

1 Inhaltsverzeichnis

2	Projekt.....	5
2.1	Entstehung.....	5
2.2	Aufgabenstellung.....	5
3	Pflichtenheft.....	6
3.1	Ziel.....	6
3.2	Funktionsumfang.....	6
3.3	Leistungen und Genauigkeit.....	6
3.3.1	Sensoren.....	6
3.3.2	Sensorboard.....	6
3.3.3	Messwertanzeige.....	6
3.4	Technische Produktumgebung.....	7
3.4.1	Entwicklungsumgebungen und Programmiersprachen.....	7
3.4.2	Hardware.....	7
3.5	Benutzeroberfläche.....	7
3.5.1	Sensorboard.....	7
3.5.2	Auswertung.....	7
4	Terminplan.....	8
4.1	Einleitung.....	8
4.2	Aufwandsschätzung.....	8
4.3	Aufwandkontrolle.....	9
4.4	Kommentar.....	9
5	Konzept.....	10
5.1	Grundgedanke.....	10
5.2	Konzeptdiagramm.....	10
6	Kommunikationsprotokoll.....	11
6.1	Ausgangslage und Anforderungen.....	11
6.2	Definition.....	11
6.2.1	Kontinuierliche Messung.....	11
6.2.2	Statusabfragen.....	12
6.2.1	Ping.....	12
6.3	Kommentar.....	12
7	IMU Hardware.....	13
7.1	Sensoren.....	13
7.1.1	Anforderungen.....	13
7.1.2	Auswahl.....	13
7.2	Prozessor.....	14
7.2.1	Anforderungen.....	14

7.2.2	Auswahl.....	14
7.2.3	Beschreibung.....	15
7.2.4	Programmierschnittstelle.....	15
7.2.5	Beschaltung.....	15
7.3	Grafikdisplay.....	15
7.4	Funkmodul.....	16
7.5	Schema.....	16
7.6	Stückliste.....	17
7.7	Aufbau.....	18
7.8	Bauphase.....	18
7.9	Inbetriebnahme.....	20
8	AVR Software.....	21
8.1	Funktionsprinzip.....	21
8.2	Ablaufschema.....	21
8.3	Aufbaudiagramm.....	22
8.4	Software.....	22
8.4.1	I2C Treiber.....	22
8.4.2	UART Treiber.....	23
8.4.3	Display Treiber.....	23
8.4.4	Timer / Interrupts.....	24
8.4.5	Magnetometer.....	24
8.4.6	Gyro.....	25
8.4.7	Acceleromter.....	25
8.4.8	Initialisierung.....	25
8.4.9	Hauptschleife.....	25
8.5	Konfiguration XBee.....	26
9	PC Software.....	27
9.1	Funktionsprinzip.....	27
9.2	Ablaufschema.....	27
9.3	Klassendiagramm.....	28
9.4	Software.....	28
9.4.1	UART.....	28
9.4.2	AVRInterface.....	29
9.4.3	Converter.....	29
9.4.4	Filter.....	29
9.4.5	DCM Filter.....	30
9.4.6	SensorData3.....	30
9.4.7	Log.....	30
9.4.8	IMU.....	30
9.4.9	Frontend.....	31
9.5	Controls.....	31
9.5.1	RotaryControl.....	31
9.5.2	CubeControl.....	32

9.6	Oberfläche	32
10	Tests und Auswertung	34
10.1	Software.....	34
10.2	Messungen	34
10.2.1	Filtertest.....	34
10.2.2	Sensortest	35
10.2.3	Spannungsmessung.....	35
10.2.4	Temperaturmessung.....	35
10.2.5	Verbrauchsmessung.....	35
10.3	Auswertung.....	35
10.3.1	Messresultate	35
10.3.2	Funktionsumfang	36
11	Rückblick und Schlusswort.....	37
11.1	Projekt.....	37
11.2	Hardware	37
11.3	Software.....	37
11.4	Dokumentation.....	37
11.5	Zukünftiges.....	37
12	Abkürzungen / Erklärungen.....	39
13	Verzeichnisse	40
13.1	Informationsquellen	40
13.2	Tabellenverzeichnis	40
13.3	Abbildungsverzeichnis	40
13.4	Formelverzeichnis.....	41
14	Anhang	42
14.1	Persönliche Angaben	42
14.2	Bedienungsanleitung.....	43
14.2.1	Installation.....	43
14.2.2	Betriebsanleitung	43
14.3	Aufgabenstellung	44
14.4	Sitzungsprotokoll.....	45
14.4.1	Sitzung 1.....	45
14.4.2	Sitzung 2.....	46
14.5	CD.....	47
14.5.1	Datenblätter	47
14.5.2	Dokumente	47
14.5.3	Hardwareschemas.....	47
14.5.4	Sitzungsprotokolle.....	47
14.5.5	Software	47
14.5.6	Filme	47
14.5.7	Bilder.....	47
14.5.8	Zusätzliches	47

2 Projekt



Abbildung 1 Fliegender Oktokopter

2.1 Entstehung

Die Idee für ein derartiges Projekt kam mir bereits vor einem Jahr, als ich für meine damalige Firma einen Prototyp eines Oktokopters für Flugaufnahmen zusammenbaute. Die Technologie hinter der Lagekontrolle für ein derartiges Gerät interessierte mich sehr und ich fragte mich, ob es nicht auch mit einem deutlich einfacheren Aufbau möglich wäre, ähnliche Resultate zu erzielen. Ein weiterer Anreiz war es, nicht nur Software zu entwickeln, sondern auch beim Entstehen der Hardware mitzuwirken.

Deshalb reichte ich als Vorschlag für meine Vordiplomarbeit dieses Projekt ein. Es beinhaltet das aufbauen und auswerten eines Inertialsensors, auch genannt IMU (inertial measurement unit).

2.2 Aufgabenstellung

Folgende zugeteilte Vordiplomarbeit soll selbstständig gelöst werden:

Vordiplomarbeit 2011 – Inertial Sensor Board

Inertialsensoren (Trägheitssensoren) werden zur Messung von Bewegungen verwendet. Sie sind ein Hauptbestandteil von Navigationssystemen und kommen auch in der Bildstabilisierung von optischen Systemen vor.

Mit dieser Vordiplomarbeit soll ein IMU Development Board so programmiert werden, dass es die Sensordaten auswerten und an einen PC übertragen kann. Der PC soll die Daten grafisch aufbereitet darstellen können.

Die Aufgabe umfasst folgende Punkte:

- Projektplanung
- IMU Development Board hardwaremässig komplettieren um Bewegungen messen zu können
- Programmierung des Sensor Boards um Sensordaten auslesen und zum PC übertragen zu können
- Programmierung einer PC-Software, welche die Daten entgegennehmen und aufbereiten kann
- Visualisierung der Sensordaten mit der PC-Software
- Test der Software
- Dokumentation aller Arbeitsschritte

Optionale Ziele:

- Möglichkeit der Filterung der Daten (Fehlerkorrektur)
- Aufbau einer Plattform zur Demonstration des Sensorboards

3 Pflichtenheft

3.1 Ziel

Das Ziel dieser Vordiplomarbeit ist es, ein Sensorboard und dazugehörige Software zu entwickeln, welche die grundlegenden Funktionen und Möglichkeiten eines Inertialsensors vereinfacht zur Verfügung stellt. Das Sensorboard wird so aufgebaut, dass Messungen der Dreh- und Beschleunigungswerte sowie des Magnetfeldes in allen drei Dimensionen möglich ist. Die Daten werden von einem Mikrocontroller gesammelt und zwischengespeichert, bis diese dann über ein Funkmodul an den Computer gesendet werden. Das Board verfügt über einen Akku, welcher den Betrieb ohne Kabel über mehrere Stunden ermöglicht. Die Computer Software nimmt schlussendlich die Daten entgegen und stellt sie grafisch auf dem Monitor dar.

3.2 Funktionsumfang

Folgende Musskriterien sind für dieses Projekt definiert:

- Das Sensorboard ist Funktionsfähig.
- Funktionierende Software zum Auswerten der Daten.
- Gemessen wird die Beschleunigung, Drehbewegung und das Magnetfeld.
- Die Daten werden möglichst in Echtzeit am Computer dargestellt.
- Das Projekt ist sauber dokumentiert.

Folgende Kriterien sind wünschenswert:

- Einen Filter implementieren welcher die Daten aufbereitet.
- Erweiterte grafische Oberfläche um die gefilterten Daten anzuzeigen.
- Testplattform erstellen welche die Funktionen demonstriert.

3.3 Leistungen und Genauigkeit

3.3.1 Sensoren

Da es praktisch nicht möglich ist, die Genauigkeit der Sensoren zu überprüfen ohne geeignete Messgeräte, gilt es diese bestmöglich nach Datenblatt zu initialisieren. Weitere Möglichkeiten zur Kalibrierung sind je nach Sensor zu integrieren um das Messergebnis weiter zu verbessern.

3.3.2 Sensorboard

Das auf dem Sensorboard integrierte Funkmodul sollte eine Reichweite von mindestens 10 Meter Sichtlinie gewährleisten. Der Akku welcher das Sensorboard mit Strom versorgt, sollte für mindestens 4 Stunden Betrieb Strom liefern. Die Bauteile sind daher so auszulegen, dass der Gesamtstromverbrauch möglichst tief gehalten wird.

3.3.3 Messwertanzeige

Die Anzeige der aktuellen Messwerte soll möglichst in Echtzeit und flüssig geschehen. Beim Aufbau der Hardware sowie beim Entwickeln der Software ist daher auf die Performance zu achten.

3.4 Technische Produktumgebung

3.4.1 Entwicklungsumgebungen und Programmiersprachen

Das Sensorboard mit dem darauf verbauten Mikrocontroller wird in C programmiert. Als Entwicklungsumgebung kommt AVR Studio 4 von Atmel zum Einsatz mit dem freien C-Compiler AVR-GCC. Für die auf dem Computer laufende Software hingegen wird C# eingesetzt mit der IDE Visual Studio 2010 Ultimate.

3.4.2 Hardware

Für die Auswertung der Messdaten wird ein Computer benötigt mit Windows XP oder höher. Als zusätzliche Hardware wird ein ISP Programmiergerät vorausgesetzt, mit welchem der Mikrocontroller programmiert wird. Um den Akku zu laden, ist zudem ein Ladegerät speziell für LiPo-Akkus zwingend notwendig.

3.5 Benutzeroberfläche

3.5.1 Sensorboard

Um Zustandsinformationen auf dem Sensorboard anzuzeigen, soll ein Grafikdisplay integriert werden. Diese kann zusätzlich zum Debuggen der Software verwendet werden.

3.5.2 Auswertung

Die Messdaten sollen als umgerechnete Werte in einem Liniendiagramm auf dem Monitor dargestellt werden. Optional ist eine Anzeige der gefilterten Daten zu integrieren. Desweiteren sind Elemente zur Steuerung des Sensorbords vorhanden.

4 Terminplan

4.1 Einleitung

Die Terminplanung dient in dieser Vordiplomarbeit als grundlegendes Element der Kontrolle. Daher entstand bereits nach einem kurzen zusammentragen der grundlegendsten Informationen die dieses Projekt betreffen, eine grobe Zeitplanung. Diese wurde laufend mit aktuellen Werten ergänzt, um einen maximalen Überblick über den Fortschritt zu erhalten.

4.2 Aufwandsschätzung

Die nun folgende Tabelle zeigt den geschätzten Zeitaufwand der jeweiligen Arbeitsschritte. Die Schätzung beruht vorzugsweise auf Erfahrungswerten und ähnlichen, bereits realisierten Projekte aus der Vergangenheit. Genaue Angaben sind allerdings schwierig, da viele noch unbekannte Faktoren sowie Technologien vorhanden sind, die jederzeit zu Verzögerungen führen können. In jedem Arbeitsschritt ist deshalb eine Pufferzone von etwa 20% einkalkuliert. Sollte schlussendlich noch Zeit vorhanden sein, kann diese für weitere Wunschziele eingesetzt werden.

Vorgang	Datum	Geplante Dauer (h)
Erhalten der Aufgabenstellung	22.2.2011	
Pflichtenheft erstellen		5
Projektablauf und Zeitplan erstellen		5
Grundlagen der Funktionsweise der einzelnen Sensoren erarbeiten		15
Auswahl der Sensoren		5
Auswahl der restlichen Hardware (Prozessor, Kommunikationsinterface)		5
Hardwareschema ausarbeiten		15
Zusammenbau der Hardware		10
Planung der Mikrocontroller Software und des Kommunikationsprotokolls		10
Planung der PC Software		15
Sitzung 1	30.3.2011	
Realisierung der AVR Software		15
Realisierung der PC Software		10
Software und Hardware Tests		10
Sitzung 2	8.6.2011	
Dokumentation komplettieren		30
Präsentation vorbereiten		5
Abgabe Dokumentation	22.6.2011	
Präsentation	9.7.2011	
Zusatz: Filter studieren + implementieren		20
Zusatz: Grafischer Oberfläche erweitern		10
Zusatz: Demonstrationsplattform erstellen		20
Total		155 (+50)

Tabelle 1 Aufwandsschätzung

4.3 Aufwandkontrolle

Die Dauer wurden jeweils am Ende des Tages oder nach beendigen eines Arbeitsschrittes ergänzt. Das Datum wurde nach beendigen des Vorgangs eingetragen und dient zur groben Übersicht, wann diese erledigt wurden.

Vorgang	Datum	Dauer (h)
Erhalten der Aufgabenstellung	22.2.2011	
Pflichtenheft erstellen	30.2.2011	6
Projekttablauf und Zeitplan erstellen	4.3.2011	3
Grundlagen der Funktionsweise der einzelnen Sensoren erarbeiten	5.3.2011	17
Auswahl der Sensoren	7.3.2011	6
Auswahl der restlichen Hardware (Prozessor, Kommunikationsinterface)	7.3.2011	4
Hardwareschema ausarbeiten	10.3.2011	16
Zusammenbau der Hardware	15.3.2011	15
Planung der Mikrocontroller Software und des Kommunikationsprotokolls	20.3.2011	7
Planung der PC Software	25.3.2011	9
Sitzung 1	30.3.2011	
Realisierung der AVR Software	6.4.2011	25
Realisierung der PC Software	23.4.2011	10
Software und Hardware Tests	21.5.2011	5
Sitzung 2	8.6.2011	
Dokumentation komplettieren	21.6.2011	45
Präsentation vorbereiten	18.6.2011	6
Abgabe Dokumentation	22.6.2011	
Präsentation	9.7.2011	
Zusatz: Filter studieren + implementieren	19.5.2011	30
Zusatz: Grafischer Oberfläche erweitern	19.5.2011	15
Zusatz: Demonstrationsplattform erstellen		-
Total		174 (+45)

Tabelle 2 Aufwandkontrolle

4.4 Kommentar

Wie aus der Tabelle ersichtlich, wurde der Zeitplan teilweise massiv überschritten. Dies obwohl in jedem Vorgang noch zusätzliche Pufferzeiten vorhanden waren.

