



DIPLOMARBEIT

Füllstandsevaluierung geschlossener Getränkebehälter

Ausgeführt im Schuljahr 2018/19 von:

Betreuer/Betreuerin:

Pia Wenty
Nina Nenning

Dipl.-Ing. Dr. Reinhard Steindl

St. Pölten, am 03. April 2019

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Pia Wenty

Nina Nenning

DIPLOMARBEIT DOKUMENTATION

Namen der Verfasser/innen	Pia Wenty, Nina Nenning
Jahrgang Schuljahr	5BHELS 2018/19
Thema der Diplomarbeit	Füllstandsevaluierung geschlossener Getränkebehälter
Kooperationspartner	Keine Angaben

Aufgabenstellung	Das Ziel ist die erfolgreiche Erfassung der Messdaten und deren Übertragung, sowie die anschließende Ausgabe und Auswertung der Daten in einer Webapplikation.
------------------	--

Realisierung	Zur Füllstandsmessung wird eine Wägezelle verwendet, welche in einem Getränkeuntersetzer verbaut ist. Die gemessenen Werte werden mithilfe eines ADC's verarbeitet und dem IO-Board Arduino Nano übergeben. Der Arduino Nano besitzt einen Atmega328P und formatiert die Messwerte, welche mit einem Ra02-LoRa-Modul auf einer Frequenz von 434MHz gesendet werden. Ein Raspberry Pi empfängt und verarbeitet diese Daten und speichert sie anschließend in eine Datenbank. Über eine Webapplikation erhält man Zugriff auf die gemessenen Daten. Dazu muss sich der Raspberry Pi im selben Netzwerk wie der Computer bzw. das Smartphone befinden, mit dem die Webseite aufgerufen wird.
--------------	--

Ergebnisse

Der entwickelte Umsetzter ist durch eine einmalig stattfindende Kalibrierung in der Lage, die Füllstandsdaten mit einer Genauigkeit von einem Milliliter zu erfassen.

Der Umsetzter wird über eine Wireless-Charging – Station geladen. Der Akkustand wird über die Messplattform des Umsetzers angezeigt, indem im Umsetzer verbaute LED's je nach Akkustand ihre Farbe ändern.

Der Umsetzter liegt vollfunktionsfähig vor. Das bedeutet er beinhaltet die Wägezelle, das LoRa-Modul inklusive Antenne, den Akku samt Akkuüberwachungsschaltung, zugehörigen PCB's und der Ladestation.

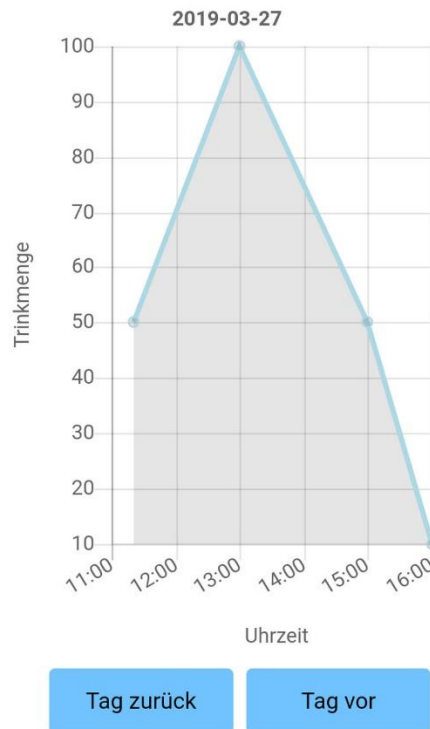
Der Umsetzer sollte nicht mit Flüssigkeit übergossen werden, ist aber gegen Flüssigkeitstropfen geschützt, insofern diese nicht zwischen Messplattform und Außenwand fließen.

Das Ra02-LoRa-Modul des Senders sendet die gemessenen Daten auf einer Frequenz von 434MHz und das Ra02-LoRa-Modul des Empfängers empfängt und speichert sie in eine Datenbank.

Der Raspberry Pi, der sowohl als Empfänger, als auch als Server fungiert, beinhaltet die Datenbank, in welche die empfangenen Daten gespeichert werden. Ebenso werden darin Benutzer angelegt.

Es können nur Personen, die bereits als Benutzer registriert sind, ihre Trinkdaten in Form eines Diagramms auf der Webapplikation einsehen. Um sich registrieren zu können, benötigt man die Seriennummer des Umsetzers. Diese ist im Lieferumfang des Umsetzers enthalten.

Typische Grafik, Foto etc.
(mit Erläuterung)



Im oben dargestellten Diagramm sieht man die am 27.03.2019 zu sich genommene Trinkmenge. Allerdings wird hierbei nicht die Summe aller gemessenen Werte angezeigt, sondern die Menge, die zu einem gewissen Zeitpunkt getrunken wurde. Werden keine Punkte angezeigt, wurde nicht getrunken. Also wurde in der Darstellung vier Mal getrunken: um 11:20 Uhr, um 13 Uhr um 15 Uhr und um 16 Uhr.



Diese Abbildung zeigt den Empfänger im Gehäuse. Darin ist der Raspberry Pi inklusive LoRa-Modul mit Antenne.



In diesem Bild sieht man die Ladestation des Untersetzers.



In dieser Abbildung sieht man den fertigen Untersetzter.

Teilnahme an Wettbewerben,
Auszeichnungen

Keine Angaben

Möglichkeiten der
Einsichtnahme in die Arbeit

Einsicht kann in der HTBLuVA St.Pölten, Waldstraße 3, 3100 St.
Pölten erlangt werden

Approbation
(Datum / Unterschrift)

Prüfer/Prüferin

Direktor/Direktorin
Abteilungsvorstand/Abteilungsvorständin

Dipl.-Ing. Dr. Reinhard Steindl

Dipl.-Ing. Wolfgang Kuran

	HÖHERE TECHNISCHE BUNDESLEHRANSTALT ST. PÖLTEN COLLEGE of ENGINEERING
	Department: Elektronische und Technische Informatik Educational focus: Wireless- & Embedded Systems

DIPLOMA THESIS DOCUMENTATION

Author(s)	Pia Wenty, Nina Nenning
Form Academic year	2018/19
Topic	Measuring the amount of liquid in a container
Co-operation Partners	Not specified

Assignment of Tasks	The target is the successful acquisition and transmission of the measured data, as well as displaying the data in a web application.
---------------------	--

Realisation	A load cell which is built into a coaster is used for measuring the liquid level. The measured values are processed using an ADC and transferred to the Arduino Nano IO board. The Arduino Nano holds an Atmega328P and it formats the measured values, which are transmitted with a Ra02-LoRa module at frequency of 434MHz. A Raspberry Pi receives and processes this data and then stores it into a database. A web application provides access to the measured data. The Raspberry Pi must be in the same network as the computer or smartphone used to access the website.
-------------	---

Results	The developed coaster can record the liquid level data with an accuracy of one millilitre after a one-time calibration. The coaster gets charged with wireless charging station. The battery level is displayed by LEDs installed in the coaster, changing the colour of the platform depending on the battery level. The coaster is fully functional. This means it contains the load cell, the LoRa module including antenna, the battery including battery monitoring circuit, associated PCB's and the charging station.
---------	---

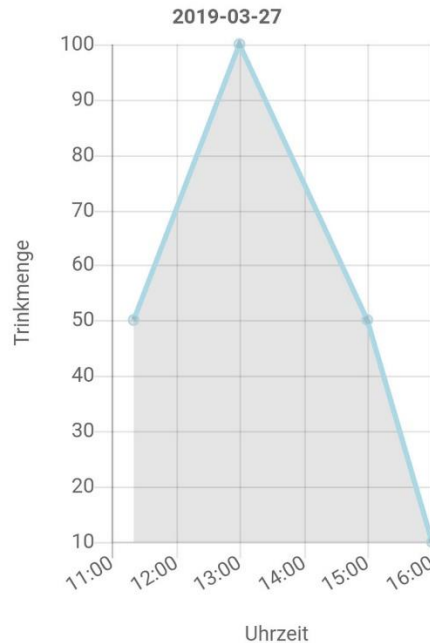
No liquid should be poured over the coaster, but it is protected against drops of liquid, as long as they do not drain off between the measuring platform and the outer wall.

The transmitter's Ra02 LoRa module transmits the measured data at a frequency of 434MHz and the receiver's Ra02 LoRa module receives and stores it into a database.

The Raspberry Pi, which acts both as a receiver and a server, contains the database in which the received data is stored. User data is also stored in the database.

Only people who are already registered as users can view their drinking data in a diagram on the web application. To be able to register, the serial number of the coaster is needed. The serial number is included in the delivery, when a coaster gets achieved.

Illustrative Graph, Photo
(incl. explanation)



Tag zurück

Tag vor

In the diagram above the amount of drunk liquid is shown, on 01.01.2019. But the diagram does not show the sum of the drunk liquid but only the amount which has been drunken at one time. If no data points are displayed, the amount of liquid drunk is zero. This means in the displayed diagram the user drank four times: at 11:20, 13:00, 15:00 and 16:00 o'clock.



The figure above shows the receiver inside its case.



The figure above shows the charging station of the device.



The figure above shows the coaster.

Participation in Competitions
Awards

Not specified

Accessibility of
Diploma Thesis

Insight can be achieved at the HBLuVA St.Pölten, Waldstraße 3, 3100
St. Pölten

Approval
(Date / Signature)

Examiner

Dipl.-Ing. Dr. Reinhard Steindl

Head of College/Department

Dipl.-Ing. Wolfgang Kuran

Inhaltsverzeichnis

1. Einleitung	1
1.1. Beschreibung der Rollen im Projekt	1
1.1.1. Projektmanager (PM)	1
1.1.2. Projektteammitglied (PTM)	1
1.1.3. Betreuer	1
1.2. Meilensteine	2
1.3. Aufteilung des Arbeitsbereichs	2
1.4. Projektplan	3
2. Individuelle Zielsetzung	5
2.1. Pia Wenty	5
2.2. Nina Nenning	6
3. Entwicklungsumgebung	7
3.1. L ^A T _E X & Overleaf	7
3.2. Alitum Designer & Autodesk Inventor	8
3.3. Arduino Studio	8
3.4. Verwendete Technologien	9
4. Übertragungstechnik	11
4.1. Low Power Wide Area Network	11
4.2. LoRaWAN	11
4.2.1. Unterschied LoRa und LoRaWAN	12
4.2.2. Netzwerktopologie	13
4.2.3. Sicherheit	13
4.3. Ra02-Modul	14
4.3.1. Spezifikationen	14
4.3.2. Pinbelegung & Bemaßung	14
4.4. LoRa-Modulation	15
4.4.1. LoRa-Nachricht	16
4.4.2. Berechnungen	17
4.4.3. Messungen	18
4.4.4. Zeitmessung	19
4.4.5. Chirp Messung	20

5. Sender	23
5.1. Funktionsbeschreibung	23
5.1.1. Arduino Nano	23
5.1.2. Wägezelle	24
5.1.3. DMS	25
5.1.4. Real Time Clock	26
5.2. Akku	27
5.2.1. Lithium Polymer	27
5.2.2. Schutzschaltung	28
5.2.3. Laden	29
5.2.4. Lebensdauer	29
5.3. Akku Management	30
5.3.1. Beschreibung	30
5.3.2. MCP73833	30
5.3.3. Laden	31
5.3.4. Entladen	31
5.4. Wireless Charging	32
5.4.1. Funktionsprinzip	32
5.4.2. Benötigter Eingangsstrom	34
5.4.3. Messungen	34
5.5. Schaltungen	35
5.5.1. Schaltungsstruktur	35
5.5.2. HEX711 - 24Bit ADC	36
5.5.3. Real Time Clock	37
5.5.4. LoRa	37
5.5.5. Akku Management	38
5.5.6. Gesamt Schaltung	40
5.6. PCB Entwicklung	41
5.6.1. Mechanische Anforderungen	41
5.6.2. Thermodynamische Anforderungen	41
5.6.3. Verwendete Bauteile	42
5.6.4. PCB Layout	43
5.7. Gehäuse Design	44
5.7.1. Untersetzer	44
5.7.2. Ladesstation	44
5.7.3. Gehäuse - Untersetzer	45
5.7.4. Statusanzeigen	47
5.7.5. Ladestation	48
5.7.6. Materialwahl	50
5.8. Software	51
5.8.1. Softwarestruktur	51
5.8.2. LoRa Konfiguration	52

5.8.3.	Packete Senden	53
5.8.4.	LoRa Energiesparmodus	53
5.8.5.	Akku Überwachung	54
5.8.6.	Messdaten	56
5.8.7.	EEPROM	57
5.8.8.	Sendealgorithmus	59
6.	Empfänger	61
6.1.	Aufsetzen des Raspberry Pi als Empfänger	61
6.1.1.	Erstmaliges Booten und Grundkonfigurationen	61
6.1.2.	SSH Verbindung zum Raspberry Pi	63
6.1.3.	Einstellen einer statischen IP-Adresse	63
6.1.4.	LoRa - Hardwarekonfiguration	64
6.1.5.	LoRa - Softwarekonfiguration	65
6.2.	Anzeige der aktuellen Konfiguration	65
6.3.	Empfangen und Verarbeiten der Daten	66
6.3.1.	Empfangen der gesendeten Daten	66
6.3.2.	Umwandlung des ASCII-Codes zu Text	66
6.3.3.	Splitten des Datenpakets	67
6.4.	Speichern der verarbeiteten Daten in die Datenbank	68
6.4.1.	Datenbankstruktur	68
6.4.2.	Apache installieren	69
6.4.3.	MySQL und phpMyAdmin installieren	69
6.4.4.	Installieren und Konfigurieren von php	70
6.4.5.	Speichern der Daten in die Datenbank	70
6.4.6.	Zugriff auf die Datenbank mit phpMyAdmin	71
6.5.	Schaltung	72
6.6.	PCB Entwicklung	72
6.6.1.	Verwendete Bauteile	73
6.6.2.	Routing	73
6.7.	Gehäuse Design	74
6.7.1.	Gehäuse Anforderungen	74
6.7.2.	Gateway - Gehäuse	74
6.7.3.	Gateway - Gehäusedeckel	75
6.7.4.	Zusammengebautes Gehäuse	76
7.	Webapplikation	77
7.1.	Aufbau der Webapplikation	79
7.1.1.	Navigation	80
7.1.2.	Bedingungen für die Registrierung	80
7.2.	Umsetzung in Code	81
7.2.1.	im Hintergrund ausgeführte Skripte	82

7.2.2. im Vordergrund ausgeführte Skripte	83
8. Progressive Webapp	91
8.1. Einstellungen im Projekt	91
8.1.1. Überspielen der App auf ein Smartphone	95
9. Benutzer Handbuch	97
9.1. Einleitung	97
9.2. Entsorgungshinweis	97
9.3. Richtlinien für einen sicheren und effizienten Verbrauch	97
9.4. Lieferumfang	98
9.5. Hinweise zum Akku	99
9.6. Erstmalige Inbetriebnahme	99
9.7. Zugang zur Webplattform	100
9.8. Navigation in der Webplattform	101
10. Betriebswirtschaftliche Kalkulation	105
10.1. Einzelkosten	105
10.1.1. Fertigungskosten der PCB's	105
10.1.2. Material- bzw. Bauteilkosten	105
10.1.3. Entwicklungskosten	106
10.2. Gemeinkosten (Overheads)	106
10.2.1. Test-, Bestückungs- und Kalibrierarbeiter	106
10.3. Amortisationsrechnung - Sender	109
10.4. Amortisationsrechnung - Empfänger	111
10.5. Interpretation	113
A. Aufwand	115
B. Betreuungsprotokolle gemäß §9 bs. 2 PrO	119
C. Abkürzungsverzeichnis	123
D. Abbildungsverzeichnis	125
E. Listings	129
F. Tabellenverzeichnis	131
G. Danksagungen	133
H. Literaturverzeichnis	135

1. Einleitung

1.1. Beschreibung der Rollen im Projekt

1.1.1. Projektmanager (PM)

Nina Nenning besucht die 5. Klasse Elektronik und Technische Informatik an der HTL St. Pölten und hat bereits einiges an praktischer Erfahrung im Bereich Softwareentwicklung in ihren Praktika und ihrer Freizeit gesammelt. Sie übernimmt die Rolle des Projektmanagers.

1.1.2. Projektteammitglied (PTM)

Pia Wenty besucht die 5. Klasse Elektronik und Technische Informatik an der HTL St. Pölten und hatte die Idee zu dieser Diplomarbeit. Sie hat bereits einiges an praktischer Erfahrung in Bereich der Hardwareentwicklung in ihren Praktika und ihrer Freizeit gesammelt. Sie wird als Entwicklerin die Rolle des Projektteammitglieds übernehmen.

1.1.3. Betreuer

Als Betreuer unterstützt Prof. Dipl.-Ing. Dr. Reinhard Steindl von Seiten der HTL St. Pölten. Er unterrichtet Kommunikation und Netzwerk Systeme (KSN). Daher ist er ausgezeichnet für die Rolle als Projektbetreuer geeignet.

1.2. Meilensteine

Zwei Meilensteine geben den Start sowie das Ende des Projekts an. Die restlichen vier Meilensteine sind jeweils der Abschluss einer Phase Tabelle 1.1 zeigt diese.

Projektmanagement	
Festlegen der zu verwendenden Messmethode	17.10.2018
Entwicklung	
Abrufen und anzeigen von Testdaten aus der Datenbank in der Webapplikation	23.11.2018
Fertigstellung der Hardware zur Füllstandsmessung	20.12.2018
Gemessene Füllstandsdaten senden können	27.02.2019
Testphase	
Gesamttests abgeschlossen	20.03.2019
Projektdokumentation	
Abgabe der Diplomarbeit	01.04.2019

Tabelle 1.1.: Meilensteine

1.3. Aufteilung des Arbeitsbereichs

Abbildung 1.1 zeigt das diese Diplomarbeit in zwei Arbeitsbereiche aufgeteilt wurde. Ein Bereich ist die Hardware sowie Software des Senders und der zweite Bereich ist die Software der Webapplikation.

Die Software Entwicklung ist der Aufgabenbereich von Nina Nenning und die Hardware Entwicklung der von Pia Wenty.

Die Übertragung der Daten überlappt sich mit den eingeteilten Arbeitsbereichen. Sowohl Nina Nenning als auch Pia Wenty arbeiteten gemeinsam daran.

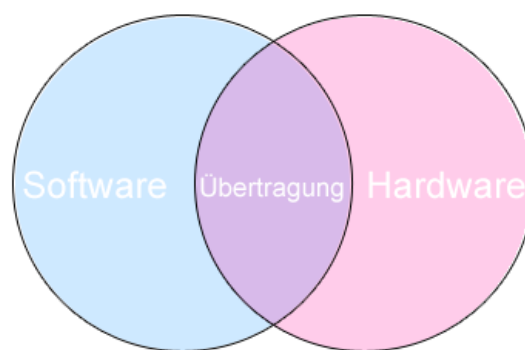


Abbildung 1.1.: Aufteilung der Arbeitsbereiche

1.4. Projektplan

Der Projektplan ist die Zusammenfassung der Meilensteine und der Aufteilung des Arbeitsbereiches auf zeitlicher Ebene.

Mithilfe des Zeitstrahls in der Tabelle 1.3 kann man sich einen Überblick über den Verlauf des Projekts machen.

Bei der Erstellung des Projektplans wurde darauf geachtet, dass genügend Zeitreserven als Puffer vorhanden sind, um nicht in einen zeitlichen Verzug zu geraten.

Des Weiteren sind in der Tabelle 1.3 die Meilensteine beschriftet. Die Zeitspanne des Projekts umfasst 159 Tage.

Bei der Betrachtung des Projektplans muss allerdings berücksichtigt werden, dass die 159 Tage nicht die volle Arbeitszeit, sondern die gesamte Durchlaufzeit darstellen.

Die reine Arbeitszeit sollte pro Kopf in etwa 180 Stunden betragen. Der größte Aufwand fand in der Entwicklungs- und Testphase statt.

Die farbliche Markierung des Aufgabenbereichs von Pia Wenty, des Aufgabenbereichs von Nina Nenning, sowie der Meilensteine wurde im gesamten Dokument beibehalten, um dem Leser die Aufteilung besser zu veranschaulichen.

Die Tabelle 1.2 zeigt die Legende des Projektplans Tabelle 1.3 auf der nachfolgenden Seite.

Part	Farbe
Sender	
Empfänger	
P + N	
Meilensteine	

Tabelle 1.2.: Projektplan Legende

	09/18	10/18	11/18	12/18	01/19	02/19	03/19	04/19
Informationen einholen	20h							
Festlegen der Messmethode		17.10						
Sender Konfiguration	18h	18h						
Schaltungs Entwicklung		18h	18h	18h				
Fertigstellung der Hardware				20.12				
Sender Software Entwicklung				18h	18h			
Gesammelte Daten senden						27.03		
Gehäuse Design					18h	18h	18h	
Empfänger Konfiguration	15h	15h						
Pi Konfiguration		15h	15h					
Datenbank aufsetzen			15h	15h				
Test Daten abrufen können			23.11					
Webapplikation Entwicklung			15h	15h	15h			
Webapplikations Optimierungen					15h	15h	15h	
Final Test/Letzte Korrekturen							20h	
Test Abgeschlossen							20.03	
Diplomabgabe								01.04

Tabelle 1.3.: Projektplan

2. Individuelle Zielsetzung

In diesem Kapitel wird die individuelle Zielsetzung der einzelnen Projektteammitglieder festgelegt.

2.1. Pia Wenty

Pia Wenty sollte folgende Punkte des Senders und Empfängers Entwickeln:

- Akkuüberwachung beim Laden und Entladen - Sender
- Füllstandsmessung des Getränkebehälters - Sender
- Verarbeitung und Aufbereitung der Daten - Sender
- Senden der Daten an ein entsprechendes Gateway (Empfänger) - Sender
- PCB Entwicklung - Sender & Empfänger
- Gehäuse Entwicklung - Sender & Empfänger

Es soll eine kompakte und vielseitig einsetzbare Hardware, in Form eines Untersetzers im Durchmesser von 10cm entwickelt werden. Der Untersetzter soll den Füllstand eines Getränkebehälters von 1ml bis zu 1,5l messen.

Nach dem Messen und Verarbeiten der Daten sollen diese mithilfe eines LoRa-Moduls versendet werden. Ein entsprechendes Gateway empfängt diese Daten. Der Akku muss dementsprechend überwacht werden, um eine Zerstörung des Geräts zu verhindern. Des Weiteren soll ein kompaktes und gut handhabbares Gehäuse entworfen werden.

2.2. Nina Nenning

Nina Nenning sollte folgende Punkte des Empfängers so wie der Webapplikation Entwickeln:

- Einrichten des Empfängers
- Aufsetzen einer Datenbank
- Speichern und Aufbereiten der empfangenen Daten
- Anzeige der Daten in der Webapplikation
- Erstellen von Benutzerkonten

Es soll eine allgemeinverständliche Software, in Form einer Webapplikation entwickelt werden. Spezielles Augenmerk soll auf die Einfachheit des Aufbaus gelegt werden, um dem Benutzer ein müheloses Einsehen seiner Trinkgewohnheiten zu gewähren.

Nach dem Empfangen der Daten mithilfe eines LoRa-Moduls werden diese für den Benutzer aufbereitet und in einem Diagramm angezeigt.

Die Webapplikation soll die täglichen Trinkgewohnheiten des Benutzers darstellen und somit zur Verbesserung der Gesundheit beitragen.

Ebenso soll es für Benutzer möglich sein, sich einfach Registrieren zu können, um die anzuzeigenden Daten trennen und für jeden Benutzer individuell ausgeben zu können.

3. Entwicklungsumgebung

Im folgenden Kapitel wird die Entwicklungsumgebung beschrieben. Dabei handelt es sich um Software sowie Hardware-Komponenten. Die Infrastruktur wurde gründlich durchdacht und ist auf die einfache Bedienbarkeit ausgelegt, sowie zur etwaigen Weiterentwicklung.

3.1. $\text{\LaTeX}$ & Overleaf



Abbildung 3.1.: $\text{\LaTeX}$ Logo [1]



Abbildung 3.2.: Overleaf Logo [2]

$\text{\LaTeX}$ wurde verwendet zum erstellen dieser Dokumentation. Abbildung 3.1 zeigt das Logo von $\text{\LaTeX}$. $\text{\LaTeX}$ bietet unbegrenzte Möglichkeiten zum Dokumentieren. Weiteres ist es im Sektor wissenschaftlicher Arbeiten weit verbreiten. Durch seine umfangreiche Palette an Funktionen und dem modularen Aufbau eignet es sich ideal zur Dokumentation der Diplomarbeit.

Overleaf ist ein Webeditor für $\text{\LaTeX}$ und bietet die Möglichkeit, dass mehrere Personen gleichzeitig am selben Dokument arbeiten. So wurde das Zusammenfügen einzelner Textabschnitte noch einfacher Hand-zu-haben. Abbildung 3.2 zeigt das dazugehörige Logo.

3.2. Altium Designer & Autodesk Inventor



Abbildung 3.4.: Inventor Logo [4]

Abbildung 3.3.: Altium Logo [3]

Altium Designer ist unter anderem ein Schaltungsdesign-Tool. Es kann zum Schaltungs- und PCB-Design verwendet werden. Altium Designer ist weit verbreitet und stellt ein breites Spektrum an Funktionen zur Verfügung. Da während des Unterrichts schon fünf Jahre damit gearbeitet und Erfahrung gesammelt wurde, wurde es auch für die Diplomarbeit verwendet. Abbildung 3.3 zeigt des Logo von Altium Designer.

Autodesk Inventor ist ein 3D-Design-Tool. Es kann zum designen von dreidimensionalen Objekten verwendet werden. Autodesk Inventor ist aufgrund des breiten Funktionsspektrums weit verbreitet. Abbildung 3.4 zeigt das dazugehörige Logo.

3.3. Arduino Studio



Abbildung 3.5.: Arduino Logo [5]

Die Open-Source-Arduino-Software (IDE) läuft unter Windows, Mac OS X und Linux. Die Umgebung ist in Java geschrieben und basiert auf Open-Source-Software. Diese Software kann mit jedem Arduino-Board verwendet werden. Abbildung 3.5 zeigt das dazugehörige Logo.

3.4. Verwendete Technologien

Mit den verwendeten Technologien Raspberry Pi und LoRa wurde diese Diplomarbeit realisiert.



Abbildung 3.6.: Raspberry Pi Logo [6]

Raspberry Pi Raspberry Pi ist ein Ein-Platinen-Computer, er enthält ein Ein-Chip-System von Broadcom, mit einem ARM-Mikroprozessor. Der Raspberry Pi ist für seine hervorragende Leistung erstaunlich Preiswert. Er wird als Gateway, Server und zur Datenverarbeitung eingesetzt. Abbildung 3.6 zeigt das dazugehörige Logo.



Abbildung 3.7.: LoRa Logo [7]

LoRa LoRaWAN ist ein Low-Power-Wireless-Netzwerkprotokoll. Die LoRaWAN-Spezifikation wird von der LoRa Alliance festgelegt, ist frei verfügbar und nutzt die Chirp-Spread-Spectrum-Modulationstechnik. LoRaWAN ist auf die Energieeffizienz der IoT-Geräte ausgerichtet und erreicht dabei hohe Reichweiten (>10 km). Abbildung 3.7 zeigt das dazugehörige Logo.

4. Übertragungstechnik

In diesem Kapitel ist die allgemeine Struktur der Daten-Übertragungstechnik beschrieben und dokumentiert. Ein Teil dieses Kapitels enthält Teile aus der Projektarbeit "Datenübertragung mit LoRa" der HTL St.Pölten. Zusätzlich dazu wurden Informationen von folgenden Webseiten verwendet: Was ist ASE Encryption [8] und Smartmaker [9].

4.1. Low Power Wide Area Network

Low Power Wide Area Network kurz LPWAN beschreibt eine Klasse von Netzwerkprotokollen welche darauf ausgelegt sind Niederenergiegeräte zu Verbinden. Darunter fallen zum Beispiel batteriebetriebene Sensoren. Das Ziel der Protokolle ist es, den Energieverbrauch der Endgerät bei hoher Reichweite so gering wie möglich zu halten um die Batteriebensdauer zu erhöhen.

Die Netzwerkarchitektur besteht aus Endgeräten (Nodes) und Basisstationen (Gateways), welche die Daten zwischen den Nodes und Netzwerkservern weiterleiten.

Die Verbindungen zwischen Nodes und Gateways erfolgen über lizenzfreie Frequenzen des Radiospektrums (ISM-Band [10]) oder Mobilfunkfrequenzen. Zwischen Gateways und Netzwerkserver besteht die Verbindung in Form einer IP-Verbindung.

4.2. LoRaWAN

In dieser Arbeit wurde LoRaWAN als LPWAN eingesetzt. LoRaWAN Spezifikationen wurde von der LoRa Alliance festgelegt und steht frei zur Verfügung. Das dazugehörige Übertragungsverfahren LoRa basiert auf dem Chirp Spread Spectrum Modulationstechnik und wurde von der Firma Semtech entwickelt. Sie ist asymmetrisch auf Energieeffizienz der Nodes ausgerichtet und erreicht hierbei hohe Reichweiten über 10 km für die Uplink-Kommunikation, also das Senden von der Node an ein entsprechende Gateway.

Im Grunde ist LoRa die Physikalische Schicht (PHY): Der Schaltkreis der die Modulation/Demodulation vornimmt. LoRaWAN ist die MAC-Schicht: Die Software, die die Vernetzung ermöglicht.

4.2.1. Unterschied LoRa und LoRaWAN

LoRa und LoRaWAN gehen Hand in Hand, doch ein paar feine Unterschiede gibt es doch.

LoRa ermöglicht einem erst die weitreichende Kommunikationsverbindung. LoRa verwendet eine drahtlose Übertragung und ermöglicht eine Kommunikation mit sehr geringen Stromverbrauch. Die LoRa-Modulation ist vergleichbar mit der ersten Schicht des OSI-Modells (Open Systems Interconnection Model) dar.

LoRaWAN definiert das Kommunikationsprotokoll und die Systemarchitektur für das Netzwerk. Es ist vergleichbar mit den Schichten eins bis fünf des OSI-Modells. Es gibt insgesamt drei verschiedene LoRaWAN-Klassen. Das Protokoll und die Netzwerkarchitektur haben den größten Einfluss bei der Bestimmung der Batterielebensdauer eines Knotens, der Netzwerkkapazität, der Servicequalität und der Sicherheit.

Es besitzt eine sternförmige Netzwerktopologie (die Teilnehmer sind mittels Punkt-zu-Punkt Verbindungen an einzelne Gateways verbunden).

Europa	
Frequency Band	434/868MHz
Channels	1-10
Channel BW Up	125/250/500kHz
Channel BW Dn	125kHz
TX Power Up	+14dBm
TX Power Dn	+14dBm
Spreading Factor	7-12
Datarate	250bps-50kbps
Link Budget Up	155dB
Link Budget Dn	155dB

Tabelle 4.1.: LoRaWAN Spezifikationen

4.2.2. Netzwerktopologie

Die netzwerktopologie des LoRaWAN's ist Sternförmig aufgebaut. Das bedeutet das die einzelnen Nodes nicht fixen mit einem Gateway verbunden sondern die gesendeten Daten werden von mehreren Gateways empfangen. Die Gateways senden die Datenpakete dann weiter an einen Netzwerkservers mit IP-Verbindung. Dieser filtert redundante Pakete heraus. Ebenso verfügt er über Schnittstellen, um an IoT-Plattformen und Applikationen angebunden zu werden.

