

SS2018 Embedded Systems

LoRa Messnetzwerk

Sinan Ayhan, Nick Dirlein, Holger Herbersagen
Marcel Hizli, René Schießwohl

13. August 2018

Inhaltsverzeichnis

1 Projektvorstellung	1
2 Ausgangssituation	2
3 Einkaufsliste	3
4 Überblick der Bauteile	4
4.1 Gateway	4
4.2 Wägezelle	5
4.3 Gewichtssensor	6
4.4 LoRa Node	7
5 Projektplan	8
6 Komponentenmodell	9
7 Gewichtsmessung mit einer Wägezelle	10
7.1 Funktionsweise der Wägezelle	10
7.2 Versuchsaufbau zur Gewichtsmessung	13
7.3 Programmlogik unseres Microchips	15
7.3.1 Vereinfachte Funktionsweise für momentane Gewichtsmessung .	15
7.3.2 Algorithmus für Rausch- und Driftunterdrückung	15
7.3.3 Algorithmus für Gewichtssendung mit dem LoRa-Modul	16
7.4 Demonstration der Gewichtsmessung	16
7.5 Use-Cases von Straight bar Wägezellen	16
7.6 Alternative Bauformen von Wägezellen	17
7.6.1 Disk-Wägezellen	17
7.6.2 Wägesensoren	18
7.6.3 S-Typ-Wägezellen	18
7.6.4 Kompressions-Wägezellen	19
7.7 Alternative Gewichtssensoren	20
7.7.1 Force Sensitive Resistor (FSR)	20
7.7.2 Piezoelektrischer Sensor	20
8 Drahtlose Verbindung zwischen den Komponenten mit dem LoRaWAN Protokoll	21
8.1 Aufsetzen eines Servers (LoRa-Gateway)	21
8.2 Erstellen einer LoRa-Applikation zum Abrufen der Daten	23
8.3 Aufsetzen eines Clients (LoRa-Node)	26

8.4 Spreading Factor	28
8.5 Security / Datensicherheit	29
8.6 RN2483- und UART-Methoden und ihre Funktionalität	30
8.6.1 RN2483.c	30
8.6.2 uart.c	30
9 Application Server	31
9.1 Visualisierung der empfangenen Daten mit Angular CLI	31
9.2 NodeJS Server für die Verarbeitung der Datenpakete	31
9.3 TTN	32
9.4 Inaktivität	33
9.5 Voraussetzungen und Starten des Node-Servers	33
10 Schaltplan und Verlöten der Platine	34
11 Ausblick	37
11.1 LoRaWAN Modul Code verbessern	37
11.2 Front-End Lizenzkosten vermeiden	37
11.3 Erstellen eines wasserfesten Gehäuses	38

Abbildungsverzeichnis

4.1 iC880A-SPI Concentrator Board (links) und Raspberry Pi Model 2B (rechts) bilden zusammen das Gateway	4
4.2 20Kg Wägezelle zwischen zwei Spanholzplatten befestigt	5
4.3 HX711 24-Bit ADC für Gewichtssensoren	6
4.4 Data Link LoRa RN2483 als LoRa-Node	7
5.1 Projektplan	8
6.1 Komponentenmodell	9
7.1 Aufbaubeschreibung einer Wägezelle	10
7.2 Funktionsweise einer Wägezelle bei Gewichtsmessung	11
7.3 Verschaltung der Dehnmessstreifen	12
7.4 Verschaltung der Dehnmessstreifen mit Spannungsmessung	12
7.5 20Kg Wägezelle zwischen zwei Spanholzplatten befestigt	13
7.6 Verdrahtung der Wägezelle	14
7.7 HX711 ADC-Wandler, Ansicht auf Bauteil	15
7.8 Disk-Wägezelle	17
7.9 Wägesensor	18
7.10 S-Typ-Wägezelle	18
7.11 Kompressions-Wägezelle	19
7.12 Force Sensitive Resistor	20
7.13 Piezoelektrischer Sensor	20
8.1 LoRa Schaubild	21
8.2 TTN Gateways	22
8.3 TTN Applications	23
8.4 Application hinzufügen	23
8.5 Registrierte Geräte	24
8.6 Geräteregistrierung	24
8.7 Aktivierungsmethode	25
8.8 Beispielcode	25
8.9 UART Interface	26
8.10 Bootzeit	27
8.11 Spreading Factor	28
8.12 Security	29
10.1 Schaltplan	34

10.2 Rückseite der verlötzten Platine unseres ersten Prototyps	35
10.3 Vorderseite der verlötzten Platine unseres ersten Prototyps	36

Tabellenverzeichnis

3.1 Einkaufsliste	3
-----------------------------	---

1 Projektvorstellung

Wir haben uns in der Veranstaltung Embedded Systems des Studiengangs Software-Engineering bei Prof. Dr. Jürgen Doneit und Herr Ulrich Straus im Sommersemester 2018 für das Projekt LoRa-Messnetzwerk entschieden. Die Aufgabe bestand darin, ein Messnetzwerk aufzubauen, welches mit Daten von Sensoren befüllt wird und über Long Range Wide Area Network, kurz LoRa, verschickt werden.

Unser genauer Use Case ist wie folgt:

In einem Restaurant wird das Gewicht jeder Kaffeekanne durch einen Wäge Sensor gemessen, sodass man ungefähr weiß, wie viel Kaffee sich noch in der jeweiligen Kanne befindet. Damit dies passieren kann, muss die Kaffeekanne auf der Wäge Plattform abgestellt werden. Die Messdaten werden an einzelne dazugehörige LoRa-Nodes übertragen, die daraufhin über LoRa mit dem LoRa-Gateway kommunizieren. Das LoRa-Gateway empfängt die Sensorgewichtsdaten und leitet diese auf einen Server weiter, wodurch dann die Füllstände der Kaffeekannen visualisiert werden. Hierbei befindet sich dann beispielsweise ein Dashboard in der Küche, wodurch ein genauer Überblick über die einzelnen Füllstände jeder Kaffeekanne gewährleistet wird, sodass nicht ständig manuell durch den Kellner geprüft werden muss, ob eine Kaffeekanne noch genügend Kaffee beinhaltet. Die Wäge Plattformen werden durch Akkus betrieben, sodass eine längere Laufzeit möglich ist, da LoRa sehr energiesparend ist.

2 Ausgangssituation

Es gab noch kein Projekt aus den vorherigen Semestern, sodass wir ganz von Anfang damit beginnen konnten. Hierbei war es sehr wichtig viel Recherche zu betreiben, da noch keine Vorahnung vorhanden war. Niemand hatte bis jetzt mit LoRa oder Sensoren gearbeitet, sodass wir uns anfangs viel einlesen mussten. Durch das LCD-Display Projekt am Anfang des Semesters konnten wir uns in die Programmiersprache C einarbeiten. Außerdem hat der Großteil der Gruppe noch keine Erfahrung mit dem Löten, sodass auch hier die Fähigkeiten erlernt und verbessert wurden. Wir arbeiteten anfangs mit dem AVR STK 500 und dem Mikrokontroller ATMega644PA und konnten uns durch Foreneinträge, Tutorials und Videos schnell darin einarbeiten.

3 Einkaufsliste

Tabelle 3.1: Einkaufsliste

Komponente	Preis
LoRa-Gateway iC880A-SPI + Antenne + Pigtail cable + Raspberry Pi 2 B	167.90 €
LoRa-Node Data Link LoRa RN2483 + Antenne	Ca. 70 €
Wägezelle + HX711	10.71 €

4 Überblick der Bauteile

4.1 Gateway

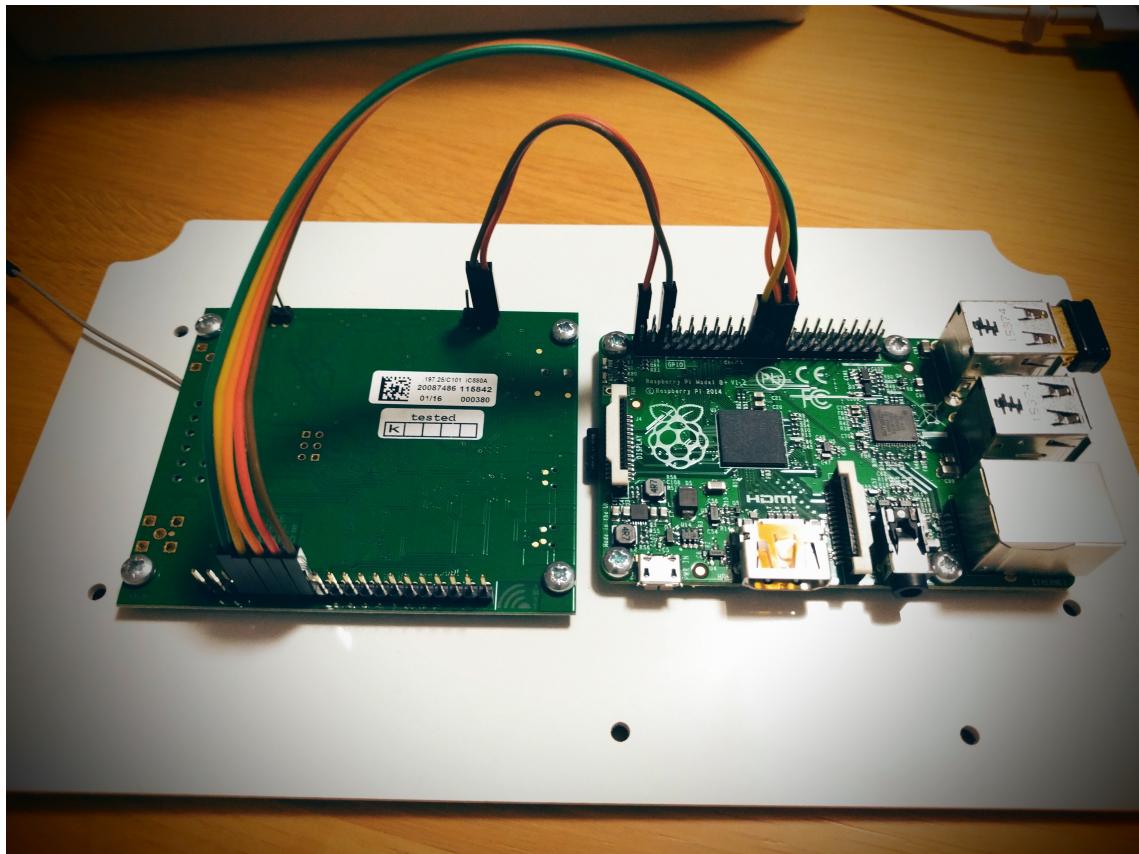


Abbildung 4.1: iC880A-SPI Concentrator Board (links) und Raspberry Pi Model 2B (rechts) bilden zusammen das Gateway