Die erste Abweichung ist beim Zusammenbau der Hardware zu erkennen. Diese ergibt sich durch den höheren Aufwand, welche durch die komplizierte Verdrahtung und Fehlersuche entstand.

Weitere Abweichungen sind zu erkennen beim Realisieren der Software. Diese sind allerdings nicht zurückzuführen auf mangelnde Planung, sondern durch immer wieder auftauchende unvorhersehbarer Probleme aufgrund zu geringer Erfahrungen in diesen Programmiersprachen.

Die Zeitüberschreitung bei der Dokumentation ergibt sich aus den ständig wiederkehrenden Problemen mit der Software. Komplizierte Fragen bezüglich der Formatierung, Abstürze und inkonsistente Zuständen führten zu ständigen Verzögerungen.

5 Konzept

5.1 Grundgedanke

Bereits zu Beginn des Projekts war klar, dass nicht die gesamte Logik und Auswertung auf dem Sensorboard selbst realisiert werden kann. Dies hätte einen grossen Aufwand zur Folge gehabt und wäre für spätere Testzwecke schlecht geeignet gewesen. Deshalb sind Filter und die grafische Darstellung auf einem sehr viel leistungsfähigeren Desktop Computer ausgelagert. Das Sensorboard hat nun die einzige Aufgabe, die Sensoren zu managen. Also zu Initialisieren, Register zu setzen und Daten abzufragen. Die Messwerte werden dann über eine Schnittstelle an den Computer übergeben. Dieser wandelt sie in die benötigten Einheiten um, filtert sie und stellt diese grafisch aussagekräftig auf einem Monitor dar.

5.2 Konzeptdiagramm

Dieses vereinfachte Diagramm zeigt den gesamten Zusammenhang von Computer und Sensorboard sowie die groben internen Verknüpfungen der Software.

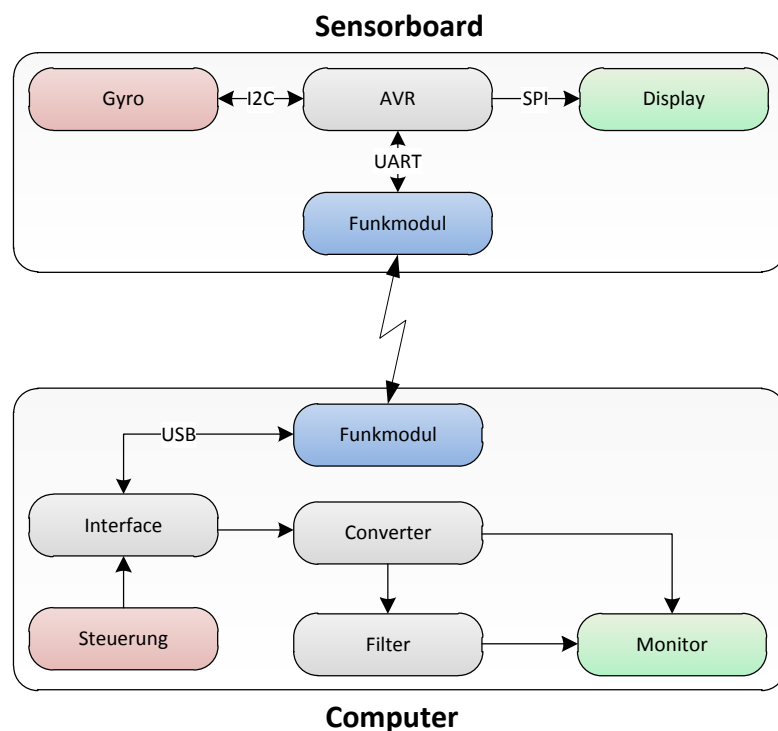


Abbildung 2 Konzeptdiagramm

6 Kommunikationsprotokoll

6.1 Ausgangslage und Anforderungen

Die Kommunikation zwischen Computer und Sensorboard erfolgt über eine UART Schnittstelle, welche eine einfach Grundlage zur Übermittlung von Daten zur Verfügung stellt.

Es sind allerdings mehrere kritische Punkte vorhanden, welche beim Entwerfen des Protokolls zu beachten sind.

- Es ist unklar, ob der AVR sich zu Beginn der Transaktion gerade in einem mehrere ms dauernden Interrupt befindet. Es könnten daher aufgrund des nur sehr kleinen UART Puffers, Teile der Nachricht verloren gehen.
- Es ist nicht festgelegt, dass nach einer Abfrage direkt die entsprechende Antwort folgt.
- Das eingesetzte Funkmodul verfügt nur über eine beschränkte Übertragungsgeschwindigkeit.
- Die Übertragungen sollten möglichst kurz sein, um den AVR nicht unnötig lange zu blockieren.
- Die Übertragung soll einen leserlichen Aufbau haben, um das Arbeiten über die Konsole zu ermöglichen.
- Logischer Aufbau der Nachricht, für eine möglichst einfache und ressourcenschonende Decodierung.

6.2 Definition

Grundlegen ist jeder Command gleich aufgebaut. Es handelt sich um eine einfache Doppelpunkt getrennte Parameterliste, wobei jeweils der erste Parameter den Inhalt definiert.

Die nun folgenden Definitionen enthalten alle zurzeit möglichen Befehle. Die variablen Parameter sind jeweils mit einem „{}“ bezeichnet, wobei deren Inhalt den Datentyp und das Zeichen „?“ einen optionalen Parameter darstellt. Am Ende eines Befehles wird mit `\n` abgeschlossen.

Die Orange markierten Zeilen sind Befehle des Sensorboards, die restlichen die der Computer Software.

6.2.1 Kontinuierliche Messung

Der kontinuierliche Messvorgang ist die wichtigste vorhandene Funktion. Ist sie gestartet, so wird alle 40ms ein Messergebnis zurückgeschickt welches die Daten aller drei Sensoren beinhaltet.

Das Messergebnis besteht aus 9 float zahlen welche die X,Y und Z Werte vom Gyro, Accelerometer und Magnetometer enthalten. Optional kann noch ein weiterer, frei definierbarer Parameter mitgegeben werden.

Command	Bemerkung
M:start	Messvorgang starten
M:stop	Messvorgang abbrechen
M:{F};{F};{F};{F};{F};{F};{F};{F};{F};{?}	Messergebnis

6.2.2 Statusabfragen

Diverse Werte des Sensorboards können zusätzlich abgefragt werden. Zurzeit handelt es sich dabei um die Werte der Akkuspannung und des Temperatursensors.

Command	Bemerkung
G:akku	Abfrage der Akkuspannung
A:{int16}	Aktuelle Akkuspannung
G:temp	Abfrage der Temperatur
T:{int16}	Aktuelle Temperatur

6.2.1 Ping

Um zu überprüfen ob das Sensorboard erreichbar ist, kann ein Ping Befehl abgesetzt werden. Die Antwort wird ebenfalls als Ping zurückgeschickt. Das Sensorboard sendet zudem beim Einschalten zur Begrüssung jeweils einen Ping.

Command	Bemerkung
P:{?}	Abfrage ob Sensorboard erreichbar
P:{?}	Begrüssung oder Antwort auf Ping

6.3 Kommentar

Das Kommunikationsprotokoll stellt sich bereits während der Aufbauphase als relativ optimal für dieses Projekt heraus. Von Anfang an war selbst das manuelle abfragen von Werten und das anschliessende auslesen sowie analysieren über die Konsole kein Problem. Auch später während längerem Betrieb und dem kontinuierlichem senden von Messwerten, hat sich nie ein fehlerauffälliges Verhalten gezeigt.

Die Zerlegung der Anfragen auf dem AVR hat sich allerdings nicht als ganz so einfach wie vermutet herausgestellt, da der Funktionsumfang in C sich etwas zu den gewohnten Methoden unterscheidet. Die Auswertung erfolgt nun aber zuverlässig und kann einfach erweitert werden.

7 IMU Hardware

7.1 Sensoren

Die Sensoren stellen das zentrale Element der Hardware dar, da von ihnen massgeblich das Endresultat abhängig ist. Mit minderwertigen oder stark rauschenden Sensoren wäre die schlussendliche Messung folglich von schlechterer Qualität. Um die Qualität zu gewährleisten, verlasse ich mich neben den Spezifikationen in den Datenblättern vor allem auf Erfahrungswerten von ähnlichen bekannten Systemen.

7.1.1 Anforderungen

Für einen Inertialsensor werden mindestens Beschleunigungssensor (Accelerometer) sowie Drehratensensoren (Gyros) benötigt. Beide müssen jeweils in dreifacher Ausführung vorhanden sein und orthogonal angeordnet werden, um die Messung in X,Y sowie der Z Achse zu ermöglichen. Um das Endresultat zu verbessern, soll zusätzlich ein Magnetfeldsensor (Magnetometer) integriert sein.

Die Sensoren müssen über einen I2C Busanschluss verfügen. Somit kann auf eine externen oder den im AVR integrierten A/D Wandler verzichtet werden, was viel Programmier- und integrationsarbeit erspart. Zusätzlich soll damit ein besseres Ergebnis erzielt werden, da bereits integrierte Wandler optimal auf den jeweiligen Sensor abgestimmt und beschaltet sind.

Die Sensoren sollten mindestens die folgenden Messbereiche abdecken.

- Accelerometer: +- 2g (+- 19.62m/s²)
- Gyro: +- 1 rad/s
- Magnetometer: +- 1G

7.1.2 Auswahl

Die definierten Anforderungen lassen viel unterschiedlich Sensoren zu. Um eine Entscheidung zu treffen, wurden drei der gängigsten, gut erhältlichen aber nicht zu teuren Sensorkombinationen miteinander verglichen.

	Razor IMU	9DOF Stick+ AVR	ITG-3200 + LSM303 + AVR
Preis	200 CHF	190 CHF	170 CHF
Auflösung	Gyro: +-300°/s, 0.83mV/°/s Accelerometer: +- 2g - +-16g, 4mg/LSB Magnetometer: +- 4G, 7mG / LSB	Gyro: +-2000°/sec, 14,375 LSB per °/sec Accelerometer: +- 2g - +-16g, 4mg/LSB Magnetometer: +-4G, 7mG / LSB	Gyro: +-2000°/sec, 14,375 LSB per °/sec Accelerometer : +=2g – +-8g, 3,3mg/LSB Magnetometer : +=1.3 G - +- 8.1G, 7.4mG / LSB
Erweiterbarkeit	Schwer erweiterbar, da der AVR bereits fest integriert ist.	Frei erweiterbar	Frei erweiterbar
Aufbauaufwand	Schaltung grösstenteils aufgebaut.	Sensoren sind bereits aufgebaut und über I2C ansprechbar.	Keine vormontierte Hardware.
Sonstiges	Demo Software vorhanden.	Geringe Abmessungen	
Auswertung	Die Schaltung ist grösstenteils aufgebaut, dadurch allerdings schlecht erweiterbar.	Gutes Preis/Leistung Verhältnis, geringe Abmessungen.	Gutes Preis/Leistung Verhältnis, allerdings viel Aufwand da Hardware nicht vormontiert ist.

Tabelle 3 Auswahlmatrix Sensor



Abbildung 3 9 DOF Sensor Stick

Mithilfe von diesem Vergleich viel die Auswahl auf einen 9DOF Sensor Stick in Verbindung mit einem zusätzlichen AVR. Der Aufwand ist damit etwas grösser als bei dem bereits fertig montierten Razor IMU, jedoch ist ein klarer Vorteil die frei erweiterbare Schaltung.

Die Abmessungen sind mit 10 x 35mm sehr gering, trotzdem ist aber eine Fixiervorrichtung in Form eines Bohrloches vorhanden. Die vier Anschlusspins sind für die Stromversorgung und den I2C Bus. Ein Integrierter Festspannungsregler ermöglicht eine Versorgung zwischen etwa 5 – 10V. Alle drei Sensoren haben einen sehr grossen Operationsbereich. Dieser Baustein bietet daher optimale Voraussetzungen für dieses Projekt.

7.2 Prozessor

Der Mikrocontroller ist für die Steuerung und Verarbeitung der einzelnen Sensoren sowie für die Kommunikation zu dem PC verantwortlich. Zum Einsatz wird ein Controller aus der Atmel AVR Familie kommen. Hauptsächlich darum, weil ich mit diesem bereits einige Erfahrungen gesammelt habe in anderen Projekten, andererseits aber auch, da er aufgrund unterschiedlichster verfügbaren Bauarten und Schnittstellen stark auf die jeweilige Anwendung skalierbar ist. Dank seiner grossen Verbreitung sind auch viele Referenzen vorhanden und ein guter Support für Hardware sowie Software.

7.2.1 Anforderungen

Die Anforderungen für den AVR sind relativ gross, da er nicht nur für dieses Projekt, sondern auch für spätere noch genug Leistungsreserven vorweisen soll.

Besonders wichtig ist, dass der I2C Busanschluss sowie UART für die Kommunikation über RS232, hardwareseitig integriert ist. Damit stehen die integrierten Speicherpuffer und Interrupts zur Verfügung. Ansonsten wäre eine Realisierung via Software notwendig, welches vor allem Programmierzeit und viel Prozessorleistung beanspruchen würde.

Da einige komplizierte Teile der C Bibliothek integriert werden und viel mit float Typen gerechnet wird, muss mindestens 8kB Flash und 2kB RAM Speicher vorhanden sein. Bedingt ist die RAM Grösse vor allem dadurch, dass bereits der Treiber des Grafikdisplays 1kB für den Zwischenspeicher benötigt.

