Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN

Fernwartesystem für PV-Inselsysteme in Äthiopien

Bachelor-Thesis im Studiengang
Elektrotechnik und Informationstechnologie

Autor:
Benedikt Borter
benedikt.borter@stud.hslu.ch

Dozent:
Dr. Jonas Mühlethaler
jonas.muehlethaler@hslu.ch

Industriepartner:
Sahay Solar Verein Schweiz, Roger Buser
roger.buser@hslu.ch

Experte:
Martin Hässig
martin.haessig@swissgrid.ch

10. Juni 2022

Bachelor-Thesis an der Hochschule Luzern - Technik & Architektur

Titel	**Fernwartesystem für PV-Inselsysteme in Äthiopien**
Diplomandin/Diplomand	**Borter, Benedikt**
Bachelor-Studiengang	**Bachelor Elektrotechnik und Informationstechnologie**
Semester	**FS22**
Dozentin/Dozent	**Mühlethaler, Jonas**
Expertin/Experte	**Hässig, Martin**

Abstract Deutsch
Die vorliegende Arbeit befasst sich mit der Entwicklung eines Fernwarte- bzw. Monitoringsystems für Photovoltaik-Inselsysteme in Äthiopien. Der Auftraggeber ist Sahay Solar, ein Verein, der die Verbreitung der Solarenergie für Gesundheit und Entwicklung in Afrika fördert. Die Wartung der eingesetzten Anlagen gestaltet sich aufgrund der Geographie vor Ort als schwierig. Daher ist ein System gefragt, mit welchem der Zustand der Solaranlagen von einem zentralen Punkt aus überwacht werden kann, um allfällige Probleme zu identifizieren. Dazu wurden zu Beginn des Projektes die Anforderungen erfasst und ein passendes Konzept entwickelt. Dieses wurde in Form eines Prototyps umgesetzt und in Betrieb genommen. Der Prototyp zeichnet Ströme, Spannungen und weitere Messgrössen auf und überträgt diese in die Datenbank InfluxDB. Die Daten werden im Visualisierungstool Grafana dargestellt. Verschiedene Tests haben gezeigt, dass der Prototyp einsatzfähig ist, jedoch in einigen Bereichen noch über Verbesserungspotential verfügt.

Abstract Englisch
The present paper deals with the development of a remote monitoring system for photovoltaic stand-alone systems in Ethiopia. The project partner Sahay Solar is an association that promotes the dissemination of solar energy for health and development in Africa. The maintenance of the deployed installations is difficult due to the local geography. Therefore, a system is needed to monitor the state of the solar plants from a central point in order to identify any problems. For this purpose, the requirements were defined at the beginning of the project and a suitable concept was developed. It was implemented in the form of a prototype and put into operation. The prototype records currents, voltages and other values and transfers them to the InfluxDB database. The data is displayed in the visualization tool Grafana. Various tests have shown that the prototype is operational, but still has potential for improvement in some areas.

Ort, Datum Horw, 10.06.2022

© Benedikt Borter, Hochschule Luzern – Technik & Architektur

Alle Rechte vorbehalten. Die Arbeit oder Teile davon dürfen ohne schriftliche Genehmigung der Rechteinhaber weder in irgendeiner Form reproduziert noch elektronisch gespeichert, verarbeitet, vervielfältigt oder verbreitet werden.

Sofern die Arbeit auf der Website der Hochschule Luzern online veröffentlicht wird, können abweichende Nutzungsbedingungen unter Creative-Commons-Lizenzen gelten. Massgebend ist in diesem Fall die auf der Website angezeigte Creative-Commons-Lizenz.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Ausgangslage . . . 2
1.2 Aufgabenstellung . . . 2

2 Konzept **3**
2.1 Anforderungen . . . 3
2.2 Grobkonzept . . . 4
2.3 Lösungsfindung . . . 5
2.3.1 Datenerhebung . . . 5
2.3.2 GSM-Board . . . 5
2.3.3 SIM-Karte . . . 6
2.3.4 Speisung . . . 6
2.3.5 Empfangssystem . . . 7
2.4 Spezifikationen Prototyp . . . 8
2.5 Feinkonzept . . . 9

3 Implementation **10**
3.1 Sendesystem - Hardware . . . 10
3.1.1 Mainboard - Schema . . . 10
3.1.2 Mainboard - PCB-Layout . . . 11
3.1.3 RS-232-Board - Schema . . . 12
3.1.4 RS-232-Board - PCB-Layout . . . 13
3.1.5 Gehäuse . . . 13
3.2 Sendesystem - Software . . . 15
3.2.1 Konfiguration . . . 16
3.2.2 RS-232 . . . 17
3.2.3 GSM . . . 18
3.2.4 HTTP Post Request . . . 19
3.2.5 LEDs . . . 20
3.2.6 Setup und Loop . . . 20
3.3 Empfangssystem . . . 21
3.3.1 InfluxDB . . . 21
3.3.2 Grafana . . . 22
3.4 Kosten . . . 25

4 Inbetriebnahme **26**
4.1 Speisung . . . 26
4.2 RS-232-Verbindung . . . 26
4.3 Datenübertragung in die InfluxDB . . . 28
4.4 Datenmenge . . . 29
4.5 Anzeige der Energie . . . 31
4.6 Flash-Speicher . . . 31
4.7 Komplettes System . . . 32

5 Schlussfolgerungen **33**

Literatur **34**

Anhang **36**

1 Einleitung

Der Verein Sahay Solar ist im Jahr 2009 gegründet worden. Seither verfolgt er das Ziel, aktiv die Verbreitung von Solarenergie in Afrika, vorwiegend in Äthiopien, zu fördern. Durch die verschiedenen Aktivitäten des Vereins sollen nachhaltige Strukturen aufgebaut werden. Zu diesen gehört das Umsetzten von sozialen Solarprojekten und die Ausbildung im Gebiet der Solarenergie an Universitäten in Äthiopien.

Im Jahr 2018 leben gemäss UNDP (*United Nations Development Programme*) 67,3 % der ruralen äthiopischen Bevölkerung ohne Zugang zu Elektrizität [1]. Die Stromnetze auszubauen ist sehr kostspielig und macht aufgrund weiter Entfernungen in ländlichen Regionen oftmals wenig Sinn. Bereits bestehende Energiequellen wie z.B. Holz, Diesel oder Petroleumlampen sind schwierig zu beschaffen, teuer, umwelt- und gesundheitsschädlich. Der Mangel an Energiequellen hat Auswirkungen auf das gesamte öffentliche Leben, wie beispielsweise den Zugang zu Bildung oder die Sicherstellung der Gesundheitsversorgung. Durch eine funktionierende Stromversorgung kann an Schulen abends unterrichtet werden. Dadurch werden die Klassen verkleinert sowie die Bildung für alle, auch für die, die tagsüber mit anderen Arbeiten beschäftigt sind, zugänglich. Gesundheitszentren brauchen Strom, um Medikamente und Impfstoffe gekühlt zu lagern und mikroskopische Analysen im Labor durchzuführen. Zudem sinkt in elektrifizierten und beleuchteten Gesundheitszentren die Sterblichkeitsrate bei nächtlichen Geburten.

Neben den Solarprojekten setzt sich Sahay Solar auch für die Ausbildung vor Ort ein. Dabei führt der Verein mit Dozierenden, Studierenden und Interessierten *Solar Trainings* an Universitäten in Äthiopien durch. Zudem entsteht an der Arba Minch Universität ein Kompetenzzentrum für Solarenergie. Dadurch möchte man jungen Ingenieuren und Ingenieurinnen vor Ort die Chance geben, durch die Solarenergie eine wirtschaftliche Existenz aufbauen zu können [2].

Abbildung 1: Bei einem Gesundheitszentrum werden Solarzellen installiert [2]

1.1 Ausgangslage

An öffentlichen Gebäuden in Äthiopien hat Sahay Solar bereits 43 Solaranlagen installiert, der grösste Teil davon kommt in Gesundheitszentren zum Einsatz. Auch einige Grund- und Landwirtschaftsschulen sind ausgestattet worden. Zudem beteiligte sich der Verein an Trinkwasserprojekten. Dabei werden Solarpumpen für sauberes und sicheres Trinkwasser eingesetzt. Den Projektschwerpunkt setzt Sahay Solar auf die Stadt Arba Minch sowie die umgebende Gamu Gofa Zone, eine Region im Südwesten des Landes. Die Stadt verfügt über ein Stromnetz, die Versorgungslage ist allerdings unsicher und das Netz fällt regelmässig aus. In der umliegenden Gamu Gofa Zone sind die meisten Dörfer aufgrund der sehr bergigen Landschaft und der weiten Entfernungen zwischen den Dörfern gar nicht oder nur teilweise am elektrischen Netz angeschlossen. Damit sind die installierten Solaranlagen in dieser Region hilfreich und sinnvoll, jedoch bei Problemen und Defekten nur schwer erreichbar. Die Wartung, Reparaturen und Instandhaltung der Solaranlagen gestaltet sich dadurch als schwierig [2].

Abbildung 2: Gamu Gofa Zone in Äthiopien [3]

1.2 Aufgabenstellung

Um die Solaranlagen von einer zentralen Stelle aus zu überwachen und Probleme zu identifizieren, benötigt Sahay Solar ein Fernwarte- bzw. Monitoring-System, welches in diesem Projekt entwickelt wird. Das Ziel ist es, mit diesem System den Zustand der PV-Inselsysteme bestimmen zu können. Dazu muss das System verschiedene Grössen, z.B. Ströme und Spannungen von Batterien und Solarpanels messen und aufzeichnen. Auf einem Empfangssystem können diese Daten visualisiert werden. Zudem soll es universell für alle von Sahay Solar verwendeten Anlagen einsetzbar sein. Im Rahmen dieses Projektes wird somit ein Systemdesign erstellt, welches in Form eines Prototyps umgesetzt wird.

Wichtig dabei ist, dass das Empfangssystem ortsunabhängig ist. Dadurch können Mitglieder von Sahay Solar sowohl in der Schweiz als auch in Äthiopien die Daten abrufen und die Anlagen überwachen. Anhand der Messungen, welche im Empfangssystem ersichtlich sind, können so Probleme identifiziert werden, wie beispielsweise ein zu hoher Stromverbrauch, eine defekte Batterie oder beschädigte Solarpanels. Dadurch ist es möglich, den Prozess von Wartung, Reparaturen und Instandhaltung zu vereinfachen.

2 Konzept

Dieses Kapitel stellt das Konzept und dessen Entwicklung vor. Anhand der Aufgabenstellung ergeben sich die Anforderungen an das System, welche am Anfang dieses Teiles beschrieben werden. Darauf folgt das Grobkonzept, welches einen ersten Überblick über den Systemaufbau zeigt. Anschliessend wird die Problemstellung in verschiedene Teilprobleme unterteilt und die einzelnen Entscheide werden begründet. Daraus ergeben sich die Spezifikationen für den Prototyp sowie das Feinkonzept mit den diversen Systemkomponenten.

2.1 Anforderungen

Die folgende Anforderungsliste wurde mit dem Auftraggeber abgesprochen. Sie unterscheidet zwischen Festanforderungen (F), Mindestanforderungen (M) und Wunschanforderungen (W):

1. Allgemeine Anforderungen an das Fernwartesystem
   1.1. Das Fernwartesystem besteht aus zwei Subsystemen: das Sendesystem, welches sich bei der Solaranlage befindet und das Empfangssystem, welches ortsunabhängig ist. (F)
   1.2. Das System kann universell alle von Sahay Solar verwendeten PV-Anlagen überwachen. Das bedeutet, es muss kompatibel sein mit folgenden Anlagen und Komponenten: (F)
      - SMA Grid Inverter Sunny Boy *5000TL-21* / Sunny Boy *5.0*
        SMA Inverter Sunny Island *S16.0H-11* / *S16.0H-12*
        SMA Sunny Remote Control *SRC-20*
      - Victron EasySolar *48/5000/70-100 MPPT150/100 CCGX*
        Victron Interface *MK3-USB*
        Victron Battery Monitor *BMV 700*
        Victron Venus GX
      - Steca Solarix PLI *5000-48*
2. Anforderungen an das Sendesystem
   2.1. Zur Kompatibilität mit allen Wechselrichtern wird ein austauschbares Modul entwickelt, welches je nach Wechselrichter eingesetzt werden kann. Dieses Modul ist für die Kommunikation zwischen Sendesystem und Wechselrichter verantwortlich. (F)
   2.2. Das Sendesystem benutzt eine SIM-Karte, welche den Internetzugang ermöglicht. (F)
   2.3. Das Sendesystem kann folgende Daten messen bzw. aufzeichnen: (F)
      - Ausgangsspannung sowie Ausgangsstrom der Solarzellen
      - Batteriespannung und Batteriestrom
   2.4. Die Daten werden via Internet an das Empfangssystem versendet. (F)
   2.5. Es erfolgt mindestens eine Messung pro Stunde. Mindestens einmal pro Tag werden die Daten übertragen. (M)
   2.6. Die elektrische Speisung erhält das Sendesystem entweder vom Blei-Akku der PV-Anlage (48 VDC) oder vom Inselnetz (230 VAC). (F)
   2.7. Ist dieser leer, darf das System ausfallen, allerdings muss der Tiefentladeschutz gegeben sein. (F)
   2.8. Beim Ausfall der Speisung gehen keine Daten verloren. Nicht versendete Daten werden in einem nichtflüchtigen Speicher abgelegt und zu einem späteren Zeitpunkt, wenn die Speisung wieder vorhanden ist, versendet. (F)
3. Anforderungen an das Empfangssystem
   3.1. Das Empfangssystem kann die gemessenen Daten visualisieren. Dabei ist die Leistung der Solarzellen sowie der Batterieladezustand über die Zeit und die produzierte Energie pro Tag ersichtlich. (F)
   3.2. Für das Empfangssystem fallen keine monatlichen Gebühren an. (F)
   3.3. Im Empfangssystem kann das Mess- und Sendeintervall eingestellt werden. (W)

2.2 Grobkonzept

Anhand der erwähnten Anforderungen und der Architektur eines PV-Inselsystems wird das Grobkonzept aufgezeichnet. Es wird in Abbildung 3 am Beispiel des Steca Solarix PLI aufgezeigt. Der untere Teil des Blockschaltbildes zeigt das bestehende PV-Inselsystem. Dazu gehören die PV-Module, die Batterie, die elektrische Last und der Steca Solarix PLI. Hier übernimmt der Steca Solarix PLI die Aufgabe des Ladereglers und des Wechselrichters. Darum sind die PV-Module, die Batterie und die elektrische Last direkt an ihm angeschlossen. Dieses PV-Inselsystem wird durch zusätzliche Hardware (1), durch das Sendesystem, erweitert. Dieses ermöglicht den Zugang ins Internet, um Daten der PV-Anlage in eine Datenbank (2), welche sich in der Cloud befindet, zu schreiben. So können diese Daten anschliessend ortsunabhängig vom Client (3) abgerufen werden. Die Datenbank in der Cloud sowie die Anzeige auf dem Client bilden gemeinsam das Empfangssystem. In den folgenden Kapiteln werden die beiden Systeme sowie deren Komponenten im Detail hergeleitet und erläutert.

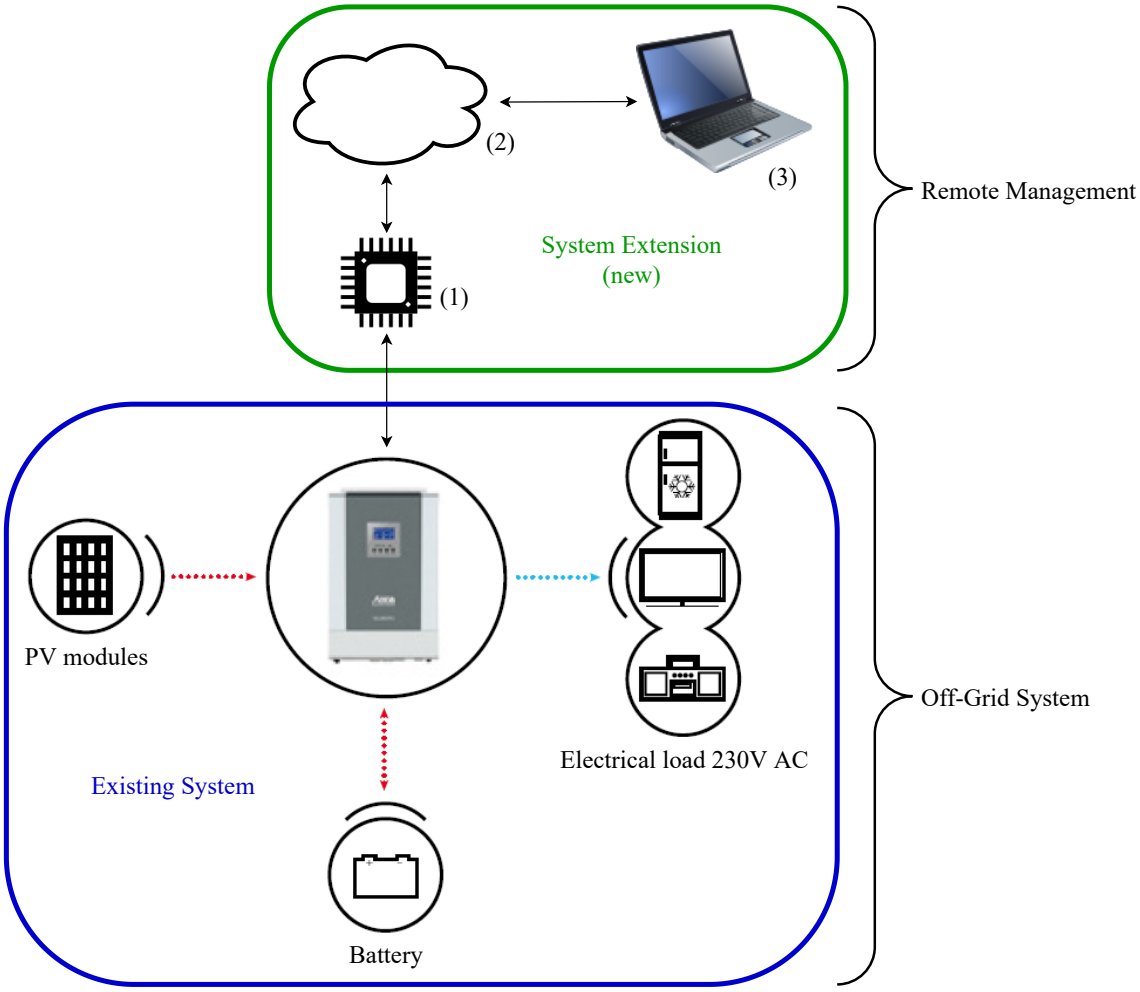


Abbildung 3: Blockschaltbild des Inselnetzes mit Fernwartesystem [4]

2.3 Lösungsfindung

In diesem Abschnitt wird die Problemstellung unterteilt. Bei den jeweiligen Teilproblemen werden Entscheide getroffen, welche für das Design des Prototyps ausschlaggebend sind. Es handelt sich um die Themen Datenerhebung, Zugang zum GSM-Netz, Speisung und den Aufbau des Empfangssystems.