Quelle: <https://raw.githubusercontent.com/ttn-zh/ic880a-gateway/spi/images/mounted-boards.jpg>

4.2 Wägezelle



Abbildung 4.2: 20Kg Wägezelle zwischen zwei Spanholzplatten befestigt

4.3 Gewichtssensor

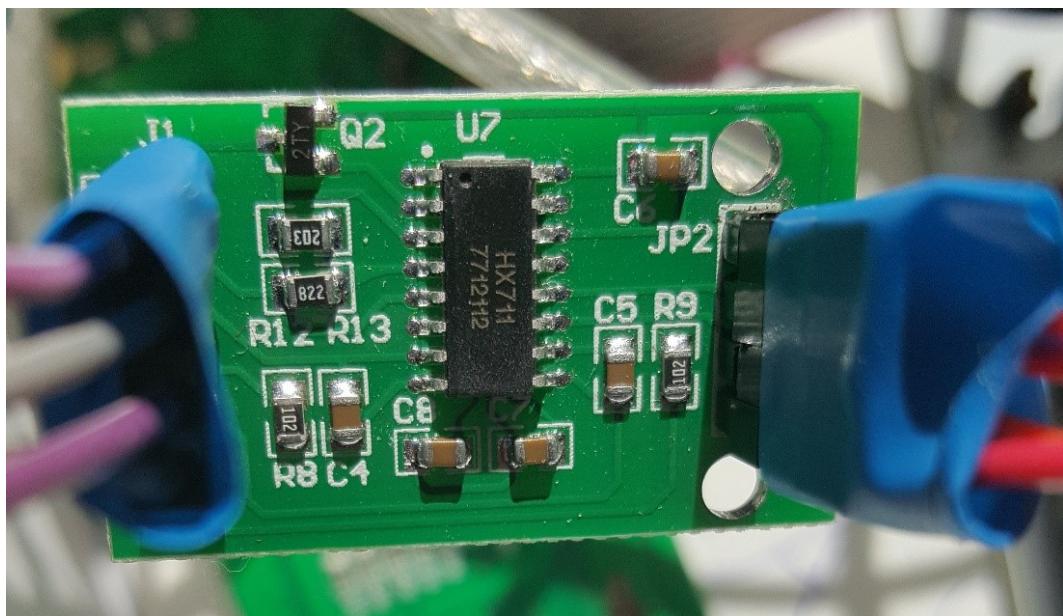


Abbildung 4.3: HX711 24-Bit ADC für Gewichtssensoren

4.4 LoRa Node

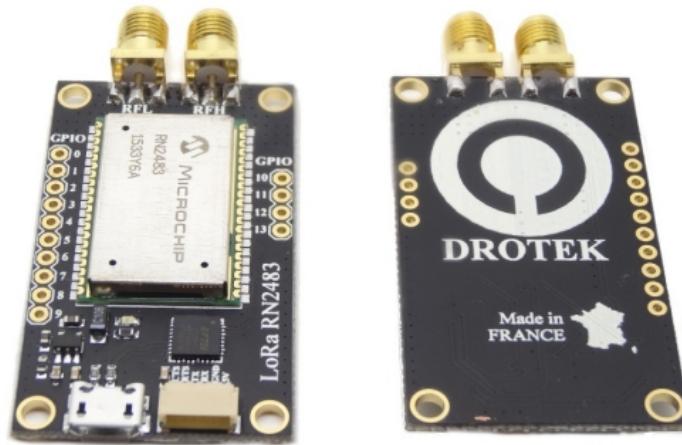


Abbildung 4.4: Data Link LoRa RN2483 als LoRa-Node

5 Projektplan

Der Projektplan unseres Projektes hat sich, wie schon von uns vermutet, im Laufe des Semesters mehrmals verändert. Zum Projektstart kamen wir wie geplant recht zügig voran und es machten sich sehr schnell gute Fortschritte bemerkbar. Die ersten Realisierungen auf dem STK wurden in der geplanten Zeit fertiggestellt. Im späteren Teil des Projektes verbrauchten wir allerdings unseren zuvor angelegten Puffer etwas, um Verbesserung der Software und den Umzug auf die Platine zu gestalten. Die Tests am Ende des Projektes konnten bis kurz vor der Abschlusspräsentation abgeschlossen werden, sodass wir zur Präsentation ein gut funktionierendes System vorweisen konnten. Nach der Präsentation stellten wir dann gemeinsam die Dokumentation fertig.

Gefährdet	Aufgabenname	Start Datum	Ende Datum	Zugewiesen	% vollständig
	Projektstart	26.03.18	03.04.18		100%
	Kick-Off	26.03.18	26.03.18	Alle	100%
	Konzeptionierung	26.03.18	26.03.18	Alle	100%
	Arbeitseinteilung	26.03.18	26.03.18	Alle	100%
	Recherche	27.03.18	03.04.18	Rene	100%
	Kraftsensor und Verteilung des gewichts	27.03.18	03.04.18	Holger	100%
	LORA Protokoll	27.03.18	03.04.18	Nick	100%
	LORA Server	27.03.18	03.04.18	Sinan	100%
	Stromversorgung	27.03.18	03.04.18	Marcel	100%
	Grobe Realisierung	03.04.18	17.04.18		100%
	Bestellung	03.04.18	03.04.18	Alle	100%
	Steckboard bestücken	03.04.18	17.04.18	Nick/Holger	100%
	LORA Server	03.04.18	17.04.18	Sinan	100%
	Prototypische Implementierung	16.04.18	25.06.18		100%
	Bestellte Teile einbauen	16.04.18	03.05.18	Alle	100%
	Übertrag auf Steckplatine	07.05.18	15.06.18	René/Marcel	100%
	Schaltplan erstellen	07.05.18	25.06.18	René/Marcel	100%
	Produkt auf Platine umziehen	21.05.18	25.06.18	René/Marcel	100%
	Testen	11.05.18	02.07.18		100%
	Hardwaretest	11.05.18	21.05.18	Alle	100%
	Hardwaretest mit Platine	25.06.18	02.07.18	Alle	100%
	Softwaretest	21.05.18	02.07.18	Alle	100%
	Dokumentation	29.06.18	06.07.18		100%
	Zusammenführen	29.06.18	06.07.18	Alle	100%
	Formatieren	29.06.18	06.07.18	Alle	100%
	Rechtschreibprüfung	29.06.18	06.07.18	Alle	100%

Abbildung 5.1: Projektplan

6 Komponentenmodell

Hier sieht man unser aktuelles Komponentenmodell. Dort ist zu erkennen, welche Komponenten und Bauteile von uns verwendet werden. Auch wie diese miteinander kommunizieren kann man aus dem Modell entnehmen.

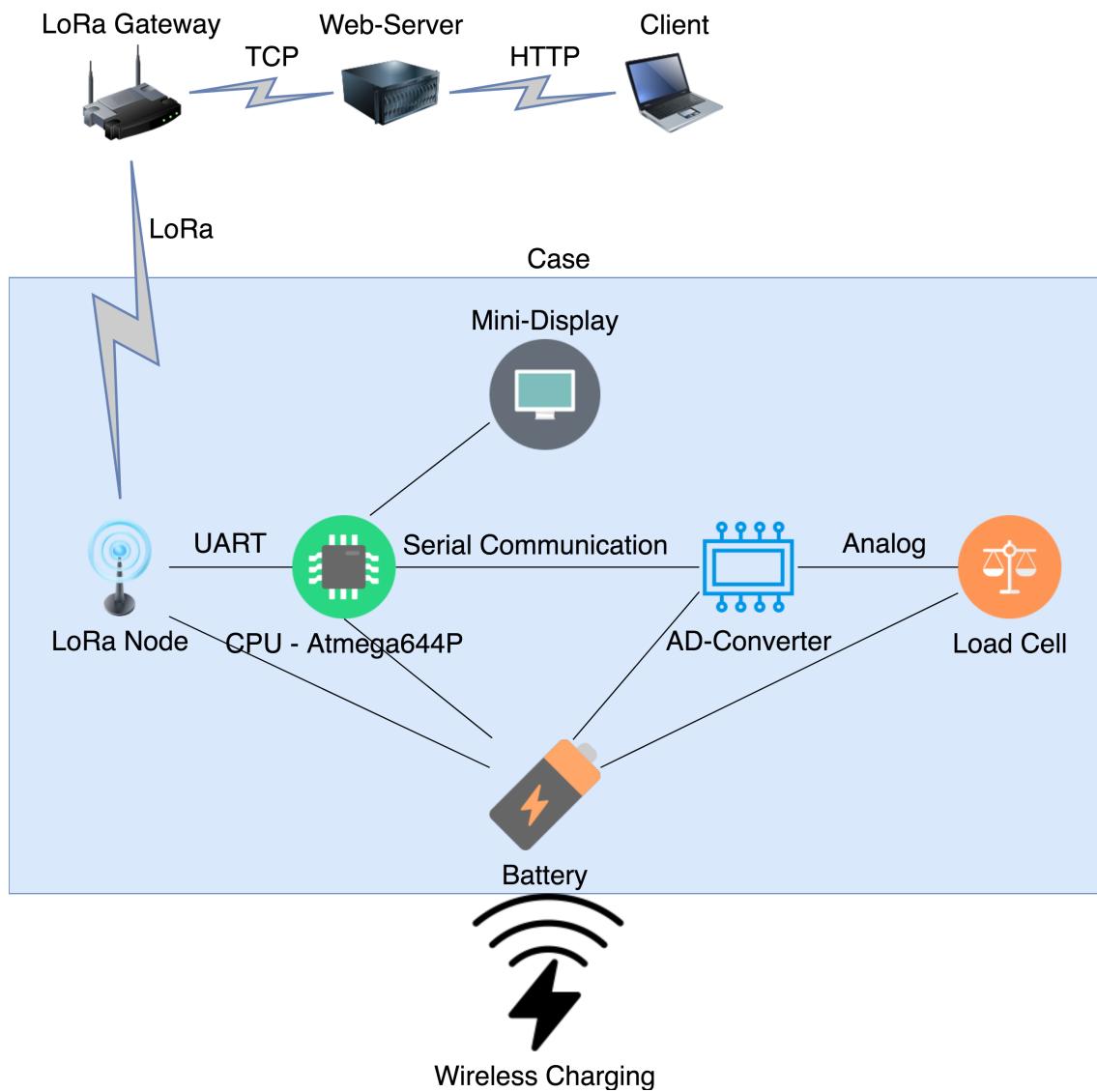


Abbildung 6.1: Komponentenmodell

7 Gewichtsmessung mit einer Wägezelle

7.1 Funktionsweise der Wägezelle

Wägezellen sind Sensoren, die das Gewicht über Verformung ihres Materials mit Hilfe von Dehnungsmessstreifen messen können. Es sind vier Dehnungsmessstreifen, wie im Bild unten zu sehen, an der Wägezelle angebracht.

