Mein Pi1 - Projekt

Zielsetzungen - der Raspi1b soll:

Meinen Solarheizung für den Whirlpool Ein- und ausschalten in Abhängigkeit der Temperaturen.

Hardware:

- Raspi 1b
- Zwei Relais auf einer Platine
- Dlan-Adapter
- Gartenpumpe
- Schläuche
- zwei Kollektormatten

Software:
- raspbian stretch lite

Vorbemerkung:
Das folgende Dokument ist quasi ein Protokoll meiner Aktivitäten zum Aufsetzen des Raspi wie oben beschrieben erstellt damit ich später auch noch weiß was ich getaen habe.
Die Anleitungen sind aus diversen Quellen zusammengesucht und ich hoffe ich verletze hiermit keine Copyrights.
Im wesentlichen gehe ich davon aus, auf dem Raspi als Administrator und auf dem Laptop als normaler Benutzer angemeldet zu sein. Normaler Text wird so geschrieben.
```
Befehlszeilen bzw. Code in Skripts sieht so aus.
```

Und zu guter Letzt, bei mir funktionierts, wer dies jedoch als Anleitung nutzt ist selber schuld.

(schuchardt.j@gmail.com)

Inhaltsverzeichnis

1 Grundinstallation

1.1 Vorbereitungen auf dem Laptop:

Siehe BrauPi
Mit dem Tool raspi-config zusätzlich das Verwenden der „predictable interface names“ einstellen. Dann heißen die Netzwerkkarten eth0 und wlan0. Macht das weitere einfacher.

1.2 Einstellungen die direkt am Raspi vorgenommen werden müssen / sollten

Siehe BrauPi

Vergeben einer statischen IP:
In der `/etc/dhcpcd.conf` die entsprechenden Abschnitte so anpassen:

```
# Example static IP configuration:
interface eth0
static ip_address=192.168.1.91/24
static routers=192.168.1.11
#static domain_name_servers=192.168.0.1 8.8.8.8
interface wlan0
static ip_address=192.168.1.90/24
static routers=192.168.1.11
#static domain_name_servers=192.168.0.1 8.8.8.8
```

(DNS und IP6 Adressen braucht der Raspi nicht)
An diese Stelle empfiehlt sich ein Neustart (init 6).

1.3 Weitere Konfiguration über das Netzwerk

Login mit ssh aus dem xterm des Laptops: `ssh pi@192.168.1.91`

Und da viele weitere Aktionen Administratorrechte erfordern gleich auf „root“ wechseln.

```
sudo su
```

1.3.1 Neuen Benutzer anlegen (optional)

```
adduser pi1
```

Eine Sicherheitskopie der /etc/group anlegen
`cp /etc/group /etc/group.bak`
und in der `/etc/group` mit `nano alle „pi“ in „pi1“` ändern.
Dann ausloggen und mit:

```
ssh pi1@192.168.1.91
```

unter dem neuen Namen anmelden und den Standardbenutzer pi mit

```
usermod -L pi
```

deaktivieren.

1.3.2 Verzeichnis zu Datenaustausch auf dem Raspi erstellen und einrichten

```
mkdir /home/pi1/export
```
(als User pi1)

und für alles freigeben:

```
chmod 777 /home/pi1/export
```

Den NFS Server installieren:

```
apt-get install nfs-kernel-server
apt-get install nfs-common
```

Die /etc/exports anpassen:

```
/home/pi1/export  192.168.1.0/255.255.255.0(rw,async,no_subtree_check)
/var/www/html     192.168.1.0/255.255.255.0(rw,async,no_subtree_check)
```

Und die Berechtigungen anpassen.

Die geänderte Datei neu einlesen und den NFS Server neu starten:

```
exportfs -ra
/etc/init.d/nfs-kernel-server restart
```

1.3.3 ssh – keys mit Laptop und KellerPi austauschen

Es ist nur ein ssh key pro Rechner möglich, da nein Abgleich mit der known-hosts Datei erfolgt. Daher nur die ssh keys für die Wlan-Verbindung ausgetauscht.

