Mobile Solarstation mit digitaler Überwachung

Am Fachbereich 4 |Ausgewählte Kapitel mobiler Anwendungen

Prof. Dr. Alexander Huhn

Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Wilhelminenhofstr. 75A

Lucan Raspe | s0559625

Zugehöriges Repository: github.com/PyroFourTwenty/SolarStation

Contents

Abstract

Diese Arbeit zeigt Detail den Aufbau einer vollständigen und mobilen Solarstation mit der Möglichkeit der Überwachung per Grafana aus leicht zu erhaltenen oder zu vervielfältigenden Komponenten, die vielerorts zur Verfügung stehen und recycelt werden können. Die Arbeit ist als Anleitung zu verstehen und soll sowohl zum Nachbau, als auch zur bewussteren Nutzung von Elektrizität und vermeintlichem Abfall anregen anregen.

Einleitung

Mit fortschreitendem Klimawandel wird das Thema der nachhaltigen Energiegewinnung für die Menschheit immer wichtiger. Populärer Hoffnungsträger dieses Sachvehalts ist mit Abstand die Photovoltaik. Wie viel Strom Photovoltaikanlagen generieren können, ist jedoch von vielen Faktoren abhängig. Um den Ertrag einer solchen Anlage überwachen zu können, bedarf es lediglich weniger technischer Komponenten, die zum Großteil geringpreisig verfügbar sind. Eine Beispielanlage zur Abschätzung des Potentials einer Solarstation unter Verwendung von Sensoren zu Messung des Ertrags soll in dieser Abhandlung erklärt werden.

Grundlegende Anforderungen

Konstruktion

Der Grundgedanke beim Entwurf der Station galt dem Recycling von bereits verfügbaren Komponenten, die zusätzlich auch andernorts den Nachbau erleichtern sollen. Das Grundgerüst bilden zwei Europaletten, die in einem spitzen Winkel zueinander aufrecht stehen, an der unteren Seite mit Querstreben verbunden sind und auf vier Rollen stehen, was die Mobilität der Anlage zumindest in einem gewissen Rahmen ermöglicht, sodass sie nicht nur an einem einzigen Ort Strom zur Verfügung stellt, sondern auch an andere Plätze verfrachtet werden kann. Dies ermöglicht auch die Nachjustierung zur Anpassung an den sich stetig wandelnden Sonnenstand.

Solarpanel

Als Stromquelle wurde ein Solarpanel mit einer Spitzenleistung von 120 Watt bei 12V gewählt. Ein Regler (im Verlauf dieser Arbeit auch Controller genannt) mit maximalem Eingangsstrom von 10 Ampere kontrolliert und steuert die Ladung einer 12V Bleisäurebatterie mit unbekannter Kapazität. Diese Batterie ist ein Abfallprodukt eines PKW’s und stand dementsprechend kostenfrei zur Verfügung.

Messanforderung

Es gilt drei Stromflüsse zu überwachen. Zum einen ist das der Stromfluss des Solarpanels zum Regler, der Stromfluss zwischen Regler und Batterie und zu guter Letzt der Stromfluss zur Last.

Nicht nur die Stromstärke spielt eine Rolle, auch die Spannung muss gemessen werden. Die Wahl fiel hierbei auf ACS712-Module und Spannungsteiler. Die Spannungsteiler konnten als Module mit Arduino-kompatiblen Pins gekauft werden.

Messung der Stromstärke

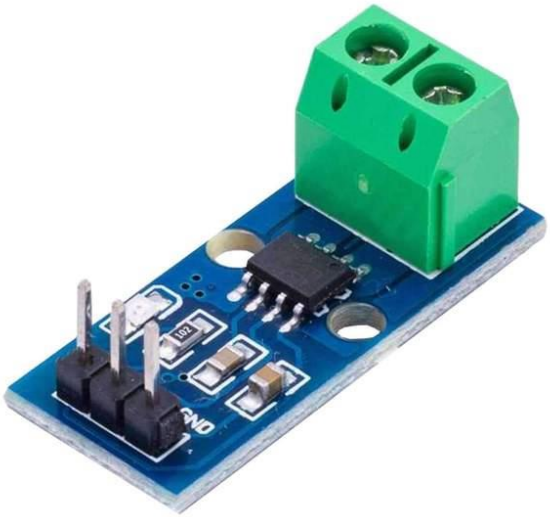

Abbildung 1: ACS712-30A

[Quelle: https://cdn.shopify.com/s/files/1/1509/1638/products/acs712-30a-ampere-stromsensor-range-modul-current-sensor-fur-arduino-bascomarduino-zubehorazdeliveryaz-delivery-23418724_600x.jpg vom 15.07.20]

Die Messung von Spannung und Stromstärke könnte zwar von einem einzigen Modul wie beispielsweise dem INA-219 erfasst werden. Dies hätte den Vorteil, dass beide Messungen von nur einer Komponente übernommen werden könnten, was die Komplexität und dadurch auch die Fehleranfälligkeit des Aufbaus verringern könnte.

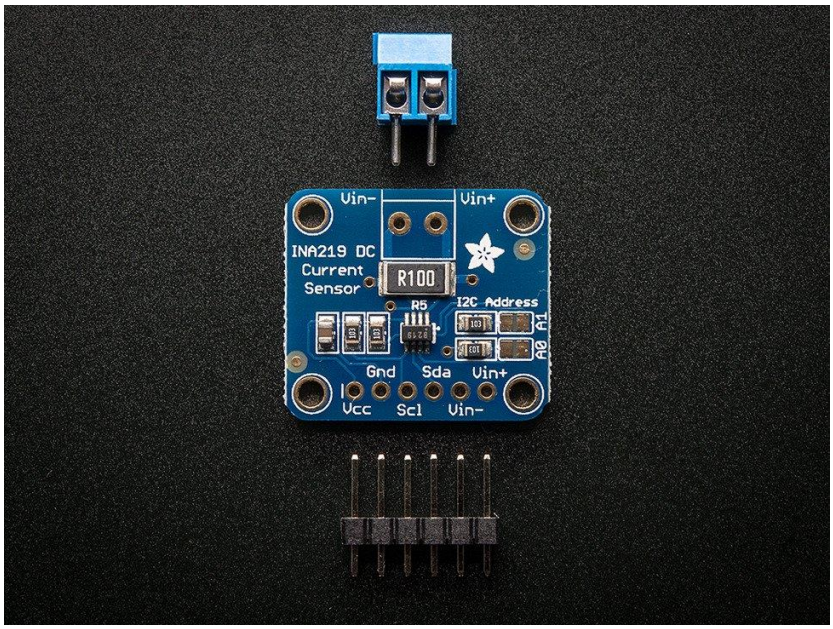

Abbildung 2: INA-219

[Quelle: https://cdn-shop.adafruit.com/970x728/904-00.jpg vom 15.07.20]

Da die Breakout-Variante dieses Moduls jedoch laut Datenblatt vom Verkäufer ‚Adafruit' nur in einem Bereich von ±3.2A Messungen erfassen kann, ist es für diese Anwendung nicht brauchbar. Zwar ist an der Stelle auch explizit erwähnt, dass es möglich ist, den verbauten 0.1 Ω Strommesswiderstand durch einen 0.01 Ω Widerstand zu ersetzen, um bis zu ±32A erfassen zu können, jedoch erschien mir diese Modifikation nicht als praktikabel, hinsichtlich des Ziels eine Lösung zu entwickeln, die leicht zu vervielfältigen ist.

