

# Farming Surveillance System

Andreas Auer

Abschlussarbeit „Maturaprojekt“

Technologische Fachoberschule „Oskar von Miller“ Meran

Tutoren: Martin De Tomaso & Ivan Huber

Schuljahr 2021 / 2022

5C EL

## Table of Contents

<b>1.</b>	<b>EINLEITUNG .....</b>	<b>4</b>
1.1.	VORWORT.....	4
1.2.	ABSTRAKT .....	5
1.3.	ABSTRACT .....	5
1.4.	ASTRATTA .....	5
1.5.	INSPIRATION.....	6
<b>2.</b>	<b>PROJEKTÜBERBLICK .....</b>	<b>7</b>
2.1.	PROJEKTANTRAG .....	7
2.2.	PFLICHTENHEFT.....	10
2.3.	ZEITPLAN .....	12
2.4.	WORKINGPACKAGES .....	12
2.5.	WÖCHENTLICHE PROJEKTBERICHTE .....	13
<b>3.</b>	<b>HARDWARE .....</b>	<b>14</b>
3.1.	ERSTE IDEEN.....	14
3.2.	SENSOREN.....	15
3.2.1.	Feuchtigkeitssensor BME680.....	15
3.2.2.	Stromsensor SEN0211.....	16
3.2.3.	Funkmodule .....	17
3.2.4.	Relais RT14730.....	19
3.2.5.	Temperatursensor AD7415.....	20
3.2.6.	Türsensor .....	21
3.2.7.	Wasserdrucksensor Sensata 116CP31-M10S3-50.....	22
3.3.	WICHTIGE BAUTEILE .....	23
3.3.1.	Microcontroller MC9S08LL16.....	23
3.3.2.	Raspberry-Pi4 mit Display.....	26
3.3.3.	LM324 .....	27
3.3.4.	Spannungsregler LM1117-3.3, LM1117-5.0, LM317 und Li-Ion zu 3V3 .....	28
3.3.5.	Solarpaneele .....	30
3.3.6.	Uhrenbaustein PCF8563.....	31
<b>4.</b>	<b>PLATINEN.....</b>	<b>32</b>
4.1.	ÜBERBLICK .....	32
4.2.	MOD-V .....	33
4.2.1.	MOD-V_01 .....	33
4.3.	MOD-W .....	34
4.3.1.	MOD-W_V01 .....	34
4.3.2.	MOD-W_V02 .....	35
4.3.3.	MOD-W_V03 .....	36
4.4.	MOD-X .....	38
4.4.1.	MOD-X_V01 .....	38
4.4.2.	MOD-X_V02 .....	39
4.5.	MOD-Y .....	40
4.5.1.	MOD-Y_V01 .....	40
4.5.2.	MOD-Y_V02 .....	41
4.5.3.	MOD-Y_V03 .....	42
4.6.	MOD-Z .....	43
4.6.1.	MOD-Z_V01 .....	43
4.6.2.	MOD-Z_V02 .....	44
<b>5.</b>	<b>SOFTWARE .....</b>	<b>45</b>
5.1.	ÜBERBLICK .....	45
5.2.	TESTPROGRAMME .....	45

5.3. SOFTWARE-SENSOREN.....	45
5.3.1. Feuchtigkeitssensor BME680.....	45
5.3.2. Stromsensor SEN0211.....	46
5.3.3. Funkmodule .....	47
5.3.4. Relais.....	47
5.3.5. AD7415 .....	47
5.3.6. Türsensor .....	48
5.3.7. Wasserdrucksensor Sensata 116CP31-M10S3-50.....	49
5.4. SOFTWARE-PLATINEN.....	49
5.5. BASISSOFTWARE MICROCONTROLLER.....	50
<b>6. GEHÄUSE .....</b>	<b>52</b>
<b>7. PROJEKTMANAGEMENT: .....</b>	<b>52</b>
<b>8. KOSTENRECHNUNG PROTOTYP .....</b>	<b>53</b>
<b>9. KOSTENRECHNUNG SERIENTYP .....</b>	<b>53</b>
<b>10. ANHANG .....</b>	<b>54</b>
10.1. FAZIT .....	54
10.2. ABBILDUNGSVERZEICHNIS.....	55
10.3. QUELLEN .....	57
10.4. BILDQUELLEN.....	57
10.5. COPYRIGHT .....	59

## 1. Einleitung

### 1.1. Vorwort

Sehr geehrter Leser,

Bei dieser Arbeit von mir, Auer Andreas, handelt es sich um eine Dokumentation meines Maturaprojektes „Farming Surveillance System“ - kurz „FSS“. Wie jedes Jahr erarbeitet jeder Schüler der fünften Klasse der Technologischen Fachoberschule „TFO – Meran“ „Oskar von Miller“ im Laufe des Jahres eigenständig ein Projekt seiner Wahl.

Am Anfang des Jahres musste sich jeder Schüler der fünften Klasse Elektronik ein Projekt aussuchen. Das gewählte Projekt wird zuerst mit den beiden Professoren Ivan Huber und Martin De Tomaso besprochen und von ihnen abgesegnet (Kapitel 2.1 Projektantrag). Sobald das Projekt akzeptiert ist, beginnt das Planen und Umsetzen des Projektes. Die Arbeit muss allein durchgeführt werden, jedoch stehen einem die beiden bereits genannten Professoren im Notfall zur Verfügung.

Das Ziel des Projektes ist das gezielte Anwenden des Wissens und der Erfahrung, die man sich in den letzten Schuljahren in den Fächern TPS (Technologie und Projektierung elektrischer und elektrotechnischer Systeme), Automation und Elektronik angeeignet hat. Auch sollen die Hindernisse und Probleme, die man über das Jahr verteilt meistert, für die künftige Arbeitswelt von Vorteil sein.

Die Auswahl des Projektes war weitgehend frei. Die wenigen Voraussetzungen waren das Verwenden eines Microcontrollers, das Erstellen einer Schaltung und natürlich das Interesse am Erschließen des Projektes. Eine weitere Vorgabe war der Abgabetermin, an dem das Projekt vollständig beendet und der Klasse inclusive den beiden Lehrpersonen vorgestellt werden muss.

Für das Erarbeiten des Projektes standen einem so gut wie alle TPS-Wochenstunden (fünf von sieben Stunden die Woche) zur Verfügung. Aufgrund der Corona-Pandemie waren an den Montag- und Mittwochnachmittagen nur selten Laborstunden zur Verfügung. Bei den Laborstunden ist nachmittags außerhalb der Schulstunden das Labor geöffnet und eine Lehrperson vor Ort, die mit Rat und Tat zur Verfügung steht. Die gesamte restliche Arbeit musste zu Hause ausgeführt werden. In meinem Falle habe ich die meiste Arbeit zu Hause verrichtet, habe die Laborstunden am Nachmittag nur sehr selten genutzt und mich in den Unterrichtsstunden darauf konzentriert, Arbeiten zu erledigen, welche ich zuhause nicht durchführen konnte.

## 1.2. Abstrakt

Mein Projekt beschäftigt sich mit einem erweiterten elektronischen Überwachungssystem des Hofes meiner Familie, welches variabel auf verschiedenste Objekte angewendet werden kann.

Das Ziel meines Projektes war es Folgendes zu überwachen: welche Türen offen oder geschlossen sind, verschiedene Temperaturen (Luft, Wasser, Heu), den Wasserdruck der Wasserzuleitung des Hofes, die Luftfeuchtigkeiten im Stadel und im Stall, den Stromverbrauch und die Stromversorgungssicherungen auf dem Hof. Die errechneten, gemessenen und ermittelten Daten sollten an einem zentralen Punkt gesammelt und per Funk an das 300 Meter entfernte Haus gesendet werden.

Mein Vorhaben stellte sich als eine große Herausforderung für mich heraus. Ich habe mir sehr viele wertvolle Errungenschaften angeeignet, sei es in der Planung, der Umsetzung oder auch der Geduld. Ich habe viele Fähigkeiten erlernt, welche mir in der Zukunft weiterhelfen werden.

## 1.3. Abstract

My project is about an extended electrical surveillance system for the farm of my family, though it can be used on several different objects.

The goal of my project was checking the following parameters: doors are open or closed, different temperatures (air, water, hay), the water pressure of the input water pipe of the farm, the air humidity in the barn and the stable, the current consumption and the incoming current fuses at the farm. The calculated, measured and collected data should be gathered at a central point and then send to the house via radio technology.

My project turned out to be a huge challenge for me. I gained a lot of valuable experience by completing this endeavour. I acquired a lot of capabilities which will surely help me in the future.

## 1.4. Abstract

Il mio progetto riguarda un sistema di monitoraggio elettronico esteso dell'azienda agricola della mia famiglia, che può essere applicato in modo variabile a diversi oggetti.

Lo scopo del mio progetto era quello di monitorare: le porte aperte o chiuse, le diverse temperature (aria, acqua, fieno), la pressione dell'acqua della rete idrica della fattoria, i livelli di umidità nel fienile e nella stalla, il consumo di energia e i fusibili dell'alimentazione della fattoria. I dati calcolati, misurati e determinati dovevano essere raccolti in un punto centrale e inviati via radio a una casa a 300 metri di distanza.

Il mio progetto si è rivelato una grande sfida per me. Ho acquisito molte competenze preziose, che si tratti di pianificazione, attuazione o pazienza. Ho imparato molte abilità che mi saranno utili in futuro.

### 1.5. Inspiration

Bereits im Schuljahr 2020/2021 wurde uns mitgeteilt, dass im darauffolgenden Schuljahr eine Abschlussarbeit im Fach TPS (Technologie und Projektierung elektronischer und elektrotechnischer Systeme) auszuarbeiten ist. Daraufhin habe ich mir im Sommer zwischen den beiden Schuljahren bereits einige Gedanken gemacht und mich dazu entschlossen, ein Videospiel umzuprogrammieren. Diese Idee stellte sich aber sehr schnell als viel zu einfach umsetzbar dar.

Im Herbst 2021 hatten wir dann ungefähr zwei Wochen Zeit, uns ein Projekt zu überlegen. Anfangs hatte ich keine Ahnung welches Projekt ich umsetzen sollte. Dank den Ideen meiner Mitschüler und der Unterstützung meiner Familie habe ich mich dann nach langem Überlegen für ein Projekt entschieden. Zuerst wollte ich die Türen auf dem Hof meiner Familie überwachen, da dieser circa einen Kilometer entfernt von unserem Wohnhaus steht und somit nicht unbedingt einbruchsicher ist.

Aus dieser kleinen Ursprungsidee ist schnell ein großes Projekt entstanden. Zusätzlich zur Überwachung der Türen wollte ich auch die Temperatur im Stall sowie die Außentemperatur wissen. Wichtig jedoch erschien es mir, die Temperatur des Wassers zu kontrollieren, da im Winter ständig die Gefahr droht, dass es aufgrund der niederen Temperaturen abfriert. Im Sommer wiederum wäre es wichtig, die Temperaturen des Heus im Stadel zu wissen, um einen möglichen Brand auszuschließen, weshalb ich mich dazu entschlossen habe, Temperatursensoren für all das vorzusehen. Zusätzlich zur Wassertemperatur wollte ich auch den Wasserdruck überwachen, um einen Wasserausfall zu vermeiden und den Tieren Wasser zum Trinken zu garantieren. Auch die Feuchtigkeit im Stadel und im Stall stellten für mich eine gewisse Priorität dar, da die richtige Luftfeuchtigkeit im Stall für die Gesundheit der Tiere ausschlaggebend ist und das Heu im Stadel nicht mit zu feuchter Luft belüftet werden darf. Am Ende entstand ein sehr großes Projekt auf dessen Abschluss ich sehr stolz bin. Immer wieder habe ich mich vom Enthusiasmus meiner Mitschüler und meiner Familie anstecken lassen und aktiv weitergearbeitet.

## 2. Projektüberblick

### 2.1. Projektantrag

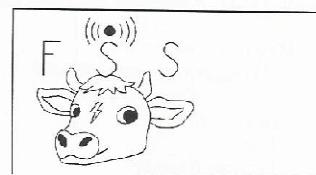
# Projektantrag

Ansuchen für ein Maturaprojekt, Fassung September 2021

Projekttitle:  
Projektleiter:  
Klasse und Schuljahr:

FSS (Farm Surveillance System)	
Andreas Auer	
5C EL	Schuljahr 2021 / 2022

Logo des Projektes:



Kurzbeschreibung des Projektes: Überwachungssystem des Hofes. Es soll verschiedene Parameter am Hof ermittelt und diese dann per Funk zum 300 Meter Luftlinie entfernten Haus gesendet werden. Siehe Anhang

Meilensteine des Projektes: Prototyp Platinen (Mod-X, Mod-Y); Ansteuerung Funkmodule; Kommunikation Funkmodule; Protokoll Module; Raspberry-Pi in Betrieb nehmen; erste Version des Raspberry-Pi Codes; Zusammenführung der Funkmodule mit dem Raspberry-Pi;

Anhänge:  JA (2 Seiten)  NEIN

Projekt genehmigt:  JA  NEIN

Maximal erreichbare Note bei 100% Funktionalität: 10

Martin De Tomaso  
Ivan Huber



Der Projektleiter

Andreas Auer

Abbildung 1: Projektantrag Seite1

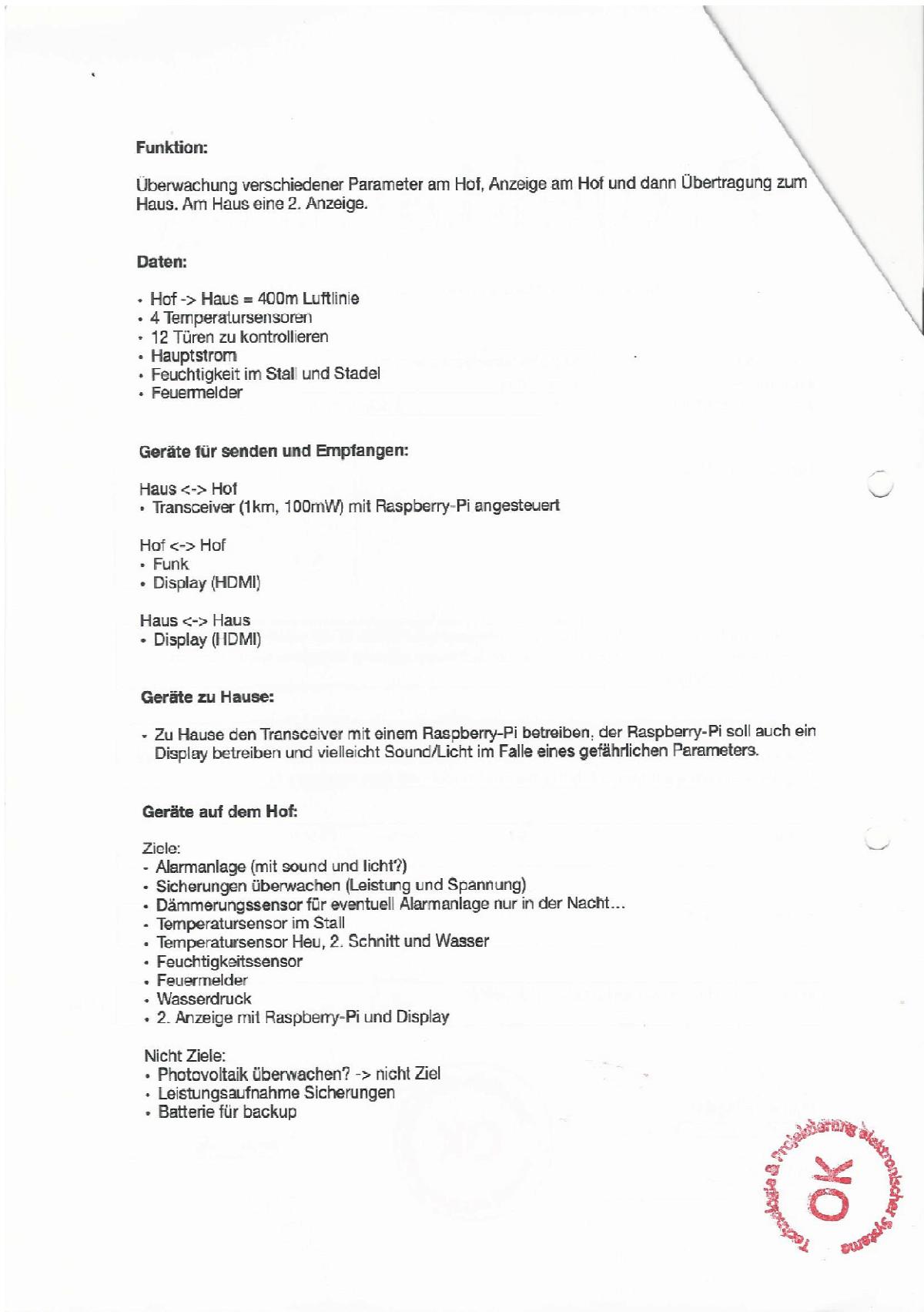


Abbildung 2: Projektantrag, Seite 2



FSS

23.09.2021

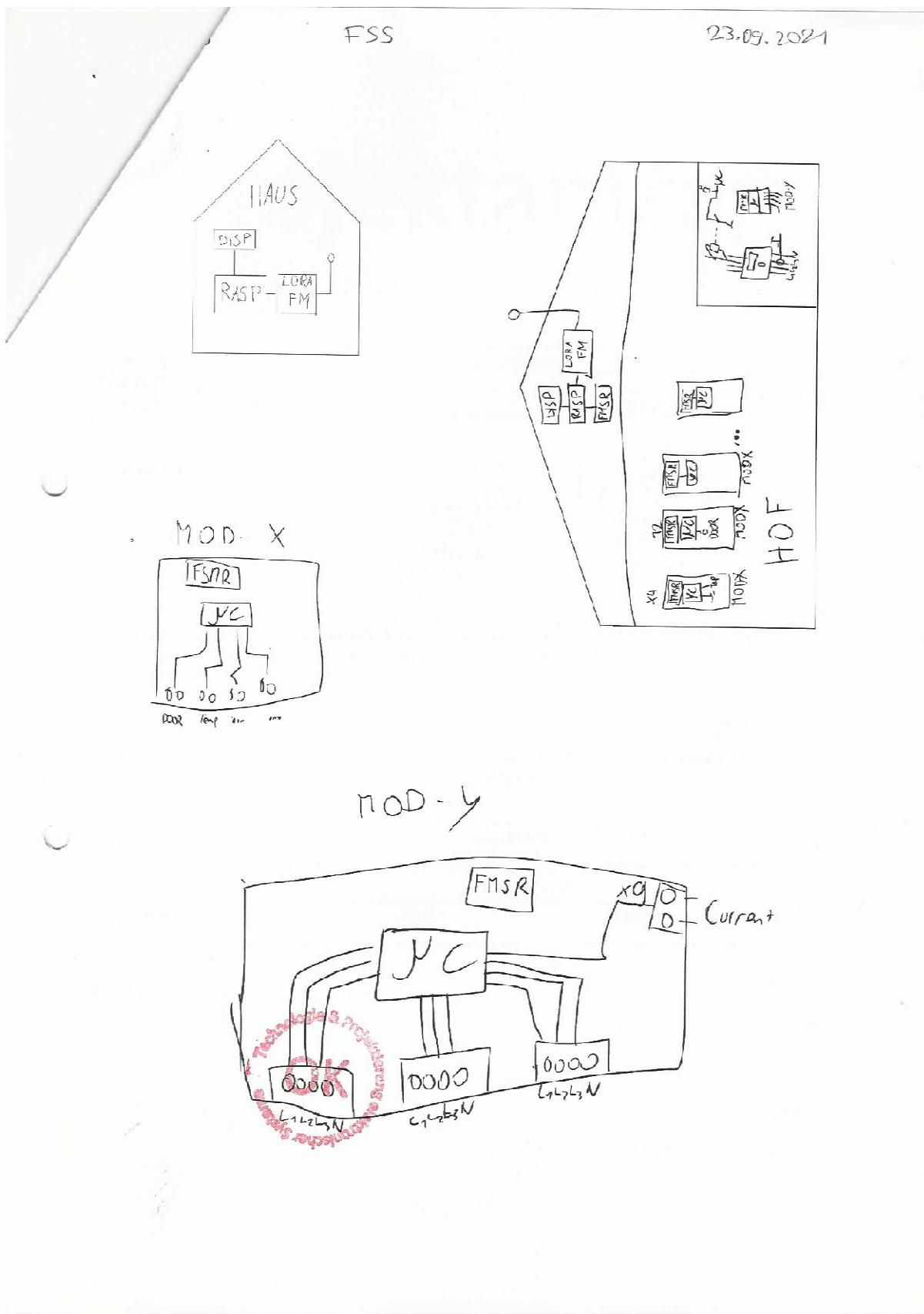


Abbildung 3: Projektantrag Seite 3

## 2.2. Pflichtenheft

Andreas Auer 28.09.2021

Pflichtenheft

28.09.2021

### Übersicht

Der Auftrag ist, mithilfe von Microcontrollern, Raspberry Pi's, Funkmodulen und Sensoren Verschiedene Parameter am Hof (Entfernung Haus <-> Hof = 400m) zu messen und dann zum Haus zu senden.