Mit:

```
ssh-keygen -b 4096
```

auf Laptop und KellerPi das Schlüsselpaar erzeugen und dann auf den pi1 kopieren:

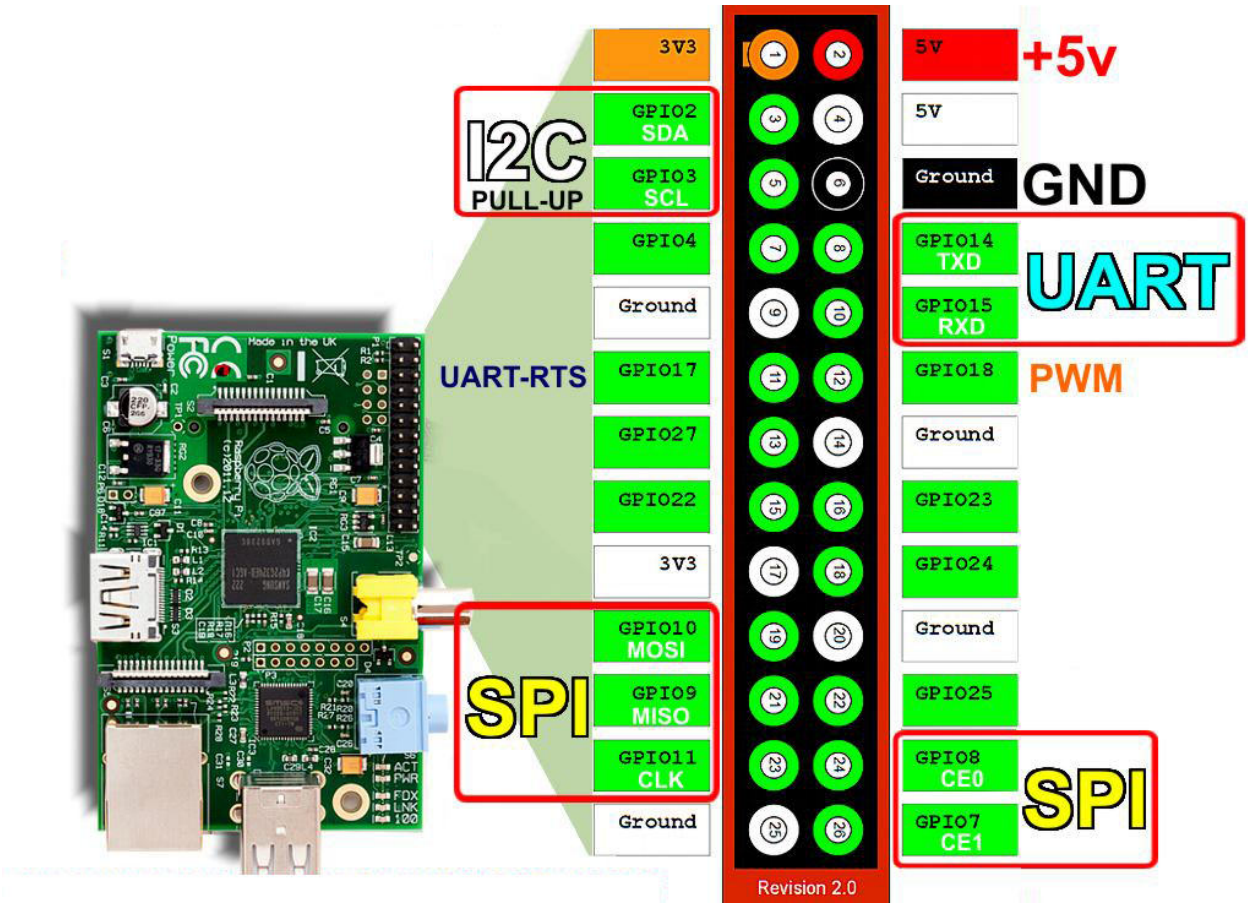
```
ssh-copy-id -i .ssh/id_rsa-xxxxxxx.pub pi1@192.168.1.90
ssh-copy-id -i .ssh/id_rsa-xxxxxxx.pub pi1@192.168.1.90
```

Die Schlüsselnamen id_rsa-xxxxxxx sind frei wählbar und werden vom ssh-keygen abgefragt.

(Warum beim Login vom KellerPi noch ein Passwort verlangt wird weiß ich im Moment nicht. Bisdarauf das das eine ssh-Sitzung in einer ssh-Sitzung ist alles wie vom Laptop.)

2 Die Hardware (mit GPIO Verkabelung)

Die GPIO Schnittstelle (vom 1er)



Festlegung:
Die Ansteuerung der Relais erfolgt mit den Pins 9, 13, und 15.
Pin 9 ist mit der Masse verbunden (grün).
Pin 13 (GPIO27) schaltet Relais 1 (orange / braun)
Pin 15 (GPIO22) schaltet Relais 2. (gelb / schwarz)
Pin 12 (GPIO18) wird mit Pin 13 verbunden und als Input verwendet (grau)
Pin 16 (GPIO23) wird mit Pin 15 verbunden und als Input verwendet (weiß)

Die Verkabelung der Relais analog KellerPi-Projekt.

Temperaturmessung
Anschluß des Sensors:
VCC (rot) an die 3,3V – PIN1
DATA (gelb) an den Pin 7(GPIO 4)
GND (schwarz) des Temperatursensors an den GND des Raspi – Pin6.

