

# vcontrol mit Raspberry Pi

Klaus.R edited this page on 27 May 2017 · 34 revisions

[Edit](#)
[New Page](#)

## vcontrol mit Raspberry Pi

► Pages 72

Der Raspberry Pi eignet sich nicht nur als stromsparendes Embedded System zum Erfassen der Heizungsdaten, sondern bietet auch die Möglichkeit der grafischen Darstellung und Auswertung.

In der Heizung ist bei mir immer noch ein alter Windows XP PC im Einsatz. Nicht nur dass dieser unnötig viel Energie verbraucht: im Frühling 2014 ist auch das Aus von Windows XP gegeben. Weil es die Synology USB Station 2 nicht bis in die Heizung geschafft hat und ich schon seit Sommer 2012 zwei Raspi's habe (Rev. 1 mit 256MB RAM), soll jetzt ein Raspi für vcontrol eingerichtet werden. Ich hatte das schon vor langem einmal durchgespielt, doch nun soll alles von Anbeginn dokumentiert werden und auch bisherige Diskussionen aus dem Forum berücksichtigt werden.

**Voraussetzungen Hardware:**

- Raspberry Pi (v1, v2 oder v3)
- ggf. Tastatur und Bildschirm
- Netzwerkverbindung oder WLAN Dongle
- Netzteil (wichtig: stabile Stromversorgung, min 1A)
- 4GB SD Karte (oder grösser)
- Optolink Adapter mit USB (Optolink über GPIO wäre auch möglich, wird hier aber nicht weiter verfolgt)
- Vitotronic-Steuerung, welche von vcontrol unterstützt wird (hier mit Vitotronic 200 KW2 verifiziert) Software:
- Raspbian (Für Raspberry Pi angepasstes Debian 7 aka Wheezy)
- SW zum Bilden von vcontrol
- Weitere SW nach Bedarf (rrd, munin, ...)

Solange es kein fertiges vcontrol image zur Installation gibt, ist es am einfachsten, vcontrol auf dem Raspberry Pi selber zu bilden. *vcontrol ist so schlank, dass sich ein Cross-Compiling auf einem PC nicht lohnt.*

## Schritt für Schritt Anleitung

===1. Inbetriebnahme Raspberry Pi=== Ich verwende hier Raspbian (Debian Wheezy for Raspberry) und zwar direkt das Raspbian Image. Bei Installation von Raspbian via NOOBS reicht eine 4GB Speicherkarte nicht mehr (Dez. 2013). Aktuelles Raspbian Image herunterladen und mit Win32DiskImager (Windows) oder dd (Linux) auf die SD Karte übertragen Vor dem Entfernen der Speicherkarte evtl. /boot/config.txt anpassen z.B.

```
disable_overscan=1 (vermeidet schwarzen Rand)
hdmi_force_hotplug=1 (HDMI auch dann aktiv, wenn beim booten noch nicht angeschlossen)
```

Ich mache auch immer einen Kommentar in die config.txt Datei und setze ein Disk-Label, damit ich die SD-Karten (es liegen einige solche herum) mit jedem PC identifizieren kann.

Nach dem ersten Start mit der neu aufgesetzten SD Karte die übliche Einstellungen vornehmen mit „sudo raspi-config“:

- Passwort für user pi ändern
- Tastaturlayout und Zeitzone anpassen
- Filesystem expandieren (Neustart nötig)

### Einführung

#### Optolink Schnittstelle

- [Adapter Eigenbau](#)
- [Bauanleitung RS232](#)
- [Bauanleitung USB](#)
- [LAN/USB Kombiadapter](#)
- [Bauanleitung LAN-Ethernet](#)
- [Bauanleitung 3.3V TTL](#)
- [Bauanleitung LEGO™](#)
- [Bauanleitung Raspberry Pi](#)
- [Bauanleitung CAN](#)
- [Bauanleitung ESP8266](#)
- [Bauanleitung OptoPi](#)
- [openHab Integration](#)
- [Bauanleitung Hovilink](#)
- [Bauanleitung ESP32](#)

#### Protokolle

- [GWG](#)
- [KW](#)
- [300](#)



#### Geräte

- [Adressen](#)
- [Weitere Adressen](#)
- [Vito-Masterdateien](#)

#### Linux Software

- [vcontrol](#)
  - [Konfig vcontrol](#)
  - [vclient](#)
  - [Auswertung mit RRDB](#)
  - [XML-Konfiguration](#)
  - [Hinweise XML Konfig](#)
  - [Vcontrol Kompilieren](#)
- [vconnect](#) 
- [ViTalk](#) 

#### Microcontroller

- [VitoWifi ESP8266/ESP32](#) 
- [esp-link](#) 

#### Windows Software

- [v-control](#)
- [v-commDLL](#)
- [IpSymcon Interface](#)
- [RS232 Test / VitoTest](#)
- [voIdent](#)
- [VitoGraph](#)
- [Viess-Data, Viess-Data 2.0](#)
- [Vies-sion](#)
- [Windows "Daemon"](#)

Beim Neustart wird das Filesystem erweitert, so dass die gesamte Größe der SD Karte genutzt wird. Ein Neustart kann auf der Kommandozeile jederzeit mit "sudo shutdown -r now" gemacht werden. Vor dem Ausschalten muss mit "sudo shutdown -h now" das System sauber heruntergefahren werden. Sonst kann es passieren, dass das System nicht mehr hochfährt und auch die Daten nicht wiederhergestellt werden können (Einmaliges Backup des neuen Image und regelmässiges Backup der Daten sind wichtig).

Nach dem Neustart System aktualisieren mit:

```
sudo apt-get update
sudo apt-get upgrade
```

Falls Pakete als "not upgraded" aufgezählt werden, können diese mit sudo apt-get dist-upgrade aktualisiert werden.

Weitere Tipps:

- Um den Raspi mit SSH (PuTTY) zu administrieren oder generell um ihn im Netz zu finden empfiehlt sich eine fixe IP Adresse zu vergeben. Am besten wird im DHCP Server (Router) eine feste Adresse eingestellt (die MAC Adresse, wenn nicht im Router ersichtlich) kann im Raspi mit ifconfig auf der Kommandozeile ermittelt werden (HWaddr).
- Für den Datei-Zugriff vom normalen Rechner (z.B. log-Dateien, xml-Dateien usw.) empfiehlt sich die Installation des Samba-Servers. Dann kann man den Raspi z.B. als Netzwerklaufwerk am Windows-Rechner anmelden.
- Unter Windows kann mit dem Win32DiskImager auch ein Image von einer SD Karte erstellt werden.

## 2. Zusätzliche Software installieren

Bevor folgende Pakete installiert werden, kann auch mit dpkg -l | grep in den bereits installierten Paketen nach gesucht werden.

