Klimadatenerfassung mittels der Mikrocontrollerplattform Arduino

GEORG-WILHELM-STELLER-GYMNASIUM

SEMINARARBEIT

vorgelegt von

Simon Stadlinger

8. November 2016

jugend forscht

Korrektor: StD Dr. Stephan Bärthlein

Bewertung der Arbeit: ___ Punkte

Bewertung der Präsentation: ___ Punkte

Inhaltsverzeichnis

1 Projekterklärung

1.1 Historischer Hintergrund

„Michel Joseph de Montgolfier (1740-1810) und sein Bruder Étienne Jacques (1745-1799) ließen im französischen Annonay bei Lyon am 5. Juni 1783 über dem Marktplatz einen unbemannten Heißluftballon 1000 Meter hoch aufsteigen. Der Ballon flog in zehn Minuten 2,5 Kilometer weit.“ [1].
Gegen unser heutiges Wissen vermuteten die Brüder damals noch, das Rauch von verbrennendem Material für den Auftrieb eines Ballons verantwortlich war. Der erste bemannte Freiflug fand am 21. November 1783 statt. Ebenfalls von den Montgolfiers konstruiert blieb der Ballon mit zwei französischen Adeligen für 25 Minuten in der Luft. Der französische Physiker Jacques Charles hatte das Auftriebsprinzip verstanden und flog mit dem ersten Gasballon gefüllt mit Wasserstoff zusammen mit seinem Konstruktionshelfer noch im selben Jahr am 1. Dezember 36 Kilomenter. Er erreichte im anschließenden Soloflug eine Höhe von 3467 Metern [2]. Zur Forschung mit dem Ballon kam es erst in den 1890er Jahren. Erste Messonden wurden gebaut, welche mit Fühlern ausgestattet in für bemannte Fahrten unerreichbare Höhen aufstiegen. Victor Franz Hess wird als Pionier auf dem Gebiet der Forschung mittels Ballon angesehen. Ihm gelang es mithilfe von sieben Versuchsflügen ausgestattet mit entsprechenden Amaturen im Jahr 1912 β- und γ-Strahlung in einer Höhe von bis zu 5200 Metern nachzuweisen [3, S. 1084, S. 1089].

1.2 Projektanreiz

Auch heute noch findet Forschung mithilfe von Radiosonden, die an einem Ballon befestigt sind, statt. Der Deutsche Wetterdienst lässt täglich mehrere solcher Sonden in die Atmosphäre aufsteigen und entnimmt durch live-Übertagung per Radiosignal Daten. Der Vorteil des Systems des Wetterballons liegt darin, dass so ein Vertikalschnitt der aufgezeichneten Daten durch die Atmosphärenschichten ersichtlich wird, wohingegen mit Satelliten nur eine Sicht aus der Vogelperspektive möglich ist [4]. Ein Nachteil des praktizierten Verfahrens ist der hohe Kostenfaktor der Radiosonden, die nach ihrer Rückkehr auf den Boden keine weitere Verwendung finden. Auch werden nur Luftfeuchte, Luftdruck und Windgeschwindigkeiten übertragen. Ein grundlegender Anspruch des Projekts „Klimaforschung in der Atmosphäre mittels Wetterballon“ stellt somit die möglichst kostengünstige Konzeption und Umsetzung einer für diesen Zweck geeigneten Sonde dar. Zudem werden wesentlich mehr physikalische und auch chemische Messgrößen erfasst. Das Projekt wurde zusammen mit meinem Projektpartner Niclas Popp, dessen Arbeit „Praktische Konzeption einer Klimasonde“ [5] als Komplementär zu dieser gelesen werden muss, im September 2015 begonnen und der erste Freiflug fand am 22. August 2016 statt. Die beiden Arbeiten stellen einen Zwischenbericht des Projekts dar und sind als Auszug der einzelnen Teilarbeitsbereiche zu verstehen.

2 Ausarbeitung des Teilarbeitsbereichs Messwerterfassung

In diesem Kapitel werden das Prinzip der Datenerfassung erklärt und die einzelnen Komponenten der Sonde, die für die jeweiligen Ebenen dieses Grundprinzipes zuständig sind, genauer charakterisiert. Hierbei wird im Speziellen auf die periphere und zentrale Hardware, die Kommunikationsprotokolle und die Softwarestruktur eingegangen. Anschließend wird die Auswertung der erfassten Werte unter genauerer Betrachtung des Lufttemperaturgradienten aufgeführt und der theoretische Ansatz zur Ermittelung der Magnetfeldinklination mithilfe des Messdatensatzes erklärt. Abschließend werden Verbesserungen und Erweiterungen des bestehenden Systems besprochen.

2.1 Grundlegende Definition

Die Bedeutung des Wortes „Messdatenerfassung“ lässt sich durch die Worte „Messung“, „Verarbeitung“ und „Speicherung“, die in dieser Reihenfolge zeitlich stattfinden, in einem ersten Schritt der Verfeinerung ausdrücken. Die Spezialisierung auf die Klimasonde ergibt ein Schema, das in Abbildung 1 gezeigt ist. Sensoren messen physikalische und chemische Messgrößen. Diese aufgenommenen Daten werden in einem Messzyklus durch die verarbeitende Einheit, einem Mikrocontroller, abgefragt, verarbeitet und weiter zur Sicherung auf ein Speichermedium übertragen. Schnittstellen zwischen den drei Ebenen bilden Kommunikationsprotokolle aus der Elektrotechnik. Mit I^2C werden vier von sechs Sensoren angesteuert und SPI dient als Datentransferprotokoll auf die SD-Karte. Beide Kommunikationsprotokolle werden in Abschnitt 2.2.2 genauer betrachtet. Die Softwarekomponente des Systems befindet sich auf dem Mikrocontroller und wird von diesem ausgeführt.

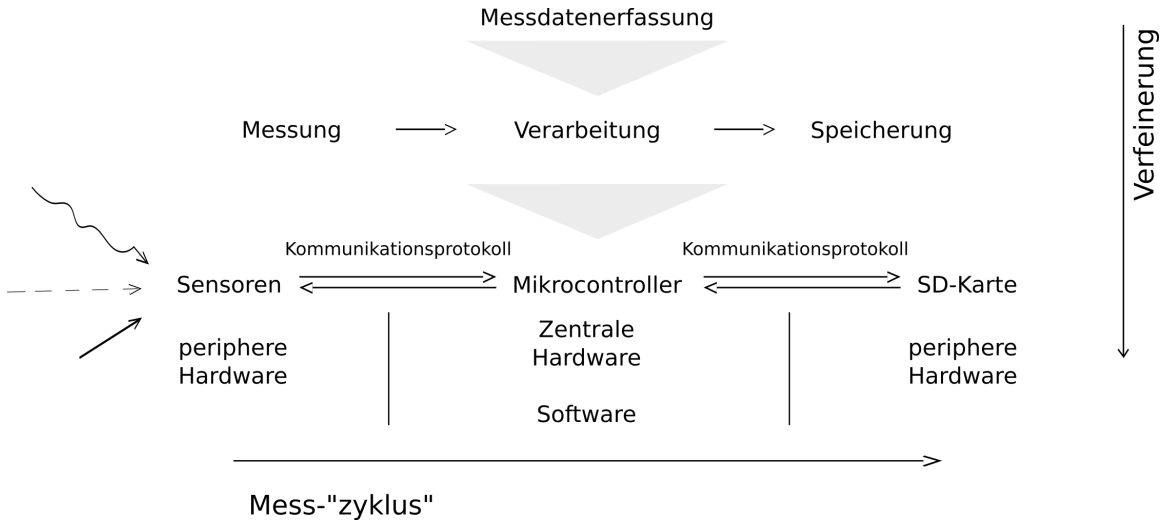


Abbildung 1: Schematische Darstellung des in 2.1 beschriebenen Messzyklus. Das System ist ein Zyklisches zu nennen, da die Software die Hardware den Messvorgang immer wieder durchlaufen lässt. Die vertikale Ebenenaufteilung erfolgt in zwei periphere Hardwareebenen und eine zentrale Hardwareebene sowie Software auf dem Mikrocontroller.

