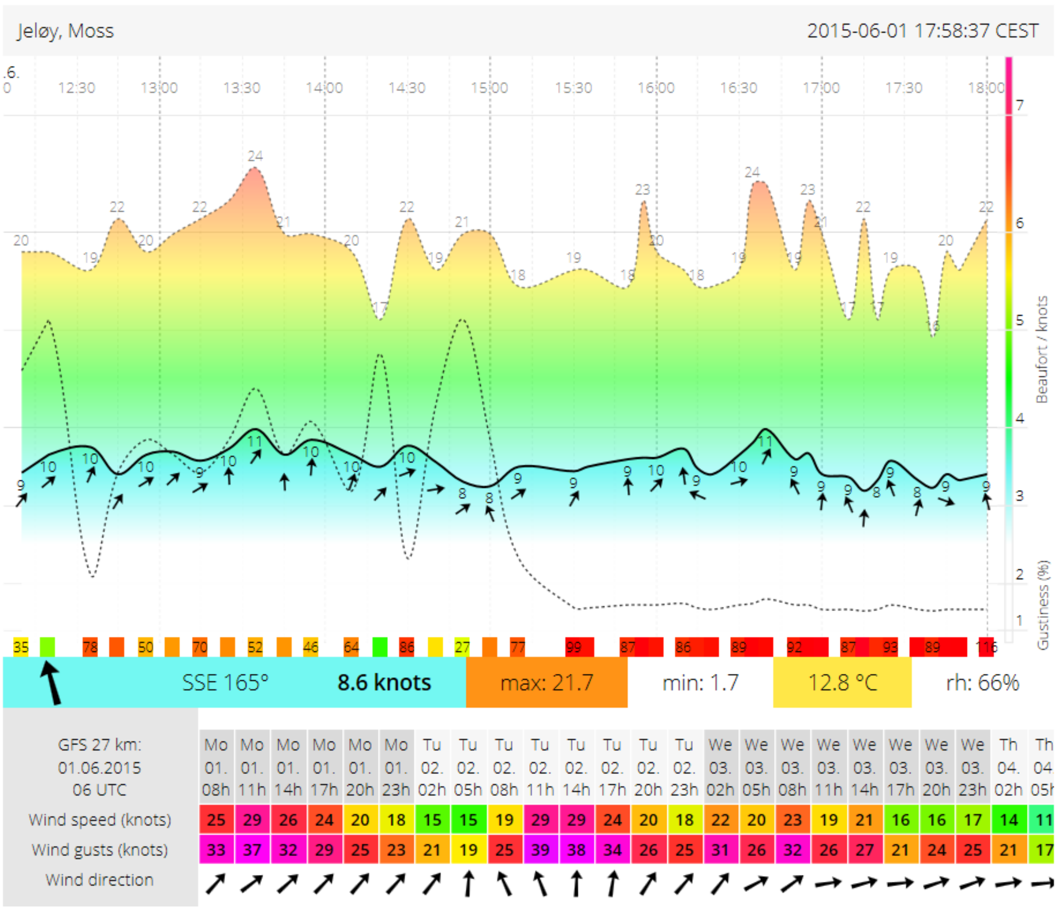
New *weathercloud.net* users should first register with the *weathercloud.net* service[26] and obtain a unique station ID and access key/password. The station ID and access key/password should then be inserted into the `config.py` configuration file.

WOSPi users should register their station type as "Davis WeatherLink" (*not* the "Davis WeatherLink/Network" station type) as *WOSPi* mimics the WeatherLink-interface during uploads to the weathercloud.net service.

[26] `http://www.weathercloud.net`

7.3 WindGURU

Uploads to *WindGURU* require the LOOP2 packet format, supported by console firmware versions 1.90 and later. The WG_UID option in the config.py configuration file should either be left empty or contain the *WindGURU* station ID (UID). Once the data has been processed by *WindGURU*, it will be available for presentation as shown in figure 20. Data uploads takes place at 5-minute intervals as requested by the *WindGURU* developers.



Jeløy, Moss — 2015-06-01 17:58:37 CEST

SSE 165° | 8.6 knots | max: 21.7 | min: 1.7 | 12.8 °C | rh: 66%

GFS 27 km: 01.06.2015 06 UTC	Mo 01. 08h	Mo 01. 11h	Mo 01. 14h	Mo 01. 17h	Mo 01. 20h	Mo 01. 23h	Tu 02. 02h	Tu 02. 05h	Tu 02. 08h	Tu 02. 11h	Tu 02. 14h	Tu 02. 17h	Tu 02. 20h	Tu 02. 23h	We 03. 02h	We 03. 05h	We 03. 08h	We 03. 11h	We 03. 14h	We 03. 17h	We 03. 20h	We 03. 23h	Th 04. 02h	Th 04 05
Wind speed (knots)	25	29	26	24	20	18	15	15	19	29	29	24	20	18	22	20	23	19	21	16	16	17	14	11
Wind gusts (knots)	33	37	32	29	25	23	21	19	25	39	38	34	26	25	31	26	32	26	27	21	24	25	21	17
Wind direction	↗	↗	↗	↗	↗	↗	↗	↑	↖	↖	↑	↑	↗	↗	↗	↗	↗	→	→	→	↗	↗	→	→

Figure 20: WindGURU presentation of WOSPi-provided weather data.

New *WindGURU* users should first register with the *WindGURU* service[27] and obtain a unique station UID. The station UID should then be inserted into the config.py configuration file.

[27] https://stations.windguru.cz/register.php

7.4 WindFinder

Uploads to *WindFinder* require the LOOP2 packet format, supported by console firmware versions 1.90 and later. The `WF_StationID` and `WF_Password` options in the `config.py` configuration file should either be left empty or contain the *WindFinder* station ID and the associated password. Data uploads takes place at the interval specified by the `CSVINTERVAL` setting.

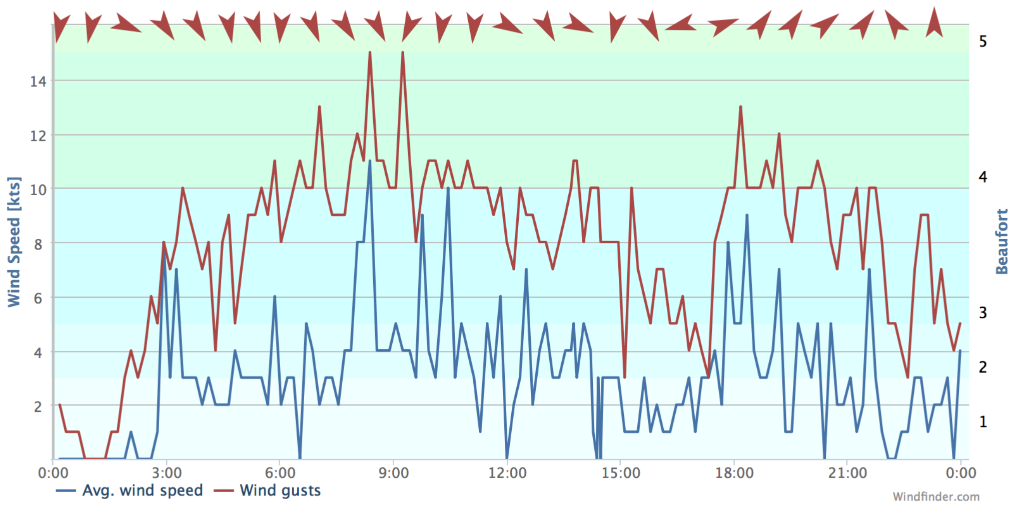


Figure 21: WindFinder presentation of WOSPi-provided wind data.

7.5 Plain-text weather report of current observations

Figure 22 refers.

7.6 Plain-text min/max report

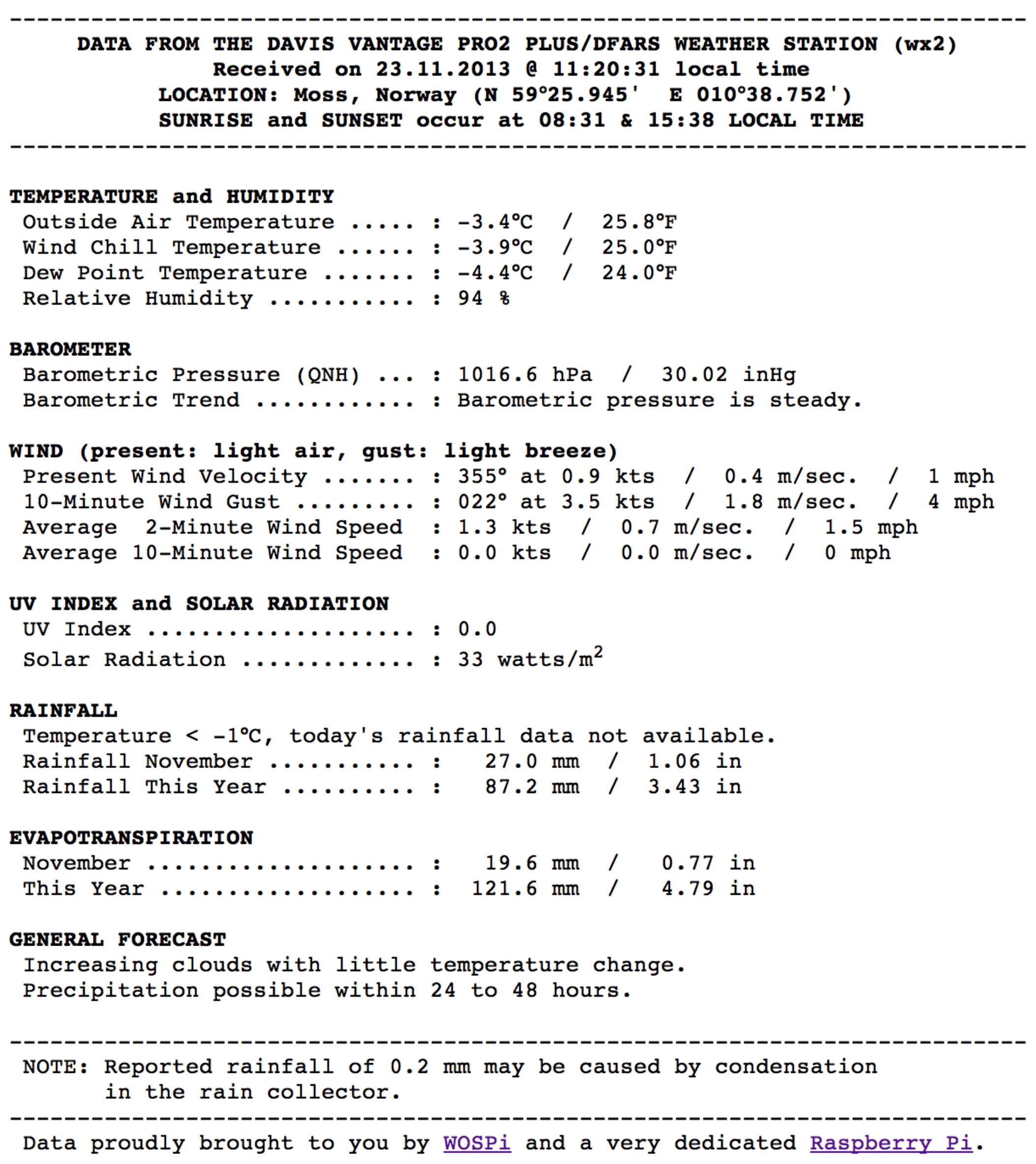
The min/max textual report contains the min/max values as returned by the console as well as various console status parameters. Console `RXCHECK` values are included, provided they are properly returned by the console. When available, the `RXCHECK` values will appear according to the following format: `TOT:0, LST:0, RESYNC:0, CONT:0, CRC:0`. If unavailable, "Not available" will be indicated. Section 9.6 contains additional details. Figures 23 and 24 refer.

