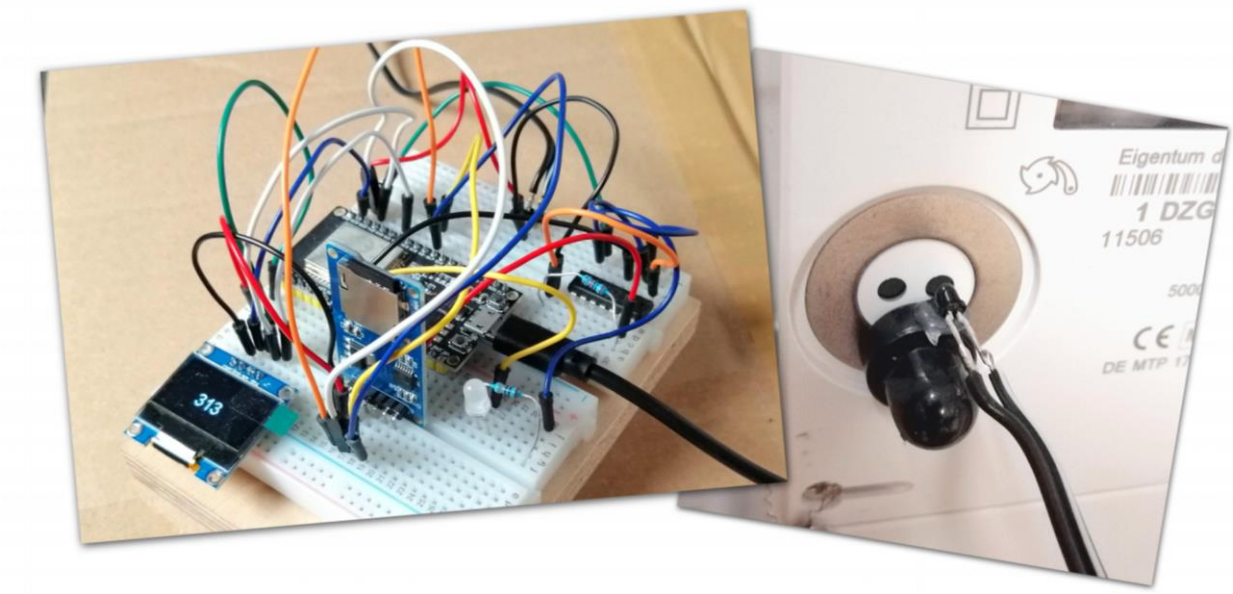
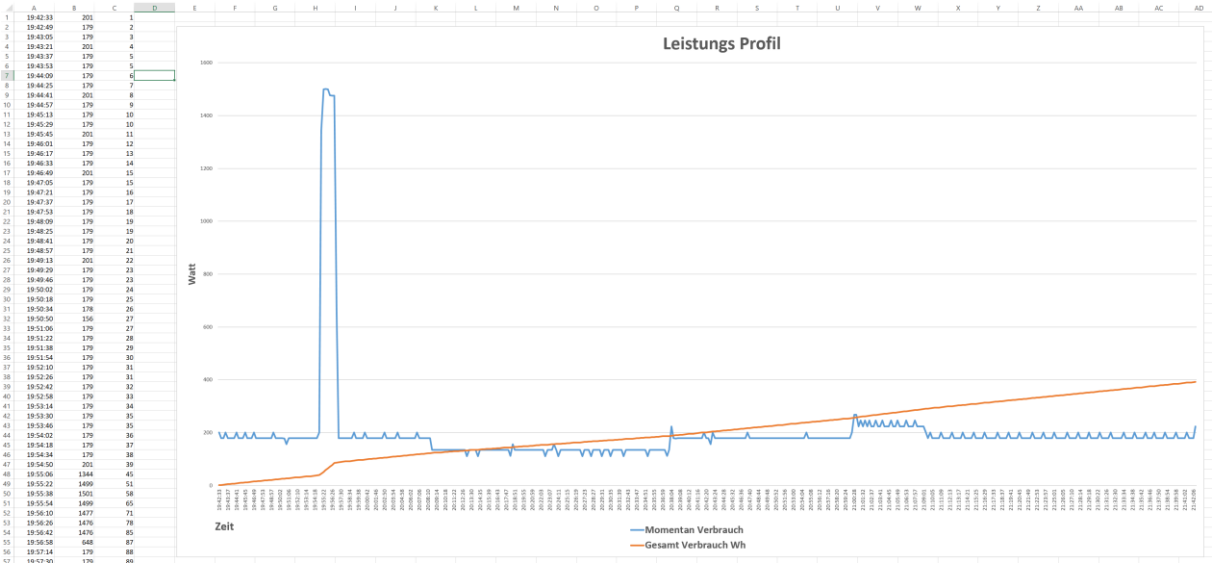
ESP32 Smart Meter V2