(Quelle: [https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina219-current-sensor-breakout.pdf];Absatz „How does it work?"; Seite 4; Fassung vom 14.07.20)

Statt des „Alleskönnermoduls" (INA-219) fiel die Entscheidung auf eine Kombination aus Stromstärke-Sensoren und simplen Spannungsteiler-Modulen.

Bei den Stromstärkesensoren handelt es sich um ACS712ELCTR-30A-T-Breakout-Module mit Hall-Sensoren, die einen Stromstärkebereich von ±30A abdecken. (Quelle: [https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf]; Absatz „Selection Guide", Seite 2; Fassung vom 14.07.20)

Dieser Bereich ist ausreichend für eine Solaranlage dieser Größe, da der stärkste vermutete Eingangsstrom des Solarpanels 10A betragen würde. Die Wahl eines scheinbar überdimensionierten Sensors wurde außerdem aus dem Grund getroffen, da es möglich ist, dass der Solarladeregler eines Tages ausgetauscht würde, um größere Lasten anzuschließen, die bis zu 30A bei 12V erfordern würden. Ganz explizit ist hierbei die Nutzung des Solarstroms für den 3D-Druck gemeint, der

kurzzeitig Ströme dieser Größen erfordert, um beispielsweise die Druckplatte zu beheizen. Zurzeit wird die Anlage genutzt, um den Akku eines Elektro-Lastenrads zu laden. Die maximale Ladeleistung des Akku-Netzteils beträgt hierbei ca. 90W (7.5A bei 12V), was ausreicht, um die maximale Last von 10A des derzeitigen Ladereglers nicht zu überschreiten.

Messung der Spannung

Die Module zur Spannungsmessung sind nichts weiter als vorgefertige Spannungsteiler mit einem Messbereich von bis zu 25V. Dieser Bereich ist absolut ausreichend für eine 12V-Anlage. Die Spannungsteiler bestehen eigenen Messungen nach aus Widerständen mit 30 kΩ und 7.455 kΩ.

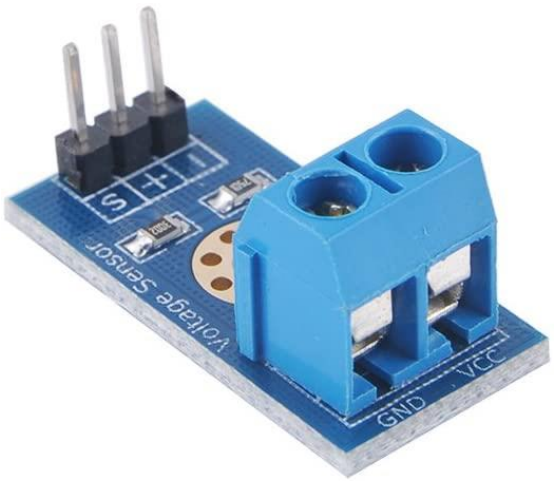

Abbildung 3: 0-25V Spannungssensor

[Quelle: https://images-na.ssl-images-amazon.com/images/I/51hqXaCrskL._AC_SL1001_.jpg vom15.07.20]

Mobilität

Die Mobilität der Anlage gewährleisten vier Lenkrollen mit einer Tragfähigkeit von jeweils 60 kg. Zwei der Rollen sind feststellbar, um ungewollte Bewegungen der Anlage zu verhindern.

Abbildung 4: Lenkrollen der Marke DSL [Quelle: https://images-na.ssl-images-amazon.com/images/I/7125M79rgLL._AC_SL1200_.jpg vom 15.07.20]

Erfassung und Übertragung der Messwerte

Die wichtigste Anforderung an die Anlage ist jedoch die kabellose Messwertübertragung der fließenden Ströme. Diese Anforderung geht einher mit der Anforderung der Mobilität, die kabelgebundene Lösungen unpraktikabel macht. Um dies zu bewerkstelligen, fiel die Wahl auf das ESP8266-Modul, das flexibel im Einsatz und gleichzeitig leicht zu programmieren ist. Die Stromversorgung des Mikrocontrollers wird über einen der USB-Anschlüsse des Solarladereglers sichergestellt.

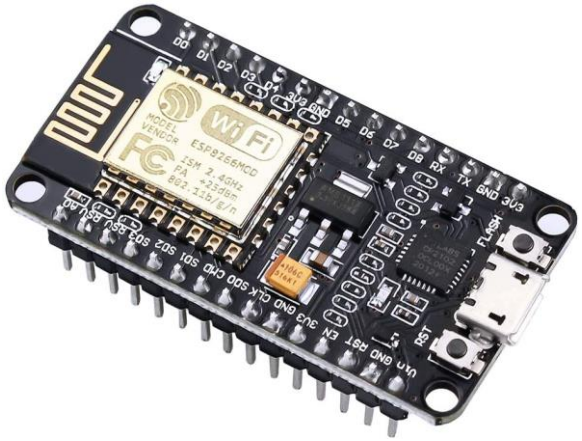

Abbildung 5: ESP8266 [https://images-na.ssl-images-amazon.com/images/I/611F%2B40fjSL._AC_SL1001_.jpg vom 15.07.20]

Erweiterung der analogen Eingänge

Da die oben beschriebenen zu benutzenden Sensoren analoge Messwerte ausgeben, kann das ESP8266-Modul nicht ohne weitere Module an alle sechs Sensoren (drei Stromstärke- und drei

Spannungssensoren) angeschlossen werden. Der Grund hierfür ist das Fehlen von weiteren Analog-Pins am ESP8266. Um also weitere (analoge) Sensoren auslesen zu können fiel die Wahl auf einen Analog-Multiplexer mit 16 Kanälen. Zugegebenermaßen sind 16 analoge Anschlüsse mehr als die benötigten sechs, jedoch traf ich diese Entscheidung mit dem Wissen, den Datalogger eines Tages um weitere Sensoren, wie zB. dem fast schon obligatorischen Temperatur-/Luftfeuchtigkeitssensor und einem Orientierungssensor, zur Messung der Ausrichtung der Anlage, erweitern zu können.

