Dominikus-Zimmermann-Gymnasium
Landsberg am Lech

Abiturjahrgang 2020

S E M I N A R A R B E I T

Rahmenthema des Wissenschaftspropädeutischen Seminars:

Physik im Alltag und Technik

Leitfach: Physik

Thema der Arbeit:

Aufbau und Funktion eines Drohnensensors – Verarbeitung bestimmter Daten

Verfasser:
Marc Pascal Lohscheidt

Kursleiter:
Walter Kastenmeyer

Abgabetermin:
05.11.2019

Datum und Unterschrift des Kursleiters

Inhaltsverzeichnis

Einleitung

Quadrocopter (im folgenden auch Drohnen genannt) gehören zu den wendigsten ferngesteuerten Flugobjekten der Welt. Ähnlich wie Helikopter können sie in der Luft schweben, sich in alle Himmelsrichtungen bewegen und sich bei geringen Geschwindigkeiten auf der Stelle drehen. Diese Flugeigenschaften verleihen viele Freiheitsgrade, welche die Drohne zu einem sehr flexiblen Gefährt macht. Die Einsatzgebiete einer Drohne sind die Photographie, die Filmproduktion, der Hobbymodellflug, Flugrennen und vieles mehr. Ein Helikopter ist durch seine Flexibilität und seine Freiheitsgrade in der Steuerung sehr schwer zu handhaben. Aus genau diesem Grund benötigen Quadrocopter eine Unterstützungsfunktion in der Steuerung. Neben bereits existierenden „Antikollisionsalgorithmen“, die Drohnen autonom an Hindernissen vorbeisteuern, selbst wenn der Pilot darauf zufliegt, gibt es eine wirklich essentielle Funktion: der autonome Schwebeflug. Dieser Unterstützungsalgorithmus sorgt dafür, dass sich die Drohne in den schwersten Situationen stabil in der Luft verhält. Sollte also der Pilot die Fernsteuerung loslassen, so bremst die Drohne sich automatisch durch die richtig gewählte Neigung ab (ähnlich wie bei einem Helikopter). Um diese Leistung zu bewältigen benötigt die Drohne zwei Bauteile, die dem Algorithmus die notwendigen Daten übermitteln. Es handelt sich um das Gyroskop und den Accelerometer, welche im Verbund auch IMU[1] genannt werden. Das Gyroskop alleine misst Drehraten, wohingegen der Accelerometer Linearbeschleunigungen erfasst. Beide arbeiten in jeweils drei aufeinander orthogonalen Raumachsen und können somit verwendet werden, um die aktuelle Lage im Raum festzustellen, ähnlich wie es in Flugzeugen mit einem künstlichen Horizont ermöglicht wird. Diese IMUs funktionieren jedoch komplett elektronisch und sind wesentlich genauer. Auch im Alltag finden diese hochgenauen Messgeräte Verwendung, da jedes moderne Smartphone eines besitzt. Jedes Mal, wenn man sein Handy dreht und sich das Bild automatisch in Querformat wandelt, arbeitet eine IMU. Die Frage ist jedoch, wie man in extremen Situationen, wie beispielsweise einem rasanten Drohnenflug, Daten eines Gyrosensors oder eines Accelerometers sinnvoll auswertet. Welche Probleme kommen auf den Algorithmus zu? Wie funktionieren diese beiden Bauteile?

[1] Inertial Measurement Unit (auf dt. inertiale Messeinheit)

In dieser Seminararbeit behandle ich zum einen den Aufbau und die Funktionsweise einer solchen inertialen Messeinheit. Zum anderen beschreibe ich einen Lösungsansatz zur korrekten Datenverarbeitung einer solchen innerhalb eines Quadrocopters, um einen Schwebeflug zu ermöglichen.

Aufbau und Funktionsweise einer IMU[2]

Es gibt viele verschieden IMUs, die für spezielle Aufgabenbereiche eingesetzt werden können. Als IMU bezeichnet man die Kombination verschiedener Inertialsensoren[3]. Meistens besteht eine solche Messeinheit aus einem Beschleunigungssensor und einem Drehratensensor. Da diese beiden Sensoren alle dreidimensionalen räumlichen Bewegungen erfassen können, sind sie damit für die Luftfahrt oder zum Beispiel für eine Drohne ideal. Auch bei Beschleunigungssensoren und Drehratensensoren gibt es viele verschiedene Typen, die für spezielle Einsatzgebiete konzipiert wurden. Ich werde in dieser Arbeit jedoch nur wenige nennen und mich auf die wesentlichen konzentrieren. Der „MPU-6050“, der InvenSense Inc, ist eine IMU, die auf jeweils drei Achsen eine Linearbeschleunigung und eine Drehrate misst. Diese Messeinheit verwende ich auch für meine Versuche im dritten Kapitel und werde deswegen auf diesen Typus genau eingehen.

Bei den Beschleunigungssensoren unterscheidet man zwischen spannungsmessenden Beschleunigungssensoren, welche nur für dynamische Beschleunigungsmessungen geeignet sind und wegmessenden Beschleunigungssensoren, die auch statische Beschleunigungen messen. Im Allgemeinen ist hier die Bau- und Messweise entscheidend. Die wegemessenden Sensoren werden in der Regel durch Messungen von Kapazitäten, die durch eine seismische Masse mechanisch verändert werden, realisiert. Im Gegensatz dazu verwenden spannungsmessende Beschleunigungssensoren piezoelektrische Elemente, welche die Eigenschaft besitzen, Kristallgitterverformungen in Spannung umzuwandeln. Bei diesen Elementen kann jedoch nur eine Spannungsänderung und damit nur eine Beschleunigungsänderung gemessen werden.

[2] (Hüning 2016)

[3] (VectorNav kein Datum)

Ein Drehratensensor, auch Gyroskop genannt, gibt eine Drehrate an. Oft machen sich derartige Sensoren die Corioliskraft zunutze, die eine Auslenkung einer Masse verursacht. Diese Auslenkung kann dann auf verschiedenen Wegen gemessen werden.

Solche Inertialsensoren werden sehr oft auf MEMS[4]-Basis realisiert. Solche MEMS-Sensoren integrieren elektrische und mechanische Schaltungen auf kleinstem Raum und wurden dafür entwickelt, kleine Beschleunigungen kapazitiv zu messen. Diese Sensoren sind sehr empfindlich und können Temperatureinflüssen gut standhalten.[5] Beeindruckend ist hier auch die Größe dieser Bausteine, da IMUs oft in Chips eingebaut sind, die nur wenige Millimeter groß sind. Gerade diese Eigenschaft ist auch heute von großem Nutzen, da diese Sensoren in jedem Smartphone Platz finden und wenig Gewicht beanspruchen.

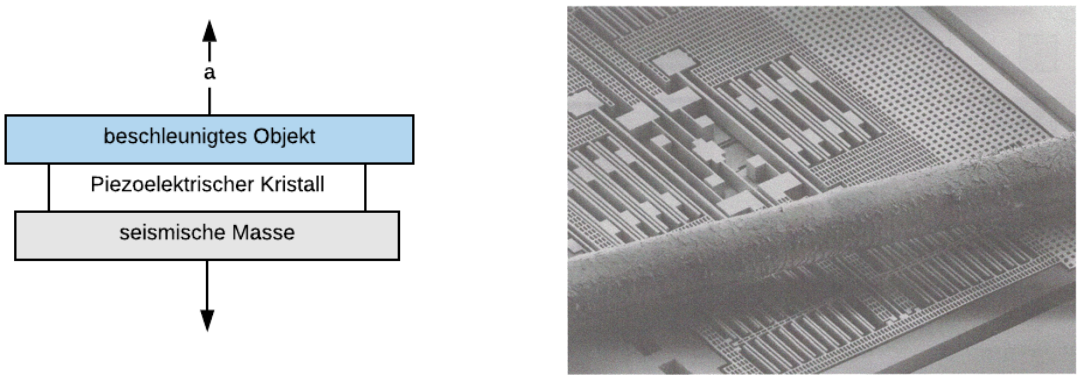


Abbildung 1: Prinzip eines Beschleunigungssensors mit Piezoelektrischen Kristall (in Anlehnung an: (Hüning 2016) vgl. S. 102 Abb. 4.30) (links) und Ausschnitt der Struktur eines 3D-MEMS-Beschleunigungssensors im Vergleich zu einem menschlichem Haar, Strukturgröße < 20 µm ((Hüning 2016) vgl. S. 26 Abb. 2.23)

