Mein BrauPi – Projekt

(

in Arbeit

derzeit on hold – Ansatz mit der Poolsteuerung scheint hierfür auch besser zu sein

)

Zielsetzungen - der Raspi soll:
- in einen Koffer eingebaut
- eine Herdplatte und ggf. ein Rührwerk ansteuern
- die Temperatur im Topf Messen und regeln können
- ein Temperaturprogramm abarbeiten
- über eine Webseite
    - manuell die Herdplatte und das Rührwerk schalten können
    - manuell die gewünschte Temperatur wählbar machen
    - die Temperaturprogramme auswählen und konfigurieren können
    - die aktuelle Temperatur anzeigen können
    - einen Temperaturverlauf mit Soll / Ist darstellen können

(für später Haken: √)

Hardware:
- Raspi Model ?? (eigentlich sollte der 1b reichen)
- Dlan Adapter
- Zwei Relais (Herdplatte, Rührwerk)
- Temperaturfühler
- Gehäuse (vorzugsweise Werkzeugkoffer)
- Jumperkabel (weiblich-weiblich)
- Netzteil (ggf als Steckdose mit USB)

Software:
- raspbian stretch lite
- apache2 webserver / ggf. node.js

Benutzte Programme:
- Linux Mint auf Laptop
- Bluefish html-Editor

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.`

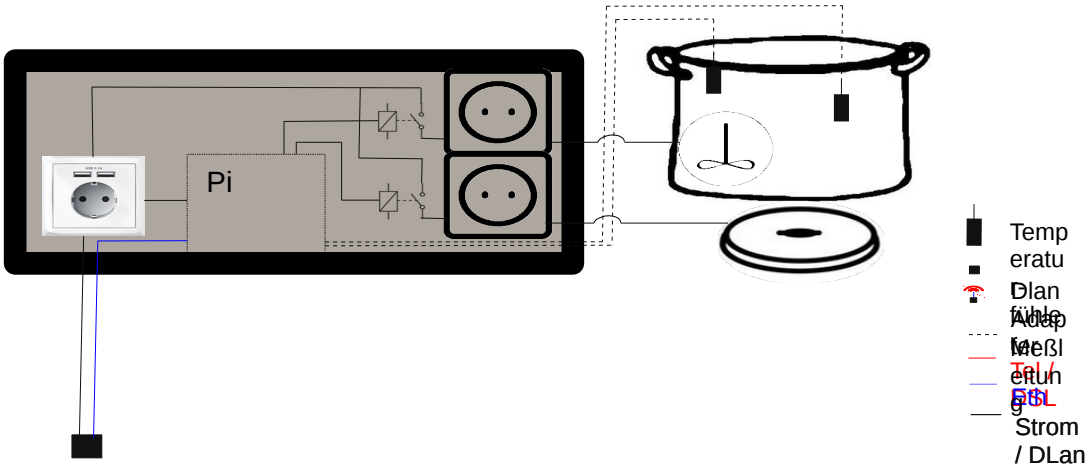
Und zu guter Letzt, bei mir funktioniert‘s hoffentlich irgendwann, wer dies jedoch als Anleitung nutzt ist selber schuld.
(schuchardt.j@gmail.com)

Inhaltsverzeichnis

1.

2.

1. Schema



Der BrauPi, eingebaut in einem Koffer, bekommt 240V Spannungsversorgung sowie Netzwerk über Ethernet-Kabel. Zwei Temperaturfühler sind mit langem und flexiblem Kabel an der GPIO-Schittstelle angeschlossen (Zugentlastung). Mit zwei Relais können zwei Steckdosen je nach gemessener Temperatur ein- und ausgeschaltet werden. Hiermit werden die an die Steckdosen anzuschließenden Herdplatten gesteuert.

Die Software soll dann ermöglichen, eine Temperatur zu halten bzw. ein Temperaturprogramm abzufahren.