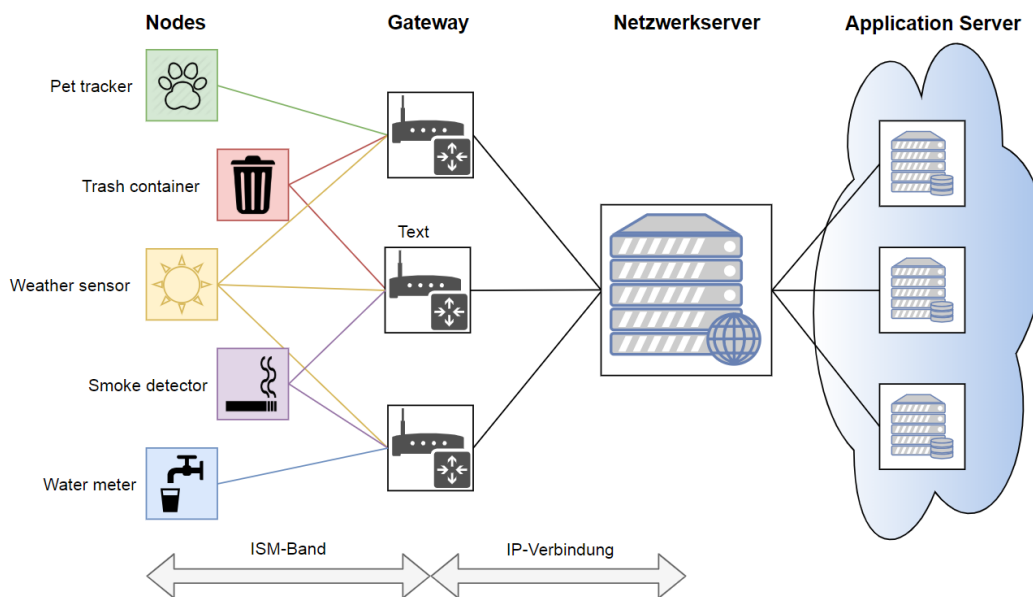


Abbildung 4.1.: LoRa Netzwerkachitektur

4.2.3. Sicherheit

LoRaWAN nutzt zwei Sicherheitsstufen: eine für das Netzwerk und eine für die Applikation.

Die Netzwerksicherheit sichert die Authentizität des Knotens im Netzwerk, während die Applikationsstufe den Zugriff des Netzwerkoperators auf die Daten des Endbenutzers verweigert. Dazu wird Advanced Encryption Standard kurz AES verwendet. AES ist ein symmetrisches Verschlüsselungsverfahren, das bedeutet der Schlüssel zum Ver- und Entschlüsseln ist identisch. Der Algorithmus darf auch ohne Lizenzgebühren in Hard- und Software eingebaut werden.

4.3. Ra02-Modul

Das Ra02-Modul ist ein Modul welches die LoRa-Modulation erzeugen und verwenden kann.

4.3.1. Spezifikationen

Die Spezifikationen des Ra02-Moduls V1.4 sind in der Tabelle 4.2 in der mittleren Spalte aufgelistet. Die daraus gewählten Werte welche für Österreich laut ÖVSV [10] genehmigt sind, oder nicht verändert werden können, sind in der rechten Spalte aufgelistet.

Spezifikationen		Gewählte Werte
Versorgung [VCC]	2.5-3.7V	3.3V
Bandbreite [BW]	125kHz - 500kHz	125kHz
Spreading Factor [SF]	7-12	7
Frequenz	410-525MHz	434MHz
Error Correction Code [CR]	1 - 4	1
Output Power	max. 18dBm	13dBm
Payload length	0 - 64 Bytes	11 Bytes
Preamble length	12 Symbols	12 Symbols

Tabelle 4.2.: Ra02-Modul Spezifikationen und Gewählte Werte

4.3.2. Pinbelegung & Bemaßung

Abbildung 4.2 und 4.3 zeigen die Pinbelegung sowie die Bemaßung des Ra02-Moduls.

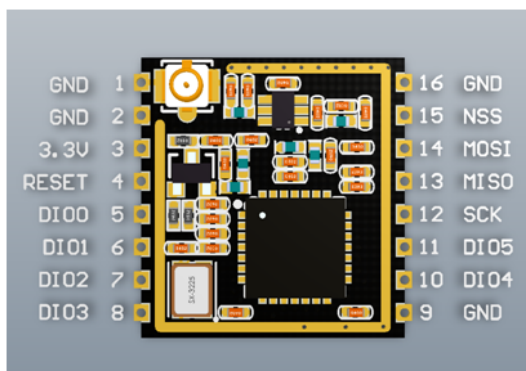


Abbildung 4.2.: Pinbelegung

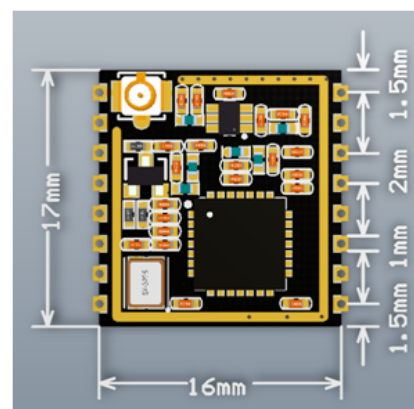


Abbildung 4.3.: Bemaßung

4.4. LoRa-Modulation

Die LoRa-Modulation bedient sich der Technologie des Chirp Spread spectrum. Hierbei wird ein Chirpsignal, welches die Frequenz in der gegebenen Bandbreite kontinuierlich verändert, erzeugt. Das bedeutet es werden für "1" und "0" jeweils ein Chirp definiert. Hier ist ein Up-Chirp ein "1" und ein Down-Chirp ein "0". Abbildung 4.4 zeigt Up- und Down-Chirp welche das Ra02-Modul produziert und verwendet. Der Vorteil dieser Methode ist, dass Timing- und Frequenz-Offsets zwischen Sender und Empfänger gleichwertig sind, dies reduziert die Komplexität des Empfängerkonzepts erheblich.

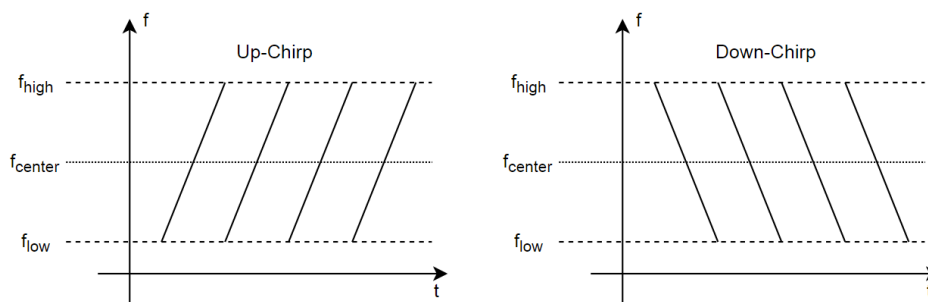


Abbildung 4.4.: Up- und Down-Chirp's

Abbildung 4.5 zeigt das bevor die eigentliche Nachricht übertragen wird zuerst eine Kombination aus Up- und Down-Chirps geschickt wird um den Empfänger darauf vorzubereiten das jetzt eine Nachricht kommt. Dies wird ebenfalls gleich dazu genutzt um sich mit dem Empfänger auf zu synchronisieren. Bei LoRa sind das 10 Up-chirps gefolgt von 2 Down-Chirps und dann die Nachricht. Time on Air beschreibt die Zeit die der Sender benötigt um die gesamte Nachricht (vorausgesetzt sie passt in die maximale Payload von 59 Symbole) zu senden. Wobei Preamble Time die Zeit für den Teil der Synchronisation beschreibt und Payload Time die Zeit für die eigentliche Nachricht.

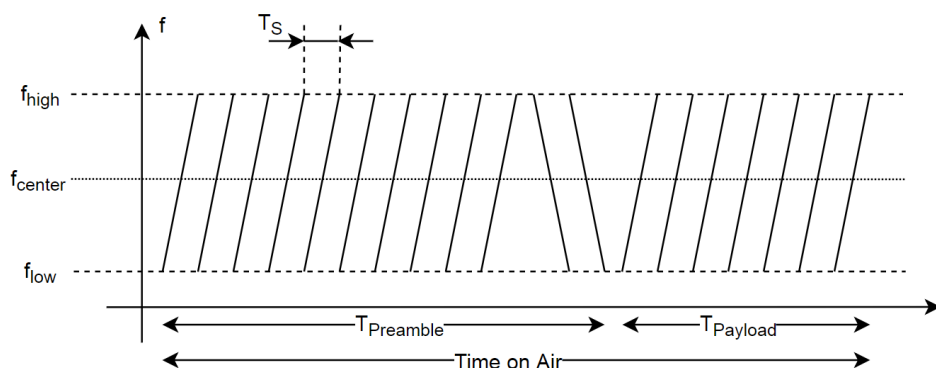


Abbildung 4.5.: LoRa-Paket Skizze

4.4.1. LoRa-Nachricht

Eine LoRa-Nachricht besteht aus mehreren Teilen, einer Preamble welche für die Synchronisation zuständig ist, einen Header, der Nachricht und dem Cyclic redundancy check kurz CRC Teil. Der Header teilt sich nochmal in drei Teile, der Angabe für die Länge der Nachricht, CR und den CRC Vorbote. Passt die zusendende Nachricht nicht in die maximale Payload splittet der LoRa-Sender die Nachricht und sendet diese zweimal. Abbildung 4.6 zeigt die Zusammensetzung einer LoRa-Nachricht.

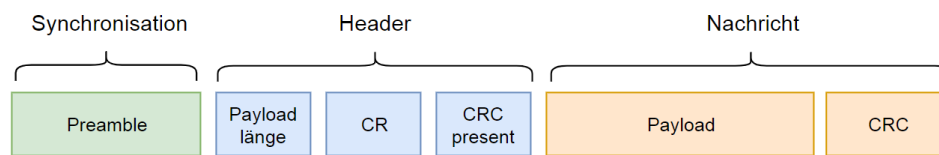


Abbildung 4.6.: LoRa-Nachricht

Abbildung 4.7 zeigt einen Sogenannten LoRaWAN-Protokollstack. Er besteht aus einer Application Ebene, einer MAC-Schicht und einer Physikalischen Schicht. Die MAC-Schicht erstellt aus den MAC-Nutzdaten welche sie von der Anwendungsebene bekommt, den MAC-Frame. Die MAC-Nutzdaten enthalten einen Frame-Header (mit den Quell- und Zieladressen sowie einem Frame Counter), einen Frame-Port und die Frame-Nutzdaten (mit den Anwendungsdaten). Mithilfe des Frame-Ports wird festgestellt, ob der Frame ausschließlich MAC-Befehle oder anwendungsspezifische Daten enthält. Schließlich verwendet die PHY-Ebene den MAC-Frame als PHY-Nutzdaten und erstellt so die LoRa-Nachricht, nachdem Pramble, PHY-Header und CRC eingefügt wurden.

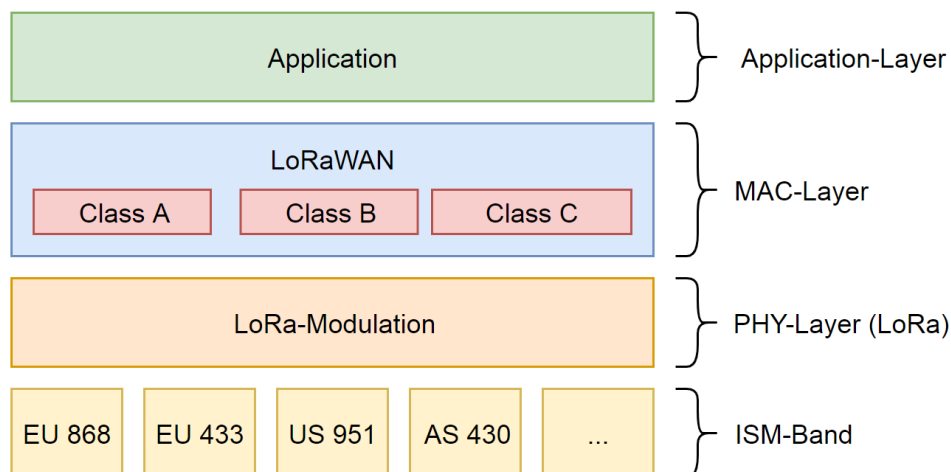


Abbildung 4.7.: LoRa-Protokollstack

4.4.2. Berechnungen

Die gewünschten Parameter von Datenbitrate $[R_{\text{Bit}}]$, Symbolrate $[R_{\text{Symbol}}]$, Chiprate $[R_{\text{Chip}}]$, Chip-Time $[T_{\text{S}}]$, Preamble-Time, Payload-Time sowie Time on Air sind bei der LoRa-Modulation wie folgt zu berechnen, wobei Spreading Factor mit SF, Coding Rate mit CR und Bandwidth mit BW abgekürzt wird:

$$R_{\text{Bit}} = SF * \frac{\left\lceil \frac{4}{4+CR} \right\rceil}{\left\lceil \frac{2^{\text{SF}}}{BW} \right\rceil} [\text{bits/s}] \quad (4.1)$$

$$R_{\text{Symbol}} = \frac{BW}{2^{\text{SF}}} [\text{symbols/s}] \quad (4.2)$$

$$R_{\text{Chip}} = R_{\text{Symbol}} * 2^{\text{SF}} [\text{chips/s}] \quad (4.3)$$

$R_{\text{Bit}} [\text{bits/s}]$	$R_{\text{Symbols}} [\text{symbols/s}]$	$R_{\text{Chip}} [\text{chip/s}]$
5,46k	17,8k	125k

Tabelle 4.3.: Rate Parameter

Man unterscheide den sogenannten Chirp welcher ein Symbol darstellt und dadurch die Dauer eines Chirps wie folgt berechnet werden kann (Formel 4.4) und eine Chiprate welche den Chirp in 2^{SF} kleine Teilstücke (Chips) unterteilt. Dadurch treten die für die LoRa-Modulation typischen zyklisch verschobenen Up-Chirps auf welche in Kapitel 4.4.5 betrachtet werden können.

$$T_{\text{S}} = \frac{1}{R_{\text{S}}} \quad (4.4)$$

Daraus wiederum ergibt sich die Time on Air"das ist jene Zeit, in der der Sender einen Teil oder eine ganze Nachricht sendet. Sie setzt sich aus der Preamble-Zeit und der Payload-Zeit zusammen. Tabelle 4.4 beinhaltet die ausgerechneten Parameter

$$T_{\text{OA}} = T_{\text{preamble}} + T_{\text{payload}} \quad (4.5)$$

$$T_{\text{preamble}} = (n_{\text{preamble}} + 4, 25) * T_{\text{S}} \quad (4.6)$$

$T_{\text{S}} [\text{s}]$	$T_{\text{Preamble}} [\text{s}]$	$R_{\text{Payload}} [\text{s}]$	$T_{\text{OA}} [\text{s}]$
1,02m	12,54m	28,68	41,22m

Tabelle 4.4.: Time Parameter

4.4.3. Messungen

Um zu überprüfen ob die Theorie mit der Praxis übereinstimmt wurden Frequenz-, Zeit- und Chirp-Messungen durchgeführt.

Frequenzmessung

Um sicherzugehen, dass das Ra02-Modul auf der gewählten Frequenz sendet wurde eine Testmessung durchgeführt. Dazu wurde ein kurzes Testprogramm geschrieben mit welchem der Sender dauerhaft auf 434MHz sendet.

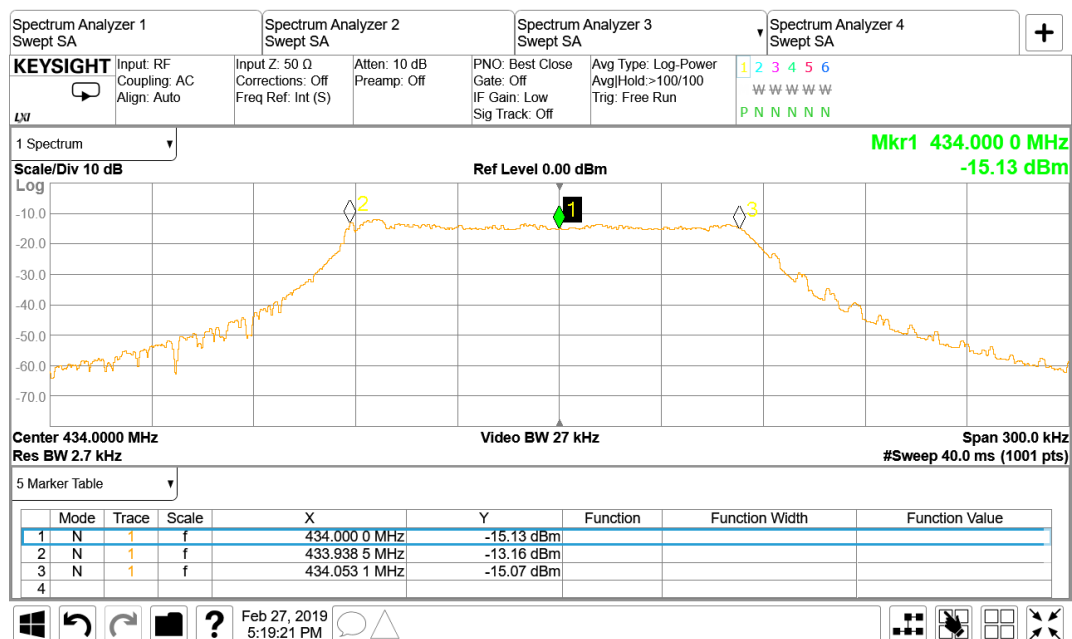


Abbildung 4.8.: Ra02-Modul Sende Frequenz

In der Abbildung 4.8 sieht man, dass das Ra02-Modul genau auf der eingestellten Frequenz von 434MHz sendet. Sowie mit der eingestellten Bandbreite von 125kHz. Für diese Messung wurden am Spektrumsanalysator zusätzlich noch folgende Einstellungen vorgenommen:

Span	Center Frequency	Bandwidth	Tracer
300kHz	434MHz	2.7kHz	Mitteln

Tabelle 4.5.: Spektrumsanalysator Einstellungen

4.4.4. Zeitmessung

Für die Messung, wie lange eine Übertragung von exakt 10 Byte benötigt, wurde ein neues Testprogramm geschrieben. Mit dem Programm kann der Sender eine Nachricht von 10 Byte überträgt und macht dann eine Pause von 20ms. Abbildung 4.9 zeigt die dazugehörige Zeitmessung. Damit die x-Achse des Spektrum Analysators auf Zeit gestellt ist und nicht mehr auf Frequenz muss ein Span von 0Hz und eine Sweeptime eingestellt werden. Damit man sehen kann wie lange eine Nachricht dauert wurde eine Sweeptime von 150ms eingestellt.



Abbildung 4.9.: Sendezeit Messung

Aus Abbildung 4.9 kann man herauslesen, dass das Ra02 Modul in etwa 41ms benötigt, um eine Mitteilung von 10 Byte zu senden. Dieser Wert passt mit der gerechneten Time on Air aus Kapitel 4.4.2 überein. Die genaue Chirp Aufteilung des Pakets wird in Kapitel 4.4.5 gemessen.

4.4.5. Chirp Messung

Um zu überprüfen ob die gerechneten Werte von Preamble- und Payload-Zeit aus Kapitel 4.4.2 stimmen, wurde eine Chirp-Messung durchgeführt. Die Aufnahme der Abtastwerte erfolgt über das Oszilloskope von Rohde und Schwarz. Diese Abtastwerte wurden in Form einer .csv Datei gespeichert. Mit Hilfe der Signal-Analyser-App in Matlab wurde ein Spektrogramm erstellt.

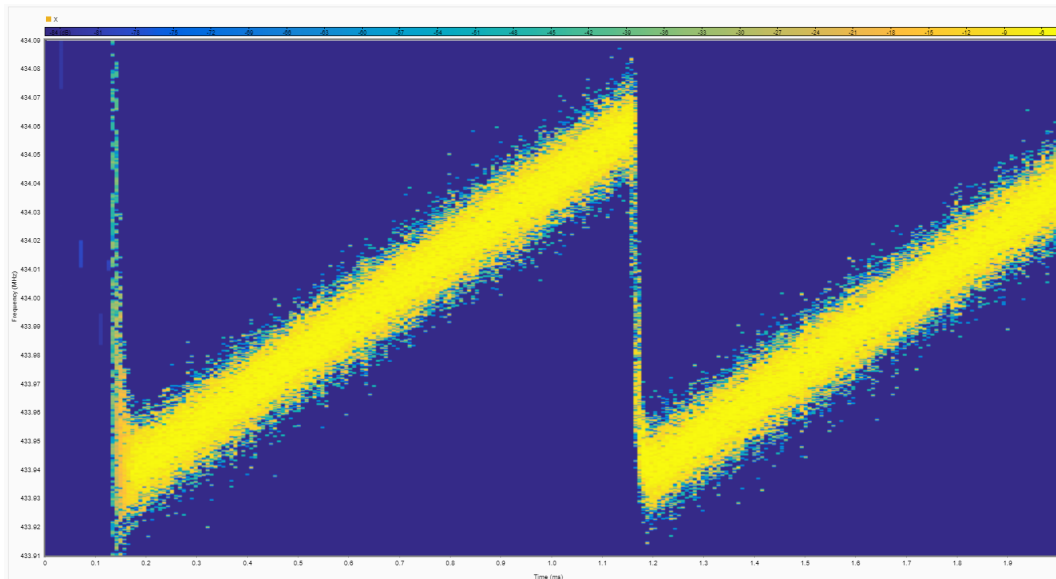


Abbildung 4.10.: Spektrogramm

Abbildung 4.10 zeigt das erstellte Spektrogramm. Das Spektrogramm zeigt 1 und 7/8 Up-Chirps. Man erkennt ohne Probleme das ein Chirp wie berechnet 1ms lang ist und seine Frequenz kontinuierlich in einer Bandbreite von 125kHz ändert. Diese zwei Up-Chirps stammen aus der Preamble eines LoRa-Paketes. Abbildung 4.11 zeigt ein Spektrogramm eines LoRa-Paketes mit der vollständigen Preamble und einen Teil der Payload.

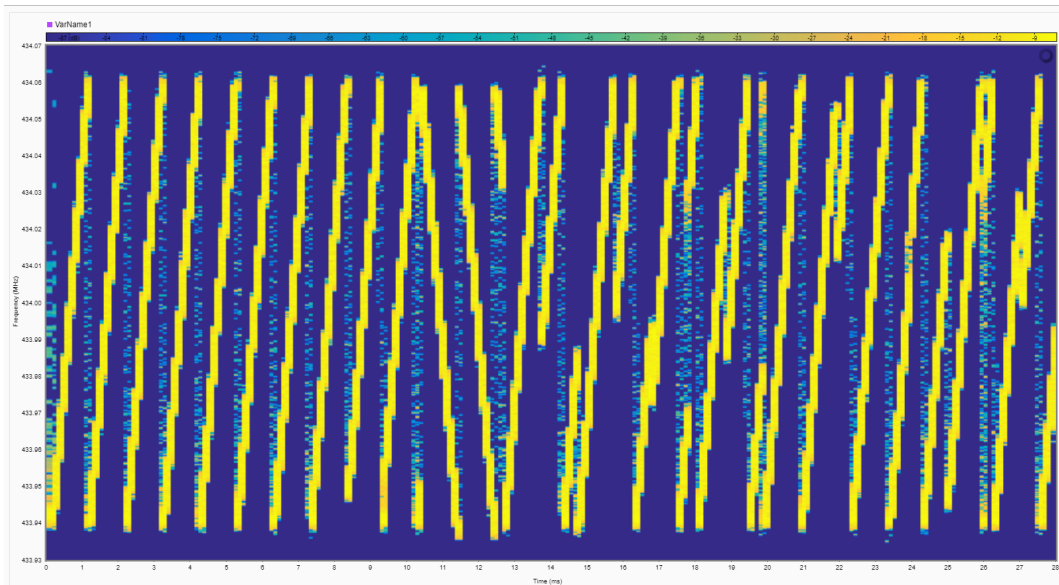


Abbildung 4.11.: Nachrichten Spektrogram

Abbildung 4.11 zeigt das Spektrogramm eines LoRa-Paket. Es lassen sich die 10 Up-Chirps gefolgt von 2 Down-Chirps für die Preamble erkennen sowohl als auch einen Teil einer Payload. Das gewünschte Datensignal wird mit einer höheren Datenrate (2^{SF}) gechipt und auf das Chirpsignal moduliert. Durch die Modulation entstehen die für LoRa typischen zyklisch verschobenen Up-Chirps. Es scheint als würde die Frequenz "herum springen", aber es gibt keine Sprünge in der Phase, die zu unerwünschten Oberwellen im Form von Störungen im Frequenzband führen würden. Bis auf ein paar Chirps, die den Anfang einer Nachricht signalisieren, sind alle Chirps gleich lang.

5. Sender

In diesem Kapitel ist die allgemeine Struktur des Senders beschrieben und dokumentiert. Um das Projekt zu realisieren wurden die Technologien LoRaWAN und Arduino verwendet. Im gesamten Kapitel werden Daten und Informationen aus den Datenblättern des MCP7833 [11], des HX711 [12], des RTC DS3231 [13], des R5421N [14], des FDS6890A [15], des S8560 [16], des 24AA256 EEPROM (Electrically Erasable Programmable Read-Only Memory) [17] und des ATmega328P auf dem IO-Board mit dem Namen Arduino Nano [18] verwendet. Dabei wird nicht jedes einzelne Mal darauf verwiesen sondern nur einmalig hier.

5.1. Funktionsbeschreibung

Als Methode für das Erfassen der Füllstandsdaten wurde das Wiegen gewählt. Ein DMS-Modul reagiert auf die Füllstandsänderung mit einer Spannungsänderung an deren Ausgangspins. Der HX711-ADC (Analog Digital Converter) greift diesen Analogwert auf und konvertiert ihn für den Arduino Nano in einen Digitalwert. Aus dem EEPROM (Electrically Erasable Programmable Read-Only Memory) liest das IO-Board mit dem Namen Arduino Nano die Seriennummer des Geräts. Von dem DS1307-RTC-IC (Real Time Clock) bekommt der Arduino Nano die aktuelle Uhrzeit. All diese Datenschnipsel werden zusammen zu einem Datenblock konvertiert und mit dem LoRa-Modul versendet. In den folgenden Kapiteln werden die genauen Aufgaben der einzelnen Bausteine genauer beschrieben. Auch wird die Realisierung der Schaltungen, des PCB-Designs, der Software sowie des Gehäuse-Designs geschildert.

5.1.1. Arduino Nano

Das IO-Board mit dem Namen Arduino Nano besitzt einen ATmega328P Mikrocontroller (μC) und einen FT232RL USB-Treiber und ein USB-mini-B Stecker. Somit kann er über einen USB-Port programmiert werden. Ebenfalls befindet sich ein $\mu A78M$ Spannungsregulator zur Stabilisierung der Eingangsspannung auf dem IO-Board. Auch ist der benötigte Quarz für den ATmega328P schon vorhanden.

5.1.2. Wägezelle

Welche Wägezelle eignet sich für welche Anwendung?

Entsprechend ihrer Gesamtleistungen werden Wägezellen in verschiedene Genauigkeitsklassen bzw. Genauigkeitsstufen unterteilt. Eine Genauigkeitsstufe umfasst mehrere Parameter, wie zum Beispiel: Linearität, Hysterese, Temperatureffekte, usw.

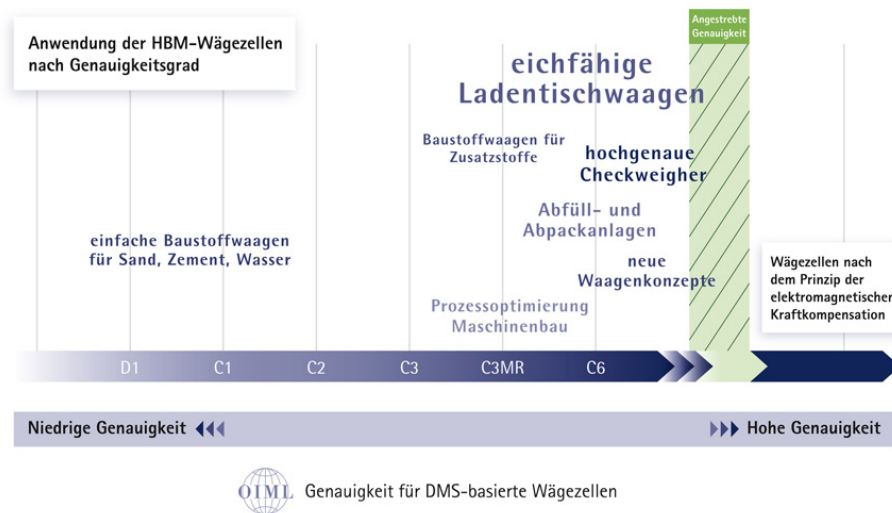


Abbildung 5.1.: Wägezelleklassen [19]

In Abbildung 5.1 sind die unterschiedlichen Genauigkeitsklassen aufgezeichnet: die Abbildung beginnt mit der niedrigsten Klasse (D1) und endet bei der Höchsten (C6).

Klasse D1 bis C2 sind ausreichend für einfache Baustoffwaagen zum Wiegen von Sand, Zement, oder Wasser.

Will man Additive zu den Baustoffen hinzufügen benötigt man schon eine Wägezelle mit der Klasse C3 oder höher.

Jede Wägezelle, welche eine Genauigkeitsstufe von C4 oder höher hat, kann Milli- bis sogar Mikrogramm genau messen. Damit ist eine solche Wägezelle passend für dieses Projekt.

Die verwendete Wägezelle entspricht einer Genauigkeitsklasse von C5.

5.1.3. DMS

DMS ist die Abkürzung für Dehnmessstreifen. DMS sind Messeinrichtungen zur Erfassung von dehnenden und stauchenden Verformungen. Sie ändern bei Verformungen ihren elektrischen Widerstand und werden als Dehnungssensoren eingesetzt. Wird ein DMS gedehnt, nimmt der Widerstand zu, wird er gestaucht nimmt der Widerstand ab. (vgl. Wikipedia „Dehnmessstreifen“ [20]).

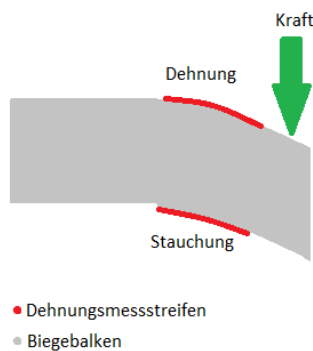


Abbildung 5.2.: Wägezelle

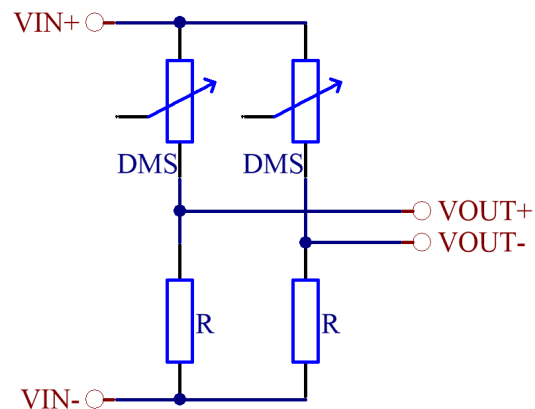


Abbildung 5.3.: LoadCell

In Abbildung 5.2 ist das grundsätzliche Funktionsprinzip einer Wägezelle mit DMS-Sensoren skizziert. Sobald eine Kraft (Grüner Pfeil) auf die Wägezelle trifft, werden die DMS gedehnt (roter Streifen auf der Oberseite) oder gestaucht (rote Streifen an der Unterseite). Dadurch verändert sich der elektrische Widerstand der DMS und bei der daneben skizzierten Schaltung (Abbildung 5.3) ändert sich der Spannungsteiler. Somit ändert sich auch die Ausgangsspannung, welche der HX711-ADC aufgreift und in einen digitalen Wert umwandelt. So wird festgestellt, welches Gewicht sich auf der Wägezelle befindet.

Im unbelastetem Zustand hat ein DMS zwischen 250Ω und $1k\Omega$. Der Widerstand des verwendeten DMS liegt ohne Belastung bei $1k\Omega$. Mit folgender Formel kann die Widerstandsänderung bei dehnung oder stauchung berechnet werden:

$$\Delta R = k * \frac{\Delta l}{l} = k * \varepsilon \quad (5.1)$$

Wobei k den spezifischen Werkstoff Faktor beschreibt und ε die relative Längsdehnung.