Des Weiteren muss zwischen den 3,3V und GPIO 4 ein 4,7k Ohm Widerstand eingesetzt werden (5kOhm tuns auch).

Einrichtung des Temperatursensors:

Installieren der Kernelmodule:

```
modprobe w1-gpio
modprobe w1-therm
```

Konfigurieren des Sensors:

Am Ende von /boot/config.txt

```
dtoverlay=w1-gpio,gpiopin=4,pullup=on
```

einfügen.

Nach einem Neustart sollte der Sensor unter /sys/bus/w1/devices/ mit einer Nummer im Format xx-xxxxxxxxxxxx aufgeführt sein.
Er kann dann mit `cat /sys/bus/w1/devices/xx-xxxxxxxxxxxx/w1_slave` ausgelesen werden.
Die Ausgabe sieht dann so aus:

```
56 01 4b 46 7f ff 0c 10 7b : crc=7b YES
56 01 4b 46 7f ff 0c 10 7b t=21375
```

t= ist die Temperatur in Milligrad.

Der zweite Temperatursensor kann in Reihe am gleichen Pin angeschlossen werden. (d.h. einfach alle Kontakte 1:1 verbinden. Er steht dann unter `/sys/bus/w1/devices/` mit eigener Adresse zur Verfügung.

Um die Sensoren auch in den Pool zu bekommen, muss die Leitung noch mit einem Messkabel verlängert werden. Die Zuordnung zu den Adern ist wie folgt (wilkürlich festgelegt):

	VCC	Data	GND
Sensor x	Rot	Gelb	Schwarz
Kabel für Sensor 1	Rot	Gelb	Weiß
Kabel für Sensor 2	Blau	Grün	Braun

(Fällt mir erst jetzt beim Schreiben ein, die Sensoren einzeln zu Verlängern ist überflüssig. Man hätte die auch in Reihe zusammengeschaltet verlängern können…).

Die Relais müssen während des Bootvorganges initialisiert werden. Dies kann in dem Startskript `/etc/rc.local` erledigt werden. Vor `exit 0` das Skript `init-relais.sh` aufrufen.

```
# GPIOs initialisieren
/root/boot/init-relais.sh start
```

Damit der Benutzer root auch auf die GPIOs zugreifen darf muß er noch der Gruppe hinzugefügt werden:

```
usermod -aG gpio root
```

2.1 Schalten der Relais und Lesen des Temperatursensors

Zum Ansteuern der Relais und zum Auslesen des Temperatursensors wurden mehrere Skripte erstellt:

`./relais.sh` - die Relais schalten
`./relais-status.sh` - den Status der Relais ausgeben
`./t-mess2.sh` - Temeratur in Milligrad mit Statusinformationen

`./t-mess.sh`	-	Temeratur in Milligrad
`./t-log.sh`	-	Temeratur in Milligrad kont. in die Datei t-log.txt schreiben
`./paus.sh`	-	Pumpe ein
`./pein.sh`	-	Pumpe aus
`./pstat.sh`	-	Pumpe Status ausgeben

pein.sh; paus.sh und pstat.sh sind im Prinzip überflüssig und nur als Kurzfassung der relais-Skripte gedacht.

Die Regelung der Pumpe erfolgt ebenfalls mittels eines Shellskriptes welches sekündlich die Temperaturen abgleicht und die Pumpe schaltet.
Hierbei werden mehrere Statusdateien erzeugt um Informationen mit der Webseite auszutauschen. Eigentlich hatte ich hierfür eine Datenbank vorgesehen, mit Rücksicht auf die Ressourcen des Pi1 allerdings nicht installiert.

`./regelung.sh`	-	Temperaturregelung für die Solarthermie (Pumpe ein/aus)
`./regelung2.sh`	-	etwas anderer Ansatz für die Regelung funzt noch nicht
`./pool-logger.sh`	-	mitschreiben der Temperaturen etc. in eine Textdatei

Da die Skripte ihren Status in Textdateien speichern, die z. Bsp. bei einem Absturz stehen bleiben, muss noch dafür gesorgt werden, dass die Statusdateien beim Booten gelöscht werden.

Aus der /etc/rc.local /root/boot/clean-status.sh aufrufen.

Die Skripte kommen in den Anhang.

3 Apache Webserver und Webseite

Der apache Webserver wird mit: `apt-get install apache2`
Installiert. Wenn das erfolgreich durchgelaufen ist, ist die Standardseite index.html unter 192.168.1.90 erreichbar. Damit auch php-Skripte funktionieren noch `apt-get install` php7.0.