Spannungsmessender Beschleunigungssensor[6] [7]

Hauptelement eines jeden Beschleunigungssensors ist in der Regel eine Masse, die dafür sorgt, dass ein Messelement ausgelenkt wird. Es handelt sich um die sogenannte seismische Masse. Letztendlich wird hier das Prinzip der Trägheit verwendet, welches besagt, dass ein ruhender Körper in Ruhe bleibt, wenn keine äußere Kraft auf ihn einwirkt. Wird ein Objekt

[4] Micro-Electro-Mechanical System

[5] (Hering, et al. 2018) vgl. S. 265 K. 3.3.5

[6] (Hering, et al. 2018) vgl. S. 3 K. 2.1.1

[7] (Hüning 2016) vgl. S. 106 K. 4.8.2

beschleunigt, so wirkt eine Kraft entgegen der Bewegungsrichtung auf das beschleunigte Objekt. Abbildung 1 (links) zeigt den Aufbau eines solchen Sensors. Die seismische Masse ist über einen piezoelektrischen Kristall mit dem beschleunigten Objekt verbunden. In diesem Fall wirkt die entgegenwirkende Kraft der seismischen Masse auf den Kristall ein und verformt diesen. Piezokristalle erzeugen eine Potentialtrennung durch Verschiebung der Ladungsschwerpunkte bei Kristallgitterverformungen. Misst man nun diese Spannung, so kann man Aussagen über die Beschleunigung treffen. Jedoch lassen sich nur dynamische Änderung der Beschleunigung mit einem solchen Sensor erfassen. Dies liegt an der Hochpasscharakteristik eines Piezokristalls, da dieser Kristall nur bei einer Änderung eine Spannung erzeugt, diese jedoch nicht konstant hält.

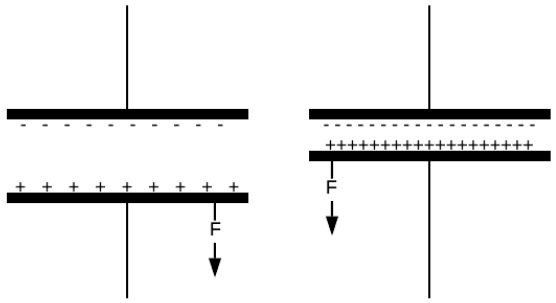


Abbildung 2: Zwei Kondensatoren mit verschiedenen Plattenabständen bedingt durch eine Kraft

Wegmessender Beschleunigungssensor[8]

Der wegmessende Beschleunigungssensor hat ebenso eine seismische Masse, die in einer Federung gelagert ist. Hier werden in der Regel jedoch kapazitive Effekte genutzt, um eine Beschleunigung zu erfassen. Wie in Abbildung 2 stark vereinfacht dargestellt, wirkt eine Kraft orthogonal zu den frei beweglichen Kondensatorplatten, welche distrahiert oder kontrahiert werden. Da Kondensatorplatten durch Abstandsänderungen Kapazitätsschwankungen erzeugen, kann man diese Eigenschaft nutzen, um die Kräfte zu errechnen. Natürlich ist hier vorausgesetzt, dass die Kräfte nur entlang einer Geraden gemessen werden können, solange man nicht mehrere dieser Kondensatoranordnungen in verschiedenen Richtungen einsetzt.

[8] (Hüning 2016) vgl. S. 102 K. 4.8.1

Außerdem muss man sagen, dass dieser Aufbau stark vereinfacht ist und es noch weitere Bauteile benötigt, um ein funktionales System darzustellen. Diese wegmessenden Sensoren haben gegenüber den spannungsmessenden den Vorteil, dass sie sowohl dynamische als auch statische Beschleunigungen sehr gut messen können. Deswegen werden gerade diese für IMUs genutzt.

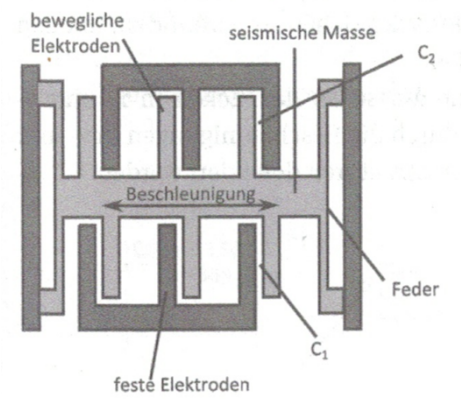


Abbildung 3: Beschleunigungssensor mit Kammstruktur an Differentialkondensatoren ((Hüning 2016) vgl. S. 103 Abb. 4.31)

Aufbau eines wegmessenden Beschleunigungssensors[9]

Wie bereits beschrieben benötigt ein derartiger Sensor eine seismische Masse, welche für die Auslenkung bei Bewegungsänderungen verantwortlich ist. Dieser Ansatz wird meistens durch eine Kammstruktur wie in Abbildung 3 ersichtlich realisiert. Dabei ist die seismische Masse linear beweglich federnd aufgehängt und bildet durch ihre Kammspitzen auf beiden Seiten bewegliche Elektroden. Die Gegenstücke, feste Elektroden, haben ebenfalls Kammspitzen und sind leicht versetzt. Die festen und beweglichen Elektroden können, durch Anlegen einer Spannung, ein elektrisches Feld erzeugen und dadurch als Kondensatorplatten agieren. Durch die leichte Versetzung auf der jeweiligen Seite kommt es bei einer Bewegung der seismischen Masse zu einem Gegenspiel. Wird die Kapazität, hier C_1, auf der einen Seite größer, weil der Abstand der Elektroden abnimmt, so wird C_2 auf der anderen Seite umso kleiner, da der Abstand der Elektroden zunimmt. Dieses Gegenspiel sorgt für eine Linearisierung des Ausgangsignals und kompensiert Temperatureinflüsse. Da dieser Aufbau nur eine lineare Auslenkung misst, benötigt man, um alle möglichen Bewegungen abzudecken, einen für jede

[9] (Hüning 2016) vgl. S. 102 K. 4.8.1

der drei orthogonal zueinanderstehenden Raumachsen im intrinsischen Koordinatensystem. Für Auslenkungen zwischen zwei Raumachsen kann man Ausgangsignale der jeweiligen Achse vektoriell addieren und einen Bewegungsvektor erfassen.

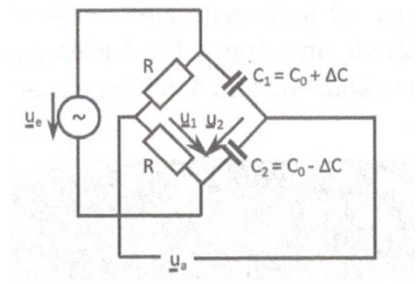


Abbildung 4: Vollbrücke mit Differentialkondensator als Auswerteschaltung ((Hüning 2016) vgl. S. 103 Abb. 4.31)

Funktionsweise eines wegmessenden Beschleunigungssensors[10]

Auch wenn die Beschleunigung des Sensors Kapazitätsschwankungen erzeugt, werden diese nur indirekt gemessen. Tatsächlich wird hier eigentlich eine Ausgangspannung gemessen, die durch variable Impedanzverhältnisse verändert wird. Die folgende Abbildung 4 zeigt ein Schaltbild, das die Messvorrichtung vereinfacht darstellt. Die beiden Kondensatoren repräsentieren die Teilkapazitäten $C_{1,2}$. Die gleich großen Widerstände R bilden einen Referenzpunkt, der den Mittelwert der sinusförmigen Wechselspannung U_e des Generators aufweist. Die Messung der Ausgangsspannung U_a erfolgt, wie in Abbildung 4 klar zu erkennen, zwischen dem gerade erwähnten Referenzpunkt und den beiden Kondensatoren (Mittenabgriff des kapazitiven Spannungsteilers, gebildet aus C_1 und C_2). Außerdem sind weitere zwei Spannungen von Interesse: zum einen der Spannungsabfall $\underline{U}_1$ am ohmschen Spannungsteiler und zum anderen der Spannungsabfall $\underline{U}_2$ am kapazitiven Spannungsteiler. In dem Fall, dass C_1 und C_2 kapazitätsgleich sind, ist U_a gleich 0. Jedoch steigt bei zunehmender Differenz der beiden Kapazitäten U_a an. Die Phasenlage der oszillierenden Ausgangspannung bestimmt über das Vorzeichen der Bewegungsrichtung.

