Bau eines Quadrocopters

Alexander Eichhorn

Betreuung: Marco Bettinaglio

8. Januar 2018

Maturitätsarbeit 2018

MNG Rämibühl

Abstract

Das Ziel dieser Arbeit war, einen Quadrocopter selbst zusammenzubauen und die komplette Software dafür selbst zu programmieren. Der Quadrocopter wurde mit einer Kamera ausgerüstet und durch ein Smartphone steuerbar gemacht.
Der Weg zu diesem Quadrocopter wird in dieser Arbeit beschrieben. Zuerst werden die einzelnen Bauteile und Sensoren erklärt. Die aufgetretenen Probleme mit den Sensoren werden aufgezeigt und Lösungen dafür gegeben. Es wird erläutert, wie sich ein Quadrocopter bewegen kann und wie die Algorithmen funktionieren, die ihn automatisch stabilisieren. Die Verbindung zwischen Copter und Smartphone wird erklärt sowie die App, mit der gesteuert wird.

Inhaltsverzeichnis

1. Einleitung

Ein Quadrocopter ist ein Flugobjekt, das sich mit Hilfe von vier nach oben gerichteten Rotoren in der Luft halten und bewegen kann. Damit besitzt er ähnliche Fähigkeiten wie ein Helikopter. 1907 wurde der erste Quadrocopter von Louis Breguet und Charles Richet gebaut. In ihm konnte eine Person mitfliegen. Er konnte jedoch nur leicht über dem Boden schweben und hatte keine Steuerung. Von da an wurde diese Idee immer wieder für militärische Zwecke aufgegriffen. 1958 wurde mit dem *Curtiss-Wright VZ-7* ein tatsächlich einfach steuerbarer Quadrocopter gebaut. Da er jedoch nicht dem Standard des US-Militärs entsprach, wurde das Projekt nicht weiterverfolgt. [13, 14, 16]

In den letzten Jahren wurde dieses Thema wieder aufgegriffen und die Idee des Quadrocopters als Drohne erfolgreich umgesetzt. Mit der heutigen Technik ist es möglich, Quadrocopter mit nützlicher Flugdauer, Stabilität und Grösse zu bauen, mit einer grossen Vielfalt an Anwendungsmöglichkeiten. Drohnen werden nun zum Beispiel genutzt für Luftfotographie, die früher sehr teuer war. Sie können aber auch genutzt werden, um Waren autonom vom einen zum anderen Ort zu transportieren. Gerade daran wird zur Zeit stark gearbeitet.

Mein Ziel war es, einen Quadrocopter selbst zusammenzubauen und mit einem Smartphone steuerbar zu machen. Die Komponenten der Drohne habe ich einzeln für meine Bedürfnisse ausgesucht und verbaut. Dazu wollte ich die Software, die für die automatische Stabilisierung, Navigation und Verbindung zum Smartphone zuständig ist, komplett alleine schreiben.

2. Bauteile

2.1. Quadrocopter Rahmen und Landegestell

Als Basis zum Bau eines Quadrocopters wird zu allererst ein geeigneter Rahmen benötigt. Dieser sollte ein geringes Gewicht haben, aber gleichzeitig auch starken Belastungen standhalten können. Ein Material, das diese Bedingungen sehr gut erfüllt, sind Kohlenstofffasern. Für einen Quadrocopter in der Grösse, wie ich ihn bauen will, ist es allerdings zu teuer, weswegen ich auf einen Glasfaser-Rahmen zurückgreife. Glasfasern sind zwar etwas weniger stabil, aber deutlich billiger [6]. Die Arme, welche die Motoren mit dem mittleren Teil des Copters verbinden, sind bei meinem Rahmen jedoch aus Plastik, da diese auf Grund von stärkerer Belastung dicker sein müssen.
Da ich wusste, dass ich viele verschiedene Komponenten, wie Sensoren oder Sender und Empfänger, einbauen muss und diese viel Platz in Anspruch nehmen, wählte ich einen Rahmen aus, der in der Mitte viel Platz zur Verfügung hat. Ich wählte schliesslich einen Rahmen mit einer Distanz von 500 mm zwischen zwei gegenüberliegenden Motoren. In der Mitte ist er länglich, was nebst dem vielen Platz auch den Vorteil bringt, dass vorne eine Kamera befestigt werden kann, ohne die Propeller im Bild zu haben.
Der Rahmen wird zudem mit einem Landegestell erweitert, das vor allem den Zweck hat, die am Boden angebrachte Kamera beim Landen und Starten zu schützen.

2.2. Motoren / Propeller

Abbildung 1: Der benutzte Motor ohne Propeller

Als Motoren habe ich das Modell Sunnysky 2216-12 ausgewählt. Dies ist ein bürstenloser Motor, der deswegen über einen elektronischen Drehzahlregler (siehe 2.3) mit Wechselstrom versorgt werden muss. Da eine Batterie immer eine Gleichstromspannung hat, könnte man eigentlich einen normalen Bürstenmotor benutzen, welcher einen internen Polwechsel durchführt. Da bürstenlose Motoren jedoch eine bessere Leistung, eine höhere Geschwindigkeit und eine längere Lebensdauer haben, sind Bürstenmotoren in Quadrocoptern selten [3]. Der ausgewählte Motor hat eine eher langsame Rotationsgeschwindigkeit, die mit 800 KV angegeben ist. Dieser KV-Wert kann durch Multiplikation mit der angelegten Spannung in eine maximale Umdrehungszahl pro Minute (rpm) umgerechnet werden [1]. Beim ausgewählten Motor wäre die maximale Umdrehungszahl ohne Last 13400 rpm. Der Motor hat dafür einen grösseren Drehmoment, was erlaubt, grössere Propeller zu benutzen. Da grössere Propeller mehr Luft nach unten schieben können, hat der Quadrocopter einen stärkeren Auftrieb, was ein höheres Startgewicht ermög-

licht. Daher konnte ich einen 10“ x 4.5“ Propeller mit zwei Blättern verwenden. Mit ” ist dabei die Masseinheit Zoll gemeint. Die 10“ beschreiben den Durchmesser des Propellers, der umgerechnet etwa 25.4 cm beträgt. Viel länger dürfte er nicht sein, da ansonsten zwei nebeneinanderliegende Propeller kollidieren würden. Mit 4.5“ wird die Steigung des Propellers beschrieben. Dieser Wert beschreibt die Distanz, um welche sich der Propeller nach einer Drehung um 360° nach oben bewegte [7]. Dieser Wert ist jedoch nur ein theoretischer Wert, bei dem angenommen wird, dass keine äusseren Kräfte auf den Propeller wirken.

Abbildung 2: Benutzter Propeller

Die gewählte Kombination von Motor und Propeller kann unter vollem Schub etwa 12 N (~ 1.2 kg) an Auftrieb generieren [2, 10]. Das ergibt einen Auftrieb von etwa 48 N (~ 4.8 kg) für alle vier Motoren gemeinsam. Da ein Quadrocopter mit allen vier Motoren auf vollem Schub nicht mehr manövrierfähig ist, ist man auf der sicheren Seite, wenn der Copter bei etwa der Hälfte des maximalen Auftriebs schweben kann. Daher sollte das Gewichts des Quadrocopters nicht über 2.4 kg sein.

2.3. Elektronische Drehzahlregler (ESC)

Der Elektronische Drehzahlregler wird meist mit dem Englischen 'Electronic Speed Controller’ beziehungsweise der Abkürzung ESC bezeichnet. Er ist dafür zuständig, die Geschwindigkeit des Motors zu regulieren.

Da es sich um einen bürstenlosen Motor handelt, benötigt dieser eine Wechselspannung auf 3 Phasen. Je nach Stromfluss zwischen den Phasen entstehen verschiedene Magnetfelder und der drehende Teil des Motors, der mit Magneten bestückt ist, verschiebt sich in eine gewisse Richtung. Das Ziel des ESC ist es also den drehenden Teil des Motors immer in die gleiche Richtung drehen zu lassen. Er muss das Magnetfeld und daher den Stromfluss der Phasen mit dem Rotor mitdrehen. Die Geschwindigkeit kann durch langsamere Änderungen des Stroms zum Beispiel verlangsamt oder durch schnellere Änderungen beschleunigt werden [20].

Die Richtung, in welche sich der Motor dreht, kann durch die Anordnung der 3 Phasen verändert werden. Beim Wechsel zweier Phasen wird die Richtung jeweils umgedreht.

Da ESCs grosse Ströme schalten müssen, generieren sie auch eine gewisse Abwärme. Diese Abwärme müssen die ESCs überstehen können. Deswegen haben ESCs immer ein Maximum an zugelassenem Strom. Für diesen Quadrocopter habe ich mich für vier ESCs mit einem Maximalstrom von je 30 A entschieden. Da laut Motorenhersteller maximal etwa 15 A pro Motor verbraucht werden, ist man damit auf der sicheren Seite.

2.4. Akku

Abbildung 3: Verwendeter Akku beim Quadrocopter

Damit ein Akku zum Fliegen einer Drohne benutzt werden kann, muss er einige Kriterien erfüllen. Da ein Quadrocopter ständig viel Schub produzieren muss, um sein eigenes Gewicht auszugleichen, braucht er entsprechend viel Strom. Die meisten Akkus können nicht den Strom liefern, der gebraucht würde. Dies liegt zum einen daran, dass sie grosse Innenwiderstände haben und daher die Spannung des Akkus drastisch einbrechen würde. Zum anderen kann es zu Schäden am Akku kommen, wenn die chemischen Reaktionen zu schnell ablaufen [17].
Ein perfekter Akku muss aber auch möglichst viel Energie in einem möglichst kleinen Volumen beziehungsweise Gewicht speichern können. Bei Drohnen ist dabei vor allem die gespeicherte Energie pro Gewicht wichtig, da das Gewicht des Akkus genauso ständig in der Luft gehalten werden muss.
Akkus, die diese beiden Kriterien erfüllen können, sind Lithium-Polymer-Akkus. Diese Akku-Technologie hat jedoch auch Nachteile. Wird die Batterie physisch beschädigt oder zu viel Strom verbraucht, kann sie sich entzünden. Eine Beschädigung kann bei einem Absturz des Quadrocopters durchaus passieren, weswegen ich meinen Akku mit einem starken Gewebeband ummantelt habe, was diese Gefahr etwas reduziert.
Ein Lithium-Polymer-Akku verändert, wie andere Batterien auch, je nach Ladungsstand seine Spannung. Die Spannung einer voll aufgeladenen Akkuzelle liegt bei 4.2 Volt. Der Akku hat bei vollständig aufgebrauchter Ladung eine Spannung von 3.0 - 3.5 Volt (je nach Akku). Weitere Entladung wäre bei dieser Spannung zwar möglich, würde aber dazu führen, dass die Akkuzellen über längere Zeit an Kapazität verlieren oder sogar explodieren. Dies ist ein weiteres Problem, das man mit anderen Akku-Technologien nicht hat. Man kann es jedoch lösen, indem man die Spannung während des Fluges misst und bei zu niedriger Spannung den Piloten am Boden warnt. Auch die Maximalspannung von 4.2 Volt sollte nicht überschritten werden, weswegen die Akkus mit speziellen Ladegeräten geladen werden müssen [12].
Da eine Akkuzelle nur eine Nominalspannung von 3.7 Volt hat, benutzt man Akkus, die mehrere Zellen in Serie geschaltet haben. Für meinen Quadrocopter habe ich mich für 4 Zellen in Serie entschieden, so dass die Nominalspannung auf 14.8 Volt vervierfacht wird. Je grösser die Spannung ist, desto schneller können die Motoren drehen. Bei zu grosser Spannung können aber auch die Motoren zerstört werden. Für die eingesetzten Motoren ist die gewählte Spannung gerade adäquat.
Bei der Wahl der Kapazität des Akkus ist das Ziel, eine möglichst lange Flugdauer pro Ladung zu erreichen. Je mehr Kapazität ein Akku besitzt, desto schwerer ist er. Mit mehr Ka-

pazität kann man zwar länger fliegen, aber mit mehr Gewicht dafür weniger lange. Es gibt daher die ideale Kapazität, die die maximale Flugdauer bringen soll. Diese Kapazität herauszufinden ist jedoch sehr schwierig. Deswegen habe ich bei der Wahl vor allem auf das Gewicht geachtet und wollte, dass die maximale Schubkraft der Motoren die Drohne mit dem Akku zusammen etwa zweimal hochheben könnte. Ich habe mich schliesslich für einen Akku mit gemäss Hersteller 5500 mAh und einem Gewicht von etwa 468 g entschieden [5].
Ein weiterer Wert einer Lithium-Polymer-Batterie ist die maximale kontinuierliche Entladungsrate, welche mit der Einheit „C“ beschrieben wird. Den maximalen Strom I_{max} kann man aus dieser Entladungsrate DR und der Kapazität K wie folgt berechnen:

$$I_{max} = DR * K$$

Da die gewählten Motoren nach Herstellerangaben einen Maximalstrom pro Motor von etwa 15 A benötigen, braucht die Batterie mindestens folgende Entladungsrate:

$$DR = \frac{I_{max}}{K} = \frac{4 * 15A}{5.5Ah} = 10.9C$$

Der Hersteller gibt an, dass die gekaufte Batterie eine maximale Entladungsrate von 45 C hat, was daher völlig reicht.

3. Kamera & Gimbal

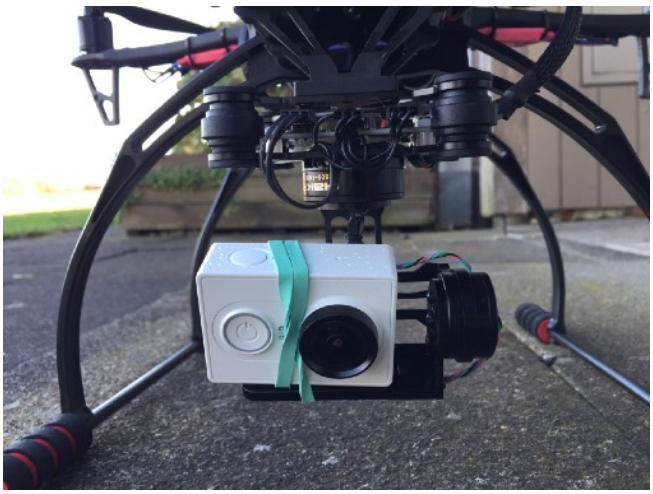


Abbildung 4: Gimbalsystem mit angebrachter Kamera

Der Quadrocopter soll ebenfalls die Möglichkeit erhalten, von seiner Position aus zu filmen. Dafür verwende ich eine Actionkamera. Actionkameras sind optimal dafür, da sie sehr leicht sind und meistens auch die Funktion einer Liveausgabe des Bildes haben. Hier verwende ich eine Kamera namens ‚Xiaomi Yi' eines chinesischen Herstellers. Diese liefert trotz des günstigen Preises eine gute Bildqualität und eine zuverlässige Bildübertragung. Zudem kann man relativ einfach eine Fernauslösung von Fotos oder Videos veranlassen.

Würde die Kamera nun einfach an die Drohne montiert werden, wären Verwacklungen garantiert. Sobald sich der Copter bewegen würde, würde man dies auch auf dem Bild sehen. Um dieses Problem zu umgehen und schöne, ruhige Kameraaufnahmen zu ermöglichen, verwende ich ein sogenanntes Gimbalsystem. Dieses ist durch vier Gummistücke vor feineren Vibrationen geschützt und besteht ansonsten hauptsächlich aus drei Motoren. Diese versuchen, aktiv Bewegungen auszugleichen, indem die Motoren so drehen, dass die Kamera immer waagerecht bleibt. Jeder Motor ist dabei für eine Achse zuständig. Ein Controller ganz oben berechnet dabei die Motorenbewegungen mit Hilfe von Sensoren an der Kamera und beim Controller selber.
Ausserdem lässt sich die gewünschte Position der Kamera auch verändern. Der Flugcontroller kann so zum Beispiel steuern, ob die Kamera direkt nach unten filmen soll oder geradeaus.

