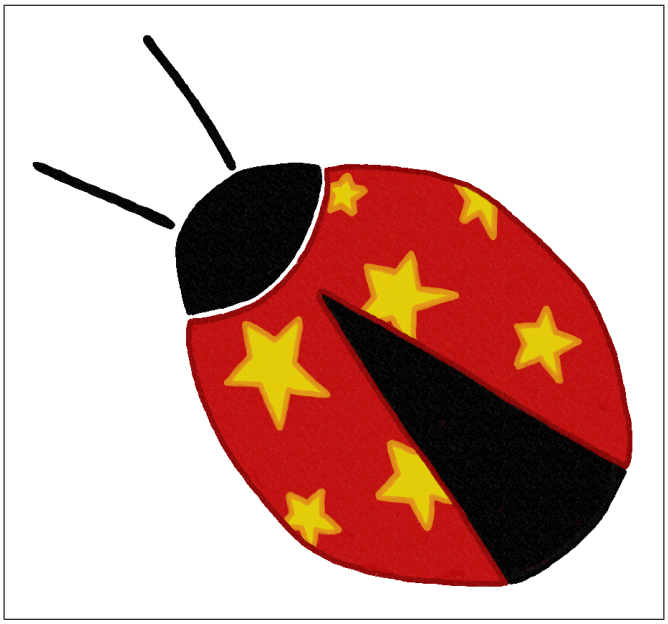
Deutscher CanSat-Wettbewerb 2017



Team Starbugs

Sönke Allen

Lennard Anders

Joscha Knobloch

Swantje Knüwer

Marius Kriworuschenko

Gianluca Müggenburg

Uwe Hülße

David Rowley

23. September 2017

Inhaltsverzeichnis

1 Einleitung **3**
- 1.1 Projekt . . . 3
- 1.2 Motivation . . . 3
- 1.3 Team . . . 4

2 Missionsziele **5**
- 2.1 Kommunikation zwischen Satelliten . . . 5
  - 2.1.1 Hintergrund . . . 5
  - 2.1.2 Umsetzung . . . 5
- 2.2 Fehlertoleranz . . . 6
- 2.3 Modularität . . . 6

3 Technische Beschreibung **7**
- 3.1 Mechanisches Design . . . 7
  - 3.1.1 Hülle . . . 7
  - 3.1.2 Fallschirm . . . 8
  - 3.1.3 Innerer Aufbau . . . 8
- 3.2 Elektrisches Design . . . 9
  - 3.2.1 Stromversorgung . . . 9
  - 3.2.2 Primärmission . . . 11
  - 3.2.3 Sekundärmission . . . 13
- 3.3 Softwaretechnisches Design . . . 14
  - 3.3.1 Primärmodul . . . 14
  - 3.3.2 Sekundärmodul . . . 14
- 3.4 Bodenstation . . . 14
  - 3.4.1 Empfänger . . . 15
  - 3.4.2 Datenbank . . . 15
  - 3.4.3 Datenverarbeitung . . . 15
  - 3.4.4 Receive Client . . . 16
  - 3.4.5 Transmit Client . . . 17
  - 3.4.6 CanSat Launcher . . . 18
  - 3.4.7 Analyse . . . 18

4 Projektplanung **19**
- 4.1 Entscheidungen . . . 19
  - 4.1.1 GPS . . . 19
  - 4.1.2 Befestigung der Platinen . . . 19
  - 4.1.3 Sekundärmodul . . . 19

4.1.4 Gehäuse . . . 20
4.2 Stärken und Schwächen des Teams . . . 20
4.2.1 Planung . . . 20
4.2.2 Treffen . . . 20
4.2.3 Kommunikation . . . 20
4.2.4 Zusammenarbeit . . . 21
4.2.5 Ziel . . . 21
4.2.6 Vorwissen . . . 21
4.2.7 Fazit . . . 21
4.2.8 Änderungen . . . 21
4.3 Aktueller Stand . . . 21
4.3.1 Hardware . . . 21
4.3.2 Software . . . 22
4.3.3 Hülle . . . 22
4.4 Unterstützung . . . 22

5 Kostenplanung **23**
5.1 Sponsoren . . . 23
5.1.1 b.r.m. . . . 23
5.1.2 Söffge . . . 23
5.1.3 Freiplan Ingenieure . . . 23
5.1.4 Bremer Rechenzentrum . . . 23
5.1.5 Watterott . . . 23
5.2 Ausgaben . . . 24

6 Öffentlichkeitsarbeit **27**
6.1 Website . . . 27
6.2 Facebook . . . 27
6.3 Instagram . . . 27
6.4 Tagebuch . . . 27
6.5 Maker Faire . . . 28

A Technische Anforderungen **30**

B Bilder **31**

C Gantt-Diagramm **45**

Kapitel 1

Einleitung

1.1 Projekt

Der Deutsche CanSat-Wettbewerb ist ein Wettbewerb, bei dem die zehn teilnehmenden Teams aus ganz Deutschland einen eigenen Satelliten in der Größe einer Getränkedose entwickeln. Wir haben uns mit unserer Mission, welche sich mit der Kommunikation zwischen Satelliten beschäftigt, beworben und wurden angenommen. Dies ist der abschließende Bericht, der unser Projekt zwei Wochen vor dem Start der Satelliten beschreibt.

1.2 Motivation

Wir waren für das fünfte Prüfungselement im Rahmen der Abiturprüfung auf der Suche nach einem interessanten Thema, als uns unser Lehrer den Deutschen CanSat-Wettbewerb vorstellte. Den Bau eines Satelliten fanden wir sehr interessant, da er verschiedene Bereiche, wie die Software-, die Hardwareentwicklung und die Physik miteinander verbindet. Außerdem hat uns die Möglichkeit eine eigene Mission zu entwickeln sehr fasziniert. Da man mit dem Gewinn des deutschlandweiten Wettbewerbs die Berechtigung erhält am europaweiten Wettbewerb teilzunehmen, haben wir unseren Englischlehrer Herrn Rowley dazu geholt, weil die Teilnahme am europaweiten Wettbewerb unser Ziel ist.

1.3 Team

Unser Team besteht aus sechs Schülern der Klasse DQI14 des Schulzentrum Utbremen. Für die Arbeit an diesem Projekt haben wir uns in einzelne Gruppen aufgeteilt. Diese Aufteilung mussten wir mehrfach überarbeiten. Aktuell ist sie wie folgt:

- Die **Hülle** unseres Satelliten wird von Lennard Anders entwickelt.
- Sönke Allen arbeitet an unserem **Fallschirm**.
- Für die Entwicklung der **Platinen** ist Joscha Knobloch verantwortlich.
- Die **Software des Satelliten** wird von Swantje Knüwer und Marius Kriworuschenko entwickelt.
- Die **Bodenstation** wird von Gianluca Müggenburg, Joscha Knobloch und Swantje Knüwer entwickelt.
- Für die **Öffentlichkeitsarbeit** ist Marius Kriworuschenko zuständig.
- Unsere **Finanzen** werden von Marius Kriworuschenko verwaltet.
- Die **Zusammenarbeit** wird von Swantje Knüwer organisiert.

Maik Jostes hat unser Team gegen Ende des Projekts verlassen. Er hat unsere Webseite aufgebaut. Gründe für die häufige Änderung der Teamaufteilung werden im Kapitel Projektplanung erläutert.

Kapitel 2

Missionsziele

Unsere Sekundärmission besteht aus drei Zielen. Wir wollen zum Einen die Kommunikation zwischen zwei Satelliten ermöglichen und dafür als ersten Schritt einen Satelliten am Boden simulieren, welcher Daten sendet, die wir mit unserem Satelliten auffangen und an unsere Bodenstation weiterleiten. Zum Anderen wollen wir zum Bau unseres Satelliten Verfahren verwenden, die zu einem möglichst fehlertoleranten und robusten System führen. Unser drittes Ziel ist es ein modulares System umzusetzen, in welchem wir das Sekundärmodul austauschen können, um den Satelliten so für andere Missionen anzupassen.

2.1 Kommunikation zwischen Satelliten

2.1.1 Hintergrund

In der Raumfahrt besteht das Problem, dass erdnahe Satelliten von einer Bodenstation auf der Erde nur für eine sehr kurze Zeit zu erreichen sind, da die Funkstrecke dann erneut durch die Erde blockiert wird. Um dieses Problem zu umgehen, werden geostationäre Satelliten neben ihrer normalen Funktion zur Bildung einer Funkbrücke zwischen dem erdnahen Satelliten und der Bodenstation eingesetzt.

2.1.2 Umsetzung

Anders als im Zwischenbericht beschrieben, werden wir die Daten nur von der Bodenstation zum Transceiver des Sekundärmoduls senden. Von dort werden die Daten an das Primärmodul übergeben und über dessen Transceiver wieder zurück zur Bodenstation gesendet. Der Satellit der zweiten Gruppe entfällt aus dieser Umsetzung, da bei Problemen des Satelliten der anderen Gruppe, ein Gelingen unserer Mission nicht möglich ist. Zudem ist eine redundante Umsetzung, in welcher Daten von Bodenstation und Kooperationsgruppe empfangen werden, aufgrund fehlender frei zugänglicher Frequenzbänder ebenfalls ausgeschlossen.

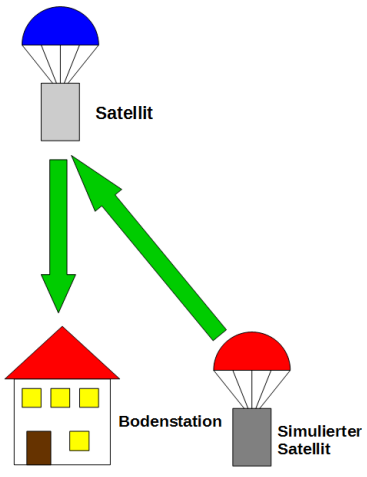


Abbildung 2.1: Umsetzung

2.2 Fehlertoleranz

Unser Satellit soll möglichst fehlertolerant sein, sodass bei einem Ausfall keine Daten verloren gehen. Unsere Idee ist es eine doppelte Stromversorgung einzusetzen, damit notfalls die zweite Stromversorgung die Erste ersetzen kann. Die zweite Stromversorgung wird im Sekundärmodul platziert. Eine weitere Sicherung ist das zweite SD-Karten Modul im Sekundärmodul. Dieses schreibt auch alle Daten mit, die es vom Primärmodul erhält.

2.3 Modularität

Unser Satellit wird aus zwei Modulen bestehen. Das Primärmodul enthält alle Bauteile, um die durch den Wettbewerb vorgegebenen Ziele zu erreichen, sowie weitere Bauteile, die wir für sehr wichtig erachten. Das Sekundärmodul enthält unsere Sekundärmission, sowie weitere Akkus, die eine höhere Betriebsdauer ermöglichen und eine weitere SD-Karte für eine doppelte Datensicherung.
Die Modularität und Wiederverwendbarkeit des Satelliten erlaubt den einfachen Tausch des Sekundärmoduls. Das Primärmodul kann als Basissatellit für beliebige Missionen genutzt werden. Die missionsspezifischen Funktionen können dann über das Sekundärmodul hinzugefügt werden. Dafür kann einfach eine neue Hülle gedruckt werden.

Kapitel 3

Technische Beschreibung

Wir haben uns für eine starke Trennung der zwei Missionen entschieden. Beide Module sind vollständig autark und können frei vom anderen Modul verwendet werden. Trotzdem werden wir einige Schnittstellen zwischen den gekapselten Modulen implementieren. Die Stromversorgung zwischen den Modulen werden wir miteinander verbinden. Im Falle eines Ausfalls der Stromversorgung in einem der Module kann so das jeweils andere Modul die Versorgung übernehmen. Außerdem wird es eine serielle Verbindung zwischen den jeweiligen Platinen geben, um die Daten ebenfalls doppelt zu speichern. Die Liste der technischen Anforderungen befindet sich im Anhang.

3.1 Mechanisches Design

3.1.1 Hülle

Da unser Satellit aus unabhängig voneinander funktionierenden Modulen bestehen soll, mussten wir uns überlegen, wie die Sekundärmission mit der Primärmission zusammengebaut werden soll. Ursprünglich war unser Plan das Primärmodul vollständig in das Sekundärmodul hineinzusetzen. Schwachstelle an diesem Design war der verschwendete Platz in der Hülle, da die Hülle auf Höhe der Primärmission doppelwandig wäre. Außerdem war es schwierig die einmal in die Sekundärmission hinein gesetzte Primärmission wieder zu entfernen, da man keine Möglichkeit hatte diese festzuhalten. Die neu designte Hülle B.3 umgeht diese Probleme, indem nur ein kleiner Teil der Missionen zusammengesteckt wird. Beide Teile haben nun eine einheitliche Wanddicke von 3 mm. Dadurch ergibt sich ein Innendurchmesser von 61 mm. Durch den größeren und einheitlichen Durchmesser der Missionen wurde das Platinendesign erheblich leichter und wir konnten die komplette Primärmission auf einer einzigen Platine unterbringen. Dies wiederum ermöglicht eine flachere Primärmission von nur 35 mm Höhe im Inneren, welche 70 mm Platz für jede Art von Projekten im inneren des Sekundärmoduls lässt. Die zusammengeschobenen Missionen werden mit vier Schrauben und Muttern an der Seite zusammengehalten. Mit dem 5 mm dicken Boden und der ebenso dicken Decke erreicht die Dose somit eine Gesamthöhe von 115 mm und einen Durchmesser von 67 mm. Weiterhin bestehend, wenn auch leicht verändert, ist das Sensorloch an der Seite der Primärmission. Dieses ermöglicht dem von uns eingebauten Infrarottemperatursensor das schnelle und zuverlässige Messen der Temperatur anhand eines Luft umströmten Metallplättchens welches am Äußeren des Satelliten angebracht ist.

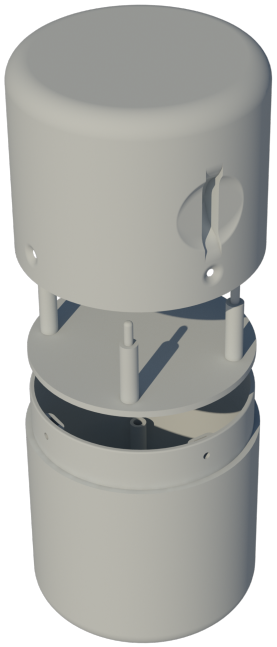


Abbildung 3.1: Hülle

Eine weitere Herausforderung war die Befestigung der Platinen im Inneren des Satelliten. Wir haben uns für Stäbe oberhalb und unterhalb der Platine entschieden zwischen welche die Platine geklemmt wird. Der untere Stab enthält am oberen Ende einen Stift. Dieser Stift wird durch Löcher in der Platine gesteckt und auf der anderen Seite in Löcher innerhalb der gegenüberliegenden Stäbe eingeführt. Um das nötige Gewicht von 340 g bis max. 350 g werden wir die Hülle mit nötigen Gewicht ausstatten und dieses entsprechend befestigen um ein verrutschen zu vermeiden. Das Gewicht wird im Sekundärmodul untergebracht, da nach unserem Konzept das austauschbare Modul unterschiedliches Gewicht haben kann. Die gesamte Hülle wurde in AutoCAD 2018 entworfen und dann gedruckt.

3.1.2 Fallschirm

Unser Fallschirm B.1 B.2 wurde aus stabilem luftundurchlässigem Stoff genäht. Zur Verbindung mit dem Satelliten verwenden wir 12 Schnüre, die jeweils zu viert an einer Ringmutter am Satelliten befestigt sind. Jeder dieser Schnüre hält einem Gewicht von 100kg stand. Zur Berechnung der Fläche des Fallschirms haben wir folgende Formel verwendet:

$$A = \frac{m * g}{c_w * \frac{1}{2} * \rho * v^2}$$

Wir erhielten dabei einen Wert von $191,2cm^2$ für die Fläche des Fallschirms:

$$A = \frac{0,35kg * 9,81\frac{m}{s^2}}{1,33 * \frac{1}{2} * 1,2\frac{kg}{m^3} * 15\frac{m}{s^2}^2} \approx 0,01912m^2 = 191,2cm^2$$

Daraus ergaben sich für einen sechseckigen Fallschirm mit einem sechseckigen Loch in der Mitte folgende Abmessungen:

- Fläche des des Lochs: $A = \frac{3}{2} * (1,5cm)^2 * \sqrt{3} \approx 5,85cm^2$
- Fläche des Fallschirm inklusive Loch: $A_g = 191,2cm^2 + 5,85cm^2 = 196,05cm^2$
- Radius des Fallschirms: $r = \sqrt{\frac{197,05cm^2}{\frac{3}{2}*\sqrt{3}}} \approx 8,7cm$

Der Fallschirm wird auf die Hülle hinauf gelegt, sodass dieser nur wenig über die Hülle hinausragt und somit den Technischen Anforderungen entspricht.
Da unser Satellit zum aktuellen Stand noch nicht vollständig zusammen gebaut ist, konnten wir die entsprechende Kraft, die der Satellit aushalten soll, nicht zusätzlich durch einen Test beweisen. Zudem ist uns aufgefallen, dass laut den technischen Anforderungen das Bergungssystem eine Kraft von 1000N aushalten soll. Zudem glauben wir, dass es sich bei den geforderten 1000N um einen Fehler handelt. Bei einem Gewicht von 350 g entsprechen 1000N nach der Grundgleichung der Mechanik einer, im Falle des Fallschirms negativen, Beschleunigung von $2857m/s^2 (\approx 291g)$.

$$1000N = 350g * a$$

Wir glauben nicht, dass so eine hohe negative Beschleunigung entstehen kann, insbesondere, da sich der Fallschirm direkt nach dem Auswurf des Satelliten zu einem Zeitpunkt öffnet, zu dem der Satellit noch recht langsam fällt.

