

# Projekt-Bericht

Isabell Albrecht, Erik Engelhardt, Oliver Kochan, Florian Steffens

Solarbetriebene, mobile Wetterstation

Betreuung durch: Prof. Dr. Franz Schubert  
Eingereicht am: 13. Januar 2019

*Fakultät Technik und Informatik  
Department Informations- und Elektrotechnik*

*Faculty of Computer Science and Engineering  
Department Information and Electrical Engineering*

## **Inhaltsverzeichnis**

<b>1 Einführung</b>	<b>1</b>
1.1 Die Wetterstation . . . . .	1
1.2 Anforderungen . . . . .	2
1.3 Vorüberlegungen . . . . .	2
<b>2 Spannungsversorgung</b>	<b>3</b>
2.1 Anforderungen . . . . .	3
2.2 Power-Board . . . . .	3
2.3 Sensor-Board . . . . .	6
<b>3 Sensoren</b>	<b>7</b>
3.1 Kompassmodul QMC5883L . . . . .	7
3.1.1 Montage . . . . .	7
3.2 NEO-6M GPS-Modul . . . . .	8
3.2.1 Montage . . . . .	9
3.3 Anemometer und Windfahne: SEN-08942 . . . . .	9
3.3.1 Montage . . . . .	11
3.4 Neigungssensor MPU6050 . . . . .	11
3.4.1 Montage . . . . .	12
3.5 BME280 . . . . .	12
3.5.1 Montage . . . . .	13
3.6 Anschlagssensor . . . . .	13
3.6.1 Montage . . . . .	13
<b>4 Mikrocontroller</b>	<b>14</b>
4.1 Auswahl des Mikrocontrollersystems . . . . .	14
4.2 Pin-Belegung . . . . .	15
<b>5 Firmware</b>	<b>17</b>
5.1 Auswertung der Sensoren ( <code>sensorlib</code> ) . . . . .	18
5.1.1 BME280 . . . . .	18
5.1.2 CPU-Temperatursensor . . . . .	20
5.1.3 QMC5883L . . . . .	21
5.1.4 MPU6050 . . . . .	25
5.1.5 Anemometer und Windfahne . . . . .	27

5.1.6	Strom- und Spannungsmessung . . . . .	30
5.2	GPS . . . . .	32
5.2.1	Interpretation der NMEA Sentences . . . . .	32
5.2.2	Versorgung des GPS-Moduls . . . . .	34
5.2.3	GPS-Datenstruktur im Speicher . . . . .	35
5.3	Bluetooth . . . . .	35
5.3.1	AT-Befehlssatz . . . . .	35
5.3.2	Energiesparmaßnahmen . . . . .	43
5.3.3	Pairing-Daten . . . . .	43
5.4	Messdatenprotokoll auf einer SD-Karte . . . . .	43
5.5	Motorsteuerung . . . . .	44
5.5.1	Referenzfahrt für die Panelausrichtung . . . . .	44
5.5.2	Sicherheitsfunktion . . . . .	44
5.6	Statusanzeige über LEDs . . . . .	44
5.6.1	Grüne LED: Systemstatus . . . . .	45
5.6.2	Blaue LED: Bluetooth . . . . .	45
5.6.3	Gelbe LED: Motorsteuerung . . . . .	45
<b>6</b>	<b>Ausrichtung des Solarpanels</b>	<b>46</b>
6.1	Berechnung der Sonnenposition . . . . .	46
<b>7</b>	<b>Benutzeroberfläche</b>	<b>48</b>
7.1	Funktionen . . . . .	48
7.2	Entwicklung . . . . .	53
<b>8</b>	<b>3D gedruckte Komponenten</b>	<b>57</b>
8.1	Hauptgehäuse . . . . .	57
8.2	Nebengehäuse . . . . .	59
8.3	Adapter . . . . .	61
8.4	Herstellung . . . . .	63
<b>9</b>	<b>Fazit</b>	<b>64</b>
<b>Literatur</b>		<b>66</b>
<b>A</b>	<b>Stromlaufplan Power-Board</b>	<b>i</b>
<b>B</b>	<b>Stromlaufplan Sensor-Board</b>	<b>ii</b>

## 1 Einführung

Der folgende Bericht gibt einen Überblick über den Entwurf und die Umsetzung einer solarbetriebenen Wetterstation im Rahmen der Sensorikvorlesung. Hierbei sollen zunächst die Anforderungen und bereits vorhandene Konzepte/Komponenten kurz vorgestellt und dann sowohl auf die Soft- als auch die Hardwareseitige Umsetzung eingegangen werden.

### 1.1 Die Wetterstation



Abbildung 1: Gegebener Aufbau der Wetterstation

In Abbildung 1 ist der gegebene Aufbau der Wetterstation dargestellt. Dieser besteht aus einem kippbaren Solarpanel, das auf einer drehbaren Platte montiert ist. Sowohl Dreh- als auch Kippbewegung wird über jeweils einen Getriebemotor ermöglicht. Des Weiteren ist bereits eine Windfahne zur Erfassung der Windrichtung und -geschwindigkeit montiert.

## 1.2 Anforderungen

Die Wetterstation soll folgende Werte erfassen:

- Temperatur
- Luftdruck
- Luftfeuchte
- Höhe über NN
- Windgeschwindigkeit
- Windrichtung
- Standort

Weiterhin soll die Wetterstation über das Solarpanel mit Strom versorgt werden, wobei aber dieses durch einen 12V Akku gepuffert werden soll, um z.B. die Nachtstunden ohne Ausfall der Spannungsversorgung überbrücken zu können. Für eine optimale Energieausbeute soll das Panel abhängig vom Sonnenstand über die beiden Getriebemotoren ausgerichtet werden. Außerdem soll der Akkuzustand erfasst und die drahtlose Kommunikation mit einem PC ermöglicht werden. Schließlich sollen die erfassten Sensordaten auf einer SD-Karte gespeichert werden.

## 1.3 Vorüberlegungen

Bevor die beschriebenen Anforderungen umgesetzt werden, sind einige Vorüberlegungen und die Auswahl der geeigneten Sensorik nötig. Letztere werden ausführlich in Abschnitt 3 dargestellt. Die Drahtloskommunikation soll über ein Bluetooth-Modul realisiert und um eine eigene GUI, die die Darstellung aktueller Sensordaten auf einem PC ermöglicht, erweitert werden. Des Weiteren wird die bisher vorhandene Windfahne ausgetauscht und ersetzt, da diese nur über eine unzureichende Dokumentation verfügt. Ebenfalls ist eine wasserdichte Ausführung der Wetterstation wünschenswert, die Umsetzbarkeit muss überprüft werden. Um Energie zu sparen, soll die Ausrichtung der Wetterstation sowie die Abfrage der Sensordaten nicht kontinuierlich sondern in noch festzulegenden Zeitabständen erfolgen.

## 2 Spannungsversorgung

In diesem Abschnitt soll auf die hardwareseitige Umsetzung der Spannungsversorgung für die Wetterstation eingegangen werden. Ziel ist der Entwurf einer Platine, auf der sämtliche Anforderungen umgesetzt werden.

### 2.1 Anforderungen

Zunächst sollen in diesem Abschnitt die Anforderungen, die sich aus der Aufgabenstellung ableiten lassen, sowie solche, die sich aus den weiteren Überlegungen zur Umsetzung der Wetterstation ergeben.

- Messung des Ladestroms
- Messung der Batteriespannung (Ladezustand)
- Messung des Stromverbrauchs der Wetterstation

Des Weiteren soll der Stromverbrauch der Wetterstation so niedrig wie möglich sein, um die Puffer-Batterie zu schonen und sonnenarme Phasen bzw. die Nacht ohne Stromausfall überbrücken zu können. Die verwendete Batterie hat eine Ladeschlussspannung von 12 V. Da für den Mikrocontroller und die Sensoren allerdings Spannungspegel von 3,3 V und 5 V benötigt werden, müssen diese auf der Platine erzeugt werden.

Aus den mechanischen Anforderungen, dass Mikrocontroller, Platine und Sensoren möglichst in einem Gehäuse untergebracht werden sollen, ergibt sich, dass die entworfene Platine auf die Pinheader des Mikrocontrollers gesteckt werden soll. Auf Grund des Umfangs werden zwei Platinen Layoutet: Ein Powerboard zur Erzeugung der benötigten Spannungen und ein Sensor-Board, auf dem sich der Schaltungsteil für die Sensoren befindet.

### 2.2 Power-Board

In Abbildung 2 wird das finale Layout des Power-Boards gezeigt. Im folgenden sollen einige wichtige Details des Stromlaufplans dargestellt.

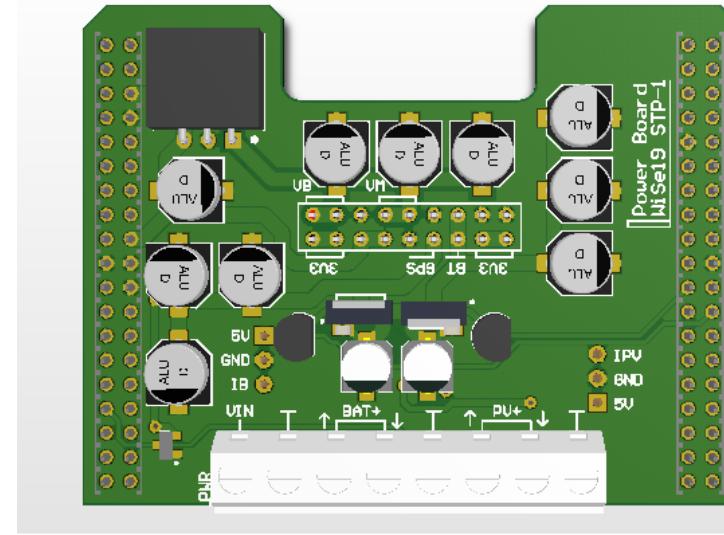


Abbildung 2: Power-Board

Wesentliche Komponenten des Power-Boards sind die Erzeugung der 3V3 und 5V Spannungen. Die 5 V werden von einem Mornsun Festspannungsregler mit einer V\_IN Range von 6,5 bis 30 V. Die 3V3 werden direkt vom Mikrokontroller, der mit 5 V versorgt wird, erzeugt. Über zwei Spannungsteiler wird die Batterie- bzw. Solarpanelspannung gemessen (s. Abb. 3):

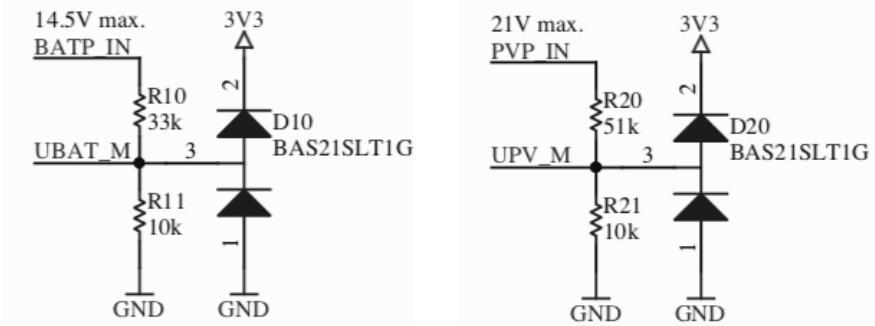


Abbildung 3: Spannungsteiler

Im Zuge der Überlegungen bezüglich möglicher Energiesparmaßnahmen wurden sowohl das Bluetooth- als auch das GPS-Modul als groÙe Verbraucher ermittelt. Da beide Module auch nicht dauerhaft benötigt werden – das GPS-Modul nur alle 15 Minuten zur

## 2 Spannungsversorgung

---

Neuausrichtung des Panels und das Bluetooth-Modul nur nach Bedarf – ist es sinnvoll, die Betriebsspannungen beider Module schaltbar zu machen. Eine Möglichkeit dafür ist die Verwendung eines p-Kanal-Mosfets, der von einem n-Kanal-Mosfet getrieben wird. Diese Schaltung wird in Abbildung 4 dargestellt.

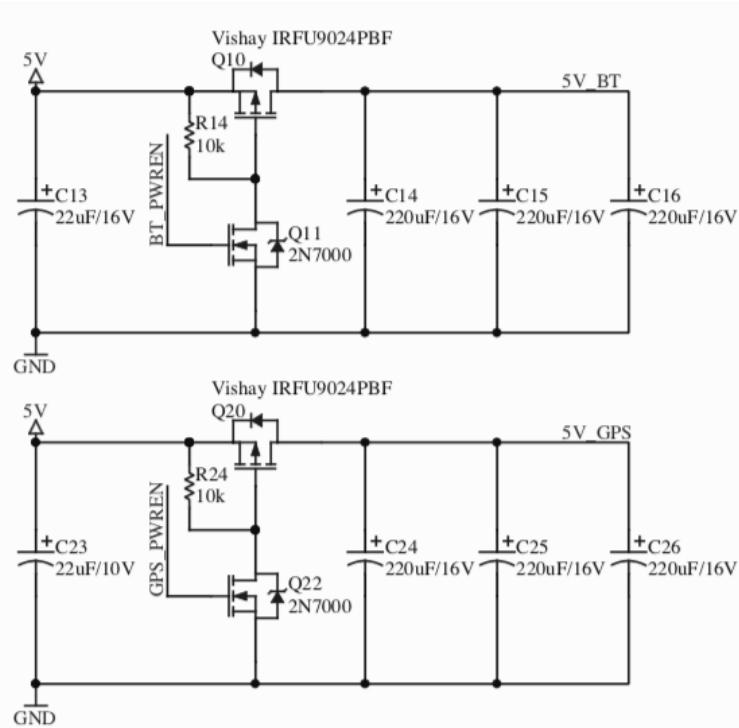


Abbildung 4: Spannungsabschaltung 5VBT und 5VGPS

Die Messung des Ladezustands wird über einen einfachen Spannungsteiler realisiert, der so dimensioniert ist, dass maximal 3,37 V bei vollgeladenem Akku ausgegeben werden. Die genauen Werte sind dem angehängten Stromlaufplan zu entnehmen. Die Ströme werden mittels zweier ACS712 Stromsensoren, die von 5 V versorgt werden, gemessen und vom Mikrokontroller ausgewertet.

Die Pins des Mikrocontrollers werden über das Powerboard zum im folgenden erläuterten Sensor-Board durchgeschliffen.

## 2.3 Sensor-Board

In Abbildung 5 wird das finale Layout des Sensor-Boards gezeigt:

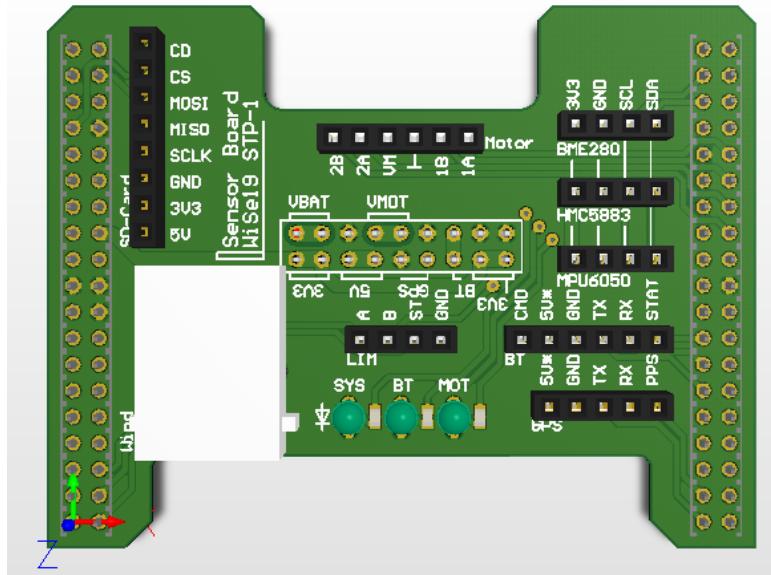


Abbildung 5: Sensor-Board

Das Sensor-Board beschränkt sich im Wesentlichen auf die Verdrahtung von Signal- und Spannungsleitungen der Sensoren mit dem Mikrokontroller und den benötigten Spannungen. Die Sensoren werden dabei über Pinheader mit dem Board konnektiert. Drei Status-LEDs, die im späteren Aufbau im Hauptgehäuse nach außen geführt werden, geben Aufschluss über den Status des Systems, Bluetooth-Verbindung und Motoransteuerung.

## 3 Sensoren

### 3.1 Kompassmodul QMC5883L

Als Kompass dient uns der **QMC5883L**. Das Bauteil ist zwar mit **HMC5883L** beschriftet, verbaut ist aber ein **QMC5883L**. Überprüfen lässt sich dies über ein Auslesen der Registeradresse, welche den Wert *0x0D* statt wie erwartet *0x3D/0x3C* zurückgibt.

Verwendet wird er, um die Ausrichtung der Wetterstation gen Norden zu messen (s. Kap. 6). Diese Information wird benötigt, um das Solarpanel korrekt zur Sonne auszurichten zu können.

Auf die Software zu dem Kompassmodul wird in Kapitel 5.1.3 eingegangen.

#### 3.1.1 Montage

Das Kompassmodul hat 5 Anschlüsse, von denen vier verwendet werden:  $V_{CC}$ , GND, SDA und SCL. Der fünfte Anschluss wird nur bei einer häufigen Datenübertragung benötigt. Er sendet die Information, dass Daten zum Abruf bereit stehen. Dargestellt ist die Verschaltung in Abbildung 6.