4. Sensoren

4.1. Lage im Raum

Gyroskop

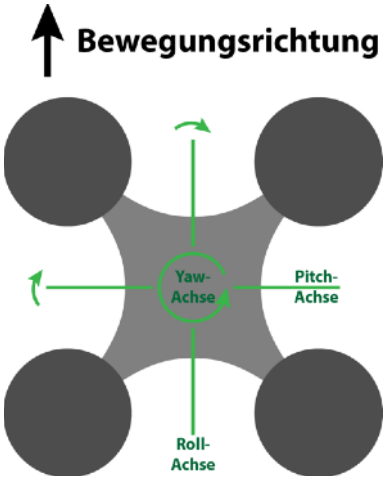


Abbildung 5: Die Namen der drei Achsen aufgezeichnet, an einem Quadrocopter.

Ein Gyroskop ist dafür da, Winkelgeschwindigkeiten zu messen. Dies tut es auf 3 Achsen, welche man auch Pitch, Roll und Yaw nennt. Die Pitch-Achse neigt den Quadrocopter nach vorne und hinten, die Roll-Achse zur Seite und die Yaw-Achse horizontal.
Diese Werte können später benutzt werden, um die Veränderungen der Lage im Raum der Drohne zu überwachen. Den absoluten Winkel α zum Zeitpunkt t könnte man in der Theorie ebenfalls mit bekanntem Anfangswinkel α_0 herausfinden. Eine Funktion $\omega(t)$ würde dabei die Winkelgeschwindigkeit zum Zeitpunkt t ausgeben:

$$\alpha(t) = \int_0^t \omega(\tau)\,\mathrm{d}\tau \; + \; \alpha_0$$

In der Praxis ist dies jedoch nur begrenzt möglich. Da die Winkelgeschwindigkeiten nur 400 Mal pro Sekunde gemessen werden und der Sensor selbst einen kleinen Fehler in den gemessenen Werten hat, addieren sich die Fehler in der Integration immer weiter auf, was dazu führt, dass der absolute Winkel immer weiter vom tatsächlichen Wert abkommt.

Beschleunigungssensor

Ein Beschleunigungsmesser wird verwendet, um die Beschleunigungen auf 3 Achsen zu messen. Dadurch kann ein Beschleunigungsvektor im Raum erzeugt werden. Bei absolutem Stillstand wird ausschliesslich die Erdbeschleunigung gemessen. Der Betrag des Vektors ist dabei also genau $1\ g$ beziehungsweise $9.81\ m/s^2$. Der Beschleunigungssensor wird in diesem Quadrocopter ausschliesslich dafür benutzt, den Winkel zum Beschleunigungsvektor der Erde zu finden und dadurch seine Lage zu berechnen. Die Erdbeschleunigung lässt sich jedoch nicht einfach von anderen Beschleunigungen unterscheiden. Da die Erdbeschleunigung oft den grössten Teil des Vektors ausmacht, geht man näherungsweise davon aus, dass die Gesamtbeschleunigung auch die Erdbeschleunigung ist.
Dadurch kann man nun die Winkel zu einer Achse bestimmen. In diesem Beispiel werden die Winkel α und β zur z-Achse mit den Beschleunigungswerten ax, ay und az berechnet:

$$\alpha = \arctan2(-ax, \sqrt{ay^2 + az^2})$$
$$\beta = \arctan2(ay, az)$$

Die Funktion $\arctan2(y, x)$ *entspricht dabei der Funktion* $\arctan\left(\frac{y}{x}\right)$*, wobei jedoch dank der Berücksichtigung der Vorzeichen beider Variablen ein Winkel von -180° bis 180° berechnet werden kann. Bei der normalen arctan-Funktion geht diese Information verloren und das Resultat beschränkt sich auf den Bereich von -90° bis 90°.*

Das Problem bei der Berechnung des Winkels über den Beschleunigungssensor wird sein, dass bei grossen Beschleunigungen des Quadrocopters selbst die Winkel komplett falsch sind. Bei kleineren Beschleunigungen stimmt der Winkel schon relativ genau. Im Gegensatz zum Gyroskop kann durch den Beschleunigungsmesser der absolute Winkel ermittelt werden.

Magnetometer

Ein Magnetometer misst die magnetische Flussdichte auf 3 Achsen. Werden diese Werte als ein Vektor angesehen, zeigt dieser die Richtung und Stärke der Magnetfeldlinien am gemessenen Ort. Daraus kann die Inklination sowie die Deklination berechnet werden. Die Inklination beschreibt dabei den Winkel, mit dem die Magnetfeldlinien relativ zur Horizontalen auftreffen. Dieser hängt vor allem von der geographischen Breite ab. In Zürich beträgt sie zum Beispiel 63.3° [21]. Die Deklination allerdings beschreibt, in welche Richtung die Magnetfeldlinien auf der Horizontalen zeigen. Diese zeigen theoretisch immer Richtung Norden. Durch die Werte der x- und y-Achse (mx, my) kann die Deklination γ, also der Winkel zum magnetischen Norden, berechnet werden:

$$\gamma = \arctan2(mx, my)$$

Beim Berechnen dieses Winkel geht man davon aus, dass der Sensor ausschliesslich das Erdmagnetfeld misst. In Wirklichkeit gibt es allerdings noch viele weitere Objekte in unserer Umwelt, die magnetische Felder erzeugen. So können Hochspannungsleitungen die Berechnung stark verfälschen. Ausserdem fliessen auch innerhalb des Quadrocopters starke Ströme, welche innerhalb von Millisekunden variieren. Dadurch entstehen ebenfalls störende Magnetfelder. Die einzige Lösung dafür ist, den Magnetometer möglichst weit weg von Starkstromkabeln zu platzieren und nicht zu nahe an Hochspannungsleitungen zu fliegen.

Kombinieren der Sensoren

Um die Lage im Raum durch die drei Winkel α, β und γ zu beschreiben, müssen die Informationen der drei Sensoren kombiniert werden. Die beiden Winkel zur horizontalen Ebene, α und β, sind später für die Stabilisation des Quadrocopters am wichtigsten. Diese werden Pitch und Roll genannt. Informationen zu diesen Winkeln werden sowohl vom Beschleunigungssensor sowie vom Gyroskop gegeben. Das Gyroskop misst dabei einzelne Datenpunkte zwar sehr genau. Durch dessen Integration werden die Winkel allerdings im Laufe der Zeit immer ungenauer. Der Beschleunigungssensor hingegen misst zu jedem Zeitpunkt die absoluten Winkel, hat allerdings einen grossen Messfehler, weil andere Beschleunigungen stören können. Die Lösung dieses Problems ist ein sogenannter Komplementärfilter:

$$\alpha(t) = C * (\alpha(t - \Delta t) + \omega_{\alpha}(t)\Delta t) \; + \; (1 - C) * \alpha_{Beschl.}(t)$$

C ist dabei eine gewählte Konstante, die mit der Bedingung $0 \leq C \leq 1$ gewählt werden kann. In diesem Beispiel muss $0.95 \leq C < 1$ sein, damit der Filter seinen Zweck erfüllt. Der Filter addiert die Veränderung des Winkels in der Zeit zwischen den zwei Messpunkten ($\omega_{\alpha}(t) * \Delta t$) zum vorherigen Winkel. Dieser Winkel ist kurzfristig sehr genau, weswegen dieser den grössten Teil des Endresultats ausmacht. Der ungenaue Winkel des Beschleunigungssensor $\alpha_{Beschl.}(t)$ wird mit einem sehr kleinen Gewicht ebenfalls dazugerechnet. Über längere Zeit steigt jedoch insgesamt die Bedeutung des Beschleunigungssensors und verhindert so den Anstieg des Fehlers wegen der Integration. In der Praxis wurde für C ein Wert von 0.998 eingesetzt.
Verbessert werden kann dieser Filter, indem zusätzlich überprüft wird, ob der Beschleunigungssensor ein einigermassen brauchbares Resultat liefert. Das Resultat ist am besten, wenn nur die Erdbeschleunigung gemessen wird, und die gesamte Beschleunigung $a_{insg.}$ ungefähr beim Betrag davon (1 g) liegt. Um das Resultat nun zu verbessern, kann bei Nichterfüllung der Bedingung $0.9g \leq a_{insg.} \leq 1.1g$ das $C = 1$ gesetzt werden. Dadurch hat der Beschleunigungssensor in diesem Zeitpunkt keinen Einfluss auf den Winkel. Es wird dadurch vor allem verhindert, dass starke Beschleunigungen über längere Zeit zu einem grossen Winkelfehler führen können.

Vibrationen

Der gebaute Quadrocopter hatte anfangs starke Probleme wegen Vibrationen. Diese werden von den schnell drehenden Motoren erzeugt. Ist eine Seite des drehenden Elements des Motors leicht schwerer als die andere, führt das beim Drehen zu Vibrationen. Der Propeller hat dabei meist das grösste Ungleichgewicht. Dieser kann zwar ausbalanciert werden, eine perfekte Gewichtsverteilung ist aber fast unmöglich. Vibrationen entstehen aber auch dadurch, dass die vom Propeller nach unten geblasene Luft auf den Arm, der den Motor mit dem Rest des Copters zusammenhält, trifft. Dies allerdings nicht immer mit einer konstanten Kraft, da das Rotorblatt nur zweimal pro Umdrehung wirklich über dem Arm steht.

Diese entstanden Vibrationen sind dann auch in den Messdaten des Gyroskops und Beschleunigungssensors vorhanden. Bei diesem Quadrocopter waren die Vibrationen so stark, dass sich die gemessene Beschleunigung beim Drehen der Motoren zwischen 0 g und 4 g auf und ab bewegte. Unter diesen Bedingungen konnten die Messungen nicht weiterverarbeitet werden. Zur Lösung des Problems musste ich die Messdaten filtern. Dafür implementierte ich einen Low-Pass-Filter des Filtersystems namens ‚Biquad Filter'. Ein Low-Pass-Filter ist ein Filter, der die Daten von den hohe Frequenzen, also den Vibrationen, befreit. Der Nachteil eines Filters ist jedoch, dass dieser zu einer leichten Verzögerung führt. Je stärker der Filter eingestellt wird, desto grösser wird die Verzögerung. Ist ein Filter also zu stark, reagiert der Quadrocopter in gewissen Situationen viel zu langsam. Mit den richtigen Einstellungen des Filters verbesserte sich die Situation stark.

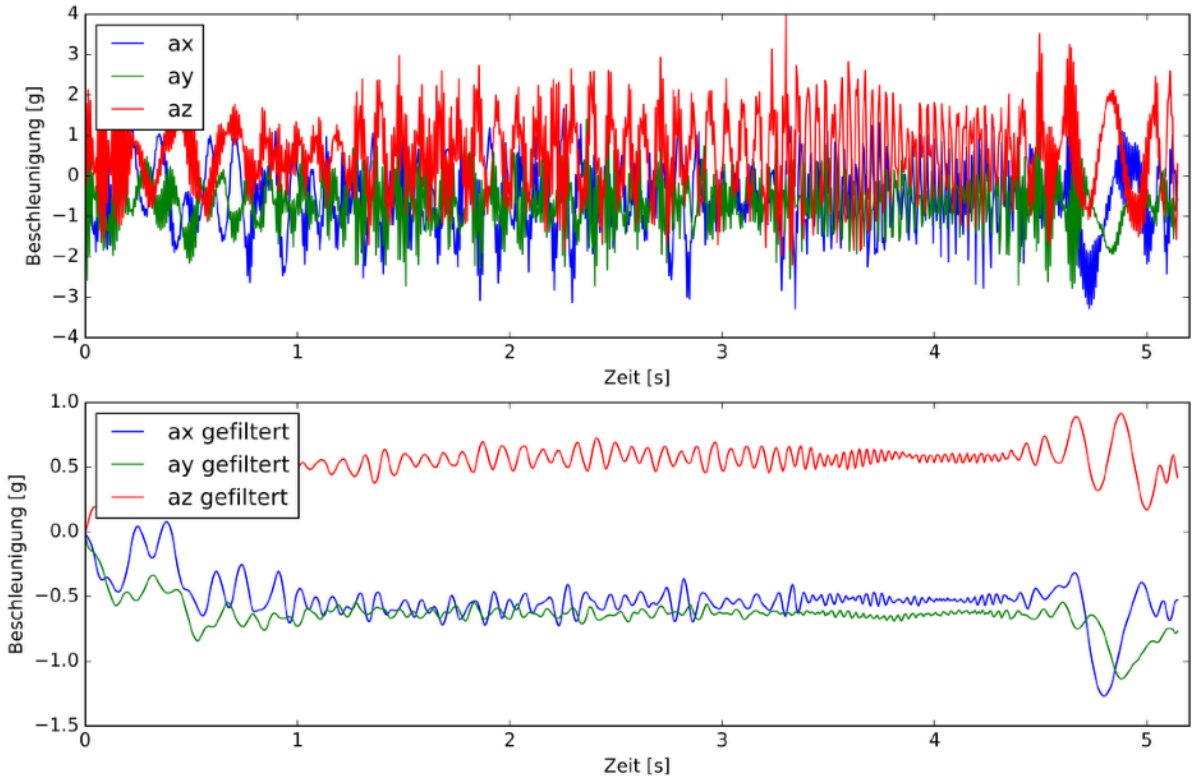


Abbildung 6: Bei einem kurzen Test mit laufenden Motoren aufgezeichnete Beschleunigungsdaten auf drei Achsen. Oben sind die rohen Daten zu sehen. Unten wurden diese gefiltert.

Als weitere Massnahme zur Dämpfung der gemessenen Vibrationen, setzte ich einen zweiten Sensor auf eine Platte, die mit den restlichen Bauteilen nur über vier Gumminoppen verbunden ist. Dies soll dazu führen, dass Vibrationen am Quadrocopter die Platte und den Sensor nur in etwas gedämpfter Form erreichen. Tatsächlich half diese Vibrationsdämpfung ebenfalls, leicht bessere Messdaten zu erhalten.

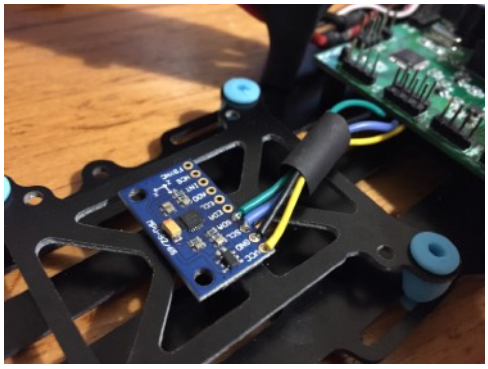


Abbildung 7: Externer Sensor auf einer vibrationsgedämpften Platte

4.2. Höhe

Barometer

Ein Barometer ist ein Messgerät, das den absoluten Luftdruck messen kann. Da der Luftdruck bei niedriger Höhe grösser ist als bei grosser, kann man mit dem gemessen Druck auf die relative Höhe schliessen. Misst man den Luftdruck p_0 auf einer Höhe h_0 und in einer zweiten Höhe h den Druck p, kann der Höhenunterschied nach folgendem Modell berechnet werden [11]:

$$\Delta h = h - h_0 = \frac{T}{0.0065} * \left(1 - \left(\frac{p}{p_0} \right)^{\frac{1}{5.255}} \right)$$

T ist dabei die Temperatur auf Höhe h_0. Dafür kann man als Annäherung eine Temperatur wie 15°C beziehungsweise 288.15 K einsetzen und kann damit die Höhe ausschliesslich mit dem Druck berechnen:

$$\Delta h \approx 44330 * \left(1 - \left(\frac{p}{p_0} \right)^{\frac{1}{5.255}} \right)$$

In der Praxis ist es am sinnvollsten, h_0 als Höhe, bei der die Drohne startet, zu wählen. Dadurch kann der entsprechende Luftdruck p_0 direkt beim Starten gemessen und gespeichert werden.