- subversion
- automake
- autoconf
- build-essential (schon installiert, Dez. 2013)
- libxml2-dev
- telnet

```
sudo apt-get install subversion automake autoconf telnet libxml2-dev
```

## 3. vcontrold erstellen

Als user pi einloggen Ein Unterverzeichnis erstellen z.B. /home/pi/openv und in dieses Verzeichnis wechseln:

```
mkdir openv
cd openv
```

Source mit svn holen:

```
svn checkout svn://svn.code.sf.net/p/vcontrold/code/trunk vcontrold-code
```

Ins Verzeichnis wechseln, build mit folgenden Commands durchführen:

- [vcontrold "lite"](#)
- [vcontrold & cygwin](#)

### Embedded Linux Systeme


- [Raspberry Pi](#)
- [Synology USB Station 2](#)
- [FriendlyARM mini2440](#)
- [vcontrold auf OpenWrt](#)

### KM-Bus

- [KM-Bus cmd 0xBF\(u.c.\)](#)
- [KM-Bus Interface](#)
- [AVR-LAN-Interface](#)
- [Anlagen Galerie](#)
- [Motivation](#)
- [Links](#)

### Clone this wiki locally

<https://github.com/openv/op> 

 Clone in Desktop

```
cd vcontrold-code/vcontrold
chmod +x auto-build.sh
./auto-build.sh
./configure
make
sudo make install
```

Wenn alles erfolgreich durchgeführt wurde, dann sind die ausführbaren Dateien vcontrold, vclient und vsim erstellt worden. vcontrold und vclient sind in /usr/local/bin kopiert worden und deshalb von überall her ausführbar

## 4. Schnittstelle prüfen

Den Optolink Adapter anschließen und mit lsusb prüfen, ob er erkannt worden ist, z.B.:

```
cd ~/openv/vcontrold-code/vcontrold

lsusb

Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 007: ID 046d:c31d Logitech, Inc.
Bus 001 Device 008: ID 045e:00cb Microsoft Corp. Basic Optical Mouse v2.0
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB
Bus 001 Device 005: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-
Serial (UART) IC
Bus 001 Device 006: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB
```

Mit dmesg die Schnittstelle ermitteln z.B.:

```
pi@raspberrypi ~/openv/vcontrold-code/vcontrold $ dmesg | grep "now attached"
usb 1-1.3.1: FTDI USB Serial Device converter now attached to ttyUSB0
```

===4.1 Schnittstellenbenennung (optional)=== Sollten mehrere Adapter am PI hängen (also beispielsweise auch eine oder zwei zum Auslesen des Stromzählers), werden diese beim Reboot nicht zwangsläufig in der gleichen Reihenfolge durchnummeriert. Hierfür ist es wichtig dem Adapter einen eindeutigen Namen zuzuweisen:

Ausfindig machen der ID\_SERIAL\_SHORT

```
/sbin/udevadm info --query=all --name=/dev/ttyUSB0
[...]
P: /devices/platform/bcm2708_usb/usb1/1-1/1-1.3/1-1.3.3/1-1.3.3:1.0/ttyUSB0/tty/ttyUSB0
E: ID_SERIAL_SHORT=A701YM8E
E: ID_VENDOR_FROM_DATABASE=Future Technology Devices International, Ltd
[...]
```

Nun erweitert man (oder legt die Datei neu an) /etc/udev/rules.d/70-lesekopf.rules

```
SUBSYSTEM=="tty", ATTRS{product}=="FT232R USB UART", ATTRS{serial}=="A701YM8E",
NAME="vitoir0"
```

### 4.1.1 Alternative um an eine funktionierende udev-rule zu kommen

Annahme: Der Adapter wurde in Schritt4 als ttyUSB0 erkannt.

```
sudo udevadm info --name=/dev/ttyUSB0 --attribute-walk
```

Es werden verschiedene "parent devices aufgelistet". Den Filter der neuen udev-rule passt man nun so an, dass er in einem Block "parent device" zieht. Am Besten dort, wo ATTRS{product}=="FT232R USB UART" und ATTRS{serial}=="AIXYZXYZ" **gemeinsam** vorkommen. Achtung auch auf den Unterschied SUBSYSTEM vs SUBSYSTEMS.

Nun erweitert man (oder legt die Datei neu an) /etc/udev/rules.d/70-lesekopf.rules

```
SUBSYSTEMS=="usb", ATTRS{serial}=="AIXYZXYZ", SYMLINK+="vitoir0"
```

Sauberer ist es auch, mit dem Attribut SYMLINK zu arbeiten, anstatt NAME.

Nach einem Restart des udev

```
sudo service udev restart
```

bzw spätestens nach einem

```
sudo udevadm trigger
```

ergibt ein ls auf den Adapter ergibt dann:

```
ls -l /dev/serial/{by-path,by-id}/*
lrwxrwxrwx 1 root root 15 Jan  1  1970 /dev/serial/by-id/usb-
FTDI_FT232R_USB_UART_A701YM8E-if00-port0 -> ../../vitoir0
lrwxrwxrwx 1 root root 15 Jan  1  1970 /dev/serial/by-path/platform-bcm2708_usb-usb-
0:1.3.3:1.0-port0 -> ../../vitoir0
```

bzw. wenn das Attribut NAME in der Rule verwendet wurde:

```
ls -l /dev/vitoir0
lrwxrwxrwx 1 root root 15 Jan  1  1970 /dev/vitoir0 -> ttyUSB0
```

Nun kann der Adapter in der vcontrold.xml immer über /dev/vitoir0 angesprochen werden, und genau der Adapter mit der gefilterten Serien-Nummer wird in Zukunft immer als /dev/vitoir0 erreichbar sein, egal ob er initial als ttyUSB0 oder ttyUSB1 eingeordnet wurde.

