Designdokument

Bremen

Team Gamma

Autoren:
Robin Bley
Alexander Brennecke
Alexander Feldmann
Marc Huisinga
Kevin Neumeyer
Till Schlechtweg
Steffen Wissmann

Betreuer:
Harm Hörnlein-Roboom
Frank Marschall

20. September 2015

Inhaltsverzeichnis

1 Einleitung

Diese Dokumentation dokumentiert die Arbeit an dem Dosensatelliten "Apollo 13" und die Entwicklung einer Bodenstation zur Datenverarbeitung der Daten des Satelliten im Rahmen des Deutschen CanSat-Wettbewerbs 2015.

1.1 Teamorganisation und Aufgabenverteilung

Das gesamte Team besteht aus sieben Schülern und zwei betreuenden Lehrern. Die sieben Schüler sind intern in mehrere kleinere Teams aufgeteilt. Innerhalb der Teams ist kein Teammitglied vollkommen an seine Aufgaben gebunden, da uns ein guter Austausch und eine hervorragende Zusammenarbeit zwischen den einzeln Teammitgliedern und Teams wichtig ist. Die Arbeiten der Gruppen und der einzelnen Personen werden im Folgenden erläutert:

- Das Hardware-Team besteht aus drei Personen, welche sich um den Bau des Satelliten selber, das Design und den Bau der Dose sowie die Programmierung des Mikrocontrollers kümmern. Zu diesem Team zählen folgende Personen:

  Alexander Brennecke ist verantwortlich für das Design der Dose. Dazu zählt die Konstruktion der eigentlichen Dose und die Anordnung der Sensoren im Inneren der Dose.

  Till Schlechtweg ist verantwortlich für die Funktionalität des Mikrocontrollers und der ausgewählten Sensoren.

  Steffen Wißmann ist verantwortlich für die Übertragung der Daten zur Bodenstation und den Programmcode des Mikrocontrollers.

- Das Software-Team besteht aus vier Personen, welche sich um das Programmieren der Bodenstation und der Android-Applikation kümmern. Allgemein gilt für alle Personen dieser Gruppe, dass die Grenzen der Zuständigkeitsbereiche der verschiedenen Personen verfließen, wobei jede Person allerdings noch ein gewisses Spezialgebiet besitzt. Dieses Team besteht aus folgenden Personen:

  Robin Bley ist verantwortlich für das Implementieren der Datenverarbeitung der Bodenstation und für das Testen von kritischen Bereichen innerhalb der Datenverarbeitung.

  Alexander Feldmann ist primär verantwortlich für die Entwicklung der Android-Applikation, die Konstruktion des Fallschirms und die Konstruktion der Antenne.

  Marc Huisinga ist ebenfalls verantwortlich für das Implementieren der Datenverarbeitung der Bodenstation, für die Entwicklung der Datenvisualisierungskomponenten und für die Architektur der Datenverarbeitung.

  Kevin Neumeyer ist verantwortlich für die zusammenführende Architektur der Bodenstation, die grafische Umgebung und die Administration der Software-Repositories.

- Zudem gibt es ein Team, bestehend aus Alexander Brennecke und Till Schlechtweg, welches sich um die Organisation, die Kommunikation mit Sponsoren und die Öffentlichkeitsarbeit kümmert.

- Betreut wird das Projekt durch zwei Lehrer unserer Schule:

  Mathematiklehrer Harm Hörnlein-Roboom, welcher als Ansprechpartner für die Software-Gruppe zur Verfügung steht.

  Physiklehrer Frank Marshall, welcher als Ansprechpartner für die Hardware-Gruppe und den CanSat-Wettbewerb zur Verfügung steht.

Die Arbeit an dem Projekt findet zum größten Teil wöchentlich am Dienstag und Mittwoch Nachmittag in den Laboren unserer Schule statt. Die Labore sind mit diversen Werkzeugen ausgestattet, sodass sowohl die Software- als auch die Hardware-Gruppe dort problemlos arbeiten können. Zusätzlich zu diesen vier bis acht Stunden pro Woche kommen fünf Projekttage, welche uns von der Schule breitgestellt wurden, an welchen wir jeweils ebenfalls sechs bis acht Stunden

arbeiten können. Natürlich arbeitet jedes Teammitglied auch außerhalb dieser Treffen an seinem Fachgebiet, soweit dies möglich ist. Zusätzlich gibt es auch immer wieder Treffen mit externen Unterstützungen, oder Arbeitszeit in der Schule, wenn Vertretungs- oder Mitbetreuungsunterricht stattfindet.

1.1.1 Stärken des Teams

Die große Stärke des Teams ist es, dass es auch schon vor diesem Projekt existiert hat und sich somit sehr gut kennt. Die Teilnahme am Europäischen CanSat Wettbewerb 2014 hat dazu geführt, dass jedes Teammitglied gewisse Vorkenntnisse mitbringt. Diese Vorkenntnisse spiegeln sich vor allem in dem Bau eines CanSats, der Planung des Projektes und dem Umgang mit Fehlern wieder. Ebenfalls von Vorteil ist, dass jedes Teammitglied durch unsere schulische Ausbildung genügend Erfahrung hat, um auch außerhalb seines Fachgebietes unterstützend tätig zu sein. Zudem ist die Arbeits- und Leistungsbereitschaft der meisten Teammitglieder überdurchschnittlich gut.

1.1.2 Verbesserungsbereiche des Teams

Der größte Verbesserungsbereich des Teams liegt ganz klar im Zeitmanagement. Für viele Aufgaben wird zu wenig Zeit eingeplant. Oft kommt es auch vor, dass der Schwerpunkt der Arbeit auf Dingen liegt, welche nicht höchst priorisiert sind und somit Zeit beanspruchen, welche an anderen Stellen dringender benötigt würde. Ebenfalls problematisch ist, dass die meisten Teammitglieder gerne neue Technologien oder Praktiken ausprobieren wollen. Dieses Interesse ist zwar löblich und für die Einzelperson sehr lehrreich, jedoch kommt es bei neuen Technologien und Praktiken oft zu Problemen, die man bei bereits bekannten deutlich schneller lösen könnte.

1.1.3 Veränderungen im Team

Während des Projektes hat sich der Aufbau unseres Teams verändert. Till Schlechtweg musste leider aufgrund von gesundheitlichen Problemen das Team in den letzten zwei Monaten des Projektes verlassen. Die Aufgaben von Till Schlechtweg wurden von seinen Team-Mitgliedern Alexander Brennecke und Steffen Wißmann geteilt übernommen.

1.2 Missionsziel

Die Idee hinter dem gesamten Projekts bezieht sich auf die extreme Umweltbelastung und ihre Folgen für den menschlichen Körper. Ausschlaggebend für diese Idee ist ein Zeitungsartikel der Zeit, welcher über eine drohende Klage der EU-Kommission in Brüssel berichtet. (vgl. Die Zeit, 24.10.2014). Die Klage richtet sich gegen Deutschland, da die deutsche Bundesregierung bisher zu wenig Aufwand betreibt, um die Feinstaubkonzentration in der Luft zu reduzieren. Wir möchten diesen Aspekt aufgreifen und Messungen durchführen um die tatsächlichen Werte zu bestimmen. Der CanSat Wettbewerb eignet sich optimal dazu, da er uns die Möglichkeit bietet, die Messungen nicht nur auf dem Boden, sondern auch in verschiedenen Höhen durchzuführen. Feinstäube stehen in Verdacht, Krankheiten wie Asthma, Herz-Kreislauf-Beschwerden und Krebs zu begünstigen.

Da der menschliche Körper nicht nur durch Feinstaub belastet wird, haben wir uns entschlossen, auch die Intensität der UV-Strahlung, welche die Hauptursache für Hautkrebserkrankungen ist, zu messen. Zusätzlich soll auch der Ozonwert bestimmt werden, da Ozon bereits in geringen Konzentrationen gesundheitsschädlich ist und zu Reizungen der Atemwege führen kann.

Für sich genommen ist jede dieser drei Größen schädlich für den Menschen. Im Zuge des Projektes wollen wir jedoch versuchen herauszufinden, ob es einen Zusammenhang zwischen ihnen gibt. Beispielsweise ist herauszufinden, ob ein höherer Ozongehalt gleichzeitig einen niedrigeren Feinstaubgehalt mit sich bringt.

Zusätzlich zum Bau des Messsystems im CanSat, ist es unser Ziel, eine einwandfreie Verarbeitung, Analyse und Präsentation der gemessenen Werte zu erzielen. Um dies zu garantieren, programmieren wir ein eigenes Analysetool. Dieses Tool ermöglicht es uns, die gemessenen Werte während des Fluges des Satelliten auszuwerten. Die Werte sollen dabei anschaulich und in Abhängigkeit zueinander dargestellt werden. Die Bodenstation soll zusätzlich darauf ausgelegt

sein, nicht nur unseren Satelliten zu unterstützen. Viel mehr soll das Analysetool auch anderen CanSat-Missionen eine Plattform bieten, auf der die gemessenen Daten ausgewertet werden können.

Um die Daten auch mobil verfügbar zu haben, möchten wir eine Android Applikation bereitstellen. Diese Applikation soll vorerst nur für unser Projekt optimiert sein, bei Erfolg jedoch auch die Werte anderen Teams anzeigen können.

2 Beschreibung des CanSat

2.1 Missionsüberblick

Wir haben uns für den Satelliten überlegt, dass dieser so individuell wie möglich sein soll. Daher greifen wir nicht auf das vom Wettbewerb bereitgestellte T-Minus-CanSat-Kit zurück. Stattdessen haben wir uns im Detail überlegt, welche Sensoren unseren Erwartungen entsprechen und wie wir diese bestmöglich innerhalb der Dose platzieren können. Zusätzlich möchten wir nicht auf eine standardisierte Getränkedose als Hülle zurückgreifen, sondern auch hier unser eigenes Design erschaffen.

2.2 Mechanik- und Strukturdesign

Wir haben den CanSat in drei Komponenten aufgeteilt: Die Hülle, die Innenwand und die Sensorikplatine. Diese drei Komponenten bilden den Hauptbestandteil des CanSats und haben maßgeblich zu dem mechanischen und strukturellem Design beigetragen. Im Nachfolgenden wird kurz auf jeden dieser Komponenten eingegangen und die exakte Funktion im Zusammenhang erklärt.

2.2.1 Fachliche Grundlagen

Um die 3D-gedruckte Wand zu erzeugen, wurde die 3D-Moddelierungssoftware Sketchup von Google verwendet. Sketchup bietet die Möglichkeit, vergleichsweise einfach 3D-Modelle zu zeichnen. Um dies zu tun, muss klar sein, welche Objekte gezeichnet werden sollen. Diese Objekte müssen vermessen und innerhalb von Sketchup gezeichnet werden. Dies erfordert die Kenntnis über gewisse mathematische Methoden zur Berechnung von Kreisen, Flächen und Körpern. Die meisten 3D-Drucker benötigen Dateien des Typs .stl, welche in Sketchup mit einem Plugin erzeugt werden können. Zum Anfertigen von GFK-Komponenten wird ein Körper benötigt, auf welchen das GFK laminiert werden kann. In unserem Fall ist dieser Körper zylindrisch, mit einem Durchmesser von 31,5 mm, und aus Aluminium gefräst.
Um die Platine zu erstellen, wurde die Design-Software Eagle PCB verwendet. Eagle bietet die Möglichkeit, sowohl Schaltpläne als auch das entsprechende Layout zu erstellen. Im Anschluss wurde die Platine, mit Hilfe und Mitteln des Hackerspace Bremen e.V., geätzt.