2. Grundinstallation

Hier nur kurz, was getaen wurde. Ausführlich steht das im KellerPi-Projekt.
- mit raspi-config die wesentlichen Einstellungen vornehmen (inkl. GPIO Netzwerkzugriff)
- Benutzer pi1 anlegen und pi deaktivieren
(sudo Passwort = Benutzerpasswort; /etc/groups und /etc/sudoers anpassen)
- statische IP auf 192.168.1.91 setzen
- ssh key austauschen (login ohne Passwort)
- Software aktualisieren und weitere installieren
(apt-get update, apt-get upgrade, apt-get install nfs-kernel-server mc cifs-utils
)
- export Verzeichnis anlegen und exportieren
- ssh portforwarding Im Router aktivieren, damit kann über joes.goip.de auf den BrauPi zugegriffen werden.
Randbemerkung: Der Pi1b mag ja ausreichen, schneller geht das ganze aber schon mit dem pi3.

Die Netzwerkkonfiguration unterscheidet sich vom KellerPi:
Da der BrauPi sich auch per Wlan mit einem Netzwerk verbinden können soll muß auch noch dieses eingerichtet werden und aus der einfachen Netzwerkbrücke wird ein Gateway, damit der BrauPi auch im Accesspoint Modus ohne Kabelanschluß eine eigene IP behält:
Das Kernelmodul für den ASUS Netzwerkdongle wird automatisch installiert. Kann man mit lsusb und lsmod überprüfen.
Ifconfig liefert dann den Namen der Netzwerkkarte (wlx107b4458e6e7).
In der wpa_supplicant muß dann das Wlan Netzwerk eingetragen werden, die vollständige Datei sieht dann so aus:

```
country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
      ssid="WLAN-NAME"
      #psk="WLAN-PASSWORT"
      psk=3dac46350727ba3dbcc0c70254d9c5b155fea6406b0f6143d4d1d23d889e7caa
}
```

Der „network“ Eintrag kann als root auch automatisch mit dem Befehl:

```
wpa_passphrase "WLAN-NAME" "WLAN-PASSWORT" >>
/etc/wpa_supplicant/wpa_supplicant.conf
```

erzeugt werden. Der verschlüsseltet psk schlüssel muss sonst von Hand eingetragen werden.

Weiterhin muß in der /etc/dhcpcd.conf noch die Konfiguration der statischen IP-Adressen aktualisiert werden, der entsprechende Abschnitt:

```
# Meine (Example) static IP configuration:
        interface enxb827ebe9fcfe
        static ip_address=192.168.1.91/24
        static routers=192.168.1.11
        static domain_name_servers=192.168.1.11 8.8.8.8

        interface wlx107b4458e6e7
        static ip_address=192.168.1.90/24
        static routers=192.168.1.11
        static domain_name_servers=192.168.1.11 8.8.8.8
```

Nach einem Neustart sollte sich der Raspi mit dem konfigurierten WLan verbinden.
(mal wieder scheiße instabil)

Zur Konfiguration des Accesspoint werden die folgenden Packete
apt-get install rfkill hostap hostap-utils dnsmasq verwendet.

Der Raspi wird einen eigenen DHCP Server betreiben, daher bekommt er sein eigens Subnetz:
/etc/dhcpcd/dhcpcd.conf

```
      interface wlx107b4458e6e7
      static ip_address=192.168.1.90/24
#       static routers=192.168.1.11
#       static domain_name_servers=192.168.1.11 8.8.8.8
```

Der Hostadapter wird ohne Brückenfunktion arbeiten:

```
/etc/hostapd/hostapd.conf  (NICHT MIT TAB EINRÜCKEN FORMATIEREN)
# Bridge-Betrieb
# bridge=br0
# Schnittstelle und Treiber
interface=wlx107b4458e6e7
#driver=RTL8192CU
#WLAN-Konfiguration
ssid=BrauPi
channel=5
```

```
hw_mode=g
ieee80211n=1
ieee80211d=1
country_code=DE
wmm_enabled=1
# WLAN-Verschlüsselung
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=Braumeister
```

Dafür wird eine Network Address Translation eingerichtet:
/etc/dnsmasq.conf
domain-needed
interface=wlx107b4458e6e7
dhcp-range=192.168.1.1,192.168.1.9,255.255.255.0,12h
```
dhcp-option=252,"\n"
```

Der Dienst muß mit:
```
service dnsmasq restart
```
neu gestartet werden.