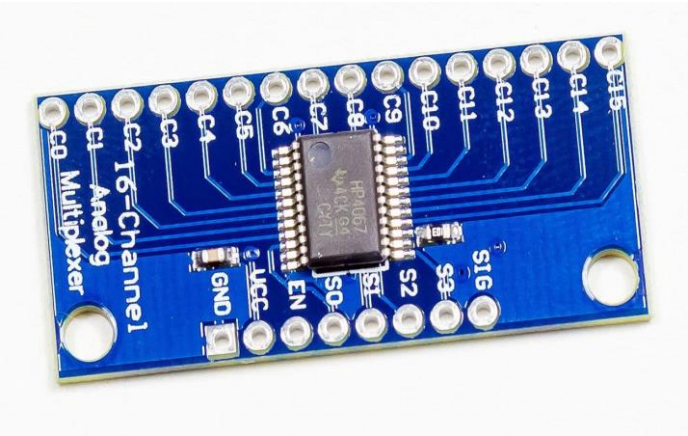

Abbildung 6: CD74HC4067 Analog Multiplexer/Demultiplexer [Quelle: https://images-na.ssl-images-amazon.com/images/I/71yxRIdvlkL._AC_SL1300_.jpg vom 15.07.2020]

Stromversorgung der Sensoren

Um alle Sensoren zuverlässig mit Strom versorgen zu können, entschied ich mich aus folgenden Gründen gegen die Nutzung der 3.3V-Pins des ESP8266. Zum einen ist die Anzahl schlichtweg nicht ausreichend (drei Ausgänge am ESP8266, sechs werden benötigt). Zum anderen gab es keine Informationsquellen zum Strombedarf der Sensoren. Laut Datenblatt des Herstellers Espressif zum ESP8266 beträgt der maximale Ausgangsstrom der I/O-Pins 12 mA (Quelle: [https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf] Absatz ‚5.1 Electrical Characteristics'; Seite 18; Fassung vom 15.07.20). Um also kein Risiko einzugehen, dass die angeschlossenen Sensoren nach Installation der Apparatur in der Solarstation aufgrund von unzureichender Stromversorgung nicht wie erwartet funktionieren, fiel die Wahl auf die Nutzung des VIN-Pins. Dieser Pin am ESP8266 gibt die eigene Versorgungsspannung weiter. Anstatt also die Stromversorgung der Sensoren mit dem ESP8266 in Serie zu schalten, wird sie parallel zur Versorgung des ESP8266 geschaltet, um etwaige ‚bottle-necks' in der Stromversorgung zu umgehen. Dieser Lösungsweg bringt jedoch seine eigenen Probleme mit sich, da die Versorgungsspannung des ESP8266 zwar mit USB-Spannung, also 5V, arbeitet, die GPIO-Pins des

Mikrocontrollers jedoch nur 3.3V-tolerant sind und auch nur in diesem Bereich Messungen erfassen können. Um die 5V auf ungefähr 3.3V zu bringen, wird ein MP1584EN-Modul verwendet.

Abbildung 7: MP1584EN [Quelle: https://eckstein-shop.de/media/image/product/803/lg/mini-dc-dc-step-down-spannungsregler-mp1584en-buck-power-module-outout-08-20v-3a.jpg vom 15.07.20]

Die Übertragung der Messwerte findet über das HTTP-Protokoll unter Nutzung einer Rest-API statt. Um die Messwerte zu empfangen, zu verarbeiten und zu visualisieren wurde ein Raspberry Pi 3 Model B+ verwendet. Der Raspberry Pi fungiert als Rest-API und als Grafana-Server.

Übertragung und Persistierung

Um die Messwerte vom ESP8266 auf den Raspberry Pi zu übertragen, werden diese in einem GET-Request „verpackt“. Vorteil dieser Methode ist, dass diese URL leichtgewichtig in Bezug auf CPU-Leistung des ESP8266 zusammengestellt werden kann. Zudem ist der HTTP-Returncode klarer Indikator dafür, ob die Messung erfolgreich auf den Server übertragen wurde. Serverseitig wird der Aufruf der Methode innerhalb einer in Python geschriebenen API unter Nutzung des Flask-Frameworks registriert und die Parameter des Requests verarbeitet, also voneinander getrennt in einer Datenbank persistiert. Da die Messwerte zeitorientiert sind, entschied ich mich für die Nutzung einer Influx-Datenbank. Vorteil der Nutzung dieser Datenbank ist die Kompatibilität mit Grafana selbst. Aufgabe der Rest-API ist also nicht nur die Entgegennahme der Messwerte sondern auch das Abspeichern in der für Grafana nötigen Form.

Konstruktion der Solarstation

Wie im Absatz ‚Konstruktion' erläutert besteht die Anlage aus zwei in einem spitzen Winkel zueinander stehenden Paletten, die mit Querstreben auf der Unterseite miteinander verbunden sind. Am besten lassen sich die Paletten am sich zuspitzenden Ende miteinander verbinden, indem eine Palette mit der Unterseite nach oben auf den Boden gelegt wird, um die Winkel zu montieren.

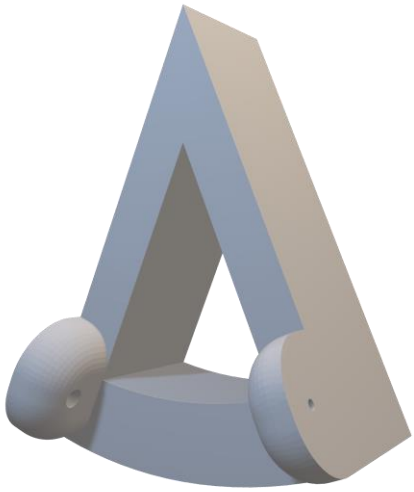

Abbildung 8: Verbindungsstück der Paletten [Dateiname: MidAngle.stl; siehe Repository]

Sofern die beiden Verbindungsstücke auf einer der beiden Paletten montiert sind, bietet es sich an die Palette auf die Seite zu stellen.

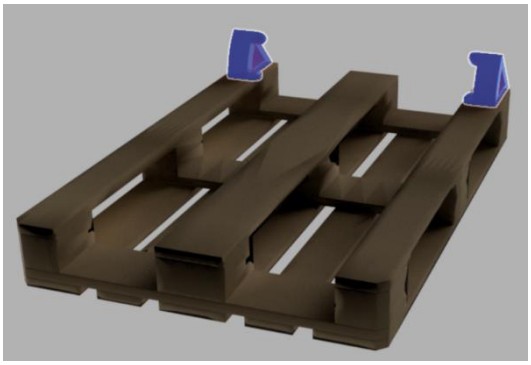

Abbildung 9: Die montierten Verbindungsstücke auf der umgedrehten Palette

Eine helfende Hand kann das Umfallen verhindern. Anschließend wird die andere Palette, ebenfalls auf der Seite stehend, an die Verbindungsstücke bewegt und festgeschraubt. Dieser Prozess gestaltet sich durchaus schwierig, da Europaletten eigenen Messungen nach bis zu 20 kg wiegen können.

