Maturitätsarbeit an der Kantonsschule Zürich Nord

Regelungstechnik PID-Parametrisierung anhand eines selbstgebauten Quadrocopters

—

26. November 2019

Inhaltsverzeichnis

1 Einleitung

Ein Quadrocopter ist ein Flugobjekt mit vier Propellern. Schon in den 1920er Jahren wurde diese Technik eingehend untersucht. Der Luftfahrtpionier Étienne Œhmichen, der 1920 den Oehmichen No. 2 gebaut hatte, machte schon Versuch mit Flugobjekten mit vier Propellern. Damals waren die Propeller elastisch und man konnte mit Seilzügen den Anstellwinkel der Propeller einstellen. Er konnte damit einige Rekorde aufstellen.[10]

In der heutigen Zeit finden Quadrocopter viele Anwendungen, wie z.B. bei Suchaktionen, im Militär oder auch als Spielzeug. Mit der schnellen Entwicklung von Mikroprozessoren wurden Quadrocopter immer kleiner und einfacher zu realisieren. Auch können damit teurere Helikopterflüge für Kartographie oder Luftaufnahmen billiger realisiert werden.[11]

Die Intention zu dieser Arbeit folgt aus einem Projekt aus dem Freifachkurs Robotik. Damals kam der Raspberry Pi zum Einsatz. Da dieser ein Betriebsystem parallel zum Flightcontroller am laufen hat, war die Regelung des Flightcontrollers zu träge. Auch war das Stueuersignal an den Motoren viel zu zitterig und so war waren diese unbrauchbar. Desswegen wird auch ein neuer Chip evaluiert, der den Echtzeitanforderungen entspricht und eine richtige Hardware Pulsweitenmodulation (PWM) hat.

Die Fragestellung lautet, wie die Regelung des Flightcontrollers implementiert werden muss, damit diese funktioniert und welche Faktoren diese stören und wie schnell die Daten verarbeitet werden müssen. Auch wird untersucht , welche Lösungen es für diese Probleme gibt und es wird vcersucht diese zu implementieren.

In der Arbeit wird der Quadrocopter selber gebaut, mit dem Hauptziel den Flightcontroller selbst zu designen und zu programmieren. Dannach werden Versuche gemacht, die bestimmten wie der PID-Regler stabiler gemacht wird und wie ungleichheiten, die durch Vibrationen enstehen, ausgeglichen werden können. Die Fernbedienung wird auch selbst gebaut und dient auch dazu Daten wärend des Fluges auszuwerten.

2 Theorie

2.1 Funktionsweise eines Quadrocopters

Man könnte meinen, dass man einen stabilen Flug erreichen kann, wenn der Schub der vier Rotoren gleich ist. Aber dies ist leider nicht möglich, da viele Umwelteinflüsse einen stabilen Zustand verhindern wie z.B. Wind, unterschiede in den Rotoren oder Motoren, Assymetrie des Rahmens etc. So muss ein Computer, der Flightcontroller (FC), das Gleichgewicht erhalten, indem dieser den Schub der vier Motoren so regelt, dass der Quadrocopter im Gleichgewicht ist oder dieser einen anderen bestimmten Winkel hält. Abbildung 1 zeigt die Rotationsrichtung der Motoren. Die benachbarten Motoren müssen eine gegensätzliche Rotation haben, da sonst der Quadrocopter ins Rotieren gerät. So wird das Drehmoment des anderen Motors ausgeglichen. Auch wird gezeigt, dass man den Quadrocopter mit erhöhen bzw. erniedrigen der Motorleistung der richtigen Motoren, diesen in die gewünschte Richtung steuern kann. Kernelement dieser Steuerung ist der PID-Regler.

2.2 Was ist ein Regler?

Eine Regelung ist eine Steuerung mit Rückkoppelung. Ein Wert z.B. die Drehzahl eines Motors überwacht und je nach gewünschter Drehzahl, das Drehmoment des Motors geregelt, dass er auch diesen halten kann, auch wenn eine Last am Motor hängt.[7]
Im Reglekreis (siehe Abb.2) wird der Ist-Wert z.B. mit einem Sensor gemessen. Dieser wird mit dem Soll-Wert verglichen, um die Regelabweichung zu bestimmen. So kann die Reglegrösse bestimmt werden, um so das System dem Soll-Wert anzugleichen.

Die Regelabweichung kann dabei einfach mit der Differenz des Ist-Wertes x mit dem Soll-Wertes w bestimmt werden.[7]

$$e(t) = w - x \tag{1}$$

In Abbildung 3 sieht man, wie vorher beschrieben, wie ein solcher Regelkreis funktionert. In der Regelungstechnik wird versucht einen solchen Regelkreis mathematisch zu modellieren. In diesem Kapitel wird nur der PID-Regler und seine Komponenten besprochen, da dieser häufig bei Quadrocoptern benutzt wird.[7]

Der PID-Regler besteht aus mehreren Relgern und dem D-Glied. In der Folge wird beschrieben wie die einzelnen Komponenten wirken.