5.1.4. Real Time Clock

Zusätzlich zu den Füllstandsdaten wird auch die Uhrzeit und das Datum mitgesendet, damit das in Kapitel 7.1 erzeugte Diagramm richtig dargestellt werden kann. Die Wahl der Real Time Clock fiel auf die DS1307.

DS1307

Der IC ist ein Extremely Accurate I²C-Integrated RTC/TCXO/Crystal. Die meisten RTC's verwenden einen externen 32kHz Quarz um die Zeit mit geringer Stromaufnahme ermitteln zu können. Allerdings haben diese Quarze einen leichten Drift, besonders wenn sich die Temperatur ändert. Bei diesem IC befindet sich der Quarzkristall im Gehäuse. Direkt neben dem integrierten Quarz befindet sich ein Temperatursensor. Dieser Sensor gleicht die Frequenzabweichung aus, indem er Ticks hinzufügt oder entfernt, sodass die Zeitmessung planmäßig bleibt. Die Kommunikation mit dem IC funktioniert über einen I²C-Bus. Abbildung 5.4 zeigt das Block Diagramm des DS1307.

SQW/OUT kann über einen Internen Buffer eine Tacktfrequenz von 1Hz, 4kHz, 8kHz oder 32kHz ausgeben.

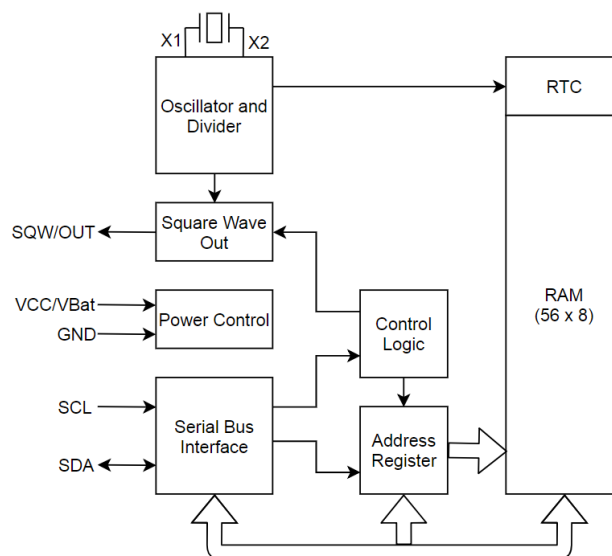


Abbildung 5.4.: Block Diagramm DS1307

LIR2032

Damit auch bei Ausfall des Hauptakkus die Uhrzeit korrekt bleibt, wurde als Backup-Batterie die wiederaufladbare Knopfzelle LIR2032 gewählt.

5.2. Akku

In diesem Kapitel wird die Struktur des Akkus, der Akkuüberwachung, des Akkuladeverfahren und der Akku-Schutzschaltung beschrieben dabei werden Information von den Webseiten IT-Wissen [21] und Adafruit [22], sowie von den Datenblättern des R5421N [14] und des FDS6890A [15] bezogen.

5.2.1. Lithium Polymer

Ein Lithium-Ionen-Polymer Akkumulator wurde ausgewählt, da dieser eine wiederaufladbarer Spezialform des Lithium-Ionen-Akkumulators darstellt. Das spezielle an einem LiPo ist, dass normalerweise flüssige Elektrolyt in fester bis gel-artiger Form auf Polymerbasis verwendet wird. Somit wird der Aufbau von flachen Zellen ermöglicht, welche platzsparender sind.

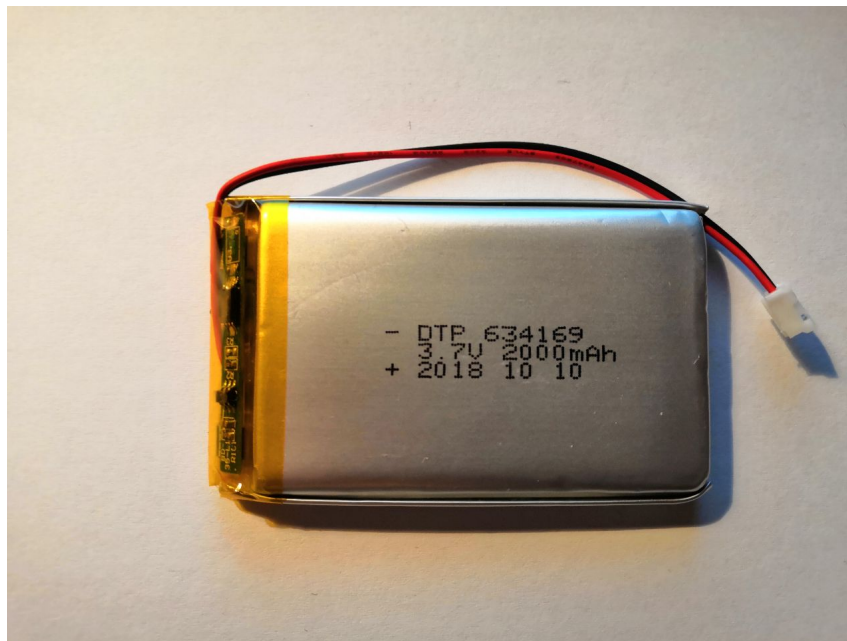


Abbildung 5.5.: Verwendeter Akku

Abbildung 5.5 zeigt den verwendetet Akku mit der Kennung: DTP634169. Der ausgewählte Akku besitzt eine Nennspannung von 3.7V und eine Kapazität von 2000mAh. Die Nummer 10.10.2018 ist das Herstellungsdatum. Auch besitzt er eine integrierte Schutzschaltung welche auf der nächsten Seite genauer beschrieben wird.

5.2.2. Schutzschaltung

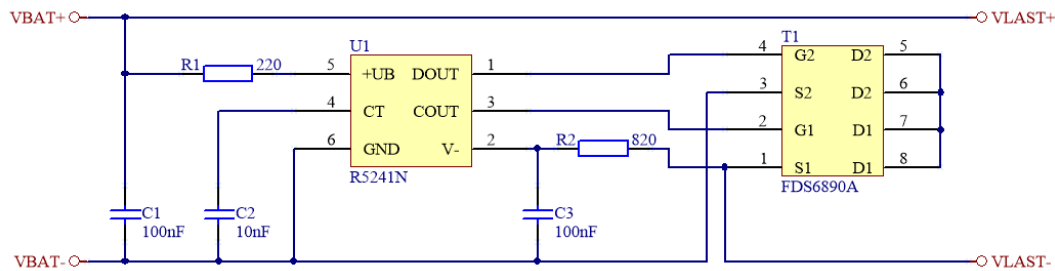


Abbildung 5.6.: Schutzschaltung

Abbildung 5.6 zeigt die Schutzschaltung des LiPo's. Der R5241N als Überwachungs-IC bildet das Herz der Schutzschaltung. Der Dual N-Channel MOSFET IC FDS6890A ist dafür da den Akku gegebenenfalls von der Last zu trennen. Der Widerstand R1 und der Kondensator C1 stabilisieren die Betriebsspannung und verhindern das der R5241N verfrüht reagiert.

Zur Überwachung der Ladeschlussspannung überwacht der R5241N die Zellspannung und kann über einen Levelshifter den nachgeschalteten MOSFET gegebenenfalls sperren.

Zur Überwachung der Tiefentladung kann der IC gegebenenfalls den Ausgang des Akku's sperren und gibt ihn erst wieder frei wenn eine Spannung zum Laden angelegt wird.

Zur Überwachung eines Kurzschlusses oder Überstroms schalten die MOSFET's aufgrund des internen $R_{DS(on)}$ nur durch, wenn eine Spannung von über 0.9V anliegt. Innerhalb von $50\mu s$ können die MOSFET's gesperrt werden und so eine Zerstörung des Akku's verhindern werden.

Bei Erkennung von Überlastungen kann die Last innerhalb von 17ms vom Akku getrennt werden. Wird eine neuer Ladezyklus gestartet wird die elektronische Sicherung zurückgesetzt und der Akku kann normal weiter verwendet werden.

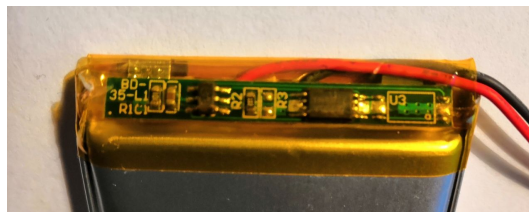


Abbildung 5.7.: Integrierte Schutzschaltung

Abbildung 5.7 zeigt die integrierte Schutzschaltung.

5.2.3. Laden

Das Laden funktioniert in zwei Schritten nach dem IU-Ladeverfahren oder auch CCCV (Constant Current Constant Voltage).

In der ersten Phase wird mit einem konstanten Strom geladen, dadurch steigt die Spannung am Akku.

In der zweiten Phase wird von konstant Strom auf konstante Spannung umgestellt. Dabei sinkt mit zunehmendem Ladestand des Akkus der Ladestrom selbstständig.

Beendet ist das ganze, wenn ein gewisser Mindestladestrom unterschritten wurde. Diesen Prozess überwacht meistens ein speziell dafür hergestellter IC, in diesem Fall der MCP73833.

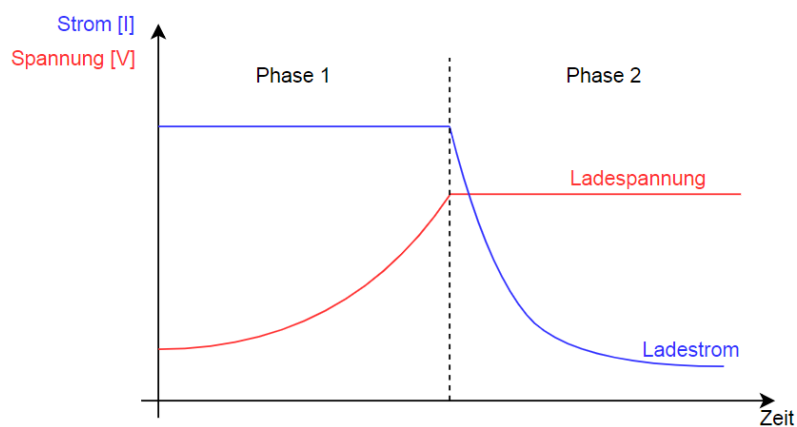


Abbildung 5.8.: CCCV Skizze

Abbildung 5.8 zeigt eine Skizze des CCCV-Ladeverfahrens. Die Skizze zeigt den Anstieg der Spannung in der ersten Phase, als auch den Abfall des Stromes in der zweiten Phase, in Abhängigkeit zur Zeit.

5.2.4. Lebensdauer

Die Lebensdauer eines LiPo-Akkus ist in Zyklenfestigkeit und kalendarische Alterung unterteilt. Allgemein gilt ein Akku als verbraucht, wenn er weniger als 80% seiner Nennkapazität besitzt.

Unter Zyklenfestigkeit versteht man, wie oft man einen Akku Entladen und wieder Laden kann. Die Zyklenfestigkeit hängt von der Tiefe der Entladung ab. Flach gezykelte LiPo's halten länger als jene, die immer an ihrer Belastungsgrenze betrieben werden.

Mit kalendarischer Alterung ist die Lebensdauer in Monaten bzw. Jahren gemeint. Sie schreitet auch voran ohne das der Akku verwendet wird.

5.3. Akku Management

In diesem Kapitel wird die Struktur des Akkumanagements beschrieben dabei werden Information von den Webseiten IT-Wissen [21] und Adafruit [22], sowie des Datenblattes des MCP73833 [11] bezogen.

5.3.1. Beschreibung

Die Entscheidung fiel auf einen Lithium-Ionen und Lithium-Polymer-Lade IC - den MCP73833.

Als Anschluss wurde ein induktives Ladeverfahren gewählt, so gibt es keine offenen Stecker am Gehäuse, in welchen sich Wasser sammeln könnte. Zusammen mit der passenden Ladestation kann der Akku geladen werden. Der IC verfügt über drei Anschlüsse für Anzeige LEDs - eine blaue, die leuchtet, wenn der IC arbeitet, eine orange die signalisiert, dass der Akku geladen wird und eine grüne, die zu leuchten beginnt, wenn der Akku fertig geladen ist.

5.3.2. MCP73833

Der IC ist ein Stand-Alone Linear Li-Ion / Li-Polymer-Charger Management-Controller (vgl. Microchip Datastheet MCP73833/4 [11]). Der IC unterstützt einen 10K NTC-Thermistor, welcher als Temperaturwächter bei hohen Laderaten verwendet werden kann. Der IC wurde in den Gehäuse-Typ MSOP-10 eingesetzt (SMD-Typ).

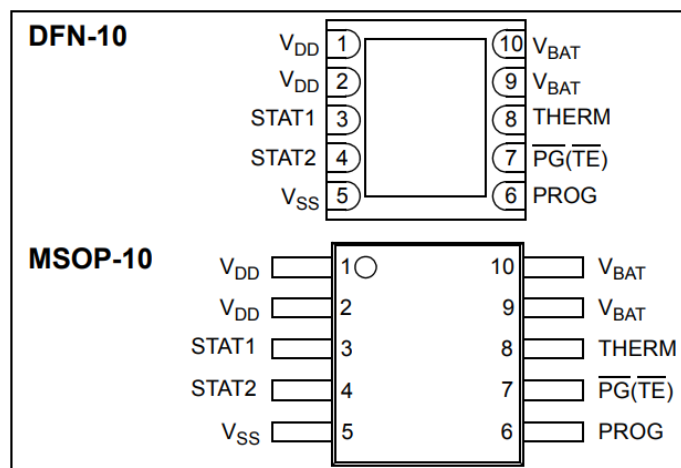


Abbildung 5.9.: Packing Typs

5.3.3. Laden

Das Laden erfolgt in drei Stufen: zuerst eine Vorkonditionierungsladung, dann eine Schnellladung mit konstantem Strom und schließlich eine Erhaltungsladung mit konstanter Spannung, um denn Akku aufzuladen. Der Schnellladestrom beträgt standardmäßig 500mA, ist aber durch das einlöten eines anderen Widerstands an Pin 6 von 100mA bis 1000mA einstellbar.

Widerstand	Lade Strom
10k Ω	100mA
5k Ω	200mA
2k Ω	500mA
1k Ω	1000mA

Tabelle 5.1.: Ladestrom Einstellungen

5.3.4. Entladen

Das Entladen des Akkus überwacht der ATmega328P.

Die Akkuspannung beträgt 4.2V bei einem voll geladenen Akku. Ist der Akku vollständig entladen, fällt die Akkuspannung auf 3.7V herab. Sobald die Akkuspannung unter 4.0V fällt, wird eine rote LED aktiviert, die alle fünf Sekunden blinkt. Fällt die Akkuspannung weiter, also bis auf einen Wert von 3.8V, blinkt die LED alle zwei Sekunden um mehr Aufmerksamkeit zu erregen und alle Zusatzfunktionen werden bis auf das nötigste deaktiviert.

Um die minimale Spannungsschwelle des Akkus nicht zu unterschreiten, wird bei einer Akkuspannung von 3.7V das Gerät komplett ausgeschaltet.

Akkuspannung	Modus
4.2V - 4.0V	LED aus
4.0V - 3.8V	LED alle 5s blinken
3.8V - 3.7V	LED alle 2s blinken
3.7V	Gerät herunterfahren

Tabelle 5.2.: Akku Modus

5.4. Wireless Charging

Das Laden des Senders erfolgt über ein Wireless Charging Modul (WCM), die Funktion dieses Moduls als auch der Wirkungsgrad sind in diesem Kapitel beschrieben. Dafür werden Informationen von der Webseite Adafruit [23] und dem Studien Fachartikel über Wireless Charging von dem Bundesamt für Energie aus der Schweiz [24] bezogen.

5.4.1. Funktionsprinzip

Das Funktionsprinzip des verwendeten WCMs ist eine induktives Laden. Das bedeutet das, dass wie bei einem Transformator zwei Luft gekoppelte Spulen direkt Übereinander gehalten werden um die Benötigte Energie zu übertragen.

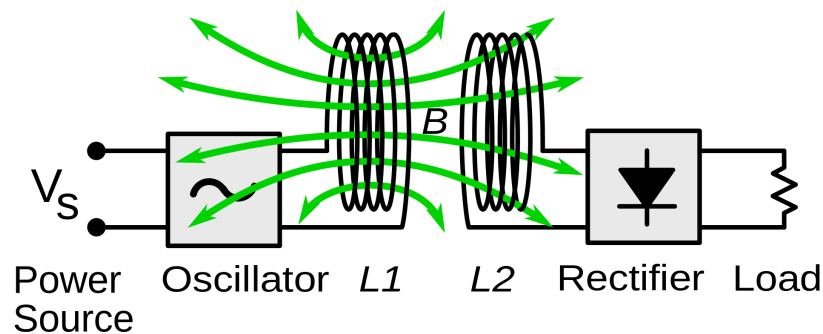


Abbildung 5.10.: Energieübertragung mittels Induktionsspulen [25]

Abbildung 5.10 zeigt das Gundsätzliche Funktionsprinzip eines WCMs. Es können drei Teilsysteme unterschieden werden. Das System des Wireless Charging zeichnet sich durch die komplette Galvanische Trennung des Senders und Empfängers aus. Ein WCM arbeitet in kHz- bis MHz-Bereich, denn hier wird wenig Energie in den freien Raum abgestrahlt. Einmal abgestrahlte Energie kann nur zu einem kleinen Teil vom Empfänger wieder genutzt werden. Das Verwendete System basiert auf magnetischer Induktion. Die Sendespule erzeugt ein magnetisches Wechselfeld welches in der Empfangsspule eine Spannung induziert. Diese Spannung erzeugt wieder einen Strom der nach dem Gleichrichter zum Aufladen des Akkus verwendet wird.

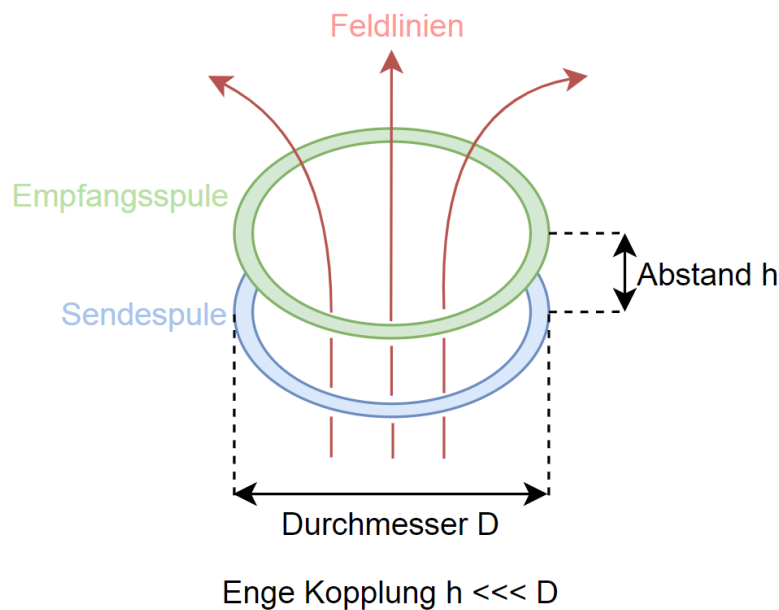


Abbildung 5.11.: Enge Kopplung bei induktivem Leistungstransfer

Abbildung 5.11 zeigt, dass obwohl die Spulen räumlich getrennt sind man immer noch von einer "engen" Kopplung spricht. Durch die starke magnetische Kopplung befinden sich die Stromkreise fast in Antiphasen (180° Phasenverschiebung). Dadurch werden die Magnetfelder beinahe aufgehoben und gibt es nur geringe Streufelder. Allerdings umso weiter die Sende- und Empfangsspule von einander entfernt sind oder seitlich verschoben, desto mehr nimmt die magnetische Kopplung ab.

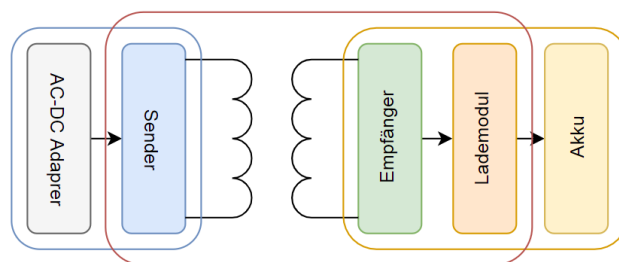


Abbildung 5.12.: Kabelloses Laden Darstellung

Abbildung 5.12 zeigt die Schematische Darstellung des kabellosen Ladens. In der magenta Box sind das WCM eingezeichnet. Rechts ist das Endgerät und links der Sender mit einem AC-DC Adapter zur Stromversorgung.

5.4.2. Benötigter Eingangsstrom

Da es sich um eine Luftkoppelung der Spulen handelt, werden laut Webseite [23] nur etwa 40% (η) der eingespeisten Leistung übertragen. Um festzustellen, wie viel Eingangsstrom die Sendespule benötigt um die gewünschten Ausgangsspannung und Ausgangsstrom auf der Empfangsseite verwenden zu können gilt:

$$I_{\text{Eingang}} = \frac{I_{\text{Ausgang}} * V_{\text{Ausgang}}}{V_{\text{Eingang}} * \eta} \quad (5.2)$$

Wenn man die gewollten Werte von 500mA und 5V am Ausgang und 9V am Eingang einsetzt, ergibt sich ein benötigter Strom von: 694mA $\sim$ 700mA.

5.4.3. Messungen

Die Messung des Stromes abhängig vom Abstand wurde mit einem Multimeter und einem Messband durchgeführt. Der Abstand der Spulen wurde mittels eines dritten armes fixiert und dann die Strommessung durchgeführt. Anschließend wurden verschiedene nicht leitende Materialien zwischen den Spulen gehalten und noch einmal der Strom gemessen. Die Ergebnisse sind in Tabelle 5.3 festgehalten.

Verwendbarer Strom	Abstand zwischen den Spulen
10mA	14mm
100mA - 200mA	7mm
$\sim$ 500mA	2-3mm

Tabelle 5.3.: Übertragener Strom mit wachsenden Abstand

Aus der Tabelle 5.3 kann man den übertragenen Strom in Korrelation zum Abstand auslesen. Materialien, die weder leitend noch eisenhaltig sind, können anstelle von Luft zwischen den Spulen verbaut werden. Die Werte aus Tabelle 5.3 sind dann immer noch gültig. Beispiele für solche Materialien sind Luft, Holz, Leder, Kunststoff, Papier oder Glas.

70mA ist der fixe Ruhestrom, der von der Ladestation gebraucht wird, auch wenn sich die Empfangsspule nicht in der Nähe befindet.

5.5. Schaltungen

In diesem Kapitel wird die Realisierung der Schaltung des Senders beschrieben.

5.5.1. Schaltungsstruktur

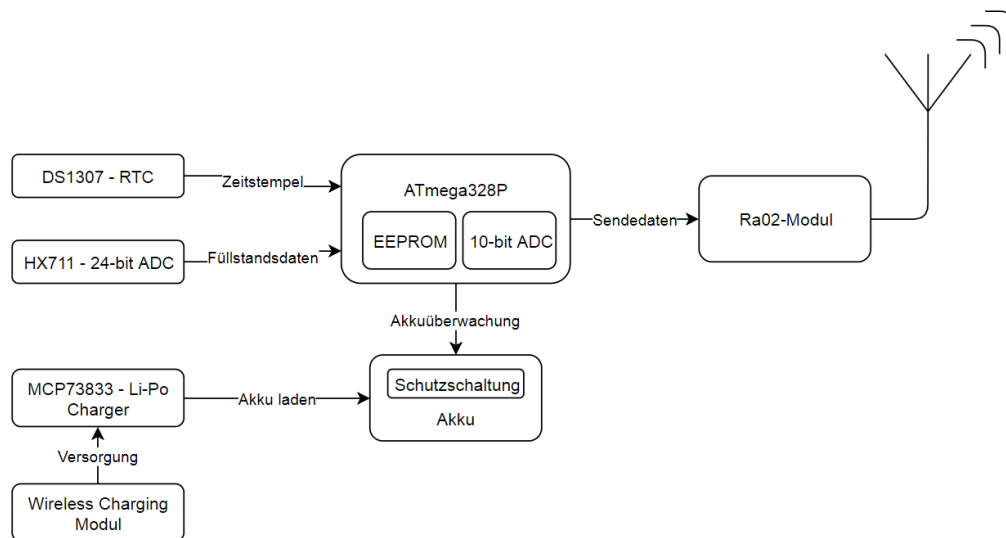


Abbildung 5.13.: Blockschaltbild

In Abbildung 5.13 ist die grundsätzliche Struktur der Schaltung aufgezeichnet. Die Funktionsblöcke RTC, HX711 und das EEPROM liefern dem ATmega328P den aktuellen Zeitstempel, Füllstandsdaten und die Seriennummer des Geräts. Die RTC wird über I²C angesprochen.

Das EEPROM beinhaltet die neunstellige Seriennummer zur eindeutigen Identifikation des Geräts. Mit Hilfe dieser Daten generiert der μC die Sendedaten und leitet sie an das Ra02-Modul über SPI weiter (siehe Kapitel 5.8.2). Das Ra02-Modul sendet die Daten auf 434MHz an ein entsprechendes Gateway.

Das Laden des Akkus erfolgt extern über eine speziell dafür ausgelegten Li-Po Lade IC. Die Akkuüberwachung übernimmt der interne ADC des ATmega328P. Er liefert die Akkuspannung in mV zur weiteren Verarbeitung an den μC zurück (siehe Kapitel 5.8.5).

5.5.2. HEX711 - 24Bit ADC

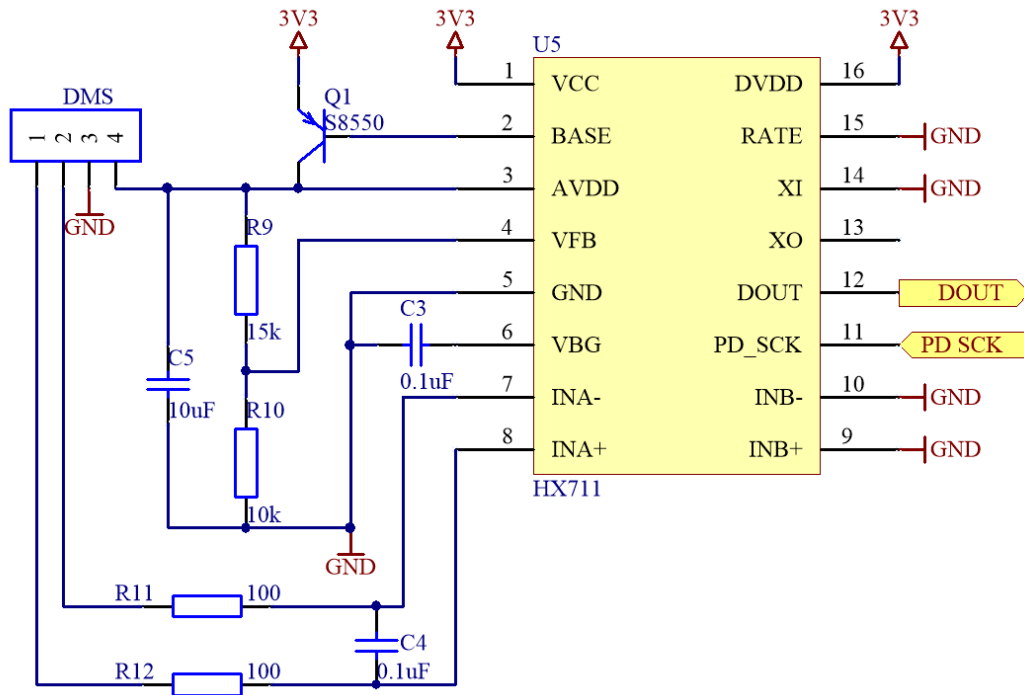


Abbildung 5.14.: HX711 Grundschtung

Die Grundbeschaltung des HX711 findet man im Datenblatt (vgl. Semiconductor Datastheet HX711[12]).

Die nicht verwendeten PIN's INB- und INB+ werden auf GND gelegt um die Störuneempfindlichkeit zu verringern.

Die Funktionsweise des HX711 beruht darauf das er die Spannung an dem Ausgangspin BASE solange anpasst bis die anliegende Spannung an V_{FB} durch einen Spannungsteiler von R9 und R10 bei 1.25V liegt.

Der Arbeitspunkt der V_{FB} liegt bei 1.25V. Um zu vermeiden das eine zusätzliche Gleichspannungsquelle verwendet werden muss wird ein Spannungsteiler zwischen R9 und R10 berechnet.

$$V_{FB} = \frac{R_{10}}{R_9 + R_{10}} * 3V3 \quad (5.3)$$

Bei einer Versorgungsspannung von 3.3V ergeben sich dann folgende Widerstandswerte: $R_9 = 15k$, $R_{10} = 10k$.

5.5.3. Real Time Clock

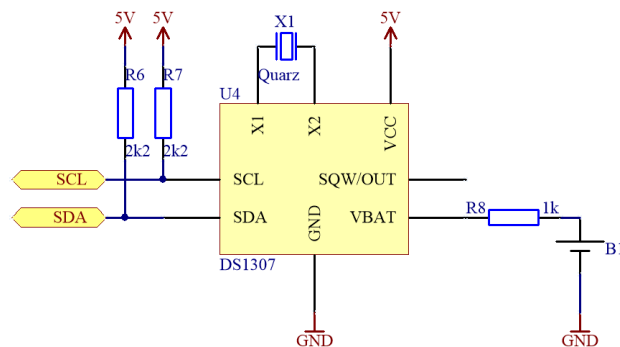


Abbildung 5.15.: RTC Grundsaltung

Auch wenn es nicht unbedingt nötig ist an den DS1307 einen externen Quarz mit 37kHz zu schließen wird trotzdem die Möglichkeit offengelassen und in der Schaltung sowohl als auch dann am PCB der Quarz eingezeichnet und verdrahtet. R1 und R2 stellen Pull Up Widerstände für die I²C Leitung dar. SQW wird nicht verwendet somit bleibt er Not Connected (NC).

5.5.4. LoRa

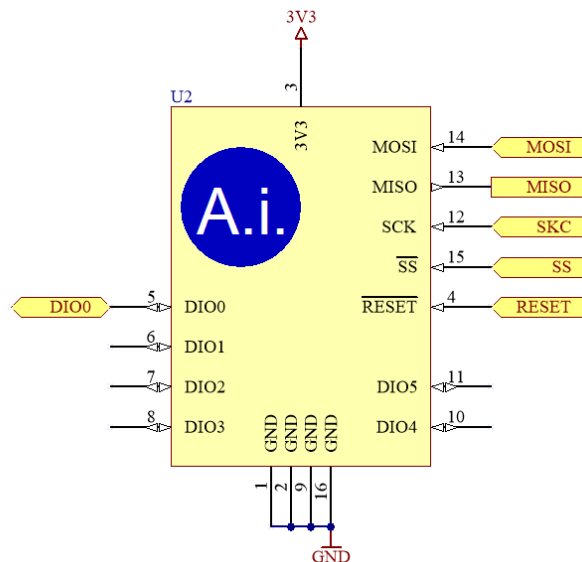


Abbildung 5.16.: Ra02-Modul Schaltung

Der Arduino Nano übermittelt die Sendedaten mittels SPI an das Ra02-Modul.