3.1.3 Innerer Aufbau

Um Platz zu sparen haben wir unsere eigene Platine entwickelt und in China bestellt. Da wir auf Kabel weitestgehend verzichten wollen, haben wir uns zum Ziel gesetzt alles auf einer Platine zu realisieren. Dadurch ist das Layout sehr eng geworden. Schlussendlich war das aber kein Problem. Durch die Fertigung in China hatten wir die Möglichkeit einen Mikroprozessor mit sehr feinem Abstand zwischen den Pins zu verwenden. Außerdem können wir so eine zweiseitige Leiterplatte realisieren, was uns beim Ätzen nicht gelungen wäre.

Aus Platzgründen haben wir uns von vornherein gegen Lochrasterplatinen entschieden. Um die Machbarkeit zu testen, haben wir eine Testplatine bestellt und die Komponenten verlötet. Bis auf das GPS hat dort direkt alles funktioniert. Da wir durch das GPS keine Daten empfangen konnten, bzw. nicht entschlüsseln konnten, haben wir uns dafür entschieden ein anderes Modul, das GPS-GY6MV2, zu verwenden. Durch die unterschiedliche Größe musste das Platinenlayout geändert werden. Außerdem bestellten wir gleichzeitig noch die Platine für das Sekundärmodul. Die Verbindung des Primärmoduls mit dem Sekundärmodul wird mit speziellen Pico-loc Steckern gelöst. Damit haben wir keine losen Kabel im Satelliten, welche sich von der Platine lösen könnten und alles bleibt sehr überschaubar. Mit dem selben Verfahren sind die Akkus an der Platine befestigt, sodass diese leicht ausgetauscht werden könnten. In der Platine sind Löcher eingebaut, damit diese in der gedruckten Dose befestigt werden kann. Dies gilt für das Primär- und Sekundärmodul. Dadurch hatten wir sehr wenig Platz auf den Platinen und die Löcher mussten leicht versetzt werden, sodass diese nicht in allen Ecken gleich positioniert sind.

3.2 Elektrisches Design

3.2.1 Stromversorgung

Beide Missionen erhalten eine eigene Stromversorgung, können sich selber versorgen, teilen sich aber die Last wenn sie verbunden sind. Wir haben uns aufgrund der guten Kapazitäten für Lithium-ionen Akkus entschieden. Um den Spannungsregler nicht am Limit zu fahren, setzen wir zwei Zellen in Serie ein.

Bei dem Spannungswandler haben wir uns nochmal umentschieden. Statt des Linearreglers verwenden wir jetzt einen Step-down Schaltregler. Das Modul ist in dem Bereich, für den wir es verwenden laut Datenblatt immer mit über 90% Effizienz angegeben.

Um die Module auch verbinden zu können, wenn die Akkuspannungen ungleich ist, haben wir die Verbindung der Stromversorgung zuerst über eine Diode geleitet und erst dann mit der 3.3V Schiene des Satelliten und dem anderen Modul verbunden. So kann kein Ausgleichsstrom zwischen den Spannungswandlern fließen. Der Akku mit mehr Spannung versorgt dann immer das ganze System. Wenn beide Akkus die gleiche Spannung erreicht haben werden sie gleichmäßig genutzt. Der einzige Nachteil ist der Spannungsabfall über der Diode. Deshalb haben wir eine Shottkeydiode mit einem sehr geringen Spannungsabfall von knapp 0,4V genommen. Da unser System auf max. 500mAh konzipiert ist verbraucht die Diode im Maximalfall 0,2W.

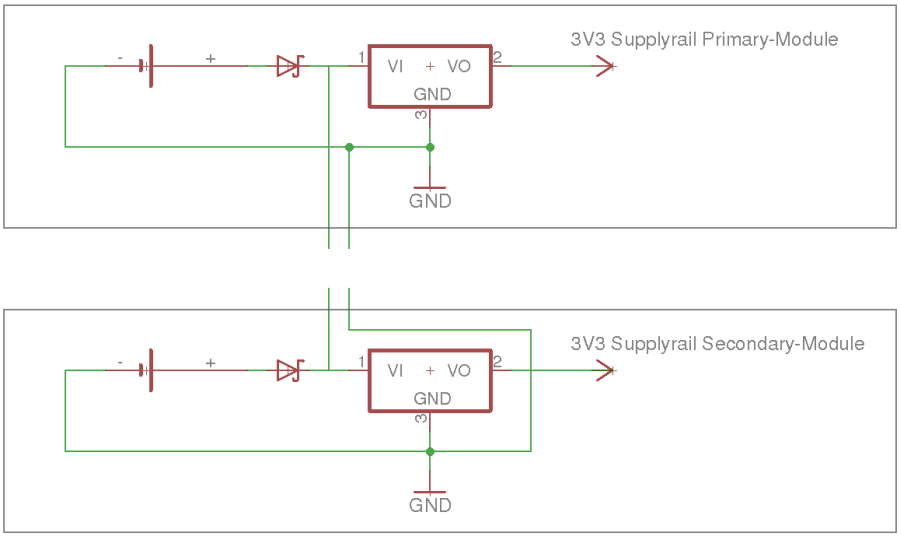


Abbildung 3.2: Steckverbindung

Leider erlauben unsere Akkus nur einen maximalen Entladestrom von 500mA. Deshalb darf eine beliebige Sekundärmission keinen Stromverbrauch über 320mA haben, da im Zweifel das Primärmodul nicht beide Module mit Strom versorgen könnte. Ein solches Sekundärmodul müsste entweder auf die redundante Stromversorgung verzichten und die Verbindung trennen, oder Akkus in das Primärmodul einbauen, die einen höheren maximalen Entladestrom haben.

Die Ladeelektronik ist aus Platzgründen nicht im Satelliten verbaut. Die Akkus können leicht herausgenommen werden. Zum Beispiel um sie zu tauschen. Können der Einfachheit halber aber auch im System geladen werden. Die nötigen Leitungen für das balancierte Laden sind herausgeführt. Unser externer Akkulader kann dann sogar alle 4 Zellen gleichzeitig aufladen.

Um den Ladezustand zu überwachen ist über einen Spannungsteiler ein ADC vom Mikrocontroller angeschlossen. Die Spannungsüberwachung ist auf beiden Platinen implementiert.

Für das Ein- und Ausschalten haben wir uns für einen Schlüsselschalter entschieden. Dieser benötigt zwar bauartbedingt nicht zwingend den Schlüssel um geschaltet zu werden, sollte aber dennoch von keiner auftretenden Kraft umgelegt werden. Die Benötigte Kraft müsste dafür rotierend und ziemlich groß sein.

Um den Schalter zu betätigen muss zwar die Platine aus dem Gehäuse genommen werden. Wir sehen das in unserem Fall trotzdem als leicht erreichbar an, denn die Platine kommt mit allen Bauteilen sehr einfach aus dem Gehäuse. Von außen bedienbar haben wir den Schalter nicht gemacht, weil wir dann bei Regen weniger Spritzwasserschutz hätten und weil der Schalter wirklich nur absichtlich bedient werden können soll. Außerdem könnte bei der Landung Dreck in der Schlüsselschalter kommen.

Akkukapazität

Um die Akkukapazitäten zu berechnen und einen Beweis der Laufzeit von vier Stunden zu erbringen haben wir zuerst den Stromverbrauch gemessen. Dann haben wir auf Basis unserer Akkus und unserer Spannungswandlung errechnet, wie viel Strom wir über einen Zeitraum von 4 Stunden bereitstellen können.
Da wir in beiden Modulen die gleich Stromversorgung haben können wir den maximalen Strom leicht berechnen:

Durchschnittliche Akkuspannung (2 Lithium-ionen Zellen):

$$3,7V * 2 = 7,4V$$

Akkuspannung hinter der Diode:

$$7,4V - 0,4V = 7V$$

Energie, die dem Spannungswandler zur Verfügung steht:

$$7V * 500mAh = 3,5Wh$$

Energie hinter dem Spannungswandler auf der 3,3V Schiene:

$$3,5Wh * 0,9 = 3,15Wh$$

Zur Verfügung stehender Strom auf der 3,3V Schiene:

$$(3,15Wh/3,3V)/4h \approx 238mA$$

Die Stromversorgung jedes einzelnen Moduls kann über vier Stunden hinweg maximal **238mA** liefern.
Das Primärmodul kann sich mit **180mA** Stromverbrauch über **4** Stunden selbst versorgen. Laut unseren Berechnungen sollte es **5,3** Stunden durchhalten.

Das Sekundärmodul benötigt nur **100mA** und kann deshalb ganze **9,5** Stunden alleine laufen.
Sollte eine der Stromversorgungen nicht funktionieren springt sie für beide Module ein. Dann hält der Akku keine 4 Stunden mehr, sondern nur noch **3,4** Stunden.
Sollten wie geplant beide Stromversorgungen funktionieren kann der Satellit sogar bis zu **6,8** Stunden laufen.
Das Primärmodul kann sich also problemlos selbst versorgen und ist unabhängig von der Sekundärmission. Falls eine Sekundärmission aber wesentlich längere Laufzeit erfordern sollte ist hiermit der Grundstein gelegt um mehr Kapazität hinzuzufügen.

3.2.2 Primärmission

Es ist uns gelungen sämtliche Bauteile des Primärmoduls auf einer Platine B.4 B.5 B.6 unterzubringen. Diese haben wir in Eagle entwickelt. Um möglichst viel Platz zu sparen, haben wir die Widerstände, Kondensatoren und LEDs als SMD-Bausteine verbaut.
Die Platine besteht aus zwei Layern. Zusätzlich zu den verbauten Sensoren messen wir die aktuelle Akkuspannung über einen ADC des Mikrocontrollers. Da dieser nur in einem Spannungsbereich von 0-1.1V lesen kann haben wir einen Spannungsteiler vorgeschaltet, der die Akkuspannung durch 8 teilt. So lässt sich der Wert später leicht wieder zurück rechnen.

Mikrocontroller

Anforderungen

Abbildung 3.3: Mikrocontroller

- Klein
- Unterstützung von Bibliotheken
- Betrieb auf 3,3V
- I^2C Bus
- Serielle Ports
- ADC

Wir haben uns für die Primärmission umentschieden und haben einen Mikrocontrollersystem von Atmel, den ATMega2560V [5], im Satelliten verbaut. Da es per Hand noch relativ gut lötbar ist haben wir den Mikroprozessor im TQFP-100 Gehäuse gekauft. Der Chip ist damit 16x16mm groß. Trotz seiner geringen Größe, besitzt er alle nötigen Schnittstellen um die Sensoren und andere Bauteile des Satelliten anzuschließen. Außerdem unterstützt er sehr viele Bibliotheken von Arduino. Da wir nur den Mikrocontroller selbst verbauen und nicht das dazugehörige E/A-Board (ATMega2560), sparen wir erheblich Platz, denn der Mikrocontroller braucht nicht mehr als einen Schwingquarz und eine Stromversorgung. Außerdem können wir dadurch, wie schon geplant war, den gesamten Satelliten mit einer Spannung von 3,3V betreiben. Durch die geringere Spannung darf der Mikrocontroller jedoch nicht mehr mit den ursprünglichen 16MHz betrieben werden, sondern nur noch mit 10Mhz. Wir werden den Satelliten jedoch nur mit 8MHz takten, was für unsere Zwecke definitiv ausreicht.

Sensorik

Abbildung 3.4: Infrarotsensor

Infrarottemperatursensor Wir haben uns für einen Infrarottemperatursensor (MLX90614) [12] zur Bestimmung der Temperatur entschieden, da eine Temperaturmessung auf diesem Weg sehr schnell und genau erfolgt. Ein Infrarottemperatursensor hat damit entscheidende Vorteile gegenüber einem temperaturabhängigen Widerstand, da dieser sich erst an die Umgebungstemperatur anpassen muss und somit wesentlich langsamer ist. Ein weiterer Vorteil

ist, dass der Sensor wie die meisten unserer Sensoren, über einen I^2C Bus angeschlossen und mit 3,3V betrieben werden kann. Um die Temperatur genau zu messen ist eine von Luft umströmte Metallplatte nötig, auf die der Temperatursensor zeigt. Mit der „Sparkfun MLX90614“ Bibliothek werden die Daten vom Sensor ausgelesen.

Luftdrucksensor Die Messung des Luftdrucks wird in unserem Satelliten durch den Sensor BMP180 [8] durchgeführt werden. Wir haben uns für diesen Sensor entschieden, da er wie die anderen Bauteile ebenfalls mit 3,3V arbeitet und wie diese an einen I^2C Bus angeschlossen werden kann. Des Weiteren ist der Sensor sehr klein und kompakt, um viel Platz im Satelliten zu sparen. Die Luftdruckmessung erfolgt über eine Membran, die auf den herrschenden Luftdruck reagiert. Das Breakoutboard, welches wir verwenden, enthält neben dem Luftdrucksensor auch einen Temperatursensor, welcher die Temperatur jedoch über einen temperaturabhängigen Widerstand bestimmt und sich aus diesem Grund wegen seiner Trägheit nicht für die benötigten schnellen Messungen eignet.

Abbildung 3.5: Luftdrucksensor

Abbildung 3.6: IMU

IMU Wir haben uns für die Messung der Beschleunigung und der Position für die Verwendung des GY-521 Breakouts des MPU6050 [11] entschieden. Bei diesem Sensor handelt es sich um eine inertiale Messeinheit (IMU), also einen Sensor, der einen Beschleunigungssensor und ein Gyroskop kombiniert. Mit Hilfe der von uns verwendeten Bibliothek können die Berechnungen zur Ermittlung der Beschleunigung aus der Sicht eines Beobachters bereits durch den Mikrocontroller durchgeführt werden.

Transceiver Für den Transceiver [13] haben wir uns zum HC-12 SI4463 umentschieden, da der vorherige Transceiver CC1101 nicht die gewünschte Reichweite hatte und außerdem viel zu kompliziert angeschlossen wurde. Dieser wird über eine serielle Schnittstelle mit unserem Mikrocontroller verbunden und mit einer Spannung von 3,3V betrieben. Der Transceiver ist mit dem einzigen Befehl Serial.write(); zum Senden sehr einfach zu bedienen, was uns die Verwendung Bibliotheken erspart. Er sendet auf dem 433 MHz-Band und hat eine Reichweite von 1-2 Km. Unser Transceiver sendet auf 433,4 MHz.

Abbildung 3.7: Transceiver

Abbildung 3.8: Micro-SD-Kartenadapter

Micro-SD-Kartenadapter Als Micro-SD-Kartenadapter entschieden wir uns für den Sparkfun Openlog DEV-13712[4]. Einer der Hauptaspekte warum wir uns für dieses Modell entschieden haben war, dass wir einen möglichst kleinen Micro-SD-Kartenadapter haben wollten, welcher außerdem mit 3,3V arbeitet. Zusätzlich ist auf dem Micro-SD-Kartenadapter schon ein ATmega328p-PU Mikrocontroller verbaut. Durch diesen Mikrocontroller ist eine einfache Kommunikation zwischen unserem Mikrocontroller und dem Micro-SD-Kartenadapter über eine serielle Schnittstelle möglich.

GPS-Empfänger Unser GPS Modul ist das GY-GPS6MV1 [14] Modul, welches über eine serielle Schnittstelle angesprochen wird. Dieses Modul wird ebenfalls mit einer Spannung von 3,3V versorgt und ist mit der Bibliothek TinyGPS++ sehr leicht auszulesen. Wir haben uns wieder umentschieden, da der GPS Sensor A2235-H, welcher viel kleiner ist und auch über I^2C angeschlossen werden kann, Daten bekam, die wir nicht entschlüsseln konnten.

Abbildung 3.9: GPS-Empfänger

Auch nach langer Recherche im Internet kamen wir zu keinem Ergebnis. Der ursprünglich geplanten EM-411 entfiel schnell, da dieser eine Spannung von 5V benötigte.

Buzzer Unsere Platine verfügt über einen magnetischen Summer. Diesen werden wir verwenden, um den Satelliten über die akustischen Signal nach der Landung zu finden.

Bauteil	Modell	Kommunikation	Spannung
Infrarottemperatursensor	MLX90614	I^2c	3,3V
Luftdrucksensor	BMP180	I^2c	3,3V
IMU	MPU6050	I^2c	3,3V
Transceiver	HC-12 SI4463	UART	3,3V
Mico-SD-Kartenadapter	Openlog	UART	3,3V
GPS-Modul	GY-GPS6MV1	UART	3,3V

Tabelle 3.1: Sensorik Primärmodul

3.2.3 Sekundärmission

Die Platine B.7 B.8 B.9 des Sekundärmoduls ist ähnlich aufgebaut wie die des Primärmoduls. Bei elementaren Bauteilen wie dem Spannungswandler, dem Schalter und dem Micro-SD-Kartenadapter verwenden wir die gleichen Bauteile wie im Primärmodul. Statt dem Atmega2560v verwenden wir jedoch einen Atmega328p-pu.

Mikrocontroller

Anforderungen

- Klein
- Unterstützung von Bibliotheken
- Betrieb auf 3,3V
- Serielle Ports

Abbildung 3.10: Mikrocontroller Sekundärmodul

Wir haben uns für das Sekundärmodul für ein Mikrocontrollersystem [7] [6] von Atmel entschieden, denn die Unterstützung durch Bibliotheken ist aufgrund der Arduinos sehr gut. Wir setzen allerdings aufgrund des größeren verfügbaren Platzes und der kleineren Menge an benötigten Schnittstellen auf einen ATMega 328p-pu. Dieser Chip ist auch auf dem Arduino Uno verbaut und benötigt im Prinzip nur einen Schwingquarz und eine Stromversorgung wie beim Primärmodul. Der Atmega328p-PU verfügt über alle von uns benötigten Schnittstellen im Sekundärmodul und kann auch mit einer Spannung von 3,3V betrieben werden. Bei 3,3V darf dieser ebenfalls nur noch mit einem 8MHz Quarz betrieben werden.