Und das Forwarden ermöglicht:
```
/etc/sysctl.conf
```
Uncomment the next line to enable packet forwarding for IPv4
```
net.ipv4.ip_forward=1
```

Das ganze kann dann mit:
```
iptables -t nat -A POSTROUTING -j MASQUERADE
hostapd -B /etc/hostapd/hostapd.conf
```
gestartet werden.

Zum automatischen Start wird folgendes Skript angelegt und nach Runlevel 2 verlinkt:
```
nano /etc/init.d/BrauPiAP
#!/bin/sh
# Configure Wifi Access Point.
#
### BEGIN INIT INFO
# Provides: WifiAP
# Required-Start: $remote_fs $syslog $time
# Required-Stop: $remote_fs $syslog $time
# Should-Start: $network $named slapd autofs ypbind nscd nslcd
# Should-Stop: $network $named slapd autofs ypbind nscd nslcd
# Default-Start: 2
# Default-Stop:
# Short-Description: Wifi Access Point configuration
# Description: Sets forwarding, starts hostap, enables NAT in iptables
### END INIT INFO
iptables -t nat -A POSTROUTING -j MASQUERADE
hostapd -B /etc/hostapd/hostapd.conf

chmod +x /etc/init.d/pinpoint
update-rc.d BrauPiAP start 99 2
```
(gibt zwar eine Warnung, funktioniert aber)

Damit sich der Raspi auch mit einem Wlan verbindnen kann müssen Skripte zum reboot erstellt werden.

Eine Änderung der Konfiguration ohne Neustart ist mir nicht gelungen, da die Netzwerkverbingung für beide Karten immer abschmiert wenn die IP Adresse der Wlan Karte geändert wird. Bis dahin war das Skript recht erfolgreich:

```
# Accesspoint stoppen und mit dem WLan aus der wpa_supplicant verbinden
echo "iptables -F"
iptables -F
echo "service dnsmasq stop"
service dnsmasq stop
echo "service hostapd stop"
service hostapd stop  #systemctl stop hostapd alternativ
echo "service dhcpcd restart – mit geänderter conf"
cp /etc/dhcpcd.conf.wl_cl /etc/dhcpcd.conf
#       service dhcpcd force-reload   - möglich aber bringt nichts
#       systemctl daemon-reload       - möglich aber bringt nichts
service dhcpcd restart
cp /etc/dhcpcd.conf.wl_ap /etc/dhcpcd.conf
#ifconfig wlx107b4458e6e7 192.168.1.90
-----Und an dieser Stelle Schmiert das Netzwerk ab
----- Weitere Möglichkeiten, sollte aber nicht erforderlich sein
#ifconfig wlx107b4458e6e7 netmask 255.255.255.0
#/etc/init.d/networking restart
#route add default gw 192.168.1.11
```

Das Skript zum Neustart kopiert eigentlich nur Konfigurationsdateien um und macht einen Neustart:

```
# Accesspoint stoppen und mit dem WLan aus der wpa_supplicant verbinden
# dhcpcd.conf ändern
cp /etc/dhcpcd.conf.wl_cl /etc/dhcpcd.conf
# autostart von ap verhindern
# update-rc.d BrauPiAP disable
rm /etc/rc2.d/S01BrauPiAP
# ap Zustand wieder herstellen wird durch /etc/init.d/BrauPiAPrea beim Reboot im
Runlevel 3 erledigt und löscht sich selber aus der Bootsequenz
# ln -s /etc/init.d/BrauPiAPrea /etc/rc3.d/S01BrauPiAPrea
# Ist dies aktiv aktiviert sich der AP beim neustart wieder, ohne das #
funktioniert der Wlan Client mehr schlecht als recht.
# Neustart
init 6
```

