CanSat 2015 Team Gamma Dokumentation

Alexander Brennecke Till Schlechtweg Marc Huisinga
Robin Bley Steffen Wißmann Alexander Feldmann
Kevin Neumeyer

4. Mai 2015

Inhaltsverzeichnis

1 Einleitung

1.1 Die Idee

Die Idee hinter dem gesamten Projekts bezieht sich auf die extremen 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 EU-Kommission droht mit einer Klage 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 druchführen um die tatsächlichen Werte zu bestimen. Der CanSat Wettbewerb eignet sich optimal dazu, da er uns die Möglichkeit bietet die Messungen nicht nur auf dem Boden sondern in verschiedenen Schichten der Athmospähre 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 heruaszufinden, ob es einen Zusammenhang zwischen ihnen gibt. Beispielsweiße ist herauszufinden, ob ein höherer Ozon Gehalt gleichzeitig einen niedrigereren Feinstaubgehalt mit sich bringt.

Zusätzlich zum Bau des Messystems 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 Sateliten, auszuwerten. Die Werte sollen dabei anschaulich und in Abhängigkeit zueinander dagestellt werden.

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

1.2 Das Team

Das gesamte Team besteht aus sieben Schülern und zwei betreuenden Lehrern. Die sieben Schüler sind jedoch intern in mehrere kleinere Teams aufgeteilt. Innerhalb der Teams ist jedoch 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 Arbeit der Gruppen und der einzelnen Personen werden im folgenden erläutert:

Das Hardware Team besteht aus drei Personen, welche sich um den Bau des Sateliten selber, dem Design und dem Bau der Dose sowie der Programmierung

des Mikrocontrollers kümmern. Zu disem 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 den ausgewählten Sensoren.

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

Das Software Team besteht aus vier Personen, welche sich um das Programmieren des Analysetools und der Android Applikation kümmern. Dieses Team besteht aus folgenden Personen:

Robin Bley

Alexander Feldmann

Marc Huisinga

Kevin Neumeyer

Zudem gib es ein Team, bestehend aus Alexander Brennecke und Till Schlechtweg, zur Organisation, Kommunikation mit Sponsoren und Öffentlichkeitsarbeit.

2 Der CanSat

2.1 Einleitung

Wir haben uns für den Satelliten überlegt, dass dieser so weit wie möglich individuell sein sollte. Daher greifen wir nicht auf das, vom Wettbewerb bereitgestelte 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 plazieren können. Zusätzlich möchten wir nicht auf eine Cola-Dose als Hülle zurück greifen, sondern möchten auch hier unser eigenes Design erschaffen.

2.1.1 Elektronische Grundlagen

Die Berechnung von Strom im geschlossenen System wird für unser Projekt benötigt, allerdings wegen der häufigen Anwendung von Digitaltechnik eher weniger. Außerdem müssen wir die Berechnung von Wiederständen in Parallel- und Reihenschaltungen kennen und selber anwenden können.

$$U * R = I$$

Reihenschaltung

$$R_{ges} = R_1 + R_2 + ... + R_n = \sum_{i=1}^{n} R_i$$

$$U_{ges} = U_1 + U_2 + ... + U_n$$

$$I_{ges} = I_1 = I_2 = ... + I_n$$

Parrallelschaltung

$$\frac{1}{R_{ges}} = \frac{1}{R_1} + \frac{1}{R_2} + ... + \frac{1}{R_n} = \sum_{i=1}^{n} \frac{1}{R_i}$$

$$U_{ges} = U_1 = U_2 = ... = U_n$$

$$I_{ges} = I_1 + I_2 + ...I_n = \sum_{i=1}^{n} I_i$$

Reihen und Parallelschaltungen für Stromquellen haben wir nicht in Erwägung gezogen und gehört deshalb auch nicht zu den fachlichen Grundlagen. Weiterhin gehörte zu den elektrischen Grundlagen, dass Verständnis über einfache Dioden und über mehrere digitaltechnische Technologien welches in den nächsten Abschnitten erklärt wird.

