

Maturitätsarbeit an der Kantonsschule Zürich Nord

Regelungstechnik

PID-Parametrisierung anhand eines selbstgebauten Quadrocopters

Charpoan Kong

M6d

Kantonsschule Zürich Nord

23. November 2019

Inhaltsverzeichnis

1	Einleitung	1
2	Was ist ein Quadrocopter?	2
3	PID-Regler und Filter	3
3.1	Was ist ein Regler?	3
3.1.1	P-Regler	4
3.1.2	I-Regler	4
3.1.3	D-Glied	4
3.1.4	PID-Regler	4
3.2	Digitaler Low-Pass Filter	5
4	Äusserer Aufbau	7
4.1	Motoren	7
4.2	Akku	7
5	Der Flightcontroller	8
5.1	Komponenten	8
5.1.1	Microcontroller	8
5.1.2	Sensoren	8
5.1.3	Empfänger	10
5.2	Leiterplatte	10
5.2.1	Design	10
5.2.2	Bau	11
5.3	Programmierung	12
5.3.1	Motoren und Timing	13
5.3.2	PID Implementierung	13
6	Fernbedienung	14
7	Versuche	15
8	Schlussfolgerung	16
9	Quellenverzeichnis	17
9.1	Literaturverzeichnis	17
9.2	Abbildungsverzeichnis	17
10	Anhang 1	18

1 Einleitung

Ein Quadrocopter ist ein Flugobjekt mit vier Propellern. In der Vergangenheit wurden schon Versuche mit Flugobjekten mit vier Propellern gemacht z.B. wie der Luftfahrtpionier Étienne Oehmichen, der 1920 den Oehmichen No. 2 gebaut hatte. Damals waren die Propeller elastisch und man konnte mit Seilzügen den Anstellwinkel der Propeller einstellen. Er konnte damit einige Rekorde aufstellen.[8]

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.(Quelle)

Die Intention zu dieser Arbeit folgt aus einem Projekt aus dem Freifachkurs Robotik. Damals kam der Raspberry Pi zum. Da der ein Betriebssystem parallel zum Flightcontroller am laufen hat, war die Regelung des Flightcontrollers zu träge. Dessenwegen wird auch ein neuer Chip evaluiert, der den Echtzeitanforderungen entspricht.

Die Fragestellung lautet, wie der 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 versucht 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 Bestimmen wie der PID-Regler stabiler gemacht wird und wie ungleichheiten, die durch Vibrationen entstehen, ausgeglichen werden können. Die Fernbedienung wird auch selbst gebaut und dient dazu Daten während des Fluges auszuwerten.

2 Was ist ein Quadrocopter?

Ein Quadrocopter ist wie in der Einleitung beschrieben ein Flugobjekt. 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.