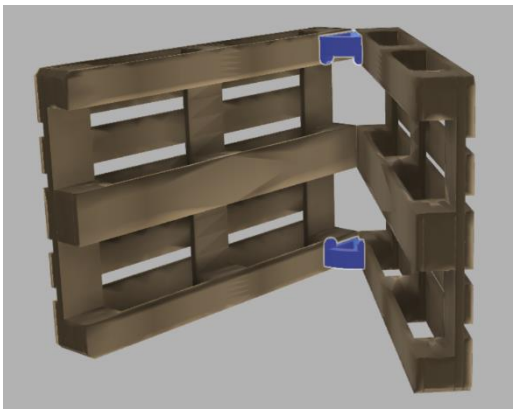

Abbildung 10: Die seitlich stehenden und verbundenen Paletten

Um die aus dem 3D-Drucker stammenden Verbindungsstücke nicht zu stark einseiting zu belasten, müssen die Querstreben montiert werden, bevor die Konstruktion aufgerichtet werden kann. Die Querstreben bestehen aus Kanthölzern mit angewinkelt (67.5°) abgesägten äußeren Kanten. Ich verwendete Kanthölzer mit einem Durchschnitt von 4 cm Höhe und 2 cm Breite. Die Länge der oberen Kante der Querstreben beträgt bei diesen Dimensionen ca. 88,5 cm und die und die Länge der unteren Kante ca. 91,8 cm. Bei diesen Querstrebenlängen ist es vollkommen ausreichend ein einziges Kantholz dieser Dimension in einer Länge von 2 m zu besorgen. Montiert werden die Querstreben an den Paletten mithilfe von normalen 90°-Winkeln, da die von mir entworfenen und gedruckten Querstrebenhalterungen den Belastungen nicht lange standhielten.

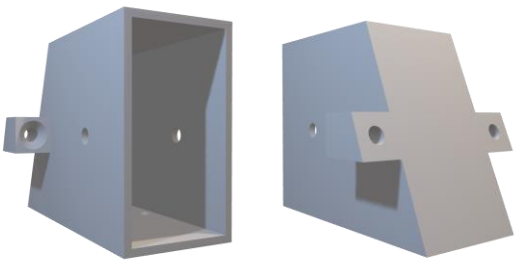

Abbildung 11: Vorder- und Hinteransicht der verworfenen Querstrebenhalterung

Nachdem die Querstreben montiert wurden sollte die Konstruktion der unteren Abbildung entsprechen.

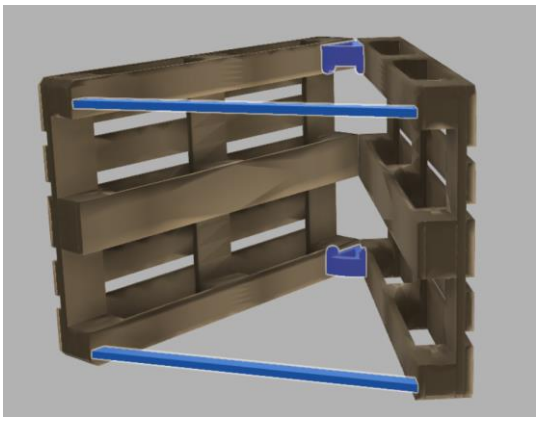

Abbildung 12: Ansicht der Konstruktion mit montierten Querstreben

Anschließend werden die Rollenhalterung montiert. Die von mir entworfenen Halterungen orientieren sich an Rollen mit einer Montageplatten-Grundfläche von 5 cm Breite und 5 cm Tiefe. Zwar wäre es möglich die Rollen ohne jene Halterungen an den Paletten zu montieren, jedoch helfen diese Winkel, die statische Belastung geradlinig nach unten abzugeben. Außerdem können die Rollen auf diese Art frei rotieren, wodurch das Potenzial der Rollen vollständig ausgenutzt werden kann.

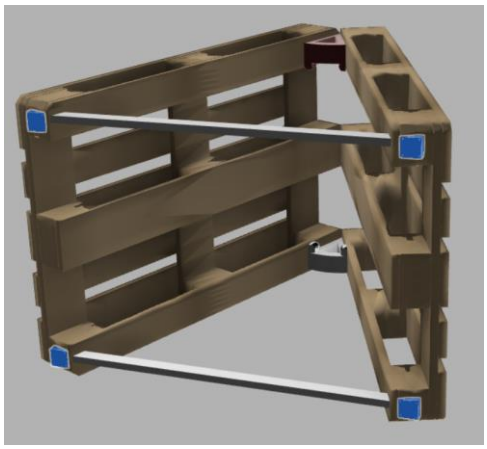

Abbildung 13: Die montierten Rollenhalterungen [Dateiname: RollerAngle.stl; siehe Repository]

Nachdem die Konstruktion wie beschrieben zusammengebaut wurde, kann sie vorsichtig aufgerichtet werden, sodass sie auf den Rollen steht.

Eine der Außenseiten der Paletten sollte mit einer ausreichend dicken und UV-beständigen Folie abgedeckt werden, um die später zu installierende Elektronik vor Regen zu schützen. Auf dieser Folie wird später das Solarpanel montiert, dessen Kabel durch kleine Löcher in der Folie und die Schlitze der Palette ins Innere der Konstruktion geführt werden, wo sie mit dem Rest der Elektronik verbunden wird. Ich persönlich entschied mich nach ungefähr einer Woche Betrieb auch die andere Palette mit Folie abzudecken, da die Anlage bei mir recht ungeschützt steht und von jeder Seite von

Regen betroffen sein kann. Eventuell ist das nicht an anderen Standorten notwendig, wie zum Beispiel an einer Hauswand.

Elektronik

An dieser Stelle möchte ich noch einmal die einzelnen Komponenten aufzählen, die in der Solarstation verbaut wurden, um die Messungen von Spannung und Strom zu erfassen. Ein ESP8266-12E-Mikrocontoller bildet das Gehirn der Messstation. An diesen Mikrocontroller ist ein 16-Channel-Multiplexer angeschlossen, an den wiederum drei Spannungssensoren und drei Stromstärkesensoren angeschlossen sind. Die Stromversorgung der Sensoren wird durch einen Stepdown-Converter sichergestellt, der aus den 5 Volt Eingangspannung des Mikrocontrollers ungefähr 3,3 Volt an die Stromstärkesensoren weitergibt. Die Spannungssensoren benötigen keine zusätzliche Stromversorgung. Wie die Sensoren anzuschließen sind, soll nicht in Textform dargelegt sein, um den Leser nicht zu langweilen.