7.7 Current observations as XML data

A simple, XML-like output of the *wxDict* dictionary is provided by the `writeXML` function. The XML file is updated twice a minute. The default filename for the XML file is `wxdata.xml`. Listing 10 refers.

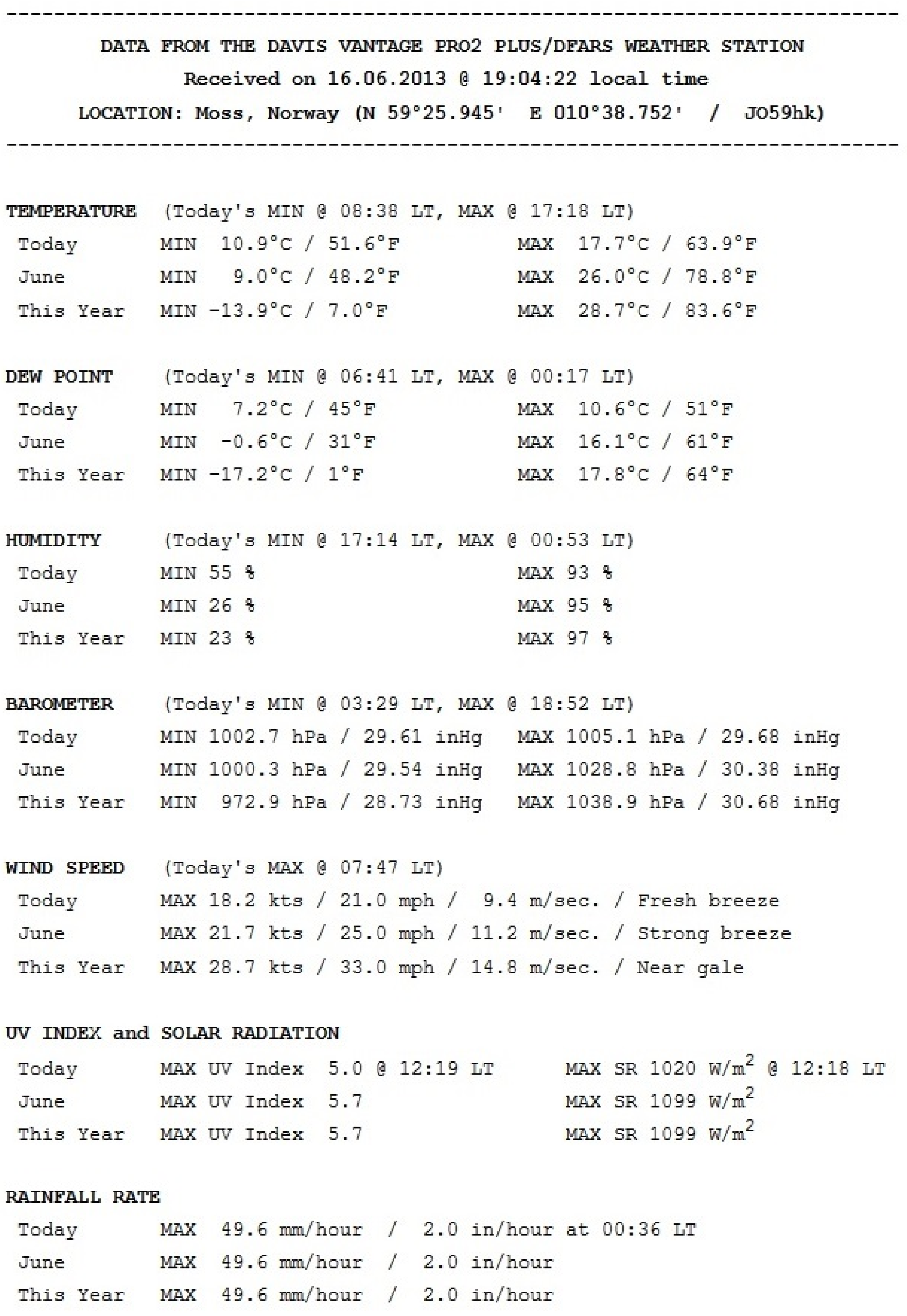
Listing 10: XML-like listing of the *wxDict* dictionary, as produced by the `writeXML` function. The below listing may not be complete and is provided as an example only.

```
<?xml version="1.0"?>
<wxdata>
<bardata>
OK
BAR 29802
ELEVATION 66
DEW POINT 11
VIRTUAL TEMP 29
C 4
R 1003
BARCAL 0
GAIN 0
OFFSET -30
</bardata>
<avgwind2_mph>5.7</avgwind2_mph>
<solar_w>97</solar_w>
<barometer_inhg>29.8</barometer_inhg>
<avgwind2_msec>2.5</avgwind2_msec>
<stormrain_mm>0.0</stormrain_mm>
<gust10_msec>5.4</gust10_msec>
<inhum_p>22</inhum_p>
<wind_msec>2.2</wind_msec>
<rainfall24h_mm>0.0</rainfall24h_mm>
<dewpoint_f>11.0</dewpoint_f>
<dewpoint_c>-11.7</dewpoint_c>
<batterystatus>0</batterystatus>
<winddir>051</winddir>
<gust10dir>022</gust10dir>
<outhum_p>51</outhum_p>
<et_month_mm>11.3</et_month_mm>
<avgwind10_mph>5.5</avgwind10_mph>
<avgwind10_kts>4.8</avgwind10_kts>
<voltage>4.0</voltage>
<avgwind10_msec>2.5</avgwind10_msec>
<timestamp_pc>2013-03-13 12:34:57.641141</timestamp_pc>
<intemp_c>19.7</intemp_c>
<barotrend>0</barotrend>
<intemp_f>67.4</intemp_f>
<rainrate_mmhr>0.0</rainrate_mmhr>
<crc_pad>2567</crc_pad>
<ver>Sep 29 2009 </ver>
<nver>1.90 </nver>
<et_year_mm>23.1</et_year_mm>
<fctext>Increasing clouds with little temperature change.</fctext>
<uvindex>1.4</uvindex>
<stormstart>01.01.1970</stormstart>
<gust10_mph>12</gust10_mph>
<gust10_kts>10.4</gust10_kts>
<sunset_lt>18:16</sunset_lt>
<wind_kts>4.3</wind_kts>
<wind_mph>5</wind_mph>
<rainfall15_mm>0.0</rainfall15_mm>
<wc_c>-5.6</wc_c>
<fcicon>6</fcicon>
<barotrendtext>Barometric pressure is steady.</barotrendtext>
<wc_f>22.0</wc_f>
<timestamp_aprs>Mar 13 2013 12:34</timestamp_aprs>
<timestamp>13.03.2013 12:34:54</timestamp>
<timestamp_wx>Received on 13.03.2013 @ 12:34:54 local time</timestamp_wx>
<sunrise_lt>06:41</sunrise_lt>
<dayrain_mm>0.0</dayrain_mm>
<monthrain_mm>0.0</monthrain_mm>
<yearrain_mm>0.0</yearrain_mm>
<crc-calc>0</crc-calc>
<rainfall60_mm>0.0</rainfall60_mm>
<outtemp_f>27.5</outtemp_f>
<avgwind2_kts>5.0</avgwind2_kts>
<et_day_mm>0.5</et_day_mm>
<outtemp_c>-2.5</outtemp_c>
<barometer_hpa>1009.1</barometer_hpa>
<freeze>True</freeze>
<condensation>False</condensation>
<stationmodel>Vantage Pro/Vantage Pro2 (16)</stationmodel>
<dataerror>False</dataerror>
</wxdata>
```



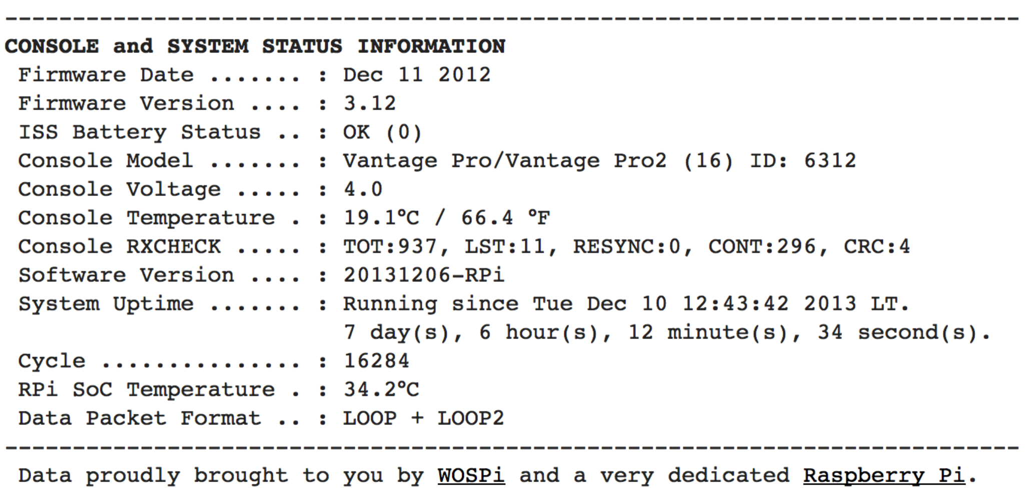
```
----------------------------------------------------------------------------
     DATA FROM THE DAVIS VANTAGE PRO2 PLUS/DFARS WEATHER STATION (wx2)
               Received on 23.11.2013 @ 11:20:31 local time
           LOCATION: Moss, Norway (N 59°25.945'  E 010°38.752')
           SUNRISE and SUNSET occur at 08:31 & 15:38 LOCAL TIME
----------------------------------------------------------------------------

TEMPERATURE and HUMIDITY
 Outside Air Temperature ..... : -3.4°C  /  25.8°F
 Wind Chill Temperature ...... : -3.9°C  /  25.0°F
 Dew Point Temperature ....... : -4.4°C  /  24.0°F
 Relative Humidity ........... : 94 %

BAROMETER
 Barometric Pressure (QNH) ... : 1016.6 hPa  /  30.02 inHg
 Barometric Trend ............ : Barometric pressure is steady.

WIND (present: light air, gust: light breeze)
 Present Wind Velocity ....... : 355° at 0.9 kts  /  0.4 m/sec.  /  1 mph
 10-Minute Wind Gust ......... : 022° at 3.5 kts  /  1.8 m/sec.  /  4 mph
 Average  2-Minute Wind Speed  : 1.3 kts  /  0.7 m/sec.  /  1.5 mph
 Average 10-Minute Wind Speed  : 0.0 kts  /  0.0 m/sec.  /  0 mph

UV INDEX and SOLAR RADIATION
 UV Index .................... : 0.0
 Solar Radiation ............. : 33 watts/m^2

RAINFALL
 Temperature < -1°C, today's rainfall data not available.
 Rainfall November ........... :   27.0 mm  /  1.06 in
 Rainfall This Year .......... :   87.2 mm  /  3.43 in

EVAPOTRANSPIRATION
 November .................... :   19.6 mm  /   0.77 in
 This Year ................... :  121.6 mm  /   4.79 in

GENERAL FORECAST
 Increasing clouds with little temperature change.
 Precipitation possible within 24 to 48 hours.

----------------------------------------------------------------------------
 NOTE: Reported rainfall of 0.2 mm may be caused by condensation
       in the rain collector.
----------------------------------------------------------------------------
 Data proudly brought to you by WOSPi and a very dedicated Raspberry Pi.
```