7.2.2 Auswahl

Unterschiedliche AVR Modelle passen zu den gestellten Anforderungen. Für die Entscheidung wurden daher drei der gängigsten, welche allen Anforderungen genügen, miteinander verglichen.

	ATmega328	ATmega32	ATmega644
Preis	5 CHF	9 CHF	14 CHF
Flash	32kB	32kB	64kB
Ram	2kB	2kB	4kB
I/Os	23	32	32
Timer	2x 8-bit, 1x16-bit	2x 8-bit, 1x 16bit	2x 8-bit, 1x 16bit
Peripherie	I2C, UART, SPI, ISP	I2C, UART, SPI, JTAG, ISP	I2C, UART, SPI, JTAG, ISP
Taktfrequenz	< 20 MHz	< 16 MHz	< 20 MHz
Anzahl ADC	8	8	8
PWM Kanäle	6	5	6
Auswertung	Sehr gutes Preis / Leistung Verhältnis.	Die Maximale Taktfrequenz ist etwas zu Tief.	Der Preis ist höher, doch der grosse Speicher und die Anzahl an I/Os gleichen dies aus.

Tabelle 4 Auswahlmatrix Mikrocontroller

Die Entscheidung viel dann auf den ATmega644. Obwohl der ATmega328 für einen geringeren Preis ähnliche vielfalt bietet, ist der ATmega644 aufgrund der grösseren Anzahl I/Os und des doppelt so grossen RAMs besser geeignet.



Abbildung 4 AVR ATmega644

7.2.3 Beschreibung

Als Hauptprozessor wird ein ATmega644 verwendet. Es handelt sich dabei um einen 8 Bit Mikrocontroller der AVR Familie. Das Arbeiten mit diesem Prozessor ist relativ angenehm, da nur minimale Hardwareanforderungen zum Betrieb und wenig Software zum Programmieren notwendig ist. Er verfügt über 64kB Programmspeicher, 4kB SRAM und ein 2kB grosses EEPROM. Dies sollte auch für umfangreichere Programme genügen Platz zur Verfügung stellen. Gesamthaft sind 32 I/Os verfügbar, welche wahlweise als Ein- oder Ausgänge verwendet werden können. Zudem sind die meisten Pins mit mehreren Funktionen belegt, beispielsweise UART, I2C, SPI, PWM oder ADCs. Zwei unabhängige 8 Bit Timer und ein 16 Bit Timer ermöglichen unterschiedliche Interrupts und dienen als Basis der 6 PWM Ausgänge. Zudem verfügt der Mikrocontroller über einen internen 10 Bit A/D Aandler, welcher auf 8 Pins zur Verfügung steht. JTAG und ISP Programminterface, Hardware I2C und SPI Anschlüsse stehen ebenfalls bereit. Die Betriebsspannung erstreckt sich von 2.7 – 5.5 V, optimal für das Arbeiten mit 5 V oder 3.3 V.

7.2.4 Programmierschnittstelle

Der ATmega644 kann standardmässig über ISP oder JTAG programmiert werden. Optional könnte auch via Boot-Fuse ein eigener Bootloader angesprochen werden, damit liese sich beispielsweise eine Programmierung über UART realisieren. Der Einfachheit halber wird in diesem Projekt allerdings nur die ISP Schnittstelle verwendet.

Bei dem hier eingesetzten ISP Programmiergerät handelt es sich um einen AVRISP Mk2 Klon, welcher als STK500 in jeder gängigen DIE erkannt wird. Der Anschluss erfolgt direkt über USB.

7.2.5 Beschaltung

Der Mikrocontroller ist sehr anspruchslos wenn es um die minimale Beschaltung geht. Nötig ist lediglich der Anschluss der Versorgungsspannung und ein Pull-Up Widerstand am Reset (Reset ist intern invertiert).

Um UART nutzen zu können, ist eine externe Taktquelle erforderlich, da die interne zu ungenau ist. Daher ist die Schaltung mit einem externen 18.432 MHz Quarz erweitert, dies bringt zudem mehr Leistung gegenüber den Standard 8 MHz. Es handelt sich bei dem Quarz um einen Speziellen Baudratenquarz. Das bedeutet die Taktfrequenz lässt sich durch die Baudrate teilen. Folgende Rechnung verdeutlicht dies mit den in diesem Projekt verwendeten Grössen:

$$\frac{18432000 \text{ Hz}}{11520 \text{ Baud}} = 160$$

Formel 1 Quarzberechnung

7.3 Grafikdisplay



Abbildung 5 OLED Grafikdisplay

Um das Debuggen der Hardware selbst zu vereinfachen, viel schon früh in der Planungsphase der Entschluss, ein Grafikdisplay zu integrieren. Die Entscheidung **viel dabei auf ein 1" grosses Monochromes OLED Display mit 128x64 Pixel** Auflösung. Trotz der extrem geringen Grösse und der hohen Auflösung, ist es dank der OLED Technologie und dem damit verbundenen extrem hohen Kontrast noch gut lesbar.

Das Display ist bereits fest auf einem SSD1306 Treiber verlötet. Dieser ermöglicht das Ansteuern des Displays nicht nur über einen 8-bit Parallel Anschluss, sondern auch über einen I2C und einen SPI Bus. Die Stromversorgung ist mit 3.3V möglich, der Stromverbrauch liegt dabei bei maximal 20mA.

7.4 Funkmodul

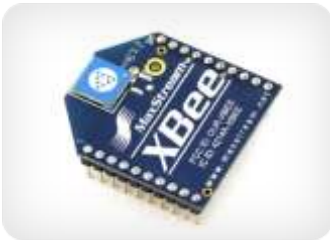


Abbildung 6 XBee Funkmodul

Die Daten des Sensorboards sollen über Funk zu einem Empfänger welcher am PC installiert ist, gesendet werden. Zur Auswahl stand hierbei entweder ein Bluetooth oder XBee Funkmodule.

Das Bluetooth Modul bietet hier den Vorteil, dass die meisten Computer bereits ein entsprechendes Gegenstück integriert haben, allerdings ist der Aufwand des Konfigurierens relativ gross und kompliziert. Das XBee Modul hingegen hat zwar eine viel tiefere maximale Datenübertragungsrate, ist dafür sehr einfach zu konfigurieren und auch als Host verwendbar. Die Auswahl viel daher auf das XBee Modul, besonders im Hinblick auf eine schnelle und reibungslose Inbetriebnahme. Das ausgewählte Modul hat 1mW Sendeleistung und eine Theoretische Reichweite von 100 Meter.

7.5 Schema

Das Schema stellt den kompletten elektrischen Aufbau der Schaltung dar. Die notwendigen Beschaltungen der Bauteile wurden den dazugehörigen Datenblättern entnommen. Erstellt wurde das Schema sowie der Aufbauplan und die Stückliste mit der Software Eagle 5.11.0.

Auf dem Board sind zwei unterschiedliche Spannungen. Als Eingangsspannung liegt zwischen 5-15 V an, diese wird für den Sensor und den Festspannungsregler benötigt. Der Festspannungsregler hat einen Output von 3.3 V. Diese Spannung wird für alle restlichen Komponenten verwendet.

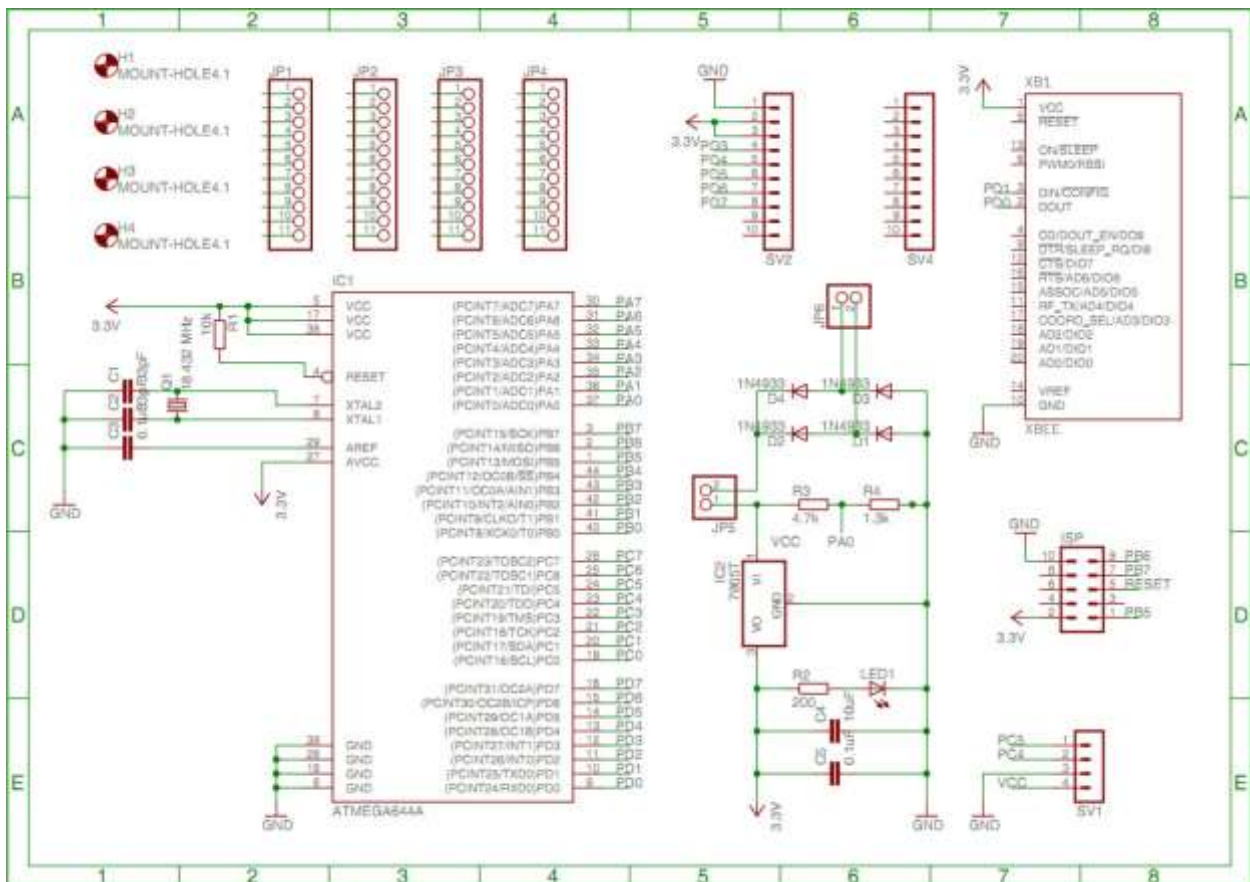


Abbildung 7 Hardwareschema

SV1: Steckplatz für den Sensor, stellt die Versorgungsspannung und I2C zur Verfügung.

SV2+SV4: Steckplatz für das Display, stellt 3.3V zur Verfügung und ist mit Steuerleitungen an PORT D verbunden.

JP1-JP4: Steckplatz für den Mikrocontroller.

JP5: Stecker für Ein-/Aus Schalter.

JP6: Stecker für Versorgungsspannung (z.B. Akku oder Netzteil). Dieser Anschluss ist verpolungssicher, daher ist keine Markierung für die Polung notwendig.

H1-H4: Bohrlöcher auf der Platine

IC2: 3.3V Festspannungsregler

D1-D4: Ein einfacher Gleichrichter, damit es zu keinem falschen Anschluss des Akkus kommen kann.

R3+R4: Spannungsteiler um die Eingangsspannung zu messen.

ISP: AVR Programmierschnittstelle.

LED1: Leuchtdiode für die Signalisierung des eingeschalteten Zustands.

7.6 Stückliste

Die Stückliste enthält alle zum Bau des Sensorboards nötigen Komponenten. Die Positionen sind jeweils auf dem Schema wieder zu finden.

Menge	Wert	Bauteile	Position	Preis in CHF
1		LED3MM	LED1	-
1		MA04-1	SV1	-
2		MA10-1	SV2, SV4	-
1		ML10	ISP	-
2		PINHD-1X2	JP5, JP6	-
4		PINHD-1X11	JP1, JP2, JP3, JP4	-
2	0.1uF	C-EU025-024X044	C3, C5	-
1	1.3k	C-EU050-025X075	R4	-
4	1N4933	1N4933	D1, D2, D3, D4	-
1	4.7k	C-EU050-025X075	R3	-
1	10k	R-EU_0207/7	R1	-
1	10uF	C-EU025-024X044	C4	-
1	18.432 MHz	CRYSTALHC49S	Q1	-
2	33pF	C-EU025-024X044	C1, C2	-
1	200	R-EU_0207/7	R2	-
1	3.3V	7805T	IC2	-
1	ATMEGA644A	ATMEGA644A	IC1	14.-
4		MOUNT-HOLE4.1	H1, H2, H3, H4	-
2		XBee	XB1	25.-
1		XBeeExplorer USB		20.-
1		Ein / Aus Schalter		-
1	500mAh / 7.2V	Akku LiPo		10.-
1		OLED Display		20.-
1		9DOF Stick		100.-
1	200x200mm	Leiterplatte		-
3		Schraube + Mutter		-
1		Gehäuse		10.-

Tabelle 5 Stückliste

Bei den grösseren Posten ist in der Stückliste jeweils der entsprechende Preis angegeben, die Kleinteile fallen jedoch nicht ins Gewicht. Die Gesamtkosten belaufen sich aufgerundet auf ca. 220 CHF.

7.7 Aufbau

Der Aufbau der Hardware ist so ausgelegt, dass er in einem Gehäuse mit den inneren Abmessungen 85 x 80mm Platz findet. Eine Aussparung ermöglicht das Unterbringen des Akkus direkt neben der Schaltung. Vier Bohrlöcher für die Montage im Gehäuse sind vorhanden.

Wegen des begrenzten Platzes sind einige Bauteile unterhalb anderer Elemente eingebaut. Unterhalb des Mikroprozessors befindet sich der Quarz und der Reset Widerstand, unter dem Display der Spannungsteiler und unter dem XBee eine Kontroll-LED mit Vorwiderstand.