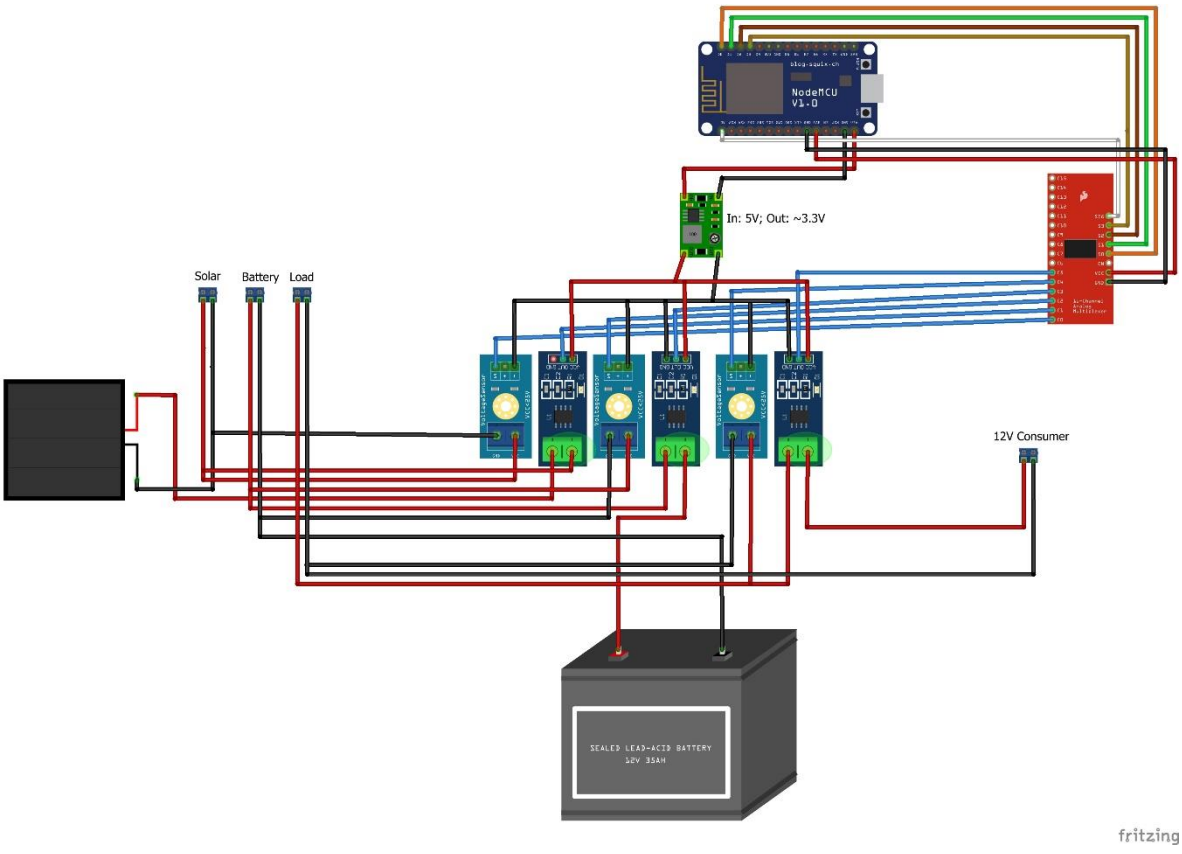

Abbildung 14: Ein Bild sagt mehr als hundert Worte, ein Fritzing-Diagramm mehr als tausend [Dateiname: Solarstation.fzz; siehe Repository]

Hierbei sei noch erwähnt, dass die Terminals „Solar“, „Battery“ und „Load“ den Solarladeregler repräsentieren, dass die Batterie nicht unbedingt eine Kapazität von 35 Ah haben muss, dass das „12V Consumer“-Terminal den anzuschließenden Verbraucher widerspiegelt und dass die

Stromversorgung des ESP8266 theoretisch parallel am „Load“-Terminal verschaltet ist. Letzteres befindet sich allerdings intern im Solarladeregler und soll damit nicht Gegenstand dieses Schaltplans sein.

Software

Arduino-Code

Der einzig relevante Code für diese Arbeit sind die folgenden Ausschnitte zur Messung. Der restliche Code ist zwar essentiell für die Arbeitsweise der Anlage, jedoch wurde dieser in einem anderen Projekt entwickelt und in diesem Projekt wiederverwendet. Ganz genau ist damit der Code gemeint, der dafür sorgt, dass der ESP8266 die Wifi-Credentials nicht hardcoded im Code erhält, sondern nach dem Flashvorgang per HTML-Formular entgegennimmt. Dieser Prozess macht sich den EEPROM zunutze, um auch nach einem Stromausfall oder einem Neustart des Controllers diese essentiellen Informationen zu behalten.

Der Vorgang zur Eingabe der Wifi-Credentials findet vereinfacht formuliert folgendermaßen statt: Wenn der ESP8266 startet, werden Daten aus dem EEPROM gelesen und versucht diese Daten in SSID, Passwort und Ziel-IP (in dem Fall der Raspberry Pi) aufzulösen. Wenn nach einer gewissen Zeit keine Verbindung zu dem gespeicherten Netzwerk hergestellt werden konnte, werden die Wifi-Credentials invalidiert, also auf leere Wörter bzw. Null-Pointer zurückgesetzt, in den EEPROM geschrieben und der ESP8266 startet neu. Wenn nach dem Neustart keine valide Ziel-IP gelesen wurde, startet der ESP8266 einen Soft-Accesspoint. Verbindet man sich per Mobilgerät mit dem AP, kann per Browser über die IP ‚192.168.1.1‘ ein HTML-Formular abgerufen werden. Trägt man in dieses Formular valide Daten ein, also SSID eines verfügbaren Wifi-Netzwerkes mit dem korrekten Passwort und eine valide IP-Adresse des Zielgeräts, können sie gespeichert werden. Es folgt eine weitere Seite zur Bestätigung der eingegeben Daten mit der Möglichkeit die Daten zu korrigieren oder abzuspeichern, also wieder in den EEPROM zu schreiben. Nach dem Speichern startet der ESP8266 neu, liest die just eingegeben Daten und versucht sich mit dem Netzwerk zu verbinden. Sollte der ESP8266 nicht in der Lage sein eine Verbindung mit dem Netzwerk innerhalb einer gewissen Zeit aufzubauen, werden die Credentials erneut invalidiert und der ESP8266 startet den Accesspoint erneut.

Messungen der Spannung

In Versuchen, die Spannung möglichst korrekt zu messen, wurde klar, dass die Sensoren keine korrekten Ergebnisse lieferten. Der ungenaue (interne) Analog-Digital-Converter des ESP8266 verschlimmerte diesen Sachverhalt zusätzlich.

Prinzipiell sind die verwendeten Spannungssensoren nichts weiter als Spannungsteiler, bestehend aus zwei unterschiedlichen Widerständen (30 kΩ als R1 und 7,455kΩ als R2).

Mit folgender Formel lässt sich die angelegte Spannung messen:

$$U_{angelegt} = U_{gemessen} * \frac{R_2}{R_1 + R_2}$$

Diese Formel funktioniert jedoch nur in der Theorie in ihrer Reinform. Um annehmbare Werte zu erhalten musste ich die Formel modifizieren. Die von mir verwendete Formel hat einen Dividenden, den ich aus einer Messreihe von Ist- und Sollwerten ermittelte. In meinen Tests versuchte ich auch über eine Differenz den vom Sensor gemessenen Wert näher an den tatsächlichen Spannungswert zu bringen, jedoch erwies sich ein Faktor als stabiler, wie folgender Auszug meiner Versuche zeigt.