2.2.1 P-Regler

Der P-Regler wirkt linear. Dieser Regler gibt die Regelabweichung verstärkt und unverzögert mit dem Faktor Kp weiter (Vgl. 2). Das Problem dabei ist, dass diese Abweichung bleibend ist und somit den Soll-Wert über- bzw. unterschiesst. Dieser Regler wirkt schnell.[7]

$$y(t) = Kp \cdot e(t) \tag{2}$$

2.2.2 I-Regler

Beim Integralregler wird die Regelabweichung über die Zeit summiert und mit dem Faktor Ki verstärkt (Vgl. 3). Dabei werden Abweichungen volständig eliminiert, da dieser Regelwert immer anwächst, solange die Regelabweichung nicht Null ist. Dieser Regler wirkt langsam.[7]

$$y(t) = Ki \int_0^t e(t)dt \tag{3}$$

2.2.3 D-Glied

Das Differenzialglied schaut auf die Differenz der Regelabweichung zur vorherigen Regelabweichung und wird mit dem Faktor Kd verstärkt (Vgl. 4). Desshalb reagiert dieser sehr schnell und gibt den beiden anderen Vorhaltezeit.Differenzialglied ist kein Regler da es alleine nichts regleln kann, sondern nur auf Veränderungen in der Regelabweichung reagiert.[7]

$$y(t) = Kd \cdot \dot{e}(t) \tag{4}$$

2.2.4 PID-Regler

Beim PID-Regler werden die Eigenschaften der einzelnen Regler und dem D-Glied vereint (Vgl. 5).

$$y(t) = Kp \cdot e(t) + Ki \int_0^t e(t)dt + Kd \cdot \dot{e}(t) \tag{5}$$

2.3 Digitaler Low-Pass Filter

Ein Low-Pass Filter filtert hohe Frequenzen heraus. Der Filter ist aus der Elektronik oder der Tontechnik bekannt. Beim Quadrocopter wird ein solcher Filter für das Unterdrücken des Rauschens des Inertialsensors verwendet. Dieses Rauschen wird durch die Vibrationen der Motoren erzeugt und liegt ca. bei 133Hz.

Beim vorliegenden Digitalen Low-Pass Filter (DLPF) wird ein analoger RC-Low-Pass Filter emuliert, wie der in Abbildung 4.[9]

Im letzten Kapitel wurde der PID-Regler eingeführt, doch Vibrationen verursachen Störungen in der Regelung und führen so zu falschen Korrekturen. Zum Beispiel verursachen diese beim D-Glied starke Auschläge, wenn diese hohe Amplituden haben. Auch wird dieser dann unbrauchbar, da die Regelabweichung durch die Vibrationen verfälscht wird. Ein Filter versucht diese Vibrationen zu dämpfen und verhindert so, starke Ausschläge. Wie man in 6 sieht, ist die Eingang- und Ausgangsspannung vom Widerstand und von der Stromstärke abhängig.

$$v_{\text{in}}(t) - v_{\text{out}}(t) = R \cdot I(t) \tag{6}$$

Der Strom kann man auch abhängig vom Kondensator und der Ausgangsspannung darstellen, wie man in 7 und 8 sieht.

$$Q_c(t) = C \cdot v_{\text{out}}(t) \tag{7}$$

$$I(t) = \dot{Q}_c(t) = C \cdot \dot{v}_{\text{out}}(t) \tag{8}$$

So kann man die Differenzialgleichung 6 wie 9 darstellen.

$$v_{\text{in}}(t) - v_{\text{out}}(t) = RC \cdot \dot{v}_{\text{out}}(t) \tag{9}$$

Jetzt wird $v_{\text{in}}(t)$ mit x_i und v_{out} mit y_i dargestellt und die Differenzialgleichung diskretisiert zu der Differenzengleichung 10.

$$x_i - y_i = RC \cdot \frac{y_i - y_{i-1}}{\Delta t} \tag{10}$$

Die Gleichung 11 (Ganze Auflösung siehe Anhang 2 Auflösung 1) wird nach y_i umgeformt und man kann $^{\Delta t}/_{\Delta t + RC}$ mit α und $^{RC}/_{\Delta t + RC}$ mit $\alpha - 1$ substituieren wie in 12 gezeigt.

$$yi = x_i \frac{\Delta t}{\Delta t + RC} + y_{i-1} \frac{RC}{\Delta t + RC} \tag{11}$$

$$yi = \alpha x_i + (1 - \alpha) y_{i-1} \quad \text{mit} \quad \alpha := \frac{\Delta t}{\Delta t + RC} \tag{12}$$

RC kann mit der Formel für den Low-Pass Filter durch f_c dargestellt werden wie unten gezeigt wird.

$$f_c = \frac{1}{2\pi RC}$$
$$RC = \frac{1}{2\pi f_c}$$

Jetzt kann auch das RC in der Definition von α ersetzt werden, damit die Gleichung nicht mehr von RC abhängig ist (Ganzer Beweis siehe Anhang 2 Beweis 1).

$$\alpha = \frac{2\pi f_c \Delta t}{1 + 2\pi f_c \Delta t}$$

3 Versuchsaufbau

3.1 Quadrocopter

3.1.1 Motoren

Die verwendeten Motoren sind sogenannte bürstenlose Motoren. Diese werden mit einem Electronic Speed Controller (ESC) angesteuert, dieser wandelt die Steuerimpulse in die richtige Spannung für den Motor. Da ihre RPM von Spannung abhängig ist, wird einen Kv-Wert angegeben. So kann man mit der Formel

$$a = Kv \cdot U \quad mit \quad [a] = \text{rpm} \tag{13}$$

die Rotationen pro Minute berechnen.