Damit der Webserver auch von außen erreichbar ist, und auf Port 80 schon ein anderer Server läuft werden die Ports des Webservers noch auf 3000 / 3001 geändert:
`nano /etc/apache2/ports.conf`
Inhalt:

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf
Listen 3000
<IfModule ssl_module>
        Listen 3001
</IfModule>
<IfModule mod_gnutls.c>
        Listen 3001
</IfModule>
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Damit der apache auch auf die GPIOs zugreifen kann muß in der /etc/group noch www-data zur Gruppe gpio hinzugefügt werden. Wird erst nach einem Neustart wirksam.

```
nano /etc/group
gpio:x:997:pi1,root,www-data
```

Die Webseite soll in einem Passwortgeschützen Bereich liegen. Die Passwortabfrage muß in der apache Konfiguration aktiviert werden:

```
nano /etc/apache2/sites-available/000-default.conf
### für .htaccess
<VirtualHost *>
    <Directory /var/www/html/pwd/>
        Options +FollowSymLinks
        AllowOverride All
    </Directory>
</VirtualHost>
```

Ein Passwort erzeugen:

```
htpasswd -cs /etc/apache2/passwd pi1
```

Der Passwortgeschützte Bereich ist ein Unterverzeichnis in dem die Datei .htaccess liegt:

```
mkdir /var/www/html/pwd
nano /var/www/html/pwd/.htaccess
```

Inhalt:

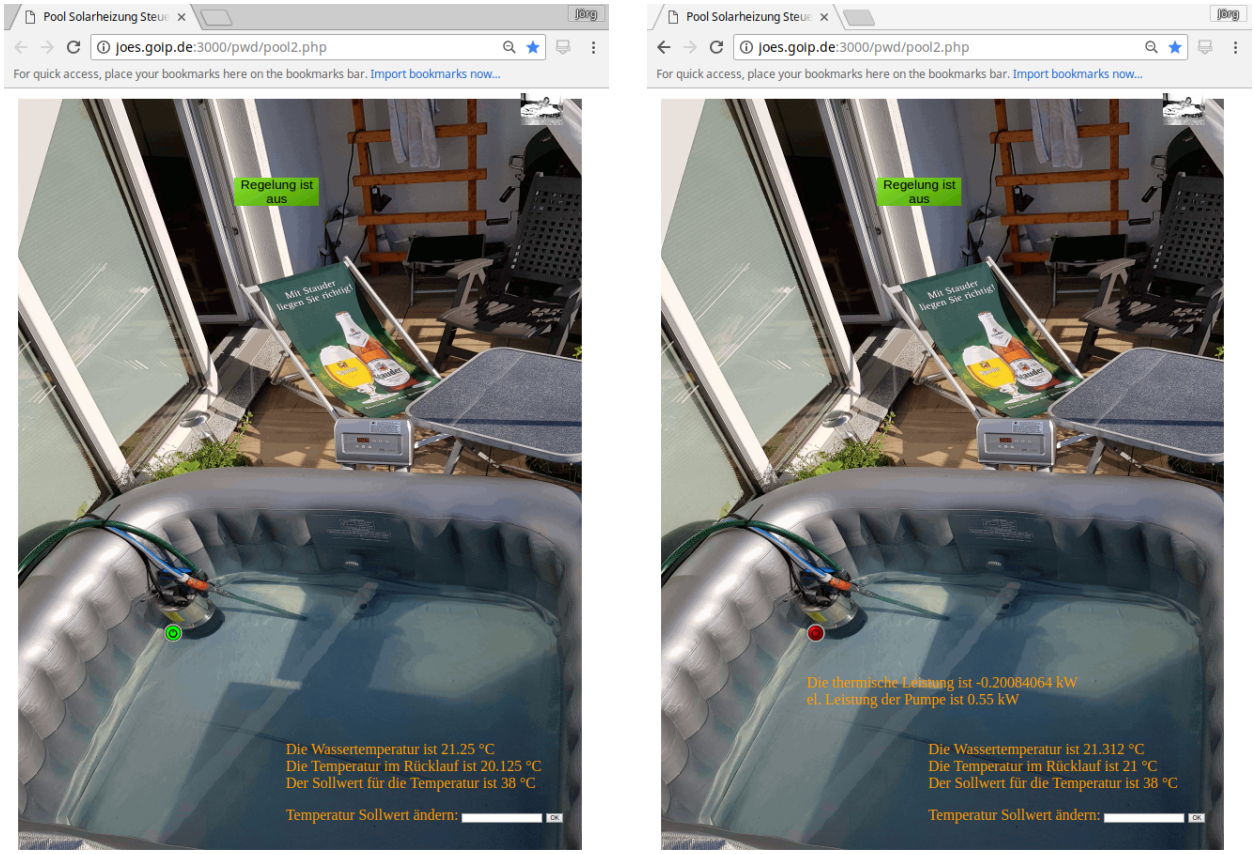
```
      Options -Indexes +FollowSymLinks
      AuthType Basic
      AuthUserFile /etc/apache2/passwd
      AuthName "KellerPi_aktionen"
      require valid-user
      IndexIgnore *
      DirectoryIndex pool.html
```

Wenn der Router richtig konfiguriert ist, ist der Server nach einem Neustart
`/etc/init.d/apache2 restart`
nun unter joes.goip.de:3000 erreichbar.