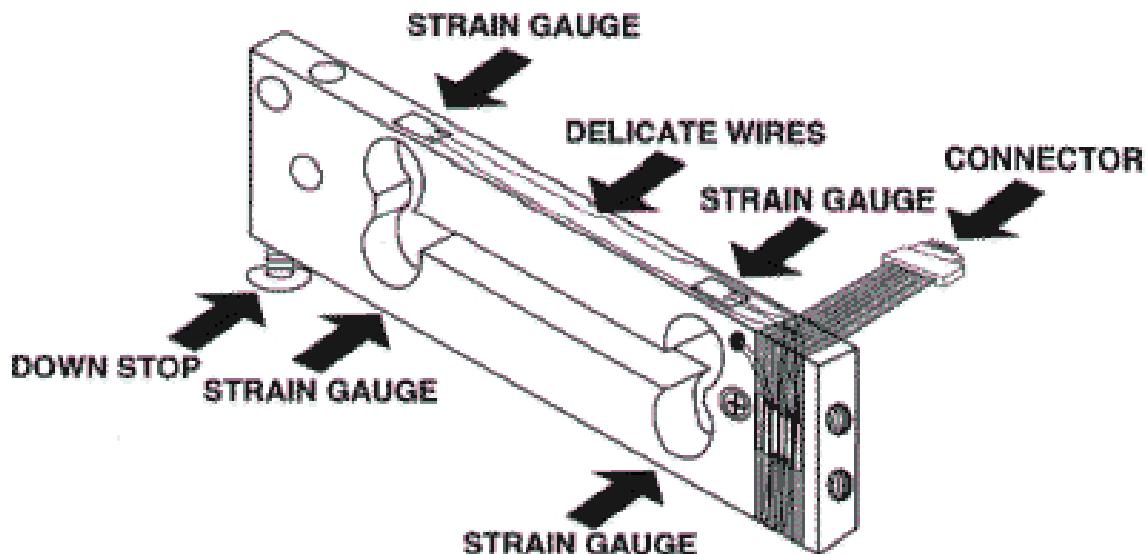


Abbildung 7.1: Aufbaubeschreibung einer Wägezelle

Quelle: <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells>

Bei Auflage von Gewicht bzw. Verformung der Wägezelle messen zwei Dehnungsmessstreifen die Kompression und die anderen zwei die Spannung. Bei Kompression des Dehnungsmessstreifens werden die Leiter dicker (höherer Leiterquerschnitt) und kürzer (geringere Leiterlänge). Laut der Formel für den Leiterwiderstand ($R = \rho * l/A$, Leiterwiderstand = spezifischer Widerstand * Leiterlänge/Leiterquerschnitt) verringert sich der Widerstand des Dehnungsmessstreifens. Bei Spannung des Dehnungsmessstreifens werden die Leiter dünner und länger, somit erhöht sich auch der Widerstand des Dehnungsmessstreifens.

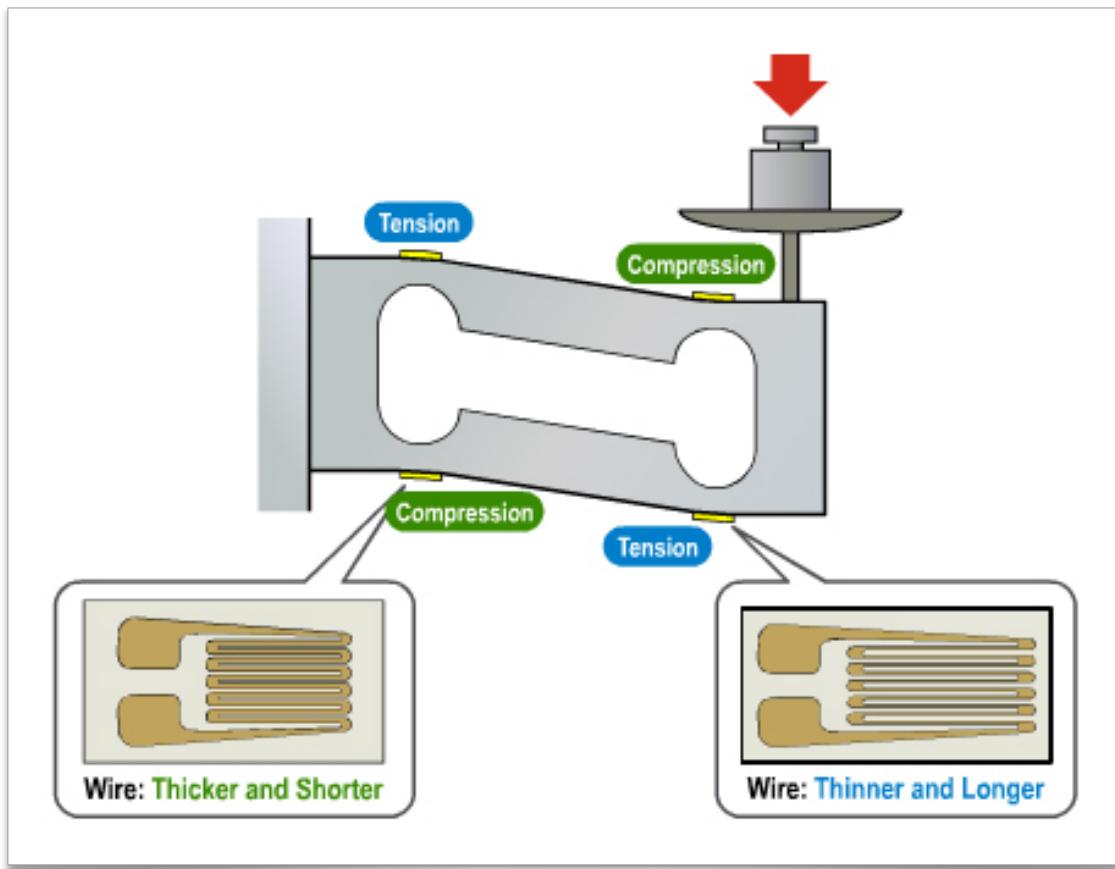


Abbildung 7.2: Funktionsweise einer Wägezelle bei Gewichtsmessung

Quelle: <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells>

Um die Widerstandsveränderungen der Dehnungsmessstreifen messen zu können, sind die Dehnungsmessstreifen innerhalb der Wägezelle in einer Wheatstone-Brücken-Formation geschalten.

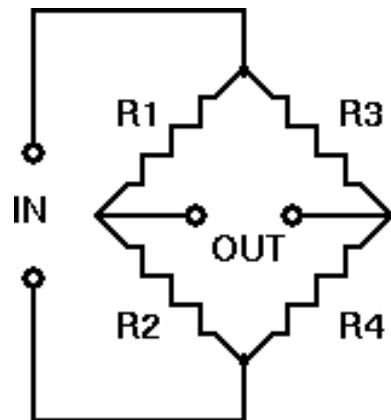


Abbildung 7.3: Verschaltung der Dehnmessstreifen

Quelle: <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells>

Wenn gilt $R1/R2 = R3/R4$, dann misst man 0V an Out. Falls sich ein Widerstand verändern sollte, zum Beispiel durch Verformung eines Dehnmessstreifens, lässt sich eine Spannung an Out laut dieser Formel messen:

$$V_{out} = [(R3/(R3+R4)-R2/(R1+R2))]*V_{in}$$

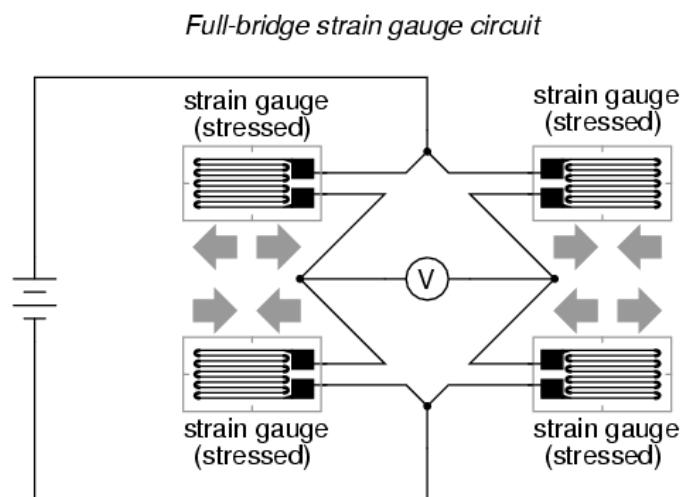


Abbildung 7.4: Verschaltung der Dehnmessstreifen mit Spannungsmessung

Quelle: <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells>

Die an V Out gemessene Spannung ist sehr klein und analog, deshalb benutzen wir den HX711, welcher das Signal verstärkt und in digitale Werte [0 bis 16777217] umwandelt.

7.2 Versuchsaufbau zur Gewichtsmessung

Die Wägezelle wurde an zwei gegenüberliegenden Punkten an zwei Holzplatten geschraubt, damit sich das Gewicht auf alle Dehnungsmesstreifen auswirkt.