Figure 22: Plain-text weather report of current observations, which can easily be embedded on a HTML page. The current weather report is updated twice a minute.



```
---------------------------------------------------------------------------
        DATA FROM THE DAVIS VANTAGE PRO2 PLUS/DFARS WEATHER STATION
               Received on 16.06.2013 @ 19:04:22 local time
      LOCATION: Moss, Norway (N 59°25.945'  E 010°38.752'  /  JO59hk)
---------------------------------------------------------------------------

TEMPERATURE  (Today's MIN @ 08:38 LT, MAX @ 17:18 LT)
 Today       MIN  10.9°C / 51.6°F          MAX  17.7°C / 63.9°F
 June        MIN   9.0°C / 48.2°F          MAX  26.0°C / 78.8°F
 This Year   MIN -13.9°C / 7.0°F           MAX  28.7°C / 83.6°F

DEW POINT    (Today's MIN @ 06:41 LT, MAX @ 00:17 LT)
 Today       MIN   7.2°C / 45°F            MAX  10.6°C / 51°F
 June        MIN  -0.6°C / 31°F            MAX  16.1°C / 61°F
 This Year   MIN -17.2°C / 1°F             MAX  17.8°C / 64°F

HUMIDITY     (Today's MIN @ 17:14 LT, MAX @ 00:53 LT)
 Today       MIN 55 %                      MAX 93 %
 June        MIN 26 %                      MAX 95 %
 This Year   MIN 23 %                      MAX 97 %

BAROMETER    (Today's MIN @ 03:29 LT, MAX @ 18:52 LT)
 Today       MIN 1002.7 hPa / 29.61 inHg   MAX 1005.1 hPa / 29.68 inHg
 June        MIN 1000.3 hPa / 29.54 inHg   MAX 1028.8 hPa / 30.38 inHg
 This Year   MIN  972.9 hPa / 28.73 inHg   MAX 1038.9 hPa / 30.68 inHg

WIND SPEED   (Today's MAX @ 07:47 LT)
 Today       MAX 18.2 kts / 21.0 mph /  9.4 m/sec. / Fresh breeze
 June        MAX 21.7 kts / 25.0 mph / 11.2 m/sec. / Strong breeze
 This Year   MAX 28.7 kts / 33.0 mph / 14.8 m/sec. / Near gale

UV INDEX and SOLAR RADIATION
 Today       MAX UV Index  5.0 @ 12:19 LT      MAX SR 1020 W/m2 @ 12:18 LT
 June        MAX UV Index  5.7                 MAX SR 1099 W/m2
 This Year   MAX UV Index  5.7                 MAX SR 1099 W/m2

RAINFALL RATE
 Today       MAX  49.6 mm/hour  /  2.0 in/hour at 00:36 LT
 June        MAX  49.6 mm/hour  /  2.0 in/hour
 This Year   MAX  49.6 mm/hour  /  2.0 in/hour
```

Figure 23: Plain-text min/max report, by default updated at 10-minute intervals.



```
----------------------------------------------------------------------------
CONSOLE and SYSTEM STATUS INFORMATION
 Firmware Date ....... : Dec 11 2012
 Firmware Version .... : 3.12
 ISS Battery Status .. : OK (0)
 Console Model ....... : Vantage Pro/Vantage Pro2 (16) ID: 6312
 Console Voltage ..... : 4.0
 Console Temperature . : 19.1°C / 66.4 °F
 Console RXCHECK ..... : TOT:937, LST:11, RESYNC:0, CONT:296, CRC:4
 Software Version .... : 20131206-RPi
 System Uptime ....... : Running since Tue Dec 10 12:43:42 2013 LT.
                         7 day(s), 6 hour(s), 12 minute(s), 34 second(s).
 Cycle ............... : 16284
 RPi SoC Temperature . : 34.2°C
 Data Packet Format .. : LOOP + LOOP2
----------------------------------------------------------------------------
 Data proudly brought to you by WOSPi and a very dedicated Raspberry Pi.
```

Figure 24: Console status information, included in the plain-text min/max report.

7.8 24-hour data plot — wind speed and wind direction

Figure 25 refers.

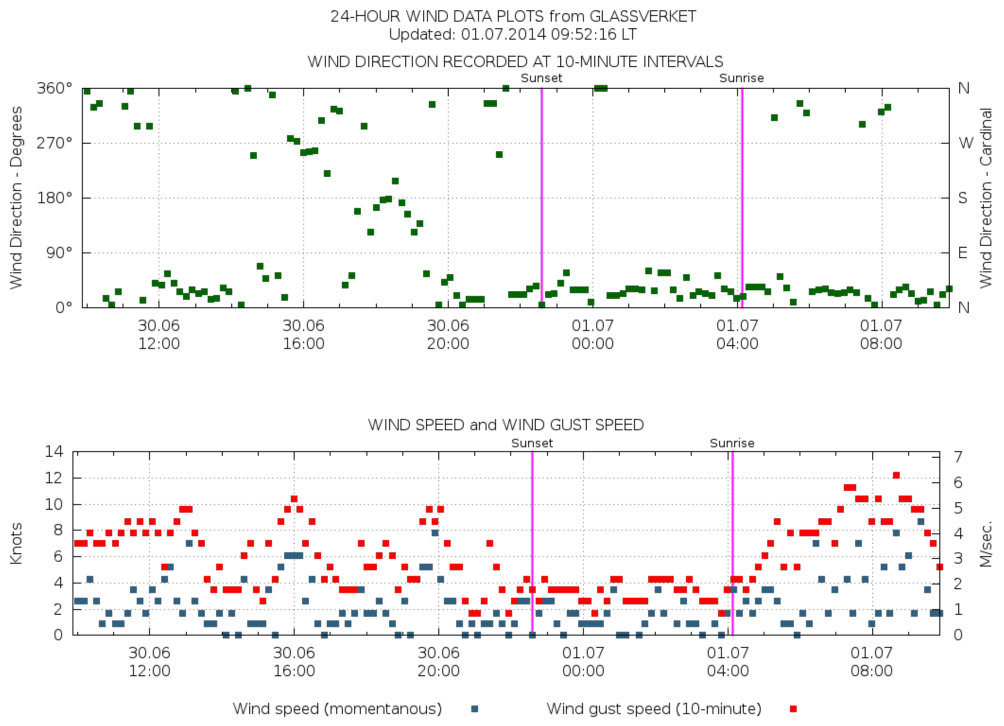


Figure 25: 24-hour data plot showing wind direction, wind speed and wind gust speed. The plot can be configured for knots & m/sec. *or* knots & MPH units. Refer to comments embedded in the `plot24windL1.input` or `plot24windL2.input` and `config.py` files for configuration options.

Configuration details

- *gnuplot* script file: `plot24windL1.input` *or* `plot24windL2.input`
- resulting output file: `wind_24hr.png`
- default update interval: 10 minutes
- `config.py` configuration options:
  - `CSVINTERVAL`
  - `PLOT24WINDTITLE`
  - `PLOT24WIND` (path + file name)
  - `SCPCOMMAND_PLOT24WIND`

7.9 24-hour data plot — temperature, solar/UV radiation, barometric pressure

Figure 26 refers. Temperature can be plotted in °C or °F. Refer to customisation options and comments embedded in the `plot24.input` file.

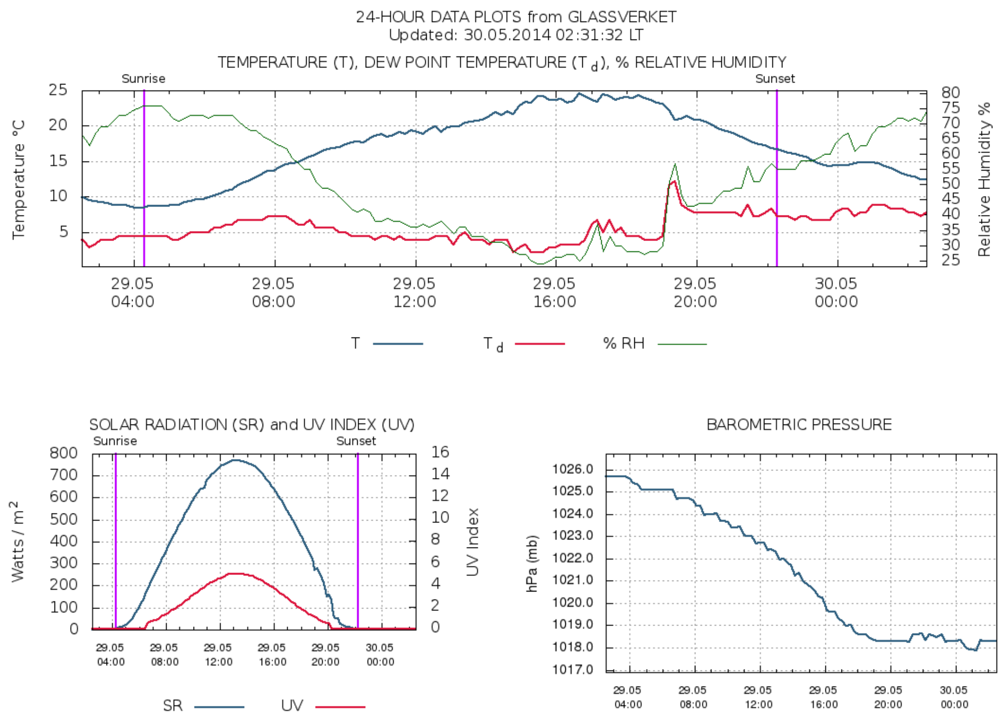


Figure 26: 24-hour data plot showing temperature (°C or °F), dew point temperature (°C or °F), relative humidity, solar radiation, UV index and barometric pressure (hPa/mb or inHg).

Configuration details

- *gnuplot* script file: `plot24.input`
- resulting output file: `wx_24hr.png`
- default update interval: 10 minutes
- `config.py` configuration options:
  - `CSVINTERVAL`
  - `PLOT24TITLE`
  - `PLOT24FILE` (path + file name)
  - `SCPCOMMAND_PLOT24FILE`

7.10 Last month's rainfall histogram

Figure 27 refers. Refer to comments embedded in the `plotRainMonth.input` file for customisation options. The `COMMISSIONDATE` setting in the `config.py` configuration file should be set according to the instructions provided in section 8.5.1.