Nun lassen sich durch entsprechende physikalische Gesetze folgende Formeln und Abhängigkeiten aufstellen:[11]

[10] (Hüning 2016) vgl. S 102 K. 4.8.1

[11] (Hüning 2016) vgl. S. 103 und S. 104 Abb. 4.37 – 4.43

Die Kapazitäten in Ruhelage (Δd = 0) und die Kondensatorformel:

$$C_1 = C_2 = \frac{\varepsilon A}{d}$$

Die Auslenkung der seismischen Masse um Δd erzeugt eine Änderung der beiden Kapazitäten:

$$C_1 = \frac{\varepsilon A}{d - \Delta d}$$

$$C_2 = \frac{\varepsilon A}{d + \Delta d}$$

Die Impedanzen (Scheinwiderstände) der Kondensatoren lauten demnach:

$$\underline{Z}_{C1} = \frac{1}{j\omega C_1} = \frac{d - \Delta d}{j\omega \epsilon A}$$

$$\underline{Z}_{C2} = \frac{1}{j\omega C_2} = \frac{d + \Delta d}{j\omega \varepsilon A}$$

Aus diesen Impedanzen ergibt sich die Formel für den Spannungsabfall $\underline{U}_2$:

$$\underline{u}_2 = \underline{u}_e \cdot \frac{\underline{Z}_{C2}}{\underline{Z}_{C1} + \underline{Z}_{C2}} = \underline{u}_e \cdot \frac{\frac{d + \Delta d}{j\omega \varepsilon A}}{\frac{d - \Delta d}{j\omega \varepsilon A} + \frac{d + \Delta d}{j\omega \varepsilon A}} = \underline{u}_e \cdot \frac{\frac{d + \Delta d}{j\omega \varepsilon A}}{\frac{2d}{j\omega \varepsilon A}} = \underline{u}_e \cdot \frac{d + \Delta d}{2d}$$

U_a ergibt sich aus der Differenz von U_1 ($U_e/2$) und U_2:

$$\underline{u}_a = \underline{u}_1 - \underline{u}_2 = \frac{\underline{u}_e}{2} - \underline{u}_e \cdot \frac{d + \Delta d}{2d} = \underline{u}_e \cdot \left(\frac{2}{2d} - \frac{d + \Delta d}{2d}\right) = -\underline{u}_e \cdot \frac{\Delta d}{2d}$$

Zu beachten ist, dass hier im Bereich der komplexen Zahlen gerechnet wird, weil sich dies in der Wechselstromrechnung elegant bewährt hat. Die imaginäre Einheit ist hier, wie in der Physik üblich, als j angegeben. Die Ausgangsspannung U_a wird, wie in der letzten Formel zu sehen, durch die Differenz von U_1, die am ohmschen Spanungsteiler entsteht, und U_2, die am kapazitiven Spannungsteiler entsteht, gebildet. Der ohmsche Spannungsteiler bleibt hier immer gleich und halbiert somit die Eingangsspannung. Der Subtrahend U_2 ist flexibel, da er durch ein Impedanzverhältnis erzeugt wird. Dieses Verhältnis wird durch den Abstand der Kondensatorplatten, also durch den kapazitiv erzeugten Blindwiderstand, verändert. Da für die gesamte Schaltung die gleiche Wechselstromquelle benutzt wird, streicht sich die Frequenz ω heraus. Das gleiche gilt für die Konstante ε, die sich aus der elektrischen Feldkonstante ε_0 und der relativen Permittivität des Dialektrikums ε_r zusammensetzt und für die Fläche A, da beide Kondensatoren die gleichen Größen haben. So ist klar, dass für U_2 nur

die Eingangsspannung U_e, der Abstand der Kondensatorplatten, also die Auslenkung Δd, und der Abstand der Kondensatorplatten in Ruhelage, also d, entscheidend sind. Außerdem ist anzumerken, dass U_a weiterhin eine oszillierende Spannung ist, bei der die Amplitude und Phase gemessen werden muss.

Dieses Modell habe ich noch einmal in einer Simulation aufgebaut und als Videodatei im Anhang gespeichert. Die Simulation zeigt die Beziehungen der oben genannten Größen klar auf und soll deutlich machen, was eine Kapazitätsänderung der beiden Kondensatoren C_1 und C_2 für U_a bedeutet.

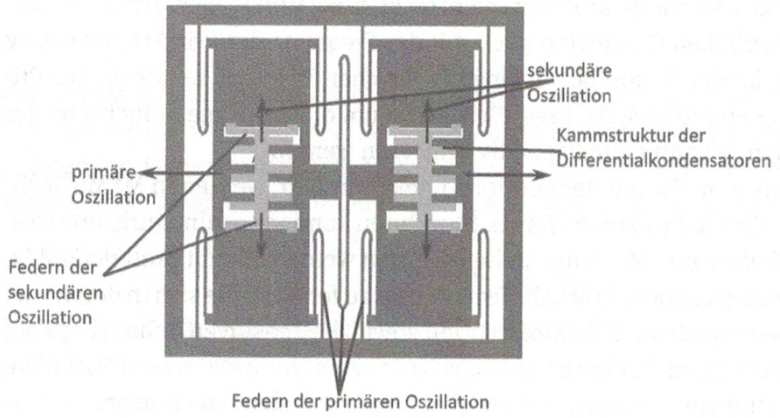


Abbildung 5: Prinzip der Realisierung des Drehratensensors auf MEMS-Basis ((Hüning 2016) vgl. S. 92 Abb. 4.22)

Drehratensensor

Ein Drehratensensor hat viele Gemeinsamkeiten mit dem vorher beschriebenen wegmessenden Beschleunigungssensor. Dies ist auch in Abbildung 5 zu erkennen, da auch hier eine seismische Masse mit Kammstruktur federnd aufgehängt ist. Daneben existiert noch ein weiteres Merkmal. Die gesamte Struktur, welche die Kammstruktur festhält, ist nochmal orthogonal zur Kammstruktur federnd aufgehängt. Es existieren also zwei Freiheitsgrade (Richtungen) in welche die Struktur oszillieren kann. Eine kapazitive Messvorrichtung existiert jedoch nur in der Kammstruktur. Um zu verstehen, warum dieser MEMS-Baustein so angeordnet ist, müssen wir zuerst die Corioliskraft genauer betrachten.

Die Corioliskraft ist eine Scheinkraft, die im rechten Winkel zur Bewegungsrichtung wirkt, sofern sich die Bewegungsrichtung auf ein rotierendes System bezieht. Letztendlich kann man sagen, dass die Bahn eines geradlinig bewegten Körpers innerhalb eines rotierenden Systems

eine Spirale erzeugt. Festzuhalten ist hier, dass der Körper mit einer bestimmten Masse sowohl eine Bewegungsrichtung als auch ein rotierendes System benötigt, um diese Kraft zu erzeugen. Die Corioliskraft wird durch folgende Formel beschrieben:

$$\overrightarrow{F_c} = -2m * (\vec{\omega} \times \vec{v})$$

Die vektorielle Form der Gleichung gibt an, in welche Richtung die Kraft wirkt. Da hier das Kreuzprodukt zweier Vektoren gebildet wird, handelt es sich um eine Kraft, die orthogonal zu den beiden Vektoren $\vec{\omega}$ und $\vec{v}$ wirkt. $|\vec{\omega}|$ ist hier die Winkelgeschwindigkeit des Systems deren Vektor entlang der Rotationsachse wirkt. $|\vec{v}|$ ist die Bewegungsgeschwindigkeit, die geradlinig in eine Richtung verläuft. m ist die Masse des Messkörpers. Klar ist, wenn eine dieser Größen Null wird, beziehungsweise der Betrag der Vektoren Null wird, so wirkt keine Corioliskraft. Daraus ergibt sich der Umstand, dass bei Drehratenmessungen in Ruhelage, also wenn sich das Objekt nicht bewegt, kaum eine Kraft gemessen wird, also die Rotationsbewegung nicht erfasst wird.