#### Platine Modul-X

Die Platine X ist für die Temperaturmessung, Tür auf/zu Messung, Feuchtigkeitsmessung, Wasserdruck-Messung und den Feuermelder. Jede Platine wird mit maximal 3 der genannten Sensoren bestückt. Es gibt 12 Türsensoren (Digitalsensor Alibaba), 4 Temperatursensoren (AD7415), 2 Feuchtigkeitssensoren (BME680), 1 Wasserdruck Sensor (sensata\_10292018\_116cp\_series\_pressure\_sensor), 2 Feuermelder (BML680) die folgend bestückt werden: es gibt 11 PCB's mit je einem Türsensor, 1 PCB mit einem Temperatursensor, 1 PCB mit einem Tür-, einem Temperatur- und einem Feuchtigkeitssensor, 1 PCB mit einem Temperatur und Feuchtigkeitssensor und 1 PCB mit einem Temperatur- und Wasserdrucksensor.  
Auch wenn jeweils nur bestimmte Sensoren pro Platine verwendet werden, werden immer alle Klemmen nach aussen geführt und Steckplätze für THT Bauteile bereitgestellt.

#### Platine Modul-Y

Die Platine Y ist für die Strommessung und für die Kontrolle der Sicherungen. Der Strom wird hier 3 mal gemessen. Es gibt drei drei-Phasen-Sicherungen bei denen jeweils eine Phase zur Leistungsberechnung gemessen wird. Es werden insgesamt 9 Phasen mit einem Relay überwacht. Dabei wird das Relay von der Phase durchgeschaltet um dann 3,3V durchzuschalten oder zu sperren. Diese 3,3V werden dann mithilfe des digitalen Eingangs des Controllers gemessen und dann kann man sagen, ob die Sicherung ein- oder ausgeschalten ist.

Der Strom wird mithilfe des Moduls SEN0211 gemessen. Der Stromsensor hat ein Zwischenbauteil das die Ausgangsspannung umwandelt und dann Analog mit dem Controller misst.

#### Funkmodul LoRa

Das LoRa Funkmodul dient dazu, um eine Funkverbindung zwischen Haus und Hof herzustellen. Es handelt sich dabei um ein ERIC9 Modul. Das Modul kann auf mehreren Frequenzen senden, ich werde aber 868MHz verwenden, da das Modul um die 400m Luftlinie überbrücken muss. Das Modul wird mit 2,4V bis 6V betrieben. Die Versorgungsspannung muss clean, noise and ripple free sein. Ich werde das Modul mit 3,3V betreiben. Das Modul braucht einen maximalen Strom von 33mA beim Senden von Daten.

#### Funkmodul SoRa

Das SoRa Funkmodul ist für die Datenübertragung am Hof zwischen dem Master (Raspberry Pi) und den Slaves (Modul-X und Modul-Y) herzustellen. Es handelt sich um ein CC2500 Funkmodul. Das Modul wird von 0V - 3,9V betrieben. Ich werde 3,3V verwenden. Der maximale Stromverbrauch des Moduls liegt bei 21mA. Das Modul sendet mit 2,4GHz

#### Microcontroller

Als Microcontroller wird ein MC9S08LL16CLH verwendet. Es handelt sich hier um ein 64 Pin Microcontroller. Der Microcontroller wird mit einer Spannung von 0-3,8V betrieben. Ich werde hierbei 3,3V verwenden. Der Strom der bei laufendem Controller verwendet wird ist 10mA.

#### Sensoren

- Türsensor: Der Türsensor wird mit 3,3 betrieben und schaltet durch, sobald die Kontakte unterbrochen werden. Man kann dann einfach mit dem Controller Digitaleingang messen, ob der Eingang auf LOW oder HIGH ist und dementsprechend handeln.
- Temperatursensor: Der Temperatursensor AD7415 wird mit IIC angesteuert und gibt einem die Temperatur aus. Arbeitsbereich: -40 bis +125 Grad Celsius. Betrieben mit 2,7 bis 5,5V. Ich werde 3,3V nehmen. Hat eine Genauigkeit von 0,25 Grad Celsius. Wird mit 1mA betrieben und hat einen maximalen Strom im Einschaltmoment von 1,2mA.
- Feuchtigkeitssensor: Der Feuchtigkeitssensor BME680 ist ein multifunktionaler Sensor der auch die Gasmessung (Feueralarm) für mich machen wird. Angesteuert wird er dabei über IIC. Man liest dann einfach die entsprechenden Register aus. Betrieben wird der Sensor mit 1,71 bis 3,6V. Ich werde ihn mit

Abbildung 4: Pflichtenheft Seite 1

Andreas Auer 28.09.2021

Pflichtenheft

28.09.2021

3,3V betreiben. Der Sensor braucht einen maximalen Strom von 18mA bei einer Gasmessung und 900 nA bei einer Druck oder Feuchtigkeitsmessung.

- Wasserdruck-Sensor: Der Wasserdruck-sensor sensata 116CP M10 arbeitet mit 5VDC. Der Sensor hat einen Betriebsstrom von 8mA. Das Ausgangssignal ist ein einfacher Analogwert zwischen 0,5 und 4,5V. Dieses Signal werde ich in ein 0 - 3,3V Signal umwandeln. Die Arbeitstemperatur des Moduls liegt bei 2 bis 90 Grad Celsius.
- Stromsensor: Der Stromsensor SEN0211 wird mit 3,3-5,5V betrieben. Das Ausgangssignal ist ein Analogsignal von 0,2 - 2,8V. Dieses werde ich mit einem OPV auf 0 - 3,3V bringen und dieses Signal wird dann mit einem ADC des Controllers verarbeitet. Der Sensor kann einen Strom von 0-20A AC messen. Die Stromaufnahme des Moduls ist unbekannt.
- Relay: Das Relay wird bei allen drei Phasen der drei Sicherungseinheiten zwischengeschalten. An der coil Seite des Relays werden 3,3V angelegt und diese dann vom Controller gemessen.

Ich werde alle Sensoren ausser dem Wasserdruck-Sensor mit 3,3V betreiben, der Wasserdruck Sensor wird mit 5V betrieben.



Abbildung 5: Pflichtenheft Seite 2

### 2.3. Zeitplan

Für das Projekt stand uns so gut wie das ganze Schuljahr zur Verfügung. Das beinhaltete fünf Stunden TPS die Woche und die Zeit, die wir zu Hause mit dem Projekt verbrachten. Insgesamt habe ich mich 33 Wochen lang mit dem Projekt auseinandergesetzt, wobei ich nur ungefähr eine Woche nicht daran gearbeitet habe und mich die restliche Zeit immer zwischen fünf und 25 Stunden in der Woche mit dem Projekt beschäftigt habe. Gegen Ende des Schuljahres hat sich mein Arbeitsaufwand fast verdoppelt, da die Fertigstellung des Projektes nochmal ein wenig mehr Arbeit als geplant in Anspruch nahm.

### 2.4. Workingpackages

Am Anfang des Jahres mussten wir unser Projekt in Workingpackages unterteilen. Ich unterteilte mein Projekt in 22 Workingpackages.

WP1: Projektdefinition. Bei der Projektdefinition (2.1) ging es darum, festzulegen was die Ziele des Projektes waren.

WP2: Pflichtenheft. Beim Erstellen des Pflichtenheftes (2.2) war es wichtig, niederzuschreiben welche Sensoren und welcher Microcontroller oder bei meinem Projekt auch welche Funkmodule verwendet werden, um die wichtigsten Daten der Bauteile zu erfassen.

WP3 + 4: Planung Schaltplan. Nachdem ich entschieden habe, welche Bauteile ich zu verwenden beabsichtigte, habe ich einen Schaltplan rund um die Sensoren entwickelt. Dabei haben sich dann alle sechs Platinen im Laufe der Zeit entwickelt.

WP5: Test der Hardwarekomponenten. Damit man mit den verschiedenen Sensoren arbeiten kann, ist es wichtig die Bauteile einer Schaltung bestens zu kennen und auch zu wissen, wie man sie ansteuert.

WP6: Zeichnen der Platinen und der Schematic. Ein Grundbaustein eines jeden Abschlussprojekts ist die Platine. Ohne eine Schematic kann man keine Platine entwickeln.

WP7: Ansteuern des Funkmoduls. Die Ansteuerung des Funkmoduls ist eine der wichtigsten Softwares, die in meinem Projekt präsent ist, da das Funkmodul alle Schaltungen miteinander verbindet.

WP8: Kommunikation zwischen zwei Funkmodulen. Hierbei ist es wichtig, dass ich eine Kommunikation mit dem Funkmodul aufbauen und Daten senden kann.

WP9 + 10: Bestücken der Platine. Nach dem Bestellen und dem Erhalt der Platine muss sie bestückt werden, sofern das nicht bei der Firma, die die Platinen erstellt hat, bereits erledigt wurde.

WP11: Inbetriebnahme der Platine. Nach dem Bestücken muss eine Platine getestet und eventuelle Fehler festgestellt werden.

WP12: Ansteuerung Temperatursensor. Nachdem die Platine betriebsbereit ist, wird mit dem Ansteuern der Sensoren begonnen.

WP13 + 14: Erstellen der Version zwei der Platinen. Falls gröbere oder nicht zu beseitigende Fehler auftreten, muss eine neue Version der Platine entworfen werden.

WP15 + 16 + 17 + 18 + 19: Ansteuerung der Sensoren. Nach dem Entwickeln einer zweiten Version beginnt die Ansteuerung der weiteren Sensoren.

WP20: Kommunikation Microcontroller mit Raspberry-Pi. Sobald alle Sensoren ansteuerbar sind, ist es wichtig eine Kommunikation zwischen allen Platinen und dem Raspberry-Pi aufzubauen.

WP21: Touchdisplay. Ansteuerung des Displays mit dem Raspberry-Pi.

WP22: Menu. Erstellen des Menus für den Raspberry-Pi. Darstellung aller Sensoren und Parameter.



## 2.5. Wöchentliche Projektberichte

Es war Pflicht, jede Woche einen sogenannten Wochenbericht auszufüllen und in die Projektmappe zu legen. Dabei handelt es sich um ein einfaches Dokument, in dem niedergeschrieben wird, was man in der entsprechenden Woche erarbeitet und wie viele Stunden man für das Projekt verwendet hat.

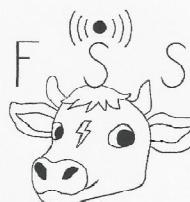
Wöchentlicher Projektbericht		
		
Projekttitle:	FSS (Farm Surveillance System)	
Projektleiter:	Andreas Auer	
Projektwoche / Datum:	PW: 32	25.04.2022 - 01.05.2022
<b>Kurzbericht der erledigten Arbeiten in der Projektwoche:</b> Fertigstellung und testen der Software zur Datenübertragung von Raspberry-Pi zu Raspberry-Pi Erste Versuche der Kommunikation von mehreren Platinen mit dem Raspberry-Pi und darstellung auf dem Display		
Im Zeitplan laut GANTT-Diagramm: <input checked="" type="checkbox"/> JA <input type="checkbox"/> NEIN		
Wenn NEIN, welche Working Packages (WP) liegen in Verzug? _____		
Vorgeschlagene Maßnahmen, um den Zeitplan wieder aufzuholen: _____		
Wurden die Maßnahmen aus letzter Woche planmäßig umgesetzt? <input checked="" type="checkbox"/> JA <input type="checkbox"/> NEIN		
Kurze Zusammenfassung des Zeitaufwandes der Projektwoche:		
Stunden in der Schule (Labor)	5	Stunden
Stunden im offenen Labor		Stunden
Stunden Privat / Zuhause	15	Stunden
Gesamtstundenzahl	20	Stunden
Datum: 18.10.2021	Unterschrift verantwortlicher Projektleiter: <i>Auer Andreas</i>	

Abbildung 6: Beispiel Wochenbericht

### 3. Hardware

#### 3.1. Erste Ideen

In den ersten ein, zwei Wochen begann die Planung meines Projektes. Erste Zeichnungen entstanden, mithilfe denen ich den Grundbaustein für das Projekt legen konnte. Diese Zeichnungen spornten meinen Willen an, dieses Projekt unbedingt zu schaffen.

Schnell wurde mir klar, dass die Zeit knapp ist und ich sofort mit dem Organisieren und Auswählen der Sensoren beginnen musste. Nachdem ich die Sensoren gefunden hatte, die ich verwenden wollte, begann ich, mich mit ihrer Funktionsweise und Ansteuerung vertraut zu machen. Auch musste ich mir Gedanken machen, wie ich die Daten der einzelnen Sensoren vereinen und visualisieren könnte. Deshalb habe ich mich dazu entschieden, die Daten mithilfe von Funkmodulen zu senden, mit einem Raspberry-Pi zu empfangen und dann zu verarbeiten. Aufgrund der Distanz von meinem Wohnhaus zum Hof, auf dem die Anlage installiert werden soll, habe ich mich für ein zusätzliches, stärkeres Funkmodul entschieden, mit welchem ich die Daten vom Hof zum Haus senden kann.

Als ich genug Informationen über die Sensoren gefunden hatte, begann ich damit, die Platinen zu planen. Dafür entwarf ich zuerst Schaltungsschemen auf Papier, mithilfe denen ich eine generelle Grundidee für die Platinen bekam und die Bauteile, die für die Platinen wichtig sind, aussuchen konnte.



### 3.2. Sensoren

#### 3.2.1. Feuchtigkeitssensor BME680

Beim Feuchtigkeitssensor BME680 handelt es sich um einen Sensor, der nicht nur die Feuchtigkeit, sondern auch die Temperatur, den Druck und die Gasqualität messen kann. Laut Datenblatt ist der Sensor unter anderem für die Messung der Luftqualität im Haus, die Wettervorhersage und auch die vertikale Geschwindigkeitserkennung geeignet. Ich verwende den Sensor, um zwei Temperaturen und Feuchtigkeiten zu messen. Einmal im Stadel, welcher aufgrund der guten Durchlüftung die Außentemperatur und die Feuchtigkeit der Umgebung hat und einmal im Stall, wo es wichtig ist, die Feuchtigkeit und Temperatur, sowie auch die Luftqualität im Auge zu behalten, um das Wohlbefinden der Tiere zu erhöhen.

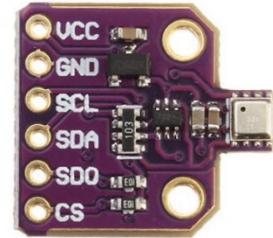


Abbildung 7: Ansicht BME680

Der Sensor ist mit 3,3 Volt Spannung versorgt und verfügt über ein I2C- sowie ein SPI-Interface. Beim BME680 handelt es sich um einen „low-current“ Sensor, das bedeutet, dass der Stromverbrauch auf ein Minimum gebracht wird. Die maximale Stromaufnahme laut Datenblatt ist circa 850 Microampere, was vernachlässigbar ist.

```
Floating point:
var1 = ((double)t_fine / 2.0) - 64000.0;
var2 = var1 * var1 * ((double)par_p6 / 131072.0);
var2 = var2 + (var1 * (double)par_p5 * 2.0);
var2 = (var2 / 4.0) + ((double)par_p4 * 65536.0);
var1 = (((double)par_p3 * var1 * var1) / 16384.0) +
    ((double)par_p2 * var1)) / 524288.0;
var1 = (1.0 + (var1 / 32768.0)) * (double)par_p1;
press_comp = 1048576.0 - (double)press_adc;
press_comp = ((press_comp - (var2 / 4096.0)) * 6250.0) / var1;
var1 = ((double)par_p9 * press_comp * press_comp) / 2147483648.0;
var2 = press_comp * ((double)par_p8 / 32768.0);
var3 = (press_comp / 256.0) * (press_comp / 256.0) *
    (press_comp / 256.0) * (par_p10 / 131072.0);
press_comp = press_comp + (var1 + var2 + var3 +
    ((double)par_p7 * 128.0)) / 16.0;
```

Abbildung 8: Berechnung des Drucks BME680

#### 1.1 General Electrical Specification

Table 1: Electrical parameter specification  
OPERATING CONDITIONS BME680

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Supply Voltage Internal Domains <sup>1</sup>	V <sub>DD</sub>	ripple max. 50 mVpp	1.71	1.8	3.6	V
Supply Voltage I/O Domain	V <sub>DIO</sub>		1.2	1.6	3.6	V
Sleep current	I <sub>DDSL</sub>			0.15	1	µA
Standby current (inactive period of normal mode)	I <sub>DDSB</sub>			0.29	0.8	µA
Current during humidity measurement	I <sub>DDH</sub>	Max value at 85 °C		340	450	µA
Current during pressure measurement	I <sub>DDP</sub>	Max value at -40 °C		714	849	µA
Current during temperature measurement	I <sub>DDT</sub>	Max value at 85 °C		350		µA

Abbildung 9: Elektrical characteristics BME680

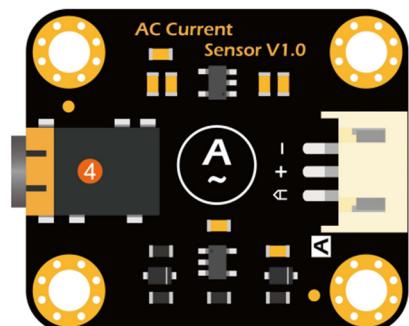


### 3.2.2. Stromsensor SEN0211

Der Stromsensor SEN0211 ist ein AC (50Hz, 230V) Stromsensor, welcher bis zu 20A Wechselstrom messen kann. Es gibt verschiedene Typen, welche sich jeweils in der maximalen Stromaufnahme unterscheiden. Da auf meinem Hof teilweise sehr große Ströme gebraucht werden, habe ich mich für den stärksten Sensor entschieden, welcher erwähnte 20A schafft.

Versorgt wird der Sensor mit +3,3 – 5V und Masse. Der Sensor gibt ein Analogsignal zurück, welches zwischen 0,2 und 2,8V je nach anliegendem Strom variiert.

#### Board Overview



LABEL	NAME	Function Description
1	-	GND
2	+	Power Input (3.3V-5.5V)
3	A	Signal Output (0.2-2.8VDC)
4	Φ3.5mm 3P plug	AC Transformer Input Signal

Abbildung 10: Board overview SEN0211

#### Connection Diagram

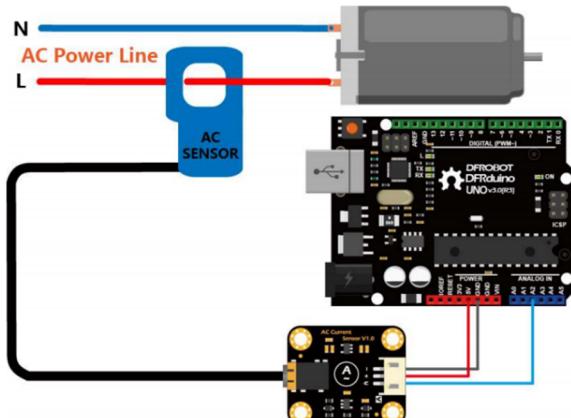


Abbildung 12: Benutzung Stromsensor



Abbildung 11: Ansicht Stromzange SEN0211

### 3.2.3. Funkmodule

Für mein Projekt benötigte ich zwei Funkmodule. Ein kostengünstiges, mit welchem ich die Daten auf dem Hof sammeln kann, und eines, das leistungsstark genug ist, um die gesammelten Daten vom Hof zum Haus zu senden.

#### 3.2.3.1. SoRa CC2500

Das SoRa (= short range) Funkmodul CC2500 ist ein sehr günstiges Funkmodul, das auf einer Frequenz von 2400MHz sendet, also WLAN Frequenz. Jede meiner Platinen ist mit einem eigenen CC2500 Funkmodul ausgestattet, welches dazu verwendet wird, die ausgelesenen Daten an den Raspberry-Pi, welcher auch mit einem CC2500 Funkmodul ausgestattet ist, zu senden.

Das Funkmodul ist mit 3,3V anzusteuern und hat einen maximalen Stromverbrauch von ungefähr 22 mA im Sendemodus. Das Modul besitzt ein 4-Channel SPI-Interface, welches ich verwendet habe, um mit dem Modul zu kommunizieren.

Das Modul, das ich verwendet habe, ist eine Platine, welche das Funkmodul bereits installiert hat, wichtige Schaltungen rund um den Chip besitzt und alle wichtigen Pins nach außen geführt hat. Diese Variante des Funkmoduls habe ich auf E-Bay gefunden und gekauft.



Abbildung 13: Ansicht CC2500

## 3 General Characteristics

Parameter	Min	Typ	Max	Unit	Condition/Note
Frequency range	2400		2483.5	MHz	There will be spurious signals at $n/2$ -crystal oscillator frequency ( $n$ is an integer number). RF frequencies at $n/2$ -crystal oscillator frequency should therefore be avoided (e.g. 2405, 2418, 2431, 2444, 2457, 2470 and 2483 MHz when using a 26 MHz crystal).
Data rate	1.2 1.2 26		500 250 500	kBaud kBaud kBaud	2-FSK GFSK and OOK (Shaped) MSK (also known as differential offset QPSK) Optional Manchester encoding (the data rate in kbps will be half the baud rate).

