Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Viktor Felker

Überprüfung und Optimierung einer Messwerterfassung für Kleinwindanlagen

Fakultät Technik und Informatik

Faculty of Engineering and Computer Science

Viktor Felker

Überprüfung und Optimierung einer Messwerterfassung für Kleinwindanlagen

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Mechatronik
an der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Usbeck
Zweitgutachter : Ing. Breuer

Abgegeben am 20. Juli 2017

Viktor Felker

Thema der Bachelorthesis

Überprüfung und Optimierung einer Messwerterfassung für Kleinwindanlagen

Stichworte

Kleinwindanlage, Afrika, Senegal, Messwerterfassung, Optimierung, Fehleranalyse, Strommessung, Spannungsmessung, Drehzahlmessung, Temperaturmessung, Feuchtigkeitsmessung, Arduino, Platinenerstellung, Kalibrierung

Kurzzusammenfassung

Diese Arbeit umfasst die Überprüfung und Optimierung einer Messwerterfassung für Kleinwindanlagen, die in Afrika eingesetzt werden sollen. Ein Messinstrument erfasst und sammelt unterschiedliche Größen eines Windkraftwerks und sendet diese an einen Datenserver, worauf über das Internet zugegriffen werden kann, um die Messdaten auswerten zu können. In dieser Arbeit soll der ganze Vorgang geprüft und optimiert werden.

Viktor Felker

Title of the paper

Examining and optimizing the measurement of small wind turbines

Keywords

Small wind turbines, Africa, Senegal, measurement, optimizing, error analysis, current measurement, voltage measurement, speed measurement, temperature measurement, humidity measurement, Arduino, PCB production, calibration

Abstract

This work corresponds to the verification and optimization of a measured value acquisition for small wind turbines to be used in Africa. A measuring instrument collects various parameter of a wind turbine and sends it to the data server, which can be accessed via Internet in order to evaluate the measured data. This thesis examines and optimizes the whole process.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank Frau Prof. Usbeck, sowie Herrn Ing. Breuer die meine Bachelorarbeit betreut und begutachtet haben. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.

Ebenfalls möchte ich mich bei meinen Kommilitonen aus meinem Studiengang Mechatronik bedanken, die mir mit viel Geduld, Interesse und Hilfsbereitschaft zur Seite standen. Bedanken möchte ich mich für die zahlreichen interessanten Debatten und Ideen, die maßgeblich dazu beigetragen haben, dass diese Bachelorarbeit in dieser Form vorliegt.

Meinen Freunden Adrian Traistar und Nicolas Walter und Andrija Krizanac danke ich besonders für den starken emotionalen Rückhalt über die Dauer meines gesamten Studiums.

Abschließend möchte ich mich bei meinen Eltern und meiner Ehefrau bedanken, die mir mein Studium durch ihre Unterstützung ermöglicht haben und stets ein offenes Ohr für meine Sorgen hatten.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einführung

1.1. Motivation

Heutzutage wird immer wieder von erneuerbaren Energien oder auch von sauberen Energien gesprochen. Ganz groß im Kommen sind Windkrafträder zur Erzeugung elektrischer Energie. Dabei werden riesige Propeller, die an einem Generator befestigt sind, vom Wind angetrieben.

Windkrafträder werden schon seit dem 19. Jahrhundert zur Erzeugung von elektrischer Energie aufgestellt. Richtig populär sind diese aber erst seit ca. 40 Jahren. Nicht nur die Menge an Windkraftwerken ist deutlichen gestiegen, sondern auch die Leistung der einzelnen Energieanlagen. Allein in den letzten 20 Jahren ist die Leistung eines Windkraftwerks um den Faktor 500 gestiegen! Dazu kommt noch, dass die Anlagen deutlich größer geworden sind. Im Jahr 1980 waren die Anlagen bei einer Höhe von 30m, wobei sie heutzutage bei über 130m sind. Die Rotorblätter hatten einen Durchmesser von 15m und heute sind sie bei über 120m.

Die steigende Höhe, die größeren Rotordurchmesser und die höheren Leistungen sind nicht nur technisch bedingt. Bei genauerer Betrachtungsweise wird deutlich, dass die Windkraftanlagen früher eher in der Prandtl-Schicht zum Einsatz kamen, wobei die heute in der Ekman-Schicht eingesetzt werden. Die Prandtl-Schicht reicht bis ca. 100m Höhe. Daran grenzt die Ekman-Schicht, die von 100m bis ca. 1000m Höhe reicht. Die Abbildung 1.1 beschreibt die Windgeschwindigkeiten in den beiden Schichten. Dabei wird ersichtlich, dass ab ca. 80m Höhe die Windgeschwindigkeiten kaum mehr zunehmen. Etwaige Turbulenzen sind eher Effekte, die in der Prandtl-Schicht zu finden sind. In der Abbildung 1.2 wird gezeigt, wie mit zunehmender Höhe die Turbulenzen nachlassen, was dazu beiträgt, dass die Windanlagen so hoch gebaut werden. Obwohl seit mehr als 100 Jahren die Möglichkeit besteht, elektrische Energie zu erzeugen, ist heute noch vielerorts keine Elektrizität vorhanden. Vor allem in Afrika besteht das Problem großflächig. Senegal hat viele kleine Dörfer, die bis vor kurzem noch komplett ohne Elektrizität ausgekommen sind. Um das zu ändern und den Menschen einen besseren Lebensstandard zu bieten, bauen die Einwohner für sich selbst Windkraftanlagen. Preiswerte Bausätze erleichtern den Einsatz. Die Eigenbauten in Senegal sind bis

zu 20m hoch und werden dementsprechend in der Prandtl-Schicht betrieben. Es sind Insellösungen, die für den Einsatz in den Dörfern Senegals vorerst ausreichen. Die selbst gebauten Windkraftanlagen in Senegal können bis zu 800W erzeugen, wobei der Durchschnittswert bei ungefähr 300W liegt. Mit dieser Leitung wird den Menschen ermöglicht z.B. nachts Licht in den Häusern einzuschalten oder sogar Wasser zu kochen. Aus technischen Gründen ist es in den Dörfern derzeit nicht möglich, die Anlagen höher zu bauen, um die besseren Windeigenschaften sich zunutze zu machen.

Einige Anlagen sind bereits in den Betrieb genommen worden. Zur Speicherung der Energie über den Tag ist ein Akku hinter der Windkraftanlage geschaltet. Ein Spannungsregler regelt die erzeugte elektrische Energie auf 24V, dabei wird der Akku geladen. Bis jetzt wurde keine Möglichkeit geschaffen, die Spannungen, Ströme und weitere Parameter der Windkraftanlage aufzunehmen. Die Studenten der HAW-Hamburg haben deshalb im Rahmen einer Studienarbeit ein Lowbudget-Modul entwickelt, das einige Messparameter der Anlage aufnehmen und an einen Server senden kann. Der Server dient als Datenbank und wurde ebenfalls von den Studenten aufgebaut. Er kann weltweit eingesetzt werden. In diesem Fall soll der Server in Dakar zum Einsatz kommen. Zur Auswertung der Daten auf dem Server muss man sich nur auf dem Server online einloggen, dann steht die komplette Datenbank zur Verfügung.

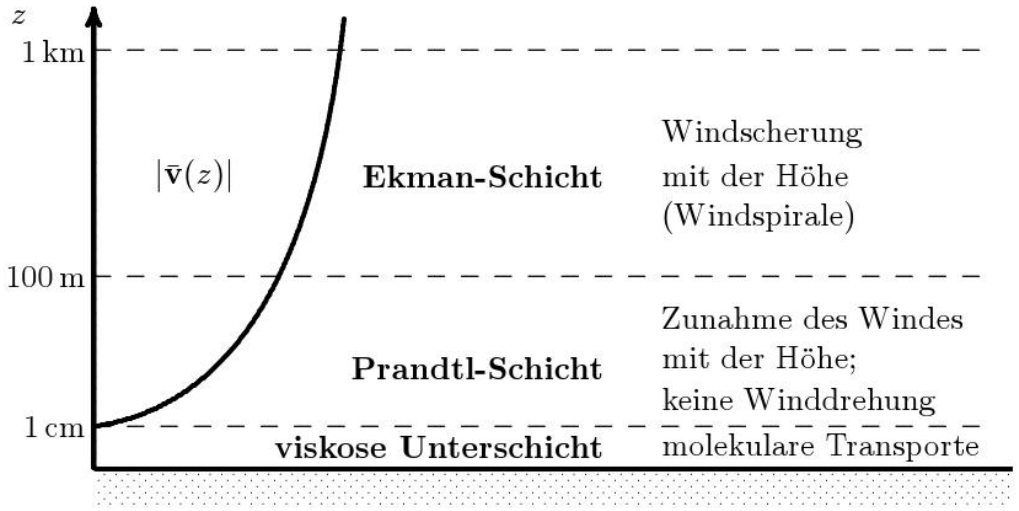


Abbildung 1.1.: Atmosphärische Grenzschichten
http://www.grund-wissen.de/diplomarbeit/_images/grenzschicht.jpg

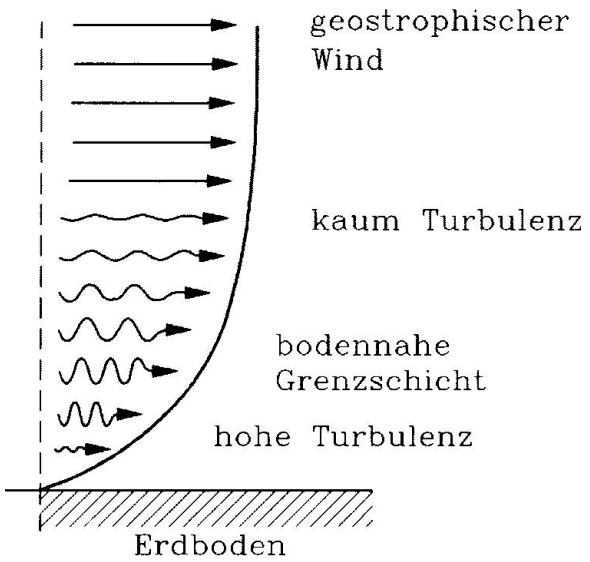


Abbildung 1.2.: Turbolenzen in Abhängigkeit von der Höhe
http://www.elite.tugraz.at/Jungbauer/pics/Image288.gif

1.2. Ziel der Abschlussarbeit

Das Ziel dieser Abschlussarbeit ist die Überprüfung und Optimierung des Datenloggers und des Servers.

Der Datenlogger nimmt Messdaten eines Windkraftwerks auf, speichert diese auf einer SD-Karte und überträgt die gesammelten Daten über das 3G-Netz an einen Datenserver. Mit einem Teststand, der ein Windkraftwerk simuliert, soll an der HAW getestet werden, ob und inwieweit der Datenlogger funktionstüchtig ist. Dabei ist zu prüfen, ob die Messaufnahme funktioniert, wie genau die Messwerte sind, die aufgenommen werden, und ob die Datenübertragung funktioniert. Zusätzlich soll eine Liste erstellt werden, die nützliche Verbesserungen und Erweiterungen auflistet. Ein Raspberry Pi 3, der als Datenbankserver dient, empfängt und speichert die gemessenen Daten über das 3G-Netz. Jeder, der die IP-Adresse des Datenservers kennt, kann über das Internet auf die gespeicherten Daten der Datenbank zugreifen. Der Datenempfang wird auf Zuverlässigkeit getestet, indem man den Server einige Tage laufen lässt und somit die Überprüfung seiner Stabilität ermittelt.

Da Kleinwindanlagen hauptsächlich in Entwicklungsländern eingesetzt werden, wird mit Hilfe des Datenloggers eine Datenaufnahme unterschiedlicher physikalischer und elektrischer Größen ermöglicht. Diese Daten sind auswertbar und können zur Optimierung der Windkraftwerke beitragen. Zusätzlich wird eine Fehleranalyse der Anlage möglich.

1.3. Aufbau der Arbeit

In diesem Abschnitt meiner Arbeit möchte ich auf alle Kapitel eingehen. Eine kurze Beschreibung soll dem Leser einen Überblick über jedes Kapitel liefern.

1. Kapitel: Erläuterung der Grundlagen zum allgemeinen Verständnis von Windkraftanlagen.
2. Kapitel: Beschreibung der Versuche und deren Durchführung.
3. Kapitel: Zusammenfassung der Versuche aus dem Kapitel 2.
4. Kapitel: Vorstellung der Verbesserungsvorschläge zur Erweiterung und Optimierung der Messwerterfassung.
5. Kapitel: Durchführung einiger Verbesserungen und Optimierungen, die im Kapitel 4 vorgestellt wurden.
6. Kapitel: Zusammenfassung nebst einem Ausblick, wohin sich die Arbeit entwickeln sollte.

1.4. Erwartungen

Jedes Projekt birgt eine gewisse Erwartungshaltung, die nicht immer erfüllt wird. Man sollte sich Gedanken machen, wie man diverse Probleme vermeiden und ggf. früher erkennen und beheben kann. Es ist besonders wichtig, einen erfolgreichen Versuch zu erzielen, weil der Datenlogger in naher Zukunft in Entwicklungsländern eingesetzt werden soll. Der Datenlogger muss die Daten aufnehmen und an den Server übertragen können. Die gemessenen Daten des Datenloggers sollen mit den Daten der Messinstrumenten übereinstimmen. Zusätzlich müssen diese korrekt an den Server (die Datenbank) übertragen werden. Der Datenlogger und der Server sollen bei einem Langzeittest ununterbrochen arbeiten. Keiner der Geräte darf ausfallen.

Wichtig ist, die Möglichkeiten zur Erweiterung der Messwerterfassung zu schaffen, indem das Aufnehmen der noch nicht vorhergesehenen Daten ermöglicht wird. Dabei sollen weitere sinnvolle Vorschläge zur Datenerfassung durch den Datenlogger genannt werden.

2. Analyse des Gesamtsystems

2.1. Versuchsaufbau

Beim Versuchsaufbau werden alle Geräte miteinander verbunden. Der Datenlogger und sämtliche Messinstrumente zu seiner Überprüfung werden an einen Generatorteststand angeschlossen. Der Server wird mit dem Internet verbunden und läuft dauerhaft.

2.1.1. Messeinrichtung

Die Messeinrichtung besteht aus zwei Komponenten: einen Datenlogger und einen als Datenbank dienenden Server.

Datenlogger

Der Datenlogger ist das eigentliche Messinstrument zur Datenaufnahme unterschiedlicher physikalischer und elektrischer Größen.

Das Herzstück des Datenloggers ist ein Arduino Mega, mit dem sämtliche Rechenprozesse durchgeführt werden. Es werden einige Module dazugeschaltet, um die SIM-Karte zu betreiben und Messungen vorzunehmen. Die SIM-Karte wird benötigt, um die Messdaten, die auf der SD-Karte gespeichert werden, an den Server über das 3G-Netz zu senden. Das ACS712-Modul ist dazu gedacht, den Strom zu messen. Zur Messung von Spannungen wird ein Spannungsteiler entwickelt. Abbildung 2.2 zeigt den Schaltplan des Spannungsteilers. Über einen Sensor wird die Umgebungstemperatur aufgenommen und die Luftfeuchtigkeit gemessen. Eine Drehzahlmessung wird mit Hilfe eines Hall-Sensors ermittelt. Am Rotor wird ein Permanentmagnet befestigt. Passiert der Magnet den Hall-Sensor, wird im Verhältnis zur Zeit die Umdrehung berechnet.

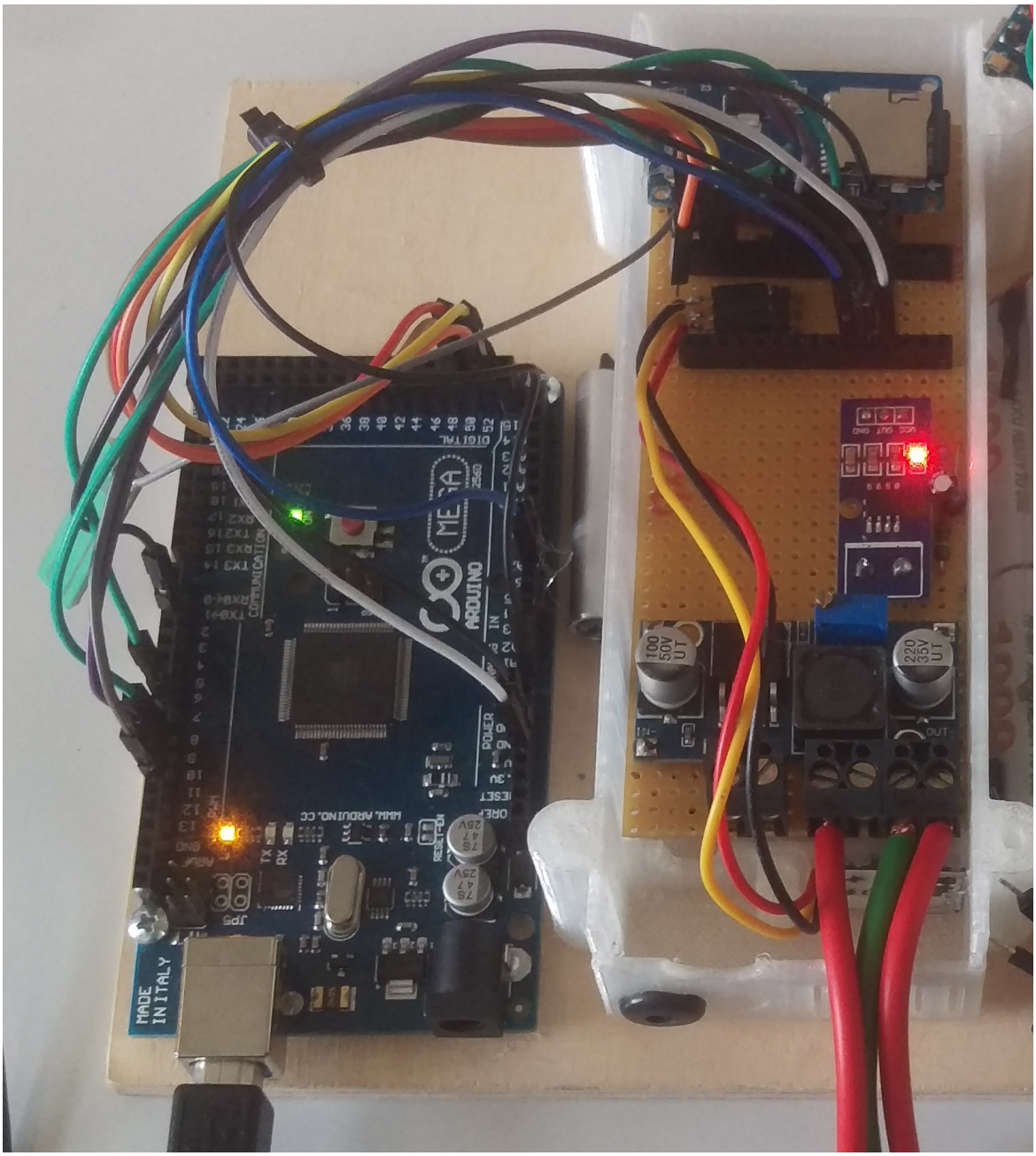


Abbildung 2.1.: Messwerterfassung: Datenlogger

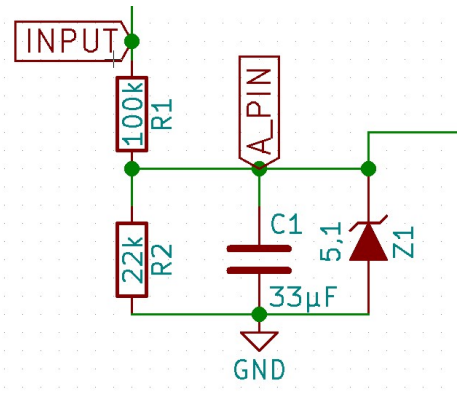


Abbildung 2.2.: Spannungsteiler zur Messung von Spannungen

Server (Datenbank)

Der Server empfängt die Daten von dem Datenlogger über das 3G-Netz und speichert diese in einer Datenbank ab.