5.5.5. Akku Management

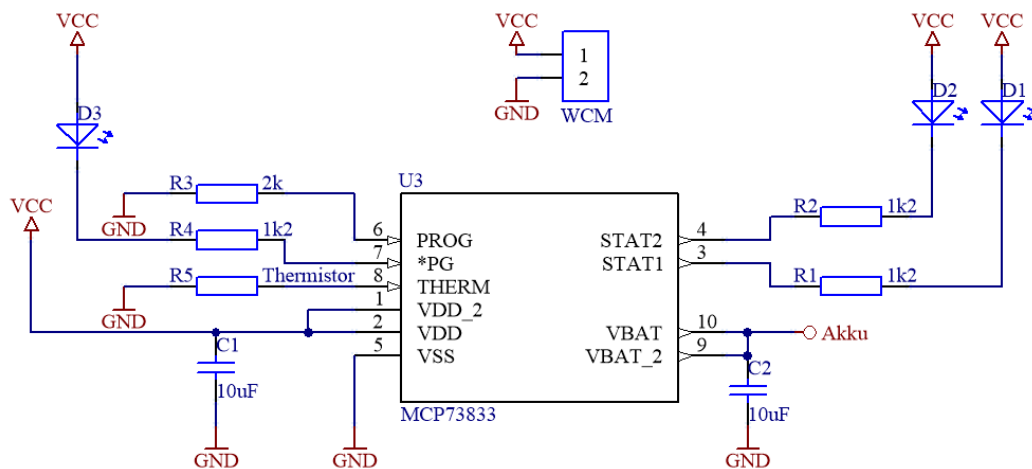


Abbildung 5.17.: Akku Management Grundschtaltung

Für den gewünschten Ladestrom von 500mA wird an PIN 6 ein 2k Widerstand eingelötet. V_{CC} sind 5V allerdings kommen diese 5V von dem Ladegerät und auch nur dann, wenn das Gerät auf der Ladestation steht.

Statusanzeige

Als Statusanzeige werden LEDs mit großer Lichtstärke gewählt, um deren Status auch bei heller Umgebung und geringem Stromverbrauch noch gut erkennen zu können. Dafür werden LEDs mit einer angegebenen Lichtstärke von 100mcd gewählt. Laut Datenblatt [26] hat diese eine Flussspannung von 2,5V. Die verwendete Schaltung ist in Abb.5.17 zu sehen.

Bei einer maximalen Versorgungsspannung von etwa 5V müssen an einem Vorwiderstand maximal 2,5V abfallen. Um den Energiebedarf möglichst gering zu halten, wird die LED nur mit maximal 2mA betrieben. Die genannte Lichtstärke erreicht die LED bei 20mA, eine höhere Lichtstärke ist auf Kosten der Lebensdauer möglich (siehe Abb. 5.18). Aus dem Diagramm geht auch hervor, dass die LED bei 2mA nur etwa ein Fünftel der Lichtstärke, also 20mcd erreichen wird. Das ist mehr als ausreichend für dieses Projekt.

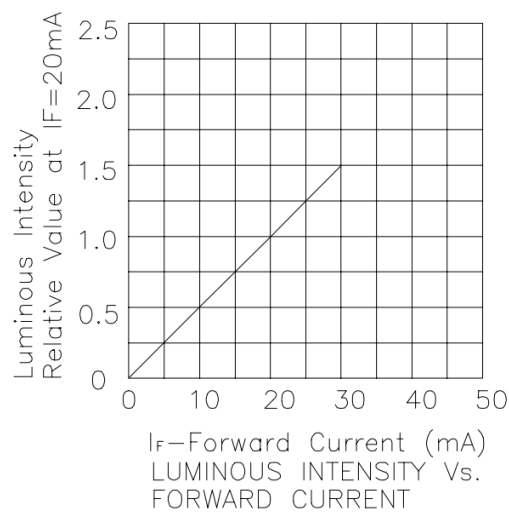


Abbildung 5.18.: Relative Lichtstärke der LED zur Lichtstärke bei 20mA

Mit der folgenden Formel wird der Vorwiderstand der LED berechnet:

$$R_{1-3} = \frac{V_{CC} - V_{LED (max.)}}{I_{LED (max.)}} = \frac{2,5V}{2mA} = 1k2\Omega \quad (5.4)$$

Die verschiedenen LEDs zeigen den Akku Ladezustand an, D1 zeigt an das Power vorhanden ist. D2 leuchtet wenn der Akku geladen wird und D3 wird langsam immer stärker bis der Ladevorgang beendet ist.

Thermistor

Da mit 500mA geladen wird, ist es sinnvoll an Pin 8 einen 10k NTC-Thermistor einzubauen welcher als Temperaturwächter eingesetzt wird. Umso höher der Ladestrom ist umso höher ist die Temperatur des Lade IC's und des Akkus. Dies könnte zu einer Beschädigung bis hin zu einer Zerstörung des Akkus oder des IC's führen. Der Thermistor wirkt dagegen indem der den IC abschalten lässt bei einer zu starken Hitzeentwicklung. Der IC ladet den Akku mit mindestens 3.7V und 500mA laden, das ergibt 650mW. Dem Datenblatt kann man entnehmen das der IC 113°C/W erzeugt. Aufgrund dieser gewaltigen Hitzeentwicklung musste am PCB ebenfalls eine Kühlfläche definiert werden. Mehr dazu siehe in Kapitel 5.6.2.

5.5.6. Gesamt Schaltung

Abbildung 5.19 zeigt die Vollständige Schaltung welche sich aus all den Vorigen Elementen und dem Arduino Nano zusammensetzt.

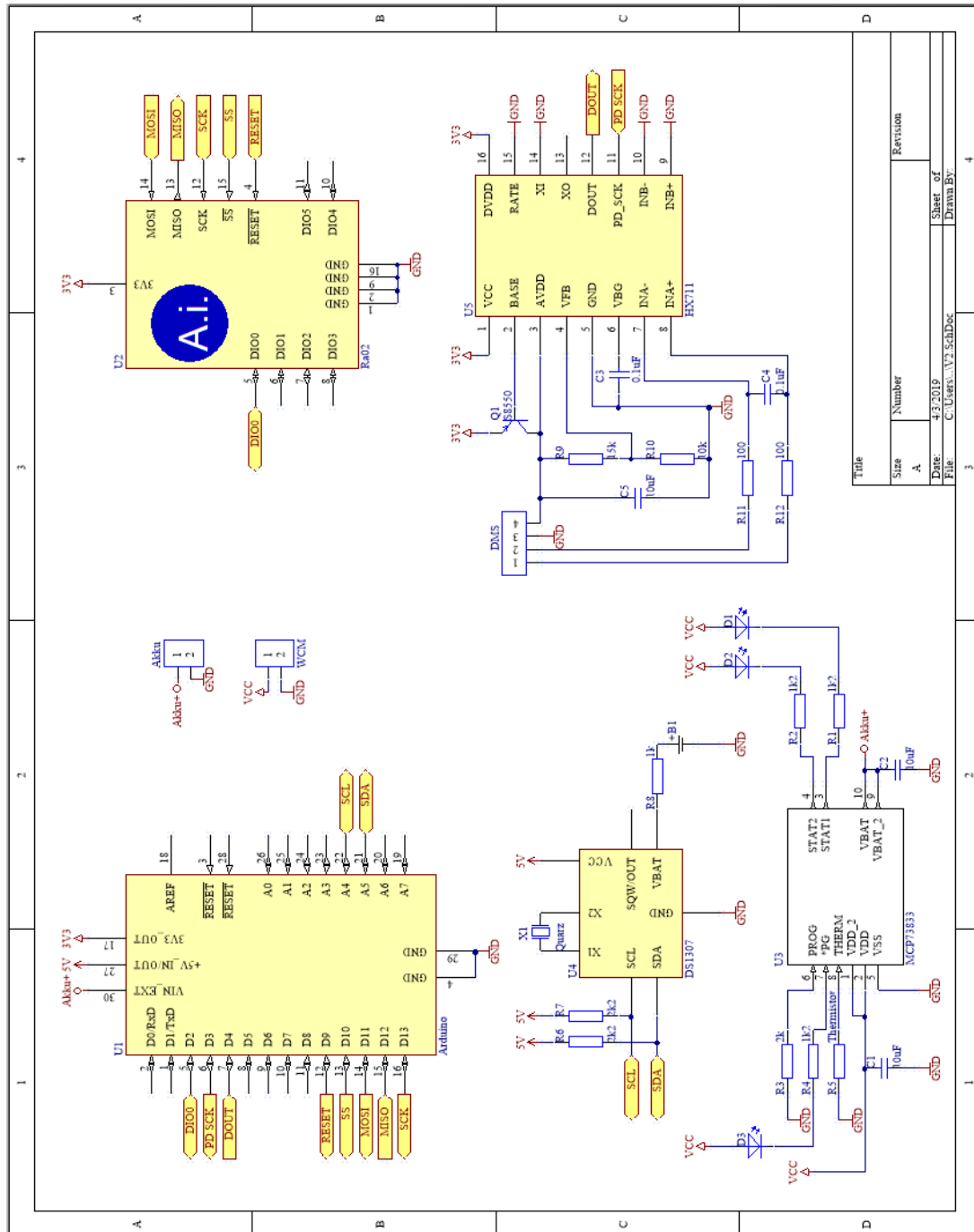


Abbildung 5.19.: Gesamtschaltung

5.6. PCB Entwicklung

Bei der Umsetzung der Schaltung fiel die Entscheidung auf die externe Fertigung eines PCBs. Als Hersteller wird ALL-PCB.com beauftragt, der den mit Altium erstellten Print lediglich fertigt. Die Bestückung erfolgte im Anschluss per Hand, wobei Bauteile von RS-Components bezogen wurden.

5.6.1. Mechanische Anforderungen

- Rechteckige Form, wobei die Seitenlänge kleiner 4.5cm sein sollte
- Befestigung mit 4 Schrauben ($d = M2$) nahe der Ecken des PCBs
- Stabile Schraubenverbindung zur Wägezeile
- Fertigung mit maximal zwei Lagen

Mechanisch musste beachtet werden das die Wiegeplatteform annähernd frei schweben kann. Das bedeutet sie darf von unten auf keinem Bauteil aufliegen da sonst die Messdaten verfälscht werden.

5.6.2. Thermodynamische Anforderungen

- Hitze entwickelnde Bauteile außen
- Kein Kühlkörper

Durch den begrenzten Platz von maximal 4.5cm pro Seitenlänge und das geschlossene Gehäuse ist kein Platz für einen Kühlkörper. 90% der Bauteile wurden in SMD Bauform verwendet. Dadurch kann der Bottom Layer einerseits als GND Plain, gegen EMV Störungen verwendet werden, andererseits auch als Kühlfläche. Abbildung 5.20 zeigt den Querschnitt des PCB's wo die Wärmeübertragung für FR-4-Leiterplatten verbessert wurde indem thermische Durchkontaktierungen (Vias - Plated Through-Holes, PTH) zwischen leitenden Leiterplatten hinzugefügt wurden.

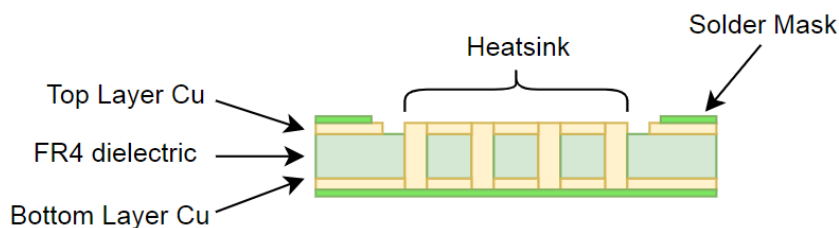


Abbildung 5.20.: Querschnitt des PCB's

5.6.3. Verwendete Bauteile

Bevor das Entwickeln des PCBs los gehen kann müssen die Eigenschaften der realen Bauteile wie z.b. geometrische Abmessungen, Befestigungsmethode, Wärmeentwicklung, usw. festgelegt werden. Die "Bill of Materials" kurz BOM in Tabelle 5.4 fast die verwendeten Bauteile kurz und Übersichtlich zusammen. 90% der verwendeten Bauteile sind in SMD Ausführung verwendet worden.

Comment	Description	Designator	Footprint	Quantity
Arduino Nano	IO Board	U1	Arduino_nano	1
Ra02	LoRa Modul	U2	AI_RA02	1
DS1307	RTC	U3	SO-8	1
HX711	ADC	U4	SO-16	1
MCP73833	Li-Ion Charger Manager	U5	MSOP-10	1
KO	Kondensator	C1 - C5	0805	5
LED	Diode	D1 - D3	LED-3	1
DMS	Stecker	J1	Sip-4	1
Batterie, Akku, WC	Stecker	J2 - J4	SIP-2	3
S8550	PNP Transistor	Q1	SOT23	1
RES	Widerstand	R1 - R11	0805	11
Quarz	Quarz	X1	Quarz	1

Tabelle 5.4.: Bill of Materials

5.6.4. PCB Layout

Die vom Hersteller festgelegten Routing Regeln können auf der Webseite [27] nachgelesen werden.

Das Layout in Abbildung 5.21 und 5.22 wurde unter der Beachtung der gegebenen Anforderungen erstellt.

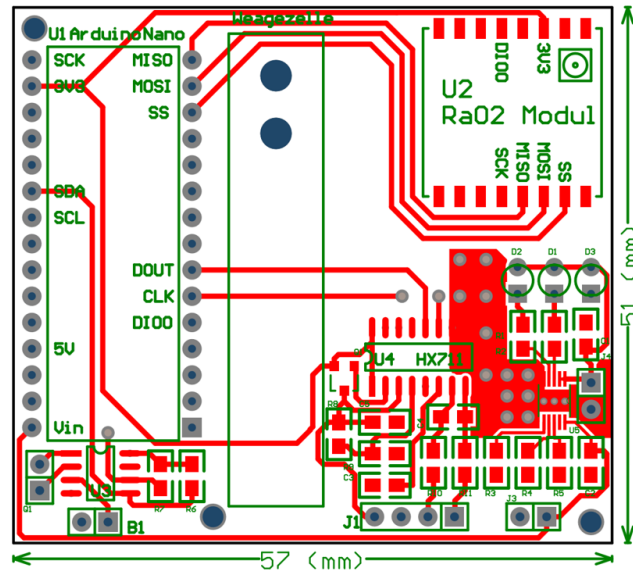


Abbildung 5.21.: Ergebnis des Routings des Senders - Top-Layer

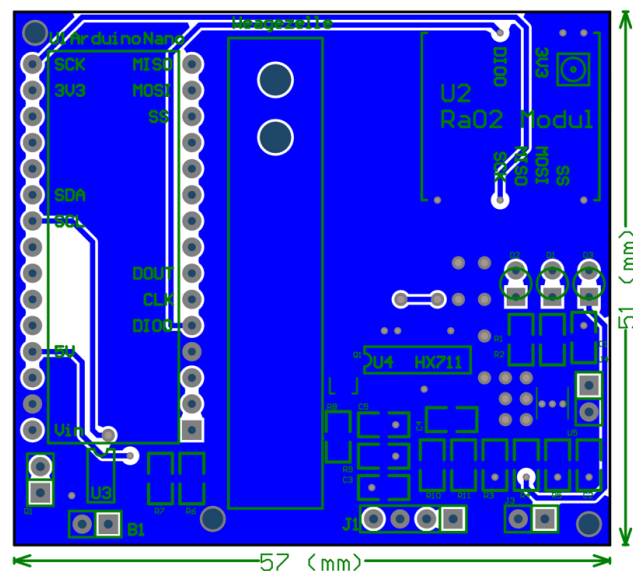


Abbildung 5.22.: Ergebnis des Routings des Senders - Bottom-Layer

5.7. Gehäuse Design

Das Gehäuse Design gliedert sich in zwei Teilbereiche. Der erste Teilbereich ist der Untersetzer für sich und der Zweite die Ladesstation auf die der Untersetzer gestellt werden muss wenn man ihn aufladen möchte/muss. Beide Teile kommen aus dem 3D Drucker. An jeden Teilbereich sind spezielle Anforderungen gestellt welche in den nächsten Seiten behandelt werden.

5.7.1. Untersetzer

Die Anforderungen an den Untersetzer sind:

- Runde Form, wobei der Durchmesser nicht größer als 10cm sein sollte
- 3mm Status-LEDs gut sichtbar platziert
- Wasser- und Staabdicht laut IP33
- Maximale Dicke von 5cm
- Wiegeplattform nicht einsperren
- Halterungen für Akku und das Wireless Charging Modul
- Wasserabweisendes Gehäusematerial

5.7.2. Ladesstation

Die Anforderungen an die Ladestation sind:

- Runde Form, wobei der Durchmesser nicht größer als 11cm sein sollte
- Inkludiertes Kabel, mit einer Mindestlänge von 30cm
- Wasserabweisendes Gehäusematerial

5.7.3. Gehäuse - Untersetzer

Das Gehäuse teilt sich in zwei Teil. Der erste Teil ist der Boden des Untersetzers. Er bildet das Herzstück des Untersetzers, denn an ihm sind alle notwendigen Bauelemente fixiert. Der zweite Teil ist ein Ring. Er bildet die Verkleidung des Bodens und dient als Rahmen.

Boden

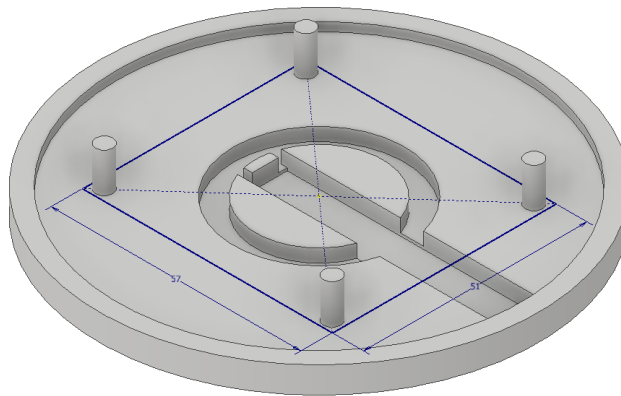


Abbildung 5.23.: Herzstück des Untersetzers

Abbildung 5.23 zeigt den entwickelten Boden des Untersetzers. Insgesamt hat er einen Durchmesser von 97mm. Die Kreiseinkerbung in der Mitte des Untersetzers bildet die Nische für das Wireless Charging Modul. Das blaue Viereck kennzeichnet die Größe des PCB's mit 57mm x 51mm. Die vier Abstandshalter an den vier Ecken des PCB-Platzes dienen zum Abstand halten gegenüber dem Akku welcher zwischen Wireless Charging und PCB Montiert wird.



Abbildung 5.24.: Lade Modul

Abbildung 5.24 zeigt den fertigen Boden des Untersetzers mit eingesetzten Wireless Charging Modul.

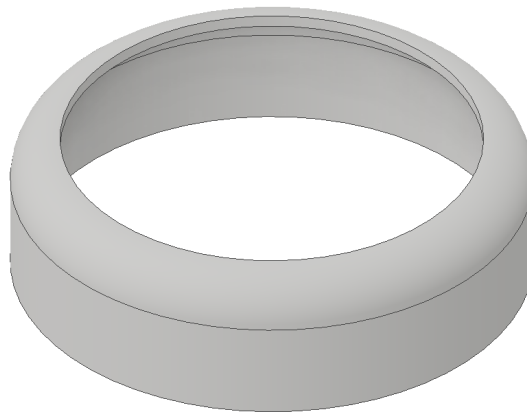
Rahmen

Abbildung 5.25.: Rahmen des Untersetzers

Abbildung 5.25 zeigt den Rahmen des Untersetzers. Der Rahmen ist insgesamt 3cm hoch und hat an der oberen Kante eine Rundung nach innen. Da sie leicht über die Wiegeplattform steht verhindert sie ebenfalls das Eindringen von Wasser oder Staub.

Zusammengebautes Gehäuse

Abbildung 5.26.: Rahmen des Bodens

Abbildung 5.26 zeigt das zusammengebaute Gehäuse des Untersetzers.

5.7.4. Statusanzeigen



Abbildung 5.27.: Power



Abbildung 5.28.: Ladet



Abbildung 5.29.: Fertig geladen



Abbildung 5.30.: Bitte Laden

Die LED's wurden so platziert, dass sie von unten leicht durch das Milchglas des Untersetzers hindurch leuchten. Abbildung 5.27 bis 5.30 zeigt die verschiedenen Modi der Statusanzeige. Wie schon zuvor beschrieben leuchtet eine blaue LED um zu signalisieren, dass der Untersetzter auf der Ladestation platziert wurde, und eine Spannung induziert wird. Eine orange LED um zu signalisieren das der Akku geladen wird und eine grüne LED die beginnt zu leuchten, wenn der Akku fertig geladen ist.

Zusätzlich zu den Statusanzeigen blau, orange und grün, welche nur Statisch leuchten und nur fürs Laden zuständig sind. Gibt es noch eine rote Statusanzeige für das Entladen des Akkus. Die rote LED ändert ihre Blinkfrequenz je nach Akku Kapazität um den Benutzer zu signalisieren, dass der Untersetzter geladen werden muss.

5.7.5. Ladestation

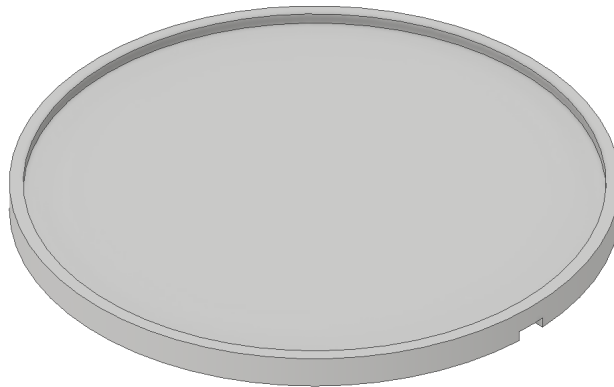


Abbildung 5.31.: Ladestation oben

Abbildung 5.31 und 5.32 zeigt die Ladestation von oben und von unten. Die Ladestation ist schlicht gehalten verfügt nur über ein Wireless Charging Modul welches in die runde Einkerbung am Boden inkludiert wird. Die kleine Ausnehmung am rechten Rand der Ladestation in Abbildung 5.31 ist für das Ladekabel gedacht.

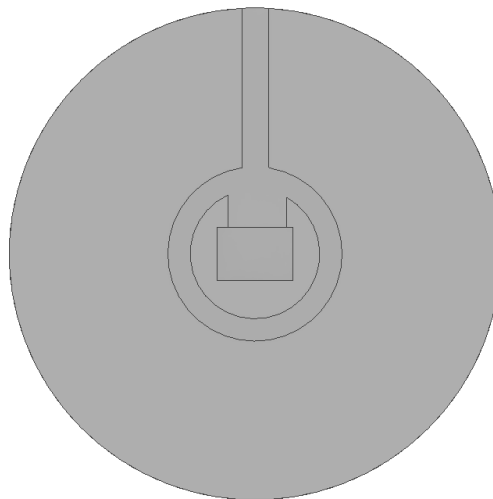


Abbildung 5.32.: Ladestation unten

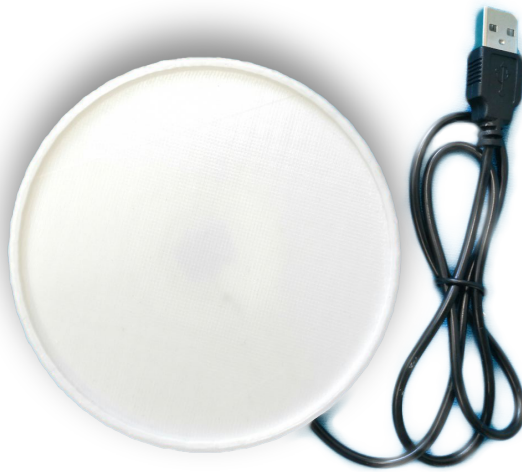


Abbildung 5.33.: Ladestation real

Abbildung 5.33 zeigt die Realisierung der Ladestation von oben. Die Abbildung zeigt ebenfalls das Ladekabel der Ladestation. Das Ladekabel ist ein 50cm langes schwarzes Kabel mit einem USB-Stecker Typ A am Ende, zum Anstecken an einen USB-Wandadapter, der mindestens 700mA liefern kann. Insgesamt hat die Ladestation einen Durchmesser von 11cm.

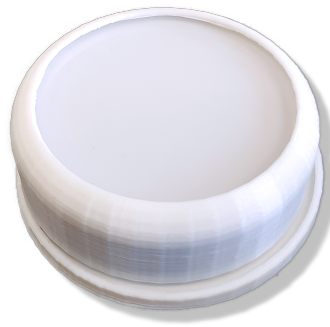


Abbildung 5.34.: Untersetzer auf der Ladestation

Abbildung 5.34 zeigt den Untersetzer auf der Ladestation bereit zum Laden.

5.7.6. Materialwahl

Alle Informationen zur Materialwahl kommen von der Technischen Universität aus Dresden [28]. Die Materialwahl fiel auf PLA. PLA ist englisch und steht für Polylactic oder auch Polymilchsäure. Es ist biologisch abbaubar und besteht zum größten Teil aus Maisstärke oder Zuckerrohr. Abbildung 5.35 zeigt die zwei verschiedenen Arten zur Herstellung von PLA auf chemischer Basis. Die erste Methode geht über die direkte Polykondensation von Milchsäure. Die zweite Methode ist die Ringöffnungspolymerisation über einen Zwischenschritt von ringförmigen Lactiden.

Bei der ersten Methode wird das Abfallprodukt Wasser erzeugt und es muss schnell abgepumpt werden um hohe Reinheitsgrade zu erzielen. Das erschwert die Herstellung von PLA erheblich und es wird dadurch nicht besonders oft verwendet.

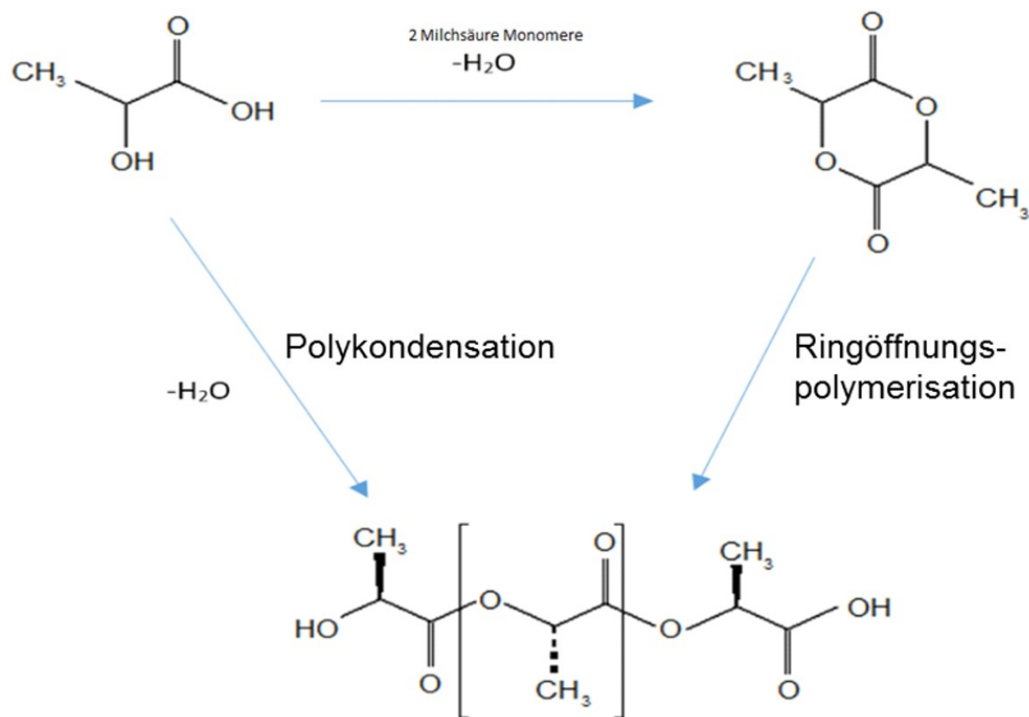


Abbildung 5.35.: PLA Herstellung auf chemischer Basis [28]

PLA wird häufig zur Herstellung von Folien, Dosen, Becher oder Schalen verwendet, da es für den Menschen ungiftig ist. Auch ist PLA nicht brennbar und besitzt eine Bruchfestigkeit von 60MPa. Diese Parameter waren ausschlaggebend für die Entscheidung PLA als 3D-Druck Material zu verwenden. Auch das Gehäuse des Gateways besteht aus PLA.

5.8. Software

In diesem Kapitel wird die allgemeine Struktur der Software beschrieben. Hierbei werden Informationen aus den Datenblättern EEPROM [17] und RTC [13] sowie des LoRa-Moduls [29] verwendet.

5.8.1. Softwarestruktur

Abbildung 5.36 zeigt die grundlegende Struktur der Software.

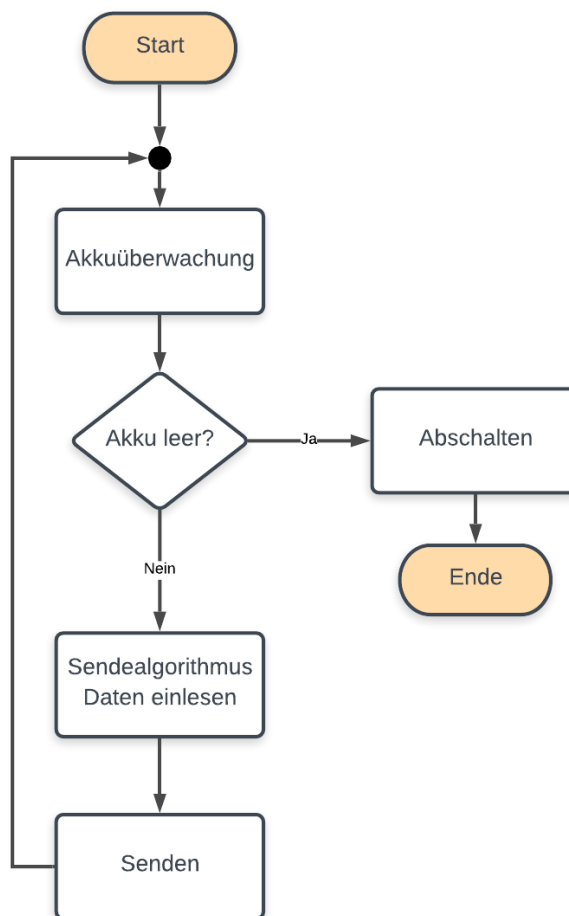


Abbildung 5.36.: Sender Softwarestruktur

Die oberste Priorität hat in der Software die Akkuüberwachung. Ist der Akku leer wird das Gerät abgeschaltet. Trifft das nicht zu, wird der Datenblock den der Sendealgorithmus erzeugt, eingelesen und mit dem LoRa-Modul verwendet. Nach der Sendung eines Datenpakets wird das Programm von vorne gestartet.

5.8.2. LoRa Konfiguration

Zur Konfiguration des Ra02 mussten folgende Parameter angepasst werden:

```
1      LoRa.setSpreadingFactor(7);
2      LoRa.setCodingRate4(1);
3      LoRa.setSignalBandwidth(125E3);
4      LoRa.setFrequency(434E6);
```

setSpreadingFactor ist der Faktor wie weit das Signal gespreizt wird. Man kann einen Wert von 6 - 12 einstellen. Voreingestellt ist 7. Wenn ein Spreizfaktor von 6 eingestellt ist, muss der implizite Header Mode zum Senden und Empfangen von Paketen verwendet werden.

setCodingRate4 bezeichnet die pro Codewort übertragenen Informationssymbole im Verhältnis zur Länge der Wörter (=R). Einstellbare Werte liegen von 1-4 (=C). Der Codieratenzähler ist fest auf 4 eingestellt. (=n).

$$R = \frac{n}{C + 4} \quad (5.5)$$

setSignalBandwidth steht die zu verwendende Bandbreite ein. Unterstützte Bandbreiten sind: 7.8kHz, 10.4kHz, 15.6kHz, 20.8kHz, 31.25kHz, 41.7kHz, 62.5kHz, 125kHz, and 250kHz.

setFrequency setzt die Frequenz mit der der Sender sendet. Unterstützte Frequenzen sind: 434MHz, 868MHz, 915MHz

Verwendete Einstellungen Wie schon in Kapitel 4.4.2 beschrieben wurde ein Spreading Factor (SF) von 7, eine Coding Rate von 1, eine Bandbreite (BW) von 125kHz und eine Frequenz von 434MHz verwendet.

