

# Entwicklung eines portablen Wägesystems mit Aufzeichnungsfunktion

**Studienarbeit**

**für das 3. Studienjahr**

im Studiengang Informatik

an der Dualen Hochschule Baden-Württemberg Heidenheim

vorgelegt von:


Johannes Fahr

Marcel Vidmar

## Erklärung

Wir versichern hiermit, dass wir unsere Studienarbeit mit dem Titel „Entwicklung eines portablen Wägesystems mit Aufzeichnungsfunktion“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden.

Aalen / Rosenberg, den 15.07.2022

  
\_\_\_\_\_

Johannes Fahr / Marcel Vidmar

## Gender Erklärung

Um die Arbeit besser lesbar zu gestalten, wird in der vorliegenden Arbeit auf die gleichzeitige Verwendung männlicher/weiblicher Sprachformen verzichtet. Es wird das generische Maskulinum verwendet wobei dies Geschlechtsneutral wahrzunehmen ist.

## **Abstract**

Das Ziel der Arbeit mit dem Titel „Entwicklung eines portablen Wägesystems mit Aufzeichnungsfunktion“ ist es eine Waage zu entwickeln die mit einer hohen Frequenz Wägungen durchführt und die Werte dieser speichert. Die Messung soll mit wenig Aufwand durchführbar sein auch die Daten sollen einfach auslesbar sein.

The aim of this work titled "Development of a portable weighing system with a recording function" is to develop a scale that weighs with a high frequency and stores the measurements. The measuring-process should be possible with little effort and the data should be easily accessible.

Erklärung .....	ii
Gender Erklärung .....	ii
Abstract .....	iii
Abbildungsverzeichnis .....	v
Tabellenverzeichnis .....	v
Abkürzungsverzeichnis .....	v
1 Einleitung .....	1
1.1 Anforderungen .....	1
1.2 Übersicht Komponenten .....	2
1.3 Benötigtes Werkzeug .....	8
2 Prototyp .....	9
2.1 Proof of Concept .....	9
2.2 Erster Gehäuse-Entwurf .....	10
2.3 Bauanleitung .....	12
3 Bedienung .....	14
3.1 Durchführung Messung .....	14
3.2 Umgang mit den Daten der Messung .....	15
3.3 Uhrzeit einstellen .....	16
3.4 Austausch Arduino .....	16
4 Probleme .....	17
4.1 HX711-01 .....	17
4.2 Kalibrierung .....	19
4.3 Lose Kabelverbindungen .....	20
4.4 SD-Karten-Modul .....	21
4.5 Spannungsversorgung .....	21
4.6 Ungenauigkeit der RTC .....	21
4.7 Schwankungen in der Messfrequenz .....	22
5 Fazit .....	23
Quellcode .....	24
WaegeProgramm.ino .....	24
SetTime.ino .....	27
Literaturverzeichnis .....	28

## Abbildungsverzeichnis

Abbildung 1 HX711-01 (2) .....	3
Abbildung 2 Wägezelle und altes HX711-01 Modul (3) .....	4
Abbildung 3 Wägezelle mit DMS (4) .....	4
Abbildung 4 RTC Echtzeituhr (6) .....	5
Abbildung 5 SD-Karten Lesemodul (7) .....	6
Abbildung 6 Arduino Nano (8) .....	7
Abbildung 7 Schalter mit RGB Ring (9) .....	6

## Tabellenverzeichnis

Tabelle 1 Übersicht der verwendeten Komponenten .....	2
---	---

## Abkürzungsverzeichnis

AWG	american wire gauge
CAD	computer aided design
CSV	character separated values
DC	direct current
DMS	Dehnungsmessstreifen
FS	Full Scale
IDE	integrated development environment
LED	light-emitting diode
PC	Personal Computer
RGB	Red, Green, Blue Farbraum
RTC	real time clock
SD	secure digital (memory card)
USB	universal serial bus

# 1 Einleitung

Ein Interesse dafür, das Gewicht von Gegenständen zu bestimmen besteht bei Menschen schon sehr lange. Laut LEIFIphysik soll es die ersten Waagen schon vor zehntausend Jahren, im alten Ägypten, gegeben haben. Diese Waagen waren sehr primitiv und auch ohne Eichung von Gewichten, dementsprechend konnten Gewichte nur direkt untereinander verglichen werden und es gab keine klaren Einheiten wie zum Beispiel Kilogramm. Zum Vergleich wurden Getreidekörner oder Steine gewählt je nach Gewicht des zu messenden Objekts. (1)

Waagen in der heutigen Zeit, ob mechanisch oder elektronisch sind aufgrund der stetig voranschreitenden Technik sehr genau. Die Waage, die aus dieser Projektarbeit entstanden ist, kann zwar auch Gewichte relativ genau wägen, jedoch ist der Hauptfokus die Geschwindigkeit der Wägungen und die Aufzeichnung dieser. Innerhalb von diesem Kapitel, werden die einzelnen verwendeten Komponenten und auch die allgemeinen Anforderungen an die Waage beschrieben.

## 1.1 Anforderungen

Die Anforderungen entstammen der Projektbeschreibung und der Besprechung zum Projekt. Es soll eine Waage entwickelt werden, die Objekte bis zu einer Masse von etwa 1kg, mit einer hohen Messgeschwindigkeit (mehr wie 50Hz) misst. Die Messwerte sollen auf einem portablen Speichermedium gespeichert werden, sodass die gesammelten Daten auf einem PC ausgelesen und weiterverarbeitet werden können. Der Nutzen dieser Waage soll es sein, Änderungen der Gewichtskraft eines darauf gestellten Pendels oder auch Beschleunigungskräfte während einer Fahrt im Aufzug zu messen. Eine absolute Genauigkeit ist also nicht gefordert, es reichen sichtbare Änderungen der Messwerte.

## 1.2 Übersicht Komponenten

Artikel	Preis	Anzahl	Artikelnummer	Ausführung
<a href="#">HX711 und Wägezelle</a>	6,00 €	1	DEBO HX711-01	
<a href="#">HX711**</a>	6,99€	1		
<a href="#">Arduino Nano</a>	17,85 €	1	ARDUINO NANO	
<a href="#">RTC Echtzeituhr*</a>	2,99€	1	DS1307	
<a href="#">SD Karten Leser</a>	0,77€	1		SD-Card-Module
<a href="#">Batterieanschluss 9v-block</a>	0,24€	1	CLIP 9V-T	
	1,99 €	1	VAR MT 9V	
<a href="#">RGB-LED***</a>	0,39€	1	LED LL 5-8000RGB	
<a href="#">19mm Schalter</a>	2,83€	1		RGB, 5V, RingLight,Latching
<a href="#">DC/DC-Wandler (LM2596)</a>	0,5€	1		
<a href="#">SD-Karte*</a>	4,49€	1		4GB

Tabelle 1 Übersicht der verwendeten Komponenten

Anmerkung: Sowohl die LED, Batterie, Widerstände und auch andere billige Bauteile, können kaputt gehen, weshalb es sinnvoll ist, hiervon mehrere zu kaufen. Die Artikel können auch bei anderen Anbietern, unter anderem AliExpress bestellt werden, um Geld zu sparen. Die Liste wurde zu Beginn des Projekts geschrieben, deshalb kann es gut sein, dass Artikel nicht mehr vorhanden sind oder Preise sich stark unterscheiden.

\* Artikel sind nicht die tatsächlich verwendeten, erfüllen aber denselben Zweck. Die tatsächlich verwendeten Bauteile sind teilweise Überreste aus alten Projekten.

\*\*Bauteil war zwar schon mit der Wägezelle Kombi vorhanden, allerdings kam es zu einem Problem bei hohen Messfrequenzen. Weshalb ein neues Bauteil, was die gewünschten 80 Hz unterstützt, benötigt wurde. Problem wird später genauer noch beschrieben im Kapitel [Probleme->HX711-01](#)

\*\*\*LED wurde durch einen Schalter mit eingebauter RGB ersetzt, dies macht aber nur einen visuellen Unterschied und gibt dem Benutzer Feedback bei der Bedienung.