Basierend auf einem Raspberry-Pi 3 mit einer 16GB großen Speicherkarte ist es möglich, die Daten zu empfangen und abzuspeichern. Zur Veranschaulichung wird in Abbildung 2.3 der Raspberry-Pi 3 dargestellt. Auf dem Raspberry-Pi 3 ist eine ownCloud installiert und für die nötige Anwendung eingestellt. Nicht nur die Messdaten werden erfasst, sondern auch das Datum und die Uhrzeit. Aus der ownCloud ist es möglich, Daten in unterschiedlichen Dateiformaten herunterzuladen. Zur Auswertung der Daten ist es nicht notwendig, die Daten herunterzuladen, denn in der ownCloud sind bereits Tools zur Auswertung sämtlicher Messdaten vorhanden.

2.1.2. Anschlussplan

Zunächst wird der Teststand für den Generator in Betrieb genommen. Dafür wird der Transformator an eine Starkstromquelle (400V) und der Frequenzumrichter an den Transformator angeschlossen. An den Frequenzumrichter wird der eigentliche Teststand angeschlossen. Hinter den Teststand wird ein Widerstand geschaltet, der als Lastwiderstand zur Verfügung steht. Der Generatorregler wird an den Generator angeschlossen. Zur Strom- und Spannungsmessung wird je ein Multimeter geschaltet. Es findet eine spannungsrichtige Messung statt, deswegen wird das Spannungsmessgerät zwischen dem Drehstromgenerator/Strommessgerät und dem Datenlogger/Widerstand geschaltet. Das Strommessgerät wird zwischen dem Drehstromgenerator und dem Datenlogger geschaltet. Des Weiteren wird der Datenlogger an den Generator angeschlossen. Anhand der Abbildung 2.4 wird der Anschluss der einzelnen Geräte theoretisch dargestellt.

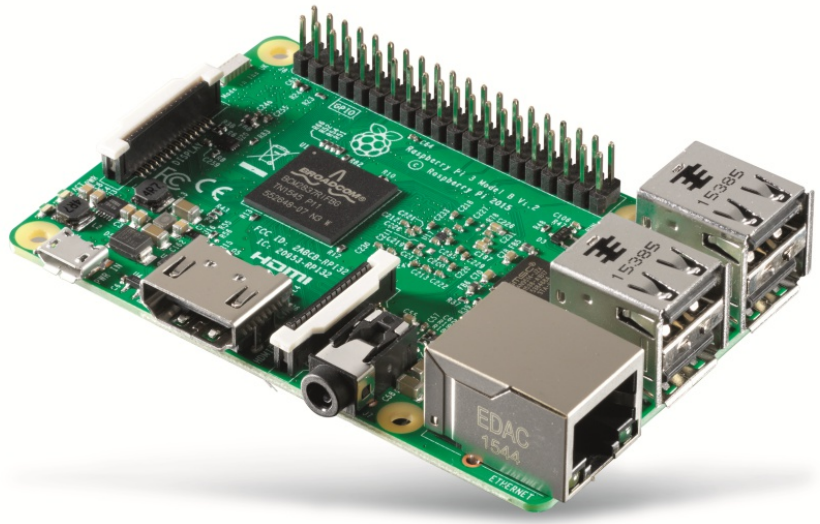


Abbildung 2.3.: Raspberry Pi 3
http://cdn.pollin.de/article/xtrabig/X702850.1.JPG

Für die Drehzahlerfassung an dem Teststand, wird ein Hall-Sensor angebracht, sowie drei Permanentmagnete an die Welle des Generators. Sobald ein Magnet den Sensor passiert, meldet der Hall-Sensor eine Messflanke, die vom Arduino detektiert wird. In der Abbildung 2.6 wird der angebrachte Sensor, sowie die Magnete am Teststand dargestellt.

2.1.3. Messinstrumente

Der Datenlogger nimmt Messwerte auf. Um die Richtigkeit der Werte zu kontrollieren, ist es notwendig, geeichte Messinstrumente zum Abgleich zu verwenden. Nachstehend werden die verwendeten Messinstrumente vorgestellt.

Multimeter

Mit einem Multimeter ist es möglich, unterschiedliche elektrische Messgrößen zu messen. In den folgenden Versuchen wird das Gerät METRA HIT 26S verwendet, um folgende Messgrößen aufzunehmen:

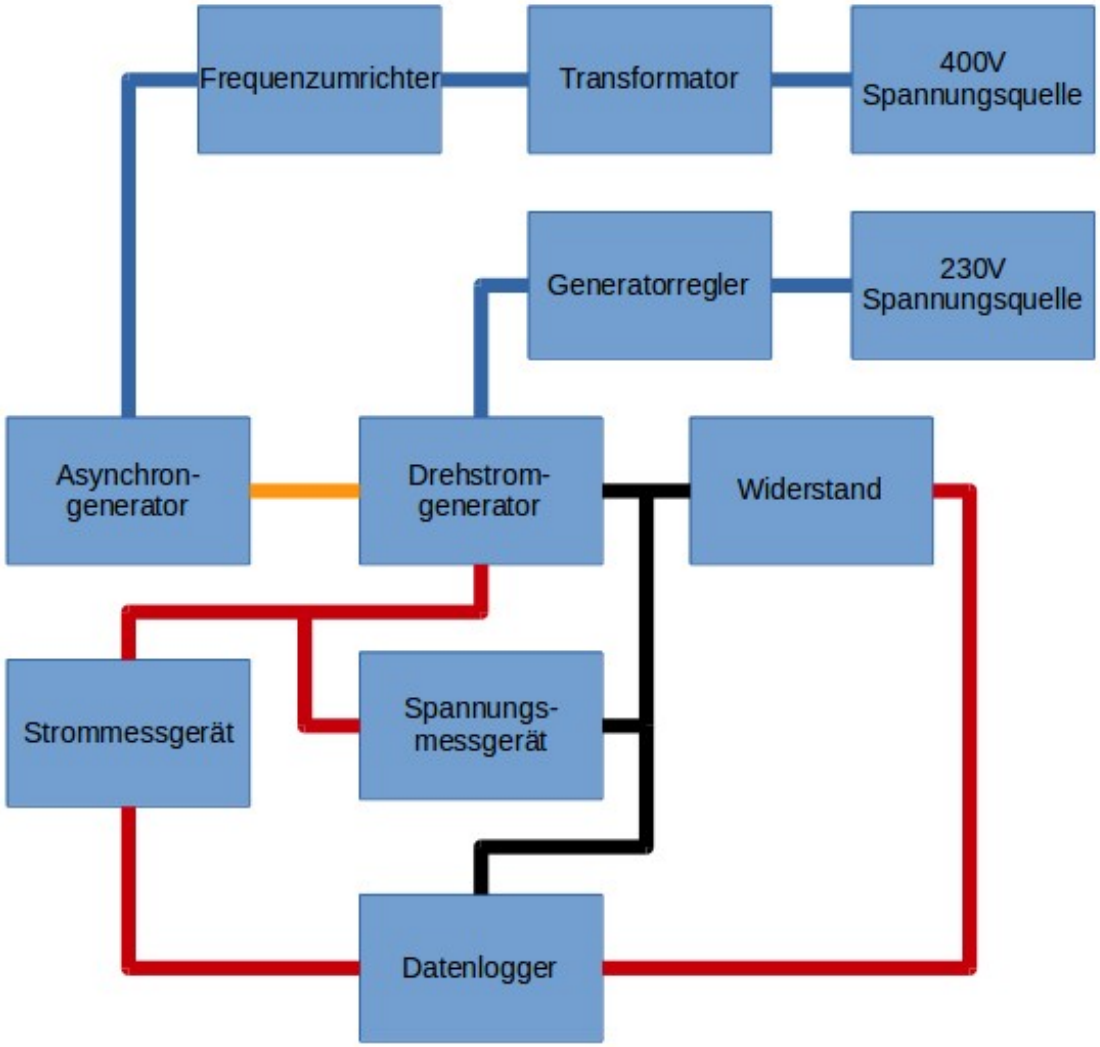


Abbildung 2.4.: Versuchsaufbau des Teststandes zur Betreibung des Generators mit dem Anschluss der Messwerterfassung. Blau-Anschlussleitung; Rot-Plusleitung Schwarz-Minusleitung; Gelb-Physikalische Übersetzung

Abbildung 2.5.: Versuchsstand

- Strom in Ampere
- Spannung in Volt

In der Abbildung 2.7 wird das verwendete Multimeter dargestellt.

Frequenzumrichter

Der Frequenzumrichter regelt die Drehzahl des Asynchronmotors. Aus anderen Versuchen geht hervor, dass die Übersetzung zwischen dem Asynchronmotor und dem Generator 3,2 beträgt. Um auf die Drehzahl des Generators zu kommen, wird die Asynchronmotordrehzahl mit dem Übersetzungsfaktor multipliziert. Die Abbildung 2.8 stellt den verwendeten Frequenzumrichter dar.

Stroboskop

Ein Stroboskop ist ein Blitzgerät, das Lichtblitze in sehr regelmäßigen zeitlichen Abständen abgibt. Dadurch erscheinen bei dunkler Umgebung Bewegungen als eine Abfolge von stehenden Bildern. Wird eine Markierung auf einer Drehachse gezeichnet, erscheint sie für das

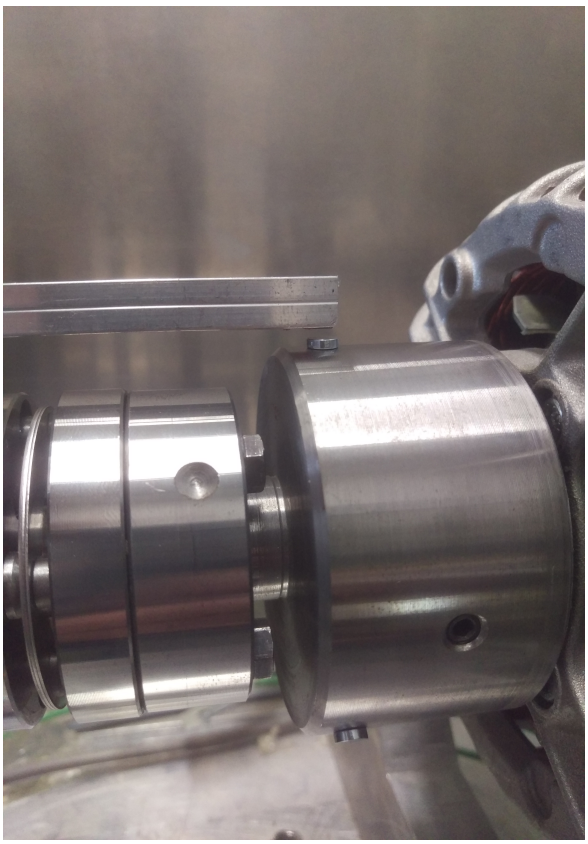


Abbildung 2.6.: Anbringung des Drehzahlsensors am Teststand

Abbildung 2.7.: Multimeter METRA HIT 26S
https://www.gossenmetrawatt.com/resources/zz_tam/hit22-26m/hit26s_pb.jpg

Abbildung 2.8.: Frequenzumrichter der Marke Siemens

menschliche Auge im Stillstand, wenn die Drehzahl der Achse mit der Frequenz des Stroboskops übereinstimmt. In der Abbildung 2.9 wird ein Beispiel für ein Stroboskop dargestellt.

Abbildung 2.9.: Stroboskop
https://asset.conrad.com/media10/isa/160267/c1/-/de/121250_LB_00_FB/voltcraft-ds-01-digitales-stroboskop-zur-visualisierung-von-bewegungsablaeufen.jpg?x=520&y=520

Thermometer

Mit diesem Messinstrument ist es möglich, folgende Messwerte zu messen:

- Lufttemperatur in Grad Celsius
- Luftfeuchtigkeit in Prozent

Zur Erfassung der Lufttemperatur und der Luftfeuchtigkeit wird das folgende Gerät in der Abbildung 2.10 genutzt.

2.1.4. Auswahl eines Generators

Zur Simulation einer Kleinwindanlage soll ein Generator eingesetzt werden. An der HAW sind zwei unterschiedliche Generatoren verfügbar. Anhand von bestimmten Parametern soll ermittelt werden, welcher der beiden besser zur Simulation von Kleinwindanlagen geeignet ist.

Zur Verfügung stehen folgende Generatoren:

Abbildung 2.10.: Thermometer

Parameter	Permanent Magnet Generator	Drehstromgenerator	Windkraftwerk
Leistung	800W	1190W	800W
Spannung	40V	14,4V	24V
Strom	20A	85A	33A
Drehzahl	4000U/min	4000U/min	300U/min
Betriebsart	Permanentmagnet	Spule	Permanentmagnet
Phasen	2	3	3
Regler	nicht vorhanden	vorhanden	vorhanden

Tabelle 2.1.: Generatorvergleich

- Permanent Magnet Generator: Die Lichtmaschine ist permanent erregt dank der Dauermagneten und leistet bis zu 800 Watt. Eine Besonderheit ist, dass es nur eine Zweiphasen-Lichtmaschine ist. Normalerweise haben Generatoren drei Phasen.
- Drehstromgenerator: Die Drehstromlichtmaschine ist normalerweise für den Einsatz in einem Kraftfahrzeug vorgesehen. Die Studenten hatten den Generator für thermische Versuche eingesetzt und dafür einen Teststand entwickelt. Generatoren für die Automobilindustrie sind bereits mit eingebauten Spannungsreglern bestückt, da die hauptsächlich zur Ladung von Auto-Batterien und zur Spannungsversorgung sämtlicher elektrischer Verbraucher verwendet werden.

Die Wahl fällt eindeutig auf den Drehstromgenerator. Die wesentlichen Vorteile sind ein vorhandener Teststand und ein bereits eingebauter Spannungsregler, die den Versuch erheblich

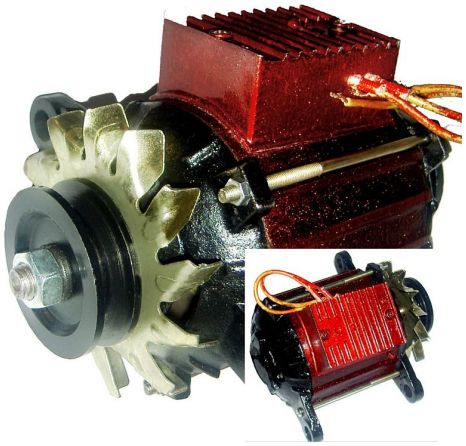


Abbildung 2.11.: Permanent Magnet Generator
https://www.hood.de/img1/full/9204/92040758.jpg

Abbildung 2.12.: Drehstromgenerator
http://remgen.ru/blog/generator-0124225010-0986042470-0986042471-1303701bo-yle102420-yle102430-gnu2630-gnu2632

erleichtern. Das Anlaufmoment ist relativ gering, da der Generator mit der Hand angetrieben werden kann. Die Spulen ersetzen die Magneten. Dafür müssen sie mit einem Erregerstrom belastet werden, um ein Magnetfeld zu erzeugen.

Der permanent Magnet Generator ist, wie der Name schon sagt, ein permanent erregter Zweiphasen-Generator. Er könnte mit dem Teststand betrieben werden, aber benötigt dafür eine Kupplung, um eine Kraftübertragung zwischen dem Teststand und der Antriebswelle des Generators zu realisieren. Dadurch ist ein viel höheres Anlaufmoment notwendig. Ein Spannungsregler ist für den Generator nicht vorhanden und müsste erst für den Versuch entwickelt werden.

2.1.5. Inbetriebnahme des Servers

Zur Inbetriebnahme des Servers wird zunächst der physikalische Anschluss und anschließend das Einrichten im Netzwerk erläutert.

Vorerst muss der Raspberry-Pi mit dem Strom versorgt werden. Dafür wird mittels eines dafür vorgesehenen Stromkabels der Raspberry-Pi an einer Schuko-Steckdose angeschlossen. Um den Server mit dem Internet zu verbinden, wird ein LAN-Anschluss des Servers und ein LAN-Anschluss des Internet-Routers mit einem Cat5-Patchkabels verbunden. Eine grüne LED leuchtet dabei an der LAN-Buchse am Server auf. Somit ist die physikalische Verbindung mit dem Router hergestellt.

Nun muss der Server in dem Netzwerk freigeschaltet werden, damit die Datenpakete empfangen werden können und auf die Datenbank zugegriffen werden kann. Eine Freigabe des Raspberry-Pi als HTTP-Server ist in der Routereinstellung notwendig. Es sollte dabei beachtet werden, dass der Port 2500 extra freigegeben wird. Über den Port 2500 werden die Messdaten an den Server übertragen. Wird der nicht freigegeben, kann keine Datenübertragung stattfinden.

Damit die Daten von dem Datenlogger an den Server gesendet werden können, muss man im Quellcode des Datenloggers die IP-Adresse des Servers einstellen. Die Einstellung einer DNS ist im Quellcode nicht vorgesehen und somit auch nicht möglich. Normalerweise haben öffentliche Einrichtungen oder Unternehmen statische und Privathaushalte dynamische IP-Adressen. Man kann also davon ausgehen, dass der Server mit einer statischen IP-Adresse versehen wird. Das Anpassen der IP-Adresse im Quellcode findet in der gprs.h statt. Der Port 2500, über den das Empfangen der Datenpakete stattfindet, ist bereit im Quellcode voreingestellt. Sollte der Port bereits für andere Zwecke genutzt werden, muss er sowohl im Quellcode als auch am Server geändert werden. Die Ports von 0 bis 1023 sind bereits für Standardanwendungen in der Informatik reserviert. Alle anderen Ports können von normalen Nutzern ohne besondere Rechte genutzt werden.

2.2. Versuchsdurchführung

2.2.1. Inbetriebnahme der Messsystems

Sobald alles angeschlossen ist, wird der Hauptschalter an der Steckdose im Versuchslabor eingeschaltet. Rote Leuchtmittel an der Steckdose leuchten auf, was auf einen geschlossenen Stromkreis hinweist. Das Spannungsmessinstrument am Transformator muss auf über 400V ausschlagen. Die Abbildung 2.13 zeigt den Spannungsmesser im ausgeschlagenen Zustand.

Abbildung 2.13.: Voller Spannungsausschlag am Transformator

Sobald mindestens 400V erreicht sind, kann der Hebel am Frequenzumrichter betätigt und der Frequenzumrichter eingeschaltet werden. Die Lüftung des Frequenzumrichters ist zu hören und das Display am Frequenzumrichter leuchtet auf. Auf dem Teststand befindet sich der Drehstromgenerator, der von einem Asynchronmotor angetrieben wird. Mit Hilfe eines Keilrippenriemens und einer Übersetzung von 3,2 treibt der Asynchronmotor den Drehstromgenerator an. Da bei der Simulation keine Akku-Batterie verwendet wird, die vom Generator gespeist ist, wie im realen Betrieb, wird ein Widerstand als Last verwendet. Mit dem Widerstand wird die erzeugte Energie vom Generator in Wärme umgewandelt. Wird der grüne Button am Frequenzumrichter betätigt, läuft der Asynchronmotor an und treibt den Generator

an. Nachdem das Erreger-Modul an eine Spannungsquelle angeschlossen wurde, wird der Strom über das Erregerstrom-Modul in die Spulen des Drehstromgenerators geleitet und es entsteht ein Magnetfeld. Dank dem Magnetfeld erzeugt der Drehstromgenerator elektrische Energie. Natürlich hat das Injizieren des Stroms in die Spulen eine Kraft erzeugt, die auf das Drehen des Generators einen negativen Einfluss bewirkt.