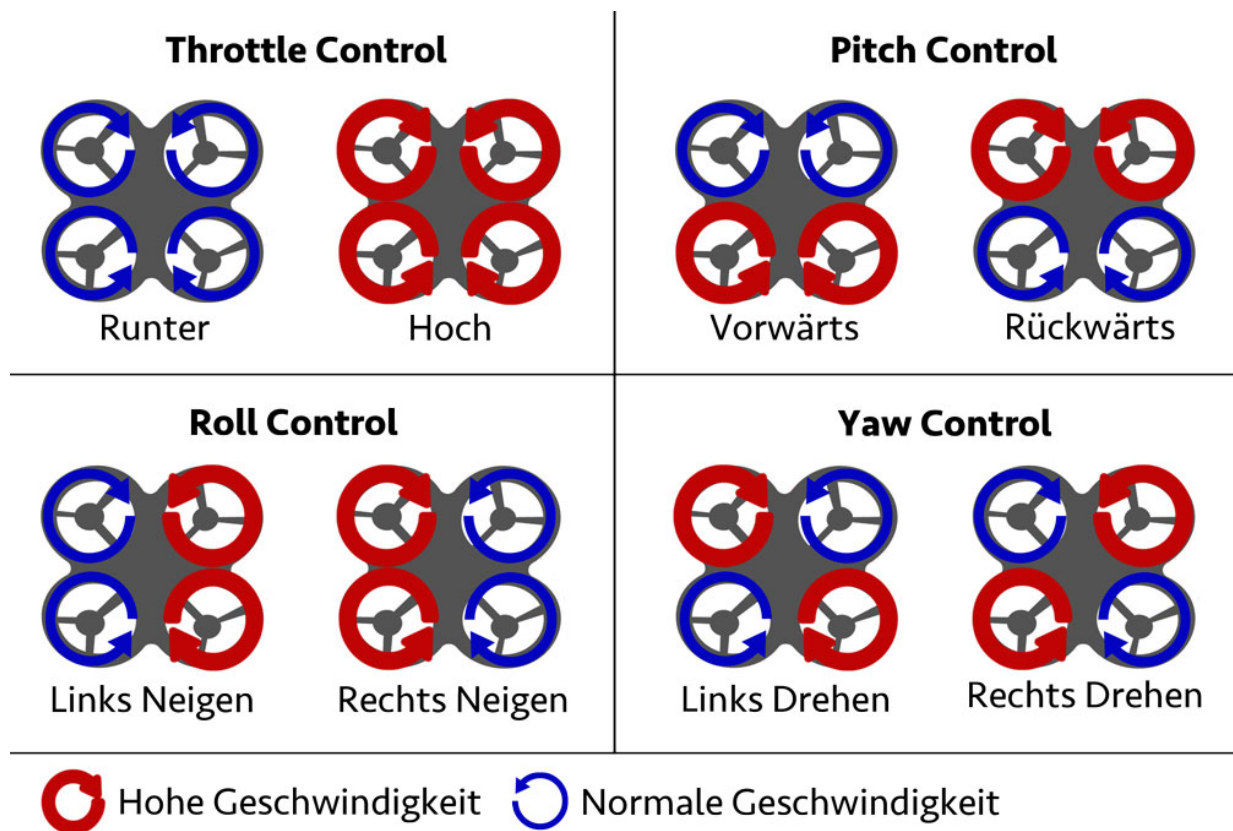


Abbildung 1: Drehrichtung und Schub bei verschiedenen Steuereingaben

Abbildung 1 zeigt die Rotationsrichtung der Motoren gezeigt. Die benachbarten Motoren müssen eine gegensätzliche Rotation haben, da sonst der Quadrocopter ins Rotieren kommt. 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.

3 PID-Regler und Filter

3.1 Was ist ein Regler?

Eine Regelung ist eine Steuerung mit Rückkoppelung. Es wird 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.[5]

Im Regelkreis wird der Ist-Wert z.B. mit einem Sensor gemessen. Dieser wird mit dem Soll-Wert verglichen um die Regelabweichung zu bestimmen. So kann der Reglergröße bestimmt werden um so das System dem Soll-Wert anzugleichen.

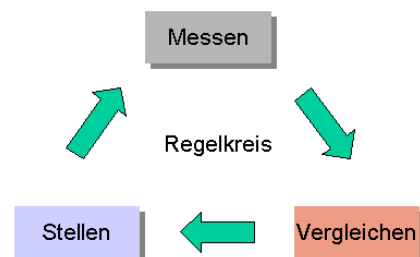
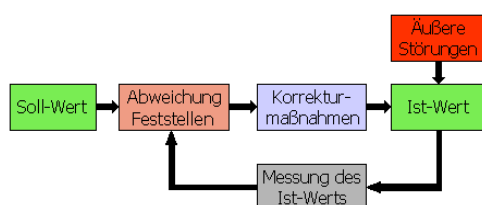


Abbildung 2: Der Regelkreis

Die Regelabweichung kann dabei einfach mit der Differenz des Ist-Wertes x mit dem Soll-Wertes w bestimmt werden.[5]

$$e(t) = w - x \quad (1)$$



In Abbildung 3 sieht man, wie vorher beschrieben, wie ein solcher Regelkreis funktioniert. 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.(Quelle)[5]