### 1.2.1 HX711-01

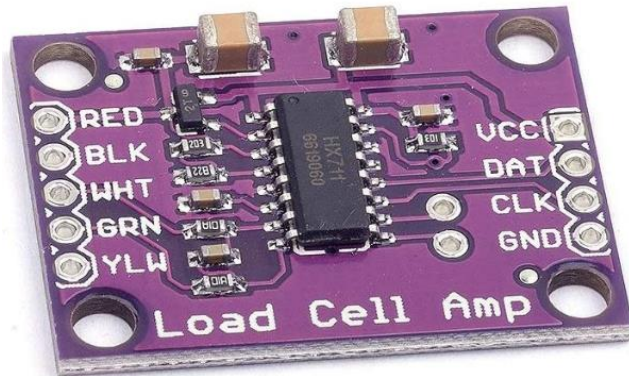


Abbildung 1 HX711-01 (2)

Das Verstärker Board erhält die Daten der Wägezelle und formatiert diese um. Die formatierten Daten werden an den Arduino Nano weitergeben. Das Board kann entweder mit 10 Hz oder 80 Hz betrieben werden, mehr hierzu im Kapitel [Probleme->HX711-01](#).



## 1.2.2 Wägezelle



Abbildung 2 Wägezelle und altes HX711-01 Modul (3)

Eine Wägezelle besteht aus einem Federkörper der mehreren DMS<sup>1</sup> auf sich befestigt hat, durch eine Streckung dieser, ändert sich der Widerstand. Anhand von der Widerstandsänderung lässt sich nun die Belastung bestimmen. (5)

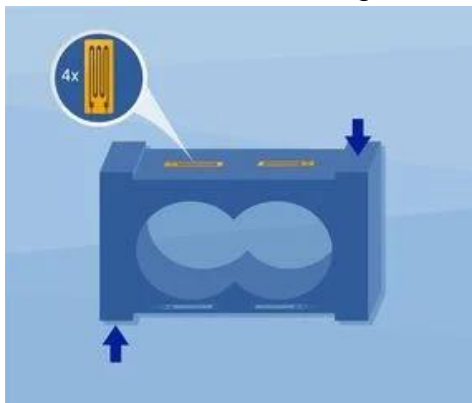


Abbildung 3 Wägezelle mit DMS (4)

Die Wägezelle muss kalibriert werden anhand von einem Körper dessen Gewicht bekannt ist. Diese Kalibrierung sollte auch das Gewicht von der Plattform beinhalten. Die ausgewählte Wägezelle kann Gewichte bis zu 1 Kg mit einer Genauigkeit von  $\pm 0,02\%$  FS<sup>2</sup> bestimmen. Es kann im 80Hz Modi jedoch höhere Ungenauigkeiten entstehen.

---

<sup>1</sup> auf Folie befindet sich ein elektrischer Leiter

<sup>2</sup> Full Scale steht für die Abweichung des gemessenen Wertes im Vergleich zum tatsächlichen Wert.

### 1.2.3 RTC Echtzeituhr

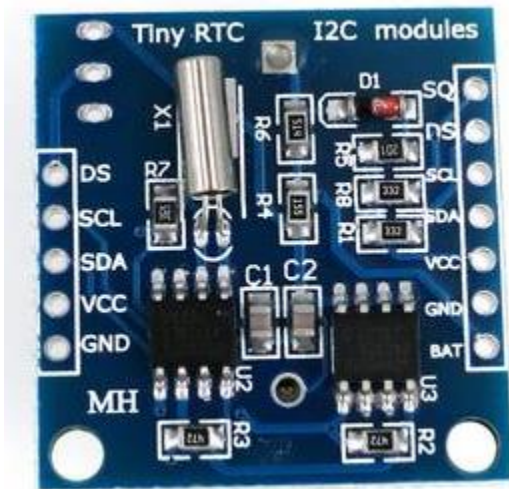


Abbildung 4 RTC Echtzeituhr (6)

Die Echtzeituhr läuft mittels einer Knopfzellen-Batterie, somit kann auch wenn der Arduino ausgeschaltet ist, mit einer gewissen Genauigkeit (Abhängig vom Modell und Quartz) die Zeit vorgeben werden. Sollte die vorhandene Zeit zu sehr von der tatsächlichen abweichen kann die Zeit mit einem Programm auch einfach überschrieben werden. Wie genau dies funktioniert ist [hier](#) beschrieben. Benötigt wird diese Uhr, um beim Erstellen von Messungen einen Startpunkt zu vergeben. Dieser wird auch benutzt, um den Dateinamen anzugeben, so kann man erkennen, welche Messung wann gemacht wurde.

### 1.2.4 SD-Karten Leser



Abbildung 5 SD-Karten Lesemodul (7)

Die durchgeführten Messungen müssen gespeichert werden und auch einfach übertragbar sein, um diese auf z.B. einen Computer zu übertragen. Diese Daten können dann dort weiter verarbeitet/analysiert werden. Eine SD-Karte ist als Speichermedium hier aufgrund der kleinen Größe sehr gut geeignet. Die Größe der Speicherkarte bestimmt die Summe der Dauer der einzelnen Aufzeichnungen. Die Maximale Messdauer bezogen auf die Speicherkarte ist [hier](#) berechnet. Die aktuell verwendete SD-Karte ist 4GB groß, kann aber jederzeit durch eine Karte mit mehr Speicherkapazität ausgetauscht werden.

### 1.2.5 Schalter



Abbildung 6 Schalter mit RGB Ring (9)

Der Schalter wird zum Starten der Aufnahme verwendet, zusätzlich wird durch die Farbe des RGB Rings, der aktuelle Zustand ausgegeben. Durch die Ausgabe eines Status wird die Bedienung der Waage deutlich Benutzerfreundlicher.

### 1.2.6 Arduino Nano

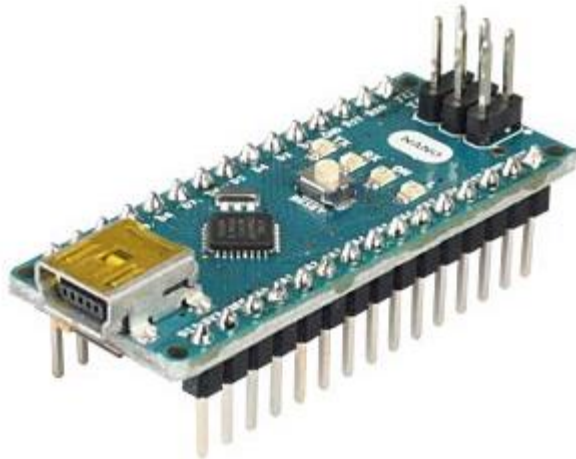


Abbildung 7 Arduino Nano (8)

Der Arduino Nano ist das Herzstück der Wage. Dieser Mikrocontroller verbindet alle Komponenten miteinander und führt den gespeicherten Quellcode aus, um die gewünschten Funktionen zu erzielen. Programm kann über USB abgespeichert und debuggt werden. Im Rahmen dieses Projekts wird ein Arduino Nano mit USB-C-Buchse verwendet. Der Quellcode als auch eine grobe [Anleitung](#) wie Quellcode auf Arduino initialisiert wird, ist innerhalb des Dokuments vorhanden.

### 1.2.7 Spannungswandler



Abbildung 8 Spannungswandler (10)

Da die Spannung der 9V Batterie für den Arduino zu groß ist, wird ein Spannungswandler benötigt, welcher wie der Name schon sagt die Spannung auf 5V wandelt.

### **1.2.8 Kabel und Stecker**

- 26 oder 28 AWG-Kabel, idealerweise in verschiedenen Farben
- Dupont-Stecker
- 2x Dupont Buchsenleiste, 7 Pole
- Flachstecker weiblich, 2,8mm und 4,8mm Steckbreite

### **1.3 Benötigtes Werkzeug**

- LötKolben und LötZinn: Manche Verbindungsstellen müssen gelötet werden. Außerdem wird der Aufbau deutlich kleiner und kompakter, wenn die Teile nicht mit einem Breadboard verbunden werden müssen.
- PC/Laptop: Der Quellcode muss auf den Nano übertragen werden und auch zum Testen ist es ratsam, die Daten zum Beispiel zu plotten.
- 3D-Drucker: Das entwickelte Gehäuse wurde in einem CAD-Programm entwickelt und danach mittels 3D-Drucker gedruckt. Alternativ kann auch ein eigenes Gehäuse, aus beispielsweise Holz oder Acrylglas, entwickelt werden.
- SD-Karte-Adapter: Die Aufnahme der Messung wird auf einer SD-Karte gespeichert. Diese muss, um die Ergebnisse auswerten zu können, mittels eines Adapters an einem Computer ausgelesen werden.
- Crimp-Zange für Dupont-& Flachstecker: Um das System simpler zu gestalten werden viele Verbindungen gesteckt statt verlötet.