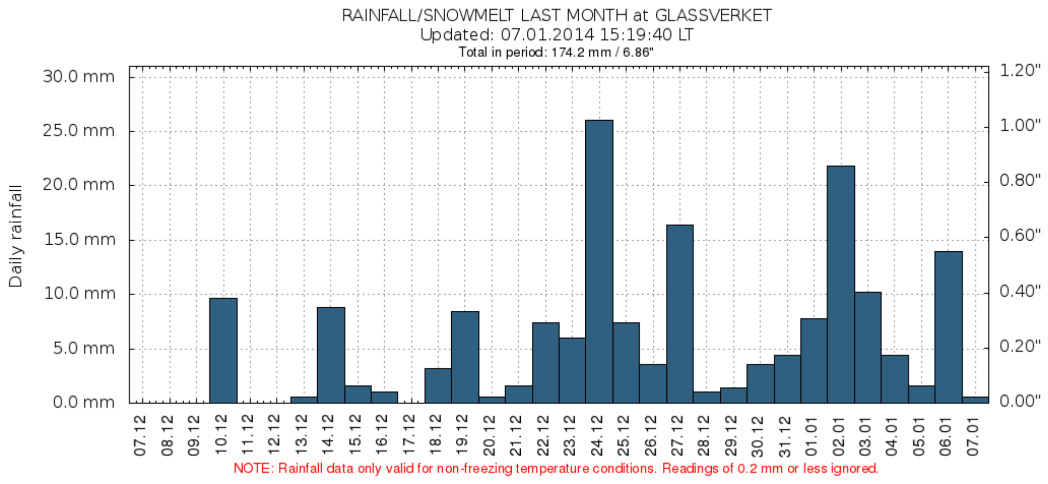


Figure 27: Last month's rainfall data.

Configuration details

- *gnuplot* script file: `plotRainMonth.input`
- resulting output file: `plotraindays.png`
- default update interval: 10 minutes
- `config.py` configuration options:
  - `CSVINTERVAL`
  - `COMMISSIONDATE`
  - `PLOTRAINMONTHTITLE`
  - `PLOTRAINMONTH` (path + file name)
  - `SCPCOMMAND_PLOTRAINMONTH`
  - `RAINTHRESHOLD_MM`
  - `RAINTHRESHOLDTEXT`

7.11 Rainfall per month histogram

Figure 28 refers. Refer to comments embedded in the `plotRainPerMonth.input` file for configuration options. The `COMMISSIONDATE` setting in the `config.py` configuration file should be set according to the instructions provided in section 8.5.1.

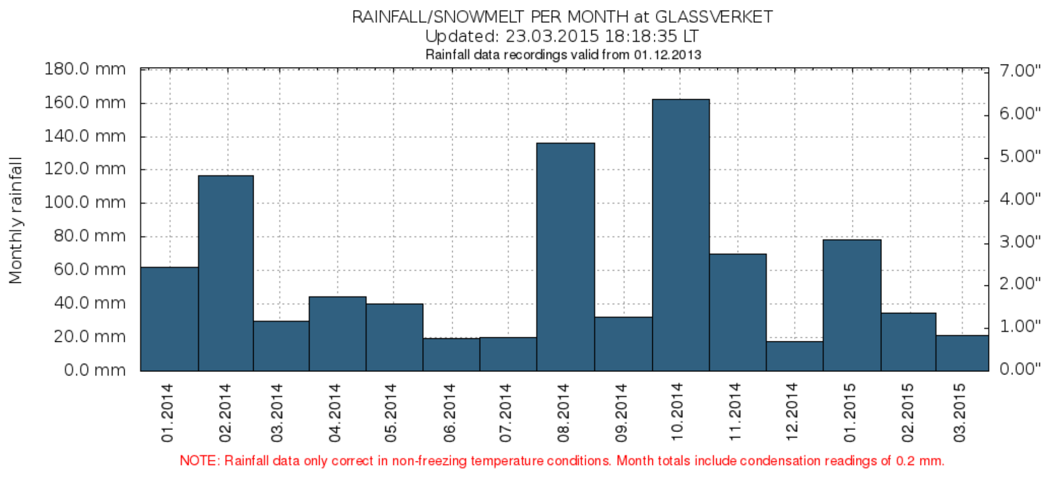


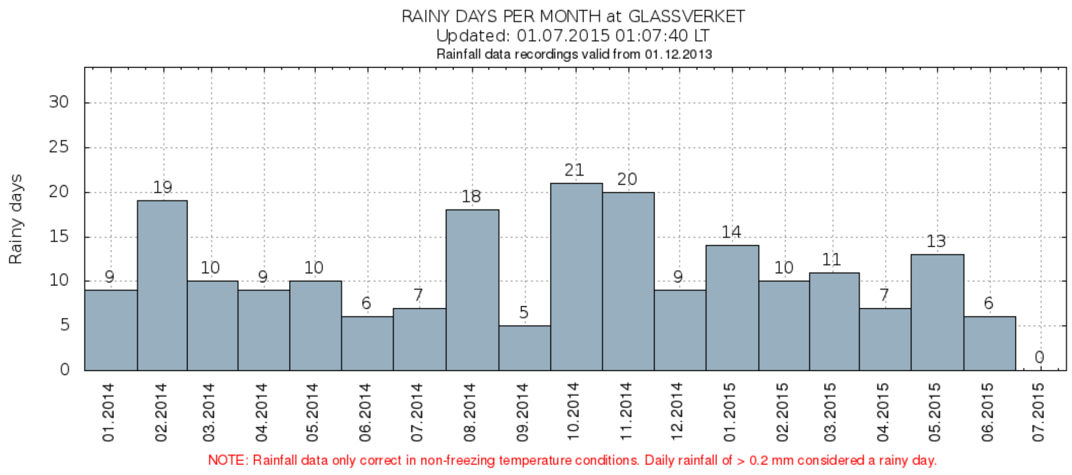
Figure 28: The monthly rainfall values are read directly from the console and may include small erroneous rainfall values caused by condensation in the rainfall collector.

Configuration details

- *gnuplot* script file: `plotRainPerMonth.input`
- resulting output file: `monthlyrain.png`
- default update interval: 10 minutes
- `config.py` configuration options:
  - `CSVINTERVAL`
  - `COMMISSIONDATE`
  - `PLOTRAINPERMONTHTITLE`
  - `PLOTRAINPERMONTH` (path + file name)
  - `SCPCOMMAND_PLOTRAINPERMONTH`
  - `RAINTHRESHOLD_MM`
  - `RAINTHRESHOLDTEXT`

7.12 Rainy days per month histogram

Figure 29 refers. The `COMMISSIONDATE` setting in the `config.py` configuration file should be set according to the instructions provided in section 8.5.1.



Month	Rainy days
01.2014	9
02.2014	19
03.2014	10
04.2014	9
05.2014	10
06.2014	6
07.2014	7
08.2014	18
09.2014	5
10.2014	21
11.2014	20
12.2014	9
01.2015	14
02.2015	10
03.2015	11
04.2015	7
05.2015	13
06.2015	6
07.2015	0

Figure 29: Number of rainy days per month. The histogram includes data for the last 24 months (provided data is available).

Configuration details

- *gnuplot* script file: `plotRainDaysPerMonth.input`
- resulting output file: `monthlyraindays.png`
- default update interval: 10 minutes
- `config.py` configuration options:
  - `CSVINTERVAL`
  - `COMMISSIONDATE`
  - `PLOTRAINDAYSPERMONTHTITLE`
  - `PLOTRAINDAYSPERMONTH` (path + file name)
  - `SCPCOMMAND_PLOTRAINDAYSPERMONTH`
  - `RAINTHRESHOLD_MM`
  - `RAINTHRESHOLDTEXT`

7.13 Min/max temperatures from the last 12 months

Figure 30 refers. There is normally no need to update the min/max temperature plot more than once or possibly twice a day. There is a separate *Python* program named `plotMinMaxTemp.py` which should be added to the system's *crontab* to achieve this, as *WOSPi* will not itself update the min/max temperature plot. Section 9.8 refers.

The temperature values are read from the CSV file as described in section 7.17. There may as such be slight deviations from the min/max temperature readings reported in the HILOWS packets.

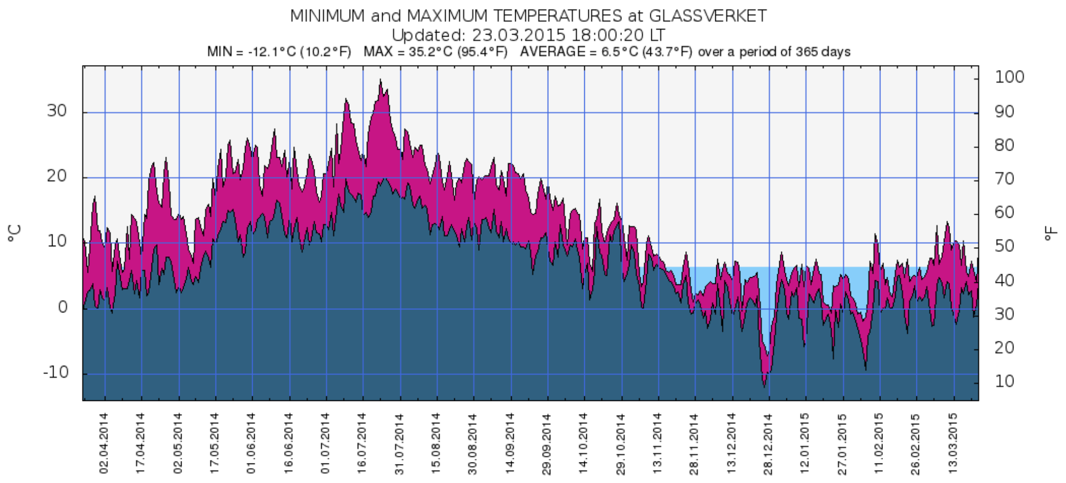


Figure 30: Minimum and maximum temperature readings for a period of up to 12 months. The background is colored light gray/light blue to indicate temperature readings above/below the average for the period shown in the data plot.

Configuration details

- *gnuplot* script file: `plotMinMaxTemp.input`
- .py program file: `plotMinMaxTemp.py`
- resulting output file: `plotminmax.png`
- default update interval: as specified in the system's *crontab*
- `config.py` configuration options:
  - `PLOTMINMAXTITLE`
  - `PLOTMINMAXTEMP` (path + file name)
  - `SCPCOMMAND_PLOTMINMAXTEMP`

7.14 Daily max solar and max UV radiation from the last 12 months

Figure 31 refers. There is normally no need to update the max daily solar/max daily UV radiation plot more than once a day, preferably in the late afternoon/early evening. There is a separate *Python* program named `plotSolar.py` which should be added to the system's *crontab* to achieve this, as *WOSPi* will not itself update the max daily solar/max daily UV radiation plot. Section 9.8 refers.

The radiation values are read from the CSV file as described in section 7.17. There may as such be slight deviations from the values reported in the HILOWS packets.