2.2.2. Aufnahme der Strom- und Spannungswerte

Durchführung

Die Aufnahme der Strom-, Spannungs- und Drehzahlwerte durch den Datenlogger soll in diesem Versuch überprüft werden. Der Versuch trägt dazu bei, die Genauigkeit der gemessenen Werte zu ermitteln. Bei der Aufnahme der Messwerte werden unterschiedliche Drehzahlen am Generator eingestellt. Dadurch sollen verschiedene Zustände simuliert werden, die durch die Natur beziehungsweise durch die Wetterlage in der Realität entstehen.

Bei dem Versuch wird die Generatordrehzahl von 800U/min in 100U/min schrittweise auf 300U/min gesenkt. Danach wird die Drehzahl wieder von 300U/min auf 800U/min in 100-ter Schritten hochgefahren. Beide Testläufe, die aus den Tabellen 2.2, 2.3 ersichtlich sind, laufen um einen Tag versetzt.

Hinzu kommt, dass die Batterie, die die erzeugte Leistung des Windkraftwerks speichert, je nach Ladezustand und Temperatureinfluss, einen anderen Lastwiderstand annehmen kann. Aus diesem Grund werden drei Messungen je mit einem anderen Lastwiderstand durchgeführt. Es wird ein Lastwiderstand von 10 Ohm, 6 Ohm und 3 Ohm für die Messungen verwendet.

Ergebnisse

In den Tabellen 2.2 und 2.3 werden die aufgenommenen Messdaten dargestellt.

Auswertung

In der Tabelle 2.4 wird je ein Mittelwert für die Größen aus der Tabelle 2.2 berechnet. Die Differenz zwischen den gemessenen Werten der Instrumente und dem Datenlogger wird als Fehlertoleranz des Datenloggers dargestellt. Die Fehlertoleranzen werden dabei als Absolutwerte und in Prozent angegeben. Gleiches wird in der Tabelle 2.5 für die Tabelle 2.3 dargestellt.

Instrumente			Datenlogger		Last
n in U/min	I in A	U in V	I in A	U in V	R in Ohm
800,00	1,49	14,40	1,44	14,42	10 Ohm
700,00	1,48	14,34	1,42	14,31	
600,00	1,38	13,36	1,33	13,29	
500,00	1,20	11,61	1,13	11,57	
400,00	1,01	9,77	0,95	9,75	
300,00	1,15	11,11	1,11	11,08	
400,00	1,12	10,87	1,05	10,81	
500,00	1,13	10,93	1,08	10,94	
600,00	1,45	14,07	1,40	14,06	
700,00	1,46	14,18	1,41	14,13	
800,00	1,47	14,24	1,41	14,17	
800,00	2,42	14,18	2,35	14,04	6 Ohm
700,00	2,41	14,14	2,37	14,02	
600,00	2,24	13,15	2,19	13,01	
500,00	1,96	11,48	1,86	11,36	
400,00	1,63	9,59	1,50	9,45	
300,00	1,80	10,61	1,78	10,48	
400,00	1,80	10,53	1,79	10,48	
500,00	1,82	10,65	1,66	10,51	
600,00	1,80	10,60	1,63	10,51	
700,00	1,80	10,58	1,77	10,42	
800,00	2,44	14,31	2,38	14,19	
800,00	4,65	14,18	4,59	13,83	3 Ohm
700,00	4,43	13,50	4,35	13,16	
600,00	3,96	12,11	3,89	11,79	
500,00	3,51	10,71	3,46	10,45	
400,00	3,06	9,31	2,93	9,05	
300,00	3,06	9,33	2,98	9,06	
400,00	3,06	9,33	3,04	9,09	
500,00	3,05	9,32	3,02	9,05	
600,00	3,06	9,33	3,01	9,07	
700,00	3,06	9,34	3,03	9,03	
800,00	3,05	9,34	3,02	9,04	

Tabelle 2.2.: Erste Messung nach dem Generatorregler bei unterschiedlichen Drehzahlen und Widerständen

Instrumente			Datenlogger		Last
n in U/min	I in A	U in V	I in A	U in V	R in Ohm
800,00	1,48	14,3	1,42	14,26	10 Ohm
700,00	1,47	14,25	1,41	14,17	
600,00	1,4	13,55	1,36	13,44	
500,00	1,2	11,62	1,16	11,56	
400,00	1,17	11,29	1,13	11,23	
300,00	1,17	11,3	1,1	11,2	
400,00	1,17	11,27	1,13	11,24	
500,00	1,17	11,29	1,13	11,24	
600,00	1,17	11,29	1,13	1,24	
700,00	1,17	11,3	1,13	11,23	
800,00	1,49	14,39	1,45	14,33	
800,00	2,44	14,26	2,38	14,09	6 Ohm
700,00	2,42	14,13	2,36	13,99	
600,00	2,23	13	2,15	12,88	
500,00	1,94	11,35	1,89	11,24	
400,00	1,63	9,55	1,58	9,42	
300,00	1,81	10,53	1,76	10,42	
400,00	1,81	10,67	1,77	10,55	
500,00	1,83	10,65	1,76	10,51	
600,00	1,83	10,72	1,75	10,57	
700,00	1,83	10,66	1,76	10,51	
800,00	1,83	1,67	1,7	10,55	
800,00	4,67	14,18	4,56	13,78	3 Ohm
700,00	4,46	13,54	4,39	13,24	
600,00	4	12,14	3,93	11,79	
500,00	3,54	10,75	3,46	10,43	
400,00	3	9,11	2,93	8,82	
300,00	3,08	9,28	3,03	9,05	
400,00	3,12	9,46	3,08	9,19	
500,00	3,12	9,47	3,05	9,17	
600,00	3,12	9,46	2,71	9,16	
700,00	3,11	9,42	3,05	9,13	
800,00	3,1	9,39	2,99	9,11	

Tabelle 2.3.: Zweite Messung nach dem Generatorregler bei unterschiedlichen Drehzahlen

Aus der Tabelle 2.6 wird deutlich, dass der Datenlogger eine Abweichung der Strommessung von 2,99% misst, was etwa 0,07 Ampere entspricht. Bei den Spannungsmessungen liegt die Messabweichung bei 1,63%, das sind ca. 0,18 Volt.

Allgemein sind das schon sehr genaue Messergebnisse, ohne dass der Datenlogger richtig kalibriert wurde. Eine gewisse Abweichung war abzusehen, jedoch sollte die Kalibrierung vorgenommen werden, um die Messabweichungen zu minimieren. Ein guter Wert wäre im Bereich unter einem Prozent.

	Instrumente	**Datenlogger**	**Fehler**	**Fehler in Prozent**
Strom in A	2,25	2,19	0,06	2,74
Spannung in V	11,22	11,07	0,15	1,36

Tabelle 2.4.: Mittelwertbildung und Fehlerberechnung aus der Tabelle 2.2

	Instrumente	**Datenlogger**	**Fehler**	**Fehler in Prozent**
Strom in A	2,24	2,17	0,07	3,23
Spannung in V	10,76	10,56	0,20	1,89

Tabelle 2.5.: Mittelwertbildung und Fehlerberechnung aus der Tabelle 2.3

	Instrumente	**Datenlogger**	**Fehler**	**Fehler in Prozent**
Strom in A	2,25	2,18	0,07	2,99
Spannung in V	10,99	10,82	0,18	1,63

Tabelle 2.6.: Gesamtmittelwertbildung und Gesamtfehlerberechnung aus den Tabellen 2.4 und 2.5

2.2.3. Aufnahme der Drehzahlwerte des angetriebenen Generators

Durchführung

Zur Drehzahlaufnahme wird ein Permanentmagnet an die Welle des Generators angebracht. Bei die langsamen Windgeschwindigkeiten wird sich das Windrad nur sehr langsam drehen, wobei Drehzahlen von 10 U/min und weniger erreicht werden. In solchen Fällen wird der Datenlogger relativ ungenau, da alle zwei Sekunden ein Wert aufgenommen wird. Um die Genauigkeit zu erhöhen, werden an die Welle des Generators drei Permanentmagnete im Abstand von 120° angebracht. So werden auch 1/3 Umdrehungen pro zwei Sekunden erkannt.

Die Anpassung der Software ist nötig, damit die Berechnungen richtig vollzogen werden. In der Datei *sensors.cpp* wird in der Routine *getRpsAverage ()* die Variable *rps_average* durch drei dividiert und anschließend mit 60 multipliziert. Somit wird von Sekunden auf Minuten umgerechnet. Die Abbildung 2.14 stellt einen Ausschnitt aus dem Quellcode dar, in dem die Veränderung durchgeführt wurde.

```
double Sensors::getRpsAverage()
{
  double rps_average = 0;
  for (int i = 0; i < MAX_COUNT; i++)
  {
    rps_average += rps[i];
  }
  rps_average /= (MAX_COUNT * 3);                    // MIT 3 MULTIPLIZIERT, DA 3 MAGNETE!
  rps_average *= 60;                                 // MIT 60 MULTIPLIZIERT, UM AUF MINUTEN ZU KOMMEN

  if ((rps_average >= 0) || (rps_average < 60000)){
    return rps_average;
  }
  else return 0;

}
```

Abbildung 2.14.: CODE: Mittelwertbildung der Drehzahl des Windrads

Um die Drehzahl exakt zu erfassen, wird eine Stroboskoplampe verwendet. Dafür ist eine beliebige Markierung an der Welle nötig. Wird die richtige Frequenz an der Lampe eingestellt, erscheint die Welle des Generators für das menschliche Auge im Stillstand. Zur Einstellung der richtigen Frequenz werden am Stroboskop Drehköpfe auf- oder zugedreht.

Ein Windrad dreht sich entweder gar nicht oder erreicht je nach Windstärke eine gewisse Drehzahl. Aus bisherigen Berichten geht hervor, dass die Drehzahlen nicht über 800 U/min hinausgehen. Um einen Puffer zu schaffen, sollen Drehzahlen von 1200 U/min bis 10 U/min gefahren werden. Zwischen den beiden Extremen sind weitere 9 beliebige Drehzahlmesswerte aufzunehmen.

Ergebnisse

Die Messergebnisse sind in der Tabelle 2.7 dargestellt.

Auswertung

Der Versuch hat gezeigt, dass die Drehzahlerfassung mit dem Hallsensor und den drei Magneten hervorragend funktioniert. Es sind hervorragende Ergebnisse gemessen worden. Mit dem Stroboskop lassen sich nur ganze Werte einstellen. Daher sind die Messwerte aus der Tabelle 2.7 als perfekt anzusehen.

Instrumente	Datenlogger
1200,0	1200,0
1060,0	1060,0
920,0	920,0
840,0	840,0
720,0	720,0
600,0	600,0
474,0	473,4
340,0	340,0
100,0	100,0
40,0	40,0
12,0	11,6

Tabelle 2.7.: Versuch 1: Aufnahme der Drehzahlmesswerte. Alle Werte werden in U/min gemessen.

Niedrige Drehzahlen lassen sich nur schwer an Frequenzumrichter einstellen. Aus dem Grund ist die niedrigste gemessene Drehzahl 12U/min.

2.2.4. Aufnahme der Umgebungstemperatur und der Feuchtigkeit

Durchführung

Weitere wichtige Messdaten sollten erfasst und auf Genauigkeit überprüft werden. Die Außentemperatur und Feuchtigkeit werden über einen Sensor an dem Datenlogger erfasst und an die Datenbank gesendet. Diese Messdaten können ausgewertet werden.

Bei der Durchführung werden Messwerte in unterschiedlichen Räumlichkeiten aufgenommen. Dadurch lassen sich verschiedene Feuchtigkeiten und Temperaturen einfach realisieren.

Ergebnis

Alle Messergebnisse sind in den Tabellen 2.8 und 2.9 hinterlegt.

Messinstrument		**Datenlogger**	
Temperatur °C	**Feuchtigkeit %**	**Temperatur °C**	**Feuchtigkeit %**
19,2	58	19,3	61,2
21,5	47	21,4	48,1
21,8	47	21,8	47,3
22,5	49	22,9	51,8
21,9	50	22,3	54,4
22,3	54	22,8	59,1
22,6	62	22,9	65,8
23,6	59	24,0	63,6

Tabelle 2.8.: Erste Messwertaufnahme zum Vergleich von Drehzahl, Umgebungstemperatur, Feuchtigkeit

Messinstrument		**Datenlogger**	
Temperatur °C	**Feuchtigkeit %**	**Temperatur °C**	**Feuchtigkeit %**
22,3	50	22,9	52,3
22,7	48	23,2	50,1
21,9	52	22,1	55,7
22,5	56	23,0	60,1
23,1	54	23,2	55,8
22,1	58	22,3	61,2
24,7	58	25,3	60,4
24,3	54	24,9	55,3

Tabelle 2.9.: Zweite Messwertaufnahme zum Vergleich von Drehzahl, Umgebungstemperatur, Feuchtigkeit

Auswertung

Alle Messergebnisse konnten erfolgreich aufgenommen werden. Aus beiden Versuchstabellen 2.8 und 2.9 wird ersichtlich, dass die Messwerte der Messinstrumente und des Datenloggers nah beieinander liegen. Um die Daten besser auswerten zu können, wird eine Mittelwertbildung der beiden Versuchstabellen durchgeführt, die in der Tabelle 2.10 ersichtlich ist.

Messinstrument		**Datenlogger**	
Temperatur °C	**Feuchtigkeit %**	**Temperatur °C**	**Feuchtigkeit %**
22,4	53,5	22,8	56,4

Tabelle 2.10.: Mittelwertbildung der Ergebnisse aus den Tabellen 2.8 und 2.9

Eine Auswertung der Tabelle 2.10 ergibt, dass im Durchschnitt die Temperatur um 0,4 ° Celsius mehr gemessen wird. Die Abweichung der Feuchtigkeitsmessung beträgt im Durchschnitt 2,9%.

Die hier gemessenen Abweichungen sind für den Versuch absolut akzeptabel und auf die unterschiedlichen Materialien der einzelnen Gerätschaften zurückzuführen. Eine Optimierung der Messwerterfassung im Bereich der Temperatur- und Feuchtigkeitsmessung ist nicht notwendig. Messabweichungen von unter einem Grad Celsius und Feuchtigkeiten von unter 5% sind für diesen Versuch und den Einsatz des Datenloggers akzeptabel.

2.2.5. Übertragungsprüfung

Durchführung

Die erfassten Daten werden an einen Server gesendet. Dies geschieht über das 3G-Netz. Die Daten werden dabei in Datenpakete aufgeteilt. Es ist zu überprüfen, ob sie angekommen und keine Probleme dabei aufgetreten sind.

Der Datenlogger ist mit einer SIM-Karte ausgestattet, die das Übertagen von Daten ermöglicht. Der Server empfängt die Daten und speichert sie in einer Datenbank.

Für den Versuch werden Messdaten von dem Datenlogger aufgenommen. Sobald die Daten in der Datenbank ersichtlich sind, werden sie mit denen von dem Datenlogger verglichen und auf Vollständigkeit geprüft.

Ergebnisse

Eine zuverlässige Übersendung der Daten ist nicht möglich. Die Übersendung einer Datei an den Server ist zwar geglückt, aber die Datei ist leer. Die von dem Datenlogger aufgenommenen Daten erscheinen dadurch nicht in der Datenbank.

Sobald eine Datei gesendet wurde, misslingt das Senden weiterer Datenpakete. Bei jedem weiterem Versendeversuch erscheint *"Joning APN: failed"*. Wird das SIM-Modul manuell neu gestartet, findet eine Datenübertragung statt. In der Abbildung 2.15 wird gezeigt, wie über eine serielle Schnittstelle auf den Datenlogger zugegriffen wird und der Verbindungsversuch dabei misslingt.

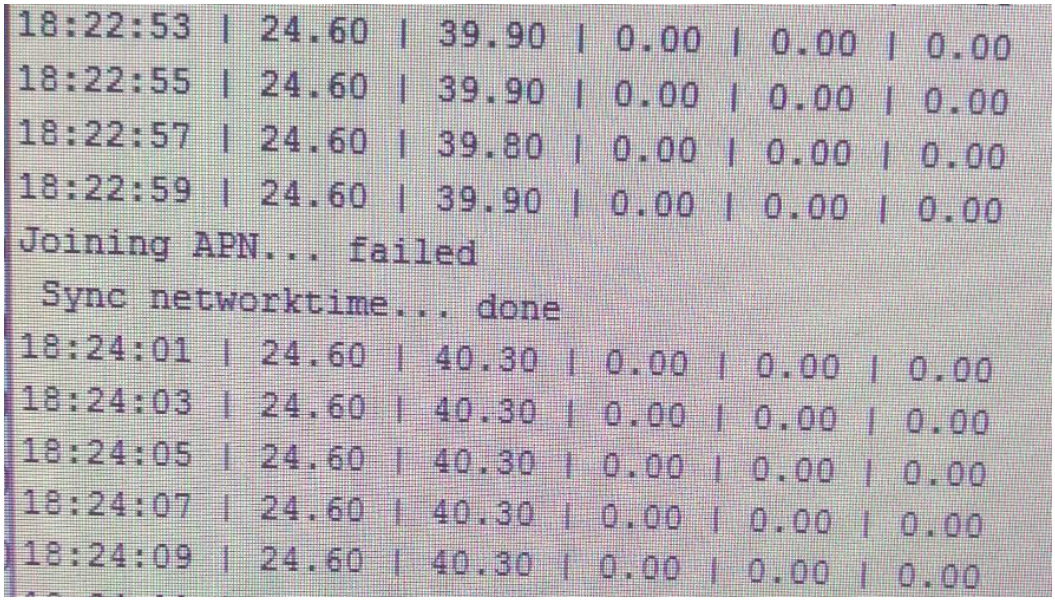


Abbildung 2.15.: Fehlermeldung *Joining APN: failed*

Auswertung

Das Übertragen der Daten von dem Datenlogger zum Server ist nur bedingt gelungen. Es konnte zwar eine Datei an den Server übertragen werden, jedoch war sie leer. Dadurch ist die Datei unbrauchbar. Sämtliche Messergebnisse beziehungsweise Messdaten gehen dabei verloren.

Der Versuch ist somit misslungen und sollte untersucht werden. Ohne eine zuverlässige Übertragung hat das System keinen Wert.

3. Zusammenfassung der Versuche

Bei den Versuchen zur Strom- und Spannungsmessung wurde deutlich, dass die Messwerterfassung zuverlässige Ergebnisse liefert. Trotz einigen Fehlertoleranzen, die für einen Prototypen sehr positiv ausgefallen sind, wurden bessere Messergebnisse erzielt als erwartet. Eine Kalibrierung des Datenloggers ist dennoch sinnvoll, um die Fehlertoleranzen noch geringer zu halten.

Die Drehzahlversuche haben gezeigt, dass die Erfassung hervorragend funktioniert. Es wurde erwartet, dass einige Messflanken wegen des Programmcodes geschluckt werden, was nicht aufgetreten ist. Bessere Ergebnisse hätten nicht erzielt werden können. Damit ist die Drehzahlaufnahme fertiggestellt und muss nicht noch optimiert werden.

Genau wie bei den Versuchen aus dem Kapitel 2.2.2 sind die Messergebnisse absolut positiv ausgefallen. Das belegen sehr geringe Fehlertoleranzen. Es ist kaum vorstellbar solche Ergebnisse zu messen ohne eine richtige Kalibrierung durchgeführt zu haben. Die Tabelle 2.10 veranschaulicht das deutlich.