2.3.1 Datenerhebung

Um Messgrössen der Solaranlage aufzuzeichnen, gibt es zwei Varianten. Entweder misst das Sendesystem die Grössen selber oder es kommuniziert mit dem Wechselrichter, da dieser bereits verschiedene Messungen durchführt. Das Diagramm in Abbildung 3 zeigt, dass das Sendesystems die Ströme und Spannungen nicht selber misst, sondern mit dem Wechselrichter kommuniziert. Dies hat folgende Gründe:

- Die Entwicklung der Elektronik wäre viel aufwendiger, wenn das Sendesystem die Messungen selber ausführen müsste.
- Durch die Kommunikationsschnittstelle kann die Komplexität der Hardware im Vergleich zu Messschaltungen massiv reduziert werden.
- Die Ströme und Spannungen werden bereits von den Wechselrichtern und Ladereglern gemessen und können über eine serielle Schnittstelle abgefragt werden. So es nicht nötig, nochmals dieselben Grössen zu messen.
- Die Verkabelung und das Anschliessen des Sendesystems wäre komplizierter, da für die Messungen mehr Verbindungen notwendig wären als bei der Kommunikation.

Bei den Systemen, in denen der Steca Solarix PLI eingesetzt wird, läuft die serielle Kommunikation über eine RS-232-Schnittstelle [5]. Mit den Produkten von Victron kann hingegen über TCP/IP, RS-232 oder CAN-Bus kommuniziert werden [6], je nachdem mit welchem Gerät man sich verbinden will. SMA nutzt beispielsweise beim Inverter Sunny Boy 5.0 den Feldbus Speedwire [7]. All diese verschiedenen Schnittstellen sind allerdings eine Herausforderung an die Hardware. Damit auf der Hardware nicht mehrere Schnittstellen implementiert werden müssen und dann jeweils nur eine in Gebrauch ist, soll die Schnittstelle durch ein austauschbares Modul realisiert werden. So kann später nur dieses Modul ausgewechselt und die Software angepasst werden, um das Gerät bei den verschiedenen Anlagen einzusetzen.

2.3.2 GSM-Board

Um den Internetzugang zu ermöglichen, benötigt das Sendesystem ein Mobilfunk- bzw. GSM-Modul. WLAN ist keine Option, weil im Einsatzgebiet in der Regel keine WLAN-Netze vorhanden sind. Das Mobilfunknetz in Äthiopien unterstützt zum grössten Teil den Mobilfunkstandard 3G und zum Teil auch 4G [8]. Daher soll das GSM-Modul sicher die 3G-Technologie unterstützen. Auf dem Markt existieren verschiedene IoT-Boards, welche über einen Mikroprozessor gemeinsam mit einem GSM-Modul verfügen. In diesem Projekt wird das Arduino MKR GSM 1400 benutzt, ein IoT-Board vom Hersteller Arduino. Es wird verwendet, weil die Produkte von Arduino einfach und schnell eingesetzt werden können. Zudem existieren viele Software-Libraries, welche benutzt werden können und für die Softwareentwicklung hilfreich sind. Auf verschiedenen Foren findet man viele Beispielprojekte und Communities, welche bei Problemen weiterhelfen können. Um die Messwerte in einem nichtflüchtigen Speicher abzulegen, kann der interne Flash-Speicher von 256 Kilobytes benutzt werden.

2.3.3 SIM-Karte

Für die Verbindung ins Internet benötigt das Arduino MKR GSM 1400 eine SIM-Karte. Da Arduino selber auch IoT-SIM-Karten anbietet, ist diese Lösung naheliegend. Allerdings ist Äthiopien im Netz von Arduino SIM nicht abgedeckt [9], was für diese Anwendung essenziell wäre. Darum werden SIM-Karten des Anbieters Hologram ausgewählt, da Äthiopien in deren Netzabdeckung inbegriffen ist [10]. Die Kosten betragen mit dem *Global Flexible Data Plan* monatlich 0,70 $ pro SIM-Karte, hinzu kommen 0,19 $ pro Megabyte Datenverbrauch [11].

2.3.4 Speisung

Grundsätzlich wird eine Gleichspannung von 5 V benötigt, um das Arduino zu speisen. Daher stellt sich die Frage, ob das Sendesystem vom Blei-Akku der PV-Anlage (48 VDC) oder vom Inselnetz (230 VAC) gespiesen werden soll. Dabei muss die Anforderung bezüglich des Tiefentladeschutzes beachtet werden. Aufgrund folgender Argumente wird das Sendesystem am Inselnetz angeschlossen:

- Dadurch, dass der Wechselrichter den Akku bereits gegen Tiefentladung schützt und das 230 VAC-Netz allenfalls ausschaltet, kann das angeschlossene Gerät den Akku im tief entladenen Zustand nicht weiter belasten. So ist keine zusätzliche Schutzschaltung nötig.
- Die Elektronik auf dem Sendesystem ist weniger komplex und dadurch weniger aufwendig, da es keine zusätzliche Schaltung zum Tiefentladeschutz braucht.
- Mit einem Netzteil ist es sehr einfach, das Gerät anzuschliessen und zu versorgen.
- Die Schwellspannungen, mit welchen der Wechselrichter arbeitet um die Batterie zu schützen, sind nicht bekannt. Daher könnte es bei einer eigenen Tiefentladeschutzschaltung vorkommen, dass das Sendesystem bereits ausgeschaltet und das Netz weiter in Betrieb ist oder umgekehrt.
- Bei einer eigenen Tiefentladeschutzschaltung würde die Batterie im leeren Zustand stärker belastet werden, da gleichzeitig zwei Schaltungen, die vom Wechselrichter und die des Sendesystems, aktiv wären.
- Preislich unterscheiden sich die beiden Varianten kaum. Entweder bräuchte man einen DC-DC-Wandler, welcher die Spannung von 48 VDC auf 5 VDC regelt oder ein gewöhnliches Netzteil mit einer Ausgangsspannung von 5 VDC.

2.3.5 Empfangssystem

Hier sind grundsätzlich zwei Software-Tools gefragt. Eines ist für die Speicherung der Daten verantwortlich und das andere stellt die Daten graphisch dar. Als Datenbank wird InfluxDB verwendet und als Tool zur Darstellung Grafana. Beide Tools sind für IoT-Anwendungen (*Internet of Things*) gemacht und daher passend für diese Anwendung. Da Grafana und InfluxDB zwei verbreitete IoT-Tools sind, sind sie aufeinander abgestimmt. So können die Daten aus der InfluxDB einfach in Grafana verarbeitet und dargestellt werden. Hinzu kommt der Vorteil, dass beides Open-Source-Anwendungen sind. Dadurch kann man sie in der Cloud nutzen oder auf eigener Hardware installieren und betreiben. Nutzt man sie in der Cloud, gibt es bei den kostenlosen Versionen jeweils einige Einschränkungen:

- Grafana Cloud [12] :
  - Speicherlimit: 10 000 Messreihen bzw. Serien
  - Speicherdauer: 14 Tage
  - Maximale Anzahl Benutzer: 3
- InfluxDB [13] :
  - Speicherlimit: 10 000 Messreihen bzw. Serien
  - Speicherdauer: 30 Tage
  - Schreibbefehle: 5 MB/min
  - Datenbankabfrage: 300 MB/min

Die Einschränkungen der Grafana Cloud bezüglich Speicherlimit und -dauer sind hier allerdings nicht ausschlaggebend, da die Daten in der InfluxDB gespeichert werden. Das Maximum von 10 000 Messreihen stellt ebenfalls kein Problem dar. Werden beispielsweise pro Solaranlage zehn Grössen gemessen, können insgesamt 1000 Anlagen überwacht werden. Die einzige kritische Einschränkung bezüglich der Anwendung in diesem Projekt ist, dass die InfluxDB Daten nur 30 Tage lang speichert, ältere Werte werden gelöscht. Umgehen kann man dies, indem man die InfluxDB auf einem eigenen Server installiert und betreibt. Grafana kann dann weiterhin in der Cloud laufen und auf diese Daten zugreifen, sofern der Server korrekt aufgesetzt wird.

2.4 Spezifikationen Prototyp

Aufgrund der Anforderungen und der beschriebenen Lösungen sind folgende Spezifikationen für den Prototyp, welcher im Rahmen dieses Projektes entwickelt und gebaut wird, festgelegt. Sie wurden mit dem Auftraggeber abgesprochen.

1. Allgemein
   - 1.1. Für das Sendesystem wird ein Hardware-Prototyp auf einer Leiterplatte mit einem Gehäuse entwickelt und realisiert.
   - 1.2. Das Empfangssystem wird in der Cloud aufgesetzt und betrieben.
2. Sendesystem
   - 2.1. Als modulare Schnittstelle wird die RS-232-Schnittstelle zum Steca Solarix PLI *5000-48* realisiert. So kann diese später durch eine andere Schnittstelle für die weiteren verwendeten Wechselrichter ausgetauscht werden.
   - 2.2. Der Zugang zum Internet wird durch das Arduino MKR GSM 1400 ermöglicht.
   - 2.3. Das Arduino verwendet eine IoT-SIM-Karte vom Anbieter Hologram.
3. Empfangssystem
   - 3.1. Zur Speicherung der Daten wird die Datenbank InfluxDB benutzt. Diese läuft in diesem Fall in der Cloud, bietet aber zusätzlich die Möglichkeit, auf einem eigenen Server installiert und betrieben zu werden.
   - 3.2. Zur Darstellung und Visualisierung der Daten wird die plattformübergreifende Open-Source-Anwendung Grafana verwendet.

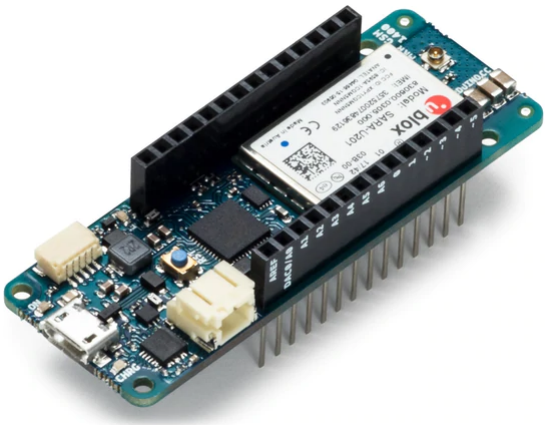


Abbildung 4: Arduino MKR GSM 1400 [14]

2.5 Feinkonzept

Die Übersicht in Abbildung 5 zeigt, wie der Prototyp des Fernwartesystems aufgebaut ist. Das eingerahmte Sendesystem zeigt die verschiedenen Elemente der Hardware. Gespiesen wird es via Netzteil (AC Adapter) und erhält dabei direkt die vom Arduino benötigten 5 V. Damit wird auch der Rest der Elektronik versorgt. Zusätzlich befindet sich ein 3,3 V-Regler auf dem Board. Dieser wird in diesem Fall, bei dem die RS-232-Schnittstelle verwendet wird, nicht gebraucht, da diese ebenfalls mit 5 V versorgt wird. Allerdings ist es möglich, dass die RS-485-Schnittstelle, die später noch hinzugefügt werden kann, 3,3 V benötigt. Die RS-232-Schnittstelle wird daher auf einer separaten Platine implementiert, damit sie ausgetauscht werden kann. Sie erhält die Daten der UART (*Universal Asynchronous Receiver / Transmitter*) des Arduinos und übersetzt sie mit einem Pegelwandler auf auf die RS-232-Spannungspegel. Zusätzlich sind einige LEDs vorhanden, damit das Arduino Statusinformationen abgeben kann. So kann es beispielsweise anzeigen, ob es Netzempfang hat oder nicht. Die Verbindung in die Datenbank läuft über das Internetkommunikationsprotokoll HTTPS (*Hypertext Transfer Protocol Secure*). Damit werden die Daten vom Arduino in die InfluxDB übertragen. Sind die Daten in der Datenbank abgelegt, kann Grafana darauf zugreifen und die Daten darstellen. InfluxDB und Grafana, welche gemeinsam das Empfangssystem bilden, laufen beide in der Cloud.

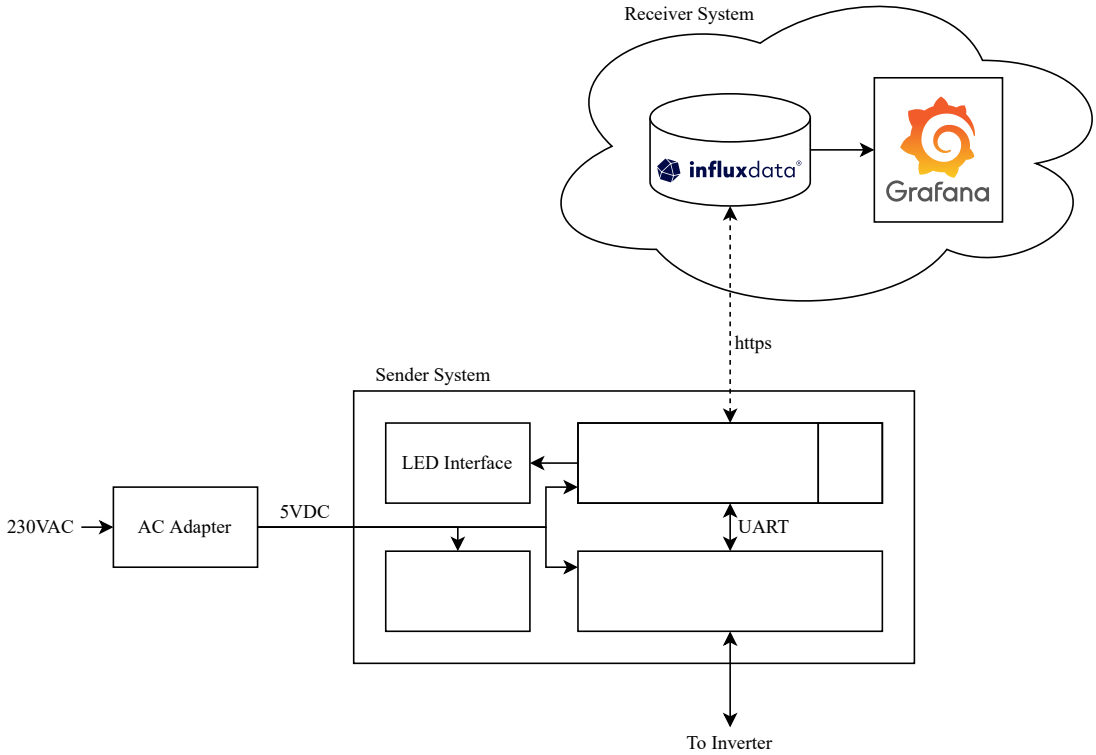


Abbildung 5: Blockschaltbild des Fernwartesystems

3 Implementation

In diesem Kapitel wird die technische Umsetzung des Konzepts in Hard- und Software detailliert beschrieben. Dazu gehören elektrische Schemas, Ausschnitte aus dem Code usw. Zudem werden die Kosten der Hardware aufgeführt.

3.1 Sendesystem - Hardware

Die Hardware des Sendesystems besteht aus zwei Leiterplatten. Zum einen das Mainboard, welches die Speisung und das Arduino beinhaltet, zum anderen das RS-232-Board, welches die RS-232-Schnittstelle bildet. So kann diese später beispielsweise durch eine RS-485-Schnittstelle ausgetauscht werden.

3.1.1 Mainboard - Schema

Speisung

Als Netzteil wird der AC-Adapter *SWH15-5B-EB-P5* vom Hersteller CUI Inc. verwendet. Die Ausgangsspannung beträgt 5 V und es kann einen maximalen Ausgangsstrom von 3 A liefern. Dieser Ausgangsstrom ist notwendig, weil das Arduino Spitzenströme von etwa 2 A braucht, um beispielsweise ein GSM-Handshake durchzuführen [14]. Die Buchse J1 in Abbildung 6 ist der Eingang, bei dem das Netzteil angeschlossen wird. Mit der LED3 wird angezeigt, dass das Gerät gespiesen ist. Anschliessend folgt der Spannungswandler, welcher die 3,3 V generiert. Hierzu wird der Regler *TLS850B0TBV33* benutzt, welcher für einen maximalen Ausgangsstrom von 500 mA ausgelegt ist. Er besitzt einen Enable-Eingang. Dieser wird durch einen digitalen Ausgang des Arduinos gesteuert und ist high-aktiv. Daher wird der Pull-Down Widerstand R4 eingesetzt, damit der Regler ausgeschaltet ist, falls kein Signal angelegt wird. Die Kapazitäten C1, C2 und C4 sind anhand des Datenblatts dimensioniert.

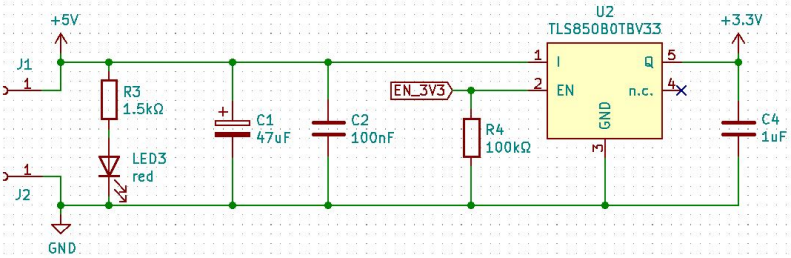


Abbildung 6: Schema des Spannungsreglers

Arduino

Abbildung 7 zeigt das Schema des Arduino MKR GSM 1400. Die Speisung erhält es über den Pin VIN (27). Der +5 V-Pin (28) hingegen ist der Ausgang der USB-Versorgungsspannung, sobald das Arduino über den Micro-USB-Port versorgt wird. Deshalb dürfen Pin 27 und Pin 28 nicht miteinander verbunden werden, da beide Ausgänge einer Spannungsquelle sind. Die Schotky-Diode D1 sorgt dafür, dass das Mainboard auch durch den USB-Port gespiesen werden kann und dabei das Netzteil nicht angeschlossen werden muss. Hinzu kommen zwei Entkopplungskondensatoren C3 und C5. Durch die beiden LEDs kann das Arduino über seinen Zustand informieren, beispielsweise ob Netzempfang vorhanden ist. Mit dem Signal *EN_3V3* kann der Spannungsregler ein- und ausgeschaltet werden. Auf die beiden Buchsenleisten J3 und J4 wird das RS-232-Board aufgesteckt. Daher werden die beiden Versorgungsspannungen 5 V und 3,3 V darauf verbunden. Wichtig in diesem Fall sind die RX- und TX-Leitungen, über welche die UART-Kommunikation läuft. Zusätzlich werden noch weitere Ein- und Ausgänge des Arduinos auf die Buchsenleisten verbunden, für den Fall, dass für weitere Schnittstellen, wie beispielsweise RS-485, mehr Signale benötigt werden.