ESP-32 Board mit Display, 74LS14 Schmitt Trigger, IR-Fototransistor am SM, SD-Speichermodul

In Zeiten der Energiewende, hilft das Einsparen von elektrischer Energie nicht nur dem Klima, sondern auch dem Geldbeutel. Dieses ESP32 Projekt ermöglicht es ein Leistung Verbrauchs- Profil zu erstellen, um Energiefresser im Haushalt leichter zu identifizieren.



EXCEL Grafik: Tabellenkalkulation mit Verbrauchswerten

In Version 2 sind einige Korrekturen am Filesystem und eine Erweiterung in der Datensammlung hinzugekommen.
Der Gesamt Verbrauch in Watt kann jetzt im Diagramm dargestellt werden. Für jede Datei (LogDatxx.csv) wird von 0 beginnend der Verbrauch in Watt (**hier in Orange**) aufsteigend dargestellt, und kann anhand der Datenpunkte bestimmt werden. Eine EXCEL Beispieldatei (LogDat0.xlsx) ist beigefügt. Ab der 2. Datei (LogDat01.csv) wird das Erstellungsdatum und der Wh Startwert gespeichert.

WEB Browser Anzeige

Die aktuellen Verbrauchsdaten werden im Internetbrowser (nur im Heim Netz) angezeigt.

Damit ist möglich, mit dem Smartphone in „Echtzeit“ die aktuelle Leistungsaufnahme anzusehen.

Ein im Kellerraum installierter Wechselstromzähler (Smart Meter) mit ESP SM, Kann so die aktuellen Verbrauchswerte in die Wohnung übertragen und am PC, Tablet oder Smartphone anzeigen.

Voraussetzung für das Projekt ist ein Smart Meter, wie es in den meisten Haushalten bereits installiert und für Neubauten vorgeschrieben ist.

Im Prinzip leistet das Projekt die Kontrolle des Stromverbrauchs, wie es auch mit der Smart Meter Telemetrie möglich ist, wenn im Smart Meter ein Internetmodul verbaut ist.

Das Projekt liest die Verbrauchsdaten über eine genormte optische Infrarot-Schnittstelle. Ein im ESP32 Modul laufender WEB-Server stellt die Verbindung mit dem Heim Netz her. Auf dem SD Speichermodul (Micro SD) werden die Verbrauchsdaten erfasst. Diese Daten dienen der späteren Auswertung mit einer Tabellenkalkulation, im Beispiel EXCELL.

DZG Smart Meter WS7412.1

Der Wechselstromzähler (Smart Meter) der Firma DZG sendet mit einem IR-LED, kontinuierlich alle 2 Sekunden Verbrauchsdaten.

Das Verfahren ist in genormt.

Der Wechselstrom Zähler (Smart Meter) muss, um die Verbrauchsdaten zu übermitteln, dafür vorbereitet werden. Hierfür bieten alle Smart Meter die Möglichkeit mittels einer am Zähler angebrachten Taste und/oder einer Lichtschnittstelle (Taschenlampe), das Smart Meter zu bedienen.

Die Anzahl der Daten, die das Smart Meter ausgibt, ist von Hersteller zu Hersteller unterschiedlich. Das Sende Datenprotokoll selbst ist genormt nach SML 1.0x (Smart Message Language) Spezifikation. Das Projekt benötigt Daten der aktuell abgenommenen Wirkleistung (Momentanleistung).
Das Smart Meter vom Typ **DZG WS7412.1**, zeigt die Momentanleistung (P) am Display an, aber gibt die Daten nicht über die optische Schnittstelle aus. Lesen Sie dazu die Bedienungsanleitung des Smart Meter Herstellers.

Unter „**inF oFF**“ bzw. „**inF on**“ im Smart Meter Menü, müssen die Zusatzinformationen der optischen Info-Schnittstelle dafür **aktiviert** werden (**inF on**).