Abbildung 7.5: 20Kg Wägezelle zwischen zwei Spanholzplatten befestigt

Die Wägezelle wurde wie folgt an das HX711-9Modul angeschlossen:

- Rot an E+ (Excitation+/Vin+)
- Schwarz an E- (Excitation-/Vin-)
- Weiß an A+ (Output+/Vout+)
- Grün an A- (Output-/Vout-)

Veranschaulichung der Verdrahtung (Achtung! Gelbe Verbindung ist bei uns weiß):
Die A-Anschlüsse und B-Anschlüsse sind für verschiedene Verstärkungskanäle. Der

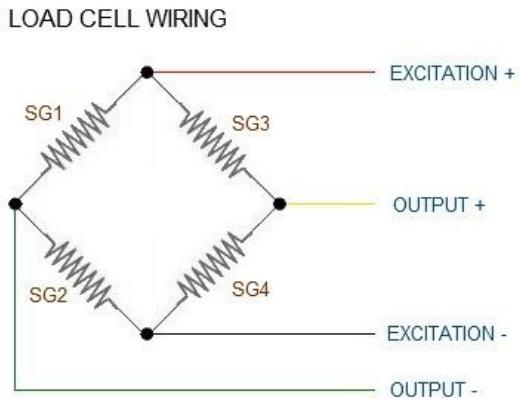


Abbildung 7.6: Verdrahtung der Wägezelle

Quelle: [https://learn.sparkfun.com/tutorials/
load-cell-amplifier-hx711-breakout-hookup-guide](https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide)

A-Kanal hat einen programmierbaren Verstärkungsfaktor von 128 oder 64 und der
B-Kanal hat einen festen Verstärkungsfaktor von 32.

Die rechte Seite des HX711 wird wie folgt angeschlossen:

- GND an Ground
- VCC an Versorgungsspannung in unserem Fall 5V
- SCK an PD5 unseres Microcontrollers/Atmega 644PA
- DT (Data) an PD6 unseres Microcontrollers/Atmega 644PA

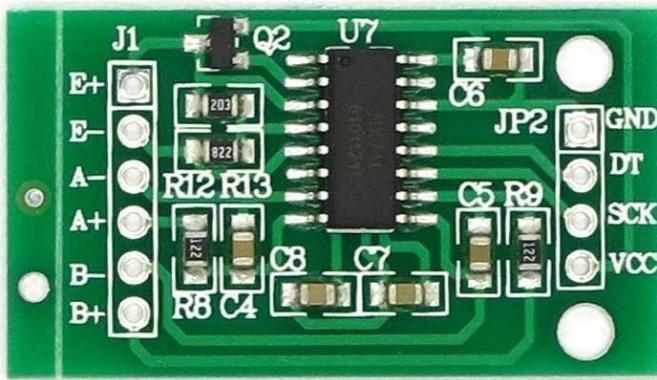


Abbildung 7.7: HX711 ADC-Wandler, Ansicht auf Bauteil
Quelle: <https://tinyurl.com/y8grwwdb>

7.3 Programmlogik unseres Microchips

Das Programm basiert auf getsidd's HX711 AVR Bibliothek (<https://github.com/getssidd/HX711>). Die wichtigsten Additionen von uns sind die Möglichkeit, negative Gewichtsveränderungen wahrnehmen zu können, auf Tastendruck eine Tar-Funktion auszuführen und der Gewichts-Sende-Algorithmus in Verbindung mit dem LoRa-Modul.

7.3.1 Vereinfachte Funktionsweise für momentane Gewichtsmessung

Beim vom HX711 gelesenen Gewicht wird das getarte Gewicht (Tar Funktion/Gewichtsnullung) abgezogen und durch die Kalibration dividiert, um einen lesbaren Wert in Kilogramm zu erhalten.

7.3.2 Algorithmus für Rausch- und Driftunterdrückung

Das Differenzgewicht von der letzten und der momentanen Gewichtsmessung wird mit einer vorher gewählten Veränderungsrate verglichen. Wenn das Differenzgewicht höher als die gewählte Veränderungsrate ist, wird der momentan gemessene Wert verwendet und angezeigt, im anderen Fall wird der Wert verworfen, weil er vermutlich

durch Rauschen oder Drift verursacht wurde.

Der Veränderungswert gibt auch die kleinstmögliche messbare Veränderung an. Am Beispiel unseres Projektes könnte bei einem Veränderungswert von ca. 5g, jemand mit einem Strohhalm Flüssigkeit aus dem Behältnis saugen, ohne dass das Programm die Gewichtsveränderung wahrnimmt.

7.3.3 Algorithmus für Gewichtssendung mit dem LoRa-Modul

Mit Hilfe eines Interrupts, wird jede Sekunde eine Gewichtsmessung durchgeführt. Beim Ablauf des Sendeintervalls (in unserem Projekt momentan eingestellt auf fünfzehn Sekunden), wird in der Menge der Gewichtsmessungen (in unserem Fall fünfzehn Messungen, jede Sekunde eine), nach den letzten fünf Gewichtsmessungen gesucht, die den gleichen Wert haben. Wenn die Messungen diese Bedingung nicht erfüllen, wird kein Wert gesendet.

7.4 Demonstration der Gewichtsmessung

Hier eine Demo zur Gewichtsmessung, die wir auch in der Zwischenpräsentation gezeigt haben:

<https://streamable.com/ycxf3>

Alternativ: <https://www.youtube.com/watch?v=2tk0ydRsXgg>

7.5 Use-Cases von Straight bar Wägezellen

Die Art von uns verwendeter Wägezelle wird unter anderem in diesen Bereichen verwendet:

- Küchenwaagen
- Industrielle Gewichtsmessung, die an einem Punkt erfolgt

7.6 Alternative Bauformen von Wägezellen

Wägezellen werden in verschiedenen Bauformen hergestellt, hier werden diese aufgezählt und untersucht um die richtige Wägezelle bei Änderung der Projektanforderungen auswählen zu können

7.6.1 Disk-Wägezellen

Disk-Wägezellen haben eine runde Form und sind kompakter gebaut.



Abbildung 7.8: Disk-Wägezelle

Quelle: http://www.forsentek.com/prodetail_13.html

7.6.2 Wägesensoren

Wägesensoren besitzen nur einen Dehnmessstreifen anstatt herkömmliche, die vier besitzen. Man kann vier Wägesensoren in einer Wheatstone-Bridge-Formation zusammen schalten und wie eine Straight-Bar-Wägezelle betreiben. Dabei kann man die einzelnen Wägesensoren auf einer größeren Fläche betreiben und sie als Personenwaage, Fahrzeugwaage oder allgemein als Waage für große Objekte verwenden.



Abbildung 7.9: Wägesensor

Quelle:

<https://www.botshop.co.za/product/load-cell-sensor-resistance-strain-50kg/>

7.6.3 S-Typ-Wägezellen

Diese Wägezellen lassen sich, dank ihrer S-Form, in Spannungs- und Kompressionsbetrieb betreiben.



Abbildung 7.10: S-Typ-Wägezelle

Quelle: <https://www.coventryscales.co.uk/scale-type/load-cells/tension-s-type-load-cells/s-type-load-cell/>

7.6.4 Kompressions-Wägezellen

Diese Art von Wägezellen sind nur in Kompression belastbar.



Abbildung 7.11: Kompressions-Wägezelle

Quelle: [http://www.eilersen.com/compression-load-cell/product/
atex-compression-load-cell-dla/](http://www.eilersen.com/compression-load-cell/product/atex-compression-load-cell-dla/)

7.7 Alternative Gewichtssensoren

7.7.1 Force Sensitive Resistor (FSR)

Bei Kraftzunahme beziehungsweise Kompression des Sensors verringert sich der Abstand zwischen den Sensorfolien, die voneinander mit einer speziellen Tinte getrennt sind, und somit auch ihr Widerstand. Diese Sensoren sind sehr platzsparend, aber auch ungenauer als Wägezellen, weshalb sie nicht sehr gut für die Gewichtsmessung geeignet sind.



Abbildung 7.12: Force Sensitive Resistor

Quelle: <https://solarbotics.com/product/50803/>

7.7.2 Piezoelektrischer Sensor

Bei Krafteinwirkung produzieren diese Sensoren eine elektrische Ladung, die gemessen werden kann. Sie sind eher für dynamische Anwendungen geeignet, da Signale nur bei Kraftänderungen gemessen werden können, beziehungsweise die Ladung sehr schnell wieder gegen Null geht. Es gibt spezielle Varianten, die ihre Ladung bis zu einer Minute halten können, diese weisen jedoch einen hohen Drift auf.

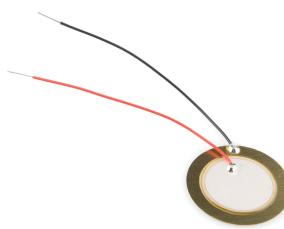


Abbildung 7.13: Piezoelektrischer Sensor

Quelle: <https://www.sparkfun.com/products/10293>

8 Drahtlose Verbindung zwischen den Komponenten mit dem LoRaWAN Protokoll

8.1 Aufsetzen eines Servers (LoRa-Gateway)

Unser LoRa-Gateway hat im Prinzip die Funktion eines WLAN-Routers. Alle Clients (LoRa-Nodes) melden sich an dem Gateway an und kommunizieren mit dem Gateway. Wie der Begriff „Gateway“ schon andeutet, ist dies ein Tor, das zum Internet führt (siehe folgende Abbildung 8.1).