Der Versuch, die Messdaten von dem Datenlogger an den Datenserver zu übersenden, ist misslungen. Selbst beim Versuch die Übertragung kurzfristig zu debuggen, konnte keine erfolgreiche Übertragung geschaffen werden. Dieser Punkt muss analysiert und aufgearbeitet werden.

Mit allen Versuchen konnte deutlich dargestellt werden, dass das Messsystem teilweise erfolgreich funktioniert. Alle Werte konnten gemessen werden und die Fehlertoleranzen sind für einen Prototypen relativ gering ausgefallen. Nur die Übertragungsprüfung stellt ein Hindernis dar. Keine Messwerte konnten übertragen werden. Der Serverzugriff erweist sich jedoch als sehr einfach. Auf die Datenübertragung sollte noch einmal richtig eingegangen werden, da die Datenübertragung vor allem eine primäre Eigenschaft des ganzen Systems ist.

In der folgend aufgeführten Aufzählung sind alle positiven und negativen Ergebnisse der Versuche aufgeführt.

Positive Ergebnisse:

- Erfassung aller Messdaten

- Fehlertoleranzen aller Messgrößen gering
- Einfacher Zugriff auf alle Messdaten aus dem Datenlogger, als auch am Server
- Einfache Handhabung der Anschlüsse am Datenlogger
- Offener und gut dokumentierter Quellcode

Negative Ergebnisse:

- Auftreten von Wackelkontakten
- Schnell lösende Steckverbindungen
- Keine Übersicht über die Verkabelung
- Kalibrierung für noch bessere Messergebnisse notwendig
- Keine DNS für den Server
- Fehler im Quellcode
- Siemens Teststand liefert falsche Drehzahlwerte
- Datenübertragung an den Server fehlgeschlagen

Zukünftig sollte an dem Messsystem noch einiges optimiert werden. Obwohl die Tests relativ positiv ausgefallen sind, sollten die negativen Aspekte nicht außer Acht gelassen werden. Schließlich soll das ganze System für Menschen in Afrika eingesetzt werden.

4. Verbesserungsvorschläge

Da der Datenlogger noch nicht komplett ausgereift ist, werden einige sinnvolle Verbesserungsvorschläge genannt. Die Vorschläge sind aus technischer, anwendungsbezogener und analytischer Sicht nützliche Erweiterungen und Optimierungen. Jede Erweiterung und ihre Sinnhaftigkeit wird in diesem Kapitel beschrieben.

Die meisten Vorschläge unterscheiden sich stark voneinander und müssen daher in einer Prioritätenliste priorisiert werden. In der unten aufgeführten Liste 4 werden alle Vorschläge gewichtet. Dabei trägt die kleinste Zahl die größte Priorität. Die Gewichtung der einzelnen Punkte geschieht nach der Sinnhaftigkeit. Der Grund dafür ist, dass es sinnvoller erscheint, vorerst alles fertig zu entwickeln und dann die Platine zu erstellen.

Prioritätenliste:

1. Herstellung der Datenübertragung mit dem Server
2. Integration eines Akkus für den Datenlogger
3. Optimierung der Drehzahlbestimmung durch Software
4. Messung der Windgeschwindigkeit
5. Möglichkeiten zur Erweiterung der Messparameter
6. Messung im Wechselspannungsbetrieb
7. Entwicklung einer Shield-Platine
8. Gehäuseentwicklung für den Datenlogger
9. Gehäuseentwicklung für den Server
10. Kalibrierung des Datenloggers für genauere Messergebnisse
11. Softwareoptimierung
12. Integration einer Fehleranalyse
13. Erweiterung der Strommessung

4.1. Herstellung der Datenübertragung mit dem Server

Da der Test im Kapitel *Übertragungsprüfung* 2.2.5 misslungen ist und die Daten dabei nicht vom Datenlogger an den Datenserver übertragen werden konnten, muss eine Überprüfung und Behebung des Problems durchgeführt werden.

4.2. Integration eines Akkus für den Datenlogger

Das Hinzufügen eines Akkus ist dringend notwendig. Bei einem Windstillstand oder zu geringer Windstärke wird der Datenlogger nicht mehr mit dem Strom versorgt. Das führt zu einem Ausfall des Messsystems. Die Daten werden nicht mehr erfasst und an die Datenbank gesendet. Es ist allgemein bekannt, dass die Akkus mit einem konstanten Strom geladen werden sollen. Damit wird die Lebensdauer des Akkus verlängert und die Ausfallsicherheit gesteigert. Zur Berücksichtigung dieses Aspekts ist es ratsam, eine Konstantstromquelle einzubauen.

4.3. Optimierung der Drehzahlbestimmung durch Software

Aus dem Test im Kapitel *Aufnahme der Drehzahlwerte des angetriebenen Generators* 2.2.3 geht hervor, dass die Daten zuverlässig aufgenommen werden konnten. Weitere Überprüfungen sollten zeigen, dass die Drehzahlbestimmung zuverlässig funktioniert. Des Weiteren muss in der Software der Drehzahlwert zur Bestimmung der U/min am Generator mit dem errechneten Multiplikator aus dem Kapitel 2.2.3 erweitert werden.

4.4. Messung der Windgeschwindigkeit

Ein schon jetzt wichtiger Parameter ist die Windgeschwindigkeit. Anhand der Windgeschwindigkeit wird es möglich sein, die Leistung des Windrades zu bestimmen und zu optimieren. Daraus resultieren auch günstigere Standorte für die Windkraftwerke. Der Sensor in der Abbildung 5.10 eignet sich für solche Messungen.

4.5. Möglichkeit zur Erweiterung der Messparameter

Mit der Erfahrung aus früheren Projekten weiß man, dass im Laufe des Einsatzes immer wieder neue Ideen und notwendige Parameter benötigt werden. Dementsprechend sollte eine einfache Möglichkeit geschaffen werden, neue Parameter zu implementieren. Es könnten z.B. weitere Anschlüsse vorbereitet werden, um Daten zu erfassen. Natürlich muss dazu der Quellcode offen stehen.

4.6. Messungen im Wechselspannungsbetrieb

Zu einer sinnvollen Erweiterung der Messeinrichtung gehört die Messwertaufnahme zwischen dem Generator und dem Generatorregler. Der Generator erzeugt eine Wechselspannung. Es werden Messungen des Wechselstroms, der Wechselspannung und die Frequenz benötigt. Genauere Werte der erzeugten elektrischen Leistung werden berechenbar, weshalb auch die Optimierung des Generators erleichtert wird. Ein weiterer Vorteil ist das Analysieren von Fehlern. Eventuelle Fehler können dank diesen Messungen zwischen dem Generator und dem Regler unterschieden werden. In der Abbildung 4.1 wird ein Sensor abgebildet, der den Gleichstrom auch als Wechselstrom bis 50A messen kann.

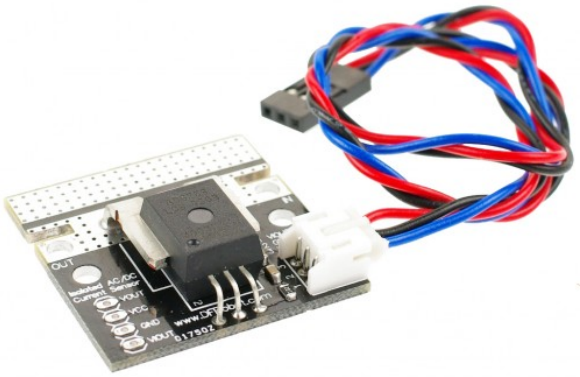


Abbildung 4.1.: Strommesssensor bis 50A AC/DC
http://www.komputer.de/zen/images/SEN0098.jpg

4.7. Entwicklung einer Shield-Platine

Derzeit sind alle Bauteile des Datenloggers über eine Lochrasterplatine, worauf die Leiterbahnen gelötet oder über Kabel verbunden sind, miteinander verbunden. Eine schönere Lösung ist die Entwicklung einer Shield-Platine, worauf alle Bauteile gesteckt werden können. Somit ist die Verwendung von Kabeln zwischen den einzelnen Modulen nicht notwendig. Eine Verkleinerung der Datenlogger wird dadurch ermöglicht.

4.8. Gehäuseentwicklung für den Datenlogger

Unterschiedliche Witterungsverhältnisse tragen dazu bei, dass die Messeinrichtung geschützt werden muss. Feuchtigkeit und Regen können zur Korrosion führen. Starke Winde tragen kleine Partikel mit sich, die bei Kollision mit den elektrischen Bauteilen die Messwerterfassung zerstören können. Bei einem Fall aus einer bestimmten Höhe ist die Wahrscheinlichkeit kleiner, die Messwerterfassung beim Aufprall auf den Boden zu beschädigen oder zu zerstören. Kurzschlüsse oder Verletzungen durch das Schneiden an den Bauteilen oder durch Stromschläge werden ebenso minimiert. Außerdem können elektrostatische Entladungen (ESD), die die Platine oder deren Bauteile beeinträchtigen oder zerstören können, verhindert werden.

Sobald die Baugröße für den Datenlogger feststeht, kann ein Gehäuse dafür konstruiert und entwickelt werden.

4.9. Gehäuse für den Server

Im Gegensatz zu dem Datenlogger wird ein Server in den meisten Fällen in einem Gebäude betrieben. Der Server ist dadurch weniger den Witterungsverhältnissen ausgesetzt als der Datenlogger. Physikalische Beschädigungen können trotzdem auftreten genauso wie Menschen sich an den scharfen Kanten und Ecken am Raspberry Pi verletzen können. Ein wichtiger Aspekt ist das verhindern der elektrostatische Entladung (ESD), die zur Beeinträchtigung oder Zerstörung der Platine oder deren Bauteile führen kann. die die Platine Allein aus diesen Gründen ist ein Gehäuse für den Server notwendig.

4.10. Kalibrierung des Datenloggers für genauere Messergebnisse

Die Versuche aus dem Kapitel 2.2 haben gezeigt, dass der Datenlogger schon jetzt relativ genau misst. Es sollte jedoch versucht werden, eine Fehlertoleranz von einem Prozent nicht zu überschreiten. Um bessere Messtoleranzen zu erzielen, muss keine zusätzliche Hardware angeschafft werden. Das Kalibrieren lässt sich per Software einfach und preiswert gestalten.

4.11. Softwareoptimierung

Die Software ist eine mächtige Komponente, die aber auch viel Fehlerpotential liefert. Sehr schnell finden falsche Berechnungen statt, es werden Variablen verwechselt oder ähnliches. In diesem Fall tragen z.B. Timer zu Fehlfunktionen bei. Der Quelltext sollte gründlich auf Fehler oder Mängel durchsucht werden.

4.12. Integration einer Fehleranalyse

Fehler werden zur Zeit nicht vom System in irgendeiner Weise erkannt. Deshalb wäre ein Fehleranalysesystem sinnvoll. Das System könnte beispielsweise Standardfehler melden oder in einem Fehlerspeicher zum Auslesen abspeichern. Schäden oder Ausfälle an der Windkraftanlage oder am Messsystem könnten damit verhindert und lokalisiert werden. Fehlerlokalisierungen werden damit beschleunigt. Die Integration einer Watchdog zur Fehlerbereinigung würde den Datenlogger, wie auch den Server stabiler laufen.

4.13. Erweiterung der Strommessung

Sollte die Windkraftanlage modifiziert werden oder aus irgendwelchen Gründen die Leistung der Anlage sich steigern, wird höchstwahrscheinlich ein größerer Strom produziert. Die Strommessung ist nur bis 30A ausgelegt. Um die Messelektronik zu schützen und größere Ströme zu messen, ist es ratsam, ein anderes Strommessmodul zu verwenden. In der Abbildung 4.1 wird ein Beispielmodul gezeigt, das bis 50A messen kann. Außerdem sind damit Messungen im Wechselstrombetrieb realisierbar.

4.14. FMEA

Zur Vermeidung von Fehlern sollte so früh wie möglich eine FMEA durchgeführt werden. In den folgenden Abbildungen 4.2 und 4.3 wird je eine FMEA für den Datenlogger und Rasspberry-Pi 3 erstellt.

FMEA Formblatt Seite 1

Erstellt durch: Viktor Felker
Datum: 17.04.17

Prozess-FMEA
Produkt-FMEA X

Prozess- / Produktname: Datenlogger
Überarbeitet durch / am:

Nr.	Fehlerort / Fehlermerkmal	Potentielle Fehler	Fehlerfolge	Fehlerursache	derzeitige Kontroll-maßnahmen	Derzeitiger Zustand A*	B*	E*	RPZ*	Verbesserter Zustand Verant-wortlich	durchgeführte Maßnahmen	A*	B*	E*	RPZ*
1.	Schraubanschluss für Strom, Spannung	Keine Verbindung	Keine Messung von Strom, Spannung	Keine physikalische Verbindung	Sichtprüfung	1	7	1	7	Viktor Felker	Sichtprüfung	1	7	8	56
2.		Flasche Verbindung	Negative Messwerte	Falsche Verpolung	Serielles debuggen	7	5	5	175	Viktor Felker	Anschlüsse beschriften	2	5	3	30
3.	SD-Modul	Keine Kommunikation	Daten speichern, auslesen nicht möglich	Kabel nicht richtig gesteckt	Sichtprüfung	4	10	3	120	Viktor Felker	Shield-Platine	1	10	8	80
4.				Wackelkontakt	Sichtprüfung	8	9	6	432	Viktor Felker	Shield-Platine	2	9	8	144
5.				Programmcode	Sichtprüfung	3	8	3	72	Viktor Felker	Sichtprüfung	2	8	3	48
6.	SIM-Modul	Keine Kommunikation	Daten senden, empfangen nicht möglich	Kabel nicht richtig gesteckt	Sichtprüfung	4	10	3	120	Viktor Felker	Shield-Platine	1	10	8	80
7.				Wackelkontakt	Sichtprüfung	8	9	6	432	Viktor Felker	Shield-Platine	2	9	8	144
8.				Programmcode	Sichtprüfung	3	8	6	144	Viktor Felker	Sichtprüfung	2	8	6	96
9.	Drehzahlsensor	Kein Kommunikation	Keine Datenempfang	Stecker verdreht	Sichtprüfung	7	6	3	126	Viktor Felker	Verpolsichere Steckerverbindung	1	6	1	6
10.				Programmcode	Sichtprüfung	3	6	3	54	Viktor Felker	Sichtprüfung	2	6	3	36
11.				Wackelkontakt	Sichtprüfung	8	5	5	200	Viktor Felker	Shield-Platine	2	5	8	80

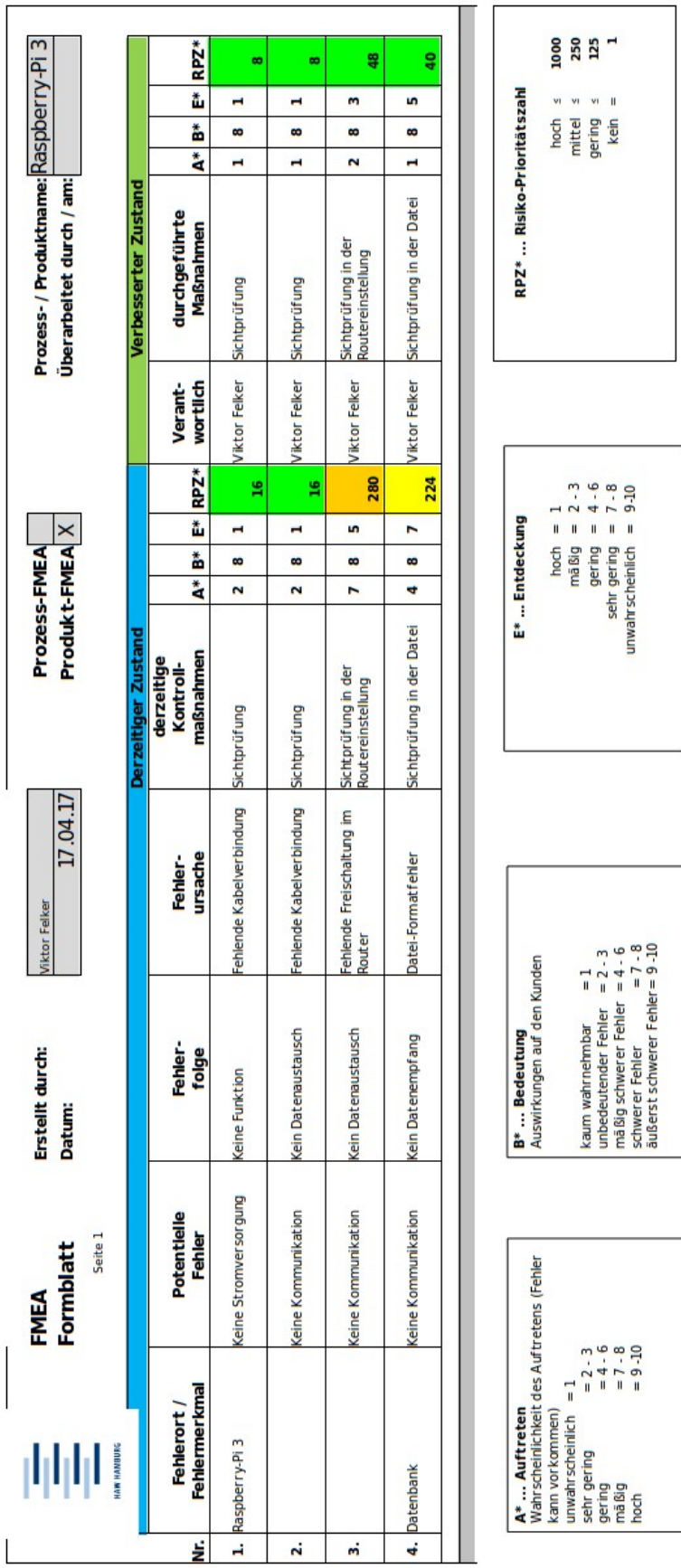
A* ... Auftreten
Wahrscheinlichkeit des Auftretens (Fehler kann vorkommen)
unwahrscheinlich = 1
sehr gering = 2 - 3
gering = 4 - 6
mäßig = 7 - 8
hoch = 9 -10

B* ... Bedeutung
Auswirkungen auf den Kunden
kaum wahrnehmbar = 1
unbedeutender Fehler = 2 - 3
mäßig schwerer Fehler = 4 - 6
schwerer Fehler = 7 - 8
äußerst schwerer Fehler = 9 -10

E* ... Entdeckung
hoch = 1
mäßig = 2 - 3
gering = 4 - 6
sehr gering = 7 - 8
unwahrscheinlich = 9-10

RPZ* ... Risiko-Prioritätszahl
hoch ≤ 1000
mittel ≤ 250
gering ≤ 125
kein = 1

Abbildung 4.2.: FMEA für den Datenlogger



FMEA Formblatt — Seite 1

Erstellt durch: Viktor Felker
Datum: 17.04.17

Prozess-FMEA
Produkt-FMEA X

Prozess- / Produktname: Raspberry-Pi 3
Überarbeitet durch / am:

Nr.	Fehlerort / Fehlermerkmal	Potentielle Fehler	Fehler-folge	Fehler-ursache	derzeitige Kontroll-maßnahmen	A*	B*	E*	RPZ*	Verant-wortlich	durchgeführte Maßnahmen	A*	B*	E*	RPZ*
						Derzeitiger Zustand						Verbesserter Zustand			
1.	Raspberry-Pi 3	Keine Stromversorgung	Keine Funktion	Fehlende Kabelverbindung	Sichtprüfung	2	8	1	16	Viktor Felker	Sichtprüfung	1	8	1	8
2.		Keine Kommunikation	Kein Datenaustausch	Fehlende Kabelverbindung	Sichtprüfung	2	8	1	16	Viktor Felker	Sichtprüfung	1	8	1	8
3.		Keine Kommunikation	Kein Datenaustausch	Fehlende Freischaltung im Router	Sichtprüfung in der Routereinstellung	7	8	5	280	Viktor Felker	Sichtprüfung in der Routereinstellung	2	8	3	48
4.	Datenbank	Keine Kommunikation	Kein Datenempfang	Datei-Formatfehler	Sichtprüfung in der Datei	4	8	7	224	Viktor Felker	Sichtprüfung in der Datei	1	8	5	40

A* ... Auftreten
Wahrscheinlichkeit des Auftretens (Fehler kann vorkommen)
unwahrscheinlich = 1
sehr gering = 2 - 3
gering = 4 - 6
mäßig = 7 - 8
hoch = 9 -10

B* ... Bedeutung
Auswirkungen auf den Kunden
kaum wahrnehmbar = 1
unbedeutender Fehler = 2 - 3
mäßig schwerer Fehler = 4 - 6
schwerer Fehler = 7 - 8
äußerst schwerer Fehler = 9 -10

E* ... Entdeckung
hoch = 1
mäßig = 2 - 3
gering = 4 - 6
sehr gering = 7 - 8
unwahrscheinlich = 9-10

RPZ* ... Risiko-Prioritätszahl
hoch ≤ 1000
mittel ≤ 250
gering ≤ 125
kein = 1

Abbildung 4.3.: FMEA für den Raspberry-Pi 3

5. Eigene Optimierungen

5.1. Datenübertragung an den Server

Der fehlgeschlagene Versuch im Kapitel 2.2.5 erfordert eine Analyse und Fehlerbehebung der Datenübertragung an den Server.