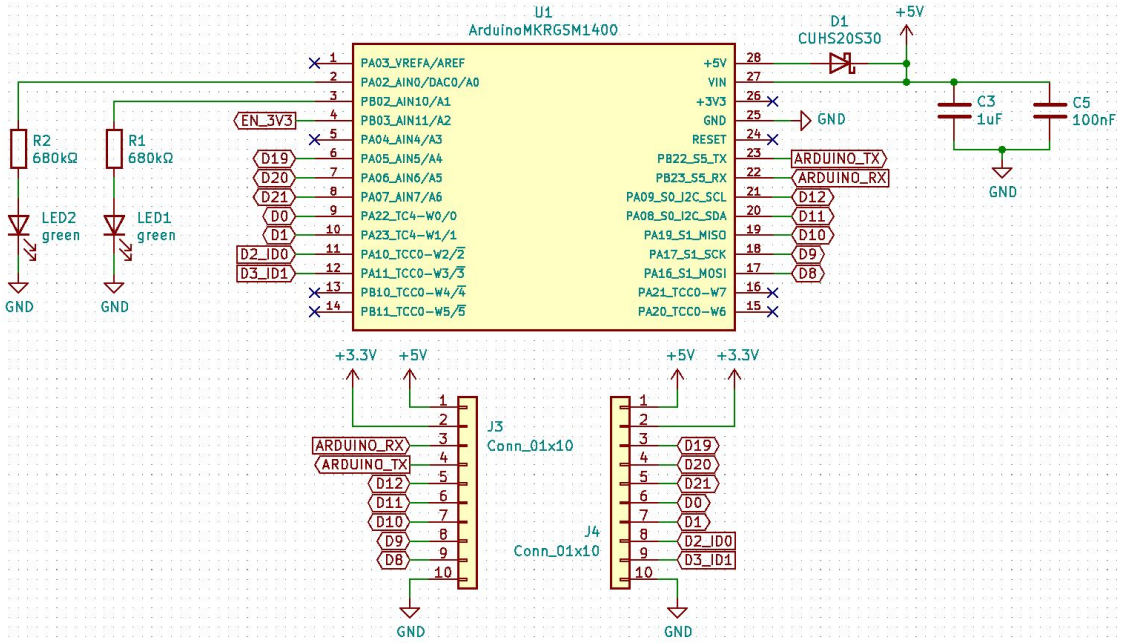


Abbildung 7: Schema vom Arduino MKR GSM 1400 mit Verbindung zum RS-232-Board

3.1.2 Mainboard - PCB-Layout

Das Layout des Mainboards ist in Abbildung 8 dargestellt. Die Aussenmasse der Leiterplatte betragen 54,6 mm × 80 mm Auf der oberen Seite wird das Arduino aufgesteckt, darunter das Kommunikationsmodul, in diesem Falle das RS-232-Board und rechts davon befinden sich die Anschlüsse der LEDs und der Spannungsregler. Der Ausschnitt der oberen rechten Ecke bietet Platz, damit die LEDs ins Gehäuse montiert werden können. So können die LEDs im Gehäuse fix verbaut und mit Kabel angeschlossen werden.

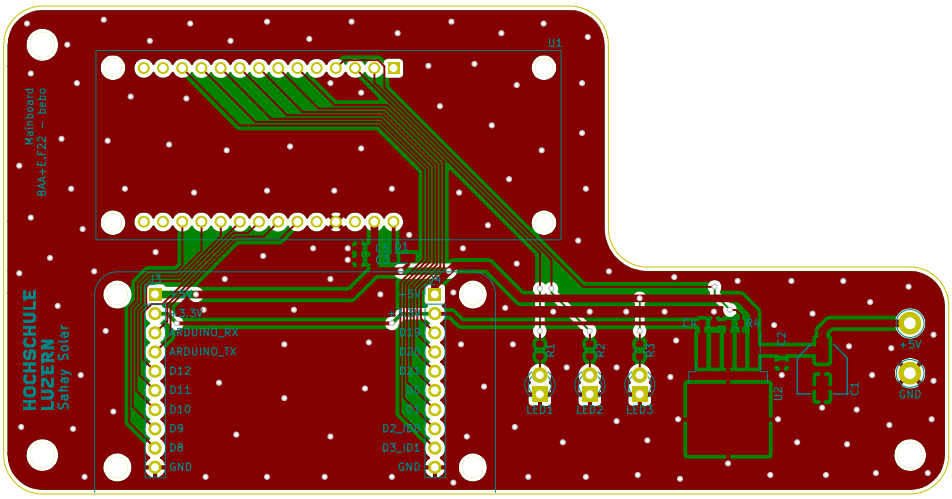


Abbildung 8: PCB-Layout des Mainboards

3.1.3 RS-232-Board - Schema

Die RS-232-Schnittstelle in Abbildung 9 wird mit dem integrierten Schaltkreis *MAX232* realisiert. Dieser dient als Pegelwandler, um die Signale zwischen RS-232 (bis zu ±15 V) und TTL (*Transistor–transistor logic*, 5 V) zu konvertieren. Der *MAX232* benötigt für interne Ladungspumpen einige externe Kondensatoren. Diese werden gemäss dem *Typical Operating Circuit* aus dem Datenblatt dimensioniert und angeschlossen. Nach dem *MAX232* gehen die Signale auf den 9-poligen D-Sub-Steckverbinder J3, bei dem das serielle Kommunikationskabel zum Steca Solraix PLI angeschlossen wird.

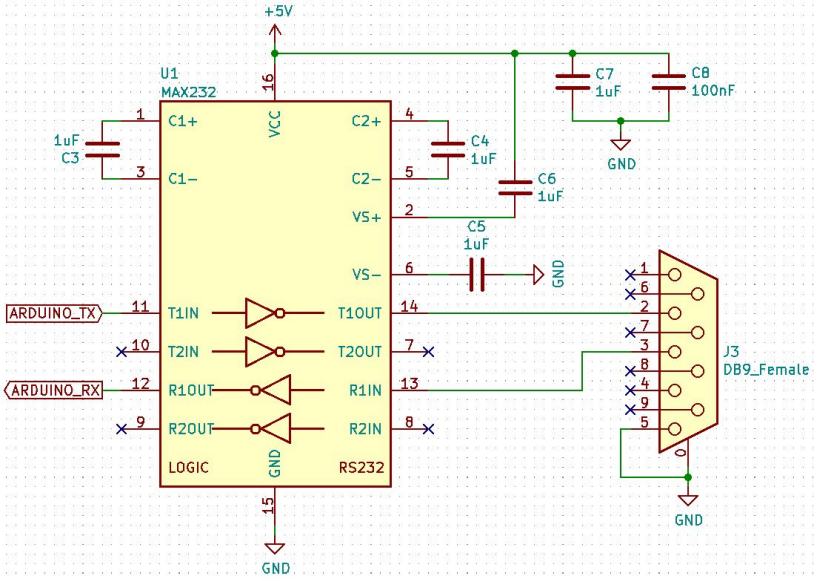


Abbildung 9: Schema der RS-232-Schnittstelle

Abbildung 10 zeigt die Steckerleisten des RS-232-Boards, mit denen es auf das Mainboard gesteckt wird. Durch die Pull-Down Widerstände R1 und R2 werden zwei digitale Eingänge des Arduinos auf Ground gezogen. Damit kann das Arduino die Kommunikationsstelle identifizieren. Wird beispielsweise eine RS-485-Schnittstelle implementiert, sollte darauf mindestens einer der beiden Widerstände auf 5 V verbunden werden. Dadurch wäre es möglich, dass die Software selbstständig erkennt, welche Kommunikationsschnittstelle aufgesteckt ist.

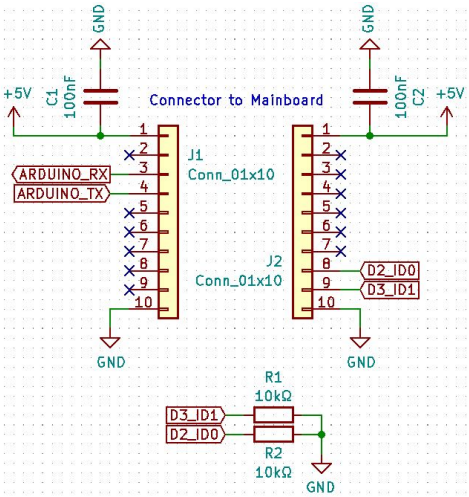


Abbildung 10: Schema der Verbindung zum Mainboard

3.1.4 RS-232-Board - PCB-Layout

In Abbildung 11 ist das Layout des RS-232-Boards ersichtlich. Die Leiterplatte ist 32 mm × 53 mm gross. Der D-Sub-Steckverbinder befindet sich am unteren Rand des PCBs, damit er durch eine Öffnung in der Gehäusewand herausgeführt werden kann. Auf den Seiten befinden die Steckerleisten, mit denen das Modul auf das Mainboard aufgesteckt wird.

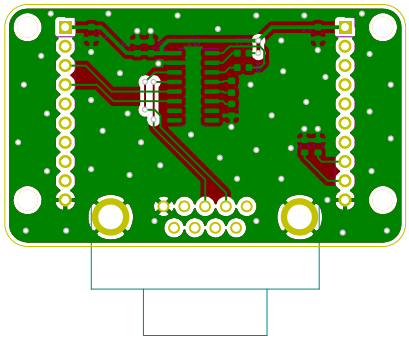


Abbildung 11: PCB-Layout des RS-232-Boards

3.1.5 Gehäuse

Das Gehäuse *1591XXDFLBK* vom Hersteller Hammond Manufacturing ist 176,15 mm × 82,4 mm × 50,5 mm gross. Mit seinem Flanschdeckel kann das Gerät gemeinsam mit den anderen Komponenten der Solaranlage an einer Wand befestigt werden. Abbildung 12 zeigt das komplette Sendesystem im Gehäuse. Zu erkennen sind die drei LEDs auf der Frontseite, der runde Spannungsversorgungssteckverbinder und der D-Sub-Steckverbinder für die RS-232-Verbindung zum Steca Solarix PLI. Die detaillierten Pläne für die Aussparung und die Bohrlöcher im Gehäuse befinden sich im Anhang.

Abbildung 12: Aussenansicht des vollständigen Sendesystems

Die Antenne des GSM-Moduls wird auf der Innenseite an die Wand geklebt und das Mainboard in drei Löchern im Gehäusedeckel angeschraubt. Da die grösste mechanische Einwirkung am D-Sub-Steckverbinder aufkommen kann, wird auch das RS-232-Board mit vier Schrauben befestigt. Der Spannungsversorgungssteckverbinder befindet sich in der Seitenwand und wird durch zwei Litzen mit Kabelschuhen auf das Mainboard verbunden. Dies zeigt die Innenansicht in Abbildung 13. Gut erkannbar ist das Mainboard, die grosse Leiterplatte, welche sich zuunterst befindet. Auf das Mainboard aufgesteckt werden das Arduino, das blaue Board und das RS-232-Board mit dem D-Sub-Steckverbinder.

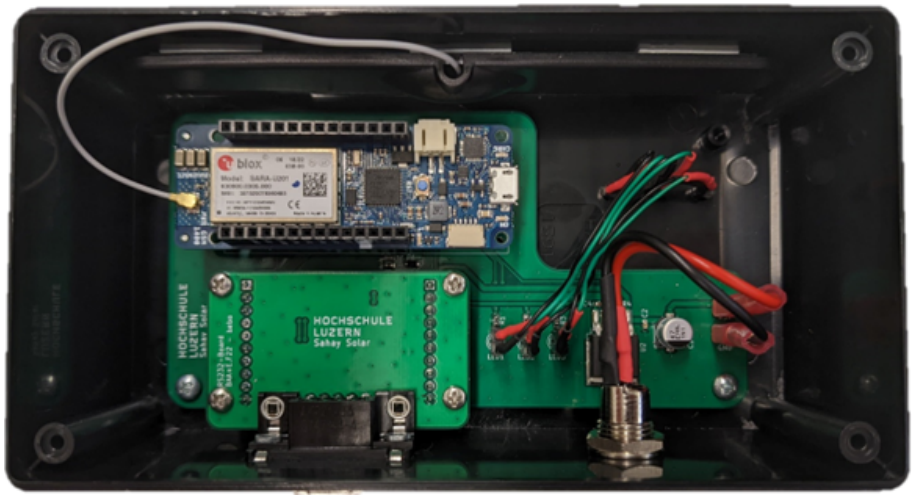


Abbildung 13: Innenansicht des Gehäuses mit montierter Elektronik

3.2 Sendesystem - Software

Die Software, die auf dem Arduino läuft, wird in der Arduino IDE 1.8.19 entwickelt und mit dieser auf das Arduino Board geladen. Sie befolgt den Ablauf in Abbildung 14. Zu Beginn wird die Hardware initialisiert und die Verbindung zum GSM-Netz hergestellt. Anschliessend folgt periodisch die Statusabfrage über RS-232 an den Steca Solarix PLI, welcher darauf mit den geforderten Messwerten antwortet. Diese werden auf dem Arduino gespeichert. Sobald zehn Messungen gesammelt sind, werden diese gemeinsam in die Datenbank übertragen. Am Ende der Schleife wird immer solange gewartet, bis eine Minute seit der letzten RS-232-Anfrage vergangen ist. So kann sichergestellt werden, dass die Messgrössen in einem regelmässigen Zyklus, in diesem Fall jede Minute, aufgezeichnet werden, auch dann, wenn die Werte in die Datenbank übertragen werden, was etwa 5-10 Sekunden dauern kann.

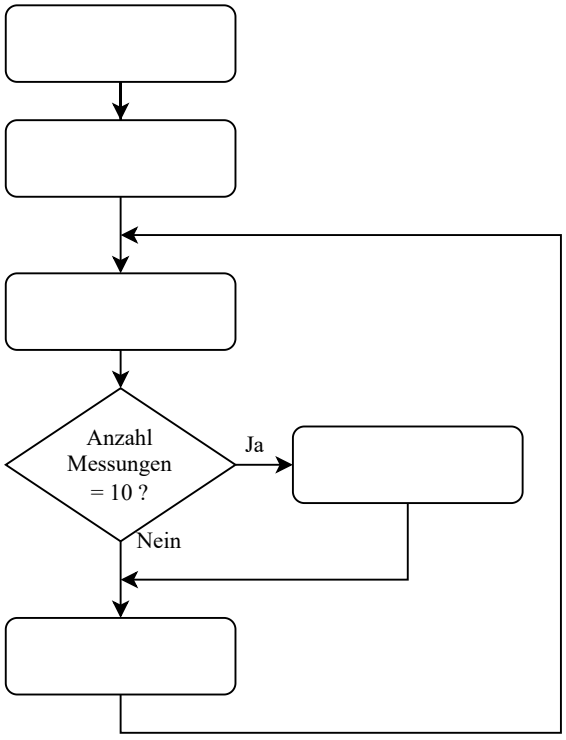


Abbildung 14: Flussdiagramm der Software des Sendesystems

Abbildung 15 zeigt, in welche Module die Software unterteilt ist und wie diese voneinander abhängig sind. Zuoberst in der Grafik befindet sich die Hauptdatei *Arduino_SahayRemoteFinal*, welche mit den Funktionen *setup()* und *loop()* das Hauptprogramm darstellt. Das Modul rs232.h ist für die Kommunikation zum Steca Solarix PLI verantwortlich, influxdb.h für die GSM-Verbindung und das Übertragen in die Datenbank. Beide greifen auf das Modul leds.h zu, um je ihren Status mit einer LED anzuzeigen. Informationen wie Pin-Nummer der SIM-Karte und Passwörter werden in der separaten Datei arduino_secrets.h gespeichert. Diese wird daher nur vom Moudl influxdb.h genutzt. Es verwendet zudem die öffentlichen Libraries MKRGSM.h und ArduinoBearSSL.h für die Internetverbindung sowie die Datenübermittlung in die InfluxDB. Um den Zeitstempel einer Messung zu generieren, greift das Modul rs232.h zum Zeitpunkt der Statusabfrage auf eine Funktion im Modul influxdb.h zu. Die Datei config.h ist für die ganze Anwendung von Bedeutung, da darin verschiedene Einstellungen und Konfigurationen vorgenommen werden können.

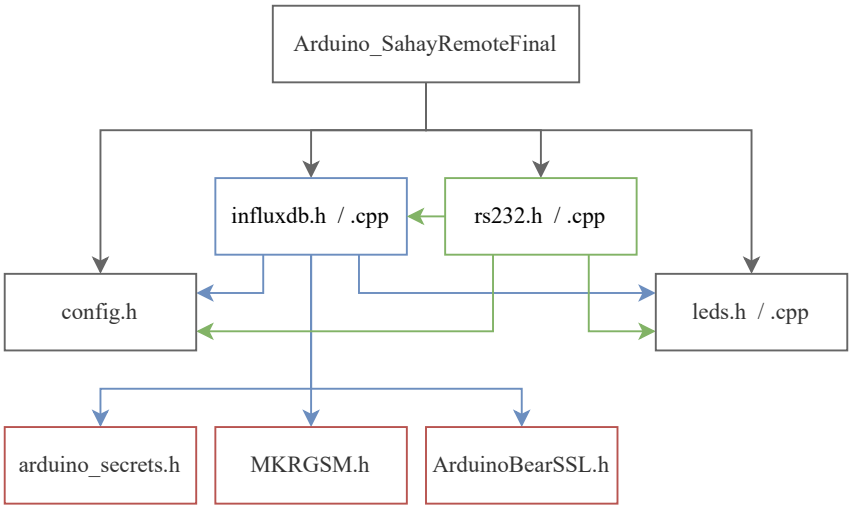


Abbildung 15: Übersicht der verschiedenen Software-Module

3.2.1 Konfiguration

Die Datei config.h definiert verschiedene Makros, mit denen das Sendesystem konfiguriert wird. Mit dem Makro *SERIAL_PRINTF_ON* kann eingestellt werden, ob das Arduino per USB-Verbindung Debug-Informationen ausgeben soll. Damit erhält man über eine serielle Schnittstelle zusätzliche Informationen, was beispielsweise bei der Fehlersuche hilfreich sein kann. Die beiden Angaben *LOCATION* und *INVERTER_TYPE* beinhalten Informationen zur Solaranlage, bei der das Sendesystem eingesetzt wird. Diese Angaben werden auch in die Datenbank übertragen, wie sich in den nächsten Kapiteln zeigen wird. Das Makro *NUMBER_OF_DATAPOINTS* bestimmt, wie viele Messwerte gesammelt werden, bis sie gemeinsam in die Datenbank übertragen werden. In diesem Fall sind es zehn Messwerte. Zudem kann mit *INTERVAL_MEASURE* die Zeit des Messintervalls in Sekunden eingestellt werden. Sie beträgt hier eine Minute. Dadurch wird alle zehn Minuten ein Paket von zehn Messungen in die Datenbank gesendet.

```
#define SERIAL_PRINTF_ON      0 // for serial debugging, set to 0 for deployment
#define LOCATION              (String)"Kriens" // location of the solar system
#define INVERTER_TYPE         (String)"Steca" // inverter type of the solar system

#define INTERVAL_MEASURE      60 // in seconds
#define NUMBER_OF_DATAPOINTS 10 // number of datapoints collected and sent togheter
```

3.2.2 RS-232

Das Abfragen der Werte via RS-232 wird gemäss dem Kommunikationsprotokoll des Steca Solarix PLI implementiert. Dabei wird eine allgemeine Statusabfrage der Solarix PLI Parameter ausgeführt. Dazu wird folgende Nachricht vom Arduino per RS-232 versendet:

```
QPIGS<CRC><cr>
```

Der Steca Solarix PLI antwortet darauf mit einer Nachricht im untenstehenden Format. Dabei stellt die Klammer an erster Stelle das Start-Byte dar. Anschliessend folgen verschiedene Messgrössen, welche hier durch Grossbuchstaben dargestellt werden. Die übrigen Daten sind einzelne Status-Bits und stellen Informationen zum Gerätestatus des Steca Solarix PLI dar [5].

```
(BBB.B CC.C DDD.D EE.E FFFF GGGG HHH III JJ.JJ KKK OOO TTTT EEEE UUU.U WW.WW PPPP
    b7b6b5b4b3b2b1b0 QQ VV MMMMM b10b9b8<CRC><cr>
```

Die Messwerte, die in der Tabelle 1 aufgelistet sind, werden vom Arduino aufgezeichnet und in die Datenbank übertragen. Die restlichen Grössen werden ignoriert, da sie für die Überwachung der Solaranlage nicht relevant sind. Die Auswahl, welche Grössen aufgezeichnet und welche verworfen werden, wurde gemeinsam mit dem Auftraggeber getroffen.

