



# KONZEPTION UND REALISIERUNG EINER ENERGIEVERSORGUNG VIA SOLARMODULE FÜR EINPLATINENCOMPUTER

Frankfurt University of Applied Sciences  
Fachbereich 2 - Studiengang Informatik

## **Bachelorthesis**

zur Erlangung des akademischen Grades  
Bachelor of Science

vorgelegt von

**Daniel Mohr**  
geboren am 23.12.1989

1095797

<b>Erstprüfer:</b>	Prof. Dr. Christian Baun
<b>Zweitprüfer:</b>	Prof. Dr. Thomas Gabel

## Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Abschlussarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: \_\_\_\_\_ Unterschrift: \_\_\_\_\_

## **Danksagungen**

Hiermit möchte ich mich recht herzlich bei meiner Familie für die Unterstützung und Ratschläge bedanken. Dabei geht ein besonderer Dank an meinen Vater Stefan Mohr.

Zusätzlich bedanke ich mich auch hiermit an die Korrekturleser.

Weiterhin danke ich Prof. Dr. Christian Baun für die Bereitstellung dieses Themas und die zu Verfügung stehenden Mittel zur Bewerkstelligung dieser Arbeit. Genauso danke ich dem Zweitprüfer Herrn Prof. Dr. Thomas Gabel für die Annahme der Arbeit.

## **Zusammenfassung**

Diese Bachelorarbeit bezieht sich auf die Anwendung der erneuerbaren Energieversorgung mit Solarenergie und Batterie im Verbund für Einplatinencomputer. Die Überwachung des Einplatinencomputers, hier ein Raspberry Pi 3 Model B, wird zusätzlich von einem Skript übernommen. Die Aufgabe besteht darin, dass sich das Raspberry Pi bei nicht ausreichender Energieversorgung durch den Skriptbefehl ausschaltet. Zusätzlich soll eine individuelle Konstruktion der Energieversorgung via Solarmodule zeigen inwieweit diese den Einplatinencomputer mit Energie versorgen kann. Dieser Ansatz soll zeigen, dass eine alternative Energieversorgung möglich ist und zudem ein ordentliches Herunterfahren des Einplatinencomputers gewährleistet. Des Weiteren werden detailliert die einzelnen Schritte für den Aufbau des kompletten Systems beschrieben. Darunter fallen die physischen Komponenten und die Implementierung des Skripts. Grundlage der Bachelorarbeit sind Kenntnisse in den Fachbereichen der Informatik sowie Grundlagen der Elektronik.

## **Abstract**

The bachelor thesis presents the usage of alternative energy supply with cooperation of solar and battery for single board computers. The surveillance works with a script file on the single board computer, in this case a Raspberry Pi 3 module B. The primary task is to supply the Raspberry Pi with energy from solar and battery. If the energy supply is not enough, the Raspberry Pi should shut down through the script. Additionally, an individual solar powered construction should show how the energy supply is suitable for the single board computer. This approach gives an overview of the possibilities from alternative energy supply and the properly shut down of the Raspberry Pi. Furthermore, the thesis explains in detailed way how the system is build and which physical components were used as well as the implementation of the script. The basics of the thesis cover the areas of informatics and a little of electrical basics.

# Inhaltsverzeichnis

Eidesstattliche Erklärung . . . . .	2
Danksagungen . . . . .	3
Zusammenfassung . . . . .	4
Abstract . . . . .	4
<b>1 Einleitung</b>	<b>13</b>
1.1 Zielsetzung . . . . .	13
<b>2 Grundlagen</b>	<b>14</b>
2.1 Solarenergie / Solartechnik . . . . .	14
2.2 Raspberry Pi . . . . .	17
2.2.1 Raspberry Pi 3 Model B . . . . .	18
2.3 Der Schrittmotor . . . . .	21
2.4 SPI-Bus . . . . .	22
2.5 Grundlagen der Elektronik . . . . .	23
2.5.1 Stromkreis . . . . .	23
2.5.2 Ohmsche Gesetz . . . . .	24
2.5.3 Elektrische Masse . . . . .	24
2.5.4 Reihenschaltung und Parallelschaltung . . . . .	25
<b>3 Stand der Technik</b>	<b>27</b>
3.1 Klassifizierung von den Anforderungen . . . . .	27
3.2 Themenbezogene Veröffentlichungen . . . . .	28
3.2.1 Raspberry Pi mit Sonnenenergie betreiben . . . . .	28
3.2.2 Spannungsmessung mit AD-Wandler am Raspberry Pi . . . . .	30
3.3 Vergleich mit den Anforderungen . . . . .	33
3.4 Eingesetzte Technologie . . . . .	34
<b>4 Vorbereitung und Installation</b>	<b>41</b>
4.1 Installation Betriebssystem Raspbian . . . . .	41
4.2 Verbindungsmöglichkeiten mit dem Raspberry Pi ohne Peripherie . . . . .	42
4.2.1 Raspberry Pi über LAN oder WLAN . . . . .	42
4.2.2 Raspberry Pi mit PuTTY . . . . .	42
4.2.3 Raspberry Pi mit Remotedesktopverbindung . . . . .	43
4.3 Konfiguration Betriebssystem Raspbian . . . . .	44

<b>5</b>	<b>Zusammenführung der Komponenten</b>	<b>46</b>
5.1	Design des Gesamtkonzeptes . . . . .	46
5.2	System Steckplatine . . . . .	50
5.3	System Schaltplan . . . . .	51
5.4	Details zur Implementierung . . . . .	52
5.4.1	Parallelschaltung der Solarmodule . . . . .	52
5.4.2	Energieüberwachung des Akkus . . . . .	53
5.4.2.1	Aufbau Steckplatine . . . . .	53
5.4.2.2	Programme für die Energieüberwachung . . . . .	56
5.4.3	Ansteuerung der Schrittmotoren . . . . .	62
5.4.3.1	Verbindungsaufbau der Schrittmotoren mit dem Raspberry Pi . . . . .	62
5.4.3.2	Der Dienst cron . . . . .	64
5.4.3.3	Programme für die Ansteuerung der Schrittmotoren . . . . .	65
5.4.3.4	Implementierung Sonnenverlauf Algorithmus . . . . .	67
5.4.4	Aufbau Drehgestell . . . . .	68
5.4.4.1	Aufbau modifizierter Standfuß . . . . .	68
5.4.4.2	Aufbau Gestell . . . . .	70
<b>6</b>	<b>Ergebnisse</b>	<b>74</b>
6.1	Testlauf ohne Algorithmus . . . . .	74
6.2	Testlauf mit Algorithmus . . . . .	76
6.3	Testlauf der Energieüberwachung . . . . .	77
6.4	Anschlussfehler des Spannungsreglers (DC-DC-Wandler) . . . . .	78
<b>7</b>	<b>Diskussion</b>	<b>79</b>
7.1	Zusammenfassende Bewertung . . . . .	79
7.2	Ausblick . . . . .	80
<b>8</b>	<b>Anhang</b>	<b>85</b>
8.1	emailHandler.py . . . . .	85
8.2	stepperpitch.py . . . . .	88
8.3	stepperyaw.py . . . . .	91
8.4	sunsetsunrise.py . . . . .	94
8.5	Schrittmotor (28BYJ-48 – 5V Stepper Motor) Datenblatt . . . . .	97
8.6	Abmessungen zum Aufbau Gestell . . . . .	98
8.6.1	Solarmodul Rahmen Zusatzprofil . . . . .	98
8.6.2	Solarmodul Rahmen Abmessung . . . . .	99
8.6.3	Gestell Stirnseite Abmessung . . . . .	100
8.6.4	Zapfenband . . . . .	101
8.6.5	Zapfenband Abmessung . . . . .	102
8.7	Spannungsmessung des Systems . . . . .	103

# Abbildungsverzeichnis

2.1	Solarzelle Funktion. . . . .	14
2.2	Solarertrag nach Ausrichtung des Panels. . . . .	16
2.3	Raspberry Pi 3 Model B . . . . .	18
2.4	Raspberry Pi 3 Model B Pinbelegung . . . . .	20
2.5	Schrittmotor Halbschrittverfahren . . . . .	21
2.6	SPI-Bus . . . . .	22
2.7	Reihenschaltung. . . . .	25
2.8	Parallelschaltung . . . . .	26
3.1	Komponenten des Systems. . . . .	29
3.2	Sensorik für die Energieüberwachung. . . . .	30
3.3	Schaltbild für die Energieüberwachung. . . . .	31
3.4	12 Volt Akku . . . . .	34
3.5	Einplatinencomputer Raspberry Pi 3 Model B . . . . .	35
3.6	Solarmodule . . . . .	35
3.7	Solarladeregler . . . . .	36
3.8	DC-DC-Wandler (Spannungsregler) . . . . .	37
3.9	AD-Wandler mit Komponenten . . . . .	37
3.10	Schrittmotor . . . . .	38
3.11	Gestell mit Standfuß . . . . .	39
4.1	Programm für das Schreiben des Images auf MicroSD-Karte. . . . .	41
4.2	SSH Verbindung mit PuTTY. . . . .	43
4.3	Remotedesktopverbindung. . . . .	43
4.4	Einstellungen. . . . .	44
5.1	Steckplatine des Systems . . . . .	50
5.2	Schaltplan des Systems . . . . .	51
5.3	Parallelgeschaltete Solarmodule . . . . .	52
5.4	Verbindungsklemme (Wagoklemme) . . . . .	53
5.5	Steckplatine mit Komponenten . . . . .	54
5.6	Schaltplan Energieüberwachung . . . . .	55
5.7	nohup Ausgabe . . . . .	59
5.8	Verbindungen Schrittmotoransteuerung . . . . .	62
5.9	Schaltplan Schrittmotoransteuerung . . . . .	63
5.10	cron . . . . .	64
5.11	Schrittmotor internes Getriebe . . . . .	65

5.12	Modifizierter Standfuß . . . . .	68
5.13	Unterseite, Riemenantrieb . . . . .	69
5.14	Gestell Vorderseite . . . . .	70
5.15	Profile und Winkel . . . . .	71
5.16	Winde . . . . .	72
6.1	Entladekurve Akku . . . . .	74
6.2	Entladekurve Akku mit Einfluss der Solaranlage . . . . .	76
6.3	Spannungsregler Rückseite . . . . .	78
8.1	Schrittmotor Datenblatt . . . . .	97
8.2	Zusatzprofil für Solarmodule . . . . .	98
8.3	Solarmodul Rahmen Abmessung . . . . .	99
8.4	Stütze mit zweitem Rahmen . . . . .	100
8.5	Zapfenband . . . . .	101
8.6	Zapfenband . . . . .	102



# Tabellenverzeichnis

2.1	Stromverbrauch ohne angeschlossene Geräte . . . . .	17
5.1	Leerlaufspannung der Solarmodultypen . . . . .	47
8.1	Messungen ohne Sonnenverlauf Algorithmus . . . . .	103
8.2	Messungen mit Sonnenverlauf Algorithmus . . . . .	104

# Programme

3.1	Auslesen der Daten vom MCP3008 . . . . .	32
5.1	Skript rc.local . . . . .	56
5.2	MCP3008.py . . . . .	57
5.3	py_script.py . . . . .	58
5.4	settings.ini . . . . .	60
5.5	sendEmail.py . . . . .	61
8.1	emailHandler.py . . . . .	85
8.2	stepperpitch.py . . . . .	88
8.3	stepperyaw.py . . . . .	91
8.4	sunsetsunrise.py . . . . .	94

# Abkürzungsverzeichnis

<b>V</b>	Volt
<b>A</b>	Ampere
$\Omega$	Ohm
$\Phi$	Phi
<b>mA</b>	Milliampere
<b>mV</b>	Millivolt
<b>k<math>\Omega</math></b>	Kiloohm
<b>M<math>\Omega</math></b>	Megaohm
<b>W</b>	Watt
<b>Ah</b>	Amperestunde
<b>USB</b>	Universal Serial Bus
<b>MHz</b>	Megahertz
<b>GHz</b>	Gigahertz
<b>Bit</b>	binary digit
<b>MBit</b>	Megabit
<b>GB</b>	Gigabyte
<b>RAM</b>	Random Access Memory
<b>WLAN</b>	Wireless Local Area Network
<b>LAN</b>	Local Area Network
<b>MicroSD</b>	Micro Secure Digital
<b>GPIO</b>	General Purpose Input Output
<b>DSI</b>	Display Serial Interface

<b>CSI</b>	Camera Serial Interface
<b>LED</b>	light-emitting diode - Leuchtdiode
<b>GND</b>	Ground
<b>HAT</b>	Hardware Attached on Top
<b>SPI</b>	Serial Peripheral Interface
<b>SSH</b>	Secure Shell
<b>IP</b>	Internetprotokoll
<b>Poti</b>	Potentiometer
<b>API</b>	Application Programming Interface
<b>URL</b>	Uniform Resource Locator
<b>CSV</b>	Comma-separated values
<b>JSON</b>	JavaScript Object Notation

# 1 Einleitung

Die Energienutzung im Allgemeinen gesehen erfährt eine neue Umstrukturierung. Fossile Brennstoffe wie Gas, Öl und Kohle gehen langsam zur Neige und sind zugleich für die Energieerzeugung der Verbraucher umweltschädlich und keine langfristige Ressource für eine effiziente und klimafreundliche Energieerzeugung. Aus diesem Grunde sind alternative / erneuerbare Energiequellen von immenser Bedeutung für unsere Umwelt. Eine beispielhafte Alternative ist die Solarenergie. Die Sonne stellt eine unerschöpfliche Energiequelle dar. Leider ist diese Energiequelle nur tagsüber und auch meistens nicht zu hundert Prozent zuverlässig nutzbar. Wechselnde Wetterverhältnisse können die Leistung mindern. Nichtsdestotrotz stellt diese Energiequelle für einige Verbraucher die perfekte alternative Energieversorgung zur Verfügung. Für kleine bis mittelgroße Angelegenheiten können im Hardwaresegment sogenannte Einplatinencomputer von großen Nutzen sein. Diese Computersysteme besitzen meistens die Größe einer Kreditkarte und haben alle notwendigen elektronischen Komponenten auf der Platine installiert. Ein beliebtes und zum Teil einfach zu bedienendes Gerät wäre das Raspberry Pi. Der Raspberry Pi wurde ursprünglich für junge Anwender entwickelt um einen einfachen Einblick in die Programmier- und Hardware Welt zu bekommen. Anwendung finden die Einplatinencomputer heutzutage größtenteils in der Industrie für Mess-, Steuer- und Regelungstechnik[1]. Somit sind diese kleinen Platinen perfekt konstruiert um spezielle Aufgaben zu bewältigen. Dabei ist auch der Stromverbrauch dieser Platinen sehr gering und stellt in Zusammenarbeit mit Solarenergie eine wichtige und effiziente Konstruktion dar.

## 1.1 Zielsetzung

Das Ziel dieser Arbeit ist eine alternative Energieversorgung eines Einplatinencomputers (Raspberry Pi 3 Model B) darzustellen. Die Konstruktion gewährleistet eine rund um die Uhr Versorgung des Einplatinencomputers. Zu den Komponenten der Konstruktion gehören die Solarpaneele, ein Solarladeregler, eine aufladbare Batterie für die nächtliche Versorgung und ein AD-Wandler, welcher die analogen Signale in digitale Signale umwandelt. Diese Signale werden dann für die kontrollierte Energieversorgung benutzt um daraufhin bei nicht ausreichender Energie den Raspberry Pi mit Hilfe eines Skriptes ordnungsgemäß herunterfahren zu lassen. Die Solaranlage ist so konstruiert, dass diese eine optimale Ausnutzung der Sonnenstrahlen mit Hilfe eines Programms sicherstellt. In den weiteren Kapiteln wird dann detailliert auf die Konstruktion beziehungsweise Aufbau eingegangen.

## 2 Grundlagen

Zur Einführung in die technischen Aspekte der Arbeit folgen nun die Grundlagen, die für ein besseres Verständnis sorgen. Als erstes wird der Begriff Solarenergie näher erklärt und dazu die notwendige Technik erläutert. Des Weiteren folgt der allgemeine Aufbau des Raspberry Pi und dazu noch einige Grundlagen für das Verständnis der Elektronik.

### 2.1 Solarenergie / Solartechnik

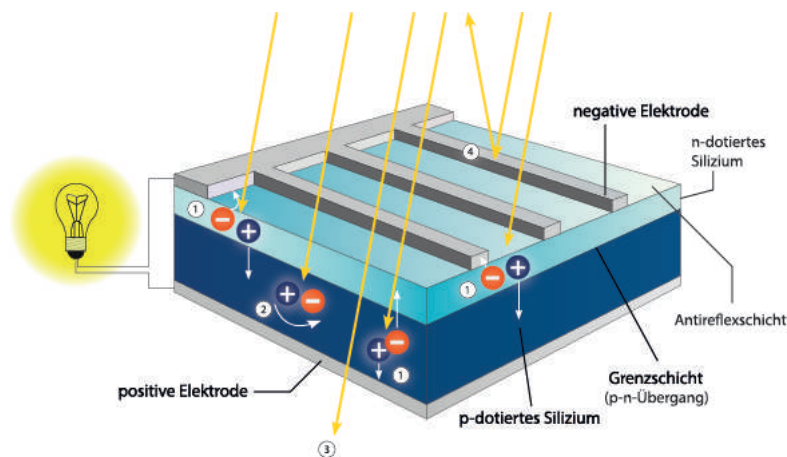


Abbildung 2.1: Solarzelle Funktion<sup>1</sup>

<sup>1</sup>Bildquelle: <http://www.solarladen.de/photovoltaik-funktion>.

Solarenergie bezeichnet im Allgemeinen den technischen Nutzen der Sonnenstrahlen durch den Menschen. Die Technik für die Gewinnung der Energie kann auf verschiedene Art und Weise erfolgen. Eine davon ist die sogenannte Photovoltaik. Diese Technik dient zur direkten Umwandlung der Sonnenstrahlen in elektrische Energie via Solarzellen.

Die Funktion einer Solarzelle beruht auf dem Halbleitermaterial Silizium (Si). Wenn Licht in die Solarzelle hineinscheint, erzeugt dieser Vorgang freie, geladene Teilchen - "der innere Photoeffekt".[2]

Die meisten Solarzellen besitzen eine Gleichspannung von 0.5 V. Das kommt aber immer auf das verwendete Material in den Solarzellen an, beispielsweise Silizium. Bei Vergrößerung oder Verkleinerung der Solarzellen, bleibt die Gleichspannung trotzdem gleich. Bei der Stromstärke ist es hingegen anders. Bei einer Vergrößerung der Solarzellen Fläche nimmt auch die Stromstärke zu sowie bei einer stärkeren Beleuchtung.[3]

### **Innere Photoeffekt:**

In der Solarzelle befinden sich Elektronen (negativ geladene Teilchen), die eine feste Position innerhalb der Solarzelle haben. Die Positionen kann man sich wie eine Gitterform vorstellen. Sobald genug Licht in die Solarzelle hinein scheint, verlassen die negativ geladenen Elektronen ihre Verankerung beziehungsweise das Loch. Dadurch entsteht in dem freigewordenen Loch eine positive Ladung, welche dann für den Strom zum Teil verantwortlich sind. In diesem Zusammenhang gibt es einen kleinen Nachteil. Die freigewordenen Elektronen sind nur kurzzeitig entfesselt und neigen dazu sich schnell wieder mit dem Loch zu verbinden. Dieser Prozess ist natürlich unerwünscht, weshalb der Photovoltaische Effekt hinzukommt.[2]

### **Photovoltaischer Effekt:**

Für die bessere Leitfähigkeit sind die Halbleiter "dotiert"[4]. Dazu sind Fremdatome im Halbleiterkristall integriert. Diese Fremdatome können zum Beispiel Bor und Phosphor sein. Der Photovoltaische Effekt dient zur Verhinderung der Zusammenführung von freien Elektronen und den Löchern. Diese Aufgabe übernimmt der pn-Übergang. Der pn-Übergang befindet sich zwischen zwei Halbleiterschichten (siehe Abbildung 2.1). Das sind zum einen die p-dotierte Schicht und die n-dotierte Schicht. In der p-dotierten Schicht befinden sich eine Überzahl an Löchern und in der n-dotierten Schicht eine Überzahl an freien Elektronen. Somit entstehen freie positive und negative Ladungen, die daraufhin bei Hinzuschaltung eines Verbrauchers sich bewegen und für den Strom des Verbrauchers sorgen.[2]

### **Solaranlage Ausrichtung:**

Diese Technik ist natürlich nur tagsüber und bei ausreichenden Sonnenstrahlen optimal nutzbar. Dabei müssen einige Kriterien beachtet werden. Im Idealfall sollte die Solaranlage in Richtung Süden zeigen um die Mittagssonne am besten auszunutzen. Der Neigungswinkel der Solaranlage spielt hierbei auch eine wichtige Rolle. Die optimale Energiegewinnung erzielt man bei einer senkrechten Sonneneinstrahlung auf die Solaranlage. In Deutschland liegt der ideale Neigungswinkel zwischen 30 und 36 Grad. Dabei ist natürlich darauf zu achten ob die Solaranlage sich in Süd- oder Norddeutschland befindet. In Norddeutschland müsste der Neigungswinkel ein wenig steiler eingestellt sein als in Süddeutschland, da dort die Sonne am Himmel etwas flacher steht. Die folgende Grafik zeigt den Ertrag bei verschiedenen Ausrichtungen der Solaranlage.[5]

		Dachausrichtung																		
		Süd		Südost Südwest						Ost West		Nordost Nordwest						Nord		
		0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180
Dachneigung	0°	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%	87%
	10°	93%	93%	93%	92%	92%	91%	90%	89%	88%	86%	85%	84%	83%	81%	81%	80%	79%	79%	79%
	20°	97%	97%	97%	96%	95%	93%	91%	89%	87%	85%	82%	80%	77%	75%	73%	71%	70%	70%	70%
	30°	100%	99%	99%	97%	96%	94%	91%	88%	85%	82%	79%	75%	72%	69%	66%	64%	62%	61%	61%
	40°	100%	99%	99%	97%	95%	93%	90%	86%	83%	79%	75%	71%	67%	63%	59%	56%	54%	52%	52%
	50°	98%	97%	96%	95%	93%	90%	87%	83%	79%	75%	70%	66%	61%	56%	52%	48%	45%	44%	43%
	60°	94%	93%	92%	91%	88%	85%	82%	78%	74%	70%	65%	60%	55%	50%	46%	41%	38%	36%	35%
	70°	88%	87%	86%	85%	82%	79%	76%	72%	68%	70%	58%	54%	49%	44%	39%	35%	32%	29%	28%
	80°	80%	79%	78%	77%	75%	72%	68%	65%	61%	56%	51%	47%	42%	37%	33%	29%	26%	24%	23%
90°	69%	69%	69%	67%	65%	63%	60%	56%	53%	48%	44%	40%	35%	31%	27%	24%	21%	19%	18%	

 Abbildung 2.2: Solarertrag nach Ausrichtung des Panels<sup>2</sup>
<sup>2</sup>Bildquelle: [https://bilder.pcwelt.de/3872217\\_original.jpg](https://bilder.pcwelt.de/3872217_original.jpg)

Wie schon beschrieben sind dies die optimalen Kriterien um den besten Ertrag zu erzielen. Bei flachem Lichteinfall auf die Solarmodule ist die Reflexion größer als bei senkrechtem Einfall. Dadurch kann ein Teil der zu nutzenden Sonnenstrahlen verloren gehen und somit die optimale Ausnutzung. Abhilfe schafft hierbei eine Antireflexbeschichtung. Bei nicht optimaler Ausrichtung, ist der Verlust des Ertrages jedoch gering, wie man auf Abbildung 2.2 sehen kann. Zeigen die Solarmodule genau nach Süden, führen Abweichungen des Neigungswinkels nicht zu großen Verlusten des Ertrages. Beispielsweise ist der prozentuale Betrag bei exakter Südausrichtung von 70 Grad immerhin noch 88 Prozent. Diese prozentualen Angaben können jedoch abweichen, wenn man unterschiedliche Antireflexionsschichten verwendet.[5]



## 2.2 Raspberry Pi

Der Einplatinencomputer Raspberry Pi wurde von der Raspberry Pi Foundation im Februar 2012 entwickelt. Ursprünglich war die Idee ein Gerät zu entwickeln, welches die Informatik bzw. Programmierung näher bringen sollte[1]. Über die Jahre hinweg wurden diese Einplatinencomputer immer leistungsfähiger und boten viele verschiedene Schnittstellen für zusätzliche Peripherie, die an die Einplatinencomputer angeschlossen werden können. Das bedeutet hingegen auch eine höhere Stromversorgung der Einplatinencomputer, welcher im Bezug auf die Realisierung der Stromversorgung via Solarmodule ein wichtiger Faktor ist. Für den Vergleich des Verbrauches folgen nun einige Raspberry Pi Modelle.

Modell	Spannung	Strom	Leistung	Strom(höchster Wert)
RPi A	5,157 V	0,116 A	0,603 W	0,198 A
RPi A+	5,160 V	0,085 A	0,438 W	0,150 A
RPi Zero	5,166 V	0,084 A	0,433 W	0,170 A
RPi B	5,139 V	0,341 A	1,752 W	0,422 A
RPi B+	5,150 V	0,203 A	1,045 W	0,264 A
RPi 2 B	5,151 V	0,199 A	1,025 W	0,345 A
RPi 3 B	5,134 V	0,265 A	1,360 W	0,541 A

Tabelle 2.1: Stromverbrauch ohne angeschlossene Geräte [6]

Alle Werte bis auf die des Raspberry Pi 3 Model B wurden nicht persönlich gemessen und dienen nur als Vergleich zur besseren Veranschaulichung. Die Tabelle zeigt einen Vergleich unter verschiedenen Raspberry Pi Modellen. Die Spalte Modell beinhaltet die Raspberry Pi Modelle. Die Spalte Spannung zeigt die benötigte Spannung, damit der Raspberry Pi ordnungsgemäß läuft. Die erforderliche Spannung (Betriebsspannung) beträgt 5 bis 5.1 V[7]. Die Spannung kann auch unter den erforderlichen 5 V liegen. Bei 4,75 V sollte man aber darauf achten, dass der Raspberry Pi nicht in einen instabilen Zustand gerät[7]. Falls der Raspberry Pi eine USB-Schnittstellen hat, schaltet der Raspberry Pi diese bei einer Unterversorgung unter 5 V ab. USB-Schnittstellen brauchen eine Nennspannung von 5 V[8]. Somit sollte man darauf achten, dass der Raspberry Pi immer die erforderliche Spannung zur Verfügung gestellt bekommt. Die nächste Spalte Strom zeigt den Stromverbrauch der Geräte an. Die Spalte Leistung zeigt das Produkt aus Spannung und Strom an. Die elektrische Leistung  $P$  ist das Produkt der elektrischen Spannung  $U$  und der elektrischen Stromstärke  $I$ [9].

$$P = U \cdot I \quad (2.1)$$

Die letzte Spalte Strom (höchster Wert) soll zum Beispiel bei Spannungsschwankungen bei einer bestimmten Operation den höchsten Wert anzeigen. Schwankungen können während des Hochfahrens des Raspberry Pis anliegen. Nun sind folgenden Werte von der Spalte Strom wichtig für die Entscheidung welches Raspberry Pi Modell für die Realisierung der Energieversorgung via Solarmodule benutzt wird. Die Überlegung ein Modell mit wenig Stromverbrauch zu nehmen schlägt sich natürlich bei der Auswahl der Komponenten für die Stromversorgung nieder. Genauso sieht es mit dem Modell mit der höchsten Stromversorgung aus.