Hierzu müssen parallel die jeweiligen Konfigurationen vorhanden sein (ohne Kommentarzeilen):

```
/etc/init.d/BrauPiAP
iptables -t nat -A POSTROUTING -j MASQUERADE
hostapd -B /etc/hostapd/hostapd.conf

/etc/init.d/BrauPiAPrea
#update-rc.d BrauPiAP enable
ln -s /etc/init.d/BrauPiAP /etc/rc2.d/S01BrauPiAP
cp /etc/dhcpcd.conf.wl_ap /etc/dhcpcd.conf
rm /etc/rc3.d/S01BrauPiAPrea

/etc/dhcpcd.conf.wl_ap (Abschnitt)
# Meine (Example) static IP configuration:
        interface enxb827ebe9fcfe
        static ip_address=192.168.1.91/24
        static routers=192.168.1.11
        static domain_name_servers=192.168.1.11 8.8.8.8

        interface wlx107b4458e6e7
        static ip_address=192.168.2.91/24

/etc/dhcpcd.conf.wl_cl (Abschnitt)
# Meine (Example) static IP configuration:
        interface enxb827ebe9fcfe
```

```
        static ip_address=192.168.1.91/24
        static routers=192.168.1.11
        static domain_name_servers=192.168.1.11 8.8.8.8

        interface wlx107b4458e6e7
        static ip_address=192.168.1.90/24
        static routers=192.168.1.11
        static domain_name_servers=192.168.1.11 8.8.8.8
```

xxx Beim Neustart mit reaktivierung der Konfiguration des Accesspoint steht die Wlan-Verbindung, geht dann aber wieder verloren und der hostapd läuft immer noch, führt nur das Verlinken der Konfiguration nach Runlevel 2 auch zur Ausführung? Falls ja, wie kann man das verhindern.

Zukunft: Mehrere Wlan Konfigurationen mit case – Anweisung und untersch. WPA_supplicant

[Das shellskript war für die vorherige Konfiguration und wird auf dem KellerPi weiter verwendet, ob es hier noch funktioniert wird nicht getestet.
Zum Ein- und Ausschalten (im VZ root, da sowieso root-Rechte erf.):

/root/ap.sh

```
#!/bin/bash
# Netzwerkbrücke ein / aus
#echo $1
case $1 in
        start)
#       echo "Parameter start"
        systemctl start hostapd
        systemctl enable hostapd
        ;;

        stop)
#       echo "Parameter stop"
        systemctl stop hostapd
        ;;
        status)
#       echo "Parameter status"
        systemctl status hostapd
        ;;
        "?")
        echo "Parameter start, stop oder status"
        ;;
esac]
```

3. Zugriff auf die GPIO‘s mit node.js

3.1. Installation des node Webservers

[Installation Node.js (von http://www.iobroker.net/docu/?page_id=5106&lang=de)
Das Packet kann man sich auch mal näher ansehen.
Die alten node & node.js Versionen deinstallieren (bei Raspbian Light nicht notwendig)

```
apt-get --purge remove node
apt-get --purge remove nodejs
apt-get autoremove
reboot
```

als Root über Putty anmelden
Node.js neu installieren für Raspbery 2/3

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install -y build-essential libavahi-compat-libdnssd-dev libudev-dev nodejs
reboot]
```

Node.js neu installieren nur für Raspi 1

```
wget http://node-arm.herokuapp.com/node_archive_armhf.deb
sudo dpkg -i node_archive_armhf.deb
sudo apt-get install build-essential libavahi-compat-libdnssd-dev libudev-dev
reboot
```

`node -v` liefert dann die Versionsnummer wenn die Installation geklappt hat.

und mit dem node Packet manager gpio Tools installieren
`npm install rpi-gpio` (mal als Benutzer pi1 ausgeführt).
Es gibt auch noch andere wie z. Bsp. `pi-gpio`. Was man dann benötigt wird sich später zeigen.

3.2. Erster Start desWebservers

Zu testen die Beispielskripte von „https://www.npmjs.com/package/rpi-gpio“ unter ./node_skripts/gpio abgelegt und ein „Hello World“ erstellt.