2.2 Spezielle Realisierung

2.2.1 Hardware

Zentrale Hardware

Die Anforderungen an die zentrale Recheneinheit sind vielfältig. Primär muss sie digitale und analoge Signale von entsprechenden Sensoren verarbeiten und diese in externen oder internen Speichermedien ablegen können. Da Maße und Gewicht in der Raumfahrt die häufigsten limitierenden Faktoren sind, soll beides entsprechend klein ausfallen. Außerdem muss die Benutzeroberfläche und somit die Bedienung für Amateure im Bereich der Informatik einfach zu erlernen sein. Die Mikrocontroller Plattform Arduino wurde aus diversen Gründen der bekannteren Raspberry Pi vorgezogen. So stellt der Raspberry Pi einen vollwertigen Computer mit dezidiertem Prozessor und Grafikeinheit dar. Ebenso arbeitet eine speziell modellierte Version des Betriebssystems Linux als Benutzeroberfläche und er ist dementsprechend vor allem für Projekte geeignet, in denen graphische Benutzeroberflächen und die Interaktion mit diesen im Vordergrund stehen. Dagegen ist die Fähigkeit des Arduinos, Code direkt von einer integrierten Benutzerumgebung, der Arduino IDE, umzusetzen, leichter zu beherrschen. Der Arduino ist gemäß seiner Natur als Mikrocontroller dafür konzipiert, einfache Sensoren und Boards (Shields) zur Erweiterung der Basisfähigkeiten anzusteuern. Er basiert auf Atmel Mikrocontrollern und ist als Breakout-Board konstruiert [6, S. 6] (Abbildung 2). Diese Boards ermöglichen es, die sehr kleinen Pin-Ausgänge des Mikrocontrollerchips über größere, sogenannte „Female“-Pins mit Jumper-Kabeln zu erreichen. Somit ist man in der Lage, Peripheriefunktionen der Mikrocontrollereinheit (MCU) zu nutzen. Zu diesen gehören Analog-Digital-Wandler (ADC), Input/Output-Pins (I/0), Kommunikationsprotokolle wie I^2C und SPI und Serielle Schnittstellen (USB) zur einfachen Programmierung [6, S. 6].
Schon in der Testphase des Prototypen wurde das Basismodel Arduino UNO an die Grenzen seiner RAM-Kapazität gebracht. Um Stabilitätsprobleme bei der Messung zu vermeiden, wird aktuell der größere Arduino MEGA 2560 eingesetzt. Dieser verfügt mit 8 kB RAM über das Vierfache des Arduino UNO. Die MCU ATmega2560 von Atmel wird mit 16 MHz getaktet und arbeitet mit einer unter Mikrocontrollern standardisierten Betriebsspannung von 5 V [7].

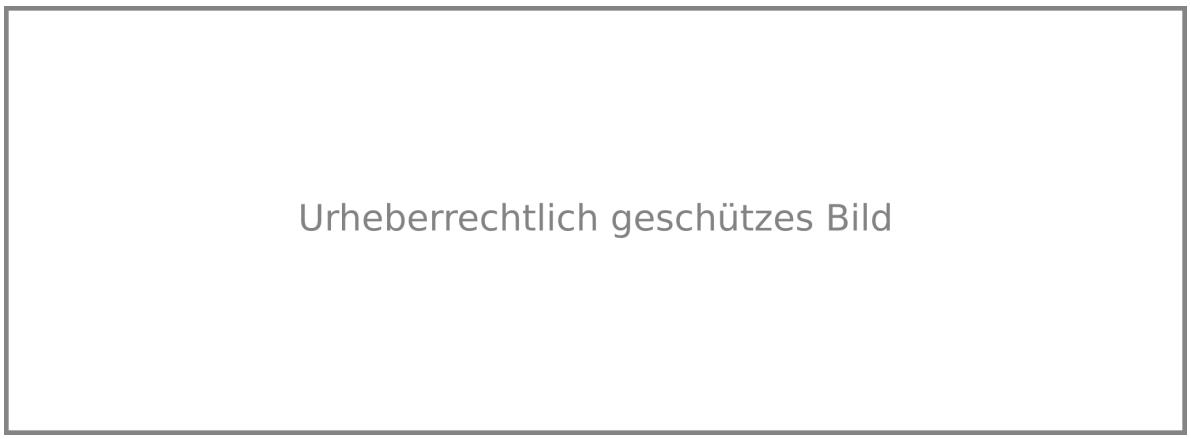


Abbildung 2: [8] Rechts zu sehen ist der Arduino UNO mit den Maßen 68.6 x 53.4 mm und einem Gewicht von 25 g [9]. Der Arduino MEGA 2560 hat ein Gewicht von 37 g und misst 101.5 x 53.3 mm [7].

Periphere Hardware

Die periphere Hardware übernimmt die Aufgaben der beiden Randebenen des in Abbildung 1 gezeigten Schemas. Die Sensoren auf der einen Seite registrieren chemische und physikalische Größen und leiten sie an den zentralen Arduino MEGA 2560 weiter. Nach der Kalkulation durch diesen werden die Daten auf einem Speichermedium abgelegt, welches das Ende der Verarbeitung darstellt.
Beim ersten Stratosphärenflug am 22. August 2016 wurden folgende klimarelevanten Daten gemessen, wobei derjenige Sensor, mit dem diese gemessen wurden dahinter aufgeführt ist: Luftfeuchtigkeit, -temperatur, -druck (BME280) [10], Lichstärke, Infrarotstrahlung, Ultraviolettindex (SI1145) [11], Achsenbeschleunigung auf allen drei Raumdimensionen, Winkelbeschleunigung auf drei Achsen (MPU6050) [12], Teilbetrag des Erdmagnetfeldvektors auf allen drei Achsen (HMC5883L) [13], $C0_2$ (MQ135) und O_3 (MQ131). Durch spezielle Breakout-Boards werden die Funktionen der Sensoren optimal und für eine standardisierte Pin-Breite von 2,54 mm zugänglich gemacht. Die letzten beiden Sensoren MQ131 und MQ135 müssen ob der Art ihrer Messwertübermittlung von den ersten Vier abgegrenzt werden. Sie arbeiten analog und werden jeweils an einem der 16 analogen Input-Pins an den Arduino Mega angeschlossen [7]. Ein Auswahlkriterium für die vier erstgenannten Sensoren war die digitale Arbeitsweise und die damit verbundene Unterstützung des Kommunikationsprotokolls I^2C. Zudem können alle Sensoren mit einer geringen Stromstärke von 3,8 μA (BME280) [10, S. 2] bis 3,46 mA (MPU6050) [12, S. 10] betrieben werden, wodurch eine Gewichtszunahme aus Gründen der Akkukapazität vermieden werden sollte.
Die SD-Karte wird über ein dafür vorgesehenes Board an den Arduino Mega angeschlossen. Dieses muss SPI in Hardware unterstützen. Bei der Auswahl der SD-Karte muss beachtet werden, dass der Arduino nur das Dateisystem FAT16 unterstützt und somit nur SD-Karten mit einer Maximalspeicherkapazität

von 4 Gigabyte ansprechen und beschreiben kann.

Abbildung 3: Zu sehen sind der O_3-Sensor [14] (links), die beiden I^2 Sensoren HMC5883L [15] und MPU6050 [16] (mittig) und ein dem unseren ähnelndes SD-Karten Shield [17] (rechts).

2.2.2 Kommunikationsprotokolle

Um den korrekten Datentransfer zwischen den einzelnen Komponenten eines Systems der Elektrotechnik zu garantieren, müssen Abläufe, charakterisiert durch Spannungs-Zeit-Verläufe gesagt Stromstöße, die diesen Austausch darstellen, genauestens definiert werden. Dies geschieht mithilfe von Kommunikationsprotokollen. „In seiner einfachsten Form kann ein Protokoll definiert werden als eine Menge von Regeln, die Syntax, Semantik und Synchronisation der Kommunikation bestimmen.“ [18]. Diese Regeln und syntaktischen und semantischen Angaben müssen allen Kommunikationspartnern bekannt sein. Auch muss ein Protokoll der Elektrotechnik sowohl in Hardware als auch Software implementiert werden. Sollte das Regelwerk einem der Teilnehmer eines Datenaustausches nicht oder fehlerhaft vorliegen, kommt es unvermeidlich zu Komplikationen. Bits werden falsch interpretiert und Messergebnisse nicht korrekt abgespeichert. Zudem ist im speziellen Fall der Amateurraumfahrt die Hardwarekomponente stärker zu beachten. Die Anzahl der Verbindungen zwischen den Bestandteilen des Messapparates ist, da sie meist nur aus einfachen Jumperkabeln besteht, ausschlaggebend für die Fehleranfälligkeit der Kommunikation.
So wird bei der Konstruktion der Wetterballonsonde auf zwei einfache und weit verbreitete Kommunikationsprotokolle zurückgegriffen, deren jeweilige Stärke optimal genutzt wird. I^2C wird für das Auslesen der digitalen Sensoren verwendet und über SPI die SD-Karte beschrieben.

I^2C

Das Protokoll I^2C wurde von Phillips in den 1980er Jahren eingeführt und steht für „Inter Integrated Circuit“. In den 1990ern standardisiert, wurde es von vielen Firmen adaptiert [6, S. 164]. Es benötigt nur zwei Verbindungen um eine Kommunikation stattfinden zu lassen und ist so auch unter dem Namen TWI („two-wire-interface“) zu finden. Viele nicht I^2C genannte Protokolle, welche ebenfalls zwei Leitungen verwenden basieren auf den im Folgenden beschriebenen Aufbau und Kommunikationsablauf.