Wert	Beschreibung	Einheit
DDD.D	AC-Ausgangsspannung	V
EE.E	AC-Ausgangsfrequenz	Hz
GGGG	AC-Wirkleistung	W
JJ.JJ	Batteriespannung vom Wechselrichter gemessen	V
KKK	Batterie-Ladestrom	A
OOO	Batterie-Kapazität (ca.)	%
TTTT	Kühlkörper-Temperatur	°C
EEEE	PV-Eingangsstrom	A
UUU.U	PV-Eingangsspannung	V
WW.WW	Batteriespannung vom Laderegler gemessen	V
PPPP	Batterie-Entladestrom	A
MMMMM	PV-Ladeleistung	W

Tabelle 1: Messgrössen des Steca Solarix PLI, die ausgelesen werden [5]

In der Datei rs232.h sind zwei Funktionen als Schnittstelle zu den anderen Software-Modulen definiert. Die erste initialisiert die serielle Schnittstelle an den Pins RX und TX bzw. Pins D13 und D14. Es wird eine Baudrate von 2400/s eingestellt, da dies durch das Kommunikationsprotokoll der Steca Solrix PLI vorgegeben ist [5]. Die zweite Funktion führt die Statusabfrage durch, liest und speichert die Werte aus Tabelle 1. Um die Werte zu speichern, wird die Adresse von einem Element des Typs *steca_data_t* übergeben, an der die Daten abgelegt werden. Der Rückgabewert der Funktion ist *true*, falls die Werte korrekt eingelesen werden konnten. Ansonsten ist er *false*.

```
void RS232_Init(void);
bool RS232_GetData(steca_data_t* packet_received);
```

Die Struktur *steca_data_t*, in der die Daten gespeichert werden, beinhaltet für alle Werte aus Tabelle 1 eine Variable, was die untenstehende Definition aufzeigt. Zusätzlich besitzt sie einen Zeitstempel, in dem der Zeitpunkt der Messung abgespeichert wird. Diesen erhält man durch den Aufruf der Funktion *getTime()*. Da diese Funktion GSM-Objekte verwendet, befindet sie sich im Modul rs232.h. Die Definition der Struktur *steca_data_t* befindet sich in der Datei config.h, da diese Struktur in mehreren Modulen der Software verwendet wird.

```
struct steca_data_t
{
   float          grid_voltage = 0;
   float          grid_frequency = 0;
   uint16_t       battery_capacity = 0;
   float          battery_voltage_inv = 0;
   float          battery_voltage_chc = 0;
   int16_t        battery_chargingcurrent = 0;
   int16_t        battery_dischargingcurrent = 0;
   int16_t        pv_chargingpower = 0;
   float          pv_voltage = 0;
   float          pv_current = 0;
   int16_t        ac_activepower = 0;
   int16_t        heatsink_temp = 0;
   unsigned long  timestamp = 0;
};
```

3.2.3 GSM

Um das Arduino mit dem Internet zu verbinden, werden einige Makros aus der Datei arduino_secrets.h verwendet. Zudem werden Objekte und Funktionen aus der Library MKRGSM.h benutzt. Mit folgendem Code verbindet sich das Arduino mithilfe der SIM-Karte mit dem GSM-Netz. Dieser Abschnitt ist blockierend, was bedeutet, dass so lange gewartet wird, bis die Verbindung hergestellt werden kann.

```
bool connected = false;
while (!connected)
{
  if ((gsmAccess.begin(PINNUMBER) == GSM_READY) && (gprs.attachGPRS(GPRS_APN,
      GPRS_LOGIN, GPRS_PASSWORD) == GPRS_READY))
  {
    connected = true;
  }
  else
  {
    delay(1000);
  }
}
```

3.2.4 HTTP Post Request

Um Daten in die InfluxDB zu schreiben, gibt es verschiedene Möglichkeiten. Eine einfache und mit dem Arduino realisierbare Lösung ist ein HTTP Post Request. Dabei stellt HTTP (*Hypertext Transfer Protocol*) das Protokoll dar und ein Post Request ist eine Anfrage oder Übermittlung von Daten an einen Server [15]. Der Post Request wird von der Funktion *Influxdb_Write(...)* ausgeführt und versendet folgende Nachricht:

```
POST /api/v2/write?bucket=SahayBucket&org=sahaysolar&precision=s HTTP/1.1
Host: eu-central-1-1.aws.cloud2.influxdata.com
Authorization: Token YOUR_API_TOKEN
Content-Length: 153

Steca,location=Horw gr_v=230.20,gr_f=49.90,bt_c=59,bt_vi=47.56,bt_vc=47.46,bt_ci
    =16,bt_di=48,pv_p=459,pv_v=19.50,pv_i=45.00,ac_p=1234,hs_t=27 1652774998
```

Diese Informationen sind im Post Request erkennbar:

- Es handelt sich um eine Post-Anfrage.
- Host bzw. Name des Webservers, an den die Anfrage geschickt wird.
- Speicherort bzw. Verzeichnis des Webservers, in welches die Daten geschrieben werden.
- Ein *Token*, um den Sender zu autorisieren. Besteht aus einer Zeichenkette, ist hier allerdings ersetzt durch *YOUR_API_TOKEN* (Details im Kapitel InfluxDB).
- Wie viele Zeichen bzw. Bytes im Inhalt übermittelt werden.
- Eigentlicher Inhalt, der die Daten enthält. Es wird ersichtlich, dass die Messung einen Steca-Inverter betrifft und an welchem Standort sich die Solaranlage befindet. Anschliessend folgen die verschiedenen Variablen bzw. Messgrössen und zum Schluss der Zeitstempel.

Der Zeitstempel der Messung wird im Format des *Unix Time Stamp* übergeben. Dies ist eine Zeitdefinition, welche die Anzahl Sekunden seit dem 01.01.1970 angibt [16]. Die Schnittstellendefinition in der Datei influxdb.h definiert die Funktion *getTime()*, welche den aktuellen *Unix Time Stamp* zurückgibt. Über das verbundene GSM-Netz kann das Arduino regelmässig die Zeit abfragen und synchronisieren. Zudem sind in der Datei influxdb.h zwei weitere Funktionen definiert. Eine zur Initialisierung des Moduls, welche die GSM-Verbindung herstellt, die andere schreibt die Daten mittels obigem HTTP Post Request in die Datenbank. Sie verpackt mehrere Messungen in einen Post Request. Deshalb übergibt man ihr neben den Messdaten auch die Anzahl der Messungen, die zu übertragen sind. Um die Messdaten der Funktion zu übergeben wird wiederum die Struktur *steca_data_t* verwendet. Der Rückgabewert ist *true*, wenn die Übermittlung an den Webserver erfolgreich war. Dies kann die Funktion anhand des HTTP-Statuscodes in der Antwort des Webservers erkennen.

```
void Influxdb_Init(void);
bool Influxdb_Write(steca_data_t data_packet[], uint16_t numberOfDatapoints);
unsigned long getTime();
```

Da der Webserver der InfluxDB die Daten am Port 443 erwartet, ist es zwingend nötig, dass die Daten über SSL/TLS (*Secure Sockets Layer / Transport Layer Security*) geschützt werden. Port 443 ist der Standard-Port für HTTPS-Nachrichten (*Hypertext Transfer Protocol Secure*), was bedeutet, dass er die Nachricht nicht im Plaintext-Format akzeptiert. Sie muss somit via SSL/TLS verschlüsselt und zertifiziert werden. Dazu wird die Library ArduinoBearSSL.h verwendet.

3.2.5 LEDs

Um die LEDs anzusteuern, stehen im Modul leds.h die verschiedene Funktionen zur Verfügung. Vor dem Ein- oder Ausschalten einer LED sollte die Initialisierungsfunktion aufgerufen werden, welche die entsprechenden Pins als Ausgänge definiert und die LEDs standardmässig ausschaltet. Eine grüne LED zeigt den Status der RS-232-Verbindung an. Sie schaltet aus, sobald eine Anfrage verschickt wird und schaltet erst wieder ein, wenn die Antwort erfolgreich gelesen werden kann. Analog dazu zeigt die zweite grüne LED die Verbindung zum GSM-Netz und zur InfluxDB an. Ist die Verbindung ins GSM-Netz erfolgt, leuchtet sie auf. Beginnt der HTTP Post Request, schaltet sie aus und schaltet erst wieder ein, sobald sie eine positive Antwort des Webservers erhält. Beide grünen LEDs sollten im Normalbetrieb daher grösstenteils eingeschaltet bleiben und jeweils nur für einen kurzen Moment abschalten. Die Dauer des Abschaltens bei der RS-232-LED sollte dabei höchstens im Bereich einer Sekunde sein, bei der anderen kann es durchaus um die 10 Sekunden dauern, bis die Daten übers Netz übertragen sind und die Antwort des Webservers angekommen ist.

```
typedef enum {led_rs = 15, led_gsm = 16} led_e;

void Leds_Init(void);
void Led_On(led_e led);
void Led_Off(led_e led);
void Led_Set(led_e led, bool state);
```

3.2.6 Setup und Loop

Durch die Funktionen der verschiedenen Module und den Ablauf im Flussdiagramm in Abbildung 14 ergibt sich der Code für das Hauptprogramm, welches aus den beiden Funktionen *setup()* und *loop()* besteht. Die Funktion *setup()* wird zu Beginn einmal ausgeführt, anschliessend wird die Funktion *loop()* in einer Endlosschleife ausgeführt.

```
void setup() {
  pinMode(PIN_ENABLE_3V3, OUTPUT);
  digitalWrite(PIN_ENABLE_3V3, LOW);

  Leds_Init();
  RS232_Init();
  Influxdb_Init();

  previousMillis = millis();
}

void loop() {
  for(uint8_t i = 0; i < NUMBER_OF_DATAPOINTS; i++)
  {
    while(RS232_GetData(&data_packet[i]) == false);

    if(i == (NUMBER_OF_DATAPOINTS-1))
    {
      while(Influxdb_Write(data_packet, NUMBER_OF_DATAPOINTS) == false);
    }

    currentMillis = millis();
    while((currentMillis - previousMillis) < interval)
    {
      currentMillis = millis();
    }
    previousMillis = currentMillis;
  }
}
```

3.3 Empfangssystem

Die Datenbank InfluxDB und die Darstellungs-Anwendung Grafana bilden gemeinsam das Empfangssystem zur Überwachung der Solaranlagen. Beide Tools werden in der Cloud betrieben und nicht auf einer separaten Hardware implementiert bzw. installiert.

3.3.1 InfluxDB

Um die Messreihen zu speichern, wird die Version InfluxDB Cloud 2.0 verwendet. Die Datenbank ist durch folgende Elemente strukturiert [17]:

- Eine *Organization* stellt einen Arbeitsbereich für mehrere User dar. Alle *Buckets* und *Users* gehören einer *Organization* an.
- Ein *Bucket* kann als Ordner auf oberster Ebene angesehen werden. Alle Daten werden in einem *Bucket* gespeichert.
- Die Kolumne *Measurement* beinhaltet den Namen der Messung als String. Ein *Measurement* bildet eine Art Container für *Tags*, *Fields* und *Timestamps*.
- *Tags* stellen eine weitere Kolumne dar, um Daten zu sortieren und einzuteilen. Es können selber *Tags erstellt* und weitere hinzugefügt werden. Es wird unterschieden zwischen *Tag key* und *Tag value*.
- Parallel dazu besteht ein *Field* aus einem *Field key* und einem *Field value*. Der *Field key* beinhaltet den Namen der entsprechenden Grösse, *Field value* ist der entsprechende Wert.
- Alle gespeicherten Datenpunkte besitzen zudem einen *Timestamp*.

Abbildung 16 zeigt, wie die Datenbank zur Überwachung der Solaranlagen strukturiert ist. Dazu gibt es ein *Bucket* namens SahayBucket, welches alle Daten zur Überwachung der Solaranlagen beinhaltet. Das *Measurement* gibt an, um welchen Inverter-Typ es sich handelt. Des Weiteren gibt es den *Tag key* namens location mit den *Tag values* Horw, Luzern und Kriens in diesem Beispiel. Mit dem *Tag* werden die Daten somit nach Standort sortiert. Ganz rechts in Abbildung 16 folgen die verschiedenen *Field keys*, welche den verschiedenen Messgrössen entsprechen, so steht beispielsweise die Bezeichnung ac_p für die Wechselstrom-Wirkleistung. Dahinter befindet sich die Messreihe aus einzelnen Werten bzw. *Field values*, welche jeweils mit einem *Timestamp* gekoppelt sind.

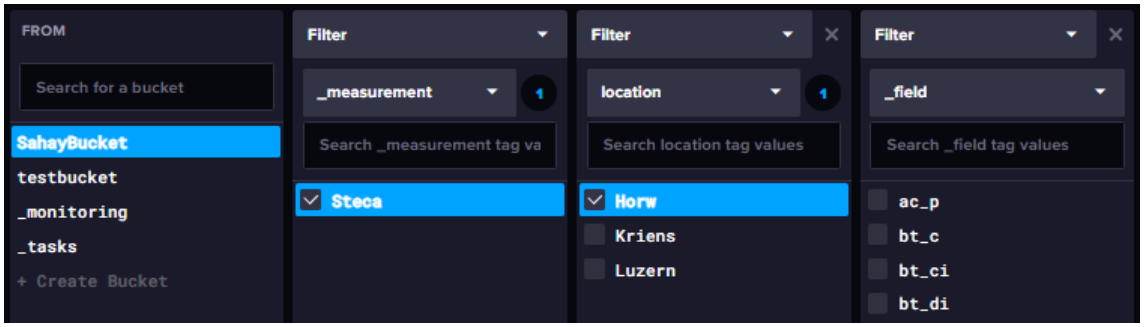


Abbildung 16: Sortierung der Daten in der InfluxDB

Zusätzlich können in der Datenbank *Tokens* erstellt werden. Ein *Token* ist eine Zeichenkette, welche benötigt wird, um Daten in ein *Bucket* zu schreiben oder daraus zu lesen. So muss beispielsweise das *Token* beim HTTP Post Request angegeben werden. Das *Token* kann mit einem Passwort verglichen werden, mit dem man Schreib- und Lesezugriff auf die Datenbank erhält.

3.3.2 Grafana

Um die Daten aus der Datenbank in Grafana zu importieren, muss in den Einstellungen der Grafana Cloud die Datenquelle InfluxDB hinzugefügt werden. Hierbei sind Angaben wie die URL der Datenbank, die *Organization* und das *Token* notwendig. Die eingerichtete Datenquelle kann in einem *Panel*, ein Visualisierungs-Block in Grafana, ausgwählt werden. Darin können mit Flux, einer von InfluxDB eigenen Skript- und Abfragesprache für Zeitreihen-Datenbanken, die Daten eingelesen werden [18]. So zeigt der untenstehende Code, wie aus dem *Bucket* namens SahayBucket und dem *Measurement* names Steca die AC-Wirkleistung mit dem Variablennamen ac_p eingelesen wird. Zudem muss definiert sein, welcher Standort angezeigt wird. Dies geschieht durch das Einlesen einer Variable in der letzten Zeile des folgenden Codes.

```
from(bucket: "SahayBucket")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "Steca")
  |> filter(fn: (r) => r["_field"] == "ac_p")
  |> filter(fn: (r) => r["location"] == ${Location:doublequote})
```

Diese Variable wird innerhalb des Grafana Dashboards erstellt und liest die verschiedenen Werte des *Tag keys* namens location, also die verschiedenen Standorte, von der InfluxDB aus. Dies wird ebenfalls in Flux realisiert. Zuoberst im Dashboard kann zwischen den verschiedenen Standorten, die in der Datenbank eingetragen sind, ausgewählt werden, wie in Abbildung 17 zu sehen ist. Die verschiedenen *Panels*, die die Zeitreihen darstellen, greifen gemäss obigem Flux-Code auf die Variable zu und zeigen demnach die Messdaten des ausgewählten Standortes an. Dadurch muss beim Einsatz eines neuen Gerätes einzig in der Arduino-Software der entsprechende Standort in der Datei config.h eingetragen werden. Sobald das Arduino Daten in die InfluxDB geschrieben hat, erscheint die entsprechende Ortschaft automatisch im Grafana Dashboard und kann ausgewählt bzw. angezeigt werden.

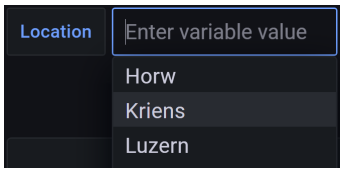


Abbildung 17: Auswahl des Standortes in Grafana

Das Dashboard ist unterteilt in die Abschnitte Batterie, Photovoltaik und Wechselstromgrössen. Diese Abschnitte werden in Grafana *Rows* genannt. Abbildung 18 zeigt den Abschnitt mit den verschiedenen Batteriespannungen und -strömen. Das *Panel* links zeigt den Verlauf der Batteriespannung, dabei wird die vom Wechselrichter und die vom Laderegler gemessene Spannung angezeigt. Es wird unterteilt in einen roten und grünen Bereich, damit der Benutzer einfach erkennen kann, ob die Batteriespannung den kritischen Bereich erreicht hat. Die Grenze zwischen dem grünen und roten Bereich beträgt 43,2 V, was einer Zellenspannung von 1,8 V entspricht und gemäss Datenblatt nicht unterschritten werden sollte [19]. Rechts davon ist der Lade- und Entladestrom sowie die Kapazität der Batterie in Prozent ersichtlich.

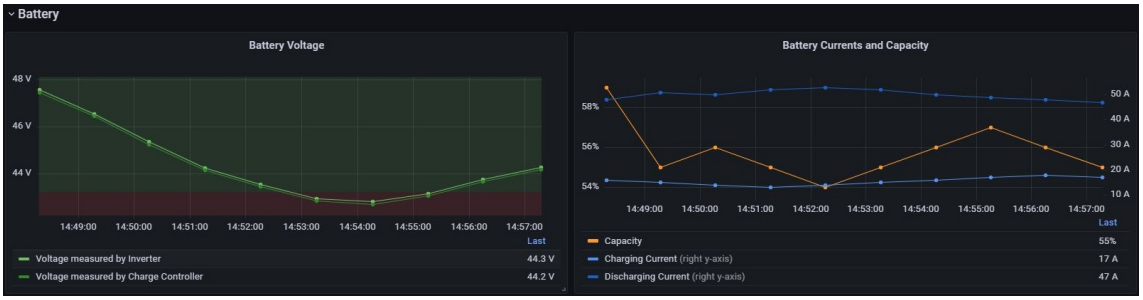


Abbildung 18: Anzeige der Batterie-Grössen in Grafana

Im Abschnitt der Photovoltaik-Grössen gibt es ein *Panel* für die PV-Eingangsspannung und den PV-Eingangsstrom. Das zweite *Panel* zeigt den Verlauf der PV-Ladeleistung. Ganz rechts in Abbildung 19 ist zusätzlich die produzierte Energie ersichtlich. Diese wird durch die Integration der Leistungskurve direkt in Grafana berechnet. Dazu kann eine Funktion der Sprache Flux benutzt werden. Es wird immer das Intergral des aktuell angezeigten Zeitbereichs berechnet. Der Zeitbereich, welcher angezeigt werden soll, kann im Dashboard vom Benutzer angepasst werden.

Abbildung 19: Anzeige der PV-Grössen in Grafana

Abbildung 20 zeigt den dritten und letzten Abschnitt des Grafana Dashboards. Auf der linken Seite wird oben die AC-Wirkleistung angezeigt. Darunter ist die AC-Ausgangsspannung gemeinsam mit der Ausgangsfrequenz zu sehen. Damit kann der Verbrauch des Inselnetzes beobachtet werden. Zudem wird rechts davon die Temperatur des Kühlkörpers dargestellt.

