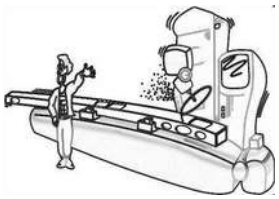
-



9. Jul. 2021

Slot Finder

Ein Raspberry Pi auf einer 8-fach Relaiskarte, etwas Software zur Datenspeicherung und Visualisierung und Python-Skripte zur Auswertung einer Konfigurationsdatei und Einlesen von stündlichen Strompreisen über die aWATTar API.

aWATTar bietet einen Stromtarif mit stündlichen Preisen an. Die Preise orientieren sich am Börsenpreis plus diversen Aufschlägen. Die Herausforderung besteht darin, die jeden Tag anders liegenden Preistäler für stromhungrige Geräte zu finden und über Relais dann Freigaben zu schaffen.

Eine Konfigurationsdatei erlaubt es beliebige Aufgaben (Tasks) zu definieren. Darin enthalten sind Angaben, ob der Task gerade benutzt werden soll, wie lange er aktiv sein soll und in welchem Zeitraum ein Preistal gesucht werden soll. Jedem Task ist eine Kommandodatei zugeordnet für das Einschalten und das Ausschalten.

Das Python-Skript schreibt eine ToDo-Liste für das Betriebssystem (Crontab) mit den Uhrzeiten für den Aufruf der Kommandodateien. Diese Crontab wird täglich neu erstellt.

Der Aufruf der Python-Skripte zur Preisabfrage und zur Slot-Findung stehen auch in einer Crontab und werden jeden Tag gestartet. Die Preisabfrage startet um 23.00 Uhr und die Slot-Findung um 23.50 Uhr.

Zusammengestellt:

Peter Fürle, pf@nc-x.com

31.05.2021

Inhaltsverzeichnis

Verwendete Hardware....3
Raspberry Pi....3
Relais-Karte....3
verwendete Software....4
InfluxDB....4
Grafana....4
Python....5
Inbetriebnahme....6
Start des Raspberry....6
Wartung und Pflege....6
Manuelle Eingriffe....7
System-Crontab....7
User-Crontab....7
die Konfigurationsdatei....8
Das Dashboard....10
Der untere Teil des Dashboards:....11
Einstellungen am Dashboard....12
Hilfsprogramme....13
Einzel-Relais unabhängig ein- ausschalten....14
Beispiele....15
Darstellung Relaiskarte zusammengebaut....16
Zusätzliche Satelliten-Relais per Wlan....17
Grafana Sat-Relais: Schaltzustände....18
Grafana Sat-Relais: Temp-Sensor....19
Aufbau Satelliten-Relais....20
Manuelle Eingriffsmöglichkeiten....21
Sat-Relais: Taster-Modus....22
Raspberry kommuniziert an Sat-Relais....23
Relais mit NodeMCU verbinden....24
Erweiterungsmöglichkeiten....25

Verwendete Hardware

Raspberry Pi

Einplatinencomputer Raspberry Pi 3 Modell B Plus (B +) 1,4 GHz 64-Bit-Quad-Core-Prozessor mit 1 GB LPDDR2 SDRAM

Dual-Band 2,4 GHz und 5 GHz IEEE 802.11.b / g / n / AC WLAN, Bluetooth 4.2, BLE. Ethernet über USB 2.0 (maximaler Durchsatz 300 Mbps)

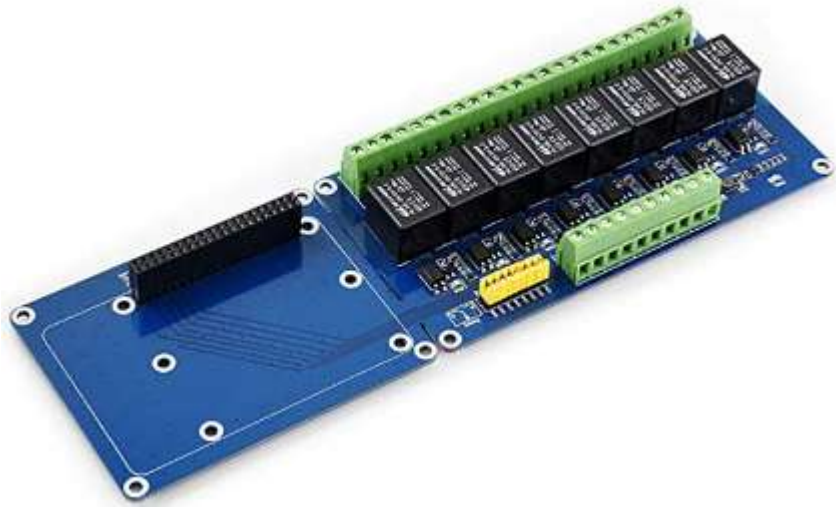
Netzadapter mit praktischem Ein- / Ausschalter 5V 2,5A-Strom

Relais-Karte

Erweiterungskarte für den Standard 40PIN GPIO Header.

Hochwertiger Relais mit bis zu 5A bei 250V AC oder 5A 30V DC. Isolation über Optokoppler, galvanisch getrennt.

Hutschienenmontage, Onboard LEDs zeigen den Relais-Status



verwendete Software

InfluxDB

Die Live-Daten des Discovergy-Zählers und die Preise je Stunde werden von einem Python-Script abgeholt, aufbereitet und in eine Influx-Datenbank geschrieben. Somit lassen sich die Daten anschließend in beliebigen Zeiträumen auswerten.

Das können die letzen 3 Stunden, oder auch der Dienstag vor 4 Wochen sein.

- Die verwendete Influx Datenbank heißt „`awattar`“,
- hat den Benutzer „`grafana`“
- und das Passwort „`herbi`“

Grafana

Grafana ist der Name einer OpenSource Software, mit der sich Daten aus verschiedenen Datenquellen optisch aufbereiten und in interaktiven, dynamischen Dashboards visualisieren lassen. Die Software steht für unterschiedliche Plattformen und Betriebssysteme zur Verfügung. Die Daten sind in vielen unterschiedlichen Chart und Graphen-Typen darstellbar. Ebenfalls unterstützt wird die Alarmierung auf Basis der Daten.

Wesentliche Komponente der Software ist ein HTTP(S)-Server, der eine grafische Benutzeroberfläche bereitstellt. Die Anbindung der verschiedenen Datenquellen und weitere Funktionalitäten sind über ein Plug-in-System realisiert.

Häufig kommt Grafana für Monitoring-Aufgaben und die Visualisierung von Messdaten um Einsatz. Die Software arbeitet mit zahlreichen Zeitreihen-Datenbanken wie InfluxDB und anderen zusammen.

