Projekt: solaranzeige.de

Eigene Erweiterungen der Solaranzeige schreiben.

Stand März 2021

Dokument EE017

Inhaltsverzeichnis

Übersicht:

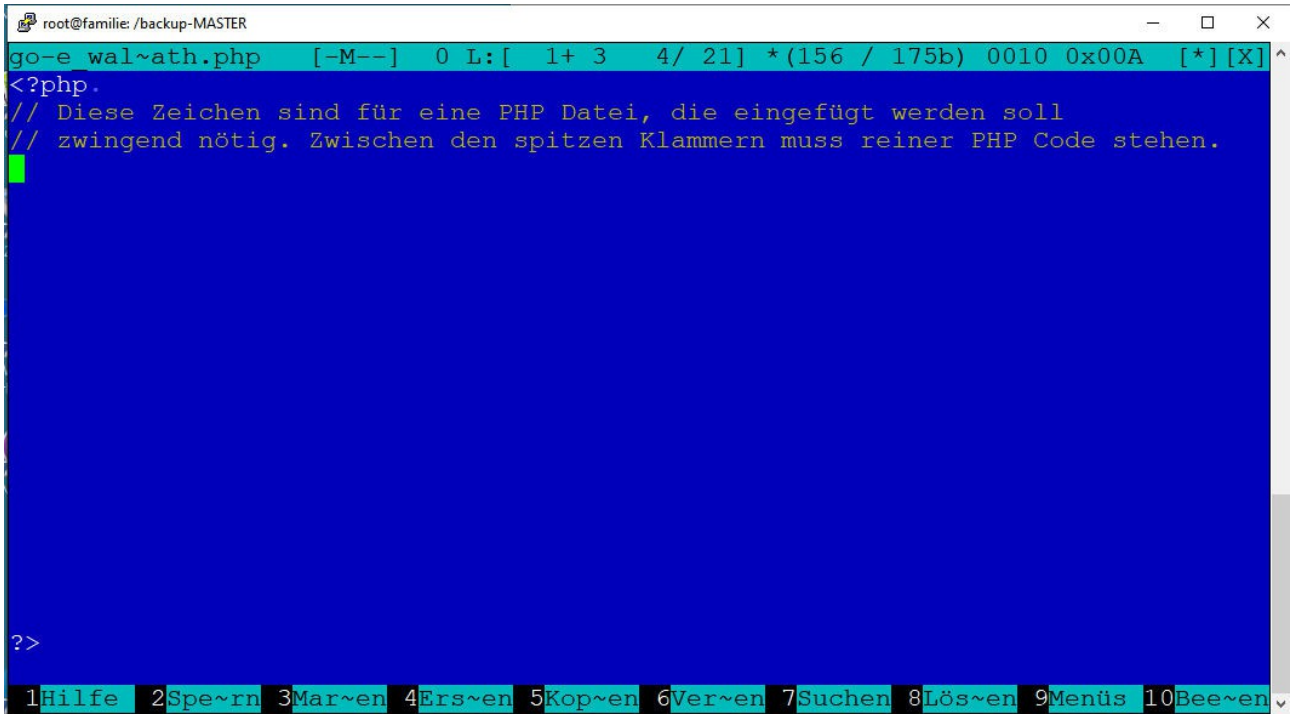
Ab der Image Version 4.7.2 ist es möglich, eigene Erweiterungen für die einzelnen auszulesenden Geräten zu schreiben, ohne dass diese bei einem Update überschrieben werden. So können eigene Berechnungen angestellt werden oder zusätzliche Datenbankabfragen integriert werden. Fast alles ist möglich, wenn man PHP programmieren kann. Auch ist es so möglich Python Scripte mit zu verarbeiten. Bitte nach „PHP and Python Integration“ oder „Python in PHP integrieren“ googlen.

Dort sind Tutorials, wie das geht, zu finden. Jede Erweiterung bezieht sich auf genau eine Geräteklasse. Die Erweiterungen sind auch austauschbar, wenn man darauf bei dem Erstellen achtet. So kann man auch Erweiterungen für Andere schreiben. Bitte darauf achten dass es sich um wirkliche PHP Dateien handelt. Also mit den Zeichen „**<?php**“ beginnend und mit „**?>**“ endend. Ansonsten baut man sich ungewollt eine Sicherheitslücke ein. Funktionieren würde es trotzdem.

Was ist zu tun?