2.1.2 Embedded System

Ein Embedded System ist in unserem Fall der BeagleBone mithilfe vom ARM Cortex-A6 mit 1Ghz, ist ein leich modifizierte linux kernel mit Frontend installiert, dass liebevoll Angstrom genannt wurde. Um ein paar andere Beispiele für ein eingebettetes System sind etwa ein Smart TV oder ein Router, beide haben eine Art eingebettetes System, dass immer öfter auf dem Linux Kernel basiert und je nach Anwendung angepasst wurde. In unserem Fall unterstützt das BeagleBone verschiedene Technologien zum empfangen von Daten verschiedener Bauteile, wie etwa UART, I-2-C, SPI, Analog, Digital, PWM, Timer und PRU. Viele dieser Technologien sind in unserem Projekt nicht in Verwendung alle anderen Grundlagen sind unten beschrieben.

2.1.3 Transistor-Transistor-Logik

5V werden immer als logische 1 bezeichnet, damit ist gemeint wenn der Sensor den höchsten Messwert erreicht gibt er eine Spannung von 5V. Ist dies nicht der Fall hat der Sensor eine andere Kennkurve die zum Beispiel bei 3.3V aufhört. Allgemein wird aber Transistor-Transistor-Logik genutzt, welche 5V als logische 1 und geerdet als logische 0 ansieht, es gibt natürlich Toleranzen, diese sind aber bei verschiedenen integrierte Schaltkreisen und Mikrokontrollern unterschiedlich.

2.1.4 Analog-to-Digital-Converter

Andere Sensoren wie der UV-Sensor die nur über einen internen Wiederstand verfügen der sich, je nach Konzentration, an einer mathematischen Kurve orientierend, im Wert leicht verändert und dadurch die ankommende Spannung am jeweiligen Analog Pin ändert. Mithilfe eines Analog-to-Digital-Converter konvertieren wir das analoge Signal, zum Beispiel 5V, in das äquivalente digitale Signal mit der Auflösung von 12 Bits.

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

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

Das bedeutet jeder 0.001220703125V kann dargestellt werden, wobei der Arduino Mega 2560 nur 10 Bits zur Verfügung stellt.

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

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

Der Arduino kann den Wert der am analogen Pin ankommt viel ungenauer darstellen, als das BeagleBone.

2.1.5 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. Transciever und Reciever. Nun wird zwischen vielen verschiedenen Arten von UART unterschieden in unserem Fall die TTL-UART Variante welche die beim Analog-to-Digital-Converter genannten 5V als logische 1 bezeichnen.

2.2 Hülle und Platzmanagement - Fachliche Grundlage

Um die 3D gedruckte Wand zu erzeugen wurde die 3D Moddelierungssoftware Sketchup von Google verwendet. Sketchup bietet die Möglichkeit vergleichsweiße 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 Kentniss über gewisse mathematische Methoden zur Berechnung von Kreisen, Flächen und Körpern. Die meisten 3D-Drucker benötigen Datein des Types .stl, welche in Sketchup mit einem Plugin erzeugt werden können. Zum fertigen von GFK Komponenten wird ein Körper benötigt, auf welchen das GFK laminiert werden kann.

2.2.1 Inter-Integrated-Circuit

I-2-C ist ein serieller Datenbus der über zwei Kabel mit einer 10-Bit-Adressierung, 1024 IC's steuern kann mit einer maximalen Geschwindigkeit 5 Mbit/s. Der Sinn des Bussystems ist es mithilfe von einer Adressen einen Datensatz oder Befehl nur an den gewünschten Empfänger zu senden, obwohl nur eine Datenleitung genutzt wird, eine Art Master/Slave System. Der Master sagt wer wann zu sprechen hat und welche Befehle von wem zu empfangen sind.