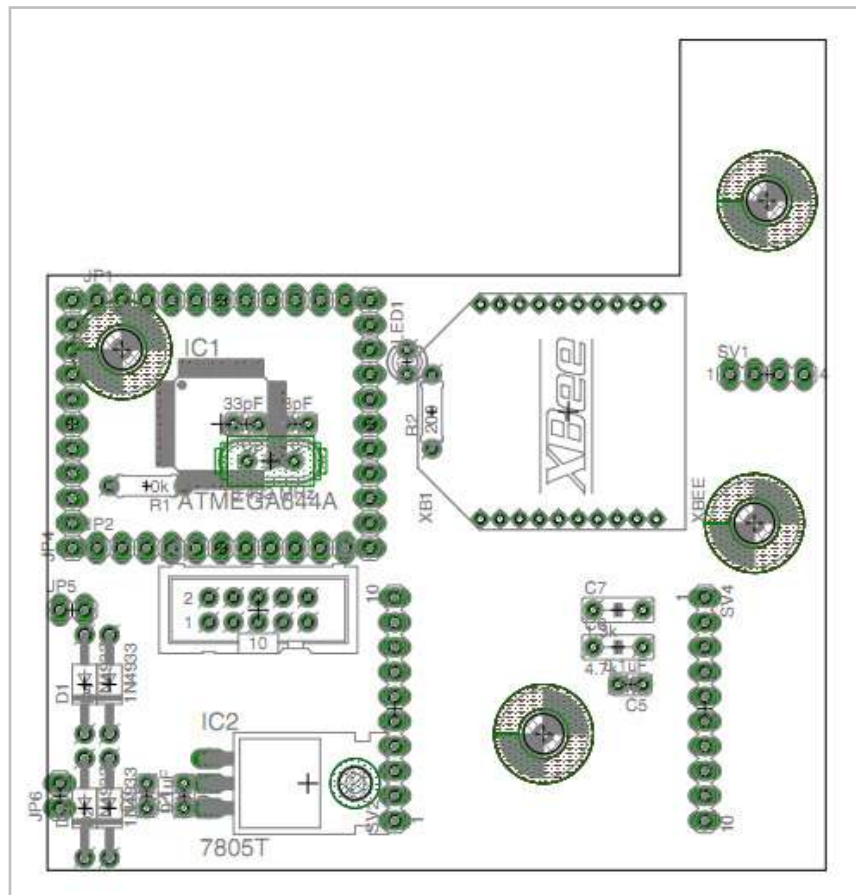


Abbildung 8 Aufbauplan der Hardware

7.8 Bauphase

Nach der Planung der Hardware und dem Eintreffen der bestellten Bauteile, folgte der Zusammenbau. Dieser ist hier mit einigen Bildern der Zwischenschritte dokumentiert.

Die Hauptplatine ist zugeschnitten und die Bohrlöcher werden gebohrt. Der AVR Sockel ist probenhalber bereits in Position.

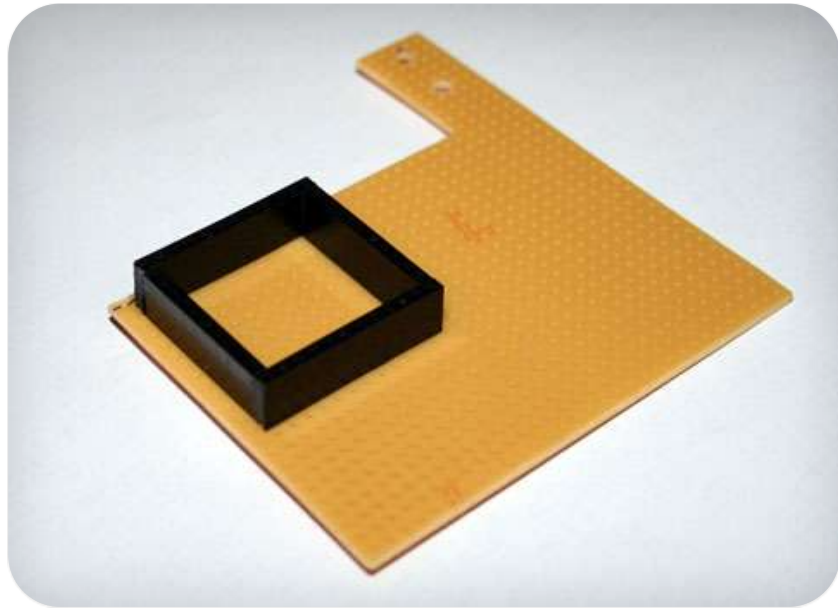


Abbildung 9 Hardware Grundplatine

Die Meisten Komponenten sind nun eingelötet. Der AVR kann bereits mit einer Testsoftware bespielt werden um die korrekte Verdrahtung zu prüfen. Nach und nach werden die einzelnen Komponenten dann getestet.

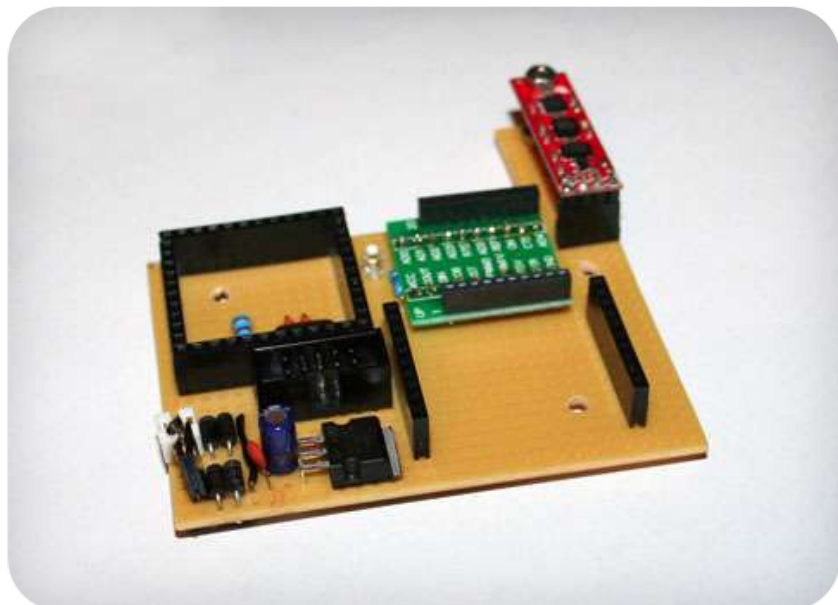


Abbildung 10 Hardwareaufbau Zwischenschritt

Die Hardware ist fertig aufgebaut. In der Front wurde zusätzlich ein Kippschalter integriert, welcher zum Ein- und Ausschalten dient. Die Platine findet optimal Platz in dem Gehäuse.



Abbildung 11 Die Hardware fertig aufgebaut

7.9 Inbetriebnahme

Die Inbetriebnahme verlief schrittweise. Zuerst wurde die Schaltung ohne die steckbaren Module unter Strom gestellt um zu prüfen, ob überall korrekte Spannungen anliegen. Danach sind nacheinander einzelne Module hinzugefügt und jeweils mit einer einfachen Testroutine getestet worden. Ausser einigen falsch angehängten Steuerleitungen an den Pins des Mikrocontrollers, waren keine Fehler vorhanden. Die Inbetriebnahme dauerte zwar etwas länger, verlief aber gut.

8 AVR Software

8.1 Funktionsprinzip

Die Software welche auf dem AVR eingesetzt wird verfolgt ein einfaches Prinzip, nämlich das der Hauptschleife, auch Mainloopgenannt. Das bedeutet, nach dem Initialisieren befindet sich das Programm in einer Endlosschleife welche bis zum Ende, also dem Abschalten des Prozessors, durchlaufen wird.

In der Hauptschleife wird abgefragt, ob Daten im UART Puffer liegen oder das Interrupt-Flag gesetzt ist. Je nach Fall wird eine entsprechende Funktion ausgeführt. Beim empfangen von UART Daten muss der entsprechende Befehl analysiert und beantwortet werden. Ist das Interrupt-Flag gesetzt, so wird ein Messdatensatz vorbereitet und zurückgeschickt.

Das Interrupt-Flag wird durch einen internen Timer gesetzt. Die Frequenz des Timers ist etwas höher eingestellt als die der Sensoren, so wird sichergestellt dass kein Datensatz verpasst wird.

8.2 Ablaufschema

Folgende Grafik zeigt den Ablauf des Programms.

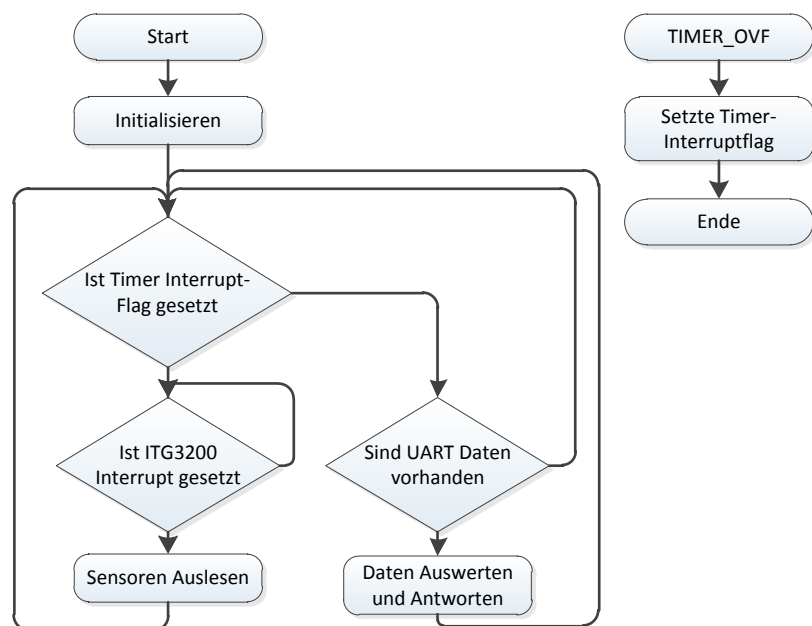


Abbildung 12 Ablaufschema

Beim einmaligen durchlaufen des Startvorgangs werden alle Komponenten initialisiert. Danach befindet sich das Programm in der Hauptschleife, in welcher wiederholt abgefragt wird, ob das Interrupt-Flag gesetzt ist oder Daten im UART Puffer vorhanden sind.

Parallel dazu läuft ein Timer, welcher beim Erreichen des Overflows den Interrupt TIMER_OVF auslöst, dieser wiederum setzt das Interrupt-Flag.

8.3 Aufbaudiagramm

Die Programmiersprache stellt leider keine Möglichkeit zur Verfügung mit Klassen zu arbeiten. Trotzdem sind die Methoden logisch in Gruppen unterteilt, welche in jeweilig benannten Files liegen, die sehr ähnlich denen von Klassen aufgebaut sind. Wenn viele Methoden in einem File vorhanden sind, so werden diese mit einem Präfix ergänzt (beispielsweise `i2c_writeRegister`) damit die Zugehörigkeit in anderen Files besser ersichtlich ist.

Das Aufbaudiagramm ist von weitem betrachtet sehr simpel, wie in der folgenden Grafik ersichtlich.

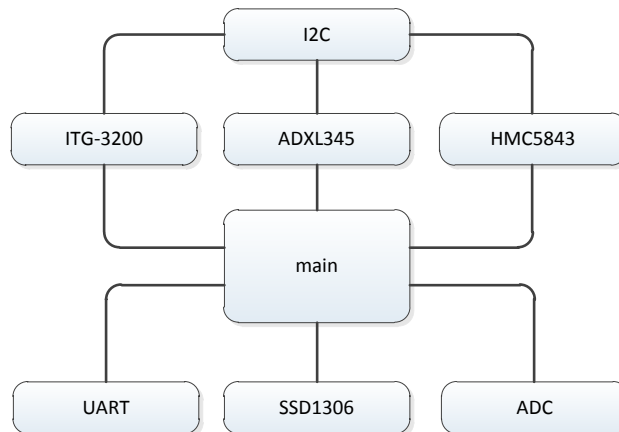


Abbildung 13 Aufbaudiagramm AVR Software

Da die Mainmethode den gesamten Ablauf steuert, sind dort alle Files eingebunden ausser die I2C Bibliothek. Dieses wird nur von den jeweiligen Sensorbibliotheken benötigt um die Sensoren anzusteuern.

8.4 Software

Entwickelt wird die Software mit AVR-Studio Version 4.15. Damit diese korrekt kompiliert werden kann, sind noch einige Einstellungen in der IDE notwendig. Ansonsten können die float Datentypen nicht über `printf` ausgegeben werden. Folgende Bibliotheken müssen daher eingebunden werden:

- `Libprintf_float.a`
- `Libm.a`

In den „Custom Options“ ist zudem die zusätzliche Compiler-Option „`-Wl,-u,vfprintf`“ notwendig.

8.4.1 I2C Treiber

Die Kommunikation mit den Sensoren erfolgt über die Hardware I2C Schnittstelle des AVR. Als Bibliothek kommt eine von Atmel selbst zur Verfügung gestellte zum Einsatz, welche die wichtigsten Methoden zur Kommunikation bereits beinhaltet. Für einen komfortablen Einsatz wurde diese von mit noch mit „`i2c_writeRegister`“, „`i2c_read8`“ und „`i2c_read16`“ ergänzt.

`I2c_writeRegister`: Schreibt ein Byte in das Register eines Busteilnehmers. Als Parameter muss neben dem Datensatz die Zieladresse und die Registeradresse mitgegeben werden.

`I2c_read8`: Liest ein Byte aus einem Register eines Busteilnehmers. Wird vor allem verwendet um die ein Byte grossen Konfigurationsregister auszulesen. Als Parameter werden die Zieladressen mitgegeben. Der Rückgabewert ist direkt der empfangene Datensatz.

I2c_read16: Da alle Sensoren 16 Bit grosse Werte liefern, wurde eine zusätzliche Methode integriert, die automatisch zwei nebeneinander liegende 8 Bit Register ausliest. Als Parameter werden die Zieladressen mitgegeben und noch ein zusätzliches Bit für die Auswahl der Byte Anordnung. Der Rückgabewert ist ein aus beiden Registern zusammengesetzter 2 Byte grosser Integer.

Eine Konfiguration des I2C Busses ist nahezu keine notwendig. Da die Schnittstelle Hardwareseitig integriert ist, sind die Pins bereits korrekt belegt. Die Taktrate wurde mit der Methode „i2cSetBitrate“ auf 350kHz erhöht. Theoretisch müsste 400kHz möglich sein, da alle Komponenten den „Fast Mode“ unterstützen. Aufgrund nicht genau feststellbarer Probleme, viel jedoch jeweils nach einigen Sekunden Busteilnehmer aus und waren nichtmehr erreichbar.