Jedes Gerät wird mit einem eigenen Script ausgelesen. Die Verbindung zwischen Regler Nummer und Script Name findet man in der Datei „regler_auslesen.php“ Nehmen wir einmal an, wir haben eine go-eCharger Wallbox. Die hat die Regler Nummer 29. In der „regler_auslesen.php“ Datei finden wir Regler 29 in Verbindung mit der Datei „go-e_wallbox.php“

Möchte ich jetzt eine Erweiterung für dieses Gerät selber schreiben muss ich eine Datei mit dem Namen „go-e_wallbox_math.php“, in dem Verzeichnis /var/www/html/ neu anlegen. Diese Datei muss eine PHP Datei sein die mindestens folgende Zeilen hat:



Die Dateien, in denen die eigene PHP Software eingefügt wird muss immer den Zusatz „_math“ haben. Heißt der Script zum Auslesen des Huawei SmartMeter „huawei_SL.php“ so muss die eigene Datei „huawei_SL_math.php“ heißen. Bei dem Script „victron_solarregler.php“ folglich „victron_solarregler_math.php“

Ist so eine Datei im Verzeichnis /var/www/html/ vorhanden, wird sie im Programm, nach dem Auslesen des Gerätes, durchlaufen. Ist sie nicht vorhanden, passiert nichts. Möchte man auf das Abspeichern der Daten Einfluss nehmen, dann so eine Datei erstellen und den eigenen PHP Code darin platzieren. Auf diese Weise können auch Python Scripte inkludiert werden. Wie das genau geht steht im Internet. Die eigene Datei wird genau da verarbeitet, wo das Auslesen des Gerätes beendet wird und die Daten dann per MQTT eventuell versendet werden. Danach werden die Daten in die Datenbank abgespeichert. Die Daten werden immer in dieser Reihenfolge verarbeitet.

1. Falls das Gerät dafür vorbereitet ist, werden Befehle zum Gerät gesendet.
2. Die Daten werden aus dem Gerät ausgelesen.
3. Hier werden die eigenen Erweiterungen verarbeitet.
4. Die Daten werden, falls gewünscht per MQTT zum nächsten Brocker gesendet.
5. Die Daten werden zur entfernten Datenbank gesendet. (Falls gewünscht)
6. Die Daten werden in die lokale Datenbank geschrieben.
7. Falls gewünscht werden Daten zur HomeMatic gesendet.
8. Falls nötig werden Nachrichten über Pushover gesendet.

Welche Variablen stehen in der Erweiterung zur Verfügung?

1. $aktuelleDaten[]
2. $Tracelevel
3. $Homematic
4. $Messenger
5. $MQTT
6. $Pfad
7. $RaspiTemp

Im Prinziep alle Variablen der jeweiligen x.user.cpnfig.php und das Array $aktuelleDaten in dem die ausgelesenen Werte des Gerätes stehen. Das Array hat je nach Gerät immer einen anderen Aufbau. Am besten das Array in die LOG Datei schreiben lassen, dann kann man sich den Aufbau genau ansehen. Mit dieser Zeile wird das Array in die LOG Datei geschrieben.

```
$funktionen->log_schreiben(print_r($aktuelleDaten,1),"    ",1);
```

Möchte man zusätzliche Daten in die Datenbank schreiben, dann geht das so:
Man kann zu den bestehenden Measurement's Felder hinzu fügen oder auch eigene Measurement's mit Feldern erzeugen. Die nötige Query muss folgenden Aufbau haben:

```
Measurement   Feldname=Wert  zentralerTimestamp
```

Beispiel:

```
$aktuelleDaten['ZusatzQuery'] = 'Summen  NeuerFeldname=Wert1 '.$aktuelleDaten['zentralerTimestamp'];
```

oder auch

```
$aktuelleDaten['ZusatzQuery'] = 'DatenNeu NeuerFeldname1=Wert,
NeuerFeldname2=Wert '.$aktuelleDaten['zentralerTimestamp'];
```

(Alles in einer Zeile)

Möchte man 2 Measurements neu schreiben, dann muss ein 'New Line' Zeichen dazwischen „\n“
Die gesamte Query muss in die Variable $aktuelleDaten['ZusatzQuery']

Die Zusatz Query kann mehrere Measurements sowie mehrere Feldnamen enthalten. Sie darf jedoch keine schon existierende Feldnamen enthalten, da die ja kurz danach von dem normalen Script eingefügt werden. Bitte Fragen dazu im Forum stellen. Die Reihenfolge der Measurement's ist nicht wichtig. Darauf muss man nicht achten. Bitte kurz bevor man mit den eigenen Erweiterungen anfängt noch einmal ein Update machen, da sich diese Funktionen noch in der Entwicklung befinden.

Hier bietet es sich natürlich auch an, die Daten zusätzlich in eine 3. Influx noch zu schreiben oder in eine andere Datenbank wie z.B. eine MySQL oder SQLite. Je nachdem in welchem Format man die Daten benutzen möchte.