## 2 Prototyp

Im Laufe der Entwicklung durchschritzt das Wägesystem mehrere Entwicklungen bzw. Iterationen, bevor die finale Version entstanden ist. Ein paar dieser Prototypen werden innerhalb von diesem Kapitel beschrieben.

### 2.1 Proof of Concept

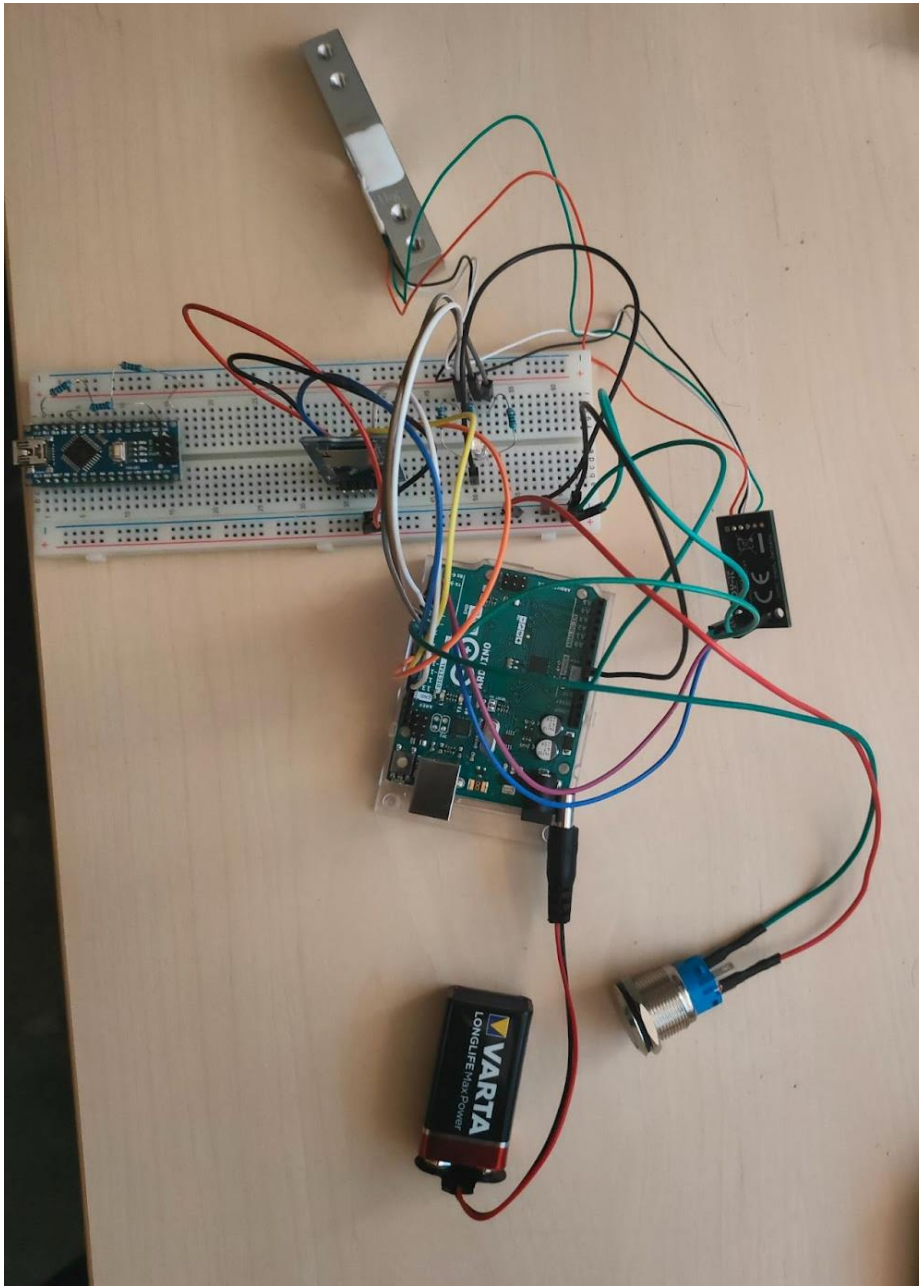


Abbildung 9 Erster Prototyp nach dem Proof of Concept Prinzip

Waage nach dem Proof of Concept Prinzip, Komponenten sind lose verbunden, um Funktion aller Elemente testen zu können und das Programm zu schreiben. Einige Funktionen fehlen hier noch wie zum Beispiel die Real-Time-Clock.

### 2.1.1 Funktionen

- Messwert wird von der Wägezelle erhalten.
- Messwert und die verstrichene Zeit zum nächsten Messwert werden auf die SD-Karte gespeichert.
- Knopf zum Starten/Beenden der Aufnahme.
- LED gibt den Aufnahmestatus wieder (Grün: Bereit, Blau: Aufnahme läuft, Rot: Störung).

### 2.1.2 Benötigte Änderungen

- Verwendung interner Uhr: Um Dateien leichter identifizieren zu können soll der Start-Zeitpunkt einer Aufnahme als Dateiname verwendet werden, dies ermöglicht auch das einfache Speichern von mehreren Messungen.
- Austausch von Arduino UNO durch Arduino Nano, um möglichst platzsparend zu agieren und kleineres Gehäuse zu ermöglichen.
- Kabel müssen noch direkt gelötet oder gecrimpt werden, um Platz zu sparen und um zu gewährleisten, dass Kabel sich nicht einfach lösen.
- Schalter für das ein/ausschalten des Arduino.
- Aufnahmestatus direkt über RGB Ring am Schalter realisieren anstelle der extra LED.
- Einbauen eines Spannungswandler, um Komponenten mit 5V zu versorgen, statt der Verwendung der im Arduino eingebauten Spannungswandlung. Warum dies relevant ist, wird im Kapitel [Spannungswandlung](#) beschrieben.
- Wechsel auf anderes HX711-01 Modul, um der Anforderung bezüglich der Messfrequenz zu entsprechen. Begründung dafür [hier](#).

## 2.2 Erster Gehäuse-Entwurf

Zunächst werden alle Komponenten im CAD-Programm<sup>3</sup> modelliert. Hierzu werden wichtige Dimensionen wie die äußeren Abmaße und Größe sowie Position der Montagelöcher gemessen und modelliert. Wenn alle Komponenten modelliert sind, können diese frei im virtuellen Raum verschoben werden, um eine optimale Anordnung bezüglich Ordnung und Platz zu erreichen. Hierbei muss bedacht

---

<sup>3</sup> Verwendet wurde hier Fusion 360, da dies kostenlos ist und die Verwendung bereits vertraut ist.



werden, dass die Komponenten auch verkabelt werden müssen und das Gehäuse druckbar bleiben soll. Das bedeutet, dass Überhänge vermieden und Wandstärken passend gewählt werden müssen.

Nach dem Drucken des Ersten Prototyps und der Montage der Wägezelle ist ein zuvor unbedachtes Problem aufgetreten. Die (auf dem Bild rechte) Wand wölbt sich bei einem Kraftaufwand auf die Wägezelle nach innen. Dies verfälscht das Messergebnis und eine zweite Version muss entwickelt werden, bei dieser wurde eine interne Verstärkung unter der Wägezelle eingefügt. Diese hat ein Loch in der Mitte, um die Kabelführung zu ermöglichen. Auch ein Mechanismus zur Sicherung der 9V-Batterie wurde implementiert. Diese neu Version ist deutlich stabiler bei Belastungen. Mit dieser Version wurde auch die extra LED zur Anzeige des Status durch den RGB Ring an einem Schalter ersetzt.