Der Barometer hat allerdings auch seine Probleme. Weht ein Wind, entstehen zum Teil innerhalb kurzer Zeit starke Druckunterschiede, welche von der Software als Höhenveränderung aufgefasst werden. Zum Schutz vor dem Wind kann der Sensor mit Schaumstoff um-

hüllt werden. Dieser Wind wird jedoch auch von den vier Propellern selbst produziert. Dieser ist vor allem sehr nahe am Boden problematisch, da der durch die Propeller verursachte hohe Druck nur zur Seite, statt auch nach unten, ausweichen kann, so dass der Fehler zu gross wird. Daher ist der Barometer in Bodennähe, also bei einer Höhe von weniger als einem Meter, nicht brauchbar.

Ultraschallsensor

Abbildung 8: Verwendeter Ultraschallsensor MB1240 [9]

Ein zweiter Weg, die Höhe des Quadrocopters zu messen, ist die Messung über einen Ultraschallsensor (auch genannt: Sonar). Dieser sendet für eine sehr kurze Zeit Schallwellen stark gerichtet mit einer Frequenz von etwa 42 kHz aus, was für Menschen nicht hörbar ist [9]. Wird das ausgesendete Signal von einem Objekt reflektiert, wird es vom Sensor empfangen. Dabei wird die Zeit zwischen Aussenden und Empfangen Δt gemessen. Es ist die Zeit, in der die Schallwellen die Distanz zu diesem Objekt zweimal zurückgelegt haben:

$$2d = v * \Delta t = c_{Luft} * \Delta t$$

c_{Luft} ist dabei die Geschwindigkeit, mit der sich Schall in der Luft ausbreitet. Dieser Wert hängt vom Druck und der Temperatur der Luft ab, kann aber in der Praxis als Konstante betrachtet werden, da die Druck- und Temperaturschwankungen in diesem Anwendungsfall den Wert nur leicht ändern lassen. Für c_{Luft} kann also der Wert bei 20°C und Normaldruck verwendet werden [4]:

$$2d = 344\ m/s * \Delta t$$

$$d = \frac{344\ m/s * \Delta t}{2}$$

Diese Art von Höhenbestimmung ist deutlich genauer als der Barometer, hat aber den Nachteil, dass sie nur in Bodennähe durchgeführt werden kann.
Wegen der starken, durch die Propeller verursachten Turbulenzen sind normale Ultraschallsensoren aber oft nicht fähig, die Distanz korrekt zu messen [8]. Auch ich hatte zuerst normale Ultraschallsensoren verwendet. Diese haben meistens jedoch nur zufällige Werte gemessen. Es gibt allerdings dafür optimierte Sensoren, die dementsprechend deutlich mehr kosten. Doch auch der speziell dafür ausgelegte Ultraschallsensor hat nur etwa jedes vierte Mal einen korrekten Wert gemessen, sobald der Quadrocopter in der Luft war. Den Grund

dafür konnte ich nicht herausfinden. Das machte den Ultraschallsensor für die Höhenberechnung praktisch wertlos.

Um die jeweiligen Nachteile zu kompensieren, werden die beiden Messarten kombiniert. Kann die Distanz zum Boden durch den Ultraschallsensor gemessen werden, wird diese Distanz als Höhe verwendet. Ansonsten wird die ungenauere Höhe des Barometers verwendet. Zudem wird bei einer erfolgreichen Messung des Ultraschallsensors der Barometer mit dieser Höhe neu kalibriert.

5. Hauptplatine & Flugcontroller

Mit zunehmender Komplexität der Planung des Quadrocopters mussten immer mehr Elektronikkomponenten miteinander verbunden werden. Eine Leiterplatte sollte ein Durcheinander verhindern und auch viel Platz sparen. Daher habe ich einen entsprechenden Schaltplan in einem Onlineprogramm namens ‚EasyEDA' entworfen und mehrere Male überprüft und verbessert. Später konnte mit dessen Hilfe eine Platine im Programm entworfen werden. Dabei hatte ich bereits im Kopf, wie der Copter ungefähr aufgebaut werden sollte und konnte daher Anschlüsse zu externen Komponenten an den richtigen Rand setzen. Die Schwierigkeit dabei war vor allem, die vielen Leiterbahnen so zu verlegen, dass sie sich auf diesem kleinen Platz nicht überkreuzen. Mit der Verwendung einer Platine mit zwei Schichten waren alle Verbindungen schliesslich möglich. Die entworfene Platine konnte ich danach extern produzieren lassen. Sämtliche Elektronikbauteile mussten allerdings von Hand angelötet werden. Da diese zum Teil sehr klein waren, musste ich dafür eine Heissluftpistole verwenden. Diese erwärmt Luft auf 300-400 °C, bläst diese hinaus und schmilzt dadurch den Lötzinn. Dabei kam es immer wieder zu Überbrückungen zweier Pins an einem Chip. Besonders der kombinierte Chip mit Beschleunigungsmesser, Gyroskop und Magnetfeldmesser machte Schwierigkeiten, da er speziell kleine Pins hat. Er ist unter anderem als Sensor in Smartphones gedacht, wo die Grösse eine wichtige Rolle spielt.

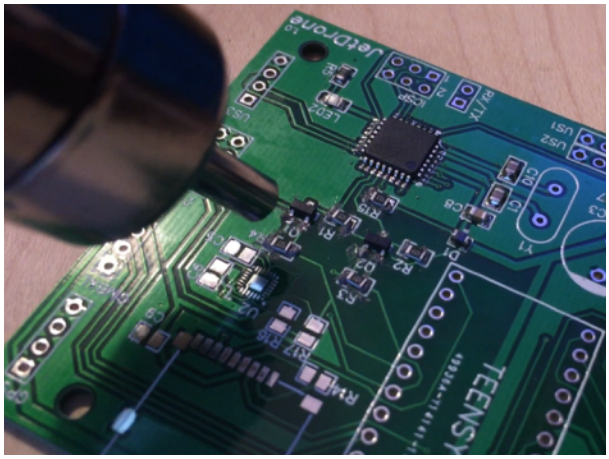


Abbildung 9: Anlöten von SMD-Komponenten mit einer Heissluftpistole

Die Platine wird direkt an den Akku angeschlossen, der über eine Spannung zwischen 12.8 und 16.8 V verfügt. Da für die meisten Bauteile diese Spannung zu gross ist, verfügt die Platine über einen Spannungswandler, der jede höhere Spannung auf 5 Volt reduziert. Dabei wurde eine Version des Spannungswandlers gewählt, die sehr energieeffizient ist. Da auch hier teilweise viel Strom verbraucht wird, wird so eine Überhitzung des Spannungswandlers verhindert. Der Flugcontroller reduziert diese Spannung später noch weiter auf 3.3 V, um damit den grössten Teil der Sensoren zu betreiben, die keine höhere Spannung akzeptieren.

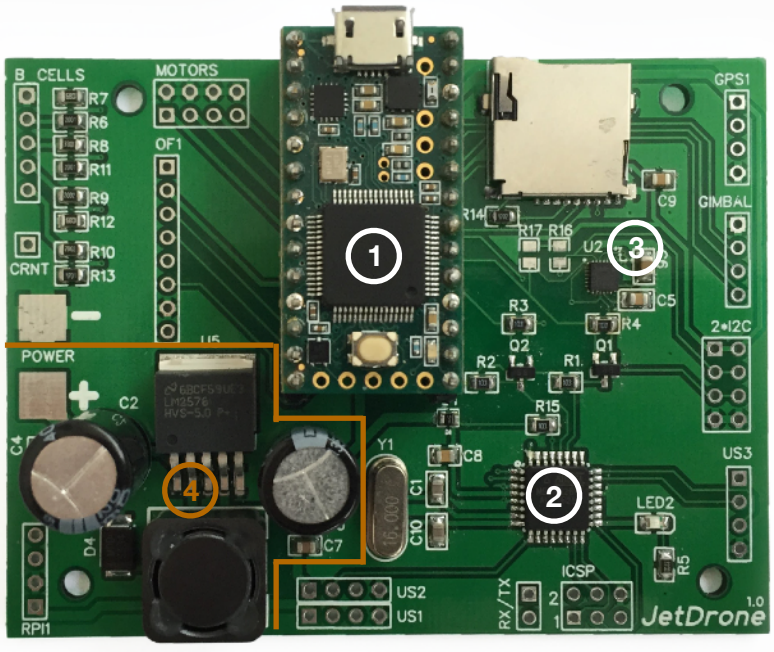
Das Herzstück der Platine und auch der Drohne ist ein Mikrocontroller namens Teensy[1]. Dieser übernimmt die Funktion des sogenannten Flugcontrollers, der sämtliche Sensoren ausliest und daraus die Signale an die Motoren berechnet. Er ist mit einem Chip namens

[1] ein Teensy 3.2 mit einem Cortex-M4 Prozessor wird hier verwendet

MPU9250 direkt verbunden, der Beschleunigung, Gyroskop und Magnetfeld misst. Mit diesem wird über das Protokoll I2C kommuniziert. Es stehen ausserdem zwei weitere Anschlüsse für externe Sensoren über I2C zur Verfügung. Diese werden von einem zweiten MPU9250 und dem Barometer genutzt. Zudem gibt es Anschlüsse für alle Zellen des Akkus, wodurch der Flugcontroller durch Spannungsmessungen den Ladungsstand des Akkus ermitteln kann. Anschlüsse für ein externes GPS Modul sowie einer sogenannten Optical-Flow-Kamera sind ebenso vorhanden, werden aber zurzeit nicht benutzt. Auf der Platine mit dem Flugcontroller verbunden befindet sich auch ein SD-Karten-Leser, der dazu dient, Flugdaten aufzuzeichnen und später einfach auszuwerten. Natürlich sind auch Anschlüsse für die vier Motoren und den Gimbal vorhanden. Hierfür wird ein sogenanntes Pulsweitenmodulationssignal verwendet, das durch die Länge des Signals die Geschwindigkeit der Motoren oder die Position des Gimbals bestimmt.

Sämtliche Sensoren, die an den Flugcontroller angeschlossen sind, laufen - wie er selbst auch - auf 3.3 Volt. Es befindet sich darunter ein zweiter Mikrocontroller[2], der im Unterschied zum ersten mit einer Spannung von 5 V läuft und deutlich langsamer ist. Er ist dafür zuständig, die Ultraschallsensoren zu verwalten. Die Ultraschallsensoren haben bei 5 V die grösste Reichweite. Eine Messung blockiert sämtliche anderen Vorgänge des Mikrocontrollers. Beim Hauptmikrocontroller würde diese lange Wartezeit die sonstigen Berechnungen stören. Daher misst der zweite Mikrocontroller und sendet die gemessenen Daten verzögert an den Flugcontroller weiter, damit dieser nicht blockiert wird. Um die Kommunikation zwischen einem 3.3-V- und einem 5-V-Mikrocontroller zu ermöglichen, ist eine kleine Schaltung dazwischen nötig, die die Signale den Spannungsebenen anpasst.

Ein Raspberry Pi, der später für die Verbindung zur Fernsteuerung zuständig ist, wird ebenso angeschlossen. Dieser wird direkt über die Platine mit Strom versorgt und kann gleichzeitig mit dem Flugcontroller über eine serielle Schnittstelle kommunizieren.



1 **Flugcontroller (Teensy)**

2 **Zweiter Mikrocontroller für Ultraschallsensoren**

3 **interner Beschleunigungssensor, Gyroskop und Kompass**

4 **Spannungswandler auf 5 V**

Abbildung 10: Die Hauptplatine mit allen Komponenten angelötet

6. Bewegungsmöglichkeiten

Für seine Flugfähigkeit hat der Quadrocopter ausschliesslich seine vier Motoren zur Verfügung. Diese müssen einerseits ständig genug Auftrieb generieren, um den Quadrocopter über Boden zu halten, gleichzeitig aber auch diverse Flugmanöver ausführen. Anders als bei einem Flugzeug gibt es zusätzlich keine Klappen, die Strömungen verändern können. Sämtliche Manöver werden ausschliesslich durch verschieden stark drehende Motoren ausgeführt.

Damit ein Quadrocopter sich nicht ungewollt um die eigene Achse dreht, müssen jeweils ein Paar Motoren in die eine Richtung drehen, während die zwei anderen in die andere Richtung drehen. Dadurch wird das Drehmoment gegenseitig ausgelöscht. Diese Motoren liegen sich dabei gegenüber. Man kann die Drohne jedoch auch bewusst um die eigene Achse drehen lassen, indem man das eine Motorenpaar schneller und das andere langsamer drehen lässt. Laufen alle Motoren zum Beispiel auf 50 % der Höchstgeschwindigkeit, kann man die Motoren 2 und 4 (siehe Abbildung) mit 55 % laufen lassen und Motoren 1 und 3 mit nur 45 %. Der Auftrieb nach oben bleibt bei 50 %, die Drohne dreht sich jedoch im Uhrzeigersinn um die eigene Achse, also in die andere Richtung als die schneller drehenden Motoren.

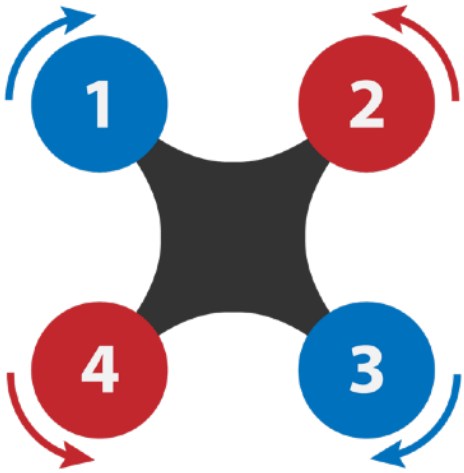


Abbildung 11: Illustration einer Drohne mit vier Motoren, die jeweils mit einer Zahl markiert sind.

Um den Quadrocopter zu bewegen, muss er in die gewünschte Richtung geneigt werden. Je grösser die Neigung, desto mehr beschleunigt der Copter in diese Richtung. Durch die Neigung wird der Schub des Quadrocopters nicht mehr ausschliesslich genutzt, um sich über Boden zu halten, sondern auch, um seitlich zu beschleunigen (siehe Abbildung).

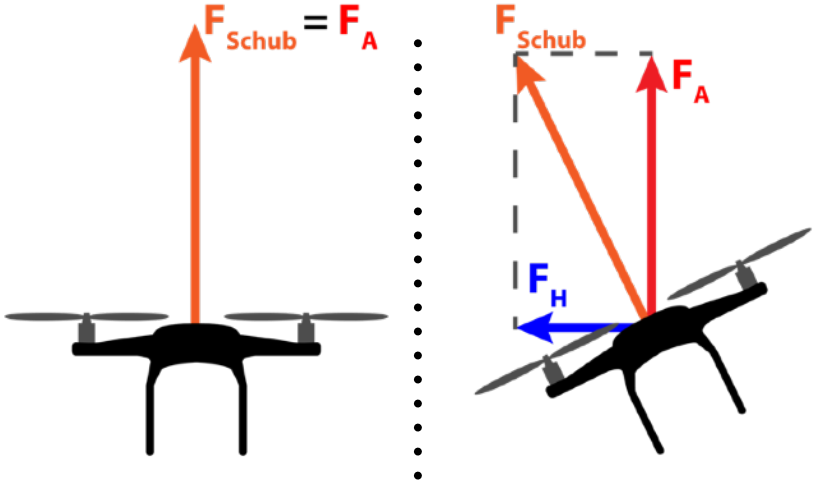


Abbildung 12: Beide Quadrocopter geben den selben Schub (F_{Schub})
Links steht der Quadrocopter horizontal. Der Auftrieb F_A ist daher gleich gross.
Rechts ist er geneigt und F_{Schub} teilt sich in Auftrieb und eine horizontale Kraft F_H auf.