Grafana wurde 2014 entwickelt, ist sehr beliebt und hat eine weltweit aktive Community. Zahlreiche Unternehmen und Plattformen wie PayPal, eBay, Wikipedia, Intel, Vimeo oder Booking.com nutzen Grafana. Die aktuelle Version der unter Apache-2.0-Lizenz stehenden Software ist Grafana 7.5.3 (Stand April 2021).

Grafana wird aufgerufen mit dem Hostnamen des Raspberry Pi und dem Port 3000.

`http://pirelais:3000`

Das Anmelden erfolgt mit dem

- Benutzer „`admin`“ und dem
- Passwort „`herbi`“

Python

Bei Python handelt es sich um eine Programmiersprache mit einer klaren Syntax und guten Lesbarkeit. Sie gilt als leicht zu erlernen und ist in den gängigen Betriebssystemen interpretierbar. Der Name leitet sich von „Monty Python's Flying Circus" ab.

Der Python-Quell-Code ist unter der Python-Software-Foundation-License frei verfügbar. Im Netz existiert eine breite Anhängerschaft und eine große Community.

Python genießt einen Ruf als einfache und saubere Programmiersprache mit klarer Struktur. Ihr Programmcode ist intuitiv nutzbar und gleichzeitig leicht lesbar. Trotz der Einfachheit bietet Python eine gute Skalierbarkeit und ist für komplexe Softwareprojekte einsetzbar.

Aufgrund der ausdrucksstarken, minimalistischen Syntax sind Anwendungen mit wenigen Codezeilen und geringer Anfälligkeit für Programmierfehler realisierbar. Um für Einfachheit und Übersichtlichkeit zu sorgen, kommt Python mit sehr wenigen Schlüsselwörtern aus und verwendet Einrückungen als Strukturierungselemente.

Für die gängigen Betriebssysteme ist Python frei verfügbar. In vielen Linux-Distributionen gehört die Programmiersprache zur Standardausstattung.

Inbetriebnahme

Es handelt sich um keine Fertiglösung sondern um einen Bausatz.

Der Raspberry Pi muss mit der Oberseite auf die Relaiskarte gesetzt werden. Die 40polige Stiftleiste passt in die 40polige Buchenleiste.

Die Relaiskarte wird auf eine Hutschiene gesteckt. Davor werden die Seitenteile mit den beiliegenden Schrauben fixiert.

Die SD-Karte muss in den Slot gesteckt werden, so, dass die Kontakte zur Platinenseite zeigen.

Ein Ethernet-Kabel wird vom Switch zum Port am Raspberry geführt und eingesteckt.

Die Relais werden von der Elektrofachkraft verdrahtet.

Das Netzkabel wird eingesteckt und eingeschaltet.

Start des Raspberry

Wird der Raspberry Pi eingeschaltet, werden die Datenbank und der Webserver gestartet. Die Relaisplatine wird initialisiert. Dabei werden die Relais kurz eingeschaltet und sofort wieder ausgeschaltet. Der Raspberry startet immer mit ausgeschalteten Relais.

Um 23.00 Uhr holt der Raspberry die Preise für den nächsten Tag und trägt diese in die Influx-Datenbank ein.

Um 23:50 Uhr liest der Raspberry die Konfigurationsdatei ein und sucht für jeden Task den günstigsten Slot. Er schreibt eine User-Crontab für den nächsten Tag.

Alle 2 Minuten fragt der Raspberry die Zählerstände bei Discovergy ab.

Wartung und Pflege

Es ist keine weitere Wartung notwendig. Es gilt zu beachten, dass die SD-Karten bauartbedingt eine begrenzte Lebensdauer haben. Es empfiehlt sich davon eine Kopie anzulegen. Dies kann am PC geschehen.

Updates stellt die Raspberry Pi Organisation kostenfrei zur Verfügung. Dazu muss ein Prozess am Raspberry gestartet werden. Dazu finden sich zahlreiche Artikel im Internet oder wir empfehlen das Buch „Raspberry Pi“ von Kofler, Kühnast und Scherbeck.

Manuelle Eingriffe

Unabhängig davon, dass die Prozesse automatisch ablaufen, bestehen Eingriffsmöglichkeiten.

Dazu ist es notwendig auf die Betriebssystemebene des Raspberry Pi zu kommen. Dort gibt es eine Textkonsole in der Kommandos eingegeben werden können.

Am Raspberry ist der SSH-Zugang freigeschaltet. Unter Linux erreicht man diesen mit:

```
ssh pi@pirelais -o ForwardX11=yes
```

Der Benutzer „`pi`“ ist im Aufruf schon angegeben, das Passwort lautet „`herbi`“

System-Crontab

In der System-Crontab steht der Aufruf der beiden Hauptskripte und die Abfrage des Discovergy-Zähler-Portals alle 2 Minuten. Eingabe: `sudo nano /etc/crontab`

```
#
*/2 *   * * *   pi      /usr/bin/python3 /home/pi/discovergy.py
0 23   * * *   pi      /usr/bin/python3 /home/pi/scripte/slot_influx.py
50 23  * * *   pi      /usr/bin/python3 /home/pi/scripte/slot.py
```

User-Crontab

Die User-Crontab wird von `SLOT.PY` täglich geschrieben. In ihr befindet sich der Aufruf der Kommandoprogramme die die Relais ein- und ausschalten. Eingabe: `crontab -l` oder -e

```
pi@pirelais:~ $ crontab -l
# hier werden automatisch die Relais ein und ausgeschaltet
# Einträge sollten kurz vor Mitternacht vorgenommen werden.
#


0 14 * * * /bin/bash /home/pi/awattar/starte_R1.sh  >> /home/pi/awattar/picron.log # automatic Relais aWATTar Tag:30
0 16 * * * /bin/bash /home/pi/awattar/beende_R1.sh  >> /home/pi/awattar/picron.log # automatic Relais aWATTar Tag:30
0 16 * * * /bin/bash /home/pi/awattar/starte_R1.sh  >> /home/pi/awattar/picron.log # automatic Relais aWATTar Tag:30
54 16 * * * /bin/bash /home/pi/awattar/beende_R1.sh  >> /home/pi/awattar/picron.log # automatic Relais aWATTar Tag:30
0 15 * * * /bin/bash /home/pi/awattar/starte_R2.sh  >> /home/pi/awattar/picron.log # automatic Relais aWATTar Tag:30
0 17 * * * /bin/bash /home/pi/awattar/beende_R2.sh  >> /home/pi/awattar/picron.log # automatic Relais aWATTar Tag:30
0 14 * * * /bin/bash /home/pi/awattar/starte_R3.sh  >> /home/pi/awattar/picron.log # automatic Relais aWATTar Tag:30
```

Hinweis: War der Raspberry ein paar Tage ausser Betrieb ist die Crontab manuell zu löschen. Dazu `crontab -e` aufrufen und jede Zeile mit dem Kommentar `# automatic Relais…` durch den Tastendruck `STRG-K` löschen. Im Automatic-Betrieb wird nur der Vortag gelöscht.

die Konfigurationsdatei

Eingabe: `nano scripte/awattar_scheduler.conf`

```
 GNU nano 3.2                                  scripte/awattar_scheduler.conf

[General]
authorization =
#
#Muster: /bin/bash /home/pi/awattar/starte_R1.sh  >> /home/pi/awattar/picron.log
#Startzeit+Periode könnte Ladebeginn sein ! Danach kommt die Dauer dazu.


[Task_BWWP]
enable = true
info = Relais 1, Pin 5
starttime = 0
periode = 12
duration = 2
commando_start = /bin/bash /home/pi/awattar/starte_R1.sh  >> /home/pi/awattar/picron.log
commando_stop = /bin/bash /home/pi/awattar/beende_R1.sh  >> /home/pi/awattar/picron.log
```

Jeder Task beginnt mit dem Schlüsselwort `Task_` und muss in [ ] geschrieben sein. Es können beliebig viele angelegt werden.