## 5. vcontrold konfigurieren

Konfiguration der Schnittstelle in ~/openv/vcontrold-code/xml-32/xml/vcontrold.xml unter eintragen (hier für /dev/ttyUSB0 oder /dev/vitoir0, falls du Schritt 4.1 ausgeführt hast):

```
<config>
  <serial>
    <tty>/dev/ttyUSB0</tty>
  </serial>
  <net>
  ...
```

Die beiden XML-Konfigurationsdateien von ~/openv/vcontrold-code/xml-32/xml ins Verzeichnis /etc/vcontrold kopieren:

```
sudo mkdir /etc/vcontrold
sudo cp ~/openv/vcontrold-code/xml-32/xml/vito.xml /etc/vcontrold/
sudo cp ~/openv/vcontrold-code/xml-32/xml/vcontrold.xml /etc/vcontrold/
```

Wenn diese Dateien auf einem anderen Rechner aktualisiert werden, dann müssen diese auf den Raspberry Pi kopiert werden. Dazu gibt es unterschiedliche Verfahren:

- Zugriff auf den Raspi mittels ftp, scp (WinSCP) oder ähnliches
- vom PC auf die FAT-Partition (boot) der SD Karte kopieren (dazu muss aber jedesmal der Raspi heruntergefahren werden)

Vollständige Konfiguration vcontrold.xml unter :

```
<config>
  <serial>
    <tty>/dev/ttyUSB0</tty>
  </serial>
  <net>
    <port>3002</port>
    <allow ip='127.0.0.1' />
    <allow ip='192.168.0.0/24' />
  </net>
  <device ID="2098" />
</config>
```

Wichtig ist vor allem, dass die richtige Steuerung eingetragen ist. (hier ID="2098" für die V200 KW2). Die zweite „allow ip“ Einstellung muss dem eigenen Netz angepasst werden. Das ist nur nötig, wenn das telnet-Interface von anderen Rechnern aus genutzt werden soll. Es kann auch ein einziger externer PC konfiguriert werden (mit /24 wird im Beispiel allen Rechnern im 192.168.0.\* Netz Zugang gewährt).

## 6. Service starten (inkl. automatischer Start)

Zuerst sollte der vcontrold mit dem Parameter -n (no fork) aufgerufen werden, um zu verifizieren, dass er korrekt starten kann. vcontrold kann danach mit Ctrl-C abgebrochen werden und bleibt nicht im Speicher. Startup script "/etc/init.d/vcontrol" erstellen Bsp:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          vcontrold
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Initscript to start vcontrold daemon
# Description:       This file should be used to construct scripts to be
#                    placed in /etc/init.d.
### END INIT INFO

# Author:           Michael Pucher <tech@michaelpucher.net>
# Homepage:         openv.wikispaces.org
#
# Please remove the "Author" lines above and replace them
# with your own name if you copy and modify this script.

# Do NOT "set -e"

# PATH should only include /usr/* if it runs after the mountnfs.sh script
PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="vcontrold daemon"
NAME=vcontrold
DAEMON=/usr/local/sbin/$NAME
#DAEMON_ARGS="--options args"
```

```

DAEMON_ARGS=""
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

# Exit if the package is not installed
[ -x "$DAEMON" ] || exit 0

# Read configuration variable file if it is present
[ -r /etc/default/$NAME ] && . /etc/default/$NAME

# Load the VERBOSE setting and other rcS variables
. /lib/init/vars.sh

# Define LSB log_* functions.
# Depend on lsb-base (>= 3.2-14) to ensure that this file is present
# and status_of_proc is working.
. /lib/lsb/init-functions

#
# Function that starts the daemon/service
#
do_start()
{
    # Return
    # 0 if daemon has been started
    # 1 if daemon was already running
    # 2 if daemon could not be started
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON --test > /dev/null \
        || return 1
    start-stop-daemon --start --quiet --pidfile $PIDFILE --exec $DAEMON $DAEMON_ARGS \
        # Add code here, if necessary, that waits for the process to be ready
        # to handle requests from services started subsequently which depend
        # on this one. As a last resort, sleep for some time.
}

#
# Function that stops the daemon/service
#
do_stop()
{
    # Return
    # 0 if daemon has been stopped
    # 1 if daemon was already stopped
    # 2 if daemon could not be stopped
    # other if a failure occurred
    start-stop-daemon --stop --quiet --retry=TERM/30/KILL/5 --pidfile $PIDFILE --name $NAME
    RETVAL="$?"
    [ "$RETVAL" = 2 ] && return 2
    # Wait for children to finish too if this is a daemon that forks
    # and if the daemon is only ever run from this initscript.
    # If the above conditions are not satisfied then add some other code
    # that waits for the process to drop all resources that could be
    # needed by services started subsequently. A last resort is to
    # sleep for some time.
    start-stop-daemon --stop --quiet --oknodo --retry=0/30/KILL/5 --exec $DAEMON
    [ "$?" = 2 ] && return 2
    # Many daemons don't delete their pidfiles when they exit.
    rm -f $PIDFILE
    return "$RETVAL"
}

#
# Function that sends a SIGHUP to the daemon/service
#
do_reload() {
    #
    # If the daemon can reload its configuration without
    # restarting (for example, when it is sent a SIGHUP),
    # then implement that here.
    #
    start-stop-daemon --stop --signal 1 --quiet --pidfile $PIDFILE --name $NAME
    return 0
}

case "$1" in
start)
    [ "$VERBOSE" != no ] && log_daemon_msg "Starting $DESC" "$NAME"
    do_start

```

```

case "$?" in
  0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
  2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
esac
;;
stop)
[ "$VERBOSE" != no ] && log_daemon_msg "Stopping $DESC" "$NAME"
do_stop
case "$?" in
  0|1) [ "$VERBOSE" != no ] && log_end_msg 0 ;;
  2) [ "$VERBOSE" != no ] && log_end_msg 1 ;;
esac
;;
status)
status_of_proc "$DAEMON" "$NAME" && exit 0 || exit $?
;;
#reload|force-reload)
#
# If do_reload() is not implemented then leave this commented out
# and leave 'force-reload' as an alias for 'restart'.
#
#log_daemon_msg "Reloading $DESC" "$NAME"
#do_reload
#log_end_msg $?
#;;
restart|force-reload)
#
# If the "reload" option is implemented then remove the
# 'force-reload' alias
#
log_daemon_msg "Restarting $DESC" "$NAME"
do_stop
case "$?" in
  0|1)
do_start
case "$?" in
  0) log_end_msg 0 ;;
  1) log_end_msg 1 ;; # Old process is still running
  *) log_end_msg 1 ;; # Failed to start
esac
;;
*)
# Failed to stop
log_end_msg 1
;;
esac
;;
*)
#echo "Usage: $SCRIPTNAME {start|stop|restart|reload|force-reload}" >&2
echo "Usage: $SCRIPTNAME {start|stop|status|restart|force-reload}" >&2
exit 3
;;
esac
:

```

Startup script registrieren:

```

cd /etc/init.d/
sudo chmod u+x vcontrol
sudo update-rc.d vcontrol start 99 2 3 4 5 . stop 99 0 1 6 .

```

## 7. Test

Wenn die obigen Schritte erfolgreich durchgeführt wurden, dann ist nach Aufruf von "sudo /etc/init.d/vcontrol start" der vcontrold-Server bereit.

Am besten wird das System einmal neu gestartet, damit man sicher ist, dass alles funktioniert. Es empfiehlt sich, alle Dateien, welche man für das System angepasst hat auf dem Arbeits-PC zu sichern. Die Einstellungen werden verloren gehen, wenn einmal die SD Karte nicht mehr lesbar ist, was z.B. durch ein unkontrolliertes Ausschalten des Raspi passieren kann.

Am einfachsten ist es, die Telnet Schnittstelle mit PuTTY zu testen.

PuTTY Telnet-Session Einstellungen:

- Connection type = Telnet
- Host = ip Adresse des Raspberry Pi
- Port = 3002
- Bei den Terminal Einstellungen sollte noch die Option „Implicit CR in every LF“ gesetzt werden, damit die Zeilenenden richtig interpretiert werden
- Bei den Einstellungen unter „Connection – Telnet“ den Negotiation Mode = Passiv setzen

(andernfalls werden negotiation commands an vcontrold geschickt, welche von diesem nicht verarbeitet werden. Das führt bei der ersten Eingabe immer zu einem Command not found Fehler).

Man kann auch den SSH Zugang verwenden und in der Shell des Raspis das Command „telnet localhost 3002“ aufrufen.

Wenn der Kommando-Prompt vctrl> erscheint, dann hat alles geklappt!

Durch Eingabe von help werden die gültigen Befehle aufgelistet.

version zeigt die vcontrold Version an. device den in vcontrold.xml konfigurierten device.

commands zeigt alle Befehle für die entsprechende Steuerung an.

Wenn nun getTempA oder getDevType funktioniert, dann ist auch die serielle Kommunikation in Ordnung.

Gratulation, du hast es geschafft!

user:vitoopen

## 8. Auswertung mittels RRDB

===8.1. RRDB anlegen=== Zuerst einmal muss das RRDB-Paket runter geladen werden.

```
sudo apt-get install rrdtool
```

Nun muss eine Datenbank angelegt werden. Für eine Datenbank, in der man einen Monat alle 2 Minuten 16 Werte erfassen möchte würde das folgendermaßen aussehen:

```
rrdtool create vito.rrd --step 120 \
DS:TempA:GAUGE:300:-50:100 \
DS:TempLuftVL:GAUGE:300:-50:100 \
DS:TempLuftRL:GAUGE:300:-50:100 \
DS:TempSekVL:GAUGE:300:0:100 \
DS:TempSekRL:GAUGE:300:0:100 \
DS:TempVerdampfer:GAUGE:300:-50:100 \
DS:DruckVerdampfer:GAUGE:300:0:100 \
DS:DruckKondensator:GAUGE:300:0:100 \
DS:tempWist:GAUGE:300:0:100 \
DS:TempSTSlist:GAUGE:300:0:100 \
DS:TempM2VList:GAUGE:300:0:100 \
DS:Speicherladepumpe:GAUGE:300:0:1 \
DS:Verdichter:GAUGE:300:0:1 \
DS:Zirkulation:GAUGE:300:0:1 \
DS:DLE1:GAUGE:300:0:1 \
DS:DLE2:GAUGE:300:0:1 \
RRA:AVERAGE:0.5:2:22500 \
RRA:MIN:0.5:12:4000 \
RRA:MAX:0.5:12:4000 \
RRA:AVERAGE:0.5:15:12000
```



Zur Erklärung Über DS werden die Werte definiert Über RRA werden dann die Datenbanken definiert

```
RRA:AVERAGE:0.5:1:22500 \
RRA:MIN:0.5:12:4000 \
RRA:MAX:0.5:12:4000 \
RRA:AVERAGE:0.5:15:12000
```

heißt es werden in derselben DB gleich vier "Round Robin Archive" verwaltet. Im ersten wird pro Intervall (--step 120) ein Wert gespeichert

Bei MIN und MAX wird alle 12 Minuten ein min bzw. max Wert gespeichert. Diese beiden Archive sind 2400 Werte groß. 2400 12 Minuten = 20 Tage Und bei AVERAGE alle 15 Minuten, 12000 Werte = 125 Tage.

## 8.2. Das Schreiben in die RRDB

geschieht dann über die TPL-Datei, in der via cron gesteuert alle 2 Minuten die 16 Werte eingetragen werden

```
update vito.rrd N:$1:$2:$3:$4:$5:$6:$7:$8:$9:$10:$11:$12:$13:$14:$15:$16
```

===8.3. Die Auswertung=== geschieht mittels nachstehendem Aufruf - Erklärungen folgen - das Ergebnis sieht so aus <http://warmup.mypump.de/Live-Daten/dailyview/g4.png>

```
rrdtool graph "graph.png" -a PNG \
--start "now-24h" --end "now" --width 1400 --height 600 \
--title "Vito-Messwerte - `date`" \
--slope-mode \
DEF:TempA=vito.rrd:TempA:AVERAGE \
DEF:TempSekVL= vito.rrd:TempSekVL:AVERAGE \
DEF:TempSekRL= vito.rrd:TempSekRL:AVERAGE \
DEF:tempWwist= vito.rrd:tempWwist:AVERAGE \
DEF:DruckVerdampfer= vito.rrd:DruckVerdampfer:AVERAGE \
DEF:DruckKondensator= vito.rrd:DruckKondensator:AVERAGE \
DEF:TempVerdampfer= vito.rrd:TempVerdampfer:AVERAGE \
DEF:TempM2VList= vito.rrd:TempM2VList:AVERAGE \
DEF:Speicherladepumpe= vito.rrd:Speicherladepumpe:AVERAGE \
DEF:DLE= vito.rrd:DLE2:AVERAGE \
DEF:Zirk= vito.rrd:Zirkulation:AVERAGE \
DEF:Verdichter= vito.rrd:Verdichter:AVERAGE \
CDEF:SL=Speicherladepumpe,5,* \
CDEF:VD=Verdichter,10,* \
CDEF:DE=DLE,2,* \
CDEF:ZI=Zirk,0.5,* \
HRULE:-20#AAAAAA HRULE:-10#AAAAAA HRULE:0#AAAAAA HRULE:10#AAAAAA \
HRULE:20#AAAAAA \
HRULE:30#AAAAAA HRULE:40#AAAAAA HRULE:50#AAAAAA HRULE:60#AAAAAA \
AREA:VD#CCCCC:"Verdichter" \
AREA:SL#AAAAAA:"Warmw. Ladepumpe" \
AREA:DE#CBCEFC:"Zusatzheizung" \
AREA:ZI#FCCBCE:"Zirkulation" \
LINE1:TempA#000000:"Aussen [C]" \
LINE1:TempSekVL#FF4500:"Sek VL [C]" \
LINE1:TempSekRL#3A5FCD:"Sek RL [C]" \
LINE2:TempM2VList#FF0000:"FBH V1 [C]" \
LINE2:tempWwist#FFA500:"Warmw. [C]" \
LINE1:DruckVerdampfer#008B00:"Verdampfer [bar]" \
LINE1:DruckKondensator#00FF00:"Kondensator [bar]" \
LINE1:TempVerdampfer#00FFAA:"Verdampfer [C]" \
VDEF:tempAkt=TempA,AVERAGE \
VDEF:tempAmax=TempA,MAXIMUM \
VDEF:tempAmin=TempA,MINIMUM \
VDEF:tempWwAkt=tempWwist,LAST \
VDEF:tempWwmax=tempWwist,MAXIMUM \
VDEF:tempWwmin=tempWwist,MINIMUM \
VDEF:DruckKondensatorMax=DruckKondensator,MAXIMUM \
```