Beim vorliegenden Quadrocopter handelt es sich um ReadyToSky RS2205 Motoren mit einem Kv von 2300. Diese Motoren wurden ausgesucht, da sie für genug Leistung bringen für einen Quadrocopter mit einer Rahmenlänge von 250mm.

3.1.2 Akku

Der Akku (Abb.**??**) versorgt alle Komponenten, die Strom brauchen mit Strom. Im Quadrocopter wird ein LiPo Akku benutzt, der drei Zellen in Serie geschaltet hat mit einer Kapazität von 2.4Ah. Die miminale Spannung beträgt 11.1V. Die Entladerate wird C-Rate angegeben. Mit der Kapazität multipliziert wird der maximale Entladestrom ausgerechnet.[3]

$$I_{\max} = Q \cdot C \tag{14}$$

Man benutzt in Quadrocoptern meist LiPo Akkus, da sie eine hohe Energiedichte besitzen und viel Leistung abgeben können. Dies aber hat auch zur Folge, dass man diese Akkus sorgfältig behandeln muss, da sie im Extremfall, Feuer fangen oder explodieren können. So darf man die einzelnen Zellen nicht unter einer Spannung von 2.7V fallen lassen. Desshalb wird der Akku auch an einem Akkuüberwacher angeschlossen, der piepst, wenn der Akku verbraucht ist. Auch braucht man ein Balanceladegerät, das kontrolliert, dass alle Zellen gleichmässug geladen werden. Eine hohe Kapazität garantiert nicht immer eine längere Flugzeit, da mehr Kapazität auch mehr Gewicht bedeutet.[3]

3.2 Der Flightcontroller

Der Flightcontroller ist das Herz des Quadrocopter. Dieser liest alle Sensoren und gibt den ESCs die Steuerimpulse. In dem Kapitel wird zuerst in die wichtigsten Komponenten eingegangen, dann wird der Design-, Bau und Propgrammierprozess beschrieben.

3.2.1 Komponenten

3.2.1.1 Microcontroller

Der Microcontroller, der auf dem FC benutzt wird, ist ein ARM-Cortex-M7 STM32F722RET6 Microcontroller von STMicroelectronics. Dieser besitzt alle Schnittstellen um die einzelnen Sensoren und andere Komponenten auszulesen und anzusteuern. Da dieser nicht genug Pins hat, wird noch ein zweiter Microcontroller benutzt und zwar der ATMega328P, der auch auf dem Arduino/Genuino UNO zu finden ist. Dieser liest den Drucksensor und einige ADCs aus und kommuniziert mit dem I2C-Protokoll mit dem STM32.

Microcontroller können anders als Mikroprozessoren, der auf dem Raspbery Pi verbaut ist, ohne Zusatz Pehpetrie benutzt werden, da sie schon einen Flash, RAM oder ähnliches besitzen.

3.2.1.2 Sensoren

Die wichtigsten Sensoren des Quadrocopter sind der Gyrosensor und der Beschleunigungssensor. Der Gyrosensor misst die Winkelbeschleunigung der Achsen. In dem FC wurde eine ICM-20689 von TDK InvSense benutzt. Dieser hat beide Sensoren in einem Chip integriert und wird per SPI-Protokoll ausgelesen. Als Backup wird ein MPU-6050 benutzt, auch vom gleichen Hersteller, dieser wird mit dem langsameren I2C-Protokoll ausgelesen. Beide sind sogenannte MEMS-Sensoren. MEMS bedeutet microelectromechanical systems. Das bedeutet, dass diese Sensoren auf einer mechanischen Basis funktionieren.[2]. Man ätzt die Sensorstrukturen und deren auslese Schaltung auf einen Chip. Abbildng 5 zeigt die eingeätzte Siliziumstruktur des MPU6050.