Die Teilnehmer am Bussystem werden als „Master“ und „Slave“ bezeichnet. Beide können sowohl Informationen versenden und empfangen. Der Master gibt die Taktfrequenz der Kommunikation vor. Die Hardware muss neben Versorgungsspannung und Erdung zwei separate Leitungen bereitstellen. Diese werden SCL („Serial-Clock-Line“) und SDA („Serial-Data-Line“) abgekürzt. Die Datenleitung (SDA) wird bidirektional verwendet und verschiebt bei jedem Puls der Taktleitung (SCL) einen Bit zwischen den Busteilnehmern. Auf beiden Leitungen werden Pull-Up Widerstände eingebunden. Diese Widerstände regeln bei variierender Sensorzahl, die den Bus nutzt, die Signalstabilität. Der Arduino Mega bietet ein vordefiniertes Hardware-I^2C an, bei dem vor den zwei benötigten Pins auf dem Breakout-Board passende Pull-Up Widerstände eingebaut sind [7].
Die Besonderheit des Hardwareaufbaus besteht darin, dass alle Sensoren parallel auf die Leitungen geklingt werden. So entsteht ein Schaltplan nach [6, S. 165] (siehe Anhang S. 24). Das Protokoll ermöglicht es mehrere Slaves mit einem Master über nur zwei Leitungen anzusteuern. Die Kommunikation kann nur durch einen Master eingeleitet werden, wohingegen Slaves nur reagieren können. Andernfalls würden Informationen durch gleichzeitig stattfindende Schreib- und Lesevorgänge der Teilnehmer verfälscht werden. Alle Slaves besitzen verschiedene 7-Bit-Adressen, damit sie vom Master gezielt aufgerufen werden können. Nur so kann garantiert werden, dass Informationen am richtigen Abnehmer ankommen [6, S. 166].
Der grundlegende Kommunikationsablauf setzt sich wie folgt zusammen, variiert aber von Bauteil zu Bauteil in speziellen Sequenzen: Der Master sendet einen Start Bit. Es folgt die 7-Bit-Adressse. Da nach dem Start Bit alle Slaves vorgewarnt sind, greift nach der 7-Bit-Adresse nur noch der adressierte Slave Daten von SDA ab. Es folgt ein Bit vom Master, das festlegt ob von den Speicherregistern des Slaves gelesen (1), oder darauf geschrieben werden soll (0). Nach einem „acknowledge“ (ACK)(= logische 0) vom Slave beginnt der eigentliche Datenaustausch. Dadurch, dass von den Sensoren Messdaten gespeichert werden sollen wird der Read-Mode angewendet. Hierbei wird ein Byte an Informationen vom Slave zum Master transferiert und von diesem ein ACK an den Slave gesendet. Abhängig von der Menge an Daten folgen nun so viele Bytes und dazwischen liegende ACK’s, wie nötig sind, um die Daten vollständig zu erhalten. Ein Stop Bit vom Master beendet die Datenübertragung [6, S. 167-168].
Der Grund für die ausführliche, bitweise Erklärung des Ablaufes einer einfachen Datenübertragung ist es zu zeigen, worin die große Schwäche von I^2C liegt. Dadurch, dass nur eine einzige serielle Datenleitung vorhanden ist, kann nur ein Sensor zeitgleich abgefragt werden und dieser muss sich die bidirektionale Verbindung mit dem Arduino teilen. Sollen alle Sensoren der Reihe nach abgefragt werden, schlägt sich das auf die Dauer eines Messzyklus nieder. So sind mit I^2C Datenübertagungsraten von maximal 100 KBits/s bei einem Sensor möglich [19, S. 271]. Der große Vorteil diese Protokolls besteht jedoch darin, dass alle Sensoren mit den selben vier Leitungen angesteuert werden können. So wurden BME280, SI1145, HMC5883L und MPU6050 modular auf eine speziell geätzte Leiterplatine gesteckt (Abbildung 4).

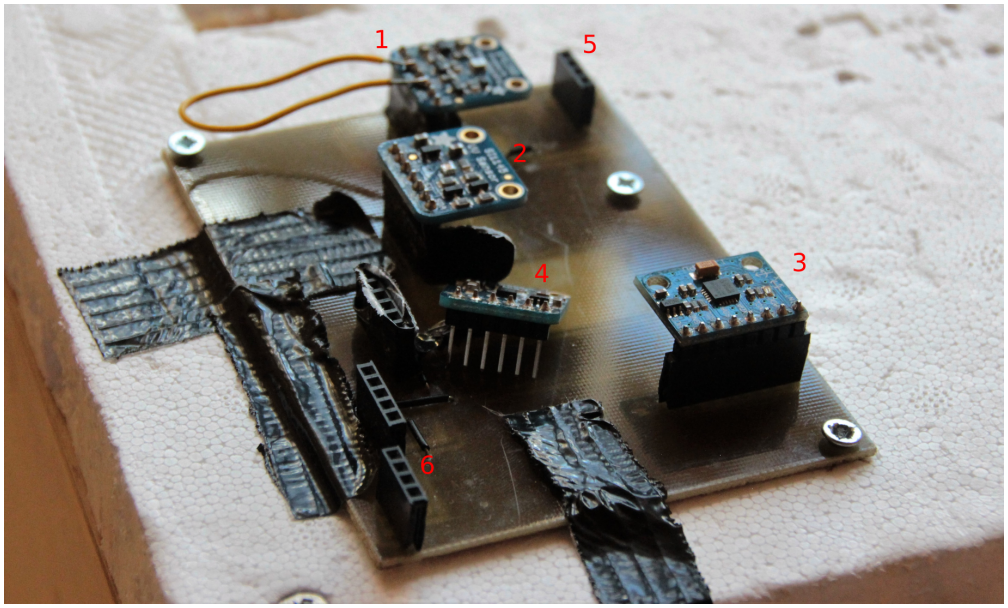


Abbildung 4: Selbstgeätzte Leiterplatine [5, S. 10] mit den Sensoren BME280 (1), SI1145 (2), MPU6050 (3) und HMC5883L (4). Eine Viererreihe Female-Pins (5) dient dazu, die vier für I^2C benötigten Leitungen vom Arduino kommend an die Platte anzuschließen. Durch die letzte Pinreihe (6) wurde die Möglichkeit einer Erweiterung offen gehalten.

SPI

Der SPI-Bus ist für ein ähnliches Einsatzgebiet wie I^2C konzipiert. Es wurde ursprünglich von Motorola eingeführt. Da ein formaler Standard nie festgelegt wurde, findet man viele SPI-Module, die eine leicht abgeänderte Form in Hardware oder Software haben [6, S. 182]. In diesem Paragraph wird die Form behandelt, die auch bei der SD-Karte verwendet wurde.

Der große Unterschied zu I^2C liegt in der Hardwarekonstellation. Hier werden für die Kommunikation vier Leitungen benötigt: MISO (**M**aster **I**n **S**lave **O**ut) wird für die serielle Datenübertragung von Slave zu Master genutzt. Sein Pendant MOSI (**M**aster **O**ut **S**lave **I**n)wird für die entgegengesetzte Richtung gebraucht. Ein gemeinsamer Taktgeber SCK (Serial Clock/Shared Clock) synchronisiert die beiden unidirektionalen Datenleitungen, welche dann wie bei I^2C bei jedem Puls von SCK Bits verschieben. Die Auswahl des Slaves findet hierbei nicht mit 7-Bit-Adressen statt, sondern ist durch Hardware gelöst. So benötigt jeder Slave eine extra, CS (Chip Select) genannte, Leitung zum Master. Es ergibt sich ein beispielhaftes Hardwareschema nach [6, S. 185] (siehe Anhang S. 24).

Ein typischer Kommunikationsablauf lässt sich wie folgt beschreiben: CS wird zu Beginn der Kommunikation mit einem Slave vom Masterdevice auf logisch Null gesetzt. Es werden alle Slaves die Zugriff auf MOSI, MISO und SCK haben angesteuert, sie ignorieren die Informationen jedoch, sollte ihr CS-Pin nicht auf „Low“ gesetzt sein. Der eigentliche Datenaustausch kann direkt danach über MISO und MOSI ablaufen. Pro Puls der SCK-Leitung wird ein Bit verschoben. Sobald die Daten komplett übertragen

wurden, wird CS wieder auf „High“ gesetzt und die Übertragung ist beendet.
Aus der Kürze der Beschreibung wird der Vorteil von SPI gegenüber I^2C deutlich. Mit diesem Kommunikationsprotokoll sind schnellere Datenübertragungsraten von bis zu 1 MBit/s möglich [6, S. 185][19, S. 272]. Damit eignet sich dieses Protokoll gut, um die gebündelten Daten der Sensoren schnell auf die SD-Karte zu verschieben und dort zu speichern. Der damit einhergehende Nachteil ist die im Vergleich hohe Zahl an Verbindungen, die nötig sind um einen Slave anzusteuern. So wird der Hardwareaufbau bei mehr als drei Busteilnehmern schnell unübersichtlich und fehleranfällig.

2.2.3 Software

Die Aufgabe der Software ist es, die Hardwarekomponenten aus 2.2.1 mithilfe der Kommunikationsprotokolle aus 2.2.2 auf korrekte Weise zusammenarbeiten zu lassen. Das Programm ist dazu in der Laage, Daten von vier digitalen und zwei analogen Sensoren entgegenzunehmen und auf eine SD-Karte zu transferieren.
Der Arduino wird mithilfe der hauseigenen „Arduino Integrated Developement Environment “ (Arduino IDE) programmiert [20]. Diese bietet die Möglichkeit ein geschriebenes Programm direkt auf Syntaxfehler zu untersuchen, zu kompilieren und auf ein Board zu laden. Hierfür wird ein Arduino per USB an den PC angeschlossen und in den Einstellungen das entsprechende Modell, welches den Code erhalten soll, ausgewählt. Das Programm startet, sobald der Arduino mit Energie versorgt wird, der Reset-Knopf auf dem Arduino gedrückt wird oder es vollständig auf den Arduino geladen wurde. Die Programmiersprache ist C bzw. C++ [21].

