

Projekt: solaranzeige.de

Selber die Solaranzeige erweitern und eigene Ausleseroutinen für Geräte schreiben.

Stand November 2023

Dokument UD018

Inhaltsverzeichnis

Übersicht:.....	2
Schritt für Schritt Anweisung:.....	3
Die user.config.php bzw. x.user.config.php.....	3
Abspeichern der Daten in die influxDB.....	4

Übersicht:

Möchte man für ein Gerät die Ausleseroutinen selber schreiben, das Gerät aber in die Solaranzeige integrieren, so soll dieses Dokument Hilfestellung dazu geben. Die Solaranzeige ist ab der Image Version 4.7.2 vorbereitet, dass man ein eigenes Gerät selber auslesen kann. Dazu sind 2 PHP Dateien nötig. Beide Dateien müssen den PHP Richtlinien folgen.

1. user_init.php
2. user_device.php

Die user_init Datei wird nur während dem Booten einmal durchlaufen und stellt die serielle Geschwindigkeit der Schnittstelle oder andere Hardware Einstellungen auf den richtigen Wert ein. Müssen keinerlei Einstellungen vorgenommen werden, wie zum Beispiel bei einer LAN Verbindung, so ist diese Datei auch nicht nötig. Beide Dateien werden benutzt, wenn sie vorhanden sind. Fehlen beide oder eine der Dateien, so werden sie einfach nicht ausgeführt. Eine Fehlermeldung wird dadurch nicht produziert. Den angegebenen Regler in der user.config.php darf es in der offiziellen Software nicht geben. Am besten eine Nummer über 1000 nehmen.

Die Datei user_device.php ist für das eigentliche Auslesen des Gerätes verantwortlich. Die Datei darf bei einer Multi-Regler-Version nur maximal 10 Sekunden laufen. Bei einer Singel-Regler-Version bis zu 50 Sekunden. In der user_device.php muss das Auslesen und Abspeichern in die InfluxDB vorhanden sein, sowie wenn gewünscht die Übertragung der Daten als MQTT, zur HomeMatic, per Pushover usw. Man könnte natürlich das „Gerüst“ eines vorhandenen Geräte Scripts nutzen, es ist aber auch möglich ein ganz eigenen Programmierstil bis hin zu Python einzusetzen. Die PHP Datei muss am Ende nur ein „return;“ stehen haben, damit der Programmverlauf nicht abgebrochen wird. Man kann den Programmteil so schreiben, dass weder die user.config.php noch die funktionen.inc.php benutzt werden. Wichtig ist nur, dass am Ende die ausgelesenen Daten in die Influx Datenbank geschrieben werden.

Ein vollwertiger Script kann folgende Schritte beinhalten, in folgender Reihenfolge:

1. Öffnen der Verbindung zum auszulesenden Gerätes
2. Falls in der Pipe Befehle vorhanden sind, dann diese zum Gerät senden
3. Das Gerät auslesen
4. Die Daten per MQTT an einen Broker senden
5. Die Daten in die lokale und eventuell auch in die remote Influx Datenbank schreiben.
6. Schließen der Verbindung mit dem Gerät
7. Daten an eine HomeMatic senden
8. Daten an einen Messenger senden

Alle diese Funktionen müssen innerhalb von 10 Sekunden erledigt sein, da bis zu 6 Geräte ausgelesen werden können.

Für die Verarbeitung muss die Konfigurationsdatei `user.config.php` / `x.user.config.php` vorhanden sein und es können Funktionen aus der vorhandenen Datei `funktionen.inc.php` benutzt werden. Es ist jedoch viel einfacher alles selber zu schreiben und in die `user_device.php` zu packen. Dann ist alles an einer Stelle zu finden. Außerdem ist so eine Vorgehensweise auch Update sicherer. Falls man mehr als ein Gerät auslesen möchte, dann muss alles in die `user_device.php` gepackt werden. Es gibt nur einen „User Ausgang“.

Schritt für Schritt Anweisung:

Wie und mit welcher Programmiersprache die eigene Device ausgelesen wird, ist jedem selber überlassen. Die Daten können dann über die API in die Datenbank geschrieben werden. Wie die API funktioniert steht in dem [Dokument API.pdf](#)

Über die API können mit einem Aufruf alle Daten auf einmal in die Influx Datenbank geschrieben werden.

Als Reglernummer in der `user.config.php` bitte eine Nummer wählen, die es noch nicht gibt. Am besten eine Reglernummer über 1000.