Jetzt kann die eigentliche Webseite erstellt werden.

`nano /var/www/html/pwd/pool2.php`

Derzeit halt bei der zweiten Version. Der Quelltext kommt ebenfalls in den Anhang.



Die Webseite einmal mit ausgeschalteter und einmal mit eingeschalteter Pumpe.

Wenn die Pumpe läuft wird die thermische Leistung unterhalb des Statusbuttons mit ausgegeben.

Ausblick:

Soweit funktioniert das erst mal, die Regelung in einer höheren Programmiersprache umzusetzen um auf Ereignisse Reagieren zu können wäre noch interessant.

4 Quelltexte

4.1 Shellskripte

#init-relais.sh

```
# Anlegen
echo "22" > /sys/class/gpio/export
echo "27" > /sys/class/gpio/export
# Als Ausgang festlegen (alternativ "in" für Eingang)
echo "out" > /sys/class/gpio/gpio22/direction
echo "out" > /sys/class/gpio/gpio27/direction
# Initialisieren (Steckdosen ausschalten
echo "0" > /sys/class/gpio/gpio22/value
echo "0" > /sys/class/gpio/gpio27/value
# Status auslesen (0 - aus; 1 - ein)
echo "Relais 1 Status"
cat /sys/class/gpio/gpio27/value # Relais 1
echo "Relais 2 Status"
cat /sys/class/gpio/gpio22/value # Relais 2
```

 # paus.sh

```
# Relais 2 (Pumpe) einschalten
echo "0" > /sys/class/gpio/gpio22/value
```

#pein.sh

```
# Relais 2 (Pumpe) einschalten
echo "1" > /sys/class/gpio/gpio22/value
```

 # pstat.sh

```
# Relais 2 Status  (0 - aus / 1 - ein)
wert=$(cat /sys/class/gpio/gpio22/value)
#echo "Relais 2 hat Status " $wert
if [ $wert = "0" ]
then
echo "Pumpe ist aus"
elif [ $wert = "1" ]
then
echo "Pumpe ist ein"
fi
```

#regelung.sh

```
#! /bin/bash
# Temperaturregelung mittels GPIO
# soll über php aufgerufen werden und so lange laufen wie in der Datei ../regelung_status.txt "ein"
steht
# als erster Parameter soll der Sollwert in °C übergeben werden
# es wird davon ausgegangen, das das Skript aus dem übergeordnetten Verzeichnis aufgerufen wird und
dort auch alle Dateien mit Parametern/Werten liegen

#Variablentypen deklarieren, damit man damit auch rechnen kann
typeset -i t1
typeset -i t2
typeset -i t2ein
typeset -i t2aus
typeset -i tsoll
typeset -i tdelta
typeset -i tsollein
typeset -i tsollaus
typeset -i pstatus

tdelta=200  # 0,2°C Hysterese

# Temperaturen einlesen in Milligrad
temp1=`echo $(cat /sys/bus/w1/devices/28-0517024421ff/w1_slave) | tail -c8` #Wassertemperatur
temp2=`echo $(cat /sys/bus/w1/devices/28-0516a7c3e1ff/w1_slave) | tail -c8` #Rücklauftemperatur
t1=${temp1:2:8}
```

```
t2=${temp2:2:8}
tsoll=$1
if [ 0 -lt $tsoll ]
then
rm ./status/tsoll.x
echo $tsoll >> ./status/tsoll.x
tsoll=$tsoll*1000
fi

t2=$t2+$tdelta # derzeit wird die Rücklauftemperautur künstlich erhöht, da sonst der Start nicht
funktioniert
tsoll=$(cat ./status/tsoll.x)
tsoll=$tsoll*1000
tsollein=$tsoll # einschalten wenn der Sollwert unterschritten wird
tsollaus=$tsoll+$tdelta # ausschalten wenn Sollwert + delta überschritten wird
status=$(cat ./status/regelung.x)
#echo $status

while [ "$status" = "ein" ]
do
# Wenn Pooltemperatur < Rücklaufund Pooltemperatur < tsoll und Pumpe aus, dann einschalten
pstatus=$(cat /sys/class/gpio/gpio22/value)
temp1=`echo $(cat /sys/bus/w1/devices/28-0517024421ff/w1_slave) | tail -c8` #Wassertemperatur
temp2=`echo $(cat /sys/bus/w1/devices/28-0516a7c3e1ff/w1_slave) | tail -c8` #Rücklauftemperatur
t1=${temp1:2:8}
t2=${temp2:2:8}
t2=$t2+$tdelta
tsoll=$(cat ./status/tsoll.x)
tsoll=$tsoll*1000
tsollein=$tsoll # einschalten wenn der Sollwert unterschritten wird
tsollaus=$tsoll+$tdelta # ausschalten wenn Sollwert + delta überschritten wird

if [ $pstatus=0 ] && [ $t1 -lt $tsollein ] && [ $t1 -lt $t2 ]
then
./shellskripte/pein.sh
fi
if [ $pstatus -ne 0 ]
then
if [ $t1 -ge $tsollaus ] || [ $t1 -ge $t2 ]
then
./shellskripte/paus.sh
fi
fi
sleep 1
status=$(cat ./status/regelung.x)

echo "Regelung" $status
echo "Pumpe" $pstatus
echo $t1
echo $t2
echo $tsoll
echo ""

done

./shellskripte/paus.sh
```