Eine Neigung des Quadrocopters kann durch stärker und weniger stark drehende Motoren erreicht werden. Drehen zum Beispiel die beiden Motoren auf der linken Seite stärker als die auf der rechten Seite, hebt sich der linke Teil daher relativ zum rechten Teil an. Ist der Copter bereits geneigt, bleibt diese Neigung bei gleichem Schub über alle Motoren grundsätzlich bestehen. Daher ist es Aufgabe der Software, die Drohne in einer bestimmten Neigung zu halten.

Zusätzlich muss beachtet werden, dass bei zunehmender Neigung der Auftrieb im Verhältnis zum Schub abnimmt. Deswegen wird über die Software der Schub der Neigung α entsprechend korrigiert.

$$F_{Schub} = \frac{F_A}{\cos(\alpha)}$$

7. Steuerung

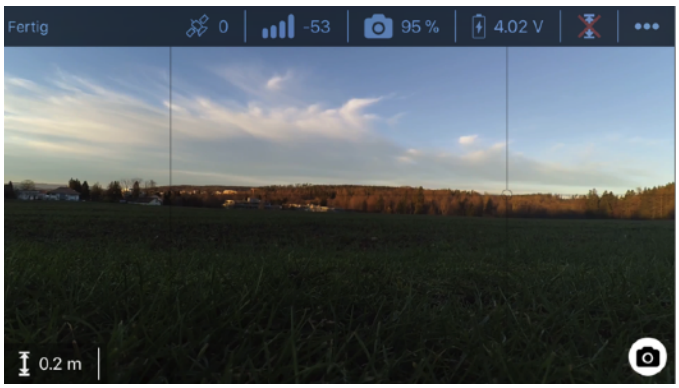


Abbildung 13: Screenshot der Steuerungsapp mit eingeblendeten Steuerungshilfen und einem Livebild der Kamera

Der gebaute Quadrocopter wird ausschliesslich mit einer eigens dafür kreierten Smartphone-App gesteuert. Die Bewegungen des Quadrocopters können dabei über die Position zweier Finger gesteuert werden. Diese verhalten sich ähnlich wie die Sticks herkömmlicher Fernbedienungen im Modellbau. Auf der rechten Seite steuert man die Neigung beziehungsweise die Flugrichtung des Quadrocopters. Die Stärke der Neigung - und daher auch die horizontale Fluggeschwindigkeit - wird durch die Distanz des Fingers zum Mittelpunkt ermittelt. Je weiter weg, desto schneller ist die gewünschte Fluggeschwindigkeit. Auf der linken Seite wird auf der vertikalen Achse der Schub der Motoren gesteuert. Je weiter oben, desto mehr Schub wird gegeben. Je nach Flugmodus, wird hier aber auch direkt die gewünschte Steig- und Sinkgeschwindigkeit vorgegeben. In der Mitte versucht der Quadrocopter bei diesem Modus daher, auf der selben Höhe zu bleiben. Auf der horizontalen Achse wird dabei die Drehung des Copters um sich selbst gesteuert. Bei Auflegen des Fingers werden Steuerungshilfen angezeigt, damit die Mitte der jeweiligen Steuerung ersichtlich ist.

Ein Problem einer Steuerung über Toucheingabe ist, dass sich die Finger auf dem Bildschirm auf keinem genauen Punkt befinden. Daher können feine Steuerungsbefehle nicht wirklich erkannt werden. Da sich dieser Quadrocopter jedoch automatisch stabilisiert, kann man auf feine Eingaben auch verzichten.
Ein Smartphonebildschirm kann ausserdem nicht dasselbe haptische Feedback wie eine gewöhnliche Fernbedienung geben. Gewisse Eingaben können also nicht komplett blind durchgeführt werden. Daher verfügt die Steuerung der Ausrichtung der Drohne über einen kleinen Bereich um den Nullpunkt, der keine Eingabe auslöst. Das verhindert, dass ungenaue Schubbefehle in der linken Hälfte auch die Ausrichtung der Drohne versehentlich beeinflussen.

Über die App kann auch der Winkel der Kamera verändert werden. Durch Halten des Kamerasymbols in der rechten unteren Ecke, setzt die Kamera die momentane Neigung des Smartphones um. Beim Loslassen wird diese Position gespeichert. Diese Art der Steuerung verhindert, dass weitere Steuerungshilfen über dem Kamerabild angezeigt werden müssen und dieses so noch mehr verdecken.

Die App dient nicht nur als Fernbedienung, sondern zeigt auch aktuelle Informationen über die Drohne an. So wird der Akkustand der Drohne selbst und der separaten Kamera angezeigt. Zudem aber auch Verbindungsqualität zwischen Drohne und Fernbedienung, sowie die aktuelle Höhe über dem Boden. Auf dem ganzen Bildschirm wird ausserdem eine Liveübertragung des Kamerabildes der Drohne angezeigt. Damit die Steuerungshilfen das Bild nicht stören, werden diese beim Entfernen des Fingers ausgeblendet und verdecken auch sonst nur minimal das Bild.

Verbindung

Ein Smartphone hat als direkte Verbindungsmöglichkeit meistens nur WLAN oder Bluetooth zur Verfügung. Bluetooth ist dabei in Reichweite und Datenübertragungsrate sehr stark eingeschränkt. Bei WLAN ist die Übertragungsrate zwar deutlich höher, was eine Livebildübertragung möglich machen würde, die Reichweite ist aber auch hier wegen der kleinen Smartphone-Antenne und des WLAN-Protokolls limitiert. Als Lösung für dieses Problem wird ein weiteres Sende- und Empfangsgerät verwendet, das - mit einem speziellen Protokoll - eine Verbindung auf grössere Distanz mit der Drohne erzielen kann. Dieses Gerät ist hier ein Raspberry Pi 3 mit einem Linux Betriebssystem. Es baut über herkömmliches WLAN eine Verbindung mit dem Smartphone auf und leitet sämtliche Pakete in beide Richtungen weiter. Auf der Drohne befindet sich ein Raspberry Pi in kleinerer Version (Raspberry Pi Zero), der diese Daten empfangen kann und diese dem Flugcontroller weiterleitet. Ausserdem kommuniziert er direkt mit der Kamera und erhält dadurch das Livebild sowie einige Daten der Kamera. Dies geschieht, trotz der geringen Distanz, über WLAN, da die verwendete Kamera nur darüber kommunizieren kann.

Die Funkverbindung zwischen den beiden Raspberry Pis findet etwa im Frequenzbereich von 2.4 GHz statt. Jeder Raspberry Pi ist mit einem speziellen WLAN-Sender ausgerüstet, welcher so umfunktioniert werden kann, dass er nicht mehr dem limitierenden WLAN-Protokoll entspricht. Das Open-Source-Programm ‚Wifibroadcast'[3] programmiert die WLAN-Sender so, dass sie alle Pakete in einem definierten Frequenzband empfangen und weiterleiten. Beim Senden wird hierbei nicht sichergestellt, dass das Paket beim Empfänger ankommt. Das ist der grosse Unterschied zum WLAN-Protokoll, bei dem jedes Paket vom Empfänger bestätigt werden muss und daher auch zuerst eine Verbindung aufgebaut werden muss. Da nun jedes Paket roh weitergeleitet wird, wird es stattdessen durch das Programm ‚Wifibroadcast' weiterverarbeitet. Dies ist eine speziell für den Raspberry Pi geschriebene Software, um eine Liveübertragung über weite Distanzen mit Hilfe normaler WLAN-Sender zu übertragen. Diese Software habe ich stark modifiziert, damit meine Kamera damit funktioniert, aber auch um Datenübertragungen in beide Richtungen zu ermög-

[3] https://befinitiv.wordpress.com/wifibroadcast-analog-like-transmission-of-live-video-data/

lichen. Nach der Modifikation kann Livebild und Steuerung auf dem selben Kanal übermittelt werden.

Abbildung 14: Raspberry Pi mit einem WLAN-Sender auf der Drohne, der für die Datenübertragung zuständig ist

Jegliche Datenübertragungen (ausser der Videoübertragung) geschehen über ein selber entwickeltes Protokoll. Jedes Paket enthält am Anfang den Zielort und den Typ des Pakets. Die wichtigsten beiden Pakettypen sind das Kontrollpaket, welches unter anderem die momentane Steuerungseingabe zehnmal pro Sekunde übermittelt, und das Telemetriepaket, welches jede Sekunde die wichtigsten Informationen über die Drohne an die Fernbedienung übermittelt. In jedem Paket ist ausserdem eine Prüfsumme integriert, um Übertragungsfehler zu entdecken. Es wird dafür eine Prüfsumme benutzt, die in ihrer Berechnung nicht rechenintensiv ist [4]. Dies ist unbedingt notwendig, da der Flugcontroller nur eine beschränkte Rechenleistung besitzt.

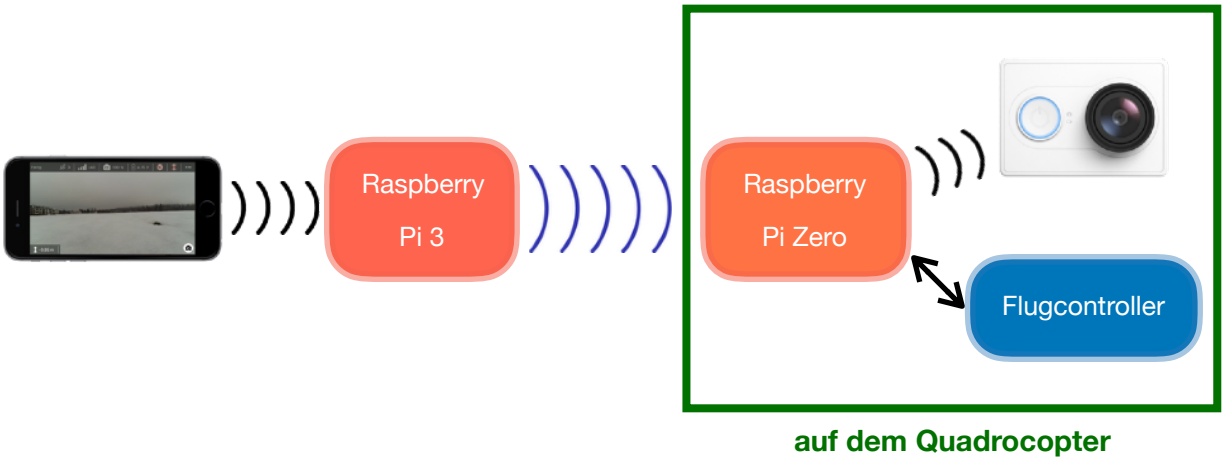


Abbildung 15: Verbindungsstationen von der App bis zum Quadrocopter skizziert [18]

[4] die sogenannte CRC16-Prüfsumme wird dafür verwendet

8. Steuerungsalgorithmen

8.1. PID-Controller [15]

Ein PID-Controller ist ein Algorithmus, der versucht, einen gewünschten Zustand eines Systems zu erreichen mit Hilfe von Messungen des momentanen Zustands. Der PID-Controller steuert dabei gewisse Hilfsmittel - wie in diesem Fall die Motoren - so an, dass dieses Ziel möglichst schnell erreicht wird. Bei einem Quadrocopter ist er zum Beispiel dafür nützlich, um eine gewünschte Lage im Raum zu erreichen. Der PID-Controller berechnet dabei die Motorengeschwindigkeiten, um dieses Ziel zu erreichen. Er ist auch dafür da, einen gewissen Zustand zu halten und auf äussere Störungen zu reagieren.
Der Name ‚PID' beschreibt bereits die Funktionalität dieses Controllers. Es steht im Englischen für „proportional–integral–derivative" und beschreibt die drei Glieder, aus denen dieser PID-Controller besteht. Diese drei Glieder werden oft als P-, I- und D-Controller bezeichnet. Alle diese drei Glieder arbeiten mit dem Fehler zwischen dem Momentanwert $y(t)$, der meist durch gewisse Sensoren gemessen wird, und dem Sollwert $u(t)$. Der Fehler wird hier als $e(t)$ bezeichnet.

$$e(t) = u(t) - y(t)$$

P-Controller

Der P-Controller nimmt diesen Fehler $e(t)$ und multipliziert ihn mit einer Konstanten K_p. Das P steht dabei für Proportional. Sein Output ist immer proportional zum Fehler. Bei einem grösseren Fehler reagiert er daher stärker und bekämpft diesen auch stärker.

$$P(t) = K_p * e(t)$$

I-Controller

Der I-Controller integriert den Fehler, ganz nach seinem Namen, über die gesamte Zeit. Dadurch wird nicht nur der momentane Fehler, sondern auch die Fehler in der Vergangenheit berücksichtigt. Er kann dadurch sehr gut konstante äussere Einflüsse bekämpfen. Beim Quadrocopter wäre dies beispielsweise der Wind, den dieser Controller dadurch frühzeitig bekämpfen kann, ohne dass der Wind zuerst einen Fehler entstehen lassen muss, damit der P-Controller reagieren würde. Die integrierten Fehler werden ebenfalls mit einer Konstante K_i multipliziert.

$$I(t) = K_i * \int_0^t e(\tau)\,\mathrm{d}\tau$$

D-Controller

Das D in D-Controller steht für ‚Derivative', was zu deutsch Ableitung heisst. Dieser Teil des Controllers beschäftigt sich mit der Fehlerveränderung. Je schneller sich der Fehler ändert, desto heftiger reagiert der D-Controller. Eine schnelle Fehlerveränderung kommt meistens durch eine schnelle Veränderung des gemessenen Momentanwerts zustande, kann aber natürlich auch durch eine plötzliche Änderung des Sollwerts entstehen. Die Ableitung des Fehlers bietet dabei einen gewissen Blick in die Zukunft. Vergrössert sich der Momentanwert schnell, wird er in der näheren Zukunft mit grösster Wahrscheinlichkeit grösser werden. Daher kann dieser Teil des PID-Controllers einen zu grossen Fehler bereits sehr früh erkennen und verhindern. Zudem wird bei einer starken Abnahme des Fehlers das System gebremst, um zu verhindern, dass der Wert über das Ziel hinausschiesst. Es dämpft daher in gewissen Situationen den P-Controller, der alleine oft zu Oszillationen führen kann. Auch der D-Controller wird mit einer Konstante K_d multipliziert.

$$D(t) = K_d * \frac{\Delta e(t)}{\Delta t}$$

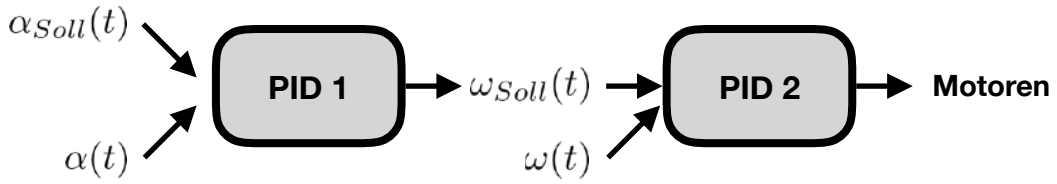
Der Output des PID-Controller $O(t)$ setzt sich schlussendlich aus der Summe der Werte der drei Controller zusammen.

$$\begin{aligned} O(t) &= P(t) + I(t) + D(t) \\ &= K_p * e(t) + K_i * \int_0^t e(\tau)\,\mathrm{d}\tau + K_d * \frac{\Delta e(t)}{\Delta t} \end{aligned}$$

Ein PID-Controller kann in sehr vielen verschiedenen Bereichen genutzt werden. Dabei müssen die drei Konstanten K_p, K_i und K_d der Anwendung und dem gewünschten Verhalten entsprechend eingestellt werden. Oft werden auch einzelne Teile des PID-Controllers nicht verwendet, also die Konstante auf Null gesetzt. Zudem kann je nach Anwendungsfall der Output des PID-Controllers anders verwendet werden. Er kann zum Beispiel direkt als Befehl an die Motoren weitergegeben werden, aber auch zu einem vorherigen Wert hinzugezählt werden, der dann als Befehl zu einem Motor geht.