Zunächst wird der Server hardwareseitig untersucht. Der Raspberry Pi 3 wird mit einem Steckernetzteil für die Spannungsversorgung verbunden. Da einige LED's auf dem Raspberry Pi 3 aufleuchten, ist davon auszugehen, dass der Raspberry Pi 3 mit Spannung versorgt wird. Das Patchkabel ist ordnungsgemäß an den Raspberry Pi 3 angeschlossen und die Gegenseite an den Modem-Router. Grüne LED's an den Anschlussbuchsen leuchten auf, was für eine erfolgreiche Hardwareverbindung steht. Zum Testen wird ein Monitor mit einem HDMI-Kabel angeschlossen, sowie eine USB-Tastatur und eine USB-Maus.

Folgend wird die Softwareseite des Servers untersucht. Der Programmcode wird auf Fehler analysiert.

Folgende Punkte sind dabei optimiert worden:

- Alle Skripte und Programme, die abgearbeitet sind, schließen sich und sind nicht mehr verfügbar. Ein Dienst, der automatisch Skripte und Programme zu vorgegebenen Zeiten starten kann, ist eine *crontab* oder auch *Cron-Daemon* genannt. Die bisherige Einstellung ist auf 23:00 Uhr programmiert. Wird also eine Datei an den Server gesendet, arbeitet der Raspberry seine Skripte einmal ab und wartet dann bis 23.00 Uhr, um die Skripte neu zu starten. Sollte eine Datenübertragung mehr als einmal täglich stattfinden, wird es nicht möglich sein sich mit dem Server zu verbinden. In der Abbildung 5.1 wird eine *crontab* im Terminal des Raspberry-Pi 3 dargestellt.

  Zur Umgehung dieses Problems wird der Neustart der Skripte auf alle 3 Minuten programmiert. Eine bessere Lösung ist, einen Dienst zu programmieren, der permanent läuft.

- In der Datenbank ist der Ordner *piggot* eingefügt, worin die empfangenen Daten gespeichert werden. Der Ordner ist mit dem *Location* Verzeichnis verbunden, wo die Standorte der Windräder in Breiten- und Längengraden gespeichert sind. Da das

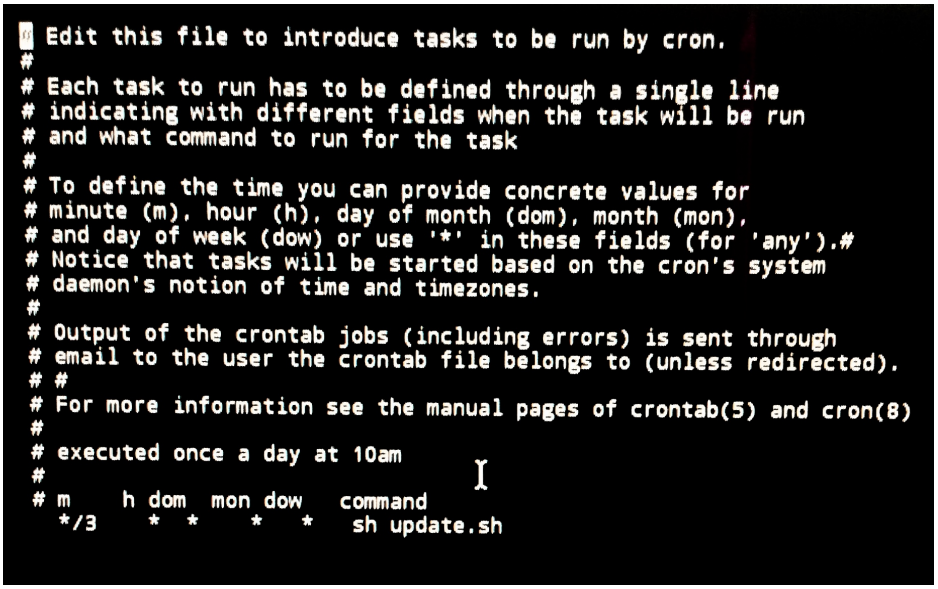


Abbildung 5.1.: Ausgeführter crontab

Übertragen der Daten an die Datenbank bisher gescheitert ist, wird zu Testzwecken ein neues Verzeichnis *test* in der SQL-Datenbank angelegt. Hinzu kommt eine Anpassung an den Webserver, um auf die Daten über die IP-Adresse zuzugreifen. Das Verzeichnis *test* dient zunächst als Ersatz für das Verzeichnis *piggot*. Ab diesem Zeitpunkt ist die Datenübertragung immer erfolgreich, deswegen wird auf das Verzeichnis *piggot* verzichtet.

Die Software des Datenloggers wird an die Software des Servers angepasst, dabei sind folgende Punkte verändert und optimiert worden:

- Für die Übertragung muss in der CSV-Datei ein bestimmter Ablauf für jede gespeicherte Datenzeile befolgt werden. Stimmt der Ablauf nicht, sieht der Server die Daten als unbrauchbar an und speichert sie nicht in der Datenbank ab, obwohl die Datei erfolgreich an der Server übertragen wurde. In der Abbildung 5.2 ist der Programmcode zum Speichern der Werte auf der SD-Karte dargestellt. Die Windrad-ID und das Datum wurde hinzugefügt.

  WINDRAD-ID (ganze Zahl); DATUM (JJJJ-MM-TT); UHRZEIT (HH:MM:SS); MESSGRÖßE-1 (00.00); MESSGRÖßE-2 (00.00); MESSGRÖßE-3 (00.00); MESSGRÖßE-4 (00.00); MESSGRÖßE-5 (00.00); MESSGRÖßE-6 (00.00); -

  Beispiel: *1; 2017-07-03; 20:23:54; 23.4; 49.9; 3.06; 3.13; 24.00; 5; -*

_04_07_17 | errorlog.cpp | errorlog.h | gprs.cpp | gprs.h | rotation.cpp | rotation.h | sd.cpp § | sd.h | sens

```
void Sd::saveData(char* logfile)
{
  char time_h[8];
  char date[8];
  File file;
  timer->getTime(time_h);
  timer->getDate(date);

  if (file = SD.open(logfile, FILE_WRITE))
    {
      file.print(WIND_TURBINE_NUMBER);                    // WINDRAD-ID
      file.print(";");
      file.print(date);                                   // DATUM
      file.print(";");
      file.print(time_h);                                 // UHRZEIT
      file.print(";");
      file.print(sens->getTemperature());                 // TEMPERATUR
      file.print(";");
      file.print(sens->getHumidity());                    // FEUCHTIGKEIT
      file.print(";");
      file.print(sens->getRpsAverage());                  // WINDRADDREHZAHL (DURCHSCHNITT)
      file.print(";");
      file.print(sens->getCurrentAverage());              // STROM (DURCHSCHNITT)
      file.print(";");
      file.print(sens->getVoltageAverage());              // SPANNUNG (DURCHSCHNITT)
      file.print(";");
      file.print(sens->getWindAverage());                 // WINDGESCHWINDIGKEIT (DURCHSCHNITT)
      file.print(";");
      file.print(" - ");
      file.print("\n");
      file.close();
    }
    else Serial.println("failed, saveData");
}
```

Abbildung 5.2.: CODE: SD-Speicherfunktion

- Das Datum muss das richtige Format haben. Derzeit sieht das Format folgend aus *TT.MM.JJJJ* und muss auf das Format *JJJJ-MM-TT* geändert werden. In der Abbildung 5.3 werden die geänderten Programmzeilen dargestellt.

```
void Timer::getDate(char* date)
{
  //sprintf(date, "%.2i.%.2i.%.2i", day(), month(), year());          //Original
  sprintf(date, "%.2i-%.2i-%.2i", year(), month(), day());
}
```

Abbildung 5.3.: CODE: Datumsfunktion

- Wird eine neue Datei auf der SD-Karte erzeugt, sorgt eine Funktion in der Abbildung 5.4 dafür, dass die Datei einen Namen erhält.

  WINDRAD-ID, TAG, MONAT, JAHR

  Beispiel: *1040717.csv*

```
void Sd::getFilename(char* filename)
{
  //sprintf(filename, "%d%02d%02d%02d.csv", WIND_TURBINE_NUMBER, day(), month(), (year() & 0x1F));   // Original
  sprintf(filename, "%d%02d%02d%02d.csv", WIND_TURBINE_NUMBER, day(), month(), (year() - 2000));
}
```

Abbildung 5.4.: CODE: Erzeugung eines Dateinamens

Der Dateiname besteht aus einer Zusammenführung unterschiedlicher Zahlen. Folgend wird die Zusammensetzung veranschaulicht. Es werden nur die Jahre von 00 bis 99 verwendet, z.B. im Jahr 2017, soll nur die 17 am Ende des Dateinamens verwendet werden. Um das zu realisieren, wurde das Jahr zuerst hexadezimal mit einem AND-Operator überschrieben und das ist falsch.

	0111	1110	0001
&	0000	0001	1111
=	0000	0000	0001

In der oberen Beispielrechnung wird der Fehler ersichtlich dargestellt. Übersetzt man das Ergebnis, erhält man den dezimalen Wert 01, obwohl das erwartete Ergebnis 17 ist. Um auf das richtige Ergebnis zu kommen, wird das Jahr um 2000 subtrahiert.

- Bisher wurden beim Erstellen einer CSV-Datei einige Parameter immer direkt in die Datei geschrieben. Diese werden nicht benötigt und sind daher ausgeklammert. Die Abbildung 5.5 veranschaulicht den geänderten Code.

```
void Sd::newCsv(char* logfile)
{
  char h[8];
  char date[8];
  timer->getTime(h);
  timer->getDate(date);
  getFilename(logfile);
  File file;
  if (!SD.exists(logfile))
  {
    file = SD.open(logfile);
  /*
    if (file = SD.open(logfile, FILE_WRITE))                    // Auskommentiert, da nicht benötigt
    {
      file.print(time);
      file.print(";");
      file.println(date);
      file.println(String(HEAD));
      file.close();
    }*/
   file.close();
  }
}
```

Abbildung 5.5.: CODE: Erzeugung einer neuen CSV-Datei

Bei einem Langzeittest von über 6 Stunden wird bewiesen, dass die Datenübertragung über mehrere Stunden zuverlässig funktioniert. Alle Messdaten sind erfolgreich im 5 minütigem Takt ohne einen Ausfall übersendet worden. Das spricht für eine zuverlässige Funktionalität des Messsystems, denn alle dabei gemessenen und übersendeten Datenwerte stimmen mit den tatsächlichen Datenwerten überein.

5.2. Implementierung eines Akkus

Dank einem Akku kann der Datenlogger auch dann betrieben werden, wenn das Windkraftwerk keine elektrische Energie liefert. Damit der Akku richtig zum Einsatz kommt, muss ein Schaltkreis integriert werden, der den Akku wieder auflädt, sobald das Windkraftwerk elektrische Energie liefert. Dabei sollte auf mehrere Aspekte, die in den Anforderungen sind, eingegangen werden.

5.2.1. Anforderungen

- Aufladen des Akkus
- Bei vollem Akku Laderegelung abschalten
- Primäre Spannungsversorgung soll von der Windkraftanlage kommen
- Akku mit einem konstanten Strom laden
- Im Akkubetrieb den Akku abschalten, wenn die Endladegrenze erreicht ist
- Tiefenentladungsschutz hardwaretechnisch lösen
- Den Akku über einen Stecker mit der Platine verbinden
- Produkt möglichst preiswert herstellen
- Kompakte Bauweise

5.2.2. Entwicklung

Bei der Entwicklung einer Akku-Implementierung wird nicht nur die Laderegelung realisiert, sondern auch die Spannungsversorgung des Systems optimiert. Primär wird die benötigte Leistung von der Windkraftanlage genutzt und sekundär von dem Akku. In der Abbildung 5.6 wird der entworfene Schaltkreis mit den Bauteilen dargestellt.

Für die Auswahl der Bauteile wird auf die Internetseite *www.conrad.de* zurückgegriffen. Conrad ist einer der bekanntesten Händler für Elektronikartikel. Auf der Internetseite von Conrad lassen sich nicht nur die Bauteile heraussuchen sondern auch die Preise für die einzelnen Bauteile können verglichen und die Verfügbarkeit eingesehen werden.

Die verwendeten Bauteile sind auf die SMD-Baugröße bezogen und herausgesucht. Damit lässt sich Platz auf der Platine schaffen. Preislich sind die Bauteile in der SMD-Baugröße meistens günstiger als die Standard Baugröße. Unterschiedliche Sicherheitsmechanismen

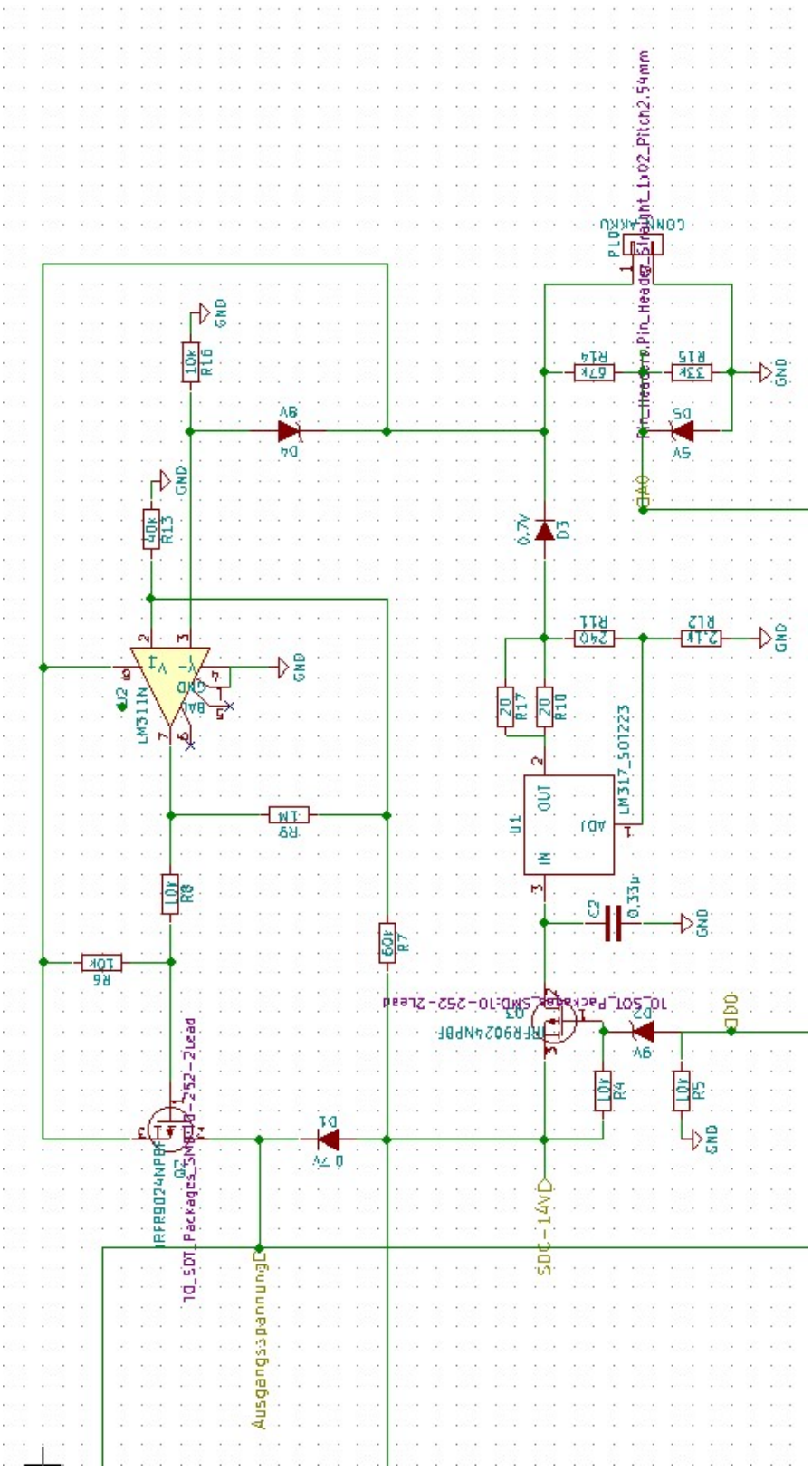


Abbildung 5.6.: Schaltkreis zur Implementierung eines Akkus

werden in die Schaltung miteingebaut, um die Bauteile von einer Zerstörung durch z.B. Überspannung oder Kriechströme zu schützen. Für solche Sicherheitsmechanismen werden verschiedene Bauteile benötigt, die Geld und Platz kosten, aber auch die Stabilität, die Sicherheit und die Langlebigkeit des Systems deutlich erhöhen.

Auf folgende Anforderungen wird dabei geachtet:

- Kompakte Bauweise durch SMD-Baugröße
- Preiswert durch die SMD-Bauweise und Vergleich der Bauteile auf der Conrad Homepage

Laderegelung des Akkus

Um den Akku zu schonen, sollte der Akku mit einem konstanten Strom geladen werden. Dafür wird der IC LM317 verwendet. Aus dem Datenblatt des LM317 wird der Schaltkreis für die Batterieaufladung verwendet. In der Abbildung 5.6 kann die Schaltung eingesehen werden. Der konstante Ladestrom kann dabei über den Widerstand R10 eingestellt werden. Die Ladespannung wird über die Widerstände R11 und R12 eingestellt. Anhand des Ohmschen Gesetzes können die Widerstände berechnet werden.

Das Ohmsche Gesetzt:

$$\frac{U}{I} = R$$

Der Arduino-Mega überwacht die Laderegelung des Akkus, damit der Akku nicht überladen wird. Zur Spannungsmessung des Akkus dient der Spannungsteiler, der Parallel zum Akku geschaltet ist. Immer wenn der Arduino eine Messung vornimmt, schaltet der Arduino über den MOSFET-Transistor vor dem LM317 die Laderegelung ab. Somit wird die tatsächliche Spannung am Akku gemessen. Falls der Mikrocontroller feststellt, dass der Akku noch nicht voll geladen ist, schaltet der Arduino über den MOSFET-Transistor die Laderegelung wieder zu.