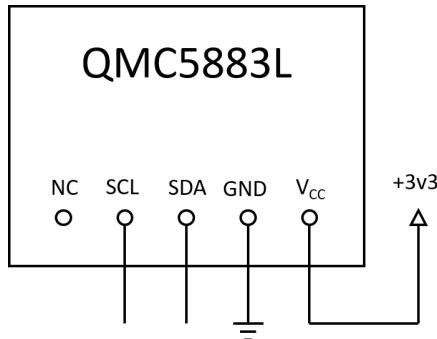


Abbildung 6: Verschaltung des QMC5883L

Um den Sensor fest zu verbauen, aber gleichzeitig ein Austauschen möglich zu machen, wurde eine Lochrasterplatine zugeschnitten und dann mit entsprechenden Steckverbindungen (*male* für das Kabel, *female* für den Sensor) ausgestattet. Anschließend wurde diese in einem Deckel der Nebengehäuse (s. Kap. 8.2) eingeklebt. Als letzten Schritt

musste nur noch ein Kabel mit 4 Adern gefertigt werden, um die Lochrasterplatine mit dem Board zu verbinden.

Da der Sensor das Magnetfeld misst, muss dieser zudem entfernt von der Windfahne und dem Anemometer platziert werden, da beide mit magnetischen Bauteilen arbeiten. Andernfalls könnte es zu Fehlmessungen kommen. Umgesetzt wurde dies so, dass an der einen Stütze des Solarpanels das Kompassmodul und an der anderen die Windfahne, das Anemometer und das gegenüber magnetischen Beeinflussungen relativ unempfindliche GPS-Modul angebracht wurden (s. Abb. 7).



Abbildung 7: Platzierung des Kompassmoduls auf der linken Stütze und Windfahne, Anemometer und GPS-Modul auf der rechten Stütze. An der Rückseite des Solarpanels befindet sich der Neigungssensor.

## 3.2 NEO-6M GPS-Modul

Um den Standort des Geräts zu bestimmen und so die Berechnung der korrekten Auslenkung des Solarpanels auf Grund des Standorts zu ermöglichen, musste ein GPS-Modul genutzt werden. Wir haben uns für den **NEO-6M** von der Firma **u-blox** entschieden. Er ermöglicht eine Bestimmung der aktuellen Position bei einer Abweichung von 2.5 m

[21]. Das Modul wird mittels *USART* angesprochen, die verwendete Software wird in Kapitel 5.2 genauer erklärt.

#### 3.2.1 Montage

Da das GPS-Modul in Einheit mit einer Antenne funktioniert, musste das Modul entfernt von der Elektronik und damit außerhalb des Gehäuses platziert werden. Dazu bot sich an, das GPS-Modul an der zweiten Stütze des Solarpanels zu platzieren. Das Modul wurde dafür ebenso in einem Nebengehäuse untergebracht (s. Kap. 8.2). Die Antenne des Moduls wurde dabei mittels Kabelbinder an der Stütze angebracht. Dieses Vorgehen war einer dauerhaften Anbringung, etwa durch Kleber, vorzuziehen, da die Bauteile so leichter ausgetauscht werden können. Zu sehen ist dies in Abbildung 7.

Das Modul hat fünf Anschlüsse V<sub>CC</sub>, GND, TXD, RXD und PPS. Der PPS-Anschluss blieb unverbunden (s. Abb. 8).

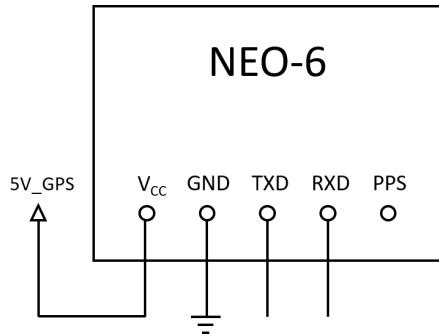


Abbildung 8: Verschaltung des NEO-6

#### 3.3 Anemometer und Windfahne: SEN-08942

Bei unseren Vorgängern waren diese Sensoren ein Problem, da sie keine Dokumentation zu einer seriellen Kommunikation hatten und zunächst aufwändig mittels Reverse-Engineering herausfinden mussten, wie sie den Sensor zu verwenden hatten. Bei diesem Projekt entschied man sich daher für einen neuen Sensor. Die Suche wurde dadurch erschwert, dass Anemometer und Windfahnen oft in einem System mit einer schon vorhandenen Wetterstation angeboten werden. Die Entscheidung fiel schlussendlich zu Gunsten des **SEN-08942**. Dieses System besteht aus einer Windfahne, einem Anemometer und einem Regenmesser.

Das Anemometer misst den Wind in dem es bei jeder Umdrehung einen Impuls ausgibt. In dem rotierendem System ist ein Magnet verbaut der an einem Schalter vorbei rotiert und so den Impuls verursacht. Ein Impuls pro Sekunde entspricht dabei laut Datenblatt einer Windgeschwindigkeit von  $2.4 \text{ km h}^{-1}$  [3].

Die Funktion der Windfahne basiert darauf, dass 8 Schalter, jeweils mit einem anderen Widerstand verbunden, verbaut sind (s. Abb 9). Je nach Position der Fahne sind bis zu zwei Schalter geschlossen. Mit einem externen Widerstand kann dann ein Spannungsteiler genutzt werden, um die zu messende Spannung in eine Windrichtung überführen.

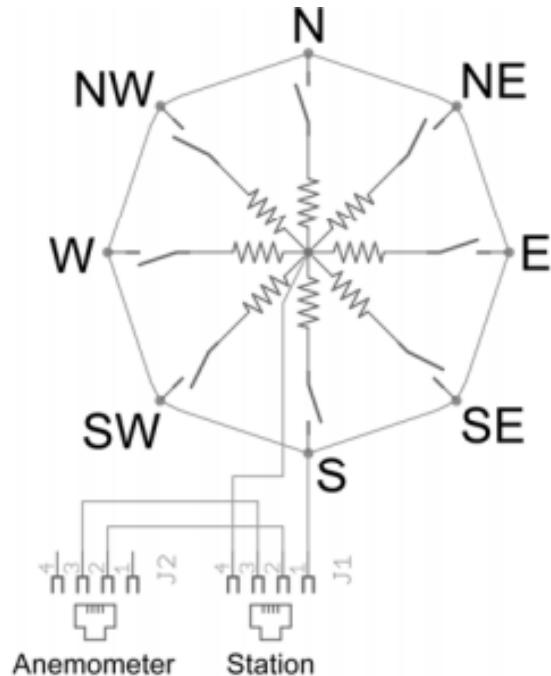


Abbildung 9: Aufbau der Windfahne und Anschluss an das Board[2]

Für einen Spannungsteiler mit einem externen Widerstand von  $10 \text{ k}\Omega$  ergeben sich die Werte aus Abbildung 10.

Direction (Degrees)	Resistance (Ohms)	Voltage (V=5v, R=10k)
0	33k	3.84v
22.5	6.57k	1.98v
45	8.2k	2.25v
67.5	891	0.41v
90	1k	0.45v
112.5	688	0.32v
135	2.2k	0.90v
157.5	1.41k	0.62v
180	3.9k	1.40v
202.5	3.14k	1.19v
225	16k	3.08v
247.5	14.12k	2.93v
270	120k	4.62v
292.5	42.12k	4.04v
315	64.9k	4.78v
337.5	21.88k	3.43v

Abbildung 10: Beispielwerte der Windfahne mit  $10\text{k}\Omega$  [3]

Der Regenmesser wurde im vorliegenden Aufbau nicht verwendet. Die Software zum Auslesen der beiden Sensoren findet sich in Kapitel 5.1.5.

### 3.3.1 Montage

Die Windfahne und das Anemometer wurde auf der mitgelieferten Strebe befestigt. Mit Hilfe des gedruckten Adapters (s. Kap. 8.3) wurden die beiden Sensoren, wie in Abbildung 7 zu sehen, auf einer der Stützen des Solarpanels befestigt.

Die Anschlüsse wurden wie in Abbildung 9 vorgenommen. Das mitgelieferte RJ 11 Kabel wurde direkt an das Board angeschlossen. Pin 2 und 3 sind dabei mit GND verbunden. PIN 4 wurde mit *Tim3\_ANEM\_CH1* und PIN 5 mit *ADC1\_WV\_IN16* verbunden. PIN 5 wurde zudem über einen Widerstand von  $10\text{k}\Omega$  mit der  $+3v3$  Leitung verbunden.

## 3.4 Neigungssensor MPU6050

Der Neigungssensor soll genutzt werden, um die Neigung des Solarpanels überprüfen zu können und so eine optimale Ausrichtung zu ermöglichen. Bei dem verwendeten Sensor handelt es sich um einen 6-Achsen Bewegungssensor, der ein 3-Achsen Gyroskop

mit einem 3-Achsen Beschleunigungssensor verbunden. Er kann mittels I<sup>2</sup>C-Schnittstelle angesprochen werden. [12]

Auf die Software wird in Kapitel 5.1.4 eingegangen.

#### 3.4.1 Montage

In Abbildung 11 sieht man, wie der Sensor verbunden ist. Er wurde wie das GPS-Modul und der Kompass auf eine Lochrasterplatine gesteckt und dann in einem der drei Nebengehäuse untergebracht. Anschließend wurde das Gehäuse mittels beidseitigem Klebeband hinten am Solarpanel angebracht (s. Abb. 7).

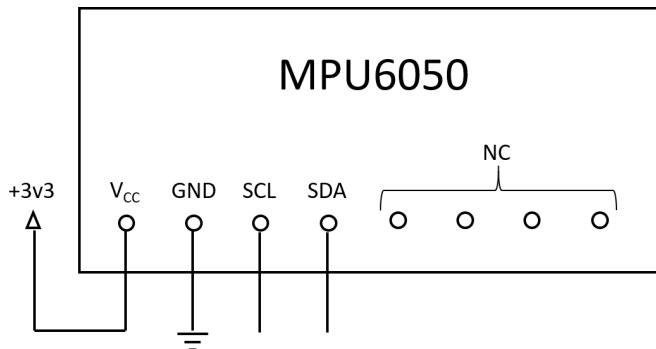


Abbildung 11: Verschaltung des MPU6050

#### 3.5 BME280

Der **BME280** ist ein Sensor, der Luftfeuchtigkeit, Luftdruck und Temperatur aufnehmen kann. Hergestellt wird er von der Firma **BOSCH**. Angesteuert wird er über die I<sup>2</sup>C-Schnittstelle. Die Luftfeuchtigkeit wird mit einer Genauigkeit von  $\pm 3\%$  bestimmt. Je nach Umgebungstemperatur kann der Luftdruck mit einer Genauigkeit von  $\pm 1$  bis  $\pm 1.7$  hPa genau bestimmt werden. Die Genauigkeit der Temperatur geht von minimal  $\pm 0.5^\circ\text{C}$  bis  $\pm 1.5^\circ\text{C}$ . Zudem bietet der Sensor die Möglichkeit, ihn in mehreren Betriebsmodi zu betreiben. So kann er etwa für längere Zeit im Sleepmodus gehalten werden und wird erst zur Messwertabfrage geweckt, was den Stromverbrauch reduziert.[9]

Die Software zu dem Sensor wird in Kapitel 5.1.1 genauer erläutert.

### 3.5.1 Montage

Der Sensor wurde direkt auf dem Board angebracht. Da das Hauptgehäuse (s. Kap. 8.1) offen ist, wird die Umgebungsluft ungehindert an den Sensor gelangen. Die Anschlüsse können aus Abbildung 12 entnommen werden.

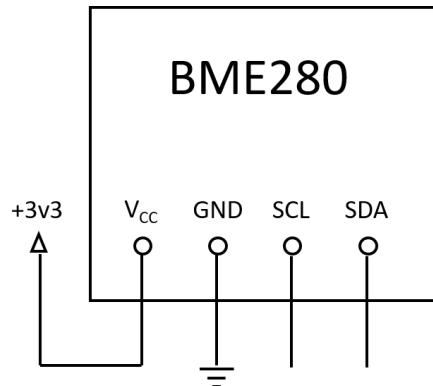


Abbildung 12: Verschaltung des BME280

## 3.6 Anschlagssensor

Um den Motor, der für die Rotation des Solarpanels zuständig ist, vor Schäden durch Überdrehen zu schützen, wurden die von unseren Vorgängern am Aufbau befestigten Anschlagssensoren genutzt. Diese beiden Sensoren sitzen nahe der Rotationsachse des Panels Rauf der Rückseite des Aufbaus. Bei Drehen des Solarpanels auf Anschlag kontaktiert einer der Sensoren und gibt ein Signal an den Mikrocontroller weiter. Software-seitig wird nun der Motor ausgeschaltet und dient zunächst als Notaus.

Des Weiteren können die Anschlagssensoren zur Kalibrierung der Stellung des Solarpanels verwendet werden.

### 3.6.1 Montage

Wie schon erwähnt sind die beiden Sensoren achsennah montiert. Angeschlossen sind die beiden Sensoren mit drei Kabeln: GND, einer Versorgungsspannung und einer Signalleitung.

## 4 Mikrocontroller

Die zentrale Steuereinheit der Wetterstation wird durch einen *STM8L152* Mikrocontroller von ST Microelectronics gebildet. Dieser ist zusammen mit einer 3.3V Spannungsversorgung, einem Quarzoszillator und einem Debug- und Programmieradapter auf einer kommerziell erhältlichen Entwicklungsplatine montiert. Die Verbindung zu den anwendungsspezifischen Modulen der Wetterstation geschieht über zwei zweireihige 38-pin Stiftleisten.

Die Anschlussbelegung der *Nucleo-64* Entwicklungsplatine ist standardisiert [19]; neben dem hier verwendeten *STM8* Mikrocontroller bietet ST Microelectronics Entwicklungsplatten in diesem Formfaktor für eine Reihe von Controllerfamilien an.

### 4.1 Auswahl des Mikrocontrollersystems

In der Voruntersuchungsphase des Projektes wurden unterschiedliche Mikrocontrollersysteme für den Einsatz in der Wetterstation untersucht. Hierbei konnte auf bereits vorliegende Hardware zugegriffen werden. Folgende Controllerfamilien wurden ausgewertet:

- Atmel, 8-bit AVR: ATmega328P (Arduino Uno)
- Atmel, 8-bit AVR: ATmega2560 (Arduino Mega)
- Atmel, 32-bit ARM Cortex-M3: SAM3X8E (Arduino Due)
- ST Microelectronics, 32-bit ARM Cortex-M4F: STM32F446 (NUCLEO-F446RE)
- ST Microelectronics, 8-bit STM8: STM8L152R8 (NUCLEO-8L152R8)

Für die Auswahl wurde neben den Hardwarespezifikationen auch die Qualität der Software-Werkzeuge und der Entwicklungskomfort in Betracht gezogen. Die Ergebnisse sind nachfolgend in der Entscheidungsmatrix (s. Tab. 1 und 2) aufgezeichnet.

Controller	UART	SPI	I <sup>2</sup> C	Analog In	Sleep I <sub>CC</sub>
ATmega328P [5]	1	1	1	6	4 mA
ATmega2560 [4]	4	1	1	16	7 mA
SAM3X8E [6]	5	4	2	16	8 mA
STM32F446RE [18]	6	4	4	16	10 mA
STM8L152R8 [17]	3	2	1	28	4 mA

Tabelle 1: Entscheidungsmatrix für die Auswahl des Mikrocontrollersystems (Hardware-spezifikationen)

Controller	IDE	Debugger	MCU Library	Periph. Library	Sleep	Aufwand
ATmega328P	Arduino IDE	nein	Arduino	ja	nein	sehr einfach
ATmega328P	Atmel Studio	ja (ext)	ASF	nein	ja	einfach
ATmega2560	Arduino IDE	nein	Arduino	ja	nein	sehr einfach
ATmega2560	Atmel Studio	ja (ext)	ASF	nein	ja	hoch
SAM3X8E	Atmel Studio	ja (ext)	ASF	nein	ja	sehr hoch
STM32F446RE	TrueStudio	ja	CMSIS	nein	ja	sehr hoch
STM8L152R8	STVD	ja	STM8-SPL	nein	ja	einfach

Tabelle 2: Entscheidungsmatrix für die Auswahl des Mikrocontrollersystems (Entwicklungsgrundlage)

Ausgehend hiervon wurde die Entscheidung getroffen, den *STM8L152R8* Mikrocontroller für das Projekt einzusetzen. Der primäre Kandidat, *ATmega328P* konnte aufgrund der zu geringen Anzahl von UART-Schnittstellen nicht eingesetzt werden.

## 4.2 Pin-Belegung

Für die Planung der Pin-Belegung wurde das Tool *STM8CubeMX* eingesetzt (s. Abb. 13), welches neben der Pin-Zuordnung auch die Konfiguration des Clock Tree dokumentieren kann und ein Analysewerkzeug für die Leistungsaufnahme bereitstellt.

Aus der grafischen Visualisierung des Controller-IC ist zu erkennen, dass die Anzahl der verfügbaren I/Os gerade optimal für diesen Anwendungsfall ist – der Controller ist also weder überdimensioniert, noch mussten Abstriche an den I/Os gemacht werden.