#regelung2.sh

in Arbeit

 # relais.sh

```
#!/bin/bash
if [ $# != 2 ]
then
echo " Es werden zwei Parameter erwartet."
echo " Bsp.: ./relais.sh 1/2 aus/ein"
echo " oder: ./relais.sh 1/2 0/1"
exit
fi
if [ $2 = "aus" ]
then
wert="0"
```

```
elif [ $2 = "0" ]
then
wert="0"
fi
if [ $2 = "ein" ]
then
wert="1"
elif [ $2 = "1" ]
then
wert="1"
fi
if [ $1 = 1 ]
then
echo $wert > /sys/class/gpio/gpio27/value
fi
if [ $1 = 2 ]
then
echo $wert > /sys/class/gpio/gpio22/value
fi
if [ $wert = "0" ]
then
wert="aus"
fi
if [ $wert = "1" ]
then
wert="ein"
fi
echo "Relais " $1 " ist " $wert
```

#relais-status.sh

```
#!/bin/bash
if [ $# != 1 ]
then
echo " Es wird ein Parameter erwartet."
echo " Bsp.: ./relais-status.sh 1/2"
exit
fi
if [ $1 = "1" ]
then
wert=$(cat /sys/class/gpio/gpio27/value)
elif [ $1 = "2" ]
then
wert=$(cat /sys/class/gpio/gpio22/value)
fi
echo "Relais " $1 " hat Status " $wert
if [ $wert = "0" ]
then
echo "aus"
elif [ $wert = "1" ]
then
echo "ein"
fi
```

#t1.sh

```
# Temperatur in Milligrad
t=`echo $(cat /sys/bus/w1/devices/28-0517024421ff/w1_slave) | tail -c8`
echo $t m°C
```

#t2.sh

```
# Temperatur in Milligrad
t=`echo $(cat /sys/bus/w1/devices/28-0516a7c3e1ff/w1_slave) | tail -c8`
echo $t m°C
```

 # ./shellskripte/pool-logger.sh

```
# maxlog ist die maximale Groesse für das Logfile
# in kB
maxlog=5000
logs=0

intervall=5    #default
if [ $1 -ge 0 ]
then
```

```
intervall=$1
fi

dname=_pooldata.txt
# Datum und Uhrzeit   Sekunden ergänzen
d=`date +%Y-%m-%d-%H-%M-%S`
dname="./log/$d$dname"
echo $dname
echo $dname >> $dname
echo "" >> $dname

status=$(cat ./status/log.x)
#echo $status
while [ "$status" = "ein" ]
do
d=`date +%Y-%m-%d-%H-%M-%S`
# Temperatur in Milligrad
t1=`echo $(cat /sys/bus/w1/devices/28-0517024421ff/w1_slave) | tail -c8`
t2=`echo $(cat /sys/bus/w1/devices/28-0516a7c3e1ff/w1_slave) | tail -c8`
pstatus=`echo $(./shellskripte/pstat.sh)`
rstatus=$(cat ./status/regelung.x)
tsoll=$(cat ./status/tsoll.x)

#echo $d $t1 m°C $t2 m°C     $pstatus     Regelung ist $rstatus Sollwert $tsoll 
echo $d $t1 m°C $t2 m°C     $pstatus     Regelung ist $rstatus Sollwert $tsoll °C >> $dname

logs=`echo $(ls -s $dname)`
set $logs
logs=$1
if [ $logs -ge $maxlog ]
then
echo "maximale Dateigröße erreicht" >> $dname
echo "aus" >> ./status/log.x
exit 1
fi

sleep $intervall
status=$(cat ./status/log.x)
done
```