Die optische Schnittstelle vom DZG WS7412.1 übergibt dann die Verbrauchswerte (Wirkenergie Bezug +A Total), mit vier Nachkommastellen genau. Das reicht um die aktuelle Momentanleistung zu berechnen.

Alle zwei Sekunden wird ein erweiterter Datensatz an der optischen Schnittstelle ausgegeben.

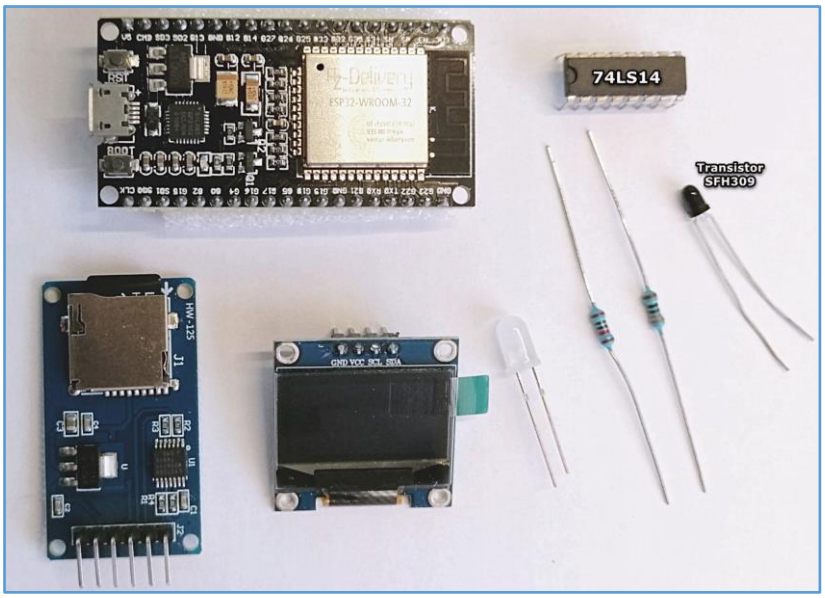
Das Projekt sammelt 8 Datensätze ein, um den Momentan Verbrauch zu berechnen. In dieser Zeit ist auch ein Korrekturdatum enthalten.

Das bedeutet, dass alle 16 Sekunden ein neuer Momentan Wert vom ESP32 Projekt angezeigt wird.

Wenn ein Verbraucher mit der WEB - ESP32 Smart Meter Anzeige überprüft wird, ist eine Zeitspanne von mindestens 16 Sekunden abzuwarten.

Zum Betrieb, benötigt das Projekt nur den **Namen vom Heim Netzwerks und das Zugangspasswort** vom Router. Das Programm muss dann mit der Arduino IDE kompiliert werden.
Es ist so programmiert, dass es auch ohne SD Modul und Aktivität LED funktioniert.