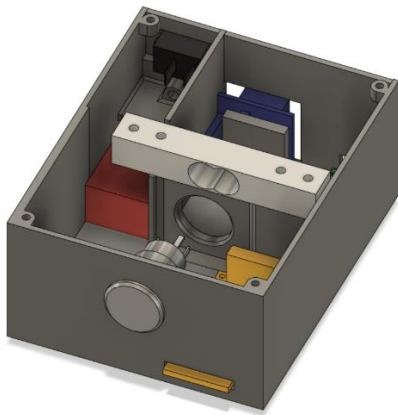


Abbildung 10 Vorderseite des zweiten gedruckten Prototyps

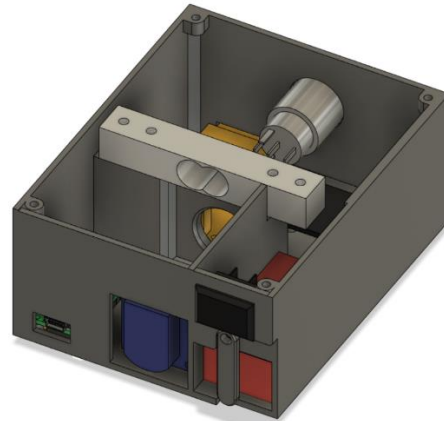


Abbildung 11 Rückseite des zweiten gedruckten Prototyps

Mittig befindet sich der **runde Knopf** zum Starten und Beenden der Aufnahme, welcher ab dieser Version auch den Status darstellt, rechts darunter das **Modul für die SD-Karte**.

Zu sehen ist hier die Sicherung der **Batterie** unter dem **An/Aus-Schalter**, der USB-Port des Arduino links sowie die **RTC** in der Mitte.



## 2.3 Bauanleitung

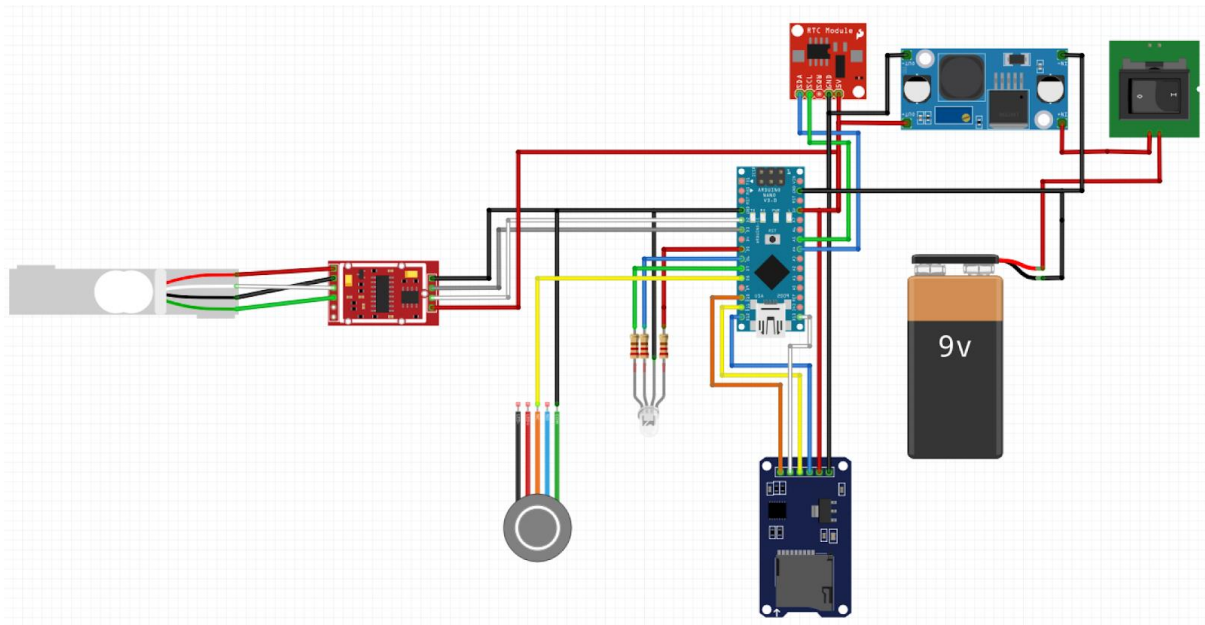


Abbildung 12 Anschlussplan des Wägesystems über Fritzing

Fritzing, welches als Tool zur Erstellung des Anschlussplans benutzt wurde bietet leider keinen RGB-Schalter wie er verwendet wird. Deshalb ist in der obigen Darstellung der Schalter und die LED getrennt dargestellt, wie es auch im ersten Entwurf vorgesehen war. Aufgrund der Einschränkungen des Programms wird darunter in der Tabelle alle Komponenten und deren Anschlüsse mit den jeweiligen Pins und Kabelfarben genau aufgelistet.

Beim Zusammenbauen hat es sich bewährt, Dupont-Pin-Leisten für Masse sowie 5V zu verwenden. Dies ermöglicht einen modularen Aufbau und ist deutlich platzsparender als beispielsweise Wagoklemmen. Die einzelnen Bauteile bis auf den DC/DC-Spannungswandler werden in das Grundgehäuse geschraubt. Verwendet werden verschiedene Schrauben der Dimensionen M2, M2,5, M3 sowie M4 für die Wägezelle. Für die interne Verkabelung wurde unter der Wägezelle in der Verstrebung ein Loch vorgesehen.

Von Gerät	Pin	Kabelfarbe	Zu Gerät	Pin
Arduino	D2	Grau	HX711	Clk
	D3	Weiß		Dat
			Druckschalter	
	D5	Gelb		
	D6	Grün		
	D7	Blau		
	D8	Schwarz		
			SD-Modul	
	D10	Braun		CS
	D11	Rot		MOSI
	D12	Gelb		MISO
	D12	Orange		SCK
			RTC	
	A4	Grau		SDA
	A5	Lila		SCL
	Gnd	Orange	Druckschalter	Gnd
	Gnd	Schwarz	SD-Modul	Gnd
9V-Batterie	+	Rot	An/Aus-Schalter	Pin1
An/Aus-Schalter	Pin2	Rot	DC-DC-Wandler	In+
Common-Rail Gnd		Schwarz	Druckschalter	
		Schwarz	SD-Modul	Gnd
		Schwarz	RTC	Gnd
		Schwarz	DC-DC-Wandler	In-
		Blau		Out-
		Schwarz	HX711	Gnd
		Schwarz	9V-Batterie	-
Common-Rail 5V		Weiß	SD-Modul	5V
		Weiß	RTC	VCC
		Lila	HX711	VCC
		Rot	DC-DC-Wandler	Out+
		Rot	Arduino	5V

Abbildung 13 Bild von tabellarischer Beschreibungen der Kabelverbindungen

## 3 Bedienung

Wie die Waage bedient wird, ist innerhalb des Kapitels beschrieben. Da aber auch der Arduino kaputt gehen kann und die interne Uhr nach einiger Zeit sehr ungenau wird und nicht mehr mit der tatsächlichen Uhrzeit übereinstimmt, werden auch Wartungen zu diesen Komponenten beschrieben.

### 3.1 Durchführung Messung

Um eine Messung durchzuführen, sollte die Waage zuallererst auf eine ebene Oberfläche gestellt werden. Anschließend muss auf der Rückseite, der Schwarze An/Aus-Schalter eingeschaltet werden. Nun sollte das Ringlicht des Druckschalters auf der Vorderseite grün leuchten.

Leuchtet dieser Ring nicht siehe [Problem 1](#) und falls dieser rot leuchtet, siehe [Problem 2](#).

Leuchtet der Ring grün, ist die Waage bereit, eine Messung durchzuführen. Hierzu den Druckschalter mit dem Ringlicht drücken. Die Farbe des Ringlichts sollte nun auf Blau übergehen und die Aufnahme läuft. zum Beenden der Aufnahme erneut auf den Druckschalter drücken, und die Farbe sollte dann wieder auf Grün wechseln. Es kann nun mit einer weiteren Messung fortgefahren werden oder das System kann mit dem Schalter, der sich auf der Rückseite befindet, wieder ausgeschaltet werden.

Nach dem Durchführen einer Messung kann die SD-Karte entnommen werden und mittels eines Adapters an einem Computer angeschlossen. Die Dateien sind nach der Uhrzeit des Starts der Aufnahme benannt und können so etwas differenziert werden. Anzumerken ist, dass die Uhrzeit mit dem Laufe der Zeit weiter abweicht, oder bei einer leeren Knopfzelle des RTC-Moduls neu gesetzt werden muss. Hierzu siehe Kapitel [Uhrzeit einstellen](#).

Wurde die gewünschte Datei gefunden, kann mit der Auswertung begonnen werden. Beispielhaft wird dies im nachfolgenden Kapitel erklärt.