Abbildung 3: Wirkungsweise des Regelkreises

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

3.1.1 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 mittelschnell.[5]

$$y(t) = Kp \cdot e(t) \quad (2)$$

3.1.2 I-Regler

Beim Integralregler wird die Regelabweichung über die Zeit summiert und mit dem Faktor Ki verstärkt.(Vgl. 3) Dabei werden Abweichungen vollständig eliminiert, da dieser Regelwert immer anwächst solange die Regelabweichung nicht Null ist. Dieser Regler wirkt langsam.[5]

$$y(t) = Ki \int_0^t e(t)dt \quad (3)$$

3.1.3 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 regeln kann sondern nur auf Veränderungen in der Regelabweichung reagiert.[5]

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

3.1.4 PID-Regler

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

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

3.2 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.[7]

Im letzten Kapitel wurde der PID-Regler eingeführt, doch Vibrationen verursachen Störungen in der Regelung und führen so zu falschen korrektoren. Zum Beispiel verursachen Vibrationen beim D-Glied starke Ausschläge, 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.

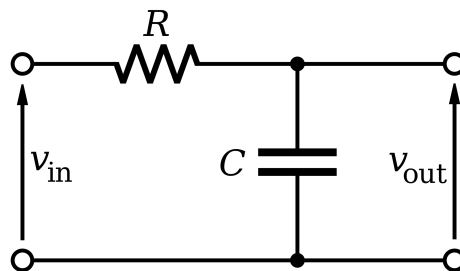


Abbildung 4: RC Low-Pass Filter

Wie man in 6 sieht, ist die Eingangs- und Ausgangsspannung vom Widerstand und von der Stromstärke abhängig.

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

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

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

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

So kann man die Differenzialgleichung 6 wie 9 darstellen.

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

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

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

Die Gleichung 11 (Ganze Auflösung siehe Anhang 1 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.

$$y_i = x_i \frac{\Delta t}{\Delta t + RC} + y_{i-1} \frac{RC}{\Delta t + RC} \quad (11)$$

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

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 1 Beweis 1).

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

4 Äusserer Aufbau

4.1 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

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

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.(Bild 1)

4.2 Akku

Der Akku 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 minimale Spannung beträgt 11.1V. Die Entladerate wird C -Rate angegeben. Mit der Kapazität multipliziert wird der maximale Entladestrom ausgerechnet.[2]

$$I_{\max} = Q \cdot C \quad (14)$$

(Bild r)

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 auch im Extremfall, Feuer fangen oder explodieren können. So darf man die einzelnen Zellen nicht unter einer Spannung von 2.7V fallen lassen. Deshalb 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 gleichmässig geladen werden. Eine hohe Kapazität garantiert nicht immer eine längere Flugzeit, da mehr Kapazität auch mehr Gewicht bedeutet.[2]

5 Der Flightcontroller

Der Flightcontroller ist das Herz des Quadrocopter er 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 Programmierprozess beschrieben.

5.1 Komponenten

5.1.1 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 ohne Zusatz Pehpetrie benutzt werden, da sie schon einen Flash, RAM etc. besitzen. Aber dies kommt mit einer tieferen Taktrate daher. Dies stört aber den Quadrocopter ist da diese völlig ausreichend ist. Der Hauptcontroller läuft mit einer Taktfrequenz von 216MHz.

5.1.2 Sensoren

Die wichtigsten Sensoren des Quadrocopter sind der Gyrosensor und der Beschleunigungssensor. 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.[1]. Manätzt die Sensorstrukturen und deren auslese Schaltung auf einen Chip.