Die Auswahl des Modells ist dahingehend entscheidend, da eine Stromversorgung für den beispielsweise RPi Zero mit weniger Solarmodule oder beziehungsweise schwächere Solarmodule genügt. Dabei kann aber eine leistungsstärkere Variante, wie der RPi B, mit dieser Konstruktion nicht betrieben werden. Deswegen ist eine Konstruktion der Stromversorgung für ein leistungsstärkeres Modell mit hohem Stromverbrauch empfehlenswerter, da man diese Konstruktion auch für schwächere Modelle mit weniger Stromverbrauch benutzen kann. Zudem soll das Raspberry Pi Modell für etwaige Vorhaben gerüstet sein. Das könnten zum Beispiel zusätzliche Geräte wie Maus, Tastatur oder auch ein Bildschirm sein. Die Wahl fällt somit auf das Raspberry Pi 3 Model B, welches in dem Sortiment der Modelle auch einen hohen Stromverbrauch vorweist. Der RPi B hat im Grunde genommen einen höheren Stromverbrauch, bietet aber von der Hardwareleistung und den Schnittstellen weniger als das Raspberry Pi 3 Model B.

### 2.2.1 Raspberry Pi 3 Model B

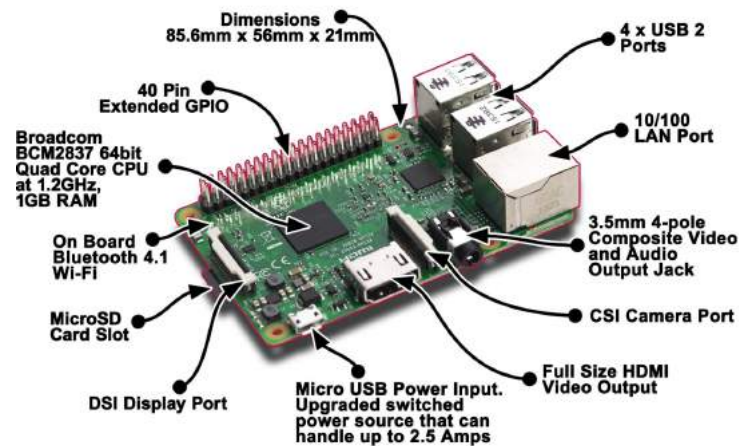


Abbildung 2.3: Raspberry Pi 3 Model B<sup>3</sup>

<sup>3</sup>Bildquelle: <https://www.play-asia.com/raspberry-pi-3-model-b/13/709v93>

Das Raspberry Pi 3 Model B wurde im Februar 2016 entwickelt und ist gegenüber den anderen Modellen leistungsstärker geworden[1]. Die bessere Leistung bezieht der Raspberry Pi von einem 64-Bit-Prozessor, der mit 1200 MHz taktet. Im Vergleich zum RPi 2 B sind das 300 MHz mehr[1]. Die 1 GB RAM Arbeitsspeicher bleiben nach wie vor. Zwei weitere Neuerungen sind zum einen das integrierte WLAN mit 2,4 GHz und zusätzlich noch Bluetooth 4.1 Funktion. Der MicroSD Card Slot (MicroSD Karten Schacht), der sich auf der Unterseite der Platine befindet, dient für die MicroSD-Karte auf welcher das Betriebssystem installiert ist. Für die Internetverbindung oder eine Verbindung mit einem anderen Netzwerk kann man die integrierte WLAN-Funktion oder den 10/100-MBit-Ethernet Anschluss benutzen. Es besteht unter anderem die Möglichkeit einen Bildschirm direkt über HDMI anzuschliessen oder über den DSI Display Port (DSI Bildschirm Anschluss). An dem DSI Display Port ist es möglich ein 7 Zoll Touchdisplay anzuschliessen, welches seit September 2015 erhältlich ist[1].

Über den HDMI-Anschluss schließt man normale TV-Geräte oder Monitore an. Zusätzlich kann man über den CSI Camera Port (CSI Kamera Anschluss) eine Kamera anschließen. Des Weiteren gibt es für den analogen Audio Ausgang eine 3,5 mm Klingenbuchse und für den digitalen Audio Ausgang ist der HDMI-Anschluss zusätzlich verantwortlich[1]. Im Bezug auf die erhöhte Leistung der 64-Bit Architektur muss natürlich der 64-Bit-Modus aktiviert sein. Standardmäßig läuft der Raspberry Pi 3 Model B im 32-Bit-Modus, da das Image vom Betriebssystem auch eine 32-Bit-Version beinhaltet[10]. Darüber hinaus sind die 1 GByte RAM Arbeitsspeicher zu wenig um mit der erhöhten 64-Bit-Architektur arbeiten zu können. Diese Einstellung gibt dem Raspberry Pi 3 Model B wahrscheinlich nur aufgrund der höheren Taktfrequenz einen kleinen Geschwindigkeitsvorteil[10]. Eine wichtige Komponente ist die Schnittstelle GPIO, welche in dem nächsten Abschnitt ausführlich erklärt wird.

## GPIO (General Purpose Input Output):

Die GPIO-Schnittstelle gibt die Möglichkeit programmierbare Ein - und Ausgänge für Geräte wie zum Beispiel Sensoren oder auch LEDs zu benutzen. Der Raspberry Pi kann über die GPIO-Schnittstelle digitale Signale senden und digitale Signale empfangen[11]. Die GPIO-Schnittstelle des Raspberry Pi 3 Model B besitzt 40 Pins. Die folgende Abbildung zeigt die doppelreihige Stiftleiste mit den Bezeichnungen für die Pinbelegung.

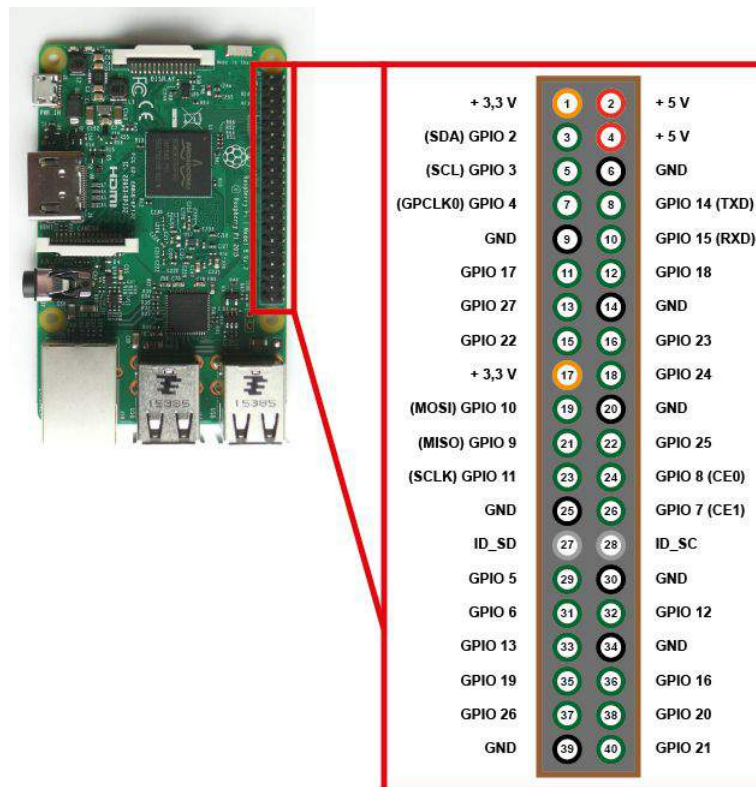


Abbildung 2.4: Raspberry Pi 3 Model B Pinbelegung<sup>4</sup>

<sup>4</sup>Bildquelle: <https://www.elektronik-kompodium.de/sites/raspberry-pi/1907101.htm>

Die grün umrandeten 26 Pins dienen für die programmierbaren Ein - und Ausgänge, welche Daten von Geräten oder digitalen Schaltungen empfangen, sowie senden können. Die orange umrandeten Pins 1 und 17 verfügen über eine 3,3 V und ungefähr 50 mA mit denen man höchstens LEDs betreiben kann[12]. Die rot umrandeten Pins 2 und 4 sind ebenso für eine 5 V Stromversorgung verantwortlich. Dabei können diese Pins als Ausgangsversorgung für andere Verbraucher dienen, oder eine externe Stromquelle versorgt den Raspberry Pi über einen dieser Pins mit Strom. Als nächstes folgen zwei grau umrandete Pins 27 und 28. Diese Pins sind für HAT Erweiterungsplatinen des Raspberry Pi zuständig. Diese Platinen steckt man direkt auf die doppelreihige Stiftleiste des Raspberry Pis[13]. Die nächsten sechs schwarz umrandeten Pins sind für die elektrische Masse (GND) zuständig. Die elektrische Masse gibt in einer Schaltung das Bezugspotential an, welches immer 0 V ist und am Minuspol liegt.

## 2.3 Der Schrittmotor

Der Schrittmotor, kurz Stepper genannt, spielt in der Konstruktion eine wichtige Rolle. Als erstes soll die Funktionsweise erläutert werden. Die Abbildung 2.5 zeigt den Aufbau eines Schrittmotors mit dem Halbschritt-Antrieb. Der Schrittmotor besteht aus einem Permanentmagneten, den Rotor und die vier Elektromagneten, den Stator. Die unipolare Bauweise ermöglicht eine leichtere Ansteuerung. Es müssen nur die Magnete ein- oder ausgeschaltet werden.[14]

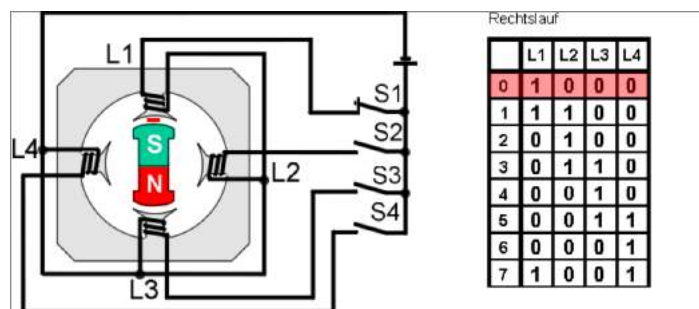


Abbildung 2.5: Schrittmotor Halbschrittverfahren<sup>5</sup>

<sup>5</sup>Bildquelle: <http://www.tuf-ev.de/workshop/schrittmotor/halbschr.htm>

Neben dem Halbschritt-Antrieb gibt es auch noch den Vollschritt-Antrieb. Dieser soll aber nicht das Thema sein, denn in den Programmen kommt nur der Halbschritt-Antrieb zum Einsatz. In der Abbildung 2.5 sieht man die Wertetabelle für den Rechtslauf des Schrittmotors. Die Wertetabelle zeigt alle vier Spulen und die acht Schritte. Eine eins aktiviert den Elektromagneten und die null zeigt die Deaktivierung des jeweiligen Elektromagneten. Anhand der Wertetabelle sieht man nun, dass der Rotor in den Schritten 0,2,4 und 6 vor den Elektromagneten steht. In den anderen Schritten ist der Rotor immer zwischen den jeweils mit einer eins angesteuerten Elektromagneten.[14]

## 2.4 SPI-Bus

Der SPI-Bus findet seine Anwendung im Programm für die Energieüberwachung. Das Verständnis für die Funktion soll hier schon mal die grundsätzliche Kommunikation zwischen zwei Geräten erläutern.

Die Abbildung 2.6 zeigt die notwendigen Leitungen für den Aufbau.

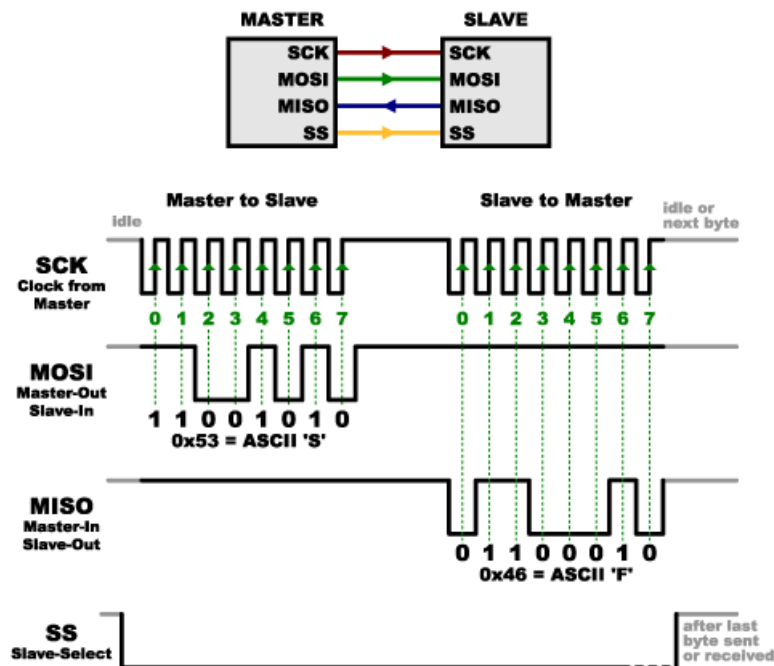


Abbildung 2.6: SPI-Bus<sup>6</sup>

<sup>6</sup>Bildquelle: <https://cdn.sparkfun.com/assets/c/7/8/7/d/52ddb2dcce395fed638b4567.png>

Die SCK (Serial Clock) Leitung führt vom Master zum Slave und stellt die Taktleitung dar. MOSI (Master Out / Slave In) dient zum Senden von Daten vom Master zum Slave. Umgekehrt sendet der Slave die Daten zum Master mit der MISO (Master In / Slave Out) Leitung. Die SS (Slave Select) oder auch manchmal CS Leitung bestimmt den Slave, der Daten an den Master senden oder auch selbst empfangen kann. Dabei ist die SS-Leitung auf LOW gesetzt. Im Normalzustand ist die SS-Leitung auf HIGH gesetzt, wobei der Slave nicht mit dem Master verbunden ist. Die MOSI und MISO Leitungen sind unabhängig voneinander und somit ergibt sich ein Vollduplex-Bus. Das bedeutet gleichzeitiges Senden und Empfangen.[15]

Für eine Kommunikation zieht der Master die SS-Leitung nach unten. Danach kann der Master an den Slave senden oder der Slave kann auch direkt an den Master etwas senden. Nach der Kommunikation zieht der Master die SS-Leitung wieder nach oben und der Slave ist somit nicht mehr verbunden.[15]

## 2.5 Grundlagen der Elektronik

Das Unterkapitel beschäftigt sich mit den Grundlagen der Elektronik, die in dieser Arbeit von Bedeutung sind um eine richtige Konzeption dieses Aufbaus im Bezug auf die Elektronik zu erstellen.

### 2.5.1 Stromkreis

Für einen funktionierenden Stromkreis benötigt man beispielsweise drei Komponenten:

- Stromquelle (Batterie)
- Verbraucher (Glühlampe)
- Leitfähiges Material (Kupferdraht)

Somit kann man eine ganz einfache Schaltung realisieren. Doch wie funktioniert diese Schaltung im Detail? Für die Stromrichtung gibt es nochmal zwei weitere Begriffe.

Elektrischer Strom:

Der elektrische Strom fließt vom Pluspol zum Minuspol. Diese Definition wurde noch damals entworfen, als man von den wirklichen Vorgängen im Stromkreis noch keine Ahnung hatte.[16]

Elektronenstrom:

Der Elektronenstrom fließt vom Minuspol zum Pluspol. Hierbei handelt es sich um die physikalische Betrachtung des Stromflusses und somit auch die Richtige. Denn in den Leitungen fließen die Elektronen, die negativ geladenen Teilchen, die zum Pluspol streben.[16]

Auch wenn die Definition des elektrischen Stroms falsch ist, ist sie trotzdem noch in Verwendung. Im Grunde genommen kann man sich die Begriffe wie folgt einordnen. Die Definition des Elektronenstroms findet Anwendung bei Schaltungen und die Definition des elektrischen Stromes bei physikalischen Vorgängen innerhalb des Stromkreises.[16]

### 2.5.2 Ohmsche Gesetz

$$\text{elektrische Stromstärke} = \frac{\text{elektrische Spannung}}{\text{elektrischer Widerstand}} \quad (2.2)$$

Die Funktion dieses Gesetzes sagt aus, dass wenn sich bei gegebener elektrischen Spannung, der elektrische Widerstand verringert, führt dies zu einer Erhöhung der elektrischen Stromstärke. Durch Erhöhung der elektrischen Stromstärke, ist gleichzeitig auch die Wärme in den Leitungen, durch die der Strom fließt, höher. Deswegen sollten die Bauelemente in der Schaltung gut durchdacht sein, damit keine Schäden an Bauteilen entstehen können[17].

Kurzschreibweise:

$$I = \frac{U}{R} \quad R = \frac{U}{I} \quad U = R \cdot I \quad (2.3)$$

Maßeinheiten:

Für die elektrische Spannung steht das Volt (V). Die elektrische Stromstärke ist in Ampere (A) angegeben und für den elektrischen Widerstand ist das Ohm ( $\Omega$ ) zuständig[18]. In der Praxis benutzt man noch andere Einheiten. Hier eine Auflistung:

$$\begin{array}{ll} 1 \text{ A} = 1000 \text{ mA} & 1 \text{ mA} = 0,001 \text{ A} \\ 1 \text{ V} = 1000 \text{ mV} & 1 \text{ mV} = 0,001 \text{ V} \\ 1 \Omega = 0,001 \text{ k}\Omega & 1 \text{ k}\Omega = 1000 \Omega \\ & 1 \text{ M}\Omega = 1000 \text{ k}\Omega \end{array}$$

### 2.5.3 Elektrische Masse

Folgendes Beispiel erläutert die elektrische Masse:

Es soll nun eine kleine Schaltung vom Raspberry Pi über die GPIO-Schnittstelle mit Strom versorgt werden. Dazu benutzt man den Pin 2 mit 5 V und den Pin 5 als elektrische Masse (GND). Von Pin 2 (Pluspol) führt eine Leitung direkt zu dem Verbraucher, der ungefähr eine Spannung von 5 V benötigt. Vom Verbraucher führt eine weitere Leitung direkt zum Pin 5 mit der elektrischen Masse (Minuspole). Dies bedeutet lediglich, dass der Spannungsunterschied zwischen den beiden Schaltungen gleich bleiben soll. Deswegen ist immer ein Bezugspotential von 0 V mit eingefügt[19].

$$U = \Phi_2 - \Phi_1 = 5 \text{ V} - 0 \text{ V} = 5 \text{ V} \quad (2.4)$$

In dem Stromkreis hat die positive Leitung das Potential von 5 V und die negative Leitung das Potential von 0 V. Somit haben beide Schaltungen die gleiche Spannung.



### 2.5.4 Reihenschaltung und Parallelschaltung

In diesem Unterpunkt werden die Begriffe Reihenschaltung und Parallelschaltung eines Stromkreises anhand von angeschlossenen Stromquellen (Solarmodule) erläutert.

#### Reihenschaltung:

Der Aufbau einer Reihenschaltung beruht auf hintereinander verbundene Stromquellen (Solarmodule). Folgendes Bild soll den Aufbau erläutern.

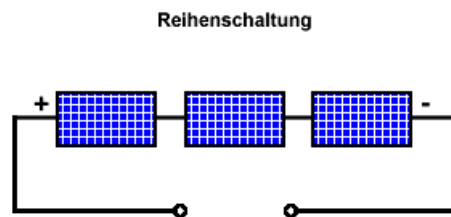


Abbildung 2.7: Reihenschaltung<sup>7</sup>

<sup>7</sup>Bildquelle: <https://www.solaranlagen-portal.de/solarenergie-komponenten/solarmodule.html#!>

Jedes Solarmodul besitzt zwei Kabel. Eine Leitung führt zum Pluspol und eine Leitung führt zum Minuspol. Wenn mehrere Solarmodule in Reihe geschaltet sind, müssen die einzelnen Kabel mit einander verbunden sein, sodass eine Reihe entsteht. Nun entstehen folgende Gesetze, die in diesem Stromkreis auftreten. Die einzelnen Spannungen von jedem Solarmodul werden aufaddiert und die Stromstärke innerhalb des Stromkreises bleibt gleich. Somit erreicht man eine Erhöhung der Gesamtspannung.

$$I = I_1 = I_2 = I_3 \dots \quad (2.5)$$

$$U = U_1 + U_2 + U_3 \dots \quad (2.6)$$

Vorteil einer solchen Schaltung sind die Kosten für die Installation sowie der Kabelaufwand. Der Nachteil ist der mögliche Ausfall einer Komponente und bedeutet dadurch ein Komplettausfall der Schaltung. Des Weiteren kann eine Verschattung eines Solarmodules die Leistung der Schaltung mindern. Denn in diesem Fall bestimmt die schwächste Komponente in der Reihe die Stromstärke. In diesem Fall benutzt man eine Parallelschaltung gegen solche Leistungsabfälle.[20]

## Parallelschaltung:

Die Parallelschaltung besitzt einen anderen Aufbau und hat zudem auch im Gegensatz zur Reihenschaltung andere Gesetze. Zweck der Parallelschaltung ist die Spannung für alle Komponenten gleich zu setzen und dafür die Stromstärke für jede einzelne Komponente im Schaltkreis aufzuaddieren. Die Gesamtstromstärke erhöht sich.

$$I = I_1 + I_2 + I_3 \dots \quad (2.7)$$

$$U = U_1 = U_2 = U_3 \dots \quad (2.8)$$

Die Verbindungen der einzelnen Komponenten führen über zwei Hauptkabel. An der Leitung, die zum Pluspol führt, sind alle Leitungen von den Solarmodulen, die auch zum Pluspol führen, verbunden. Genau das gleiche geschieht auch mit den Leitungen, die zum Minuspol führen.

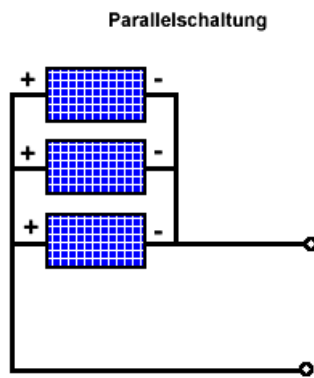


Abbildung 2.8: Parallelschaltung<sup>8</sup>

<sup>8</sup>Bildquelle: <https://www.solaranlagen-portal.de/solarenergie-komponenten/solarmodule.html#!>

Speziell wegen der Verschattung einzelner Module in der Schaltung, kann die Spannung ein wenig abfallen, doch fließt noch immer Strom durch die Schaltung. In diesem Fall ist die Gesamtstromstärke geringer. Eine Parallelschaltung kann unter Umständen auch ein wenig mehr kosten, da es mehr Kabel für die Verbindungen gibt.[20]  
 Letztendlich kommt es natürlich immer darauf an welche Anforderungen gestellt sind. Ein Verbraucher, der eine hohe Spannung voraussetzt, sollte eine Reihenschaltung verwenden. Eine Parallelschaltung kann zum Beispiel bei einem Akku von Vorteil sein, der mit einem hohen Ladestrom arbeitet und somit eine hohe Gesamtstromstärke voraussetzt.

## 3 Stand der Technik

Das Vorhaben eine alternative Energieversorgung mit Solarmodulen für Einplatinencomputer zu realisieren hört sich im ersten Moment nicht so schwierig an. Das kommt natürlich darauf an, welche Anforderungen an das System gestellt sind. In diesem Kapitel werden unter anderem auf die einzelnen Anforderungen des Systems eingegangen. Außerdem wird zu diesem Thema eine schon veröffentlichte Konzeption vorgestellt um einen Vergleich mit den vordefinierten Anforderungen zu bekommen. Zum Schluss werden anhand der Anforderungen die genutzten Technologien für das System erläutert.

### 3.1 Klassifizierung von den Anforderungen

Für ein ordentlich funktionierendes System müssen vorerst die Anforderungen stimmen, damit in erster Linie eine qualifizierte Auswahl von Komponenten stattfinden kann um dem System die besten Voraussetzungen für einen einwandfreien Betrieb zu garantieren. Die Anforderungen an das System spalten sich in mehrere Unterpunkte. An vorderster Stelle stehen allgemeine Forderungen für die Energieversorgung.

Anforderung der Energieversorgung:

- Die grundsätzliche Idee ist es, dass der Einplatinencomputer rund um die Uhr ausreichend Energie bekommt. Hauptsächlich natürlich mit Solarenergie. Der Schwachpunkt dieser Energiegewinnung sind die wechselnden Wetterverhältnisse und zudem noch die Nacht. Genau in dieser Zeitspanne kann die Solaranlage keine Energie über Nacht liefern. Aus diesem Grund ist es zwingend notwendig eine Alternative für den Nachtbetrieb mit einzuplanen. Eine Batterie beziehungsweise ein Akku könnte hier Abhilfe schaffen.
- Somit ist die Energieversorgung in Solarbetrieb und Akkubetrieb zu unterteilen.
- Im Bezug auf die Auswahl des Akkus ist es wichtig, dass dieser die nächtliche Energieversorgung bewerkstelligen kann. Ein Akku mit genügend Kapazität ist Voraussetzung. Die Kapazität sollte auch die Nächte im Winter überbrücken können.

Anforderungen an Funktionalitäten des Systems:

- Primär geht es in erster Linie um die Konzeption einer Energieversorgung via Solarmodule. Doch falls das System nicht ordnungsgemäß läuft, müssen Sicherheitsmaßnahmen eingreifen. Eine Sicherheitsmaßnahme wäre die sichere und automatische Abschaltung des Einplatinencomputers, falls die Energie der Solarmodule und die des Akkus nicht mehr ausreichen um den Einplatinencomputer mit Energie zu versorgen. Grund dafür sind zum Beispiel die vorherige Sicherung von Daten.
- Für die Überwachung der Energieversorgung soll eine Software die aktuellen analogen Daten erfassen und diese protokollieren.
- Diese Daten benutzt der Einplatinencomputer für die Abschaltung um genau zu wissen, wann der kritische Punkt für eine nicht ausreichende Energieversorgung erreicht ist.
- Darüber hinaus soll mit Hilfe eines Programms eine Benachrichtigung via E-Mail stattfinden, wenn der Einplatinencomputer herunterfährt.

## 3.2 Themenbezogene Veröffentlichungen

Für eine Konzeption einer Energieversorgung mit Solarenergie gibt es einige Beispiele im Internet zu sehen. In den folgenden Veröffentlichungen werden die Ideen und angewendeten Technologien beschrieben um darauffolgend einen Vergleich mit der Lösung dieser Arbeit zu bekommen.