#### 3.1.1 Probleme

##### 3.1.1.1 Ringlicht leuchtet trotz eingeschaltetem Schalter auf Rückseite nicht:

Vermutlich ist die 9V-Batterie leer und sollte getauscht werden. Hierzu den Sperrhebel des Batteriefachs öffnen und durch leichtes Schütteln der Waage sollte sich die Batterie lösen. Nun kann diese vom Stecker getrennt werden und durch eine volle Batterie getauscht werden.

### 3.1.1.2 Ringlicht leuchtet rot:

Dies bedeutet, dass ein Fehler mit der SD-Karte vorliegt. Zunächst sollte geprüft werden, ob sich eine SD-Karte im vorgesehenen Slot befindet. Eventuell hilft es, diese neu aus- und wieder einzuführen. Da die Karte über einen Latch-Mechanismus gehalten wird sollte nicht nur gezogen werden, sondern gedrückt, bis ein Klicken zu hören ist. Beim Einführen sollte ebenfalls ein Klicken hör- und spürbar sein. Hilft dies nicht, ist die SD-Karte eventuell falsch formatiert, oder voll. Hierzu die SD-Karte mittels Adapter an einen Computer anschließen und eine Formatierung auf das Dateisystem FAT, oder FAT32 durchführen. Wird die SD-Karte vom Computer nicht erkannt, ist diese vermutlich defekt und sollte ersetzt werden.

## 3.2 Umgang mit den Daten der Messung

Die Daten werden als CSV-Datei auf der SD-Karte gespeichert. Eine CSV-Datei ist wie folgt aufgebaut, es gibt einzelne Zellwerte innerhalb einer Zeile, die durch ein Trennzeichen separiert werden. Dies ist zumeist ein Semikolon oder ein Komma. Die Zeilen werden mittels eines Zeilenumbruchs getrennt.

In diesem Projekt wird folgende Struktur verwendet:

Zeit seit Start der Messung [ms]	Absoluter Messwert	Kalibrierter Messwert [g]
----------------------------------	--------------------	---------------------------

Wird die SD-Karte (mittels Adapter) an einen Computer angeschlossen kann auf die Datei mit den Messwerten zugegriffen werden. Programme wie Microsoft Excel oder LibreOffice Calc können die CSV-Datei öffnen und die erfassten Werte in verschiedenen Formen z.B. rein tabellarisch oder auch in einem Diagramm darstellen

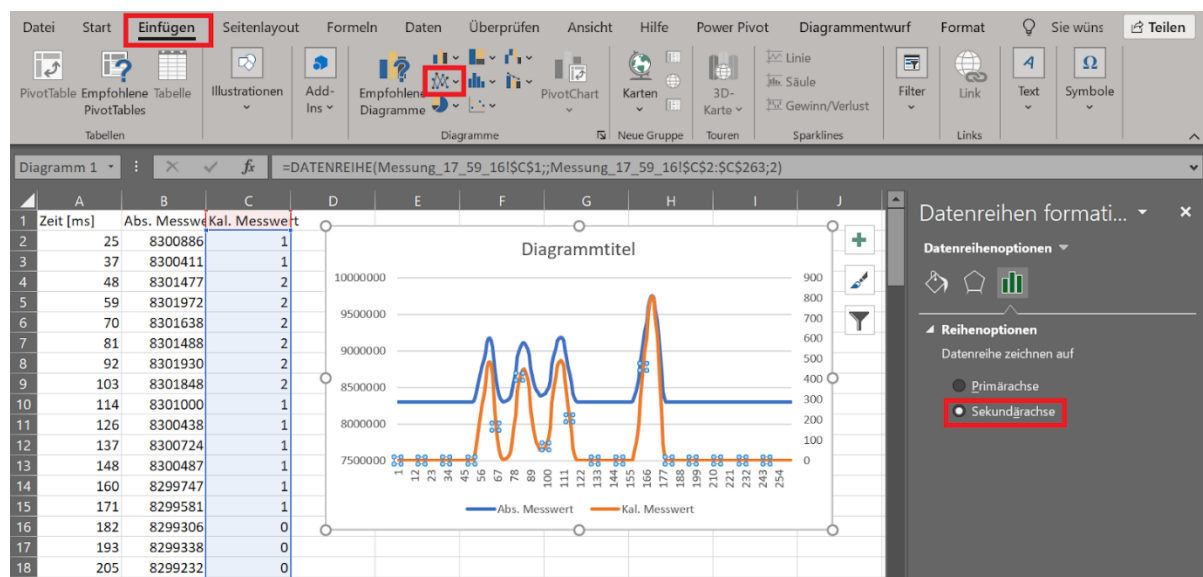


Abbildung 14 Microsoft Excel mit geladenem Datensatz zur Erklärung der Erstellung eines Diagramms

Um sich ein Diagramm generieren zu lassen wählt man nach Bedarf die Spalten Abs. Messwert und Kal. Messwert aus, klickt in Excel oben auf den Reiter **Einfügen** und dort auf **2D Linie**. Da die Wertebereiche der Spalten weit auseinander liegen bietet es sich an, eine zweite Y-Achse zu nutzen. Hierzu wählt man eine Werte-Linie aus, Rechtsklick -> **Datenreihen formatieren** und wählt dort **Sekundärachse** aus.

Möchte man die Abstände zwischen den Messwerten erhalten, schreibt man in die Zelle D3 die Formel **=A3-A2** und führt diese über Doppelklick in der rechten unteren Ecke zum Ende der Datensätze Fort. Um den Mittelwert darüber zu erhalten, gibt man in eine Leere Zelle (hier E3) **=MITTELWERT(D:D)** ein. Für die Frequenz der Messwerte in Hz gibt man in eine weitere Zelle **=1000/E3** ein. Dies ist beispielhaft in der Datei **MUSTER\_Messung.xlsx** im [Github repository](#) vorbereitet.

### 3.3 Uhrzeit einstellen

Das RTC-Modul verfügt als Spannungsquelle über eine Knopfzelle des Typ CR 2032. Sollte diese leer sein, stimmen die Dateinamen der Messergebnisse nicht mehr, und eventuell werden künftige Messungen durch Nachfolgende überschrieben. Es sollte also die Batterie gewechselt werden. Hierzu mit einem kleinen Schraubendreher vorsichtig die Knopfzelle aus der Halterung hebeln und eine neue einsetzen. Nun muss dieses Modul noch mit der aktuellen Uhrzeit beschrieben werden. Hierzu muss der mitgelieferte Code "[SetTime.ino](#)" in der Arduino-Entwicklungsumgebung geöffnet und geflashed werden. Anschließend muss erneut das Wäge-Programm "[Waegeprogramm.ino](#)" geflashed werden.

Um ein Programm auf den Arduino zu flashen wird die [Arduino IDE](#) benötigt. Der Arduino muss mittels USB-C Kabel an den Computer angeschlossen werden und in der IDE unter Werkzeuge->Port der entsprechende COM-Port des Arduino ausgewählt werden. Anschließend ist als Board "Arduino Nano", als Prozessor "ATmega328P" und als Programmer "AVRISP mkII" auszuwählen dies kann unter Werkzeuge->Board gesetzt werden. Nun kann das zu flashende Programm geladen werden bzw. eingefügt werden und über den Pfeil (→) -Button auf den Arduino geflashed werden. Am unteren Bildschirmrand befindet sich eine konsolenartige Ausgabe, welche über den Fortschritt und Erfolg Auskunft gibt.

### 3.4 Austausch Arduino

Sollte der Arduino kaputt gehen muss ein neuer Arduino eingebaut werden und der Quellcode initialisiert werden. Die Verkabelung des Arduinos ist unter dem Unterkapitel [Bauanleitung](#) beschrieben. Danach muss nur noch der Quellcode des Programms "[WaegeProgramm.ino](#)" auf den Arduino geflashed werden. Wie geflashed wird, ist im letzten Kapitel beschrieben.

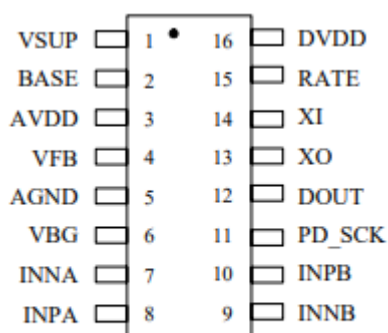
## 4 Probleme

Während der Entwicklung und dem Testen des Aufbaus sind mehrere Fehler aufgetreten, diese werden innerhalb dieses Kapitels beschrieben, und auch die Lösungen insofern diese vorhanden sind.