Struktur der Erfassungsroutine

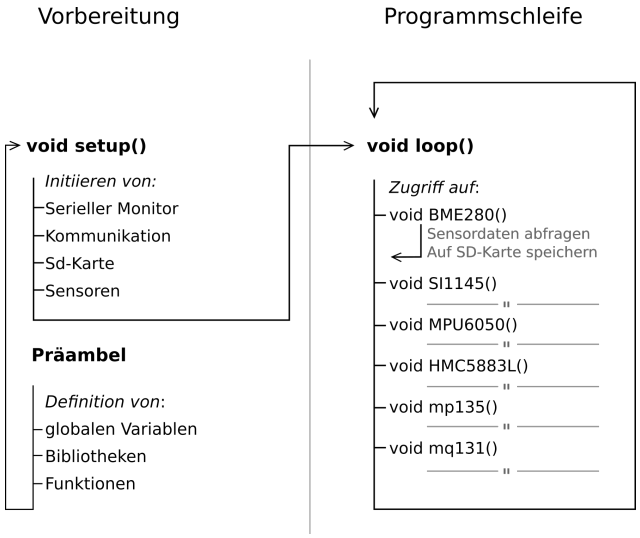


Abbildung 5: Grundstrukur der Erfassungsroutine.

Jedes Programm muss, um auf einem Arduino zu funktionieren, über zwei Hauptmethoden -in C auch Funktionen genannt- verfügen. Ähnlich der „main()“ in C werden diese automatisch beim Start des Programms ausgeführt. Der Unterschied zwischen Beiden liegt in der Art der Ausführung. So wird zunächst „void setup()“ einmalig ausgeführt und dementsprechend für das einmalige Initialisieren von Pin-Richtungen oder Sensorkommunikationen genutzt [6, S. 16]. Sie wird als Vorbereitung der anschließenden „void loop()“ verstanden. Diese Funktion wird in einer endlichen Schleife vom Arduino ausgeführt, bis die Stromzufuhr unterbrochen wird. Hier findet die eigentliche Arbeit statt. Das verfeinerte Grundprinzip der Software der Klimasonde ist in Abbildung 5 schematisch dargestellt. Der zugehörige Code ist immer am Ende des Paragraphen abgebildet.
Grundsätzlich kann der Sketch in drei Blöcke unterteilt werden. Zwei davon stellen „void setup()“ und „void loop()“ dar. Der Dritte befindet sich noch vor der ersten Funktion und dient als Präambel des gesamten Programms. Hier werden globale Variablen deklariert und Softwarebibliotheken geladen. Genutzt werden Bibliotheken für das SPI und I^2C Protokoll und zur Ansteuerung der Sensoren BME280, HMC5883L und SI1145 sowie der SD-Karte. Softwarebibliotheken sind vorgefertigte Codesammlungen, werden bei vielen I^2C fähigen Sensoren vom Hersteller selbst geschrieben und erleichtern dadurch, dass die in ihnen enthaltenen Subroutinen und -methoden vom Hauptprogramm des Arduino modular verwendet werden können, das Ansteuern der Sensoren und der SD-Karte [22]. Zudem müssen alle Köpfe von Funktionen, welche einen Übergabeparameter besitzen, vor „void loop()“ einmal genannt werden.
In oben genanntem Schema wird die initialisierende Aufgabe von „void setup()“ deutlich. Um die Sensoren überprüfen zu können wird ganz zu Beginn I^2C initialisiert. Hiernach wird mit dem seriellen Monitor ein weiteres wichtiges Element der Softwareentwicklung mit dem Arduino gestartet. Über den Monitor können Informationen, die vom Arduino geschickt werden, in einem extra Fenster am PC Bildschirm angezeigt werden. Somit ist die Nutzung des seriellen Monitors nur stationär am Boden möglich und er wird zur Fehleranalyse und Kontrolle des Beta-Sketches verwendet. Sobald im späteren Programmverlauf -„void setup()“ wie „void loop()“- ein Codeabschnitt durchlaufen oder nicht ordnungsgemäß durchlaufen wurde, wird eine entsprechende Meldung ausgegeben. Ebenso wird jeder Wert der Sensoren in der Testphase am Boden angezeigt, um Anomalien feststellen zu können (siehe 3.1). Anschließend wird jeder Sensor einzeln auf seine Einsatzbereitschaft getestet, wobei das Programm einen Fehler anzeigt, sobald ein Sensor über den Bus nicht antwortet. Ein weiterer Aspekt in „void setup()“ ist die Initialisierung der SD-Karte und das erstmalige Generieren der „.dat“-Datei. Hierbei ist zu beachten, dass im selben Zug die ersten zwei Zeilen geschrieben werden. Die Erste stellt eine Zahlenabfolge von 1 bis Anzahl der gemessenen Werte$+1$ dar. In der Zweiten stehen Abkürzungen für die gemessenen Größen unterhalb der Nummern der ersten Zeile. Die erste Spalte enthält den Zeitpunkt seit dem Programmstart in 10tel Sekunden, zu dem die nachfolgenden Daten aufgenommen wurden. Somit sind die Spaltenüberschriften der einzelnen Messwerte einmalig und durchnummeriert gegeben.
Zu Beginn des „Protyping[s]“ [5] mussten nur drei Sensoren angesteuert werden. Der gesamte Code zum Auslesen dieser und zum Beschreiben der SD-Karte befand sich in „void loop()“. Da die Sonde immer weiter ausgebaut wurde, wurden die einzelnen Teile des Codes zugehörig zu den entsprechen-

den Sensoren ausgelagert. So ist eine flexiblere Fehlerbehebung und verbesserte Übersicht gegeben. Die Auslagerung geschieht durch Subfunktionen, welche den Sensoren entsprechend benannt werden. Diese Subfunktionen werden von „void loop()“ der Reihe nach aufgerufen und führen in ihren Methodenrümpfen in jeweils abgewandelter Form immer zwei identische Aktionen durch: Die momentanen Sensormesswerte werden in lokalen Variablen gespeichert und in die „.dat“-Datei auf die SD-Karte abgelegt. Damit die einzelnen Sensorfunktionen in die Datei schreiben können, wird diese vor allen Subfunktionen geöffnet und abschließend wieder geschlossen. Um zu garantieren, dass jeder Messwert unter der passenden Spaltenüberschrift gespeichert wird, ist die konkrete Reihenfolge, in der die Subroutinen ausgeführt werden, an das einmalige Generieren der Spaltenüberschriften in „void setup()“ und der darin festgelegten Reihenfolge angepasst. In denjenigen Unterfunktionen, deren zugehörige Sensoren mehr als einen Messwert ausgeben können, muss ebenfalls darauf geachtet werden, dass die korrekte Reihenfolge eingehalten wird. Jede gespeicherte Zeile der Messwerte stellt die Quintessenz eines Messzyklus dar. Sie beginnt mit der vergangen Zeit seit Start des Programms in 10tel-Sekunden und endet mit einem Zeilenumbruch. Beides ist jeweils in der ersten und letzten Subroutine implementiert.

```
void setup()                                                                    1
{
  Wire.begin();
  //heizungspins
  pinMode(x, OUTPUT);
  pinMode(y, OUTPUT);                                                           6
  pinMode(z, OUTPUT);
  Serial.begin(9600);
  Serial.print("Initializing SD card...");
  pinMode(SS, OUTPUT);
                                                                                11
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
   while (1) ;
  }
  Serial.println("card initialized.");                                          16


  dataFile = SD.open("datalog.dat", FILE_WRITE);
  if (! dataFile) {
    Serial.println("error opening datalog.dat");                                21
    while (1) ;
  }
  else {
    dataFile.println("#1          2     3  4     5     6     7     8     9     10 ...);
    dataFile.println("#sec/100s   p280  t  hum   Vi    IR    UV    AcX   AcY   AcZ...)26
    dataFile.close();

  }
...
                                                                                31
void loop()
{
```

```
  dataFile = SD.open("datalog.dat", FILE_WRITE);
  delay(100);
  BME280(dataFile, tempoutdoor);                                    36
  delay(100);
  SI1145(dataFile);
  delay(100);
  MPU6000(dataFile);
  delay(100);                                                       41
  HMC5883L(dataFile);
  delay(100);
  Mq135(dataFile);
  delay(100);
  Mq131(dataFile);                                                  46
  delay(100);
  regulation(dataFile);
  dataFile.flush();
  dataFile.close();
}                                                                   51
```

Auslesen des Sensors MPU6050

In der Subroutine des MPU6050 geschieht die Kommunikation und das Werteauslesen nicht mit einer herstellereigenen Softwarestütze, sondern wird mithilfe der Wire.-Library von Arduino realisiert. Diese Softwarebibliothek ist dafür konzipiert, die abstrakten Verschiebungen des I^2C-Protokolls im Programmcode einfacher auszudrücken. Hierbei muss jedoch trotzdem - wie in 2.2.2 angesprochen - eine konkrete Abfolge eingehalten werden.