Im Beispiel ist der `Task_BWWP` für die Brauchwasser-Wärmepumpe an Relais 1 gedacht.

Mit `enable = true` wird der Task berücksichtigt. Wird hier `false` eingetragen, wird für diesen Task kein Zeitslot gesucht.

`starttime = 0` und `periode = 12` sagt, dass zwischen 0 Uhr und 12 Uhr ein Zeitslot gesucht wird.

`duration = 2` bestimmt den Zeitslot mit der Dauer von 2 Stunden. Dieser Zeitslot kann zu beliebiger voller Stunde zwischen 0 und 12 Uhr beginnen. Achtung, er kann auch um 12 Uhr beginnen und dann bis 14.00 Uhr gehen.

`commando_start` und `commando_stop` werden so in die User-Crontab übernommen. Diese sind voreingestellt für die 8 Relais.

Für die BWWP wurde ein weiterer Task angelegt.

```
[Task_2BWWP]
info = Relais 1, Pin 5
enable = true
starttime = 14
periode = 10
duration = 0.9
commando_start = /bin/bash /home/pi/awattar/starte_R1.sh  >> /home/pi/awattar/picron.log
commando_stop = /bin/bash /home/pi/awattar/beende_R1.sh  >> /home/pi/awattar/picron.log

[Task_WP]
info = Relais 2 Pin 6
enable = true
starttime = 0
periode = 20
duration = 2
commando_start =  /bin/bash /home/pi/awattar/starte_R2.sh  >> /home/pi/awattar/picron.log
commando_stop = /bin/bash /home/pi/awattar/beende_R2.sh  >> /home/pi/awattar/picron.log
```

Das ist ein Beispiel um ein Relais an einem Tag zweimal einschalten zu lassen. Der zweite Eintrag startet um 14 Uhr und erlaubt eine 10-Stunden Periode mit einer 55 Minuten Dauer. Er könnte also um 14 Uhr nochmal für 55 Minuten laufen, oder um 14+10 = 24 Uhr für 55 Minuten starten.

ACHTUNG: Das würde nicht funktionieren, da um 23.50 Uhr schon die Einträge gelöscht werden um die Einträge für den nächsten Tag zu schreiben. Besser die periode auf 8 setzen, dann geht der Betrieb maximal bis 14+8+0.9 = kurz vor 23 Uhr.

Im obigen Beispiel ist der `Task_WP` als Heizungswärmepumpe angelegt.

Zwischen 0 und 20+2 sucht diese Konfiguration ein 2 Stunden Zeitslot für die Heizungswärmepumpe an Relais 2

Am Relais 3 wird der Geschirrspüler erwartet. Alle anderen Tasks sind mit `Task_R4` bis `Task_R8` bezeichnet und in der Konfigurationsdatei angelegt.

Das Dashboard

Mit Grafana werden Dashboards eingerichtet. Eine angepasste Version ist schon auf der SD-Karte installiert.

Dieses Dashboard kann in jedem aktuellen Browser aufgerufen werden. Dazu wird eingegeben:

http://pirelais:3000

Dieser Aufruf kann am Handy, am Tablet oder am PC erfolgen. Ich benutze überall den Chrome-Browser dazu.

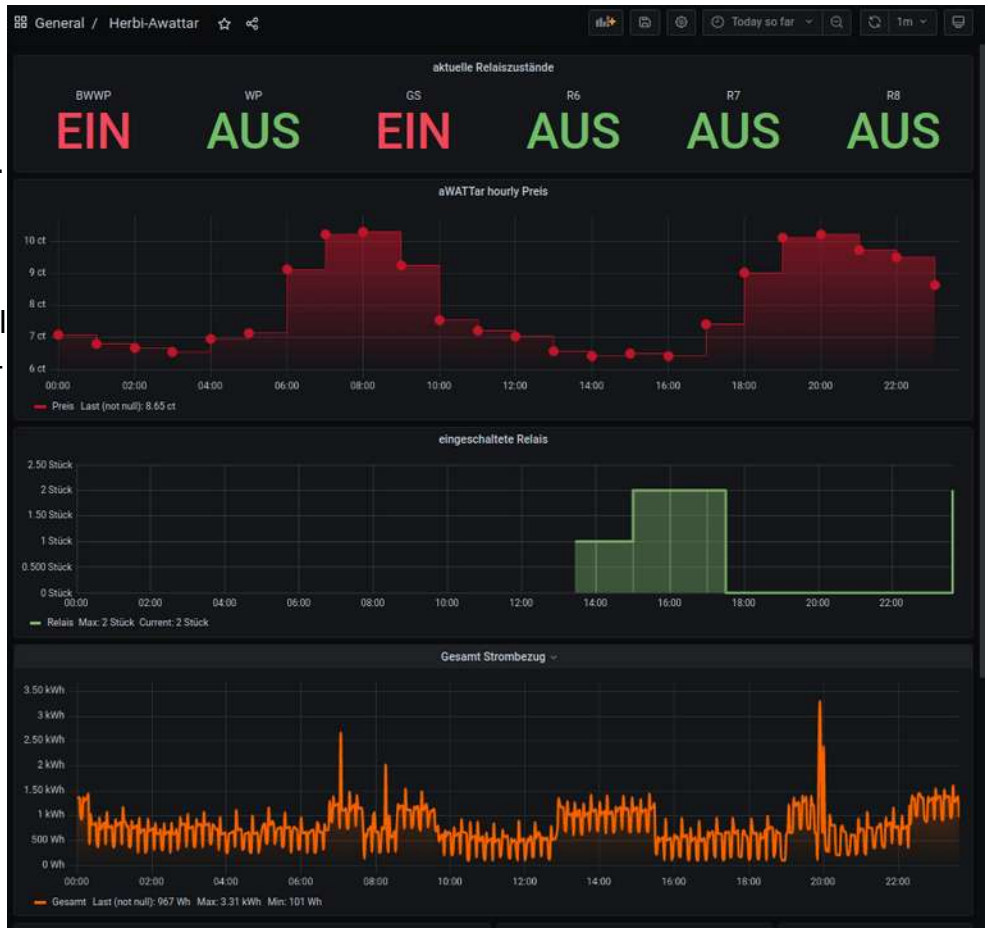
Ganz oben werden die Zustände der Relais angezeigt. Hier sind die BWWP und der Geschirr-spüler an.