5.8.3. Packete Senden

Um ein Paket zu senden muss es zuerst geöffnet werden, dann hineingeschrieben was gesendet werden soll und dann wieder geschlossen. Jedes Packet kann bis zu 255 Bytes aufnehmen.

```
1      LoRa.beginPacket();
2      LoRa.print("Hello_World");
3      LoRa.endPacket();
```

beginPacket startet des Sendevorgangs eines Packetes. Liefert 1 wenn das Funkgerät sendebereit ist und 0 wenn es besetzt oder ausgefallen ist.

print/write schreibt die Daten in das Packet entweder Byteweise mit definierter Länge oder als String. Es liefert ebenso die Anzahl der geschriebenen Bytes zurück.

endPacket beendet das Senden eines Packetes. Liefert 1 bei Erfolg zurück und 0 bei Misserfolg.

5.8.4. LoRa Energiesparmodus

Da der Sender nicht die ganze Zeit Daten senden muss, kann man das Ra02 Modul in sleep oder standby Modus setzten und immer nur dann aufwecken wenn man ein neues Packet senden möchte. So spart man Akku ohne ständig neu booten zu müssen.

```
1      LoRa.sleep();
2      LoRa.idle();
```

sleep versetzt das Ra02 Modul in Schlafmodus, das bedeutet es werden alle Funktionen heruntergefahren und nur der Prozessor wird versorgt damit die Konfiguration nicht vergessen werden. Dabei verbraucht das Modul $0.6\mu\text{A}$.

idle versetzt das Ra02 Modul in Ruhezustand. Hier bleiben mehr Funktionen erhalten. So auch die serielle Kommunikation mit dem Arduino Nano, allerdings benötigt es dann 1.4mA .

5.8.5. Akku Überwachung

Um eine vollständige Zerstörung des Akkus zu verhindern wurde eine Akku Überwachung mit dem Atmega328P implementiert. Der an Board befindliche Analog Digital Converter ermöglicht dieses ohne einen weiteren IC verbauen zu müssen. Das Programm muss den gegebenen ADC-Wert in einen verwendbaren VCC-Wert umrechnen. Das wird wie folgt beschrieben durchgeführt.

Berechnungen

Der ADC arbeitet nach folgender Formel:

$$ADC_{\text{Wert}} = \frac{V_{\text{Eingang}} * 1023}{V_{\text{Referenz}}} \quad (5.6)$$

Die interne Referenzspannung oder auch "Bandgap Voltage" V_{BG} , welche bei einem Atmega328P 1.1V beträgt, kann man als Eingangsspannung einsetzen. Nun entspricht $V_{\text{Eingang}} = V_{BG}$ als nächstes kann man sagen das $V_{\text{Referenz}} = V_{CC}$.

Neue Werte eingesetzt und umgeformt ergibt sich:

$$V_{CC} = \frac{V_{BG} * 1023}{ADC_{\text{Wert}}} \quad (5.7)$$

V_{CC} entspricht nun dem Wert der Versorgungsspannung in mV. Daher das V_{BG} einen fixen Wert von 1.1V hat ergibt sich für $V_{BG} * 1023$ ein fixer Wert von 1125300. Mit Hilfe der Formel 5.7 kann nun in einem Programm der aktuelle Wert der Versorgungsspannung berechnet werden.

Programmstruktur

Abbildung 5.37 zeigt die Programmstruktur der Software zur Überwachung des Akkusstandes.

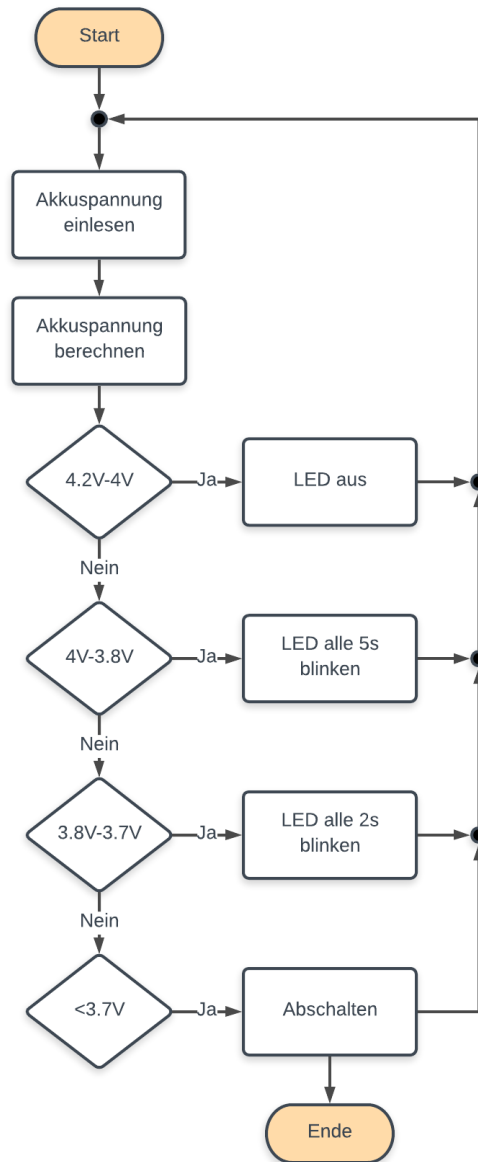


Abbildung 5.37.: Akkuüberwachungsprogrammstruktur

5.8.6. Messdaten

Die Messdaten sind der wichtigste Teil vom Sendalgorithmus-Datenblock. Sie werden über ein DMS-Modul und einen 24Bit ADC (HX711) eingelesen. Der HX711 kommuniziert mit dem Arduino Nano über die serielle Schnittstelle I²C.

Kalibrierung

Das Messequipment muss einmalig vor dem Gebrauch kalibriert werden da es eine natürliche Dehnung und Spreizung der DMS gibt und so die Messwerte verfälscht werden könnten. Für die Kalibrierung wurde das folgende Programm geschrieben. Über die serielle Schnittstelle zwischen Atmega328P und dem PC können Befehle an den μ C gesendet werden.

Der `calibration_factor` wurde über eine Messung mit einem Referenz Gewicht, von einer 1€ Münze mit einem Gewicht von 7.5g ermittelt. Dabei ergab sich ein Wert von 1600000.

```
1         float calibration_factor = 1600000;  
2         scale.set_scale(calibration_factor);
```

float calibration_factor ist der Korrekturfaktor der softwaretechnisch bei jedem HX711 individuell nachgestellt werden muss.

set_scale(float x) setzt den Korrekturfaktor softwaretechnisch in der scale Klasse.

Einlesen

Zum Einlesen des derzeitigen Gewichts wurde `get_units()` verwendet.

```
3         x = scale.get_units();
```

get_units() liest das aktuelle anliegende Gewicht aus und gibt es in dem Datentyp float zurück.

5.8.7. EEPROM

EEPROM bedeutet Electrically Erasable Programmable Read Only Memory. Der Arduino Nano besitzt einen Atmega328P Microcontroller welcher einen 1024 Byte großes EEPROM enthält. Im EEPROM wird die Seriennummer zur Authentifizierung des jeweiligen Gerätes gespeichert. Die Seriennummer besteht aus dem Aktivierungsdatum in der Form: YY/MM/DD. Und einer fortlaufenden Nummer mit drei Stellen, startend bei 000.

Auslesen

Es gibt drei Möglichkeiten ein EEPROM auszulesen.

- **Current Address read**

Mit current Adresse read wird das Datenbyte aus der Adresse ausgelesen wo der Adresspointer gerade hin zeigt.

- **Random Address read**

Beim random Adresse read kann genau die Speicheradresse angegeben werden von welchem Speicherplatz ausgelesen werden soll.

- **Sequenziell Address read**

Bei sequenziell Adresse read gibt man die Startadresse an ab der die Daten ausgelesen werden sollen.

Verwendet wurde random Adresse read. Abbildung 5.38 zeigt die Bitfolge an der I²C Leitung der Funktion Random Address read.

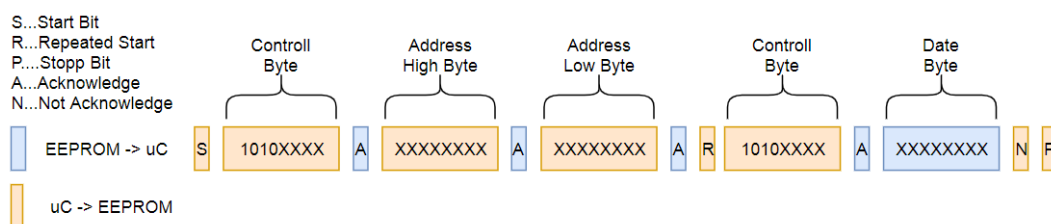


Abbildung 5.38.: Random Address Read

Mit dem nachfolgenden Befehl kann Random Address read verwendet werden.

```
1 EEPROM.read ( Adresse );
```

Speichern

Auch beim reinschreiben gibt es zwei Möglichkeiten.

- **Byte write**

Bei Byte write wird genau die Speicheradresse im EEPROM angegeben bei der das Datenbyte gespeichert werden soll.

- **Page write**

Bei Page write wird eine Startadresse angegeben, ab der bis zu 32 Bytes hintereinander gespeichert werden können. Wenn danach keine neue Startadresse angegeben wird, und weitere Datenbytes gesendet werden kommt es zu einem Rollover. Das heißt die alten Daten werden beginnend bei der ersten Speicheradresse überschrieben. Page write geht nur innerhalb eines Pages und nicht über Pagegrenzen hinaus.

Byte write wurde verwendet. Abbildung 5.39 zeigt die Bitfolge der Funktion Byte write am I²C Bus.

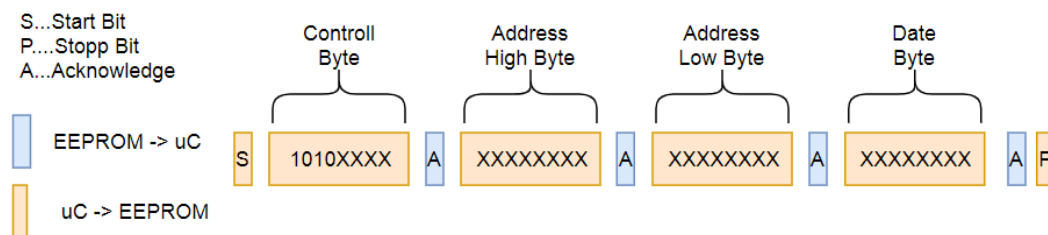


Abbildung 5.39.: Byte write

Mit dem Nachfolgenden Befehl kann Byte write verwendet werden.

```
2      EEPROM.write(Adresse , Wert);
```

write schreibt an der angegebenen Adresse den angegebenen Wert in das EEPROM. Jeder Speicherplatz kann einen Wert in der Größe von 0-255 halten.

5.8.8. Sendealgorithmus

Der Sendealgorithmus ist ein eigens ausgedachter Algorithmus, mit dem die Füllstands Differenz feststellen und die zu versendenden Datenblöcke kreiert werden. Abbildung 5.40 zeigt die Programmstruktur des Sendealgorithmus.

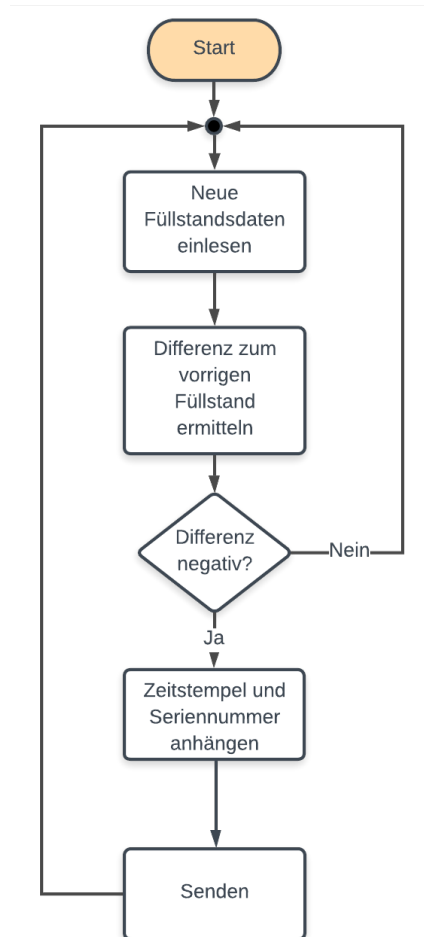


Abbildung 5.40.: Datenflussstruktur

Die Differenz der Füllstandsdaten müssen ermittelt werden, damit das Diagramm in Abbildung 7.8 erstellt werden kann.

6. Empfänger

Die vom Sender gesendeten Daten werden vom Empfänger, welcher durch einen Raspberry Pi in Verbindung mit einem Ra02-Modul realisiert wird, empfangen. Danach werden die Daten in der Konsole ausgegeben und in einer Datenbank gespeichert, um sie später über eine Webapplikation auslesen zu können.

Spezifikationen zum Ra02-Modul siehe Tabelle 4.2.

6.1. Aufsetzen des Raspberry Pi als Empfänger

Der Raspberry Pi benötigt ein Betriebssystem, um in weiterer Folge darauf arbeiten zu können. Für die Projektarbeit wurde **Raspian Stretch with desktop and recommended software** als Betriebssystem gewählt.

Raspian Stretch kann auf der offiziellen Raspberry-Webseite in drei verschiedenen Ausführungen heruntergeladen werden. Raspian Stretch ist zurzeit die aktuellste Raspian Version (Veröffentlichungsdatum 2018-11-13).

6.1.1. Erstmaliges Booten und Grundkonfigurationen

Zuerst wurde der Raspberry Pi an die folgende Peripherie angeschlossen:

- HDMI-Kabel
Je nach Bildschirm wird ein Adapter (DVI, VGA) benötigt.
- Bildschirm
- Tastatur
- Maus
- Raspberry Pi Netzteil

Insofern sich ein Schalter am Netzteil befindet, kann dieser nun eingeschalten werden. Ansonsten fährt der Raspberry Pi von alleine hoch.

Beim erstmaligen Booten öffnet sich ein das *Raspi-Config*-Fenster. Es können nun Daten zur Zeitzone und Region geändert werden.

Danach wird gefragt, ob das Filesystem vergrößert werden soll. Das dient der besseren Ausnutzung der Speicherkartengröße und sollte durchgeführt werden.

Im Anschluss wird der Raspberry Pi neu gebootet.

Nach erneutem Hochfahren verlangt der Raspberry Pi nach einem Login und Passwort. Folgende Daten sind dafür einzutragen:

```
1      login: pi
2      password: raspberry
```

Listing 6.1: Erstmaliges Einloggen

Wenn das Einloggen erfolgreich verlief, kann das Terminal geöffnet werden. Nach Eingabe des Befehls *raspi-config* öffnet sich erneut ein Konfigurationsfenster. Dort können ein neues Passwort und Hostname eingestellt werden. Weiters wurden in den *Interface-Options* SPI und SSH aktiviert.

Anschließend kann man das das Konfigurationsmenü verlassen und den Raspberry Pi mit *apt-get update* und *apt-get upgrade* aktualisiert werden.

6.1.2. SSH Verbindung zum Raspberry Pi

Um sich von seinem Arbeitsplatz (Laptop) zum Raspberry Pi verbinden zu können, wird SSH verwendet. Dazu wird ein SSH-Client (zB. PuTTY) und die IP-Adresse des Raspberry Pi's benötigt.

Es gibt verschiedene Möglichkeiten, um die IP-Adresse des Raspberry Pi's herauszufinden. Eine der Möglichkeiten wird in Folge erläutert.

6.1.3. Einstellen einer statischen IP-Adresse

Nach dem Anschließen des Raspberry Pi's an die oben aufgelistete Peripherie wird er hochgefahren und es erscheint der Desktop. Um festzustellen, welche IP-Adresse der Raspberry Pi besitzt, wurde der Befehl *ifconfig* in das Terminal eingegeben.

Mit dieser IP-Adresse kann man sich nun über einen SSH-Client mit seinem Raspberry Pi verbinden. Damit wird anstelle aufgelisteten Peripherie nurmehr zB. ein Laptop mit Verbindung zum Netzwerk benötigt, um Zugriff auf seinen Pi zu erhalten.

Im Anschluss wurde die Datei */etc/network/interfaces* bearbeitet, bis der *eth0*-Eintrag dem folgenden Codebeispiel glich:

```
1      auto eth0
2      allow-hotplug eth0
3      iface eth0 inet static
4      address IP-ADRESSE
5      netmask NETZMASKE
6      gateway GATEWAY-ADRESSE
7      dns-nameservers DNS-SERVER-ADRESSE
```

Listing 6.2: IP-Adressen Konfiguration

Die im oben gezeigten Code großgeschriebenen Codeteile wurden durch die eigenen Adress-Daten ersetzt. Um die passenden Daten für das Schulnetzwerk zu erlangen, wurde Rücksprache mit den zuständigen Fachlehrkräften gehalten.

6.1.4. LoRa - Hardwarekonfiguration

Der Raspberry Pi ist wie in der folgenden Abbildung mit einem Ra02-LoRa-Modul verbunden, welches die gesendeten Daten empfängt und an den Raspberry Pi weitergibt.

Ra-02 LoRa	RaspPi GPIO
MOSI	GPIO 10
MISO	GPIO 9
SCK (SCLK)	GPIO 11
NSS	GPIO 7 (CE1)
DIO0 (IRQ)	GPIO 25
DIO1	GPIO 24
DIO2	GPIO 23
DIO3	GPIO 18
RST (Reset)	GPIO 22
LED	GPIO 27

Abbildung 6.1.: Verkabelung Raspberry Pi - Ra02-Modul

Nach dem Verbinden des Raspberry's mit dem Modul ist der Empfänger fertig aufgebaut. Für das Projekt erfolgte der Aufbau auf dem Steckbrett.

Antenne

Da das LoRa-Modul Daten empfangen soll, benötigt es eine Antenne. Die Abbildung 4.2 zeigt die Pinbelegung des LoRa-Moduls, sowie den Anschluss (U.FL-Stecker) der Antenne in der linken oberen Ecke. Die Antenne besitzt eine Fläche von etwa zwei Quadratzentimeter.

Versorgung

Das LoRa-Modul wird über *Pin Nr. 1 (3V3)* des Raspberry Pi's versorgt.

6.1.5. LoRa - Softwarekonfiguration

Auf dem Raspberry Pi müssen mit die passenden Bibliotheken installiert werden, um die gesendeten Daten empfangen und verarbeiten zu können.

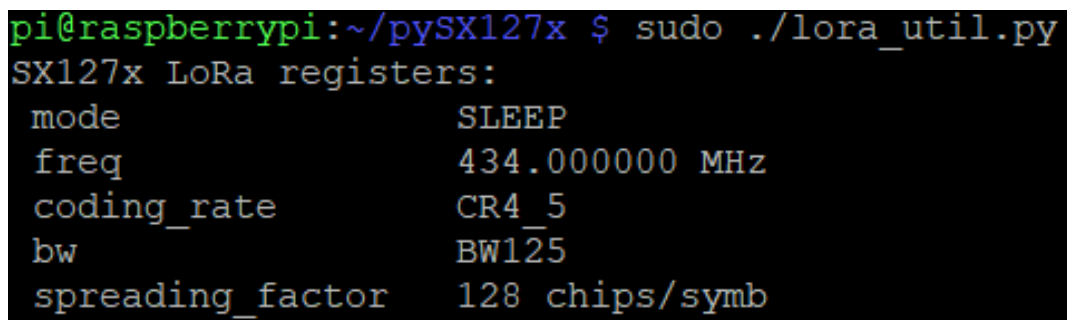
Dazu müssen mit *pip install* die unten aufgelisteten Pakete heruntergeladen werden. *pip* ist ein Programm zur Verwaltung von Python-Paketen [30].

- **SPIdev**
Spidev ist ein Python-Modul, welches eine Schnittstelle zum SPI-Bus des Rasperrys ermöglicht.
- **pyLoRa**
pyLora [31] ist ein Erweiterungsmodul zum Senden und Empfangen von Daten, durch einen LoRa-Transceiver, welcher auf einem Semtech SX127 IC basiert.

Danach wird in der Datei *bashrc* der Environmentvariable PYTHONPATH [32] der Pfad */home/pi/pySX127x/* zugewiesen.

6.2. Anzeige der aktuellen Konfiguration

Zur Anzeige der derzeitigen Konfiguration muss das *lora_util.py* - Programm ausgeführt werden. Dies geschieht mit der Befehlseingabe *python ./lora_util.py*. Darunter finden sich, wie in Abbildung 6.2. dargestellt, folgende Daten:



```
pi@raspberrypi:~/pySX127x $ sudo ./lora_util.py
SX127x LoRa registers:
mode                SLEEP
freq                434.000000 MHz
coding_rate         CR4_5
bw                  BW125
spreading_factor    128 chips/symb
```

Abbildung 6.2.: Aktuelle Raspberry Pi - Lora Konfiguration

Wichtig ist, dass die Frequenz, der Coding- und der Spreading-Faktor mit dem Sender übereinstimmen.

6.3. Empfangen und Verarbeiten der Daten

Nun werden die bereits gesendeten Messdaten empfangen. Da ein Datenpaket aus mehreren einzelnen Daten (Datum, Uhrzeit, getrunkene Menge, Seriennummer) besteht, muss jedes Datenpaket zerteilt werden, um es richtig in die Datenbank speichern zu können.

Das folgende Kapitel enthält Ausschnitte aus der Projektarbeit SSenden mit LoRaäus der HTBLuVA St.Pölten.

6.3.1. Empfangen der gesendeten Daten

Zum Empfangen der gesendeten Daten muss das Programm `rx_cont.py` im Ordner `pySX127x` ausgeführt werden.

Danach wird erneut die aktuelle LoRa-Konfiguration ausgegeben.

Bei Klick der Enter-Taste wird der Empfangsvorgang gestartet. Wenn Daten empfangen werden, werden diese sofort in der Konsole ausgegeben. Ein Datensatz wird in jeweils eine Zeile geschrieben.

Die Ausgabe der gesendeten Daten mit dem bereits von der Bibliothek vorgegebenen Programms `rx_cont.py` erfolgt in ASCII-Code, da das Programm noch keine Decodierung vornimmt.

Die folgenden Kapitel behandelt die Umwandlung des ASCII-Codes zu Text.

Das Programm beendet sich nicht nach einer bestimmten Zeit von selbst, somit muss es mit `Strg + C` abgebrochen werden.

6.3.2. Umwandlung des ASCII-Codes zu Text

Um die Daten richtig weiterverarbeiten zu können, muss der ASCII-Code in Text umgewandelt werden. Die dazu benötigten Code-Zeilen werden in das bereits vorhandene Programm `rx_cont.py` hinzufügt.

Um nun die empfangenen Daten decodieren zu können, ist es wichtig zu wissen, dass die Daten in einer Liste mit dem Namen `payload` gespeichert werden. Da nur ein einzelnes Listenelement, nicht aber die ganze Liste decodiert werden kann, muss mithilfe einer Schleife jedes einzelne Listenelement mit der Funktion `unichr()` in Unicode umgewandelt werden.

Abbruch des Empfangsprogramms erfolgt erneut durch `Strg + C`.

6.3.3. Splitten des Datenpakets

Wie bereits erwähnt wird ein Datenpaket gesendet, welches insgesamt vier verschiedene Arten von Daten beinhaltet: Datum, Uhrzeit, getrunkene Menge und die Seriennummer des Untersetzers.

Im folgenden Code-Abschnitt wird die *payload-Liste* in einen Unicode-String *value* umgewandelt. Der bereits definierte String *splitValues* beinhaltet die bei `'/'` gesplitteten Daten. Der Grund für das Zerteilen an dieser Stelle ist, dass die Daten beim Senden des Datenpakets durch ein `'/'` getrennt wurden.

```
1 payloadlength=len(payload)
2 while(i<payloadlength):
3     value = value + str(unichr(payload[i]))
4     i+=1
5     splitValues = value.split('/')

```

Listing 6.3: Umwandlung von ASCII in Text

Weiters kann mit *splitValues[0-3]* auf die gewünschten Daten zugegriffen werden. So werden, um die Leserlichkeit des Codes zu vereinfachen, die Ergebnisse in neue Variablen gespeichert:

```
date = str(splitValues[0])
time = str(splitValues[1])
difference = str(splitValues[2])
serialnumber = str(splitValues[3])

```

6.4. Speichern der verarbeiteten Daten in die Datenbank

Das folgende Kapitel beschäftigt sich mit dem verarbeiten der empfangenen Daten, damit sie im Anschluss korrekt in die Datenbank eingetragen werden können.

6.4.1. Datenbankstruktur

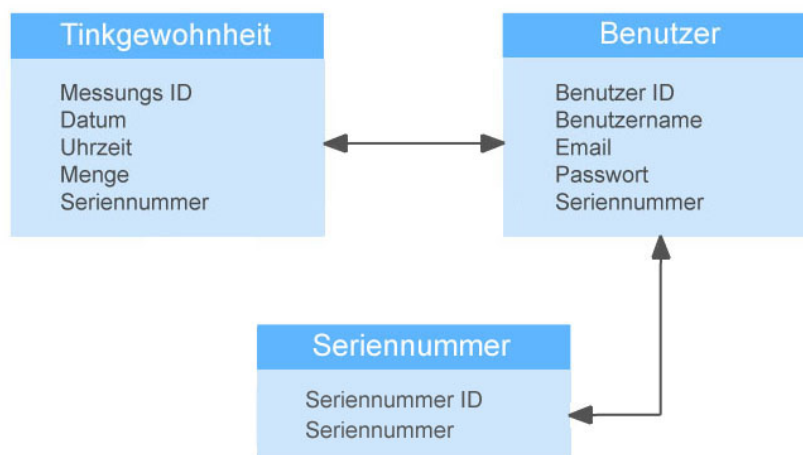


Abbildung 6.3.: Datenbankstruktur

In der Datenbank *Drinkable* gibt es drei Tabellen, wie in Abbildung 6.3 dargestellt:

- **Trinkgewohnheit**

In die Trinkgewohnheit-Tabelle werden die empfangenen Daten eingetragen. Mit jedem neuen Datensatz entsteht automatisch eine neue Messungs-ID. Über die Seriennummer wird die Trinkgewohnheit-Tabelle mit der Benutzer-Tabelle verknüpft. Damit können Daten aus der Trinkgewohnheit-Tabelle für einen bestimmten Benutzer ausgegeben werden.

- **Seriennummer**

In der Seriennummer-Tabelle befinden sich die vorhandenen Seriennummern der Untersetzer. Der Benutzer kann sich nur dann registrieren, wenn die von ihm eingegebene Seriennummer mit einer Seriennummer aus dieser Tabelle übereinstimmt. Die Seriennummern müssen zuvor angelegt werden. Das dient dem Zweck, dass die Datenbank bei Manipulation der

Seriennummer im Untersetzer nicht mit falschen Daten befüllt werden kann.

- **Benutzer**

In der Benutzer-Tabelle stehen alle registrierten Benutzer mit einer eindeutigen ID und Seriennummer. Bei der Seriennummer handelt es sich ebenfalls um die Seriennummer des Untersetzers, welche in der Tabelle einzigartig ist. Ebenso darf die E-Mailadresse nicht öfter als ein Mal in der Tabelle vorhanden sein.

6.4.2. Apache installieren

Um den Raspberry Pi als Webserver (auf dem anschließend eine Datenbank läuft) verwenden zu können, muss Apache darauf installiert werden [33].

```
1 sudo apt-get install apache2
```

6.4.3. MySQL und phpMyAdmin installieren

MySQL ist ein Datenbankverwaltungssystem. Folgende Schritte zur Installation [34] wurden ausgeführt:

Zuerst muss auf dem Raspberry Pi der MySQL-Server installiert werden.

```
1 sudo apt-get install mysql-server
```

Um auf der Datenbank etwas ändern zu können, muss die folgende Zeile eingegeben werden.

```
1 sudo mysql -u root -p
```

Die nächste Code-Zeile ermöglicht eine Interaktion zwischen einem Python-Programm und MySQL.

```
1 sudo apt-get install python-mysqldb
```

Der nächste Schritt besteht darin, einen Benutzer für die Datenbank anzulegen und diesem schließlich Rechte zuzuweisen.

```
1 GRANT ALL PRIVILEGES ON mydb.* TO 'username'@'  
localhost' IDENTIFIED BY 'password';
```

Nun kann phpMyAdmin installiert werden, um auch im Webbrowser auf die Datenbank zugreifen und Änderungen vornehmen zu können. Ebenso muss die Apache-Konfigurationsdatei geöffnet werden.

```

1      sudo apt-get install phpmyadmin
2      sudo nano /etc/apache2/apache2.conf

```

Am Ende der eben geöffneten Konfigurationsdatei wird dann die folgende Zeile eingefügt. Anschließend wird der Apache-Service neu gestartet.

```

1      Include /etc/phpmyadmin/apache.conf
2      sudo /etc/init.d/apache2 restart

```

6.4.4. Installieren und Konfigurieren von php

Damit *php*-Dateien auf dem Raspberry Pi funktionieren, die im Anschluss benötigt werden eine dynamische Webapplikation zu erstellen und Datenbankzugriffe zu ermöglichen, wurde folgender Befehl in der Raspberry Pi-Konsole eingegeben:

```

1      sudo apt-get install php

```

Nach der Installation wurde die **php.ini**-Datei bearbeitet, indem `TimeZone()` auf `TimeZone(Europe Vienna)` geändert wurde. Das dient der in Kapitel 7.2.2 erklärten Ermittlung des Tages, denn die in der Webapplikation verwendete Funktion `date()` greift auf die eingestellte Zeitzone des *php.ini* - Files zurück.

6.4.5. Speichern der Daten in die Datenbank

Da die Daten in Kapitel 6.3.3 bereits richtig aufgeteilt wurden, können sie nun in die Datenbank gespeichert werden. Anschließend Erklärungen zu Code-Beispielen beziehen sich erneut auf **rx_cont.py**, welches bereits in den vorigen Kapiteln bearbeitet wurde.

Zuerst muss eine Verbindung zur Datenbank aufgebaut werden. Das geschieht im folgenden Code-Abschnitt. Dieser wurde original in einen *Try-Block* geschrieben, um anfallende Fehler zu beheben. Ein Try-Block testet den beinhalteten Code auf Fehler und wirft eine Exception, insofern einer auftritt. Wesentlich für die Verbindung zur Datenbank ist allerdings der folgende Teil:

```

1 mysql_connection = pymysql.connect(
2     host=HOSTNAME,
3     user=BENUTZER,
4     password=PASSWORT,
5     db=DATENBANKNAME',
6     autocommit=True)

```

Listing 6.4: Datenbankverbindung

Bemerkung: Die in Blockbuchstaben geschriebenen Zuweisungen müssen durch die korrekten Daten ersetzt werden.

Verlief das Verbinden erfolgreich, so wird ein **mysql-connection.cursor()** erstellt. MySQLCursor ist eine Klasse, welche Objekte instanziert, die SQL-Befehle ausführen können und mit dem SQL-Server kommunizieren.

In **sql** wird der gesamte SQL-Einfügebefehl geschrieben. Im Falle der Diplomarbeit muss der Datensatz in die Spalten *Datum*, *Uhrzeit*, *Menge* und *Seriennummer* in die Tabelle *Trinkgewohnheit* eingefügt werden. Für die Werte wurde *%s* als Platzhalter eingefügt, da der String *val* im Anschluss an dieser Stelle eingefügt wird. *val* beinhaltet die empfangenen und verarbeiteten Werte aus Kapitel 6.3.3. Das erstellte Cursor-Objekt führt nun den SQL-Befehl aus. Damit ist das Speichern der Daten in die Datenbank beendet.

```

1 mysql_cursor = mysql_connection.cursor()
2 with mysql_cursor as cursor:
3     sql = "INSERT INTO Trinkgewohnheit_ (Datum, Uhrzeit ,
        Menge , Seriennummer ) VALUES_ (%s,%s,%s,%s) "
4     val = (date , time , difference , serialnumber)
5     cursor.execute(sql , val)
```

Listing 6.5: Datensatz hinzufügen Raspberry Pi

Schließlich müssen sowohl das Cursor-Objekt, als auch die Verbindung beendet bzw. geschlossen werden. Die beiden folgenden Code-Zeilen werden in einen **finally** - Block eingefasst, da diese Zeilen immer ausgeführt werden.

```

1 mysql_cursor.close()
2 mysql_connection.close()
```

Listing 6.6: Beenden der Datenbankverbindung

6.4.6. Zugriff auf die Datenbank mit phpMyAdmin

phpMyAdmin wird in der Adresszeile des Browsers folgendermaßen aufgerufen: *IP Adresse des Raspberry Pi's* + "/" + **phpmyadmin**. Also kann die Adresse beispielsweise so aussehen: <http://10.53.14.200/phpmyadmin/>

Ist die Website erreichbar, so kann man sich mit seinen Login-Daten, die zuvor bei der Installation von MySQL eingestellt wurden, anmelden.