### 3.2.1 Raspberry Pi mit Sonnenenergie betreiben

Es geht grundsätzlich um eine Energieversorgung via Solarmodule mit zusätzlichem Akku. Auch hier ist das Ziel einen rund um die Uhr Betrieb des Einplatinencomputers zu gewährleisten.[21]

Verwendete Technologie:

- Als Einplatinencomputer wurde hier ein Raspberry Pi genommen. Um welches Modell es sich handelt, ist im Text nicht ersichtlich.
- Solaranlage
- Akku
- Ein Solarladeregler sorgt für das ordnungsgemäße Laden des Akkus und schützt gleichzeitig vor Tiefentladung.
- Spannungsregler

Die eingesetzten Komponenten mit den technischen Details:



Abbildung 3.1: Komponenten des Systems<sup>7</sup>

---

<sup>7</sup>Bildquelle: <http://www.rustynailworkshop.com/archives/6>

Die erste Komponente ist das 50 W Solarmodul mit 12 V Spannung. Daraus resultiert sich eine Stromstärke von 4,16 A. Von dem Solarmodul führen Kabel direkt zum Solarladeregler. An den Solarladeregler ist der Akku angeschlossen. Der 12,8 V Akku bietet eine Leistung von 20 Ah und besitzt einen Lithium Kern. Bevor die Konstruktion an den Raspberry Pi angeschlossen werden kann, muss noch ein Spannungsregler dazwischen geschaltet werden. Dieser Spannungsregler reduziert die 12 V Spannung von dem Akku auf 5 V um dem Raspberry Pi die erforderliche Eingangsspannung zu geben.[21]

### 3.2.2 Spannungsmessung mit AD-Wandler am Raspberry Pi

Der Grund für die Konstruktion ist die Überwachung der Spannung für die Batterie. Die Konstruktion besteht aus einer Platine mit den dazugehörigen Komponenten. Um die Spannung zu messen, die aus der Batterie fließt, führt ein Kabel von der Batterie zum AD-Wandler, welcher sich auf der Platine befindet. Um die analogen Daten (Spannungswerte) von der Batterie zu lesen, wandelt der AD-Wandler diese in digitale Signale um. Für dieses Vorgehen wird ein AD-Wandler benutzt.[22]. Anschließend führen vom AD-Wandler einige Verbindungskabel zum Raspberry Pi, die an die GPIO-Schnittstelle angeschlossen sind. Vom Raspberry Pi liest ein Skript die entsprechenden GPIO-Pins um die digitalen Signale auszuwerten.

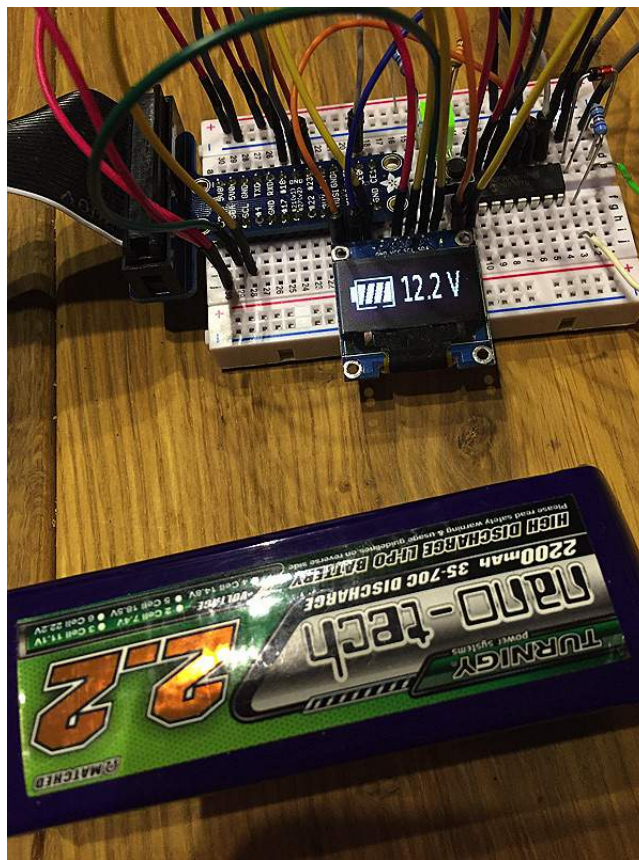


Abbildung 3.2: Sensorik für die Energieüberwachung<sup>8</sup>

<sup>8</sup>Bildquelle: <https://www.direcs.de/2017/03/spannungsmessung-mit-ad-wandler-am-raspberry-pi/>

Auflistung der einzelnen Komponenten für die Energieüberwachung:

- AD-Wandler (MCP3008)
- Verbindungskabel
- Li-Po-Akku (Lithium-Polymer-Akkumulator) 12,3 Volt
- Widerstand, Z-Diode (Zener-Diode)

Für den Aufbau muss man einige Punkte beachten. Der AD-Wandler (MCP3008) hat eine Versorgungsspannung von 2,7 V Minimum und 5,5 V Maximum[23],[22]. Das bedeutet, dass der AD-Wandler in diesem Spannungsbereich operiert. Die Spannung von dem Li-Po-Akku mit den 12,3 V ist daher zu hoch um den Akku direkt mit dem AD-Wandler zu verbinden. Für eine Absenkung der Spannung kann an dieser Stelle ein Widerstand in Verbindung mit einer Z-Diode hilfreich sein[22].

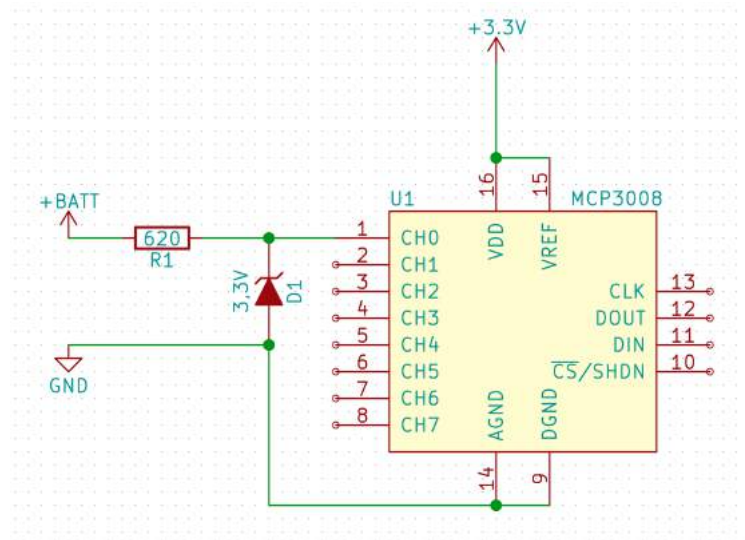


Abbildung 3.3: Schaltbild für die Energieüberwachung<sup>9</sup>

<sup>9</sup>Bildquelle: <https://www.direcs.de/2017/03/spannungsmessung-mit-ad-wandler-am-raspberry-pi/>

Der Widerstand soll die Spannung auf einen bestimmten Wert zwischen 2,7 V und 5,5 V begrenzen. Hierfür wurde der Widerstandswert auf 620 Ohm gewählt um die Stromstärke zu begrenzen.

$$I = \frac{12,3 \text{ V}}{620 \Omega} = 0,01983 \text{ A} = 19,83 \text{ mA} \quad (3.1)$$

Die Z-Diode dient in der Schaltung, beziehungsweise für den AD-Wandler (MCP3008) als Spannungsstabilisierung. Das ist notwendig, da die Spannung für den AD-Wandler an den analogen Eingängen sich nicht über 3,3 V erhöhen darf. Die Spannung, die an den AD-Wandler geht, erreicht die Z-Diode mit 3,3 V. Diese Z-Diode verträgt bis zu 500 Milliwatt[22].

$$P = 0,01983 \text{ A} \cdot 12,3 \text{ V} = 0,243 \text{ W} = 243,90 \text{ mW} \quad (3.2)$$

Die 243,90 Milliwatt Leistung können somit getrost durch die Z-Diode fließen. Damit man die Daten ausgelesen kann, muss an dieser Stelle ein Programm auf dem Einplatinencomputer eingreifen. Das Programm ist in der Programmiersprache Python geschrieben. Für das Auslesen der Daten benutzt das Programm die Python Bibliothek py-spidev, die über den SPI-Bus die Anweisungen angibt.[22].

Das Programm für die Ausgabe der Daten sieht wie folgt aus:

Programm 3.1: Auslesen der Daten vom MCP3008

```
1 #!/usr/bin/python
2 # coding=utf-8
3
4 ##### AD converter stuff
5 from MCP3008 import MCP3008
6 adc = MCP3008()
7 voltage = 0
8 value = 0
9
10 # read AD converter (battery voltage)
11 # use channel 0 on IC
12 value = adc.read(channel = 0)
13 # 2.73 V = 12.32 V (measured) > 1023 / 3.3 * 2.73 / 12.32 = 68.693182
14 voltage = (value / 68.693182)
15 print("Voltage:␣%.1f" % voltage)
```

In Zeile 12 bekommt die Variable value den Wert aus der Funktion read. Der Übergabeparameter ist der Kanal 0, da in der Abbildung 3.3 die Leitung von dem Akku direkt an den Pin 1 (CH0) angeschlossen ist. Die analogen Signale wandern in den Kanal 0 und der AD-Wandler wandelt diese in digitale Signale um.



### 3.3 Vergleich mit den Anforderungen

Die themenbezogenen Veröffentlichungen decken zum größten Teil die im Punkt 3.1 Klassifizierung von den Anforderungen ab. Die Veröffentlichung, im Punkt 3.2.1 Raspberry Pi mit Sonnenenergie betreiben, kann die eigentliche Konstellation der Energieversorgung vorgeben. Das Zusammenspiel aus Solarmodul, Solarladeregler und Batterie ist somit gut geeignet um eine allgemeine Energieversorgung des Einplatinencomputers zu gewährleisten. Allgemein kann man hier nur von der Architektur sprechen. Speziell für einen 24 Stunden Betrieb, ist es notwendig die Komponenten an die Ansprüche des Systems anzupassen. An erster Stelle fängt dies schon mit der Auswahl der Solarmodule an. Wobei die Solarmodule im eigenen Projekt schon vorhanden sind und somit in diesem Bereich eine eigene individuelle Lösung voraussetzen. Für die Auswahl des Akkus ist es deswegen von großer Wichtigkeit, dass die Leistungswerte der vorhandenen Solarmodule schon ausgewertet sind um eine geeignete Auswahl für den Akku zu finden. Die Anforderungen des Solarladereglers sind bei der Veröffentlichung sowie bei den eigenen Anforderungen für die Tiefentladung und Überladeschutz identisch. Im Großen und Ganzen kann man somit erstmal die eigentliche Architektur der Veröffentlichung benutzen. Für eine geeignete Komponentenauswahl müssen verschiedene Tests der Teile des Systems erst durchgeführt werden.

Die zweite Veröffentlichung im Punkt 3.2.2 Spannungsmessung mit AD-Wandler am Raspberry Pi deckt alle Anforderungen im Bezug auf die Messung der Spannungsversorgung durch die Batterie ab. Die Architektur und die darin enthaltenen Komponenten kann man zum größten Teil benutzen. Der Aufbau besteht aus den Verbindungen, wie in der Abbildung 3.3 des Schaltbildes. Die Anforderung für die Überwachung der Energieversorgung des Einplatinencomputers erledigt die vorgegebene Software. Das automatische Abschalten des Einplatinencomputers bei nicht ausreichender Energie, übernimmt das Programm.

Alles in allem ist eine Verwendung der Lösungen der Veröffentlichungen für die meisten Anforderungen optimal nutzbar. Das einzige Problem ist die Realisierung der vorhandenen Solarmodule für eine ausreichende Energieversorgung im Verbund mit einem geeigneten Akku. In diesem Fall ist es notwendig eine individuelle Lösung zu erarbeiten. Diese Lösung muss daraufhin eingehend getestet werden.

### 3.4 Eingesetzte Technologie

Die eingesetzten Technologien beinhalten die Bereiche Hardware, Software und das dazu gehörige individuelle, drehbare Gestell, welches die Solarmodule trägt. Als erstes werden die Hardwarekomponenten vorgestellt und danach der Softwareanteil.

#### Hardwarekomponenten:

- Akku:

Der 12 V Akku besitzt eine Amperestunden Anzahl von 6,5 Ah. Der Grund für die Auswahl ist, dass der Akku über Nacht den Einplatinencomputer ausreichend mit Strom versorgt. Theoretisch verbraucht der Raspberry Pi ungefähr 0,265 A. Bei einem Wert von 6,5 Ah wäre der Akku nach ungefähr 24,52 Stunden Betriebsdauer leer.

$$\frac{6,5 \text{ Ah}}{0,265 \text{ A}} = 24,52 \text{ h} \quad (3.3)$$

Diese Zeit ist natürlich nur rein theoretisch zu sehen, da während des Betriebes unterschiedliche Leistungen anfallen und die Betriebsdauer des Akkus dadurch eingeschränkt ist. Zusätzlich kann der Akku nicht komplett entladen werden, da der Solarladeregler eine Tiefentladung des Akkus verhindert. Zur Anwendung kommt ein Blei-Akku, da die Entladekurve dieses Typs nicht abrupt fällt, wie es bei Lithium-Akkus der Fall ist, sondern besitzt eine leicht abfallende Entladekurve, welche für die Notabschaltung des Raspberry Pis von Vorteil ist.



Abbildung 3.4: 12 Volt Akku

- Einplatinencomputer:



Abbildung 3.5: Einplatinencomputer Raspberry Pi 3 Model B

- Solarmodule:

Es gibt zwei unterschiedliche Solarmodule zur Auswahl. Auf dem ersten Blick erscheinen diese jedoch völlig gleich. Bei genauerem Hinschauen kann man die einzelnen senkrecht verlaufenden Zellen auf dem Solarmodul erkennen. Das obere Solarmodul besitzt weniger senkrecht verlaufende Zellen, als das untere Solarmodul, dafür jedoch breitere. Von dem oberen Solarmodul sind insgesamt 11 Stück vorhanden und vom unteren sind 17 Stück vorhanden. Aus dieser Auswahl muss man dann nur noch eine geeignete Zusammenstellung treffen.



Abbildung 3.6: Solarmodule

- Solarladeregler:

Der Solarladeregler regelt die Stromzufuhr von der Energiequelle (Solaranlage) zum Akku. Dabei sind folgende Eigenschaften wichtig. Die Nennspannung des Solarladereglers liegt bei 12 / 24 V. Des Weiteren sind Lastströme bis zu 10 A möglich. Weitere wichtige Funktionen ist der Tiefentladeschutz und der Überladeschutz.



Abbildung 3.7: Solarladeregler

Auf dem Solarladeregler gibt es LED-Anzeigen. Auf der Abbildung 3.7 sind es genau vier LEDs. Von links gesehen gibt es einmal die Info-LED danach die rote Batterie-LED, gelbe Batterie-LED und die grüne Batterie-LED. Diese LEDs haben folgende Funktionen[24]:

- Info-LED:
  - Leuchtet grün: Normalbetrieb
  - Blinkt langsam rot: Systemfehler
- Rote Batterie-LED:
  - Blinkt schnell: Batterie leer, Warnung vor Abschaltung bei Unterspannung
  - Blinkt langsam: Tiefentladeschutz aktiv, Verbraucher abgeschaltet
- Gelbe Batterie-LED:
  - Leuchtet: Batterie schwach, Verbraucher eingeschaltet
  - Blinkt langsam: Wiedereinschaltsschwelle nach der Tiefentladung noch nicht wieder erreicht, Verbraucher noch abgeschaltet
- Grüne Batterie-LED:
  - Leuchtet: Batterie geladen
  - Blinkt schnell: Batterie voll, Laderegulierung aktiv

- Spannungsregler:

Der Spannungsregler (DC-DC-Wandler) reduziert die Spannung mit Hilfe eines Potis (Potentiometer, blaues Gehäuse mit Stellschraube), der sich auf der Platine befindet. Wichtig ist hierbei, dass die ausgehende Spannung aus dem Solarladeregler auf ungefähr 5,1 V Betriebsspannung eingestellt wird um dem Raspberry Pi die geeignete Spannung zu liefern.

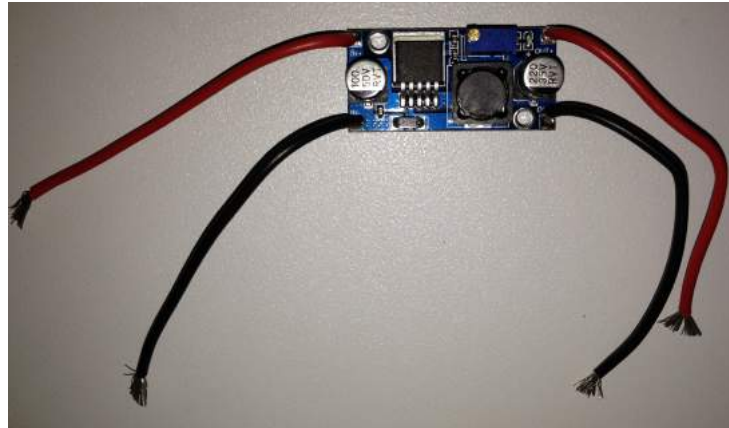


Abbildung 3.8: DC-DC-Wandler (Spannungsregler)

- Komponenten zur Spannungsmessung:

Die Komponenten, wie Verkabelung, Mikrocontroller sowie Z-Diode sind wie in der schon zuvor beschriebenen Veröffentlichung, Spannungsmessung mit AD-Wandler am Raspberry Pi, im Grunde genommen identisch. Der einzige Unterschied liegt in der Verwendung eines Potentiometers (blauer Kasten mit Stellschraube) im Gegensatz zu einem festgelegten Widerstand. Vorteil des Potentiometers ist die Funktion den Widerstand dynamisch mit Hilfe der Stellschraube an die jeweiligen Bedürfnisse anpassen zu können.

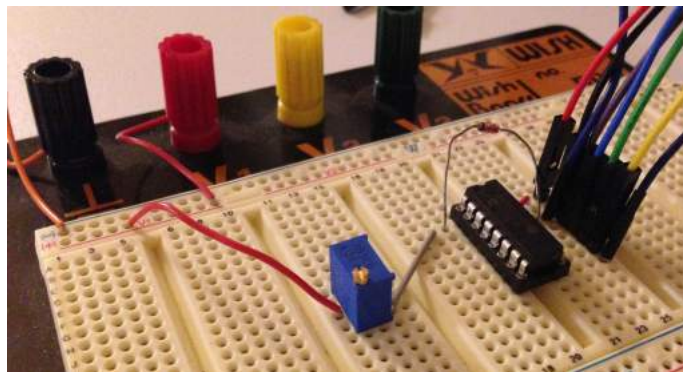


Abbildung 3.9: AD-Wandler mit Komponenten

- Schrittmotor:

Eine wichtige Komponente ist der Schrittmotor, der für die individuelle Lösung der Solaranlage gebraucht wird. Die Betriebsspannung beträgt 5 V. Der Vorteil ist die Möglichkeit den Schrittmotor direkt über den Raspberry Pi mit Strom zu versorgen. Zum Schrittmotor gehört zusätzlich eine Treiberplatine, die die Befehle von Raspberry Pi entgegen nimmt um daraufhin Signale an den Motor zu übermitteln. Es gibt zwei dieser Motoren.



Abbildung 3.10: Schrittmotor

Der Schrittmotor besitzt im Gehäuse vier Magnete und einen Magnetzylinder in der Mitte, die für die Drehungen verantwortlich sind. Insgesamt kann man in maximal acht Halbschritten die Magnete des Motors antreiben.[25]



- Drehgestell:

Das Drehgestell besteht aus zwei Hauptkomponenten. Das Gestell und der Standfuß mit Kugellager für die Drehfunktion. Das Gestell besteht aus einem Rahmen für die Solarmodule, einem zweiten Rahmen mit Stützen für den Rahmen der Solarmodule und einer Bodenplatte.



Abbildung 3.11: Gestell mit Standfuß

Das Gestell ist komplett aus Aluminium Profilen angefertigt. Diese sind im Baumarkt erhältlich. Der Standfuß ist eingekauft und ist zusätzlich an die Anforderungen des Systems modifiziert.

## Eingesetzte Software:

- Unix-Shell:

Die Unix-Shell oder auch kurz Shell genannt, ist eine Schnittstelle, die es erlaubt, Kommandos über ein Terminal einzugeben, um dem Linux basierten Betriebssystem Anweisungen zu übermitteln. Die Shell gehört zu den Skriptsprachen, die man für Programmierung und automatisierte Aufgaben verwendet.[26]

Für die Belange der Anwendung ist die Funktion der Shell für die Aufgaben der automatischen Abschaltung des Systems sowie das automatische Aufrufen der Skripte für die Energieversorgung wichtig. Dabei kommen folgende Techniken zum Einsatz:

Der Befehl für die automatische Ausführung des Energieüberwachungsskriptes ist in der rc.local Datei gespeichert. Diese ist für die automatische Ausführung von Skripten, Programmen oder Diensten beim Systemstart verantwortlich[27].

Des Weiteren benutzt das System das Zusatzprogramm cron. Dieses ist über die Terminaloberfläche konfigurierbar. Die Aufgabe von cron besteht darin, Programme oder auch Skripte zu einer bestimmten Zeit auszuführen[28].

- Python:

Python besitzt mehrere Programmierparadigmen, wie zum Beispiel objektorientierte, aspektorientierte sowie die funktionale Programmierung. Somit ist Python auch durch die dynamische Typisierung eine Skriptsprache. Vorteile bietet die Sprache hinsichtlich der einfachen Syntax für einfaches Lesen und Schreiben des Codes. Zudem besitzt Python eine umfangreiche Standardbibliothek.[29]

Man könnte auch sagen, dass Python die Standard Programmiersprache für den Raspberry Pi ist. Auf den aktuellsten Betriebssystemen sind alle wichtigen Standardbibliotheken von Python und Entwicklerwerkzeuge vorinstalliert.



## 4 Vorbereitung und Installation

In diesem Kapitel werden alle nötigen Schritte erklärt, die für die Inbetriebnahme des Raspberry Pi wichtig und vonnöten sind.

### 4.1 Installation Betriebssystem Raspbian

Die Installation des Betriebssystems auf dem Raspberry Pi ist im Grunde genommen der erste Schritt um mit dem Raspberry Pi interagieren zu können. Es gibt viele Betriebssysteme zur Auswahl. Zur Anwendung kommt das offizielle Betriebssystem Raspbian, welches auf der Raspberry Pi Internetseite zur Verfügung steht[30]. Das heruntergeladene Image kann man daraufhin mit dem Programm Win32DiskImager auf eine SD-Karte schreiben.

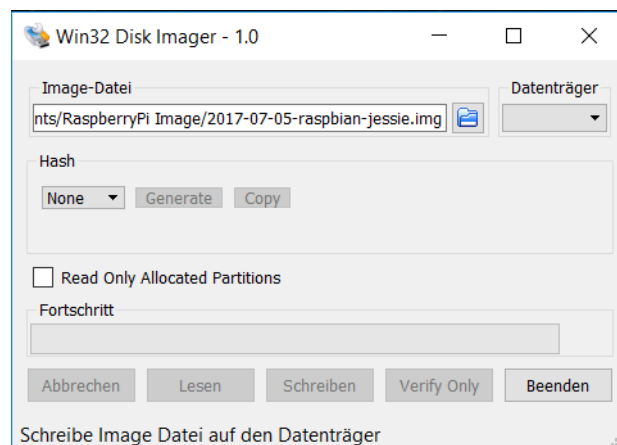


Abbildung 4.1: Programm für das Schreiben des Images auf MicroSD-Karte

Auf der graphischen Oberfläche des Programms wählt man als erstes im Bereich der Image-Datei das entpackte Image aus. Das Datenträger Feld dient zur Auswahl der zu benutzten MicroSD-Karte. Als nächstes ist die Schaltfläche Schreiben zu betätigen um das Image auf den ausgewählten Datenträger zu schreiben. Anschließend erscheint eine Statusmeldung über den abgeschlossenen Vorgang. Hiermit ist das Betriebssystem auf der MicroSD-Karte installiert und kann in den, auf der Rückseite liegenden Schacht des Raspberry Pis eingefügt werden.

## 4.2 Verbindungsmöglichkeiten mit dem Raspberry Pi ohne Peripherie

Es gibt zahlreiche Möglichkeiten auf den Raspberry Pi zu zugreifen. Was aber wenn der Einplatinencomputer in einer unwegsamen Art und Weise verbaut worden ist und eine Kommunikation via Peripherie Geräten ausgeschlossen ist? Deswegen werden hier zwei von diesen Möglichkeiten, die ohne angeschlossene Peripherie Geräte auskommen, nun im folgenden näher erläutert.

### 4.2.1 Raspberry Pi über LAN oder WLAN

Um eine Verbindung mit dem Raspberry Pi herstellen zu können, muss als erstes eine Verbindung via LAN erfolgen, da die Verbindung via WLAN noch deaktiviert ist. Für die LAN-Verbindung wird ein Patch-Kabel vom Ethernet-Anschluss des Raspberry Pis an den Router angeschlossen. Die Vergabe der IP-Adresse funktioniert hier automatisch, da normalerweise auf jedem Router ein DHCP-Server installiert ist und jedem Gerät eine IP-Adresse vergibt[31]. Wenn die Remoteverbindung steht, kann man daraufhin die WLAN-Verbindung aktivieren. Dabei ist auf eine Kleinigkeit zu achten. Installiert man die Light-Version von Raspbian, ist eine manuelle Konfiguration des WLAN-Anschlusses vonnöten[32]. In diesem Fall wurde die Vollversion des Raspbian Betriebssystems verwendet und somit kann die WLAN-Verbindung wie bei herkömmlichen Benutzeroberflächen eingestellt werden.

### 4.2.2 Raspberry Pi mit PuTTY

PuTTY ist eine Software um eine Client Server basierte Verbindung aufbauen zu können[33]. Der Client wäre in diesem Fall der Host, der das Programm bei sich auf dem Rechner ausführt. Der Server wäre der Raspberry Pi um auf diesen zugreifen zu können. Die Verbindung in diesem Beispiel basiert auf der Secure Shell, kurz SSH, Technologie. SSH dient hierbei als Netzwerkprotokoll und hat die Aufgabe über eine gesicherte beziehungsweise verschlüsselte Verbindung die Möglichkeit zu eröffnen, den Einplatinencomputer fernzusteuern[34]. Gerade am Anfang, wenn das Betriebssystem auf die MicroSD-Karte neu installiert ist, muss man eine Verbindung herstellen um weitere Einstellungen vornehmen zu können. Vorerst bindet man nach der Installation des Betriebssystems auf der MicroSD-Karte eine zusätzliche Datei ein. Denn die SSH-Funktion ist allgemein ausgeschaltet. Diese Änderung ist seit dem im November 2016 herausgekommenen Update vorhanden[35].

Auf der MicroSD-Karte navigiert man in den Ordner boot. Dort ist eine leere Datei mit der Bezeichnung ssh und ohne Dateiendung zu erstellen. Somit ist der SSH-Zugriff aktiviert.

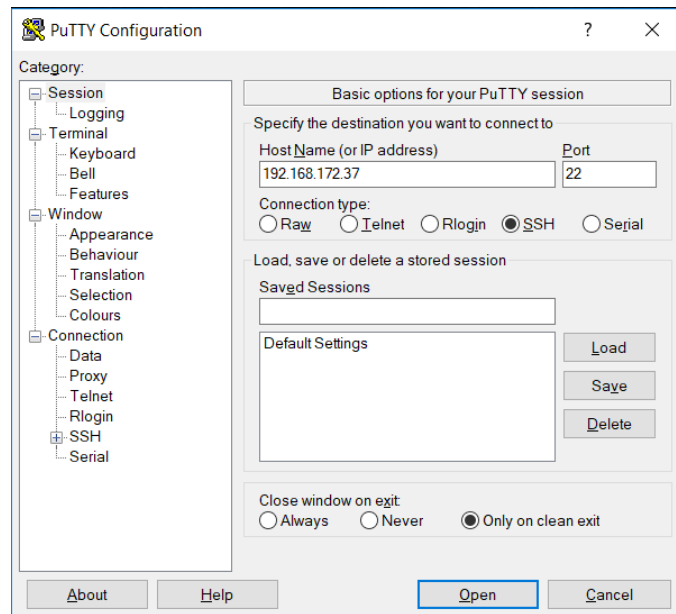


Abbildung 4.2: SSH Verbindung mit PuTTY

Das wichtigste Feld zum Ausfüllen ist HostName(or IP address). Dort wird entweder die IP-Adresse oder der Hostname angegeben. Da die IP-Adresse am Anfang von dem Einplatinencomputer nicht bekannt ist, gibt es die Möglichkeit die IP-Adresse über den Router in Erfahrung zu bringen[35]. Alle anderen Felder bleiben so wie voreingestellt und per Klick auf die Schaltfläche open ist eine Verbindung aufgebaut.