Mit diesem Systementwurf werden folgende Punkte erfüllt:

- Laden mit einem konstanten Strom
- Messung der Akkuspannung
- Abschaltung der Laderegelung bei voller Akkuspannung

Spannungsversorgung Generator/Akku

Die Spannungsversorgung der ganzen Platine soll effizient genutzt werden. Eine Windkraftanlage erzeugt Elektrizität, die die Menschen in Afrika nutzen möchten. Es ist daher darauf zu achten, dass der Datenlogger möglichst wenig Leistung zum Arbeiten benötigt. Dadurch bleibt den Menschen mehr Energie zur Verfügung.

Die Platine wird entweder mit der Spannung vom Generator oder vom Akku versorgt. Primär wird immer der Generator aus der Windkraftenergie dafür genutzt, den Datenlogger mit Elektrizität zu versorgen. Liefert die Windkraft keine Elektrizität, wird der Akku automatisch zugeschaltet. Der Baustein IC LM311 vergleicht dafür die Spannungen aus dem Generator und dem Akku. Sobald die Spannung des Generators zusammenbricht, ist die Spannung des Akkus größer und der LM311 schaltet den MOSFET Q2. Somit wird der Datenlogger mit elektrischer Energie versorgt. Ein Ausfall des Datenloggers wird somit vermieden.

Ein solches System kann entweder per Software oder per Hardware realisiert werden. Die Hardwarelösung wird in diesem Fall vorgezogen. Bei einem Stromausfall oder einem Akkuaustausch kann die Funktion des Systems gewährleistet werden. Außerdem kann ein Abschalten des gesamten Systems bei Erreichung der Entladespannung realisiert werden.

Um ein System zu realisieren, wird der IC LM311 benutzt. Der Baustein ist ein Komparator, der zwei Spannungen miteinander vergleicht und wenn Spannung A größer ist als Spannung B wird der OUTPUT geschaltet. Die Tabelle 5.1 stellt die Spannungsversorgung der Platinen in Abhängigkeit vom Generator und dem Akku dar.

Generator	**Akku**	**Vout**
0	0	0
0	X	Akku
X	0	Generator
X	X	Generator

Tabelle 5.1.: Wahrheitstabelle zur Spannungsversorgung zwischen dem Generator und dem Akku

Folgende Anforderung werden damit erfüllt:

- Primäre Spannungsversorgung durch den Generator
- Tiefenentladungsschutz hardwaretechnisch gelöst

Programmcodeerweiterung für die Laderegelung

Nachdem die Laderegelung hardwareseitig komplett fertiggestellt ist, muss noch die Software dafür angepasst werden. Nachstehend werden die hinzugefügten Programmzeilen dargestellt. Zunächst werden eine neue cpp Datei und die dazugehörige Header Datei, die *chargecon* benannt werden, erstellt. Am Arduino wird der Pin A0 für die Messung der Spannung am Akku und der Pin D6 für das Zu- und Ausschalten der Laderegelung verwendet.

_04_07_17 | chargecon.cpp | chargecon.h | errorlog.cpp | errorlog.h | gprs.cpp | gprs.h

```
void Chargecon::initCharger()
{
  pinMode(CHARGING, OUTPUT);
}


bool Chargecon::voltage ()
{
  digitalWrite (CHARGING, HIGH);                // Undock chargecontrol from charging
  delay(100);
  if (analogRead(BATT_VOLTAGE) > 650){          // If voltage > 9,6V
    return 0;
  }
  else return 1;
}


void Chargecon::charge_control ()
{
  if (voltage() == 1){
    digitalWrite (CHARGING, LOW);               // Undock chargecontrol from charging
  }
  else digitalWrite(CHARGING, HIGH);            // Dock chargecontrol to charging
}
```

Abbildung 5.7.: CODE: Eingefügte Datei chargecon.cpp und chargecon.h. Die Funktionen in der chargelog.cpp werden einmal alle dargestellt.

Mit dem Einfügen der beiden Dateien ist die Implementierung des Akkus noch nicht ganz vollbracht, denn ein neuer Timer muss erstellt werden, damit er zyklisch die Spannung des Akkus misst. In der Abbildung 5.8 wird in der timer.h ein neuer Wert definiert der *TM_BATT_CON* heißt und in der Abbildung 5.9 wird in der Main-Datei der Programmcode erweitert, damit die Timervariable auch ausgeführt wird.

5.3. Messwerterweiterung: Windgeschwindigkeit

Die Windgeschwindigkeit wird benötigt, um Aussagen über die Effizienz der Windkraftanlage zu treffen. Anhand der Messung der Windgeschwindigkeit wird es möglich sein, unterschied-

```
//Times
#define TM_READ_SENSOR 2 // s Timeout for reading senosr data
#define TM_SAVE_DATA 5  // sec //min Timeout for saving the sensor data
#define TM_SEND_DATA 3  // min Timeout for sending the .csv file to server
#define TM_BATT_CON 5       //s Timeout for checking battery voltage
#define TM_TIMEOUT_SLEEP 36 //h Timeout for putting the Arduino in sleep mode
```

Abbildung 5.8.: CODE: Eingefügte Datei chargecon.cpp und chargecon.h. Die Funktionen in der chargelog.cpp werden einmal alle dargestellt.

```
else if (timer.checkTimer(&tm_batt_con))                // If timer for chargecontrol activ expired
{
  chargecon.charge_control();
}
```

Abbildung 5.9.: CODE: Die Auführung der Laderegelung in der Main-Datei nach Ablauf des Timers.

liche Analysen zu treffen. Werte wie Wirkungsgrad, Windhäufigkeit und Windstärke tragen dazu bei, bessere Standorte und Höhen der Windkrafträder zu wählen.

Zur Verfügung steht ein Windmesser aus der Abbildung 5.10. Die Funktion des Windmessers wird mit zwei Stabmagneten und einem Reedkontakt realisiert. Sobald ein der beiden Magnete während der Drehung durch den Wind über den Reedkontakt gelangt, wird der Reedkontakt geschaltet. Ein Reedkontakt hält ca. 100 Mio. Schaltzyklen. Zunächst erscheint es viel, jedoch sind 100 Mio. sehr schnell erreicht durch den ständigen Windeinfluss. Hinzu kommt, dass ein Reedkontakt zwei mal pro Umdrehung schaltet. Aus diesen Gründen schafft der Windmesser nur ca. 50 Mio. Umdrehungen, bevor er defekt wird.

Um den Ausfall des Windmessers zu minimieren, wird der Windmesser mit einem Hallsensor optimiert. Dafür wird der Windsensor und die beiden Stabmagnete entfernt. Anstatt zwei Kabel müssen drei Kabel zum Sensor gezogen werden. Ein Hallsensor hat die Vorteile, dass er schmutzunempfindlich und berührungslos ist, also nicht physikalisch schaltet. Dadurch entfällt der physikalische Verschleiß und die Lebensdauer des Windmessers wird deutlich erhöht. Dabei muss darauf geachtet werden, dass die zwei neuen Permanentmagneten so montiert werden, dass das homogene Magnetfeld direkt über dem Hallsensor liegt, denn die Hallsensoren reagieren nicht auf inhomogene Magnetfelder. In der Abbildung 5.11 wird gezeigt, wie die Magnete im Windmesser verbaut werden. Mit einer Heißklebepistole werden die Permanentmagnete richtig gepolt an die innere Platte im Windmesser geklebt.

Der Hallsensor, der in der Abbildung 5.12 zu sehen ist, wird unter die Magneten fixiert und mit einem dreiadrigen Kabel verbunden. Ein Kabel liefert dabei eine 5V Spannung, ein weiteres den GND-Anschluss und das dritte Kabel liefert dem Arduino die Spannungsflanken zur Berechnung der Drehzahl. Der Hallsensor wird elektronisch geschaltet, weshalb er keinen

Abbildung 5.10.: Windmesssensor
https://www.floridaautomatedshade.com/v/vspfiles/photos/9050110-2.jpg

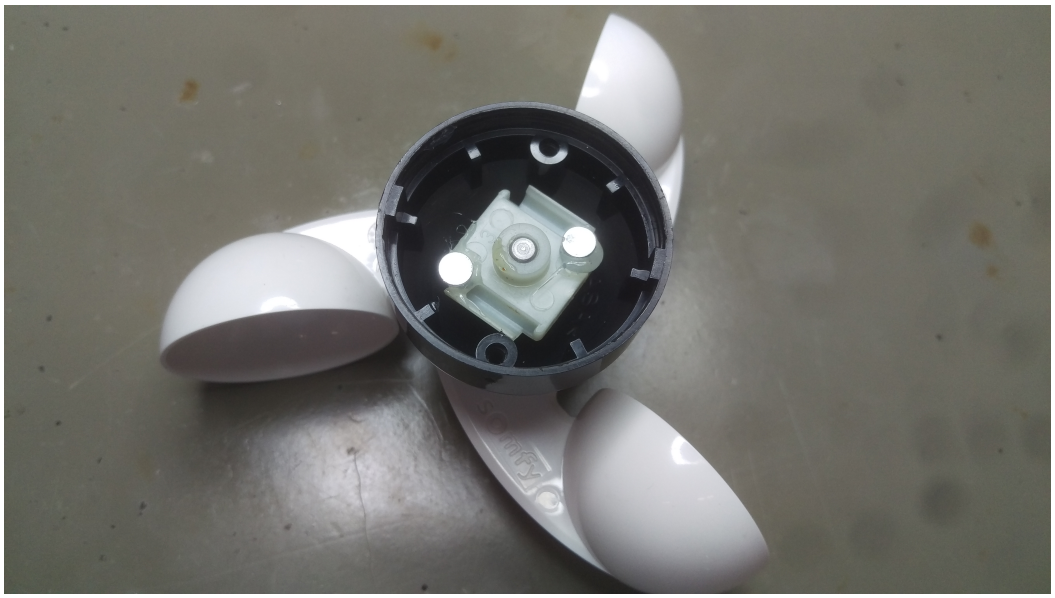


Abbildung 5.11.: Umgerüsteter Windsensor (Innenleben)

mechanischen Verschleiß aufweist. Preislich ist er z.B. bei Conrad schon ab 1,37 € erhältlich.

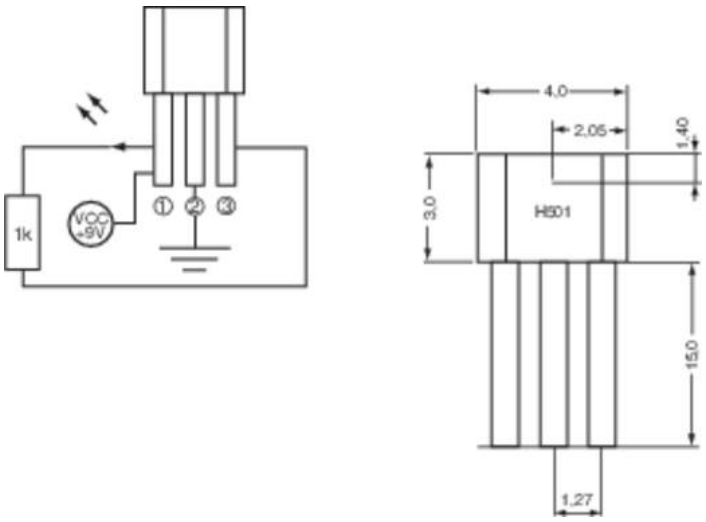


Abbildung 5.12.: Hallsensor PIC H501 zur Verwendung als Drehzahlsensor

Die Magnete werden aus einer Paper-Schachtel für Zigaretten verwendet, da die klein und von Rauchern kostenlos für den Versuch zur Verfügung gestellt wurden.

Um die hohe Genauigkeit des Sensor auch bei sehr hohen Drehzahlen zu beweisen, wird ein kleiner Versuch zur Darstellung gefahren. Dabei wird eine Drehzahl von über 1000 U/min eingestellt. Die Abbildung 5.13 stellt die Erfassung der Flanken an einem Oszilloskop dar. Bedenkt man, dass zwei Magnete am Windsensor angebracht sind, wird es deutlich, dass pro Minute über 2000 Flanken problemlos erkannt werden. Daraus resultiert, dass noch höhere Drehzahlen gefahren werden können.

Um die Messdaten des Windmessers zu erfassen, abzuspeichern und senden zu können, muss der Quellcode erweitert werden. Die Abbildung 5.14 veranschaulicht für die Windmessungen die Erweiterungen im Quellcode (sensor.h). Die Funktionen sind die gleichen, wie für die Drehzahl des Windrades, mit dem Unterschied, dass hier zwei Magnete verwendet werden und nicht drei. Für die Speicherung auf der SD-Karte wird der Programmcode ebenfalls in der sd.cpp erweitert, was in der Abbildung 5.2 ersichtlich ist. Der Abbildung 5.15 ist zu entnehmen, wie das Implementieren der Windmessung in der *loop* aussieht.

5.4. Kalibrierung der Messdatenerfassung

Die Messwerterfassung weist einen Messfehler von 2,99% bei Strommessungen und 1,63% bei Spannungsmessungen auf. Dies geht aus dem Versuch 2.2.2 hervor. Um die Messge-

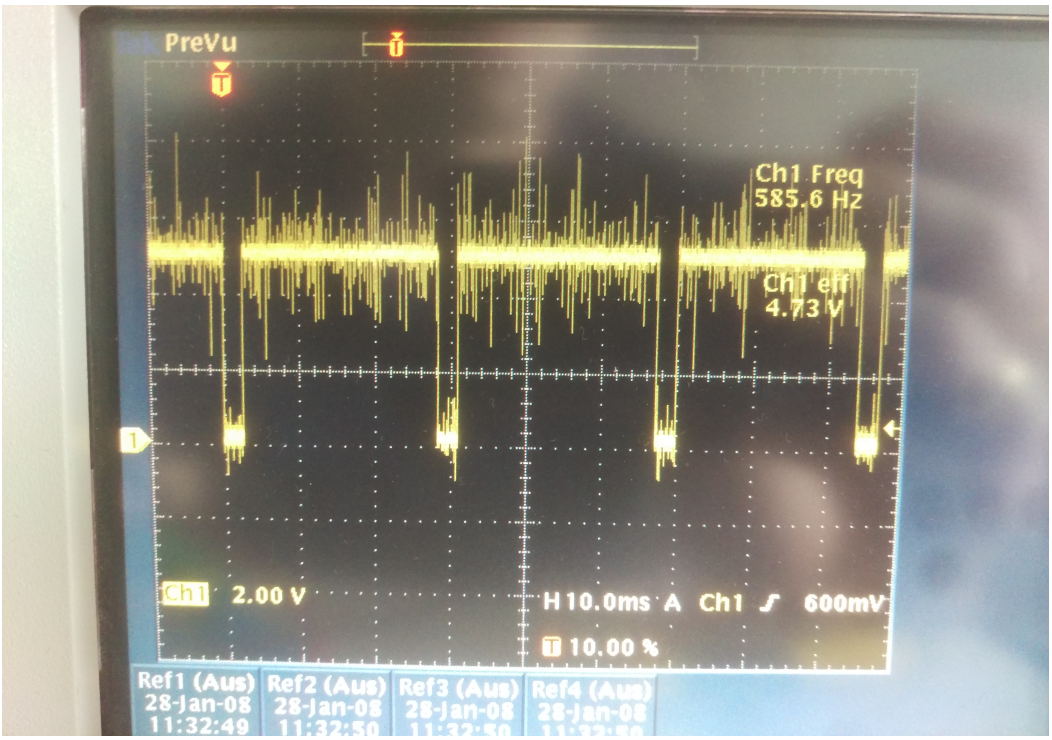


Abbildung 5.13.: Flankenerfassung eines Hall-Sensors mit einem Oszilloskop zur Drehzahlbestimmung

```
 // Saving the windspeed given to an array and the last time the wind sensor was read
double readWind(long int wind_start, unsigned int wind_rps);
 // Calculates the average rps from array
double getWindAverage();
```

Abbildung 5.14.: CODE: Windsensor Funktionserweiterungen

```
if(timer.checkTimer(&tm_read))
{
#if SENSOR_DEBUG
  char output[60];
  char tem[4];
  char hum[4];
  char rps[4];
  char cur[4];
  char vol[4];
  char win[4];
  dtostrf(sens.getTemperature(), 4, 2, tem);
  dtostrf(sens.getHumidity(), 4, 2, hum);
  dtostrf(sens.getRpsAverage(), 4, 2, rps);
  dtostrf(sens.getCurrentAverage(), 4, 2, cur);
  dtostrf(sens.getVoltageAverage(), 4, 2, vol);
  dtostrf(sens.getWindAverage(), 4, 2, win);
  char h[8];
  timer.getTime(h);
  sprintf(output, "%s | %s | %s | %s | %s | %s | %s ",
          h, tem, hum, rps, cur, vol, win);
  Serial.println(output);
#endif

  sens.readVoltage();
  sens.readCurrent();
  sens.readRps(hall_start, hall_rps);
  sens.readWind(wind_start, wind_rps);
}
```

Abbildung 5.15.: CODE: Windsensorerweiterung in der loop-Funktion

nauigkeit zu erhöhen, muss der Datenlogger kalibriert werden. Die Kalibrierung wird per Software vorgenommen. In der Datei *sensorik.cpp* findet die Berechnung der Strom- und Spannungsmessung statt. 5.16

```
// Output of currentsensor in mV/A
//#define MVPERAMP 0.068                                          // ORIGINALER WERT
#define MVPERAMP 0.067                                            // VERÄNDERTER WERT

// Offset from -30A to 0A
#define CURROFFSET 512

// Value to calculate digital input from AD-converter to current
#define TO_CURRENT 5.25/1024                                      // VERÄNDERTER WERT
//#define TO_CURRENT 4.97/1024                                    // ORIGINALER WERT

// Value to calculate digital input from AD-converter to voltage
//#define TO_VOLTAGE (5.03 * (995000.0+219500.0)/219500.0) /1023.0 // ORIGINALER WERT
#define TO_VOLTAGE (4.93 * (995000.0+219500.0)/219500.0) /1023.0   // VERÄNDERTER WERT
```

Abbildung 5.16.: CODE: Programmcodeausschnitt zur Kalibrierung der Strom- und Spannungswerte

Testweise wird dabei der Generator auf 400U/min eingestellt und der Lastwiderstand auf 3 Ohm. So lässt sich ein Referenzwert zum Versuch 2.2.2 erstellen. Sobald die Messwerte den realen Werten nahe sind, werden drei Versuche gefahren. Die Drehzahl wird von 800 U/min in 100er Schritten auf 300 U/min gefahren. Da mit den Lastwiderständen 3, 6, 10 Ohm gemessen werden soll, muss der Versuch je mit einem Widerstand gefahren werden. In der Tabelle 5.2 werden die kalibrierten Messwerte dargestellt.

Aus der Tabelle 5.3 wird ersichtlich, dass die Kalibrierung eine deutlich verbesserte Genauigkeit aufweist. Die Fehlertoleranz beträgt im Durchschnitt 0,77% für die Strommessung und 0,40% für die Spannungsmessung. Das sind deutlich geringere Werte als noch im Versuch 2.2.2.

5.5. Gehäuse für den Datenlogger

Ein Gehäuse ist aus mehreren Gründen sinnvoll. Zum einen soll ein Gehäuse die ganzen Bauteile vor physikalischen Schäden und vor den Witterungseinflüssen schützen. Zum anderen sollen damit auch Menschen vor Schäden durch die Elektronik bewahrt werden. Außerdem wird die Handhabung mit dem Gerät vereinfacht. Beim Anbringen des Gerätes an ein Windkraftwerk muss man sich keine Gedanken machen, dass die Bauteile eventuell verloren gehen oder beschädigt werden.