8.4.2 UART Treiber

Um die Hardware-UART Schnittstelle zu steuern, kommt eine oft verwendete Bibliothek von Peter Fleury zum Einsatz.¹ Diese verfügt über alle notwendigen Methoden um die Schnittstelle zu konfigurieren, Daten zusenden und zu empfangen. Die Auswertung der eintreffenden Daten erfolgt hier in der Hauptschleife, möglich wäre aber auch ein auswerten des Interrupts.

8.4.3 Display Treiber

Das Display wird von einem SSD1306 Treiberbaustein gesteuert. Die Bibliothek für diesen Treiber welche zurzeit erhältlich ist, beinhaltet nur sehr rudimentäre Funktionen und ist zudem in C++ geschrieben². Es war daher notwendig, diese in C zu übersetzen und Funktionen welche in den C Bibliotheken nicht vorhanden sind, möglichst identisch nachzustellen.

Die Konvertierung von C++ in C Code war in diesem Fall relativ simpel. grösstenteils genügte es, lediglich die Statischen Klassendefinitionen zu entfernen. Die Methoden „shiftOut“ und „digitalWrite“ waren allerdings aufgrund der fehlenden C++ Bibliotheken nicht vorhanden, deshalb mussten diese neu implementiert werden.

digitalWrite: Setzt einen Pin auf High oder Low. Als Parameter wird der Pin des Ports übergeben und der Ausgangspegel (entweder High oder Low). Die Umsetzung ist sehr einfach.

```
void ssd1306_digitalWrite(int pin, int level){
    if(level == HIGH)
        PORTD |= (1<<pin);
    else
        PORTD &= ~(1<<pin);
}
```

shiftOut: Schiebt ein Byte Seriell über einen Pin und gibt zusätzlich ein Clock Signal aus. Eine Verzögerung nach jedem Bit ist nicht notwendig. Als Parameter werden die Pins des Ports und das Byte selbst übergeben:

```
void ssd1306_shiftOut(int pinOut, int pinClock, uint8_t byte){
    for(int i = 8; i; i--){
        ssd1306_digitalWrite(pinOut,LOW);
        if(byte & 0x80)
            ssd1306_digitalWrite(pinOut,HIGH);

        byte += byte;
        ssd1306_digitalWrite(pinClock,LOW);
        ssd1306_digitalWrite(pinClock,HIGH);
    }
}
```

¹ Quelle: <http://homepage.hispeed.ch/peterfleury/avr-circuits.html>

² Originalreferenz auf CD (Software\Referenzen\ssd1306\)

Die restliche Bibliothek ist an sich relativ simpel. Es existiert ein 1kB grosser Zwischenspeicher in Form eines Arrays (1024 x 8 Bit) welches der Displayauflösung (64 x 128 Pixel) entspricht. Mit der Methode „`ssd1306_setpixel`“ wird direkt in diesen Zwischenspeicher geschrieben. Mit der Methode „`ssd1306_display`“ wird der Speicher über SPI zum Treiber übertragen. Die SPI Schnittstelle ist direkt in der Bibliothek integriert und greift der Einfachheit halber nicht auf die im AVR Integrierte Hardware SPI Schnittstelle zu. Es stehen zudem zusätzlich unterschiedliche Methoden zum Zeichnen von Kreisen, Linien und Rechtecken zur Verfügung.³

Benötigt werden für die gesamte Applikation drei unterschiedliche Anzeigen für verschiedene Programm Situationen. Beim Einschalten des Gerätes wird das HFU Logo als Bildschirmschoner geladen (Abbildung 14). Dieses liegt bereits beim Initialisieren als Vorladewert im Zwischenspeicher. Wird der Messvorgang gestartet, so wird die Anzeige mit den Statusinformationen der Kalibrierung aktualisiert (Abbildung 15). Sind alle drei Sensoren auf „ok“, war der Vorgang erfolgreich. Beim Anhalten des Messvorgangs wird dies ebenfalls auf dem Display angezeigt (Abbildung 16).



Abbildung 14 Anzeige des Startbildschirms



Abbildung 15 Anzeige bei Start des Messvorgangs



Abbildung 16 Anzeige beim Halt des Messvorgangs

8.4.4 Timer / Interrupts

Der Timer gibt in diesem Programm vor, in welchem Zeitintervall der Sensor abgefragt werden soll. Für die Berechnung der Zeit bis zum nächsten Überlauf des Timers wird folgende Formel verwendet:

$$\frac{\text{Taktfrequenz}}{\text{Prescaler} * 65536} = \text{Zeit in ms}$$

Formel 2 Timerfrequenz

Daraus ergibt sich ein Intervall von 35.2ms (Taktfrequenz = 18.432 MHz, Prescaler = 8). Bei einer Sensorfrequenz von 25Hz bleiben daher jeweils noch fast 5ms Reserve. Diese werden bei der Interrupt Abfrage des ITG-3200 verwendet, um die Werte im optimalen Moment auszulesen.

8.4.5 Magnetometer

Der HMC5843 enthält zwei Register, für den Ausgabemodus und für das Setzen der Skalierung. Beide werden beim Initialisieren in den Ausgangszustand gebracht.⁴

Die Kalibrierung des Sensors wird wie im Datenblatt beschrieben vorgenommen. Dabei wird ein Skalierungsfaktor ausgerechnet, welcher danach beim Auslesen der Werte multipliziert wird.

³ Spezifikationen sind im Datenblatt SSD1306 ersichtlich

⁴ Register sind dem Datenblatt HMC5843 entnommen

8.4.6 Gyro

Der ITG-3200 enthält viele unterschiedliche Register zur Konfiguration. In der Initialisierungsphase werden jeweils die Abfragefrequenz, Filter, Skalierung, Interrupts und das Powermanagement eingestellt.⁵

Bei der Kalibrierungsphase handelt es sich um eine Ermittlung des Offsets im Ruhezustand. Dafür werden nacheinander mehrere Messwerte aufaddiert, bei welchen dann schlussendlich der arithmetische Mittelwert als Offset definiert wird.

Beim Abfragen der Sensordaten ist das Vorgehen anders als bei den restlichen Sensoren. Die Daten werden nicht sofort abgerufen, denn diese sind eventuell noch nicht aktualisiert worden und es könnten Messwerte verloren gehen. Daher wird zuerst in einer Schleife das Interrupt Register abgefragt, bis sicher neue Werte vorliegen welche dann ausgelesen werden.

Für das Abfragen des Temperatursensors auf dem ITG-3200 ist eine zusätzliche Methode integriert. Diese gibt direkt den Wert des Sensors als 2 Byte langen Integer zurück.

8.4.7 Accelerometer

Der ADXL345 bietet extrem viele Einstellungsmöglichkeiten. Für die Verwendung in diesem Projekt sind allerdings nur Änderungen im Register für Powermanagement, Mode, Abfragefrequenz und der Skalierung notwendig. Diese werden beim Initialisieren gesetzt.⁶

Da die Ausgabe etwas verrauscht ist, wird beim Abfragen der Werte ein simpler Algorithmus zur Glättung angewandt. Dabei wird jeweils der arithmetische Mittelwert aus den letzten drei Messungen ausgerechnet und zurückgegeben.

8.4.8 Initialisierung

Bei jedem Start muss der AVR und alle externen Komponenten neu initialisiert werden, da immer im stromlosen Zustand alle relevanten Konfigurationen verloren gehen. Dafür werden jeweils sequenziell die folgenden Punkte bei der Initialisierung durchgegangen.

- Konfiguration des 16 Bit Timers
- Definieren der I/Os als Ein- oder Ausgänge, optionales setzen der internen Pull-Up Widerstände
- Konfiguration des internen AD Wandlers
- I2C und UART Initialisieren
- Display Initialisieren und hinterlegtes Startbild laden
- Sensoren Initialisieren
- Interrupts aktivieren
- Begrüssung senden (Ping)

8.4.9 Hauptschleife

Nach dem Initialisieren befindet sich das Programm in der Hauptschleife. Diese wird bis zum Ende der Laufzeit immer wieder durchlaufen. Die Hauptschleife besteht aus zwei Hauptteilen.

Im ersten Teil wird abgefragt ob das Interrupt Flag welches durch den Timer gesetzt wird, aktiv ist. Wenn ja, werden alle Sensordaten nacheinander abgefragt, formatiert und über UART ausgegeben.

Im zweiten Teil wird überprüft, ob Daten im UART Puffer vorhanden sind. Wenn ja, wird der gesamte String bis zum Zeilenumbruch in einer Schleife eingelesen. Der String wird dann analysiert und es wird entsprechend reagiert.

⁵ Register sind dem Datenblatt ITG-3200 entnommen

⁶ Register sind dem Datenblatt ADXL345 entnommen

8.5 Konfiguration XBee

Beide XBee Module müssen für den Betrieb einmalig konfiguriert werden. Dies ist entweder direkt über die serielle Schnittstelle mittels AT-Befehlen möglich, oder über ein vom Hersteller zur Verfügung gestellten Tool namens X-CTU.⁷

Damit beide XBee miteinander kommunizieren können, müssen sie sich im selben Netzwerk befinden. Dafür wurden die PAN IDs identisch auf 3141 eingestellt. Nun fehlt noch die jeweils korrekte eigene Adresse (MY – 16 bit Source Address) und die Verbindungsadresse (DL – Destination Address Low) des jeweils anderen XBee.

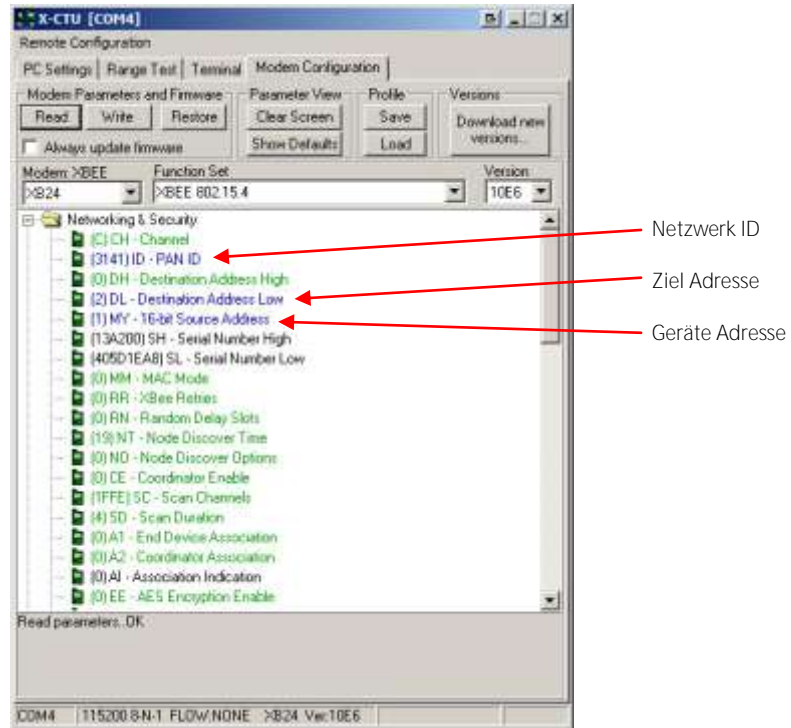


Abbildung 17 Tool X-CTU

⁷ Informationen sind im Datenblatt X-CTU ersichtlich

9 PC Software

9.1 Funktionsprinzip

Die Software welche auf dem Computer läuft, hat hauptsächlich die Aufgabe die Daten zu empfangen und auszuwerten. Für die Steuerung sind Nebenbei noch einige rudimentäre Funktionen zum Ein- und Ausschalten und Abfragen von Statuswerten vorhanden.

Ein laufender SerialPort Thread ist für das empfangen der Daten zuständig. Wird ein String empfangen, so wird dieser zerlegt und ausgewertet. Schrittweise werden die Daten nun zurück zum Frontend übertragen und immer weiter aufbereitet. Schlussendlich sind saubere, gefilterte und in Einheiten umgerechnete Werte ersichtlich.

9.2 Ablaufschema

Beim Programmstart werden zuerst die benötigten Objekte instanziiert und soweit nötig vorbereitet. Der restliche Vorgang ist eventgesteuert. Entweder wird durch drücken eines Buttons oder Empfang von Daten am SerialPort ein Event ausgelöst.

Zu beachten ist das der SerialPort in einem anderen Thread läuft und dadurch der Event ebenfalls. Wen durch ein Event, Controls verändert werden, müssen diese gesondert behandelt werden.

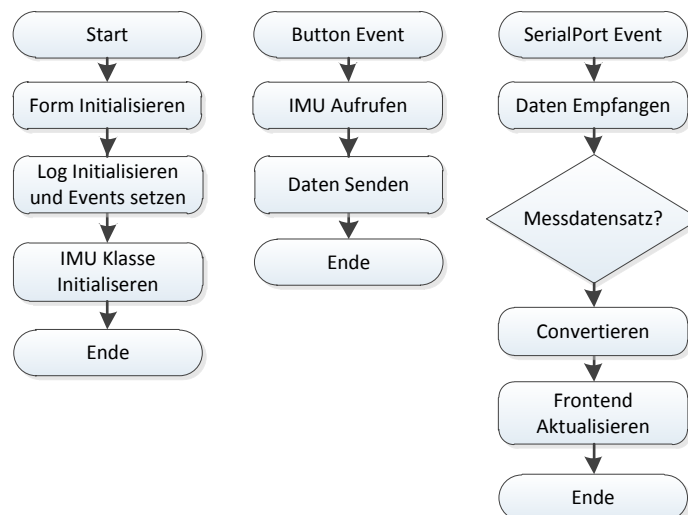


Abbildung 18 Ablaufschema

9.3 Klassendiagramm

Folgendes Diagramm stellt vereinfacht die Zusammengehörigkeit der einzelnen Klassen dar.