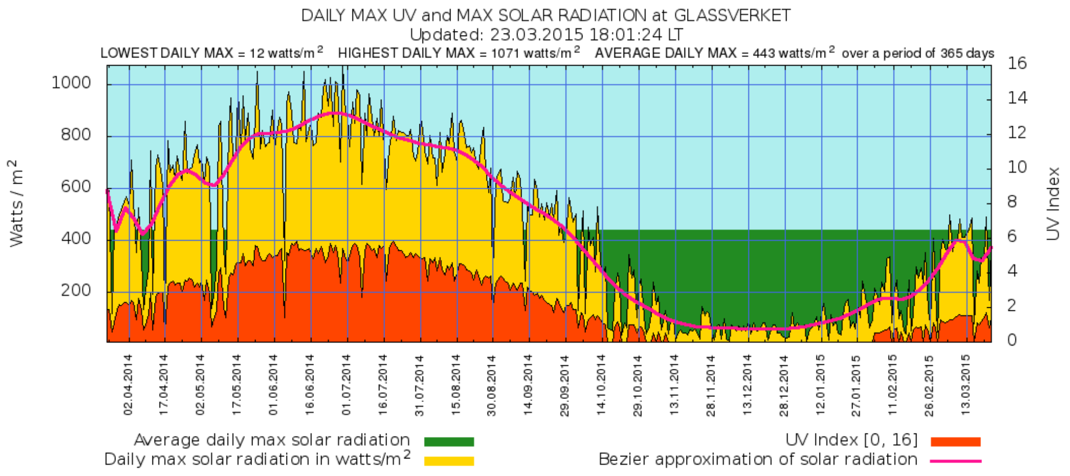


Figure 31: Plot of daily maximum solar radiation and UV radiation readings for a period of up to 12 months. The background is colored light blue/green to indicate solar radiation readings above/below the average for the period shown in the data plot. The Bezier approximation of solar radiation readings is included to show the solar radiation trend — it should be falling towards winter and increasing towards summer.

Configuration details

- *gnuplot* script file: `plotSolar.input`
- .py program file: `plotSolar.py`
- resulting output file: `plotsolar.png`
- default update interval: as specified in the system's *crontab*
- `config.py` configuration options:
  - `PLOTSOLARTITLE`
  - `PLOTSOLAR` (path + file name)
  - `SCPCOMMAND_PLOTSOLAR`

7.15 Daily max solar radiation and temperature from the last 12 months

Figure 32 refers. There is normally no need to update the max daily temperature/solar radiation plot more than once a day, preferably in the late afternoon/early evening. There is a separate *Python* program named `plotTempSolar.py` which should be added to the system's *crontab* to achieve this, as *WOSPi* will not itself update the max daily temperature/solar radiation plot. Section 9.8 refers.

The temperature and radiation values are read from the CSV file as described in section 7.17. There may as such be slight deviations from the values reported in the HILOWS packets. The values are plotted using smoothed Bezier curves.

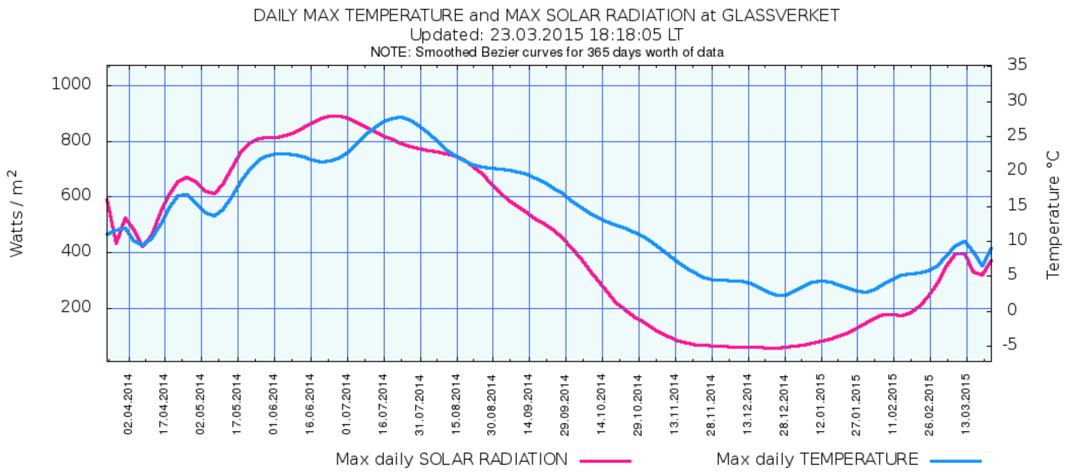


Figure 32: Plot of daily maximum temperature and solar radiation readings for a period of up to 12 months. The Bezier approximations of temperature and solar radiation readings should indicate that "temperature follows solar radiation".

Configuration details

- *gnuplot* script file: `plotTempSolar.input`
- .py program file: `plotTempSolar.py`
- resulting output file: `plottempsolar.png`
- default update interval: as specified in the system's *crontab*
- `config.py` configuration options:
  - `PLOTTEMPSOLARTITLE`
  - `PLOTTEMPSOLAR` (path + file name)
  - `SCPCOMMAND_PLOTTEMPSOLAR`

7.16 One week of barometric pressure data

Figure 33 refers. There is normally no need to update the one-week barometric pressure plot more than once or possibly twice a day. There is a separate *Python* program named `plotBaroWeek.py` which should be added to the system's *crontab* to achieve this, as *WOSPi* will not itself update the one-week barometric pressure plot. Section 9.8 refers.

Barometric pressure readings are extracted from the CSV file as described in section 7.17.

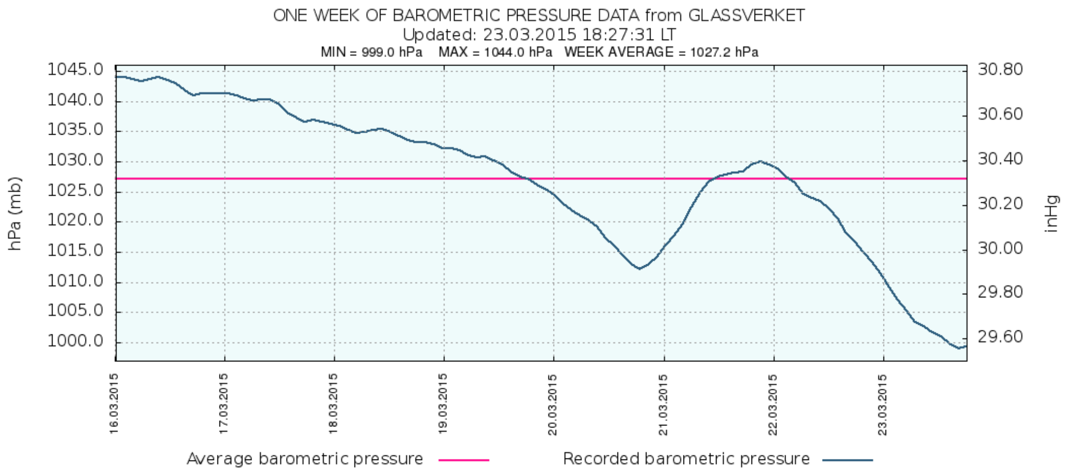


Figure 33: Plot of one week of barometric pressure data.

Configuration details

- *gnuplot* script file: `plotBaroWeek.input`
- .py program file: `plotBaroWeek.py`
- resulting output file: `baroweek.png`
- default update interval: as specified in the system's *crontab*
- `config.py` configuration options:
  - `PLOTBAROWEEKTITLE`
  - `PLOTBAROWEEK` (path + file name)
  - `SCPCOMMAND_PLOTBAROWEEK`

7.17 Weather observations CSV file format

The CSV file stores a complete set of weather observation every 10 minutes (or as specified by the `CSVINTERVAL` value in the configuration file). The CSV file can easily be analyzed by other software (e.g. spreadsheet or database applications). A separate CSV file contains rainfall data, section 7.18 refers.

Two lines from an actual CSV file are shown below (a few blank lines have been added for clarity — the actual CSV file has no blank lines in it), each line consists of 17 different values. Comma (`,`) is used as field separator, dot (`.`) is used as decimal separator. Each line contains one data set (one observation), terminated by the newline/LF (`\n`) character.

```
01.03.2013 00:26:48,3.8,54,-5.0,1016.9,323,1.7,0.0,0,0.0,0.0,0.0,0.0,2.8,2.3,7.0,360

01.03.2013 00:36:56,3.6,55,-5.0,1017.3,325,2.6,0.0,0,0.0,0.0,0.0,0.0,2.2,2.4,5.2,337
```

A new CSV file is created each month. The file name indicates the year and month number of the observations, e.g. `2013-03-wxdata.csv` for March, 2013. File size at the end of the month will vary between 250 and 400 kB, depending on the observation data. The CSV files contain the baseline data used to generate the data plots as shown in figures 25, 26, 30, 31 and 33.

The CSV data field order is as follows :

1. *Timestamp* on `dd.mm.yyyy HH:MM:SS` format
2. *Outside air temperature* in °C
3. *Outside relative humidity*
4. *Outside dew point temperature* in °C
5. *Barometric pressure* in hPa/mb
6. *Present wind direction*
7. *Present wind speed* in knots
8. *UV index* in range [0, 16]
9. *Solar radiation* (watts per m^2) in range [0, 1800]
10. *Rain rate* in mm/hour
11. *Daily rain* in mm
12. *Daily ET* in mm
13. *Monthly ET* in mm
14. *10-minute average wind speed* in knots

15. *2-minute average wind speed* in knots

16. *10-minute wind gust speed* in knots

17. *10-minute wind gust direction*

7.18 Rainfall data CSV file format

Separate CSV files contain daily rainfall values for periods of one month. The rainfall data files are named `yyyy-mm.rain`, where `yyyy` represents the year and `mm` represents the month number, e.g. `2013-10.rain`.

Comma (`,`) is used as field separator, dot (`.`) is used as decimal separator. Each line contains one data set, terminated by the newline/LF (`\n`) character.

The rainfail data file is updated at least once a day. To keep the number of write cycles at a minimum (crucial for flash storage), the file is only updated at the start of a new day *and* whenever new values are available throughout the day. Values are compared, and — if required — updated at 10-minute intervals (or as specified by the `CSVINTERVAL` value in the configuration file).

The next generation of *WOSPi* may store all retrieved data in a single database file. It is, however, unclear just how write-intensive the various open source database systems are. As such, CSV file storage is being utilized until further.