In der angelegten `user_device.php` Datei bitte am Anfang und Ende jeweils folgenden Eintrag machen:

Anfang:

```
$funktionen->log_schreiben( "----- Start user_device.php ----- ", "|--", 6 );
```

Ende:

```
$funktionen->log_schreiben( "----- Stop user_device.php ----- ", "|--", 6 );
```

Damit kann man in der LOG Datei dann sehen, ob die Datei auch durchlaufen und ob sie eventuelle mitten drin abgebrochen wird. (Dann fehlt der Eintrag am Ende der Datei.)

Die `user.config.php` bzw. `x.user.config.php`

Auch für die eigene Routine, die man schreibt muss es eine `user.config.php` geben bzw. bei einer Multi-Regler-Version eine `x.user.config.php`. In dieser Konfigurationsdatei werden folgende Variablen erwartet.

`$Regler = 1000;`

`$USBDevice = "";`

Die `$USBDevice` Variable muss nur vorhanden sein. Der Inhalt ist egal. Sonst gibt es Fehlereinträge in der `php.log`

Abspeichern der Daten in die influxDB

Mit Hilfe der Solaranzeigen API (Dokument API.pdf) können die Daten ganz einfach in eine vorhandene Influx Datenbank geschrieben werden. Soll eine andere Datenbank wie die „solaranzeige“ benutzt werden, muss sie vorher erst angelegt werden. Folgende Befehle sind auf der Konsole dazu nötig:

```
influx
create database meineDatenbank
quit
```

Und nun zur API:

Das Herzstück der API ist eine XML Datei, die per HTTP POST Request übertragen werden muss.

```
<?xml version="1.0" encoding="UTF-8" ?>
<solaranzeige>
  <version>1.0</version>
  <in_out>in</in_out>
  <database name="test">
    <measurement name="api">
      <fieldname name="Batterie_Strom"><value>-3.4</value></fieldname>
    </measurement>
  </database>
</solaranzeige>
```

Folgende Elemente sind Pflicht und dürfen nur einmal pro XML Datei vorkommen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<solaranzeige>
  <version>1.0</version>
  <in_out>in</in_out>
</solaranzeige>
```

Zwischen den Elementen <solaranzeige> und </solaranzeige> darf folgendes Element mehrfach vorkommen:

```
<database name="meineDatenbank">

</database>
```

Im Attribut „name“ muss der Name der Datenbank stehen. Diese Datenbank muss in der InfluxDB vorhanden sein, ansonsten kommt ein Fehler zurück.

Zwischen

<database name="test"> und </database>

können mehrfach folgende Elemente stehen:

<measurement name="api">

</measurement>

Das Attribut „name“ beinhaltet wieder den Namen des Measurements in das die Daten geschrieben werden sollen. Existiert das Measurement nicht, wird es angelegt!

Zischen <measurement name="api"> und </measurement>

können mehrfach folgende Elemente stehen:

<fieldname name="Batterie_Spannung">

<value type="num">48,3</value>

</fieldname>

Hier befinden sich die eigentlichen Datenwerte. Bei dem Element <value> kann man das Attribut „type“ angeben. Das bewirkt zum Beispiel, dass ein Wert 34,5 in 34.5 umgewandelt wird, da es nur so in der Datenbank in einem Numerischen Feld abgespeichert werden kann. Es gibt folgende Typen:

'num' = Zahlen +/-

'str' = Zeichenketten

'hex' = Hex Werte

Fehlt das Attribut „type“ dann erfolgt keine Wandlung und es wird so wie gesendet in die Influx Datenbank gespeichert. D.h. In diesem Fall dürfen numerische Werte kein Komma enthalten.

Mit diesen XML Elementen kann ein Wert oder viele Werte in unterschiedlichen Measurements und unterschiedlichen Datenbanken abgespeichert werden. **Achtung die Datenbank muss vorhanden sein, also eventuell erst vorher anlegen!**

Übertragen werden muss die xml Datei mit folgendem URL Aufruf:

<http://solaranzeige.local/api/control.php> oder

http://<IP-Adresse des Raspberry>/api/control.php

Mit der API können genauso einfach vorhandene Werte aus einer der Influx Datenbanken gelesen werden. Beschrieben ist das in dem Dokument API.pdf

Das Dokument wird noch verfeinert. Über Eure Anregungen und Berichte würde ich mich freuen.
(März 2023)

Wer ein Gerät über diesen Weg anbindet, bitte meldet Euch. Zusammen können wir auch die Beschreibung verfeinern. Ich helfe gerne, wenn ihr Fragen im Detail habt. support@solaranzeige.de

Bitte die beiden Dateien im Anschluss zum Download im Forum einstellen. Damit kann dann jedes Mitglied das Gerät auslesen und die Anzeige benutzen. Schön wäre noch ein Dashboard für das Gerät. Auch das kannst du allen zur Verfügung stellen.

© Solaranzeige.de Nachdruck und Verbreitung nur mit unserer schriftlichen Genehmigung.