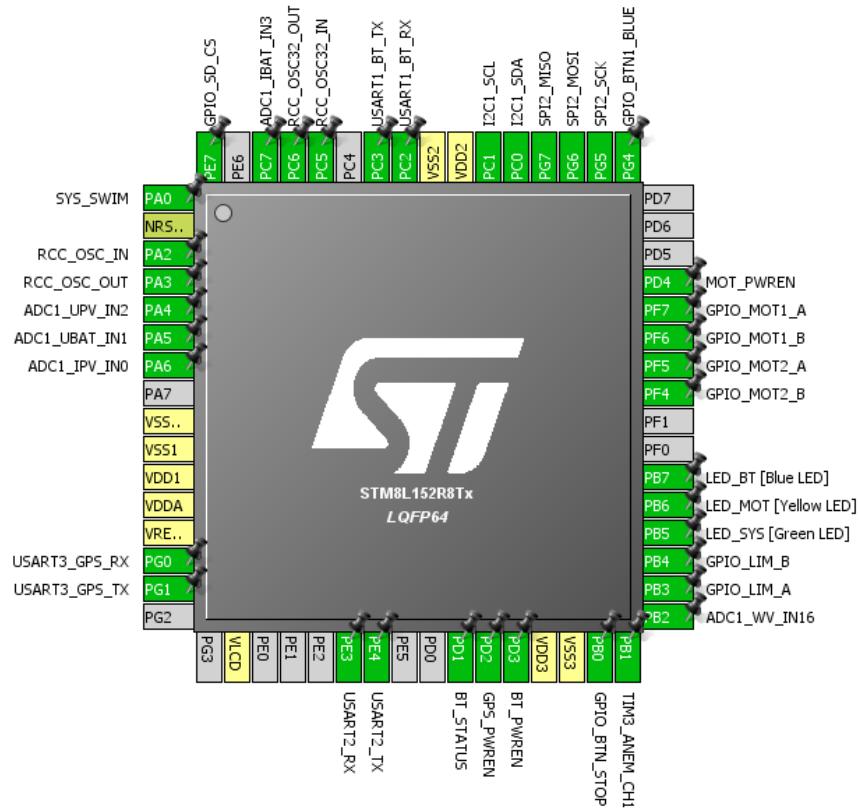


Abbildung 13: Pin-Zuordnungen für den STM8-Mikrocontroller

Peripheral	Verwendung
ADC1	CPU-Temperatursensor, Strom- und Spannungssensoren, Richtungssignal Windfahne
GPIO	Taster, LEDs, Endlagenschalter, Motor- treiber Steuersignale
I2C	Digitale Sensoren
SPI2	SD-Karte
TIM3	Zähler für Anemometer-Geberimpulse
UART1	UART-over-Bluetooth Umsetzer
UART2	printf()-Debugausgaben
UART3	NMEA 0183 Sentences vom GPS-Modul

Tabelle 3: Mikrocontroller-Peripheriemodule und ihr Einsatzzweck für die Wetterstation

## 5 Firmware

Die entwickelte Mikrocontroller-Firmware definiert den Funktionsumfang der Wetterstation: neben der Auswertung und Aufzeichnung von Messdaten erfolgt die Ausrichtung des Solarpanels softwaregesteuert.

Das Softwareprojekt ist dabei modular aufgebaut – voneinander unabhängige Softwarekomponenten sind in eigene Module aufgeteilt. Funktionen zur Umrechnung der rohen Sensordaten in die gewünschten Darstellungen werden durch entsprechende Unterteilung in verschiedene Headerdateien vor anderen Teilen der Software “versteckt”.

Hieraus wurde eine die folgende Projektstruktur entwickelt:

- `main` – Initialisierung des Systems beim Start und zyklische Datenverarbeitung im Hintergrund-Task
- `commlib/` – Module zur Ansteuerung der seriellen Kommunikationsschnittstellen (UART, SPI, I<sup>2</sup>C)
- `fslib/` – Implementierung des FAT-Dateisystems für die SD-Karte, basierend auf der Open-Source Bibliothek “FatFS” [10]
- `motorlib/` – Motorsteuerung mit Reaktion auf Betätigung der Endlagenschalter in einer Interruptserviceroutine
- `powerlib/` – C-Präprozessormakros für den Wechsel in den “wait-for-interrupt” Wartezustand
- `sensorlib/` – Module zur Auswertung der digitalen und analogen Sensoren
- `stm8lib/` – STM8 Standard Peripheral Library vom Mikrocontroller-Hersteller ST Microelectronics
- `userlib/` – Module zur Steuerung des Systems auf einer höheren Abstraktionsebene

Die Software wurde dabei für eine interruptgesteuerte Verarbeitung ausgelegt. Der Mikrocontroller wird durch Timer-Interrupts in regelmäßigen Zeitabständen “aufgeweckt”, um Steuerungsaufgaben auszuführen. Nach Abschluss der Ausführung wird er wieder in den Wartezustand “wait-for-interrupt” versetzt, um die Stromaufnahme während der Leerlaufzeit zu verringern.

## 5.1 Auswertung der Sensoren (**sensorlib**)

In diesem Abschnitt wird die firmwareseitige Konfiguration und Auswertung einiger der verwendeten Sensoren beschrieben.

### 5.1.1 BME280

Der BME280 Klimasensor kommuniziert über eine I<sup>2</sup>C-Schnittstelle mit dem Mikrocontroller. In der Initialisierungsphase des Systems wird der Sensor zunächst über einen *Reset*-Befehl auf seinen Einschaltzustand zurückgesetzt. Anschließend werden die im nicht-flüchtigen Speicher des Sensors programmierten Kalibrierungswerte für die Temperatur-, Luftfeuchte- und Luftdruckmung ausgelesen. Der Sensor wird bereits im kalibrierten Zustand durch den Hersteller ausgeliefert.

#### Sleep-Mode und Messwerterfassung

Nach dem Reset befindet sich der Sensor im “Sleep”-Mode, und es findet keine Messwerterfassung statt. Wenn der Mikrocontroller im zeitgesteuerten Interrupt aufgeweckt wird, versetzt dieser den Sensor in einen aktiven Zustand und startet die Aufzeichnung. Sobald die Messwerterfassung abgeschlossen ist, wird der Messwert durch den Controller ausgelesen. Anschließend wird der Sensor wieder in den “Sleep”-Mode zurück versetzt - die rechenintensive Temperaturkompensation der rohen Messwerte erfolgt währenddessen im Mikrocontroller.

#### Umrechnung der Rohdaten

Die vom Sensor ausgelesenen Rohdaten müssen für einen durch die Umgebungstemperatur verursachten Fehler kompensiert werden. Für die Implementierung der Kompensationsroutinen wurde der Quellcode der von Bosch Sensortec bereitgestellten “BME280\_driver” [8] Bibliothek als Grundlage gewählt und für den Einsatz auf dem STM8 Mikrocontroller angepasst.

#### Sensordaten-Speicherstruktur

Die Sensor-Kalibrierungs- und -Messdaten werden im Arbeitsspeicher des Mikrocontrollers zwischengespeichert, um die Zugriffszeit auf Kalibrierungsdaten zu verringern. Die

Daten werden hierbei in einem Strukturtyp (s. Listing 1) angeordnet – es können so bei Bedarf mehrere BME280-Sensoren am System angeschlossen werden<sup>1</sup>.

```
/*! ****
 * @brief
 * Sensordaten- und Konfigurationsstruktur für BME280
 *
 * @date 28.10.2019
 ****/
typedef struct tag_BME280_Sensor {
    /*! I2C Slave-Adresse */
    uint8_t ucSlaveAddr;

    /*! Kalibrierungsdaten */
    struct
    {
        /*! Kalibrierungsdaten für Temperaturmessung */
        uint16_t uiDigT1;
        int16_t iDigT2;
        int16_t iDigT3;

        /*! Kalibrierungsdaten für Druckmessung */
        uint16_t uiDigP1;
        int16_t iDigP2;
        int16_t iDigP3;
        int16_t iDigP4;
        int16_t iDigP5;
        int16_t iDigP6;
        int16_t iDigP7;
        int16_t iDigP8;
        int16_t iDigP9;

        /*! Kalibrierungsdaten für Luftfeuchtemessung */
        uint8_t ucDigH1;
        int16_t iDigH2;
        uint8_t ucDigH3;
        int16_t iDigH4;
        int16_t iDigH5;
        int8_t cDigH6;
    } sCalib;

    /*! Rohdaten */
    struct
    {
        /*! Kompensationswert für Lufttemperatur */
        int32_t lTfine;

        /*! Rohdaten für Temperatur */
    }
}
```

---

<sup>1</sup>Hierzu muss der SDO-Pin eines der beiden Sensoren an VCC angeschlossen werden, damit die I<sup>2</sup>C Slave-Adressen sich nicht überlagern [9, Kap 6.3].

```

    uint32_t ulRawTemp;

    /*! Rohdaten für Luftdruck
    uint32_t ulRawPress;

    /*! Rohdaten für Luftfeuchte
    uint16_t uiRawHum;
} sRaw;

/*! Kompensierte Messwerte
struct
{
    /*! Lufttemperatur in 0.01°C
    int16_t iTemperature;

    /*! Luftdruck in 0.01 hPa
    uint32_t ulPressure;

    /*! Luftfeuchtigkeit in 0.01 %RH
    uint32_t ulHumidity;
} sMeasure;
} BME280_Sensor;

```

Listing 1: Typdefinition für die Sensordaten-Speicherstruktur des BME280

### 5.1.2 CPU-Temperatursensor

Als weiterer Sensor für die Umgebungstemperatur wird der auf dem Mikrocontroller-Die integrierte Temperatursensor ausgewertet. Über das Messverfahren werden werder im Datenblatt noch im *Reference Manual* genauere Angaben gemacht, jedoch ist eine Realisierung des Messelements als “Silicon Bandgap” mit pn-Übergängen im Halbleitermaterial wahrscheinlich.

Das Ausgangssignal der Messschaltung kann über den Analog-Multiplexer auf den Eingang des internen ADU durchgeschaltet und aufgezeichnet werden. Kalibrierungswerte für den Offset und die Steigung der Temperaturkennlinie werden vom Hersteller als Teil des Fertigungsprozesses bauteilspezifisch ermittelt und im FLASH des Mikrocontrollers einprogrammiert.

#### Aktivierung und Messwerterfassung

Die Versorgung des internen Temperatursensors wird in der Initialisierungsphase des Mikrocontrollers aktiviert. Für die Temperaturmessung wird der Ausgang des Sensors

auf den ADU geschaltet, und der umgesetzte Wert eingelesen. Die Umrechnung in einen Temperaturwert erfolgt in diesem Fall unter Berücksichtigung von Datenblattangaben für eine typische Sensorcharakteristik. Eine genauere Kompensation ist mithilfe der im FLASH gespeicherten Kalibrierungswerte möglich, wurde aber aus Zeitgründen nicht umgesetzt.

### Sensordaten-Speicherstruktur

Zur besseren Lesbarkeit des Codes wurde für den CPU-Temperatursensor ebenfalls eine Speicherstruktur definiert, sodass die Temperaturmesswerte auf gleiche Weise wie andere Sensoren ausgewertet werden kann (s. Listing 2). Die Strukturdefinition könnte zukünftig durch Felder für Kalibrierungsdaten erweitert, und so zur Verwendung für allgemeine Temperatursensoren mit annähernd linearer Kennlinie eingesetzt werden.

```
/*!*****
 * @brief
 * Sensor-Struktur für den CPU-Temperatursensor
 *
 * @date 31.10.2019
 *****/
typedef struct tag_CPUTemp_Sensor
{
    /*! Rohdaten
    struct
    {
        /*! Temperatur-Rohwert
        uint16_t uiRawTemp;
    } sRaw;

    /*! Umgerechnete Messwerte
    struct
    {
        /*! Prozessortemperatur in 1°C
        int8_t cTemp;
    } sMeasure;
} CPUTemp_Sensor;
```

Listing 2: Typdefinition für die Sensordaten-Speicherstruktur des internen Temperatursensors

#### 5.1.3 QMC5883L

Zur Bestimmung der Kompassrichtung für den Regelkreis der Turmausrichtung wird das Magnetometer QMC5883L von QST eingesetzt. Dieser Sensortyp ist eine unter Lizenz

von Honeywell gefertigte Version des HMC5883 Magnetometers [15, S. 1].

In der Initialisierungsphase des Systems wird der Sensor mittels eines Reset-Befehls auf seinen Ausgangszustand zurückgesetzt, und die Kalibrierungsdaten in der Speicherstruktur des Mikrocontrollers werden auf empirisch ermittelte Standardwerte initialisiert. Anschließend wird die Empfindlichkeit und Filterkonfiguration des Sensors parametriert, und die kontinuierliche Messwerterfassung mit einer Frequenz von 10 Hz aktiviert.

Der Sensor kann für die Leerlaufzeit des Mikrocontrollers in einen “Sleep”-Zustand versetzt werden. Jedoch kommt es dann beim Wiedereinschalten zu Abweichungen in den Messwerten, welche sich erst nach einer unbestimmten Verzögerungszeit im Bereich von einigen Millisekunden stabilisieren.

**Kalibrierung** Bevor der Sensor zur Bestimmung der Kompassrichtung genutzt werden kann, müssen die Magnetometer-Ausgabewerte kalibriert werden. Der Turm wird hierzu zeitgesteuert um  $360^{\circ}$  rotiert, und währenddessen Minimal-, Maximal- und Mittelwert der X- und Y-Komponenten des gemessenen Magnetfeldes aufgezeichnet (s. Abb. 14).

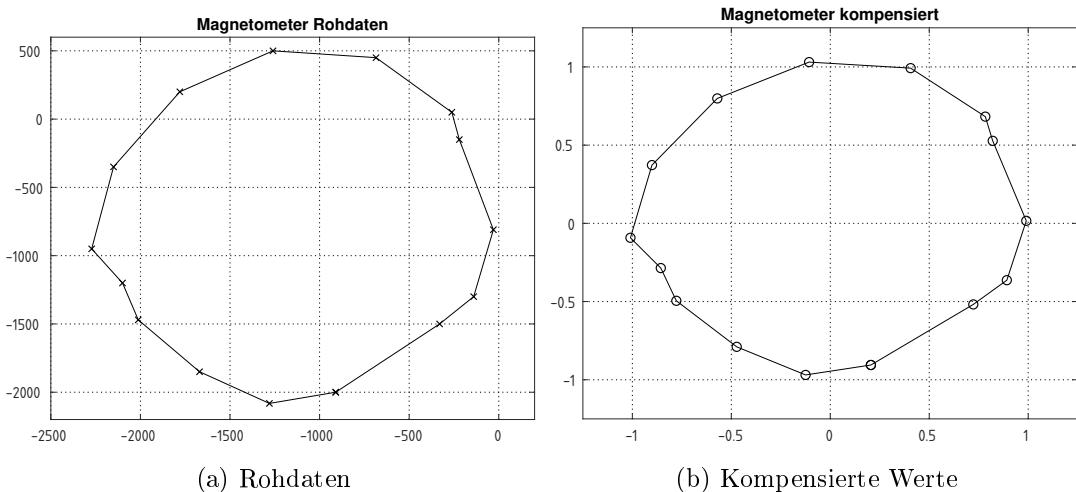


Abbildung 14: Kalibrierung des QMC5883L Magnetometers. Achsenbeschriftung ist zu beachten.

Die Kompensation der Magnetfeldkomponenten erfolgt anschließend entsprechend der Formel:

$$x_{\star} = (x - \bar{x}) \cdot \frac{2}{|\max x - \min x|} .$$

### Berechnung der Kompassrichtung

Die Kompassrichtung kann aus zwei Magnetfeldkomponenten bestimmt werden. Für diese Anwendung wurden die X- und Y-Feldkomponenten gewählt. Nachdem die oben beschriebene Kalibrierung erfolgt ist, kann daraus über den arctan die Richtung des Feldvektors relativ zur Ausrichtung des Bauteils ermittelt werden:

$$\theta = \arctan\left(\frac{y}{x}\right)$$

Zur Berücksichtigung des Falls  $x < 0$  wird dabei die Funktion atan2 eingesetzt:

$$\theta = \text{atan2}(x, y)$$

Da dies einen Winkel im Bereich  $\theta \in [-\pi, \pi]$  liefert, muss eine Offsetkompensation und Umrechnung in Winkelgrade angewendet werden:

$$\theta = 180^\circ + \text{atan2}(x, y) \cdot \frac{180^\circ}{\pi}$$

Für die Darstellung als Festkommazahl mit  $[\theta_*] = 0.1^\circ$  kann der Faktor durch eine Integerkonstante ersetzt werden mit

$$10 \cdot \frac{180^\circ}{\pi} \approx 573 \implies \theta_* = (1800 + \text{atan2}(x, y) \cdot 573) \cdot (0.1^\circ)$$

### Temperatursensor

Der QMC5883 Sensor beinhaltet zusätzlich zum Magnetometer einen Temperatursensor zur internen Kompensation. Die Messwerte des Temperatursensors können zudem über die I<sup>2</sup>C-Schnittstelle durch den Mikrocontroller ausgelesen werden.