Sensorik

Abbildung 3.11: Transceiver Sekundärmodul

Transceiver Für das Sekundärmodul verwenden wir den HM TRP868 [10], welcher ein Transceiver ist, der auf dem 868 MHz-Band sendet und mit einer Spannung von 3,3V betrieben werden kann. Dieser wird über eine serielle Schnittstelle mit dem Mikrocontroller verbunden und kann über diese leicht angesprochen werden. Unser Transceiver wird auf einer Frequenz von 869 MHz senden.

Bauteil	Modell	Kommunikation	Spannung
Mico-SD-Kartenadapter	Openlog	UART	3,3V
Transceiver Sekundärmission	HM TRP 868	UART	3,3V

Tabelle 3.2: Sensorik Sekundärmodul

3.3 Softwaretechnisches Design

Wir verwenden zum Auslesen der verschiedenen Sensoren und des GPS-Empfängers mehrere Bibliotheken. Unser Micro-SD Kartenadapter und unsere Transceiver können ohne zusätzliche Funktionen über eine serielle Schnittstelle angesprochen werden. Dies hat die Entwicklung unseres Programms erheblich vereinfacht. Da unser Programm, welches objektorientiert geschrieben war, zu Beginn häufig nach zwei Schleifendurchläufen abgebrochen ist, haben wir uns entschieden, dass gesamte Programm, welches auf dem Mikrocontroller unseres Satelliten läuft, abgesehen von den verwendeten Bibliotheken funktional zu programmieren.

Bauteil	Bibliothek
Luftdrucksensor	Adafruit-BMP085
IMU	MPU6050
Infrarottemperatursensor	SparkFun_MLX90614_Arduino_Library
GPS-Modul	TinyGPS++
simulierte serielle Schnittstellen	SoftwareSerial

Tabelle 3.3: Bibliotheken

3.3.1 Primärmodul

Nach der Initialisierung der einzelnen Sensoren, des GPS-Moduls und der seriellen Schnittstellen werden die Sensoren und das GPS-Modul sowie die serielle Schnittstelle, die die Verbindung zum Sekundärmodul darstellt nacheinander abgefragt und die vorhandenen Daten auf die SD-Karte geschrieben sowie über ein einfaches Funkprotokoll $satellitID\_dataID\_time\_data\_check$ zur Bodenstation übertragen. Abgesehen von den Daten, die der Mikrocontroller vom Sekundärmodul erhalten hat, werden außerdem alle Daten auch über die serielle Schnittstelle an das Sekundärmodul weitergegeben, um dort ebenfalls gespeichert zu werden. [3]

3.3.2 Sekundärmodul

Im Sekundärmodul müssen lediglich die echte und die durch die SoftwareSerial Bibliothek simulierten seriellen Schnittstellen initialisiert werden. Danach werden abwechselnd die serielle Schnittstelle, an die der Transceiver angeschlossen ist und die serielle Schnittstelle, die die Verbindung zum Primärmodul darstellt abgefragt und die Daten weiter verarbeitet. Hierbei werden über die Funkverbindung empfangene Daten auf die SD-Karte geschrieben und außerdem zur Speicherung an das Primärmodul weitergegeben. Die Daten, die vom Primärmodul übertragen wurden, werden nur auf die SD-Karte geschrieben.

3.4 Bodenstation

Jegliche Programme des Clients wurden in der Programmiersprache Python (in der Version 3.5) programmiert. Zusätzlich wurde bei sämtlichen Programmen mit grafischen Benutzeroberflächen die Bibliothek „PyQt“ verwendet.

3.4.1 Empfänger

Um die Daten zu empfangen haben wir eine Klasse geschrieben, die als von der GUI getrennter Thread läuft. Diese überwacht die Funkstrecke, ließt die ankommenden Daten, validiert sie mittels einer Checksumme, verarbeitet sie, speichert sie in einer Datenbank und übergibt sie an die GUI. Für die Kommunikation mit dem Funkmodul verwenden wir einen USB/UART-Converter. Das Programm kommuniziert dann direkt mit dem Funkmodul. Pakete, bei denen sowohl die Übertragung validiert werden konnte, als auch die Verarbeitung möglich ist (dies kann bei Messfehlern eventuell unmöglich sein) werden an die GUI übergeben und zusätzlich in eine Datenbank geschrieben. Fehlerhafte Pakete werden gezählt und mit der Übertragungsrate ins Verhältnis gesetzt. Der Paketverlust wird auch in der GUI dargestellt. Ein Ausfall von Paketen von 200ms wird als Verbindungsabbruch gewertet. Um die berechnete Geschwindigkeit zu kalibrieren gibt es die Möglichkeit von der GUI aus die Berechnungen auf 0 zurückzusetzen. Dies sollte bei Stillstand passieren. Um die Daten des Primär- und des Sekundärmoduls zu trennen gibt es eine Satellite-Id. Diese wird bei jeder Übertragung mit übertragen. Es ist also der Grundstein gelegt, um jede beliebige Sekundärmission leicht in die Software einzupflegen.

3.4.2 Datenbank

Die Datenbank haben wir in SQL-Lite realisiert. SQL-Lite zeichnet sich durch den Syntax, der dem uns gut bekannten Syntax von MySQL sehr ähnelt und die Verwendung einer einzigen Datei aus. Die Datenbank ist somit leicht bedienbar und portabel. Wir können leicht Sicherungskopien anlegen und diese wieder einspielen. Die Datenbank speichert:

- Alle Rohdaten vom Satelliten. Egal ob die Validierung der Checksumme erfolgreich war oder nicht
- Alle berechneten Sensordaten, die auch in der GUI angezeigt werden
- Den Paketverlust
- Die Zeitpunkte von Kalibrierungen aus der GUI

3.4.3 Datenverarbeitung

Die Datenverarbeitung ist dafür zuständig, die vom Satelliten erhaltenen Rohdaten aufzuarbeiten, zu verrechnen und auf diesem Weg Werte wie die Höhe, die Geschwindigkeit und die Position zu ermitteln.

Luftdruck und Temperatur

Diese Klasse ist für die Verarbeitung von Luftdruck- und Temperaturdaten zuständig. Sie nimmt den aktuellen Luftdruck in Pascal und die aktuelle Temperatur in Kelvin auf und berechnet daraus die Temperatur in Grad Celsius und Grad Fahrenheit und mit Hilfe der nach der Höhe umgestellten barometrischen Höhenformel [2] [1] die Höhe in Metern. Um die Berechnung der Höhe genauer durchzuführen, besteht die Möglichkeit innerhalb des Programms den Luftdruck und die Temperatur am Boden einzustellen.

Berechnung K zu $°C$: $T_C = T_K - 273,15$

Berechnung K zu $°F$: $T_F = (T_K * 1,8) - 459,67$

Berechnung h: $h = \frac{\log_{1-\frac{T_{h0}-T_{h1}}{T_{h0}}}(\frac{p_{h1}}{p_{h0}} * (t_{h0} - t_{h1}))}{0{,}03416}$

IMU

Diese Klasse ist für die Verarbeitung von Beschleunigungs- und Lagedaten zuständig. Sie nimmt neben der aktuellen Beschleunigung in g und der aktuellen Lage in ° jeweils in x-, y- und z-Richtung, den Zeitpunkt in

Millisekunden seit dem Start des Programms [3] auf. Da die Bibliothek des IMU bereits über Funktionen zur Berechnung der realen Beschleunigung (aus Sicht des Beobachters, nicht des Sensors) verfügt, müssen diese Berechnungen nicht mehr durch die Bodenstation durchgeführt werden. Die Klasse berechnet aus der Beschleunigung in g die Beschleunigung in m/s^2 und daraus über den zeitlichen Abstand zwischen zwei Messungen die aktuelle Geschwindigkeit [9] und die aktuelle Position [9]. Diese Berechnungen werden für Bewegungen in x-, y- und z-Richtung durchgeführt. Aus diesen Werten wird dann wiederum die gesamte Beschleunigung und Geschwindigkeit, sowie die aktuelle Position und die Entfernung vom Startpunkt berechnet. Für die einzelnen Berechnungen werden folgende Formeln verwendet.

Berechnung $a_{Richtung}$ in m/s^2:	$a_R = a_{Rg} * 9,81 m/s^2$
Berechnung $v_{Richtung}$:	$v_1 = v_0 + (a_1 * \Delta t)$
Berechnung $s_{Richtung}$:	$s_1 = s_0 + (v_0 * \Delta t) + (0,5 * a_1 * \Delta t^2)$
Berechnung x_{gesamt}:	$x_{gesamt} = \sqrt{x_x^2 * x_y^2 * x_z^2}$
Berechnung Strecke in m:	$s_{gesamt} = s_{gesamt} + \sqrt{(s_{x1} - s_{x0})^2 + (s_{y1} - s_{y0})^2 + (s_{z1} - s_{z0})^2}$

GPS

Die Methode verarbeitet die vom Satelliten empfangenen Daten und gibt Längen- und Breitengrad zurück.

3.4.4 Receive Client

Die Aufgabe des Receive Clients ist es die empfangenen und bereits verarbeiteten Daten des Satelliten in Echtzeit anzuzeigen und wird während dem Satellitenstart als Interface zur Überwachung der Daten dienen.

GUI

Angefangen wurde mit dem Design der grafischen Benutzeroberfläche. Hierbei achteten wir auf eine möglichst modulare Bauweise und eine große Übersicht, um wichtige Informationen direkt einsehen zu können. Im ersten Design B.12 sollte ein Graph im oberen linken Bereich Platz finden, ein Monitor der die Rohdaten des Satelliten anzeigt darunter und auf der rechten Seite wurden Informationen der Primär- und Sekundärmission auf das Nötigste reduziert und ebenfalls angezeigt. Dieser grobe Aufbau, mit dem Graphen oben links dem Rohdatenmonitor darunter und den Informationen der Missionen auf der rechten Seite, zieht sich ebenfalls durch den zweiten Entwurf und ist im finalen Design immer noch zu erkennen. In der finalen Version B.13 der Benutzeroberfläche befindet sich auf der linken Seite im oberen Bereich ein Platzhalterfeld für die später im Code erzeugten Graphen, darunter der Rohdatenmonitor und auf der linken Seite befindet sich ein „ToolBox“-Element, welches eine Aufteilung der einzelnen Informationsbereiche ermöglicht. Aufgeteilt wurden die Bereiche so, dass bei Programmstart immer die Übersicht mit den wichtigsten Daten zu sehen ist, darunter kann nun ein anderes Feld ausgewählt und die enthaltenen Informationen angezeigt werden. Im Feld „GPS“ befinden sich Statusinformationen zum GPS-Modul der Primärmission, die zuletzt empfangenen GPS-Koordinaten und ein vom Programm erzeugter Browser, welcher die aktuelle Position mit Hilfe der „Google Static Maps API“ darstellt. Im Bereich „Temperatur + Luftdruck“ lassen sich alle Informationen bezüglich der Temperatur, des Luftdruck und die aus den beiden Werten berechnete Höhe ausgeben. Im Reiter „Lage + Beschleunigung“ finden sich alle Daten zur Lage & Beschleunigung und den daraus resultierenden Werten zur Position und zurückgelegter Entfernung. Als letzter Bereich ist der Platzhalter der Sekundärmission übrig, welcher einfach ausgetauscht werden kann und erst im späteren Verlauf des CanSat-Wettbewerbs relevant wird. Als letztes befinden sich im oberen Bereich der Benutzeroberfläche die Menüleiste und am unteren Rand die „Statusbar“ welche diverse Funktionen beinhalten.

Programmierung

Der grundlegende Aufbau des Programms sieht wie folgt aus: In der GUI Klasse gibt es für jeden Graphen und jede Kategorie der Toolbox eine Methode zum aktualisieren des Graphen oder zum Laden der Daten, diese Methoden sind nun mit einem QTimer verbunden, welcher in einem angegebenen Zeitraum (standardmäßig 200 ms) die Methoden ausführt. Um die Performance zu schonen wird der Browser mit einem eigenen 5 Sekunden Timer und bei der Toolbox immer nur die angezeigten Daten neu geladen. Die Programmierung des Receive Clients gestaltete sich als relativ mühsam, da sämtliche Anzeige- und Ausgabeflächen manuell im Code mit der dazugehörigen Funktion zum Laden der Daten verbunden, bzw. „connected“, werden mussten. Zusätzlich war die Implementierung der Graphen in die Benutzeroberfläche nicht ohne weiteres möglich. Hierfür wurde anfangs die Bibliothek „PyQt-Chart“, welche eine offizielle Erweiterung der PyQt Bibliothek darstellt, verwendet. Durch die Verwendung von PyQt-Chart stellte sich nach erfolgreicher Implementation die Existenz eines Memory-leaks heraus, welcher einen langfristigen Betrieb ohne zusätzliche Performanceeinbußen nicht möglich machte. Nachdem wir herausfanden das der Memory-leak auf PyQt-Charts zurückzuführen ist, wechselten wir die Bibliothek auf „PyQtGraph“, welche nun einen performanteren Dauerbetrieb des Clients ermöglicht. Der Client ist mittlerweile mit der neuen Datenverarbeitung verbunden und kann die über Funk vom Satelliten empfangenen Daten in Echtzeit wiedergeben. Zudem ist die Rekalibrierung der Geschwindigkeit und das Zurücksetzen der Zeit durch einen Eintrag in der Menüleiste möglich geworden.

3.4.5 Transmit Client

Der Transmit Client dient der Erzeugung von Daten, welche zum Testen der Bodenstation oder zur Simulation eines Satelliten verwendet werden können. Er wird während des Satellitenstartes höchstwahrscheinlich zur Simulation eines weiteren Satelliten für die Sekundärmission verwendet.

GUI

Im Gegensatz zur Receive GUI besitzt die Transmit GUI B.15 keinen mehrstufigen Entwurfsprozess, sondern wurde mit dem ersten Entwurf übernommen und lediglich Kleinigkeiten angepasst. Die GUI des Transmit Client bietet die Möglichkeit für jeden Datentyp einen Wert einzustellen oder einen zufälligen Wert in einem bestimmten Bereich zu generieren. Außerdem kann man das Erzeugen verschiedener Datengruppen abschalten. Aufgebaut ist die GUI zeilenweise. Das bedeutet das für jeden Datentyp der generiert werden kann genau eine Zeile benutzt wird. Dabei geht es als erstes mit den Takteinstellungen los, hier kann per Knopfdruck ein Datensatz generiert, oder eine Clock eingestellt werden, welche die Erzeugung übernimmt. Unter den „Takt Einstellungen“ befinden sich die „GPS Einstellungen“, dort finden sich Einstellungsmöglichkeiten zur Anzahl der Satellitenverbindungen, der Höhe, wobei hier entschieden werden kann zwischen einer statischen, einer sinkenden und einer schwankenden Generierung und der Koordinaten. Gefolgt wird den Einstellungen mit, nach demselben Prinzip strukturierten Einstellungsgruppen für die Temperatur, den Luftdruck, die Lage und die Beschleunigung. Abgeschlossen wird die GUI mit einem Ausgabemonitor, welcher die generierten Daten in Echtzeit anzeigt und so als zusätzliche Kontrollmöglichkeit im Einsatz dient.

Programmierung

Ähnlich wie beim Receive Client werden die verschiedenen Eingabeelemente über eine Methode in der GUI Klasse abgefragt, dabei werden sie im Ausgabemonitor angezeigt und zusammen mit einem Zeitstempel in einer Logdatei gespeichert, welche in einem extra Ordner abgelegt wird. (Die Logdateien tragen dabei immer den Namen: $TransmitLog\_Tag.Monat\_Stunde.Minute$.) Nun werden die einzelnen Methoden wieder mit einem QTimer verbunden und zusätzlich mit dem Button für den manuellen Takt. Der QTimer wird mit dem vom Benutzer eingestelltem Intervall gestartet sofern die dafür vorhergesehene Checkbox aktiviert wurde.

3.4.6 CanSat Launcher

Der CanSat Launcher ist ein kleines Programm mit grafischer Benutzeroberfläche, welches das Starten verschiedenster Programme der Bodenstation, von einem zentralen Punkt aus, ermöglicht. Außerdem sorgt der Launcher dafür, dass die Konsolenausgabe sämtlicher, über den Launcher gestarteter Programme, in einem Konsolenfenster angezeigt werden.

GUI

Die GUI des Launchers B.16 hat genauso wie die GUI des Transmit Clients keine vorhergehenden Designs oder eine Entwurfsgeschichte und wurde nur wenige Male angepasst. In der Benutzeroberfläche des Launchers hat man auf der linken Seite unter dem Titel eine Auswahlmöglichkeit von verschiedenen Radio Buttons, welche zur Auswahl des zu startenden Programms dienen. Rechts neben den Radio Buttons befinden sich Label welche zusätzlich die zu öffnende/startende Datei anzeigen. Abschließend befinden sich am rechten Rand der „Start“-Button, der „Schließen“-Button und ein Button zum Öffnen der „Team Starbugs“ Website.

Programmierung

In der GUI Klasse befinden sich beim Launcher lediglich eine Methode zum Beenden des Launchers, eine Methode zum Öffnen der „Team Starbugs“ Website und die Hauptmethode des Launchers, die „startClient“ Methode. Diese Methode prüft welcher Radio Button ausgewählt wurde und startet beim Auslösen des Startbuttons den jeweiligen Client.