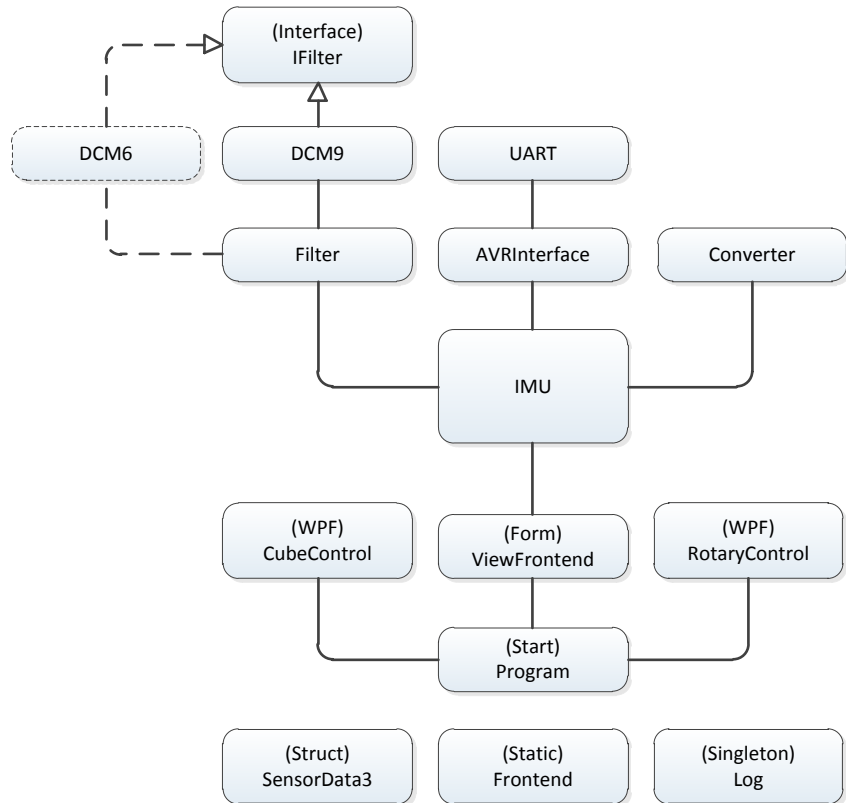


Abbildung 19 Klassendiagramm

Nicht verbundene Objekte werden entweder Statisch verwendet oder dienen als Struktur. Einfache Linien stellen Verbindungen zwischen Objekten dar, beispielsweise wenn sie andere Klassen instanzieren. Pfeile hingegen symbolisieren eine Vererbung oder Implementierung eines Interfaces.

9.4 Software

9.4.1 UART

Die Klasse „UART“ steuert und überwacht die Klasse System.UI.Ports.SerialPort aus dem .NET Framework. Hauptsächlich geht es darum, vor dem Senden, Empfangen, Verbinden oder beim Schliessen der Verbindung zu prüfen, ob sich der SerialPort in einem gültigen Zustand befindet welcher die Operation auch zulässt. Ansonsten werden Exceptiones mit der entsprechenden Fehlermeldung zurückgeworfen.

In diesem Speziellen Fall ist die „Send“ Methode leicht modifiziert, da nach dem Senden des ersten Zeichens noch eine kurze Pause vorhanden ist. Diese ist nötig um sicherzustellen, dass der AVR welche die Daten empfängt, sich nicht in einem Interrupt aufhält.

9.4.2 AVRInterface

Als Interface für den in diesem Projekt verwendeten AVR dient die Klasse „AVRInterface“. Sie beinhaltet zum einen die Kommandos welcher der AVR versteht, zum anderen decodiert er die vom AVR eintreffenden Datensätze und wirft entsprechende Events zurück.

Im Ansatz ist ein Mechanismus integriert welcher erkennt, ob das Sensorboard eingeschaltet ist. Dies ist allerdings in dieser Hinsicht problematisch, da das Sensorboard sich beim ausschalten nicht Abmelden kann und noch keine Fehlermeldung vorhanden ist, wenn ein Kommando nicht zugestellt werden kann.

9.4.3 Converter

Die Klasse „Converter“ stellt Methoden zur Umrechnung unterschiedlicher Daten zur Verfügung, im Vordergrund stehen allerdings die Rohdaten der Sensoren. Diese kommen direkt vom Sensorboard und werden hier in die gewünschte Einheit umgerechnet.

Zur Verwendung muss die Klasse instanziiert werden. Der Grund dafür liegt darin, dass später beispielsweise Temperaturwerte für die Berechnung des Gyro-Offsets individuell vergeben werden.

Temperatur: Für die Umrechnung des Bitwertes in Grad Celsius wird eine einfache Formel eingesetzt, welche sich aus dem Datenblatt ableiten lässt. Der Sensorwert wird mit 280 LSB/°C und einem Offset von -13200 bei 35° umgerechnet. Daraus ergibt sich die folgende Formel:⁸

$$\text{Temperatur } ^\circ = \frac{\text{input} + 13200}{280} + 35$$

Formel 3 Temperaturberechnung

Akkuspannung: Um die aktuelle Spannung welche am Analog/Digital Wandler anliegt zu berechnen, wird der erhaltene Bit Wert mit 1024 LSB/3.3V umgerechnet. Der Spannungsteiler wird mit den exakt gemessenen Werten 1.29kOhm und 4.63kOhm ebenfalls mit eingerechnet. Zudem werden die beiden Dioden mit jeweils 0.7V dazu addiert. Daraus ergibt sich die folgende Formel.

$$\text{Spannung}V = \frac{3.3}{1024} * \frac{\text{input}}{1.29} * 5.92 + 1.4$$

Formel 4 Spannungsberechnung

9.4.4 Filter

Die Klasse „Filter“ stellt ein Gerüst zur Aufnahme eines Filters welches das Interface IFilter implementiert zur Verfügung. An den geladenen Filter werden die bereits umgerechneten Werte des Gyro-, Magnetometer- und Accelerometers übergeben. Der Output ist ein Quaternion, dieses wird in unterschiedliche Rotationsformate umgerechnet welche als Event zurückgegeben werden.

Dass Quaternion wird mit den folgenden Formeln in Eulerwinkel umgerechnet:⁹

$$ex = \text{atan2}(2(YZ + XW), 1 - 2(X^2 + Y^2))$$

$$ey = \text{asin}(2(YW - XZ))$$

$$ez = ey = ey = \text{asin}(2(YW - XZ))$$

Formel 5 Quaternion zu Euler

⁸ Datenblatt ITG-3200

⁹ <http://de.wikipedia.org/wiki/Quaternion>

Interessanter für die meisten Anwendungen ist aber die Umrechnung in Yaw, Roll und Pitch. Folgende Formeln rechnen das Quaternion direkt um:

$$yaw = \text{atan2}(2(YZ + XW), 1 - 2(X^2 + Y^2))$$

$$roll = \text{atan}\left(\frac{2(XY + ZW)}{\sqrt{4(Y^2W^2 - X^2Z^2) + (X^2 - Y^2 - Z^2 + W^2)^2}}\right)$$

$$pitch = \text{atan}\left(\frac{2(YW - XZ)}{\sqrt{4(Y^2W^2 + X^2Z^2) + (X^2 - Y^2 - Z^2 + W^2)^2}}\right)$$

Formel 6 Quaternion zu Yaw, Roll, Pitch

9.4.5 DCM Filter

Um die Klasse „Filter“ und die Funktionsweise der darauf aufbauenden Funktionen zu testen, sind zwei DCM Filter vorhanden. Sie basieren auf einer Vorlage und wurden für diesen Zweck von C in C# übersetzt.¹⁰

„DCM9“ ist ausgelegt auf die Verarbeitung aller drei Sensoren, „DCM6“ hingegen verzichtet auf den Magnetometer. Die Zuordnung des gewünschten Filters kann in dem Konstruktor der Klasse „Filter“ angepasst werden.

9.4.6 SensorData3

Da es sich in diesem Projekt hauptsächlich um Datensätze mit jeweils einer X, Y und Z Komponente handelt, existiert dafür ein zusätzlicher Datentyp. Es hätte grundsätzlich auch der bereits vorhandene „Vector3D“ verwendet werden können, allerdings stellt dieser nicht die benötigten mathematischen Funktionen zur Verfügung.

Die Struktur ist sehr einfach aufgebaut. Sie enthält drei double Properties Y, X und Y welche direkt oder über einen Index aufgerufen und gesetzt werden können. Im Konstruktor können zudem beim Initialisieren bereits Werte vorgeladen werden. Die mathematischen Funktionen welche direkt angewendet werden können, sind sehr vielfältig. So lässt sich z.B. `SensorData(3,2,1) * 3` rechnen, wobei hier alle Werte mit dem Faktor multipliziert werden. Fehlende Operationen können später ohne grossen Aufwand ergänzt werden.

9.4.7 Log

Die Klasse Log stellt einfachste Methoden zur Verfügung um Vorgänge zu Loggen. Sie kann instanziiert werden, enthält aber zusätzlich gemäss dem Singleton Pattern zwei statische Methoden „InstanceError“ und „InstanceUart“, welche jeweils eine statisch gespeicherte Instanz der Klasse enthalten. Dadurch können alle Klassen auf denselben Logschreiben, zugreifen oder über einen Event benachrichtigt werden.

9.4.8 IMU

Für das ganze Zusammenspiel von Sensorbord, Filter und Converter ist die Klasse „IMU“ zuständig. Sie instanziiert die nötigen Klassen, leitet die Kommandos weiter an das AVRInterface, schickt die Eintreffenden Datensätze durch den Filter und wirft entsprechende Events bei Veränderungen zurück um beispielsweise die Diagramme zu aktualisieren.

¹⁰ Originalreferenz auf CD (Software\Referenzen\dcm\),
Quelle: <http://diydrones.ning.com/profiles/blogs/dcm-imu-theory-first-draft>

9.4.9 Frontend

Die Klasse Frontend stellt statische Methoden zur Verfügung um die unterschiedlichen Controls welche auf dem Hauptform vorhanden sind, aus einem Thread heraus zu manipulieren. Dies ist nötig, da der SerialPort in einem unabhängigen Thread läuft, aus welchem der Zugriff aufgrund Threadübergreifender Operationen sonst nicht möglich wäre.

UpdateChart: Fügt jeweils am Ende des Diagramms einen weiteren Messwert ein. Ist eine bestimmte Anzahl Messwerte erreicht, so werden die vorhandenen Werte jeweils um eins nach hinten gerückt.

UpdateLogTextbox: Fügt jeweils eine weitere Zeile Text ein. Zusätzlich ist eine Funktion vorhanden, welche die Maximale Länge des Textes in der TextBox begrenzt.

UpdateRotaryControl / UpdateCubeControl: Ruft die jeweils zum Control gehörende „Update“ Methode auf und übergibt ihr die neuen Werte.

UpdateLabel: Aktualisiert den Text eines beliebigen Labels.

9.5 Controls

Damit die gefilterten Werte ansprechend visualisiert werden können, sind spezielle Controls notwendig, die unterschiedliche Drehbewegungen und Schräglagen darstellen können. In der Standard .NET Umgebung sind leider keine Steuerelemente vorhanden, welche auch nur annähernd diesen Anforderungen entsprechen. Daher wurden zwei Controls speziell für diese Aufgabe entwickelt.

Die Controls wurden beide in WPF realisiert, da sie grafisch aufwändige Elemente enthalten, welche dort einfacher umzusetzen sind. Schlussendlich können sie über ein „ElementHost“ in das Windows Form integriert werden.

9.5.1 RotaryControl

Mit diesem Control kann eine Rotation von 0-360° visualisiert werden. Ein Horizontaler Balken zeigt die aktuelle Schräglage an. In der Mitte wird der Wert zusätzlich noch als Text ausgegeben.

Realisiert ist dies mit einem Hintergrund Bild (Kreis) und einem Vordergrund Bild (Balken), welches sich je nach übergebenem Winkel dreht. Bei dem mittig angezeigten Text handelt es sich um ein simples Label.

Mit der Methode „Update“ kann ein Wert in Grad gesetzt werden, welche dann auf dem Control visualisiert wird. Ist zusätzlich für die Verwendung ein Offset erforderlich, so kann dies jederzeit über das Property „Offset“ gesetzt werden (Dieses Offset wird nur auf die Grafik angewandt, nicht auf den Text welcher im Label angezeigt wird). Der Text wird automatisch formatiert, so dass keine nachkommastellen aber ein „°“ Zeichen angezeigt wird.

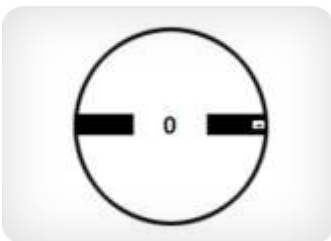


Abbildung 20 RotaryControl

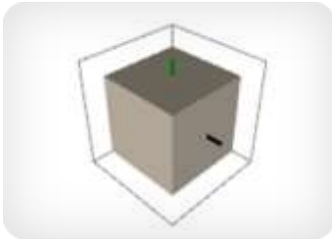


Abbildung 21 CubeControl

9.5.2 CubeControl

Das „CubeControl“ kann die Rotation um alle drei Achsen anhand eines 3D Würfels darstellen. Für die Definierung der Ausgangslage, ist im Hintergrund ein Raster eingezeichnet. Der Würfel enthält zur Orientierung verschiedenfarbige Richtungsmerkmale (Rot = Norden, Schwarz = Süden, Grün = Oben).

Realisiert ist dies in einem Viewport3D mithilfe von XAML. Folgende Elemente sind vorhanden:

- Würfel
- Orientierungs-Linien
- Hintergrundgitter
- Lichtquelle
- Kamera

Zum rotieren des Würfels sowie der Orientierungs-Linien, sind diese mit einer individuelle ID versehen, über welche sie angesteuert werden.

Linien können standartmässig nicht dargestellt werden. Dafür wird eine externe DLL „3DTools“ benötigt. Diese ermöglichen das Erstellen von geraden Linien mithilfe definierter Koordinaten, der Farbe sowie der Liniendicke.¹¹

Aktualisiert wird das Control mit der Methode „Update“. Als Parameter wird ein Quaternion erwartet. Dieses wird intern auf einen Vektor und einen Winkel aufgeteilt.

9.6 Oberfläche

Die Oberfläche ist in unterschiedliche Bereiche aufgeteilt. Auf der rechten Seite befinden sich die Diagramme aller drei Sensoren mit den jeweiligen Achsen X,Y und Z. Links sind in vier Bereiche unterteilt, weitere Anzeigeelemente und Steuerfunktionen integriert.