Abbildung 14: Generell Informationen CC2500

Parameter	Min	Max	Unit	Condition/Note
Supply voltage	-0.3	3.9	V	All supply pins must have the same voltage
Voltage on any digital pin	-0.3	VDD+0.3, max 3.9	V	
Voltage on the pins RF_P, RF_N and DCOUPL	-0.3	2.0	V	
Voltage ramp-up rate		120	kV/ $\mu$ s	
Input RF level		+10	dBm	
Storage temperature range	-50	150	°C	
Solder reflow temperature		260	°C	According to IPC/JEDEC J-STD-020D
ESD		<500	V	According to JEDEC STD 22, method A114, Human Body Model

Abbildung 15: Electrical characteristics CC2500



### 3.2.3.2. LoRa ERIC9

Das LoRa (= long range) Funkmodul ERIC9 ist im Vergleich zum CC2500 sehr leistungsstark. Es sendet mit einer Leistung von bis zu 7 dBm. Je nach Modell und Einstellung eignet sich das Funkmodul für Europa oder die USA. Ich verwende den für Europa zugelassenen 868MHz Kanal. Die Daten, die mit dem Funkmodul gesendet werden, werden zuerst vom onboard-Chip gelesen. Sobald alle Daten (maximale Daten pro Sendevorgang sind 256 Bytes) vom Chip empfangen sind, verpackt und verschlüsselt der Chip die Daten. Die Verschlüsselung ist zuständig für die Garantie, dass alle Daten sicher ankommen.

Der ERIC9 ist mit maximal 6 Volt zu versorgen, wobei die Datenleitungen nur ein Maximum von 3,3 Volt bewältigen können. Die Ansteuerung dieses Moduls erfolgt über zwei Datenleitungen, welche man mit einer SCI oder UART verbinden kann. Dabei ist auf die möglichen Spannungsdifferenzen der Controller zu achten.

Das Package, das ich verwendet habe, ist das Standardpackage des ERIC9. Es hat 24 Pads, für die ich ein Package erstellt habe, welches ich dann bei der Platine MOD-Z (4.6.1 und 4.6.2) verwendet habe.



Abbildung 16: Ansicht ERIC9

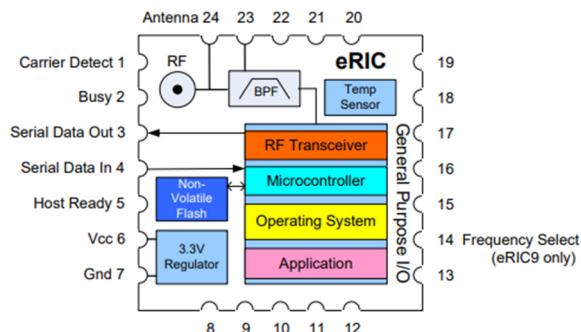


Figure 1 eRIC Transceiver Block Diagram

Abbildung 18: Aufbau ERIC9

DC Parameters	Pin	Min	Typical	Max	Units	Notes
Supply Voltage (Vcc)	6	2.4	3.6	6.0	Volts	
Internal Regulator (Vreg)		2.95	3.3	3.65	Volts	
Transmit supply current	6		32	33	mA	+10dBm RF power output
Receive supply current	6		15		mA	Continuous mode @ 250kbps
Sleep Mode current	6		1.8		uA	TBA

Abbildung 19: Electrical characteristics ERIC9

RF Parameters	24	50		Ohms	Via UFL connector or pads
Antenna Impedance	24	389	434.00	470	MHz
Operating Frequency		779	869.75	902	MHz
		902	915.00	928	MHz

Abbildung 17: Frequenzen ERIC9

### 3.2.4. Relais RT14730

Nach langem Überlegen, wie ich die Sicherungen auf dem Hof überwachen kann, damit ich weiß, wann ein Stromausfall stattgefunden hat, habe ich mich in Zusammenarbeit mit meinen Professoren für eine Relais-Schaltung (4.5) entschieden. Beim Relais RT14730 von Schrack handelt es sich um ein Relais, das durchschaltet, sobald an der Spule 230V anliegen. Die galvanisch getrennte zweite Seite, die durch die Spule durchgeschaltet wird, dient als Öffner. Sobald die Spannung an der Spule zusammenbricht, sperrt der Schalter und die Schaltung (die schaut, ob der Schalter geöffnet ist) weiß, dass die Spannung nicht mehr anliegt, und somit ist klar, dass die Sicherung nicht mehr aktiv ist.



Abbildung 20: Ansicht Relais

#### **Coil Data**

Coil voltage range, DC coil/ AC coil	5 to 110VDC / 24 to 230VAC
Operative range, IEC 61810	2
Coil insulation system according UL	class F

Abbildung 21: Informationen Spule Relais



### 3.2.5. Temperatursensor AD7415

Der Temperatursensor AD7415 ist ein sehr genauer digitaler Sensor. In meinem Projekt verwende ich den Sensor, um drei verschiedene Temperaturen zu messen. Einmal die Temperatur des Wassers in der Eingangsleitung am Hof, da die Temperaturen im Winter auf dem Hof sehr kalt sind und das Wasser bereits öfter gefroren ist. Zwei weitere Temperaturen, für die ich diese Sensoren verwendet habe, sind die Temperaturen für die zwei Belüftungskammern des Heus. Da das gedörnte Heu nach dem Einbringen mehrere Tage belüftet werden muss, um die Restfeuchtigkeit zu entziehen, war es mein Ziel, diese Temperaturen zu überwachen. Wenn das Heu nämlich nicht trocken ist, beginnt es zu gären, was dazu führt, dass sich Wärme im Inneren des Heustockes staut. Diese Wärme kann zu einem Brand führen, weshalb ich zwei Sensoren für die Überwachung der Temperaturen vorgesehen habe.

Der Temperatursensor kann Temperaturen von  $-40^{\circ}\text{C}$  bis  $+125^{\circ}\text{C}$  auf bis zu  $\pm 0,5^{\circ}\text{C}$  genau erfassen. Dabei benötigt er eine Versorgungsspannung von 2,7 bis 5,5 Volt. Da der Sensor mit einer I2C-Schnittstelle angesteuert wird, und mein Microcontroller mit 3,3 Volt betrieben wird, habe ich mich dazu entschieden, auch den Temperatursensor mit 3,3 Volt zu betreiben. Der Sensor besitzt lediglich 5 Pins für Versorgungsspannung, Masse, die zwei Datenleitungen und ein AS (Adress-Select-Pin zum Wählen zwischen 2 Adressen zum Ansteuern: HIGH oder LOW).

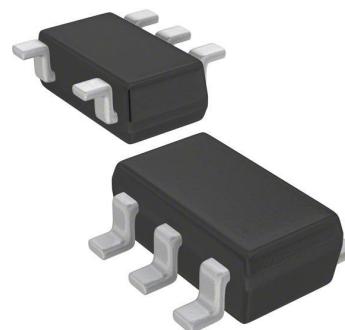


Abbildung 22: Ansicht AD7415

## ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
$V_{DD}$ to GND	-0.3 V to +7 V
SDA Input Voltage to GND	-0.3 V to +7 V
SDA Output Voltage to GND	-0.3 V to +7 V
SCL Input Voltage to GND	-0.3 V to +7 V
ALERT Output Voltage to GND	-0.3 V to +7 V
Operating Temperature Range	$-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$
Storage Temperature Range	$-65^{\circ}\text{C}$ to $+150^{\circ}\text{C}$
Junction Temperature	150°C

Abbildung 24: Electrical characteristics AD7415

## PIN CONFIGURATIONS AND FUNCTION DESCRIPTIONS

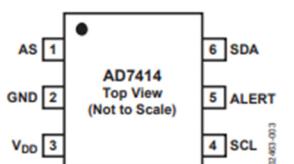


Figure 3. AD7414 Pin Configuration (SOT-23)

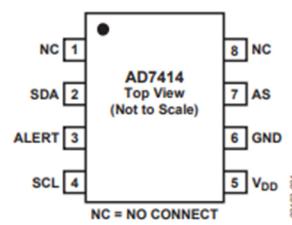


Figure 4. AD7414 Pin Configuration (MSOP)

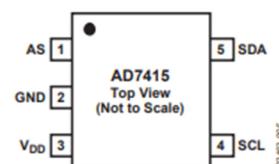


Figure 5. AD7415 Pin Configuration (SOT-23)

Abbildung 23: Packages AD7415

### 3.2.6. Türsensor

Beim Türsensor habe ich mich für den 59140 von Littelfuse-Rees Sensors entschieden. Der 59140 ist ein Öffner, der, so lange die Tür wie vorgesehen geschlossen ist, durchschaltet, und im Moment des Öffnens den Kontakt unterbricht. Der Schaltmechanismus ist ein einfacher Magnetschalter, der durch das geeignete, nicht beigelegte Gegenstück geschaltet wird. Das heißt, der gesamte Schaltmechanismus ist mechanisch und dadurch langlebig.

Verwendet wird der Sensor von Platine MOD-W (4.3) und MOD-X (4.4).



Abbildung 25: Ansicht Türsensor

### Electrical Ratings

Contact Type			Normally Open	Normally Open High Voltage	Change Over	Normally Closed
Switch Type			1	2	3	4
Contact Rating <sup>1</sup>		VA/Watt - max.	10	10	5	5
Voltage <sup>4</sup>	Switching <sup>2</sup> Breakdown <sup>3</sup>	Vdc - max. Vac - max. Vdc - min.	200 140 250	300 265 400	175 120 200	175 120 200
Current <sup>4</sup>	Switching <sup>2</sup> Carry	Adc - max. Aac - max. Adc - max.	0.5 0.35 1.2	.40 0.30 1.4	0.25 0.18 1.5	0.25 0.18 1.5
Resistance <sup>5</sup>	Contact, Initial Insulation	$\Omega$ - max. $\Omega$ - min.	0.2 $10^{10}$	0.2 $10^{10}$	0.2 $10^9$	0.2 $10^9$
Capacitance	Contact	pF - typ.	0.3	0.2	0.3	0.3
Temperature	Operating	°C	-40 to +105	-20 to +105	-40 to +105	-40 to +105

Abbildung 26: Electrical characteristics Türsensor

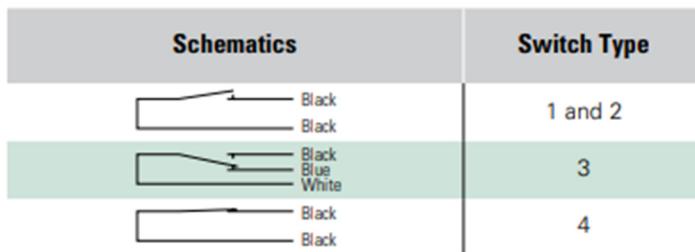


Abbildung 27: Ersatzschaltung Türsensor

### 3.2.7. Wasserdrucksensor Sensata 116CP31-M10S3-50

Für eine zusätzliche Kontrolle der Wasserversorgung der Tiere wollte ich den Wasserdruck überprüfen, um ein eventuelles Unterdruckproblem sofort festzustellen. Nach einigen Recherchen auf Mouser.it und Farnell.it ist mir der Wasserdrucksensor von Sensata ins Auge gesprungen. Je nach Sensortyp kann man bis zu 16 bar Druck messen. Da bei mir auf dem Hof standartmäßig 6-7 bar Druck anliegen, habe ich mich für eine Variante entschieden, welche bis zu 10 bar Druck messen kann.

Der Wasserdrucksensor ist mit einer Versorgungsspannung von 5 Volt und Masse zu versehen. Der Wasserdruck wird mit der dritten Leitung über eine analoge Spannung ausgegeben. Dabei variiert die ausgegebene Spannung zwischen 0,5 und 4,5 Volt.



Abbildung 28: Ansicht  
Wasserdrucksensor

#### Electrical

<b>Supply Voltage</b>	5VDC (ratiometric) or 8-30VDC (voltage regulated)
<b>Output Voltage</b>	Commonly 0.5 to 3.5V or 0.5 to 4.5V (Others available under ordering options)
<b>Supply Current</b>	8.0mA
<b>Output Current</b>	5.0mA (max)
<b>Output Load</b>	10kOhm pull down
<b>Output Response Time</b>	<10ms
<b>Oversupply Protection</b>	±40V
<b>Reverse Voltage</b>	-14V
<b>Short Circuit Protection</b>	Up to 40V
<b>EMC</b>	EMC Conforms to EN 61326-2-3 for sensors
<b>ESD</b>	Conforms to IEC 61000-4-2 Level 3, 4kV contact / 8kV air
<b>Pressure Ranges</b>	0-4 to 0-16bar (0-60 to 0-232 PSI)

Abbildung 29: Electrical characteristics Wasserdrucksensor

### 3.3. Wichtige Bauteile

#### 3.3.1. Microcontroller MC9S08LL16

Bei meinem Projekt habe ich sehr viele verschiedene Bauteile benötigt. Eines der wohl wichtigsten Bauteile ist der Mikrocontroller, welcher auf den drei Hauptplatten zu finden ist.

Beim Microcontroller, den ich verwendet habe, handelt es sich um einen MC9S08LL16 der HCS08 CPU-Familie. Ich habe mich für diesen Microcontroller entschieden, da in Zeiten wie diesen die Preise für Microprozessoren und für fast alle elektronischen Bauteile, die Silizium enthalten, drastisch angestiegen sind. Dieser Microcontroller, der eigentlich für die Ansteuerung eines LCD-Bildschirmes geeignet ist, ist sehr günstig und gleichzeitig leistungsstark. Er verfügt über alle wichtigen Peripheriegeräte wie einen ADC, eine SCI, SPI, eine I2C-Schnittstelle und vieles mehr. Man kann den Microcontroller einmal im 48- und einmal im 64-Pin Package kaufen, wobei ich mich für die 64-Pin Variante entschieden habe. Man hat die Option für einen externen Quarz von bis zu 16 MHz, um die Busfrequenz des Microcontrollers sehr genau einstellen zu können.



Abbildung 30: Ansicht Microcontroller

Table 1. MC9S08LL16 Series Features by MCU and Package

Feature	MC9S08LL16		MC9S08LL8
Package	64-pin LQFP	48-pin QFN/LQFP	48-pin QFN/LQFP
FLASH	16,384 (Dual 8K Arrays)		10,240 (8K and 2K arrays)
RAM	2080	2080	2080
ACMP	yes	yes	yes
ADC	8-ch	8-ch	8-ch
IIC	yes	yes	yes
IRQ	yes	yes	yes
KBI	8	8	8
SCI	yes	yes	yes
SPI	yes	yes	yes
TPM1	2-ch	2-ch	2-ch
TPM2	2-ch	-	-
TOD	Yes	Yes	Yes
LCD	8x24 4x28	8x16 4x20	8x16 4x20
I/O pins <sup>1</sup>	38	31	31

Abbildung 31: Ausstattung Microcontroller



## Devices in the MC9S08LL16 Series

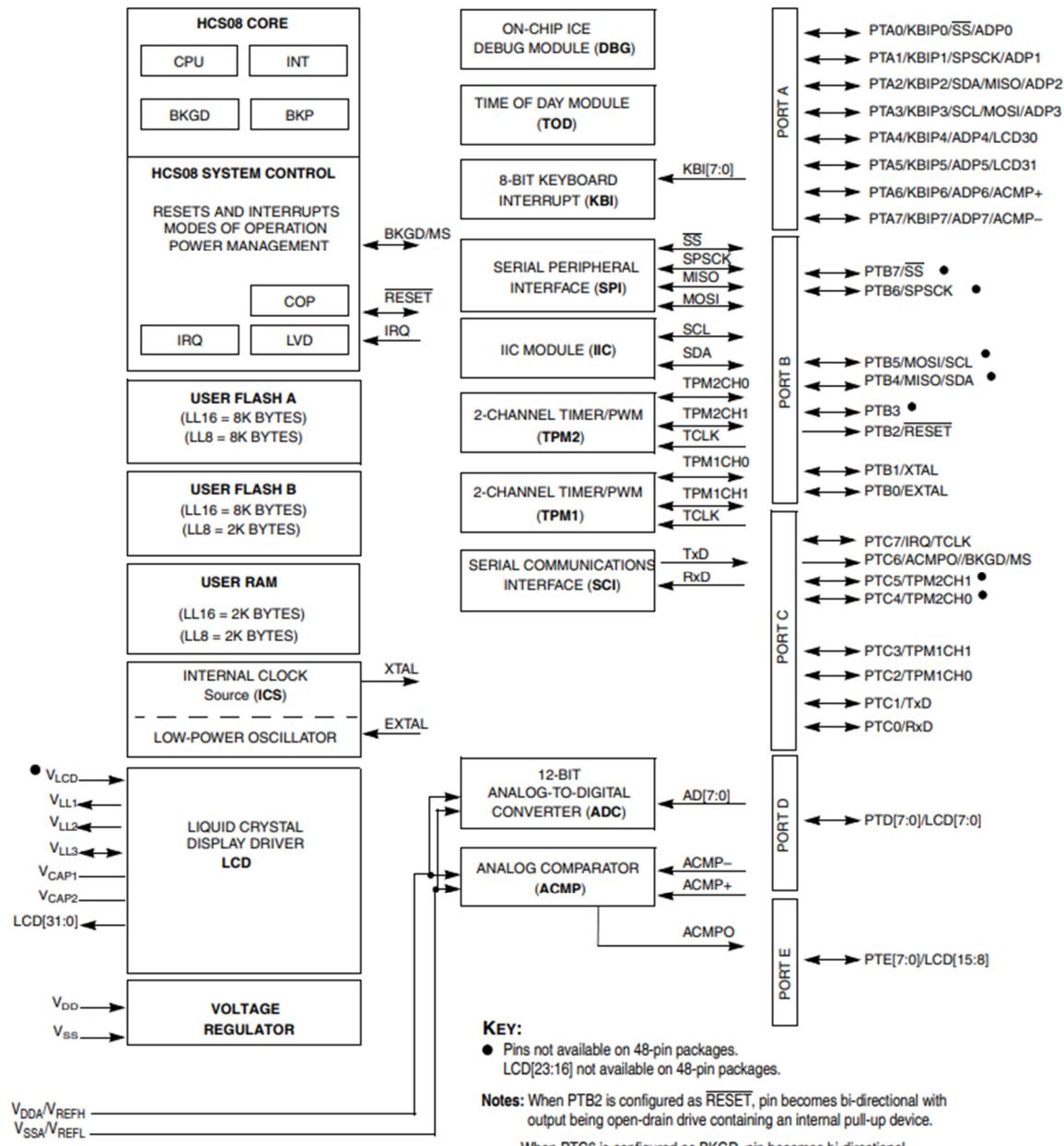


Figure 1. MC9S08LL16 Series Block Diagram

Abbildung 32: Blockdiagramm Microcontroller

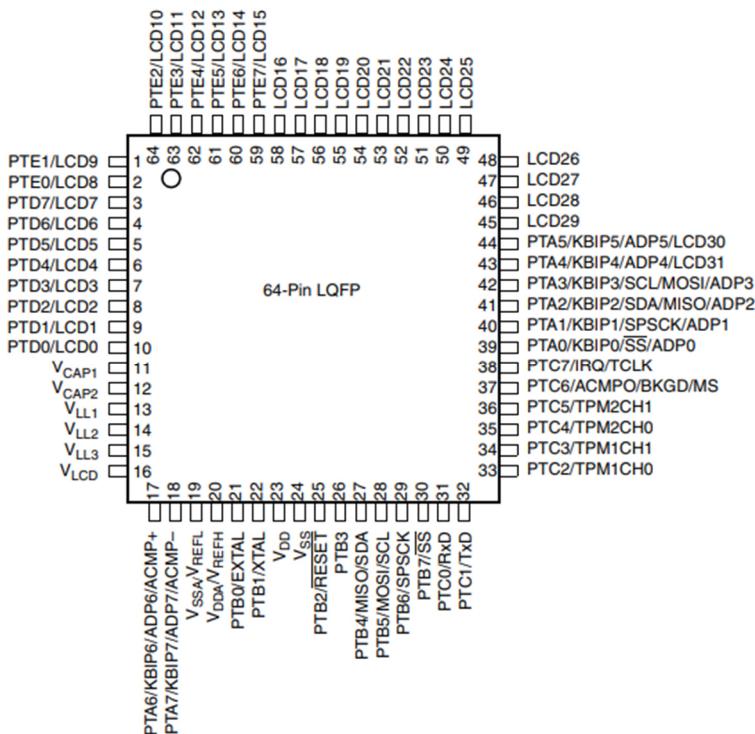


Abbildung 33: Pinbelegung Microcontroller

**Table 4. Absolute Maximum Ratings**

Rating	Symbol	Value	Unit
Supply voltage	V <sub>DD</sub>	-0.3 to 3.8	V
Maximum current into V <sub>DD</sub>	I <sub>DD</sub>	120	mA
Digital input voltage	V <sub>In</sub>	-0.3 to V <sub>DD</sub> + 0.3	V
Instantaneous maximum current Single pin limit (applies to all port pins) <sup>1, 2, 3</sup>	I <sub>D</sub>	± 25	mA
Storage temperature range	T <sub>stg</sub>	-55 to 150	°C

Abbildung 34: Electrical characteristics Microcontroller

### 3.3.2. Raspberry-Pi4 mit Display

Der Raspberry-Pi4 ist das zweite Herzstück meines Projektes nach dem Microcontroller. Das Raspberry-Pi ist ein Computer mit einer sehr großen Leistung im Verhältnis zu seiner Größe. Er hat 40 GPIO-Pins mit welchen man I2C, SPI, UART und vieles mehr ansteuern kann, die man aber auch einfach als „General-Purpose-Input-Output-Pins“ verwenden kann. In meinem Fall verwende ich das Raspberry-Pi auf zwei Arten. Ein Raspberry ist auf dem Hof lokalisiert und ist für das Sammeln der Daten zuständig. Die gesammelten Daten werden dann vom Raspberry verarbeitet und weiter an den zweiten Raspberry gesendet, welcher dann die Daten abspeichert und auf einem Display visualisiert. Für diese Verwendung ist die Ansteuerung der zwei Funkmodule wichtig. Wie bereits in Kapitel 2.3.2 erwähnt, verwendet das SoRa Funkmodul eine SPI-Schnittstelle zum Datenaustausch und das LoRa Funkmodul eine UART/SCI-Schnittstelle. Das Display ist mit Touch ausgestattet und mit einem HDMI-Anschluss versehen. Dieser wird einfach mit dem Raspberry-Pi mikro-HDMI Anschluss verbunden.