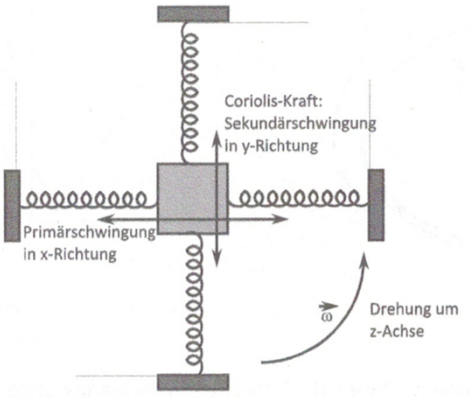


Abbildung 6: Prinzip des Drehratensensors auf Basis der Corioliskraft ((Hüning 2016) vgl. S. 92 Abb. 4.21)

Dieses Problem kann aber mit dem vorher beschriebenen Aufbau eines solchen Drehratensensors behoben werden. Um eine ständige Bewegungsgröße zu erhalten wird die gesamte Aufhängung der Kammstruktur künstlich in Schwingung versetzt. Dieses Oszillieren wird als Primärschwingung bezeichnet. Abbildung 6 zeigt ein vereinfachtes Modell des Aufbaus. Hier ist zu erkennen, dass die Primärschwingung entlang der X-Achse schwingt (die Wahl der Achsenbeschriftung ist irrelevant, solange sie orthogonal zueinander sind). Dreht man den Körper nun an der Z-Achse, so wirkt, da alle nötigen Größen vorhanden sind, eine Kraft in Y-Richtung: die Corioliskraft. Die Amplituden der Sekundärschwingung können nun mit der Kammstruktur, ähnlich wie bei wegmessenden Beschleunigungssensoren, kapazitiv

gemessen werden. Auch hier gilt, dass dieser Aufbau nur eine Achse misst und damit drei weitere Module benötigt werden, um alle Raumachsen abzudecken und damit einen vollständigen Inertialsensor für eine IMU zu erhalten.

Natürlich gibt es auch andere Vorgehensweisen, um Drehraten oder Neigungen zu messen. Selbst die Nutzung eines Beschleunigungssensors wäre möglich, in dem die Neigung im Bezug zum Erdmittelpunkt mithilfe der Erdbeschleunigung errechnet wird. Gründe, die gegen einen derartigen Lösungsansatz sprechen werden im Kapitel „Datenverarbeitung“ behandelt.

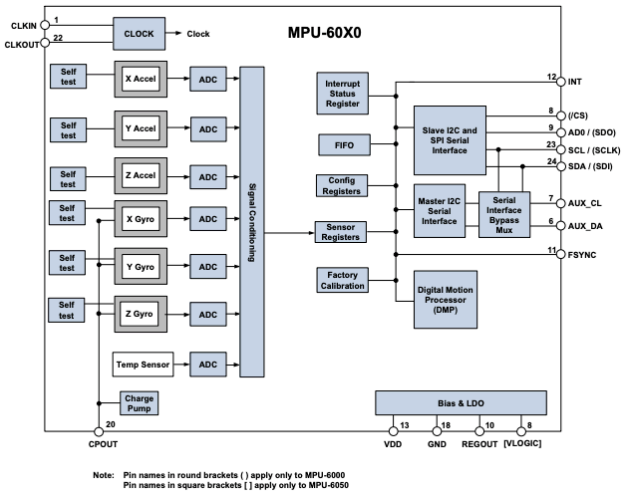


Abbildung 7: Blockdiagramm von „MPU-60X0" (InvenSense Inc 2013)

Aufbau eines Sensorchips am Beispiel des MPU-60X0™ [12]

Solche Inertialsensoren werden selten alleine eingesetzt. Im Normallfall besitzt jeder dieser Sensoren einen eigenen Mikrocontroller, der die Rohdaten auffasst und über eine Schnittstelle einfacher lesbar macht. Bereits hier werden Daten verarbeitet, beispielsweise von einem analogen in ein digitales Signal. Wie oben schon beschrieben wird im Rahmen dieser Arbeit der „MPU-6050“ als Erklärungsbeispiel genutzt. Da diese IMU dem „MPU-6000“ sehr ähnelt, gibt es im Datenblatt nur ein Blockdiagramm, das für beide gilt, unter dem Namen „MPU-60X0™“. Abbildung 7 zeigt den Chipaufbau des „MPU-60X0™“ [13] der Firma InvenSense Inc. Dieser Sensor misst auf sechs Achsen, drei Linearachsen und drei Rotationsachen. Die sechs Messeinheiten sind mit der jeweiligen Achse und einem Kürzel beschriftet. „Accel“ steht

[12] (Hüning 2016) vgl. S. 108

[13] MPU-60X0 ist ein Grundmodell von 6050 und 6000

für Accelerometer also Linearbeschleunigungssensor und „Gyro“ steht für Gyroskop, also für die Drehratenmessesensoren. Alle Messdaten werden durch einen Analogdigitalwandler(„ADC“) digitalisiert und dann aufbereitet („Signal Conditioning“). Im letzten Prozess wird dafür gesorgt, dass die Daten in den richtigen Registern gespeichert werden, damit man sie über einen Datenbus, in diesem Fall eine I^2C Schnittstelle, auslesen kann. Dieses Modul verfügt natürlich über weitaus mehr Funktionen, jedoch habe ich mich hier auf den grundlegenden Weg der Daten aus dem Sensor zum Mikrocontroller konzentriert. Eine IMU stellt also Daten zur Verfügung, die kaum gefiltert sind und zu sensitiv gegenüber kleinsten Vibrationen sind. Das macht sie oft untauglich für viele Bereiche im Alltag. Um deren ganzes Potential auszunutzen, bedarf es daher einer speziellen Datenverarbeitung.

Datenverarbeitung

In diesem Abschnitt der Seminararbeit behandle ich die sinnvolle Verarbeitung von Messdaten einer IMU. Diese ist notwendig um einem Quadrocopter, im späteren Verlauf auch Drohne genannt, den Schwebeflug zu ermöglichen. Im Folgenden werde ich mich auf die IMU „MPU-6050“ beziehen und alle erwähnten Versuche werden mit diesem Modell durchgeführt. Des Weiteren werde ich einen Lösungsansatz von Joop Brokking heranziehen und mit meiner überarbeiteten Version vergleichen.

Problemstellung

Der Quadrocopter ist eine Drohne, die es in verschiedensten Versionen und Größen gibt. Er hat vier unabhängig laufende Rotoren und einen Mikrocontroller, der diese ansteuert. Um die Drohne autonom schweben zu lassen, benötigt man reine, zuverlässige Daten über die aktuelle Rotation (Lage) im Raum und die derzeitige Bewegungsrichtung. Das Problem ist, dass die Drohne durch ihre starken Rotoren bereits eine sehr große Eigenvibration hat, welche die Messwerte verfälscht. Des Weiteren könnte es bei akrobatischen Flugmanövern zu einem sogenannten „Gimbal lock“ kommen. Dies ist ein aus Luft- und Raumfahrt bekanntes Phänomen, bei dem die Rotation nicht eindeutig durch drei virtuelle Achsen beschrieben werden kann und dadurch keine sinnvollen Daten geliefert werden. Dies kann zu schweren Fehlfunktionen führen. Ein Lösungsansatz könnte das Rechnen mit Quaternionen sein. Eine wichtige Frage bleibt die erforderliche Messgenauigkeit. Einerseits müssen Daten präzise

genug sein, um ein empfindliches Reaktionsvermögen zu haben, Andererseits sollten unnötige Messungen vermieden werden, um Verarbeitungsgeschwindigkeit einzusparen und den Mikrocontroller zu entlasten. Außerdem ist es möglich, dass nach gewisser Zeit die Referenzdaten eine „Verschmutzung“ erleiden, die durch Kalibrierungen behoben werden muss. Hinzu kommt die Tatsache, dass ein Gyrosensor nur Messdaten über die gerade wirkenden Beschleunigungen in alle Himmelsrichtungen liefert und somit die aktuelle Rotation im Raum selber errechnet werden muss. Da das Gyroskop und der Accelerometer (in der Regel) keinen dieser Vorgänge übernehmen, sondern nur Messdaten abgeben, ist es die Aufgabe des Mikrocontrollers, mithilfe eines passenden Algorithmus, die Daten zu bereinigen und auszuwerten.