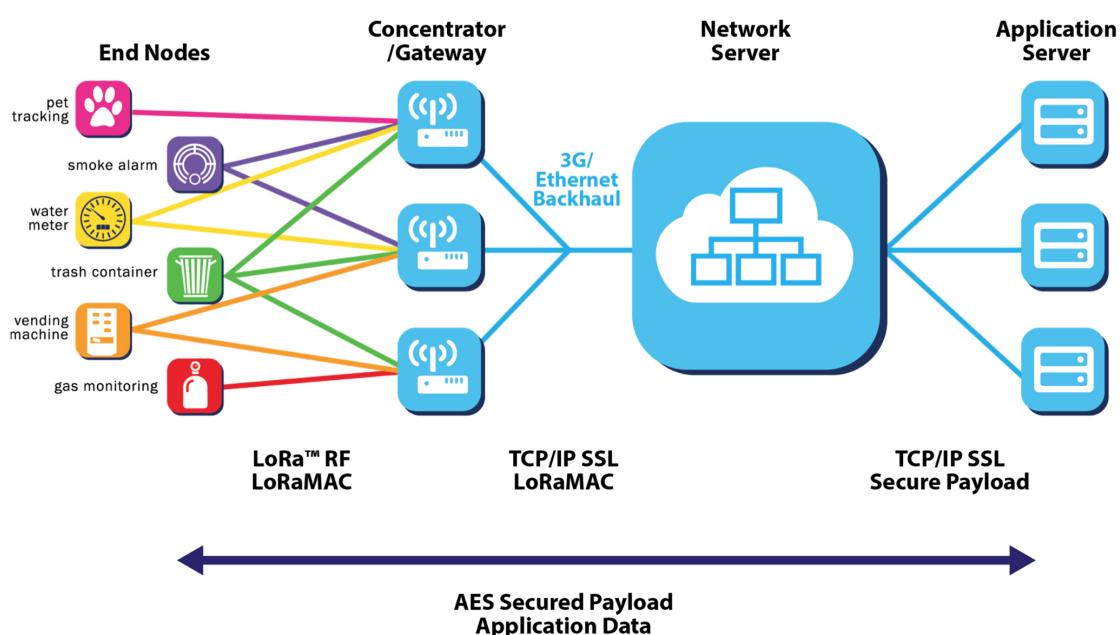


Abbildung 8.1: LoRa Schaubild

Quelle: <http://www.mstanley.co.uk/blog/wp-content/uploads/2015/11/Enabling-world-wide-mobility-for-the-IoT-image-2.jpg>

Realisiert wird das alles durch ein „iC880A-SPI Concentrator Board“, das mit Hilfe einer Antenne mit den LoRa-Nodes Daten austauscht. Diese Daten werden durch ein Raspberry Pi, der an das Serial Peripheral Interface (SPI) des Concentrator Boards angeschlossen ist, ins Internet versendet.

ACHTUNG! Es ist sehr wichtig, dass alles korrekt angeschlossen ist und schon eine

Antenne an das Concentrator Board angeschlossen ist, da es sonst zu Schäden an den Bauteilen kommen kann.

Auf dem Wiki der GitHub-Seite von The Things Network Zurich (<https://github.com/ttn-zh/ic880a-gateway/wiki>) kann man nachlesen welche Bauteile man benötigt, wie man die Pins verbinden soll, das Betriebssystem aufsetzt und anschließend das Gateway im TheThingsNetwork registriert.

Nachdem man die Anweisungen zum Konfigurieren befolgt hat, sollte es auf der Webseite für Gateways von TheThingsNetwork zu sehen sein.

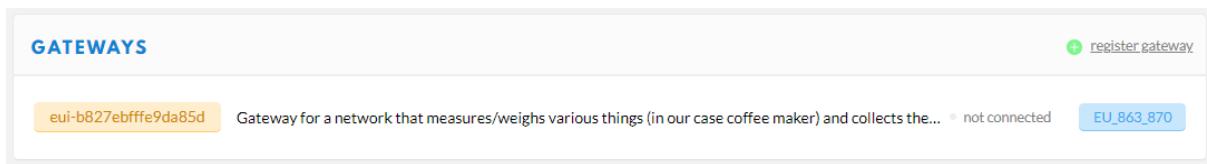


Abbildung 8.2: TTN Gateways

Quelle: <https://console.thethingsnetwork.org/gateways>

8.2 Erstellen einer LoRa-Applikation zum Abrufen der Daten

Damit wir jetzt auch mit den verschlüsselten Daten, die wir empfangen, etwas anfangen können, müssen wir erstmal eine Applikation in der Konsole von TheThingsNetwork erstellen.

The screenshot shows the 'APPLICATIONS' section of the TheThingsNetwork console. A single application named 'lora_measure_network' is listed. The description for this application is: 'Network that measures/weights various things (in our case coffee maker) and collects their status (f...'. To the right of the application name, there is a red box highlighting the 'add application' button. Below the application list, there is some network-related information: 'ttn-handler-eu' and a series of hex digits: '70 B3 D5 7E D0 00 C5 AE'.

Abbildung 8.3: TTN Applications

Quelle: <https://console.thethingsnetwork.org/applications>

The screenshot shows the 'ADD APPLICATION' form. It contains several input fields: 'Application ID' (with value 'test2414414'), 'Description' (with value 'Just a test application.'), 'Application EUI' (with placeholder 'EUI issued by The Things Network'), and 'Handler registration' (with value 'ttn-handler-eu'). At the bottom right of the form, there are two buttons: 'Cancel' and a red-bordered 'Add application' button.

Abbildung 8.4: Application hinzufügen

Quelle: <https://console.thethingsnetwork.org/applications>

Application ID: Sollte einzigartig und eindeutig sein.

Description: Eine kurze, aber aussagekräftige Beschreibung der Applikation.

Handler registration: Sollte passend zum Standort gewählt werden.

Nachdem man nun eine Applikation erstellt hat, sollte man nun auch ein Gerät zu dieser Applikation hinzufügen.

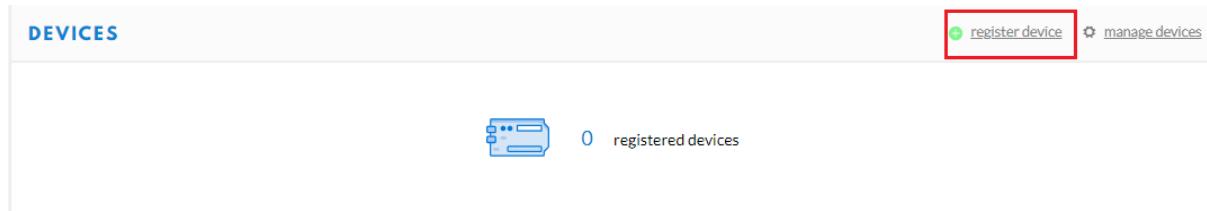


Abbildung 8.5: Registrierte Geräte

A screenshot of the 'REGISTER DEVICE' form. It has several fields: 'Device ID' (input: 'test1'), 'Device EUI' (input: ''), 'App Key' (input: ''), and 'App EUI' (input: '70 B3 D5 7E D0 01 05 17'). At the bottom right, there are 'Cancel' and 'Register' buttons, with 'Register' highlighted by a red box.

Abbildung 8.6: Geräteregistrierung

Quelle: <https://console.thethingsnetwork.org/applications>

Device ID: Sollte einzigartig und am besten aussagekräftig gewählt werden, damit man diese ID später einem Gerät zuordnen kann.

Device EUI: Kann man selber bestimmen oder generieren lassen, indem man das Zufällig-Icon bzw. den Stift-Icon anklickt, um es umzuschalten. Dies wird später benötigt, um das Gerät richtig zu konfigurieren!

App Key: Kann man selber bestimmen oder generieren lassen, indem man das Zufällig-Icon bzw. den Stift-Icon anklickt um es umzuschalten. Dies wird später benötigt um das Gerät richtig zu konfigurieren!

App EUI: Es sollte schon eine vorgewählt sein. Nachdem wir ein Gerät erstellt haben, sollten wir die Aktivierungsmethode ändern:

Dazu gehen wir in *Application* → *application_id* → *devices* → *device_id* → *Settings*. Darin kann man die Activation Method von OTAA zu ABP ändern.

Activation Method



Abbildung 8.7: Aktivierungsmethode

Quelle: <https://console.thethingsnetwork.org/applications>

Wenn wir die Änderungen speichern, sollten wir wieder auf die Seite des Geräts kommen. Scrollen wir ganz nach unten, dann sehen wir ein Codebeispiel.

```
EXAMPLE CODE

1 const char *devAddr = "26011C0D";
2 const char *nwkSKey = "FEDC44AB0EED5686F0A43523612ECE23";
3 const char *appSKey = "DAF11CBE2CE9E1345BA1715CE3407D25";
```

Abbildung 8.8: Beispielcode

Quelle: <https://console.thethingsnetwork.org/applications>

Diesen Codeausschnitt sollten wir entweder irgendwo speichern oder uns einfach merken, dass er auf der Seite des Geräts ist, da wir ihn später zum Konfigurieren des Nodes wieder brauchen werden.

8.3 Aufsetzen eines Clients (LoRa-Node)

Bevor wir die LoRa-Nodes mit Strom versorgen, sollten wir erstmal eine Antenne an den RFH (Radio Frequency High Band – PIN 23) Anschluss befestigen.

Unser LoRa-Node ist über die serielle Schnittstelle UART (PIN 6 + 7 am RN2483) an den Mikrocontroller Atmega644PA (PD0 bzw. PIN 14 + PD1 bzw. PIN 15) angeschlossen und kommuniziert so mit dem Mikrochip. Die Baudrate für die UART Verbindung beträgt standardmäßig 57600 bps, kann aber auch mit einer gewissen Character Folge geändert werden. (Siehe folgende Abbildung 8.9)

1.4 UART INTERFACE

All of the RN2483 module's settings and commands are transmitted over UART using the ASCII interface.

All commands need to be terminated with <CR><LF> and any replies they generate will also be terminated by the same sequence.

The default settings for the UART interface are 57600 bps, 8 bits, no parity, 1 Stop bit, no flow control. The baud rate can be changed by triggering the auto-baud detection sequence of the module. To do this, the host system needs to transmit to the module a break condition followed by a 0x55 character at the new baud rate. The auto-baud detection mechanism can also be triggered during Sleep to wake the module up before the predetermined time has expired.

Note: A break condition is signaled to the module by keeping the UART_RX pin low for longer than the time to transmit a complete character. For example, at the default baud rate of 57600 bps keeping the UART_RX pin low for 938 µs is a valid break condition, whereas at 9600 bps this would be interpreted as a 0x00 character. Thus, the break condition needs to be long enough to still be interpreted as such at the baud rate that is currently in use.

Abbildung 8.9: UART Interface

Quelle: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001784F.pdf>, Kapitel 1.4, Seite 15

Kommen wir nun zur Initialisierung des RN2483, wenn dieser nun am Microchip hängt und soweit ansprechbar ist:

Folgende Methoden haben wir von der Drotek RN2483 Library (<https://github.com/drotek/RN2483>) für Arduino genommen und für den AVR angepasst und umgeschrieben. Außerdem haben wir einen Teil von der UART-Library von unserem Betreuer Ulrich Straus übernommen.

Zu aller erst setzen wir den RN2483 zurück, indem wir per UART die Befehle „**sys factoryRESET**“ und „**sys reset**“ senden und jeweils mit einem Break- und New-Line-Character bestätigen. (ASCII: Break = 0x0D; New-Line = 0x0A).