Abbildung 20: Anzeige der Wechselstrom-Grössen in Grafana

3.4 Kosten

In der untenstehenden Tabelle sind alle Bauteile aufgeführt, welche zur Produktion der Hardware verwendet werden. Insgesamt ergibt sich für das Sendesystem ein Gesamtpreis von 133,78 CHF. Hier sind jeweils die Einzelpreise aufgeführt. Werden gewisse Mengen von den Bauteilen bestellt, kann der Gesamtpreis tiefer ausfallen. Das mit Abstand teuerste Bauteil ist das Arduino MKR GSM 1400 mit Antenne. Es macht etwa die Hälfte der Gesamtkosten aus.

Mainboard		Anz.	Stückpreis
C1	47 µF	1	0,68 CHF
C2, C5	100 nF, SMD 0603	2	0,18 CHF
C3, C4	1 µF, SMD 0603	2	0,29 CHF
D1	CUHS20S30	1	0,40 CHF
J1, J2	DC-Versorgungssteckverbinder	1	4,09 CHF
J3, J4	Buchsenleisten 1×10 2.54mm	2	0,90 CHF
LED1, LED2	green	2	1,87 CHF
LED3	red	1	1,87 CHF
R1, R2	680 Ω, SMD 0603	2	0,10 CHF
R3	1,5 kΩ, SMD 0603	1	0,10 CHF
R4	100 kΩ, SMD 0603	1	0,10 CHF
U1	Arduino MKR GSM 1400 + GSM-Antenne	1	67,10 CHF
	Buchsenleisten 1×14 2.54mm	2	1,50 CHF
U2	TLS850B0TBV33	1	1,93 CHF
SIM	Hologram Global IoT SIM-Card	1	4,80 CHF
Netzteil	SWH15-5B-EB-P5	1	16,56 CHF
RS-232-Board			
C1, C2, C8	100 nF, SMD 0603	3	0,18 CHF
C3, C4, C5, C6, C7	1 µF, SMD 0603	5	0,29 CHF
J1, J2	Stiftleisten 1×10 2.54mm	2	0,56 CHF
J3	DB9 Female Connector	1	4,29 CHF
R1, R2	10 kΩ, SMD 0603	2	0,10 CHF
U1	MAX232	1	4,70 CHF
Sonstiges			
Gehäuse	Hammond 1591XXDSFLBK	1	7,09 CHF
Linsenschraube	M2.5×6 BN330	8	0,02 CHF
Blechschraube	BN 14064 2,9×6,5MM	3	0,03 CHF
Abstandsbolzen	M2.5 L=10mm	4	0,29 CHF
PCBs	125mm×64.5mm, 53mm×32mm	2	5,70 CHF
	Gesamtpreis		133,78 CHF

Tabelle 2: Preisliste der Hardware

4 Inbetriebnahme

Dieser Teil der Arbeit beschreibt die verschiedenen Tests, die durchgeführt werden, um das Fernwartesystem in Betrieb zu nehmen und seine Funktionen zu überprüfen. Zudem befindet sich im Anhang ein User Manual, welches zeigt, wie das System zum jetzigen Zeitpunkt verwendet werden kann und was es dabei zu beachten gilt.

Für die verschiedenen Tests wird in der Software des Arduinos das Makro *SERIAL_PRINTF_ON* in der Datei config.h auf 1 gesetzt, damit die Debug-Informationen über den USB-Port des Arduinos angeschaut werden können. Dieses Makro muss allerdings im Normalbetrieb zwingend auf 0 gesetzt werden, da sonst die Software blockiert, solange kein COM-Port am USB-Anschluss des Arduinos geöffnet ist.

4.1 Speisung

Bei der Speisung wird getestet, ob der Spannungsregler mit dem Arduino ein- sowie ausgeschaltet werden kann. Dabei stellt sich heraus, dass in der Version V1.0 des Mainboards der Enable-Eingang des Spannungsreglers fälschlicherweise auf den Pin AREF/PA03 des Arduinos verbunden ist. Dieser Pin kann nicht als digitaler Ausgang benutzt werden und somit ist es nicht möglich, den Regler ein- und auszuschalten. Dieses Problem wird behoben, indem der Enable-Eingang mit dem Pin PB03/D17 des Arduinos verbunden wird. In der Version V1.1 der Leiterplatte ist dies angepasst und der Fehler behoben.

4.2 RS-232-Verbindung

Da zum Testen der RS-232-Verbindung kein Steca Solrix PLI zur Verfügung steht, wird die RS-232-Schnittstelle mit einem Adapter getestet. Verwendet wird der *UC232A* - Konverter USB zu seriell RS-232 des Herstellers Aten. Dadurch kann der Steca Solarix PLI mit einer normalen COM-Schnittstelle auf einem Windows-Rechner simuliert werden. Dazu kann die Software *PuTTY* oder der *Serial Monitor* innerhalb der Arduino IDE benutzt werden. Die Baudrate beträgt 2400/s.

Abbildung 21: *UC232A* - Konverter USB zu seriell RS-232 [20]

Beim Test der Verbindung zeigt sich, dass in der Version V1.0 die Signale RX und TX auf Seite des Arduinos vertauscht sind. Dies sind die beiden Kommunikationssignale der UART zwischen dem Arduino und dem *MAX232*. Dadurch funktioniert die Kommunikation nicht. Wird dieser Fehler behoben, können Daten hin- und hergeschickt werden. In der Version V1.1 der Hardware ist dieser Fehler behoben.

Zum Testen der RS-232-Verbindung werden zwei *Serial Monitors* mit der Arduino IDE geöffnet. Einer verwendet den COM-Port, an dem das Arduino per USB angeschlossen ist, mit einer Baudrate von 9600/s. Dieser ist in Abbildung 22 dargestellt. Der zweite wird in Abbildung 23 gezeigt und benutzt den COM-Port, an dem sich der Konverter *UC232A* befindet, mit einer Baudrate von 2400/s. Hier empfängt man nun die Nachricht *QPIGS*, die allgemeine Statusabfrage der Solarix PLI Parameter gemäss dessen Kommunikationsprotokoll. Anschliessend kann die Antwort, die der Steca Solrix PLI liefern würde, im Eingabefeld eingegeben werden. Beim ersten Mal werden einige Beispielwerte eingesetzt. Das Arduino kann die Nachricht interpretieren und die entsprechenden Werte einlesen, wie es bei den Debug-Informationen zeigt. Anschliessend wird eine zweite Anfrage verschickt. Diesmal wird in der Antwort absichtlich die Klammer, welche das Start Byte symbolisiert, als erstes Zeichen weggelassen. In diesem Fall kann das Arduino die Nachricht nicht einlesen, weil sie nicht dem erwarteten Format entspricht, was ebenfalls in den Debug-Informationen ersichtlich ist.

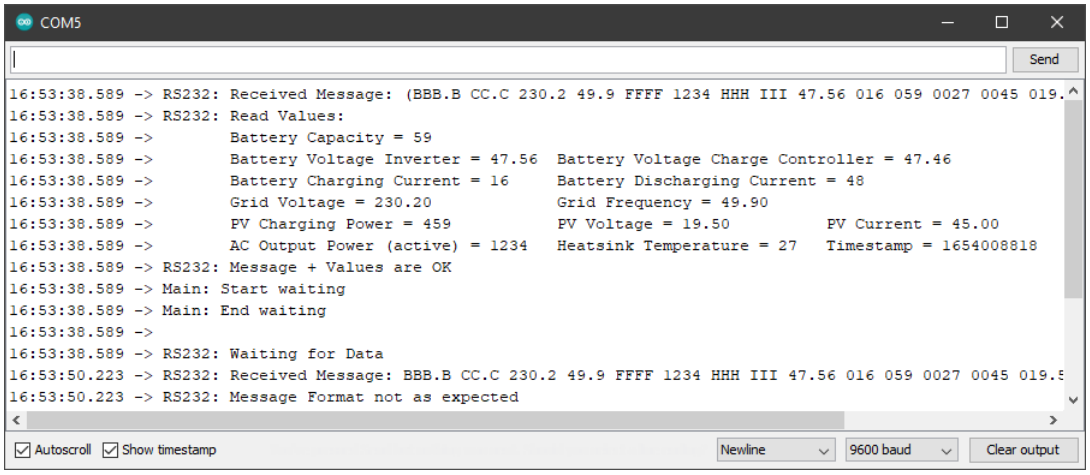


Abbildung 22: Debug-Informationen beim Test der RS-232-Verbindung

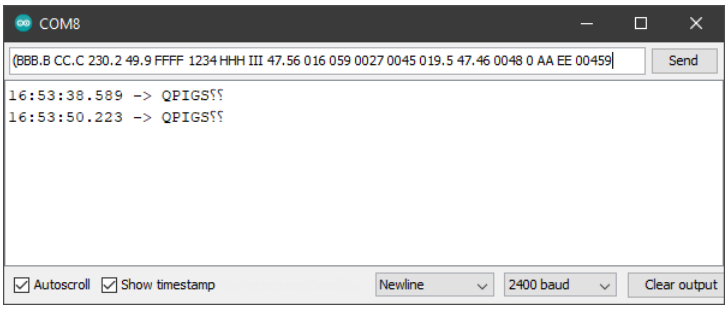


Abbildung 23: Statusabfrage des Steca Solarix PLI via *UC232A*

Zudem wird der Status der RS-232-Verbindung mittels LED2 (grün, mittlere der drei LEDs im Gehäuse) angezeigt. Sobald die *Meldung Waiting for Data* im *Serial Monitor* erscheint, schaltet sie aus. Die LED2 bleibt solange ausgeschaltet, bis eine Nachricht erfolgreich eingelesen werden kann. Wenn das Format wie im obigen Beispiel nicht stimmt, bleibt sie ausgeschaltet.

Soweit funktioniert die Kommunikation über die RS-232-Verbindung. Allerdings sollte die Kommunikation gemeinsam mit einem Steca Solarix PLI getestet werden. Damit wäre es möglich, das Gerät unter den echten Bedingungen zu überprüfen, bei denen es eingesetzt wird.

4.3 Datenübertragung in die InfluxDB

Ob das Übermitteln der Werte in die Datenbank erfolgreich gewesen ist, wird anhand des HTTP-Statuscodes in der Antwort des Webservers festgestellt. Die Antwort gibt das Arduino bei den Debug-Informationen über die USB-Schnittstelle aus. Für eine erfolgreiche Operation stehen alle Statuscodes im Format 2XX. Der Webserver der InfluxDB antwortet folgendermassen:

```
HTTP/1.1 204 No Content
Date: Tue, 17 May 2022 08:33:53 GMT
Connection: keep-alive
trace-id: 5b1f9f0d3e8ff37f
trace-sampled: false
Strict-Transport-Security: max-age=15724800; includeSubDomains
X-Influxdb-Request-ID: f7f9f630c7d7b08f499a4e3f7055807f
X-Influxdb-Build: Cloud
```

Der Statuscode 204 in der Antwort zeigt, dass die Anfrage erfolgreich war. Zudem besitzt die Antwort des Servers keinen Inhalt. In der finalen Version der Software wird die Antwort des Servers nur bis zum Statuscode gelesen, da daran bereits zu erkennen ist, ob die Übertragung erfolgreich war. Der Rest der Antwort wird verworfen, weil er für das Arduino nicht relevant ist. Dies ist im Beispiel von Abbildung 24 zu erkennen. Hier werden drei Messungen in einer Anfrage verpackt und gemeinsam übermittelt.

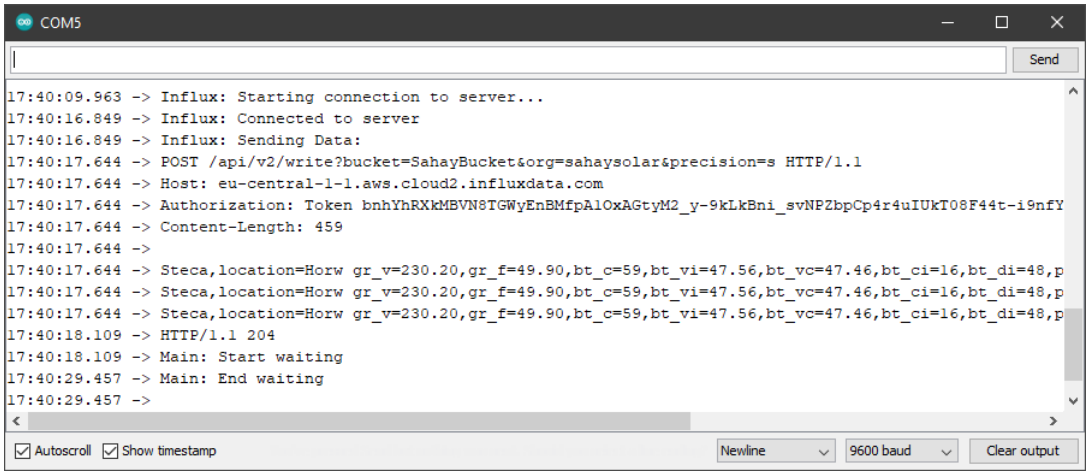


Abbildung 24: Debug-Informationen bei Datenübertragung in InfluxDB

Auch hier zeigt eine LED analog zur RS-232-Verbindung den Status der Verbindung zur Datenbank an. Sobald sich das Arduino mit dem Server zu verbinden beginnt, schaltet die LED1 (grün, rechte LED der drei LEDs im Gehäuse) aus. Zu diesem Zeitpunkt wird die Information *Starting connection to server...* per USB ausgegeben, wie Abbildung 24 zeigt. Sobald die Übermittlung der Messungen in die Datenbank abgeschlossen ist und die Antwort des Webservers eine erfolgreiche Operation bestätigt, schaltet die LED1 wieder ein. Da das Verbinden zum Server, das Senden der Werte und das Erhalten der Antwort gemeinsam etwa 10 Sekunden in Anspruch nehmen, ist die LED für diese Zeit ausgeschaltet. Beim Versuch in Abbildung 24 dauert dieser Prozess von 17:40:09.963 bis 17:40:18.109, was einer Dauer von 8,146 Sekunden entspricht.

Der Versuch zeigt, dass das Übermitteln der Messungen in die Datenbank per HTTP Post Request funktioniert. Zudem ist es möglich, mehrere Messungen gemeinsam mit einem einzelnen Post Request zu versenden.

4.4 Datenmenge

Dieser Test wird durchgeführt, um eine Aussage zum Datenverbrauch des Sendesystems zu treffen und dessen Datenverbrauch im Normalbetrieb abzuschätzen. Die folgenden Messungen und Berechnungen basieren alle auf einem Messintervall von einer Minute, was bedeutet, dass alle 60 Sekunden eine RS-232-Abfrage beim Steca Solrix PLI ausgeführt wird.

Auf dem Dashboard des SIM-Karten-Anbieters Hologram kann der Datenverbrauch einer aktivierten SIM-Karte nachverfolgt werden, was in Abbildung 25 gezeigt wird.

Recent Data Sessions Table

View JSON

START / END TIMESTAMP	LENGTH	BYTES	IMEI	NETWORK
2022-05-17 12:58:00 2022-05-17 12:58:40	00:00:40	7.59 KB	357520078560485	Salt Mobile SA 3G
2022-05-17 12:57:05 2022-05-17 12:57:56	00:00:51	7.79 KB	357520078560485	Salt Mobile SA 3G
2022-05-17 12:55:30 2022-05-17 12:57:01	00:01:31	7.82 KB	357520078560485	Salt Mobile SA 3G
2022-05-17 12:53:09 2022-05-17 12:55:26	00:02:17	8.06 KB	357520078560485	Salt Mobile SA 3G
2022-05-17 12:50:44 2022-05-17 12:53:04	00:02:20	7.89 KB	357520078560485	Salt Mobile SA 3G

Abbildung 25: Messung der Datennutzung auf Dashboard von Hologram

Im Beispiel der Abbildung 25 wurde jeweils eine Messung der Solarix PLI Parameter einzeln übertragen. Weitere Datenmessungen zeigen, dass allein das Verbinden mit dem Webserver der InfluxDB etwa ein Datenvolumen von 6,5 kB benötigt. Der HTTP Post Request braucht dementsprechend etwa 1,3 kB, wovon ungefähr 150 B die eigentlichen Daten sind. Der relativ grosse Datenverbrauch bei der Verbindungsherstellung ist darauf zurückzuführen, dass dabei für die sichere Kommunikation via SSL/TLS die Schlüssel ausgetauscht werden und eine Authentisierung stattfindet. Daher steigt der Datenverbrauch, sobald 10 Messungen in einen HTTP Post Request verpackt und gemeinsam versendet werden, nur leicht an. Dies ist in der Tabelle 3 zu erkennen. Das Hologram-Dashboard zeigt, obwohl mehrmals exakt derselbe HTTP Post Request ausgeführt wird, nicht immer die genau gleiche Datenmenge an. Dies liegt vermutlich ebenfalls am Schlüsselaustausch, an der Authentisierung bzw. am Handshake via SSL/TLS. Daher werden die beiden Mittelwerte für die folgenden Berechnungen verwendet.

Anzahl Messungen	Datenmenge pro Sendung [kB]					Mittelwert [kB]
1	7.89	8.06	7.82	7.79	7.59	7.83
10	9.47	9.78	10.16	9.51	9.77	9.74

Tabelle 3: Messungen der Datenmenge

Es wird davon ausgegangen, dass die benötigte Datenmenge pro Übertragung linear mit der Anzahl Datenpunkte, welche darin verpackt sind, ansteigt. Mit den beiden Mittelwerten aus Tabelle 3 lässt sich dadurch für jede beliebige Anzahl Datenpunkte die entsprechende Datenmenge berechnen. Dadurch ergibt sich das Diagramm in Abbildung 26. Die beiden gemessenen Punkte sind darin rot markiert. Der Y-Achsenschnittpunkt auf der linken Seite gibt die Datenmenge an, die pro Übertragung immer anfällt, auch wenn theoretisch keine Werte gesendet werden. Er beinhaltet den Datenverbrauch der Verbindung (Schlüsselaustausch, Authentisierung) und den Overhead des HTTP Post Request mit Response. Der Overhead beinhaltet den Teil der Nachricht, der keine eigentlichen Daten bzw. Werte beinhaltet.