Grundsätzliches zum Modellversuch

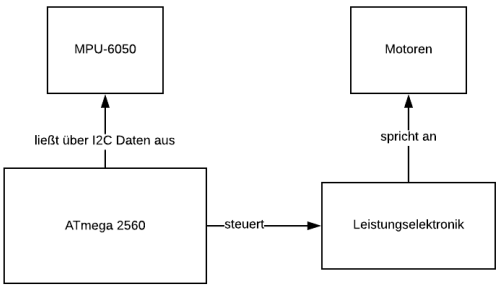


Abbildung 8: Drohnenmodellaufbauschema

Das Herz eines Quadrocopter ist ein Mikrocontroller. In meinem Modellversuch verwende ich den „ATmega 2560“, welchen ich mit der „Arduino“[14] Entwicklungsumgebung (wird später nur noch IDE[15] genannt) programmiere[16]. Dieser verbindet die Sensoren, darunter auch den Beschleunigungssensor, mit der Leistungselektronik für die Motoren und ist für das Steuern zuständig. Auf dem „ATmega 2560“ werden entsprechende Algorithmen implementiert, um Daten korrekt auszuwerten und später anwenden zu können.

Das Fundament in der IDE basiert auf sogenannten Schleifen. Hierbei wird die Hauptlogik des Mikrocontrollers immer wieder aufgerufen, um sicher zu stellen, dass das Programm nicht

[14] www.arduino.cc

[15] Abkürzung für Entwicklungsumgebung – „Integrated development environment“

[16] Die in den Beispielen verwendete Programmiersprache entspricht C++

aufhört zu arbeiten. Die Hauptlogik wird in der sogenannten „Loop“-Methode[17] implementiert. Die Aufruffrequenz der „Loop“ ist essentiell für akkurate Messungen. Neben dieser Methode gibt es noch eine weitere Standardmethode der IDE, welche nur einmal bei der Aktivierung des Mikrocontrollers aufgerufen wird. Es handelt sich um die „Setup“-Methode [18], in der meistens Konfigurationen und Eichungen von Sensoren und Schnittstellen vorgenommen werden.

Umrechnung der Daten[19]

Um zu verstehen was mit den Daten gemacht werden muss, muss zunächst verdeutlicht werden, welche Daten der „MPU-6050“ über seine I²C-Schnittstelle bereitstellt. Da der Beschleunigungssensor sowohl über einen Accelerometer als auch einen Gyrosensor verfügt, liefert er Informationen über die aktuell wirkenden Linearbeschleunigungen in alle Himmelsrichtungen und er misst Drehraten an drei Achsen. Die Accelerometersignale werden in Vielfachen von $9{,}81\frac{m}{s^2}$ übertragen. Der Gyrosensor liefert seine Daten in Grad pro Sekunde [°/s], wobei der Rohdatenwert[20] des Sensors 65,5 pro Grad pro Sekunde entspricht.

$$65{,}5 = 1\ \frac{°}{s}$$

Aus diesem Grund müssen die Daten des Sensors erst umgerechnet werden, um einen sinnvollen Wert zu erhalten. Der entsprechende Code ist hier.

gradProSekunde = ausgabeWert / 65.5;[21]

Nun haben wir eine Winkelgeschwindigkeit einer bestimmten Achse, durch die man mithilfe der Integration einen Winkel relativ zum Ursprungsmesspunkt bestimmen kann. Aus Gründen der Präzision werden sehr kleine gleichmäßige Integrationsschritte benötigt. Durch die hohe Aufruffrequenz der Loop Methode können relativ genaue Werte erzielt werden. Dennoch muss die Aufrufschleife auf eine bestimmte Taktrate festgelegt werden. Die Lösung ist hier eine dynamisch anpassbare Verzögerung, welche, selbst bei unterschiedlich lang

[17] „Loop“ englisch für Schleife – in diesem Kontext ein Methodenbezeichner

[18] „Setup“ englisch für Konfiguration – in diesem Kontext ein Methodenbezeichner

[19] (InvenSense Inc 2013) vgl. S.12 - 13

[20] Ein roher Ausgabewert entspricht hier einem nicht bearbeiteten Wert der direkt aus dem Sensor kommt

[21] Programmierbefehl der Sprache C++

andauernden Befehlsketten, eine bestimmte Frequenz taktet. In meinem Modellversuch nutze ich hierfür die Werte von Joop Brokking und damit eine Frequenz von 250 Hz. Da gilt:

$$1Hz = 1\frac{1}{s}$$

muss in jedem Programmdurchlauf der reine Ausgabewert des Sensors durch 250 und 65,5 dividiert werden. Joop Brokking fasst diese beiden Zahlen zu einer Konstanten zusammen, um eine Rechenoperation pro Durchlauf zu ersparen. Dies spart Rechenzeit. Er multipliziert den Ausgabewert mit 0,0000611, was dem Kehrwert von 250 * 65,5 entspricht. Der entsprechende Code bezogen auf Beispielsweise die X-Achse lautet:

```
aktuellerWinkelAnDerXAchse += xAchseAusgabewert * 0,0000611;
```

Der „+=“ Operator stellt lediglich eine Addition zu dem bereits existierenden Winkel dar. Die Addition ist notwendig, da es sich um eine Integration der Winkelwerte mit 250Hz-Schritten handelt. Dieser Teil des Messalgorithmus lässt sich fast identisch auf Accelerometerwerte anwenden. Sie haben jedoch einen anderen Faktor und werden erst später bei der Kalibrierung des Messinstruments interessant. Wenden wir diesen Algorithmus auf alle Rotationsachsen an, so können wir unseren Winkel, bezogen auf den Ursprungsmesspunkt auf allen 3 Achsen bestimmen. Dieser Winkel ist jedoch unbrauchbar, sollte die Drohne beim Start nicht auf waagrechtem Untergrund gestanden haben. Deshalb ist noch ein Kalibrieren notwendig. Betrachtet man die tatsächlichen reinen Messwerte des „MPU-6050“, so ist zu erkennen, dass selbst im ruhenden Zustand, Rotationen gemessen werden. Dies macht eine Eichung unumgänglich.

Eichung der Sensoren[22]

Eine Eichung der Sensoren erfordert lediglich eine kompensation der abweichenden Nullwerte, da sich in einer definierten Ruhelage die Ergebnisse Null nähern sollten. Die Eichung ist wichtig, weil bei grundsätzlich verschobenen Messwerten unvorhergesehene Effekte eintreten können. Beispielsweise könnte die Drohne ein ständiges Drehmoment erfassen, was dazu führen würde, dass der virtuell berechnete Winkel eine fortwährende, fehlerhafte Änderung aufweist. Das Resultat wäre eine sich dauerhaft drehende Drohne. Die Eichung sollte in der Konfigurationsphase durchgeführt werden, weil dort noch keine

[22] (Brokking 2016)

relevanten Messungen stattfinden und die Motoren der Drohne noch nicht aktiv sind und Störsignale erzeugen können. Dementsprechend muss der Algorithmus innerhalb der „Setup“-Methode ausgeführt werden. Der Algorithmus bezieht lediglich in einem sehr kurzen Zeitraum einige Tausend Messwerte für jede Achse, summiert diese auf und bildet den jeweiligen Durchschnitt. Der daraus resultierende Wert wird danach in jeder Iteration[23] von dem jeweiligen rohen Ausgabewert abgezogen. Dies sollte noch vor allen anderen Datenverarbeitungsschritten passieren. Joop Brokking führt eine derartige Eichung ebenfalls durch. Er eicht jedoch nur den Gyrosensor, ohne dies näher zu begründen. In dem von ihm angegebenen Code ist erkennbar, dass der Accelerometer später keine große Rolle spielt. Dennoch ist es wahrscheinlich sinnvoll den Linearbeschleunigungssensor auch zu eichen, da dieser gerade für die Erfassung von linearen Bewegungen essentiell ist und dessen Werte ohne Eichung nicht verwertbar sind. Die Eichung der Werte für den Accelerometer läuft im Allgemeinen genauso wie für den Gyrosensor ab.