Nach erfolgreicher Anmeldung kann man auf der linken Seite seine Datenbank und die zugehörigen Tabellen erstellen bzw. verwalten. Ebenso kann man auf einfache Weise kontrollieren, ob die empfangenen Daten korrekt sind und ob sie richtig in die Datenbank gespeichert wurden.

6.5. Schaltung

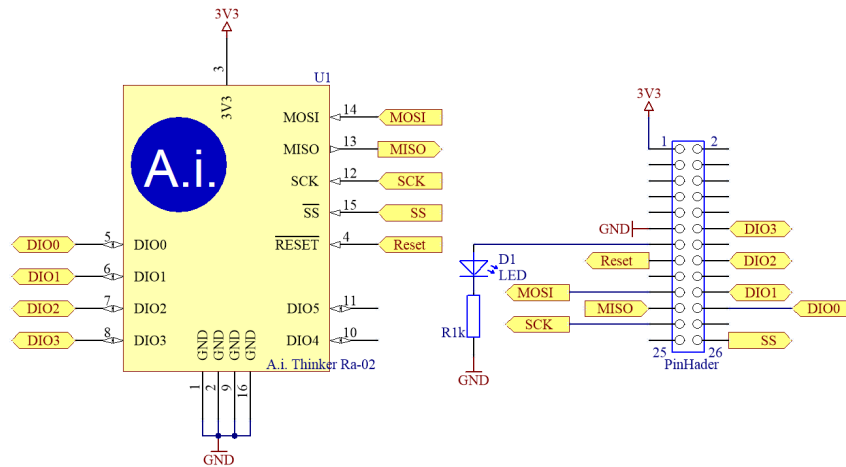


Abbildung 6.4.: Empfänger Schaltung

Die Abbildung 6.4 zeigt die Schaltung des Empfängers. Über die DIO Pins 0:4 und der SPI-Schnittstelle kann der Raspberry Pi mit dem Ra02-Modul kommunizieren. Die Stiftleiste stellt die Verbindung zum Raspberry Pi dar.

6.6. PCB Entwicklung

Bei der Umsetzung der Schaltung fiel die Entscheidung auf die externe Fertigung eines PCB's. Als Hersteller wurde JLCPCB.com beauftragt den mit Altium erstellten Print lediglich zu fertigen. Die Bestückung erfolgte im Anschluss per Hand, wobei Bauteile von RS-Components bezogen wurden.

Die Thermodynamischen Anforderungen an das PCB sind:

- Hitze entwickelnde Bauteile außen
- Kein Kühlkörper

Die Mechanischen Anforderungen an das PCB sind:

- Rechteckige Form, wobei die Seitenlänge kleiner 5cm sein sollte
- Stabile Steckverbindung zum Rasbeeri Pi mittels Pfostenstecker
- Fertigung mit maximal zwei Lagen

6.6.1. Verwendete Bauteile

Bevor das Entwickeln des PCB's los gehen kann, müssen die Eigenschaften der realen Bauteile wie z.b. geometrische Abmessungen, Befestigungsmethode, Wärmeentwicklung, usw. festgelegt werden. Die "Bill of Materials" kurz BOM in Abbildung 6.1 fasst die verwendeten Bauteile kurz und übersichtlich zusammen.

Comment	Description	Designator	Footprint	Quantity
LED	Diode	D1	LED-3	1
CON2x13	Steckerleiste	J1	PIN13	1
RES	Widerstand	R1	Axial-0.4	1
CO	Kondensator	C1	0805	1
A.i. Thinner Ra02	Ra02 Modul	U1	AI_RA02	1

Tabelle 6.1.: Bill of Materials

6.6.2. Routing

Die vom Hersteller festgelegten Routing-Regeln können auf der Webseite [27] nachgelesen werden.

Das Layout in Abbildung 6.5 und Abbildung 6.6 wurde unter der Beachtung der gegebenen Anforderungen erstellt. Wichtig für den späteren Einbau ist vor allem die Verbindung zum RaspberryPi. Sie wurde oben am äusseren Rand des PCB's platziert. Das Ra02-Modul wurde so platziert das es aus ausreichend Platz gibt für das Anlöten.

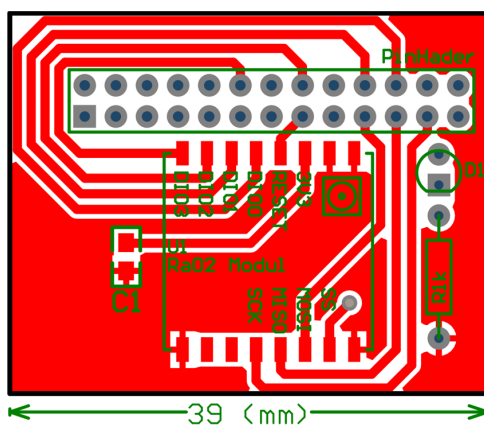


Abbildung 6.5.: Top-Layer

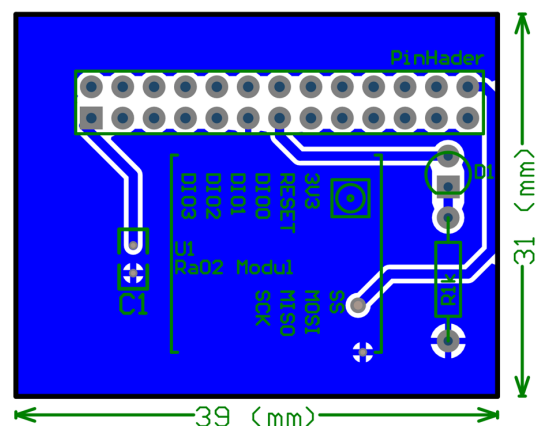


Abbildung 6.6.: Bottom-Layer

6.7. Gehäuse Design

In diesem Kapitel wird das Gehäuse Design des Gateways beschrieben.

6.7.1. Gehäuse Anforderungen

Die Anforderungen an das Gateway Gehäuse sind:

- Rechteckige Form, wobei Länge und Breite kleiner als 10cm sein sollten
- 3mm Status-LEDs gut sichtbar platziert
- Gut zugängliche Position des USB Anschluss zum aufladen
- Maximale Höhe von 5cm
- Stabiles Gehäusematerial

Das Gehäuse Design beschränkt sich auf ein umfassendes Gehäuse des Raspberry Pi's, in welchem ebenfalls das Ra02-Modul untergebracht ist. Ebenfalls bietet es Zugangsmöglichkeiten zu allen wichtigen Ports. Das Gehäuse wird mit einem 3D Drucker gefertigt. Das verwendete Material ist PLA, wie in Kapitel 5.7.6 beschrieben.

6.7.2. Gateway - Gehäuse

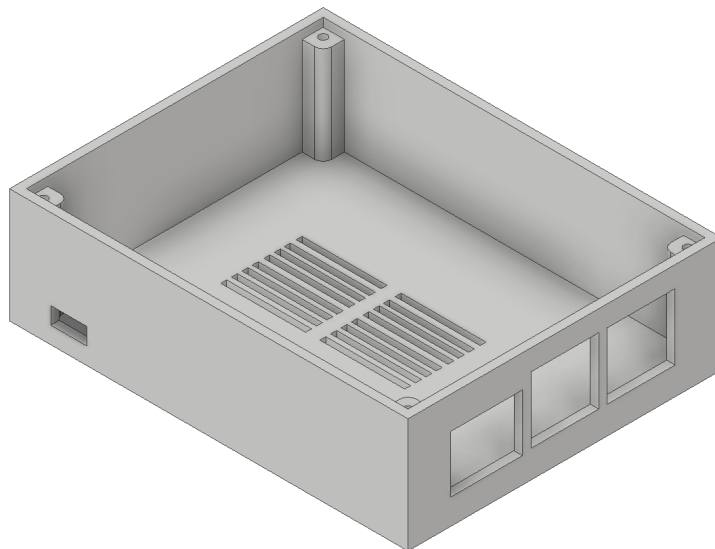


Abbildung 6.7.: Gateway Gehäuse

Abbildung 6.7 zeigt das designte Gehäuse für den Empfänger. Insgesamt hat es eine Größe von 92x72x25mm und eine Wandstärke von 2mm. Vorne sieht man deutlich die drei Ausnehmungen für den LAN-Port und die zwei USB-Ports. Auf der linken Seite gibt es auch noch eine Ausnehmung für den Power Stecker für den Raspberry Pi. Die sich am Boden befinden sich Lüftungsschlitze. An den vier Ecken des Gehäuses befinden sich vier Säulen auf welchen der Deckel des Gehäuses montiert wird.

6.7.3. Gateway - Gehäusedeckel

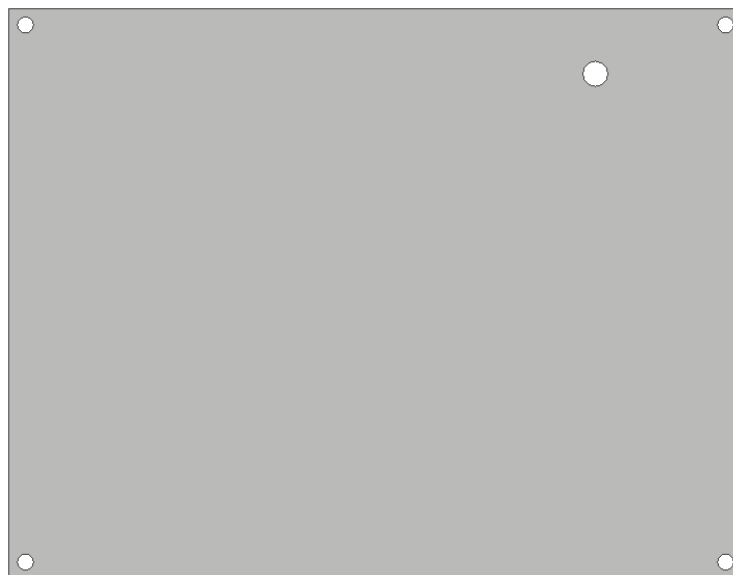


Abbildung 6.8.: Gateway-Gehäusedeckel

Abbildung 6.8 zeigt den Deckel des Gateway-Gehäuses. Die Abmessungen betragen 90x70x2mm. So schließt er perfekt mit der Oberkante des Gateway-Gehäuses ab. An den vier Ecken des Deckels befindet sich vier Löcher für die Montage der Schrauben. Das Loch im rechten Drittel des Deckels ist für die Statusanzeige geplant welche in Kapitel 6.7.4 genauer beschrieben wird.

6.7.4. Zusammengebautes Gehäuse



Abbildung 6.9.: Gateway inkl. Gehäuse

Abbildung 6.9 zeigt den eingebauten Empfänger in seinem Gehäuse. Durch das kleine Loch im Deckel scheint die Statusanzeige mit einer Lichtstärke von 20mcd durch. Sie zeigt an ob, der Raspberry Pi aktiv ist. Auch sieht man auf dem Bild sehr gut, dass der Deckel perfekt mit der oberen Kante des Gateway-Gehäuses abschließt wie geplant. Auch vorne schließen die USB-Ports mit der Außenwand des Gehäuses ab.

7. Webapplikation

Um die gemessenen Daten für den Benutzer anschaulich darzustellen, wurde eine Webapplikation entwickelt. Diese lässt sich auf Desktop- und Mobilgeräten öffnen. Zunächst werden verwendete Webentwicklungstools vorgestellt.

Hypertext Markup Language (html)

Kurz HTML genannt, ist eine textbasierende maschinenlesbare Sprache zur Strukturierung und Verknüpfung von Texten, Bildern, etc. und kann von Webbrowsern dargestellt werden.

Hypertext Preprocessor (php)

php steht für Hypertext Preprocessor und ist eine Open-Source-Scriptsprache. Es dient zum Erstellen von Webanwendungen, ermöglicht Datenbankzugriffe und kann in HTML eingebettet werden.

Cascading Style Sheets (css)

Cascading Style Sheets (CSS) wird zur Formatierung und graphischen Gestaltung des Seiteninhalts verwendet. CSS ermöglicht einen einheitlich gestalteten Webaufttritt.

Mobilansicht

Mobilgeräte besitzen ein anderes Displayformat als Desktopgeräte. Daher muss das CSS teilweise umgeschrieben bzw. dynamische Größen für die Höhe und Breite eines Bildes oder anderen Darstellungen verwendet werden. Stylesheets können mithilfe von *@media* + Bedingung (Displaygröße) ergänzt werden, sodass je nach Displaygröße / Format die richtigen Stylesheets verwendet werden.

Progressive Webapp

Eine progressive Webapp ermöglicht ein Zusammenspiel aus Webinhalten wie HTML, CSS usw. und einer App. Beim öffnen der App wird eine Website geladen und auf dem Mobilgerät dargestellt.

JavaScript

JavaScript ist eine Scriptsprache und ermöglicht dynamisches HTML zur Auswertung von Benutzereingaben oder zum Verändern von Webinhalten.

AJAX

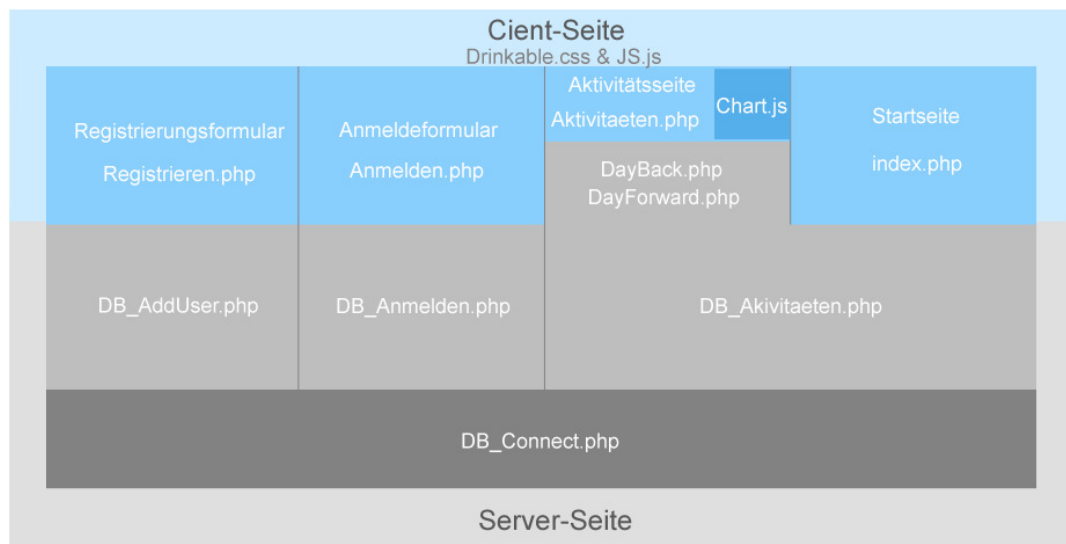
AJAX steht für Asynchronous JavaScript and XML. Es ermöglicht das Senden und Empfangen von Daten zu einem Server. Einzelne HTML- bzw. JavaScript-Elemente können im Hintergrund aktualisiert werden, ohne die Seite vollständig neu aufzubauen.

Chart.js

Chart.js ist eine Open-Source JavaScript-Bibliothek, die es ermöglicht, verschiedene Arten von Diagrammen responsiv darzustellen.

7.1. Aufbau der Webapplikation

Da es sich bei der Desktop- als auch bei der Mobilversion um den gleichen Aufbau handelt, wird dieser im folgenden Abschnitt als *Aufbau der Webapplikation* bezeichnet. Die folgende Abbildung 7.1 veranschaulicht den Aufbau der Webapplikation.



- Verbindung zum Server
- Skripten, die Zugriff auf die Datenbank nehmen. Beinhalten das Skript „DB_Connect“
- Webinhalte für die Benutzeroberfläche. Beinhalten JavaScript und CSS. Benötigen Daten aus den in der Grafik darunterliegenden Skripten.
- Chart.js Bibliothek, die für die Anzeige des Diagramms benötigt wird.

Abbildung 7.1.: Überblick über den Aufbau

7.1.1. Navigation

Nach Eingabe der URL im Browser erscheinen *Log-In-Felder*, die der Benutzer ausfüllen muss. Ist der Nutzer neu, wird er durch Klick auf den *Registrieren* - Button auf die Registrierungsseite weitergeleitet. Sobald er sich angemeldet hat, sieht der Benutzer auf der Startseite (Abbildung 7.2) die Trinkmenge, die er den ganzen Tag über zu sich genommen hat. Um einzusehen, wie viel der Nutzer zu welcher Zeit getrunken hat, muss auf den horizontalen blauen Balken geklickt werden.

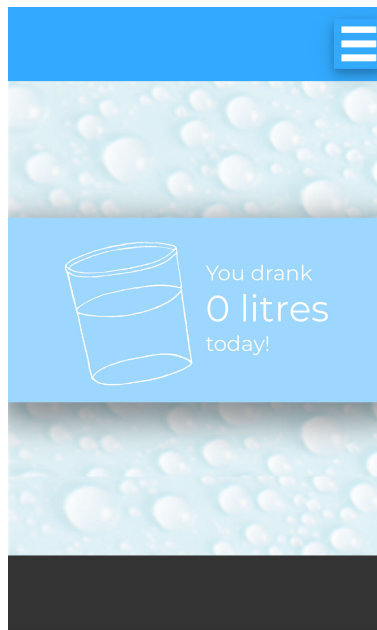


Abbildung 7.2.: Startseite

7.1.2. Bedingungen für die Registrierung

Um sich registrieren zu können, darf die Emailadresse des Benutzers noch nicht in der Benutzer-Tabelle vorliegen. Ebenso benötigt der Benutzer die Seriennummer des Untersetzers, welche allerdings bereits in der Seriennummer-Tabelle vorhanden sein muss. Das dient der Sicherheit, damit sich ein Benutzer nicht mit einer eigens eingetragenen Seriennummer registrieren und somit unnötige Einträge generieren kann. Genauer wird in Kapitel 7.2.2 erläutert. Die Abbildung 6.3 im Kapitel 6.4.1 stellt das Zusammenspiel der einzelnen Tabellen der Datenbank dar. Das erste Zusammenspiel findet bei der Registrierung statt (von Benutzer-Tabelle zu Seriennummer-Tabelle), das Zweite beim Auslesen der Daten und Anzeige im Diagramm (von Benutzer-Tabelle zu Trinkgewohnheit-Tabelle).

7.2. Umsetzung in Code

Zur Umsetzung in Code wurde die Webapplikation in verschiedene Dateien mit unterschiedlichem Zweck aufgeteilt. In diesem Kapitel wird geschildert, wie die Dateien aufgeteilt wurden.

Abbildung 7.3 zeigt den Aufbau der Webapplikation.

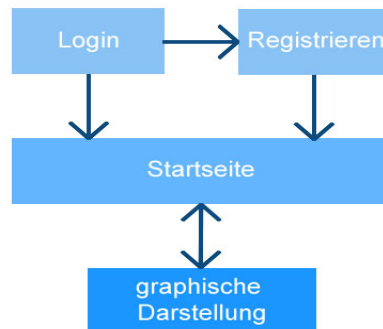


Abbildung 7.3.: Aufbau der Webapplikation

- **Anmelden.php**

Indem man die Emailadresse und das Passwort eingibt, kommt man durch Klick auf den *Login* - Button auf die Startseite. Dies funktioniert nur, insofern der Benutzer bereits in der Datenbank gespeichert ist.

- **Registrieren.php**

Falls der Benutzer noch keinen Account besitzt, kann er sich bei der Plattform registrieren, indem er einen Benutzernamen, eine Emailadresse, die Seriennummer seines Untersetzers und ein Passwort im Registrierungsformular angibt. Durch Klicken des *Account erstellen* - Buttons wird er ebenfalls auf die Startseite weitergeleitet.

- **index.php**

Index.php ist die Startseite. Hier wird die Summe der tagsüber zu sich genommenen Trinkmenge angezeigt. Klickt man auf den horizontalen Balken, in dem die Summe angezeigt wird, wird man auf die Aktivitätsseite (*Aktivitaeten.php*) weitergeleitet.

- **Aktivitaeten.php**

Hier wird die tagsüber zu sich genommene Trinkmenge zur Zeit, an der getrunken wurde, in einem Graphen angezeigt. Die Y-Achse stellt die Trinkmenge (in Liter) dar, die X-Achse die Zeitachse.

7.2.1. im Hintergrund ausgeführte Skripte

Neben den bereits genannten Dateien wurden noch weitere im Hintergrund ausgeführten Skripte verwendet, die keinen Einfluss auf das Aussehen der Webapplikation haben, allerdings wichtig für deren Funktion sind. Dabei handelt es sich größtenteils um Dateien, die Zugriff auf die Datenbank nehmen und somit Datensätze auslesen oder hinzufügen.

- **DB_connect.php**

Diese Datei erstellt eine Verbindung zur MySQL-Datenbank. Eine Verbindung zur Datenbank ist für jedes weitere Skript, das Zugriff auf die Datenbank benötigt, ebenfalls notwendig.

- **DB_Anmelden.php**

Hier wird überprüft, ob die vom Benutzer eingegebenen Daten zur Anmeldung (Nutzername, Passwort) korrekt sind - also ob die Daten im selben Datensatz der Benutzer-Tabelle vorhanden sind. Ist dem der Fall, so wird die Session-Variable des Login-Status auf TRUE gesetzt, um zu signalisieren, dass der Benutzer ab nun eingeloggt ist und der Benutzer wird auf die index.php -Seite weitergeleitet.

- **DB_AddUser.php**

Der Benutzer muss auf der Registrieren.php -Seite seine Daten samt Seriennummer des Untersetzers eingeben. Diese Daten werden folglich an DB_AddUser.php mithilfe einer POST-Variable übergeben. Hier wird nun überprüft, ob sich die eingegebene Seriennummer bereits in der Seriennummer-Tabelle befindet. Ist das richtig, werden die Benutzerdaten zur Benutzer-Tabelle hinzugefügt. Da die Emailadresse und die Seriennummer innerhalb der Tabelle einzigartig sein müssen, kann sich kein Benutzer zweimal mit derselben Emailadresse bzw. Seriennummer anmelden.

- **DB_Aktivitaeten.php**

Hier werden die Daten des aktuellen Tages aus der Trinkgewohnheit-Tabelle geladen. Diese Daten werden dann in einem Objekt an die Datei Aktivitaeten.php übergeben, welche die Daten schließlich in einem Diagramm darstellt. Zum Auslesen der Daten aus der Datenbank wird die Emailadresse in der Benutzer-Tabelle mit der von der Session-Variable übergebenen Emailadresse verglichen und die zum Datensatz gehörige Seriennummer wird mit der Seriennummer aus der Trinkgewohnheit-Tabelle verglichen. Alle Datensätze aus der Trinkgewohnheit-Tabelle, die die gewünschte Seriennummer besitzen, werden ausgelesen - wenn das Datum dem anzuzeigenden Tag entspricht. Beim Laden der Seite wird der aktuelle Tag angezeigt, dieses Thema wird in Kapitel 7.2.2 erläutert.

7.2.2. im Vordergrund ausgeführte Skripte

Hier wird auf die Bereits in Abschnitt 7.2 erwähnten Dateien der Webapplikation näher eingegangen. Bei den infolge erwähnten Dateien handelt es sich um php-Files, welche HTML- bzw. JavaScript-Elemente enthalten.

Anmelden.php

Diese Datei stellt das Anmeldeformular für den Benutzer bereit. Die vom Benutzer eingegebenen Daten (Emailadresse, Passwort) werden in einer POST-Variable an die Datei *DB_anmelden.php* weitergegeben, welche beim Abschicken der Anmeldedaten aufgerufen wird. Ebenso ist auf dieser Seite ein *Create Account* - Button zu sehen. Bei Klick darauf wird man über einen Link zur *Registrieren.php* - Seite weitergeleitet.

Die Abbildung 7.4 zeigt das Anmeldeformular.

The image shows a web form titled 'Drinkable' on a light blue background. It features two white input fields: 'Enter Email' and 'Enter Password'. Below these fields are two blue buttons: 'Login' and 'Create Account', separated by the word 'or'.

Abbildung 7.4.: Anmeldeformular

Registrieren.php

Auf dieser Seite wird das Registrierungsformular (Abbildung 7.5) angezeigt. Der Benutzer muss die Seriennummer des Untersetzers, seine Emailadresse, einen Nutzernamen und ein Passwort in die Felder eintragen. Bei Abschicken der Daten wird die Datei *DB_registrieren.php* aufgerufen und die Daten aus dem Formular werden erneut mit der POST-Methode an die aufgerufene Datei übergeben.

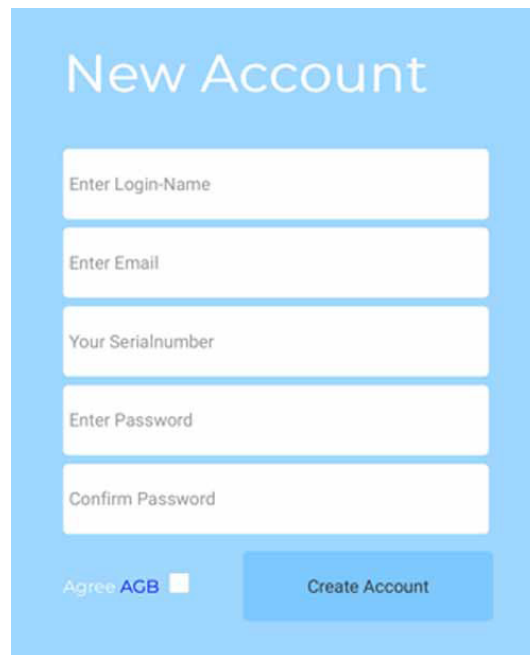
The image shows a registration form titled "New Account" on a light blue background. It contains five input fields stacked vertically: "Enter Login-Name", "Enter Email", "Your Serialnumber", "Enter Password", and "Confirm Password". At the bottom left, there is a checkbox labeled "Agree AGB" with the text "AGB" in blue. To the right of the checkbox is a blue button with the text "Create Account" in white.

Abbildung 7.5.: Registrierungsformular

Klickt man auf das dunkelblau geschriebene *AGB*, so wird man auf eine Seite verlinkt, auf der die Allgemeinen Geschäftsbedingungen aufgelistet sind. Im Falle der Diplomarbeit sind die AGB nur zur Veranschaulichung erstellt worden und somit nicht rechtskräftig.

index.php

Verlief das Anmelden bzw. Registrieren erfolgreich, gelangt der Benutzer auf die Startseite. Auf der Startseite (Abbildung 7.2) erhält man einen Überblick, wieviel man am Tag bereits getrunken hat. Hier kann man die am aktuellen Tag zu sich genommene Gesamttrinkmenge einsehen. Per Klick auf den horizontalen blauen Balken, auf dem die Trinkmenge angezeigt wird, wird man auf die *Aktivitaeten.php* - Seite über einen Link weitergeleitet.

Ebenso gelangt man auf diese Seite, wenn man auf den Menü-Button (Abbildung 7.6) in der rechten oberen Ecke klickt und den Menüpunkt *Aktivitäten* auswählt.



Abbildung 7.6.: Menü

Aktivitaeten.php

Mithilfe der *chart.js*-Bibliothek wird auf dieser Seite ein Diagramm erstellt, welches die zur jeweiligen Zeit zu sich genommene Trinkmenge anzeigt. Dazu wird wie in Abbildung 7.7 dargestellt überprüft, welche Datensätze aus der Trinkgewohnheit-Tabelle dieselbe Seriennummer wie der Benutzer besitzen. Weiters wird das Datum überprüft, denn es werden nur Datensätze benötigt, die an einem bestimmten Tag aufgenommen wurden. Anschließend werden die Daten *Datum*, *Uhrzeit* und *Menge* des richtigen Tages aus der Trinkgewohnheit-Tabelle ausgelesen.

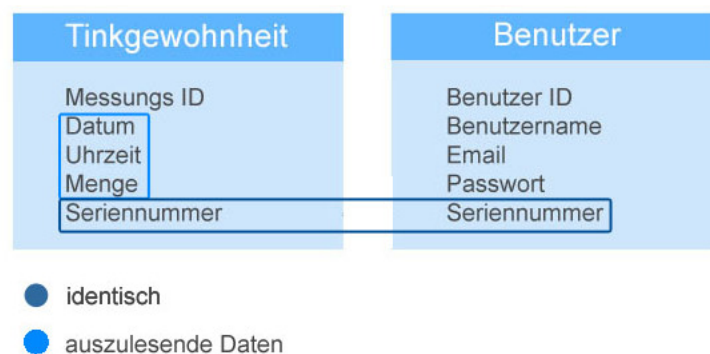


Abbildung 7.7.: Auszulesende Daten

Ermitteln des aktuellen Tages

Um die Daten des aktuellen Tages im Diagramm auf der Seite *Aktivitaeten.php* anzeigen zu können, muss der aktuelle Tag zuerst einmal ermittelt werden. Das geschieht mit der php-Funktion *date()* in der Datei *DB_Aktivitaeten.php*.

```
1    date_default_timezone_set( 'Europe/Vienna' );  
2    $currentDate = date( "Ymd" );
```

Listing 7.1: Bestimmung des aktuellen Tages

Erläuterung zu den in Listing 7.1 verwendeten Funktionen:

- **date_default_timezone_set()**
Wird benötigt, um die richtige Zeitzone im System einzustellen.
- **date()**
Die Funktion *date()* gibt einen Datums-String im angegebenen Format zurück. Wird kein Zeitstempel eingetragen, wird das Datum des aktuellen Tages zurückgegeben. Da im Falle der Diplomarbeit der aktuelle Tag erwünscht ist, wird nur mehr das Format *Ymd* eingetragen, welches den Format *JJJJMMTT* retourniert. Der Grund für dieses Format ist die Weiterverarbeitbarkeit des Wertes im Zusammenhang mit der Datenbank.

Bemerkung: Liefert die *date()*-Funktion allerdings ein falsches Datum zurück, muss die Zeitzone am Raspberry Pi überprüft und neu eingetragen werden. Dies wurde bereits in Kapitel 6.4.4 behandelt.

Erzeugen des Diagramms

Das Diagramm wird wie bereits erwähnt unter Verwendung von *chart.js* erzeugt. *Chart.js* liefert ein breites Spektrum an verschiedenen Darstellungsarten; im Rahmen der Diplomarbeit wurde ein Liniendiagramm gewählt.

```
1    var ctx = document.getElementById( "chart" ).  
        getContext( "2d" );  
2    window.myLine = new Chart( ctx , config );  
3    myLine.config.data.datasets.push( newObj );  
4    myLine.update();
```

Listing 7.2: Erstellen des Diagramms

Erläuterungen zu Listing 7.2:

- **ctx**
ctx ist eine Abkürzung für *context* und ist bei der Erzeugung von Canvas-Elementen stark verbreitet. `getContext("2d")` teilt mit, dass es sich um ein zweidimensionales Diagramm handelt.
- **config**
Hier können diverse Einstellungen zur Diagrammdarstellung vorgenommen werden. Unter anderem der Diagrammtyp (Liniendiagramm, Balkendiagramm, etc.), Farben der Linien, Überschrift, Achsenbeschriftungen, usw. Diese sind auf der offiziellen Webseite von `chart.js` [35] nachzulesen.
- **newObj**
`newObj` beinhaltet die x- und y-Werte der Daten. Der x-Wert beinhaltet die Uhrzeit zu der getrunken wurde und der y-Wert beinhaltet die getrunkene Menge.
- **myLine.update()**
Erzeugt das Diagramm *myLine*.