Zunächst erfolgt die Deklaration der lokalen Variablen. Um die 16-Bit Auflösung der Gleitkommazahlen des digitalen Sensors speichern zu können, wird der Datentyp „double“ verwendet. Die Kommunikation mit dem Sensor wird mithilfe von entsprechenden Wire.-Befehlen initialisiert. Dessen Adresse wurde in der Präambel des Programms in „MPUBeschl“ definiert. Nach erneutem Ansprechen des zuvor in Bereitschaft versetzen Sensors werden 14 Bytes erwartet. Diese teilen sich auf in jeweils zwei Bytes (16-Bit) für die einzelnen Werte. Sobald diese am Leitungsende des Sensors zur Verfügung stehen, werden sie mithilfe des I^2C-Protokolls transferiert und in die lokalen Variablen gespeichert. Sobald ein Byte gespeichert wurde, erfolgt eine Verschiebung um acht Bit, um das zweite Byte in der Variable speichern zu können. Die Reihenfolge der Bytes ist im Datenblatt des Sensors festgelegt. Sie gilt in der Reihenfolge der Variablen unbedingt zu übernehmen, da sonst Werte falsch betitelt und somit in der falschen Spalte gespeichert werden würden. Dieser Umstand erfordert es auch, dass die Bytes, die die Temperatur ausdrücken, abgenommen werden, um eine Verschiebung zu vermeiden. Hiernach folgt die Kalkulation der Werte. Die Parameter wurden manuell mithilfe mehrerer Messreihen und kontinuierlicher Werteausgabe am seriellen Monitor ermittelt. Der Code ab Zeile 32 zeigt den zweiten Aspekt der Subroutine. Wenn die Datei auf der SD-Karte bereit steht, beginnt das Beschreiben. Hierbei ist auf die Einhaltung der Reihenfolge gemäß der Spaltenbeschriftung in „void setup()“ und der Tabulatoren zu achten. Auch werden die Messergebnisse immer auf dem seriellen Monitor, falls dieser vorhanden ist, ausgegeben, was in der Testphase der Sonde am Boden unabdingbar ist. Sobald die Funktion durchlaufen ist, wird der Rest von

„void loop()“ ausgeführt.

```
void MPU6000(File measuredD1) {
  double AcX, AcY, AcZ, TEMP, gyX, gyY, gyZ;      //Von hier
  Wire.beginTransmission(MPUBeschl);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission();

  delay(10);
  Wire.beginTransmission(MPUBeschl);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPUBeschl, 14);
  while (Wire.available() < 14);
  AcXraw = Wire.read() << 8 | Wire.read();
  AcYraw = Wire.read() << 8 | Wire.read();
  AcZraw = Wire.read() << 8 | Wire.read();
  TEMP = Wire.read() << 8 | Wire.read();
  gyXraw = Wire.read() << 8 | Wire.read();
  gyYraw = Wire.read() << 8 | Wire.read();
  gyZraw = Wire.read() << 8 | Wire.read();      //bis hier analog zu HYT221

  TEMP = (TEMP / 340.0) + 32.53;

  AcX = AcXraw / 1600.0 * 1.097;
  AcY = AcYraw / 1600.0 * 1.097;
  AcZ = AcZraw / 1600.0 * 1.097;

  gyX = gyXraw / 100.0;
  gyY = gyYraw / 100.0;
  gyZ = gyZraw / 100.0;

  if (measuredD1) {
    measuredD1.print(AcX);
    measuredD1.print(" ");
    measuredD1.print(AcY);
    measuredD1.print(" ");
    measuredD1.print(AcZ);
    measuredD1.print(" ");
    measuredD1.print(gyX);
    measuredD1.print(" ");
    measuredD1.print(gyY);
    measuredD1.print(" ");
    measuredD1.print(gyZ);
    measuredD1.print("    ");
  }

  Serial.print(AcX); Serial.print(" "); Serial.print(AcY);
  Serial.print(" "); Serial.print(AcZ); Serial.println(" ");
  // Serial.print(gyX); Serial.print(" "); Serial.print(gyY);
  // Serial.print(" "); Serial.print(gyZ); Serial.print(" ");
}
```

Auslesen des Sensors BME280

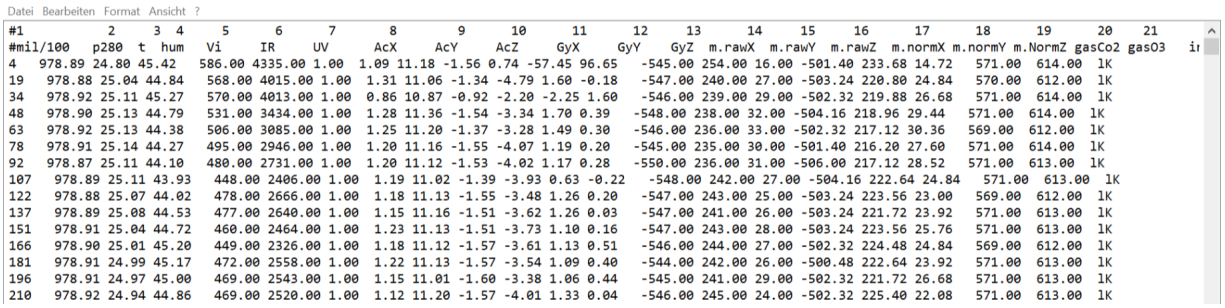
Als Vergleich der beiden Typen von Methodenrümpfen, die zum Auslesen der Sensoren angewendet werden, dient die Subroutine des BMP280. Hier wird schon in der Anzahl der Codezeilen der große Vorteil einer eingebundenen, sensorspezifischen Softwarebibliothek deutlich. So besteht das eigentliche Auslesen des Sensors und die Speicherung in lokalen Variablen aus nur noch drei Zeilen Code. Hier werden von der in der Präambel definierten Klasse „bme.“ wertspezifische Methoden zu Lufttemperatur, Luftfeuchtigkeit und Luftdruck aufgerufen, in denen die Messwertentnahme aus den Speicherregistern des Sensors stattfindet. Die Speicherung der Werte auf die SD-Karte ist analog zu der des MPU6050 zu lesen. Da die Subroutine des BME280 die Erste eines Messzyklus ist, wird in ihr zuerst der Zeitpunkt der Messung in die Datei auf der SD-Karte geschrieben (millis () / 100;).

```
void BME280(File measuredD1, float tempoutdoor) {
  float temp = bme.readTemperature();
  tempoutdoor = temp;
  float pres280 = (bme.readPressure() / 100.0F);
  float relHum = bme.readHumidity();                                    5

  if (measuredD1) {
    measuredD1.print(millis() / 100);
    measuredD1.print("    ");
    measuredD1.print(pres280);                                          10
    measuredD1.print(" ");
    measuredD1.print(temp);
    measuredD1.print(" ");
    measuredD1.print(relHum);
    measuredD1.print("    ");                                           15
  }
  Serial.print(temp); Serial.print(" "); Serial.print(pres280);
  Serial.print("      "); Serial.print(relHum); Serial.print(" ");
}
```

2.3 Auswertung

Die Auswertung der Daten, die bei einem Flug durch die Atmosphäre gesammelt werden, muss als große zweite Hälfte des Projekts verstanden werden. Der wissenschaftliche Aufwand ist deutlich größer als bei der Konzeption und Umsetzung der Sonde [5, S. 10-18].



```
Datei  Bearbeiten  Format  Ansicht  ?
#1           2     3  4     5      6      7       8       9      10      11      12     13      14     15     16      17      18      19      20    21
#mil/100   p280  t  hum   Vi     IR     UV      AcX     AcY     AcZ     GyX     GyY    GyZ  m.rawX  m.rawY  m.rawZ  m.normX m.normY m.NormZ gasCo2 gasO3   ir
4   978.89 24.80 45.42   586.00 4335.00 1.00  1.09 11.18 -1.56 0.74 -57.45 96.65   -545.00 254.00 16.00 -501.40 233.68 14.72   571.00  614.00  1K
19   978.88 25.04 44.84   568.00 4015.00 1.00  1.31 11.06 -1.34 -4.79 1.60 -0.18   -547.00 240.00 27.00 -503.24 220.80 24.84   570.00  612.00  1K
34   978.92 25.11 45.27   570.00 4013.00 1.00  0.86 10.87 -0.92 -2.20 -2.25 1.60   -546.00 239.00 29.00 -502.32 219.88 26.68   571.00  614.00  1K
48   978.90 25.13 44.79   531.00 3434.00 1.00  1.28 11.36 -1.54 -3.34 1.70 0.39   -548.00 238.00 32.00 -504.16 218.96 29.44   571.00  614.00  1K
63   978.92 25.13 44.38   506.00 3085.00 1.00  1.25 11.20 -1.37 -3.28 1.49 0.30   -546.00 236.00 33.00 -502.32 217.12 30.36   569.00  612.00  1K
78   978.91 25.14 44.27   495.00 2946.00 1.00  1.20 11.16 -1.55 -4.07 1.19 0.20   -545.00 235.00 30.00 -501.40 216.20 27.60   571.00  614.00  1K
92   978.87 25.11 44.10   480.00 2731.00 1.00  1.20 11.12 -1.53 -4.02 1.17 0.28   -550.00 236.00 31.00 -506.00 217.12 28.52   571.00  613.00  1K
107   978.89 25.11 43.93   448.00 2406.00 1.00  1.19 11.02 -1.39 -3.93 0.63 -0.22   -548.00 242.00 27.00 -504.16 222.64 24.84   571.00  613.00  1K
122   978.88 25.07 44.02   478.00 2666.00 1.00  1.18 11.13 -1.55 -3.48 1.26 0.20   -547.00 243.00 25.00 -503.24 223.56 23.00   569.00  612.00  1K
137   978.89 25.08 44.53   477.00 2640.00 1.00  1.15 11.16 -1.51 -3.62 1.26 0.03   -547.00 241.00 26.00 -503.24 221.72 23.92   571.00  613.00  1K
151   978.91 25.04 44.72   460.00 2464.00 1.00  1.23 11.13 -1.51 -3.73 1.10 0.16   -547.00 243.00 28.00 -503.24 223.56 25.76   571.00  613.00  1K
166   978.90 25.01 45.20   449.00 2326.00 1.00  1.18 11.12 -1.57 -3.61 1.13 0.51   -546.00 244.00 27.00 -502.32 224.48 24.84   569.00  612.00  1K
181   978.91 24.99 45.17   472.00 2558.00 1.00  1.22 11.13 -1.57 -3.54 1.09 0.40   -544.00 242.00 26.00 -500.48 222.64 23.92   571.00  613.00  1K
196   978.91 24.97 45.00   469.00 2543.00 1.00  1.15 11.01 -1.60 -3.38 1.06 0.44   -545.00 241.00 29.00 -502.32 221.72 26.68   571.00  613.00  1K
210   978.92 24.94 44.86   469.00 2520.00 1.00  1.12 11.20 -1.57 -4.01 1.33 0.04   -546.00 245.00 24.00 -502.32 225.40 22.08   571.00  613.00  1K
```