Abbildung 35: Ansicht Raspberry-Pi



Abbildung 36: Raspberry-Pi mit Display

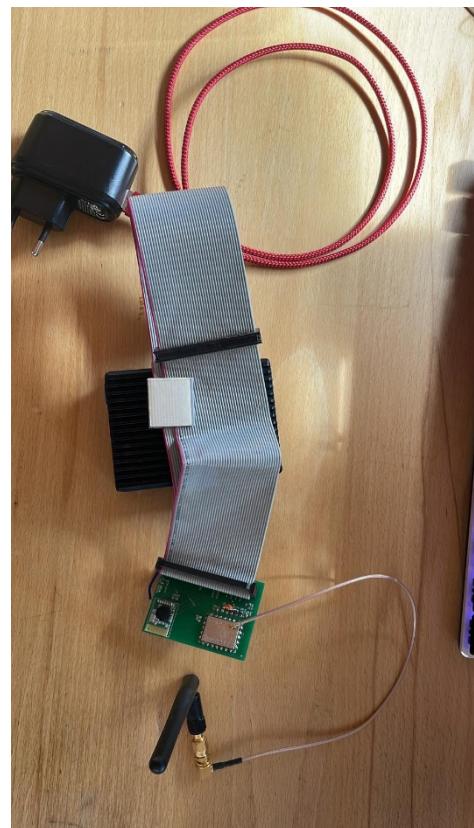


Abbildung 37: Raspberry-Pi mit Case und Funkmodul



### 3.3.3. LM324

Der LM324 ist ein einfacher Operationsverstärker, den ich verwende, um die Spannung, die der Wasserdrucksensor ausgibt, anzupassen. Die Ausgangsspannung des Wasserdrucksensors ist 0,5 – 4,5 Volt. Mein Microcontroller hat aber eine maximale Eingangsspannung von 3,3 Volt, was bei einer direkten Verkabelung ohne Pegelwandlung zu einer Zerstörung des angeschlossenen Pins führen würde. Aus diesem Grund habe ich den LM324 dazu verwendet, die Spannung an einen Pegel von 0 – 3,3 Volt anzupassen.

Ein großer Vorteil des LM324 ist, dass er sehr günstig ist. Ein großer Nachteil des LM324 ist seine sehr große Offsetspannung von 1,5 Volt, was dazu führt, dass die maximale Ausgangsspannung an allen Pins die Versorgungsspannung minus den 1,5 Volt beträgt. Man kann den OPV Bipolar (positive und negative Versorgungsspannung,  $U_{B, \max} = \pm 16V$ ) oder Unipolar (Positive Spannung und Masse,  $U_{B, \max} = 32V$ ) versorgen. Der maximale Ausgangstrom an jedem Pin beträgt um die 30 Milliampere.



Abbildung 38: Ansicht LM324

**MAXIMUM RATINGS** ( $T_A = +25^\circ\text{C}$ , unless otherwise noted.)

Rating	Symbol	Value	Unit
Power Supply Voltages Single Supply Split Supplies	$V_{CC}$ $V_{CC}, V_{EE}$	32 $\pm 16$	Vdc

Abbildung 39: Electrical characteristics LM324

### LM324, LM324A, LM324E, LM224, LM2902, LM2902E, LM2902V, NCV2902

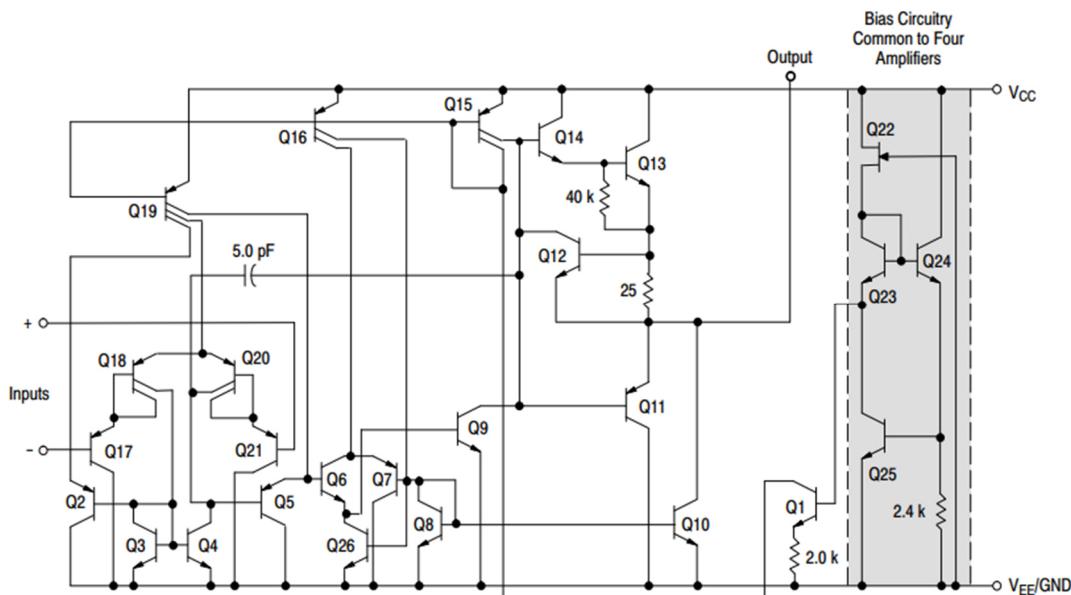


Figure 1. Representative Circuit Diagram  
(One-Fourth of Circuit Shown)

Abbildung 40: Ersatzschaltbild LM324

### 3.3.4. Spannungsregler LM1117-3.3, LM1117-5.0, LM317 und Li-Ion zu 3V3

Auf meiner Platine habe ich vier verschiedene Spannungsregler verwendet, um die Spannungen, die ich benötige, zu erzeugen.

Platine MOD-X und MOD-Y werden jeweils mit einer externen 12 Volt - Spannung versorgt. Die 12 Volt werden dann mit Hilfe der LM1117-5.0 zu 5 Volt dezimiert und danach vom LM1117-3.3 zu 3,3 Volt heruntergeregt.



Beide Spannungsregler sind vom Typ LM1117 und haben eine Drop-out-Spannung von circa 1,2 Volt.

Abbildung 41: Ansicht LM1117

## 7.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)<sup>(1)</sup>

	MIN	MAX	UNIT
Maximum Input Voltage ( $V_{IN}$ to GND)	20	V	
Power Dissipation <sup>(2)</sup>	Internally Limited		
Junction Temperature ( $T_J$ ) <sup>(2)</sup>	150	°C	
Storage Temperature, $T_{stg}$	-65	150	°C

- (1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
- (2) The maximum power dissipation is a function of  $T_{J(max)}$ ,  $R_{thJA}$ , and  $T_A$ . The maximum allowable power dissipation at any ambient temperature is  $P_D = (T_{J(max)} - T_A)/R_{thJA}$ . All numbers apply for packages soldered directly into a PCB.

Abbildung 42: Electrical Characteristics LM1117

PARAMETER		TEST CONDITIONS	MIN <sup>(1)</sup>	TYP <sup>(2)</sup>	MAX <sup>(1)</sup>	UNIT
$V_{IN}-V_{OUT}$	Dropout Voltage <sup>(4)</sup>	$I_{OUT} = 100 \text{ mA}$	$T_J = 25^\circ\text{C}$	1.1		
			over the junction temperature range $-40^\circ\text{C}$ to $125^\circ\text{C}$		1.3	V
		$I_{OUT} = 500 \text{ mA}$	$T_J = 25^\circ\text{C}$	1.15		
			over the junction temperature range $-40^\circ\text{C}$ to $125^\circ\text{C}$		1.35	V
		$I_{OUT} = 800 \text{ mA}$	$T_J = 25^\circ\text{C}$	1.2		
			over the junction temperature range $-40^\circ\text{C}$ to $125^\circ\text{C}$		1.4	V
$I_{LIMIT}$	Current Limit	$V_{IN} - V_{OUT} = 5 \text{ V}$ , $T_J = 25^\circ\text{C}$	800	1200	1500	mA

Abbildung 43: Dropout voltage LM1117



Auf meiner dritten Hauptplatine MOD-W verwende ich keine 12 Volt als Spannungsversorgung, sondern eine 3,9 Volt Li-Ion Batterie. Die 3,9 Volt der Li-Ion Batterie werden vom NCP114 zu 3,3 Volt geregtelt. Die 3,3 Volt versorgen dann den Microcontroller. Um zu verhindern, dass die Batterie leer wird, habe ich Solarpanele vorgesehen, welche durch eine einfache Schaltung mit dem Spannungsregler LM317 und einer Diode nachgeladen wird.



Abbildung 44: Ansicht NCP114

## BLOCK DIAGRAM

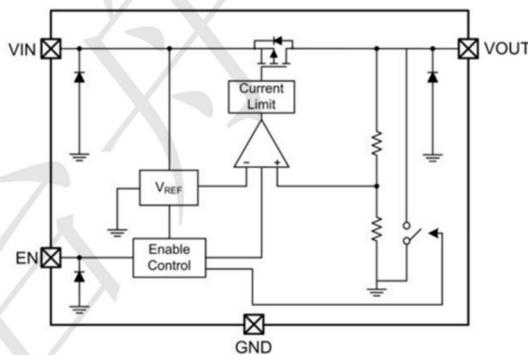


Abbildung 45: Ersatzschaltbild NCP114

Electrical Characteristics ( $T = 25^\circ\text{C}$  unless otherwise noted)(V<sub>OUT</sub> + 1 < V<sub>IN</sub> < 5.5V, T<sub>A</sub> = 25°C, unless otherwise specified)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Fixed Output Voltage Range	V <sub>OUT</sub>		0.8	--	3.45	V
DC Output Accuracy		I <sub>LOAD</sub> = 1mA	-2	--	2	%
		0.8V ≤ V <sub>OUT</sub> < 1.05V	--	0.7	0.97	
		1.05V ≤ V <sub>OUT</sub> < 1.2V	--	0.5	0.92	
		1.2V ≤ V <sub>OUT</sub> < 1.5V	--	0.4	0.57	
		1.5V ≤ V <sub>OUT</sub> < 1.8V	--	0.3	0.47	
Dropout Voltage (I <sub>LOAD</sub> = 300mA) (Note 5)	V <sub>DROP</sub>	1.8V ≤ V <sub>OUT</sub> < 2.1V	--	0.24	0.33	V
		2.1V ≤ V <sub>OUT</sub> < 2.5V	--	0.21	0.3	
		2.5V ≤ V <sub>OUT</sub> < 2.8V	--	0.18	0.25	
		2.8V ≤ V <sub>OUT</sub> < 3V	--	0.16	0.23	
		3V ≤ V <sub>OUT</sub>	--	0.15	0.2	
Dropout Voltage (I <sub>LOAD</sub> = 200mA) (Note 6)	V <sub>DROP</sub>	1.8V ≤ V <sub>OUT</sub> < 2.1V	--	0.16	0.2	V
V <sub>CC</sub> Consumption Current	I <sub>Q</sub>	I <sub>LOAD</sub> = 0mA, V <sub>OUT</sub> ≤ 5.5V V <sub>IN</sub> ≥ V <sub>OUT</sub> + V <sub>DROP</sub>	--	15	18	µA

Abbildung 46: Electrical characteristics NCP114

### 3.3.5. Solarpaneele

Um die Batterie von MOD-W vor einem vollständigen Entladen zu schützen, habe ich auf der Platine eine Schaltung vorgesehen, mit deren Hilfe ich die Batterien nachladen kann. Die benötigte Spannung und den Strom, den ich zum Nachladen brauche, liefern mir zwei in Reihe geschaltete 5 Volt Solarpaneele, welche einen maximalen Strom von circa 60 Milliampere haben.

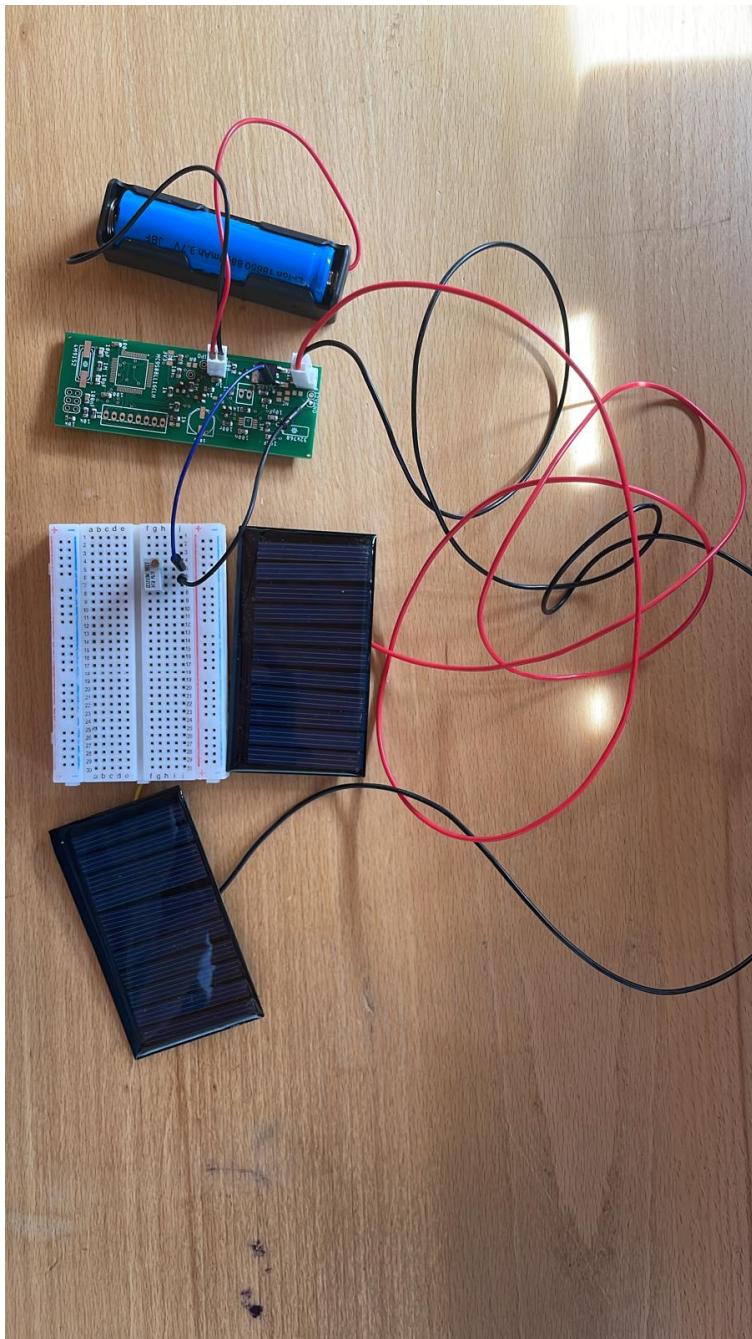


Abbildung 47: Testschaltung zum Laden der Batterien

### 3.3.6. Uhrenbaustein PCF8563

Der Uhrenbaustein PCF8563 ist auf der Platine MOD-W zu finden. Es handelt sich dabei um einen Real-Time Baustein, welcher nach der Programmierung einfach wie eine Uhr funktioniert. Er hat integrierte Register, welche die Zeiten in Sekunden, Minuten, Stunden usw. bis zu Jahren abspeichern.

Der Baustein hat optionale Alarmregister und Timerregister, welche man programmieren kann, um zum Beispiel einen Alarm bei einer gewissen Uhrzeit oder nach einer vorprogrammierten Zeit auslöst. Der Alarm kann nur intern durch ein „Flag“ dargestellt werden, oder durch einen invertierten Interrupt-Pin, der so lange „floating“ ist, bis ein „Interrupt“ auslöst, und dann auf Masse zieht.

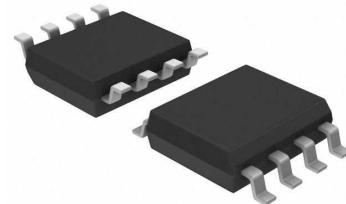


Abbildung 48: Ansicht Uhrenbaustein

## 2. Features and benefits

- Provides year, month, day, weekday, hours, minutes, and seconds based on a 32.768 kHz quartz crystal
- Century flag
- Clock operating voltage: 1.0 V to 5.5 V at room temperature
- Low backup current; typical 0.25  $\mu$ A at  $V_{DD} = 3.0$  V and  $T_{amb} = 25$  °C
- 400 kHz two-wire I<sup>2</sup>C-bus interface (at  $V_{DD} = 1.8$  V to 5.5 V)
- Programmable clock output for peripheral devices (32.768 kHz, 1.024 kHz, 32 Hz, and 1 Hz)
- Alarm and timer functions
- Integrated oscillator capacitor
- Internal Power-On Reset (POR)
- I<sup>2</sup>C-bus slave address: read A3h and write A2h
- Open-drain interrupt pin

Abbildung 49: Anwendung und Funktion des Uhrenbausteins

## 6. Block diagram

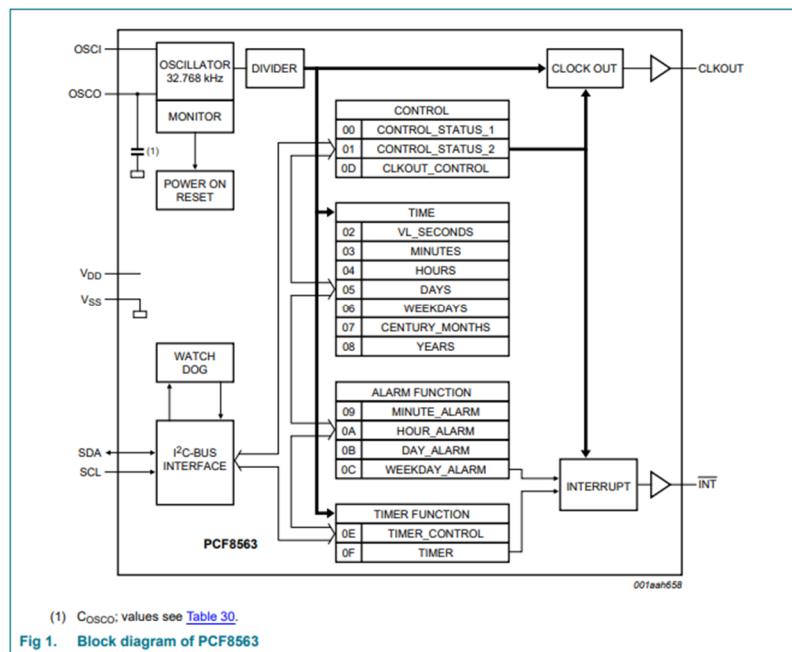


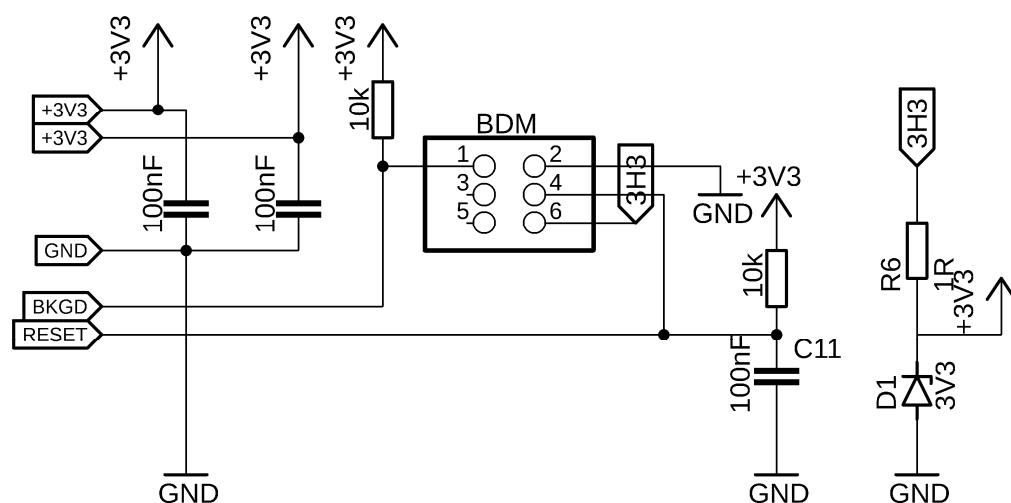
Abbildung 50: Blockschaltbild Uhrenbaustein

## 4. Platinen

### 4.1. Überblick

Insgesamt habe ich sechs verschiedene Platinen entwickelt, welche jeweils eine andere Arbeit erledigen sollen. Drei der sechs Platinen sind Hauptplatinen, was bedeutet, sie werden mit einem Microcontroller bestückt. Drei weitere haben keinen Microcontroller und sind somit Erweiterungen von anderen Platinen.