### 4.2.3 Raspberry Pi mit Remotedesktopverbindung

Falls die Terminal Ansicht für den Benutzer nicht ausreichend ist, kann man zum Beispiel über die Remotedesktopverbindung, die es schon auf Windows 10 gibt, eine Verbindung aufbauen. Diese bietet mehr Funktionen und zusätzliche Einstellungen rund um den Aufbau und Wiedergabe der Remoteverbindung.

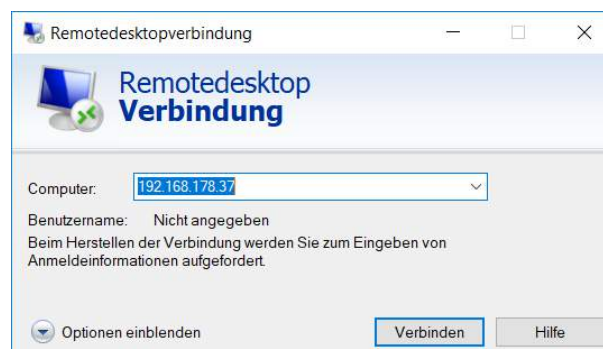


Abbildung 4.3: Remotedesktopverbindung

Auch hier ist mit der Eingabe der IP-Adresse vom Raspberry Pi die Verbindung auf den Raspberry Pi möglich. Für die Freigabe der Remoteverbindung sind folgende Befehle im PuTTY Terminal notwendig:

- `sudo apt-get install tightvncserver`
- `sudo apt-get install xrdp`

Danach sollte man die Remoteverbindung starten können.

### 4.3 Konfiguration Betriebssystem Raspbian

Um alle notwendigen Komponenten auf dem Raspberry Pi zu benutzen, ändert man einige Einstellungen. Die Einstellungen sind von der Kommandozeile mittels PuTTY und von der Benutzeroberfläche per Remotedesktopverbindung zugänglich. Das folgende Bild zeigt die Einstellungen über die Remotedesktopverbindung.

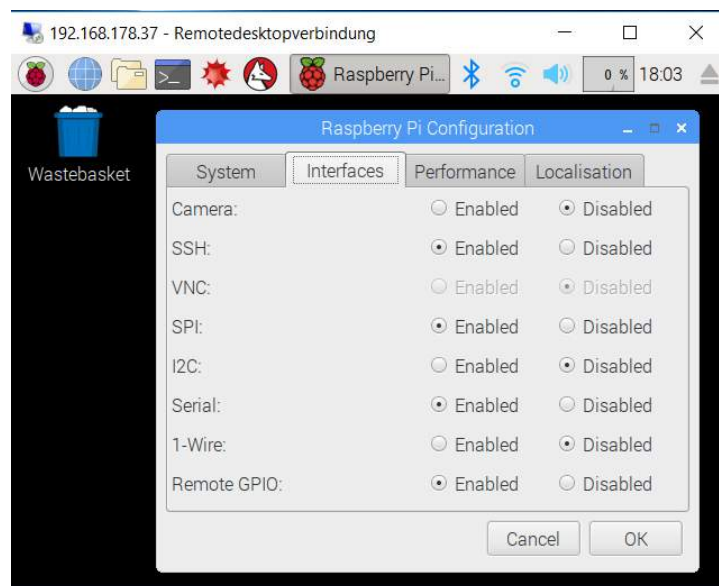


Abbildung 4.4: Einstellungen

Um die Einstellungen zu öffnen, wählt man folgenden Pfad aus. Schaltfläche(Himbeere) -> Preferences -> Raspberry Pi Configuration  
Standardmäßig sind einige der aufgelisteten Punkte deaktiviert. Folgende Punkte sind wichtig um einige Komponenten in dem Projekt benutzen zu können:

- Die SSH-Verbindung ist für die Remote-Verbindung nützlich.
- SPI-Bus (Serial Peripheral Interface) dient für die Verbindung zum AD-Wandler für die Spannungsüberwachung.

Für die aktuelle Software sind auf dem Raspberry Pi folgende Befehle wichtig[36]:

- `sudo apt-get update`
- `sudo apt-get dist-upgrade`

Jetzt sind alle Vorkehrungen getroffen, um den Raspberry Pi für die weiteren Angelegenheiten zu benutzen.

## 5 Zusammenführung der Komponenten

Es gilt jetzt die zuvor beschriebenen Komponenten so anzuwenden, dass ein voll funktionsfähiges System daraus entsteht, welches die Anforderungen erfüllt. Zu aller erst wird der Aufbau des kompletten Systems gezeigt und dazu noch die Beweggründe des Designs. Danach werden die Implementierungen der jeweiligen Komponenten im Detail beschrieben.

### 5.1 Design des Gesamtkonzeptes

Das Design des Systems soll zeigen, wie die einzelnen Komponenten miteinander agieren. Der Punkt 5.2 System Steckplatine dient hierbei als Veranschaulichung für das Design beziehungsweise für die Architektur des Systems. An diesem Bild werden alle eingesetzten Komponenten erklärt. Der Punkt 5.3 System Schaltplan zeigt hingegen die einzelnen Verbindung aus elektronischer Sicht. Wichtig ist hierbei, dass diese Bilder nur eine Übersicht darstellen. Für einen detaillierten Einblick ist der Punkt 5.4 Details zur Implementierung vorgesehen, wo einzelne Abschnitte dieser Bilder erklärt werden.

Die Zusammenführung der Komponenten ergibt schließlich folgenden Aufbau:

Die Solarmodule produzieren tagsüber die Energie. Der Solarladeregler nimmt diese entgegen und lädt mit dieser gewonnenen Energie den Akku auf. An den Solarladeregler ist der Verbraucher (Raspberry Pi 3 Model B) angeschlossen. Zwischen dem Raspberry Pi und dem Solarladeregler ist ein Spannungsregler (DC-DC-Wandler) zwischengeschaltet. An den Spannungsteiler ist ein Netzstecker (Micro-USB) angeschlossen, der an den Raspberry Pi angeschlossen ist. Der Raspberry Pi versorgt die beiden Treiberplatinen der Schrittmotoren mit Energie über die GPIO-Leiste. Zudem führen von der GPIO-Leiste Verbindungskabel zu den Treiberplatinen um den Schrittmotoren die nötigen Befehle zu übermitteln. Des Weiteren führen einige Verbindungskabel von der GPIO-Leiste des Raspberry Pis zu der Steckplatine, auf der der AD-Wandler sitzt. Auch der Raspberry Pi versorgt den AD-Wandler mit Energie. Damit der AD-Wandler die nötigen Daten für die Spannungsüberwachung bekommt, führt ein Kabel vom Pluspol des Akkus zum AD-Wandler.

Die Schrittmotoren sind am drehbaren Gestell fixiert und übernehmen dort die Aufgaben bestimmte Winkel für die Drehachsen (Gierachse und Nickachse) einzustellen. Ein zusätzliches Programm, der Sonnenverlauf Algorithmus, berechnet den Winkel des Sonnenverlaufs. Dieser Aufbau zeigt jetzt nur den Ablauf beziehungsweise die Funktion des Systems. Als nächstes werden ein paar Komponenten vorgestellt, die eine detailliertere Beschreibung zeigen.

- Solarmodule:

Die Solarmodule sind parallel geschaltet. Grund für diesen Aufbau ist die Auswahl der vorhandenen Solarmodule. Wie bereits im Punkt Stand der Technik erwähnt, gibt es zwei Typen dieser Solarmodule. Diese müssen zunächst einmal auf die Spannungswerte überprüft werden. Im Folgenden wird eine Tabelle gezeigt, die die Leerlaufspannung der einzelnen Solarmodule wiedergibt. Für die Typzuweisung gilt: Typ 1 steht für die Solarmodule mit den breiteren Zellen und Typ 2 steht für die Solarmodule mit den dünneren Zellen.

Typ 1	Typ 2
20,7 V - 21,5 V	11,7 V - 12,1 V
22,1 V - 22,8 V	18,42 V - 18,54 V
22,0 V - 24,3 V	20,6 V - 20,8 V
23,8 V - 24,0 V	15,48 V - 15,52 V
23,1 V - 23,4 V	8,72 V - 8,80 V
23,4 V - 23,6 V	9,45 V - 9,55 V
22,8 V - 23,2 V	9,89 V - 9,92 V
23,4 V - 23,7 V	16,41 V - 16,51 V
23,3 V - 23,5 V	12,79 V - 12,83 V
23,8 V - 23,9 V	17,75 V - 17,84 V
22,3 V - 23,3 V	13,83 V - 13,89 V
	17,49 V - 17,51 V
	16,19 V - 16,22 V
	14,97 V - 15,0 V
	16,70 V - 16,76 V
	8,81 V - 10,20 V
	16,04 V - 16,35 V

Tabelle 5.1: Leerlaufspannung der Solarmodultypen

Die Leerlaufspannung tritt dann auf, wenn der Stromkreis offen ist und kein Verbraucher angeschlossen ist. Die Messungen der Leerlaufspannung zeigen insofern erst einmal nur an, ob die Solarmodule überhaupt funktionieren. Diese Spannungswerte werden im realen Fall niedriger sein, wenn ein Verbraucher an den Solarmodulen angeschlossen ist. In diesem Fall spricht man von der Klemmenspannung[37]. Der Grund für die Parallelschaltung ist zum einen den Ladestrom für den Akku zu erhöhen. Eine Reihenschaltung würde den Spannungswert aufaddieren und dabei den Ladestrom nicht erhöhen. Bei einer Parallelschaltung bleibt der Spannungswert gleich. Ein Blick auf die Tabelle 5.1 zeigt, dass die Werte des Typs 1 genau dieser Vorgabe entsprechen. Weiterhin ist auch wichtig, dass die Spannungswerte jedes einzelnen Moduls sich voneinander nicht groß unterscheiden.

Beispielsweise sollte man ein 8 V Modul mit einem 24 V Modul nicht parallel schalten, da der Spannungsunterschied aus diesen unterschiedlichen Spannungswerten das Solarmodul mit einem Rückstrom beschädigen kann. Das kann auch im Falle einer Verschattung passieren, indem ein Solarmodul in der Schaltung durch einen Schatten eine niedrigere Leistung erbringt. Die Spannungs Differenz beträgt hier 16 V. Dabei würde der Ausgleichsstrom immer von der höheren Spannungsquelle zur niedrigeren Spannungsquelle fließen.[38]

Aus diesem Grund ist die parallele Schaltung der Solarmodule des Typs 1 für den Gebrauch im System geeignet. Von diesen Solarmodulen kommen zehn Stück zum Einsatz. Zudem wird die Stromstärke hier aufaddiert und somit erzielt die Schaltung einen höheren Ladestrom. Das sollte reichen, um genug Energie für das System zu liefern.

- Solarladeregler und Akku:

Wichtige Funktionen des Solarladereglers ist der Überladeschutz und der Tiefentladeschutz. Beide Funktionen dienen zur längeren Haltbarkeit des Akkus. Der Akku ist ein 12 V Bleiakku, der unter anderem vom Hersteller empfohlen wird[39].

- Schrittmotoren:

Die Schrittmotoren dienen einmal für die Nickachse und die Gierachse im Gestell für die Solarmodule. Die Nickachse stellt die Funktion des Neigungswinkels der Solarmodule dar. Die Gierachse ist für das drehen des Drehgestells zuständig. Beide Schrittmotoren benötigen eine Spannungsversorgung von 5 V. Der Raspberry Pi besitzt auf seiner GPIO-Leiste zwei dieser 5 V Pins und ist somit auch geeignet für die Energieversorgung der Schrittmotoren. Wenn man die Schrittmotoren über den Raspberry Pi nicht versorgen möchte, kann man auch externe Spannungsquellen wie Batterien anschließen. Der Nachteil ist dabei die unkontrollierte Überwachung dieser Batterien. Die Spannungsversorgung der Schrittmotoren sollte immer gewährleistet sein. Die Schrittmotoren eignen sich im Grunde genommen perfekt für das System, da die Drehwinkel für die jeweiligen Schritte durch den Sonnenverlauf Algorithmus bekannt sind.

- Drehgestell:

Das Gestell für die Solarmodule ist eigenhändig angefertigt und besitzt daher nur für die vorhandenen Solarmodule die zuständigen Maße. Das Gestell soll die zehn parallel geschalteten Solarmodule tragen. Da es aufwendig ist, jedes einzelne Solarmodule zu bewegen, hilft ein Rahmen, der die zehn Solarmodule fest umschließt, damit diese nicht aus dem Rahmen fallen können. Das Gestell besteht aus dem angefertigten Rahmen, Stützen mit einem zweiten Rahmen sowie einer Bodenplatte, die den Rahmen samt Solarmodule halten. Unter dieser Konstruktion befindet sich eine weitere Komponente, die das ganze hält und mit einem integrierten Riemenantrieb dem ganzen die Drehfunktion verleiht. Des Weiteren hilft ein Seilzug auf der Hinterseite der Solarmodule, der den Neigungswinkel der Solarmodule einstellt. Hier kommen auch die Schrittmotoren ins Spiel. Schrittmotor (Gierachse) treibt den Riemenantrieb auf der Unterseite des Drehgestells an. Der Schrittmotor (Nickachse) ist für den Seilzug zuständig. Das



Gestell besteht aus Aluminium. Aluminium ist ein Leichtmetall und besitzt trotz des geringen Gewichts eine stabile Eigenschaft. Weiterhin kann man dieses Metall gut bearbeiten (schweißen, sägen, verschrauben). Das Gewicht spielt hier eine große Rolle, denn der Schrittmotor (Gierachse) muss das gesamte Gewicht des Gestells plus Solarmodule bewegen können. Der Riemenantrieb unterstützt hierbei. Zwischen den Stützen und dem Rahmen der Solarmodule befinden sich sogenannte Zapfenbänder, die den Rahmen samt Solarmodule die Drehung nach oben und unten ermöglichen. Diese wurden speziell dafür angefertigt. Es gibt viele verschiedene Möglichkeiten die Drehung der Nickachse zu konstruieren. Die einfachste und schnellste Möglichkeit erweist sich mit dem Seilzug. Die Winde ist am Schrittmotor befestigt und besitzt zwei Seile. Ein Seil ist am oberen Rand des Rahmens befestigt und das andere Seil hingegen am unteren Rand des Rahmens. Das untere Seil ist schon in der Winde aufgewickelt. Die Funktion besteht darin, dass die Winde das obere Seil einzieht (bei Vergrößerung des Neigungswinkels) und das untere Seil herausgibt. Der Grund für die zwei Seile liegt hauptsächlich an der etwas schwerfälligen Beweglichkeit der Zapfenbänder, durch das hohe Gewicht der zehn Solarmodule. Deswegen muss das untere Seil bei Verringerung des Neigungswinkels den Rahmen wieder auf seine ursprüngliche Position zurückholen.

- Sonnenverlauf Algorithmus:

Dieser Algorithmus gibt der Funktion des Drehgestells erst seine wahre Bedeutung. Die Aufgaben des Algorithmus beziehen sich auf die Ermittlung der Urzeiten des Sonnenaufgangs und Sonnenuntergangs via API von einer Internetseite. Der Dienst cron führt diese Funktion immer vor dem Sonnenaufgang aus, um die erforderlichen Daten für diesen Tag zur Verfügung zu stellen. Aus diesen beiden Werten, plus den Koordinaten des Standorts der Solaranlage, berechnet der Algorithmus den Winkel (Azimut) des Sonnenverlaufs für den ausgewählten Tag. Der Algorithmus ruft dann die Programme für die Schrittmotoren auf. Das Programm des Schrittmotors (Gierachse) benutzt daraufhin diesen Winkel um für die Zeitdauer des Sonnenverlaufs, das Drehgestell, mit dem Verlauf der Sonne mit drehen zu lassen. Für die Berechnung des Winkels gibt es eine Formel. Es ist also das Ziel ein dynamisches Vorgehen im Bezug auf die Drehfunktion zu erreichen. Im Falle des Neigungswinkels soll keine dynamische Funktion Abhilfe schaffen. Die Winkel für die Höhe der Sonne passt das System statisch an. Insgesamt gibt es vier Anpassungen für den Neigungswinkel während des Sonnenverlaufs. Da die Sonne anfangs schnell steigt, macht es nur Sinn den Neigungswinkel schon frühzeitig auf 30 - 40 Grad einzustellen um mittags die volle Energie der Sonne auszunutzen. Alternativ könnte ein Lichtsensor oder auch mehrere Lichtsensoren der Sonne nachgehen. Diese Methode ist im Grunde genommen vermutlich präziser und unabhängiger, denn das System braucht für diese Methode keine Internetverbindung.

## 5.2 System Steckplatine

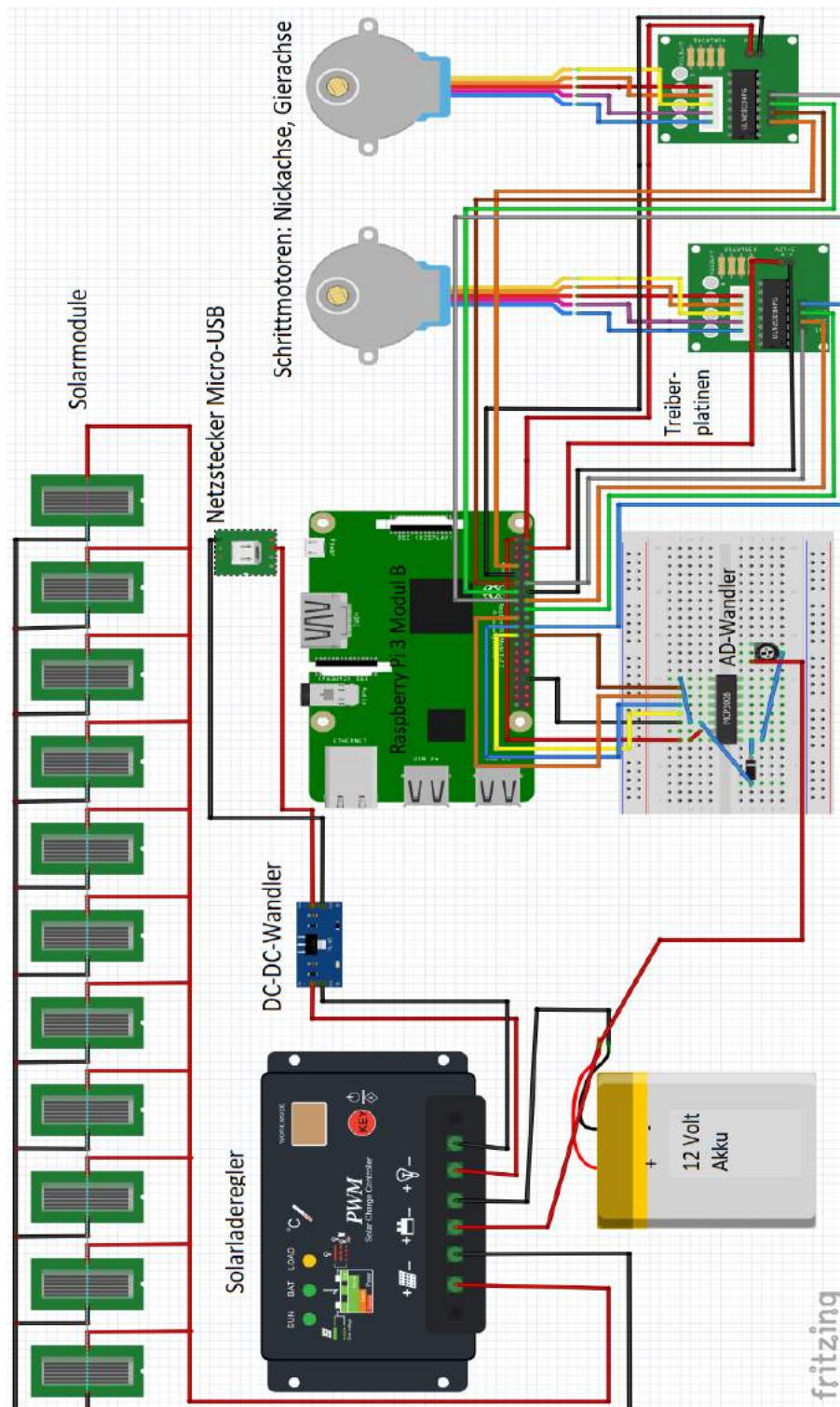


Abbildung 5.1: Steckplatine des Systems

## 5.3 System Schaltplan

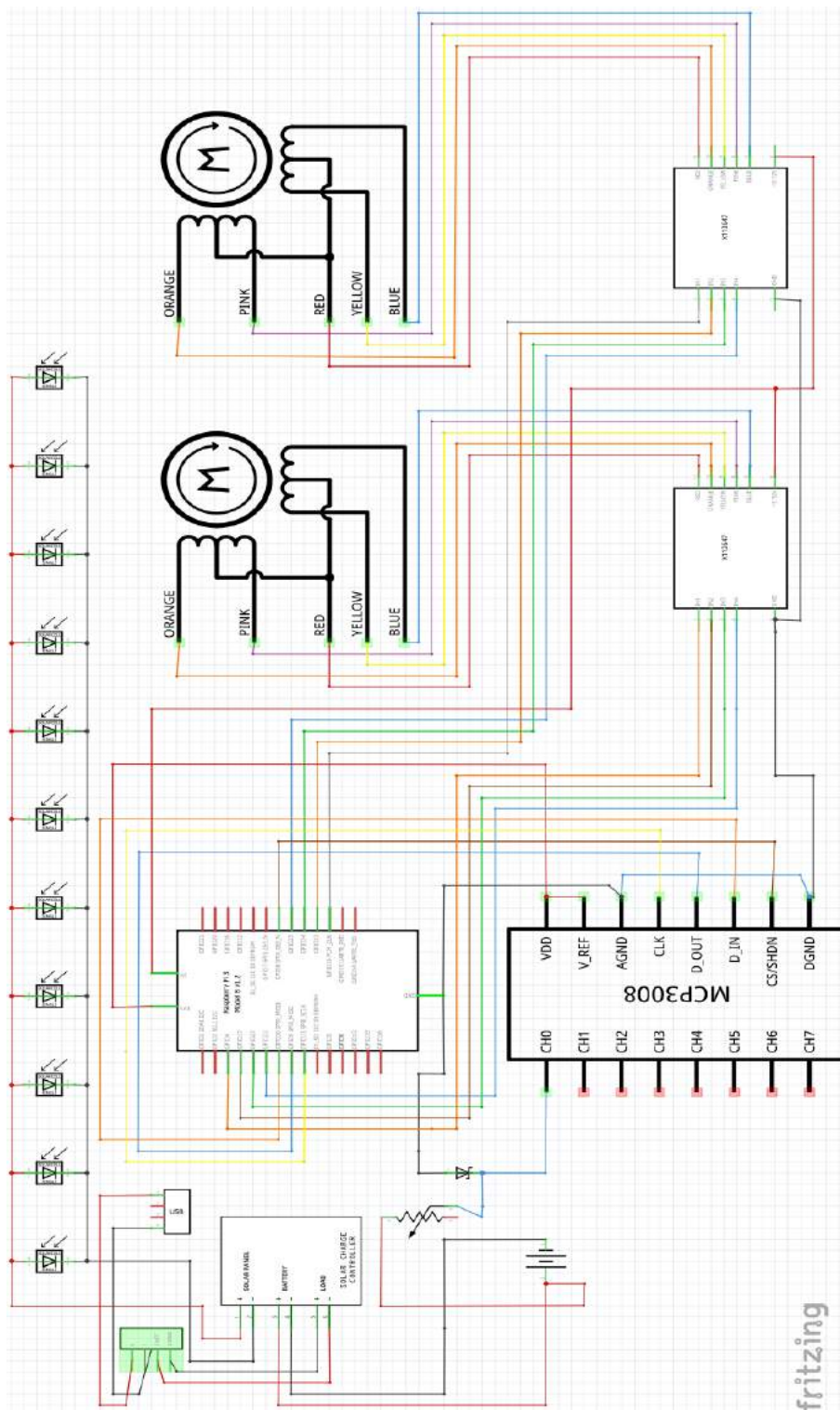


Abbildung 5.2: Schaltplan des Systems

## 5.4 Details zur Implementierung

Die Implementierung des gesamten Systems wird nun detailliert in einzelnen Schritten erklärt. Dabei sind die Schritte so strukturiert, dass für die jeweilige Komponente ein detailliertes Bild zur Verfügung steht und zusätzlich die nötigen Programme für die Interaktion zwischen den Komponenten erklärt werden.

### 5.4.1 Parallelschaltung der Solarmodule

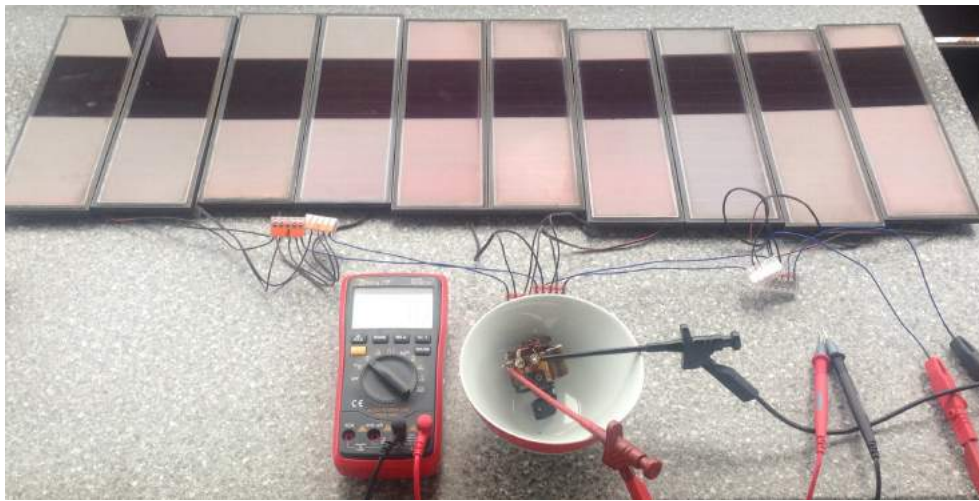


Abbildung 5.3: Parallelgeschaltete Solarmodule

Um eine Parallelschaltung der Solarmodule zu ermöglichen, verbindet man die zwei Leitungen von jedem Solarmodul mit der Hauptleitung. Die zwei Leitungen unterscheiden sich unter anderem durch die farbliche Markierung. Die negative Leitung ist komplett schwarz (Minuspole) und die positive Leitung ist schwarz, mit roten Linien (Pluspol) versehen. Die Abbildung 5.4 zeigt wie man die Leitungen miteinander verbindet.