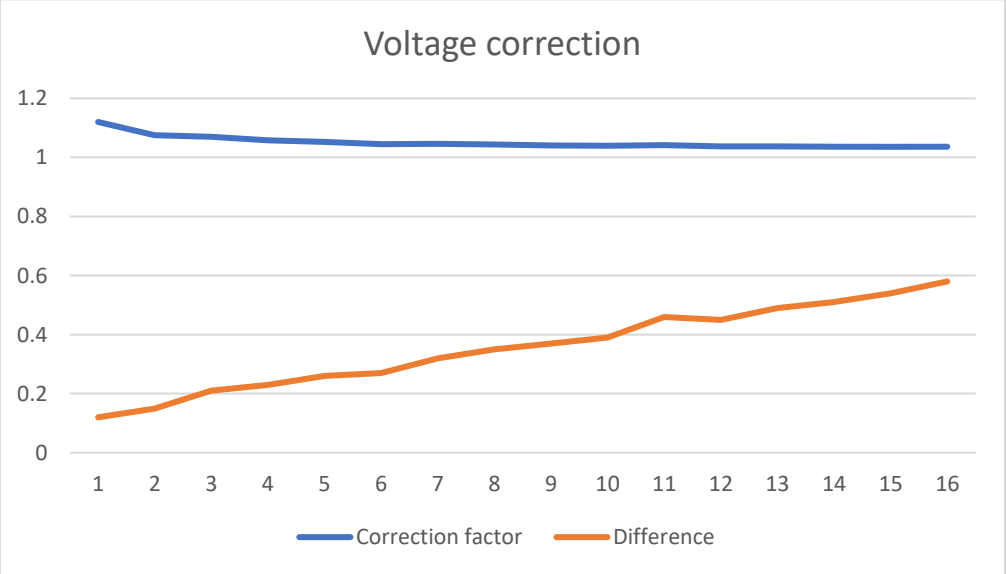

Abbildung 15: Diagramm der Faktoren und Differenzen zur Messungskorrektur der Spannung

Die X-Achse spiegelt die Spannung in Volt wider. Der blaue Graph zeigt den Wert durch den ich die Messung teilen musste, um an die korrekte Spannung zu gelangen. Der orangene Graph zeigt die Differenz des realen Wertes und des gemessenen Wertes. Das oben gezeigte Diagramm zeigt, dass der Faktor im Gegensatz zur Differenz relativ konstant bleibt, daher entschied ich mich für die

Verwendung ersteren, um die gemessenen Spannungen näher an die Realität zu bringen. Es sei an dieser Stelle noch erwähnt, dass der Spannungssensor höhere Spannungen ausgab, als erwartet wurde. Deswegen wird die Formel nicht mit dem Faktor multipliziert, sondern dividiert.

Die verwende Formel hatte schließlich folgende Struktur:

$$U_{angelegt} = \frac{U_{gemessen} * \frac{R_2}{R_1 + R_2}}{Korrektur}$$

Die Korrektur ist der Durchschnitt der einzelnen Korrekturen. Meine Messreihen ergaben als Korrektur den Wert 1,050922779.

Dies ist der im folgenden Code verwendete Korrektur-Wert. Die Methode `measureVoltage` ist für die Messung der Spannung zuständig. Die Signatur setzt sich aus den Parametern `int channel` für den zu verwendenen Channel des Analog-Multiplexers und `int repeat` für die Anzahl der zu machenden Messungen zusammen. Der Parameter `repeat` gibt an, wie viele Messungen an diesem Channel durchgeführt werden sollen. Die Messungen werden addiert und anschließend durch die Anzahl der Messungen dividiert, um stabile Messwerte zu erhalten und deren individuelle Schwankung zu eliminieren. Der Rückgabewert ist vom Typ `double`, um die Auflösung der Zahlen zu erhöhen und zusätzlich die Präzision zu erhöhen. In meinen Tests erhielt ich nach Anwendung all dieser Maßnahmen Werte die auf mindestens eine Nachkommastelle korrekte Spannungswerte zurückgab. Nachfolgend ist die vollständige Methode zur Spannungsmessung aufgeführt.

```
double measureVoltage (int channel, int repeat){

float vOUT = 0.0;
int value = 0;
float R1 = 30000.0;
float R2 = 7455.0;
double correctionfactor = 1.050922779;
double voltage = 0;
for (int i= 0; i<repeat; i++){
value = mux.read(channel);
vOUT = ((value * 3.33 ) / 1024.0)/correctionfactor;
voltage += vOUT / (R2/(R1+R2));
delay(10);
}
voltage = voltage/repeat;
return voltage;
}
```

Abbildung 16: Die Methode zur Spannungsmessung inkl. Korrekturfaktor

Messung der Stromstärke

Die Messung der Stromstärke erwies sich nicht nur als schwer, sondern auch als sehr ungenau. Ähnlich der Versuchsreihe zur Korrektur der Spannung führte ich Tests zur Findung eines ähnlichen Korrekturfaktors durch. Die Messungen zur Stromstärke konnten sich zu meinem Bedauern nur im Bereich von 0 bis 3 Ampere befinden, da das genutzte Labornetzteil lediglich diese Stromstärken liefern kann. Die Messschritte beliefen sich dadurch auf 0,5 Ampere, was in Summe 6 Messwerte ergibt. Folgendes Diagramm entsprang dem Test.

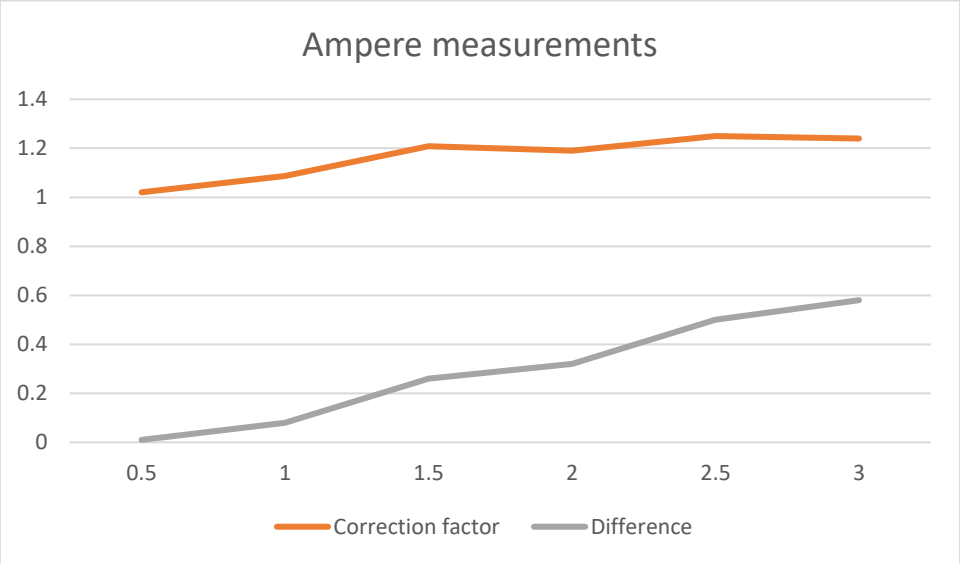

Abbildung 17: Diagramm der Faktoren und Differenzen zur Messungskorrektur der Stromstärke