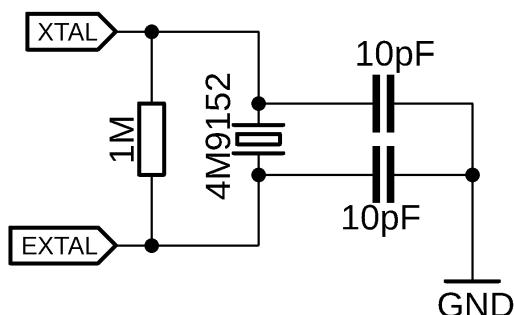
Für die drei Platinen mit dem Microcontroller war es wichtig, die Programmierschaltung für den Microcontroller mit einzubauen. Diese besteht aus einem sechs-Pol-Stecker für das Programmiergerät, einer Spannungsversorgung mit Plus und Masse, einer Leitung für die Programmierung und einer Resetleitung. Zusätzlich habe ich eine Zenerdiode vorgesehen, um den Microcontroller vor eventuellen Fehlbeschaltungen durch das Programmiergerät zu schützen.



© Programmierschaltung Auer Andreas

Abbildung 51: Programmierschaltung

Der zweite wichtige Teil einer jeden Schaltung mit Microcontroller ist der Quarz. Der Quarz ist zwar nicht zwingend notwendig, aber sollte man zum Beispiel mit seriellen Schnittstellen arbeiten, bei denen Timing elementar ist, dann ist ein Quarz unumgänglich.



© Quarzbeschaltung Auer Andreas

Abbildung 52: Quarzbeschaltung

## 4.2. MOD-V

MOD-V ist wohl meine einfachste Platine. Die Platine ist nur dazu da, die 50mil Abstände zwischen den Pads des Funkmoduls zu erweitern. Das Funkmodul CC2500 wird dann auf die Platine aufgelötet und anschließend mit einem acht-Pol-Stecker der Marke MOLEX versehen. Jede Hauptplatine wird mit einer dieser Platinen versehen, um Zugang zu einem Funkmodul zu haben.

### 4.2.1. MOD-V\_01

Bei MOD-V habe ich nur eine Version planen müssen, da die mögliche Fehlerquelle gleich null ist. Wie im Bild sichtbar, kann man die Platine mit dem Funkmodul versehen und einen zusätzlichen Stecker auflöten.

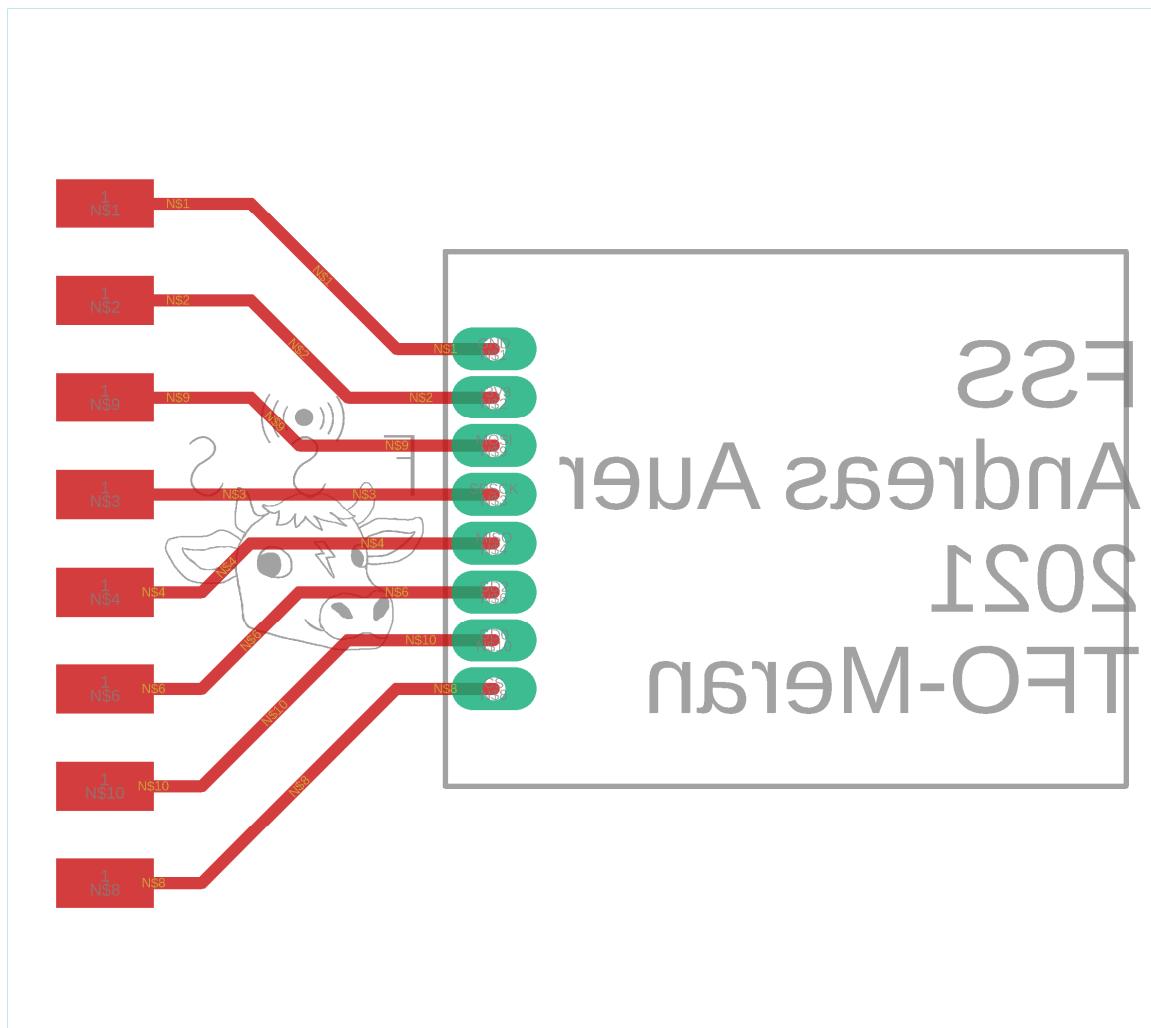


Abbildung 53: MOD-V PCB

### 4.3. MOD-W

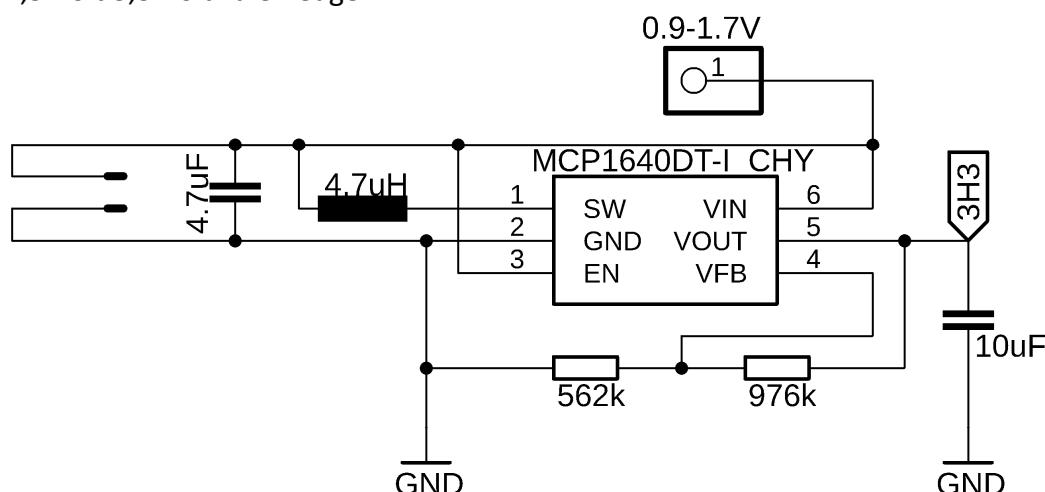
MOD-W ist eine meiner drei Hauptplatinen. Meiner Ansicht nach war es die am interessantesten zu planende Platine. Sie hat sich mit den Versionen teilweise stark verändert und mit Version 3 noch einmal einen totalen Umschwung erhalten.

Die generelle Funktion der Platine war es, erst darauf zu warten, bis der angeschlossene Türsensor öffnet (also wird die Tür, mit der der Sensor verbunden ist, geöffnet) und daraufhin den Microcontroller mit Spannung (GND) versorgt. Diese Aufgabe habe ich erst mit Dioden und Transistoren zu lösen versucht. Nach der ersten Bestellung und dem ersten Testen wurde jedoch schnell klar, dass diese Variante wegen den „floating“ Spannungen durch die ganzen Dioden und Transistoren auf der Platine nicht funktioniert. Die Spannungsversorgung war eine AA-Batterie mit circa 1,5 Volt.

Bei Version 3 habe ich mich dann entschieden, anstatt der nicht aufladbaren AA-Batterie, Lithium-Ionen-Batterien zu verwenden. Diese Batterien liefern um die 3,9V und sind wiederaufladbar was für meine neue Idee von äußerster Wichtigkeit war. Die neue Version der Platine ist nun in drei Sektoren unterteilt. Einmal gibt es den 3,9 Volt Bereich mit der gesamten Sensorik und der Batterie, dann gibt es einen Spannungsregler, der aus 3,9 Volt 3,3 Volt erzeugt, was bedeutet, dass er eine sehr geringe Drop-out-Spannung (Spannungsabfall bei der Spannungsregulierung, Eingangsspannung muss um diese Spannung größer sein) braucht. Auch benötigte ich einen Enable-Pin, da ich die 3,3 Volt nur in gewissen Momenten erzeugen will, um den Microcontroller generell ausgeschaltet zu lassen. Bei Aktivierung des Türsensors oder bei einem Interrupt des Uhrenbausteins wird der Enable aktiviert und der Microcontroller eingeschaltet. Zusätzlich gibt es noch einen Spannungsregler, der aus 5 - 12V um die 3,9 Volt erzeugt, womit die Batterie bei einer niedrigeren Spannung als 3,9 Volt geladen wird. Zur Sicherheit gibt es eine Diode. Die 5 - 12 Volt werden von den Solarpaneelen erzeugt und können die Batterie mit bis zu 30 Milliampere laden.

#### 4.3.1. MOD-W\_V01

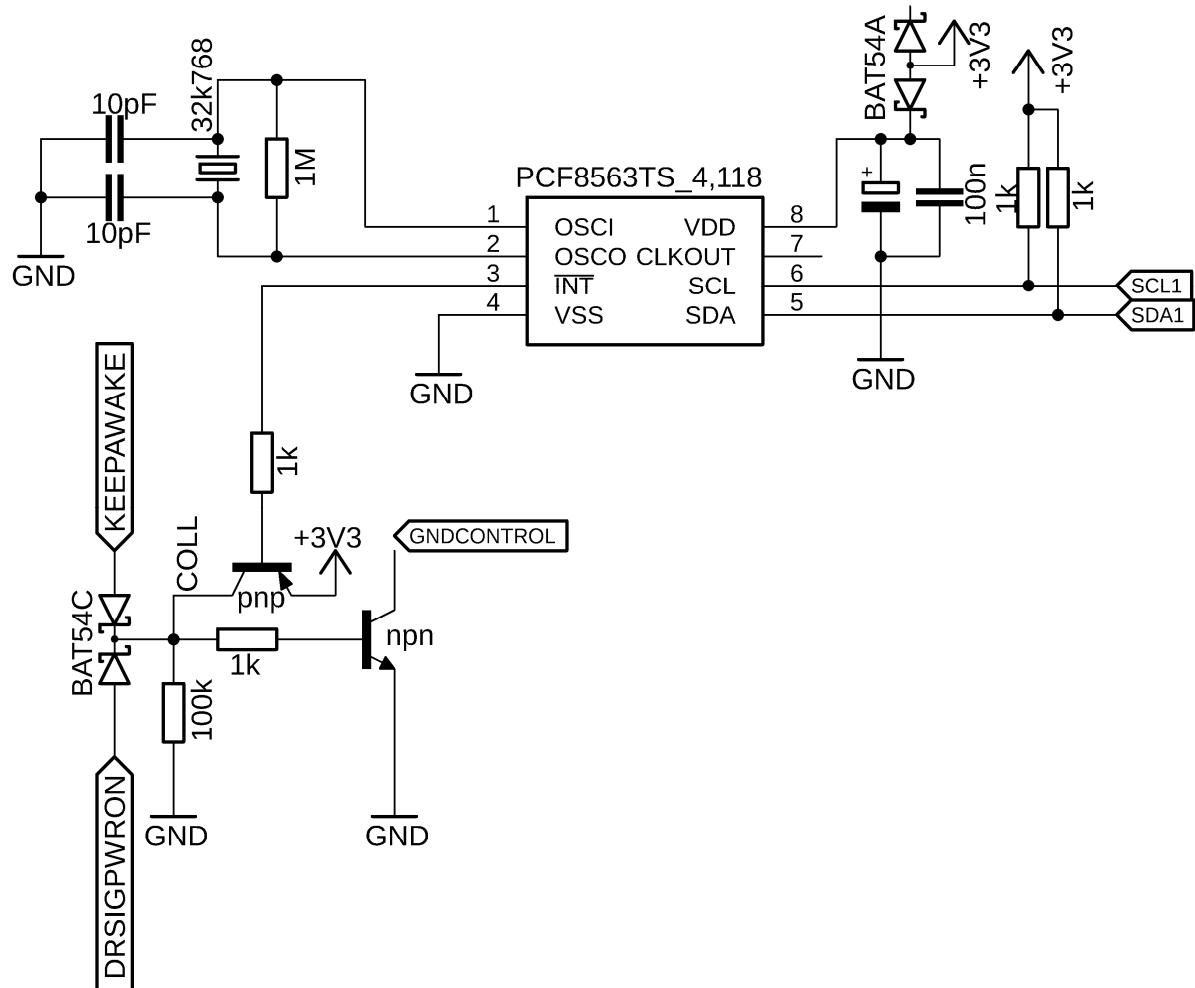
Die erste Grundform der Platine. Wie im Bild gut sichtbar sind viele Transistoren und Dioden vorhanden, welche meine erste Idee zum Scheitern verurteilt haben. Auch die Spannungsversorgung war hier speziell, da ich einen Step-Up Regler verwendet habe, um aus 1,5 Volt 3,3 Volt zu erzeugen.



© Spannungsregelung 1,5V to 3,3V Auer Andreas

Abbildung 54: Spannungsregelung 1,5V zu 3,3V

Die wichtigste Schaltung war jedoch die Aktivierungs- und Kontrollschaltung für den Microcontroller. Dabei haben der Türsensor, der Uhrenbaustein und der Microcontroller selbst mit Hilfe von Transistoren und Dioden die Möglichkeit, den Microcontroller einzuschalten.

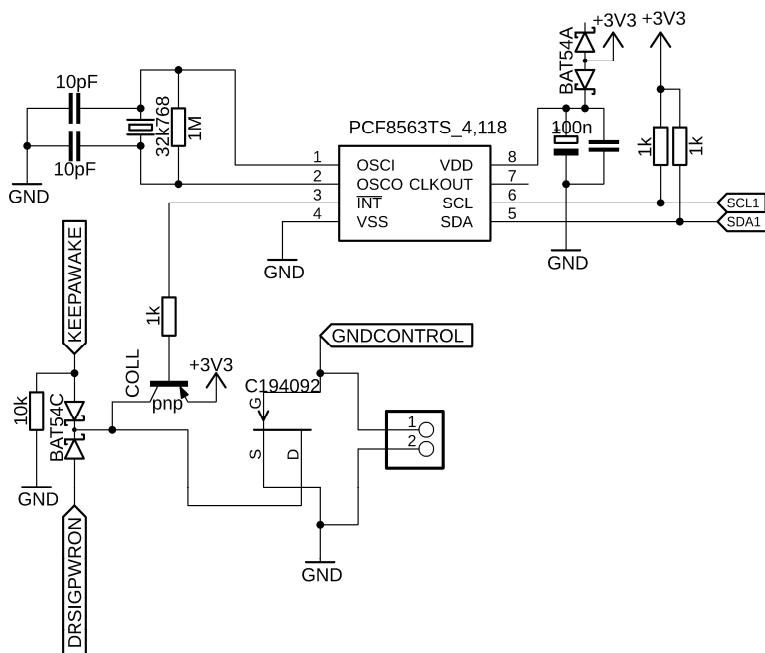


© Uhrenbaustein PCF8563 und Kontrollschaltung Auer Andreas

Abbildung 55: Uhrenbaustein und Kontrollschaltung V01

#### 4.3.2. MOD-W\_V02

Um die Probleme, die durch die Transistoren in Version eins entstanden, zu vermeiden, habe ich in Version zwei mit einem Mosfet gearbeitet. Bei Version eins waren immer wieder Restspannungen vorhanden, die die Funktion der Platine beeinträchtigt haben. Mit der Benutzung eines Mosfets habe ich versucht, diese Probleme weitgehend zu beseitigen.

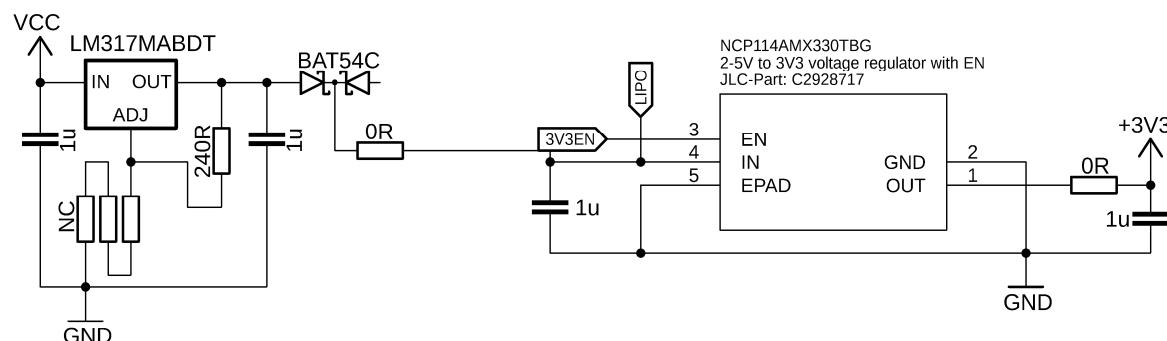


© Uhrenbaustein und Kontrollschaltung Auer Andreas  
Abbildung 56: Uhrenbaustein und Kontrollschaltung V02

#### 4.3.3. MOD-W\_V03

Bei Version drei ist der totale Umschwung in der Platine gut sichtbar. Die gesamte Strukturierung ist neugestaltet worden, sowie die meisten der Bauteile. Auf der linken Seite sieht man den LM317 für die Spannungsregelung von 5 - 12 auf 3,9 Volt. In der Mitte und links oben sieht man die 3,9 Volt mit dem Batterieanschluss, dem Uhrenbaustein und dem Türsensor. Auf der rechten Seite sieht man den Microcontroller, dessen Programmierschaltung und den Stecker für das Funkmodul.

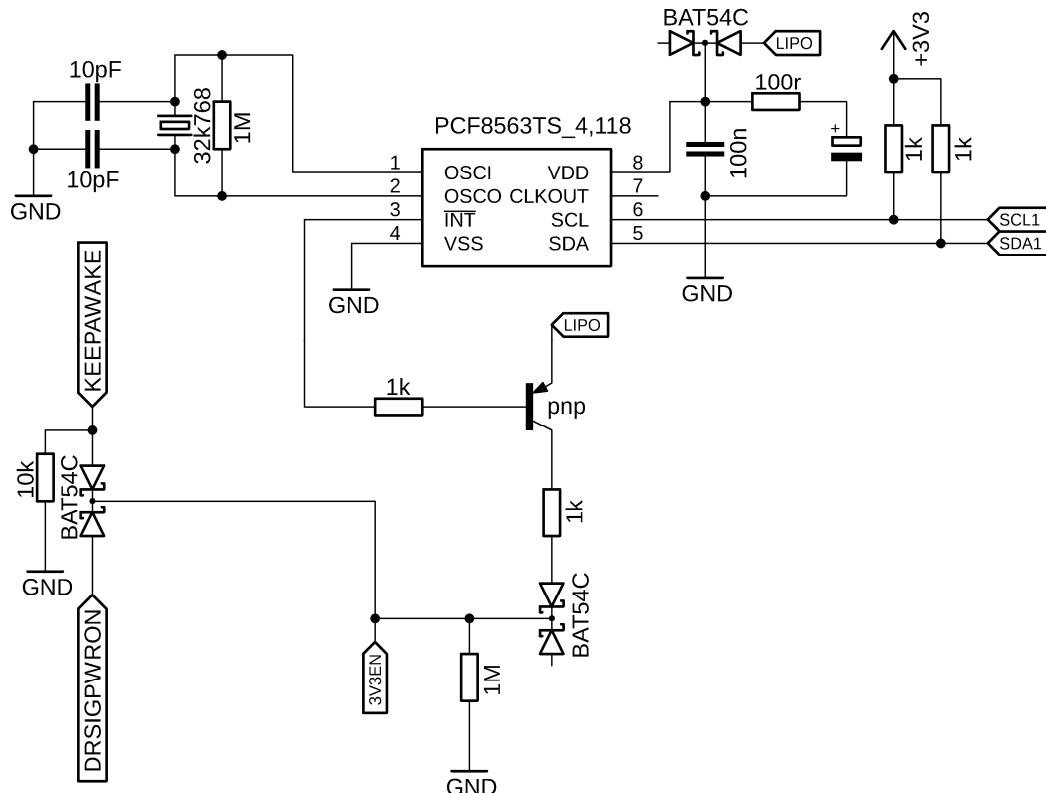
Wegen der Verwendung der Lithium-Ionen-Batterien habe ich meinen Plan geändert und bin auf den neuen Spannungsregler umgestiegen. Dieser hat eine komplett neue Schaltung. In Kombination mit dem LM317 für die Ladeschaltung der Batterie ist folgende Schaltung entstanden.