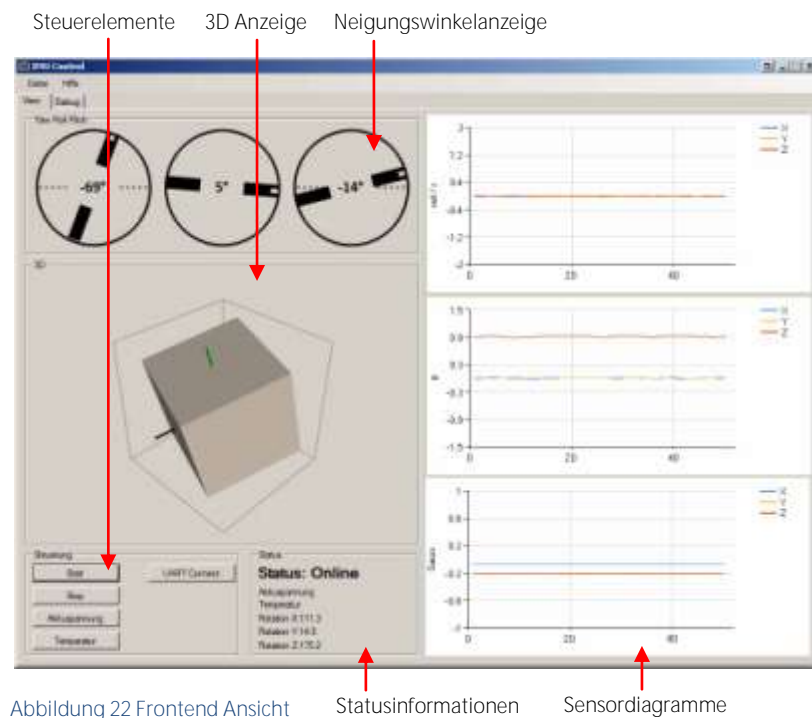


Abbildung 22 Frontend Ansicht

Statusinformationen

Sensordiagramme

¹¹ Informationen unter <http://3dtools.codeplex.com/>

Um das Debuggen zu erleichtern, sind in einem zweiten Tab noch unterschiedliche Logfenster vorhanden. Neben System-Logs werden auch alle Transaktionen des SerialPorts aufgezeichnet oder können in Echtzeit verfolgt werden.

SerialPort Aufzeichnung

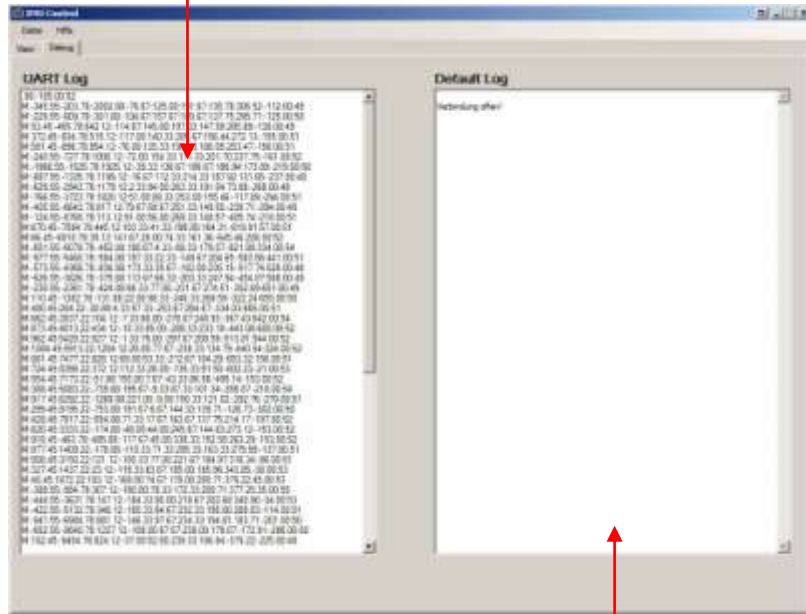


Abbildung 23 Frontend Debug Ansicht

System Logs

10 Tests und Auswertung

10.1 Software

Auf die bekannten Unit-Tests wurde in diesem Projekt verzichtet. Die meisten Funktionen sind stark von externen Inputs abhängig, die restlichen stellen lediglich einfach Event-Abfolgen dar. Der Nutzen wäre daher minimal und der Aufwand sehr gross gewesen.

Die Grundfunktionen sind daher manuell getestet. Dafür wird die Software gestartet und das Sensorboard ebenfalls. Der Status sollte sich nun auf „Online“ ändern, wenn das Gerät sich angemeldet hat. **Mit dem Button „Start“ wird der Messvorgang gestartet.** Die Werte müssen dann kontinuierlich eintreffen und die Oberfläche wird mit etwa 25fps aktualisiert (flüssig). Auf Bewegungen des Sensorboards wird sofort und ohne merkliche Zeitverzögerung reagiert.

Die Folgenden Tests wurden durchgeführt und Protokolliert.

Test	Status
Programm Startet ohne Fehlermeldung.	Ok
Kommunikationsaufbau zum Sensorboard ist erfolgreich.	Ok
Daten werden empfangen und Anzeigt.	Ok
Die Datenübertragung ist Flüssig.	Ok
Die Temperatur kann über Knopfdruck ausgelesen werden.	Ok
Die Akkuspannung kann über Knopfdruck ausgelesen werden.	Ok
Messvorgang kann gestartet und gestoppt werden.	Ok
Die Messwerte sind korrekt zugeordnet und umgerechnet.	Ok
COM-Anschluss kann manuell initialisiert werden.	Ok
Das Programm kann ohne Fehlermeldungen beendet werden.	Ok

Tabelle 6 Softwaretests

10.2 Messungen

Zur Messung verwendete Hilfsmittel:

- Ebene Fläche (Gemessen mit Wasserwage)
- Winkel 45° (Gemessen mit Winkellehre)
- Kompass

10.2.1 Filtertest

Die erste Testreihe befasste sich mit der Auswertung der Yaw, Roll und Pitch Resultate. Das Sensorboard wurde dafür jeweils auf eine Achse ausgerichtet, sehr langsam um 360° gedreht. Die Werte während der Bewegung sind dabei aufgezeichnet worden. Der Schräglage Winkel wurde dabei jeweils auf 0°, 45° und 180° variiert (Gerade, Schräg und auf dem Kopf).

	X 0°	X 180°	X 45°	Y 0°	Y 180°	Y 45°	Z 0°	Z 180°	Z 45°
Yaw	+ - 2°	+ - 2°	+ - 6°	+ - 3°	+ - 3°	+ - 4°	+ - 2°	+ - 3°	+ - 4°
Roll	+ - 3°	+ - 2°	+ - 5	+ - 2°	+ - 4°	+ - 6°	+ - 4°	+ - 3°	+ - 5°
Pitch	+ - 2°	+ - 3°	+ - 7°	+ - 3°	+ - 5°	+ - 5°	+ - 3°	+ - 4°	+ - 3°

Tabelle 7 Sensor Messungen

Die Messungen ergaben dabei eine Abweichung von jeweils zwischen maximal +- 5° in Ebener und maximal +- 7° in Schräger Lage. Diese doch relativ starken Abweichungen ergeben sich wahrscheinlich durch den nicht genullten Magnetfeldsensor.

10.2.2 Sensortest

Das testen der Einzelnen Sensoren auf die korrekte Kalibrierung ist relativ schwierig. Grundlegende Messfehler ergeben sich bereits bei der nicht hundertprozentig genauen Montage. Weitere Fehler entstehen durch die Gegebenheiten der Sensoren selbst.

Accelerometer: Dieser Sensor lässt sich relativ gut testen. Auf einer ebenen Fläche muss die Beschleunigung der jeweils nach unten zeigenden Achse mit 1G beschleunigt werden, da die Erdbeschleunigung auf den Sensor wirkt. Um 180° gedreht müsste er mit -1G beschleunigt werden. Die Auswertung ergab aufgrund dieser Tatsache folgende Resultate:

- X: + 0.9g / -1.05g
- Y: + 1g / - 1g
- Z: + 1.1g / - 0.95g

Die Messwerte sind nur minimal ungenau, mit einem Offset würde sich dies noch auf ein Maximum verbessern lassen.

Gyro: Der Gyro lässt sich sehr schwer auf die Richtigkeit testen. Daher wurde nur die Korrektheit der Achsenausrichtung ausgewertet. Diese sind alle Einwandfrei.

Magnetometer: Der Magnetometer lässt sich ebenfalls nur sehr schwer testen. Das Magnetfeld ist stark von der Position abhängig und äussere Einflüsse wirken sich relativ stark aus. Die Maximalen und Minimalausschläge bei der Testreihe ergaben folgende Resultate.

- X: +0.45 / -0.5
- Y: + 0.6 / -0.2
- Z: +0.4 / -0.5

Die Messwerte sind sehr stark verzogen. Ein Offset und eine zusätzliche Skalierung je Achse ist für einen späteren Einsatz zwingend notwendig

10.2.3 Spannungsmessung

Die Spannungsmessung wurde mithilfe eines regelbaren Netzgerätes und einem Multimeter überprüft. Mehrere Messungen im Bereich zwischen 5 und 15 V ergaben eine Toleranz von maximal +/- 0.2 V.

10.2.4 Temperaturmessung

Die Messergebnisse des Temperatursensors sind schwer zu überprüfen. Es wird nicht direkt die Umgebungstemperatur, sondern die Chip-Temperatur gemessen. Diese pendelt sich während des Betriebs bei etwa 30° ein, was ich für realistische Werte halte.

10.2.5 Verbrauchsmessung

Der Stromverbrauch des Sensorboards wurde mithilfe eines stationären Netzteils und einem Multimeter ermittelt. Die Stromaufnahme während des normalen Betriebs beträgt etwa 50mA bei 7.5V. Bei einem 500mA grossen Akku ergibt sich dadurch eine theoretische maximale Laufzeit von 10 Stunden.

10.3 Auswertung

10.3.1 Messresultate

Die Tests ergaben dass die Software grundsätzlich funktioniert und die Messwerte korrekt angezeigt werden. Die Messresultate des Magnetometers zeigen jedoch, dass dieser unbedingt für spätere Projekte noch genauer kalibriert werden müssen, da sie extrem stark verzerrt sind und den Filter dadurch negativ beeinflussen. Dies lässt sich auch durch das ständige, langsame nachkorrigieren der Position nach einer Bewegung beobachten, da der Sensor mit schlechten Werten den Filter beeinflusst.

10.3.2 Funktionsumfang

Der grösste Teil des Funktionsumfangs welcher zu Beginn im Pflichtenheft definiert wurde, konnte erfüllt werden. Zusätzlich wurden einige Funktionen die als Erweiterungen geplant waren bereits realisiert. Folgende Tabelle zeigt die Resultate:

Funktion	Ziel
Funktionsfähiges Sensorboard	Erreicht
Funktionierende Software zum Auswerten der Daten	Erreicht
Gemessen wird die Beschleunigung, Drehbewegung und das Magnetfeld	Teilweise Erreicht
Die Daten werden möglichst in Echtzeit am Computer dargestellt	Erreicht
Projekt Dokumentiert	Erreicht
Filter für die Aufbereitung der Daten	Teilweise Erreicht
Oberfläche erweitern für gefilterte Daten	Erreicht
Testplattform für die Demonstration	Nicht Erreicht

Tabelle 8 Auswertung Funktionsumfang

Da der Magnetometer noch unzureichend genau arbeitet, wurde der entsprechende Punkt nur mit „Teilweise Erreicht“ gekennzeichnet. Gleiches gilt für das Aufbereiten der Daten in einem Filter. Dieser muss noch besser ausgearbeitet werden. Für den Aufbau einer Testplattform war leider keine Zeit mehr vorhanden, daher „Nicht Erreicht“. Dies war allerdings nur ein optionaler Punkt.

11 Rückblick und Schlusswort

11.1 Projekt

Das Projekt befindet sich nun auf einem guten Stand und ich bin froh, dass sich schlussendlich alles etwa so ergeben hat wie ich es mir ursprünglich vorgestellt hatte. Die Materie war zwar anspruchsvoller als zu Beginn gedacht und dies hat sich auch deutlich in der Zeitplanung wiedergespiegelt, dennoch hat es schlussendlich funktioniert.

11.2 Hardware

Der Aufbau der Hardware zu Beginn des Projektes hat mich sehr motiviert, da er aus sehr vielen interessanten und mir teilweise auch noch nicht bekannten Teilen bestand. Die einzelnen Komponenten funktionierten zum Glück alle mehr oder weniger wie erwartet und haben auch später nie Probleme verursacht. Auch optisch ist der Aufbau trotz der beschränkten Mittel ansprechend und für Demonstrationszwecke mehr als ausreichend.

11.3 Software

Das Programmieren der Software hat länger gedauert und mehr Probleme verursacht als zuerst gedacht. Besonders die Software welche auf dem AVR läuft, stellte mich vor viele Fragen, welche ich teilweise nur mit stundenlangem experimentieren lösen konnte. Ich bin daher stolz, dass ich diese nun doch noch in einen stabilen Zustand bringen konnte und extrem viel dabei gelernt habe. Das Programmieren in C# dagegen war bereits viel angenehmer und ging auch reibungsloser voran, das Programm war aber auch weitaus weniger komplex.

11.4 Dokumentation

Das Erstellen der Dokumentation hat mich sehr viel Nerven und Motivation gekostet. Ich verwende relativ viele Funktionen von Word 2007, jedoch sind diese teilweise noch nicht vollkommen ausgereift oder dokumentiert? Von dem Arbeiten mit CSS in Verbindung mit HTML bin ich mir weitaus höherer Formatierungsmöglichkeiten und Freiheiten gewohnt. Dies hier stellt mich daher teilweise vor grossen Herausforderungen welche oft auch nicht sauber bewältigt werden konnten.

11.5 Zukünftiges

Während dem Arbeiten an diesem Projekt habe ich mir einige Notizen gemacht, was an diesem Projekt noch zu ändern, verbessern oder zu ergänzen ist. Sicher werden einige dieser Ideen zu einem späteren Zeitpunkt noch umgesetzt.

- Messvorgang der AVR Software abändern. Timer läuft ständig, ein Enable-Flag steuert die Messung. Der Timer könnte dadurch noch für andere Aufgaben verwendet werden.
- Auswählbarer COM-Port in den Einstellungen. Der Anschluss kann in den Einstellungen verändert werden und ist nichtmehr fest im Programm integriert.
- Festspannungsregler austauschen gegen einen Digitalen DC/DC Regler. Weniger Verlustleistung und Wärmeentwicklung.