Da der Gyrosensor nur Winkelgeschwindigkeit, bei den Sensoren $\frac{^\circ}{s}$ und nicht $\frac{rad}{s}$ misst, muss diese noch mit der Loop-Zeit multipliziert werden, um den Winkel zu bekommen (vgl. 15).

$$w = \omega \cdot dt \quad mit \quad [w] = ^\circ \quad und \quad [\omega] = \frac{^\circ}{s} \tag{15}$$

Wenn man den Absoluten Winkel alleine mit dem Gyrosensor misst hat dieser über die Zeit einen Drift. Der durch das Rauschen des Sensors verursacht wird. Auch kann nicht der Anfangswinkel bestimmt werden, da dieser nur die Winkelgeschwindigkeit misst und somit keine Referenz zum Normalvektor hat, somit müsste der Quadrocopter immer von einer horizontalen Ebene starten. Deswegen braucht man da den Beschleunigungssensor. Dieser misst die Beschleunigung auf allen Achsen relativ zur Erdbeschleunigug. Beim Stillstand auf einer horizontalen Ebene misst er also $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \vec{g}$. Man kann einfach die Norm des Beschleunigungsvektors $\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$ nehmen und mit der Beschleunigung an der gewünschten

Achse verrechnen, um dann den Winkel zu bekommen (vgl. 16).

$$w = \arcsin(\frac{a_{x,y}}{\vec{a}})\vec{g} \cdot \frac{180}{\pi} \quad mit \quad [w] = ^\circ \tag{16}$$

Nebenbei muss man beachten, dass C/C++ den Winkel in *rad* berechnet und man diesen noch zu *grad* konvertieren muss. So hat man den absoluten Winkel mit zwei Sensoren gemessen und muss diese nur noch Kombinieren. Dies geschieht mit Hilfe des Komplementärflilters(vgl. 17). (Man muss auch beachten, dass der Absolute Winkel nur für die Nick-und Rollachse berechnen kann da, für die Gierachse noch ein Magnetometer für eine Null-Referenz nötig wäre (Achsenbezeichnung siehe Abb.**??**).

$$w_{\text{komp}} = 0.9996 \cdot w_{\text{Gyro}} + 0.0004 \cdot w_{\text{Accl}} \quad mit \quad [w] = ^\circ \tag{17}$$

Theoretische Überlegungen führen dazu, dass dem Winkel aus dem Beschleunigungssensor ein tiefer Faktor gegeben, da dieser Winkel nicht mehr genau stimmen wird, wenn der Quadrokopter in eine Richtung beschleunigen würde, dies wird erst bei starken Beschleunigungen der Fall sein, sonst stimmt dieser Wert ungefähr. Aber so kann man den Anfangswinkel des Quadrokopter bestimmen, ihn auch aus der Hand starten lassen und bekommt auch noch stabilere Werte.

Desweiteren muss auch noch beachtet werden, dass bei einer Bewegung um die Gierachse, der Sensor keine Bewegung auf der Roll- und Nickachse misst. Desshalb müssen die Werte dieser Achsen bei einer Bewegung um die Gierachse mithilfe der Rotationsmatrix (vgl. refrotmax) ineinander übersetzt werden.

$$\begin{pmatrix} pitch' \\ roll' \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} * \begin{pmatrix} pitch \\ roll \end{pmatrix} \tag{18}$$

Somit muss man die Achsen wie folgt verrechnen (vgl 19 und 20).

$$roll' = \sin(\alpha) * pitch \tag{19}$$

$$pitch' = \sin(\alpha) * roll \tag{20}$$

3.2.1.3 Empfänger

Als Empfänger wir ein NRF24L01 Breakoutboard mit PA und LNA benutzt. Das ist ein 2.4GHz Sender und Empfänger der Firma Nordic Semiconductor. Die Kommunikation erfolgt digital und der Chip besitzt auch eine Zyklische Redundanzprüfung (CRC). Damit wird kontrolliert, ob ein Datenpaktet fehlerhafte Bits besitzt. Allenfalls wird es repariert oder verworfen. PA heisst Power Amplifying und bedeutet, dass das Signal verstärkt wird und so grössere Reichweiten erreicht werden können und LNA bedeutet Low Noise Amplifying und bedeutet, dass schwache Empfangssignale verstärkt werden.[6] Vom Hersteller wird eine Reichweite von bis zu einem Kilometer versprochen aber in einer Wohnsiedlung kommt man mit Sichtkontakt bis zu 300m, was ausreichend ist.

3.2.2 Leiterplatte

Auf Leiterplatte oder auch Printed Circuit Board (PCB) wird die Schaltung des FC realisiert. Dieses besteht aus Kunststoff mit den aufgeätzten Kupferbahnen und den verzinnten Lötstellen.