Darunter in Rot die stündlichen Preise von aWATTar.

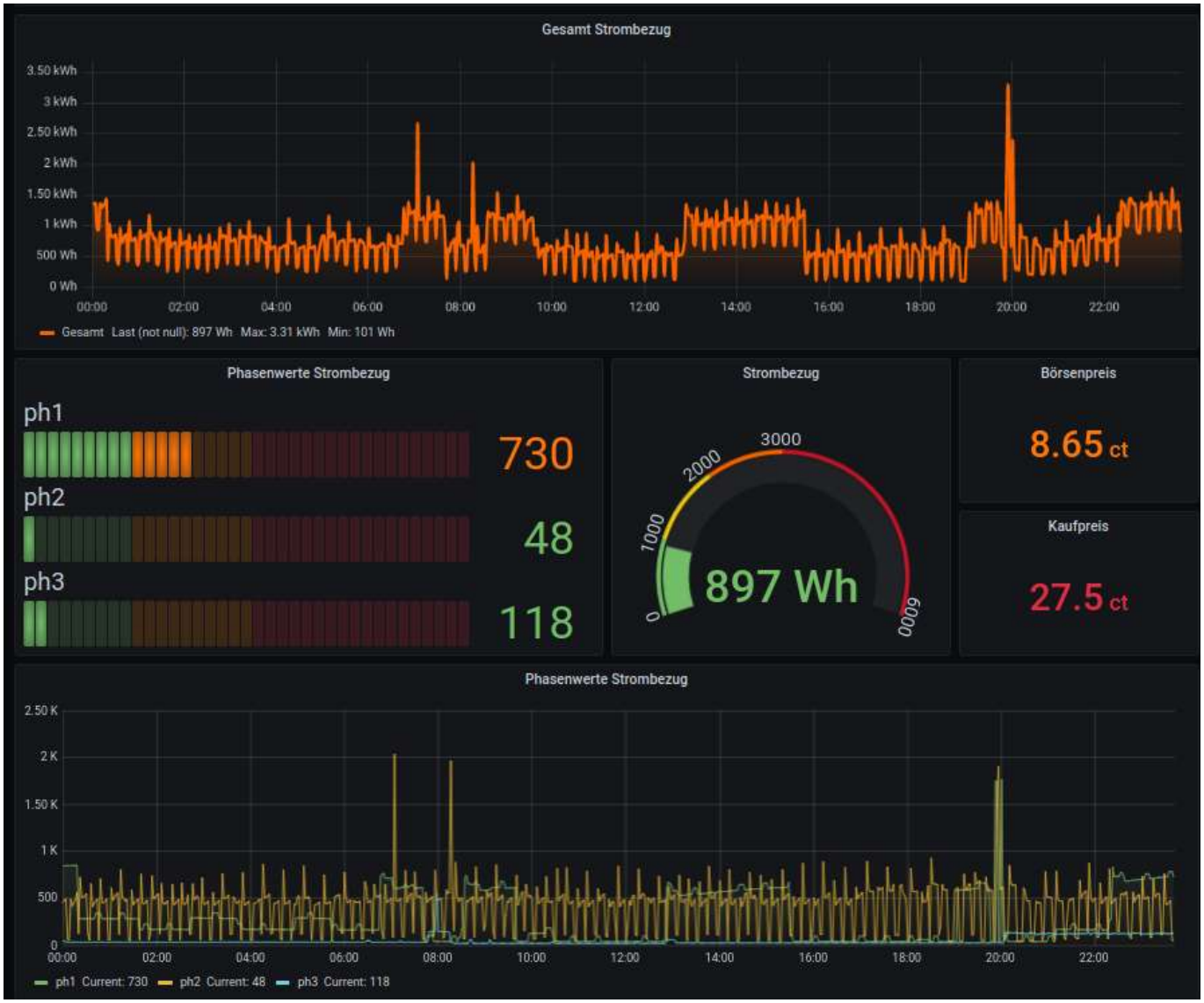
Als grüner Block die Anzahl der zu einem Zeitpunkt eingeschalteten Relais.

Im Preistief waren 2 Relais eingeschaltet.

In orange der Gesamt-Stombezug als Kurve.



Der untere Teil des Dashboards:



Die LCD-Anzeigen visualisieren die Last auf den einzelnen Phasen. Daneben als Rundinstrument der aktuelle Gesamtverbrauch über alle 3 Phasen.

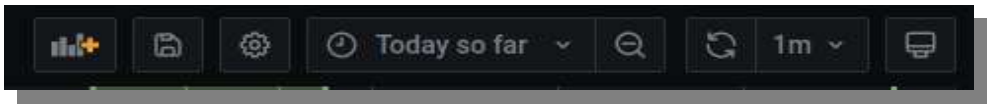
Ganz rechts der Börsenpreis am Ende der angezeigten Periode. Darunter in rot der Kaufpreis Brutto errechnet aus Börsenpreis plus 18.9 ct/kWh

Darunter ein Graph der die Auslastung der einzelnen Phasen zeigt.

Ziel des Slotfinders ist, dass immer im Preis-Tal der höchste Verbrauch stattfindet.

Einstellungen am Dashboard

Mit den Icons rechts oben kann das Dashboard angepasst werden.



Das Plus ermöglicht das Anlegen neuer Graphen,

die Diskette ermöglicht das Speichern,

das Zahnrad erlaubt weitere Einstellungen,

Die Uhr ermöglicht die Einstellung bestimmter Zeitbereiche. Dies ist sicher der wichtigste Punkt.

Die beiden Halbkreise aktualisieren die Anzeige

1m steht für automatisches aktualisieren jede Minute.

Der Monitor ganz rechts erlaubt die Anzeige unter anderen Bedingungen.

Diese Einstellung ist eine gängige Version. Weiters interessant sind andere Zeiträume wie zum Beispiel die letzen 2 Stunden, gestern, letzte Woche und so weiter.

Hilfsprogramme

auf der Terminalebene des Raspberry können vorbereitete Hilfsprogramme gestartet werden.

`alleRelais_an.sh`, `alleRelais_aus.sh`, `checkrelais.sh`, `vorbereitenRelais.sh`

Dies sind sogenannte Bash-Skripte und werden durch ein vorgestelltes bash gestartet.

`bash alleRelais_an.sh` schält alle Relais auf der Karte ein.