Instrumente			Datenlogger		Last
n in U/min	I in A	U in V	I in A	U in V	R in Ohm
800,00	1,46	13,91	1,43	13,86	10 Ohm
700,00	1,46	13,90	1,43	13,82	
600,00	1,29	12,30	1,26	12,23	
500,00	1,21	11,50	1,16	11,41	
400,00	1,17	11,16	1,15	11,09	
300,00	1,18	11,15	1,14	11,07	
800,00	2,41	13,72	2,41	13,66	6 Ohm
700,00	2,40	13,68	2,39	13,63	
600,00	2,07	11,81	2,07	11,74	
500,00	1,94	11,07	1,91	11,02	
400,00	1,81	10,30	1,77	10,23	
300,00	1,81	10,27	1,73	10,22	
800,00	4,58	13,21	4,59	13,21	3 Ohm
700,00	4,57	13,18	4,58	13,19	
600,00	3,71	10,69	3,72	10,69	
500,00	3,52	10,13	3,51	10,13	
400,00	3,07	8,83	3,08	8,79	
300,00	3,04	8,78	3,04	8,77	

Tabelle 5.2.: Aufnahme der kalibrierten Strom- und Spannungswerte

	Instrumente	Datenlogger	Fehler	Fehler in Prozent
Strom	2,35	2,37	0,02	0,77
Spannung	11,60	11,64	0,05	0,40

Tabelle 5.3.: Fehlertoleranzberechnung nach Kalibrierung der Strom- und Spannungswerte

Die Entwicklung eines Gehäuses für den Datenlogger würde diese Thesis sprengen, daher werden nur einige Punkte aufgeführt, die bei der Entwicklung des Gehäuses berücksichtigt werden sollten.

- Bei der Entwicklung der Shield-Platine wurden Befestigungspunkte zur Anbringung am Gehäuse bereits vorgesehen
- Separates Fach für den Akku vorsehen
- Gehäuse staub- und regenfest entwickeln, dabei könnten Dichtungen zum Einsatz kommen
- Montagemöglichkeit zur Anbringung am Windrad vorsehen
- Wasserdichte Anschlüsse für Kabel
- Strom-/Spannungsmessanschluss an der Platine anpassen
- Anschluss für den Switch an der Platine anpassen

5.6. Servergehäuse

Für den Server ist ein Gehäuse aus denselben Gründen wie für den Datenlogger sinnvoll. Da der Raspberry Pi 3 ein sehr weit verbreitetes Gerät ist, sind schon einige Gehäuse dafür auf dem Markt vorhanden. Viele dieser Gehäuse sind sehr preiswert zu erwerben. Recherchen im Internet weisen darauf hin, dass die Preise oft unter 10 Euro liegen. Bei *www.amazon.de* kann z.B. das offizielle Raspberry Pi 3 Gehäuse in schwarz für 9,79 Euro erworben werden. Das Gehäuse wird in der Abbildung 5.17 dargestellt. Teileweise werden im Internet noch weit aus günstigere Gehäuse zum Verkauf angeboten. Aus diesem Grund ist es sinnvoller das Gehäuse zu kaufen, als dies selbst zu entwerfen und zu produzieren.

5.7. Shield-Platine

Bei der Shield-Platine handelt es sich um ein Plug & Play Bauteil. Alle benötigten Schaltkreise sollen darauf sein. Sämtliche weitere Bauteile, die als Komponenten eingekauft werden, wie der Step-Down-Converter oder der Arduino-Mega, sollen darauf gesteckt werden können. Eine solche Platine bietet viele Vorteile:

- Plug & Play
- Keine Kabel / wenige Kabel

Abbildung 5.17.: Gehäuse schwarz für Raspberry Pi 3
https://images-eu.ssl-images-amazon.com/images/I/31xpySxXGYL.jpg

- Ein Bauteil
- Kompakte Bauweise
- Geordnete Bauteile
- Professionelle Optik
- Schutz vor einem Wackelkontakt
- Leichteres ersetzten von Bauteilen

Anforderungen

- Baugröße von 160mm x 100mm nicht überschreiten
- Alle Bauteile sollen auf einer Seite angebracht werden
- SMD-Bauform
- Pinheader für den Anschluss anderer Komponenten
- Extra Anschluss für erweiterte Sensorik
- Switch, um den Datenlogger Ein-/Ausschalten zu können
- Anschlüsse für Kabel, die vom Datenlogger abgehen

Entwicklung

Die Entwicklung der Platine beinhaltet sehr viel Wissen in der Elektronik und auch viel Zeit. Schaltkreise, wie die Laderegelung, werden auf die Platine gelötet. Sämtliche Bauteile müssen dafür aus einer riesigen Datenbank verglichen und rausgesucht werden. Datenblätter der Bauteile müssen gelesen und interpretiert werden. Sicherheiten gegen Kriechströme und Überspannungen müssen mitberücksichtigt werden.

Für die Entwicklung der Platine wird auf die Software KiCad zurückgegriffen. KiCad ist eine OpenSource Software, die für alle kostenlos zur Verfügung steht. Mit KiCad können Schaltpläne erstellt und auch ein Platinendesign entworfen werden. Außerdem lassen sich damit Gerber-Dateien erstellen, die die Platinenhersteller benötigen, um die Platinen herzustellen.

Weitere Features sind:

- Erstellung bis zu 8 Layer
- Komplett kostenlos
- 3D-Ansicht der Platine
- 3D-Ansicht der Bauteile
- Schaltplan Erstellung
- Platinenerstellung
- Ausgabe verschiedener Dateiformate, unter anderem von Gerber-Dateien
- Keine Platinengrößen Begrenzung
- Sehr viele Hilfen, Tutorials und Videos im Internet zu finden, wie z.B. auf YouTube

Die entwickelte Laderegelung aus dem Kapitel 5.2 wird dabei mit auf die Platine integriert. Alle Bauteile, die zur Laderegelung gehören, werden fest auf der Platine verlötet. Damit ist die Laderegelung ein fester Bestandteil der Platine.

Sämtliche Komponenten, wie die Laderegelung, SIM800L-Modul, Arduino-Mega und andere werden miteinander direkt verbunden. Dem Schaltplan ist zu entnehmen, welche Komponente mit welchen verbunden werden. Das Überlegen, wie verbindet man welche Komponente und wohin gehört welches Kabel, entfällt. Ein Falschanschluss wird somit vermieden.

Unterschiedliche Sicherheitsmechanismen werden verwendet, um die Schaltkreise vor Beschädigungen zu schützen. Als Überspannungsschutz werden Zenerdioden verwendet. Die Kriechströme, die durch die einzelnen Bauteile fließen und z.B. die Eingänge am Mikrocontroller zerstören können, werden dank den Widerständen vermieden. Um die Bauteile vor der

Selbstzerstörung zu schützen, werden einige Bauteile, wie z.B. der Widerstand R10, größer dimensioniert. Kleinere Bauteile lassen oft kein Spielraum Leistungen über einem Watt.

Design

Das Design der Platine wird mit KiCad erstellt. Dabei muss auf vieles geachtet werden. Das Ziel ist es, eine Shield-Platine für alle benötigten Komponente zu schaffen und dabei zusätzlich entworfene Schaltkreise fest verlötet auf der Platine anzuordnen.

Da das eine Shield-Platine ist, müssen die Anschlusspins für die einzelnen zugekauften Komponenten, wie der Arduino-Mega, spiegelverkehrt angebracht werden. Gleiches gilt für alle anderen Komponenten. Für die meisten Komponenten ist ein Datenblatt verfügbar, jedoch selten sind die Pinabstände in den Datenblättern bemaßt. Dies muss für die Designanordnung mit einer Schieblehre mühsam vermessen werden. Viel Spielraum bleibt dabei nicht, da die Anschlüsse fest verlötet werden und sich dadurch kaum bewegen lassen.

Bei den Leiterbahnen muss auf die Breite geachtet werden. Für die Signale reicht die Standardbreite für Leiterbahnen von 0,25mm aus. Nicht alle Leiterbahnen sind jedoch nur für die Signale vorhanden. Die Strommessung soll bis zu 30A messen können. Um solche Ströme über die Platine fließen lassen zu können, wird die Leiterbahnstärke für die Messungen von Strom und Spannung mit einer Breite von 1,5mm festgelegt. Alle anderen Komponenten müssen ebenfalls mit Spannungen versorgt werden. Dafür wird eine Leiterbahnstärke von 0,75mm gewählt.

Die Anordnung der Bauteile bereitet einige Schwierigkeiten. Unterschiedliche Aspekte sind dabei zu beachten.

- Wärmeableitung
- EMV-Störungen
- Vermeidung von Leiterbahnkreuzungen
- Einfacher Zugriff zur SIM-Karte, SD-Karte
- Antenne von dem SIM800L-Modul möglichst weit oben anbringen
- Alle Außenanschlüsse unten an dem Datenlogger

Da auf der Platine reichlich Komponenten und Schaltkreise Platz finden müssen, wird eine Baugröße von 160mm x 100mm (Europaplatine) gewählt. Das ist eine genormte Platinengröße, wobei alle Elemente auf eine Seite passen und trotzdem noch Platz für mindestens einen zusätzlichen Anschluss für spätere Anwendungen finden.

Da die Leiterbahnen sich öfter kreuzen, als das eine einseitige Platine ermöglicht, wird die Platine zweilagig entworfen. Brücken oder 0Ohm Widerstände werden somit nicht benötigt. In der Abbildung 5.18 und 5.19 wird die Platine in 3D von beiden Seiten dargestellt. Die Leiterbahnen sind gut ersichtlich, auch die Durchkontaktierung wird dabei ersichtlich. Aus der 2D-Ansicht in der Abbildung 5.20 wird die Komplexität der Platine erkennbar.

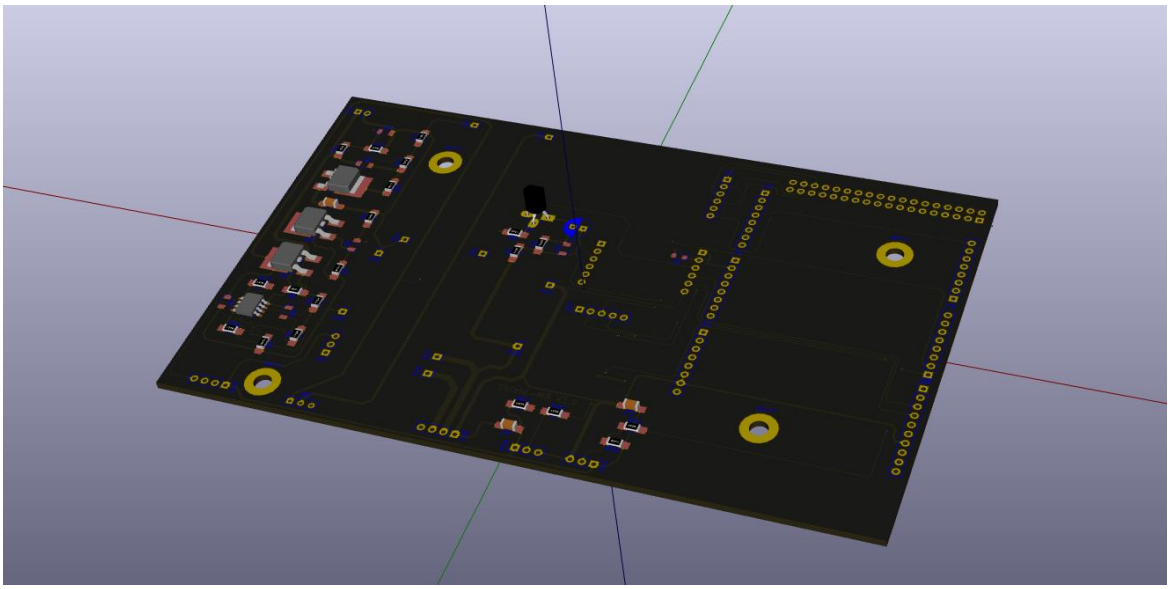


Abbildung 5.18.: 3D Platinenentwurf (Toplayer)

Allerdings wird bei dem Platinendesign nur selten an die Anschlussflächen für das Gehäuse der Platine gedacht. In diesem Fall werden 4 Punkte für die Anbringung auf einem Gehäuse vorgesehen. An diesen Punkten wird die Platine einfach auf dem Gehäuse verschraubt. Dies sollte bei der Gehäuseentwicklung mitberücksichtigt werden.

Die Abbildung 5.21 stellt die einzelnen Komponente auf der Platine angeordnet dar. Um nicht immer auf den Schaltplan zu gucken, werden die einzelnen Anschlüsse in der folgenden Auflistung einzeln beschrieben und erläutert. Dabei wird immer von der Draufsicht auf die Platine aus der Abbildung 5.21 ausgegangen.

Weiterentwicklung

Dieser Platinenentwurf ist ein Prototyp, der im Rahmen dieser Arbeit entwickelt wurde. Meistens werden Fehler oder Mängel erst durch einen Prototypen entdeckt. Wie alle Prototypen enthält auch dieser Mängel, auf die eingegangen werden muss.

Folgend werden festgestellte Verbesserungen aufgelistet:

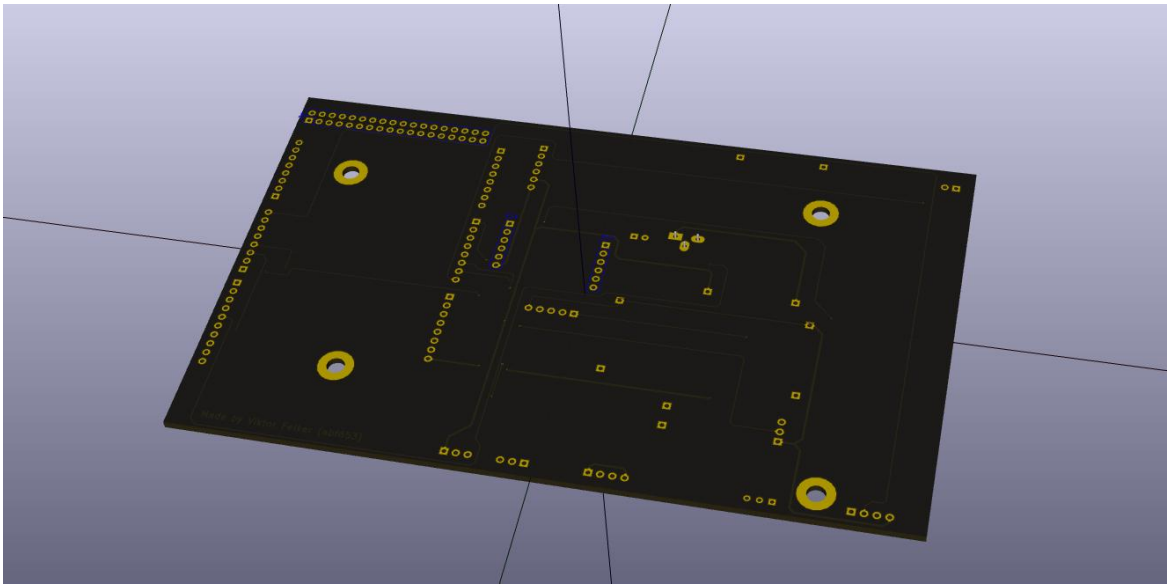


Abbildung 5.19.: 3D Platinenentwurf (Bottumlayer)

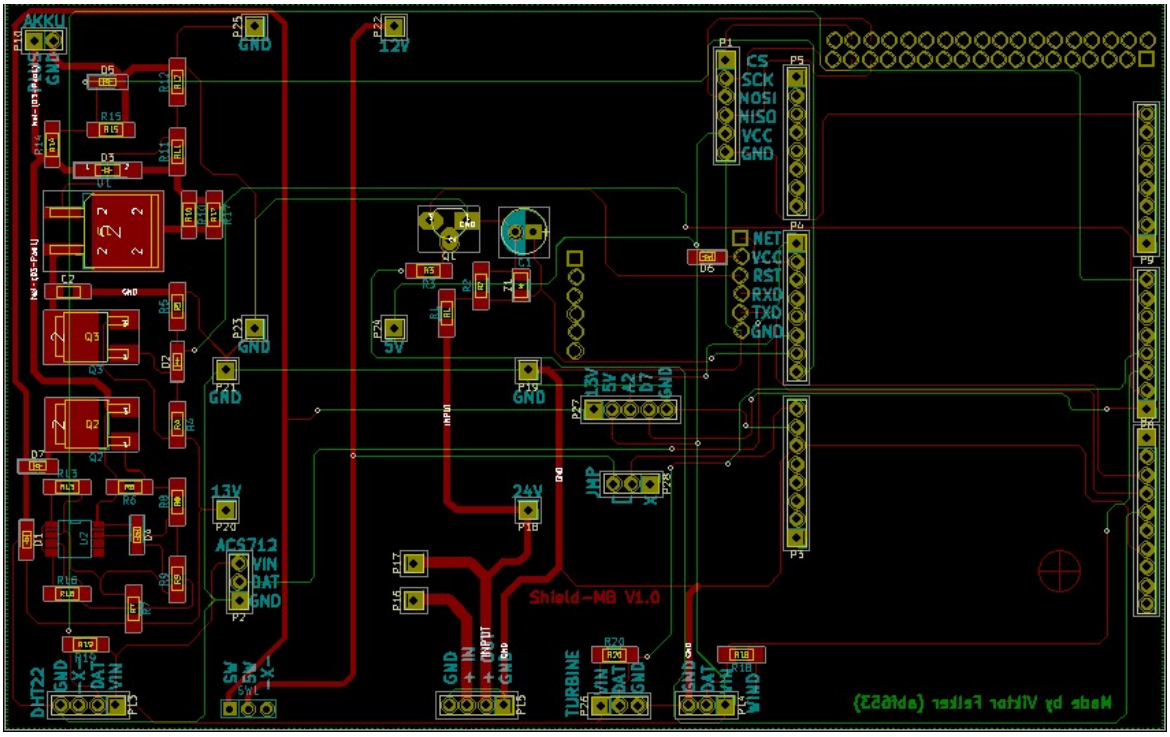


Abbildung 5.20.: 2D-Ansicht der Platine

Anschluss	PIN-Belegung
Akkuanschluss	PLUS | GND
Zusatzanschluss	13,3V | 5V | A2 | D7 | GND
DHT22	GND | NULL | D0 | 5V
Switch	OUT | IN | NULL
Messanschluss	GND | IN | OUT | GND
Drehzahlmessung Windrad	5V | D3 | GND
Drehzahlmessung Windsensor	GND | D2 | 5V

Tabelle 5.4.: Beschreibung der PIN-Belegung aus der Abbildung 5.21. PIN-Beschreibung aus der Draufsicht von links nach rechts.