Der bereitgestellte Wert beschreibt dabei jedoch nur einen Offset zu einer intern kalibrierten Referenztemperatur. Diese ist im Datenblatt des Sensors nicht angegeben, und muss durch den Anwender empirisch ermittelt werden. In diesem Fall wurde ein Referenzwert von 34°C ermittelt.

### Sensordaten-Speicherstruktur

Die Kalibrierungsdaten zur Kompensation der X- und Y-Magnetfeldkomponenten, sowie der Temperatur-Referenzwert werden in einer Speicherstruktur im Arbeitsspeicher des Mikrocontrollers festgehalten (s. Listing 3). Die umgerechneten Messwerte werden für die Auswertung durch andere Programmteile bereitgestellt.

```
/*! ****
 * @brief
 * Sensordaten- und Konfigurationsstruktur für QMC5883
 *
 * @date 31.10.2019
 ****/
typedef struct tag_QMC5883_Sensor {
    /*! I2C Slaveadresse */
    uint8_t ucSlaveAddr;

    /*! Kalibrierungsdaten */
    struct {
        /*! Referenztemperatur in 0.1°C */
        int16_t iRefTemp;

        /*! Mittelwerte */
        float fXComp;
        float fYComp;

        /*! Minima und Maxima der Felder während der Drehung */
        int16_t iXMin;
        int16_t iXMax;
        int16_t iYMin;
        int16_t iYMax;

        /*! Normierungsfaktoren */
        float fXGain;
        float fYGain;

        /*! Anzahl der Messpunkte für die Kompensation */
        uint16_t uiNumComp;
    } sCalib;

    /*! Kalibrierungsfahrt aktiv */
    bool bCalActive;

    /*! Rohdaten */
    struct {
        /*! Feld in X-Richtung */
        int16_t iRawX;

        /*! Feld in Y-Richtung */
        int16_t iRawY;

        /*! Feld in Z-Richtung */
        int16_t iRawZ;

        /*! Temperatur-Rohwert */
        int16_t iRawTemp;
    } sRaw;
}
```

```
/*! Umgerechnete Messdaten */  
struct {  
    /*! Azimuth in 0.1° */  
    uint16_t uiAzimuth;  
  
    /*! Temperatur in 0.01°C */  
    int16_t iTemperature;  
} sMeasure;  
} QMC5883_Sensor;
```

Listing 3: Typdefinition für die Sensordatenstruktur des QMC5883L

#### 5.1.4 MPU6050

Das MPU6050 von InvenSense wird zur Bestimmung des Panel-Winkels eingesetzt. Im Baustein ist neben den MEMS-Accelerometern und -Gyroskopen für 3 Achsen zusätzlich ein Temperatursensor für die interne Kompensation implementiert.

Von den sieben Sensoren werden für diese Anwendung jedoch nur die drei Accelerometerachsen ausgewertet – der Temperatursensor hatte sich in der Evaluierungsphase als zu ungenau erwiesen.

In der Initialisierungsphase des Systems wird der Baustein im “Sleep”-Modus neu gestartet, und die ungenutzten Sensoren deaktiviert. Der Messbereich für das Accelerometer wird auf  $\pm 2\text{ g}$  parametriert.

#### Aktivierung und Messdatenerfassung

In der Aufzeichnungsroutine des Mikrocontrollers, welche durch einen zeitgesteuerten Interrupt ausgelöst wird, wird der Sensor aus dem “Sleep”-Modus geweckt und die Messdatenerfassung gestartet. Die Umsetzung der Messwerte ist erst nach einer kurzen Verzögerung von einigen Prozessorzyklen abgeschlossen, und wird durch ein “Data Ready”-Flag in einem Sensor-Register signalisiert.

Die rohen Messwerte vom Accelerometer werden anschließend in die Sensordaten-Struktur im Arbeitsspeicher des Mikrocontrollers eingelesen und in einen Winkel  $\alpha_* \in [-1800; 1800] \cdot (0.1^\circ)$  umgerechnet.

### Berechnung des Ausrichtungswinkels

Für die Ausrichtung des Panels werden die Y- und Z-Komponenten der Beschleunigung relativ zur Ausrichtung des Bausteins betrachtet. Für den aus den Komponenten gebildeten Vektor wird über die arctan-Funktion der Winkel zur Grundebene ermittelt:

$$\alpha = \arctan\left(\frac{y}{z}\right)$$

Für die Behandlung des Falls  $z < 0$  wird hier, ähnlich zur Berechnung der Kompassrichtung beim QMC5883, die atan2-Funktion eingesetzt, welche einen Winkel  $\alpha \in [-\pi, \pi]$  liefert.

Dieser wird für die interne Darstellung in  $[\alpha_\star] = 0.1^\circ$  umgerechnet.

$$\alpha_\star = \text{atan2}(z, y) \cdot \frac{1800 \cdot (0.1^\circ)}{\pi}$$

Der konstante Faktor kann dabei durch eine Integerkonstante ersetzt werden:

$$\alpha_\star = \text{atan2}(z, y) \cdot 573 \cdot (0.1^\circ) .$$

### Sensordaten-Speicherstruktur

Die Sensor-Messdaten werden in einem Strukturtyp im Arbeitsspeicher des Mikrocontrollers festgehalten (s. Listing 4) und durch unterschiedliche Programmteile ausgewertet. Über ein Feld für die Slaveadresse des Gerätes können bei Bedarf mehrere MPU6050-Sensoren angeschlossen werden<sup>2</sup>.

```
/*! ****
 * @brief
 * Konfigurations- und Sensordatenstruktur für MPU6050
 *
 * @date 06.11.2019
 ****/
typedef struct tag_MP6050_Sensor {
    /*! I2C Slaveadresse */
    uint8_t ucSlaveAddr;

    /*! Temperaturmessungen aktiv */
    bool bMeasureTemp;

    /*! Rohdaten */
    struct {
        /*! Rohwert für X-Beschleunigung */
    };
```

<sup>2</sup>Hierzu muss der AD0-Pin des zweiten Bausteins an VCC angeschlossen werden [12, Kap 9.2].

```

int16_t iRawX;
/*! Rohwert für Y-Beschleunigung */
int16_t iRawY;

/*! Rohwert für Z-Beschleunigung */
int16_t iRawZ;

/*! Rohwert für Temperatur */
int16_t iRawTemp;
} sRaw;

/*! Umgerechnete Messwerte */
struct {
    /*! Einstellwinkel in 0.1° */
    struct {
        int iXZ;
        int iYZ;
    } sAngle;

    /*! Temperaturmesswert in 0.01°C */
    int16_t iTemperature;
} sMeasure;
} MPU6050_Sensor;

```

Listing 4: Typdefinition für die Sensordatenstruktur des MPU6050

### 5.1.5 Anemometer und Windfahne

Für dieses Projekt wurde ein neuer Aufbau für das Anemometer und die Windfahne eingesetzt, welcher als Schnittstelle zum Mikrocontroller eine analogen Spannung und ein digitales Gebersignal verwendet [2]. Somit konnte auf die Implementierung des proprietären Kommunikationsprotokolls für den Aufbau der Gruppe vom Vorjahr verzichtet werden.

Auf der Seite des Mikrocontrollers werden zur Auswertung dementsprechend ein Kanal des ADU und ein Timer-Zähler Modul eingesetzt.

**Anemometer** Die Windgeschwindigkeit wird über das Drehgebersignal des Anemometers ermittelt. Hierzu wird in der Initialisierungsphase des Systems ein Hardware Timer des Mikrocontrollers als Impulszähler konfiguriert. Ein Impuls vom Anemometer auf dem Zähleingang inkrementiert den Zählerstand, welcher dann in einem festen Zeitintervall ausgewertet werden muss.

Die Windgeschwindigkeit liegt dann in einer Einheit “Impulse pro Zeitintervall” vor – praktischerweise wird der Zählerstand hier mit einer Periode von 1 s ausgewertet, wodurch die Windgeschwindigkeit entsprechend in einer Einheit “Impulse pro Sekunde” vorliegt.

Aus den Datenblattangaben kann hierfür ein Umrechnungsfaktor ermittelt werden [2]:

$$v_w \approx n \cdot 2.4 \cdot \frac{\text{km/h}}{\text{s}} ,$$

mit  $n$  als Anzahl der Impulse im vergangenen Zeitintervall von 1 s. Es gilt  $[n] = 1 \text{ pps}$ .

Für die interne Darstellung in  $[v_{w*}] = 1 \text{ m/s}$  muss dieser Wert entsprechend umgerechnet werden mit

$$1 \text{ m/s} = 3.6 \text{ km/h} = 2.4 \frac{\text{km/h}}{\text{pps}} \cdot 3.6 \approx \frac{553}{2^6} \cdot \left( 1 \frac{\text{m/s}}{\text{pps}} \right)$$

in

$$v_{w*} = \left\lfloor \frac{n \cdot 553}{2^6} \right\rfloor \cdot (1 \text{ m/s}) .$$

Da die Erfassung der Windgeschwindigkeit ohnehin alle 1 s erfolgt, wird aus den Messwerten die mittlere Windgeschwindigkeit über die letzten 16 Sekunden, sowie die maximale Windgeschwindigkeit (z.B. Böen) in diesem Zeitfenster ermittelt.

## Windrichtung

Ein an der Windfahne angebrachter Magnet schließt je nach Windrichtung einen Widerstand oder zwei Widerstände in Parallelschaltung an einen Spannungsteiler. Über die Spannung am Ausgang des Teilers kann eine Zuordnung zu einer relativen Windrichtung erfolgen. Für die Messung wird in der Initialisierungsphase des Systems ein ADU-Kanal parametriert.

Für die Zuordnung zwischen ADU-Wertebereich und Windrichtung wird eine *Look-Up Table* eingesetzt. Die ADU-Werte hierfür wurden empirisch ermittelt. Die Windfahne ist mit acht Widerständen ausgestattet – es können also 16 diskrete Werte als Windrichtung erkannt werden:

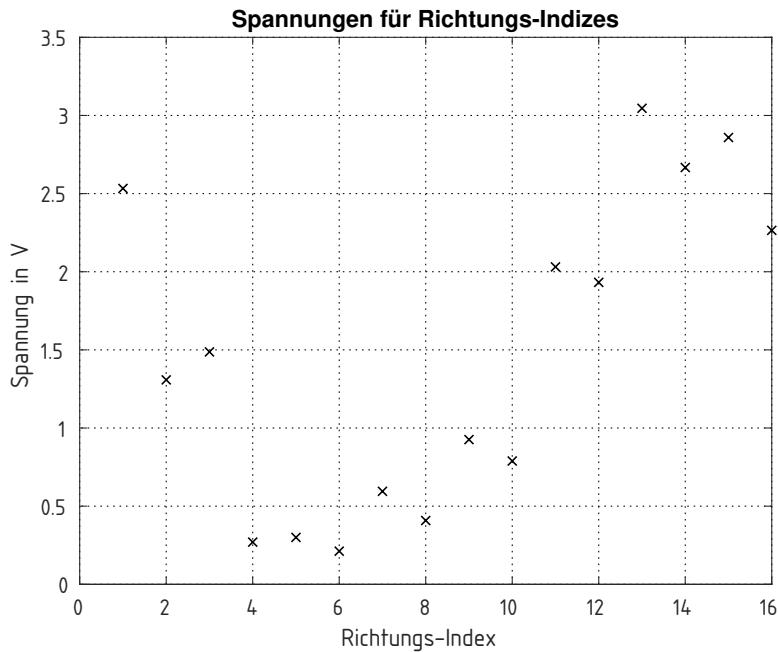


Abbildung 15: ADU-Werte für 16 diskrete Richtungs-Werte (in positive Drehrichtung) der Windfahne

Die absolute Windrichtung kann nur dann ermittelt werden, wenn der digitale Kompass QMC5883 kalibriert wurde. Der über die LUT ermittelte Richtungsindex  $k$  wird dann als Offset aufaddiert:

$$\theta_w = \theta + k \cdot 22.5^\circ ,$$

bzw. für die interne Darstellung mit  $[\theta_{w*}] = 0.1^\circ$ :

$$\theta_{w*} = \theta_* + k \cdot 225 \cdot (0.1^\circ) .$$

### Sensordaten-Speicherstruktur

Die Messwerte für die Windgeschwindigkeit werden zur Mittelwertbildung in einem Ringspeicher mit 16 Elementen abgelegt. Zusätzlich die Böen-Windgeschwindigkeit in einer weiteren Speicherstelle abgelegt.

```
/*
 * @brief
 *   Windsensor-Struktur
 *
 * @date 31.10.2019
 */
```

```

typedef struct tag_Wind_Sensor
{
    /*! Kalibrierungsdaten */
    struct
    {
        /*! Timer-Abfrageintervall */
        uint16_t uiPollInterval;
    } sCalib;

    /*! Rohdaten */
    struct
    {
        bool bRawDataUpdate;
        uint8_t ucHead;
        uint16_t auiRawVelocity[NUM_WIND_AVG];
        uint16_t uiRawDirection;
    } sRaw;

    /*! Umgerechnete Messwerte */
    struct
    {
        /*! Windgeschwindigkeit in m/s */
        uint16_t uiAvgVelocity;
        uint16_t uiMaxVelocity;

        /*! Windrichtung als Himmelsrichtung */
        Wind_Direction eDirection;
    } sMeasure;
} Wind_Sensor;

```

Listing 5: Typdefinition für die Messdaten zur Bestimmung der Windrichtung und -geschwindigkeit

### 5.1.6 Strom- und Spannungsmessung

Eine weitere Anforderung an die Wetterstation war die Aufzeichnung des Batterie-Ladezustandes in Form von Strom- und Spannungsmesswerten.

Für die Spannungsmessung wird der ADU des Mikrocontrollers mit vorgeschalteten Spannungsteilern eingesetzt. Die Strommessung erfolgt mithilfe von ACS712-Stromsensoren, welche die Ströme in analoge Spannungswerte umsetzen, welche ebenfalls über den ADU eingelesen werden. Um den Mikrocontroller vor Überspannungen an den Analogeingängen zu schützen, sind Spannungsteiler mit Begrenzungsdioden vorgeschaltet.

Für die umgesetzten Spannungswerte wird eine Offsetkompensation (Strommessung) durchgeführt und die empirisch ermittelte Kennliniensteigung angewendet.

### Sensordaten-Speicherstruktur

Die Strom- und Spannungsmessung erfolgt für Akku und Solarpanel auf gleiche Weise, daher ist die Struktur so ausgelegt, dass sie universell für Kombinationen aus Strom- und Spannungssensoren eingesetzt werden kann. Zusätzlich werden in ihr die Nummern der jeweiligen ADU-Kanäle für die Umstellung des Analog-Mux festgehalten.

```
/*! ****
* @brief
* Strukturdefinition für Strom- und Spannungsmessung
*
* @date 06.01.2020
*****/
typedef struct tag_Power_Sensor {
    /*! Kalibrierungsdaten
    struct
    {
        /*! Nullpunktabgleich für die Strommessung als Rohw. */
        uint16_t uiCurrZeroOffset;

        /*! Kennliniensteigung für den Strom in 1mA/lsb */
        int16_t iCurrSlope;

        /*! Kennliniensteigung für die Spannung in 1mV/lsb */
        uint16_t uiVoltSlope;
    } sCalib;

    /*! Konfiguration der ADC-Kanäle
    struct
    {
        /*! Kanal für die Strommessung
        ADC_Channel_TypeDef eCurr;

        /*! Kanal für die Spannungsmessung
        ADC_Channel_TypeDef eVolt;
    } sChannel;

    /*! Rohdaten
    struct
    {
        /*! ADC Rohwert für die Spannungsmessung
        uint16_t uiRawVolt;

        /*! ADC Rohwert für die Strommessung
        uint16_t uiRawCurr;
    } sRaw;
```

```
/*! Umgerechnete Messwerte */  
struct {  
    /*! Spannungsmesswert in 1mV */  
    uint16_t uiVolt;  
  
    /*! Strommesswert in 1mA */  
    int16_t iCurr;  
} sMeasure;  
} Power_Sensor;
```

Listing 6: Typdefinition für die Sensordatenstruktur der Strom- und Spannungssensoren

## 5.2 GPS

Das GPS-Modul sendet nach dem Einschalten der Versorgung sofort zyklisch Daten im NMEA 0183 Format über die UART-Schnittstelle an den Mikrocontroller. In der Initialisierungsphase wird hierfür ein GPIO-Pin für die Ansteuerung des Leistungs-MOSFETs der Versorgung, und die USART3 für den Datenempfang bei 9600 Baud (8N1) konfiguriert.

Die eingehenden Daten werden mittels eines Interrupthandlers in einen Puffer eingelesen. Hierbei können einige Eigenschaften des NMEA 0183 Protokolls ausgenutzt werden, um die Konsistenz der Daten zu sichern:

- Jeder *Sentence* beginnt mit dem Zeichen “\$”.
- Der *Sentence* endet mit dem Zeilenende, markiert durch “\r\n”.
- Auf das Startzeichen folgt eine 5-stellige *Sentence*-Kennung, gefolgt von einem Komma. Mögliche Werte hierfür sind bspw. “GPRMC” oder “GPGGA”.
- Die Datenfelder des *Sentence* sind durch Kommata getrennt.

### 5.2.1 Interpretation der NMEA Sentences

Sobald die ersten 6 Zeichen eingelesen wurden, wird eine Abfrage gestartet, ob es sich hierbei um einen für die Auswertung relevanten *Sentence* handelt.