3.4.7 Analyse

Wir fertigen ein Programm an, mit Hilfe dessen einzelne Messreihen analysiert werden können. Zum jetzigen Zeitpunkt ist das Programm in der Lage csv-Dateien eines bestimmten Formats einzulesen. Bei diesem Format stehen Daten eines bestimmten Typs, beispielsweise Uhrzeit, Höhe, Temperatur-, Luftdruck-, oder Geschwindigkeitsdaten jeweils in einer Spalte untereinander. Die sich in einer Zeile befindenden Daten sollten zu möglichst nah beieinanderliegenden Zeitpunkten aufgenommen worden sein, um mögliche Zusammenhänge zu erkennen. Um die Daten in einem Koordinatensystem darzustellen, ist es möglich einen Datensatz für die x-Achse, zum Beispiel die Höhe, und mehrere Datensätze für die y-Achse, zum Beispiel die Geschwindigkeit in horizontaler Richtung und die Temperatur, auszuwählen. In diesem Beispiel könnten so zum Beispiel die Windgeschwindigkeit und die Temperatur in verschiedenen Höhen dargestellt werden. In Zukunft soll das Programm die ausgewählten Daten in einer Tabelle anzeigen und als csv-Datei exportieren sowie den angezeigten Graphen als svg-Datei exportieren können. Die Arbeit an diesem Programm mussten wir leider abbrechen, da die anderen Bereiche des Projekts zu viel Zeit in Anspruch nehmen und wir aufgrund des durch das Abitur verkürzten Halbjahres außerdem sehr viel für die Schule arbeiten müssen.

Kapitel 4

Projektplanung

Die Planung C wurde anhand der vom Wettbewerb vorgegeben Termine und Laufzeiten erstellt. Dafür wurden alle bekannten Termine und Fristen in ein MS Project Dokument geschrieben und darauf aufbauend die vom Team geplanten Termine und Zeitrahmen gesetzt. Außerdem wurde bei den von uns geplanten Terminen ein gewisser Zeitraum als Puffer mit eingeplant um bei eventuellem Zeitverzug genügend Reserve zu besitzen. Nach diesen Grundregeln wurde am Anfang für die einzelnen Bereiche (Primär-, Sekundärmission und Bodenstation) geplant. Mit dem weiteren Verlauf des Projektes, vor allem im Frühjahr 2017, wurde die Planung immer wieder angepasst, um manche Fehleinschätzungen aus der Grundplanung zu korrigieren, da gewisse Aufgaben viel weniger oder auch etwas mehr Zeit in Anspruch nahmen oder nehmen als ursprünglich gedacht.

4.1 Entscheidungen

4.1.1 GPS

Wir haben uns gegen das EM-411 entschieden, da dieses zwar einfach angeschlossen wird, aber wir dafür noch einen Spannungswandler für eine Spannung von 5V in das Primärmodul hätten einbauen müssen. Außerdem verwarfen wir das A2235-H GPS Modul, da wir, wie oben schon genannt, die empfangenen Daten nicht entschlüsseln können. Das A2235-H GPS-Modul wurde sehr kompliziert angeschlossen, womit wir anfänglich Schwierigkeiten hatten. Da das Modul SMD gelötet wird, mussten wir vorher ein Breakout erstellen, welches uns viel Zeit kostete. Nur durch Ätzen konnten wir letztendlich das GPS Modul testen. Da wir die Daten, die wir vom Modul bekommen haben, nicht entschlüsslen konnten, haben wir uns letztendlich für das GY-GPS6MV1 entschieden, welches über eine serielle Schnittstelle angeschlossen wird und sehr einfach anzusprechen ist. Zum Entschlüsseln der Daten wird die Bibliothek TinyGPS++ verwendet.

4.1.2 Befestigung der Platinen

Um die Platinen zu befestigen, sind in diese kleine Löcher eingebaut. Diese werden dann, wie in der Zeichnung zu sehen, im Gehäuse in die entsprechenden Halterungen eingesetzt. Wenn der CanSat dann verschlossen wird, wird die Platine durch das Zusammendrücken festgehalten und kann nicht mehr verrutschen. Das gleiche Verfahren verwenden wir im Sekundärmodul.

4.1.3 Sekundärmodul

Da wir auf Komplexität verzichten wollten, haben wir unsere Sekundärmission geändert. Das Ziel ist das Gleiche: Einen Satelliten simulieren. Dafür arbeiten wir aber nicht mehr mit einem zweitem Team zusammen, sondern simulieren den Satelliten am Boden. Die Daten, die das Sekundärmodul empfängt gibt es an das Primärmodul

weiter. Dieses sendet sie an unsere Bodenstation zurück. Wir entschieden uns dafür vor allem, da wir vermeiden wollten, dass unsere Sekundärmission von einem anderen Team abhängt.

4.1.4 Gehäuse

Zu Beginn des Projekts war es das Hauptziel viele Ideen zu sammeln, wie unser Satellit aufgebaut sein sollte. Da wir uns von vornherein überlegt hatten ein modulares Design zu verwenden, bei welchem man die Sekundärmission beliebig austauschen kann, war es eine der größten Herausforderungen uns ein Konzept zu überlegen, welches dies ermöglichen würde. Einige andere Herausforderungen, die sich daraus ergaben, waren eine elektrische Verbindung zwischen den beiden Teilen herzustellen und eine Temperaturmessung mit einem Infrarotsensor durchzuführen. Unsere Ideen reichten von einer großen Hülle, für die Primärmission in welche dann die Sekundärmission geschoben wird, über ein Design, bei welchem die beiden Teile über ein Gewinde verschraubt werden, bis hin zu einem bei dem die Primärmission in die Sekundärmission geschoben wird. Die zweite Idee war nicht für den Satelliten geeignet, da die elektrische Verbindung sonst zu kompliziert, unpraktisch oder unsicher geworden wäre. Außerdem wäre diese Verbindung möglicherweise instabil geworden. Aus diesen Gründen haben wir uns für die unter 3.1.1 beschriebene Version entschieden. Da wir beim Einbau der Platinen bemerkt haben, dass 4 cm zu wenig sind, wurde die Größe des Primärmoduls auf 5 cm erhöht.

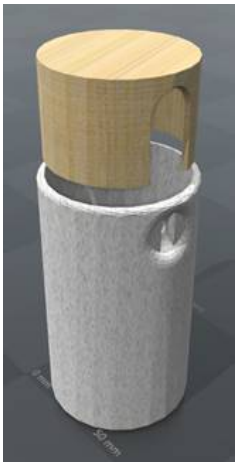


Abbildung 4.1: Früheres Design

4.2 Stärken und Schwächen des Teams

4.2.1 Planung

Unsere Planung konnten wir leider nicht komplett einhalten C. Da wir noch nicht viel Erfahrung mit der Planung von Projekten hatten, mussten wir diese mehrmals überarbeiten. Dennoch konnten wir unsere Ziele größtenteils zeitnah erreichen. Außerdem konnten wir aufgrund mangelnder Erfahrung noch nicht so gut einschätzen wie viel Zeit wir für die einzelnen Teile unseres Projekts brauchen.

4.2.2 Treffen

Da unser Projekt ziemlich groß ist, haben wir uns entschieden unser Projekt als Arbeitsgemeinschaft in der Schule anzumelden. Dadurch konnten wir uns einmal in der Woche in der Schule treffen, zusammen arbeiten und auch Probleme in der Gruppe lösen, von denen im Laufe der Zeit viele aufgetreten sind. Allerdings haben die Meetings einige von uns auch dazu veranlasst nicht mehr außerhalb dieser Treffen zu arbeiten und an zusätzlichen Treffen teilzunehmen. Dadurch ergab sich eine sehr ungleiche Verteilung der Arbeitsbelastung.

4.2.3 Kommunikation

Bei der Kommunikation in unserem Team gab es immer wieder massive Probleme. Es gab Diskussionen über wichtige Entscheidungen, wobei die Findung eines gemeinsamen Nenners oft mehr als eine Sitzung dauerte und deswegen viel Zeit verloren ging. Trotzdem haben sich einige von uns immer wieder zusammengesetzt und lange Diskussionen darüber geführt, welchen Weg wir gehen wollen. Wenn wir uns nach mehreren Treffen nicht entscheiden konnten, dann haben wir abgestimmt und die Mehrheit entscheiden lassen.

4.2.4 Zusammenarbeit

Im Laufe des Projekts gab es immer wieder Schwierigkeiten in der Zusammenarbeit, die sich hauptsächlich in unterschiedlichen Ansichten mehrerer Personen über die Mitarbeit in einem Team und Anwesenheit bei Treffen zeigten. Bei der Reaktion auf Anfragen nach fertiggestellten Arbeitspaketen und Texten für Berichte verhielt es sich ähnlich.

4.2.5 Ziel

Ein gemeinsames Ziel zu haben hat uns sehr viel geholfen. Obwohl wir sehr viel Freiheit in dem Wettbewerb hatten, konnten wir uns ziemlich schnell auf ein gemeinsames Ziel einigen. Mit diesem gemeinsamen Ziel vor den Augen konnten wir uns immer wieder zusammenreißen und versuchten Probleme innerhalb der Gruppe zu lösen.

4.2.6 Vorwissen

Unsere Gruppe hat bereits zu Beginn des Projektes viel Vorwissen mitgebracht. Jeder aus unserer Gruppe kann in Python und Java programmieren und viele von uns haben Erfahrung mit der Programmierung eines Arduino. Neben unseren Programmierkenntnissen haben uns auch die sehr guten Elektronikkenntnisse einzelner Teammitglieder bei vielen Problemen geholfen.

4.2.7 Fazit

Ein großer Teil unserer Gruppe hat, obwohl es viele Probleme gab, guten Zusammenhalt bewiesen und gemeinsam effektiv gearbeitet. Dies wollen wir auch in Zukunft so fortsetzten.
Wir haben im Laufe des Projektes viele neue Dinge, wie das die Entwicklung von Platinen in Eagle, die Entwicklung von 3D-Modellen in AutoCat, das Programmieren in C++, das Ätzen von Platinen, die Programmierung von Mikrocontrolern über einen Programmer, der Funktechnik und das verarbeiten der Daten durch spezielle Formeln, die wir umstellen mussten, erlernt. Zusätzlich ist das Wissen über die Elektrotechnik in jeglicher Form erweitert worden.

4.2.8 Änderungen

Aufgrund der aufgetretenen Schwierigkeiten bei der Zusammenarbeit, haben wir beschlossen die Zusammensetzung der einzelnen Gruppen zu ändern. Es wird in Zukunft jeweils eine Gruppe für die Bodenstation, die Hardware, die Software der Mikrocontroller, die Hülle und die Öffentlichkeitsarbeit geben. Durch diese Verteilung sind jedem Teammitglied weniger Bereiche zugeordnet, sodass die einzelnen Personen nicht mehr in einen Konflikt zwischen der Arbeit an verschiedenen Bereichen geraten.

4.3 Aktueller Stand

4.3.1 Hardware

Nachdem alle Sensoren auf unserem Testsystem funktionierten, haben wir einen Prototyp der Platine B.10 B.11 für das Primärmodul designend und bestellt. Es ging dabei eher um die Machbarkeit einer Platinenherstellung. Nachdem der Prototyp annähern perfekt funktioniert hat, haben wir noch einige Änderungen vorgenommen (neue Spannungsversorgung, Wechsel des GPS-Moduls und Fräsung von Löchern zur späteren Befestigung). Außerdem haben wir natürlich eine Platine für das Sekundärmodul entwickelt. Beide sind im Moment in der Fertigung und kommen etwa eine Woche vor Abgabe bei uns an. Da wir 10 Platinen bestellt haben, das Gehäuse

beliebig oft drucken können und genug Komponenten haben, werden wir zwei vollständige Satelliten bauen. Das zeigt außerdem, wie leicht das Primärmodul zu bauen ist, um es mit beliebigen Sekundärmodulen zu verwenden.

4.3.2 Software

Da der Hardware-Prototyp des Primärmoduls bis auf die GPS-Daten voll funktionsfähig ist, können wir unsere Software schon jetzt ausgiebig testen und erweitern. Die Sekundärmission können wir dank unserer Testsysteme auch bereits jetzt im Zusammenspiel mit der Software testen.

4.3.3 Hülle

Die Hülle ist fertig entwickelt. Sie wird jetzt gerade gedruckt. Kleinere Anpassungen könnten noch notwendig werden.

4.4 Unterstützung

Bei der Entwicklung und dem Bau unseres Satelliten wurden wir von vielen Seiten unterstützt. Insbesondere im Hackerspace Bremen haben wir viel Unterstützung erfahren. So durften wir die dortigen Räumlichkeiten und Werkzeuge verwenden und haben von den Mitgliedern viele Tipps bekommen. Weitere Unterstützung haben wir insbesondere durch die Lehrer erhalten, die an unserem Projekt eigentlich nicht beteiligt waren. Diese bezog sich insbesondere auf die Softwareentwicklung und die Elektrotechnik. Außerdem wurden wir von Dr. Manfred Gronak, einem Amateurfunker unterstützt, der uns insbesondere im Bereich der Funktechnik sehr geholfen hat. Unser Fallschirm wurde von Petra Allen genäht. Fenya Haack hat unser neues Logo designed. Wir bedanken uns bei allen Personen, die uns im Laufe des Projekts unterstützt haben.

Kapitel 5

Kostenplanung

5.1 Sponsoren

Wir haben für den CanSat Wettbewerb 2017 zwei Sponsoren gefunden.

5.1.1 b.r.m.

Unser erster Sponsor ist der regional in Bremen verankerte IT-Dienstleister b.r.m. Dieser unterstützt das Team Starbugs mit einem Geldbetrag von 200€ und stellt uns eine Möglichkeit zum Testen des Satelliten auf einem Flugplatz in Oldenburg-Hatten zur Verfügung.

5.1.2 Söffge

Unser zweiter Sponsor für die Teilnahme am Deutschen CanSat-Wettbewerb ist der in Bremen und Niedersachsen tätige Familienbetrieb für Gebäudereinigung Söffge, welcher uns bei der Entwicklung unseres Satelliten ebenfalls mit einem Betrag von 200€ unterstützt.

5.1.3 Freiplan Ingenieure

Unsere Teilnahme am Deutschen CanSat-Wettbewerb wird außerdem von dem Ingenieurbüro für Haustechnik Freiplan Ingenieure mit einem Betrag von 200€ unterstützt.

5.1.4 Bremer Rechenzentrum

Unser vierter Sponsor ist der bundesweit agierende Entgeltabrechnungs-Dienstleister Bremer Rechenzentrum GmbH der uns bei der Entwicklung unseres Satelliten ebenfalls mit 200€ unterstützt.

5.1.5 Watterott

Durch den von Watterott gesponsorten Gutschein im Wert von 200€, den jedes am Deutsche CanSat-Wettbewerb teilnehmendes Team erhält, konnten wir einige wichtige Bauteile erwerben.

Leider haben sich alle Sponsoren, die wir im Zwischenbericht erwähnt haben nicht mehr gemeldet oder uns eine Absage zugeschickt. Dies trifft auch auf diejenigen zu, die uns zuvor bereits eine mündliche Zusage erteilt hatten.

Name	Preis in €	Zusatzkosten in €	Anzahl	Gesamtpreis in €
Infrarot Thermometer	21,58	3,50	2	44,66
Beschleunigungssensor	3,50		2	9,00
ATMEGA328p-PU	3,81		5	19,05
Micro-SD-Kartenadapter	17,85		2	35,70
Atmel ICE	64,00	3,50	1	67,50
Gesamtsumme				**177,91**

Tabelle 5.1: Ausgaben Watterott

5.2 Ausgaben

Einige Bauteile wurden doppelt bestellt, da wir diese in der Primär- und Sekundärmission verwenden wollen. Desweiteren sind einige Bauteile auch beschädigt wurden und mussten neu bestellt werden. Manche Bauteile haben sich als nicht für unsere Zwecke verwendbar herausgestellt. Zudem waren manche Bauteile zu groß.