### 4.1 HX711-01

Bei den ersten Tests mit dem Proof of Concept Aufbau, wurden Werte und die aktuelle Zeit gespeichert, diese Daten wurden dann in Excel übertragen. Hier wurde die Zeit zwischen jeder Messung berechnet und daraus der Mindestwert, Maximalwert und der Durchschnittswert berechnet. Der Durchschnittswert zwischen den Messungen, war zwischen 80 ms und 90 ms. Dies entspricht umgerechnet aber nur eine Messfrequenz von ungefähr 11 Hz bis 12 Hz. Gefordert sind jedoch 80 Hz, deshalb wurde der Delay zwischen den Messungen angepasst und auch das Programm optimiert, jedoch blieb diese Rate ungefähr gleich.

Bei genauerer Betrachtung des Datenblatts des ursprünglichen HX711-01 fiel auf, das der integrierte SOP-16L Chip einen Eingang hat, um die Herzrate zwischen 10 Hz und 80 Hz zu wechseln. Dieser RATE Pin wird direkt über GND (Ground) angeschlossen. Wird nun jedoch statt 0V eine 5V Spannung angegeben wird die gewünschte Rate von 80 Hz erreicht. Wenn gewünscht ist, das die Rate geringer sein soll, kann dies Mittels einer Verzögerung im Programmcode theoretisch erreicht werden.



SOP-16L Package

Abbildung 15 SOL 16 Package (11)

Nachdem die beschriebenen Änderungen vorgenommen wurden, war die Messrate zwar zwischen ungefähr 80 Hz und 90 Hz, jedoch ging die Rate nach einiger Zeit zurück auf die vorige Rate. Die Suche nach dem Problem war erfolglos, weshalb ein neues Bauteil benötigt wurde.



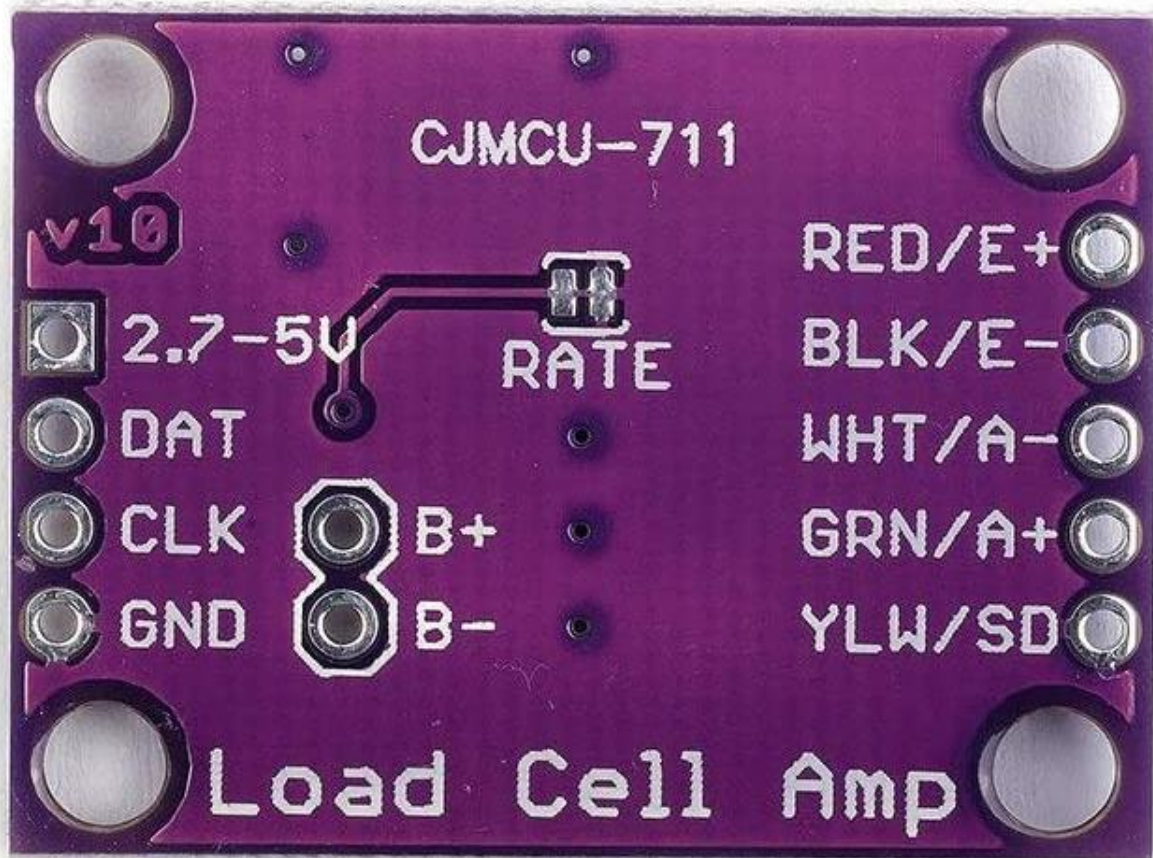


Abbildung 16 Verwendetes HX711-01 Modul (12)

Beim neuen HX711-01 Bauteil muss nur auf der Rückseite mittig, um die Rate anzupassen ein kleiner Kupferdraht durchgeschnitten werden. Sollte eine andere Frequenz gewünscht sein, können entweder die beiden Kontakte verlötet werden um 10 Hz zu erreichen, oder es kann auch die Rate auf der Softwareebene angepasst und variabel eingestellt werden. Mit dem neuen Bauteil sind die 80 Hz einfach zu erreichen, jedoch gab es Schwankungen, welche im Kapitel Schwankungen in der Messfrequenz beschrieben ist.

## 4.2 Kalibrierung

Um die Messwerte intuitiv verständlich zu machen und ihnen Bedeutung zu geben, werden die Messwerte kalibriert, in Gramm angegeben. Nachdem zunächst Objekte verschiedener Gewichtsklassen (60g, 360g, 700g) mit einer Küchenwaage als Basis gewogen werden. Dieselben Objekte werden daraufhin über 2-3 Sekunden mit dem Wägesystem gewogen. Durch die Messdauer von mehreren Sekunden kann der Mittelwert errechnet werden, da der HX711 leicht schwankende Ergebnisse liefert. Zusätzlich wird der Messwert ohne ein Gewicht erfasst, welcher als Nullpunkt dienen soll. Aus den gemessenen Werten der drei Objekten und dem Nullpunkt kann der Messwert des Objekts bestimmt werden. Als Quotient dessen und des tatsächlichen Gewichts errechnet sich ein Faktor je Objekt, wobei der Mittelwert dieser drei Faktoren als Berechnungsfaktor, für die Umrechnung des Messwerts in eine Gramm-Angabe dient. Mit der großen Messplatte beträgt der Nullpunkt 8381379 und der Berechnungsfaktor 1767.

Um die Kalibrierung zu überprüfen, werden nach Änderung der Faktoren im Sourcecode dieselben Objekte, sowie etliche weitere auf das Messsystem und die Küchenwaage gestellt. Die Differenz der Messwerte zwischen den beiden Messungen bewegt sich im Bereich von  $\pm 3g$ . Durch etliche Kalibrierungs-Anläufe über die Entwicklung hinweg, lässt sich jedoch sagen, dass die Messwerte von äußeren Einflüssen wie z.B. Temperatur oder Luftfeuchtigkeit, beeinflusst wurden. Diese Einflüsse wurden im Rahmen dieses Projekts jedoch nicht weiter untersucht. Zusätzlich anzumerken ist, dass für die kleine Messplatte ein anderer Nullpunkt gesetzt werden muss, da ansonsten die gemessenen Gewichte um die Gewichtsdivergenz der Platten abweicht.

Ein weiterer Ansatz zur Kalibrierung wäre eine Kalibrierungs-Prozedur beim Einschalten der Waage. Hier würde nach ein paar Sekunden Pause (zum Vermeiden von Messfehlern durch Berühren der Waage vom Einschalten) der Wert der leeren Waage ermittelt. Somit könnte man auch Versuchsaufbauten "nullen", jedoch wird dieses Feature bewusst nicht implementiert, da bei kleineren Tests mit Objekten auf der Plattform Probleme mit der Genauigkeit aufgetreten sind. Da die Umwandlung der Messwerte in eine konkrete Einheit schon ein Zusatz-Feature ist, wird mit dieser kleineren Weiterentwicklung kein wirklicher Mehrwert erbracht, weshalb der notwendige Aufwand nicht gerechtfertigt wäre.