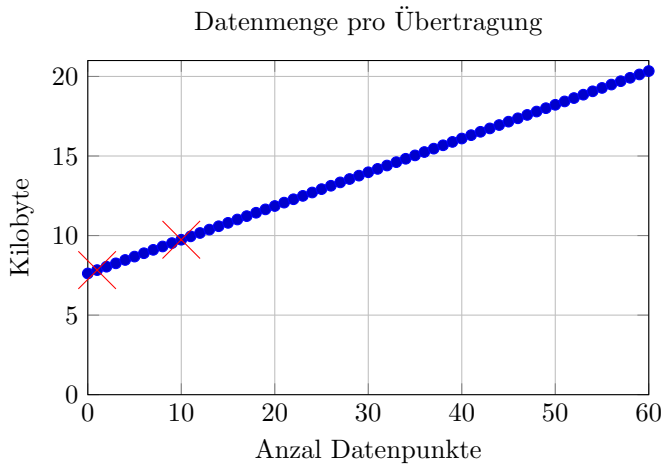


Abbildung 26: Datenmenge im Abhängigkeit der Anzahl Messpunkte pro HTTP Post Request

Durch die Werte aus Abbildung 26 kann ermittelt werden, wie gross das täglich benötigte Datenvolumen ist. Dies lässt sich mit folgender Formel berechnen und wird anhand der beiden Beispielpunkte ausgerechnet.

$$D_{tägl.} = \frac{24 * 60}{t_{Sendezyklus}} \cdot D_{proSendung}$$

$$D_{t=1min} = \frac{24 * 60}{1} \cdot 7{,}83\,\text{kB} = 11\,275\,\text{kB} = 11{,}01\,\text{MB}$$

$$D_{t=10min} = \frac{24 * 60}{10} \cdot 9{,}74\,\text{kB} = 1403\,\text{kB} = 1{,}37\,\text{MB}$$

Wird diese Rechnung für jeden Punkt ausgeführt, ergibt sich eine Hyperbel, welche im Diagramm in Abbildung 27 erkennbar ist. Es zeigt, dass je mehr Messungen gesammelt und gemeinsam übermittelt werden, desto kleiner wird der tägliche Datenverbrauch. Dies, weil durch das gemeinsame Senden mehrerer Messungen das Datenvolumen effizienter ausgenutzt wird und weniger Daten für den Schlüsselaustausch, Authentisierung und Overhead verbraucht werden. Mit dem aktuellen Sendezyklus von 10 Minuten resultiert ein täglicher Datenverkehr von 1,369 MB, was pro Monat 41,07 MB ergibt.

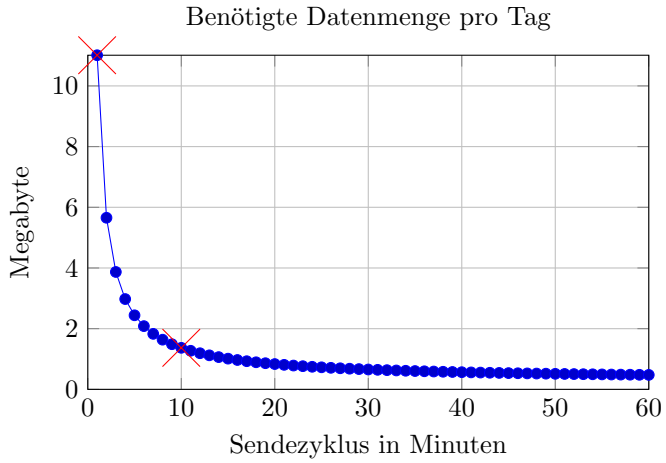


Abbildung 27: Tägliche Datenmenge in Abhängigkeit der Dauer des Sendezyklus

4.5 Anzeige der Energie

Es soll überprüft werden, ob die produzierte PV-Energie, die in Grafana angezeigt wird, korrekt berechnet wird. Dazu werden mehrere Leistungswerte manuell (z.B. mit dem Kommandozeilen-Programm cURL) in die Datenbank geschrieben. Die Zeitstempel werden so definiert, dass genau zu jeder Stunde ein Wert vorhanden ist, was Tabelle 4 zeigt.

Zeitpunkt	04:00	05:00	06:00	07:00	08:00	09:00	10:00
Leistung	0 W	6 W	21 W	29 W	31 W	35 W	49 W
	11:00	12:00	13:00	14:00	15:00	16:00	17:00
	75 W	79 W	82 W	65 W	24 W	12 W	5 W

Tabelle 4: Leistungswerte der Kurve in Abbildung 28

Die produzierte Energie berechnet sich durch Integrieren der Leistungskurve. Da hier allerdings eine diskrete Kurve vorliegt, wird das Integral bzw. die Fläche unterhalb der Kurve mit untenstehender Formel berechnet. Werden die Leistungswerte aus Tabelle 4 sowie für t jeweils 1 h eingesetzt, erhält man dasselbe Resultat wie die Integration durch Grafana. Damit ist die Berechnung korrekt.

$$E = \sum_{i=1}^{n} P_i \cdot t = 513\,\mathrm{Wh}$$

Abbildung 28: Kurve der PV-Leistung und berechnete PV-Energie in Grafana

4.6 Flash-Speicher

Gemäss dem Test bezüglich Datenmenge macht es Sinn, mehrere Messungen zu sammeln und gemeinsam zu versenden. Dafür ist der interne Flash-Speicher des Arduinos vorgesehen. Darauf sind die Daten vor einem Ausfall der Speisung geschützt und gehen nicht verloren, da es sich beim Flash um einen nichtflüchtigen Speicher handelt.

Allerdings sollte dieser Flash-Speicher nicht zu oft überschrieben werden. Daher gibt der Hersteller ein Obergrenze für die Anzahl Schreibzyklen an, welche nicht überschritten werden sollte. Typischerweise wird ein Limit von 10 000 Schreibzyklen angegeben [21]. Da in dieser Anwendung der nichtflüchtige Speicher jede Minute beschrieben werden soll, würde man damit den Speicher bereits in 10 000 Minuten bzw. 167 Stunden zerstören. Daher wird in der aktuellen Version der Software das Speichern auf dem Flash weggelassen. Die Daten werden als gewöhnliche Variablen gespeichert, wobei sie auf dem RAM, dem Arbeitsspeicher, abgelegt sind. Dadurch gehen sie verloren, sobald die Speisung des Gerätes ausfällt. Weil auf dem RAM weniger Speicherplatz zur Verfügung steht, können zurzeit nur eine begrenzte Anzahl an Werten zwischengespeichert werden. Bei Sammeln von 10 Werten zeigen die Versuche, dass der Speicher des RAMs ausreicht und kein Speicherüberlauf stattfindet.

Um dieses Problem zu lösen empfiehlt es sich, in einer Überarbeitung der Hardware einen separaten Speicherbaustein dem Mainboard hinzuzufügen. Hier könnte beispielsweise ein EEPROM mit serieller Schnittstelle wie I^2C oder SPI verwendet werden. EEPROMs besitzen tendenziell eine höhere Anzahl Schreibzyklen als Flash-Speicher. Der grosse Vorteil eines EEPROM-Speichers wäre in diesem Fall allerdings, dass im Gegensatz zum Flash einzelne Bytes überschrieben werden können. Flash-Speicher werden immer blockweise überschrieben. So könnten die Daten im EEPROM immer wieder an verschiedene Adressen gespeichert werden, damit der Speicherbaustein länger überlebt. Diese Anpassung benötigt allerdings neben zusätzlicher Hardware auch neue Software-Elemente. Das Verwalten des EEPROM-Speichers müsste zu der Arduino-Software hinzugefügt werden.

4.7 Komplettes System

Um das Sende- und Empfangssystem gemeinsam zu testen, wird eine Applikation in C# verwendet, weil zum Testen kein Steca Solrix PLI zur Verfügung steht. Die C#-Applikation übernimmt dadurch den Teil des Steca Solrix PLI und empfängt über den COM-Port und den *UC232A* Konverter die Statusabfrage. Darauf antwortet sie mit der im Kommunikationsprotokoll definierten Nachricht und gibt einige vorgegebene Werte zurück. Dadurch läuft das System selbständig. Die Werte, welche in der Ausgabe der C#-Anwendung in Abbildung 29 gezeigt werden, erscheinen kurz darauf bzw. im Zehnminutentakt auf Grafana.

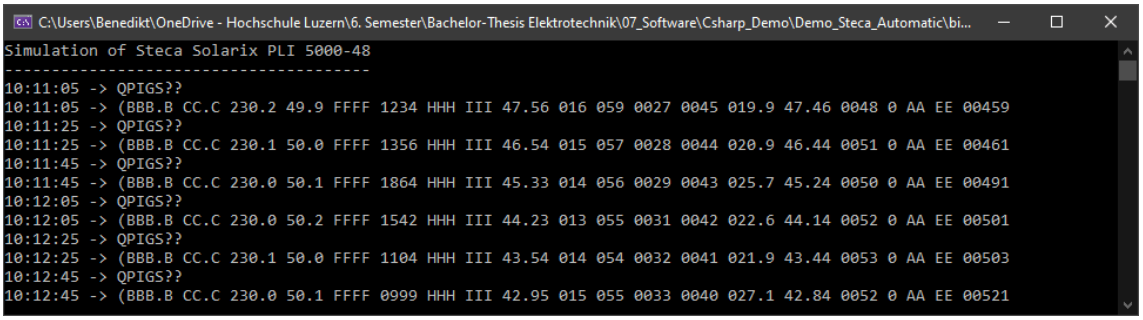


Abbildung 29: Test der Anwendung mittels C#-Applikation

Damit funktioniert der gesamte Datenfluss des Fernwartesystems: Via RS-232 werden die Daten an das Arduino gesendet, welches die Nachricht zerteilt, die benötigten Messwerte ausliest und per Mobilfunknetz in die Datenbank überträgt. Die in der InfluxDB gespeicherten Daten werden in Grafana importiert und graphisch dargestellt.

5 Schlussfolgerungen

Das Ziel dieser Arbeit ist es, ein Fernwarte- bzw. Monitoring-System für Sahay Solar zu entwerfen und in Form eines Prototypen umzusetzen. Grundsätzlich ist dieses Ziel erreicht, obwohl bei verschiedenen Punkten noch Verbesserungspotential vorhanden ist. Trotzdem kann gezeigt werden, dass die Aufzeichnung der entsprechenden Grössen und die Visualisierung im Empfangssystem in Grafana funktioniert.

Im Bezug auf die aufgelisteten Anforderungen ist zu erwähnen, dass der Prototyp einzig bei den Anlagen eingesetzt werden kann, welche einen Steca Solrix PLI verwenden. Daher ist der Prototyp nicht universell einsetzbar (Anforderung Nr. 1.2), was jedoch in den Spezifikationen angegeben ist. Im Rahmen dieses Projektes wurde ein Kommunikationsmodul, das RS-232-Board, realisiert. Das Systemdesign ermöglicht aber durch das austauschbare Kommunikationsmodul eine einfache Möglichkeit, künftig neue Schnittstellen für die weiteren Anlagen (SMA und Victron) zu implementieren. Mit der Aufteilung in ein Mainboard und ein Kommunikationsmodul konnte dazu in diesem Projekt die Grundlage geschaffen werden.

Eine Anforderung, welche deutlich nicht erfüllt ist, ist die Anforderung Nr. 2.8. Sie besagt, dass beim Ausfall der Speisung keine Daten verloren gehen. Aufgrund der beschriebenen Obergrenze von Schreibzyklen beim Flash-Speicher des Arduinos kann dieser nicht benutzt werden, um die Daten zwischenzuspeichern. Durch Hinzufügen eines externen Speicherbausteins könnte dieses Problem in einer nächsten Version behoben werden. Dieser Schritt wäre auch bezüglich der verbrauchten Datenmenge sinnvoll. Auf dem externen Speicherbaustein wäre mehr Speicherplatz vorhanden als zurzeit auf dem Arbeitsspeicher. Darum könnte, falls gewünscht, die Dauer des Sendezyklus verlängert und dadurch der Datenverbrauch gemäss den Berechnungen und Graphen in Zukunft reduziert werden.

Alle anderen Festanforderungen erfüllt der Prototyp. Die einzige Wunschanforderung (Anforderung Nr. 3.3) ist aus zeitlichen Gründen im System nicht umgesetzt. Um das Mess- und Sendeintervall im Empfangssystem einzustellen, wäre eine Anpassung der Systemarchitektur im Bereich der Internetkommunikation notwendig. Hier könnte beispielsweise das Netzwerkprotokoll MQTT eingesetzt werden, damit auch Daten an das Arduino gesendet werden können.

Bei der Umsetzung des Prototyps sind einige Fehler aufgetreten. Diese sind angepasst und in den entsprechenden Unterlagen (Version V1.1) korrigiert. Damit ist die Hardware mit den beiden Leiterplatten sowie die Software des Arduinos funktionsfähig. Die Software könnte jedoch im Bereich der Robustheit und des Stromverbrauchs optimiert werden. So ist die aktuelle Software an einigen Stellen blockierend. Das heisst, dass sie an einigen Stellen stehen bleiben kann, so wenn beispielsweise kein Netzempfang vorhanden ist. In diesem Fall wartet das Arduino an dieser Stelle, bis eine Verbindung hergestellt werden kann. Um den Stromverbrauch des Arduinos zu reduzieren, könnte es in einen Schlafmodus versetzt werden. Dies wäre an der Stelle sinnvoll, an der es abwartet, bis die nächste Messung vollzogen wird. Da der Energieverbrauch des Gerätes allerdings nicht im Fokus dieses Projektes stand, ist dies in der aktuellen Version nicht implementiert.

Zusammenfassend lässt sich sagen, dass die Entwicklung eines ersten Prototyps erfolgreich abgeschlossen ist. Der Datenfluss von der RS-232-Schnittstelle via Arduino in die InfluxDB und auf das Grafana Dashboard funktioniert. Einzig die Kommunikation via RS-232 konnte nicht abschliessend getestet werden, da im Rahmen dieser Arbeit kein Steca Solrix PLI zur Verfügung stand. Wird dieser Test nachgeholt und liefert er ein positives Resultat, kann das System theoretisch eingesetzt werden. Trotzdem wäre es sinnvoll, die genannten Verbesserungen und Optimierungen umzusetzen.

Im September 2022 soll das Fernwartesystem ein erstes Mal in Äthiopien getestet und eingesetzt werden.

Literatur

[1] United Nations Development Programme. Dashboard 1. Quality of human development. https://hdr.undp.org/data-center/documentation-and-downloads (abgerufen am 08.06.2022).

[2] Sahay Solar Verein Schweiz. Webseite Sahay Solar. http://sahay-solar.ch/ (abgerufen am 14.04.2022).

[3] Wikipedia contributors. Gamu-Gofa — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Gamu-Gofa (abgerufen am 14.04.2022).

[4] Steca Elektronik GmbH. Webinar PV Off Grid: Steca Solarix PLI. https://www.steca.com/index.php?Steca-Solarix-PLI5c10bdf0e03515c10bdf10b305#productdescription (abgerufen am 28.04.2022).

[5] Steca Elektrinik. Beispiel Kommunikationsprotokoll Steca Solarix PLI 5000-48. https://solaranzeige.de/phpBB3/download/file.php?id=79 (abgerufen am 23.05.2022), 2017.

[6] Matthijs Vader. Datenkommunikation mit Victron Energy Produkten. https://www.victronenergy.de/upload/documents/Technical-Information-Data-communication-with-Victron-Energy-products_DE.pdf (abgerufen am 20.05.2022), 2021.

[7] SMA Solar Technology. SUNNY BOY 3.0 / 3.6 / 4.0 / 5.0 inklusive SMA SMART CONNECTED. https://static.zolar.de/docs/SMA/SMA_SB_3-0-5-0_Datenblatt_de.pdf (abgerufen am 23.05.2022).

[8] nPerf SAS. MTN Mobile 3G / 4G / 5G Abdeckungskarte, Ethiopia. https://www.nperf.com/de/map/ET/-/223603.MTN-Mobile/ (abgerufen am 25.05.2022).

[9] Arduino. Arduino SIM - COVERAGE IN 100 COUNTRIES. https://store.arduino.cc/digital/sim/#coverage (abgerufen am 25.05.2022).

[10] Hologram. GLOBAL IOT COVERAGE. https://www.hologram.io/pricing/coverage#coverage-table (abgerufen am 25.05.2022).

[11] Hologram. Flexible IoT Data. https://www.hologram.io/pricing/flexible-data (abgerufen am 25.05.2022).

[12] Grafana Labs. Princing Grfana Cloud. https://grafana.com/pricing/ (abgerufen am 26.05.2022).

[13] InfluxData. InfluxDB Cloud Pricing. https://www.influxdata.com/influxdb-cloud-pricing/ (abgerufen am 26.05.2022).

[14] Arduino. Arduino MKR GSM 1400. https://store-usa.arduino.cc/products/arduino-mkr-gsm-1400 (abgerufen am 25.05.2022).

[15] InfluxData. Write data with the InfluxDB API. https://docs.influxdata.com/influxdb/cloud/write-data/developer-tools/api/#send-a-write-request (abgerufen am 30.05.2022).

[16] Dan's Tools. The Current Epoch Unix Timestamp. https://www.unixtimestamp.com/ (abgerufen am 30.05.2022).

[17] InfluxData. InfluxDB data elements. https://docs.influxdata.com/influxdb/cloud/reference/key-concepts/data-elements/ (abgerufen am 30.05.2022).

[18] InfluxData. Get started with Flux. https://docs.influxdata.com/influxdb/cloud/query-data/get-started/ (abgerufen am 30.05.2022).

[19] BAE Batterien GmbH. BAE Secura PVV BLOCK solar. https://www.levobatterien.com/images/pdf_downloads/BAE_PVV_Block_dt_2012.pdf (abgerufen am 30.05.2022).

[20] ATEN INFOTECH. UC232A1. https://www.aten.com/de/de/products/usb-periphere/usb-konverter/uc232a1/ (abgerufen am 30.05.2022).

[21] Cristian Maglie. FlashStorage library for Arduino. https://github.com/cmaglie/FlashStorage (abgerufen am 31.05.2022).