Name	Preis in €	Zusatzkosten in €	Anzahl	Gesamtpreis in €
GPS-Modul A2235-H	17,53	9,99	3	62,58
Reed-Sensor Schalter	3,24	7,45	3	17,17
LiPo 3,7V Akku 500mAh	7,90		2	15,80
Luftdrucksensor BMP180	5,95	3,50	2	15,40
10 MHz Quarz	1,00	1,50	5	6,50
LiPro Balanced Charger	40,00	13,33	1	53,33
10 MHz Quarz SMD	0,68		3	2,04
10 MHz Quarz	1,69		2	3,38
Kondensatoren 22pF	1,00	1,50	3	4,50
Spannungswandler 3,3V	0,23		5	1,15
GPS-Modul mit Antenne	11,66		1	11,66
Micro-SD-Kartenadapter	12,32		4	49,28
Beschleunigungssensor	6,59	3,50	1	10,09
Pololu 3,3V Step-Down Spannungswandler	4,49	3,50	3	16,97
906-960MHz Yagi Antenne	14,99		1	14,99
HC-12 SI4463 433 MHz Transceiver	7,95	-0,08	3	23,77
GPS-Modul mit Antenne	11,99		1	11,99
ATMEGA2560V	10,00		2	20,00
HM-TRP 868 MHz Transceiver	9,80	0,90	3	30,30
Schlüsselschalter	5,82	5,95	2	17,59
LiPo 3,7V Akku 500mAh	4,69		4	18,76
50 LED SMD 0805 Format Packung	1,79	1,09	1	2,88
Platinen Bestellung Prototyp	36,00		1	36,00
Platinen Bestellung Finale Version	30,00		1	30,00
Antennen Adapter	6,99		1	6,99
Koaxialkabel	9,00		1	9,00
ATMEGA2560V	15,20	5,95	2	36,35
Magnetischer Summer	0,93		2	1,86
Schlüsselschalter	5,82		2	11,64
ATMEGA328p-PU	3,32		1	3,32
Spannungswandler 5V	0,68		1	0,68
Kondensatoren 22pF	0,21		4	0,84
Pico-loc Wire to Board	12,50	8,27	1	20,77
Molex Stiftleiste	3,56		1	3,56
Buchsenkontakt	1,40	8,27	1	1,40
RFM69HW	30,56	8,27	1	38,83
8 MHz Quarz	0,50		2	1,00
Gesamtsumme				**613,18**

Tabelle 5.2: Ausgaben

	Herkunft	Summe in €
-	Ausgaben	791,09
+	b.r.m.	200,00
+	Söffge	200,00
+	Watterott	177,91
+	Schule	213,18
	Bilanz	**0,00**

Tabelle 5.3: Bilanz

Name	Preis in €	Anzahl	Gesamtpreis in €
Infrarot Thermometer	21,58	1	21,58
Beschleunigungssensor	3,50	1	3,50
ATMEGA2560V	10,00	1	10,00
ATMEGA328p-PU	3,81	1	3,81
Micro-SD-Kartenadapter	17,85	2	35,70
8MHz Quartz	0,50	2	1,00
LiPo 3,7V Akku 500mAh	4,69	4	18,76
Schlüsselschalter	5,82	2	11,64
HM-TRP 868 MHz Transceiver	9,80	1	9,80
GPS-Modul mit Antenne	11,99	1	11,99
Pico-loc Wire to Board	12,50	1	12,50
Molex Stiftleiste	3,56	1	3,56
Buchsenkontakt	1,40	1	1,40
Magnetischer Summer	0,93	1	0,93
50 LED SMD 0805 Format Packung	1,79	1	1,79
HC-12 SI4463 433 MHz Transceiver	7,95	1	7,95
Platinen Bestellung Finale Version	30,00	1	30,00
Pololu 3,3V Step-Down Spannungswandler	4,49	2	12,48
Kondensatoren 22pF	1,00	2	2,00
Koaxialkabel	9,00	1	9,00
Luftdrucksensor BMP180	5,95	1	5,95
Geschätzte Fallschirmkosten	10,00	1	10,00
Gesamtsumme			**225,34**

Tabelle 5.4: Wert des Satelliten

Kapitel 6

Öffentlichkeitsarbeit

Unsere Öffentlichkeitsarbeit wird auf unterschiedlichen Multi-Media-Platformen für den CanSat Wettbewerb betrieben. Dazu gehören Facebook, Youtube, Instagram und unsere Website. Außerdem schreiben wir für die Webseite heise.de mehrere Beiträge die monatlich veröffentlicht werden sollten.

- Website: https://teamstarbugs.wordpress.com/
- Facebook: https://www.facebook.com/TeamStarbugs/
- Instagram: https://www.instagram.com/teamstarbugs/

6.1 Website

Für den CanSat Wettbewerb haben wir eine Webseite erstellt, auf der wir unseren Satelliten und das Projekt der Öffentlichkeit näher bringen wollen. Wir haben uns hierbei für Wordpress entschieden damit unsere Webseite auch nach dem CanSat Wettbewerb 2017 online bleiben kann, ohne dass wir dafür bezahlen müssen. Ein weiterer Vorteil von Wordpress ist, dass mehrere Benutzer auf die Webseite zugreifen und Beiträge veröffentlichen können. Außerdem unterstützt Wordpress verschiedenste Designs. Wir haben bisher 210 Besucher erreicht.

6.2 Facebook

Ein weitere Idee zur Öffentlichkeitsarbeit war es, eine Facebookseite für unser Team zu erstellen. Auf dieser werden regelmäßig kleine Beiträge hochgeladen, um auch anderen Interessenten den Verlauf des Wettbewerbs nahe zu bringen.

6.3 Instagram

Da Instagram als Social Media Plattform immer mehr an Bedeutung gewinnt, entschieden wir uns dazu auch einen eigenen Instagram Account zu erstellen. Unter @teamstarbugs werden kleine Bild- und Videobeiträge hochgeladen um das Interesse am Wettbewerb zu steigern. Hier haben wir 27 Abonnenten erreicht, welche unsere Beiträge sehen und oft auch Liken.

6.4 Tagebuch

Wir schreiben für heise online jeden Monat einen Beitrag, der auf dieser Webseite veröffentlicht werden sollte. Dazu gehören Texte, Bilder und auch Videos. Wir sind von der Maker Faire auf das Tagebuch aufmerksam

gemacht geworden und haben dies als gute Möglichkeit für eine besonderen Öffentlichkeitsarbeit für den CanSat-Wettbewerb gesehen. Die Text wurden leider noch nicht veröffentlicht. Der Grund hierfür ist uns unbekannt.

6.5 Maker Faire

Wir sind durch eine E-Mail auf die Messe aufmerksam geworden und fanden die Idee unser Projekt auf einer Messe zu präsentieren sehr interessant und meldeten uns kurz darauf bei Herrn Steffan (Community-Manager der Maker Faires) um uns weiter zu informieren. Wir wollten uns für die Messe in Hannover anmelden, jedoch musste wir die Messe aufgrund vieler schulischer Aufgaben absagen.

Literaturverzeichnis

[1] Barometrische höhenformel. https://de.wikipedia.org/wiki/Barometrische_H%C3%B6henformel.

[2] Barometrische höhenformel rechner. https://rechneronline.de/barometer/.

[3] millis. https://www.arduino.cc/en/reference/millis.

[4] Sparkfun openlog. https://www.sparkfun.com/products/13712.

[5] atmel. *Atmel ATmega*, 2014.

[6] atmel. *ATmega Datasheet*, 2015.

[7] atmel. *ATmega328/P - datasheet summary*, 2016.

[8] bosch. *BMP180 - Digital pressure sensor*, 2013.

[9] Ekbert Hering, Rolf Martin, and Martin Stohrer. *Physik für Ingenieure*. VDI-Verlag GmbH, 1992.

[10] hoperf electronic. *HM-TRP Series 100mW Transceiver modules V1.0*, 2006.

[11] InvenSense. *MPU-6000 and MPU-6050 - Product Specification*, 2012.

[12] Melexis. *MLX90614 family*, 2015.

[13] Silicon Labs. *HC-12 Wireless Serial Port Communication Module*, 2012.

[14] u-blox. *NEO-6u-blox 6 GPS Modules Data Sheet*, 2011.

Anhang A

Technische Anforderungen

1. Die Hülle 3.1.1 unseres Satelliten hat eine Höhe von 115mm und einen Durchmesser von 67mm.
2. Der Fallschirm wird auf die Hülle gelegt und überragt diese dadurch nur wenig 3.1.2. Die verwendeten Antennen überragen den Satelliten ebenfalls nur unerheblich.
3. Unser Satellit wird mit Gewichten versehen, um das nötige Gewicht zu erreichen 3.1.1.
4. Wir verwenden keine der genannten Materialien.
5. Unser Satellit kann durch seine Akkus über 6,8 Stunden mit Strom versorgt werden 3.2.1.
6. Unsere Akkus können leicht ausgetauscht werden 3.1.3.
7. Der Satellit verfügt über einen leicht erreichbaren Hauptschalter 3.2.1.
8. Unser Satellit enthält ein GPS-Modul 3.2.2 dessen Koordinaten zur Bodenstation gesendet werden. Außerdem haben wir einen Buzzer 3.2.2 eingebaut.
9. Unser Satellit verfügt über einen pinken Fallschirm B.2.
10. Das Bergungssystem entspricht den von uns erwarteten Belastungen 3.1.2.
11. Unser Fallschirm 3.1.2 ist für eine Fallgeschwindigkeit von $15 m/s^2$ ausgelegt.
12. Unser Satellit hält einer Beschleunigung von 20g stand, da wir weitestgehend auf Kabelverbindungen verzichtet haben.
13. Der Wert 5.4 unseres Satelliten beträgt 225,34€.
14. Den Wert des Fallschirms haben wir mit 10€geschätzt.
15. Wir verwenden 869 MHz und 433,4MHz mit Frequenzmodulation.
16. Wir werden unseren Satelliten bei der technischen Abnahme am Dienstag, den 26.09.17 abgeben und danach keine Demontage mehr vornehmen.
17. Unser Satellit misst Luftdruck 3.2.2 und Temperatur 3.2.2.
18. Unsere Sekundärmission verstößt nicht gegen geltendes Recht.
19. Wir führen keine Fly-Home-Mission durch.