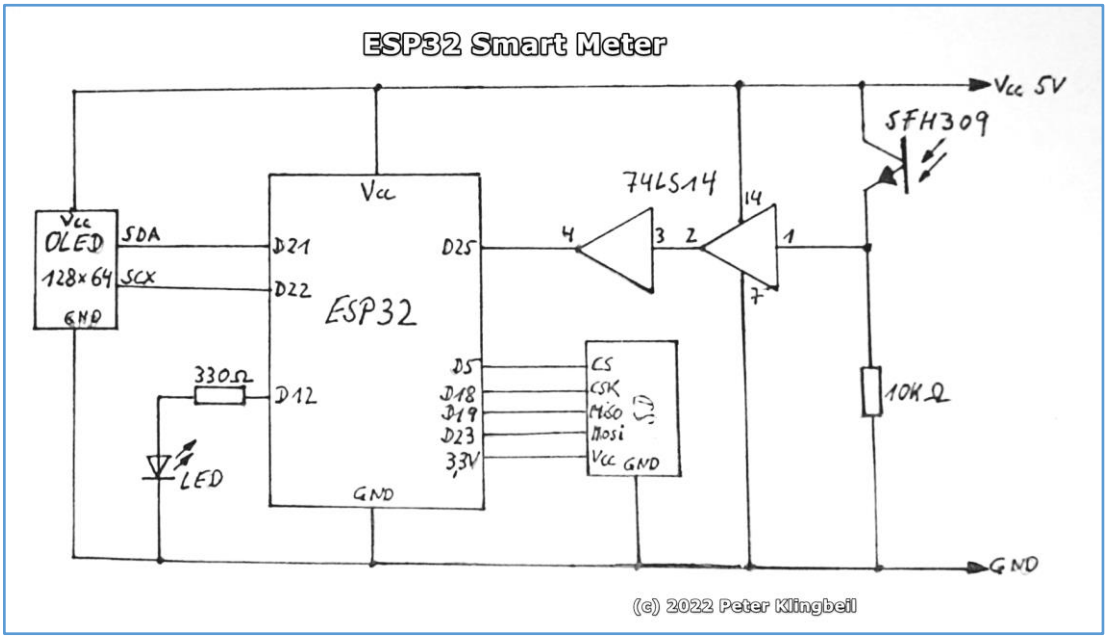
Hardware



Bauteile

- ESP32 DEV-Modul mit 4MB
- SPI Reader Micro Speicherkartenmodul
- Micro SD Karte max. 32 GB (**FAT32!**)
- 128 mal 64_Display_I2C
- TTL IC 74LS14
- Fototransistor SFH 309
- LED
- Widerstand 10 KOhm
- Widerstand 330 Ohm
- Magnet
- Steckbrett

Schaltplan:



Die Stromaufnahme beträgt ca. 50 mA. Zum Betrieb empfehle ich eine 5 Ah Powerbank.

Software
Smart-Meter.ino Version 1
Smart-Meter2.ino Version 2 (12.07.2022)
Arduino IDE => 1.8.19 mit aktuellen Bibliotheken

Inbetriebnahme
Im Programm den Heim Netz Zugang einstellen:

```
// Replace with your network credentials
const char* ssid = "Netzwerk Name";
const char* password = "Passwort";
```

Hier Ihren Heim Netz Zugang eintragen

Das Programm mit der IDE kompilieren und in das ESP32 Modul hochladen, fertig.

Messung

Der Fototransistor wird an einen kleinen Magneten geklebt, und wie im Bild am Smart Meter optischen Ausgang (LED) auf der runden Metallplatte befestigt:

Optischer Zähler Ausgang - IR LED

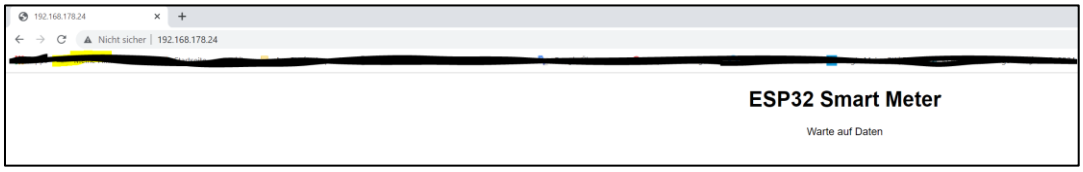
Es ist Wichtig, dass der Fototransistor direkt vor der IR-LED angebracht wird.

Nach dem Einschalten der Versorgungsspannung (USB 5 Volt), wird ein kurzer Selbsttest durchgeführt. Protokolliert wird der Test am Monitor der Arduino IDE, falls angeschlossen.
Alle Komponenten werden initialisiert und der SD-Kartenstatus für 2 Sekunden am Display angezeigt.

Es wird eine Verbindung mit dem Heim Netzwerk hergestellt. Die Verbindungsdaten werden am Display angezeigt (IP-Adresse vom ESP32). Bevor der WEB Server startet wird über das Internet die aktuelle GMT Uhrzeit abgefragt (pool.ntp.org). **Dafür muss am Router eine Internetverbindung vorhanden sein!** Weitere Verbindungen ins Internet gibt es nicht im Projekt!

Der ESP32 WEB Server wird gestartet und wartet auf eine oder mehrere Verbindungen im Heim Netz.

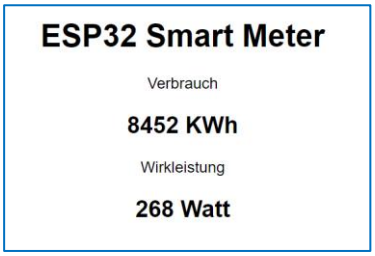
Die am Display angezeigte IP-Adresse ist im Browser einzustellen um die Daten anzuzeigen.