```
VDEF:tempVerdampferMin=TempVerdampfer,MINIMUM \
COMMENT:"\1" COMMENT:" " COMMENT:"\1" \
COMMENT:"\t\t\tmin.\t\t max.\t\t\durch." \
COMMENT:"\1" \
COMMENT:"Aussen\:" " \
GPRINT:tempAmin:"\t%03.11f C" \
GPRINT:tempAmax:"\t%03.11f C" \
GPRINT:tempAkt:"\t%03.11f C" \
COMMENT:"\1" \
COMMENT:"Warmwasser\:" " \
GPRINT:tempWwmin:"%03.11f C" \
GPRINT:tempWwmax:"\t%03.11f C" \
GPRINT:tempWwAkt:"\t%03.11f C" \
COMMENT:"\1" \
COMMENT:"Kondensator\:" " \
GPRINT:DruckKondensatorMax:"%03.11f bar" \
COMMENT:"\1" \
COMMENT:"Verdampfer\:" " \
GPRINT:tempVerdampferMin:"%03.11f C" \
COMMENT:"\1"
```

Vielen Dank an Vitoopen für den Anfang - wollte da seit Wochen schon ran Gruß user:warmup

## 9. Daten in MySQL speichern

Die Speicherung der Daten in einer MySQL-Datenbank ist nicht Raspberry PI spezifisch und kann natürlich auch auf anderen Rechnern/Geräten genutzt werden.

### 9.1 Installation von MySQL

Installation der Datenbank:

```
sudo apt-get install mysql-server
```

Um auf die Daten in der Datenbank später auch per PHP zugreifen zu können muss noch das MySQL-PHP-Modul installiert werden:

```
sudo apt-get install php5-mysql
```

### 9.2 Tabellen anlegen

Die Tabellenstrukturen sind nur Beispiele und müssen ggf. je nach "commands" angepasst werden falls zusätzliche oder weniger Werte gespeichert werden sollen:

Tabellenstruktur für Tabelle **brenner**

```
CREATE TABLE IF NOT EXISTS `brenner` (
  `timestamp` datetime NOT NULL,
  `brennerstarts` float NOT NULL,
  `brennerstunden` float NOT NULL,
  `brennerstatus` int(11) NOT NULL,
  KEY `timestamp` (`timestamp`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Tabellenstruktur für Tabelle **solar**

```
CREATE TABLE IF NOT EXISTS `solar` (
  `timestamp` datetime NOT NULL,
  `solarstunden` float NOT NULL,
  `solarleistung` float NOT NULL,
  `solarpumpe` int(11) NOT NULL,
```

```

KEY `timestamp` (`timestamp`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

### Tabellenstruktur für Tabelle **temperaturen**

```

CREATE TABLE IF NOT EXISTS `temperaturen` (
  `timestamp` datetime NOT NULL,
  `aussentemperatur` float NOT NULL,
  `warmwasser` float NOT NULL,
  `speicher_unten` float NOT NULL,
  `kollektor` float NOT NULL,
  `vorlaufsoltemperaturM2` float NOT NULL,
  KEY `timestamp` (`timestamp`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

### Tabellenstruktur für Tabelle **snapshot**

```

CREATE TABLE IF NOT EXISTS `snapshot` (
  `timestamp` datetime NOT NULL,
  `brennerstatus` float NOT NULL,
  `brennerstarts` float NOT NULL,
  `brennerstunden` float NOT NULL,
  `solarstunden` float NOT NULL,
  `solarleistung` float NOT NULL,
  `aussentemperatur` float NOT NULL,
  `warmwasser` float NOT NULL,
  `speicher_unten` float NOT NULL,
  `kollektor` float NOT NULL,
  `kesseltemperatur` float NOT NULL,
  `vorlauftemperaturM2` float NOT NULL,
  `vorlaufsoltemperaturM2` float NOT NULL,
  `raumsoltemperaturM1` float NOT NULL,
  `raumsoltemperaturM2` float NOT NULL,
  `raumsoltemperaturredM1` float NOT NULL,
  `raumsoltemperaturredM2` float NOT NULL,
  `warmwassersoll` float NOT NULL,
  `kesseltemperatursoll` float NOT NULL,
  `pumpestatusM1` float NOT NULL,
  `pumpestatusSP` float NOT NULL,
  `pumpestatussolar` float NOT NULL,
  `statusstoerung` varchar(100) CHARACTER SET latin1 COLLATE latin1_german1_ci NOT NULL,
  `systemzeit` varchar(100) CHARACTER SET latin1 COLLATE latin1_german1_ci NOT NULL,
  `error0` varchar(500) CHARACTER SET latin1 COLLATE latin1_german1_ci NOT NULL,
  `BetriebArt` varchar(20) NOT NULL,
  `BetriebArtM2` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

Die snapshot-Tabelle soll im Beispiel keine Historie enthalten, sondern immer nur eine Zeile mit den alle zwei Minuten ausgelesenen Werten. Das hat sich nach ein paar Monaten Betrieb als sinnvoll erwiesen, da der Raspberry doch ganz schön ackern muss um aus einer Tabelle mit > 150.000 Zeilen (und wachsend) die aktuellen Werte auf einer Webseite anzuzeigen.

### 9.3 Daten von vcontrold in MySQL-Datenbank speichern

Das Speichern der Daten in der MySQL-Datenbank setzt dort an, wo auch die Daten in der rrdb geschrieben werden wie unter 8.2 beschrieben. Anstatt, oder auch *zusätzlich* zu der Speicherung in der rrdb was mit dem Aufruf

```
update vito.rrd N:$1:$2:$3:$4:$5:$6:$7:$8:$9:$10:$11:$12:$13:$14:$15:$16
```

geschieht, sind für die Speicherung der Daten in einer MySQL-Datenbank folgende Zeilen an das Ende der TPL-Datei einzufügen. Dazu wird der Aufruf des Kommandozeilentools "mysql" welches bei der Installation der Datenbank enthalten ist genutzt. In dem Befehl sind , und durch die entsprechenden Werte zu ersetzen.