Für die einzelnen Verbindungen, die die Hauptleitungen entstehen lässt, kommen sogenannte Verbindungsklemmen zum Einsatz. Von diesen Verbindungsklemmen gibt es eine verschiedene Anzahl von Eingängen in denen die Kabel hineinkommen. Insgesamt sind es sechs dieser Verbindungsklemmen, also drei pro Hauptleitung, und insgesamt besitzt jede Verbindungsklemme fünf Eingänge. Das bedeutet, dass die erste von drei Verbindungsklemmen vier Solarmodule verbindet. In den fünften Eingang der Verbindungsklemme kommt ein blaues Verbindungskabel, welches die erste und zweite Verbindungsklemme verbindet. Die zweite Verbindungsklemme nimmt drei weitere Solarmodule auf. Nun sind insgesamt sieben Solarmodule an der Hauptleitung verbunden. Um die letzten drei Solarmodule zu verbinden, verbindet man nun von der zweiten Verbindungsklemme ein weiteres blaues Verbindungskabel zur dritten Verbindungsklemme. An die dritte Verbindungsklemme verbindet man nun die letzten drei Solarmodule. Die Prozedur erfolgt für die positive sowie für die negative Hauptleitung. Um die beiden Hauptleitungen an den Akku anschließen zu können, gibt es



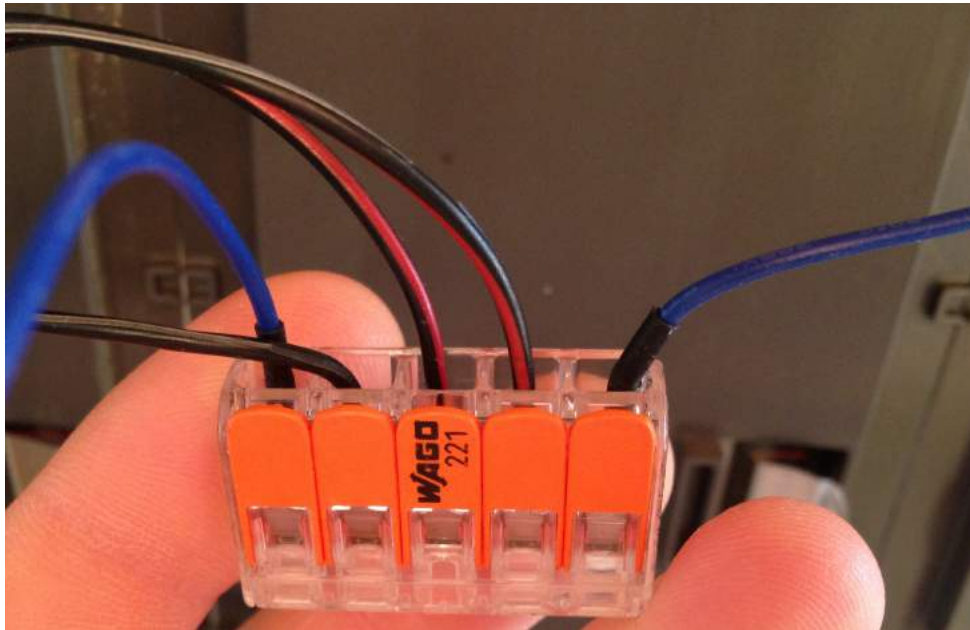


Abbildung 5.4: Verbindungsklemme (Wagoklemme)

in den letzten beiden Verbindungsklemmen jeweils einen freien Eingang. Nun verbindet man die Pluspolleitung der Hauptleitung mit dem Pluspol des Akkus. Das gleiche gilt für den Minuspol des Akkus.

## 5.4.2 Energieüberwachung des Akkus

Für die Implementierung der Energieüberwachung des Akkus kommen mehrere Komponenten zum Einsatz. Der Raspberry Pi besitzt die GPIO-Leiste, die die digitalen Signale mit einem Programm ausliest. Das Programm selbst ist auf dem Raspberry Pi gespeichert. Für die Umwandlung der analogen Signale (Spannungswerte) des Akkus in digitale Signale, gibt es eine Steckplatine mit weiteren elektrischen Bauteilen. Die Abbildung 5.5 zeigt diese Steckplatine.

### 5.4.2.1 Aufbau Steckplatine

Als erstes benötigt man eine Steckplatine. Diese Steckplatine besitzt einen Pluspol und einen Minuspol. Von dort bekommt die Schaltung ihre Energie oder auch analoge Signale. Wichtige Bauteile sind unter anderem der Potentiometer, die Z-Diode und der AD-Wandler (MCP3008). Der AD-Wandler besitzt eine Referenzspannung zwischen 2,7 V und 5,5 V. Von dem Akku fließen ungefähr 12 V in den Pluspol der Steckplatine. Diese 12 V dürfen auf keinen Fall in den AD-Wandler fließen, sonst würde die hohe Spannung die Schaltkreise des AD-Wandlers zerstören. Deswegen teilt der Potentiometer die 12 V auf 3,3 V. Der Wert 3,3 V ist einfach nur ein gewählter Wert zwischen den benötigten 2,7 V und 5,5 V. Der Potentiometer besitzt auf der Unterseite drei Beinchen. Ein Beinchen ist unterhalb, mittig angeordnet und die beiden anderen sind links und rechts angeordnet. Das rote Kabel führt den Strom vom



Konstellation (Pins vom Raspberry Pi zum AD-Wandler):

- Orange Kabel Pin 19 (GPIO10 MOSI) -> Pin 11 (DIN)
- Blaues Kabel Pin 21 (GPIO9 MISO) -> Pin 12 (DOUT)
- Gelbes Kabel Pin 23 (GPIO11 SCLK) -> Pin 13 (CLK)
- Braunes Kabel Pin 24 (GPIO8 CE0) -> Pin 10 (CS/SHDN)
- Rotes Kabel Pin 1 (3,3 V) -> Pin 15 (VREF) und Pin 16 (VDD)
- Schwarzes Kabel Pin 6 (GND) -> Pin 9 (DGND) und Pin 14 (AGND)

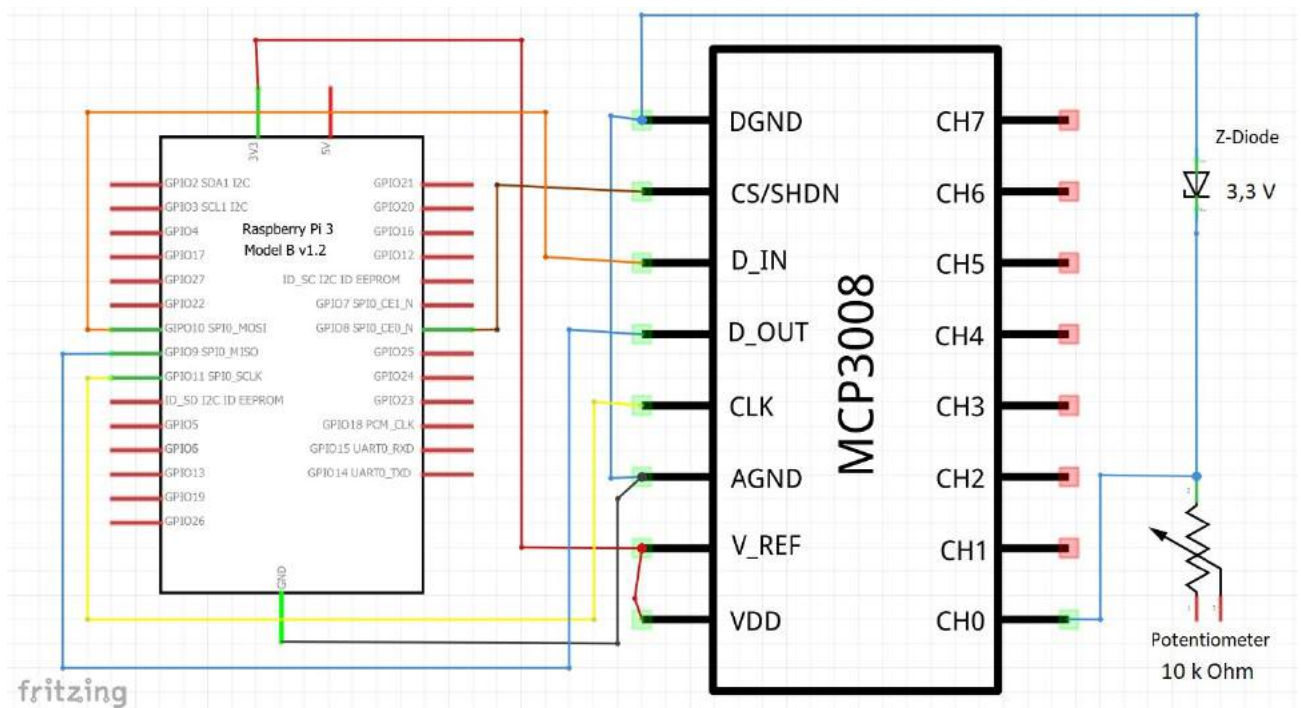


Abbildung 5.6: Schaltplan Energieüberwachung

#### 5.4.2.2 Programme für die Energieüberwachung

Für die automatische Ausführung der Programme beim Systemstart ist die `rc.local` verantwortlich. Die `rc.local` (siehe Programm 5.1) befindet sich im Ordner `etc`, der im `root`-Verzeichnis untergebracht ist. In Zeile 20 des Programms 5.1 steht der Befehl für die automatische Ausführung. Der fett gedruckte Befehl `nohup` startet das Programm im Hintergrund und schreibt gleichzeitig die Ausgaben des Programms in die `nohup.out` Datei, die sich im `root`-Verzeichnis befindet. Nach `nohup` folgt der Befehl `sudo`, der die Ausführung des Programms erlaubt. Danach zeigt der Befehl `python`, dass das folgende Programm eine Python Datei ist. Zum Schluss folgt der vollständige Pfad, der den Ort des Programms zeigt. Das anschließende Kaufmanns-Und bringt den Prozess in den Hintergrund.

Programm 5.1: Skript `rc.local`

```
1  #!/bin/sh -e
2  #
3  # rc.local
4  #
5  # This script is executed at the end of each multiuser runlevel.
6  # Make sure that the script will "exit 0" on success or any other
7  # value on error.
8  #
9  # In order to enable or disable this script just change the execution
10 # bits.
11 #
12 # By default this script does nothing.
13
14 # Print the IP address
15 _IP=$(hostname -I) || true
16 if [ "$_IP" ]; then
17   printf "My IP address is %s\n" "$_IP"
18 fi
19
20 nohup sudo python /home/pi/python_script_voltage/py_script.py &
21
22 exit 0
```

Die auszulesenden Daten aus dem AD-Wandler erfolgen mittels SPI-Protokoll. Dafür gibt es zwei Programme.

- `MCP3008.py`
- `py_script.py`

Das Programm `MCP3008.py` (siehe Programm 5.2) läuft mit der `SpiDev` Bibliothek. Diese ist im Programm 5.2 eingebunden. Zuvor muss man diese erst noch herunterladen, entpacken und installieren[40].



Kommandozeilenbefehle:

- `wget https://github.com/doceme/py-spidev/archive/master.zip`
- `unzip master.zip`
- `cd py-spidev-master`
- `sudo python setup.py install`

Programm 5.2: MCP3008.py [40]

```

1 #!/usr/bin/python
2 # coding=utf-8
3
4 from spidev import SpiDev
5
6 class MCP3008:
7     def __init__(self, bus = 0, device = 0):
8         self.bus, self.device = bus, device
9         self.spi = SpiDev()
10        self.open()
11
12    def open(self):
13        self.spi.open(self.bus, self.device)
14
15    def read(self, channel = 0):
16        adc = self.spi.xfer2([1, (8 + channel) << 4, 0])
17        data = ((adc[1] & 3) << 8) + adc[2]
18        return data
19
20    def close(self):
21        self.spi.close()

```

Im SPI-Protokoll gibt es einen Master und es können mehrere Slaves geben (Master-Slave-Prinzip). In diesem Fall ist der Master der Raspberry Pi und der Slave ist der AD-Wandler. Zeile 4 bezieht die SpiDev-Bibliothek ein um die nötigen Funktionen zu benutzen. In Zeile 7 und 12 übernehmen die Funktionen die Initialisierung des Busses. Dort öffnet die Funktion `open` den SPI-Bus 0 mit CS0. CS steht für Clock Signal, welches null ist, um dem Slave anzuzeigen, dass der Master sendebereit ist. Zeile 15 liest die Daten aus dem AD-Wandler aus. Die Funktion `xfer2`, in Zeile 16, sendet ein Byte-Array an den Slave und dabei bleibt CS dauerhaft aktiv[41]. In der Variable `adc` steht daraufhin die Antwort des Slaves. In Zeile 17 bekommt die Variable `data` den Wert aus der Antwort des Slaves. Die Funktion `close`, in Zeile 20, schließt den SPI-Bus.

Programm 5.3: py\_script.py

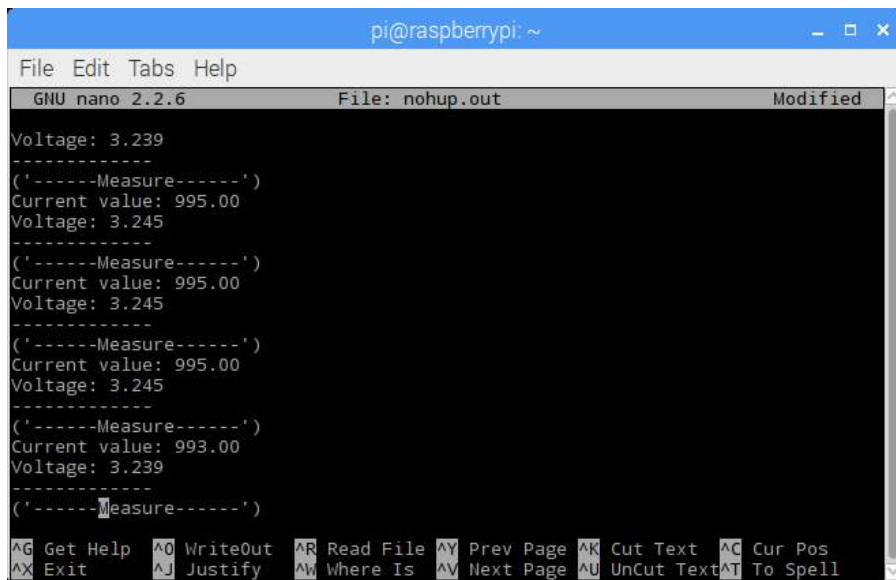
```

1  #!/usr/bin/python
2  # coding=utf-8
3  import os
4  import logging
5  import time
6  from MCP3008 import MCP3008
7
8  adc = MCP3008()
9  voltage = 0, value = 0, border = 3.285
10 logging.basicConfig(filename='status.log',
11 format='%(%asctime)s %(message)s')
12 logging.warning('python_script_started.')
13
14 def underVoltage():
15     print('under_voltage')
16     import sendEmail
17     logging.warning('python_script_ended->shutdown.')
18     time.sleep(1)
19     os.system("sudo_shutdown-hnow")
20
21 x = 0, var = 0
22 while True:
23     value = adc.read(channel = 0)
24     if value != 0:
25         print str(x) + '. Measure'
26         print('Current_value: %.2f' % value)
27         voltage = value * 3.34 / 1024
28         if voltage < border:
29             if var == 2:
30                 underVoltage()
31                 var += 1
32             else
33                 var = 0
34         print('Voltage: %.3f' % voltage)
35         print('')
36         x += 1
37         time.sleep(1)
38     else:
39         logging.warning('sensor_system_delivers_zero.')
40     break

```

Der Eintrag in der rc.local ruft als erstes das py\_script.py Programm auf (siehe Programm 5.3). Dies ist das Hauptprogramm, welches alle anderen Zusatzprogramme, wie zum Beispiel

die E-Mail-Versendung, aufruft. Die Zeilen 3 bis 6 fügen die nötigen Bibliotheken ein. Die Variable `adc` ist ein Objekt der Klasse `MCP3008()`. Zeile 9 zeigt die Variable `voltage`, welche den momentanen Spannungswert beinhaltet. Darauf folgend beinhaltet die Variable `value`, in Zeile 9, den ausgelesenen Wert aus dem AD-Wandler. Die Variable `border` zeigt den minimalen Spannungswert für die Bedingung der Abschaltung des Systems. Beim erstmaligen Start des Programms, schreibt das Programm in Zeile 10 bis 12 eine Nachricht (python script started) mit Datum und Uhrzeit in die Datei `status.log`. Die while-Schleife, in Zeile 22 bis 40, liest jede Sekunde den Wert aus dem AD-Wandler aus. Zeile 23 liest mit Hilfe der Funktion `read` den Wert am Kanal 0 aus. Falls der Wert 0 ist, muss ein Fehler in der Schaltung vorliegen, denn die Spannung des Akkus liegt immer über 0 V. Dabei springt das Programm in den `else`-Zweig (Zeile 38) und schreibt eine Nachricht (sensor system delivers zero) in die `status.log` Datei und bricht die while-Schleife ab. Liegt der Wert über Null, schreibt das Programm mit den `print`-Methoden Zeile 25, 26, 34 und 35) den Wert in die `nohup.out` Datei. Die Abbildung 5.7 zeigt diese Ausgabe.



```

pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.2.6 File: nohup.out Modified
Voltage: 3.239
-----
('-----Measure-----')
Current value: 995.00
Voltage: 3.245
-----
('-----Measure-----')
Current value: 995.00
Voltage: 3.245
-----
('-----Measure-----')
Current value: 995.00
Voltage: 3.245
-----
('-----Measure-----')
Current value: 993.00
Voltage: 3.239
-----
('-----Measure-----')

```

Abbildung 5.7: nohup Ausgabe

In Zeile 27 wandelt das Programm den ausgelesenen digitalen Wert in einen Spannungswert um. Da der MCP3008 AD-Wandler Werte von 0 bis 1023 ( $2^{10}$ ) darstellen kann, verwendet man die Formel für folgende Umrechnung[40]:

$$voltage = \frac{value \cdot 3.34 V}{1024} \quad (5.1)$$

Die Spannung 3.34 V ist mit dem höchsten aufgeladenen Spannungswert des Akkus zu vergleichen. Das sind 14 V bei einer vollen Aufladung. In dem Programm ist die Untergrenze 3.285 V, wenn die Spannung des Akkus zur Neige geht. Die 1024 ist die Anzahl an Möglichkeiten für den digitalen Wert. Somit bekommt man die Umrechnung des digitalen Wertes

in einen Spannungswert auf Basis des benutzten Spannungsbereiches im MCP3008 AD-Wandler. Falls der Spannungswert unterhalb der angegebenen Grenze liegt, führt das Programm die Funktion `underVoltage` (Zeile 30) aus. Die Abbruchbedingung steht erst nach drei nacheinander, positiv gelaufenen Überprüfungen unter des Grenzwertes. Die Funktion `underVoltage` schreibt eine Nachricht in die Datei `nohup.out` und sendet daraufhin in Zeile 16 eine E-Mail. Für das Versenden der E-Mail sind separate Programme verantwortlich. Danach schreibt das Programm eine weitere Nachricht (`python script ended -> shutdown`) in die Datei `status.log`. Zum Schluss startet der Befehl in Zeile 19 das Herunterfahren des Systems.

Um die Anforderung der Benachrichtigung via E-Mail zu ermöglichen, erfolgen zwei weitere Programme plus einer weiteren Textdatei im INI-Format. Diese Dateien sind auf der Internetseite IoT Bytes Bits and Bytes of IoT [42] erhältlich.

- `sendEmail.py`
- `emailHandler.py`
- `settings.ini`

Unter dem Punkt 8.1 im Anhang befindet sich die Datei `emailHandler.py` (siehe Programm 8.1). Es bedarf keiner Änderung in dieser Datei. Diese ist hauptsächlich für die Formatierung und für das Versenden der E-Mail verantwortlich[42].

In der `settings.ini` Datei (siehe Programm 5.4) sind folgende Änderungen erforderlich. Zeile 6 und 8 beinhalten die eigene E-Mail Adresse und in Zeile 11 ist die Angabe des Passworts erforderlich.

Programm 5.4: `settings.ini` [42]

```

1  [EMAIL]
2  SMTP_ADD=smtp.gmail.com           ; SMTP Server Address (For Gmail
3  use smtp.gmail.com)
4  SMTP_PORT=465                     ; SMTP Server Port (Gmail default
5  smtp port 465)
6  FROM_ADD=1234qwer@gmail.com        ; From email Address (For Gmail
7  you may use you email id)
8  USERNAME=1234qwer@gmail.com        ; SMTP Server User Name
9  (for authentication)
10 PASSWORD=DasPasswort123           ; SMTP Server Password
11 (for authentication)

```

Das Programm `sendEmail.py` (siehe Programm 5.5) ist die ausführende Datei, die in dem Programm `py_script.py` vorkommt um die E-Mail zu versenden (Zeile 17). Für die Nachricht in der E-Mail sind Zeilen 10 und 11 verantwortlich.

Programm 5.5: `sendEmail.py` [42]

```

1  #!/usr/bin/python
2  # coding=utf-8
3
4  # import Class_eMail from email_handler.py file
5  import datetime
6  from emailHandler import Class_eMail
7
8  now = datetime.datetime.now()
9  var_To_Email_ID = "1234qwer@gmail.com"
10 var_SUBJECT = "Raspberry_Pi_Voltage_System"
11 var_EMAIL_BODY = "Time_of_shutdown:" + now.strftime("%Y-%m-%d_%H:%M:%S")
    + "\nThe power of battery is too low. System shutdown."
12
13 # create class object
14 email = Class_eMail()
15
16 # send email
17 email.send_Text_Mail(var_To_Email_ID, var_SUBJECT, var_EMAIL_BODY)
18
19 # delete class object
20 del email

```

### 5.4.3 Ansteuerung der Schrittmotoren

Für das Ansteuern der Schrittmotoren benötigt man die GPIO-Leiste des Raspberry Pi, jeweils die zwei Schrittmotoren sowie deren Treiberplatinen. Zusätzlich kommen noch Verbindungskabel zum Einsatz (siehe Abbildung 5.8).

#### 5.4.3.1 Verbindungsaufbau der Schrittmotoren mit dem Raspberry Pi

Die Verbindung zum Schrittmotor erfolgt über vier Pins, die vom Raspberry Pi zur Treiberplatine führen. Bei zwei Schrittmotoren sind es folglich insgesamt acht Pins, die noch nicht belegt sein dürfen. Die Pins 1, 6, 19, 21, 23 und 24 sind schon durch die Energieüberwachung belegt. Welche Pins für die auf der GPIO-Leiste des Raspberry Pi benutzt werden, ist egal[43]. Für die Verbindung zur Treiberplatine sind folgende Pins eingesetzt.

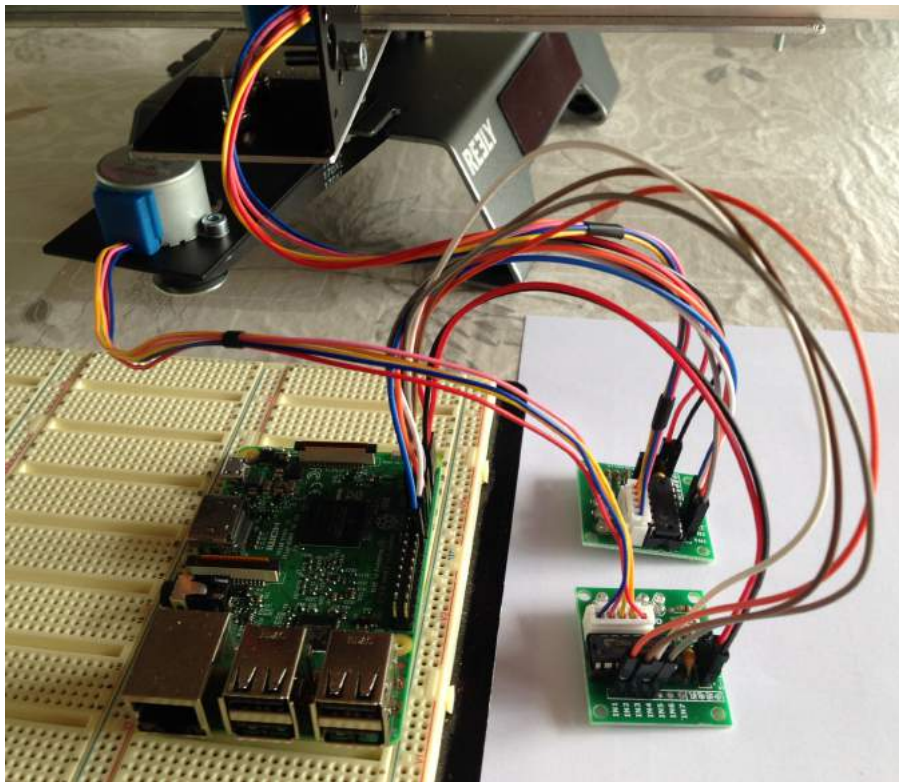


Abbildung 5.8: Verbindungen Schrittmotoransteuerung

GPIO-Leiste Raspberry Pi zur Treiberplatine (Schrittmotor Gierachse):

- Graues Kabel Pin 12 (GPIO18) -> IN1
- Orange Kabel Pin 16 (GPIO23) -> IN2
- Grünes Kabel Pin 18 (GPIO24) -> IN3
- Blaues Kabel Pin 22 (GPIO25) -> IN4

- Rotes Kabel Pin 4 (5,0 V) -> Pluspol Anschluss
- Schwarzes Kabel Pin 6 (GND) -> Minuspol Anschluss

GPIO-Leiste Raspberry Pi zur Treiberplatine (Schrittmotor Nickachse):

- Orange Kabel Pin 7 (GPIO4) -> IN1
- Braunes Kabel Pin 11 (GPIO17) -> IN2
- Grünes Kabel Pin 13 (GPIO27) -> IN3
- Blaues Kabel Pin 15 (GPIO22) -> IN4
- Rotes Kabel Pin 2 (5,0 V) -> Pluspol Anschluss
- Schwarzes Kabel Pin 9 (GND) -> Minuspol Anschluss

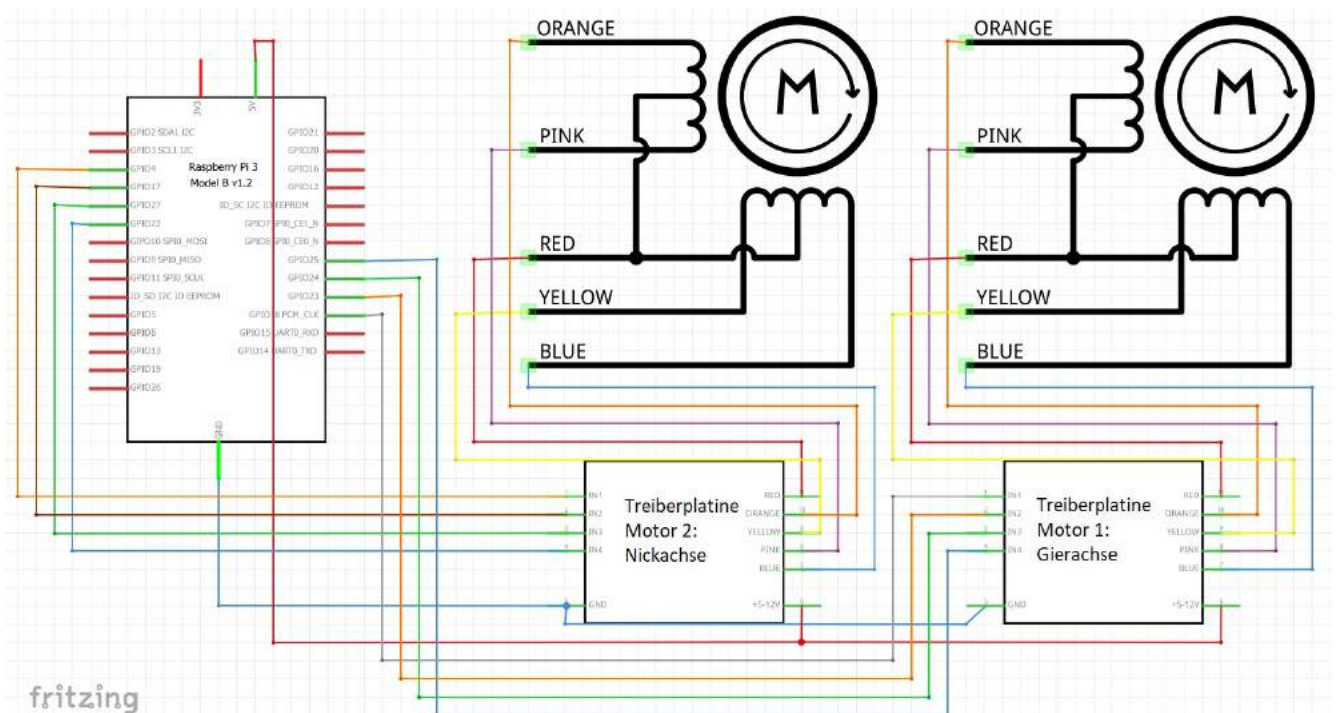


Abbildung 5.9: Schaltplan Schrittmotoransteuerung

### 5.4.3.2 Der Dienst cron

Um den Dienst cron benutzen zu können, erstellt man erstmal den Dienst für den Benutzer. In dem Terminal steht folgender Befehl.