The rainfall data file will contain the following entries:

```
01.10.2013, 0.5, 0.5, 103.2
02.10.2013, 0.5, 1.0, 103,7
...
31.10.2013, 0.0, 42.2, 145.9
```

The CSV data field order is as follows :

1. *Timestamp* on `dd.mm.yyyy` format

2. *Daily rainfall* in mm

3. *Monthly rainfall* (from the 1st of the present month) in mm

4. *Annual rainfall* (refer to section 2.2 for important details) in mm

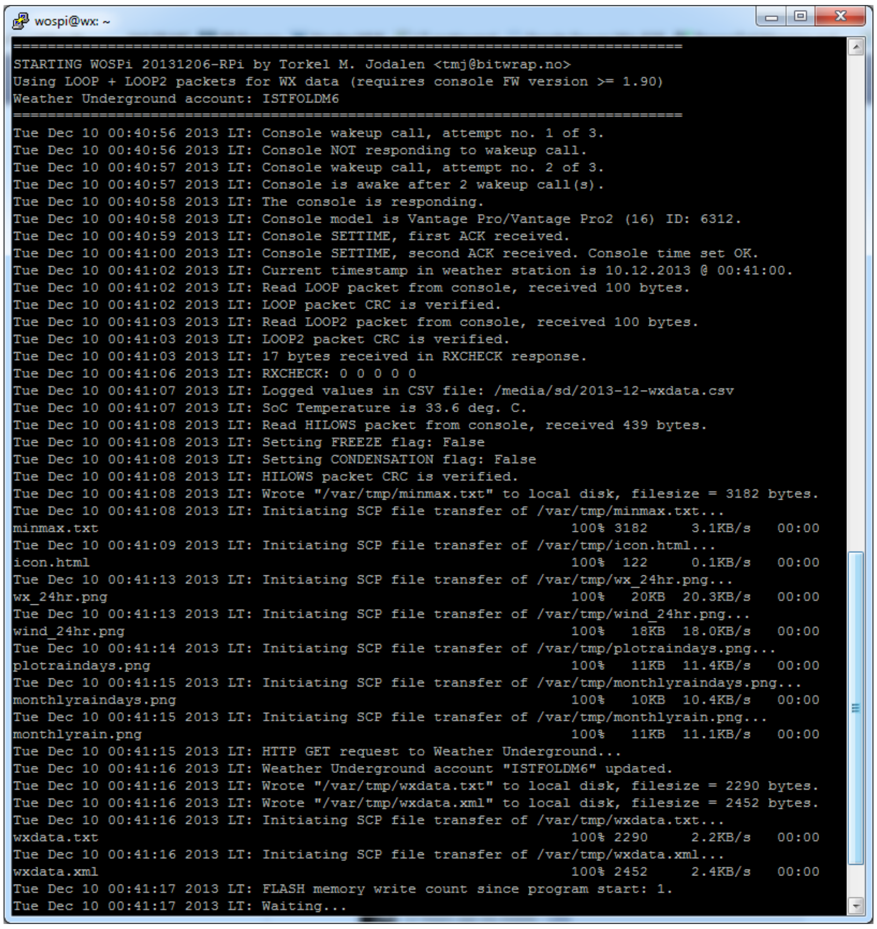
7.19 APRS weather report

For use with the APRS[28] system, an optional APRS weather report can be generated at 10-minute intervals. The report will be stored in the `/var/tmp/uiview.txt` file. Console firmware version $\geq$ 1.90 is required to extract data using the LOOP2 packet format. A typical APRS/UIView weather report file is shown below.

```
Mar 13 2013 12:59
055/003g013t028r000p000P000h49b10091
```

7.20 Terminal output

When run from the console, the *wospi.pyc* program will output a number of status messages during execution. A typical session is shown in figure 34.



```
==============================================================================
STARTING WOSPi 20131206-RPi by Torkel M. Jodalen <tmj@bitwrap.no>
Using LOOP + LOOP2 packets for WX data (requires console FW version >= 1.90)
Weather Underground account: ISTFOLDM6
==============================================================================
Tue Dec 10 00:40:56 2013 LT: Console wakeup call, attempt no. 1 of 3.
Tue Dec 10 00:40:56 2013 LT: Console NOT responding to wakeup call.
Tue Dec 10 00:40:57 2013 LT: Console wakeup call, attempt no. 2 of 3.
Tue Dec 10 00:40:57 2013 LT: Console is awake after 2 wakeup call(s).
Tue Dec 10 00:40:58 2013 LT: The console is responding.
Tue Dec 10 00:40:58 2013 LT: Console model is Vantage Pro/Vantage Pro2 (16) ID: 6312.
Tue Dec 10 00:40:59 2013 LT: Console SETTIME, first ACK received.
Tue Dec 10 00:41:00 2013 LT: Console SETTIME, second ACK received. Console time set OK.
Tue Dec 10 00:41:02 2013 LT: Current timestamp in weather station is 10.12.2013 @ 00:41:00.
Tue Dec 10 00:41:02 2013 LT: Read LOOP packet from console, received 100 bytes.
Tue Dec 10 00:41:02 2013 LT: LOOP packet CRC is verified.
Tue Dec 10 00:41:03 2013 LT: Read LOOP2 packet from console, received 100 bytes.
Tue Dec 10 00:41:03 2013 LT: LOOP2 packet CRC is verified.
Tue Dec 10 00:41:03 2013 LT: 17 bytes received in RXCHECK response.
Tue Dec 10 00:41:06 2013 LT: RXCHECK: 0 0 0 0 0
Tue Dec 10 00:41:07 2013 LT: Logged values in CSV file: /media/sd/2013-12-wxdata.csv
Tue Dec 10 00:41:07 2013 LT: SoC Temperature is 33.6 deg. C.
Tue Dec 10 00:41:08 2013 LT: Read HILOWS packet from console, received 439 bytes.
Tue Dec 10 00:41:08 2013 LT: Setting FREEZE flag: False
Tue Dec 10 00:41:08 2013 LT: Setting CONDENSATION flag: False
Tue Dec 10 00:41:08 2013 LT: HILOWS packet CRC is verified.
Tue Dec 10 00:41:08 2013 LT: Wrote "/var/tmp/minmax.txt" to local disk, filesize = 3182 bytes.
Tue Dec 10 00:41:08 2013 LT: Initiating SCP file transfer of /var/tmp/minmax.txt...
minmax.txt                                                       100% 3182     3.1KB/s   00:00
Tue Dec 10 00:41:09 2013 LT: Initiating SCP file transfer of /var/tmp/icon.html...
icon.html                                                        100%  122     0.1KB/s   00:00
Tue Dec 10 00:41:13 2013 LT: Initiating SCP file transfer of /var/tmp/wx_24hr.png...
wx_24hr.png                                                      100%   20KB  20.3KB/s   00:00
Tue Dec 10 00:41:13 2013 LT: Initiating SCP file transfer of /var/tmp/wind_24hr.png...
wind_24hr.png                                                    100%   18KB  18.0KB/s   00:00
Tue Dec 10 00:41:14 2013 LT: Initiating SCP file transfer of /var/tmp/plotraindays.png...
plotraindays.png                                                 100%   11KB  11.4KB/s   00:00
Tue Dec 10 00:41:15 2013 LT: Initiating SCP file transfer of /var/tmp/monthlyraindays.png...
monthlyraindays.png                                              100%   10KB  10.4KB/s   00:00
Tue Dec 10 00:41:15 2013 LT: Initiating SCP file transfer of /var/tmp/monthlyrain.png...
monthlyrain.png                                                  100%   11KB  11.1KB/s   00:00
Tue Dec 10 00:41:15 2013 LT: HTTP GET request to Weather Underground...
Tue Dec 10 00:41:16 2013 LT: Weather Underground account "ISTFOLDM6" updated.
Tue Dec 10 00:41:16 2013 LT: Wrote "/var/tmp/wxdata.txt" to local disk, filesize = 2290 bytes.
Tue Dec 10 00:41:16 2013 LT: Wrote "/var/tmp/wxdata.xml" to local disk, filesize = 2452 bytes.
Tue Dec 10 00:41:16 2013 LT: Initiating SCP file transfer of /var/tmp/wxdata.txt...
wxdata.txt                                                       100% 2290     2.2KB/s   00:00
Tue Dec 10 00:41:16 2013 LT: Initiating SCP file transfer of /var/tmp/wxdata.xml...
wxdata.xml                                                       100% 2452     2.4KB/s   00:00
Tue Dec 10 00:41:17 2013 LT: FLASH memory write count since program start: 1.
Tue Dec 10 00:41:17 2013 LT: Waiting...
```

Figure 34: Typical *WOSPi* terminal output.

[28]http://www.aprs.org

8 The *WOSPi* software

8.1 Free of charge for non-commercial use

The *WOSPi* software is available to those interested, for non-commercial use only. Drop me an email, detailing who you are, where you are and details of your intended use of the *WOSPi* software, including the intended URL to your weather site. You will then receive a download link via email. The source code will *not* be included. The main purpose of writing the *WOSPi* software was self-education and to create an alternative to the dated *WeatherLink* software, to be enjoyed by hobbyist weather observers. What you need to get *WOSPi* running will always be available *free of charge* — as long as you are prepared to put *a little* effort into making it work.

8.2 Credit where credit is due

As I have spent quite some time writing the software and the documentation, due credit should be given at any website using data provided by the *WOSPi* software. Please keep me informed of your weather site's URL, so that I can add it to the list of "other *WOSPi* sites" at `http://meteo.annoyingdesigns.com`.

8.3 Commercial use

Commercial use of the *WOSPi* software is available by prior agreement only.

8.4 Configuring the *WOSPi* software

Refer to the included `readme.txt` file which contains installation instructions.

8.5 Configuring the *WOSPi* software — `config.py`

The *WOSPi* software requires several changes to the default configuration file. Log in as the *wospi* user and edit the `config.py` configuration file. There are numerous comments in the `config.py` file, describing the available configuration options.

8.5.1 The *COMMISSIONDATE* setting

The `COMMISSIONDATE` setting in the `config.py` configuration file should be set to indicate *the 1st day of the first month of which rainfall data will be available.* The date format is strictly `dd.mm.yyyy`. Examples:

- Observations started on the 23rd of November, 2013 — set `COMMISSIONDATE` to 01.11.2013.
- Observations started on the 1st of December, 2013 — set `COMMISSIONDATE` to 01.12.2013.

The `COMMISSIONDATE` setting will influence the presentation of rainfall data in the relevant rainfall data plots.

9 Additional notes

9.1 Sunrise/sunset times on data plots

The sunrise/sunset times may not display correctly at extreme latitudes. The *gnuplot* script files can be edited to remove the sunrise/sunset times entirely. Section 9.22 refers.

9.2 Updating the VP2P/VP2/Vue console time

The Raspberry Pi comes without a real-time clock. As such, it relies on the NTP[29] service to set its software-based clock. Clock adjustment is performed automatically whenever an internet connection is available. It is crucial that the time zone settings are set correctly. The *WOSPi* software will attempt to adjust the console's date/time every 24 hours based on the present time in the Raspberry Pi's software clock.

Using the NTP service with a correctly configured Raspberry Pi, DST settings are applied automatically.

9.3 The *condensation* flag

Under given conditions, an unheated VP2P rain collector will indicate a light rainfall due to condensation on the rain gauge itself. The textual report of current observations will indicate this condition. The rainfall values in the *wxDict* dictionary will contain the originally recorded rainfall data.

The `wxDict['CONDENSATION']` value will be *True* in typical condensation conditions. The value is otherwise *False*. The *condensation* flag is updated at 10-minute intervals.

9.4 The *freeze* flag

An unheated VP2P rain collector will not indicate precipitation in freezing conditions. The textual report of current observations will indicate this condition. The rainfall values in the *wxDict* dictionary will contain the originally recorded rainfall data, should there be any.

The `wxDict['FREEZE']` value will be *True* in freezing conditions. The value is otherwise *False*. The *freeze* flag is updated at 10-minute intervals.

[29]Network Time Protocol

9.5 Correcting the UV radiation and solar radiation sensor readings

The `UVCF` (UV Correction Factor) and `SOLARCF` (Solar Correction Factor) settings in the `config.py` file can be used to adjust the *UV radiation* and *solar radiation* sensor readings. Values outside the [50, 150] range will be ignored, except for the value 0 which can be used to indicate that the sensor is not installed/available.

The VP2P ISS manual contains information related to annual adjustment requirements for the solar radiation and UV radiation sensors. The adjusted sensor readings will be stored in the *wxDict* dictionary and treated as reported sensor values.