Browser

Die optische Schnittstelle wird ab jetzt zyklisch abgefragt. Immer wenn ein gültiger SML Datensatz eintrifft, leuchtet das Aktivität LED kurz auf. Nach 32 Sekunden, wird dann das erste Mal der Momentan Verbrauch in Watt am Display angezeigt.

Im WEB Browser werden jetzt der Momentan Verbrauch und der bisherige Gesamtverbrauch in Watt alle 16 Sekunden angezeigt. **Die Momentan Verbrauch Anzeige hinkt der am Smart Meter dadurch etwas nach. Er ist immer die Summe aller Verbraucher zu einer Zeit!**



Anzeige im Browser Fenster

Immer wenn 10 Messwerte erfasst wurden, werden die Daten auf die Speicherkarte geschrieben. Alle 120 Minuten wird eine neue Datei mit dem Energie Verbrauchs Profil angelegt. Die Auflösung der Momentan Verbrauchswerte beträgt 16 Sekunden.

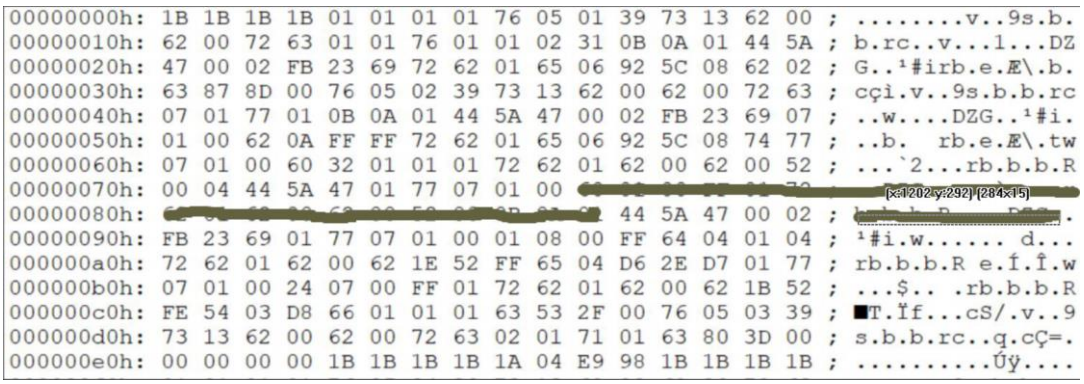
Die gespeicherte ASCII-Datei mit dem Namen **LogDatxx.csv**, kann dann einer Tabellen Kalkulation zur Auswertung zugeführt werden. Im Anhang gibt es ein Beispiel.

Das Programm

Über die optische Schnittstelle wird vom Smart Meter ein serieller Datenstrom gesendet: Datenformat 9600 Baud, 8-N-1. Der Fototransistor empfängt den Datenstrom. Mit dem nachgeschalteten Schmitt-Trigger (IC 74LS14) wird der Datenstrom stabilisiert und dem Digitaleingang D25 des ESP32 zugeführt. D25 wird hier standardmäßig genutzt.

Im Programm werden die ankommenden Daten untersucht. Sie werden mit dem Protokoll SML 1.0x gesendet (IEC 62056-21 2005). OBIS Kennzahlen im Protokoll bezeichnen einzelne Datensätze. Das Programm benötigt nur die Verbrauchs Daten. Die OBIS Kennzahl dafür lautet 1.8.0.

Jeder SML Datensatz beginnt mit einer Startkennung: 1B-1B-1B-1B-01 (hexadezimal).
Und endet mit der Sequenz 1B-1B-1B-1B (hexadezimal).



SML 1.05 erweiterter Datensatz

Immer wenn die Startsequenz vom Programm erkannt ist, werden die folgenden Daten in ein Array gespeichert. Das Aktivität LED wird kurz angesteuert.

Im Anschluss werden Daten der Wirkarbeit Bezug (+A) Zählerstand Total, OBIS 1.8.0 gesucht.
Die eigentliche Daten-Sequenz beginnt mit der Einleitung: 52-FF-65 (hexadezimal).

Wenn am Smart Meter die **Zusatzinformationen aktiv** sind, folgen 4 Bytes (32 Bit) mit dem Zählerstand, z.B. 04 D6 2E D7 wie oben gezeigt.