2.2.2 Python

Als Programmiersprache zur Programmierung des Beaglebone Black's, haben wir uns für die Programmiersprache Python entschieden. Es wäre zwar 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 embedded-System handelt, welches praktisch alle Programmiersprachen unterstützt, sofern entsprechende Librarys existieren. Allerdings haben wir uns aufgrund der Tatsache, dass Python im Gegensatz zu Java nicht objektorientiert geschrieben werden muss, und wir auf der Hardwareseite möglichst auf objektorientierte Programmierung verzichten wollen, da sie nicht nötig ist, für Python entschieden. Ein weiteres wichtiges Argument war die gute Python-Library, welche von einer großen Community permanent gewartet und aktualisiert wird.

2.3 Hülle und Platzmanagement - Dokumentation

Wir haben uns dazu entschieden, die äußere Hülle aus GFK (Glasfaser verstärkter Kunststoff) zu fertigen. Dieses hat die Eigenschaften, dass er bei einem sehr geringen Gewicht, und bei einer geringen Wandstärke trotzdem eine gewisse Stabilität aufweißt. 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 Aluminium Deckel fräsen lassen. Diese haben uns ebenfalls durch ihr geringes Gewicht überzeugt.
Um die Elektronik innerhalb der Hülle zu plazieren und zu befestigen haben wir uns dazu entschieden eine Zwischenwand mit einem 3D-Drucker anzufertigen. Diese Wand teilt die Hülle mittig und bietet so auf beiden Seiten Platz um unser Microcontroller Board und unsere Sensorik Platine 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.
Am unteren Ende der Wand befindet sich eine Aushölung, sowie ein Fuß. Diese ist zum einen dafür da um den Sharp Feinstaub Sensor 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. Da der Feinstaubsensor einen Luftzug benötigt befindet sich ein Durchlass innerhalb der Wand. Um das Microcontroller Board mit der Sensorik Platine zu verbinden existiert ein Fenster in der Mitte der Wand. Um die Sensorik Platine und das Microcontroller Board an der Wand zu befestigen existieren vier Bohrungen.

Abbildung 1: Der Sattelit

3 Bodenstation

3.1 Einleitung

In diesem Teil der Dokumentation werden wir die Bodenstation vorstellen, welche als Datenempfänger und als Datenverarbeitungsplattform fungiert.
Die Bodenstation wurde von Robin Bley, Marc Huisinga und Kevin Neumeyer entwickelt.

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.
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.

3.2 Verwendete Komponenten

Zum Erreichen unserer Ziele haben wir verschiedene Komponenten verwendet, welche einerseits der Datenvisualisierung und -analyse dienen, andererseits aber auch der Entkopplung und skalierbaren Entwicklung dienen.

Für die Bodenstation haben wir folgende Komponenten verwendet:

Java 8 als Programmiersprache, da jedes unserer Gruppenmitglieder mit Java vertraut ist, wir aber trotzdem die mächtigen funktionalen Features von Java 8 nutzen wollten

Netbeans Platform das die Möglichkeit bietet, einfach eine integrierte, modulare und entkoppelte GUI-Applikation auf Basis von Java Swing zu entwickeln

JUnit zum Testen von bestimmten, komplizierten Teilen der Applikation

org.json als JSON-Library zum Parsen von erhaltenen JSON-Daten, wie beispielsweise beim Empfangen von Daten, und zum Generieren eigener JSON-Daten, wie beispielsweise beim Sichern von Daten

JSerialComm (zum Start des Projektes noch serial-comm) zum Lesen von Daten aus seriellen Ports

NASA World Wind zum Anzeigen der Satellitenflugbahn auf einer dreidimensionalen-, Satellitenbilder-basierten Weltkugel in Echtzeit

JChart2D zum Anzeigen von übertragenen Daten in einem zweidimensionalen Graphen in Echtzeit

4 Die Android Applikation

4.1 Einleitung

Stellen sie sich vor hier würde Lorem Ipsum stehen