```
nano ./node_skripts/hello_world.js

var http = require('http');

// aus der Anfrage die Antwort erzeugen
var anfrageBearbeiter = function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
   res.write('Hier ist der BrauPi an Port 3000');
    res.end(); }

// Server erzeugen
var server = http.createServer(anfrageBearbeiter);
// auf einem Port lauschen
server.listen(3000, function() {
    console.log("Server lauscht auf http://localhost: 3000");
});
```

Scheinen zu laufen.

Zum Start des node Webserever wird das Skript /etc/init.d/node_startup.sh angelegt:

```
node /home/pi1/node_skripts/hello_world.js &
echo "Derzeit nur die Meldung 'Hier ist der BrauPi an Port 3000'"
```

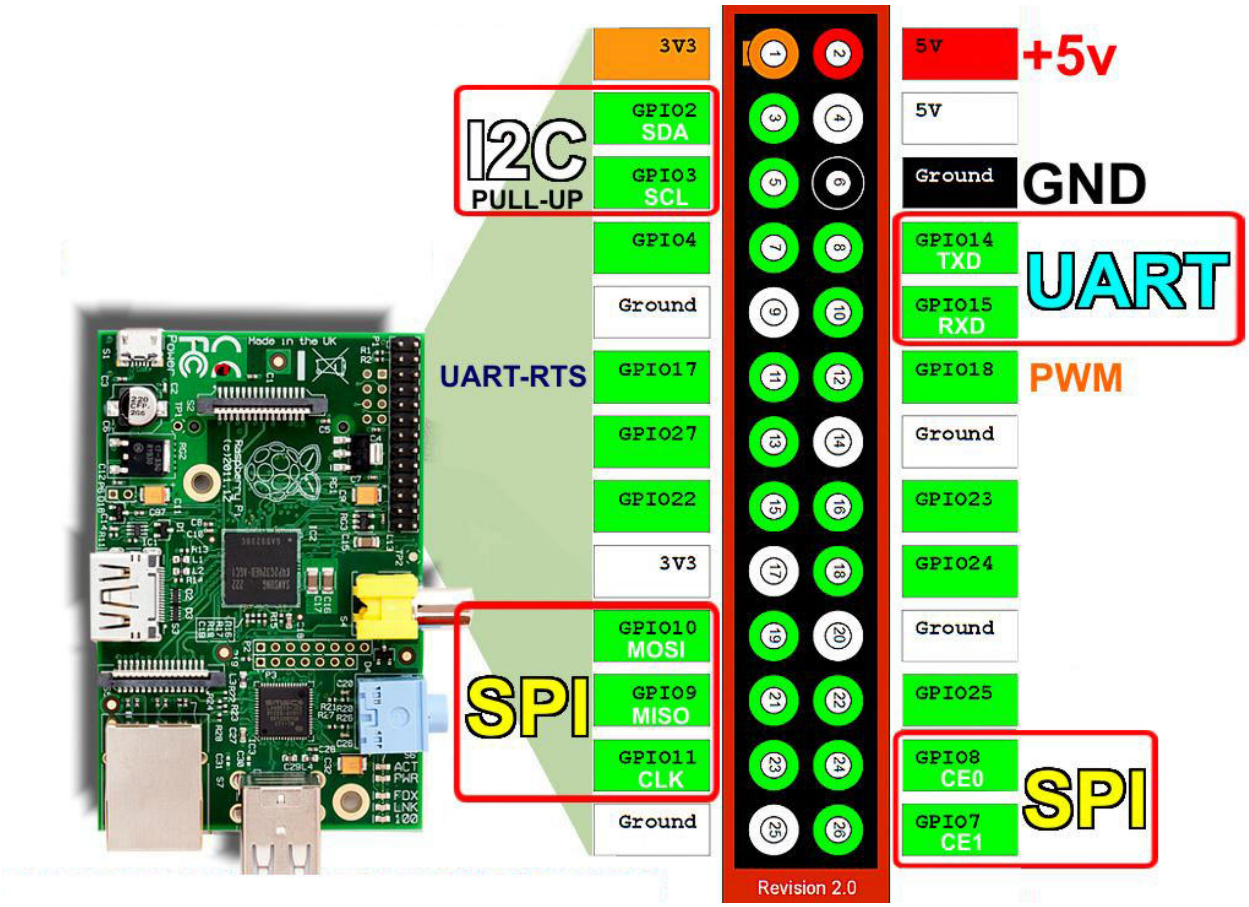
Dies startet den Server im Hintergrund. Wenn er beim Systemstart automatisch aufgerufen werden soll muß der Aufruf „/etc/init.d/node_startup.sh“ noch in /etc/rc.local vor exit 0 noch eingetragen werden. (Derzeit noch auskommentiert.)

Ob ich hierfür ssh und Passwort einrichte weiß ich noch nicht.
Anleitungen:
https://medium.com/netscape/everything-about-creating-an-https-server-using-node-js-2fc5c48a8d4e
https://www.sitepoint.com/http-authentication-in-node-js/

4. Hardwareanbindung

4.1. Schalten der Relais

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)

Hierzu werden aus den Beispielskripte entsprechende Codeschnipsel erzeugt.
Siehe Verzeichnis „Skripe“.

4.2. 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.

Dann 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.

5. Datenaufzeichnung

Als SQL-Datenbank wird mariaDB verwendet.
Installation mit