Kalibrieren des Gyrosensors[24]

Wie im Kapitel „Umrechnung der Daten“ bereits erwähnt, ist ein Gyrosensor unbrauchbar für eine Drohne, sollte seine Z-Achse zu Beginn der Aufzeichnungen nicht in negativer Richtung zum Erdmittelpunkt zeigen. Dies hängt damit zusammen, dass der Gyrosensor nur relative Messungen durchführt und ohne realen Bezugspunkt mit einem Winkel, keine reelle Aussage über die Rotationslage, im Bezug zum Erdmittelpunkt, gemacht werden kann. Der Accelerometer jedoch misst lineare Beschleunigungen und damit auch die Erdbeschleunigung. Diese wirkt immer Richtung Boden (Erdmittelpunkt) und somit stellt sie einen optimalen Bezugspunkt da.

Im besten Fall ist die Drohne zum Zeitpunkt der Kalibrierung auf perfekt waagrechtem Boden, dann misst der Sensor auf der Z-Achse ziemlich genau 1 g (9,81 $\frac{m}{s^2}$). Sollte die Drohne bereits beim Start ausgelenkt sein, so ist der Erdbeschleunigungsvektor unter Umständen auf die jeweiligen Raumachsen x und y verteilt. Nun kann man mithilfe der analytischen Geometrie die Vektorverteilung ausnutzen, um die Rotation in Grad für die zugehörige Rotationsachse zu

[23] In diesem Fall ein Durchlauf der „Loop“ Methode

[24] (Brokking 2016)

berechnen. Für die Neigung ist letztendlich nur die X- und Y-Achse interessant, da eine Drehung an der Z-Achse nur dazu führt, dass sich die Drohne dreht, aber nicht neigt. Außerdem ist die Berechnung nicht möglich, da die Rotation um die Z-Achse in keiner Beziehung zur Erdbeschleunigung steht. Um die Neigung von beiden Achsen zu bestimmen wird letztendlich der Arkussinus von dem Verhältnis des Betrages des Gesamtvektors und des Betrages eines Teilvektors erzeugt.

$$\varphi = \arcsin\frac{|\vec{x}|}{|\vec{g}|}$$

φ bezeichnet den Winkel an der Z-Achse. Der Vektor g ist der Gesamtvektor und der Vektor x ist Teilvektor von g. Betrachtet man das Beispiel in Abbildung 9 so erkennt man den Vektor g, der von dem Punkt A nach C zeigt. Ebenso erkennt man die Teilvektoren x (von A nach B) und z (von B nach C), welche g aufspannen. Stellt man sich jetzt vor, dass der Betrag des Z-Vektors nicht mehr 1 entspricht (1 = 9,81$\frac{m}{s^2}$), so wird impliziert, dass die Drohne geneigt ist. Der fehlende Anteil des Z-Vektors, angenommen die Drohne beschreibt gerade keine vertikale Bewegung, muss auf den X- oder Y-Vektor transferiert worden sein. In der Abbildung wird ein beispielhafter Idealfall dargestellt. Der Teilvektor x und Teilvektor z haben den Betrag 0,5, wodurch ein 45 Grad Winkel zwischen der Z-Achse(blau) und dem Vektor g entsteht. Diese 45 Grad resultieren in einer Rotation an der Y-Achse (Grün), da die Werte an der X-Achse ausgelenkt wurden. Es lässt sich also der Zusammenhang erschließen, dass eine Auslenkung entlang der X-Achse eine Rotation an der Y-Achse bewirkt und eine Auslenkung entlang der Y-Achse eine Rotation an der X-Achse bewirkt.

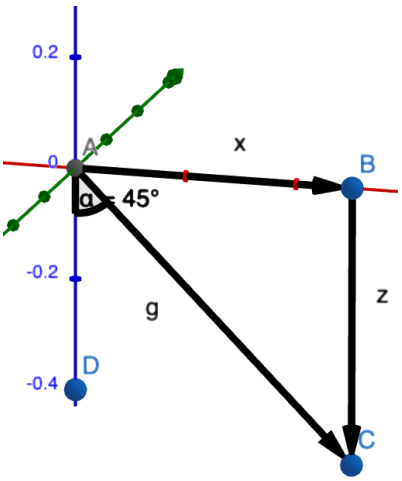


Abbildung 9: Darstellung einer vereinfachten Vektorverteilung, Bild kreiert mit „GeoGebra“

Letztendlich lässt sich die Neigung, relativ zum Raum, vollständig mit dem Linearbeschleunigungssensor berechnen. Der folgende Codeausschnitt zeigt die notwendigen Zeilen.

```
xAchsenNeigung = asin((float)accY/gesamtVektor) * 57.296;
yAchsenNeigung = asin((float)accX/gesamtVektor) * -57.296;
```

Der Arkussinus wird nochmal mit 57,296 multipliziert, um das Ergebnis in Grad umzuwandeln, da die Sinusfunktion der IDE mit Radianten arbeitet. Der Wert ergibt sich aus der Rechnung:

$$57{,}296 = 1 \,/\, (3.142 \,/\, 180)$$

Auch hier nimmt man eine Konstante, bestehend aus dem Kehrbruch von dem Verhältnis π zu 180 (Grad), um Prozessorressourcen zu sparen. Der Gyrosensor ist dennoch wichtig, da ein Accelerometer im Flugbetrieb auch noch einen Bewegungsvektor erzeugt und dadurch die Berechnungen der Neigungen erschwert werden. Darüber hinaus ist der Beschleunigungssensor wesentlich anfälliger für Vibrationen und damit alleine untauglich für den Flug. Der Drehratensensor jedoch misst die Rotationen und kann dadurch, solange sein intrinsisches Koordinatensystem dem extrinsischen entspricht, sehr präzise die aktuelle Lage angeben. Der Accelerometer kann somit für den Gyrosensor einen Bezugspunkt errechnen und ihn dadurch kalibrieren. Diese Eigenschaft wird im nächsten Verarbeitungsschritt ebenfalls von Bedeutung sein.

Fortwährendes Kalibrieren des Gyrosensors[25]

Das anfängliche Setzen des Bezugspunkts ist nicht genug. Die Werte des Gyrosensors erleiden einen sogenannten „Drift“, der dafür sorgt, dass der Gyrosensor über längeren Zeitraum den vorher errechneten Bezugspunkt zum Boden verliert. Dies wird vermutlich durch minimale Unstimmigkeiten in der Integration der Drehrate erzeugt, denn auch der Gyrosensor selbst hat einen gewissen Rhythmus, in dem er Messungen durchführt. So kann es passieren, dass kleine Abweichungen nicht gemessen werden und sich der Fehler über einen längeren Zeitraum aufsummiert. In Ruhelage sind diese Fehler zu vernachlässigen, jedoch sind sie umso signifikanter bei dynamischen Beschleunigungsänderungen. Zwei Optimierungsmöglichkeiten erscheinen sinnvoll. Zum einen könnte man die Messfrequenz deutlich erhöhen, um den Fehler gegen Null zu minimieren, was jedoch einige Komplikationen mit sich führt, da man

[25] (Brokking 2016)

einen leistungsfähigeren Prozessor benutzen muss. Zum anderen könnte man Joop Brokkings Methode verwenden, indem man den Gyrosensor immer wieder durch Kalibrieren reinigt. Hierfür verwendet er einen Komplementärfilter, bei dem das Verhältnis von zwei Werten einen neuen Wert ergibt. Zur Verdeutlichung eine Beispielrechnung:

$$x_1 = 0{,}9 * x_1 + 0{,}1 * x_2$$

In diesem Beispiel wird 90% des ersten Wertes (x_1) verwendet und nur 10% des zweiten Wertes (x_2). Sollte x_2 von x_1 abweichen so kann, bei fortwährender Ausführung der Rechnung, x_1 auf x_2 angeglichen werden. Das prozentuale Verhältnis der Werte gibt an, wie stark der eine Wert den anderen beeinflusst. Wendet man diesen Filter auf den Modelversuch, also die Drohne an, so kann man sagen, dass x_1 der errechnete Gyrosensorwert mit verlorenem Bezugspunkt ist und x_2 der kalibrierte Wert des Accelerometer. Joop Brokkings Methode erachte ich als praktikabler. Außerdem ist noch anzumerken, dass der durch den Accelerometer bestimmte Wert einen besonders kleinen Anteil im Komplementärfilter ausmachen sollte. Das liegt daran, dass der Accelerometerwert sehr stark von den Eigenvibrationen der Drohne verändert wird und sprunghaft ist.