Anhang

- A Unterlagen zur Hardware
  - A.1 Stückliste
  - A.2 Schema Mainboard
  - A.3 PCB Mainboard
  - A.4 Schema RS-232-Board
  - A.5 PCB RS-232-Board
  - A.6 Plan Gehäuse
- B Weitere Technische Unterlagen
  - B.1 User Manual Remote Monitoring
  - B.2 Beispiel Kommunikationsprotokoll Steca Solarix PLI 5000-48
  - B.3 Messungen und Berechnungen Datenmenge
- C Verschiedenes
  - C.1 Zeitplanung
  - C.2 Aufgabenstellung
- D Übersicht elektronischer Anhang (Zip-Datei TechDocs)

A.1 Stückliste

Mainboard

Ref	Qnty	Value	Part	Footprint	Description	Vendor	Article Nr	Stückpreis
C1	1	47uF	Device:CP	LibraryLoader:EEE1AA101SP	Polarized capacitor	Mouser	667-EEE-HBE470UAP	CHF 0.68
C2, C5	2	100nF	Device:C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor	HSLU / Mouser	80-C0603104K3RAC7411	CHF 0.18
C3, C4	2	1uF	Device:C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor	HSLU / Mouser	80-C0603C105K3RLR	CHF 0.29
D1	1	CUHS20S30	Device:D_Schottky	LibraryLoader:CUHS20S40H3F	Schottky diode	Mouser	757-CUHS20S30H3F	CHF 0.40
H1, H2, H3	3	Blechschraube	Mechanical:MountingHole	MountingHole:MountingHole_3.2mm_M3	BN 14064 2,9X6,5MM	HSLU / Distrelec	300-71-842	CHF 0.03
H5, H6, H7, H8	4	Linsenschraube	Mechanical:MountingHole	MountingHole:MountingHole_2.7mm_M2.5	M2.5 x 6 BN330	HSLU / Distrelec	301-72-833	CHF 0.02
J1, J2	1	Conn_01x01_Female	Connector:Conn_01x01_Female	Connector_Pin:Pin_D1.3mm_L11.0mm_LooseFit	Generic connector, single row, 01x01, script generated (kicad-library-utils/schlib/autogen/connector/)	Mouser	163-4021	CHF 4.09
J3, J4	2	Buchsenleisten 1x10	Connector_Generic:Conn_01x10	Connector_PinHeader_2.54mm:PinHeader_1x10_P2.54mm_Vertical	um RS-232-Board aufzustecken	HSLU / Distrelec	143-84-000	CHF 0.90
LED1, LED2	2	green	Device:LED	LED_THT:LED_D3.0mm	Light emitting diode	Mouser	645-558-1301-001F	CHF 1.87
LED3	1	red	Device:LED	LED_THT:LED_D3.0mm	Light emitting diode	Mouser	645-558-1101-001F	CHF 1.87
R1, R2	2	680Ω	Device:R	Resistor_SMD:R_0603_1608Metric	Resistor	HSLU / Mouser	667-ERJ-3EKF6800V	CHF 0.10
R3	1	1.5kΩ	Device:R	Resistor_SMD:R_0603_1608Metric	Resistor	HSLU / Mouser	667-ERJ-3EKF1501V	CHF 0.10
R4	1	100kΩ	Device:R	Resistor_SMD:R_0603_1608Metric	Resistor	HSLU / Mouser	667-ERJ-3EKF1003V	CHF 0.10
U1	1	ArduinoMKRGSM1400	Arduino_MKR_GSM:ArduinoMKRGSM1400	ARDUINO_ABX00012	mit GSM-Antenne	Distrelec	301-57-026	CHF 67.10
-	2	Buchsenleisten 1x14	-	-	um Arduino aufzustecken	HSLU / Distrelec	143-84-000	CHF 1.50
U2	1	TLS850B0TBV33	LibraryLoader:TLS850B0TBV33	Package_TO_SOT_SMD:TO-263-5_TabPin3	Lineare Spannungsregler	Mouser	726-TLS850B0TBV33ATM	CHF 1.93
SIM-Card	1	Hologram SIM-N2-2FF3FF4FF	-	nicht direkt auf PCB	Global IoT SIM-Card	Mouser	646-SIM-N2-2FF3FF4FF	CHF 4.80
Netzteil	1	SWH15-5B-EB-P5	-	nicht direkt auf PCB	AC-Adapter zur Wandmontage ac-dc, 5 Vdc, 3 A, SW, wall-plug, EU, P5 center pos, level VI, black	Mouser	490-SWH15-5B-EB-P5	CHF 16.56

RS232-Board

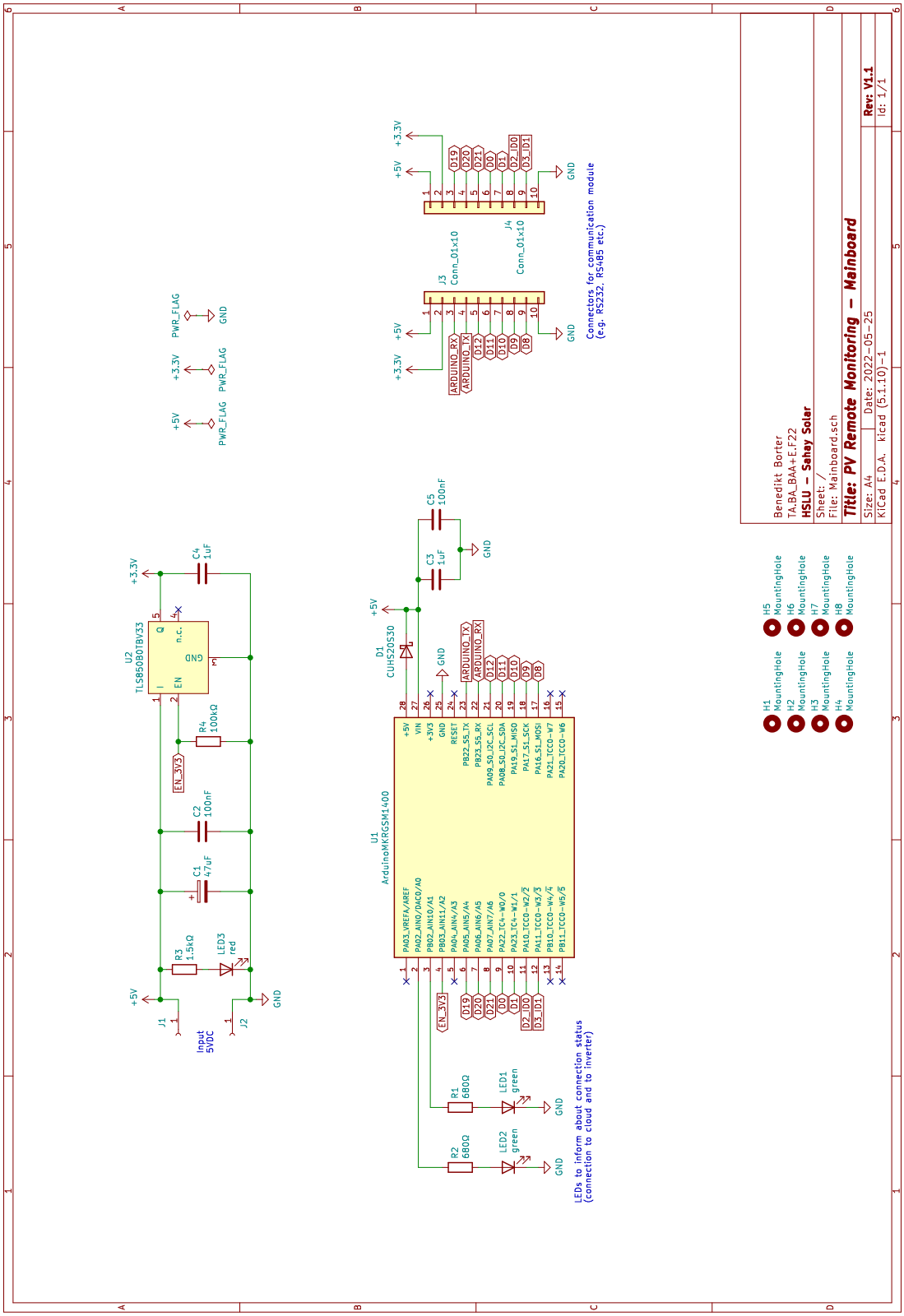
Ref	Qnty	Value	Part	Footprint	Description	Vendor	Article Nr	Stückpreis
C1, C2, C8	3	100nF	Device:C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor	HSLU / Mouser	80-C0603104K3RAC7411	CHF 0.18
C3, C4, C5, C6, C7	5	1uF	Device:C	Capacitor_SMD:C_0603_1608Metric	Unpolarized capacitor	HSLU / Mouser	80-C0603C105K3RLR	CHF 0.29
H1, H2, H3, H4	4	Linsenschraube	Mechanical:MountingHole	MountingHole:MountingHole_2.7mm_M2.5	M2.5 x 6 BN330	HSLU / Distrelec	301-72-833	CHF 0.02
J1, J2	2	Stiftleisten 1x10	Connector_Generic:Conn_01x10	Connector_PinHeader_2.54mm:PinHeader_1x10_P2.54mm_Vertical	um RS-232-Board aufzustecken	HSLU / Distrelec	143-83-920	CHF 0.56
J3	1	DB9_Female_MountingHoles	Connector:DB9_Female_MountingHoles	LibraryLoader:57478444	9-pin female D-SUB connector, Mounting Hole	Mouser	571-5747844-4	CHF 4.29
R1, R2	2	10kΩ	Device:R	Resistor_SMD:R_0603_1608Metric	Resistor	HSLU / Mouser	667-ERJ-3EKF1002V	CHF 0.10
U1	1	MAX232	Interface_UART:MAX232	Package_SO:SOIC-16_3.9x9.9mm_P1.27mm	Dual RS232 driver/receiver, 5V supply, 120kb/s, 0C-70C	Mouser	700-MAX232CSE	CHF 4.70

Mechanical

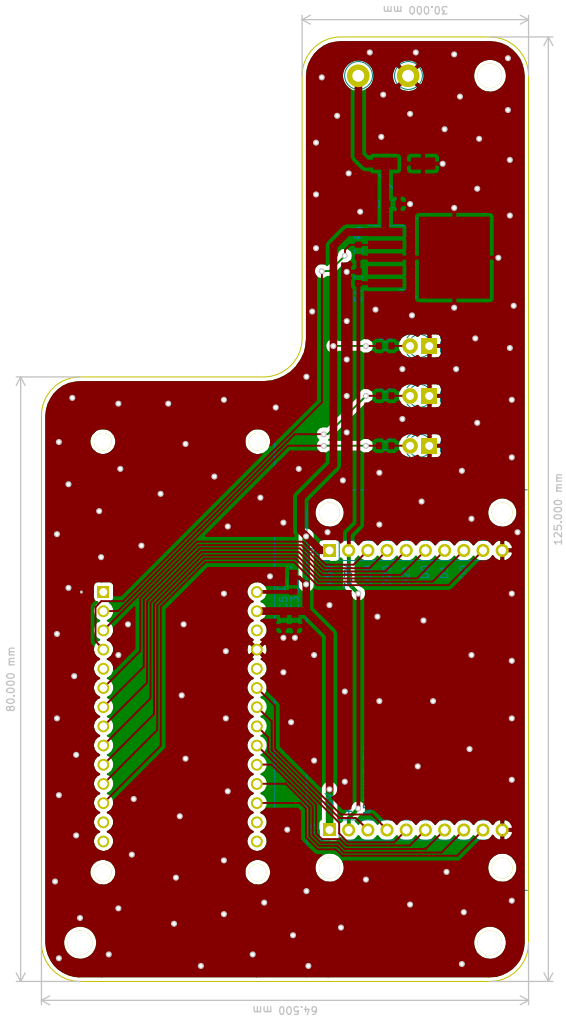
Name	Qnty	Vendor	Article Nr	Stückpreis
1591XXDSFLBK Hammond Gehäuse ABS w/Flanged Lid 6x3.2x1.8" Black	1	Mouser	546-1591XXDSFLBK	CHF 7.09
Abstandsbolzen M2.5 L=10mm	4	HSLU / Lasatec	560.251	CHF 0.29
PCBs, 1 X 125mmx64.5mm, 1 X 53mmx32mm	1	PCBWay	-	CHF 5.70

Total Preis:	CHF 133.78

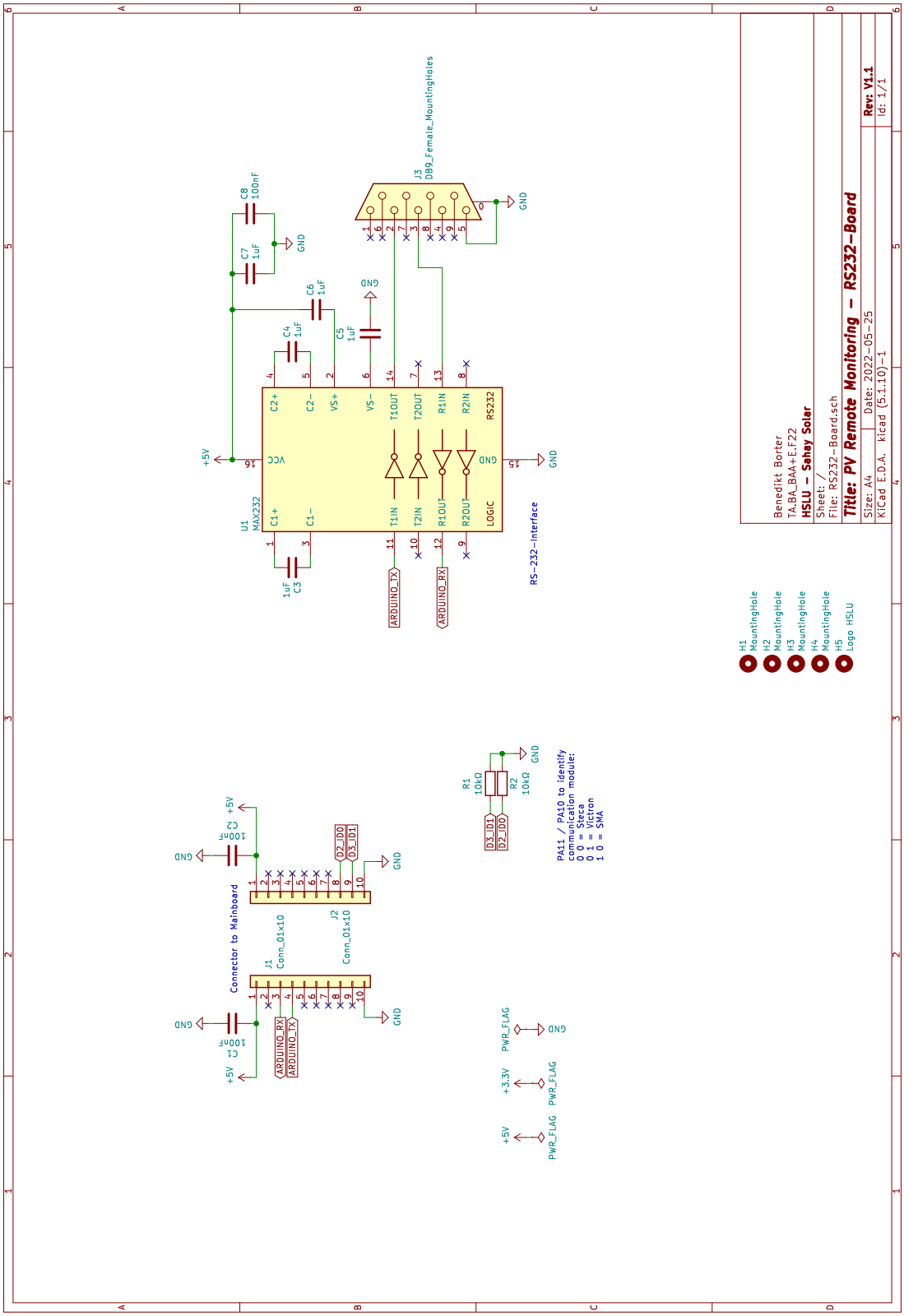
A.2 Schema Mainboard



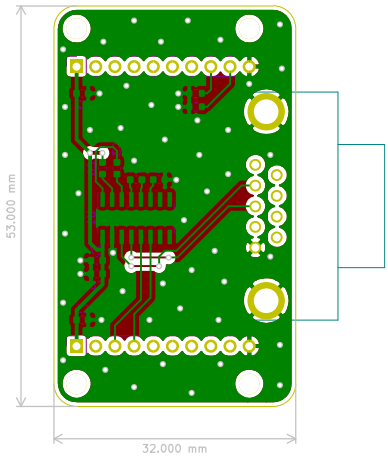
A.3 PCB Mainboard



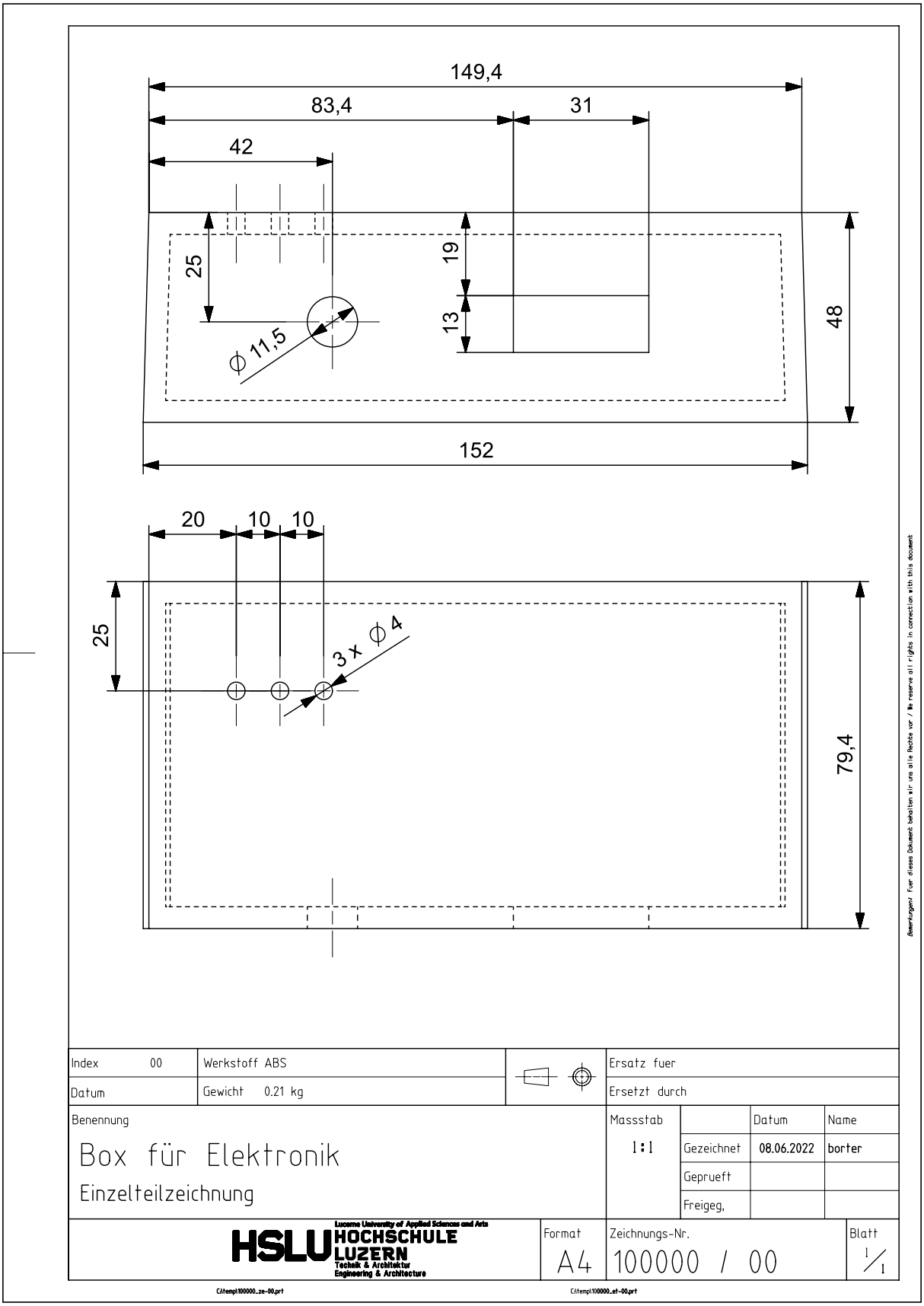
A.4 Schema RS-232-Board



A.5 PCB RS-232-Board



A.6 Plan Gehäuse



B.1 User Manual Remote Monitoring

User Manual: Sahay Solar - Remote Monitoring V1

Benedikt Borter

10. Juni 2022

1 Arduino programmieren

Bevor das Gerät eingesetzt wird, sollte die Software des Arduinos aktualisiert werden. Dazu wird die Arduino IDE benötigt (kann hier installiert werden). Es sind folgende Schritte durchzuführen:

- Die aktuelle Version der Software heisst *Arduino_SahayRemoteFinal*. In der Datei config.h sollten die untenstehenden Makros überprüft und allenfalls angepasst werden. Um das Projekt zu bearbeiten, wird in der Arduino IDE die Datei *Arduino_SahayRemoteFinal.ino* geöffnet.
- Das Makro *SERIAL_PRINTF_ON* darf nur auf 1 gesetzt sein, wenn über den *Serial Monitor* in der Arduino IDE die Debug-Informationen des Arduinos gelesen werden. Dies kann bei Problemen hilfreich sein. Ansonsten ist allerdings zwingend der Wert 0 einzugeben.
- Anschliessend muss die Software auf das Arduino geladen werden. Dazu öffnet man das Gehäuse und verbindet das Arduino per USB-Kabel mit dem Rechner (z.B. ein Windows-PC). In der Arduino IDE unter Tools:Port den COM-Port auswählen, an dem das Arduino MKR GSM 1400 angeschlossen ist. Mit dem Verify-Button das Projekt übersetzen und durch den Upload-Button die Software auf das Arduino laden.
- Erscheint eine Meldung wie in Abbildung 1, war der Upload erfolgreich.

```
#define SERIAL_PRINTF_ON      0 // for serial debugging, set to 0 for deployment
#define LOCATION              (String)"Kriens" // location of the solar system
#define INVERTER_TYPE         (String)"Steca" // inverter type of the solar system

#define INTERVAL_MEASURE      60 // in seconds
#define NUMBER_OF_DATAPOINTS 10 // number of datapoints collected and sent togheter
```

```
Write 139140 bytes to flash (2175 pages)
[==============================] 100% (2175/2175 pages)
done in 1.071 seconds

Verify 139140 bytes of flash with checksum.
Verify successful
done in 0.124 seconds
CPU reset.
```

Abbildung 1: Erfolgreicher Upload der Software auf das Arduino

1

2 Hardware

Bei der Benutzung der Box in Abbildung 2 sind folgende Punkte zu beachten:

- Das Gerät wird mit einer Gleichspannung von 5 V gespiesen. Dafür wird das Netzteil *SWH15-5B-EB-P5* verwendet. Höhere Spannungen können das Gerät beschädigen.
- Zur Kommunikation über RS-232 wird das mit dem Solrix PLI mitgelieferte serielle Kabel eingesetzt (siehe Beispiel Kommunikationsprotokoll Steca Solarix PLI 5000-48) und am RS-232-Anschluss des Steca Solarix PLI 5000-48 eingesteckt.
- Die rote LED leuchtet, sobald das Gerät gespiesen ist. Die RS-232-LED leuchtet jeweils nachdem das Arduino die erhaltenen Nachrichten vom Solrix PLI interpretieren kann. Die InfluxDB-LED leuchtet jeweils nachdem die Übertragung in die Datenbank abgeschlossen ist.
- Während der jeweiligen Kommunikation schaltet die entsprechende LED aus. Die RS-232-LED schaltet dabei für etwa 1-2 Sekunden, die InfluxDB-LED für etwa 10 Sekunden aus, da die jeweiligen Übertragungen diese Zeit in Anspruch nehmen.
- Leuchtet eine der LED für eine längere Zeit nicht auf, liegt bei der entsprechenden Schnittstelle, entweder bei der Kommunikation per RS-232 zum Solarix PLI oder mit der GSM-Verbindung bzw. Kommunikation in die Datenbank, ein Problem vor.
- Für zukünftige Produktionen sollte zwingend die Version V1.1 der Leiterplatten verwendet werden, da V1.0 Fehler beinhaltet.
- Bevor eine SIM-Karte eingesetzt wird, muss sie im Dashboard von Hologramm aktiviert werden.