#/root/boot/init-relais.sh start

```
### BEGIN INIT INFO
# Provides:          Initioalisieren der GPIO Schnittstelle für zwei Relais
# Required-Start:    ---
# Required-Stop:     ---
# Default-Start:     2
# Default-Stop:      0  6
# Short-Description:
# Description:
### END INIT INFO
# Author: <schuchardt.j@gmail.com>

case "$1" in
    start)
        # Anlegen
        echo "22" > /sys/class/gpio/export
        echo "27" > /sys/class/gpio/export
        # Als Ausgang festlegen (alternativ "in" für Eingang)
        echo "out" > /sys/class/gpio/gpio22/direction
        echo "out" > /sys/class/gpio/gpio27/direction
        # Initialisieren (Steckdosen ausschalten
        echo "0" > /sys/class/gpio/gpio22/value
        echo "0" > /sys/class/gpio/gpio27/value
        # Status auslesen (0 - aus; 1 - ein)
        echo "Relais 1 Status"
        cat /sys/class/gpio/gpio27/value # Relais 1
        echo "Relais 2 Status"
        cat /sys/class/gpio/gpio22/value # Relais 2
        ;;
    stop)
        echo "nothing"
        ;;
    *)
        echo "Benutzt: /etc/init.d/init-relais.sh {start|stop}"
        exit 1
        ;;
esac
```

```
exit 0
```

#/root/boot/clean-status.sh

```
rm /var/www/html/pwd/status/regelung.x
rm /var/www/html/pwd/status/log.x

echo "aus" >> /var/www/html/pwd/status/regelung.x
echo "aus" >> /var/www/html/pwd/status/log.x
```