Die 4 Nutzdaten Bytes müssen zur Bearbeitung weiter aufbereitet werden.

Der Zählerstand mit der OBIS Kennzahl 1.8.0 vom DZG WS7412.1 hat einen Wertebereich von:

00000000 – 3B9AC9FF 99999.9999 kWh

Jede Tetrade (Nibble) der Daten enthält eine Hexadezimal „Stelle“. Der Datensatz besteht somit aus 8 Stellen:

00 04 0D 06 02 0E 0D 07 hexadezimal = 81145559 Dezimal

Der Dezimalpunkt wird vom Ende her nach der 4. Stelle gesetzt:

81145559 = 8114.5559 KWh

Der Verbrauch steht so mit einer Auflösung von 0,1 Wh zur weiteren Berechnung an.

Um den Momentan Verbrauch zu berechnen wird über einen Zeitraum, ein Verbrauchswert akkumuliert. Weil das Smart Meter der DZG WS7412.1 spätestens nach 7 Datensätzen ein Korrektur Wert liefert, akkumuliert das Programm 8 Datensätze um den Wert korrekt zu interpretieren.

Der so ermittelte Wert wird gespeichert. Vom jeweils nächsten Leistungswert wird der gespeicherte Wert abgezogen. Es bleibt ein positiver Verbrauchs Wert übrig. Da alle 2 Sekunden ein SML Datensatz vom SM ausgegeben wird, kann jetzt der ermittelte Verbrauch für 8 Messungen (16 Sekunden) berechnet werden. Das Zeitintervall zwischen den SML Datenböcken wird mit dem internen Timer des ESP32 bestimmt.

```
// aktuellen Verbrauch aus den Verbrauchs-Daten berechnen
DTime=Time-Time2;
Power=(KWH-KWHalt) * (360000/(DTime));
KWHakku=KWHakku + Power;

KWHalt=KWH;
Time2=Time;
```

Der ermittelte momentane Verbrauch wird dann für die Ausgabe am Display alle 16 Sekunden und dem WEB jeweils alle 10 Sekunden aufbereitet ausgegeben.

Die Applikation ist bewusst pragmatisch und *minimalistisch* programmiert.

Natürlich kann der momentane Verbrauch auch direkt über eine OBIS Kennzahl gelesen werden, aber mein Verbrauchszähler liefert diesen Wert, wie bereits beschrieben, nicht.
Damit dürfte das Programm aber mit jedem Smart Meter in Deutschland funktionieren.

In meinem Video Tutorial zum Programm ESP32 Smart Meter finden sich Anwendungsbeispiele:
https://www.youtube.com/channel/UCsiwk9sO0zN03tXcwTzV5vQ

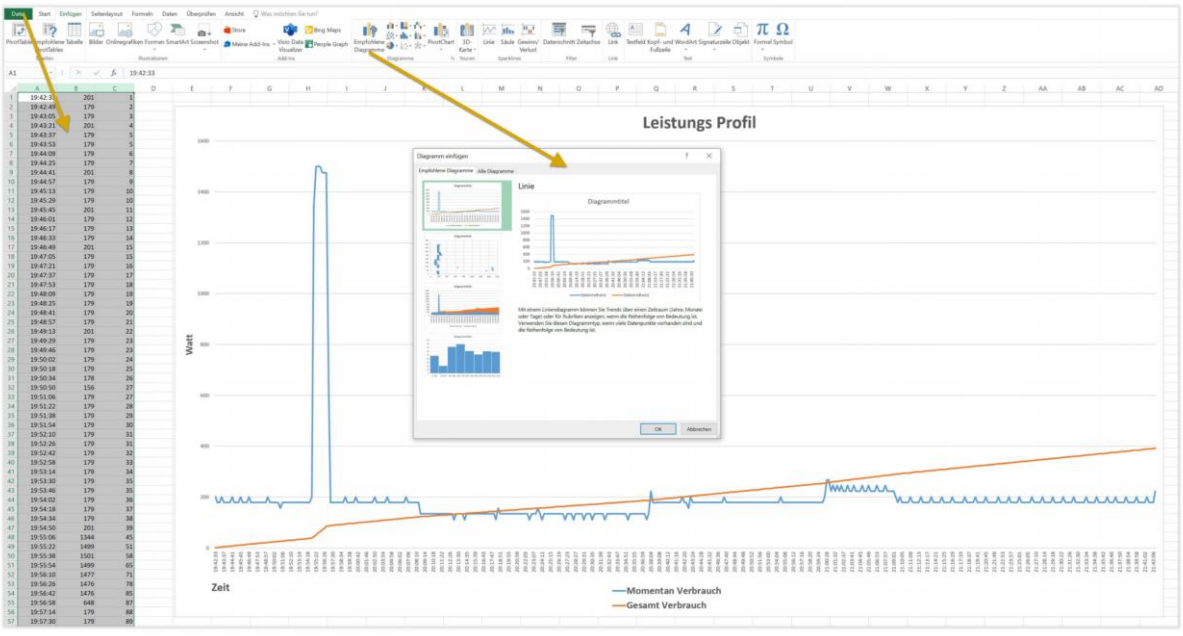
Hinweis
Wenn kein Zugang zum Internet besteht, kann mit einer Programmerweiterung die aktuelle Zeit vom Smart Meter ausgelesen werden, OBIS Kennzahl 1.0.0.