8.2. Stabilisierung

Die wichtigste Funktion des Flugcontrollers ist, die Drohne in der Luft zu stabilisieren. Würden alle vier Motoren mit der gleichen Geschwindigkeit drehen, würde sich der Copter zwar in der Theorie nicht drehen. In der Praxis gibt es jedoch sehr viele Störungsquellen, wie beispielsweise Wind oder leichte Unterschiede in den Motoren oder Propellern. Ausserdem ist es gewünscht, dem Quadrocopter eine bestimmte Neigung vorzugeben, welche dieser umzusetzen hat. Für diese Aufgabe kann ein PID-Controller genutzt werden. Der PID-Controller berechnet dabei 200 Mal in der Sekunde, welchen Geschwindigkeitsunterschied die Motoren der linken Hälfte gegenüber der rechten Hälfte haben sollen, um dabei die Drohne in die gewünschte Position zu bringen. Dabei verwende ich zwei ineinander verschachtelte PID-Controller. Der innere versucht eine gewünschte Winkelgeschwindigkeit $\omega_{Soll}(t)$ zu erreichen. Der momentane Wert ist der direkt vom Gyroskop gemessene Wert $\omega(t)$. $\omega_{Soll}(t)$ könnte dabei bereits von der Fernsteuerung gesteuert werden. Um jedoch zu erreichen, dass der Quadrocopter automatisch in einer horizontalen Lage bleibt, beziehungsweise in der Neigung, die die Fernsteuerung vorgibt, wird ein zweiter PID-Controller verwendet, um die gewünschte Winkelgeschwindigkeit $\omega_{Soll}(t)$ zu berechnen, mit der die gewünschte Neigung $\alpha_{Soll}(t)$ erreicht werden kann. Der Momentanwinkel $\alpha(t)$ wurde dabei zuvor durch die Kombination von Gyroskop und Beschleunigungsmesser ermittelt. Vom äusseren PID-Controller wird jedoch nur der proportionale P-Teil benutzt. Daher ist die gewünschte Winkelgeschwindigkeit immer proportional zum Fehler der Neigung. Die Konstante dieses P-Controllers K_{p1} bestimmt daher auch wie schnell die Zielneigung erreicht werden soll.



Dieser durch den inneren PID-Controller erhaltene Wert wird anschliessend bei den Motoren auf der einen Seite vom Schub abgezogen und auf der andere Seite addiert. Damit dieses System jedoch wirklich das tut, was es sollte, müssen die vier Konstanten K_{p1}, K_{p2}, K_{i2} und K_{d2} richtig gewählt werden. Diese Konstanten können jedoch nicht theoretisch berechnet werden, sondern müssen durch Versuche am echten Objekt getestet werden. Dabei kommt es jedoch gerade am Anfang bei komplett zufälligen Werten immer wieder zu Abstürzen, die man zwar nicht verhindern kann, aber deren Intensität man durch nahes Bodenfliegen und die Wahl von kleineren Werten für die Konstanten am Anfang vermindern kann.

Dieser PID-Controller wird zweimal verwendet. Neben der Stabilisierung der Neigung von rechts nach links, was auch Roll genannt wird, wird auch die Neigung nach vorne und nach hinten, namens Pitch, mit dem gleichen PID-Controller stabilisiert. Ich verwendete dabei für beide PID-Controller die selben vier Konstanten. Als Verbesserung könnten diese noch separat optimiert werden.

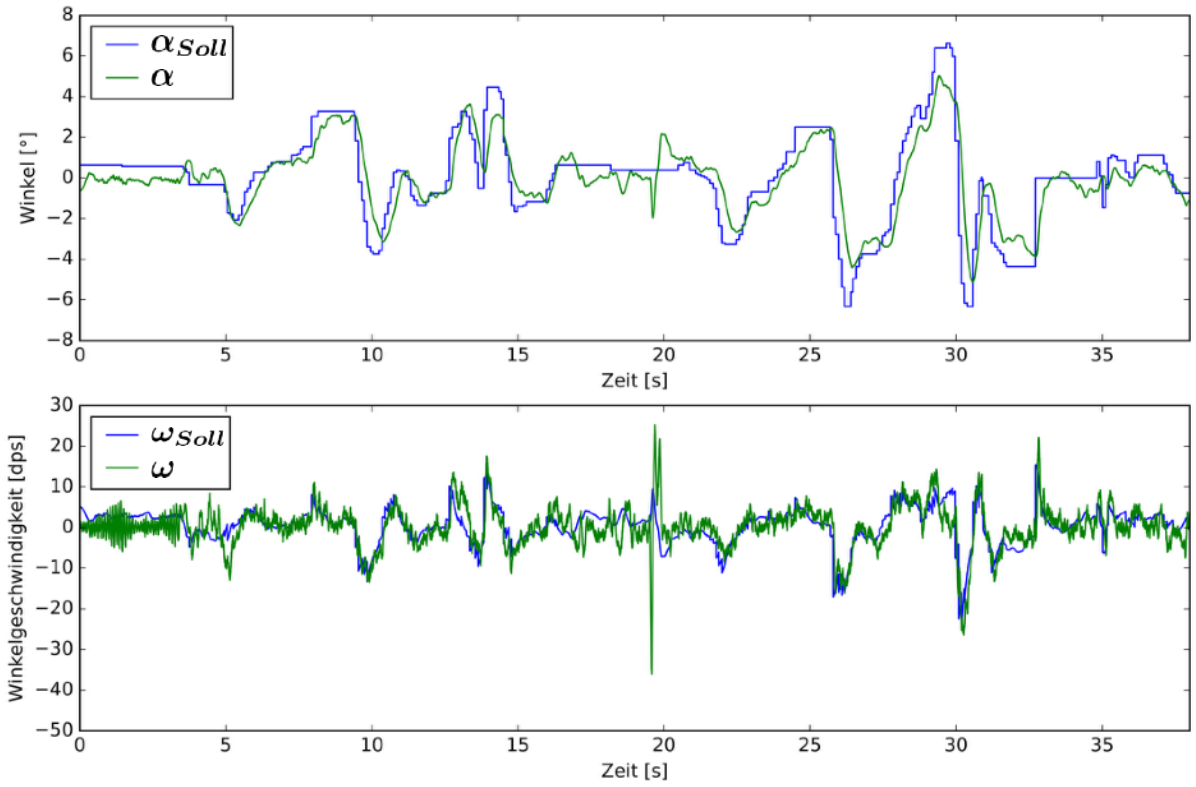
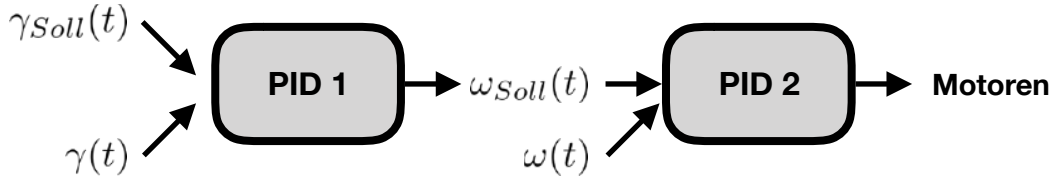


Abbildung 16: Messdaten bei einem Testflug. Das obere Diagramm zeigt, wie gut der Copter eine gewünschte Neigung behalten kann. Im unteren Diagramm wird die Einhaltung der Winkelgeschwindigkeit gezeigt, wofür der innere PID-Controller zuständig ist.

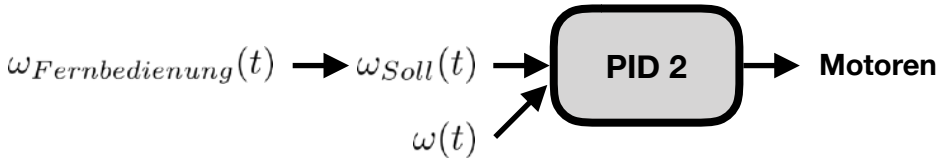
Zur kompletten Stabilisierung des Quadrocopters muss jedoch auch noch die dritte Achse beachtet werden, um die er sich drehen kann. Ein Drehen um diese Achse verändert die Ausrichtung des Copters. Diese Achse wird auch Yaw genannt. Die Stabilisierung soll verhindern, dass sich die Drohne um sich selbst dreht und seine Ausrichtung ändert, oder dies nur kontrolliert tut. Um dies zu bewerkstelligen, werden erneut zwei PID-Controller ineinander verschachtelt. Der innere PID-Controller berechnet auch hier den Fehler der Winkelgeschwindigkeit mit der tatsächlichen Geschwindigkeit $\omega(t)$, die vom Gyroskop gemessen wird, und der gewünschten Geschwindigkeit $\omega_{Soll}(t)$, die vom äusseren PID-Controller berechnet wird. Der äussere Controller rechnet mit dem durch Gyroskop und Kompass gemessenen Winkel $\gamma(t)$. Der Sollwert $\gamma_{Soll}(t)$ wird hier jedoch, im Unterschied zu den beiden anderen Achsen, nicht durch die Fernbedienung bestimmt, sondern beim Abheben des Copters festgesetzt. Die Drohne versucht daher, ihre Nase in der selben Richtung zu behal-

ten wie beim Start. Die Fernbedienung hat jedoch die Möglichkeit, die Drohne um die eigene Achse zu drehen. Dabei wird der äussere PID-Controller übersprungen und die Fernbedienung bestimmt die gewünschte Winkelgeschwindigkeit $\omega_{Soll}(t)$. Sobald die Fernbedienung die Drohne nicht mehr drehen will, wird die momentane Ausrichtung $\gamma(t)$ als neuer Sollwert $\gamma_{Soll}(t)$ gespeichert und der äussere Controller wird nicht mehr übersprungen.

Wenn Befehl der Fernbedienung $\omega_{Fernbedienung}$ ***= 0:***



Wenn Befehl der Fernbedienung $\omega_{Fernbedienung}$ ***> oder < 0:***



Bei den Stabilisierungsalgorithmen aller drei Achsen können ebenfalls Probleme wegen zu starker Vibrationen entstehen. Der innere PID-Controller verwendet im D-Teil jeweils die Ableitung der Messung des Gyroskop. Da trotz des Filtern per Software weiterhin Vibrationen vorhanden sind, verstärkt die Ableitung diese Vibrationen stark. Wenn starke Vibrationen dem Motor weitergeben werden, versucht dieser innerhalb von Millisekunden sehr stark zu beschleunigen und wieder abzubremsen. Die überschüssige Energie der Drehung beim Bremsen wird dabei zum grössten Teil zu Wärme umgewandelt. Bremst er also zu häufig stark ab, kann der Motor sehr heiss werden und dabei zerstört werden. Als Lösung wird die Ableitung ein zweites Mal gefiltert. Der Filter darf allerdings nicht zu stark sein, da ansonsten zu grosse Verzögerungen entstehen würden. Da der D-Controller vor allem dafür da ist, schnell zu reagieren, würde er seinen Zweck nicht mehr erfüllen. Da der D-Controller den P-Controller dämpfen soll, könnte er dies ebenfalls schlechter tun. Im schlechtesten Fall würde der D-Controller sogar noch den P-Controller verstärken und dabei für extreme Oszillationen sorgen [19]. Dieses Problem habe ich bei diesem Quadrocopter ebenfalls feststellen können. Ausserdem sollte die Konstante des D-Controllers K_d nicht zu gross sein.

Endgültiger Motorenschub

Um schlussendlich für jeden Motor die definitive Drehgeschwindigkeit herauszufinden, muss jeweils für jeden Motor die Distanz zur Drehachse bestimmt werden. Die Drehachsen verlaufen dabei jeweils durch den Schwerpunkt. Je weiter entfernt sich ein Motor von der Drehachse befindet, desto grösser ist der Weg, den dieser für die selbe Drehung zurücklegen muss. Die zurückgelegte Strecke ist dabei proportional zur Distanz zur Achse. Die grösste Distanz wird als 1 angesehen und die anderen angepasst. Daher wird der Schub für die Motoren 1-4 (siehe Abbildung) mit dem bereits bekannten allgemeinen Schub T folgendermassen berechnet:

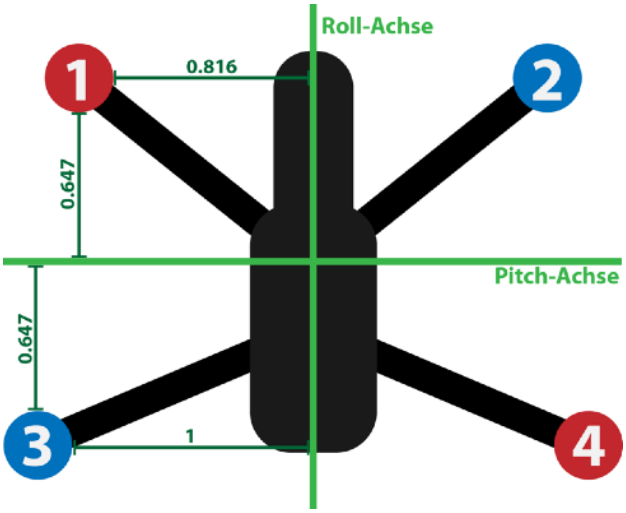
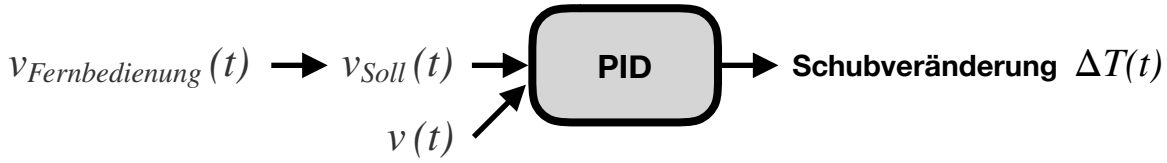


Abbildung 17: Eine Illustration der Form der Verwendeten Drohne. Die Rotationsachsen sind hellgrün markiert, wobei der Schnittpunkt den Schwerpunkt darstellt. Die Distanzen der Motoren zur Achse sind jeweils im Verhältnis zur grössten Distanz angegeben.

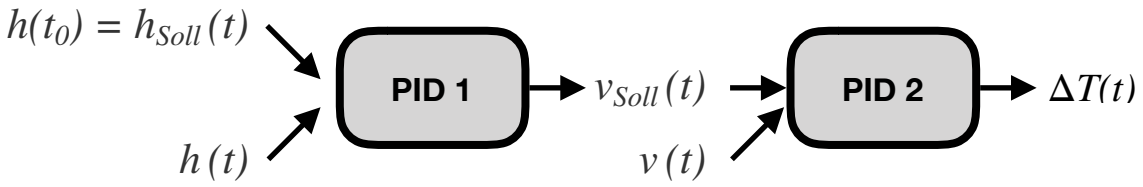
$$M_1 = T - 0.647 * PID_{Pitch} + 0.816 * PID_{Roll} + PID_{Yaw}$$
$$M_2 = T - 0.647 * PID_{Pitch} - 0.816 * PID_{Roll} - PID_{Yaw}$$
$$M_3 = T + 0.647 * PID_{Pitch} - PID_{Roll} + PID_{Yaw}$$
$$M_4 = T + 0.647 * PID_{Pitch} + PID_{Roll} - PID_{Yaw}$$

8.3. Höhenstabilisierung

Statt dass die Fernbedienung den Schub des Quadrocopters vorgibt und damit die Höhe des Quadrocopters steuert, soll der Flugcontroller die Höhe automatisch halten beziehungsweise eine vorgegebene vertikale Geschwindigkeit erfüllen. Dafür kann erneut ein PID-Controller verwendet werden. Dieser soll dann benutzt werden, wenn die Fernsteuerung eine Höhenveränderung wünscht. Der PID-Controller benötigt für seine Berechnungen die gewünschte vertikale Geschwindigkeit $v_{Soll}(t)$ und eine gemessene Geschwindigkeit $v(t)$. Die Problematik der Bestimmung dieser Geschwindigkeit wird dabei weiter unten erläutert. Der PID-Controller berechnet daraufhin die benötigte Veränderung des Schubes, der benötigt wird, um die Geschwindigkeit zu erreichen oder zu halten. Diese Schubveränderung $\Delta T(t)$ wird dabei zum vorherigen Schub $T(t - \Delta t)$ hinzugefügt.