Hierbei sei erwähnt, dass die X-Achse die Stromstärke in Ampere widerspiegelt. Auffallend ist, dass die Differenz immer weiter ansteigt und dementsprechend ungeeignet ist, um die Messungen zu präzisieren. Ebenfalls ist der Korrekturfaktor nicht besonders stabil. Da die Sensoren das elektromagnetische Feld messen, welches sich beim Stromfluss um einen Leiter aufbaut, sind die Messungen stark von umliegenden Magnetfeldern beeinflusst. Wenn also weitere stromdurchflossene Kabel um den Sensor herum geführt werden, sind die Messungen zusehends von Interferenzen betroffen und werden verfälscht. Da ich aber nicht vorhatte ein Labormessgerät zu entwickeln, fand ich mit dieser Präzision ab und versuchte die Messungen so gut es ging zu verbessern. Der Durchschnittswert zur Korrektur betrug nach meinen Tests 1,165918383.

Bei der Verwendung der ACS712-Sensoren ist zu beachten, dass sie die Hälfte der angelegten Spannung am Ausgangspin zurückliefern, wenn kein Strom gemessen wird. Wenn also 3,3 Volt Betriebspannung am Sensor anliegen, gibt der Messpin 1,65 Volt zurück. Die hat den Hintergrund, dass der Sensor positive und negative Stromstärken messen kann. Somit legt 1,65 Volt den Nullpunkt fest. Ich entschied mich, wie bereits erwähnt für die Verwendung der Sensorvarianten mit der Möglichkeit bis zu 30 Ampere (sowohl positiv als auch negativ) zu messen. Dies ist also auch gleichzeitig das Offset, das von der Berechnung abgezogen werden muss. Für jedes Ampere steigt oder sinkt die Spannung am Messungspin. Wichtig ist also hierbei genau zu messen, welche Spannung am Sensor anliegt. Meinen Messungen nach beträgt die Versorgungsspannung der ACS712-Sensoren 3,285 Volt. Dividiert man diese Spannung durch den Messbereich des Sensors, in meinem Fall also 60 Ampere (von -30 Ampere bis +30 Ampere) erhält man den Spannungsschritt pro Ampere.

Die Formel zur Stromstärkemessung lautet wie folgt:

$$I = \left(\left(\frac{U_{Messungspin}}{Spannungsdifferenz\ pro\ Ampere} \right) - Offset \right) * Korrektur$$

Nach vielen Versuchen die Stromstärkemessungen zu korrigieren, entschied ich mich für diese Struktur, da sie mir aus mir unbekannten Gründen, die besten und dennoch inkorrekten Werte lieferte.

Die folgende Methode ist für die Stromstärkemessung zuständig und beinhaltet auch den Korrekturfaktor zur Spannungsmessung. Sie nimmt die gleichen Parameter wie die Methode zur Spannungsmessung entgegen und bildet auch den Durchschnitt aus einer Abfolge von Messungen.

```
double measureCurrent (int channel, int repeat){

  double sensorVoltage=3.285;
  double voltageDifferencePerAmp = sensorVoltage/60;
  double voltageCorrectionFactor = 1.050922779;
  double currentCorrectionFactor = 1.165918383;
  double offset = 30.0;
  double current = 0;

  for(int i = 0; i < repeat; i++){
    int pinReading = mux.read(channel);
    double measuredVoltage = ((pinReading/1024.0)*3.3)/voltageCorrectionFactor;
    current += ((measuredVoltage/voltageDifferencePerAmp)-offset)*currentCorrectionFactor;
  }

  return current/repeat;

}
```

Abbildung 18: Die Methode zur Stromstärkemessung inkl. Korrekturfaktoren

Übertragung der Messwerte

Nach der Messung der Spannungen und Stromstärken an den drei Anschlüssen des Solarladereglers, werden diese in einer URL verpackt. Bei Aufruf dieser URL werden die Daten von einer Rest-API entgegen genommen. Die folgende Methode übernimmt diese Aufgabe.

```
void measureAndPost(){

  int repeat = 100;
  solarCurrent = measureCurrent(1, repeat);
  batteryCurrent = measureCurrent(3, repeat);
  loadCurrent = measureCurrent(5, repeat);

  solarVoltage = measureVoltage(0, repeat);
  batteryVoltage = measureVoltage(2, repeat);
  loadVoltage = measureVoltage(4, repeat);
  String url = "http://";
  url += credentials.getTargetIpString();
  url+=":";
  url+="5000/";
  url+="measurement/";
  url+= String(solarVoltage,2);
  url+=",";
  url+= String(solarCurrent,2);
  url+=",";
  url+= String(batteryVoltage,2);
  url+=",";
  url+= String(batteryCurrent,2);
  url+=",";
  url+= String(loadVoltage,2);
  url+=",";
  url+= String(loadCurrent,2);
  //Serial.println(url);
  http.begin(url);
  Serial.println(http.GET());
  http.end();

}
```

Die „2“ im Konstruktor der String-Objekte gibt an, dass die jeweilige Messung auf zwei Nachkommastellen gerundet werden soll. Die URL wird in einer ganz bestimmten Reihenfolge mit Messwerten zusammengebaut. Zu beachten ist, dass der Port in der URL dem der API gleicht.

REST-API

Die Schnittstelle zur Entgegennahme der Messwerte wurde in Python geschrieben und nutzt das Flask-Framework. Der Code dazu ist pythontypisch sehr simpel. Die Methode „writeMeasurements“ nimmt der Annotation (@app.route) entsprechend als Parameter die jeweiligen Messungen entgegen und schreibt diese „Measurements“ (InfluxDB-Terminologie) in die Datenbank „home“. Diese Datenbank kann jeden anderen Namen annehmen. Wichtig ist nur, dass Grafana später die richtige Datenbank ausliest, um auf die Werte zugreifen zu können.

```
@app.route('/measurement/<solarvoltage>,<solarcurrent>,<batteryvoltage>,<batterycurrent>,<loadvoltage>,<loadcurrent>')
def writeMeasurements(solarvoltage,solarcurrent,batteryvoltage,batterycurrent,loadvoltage,loadcurrent):
    influx_db.database.switch(database="home")

    solar_watts = float(solarvoltage)*float(solarcurrent)
    battery_watts = float(batteryvoltage)*float(batterycurrent)
    load_watts = float(loadvoltage) * float(loadcurrent)

    print(solar_watts, "\t", battery_watts, "\t", load_watts)
    influx_db.write_points(get_measurement_array('voltage', 'solar', solarvoltage))
    influx_db.write_points(get_measurement_array('voltage', 'battery', batteryvoltage))
    influx_db.write_points(get_measurement_array('voltage', 'load', loadvoltage))
    influx_db.write_points(get_measurement_array('current', 'solar', solarcurrent))
    influx_db.write_points(get_measurement_array('current', 'battery', batterycurrent))
    influx_db.write_points(get_measurement_array('current', 'load', loadcurrent))

    influx_db.write_points(get_measurement_array('power', 'solar', solar_watts))
    influx_db.write_points(get_measurement_array('power', 'battery', battery_watts))
    influx_db.write_points(get_measurement_array('power', 'load', load_watts))


    return 'ok'
```