Als nächstes füttern wir den RN2483 mit den Daten, die wir beim Erstellen des Geräts im Applikations-Server erhalten haben. Dazu werden folgende Befehle benötigt:

- **mac set devaddr <address>**
- **mac set nwkskey <nwksesskey>**

- **mac set appskey <appSesskey>**

Also mit dem Beispielcode von Abbildung 8.8 wäre dann:

<address> = 26011C0D;
<nwksesskey> = FEDC44AB0EED5686F0A43523612ECE23;
<appSesskey> = DAF11CBE2CE9E1345BA1715CE3407D25

Nun müssen wir noch den SpreadingFactor (kurz: SF) festlegen und anschließend die Konfiguration speichern und dem Netzwerk beitreten.

Dies wird wie folgt gemacht:

- **mac set dr 5: wobei 5 = SF7/125 kHz**
- **mac save**
- **mac join abp**

Nach jedem Befehl benutzen wir derzeit noch eine Delay-Funktion um dem RN2483 Chip Zeit zum Bearbeiten und Antworten zu lassen. Die aufgerundeten Delay-Zeiten bzw. Berechnungszeiten sieht man im folgenden Diagramm.

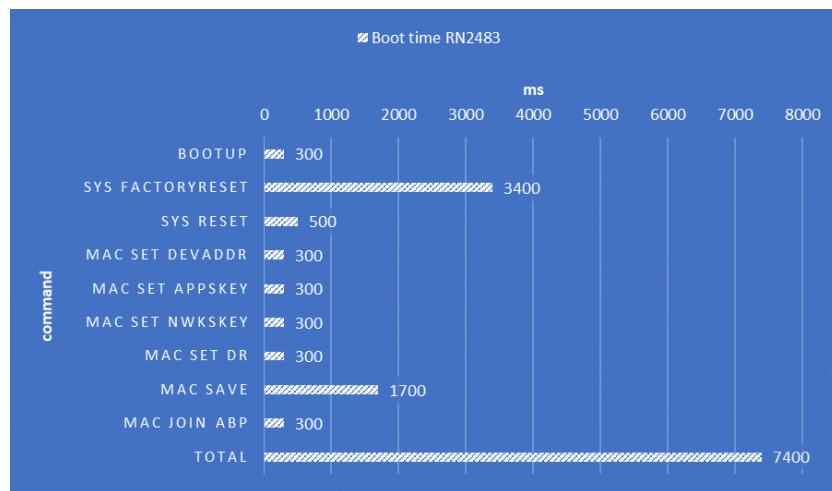


Abbildung 8.10: Bootzeit

Jetzt ist der RN2483 fertig konfiguriert und wir können mit dem Befehl „**mac tx <type> <portno> <data>**“ Nachrichten versenden. Dabei ist:

<type> = cnf (confirmed) oder uncnf (unconfirmed);

<portno> = Portnummer zwischen einschließlich 1 und 223;

<data> = Die Nachricht als Hexadezimal Wert.

Die vollständige Kommandoliste mit Syntax und Rückgabewerten findet man im RN2483 LoRa Technology Module Command Reference User's Guide: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001784F.pdf>

Wir benötigen aus dieser Liste nur die System Commands aus Kapitel 2.3 und die MAC Commands aus Kapitel 2.4

8.4 Spreading Factor

Spreading Factor (kurz SF) ist, wie der Begriff teilweise schon verrät, ein Faktor, der angibt wie weit sich eine versendete Nachricht ausbreitet, also was für eine Reichweite sie hat. Umso höher dieser Faktor, desto höher die Reichweite. Jedoch leidet die Geschwindigkeit der Übertragung darunter. Man kann sich merken: kleinerer SF = höhere Bitrate, nicht so hohe Reichweite. Hoher SF = niedrigere Bitrate, höhere Reichweite. In der folgenden Tabelle ist dies sehr gut erkennbar.

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

Abbildung 8.11: Spreading Factor

Quelle: <https://witekio.com/wp-content/uploads/2018/01/lora-wan-spreading-factor.png>

Derzeit sind unsere Payloads (Nachrichten) durchschnittlich nur 2 Bytes lang, da wir nur den Füllstand übermitteln. Beim SF7 könnten wir 647 Nachrichten am Tag senden, wenn wir unter dem 30 Sekunden Sendelimit bleiben wollen.

Die genaue Berechnung und weitere Infos zum Spreading Factor findet man auf folgender Google Docs Tabelle: <https://docs.google.com/spreadsheets/d/1QvcKsGeTTPpr9icj4XkKXq4r2zTc2j0gsHLrnplzM3I/edit>

8.5 Security / Datensicherheit

Jedes gute System sollte vor Angreifern geschützt sein. Und das ist auch bei LoRaWAN von TheThingsNetwork der Fall. Die Nachrichten werden vor dem Versenden verschlüsselt und werden beim Empfänger mit Hilfe des App-Session-Key entschlüsselt. Da die Nachrichten sowieso abgefangen werden können, weil ein Radio-Protokoll benutzt wird, kann der Angreifer diese Nachricht nicht lesen, da sie verschlüsselt ist. Er könnte aber diese Nachricht erneut senden (Replay-Angriff). Als Maßnahme, um gegen diese Art von Angriff zu schützen, wird ein Frame Counter eingesetzt der jedes Mal, wenn eine Nachricht versendet wird sich um 1 erhöht. So werden alle alten Nachrichten ignoriert und man ist vor Replay-Angriffen geschützt.

Weitere Infos hierzu gibt es auf der TheThingsNetwork Seite zu Security: <https://www.thethingsnetwork.org/docs/lorawan/security.html>

Folgende Abbildung zeigt auch wie die Komponenten (LoRa-Nodes und LoRa-Gateway) mit den Daten umgehen.

FIGURE 1-1: SIMPLE LoRa™ TECHNOLOGY NETWORK DIAGRAM

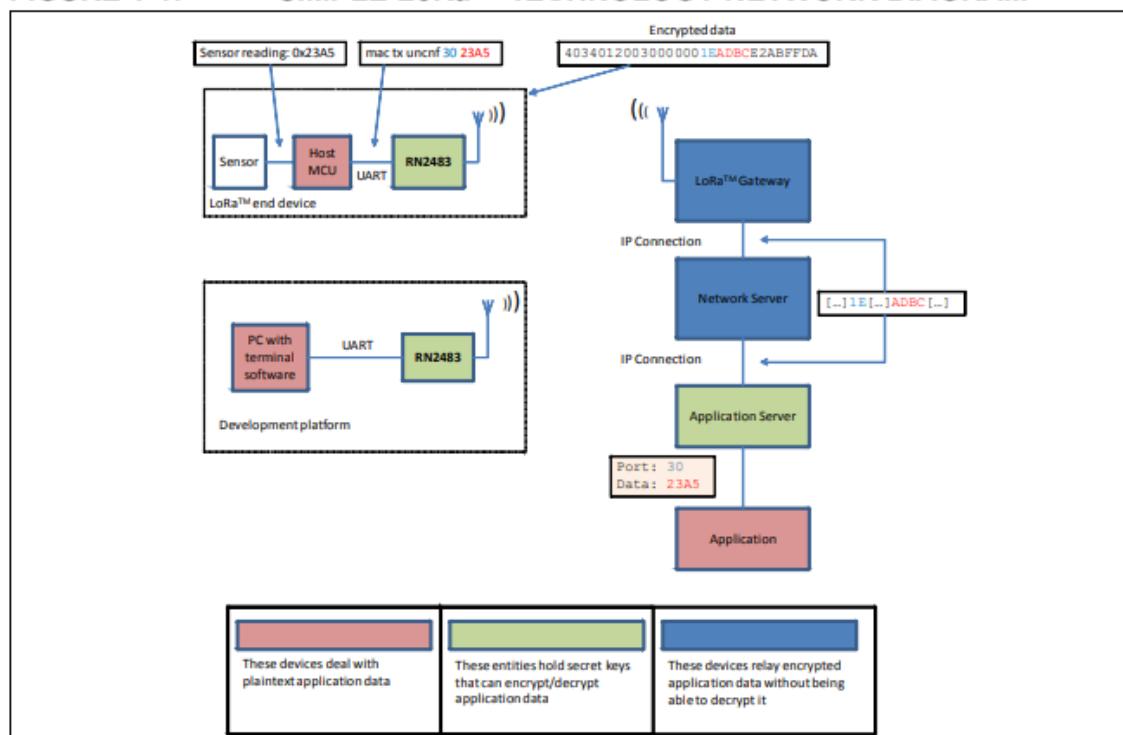


Abbildung 8.12: Security

Quelle: <http://ww1.microchip.com/downloads/en/DeviceDoc/40001784F.pdf>, Kapitel 1.1, Seite 1

8.6 RN2483- und UART-Methoden und ihre Funktionalität

8.6.1 RN2483.c

- **RN2483_init()**: Initialisiert / Konfiguriert das RN2483 Modul wie im Codebeispiel 9.1 zu sehen ist.
- **RN2483_sendData(char *s)**: Konvertiert mit sendHex2(s) das gewünschte Char-Array in Hexadezimal ASCII Code und sendet es mit der UART-Schnittstelle an das Modul, welches anschließend die Nachricht über LoRaWAN an das Gateway schickt.
- **RN2483_sendCmd(char *cmd)**: Sendet ein Kommando über die UART-Schnittstelle an das Modul.
- **RN2483_prepareMessage(int fillRatio)**: Konvertiert einen Integer Wert in Hexadezimal ASCII Code und sendet es über die UART-Schnittstelle an das Modul, welches anschließend die Nachricht über LoRaWAN an das Gateway schickt.
- **delayFunction(int number)**: Führt einen Delay aus, da bei uns maximal nur 850ms am Stück mit der Delay Funktion gewartet werden kann. Bei Parameter-Wert 1 wird 3400ms gewartet, bei 2 wird 500ms gewartet, bei 3 wird 300ms gewartet und bei 4 wird 1700ms gewartet.

8.6.2 uart.c

- **init_uart()**: Initialisiert die UART-Schnittstelle, d.h. setzt die Baudrate, aktiviert die RX und TX Pins und setzt den asynchronen UART Modus mit 8 Data bits, no parity, 1 stop bit.
- **uart_putc(unsigned char c)**: Sendet einen Buchstaben über UART, wenn die Schnittstelle bereit ist.
- **sendString(char tempStringChar[])**: Sendet ein Char-Array über UART, indem es für jeden Char die uart_putc() Methode aufruft.
- **sendHex2(char *s)**: Konvertiert das gegebene Char-Array in Hexadezimal ASCII Code und sendet es über die sendString() Methode.