3.2.2.1 Design

Zuerst wurde überlegt, was für ein Microcontroller benutzt werden sollte. Bei den ersten Versuchen wurde ein Raspberry Pi benutzt, dort gingen die Versuche schief, da dieser kein konstantes Regelverhalten aufzeigte und dessweiteren ein Betriebsystem neben dem Flightcontroller lief. Ausserdem war das PWM Signal für die ESC's nicht gut und zitterten. Desswegen wurde im Internet erkundigt, was für Microcontroller in den meisten Quadrocopter verbaut sind. So wurde die Webseite von Oscar Liang entdeckt [4]. Dort waren die einzelnen Microcontroller mit ihren Vor-und Nachteilen aufgelistet. Für die Arbeit wurde entschieden einen STM32F7xx Microcontroller zu benutzen. Als nächstes wurde überlegt, was für ein Gyro-und Beschleunigungssensor benutzt werden sollte. Vom letzten Versuch blieb noch eine MPU-6050 IMU übrig, aber da das I2c Protokoll langsam war, wurde entschieden diesen als Backup zu benutzen und es wurde recherchiert, was für IMU's die komerziellen Flightcontroller benutzen [1]. Als Empfänger wurde ein nRF24L01-Breakoutboard benutzt. Um in den Versuchen Daten aufgezeichnet werden können, wurde noch ein microSD-Karten Schnittstelle eingebaut, um Daten zu loggen. Damit eine Serielle-Verbindung zum PC hergestellt werden kann, wird ein FTDI-Chip (FT232RL) benutzt, der das UART-Protokoll für den PC übersetzt. Als Nebenmicrocontroller wurde ein ATMega328 benutzt, dieser ist für das Auslesen der Höhensensoren und des PulsPausenModulations-Signal (PPM) zuständig. Dieses wird benötigt, falls die selbstgebaute Fernbedienung nicht funktionieren würde und eine Kommerzielle stattdessen benutzt werden müsste. Als Höhensensor wurden ein Barometer-IC (BMP280) und Ultraschallsensoren benutzt. Da die ESC's einen 5V Versorgungsausgang haben, muss diese für die IC's auf 3.3V gereglet werden. Dies wird mit einem LM3940 Spannungsregler ermöglicht.

Die Leiterplatte wird mit den Program Eagle von Autodesk designt. In dem Programm kann man die einzelnen Bauteile zusammensetzen und verbinden (Siehe Abb.**??**).Nach dem Datenblatt der einzelnen IC's wurden die nötigen Kondensatoren und Wiederstände gesetzt. Die Schaltbilder für die einzelnen IC's, die nicht schon in der Standartbibliothek vorhanden sind, wurden aus dem Internet heruntergeladen. Nach mehreren Kontrollen wurde die Leiterplatte bei JLCPCB und die einzelenen Teile bei LCSC, Arrow, RS und Conrad bestellt. Dazu wurde noch eine Schablone für das SMD-löten bestellt.

3.2.2.2 Bau

Beim Zusammensetzen des FC wird ein spezielles Lötverfahren gebraucht, und zwar das SMD reflowing. Damit man die Komponenten nicht per Hand löten muss, wird mit einer Schablone Lötpaste, die Flussmittel und das Lötzinn erhalten, aufgetragen. Dann wird die Schablone entfernt und die Teile werden plaziert. Dies geschieht im industriellen Rahmem mit sogenannten "Pick and Place"Maschienen. Später wird die Leiterplatte in den Ofen geschoben und nach einer bestimmten Temperaturkurve (vgl. Abb.**??**) erhitzt, bis die Komponenten verlötet sind[8]. Da die Leiterplatte doppelseitig ist, kann man hier nicht auf beiden Seiten das Reflow-Verfahren benutzen. Deshalb wurde Zuerst auf der Rückseite Lötpaste für den microSD-Karten Slot und dem Quarzoszillator aufgetragen. Nachher wurden die Komponenten platziert. Dann wurden diese mit dem Heissluftföhn verlötet (siehe Abb.**??**). Um das Herausfallen der Teile im Ofen zu verhindern werden sie mit hitzebeständigen Klebeband fixiert.
Dann wurde auf einer Holzplatte Löcher an den Positionen der Verlöteten Komponenten

gebohrt. Damit wird erreicht, dass man die Vorderseite, Plan auf dieser Holzplatte gelegt werden kann. Mit den restlichen PCBs, man kann nur fünf auf einmal bestellen, wurde ein Rahmen erstellt, um so das teilweise gelötete Board zu fixieren. Dann wurde die Schablone ausgerichtet und festgemacht. So konnte man mit Lötpaste und einem Spachtel, die Löcher und somit auch die Lötstellen mit Lötpaste füllen (siehe Abb.**??**). Dann wurde die Schablone entfernt. Mithilfe des Mikroskops wurden die Teile dann auf der Leiterplatte plaziert (siehe Abb.**??**). Im Ofen wurden sie dann ungefähr nach der Temperaturkurve gebacken und so war die Vorderseite gelötet (siehe Abb. **??**).
Die Hinterseite wurde dann per Hand fertig gelötet. Dann wurden nur noch die Konnektoren angelötet und der Flightcontroller ist fertig zum programmieren (siehe Abb.**??**).