## **4.3 Lose Kabelverbindungen**

Mehrfach über die Entwicklung des Messsystems ist das Problem aufgetreten, dass sich Kabel oder Stecker gelöst haben oder eine instabile Verbindung hatten. Haupt-Schwachpunkt hier waren die anfänglich verwendeten Wago-Klemmen, die für die Verbindung der Masse und 5V-Kabeln verwendet wurde. In der neuesten Iteration wurde deshalb auf Dupont-Steckleisten umgestiegen, welche zum einen weniger Platz einnehmen. Zusätzlich bieten diese auch eine bessere Verbindung für die dünnen Kabel, da hier nicht die Adern selbst Kontakt haben, sondern die aufgecrimpten Stecker. Diese Steckverbindung wird im Weiteren mit Heißkleber und Isolierband gegen versehentliches Lösen gesichert, sodass das Problem der losen Kabelverbindungen möglichst eliminiert wurde.

## 4.4 SD-Karten-Modul

Bei der Umstellung vom Micro-SD-Modul auf das reguläre SD-Modul fiel auf, dass das Ende der zu schreibenden Datei gefehlt hat. Die Vermutung ist, dass durch Buffering, der Arduino nicht sofort, sondern erst verzögert auf die Karte schreibt. Selbst ein “Flush” des Schreibmechanismus führte nicht zur Lösung des Problems. Als funktionierende Methode wird nun am Ende der Messung eine Datei “.trash” erstellt, was den Schreibmechanismus dazu zwingt, die gepufferten Werte in die CSV-Datei zu schreiben.

## 4.5 Spannungsversorgung

Der Arduino Nano verfügt über einen internen Spannungswandler. Dieser kann Spannungen im Bereich von 5-12V annehmen und versorgt somit OnBoard-Komponenten, wie auch kleinere externe Teile mit Energie. Als maximale Stromstärke wird im Datasheet 800mA genannt. (13)

Bei dem ersten Prototyp wird der Arduino Nano mit der Batteriespannung von~ 9V versorgt. Nach einiger Zeit funktionierte dieser jedoch nicht mehr. Der Arduino reagiert selbst im ausgebauten Zustand nicht über USB und außer, dass die Power-LED bei Anschluss an die Batterie oder USB leuchtet, ist keine Funktion erkennbar. Um dieses Problem zu beheben, wird in der nächsten Version der LM2596 Spannungswandler eingesetzt. Dieser kann Spannungen von bis zu 40V auf ein beliebiges Potential regeln und unterstützt Ströme bis 3A. (14) Seit Einsatz dieser Komponente und der Konfiguration auf 5V, ist das Problem nicht erneut aufgetreten.

## 4.6 Ungenauigkeit der RTC

Leider wird die Uhrzeit der RTC mit der Zeit immer ungenauer (im Bereich von wenigen Minuten pro Woche). Dies erschwert über längere Zeiträume hin die Identifizierung der Dateien mit Messdaten, was betreffend Usability tragisch ist. Zur Lösung könnte eventuell ein anderes RTC-Modul eingebaut werden, da das Kriterium der korrekten Uhrzeit, hier jedoch nur eine untergeordnete Rolle spielt, wird das Problem bestehen. Eine temporäre Lösung, ist das manuelle Setzen der Uhrzeit, sodass die Uhr für eine gewisse Zeit wieder die aktuelle Zeit hat. Hierzu siehe Kapitel [Uhrzeit einstellen](#)

## **4.7 Schwankungen in der Messfrequenz**

Beim Analysieren der Messwerte fiel auf, dass es in seltenen Fällen zu einer Schwankung der Frequenz kommt, jedoch liegt der Mittelwert der Frequenz zum Glück immer noch über 80 Hz und somit relativiert sich dieser Fehler. Die genaue Ursache für diese Schwankung konnte nicht ausfindig gemacht werden. Eine Vermutung ist das die Schwankung durch das Schreiben auf die SD-Karte entsteht, oder es könnte auch durch die höhere Frequenz des HX711-01 entstehen. Da hier die Frequenz deutlich über 80 Hz ist und im Programm nicht auf diesen Wert begrenzt wird. Eventuell kommt das Modul einen komischen Zustand, da nur für 80Hz vorgesehen ist, aber mehr erreicht wird.

## 5 Fazit

Während der Entwicklung der Waage entstanden viele unterschiedliche Gehäuseprototypen aus denen, nach einigen Iteration, das finale Produkt entstand. Während dieser Entwicklung sind einige Fehler bzw. Probleme aufgetreten, die teilweise gelöst worden sind und teilweise nicht lösbar waren. All diese Probleme sind im letzten Kapitel mit dem Titel Problem beschrieben.

Was die Messdauer angeht ist diese abhängig von mehreren Faktoren. Zum einen die Speichergröße der SD-Karte. Eine Messung mit einer Dauer von 3 Minuten entspricht ungefähr 300 KB, die Speicherkarte hat eine Größe von 4 GB und somit kann diese eine maximale Messdauer von ungefähr 666 Stunden speichern. Ein anderer limitierender Faktor ist die Batterie, diese wird voraussichtlich deutlich vor den 666 Stunden leer gehen. Eine genaue Dauer wurde hier aber nicht bestimmt. Des Weiteren kann es passieren das Komponenten, während der Messung, kaputt gehen und somit die Messdauer auch reduziert wird, und im schlimmsten Fall die Messung sogar falsch oder leer sein kann. Die Messfrequenz entspricht trotz den im letzten Kapitel beschriebenen Problem, im Mittel mehr wie die geforderten 80 Hz. Durch die hohe Messfrequenz werden die Werte zwar ungenauer, aber Genauigkeit war ja kein gefordertes Maß.

Es besteht zwar die Möglichkeit die Plattform zum Wiegen zu wechseln, allerdings muss dann auch der Kalibrierungswert im Programm geändert werden, um einen akkuraten Wert für den Kalibrierten Wert zu erzeugen. Da allerdings das genaue Gewicht weniger relevant ist wie Veränderung des Gewichts, sollte dies nicht allzu relevant sein. Der Austausch ist sehr einfach umzusetzen, da die Plattform mittels zwei M4-Schrauben befestigt wurde, und so schnell durchgewechselt werden kann. Eine Kalibrierung wurde für die kleinere Plattform nicht durchgeführt. Dementsprechend müsste erst der Wert über mehrere Tests bestimmt werden und dann im Programm geändert werden.

Alles in allem entspricht die Waage den gegebenen Anforderungen vollständig, jedoch gibt es einige Änderungen bzw. Verbesserungen, die in der Zukunft denkbar wären.

- Direkte Wiedergabe des aktuellen Messwerts und des Diagramms über ein Display direkt an der Waage oder drahtlose Übertragung auf Handy mittels Bluetooth.
- Austausch der Wägezelle um auch Gewichte von mehr wie 1 KG zu ermöglichen.
- Beim Starten der Messung Waage kalibrieren, um passend zur Plattform und anderen Gegebenheiten einen passend kalibrierten Wert zu haben. Dies ist allerdings nur bedingt relevant da nur die relative Änderung zwischen den Werten wichtig ist und nicht der genaue Wert.

- Auch denkbar wäre der Wechsel auf Akkus, um einen genauen ladezustand über bereits genanntes Display wiederzugeben.

## Quellcode

Anbei das Hauptprogramm und das Programm zum Setzen der Zeit.  
Der vollständige Quellcode und auch ein paar andere Dateien können im dazugehörigen [Github repository](#) gefunden werden.