```
apt-get install mariadb-server
```

Der Zugriff auf die Datenbank erfolgt mit

```
npm install node-mariadb
```

Mit

```
mysql -u root -p
```

wird der Zugriff auf den Datenbankserver geöffnet und der Server für den Benutzer pi0 konfiguriert:

```
grant all on *.* to pi1@localhost identified by '***Passwort***' with
grant option;
```

Als „normaler Benutzer“ kann man sich dann mit mysql -u pi1 -p an der Datenbank anmelden.

Mit
CREATE DATABASE BrauDB;
wird die Datenbank angelegt.
Und eine Tabelle für die Messwerte mit:
CREATE TABLE mess_aktuell (zeit DATETIME NOT NULL UNIQUE, tmessw1 FLOAT, tmessw2 FLOAT, r1ist CHAR(10), r2ist CHAR(10));
erzeugt.

zeit	tmessw1	tmessw2	r1ist	r2ist
DATETIME	FLOAT	FLOAT	CHAR(10)	CHAR(10)

(Info – im Spaltennamen nicht zulässig Anleitung siehe MariaDB Knowledge Base)

Temperaturlogger als Javaskript könnte man auch als Fieberthermometer verwenden mit der Rückmeldung, wenn sich der Meßwert nicht mehr ändert ist die Messung abgeschlossen.

process.irgendwas könnte sehr praktisch zum Steuern des Programms werden. z. Bsp. Mit process.kill eine laufende Regelung abschießen.
Lesezeichen nutzen für weitere Anleitungen.

Derzeit muß der Temperaturlogge als eigene Instanz laufen, da die sleep Anweisung den ganzen node-Prozess anhält.
Alternative Idee:
Schleifen mit php
gpio Zugriff und auf Ereignisse reagieren mit Javascript, Eingriff in die php Schleifen über Rückmeldungen; z. Bsp While Schleife mit Bedingung mach weiter so lange „alles in Ordnung“ und Skripte können alles in Ordung auf „false“ setzen.
Datenbankzugriff mit php ha‘m wir schon.

6. Temperaturregelung

Die Temperaturregelung erfolgt als simple Ein-Aus Regelung mit einstellbarer Hysterese.

7. Funktionalitäten

8. Die Temperaturprogramme zum Brauen sowie die Meßdaten und Relaiszustände werden als relationale Datenbank (MariaDB?) gespeichert.

Bei allen Datenbanken einen nicht angezeigten Primärschlüssel verwenden.

(Idee: Sollte ein Pi mit wlan verwendet werden könnte er auch einen Accesspoint zu seiner Konfiguration zur Verfügung stellen.)

9. Die Oberflächen zur Bedienung

Teil 1:
Erstellen / laden / speichern / ändern von Brauprogrammen.
(zur Zeit: Temperatur x für soundsoviel Minuten, Wert für Hysterese pro Intervall sollte möglich sein)

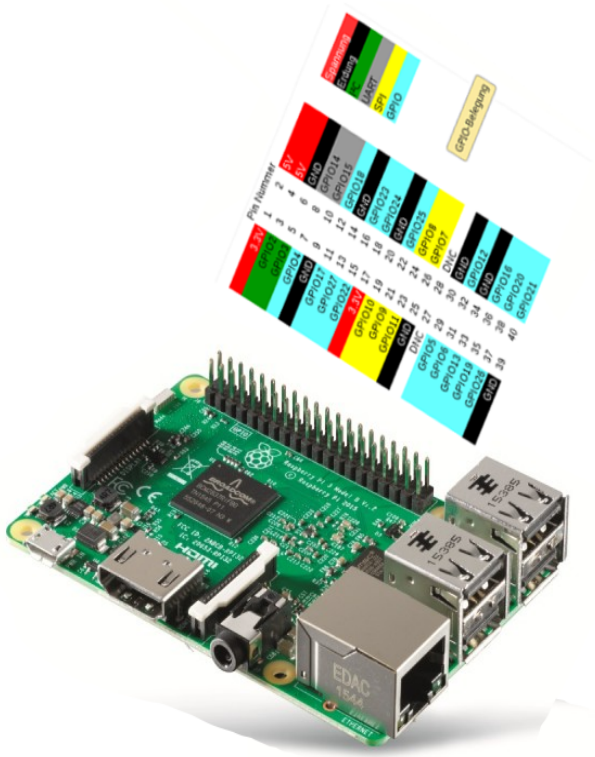
Teil 2:
Starten / stoppen von Programmen mit Trendanzeige von Temperaturen / Sollwerten und Relaiszuständen. Hierbei sollte (wofür auch immer) der Sollwert für die Temperatur und auch die Relais selber auf „Hand“ genommen werden können. (Auch während laufender Programme).

Beispieltext für die Formatierung:

```
Listen 80
<IfModule ssl_module>
       Listen 443
</IfModule>
<IfModule mod_gnutls.c>
        Listen 443
</IfModule>
```

(Warnungen)
`passwd testuser`
`funktionert nicht`

Die GPIO Schnittstelle (vom3er)



WiringPi	Pin-Namen	Pin	Pin	Pin-Namen	WiringPi

–	+ 3,3 V	**1**	**2**	+ 5 V	–
8	(SDA1) GPIO 2	**3**	**4**	+ 5 V	–
9	(SCL1) GPIO 3	**5**	**6**	GND	–
7	(GPIO_GCLK) GPIO 4	**7**	**8**	GPIO 14 (TXD0)	15
–	GND	**9**	**10**	GPIO 15 (RXD0)	16
0	(GPIO_GEN0) GPIO 17	**11**	**12**	GPIO 18 (GPIO_GEN1)	1
2	(GPIO_GEN2) GPIO 27	**13**	**14**	GND	–
3	(GPIO_GEN3) GPIO 22	**15**	**16**	GPIO 23 (GPIO_GEN4)	4
–	+ 3,3 V	**17**	**18**	GPIO 24 (GPIO_GEN5)	5
12	(SPI_MOSI) GPIO 10	**19**	**20**	GND	–
13	(SPI_MISO) GPIO 9	**21**	**22**	GPIO 25 (GPIO_GEN6)	6
14	(SPI_SLCK) GPIO 11	**23**	**24**	GPIO 8 (SPI_CE0_N)	10
–	GND	**25**	**26**	GPIO 7 (SPI_CE1_N)	11
30	(nur für I2C) ID_SD	**27**	**28**	ID_SC (nur für I2C)	31
21	GPIO 5	**29**	**30**	GND	
22	GPIO 6	**31**	**32**	GPIO 12	26
23	GPIO 13	**33**	**34**	GND	
24	GPIO 19	**35**	**36**	GPIO 16	27
25	GPIO 26	**37**	**38**	GPIO 20	28
	GND	**39**	**40**	GPIO 21	29

10.8 Fertig

Eingebaut sieht das ganze dann so aus:

???? wird sich zeigen….