Aus der Anforderungsspezifikation folgt, dass neben den Positionsdaten der aktuelle UTC-Zeitstempel und die Höhe über dem mittleren Meeresspiegel aus den GPS-Daten ermittelt werden sollen. Die relevanten NMEA Sentences hierfür sind:

- GPRMC – “Minimum recommended GPS/transit data”
- GPGGA – “Global Positioning System Fix Data”

Die zugehörigen Sentence-Formate werden nachfolgend kurz beschrieben. Bei der Auswertung werden die Sentences zeichenweise durchlaufen und eine Text-Interpretation durchgeführt.

#### **GPRMC: “Minimum recommended GPS/transit data”**

Beispiel:

\$GPRMC,225446.00,A,5355.63,N,01002.26,W,082.5,054.7,031219,020.3,E\*4D\r\n

Beschreibung:

Feld	Beschreibung	Wert
\$	Startzeichen	
GPRMC	Sentence-Typ	
225446.00	UTC-Uhrzeit im Format hhmmss.zz	22:54:46 Z
A	Gültigkeit der Daten: A = gültig, V = kein Fix	gültig
5355.63,N	Breitengrad im Format ddmm.mm, N = Nord, S = Süd	N 53° 55.63'
01002.26,E	Längengrad im Format dddmm.mm, E = Ost, W = West	E 010° 02.26'
082.5	Geschwindigkeit über Grund in Knoten	82.5 kts
054.7	Rechtweisender Kurs	054.7 °
031219	UTC-Datum im Format DDMMYY	03.12.2019
020.3,E	Variation, E = Ost, W = West	20.3 ° Ost
*4D	Prüfsumme	
\r\n	Zeilenende	

Tabelle 4: Beschreibung des GPRMC NMEA Sentence nach [7]

### GPGGA: “Global Positioning System Fix Data”

Beispiel:

\$GPGGA,225446.00,5355.63,N,01002.26,E,1,09,1.5,419.3,M,39.5,M,\*66\r\n

Beschreibung:

Feld	Beschreibung	Wert
\$	Startzeichen	
GPGGA	Sentence-Typ	
225446.00	UTC-Uhrzeit im Format hhmmss.zz	22:54:46 Z
5355.63,N	Breitengrad im Format ddmm.mm, N = Nord, S = Süd	N 53° 55.63'
01002.26,E	Längengrad im Format dddmm.mm, E = Ost, W = West	E 010° 02.26'
1	Fix Typ: 0 = kein Fix, 1 = GPS, 2 = differential GPS	GPS
09	Anzahl der empfangenen Satelliten	9
1.5	Relative Genauigkeit der Position (HDOP)	
419.3,M	Höhe über dem mittleren Meeresspiegel (MSL)	419.3 m
39.5,M	Höhe über dem WGS84-Geoid	39.5 m
(leer)	Zeit seit letztem DGPS-fix (bei GPS leer)	
(leer)	DGPS-Stationskennung (0000-4096, bei GPS leer)	
*66	Prüfsumme	
\r\n	Zeilenende	

Tabelle 5: Beschreibung des GPGGA NMEA Sentence nach [7]

#### 5.2.2 Versorgung des GPS-Moduls

Um die Energieaufnahme des Systems zu verringern, wird das GPS-Modul nach dem ersten Positions-Fix abgeschaltet und nur in längeren Zeitabständen zur Synchronisation der prozessorinternen Echtzeituhr reaktiviert.

Die Abschaltung erfolgt über einen Leistungs-MOSFET auf der Versorgungsleitung des Moduls, welcher über einen GPIO-Pin des Controllers geschaltet wird.

### 5.2.3 GPS-Datenstruktur im Speicher

Die aus den oben beschriebenen Sentences eingelesenen Daten werden in einer Speicherstruktur für die Auswertung durch weitere Programmteile, beispielsweise die Sollwertberechnung für die Nachführung des Solarpanels, bereitgestellt.

## 5.3 Bluetooth

Für die drahtlose Kommunikation mit einem Rechner wird ein Serial-over-Bluetooth Umsetzermodul eingesetzt. Es ist am Mikrocontroller an der USART2-Schnittstelle angeschlossen und kann, ähnlich wie das GPS-Modul, über einen Leistungs-MOSFET an der Versorgung abgeschaltet werden.

Nach Installation des Gerätetreibers am PC erscheint die Schnittstelle als virtueller COM-Port. Die Kommunikation kann dann über ein Terminal-Programm wie *PuTTY* oder die im Kapitel 7 beschriebene grafische Benutzeroberfläche erfolgen.

### 5.3.1 AT-Befehlssatz

Für die Kommunikation wurde ein auf dem *Hayes command protocol* basierender AT-Befehlssatz implementiert. Merkmal des Protokolls ist, dass Befehle vom Benutzerterminal stets mit der Zeichenfolge “AT” beginnen, und mit der Zeichenfolge “\r” enden.

Nachfolgend werden einige der verfügbaren Befehle beschrieben.

#### AT: Verbindungsstatus prüfen

Der Befehl “AT” kann zur Überprüfung des Verbindungsstatus genutzt werden. Die Wetterstation beantwortet alle Statusanfragen mit “OK”.

Execute-Command	Antwort
AT	OK

### ATE: Remote-Echo aktivieren oder deaktivieren

Wenn die Befehlseingabe manuell über ein Terminalprogramm erfolgt (bspw. *PuTTY*), ist eine Ausgabe der getippten Zeichen für den Anwender hilfreich. Diese *Remote Echo*-Funktion kann über den “ATE”-Befehl aktiviert werden.

Write-Command	Antwort
ATE<n>	OK
Feld	Beschreibung
<n>	Aktivierungszustand der Remote-Echo Funktion. 0: ausgeschaltet, 1: eingeschaltet

### AT+CTEMP: Temperaturmesswerte

Die aktuellen Temperaturmesswerte können mithilfe des “AT+CTEMP”-Befehls ausgelesen werden.

Read-Command	Antwort
AT+CTEMP ?	+CTEMP: <bme>, <cpu>, <qmc>, <mpu> OK
Feld	Beschreibung
<bme>	Temperatur vom BME280 in 0.01 °C
<cpu>	CPU-Temperatur in 0.01 °C
<qmc>	Temperatur vom QMC5883L in 0.01 °C
<mpu>	Temperatur vom MPU6050 in 0.01 °C

### AT+CPRES: Luftdruck

Über den “AT+CPRES”-Befehl kann der Luftdruck-Messwert vom BME280 ausgelesen werden.

Read-Command	Antwort
AT+CPRES ?	+CPRES: <bme> OK
Feld	Beschreibung
<bme>	Luftdruck-Messwert vom BME280 in 1 Pa

### AT+CHUM: Relative Luftfeuchtigkeit

Die durch den BME280 gemessene, relative Luftfeuchtigkeit kann über den “AT+CHUM”-Befehl ausgelesen werden.

Read-Command	Antwort
AT+CHUM?	+CHUM: <bme> OK

Feld	Beschreibung
<bme>	Relative Luftfeuchtigkeit in 0.01 %

### AT+CWIND: Windrichtung und -geschwindigkeit

Die Windgeschwindigkeit und die Windrichtung werden über den Befehl “AT+CWIND” abgefragt.

Read-Command	Antwort
AT+CWIND?	+CWIND: <dir>, <spd> OK

Feld	Beschreibung
<dir>	Windrichtung in 0.1 °
<spd>	Windgeschwindigkeit in 1 m/s

### AT+CTIME: Echtzeituhr

Über den “AT+CTIME”-Befehl können Datum und Uhrzeit der internen Echtzeituhr ausgelesen und überschrieben werden.

Read-Command	Antwort
AT+CTEMP ?	+CTIME: <YY>, <MM>, <DD>, <hh>, <mm>, <ss> OK

Write-Command	Antwort
AT+CTIME=<YY>, <MM>, <DD>, <hh>, <mm>, <ss>	OK

Feld	Beschreibung
<YY>	Jahr 0 bis 99
<MM>	Monat 1 bis 12
<DD>	Tag 1 bis 31
<hh>	Stunde 0 bis 23 (UTC)
<mm>	Minute 0 bis 59
<ss>	Sekunde 0 bis 59

**AT+CALIGN: Ausrichtung des Panels** Die aktuelle Ausrichtung des Solarpanels, wie sie über den Kompass und den Neigungssensor ermittelt wurde, kann über den “AT+CALIGN”-Befehl ausgelesen werden.

Read-Command	Antwort
AT+CALIGN?	+CALIGN: <azm>, <zen> OK

Feld	Beschreibung
<azm>	Azimuth in 0.1°
<zen>	Zenit in 0.1°

### AT+CGNSPOS: GPS-Position

Nachdem ein gültiges Fix vom GPS-Modul erhalten wurde, kann die Position über den “AT+CGNSPOS”-Befehl ausgelesen werden. Sollte kein Fix gefunden werden können (bspw. innerhalb von Gebäuden), kann über diesen Befehl eine Position manuell vorgegeben werden.

<b>Read-Command</b>	<b>Antwort</b>
AT+CGNSPOS?	+CGNSPOS: <lat>,<lon>,<alt> OK

<b>Write-Command</b>	<b>Antwort</b>
AT+CGNSPOS=<lat>,<lon>[,<alt>]	OK

<b>Feld</b>	<b>Beschreibung</b>
<lat>	Breitengrad 0.0001 ° N (S durch negatives Vorzeichen)
<lon>	Längengrad in 0.0001 ° E (W durch negatives Vorzeichen)
<alt>	(optional) Höhe über MSL in 0.1 m

**AT+CPWR: Strom- und Spannungsmesswerte** Die gemessenen Ströme und Spannungen an dem Akku und der Solarzelle können über den “AT+CPWR”-Befehl ausgelesen werden.

<b>Read-Command</b>	<b>Antwort</b>
AT+CPWR?	+CPWR: <v_bat>,<i_bat>,<v_solar>,<i_solar>,<v_sys> OK

<b>Feld</b>	<b>Beschreibung</b>
<v_bat>	Akkuspannung in 1 mV
<i_bat>	Ladestrom in 1 mA
<v_solar>	Panelspannung in 1 mV
<i_solar>	Panelstrom in 1 mA
<v_sys>	Systemspannung in 1 mV, konstant 3.3 V

### AT+CINTV: Messintervall

Die Periodendauer, mit welcher die Sensoren periodisch ausgewertet werden, kann über den “AT+CINTV”-Befehl angepasst werden.

Write-Command	Antwort
AT+CINTV=<int>	OK

Feld	Beschreibung
<int>	Messintervall in 1s

### AT+CGUI: Daten für die grafische Benutzeroberfläche

Die grafische Benutzeroberfläche ruft die letzten gespeicherten Messwerte über den “AT+CGUI”-Befehl ab.

Read-Command	Antwort
AT+CGUI?	+CGUI: <YY>, <MM>, <DD>, <hh>, <mm>, <ss>, <t_bme>, <t_cpu>, <t_qmc>, <t_mpu>, <w_dir>, <w_spd>, <pres>, <hum>, <zen>, <azm>, <lat>, <lon>, <alt>, <v_bat>, <i_bat>, <v_solar>, <i_solar>, <v_sys> +CGUI: ... OK

Feld	Beschreibung
<YY>	Jahr 0 bis 99
<MM>	Monat 1 bis 12
<DD>	Tag 1 bis 31
<hh>	Stunde 0 bis 23 (UTC)
<mm>	Minute 0 bis 59
<ss>	Sekunde 0 bis 59
<t_bme>	Temperatur vom BME280 in 0.01 °C
<t_cpu>	CPU-Temperatur in 0.01 °C
<t_qmc>	Temperatur vom QMC5883L in 0.01 °C
<t_mpu>	Temperatur vom MPU6050 in 0.01 °C
<w_dir>	Windrichtung in 0.1 °
<w_spd>	Windgeschwindigkeit in 1 m/s
<pres>	Luftdruck-Messwert vom BME280 in 1 Pa
<hum>	Relative Luftfeuchtigkeit in 0.01 %
<zen>	Zenit in 0.1 °
<azm>	Azimuth in 0.1 °
<lat>	Breitengrad 0.0001 ° N (S durch negatives Vorzeichen)
<lon>	Längengrad in 0.0001 ° E (W durch negatives Vorzeichen)
<alt>	(optional) Höhe über MSL in 0.1 m
<v_bat>	Akkuspannung in 1 mV
<i_bat>	Ladestrom in 1 mA
<v_solar>	Panelspannung in 1 mV
<i_solar>	Panelstrom in 1 mA
<v_sys>	Systemspannung in 1 mV, konstant 3.3 V

### **AT+CWKUP: Wakeup-Task manuell Ausführen**

Der Wakeup-Task kann durch Aufruf des “AT+CWKUP”-Befehls manuell ausgelöst werden.

Execute-Command	Antwort
AT+CWKUP	OK

### **AT+CTRACK: Nachführung des Panels**

Die Nachführung des Solarpanels kann über den “AT+CTRACK”-Befehl aktiviert werden. Sie ist beim Systemstart aus Sicherheitsgründen deaktiviert.

Read-Command	Antwort
AT+CTRACK?	+CTRACK: <stat> OK

Write-Command	Antwort
AT+CTRACK=<stat>	OK

Feld	Beschreibung
<stat>	Aktivierungszustand der Nachführung. 0: ausgeschaltet, 1: eingeschaltet

### **AT+CTURN: Manuelle Turm-Ausrichtung**

Der Turm kann manuell mithilfe des “AT+CTURN”-Befehls zeitgesteuert in eine vorgegebene Richtung gedreht werden. Außerdem wird über diesen Befehl die Kalibrierung des Magnetometers QMC5883L gestartet.

Write-Command	Antwort
AT+CTURN=<dir>	OK

Feld	Beschreibung
<dir>	Richtung des Turms relativ zur Startposition in 0.1 °. 'C' zum Start der Kalibrierung

### 5.3.2 Energiesparmaßnahmen

Das Bluetooth-Modul hat im Sendebetrieb einen hohen Energieverbrauch (etwa 500 mW). Es wird daher nach einer längeren Zeit ohne aktive Verbindung über einen Leistungs-MOSFET abgeschaltet.

Damit dennoch ohne physikalischen Zugriff eine Verbindung mit der Wetterstation aufgebaut werden kann, wird das Bluetooth-Modul in regelmäßigen Zeitabständen für etwa 30 s im “Advertise”-Modus eingeschaltet. In dieser Zeit ist das Modul für andere Bluetooth-Endgeräte in der Umgebung als “Wetterstation\_1a” sichtbar.

Alternativ kann das Bluetooth-Modul durch Drücken des blauen Tasters an der Hauptplatine für länger als 1 s manuell aktiviert werden.

### 5.3.3 Pairing-Daten

Bei der erstmaligen Verbindung eines PCs mit der Wetterstation muss zunächst ein *Pairing* zwischen dem Bluetooth-Modul und dem Rechner erfolgen. Hierbei muss am PC ein vierstelliger Pin-Code eingegeben werden, um die Verbindung zu aktivieren: **1234**

## 5.4 Messdatenprotokoll auf einer SD-Karte

Die Messdaten werden bei jeder Aufzeichnung zusätzlich auf der SD-Karte in der Textdatei “LOG.TXT” im CSV-Format abgespeichert. Jede Zeile ist dabei nach folgendem Schema zusammengestellt:

```
<YYYY>-<MM>-<DD>T<hh>:<mm>:<ss>Z,<t_bme>,<t_cpu>,<t_qmc>,<t_mpu>,<pres>,<hum>,<w_dir>,<w_spd>,<azm>,<zen>,<lat>,<lon>,<alt>,<v_bat>,<i_bat>,<v_solar>,<i_solar>
```

Die Formatierung der einzelnen Felder ist identisch zur Ausgabe des AT-Befehls “AT+CGUI”, welche zuvor auf Seite 41 dokumentiert wurde.

## 5.5 Motorsteuerung

Die Motorsteuerung nutzt einen schnellen Timer-Interrupt mit einer Periode von 100 ms zur Regelung der Panelausrichtung. Die Fahrt erfolgt dabei zeitgesteuert, und wird nach Erreichen der Sollposition mithilfe des Magnetometers QMC5883L für Abweichungen kompensiert.

Bei gleichzeitigen Sollwertänderungen für Panel- und Turmausrichtung wird zuerst das Panel verfahren. Der Turm wird erst nach Erreichen der Sollposition für die Panel-Achse verfahren, um übermäßig hohe Stromspitzen zu vermeiden.

### 5.5.1 Referenzfahrt für die Panelausrichtung

Nach der Initialisierung führt die Motorsteuerung eine Referenzfahrt auf der Panel-Achse aus, um einen Referenzwert für die Ausrichtung des Solarpanels entlang der Querachse zu erhalten. Das Panel wird dazu so lange nach unten gefahren, bis der Endlagenschalter auslöst.

### 5.5.2 Sicherheitsfunktion

Die Sicherheitsfunktion kann über den “ST”-Schaltereingang ausgelöst werden. Hierdurch werden alle Treibersignale ausgeschaltet und die Motorsteuerung gegen Wiedereinschalten gesperrt. Dieser Zustand kann nur durch einen Neustart des Systems zurückgesetzt werden.