© Spannungsregelung 12V - 3,9V - 3,3V Auer Andreas  
Abbildung 57: Spannungsregelung 12V - 3,9V - 3,3V



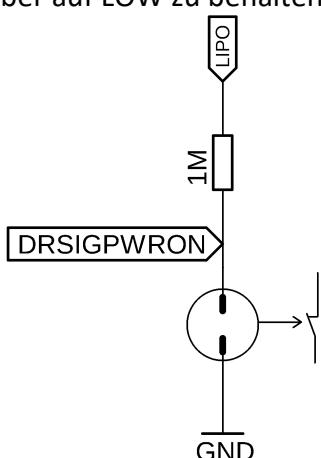
Auch die Schaltung mit dem Uhrenbaustein und die Aktivierungsschaltung des Microcontrollers habe ich bedeutend geändert. Nun ist nur noch ein Transistor in Verwendung und natürlich mehrere Dioden zur Sicherheit, sowie einige Pull-Up- und Pull-Down-Widerstände, um eventuelle Spannungen abzuführen oder auf einem gewissen Niveau zu halten.



© Uhrenbaustein und Kontrollschaltung MOD-W\_V03 Auer Andreas

Abbildung 58: Uhrenbaustein und Kontrollschaltung V03

Das letzte Problem war es, den Türsensor, der ein Öffner ist, in die Schaltung zu integrieren und möglichst wenig Strom zu verbrauchen. Auf dieser Grundlage habe ich mich dazu entschieden, den Türsensor mit Masse zu verbinden, und einen Megohm Widerstand zu verwenden, um den Enable-Pin des Microcontrollers auf HIGH zu ziehen, ihn standartmäßig aber auf LOW zu behalten.



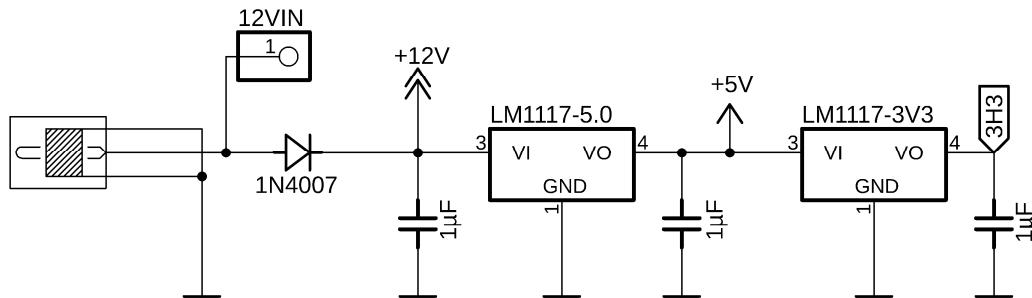
© Türsensor Auer Andreas

Abbildung 59: Türsensor



#### 4.4. MOD-X

Die Platine MOD-X ist die zweite Hauptplatine und dafür zuständig, die meisten der Sensoren anzusteuern. Dabei gibt es Datenleitungen für den BME680, den Temperatursensor, den Türsensor und den Wasserdrucksensor. Die Platine soll alle diese Sensoren ansteuern können und die ermittelten Daten via CC2500 Funkmodul an den Raspberry-Pi senden. Die Spannungsversorgung beträgt 12 Volt, welche zu 5 Volt und 3,3 Volt heruntergeregt werden.



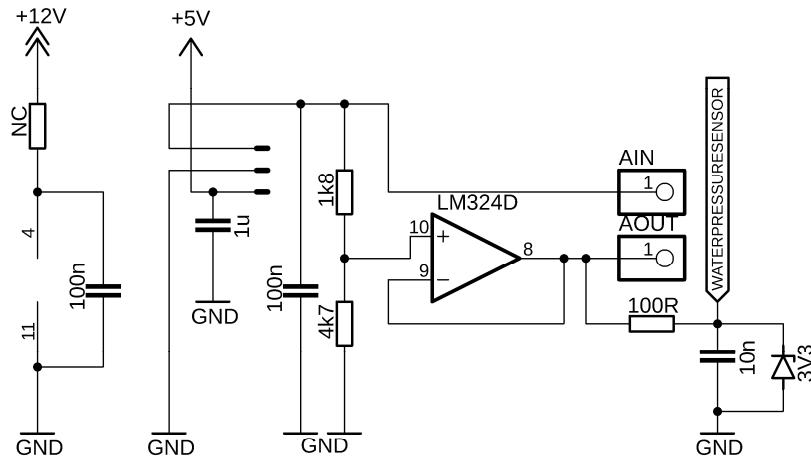
© Spannungsversorgung 12V - 5V - 3,3V Auer Andreas

Abbildung 60: Spannungsversorgung 12V - 5V - 3,3V

##### 4.4.1. MOD-X\_V01

Die Version eins dieser Platine ist die Version, die ich beim Prototyp verwendet habe. Die Platine ist nahezu fehlerfrei. Die Fehler, die dennoch existierten, habe ich in Version zwei beseitigt.

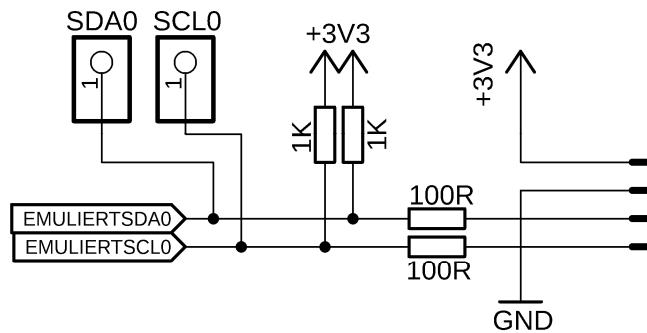
Auf dieser Platine habe ich die Möglichkeit vier Sensoren anzubauen. Dabei ist immer eine andere Schaltung notwendig. Die wohl interessanteste ist die Spannungsregelung für den Wasserdrucksensor. Der Wasserdrucksensor gibt eine Spannung von 0,5 bis 4,5 Volt aus. Diese Spannung habe ich dann auf einen Pegel bis auf 3,3 Volt angepasst.



© Pegelanpassung 4,5 Volt auf 3,3 Volt Auer Andreas

Abbildung 61: Pegelanpassung 4,5V zu 3,3V

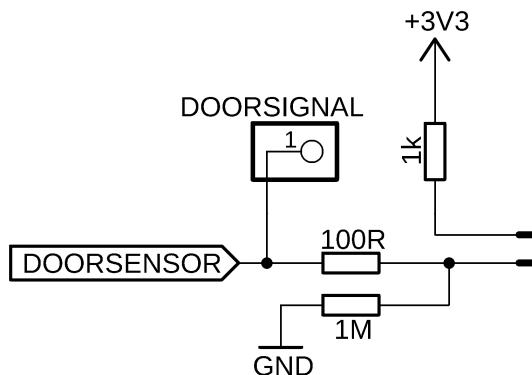
Für den Temperatursensor war eine Spannungsversorgung von Nöten, sowie eine I2C-Schnittstelle. Wie man auf der Schematic erkennen kann, habe ich die 100 Ohm-Widerstände falsch platziert, weshalb die I2C-Schnittstelle nicht funktioniert hat. Nach dem Austauschen mit 0 Ohm-Widerständen hat alles wunderbar funktioniert.



© Ansteureung AD7415 Auer Andreas

Abbildung 62: Ansteuerung AD7415

Die Ansteuerung des BME680 erfolgt über eine einfache SPI-Schnittstelle, also werden sechs Leiterbahnen zu einem Stecker gezogen, welcher dann mit dem BME680 verbunden wird. Dieselbe simple Lösung kann man beim Türsensor verwenden. Da der Türsensor ein Öffner ist, habe ich einen Pull-up Widerstand verwendet, der die Spannung am Microcontroller auf 3,3 Volt zieht, solange der Schalter geschlossen ist. Im geöffneten Zustand leitet ein Ein-Mega-Ohm Widerstand mögliche Restspannungen ab, um ein Massesignal am Microcontroller zu erzeugen.

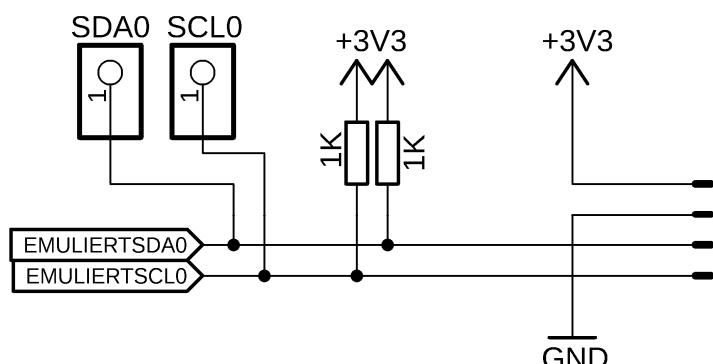


© Ansteureung Türsensor Auer Andreas

Abbildung 63: Ansteuerung Türsensor

#### 4.4.2. MOD-X\_V02

Version zwei ist die Platine mit den beseitigten Fehlern. Ich habe die Platine jedoch nicht bestellt, da die Fehler einfach manuell mit dem Austauschen einiger weniger Widerstände behoben werden konnte.



© Ansteureung AD7415 Auer Andreas

Abbildung 64: Ansteuerung AD7415

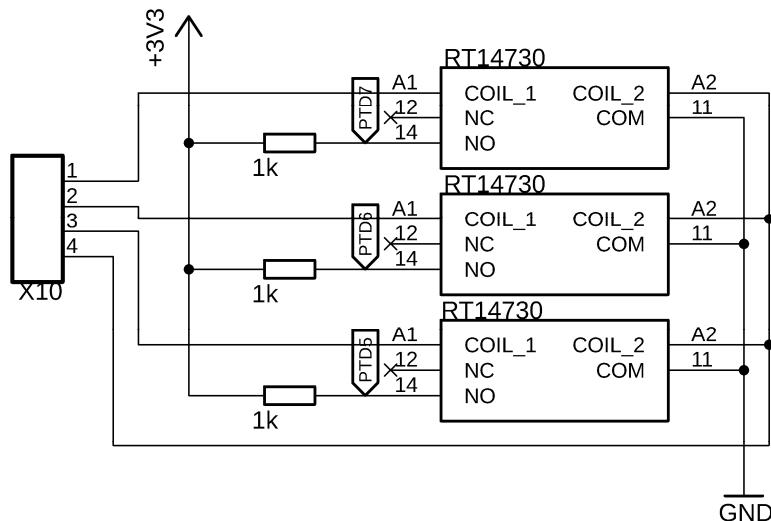
#### 4.5. MOD-Y

Die Platine MOD-Y ist die letzte Hauptplatine. Sie dient zur Ansteuerung der Stromsensoren und der Relais, welche die Sicherungen kontrollieren. Die gemessenen Daten werden dann wie bei MOD-X ausgewertet und mit dem Funkmodul an dem Raspberry-Pi gesendet. Die Spannungsversorgung sind einfache 12 Volt, welche in 5 Volt und 3,3 Volt umgewandelt werden (siehe MOD-X 4.4). Anfangs plante ich mit einer einzelnen Platine, welche ich dann in Version eins verwirklicht habe. Schnell stellte sich jedoch heraus, dass es besser ist, wenn ich die Platine in zwei Unterplatinen aufteile, um die Sicherheiten wegen den 230 Volt auf den Relais zu garantieren und bei einer eventuellen CE-Zertifizierung keine Probleme zu bekommen. Eine Unterplatine enthielt die kalte Seite mit dem Microcontroller und den Anschlüssen für die Stromsensoren, die andere die heiße Seite mit den 230-Volt-Relais. Die zwei Platinen werden mit einem vier-Pol-Stecker verbunden, welcher auf beiden Platinen vorgesehen ist.

##### 4.5.1. MOD-Y\_V01

Die erste Version der Platine stellte für mich ein Grundmodell dar. Sie beinhaltet die drei Anschlüsse für die Stromsensoren, sowie Platz für die neun Relais. Nach einem Zeichnen und dem Fertigstellen der ersten Version habe ich mich nach Absprache mit meinem Professor De Tomaso dafür entschieden, die Platine zweizuteilen.

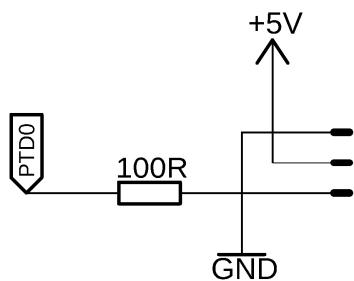
Auf dieser Platine hat die Sicherheit erste Priorität. Die Relais werden mit 230 Volt Wechselspannung durchgeschaltet, weshalb Gefahr durch Stromschläge besteht. Beim Erstellen des Schaltplans habe ich noch nicht an die späteren Probleme gedacht.



© Beschaltung Relaise Auer Andreas

Abbildung 65: Beschaltung Relais

Die Schaltung für den Stromsensor war weitgehend einfach. Der Sensor benötigt einfache 5 Volt Spannungsversorgung und gibt ein Analogsignal aus.



© Ansteuerung Stromsensor Auer Andreas

Abbildung 66: Ansteuerung Stromsensor

#### 4.5.2. MOD-Y\_V02

Der wohl größte Unterschied von Version eins auf Version zwei ist der Grundaufbau der Schaltung. Nachdem ich mich für zwei einzelne Schaltungen bei Version zwei entschieden habe, hat sich das PCB grundlegend geändert. Hier der Vergleich:

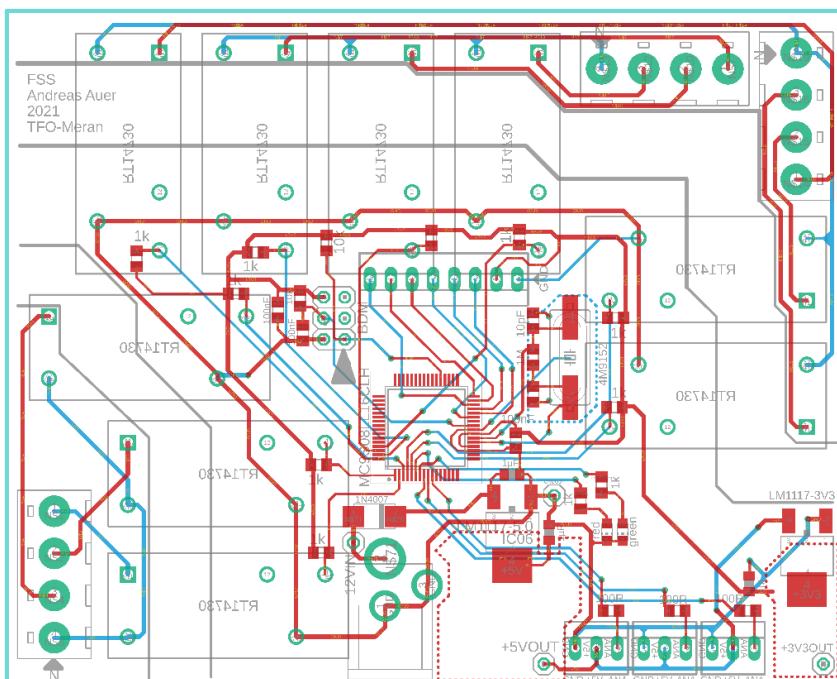


Abbildung 69: MOD-Y V01

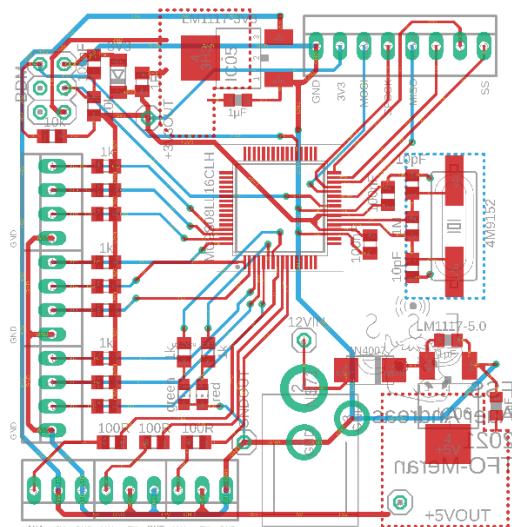


Abbildung 68: MOD-Y V02 kalte Seite

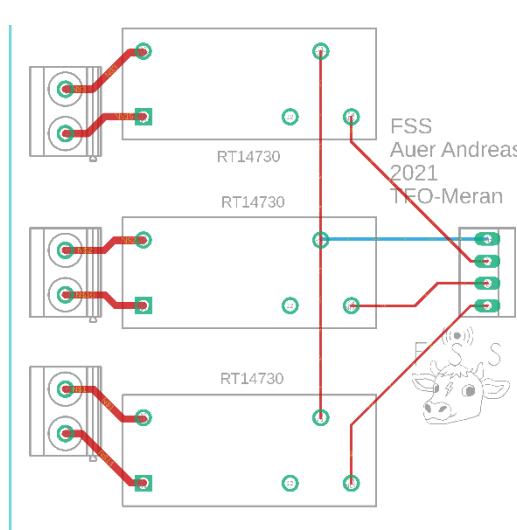


Abbildung 67: MOD-Y V02 heiße Seite

#### 4.5.3. MOD-Y\_V03

Als ich circa einen Monat nach dem Bestellen mit dem Schreiben des Codes für diese Platine begann, ist mir sofort aufgefallen, dass ich drei Leiterbahnen falsch gezogen hatte und somit die analogen Ausgänge der Stromsensoren mit digitalen Eingängen des Microcontrollers verbunden hatte. Mit Version drei dieser Platine habe ich dann diesen Fehler behoben.

Dies ist der Prototyp, welchen ich beim Testen der Gesamtschaltung verwendet habe. Bis jetzt sind noch keine Fehler aufgetreten und alle Sensoren und die Relais lassen sich einwandfrei ansteuern.

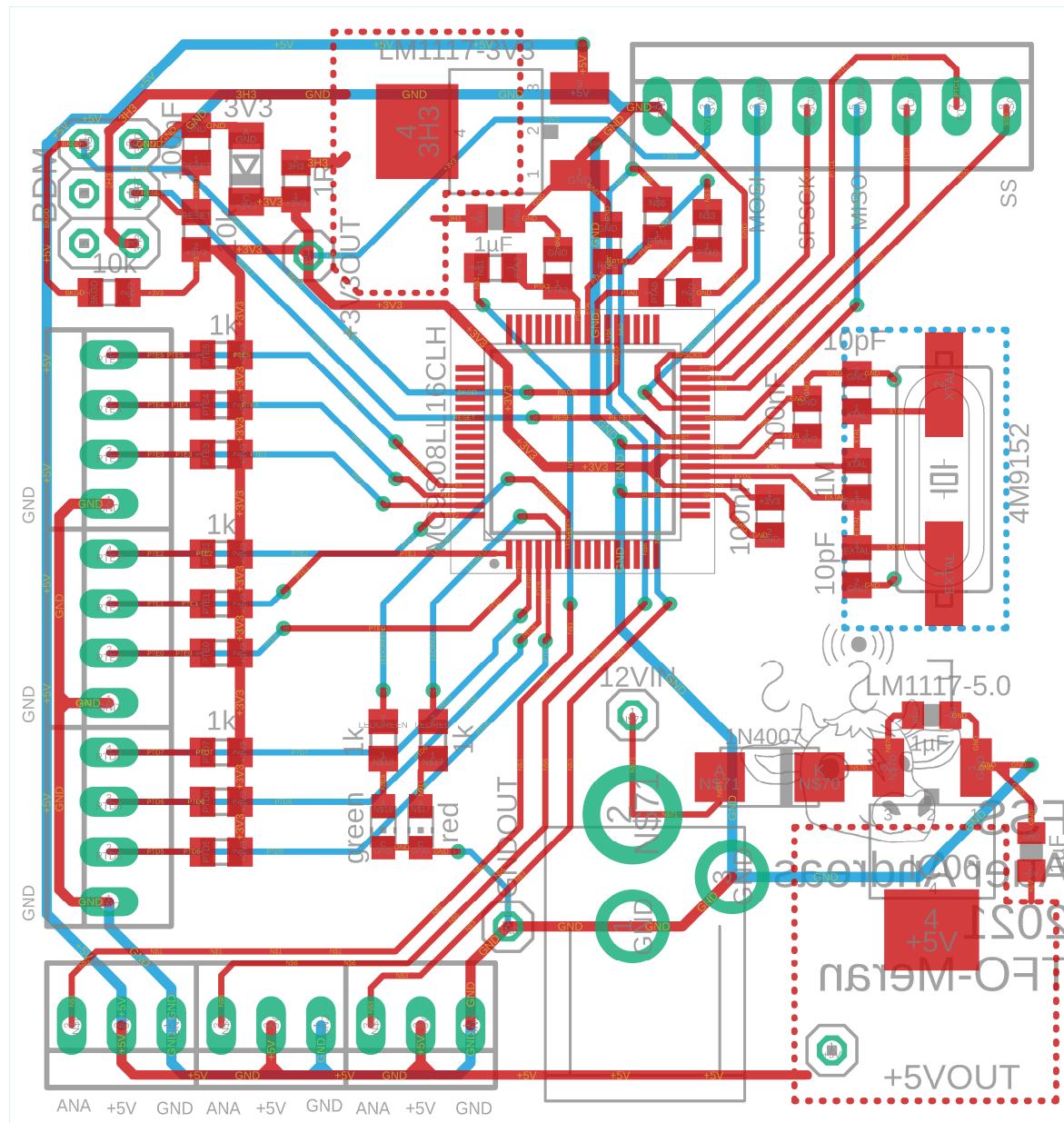


Abbildung 70: MOD-Y V03



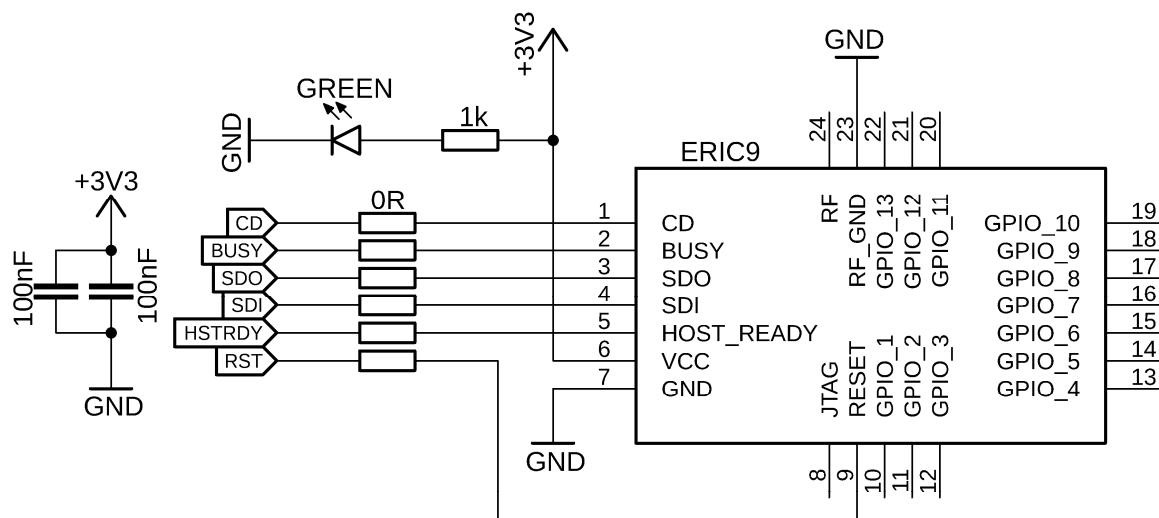
#### 4.6. MOD-Z

MOD-Z ist die letzte Platine und dafür zuständig, das Raspberry-Pi mit den Funkmodulen zu verbinden. Die Platine bietet Platz für beide Funkmodule und enthält alle Leiterbahnen. Um die Platine mit dem Raspberry-Pi zu verbinden, habe ich einen 40-Pol Stecker vorgesehen, wie er auch auf dem Raspberry-Pi zu finden ist. Die Platine und das Raspberry-Pi sind dann mit einem 40-Pol Kabel verbunden worden.