- `crontab -e`

In der Abbildung 5.10 ist die Einstellung für die Zeitplanung zu sehen[28].

- m = Minuten
- h = Stunden
- dom = Tag im Monat
- mon = Monat im Jahr
- dow = Wochentag (0 - 7) beginnend bei Sonntag

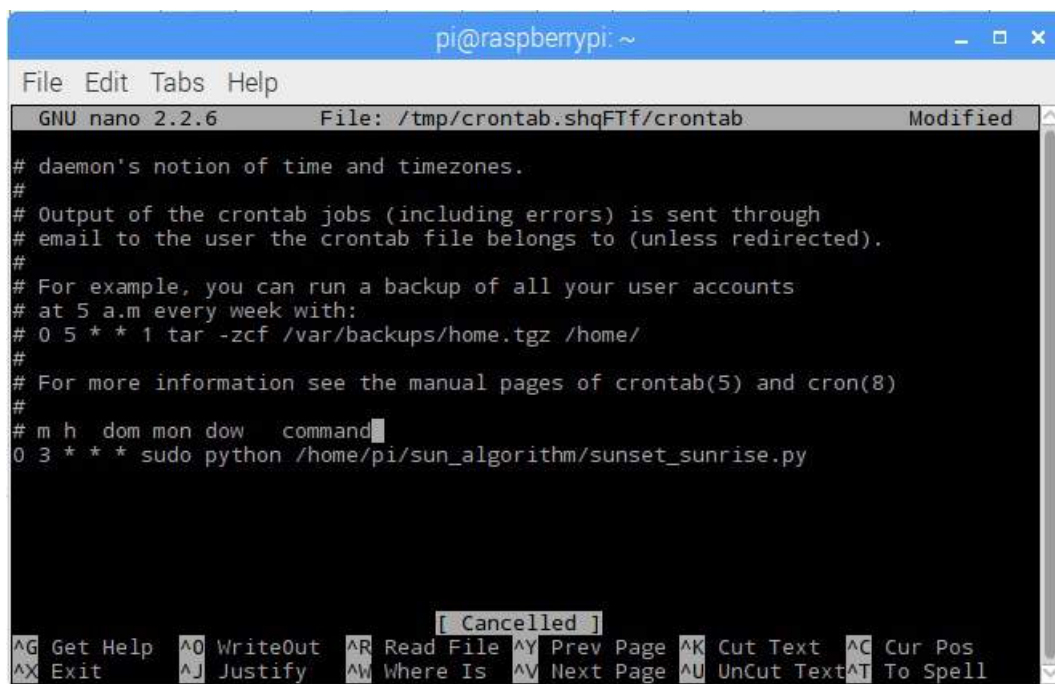


Abbildung 5.10: cron

In der obigen Abbildung führt cron das Programm `sunset_sunrise.py` jeden Tag in der Woche, egal welcher Monat oder welcher Tag im Monat um 03:00 Uhr morgens aus.



### 5.4.3.3 Programme für die Ansteuerung der Schrittmotoren

Es gibt ein Programm für jeweils einen Schrittmotor. Das Programm `stepperpitch.py` ist für den Neigungswinkel verantwortlich. Das Programm `stepperyaw.py` ist für die Drehung des Gestells verantwortlich. Allgemein ist zu sagen, dass diese Programme sich nicht groß von einander unterscheiden. Deswegen werden diese auch gemeinsam beschrieben (siehe Programme 8.2 und 8.3 im Anhang). Die Gemeinsamkeiten liegen in der Verwendung der selben Funktionen für das Drehen der Schrittmotoren. Für eine Rechtsdrehung ist die Funktion `RIGHT_TURN(deg)` verantwortlich. Die Linksdrehung geschieht mit der Funktion `LEFT_TURN(deg)`. Der Parameter `deg` steht für den einzustellenden Winkel in Grad. Bevor man diese Funktionen benutzen kann, ist es wichtig die benutzten Pins auf der GPIO-Leiste des Raspberry Pis im Programm einzutragen. Die Variablen `A`, `B`, `C`, `D` beinhalten diese. Die Befehle `GPIO.setmode(GPIO.BOARD)` steht für das GPIO-Boardschema des Raspberry Pi. Das heißt, dass Programm benutzt die Nummerierung der Pins. Der Befehl `GPIO.setup(A, GPIO.OUT)` setzt den Pin als Ausgang. Die Funktion `GPIO_SETUP(a,b,c,d)` dient zur Ansteuerung des Schrittmotors. Die Parameter beinhalten die jeweiligen Pins für die Ansteuerung. Die Ansteuerung erfolgt über acht Halbschritte um die Magnete im Motor anzutreiben (siehe Kommentar (Schemata acht Halbschritte) Programm 8.2 oder 8.3). Wichtig ist hierbei die Getriebeübersetzung von 1 zu 64 (siehe Datenblatt im Anhang Punkt 8.5).

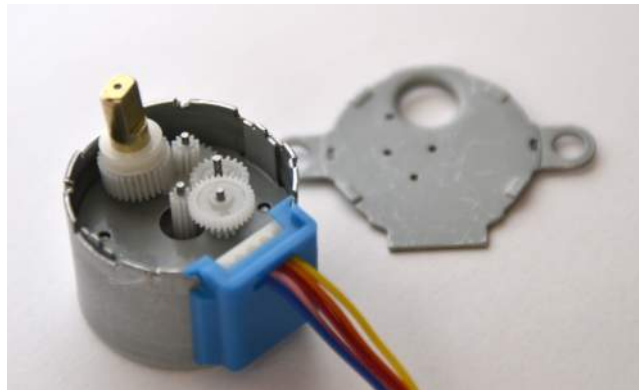


Abbildung 5.11: Schrittmotor internes Getriebe<sup>10</sup>

<sup>10</sup>Bildquelle: <https://coeleveld.com/arduino-stepper-uln2003a/>

Für eine volle 360 Grad Umdrehung des Schrittmotors, gilt folgende Berechnung.

$$512 = 8 \cdot 64 \quad (5.2)$$

Der Wert 512 steht für die Anzahl an Zyklen für die Durchführung der acht Halbschritte um eine 360 Grad Umdrehung zu erreichen. In den Funktionen `LEFT_TURN` und `RIGHT_TURN` ist dieser Wert in der Variable `full_circle` gespeichert. Für die Berechnung des Winkels (`degree`) gilt folgendes.

$$degree = \frac{512}{360 \cdot deg} \quad (5.3)$$

Die Variable `deg` ist der übergebene Parameter in der Funktion, der für den tatsächlich gewünschten Wert des Winkels steht. Die Variable `degree` hat nun die Aufgabe, in der Zählschleife die Anzahl der durchzuführenden Zyklen anzugeben. Pro Zyklus dekrementiert die Schleife die Variable `degree` um eins bis die Abbruchbedingung von 0.0 erreicht ist.

Der Unterschied zwischen den beiden Programmen liegt in der Hauptfunktion `main`. Der Sonnenverlauf Algorithmus ruft die Hauptfunktion `main` des Programms `stepperyaw.py` auf um diesem die nötigen Daten zu überreichen. Das Programm `stepperyaw.py` ruft in einem separaten Prozess die Hauptfunktion `main` des Programms `stepperpitch.py` auf, welches den Neigungswinkel viermal einstellt. Der zweite Prozess steht für die Funktion der Drehfunktion. Beide Hauptfunktionen haben also unterschiedliche Aufgaben.

Das Programm `stepperyaw.py` (siehe Anhang Programm 8.3):

Die Hauptfunktion `main(args)`, in Zeile 91, bekommt einen Parameter als Array übergeben. In diesem Array befinden sich die ausgerechneten Daten des Sonnenverlauf Algorithmus. Die Daten sind im Array wie folgt gegliedert:

- Position 0: Sonnenaufgang Stunde
- Position 1: Sonnenaufgang Minute
- Position 2: Zeitspanne des Sonnenverlaufs in Minuten
- Position 3: Winkel des Sonnenverlaufs

Das Gestell stellt sich immer nach dem Sonnenuntergang auf die Südausrichtung zurück. Das sind 180 Grad. In Zeile 102 dreht sich das Gestell auf die Startposition des Sonnenaufgangs. Ziel ist es natürlich, dass die Funktion des Drehens erst dann anfängt, wenn der Zeitpunkt des Sonnenaufgangs erreicht ist. Dafür besitzt das Programm zwei Variablen für `currentTimeHour` und `currentTimeMinute`, die die aktuelle Zeit vom Aufruf des Programms darstellen. In Zeile 105 soll nun die Differenz von der Zeit des Sonnenaufgangs mit der aktuellen Zeit des Aufrufs berechnet werden. Diese Differenz (`timeToSleep`) steht für den Zeitpunkt bis zum eigentlichen Sonnenaufgang.

Beispielsweise startet der Dienst `cron` den Sonnenverlauf Algorithmus um 3:00 Uhr morgens. Der Sonnenaufgang startet um 6:30 Uhr. Die Differenz beträgt 3 Stunden und 30 Minuten. Genau diese Wartezeit hat das Programm bis zum Sonnenaufgang (Zeile 109). In Zeile 118 startet das Programm den separaten Prozess für die Funktion des Neigungswinkels. In der Zeile 111 bestimmt der Wert des Winkels die Anzahl an Schleifendurchgängen und damit, wie oft sich das Drehgestell um einen Grad dreht. In Zeile 119 startet das Programm den zweiten Prozess auf. Das Programm stoppt den Schleifendurchlauf in Zeile 29, um die ein Grad gleichmäßig auf den Zeitverlauf zu verteilen.

Die Hauptfunktion `stepperpitch.py` (siehe Anhang Programm 8.2):

In Zeile 74 bekommt die Hauptfunktion die Zeit des Sonnenverlaufes übergeben um auch hier über diesen Zeitraum bestimmte Winkel einstellen zu können. Die Ausgangsposition der Solarmodule ist senkrecht. Die Winkel sind wie folgt auf die Zeit des Sonnenverlaufs

eingeteilt. Die Zeitspanne von 0 Grad auf 25 Grad und von 25 Grad auf 35 Grad sollen jeweils 10 Prozent von der Zeit betragen (Zeile 80 und 83). Der Winkel von 35 Grad bekommt eine Zeitspanne von 60 Prozent des Sonnenverlaufs (Zeile 86). Die Zeitspanne gilt für den Mittagsverlauf der Sonne. Danach verringert das Programm den Neigungswinkel wieder auf 25 Grad, welcher die restlichen 20 Prozent der Zeit beansprucht (Zeile 89). Zum Schluss ist die senkrechte Ausgangsposition wieder erreicht.

#### 5.4.3.4 Implementierung Sonnenverlauf Algorithmus

Die Hauptaufgaben dieses Programms (siehe Anhang Programm 8.4) bestehen darin, die Zeiten des Sonnenaufgangs und Sonnenuntergangs zu ermitteln. Danach berechnet das Programm den Azimut der Sonne mit den Zeiten und den Koordinaten vom Standort des Systems (Zeile 17). Die Zeiten bekommt das Programm von einer Internetseite (URL: Zeile 50) ein JSON-Objekt via API. Die URL bezieht noch ein paar Parameter zur Ermittlung der Zeiten:

- day (Zeile 46)
- month (Zeile 47)
- year (Zeile 48)
- longitude (Zeile 20)
- latitude (Zeile 21)

Die Ausgabe des JSON-Objekt sieht wie folgt aus (Zeile 55):

- sunrise: 08:20, sunset: 16:20

Für eine Benutzung der Zeiten, bearbeiten folgende Zeilen das JSON-Objekt um die benötigten Werte zu bekommen.

- Ausgabe: 08:20 (Zeile 57)
- Ausgabe: 16:20 (Zeile 58)
- Ausgabe: 08 (Zeile 60)
- Ausgabe: 20 (Zeile 61)
- Ausgabe: 16 (Zeile 63)
- Ausgabe: 20 (Zeile 64)

In Zeile 66 berechnet das Programm den Zeitunterschied zwischen Sonnenaufgang und Sonnenuntergang mit diesen Werten. Die Variable `SonnenaufgangWinkel` beinhaltet zuerst den eigentlichen Winkel des Sonnenaufgangs (Zeile 72). Des Weiteren besitzt die Variable `Winkel` den Wert vom kompletten Winkel des Sonnenverlaufs (Zeile 74). Die Berechnung beruht auf folgenden Gedanken:

Der Winkel des Sonnenaufgangs reicht aus um den kompletten Winkel zu berechnen. Als erstes nimmt man den Grad des Südens (180 Grad) und subtrahiert davon den Sonnenaufgang Winkel. Diese Differenz ist der Winkel zwischen Süden und Sonnenaufgang. Da die Winkel vom Sonnenaufgang und Sonnenuntergang zum Süd Winkel gleich sind, muss nur noch die ausgerechnete Differenz verdoppelt werden. Daraus resultiert der komplette Winkel des Sonnenverlaufs.

In Zeile 77 schreibt das Programm die Daten in eine CSV-Datei. Der Nutzen liegt nur in der Überprüfung, ob das Programm für den Tag die richtigen Daten besitzt. Zum Schluss führt die Zeile 86 das Programm `stepperyaw.py` mit den erforderlichen Daten aus.

Im nächsten Punkt folgt der Aufbau des Drehgestells. Dort ist dann auch ersichtlich, auf welche Komponenten die Programme Einfluss haben.

### 5.4.4 Aufbau Drehgestell

Der Aufbau des Drehgestells erfolgt in zwei Unterpunkten. Der erste Punkt zeigt den modifizierten Aufbau des Standfußes. Der zweite Punkt beschreibt dann den Aufbau des Gestells für die Solarmodule.

#### 5.4.4.1 Aufbau modifizierter Standfuß



Abbildung 5.12: Modifizierter Standfuß

Der Standfuß ist insofern modifiziert, dass der Schrittmotor für die Drehung des Gestells, an dem Standfuß montiert ist. Die Modifikation besteht aus einem schwarzen länglichen

Blech aus Stahl mit einem Loch, welches an den Standfuß mit Schrauben befestigt ist (siehe Abbildung 5.12). In diesem Fall ist das Gewicht des Stahlblechs nicht wichtig. Zudem muss das Stahlblech stabil sein, damit dieses sich bei einem Kraftaufwand des Schrittmotors nicht biegt. Der Schrittmotor befindet sich weiter hinten und ist ebenfalls mit Schrauben befestigt. Die Welle des Schrittmotors führt hingegen durch das Loch auf die Rückseite des Stahlblechs (siehe Abbildung 5.13).



Abbildung 5.13: Unterseite, Riemenantrieb

Abmessung des Stahlblechs:

- Länge: 124 mm
- Breite: 50 mm
- Höhe: 2 mm

Bei den Abmessungen ist im Grunde genommen nur die Höhe beziehungsweise Dicke des Stahlblechs wichtig. Bei einer Dicke von 2 mm ist es selbst mit den Händen schwierig dieses zu verbiegen. Auf der Rückseite (siehe Abbildung 5.13) sieht man den Riemenantrieb. Dafür sind zwei Räder mit Zähnen und ein Riemen mit identischen Zähnen wie von den Rädern vorgesehen. Das große Rad besitzt 35 Zähne und das kleine Rad besitzt 10 Zähne. Das kleine

Rad ist an der Welle vom Schrittmotor befestigt. Der Vorteil dieser Konstruktion liegt in der Aufwendung von weniger Kraft, die der Schrittmotor für die Drehbewegung aufbringen muss. Dafür muss dieser einen längeren Weg gehen, sodass bei dem größeren Rad eine komplette Umdrehung entsteht. Das Verhältnis ist in diesem Fall 10 zu 35. Das bedeutet, dass sich das Rad 3.5 mal drehen muss, sodass eine ganze Umdrehung beim großen Rad erfolgt. Hier kann man nun sehen, warum in der Zeile 107 des Programms 8.3 `stepperyaw.py`, die Funktion `LEFT_TURN` den Wert 3.5 benutzt. Dieser Wert steht dann für ein Grad am großen Rad. Das große Rad ist mit einer Schraube an das Kugellager befestigt. Für die Befestigung mit dem Gestell dienen eine Schraube und drei Unterlegscheiben.

### 5.4.4.2 Aufbau Gestell



Abbildung 5.14: Gestell Vorderseite

Das Gestell ist komplett neu angefertigt und ist für maximal zehn der Solarmodule ausgelegt (siehe Abbildung 5.14). Für die Implementierung des Gestells ist es wichtig zu wissen welche Abmessung die Solarmodule besitzen.

Abmessung der Solarmodule:

- Länge: 298 mm
- Breite: 90 mm
- Höhe: 6 mm

Für die Benutzung von zehn Solarmodulen muss der Rahmen folglich von innen eine Breite von 900 mm besitzen. Die Länge von 298 mm bleibt. Zuerst muss man den Rahmen für die Solarmodule anfertigen um daraufhin die anderen Teile des Gestells fertigstellen zu können. Die Abbildung 5.15 zeigt die Typen der Profile und die Winkel für die Befestigung. Die Profile erfüllen in der Konstruktion unterschiedliche Aufgaben und haben deshalb unterschiedliche Bauweisen. Ausschlaggebend für den Bau sind hier die Breite und die Höhe der Profile, da die Länge für die jeweiligen Anforderungen angepasst ist.



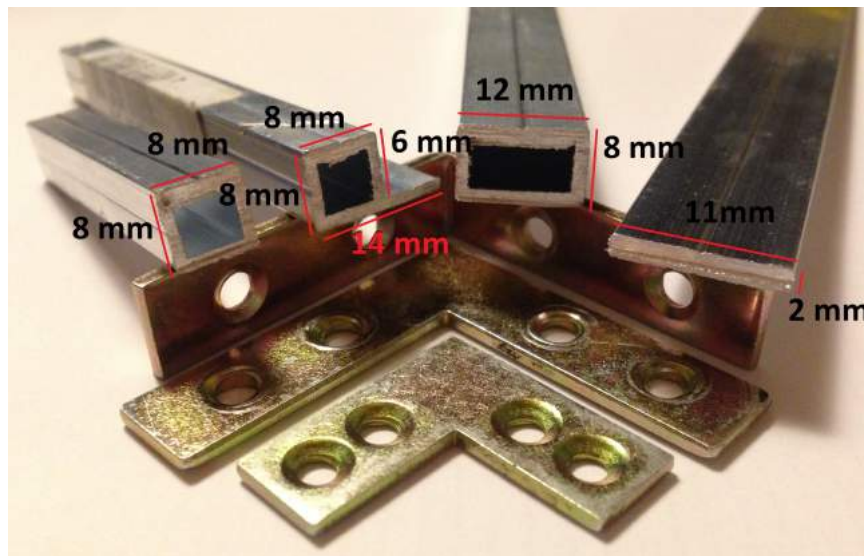


Abbildung 5.15: Profile und Winkel

In der Abbildung 5.15 sind unter anderem auch die Abmessung der einzelnen Profile aufgezeichnet. Für den Solarmodul Rahmen dienen die ersten beiden Profile auf der linken Seite. Es gibt jeweils zwei von jedem Profil pro Seite. Das quadratische Profil ist einmal links und rechts angeordnet und hat eine Länge von 315 mm. Das zweite Profil hat eine Länge von 920 mm. Für die Länge sind nochmal 20 mm hinzugekommen um den Solarmodulen im Rahmen ein wenig Spiel zu geben. Im Anhang zeigt die Abbildung 8.3 die Abmessung. Die Winkel befestigen die Profile miteinander. Insgesamt sind es acht große Winkel, einmal vier auf der Vorderseite und vier auf der Rückseite. Somit ist der Rahmen stabil gebaut. Die verbauten Profile auf der Ober- und Unterseite besitzen noch eine 6 mm lange Kante, die die Solarmodule von innen halten. Da die Solarmodule noch immer rausfallen können, müssen auf der Vorderseite des Rahmens, zwei weitere Profile die Solarmodule halten. Das äußerst rechte Profil aus der Abbildung 5.15 unterstützt hierbei. Die Abbildung 8.2 im Anhang zeigt die Platzierung des Profils mit einer Länge von 850 mm. Da die Höhe der Solarmodule 6 mm beträgt, passen diese ohne Probleme zwischen die Profile, die auch ein Spaltmaß von 6 mm haben.

Für die Drehbewegung des Neigungswinkels ist das Zapfenband verantwortlich (siehe Abbildung 8.5 und 8.6 im Anhang). Das Zapfenband befindet sich zwischen Rahmen und den senkrechten Stützen, die auf dem zweiten Rahmen befestigt sind. Auf der Abbildung 8.6 sieht man auch ganz genau das Gewinde zwischen den beiden beweglichen Elementen. Das Drehgewinde ist nicht genau in der Mitte des Rahmens platziert, sondern ungefähr 10 mm oberhalb des Mittelpunktes vom Rahmen. Der Grund ist, dass bei einer Neigung ein Ungleichgewicht herrscht und der Rahmen samt Solarmodule von alleine in die Ausgangsposition fallen soll. In der Abbildung 8.6 sind die Abmessungen des Zapfenbandes zu sehen.

Nun kommt das letzte Profil zum Zuge (siehe Abbildung 5.15 zweites von rechts). Für die Konstruktion des Rahmens benutzt man dieses Hochkant, damit der Rahmen wegen des Gewichts der Solarmodule sich nicht durch biegt. Die Abmessungen für die Stützen und den

Rahmen sind in der Abbildung 8.4 zu sehen. Die Stützen haben eine Länge von 209 mm. Diese Länge ist um Grunde genommen frei gewählt und bietet für die Position der Zapfenbänder Luft nach oben. Die Stützen sind mit den Winkeln (Hochkant) an dem Rahmen auf beiden Seiten befestigt. Der Rahmen ist mit den kleinen Winkeln (insgesamt acht Stück) befestigt. Die Abmessungen der Bodenplatte sind mit einem Din A4 Blatt zu vergleichen (Länge: 300 mm, Breite: 210 mm). Der Rahmen ist mit vier Schrauben an der Bodenplatte befestigt. In der Mitte der Bodenplatte befindet sich das Loch für die Befestigung mit dem Standfuß. Auf der Rückseite des Gestells befindet sich noch die Winde mit dem Schrittmotor für den Neigungswinkel (siehe Abbildung 5.16). Ein zusätzliches Aluminiumblech (Länge: 125 mm, Breite 50 mm), welches an der Bodenplatte mit zwei Schrauben befestigt ist, versetzt den Schrittmotor um 82 mm nach hinten. Der Schrittmotor selbst, ist an einem schwarzen Blech montiert. Das schwarze Blech stammt aus Altbeständen und besitzt idealerweise ein großes Loch für die Winde des Schrittmotors (siehe Abbildung 5.16). Die Winde besteht aus einem Kupplungs-Einsatz (Farbe: Gold) und zwei Stellringen (Farbe: Silber). Das Seil ist im Gewindestift (kleine schwarze Schraube) fixiert. Zwischen den Stellringen ist genau 1 mm Platz um das Seil einzurollen.

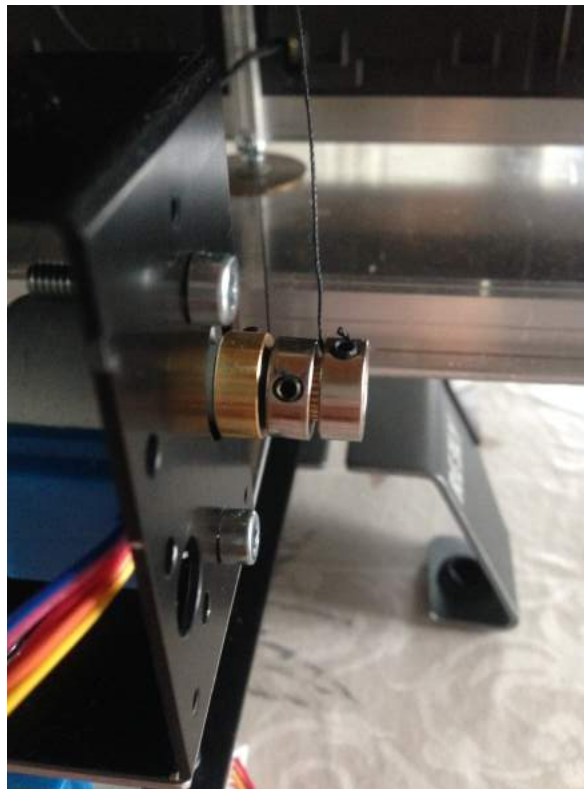


Abbildung 5.16: Winde



Ab hier kann der Wert von 40,6 im Programm 8.2 ab Zeile 81 erklärt werden. Der Stelling hat einen Durchmesser von 7 mm und der Rahmen mit den Solarmodulen minus die 10 mm Versetzung einen Durchmesser von 305 mm.

$$40,6 = \frac{305 \text{ mm}}{7 \text{ mm}} \quad (5.4)$$

Das bedeutet, dass der Stelling sich 40,6 mal drehen muss, damit eine volle Umdrehung des Rahmens entsteht. Dieser Wert ist demnach für den nötigen Neigungswinkel verantwortlich. Die Konstruktion ist somit komplett. Im nächsten Kapitel sieht man die Ergebnisse, die das System in verschiedenen Testläufen erbringt.

## 6 Ergebnisse

Die Ergebnisse umfassen unterschiedliche Testläufe und anschließend eine Bewertung des Systems im Hinblick auf die alltägliche Benutzung und inwieweit dieses bei verschiedenen Wetterverhältnissen reagiert.