2.2.2 Hülle

Wir haben uns dazu entschieden, die äußere Hülle aus GFK (Glasfaser verstärkter Kunststoff) anzufertigen. Dieser hat die Eigenschaft, dass er bei einem sehr geringen Gewicht, und bei einer geringen Wandstärke, trotzdem eine gewisse Stabilität aufweist. Aus dem GFK haben wir eine Röhre mit einem Innendurchmesser von 31,5 mm und einem Außendurchmesser von 33,5 mm laminiert. Diese Röhre wurde auf eine Länge von 111 mm gekürzt und gefeilt. Um die Röhre oben und unten zu verschließen, haben wir uns bei Thyssen Krupp System Engineering zwei Aluminiumdeckel fräsen lassen. Diese haben uns ebenfalls durch ihr geringes Gewicht und ihre hohe Stabilität überzeugt. Die Deckel haben, genauso wie die Hülle, einen Außendurchmesser von 33,5 mm. Sie sind beide 2 mm dick und haben zusätzlich eine 2mm dicke Erhöhung, welche in die Röhre eingelassen wird. Eine Abbildung der Hülle kann unter 8 gefunden werden.

2.2.3 Innenwand

Um die Elektronik innerhalb der Hülle zu platzieren und zu befestigen, haben wir uns dazu entschieden, eine Wand anzufertigen. Diese Wand teilt die Hülle mittig und bietet so auf beiden Seiten Platz, um unser Mikrokontrollerboard und unsere Sensorikplatine zu befestigen. Beide Bauteile werden mittels vier Gewindestangen an der Wand befestigt. Durch die Technik des 3D-Druckens ist es möglich, der Wand ein sehr geringes Gewicht bei einer verhältnismäßig hohen Stabilität zu verleihen. Zusätzlich gibt es uns die Möglichkeit, die Wand millimetergenau zu gestalten. Die Wand kann in ihrer finalen Form nur noch sehr schwer 3D gedruckt werden. Der Drucker muss dafür sogenannten “Support” drucken, welcher später wieder herausgeschnitten werden muss. Da dies die Druckzeit und die Qualität des Endproduktes vermindert haben

wir uns dazu entschlossen die Wand vertikal zu zerteilen. Dadurch vermindert sich die Stabilität minimal, jedoch ist dieser Verlust so minimal, dass er zu vernachlässigen ist.

Am unteren Ende der Wand befindet sich eine Aushöhlung, sowie ein Fuß. Diese ist zum einen dafür da, um den Sharp-Feinstaubsensor zu befestigen. Zum anderen gibt der Fuß der Wand, und somit dem gesamten Satelliten, eine gewisse Stabilität. Der Fuß besitzt auf der einen Seite der Wand Bohrungen. Diese Bohrungen werden verwendet, um die Aluminiumdeckel an der Wand zu befestigen. An der oberen Seite der Wand befinden sich ebenfalls solche Bohrungen, um den oberen Deckel der Hülle zu befestigen. Da der Feinstaubsensor einen Luftzug benötigt, gibt es eine Bohrung, welche vertikal durch die Wand führt. Um das Mikrokontrollerboard mit der Sensorikplatine zu verbinden, existiert ein Fenster in der Mitte der Wand. Um die Sensorikplatine und das Mikrokontrollerboard an der Wand zu befestigen, existieren in der Wand weitere vier Bohrungen. Ein Bild der Wand kann unter 9 gefunden werden, während eine 3D-Darstellung der Wand unter 10 gefunden werden kann.

2.2.4 Sensorikplatine

Die Sensorikplatine ist eine von uns geätzte Platine, welche mit unseren Sensoren bestückt ist. Es gibt mehrere positive Aspekte, die eine eigene Platine mit sich bringt. Zum einen bietet sie eine stabile Plattform für die Befestigung der Sensoren. Zum anderen sparen wir uns dadurch eine Menge Kabel, welche deutlich störanfälliger sind als eine Platine. Die Platine hat an den entsprechenden Stellen Bohrungen, um sie mit der Zwischenwand und dem Mikrokontrollerboard zu verbinden. Die Platine bietet Platz für die folgenden Module:

- BMP108 Drucksensor: Misst den Luftdruck und gibt diesen, sowie die daraus berechnete Höhe, zurück
- Sparkfun UV Sensor: Misst die Intensität der Strahlung des Spektrums 270-380 nm, welches dem UVA und UVB Spektrum entspricht
- TMP006 Infrarot Temperatursensor: Misst die Temperatur eines dünnen Aluminiumstückes in der Außenwand
- Adafruit Ultimate GPS: Bestimmt die aktuelle Position, sowie die Höhe
- APC220 Transceiver Modul: Sendet die Daten als JSON-String zur Bodenstation
- Steckplatz zum Anschluss des Sharp-Feinstaubsensors: Misst den Anteil der Partikel, welche kleiner als 10 µm sind
- Steckplatz zum Anschluss an das Mikrokontrollerboard: Bildet die Schnittstelle zwischen BeagleBone und Sensorikplatine

Der Schaltplan der Sensorikplatine kann unter 11 gefunden werden, während das Layout der Platine unter 12 gefunden werden kann.

2.2.5 Batteriehalterung

Als Stromversorgung wurden zwei Mignon-AA-Batterien gewählt. Diese liefern pro Batterie 3,6V und somit in Reihe geschaltet 7,2V. Die Batterien werden in einer Standard-Mignon-AA-Batteriehalterung platziert. Um diese Batteriehaltung in dem Design unterzubringen, wurde eine dünne Platte per 3D-Druckverfahren angefertigt. Diese Platte kann über die beiden Female-Pin-Layers des BeagleBone Black gelegt werden. Zur Befestigung dienen zwei Male-Pin-Layers, welche durch Löcher in der 3D-Platte in das BeagleBone Black gesteckt werden können. Die Batteriehalterung wurde auf die 3D-Platte geklebt und zusätzlich mit Schrauben befestigt.

Diese Schrauben dienen ebenfalls dazu, eine zweite Platine von unten an die 3D-Platte zu schrauben. Diese zusätzliche Platine ist mit einem Vorwiederstand, einer Z-Diode und einer Diode ausgestattet und dient dazu, die Ausgangsspannung des BeagleBone Black zu reduzieren und zu glätten. Die 7,2V Ausgangsspannung kann dadurch auf 5V reduziert werden, was für das Transceiver Modul notwendig ist.

2.3 Elektrische Konstruktion

Unser CanSat besteht aus mehreren Sensoren und einem zentralen Verarbeitungssystem, sowie einem Sender. Diese kommunizieren alle über verschiedene Protokolle. Im Anhang unter der Einleitung befindet sich das Blockdiagramm unseres Satelliten. Im Blockdiagramm fehlen allerdings die verschiedenen Protokolle. In unserem Fall kommuniziert der BeagleBone Black, die MCU, mit allen Sensoren und holt deren Daten ab.

Bauteil	Kommunikationsprotokoll
UV ML8511	ADC
Sharp-Feinstaubsensor	ADC
APC220	UART
Ultimate GPS	UART
TMP006	I^2C
BMP180	I^2C

Tabelle 1: Kommunikationsprotokolle

2.3.1 Fachliche Grundlagen

2.3.1.1 Embedded System Ein Embedded System ist in unserem Fall der BeagleBone Black. Das Mikrokontrollerboard taktet mithilfe eines ARM Cortex-A6 Prozessors mit 1 GHz. Auf ihm ist der leicht modifizierte Linux-Kernel Angstrom mit Frontend installiert. Andere Beispiele für ein Embedded System sind etwa ein Smart-TV oder ein Router. Beide besitzen eine Main Control Unit mit einem Betriebssystem, welches auf die Anwendung des Gerätes spezialisiert ist. In unserem Fall ist dies der BeagleBone, welcher verschiedene Technologien besitzt, um mit einzelnen Bauteilen zu kommunizieren: UART, I-2-C, SPI, Analog, Digital, PWM, Timer, PRU, ADC, DAC und viele mehr. Viele dieser Technologien sind in unserem Projekt nicht in Verwendung. Jene, die in Verwendung sind, werden später im Dokument beschrieben.

2.3.1.2 Transistor-Transistor-Logik 5V werden immer als logisch "Ein" bezeichnet. Damit ist gemeint, dass der Sensor, wenn er den höchsten Messwert erreicht, eine Spannung von 5V ausgibt. Ist dies nicht der Fall, so hat der Sensor eine andere Kennkurve, die zum Beispiel bei 3.3V aufhört. Allgemein wird aber die Transistor-Transistor-Logik genutzt, welche 5V als logisch "Ein" und geerdet als logisch "Aus" ansieht. Es gibt natürlich Toleranzen, welche aber bei verschiedenen integrierten Schaltkreisen und Mikrokontrollern unterschiedlich sind.

2.3.1.3 Analog-to-Digital-Converter Andere Sensoren, beispielsweise der UV-Sensor, verfügen lediglich über einen internen Widerstand. Der Widerstand verändert sich in Abhängigkeit zu einer mathematischen Kurve. Dadurch entstehen unterschiedliche Spannungen, welche über den jeweiligen analogen Pin ausgegeben werden. Mithilfe eines Analog-to-Digital-Converters konvertieren wir das analoge Signal, zum Beispiel 5V, in das äquivalente digitale Signal mit einer Auflösung von 12 Bits.

$$2^{12} \quad = 4096$$

So können 4096 verschiedene Stufen dargestellt werden. Da ein analoges Signal theoretisch in unendlich viele Stufen unterteilt werden kann, scheinen 4096 Stufen auf den ersten Blick nicht wie viel. Ein einzelner Schritt ist trotzdem im Digitalen (12 Bits) sehr klein, was folgende Rechnung zeigt.

$$\frac{5V}{4096} = 0.001220703125V$$

Das Ergebnis bedeutet, dass man mit 12 Bits eine 5V-Spannung in 0.001220703125V Schritten darstellt kann. Dem Arduino Mega 2560 stehen nur 10 Bits zur Verfügung. Da die Rechnung exponentiell ist, verkleinert sich dadurch die Anzahl der Stufen enorm.

$$2^{10} \quad = 1024$$

$$\frac{5V}{1024} = 0.0048828125V$$

Wie man an dem Ergebnis sieht, kann das BeagleBone den Wert, der am analogen Pin ankommt, viel genauer darstellen, als der Arduino Mega 2560.

2.3.1.4 Universal-Asynchronous-Receiver-Transmitter UART ist eine digitale, serielle Schnittstelle zum Realisieren von einfachen Kommunikationen zwischen zwei Endpunkten. Die Funktionsweise ist denkbar einfach. Wir nutzen in unserem Satelliten meist eine Baudrate von 9600bps. Baud ist die Schrittgeschwindigkeit oder Symbolrate, also 9600 bits per second. Für UART gibt es wie beim RJ45-Stecker TX und RX, die beim Aufbau einer Kommunikation gekreuzt werden. Dies liegt daran, dass der Transceiver des einen Komponenten an den Receiver des anderen angeschlossen werden muss. Nun wird zwischen vielen verschiedenen Arten von UART unterschieden. In unserem Fall wird die TTL-UART Variante genutzt, welche die beim Analog-to-Digital-Converter genannten 5V als logisch "Ein" bezeichnet.

2.3.1.5 Inter-Integrated-Circuit I-2-C ist ein serieller Datenbus, der über zwei Kabel mit einer 10-Bit-Adressierung (welche 1024 Stufen entspricht) arbeitet. Der Bus ist auf eine maximale Geschwindigkeit von 5 Mbit/s beschränkt, welche für unsere Zwecke jedoch ausreichend ist. Der Sinn des Bussystems ist es, mithilfe von einer Adresse einen Datensatz oder Befehl nur an den gewünschten Empfänger zu senden. I-2-C benötigt lediglich eine Datenleitung, welche eine Kommunikation zwischen dem Master (in unserem Fall das BeagleBone) und den Slaves (in unserem Fall die Sensoren) herstellt. Der Master kann über diese Datenleitung den Slaves mitteilen, wann welcher Slave seine Daten senden darf.

2.3.2 Sensorik

2.3.2.1 ML8511 - UV-Sensor Der UV-Sensor enthält im Inneren lediglich eine Fotodiode, welche auf eine Wellenlänge zwischen 280 und 390 nm reagiert. Zusätzlich zu der Diode existiert ein Verstärker, welcher dafür sorgt, dass auch minimale Veränderung gemessen werden können. Die Fotodiode ändert je nach Einstrahlung von UV-A und UV-B ihren Widerstand. Die dabei entstehende Veränderung in der Spannung ist messbar.