##### 4.6.1. MOD-Z\_V01

Version eins und zwei unterscheiden sich kaum. Leider habe ich zwei Leiterbahnen nicht gleich benannt, weshalb die beiden Leiterbahnen auf dem PCB nicht verbunden waren. Bei Version zwei sind diese Fehler behoben. Generell verwende ich Version eins für den Prototyp, da mit einem Kabel der Fehler behoben werden kann.

Auf dieser Platine hat die Ansteuerung des ERIC9 Priorität. Diese erfolgt über eine UART-Schnittstelle. Zur Sicherheit wird mit einer LED angezeigt, ob das Funkmodul mit Spannung versorgt ist.



© Ansteuerung ERIC9 Auer Andreas  
Abbildung 71: Ansteuerung ERIC9



## 4.6.2. MOD-Z\_V02

Zwischen Version eins und zwei sind eigentlich so gut wie keine Unterschiede. Prinzipiell habe ich nur ein Paar Leiterbahnen neu gezogen und richtig verbunden.

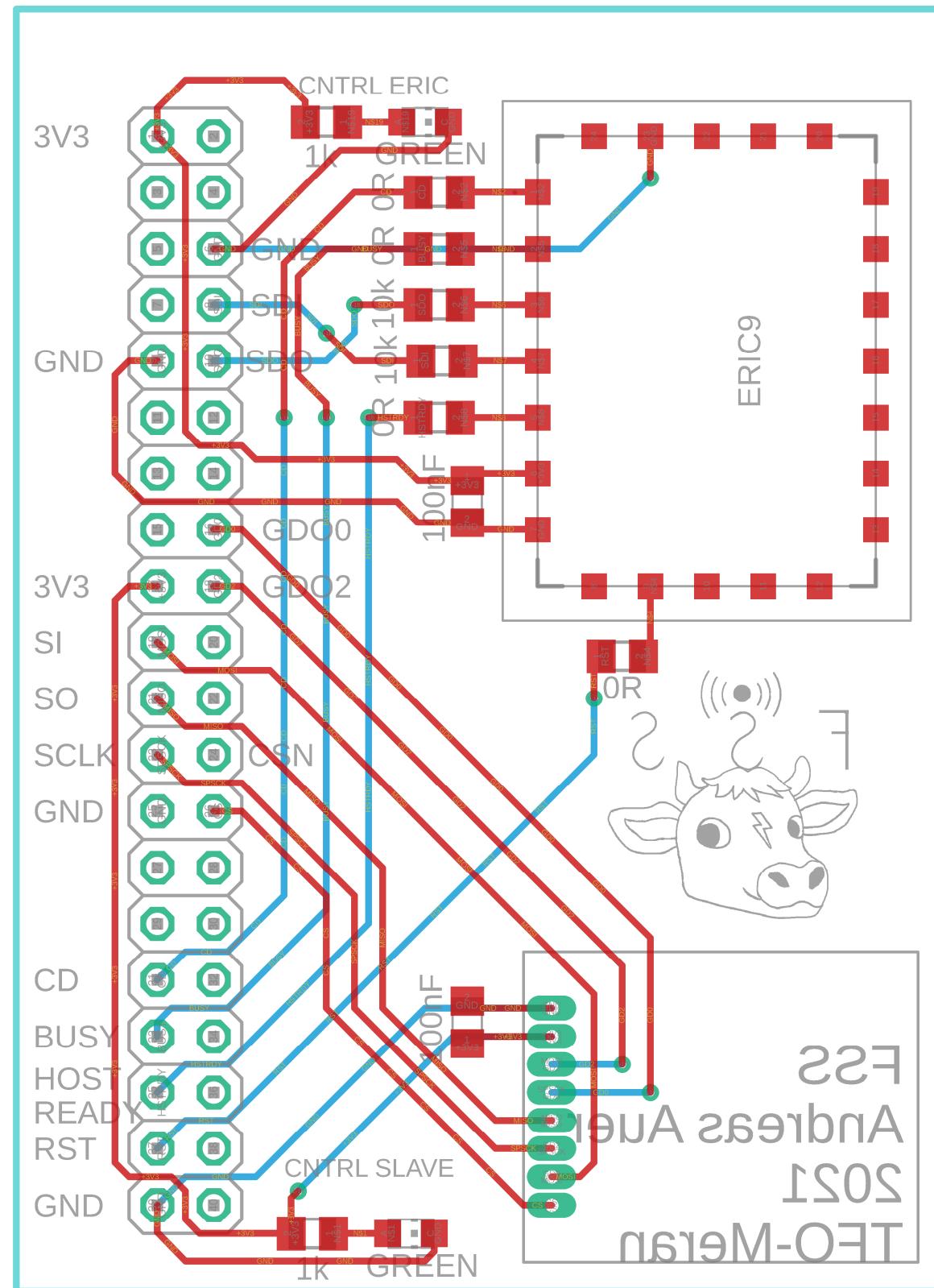


Abbildung 72: MOD-Z V02

## 5. Software

### 5.1. Überblick

Bei Erstellen der Software bin ich in drei Schritten vorgegangen. Zuerst habe ich mit Hilfe von Testprogrammen die Sensoren mit einem Arduino getestet, danach habe ich die Testsoftware in eine für den Microcontroller kompatible Software umgeschrieben und am Ende habe ich die Software der einzelnen Sensoren kombiniert und den Gesamtcode für jede Platine erstellt.

### 5.2. Testprogramme

Am Anfang wusste ich nicht wie jeder Sensor im Einzelnen funktioniert und arbeitet, auch wusste ich nicht, wie ich die Sensoren am besten ansteuern und welche Daten ich senden und empfangen kann. Deshalb habe ich mich dazu entschieden, Testprogramme zu verwenden. Für den Arduino existieren Testprogramme für fast alle nur erdenkbaren Sensoren. Man kann diese dann zum Beispiel von GIT-HUB herunterladen und dann verwenden.

### 5.3. Software-Sensoren

Nachdem ich die einzelnen Sensoren erfolgreich angesteuert und getestet hatte, habe ich damit begonnen, die Software umzuschreiben und an den Microcontroller anzupassen. Die Testprogramme, die ich verwendet habe, waren fast ausschließlich für den Arduino ausprogrammiert. Das bedeutet, dass der Code in einem C ähnlichen Style geschrieben ist. Auch der Microcontroller verwendet eine Version von C, weshalb es mir recht leichtfiel, die Testprogramme an den Microcontroller anzupassen. Jedoch bin ich immer wieder auf Hindernisse gestoßen, welche meine Arbeit gebremst haben.

#### 5.3.1. Feuchtigkeitssensor BME680

Die Software des BME680 ist zweigeteilt. Sie besteht aus der Grundsoftware, in der zum Beispiel die Register gesetzt und die rohen Daten vom Modul gelesen werden und der Software für die Schnittstelle. Die Software für die Schnittstelle ist eine einfache SPI-Ansteuerung. Dieselbe Software wird für das Funkmodul CC2500 (Abschnitt 5.3.3) verwendet. Wichtig sind dabei immer die Initialisierungen.

```
void SPI_Init(void)
{
    SPIC1 = 0b01010100;      // }
    SPIC1 = 0b01010000;      // > setting the SPI to master and other settings
    SPIC2 = 0x00;            // }
    SPIBR = 0b00000000;      //   setting the speed of the SPI (here busfrequency / 2)

    SPI_SDDD_CC2500 = 1;    // setting the pins of the SPI as input and output|
    SPI_SSD_CC2500 = 1;

    SPI_SSDD_BME680 = 1;
    SPI_SSD_BME680 = 1;

    SPI_SPSCK = 1;
    SPI_MOSI = 1;
    SPI_MISO = 0;
    PTBDS = 1;
}
```

Abbildung 73: Initialisierung BME680

Der wichtigste Teil der Software ist das Auslesen der Sensorwerte. Dabei lese ich die ganzen Register des BME680 aus, in denen diese Daten stehen. Bevor die Daten ausgelesen werden können, warte ich aber immer auf das Abschließen der letzten Messung.

```

void BME680_readValues(void)
{
    SS_type = 1;
    BME680_waitForReadings();
    for(i = 0; i < 15; i++)
    {
        BME680_values[i] = SPI_readReg(BME680_STATUS_REGISTER + i);
        Nop();
    }
}

```

Abbildung 74: Auslesen BME680

### 5.3.2. Stromsensor SEN0211

Der Stromsensor gibt ein analoges Signal von 0,2 Volt bis 2,8 Volt aus. Diese Spannung führt zu einem analogen Eingang, wo es dann von der Software eingelesen wird. Der eingelesene Wert wird anschließend an dem Raspberry-Pi gesendet, der den Wert in einen Strom umrechnet.

```

*****
**      WaterSensor_ADC: updates the Variable with the current ADC Value
**
*****
void ADC_LL16_V01_getADCValue(void)
{
    uint temp;

    temp = ADCRH << 8;
    temp += ADCRL;

    switch(ADCPort)
    {
        case 0:
        {
            CurrentSensor1 = temp;
            ADCSC1 = 0b00000001;
            break;
        }

        case 1:
        {
            CurrentSensor2 = temp;
            ADCSC1 = 0b00000010;
            break;
        }

        case 2:
        {
            CurrentSensor3 = temp;
            ADCSC1 = 0b00000000;
            break;
        }

        default:
        {
            break;
        }
    }

    ADCPort = ADCPort++;
    if(ADCPort >= 4)
        ADCPort = 0;
}

```

Abbildung 75: Auslesen des Wasserdrucksensors

### 5.3.3. Funkmodule

Für das Funkmodul CC2500 habe ich zwei verschiedene Softwares geschrieben. Einmal für den Raspberry-Pi und einmal für den Microcontroller, wobei nur das Programm für den Microcontroller interessant ist, da beide Programme identisch sind, bis auf den Unterschied, dass eines in Python und eines in C geschrieben ist. Die Ansteuerung des Funkmoduls ist aber dennoch recht primitiv. Sobald die Verbindung via SPI-Schnittstelle funktioniert, muss man nur noch die Register der Initialisierung schreiben und dann noch einige „Commands“ senden und man ist bereit, Pakete über das Funkmodul zu versenden.

```
//sends a Packet stored in sendPacketArr over the cc2500. lenght of the packet is stored in sendPacketArr[0]
void CC2500_sendPacket(void)
{
    //first byte to be send = length of the package
    //SPI_writeReg(REG_IOCFG1, 0x06);
    // Make sure that the radio is in IDLE state before flushing the FIFO
    SPI_sendStrobe(CC2500_IDLE);
    // Flush TX FIFO
    SPI_sendStrobe(CC2500_FTX);

    // SIDLE: exit RX/TX
    SPI_sendStrobe(CC2500_IDLE);

    // send packet to CC2500
    for(i = 0; i < CC2500_sendPacketArr[0]; i++)
    {
        SPI_writeReg(CC2500_TXFIFO, CC2500_sendPacketArr[i]);
    }

    Nop();
    Nop();

    // STX: exable TX
    SPI_sendStrobe(CC2500_TX);

    //TODO idk why this doesnt work but it works without it too
    // Wait for GDO0 to be set -> sync transmitted
    //while(CC2500_GDO0 == 0 && MISO_PIN == 0){ }

    // Wait for GDO0 ot be cleared -> end of Packet
    //while(CC2500_GDO0 == 1 && MISO_PIN == 1){ }

    //SPI_sendStrobe(CC2500_IDLE);

    Nop();
    Nop();
}
```

Abbildung 76: Senden eines Datenpakets mit dem CC2500

Das Programm für den ERIC9 habe ich hingegen nur in Python geschrieben. Ich habe dafür eine Klasse erstellt, die bei der Deklaration der Instanz die UART-Schnittstelle aktiviert und für den ERIC9 bereithält. Danach kann man Daten einfach über die Schnittstelle versenden und auf der anderen Seite empfangen. Die Schnittstelle kann man mit dem Befehl „serial.write(byte)“ ansteuern, ist also nicht sehr kompliziert.

### 5.3.4. Relais

Die Software für die Relais beruht auf einem einfachen Überprüfen, ob der Abgreippunkt zwischen Relais und Pull-Up-Widerstand auf Masse oder + 3,3 Volt liegt. Je nachdem wird dann eine Eins oder eine Null in eine 2 Byte-Variable geschrieben. Diese Variable wird dann an den Raspberry-Pi gesendet.

### 5.3.5. AD7415

Für den AD7415 war die Hauptherausforderung, den Code für die I2C-Schnittstelle zu schreiben. Sobald I2C funktioniert hat, war es nur noch wichtig, den Sensor zu initialisieren und in einem weiteren Schritt die gemessene Temperatur jede Minute einmal auszulesen. Nach dem Auslesen wird die Temperatur in eine Variable gespeichert und an den Raspberry-Pi gesendet. Dieser wertet die Temperatur aus und kontrolliert ihre Validität.

```

/****************************************************************************
 * Init_AD7415: Macht nicht viel, sendet nur eine Standard-Konfiguration
 * zum AD7415 (die eigentlich die Reset-Kondition sein sollte, aber sicher
 * ist sicher) und setzt dann den Adress-Pointer auf das Register "0x00",
 * dem Temperatur-Register, um dann später einfach nur die Temperaturwerte
 * auslesen zu können.
 */

void Init_AD7415_1 (void)
{
    Byte Reply;

    SDA_PORT_1 = 0;           // SDA und SCL sind EINGÄNGE, also HIGH
    SCL_PORT_1 = 0;
    AirDelay_1 = 0;

    IIC_Start_1();
    IIC_Send_Byte_1 (WRITE_ADDRESS_AD7415_1);      // Adresse (1001 001 0) (Write)
    Reply= IIC_Get_Ack_1();
    IIC_Send_Byte_1 (0x01);   // Configuration Register
    Reply= IIC_Get_Ack_1();
    IIC_Send_Byte_1 (0x40);  // 0100 0000 Filter auf 1, der Rest ist 00, gibt in Hex 0x40
    Reply= IIC_Get_Ack_1();
    IIC_Stop_1();

    IIC_Start_1();          // Einfach nur Adresse "0x00" --> Adresspointer auf Temperatur-Register setzen
    IIC_Send_Byte_1 (WRITE_ADDRESS_AD7415_1);      // Adresse (1001 001 0) (Write)
    Reply= IIC_Get_Ack_1();
    IIC_Send_Byte_1 (0x00);   // Adress-Pointer auf 0x00
    Reply= IIC_Get_Ack_1();
    IIC_Stop_1();

    IIC_Start_1();
    IIC_Send_Byte_1 (WRITE_ADDRESS_AD7415_2);      // Adresse (1001 001 0) (Write)
    Reply= IIC_Get_Ack_1();
    IIC_Send_Byte_1 (0x01);   // Configuration Register
    Reply= IIC_Get_Ack_1();
    IIC_Send_Byte_1 (0x40);  // 0100 0000 Filter auf 1, der Rest ist 00, gibt in Hex 0x40
    Reply= IIC_Get_Ack_1();
    IIC_Stop_1();

    IIC_Start_1();          // Einfach nur Adresse "0x00" --> Adresspointer auf Temperatur-Register setzen
    IIC_Send_Byte_1 (WRITE_ADDRESS_AD7415_2);      // Adresse (1001 001 0) (Write)
    Reply= IIC_Get_Ack_1();
    IIC_Send_Byte_1 (0x00);   // Adress-Pointer auf 0x00
    Reply= IIC_Get_Ack_1();
    IIC_Stop_1();
}

```

Abbildung 77: Initialisierung des AD7415

### 5.3.6. Türsensor

Die Software für den Türsensor ähnelt der der Relais sehr, jedenfalls bei der Platine MOD-X. Dort wird einfach gemessen, ob ein HIGH oder LOW Signal anliegt und dementsprechend dann dem Raspberry-Pi mitgeteilt, ob die Tür offen oder geschlossen ist.

Bei MOD-W hingegen ist es anders. Dort wird der Microcontroller nur auf zwei Weisen aktiviert. Einmal durch das Öffnen der Tür und einmal durch den Uhrenbaustein. Sobald der Microcontroller aktiviert ist, hält er den Enable-Pin des 3,3 Volt Reglers auf HIGH, bis er die Main-Loop einmal abgearbeitet hat, dann schaltet er sich selbst ab. Falls der Türsensor oder der Uhrenbaustein den Microcontroller eingeschaltet hält, beginnt die Loop von vorne. Sobald beide nicht mehr aktiv sind, schaltet sich der Microcontroller einfach aus. Zu Beginn des ersten Durchgangs wird der Timer des Uhrenbausteins zurückgesetzt.

```

void main(void) {
    EnableInterrupts; /* enable interrupts */
    /* include your code here */

    for(;;) {
        while(RealTimeFlag == 0)
        {
            __RESET_WATCHDOG(); /* feeds the dog */
        }
        RealTimeFlag = 0;
        keepAwake();
        Blinklicht();
        checkTurnOn();
        if(firstRound)
        {
            firstRound = 0;
            if(DoorOpen)
            {
                DoorSensor_PreparesPacket();
                CC2500_sendPacketArr[2] = 0xF;
                CC2500_sendPacket();
            }
            else if(timerRanOut)
            {
                CC2500_sendPacketArr[0] = 3;
                CC2500_sendPacketArr[1] = PCB << 4;
                CC2500_sendPacketArr[1] += SENSOR_DOOR;
                CC2500_sendPacketArr[2] = 0xFF;
                CC2500_sendPacket();
            }
            //PCF8563_setAlarm();
        }
        if(delayDoorSend++ >= 420 && DoorOpen) //if the door is opened it sends the current door status every minute
        {
            DoorSensor_PreparesPacket();
            CC2500_sendPacket();
            //PCF8563_setAlarm();
        }

        turnOff();
    } /* loop forever */
    /* please make sure that you never leave main */
}

```

Abbildung 78: Hauptcode MOD-W

### 5.3.7. Wasserdrucksensor Sensata 116CP31-M10S3-50

Die Software für den Wasserdrucksensor wiederum ähnelt der des Stromsensors sehr. Der Wasserdrucksensor gibt ein analoges Signal aus, das vom Microcontroller eingelesen und in Zahlen ausgedrückt wird. Die gelesenen Daten werden an den Raspberry-Pi gesendet, der dann die Berechnung des Drucks ausführt.

## 5.4. Software-Platinen

Am Ende, also sobald ich alle Grundprogramme zu meiner Zufriedenheit fertiggestellt und getestet hatte, habe ich die einzelnen Programme verknüpft und den übergeordneten Code geschrieben, welcher die Daten der einzelnen Sensoren verknüpft, auswertet und an den Raspberry-Pi sendet. Jede Platine verfügt über eine Kopie des Codes für die SPI-Schnittstelle und das Funkmodul. Die restliche Software unterscheidet sich von Platine zu Platine, fasst aber immer nur die Sensoren zusammen.