Abbildung 6: Beginn des Datensatzes des ersten Stratosphärenfluges vom 22.08.2016. Die jeweilige Werteschar, die ein einzelner Sensor liefert, ist durch den größeren Abstand der Spalten zu erkennen.

In Abbildung 6 ist der Anfang des Datensatzes zu sehen. Folgende Messgrößen wurden in dieser Reihenfolge aufgezeichnet: Zeit ($\frac{s}{10}$), Lufttemperatur (°C), Luftfeuchtigkeit (%), Lichstärke (lux), relative Infrarotstrahlung, UVindex, Beschleunigung x-Achse, y-Achse, z-Achse ($\frac{m}{s^2}$), Winkelgeschwindigkeit x-Achse, y-Achse, z-Achse ($\frac{^\circ}{s}$), Magnetische Kraft auf x-Achse, y-Achse, z-Achse (Gauss)(Rohwerte und von der Software weiterverarbeitete Normwerte) CO_2-Konzentration (ppm), O_3-Konzentration (ppm). Die letzte Angabe in der Reihe beschreibt den Zustand der Heizung [5]. Der erste Stratosphärenflug lieferte eine 3.775 Kilobyte große Datei. Zu gebrauchen waren allerdings nur etwa 534 Kilobyte, da die Sonde innerhalb des Zeitraums der Erfassung dieser Daten auf- und abstieg. Die Flugzeit betrug 97,6 Minuten. Durch großzügige Kalkulation wurde die Sonde etwa 10,6 Stunden mit Energie versorgt. Die Aufbereitung und Darstellung der Messgrößen geschieht mithilfe von GnuPlot.

2.3.1 Temperaturgradient

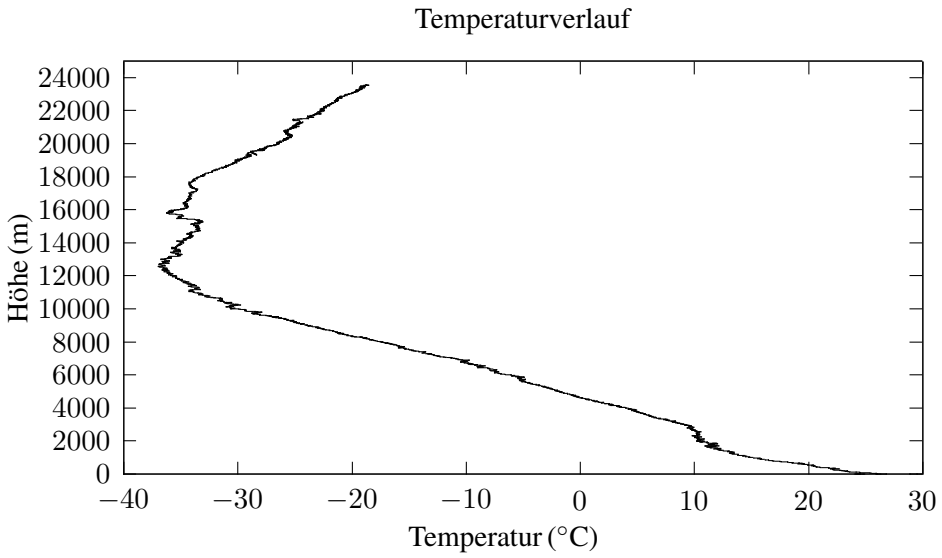


Abbildung 7: Lufttemperaturgradient geplottet mit den Werten des ersten Stratosphärenfluges am 22.08.2016

Abbildung 7 zeigt den Verlauf der Temperatur in Abhängigkeit der Flughöhe. Hierfür wurde der Datensatz des Fluges bis zu dem Punkt des niedrigsten Druckes genutzt (vgl. [5, Abb. x]). Wie es theoretische Modelle und andere praktisch erbrachte Beweise beschreiben, weist der obenstehende Graph einen charakteristischen Verlauf auf: Es ist ein nahezu konstanter Abfall der Temperatur von 25 °C am Boden bis zu -36 °C in einer Höhe von ca. 12100 Metern zu verzeichnen. Diese Tiefsttemperatur wird im weiteren Verlauf des Aufstiegs nicht mehr erreicht. Im Gegenteil steigt sie nach Stagnation bis 18000 m wieder auf -18 °C am höchsten Punkt des Fluges an. Das oben genannte Minimum der Temperatur wird in der Atmosphärengeographie als Temperaturinversion bezeichnet. Diese liegt an den Polen in 6 bis 8 und am thermisch aktiveren Äquator in 17 km Höhe, sodass sie für 50° geographischer Breite in 10 bis 12 km Höhe zu erwarten ist [23, S. 47, 48]. Luftmassen werden durch eine Temperaturumkehr daran gehindert, höher aufzusteigen. Die Varriation der Temeratur im Bereich von -32 bis -36 °C auf einer Höhendifferenz von ca. 6 km ab 12000 m Höhe ist auf die Schicht aus O_3 zurückzuführen. Werden diese Werte der Temperatur mit der Ozonkonzentration verglichen, zeigt sich eine qualitative Korrelation auf. Die hohe UV(c)-Albedo des Gases führt zur Reflektion dieser Strahlung, wodurch die darüber liegenden Luftschichten aufgewärmt werden.

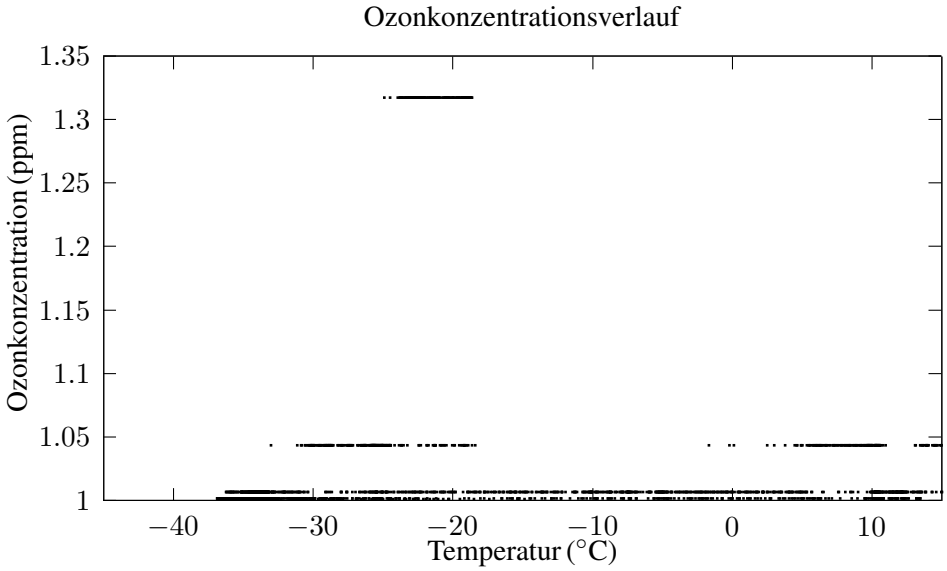


Abbildung 8: Ozonkonzentration in Abhängigkeit von der Temperatur

Auffällig ist dennoch die verhältnismäßig milde Tiefsttemperatur. Diese bleibt deutlich über -40°C, wird jedoch in vielen Quellen im Bereich -50°C bis -60°C angegeben [23, S. 47, 45] [24] [25].