`bash checkrelais.sh` zeigt den Zustand der Relais an:

```
pi@pirelais:~ $ bash checkrelais.sh
an: Brauchwasser-Wärmepumpe
Wärmepumpe Heizung ist aus
an: Geschirrspüler
R4 BCM 16 ist aus
R5 BCM 19 ist aus
R6 BCM 20 ist aus
R7 BCM 21 ist aus
R8 BCM 26 ist aus
0 1 0 1 1 1 1 1
```

Ein einzelnes Relais kann auch manuell über ein Kommando ein- oder ausgeschaltet werden. Dazu ist zu beachten, dass die Relaiskarte nicht die Nummern 1-8 verwendet, sondern die BCM-Codes. Hierzu dient diese Tabelle:

`gpio -g write 13 1`

schält das Relais 3 (BCM13) AUS, denn die 1 steht für AUS und die 0 für EIN.

`gpio -g write 20 0`

schält das Relais 6 EIN.

Allerdings erfolgt keine Meldung an die Datenbank und somit auch keine Anzeige im Dashboard !

RPi Relay Board (B)

INTERFACE

Channel	RPI pin	wiringPi	BCM	Description
CH1	29	P21	5	Channel 1
CH2	31	P22	6	Channel 2
CH3	33	P23	13	Channel 3
CH4	36	P27	16	Channel 4
CH5	35	P24	19	Channel 5
CH6	38	P28	20	Channel 6
CH7	40	P29	21	Channel 7
CH8	37	P25	26	Channel 8

【Note】The silk printing on PCB are BCM2835 codes

Einzel-Relais unabhängig ein- ausschalten

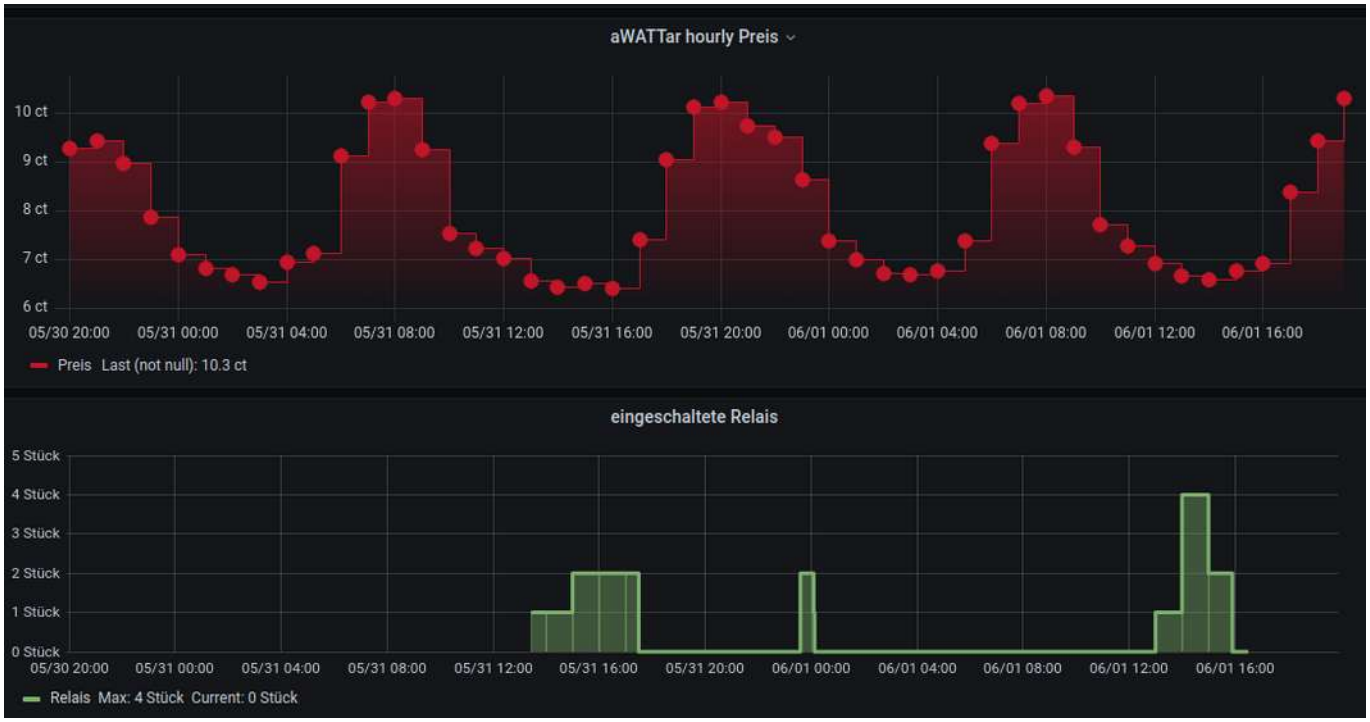
Die bessere Methode sind die bash-Skripte die sich im Verzeichnis awattar befinden. Dort liegen Skripte bereit die alle Aufgaben übernehmen. Relais 1 bis Relais 8 können gestartet oder beendet werden.

```
pi@pirelais:~ $ bash awattar/starte_R3.sh
--->  Di 1. Jun 00:07:59 CEST 2021  Start Relais 3 Geschirr
pi@pirelais:~ $ bash awattar/beende_R3.sh
<---  Di 1. Jun 00:08:03 CEST 2021  Stopp Relais 3 Geschirr
pi@pirelais:~ $ bash awattar/beende_R1.sh
<---  Di 1. Jun 00:08:07 CEST 2021  Stopp Relais 1 BWWP
pi@pirelais:~ $
```

Diese Befehle schalten das Relais, informieren die Datenbank und lassen den Schaltzustand dann auch auf dem Dashboard erscheinen.