### 6.1 Testlauf ohne Algorithmus

Dieser Test zeigt den Betrieb der Solaranlage ohne Sonnenenverlauf Algorithmus. Der Test findet am 23.12.2017 statt. Der Sonnenaufgang ist um 08:24 Uhr und der Sonnenuntergang ist um 16:22 Uhr[44]. Die Wettervorhersage besagt für den Tagesverlauf ein bewölktes und somit kein sonnenreiches Wetter.

Das System ist wie folgt aufgebaut:

- Die Solaranlage zeigt nach Süden.
- Der Neigungswinkel ist auf 35 Grad voreingestellt.
- Die Schrittmotoren sind nicht am Raspberry Pi angeschlossen.
- Die Energieüberwachung ist am Raspberry Pi angeschlossen.
- Der Akku ist aufgeladen (12.58 V).

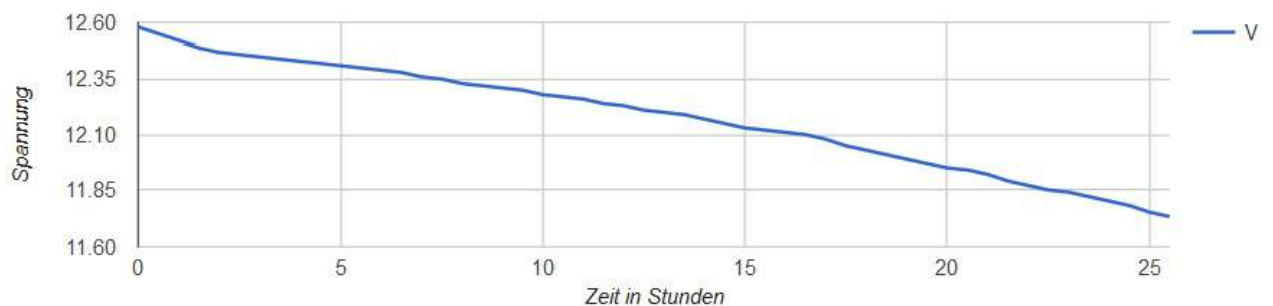


Abbildung 6.1: Entladekurve Akku

Das Diagramm (siehe Abbildung 6.1) beinhaltet die Achsen für die Zeit in Stunden (x-Achse) und den Spannungswert (y-Achse). Normalerweise würden zwei Graphen in dem Diagramm zu sehen sein. Jeweils für die Spannungswerte vom Akku und der Solaranlage.

Aufgrund der schlechten Wetterbedingungen ist die Leistung der Solaranlage nicht ausreichend um den Akku zu laden. Deswegen ist der Spannungswert der Solaranlage gleich mit dem Spannungswert des Akkus. Das bedeutet es fließt kein Strom in den Akku, da kein Potentialunterschied zwischen Akku und Solaranlage herrscht. Das Diagramm zeigt lediglich die Entladekurve des Akkus. Der Test startet um 08:48 Uhr und endet am 24.12.2017 um 10:18 Uhr. Anfangs ist die Spannung bei 12,60 V. Durch den Tiefentladeschutz des Solarladereglers ist die Endspannung bei 11,70 V. Im Anhang unter dem Punkt 8.7 zeigt die Tabelle 8.1 die Messungen im Halbstundentakt. Der Akku hat somit rund 25 Stunden durchgehalten. Die langen Nächte im Winter, die eine Dauer bis zu 16 Stunden haben, überbrückt der Akku.

Für dieses Ergebnis kann man jetzt schon sagen, dass die Solaranlage mit dem Restlicht bei komplett bewölktem Wetter keine ausreichende Energie erzielen kann um einen 12 Volt Akku zu laden.

## 6.2 Testlauf mit Algorithmus

Der Test zeigt die Solaranlage mit allen Komponenten. Dieser findet am 26.12.2017 statt. Der Sonnenaufgang ist um 08:25 Uhr und der Sonnenuntergang ist um 16:23 Uhr. Die Wettervorhersage besagt auch hier ein bewölktetes Wetter aber mit der Chance auf ein wenig Sonnenschein.

Das System ist wie folgt aufgebaut:

- Die Solaranlage zeigt nach Süden.
- Der Neigungswinkel ist auf 0 Grad eingestellt (senkrecht).
- Die Schrittmotoren sind am Raspberry Pi angeschlossen.
- Die Energieüberwachung ist am Raspberry Pi angeschlossen.
- Der Akku ist aufgeladen (12.65 V).

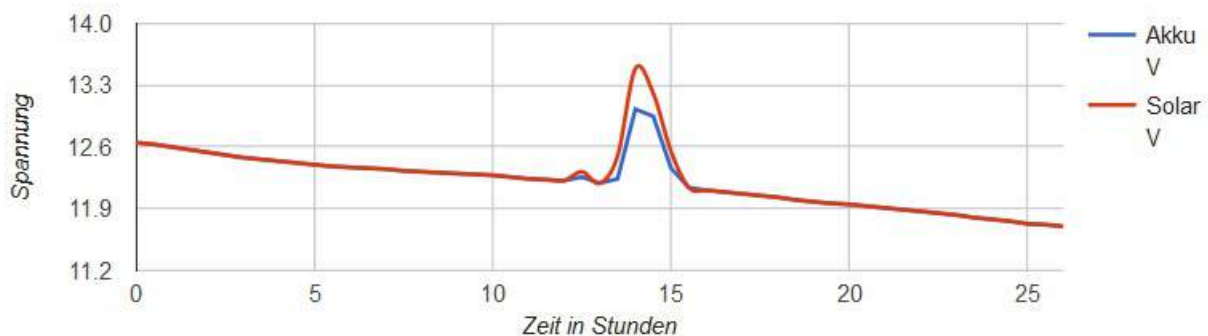


Abbildung 6.2: Entladekurve Akku mit Einfluss der Solaranlage

Das Diagramm (siehe Abbildung 6.2) beinhaltet auch hier die zwei Achsen mit der Zeit in Stunden (x-Achse) und der Spannung (y-Achse). Diesmal sind zwei Graphen in dem Diagramm zu sehen, da glücklicherweise die Sonne kurzzeitig schien. Der Test startet um 00:00 Uhr und geht bis 02:00 Uhr des 27.12.2017. Die Energieüberwachung bricht den Test vor dem Tiefentladeschutz ab. Da der Test mit einer höheren Akkuspannung startet, liegt die Testdauer bei ungefähr 26 Stunden. Die angeschlossenen Schrittmotoren verringern in diesem Fall vielleicht nur im geringen die Testdauer. Bei Aktivierung der Schrittmotoren steigt der Stromverbrauch des Raspberry Pis. Bei der Drehung sind dies ungefähr eine Sekunde und im Falle des Neigungswinkels braucht der Schrittmotor ungefähr 5 Sekunden. Der Stromverbrauch beläuft sich hierbei auf maximal 0,631 A bei der Benutzung der Schrittmotoren. Anfangs ist die Ausgangssituation wie beim Test ohne Algorithmus. Die Solaranlage kann mit dem Restlicht den Akku nicht laden. Um die Mittagszeit bricht die Wolkendecke ein wenig auf und ein paar Sonnenstrahlen kommen durch. Um 14:05 sind nur noch einzelne

Wolken am Himmel und die Sonne kann ihr Potential entfalten. Dabei steigt der Spannungswert der Solaranlage auf maximal 13,49 V (siehe im Anhang Tabelle 8.2) gefolgt von dem erhöhten Spannungswert des Akkus. Somit ist ein Potentialunterschied erkennbar und Strom fließt. Die gemessene Stromstärke beläuft sich auf maximal 0,602 A von der Solaranlage. Da der Solarladeregler zwischen Solaranlage und Akku geschaltet ist, zieht dieser ungefähr 4 mA für den Eigenbedarf. Dann verbraucht auch der Raspberry Pi ungefähr 0,265 A. In den Akku fließen letztendlich 0,333 A. Im Bezug auf einen leeren Akku wäre dieser in ungefähr 19,5 Stunden voll. Diese Berechnung ist natürlich nur theoretisch zu verstehen, da über den Tag verteilt, die angegebene Stromstärke stark variieren kann. Speziell morgens und abends sind die Sonnenstrahlen der Sonne schwächer als mittags und daher ist die oben beschriebene Stromstärke als Durchschnitt zu betrachten. Im Winter könnte dieser Wert den Akku nur bedingt aufladen, da die Tage sehr kurz sind und die Stromstärke nicht ausreichend ist. In den Sommermonaten macht dies schon eher Sinn. Hier sind die Tage länger und das Wetter ist bekanntlich freundlicher.

### 6.3 Testlauf der Energieüberwachung

Der Test der Energieüberwachung lehnt sich an beide zuvor getesteten Testläufe.

Erster Testlauf:

Das Skript `rc.local` startet das Programm der Energieüberwachung um 08:48 Uhr. Am Anfang benutzt das Programm noch die alte Methode der Abfrage für die Abschaltung des Einplatinencomputers. Diese fragt den aktuellen Spannungswert ab und wenn dieser unterhalb der angegebenen Grenze liegt, fährt der Einplatinencomputer herunter. Das erscheint anfangs logisch, aber im Falle des AD-Wandlers erscheinen manchmal während des Messens unterschiedliche Werte. Das können willkürliche Werte sein. Wenn dieser falsche Wert unter der Grenze liegt schaltet sich der Einplatinencomputer natürlich aus. Genau dieser Fall passiert im ersten Testlauf. Nach ungefähr zwei Stunden Testzeit schaltet sich das System ab, obwohl der Akku noch vollgeladen ist. Nach diesem Test bedarf es einer Verbesserung der Energieüberwachung beziehungsweise der Abbruchbedingung.

Zweiter Testlauf:

Mit der verbesserten Abbruchbedingung (siehe Programm 5.3) ist es nun nicht mehr möglich, dass das Programm vorzeitig abbricht. Die drei Schleifendurchgänge verhindern dies. Der Test startet um 00:00 Uhr durch das automatische Aufrufen des Programms durch die `rc.local`. Die Energieüberwachung bricht das Programm um 01:47 Uhr am Folgetag ab und fährt den Einplatinencomputer herunter.

## 6.4 Anschlussfehler des Spannungsreglers (DC-DC-Wandler)

In diesem Teil ist es wichtig darauf aufmerksam zu machen, dass eine falsche Benutzung des Spannungsregler im System, gravierende Folgen für die Schaltungen haben kann.



Abbildung 6.3: Spannungsregler Rückseite

Die Fließrichtung des Stroms ist in Abbildung 6.3 auf der Rückseite des Spannungsreglers ersichtlich. Die Aufschriften IN und OUT mit Plus und Minus und der Pfeil verdeutlichen dies. Auf der Vorderseite sind nur die Aufschriften abgebildet. Das bedeutet im Falle des Systems, dass der Solarladeregler die Energie vom Akku von der rechten Seite zubringt (IN) und der Raspberry Pi am anderen Ende (OUT) die verringerte Spannung aufnimmt. Nun kann es aber auch wegen einer Unachtsamkeit passieren, dass der Spannungsregler in die falsche Richtung zeigt und somit falsch angeschlossen ist. Dadurch ist die Funktion des Spannungsreglers nicht mehr gewährleistet und somit fließt ein zu hoher Ladestrom sowie eine zu hohe Spannung durch den Spannungsregler und auch zum Raspberry Pi. Die Folge ist, dass die Schaltkreise beziehungsweise die Komponenten auf dem Spannungsregler den zu hohen Ladestrom nicht verkraften und infolgedessen kaputt gehen. Am Eingang des Raspberry Pis erscheint ein kleiner Blitz, gefolgt von einer Rauchwolke. Das Endresultat entspricht daraufhin einem defekten Spannungsregler und Raspberry Pi.

## 7 Diskussion

Im abschließenden Kapitel geht es nun um eine zusammenfassende Bewertung des entwickelten Systems und dazu noch die persönliche Einschätzung des Projektes. Des Weiteren folgt der Ausblick für zukünftige Arbeiten im Bezug auf das System.

### 7.1 Zusammenfassende Bewertung

Das entwickelte System zeigt einen Weg wie man mit mehreren einzeln zusammengeschalteten Solarmodulen im Verbund mit einem Akku einen Einplatinencomputer betreibt und gleichzeitig die Energie des Akkus überwacht. Zuallererst ist hier zu sagen, dass die Nutzung der Sonnenenergie nicht zuverlässig ist. Beim Versuch ein gutes beziehungsweise ein positives Testergebnis zu bekommen, scheiterte es an den schlechten Wetterbedingungen während der Testphasen. Ein positives Testergebnis wäre zum Beispiel der Beweis für die Aufladung des Akkus durch die Solarmodule am Tage. Daher kann man nur diesen Punkt theoretisch abhandeln. Die Wintermonate sind für das Testen der Solarmodule ein wenig ungeeignet. Doch eines kann man hier auch gleich erkennen, dass der Akku für diese Zeiten eine ausreichende Kapazität haben muss. Soweit zu dem Wetter. Das System an sich soll zeigen, inwiefern man mit einem Algorithmus und einer Konstruktion für die Solarmodule, die Sonnenenergie am besten ausnutzt. Dabei helfen Programme und zwei Schrittmotoren. Diese Komponenten, plus die Komponenten der Energieüberwachung mussten in ein einziges System integriert werden. Dazu galt es alle Komponenten auszusuchen und diese einzeln auf ihre Funktionalität zu testen. Speziell das Gestell war eine Herausforderung und nahm viel Zeit in Anspruch. Hier war es wichtig die verbauten Materialien und das Zusammenspiel derer in eine kompakte Lösung zu verpacken. Am Ende entstand ein Gestell, welches auf zehn parallelgeschaltete Solarmodule zugeschnitten ist. Die Testergebnisse zeigen, dass die verbesserte Konstruktion auch bei schlechten Wetterverhältnissen keine ausreichende Energie liefern kann. Damit ist das Ziel für ein rund um die Uhr Betrieb fehlgeschlagen. Der Akku gewährleistet aber einen 24 Stunden Betrieb auch ohne Eingriff der Solaranlage. Die Energieüberwachung des Systems arbeitet hingegen unproblematisch und führt den Befehl für das Herunterfahren des Einplatinencomputers aus, wenn der Ladezustand des Akkus sich dem Ende neigt. Die Schaltung mit dem AD-Wandler funktioniert hierfür einwandfrei und liefert die nötigen Daten. Das Gestell hält alle Solarmodule in der Position, die von dem Programm vorgegeben sind. Der Aufbau an sich ist rein prototypenbasiert und stellt für den alltäglichen Nutzung noch einige Verbesserungen dar. Mögliche Verbesserungen sollten auf jeden Fall im Bereich der Mechanik und der Schrittmotoren vorgenommen werden. Die Schrittmotoren sind für die Darstellung der Funktion fürs Erste gut genug. Bei weiterer Bearbeitung sollten aber bessere und stärkere Schrittmotoren zum Einsatz kommen. Die Mechanik des Gestells hat ein paar

Schwächen im Falle von der Verbindung des Standfußes und dem Gestell. Die Auflagefläche ist zu klein um das große Gestell zu tragen. Das macht sich bei windigem Wetter bemerkbar, indem das Gestell wackelt. Allgemein ist die Nutzung nur bei gutem Wetter zu empfehlen, da dort auch die Solaranlage richtig arbeiten kann.

Vom persönlichen Standpunkt gesehen, ist die Ausnutzung der Solarenergie beziehungsweise der erneuerbaren Energien der richtige Schritt zu einem umweltfreundlichen Denken. Allein schon die Tatsache, dass diese Energie im Grunde genommen kostenlos zur Verfügung steht, wenn auch nicht konstant und zuverlässig, lässt doch den Gedanken auf mehr Forschungsdrang in diesem Gebiet erwecken.

## **7.2 Ausblick**

Das Wichtigste ist, dass im Bereich der Solarmodule weiter Forschung betrieben wird. Denn in erster Linie sind diese für die Energieerzeugung zuständig. Mit einer effizienten Nutzung eines intelligenten Systems kann auch hier noch mehr herausgeholt werden. Dabei wäre es interessant, für die Verfolgung des Sonnenverlaufes, Lichtsensoren einzusetzen. Hier könnte man schauen, ob das hier entwickelte System oder ein mit Lichtsensoren gesteuertes System bessere Erträge erzielen kann. Weiterhin ist darüber nachzudenken, ob eine zusätzliche Energieerzeugung per Windkraft dem System noch Energie liefern könnte. Diese wäre zum Beispiel über Nacht oder bei schlechten Wetterverhältnissen eine Alternative.



# Literaturverzeichnis

- [1] Raspberry pi. [https://en.wikipedia.org/wiki/Raspberry\\_Pi](https://en.wikipedia.org/wiki/Raspberry_Pi). Eingesehen am 24.10.2017.
- [2] Technik aus der weltraumforschung – so funktionieren solarzellen. <https://photovoltaiksolarstrom.com/solarzelle-funktion/>. Eingesehen am 23.10.2017.
- [3] Spannung und stromstärke einer solarzelle. <http://www.solar.lucycity.de/index.php/spannung-und-stromstaerke>. Eingesehen am 29.12.2017.
- [4] Photovoltaik: Die funktion. <http://www.solarladen.de/photovoltaik-funktion>. Eingesehen am 23.10.2017.
- [5] Ausrichtung der photovoltaikanlage. <http://www.photovoltaik.org/ausrichtung>. Eingesehen am 24.10.2017.
- [6] Raspberry pi: Stromverbrauch messen. <https://www.elektronik-kompodium.de/sites/raspberry-pi/1910071.htm>. Eingesehen am 25.10.2017.
- [7] Raspberry pi: Grundlagen der energieverorgung / stromversorgung. <https://www.elektronik-kompodium.de/sites/raspberry-pi/1912111.htm>. Eingesehen am 25.10.2017.
- [8] Universal serial bus. [https://de.wikipedia.org/wiki/Universal\\_Serial\\_Bus](https://de.wikipedia.org/wiki/Universal_Serial_Bus). Eingesehen am 25.10.2017.
- [9] Elektrische leistung. [https://de.wikipedia.org/wiki/Elektrische\\_Leistung](https://de.wikipedia.org/wiki/Elektrische_Leistung). Eingesehen am 25.10.2017.
- [10] Der mit dem 64-bit-kernel tanzt. <https://www.golem.de/news/raspberry-pi-der-mit-dem-64-bit-kernel-tanzt-1611-124475.html>. Eingesehen am 26.10.2017.
- [11] Raspberry pi: Gpio - general purpose input output. <http://www.elektronik-kompodium.de/sites/raspberry-pi/2002191.htm>. Eingesehen am 05.11.2017.
- [12] Raspberry pi pinout. <https://de.pinout.xyz/#>. Eingesehen am 26.10.2017.
- [13] Introducing raspberry pi hats. <https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/>. Eingesehen am 27.10.2017.

- [14] Schrittmotoren. <http://www.tuf-ev.de/workshop/schrittmotor/typen.htm>. Eingesehen am 30.12.2017.
- [15] Serial peripheral interface (spi). <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>. Eingesehen am 30.12.2017.
- [16] Jean Pütz. *Einführung in die Elektronik*, chapter Grundlagen der Elektronik, pages 16–18. vgs Verlagsgesellschaft Schulfernsehen, 15., aufl. edition, 1981.
- [17] Jean Pütz. *Einführung in die Elektronik*, chapter Grundlagen der Elektronik, pages 18–19. vgs Verlagsgesellschaft Schulfernsehen, 15., aufl. edition, 1981.
- [18] Jean Pütz. *Einführung in die Elektronik*, chapter Grundlagen der Elektronik, page 19. vgs Verlagsgesellschaft Schulfernsehen, 15., aufl. edition, 1981.
- [19] Minus, masse, erde - spannende grundlagen. <http://www.dieelektronikerseite.de/Lectons/Minus,%20Masse,%20Erde%20-%20Spannende%20Grundlagen.htm>. Eingesehen am 27.10.2017.
- [20] Konzepte zur verschaltung des wechselrichters. <http://www.solaranlage.eu/photovoltaik/technik-komponenten/wechselrichter/konzepte-zur-verschaltung>. Eingesehen am 11.11.2017.
- [21] Solar powered raspberry pi. <http://www.rustynailworkshop.com/archives/6>. Eingesehen am 01.11.2017.
- [22] Spannungsmessung mit ad-wandler am raspberry pi. <https://www.direcs.de/2017/03/spannungsmessung-mit-ad-wandler-am-raspberry-pi/>. Eingesehen am 03.11.2017.
- [23] Datenerfassungs-ic - analog-digital-wandler (adc) microchip technology mcp3008-i/p extern. <https://www.conrad.de/de/datenerfassungs-ic-analog-digital-wandler-adc-microchip-technology-mcp3008-ip-ext.html>. Eingesehen am 13.11.2017.
- [24] Bedienungsanleitung. [http://www.produktinfo.conrad.com/datenblaetter/100000-124999/110717-an-01-de-STECA\\_SOLSUM\\_10\\_10\\_F.pdf](http://www.produktinfo.conrad.com/datenblaetter/100000-124999/110717-an-01-de-STECA_SOLSUM_10_10_F.pdf). Eingesehen am 21.12.2017.
- [25] [tutorial | raspberry pi] schrittmotor / stepper motor. <https://www.youtube.com/watch?v=4fHL6BpJrC4>. Eingesehen am 15.12.2017.
- [26] Unix-shell. <https://de.wikipedia.org/wiki/Unix-Shell>. Eingesehen am 25.10.2017.
- [27] Raspberry pi: Programme beim systemstart ausführen. [http://www.netzmafia.de/skripten/hardware/RasPi/RasPi\\_Auto.html](http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_Auto.html). Eingesehen am 25.10.2017.

- [28] Scheduling tasks with cron. <https://www.raspberrypi.org/documentation/linux/usage/cron.md>. Eingesehen am 25.11.2017.
- [29] Python (programmiersprache). [https://de.wikipedia.org/wiki/Python\\_\(Programmiersprache\)](https://de.wikipedia.org/wiki/Python_(Programmiersprache)). Eingesehen am 27.10.2017.
- [30] Raspbian. <https://www.raspberrypi.org/downloads/raspbian/>. Eingesehen am 23.10.2017.
- [31] Initiale ip-adresse. [http://www.netzmafia.de/skripten/hardware/RasPi/RasPi\\_Network.html](http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_Network.html). Eingesehen am 24.10.2017.
- [32] Drei wege das wlan auf einem raspberry pi einzurichten. <https://linuxundich.de/raspberry-pi/drei-wege-das-wlan-auf-einem-raspberry-pi-einzurichten/>. Eingesehen am 24.10.2017.
- [33] Putty. <https://de.wikipedia.org/wiki/PuTTY>. Eingesehen am 23.10.2017.
- [34] Secure shell. [https://de.wikipedia.org/wiki/Secure\\_Shell](https://de.wikipedia.org/wiki/Secure_Shell). Eingesehen am 23.10.2017.
- [35] Raspberry pi ssh zugriff einrichten via putty (windows). <https://tutorials-raspberrypi.de/raspberry-pi-ssh-windows-zugriff-putty/>. Eingesehen am 23.10.2017.
- [36] 10. software aktualisieren. <https://www.elektronik-kompendium.de/sites/raspberry-pi/1906291.htm>. Eingesehen am 24.10.2017.
- [37] Leerlauf- und klemmenspannung. <https://www.lernhelfer.de/schuelerlexikon/physik/artikel/leerlauf-und-klemmenspannung>. Eingesehen am 31.12.2017.
- [38] Schaltungen von spannungsquellen). [http://www.hobby-bastelecke.de/grundlagen/spannungsquellen\\_schaltungen.htm](http://www.hobby-bastelecke.de/grundlagen/spannungsquellen_schaltungen.htm). Eingesehen am 10.12.2017.
- [39] Solar-laderegler 12 v, 24 v 10 a steca solsum 10.10 f. <https://www.conrad.de/de/solar-laderegler-12-v-24-v-10-a-steca-solsum-1010-f-110717.html>. Eingesehen am 11.12.2017.
- [40] Vorbereitung. <https://tutorials-raspberrypi.de/raspberry-pi-mcp3008-analoge-signale-auslesen/>. Eingesehen am 12.12.2017.
- [41] Spidev documentation. [http://tightdev.net/SpiDev\\_Doc.pdf](http://tightdev.net/SpiDev_Doc.pdf). Eingesehen am 31.12.2017.
- [42] Send email from raspberry pi using python script and gmail smtp. <https://iotbytes.wordpress.com/programmatically-send-e-mail-from-raspberry-pi-using-python-and-gmail/>. Eingesehen am 01.11.2017.

- [43] Raspberry pi schrittmotor ansteuern mit l293d / uln2003a. <https://tutorials-raspberrypi.de/raspberry-pi-schrittmotor-steuerung-l293d-uln2003a/>. Eingesehen am 14.12.2017.
- [44] <http://rolfrost.de/sunservice.html?year=2017;month=12;day=23;long=8.65;lat=50.11;tz=1;dst=0;obj=SUN>. Eingesehen am 23.12.2017.