Abbildung 2: Box mit angeschriebenen Schnittstellen

3 Zugriff auf Empfangssystem

Die Login-Informationen für den Zugriff auf das Empfangssystem (InfluxDB und Grafana) wie E-Mail und Passwörter befinden sich in einem separaten Text-File im elektronischen Anhang. Um sich anzumelden, können diese Links benutzt werden:

- InfluxDB Cloud 2.0: Datenbank, in der die Messwerte gespeichert werden, Werte werden ins SahayBucket geschrieben und können unter Data Explorer abgerufen werden.
  https://cloud2.influxdata.com/signup
- Grafana: Graphische Darstellung zur Überwachung der Solaranlagen, abrufbar unter:
  Grafana → Launch → Dashboards → Browse → SahaySolar PV Remote Monitoring (Steca)
  https://grafana.com/auth/sign-in/

Hinzu kommt das Login beim SIM-Karten-Anbieter:

- Hologram Dashboard: Verwalten und Aktivieren von SIM-Karten (innerhalb der *Organization* Sahay Solar), Einsicht in Datenverbrauch und Bezahlung.
  https://dashboard.hologram.io/

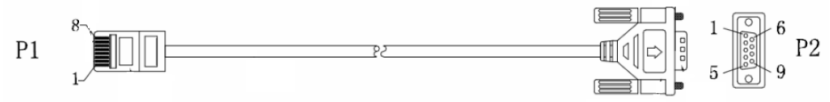
B.2 Beispiel Kommunikationsprotokoll Steca Solarix PLI 5000-48

Kommunikation mit dem Steca Solarix PLI 5000-48 Wechselrichter / Ladegerät mit MPPT Solar-Laderegler

Achtung: *vergewissern Sie sich, dass der Solarix PLI vollständig ausgeschaltet ist bevor Sie fortfahren. Stellen Sie weiterhin sicher, dass die Kabelabdeckung geschlossen ist, um die Gefahr zu vermeiden, unter Spannung stehende Teile zu berühren. Für die Kommunikation ist das Öffnen der Kabelabdeckung nicht notwendig.*

Dieses Dokument beschreibt die Kommunikation mit dem Solarix PLI 5000-48 um aktuelle Daten für das Datenlogging zu erhalten. Ein grafische Datenlogging- und Parameter-Software für die Einstellung der Solarix PLI Werte finden Sie auf der mit dem Solarix PLI mitgelieferten CD, dieses Dokument demonstriert eine alternative Kommunikationsmethode.

Bitte verwenden Sie eines der mit dem Solarix PLI mitgelieferten Kabel für die Kommunikation. Hierfür können Sie das USB Kabel benutzen (die Treibersoftware ist auf der mit dem Solarix PLI mitgelieferten CD zu finden), in diesem Fall wird nachdem Sie die Treibersoftware installiert und den Solarix PLI gestartet und über das Kabel angeschlossen haben, ein virtueller Kommunikations-Port erstellt. Alternativ können Sie auch das mit dem Solarix PLI mitgelieferte serielle Kabel verwenden um den Solarix PLI über den RS-232 Anschluss zu verbinden. Die Pinbelegung für das RS-232 Kabel lautet wie folgt (die linke Seite P1 in dem unten gezeigten Bild muss mit dem RS-232 Anschluss des Solarix PLIs verbunden werden, die rechte Seite P2 hingegen mit dem RS-232 Anschluss des Computers):



Verbindung zwischen P1 und P2:

P1	Farbe der Leitung	P2	Funktion
1	weiß und orange	2	TXD
2	blau	3	RXD
8	weiß und grün	5	GND

Unabhängig davon ob Sie den virtuellen COM Port per USB-Kabel oder das RS-232 Kabel verwenden, die aufgeführten seriellen Befehle gelten für beide Verbindungsmethoden.

Kommunikationsformat

Baudrate	Start Bit	Daten Bit	Paritätsbit	Stop bit
2400	1	8	N	1

Eingabebefehle

Die folgenden zwei Eingabebefehle „QPIGS“ und „QMOD“ ermöglichen die Abfrage aller aktuellen relevanten Daten des Solarix PLI.

QPIGS: Allgemeine Statusabfrage der Solarix PLI Parameter

Geben Sie die folgende Anfrage über die serielle Schnittstelle des Computers ein:

`QPIGS`<CRC><cr>

Während `QPIGS` die Abfrage darstellt, steht <CRC> für den *cyclic redundancy check* Code und <cr> für das Sonderzeichen Carriage-Return (Zeilenumbruch). Bei dieser Abfrage entspricht der CRC zwei Zeichen im Hexadezimal System (jedoch als Bytes gesendet): `B7A9`

Der Solarix PLI antwortet wie folgt:

```
(BBB.B CC.C DDD.D EE.E FFFF GGGG HHH III JJ.JJ KKK OOO TTTT EEEE
UUU.U WW.WW PPPPP b7b6b5b4b3b2b1b0 QQ VV MMMMM b10b9b8<CRC><cr>
```

Der zurückgesendete CRC Wert bestehend aus zwei Zeichen kann ignoriert werden. Die Bedeutung aller Werte wird in der untenstehenden Tabelle erläutert (jeder ASCII Wert wird durch ein Leerzeichen vom nächsten Wert getrennt):

	Wert	Beschreibung	Bemerkung
a	(	Start byte	
B	BBB.B	Netzspannung	B ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist V.
C	CC.C	Netzfrequenz	C ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist Hz.
D	DDD.D	AC Ausgangsspannung	D ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist V.
E	EE.E	AC Ausgangsfrequenz	E ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist Hz.
F	FFFF	AC Scheinleistung	F ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist VA.
G	GGGG	AC Wirkleistung	G ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist W.
H	HHH	Ausgangslast	H ist eine ganze Zahl zwischen 0 und 9. Die Zahl entspricht dem Prozentwert der Scheinleistung. 100% entspricht 5 kVA. Die Einheit ist %.
I	III	Interne Bus-Spannung	I ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist V.
J	JJ.JJ	Batteriespannung vom Wechselrichter gemessen	J ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist V.
K	KKK	Batterie-Ladestrom	K ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist A.
O	OOO	Batterie-Kapazität (ca.)	X ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist %.
T	TTTT	Kühlkörper-Temperatur	T ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist °C.
E	EEEE	PV Eingangsstrom	E ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist A.

U	UUU.U	PV Eingangsspannung	U ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist V.
W	WW.WW	Batteriespannung vom Laderegler gemessen	W ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist V.
P	PPPP	Batterie-Entladestrom	P ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist A.
Status bits 1	b7b6b5b4 b3b2b1b0	Gerätestatus	Jeder Wert `bX` entspricht einem einzelnen Status-Bit mit einem Wert von Eins oder Null. b7: SBU Prioritätsstatus: 1=ja, 0=nein b6: *reserviert* b5: *reserviert* b4: Last-Status: 0=Last aus, 1=Last an b3: *reserviert* b2: Ladestatus: 0=lädt, 1=lädt nicht b1: Solar-Ladestatus: 1=lädt, 0=lädt nicht b0: AC-Ladestatus: 1=lädt, 0=lädt nicht b2b1b0: 000: Kein laden 110: Laden mit Solar-Laderegler 101: Laden mit AC Quelle 111: Laden mit Solar-Laderegler und AC
Q	QQ	*reserviert*	*reserviert*
V	VV	EEPROM Version	V ist eine ganze Zahl zwischen 0 und 9.
M	MMMMM	PV Ladeleistung	M ist eine ganze Zahl zwischen 0 und 9. Die Einheit ist W.
Status bits 2	b10b9b8	Gerätestatus	b10: Kennzeichen für Ladeziel Schwebeladung b9: Eingeschaltet b8: *reserviert*

QMOD: Solarix PLI Abfragemodus

Geben Sie die folgende Anfrage über die serielle Schnittstelle des Computers ein:

`QMOD`<CRC><cr>

Während `QPIGS` die Abfrage darstellt, steht <CRC> für den *Cyclic Redundancy Check* Code und <cr> für das Sonderzeichen Carriage-Return (Zeilenumbruch). Bei dieser Abfrage entspricht der CRC zwei Zeichen im Hexadezimal System (jedoch als Bytes gesendet): `49C1`

Der Solarix PLI antwortet wie folgt:

`(M`<CR><cr>

Der zurückgesendete CRC Wert bestehend aus zwei Zeichen kann ignoriert werden. Wie auch in „QPIGS" beginnt auch hier die Antwort mit einem `(` Zeichen, welches ein Start-Byte darstellt. Alle möglichen Bedeutungen für `M` werden in der untenstehenden Tabelle aufgelistet:

CODE(M)	BEMERKUNG
P	An
S	Stand-by Modus
L	Netz Modus (By-Pass Relais geschlossen)
B	Batterie Modus
F	Fehler Modus
H	Energiespar Modus

B.3 Messungen und Berechnungen Datenmenge

t: Sendezyklus [min]	D: Daten pro Sendung [kB]	Verbrauch pro Tag [MB] $D_i*(24*60/t_i)/1024$
0	7.618	
1	7.83	11.011
2	8.042	5.655
3	8.254	3.869
4	8.466	2.976
5	8.678	2.441
6	8.89	2.084
7	9.102	1.829
8	9.314	1.637
9	9.526	1.488
10	9.738	1.369
11	9.95	1.272
12	10.162	1.191
13	10.374	1.122
14	10.586	1.063
15	10.798	1.012
16	11.01	0.968
17	11.222	0.928
18	11.434	0.893
19	11.646	0.862
20	11.858	0.834
21	12.07	0.808
22	12.282	0.785
23	12.494	0.764
24	12.706	0.744
25	12.918	0.727
26	13.13	0.710
27	13.342	0.695
28	13.554	0.681
29	13.766	0.668
30	13.978	0.655
31	14.19	0.644
32	14.402	0.633
33	14.614	0.623
34	14.826	0.613
35	15.038	0.604
36	15.25	0.596
37	15.462	0.588
38	15.674	0.580
39	15.886	0.573
40	16.098	0.566
41	16.31	0.559
42	16.522	0.553
43	16.734	0.547
44	16.946	0.542
45	17.158	0.536
46	17.37	0.531
47	17.582	0.526
48	17.794	0.521
49	18.006	0.517
50	18.218	0.512
51	18.43	0.508
52	18.642	0.504
53	18.854	0.500
54	19.066	0.497
55	19.278	0.493
56	19.49	0.489
57	19.702	0.486
58	19.914	0.483
59	20.126	0.480
60	20.338	0.477

Messwerte

Daten pro zusätzlichem Messwert [kB]

$X = (D_{10}-D_1)/(t_{10}-t_1) =$ 0.212

Overhead [kB]

$O = D_1 - X =$ 7.618

C.1 Zeitplanung

Aufgaben	SW01	SW02	SW03	SW04	SW05	SW06	SW07	SW08	SW09	SW10	SW11	SW12	SW13	SW14	Ansch
Initialisierungsphase															
Recherche															
Anforderungen															
Evaluation															
Lösungsvarianten															
Auswertung															
Konzept- und Realisierungsphase															
Design Sendesystem															
Realisierung Sendesystem															
Design Empfangssystem															
Realisierung Empfangssystem															
Abschlussphase															
Inbetriebnahme und Tests						1)				2)			3)		
Dokumentation															
Präsentationen															

1) Verbindung vom Sendesystem zum Empfangssystem testen
2) Sendesystem inkl. Kommunikation zu WR in Betrieb nehmen
3) Empfangssystem in Betrieb nehmen und das Gesamte testen

C.2 Aufgabenstellung

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

Horw, 21. Februar 2022
Seite 1/2

Bachelor Thesis im Studiengang Elektrotechnik und Informationstechnologie

Aufgabe für Herrn Benedikt Borter

Fernwartesystem für PV-Inselsysteme in Äthiopien

Fachliche Schwerpunkte

Energie- und Antriebssystem, Technische Informatik

Einleitung

Sahay Solar hat in Äthiopien bereits 30 PV Insel-Anlagen zur Elektrifizierung von Krankenstationen und Schulen gebaut. Da die Anlagen meist nicht gut erreichbar sind, gestaltet sich die Wartung als schwierig. Es braucht ein Monitoring-System für die Fernwartung.

Aufgabenstellung

Es soll ein Fernwartesystem entwickelt werden, das es ermöglicht den Zustand der PV-Inselsystemen zu bestimmen (z.B. durch Messung von Strom-/Spannungen von Solarpanels, Batterien, usw.). Dieses System soll möglichst universell für alle von Sahay Solar verbauten Systemen eingesetzt werden können.

Die Aufgabe der Studentin/des Studenten ist es zu recherchieren, was es bereits auf dem Markt gibt sowie was für Grössen sinnvoll zu messen wären. Aus den Erkenntnissen soll ein Systemdesign gemacht werden, welches dann in Form eines Prototyps umgesetzt wird. Gemessene Daten sollen aufgezeichnet, analysiert und auf dem Empfängersystem visualisiert werden können.

Die konkreten Spezifikationen sind mit dem Auftraggeber und dem Betreuer zu Beginn der Arbeit zu vereinbaren.

Termine

Start der Arbeit:	Montag 21.2.2022
Zwischenpräsentation:	Zu vereinbaren im Zeitraum 11.4. – 6.5.2022
Abgabe Schlussbericht:	Freitag 10. Juni 2022, vor 16:00 im D311
Abgabe Digitale Doku:	Gemäss separater Anweisung der Studiengangleitung
Abschlusspräsentation:	Zu vereinbaren im Zeitraum 13.6. – 1.7.2022
Diplomausstellung:	Freitag 8. Juli 2022 (Teilnahme obligatorisch!)

FH Zentralschweiz

Hochschule Luzern
Technik & Architektur

Horw, 21.2.2022
Seite 2/2
Diplomarbeit im Fachbereich
Elektrotechnik und Informationstechnologie

Dokumentation

Der gebundene Schlussbericht enthält keine Selbständigkeitserklärung und ist in einfacher Ausführung zu erstellen. Er enthält zudem zwingend

- einen sehr kurzen, englischen Abstract.
- Ein Titelblatt, gleich hinter dem Deckblatt, gemäss Weisungen der Studiengangleitung
- Eine SD-Hülle, innen, auf der Rückseite des Berichtes für den Betreuer

Alle Exemplare des Schlussberichtes müssen komplett und termingerecht gemäss Angaben der Studiengangleitung abgeben werden. Zusätzlich muss eine SD-Speicherkarte mit dem Bericht (inkl. Anhänge), dem Poster und den Präsentationen, Messdaten, Programmen, Auswertungen, usw. unmittelbar nach der Präsentation abgeben werden.
Die gesamte Dokumentation ist zudem gemäss Anweisungen der Studiengangleitung elektronisch auf einen Server zu laden. Sämtliche abzugebende Teile der Dokumentation sind Bestandteile der Beurteilung.

Fachliteratur/Web-Links/Hilfsmittel

Geheimhaltungsstufe: Öffentlich

Verantwortlicher Dozent/Betreuungsteam, Industriepartner

Dozent	Dr. Jonas Mühlethaler	jonas.muehlethaler@hslu.ch
Industriepartner	Sahay Solar (sahay-solar.ch) Schweizergasse 42 4054 Basel	
	Hr. R. Buser roger.buser@hslu.ch	Tel. 041 349 34 98
Experte	Martin Hässig martin.haessig@swissgrid.ch	Tel. +41 58 580 27 58

Hochschule Luzern
Technik & Architektur

Dr. Jonas Mühlethaler

D.1 Übersicht elektronischer Anhang (Zip-Datei TechDocs)

- 01_Anlagen_Sahay
  - Dokumente und Bilder zu gebauten Solaranlagen von Sahay Solar
  - Datenblätter, Manuals usw. von den Solaranlage-Komponenten
- 02_Sendesystem_Hardware
  - Links zu verwendeten Bauteilen
  - Pläne für Gehäuse
  - Gerberfiles für Produktion (ACHTUNG: V1.0 ist fehlerhaft)
  - Stücklisten, Schemas, PCBs
- 03_Sendesystem_KiCad
  - KiCad Projekt für Mainboard
  - KiCad Projekt für RS232-Board
  - Verwendete KiCad Libraries
- 04_Sendesystem_Software
  - Arduino: Arduino Projekte und Finale Version der Arduino Software (*Arduino_SahayRemoteFinal*)
  - Csharp_Demo: Visual Studio Solution mit C# Projekte zum Testen und Demonstrieren
  - rs232_nachricht.txt: Format der Antwort des Steca Solrix PLI 5000-48 bei RS-232-Statusabfrage
- 05_Empfangssystem
  - Grafana_Dashboard: Exportiertes Grafana Dashboard als JSON Datei
  - influxdb_api_token.txt
  - influxdb_curl.txt: cURL-Befehle, um mit HTTP Post Request von Windows CMD in InfluxDB zu schreiben
  - logins.txt: Login-Informationen für Email, Hologram, InfluxDB, Grafana
- 06_Weiteres
  - Aufgabenstellung
  - Test Datenmenge
  - User Manual Remote Monitoring