```
mysql --user=<benutzer> --password=<passwort> <dbname> -e "INSERT INTO temperaturen
(timestamp, aussentemperatur, warmwasser, kollektor, speicher_unten,
vorlaufsollltemperaturM2) values (CURRENT_TIMESTAMP,$1,$2,$4,$5,$13);"
```

```
mysql --user=<benutzer> --password=<passwort> <dbname> -e "INSERT INTO solar (timestamp,
solarstunden, solarleistung) values (CURRENT_TIMESTAMP,$9,$10);"
```

```
mysql --user=<benutzer> --password=<passwort> <dbname> -e "INSERT INTO brenner
(timestamp, brennerstarts, brennerstunden, brennerstatus) values
(CURRENT_TIMESTAMP,$6,$7,IFNULL('$19',0));"
```

Die letzte SQL-Befehl ist, wie oben bei der Tabellenstruktur der Tabelle "snapshot" schon beschrieben eine Besonderheit, da anstelle des INSERT ein UPDATE vorgenommen wird.

```
mysql --user=<benutzer> --password=<benutzer> <dbname> -e "UPDATE snapshot SET
timestamp=CURRENT_TIMESTAMP, aussentemperatur=$1, warmwasser=$2, kollektor=$4,
speicher_unten=$5, solarstunden=$9, solarleistung=$10, brennerstarts=$6,
brennerstunden=$7, kesseltemperatur=$12, vorlauftemperaturM2=$13,
vorlaufsollltemperaturM2=$14, raumsolltemperaturM1=$15, raumsolltemperaturM2=$16,
raumsolltemperaturredM1=$17, raumsolltemperaturredM2=$18, brennerstatus=IFNULL('$19',0),
warmwassersoll=$20, kesseltemperatursoll=$21, pumpestatusM1='$22', pumpestatussp='$23',
pumpestatussolar='$24', statusstoerung='$R25', systemzeit='', error0='$R27',
BetriebArt='$R29', BetriebArtM2='$R30';"
```

Gruß user:diefenbecker

## 10. Regelung der Vitotronic auf Basis von Wettervorhersage

Wenn Ihr wie ich eine Luftwaermepumpe mit Fussbodenheizung in einem gut isoliertem Haus habt, kennt ihr ja die Probleme damit: sehr träges System und die Heizung arbeitet je kälter es draussen wird mehr und ineffektiver. Ausserdem heizt die Sonne an sonnigen Tagen die Wohnung innerhalb kurzer Zeit enorm auf. Darum hab ich mir überlegt, eine Wettervorhersage für die nächsten Stunden zusammen mit der Ansteuermöglichkeit der Heizung mittels Raspberry Pi zu nutzen. Ich habe übrigens eine Vitocal 200-S, Bj 2013 und eine Vitocell 100 als Warmwasserspeicher. Die Regelung nutzt die Möglichkeit des Raspberry Pi mittels Node.js-Script die Wettervorhersage aus Openweathermap.org bzw. Wunderground.com einzulesen und aufgrund den Daten aus zukünftiger Temperatur und Wolkenbedeckung die Heizung mittels VCLIENT anzupassen. Das Node.js Script wird dann mittels Cronjob z.B. stündlich aufgerufen. Anbei erkläre ich kurz die Schritte zum nachmachen:

1. Nach Installation von node.js auf dem Raspi müssen die einzelnen Packages mit npm installiert werden:

```
sudo apt-get install nodejs
npm install jsonrequest fs dateformat math sunalc
```

2. ...fast bereit. Ich hab jetzt noch eine Hilfsroutine ausgegliedert und in interpolate.js im Verzeichnis /home/pi/Wetter gespeichert. Diese Funktion macht die Nutzung der Wettervorhersage einfacher, da die Wetterdaten damit kontinuierlich, d.h. zu jeder Zeit quasi

zur Verfügung stehen.

```
// Data Interpolation sub function. Taken from somewhere.

exports.createInterpolant = function(xs, ys) {
  var i, length = xs.length;

  // Deal with length issues
  if (length !== ys.length) { throw 'Need an equal count of xs and ys.'; }
  if (length === 0) { return function(x) { return 0; }; }
  if (length === 1) {
    // Impl: Precomputing the result prevents problems if ys is mutated later and
    // allows garbage collection of ys
    // Impl: Unary plus properly converts values to numbers
    var result = +ys[0];
    return function(x) { return result; };
  }

  // Rearrange xs and ys so that xs is sorted
  var indexes = [];
  for (i = 0; i < length; i++) { indexes.push(i); }
  indexes.sort(function(a, b) { return xs[a] < xs[b] ? -1 : 1; });
  var oldXs = xs, oldYs = ys;
  // Impl: Creating new arrays also prevents problems if the input arrays are mutated
  later
  xs = []; ys = [];
  // Impl: Unary plus properly converts values to numbers
  for (i = 0; i < length; i++) { xs.push(+oldXs[indexes[i]]); }
  ys.push(+oldYs[indexes[i]]); }

  // Get consecutive differences and slopes
  var dys = [], dxs = [], ms = [];
  for (i = 0; i < length - 1; i++) {
    var dx = xs[i + 1] - xs[i], dy = ys[i + 1] - ys[i];
    dxs.push(dx); dys.push(dy); ms.push(dy/dx);
  }

  // Get degree-1 coefficients
  var c1s = [ms[0]];
  for (i = 0; i < dxs.length - 1; i++) {
    var m = ms[i], mNext = ms[i + 1];
    if (m*mNext <= 0) {
      c1s.push(0);
    } else {
      var dx_ = dxs[i], dxNext = dxs[i + 1], common = dx_ + dxNext;
      c1s.push(3*common/((common + dxNext)/m + (common + dx_)/mNext));
    }
  }
  c1s.push(ms[ms.length - 1]);

  // Get degree-2 and degree-3 coefficients
  var c2s = [], c3s = [];
  for (i = 0; i < c1s.length - 1; i++) {
    var c1 = c1s[i], m_ = ms[i], invDx = 1/dxs[i], common_ = c1 + c1s[i + 1] - m_ -
    m_;
    c2s.push((m_ - c1 - common_)*invDx); c3s.push(common_*invDx*invDx);
  }

  // Return interpolant function
  return function(x) {
    // The rightmost point in the dataset should give an exact result
    var i = xs.length - 1;
    if (x == xs[i]) { return ys[i]; }

    // Search for the interval x is in, returning the corresponding y if x is one of
    // the original xs
    var low = 0, mid, high = c3s.length - 1;
    while (low <= high) {
      mid = Math.floor(0.5*(low + high));
      var xHere = xs[mid];
      if (xHere < x) { low = mid + 1; }
      else if (xHere > x) { high = mid - 1; }
      else { return ys[mid]; }
    }
    i = Math.max(0, high);
  }
}
```