Gibt die Fernbedienung allerdings keine Höhenveränderung vor, muss der Quadrocopter versuchen, auf gleicher Höhe zu bleiben. Würde die oben beschriebene Methode auch hier verwendet und für $v_{Soll}(t)$ daher Null eingesetzt, würde sich die Höhe der Drohne trotzdem relativ schnell ändern. Der Grund dafür ist, dass der echte Wert nie ganz genau an den gewünschten Wert herankommt, auch wenn der PID-Controller noch so gut eingestellt ist. Messfehler kommen noch hinzu. Schon ein leichter Fehler in der Geschwindigkeit hat über eine längere Zeit sehr grosse Auswirkungen.
Stattdessen werden, wie bei der Stabilisierung der Lage, zwei PID-Controller verschachtelt. Der innere ist dabei der oben beschriebene Controller, der versucht, eine gewisse vertikale Geschwindigkeit einzuhalten. Die Sollgeschwindigkeit wird jedoch nicht von der Fernbedienung vorgegeben, sondern vom äusseren PID-Controller. Dieser berechnet die gewünschte vertikale Geschwindigkeit $v_{Soll}(t)$ aus der vom Barometer oder Ultraschallsensor gemessenen Höhe $h(t)$ und der gewünschten Höhe $h_{Soll}(t)$. Auch hier wird beim äusseren PID-Controller nur der proportionale Teil verwendet. Die einzuhaltende Höhe ist dabei die Höhe, die der Quadrocopter hatte, als die Höhenhaltungsfunktion aktiviert wurde, beziehungsweise zum Zeitpunkt t_0, als die Fernbedienung zu einer gewünschten Geschwindigkeit von Null wechselte. Daher kann man $h_{Soll}(t)$ auch als $h(t_0)$ bezeichnen. Das $h_{Soll}(t)$ könnte aber natürlich auch auf einen gewissen Wert festgelegt werden, um zum Beispiel die Drohne automatisch auf 20 Meter Höhe zu bringen. Damit dies auch funktioniert, müsste jedoch zusätzlich noch die Ausgabe des äusseren PID-Controllers limitiert werden auf eine Höchstsink- und Höchststeigegeschwindigkeit. So würde eine konstante Geschwindigkeit eingehalten werden bis kurz vor Erreichen des Ziels.



Messung der vertikalen Geschwindigkeit

Kein verwendetes Messgerät kann direkt die vertikale Geschwindigkeit messen. Da jedoch die Höhe durch den Barometer und den Ultraschallsensor gemessen wird, kann man durch Ableiten die Geschwindigkeit erhalten. Da diese Höhe, vor allem die des Barometers, jedoch nicht sehr genau ist und leicht schwankt, wird dieser Effekt in der Ableitung deutlich verstärkt. Die abgeleitete Barometerhöhe erreicht teilweise Werte von bis zu 10 m/s trotz Stillstand und Windstille. Zur Reduktion dieser Schwankungen könnte ein Low-Pass-Filter verwendet werden, der die Probleme bei Gyroskop und Beschleunigungssensor lösen könnte. Dies versuchte ich, was darin resultierte, dass der Quadrocopter mit einer sehr grossen Geschwindigkeit nach oben beschleunigte und nach einigen Sekunden den Schub so stark senkte, dass die Drohne auf den Boden krachte. Der Grund dafür war, dass eine leichte Höhenreduktion stattfand, welche er kompensieren wollte. Daher war ΔT zu diesem Zeitpunkt im positiven Bereich, was den Schub steigen liess. Daraufhin gewann der Copter an Höhe. Diese nun nach oben gerichtete vertikale Geschwindigkeit bemerkte der PID-Controller jedoch nicht rechtzeitig, da der Filter eine zu starke Verzögerung zur Folge hatte. Daher wurde der Schub logischerweise weiter erhöht. Zur Verkleinerung dieser Verzögerung könnte man den Filter weniger stark einstellen. Dies würde jedoch wieder zu stark schwankenden Geschwindigkeiten führen, womit keine brauchbaren Resultate des PID-Controllers erwartet werden könnten.
Der zweite Weg zur Bestimmung der vertikalen Geschwindigkeit ist die Verwendung des Beschleunigungssensors. Wird dieser über die gesamte Zeit integriert, ergibt sich theoretisch die Geschwindigkeit. Als Beschleunigung kann dabei näherungsweise die Gesamtbeschleunigung $a(t)$ verwendet werden, von welcher der Betrag der Erdbeschleunigung $9.81\,m/s^2$ abgezogen wird. Das Problem ist allerdings, dass sich kleine Fehler aufaddieren. Je weiter weg man vom Startzeitpunkt ist, desto grösser wird der Fehler. Daher kann man auch diese Methode nicht verwenden.

Man kann allerdings beide Methoden, wie bei der Fusion von Beschleunigungsmesser und Gyroskop, kombinieren, um die tatsächliche vertikale Geschwindigkeit möglichst genau zu ermitteln. Erneut kann dazu ein Komplementärfilter wie folgt benutzt werden.

$$v(t) = C * (v(t - \Delta t) + (a(t) - 9.81 m/s^2)\Delta t) + (1 - C) * \frac{h(t) - h(t - \Delta t)}{\Delta t}$$

Mit einem C, das sehr gross ist, könnte dies sehr gut funktionieren, da die Berechnung über die Beschleunigung kurzfristig sehr gut ist und die Berechnung über die Höhe durch das Addieren mit einem sehr kleinen Faktor kurzfristig keine Rolle spielt, aber langfristig den Integrationsfehler verhindert.
Dieser Komplementärfilter konnte jedoch nie zum Einsatz kommen, da sämtliche benutzten Beschleunigungssensoren (alle gleiches Modell) einen gleichen Fehler aufwiesen, der zwar die Winkelberechnung nicht stark beeinträchtigte, aber diese Berechnung unmöglich machte. Die am Boden gemessenen Gesamtbeschleunigungen lagen bei der normalen Erdbeschleunigung. Begannen die Motoren jedoch zu drehen, war sie fast immer kleiner als die normale Erdbeschleunigung, obwohl der Copter weiter auf dem Boden stand. Das hätte im obigen Komplementärfilter zur Folge, dass die vertikale Geschwindigkeit relativ schnell auf eine starke Sinkrate fallen würde. Trotz verschiedenen Stromquellen und anderen Positionen bei den Tests trat das Phänomen immer wieder auf und ich konnte den Grund dafür nicht herausfinden. Eine Gegenüberstellung von Schub und Beschleunigung in einer Grafik und eine Regressionsanalyse brachte zwar einen ungefähr erkennbaren Trend, aber auch keine lösungsbringende Formel zur Korrektur dieses Fehlers per Software.
Da auch der Ultraschallsensor nicht genügend zuverlässig funktionierte, dessen abgeleitete Höhe ansonsten durchaus als alleinige Quelle für die Geschwindigkeit brauchbar gewesen wäre, wurde die Höhenstabiliserung für den Quadrocopter zwar implementiert, war aber nie einsatzfähig.

9. Schlussfolgerung

Es ist mir gelungen, einen Quadrocopter zu bauen und zu programmieren, der in der Luft stabil fliegt und mit einem Smartphone gesteuert werden kann. Um dies möglich zu machen, musste ich auf vielen verschiedenen Ebenen Software schreiben. Die Verbindung zwischen Bodenstation und Drohne mit Übertragung der Steuerungsbefehle und des Livebildes, die App zur Steuerung der Drohne und deren Verbindung zur Bodenstation sowie aber auch der Flugcontroller selbst mussten programmiert werden.

Die Erfahrungen mit diesem Quadrocopter zeigten, dass nicht unbedingt die Kontrollalgorithmen zur Stabilisierung einer bestimmten physikalische Grösse das schwierigste am Projekt waren, sondern eher deren Input, also die Messdaten. Auch wenn diese Sensoren auf dem Papier perfekte Messdaten bringen sollten, war dies hier bei keinem der Sensoren wirklich der Fall. Vibrationen waren dabei eines der Probleme. Sensoren wie der Beschleunigungsmesser und der Ultraschallsensor haben sogar eine Höhenstabilisierung der Drohne verunmöglicht, da ich den Grund für die Probleme nicht finden konnte. Dadurch wurde auch automatisches Landen und Starten unmöglich.

Die Drohne kann sicher an vielen Stellen verbessert werden. Mit einer funktionierenden Höhenstabilisierung, könnten die Sensoren mit einem GPS ergänzt werden, was dem Copter erlauben würde, an einem bestimmten Ort trotz Wind oder anderen äusseren Einflüsse stehen zu bleiben. Zudem wäre eine automatische Kollisionserkennung durch einen Ultraschallsensor vorne eventuell hilfreich.

10. Quellenverzeichnis

[1] Brushless Motor Kv to RPM Converter (11.11.2017)
https://kb.rcoverstock.com/tools/calculators/powerplant-calculators/brushless-motor-kv-to-rpm-converter/

[2] BuddyRC: SunnySky V2216-12 KV800 II Brushless Motor (12.11.2017)
http://www.buddyrc.com/sunnysky-v2216-12-800kv-ii-brushless-motor.html

[3] D-Edition: Brushless vs. Brushed – Motoren (12.11.2017)
http://www.d-edition.de/blog/ratgeber/brushless-vs-brushed-motoren/

[4] DMK Deutschschweiz: Formeln, Tabellen, Begriffe; Orell Füssli Verlag
Ausbreitungsgeschwindigkeit c, Seite 188

[5] Dronetest: LiPo Batteries - How to choose the best battery for your drone (26.11.2017)
http://www.dronetrest.com/t/lipo-batteries-how-to-choose-the-best-battery-for-your-drone/1277

[6] Infogram: Carbon Fiber vs Fiberglass (11.11.2017)
https://infogram.com/carbon-fiber-vs-fiberglass-1gdk8pd471dvpq0

[7] Kopterforum: Der Propeller bei Multicoptern (12.11.2017)
http://www.kopterforum.de/tutorials/article/1-der-propeller-bei-multicoptern/

[8] Maxbotix: MaxSonar Operation on a Multi-Copter
https://www.maxbotix.com/tutorials7/067-maxsonar-operation-on-a-multi-copter.htm

[9] MaxBotix: MB1240 XL-MaxSonar-EZ4
https://www.maxbotix.com/Ultrasonic_Sensors/MB1240.htm

[10] RC Groups: Sunnysky V2216 800KV Thrust Test (12.11.2017)
https://www.rcgroups.com/forums/showthread.php?2032374-Sunnysky-V2216-800KV-Thrust-Test

[11] Reinhard Rahner: Luftdruck - barometrische Höhenformel (11.12.2017)
https://www.rahner-edu.de/mikrocontroller/themen-und-projekte/höhenbestimmung/

[12] Roger's Hobby Center: A Guide to Understanding LiPo Batteries (26.11.2017)
https://rogershobbycenter.com/lipoguide/

[13] Wikipedia: Breguet Richet Gyroplane (9.11.2017)
https://en.wikipedia.org/wiki/Quadcopter

[14] Wikipedia: Curtiss-Wright VZ-7 (9.11.2017)
https://en.wikipedia.org/wiki/Curtiss-Wright_VZ-7

[15] Wikipedia: PID controller (30.12.2017)
https://en.wikipedia.org/wiki/PID_controller

[16] Wikipedia: Quadrocopter (9.11.2017)
https://en.wikipedia.org/wiki/Quadcopter

[17] World-Modellbau: LiPo vs. NiHM – Wo liegen Vor- und Nachteile? (26.11.2017)
http://www.world-modellbau.de/lipo-vs-nihm-wo-liegen-vor-und-nachteile/

[18] Yi: YI Action Camera (16.12.2017)
http://www.yitechnology.com/yi-action-camera

[19] YouTube: Cleanflight / Betaflight Advanced D Term Tuning - Part 2 (31.12.2017)
https://www.youtube.com/watch?v=FlZe0FNjuxo

[20] YouTube: ESC electronic speed controller with arduino ALL EXPLAINED (12.11.2017)
https://www.youtube.com/watch?v=8LXPcJD6hEA

[21] Zentralanstalt für Meteorologie und Geodynamik: IGRF Deklinationsrechner (10.12.2017)
http://www.zamg.ac.at/cms/de/geophysik/produkte-und-services-1/online-deklinationsrechner

11. Anhang

Sämtlicher für den Quadrocopter geschriebene Programmiercode zu finden auf:

https://losjet.com/downloads/MA_Alexander_Eichhorn.zip

11.1.Tagebuch

Frühlings- bis Sommerferien: Schaltplan nach und nach konstruiert. (Video-)übertragung programmiert und getestet -> bricht nach geringer Distanz bereits ab.

16. - 18. Juli: Ich habe das PCB (Platine) nach dem Schaltplan entworfen und überprüft. Diese habe ich danach produzieren lassen und die Komponenten dafür ebenfalls bestellt.

24. Juli: Ankunft der Platine

27. - 28. Juli: Die SMD-Komponenten habe ich auf die Platine gelötet und getestet. Kleine Fehler beim SMD-Löten, wie z.B. zwei Pins die sich verbinden, sind kaum sichtbar, aber können grosse Auswirkungen haben. Ich habe eine Stunde gebraucht bis ich gemerkt habe, dass zwei Pins verbunden waren und es deswegen nicht funktioniert.

30. - 31. Juli: Stabilität der Funkverbindung zwischen Drohne und Fernsteuerung weiter verbessert und getestet. Die Distanz bei der Video vernünftig übertragen wird, verbesserte sich leicht, ist jedoch noch zu gering. Die maximale Distanz, die ich bisher erreichen konnte, war 50-70 Meter.

1. August: Testen von Hindernisausweichung vorne mit einer Kamera, deren Bild mit einem Optical-Flow-Algorithmus ausgewertet wird, um Punkte zu finden, die sich näher an der Kamera befinden. Die Resultate waren jedoch nicht zufriedenstellend und ich werde es eventuell später nochmals aufgreifen.

5. -6. August: Verbesserung der Funkverbindung, indem die Sendestärke erhöht wurde. Um dies zu tun, musste ich den Linux-Kernel verändern und neu kompilieren. Durch die langen Kompilierzeiten waren auftretende Fehler immer erst nach langem Warten ersichtlich. Die Verbesserung ist mehr oder weniger spürbar.

7. August: Die iPhone-App, mit der man später die Drohne steuern soll, habe ich begonnen zu programmieren und eine Verbindungstest-Funktion hinzugefügt, um u.a. die Anzahl verlorenen Pakete zu messen.

8. August: Anderer Weg um den Live-Video-Feed dem Sender aufzubereiten ausprobiert. Die Kamera sollte das Bild via einem analogem Videosignal an den Prozessor in der Drohne senden, der dieses dann weiter verarbeiten kann. Die Verzögerung würde dadurch reduziert werden, die Qualität ist allerdings relativ schlecht. Deswegen werde ich das im Moment nicht weiter verfolgen.

10. August: Die für den Anfang wichtigsten Teile des Protokoll für die Kommunikation zwischen Drohne und Fernsteuerung/iPhone in einer Excel-Tabelle definiert, um die Implementation auf die verschiedenen Plattformen später nahtlos zu gestalten.

11. August: Erste Ausprogrammierung des Protokolls auf Seite der Drohne. Zudem habe ich einen 2.4-Ghz-Funkverstärker erhalten, der allerdings beim Start von Übertragung grosser Datenmengen sich selbst zerstörte, also defekt war. Daher konnte ich keine Tests damit durchführen.