Abbildung 19: API-Methode zur Entgegennahme der Messwerte

Die Methode nutzt zudem eine Hilfsmethode, um die Messwerte mit den nötigen Tags zu versehen und in eine Array-Struktur zu schreiben, die anschließend in der Influx-Datenbank persistiert werden kann. Zu beachten ist, dass beim Schreiben der Measurements kein Zeitstempel nötig ist. Dieses Problem wird von InfluxDB gelöst.

Zu beachten ist, dass aus den Messungen der Spannung und Stromstärke die Leistung nach der Formel

$$P = U * I$$

berechnet und ebenfalls persistiert wird.

Grafana

Nach der Befolgung des offiziellen Tutorials zur Installation von Grafana auf einem Raspberry Pi, muss lediglich ein Nutzer erstellt werden. Anschließend kann damit begonnen werden, das Dashboard mit Widgets zu füllen, die die Daten aus der Influx-Datenbank lesen. Genutzt werden hierfür die Tags, mit denen die Measurements vorher versehen wurden.

Je nach Geschmack lassen sich verschiedenste Darstellungsmöglichkeiten aus wählen. Ich entschied mich für folgendes Layout.

Abbildung 20: Beispiel für ein Dashboard

Das von mir erstellte Dashboard zeigt in der obersten Zeile die eingehenden Leistungen vom Solarpanel, der Batterie und die Leistungsentnahme an. Darunter befinden sich Skalen zur leichten Übersicht der jeweiligen Spannungen. Wiederum darunter befinden sich Diagramme, die mit grünen Graphen die Spannung und mit orangefarbenen Graphen die Stromstärke anzeigen.

Ein Beispiel, um zu zeigen, dass die Messungen einigermaßen aussagekräftig sind, möchte ich anhand eines Aussschnitts geben, der sich ergab als ich eine 13.000 mAh starke Powerbank auflud. Der folgende Graph veranschaulicht die Ladekurve der Powerbank anhand des Stromflusses an der Leistungsentnahme.

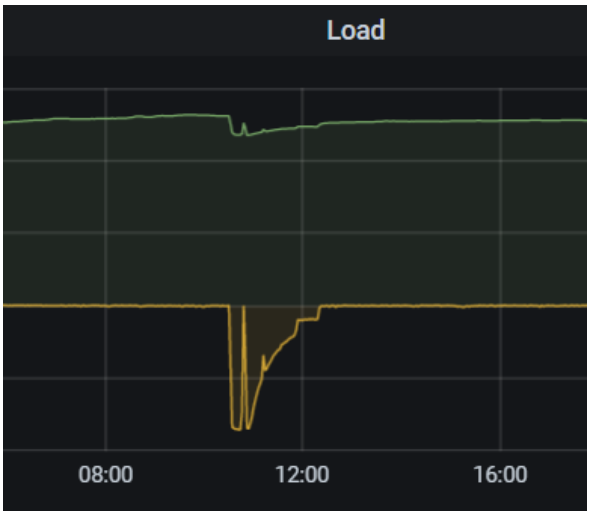

Abbildung 21: Ausschnitt der Leistungsentnahme beim Laden einer Powerbank

Fazit

Der Bau der hier dargestellten Anlage zur Speicherung von solarem Strom, hat mir viel gelehrt und geholfen zu verstehen, dass ein ausreichender Stromspeicher essentiell ist, um in Zeiten von Überfluss genug Strom für sonnenarme Tage aufnehmen und behalten zu können. Solarstrom ist in meinen Augen der größte Hoffnungsträger für eine Welt ohne fossile Brennstoffe. Jedoch müssen wir uns auch weiterhin mit der Problematik der Energiespeicherung auseinander setzen. Vielerorts wird Lithium-Ionen-Technologie genutzt, um viel Strom auf kleinem Raum zu speichern. Jedoch ist dies meiner Meinung nach nicht die Lösung für all unsere Probleme, da die Lithiumförderung, so wie die Förderung von Erdöl, mit massiven Eingriffen in die Natur einhergeht. Zwar nutze ich in meiner Anlage eine ausgediente Bleisäurebatterie, die heutzutage in so gut wie jedem Verbrennermotor-Automobil Verwendung findet, jedoch geht der Trend heutzutage meiner Beobachtung nach in Richtung Elektromobilität, sei es in Automobilen oder in zweirädrigen Fahrzeugen. Selbst in jedem modernen Akkuschraube sind Lithium-Zellen verbaut und das selbst in Billig-Geräten, die meist keine lange Lebensdauer haben. Nach mehreren sonnenarmen Tagen habe ich die Erfahrung gemacht, dass schnell der gesammelte Strom verbraucht ist, was auch mich dazu bewegt hat, eine gewisse Anzahl an Lithium-Zellen zu erstehen, um nach dem Vorbild vieler Hobby-Bastler einen großen Stromspeicher nach meinen Anforderungen zu bauen. Diese Zellen befanden sich in einem guten bis sehr guten Zustand und stammen aus unbenutzten Modem-Batterien. Ich möchte in Zukunft so gut es geht, den Strom aus meiner Anlage nutzen um Geräte zu laden, die normaler mit Steckdosenstrom geladen werden, um für mich einen kleinen Schritt in Richtung Unabhängigkeit zu gehen. Die von mir

entwickelte Konstruktion hilft mir sehr gut dabei, zu verstehen, wann der beste Zeitpunkt ist um Strombedarf zu decken. Ich bin im großen und ganzen zufrieden mit der Funktionalität der Anlage, jedoch habe ich auch Möglichkeiten zu Verbesserung im Auge. Beispielsweise kann die Stromstärkemessung verbessert werden, indem die Sensoren voneinander abgeschirmt werden um elektromagnetische Interferenzen zu minimieren. Auch können weitere Sensoren angeschlossen werden. Für mich wäre interessant, welchen Einfluss beispielsweise die Temperatur auf die Funktionsweise des Solarpanels hat. Softwaretechnisch könnte auch noch die Möglichkeit nachgerüstet werden, Kalibrierungswerte per Accesspoint in den EEPROM des Mikrocontrollers zu schreiben. Ich bin der Auffassung eine gute Basis entwickelt zu haben, auf der man aufbauen kann, um die Funktionweise und Effektivität von Solarladereglern zu überwachen und die Möglichkeiten eigenen Strom zu beziehen zu visualisieren.