2.3.2 Theoretischer Ansatz zur Bestimmung der Magnetfeldinklination

Der Begriff der Magnetfeldinklination beschreibt den Winkel, in dem das Erdmagnetfeld in den Erdboden eintritt. Das Magnetfeld der Erde wird vereinfacht mit dem eines Stabmagneten verglichen. So beträgt dessen Inklination an den magnetischen Polen, die leicht verschoben zu den geographischen Polen liegen, 90° und am geomagnetischen Äquator 0° [26].

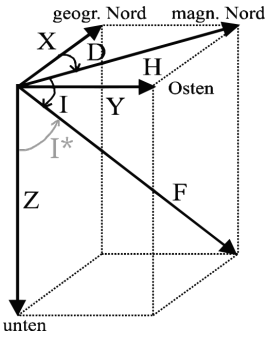


Abbildung 9: [27] Der Gesamtvektor $\vec{F}$, der aus den drei Teilvektoren $\vec{X}$, $\vec{Y}$ und $\vec{Z}$ resultiert, stellt das Magnetfeld dar. I steht in der Abbildung für den Inklinationswinkel des Erdmagnetfeldes.

Die Inklination lässt sich durch $I = 90° - I^*$ berechnen. I^* wird hierbei folgendermaßen bestimmt:

$$I^* = \cos^{-1}\left(\frac{\vec{F} \circ \vec{Z}}{\mid \vec{F} \mid \cdot \mid \vec{Z} \mid}\right) \tag{1}$$

Der Magnetfeldvektor $\vec{F}$ wird durch die Komponenten $m.\vec{norm}X$, $m.\vec{norm}Y$ und $m.\vec{norm}Z$ bestimmt. Da die Box schwankt und im Verlauf des Fluge demnach fast nie waagerecht auf $\vec{Z}$ steht, muss dieser Vektor, der „unten“ beschreibt, durch den Beschleunigungsvektor, der aus den Teilbeschleunigungen $\vec{AcX}$, $\vec{AcY}$ und $\vec{AcZ}$ besteht und immer zum Erdmittelpunkt gerichtet ist, ausgedrückt werden. Demnach berechnet sich die Inklination des Magnetfeldes durch

$$I = 90^\circ - \cos^{-1}\left(\frac{AcZ \cdot m.normZ + AcX \cdot m.normY + AcY \cdot m.normX}{\sqrt{m.normZ^2 + m.normY^2 + m.normX^2} \cdot \sqrt{AcZ^2 + AcX^2 + AcY^2}}\right) \quad (2)$$

Zu beachten ist die Befestigung der Sensorplatine an einer Wand des Sondenwürfels und die Befestigung der Sensoren auf dieser im Verhältnis zu dem jeweils anderen. Im Skalarprodukt müssen gleichgerichtete Achsen multipliziert werden, auch wenn die Bezeichnung etwas anderes impliziert. So ist (2) nur auf die Boxkonstruktion des zweiten Prototypen anwendbar und müsste im Falle einer waagerechten Platinenbefestigung angepasst werden. Die praktische Umsetzung dieser Überlegungen war wegen in 3.1 aufgeführter Gründe nicht möglich.

3 Zusammenfassung und Ausblick

3.1 Zusammenfassung

Die Aufgabenstellung zu Beginn des Projekts lautete eine Messeinheit zu konstruieren und zu programmieren. Basiernd auf der Mikrocontrollerplattform Arduino soll diese umweltrelevante Messwerte registrieren und für eine weitere Verarbeitung speichern. In 2.2 wurden die einzelnen Komponenten nämlich Sensoren, Arduino Mikrocontroller und SD-Karten Shield, Software und Kommunikationsprotokolle, die hierfür benötigt werden genauer charakterisiert. So funktioniert die Messwerterfassung nur, wenn alle diese Teile optimal aufeinander abgestimmt sind. Der resultierende Datensatz des Fluges zeigt, dass alles wie beschrieben zusammenarbeitet. In 2.3 wurde gezeigt, wie die gewonnen Erkenntnisse über das Verhalten der Temperatur in Abhängigkeit der Höhe theoretische Modelle belegen können. Zudem wurde ein theoretischer Ansatz zur Bestimmung der Magnetfeldinklination im Verlauf der Höhe entwickelt, der verwertbare Daten liefert, sobald Verbesserungen an der Sonde vorgenommen wurden. Große Teile der Auswertung sind in [5] nachzulesen.

3.2 Verbesserungen an der bestehenden Sonde

Im Bezug auf Ortung und Wiederfinden der Sonde war der erste Stratosphärenflug ein Erfolg. Auch wurden die Daten optimal und mit keiner Unterbrechung auf die SD-Karte geschrieben. Doch es gibt kleine Auffälligkeiten, die in der retrospektiven Betrachtung aufgefallen sind und die bis zum nächsten Flug behoben werden müssen.
Leider war der Lichtsensor SI1145L nach 17 Minuten Flug ausgefallen, sodass er ersetzt werden muss. Die Messrate der Sonde betrug bisher ca. 1 Messung in 1,3 Sekunden. Dieser Wert reicht aus um aufschlussreiche Aussagen über Höhen-, Luftfeuchtigkeits-, Temperatur-, und Gasgradienten zu treffen. Jedoch wird so eine dreidimensionale Positionsermittelung mittels der Winkelgeschwindigkeiten, der Achsenbeschleunigungen und der Magnetfeldteilvektoren erschwert, da sich die Sondenbox während der Messpause aufgrund von Windböen oder der Pendelbewegung der Schnur um einen nicht vernachlässigbaren Faktor von dem Ort der letzten Messung wegbewegt. Auch würde die diesem Punkt vorangegangene Überlegung zur Ermittelung der Magnetfeldinklination verwertbare Werte liefern. Eine Ticrate, die die genannten Punkte ermöglichen würde, liegt im Bereich von 200 Hz bis 400 Hz. Der Code muss entsprechend angepasst werden. Die Werte der Beschleunigungen weisen einen Betragsoffset von ca. 1,3 auf jeder Achse auf. Die Sensorplatine war senkrecht an der Sondenbox befestigt. Bringt man sie in die Waagerechte, verschwindet der Offset und die Werteschar zeigt die erwartete „9,81-0-0“-Zusammensetzung. Dieser Umstand macht eine erneute Kalibrierung des Sensors MPU6050 unabdingbar.

3.3 Erweiterungen

Es wird eine redundante Höhenerfassung geben. Die Messung des Druckes wird durch eine Speicherung des Höhenwertes ermittelt von einem Arduino UNO angesteuerten GPS-Tracker Shields ergänzt. So kön-

nen Messfehler des Sensors oder des GPS-Trackers erkannt werden. Ebenso wird eine Speicherung der Position der Sonde in geographischer Länge und Breite erfolgen, was die Ermittelung und dreidimensionale Rekonstruktion der Flugkurve erlaubt. Zudem wird die Technik des Siliziumphotomultipliers in die Sonde integriert. Mit dieser relativ neuen Technologie ist es möglich, ionisierende Strahlung in der Atmosphäre zu detektieren [28]. In Szintillatorplatten wird ein Photon ausgeschlagen, welches von den SiMPs detektiert wird. Die Vorteile derartiger Detektoren sind das geringe Gewicht, die geringe Größe und die Kompatibilität mit dem Arduino-Ökosystem . So kann mit dem Projekt „Neuland“ betreten werden.
Unabhängig davon, ob die zuletzt genannte Erweiterung praktisch umsetzbar ist und wie beschrieben funktioniert, wird es in jedem Fall weiter Flüge geben. Es müssen bestehende Aussagen verifiziert und aus den restlichen Daten weitere geschlossen werden.

4 Danksagung

Das Projekt wäre in seiner beschriebenen Form ohne großartige Unterstützung nicht möglich gewesen. Besonderer Dank gilt hierbei dem Erlanger Schülerforschungszentrum (ESFZ), allen voran den Studenten Stefan Berger und Patrick Hufschmidt. Der Großteil unserer Projektarbeit fand innerhalb der fünf Forschungswochen statt und brachte uns jedes Mal aufgrund der herausragenden finanziellen und inhaltlichen Unterstützung weiter.
Eine sehr zentrale Unterstützung für die Durchführung des Freiflugs stellte die finanzielle Hilfestellung durch den Jugend-Forscht Sponsorpool Bayern dar. Die Mittel wurden für den Ballon, den Fallschirm und den Tracker verwendet, wodurch die zeitnahe Durchführung letztendlich erst möglich wurde.
Der eigentliche Initiator unseres Projekt und der größte Unterstützer von Seiten der Schule ist unser offizieller Projektbetreuer Dr. Stefan Bärthlein.

Literatur

[1] http://www.planet-wissen.de/technik/luftfahrt/ballons/index.html, September 2015. zuletzt aufgerufen am 17. 10. 2016.

[2] https://de.wikipedia.org/wiki/Jacques_Alexandre_C%C3%A9sar_Charles#cite_note-1, Mai 2016. zuletzt aufgerufen am 17.10.2016.

[3] Victor Franz Hess. Durchdringende strahlung bei sieben freiballonfahrten. *Physikalische Zeitschrift*, pages 1084–1091, 1912.

[4] https://www.dwd.de/DE/service/lexikon/Functions/glossar.html;jsessionid=CAA1ADAAAE748B63A1063974D176C6EE.live21063?lv2=102134&lv3=102176. zuletzt aufgerufen am 19.10.2016.