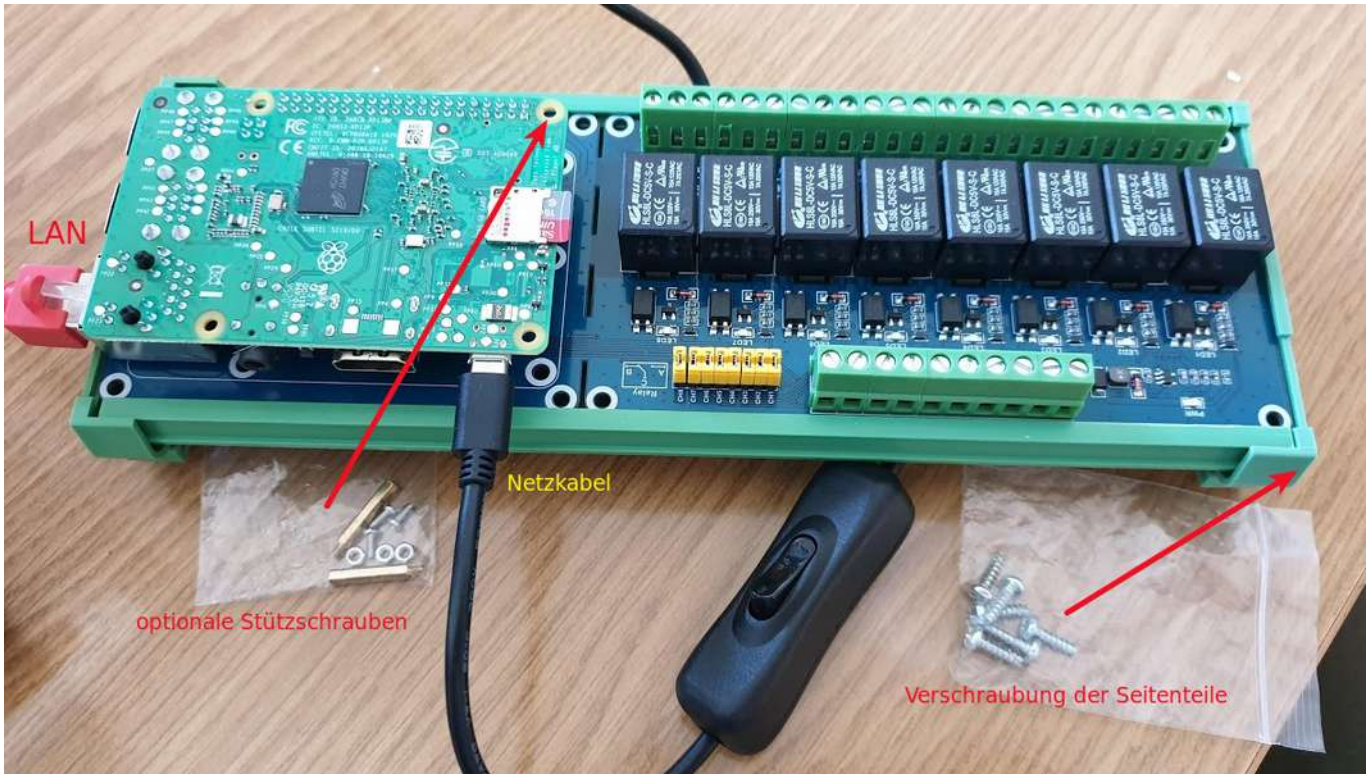
Beispiele

4 Relais an im Preistal



Darstellung der letzten 2 Tage

Darstellung Relaiskarte zusammengebaut



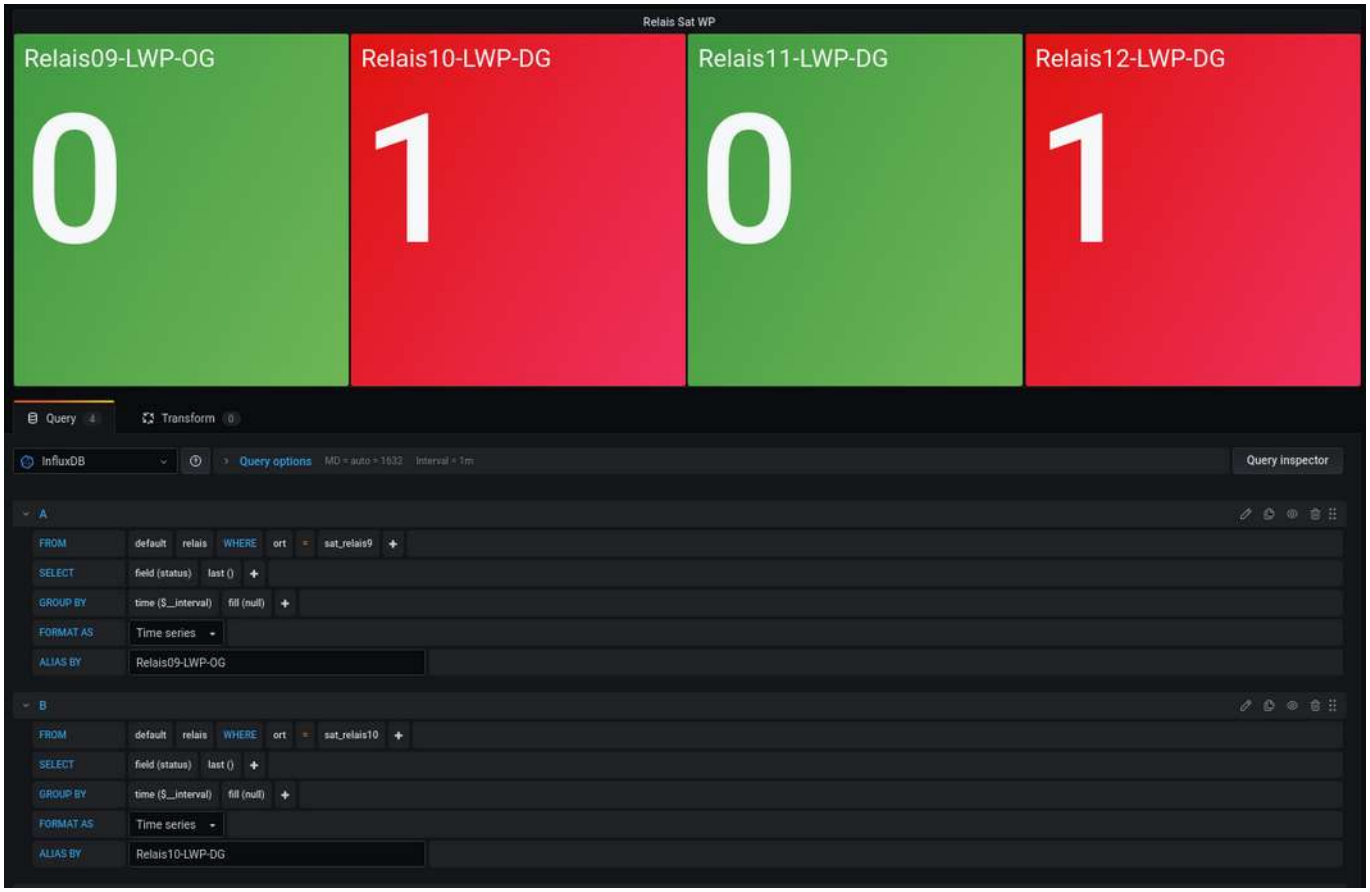
Zusätzliche Satelliten-Relais per Wlan

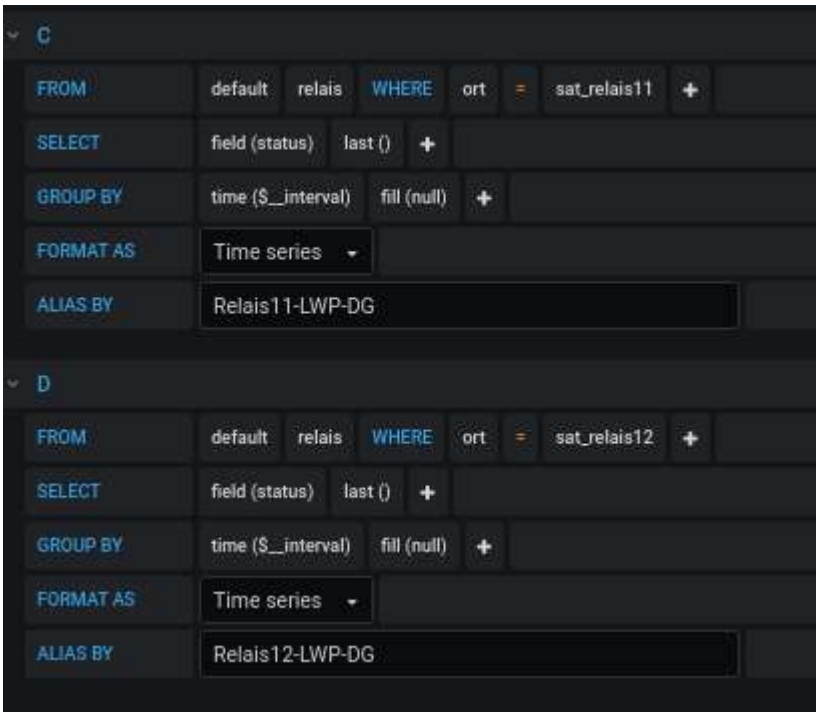
Es gibt Fälle, in denen es aufwendig ist, Leitungen vom Relaisboard bis zum Verbraucher zu ziehen. In unserem Fall sollten Wärmepumpen ein definierter Vorwiderstand untergeschoben werden der 10° entspricht. Im Normalbetrieb steht die BWWP auf 42°C und im günstigen Slot von aWATTar macht sie dann 4h lang 52°C.

Dafür sind die Satelliten-Relais ideal. Sie befinden sich im Versorgungsbereich des WLANs möglichst nah am