# 8 Anhang

## 8.1 emailHandler.py

Programm 8.1: emailHandler.py [42]

```
1  #!/usr/bin/python
2  # coding=utf-8
3
4  import ConfigParser , inspect , os
5  import smtplib
6  from email.mime.multipart import MIMEMultipart
7  from email.mime.text import MIMEText
8
9
10 #Form the absolute path for the settings.ini file
11 settings_Dir = os.path.dirname(os.path.abspath(inspect.getfile
12 (inspect.currentframe()))))
13 settings_File_Path = os.path.join(settings_Dir , 'settings.ini')
14
15
16 ===== GET SETTINGS FROM EMAIL SECTION IN settings.ini
17 FILE =====
18 def read_Email_Settings():
19
20     try:
21         config = ConfigParser.ConfigParser()
22         config.optionxform=str #By default config returns keys
23         #from Settings file in lower case. This line preserves the case
24
25         config.read(settings_File_Path)
26
27         global FROM_ADD
28         global USERNAME
29         global PASSWORD
30         global SMTP_SERVER
31         global SMTP_PORT
32
```

---

```

33     SMTP_SERVER = config.get("EMAIL", "SMTP_ADD")
34     SMTP_PORT = config.get("EMAIL", "SMTP_PORT")
35     FROM_ADD = config.get("EMAIL", "FROM_ADD")
36     USERNAME = config.get("EMAIL", "USERNAME")
37     PASSWORD = config.get("EMAIL", "PASSWORD")
38
39     except Exception as error_msg:
40         print "Error while trying to read SMTP/EMAIL Settings."
41         print {"Error" : str(error_msg)}
42 =====
43
44 read_Email_Settings()
45
46
47 class Class_eMail():
48
49     def __init__(self):
50         self.session = smtplib.SMTP_SSL(SMTP_SERVER, SMTP_PORT)
51         self.session.ehlo()
52         self.session.login(USERNAME, PASSWORD)
53
54
55     def initialise_Mail_Body(self, To_Add, Subject):
56         #Prepare Mail Body
57         Mail_Body = MIMEMultipart()
58         Mail_Body['From'] = FROM_ADD
59         Mail_Body['To'] = To_Add
60         Mail_Body['Subject'] = Subject
61         return Mail_Body
62
63
64     #Call this to send plain text emails.
65     def send_Text_Mail(self, To_Add, Subject, txtMessage):
66         Mail_Body = self.initialise_Mail_Body(To_Add, Subject)
67         #Attach Mail Message
68         Mail_Msg = MIMEText(txtMessage, 'plain')
69         Mail_Body.attach(Mail_Msg)
70         #Send Mail
71         self.session.sendmail(FROM_ADD, [To_Add], Mail_Body.as_string())
72
73
74     #Call this to send HTML emails.
75     def send_HTML_Mail(self, To_Add, Subject, htmlMessage):

```

```
76         Mail_Body = self.initialise_Mail_Body(To_Add, Subject)
77         #Attach Mail Message
78         Mail_Msg = MIMEText(htmlMessage, 'html')
79         Mail_Body.attach(Mail_Msg)
80         #Send Mail
81         self.session.sendmail(FROM_ADD, [To_Add], Mail_Body.as_string())
82
83
84     def __del__(self):
85         self.session.close()
86     del self.session
```

## 8.2 stepperpitch.py

Programm 8.2: stepperpitch.py [25]

```

1  #!/usr/bin/python
2  #coding=utf-8
3
4  import time
5  import RPi.GPIO as GPIO
6
7  GPIO.setmode(GPIO.BOARD)
8  GPIO.setwarnings(False)
9
10 A = 12
11 B = 16
12 C = 18
13 D = 22
14
15 GPIO.setup(A, GPIO.OUT)
16 GPIO.setup(B, GPIO.OUT)
17 GPIO.setup(C, GPIO.OUT)
18 GPIO.setup(D, GPIO.OUT)
19
20
21 # Schemata acht Halbschritte
22 """
23           1   2   3   4   5   6   7   8
24
25 Pin1    x   x
26 Pin2          x   x   x
27 Pin3                x   x   x
28 Pin4                        x   x   x
29
30 """
31
32 def GPIO_SETUP(a,b,c,d):
33     GPIO.output(A, a)
34     GPIO.output(B, b)
35     GPIO.output(C, c)
36     GPIO.output(D, d)
37     time.sleep(0.001)
38
39 def RIGHT_TURN(deg):

```



```
40
41     full_circle = 512.0
42     degree = full_circle/360*deg
43     GPIO_SETUP(0,0,0,0)
44
45     while degree > 0.0:
46         GPIO_SETUP(1,0,0,0)
47         GPIO_SETUP(1,1,0,0)
48         GPIO_SETUP(0,1,0,0)
49         GPIO_SETUP(0,1,1,0)
50         GPIO_SETUP(0,0,1,0)
51         GPIO_SETUP(0,0,1,1)
52         GPIO_SETUP(0,0,0,1)
53         GPIO_SETUP(1,0,0,1)
54         degree -= 1
55
56 def LEFT_TURN(deg):
57
58     full_circle = 512.0
59     degree = full_circle/360*deg
60     GPIO_SETUP(0,0,0,0)
61
62     while degree > 0.0:
63         GPIO_SETUP(1,0,0,1)
64         GPIO_SETUP(0,0,0,1)
65         GPIO_SETUP(0,0,1,1)
66         GPIO_SETUP(0,0,1,0)
67         GPIO_SETUP(0,1,1,0)
68         GPIO_SETUP(0,1,0,0)
69         GPIO_SETUP(1,1,0,0)
70         GPIO_SETUP(1,0,0,0)
71         degree -= 1
72
73 ##MAIN #####
74 def main(timeSun):
75     timeSun = timeSun * 60
76     percentageValue10 = timeSun/100*10
77     percentageValue60 = timeSun/100*60
78     percentageValue20 = timeSun/100*20
79
80     time.sleep(percentageValue10.seconds)#10%
81     LEFT_TURN(25*40.6) #25 Grad
82     GPIO_SETUP(0,0,0,0)
```

```
83     time.sleep(percentageValue10.seconds)#10%
84     LEFT_TURN(10*40.6) #35 Grad
85     GPIO_SETUP(0,0,0,0)
86     time.sleep(percentageValue60.seconds) #60%
87     RIGHT_TURN(10*40.6) #25 Grad
88     GPIO_SETUP(0,0,0,0)
89     time.sleep(percentageValue20.seconds) #20%
90     RIGHT_TURN(25*40.6) #0 Grad
91     GPIO_SETUP(0,0,0,0)
```

## 8.3 stepperyaw.py

Programm 8.3: stepperyaw.py [25]

```

1  #!/usr/bin/python
2  #coding=utf-8
3
4  import multiprocessing
5  import time
6  import sys
7  sys.path.append( '/home/pi/python_script_steppermotor' )
8  import stepperpitch
9  from datetime import datetime
10 from datetime import timedelta
11 import RPi.GPIO as GPIO
12
13 GPIO.setmode( GPIO.BOARD )
14 GPIO.setwarnings( False )
15
16 A = 7
17 B = 11
18 C = 13
19 D = 15
20
21 GPIO.setup( A, GPIO.OUT )
22 GPIO.setup( B, GPIO.OUT )
23 GPIO.setup( C, GPIO.OUT )
24 GPIO.setup( D, GPIO.OUT )
25
26 def stepperYaw( degree , degreePerTime ):
27     var = 0
28     while var < degree: # Winkel
29         time.sleep( degreePerTime.seconds )
30         LEFT_TURN( 3.5 )
31         GPIO_SETUP( 0 , 0 , 0 , 0 )
32         var += 1
33
34     RIGHT_TURN( degree / 2 * 3.5 ) #set back to 180 degrees (south)
35     GPIO_SETUP( 0 , 0 , 0 , 0 )
36
37 # Schemata acht Halbschritte
38 """
39     1   2   3   4   5   6   7   8

```

```
40
41 Pin1   x   x                               x
42 Pin2       x   x   x
43 Pin3           x   x   x
44 Pin4               x   x   x
45
46 " " "
47
48 def GPIO_SETUP(a,b,c,d):
49     GPIO.output(A, a)
50     GPIO.output(B, b)
51     GPIO.output(C, c)
52     GPIO.output(D, d)
53     time.sleep(0.001)
54
55 def RIGHT_TURN(deg):
56
57     full_circle = 512.0
58     degree = full_circle/360*deg
59     GPIO_SETUP(0,0,0,0)
60
61     while degree > 0.0:
62         GPIO_SETUP(1,0,0,0)
63         GPIO_SETUP(1,1,0,0)
64         GPIO_SETUP(0,1,0,0)
65         GPIO_SETUP(0,1,1,0)
66         GPIO_SETUP(0,0,1,0)
67         GPIO_SETUP(0,0,1,1)
68         GPIO_SETUP(0,0,0,1)
69         GPIO_SETUP(1,0,0,1)
70         degree -= 1
71
72 def LEFT_TURN(deg):
73
74     full_circle = 512.0
75     degree = full_circle/360*deg
76     GPIO_SETUP(0,0,0,0)
77
78     while degree > 0.0:
79         GPIO_SETUP(1,0,0,1)
80         GPIO_SETUP(0,0,0,1)
81         GPIO_SETUP(0,0,1,1)
82         GPIO_SETUP(0,0,1,0)
```

```
83     GPIO_SETUP(0,1,1,0)
84     GPIO_SETUP(0,1,0,0)
85     GPIO_SETUP(1,1,0,0)
86     GPIO_SETUP(1,0,0,0)
87     degree -= 1
88
89 #MAIN #####
90
91 def main(args):
92     currentTimeHour = datetime.time(datetime.now()).hour
93     currentTimeMinute = datetime.time(datetime.now()).minute
94     print currentTimeHour
95     print currentTimeMinute
96     sunriseHours = args[0]
97     sunriseMinutes = args[1]
98     timedifferenceMinutes = args[2]
99     degree = args[3]
100
101     # set the device to starting position
102     RIGHT_TURN(degree/2*3.5)
103     GPIO_SETUP(0,0,0,0)
104
105     timeToSleep = (timedelta(hours=(int)(sunriseHours), minutes=(int)
106 (sunriseMinutes))-timedelta(hours=(int)(currentTimeHour),
107
108     print timeToSleep.seconds
109     time.sleep(timeToSleep.seconds)
110
111     degreePerTime = timedifferenceMinutes*60 / degree
112
113     fct = multiprocessing.Process(target = stepperpitch.main,
114 args=(timedifferenceMinutes,))
115     fct1 = multiprocessing:Process(target = stepperYaw, args=(degree,
116 degreePerTime,))
117
118     fct.start()
119     fct1.start()
```

## 8.4 sunsetsunrise.py

Programm 8.4: sunsetsunrise.py

```

1  #!/usr/bin/python
2  # coding=utf-8
3
4  import sys
5  sys.path.append('/home/pi/python_script_steppermotor')
6  import stepperyaw
7  import urllib
8  import json
9  import datetime
10 import csv
11 import math
12 from datetime import timedelta
13
14 SonnenaufgangWinkel = 0
15
16 # Funktion um berechnen der Sonnendaten
17 def calculation(hours, minutes):
18     # 50.119609, 8.564896
19     # formulars
20     lat = 50.1196
21     long = 8.5648
22
23     tageszahl = currentDate.timetuple().tm_yday
24     K = math.pi / 180 # constant 0.01745 (zur Umrechnung von Grad- in
Bogenmass)
25
26     deklin = -23.45 * math.cos(K * 360 * (tageszahl + 10) / 365)
27
28     zeitgleichung = 60 * (-0.171 * math.sin(0.0337 * tageszahl + 0.465)
29 - 0.1299 * math.sin(0.01787 * tageszahl - 0.168))
30
31     stundenwinkel = 15 * ((int)(hours) + (int)(minutes) / 60 - (15.0
32 long) / 15.0 - 12 + zeitgleichung / 60)
33
34     x = math.sin(K * lat) * math.sin(K * deklin) + math.cos(K * lat) *
35
36     height = math.asin(x) / K
37
38     y = -(math.sin(K * lat) * math.sin(K * height) - math.sin(K * deklin))

```

```
39
40     azimut = math.acos(y) / K
41     return azimut
42
43
44     currentDate = datetime.date.today()
45
46     day = currentDate.day
47     month = currentDate.month
48     year = currentDate.year
49
50     url = "http://rolfrost.de/sunservice.html?year=%d;month=%d;
51     day=%d;long=8.5648;lat=50.1196;tz=1;dst=" %(year, month, day)
52
53     response = urllib.urlopen(url)
54     data = response.read().decode('utf-8')
55     jsonData = json.loads(data)
56
57     sunrise = jsonData['sunrise']
58     sunset = jsonData['sunset']
59
60     sunriseHours = sunrise[0:2]
61     sunriseMinutes = sunrise[3:5]
62
63     sunsetHours = sunset[0:2]
64     sunsetMinutes = sunset[3:5]
65
66     timedifference = (timedelta(hours=(int)(sunsetHours), minutes=(int)
67         minutes=(int)(sunriseMinutes)))
68     timedifferenceMinutes = timedifference.seconds / 60.0
69     timedifferenceHours = timedifferenceMinutes / 60.0
70
71     # calculation:
72     SonnenaufgangWinkel = calculation(sunriseHours, sunriseMinutes)
73     Winkel = 180 - SonnenaufgangWinkel
74     Winkel = Winkel * 2
75
76     # write to the csv file:
77     with open(' /home/pi/sun_algorithm/SunData.csv', 'wb') as csvfile:
78         spamwriter = csv.writer(csvfile, delimiter=' ',
79             quotechar='|', quoting=csv.QUOTE_MINIMAL)
80         spamwriter.writerow([sunriseHours]+[sunriseMinutes]+[sunsetHours]+
81             [sunsetMinutes]+[timedifferenceHours]+
```

```
82     [timedifferenceMinutes]+[Winkel])
83
84 # execute stepper
85 args = [sunriseHours,sunriseMinutes,timedifferenceMinutes,Winkel]
86 stepperyaw.main(args)
```





## 8.6 Abmessungen zum Aufbau Gestell

### 8.6.1 Solarmodul Rahmen Zusatzprofil

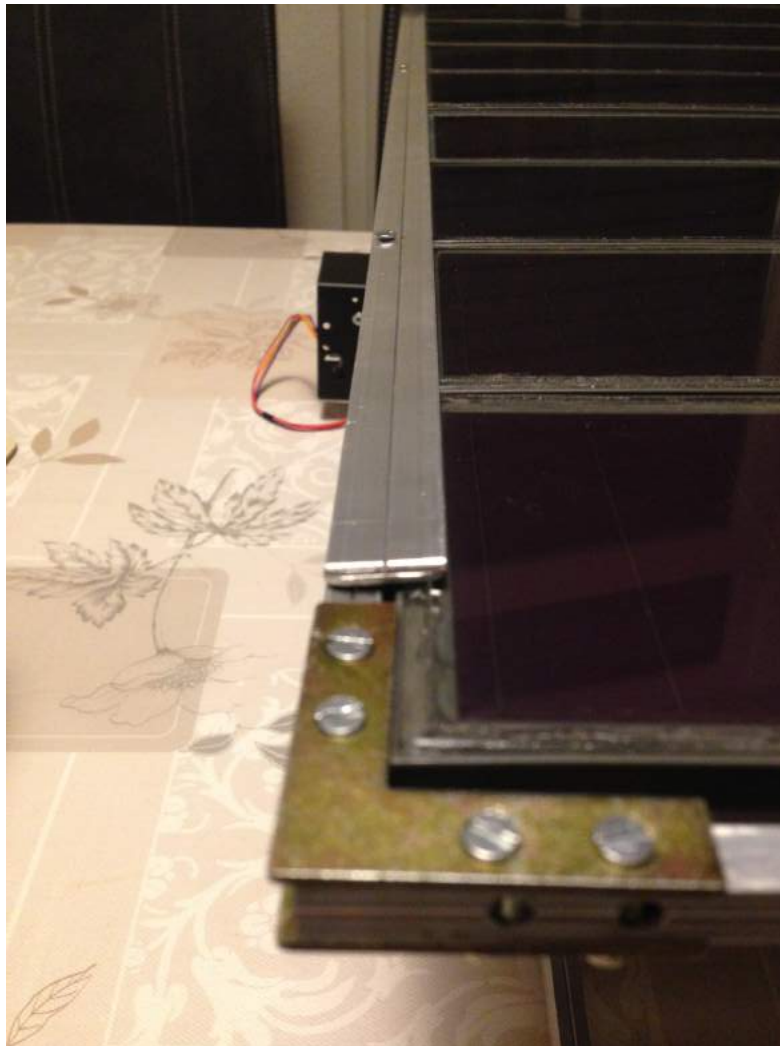


Abbildung 8.2: Zusatzprofil für Solarmodule

### 8.6.2 Solarmodul Rahmen Abmessung

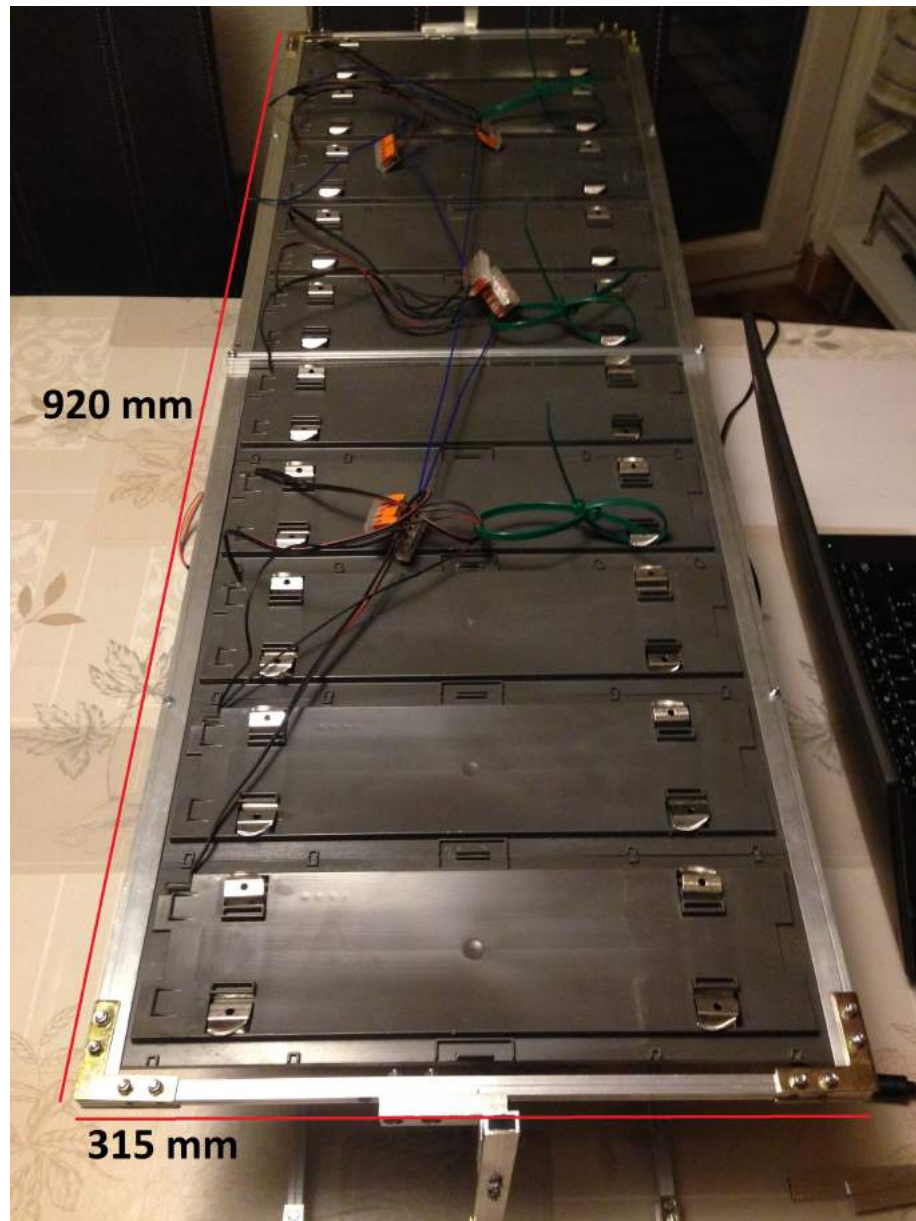


Abbildung 8.3: Solarmodul Rahmen Abmessung

### 8.6.3 Gestell Stirnseite Abmessung

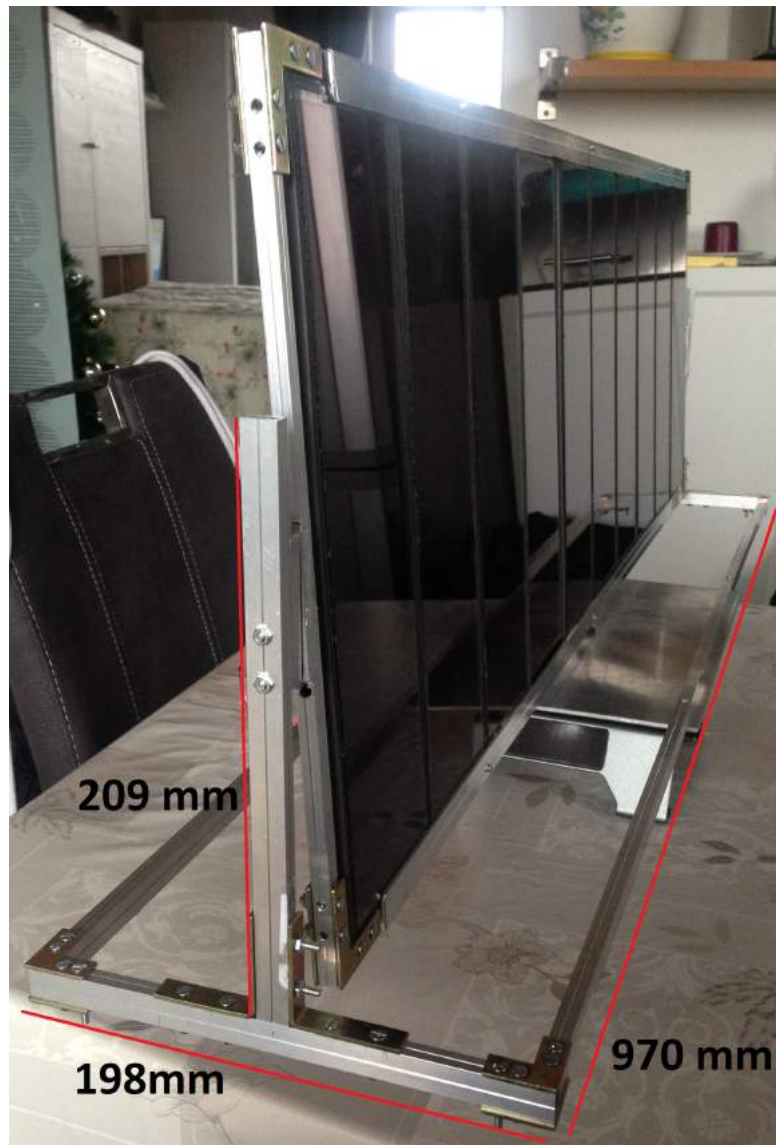


Abbildung 8.4: Stütze mit zweitem Rahmen

#### 8.6.4 Zapfenband



Abbildung 8.5: Zapfenband

### 8.6.5 Zapfenband Abmessung

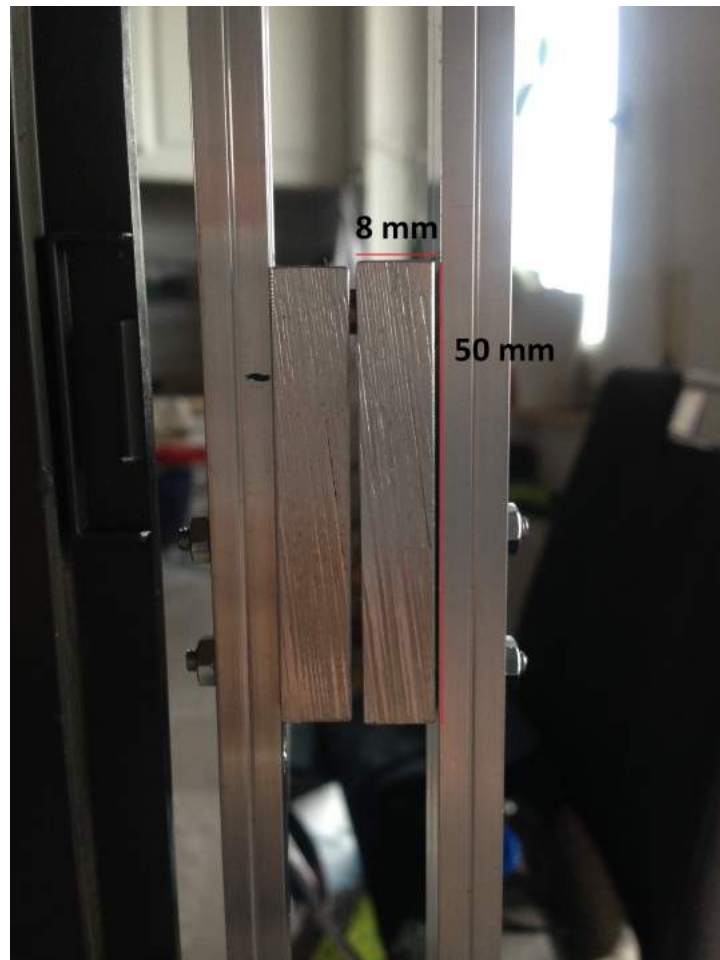


Abbildung 8.6: Zapfenband



## 8.7 Spannungsmessung des Systems

Tabelle 8.1: Messungen ohne Sonnenverlauf Algorithmus

Zeit	Akku	Solaranlage		Zeit	Akku	Solaranlage
08:48	12,58 V	12,58 V		23:18	12,15 V	12,15 V
09:18	12,55 V	12,55 V		23:48	12,13 V	12,13 V
09:48	12,52 V	12,52 V		00:18	12,12 V	12,12 V
10:18	12,49 V	12,49 V		00:48	12,11 V	12,11 V
10:48	12,47 V	12,47 V		01:18	12,10 V	12,10 V
11:18	12,46 V	12,46 V		01:48	12,08 V	12,08 V
11:48	12,45 V	12,45 V		02:18	12,05 V	12,05 V
12:18	12,44 V	12,44 V		02:48	12,03 V	12,03 V
12:48	12,43 V	12,43 V		03:18	12,01 V	12,01 V
13:18	12,42 V	12,42 V		03:48	11,99 V	11,99 V
13:48	12,41 V	12,41 V		04:18	11,97 V	11,97 V
14:18	12,40 V	12,40 V		04:48	11,95 V	11,95 V
14:48	12,39 V	12,39 V		05:18	11,94 V	11,94 V
15:18	12,38 V	12,38 V		05:48	11,92 V	11,92 V
15:48	12,36 V	12,36 V		06:18	11,89 V	11,89 V
16:18	12,35 V	12,35 V		06:48	11,87 V	11,87 V
16:48	12,33 V	12,33 V		07:18	11,85 V	11,85 V
17:18	12,32 V	12,32 V		07:48	11,84 V	11,84 V
17:48	12,31 V	12,31 V		08:18	11,82 V	11,82 V
18:18	12,30 V	12,30 V		08:48	11,80 V	11,80 V
18:48	12,28 V	12,28 V		09:18	11,78 V	11,78 V
19:18	12,27 V	12,27 V		09:48	11,75 V	11,75 V
19:48	12,26 V	12,26 V		10:18	11,73 V	11,73 V
20:18	12,24 V	12,24 V				
20:48	12,23 V	12,23 V				
21:18	12,21 V	12,21 V				
21:48	12,20 V	12,20 V				
22:18	12,19 V	12,19 V				
22:48	12,17 V	12,17 V				

Tabelle 8.2: Messungen mit Sonnenverlauf Algorithmus

Zeit	Akku	Solaranlage		Zeit	Akku	Solaranlage
00:00	12,65 V	12,65 V		14:30	12,95 V	13,21 V
00:30	12,63 V	12,63 V		15:00	12,36 V	12,55 V
01:00	12,60 V	12,60 V		15:30	12,14 V	12,14 V
01:30	12,57 V	12,57 V		16:00	12,11 V	12,11 V
02:00	12,54 V	12,54 V		16:30	12,09 V	12,09 V
02:30	12,51 V	12,51 V		17:00	12,07 V	12,07 V
03:00	12,48 V	12,48 V		17:30	12,05 V	12,05 V
03:30	12,46 V	12,46 V		18:00	12,03 V	12,03 V
04:00	12,44 V	12,44 V		18:30	12,00 V	12,00 V
04:30	12,42 V	12,42 V		19:00	11,98 V	11,98 V
05:00	12,40 V	12,40 V		19:30	11,96 V	11,96 V
05:30	12,38 V	12,38 V		20:00	11,95 V	11,95 V
06:00	12,37 V	12,37 V		20:30	11,93 V	11,93 V
06:30	12,36 V	12,36 V		21:00	11,91 V	11,91 V
07:00	12,35 V	12,35 V		21:30	11,89 V	11,89 V
07:30	12,33 V	12,33 V		22:00	11,87 V	11,87 V
08:00	12,32 V	12,32 V		22:30	11,85 V	11,85 V
08:30	12,31 V	12,31 V		23:00	11,83 V	11,83 V
09:00	12,30 V	12,30 V		23:30	11,80 V	11,80 V
09:30	12,29 V	12,29 V		00:00	11,78 V	11,78 V
10:00	12,28 V	12,28 V		00:30	11,76 V	11,76 V
10:30	12,26 V	12,26 V		01:00	11,73 V	11,73 V
11:00	12,24 V	12,24 V		01:30	11,72 V	11,72 V
11:30	12,23 V	12,23 V		02:00	11,70 V	11,70 V
12:00	12,22 V	12,22 V				
12:30	12,26 V	12,32 V				
13:00	12,19 V	12,19 V				
13:30	12,24 V	12,50 V				
14:00	13,03 V	13,49 V				