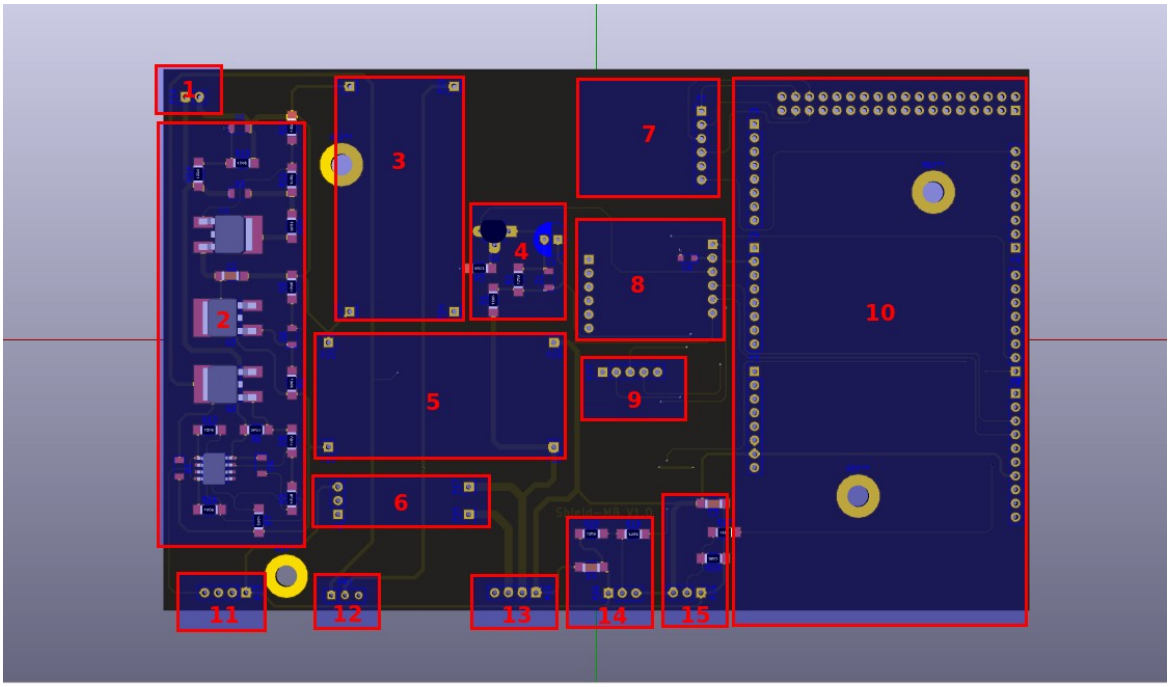


Abbildung 5.21.: Bauteileanordnung auf der Platine. 1:Akkuanschluss, 2:Laderegelung, 3:Step-Down-Converter 24V auf 14V, 4:Spannungsteiler/Spannungsmessung, 5:Step-Down-Converter 13,3V auf 5V, 6:ACS712/Strommessung, 7:SD-Modul, 8:SIM800L, 9:Zusatzanschluss für erweiterte Komponente, 10:Arduino Megra, 11:DHT22 Temperatur-/Feuchtigkeitssensor, 12:Switch, 13:Strom-/Spannungsmessanschlüsse, 14:Drehzahlmessung Windrad, 15:Drehzahlmessung Windgeschwindigkeit

1. Widerstand R10 mit 10 Ohm ausführen, eventuell zwei 20 Ohm Widerstände parallel schalten zur besseren Wärmeverteilung.
2. Kondensator C1 von 100 µF auf 33 µF ändern.
3. Die Pinabstände für die Step-Down-Converter passen nicht exakt.
4. Footprint für den LM317 ändern, da er zu klein ausfällt.
5. Jumper für den Arduino verwenden.
6. Pins für die Messanschlüsse vergrößern. Dieses sollte bei der Entwicklung des Gehäuses Gegenstand sein.
7. Hallsensorschaltungen optimieren => R18, R20 als 1k Ohm auslegen und R17, R19 mit 0 Ohm verwenden bzw. komplett entfernen und eine Leiterbahn ziehen, sowie die Kondensatoren C4 und C3 entfernen.
8. Switchkontaktanschlüsse an den Switch anpassen. Dieses sollte bei der Entwicklung des Gehäuses Gegenstand sein.
9. An der Diode D6 fällt eine zu geringe Spannung ab, Zerstörungsgefahr für das SIM-Modul.
10. Einsatz einer Diode am Drain des MOSFET's Q2.
11. Beschriftung der Pin's/Anschlüsse

Alle Mängel bis auf Punkte 6 und 8 sind behoben worden.

5.8. Softwareoptimierung

Die Software ist ein wesentlicher Bestandteil des Messsystems. Die gesamte Steuerung und Berechnung wird über die Software geregelt. Bei der Analyse des Quellcodes sind einige Schwächen des Codes sichtbar geworden.

Ein fataler Fehler ist, den Timer auf der Basis der Systemlaufzeit zu beziehen. Die Funktion *millis()* gibt die Systemlaufzeit zurück. Der Mikrocontroller zählt seine Systemzeit ca. 50 Tage lang und dann fängt er wieder von vorne an zu zählen.

Zur Berechnung einer Sekunde wird z.B. folgender Ansatz verwendet:

$$x - y > z$$
$$y = x$$

x = Systemzeit, y = Zählvariable, z = eine Sekunde

Immer wenn die Differenz zwischen x und y mehr als eine Sekunde beträgt, schaltet der Programmcode diese Berechnung von neu an und setzt dabei den y-Wert auf den vorherigen x-Wert. Daraus erschließt sich, dass nach 50 Tagen die Zeitdifferenz negativ erscheint und somit die Berechnung nicht mehr aufgeht. Es resultiert eventuell ein Systemabsturz oder die Drehzahlberechnung findet nicht mehr statt. Dieses Problem ist über den Betrag gelöst worden. Aus der Abbildung 5.22 wird der optimierte Quellcode ersichtlich.

```
void rotationCounter()
{
  hall_state++;
  if (abs(millis() - hall_start >= 1000))                    // Optimized function with "absolute value"
  {
    hall_rps = hall_state;
    hall_state = 0;
    hall_start = millis();
  }
}


void initWind()
{
  //Attach the interrupt for the hall-rotationcounter.
  pinMode(WINDSPEED_IN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(WINDSPEED_IN), windCounter, FALLING);

  wind_rps = 0;
  wind_state = 0;
}


void windCounter()
{
  wind_state++;
  if (abs(millis() - hall_start) >= 1000)                    // Optimized function with "absolute value"
  {
    wind_rps = wind_state;
    wind_state = 0;
  }
```

Abbildung 5.22.: CODE: Timeroptimierung mit der Betragfunktion

Die Berechnungen zur Messung der Drehzahl und der Windgeschwindigkeit müssen optimiert werden. Bei der Drehzahlmessung sollen drei Permanentmagnete verwendet werden, deswegen werden die Peaks am Ende jedes Zyklus durch drei dividiert. Das gleiche gilt für die Windmessung, wobei hier zwei Magnete verwendet werden, weshalb am Ende jedes Zyklus durch zwei dividiert werden muss. In der Abbildung 2.14 wird diese Optimierung für die Umdrehungen des Windrades ersichtlich.

6. Zusammenfassung und Ausblick

Nach sämtlichen Versuchstests und Optimierungen ist der Datenlogger sehr gut gelungen. Für ein Lowbudget System ist die Messeinrichtung ein richtiger Erfolg.

Trotz einiger Startschwierigkeiten bei den Versuchen und der Inbetriebnahme, haben die Versuche gezeigt, dass die Messwerterfassung die Messwerte genau erfasst. Die Ausarbeitung der Datenübertragung hat den Grundstein für Funktionsfähigkeit des Gesamtsystems gelegt. Durch die Kalibrierung konnte die Messgenauigkeit noch weiter verbessert werden. Das Erstellen der Shield-Platine hat einen großen Beitrag zu diesem Projekt beigetragen. Dadurch konnten viele Fehler und Probleme behoben werden. Die Optik der Platine sieht sehr wertig aus, was einen professionellen Eindruck auf das Projekt wirft. Der modulare und einfache Aufbau der Platine bewirkt ein einfacheres Arbeiten und Forschen mit dem Datenlogger. Durch die Softwareoptimierung wurden noch einige Fehler behoben.

Für die Zukunft sollte das System weiter ausgearbeitet werden. Die ausstehenden Verbesserungsvorschläge sollten noch integriert werden, vor allem die Fehleranalyse. Obwohl die Entwicklung einer Fehleranalyse sehr viel Zeit beansprucht, würde sie das spätere Arbeiten mit dem Datenlogger, sowie die Analyse der Windräder deutlich erleichtern.

Sobald alle Punkte ausgearbeitet und fertiggestellt sind, kann das ganze System in dem realen Betrieb eingesetzt werden. Die meisten Punkte sind bereits jetzt schon ausgearbeitet und fertiggestellt, wodurch das System nahezu einsatzbereit ist.

Literaturverzeichnis

[Bartmann 2014] BARTMANN, Erik: *Die elektronische Welt mit Arduino entdecken*. O'Reilly Verlag GmbH & Co. KG, 2014. – ISBN 978-3955611156

[Baumann 2015] BAUMANN, Peter: *Ausgewählte Sensorschaltungen*. Springer, 2015. – ISBN 978-3-658-08557-5

[Dalmaris 2016] DALMARIS, Peter: *Kicad Like a Pro*. Tech Explorations, 2016. – ISBN B01CYB95A8

[Engelhardt 2011] ENGELHARDT, Stephan: *Direkte Leistungsregelung einer Windenergieanlage mit doppelt gespeister Asynchronmaschine*. Shaker, 2011. – ISBN 978-3-8322-9864-7

[Gossner 2016] GOSSNER, Stefan: *Grundlagen der Elektronik. Halbleiter, Bauelemente und Schaltungen*. Shaker, 2016. – ISBN 978-3826588259

[Hau 2014] HAU, Erich: *Windkraftanlage*. Springer, 2014. – ISBN 978-3-642-28876-0

[Hering 2014] HERING, Ekbert: *Elektronik für Ingenieure und Naturwissenschaftler*. Springer, 2014. – ISBN 978-3-642-05498-3

[Kofler 2016] KOFLER, Michael: *Linux Kommandoreferenz: Shell-Befehle von A bis Z*. Rheinwerk Computing, 2016. – ISBN 978-3836237789

[Kofler 2017] KOFLER, Michael: *Raspberry Pi: Das umfassende Handbuch, komplett in Farbe*. Rheinwerk Computing, 2017. – ISBN 978-3836258593

[Proell 2015] PROELL, Stefan: *MySQL: Das umfassende Handbuch*. Rheinwerk Computing, 2015. – ISBN 978-3836237536

[Retzbach 2008] RETZBACH, Ludwig: *Akkus und Ladetechniken*. Franzis, 2008. – ISBN 978-3772341700

[Weydanz 2017] WEYDANZ, Wolfgang: *Moderne Akkumulatoren richtig einsetzen*. epubli, 2017. – ISBN 978-3741892592

[Wolf 2014] WOLF, Jürgen: *C++: Das umfassende Handbuch, aktuell zum Standard C++11*. Galileo Computing, 2014. – ISBN 978-3-836-22021-7

A. Inbetriebnahme der Datenloggers und des Servers

In diesem Kapitel wird auf die Inbetriebnahme des ganzen Messsystems eingegangen. Das Kapitel dient als Leitfaden beziehungsweise als Anleitung, wie das Messsystem angeschlossen und konfiguriert wird und was dabei zu beachten ist.

A.1. Inbetriebnahme des Servers

Der Server ist bereits vorkonfiguriert, daher ist kein weiterer Eingriff am Server selbst notwendig.

Folgende Punkte sind nacheinander durchzuführen:

1. Den Server an eine Spannungsquelle anschließen.
2. Verbindung des Servers mittels eines Patchkabels am Netzwerk.
3. Freigabe für den Server im Netzwerk einrichten.
4. Eine zusätzliche Freigabe für den Port 2500 einrichten.

Sind alle Schritte erfolgreich durchgeführt kann über die IP-Adresse auf den Server zugegriffen werden.

A.2. Inbetriebnahme des Datenloggers

Der Datenlogger ist vorkonfiguriert, benötigt dennoch einige Parameter, um die Daten richtig an den Server zu übertragen. In der folgenden Auflistung wird die Inbetriebnahme und die Konfiguration beschrieben. Um die Anschlüsse besser zu veranschaulichen wird die Abbildung 5.21 als Bezugsquelle verwendet. Alle Punkte müssen nacheinander durchgeführt werden.

1. Die SIM-Karte in das SIM-Modul einstecken.
2. Die SD-Karte in das SD-Modul einstecken.
3. Den Datenlogger am Windrad befestigen.
4. Den Drehzahlsensor am Windrad befestigen.
5. Den Drehzahlsensor an den Datenlogger am Punkt 14 anschließen.
6. Den Windsensor am Windrad befestigen.
7. Den Windsensor an den Datenlogger am Punkt 15 anschließen.
8. Den Akku in das vorgesehen Fach im Gehäuse legen.
9. Den Akku am Datenlogger an den Punkt 1 anschließen.
10. Für die Strommessung das Plus-Kabel, das zur 24V-Batterie führt, in zwei Hälften trennen. Die eine Hälfte des Plus-Kabels, die zum Windrad führt, mit dem zweiten Pin von links vom Anschluss 13 verbinden. Die andere Hälfte des Plus-Kabels, die zur 24V-Batterie führt, mit dem dritten Pin von links vom Anschluss 13 verbinden.
11. Bei der Spannungsmessung sind zwei Varianten möglich:
    a) An den Ground-Anschluss der zur 24V-Batterie führt wird ein zusätzliches Kabel angeschlossen, das entweder an den ersten oder vierten Pin des Anschlusses 13 angeschlossen wird.
    b) Der Ground-Anschluss der zur 24V-Batterie führt in zwei Hälften trennen. Die eine Hälfte an den ersten Pin von links des Anschlusses 13 anschließen und die zweite Hälfte an vierten Pin von links des Anschlusses 13 anschließen.
12. In der Software in der Datei *gprs.h* die IP-Adresse des Servers eingeben. Die Variable *SERVERIP* steht dafür zur Verfügung.
13. In der Software den APN für das Verwendete Mobilfunknetz in *gprs.h* eingeben. Die Variable *APN* steht dafür zur Verfügung.
14. Die Windradnummer in der Software in der Datei *sd.h* eingeben. Die Variable *WIND_TURBINE_NUMBER* steht dafür zur Verfügung.
15. Die Timer zum Daten lesen, speichern und senden in der Software sind auf default eingestellt. Bei einer Anpassung stehen in der Datei *timer.h* die Variablen *TM_READ_SENSOR, TM_SAVE_DATA* und *TM_SEND_DATA* zur Verfügung. In der Main-Datei können diesen Variablen dann noch als Sekunden, Minuten, Stunden oder Tage eingestellt werden. Aus der Abbildung A.1 ist zu entnehmen, an welchem Punkt in der Main-Datei dies veränderbar ist.

```
/* Timer initialisation. This timers are used to trigger the the events for
 * meassurement, data saving and transmitting of the csv file.
 * F_SEC = seconds, F_MIN = minutes, F_HOUR = hours F_DAYS = days */
timer.setTimer(&tm_read, TM_READ_SENSOR, F_SEC);
timer.setTimer(&tm_save, TM_SAVE_DATA, F_SEC);
timer.setTimer(&tm_send, TM_SEND_DATA, F_MIN);
timer.setTimer(&tm_batt_con, TM_BATT_CON, F_SEC);
```

Abbildung A.1.: Timer Konfiguration in der Main-Datei. Rot eingekreist sind die veränderbaren Variablen. *Timer configuration in the main file. Red are the variable variables.*

B. Commissioning the data logger and the server

This chapter describes the commissioning of the entire measuring system. The chapter serves as a guide on how the measuring system is connected and configured and what has to be observed.

B.1. Commissioning the Server

The server is already preconfigured, so no further intervention is necessary on the server itself.

The following points are to be carried out one after the other:

1. Connect the server to a voltage source.
2. Connect the server using a patch cable to the network.
3. Enable the network share server.
4. Set up an additional share for port 2500.

If all steps are successful, the server can be accessed via the IP address.

B.2. Commissioning the data logger

The data logger is preconfigured, but still requires some parameters to transfer the data to the server correctly. The commissioning and configuration are described in the following list. To better visualize the connections, use the figure 5.21 as a reference source. All points must be performed one after the other.

1. Insert the SIM card into the SIM modul.
2. Insert the SD Card into the SD modul.

3. Mount the data logger on the wind wheel.
4. Mount the speed sensor on the wind wheel.
5. Connect the speed sensor to the data logger at point 14.
6. Mount the wind sensor on the wind wheel.
7. Connect the wind sensor to the data logger at point 15.
8. Place the battery into the provided compartment in the housing.
9. Connect the battery to the data logger to point 1.
10. For the current measurement disconnect the positive cable, which leads to the 24V battery, in two halves. Connect one half of the positive cable, which leads to the wind wheel, to the second pin from the left of the connection 13. Connect the other half of the plus cable, which leads to the 24V battery, to the third pin from the left of the connection 13.
11. Two versions are possible for the voltage measurement:
    a) An additional cable, which is connected either to the first or fourth pin of the connection 13, is connected to the ground connection of the 24V battery.
    b) Seperate the ground cable of the 24V battery in two parts. Connect one part to the first pin from the left of the connector 13 and the second part to the fourth pin from the left of the connector 13.
12. In the software, enter the IP address of the server in the file *gprs.h*. The variable *SERVERIP* is available for this purpose.
13. In the software enter the APN for the mobile network used in *gprs.h*. The variable *APN* is available for this purpose.
14. Enter the wind turbine number in the software in the file *sd.h*. The variable *WIND_TURBINE_NUMBER* is available for this purpose.
15. The timers for data read, save and send in the software are set to default. For an adaptation, the variables *TM_READ_SENSOR, TM_SAVE_DATA* and *TM_SEND_DATA* are available in the file *timer.h*. In the main file these variables can still be set as seconds, minutes, hours or days. From the figure A.1 you can see at which point in the main file this can be changed.

C. Bauteileliste

In der folgenden Tabelle werden alle elektronischen Bauteile, die auf der Platine verlötet werden, aufgelistet. Anhand der Referenz kann das Bauteil auf der Platine zugeordnet werden. Zu beachten ist die Spalte Zusatzinfo, die bei der Bauteileauswahl auf wichtige Aspekte hinweist.

Referenz	Wert	Bauteil	Größe	Zusatzinfo
C1	33µF	Kondensator	THT: Radial	
C2	0,33µF	Kondensator	SMD: 1206	
D1	0,7V	Diode	SMD: DO-214	min. 2,0W
D2	9,1V	Zenerdiode	SMD: DO-214	
D3	0,7V	Diode	SMD: DO-214	
D4	8,0V	Zenerdiode	SMD: DO-214	
D5	5,0V	Zenerdiode	SMD: DO 214	
D6	1,1V	Diode	SMD: DO-214	min. 2,5W
D7	0,5V	Diode	SMD: DO-214	min. 1,5W
Q1	BC557	Transistor	THT: TO-92	
Q2	IRFR9024NPBF	P-MOSFET	SMD: TO-252	
Q3	IRFR9024NPBF	P-MOSFET	SMD: TO-252	
R1	100k Ohm	Widerstand	SMD: 1206	
R2	22k Ohm	Widerstand	SMD: 1206	
R3	470 Ohm	Widerstand	SMD: 1206	
R4	10k Ohm	Widerstand	SMD: 1206	
R5	10k Ohm	Widerstand	SMD: 1206	
R6	10k Ohm	Widerstand	SMD: 1206	
R7	60k Ohm	Widerstand	SMD: 1206	
R8	10k Ohm	Widerstand	SMD: 1206	
R9	1M Ohm	Widerstand	SMD: 1206	
R10	20 Ohm	Widerstand	SMD: 1206	
R11	240 Ohm	Widerstand	SMD: 1206	
R12	2,1k Ohm	Widerstand	SMD: 1206	
R13	40k Ohm	Widerstand	SMD: 1206	
R14	67k Ohm	Widerstand	SMD: 1206	
R15	33k Ohm	Widerstand	SMD: 1206	
R16	10k Ohm	Widerstand	SMD: 1206	
R17	20 Ohm	Widerstand	SMD: 1206	
R18	1k Ohm	Widerstand	SMD: 1206	
R19	10k Ohm	Widerstand	SMD: 1206	
R20	1k Ohm	Widerstand	SMD: 1206	
U1	LM317	IC	SMD: TO-263-2	
U2	LM311N	IC	SMD: SOIC-8-N	
Z1	5,1V	Zenerdiode	SMD: DO-214	

Tabelle C.1.: Verwendete elektrische Bauteile auf der Platine

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 20. Juli 2017	
Ort, Datum	Unterschrift