Das Filtern von Vibrationsstörungen[26]

Der MPU-6050 ist ein hochempfindliches Sensormodul, welches kleinste Erschütterungen messen kann. Quadrocopter besitzen vier Motoren, welche in bestimmten Fällen über einen Kilogramm Schub erzeugen können. Um das zu erreichen, sind bei kleinen Propellern, sehr große Drehzahlen nötig, welche zu Schwingungen führen. Der Beschleunigungssensor nimmt

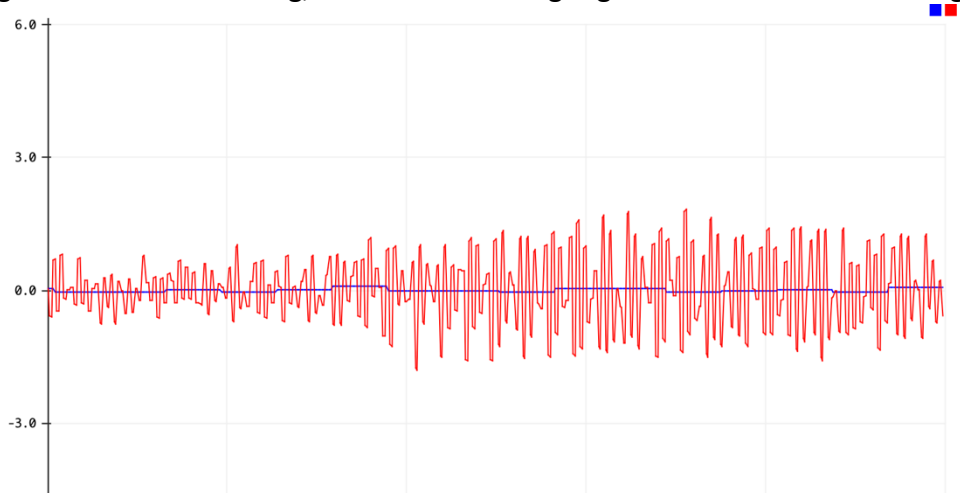


Abbildung 10: Echte Messdaten von dem „MPU-6050" bei laufenden Motoren (mit dem Funktionsplotter der IDE erstellt) Rot: Ungefilterte Messwerte Blau: Durchschnittswert

[26] (Brokking 2016)

dieses Oszillieren auf und es kommt zu einer Verunreinigung der Daten. Zur Veranschaulichung zeige ich hier echte Messdaten einer Achse des Accelerometer in Abbildung 10, bei laufenden Motoren. Klar zu erkennen ist, dass der rote Graph, motordrehzahlabhängig, nicht sinusförmig um einen Mittelwert oszilliert.

Die Vibrationen entstehen vermutlich durch die Unwucht der Rotoren, deswegen gehe ich davon aus, dass die Periodendauer einer Vibrationsschwingung genau der einer Drehung des Rotors entspricht. Um eine Schwingung zu glätten, könnte man nun den Durchschnitt der Werte innerhalb einer Schwingung bestimmen und damit die Abweichung kompensieren. Der Blaue Graph in Abbildung 10 zeigt die Werte mit einem Durchschnitt.
Wie in der Abbildung ersichtlich, oszilliert die blaue Kurve, im Gegensatz zur roten, kaum noch. Nicht erkennbar ist, dass die Werte in einer niedrigeren Frequenz dargestellt sind und damit die Reaktionsfähigkeit der Drohne eingeschränkt wird. Durch die Bildung des Durchschnitts, werden mehrere Werte zu einem zusammengefasst und damit kann die Drohne auch weniger Werte verarbeiten. Die Drehzahl der Motoren ist jedoch so hoch, dass sich die Periodendauer unter zehn Millisekunden bewegt und damit die Reaktionsfähigkeit wenig eingeschränkt werden sollte.

Joop Brokking nutzt hierfür ebenfalls einen Komplementärfilter, in dem er den aktuellen Messwert an den gerade gemessenen Wert ganz leicht angleicht. So könnte die Codezeile aussehen (Variablennamen und Werte sind hier nicht mit der Quelle übereinstimmend).

```
aktuellerWert = aktuellerWert * 0.9 + korrekturWert * 0.1;
```

Damit werden die Ausschläge ebenfalls kompensiert, da nur geringe Anteile des realen Messwerts, der stark durch Vibrationen verunreinigt ist, genommen werden. Es ist jedoch schwerer zu berechnen wie verzögert dieser Wert ist. Man kann davon ausgehen, dass in jedem Fall eine Verzögerung stattfindet, da bei einer Glättung einer Kurve Informationen verloren gehen. Bei einer Glättung durch Errechnung des Durchschnitts ist die Verzögerung klar zu berechnen, da man die Intervalle des jeweiligen Durchschnitts fest definieren kann.

Zusammensetzen der einzelnen Schritte zu einem Ganzen Algorithmus

In den vorherigen Unterkapiteln habe ich die einzelnen Verarbeitungsschritte dargestellt, die essentiell sind, um die Drohne mit auswertbaren Daten zu versorgen. Zuletzt werde ich noch die Reihenfolge dieser Datenverarbeitungsschritte festlegen. Um das zu verdeutlichen, habe ich, wie in Abbildung 3 zu sehen, ein Blockdiagramm erstellt. Der Startprozess führt initiale Befehle aus, wie zum Beispiel das Eichen oder Kalibrieren der Sensoren. Des Weiteren werden gewisse Register zugewiesen, welche über die I^2C Schnittstelle ausgelesen werden können. Eine vertiefende Behandlung dieses umfangreichen Bereichs war im Rahmen dieser Arbeit nicht möglich. Im Hauptprozess wird immer wieder der gleiche Algorithmus ausgeführt (im Diagramm erkennbar an dem Pfeil, der zum Ausgangzustand wieder zurückführt). Je öfter der Prozess pro Sekunde durchgeführt wird, desto genauer können Messdaten erhoben werden. Die Grenzen liegen hier am Sensor und der Geschwindigkeit des Mikroprozessors. Dennoch muss die Aufruffrequenz des Hauptprozesses, möglichst konstant gehalten werden, um die Integrationswerte nicht zu verfälschen.

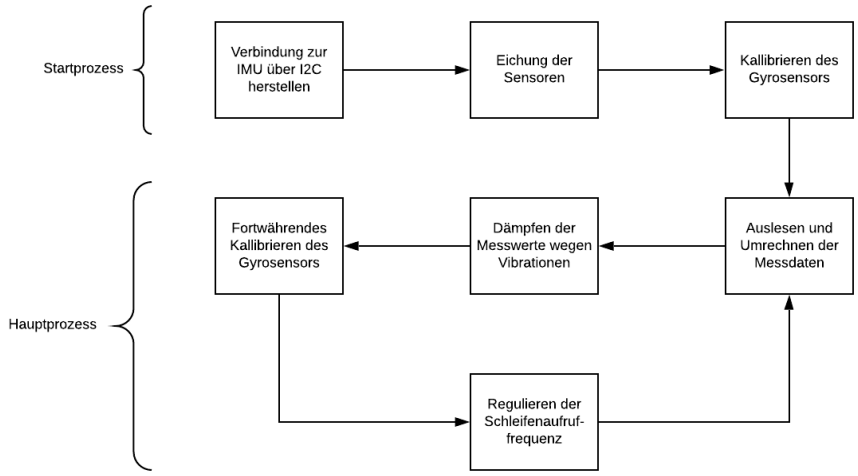


Abbildung 11: Selbsterstelltes Blockdiagramm: Prozessverlauf der Datenverarbeitung

Zusammenfassung

Auch wenn die Logik meines Algorithmus sich nur wenig von Joop Brokkings unterscheidet, so musste ich komplett andere Werte einsetzen. Die Gründe sind lediglich mechanische und logische Einflussfaktoren. Joop Brokking testet seine IMU-Logik mit einer wesentlich größeren Drohne mit größeren Rotorblättern. Meine Drohne ist sehr klein und hat kleinere Rotoren. Kleinere Rotoren benötigen größere Drehzahlen, um den gleichen Schub zu erzeugen,