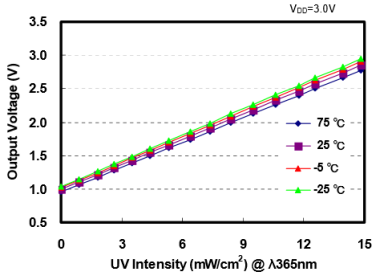

Abbildung 1: Spannungsausgabe vs. UV-Intensität

2.3.2.2 Sharp-Feinstaubsensor Der Sharp-Feinstaubsensor arbeitet, wie der ML8511, sehr simpel. Im Inneren befindet sich eine Infrarot-Diode, welche die Partikel anstrahlt. Auf der anderen Seite befindet sich ein Fototransistor, welcher dann feststellt, wie viel von diesem Licht von Partikeln reflektiert wird. Diese Veränderung ist messbar.

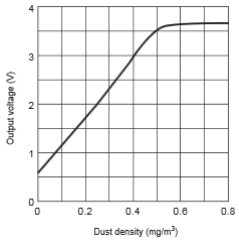

Abbildung 2: Spannungsausgabe vs. Staub

Um den Fototransistor nicht immer ganz zu bestrahlen, ist die Infrarot-Diode nicht die gesamte Zeit angeschaltet. Sie scheint lediglich für den Bruchteil einer Sekunde. Die Messung muss innerhalb dieser Zeitspanne stattfinden.

2.3.2.3 APC220 Der APC220 ist ein Transceiver, welcher entweder senden oder empfangen kann. Der BeagleBone Black schickt den formatierten JSON-String per UART an den APC220. Dieser schickt ihn über die vorher am Computer festgelegte Frequenz in den Raum weiter. Am Boden befindet sich ebenfalls ein APC220, welcher die Daten empfängt.

2.3.2.4 Ultimate GPS Der Ultimate GPS von Adafruit verbindet sich mit den allgemeinen GPS-Satelliten und sendet seriell seine Daten im NMEA-Format. Dieses Format gibt es in diversen Varianten, es verfügt aber in so gut wie allen Varianten über folgende Daten:

Nummer	Daten
1	Breitengrad
2	Längengrad
3	Zeit
4	GPS-Qualität
5	Anzahl benutzter Satelliten
6	Höhe

Tabelle 2: Auslesebare Daten

Außerdem können wir den Ultimate GPS über die serielle UART-Schnittstelle konfigurieren, um ihn zum Beispiel nur ein bestimmtes NMEA-Format ausgeben zu lassen, oder um die Bitrate der seriellen Übertragung zu ändern.

2.3.2.5 TMP006 Der TMP006 ist ein Infrarot-Temperatursensor, welcher die Temperatur von einem Objekt misst, ohne in direktem Kontakt zu stehen. Der Sensor misst die Temperatur eines Objektes anhand der ausgestrahlten Energie auf den Wellenlängen von 4 µm bis zu 16 µm. Durch die veränderte Spannung am Sensor ist eine Messung der Temperatur möglich. Je größer das Objekt ist, umso weiter entfernt muss es sich befinden, um vom "Field of View" des Sensors erfasst zu werden.

Die Messung kann sehr ungenau werden, da, je nach Außentemperatur und Temperatur der Sensorfläche selber, Fehler beim Messen entstehen können.

2.3.2.6 BMP180 Der BMP180 ist ein Drucksensor, welcher mithilfe einer Membran den Druck misst und diesen per On-Board-Controller direkt in die Höhe umrechnet. Der Sensor gibt die Daten dann per I^2C-Bus aus.

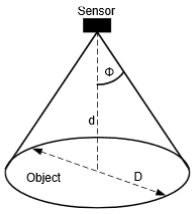

Abbildung 3: Sensor Field of View

2.3.3 Energieverbrauch

Bauteil	Stromaufnahme	Spannung	Leistungsaufnahme
BeagleBone Black	500mA	5V	2500mW
ML8511	<1mA	3.3V	<3.3mW
TMP006	<1mA	3.3V	<3.3mW
Sharp	20mA	3.3V	66mW
BMP180	<1mA	3.3V	<3.3mW
Ultimate GPS Modul	25mA	3.3V	82.5mW
APC220	35mA	5V	175mW
			2833.4mW

Tabelle 3: Energieverbrauch

2.4 Softwaredesign

2.4.1 Python als Programmiersprache

Als Programmiersprache für das BeagleBone Black haben wir uns für Python entschieden. Es wäre ebenfalls möglich gewesen, den Mikrocontroller mit den Sprachen JavaScript, Java, C, C++, C# und vielen weiteren Sprachen zu programmieren. Da es sich bei dem BeagleBone um ein Linux-basiertes Embedded System handelt, unterstützt es praktisch alle Programmiersprachen, sofern entsprechende Bibliotheken existieren. Allerdings haben wir uns aufgrund der Tatsache, dass Python im Gegensatz zu Java nicht objektorientiert geschrieben werden muss, für Python entschieden. Wir möchten auf der Hardwareseite möglichst auf objektorientierte Programmierung verzichten. Ein weiteres wichtiges Argument ist die gute Python-Bibliothek, welche von einer großen Community permanent gewartet und aktualisiert wird.

2.4.2 Datenverarbeitung auf dem BeagleBone

Die Datenverarbeitung auf dem BeagleBone verläuft relativ simpel. Zunächst werden alle von den Sensoren aufgezeichneten Daten, bei Sensoren mit I2C-Anbindung, mithilfe von Libraries, und bei den anderen mithilfe von Umrechnungsalgorithmen, gesammelt. Anschließend werden alle gesammelten Daten in einen JSON-String geparsed, welcher mithilfe von unserem Transceiver an die Bodenstation übermittelt wird. Diese übernimmt hieraufhin die weitere Verarbeitung und Darstellung der Messdaten. Zusätzlich werden die Daten auf dem internen Speicher des BeagleBones gespeichert.
Eine simple Skizze der Architektur lässt sich auch unter 4 finden.

2.5 Bergungssystem

Für unser Landesystem haben wir uns entschieden, unseren eigenen Fallschirm zu bauen. Die Hauptaufgabe ist es, eine weiche Landung auf dem Boden zu garantieren. Unsere Vorgabe war, dass der Fallschirm und die Dose eine Fallgeschwindigkeit von 15 Meter/Sekunde haben soll. Leider sind unsere Testfallschirme, die wir auch schon im Vorjahr genutzt haben, für eine deutlich

geringer Fallgeschwindigkeit ausgelegt. Dies haben wir uns zum Anlass genommen eine neue Fallschirmart zu testen. Die Idee dabei ist, dass die acht Schnüre, welche den Fallschirm an der Dose befestigen, mit sehr luftdurchlässigem Stoff, sogenannter Gaze, ersetzt werden. Wir erhoffen uns dadurch eine stabilere Lage in der Luft und ein fortschrittlicheres Design.

2.5.1 Berechnungen

Für den Bau des Fallschirm wissen wir bereits, dass dieser mit $v = 15\frac{m}{s}$ fallen soll. Außerdem haben wir einen bereits berechneten Strömungswiderstandskoeffizient (C_w) von 1,33. Für die Berechnung des Fallschirms wurde folgende Formel verwendet:

$$F_w = C_w \cdot \frac{1}{2} \cdot \rho \cdot v^2 \cdot A$$

Fw ist die Strömungswiderstandskraft. Diese kann ermittelt werden, indem man die Fallwiederstandskarft einsetzt.

$$F_w = m \cdot g = 350g \cdot 9,81\frac{m}{s^2} = 3,3433\frac{m}{s^2}kg$$

Um die Größe des Fallschirms zu berechnen, kann man nun durch Einsetzen in die erste Formel diese nach A umstellen.

$$A = \frac{2 \cdot 3,3433\frac{m}{s^2}kg}{C_w \cdot \rho \cdot v^2}$$

$$A = \frac{2 \cdot 3,3433\frac{m}{s^2}kg}{1,33 \cdot 1,2\frac{kg}{m^3} \cdot 15^2\frac{m^2}{s^2}} = 0,01862m^2 \text{ oder } 186,2cm^2$$

Eine Fläche von 186,2 cm^2 entspricht einem Durchmesser von ca. 16 cm.

2.5.2 Bau

Beim Bau der Fallschirme haben wir den Stoff von Regenschirmen verwendet. Dieser Stoff ist sehr geeignet, da er bereits in einer Art Halbkugelform mit acht aneinandergefügten Panels ist. Er bietet genug Widerstand, um den Belastungen des Fluges standzuhalten und hat als praktischen Nebeneffekt, dass es regendicht ist. Um an den Stoff zu kommen muss das Gestell des Regenschirmes vorsichtig vom Stoff heruntergeschnitten werden. Danach muss in der Mitte des Stoffes, wo alle acht Kanten aufeinandertreffen, ein rund 5 cm großes Loch geschnitten werden. Dort kann die gestauchte Luft leichter entweichen, statt am Rand unkontrolliert auszutreten. Würde sie dies nicht tun, so könnte es leicht zu gefährlichen Turbulenzen kommen. Nun kann der Fallschirm in die entsprechende Größe zugeschnitten werden. Am Rand des Fallschirms sollte ein zusätzlicher ca. 1 cm breiter Rand gelassen werden, der in den Fallschirm umgeklappt wird. Dieser muss mit einer Nähmaschine auf den Fallschirm festgenäht werden. Dies dient dazu, die Struktur des Randes besser zu schützen. Am Rand besteht die höchste Wahrscheinlichkeit, dass durch eine kleine Kerbe ein kompletter Riss entstehen könnte. Auf dieser Grundlage kann nun die Gaze, auf dem Rand, genäht werden. Dabei sollte beachtet werden, dass beim Übergang von Fallschirm und Dose am Befestigungspunkt eine hohe Belastung auftritt. An diesem Punkt hat die Gaze einen gewaltigen Nachteil gegenüber der Schnüre. Hier kann es durch eine einmalige starke Belastung zum Riss kommen, der sich im Laufe des Fluges weiter ausbreiten kann. Daher haben wir uns dort dazu entschieden, eine Verstärkung mit einem sehr rissfesten Material einzunähen, der die Spannung erst kompensiert und diese nach dem Entfalten des Fallschirms gut auf die Gaze verteilt. Eine einfache Skizze des Fallschirms kann unter 13 gefunden werden.

2.6 Antenne

Wir haben uns für den Bau einer Helixantenne entschieden. Diese hat gegenüber der Yagi-Antenne eine größere Verlässlichkeit beim Empfangen von Daten, da ihr Ausrichtungswinkel im Bezug auf den CanSat keine Rolle spielt und so durch die ständige Rotation des CanSats keine Daten verloren gehen können. Der größte Nachteil bei einer Helixantenne ist, dass sie eine fast 1m^2 große Bodenplatte besitzt. Da unsere Helixantenne von einer Person gehalten wird, haben wir darauf Wert gelegt, diese so leicht wie nur möglich zu bauen. Um das Gewicht der Antenne

zu reduzieren, haben wir eine zwei Millimeter dicke Bodenplatte verwendet, was im Gegensatz zu oft verwendeten drei Millimeter dicken Bodenplatten ziemlich gering ist. Diese erfüllt, auch wenn sie nur ein Drittel der Empfangseffektivität besitzt, die benötigte Belastungsgrenze. Zusätzlich haben wir in die Bodenplatte, in einem symmetrischen Muster, viele kleine Löcher hineingebohrt. Mit diesen zahlreichen Löchern haben wir nochmals das Gewicht der Bodenplatte um ein Drittel reduziert. Die Löcher stören dabei nicht die Empfangseffektivität, da sie nur etwa ein Zehntel der Größe haben, die sie haben müssten, damit es Probleme beim Empfang gäbe.

2.7 Aufgetretene Probleme

Während des Baus des CanSats sind selbstverständlich einige Probleme aufgetreten. Manche dieser Probleme waren relativ leicht zu lösen, andere erforderten eine detaillierte Recherche. Im Nachfolgenden werden einige dieser Probleme und unsere Lösungsansätze erläutert.