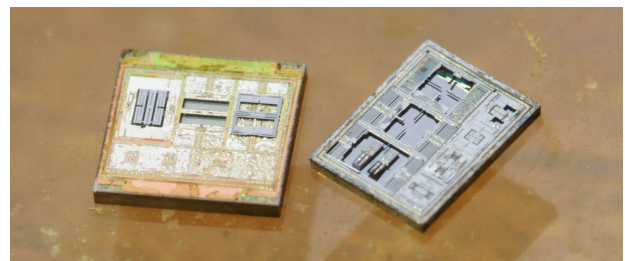


Abbildung 5: Dice des MPU6050

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.

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

Wenn man den Absoluten Winkel alleine mit dem Gyrosensor misst hat dieser über die Zeit einen Drift. Zum einen wird das durch das Rauschen im Sensor verurasacht und zu anderen auch von einem nicht genauen Loop-Zeit. Auch kann nicht der Anfangswinkel bestimmt werden und man müsste den Quadrocopter immer vom Boden aus starten. Deswegen braucht man da den Beschleunigungssensor. Dieser misst die Beschleunigung auf allen Achsen relativ zur Erdbeschleunigung. 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.

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

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 Komplementärfilters. (Man muss auch beachten das man nur den Absoluten Winkel nur für die Nick-und Rollachse berechnen kann da, für die Gierachse noch ein Magnetometer nötig wäre für eine Null-Referenz.)

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

Dem Winkel aus dem Beschleunigungssensor wird ein tiefer Faktor gegeben da dieser Winkel nicht mehr genau stimmen wird, wenn der Quadrocopter 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 Quadrocopter bestimmen, ihn auch aus der Hand starten lassen und bekommt auch noch stabilere Werte.

Desweiteren muss auch noch beachtet werden, das bei einer Bewegung um die Gierachse, der Sensor keine Bewegung auf der Roll- und Nickachse misst. Deshalb müssen die Werte dieser Achsen bei einer Bewegung um die Gierachse ineinander übersetzt werden mithilfe der Rotationsmatrix.

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

Somit muss man die Achsen wie folgt verrechnen.

$$roll' = \sin(\alpha) * pitch \quad (19)$$

$$pitch' = \sin(\alpha) * roll \quad (20)$$

5.1.3 Empfänger

Als Empfänger wird ein NRF24L01 Breakoutboard mit PA und LNA benutzt. Der ist ein 2.4GHz Sender und Empfänger von der Firma Nordic Semiconductor. Die Kommunikation erfolgt digital und der Chip besitzt auch eine Zyklische Redundanzprüfung(CRC). Damit wird kontrolliert, ob ein Datenpaket 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.[4] Vom Hersteller wird eine Reichweite von bis zu einem Kilometer versprochen aber in einer Wohnsiedlung kommt man mit Sichtkontakt bis zu 300m, was ausreichend genug ist.

5.2 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.

5.2.1 Design

Zuerst habe ich mir überlegt was für einen Microcontroller ich wählen sollte. Bei meinen ersten Versuchen mit dem Raspberry gingen schief, da der Regler nicht schnell genug reagierte und eine nicht konstantes Regelverhalten gezeigt hat. Einerseits lief ein Betriebssystem parallel zum Flightcontroller, andererseits war die Regelung nicht Echtzeit genug. Dessenwegen habe ich im Internet nachgeschaut was für Microcontroller auf den meisten Quadroptern verbaut sind. Dort werden meist MCU aus der STM32-Familie verwendet. So kam ich auf die Webseite von Oscar Liang¹ bei der die einzelnen MCU-Generationen aufgelistet waren mit den Vor- und Nachteilen. Ich entschied mich für die neueste Generation aus dieser MCU-Familie, einen STM32F7xx MCU zu benutzen. Nachher habe ich vorhandene Boards angeschaut und gesehen, dass die meisten den STM32F722RET6 benutzen aus dieser MCU-Generation. Als nächstes überlegte ich mir was für einen Gyro- und Beschleunigungssensor ich benutzen wollte. Ich habe schon eine MPU-6050 IMU aber ich wollte noch einen mit SPI-Protokoll, da dies schneller ist, und der andere als Backup, so recherchierte ich was für IMU's die FC's benutzen² und entschied mich für den ICM-20689. Als Empfänger habe ich einen NRF24L01 PA + LNA gewählt. Als Backup wird noch ein PPM Eingang eingebaut für die standard Funkübertragung über eine normale Fernbedienung. Da für die Versuche Daten gebraucht werden, wird eine microSD-Karten Schnittstelle eingebaut, um die Daten aufzuzeichnen. Damit der Flightcontroller mit dem PC kommunizieren kann um zu Debuggen muss ein FTDI-Chip(F232RL) eingebaut werden damit wird das UART-Protokoll, in ein für den PC verständliches Protokoll übersetzt. Um die Höhe zu bestimmen habe ich