Beschreibung des Diagramms

Auf der y-Achse des Diagramms ist die Trinkmenge aufgetragen, auf der x-Achse die Uhrzeit, an der die bestimmte Menge zu sich genommen wurde. Es wird nur der für den Tag relevante Zeitbereich angezeigt. Da in Abbildung 7.8 vor 11:00 keine Messungen aufgezeichnet wurden, ist das der Startwert. Ebenso bezieht sich der Maximalwert der Trinkmengen-Achse auf den höchsten gemessenen Wert. Somit ist keine Begrenzung für die täglich mögliche Trinkmenge vorhanden.

Umschalten des anzuzeigenden Tages

Unter dem Diagramm befinden sich zwei Felder, wie in Abbildung 7.8 dargestellt:

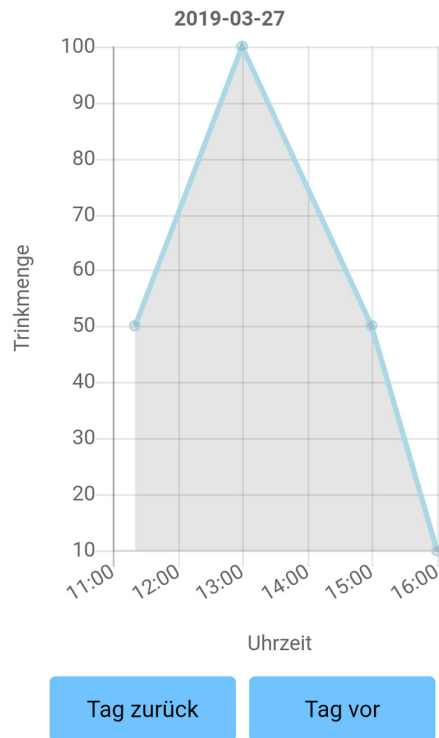


Abbildung 7.8.: Buttons zum Umschalten des anzuzeigenden Tages

Bei einem Klick auf das *Tag zurück* - Feld werden die Daten des Vortages aus der Datenbank ausgelesen. Falls *Tag vor* angeklickt wird, werden die Daten des Folgetages - aus Sicht des aktuell angezeigten Tages - geladen. Mithilfe von AJAX wird eine Anforderung an den Server gesendet und die Daten werden dynamisch neu geladen und das Diagramm aktualisiert.

Bemerkung: Mit dem Begriff *Tag zurück* ist nicht Vortag im Kalender gemeint, sondern der am nächsten in der Vergangenheit liegende Tag mit einem Messeintrag. Falls also heute der 12.01.2019 ist, kann es sein, dass bei Klick auf den *Tag zurück*-Button nicht der 11.01.2019, sondern der 01.01.2019 geladen wird, da es an den anderen Tagen keine Messeinträge gab. Dasselbe gilt auch für *Tag vor*.

Der folgende Code zeigt eine Funktion, die bei Klick des *Tag zurück*-Buttons mit der ID *DateBackButton* eine asynchrone HTTP-Anfrage (AJAX-Request) an den Server stellt.

```
1 $("#DateBackButton").click(function(){
2 $.ajax({url: "DayBack.php",
3         data:{someDate:date},
4         type:"POST",
5         success: function(DatenObject){
6     var tempObj = JSON.parse(DatenObject);
7     }});
8 });
9 );
```

Listing 7.3: AJAX Request

Der AJAX-Request wird mit der *jQuery.ajax()*-Funktion gestellt und beinhaltet die folgenden Einstellungen (*settings*):

- **url**
Beinhaltet die URL, an die die AJAX-Anfrage gesendet wird.
- **data**
Beinhaltet die Daten, die an den Server gesendet werden.
- **type**
Stellt die HTTP-Methode dar und ist standardmäßig auf "GET" gesetzt.
- **success**
Ist die Anfrage erfolgreich, so wird die zugewiesene Funktion ausgeführt.

Im Fall der Diplomarbeit wird wie in Listing 7.3 dargestellt, die Variable *date* innerhalb der POST-Variable *someDate* an die Datei *DayBack.php* übergeben. Konnte die AJAX-Anfrage erfolgreich ausgeführt werden, so wird das Daten-Objekt, welches die Informationen zu Menge (y-Achse), Uhrzeit (x-Achse) und anzuzeigendem Datum für das Diagramm enthält, an die Seite *Aktivitaeten.php* übergeben.

Weiters sind die vom Webserver empfangenen Daten im String-Format. Deshalb müssen die Daten mit *JSON.parse* in ein JavaScript-Objekt umgewandelt werden, um die Werte richtig im Diagramm anzeigen zu können.

8. Progressive Webapp

Eine progressive Webapp ist eine Webseite, die über Klick auf einen App-Icon aufgerufen wird. Dazu wird die Webapplikation, die in Kapitel 7 beschrieben wurde, in eine Android-App eingebettet. Durch das responsive Design (CSS) der Webapplikation wird der Webseiteninhalt passend für Mobilgeräte im Hochformat angezeigt.

Für das Einbinden der Website in einer Android-App wurde die Entwicklungsumgebung Android Studio gewählt, sowie die zum Android Operating System gehörige Komponente Android Webview [36] verwendet. Dazu wurde die Webview-Library inkludiert, mit deren Hilfe man ein Webview-Objekt instanzieren kann.

8.1. Einstellungen im Projekt

Da für eine Webapp ein Internetzugriff erforderlich ist, muss in der Datei *AndroidManifest.xml* eine Erlaubnis dafür erstellt werden. Das AndroidManifest ist zuständig für die Berechtigungen sowie die korrekte Anzeige der App (Icon, usw.). Weiters bestimmt sie die zu Beginn auszuführende Aktivität. Das Internetzugriff wird durch folgende Codezeile in der Datei *AndroidManifest.xml* erlaubt:

```
1 <uses-permission android:name="android.permission.INTERNET"/>
```

Listing 8.1: Internetzugriffserlaubnis

Die in Listing 8.1 gezeigte Codezeile wurde zwischen dem Beginn des Manifests und der Applikation eingefügt.

In der Datei *MainActivity.java* wurde zuerst die Webview-Bibliothek importiert. Danach wurde ein Objekt *mWebView* der Klasse Webview instanziiert. Für bessere Nutzbarkeit wurde eine Funktion für die *Zurück-Taste* am Smartphone implementiert, sodass - falls diese gedrückt wird - man auch in der Webapp auf die vorherige Seite gelangt. Diese sieht wie folgt aus:

```
1 public void onBackPressed() {  
2     if (mWebView.canGoBack()) {  
3         mWebView.goBack();  
4     } else {  
5         super.onBackPressed();  
6     }  
7 }
```

Listing 8.2: Aktivieren der Zurück-Taste

Wichtig ist das Zuweisen der URL zu dem WebView-Objekt:

```
1 mWebView.loadUrl("http://www.Drinkable.com");
```

Listing 8.3: Zuweisen der URL

Bemerkung: Die in Listing 8.3 zugewiesene Domain existiert nicht. Im Rahmen der Diplomarbeit wurde an dieser Stelle die IP-Adresse des Raspberry Pi's eingegeben. Die Eingabe einer Domain im Listing dient der besseren Verständlichkeit.

In der Datei *activity_main.xml* wird das Layout der App bzw. die Größe des einzubettenden Fensters, indem die Webseite schließlich angezeigt wird, eingestellt. Um diese Einstellung vornehmen zu können, wurde innerhalb dieser Datei auf das *Preview*-Feld geklickt, welches sich in einer vertikalen Leiste auf der rechten Seite des Programms befindet. Anschließend öffnet sich das in Abbildung 8.1 dargestellte Fenster:

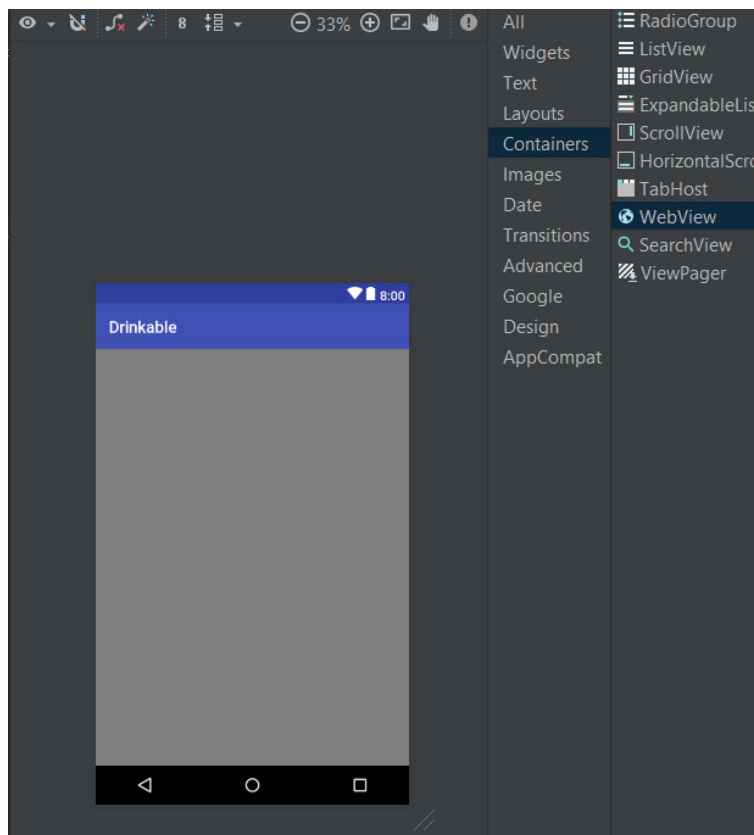


Abbildung 8.1.: Erstellen des Webview-Feldes

Um endgültig die Größe des Fensters zu bestimmen, wurde der in der obigen Abbildung markierte Eintrag *Webview* in das Smartphone-Display-Simulationsfenster gezogen und der Bereich dieses Fensters bis an die Grenzen des Displays gezogen. Damit passt sich der Code in der Datei *activity_main.xml* automatisch an die Einstellungen an.

Einfügen eines Icons

Für das Icon wurde ein Bild eines Tropfens aus dem Internet gewählt und mithilfe von Image Asset Studio [37] eingefügt. Image Asset Studio unterstützt die Dateiformate *.png* und *.jpg*, allerdings ist *.png* bevorzugt.

Im Projekt wurde durch Rechtsklick auf *app/res* Image Asset Studio geöffnet und ein neues *Image Asset* erzeugt. Dort kann man die Form des Icons wählen, sowie den zugehörigen Bildpfad eintragen, wie in Abbildung 8.2 dargestellt:

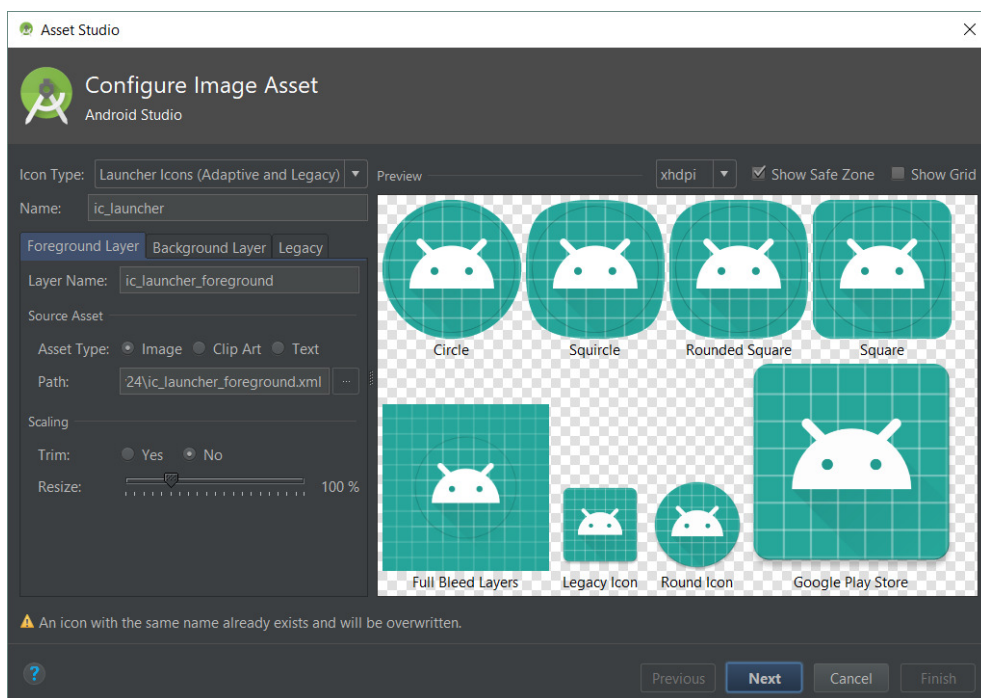


Abbildung 8.2.: Image Asset

Mit dem im Rahmen der Diplomarbeit gewählten Bild sieht der Icon folgendermaßen aus (Abbildung 8.3):



Abbildung 8.3.: Icon der App [38]

8.1.1. Überspielen der App auf ein Smartphone

Das Überspielen der App vom Computer auf ein Smartphone wurde direkt über Android Studio vorgenommen. Dazu wurde ein Handy (Xiaomi Mi A2) über ein USB-Kabel mit dem Computer verbunden und der Entwicklermodus am Smartphone wurde aktiviert. Um den Entwicklermodus zu aktivieren wurde in den **Einstellungen** des Smartphones auf den Punkt *System* und dann *Über das Telefon* geklickt. Danach wurde sieben Mal auf das Feld *Build Number* gedrückt. Anschließend erschien eine Nachricht mit dem Text *Sie sind jetzt ein Entwickler*.

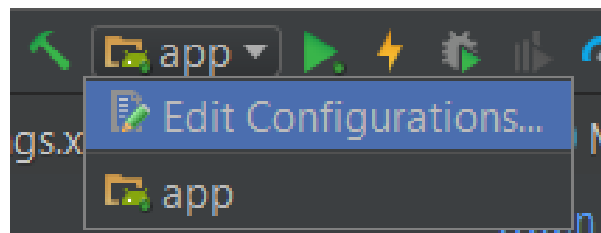


Abbildung 8.4.: Konfiguration

Weiters wurden die Konfigurationen unter *Run/Edit Configurations* (Abbildung 8.4) bearbeitet, sodass bei den *Deployment Target Options* die Option *USB Device* ausgewählt wurde. Anschließend kann man das verbundene Smartphone auswählen. Bei anschließendem Klick auf *OK* wird die Änderung bestätigt und man kann das Projekt ausführen.

Android Studio erstellt bei Klick auf *Run* (grünes Dreieck in Abbildung 8.4) ein Android Package (APK). Dieses Paket-File-Format wird verwendet, um Android Apps verteilen und installieren zu können [39].

Nachdem das APK erzeugt wurde, wird die App am angeschlossenen Smartphone installiert und öffnet sich danach automatisch. Damit kann die Funktion der App am Smartphone überprüft werden.

9. Benutzer Handbuch

Diese Handbuch wurde speziell dafür entwickelt, um Schritt für Schritt die Funktionen und Optionen dieses Geräts zu erklären.

9.1. Einleitung

Herzlichen Glückwunsch zum Erwerb einer Drinkable-Station inklusive Software! Dieses Handbuch hilft Ihnen, sich mit den gelieferten Artikel, sowie der Software, erfolgreich zu bedienen.

9.2. Entsorgungshinweis

Die Drinkable-Station enthält Stoffe, die Umwelt und Gesundheit gefährden können. Sie muss daher unbedingt als Problemstoff entsorgt werden. Auf keinen Fall sollten sie die Station im Restmüll entsorgen. Ebenso kann das Produkt anstelle einer Entsorgung an das Unternehmen mit passender Beilage retourniert werden.

9.3. Richtlinien für einen sicheren und effizienten Verbrauch

- Halten Sie das Gerät von starker Wärmeeinwirkung fern (Herdplatte, Mikrowelle, im heißen Auto vergessen)
- Das Gerät ist nicht Strahlwassergeschützt. Halten Sie es nicht unter den Wasserhahn. Zum putzen wischen Sie es mit einem leicht angefeuchteten Tuch ab.
- Stellen Sie das Gerät beim Gebrauch auf einen Ebenen Untergrund.
- Schützen Sie das Gerät vor Erschütterungen und Fall.
- Platzieren Sie nur die dafür vorgesehenen Objekte auf dem Untersetzer und üben Sie keinen starken, ungleichmäßigen Druck auf die Wiegeplatte des Untersetzers aus.

9.4. Lieferumfang

Die Lieferung beinhaltet einen Untersetzer (Abbildung 9.1), eine Ladestation (Abbildung 9.2), ein Gateway (Abbildung 9.3) und eine Micro SD-Karte mit der Vorinstallierten Software darauf (Abbildung 9.4). Ebenso wie ein LAN-Kabel (Abbildung 9.5) und ein Netzteil für das Gateway (Abbildung 9.6). Auf der mitgelieferten Visitenkarte steht die Seriennummer, die für den Zugang zur Drinkable-Plattform erforderlich ist.



Abbildung 9.1.: Untersetzer



Abbildung 9.2.: Ladestation



Abbildung 9.3.: Gateway



Abbildung 9.4.: Untersetzer



Abbildung 9.5.: Lan-Kabel



Abbildung 9.6.: Netzteil

9.5. Hinweise zum Akku

- Der Akku muss vor dem Aufladen nicht vollständig entladen werden.
- Blinkt das Gerät rot, so sollte der Akku aufgeladen werden.
- Leuchtet das zurzeit geladene Gerät grün, so ist es fertig geladen und kann wie gewohnt weiterverwendet werden.

9.6. Erstmalige Inbetriebnahme

Die erstmalige Inbetriebnahme beginnt mit dem Aufladen des Untersetzers. Dazu muss der Stecker der Ladestation eingesteckt werden und der Untersetzer, wie in Abb. 9.7 gezeigt, auf der Ladestation platziert werden.

Der Untersetzer kann nicht manuell ein bzw. ausgeschaltet werden. Er schaltet sich nur aus, wenn der Akkustand zu gering ist.

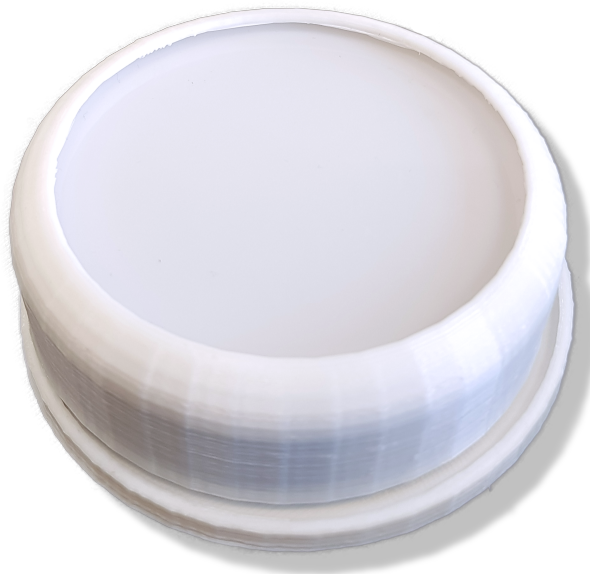


Abbildung 9.7.: Laden des Untersetzters

9.7. Zugang zur Webplattform

Wie bereits erwähnt, ermöglicht Ihnen die im Lieferumfang beinhaltete Seriennummer einen Zugang zur Drinkable-Plattform, in der Sie die aufgenommenen Daten über Ihr Trinkverhalten beobachten können. Um zur Webplattform zu gelangen, müssen Sie „ „ in die URL-Leiste eingeben. Anschließend gelangen Sie zum Anmeldeformular welches in Abbildung 9.8 gezeigt wird. Da Sie bis jetzt allerdings noch nicht als Benutzer eingetragen sind, müssen Sie auf das Registrieren-Feld klicken, welches Sie zum Registrierungsformular führt (Abbildung 9.9).



Abbildung 9.8.: Anmelden

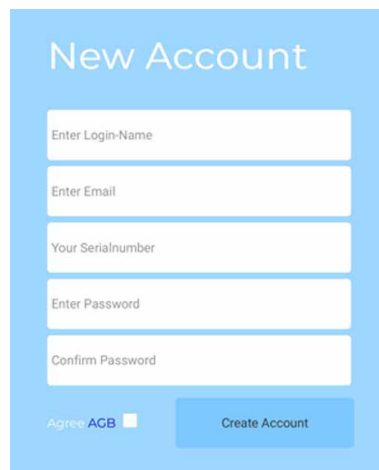


Abbildung 9.9.: Registrieren

Geben Sie nun Ihre Daten, sowie die mitgelieferte Seriennummer in die zugehörigen Felder ein.

9.8. Navigation in der Webplattform

War die Registrierung, bzw. das Anmelden erfolgreich, werden Sie auf die Startseite weitergeleitet (9.10). Hier erkennen Sie auf einen Blick die gesamte Trinkmenge (Abbildung 9.11), die Sie am aktuellen Tag zu sich genommen haben.



Abbildung 9.10.: Startseite



Abbildung 9.11.: Trinkstand

In der rechten oberen Ecke finden Sie ein Menü-Symbol siehe Abbildung 9.12.

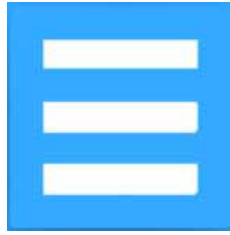


Abbildung 9.12.: Menü Button

Bei Klick darauf erscheinen Ihnen zwei Optionen (Abbildung 9.13). Diese sind entweder *Aktivitäts* - Seite und *Abmelden* oder „Startseite“ und „Abmelden“ – je nachdem, auf welcher Seite Sie sich gerade befinden. Entscheiden Sie sich, auf die Option „Abmelden“ zu klicken, so werden Sie zurück auf das Anmeldeformular geleitet und müssen sich, um wieder zu ihren Daten zu gelangen, erneut anmelden.

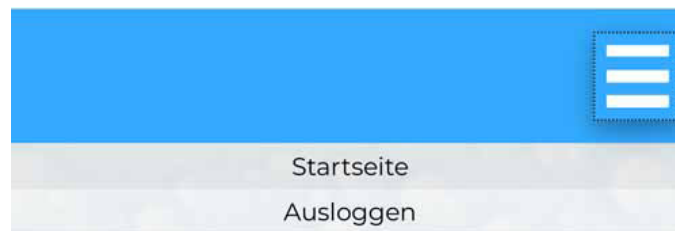


Abbildung 9.13.: Menüleiste

Klicken Sie auf den horizontalen blauen Balken, der quer durch die Mitte der Webseite verläuft, gelangen Sie auf die Aktivitäts-Seite. Hier befindet sich eine Grafik siehe Abbildung 9.14, die alle Trinkmengen zur zu sich genommenen Uhrzeit an einem Tag anzeigt. Über der Grafik finden Sie das Datum des Tages. Bei Aufruf dieser Seite wird immer zuerst der aktuelle Tag angezeigt.

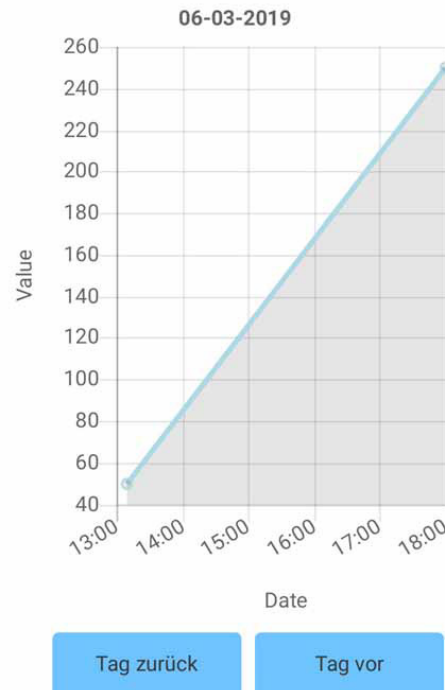


Abbildung 9.14.: Aktivitäten

Unter der Grafik befinden sich zwei Felder. Das linke Feld besitzt die Aufschrift „Tag zurück“ und das rechte Feld „Tag vor“. Damit können Sie nachsehen, wie viel Sie zB. am Vortag getrunken haben und können somit zwischen den Tagen hin und her schalten. Gibt es allerdings am Vortag keinen Eintrag, so wird automatisch der nächste in der Zeit zurückliegende Eintrag angezeigt. Damit kann es nicht passieren, dass Sie bei Nichtnutzung des Untersetzers lange zwischen den einzelnen Einträgen hin- und her blättern müssen.

10. Betriebswirtschaftliche Kalkulation

in Kalkulationen.tex geregelt

10.1. Einzelkosten

Die nachfolgenden beschriebenen Kosten sind direkt dem Sender oder dem Empfänger als Kostenträger zugeordnet.

10.1.1. Fertigungskosten der PCB's

Aufgrund externer Fertigung der PCB's bei JLCPCB.com ergeben sich von der Stückzahl indirekt proportional abhängige Stückkosten. Da weder Sender noch Empfänger individuell für jedes Gesamtsystem einzeln hergestellt werden müssen, ist eine Bestellung in großer Stückzahl möglich. Beim Hersteller liegt die maximale Quantität der Bestellung von PCB's bei 8000 Stück. Im Gegensatz dazu führt die von Herstellerseite definierte Mindestbestellmenge von 5 PCB's zu einer suboptimalen Kapazitätsausnutzung beim Empfänger, da in einem Haushalt mit mehreren Personen weniger Empfänger als Sender benötigt werden.

Kostenträger	Kapazität	Gesamtkosten [€]	Stückkosten [€]
Sender	10	2	0,20
Empfänger	10	2	0,20

Tabelle 10.1.: PCB Kosten

10.1.2. Material- bzw. Bauteilkosten

Das für die Produktion benötigte Material setzt sich aus allen Bauteilen der zugehörigen Hardware zusammen. Zusätzlich dazu kommen noch die Materialkosten für das Gehäuse, sowie die Betriebsstunden des 3D-Druckers.

Kostenträger	Stückkosten [€]
Sender	18,69
Empfänger	35,27

Tabelle 10.2.: Material Kosten

10.1.3. Entwicklungskosten

Zur Berechnung der Entwicklungskosten ist die Definition eines entsprechenden Stundensatzes notwendig.

$$K_{\text{dev}}(t) = 50 \frac{e}{h} * t \quad (10.1)$$

Kostenträger	Stundenzahl	verurs. Kosten [€]
Sender	248	12.400
Empfänger	248	12.400

Tabelle 10.3.: Material Kosten

10.2. Gemeinkosten (Overheads)

10.2.1. Test-, Bestückungs- und Kalibrierarbeiter

Sowohl die PCB's des Senders, als auch des Empfängers, liegen durch JLCP-CB.com unbestückt vor. Die notwendige Bestückung erfolgt in beiden Fällen manuell durch einen eigens angestellten Arbeiter. Neben dieser Tätigkeit fällt diesem auch die Durchführung einfacher Inspektionen aller PCB's und die Kalibrierung der Sender und die Programmierung zu. Als Berechnungsgrundlage dient das im Kollektivvertrag für die Elektro- und Elektronikindustrie verankerte Mindestgehalt eines Arbeiters der Beschäftigungsgruppe A der Grundstufe A. Unter Miteinbeziehung der Lohnnebenkosten ergeben sich die monatlichen Kosten:

$$\frac{K_{\text{Arb}}}{M} = 1.829,42 \frac{\text{€}}{m} \quad (10.2)$$

Mittels Normalwochenarbeitszeit von 38,5h lässt sich daraus der Stundenlohn berechnen.

$$\frac{K_{\text{Arb}}}{h} = \frac{1.829,42}{38,5 \frac{h}{w} * \frac{52w}{12m}} = 10.96 \frac{\text{€}}{h} \quad (10.3)$$

Damit die Kapazitätsgrenzen bestmöglich ausgeschöpft werden, besteht ein befristetes Dienstverhältnis. Dieses währt lediglich für die Dauer der Produktion einer gewissen Stückzahl an Sender und Empfänger.

Tatsächliche Kapazitätsgrenzen

Der Arbeiter benötigt für den Bestückungs-, bzw. Kalibrierprozess einen gewissen Zeitrahmen. Hochgerechnet ergeben sich daraus die monatlichen Kapazitätsgrenzen.

Kostentäger	Druchgeführte Tätigkeit	Zeit	Stk. / Woche
Sender	Best. , Pro. und Kalibr.	1h 30min.	30Stk.
Empfänger	Bestückung und Pro.	30min.	128Stk.

Tabelle 10.4.: Tatsächliche Kapazitätsgrenze

Anschaffungs- und Betriebskosten

Für die Entwicklung fallen monatliche Kosten an. Diese sind in folgender Tabelle aufgelistet:

Kosten	Menge	Einheit
Bürofläche	35	m ²
monatliche Miete	8,92	$\frac{e}{m^2}$
Miete	312,31	$\frac{e}{Monat}$
Heizung	26,92	$\frac{e}{Monat}$
Wasserpreis	1,54	$\frac{e}{m^3}$
Wasserverbrauch	2	$\frac{m^3}{Monat}$
Wasserkosten	3,08	$\frac{e}{Monat}$
Stromkosten	25	$\frac{e}{Monat}$
Internetkosten	28,33	$\frac{e}{Monat}$
Gesamtkosten	395,64	$\frac{e}{Monat}$
Dauer	9	Monate
Betriebskosten Gesamt	3.560,76	€

Tabelle 10.5.: Betriebskosten

Zur Produktion sind einmalig die nachstehenden Test- und Messgeräte anzuschaffen sowie Lizenzgebühren zu entrichten:

Lizenz/Gerät	Kosten [€]
Altium Designer Stand-alone Lizenz	3.995
Multimeter	23
Labornetzgerät	80
Lötstation	250
Anschaffungskosten Gesamt	4.348

Tabelle 10.6.: Geräte Kosten

Aufschlüsselung

Da pro Gesamtsystem eine deutlich höhere Anzahl an Sendern produziert wird, entfällt ein Übergewicht an Anschaffungs- und Betriebskosten auf diesen Kostenträger. Es wird nachfolgende Aufschlüsselung angewandt:

Kostenträger	Schlüssel [%]
Sender	30
Empfänger	60

Tabelle 10.7.: Verteilungsschlüssel

10.3. Amortisationsrechnung - Sender

Variable Kosten

Die variablen Kosten des Senders bestehen aus den Material- und Fertigungskosten.

Kostenart	Betrag [€]
Materialkosten	18,69
Fertigungs-EK	0,20
Fertigungslohn	10,96
Summe	29,85

Tabelle 10.8.: Variable Kosten des Senders

Gewinnzuschlag

Um einen anschaulichen Verkaufspreis zu erhalten, werden im Falle der Sender 67% der variablen Kosten aufgeschlagen. Damit ergibt sich der Netto-Verkaufspreis.

$$P_S = 29,85\text{€} * 167\% = 49,99\text{€} \quad (10.4)$$

Fix Kosten

Als fix sind jene Kosten zu verstehen, die sich bei Produktion innerhalb der Kapazitätsgrenzen des Senders nicht ändern. Zu diesen gehören die Entwicklungskosten und auch die aufzuschlüsselnden Betriebs- und Anschaffungskosten.

Kostenart	Betrag [€]
Betriebskosten	$3.560,76 * 30\% = 1.068,22$
Anschaffungskosten	$4.348 * 30\% = 1.304,40$
Entwicklungskosten	12.400
Summe	17.265,16

Tabelle 10.9.: Fix Kosten des Senders

Deckungsbeitrag

Die Differenz zwischen dem Verkaufspreis und den variablen Kosten ist der Deckungsbeitrag.

$$DB_S = P_S - V_S = 49,99\text{€} - 29,85\text{€} = 20,14\text{€} \quad (10.5)$$

Break-Even Point des Senders

Mithilfe des Deckungsbeitrages lässt sich diejenige Stückzahl ermitteln, bei der die Kosten der umgesetzten Produkte durch die Erlöse aufgewogen werden.

$$BEP_S = \frac{Fixkosten}{DB_S} = \frac{17.265,16}{20,14} = 858 \text{ Stk.} \quad (10.6)$$

Erst nach Erreichen dieser Umsatzgrenze ergibt sich überhaupt ein positiv erwirtschafteter Gewinn. In Abbildung 10.1 werden die gezeigten Zusammenhänge als Graphen der zugehörigen Kosten- und Erlösfunktionen dargestellt.