- **Befestigung der Dosendeckel**: Es war vorerst geplant, durchgängige Gewindestangen zu verwenden, welche durch die Wand führen und so die Deckel und die Wand miteinander verbinden. Diese Stangen kollidierten jedoch mit den Löchern für die Befestigung des Mikrokontorollers und der Sensorikplatine. Nach verhältnismäßig langem Überlegen und Ausprobieren haben wir uns dazu entschlossen, die Gewindestangen nicht durchgängig zu gestalten. Stattdessen werden sie lediglich durch den Fuß und den Kopf der Wand gesteckt und dort verschraubt. Dies hat den Vorteil, dass wir das Gewicht deutlich verringern und wir innerhalb des CanSats wesentlich mehr Freiräume haben, um Objekte zu platzieren. Nachteilig ist jedoch, dass dadurch die Stabilität verringert wird.

- **Spektrum des UV-Sensors**: Unser UV -Sensor misst die Intensität der Strahlung, welche im Spektrum 280-390 nm liegt. Dies hat die Folge, dass der Output des Sensors deutlich über dem zu erwartenden Wert liegt und es relativ schwer fällt, einen Vergleich zwischen diversen Messungen aufzustellen. Dies liegt daran, dass man nicht verifizieren kann, welche Wellenlänge mit welchem Anteil an dem Gesamtoutput beteiligt sind.

- **Intensität des Sharp-Feinstaubsensors**: Zunächst war der Sharp-Sensor dazu gedacht, die Feinstaubkonzentration in unserer Athmosphäre zu messen. Jedoch mussten wir feststellen, dass der Sensor keineswegs Feinstäube, sondern groberen Staub, wie zum Beispiel Hausstaub, misst. Dies stellte deshalb ein Problem dar, da wir bis dato das komplette Dosendesign auf den Sharp-Sensor abstimmten. Ein neuer Sensor war zwar verfügbar, konnte jedoch aufgrund des Dosendesigns nicht integriert werden, da dieser zu groß ist. Da es durchaus möglich ist mit dem Sharp-Sensor halbwegs akzeptable Werte zu erlangen, wenn man Referenzmessungen durchführt, bleibt dieser zumindest vorerst im Gesamtsystem erhalten.

- **Geeigneter Ozon-Sensor**: Es hat sich als äußert schwierig erwiesen, einen Ozon-Sensor zu finden, welcher keine lange Vorlaufzeit benötigt, verhältnismäßig klein ist und nicht übermäßig viel kostet. Daher ist es aktuell nicht mehr geplant, einen Ozon Sensor innerhalb des CanSats unterzubringen.

- **BeagleBone Black**: Zu Beginn unserer Arbeit am BeagleBone hatten wir einige Probleme mit diesem. Beispielsweise ließ sich zunächst die Python-Library für das BeagleBone nicht verwenden. Einige Pins ließen sich nicht ansteuern, was zu Fehlern führte. Letztlich fanden wir heraus, dass die Linux Distribution Debian auf dem System installiert war. Dieses ist jedoch für das BeagleBone Black noch in der Testphase und kann Bugs hervorrufen. Gelöst wurde das Problem, indem der Speicher mit dem Defaultsystem Linux Angstrom geflasht wurde. Danach funktionierte das Ausführen von Pythoncode ohne jegliche Probleme. Schließlich bootete das Beagle gegen Ende des Projektes nicht mehr und nach sorgfältiger Recherche im Internet stellte sich heraus, dass es sich um einen bekannten Fehler handelt, welcher allerdings mehrere Ursachen haben kann. Letztlich ist uns nicht bekannt weshalb das Beagle nicht mehr funkioniert, jedoch liegt die einzige Abhilfe in einem neuen Mikrocontroller. Da unser Dosendesign jedoch für das Beagle konzipiert wurde, blieb uns nichts anderes übrig als ein neues Beagle zu besorgen.

- **UART und I2C-Bus**: Bei der Übertragung mithilfe von UART und dem I²C-Bus sind wir auf Probleme in Kombination mit dem BeagleBone gestoßen. Dieses hatte verschiedene Ports und Protokolle standardmäßig nicht aktiviert. Wir mussten im Linux-System Angstrom einige Startroutinen hinzufügen, sodass bei jedem Start auch alle Ports und Protokolle aktiviert werden. Am Ende kostete dies viel Zeit, da die Dokumentation des BeagleBone in diesem Punkt nicht detailgenau war und für verschiedene Versionen verschiedene Lösungen des Problems existieren.

- **Berechnung der Fallschirmgröße**: Das Ergebnis der Größenberechnung des Fallschirms erschien uns nicht realistisch. Der genutzte Cw-Wert hat einen Formfaktor, der nicht unseren Fallschirmen entsprach. Daher wurde beschlossen, einige Tests durchzuführen, um die Werte genauer zu bestimmen (siehe Testkonzept).

- **Haltbarkeit des Gazestoffes**: Beim Bau des Fallschirms ist uns aufgefallen, dass der Stoff, der die Dose mit dem Fallschirm befestigt, eventuell nicht stark genug ist. Durch die sehr löchrige Struktur kann es schnell zu kleinen Rissen kommen. Werden diese zu stark belastet können diese sich ausbreiten.

- **Durchmesser des CanSats**: Wir sind zu Beginn des Projektes davon ausgegangen, dass der Durchmesser des CanSats dem einer standardisierten Getränkedose entspricht (67mm). Zu spät ist uns aufgefallen, dass dies ein Irrtum war, und der Durchmesser 66mm betragen muss. Dies stellt jedoch kein Problem dar, da wir die Dicke der Außenwand problemlos um 0,5mm verringern können.

- **Energieversorgung des Transceivers**: Das Transceiver-Modul unseres CanSats, ist das einzige Bauteil, welches eine 5V Spannungsversorgung benötigt. Dies stellte uns vor einige Probleme, da der 5V Output des Beagles lediglich die Betriebsspannung des Beagles ausgab, welche in unserem Fall bedingt durch die gewählten Batterien bei rund 7,2V lag. Dementsprechend würde eine überhöhte Spannung am Modul anliegen, welche dieses beschädigen würde. Letztlich wurde das Problem gelöst, indem die Spannung des Output-Pins mithilfe eines Vorwiderstandes und einer Z-Diode auf rund 5V reduziert wurde. Diese Lösung schließt Spannungsschwankungen, bedingt durch schwankende Stromstärken oder durch schwankende Eingangsspannungen am Beagle, nicht aus. Dies stellt jedoch kein Problem dar, da das Transceiver-Modul mit einer Spannung zwischen 3,5V und 5,5V arbeiten kann. Wollte man diese Schwankungen vollständig eliminieren, würde man um einen Spannungsregler nicht herum kommen, wobei mit erheblicher Verlustleistung zu rechnen wäre, welche wir nicht verkraften können da die Batterien in Bezug auf ihre Ladungsmenge bereits relativ knapp kalkuliert sind.

- **Unterbringung der Batterie**: Die Unterbringung der Spannungsversorgung unseres CanSats stellte uns gegen Ende des Projektes vor große Probleme, da wir der Unterbringung bis dato in unserem Dosendesign kaum Aufmerksamkeit geschenkt hatten. Letztlich gab es sogar Probleme ein Batteriefach für zwei Mignon-AA-Batterien unterzubringen. Dennoch schafften wir es mithilfe einer 3D-gedruckten Konstruktion das Batteriefach am Beagle selbst zu befestigen.

- **Unterbringung des Schalters**: Nachdem das Problem der Unterbringung der Spannungsversorgung gelöst war, fiel uns auf, dass wir noch einen Hauptschalter benötigen. Letztlich lösten wir das Problem, indem wir ein Batteriefach für zwei Mignon-AA-Batterien mit einem Schalter verwendeten.

- **Eine passende Batterie**: Einer der größten Nachteile am BeagleBone Black ist der vergleichsweise enorme Energiebedarf. Das Board benötigt im laufenden Betrieb ca. 200mA und während des Bootvorganges sogar kurzzeitig das Doppelte. Es gibt zwar eine Vielzahl an Lithium-Polymer (LiPo) Akkumulatoren, jedoch konnten wir keinen LiPo Akku finden, welcher die benötigte Stromstärke, Kapazität und Spannung mitbringt und zusätzlich eine Größe (bzw. ein Format) besitzt, welches sich in unser Dosendesign eingliedert. Dieses

Problem wurde schlussendlich durch zwei Mignon-AA- Batterien gelöst (näheres im Entsprechenden Kapitel). Dies ist mit Sicherheit keine optimale Lösung, für uns jedoch bei weitem die Beste.

- **Anschluss von UART und ADC Sensoren an den I²C-Bus**: Unser Plan war gewesen, alle Sensoren an den I²C-Bus anzuschließen. Daraus hätte sich ein softwareseitiger Vorteil ergeben, der die Entwicklung und die Geschwindigkeit der auf dem Mikrocontroller ausgeführten Software deutlich verbessert hätte. Jedoch entschieden wir uns aus verschiedenen Gründen dagegen. Zunächst würde jeder Sensor seinen eigenen Analog zu Digital Converter benötigen, was bei uns zu erheblichen Platzproblemen geführt hätte. Desweiteren standen wir unter zeitlichem Druck und andere Probleme erforderten dringlicher einer Lösung als die Optimierung der Übertragungsprotokolle.

2.8 Testkonzept

Um sicherzustellen, dass der CanSat problemlos funktioniert, wurden diverse Tests durchgeführt. Dazu zählt natürlich das Prüfen auf Funktionstüchtigkeit der Sensoren. Hierfür wurde jeder Sensor separat an verschiedene Mikrocontroller (BeagleBone Black, Arduino Mega) angeschlossen. Dadurch konnte verifiziert werden, dass jeder Sensor unter jedem Board den gleichen Output liefert. Zusätzlich wurde überprüft, ob die Sensoren auf eine Veränderung der zu messenden Eigenschaft reagieren. Um zu erkennen, ob die gemessenen Werte den tatsächlichen Werten entsprechen, werden diese im Umweltlabor von Atlas Elektronik getestet und kalibriert. Dort soll ebenfalls überprüft werden, wie stabil der CanSat ist, um vorherzusagen, ob er beim Aufschlag beschädigt wird. Das Testkonzept für die Sensoren beruht im Wesentlichen auf Trial and Error.

Die Antenne wird im Labor für Hochfrequenztechnik an der Universität Bremen durchgetestet. Hierzu wird die genaue Frequenz bestimmt, auf der die Antenne empfängt, und eine Impedanzanpassung durchgeführt.

Um den Fallschirm zu prüfen wurde ein Experiment durchgeführt. Dazu wurde die Zugkraft des Fallschirms gemessen. Aus einem Auto, welches mit einer Geschwindigkeit von 50 km/h fuhr, haben wir einige Testfallschirme an einem Newtonmeter befestigt und dieses aus dem Auto gehalten. Die Berechnung der Kraft sieht folgendermaßen aus:

$$F = m \cdot g$$

$$F = 300g \cdot 9,81\frac{m}{s^2} \cdot \frac{1kg}{1000g}$$

$$F = 2,943\frac{m}{s^2}kg$$

Aus der Geschwindigkeit und dem Widerstand des Newtonmeters konnte ermittelt werden, dass der Fallschirm keine 30 cm Durchmesser benötigt. Weitere Tests haben ergeben, dass 16 cm ein besserer Wert ist. Bei einem Fallschirm dieser Größe können auch schon relative kleine Veränderung der Größe die Fallgeschwindigkeit enorm beeinflussen.
Beim Test der Fallschirmgröße ist uns aufgefallen, dass die Gaze bei mehreren Versuchen kleine Risse bekommen hat. Das Material, das wir verwendet haben, ist nicht stark genug, um die Belastungen während des Fluges zu tragen. Aus diesem Grund haben wir uns dazu entschieden, die Gaze komplett wegzulassen. Stattdessen verwenden wir, wie im letzten Jahr, acht Schnüre, die an der Dosendecke befestigt werden.