Listing 11: *Python* source code example, showing how to adjust the solar radiation and UV radiation readings.

```
UVCF    = 100  # retain 100% of the value reported by the UV radiation sensor
SOLARCF = 100  # retain 100% of the value reported by the solar radiation sensor

UVCF    =  95  # retain 95% of the value reported by the UV radiation sensor
SOLARCF =  85  # retain 85% of the value reported by the solar radiation sensor

UVCF    =   0  # indicates UV radiation sensor not present
SOLARCF =   0  # indicates solar radiation sensor not present
```

If the UV or solar radiation sensor is not present, it may also be a good idea to remove the UV and solar radiation graphs by editing the *gnuplot* script file. Section 9.22 refers.

9.6 Console *RXCHECK* values

It seems that reported `RXCHECK` values do not fully match the description in the official console documentation. The values are supposed to represent, as counted from midnight:

- The total number of packets received by the console.
- Number of lost packets.
- Number of resynchronizations.
- The largest number of packets received in a row.
- Number of detected CRC errors.

It seems that the console randomly refuses to reply to the `RXCHECK` command. Also, when first issued, the `RXCHECK` command seems to reset the packet counters, restarting all counts from 0. The `RXCHECK` values, if received from the console, are included in the min/max report as described in section 7.6.

9.7 Importing the *wospi.pyc* library

For use with your own software projects, the `wospi.pyc` file can be imported into *Python* using the ordinary module import technique: `import wospi` will do the trick. Importing the library will not start any services. Please note that only one instance of the *WOSPi* software should be running at a time.

After importing the `wospi.pyc` file, use `dir(wospi)` to get an overview of available functions, etc. *Python* documentation strings have been applied throughout, providing some basic guidance for use with the `help` function: `help(wospi.wxWrite)` will display the documentation string for the *wxWrite* function. Running `help(wospi)` will also yield useful information.

9.8 Regularly running commands at specified times

The Linux distribution included with the Raspberry Pi fully supports *crontab*. The *WOSPi* software does not itself make use of *crontab*, except for the optional once/twice-a-day min/max temperature, max daily solar/UV radiation and weekly barometric pressure plotting functions described in sections 7.13, 7.14 and 7.16.

Simply edit the `/etc/crontab` file by running `sudo nano /etc/crontab`, adding the lines as shown in the excerpt below.

In this example, the `plotMinMaxTemp.py` program is run with the *wospi* user ID at 00:01 and 12:00 local time — and the `wxBackup.sh` script is run at 00:01 on the first day of each month. Similarly, the `plotSolar.py` program is run with the *wospi* user ID at 18:00 local time and the the `plotTempSolar.py` program is run with the *wospi* user ID at 18:03 local time. The `plotBaroWeek.py` program will run at 00:01 local time.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# NOTE: there are already a few lines in the existing crontab file -
#       you may want to "mute" the output from these lines by adding
#       >/dev/null 2>&1 after each line
#m h dom mon dow user    command
00 12 *  *   *   wospi   cd ~ && python plotMinMaxTemp.py >/dev/null 2>&1
00 18 *  *   *   wospi   cd ~ && python plotSolar.py >/dev/null 2>&1
03 18 *  *   *   wospi   cd ~ && python plotTempSolar.py >/dev/null 2>&1
01 00 *  *   *   wospi   cd ~ && python plotMinMaxTemp.py >/dev/null 2>&1
59 23 *  *   *   wospi   cd ~ && python plotBaroWeek.py >/dev/null 2>&1
01 00 1  *   *   wospi   ~/wxBackup.sh >/dev/null 2>&1
#
```

9.9 Choosing a web hotel

Any Linux-based web hotel supporting the SCP[30] protocol will do. Domeneshop AS[31] is highly recommended.

[30] Secure Copy

[31] `http://www.domainnameshop.com`

9.10 Raspberry Pi HDMI output

As the HDMI/video logic of the Raspberry Pi attempts to be half-intelligent, battling with equally half-intelligent life forms in attached HDMI display equipment, the result may be a display going into "power save" mode when no signal is detected at all. This is *very* likely to occur if you change from one HDMI display to another. The Raspberry Pi website contains several relevant articles describing the various HDMI settings which can be applied.

Editing the `/boot/config.txt` file, changing a few of the HDMI parameters, will probably serve as an adequate fix. Edit the file by running `sudo nano /boot/config.txt`, then make sure that the following lines are *not* commented out (edit/delete the `#` symbol occuring in the first character position on each of the following lines:

a) `hdmi_force_hotplug=1`

b) `hdmi_group=1`

c) `hdmi_mode=4`

Finally restart the Raspberry Pi, power off and then power on the attached display unit.

9.11 Password-protected websites — the *.htaccess* file

Only remotely related to the *WOSPi* software, but users have already asked how to password protect their website. Suggested reading: `http://www.domainnameshop.com/faq.cgi?id=63`.

9.12 Backing up weather observations by email — *wxBackup.sh*

You may want to backup your weather observations at regular intervals. Using *crontab* as shown in section 9.8, weather observations can be emailed to a suitable email account (e.g. *Gmail*) for safe storage. Edit the `/etc/ssmtp/ssmtp.conf` file, inserting the relevant parameters for your preferred email setup. This example makes use of the *Gmail* mail service:

```
AuthUser=YourUserID@gmail.com
AuthPass=YourPassword
FromLineOverride=YES
MailHub=smtp.gmail.com:587
UseSTARTTLS=YES
```

The `wxBackup.sh` shell script should be made *executable* by running the `chmod 700 wxBackup.sh` command. Also, the `.muttrc` configuration file residing in the home folder of the *wospi* user should be edited to force the *mutt* email client to *not* store copies of sent email messages. Edit the `.muttrc` file, inserting the following line:

```
set copy = no
```

9.13 The Raspberry Pi and SD or SDHC cards

The Raspberry Pi website contains several articles and forum posts regarding SD and SDHC cards which are known to work well with the Raspberry Pi. Although not officially confirmed, it seems that the "higher class" (class 6-10) cards have previously caused some problems. There is still a lot to be said about this topic — but SD cards should not be trusted as ever-lasting storage devices. Due to the nature of the storage technology used by SD/SDHC cards, each card will only accept a limited number of write operations. Remember to make backup copies of your SD cards at regular intervals.

There are several articles available, suggesting recommended practices for making the Raspberry Pi and SD/SDHC cards more robust for continous 24/7 operation.

9.14 SD card backup

The SD card should be backed up at regular intervals. A backup can be performed using a number of techniques, including `dd` and the somewhat more user-friendly *Partclone*[32] tool. Please note that it takes a while to perform a backup (or restore) operation, depending on system and SD card performance. This procedure may also be applied should multiple *WOSPi* installations be desired — properly configure one SD card, then duplicate it for use with the other installations.

Power down the Raspberry Pi with the `sudo halt` command, disconnect the power supply and carefully pull out the SD card. Insert the SD card into a Linux/MacOS X or Windows computer.

9.14.1 Using the *Partclone* tool

On Ubuntu systems, start by downloading *Partclone* by running `sudo apt-get install partclone`.

Proceed by installing the *Partclone* "wrapper" which can be found at `http://www.raspberrypi.org/phpBB3/download/file.php?id=442`. This "wrapper" will back up (and restore) the *boot* partition, the *system* partition and the MBR[33] of the SD card. In the below listing, `/dev/sdb` is assumed to be the mount point.

a) Insert the SD card.

b) `df -h` — provides a listing of mounted drives.

c) `dmesg` — shows kernel output messages, which may assist in locating the path to the newly inserted SD card.

d) `sudo ./rpi-backup -c -i /dev/sdb -o wospi`

e) `sudo umount /dev/sdb`

[32] `http://www.partclone.org`
[33] Master Boot Record

f) If a *restore* operation (writing the image to a new SD card) is required, insert the new SD card and proceed as follows (the operation will fail if there is insufficient storage capacity on the SD card):

g) `sudo ./rpi-restore -i wospi -o /dev/sdb`

h) `sudo umount /dev/sdb`

9.14.2 Using *dd*

Using `dd`, it is essential to know where the SD card is mounted. Insert the SD card into the card reader, then establish *where* the card is mounted. Below, `/dev/sdb` is assumed to be the mount point. It is very likely that it will be a different mount point on a different system.

a) `df -h` — provides a listing of mounted drives.

b) `dmesg` — kernel output messages may assist in locating the path to the newly inserted SD card.

c) `sudo dd if=/dev/sdb of=wospi.img`

d) `sudo umount /dev/sdb`

e) If a *restore* operation (writing the image to a new SD card) is required, insert the new SD card and proceed as follows:

f) `sudo dd of=/dev/sdb if=image.img bs=512k`

NOTE: on MacOS X systems, the SD card will typically be found at `/dev/disk1` (for use with the `dd` command) and `/dev/rdisk1` (for use when unmounting the SD card). Unmounting the SD card should be done using `sudo diskutil unmountDisk disk1` or the GUI-based *Disk Utility* application (in the *Applications/Utilities* folder.

NOTE: familiarise yourself with the damage potential of the `dd` command, as described in section 9.14.4.

9.14.3 Using the *Win32 Disk Imager*

Using the *Win32 Disk Imager* (for the Windows operating system) is self-explanatory. Download from `http://sourceforge.net/projects/win32diskimager/`.

9.14.4 Resizing the SD card image

For technical reasons, an 8 GB SD card isn't necessarily an 8 GB SD card. It may be very close to 8 GB, but the exact capacity will vary somewhat between vendors, models and individual SD cards. Problems will occur if you attempt to restore a backup copy from a "big" 8 GB SD card onto a "smaller" 8 GB SD card.

The easiest way to avoid this problem is by *not* using the *expand root partition to fill entire SD card* option in the *raspi-config* tool. If, however, the root partition has already been expanded to make use of the entire SD card, you can probably[34] get away with this trick (now is the right time to read the NOTE following this listing):

a) Insert the *source* SD card.

b) Start *gparted* by running `gksudo gparted &`

c) Select the `/dev/sdb` device.

d) Right-click on the `/dev/sdb2` (*ext4*) partition, then select *Unmount* from the pop-up menu.

e) Right-click on the `/dev/sdb2` (*ext4*) partition, then select *Resize/Move* from the pop-up menu.

f) Reduce the partition size by a few MB (200 MB should do the trick) and apply this change before quitting *gparted*.

g) Create an image of the modified SD card by running:
`sudo dd if=/dev/sdb of=wospi.img`

h) Unmount `/dev/sdb1` and `/dev/sdb2` by running `sudo umount /dev/sdb1` and `sudo umount /dev/sdb2`

i) Eject the *source* SD card and insert the *target* SD card.

j) Write the image to the *target* SD card by running:
`sudo dd if=wospi.img of=/dev/sdb bs=512k`

k) When complete, `dd` will probably exit with an error message stating that there is not enough room on the destination disk. This error message can be ignored.

l) Unmount `/dev/sdb1` and `/dev/sdb2` by running:
`sudo umount /dev/sdb` and `sudo umount /dev/sdb`

NOTE: in the above listing, `/dev/sdb` is assumed to be the location of the SD card. This may (and probably will) differ on your system. Please note that the `dd` command *will* do a lot of damage to the contents of your disk(s) if used with the wrong `of=` argument. *You will get no warning* if you specify the wrong output destination. The entire write operation will take a few minutes, and there is no feedback during the process. Be patient.

[34]Provided that the SD card isn't yet filled to capacity with actual data. Still, there is a slight chance of data loss when using this procedure.

9.15 Backing up your files via FTP

Apart from copying the entire SD card, you can use FTP to quickly move files between the Raspberry Pi and other units. Simply start the FTP server on the Raspberry Pi using the *Python*-based FTP server described in section 4.8 and use standard FTP commands to transfer files between the Raspberry Pi and your Linux/Windows/MaxOS X/Linux/... machine.

For simplicity, consider compressing the entire *wospi* home folder into one single archive file: running `zip wospi.zip -r *` will result in one single file, `wospi.zip`, which can easily be transferred via FTP. Utilities `gzip` and `bzip2` may also be used for the same purpose.

9.16 Sentinel values, out-of-range values

The official *Davis* documentation for the serial communication protocol now describes some of the sentinel values used to indicate ”sensor data not available”. Up until March, 2013, this information was not officially available. It seems that extreme-high values (such as 127 or 255 for a signed/unsigned *byte*-sized variable and 32767 or 65535 for a signed/unsigned *word*-sized variable) indicate ”sensor data not availabe”. The *WOSPi* software does not provide filtering of all sentinel values. As such, checking that sensor readings remain within reasonable ranges remains the responsibility of the programmer. Indications of out-of-range values can easily be spotted on the 24-hour data plots.

9.17 Error messages (*IOError*, *permission denied*)

IOError or *Permission denied* error messages normally result from the *wospi* user not having the adequate permissions for writing to a file or a folder. The situation is typical for the `/media/sd` folder (external USB storage device) if it is mounted without giving the *wospi* user the required write access privileges. Section 4.10 refers.

9.18 The *getBeaufort* function

The *wospi.pyc* library contains function `getBeaufort(windSpeedKTS)` which will return a textual representation of the wind speed, given *windSpeed* in knots. The textual description can be changed by redefining the 13-element dictionary `beaufortText` in the *config.py* configuration file. The default configuration file contains an example. Default values are listed in table 1.

Table 1: Beaufort scale wind descriptions as returned by the `getBeaufort` function.

Wind speed (knots)	Beaufort scale description	`beaufortText` dictionary key
≤ 1	Calm	0
$[2, 3]$	Light air	1
$[4, 6]$	Light breeze	2
$[7, 10]$	Gentle breeze	3
$[11, 16]$	Moderate breeze	4
$[17, 21]$	Fresh breeze	5
$[22, 27]$	Strong breeze	6
$[28, 33]$	Near gale	7
$[34, 40]$	Gale	8
$[41, 47]$	Strong gale	9
$[48, 55]$	Storm	10
$[56, 63]$	Violent storm	11
≥ 64	Hurricane	12

9.19 Undocumented console commands

The VP2P console supports a number of commands which are not listed in the official documentation. Please be extremely careful if you start playing around with these commands.

While you are not likely to *break*[35] anything, you are definitely in for an exciting evening if you start playing around with stuff while in "test mode" (`TST 1`). Most people have better things to do, unless they happen to stumble across console documentation which is intended for use by the manufacturer only (in that case, please forward a copy). A number of the undocumented console commands are listed below:

- `ID`
- `BOOT`
- `RXTEST`
- `NEWX`
- `RX`
- `TX`
- `ADREAD`
- `CLKON`
- `CLKOFF`
- `BVER`
- `POW`
- `PORT`
- `LCD`
- `TST`
- `XTLCAL`
- `DUMPREG`
- `PLLCAL`
- `DOMAIN`
- `CHAN`
- `BAND`
- `BUZZER`

[35]But it is very likely that your console will suffer from severe schizophrenia afterwards.

9.20 Improving the cooling of the *Pi Holder* case

Another step which really isn't *required* — but it doesn't take more than a minute or two to add a heatsink to the *Pi Holder* case, effectively decreasing the exterior case temperature by some 4–5°C. Figures 35 and 36 refer.

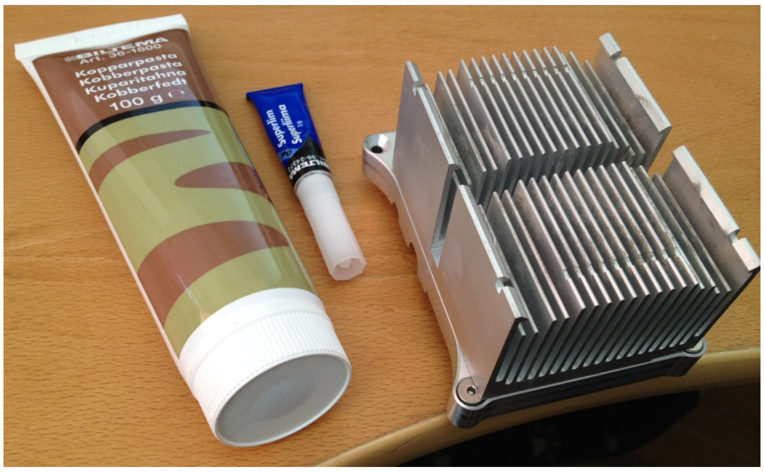


Figure 35: Suggested parts. Cooling ribs from a long-gone Intel Pentium processor, copper paste and super glue.

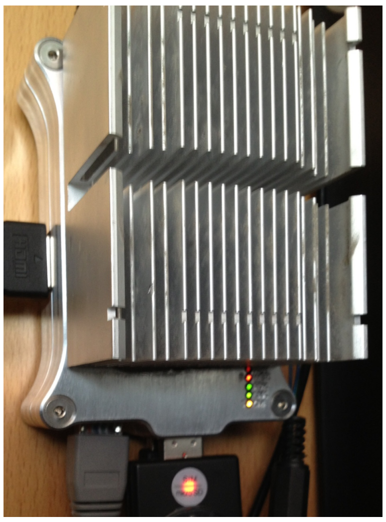


Figure 36: Now running even cooler than before. Not better-looking than before, but functionality *does* matter.

9.21 VP2P and ISS batteries

The VP2P console runs on 5V DC from an external power supply capable of delivering 0.3A (positive center connector) and has a backup power supply consisting of 3 x 1.5V LR14 batteries. The console batteries should be installed and changed at regular intervals, to aid in data collection whenever a power outage occurs.

ISS backup power is supplied by a 3V lithium battery, type CR17345[36]. It appears to be a good idea to replace this battery in late autumn, so that it will last until the next spring. This observation is based on experience gained in southern Norway, where daylight during wintertime is so limited that the ISS capacitator will not get a full charge from the solar panel. Thus, the backup battery will be kicking in to supply power in the early morning hours. A depleted battery will trigger the battery replacement message on the console and temporarily inhibit transmission of the solar and UV sensor readings.

9.22 *gnuplot* script files

The *gnuplot* script files used to generate the data plots can be found (and edited) in the home folder of the *wospi* user. The files are named `plotRainPerMonth.input`, `plotRainDaysPerMonth.input`, `plotRainMonth.input`, `plot24.input`, `plotMinMaxTemp.input`, `plotSolar.input`, `plotBaroWeek.input`, `plotTempSolar.input`, `plot24windL2.input` as well as `plot24windL1.input` — the latter for use with consoles not supporting the `LPS` command. If your station is not equipped with solar and/or UV sensors, simply adjust the `plot24.input` file accordingly to avoid meaningless solar/UV radiation plots.

Embedded comments/notes in the *gnuplot* `*.input` files describe various configuration settings, including switching between units such as °C and °F, hPa/mb and inHg as well as mm and in.

Wind speed can be plotted either in knots & m/sec. or knots & MPH units. Embedded comments in the `plot24windL1.input` and `plot24windL2.input` files describe this configuration setting.

If you change the location of the temporary files used by *WOSPi*, as specified in the `config.py` configuration file, the *gnuplot* script files have to be updated accordingly (not recommended, though).

[36] Also known as CR123, DL123A, EL CR 123AP, CR123A, K123LA and CR123R.

9.23 External data storage and presentation: *ThingSpeak.com*, *Highcharts.com*

Weather data can be forwarded to online data storage services such as *ThingSpeak*[37] by means of the following methods:

- The `config.py` configuration file contains provisions for two user-defined functions which are invoked at defined intervals. Numerous examples of *Python*-based data uploads can easily be found online.
- *crontab*-invoked scripts reading the *WOSPi* data file(s), uploading relevant data fields.

Presentation of weather data can easily be achieved using the *Highcharts*[38] service. The *Highcharts* service can retrieve data from storage services such as *ThingSpeak*. Several other options are also available.

[37]http://www.thingspeak.com
[38]http://www.highcharts.com

10 Resources and references

10.1 Thanks to ...

The *WOSPi* software wouldn't have seen daylight without the inspiring works and posts by *DeKay*, published on his `http://madscientistlabs.blogspot.no` blog. Thank you for sharing and thank you for allowing me to use your schematics in this document (figures 10 and 13, respectively).

Davis Instruments Corp. did the right thing when they made the *Serial Communication Reference Manual* available for download. Unfortunately, they messed it all up by shipping near-useless consoles running FW versions 3.00 and 3.12 in late 2012 (...and v. 3.15 in 2013), hiding behind a curtain of silly excuses. Sorry, but none of your smokescreens are quite capable of disguising the word *greed*. You still have a chance to get it right, though. And while you're at it, please update the *Serial Communication Reference Manual* to also include descriptions of *all* the commands supported by the VP2P console and the ISS unit. For those who cannot wait for *Davis* to correct the "FW version 3 mistake", this document will probably be of interest: `http://meteo.annoyingdesigns.com/DavisSPI.pdf`.

The forums at `http://www.wxforum.net` represent an indispensable source of information. Thanks to everyone in there who shares their knowledge.

10.2 Other references

Documentation for the *Python* programming language can be found at `http://www.python.org`.

For documentation projects exceeding one page of text, LaTeX remains the king of document preparation systems. Please visit `http://www.latex-project.org` for further details.

A *gnuplot* reference and relevant documentation can be found at `http://www.gnuplot.info`.

The *Raspberry Pi Foundation's* website contains everything you'll ever want to know about the Raspberry Pi: `http://www.raspberrypi.org`.

11 Contact information

Contact information, web address, Google Groups discussion forum, et cetera:

Torkel M. Jodalen
Glassv. 71
NO-1515 Moss
Norway

Email → `tmj@bitwrap.no`

Web → `http://meteo.annoyingdesigns.com`

Google Groups → `https://groups.google.com/group/wospi?hl=en`
(English language only, please).

Please use the Google Groups online discussion forum for any questions related to the *WOSPi* software. Free-of-charge end-user support is *not* available via email but the online discussion forum is a good place to ask for assistance.

WOSPi is given away for free to non-commercial users. Commercial use is allowed by prior agreement only. Website owners who utilize *WOSPi* to publish live weather data are requested to report their website URL to the author at the above email address. Thank you.

11.1 Comments, suggestions, feature requests, etc.

...are welcome. But please keep in mind that the *WOSPi* software was created primarily as a self-education project and for having a bit of fun with the Raspberry Pi, the VP2P and LaTeX.

Always have the appropriate amount of fun.