3.2.3 Programmierung

Der FC wird mit C und C++ programmiert. Dabei wurde die STM32CubeIDE von STMicroelectronics benutzt, die eine Eclipse IDE mit den entsprechenden Bibliotheken vorinstalliert ist. Es wurden für die einzelnen Komponenten auch andere Bibliotheken benutzt, die z.t. für den STM32 umgeschrieben werden mussten. Die Bibliothek für die IMU wurde aber selbst programmiert. Dabei kamen mehrere Probleme auf, z.B. dass das ChipSelect-Signal zu früh auf High ging. In diesem Kapitel geht es um die grössten Hürden und wichtigsten Abläufe bei der programmierung des Quadrocopters. Im Listening 1 wird der grobe Ablaufplan des Programms gezeigt. (Der ganze Quellcode und alle Projektdateien findet man im Link im Anhang 1.)

```
int main(){
  initPID_Values();
  initPeriphial();
  initIMU();
  initnRF24();
  initSD();

  while(true){
    loop();
  }
  return 0;
}

void loop(){
  start = elapsedticks();

  calcTrueangle();
  writeSD() //every 200th loopcycle; deactivated when not used
  readRadio();
  calcError();
  calcPID();
  setMotor();

  stop = elapsedticks();
  looptime = stop - start;
}

void setMotor(){
  if(throttle < 100){
    motspeed[0:3] = 1024;
  }else{
    motspeed[0] = throttle + roll + pitch + yaw;
```

```
33      motspeed[1] = throttle + roll - pitch - yaw;
34      motspeed[2] = throttle - roll - pitch + yaw;
35      motspeed[3] = throttle - roll + pitch - yaw;
36    }
37    PWM_SET(M1, motspeed[0]);
38    PWM_SET(M2, motspeed[1]);
39    PWM_SET(M3, motspeed[2]);
40    PWM_SET(M4, motspeed[3]);
41  }
```

Listing 1: Programmablauf Pseudocode

Die IMU wird per Interrupt ausgelesen. Ein Timer gibt einen 500Hz Takt aus und nach diesem liest der Microcontroller die IMU aus.

3.2.3.1 Motoren und Timing

Die ESC's der Motoren müssen mit dem OneShot125 Protokoll[5] angesprochen werden. Bei dem wird ein PWM-Signal von 2khz erzeugt. Der Nullpunkt, bei dem die Motoren aus sind, liegt bei einer Periode von 125us bzw. einem PWM von 25% (vgl. Abb.**??**). Die Motoren erreichen ihre maximale Drehzahl bei einer Periode von 500us bzw. bei einem PWM von 50%.

3.2.3.2 PID Implementierung

Der PID-Regler wird für den absoluten Winkel und für die Winkelgeschwindigkeit implementiert, wobei man die Gierachse nur mit der Winkelgeschwindikeit implementieren kann, da keine Referenz besteht, z.b durch ein Magnetometer. Die Regelungsdifferenz und Regelungskorrektur wird jeden Loop-Zyklus berechnet, der mit ca. 1.5kHz läuft.

3.3 Fernbedienung

Die Fernbedienung (Abb.??) wird aus Holz und PLA gebaut. Der Rahmen wurde mittels eines 3D-Drucker gedrukt. Das Frontpanel ist aus Holz gelasert worden. Die Schieberegler wurden aus einer kaputten Fernbedienung genommen. Auch besitzt diese ein LCD-CharDisplay mit 20x4 Zeichen. Dies wird benutzt um den genauen Ausschlag der Schieberegler, die Sensorwerte und die PID-Werte anzuzeigen bzw. zu bearbeiten. Als Microcontroller wird ein STM32F767ZI benutzt, der auf einem Nucleo-144 Protoboard verbaut ist. Listening 2 zeigt den Ablaufplan des Programmes.

```
int main(){
  initPeriphial();
  initADC();
  initnRF24();

  while(true){
    loop();
  }

  return 0;
}

void loop(){
  readADC();
  sendData();

  menu_print();
}

void menu_print(){
  MENU_1:
    print_ADC_VAL();
  MENU_2:
    print_ACK_VAL();
  MENU_3:
    print_PID_ROLL_PITCH(); //<-- ALSO EDIT VALUES ON 0% Throttle
  MENU_4:
    print_PID_YAW(); //<-- ALSO EDIT VALUES ON 0% Throttle
}
```

Listing 2: Programmablauf Pseudocode

3.4 Filtersimulation

Die Filtersimulation ist ein kleines Python-Programm, das den DLPF Filter auf die aufgezeichneten Werte, von der SD-Karte, anwendet und ein Graph zeichnet, der die rohen Werte und die gefilterten Werte darstellt (siehe Abb.**??**). Listening 3 zeigt den Code des Programmes.

```
from matplotlib import pyplot as plt
import numpy as np
import math


def digital_low_pass(cutoff_frequency, input):
    fc = 2 * math.pi * cutoff_frequency
    alpha = (fc * dt) / (1 + fc * dt)
    output = [0] * len(input)
    output[0] = input[0]

    print(alpha)

    for i in range(1, len(input)):
        output[i] = alpha * input[i] + (1 - alpha) * output[i - 1]
    return output


with open('LOG3','r') as file:
    line = []
    for i in file:
        line.append(i.rstrip('\n'))

time = []
dtime = []
dt = 0

pitch = []

for i in line:
    x = i.split('\t')
    pitch.append(float(x[0]))

for i in range(1, len(time)):
    dtime.append(time[i]-time[i-1])

for i in dtime:
    dt += i

dt = dt/len(dtime)
freq = 1.0/dt

fpitch = digital_low_pass(80, pitch)

plt.plot(time, pitch)
plt.plot(time, fpitch)

plt.show()
```

Listing 3: Filtersimulation

4 Versuche

Nach dem Bau des Quadrocopters und der Fernbedienung, wurden Tests in der Turnhalle der Schule gemacht, da der Quadrocopter schwerer als 0.5kg ist und dies gegen die Bestimmungen des Bundesamtes für Zivilluftfahrt ist, wurde desshalb auch eine Indoordrohne gebaut.