Um die Datenübertragung und die restlichen Features des CanSats zu überprüfen, wird das System als Ganzes getestet. Der CanSat wird eingeschaltet, die Antenne auf den CanSat gerichtet, die Bodenstation an die Antenne angeschlossen, die Satellitenkonfiguration eingestellt, der Hotspot der Bodenstation gestartet und die Android-App mit dem Hotspot verbunden. Anhand der einzelnen Komponenten der Datenübertragung (CanSat, Desktopapplikation, Android-Applikation) lässt sich etwa erkennen, an welcher Stelle der Übertragung Fehler auftreten. Funktioniert die Übertragung über alle Komponenten mit allen Sensoren, so ist der Test erfolgreich.

3 Beschreibung der Bodenstation

3.1 Desktopapplikation

3.1.1 Überblick

In diesem Teil der Dokumentation werden wir die Bodenstation vorstellen, welche als Datenempfänger und als Datenverarbeitungsplattform fungiert.
Die zentrale Aufgabe der Bodenstation ist es, die Daten, welche vom Satelliten gesammelt werden, zusätzlich sicher am Boden zu speichern. Sollte der Satellit, und damit auch die lokal gespeicherten Daten, verloren gehen, so existiert eine Kopie in unserem System.
Zusätzlich zur Datensicherung erfüllt die Bodenstation die Aufgabe, die empfangenen Daten auf verschiedene Arten zu visualisieren und somit dem Nutzer direkt während der Datenübertragung die Möglichkeit zu verschaffen, die Daten zu beobachten und diese zu analysieren.
Die Bodenstation ermöglicht es außerdem, dass gesicherte Daten auch nach der Datenübertragung noch betrachtet und analysiert werden können.
Unser Ziel bei der Entwicklung der Bodenstation war es, eine modulare und anpassbare Plattform zu entwickeln, welche nicht nur mit unserem Satelliten, sondern mit vielen verschiedenen Satelliten genutzt werden kann, ohne dass ein großer Konfigurationsaufwand besteht.
Um dies zu ermöglichen, haben wir die Bodenstation in mehrere Dimensionen skalierbar entwickelt, was es im Endeffekt sehr einfach macht, neue Satelliten und verschiedene Übertragungsprotokolle zur Bodenstation hinzuzufügen.
Eine Nutzeranleitung der Bodenstation kann unter 7.4 gefunden werden.

3.1.2 Nutzerfreundlichkeit

Die Bodenstation wurde so entwickelt, dass der Nutzer der Bodenstation sich nicht um Implementationsdetails kümmern muss und die Bodenstation als zentralen Empfänger für Daten von seinem Satelliten nutzen kann, ohne dabei etwas anderes als die grafische Benutzeroberfläche zu verwenden.

Ein wichtiger Faktor im Bereich der Nutzerfreundlichkeit ist die dynamische Benutzeroberfläche, welche es dem Nutzer erlaubt, verschiedene GUI-Komponenten in Teilpanels innerhalb der Applikation anzuzeigen und umzustellen. Diese Dynamik ermöglicht es dem Nutzer, die Benutzeroberfläche, welche für die Analyse seiner Daten am Besten ist, mithilfe der verschiedenen Panels einzustellen.

Alle Visualisierungskomponenten sind zudem intuitiv aufgebaut, sodass es nicht kompliziert ist, sich seine Daten mithilfe der Visualisierungskomponenten anzusehen. Die Anzeige über den 3D-Globus erlaubt es beispielsweise, den Globus beliebig zu bewegen und die Position auf dem Globus zu verändern, während der Graph es erlaubt, dass die Axen des Graphen beliebig ausgetauscht werden können.

3.1.3 Erweiterbarkeit

Bei der Entwicklung der Bodenstation haben wir darauf geachtet, dass die Bodenstation auf einer skalierbaren Architektur aufgebaut ist. Dies ermöglicht es, leicht neue Module und Funktionalitäten zur Bodenstation hinzuzufügen, ohne dabei besonders viel Code ändern zu müssen. Auf die folgenden Weisen ist die Bodenstation skalierbar:

- Neue GUI-Komponenten können sehr leicht hinzugefügt werden. Das Erstellen und Einbinden eines GUI-Komponenten umfasst lediglich die Erstellung eines neuen Netbeans-TopComponents.
- Unterschiedliche Satelliten können ohne eine erneute Kompilierung hinzugefügt und verändert werden, indem man die Konfigurationsdateien der Bodenstation anpasst, welche zur Laufzeit der Bodenstation geladen werden.
- Neue Konfigurationsformate können leicht hinzugefügt werden, indem man die neue Konfigurationsimplementation unter einem Interface in der Applikation hinzufügt.

- Es ist ohne viel Aufwand möglich, die verschiedenen Datenquellen, aus denen Daten bezogen werden, auszutauschen. Möchte man also die Daten von einer anderen Quelle als einem USB-Port beziehen, so ist dies leicht zu implementieren, indem man lediglich eine neue DataSource implementiert und zur Applikation hinzufügt.
- Verschiedene Datenübertragungsformate können ebenfalls ohne viel Aufwand hinzugefügt werden, indem neue Formate unter dem DataFormat-Interface implementiert werden. Die Applikation ist also nicht auf JSON als Übertragungsformat limitiert.
- Die verschiedenen Datenempfänger können auch leicht angepasst werden, indem man neue Datenempfänger unter dem Receiver-Interface implementiert, wodurch die verschiedenen Logging-Formate erweitert werden können.
- Daten können aus beliebigen Dateiformaten importiert werden, was ebenfalls leicht erweiterbar ist, indem man neue Import-Dateiformate über das Importer-Interface implementiert.
- Export-Formate können leicht erweitert werden, indem man neue Export-Formate unter dem Exporter-Interface implementiert.

3.1.4 Features

Da die Software der Bodenstation auf dem Framework Netbeans Platform basiert, lassen sich einzelne graphische Module kombinieren, welche sich per "drag and drop" verschieben lassen. Die Größe und Position dieser Module und des gesamten Frames lassen sich beliebig verändern. Des Weiteren bietet die Software die Möglichkeit, Daten von verschiedenen Satelliten zu empfangen. Empfangene Daten lassen sich mittels der graphischen Oberfläche in Graphen anzeigen, welche sich verschieden kombinieren lassen. Außerdem lassen sich die empfangenen Daten zwischenspeichern und anschließend in verschiedene Dateiformate exportieren oder live in einer Datei loggen. Diese exportierten oder geloggten Daten lassen sich anschließend wieder einlesen und anzeigen. CSV, TXT, JSON, KML und PNG sind Dateiformate, in welche exportiert werden kann. Davon lassen sich exportierte CSV- und JSON-Dateien wieder einlesen und visualisieren. TXT-Dateien werden formatiert und somit gut leserlich für den Nutzer exportiert, während exportierte KML-Dateien per Google Earth geöffnet und graphisch visualisiert werden können. Ein weiteres Feature der Bodenstationsoftware ist die Datenvisualisierung per NASA World Wind als Modul in der graphischen Oberfläche. Diese Visualisierung zeigt den Flug des Satelliten auf einem virtuellen Globus mittels Satelliten- und Luftbildern. Zudem zeigt sie auf jeder gemessenen GPS-Koordinate die gemessenen Werte der Sensoren des Satelliten. Unter anderem bietet dieses Modul der Software die Möglichkeit, an die virtuelle Erdkugel heranzuzoomen und einzelne Elemente dreidimensional darzustellen. Darstellungen mittels dieses virtuellen Erdballs sind sowohl in Echtzeit mittels eines Streams vom Satelliten als auch als Import aus einer Datei möglich.

3.2 Android-Applikation

3.2.1 Kurzbeschreibung

Die Funktion der Android-App ist es, die Datenpakete, die von der Bodenstation über einen Hotspot gesendet werden, zu empfangen, und live, in einem passenden Graphen, anzuzeigen. Dabei wird Wert darauf gelegt, dass es möglich ist, alle Werte die gesendet werden, einzeln oder in Gruppen darzustellen. Dies ermöglicht, dass alle Daten verglichen werden können. Zusätzlich wird in einem Balkendiagramm die Differenz zwischen dem höchsten und dem niedrigsten gemessenen Wert dargestellt. Diese Differenzen werden in grün für positive Veränderungen und in rot für negative Veränderung dargestellt. Nebenher sind ebenfalls Optionsmöglichkeiten vorhanden, um die Graphen nach den Wünschen des Nutzern zu gestalten.

3.2.2 Funktionen

Die App bietet insgesamt folgende Funktionen:

- Anzeigen der Werte in einem Livegraphen
- Verwaltung der angezeigten Werte im Graphen während der Laufzeit
- Anzeigen von Differenzen von Werten im Balkendiagramm
- Manuelle Start/Stop Funktion für das Balkendiagramm
- Einstellung zur Geschwindigkeit des Graphen
- Einstellung zur Regelung der Anzahl der Werte, welche gleichzeitig angezeigt werden
- Die Möglichkeit, alle Funktionen mit einem Debug-Stream zu testen

4 Projektplanung

4.1 Zeitplan der CanSat-Vorbereitung

Insgesamt wurde die Zeitplanung an verschiedenen Meilensteinen orientiert, welche sich auch in den spezifischen Zeitplanungen der Teilgruppen wiederfinden lassen. So war zum Beispiel geplant, einen guten Zwischenstand mit einem Prototypen bereits im Mai zu erreichen, wobei dieser Zeitplan von Anfang an kritisch gesehen wurde und letzten Endes auch nicht erreicht wurde. Trotzdem blieb nach diesem Termin noch genügend Zeit, das Projekt fertigzustellen, was leider letzten Endes die geplante Testphase etwas verkürzt.
Das gesamte Management der Arbeitspakete und des Zeitaufwandes wurde mit der Projektmanagementsoftware Redmine erledigt. Da diese auf unserem Server unter redmine.gamma-team.de erreichbar ist, kann jedes Teammitglied zu jedem Zeitpunkt den Fortschritt der Arbeit verfolgen. Die Planung der beiden Halbgruppen ist größtenteils voneinander getrennt. Es gibt jedoch gemeinsame Meilensteine, welche von beiden Gruppen eingehalten werden sollen. Bevor die Arbeit der Halbgruppen begonnen hat, gab es eine allgemeine Projektfindungsphase. In dieser Phase wurde ein grober Zeitplan festgelegt und es wurden alle relevanten Systeme (Webserver, Projektmanagementsoftware, GitLab etc.) aufgesetzt und eingerichtet um später einen reibungslosen Ablauf der Arbeitsphase zu garantieren. Die Idee und die Spezialisierung der Idee für das gesamte Projekt entstand ebenfalls in dieser Zeit. Anschließend wurde eine separate Zeitplanung in den beiden Halbgruppen erstellt, welche im Nachfolgenden erläutert wird.

4.1.1 Zeitplan der Hardware-Gruppe

Innerhalb der Hardwaregruppe wurde versucht, die meisten Aufgaben zu parallelisieren. Jedes Teammitglied hat sein eigenes spezielles Aufgabengebiet. Zwischen den Teammitgliedern herrscht trotzdem ein stetiger Austausch. Grund für die Parallelisierung war, dass in unseren Augen die meisten Aufgaben nur die Aufmerksamkeit einer Person benötigen. Es ist nur selten erforderlich, dass mehrere Teammitglieder an demselben Arbeitspaket arbeiten müssen. Der gesamte Arbeitsprozess wurde in diverse Abschnitte gegliedert. Diese Abschnitte lassen sich auch im GANTT-Diagramm im Anhang dieses Dokumentes wiederfinden. Die einzelnen Abschnitte sind in diverse Arbeitspakete unterteilt, Personen zugewiesen und mit einem Zeitraum versehen. Bei den Abschnitten handelt es sich um folgende:

- Planung: Erstellung von Arbeitspaketen sowie eine Verteilung dieser und eine Erstellung diverser Diagramme
- Fallschirm: Gestaltung und Bau des Bergungssystems
- Sensorik: Heraussuchen, Bestellen und Testen passender Sensoren für unser Projekt
- Beagleboard: Festlegung der Programmiersprache, IDE und der Recherche zu den elektrotechnischen Eigenschaften des Boards
- Dose: Design und Bau der Hülle und der Deckel der Dose
- Dosenmanagement: Design und Bau des Inneren der Dose sowie die Integration der Sensoren in das Gesamtsystem

Das GANTT-Diagramm der Hardware-Gruppe kann auch unter 5 gefunden werden.