[5] Niclas Popp. *Praktische Konzeption einer Klimasonde*, November 2016.

[6] Jeremy Blum. *Exploring Arduino*. John Wiley & Son, Ink., 2013.

[7] https://www.arduino.cc/en/Main/ArduinoBoardMega2560, 2016. zuletzt aufgerufen am 19.09.2016.

[8] http://www.robotshop.com/blog/en/files/arduino-uno-mega2560-front.jpg. zuletzt aufgerufen am 14.8.2016.

[9] https://www.arduino.cc/en/Main/ArduinoBoardUno, 2016. zuletzt aufgerufen am 19.09.2016.

[10] Bosch Sensortec GmbH. *Bosch BME280 - Combined humidity and pressure sensor*. Reutlingen, Mai 2015.

[11] Silicon Labroratories Inc. *Proximity/UV/Ambient Light Sensor IC with I^2 Interface*. Austin, o. A. o. A.

[12] InvenSense Inc. *MPU-6050 Produkt Specification*. Sunnyvale, September 2013.

[13] Honeywell. *3-Axis Digital Compass IC HMC5883L*. Plymouth, Februar 2013.

[14] http://d1j1kxp9fqehmk.cloudfront.net/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/0/4/04_31.jpg. zuletzt aufgerufen am 06.11.2016.

[15] https://www.adafruit.com/category/17?q=M&. zuletzt aufgerufen am 18.09.2016.

[16] http://img.dxcdn.com/productimages/sku_154602_2.jpg. zuletzt aufgerufen am 07.11.2016.

[17] Sd:http://i00.i.aliimg.com/wsphoto/v0/1645586096_3/Blue-Perfect-SD-Card-Module-Slot-Socket-Reader-For-Arduino-ARM-MCU-Read-Wr jpg. zuletzt aufgerufen am 09.09.2016.

[18] https://de.wikipedia.org/wiki/Kommunikationsprotokoll, April 2016. zuletzt aufgerufen am 24.09.2016.

[19] Klaus Wüst. *Microprozessortechnik*. Vieweg+Teubner Verlag, 4. edition, 2011. einzusehen unter https://books.google.de/books?id=sTkgBAAAQBAJ&pg=PA272&lpg=PA272&dq=daten%C3%BCbertragungsraten+von+I2c&source=bl&ots=wYNVJtXoDm&sig=8k7rGTzehFQKSN-oP1yuA3cS_Sc&hl=de&sa=X&ved=0ahUKEwjl9Y-u1LHPAhVB1iwKHTbeAc4Q6AEIIDAA#v=onepage&q=daten%C3%BCbertragungsraten%20von%20I2c&f=false.

[20] https://www.arduino.cc/en/Guide/Environment, 2016. zuletzt aufgerufen am 05.10.2016.

[21] https://de.wikipedia.org/wiki/Arduino_(Plattform), Oktober 2016. zuletzt aufgerufen am 04.10.2016.

[22] https://de.wikipedia.org/wiki/Programmbibliothek, Juni 2016. zuletzt aufgerufen am 07.10.2016.

[23] Christian-Dietrich Schönewiese Christoph Buchal. *Klima - Die Erde und ihre Atmosphäre im Wandel der Zeiten*. Wilhelm und Else Heraeus-Stiftung, Helmholtz-Gemeinschaft Deutscher Forschungszentren, 2010.

[24] https://de.wikipedia.org/wiki/Atmosph%C3%A4rischer_Temperaturgradient, Mai 2016. zuletzt aufgerufen am 25.10.016.

[25] http://wiki.bildungsserver.de/klimawandel/index.php/Aufbau_der_Atmosph%C3%A4re, Oktober 2016. zuletzt aufgerufen am 25.10.2016.

[26] https://de.wikipedia.org/wiki/Inklination_(Magnetismus), Oktober 2016. zuletzt aufgerufen am 02.11.2016.

[27] http://www-app3.gfz-potsdam.de/obs/niemegk/dt/observatorium/magnetfeld/komponenten.jpg, Juni 2008. zuletzt aufgerufen am 30.10.2016.

[28] Judith Schneider. *Charakterisierung von Siliziumphotomultipliern*. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Oktober 2015.

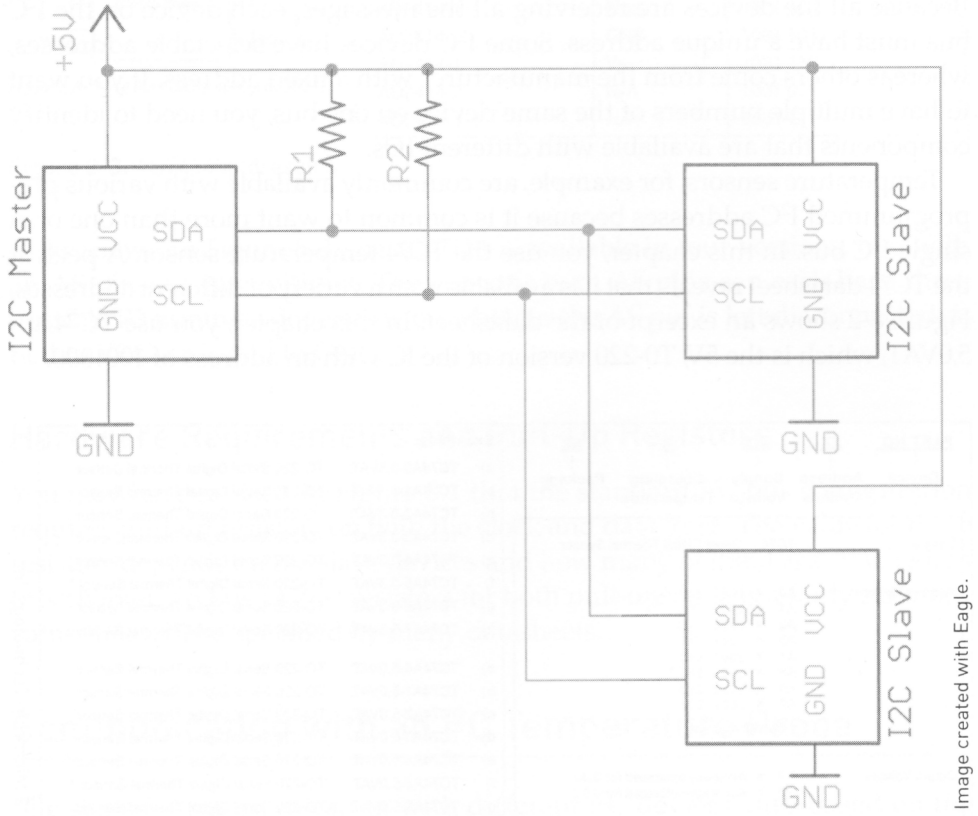


Figure 8-1: I²C reference hardware configuration

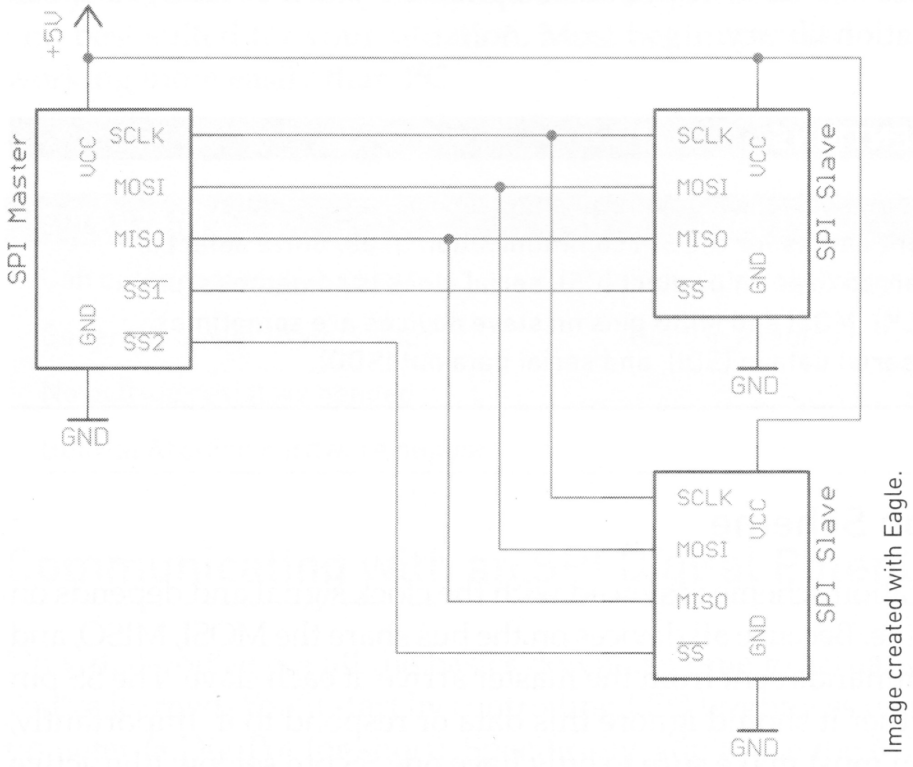


Figure 9-1: SPI reference hardware configuration

Ich erkläre hiermit, dass ich die Seminararbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benützt habe.

Bad Windsheim, den 07. November 2016 _____