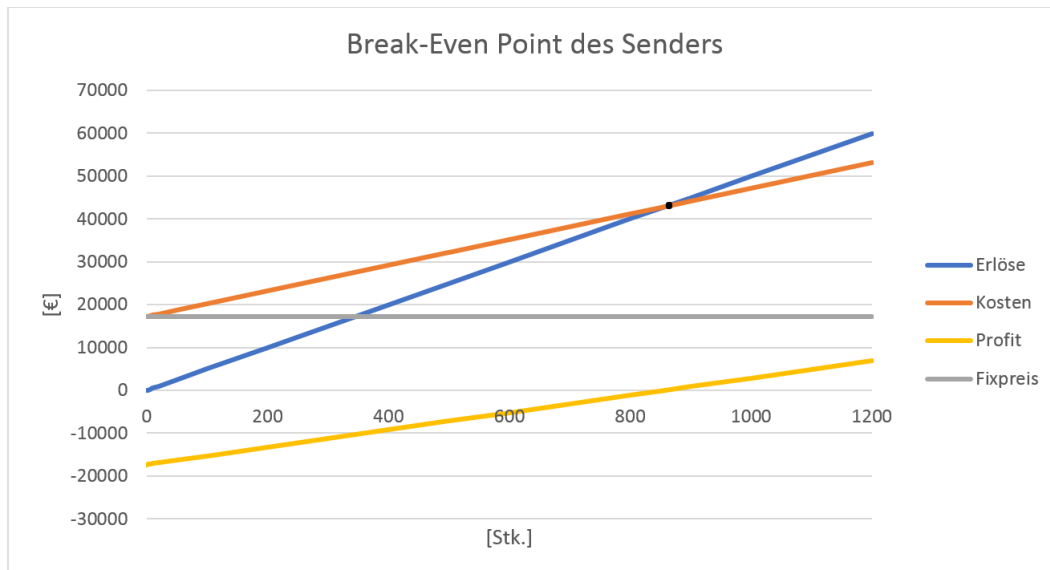


Abbildung 10.1.: Break Even Point des Senders

10.4. Amortisationsrechnung - Empfänger

Variable Kosten

Die variablen Kosten des Empfängers bestehen wie beim Sender aus den Material- und Fertigungskosten.

Kostenart	Betrag [€]
Materialkosten	35,27
Fertigungs-EK	0,20
Fertigungslohn	10,96
Summe	46,43

Tabelle 10.10.: Variable Kosten des Empfängers

Gewinnzuschlag

Um einen anschaulichen Verkaufspreis zu erhalten, werden im Falle der Empfängers 115% der variablen Kosten aufgeschlagen. Damit ergibt sich der Netto-Verkaufspreis.

$$P_S = 46,43\text{€} * 215\% = 99,99\text{€} \quad (10.7)$$

Fix Kosten

Als fix sind jene Kosten zu verstehen, die sich bei Produktion innerhalb der Kapazitätsgrenzen des Empfängers nicht ändern. Zu diesen gehören die Entwicklungskosten und auch die aufzuschlüsselnden Betriebs- und Anschaffungskosten.

Kostenart	Betrag [€]
Betriebskosten	$3.560,76 * 70\% = 2.492,53$
Anschaffungskosten	$4.348 * 70\% = 3.043,60$
Entwicklungskosten	12.400
Summe	17.936,13

Tabelle 10.11.: Fix Kosten des Empfängers

Deckungsbeitrag

Die Differenz zwischen dem Verkaufspreis und den variablen Kosten ist der Deckungsbeitrag.

$$DB_E = P_E - V_E = 99,99\text{€} - 46,43\text{€} = 53,56\text{€} \quad (10.8)$$

Break-Even Point des Empfängers

Mithilfe des Deckungsbeitrages lässt sich diejenige Stückzahl ermitteln, bei der die Kosten der umgesetzten Produkte durch die Erlöse aufgewogen werden.

$$BEP_E = \frac{Fixkosten}{DB_E} = \frac{17.936,13}{53,56} = 335 \text{ Stk.} \quad (10.9)$$

Erst nach Erreichen dieser Umsatzgrenze ergibt sich überhaupt ein positiv erwirtschafteter Gewinn. In Abbildung 10.1 werden die gezeigten Zusammenhänge als Graphen der zugehörigen Kosten- und Erlösfunktionen dargestellt.

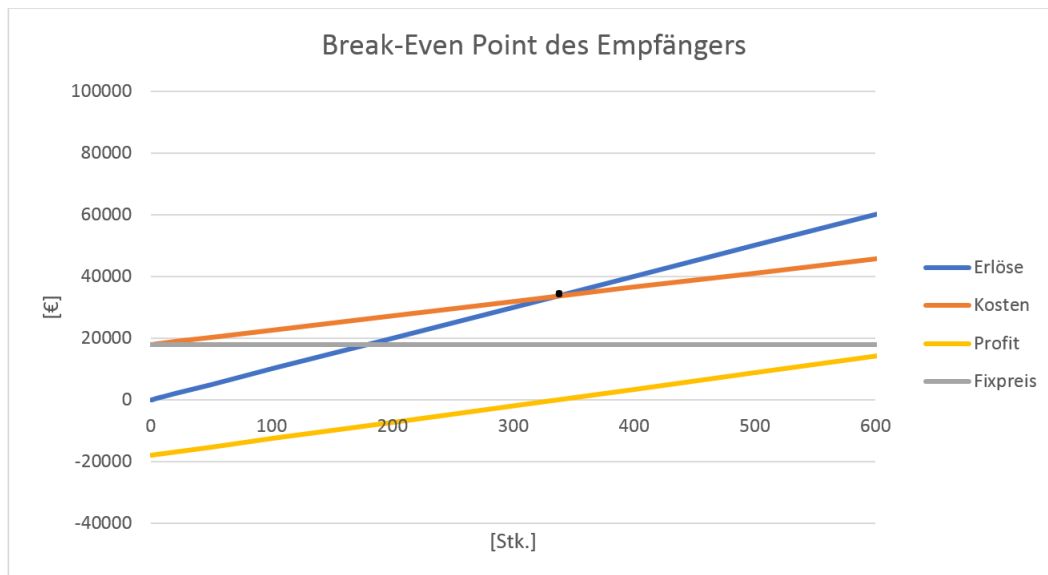


Abbildung 10.2.: Break-Even Point des Empfängers

10.5. Interpretation

Für das entwickelte Produkt besteht kein direktes Vergleichsprodukt, allerdings existiert ein Produkt, welches einen ähnlichen Zweck aufweist: der *Ozmo Smart Cup* [40]. Dabei handelt es sich um eine Trinkflasche, welche die Trinkdaten trackt und IOS-kompatibel ist. Ebenso kann es mit einer FitBit-Smartwatch oder einer AppleWatch verbunden werden. Das Senden der Daten erfolgt mittels Bluetooth. Die Flasche kann nur händisch gewaschen werden, da sich die gesamte Technik darin befindet.

Das im Rahmen der Diplomarbeit entwickelte System verwendet einen Untersetzer, um die Trinkdaten aufzunehmen. Das ermöglicht im Gegensatz zum Ozmo Smart Cup das verwenden diverse Getränkebehälter, insofern diese einen Durchmesser kleiner als 10cm besitzen. Somit kann sowohl aus einem Glas, einem Becher oder auch aus einer Flasche getrunken werden, welche sowohl händisch als auch maschinell gewaschen werden kann. Die Trinkdaten des Drinkable-Systems können sowohl in einer Android-App, als auch auf einer Webseite (Mobil oder Desktopansicht) angezeigt werden und sind somit für mehrere Betriebssystemen kompatibel.

Die Standardvariante Ozmo Smart Cups *Ozmo Active* beginnt bei 69.99USD (ca. 62€) und besitzt ein Fassungsvermögen von 473ml. Die Variante der Thermo-Flasche *Ozmo Java+* kostet 99.99 USD (88,42€) und besitzt dasselbe Fassungsvermögen wie die Standardvariante.

Das Drinkable-System kann Getränkebehälter mit einem Fassungsvermögen von etwa 1.5l erfassen und das Starterpaket (ein Untersetzer mit Aufladestation inkl. Gateway, LAN-Kabel und Netzteil) kostet 149.98€. Hierbei kostet der Untersetzer mit Aufladestation 49.99€. Das Gateway inkl. Verkabelung kommt auf 99.99€. Weiters ist hinzuzufügen, dass ein Haushalt nur ein Gateway benötigt und pro Benutzer wird ein Untersetzer benötigt. Kauft man also für eine Familie von 4 Personen ein Drinkable System, beträgt der Preis 299.95€.

Vier Ozmo Smart Cups kommen auf einen Betrag von ca. 248€. Umso mehr Benutzer das Drinkable-System verwenden, desto eher zahlt es sich gegenüber des Ozmo-Produkts aus. Das Entwickelte System lässt sich ebenfalls für ein Altersheim, Wohnheim oder andere Pflege-Einrichtungen anwenden - allerdings müsste dazu die Software verändert werden, was die Entwicklungskosten beeinflusst.

Ein Vorteil an Ozmo Active - Flasche ist die einfache Handhabbarkeit unterwegs oder beim Sport. Allerdings punktet das Drinkable-System in Haushalten oder Büros, da man sein Getränk wie gewohnt in ein Glas oder einen Becher, sowie ein Häferl oder sonstiges Gefäß füllen und daraus trinken kann. Muss das die Flasche von Ozmo gewaschen werden, so würde man einen Ersatz zum Auszeichnen der derweiligen Trinkmengen benötigen - bei dem Drinkable-System muss lediglich ein anderes Glas verwendet werden.

Ozmo Smart Cup ist das einzige System, welches vom Preis/Leistungsverhältnis ansatzweise an diese Diplomarbeit herankommt. Sonst befinden sich derzeit keine anderen Systeme am Markt mit dem Verglichen werden kann.

A. Aufwand

Auf den folgenden Tabellen wird aufgeschlüsselt wie viel Aufwand getrieben wurde. Die Arbeitsbeschreibung ist nach Kalenderwochen aufgeschlüsselt, die aufgewendete Zeit nach Monaten.

Die Gesamtarbeitszeit beträgt:

Pia Wenty	248h
Nina Nenning	248h

KW	Arbeitsinhalt
	September (insg. 20h)
36	Projektmanagement
38	Grundlegende Konzeptüberlegung
	Oktober (insg. 15h)
40	Umbau der Waage - HX711 (ADC) angeschlossen
41	LoRa-Modul Sender installieren
42	Sendemessungen durchgeführt
	November (insg. 50h)
45	Print Design aus Mechanischer sicht
46	Print Überarbeitung
47	Print Fertigstellung
	Dezember (insg. 65h)
48	Entwicklung des Sendalgorithmus
49	EEPROM auslesen und Seriennummer speichern
50	Send Software Fixes
	Jänner (insg. 48h)
01	Gehäuse Design des Sender
04	Gehäuse Design des Empfängers
	Februar (insg. 30h)
05	PCB Entwicklung des Empfängers
06	Zusammenbau
07	Erste Test gestartet
	März (insg. 20h)
09	Letzte Überarbeitung des Sendalgorithmuses
10	Dokumentation
11	Letzte Tests beendet
12	Dokumentations Überarbeitung
13	Dokumentations Fertigstellung

Tabelle A.1.: Aufwandschlüssel Pia Wenty

KW	Arbeitsinhalt
	Sebtember (insg. 20h)
36	Projektmanagement
38	Grundlegende Konzeptentwicklung
	Oktober (insg. 30h)
40	Aufsetzen des Empfängers (Raspberry Pi inkl. LoRa-Modul)
41	Raspberry Pi als Server inkl. Datenbank aufsetzen
42	Empfangen und Verarbeiten der Wiegedaten
	November (insg. 50h)
45	Der Empfangener Wert kann in eine Testdatenbank gespeichert werden
46	Erstellen einer Webapplikation mit responsivem Design und Formularen
47	Am Raspberry empfangene Daten splitten um diese korrekt in die Datenbank speichern
	Dezember (insg. 65h)
48	Verknüpfung der Webapplikation mit der Datenbank, sowie Auswertung und Anzeige der Daten in der Webapplikation
49	Umbau der Webapplikation zur App
	Jänner (insg. 25h)
03	Fertigstellen der progressiven Webapp
04	Entwicklung einer Funktion zum Blättern der angezeigten Tage im Diagramm
	Februar (insg. 25h)
06	Testen und Ändern der Blätter-Funktion
07	Erstellen einer weiteren Datenbanktabelle <i>Seriennummer</i>
	März (insg. 38h)
09	Einbinden der Tabelle <i>Seriennummer</i> und Erstellen einer Funktion zur Abfrage zur Überprüfung der Seriennummer
10	Durchführung von Tests bezüglich der Funktionalität der Webapplikation als auch der Webapp
11	Finale Änderungen in der Webapp (hinzufügen einer Beispiel-AGB, Text auf Buttons ändern, etc.)
12	Überarbeiten und Fertigstellen der Diplomarbeitdokumentation

Tabelle A.2.: Aufwandschlüssel Nina Nenning

B. Betreuungsprotokolle gemäß §9 bs. 2 PrO

Betreuungsprotokoll zur Diplomarbeit

lfd. Nr.: 01

Themenstellung: Füllstandsevaluierung geschlossener Getränkebehälter
 Kandidaten/Kandidatinnen: Nina Nenning / Pia Wenty

Jahrgang: 5BHELS
 Betreuer/in: Steindl
 Ort: St. Pölten
 Datum: 18.10.2018
 Zeit: 12:20 – 12:50

Besprechungsinhalt:

Name	Notiz
Nina Nenning	Besprechung des Projektplans, sowie folgender Projektschritte: • Empfangen der Daten mit LoRa und Raspberry Pi • Speichern der Daten in eine Datenbank
Pia Wenty	Besprechung des Projektplans, sowie folgender Projektschritte: • Wiegen und Auswerten mit der Wägezelle • Senden der Daten mit LoRa

Aufgaben:

Name	Notiz	zu erledigen bis
Nina Nenning	Projektplan überarbeiten: Dokumentation mit einplanen, konkreter Formulierungen	24.10.2018
Pia Wenty	Projektplan überarbeiten: Dokumentation mit einplanen, konkreter Formulierungen	24.10.2018

Betreuungsprotokoll zur Diplomarbeit

lfd. Nr.: 02

Themenstellung: Füllstandsevaluierung geschlossener Getränkebehälter
 Kandidaten/Kandidatinnen: Nina Nenning / Pia Wenty

Jahrgang: 5BHELS
 Betreuer/in: Steindl
 Ort: St. Pölten
 Datum: 10.12.2018
 Zeit: 13:10 – 14:00

Besprechungsinhalt:

Name	Notiz
Nina Nenning	Besprechung des Projektplans, sowie folgender Projektschritte: • Webapplikation sowie App und deren Funktion • Verarbeitung der Messdaten sowie deren Anzeige in einem Graphen
Pia Wenty	Besprechung des Projektplans, sowie folgender Projektschritte: • Daten Erfassung, Auswertung und Übertragung mit Hilfe des Mikrocontrollers • Akkumanagement und Anzeige des Ladestatus • Gehäusedesign Unterstützung durch Funktionsmuster

Aufgaben:

Name	Notiz	zu erledigen bis
Nina Nenning	• Einstellen eines bestimmten Datums für die Trinkanzeige zur Benutzerfreundlichkeit • Summe der über den Tag verteilten Trinkmengen bilden, um sie auf dem Startbildschirm anzeigen zu können.	02.01.2019
Pia Wenty	• Anzeige des Akkustandes am Gehäuse implementieren sowie eine generelle Überarbeitung des Gehäuses • Gehäuse sowie PCB-Design für den Empfänger starten	02.01.2019

Betreuungsprotokoll zur Diplomarbeit

lfd. Nr.:03

Themenstellung:	Füllstandsevaluierung geschlossener Getränkebehälter
Kandidaten/Kandidatinnen:	Nina Nenning/ Pia Wenty
Jahrgang:	5BHELS
Betreuer/in:	Steindl
Ort:	HTL St.Pölten
Datum:	18.03.2019
Zeit:	13:10 – 14:00

Besprechungsinhalt:

Name	Notiz
Pia Wenty	Dokumentation: Sender: • Daten Erfassung • Akkumanagement • PCB Design • Gehäusedesign • Empfänger (Hardware)
Nina Nenning	Dokumentation: • Progressive Webapp • Webapplikation • Empfänger (Software)

Aufgaben:

Name	Notiz	zu erledigen bis
Pia Wenty	• Ausführliches Dokumentieren • Ausbesserungsarbeiten	27.03.2019
Nina Nenning	• Ausführliches Dokumentieren • Ausbesserungsarbeiten	27.03.2019

C. Abkürzungsverzeichnis

ADC	Analog Digital Converter
AES	Advanced Encryption Standard
CCCV	Constant current constant Voltage
CRC	Cyclic Redundancy Check
CSS	Cascading Style Sheets
CSSM	Chirp Spread Spectrum Modulation
DMS	Dehnungsmessstreifen
HF	Hoch Frequenz
HTML	Hypertext Markup Language
JS	JavaScript
LiPo	Lithium Polymer Akkumulator
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPWAN	Low Power Wide Area Network
mcd	Lichtstärke in Candela
NF	Nieder Frequenz
PM	Projektmanager leitet das Projekt und ist dafür verantwortlich
PTM	Projektteammitglied ist ein Mitarbeiter des Projektes
RTC	Real Time Clock
SPI	Serial Peripheral Interface
WCM	Wireless Charging Modul

D. Abbildungsverzeichnis

1.1. Aufteilung der Arbeitsbereiche	2
3.1. L ^A T _E X Logo [1]	7
3.2. Overleaf Logo [2]	7
3.3. Altium Logo [3]	8
3.4. Inventor Logo [4]	8
3.5. Arduino Logo [5]	8
3.6. Raspberry Pi Logo [6]	9
3.7. LoRa Logo [7]	9
4.1. LoRa Netzwerkarchitektur	13
4.2. Pinbelegung	14
4.3. Bemaßung	14
4.4. Up- und Down-Chirp's	15
4.5. LoRa-Paket Skizze	15
4.6. LoRa-Nachricht	16
4.7. LoRa-Protokollstack	16
4.8. Ra02-Modul Sende Frequenz	18
4.9. Sendezeit Messung	19
4.10. Spektrogramm	20
4.11. Nachrichten Spektrogramm	21
5.1. Wägezelleklassen [19]	24
5.2. Wägezelle	25
5.3. LoadCell	25
5.4. Block Diagramm DS1307	26
5.5. Verwendeter Akku	27
5.6. Schutzschaltung	28
5.7. Integrierte Schutzschaltung	28
5.8. CCCV Skizze	29
5.9. Packing Typs	30
5.10. Energieübertragung mittels Induktionsspulen [25]	32
5.11. Enge Kopplung bei induktivem Leistungstransfer	33
5.12. Kabelloses Laden Darstellung	33
5.13. Blockschaltbild	35
5.14. HX711 Grundschtaltung	36

5.15. RTC Grundschtung	37
5.16. Ra02-Modul Schaltung	37
5.17. Akku Management Grundschtung	38
5.18. Relative Lichtstärke der LED zur Lichtstärke bei 20mA	39
5.19. Gesamtschaltung	40
5.20. Querschnitt des PCB's	41
5.21. Ergebnis des Routings des Senders - Top-Layer	43
5.22. Ergebnis des Routings des Senders - Bottom-Layer	43
5.23. Herzstück des Untersetzers	45
5.24. Lade Modul	45
5.25. Rahmen des Untersetzers	46
5.26. Rahmen des Bodens	46
5.27. Power	47
5.28. Ladet	47
5.29. Fertig geladen	47
5.30. Bitte Laden	47
5.31. Ladestation oben	48
5.32. Ladestation unten	48
5.33. Ladestation real	49
5.34. Untersetzer auf der Ladestation	49
5.35. PLA Herstellung auf chemischer Basis [28]	50
5.36. Sender Softwarestruktur	51
5.37. Akkuüberwachungsprogrammstruktur	55
5.38. Random Address Read	57
5.39. Byte write	58
5.40. Datenflussstruktur	59
6.1. Verkabelung Raspberry Pi - Ra02-Modul	64
6.2. Aktuelle Raspberry Pi - Lora Konfiguration	65
6.3. Datenbankstruktur	68
6.4. Empfänger Schaltung	72
6.5. Top-Layer	73
6.6. Bottom-Layer	73
6.7. Gateway Gehäuse	74
6.8. Gateway-Gehäusedeckel	75
6.9. Gateway inkl. Gehäuse	76
7.1. Überblick über den Aufbau	79
7.2. Startseite	80
7.3. Aufbau der Webapplikation	81
7.4. Anmeldeformular	83
7.5. Registrierungsformular	84

7.6. Menü	85
7.7. Auszulesende Daten	85
7.8. Buttons zum Umschalten des anzuzeigenden Tages	88
8.1. Erstellen des Webview-Feldes	93
8.2. Image Asset	94
8.3. Icon der App [38]	94
8.4. Konfiguration	95
9.1. Untersetzter	98
9.2. Ladestation	98
9.3. Gateway	98
9.4. Untersetzter	98
9.5. Lan-Kabel	98
9.6. Netzteil	98
9.7. Laden des Untersetzters	99
9.8. Anmelden	100
9.9. Registrieren	100
9.10. Startseite	101
9.11. Trinkstand	101
9.12. Menü Button	102
9.13. Menüleiste	102
9.14. Aktivitäten	103
10.1. Break Even Point des Senders	110
10.2. Break-Even Point des Empfängers	112

E. Listings

LoRa Konfig	52
Packet	53
Energiesparmodus	53
Kalibrierung	56
EEPROM	57
6.1. Erstmaliges Einloggen	62
6.2. IP-Adressen Konfiguration	63
6.3. Umwandlung von ASCII in Text	67
6.4. Datenbankverbindung	70
6.5. Datensatz hinzufügen Raspberry Pi	71
6.6. Beenden der Datenbankverbindung	71
7.1. Bestimmung des aktuellen Tages	86
7.2. Erstellen des Diagramms	86
7.3. AJAX Request	89
8.1. Internetzugriffserlaubnis	91
8.2. Aktivieren der Zurück-Taste	92
8.3. Zuweisen der URL	92

F. Tabellenverzeichnis

1.1. Meilensteine	2
1.2. Projektplan Legende	3
1.3. Projektplan	4
4.1. LoRaWAN Spezifikationen	12
4.2. Ra02-Modul Spezifikationen und Gewählte Werte	14
4.3. Rate Parameter	17
4.4. Time Parameter	17
4.5. Spektrumsanalysator Einstellungen	18
5.1. Ladestrom Einstellungen	31
5.2. Akku Modus	31
5.3. Übertragener Strom mit wachsenden Abstand	34
5.4. Bill of Materials	42
6.1. Bill of Materials	73
10.1. PCB Kosten	105
10.2. Material Kosten	106
10.3. Material Kosten	106
10.4. Tatsächliche Kapazitätsgrenze	107
10.5. Betriebskosten	107
10.6. Geräte Kosten	108
10.7. Verteilungsschlüssel	108
10.8. Variable Kosten des Senders	109
10.9. Fix Kosten des Senders	109
10.10. Variable Kosten des Empfängers	111
10.11. Fix Kosten des Empfängers	111
A.1. Aufwandschlüssel Pia Wenty	116
A.2. Aufwandschlüssel Nina Nenning	117

G. Danksagungen

Wir bedanken uns bei Dipl.-Ing. Dr. Reinhard Steindl, dass er uns mit Rat und Tat zur Seite stand.

Weiters bedanken wir uns bei Ing. Marc Prantl, der die Aufgabe als Zweit-Betreulehrer übernahm und uns sowohl Inspirationen zur Umsetzung als auch gute Ratschläge gab.

Ein Großer Dank geht ebenfalls an all unsere Mitschüler und Mitschülerinnen, mit denen wir uns stets beratschlagen konnten.

H. Literaturverzeichnis

- [1] Google Inc. *Latex Logo*. 2018. URL: https://upload.wikimedia.org/wikipedia/commons/9/92/LaTeX_logo.svg/ (besucht am 09.10.2018).
- [2] Google Inc. *Overleaf Logo*. 2018. URL: <http://soapboxscience.org/wp-content/uploads/2017/03/Overleaf-logo-300dpi.png> (besucht am 09.10.2018).
- [3] Google Inc. *Altium Logo*. 2018. URL: <https://www.element14.com/community/servlet/JiveServlet/showImage/2-160994-230077/17-15+Altium+AD+logo.png> (besucht am 09.10.2018).
- [4] Google Inc. *Autodesk Inventor Logo*. 2018. URL: <http://sphsengineering.com/wp-content/uploads/2018/04/Inventor-e1523957011698.png> (besucht am 09.10.2018).
- [5] Google Inc. *Arduino Logo*. 2018. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/8/87/Arduino_Logo.svg/720px-Arduino_Logo.svg.png (besucht am 09.10.2018).
- [6] Google Inc. *Raspberry Pi Logo*. 2018. URL: <https://coderdojokilkenny.files.wordpress.com/2013/10/raspberry-pi-logout1.png> (besucht am 09.10.2018).
- [7] Google Inc. *LoRa Logo*. 2018. URL: https://at.farnell.com/wcsstore/ExtendedSitesCatalogAssetStore/cms/asset/images/common/campaign/internet_of_things/lora-logo.jpg (besucht am 09.10.2018).
- [8] Marc Gutt. *ASE Encry.ption*. 2017. URL: <http://www.aesencryption.de/> (besucht am 23.03.2019).
- [9] Reimo Schaupp. *Sicherheit in LoRaWAN Anwendungen*. 2017. URL: <https://smartmakers.io/de/lorawan-sicherheit/> (besucht am 23.03.2019).
- [10] OEVS SV. *Bandplaene Oesterreich*. 2019. URL: https://www.oevsv.at/export/shared/.content/.galleries/Downloads_Referate/UKW-Referat-Downloads/UKW-Bandplan.pdf (besucht am 09.01.2019).

- [11] 2009 Microchip Technology Inc. *MCP73833 / IC datasheet*. URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/22005b.pdf> (besucht am 19.11.2018).
- [12] Semiconductor Components Industries. *HX711 / IC datasheet*. URL: https://www.mouser.com/ds/2/813/hx711_english-1022875.pdf (besucht am 04.11.2018).
- [13] Inc Maxim Integrated Products. *DS1307 / IC datasheet*. 2015. URL: <https://datasheets.maximintegrated.com/en/ds/DS1307.pdf> (besucht am 30.11.2018).
- [14] RICOH. *Li-Ion BATTERY PROTECTOR R5421N / IC Datasheet*. 1999. URL: <http://electro-shema.ru/files/datasheets/r5421n.pdf> (besucht am 01.01.2019).
- [15] AVIA SEMICONDUCTOR. *FDS6890A Dual N-Channel MOSFET / Datasheet*. 2017. URL: <https://www.onsemi.com/pub/Collateral/FDS6890A-D.PDF> (besucht am 01.01.2019).
- [16] Fairchild Semiconductor Corporation. *S8550 / Datasheet*. 2002. URL: <http://www.e-ele.net/DataSheet/S8550D.pdf> (besucht am 30.11.2018).
- [17] Microchip Technology. *EEPROM / Datasheet*. 2012. URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/20001203U.pdf> (besucht am 30.10.2018).
- [18] Released under the Creative Commons Attribution Share-Alike 2.5 License. *Arduino Nano / Datasheet*. 2008. URL: <https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf> (besucht am 07.11.2018).
- [19] HBM Österreich. *Wägezellenklassen einteilung*. 2017. URL: <https://www.hbm.com/de/2637/oiml-accuracy-classes-explained/> (besucht am 09.01.2019).
- [20] Wikipedia. *Dehnungsmessstreifen*. 2018. URL: <https://de.wikipedia.org/wiki/Dehnungsmessstreifen> (besucht am 21.10.2018).
- [21] DATACOM Buchverlag GmbH. *CCCV-Ladeverfahren*. 2019. URL: <https://www.itwissen.info/CCCV-constant-current-constant-voltage-IU-Ladeverfahren.html> (besucht am 21.10.2018).
- [22] ENGINEERED IN NYC Adafruit. *Lithium Polymer Akku*. 2019. URL: <https://www.adafruit.com/product/2011> (besucht am 21.10.2018).
- [23] ENGINEERED IN NYC Adafruit. *Wireless Charging Modul*. 2019. URL: <https://www.adafruit.com/product/1407> (besucht am 03.03.2019).

-
- [24] Marco Zahner. *Energieeffizienz und EMV-Immissionen von integrierten Induktionsladestationen*. 2017. URL: https://www.emf.ethz.ch/fileadmin/redaktion/public/downloads/3_angebot/wissensvermittlung/studien_fachartikel/Wireless_Charging.pdf (besucht am 23.03.2019).
 - [25] Wikipedia. *Energieübertragung mittels Induktionsspulen*. 2017. URL: https://upload.wikimedia.org/wikipedia/commons/thumb/b/b5/Wireless_power_system_-_inductive_coupling.svg/2000px-Wireless_power_system_-_inductive_coupling.svg.png (besucht am 23.03.2019).
 - [26] Allen Lu. *LED Datenblatt*. 2002. URL: <https://docs-emea.rs-online.com/webdocs/032f/0900766b8032f180.pdf> (besucht am 09.01.2019).
 - [27] 2018 JLCPCB.COM. *PCB Routing Regeln*. 2019. URL: <https://jlcpcb.com/capabilities/Capabilities> (besucht am 09.01.2019).
 - [28] Prof. Dr. Hans-Peter Wiesmann. *WIKI Werkstoffwissenschaft / POLY-MILCHSÄURE/PLA*. 2012. URL: <http://www.e-lms.de/biowiki/artikel.php?id=84&lan=1> (besucht am 24.03.2019).
 - [29] 2015 Semtech Corporation. *LoRa Modulation Basics*. 2018. URL: <https://www.semtech.com/uploads/documents/an1200.22.pdf> (besucht am 09.03.2019).
 - [30] Raspberry Pi Foundation. *Installing Python packages*. URL: <https://www.raspberrypi.org/documentation/linux/software/python.md> (besucht am 20.03.2019).
 - [31] Rui Silva. *pyLoRa 0.2.6*. URL: <https://pypi.org/project/pyLoRa/> (besucht am 20.03.2019).
 - [32] Python Software Foundation. *1.2. Environment variables*. URL: <https://docs.python.org/3/using/cmdline.html#environment-variables> (besucht am 20.03.2019).
 - [33] tutorialsRaspberryPi. *Raspberry Pi Webserver Installation Teil 1 – Apache2*. URL: <https://tutorials-raspberrypi.de/webserver-installation-apache2/> (besucht am 20.03.2019).
 - [34] Pi My Life Up. *Raspberry Pi MySQL & PHPMyAdmin Tutorial*. URL: <https://pimylifeup.com/raspberry-pi-mysql-phpmyadmin/> (besucht am 20.03.2019).
 - [35] chart.js. *chart.js*. URL: <https://www.chartjs.org/docs/latest/> (besucht am 19.03.2019).

- [36] Google Developers. *Building web apps in WebView*. URL: <https://developer.android.com/guide/webapps/webview> (besucht am 20.03.2019).
- [37] Google Developers. *Create app icons with Image Asset Studio*. URL: <https://developer.android.com/studio/write/image-asset-studio> (besucht am 20.03.2019).
- [38] EmojiTerra.com. *Emoji: Tropfen*. URL: <https://emojitera.com/de/tropfen/> (besucht am 20.03.2019).
- [39] Wikipedia. *APK*. URL: https://en.wikipedia.org/wiki/Android_application_package (besucht am 27.03.2019).
- [40] PBC 2019 Kickstarter. *Kickstarter Smart Cup*. 2019. URL: <https://www.kickstarter.com/projects/ozmo/ozmo-smart-cup-water-hydration-and-coffee-enjoymen/> (besucht am 05.03.2019).