Anhang B

Bilder

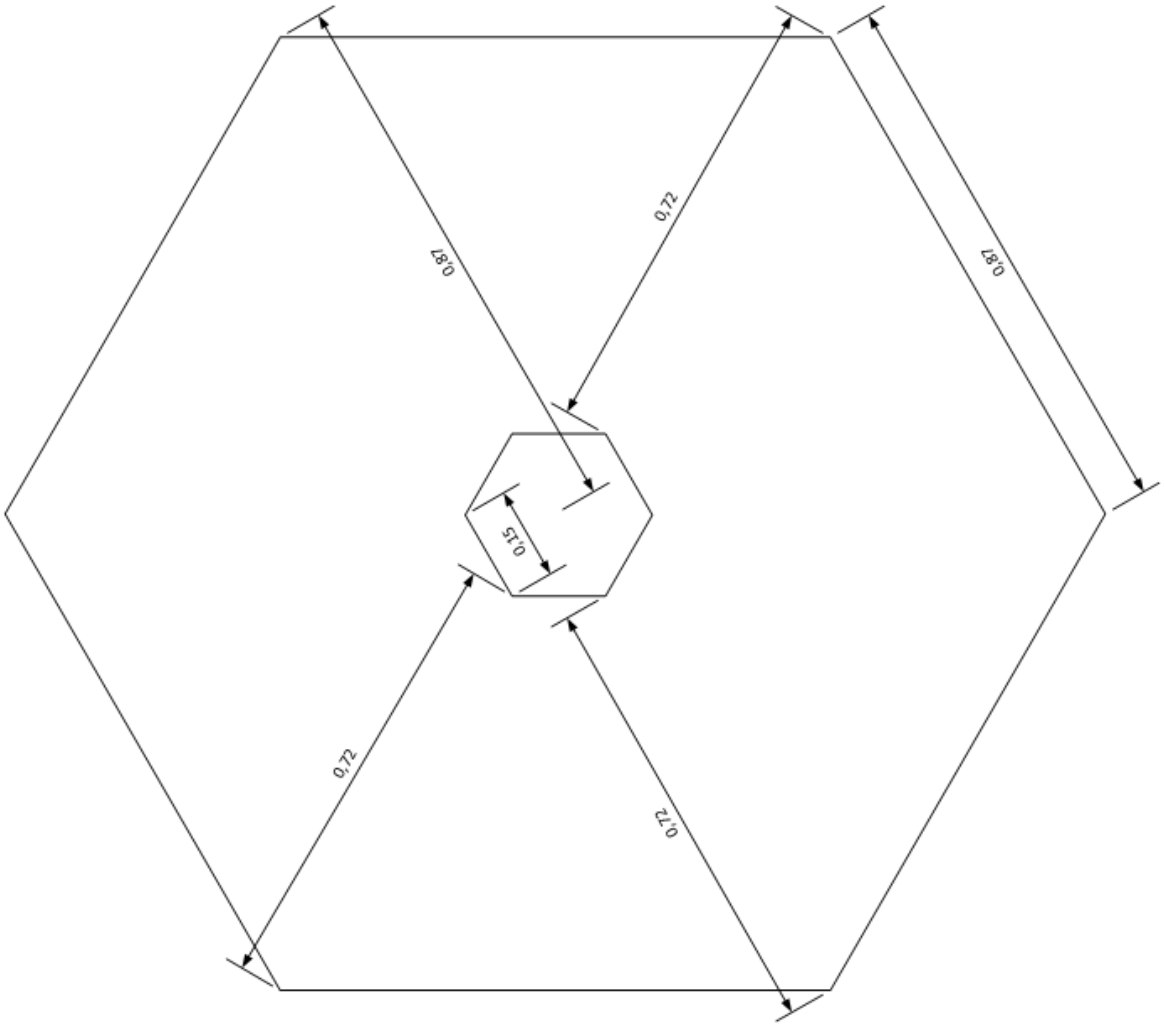


Abbildung B.1: Fallschirm

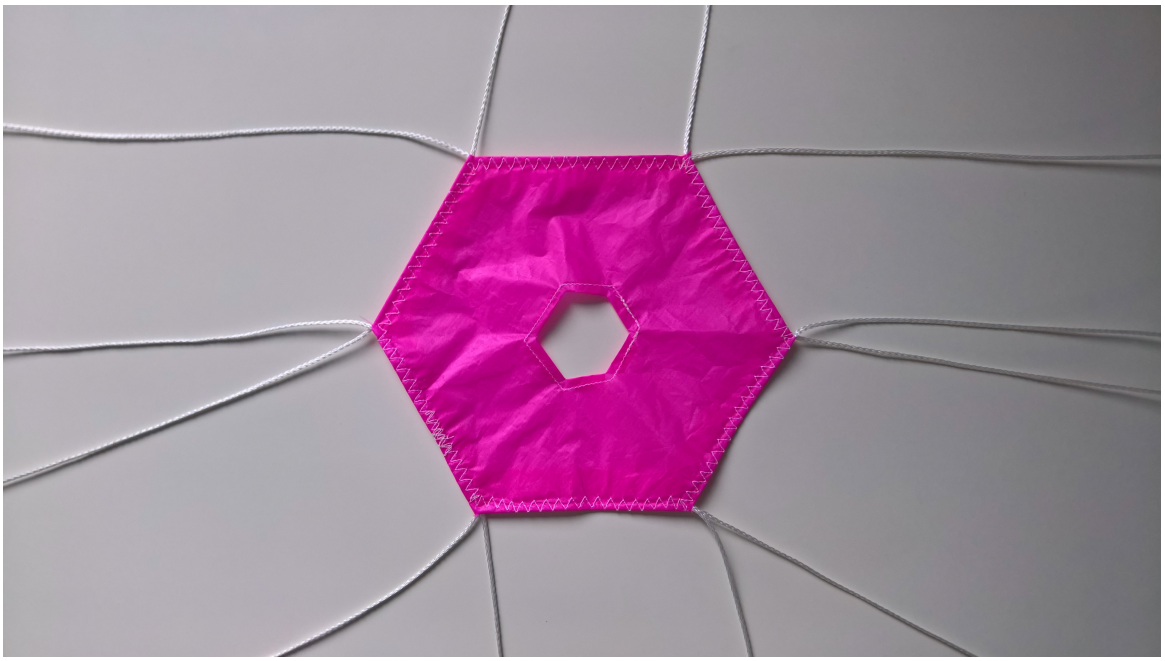


Abbildung B.2: Fallschirm

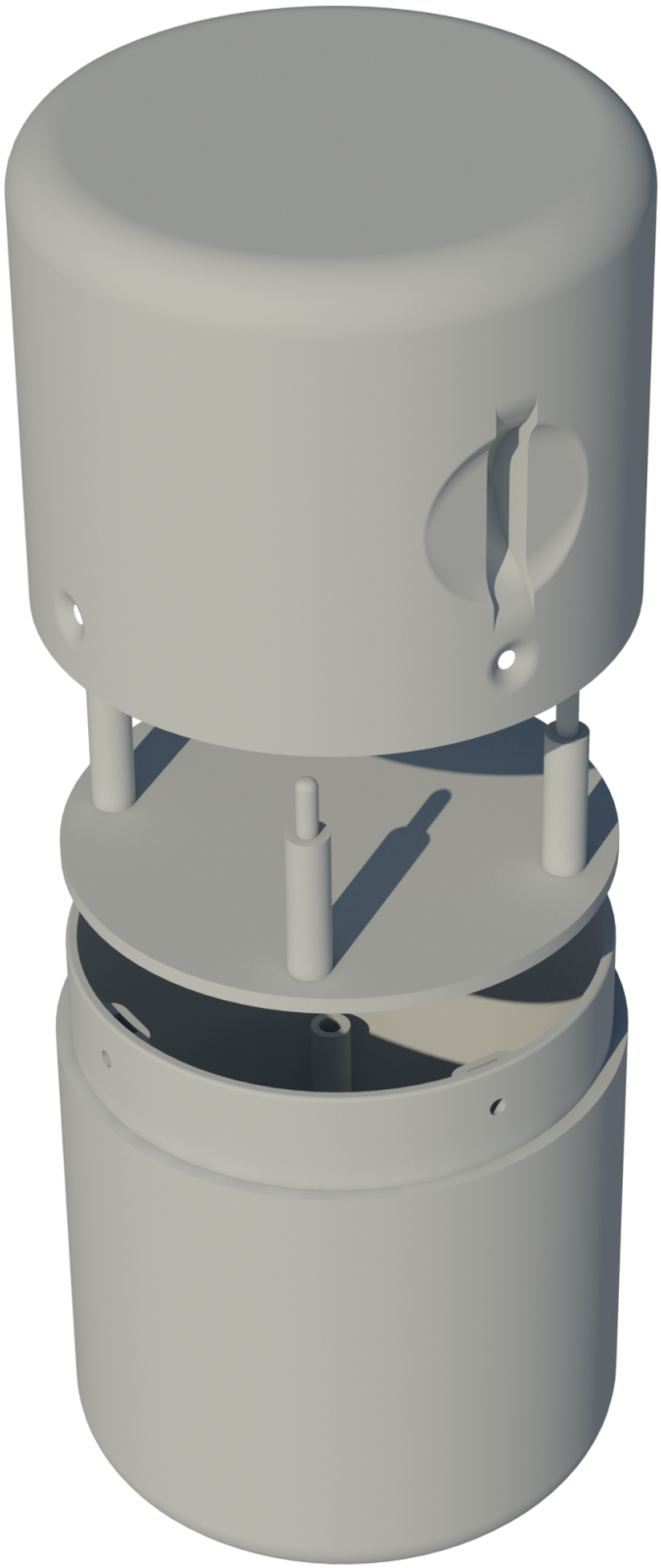


Abbildung B.3: Hülle

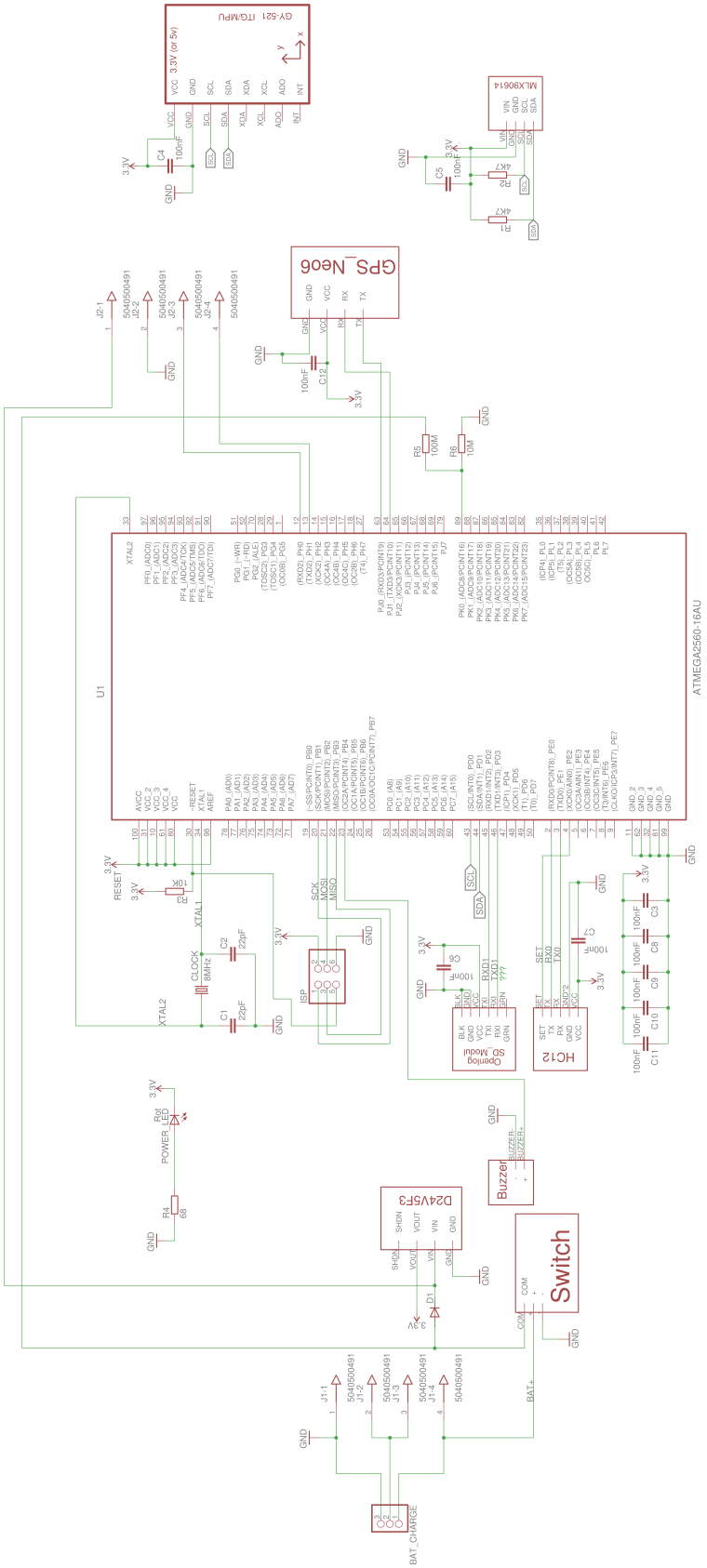


Abbildung B.4: Schematische Darstellung der Platine des Primärmoduls

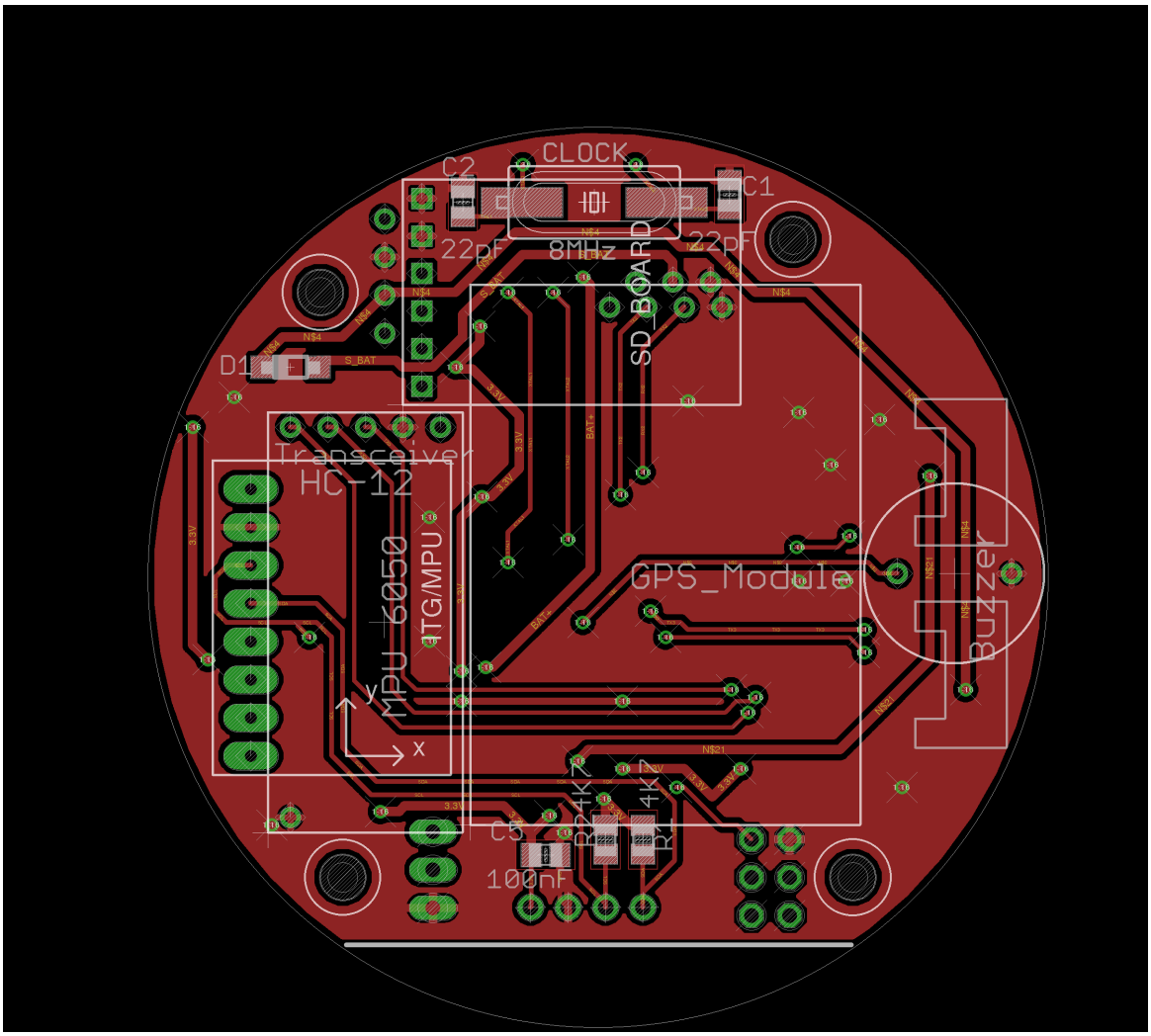


Abbildung B.5: Oberseite der Platine des Primärmoduls

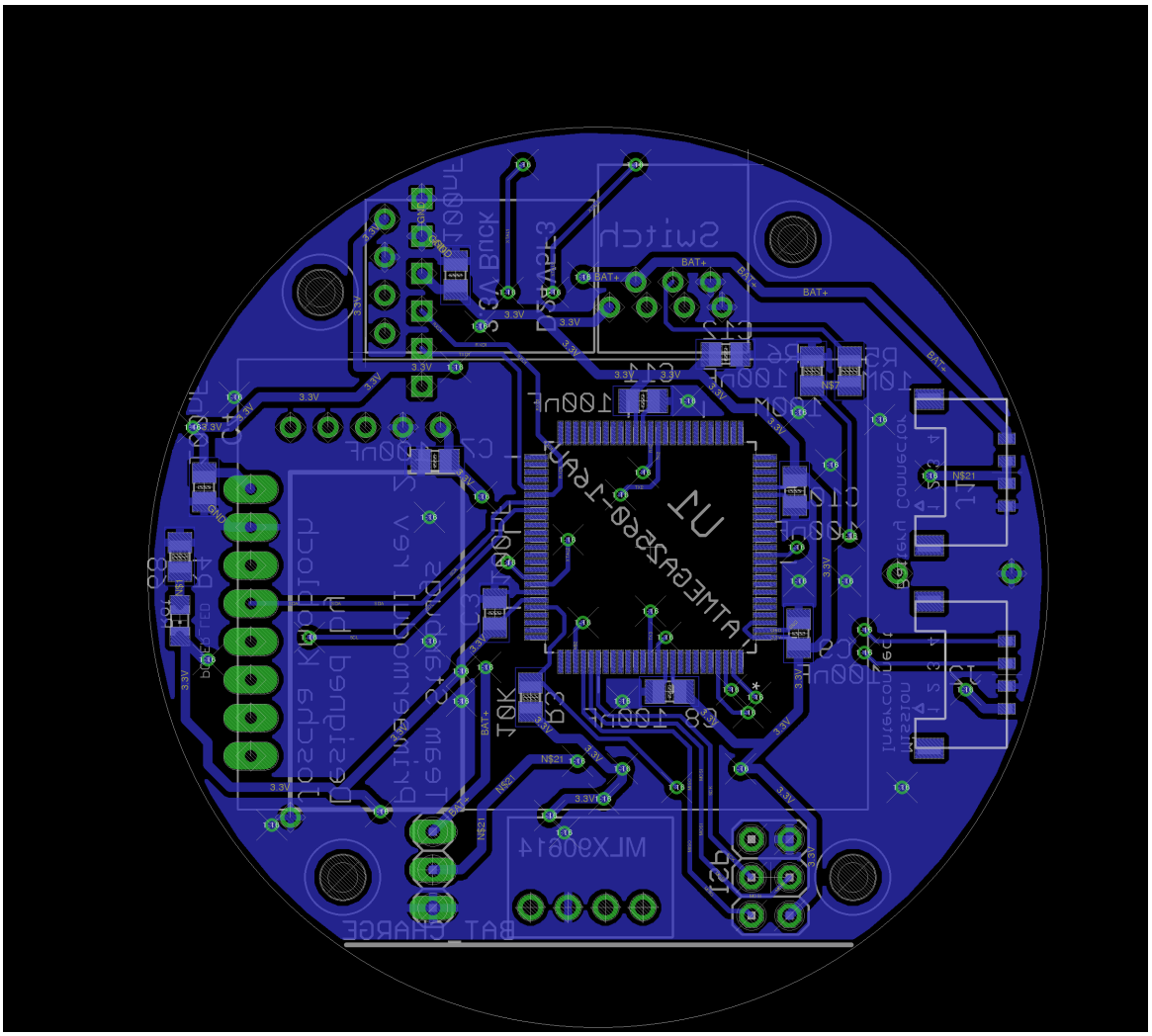


Abbildung B.6: Unterseite der Platine des Primärmoduls

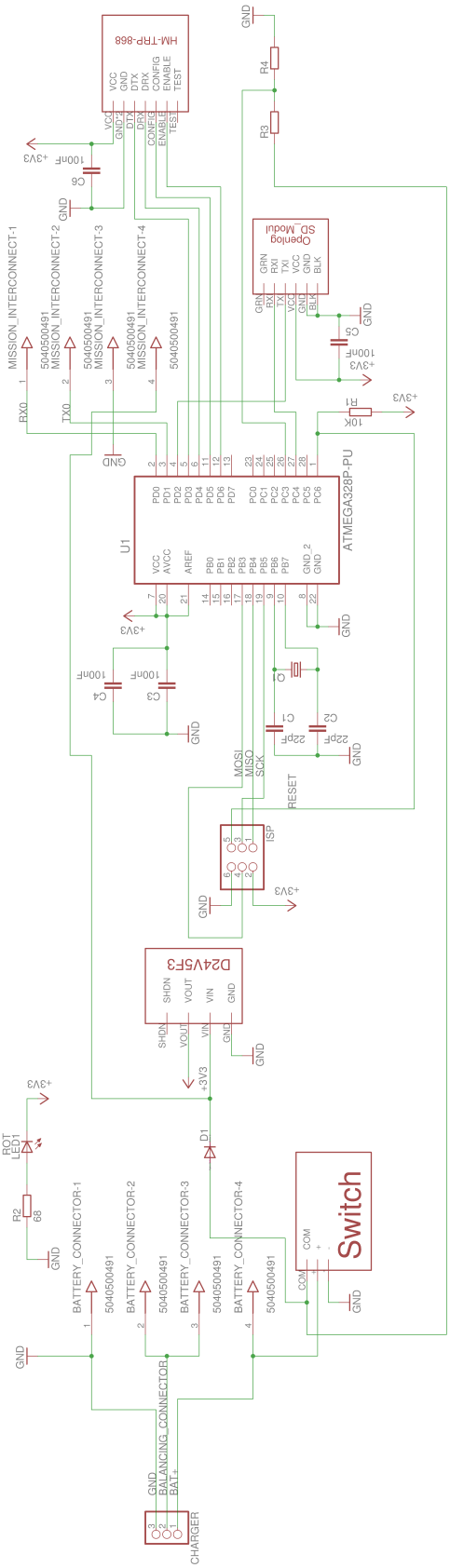


Abbildung B.7: Schematische Darstellung der Platine des Sekundärmoduls

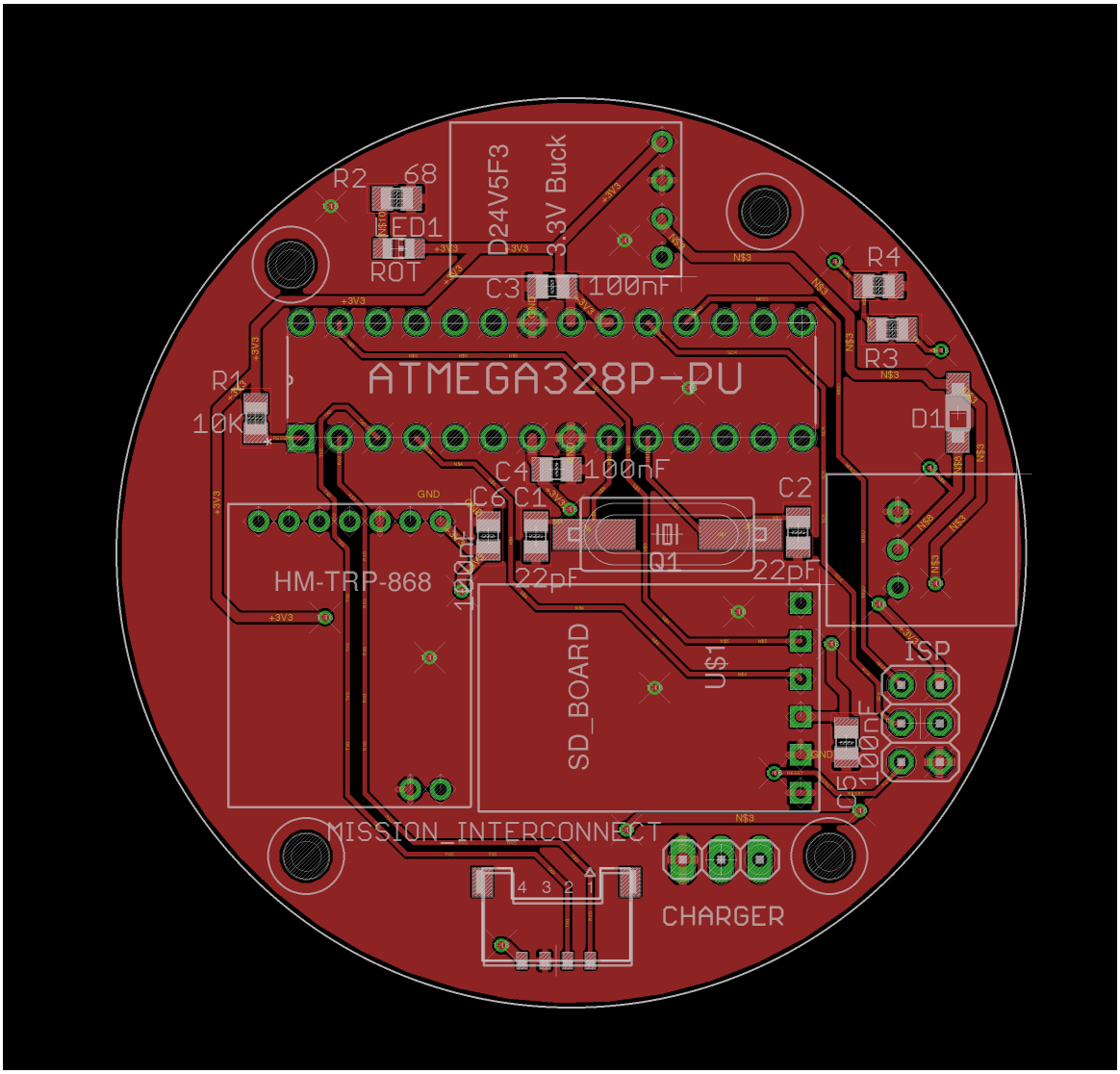


Abbildung B.8: Oberseite der Platine des Sekundärmoduls

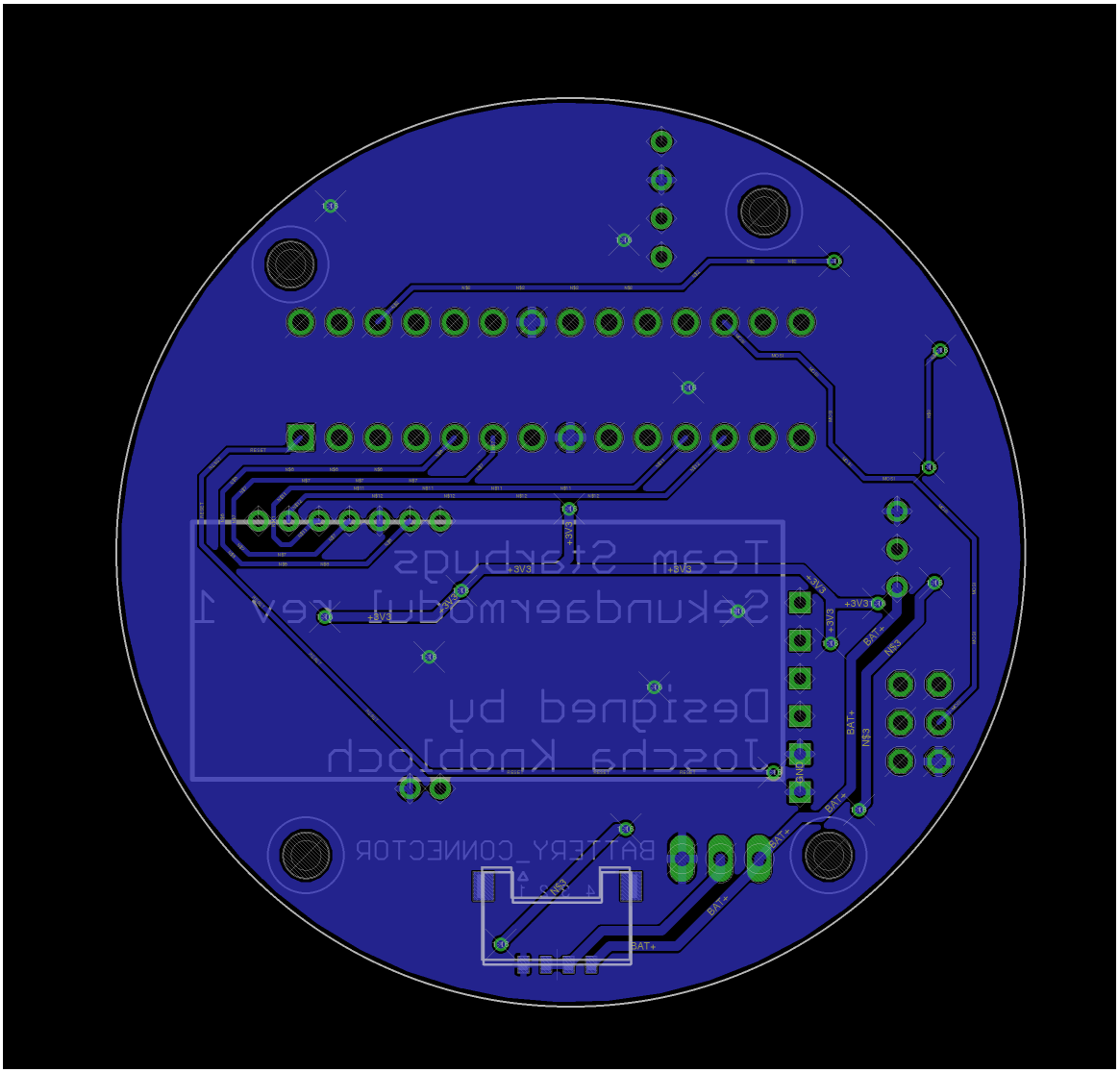


Abbildung B.9: Unterseite der Platine des Sekundärmoduls

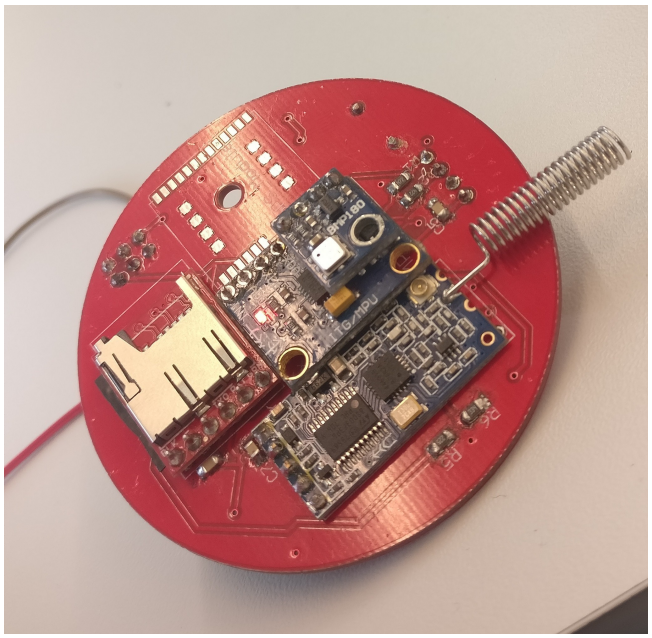


Abbildung B.10: Platine Prototyp

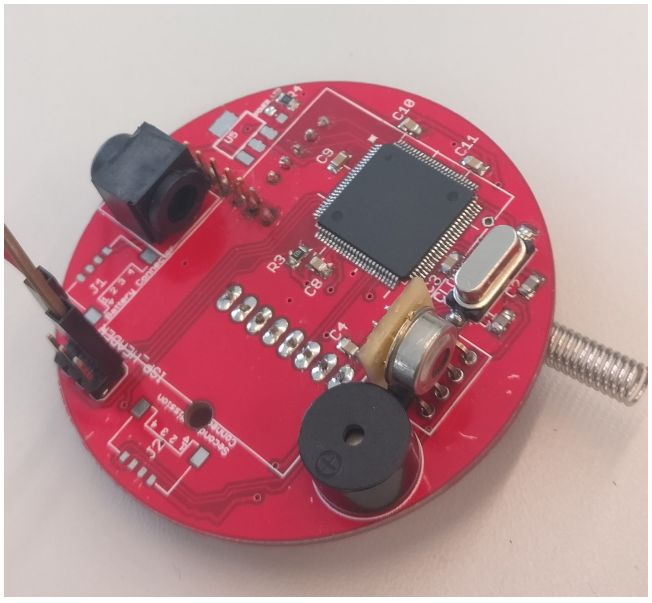


Abbildung B.11: Platine Prototyp

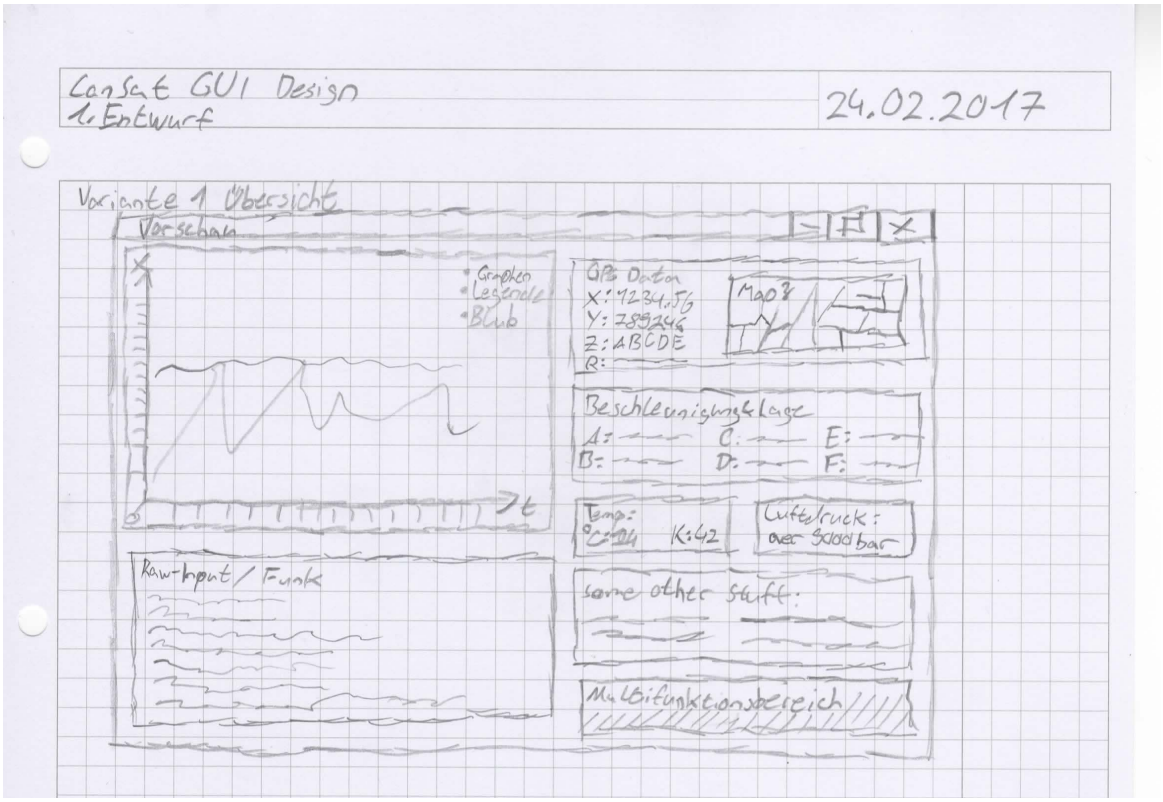


Abbildung B.12: Receive Client GUI 1. Entwurf

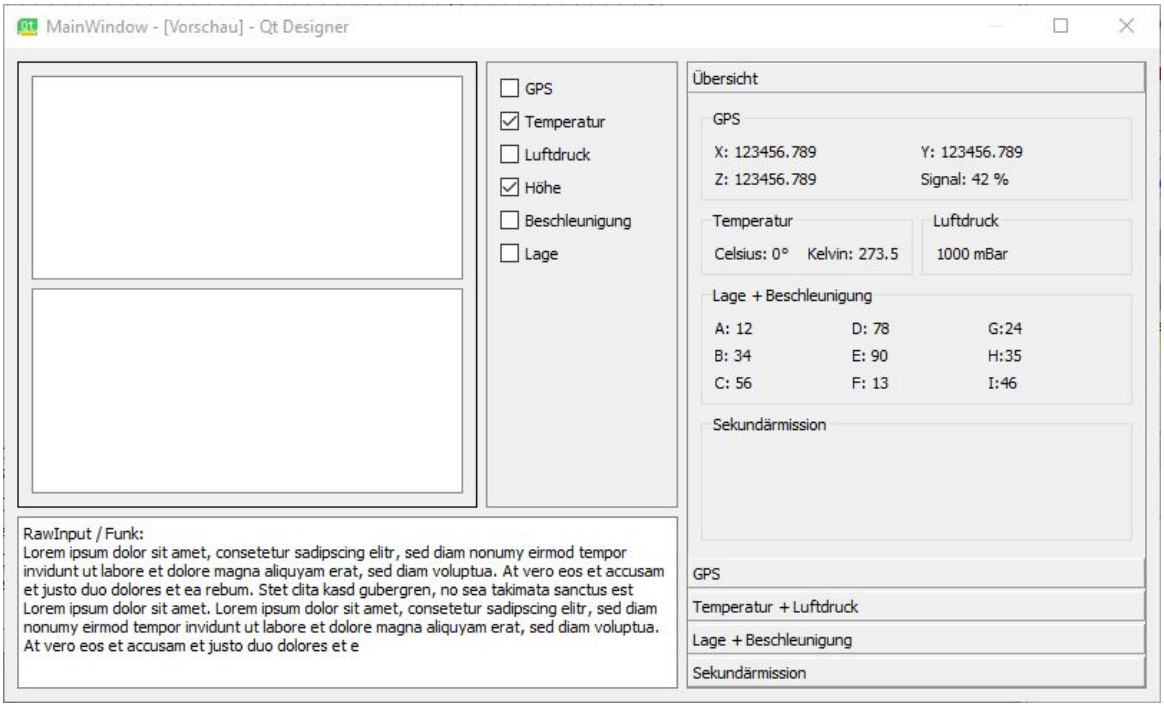


Abbildung B.13: Receive Client GUI 3. Entwurf

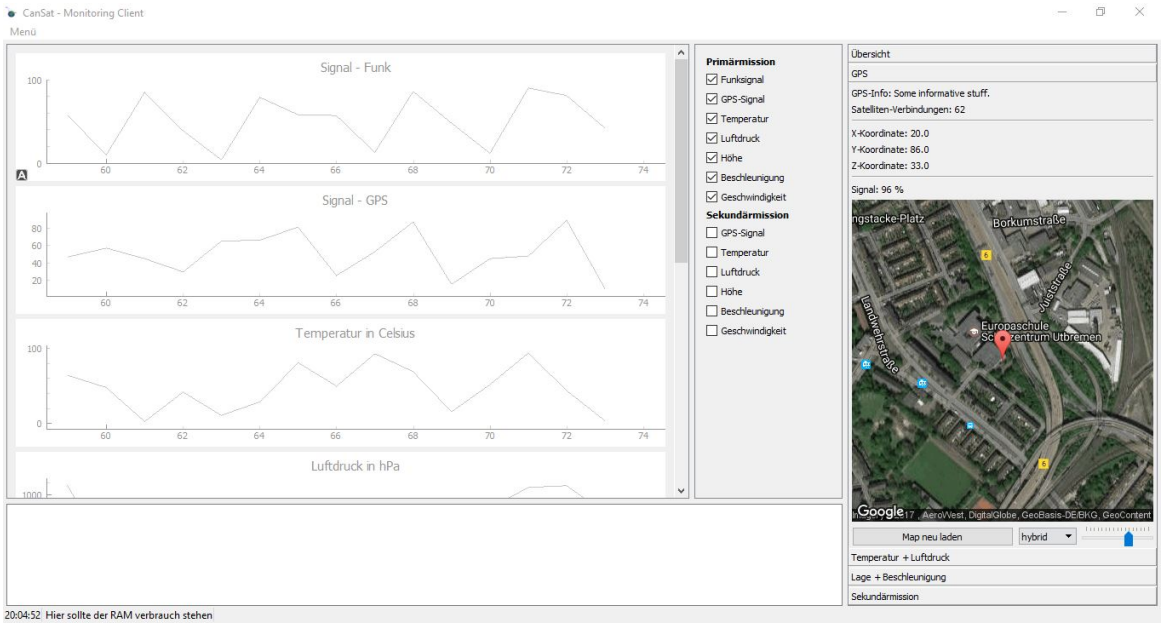


Abbildung B.14: Receive Client GUI

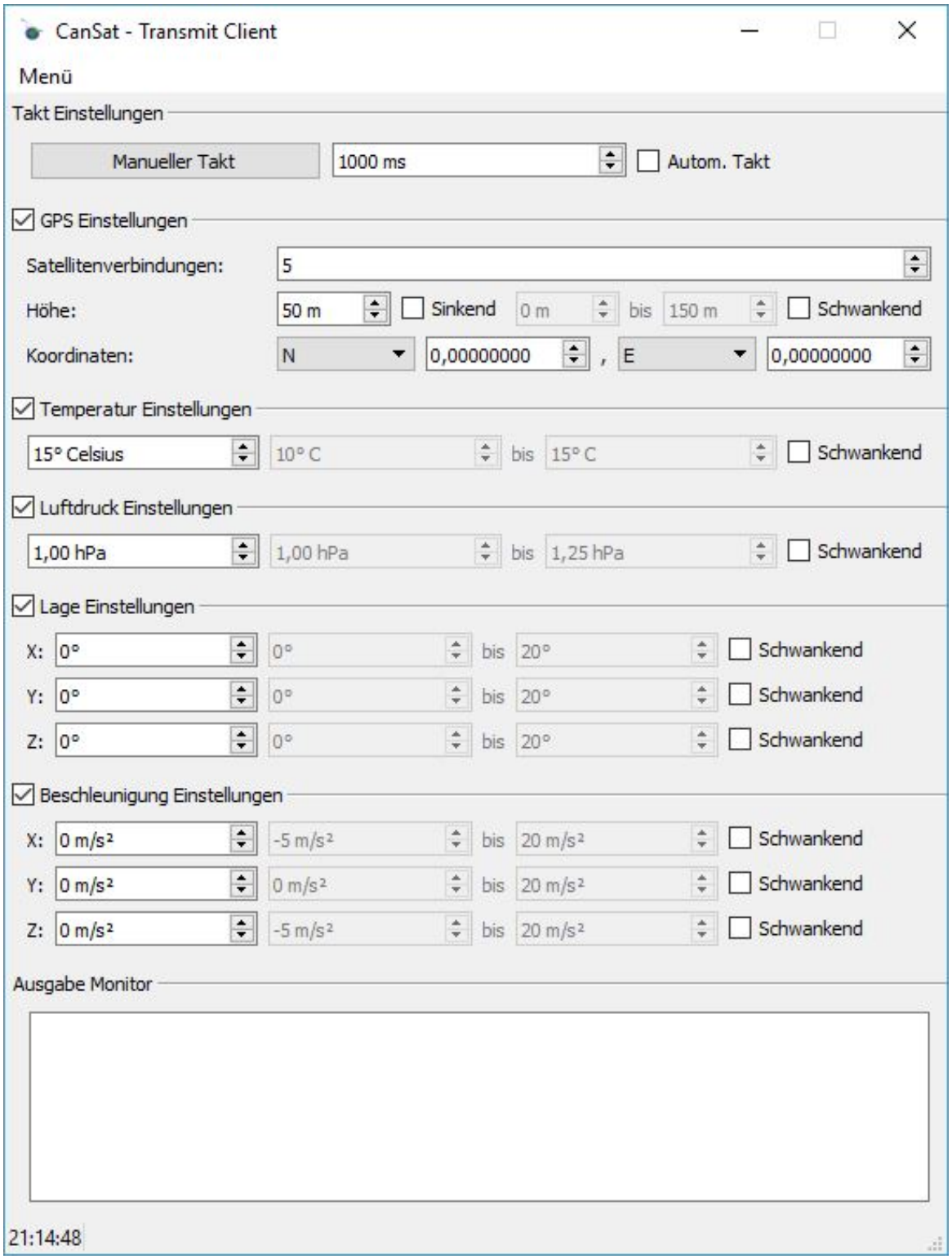


Abbildung B.15: Transmit Client GUI

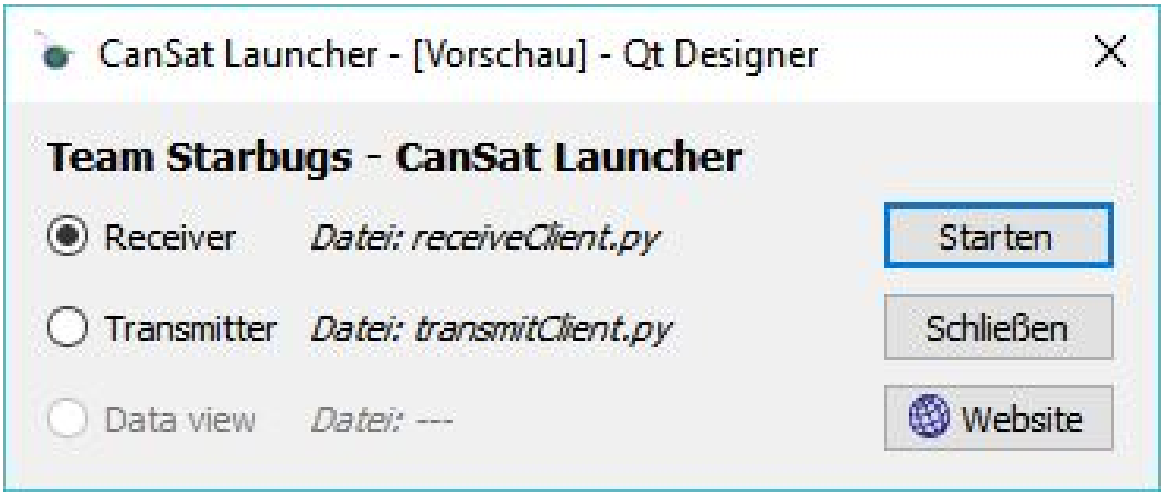


Abbildung B.16: Launcher GUI

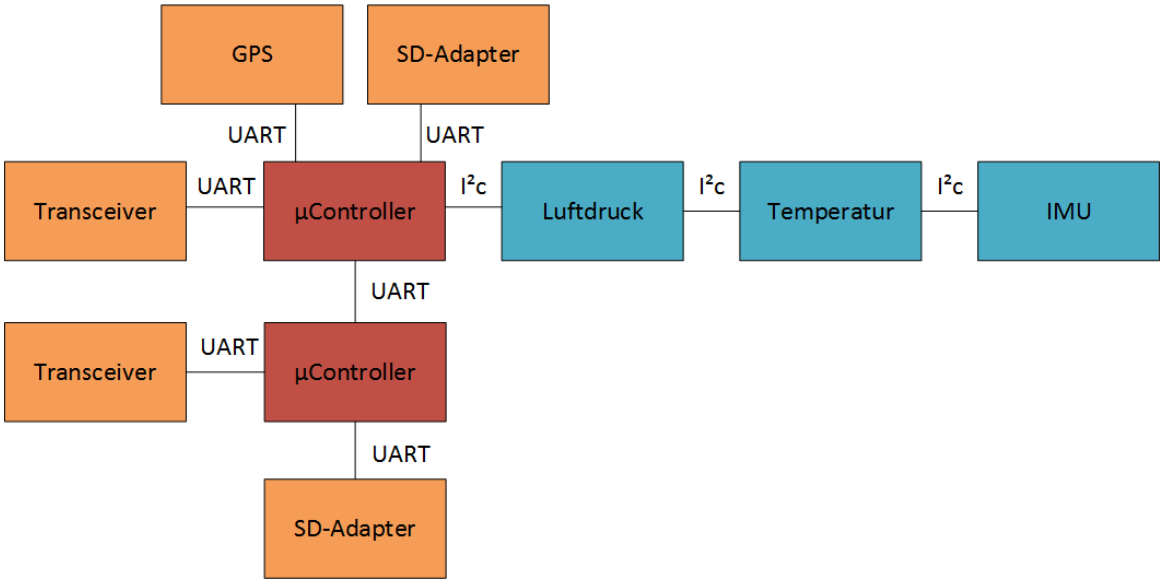
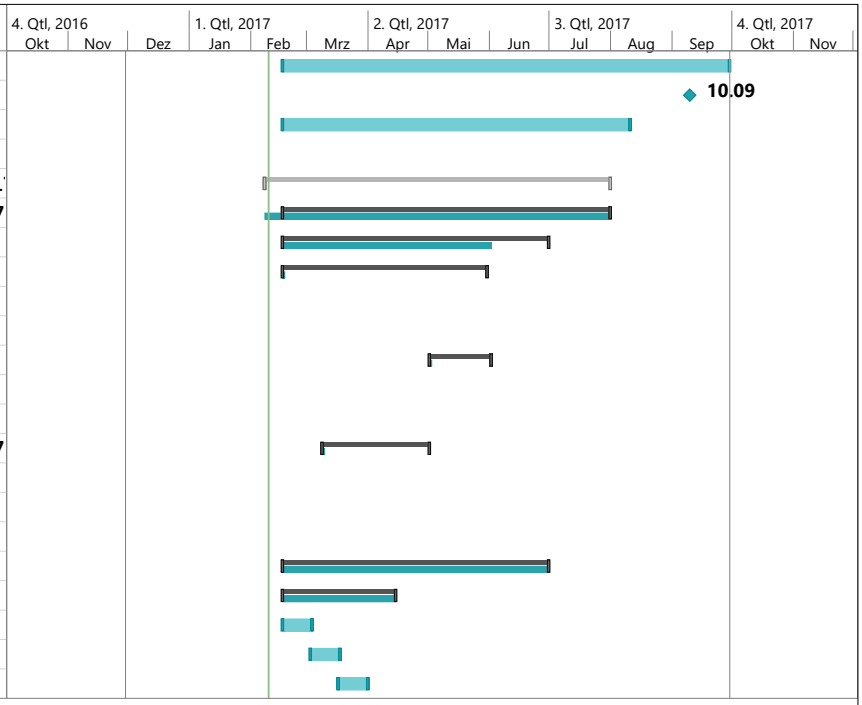


Abbildung B.17: Die Komponenten und Datenverbindungen im Blockdiagramm

Anhang C

Gantt-Diagramm

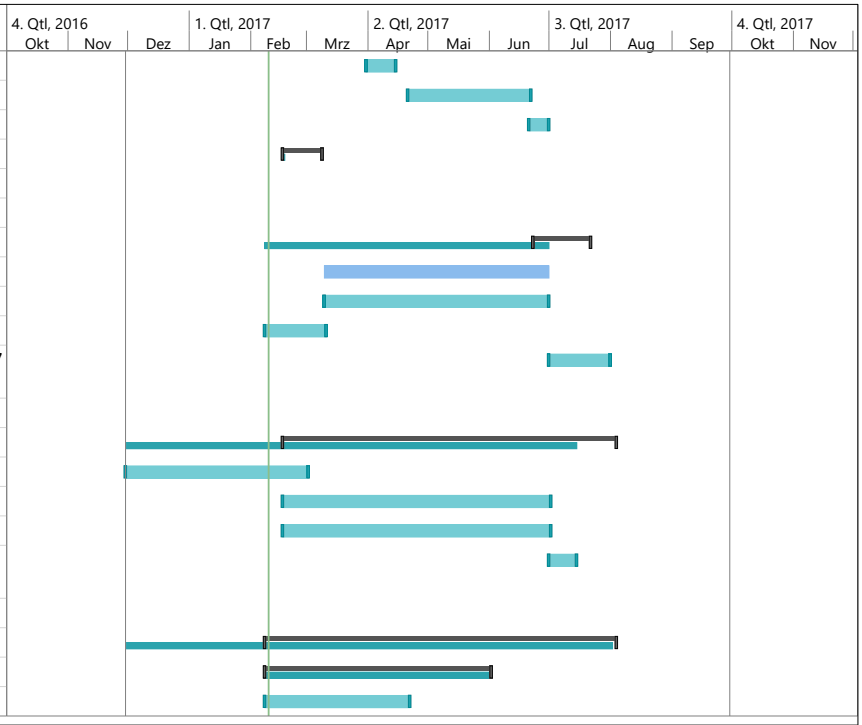
Nr.	Vorgangsname	Dauer	Anfang	Ende
1	CanSat	161 Tage	Fre 17.02.17	Fre 29.09.17
2	Projektabschluss	0 Tage	Son 10.09.17	Son 10.09.17
3	Entwicklung	125 Tage	Fre 17.02.17	Don 10.08.17
4				
5	**Bodenstation**	**124 Tage**	**Mit 08.02.17**	**Mon 31.07.17**
1	**Bodenstation**	**117 Tage**	**Fre 17.02.17**	**Mon 31.07.17**