12. August: Ich habe den Chip, mit integriertem Beschleunigungsmesser, Gyroskop und Magnetfeldmesser, auf meine sonst bereits gelötete Platine gelötet. Dies war jedoch wieder eine Herausforderung, da die Pins dieses Chips äussert klein sind.

15. August: In der iPhone-App, über die die Drohne gesteuert werden kann, habe ich die beiden Sticks für manuelles Fliegen designed und einprogrammiert. Die Schwierigkeit besteht darin, die Steuerung so zu bauen, dass trotz fehlendem haptischem Feedback von gewöhnlichen Fernsteuerungen sie einfach zu bedienen ist.

7. September: Erster Zusammenbau, der erstmals nur die wichtigsten Sensoren und Komponente beinhalten sollte, geplant. Ich weiss jetzt genau, wo was im Quadrocopter hinkommen soll, um den wenigen Platz optimal zu nutzen und aber auch den Schwerpunkt möglichst zentral zu halten. Der Teil, der für die Bild- und Steuerungsübertragung verantwortlich ist, habe ich bereits gelötet und getestet.

9. September: Die Stromverteilung an die Motoren habe ich so gelötet, dass die Distanz von Batterie zu Motor jeweils möglichst klein ist. Einen Stromverbrauchsmesser habe ich dabei am nächsten zur Batterie platziert. Ausserdem habe ich zwei von vier Arme des Quadrocopters zusammengebaut. Ein Arm enthält jeweils ein Motor und eine Motorsteuerung, die die Geschwindigkeit als digitalen Input zu einem 3-phasigen Wechselstrom verwandelt, mit dem der Motor direkt betrieben wird.

10. September: Ich habe nun alle vier Arme zusammengebaut. Jeden Arm habe ich danach an den mittleren Teil angeschraubt und mit der Stromversorgung, die ich zuvor gebaut hatte, verbunden. Dies stellte sich allerdings als relativ schwierig dar, da ich, je mehr Arme sich am Quadrocopter befanden, weniger Platz zum Arbeiten hatte.
Die Hauptplatine habe ich danach provisorisch in die Mitte geklebt und deren Stromversorgung gesichert. Der Sende- und Empfangsteil mitsamt dessen Antenne habe ich ebenfalls provisorisch festgeklebt und mit der Hauptplatine verbunden.

11. September: Am frisch zusammengebauten Prototyp-Quadrocopter habe ich noch einige Funktionen erneut getestet wie zum Beispiel die Kommunikation mit den Motoren und diese zwischen Flugcontroller und Sende- und Empfangsmodul. Es schien alles zu funktionieren. Ausserdem habe ich begonnen den Code des Flugcontrollers zu schreiben. Zuvor hatte ich immer nur Testprogramme, die Funktionen und Verbindungen überprüften, aber diese nicht weiter verarbeiteten.

22. September: Der verbundene Gyroskop (Rotationsgeschwindigkeitsmesser) liefert die Daten an die Drohne. Diese Daten werden nun so verarbeitet, dass die relativen Winkel zu den Anfangswinkel (bei Start der Drohne) berechnet werden kann. Diese driften relativ schnell ab, weswegen ich eine Kalibrierung des Sensors einprogrammiert habe. Trotzdem

driften sie noch, was sich mit zusätzlichen Daten vom Beschleunigungsmesser deutlich reduzieren sollte.

23. September: Die drei Achsen des Beschleunigungsmesser werden nun benutzt, um die Gyroskopachsen zu erweitern und damit die Lage der Drohne genauer zu ermitteln. Ich habe eine Kombinierungsalgorithmus verwendet, der sehr wenig Rechenleistung benötigt. Es gibt aber noch bessere, die mehr Leistung benötigen. Diese kann ich, wenn es nötig sein wird, später testen.

24. September: Der Magnetometer, der auf 3 Achsen die Magnetfeldstärke ausgibt, konnte ich auslesen und den Winkel zum magnetischen Nordpol ausrechnen. Damit dies einigermassen genau funktioniert muss der Sensor zuvor kalibriert werden, um andere umgebende Magnetfelder aus der Berechnung auszuschliessen. Dafür musste ich ein Algorithmus entwickeln, der dies zuverlässig ausführt. Mit Kalibrierung stimmt der Winkel grob mit dem wahren Norden überein. Er oszilliert allerdings innerhalb von wenigen Millisekunden stark, was der Stabilität des Quadrocopters schaden würde. Deswegen kombinierte ich die Kompassdaten mit dem Gyroskop auf derselben Achse.

25. September: Der Flugcontroller kann nun Daten vom Empfängermodul auf dem Quadrocopter empfangen und teilweise verarbeiten. Ein kleiner Teil des am 10.08. erstmals definierten Protokolls habe ich dabei bereits umgesetzt. Ich habe jedoch noch nichts richtig getestet.

28. September: Beim Flugcontroller habe ich eine Art von Threads programmiert. Da der Quadrocopter verschiedene Sensoren auslesen und Berechnungen in kürzester Zeit durchführen muss, muss er immer wissen, welche Aufgabe er an einem gewissen Zeitpunkt am besten erledigen sollte. Dies habe ich umgesetzt, indem jede zu erledigende Aufgabe verschiedene Parameter hat wie: Wichtigkeit, Wiederholungsfrequenz, geschätzte Zeit zum Ausführen. Wie gut dies wirklich funktioniert, wird sich noch zeigen.

5. Oktober: Einen sogenannten PID-Controller habe ich einprogrammiert. Dieser soll der Drohne die Stabilität bringen. Dieser muss jedoch für jede Achse einzeln eingestellt werden, damit er funktioniert, was man während dem Fliegen machen muss. Ausserdem habe ich die Verbindung zum Empfängermodul so programmiert, dass nun die Pakete mit den wichtigsten Daten beim Flugcontroller ankommen und er diese verarbeiten kann. Dafür habe ich auch die App ausgebaut, damit sie die Steuerungseingaben gemäss Protokoll senden kann. Mit der App kann man den Quadrocopter nun also einigermassen steuern.

6. Oktober: Die in jedem Paket vorhandene Prüfsumme habe ich auf beiden Seiten ausprogrammiert, um Fehler bei der Übertragung aufzuspüren. Die Prüfsumme ist eine möglichst einfach zu berechnende, um den nicht rechenstarken Prozessor des Flugcontrollers nicht zu überfordern (CRC16). Ausserdem ist es nun möglich über die App die Drohne, sofern sie nicht am Fliegen ist, bzw. das Empfängermodul herunterzufahren. Da das Empfängermodul auf Linux läuft, ist die Gefahr da, dass das Dateisystem Probleme bekommt nach einem Unterbruch des Stromzuflusses ohne vorherigem Herunterfahren.

9. Oktober: Um alle PID-Controller richtig einzustellen, kann man nun während des Flugs die jeweiligen Koeffizienten direkt vom Boden aus ändern. Das beschleunigt den Prozess damit deutlich.

10. Oktober: Ich habe den einprogrammierten PID-Controller in Excel annähernd darstellen können und dabei herausgefunden, dass es mit einer kleinen Umstellung besser funktionieren würde. Statt direkt den gewünschten Winkel auf den 2 Achsen als Eingabe zu benutzen, wird mit dem Ziel- und Momentanwinkel die gewünschte Winkelgeschwindigkeit berechnet, welche dann vom PID-Controller besser verarbeitet werden kann.
Ausserdem habe ich verschiedene Positionen des Quadrocopters per Hand simuliert, um zu überprüfen, ob dieser ungefähr richtig reagiert (natürlich alles ohne Propeller). Dabei habe ich einen Fehler gefunden, der den Quadrocopter im Flug zum Absturz gebracht hätte.

11. Oktober: Um den Flug im Nachhinein analysieren zu können, habe ich eine SD Karte geplant, die sämtliche wichtigen Daten aufzeichnet. Ich habe nun begonnen dies möglichst effizient zu programmieren.

12. Oktober: Ich konnte den Quadrocopter zum ersten Mal fliegen lassen. Die Werte, um stabil zu fliegen, einzustellen, habe ich auch versucht. Es stellte sich jedoch als sehr schwierig heraus, da der Quadrocopter sehr schwer ist und daher auch sehr gefährlich werden kann, wenn man ihn nicht mehr kontrollieren kann. Ich konnte die Stabilität etwas verbessern. Trotzdem hoffe ich, dass es mit einer Aufzeichnung aller Berechnungen auf einer SD-Karte einfacher gehen wird.

14. Oktober: Mit Hilfe der Aufzeichnungen der Flugdaten auf die SD-Karte, konnte ich die Flugschwierigkeiten im Nachhinein grafisch ansehen. Diese zeigten mir, dass die Werte des Beschleunigungsmesser sowie des Gyroskop sehr stark oszillieren. Der Beschleunigungsmesser mass Kräfte von bis zu 4g, die innerhalb von wenigen Millisekunden komplett die Richtung änderten. Ich vermute, dass die Sensordaten noch weiter gegen hohe Frequenzen gefiltert werden müssen und diese zuerst feinen Oszillationen durch meine Software immer weiter verstärkt werden, was den Quadrocopter praktisch flugunfähig macht.

15. Oktober: Hohe Frequenzen werden nun direkt vom Sensor verstärkt herausgefiltert. Dies reduzierte die Vibrationen jedoch nur sehr schwach. Da die Vibrationen nur bei Betrieb mit Propellern so gross sind, habe ich die Propeller so ausgeglichen, dass sie den Schwerpunkt genau in der Mitte haben. Dies reduzierte die Vibrationen zwar, allerdings nicht genug. Eventuell muss ich auch selbst einen zusätzlichen Lowpass-Filter programmieren.

16. Oktober: Durch das Entfernen des provisorischen Gegenwichts vorne an der Drohne, gingen die Vibrationen etwas zurück. Jedoch sind sie für eine Winkelberechnung mit dem Beschleunigungssensor noch zu ungenau.

18. Oktober: Ich habe einen sogenannten Biquad-Filter einprogrammiert, womit in Zukunft die stark gemessenen Vibrationen herausgefiltert werden können. Dieser Filter enthält ein Lowpass-Filter der weniger Verzögerung zu Folge hat als herkömmliche. Ich habe an verschiedenen Einstellungen herumgeschraubt bis die Vibrationen am besten ausgefiltert wurden. Dafür habe ich noch ein kleines Programm geschrieben, dass die aufgezeichneten Daten in einem Frequenzspektrum aufzeigt.
Ausserdem habe ich die Frequenz, mit welcher der Quadrocopter neue Sensordaten aufruft auf 1 kHz vergrössert, um so gewisse Vibrationen besser herauszufiltern.

19. Oktober: Ich habe mit den aktivierten Filtern einige Tests am Quadrocopter gemacht, um zu schauen, ob sie tatsächlich funktionieren und stark genug sind, jedoch ohne zu flie-

gen. Es scheint alles zu funktionieren. Ausserdem misst der Quadrocopter nun selbst die Spannung des Akkus und sendet ihn an die Fernsteuerung.

20. Oktober: Die Drohne konnte sich erstmals richtig stabilisieren und somit über längere Zeit fliegen. Sie zittert jedoch noch sehr beim Stabilisieren, was an den noch nicht richtig eingestellten PID-Koeffizienten liegt. Zudem habe ich die Auslesung des Barometer integriert. Dieser soll die Drohne später auf konstanter Höhe halten können.

21. Oktober: Durch weitere Testflüge mit der Drohne konnte ich die Werte zum Stabilisieren weiter verbessern. Leider ist sie mir dabei einmal von grosser Höhe abgestürzt, da entweder der Funkkontakt kurzzeitig abbrach und somit nach zwei Sekunden ohne Signal alle Motoren zur Sicherheit ausgeschalten werden oder ich selbst einen Fehler bei der Steuerung machte, was durch die schwere Steuerung auf dem iPhone auch erklärbar ist. Die Aufzeichnung des Fluges habe ich nun so angepasst, dass alle Steuerungseingaben aufgezeichnet werden und so künftig der wirkliche Grund ermittelt werden kann. Eine neue Anzeige auf dem iPhone, die die Signalstärke mit der Drohne anzeigt, soll ebenfalls etwas helfen. Ausserdem habe ich die iPhone-Steuerung vorübergehend etwas vereinfacht, falls es an dem gelegen haben sollte.

23. Oktober: Ich habe das Strommessgerät, das ich auf dem Quadrocopter verbaut habe, kalibriert. Dafür habe ich den Strom mit einem bereits kalibrierten Messgerät gemessen und gleichzeitig den Wert der Strommessung aufgezeichnet. Mit dem Wissen, dass sich der Wert linear zum Strom verhält, konnte ich den Faktor ausrechnen, um auf die Einheit Ampere zu gelangen.

24. Oktober: Ich versuchte die Gründe für die kurzen Verbindungsunterbrüche zu ermitteln. Ich konnte das Problem jedoch trotz mehreren Versuchen nicht richtig rekonstruieren.

25. Oktober: Ich habe Konditionen gefunden, bei denen die Verbindungsunterbrüche (mehr oder weniger schnell) immer auftreten. Dadurch konnte ich nun die Probleme analysieren. Die Fehlersuche führte mich in eine völlig unerwartete Richtung. Scheinbar ist die Verbindung zwischen iPhone und Sender nicht so, wie ich es gerne hätte. Neue Pakete scheinen bei nicht optimaler Verbindung in eine Warteschleife zu kommen, um mehrere Pakete gleichzeitig zu senden und so Bandbreite zu sparen. Da meine Pakete sehr klein sind, geht dies bis zu 10 Sekunden bis diese Pakete alle zusammen gesendet werden.

28. Oktober: Die Übertragung von iPhone zu Sender erfolgt nun mit einem anderen Protokoll (UDP), was viel weniger Verzögerungen hat. Die Verbindungsunterbrüche sind zwar nun seltener geworden, aber noch vorhanden. Scheinbar gibt es noch ein zweites Problem beim Empfangsmodul. Ich habe bereits eine Vermutung, konnte dies aber noch nicht eindeutig verifizieren, da das Problem plötzlich nicht mehr auftauchen wollte. Ich warte also noch, bis ich die Vermutung bestätigen kann, bis ich das vermutete Problem behebe.
Die Stabilität des Quadrocopters konnte ich beim Versuch einen Verbindungsabbruch zu generieren weiter verbessern.

29. Oktober: Um die Höhenmesser zu installieren und auch ein grösseres Landegestell zu befestigen, müssen einige Löcher gebohrt werden. Da das Material Glasfaser ist, testete ich dies zuerst an einem Reststück, was erfolgreich war.

31. Oktober: Damit der Quadrocopter die momentane Höhe automatisch beibehält, habe ich einen Höhenstabiliserungsalgorithmus entwickelt. In einer einfachen Excel-Simulation habe ich den Algorithmus zuvor so angepasst, dass er am besten funktioniert. Ob die Simulation jedoch wirklich mit der Realität übereinstimmt, muss sich noch zeigen.

3. November: Ich habe den Quadrocopter wieder auseinander gebaut, um den Barometer geschützt vor kleinen Windstössen einzubauen. Dabei habe ich auch einen zweiten Sensoren, der Beschleunigungen, Winkelgeschwindigkeiten und Magnetfelder misst, eingebaut. Dieser ist zusätzlich abgefedert, um Vibrationen zu verringern, und weiter entfernt von grossen Strömen, so dass auch der Kompass besser arbeiten kann. Es gibt momentan noch Probleme mit dem Sensor, weswegen ich im Moment noch den direkt angelöteten benutzen muss.
Ich habe ausserdem noch Löcher für das spätere Landgestell und die Hauptplatine gebohrt, das zuvor nur angeklebt war.

4. November: Der zweite Sensor funktioniert jetzt ebenfalls und einen Grund für Verbindungsunterbrüche konnte ich beheben. Ich hoffe, es war der Letzte.

9. November: Nach den Umbauten am Quadrocopter war ich ihn wieder draussen testen, um zu sehen, ob es dadurch irgendwelche Veränderungen gab. Tatsächlich flog er sehr instabil und nicht wie vorher. Nach mehreren Versuchen kam ich zum Schluss, dass wahrscheinlich neue Propeller der Grund sind. Diese haben zwar die gleiche Form, sind jedoch etwas schwerer.