## 5.5. Basissoftware Microcontroller

Die meisten Probleme bei der Programmierung des Microcontrollers hatte ich am Anfang. Alle Programme, die ich bisher geschrieben habe, waren ausschließlich für den DZ60 und in Assembler geschrieben. Für meine Anwendungen war Assembler aber zu primitiv und ich habe mich für C entschieden, da ich sehr viel mit Arrays (Ketten aus Bytes) gearbeitet habe. Zusätzlich hatte ich zuerst keine Ahnung wie ich beginnen sollte. Ich hatte einen komplett neuen Microcontroller, den ich noch nie verwendet hatte und keine Ahnung wie ich die Grundsoftware schreiben sollte. Zum Glück hatte ich eine alte Software für den AW32 die in C geschrieben war. Mit deren Hilfe und der Grundsoftware für den DZ60 in Assembler war es mir möglich die Grundsoftware für den LL16 zu schreiben.

Das letzte Problem, das ich beseitigen musste, war die Tatsache, dass mein Microcontroller keine interne Hardware hatte, die für eine „real-time“ Programmierung vorgesehen war. Die „real-time“ ist der Grundaufbau des Microcontroller-Codes in Kombination mit der „Main-Loop“. Die „real-time“ ist ein Code, der exakt alle 10 Millisekunden ein „Flag“ gibt. Mit dem „Flag“ beginnt ein Durchgang in der „Main loop“, bei dem alle Unterprogramme einmal aufgerufen werden. Ich musste mir dann eine eigene „real-time“ mithilfe eines „Timer/Pulse-Width-Modulators“ erstellen.

```
#pragma TRAP_PROC
interrupt void _Realtime(void)
{
    uchar temp = TPM1C0SC;      // read register to continue timer
    uint compValue = TPM1COV;   // load the current time Value

    __asm bclr    7, TPM1C0SC // continue counting time

    compValue += 0x2F;          // set new time instance to 3.33 ms in the future
    TPM1COV = compValue;       // save new time instance

    RealTimeCounter++;
    if(RealTimeCounter == 3)
    {
        RealTimeFlag = 1;      // set real time flag
        RealTimeCounter = 0;   // one main loop round will be performed
    }
}

void InitRealtime(void)
{
    TPM1SC = 0b00001000;        //}
    TPM1C0SC = 0b01010000;     //> Registerdefinition
    TPM1COV = 0x2F;            //}

    RealTimeFlag = 0;           // set start values to 0
    RealTimeCounter = 0;
}
```

Abbildung 79: Realtime Microcontroller

```
void main(void) {

    EnableInterrupts; /* enable interrupts */
    /* include your code here */

    for(;;) {
        while(RealTimeFlag == 0)
        {
            __RESET_WATCHDOG(); /* feeds the dog */
        }
        RealTimeFlag = 0;

        //here my tasks start
        Blinklicht();

    } /* loop forever */
    /* please make sure that you never leave main */
}
```

Abbildung 80: Main loop Microcontroller

Der dritte wichtige Punkt des Basiscodes ist der „Startup-Code“. In ihm werden alle wichtigen Register für den Microcontroller, dessen Busfrequenz und die Initialisierung der Programme ausgeführt. Auch werden alle Eingänge als Eingänge deklariert.

```
PTADD = 0x00;
PTBDD = 0x00;
PTCDD = 0x00;
PTDDD = 0x00;
PTEDD = 0x00;

PTAD = 0;
PTBD = 0;
PTCD = 0;
PTDD = 0;
PTED = 0;

InitMCU();
InitRealtime();
InitBlinklicht();

EnableInterrupts; /* enable interrupts */
```

Abbildung 81: Initialisierung Microcontroller

Was mir zuerst nicht bewusst war, ist die Tatsache, dass der Microcontroller MC9S08LL16 für die Ansteuerung von „Liquid Crystal Displays“ existiert. Deshalb muss man die Eingänge von „open drain“ (kann nur „Floating“ oder Masse sein) auf „full complementary drive“ umgeschaltet werden. Dafür muss man die richtigen Register auf die richtigen Werte setzen.

```
void InitBlinklicht(void){
    RUNLEDPORTGREEN=1; // Das Pin des RunLed wird als Ausgang deklariert (PD1)
    RunLedCounter=10;

    LCDSUPPLY = 0b00110111; // Microcontroller Pins set to full complementary
    LCDC1 = 0b00000100;
}
```

Abbildung 82: Initialisierung der Ports



## 6. Gehäuse

Für die drei Hauptplatinen ist für die Zukunft ein Gehäuse vorgesehen, welches die Platine, das Funkmodul und die angebauten Sensoren beinhaltet. Bis dato habe ich aber noch kein Gehäuse entwickelt, werde das aber nach dem Schulabschluss so bald als möglich nachholen.

## 7. Projektmanagement:

Damit man während der Ausarbeitung des Projektes nicht den Überblick verliert, verwendet man ein sogenanntes Gantt Diagramm. In diesem Diagramm wird im Voraus jedes Workingpackage eingetragen und zusätzlich noch angegeben, in welchem Zeitraum dieses erarbeitet werden soll. Auch Pausen werden eingeplant. Falls man irgendwo im Rückstand ist, wird dies ROT eingetragen. Falls man zeitlich in Ordnung ist oder einen Teil abgeschlossen hat, dann ist dieser GRÜN markiert.

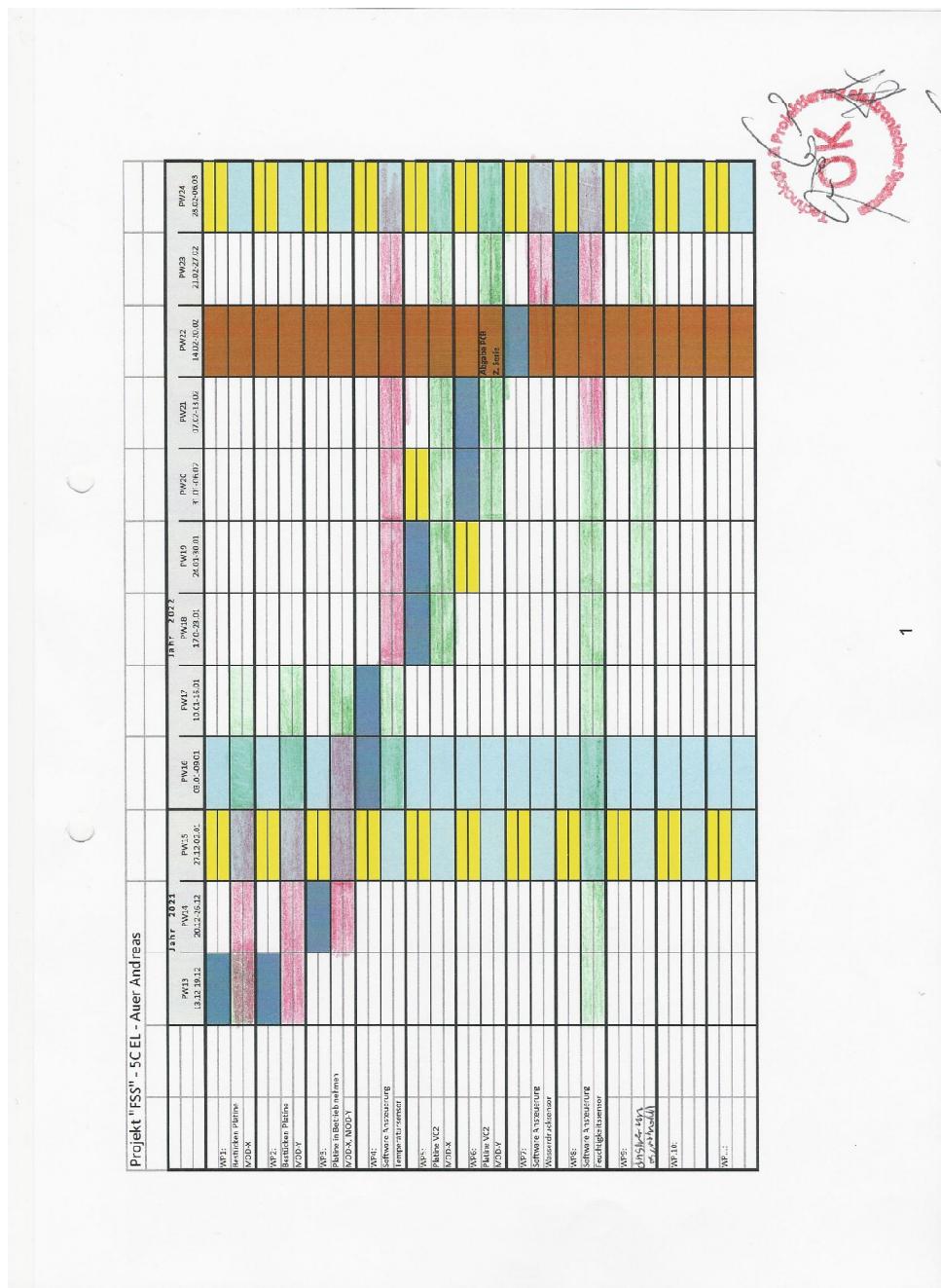


Abbildung 83: Beispiel GANTT-Diagramm

## 8. Kostenrechnung Prototyp

Die Kostenberechnung des Prototyps ist sehr teuer. Dabei müssen alle Arbeitsstunden mit einberechnet werden, alle Bauteile, die man bei der Ausarbeitung verwendet hat, sowie die Beraterstunden. Die Stunden sind real, die Preise jedoch nur angenommen und entsprechen nicht der Realität. Die Kosten der Bauteile sind vollkommen richtig.

	Anzahl	\$/Anzahl	Preis
Kosten Bauteile	1 st.	€ 1,021.65	€ 1,021.65
Kosten Bauteile Schule	1 st.	€ 10.00	€ 10.00
Kosten Beraterstunde	12 st.	€ 100.00	€ 1,200.00
Arbeitsstunden	419 st.	€ 40.00	€ 16,760.00
Gesamt			€ 18,991.65

Abbildung 84: Kostenrechnung Prototyp

Alles in allem hätte der Prototyp einen Arbeitgeber um die 19.000 Euro gekostet. Dabei entstehen die meisten Kosten durch die große Anzahl der Arbeitsstunden. Die Bauteile sowie die Beraterstunden nehmen einen kleinen Teil ein. Beraterstunden sind teuer, hätte man jedoch keinen Berater zur Hilfe gerufen, hätte man viele Stunden mit einem Problem verbracht, was im Endeffekt teurer ist, als einen Berater zu fragen, der die Lösung sofort weiß.

## 9. Kostenrechnung Serientyp

Beim Serientyp ist es wichtig zu wissen, dass bei einer Massenproduktion die Preise stark gesenkt werden. In meinem Fall ist es schwer eine Berechnung für einen Serientyp zu erstellen. Zum einen ist es variabel, welche Platinen angebaut werden und welche nicht, zum anderen muss das Menü für jeden potenziellen Kunden neu erstellt werden, da sich darauf eine maßgetreue Abbildung der Grundstruktur des Hofs befindet. Aufgrund des Arbeitsaufwandes und der Preise der Bauteile nehme ich an, dass ein vollständiger Serientyp einen Preis zwischen 1.000 Euro und 2.000 Euro annehmen wird, je nachdem wie lange die Änderungen des Menus beträgt und wie viele Platinen eingebaut werden sollen.

## 10. Anhang

### 10.1. Fazit

Alles in allem habe ich bei der Ausarbeitung dieses Projektes sehr viel dazugelernt. Sei es beim Thema Platinendesign, beim Programmieren oder einfach beim Organisieren und Planen. Ich habe mir sehr viele wertvolle Errungenschaften angeeignet, welche mir beim Studium und in der Arbeitswelt sicher weiterhelfen werden. Mit dem Schreiben dieser Dokumentation wird mir immer mehr bewusst, um welch großes Projekt es sich bei „FSS“ handelt. Ich habe sehr viele verschiedene Sensoren verwendet, mehrere Platinen entwickelt, mich mit verschiedenen Programmiersprachen befasst und das Projekt so gut wie möglich gestemmt. Am Ende kann ich mit gutem Gewissen behaupten, dass ich stolz auf mich selbst bin und auch in Zukunft an diesem und anderen Projekten weiterarbeiten werde.

## 10.2. Abbildungsverzeichnis

Abbildung 1: Projektantrag Seite1.....	7
Abbildung 2: Projektantrag Seite 2 .....	8
Abbildung 3: Projektantrag Seite 3 .....	9
Abbildung 4: Pflichtenheft Seite 1 .....	10
Abbildung 5: Pflichtenheft Seite 2 .....	11
Abbildung 6: Beispiel Wochenbericht .....	13
Abbildung 7: Ansicht BME680 .....	15
Abbildung 8: Berechnung des Drucks BME680.....	15
Abbildung 9: Elektrical characteristics BME680 .....	15
Abbildung 10: Board overview SEN0211 .....	16
Abbildung 11: Ansicht Stromzange SEN0211 .....	16
Abbildung 12: Benutzung Stromsensor .....	16
Abbildung 13: Ansicht CC2500.....	17
Abbildung 14: Generelle Infomationen CC2500.....	17
Abbildung 15: Electrical characteristics CC2500.....	17
Abbildung 16: Ansicht ERIC9 .....	18
Abbildung 17: Frequenzen ERIC9.....	18
Abbildung 18: Aufbau ERIC9 .....	18
Abbildung 19: Electrical characteristics ERIC9.....	18
Abbildung 20: Ansicht Relais.....	19
Abbildung 21: Informationen Spule Relaise .....	19
Abbildung 22: Ansicht AD7415 .....	20
Abbildung 23: Packages AD7415 .....	20
Abbildung 24: Electrical characteristics AD7415 .....	20
Abbildung 25: Ansicht Türsensor .....	21
Abbildung 26: Electrical characteristics Türsensor .....	21
Abbildung 27: Ersatzschaltung Türsensor.....	21
Abbildung 28: Ansicht Wasserdrucksensor .....	22
Abbildung 29: Electrical characteristics Wasserdrucksensor .....	22
Abbildung 30: Ansicht Microcontroller.....	23
Abbildung 31: Ausstattung Microcontroller .....	23
Abbildung 32: Blockdiagramm Microcontroller .....	24
Abbildung 33: Pinbelegung Microcontroller.....	25
Abbildung 34: Electrical characteristics Microcontroller.....	25
Abbildung 35: Ansicht Raspberry-Pi .....	26
Abbildung 36: Raspberry-Pi mit Display .....	26
Abbildung 37: Raspberry-Pi mit Case und Funkmodul .....	26
Abbildung 38: Ansicht LM324 .....	27
Abbildung 39: Electrical characteristics LM324 .....	27
Abbildung 40: Ersatzschaltbild LM324.....	27
Abbildung 41: Ansicht LM1117 .....	28
Abbildung 42: Electrical Characteristics LM1117 .....	28
Abbildung 43: Dropout voltage LM1117 .....	28
Abbildung 44: Ansicht NCP114 .....	29
Abbildung 45: Ersatzschaltbild NCP114 .....	29
Abbildung 46: Electrical characteristics NCP114 .....	29

Abbildung 47: Testschaltung zum Laden der Batterien.....	30
Abbildung 48: Ansicht Uhrenbaustein .....	31
Abbildung 49: Anwendung und Funktion des Uhrenbausteins.....	31
Abbildung 50: Blockschaltbild Uhrenbaustein.....	31
Abbildung 51: Programmierschaltung .....	32
Abbildung 52: Quarzbeschaltung.....	32
Abbildung 53: MOD-V PCB.....	33
Abbildung 54: Spannungsregelung 1,5V to 3,3V .....	34
Abbildung 55: Uhrenbaustein und Kontrollschaltung V01 .....	35
Abbildung 56: Uhrenbaustein und Kontrollschaltung V02 .....	36
Abbildung 57: Spannungsregelung 12V - 3,9V - 3,3V .....	36
Abbildung 58: Uhrenbaustein und Kontrollschaltung V03 .....	37
Abbildung 59: Türsensor .....	37
Abbildung 60: Spannungsversorgung 12V - 5V - 3,3V .....	38
Abbildung 61: Pegelanpassung 4,5V zu 3,3V .....	38
Abbildung 62: Ansteuerung AD7415 .....	39
Abbildung 63: Ansteuerung Türsensor .....	39
Abbildung 64: Ansteuerung AD7415 .....	39
Abbildung 65: Beschaltung Relais.....	40
Abbildung 66: Ansteuerung Stromsensor.....	41
Abbildung 67: MOD-Y V02 heiße Seite .....	41
Abbildung 68: MOD-Y V02 kalte Seite .....	41
Abbildung 69: MOD-Y V01 .....	41
Abbildung 70: MOD-Y V03 .....	42
Abbildung 71: Ansteuerung ERIC9 .....	43
Abbildung 72: MOD-Z V02 .....	44
Abbildung 73: Inizialisierung BME680 .....	45
Abbildung 74: Auslesen BME680 .....	46
Abbildung 75: Auslesen des Wasserdrucksensors.....	46
Abbildung 76: Senden eines Datenpakets mit dem CC2500 .....	47
Abbildung 77: Inizialisierung des AD7415 .....	48
Abbildung 78: Hauptcode MOD-W .....	49
Abbildung 79: Realtime Microcontroller .....	50
Abbildung 80: Mainloop Microcontroller .....	50
Abbildung 81: Inizialisierung Microcontroller .....	51
Abbildung 82: Inizialisierung der Ports .....	51
Abbildung 83: Beispiel GANTT-Diagramm .....	52
Abbildung 84: Kostenrechnung Prototyp .....	53

### 10.3. Quellen

Alle Informationen, die in diesem Dokument niedergeschrieben sind, sind entweder von mir selbst erfasst und gesammelt oder aus den Datenblättern der Bauteile gefiltert worden.

### 10.4. Bildquellen

BME680:

<https://www.sensatio.io/wp-content/uploads/2020/05/BME680.jpg>

Datenblatt des BME680

SEN0211:

[https://m.media-amazon.com/images/I/51QS-gxzRCL.\\_SX522\\_.jpg](https://m.media-amazon.com/images/I/51QS-gxzRCL._SX522_.jpg)

Datenblatt des SEN0211

CC2500:

<https://i.ebayimg.com/images/g/A5gAAOSwX-pZufFF/s-l400.jpg>

Datenblatt des CC2500

ERIC9:

Datenblatt des ERIC9

RT14730:

[https://it.farnell.com/productimages/standard/en\\_GB/2060767-40.jpg](https://it.farnell.com/productimages/standard/en_GB/2060767-40.jpg)

Datenblatt des Relais

Türsensor:

Datenblatt des Türsensors

Wasserdrucksensor:

[https://www.mouser.it/images/sensatechnologies/lrg/116CP21-series\\_SPL.jpg](https://www.mouser.it/images/sensatechnologies/lrg/116CP21-series_SPL.jpg)

Datenblatt des Wasserdrucksensors

Microcontroller:

<https://media.digikey.com/Renders/Freescale%20Renders/64-LQFP.jpg>

Datenblatt des Microcontrollers

LM324:

<http://electrobist.com/wp->

<content/uploads/2018/08/9c625ea59894d96b93ecef064f21a7c3.jpg>

[https://hallroad.org/images/detailed/31/LM324\\_SMD\\_General\\_Purpose\\_Op\\_Amp\\_IC\\_In\\_Lahore\\_Islamabad\\_Karachi\\_Faisalabad\\_Multan\\_Quetta\\_Pakistan\\_1.jpg](https://hallroad.org/images/detailed/31/LM324_SMD_General_Purpose_Op_Amp_IC_In_Lahore_Islamabad_Karachi_Faisalabad_Multan_Quetta_Pakistan_1.jpg)

Datenblatt des OPVs

Temperatursensor:

<https://sigma.octopart.com/158621876/image/Analog-Devices-AD7415ARTZ-0REEL7.jpg>

Datenblatt des Sensors

Spannungsregler:



[https://elcodis.com/photos/12/65/126582/td03b\\_sml.jpg](https://elcodis.com/photos/12/65/126582/td03b_sml.jpg)

<https://media.digikey.com/Renders/On%20Semi%20Renders/488;517CU;MX;4.jpg>

Datenblatt des Reglers

Raspberry-Pi:

[https://it.farnell.com/productimages/large/en\\_GB/3051885-40.jpg](https://it.farnell.com/productimages/large/en_GB/3051885-40.jpg)

Alle Informationen, die nicht angegeben sind, stammen von mir selbst.

## 10.5. Copyright

Hiermit bestätige ich, Auer Andreas, dass ich das Projekt „FSS“ eigenständig im Schuljahr 2021/22 erarbeitet habe. Im Notfall halfen die beiden Berater Martin De Tomaso und Ivan Huber. Alle Text- und Bildquellen sind im Abschnitt Quellen bzw. Bildquellen verlinkt. Alle anderen Bilder und Textquellen sind eigenständig von mir erstellt oder fotografiert worden.

Der Projektleiter Andreas Auer:

---

Der Tutor und Professor Martin De Tomaso:

---

Der Tutor und Professor Ivan Huber:

---

Der Direktor der TFO-Meran „Oskar von Miller“ Alois Heinrich Weis:

---

Ort und Datum:

---