4.1 Webseite in php

Pool2.php (wird laufend aktualisiert, besser in die Originaldatei schauen)

```
<html><head>

  <meta http-equiv="CONTENT-TYPE" content="text/html; charset=utf-8">
  <title>Pool Solarheizung Steuerung</title>

  <style>

  main {
   position: absolute;
   }
  #Hintergrund {
   background: top center no-repeat;
<!--   background-image: url("./pic/pool2.gif");-->
   background-size: cover;
   position: absolute;
   width = 100%;
   height = 100%;
   left: -10px;
   top: -10px;
   border: 0px;
   padding: 0%;
   margin: 0;
   font-size: 24pt;
   color:#FF9F00;
     }
  #pumpe {
   border: 0px;
   padding: 0%;
   position: absolute;
   left: 350px;
   top: 1150px;
     }
  #pumpebutton {
   border: 0px;
   padding: 0%;
   position: absolute;
   left: 0px;
   top: 0px;
     }
  #regelung {
   border: 0px;
   padding: 0%;
   position: absolute;
   left: 250px;
   top: 100px;
   width = 80px;
   height = 60px;
     }

   #zurueckB {
   border: 0px;
   padding: 0%;
   width = 200Â%;
   position: absolute;
   right: 80px;
   top: 20px;
     }
  #temperatur {
   border: 0px;
   padding: 0%;
   position: absolute;
   right: 80px;
   top: 1400px;
   width: auto;
   }
</style>

<p>
<div id="Hintergrund">
  <figure>
    <img src="./pic/pool2.gif" alt="">
```

```
  </figure>
<div id="zurueckB">  <a href="https://joes.goip.de"><img src="./pic/ich90x68.gif" alt="JoeS" > </a>
</div>
  <div id="temperatur">
    <?php
    $t1 = shell_exec("./shellskripte/t1.sh"); //Pool
    $t2 = shell_exec("./shellskripte/t2.sh"); //RÃ¼cklauf

    $pumpe = shell_exec("/var/www/html/pwd/shellskripte/pstat.sh");
    $r1 = shell_exec("/var/www/html/pwd/shellskripte/relais-status.sh 1");
    $r2 = shell_exec("/var/www/html/pwd/shellskripte/relais-status.sh 2");
        $durchfl = "0.154"; // Durchfluss gemessen mit zwei Kollektoren
    $durchfl= floatval($durchfl);

    $wassertemp = substr($t1, 2, -3);
    $ruecklauftemp = substr($t2, 2, -3);
    $wassertemp = floatval($wassertemp) / 1000;
    $ruecklauftemp = floatval($ruecklauftemp) / 1000;
    $pth = $durchfl * ( $ruecklauftemp - $wassertemp ) * 4.18;
    //$tsoll = $_POST["tsoll"];
    $tsoll = floatval(shell_exec("cat ./status/tsoll.x"));
    echo "Die Wassertemperatur ist ". $wassertemp . " Â°C";
    ?><br><?php
    echo "Die Temperatur im RÃ¼cklauf ist " . $ruecklauftemp ." Â°C";
    ?><br><?php
    echo "Der Sollwert fÃ¼r die Temperatur ist " . $tsoll ." Â°C";
    ?>
    <form action="tsoll.php" method="post">
              <p>
              Temperatur Sollwert Ã¤ndern:
              <input type="text" name="tsoll">
              <input type="submit" value="OK">
              </p>
       </form>


  </div>

  <div id="pumpe">
    <?php
    $statuspumpe = shell_exec("./shellskripte/pstat.sh");
    //FÃ¼r Butto
    if(isset($_GET['pumpeein'])) {
       //echo "Pumpe ein";
       $statuspumpe = shell_exec("./shellskripte/pstat.sh");
       echo shell_exec("./shellskripte/pein.sh");
       while($statuspumpe=="aus") {
        sleep(1);
        $statuspumpe = shell_exec("./shellskripte/pstat.sh");}
        header('location:'.$_SERVER['PHP_SELF']);}
    if(isset($_GET['pumpeaus'])) {
       //echo "Pumpe aus";
       $statuspumpe = shell_exec("./shellskripte/pstat.sh");
       echo shell_exec("./shellskripte/paus.sh");
       while($statuspumpe=="ein") { sleep(1);$statuspumpe = shell_exec("./shellskripte/pstat.sh");}
        sleep(1);
        header('location:'.$_SERVER['PHP_SELF']);}

    if (strpos($statuspumpe,"ein")!==false) { //echo "Pumpe ist an<br>";
       ?>
        <form id="pumpebutton" action="?pumpeaus=" method="POST">
       <input type="image" src="./pic/ausschalter_40x40.gif" alt="Pumpe ausschalten">
       </form>
        <br><br><br>
       <?php
       echo "Die thermische Leistung ist " . $pth ." kW";  ?><br><?php
       echo "el. Leistung der Pumpe ist 0.55 kW";  ?><br><?php
       }
    elseif (strpos($statuspumpe,"aus")!==false) { //echo "Pumpe ist aus<br>";
       ?>
       <form id="pumpebutton" action="?pumpeein=" method="POST">
              <input type="image" src="./pic/einschalter_40x40.gif" alt="Pumpe einschalten">
       </form>
       <?php
       }
    else
       {echo "Auswertung fehlgeschlagen<br>";}
//     ?>
  </div>
  <div id="regelung">
```

```
    <?php
    $statusregelung = shell_exec("cat ./status/regelung.x");
    //Reaktionen auf Button
    if(isset($_GET['regelungein'])) {
        shell_exec("rm ./status/regelung.x");
        shell_exec("echo ein >> ./status/regelung.x");
        shell_exec("./shellskripte/regelung.sh $tsoll 2>/dev/null >/dev/null &");
         $statusregelung = shell_exec("cat ./status/regelung.x");

         // hier fehlt noch eine vernÃ¼nftige Abfrage wann die Regelung lÃ¤uft fÃ¼r den Status der
Pumpe
         sleep(5);

         header('location:'.$_SERVER['PHP_SELF']);}
    if(isset($_GET['regelungaus'])) {
        shell_exec("rm ./status/regelung.x");
        shell_exec("echo aus >> ./status/regelung.x");
         $statusregelung = shell_exec("cat ./status/regelung.x");
         sleep(1);
        header('location:'.$_SERVER['PHP_SELF']);}
    if (strpos($statusregelung,"ein")!==false) { //echo "Regelung ist an<br>";
         ?>
        <form id="regelung" action="?regelungaus=" method="POST">
        <input type="image" src="./pic/ristein_180x60.gif" alt="Regelung ausschalten">
        </form>
        <?php
         }
    elseif (strpos($statusregelung,"aus")!==false) { //echo "Regelung ist aus<br>";
         ?>
         <form id="regelung" action="?regelungein=" method="POST">
        <input type="image" src="./pic/ristaus_180x60.gif" alt="Regelung einschalten">
         </form>
         <?php
         }
    else
         {?>
        <form id="regelung" action="?regelungaus=" method="POST">
        <input type="image" src="./pic/AusFehler.gif" alt="Auswertung  Regelung fehlgeschlagen">
        </form>
        <?php
        }
         ?>
  </div>
</div>
```

t-soll.php

```
<?php

$tsoll = $_POST["tsoll"];
shell_exec("rm ./status/tsoll.x");
shell_exec("echo $tsoll >> ./status/tsoll.x");
header('Location:'.$_SERVER['HTTP_REFERER']);  //Sprung zurück

?>
```