Tabellenkalkulation
Entnehmen Sie die SD Speicherkarte (abwarten bis ein neuer Wert angezeigt wurde, erst dann ESP32 ausschalten!) und kopieren die Excel Dateien LogDatxx.csv In die Tabellenkalkulation.

```
19:42:33;201;1
19:42:49;179;2
19:43:05;179;3
19:43:21;201;4...
```

Um ein Programm Überlauf zu verhindern, wird alle 2 Stunden eine neue Log Datei geschrieben und am Namen der Log Datei eine Index Zahl angehangen. Das Daten Array wird mit dem aktuellen Datum zurückgesetzt. Die SD Karte muss FAT32 formatiert sein.

Die Daten sind so angeordnet, dass Excel sie direkt verarbeiten kann. Die Uhrzeit in GMT ist getrennt mit einem Semikolon von der Momentanleistung in Watt und dem gesamt Verbrauch gespeichert. In Excel markieren Sie die drei Spalten und generieren dann ihr Diagramm:



Excel Arbeitsblatt

Die Daten haben eine zeitliche Auflösung von 16 Sekunden. Auf dem Excel Arbeitsblatt können auch kleinere Zeitabschnitte selektiert werden um die optische Auflösung im Diagramm zu erhöhen.
Die Diagramm Zeit ist GMT - 2 Stunden Sommerzeit (Vergleich wann die LogDatei angelegt wurde).

Ab der zweiten LogDatei wird der Anfang der Datei mit dem Start Datum / Uhrzeit und der bisher verbrauchten Wirkleistung ergänzt.
Diese beiden Zeilen müssen dann zur Diagrammdarstellung gelöscht werden

Tue Jul 12 07:34:34 2022
9268280 Wh
7:34:50;313;0
7:35:6;313;2
7:35:22;290;3
7:35:38;313;5
7:35:54;290;6

Damit ist es möglich aus dem EXCEL Diagramm (orange Kurve) die verbrauchte Energie direkt einem Ereignis zuzuordnen. Die Leistungskurve jeder LogDatei beginnt immer mit 0 Wh (Watt Stunden).

Wenn das Programm die Datensequenz „Wirkarbeit Bezug“ der Smart Message Language nicht findet, sendet die optische Schnittstelle keine „Erweiterten“ Daten oder die einleitende Datensequenz ist anders formatiert. Dann hilft es den „Daten Part“ im Programm mit den auskommentierten Stellen zu aktivieren, um eine Lösung zu finden.

```
  //if (inByte < 0x10)Serial.write('0');
  //Serial.print(inByte,HEX);
  //Serial.write (' ');
....
   //if (inByte < 0x10)Serial.write('0');
   //Serial.print(inByte,HEX);
   //Serial.write ('-');
```

Nachweise

github.com

https://github.com/espressif/arduino-esp32

SML im Internet
Technische Richtlinie BSI TR-03109-1 - Teil b „SML –Smart Message Language“:
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03109/TR03109-1.html

Wikipedia
https://de.wikipedia.org/wiki/Smart_Message_Language
https://wiki.volkszaehler.org/start

ESP32 Smart Meter ist ein Public Domain Projekt von Peter Klingbeil, film-werk56.de. Nachbau auf eigene Verantwortung.