¹<https://oscarliang.com/f1-f3-f4-flight-controller/> aufgerufen am 31.03.2019

²<https://blog.dronetrest.com/inertial-sensor-comparison-mpu6000-vs-mpu6050-vs-mpu6500-vs-icm20602/> aufgerufen am 31.03.2019

mich für den BMP280 Drucksensor ausgewählt um die barometrische Höhe zu bestimmen. Da dieser für das Auslesen eine lange Zeit braucht habe ich überlegt noch eine NebenMCU einzubauen. Deshalb habe ich den ATmega328 gewählt da er auch auf den Arduino UNO drauf ist und ich mich damit schon auskenne. Über diese MCU werden auch die Ultraschallsensoren ausgelesen, die eine genauere Höhenmessung ermöglicht, aber nur bis zu sechs Metern. Da die ESC's einen 5V Versorgungsausgang haben muss noch ein Spannungsregler eingebaut werden, da die meisten IC's 3.3V brauchen, der Schulmechaniker Herr Thurnherr hat mir den LM3940 empfohlen.

Die Leiterplatte wird mit dem Programm Eagle von Autodesk designt. In dem Programm kann man die einzelnen Bauteile zusammensetzen und verbinden (Siehe Abbildung 5). Ich musste nach dem Datenblatt der einzelnen IC's, Kondensatoren und Widerstände einbauen. Die Schaltbilder für die einzelnen IC's, die nicht schon in der Standardbibliothek vorhanden sind, habe ich aus dem Internet heruntergeladen. Herr Thurnherr hat mich dabei unterstützt und alle Fragen beantwortet. Nach mehreren Kontrollen wurde die Leiterplatte bei JLCPCB und die einzelnen Teile bei LCSC, Arrow, RS und Conrad bestellt. Dazu wurde noch eine Schablone für das SMD-löten bestellt.

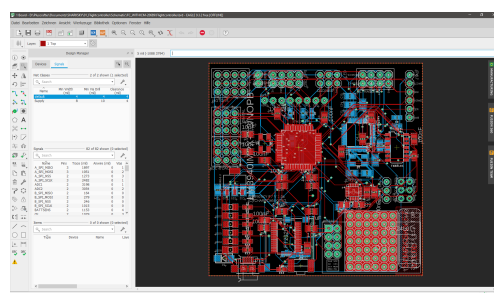


Abbildung 6: Autodesk Eagle

5.2.2 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 platziert. Dies geschieht im industriellen Rahmen mit sogenannten "Pick and Place" Maschinen. Später wird die Leiterplatte in den Ofen getan und nach einer bestimmten Temperaturkurve erhitzt, bis die Komponenten verlötet sind [6].