Verbraucher. Der Raspberry Pi auf der Relaisplatine kann über ein Netzwerkkommando ein Signal an einen Microprozessor senden. Es kommt ein NodeMCU Lolin V3 zum Einsatz. An diesen ist ein 2-fach-Relaisboard angeschlossen. Der NodeMCu kann noch einen Schalter handhaben und einen oder mehrere Temperaturfühler.

Die Erweiterung des Grafana-Dashboards zeigt dann die Satelliten-Relais als Relais 09-12 mit den jeweiligen Zuständen an. Darüber den Verlauf der Temperatur wie vom DS18B20 gemessen.

Grafana Sat-Relais: Schaltzustände

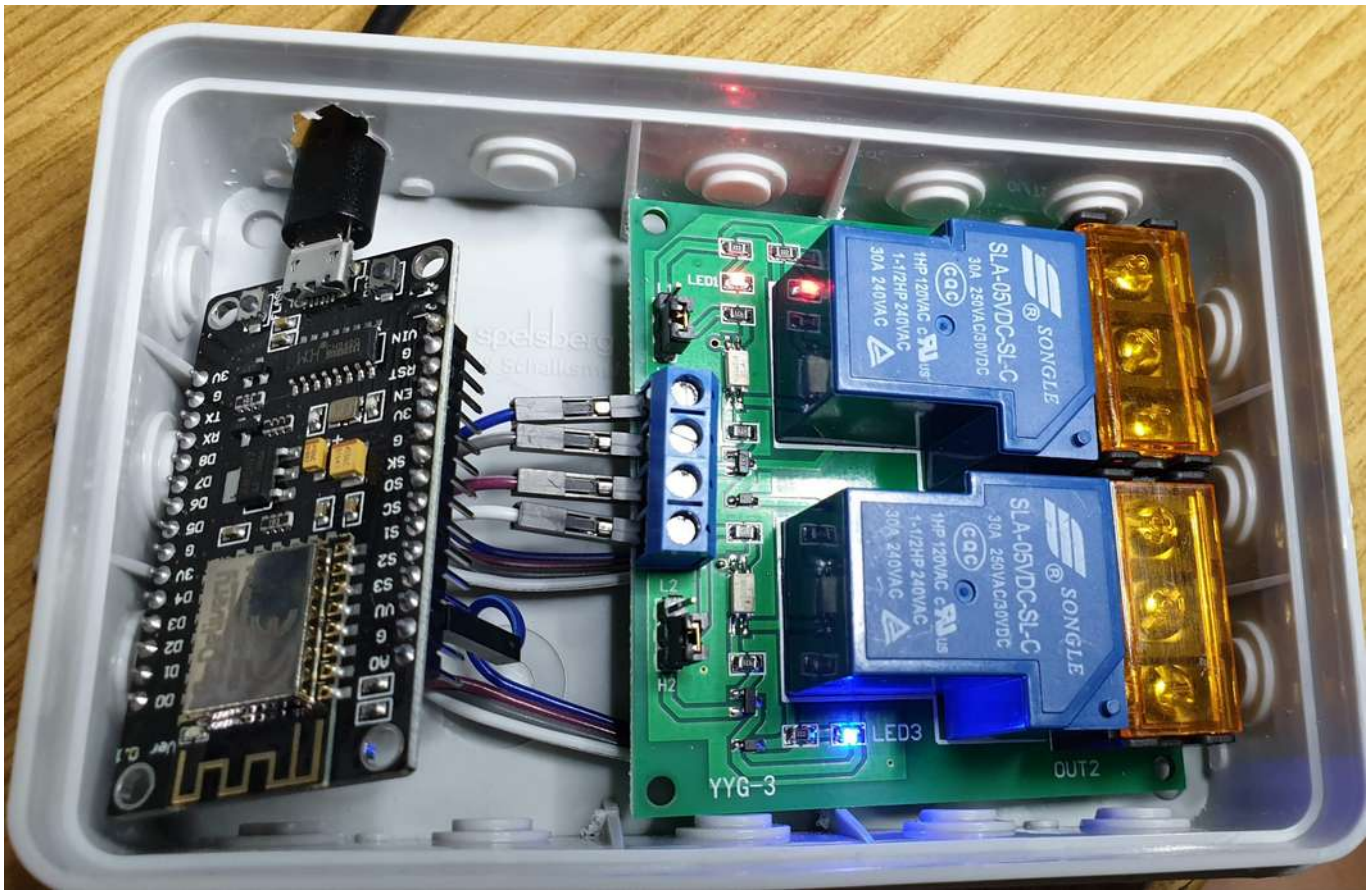




Grafana Sat-Relais: Temp-Sensor

Aufbau Satelliten-Relais

Das Relais-Board und der NodemCU sind in Verteilerdosen untergebracht.