```

    // Interpolate
    var diff = x - xs[i], diffSq = diff*diff;
    return ys[i] + c1s[i]*diff + c2s[i]*diffSq + c3s[i]*diff*diffSq;
  };
};

```

3. Das eigentliche Steuerscript sieht nun wie folgt aus und habe ich unter /home/pi/Wetter als Regelung.js abgespeichert:

```

// Node.js File to Controll Room Setpoint Temperature of Vitotronic related to weather
forecast
// Author: konni100000 22/03/2016

var JsonRequest = require("jsonrequest");
var fs = require("fs");
var dateFormat = require('dateformat');
var math = require('math');
var child_process = require('child_process');
var SunCalc = require('suncalc');
var ip = require('/home/pi/Wetter/interpolate.js');

// Main function to transfer forecast data to controll function
function controll_vito(x){
  JsonRequest("http://api.openweathermap.org/data/2.5/forecast/city?
q=berlin&APPID=xxxxxxx", \ use own city and get APPID from OWM
  function (err, data) {
    console.log(data.list[0].main.temp);
    console.log(err);
    set_new_setpoint(data,x)}
  )
};

// Subfunction to calculate new setpoint
function set_new_setpoint(data,x){
  // Weather forecast in x
  hours
  var d = [];
  var temp =[];
  var clouds = [];
  var wind = [];
  for (var i =0; i <= 6; i++) {
    // read the Weather data
    from Json
    d[i] = new Date(data.list[i].dt);
    temp[i] = data.list[i].main.temp;
    clouds[i] = data.list[i].clouds.all;
    wind[i] = data.list[i].wind.speed;
  }
  var temp_fc = ip.createInterpolant(d, temp);
  var clouds_fc = ip.createInterpolant(d, clouds);
  var wind_fc = ip.createInterpolant(d, wind);
  var now = new Date().getTime()/1000;
  var delta_temp_xh = temp_fc(now + x * 3600)-temp_fc(now); // temperature difference
  from now to x hours in the future
  var clouds_xh = clouds_fc(now + x * 3600);
  var wind_xh = wind_fc(now + x * 3600);
  var hours = new Date().getHours();
  var formatted_dat= dateFormat(new Date().getTime(), "dd.mm.yyyy HH:MM:ss");
  var cmd = String("vclient -h localhost -p 3002 -c getTempSekRL");
  child_process.exec(cmd,function(error,stdout,stderr){
    stdout = stdout.replace("\n", " ");
    stdout = stdout.replace("Degrees Celsius","");
    stdout = stdout.replace("getTempSekRL: ", "");
    var RTsoll = 22;
    var rueckl = parseInt(stdout);
    var sonne = math.max(((100-clouds_fc(now))/100*sunheat(0)),((100-
clouds_xh)/100*sunheat(x)));
    var setpoint = math.round(RTsoll-delta_temp_xh*0.4-sonne); // Concept: when
temp is higher in x h, the RTsoll should be
// reduced (0.4 is
the slope of the heat curve
// sonne is the
sun heat at the location and time
    console.log("Berechneter Setpoint " + setpoint + "Tempreduktion durch Sonne" +sonne);
    if (rueckl > 26 && rueckl < 30) {setpoint--}; // when heat
return is too high reduce setpoint

```

```

    if (rueckl > 27 && rueckl <30) {setpoint--}; // or even
far too high
    if (setpoint < 16) {setpoint = 16}; // controll
min is 16°C
    if (setpoint > 24) {setpoint = 24}; // controll
max is 24°C
    console.log(setpoint + " " + rueckl); // verify
what is done

// log time and setpoint in file
    fs.appendFile('/home/pi/log/Regelung.txt', formatted_dat + " Raumtemperatursoll "
+ setpoint + "\n", function (err) {
    if (err) return console.log(err)
    });
// write the setpoint via vclient to Vitotronic
    var cmd = String("vclient -h localhost -p 3002 -c setRT"+setpoint);
    child_process.exec(cmd,function(error,out,err){
        console.log(out||err)
    });
});
}

// Determine the sun heat related to the sun position and location: in this case highest
point of sun in summer returns a
// heating temperature reduction of 9K. In winter this corresponds to 3K at noon, if
there are no clouds

function sunheat(x){
    var now = new Date().getTime();
    var times = SunCalc.getTimes(now, 47.2873419,11.4050252); // use own
koordinates e.g. from google maps
    var sunrise = times.sunrise.getHours() + ':' + times.sunrise.getMinutes();
    var sunset = times.sunset.getHours() + ':' + times.sunset.getMinutes();
    // console.log("heute " + sunrise + " " + sunset);
    var zeit = now + x*3600000;
    var position = SunCalc.getPosition(zeit,47.2873419,11.4050252); // use own
koordinates e.g. from google maps
    if (now > times.sunrise && now < times.sunset) {
        var strahlung = math.sin(position.altitude)*10;
    }
    else {strahlung = 0};
    return strahlung;
}

// main funtion call: e.g. look at the weather in 4h
controll_vito(4);

```

4. Das Script nutzt definierte Commands aus der vito.xml und vcontrold.xml. Meine Dateien hab ich Euch hier angehängt:

[vito.xml](#)

[vcontrold.xml](#)

5. Das node.js-File sollte jetzt bei Aufruf

node /home/pi/Regelung.js

die Raumtemperatur-Soll ändern und keinen Fehler mehr zurückliefern.

6. Wenn dies der Fall ist, lässt man Cronjob den Rest automatisch erledigen:

```

crontab -e

----
0,30 * * * * node /home/pi/Wetter/Regelung.js

mit ":w" und ":q" schreiben und beenden.

```

Jetzt startet Cronjob die Regelung halbstündlich. Auch wenn der Pi aus- und wieder eingeschaltet wird, funktioniert die Regelung automatisch wieder. Zu meiner Erfahrung: Das obige System läuft jetzt bei mir seit Ende Januar ohne Ausfall und bislang mussten wir noch nicht frieren ;). Tatsächlich sehe ich an bewölkten Tagen mit wenig Außentemperaturänderung kaum einen Eingriff der Regelung. Hingegen greift an sonnigen Tagen mit starker Temperaturschwankung die Regelung z.T. drastisch ein und sorgt auch für den gewünschten Effekt. Für meinen Standort habe ich mich für Openweathermap.org als Wettervorhersage entschieden, nachdem ich auch Wunderground.com ausprobiert haben. Zwar passen hier auch die tatsächlichen Temperaturen nicht immer, allerdings wird in der Regelung lediglich die Temperaturänderung berücksichtigt, da hier ja lediglich die Raumsolltemperatur geändert wird. Die Vorlauftemperatur wird immer noch von der Viessmansteuerung auf Basis der gemessenen Außentemperatur direkt ermittelt.

7. Bereiten des Warmwassers zur besten Tageszeit auf Basis der Wettervorhersage: Bisher hab ich mich noch nicht an das Schreiben in die Zeitregister herangetraut. Aber es gibt ja einen rel. einfachen Steuerbefehl in der Vitotronic: "1xWW". Hierbei wird der Warmwasserspeicher einmalig durchgeheizt. Sinn hiervon ist natürlich nicht die tägliche Warmwasserbereitung, sondern das hygienisieren des Warmwasserspeichers auf 60°C. Diese Funktion ist wohl lediglich bei Mehrfamilienhäusern von Bedeutung. Wenn man allerdings die maximale Warmwassertemperatur auf die gewünschte Warmwassertemperatur stellt (z.B. 45°C), kann man die Funktion rel. einfach für das tägliche Aufwärmen des Warmwasserspeichers nutzen. Dafür nutze ich ein separates Script, das einmalig am Tag aufgerufen wird (z.B. 7:00 Uhr morgens). Das Script schreibt dann die günstigste Tageszeit in eine Datei, die dann von einem anderen Script alle 30min überprüft wird. Ist die Zeit gekommen, sendet das Script den "1xWW" Befehl an die Vitotronic. In meinem Fall habe ich einen um 20% günstigeren Nachttarif zwischen 20:00Uhr und 7:00Uhr morgens. Typischerweise ist um 20Uhr die wärmste Zeit im Nachttarif und somit die günstigste Tageszeit, um das Warmwasser zu erwärmen. Allerdings kann die Temperatur, v.a. im Sommer untertags deutlich wärmer werden und somit eine effizientere Erwärmung erlauben. Dies wird von dem Script geprüft. Bei Euch kann das natürlich ganz anders sein und muss wohl angepasst werden.

Script zum Finden der Besten Tageszeit (waterheating.js):

```
var JsonRequest = require("jsonrequest");
var fs = require("fs");
var dateFormat = require('dateformat');
var math = require('math');
var child_process = require('child_process');
var ip = require('/home/pi/Wetter/interpolate.js');

// Main function to call forecast data and call subfunction
// to get best heating time within the next 24 hours

function waterheating(){
  JsonRequest("http://api.openweathermap.org/data/2.5/forecast/city?q=moscow&APPID=xxxx",
    function (err, data) {set_new_time(data)} );
}

function set_new_time(data){
  var besthour = 20; // from 20:00 there is night tarif and it is the warmest time in the
  night Tarif
  var tempdiffmax = 6; // difference when heating in the high tarif during the day is
  cheaper
  var d = [];
  var temp =[];
  var maxtemp = 0;
  for (var i =0; i <= 7; i++) { // get the warmest time for the next 24hours
    d[i] = data.list[i].dt;
    temp[i] = data.list[i].main.temp;
    if (temp[i]>maxtemp){maxtemp = temp[i];var maxtime=d[i]}
  }
  var temp_fc = ip.createInterpolant(d, temp);
  var now = new Date().getTime()/1000;
  var hours = new Date().getHours(); var todaybesthour = now + 3600*(besthour-hours);
  tempdiff = temp_fc(maxtime)-temp_fc(todaybesthour); // from 20:00 there is the night
  tarif if (tempdiff< tempdiffmax) {var wwDate = todaybesthour} else {var wwDate =
  maxtime};
  var heatingtime = new Date(wwDate*1000); // just for logging
  console.log("heating of water today @ " + heatingtime + " with a air temp of "
    + math.round((temp_fc(wwDate)-273.15)*100)/100 + "C \n");
  fs.writeFile('/home/pi/log/WW.txt', wwDate, function(err){console.log(err)} );
}
```



```

}

// funtion call
waterheating();

```

und hier das 2. Script zum Setzen des Steuerbefehls:

```

function waterheating(){
  var now = new Date().getTime()/1000;
  fs.readFile("/home/pi/log/WW.txt","utf8",function(error,data) {
    if ((data-now)<0 && (data-now)>-1800){ // check if time has come to
    set heating
    // set 1xWW with a temp 45C
    var cmd = String("vclient -h localhost -p 3002 -c set1xWWein setTempWWSoll2_45");
    child_process.exec(cmd,function(error,out,err){console.log(out|err)});
    var formatted_dat= dateFormat(new Date().getTime(), "dd.mm.yyyy HH:MM:ss");
    fs.appendFile('/home/pi/log/1xWW.txt', formatted_dat + " 1xWW " + "\n", function
(err) {
      if (err) return console.log(err)<span style="font-family: arial,Helvetica,sans-
serif;"> }> });
    </span>
      else { if ((data - now) < 0) {console.log("Letzte 1xWW vor " + math.round((data-
now)/3600) + " Stunden")}
      else {console.log("Dauert noch: " + (data-now) + "sec");
      }
    }
  });
}
waterheating();

```

So. Das mit der Warmwassererhitzung hab ich zwar getestet, läuft allerdings noch nicht wirklich lange. Ich denke aber, das es funktionieren sollte. Die oberen Scripts kann man dann auch doppelt anlegen, wenn man z.B. wöchentlich eine Erwärmung auf 55°C will. Das funktioniert dann im Prinzip genauso.

Wenn ihr Feedback oder eigene Verbesserungen/Erfahrungen habt, bitte melden. Würde mich natürlich sehr freuen :)

Grüße user:konni100000

Update: Mittlerweile läuft meine Regelung seit einem Jahr ununterbrochen und ich habe noch ein paar Verbesserungen eingebracht:

- Berücksichtigung der aktuellen Raumtemperatur über einen Funksensor
- Fenstermodell fürs Haus zum berücksichtigen der Einstrahlintensität
- Auslagerung der Parameter in eine Config-Datei, die dann über Dropbox /Exceldatei änderbar ist => volle Kontrolle auch unterwegs
- Komforttemperaturen für definierte Zeitfenster

Da Nebel nicht in der Vorhersage zuverlässig erkannt wird gab es ein paar kalte Tage :) Darum musste ich den Raumtemperatursensor miteinbinden, der dann die Heizung eingeschaltet lässt, wenns zu kühl wird. Mittlerweile bin ich begeistert => An Sonntagen schaltet die Heizung früh morgens aus und erst Nachts wieder an, wenn die Raumtemperatur abfällt. Dadurch wird unser Haus auch nicht mehr so stark bei Sonnenschein überhitzt. Die 4€/a Stromkosten des Raspberry hole ich sicherlich wieder rein :) Wenn ihr Interesse an einer Diskussion habt, meldet Euch einfach.

## Beste Grüsse, user:konni100000

TODO:

- Webserver auf dem Raspberry Pi mit Website zur Darstellung der Daten
- Auswertung mit munin
- ...

Bei Bedarf kann diese Anleitung von jedermann erweitert werden.

durchnummeriert

+ Add a custom footer