4.1.2 Zeitplan der Software-Gruppe

In der Softwaregruppe haben wir uns, wie in der Hardwaregruppe, dafür entschieden, die Aufgaben untereinander zu verteilen. Dabei haben wir zuerst die Bodenstation und die Android-Applikation komplett voneinander getrennt. Die Bodenstation war von Beginn an fester Bestandteil unseres Projektes. Die Android-App kam erst später hinzu und musste daher separiert behandelt werden. Noch vor dem eigentlichen Start des Projektes wurde diskutiert, wie die Software aufgebaut sein soll und welche Technologien für die Bodenstation verwendet werden kann.

Nachdem die ersten Entscheidungen getroffen waren, haben wir angefangen, das Projekt in verschiedene Meilensteine zu unterteilen. Daraus ist folgende Gliederung entstanden:

- Erstellung einer detaillierten Ticketübersicht
- Basisversion, mit allen Tickets, welche zur Bereitstellung der ersten Features nötig sind
- Export, mit allen Tickets, welche zum Exportieren von Daten nötig sind
- Kartenvisualisierung, mit allen Tickets, welche für das Darstellen des Satellitenfluges nötig sind
- Wissenschaftliche Analyse, mit allen Tickets, welche zur wissenschaftlichen Analyse der gesammelten Daten nötig sind
- Finale Version, mit allen Tickets, welche zur Zusammenführung der Bodenstation nötig sind
- RC 1 v1.1, mit allen Tickets, welche für den letzten Schliff der Bodenstation nötig sind

Innerhalb dieser Meilensteine haben wir das Projekt daraufhin in einzelne Tickets unterteilt. Jedes dieser Tickets spiegelt eine einzelne Aufgabe zur Fertigstellung der Bodenstation wieder. Diese Aufteilung hat es uns ermöglicht, die Aufgaben innerhalb der Bodenstation relativ flexibel zu verteilen. Dies hat dazu beigetragen, dass selten jemand auf eine andere Aufgabe warten musste. Die Tickets, welche sich innerhalb der Meilensteine befinden, haben sich während der Durchführung des Projektes immer wieder verändert. Dies war abhängig von den Ansprüchen, welche sich in dem jeweiligen Moment ergeben haben. Das im Anhang unter 6 vorhandene GANTT-Diagramm gibt also nicht nur unsere Planung zum Anfang des Projektes wieder, sondern auch die kontinuierliche Präzisionsplanung während der Durchführung des Projektes.

Die Idee der App entstand deutlich später als die des restlichen Projektes. Daher wurde bei der Planung in Wochenschritten gedacht. Durch diese Methode konnten alle zwei Wochen kontrolliert werden, ob eine Komponente fertiggestellt ist. Falls mehr Zeit benötigt wurde, so war es möglich, am Wochenende an der Android-App zu arbeiten. Zusammengefasst existieren insgesamt sieben Arbeitspakete:

- Der Debugger, der die Livedaten simulieren soll
- Liniengraph GUI
- Liniengraph Logic
- Balkendiagramm GUI
- Balkendiagramm Logic
- Optionen
- Menü

Das GANTT für die Android-Applikation lässt sich unter 7 finden.

4.2 Einschätzung der Mittel

4.2.1 Budget

Um das CanSat Projekt zu finanzieren, konnten wir aktuell noch keine Sponsoren finden. Jedoch konnten wir uns mit unserem Schulverein verständigen, welcher uns finanziell unterstützen wird. Da wir nicht auf das T-Minus-Kit zurückgreifen, sondern stattdessen ein anderes Mikrokontrollerboard verwenden, können wir ungefähr 150€sparen. Der 200€Watterot Gutschein, welcher vom Wettbewerb gestellt wird, ist in unserer Rechnung noch nicht inbegriffen. Dies liegt daran, dass noch nichts bei Watterot bestellt wurde, bzw. die Bestellung lange vor der Annahme am Wettbewerb getätigt wurde. Im Nachfolgenden sind alle Ausgaben und Einnahmen aufgelistet.

Ausgabe	Datum	Empfänger	Grund
-12,16 €	08.01.2015	Watterott	BMP180 Breakout
-28,99 €	09.01.2015	eBay - rcskymodel	Ultimate GPS
-14,32 €	10.01.2015	Spark Fun Electronics	UV-Sensor
-51,99 €	10.01.2015	Amazon	BeagleBone Black
-17,30 €	01.12.2014	eBay - hdt-preiswert	GFK-Set 1kg Polyesterharz + 20g Härter + $2m^2$ Glasfasermatte
-3,54 €	23.03.2015	toom baumarkt	6 x Schleifpapier
-3,79 €	23.03.2015	toom baumarkt	Filzrolle
-4,49 €	23.03.2015	toom baumarkt	Plüschwalzen
-2,19 €	23.03.2015	toom baumarkt	Mundschutz
-1,99 €	23.03.2015	toom baumarkt	Farbwanne
-4,99 €	23.03.2015	toom baumarkt	Einmalhandschuhe
-133,04 €	02.06.2015	Alexander Brennecke	Rückzahlung bisheriger Ausgaben
-139,80 €	23.03.2015	Amazon	2x BeagleBone Black
- 145,75 €			

Tabelle 4: Ausgaben

Einnahmen	Datum	Absender	Grund
17,30 €	01.12.2014	Alexander Brennecke	GFK-Kauf
107,46 €	10.01.2015	Alexander Brennecke	Sensorenkauf
20,99 €	23.03.2015	Alexander Brennecke	toom Einkauf
133,04 €	02.06.2015	Schulferein der Europaschule SII Utbremen	Sponsoring
145,75 €			

Tabelle 5: Einnahmen

4.2.2 Externe Unterstützung

Externe Unterstützung erhielten wir von vielen Lehrern unserer Schule, welche uns Fragen zur Elektrotechnik und Softwareprogrammierung beantworten konnten. Zusätzlich haben wir finanzielle Unterstützung durch den Schulverein unserer Schule erhalten (siehe 4.2.1). Unterstützung außerhalb unserer Schule erhielten wir durch folgende Personen/Organisationen:

- Das Hackerspace Bremen e.V., welches uns ihren 3D-Drucker zur Verfügung gestellt hat. Zusätzlich konnten wir dort unsere Platine ätzen.
- Prof. Martin Schneider vom Hochfrequenzlabor der Universität Bremen, welcher uns geholfen hat, unsere Antenne an die Frequenz und die Wellenimpedanz anzupassen.
- Das Umweltlabor der Atlas Elektronik GmbH hat uns geholfen, den CanSat, hinsichtlich seiner Stabilität, zu testen und die Sensoren korrekt zu kalibrieren.

5 Öffentlichkeitsarbeit

5.1 Website

Unsere Website Team Gamma wurde bereits für den Europäischen CanSat Wettbewerb 2014 verwendet. Diese haben wir weiter geführt und dort in unregelmäßigen Abständen aktuelle Informationen über das Projekt veröffentlicht. Da die Website durch den europäischen Wettbewerb bei anderen europäischen Teams bekannt ist, wird die Website in Englisch geführt. Man findet dort zusätzlich einige Dokumente, Fotos und Videos. Die Informationen auf der Website sind meist relativ detailliert verfasst.

5.2 Ausstellung am MINT-Projekttag unserer Schule

Im Schuljahr 2015/2016 findet an unserer Schule ein Tag der MINT Projekte statt. Dieser Tag wird von einer Schülergruppe unserer Parallelklasse organisiert. Wir möchten an diesem Tag natürlich auch unser Projekt vorstellen.

5.3 Logo

Das Logo wurde ebenfalls aus Gründen der Wiedererkennbarkeit aus dem vorherigen Jahr übernommen. Das Aussehen des Logos wurde von drei Faktoren beeinflusst:

- Das Zeichen in der Mitte soll dem Gamma Logo ähneln, welches zu unserem Teamnamen passt
- Das Zeichen soll zusätzlich, wenn man es um 180° dreht, dem Lambda Logo ähneln. Da wir bei dem Entwurf unserer Antenne immer wieder auf Lambda gestoßen sind, sind daraus diverse interne Späße entstanden, die wir in das Logo einfließen lassen wollten.
- Das Logo des Computerspiel Halflife, welches von einigen Teammitgliedern gespielt wird

6 Anforderungen

Die folgenden Werte sind Momentanwerte, welche sich durchaus noch ändern können. Einige Werte wurden bisher nicht weiter durch den Wettbewerb verifiziert, weshalb wir darauf verzichten mussten, diese einzubringen.

Anforderung	**Messwert**
Masse des CanSat	230 g
Höhe des CanSat	115 mm
Durchmesser des CanSat	67 mm
Länge des Bergungssystems (vgl. Pkt. 2 Anhang 1)	400 mm
Planmäßige Flugzeit	xy Sekunden
Berechnete Sinkgeschwindigkeit	15 m/S
Genutzte Funkfrequenz	434 mHz
Energieverbrauch	2833.4 mW
Gesamtkosten	145,75 €