### WaegeProgramm.ino

```
#include <Q2HX711.h>

const byte hx711_data_pin = 3;
const byte hx711_clock_pin = 2;

Q2HX711 hx711(hx711_data_pin, hx711_clock_pin);

int switchPin=8;

//LED
int ledRed=6;
int ledGreen=5;
int ledBlue=7;

//Status-Beschreibungen:
//Grün: Bereit zum Aufnehmen
//Blau: Aufnahme läuft
//Rot: Fehler

// SD-Card
#include <SPI.h>
#include <SdFat.h>
SdFat SD;
File myFile;
int millisecs=0;

//Time
#include <Time.h>
#include <DS1307RTC.h>
```

```
boolean stateRecording=false;

void setLED(int red, int green, int blue){
  analogWrite(ledRed, red);
  analogWrite(ledGreen, green);
  analogWrite(ledBlue, blue);
}

void error(){
  setLED(255,0,0);
  Serial.println("ERROR!");
  while(1);
}

void setup() {
  Serial.begin(9600);
  pinMode(switchPin,INPUT_PULLUP);
  pinMode(ledRed, OUTPUT);
  pinMode(ledGreen, OUTPUT);
  pinMode(ledBlue, OUTPUT);

  setSyncProvider(RTC.get);

  //SD-Karte prüfen
  if (!SD.begin(10)){
    Serial.println("SD initialization failed!");
    error();
  }

  Serial.println("SD initialization done.");
  //Bereit für Aufnahme -> LED grün
  setLED(0,255,0);
}

void loop() {
  if(digitalRead(switchPin)){
    if(!stateRecording)
    {
      stateRecording=true;
      //led auf blau
      setLED(0,0,255);

      millisecs=millis();

      String str="Messung_";
      str+=hour();
      str+="_";
    }
  }
}
```

```
    str+=minute();
    str+=" ";
    str+=second();
    str+=" .csv";

    myFile = SD.open(str, FILE_WRITE);
    Serial.print("Recording starting, filename: ");
    Serial.println(str);
    myFile.println("Zeit [ms];Abs. Messwert;Kal. Messwert");
}

//Datei beschreiben
if(myFile){
    unsigned long wert=hx711.read();
    //Kalibrierten Wert ausrechnen (nicht exakt!!)
    unsigned long wertCal=(wert-8381379)/1767;
    if(wertCal>2000000) wertCal=0;
    myFile.print(millis()-millisecs);
    myFile.print(";");
    myFile.print(wert);
    myFile.print(";");
    myFile.println(wertCal);
    Serial.print(wert);
    Serial.print(":");
    Serial.println(wertCal);

}

}else{
    error();
}

}else
{
    if(stateRecording)
    {
        stateRecording=false;
        setLED(0,255,0);
        myFile.flush();
        myFile.close();

        myFile = SD.open(".trash", FILE_WRITE);
        if(myFile)
            myFile.println(".");
        myFile.close();
        Serial.println("Recording stopping");
    }
}
}
```

## SetTime.ino

```
#include <Wire.h>
#include <TimeLib.h>
#include <DS1307RTC.h>

const char *monthName[12] = {
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
};

tmElements_t tm;

void setup() {
  bool parse=false;
  bool config=false;

  // get the date and time the compiler was run
  if (getDate(__DATE__) && getTime(__TIME__)) {
    parse = true;
    // and configure the RTC with this info
    if (RTC.write(tm)) {
      config = true;
    }
  }

  Serial.begin(9600);
  while (!Serial) ; // wait for Arduino Serial Monitor
  delay(200);
  if (parse && config) {
    Serial.print("DS1307 configured Time=");
    Serial.print(__TIME__);
    Serial.print(", Date=");
    Serial.println(__DATE__);
  } else if (parse) {
    Serial.println("DS1307 Communication Error :-{");
    Serial.println("Please check your circuitry");
  } else {
    Serial.print("Could not parse info from the compiler, Time=\");
    Serial.print(__TIME__);
    Serial.print("\", Date=\");
    Serial.print(__DATE__);
    Serial.println("\");
  }
}

void loop() {
}
```



```
bool getTime(const char *str)
{
    int Hour, Min, Sec;

    if (sscanf(str, "%d:%d:%d", &Hour, &Min, &Sec) != 3) return false;
    tm.Hour = Hour;
    tm.Minute = Min;
    tm.Second = Sec;
    return true;
}

bool getDate(const char *str)
{
    char Month[12];
    int Day, Year;
    uint8_t monthIndex;

    if (sscanf(str, "%s %d %d", Month, &Day, &Year) != 3) return false;
    for (monthIndex = 0; monthIndex < 12; monthIndex++) {
        if (strcmp(Month, monthName[monthIndex]) == 0) break;
    }
    if (monthIndex >= 12) return false;
    tm.Day = Day;
    tm.Month = monthIndex + 1;
    tm.Year = CalendarYrToTm(Year);
    return true;
}
```

## Literaturverzeichnis

1. **LEIFlphysik.** leifiphysik.de. [Online] [Zitat vom: 7. Juli 2022.] <https://www.leifiphysik.de/mechanik/kraft-und-masse-ortsfaktor/geschichte/kurze-geschichte-der-waagen>.
2. **Amazon.** amazon.de. [Online] [Zitat vom: 1. Mai 2022.] [https://m.media-amazon.com/images/I/61pTP44mEHL.\\_AC\\_SL1001\\_.jpg](https://m.media-amazon.com/images/I/61pTP44mEHL._AC_SL1001_.jpg).
3. **reichelt.** reichelt.de. [Online] [Zitat vom: 1. Mai 2022.] [https://cdn-reichelt.de/bilder/web/artikel\\_ws/A300/SEN-HX711\\_1.jpg](https://cdn-reichelt.de/bilder/web/artikel_ws/A300/SEN-HX711_1.jpg).
4. **Schmidt, Stefan.** hbm.com. [Online] [Zitat vom: 2. Mai 2022.] <https://www.hbm.com/de/6768/wie-funktioniert-eigentlich-eine-waagezelle/>.

5. —. hbm.com. [Online] [Zitat vom: 2. Mai 2022.]  
[https://www.hbm.com/fileadmin/\\_processed\\_/6/0/csm\\_Plattformwaagezelle\\_16b4b859ff.jpg](https://www.hbm.com/fileadmin/_processed_/6/0/csm_Plattformwaagezelle_16b4b859ff.jpg).
6. **conrad**. conrad.com. [Online] [Zitat vom: 6. Mai 2022.]  
<https://asset.conrad.com/media10/is/160267/19d4690e133380f076daee5ebc9306f73/c3/-/a3c5c4817724545e78fbe81dcff7348e1/ds1307-rtc-echtzeituhr-modul-i2c-for-arduino-cr2032-batterie.jpg?x=480&y=480&format=jpg&ex=480&ey=480&align=center>.
7. **AliExpress**. de.Aliexpress.com. [Online] [Zitat vom: 16. Mai 2022.]  
<https://de.aliexpress.com/item/1005002530096777.html>.
8. **reichelt**. reichelt.de. [Online] [Zitat vom: 16. Mai 2022.] [https://cdn-reichelt.de/bilder/web/artikel\\_ws/B300/ARDUINO\\_NANO\\_01.jpg](https://cdn-reichelt.de/bilder/web/artikel_ws/B300/ARDUINO_NANO_01.jpg).
9. **AliExpress**. de.aliexpress.com. [Online] [Zitat vom: 16. Mai 2022.] [https://cdn-reichelt.de/bilder/web/artikel\\_ws/B300/ARDUINO\\_NANO\\_01.jpg](https://cdn-reichelt.de/bilder/web/artikel_ws/B300/ARDUINO_NANO_01.jpg).
10. —. de.aliexpress.com. [Online] [Zitat vom: 20. Mai 2022.]  
<https://de.aliexpress.com/item/33004374185.html>.
11. **sparkfun**. cdn.sparkfun.com. [Online] [Zitat vom: 3. Juni 2022.]  
[https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711\\_english.pdf](https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf).
12. **amazon**. amazon.de. [Online] [Zitat vom: 3. Juni 2022.]  
[https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711\\_english.pdf](https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf).
13. **arduino**. docs.arduino.cc. [Online] [Zitat vom: 31. Juni 2022.]  
<https://docs.arduino.cc/static/f0d6ee3b3443d6307d1d341f2fc8de2d/A000005-datasheet.pdf>.
14. **Texas Instruments**. ti.com. [Online] [Zitat vom: 3. Juli 2022.]  
<https://www.ti.com/lit/gpn/lm2596>.
15. **King, Chris**. bronkhorst.com. [Online] [Zitat vom: 3. Mai 2022.]  
<https://www.bronkhorst.com/de-de/blogbeitr%C3%A4ge/genauigkeit-und-wiederholgenauigkeit-in-der-durchflusstechnik-was-ist-das-eigentlich/>.