## 5.6 Statusanzeige über LEDs

Die Wetterstation verfügt über drei verschiedenfarbige Leuchtdioden, welche am Gehäuse der Hauptplatine montiert sind. Hierüber werden einige wichtige Systemzustände angezeigt. Die Bedeutung der Blink-Pattern wird nachfolgend kurz beschrieben. Die Pattern werden alle 1.5 s wiederholt.

### 5.6.1 Grüne LED: Systemstatus

Die grüne LED zeigt den Systemzustand der Wetterstation an.

Blink-Pattern	Zustand
1x kurzes Blinken	Energiesparmodus aktiv
2x kurzes, 1x langes Blinken	Wakeup-Task wurde ausgeführt
Blinken ohne Pause	Fehler beim Systemstart

### 5.6.2 Blaue LED: Bluetooth

Der Aktivierungszustand des Bluetooth-Moduls wird über die blaue LED signalisiert.

Blink-Pattern	Zustand
ausgeschaltet	Modul ausgeschaltet.
2x kurzes Blinken	“Advertise”-Modus. Nicht verbunden.
dauerhaft ein	Verbindung aktiv.

### 5.6.3 Gelbe LED: Motorsteuerung

Die gelbe LED zeigt den aktuellen Zustand der Motorsteuerung an.

Blink-Pattern	Zustand
ausgeschaltet	Motorsteuerung deaktiviert.
1x kurzes, 1x langes Blinken	Referenzfahrt angefordert.
2x kurzes Blinken	Referenzfahrt beendet.
1x kurzes Blinken	Motorsteuerung aktiv.
Blinken ohne Pause	Sicherheitsfunktion ausgelöst.

## 6 Ausrichtung des Solarpanels

Um die Leistungsaufnahme des Solarpanels zu optimieren ist es notwendig dieses direkt auf die Sonne auszurichten und diese Ausrichtung auch in geeigneten Zeitabständen zu korrigieren. Im Vergleich mit einem fest ausgerichteten Solarpanel konnten J. Rizek *et al.* mit einem nachgeführten Solarpanel beispielsweise die Leistungsaufnahme um durchschnittlich 30% erhöhen [13].

Hierfür kommen grundsätzlich verschiedene Methoden in Frage. In diesen Fall soll die Position der Sonne relativ zur Wetterstation auf Grundlage des Längen- und Breitengrades, der Uhrzeit und des Kalendertages berechnet werden. Diese Information werden über das GPS-Modul bereitgestellt. Anschließend wird das Solarpanel mit Hilfe der Motoren, des Kompass-Moduls und des am Panel befestigten Neigungssensors auf die Sonne ausgerichtet.

### 6.1 Berechnung der Sonnenposition

Da die Formeln zur Berechnung der Sonnenposition in diesem Projekt lediglich benutzt werden, wird an dieser Stelle auf eine Herleitung verzichtet. Die verwendeten Formeln können beispielsweise im *Astronomical Almanac* [1] gefunden werden. M. L. Roderick beschreibt die nötigen Berechnungen in seinem Report [16] und liefert zudem compilierbaren C-Code. Dieser wird im folgenden zur Berechnung der Sonnenposition verwendet.

Die Position der Sonne, im Bezug auf einen Beobachter auf der Erde, lässt sich durch die Werte *Zenith* und *Azimut* eindeutig beschreiben. Der Zenith beschreibt den Winkel zwischen einer Linie vom Beobachter zur Sonne und der Vertikalen. Der Azimut den Winkel zwischen der Horizontalen und Norden. Dabei stehen beispielsweise ein Azimut von 90° für Osten, 180° für Süden und 270° für Westen. Eine Veranschaulichung kann in Abbildung 16 gefunden werden.

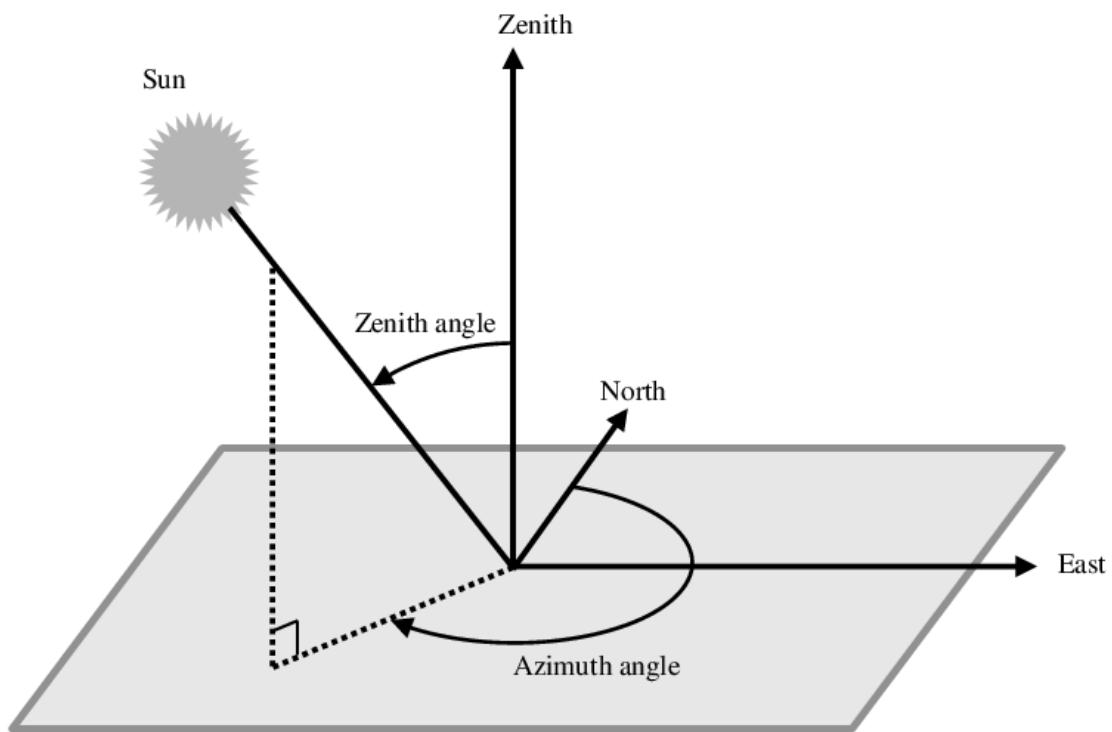


Abbildung 16: Beschreibung der Sonnenposition durch Zenith und Azimut [14]

Die Genauigkeit der verwendeten Formeln beträgt laut dem *Astronomical Almanac*  $0.01^\circ$  bezogen die auf die Position und  $0.1\text{ min}$  bezogen auf die Zeit.

## 7 Benutzeroberfläche

Um eine leichte Handhabung der Wetterstation zu ermöglichen wird eine, auf dem PyQt5 [20] Framework basierende, Benutzeroberfläche verwendet. Für die Kommunikation mit der Wetterstation muss der Computer über eine Bluetooth-Schnittstelle verfügen

In diesem Kapitel werden die zur Verfügung stehenden Funktionen sowie die Entwicklung der Benutzeroberfläche beschrieben.

### 7.1 Funktionen

Die Benutzeroberfläche (s. Abb. 17) verfügt über zwei Hauptansichten: eine graphische (s. Abb. 18) und eine tabelarische (s. Abb. 19).

Diese unterscheiden sich jeweils nur durch die Anzeigeart (s. Abb. 17 und 19 (1)).

Über das Konsolenfenster (s. Abb. 18 (2)) können die im Kapitel 5.3 beschrieben AT-Befehle an die Wetterstation gesendet werden. Sowohl die gesendeten als auch die empfangenen Daten werden im Konsolenfenster angezeigt. Über die Schaltfläche *Clear* lässt sich der Inhalt des Konsolenfensters löschen. Die Größe des Konsolenfensters und des Anzeigefensters kann beliebig verändert werden.

Das Menü (s. Abb. 18 (3)) enthält die zur Bedienung notwendigen Befehle. Es sind nicht alle AT-Befehle implementiert. Die AT-Befehle welche nicht implementiert sind, können manuell über das Konsolenfenster an die Wetterstation gesendet werden. Einige der Menübefehle wurden aufgrund von Zeitmangel nicht implementiert und sind daher ausgegraut. Das Menü enthält die folgenden Befehle:

- File
  - Save (nicht implementiert): Speichert die empfangenen Daten in einer CSV-Datei ab.
  - Open (nicht implementiert): Lädt die Daten aus einer CSV-Datei in das Programm.
- Control
  - Set Time

- \* UTC: Setzt die Uhrzeit und das Datum der Wetterstation auf die Koordinierte Weltzeit. Die Uhrzeit wird der Computeruhr entnommen.
- \* Custom (nicht implementiert): Öffnet ein Fenster in welches eine beliebige Uhrzeit und Datum eingegeben werden kann. Diese wird anschließend auf die Wetterstaion gespielt.
- Set Position
  - \* Hamburg: Setzt die Postition der Wetterstation auf (53.556354, 10.022650) (HAW Hamburg).
  - \* Custom (nicht implementiert): Öffnet ein Fenster in welches beliebige Koordinaten eingegeben werden können. Diese werden anschließend auf die Wetterstation übertragen.
- Adjust Orientation (nicht implementiert): Gibt der Wetterstation den Befehl sich sofort neu auszurichten.
- Set Update-Intervall: Bestimmt das Intervall, in welchem eine Verbindung mit der Wetterstaion aufgebaut wird. Nach dem Aufbau der Verbindung werden die neuen Messdaten angefordert, empfangen und dargestellt. Anschließend wird die Verbindung, um Energie zu sparen, wieder geschlossen.
  - \* 5 sec: fünf Sekunden
  - \* 1 min: eine Minute
  - \* 15 min: fünfzehn Minuten
  - \* 1 h: eine Stunde
  - \* Manuel: Kein automatisches Anfordern der Messdaten. Zum Anfordern der Messdaten muss die *Update*-Schaltfläche betätigt werden.
- Set Measuring-Intervall: Schickt einen Befehl an die Wetterstation, welcher vorgibt in welchem Intervall Messungen durchzuführen sind.
  - \* 5 sec: fünf Sekunden
  - \* 15 sec: fünfzehn Sekunden
  - \* 1 min: eine Minute

- View: Bestimmt das Zeitintervall, welches in der graphischen Anzeige dargestellt wird.
  - \* Last Minute: letzte Minute
  - \* Last 15 Minutes: letzten fünfzehn Minuten
  - \* Last Hour: letzte Stunde
  - \* Last Day: letzter Tag
  - \* Last Week: letzte Woche
  - \* Last Month: letzter Monat
  - \* Custom (nicht implementiert): Der dargestellte Bereich wird über die Elemente zur Darstellung der Start- und Stopzeit (s. Abb. 18 (4)) eingestellt. Der Bereich wird im Gegensatz zu den anderen Optionen nicht automatisch mit voranschreiten der Zeit geupdatet.
- Mode:
  - Enable Debug: Aktiviert den Debug-Modus. Debugnachrichten werden angezeigt.
  - Disable Debug: Deaktiviert den Debug-Modus. Debugnachrichten werden nicht angezeigt.
  - Enable Com: Baut manuell eine Bluetooth-Verbindung zur Wetterstation auf. Dies ist notwendig, wenn Befehle zur Wetterstation gesendet werden sollen. Für das automatische, periodische Einlesen der Messdaten ist dies nicht notwendig. Es gibt momentan keine Möglichkeit den für die Verbindung verwendeten Com-Port über die Benutzeroberfläche anzupassen. Stattdessen muss dieser manuell in der Datei *main\_ctrl.py* geändert werden.
  - Disbale Com: Schließt die Bluetooth-Verbindung zur Wetterstation.

Die Anzeigen (4) (s. Abb. 18) zeigen, wie bereits erwähnt, das Intervall an welches dargestellt wird. Ist dieses Intervall nicht auf *Custom* gestellt, so ist die Endzeit immer die aktuelle Zeit. Die Startzeit ergibt entsprechend des eingestellten Zeitintervalls.

Über die Schaltflächen (5) (s. Abb. 18) können zwei Messkurven ausgewählt werden, welche dargestellt werden sollen. Die Schaltfläche *Update* ist für das manuelle Anfordern

## 7 Benutzeroberfläche

---

der Messdaten notwendig. Unter den Schaltflächen werden die, über das Fadenkreuz (6) (s. Abb. 18) ausgewählten, Messpunkte schriftlich dargestellt.

Im aktuellen Softwarestand wird die Kommunikation mit der Wetterstation nicht in einem separaten Task ausgeführt. Dies führt zu Problemen in der Bedienbarkeit der Anwendung. Während das Programm Daten mit der Wetterstation austauscht kann nicht, durch den Nutzer ausgelöste, Ereignisse reagiert werden. Um den Bedienkomfort zu erhöhen sollte die Kommunikation in einen eigenen Task ausgelagert werden. Dies konnte aufgrund der zeitlichen Beschränkung des Projekts nicht realisiert werden.

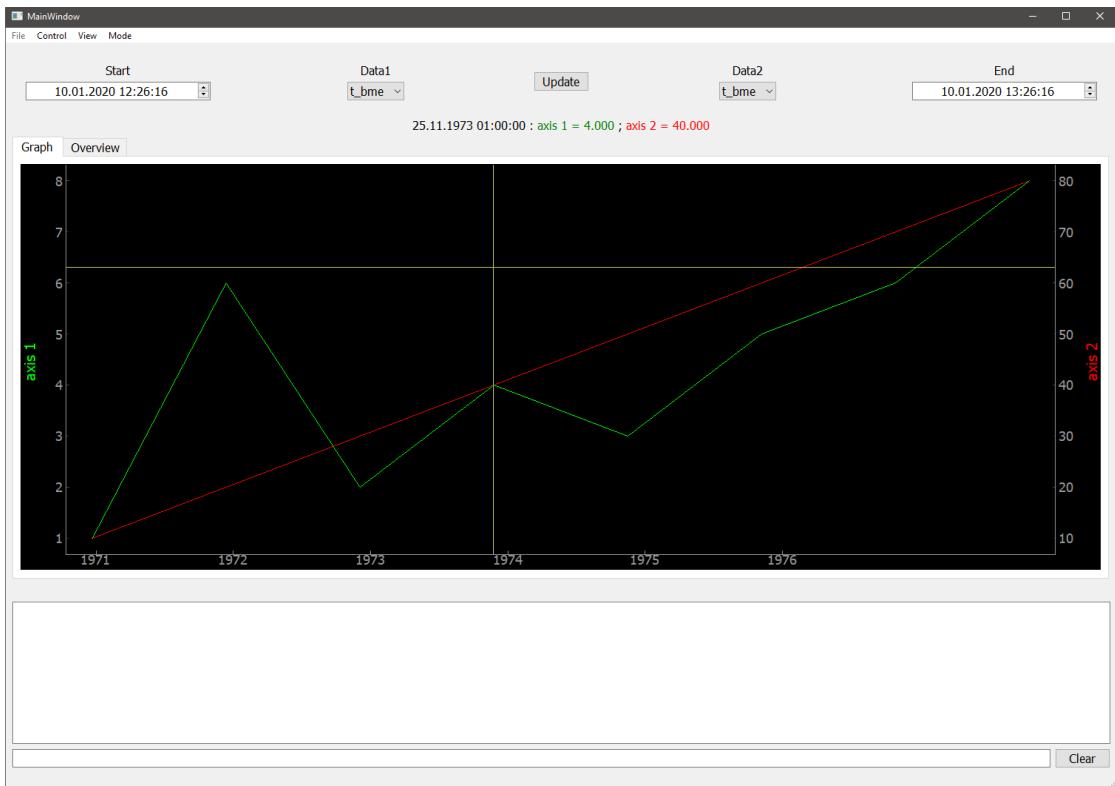


Abbildung 17: Benutzeroberfläche nach dem Öffnen. Dargestellt sind zwei vordefinierte Testkurven, welche nach Empfang des ersten Messwertes gelöscht werden.

## 7 Benutzeroberfläche



Abbildung 18: Benutzeroberfläche: (1) Graphische Darstellung der Messwerte. (2) Konsolefenster. (3) Kontrollmenü mit den wichtigsten Befehlen. (4) Start- und Endzeit der graphischen Darstellung. (5) Auswahl der dargestellten Messwerte, Schaltfläche zum manuellen Update der Messwerte und Anzeige der Messwerte an der aktuellen Fadenkreuzposition. (6) Fadenkreuz zum Anzeigen spezifischer Messwerte.