Tabelle 6: Anforderungen an den CanSat

7 Anhang

7.1 Einleitung

7.1.1 Blockdiagramm

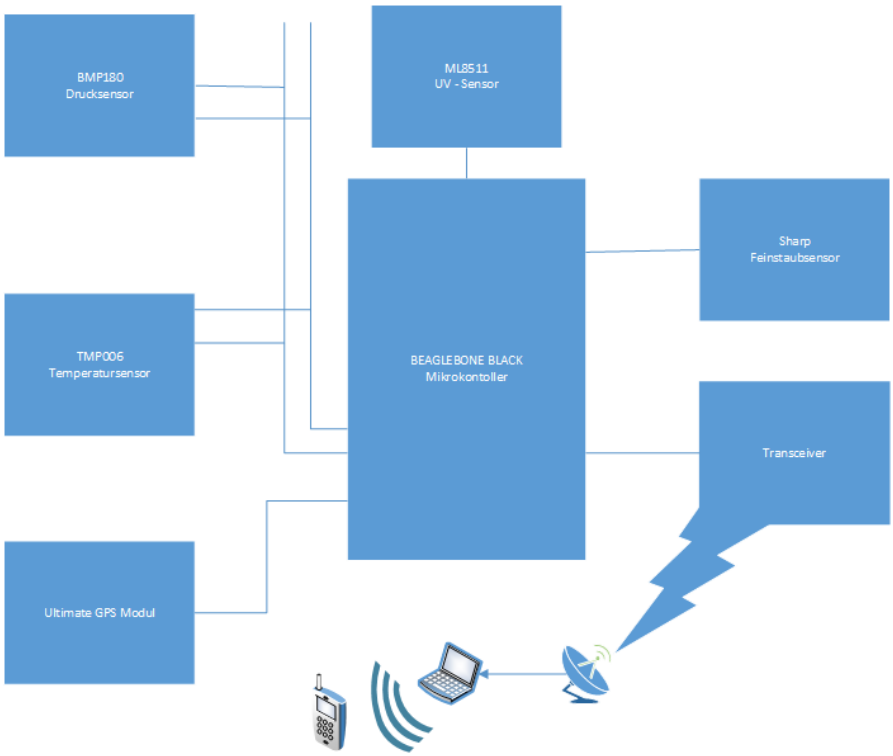

Abbildung 4: Blockdiagramm vom CanSat

7.2 GANTT-Diagramme

7.2.1 Hardware-GANTT

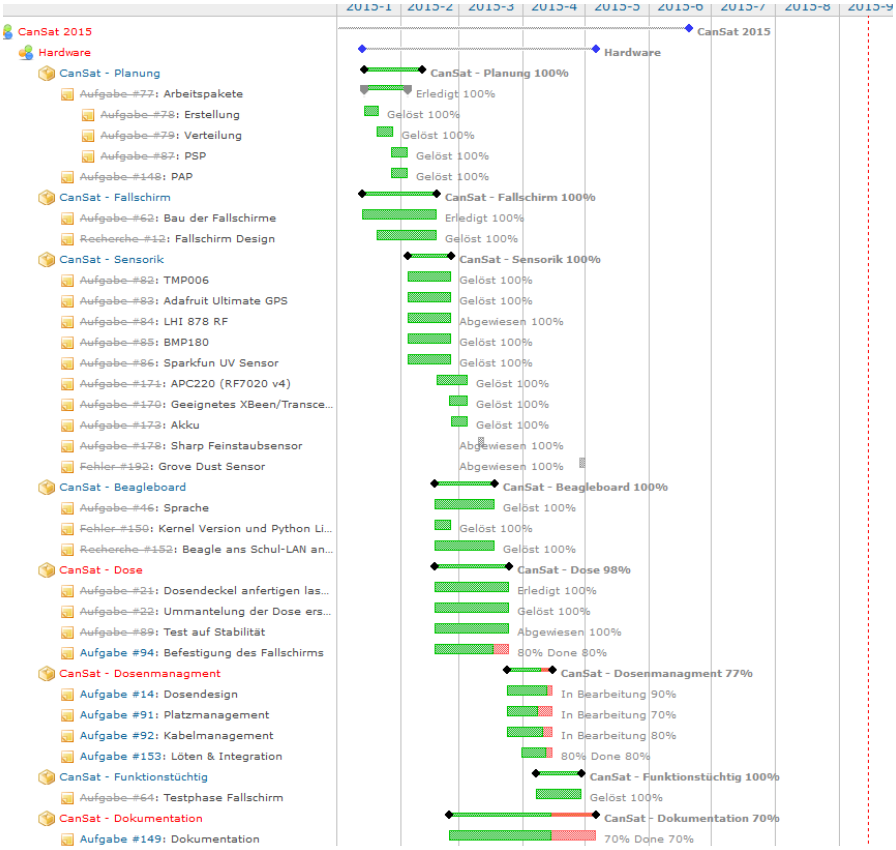

Abbildung 5: Das GANTT-Diagramm der Hardware

7.2.2 Bodenstation-GANTT

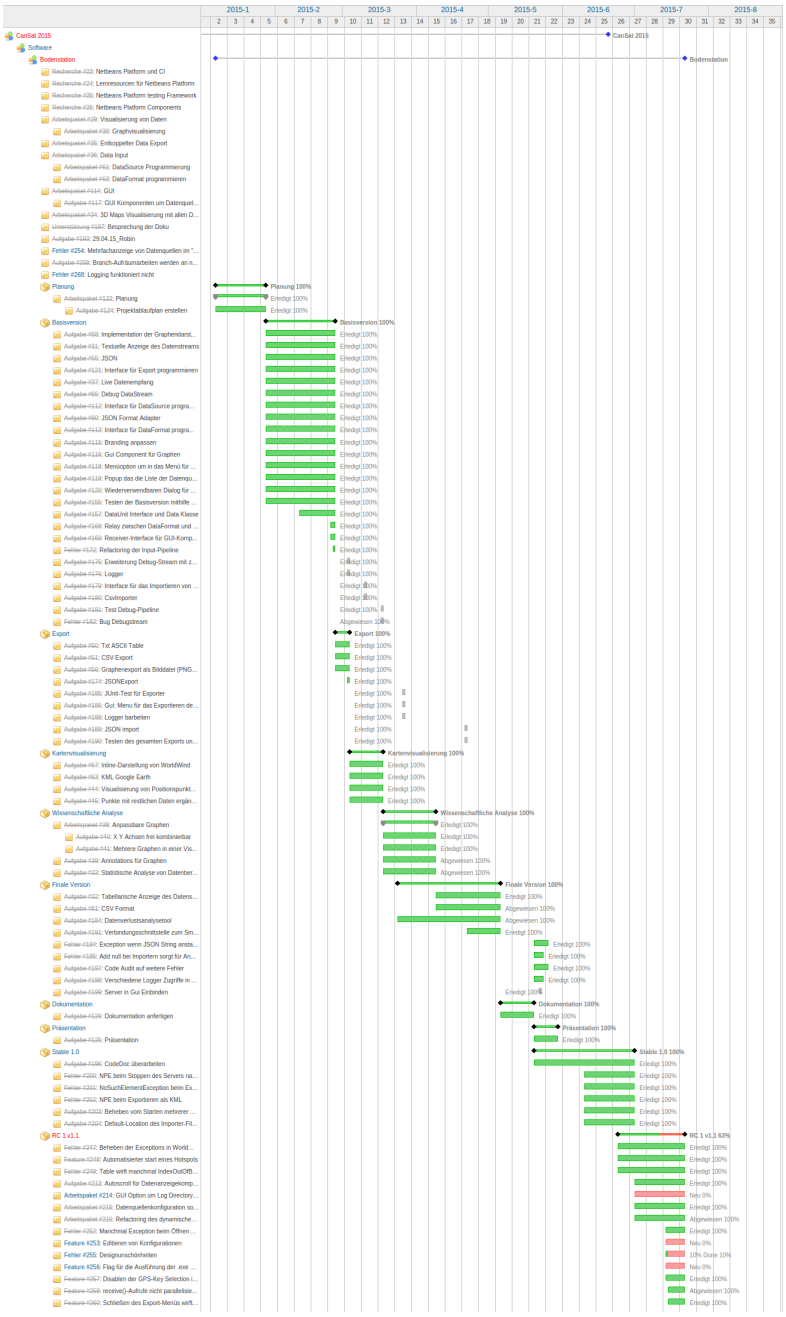

Abbildung 6: Das GANTT-Diagramm der Bodenstation

7.2.3 Android-App-GANTT

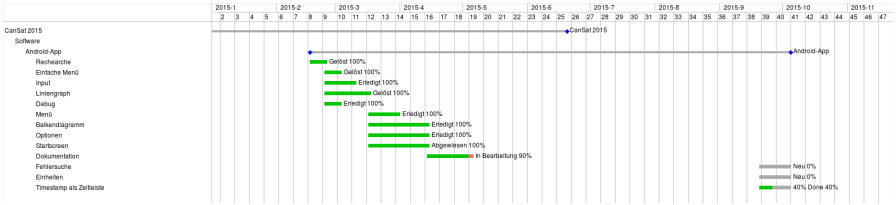

Abbildung 7: Das GANTT-Diagramm der Android-App

7.3 Der CanSat

Abbildung 8: Die Hülle und ein Dosendeckel

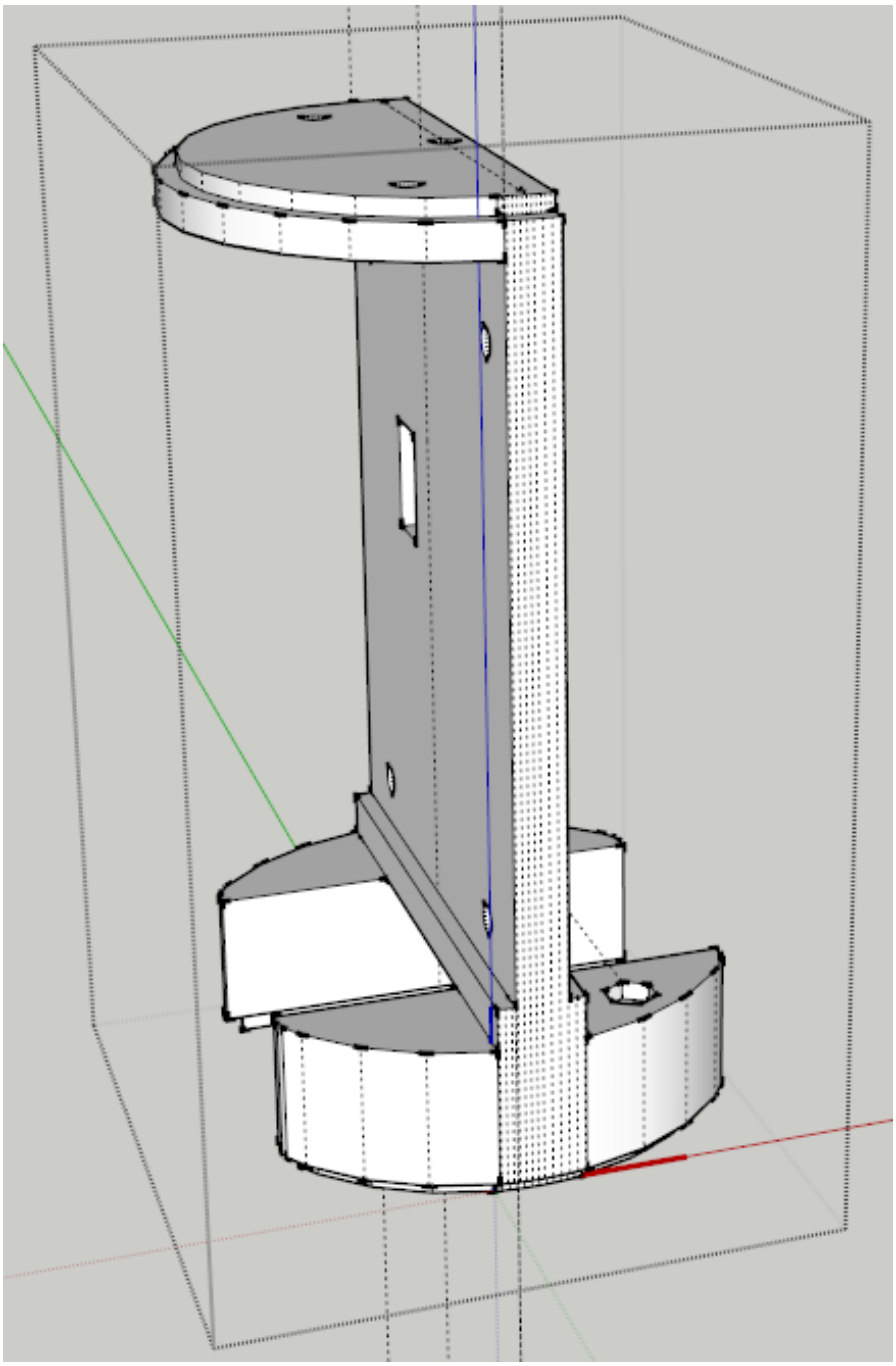

Abbildung 9: Screenshot der Zwischenwand aus Sketchup

Abbildung 10: Der Satellit (Diese Zeichnung ist möglicherweiße nicht sichtbar, da es eine 3D-Zeichnung ist. Bitte verwenden Sie den Adobe Acrobat Reader)