9 Application Server

9.1 Visualisierung der empfangenen Daten mit Angular CLI

Das Front-End ist die Ansicht, auf welche der Kunde blickt, um die Füllstände aller Behälter zu sehen. Es ist minimalistisch gehalten, sodass keine Überladung entsteht und das menschliche Auge die relevante Information schneller erkennt.

Wie auf dem Bild zu sehen ist, gibt es zwei Seiten, auf die man blicken kann. Auf der Haupt- / Startseite (Dashboard) kann man die Zylinder mit Ihren Inhalten sehen. Jeder aktive Node wird hier mit ihren aktuellen Füllständen angezeigt. Die zweite Seite (Statistiken) zeigt auf einem größeren Diagramm die historischen Werte bis zum aktuellen Zeitpunkt an. Die Historischen Werte zeigen jedoch nur die Daten des Vortags bis zum neuesten Stand an.

Die Visualisierung ist mithilfe des Front-End-Frameworks Angular CLI geschrieben worden. Die Programmiersprache von Angular ist TypeScript, eine Weiterentwicklung von JavaScript, mit einigen Verbesserungen wie die Typisierung von Variablen. Weiterhin wurden die Zylinder-Behälter mit der „AmCharts“ Angular-Bibliothek implementiert. Diese Bibliothek bietet die Möglichkeit diese Diagramme mit Hilfe einer „create“ Funktion zu erstellen und einer „update“ Funktion mit neuen Daten zu versorgen und damit das Diagramm in Echtzeit zu verändern. Darüber hinaus wurden die Statistikdiagramme mit Hilfe der gleichen Bibliothek erstellt. Diese Diagramme bieten die gleichen Funktionen wie die Zylinderdiagramme.

9.2 NodeJS Server für die Verarbeitung der Datenpakete

Der Server ist in der Programmiersprache NodeJS geschrieben. Darin wurde ein Express Server implementiert. Dieser Express Server nutzt ein WebSocket, um die Verbindung zwischen Server und Client herzustellen. Diese Verbindung wird von dem NPM (Node Package Manager) Modul „ttn“ genutzt. Hierbei ist ttn eine Bibliothek des Networkservers „The Things Network“. Durch das ttn Modul bekommt der Server die Nachrichten vom Network Server und damit kann das Paket verarbeitet und per WebSocket an den Client weitergeleitet werden. Während der Weiterverarbeitung erfolgt auch die Speicherung einkommender Daten.

9.3 TTN

Das TTN Modul (SDK) ist die Schnittstelle zwischen dem Express Server und dem Network Server. Die Verbindung wird automatisch hergestellt. Damit jedoch eine Verbindung hergestellt werden kann, müssen im Voraus zwei Werte vorhanden sein. Der erste ist die Application ID. Diese ID beschreibt das Projekt, welches vorher bei der TheThingsNetwork-Webseite festgelegt wurde. In diesem Fall ist die Application ID: lora_measure_network. Der zweite Wert ist der accessKey. Dieser ist für die Authentifizierung des SDK's mit dem Network Server erforderlich. Der Key muss vorher über die Webseite generiert werden. Die Nachrichten, zwischen Network Server und SDK, sind in einem bestimmten Format:

```
{  
    app_id: 'lora_measure_network',  
    dev_id: 'rn2483_node1',  
    hardware_serial: '0004A30B001C6A30',  
    port: 1,  
    counter: 0,  
    payload_raw: <Buffer 33 30>,  
    metadata: { time: '2018-07-05T18:04:13.482082016Z' },  
    message: '30'  
}
```

Listing 9.1: Payload

- **app_id:** Application ID
- **dev_id:** Das Gerät, welches die Nachricht gesendet hat
- **hardware_serial:** Hardwarenummer des Nodes
- **counter:** Messagecounter der bei jeder Nachricht um 1 inkrementiert
- **payload_raw:** Die Nachricht wird vom Network Server als Hexadezimal
- **metadata:** In diesem Objekt ist ein Zeitstempel enthalten. Dieser Zeitstempel ist von dem Zeitpunkt, als die Nachricht im Network Server angekommen ist.
- **message:** Das ist die Payload, welche vom payload_raw geparsst wurde

9.4 Inaktivität

Sobald ein Gerät inaktiv wird (d.h. eine Stunde ohne Nachrichtenempfang), ist es für die Nutzer in der UI nicht mehr möglich, das Gerät zu sehen. Diese Inaktivität würde bei einem Thermobehälter die Abkühlung bedeuten. Das Limit basiert auf dem Resultat einer Gesprächsrunde innerhalb der Entwickler. Sobald jedoch das Gerät wieder sendet, wird es in der UI eingeblendet. Dieses Limit hat den Hintergrund, dass ein Gerät unerwartet vom Netz genommen wird. Wenn dieser Fall eintreten würde, könnte man in der Visualisierung nicht erkennen, ob das Gerät überhaupt noch existiert.

9.5 Voraussetzungen und Starten des Node-Servers

Unterstützte Betriebssystem

- Windows
- Linux
- Unix

Benötigte Software

- NodeJS (inkl. NPM)
- Angular

Nutzung

- Starte zwei Kommandozeilenprogramme
 - Konsole 1 im Ordner „server“: node server.js
 - Konsole 2 im Ordner „client“: ng serve -open

10 Schaltplan und Verlöten der Platine

Der Schaltplan wurde mit dem Programm Fritzing (<http://fritzing.org/home>) erstellt.

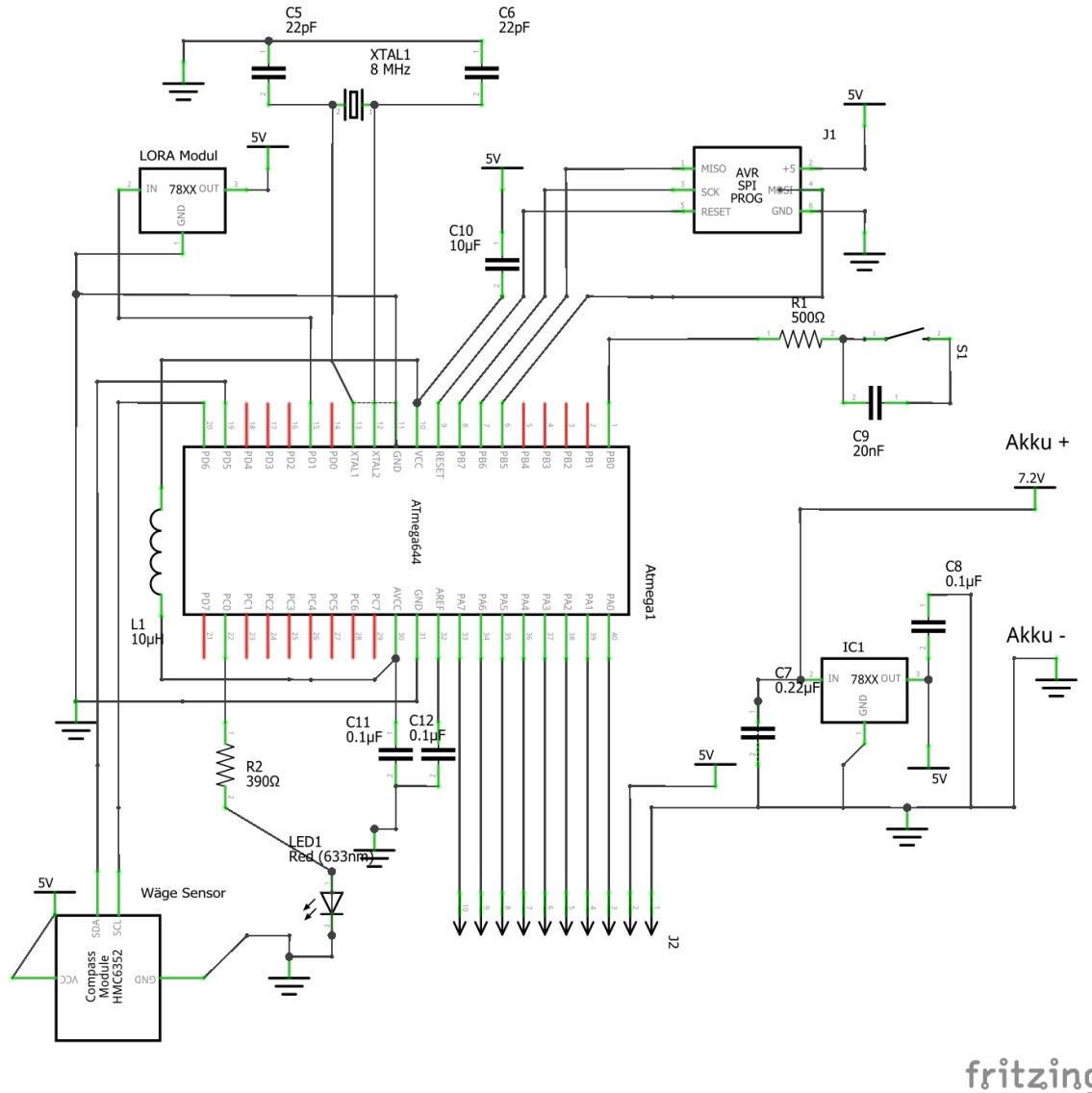


Abbildung 10.1: Schaltplan

fritzing

Als Mikrocontroller haben wir im Schaltplan einen ATMega644 statt einem ATMega644PA verwendet, da diese fast gleich sind. Ausgehend vom Mikrocontroller werden an den Pins XTAL1 und XTAL2 der Quarz angeschlossen. An Pin PA0 bis PA7 werden die Pins für das LCD-Display angeschlossen. An PC0 wird eine rote LED und davor der Widerstand R2 mit 390 Ohm angeschlossen. Der Wäge Sensor wird mit den Pins PD5 und PD6 angeschlossen. Hierbei wurde für den Sensor einfach ein Platzhalter mit vier Pins genommen, da kein passendes Teil für unsere Wägezelle gefunden wurde. Das LoRa-Modul wird an PD1 angeschlossen und es wurde ebenso eine Platzhalterkomponente mit drei Pins verwendet. Zudem haben wir einen Taster auf PD0 und an PD5 – PD7 einen ISP-Anschluss angelötet. Die Platine und dessen Bauteile werden mit zwei in Reihe geschalteten Akkus versorgt die jeweils ca. 3,6V liefern. Dazwischen geschaltet ist ein DC-DC Wandler, der den Stromkreis mit konstanten 5V versorgt. Dann haben wir anhand unseres Schaltplans unsere Platine verlöten.