## 7 Benutzeroberfläche

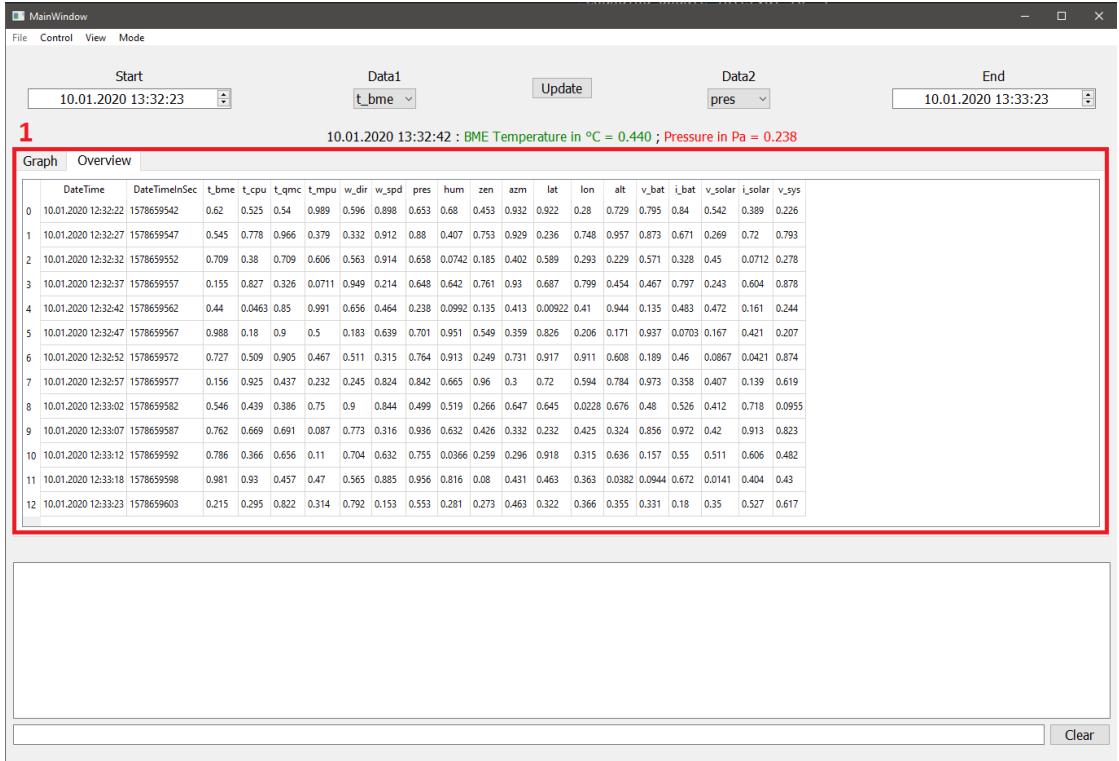


Abbildung 19: Benutzeroberfläche: (1) Tabellarische Darstellung von simulierten Werten.

## 7.2 Entwicklung

In diesem Kapitel werden die, für die Benutzeroberfläche implementierten Funktionen, *nicht* im Detail beschrieben. Es soll lediglich ein Überblick über die verwendeten Technologien und den Aufbau der Projekts gegeben werden. Bei der Entwicklung wurde versucht den Code in einer Art und Weise zu schreiben, welcher es anderen Personen ermöglicht, ohne eine ausführliche Beschreibung des gesamten Codes, an diesem weiterzuarbeiten.

Als Programmiersprache wurde Python 3 verwendet. Der verwendete Quellcode befindet sich im Ordner UI. Die Oberfläche wurde nach dem Model-View-Controller Entwurfsmuster (s. z.B. [11], s. Abb. 20) entwickelt. Für einige der Funktionalitäten wurde Code aus den, im PyQt5-Package enthaltenen, Beispielen als Vorlage verwendet.

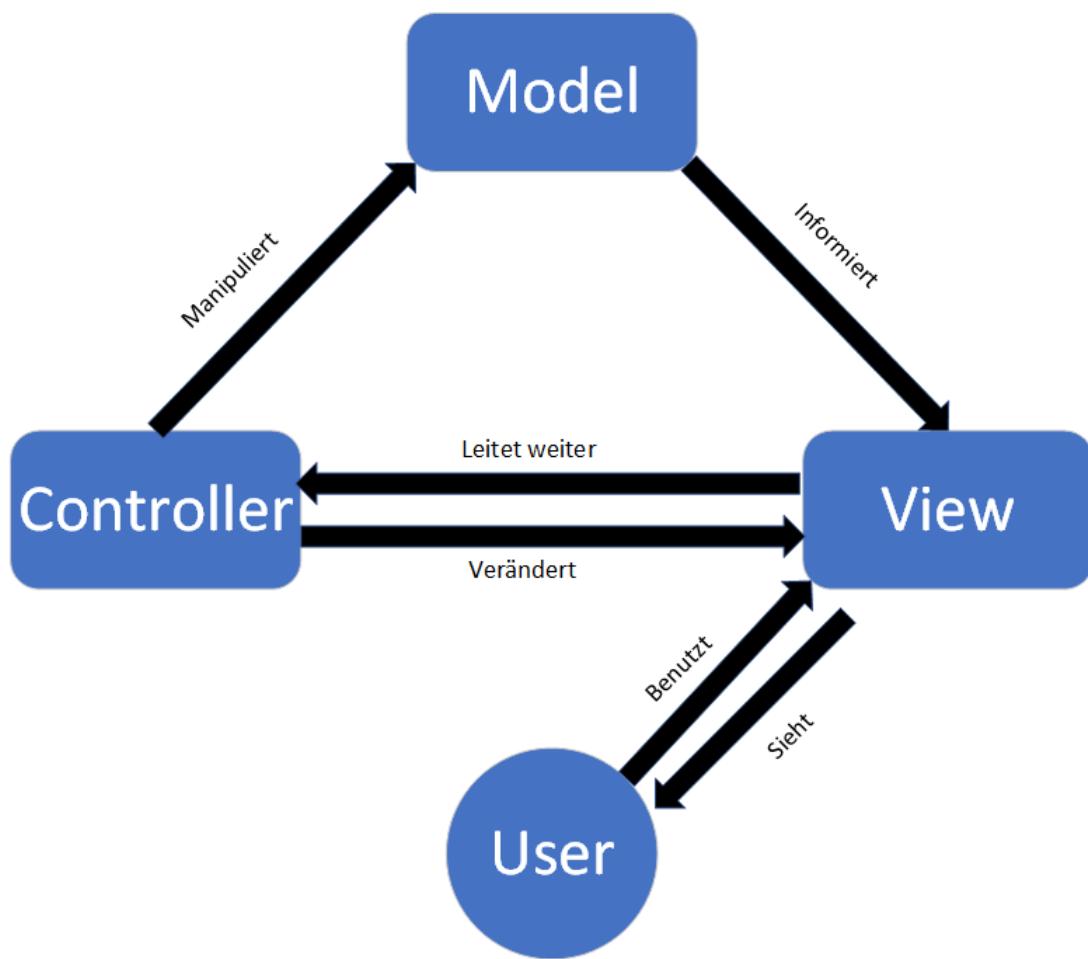
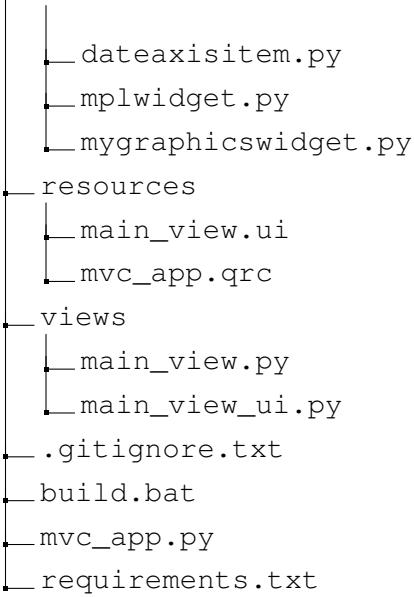


Abbildung 20: Model-View-Controller Entwurfsmuster, symbolische Darstellung

### Projektstruktur

Das Projekt ist wie folgt strukturiert:

```
UI
├── .git
└── controllers
    └── main_ctrl.py
├── model
    └── model.py
└── mywidgets
    └── dateaxis.py
```



Die Hauptdatei des Projekts ist *mvc\_app.py*. In ihr werden das Model, der Controller und das View initialisiert und verbunden. Das Model (*model.py*) beinhaltet alle Datenstrukturen, also sowohl die Messwerte als auch Variablen, welche den Programmablauf steuern. Das View (*main\_view.py* und *main\_view.ui.py*) beinhaltet die Benutzeroberfläche und bestimmt welche Funktionen beim Aktivieren welcher Schaltflächen aufgerufen werden. Diese Funktionen befinden sich im Controller (*main\_ctrl.py*). Im Ordner *mywidgets* befinden sich die im View implementierten, benutzerdefinierten graphischen Anzeigen.

Die Verbindung zwischen Model, View und Controller ist über *Signale* und *Slots* realisiert.

Die Datei *requirements.txt* beinhaltet eine Liste aller benötigten Python-Packages. Mit dem Befehl „`pip install -r requirements.txt`“ können diese automatisch installiert werden.

Im Ordner *resources* befinden die mit *PyQt5-Designer* erstellten UI-Dateien.

### Erstellung der Oberfläche

Zum Erstellen der Oberfläche wurde das graphische Tool *PyQt5-Designer* (s. Abb. 21) verwendet. Das Übersetzen der mit diesem Tool erstellten *.ui*- und *.qrc*-Dateien in ausführbaren Python-Code erfolgt über das Tool *pyuic5*. Zum Automatisieren dieses Pro-

## 7 Benutzeroberfläche

zesses wurde das Skript *build.bat* angelegt. Nach Bearbeitung der Oberfläche im *PyQt5-Designer* sollte dieses ausgeführt werden um die Änderungen zu übernehmen.

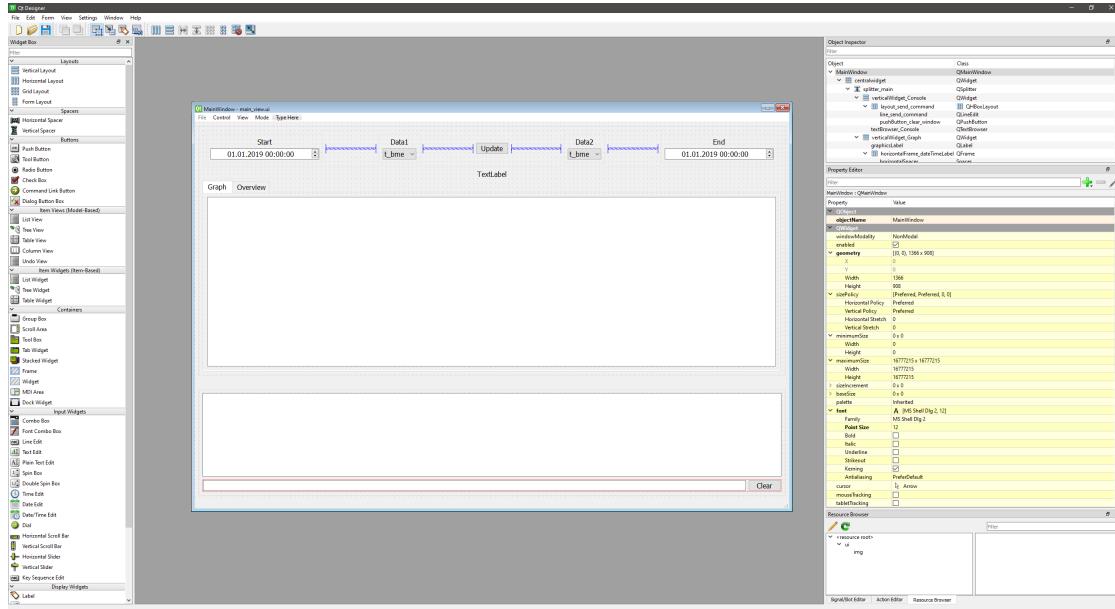


Abbildung 21: Erstellen der Oberfläche mit PyQt5-Designer.

## Verwendete Python-Packages

Es werden die folgenden Python-Packages Programm verwendet:

- PyQt5: Als Framework für die Oberfläche.
- pyqtgraph: Für die graphische Darstellung der Messdaten.
- serial: Für die serielle Kommunikation, über Bluetooth, mit der Wetterstation.
- pandas: Für die Strukturierung der Messdaten.
- numpy: Für das Erstellen von Testdaten.

## 8 3D gedruckte Komponenten

Für das Unterbringen des Mikroprozessors und der Sensoren werden zwei verschiedene Gehäusevarianten verwendet. Dieses Kapitel beschäftigt sich mit deren Entwurf, sowie dem Entwurf des Adapters, mit welchem die Windfahne und das Anemometer an der Wetterstation befestigt sind.

Für den Entwurf der Komponenten wurde die Studentenversion von Autocad Fusion 360 verwendet.

### 8.1 Hauptgehäuse

Das Hauptgehäuse beinhaltet das Mikroprozessor-Board mit den zwei aufsteckbaren Platinen sowie den Sensoren. Drei der Sensoren befinden sich nicht im Hauptgehäuse (s. Kap. 8.2).

Das Gehäuse besteht aus zwei Teilen (s. Abb. 22), dem Hauptteil (links) und dem Deckel (rechts). Das STM8 Nucleo Board wird von drei Pins in Position gehalten (s. Abb. 22, 23). In der Vorderseite befinden sich zwei Aussparungen (s. Abb. 24). Über die Rechte ist der USB-Port des Mikroprozessor-Boards erreichbar. Hinter der Linken befindet sich der Motortreiber, welcher auf einer leicht erhöhten Plattform plaziert wird (s. Abb. 23). Über die Aussparung an der rechten Seite kann die SD-Karte erreicht werden. Die Aussparung an der hinteren Seite wird verwendet, um alle externen Komponenten mit dem Mikroprozessor-Board zu verbinden.

Im Deckel befinden sich drei Löcher, in welche Status-LEDs untergebracht werden.

Das Hauptteil und der Deckel werden lediglich aufeinander gesteckt. Bedingt durch die Geometrie entsteht ein relativ fester Halt. Optional können die Komponenten über die vier Aussparungen an den Seiten, mit beispielsweise Kabelbindern, zusätzlich gesichert werden.

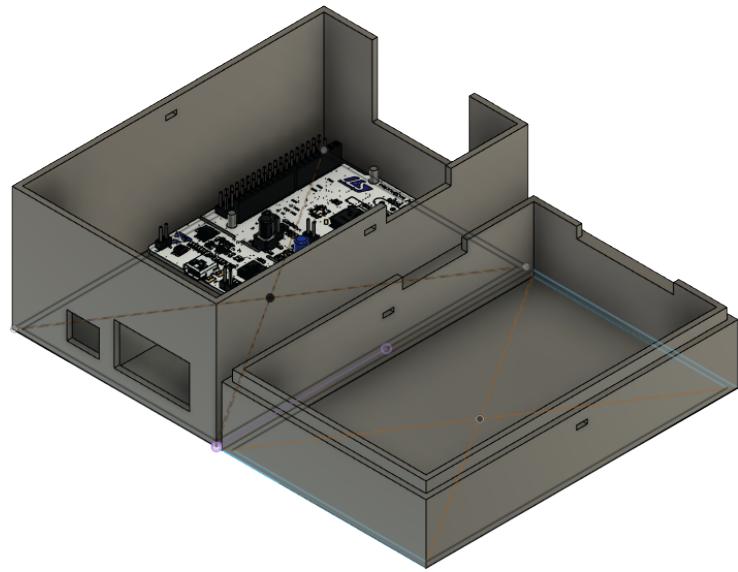


Abbildung 22: Hauptgehäuse

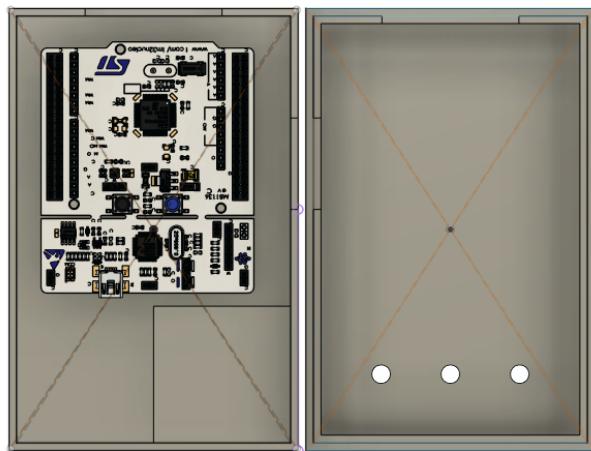


Abbildung 23: Hauptgehäuse, Ansicht von oben

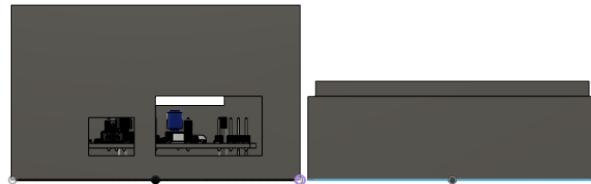


Abbildung 24: Hauptgehäuse, Ansicht von vorne

## 8.2 Nebengehäuse

Es wird jeweils ein Nebengehäuse für folgende Sensoren verwendet:

- der Neigungssensor, welcher an der Unterseite des Solarpanels befestigt werden muss
- der GPS-Empfänger, welcher für besseren Empfang am oberen Ende der Wetterstation befestigt wird
- das Kompass-Modul, welches empfindlich auf elektro-magnetische Störungen reagiert und aus diesem Grund auch am oberen Ende der Wetterstation untergebracht wird (auf der Seite gegenüber des GPS-Empfängers)

Die Nebengehäuse bestehen aus zwei Teilen, welche aufeinandergesteckt werden und zusätzlich, optional, mittels Kabelbinder gesichert werden (s. Abb. 25, 27).

Die Nebengehäuse werden mit einer Schraube und einer Mutter am Aluminium-Profil der Wetterstation befestigt (s. Abb. 26). Das Nebengehäuse, welches sich auf dem Solarpanel befindet, kann nicht geschraubt werden und wird entsprechend mit Klebstoff befestigt.