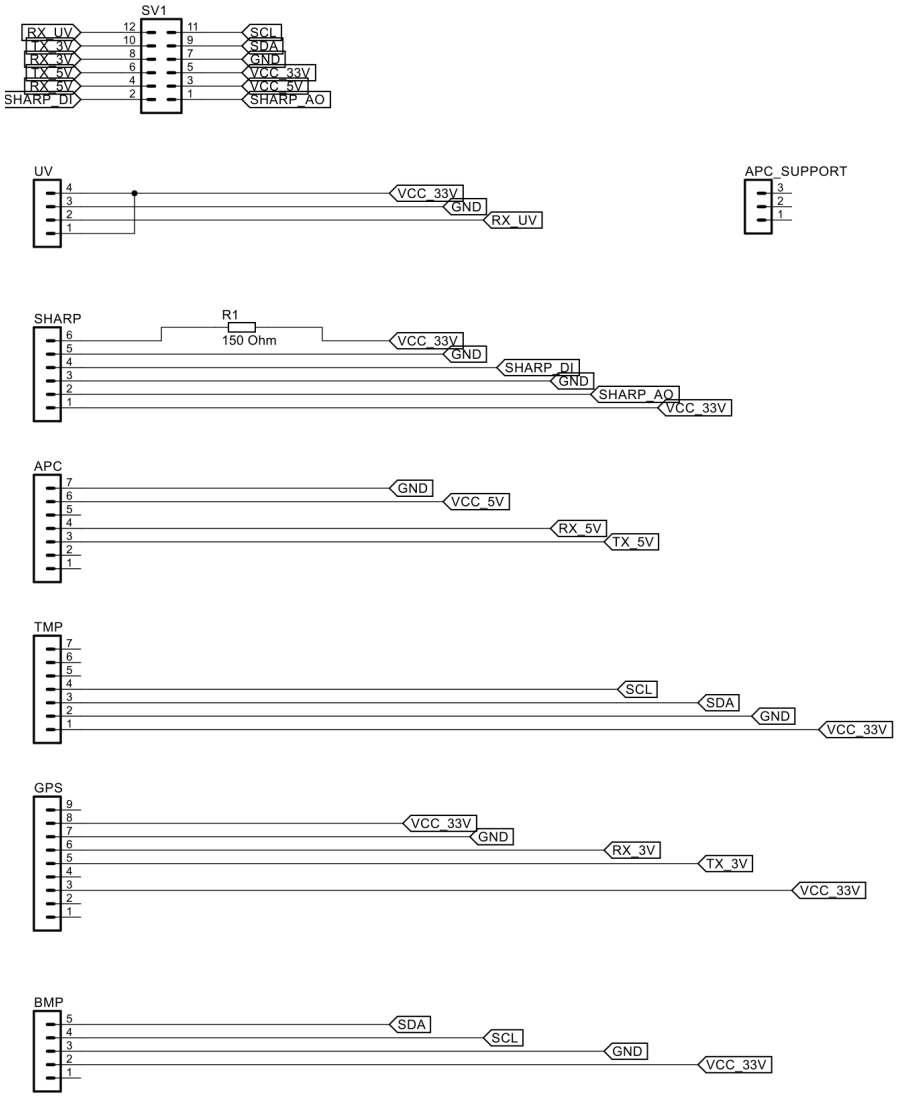

Abbildung 11: Der Schaltplan der Sensorikplatine

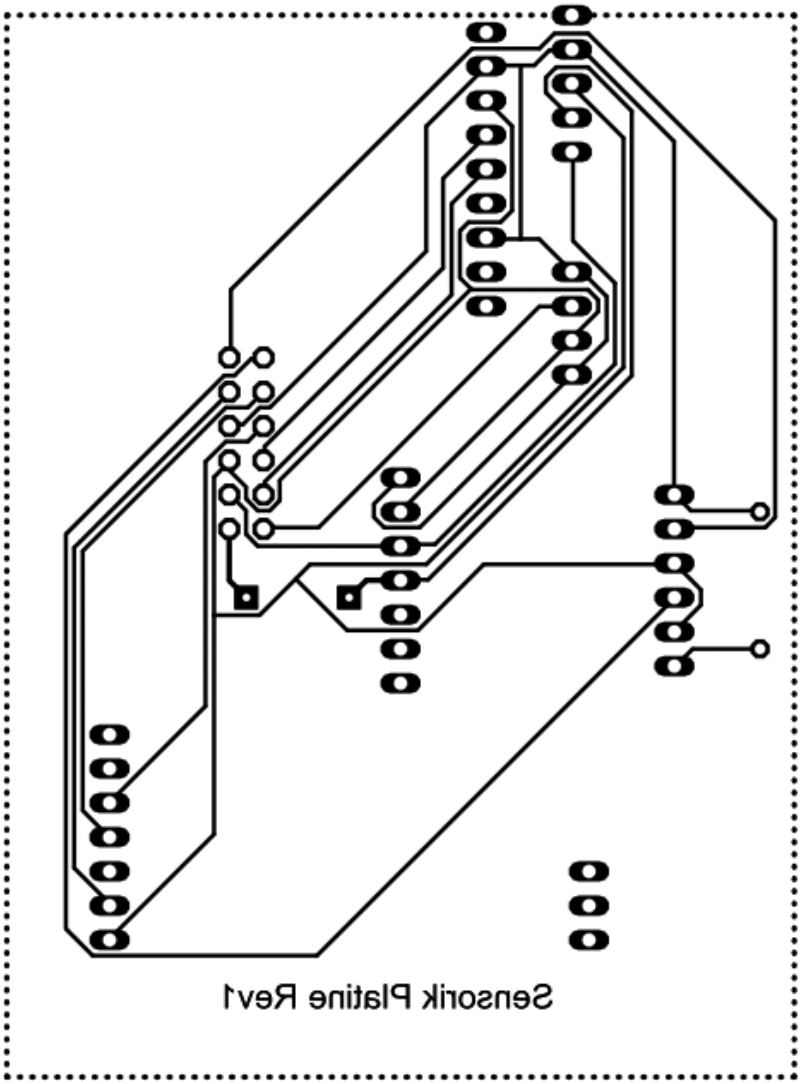

Abbildung 12: Das Layout der Sensorikplatine

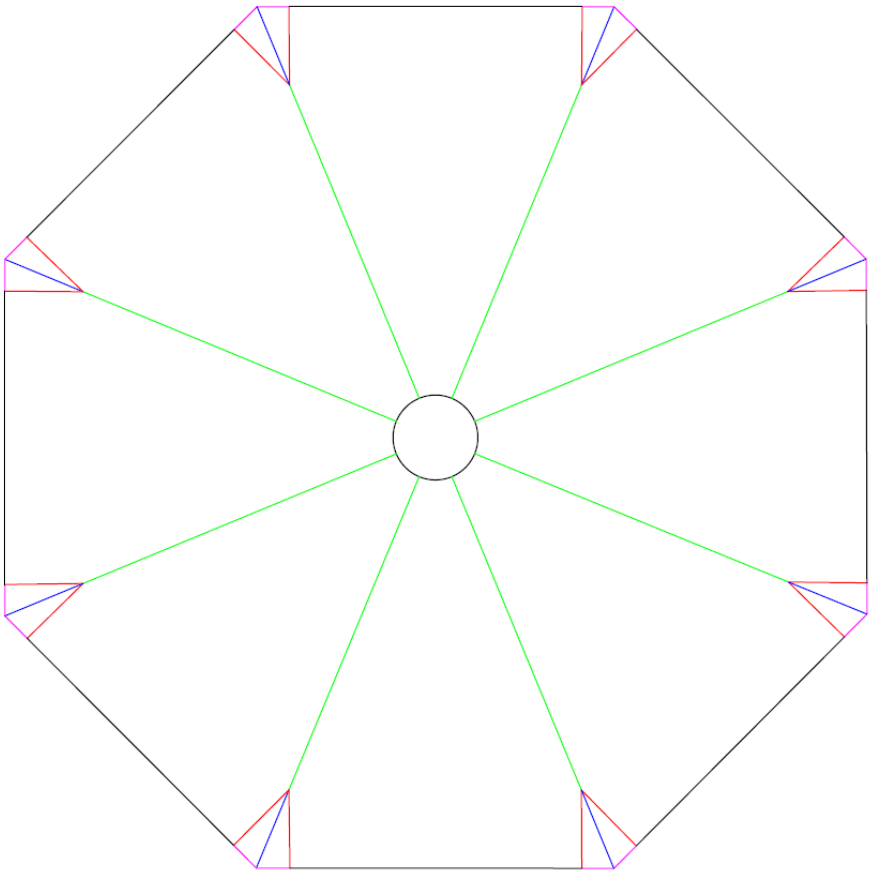

Abbildung 13: Skizze des Fallschirms

7.4 Bodenstationsnutzeranleitung

7.4.1 Datenempfang

Um Daten in Echtzeit zu empfangen, muss eine Verbindung zu einem Satelliten aufgebaut werden. Hierzu muss generell zu allererst ein Satellit erstellt werden. Hierfür wählt man unter dem Menüpunkt “File” den Unterpunkt “Satellites” aus. Dort ist es möglich, unter “Add”, einen Satelliten hinzuzufügen. Um einen Satelliten zu laden, wählt man im selben Unterpunkt “Manage” aus. Dort wählt man den Satelliten aus, von welchem man Daten empfangen will. Anschließend sieht man einen Dialog mit Konfigurationsmöglichkeiten für den Datenempfang vom Satelliten (z. B. Serieller Port). Wenn man die Konfiguration abgeschlossen hat, beendet man den Dialog mit einem Klick auf “Load configuration”. Möchte man den Datenempfangen beginnen, dann klickt man auf den “Play”-Button in der Toolbar, welcher die Datenübertragung startet. Ab hier können verschiedene Visualisierungen geöffnet werden, um die empfangenen Daten darzustellen.

7.4.2 Datenimport

Um Daten aus einer Datei zu importieren, wird im Menüpunkt “File” der Untermenüpunkt “Import” ausgewählt. Anschließend wird eine Datei ausgewählt, welche importiert werden soll. Mit der Bestätigung werden die Daten dieser Datei eingelesen.

7.4.3 Datenexport

Für das Exportieren der Daten gilt, dass alle aktuell geladenen Daten exportiert werden. Darunter fallen entweder zwischengespeicherte Daten einer Liveübertragung, oder Daten, welche aus einer Datei importiert wurden. Zum Exportieren der Daten wird im Menüpunkt “File” der Untermenüpunkt “Export” ausgewählt. Unter diesem Menüpunkt ist das Datenformat wählbar, in welches die gesammelten Daten gespeichert werden (Beim Graphenexport ist es wichtig, dass die Graphenvisualisierung während des Vorgangs geöffnet ist). Anschließend ist ein Pfad und ein Name wählbar, unter dem die Datei gespeichert wird. Mit der Bestätigung werden die Daten exportiert.

7.4.4 Datenweiterleitung

Per Klick auf das rote runde Icon in der Toolbar wird ein Hotspot-Server gestartet. Dieser sendet empfangene Daten in Echtzeit an alle Clients weiter, die mit dem Hotspot-Server verbunden sind.

7.4.5 Oberflächenpersonalisierung

Der Oberfläche können einzelne Komponenten hinzugefügt und entfernt werden. Diese Komponenten können unterschiedlich angeordnet werden. Um Visualisierungs-Komponenten hinzuzufügen wird unter dem Menü “Window” eine Visualisierung ausgewählt. Um einen der bestehenden Komponenten zu entfernen wird das Kreuz angeklickt, welches sich am Tab des Komponenten befindet. Per “drag and drop” können diese Komponenten neu angeordnet werden, dazu muss der Tab des Komponenten ausgewählt werden. In verschiedenen Bereichen können Komponenten angeheftet oder verschoben werden. Außerdem können diese übereinander verlagert werden um sie in verschiedenen Tabs und in dem selben Bereich zu verwalten.

7.4.6 Kartenvisualisierung

Die Kartenvisualisierung startet über den Untermenüpunkt “Map Visualization” im Menüpunkt “Window”. Geladene Werte werden dort angezeigt. Einzelne Messpunkte sind mit einem Punkt gekennzeichnet und mit einer Linie verbunden, was den Flugweg des Satelliten anzeigt.

7.4.7 Graphenvisualisierung

Um einen Graphen zu erzeugen, wählt man unter dem Menüpunkt “Window” - “Graph Visualization” aus. Anschließend wird in der Oberfläche ein Graph erzeugt. Die Achsen des Graphs

sind mit Sensorwerten belegbar. Um die Belegung der Achsen zu verändern, wählt man an den Achsen den jeweiligen Sensor aus und drückt den Button “Refresh axes”.

7.4.8 Textdarstellung

Zusätzlich können die empfangenen Daten auch direkt dargestellt werden, indem man den Menüpunkt “Window” - “Text Stream” auswählt.

7.4.9 Tabellendarstellung

Die empfangenen Daten können ebenfalls in einer Tabelle dargestellt werden, welche über den Menüpunkt “Table Visualization” geöffnet werden kann.

7.4.10 Fenster zurücksetzen

Die Anordnung der Komponenten der Oberfläche kann im Menüpunkt “Window” unter “Reset Windows” zurückgesetzt werden.

7.4.11 Beenden des Programms

Um das Programm zu beenden, gibt es zwei Möglichkeiten. Zum einen wird das Programm beendet, wenn das Kreuz am oberen rechten Rand der Oberfläche angeklickt wird. Zum anderen kann das Programm über den Menüpunkt “File” geschlossen werden, in dem man dort “Exit” auswählt.