Unsere Platine sieht im Moment so aus:

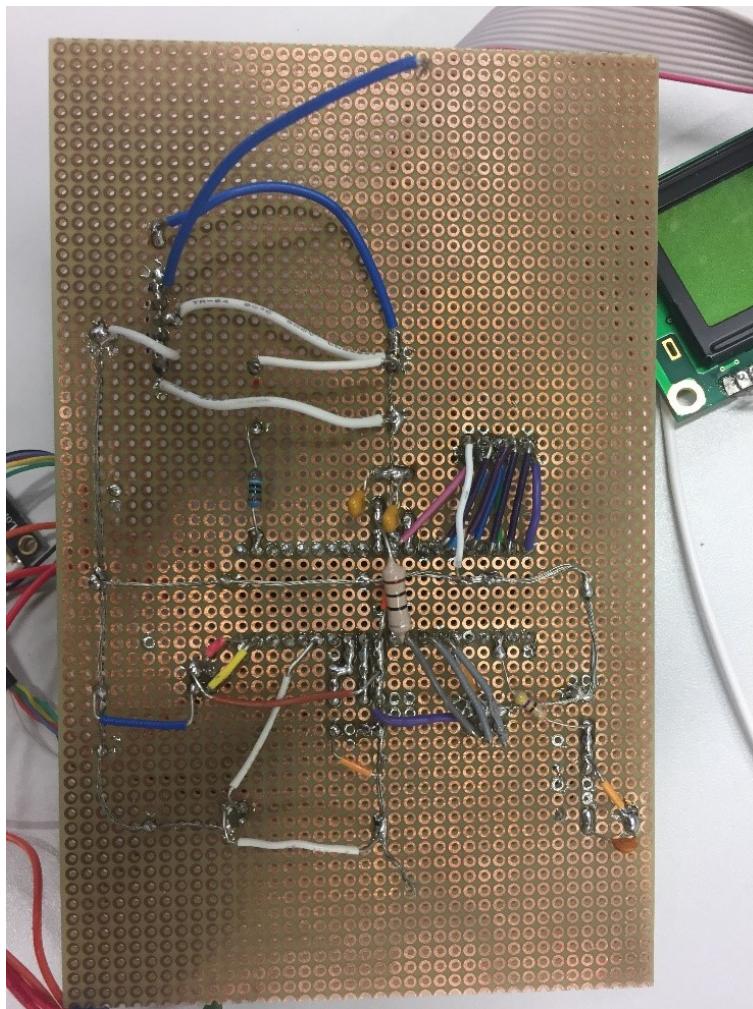


Abbildung 10.2: Rückseite der verlötenen Platine unseres ersten Prototyps

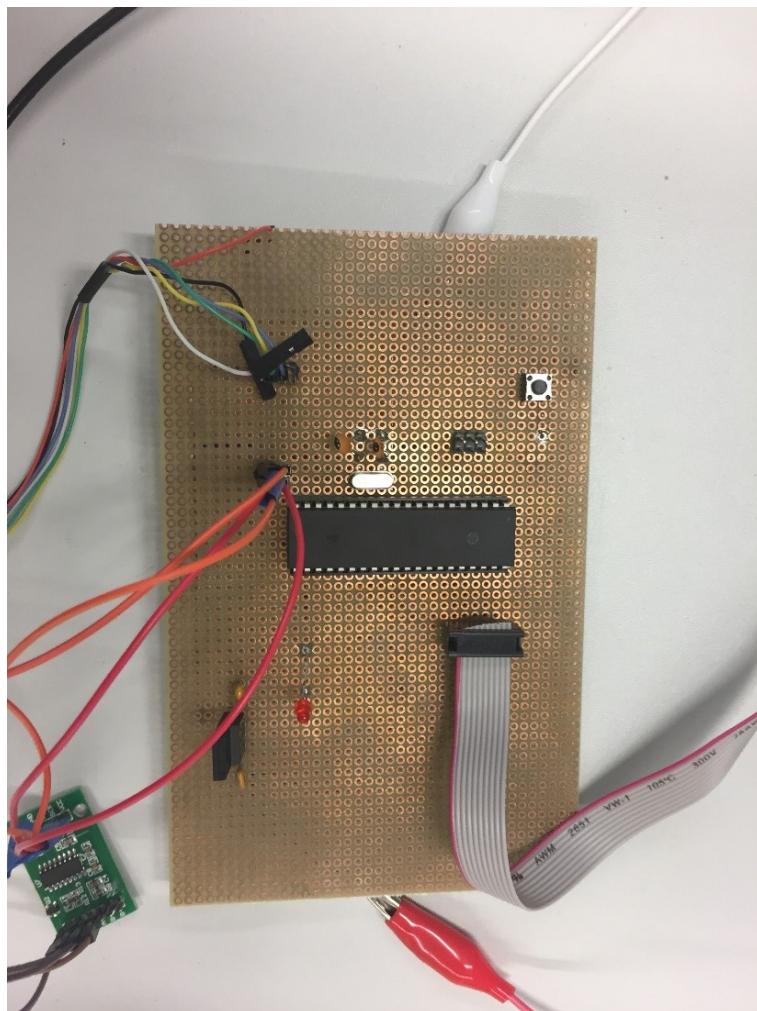


Abbildung 10.3: Vorderseite der verlöteten Platine unseres ersten Prototyps

Leider ist unser Taster im Moment nicht funktionsfähig und es kann somit nur durch aus und einschalten das Gewicht genullt werden. Des Weiteren kann für spätere Verwendungszwecke das Display wieder abgenommen werden, um somit Platz und Strom zu sparen. Es dient zurzeit nur zur Visualisierung des Gewichtes, ist aber für die Funktionsweise nicht von Bedeutung.

11 Ausblick

11.1 LoRaWAN Modul Code verbessern

- Firmware vom **RN2483** updaten
- Gebrauch von der sleep Funktion „sys sleep <length>“ machen um Strom zu sparen
- Beim Ausschalten den Frame-Counter mit „mac get upctr“ auslesen und in den EEPROM vom Atmega644PA speichern. Beim Einschalten den Frame Counter aus dem EEPROM auslesen und mit „mac set upctr <fCntUp>“ wieder neu setzen, damit man nicht jedes Mal den Frame Counter in der Konsole von The-ThingsNetwork zurücksetzen muss.
- Empfang von Nachrichten oder Befehlen vom Gateway ermöglichen und durch UART-Receive diese Nachrichten bzw. Befehle im Mikrochip auslesen.
- Delays mit Interrupts ersetzen, so dass das Programm weiterarbeiten kann, während es auf ein Ereignis wartet.
- Verschiedene Profile oder Modis passend zu den Spreading Factors programmieren z.B.: niedrige Reichweite + hohe Bandbreite, hohe Reichweite + niedrige Bandbreite, etc.

11.2 Front-End Lizenzkosten vermeiden

Das Frontend hat das Problem, dass es die ÄmCharts "Bibliothek nutzt. Die Bibliothek ist für Open-Source und nicht kommerzielle Projekte frei verfügbar. Die Lizenz (Stand 06.07.2018) kostet für ein Jahr 4590 €. Darin sind die Diagramme in den Seiten Dashboard und Statistik enthalten. In Zukunft sollen selbstgeschriebene Diagramme verwendet werden, um Lizenzkosten zu vermeiden.

11.3 Erstellen eines wasserfesten Gehäuses

In naher Zukunft ist es außerdem noch wichtig, dass sich die Platine und die nötige Hardware in einem Gehäuse befinden, da es sonst nicht im Kommerziellen nutzbar wäre. Zudem sollte das Gehäuse wasserfest sein, da sonst Kaffee oder ähnliches hindurchkommen und etwas beschädigen könnte.

Dadurch wird unser Produkt deutlich kompakter und nutzvoller für Hotels. Unser erster Ansatz wäre ein Prototyp mit einem 3D-Drucker zu erstellen.

Literatur

- Electronics, Drotek. *DataLink LoRa RN2483*. <https://drotek.com/shop/en/lora/763-data-link-lora-rn2483.html>.
- Electronics, Mouser. „Analog-to-Digital Converter“. https://www.mouser.com/ds/2/813/hx711_english-1022875.pdf.
- Inc., Microchip Technology. 2015-2017a. *Low-Power Long Range LoRa Technology Transceiver Module*. <http://ww1.microchip.com/downloads/en/DeviceDoc/50002346C.pdf>.
- . 2015-2017b. *RN2483 LoRa Technology Module Command Reference User's Guide*. <http://ww1.microchip.com/downloads/en/DeviceDoc/40001784F.pdf>.
- Kaiser, Reto. *ic880a Gateway*. <https://github.com/ttn-zh/ic880a-gateway/wiki>.
- LoRa(WAN) airtime calculator*. <https://docs.google.com/spreadsheets/d/1QvcKsGeTPPr9icj4XkKXq4r2zTc2j0gsHLrnplzM3I/edit#gid=0>.
- mstanley. *LoRa WAN*. <http://www.mstanley.co.uk/blog/wp-content/uploads/2015/11/Enabling-world-wide-mobility-for-the-IoT-image-2.jpg>.
- Network, The Things. *LoRaWAN Security*. <https://www.thethingsnetwork.org/docs lorawan/security.html>.
- Sparkfun. *Gewichtsmessung*. <https://learn.sparkfun.com/tutorials/getting-started-with-load-cells>.
- Wikipedia contributors. 2018a. *Force-sensing resistor – Wikipedia, The Free Encyclopedia*. [Online; accessed 13-August-2018]. https://en.wikipedia.org/w/index.php?title=Force-sensing_resistor&oldid=846591724.
- . 2018b. *Load cell – Wikipedia, The Free Encyclopedia*. [Online; accessed 13-August-2018]. https://en.wikipedia.org/w/index.php?title=Load_cell&oldid=851413016.
- . 2018c. *Piezoelectric sensor – Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Piezoelectric_sensor&oldid=839026087. [Online; accessed 13-August-2018].
- Witekio. *LoRa WAN spreading factor*. [https://witekio.com/wp-content/uploads/2018/01/lora-wan-spreading-factor.png](http://witekio.com/wp-content/uploads/2018/01/lora-wan-spreading-factor.png).