2	**Datenübertragur**	**96 Tage**	**Fre 17.02.17**	**Fre 30.06.17**
3	**Satellit**	**73 Tage**	**Fre 17.02.17**	**Die 30.05.17**
4	Senden			
5	Empfangen			
6	**Netzwerk**	**23 Tage**	**Die 02.05.17**	**Don 01.06.17**
7	Senden			
8	Empfangen			
9	**Datenmanageme**	**38 Tage**	**Don 09.03.17**	**Mon 01.05.17**
10	Senden			
11	Empfangen			
12	Clientverwaltu			
13	**Datenverarbeitu**	**96 Tage**	**Fre 17.02.17**	**Fre 30.06.17**
14	**Eigener Satelli**	**41 Tage**	**Fre 17.02.17**	**Fre 14.04.17**
15	GPS	11 Tage	Fre 17.02.17	Fre 03.03.17
16	Relative Pos	11 Tage	Fre 03.03.17	Fre 17.03.17
17	Luftdruck	11 Tage	Fre 17.03.17	Fre 31.03.17



Projekt: Planung Neu
Datum: Fre 10.02.17

Vorgang	Inaktiver Sammelvorgang	Externe Vorgänge
Unterbrechung	Manueller Vorgang	Externer Meilenstein
Meilenstein	Nur Dauer	Stichtag
Sammelvorgang	Manueller Sammelrollup	In Arbeit
Projektsammelvorgang	Manueller Sammelvorgang	Manueller Fortschritt
Inaktiver Vorgang	Nur Anfang	
Inaktiver Meilenstein	Nur Ende	

Nr.	Vorgangsname	Dauer	Anfang	Ende
18	Temperatur	11 Tage	Fre 31.03.17	Fre 14.04.17
19	Fremddaten	44 Tage	Fre 21.04.17	Mit 21.06.17
20	Kommunikatio	8 Tage	Mit 21.06.17	Fre 30.06.17
21	**Datenbank**	**14 Tage**	**Fre 17.02.17**	**Mit 08.03.17**
22	Datenbankdes			
23	Abfragen			
24	**Visualisierung**	**21 Tage**	**Fre 23.06.17**	**Fre 21.07.17**
25	Kommunikatio	81 Tage	Fre 10.03.17	Fre 30.06.17
26	Aufbereitung	81 Tage	Fre 10.03.17	Fre 30.06.17
27	GUI	23 Tage	Mit 08.02.17	Fre 10.03.17
28	Zusammenfügen Programme/Serv	22 Tage	Sam 01.07.17	Mon 31.07.17
6				
7	**Sekundärmission**	**120 Tage**	**Fre 17.02.17**	**Don 03.08.17**
8	Funk	66 Tage	Mit 30.11.16	Mit 01.03.17
9	Platine	97 Tage	Fre 17.02.17	Sam 01.07.17
10	Software	97 Tage	Fre 17.02.17	Sam 01.07.17
11	Zusammenfügen Hardware/Software	11 Tage	Sam 01.07.17	Fre 14.07.17
12				
13	**Primärmission**	**127 Tage**	**Mit 08.02.17**	**Don 03.08.17**
14	**Gehäuse**	**82 Tage**	**Mit 08.02.17**	**Don 01.06.17**
15	Entwurf	53 Tage	Mit 08.02.17	Fre 21.04.17