10. November: Ich konnte durch verändern der PID-Koeffizienten den Quadrocopter wieder einigermassen stabil fliegen. Jedoch war draussen starker Wind, was das genauere Einstellen schwierig machte. Zudem habe ich gemerkt, dass das Aufzeichnen der Flugdaten, welche mit der Zeit immer mehr beinhalten, viel Rechenzeit beansprucht, da es direkt auf dem Quadrocopter im CSV-Format auf eine SD-Karte abgespeichert wird. Deswegen lasse ich jetzt nur noch die Rohdaten speichern und konvertiere es später mit einem selbstgeschriebenen Programm auf dem Computer zu einer CSV-Datei. Durch diese Massnahme konnte die Zeit von 1300 us auf 26 us pro Speicherdurchgang reduziert werden.

14. November: Mit einem Testflug konnte ich herausfinden, wie gut der Barometer die Höhe tatsächlich bestimmen kann. Mir ist aufgefallen, dass je stärker die Motoren drehen, desto tiefer die Höhe berechnet wird. Dies liegt wahrscheinlich am Abwind durch die Propeller, die einen höheren Druck verursachen. Mit dem habe ich durchaus gerechnet und habe als Lösung einen Ultraschalldistanzmesser geplant, um für geringe Höhen, die Höhe viel genauer zu bestimmen. Diesen habe ich sogleich provisorisch befestigt und den entsprechenden Code geschrieben, damit die Höhe ausgelesen wird und mit der Höhe des Barometers kombiniert.

16. November: Ich habe weitere Testflüge durchgeführt, um zu sehen, wie sich der Ultraschallsensor verhaltet. Mir ist aufgefallen, dass dieser eine zu hohe Höhe misst, sobald die Propeller sich drehen. Eventuell wird die selbe Ultraschallfrequenz von den Propellern erzeugt wie der Ultraschallsensor produziert und misst.

17. November: Nach einer kurzen Recherche habe ich herausgefunden, dass das Problem vermutlich tatsächlich mit dem Abwind der Propeller zu tun haben. Das Ultraschallsignal hat

Probleme ungestört durch diesen hindurchzukommen. Deswegen habe ich eine andere Version eines Ultraschallsensor ausprobiert, der deutlich besser funktionierte. Zwar lieferte er auch nicht immer einen Wert. Wenn er jedoch keinen messen konnte, gab er einen Fehler zurück statt eine zufällige Höhe wie der vorherige.

18. November: Da sich der Quadrocopter relativ gut stabilisiert, aber noch nicht perfekt, optimierte ich noch einmal die Stabilisierungswerte während dem Flug. Danach war es zwar besser, aber kleine Oszillationen sind immer noch zu bemerken. Ich habe nach den Testflügen herausgefunden, dass diese Oszillationen wahrscheinlich deswegen entstehen, weil der eine Teil des PID-Controllers eine Ableitung berechnet, die ich zusätzlich mit einem Low-Pass-Filter bearbeitete, und deswegen eine leichte Verzögerung hat. Diese Verzögerung ist jedoch so stark, dass sie ihren Zweck, schnelle Veränderungen zu dämpfen, nicht immer erfüllen kann.

20. November: Bei einem Testflug konnte ich tatsächlich ein besseres Flugverhalten feststellen, nachdem ich den D-Teil des PID-Controllers weniger stark gefiltert habe. Ausserdem habe ich den Ultraschallsensor an eine andere Position montiert, damit er weniger Abwind verspürt. Er konnte tatsächlich etwas besser die Höhe messen.

24. November: Ich habe zum ersten Mal beim Flug die Kamera provisorisch montiert und überprüft, ob die Liveübertragung des Bildes, was ich schon zuvor programmiert habe, immer noch problemlos funktioniert, was sie zum Glück tat. Zumindest für diese kleine getestete Distanz.
Danach habe ich den Algorithmus zur Höhenhaltung zum ersten Mal getestet. Ich verwendete dabei nur den Barometer, da beim Ultraschallsensor immer wieder Fehler auftreten. Beim Einstellen der Höhenstabilisierung schoss die Drohne jedoch mit Höchstgeschwindigkeit nach oben für etwa eine Sekunde und fiel danach wie ein Stein von 10 Metern Höhe. In den aufgezeichneten Daten konnte man sehen, dass der Quadrocopter erst viel zu spät auf das Erreichen der Höhe reagiert und dann die Motoren alle ausschaltet, um so schnell wie möglich nach unten zu kommen. Die schwierige Aufgabe wird nun sein, die Werte des Algorithmus so anzupassen, dass er auf der Höhe bleibt, ohne weitere Schäden zu verursachen.

25. November: Da die Höhenbestimmung nahe am Boden mit dem Ultraschallsensor nur unbefriedigend funktioniert hat und auch der Barometer nahe am Boden falsche Werte ausgibt, habe ich einen Ultraschallsensor gekauft, der speziell für Turbulenzen durch Propeller angepasst ist und zudem bis zu 7 Meter messen kann. Er kostete dementsprechend auch 20 Mal mehr als der bisherige.

26. November: Beim genaueren Analysieren des Absturzgrundes vom 24.11. habe ich das Hauptproblem herausgefunden. Im Code habe ich eine Variable verwechselt, die ähnlich hiess. Es erhielt deswegen den Befehl mit 1000 m/s in die Höhe zu fliegen, was es ja zumindest versuchte.

28. November: Ich habe den Höhenstabilsierungscode etwas verändert und ihn nach dem ersten Fehlversuch ohne Propeller getestet. Es schien alles gut zu laufen, allerdings hat es zum Teil eine zu grosse Verzögerung bis der Barometer eine Höhenveränderung erkennt. Wie sich das auswirkt, lässt sich bei einem realen Test sehen.

30. November: Das Protokoll des neuen Ultraschallsensor habe ich in den Code implementiert und getestet, was zumindest auf dem Boden gut funktionierte.

3. Dezember: Ich habe die Höhenhaltungsfunktion nach dem Verbessern dieser erneut getestet. Dies führte wieder zu einem Absturz, da der Quadrocopter erneut in die Höhe schnellte, allerdings deutlich weniger als letztes Mal. Bei der Analyse der aufgezeichneten Daten zeigt sich, dass der Barometer weiterhin die Höhe mit einer Verzögerung von bis zu einer Sekunde misst. Ein weiteres Problem war der Schnee, der beim Absturz sich über die Elektronik verteilte. Schäden konnte ich noch keine feststellen.

7. Dezember: Die Verzögerung lag an einer Einstellung des Barometers, der die gemessenen Daten innerhalb des Barometers sehr stark filterte. Dies habe ich nun ausgeschaltet. Zudem habe ich einen Algorithmus entwickelt, der die Ausrichtung der Drohne absolut hält. Zuvor versuchte sie lediglich die Geschwindigkeit bei 0°/s zu halten. Dadurch entstanden jedoch schnell Fehler. Um den absoluten Winkel der Ausrichtung herauszufinden werden die Daten des Magnetfeldmessers mit dem Gyroskop kombiniert.

9. Dezember: Implementierung der Funktion zur Steuerung des Kamerawinkels am Quadrocopter über die App. Der Winkel kann durch Drücken eines Knopfes und der Neigung des Smartphones eingestellt werden.

12. Dezember: Der Flugcontroller bezieht nun die Daten von jeweils beiden Beschleunigungssensoren und Gyroskopen, die verbaut sind. Es werden allerdings weiterhin nur die einen benutzt, aber die Daten der anderen auf die SD Karte aufgezeichnet. Nach einem Testflug werde ich versuchen anhand der aufgezeichneten Daten die Nützlichkeit des Kombinieren der Daten zu analysieren.

13. Dezember: Testflug mit Aufzeichnung beider Beschleunigungs- und Gyroskopsensoren gleichzeitig. Bei der Analyse kam heraus, dass die externen Sensoren, weniger Vibrationen hatten. Die Vibration sind jedoch bei den Sensoren etwa gleichmässig zu sehen. Die Kombination zweier gleicher Sensoren würde daher kein besseren Messwert zur Folge haben.

16. Dezember: Beim Fliegen der Drohne fällt auf, dass der Beschleunigungssensor eine totale Beschleunigung von weniger als 1 g in der meistens Zeit misst. Auch Werte darüber sind meist nicht deutlich stärker. Fliegt und bewegt sich die Drohne nicht, wird ein korrekter Wert von 1 g angezeigt. Dies würde bedeuten, dass die Drohne die meiste Zeit nur nach unten beschleunigt, was nicht den Beobachtungen, dass die Drohne fliegt, entspricht. Daher wird wahrscheinlich das Problem am Sensor liegen. Der zweite Beschleunigungssensor hat allerdings das selbe Problem. Diese Beschleunigungsdaten wären wichtig, um die Geschwindigkeit des Sinken oder Steigens genauer zu ermitteln.

18. Dezember: Eine mögliche Ursache des Problems wäre, dass die starken Wechselströme, die durch die Motoren erzeugt werden, die Stromzufuhr der Sensoren entweder direkt oder über elektromagnetische Interferenzen stört. Tatsächlich konnte ich bei Tests ohne Abheben eine oszillierende Spannung am Sensor messen. Die Spannung schwankte im Bereich von 100 mV. Ohne angeschaltete Motoren nur im Bereich von 20 mV.

19. Dezember: Um das Beschleunigungssensor-Problem zu lösen, wollte ich versuchen, den externen Sensor statt an 3.3 V, das er tatsächlich verwendet, an 5 V anschliessen. Dies funktioniert weil der externe Sensor einen Spannungswandler an Bord hat, der die 5 V auf

3.3 V herunterregelt. Davon erhoffe ich mir, dass die Spannung des Sensor stabiler wird, da sie vor Ort erzeugt wird. Das selbe tat ich auch für den Barometer. Um dies durchzuführen, musste ich jedoch fast die komplette Drohne auseinanderbauen. Beim Zusammenbau habe ich zusätzlich noch das erweiterte Landegestell montiert. Dadurch wird das Innere der Drohne im Schnee auch nicht nass. Nach dem Zusammenbau habe ich jedoch bemerkt, dass die Kommunikation zwischen Sensoren und Flugcontroller nach einer gewissen Zeit abbricht. Ich hatte zuvor zwar an der offenen Drohne die Sensoren getestet, allerdings nicht über eine längere Zeit.

21. Dezember: Ich vermute, dass das Problem daran lag, dass der Barometer die Kommunikationslinien auf 5 V betrieb und alle anderen auf 3.3 V. Daher habe ich die Drohne geöffnet und den Barometer wieder mit 3.3 V verbunden. Trotzdem gab es nach längerer Zeit noch Probleme. Nach vielen Recherchen und Versuchen habe ich herausgefunden, dass das Problem nicht auftritt, wenn der festverlötete Beschleunigung-/Gyroskopsensor von der Software nie ausgelesen wird. Da ich gesehen habe, dass ich den nicht unbedingt brauche, habe ich ihn deaktiviert. Den Grund dafür habe ich allerdings nicht herausfinden können.

23. Dezember: Ich habe am Landegestell provisorisch das Gimbalsystem montiert. Das soll für eine wackelfreie Aufnahme der Kamera sorgen, indem es die Bewegungen aktiv mit Motoren entgegenwirkt. Zudem erlaubt es auch die manuelle Bewegung der Kamera, das ich bereits implementiert habe. Dieses Gimbal musste ich jedoch zuerst korrekt einstellen, damit es für meine Zwecke am besten funktioniert.

24. Dezember: Ich führte zwei Testflüge durch. Beim ersten Flug wollte ich überprüfen, ob das Landegestell dem Quadrocopter keine Probleme bereitet, was auch so war. Zudem wollte ich die aufgezeichneten Daten des Beschleunigungssensor auswerten. Das Problem der falsch gemessenen Beschleunigung hat sich allerdings deutlich verschlimmert. Nun werden im Schwebeflug zum Teil -0.2 g statt 1 g gemessen. Dies ist ein Fehler von 1.2 g. Der Betrieb über 5 V scheint die Situation des Beschleunigungssensor aus unbekannten Gründen schlimmer zu machen. Man kann daraus jedoch sehen, dass das Problem des Sensors wahrscheinlich nicht mit den mechanischen Vibrationen zu tun hat, sondern eher mit der Stromzufuhr.
Anschliessend habe ich einen zweiten Flug durchgeführt mit der Kamera und dem Gimbal. Beim Starten des Quadrocopters schalteten sich die Motoren des Gimbals jedoch aus. Da sowohl Beschleunigungssensor wie auch Gimbal etwa gleichzeitig Probleme bekommen, kann ich annehmen, dass ein gemeinsames grösseres Problem vorliegt.

25. Dezember: Ich habe am Quadrocopter zu Testzwecken eine zweite kleine Batterie befestigt, die ausschliesslich das Gimbalsystem mit Strom beliefern soll. Beim Starten der Motoren hat sich der Gimbal dieses Mal nie ausgeschaltet. Das bestätigt die Theorie, dass die Stromzufuhr das Problem ist.

26. Dezember: Ich schloss den Beschleunigungssensor wieder zurück auf 3.3 V an, was zuvor zu einem viel kleineren Fehler führte. Das Problem war jetzt jedoch gleich gross wie unter 5 V. Ich nehme daher an, dass der Sensor aus irgendeinem Grund beschädigt wurde. Also verwendete ich provisorisch nur noch den internen Beschleunigungssensor, damit nicht auch die Winkelberechnung durch die fehlerhaften Beschleunigungsdaten gestört wird. Der interne Sensor hat einen viel kleineren Fehler. Ich versuchte die Motoren am Boden unter

verschiedenen Geschwindigkeiten laufen zu lassen. Die aufgezeichneten Daten stellte ich dann in einer Grafik dar. Ich stellte Motorengeschwindigkeit dem Fehlers der Beschleunigung gegenüber. Die Punkte wurden einigermassen auf einer Linie angesiedelt. Trotzdem gab es viele Ausreisser, was eine erhoffte Softwarekorrektur des Fehlers verunmöglichte.

28. Dezember: Ich habe mit einem grossen Kondensator, den ich direkt neben den Beschleunigungssensor angelötet habe, versucht eventuelle Spannungsschwankungen, die vielleicht zum Problem führen könnten, zu beheben. Dies zeigte jedoch keine Wirkung.

30. Dezember: Ich habe den neuen Ultraschallsensor montiert und wollte einen Testflug machen. Es trat jedoch das selbe Problem auf wie zuvor beim billigen Ultraschallsensor. Sobald die Propeller zu viel Abwind generierten, wurden zufällig kleine Werte gemessen, obwohl dieser Sensor versprach, genau diese Schwierigkeit meistern zu können. Auch in der Höhe konnte der Ultraschallsensor nur zweimal in 10 Sekunden den richtigen Wert ermitteln. Auch hier versuchte ich Spannungsschwankungen mit einem Kondensator beim Sensor zu eliminieren, doch auch das brachte keine Verbesserung.

31. Dezember: Unter den Quadrocopter habe ich eine zweite kleine Batterie angebracht, um sämtliche Sensoren nur noch damit mit Strom zu versorgen. Ich erhoffte mir dabei, das Problem mit dem Beschleunigungs- und Ultraschallsensor zu lösen, indem die Spannungsschwankungen durch die Motoren die Sensoren nicht mehr beeinflussen können, da es sich um einen separaten Stromkreislauf handelte. Dies brachte leider keinen Erfolg.

Bestätigung der Eigenständigkeit

Der/die Unterzeichnete bestätigt mit Unterschrift, dass die Arbeit selbständig verfasst und in schriftliche Form gebracht worden ist, dass sich die Mitwirkung anderer Personen auf Beratung und Korrekturlesen beschränkt hat und dass alle verwendeten Unterlagen und Gewährspersonen aufgeführt sind.

Ort: _____ Datum: _____

_____