Auch könnte man die Datenbankstruktur vollkommen ändern. Es bieten sich viele weitere Möglichkeiten an.

Auslesen der Influx Datenbank:

Hier ist ein CODE Schnipsel, wie man eine Datenbank auslesen und die Werte weiterverarbeiten kann. In diesem Fall geht es um die Datenbank „solaranzeige“ mit dem Measurement „PV“

```
$Datenbank = "solaranzeige";
$Measurement = "PV";
$sql = urlencode('select * from '.$Measurement.' order by time desc limit 1');
$ch = curl_init('http://localhost/query?db='.$Datenbank.'&precision=s&q='.$sql);

curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
curl_setopt($ch, CURLOPT_TIMEOUT, 15);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 12);
curl_setopt($ch, CURLOPT_PORT, 8086);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$Ergebnis["result"] = curl_exec($ch);
$Ergebnis["rc_info"] = curl_getinfo ($ch);
$Ergebnis["JSON_Ausgabe"] = json_decode($Ergebnis["result"],true,10);
$Ergebnis["errorno"] = curl_errno($ch);

if ($Ergebnis["rc_info"]["http_code"] == 200 or $Ergebnis["rc_info"]["http_code"] == 204)
{
  $Ergebnis["Ausgabe"] = true;
}
curl_close($ch);
unset($ch);

if (!isset($Ergebnis["JSON_Ausgabe"]["results"][0]["series"])) {
  log_schreiben("Es fehlt die Datenbank solaranzeige mit dem Measurement PV oder sie ist
leer.","|- ",3);
  log_schreiben("Fehler: ".$Ergebnis["JSON_Ausgabe"]["results"][0]["error"],"|- ",4);
}
else {
  for ($h = 1; $h < count($Ergebnis["JSON_Ausgabe"]["results"][0]["series"][0]
["columns"]); $h++) {
    $DB1[$Ergebnis["JSON_Ausgabe"]["results"][0]["series"][0]["columns"][$h]] =
$Ergebnis["JSON_Ausgabe"]["results"][0]["series"][0]["values"][0][$h];
  }
  log_schreiben("Datenbank: solaranzeige DB1 ".print_r($DB1,1),"  ",10);
}
```

Fehleranalyse:

Hat man PHP Fehler in dem eigenen Script, läuft das gesamte Programm nicht mehr. Fehler kann man in der LOG Datei /var/www/log/php.log sehen. Funktioniert die gesamte solaranzeige nicht mehr, dann zu erst in die php.log schauen. Läuft das Programm, aber der Output ist nicht so wie erwartet, dann bitte in die LOG Datei /var/www/log/solaranzeige.log schauen.
Um Fehler auf zu spüren den Tracelevel auf 8 oder 9 erhöhen. Normal ist 7. Möchte man wenig Logeinträge dann auf 5 reduzieren.

Temperatur des Raspberry abspeichern:

Eine weitere Übung ist, die Temperatur des Raspberry's in der Influx Datenbank im Measurement „Service“ abzuspeichern. Natürlich kann man die Daten auch in jedem anderen oder auch neuen Measurement abspeichern. Dazu muss man eine „_math“ Datei, wie vorher beschrieben, erstellen und dort folgende Einträge machen:

```
/************************************************************************
//  Raspberry Temperatur in die Infux Datenbank speichern
//  Die Temperatur steckt in der Variable $RaspiTemp
************************************************************************/

//  So wird die Zusatz Query zusammengestellt.
//  Alle Daten werden in die aktuelle Datenbank des Gerätes in das Measurement "Service" geschrieben
//  Der Zeitstempel ist der 'zentrale Timestempel'
//  Damit ist die Visualisierung in Grafana sehr einfach.

$aktuelleDaten["ZusatzQuery"]  = "Service  RaspiTemp=".round($RaspiTemp,1);
$aktuelleDaten["ZusatzQuery"] .= "   ".$aktuelleDaten["zentralerTimestamp"];

//  Wenn der Wert auch in die LOG Datei geschrieben werden soll.

$funktionen->log_schreiben("Rasperry Temperatur: ".round($RaspiTemp,1)." °C",">  ",5);
```

In der Variable „$RaspiTemp“ steht die Temperatur. Diese wird über die ZusatzQuery in die datenbank geschrieben und kann so bequem im Dashboard angezeigt werden. Die „_math“ Datei kann natürlich zusätzlich noch für andere Berechnungen genutzt werden.

Möchte man prüfen, ob die Daten wirklich in die InfluxDB geschrieben werden, dann auf der Konsole folgende Eingaben machen:

influx

use solaranzeige

precision rfc3339

select * from Service order by time desc limit 10

quit

Jetzt sollte man in der Spalte „RaspiTemp“ die Temperatur in Grad sehen.