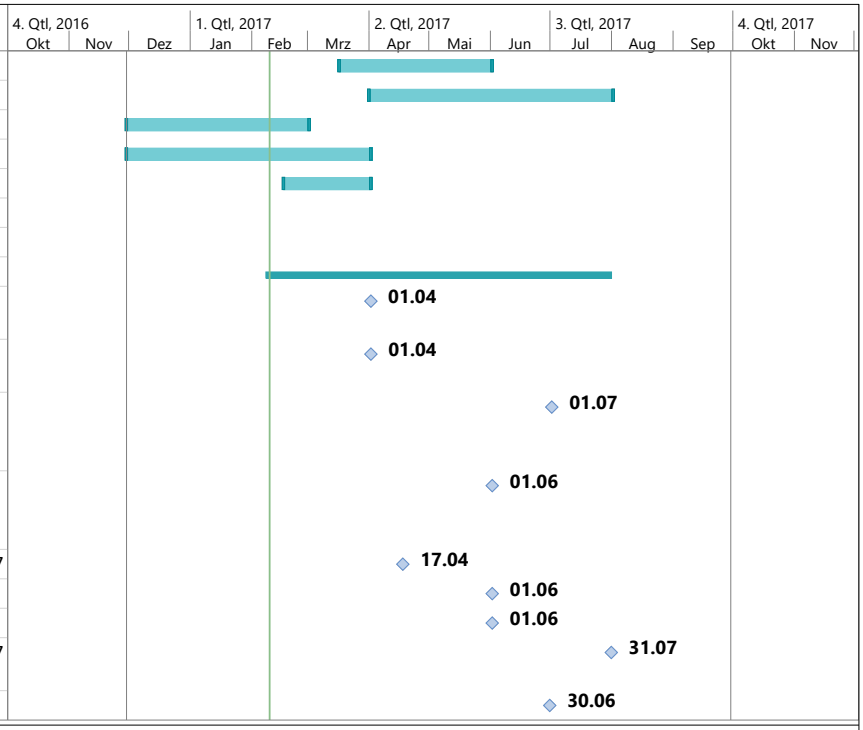


Projekt: Planung Neu
Datum: Fre 10.02.17

Vorgang
Unterbrechung
Meilenstein
Sammelvorgang
Projektsammelvorgang
Inaktiver Vorgang
Inaktiver Meilenstein
Inaktiver Sammelvorgang
Manueller Vorgang
Nur Dauer
Manueller Sammelrollup
Manueller Sammelvorgang
Nur Anfang
Nur Ende
Externe Vorgänge
Externer Meilenstein
Stichtag
In Arbeit
Manueller Fortschritt

Seite 2

Nr.	Vorgangsname	Dauer	Anfang	Ende
16	Herstellung	55 Tage	Fre 17.03.17	Don 01.06.17
17	Platinen	88 Tage	Sam 01.04.17	Die 01.08.17
18	Funk	66 Tage	Mit 30.11.16	Mit 01.03.17
19	Sensorik	89 Tage	Mit 30.11.16	Sam 01.04.17
20	Software	32 Tage	Fre 17.02.17	Sam 01.04.17
21				
22				
23	**Meilensteine:**			
24	Prototyp Primärmission			Sam 01.04.17
25	Prototyp Sekundärmission			Sam 01.04.17
26	Kommunikation zwischen Primär und			Sam 01.07.17
27	Kommunikation Satellit - Bodenstation			Don 01.06.17
28	Platine entwickelt			Mon 17.04.17
29	Gehäuse Primär			Don 01.06.17
30	Gehäuse Sekundär			Don 01.06.17
31	Bodenstation fertiggestellt			Mon 31.07.17
32	**Datenübertragung**			Fre 30.06.17



Projekt: Planung Neu
Datum: Fre 10.02.17

Vorgang
Unterbrechung
Meilenstein
Sammelvorgang
Projektsammelvorgang
Inaktiver Vorgang
Inaktiver Meilenstein
Inaktiver Sammelvorgang
Manueller Vorgang
Nur Dauer
Manueller Sammelrollup
Manueller Sammelvorgang
Nur Anfang
Nur Ende
Externe Vorgänge
Externer Meilenstein
Stichtag
In Arbeit
Manueller Fortschritt

Seite 3

Nr.	Vorgangsname	Dauer	Anfang	Ende
33	**Datenverarbeitung**			Fre 30.06.17
34	Datenmanagement			Mon 01.05.17
35	**Datenbank**			Mit 08.03.17
36	**Visualisierung**			Fre 21.07.17
37				

Projekt: Planung Neu
Datum: Fre 10.02.17

Vorgang	Inaktiver Sammelvorgang	Externe Vorgänge
Unterbrechung	Manueller Vorgang	Externer Meilenstein
Meilenstein	Nur Dauer	Stichtag
Sammelvorgang	Manueller Sammelrollup	In Arbeit
Projektsammelvorgang	Manueller Sammelvorgang	Manueller Fortschritt
Inaktiver Vorgang	Nur Anfang	
Inaktiver Meilenstein	Nur Ende	