Die runden Aussparungen an der linken und rechten Seiten dienen dem Durchführen von benötigten Leitungen.

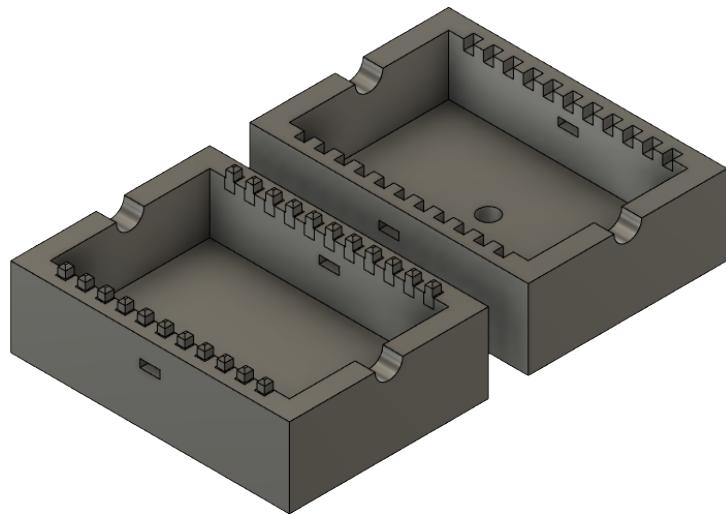


Abbildung 25: Nebengehäuse

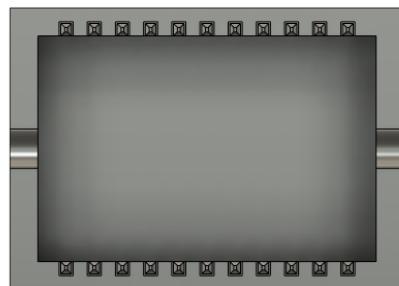
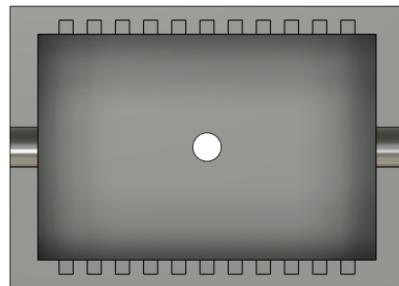


Abbildung 26: Nebengehäuse, Ansicht von oben

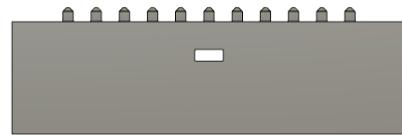


Abbildung 27: Nebengehäuse, Ansicht von vorne

### 8.3 Adapter

Der Adapter wird verwendet, um eine feste Verbindung zwischen dem Aluminium-Profil der Wetterstation und dem Mast mit Windfahne und Anemometer herzustellen.

Die Verbindung zur Wetterstation erfolgt durch Stecken der vier trapez-förmigen Steckkontakte am unteren Ende des Adapters (s. Abb. 28, 30).

Der Mast wird von oben auf die angepasste Aussparung gesteckt. Optional kann dieser mittels Schraubverbindung fixiert werden (s. Abb. 28, 29).

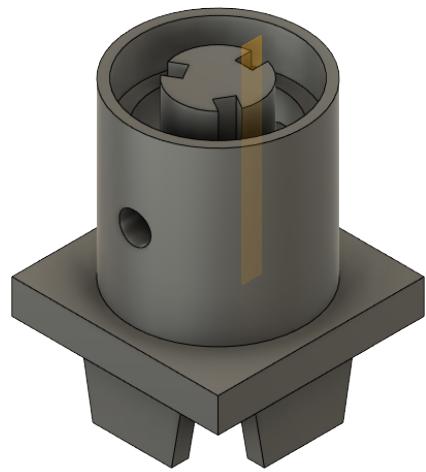


Abbildung 28: Adapter

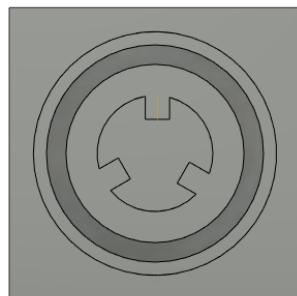


Abbildung 29: Adapter, Ansicht von oben

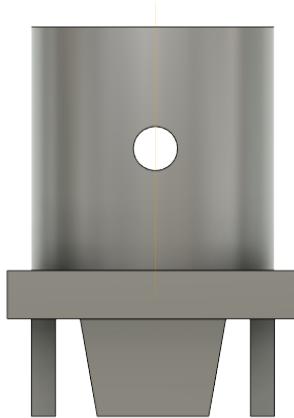


Abbildung 30: Adapter, Ansicht von vorne

#### 8.4 Herstellung

Alle Komponenten wurden in PLA mit einer Schichtdicke von 0.2 mm gedruckt. Es wurden zwei bis drei Top-, Bottom- und Wandschichten gedruckt. Da die Komponenten keine großen Kräfte aushalten müssen wurden sie lediglich mit einem Infill von 10 % bis 20 % gedruckt.

## 9 Fazit



Abbildung 31: Fertiggestellte Wetterstation

In Abbildung 31 wird der fertige Aufbau der Wetterstation dargestellt. Die Anforderung der Erfassung von Temperatur, Luftdruck und -feuchte wird mit dem BME280, der im Hauptgehäuse platziert ist, umgesetzt. Für die Erfassung von Windrichtung und -geschwindigkeit konnte ein neues Anemometer gefunden und implementiert werden.

Im Hauptgehäuse ließen sich Mikrocontroller, Platinen und Sensoren sinnvoll unterbringen (Abbildung 32). Die Wasserfestigkeit des Aufbaus konnte leider, mit den verfügbaren Mitteln, in der vorgegebenen Zeit, nicht erreicht werden. Das verwendete 3D-Druck-Material ist nicht für einen abgedichteten Aufbau geeignet.

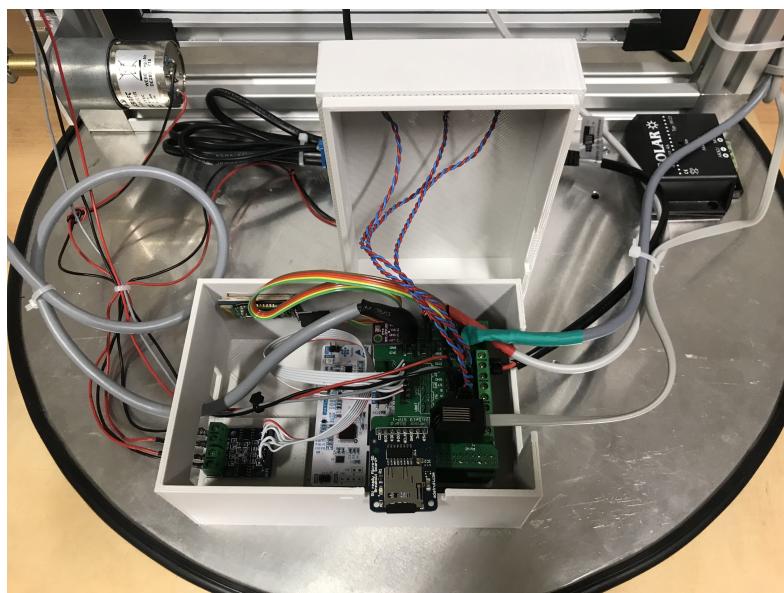


Abbildung 32: Blick in das Hauptgehäuse

Sowohl hardware- als auch softwareseitig wurden verschiedene Energiesparmaßnahmen implementiert. Die letztendliche Laufzeit müsste noch in einem Langzeittest ermittelt werden. Der verbaute Neigungssensor wird letztendlich nicht zur Kompensation bei der Ausrichtung des Panels benutzt, ist aber in diesem Fall auch nicht nötig, da die Funktionalität trotzdem gegeben ist. Die Aufhängung des Panels ließe sich in ihrer Mechanik noch verbessern, da sie in ihrer aktuellen Umsetzung relativ instabil ist, sodass das Panel leicht schief hängt. Diese Verbesserung war jedoch im vorgegebenen Zeitrahmen nicht möglich.

Grundlegend lässt sich sagen, dass alle gegebenen Anforderungen an das Projekt erfüllt wurden. Zusätzlich wurde eine Benutzeroberfläche zum Bedienen der Wetterstation und Anzeigen der Messwerte entwickelt.

## Literatur

- [1] ANON.: *The Astronomical Almanac*
- [2] ARGENT DATA SYSTEMS: *Weather Sensor Assembly p/n 80422.* : . – URL <https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly..pdf>. – Zugriff: 09.10.2019
- [3] ARGENT DATA SYSTEMS: *Weather Sensor Assembly p/n 80422.* : , unbekannt. – URL <https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly..pdf>. – Zugriff: 11.01.2020
- [4] ATMEL CORPORATION: *ATmega2560 8-bit Atmel Microcontroller with 64KB In-System Programmable Flash.* : , 2014. – URL [https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561\\_datasheet.pdf](https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf). – Zugriff: 11.01.2020
- [5] ATMEL CORPORATION: *Atmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash.* : , 2015. – URL [http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\\_Datasheet.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf). – Zugriff: 11.01.2020
- [6] ATMEL CORPORATION: *SAM3X/SAM3A Series SMART ARM based MCU.* : , 2015. – URL [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf). – Zugriff: 11.01.2020
- [7] BADDELEY, Glenn: *GPS - NMEA sentence information.* – URL <http://aprs.gids.nl/nmea/>. – Zugriff: 06.11.2019
- [8] BOSCH SENSORTEC: *BME280 sensor driver / sensor API.* – URL [https://github.com/BoschSensortec/BME280\\_driver](https://github.com/BoschSensortec/BME280_driver). – Zugriff: 28.10.2019
- [9] BOSCH SENSORTEC: *BME280 Combined humidity and pressure sensor.* : , 2018. – URL [https://www.bosch-sensortec.com/media/boschsensortec/downloads/environmental\\_sensors\\_2/humidity\\_sensors\\_1/bme280/bst-bme280-ds002.pdf](https://www.bosch-sensortec.com/media/boschsensortec/downloads/environmental_sensors_2/humidity_sensors_1/bme280/bst-bme280-ds002.pdf). – Zugriff: 01.10.2019

- [10] CHAN: *FatFs - Generic FAT Filesystem Module.* – URL [http://elm-chan.org/fsw/ff/00index\\_e.html](http://elm-chan.org/fsw/ff/00index_e.html). – Zugriff: 11.01.2020
- [11] DEACON, John: *Model-View-Controller (MVC) Architecture.* 2009. – URL <https://www.rareparts.com/pdf/MVC.pdf>. – Zugriff: 10.01.2020
- [12] INVENSENSE: *MPU-6000 and MPU-6050 Product Specification.* : , 2013.  
– URL <https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>. – Zugriff: 01.10.2019
- [13] J., Rizek ; Y., Chaiko: Solar Tracking System: More Efficient Use of Solar Panels. In: *World Academy of Science, Engineering and Technology* 41 (2008).  
– URL <http://www.mugla.edu.tr/data/03060010/belgeler/solar%20tracking.pdf>. – Zugriff: 17.10.2019
- [14] NOU, Julien ; CHAUVIN, Rémi ; THIL, Stéphane ; GRIEU, Stéphane: A new approach to the real-time assessment of the clear-sky direct normal irradiance. In: *Applied Mathematical Modelling* 40 (2016), 03
- [15] QST CORPORATION: *3-Axis Magnetic Sensor QMC5883L.* : , 2016. – URL [https://datasheet.lcsc.com/szlcsc/QST-QMC5883L-TR\\_C192585.pdf](https://datasheet.lcsc.com/szlcsc/QST-QMC5883L-TR_C192585.pdf). – Zugriff: 01.10.2019
- [16] RODERICK, M.L.: Methods for calculating solar position and day length including computer programs and subroutines. In: *Resource Management Technical Reports* (1992). – URL <https://researchlibrary.agric.wa.gov.au/cgi/viewcontent.cgi?article=1122&context=rmtr>. – Zugriff: 17.10.2019
- [17] ST MICROELECTRONICS: *STM8L152x6/8 8-bit ultra-low-power MCU.* : , 2018.  
– URL <https://www.st.com/resource/en/datasheet/stm8l152r8.pdf>. – Zugriff: 11.01.2020
- [18] ST MICROELECTRONICS: *STM32F446xC/E Arm® Cortex®-M4 32-bit MCU+FPU.* : , 2019. – URL <https://www.st.com/resource/en/datasheet/stm32f446re.pdf>. – Zugriff: 11.01.2020
- [19] ST MICROELECTRONICS: *UM1724, STM32 Nucleo-64 boards.* : , 2019.  
– URL [https://www.st.com/resource/en/user\\_manual/dm00105823.pdf](https://www.st.com/resource/en/user_manual/dm00105823.pdf). – Zugriff: 11.01.2020
- [20] TOMPSON, Phil: *Python bindings for the Qt cross platform application toolkit.* 2019.  
– URL <https://pypi.org/project/PyQt5/>. – Zugriff: 28.12.2019

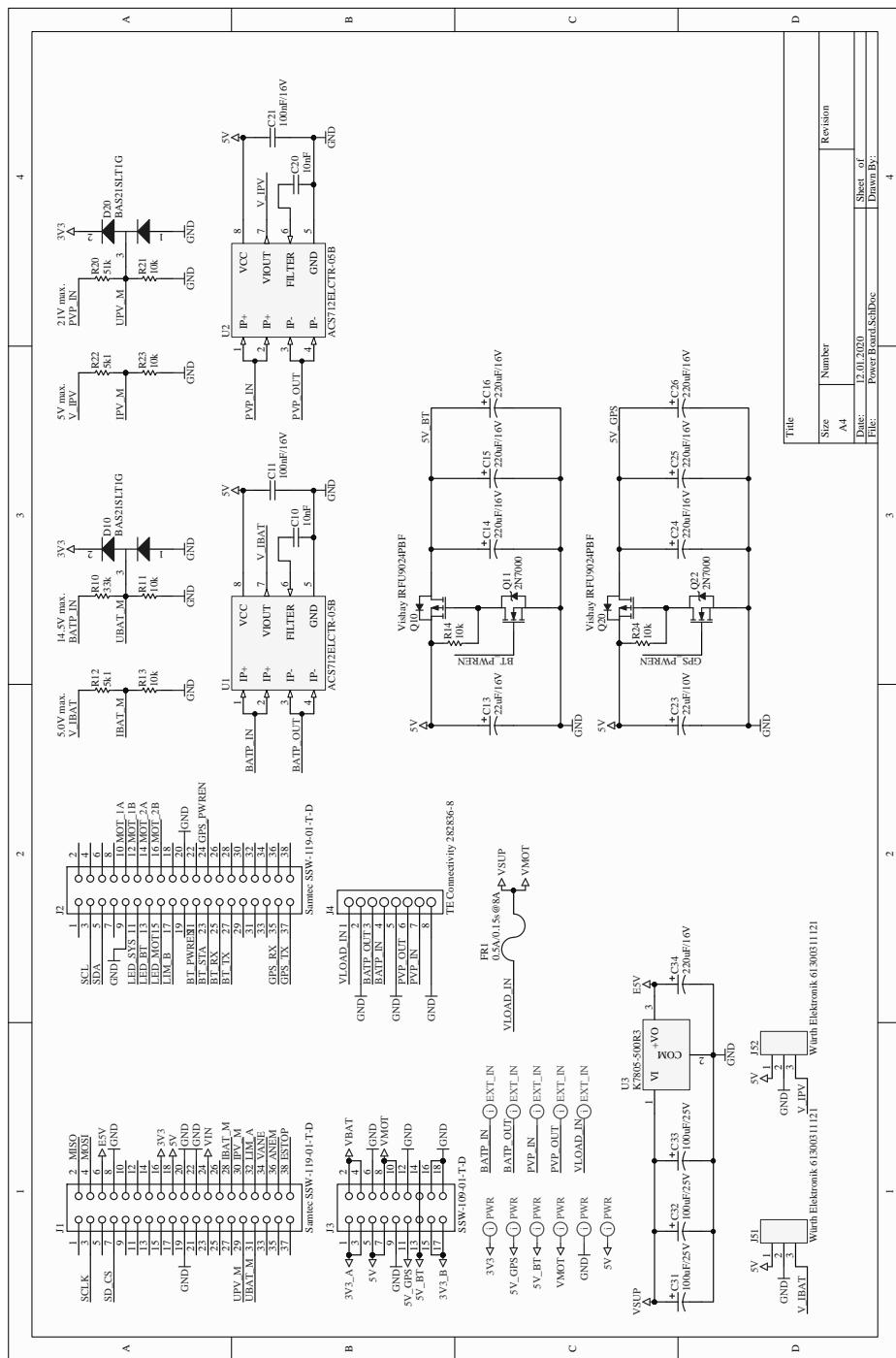
## *Literatur*

---

- [21] U-BLOX: *NEO-6 - Data Sheet.* : , 2011. – URL [https://www.u-blox.com/sites/default/files/products/documents/NEO-6\\_DataSheet\\_%28GPS.G6-HW-09005%29.pdf](https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf). – Zugriff: 11.01.2020

## A Stromlaufplan Power-Board

# A Stromlaufplan Power-Board



## B Stromlaufplan Sensor-Board