Zuerst wurden Versuche gemacht, um den richtigen Wert des P-Reglers der Nick-und Rollachse zu bestimmen. Der Wert wurde so lange erhöht, bis der Quadrocopter zu oszillieren begann. Dann wurde er in die Mitte der Turnhalle gestellt und es wurden die ersten Flugversuche gestartet. Später wurde probiert das D-Glied einzustellen. Auch nach der Achsenkorrektur, die später in den Ergebnissen besprochen wird, waren diese Versuche gescheitert.

In einem zweiten Versuch wurde der Quadrocopter mit einem komerziellen Quadrocopter des Mechanikers verglichen. Dadurch wurden einige Fehler, die im Kapitel Ergebnisse besprochen werden, korrigiert. Durch diese Korrektur kam es zu einem halberfolgreichen Versuch.

Die Gierachse wurde auch im ersten Versuch getestet. Dabei wurde bemerkt, dass diese zu stark reagierte. Der Mechaniker hat dann behauptet, dass die Gierachse auf 5Hz gefiltert werden muss. Dabei wurde ein kleines Python-Programm geschrieben, dass die aufgezeichneten Werte filterte. Natürlich konnte so nicht der Effekt des Filtes simuliert werden. Diese Versuche waren nicht befriedigend und so wurde das Augenmerk zunächst auf die Nick-und Rollachse geworfen.

Beim leztzen Versuch ist es leider zu einem Unfall gekommen und einer der Motoren wurde beschädigt, somit kann auch die Versuchsreihe bis zur Abgabe der Maturarbeit nicht weitergeführt werden, da die Motoren eine lange Lieferfrist haben und auch ein neuer Rahmen von Vorteil ist, da beim alten die Beine beschädigt waren und nicht richtig repariert werden konnten. Somit ist der Quadrocopter bis zur Abgabe noch nicht flugfähig.

5 Ergebnisse

Die erste Versuchsreihe war gescheitert. Dabei spielen viele Gründe eine Rolle. Erstens wurde entdeckt, dass der Intertialsensor bei einer Gierbewegung die Nick-und Rollachse nicht ineinander übersetzt werden, wie es in Kapitel über den Intertialsensor beschrieben wird. Auch wurden die rauschigen Sensorwerte, die mit der SD-Karte geloggt wurden, entdekt. So wurde ein DLPF einprogrammiert, der die Sensorwerte filtert. Auch wurde wie vorher beschrieben eine kleine Simulation geschrieben um den Filter zu testen. Dieser lieferte dann auch bessere Sensorwerte, die nicht rauschten. Auch nach diesen Verbesserungen ist es dem Quadrocopter nicht gelungen zu fliegen.

In der zweiten Versuchsreihe wurde dann entschieden den Quadrocopter mit einem anderen zu Vergleichen der fliegt. Dabei wurde bemerkt, dass die P-Werte viel zu klein waren und so wurden sie um den Faktor fünf erhöht. Dieser zu kleine Wert zeichnete sich damit aus, dass die Reaktion beim fliegenden Quadrocopter viel stärker war als bei der selbstgebauten.

Dennoch als der Quadrocopter mehr oder weniger flog, zeigte er ein Verhalten auf, dass dieser beim Flug oszillierte und dieser immer zu einer Seite driftete. Dieses Verhalten konnte nicht mehr weiter erforscht werden, da der Quadrocopter beschädigt wurde und nicht in der Frist bis zu Abgabe repariert werden kann.

6 Diskussion

Da keine weiteren Versuche gemacht werden können, kann nur spekuliert werden, wieso der Quadrocopter einen unstabilen Flug absolviert hat. Die Werte waren sicher nicht optimal und das D-Glied, dass solche Oszillationen verhindern solle, war auch nicht Ideal eingestellt worden. Auch ist es möglich, dass der Filter der angewant wurde, die Regelung verzögert und somit auch die Antwort des FC's auf die Oszillationen, da auch die IMU einen DLPF besitzt und so zwei Filter in Reihe geschaltet sind. Nach der Abgabe des Schriftlichen Teils werden weitere Versuche getätigt sobald der Quadrocopter repariert ist, um so die Ursache der Oszillationen herauszufinden und diese zu korrigieren.