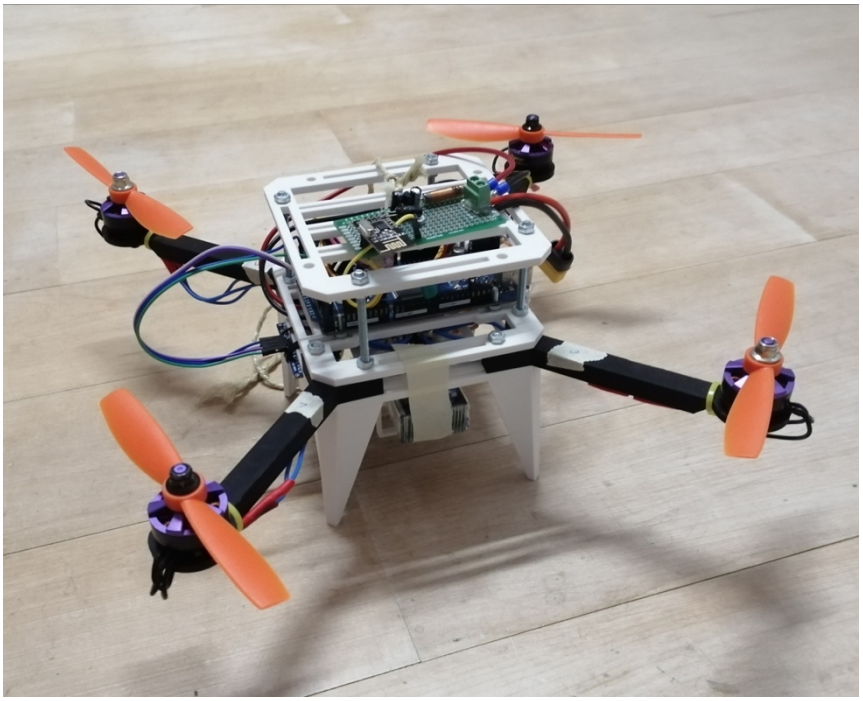
aufgrund des geringeren Auftriebs. Außerdem besitzt mein Versuchsaufbau noch weitere Sensoren und Bausteine. All diese manipulieren den Hauptprozess und verursachen damit zeitliche Aufschübe. Um diese Datenverarbeitung zu optimieren, kann man nur selten ein mathematisches Prinzip anwenden. Meistens müssen die optimalen Werte, wie zum Beispiel die Verteilung innerhalb eines Komplementärfilters, durch viele Versuche und Beobachtungen, also empirisch, gefunden werden. Dies stellt meiner Meinung nach das größte Problem dar. Hinzufügen möchte ich noch, dass es bei meinen Versuchen zu keinem „Gimbal lock“ kam, da die Flugmanöver der Drohne eher simpel gehalten wurden. Das Rechnen mit Quaternionen, was ich als möglichen Lösungsansatz vorgestellt hatte, würde aber meine Arbeit zu umfangreich gestalten. Im Anhang sind verschiedene Versuche mit der Drohne, auf Video festgehalten, abgespeichert.

Darüber hinaus möchte ich noch erwähnen, wofür die Daten aufbereitet werden. Die Drohne braucht genaue zuverlässige Daten über ihre Neigung im Raum, um die richtigen Kontrollschübe zu initialisieren. Sollte die Drohne geneigt sein, so muss sie gegensteuern, um in der Luft zu bleiben. Diese Korrekturen dürfen weder zu stark noch zu schwach sein, weil die Drohne sonst ihren Optimalzustand übertrifft und anfängt zu oszillieren. Hier übernimmt die sogenannte Regeltechnik, welche gewisse Lösungsansätze mit sich bringt, um solche Korrekturen zu optimieren. Der bekannteste ist der PID-Reger. Dieser reguliert über drei verschiedene mathematische Funktionen, den optimalen Schub.[27] Auch hier gibt es gewisse Stellgrößen, die für jede Situation individuell angepasst werden müssen. Die geregelten Schübe müssen dann über eine Kontrollmatrix auf die verschiedenen Motoren verteilt werden. Gerade die Regeltechnik ist in den letzten Jahren im Alltag immer stärker zu sehen. Man sieht Autos, die autonom ihre Spur wechseln. Man sieht Raketen, die ähnlich wie eine Drohne ausbalanciert werden und auf dem Boden autonom Landen. Solche Raketen sind durch ihre Länge, ihren Schwerpunkt und durch ihre eingeschränkte Manövrierfähigkeit äußert schwer zu kontrollieren und dennoch schafft es zum Beispiel die Firma SpaceX eine Trägerrakete auf einer relativ kleinen Plattform im Meer zu landen.[28] Damit solche Dinge funktionieren, brauchen derartige Regler akkurate und verlässliche Messwerte. Sollte man

[27] (Waste 2014)

[28] (SpaceX 2015)

also versuchen einer Drohne den Schwebeflug zu ermöglichen, so ist es unmöglich auf ein genauen und verlässlichen Messalgorithmus zu verzichten.



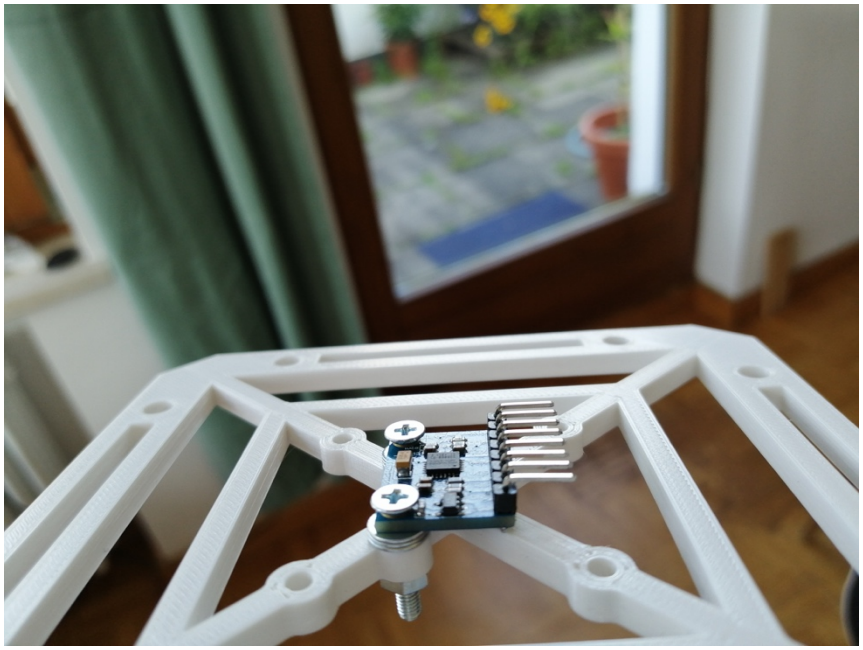


Abbildung 12: Hier ein Bild der Drohne. Strukturgebende Bauteile selbst konzipiert und mit einem 3D-Drucker gedruckt (oben).
Hier ein Bild des „MPU-6050“ auf einem fertigen Breakoutboard. Der „MPU 6050“ ist hier auf der untersten Strukturplatte der Drohne bereits montiert (unten).

Abbildungsverzeichnis

Literaturverzeichnis

Brokking, Joop. 2016. *Brokking.net.* Zugriff am 1. November 2019. http://www.brokking.net/ymfc-al_main.html.

Hüning, Prof. Dr. rer. nat. Felix. 2016. *Sensoren und Sensorschnittstellen.* Leck: De Gruyter Studium.

Hering, Prof. Dr. rer. pol. Dr. rer. nat. Ekbert, Dr. Gert Schönfelder, Prof. Dr. Hartmut Bärwolf, Stefan Basler, Dr. Karl-Ernst Biehl, Thomas Burkhart, Dr. Thomas Engel, et al. 2018. *Sensoren in Wissenschaft und Technik.* 2. Herausgeber: Ekbert Hering und Gert Schönfelder. Heubach und Dresden: Springer Vieweg.

InvenSense Inc. 2013. *MPU-6000 and MPU-6050 Product Specification Revision 3.4.* 19. August.

SpaceX. 2015. *SpaceX.* 10. Juni. Zugriff am 1. November 2019. https://www.spacex.com/news/2013/03/31/reusability-key-making-human-life-multi-planetary.

VectorNav. kein Datum. *VectorNav.* Zugriff am 2. November 2019. https://www.vectornav.com/support/library/imu-and-ins.

Waste. 2014. *rn-wissen.* 13. September. Zugriff am 1. November 2019. https://rn-wissen.de/wiki/index.php/Regelungstechnik.