- Integrieren eines UART-Bootloaders. Der AVR könnte damit direkt über Funk und der eigenen Software neu programmiert werden. Es wäre keine Programmierhardware mehr notwendig.
- Kalibrierungswerte im EEPROM abspeichern. Die Kalibrierung wäre auch nach einem Neustart noch vorhanden.
- Kalibrierungsmethoden verbessern, dadurch erhöhte Messgenauigkeit.
- Möglichkeit zur Aufzeichnung und Speicherung der Messdaten.
- Test eines Kalman-Filters, dadurch werden eventuell noch bessere Ergebnisse erzielt als mit dem DCM Filter.
- Datensätze welche zur Kommunikation verwendet werden, nichtmehr als String, sondern als Bytearray übertragen. Dadurch würde die Auslastung um etwa 70 % zurückgehen.
- Kommunikationsprotokoll mit einer Checksumme erweitern. Dadurch könnten Übertragungsfehler verhindert werden.
- Kommunikationsprotokoll erweitern, so dass auf eine Anfrage zwingend die Antwort folgen muss. Dadurch erhöhte Sicherheit und Möglichkeiten zur Überprüfung ob das Sensorboard Online ist.

12 Abkürzungen / Erklärungen

IMU	Inertial Measurement Unit
DCM	Direction-Cosine-Matrix, einfacher Filter zum Auswerten von Sensordaten.
Gyro	Gyroskop, Kreiselinstrument, im Programmcode oftmals mit „g“ abgekürzt. Misst Drehbewegungen.
Acc	Accelerometer, Beschleunigungssensor, im Programmcode oftmals mit „a“ abgekürzt. Misst die auf ihn wirkende Beschleunigung.
Mag	Magnetometer, Magnetfeldsensor, im Programmcode oftmals mit „m“ abgekürzt. Misst die Magnetische Feldstärke.
AVR	Einfacher 8-Bit Mikrocontroller.
UART	Universal Asynchronous Receiver Transmitter. Dient zur Realisierung von digitalen seriellen Schnittstellen.
I2C	Inter-Integrated Circuit, einfacher Bus zur Kommunikation unterschiedlicher Geräte.
ISP	In-System-Programming, ermöglicht das Programmieren direkt im Zielsystem.
SPI	Serial Peripheral Interface, ein einfach aufgebauter Bus zur Kommunikation zwischen unterschiedlichen Geräte.
LiPo	Lithium Polymer. Ein wiederaufladbarer Akku.
ADC	Analog Digital Converter. Wandelt ein Analoges Signal in ein Digitales um.
Quaternion	Es handelt ist dabei um ein Zahlensystem ähnlich den komplexen Zahlen. Es besteht aus einem eindimensionalen Realteil und einem dreidimensionalen Imaginärteil, der auch Vektoranteil genannt wird.
XAML	Extensible Application Markup Language, wird verwendet zum Beschreiben der Oberfläche in WPF.
WPF	Windows Presentation Foundation, ein auf .NET basierendes Grafik Framework.

13 Verzeichnisse

13.1 Informationsquellen

Informationen bezüglich der Mikrocontroller Programmierung:
<http://www.mikrocontroller.net/>

Grundlagen des IMU:
http://www.starlino.com/imu_guide.html

Grundlagen und Funktionsweise von Quaternions:
<http://www.x-io.co.uk/res/doc/quaternions.pdf>

Informationen über AVR:
<http://www2.atmel.com>

Informationen bezüglich der C# Programmierung:
<http://msdn.microsoft.com/en-us/vcsharp/>

Download von Eagle, zudem diverse Anleitungen online verfügbar:
<http://www.cadsoft.de>

13.2 Tabellenverzeichnis

Tabelle 1 Aufwandsschätzung.....	8
Tabelle 2 Aufwandkontrolle.....	9
Tabelle 3 Auswahlmatrix Sensor.....	13
Tabelle 4 Auswahlmatrix Mikrocontroller.....	14
Tabelle 5 Stückliste.....	17
Tabelle 6 Softwaretests.....	34
Tabelle 7 Sensor Messungen.....	34
Tabelle 8 Auswertung Funktionsumfang.....	36

13.3 Abbildungsverzeichnis

Abbildung 1 Fliegender Oktokopter.....	5
Abbildung 2 Konzeptdiagramm.....	10
Abbildung 3 9 DOF Sensor Stick.....	14
Abbildung 4 AVR ATmega644.....	15
Abbildung 5 OLED Grafikdisplay.....	15
Abbildung 7 Hardwareschema.....	16
Abbildung 6 XBee Funkmodul.....	16
Abbildung 8 Aufbauplan der Hardware.....	18
Abbildung 9 Hardware Grundplatine.....	19
Abbildung 10 Hardwareaufbau Zwischenschritt.....	19
Abbildung 11 Die Hardware fertig aufgebaut.....	20
Abbildung 12 Ablaufschema.....	21
Abbildung 13 Aufbaudiagramm AVR Software.....	22
Abbildung 14 Anzeige des Startbildschirms.....	24
Abbildung 15 Anzeige bei Start des Messvorgangs.....	24
Abbildung 16 Anzeige beim Halt des Messvorgangs.....	24
Abbildung 17 Tool X-CTU.....	26
Abbildung 18 Ablaufschema.....	27
Abbildung 19 Klassendiagramm.....	28
Abbildung 20 RotaryControl.....	31
Abbildung 22 Frontend Ansicht.....	32

Abbildung 21 CubeControl.....	32
Abbildung 23 Frontend Debug Ansicht	33

13.4 Formelverzeichnis

Formel 1 Quarzberechnung.....	15
Formel 2 Timerfrequenz.....	24
Formel 3 Temperaturberechnung.....	29
Formel 4 Spannungsberechnung.....	29
Formel 5 Quaternion zu Euler	29
Formel 6 Quaternion zu Yaw, Roll, Pitch.....	30

14 Anhang

14.1 Persönliche Angaben



Philipp Weber

Hohmattring 7

8488 Turbenthal

Tel: 076 422 24 41

Email: pw@optimanet.ch

Ausbildung

Automatiker

Arbeitgeber

optimaNet Schweiz AG

Stationsstrasse 56

8472 Seuzach

Tätigkeitsfeld

Webapplikationsentwicklung, Systemadministration

14.2 Bedienungsanleitung

14.2.1 Installation

Eine Installation der Software ist nicht notwendig, da keine speziellen Bibliotheken, Datenbanken oder sonstige Konfigurationen notwendig sind. Die **exe Datei im Ordner „Software\PC\Run\“ kann direkt gestartet werden.**

Folgende Systemvoraussetzungen sollten mindestens erfüllt sein:

- Mindestens Windows XP (besser Windows 7)
- .NET Framework 4.0 oder höher
- Genügend Leistungsreserven (ansonsten kann es zu einer verzögerte Wiedergabe kommen)

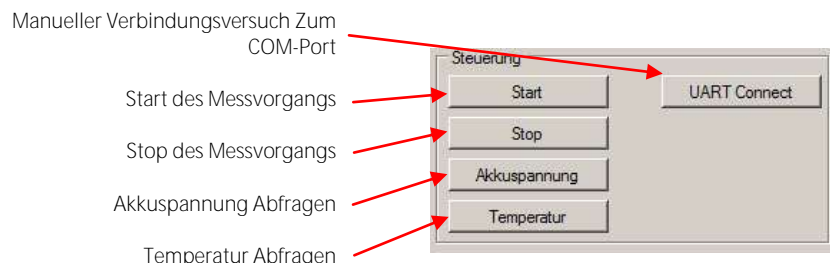
14.2.2 Betriebsanleitung

Gehen Sie vor dem Start der Applikation sicher, dass die Schnittstelle welche Sie zur Kommunikation verwenden möchten, auf COM4 gelegt ist.

Ist der COM-Port nicht in Betrieb, so kann die Anwendung auch funktionslos gestartet werden. Zu finden ist die Anwendung auf der CD unter „Software\PC\Run\imu.exe“. Die Datei 3DTools.dll muss sich beim Start im selben Ordner befinden. Das Hauptfenster öffnet sich nach dem Doppelklicken der exe Datei.

Schalten Sie nun falls noch nicht geschehen, das Sensorboard mit dem darauf vorhandenen Kippschalter ein. Der Status sollte **nun von „Offline“ auf „Online“** wechseln. Ist dies nicht geschehen, so liegt wahrscheinlich ein Fehler der Konfiguration des COM-Ports vor oder das Sensorboard wurde nicht korrekt initialisiert.

Alle benötigten Funktionen sind im Feld „Steuerung“ ersichtlich.



Wird mit „Start“ der Messvorgang gestartet, so darf das Sensorboard nicht bewegt werden bis das Display auf dem Board anzeigt, dass alle Sensoren kalibriert sind. Sollte beim Kalibrieren einer der Sensoren einen Fehler anzeigen, so ist ein erneutes Starten erforderlich.

Die Akkuspannung und Temperatur des Sensorboards werden nicht automatisch abgefragt. Daher sind für beide Abfrageknöpfe vorhanden. Werden diese gedrückt, wird die Messung nach einigen Sekunden im Status angezeigt.

Zum Beenden der Applikation kann diese unter „Datei->Beenden“ oder wie gewohnt geschlossen werden. Das Sensorboard wird mit dem Kippschalter wieder in den Stromlosen Zustand gesetzt.

14.3 Aufgabenstellung



Herr
Philipp Weber
Hohmattring 7
8488 Turbenthal

Uster, 14. Februar 2011

Vordiplomarbeit 2011

Sehr geehrter Herr Weber

Sie erhalten folgende Vordiplomarbeit zugeteilt, welche Sie selbständig zu lösen haben.

Inertial Sensor Board

Inertialsensoren (Trägheitssensoren) werden zur Messung von Bewegungen verwendet. Sie sind ein Hauptbestandteil von Navigationssystemen und kommen auch in der Bildstabilisierung von optischen Systemen vor.

Mit dieser Vordiplomarbeit soll ein IMU Development Board so programmiert werden, daß es die Sensordaten auswerten und an einen PC übertragen kann. Der PC soll die Daten grafisch aufbereitet darstellen können.

Die Aufgabe umfasst folgende Punkte:

- Projektplanung
- IMU Development Board hardwaremäßig komplettieren um Bewegungen messen zu können
- Programmierung des Sensor Boards um Sensordaten auslesen und zum PC übertragen zu können
- Programmierung einer PC-Software, welche die Daten entgegennehmen und aufbereiten kann
- Visualisierung der Sensordaten mit der PC-Software
- Test der Software
- Dokumentation aller Arbeitsschritte

Optionale Ziele:

- Möglichkeit der Filterung der Daten (Fehlerkorrektur)
- Aufbau einer Plattform zur Demonstration des Inertial Sensor Boards

Versand der Aufgabenstellung
Abgabe der 2 Dokumentationen
Präsentation der Arbeit

Freitag, 18. Februar 2011
Dienstag, 21. Juni 2011
Samstag, 9. Juli 2011

Freundliche Grüsse

Daniel Kälin

14.4 Sitzungsprotokoll

14.4.1 Sitzung 1

Höhere Fachschule Uster

HFU

Protokoll Betreuungssitzung Nr: 1

Art der Arbeit	<input checked="" type="checkbox"/> Vordiplomarbeit <input type="checkbox"/> Diplomarbeit <input type="checkbox"/> Abschlussarbeit NDS
Student	Philipp Weber
Betreuer	Daniel Kälin
Thema	Inertial Sensor Board
Ort, Datum, Zeit	Uster, 30. März 2011, 1715

Arbeitsstand	<i>im Zeitplan</i>
Probleme, Fragen	—
Weiteres Vorgehen	<i>gemäss Terminplanung</i>
Beschlüsse	Die Aufgabenstellung ist verstanden. Sie ist im Pflichtenheft ergänzt und präzisiert. Das Pflichtenheft ist genehmigt.
Nächster Termin	<i>Mitte Mai</i>

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

Unterschrift Betreuer

Daniel Kälin

Unterschrift Student

Philipp Weber

14.4.2 Sitzung 2

Höhere Fachschule Uster

HFU

Protokoll Betreuungssitzung Nr: 2

Art der Arbeit	<input checked="" type="checkbox"/> Vordiplomarbeit <input type="checkbox"/> Diplomarbeit <input type="checkbox"/> Abschlussarbeit NDS
Student	Philipp Weber
Betreuer	Daniel Kälin
Thema	Inertial Sensor Board
Ort, Datum, Zeit	Uster, 8. Juni 2011, 1645

Arbeitsstand	<i>Dokumentation beenden Präsentation vorbereiten</i>
Probleme, Fragen	—
Weiteres Vorgehen	<i>Abschluss der Arbeit.</i>
Beschlüsse	
Nächster Termin	-

Der Betreuer gibt dem Studenten jeweils eine Kopie. Er sammelt die ausgefüllten Protokolle und bewahrt sie auf bis 1 Monat nach der Präsentation (Ablauf Rekursfrist).

Unterschrift Betreuer

Daniel Kälin

Unterschrift Student

P. Weber

14.5 CD

Auf der hier angefügten CD sind folgende Ordner Enthalten:

14.5.1 Datenblätter

Alle in diesem Projekt benötigten Datenblätter sind hier enthalten.

14.5.2 Dokumente

Enthält die Dokumentation, Diagramme und Schemas die während des Projektes entstanden.

14.5.3 Hardwareschemas

Enthält die Schemas der in diesem Projekt verwendeten Hardware.

14.5.4 Sitzungsprotokolle

Beide Sitzungsprotokolle sind als Kopie beigelegt.

14.5.5 Software

Enthält alle in diesem Projekt erzeugte Software. Die unterschiedlichen Versionen sind in den Entsprechenden Unterordnern abgelegt. Unter Referenzen sind die Kopie des Original Codes, welche als Ausgangslage für Teile der Anwendung eingesetzt wurden.

14.5.6 Filme

Kurzes Demonstrationsvideos für die bessere Veranschaulichung des Projekts.

14.5.7 Bilder

Enthält Diverse Bilder des Hardware Aufbaus sowie der Zwischenschritte.

14.5.8 Zusätzliches

Enthält einen Teil der in diesem Projekt verwendeten Programme.