Links der NodeMCU ESP8266 Lolin V3 und rechts die beiden Relais auf einer Platine

Im einfachsten Fall reichen eine USB-Stromversorgung und 4 Kabel zum Relaisboard. Eines für VCC und GND und die beiden Steuerkabel für Relais 1 und Relais 2.

Das Lolin V3 eignet sich deshalb so gut, da es eine 5V Spannung ausgeben kann. Diese reicht aus um die Relaisplatine zu versorgen. Die beiden Steuerleitungen mit 3.3V reichen um zuverlässig die Relais zu schalten.

Manuelle Eingriffsmöglichkeiten

Die Satelliten-Relais ermöglichen die Ansteuerung auch über ein vom NodeMCU bereitgestellte HTML-Seite. Ein Button steuert das erste Relais. Dazu wird die IP-Adresse des jeweiligen Satelliten Relais eingegeben. Sie lassen sich leicht über eine Fritzbox! Identifizieren, dort reicht dann ein Doppelklick:

Für einen beständigen Betrieb sollte den Satelliten-Relais immer die selbe IP-Adresse zugeordnet werden. Das erreicht man an der Fritzbox! über ‚WLAN', dann ‚Funknetz' und über das Stift-Icon am rechten Rand der Zeile.

Sat-Relais: Taster-Modus

Es kann ein manueller Taster verbaut werden. Dieser schält im Toggle-Mode den Zustand des ersten Relais um. Also von EIN nach AUS nach EIN nach AUS ….

Der Taster ist am NodeMCU an PIN D2 (GPIO4) und GND angeschlossen.

Raspberry kommuniziert an Sat-Relais

Damit die Satelliten-Relais dem Raspberry bekannt gemacht werden muss wie folgt vorgegangen werden:

- Weitere [TASK_xx] anlegen in der /home/pi/scripte/awattar_scheduler.conf
- Dort bash-Scripte verknüpfen
- Im Verzeichnis /home/pi/awattar bash-Scripte anlegen starte_R9.sh / beende_R9.sh
- In den bash-Scripten das passende curl-Kommando aufrufen
  - curl -I -XGET 'http://192.168.178.xx/?relais=1'
  - Hinweis: das zweite Relais wird mit '../?relais2=' angesprochen
  - Hinweis: die zweite Relais-Box hat eine andere IP-Adresse

Dazu muss die IP-Adresse der Box mit den Relais 9 & 10 bekannt sein. Die Box mit den Relais 11 & 12 hat eine eigene IP-Adresse.

Ein Task könnte so aussehen:

```
[Task_BWWP-R9]
Info = 1.Relais des 1.Satelliten mit Temp-Fühler
enable = true
starttime = 0
periode = 20
Duration = 3.9
commando_start = /bin/bash /home/pi/awattar/starte_R9.sh  >>
/home/pi/awattar/picron.log
commando_stop = /bin/bash /home/pi/awattar/beende_R9.sh  >>
/home/pi/awattar/picron.log
```

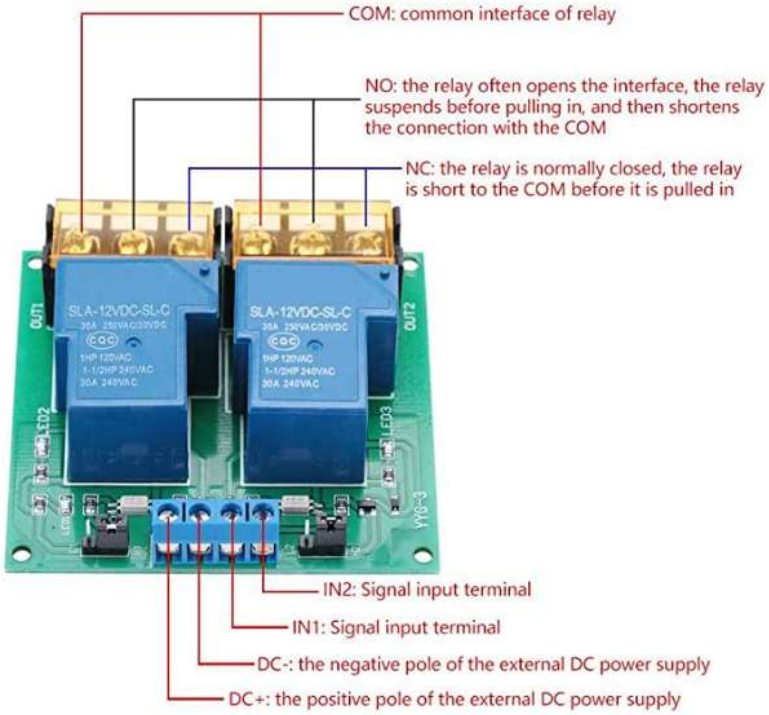
die passende starte_R9.sh könnte so aussehen:

```
#!/bin/bash
# Brauchwasser WP, Relais mit Thermostat
# /home/pi/awattar/starte_R9.sh
# Relais 9 an Satellit 1 einschalten ist 1, ausschalten ist 0 !!
echo "--->  $(date)  Start Relais 9 "
curl -I -XGET 'http://192.168.178.xx/?relais=1'
```

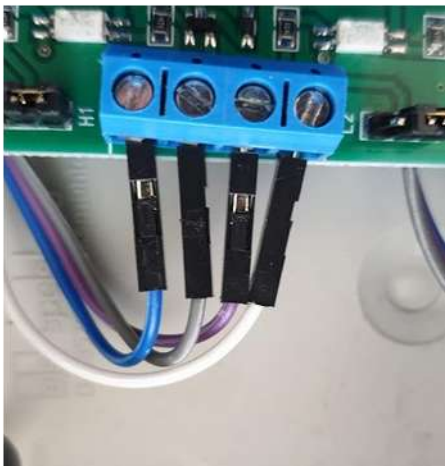
die passende beende_R9.sh könnte so aussehen:

```
#!/bin/bash
# Brauchwasser WP, Relais mit Thermostat
# /home/pi/awattar/beende_R9.sh
# Relais 9 an Satellit 1 einschalten ist 1, ausschalten ist 0 !!
echo "<---  $(date)  Stopp Relais 9 "
curl -I -XGET 'http://192.168.178.xx/?relais=0'
```

Relais mit NodeMCU verbinden



Relais:	Farbe:	NodeMCU:
DC+	Blau	VV
DC-	Grau	GND
IN1	Lila	D1 (GPIO5)
IN2	Weiss	D5 (GPIO14)



Erweiterungsmöglichkeiten

- Es können mehrere Temperaturfühler angeschlossen werden
- Es kann ein Wandler-StromzErweiterungsmöglichkeitenähler angeschlossen werden
- Statt nur in eine Log-Datei zu schreiben können Infos auch an einen Telegram (ähnlich Signal) Channel gesendet werden.
- Es können mehrere solche Satelliten-Relais angebunden werden.