Dennoch wurden viele der Ziele erreicht. Das designen und bauen des Flightcontrollers war erfolgreich und dieser zeigte auch das richtige Reglelverhalten. Auch konnte in der testfreien Phase, der Programmloop des FC's von 200Hz auf ca. 1.5kHz erhöht werden, was zu einer schnelleren Regelung führt.

Im Vergleich zum Flightcontroller mit dem Raspberry Pi, konnten alle Probleme gelöst werden, die dieser Verursacht hat. Es gab beim programmieren fast keine Probleme und auch wurde gelernt wie man auf der STM32-Plattform programmiert.

Zum Schluss möchte ich mich noch an dem Physiklabroant Bruno Turnherr bedanken, der sein Wissen über Quadrocopter, PCB-Design, Reflow-Löten gezeigt hat und für die Hilfe beim Bau des Quadrocopters.

7 Quellenverzeichnis

7.1 Literaturverzeichnis

[1] Alex. *Inertial Sensor Comparison MPU6000 vs MPU6050 vs MPU6500 vs ICM20602*. Webseite. 31. März 2019. URL: https://blog.dronetrest.com/inertial-sensor-comparison-mpu6000-vs-mpu6050-vs-mpu6500-vs-icm20602/.

[2] elektronik-kompendium.de. *MEMS - Micro-Electro-Mechanical Systems*. Webseite. 15. Okt. 2019. URL: https://www.elektronik-kompendium.de/sites/bau/1503041.htm.

[3] fpvracing.ch. *Multicopter Komponenten*. Webseite. 10. Okt. 2019. URL: https://fpvracing.ch/de/content/8-multicopter-komponenten.

[4] Oscar Liang. *F1, F3, F4, F7 and H7 Flight Controller Explained*. Webseite. 31. März 2019. URL: https://oscarliang.com/f1-f3-f4-flight-controller/.

[5] Oscar Liang. *What is Oneshot ESC Protocol – Active Braking*. Webseite. 15. Okt. 2019. URL: https://oscarliang.com/oneshot125-esc-quadcopter-fpv/.

[6] Sparky256 (Fiktiver Name). *What is a PA/LNA?* Webseite. 10. Okt. 2019. URL: https://electronics.stackexchange.com/questions/237267/what-is-a-pa-lna/237278.

[7] Waste(Fiktiver Name). *Regelungstechnik*. Webseite. 10. Okt. 2019. URL: https://rn-wissen.de/wiki/index.php/Regelungstechnik.

[8] sauter-elektronik.de. *What is a PA/LNA?* Webseite. 10. Okt. 2019. URL: https://www.sauter-elektronik.de/wissenswertes/reflow-loeten.

[9] Wikipedia. *Low-pass filter*. Webseite. 14. Okt. 2019. URL: https://en.wikipedia.org/wiki/Low-pass_filter#Simple_infinite_impulse_response_filter.

[10] Wikipedia. *Quadrocopter*. Webseite. 10. Okt. 2019. URL: https://de.wikipedia.org/wiki/Quadrocopter#Entwicklung.

[11] Wikipedia. *Quadrocopter*. Webseite. 25. Nov. 2019. URL: https://de.wikipedia.org/wiki/Quadrocopter#Einsatz.

7.2 Abbildungsverzeichnis

8 Anhang 1 QR-Code zum Quellcode

https://github.com/Pluscrafter/SHARKSKY/tree/Abgabe

9 Anhang 2 Beweis und Auflösung des DLPF

Auflösung 1:

$$
\begin{aligned}
x_i - y_i &= RC \cdot \frac{y_i - y_{i-1}}{\Delta t} \\
x_i - y_i &= \frac{RCy_i - RCy_{i-1}}{\Delta t} \\
x_i\Delta t - y_i\Delta &= RCy_i - RCy_{i-1} \\
y_i\Delta t + RCy_i &= x_i\Delta t + RCy_{i-1} \\
y_i(\Delta t + RC) &= x_i\Delta t + RCy_{i-1} \\
yi &= x_i\frac{\Delta t}{\Delta t + RC} + y_{i-1}\frac{RC}{\Delta t + RC}
\end{aligned}
$$

Beweis 1:

$$
\begin{aligned}
\alpha &= \frac{\Delta t}{\Delta t + RC} \\
\alpha\Delta t + RC\alpha &= \Delta t \\
RC\alpha &= \Delta t\alpha\Delta t \\
RC &= \Delta t\frac{1-\alpha}{\alpha} \\
\frac{1}{2\pi f_c} &= \frac{\Delta t - \alpha\Delta t}{\alpha} \\
\frac{1}{2\pi f_c}\alpha &= \Delta t - \alpha\Delta t \\
\frac{1}{2\pi f_c}\alpha + \alpha\Delta t &= \Delta t \\
\alpha &= \frac{\Delta t}{\frac{1}{2\pi f_c} + \Delta t} \\
\alpha &= \frac{\Delta t}{\frac{1+2\pi f_c\Delta t}{2\pi f_c}} \\
\alpha &= \frac{2\pi f_c\Delta t}{1 + 2\pi f_c\Delta t}
\end{aligned}
$$