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 ATmega, dessen Quarzoszillator und dem SD-Kartenslot aufgetragen und die Komponenten wurden platziert. Dann wurden diese mit dem Heissluftföhn verlötet. Um das Herausfallen der Teile 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

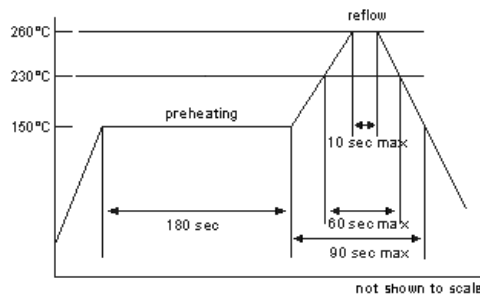


Abbildung 7: Temperaturkurve für Reflowlötten

festgemacht. So konnte man mit Lötpaste und einem Spachtel die Löcher und somit auch die Lötstellen mit Lötpaste füllen. Dann wurde die Schablone entfernt. Mithilfe des Mikroskops wurden die Teile dann auf der Leiterplatte platziert. Im Ofen wurden sie dann ungefähr nach der Temperaturkurve gebacken und so war die Vorderseite gelötet. Die Hinterseite wurden dann per Hand fertig gelötet worden. Dann wurden nur noch die Konnektoren angelötet und der Flightcontroller ist fertig zum programmieren.

5.3 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. umgeschrieben werden mussten für den STM32. Die Bibliothek für die IMU wurde aber selbst programmiert. Dabei kamen mehrere Probleme auf, z.b. das dass ChipSelect-Signal zu früh auf High ging.(Bild). In diesem Kapitel geht es um die grössten Hürden und wichtigsten Abläufe bei der programmierung des Quadrocopters. Im folgenden Listing wird der grobe Ablaufplan des Programms gezeigt.

```

1 int main(){
2     initPID_Values();
3     initPeriphial();
4     initIMU();
5     initnRF24();
6     initSD();
7
8     while(true){
9         loop();
10    }
11
12    return 0;
13 }
14
15 void loop(){

```

```

16  start = elapsedticks();
17
18  calcTrueangle();
19  writeSD() //every 200th loopcycle;
20  calcError();
21  calcPID();
22  setMotor();
23
24  stop = elapsedticks();
25  looptime = stop - start;
26 }
27
28 void setMotor(){
29     if(throttle < 100){
30         motspeed[0:3] = 1024;
31     }else{
32         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
38     PWM_SET(M1, motspeed[0]);
39     PWM_SET(M2, motspeed[1]);
40     PWM_SET(M3, motspeed[2]);
41     PWM_SET(M4, motspeed[3]);
42 }

```

Listing 1: Programmablauf Pseudocode

Die IMU und der Empfänger werden per Interrupt ausgelesen. Die IMU gibt im 1KHz Takt ein Interruptsignal, dass beim lesen dann zurückgesetzt wird. Beim Empfänger wird ein Interruptsignal ausgegeben sobald neue Daten vorhanden sind.

5.3.1 Motoren und Timing

Die ESC's der Motoren müssen mit dem OneShot125 Protokoll[3] 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%. Die Motoren erreichen ihre maximale Drehzal bei einer Periode von 500us bzw. bei einem PWM von 50%. (Bild).

5.3.2 PID Implementierung

Der PID-Regler wird für den absoluten Winkel und für die Winkelgeschwindigkeit implementiert, wobei man die Gierachse nur mit der Winkelgeschwindigkeit implementieren kann, da keine Referenz besteht, z.b durch ein Magnetometer. Die Regelungsdifferenz und Regelungskorrektur wird jeden Loop-Zyklus berechnet mit ca 2kHz.

6 Fernbedienung

Die Fernbedienung wird aus Holz und PLA gebaut. Der Rahmen wurde aus dem 3D-Drucker gedruckt. (Bild) 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. Im folgenden sehen sie den Ablaufplan des Programms.

```
1  int main(){
2      initPeriphial();
3      initADC();
4      initnRF24();
5
6      while(true){
7          loop();
8      }
9
10     return 0;
11 }
12
13 void loop(){
14     readADC();
15     sendData();
16
17     menu_print();
18 }
19
20 void menu_print(){
21     MENU_1:
22     print_ADC_VAL();
23     MENU_2:
24     print_ACK_VAL();
25     MENU_3:
26     print_PID_ROLL_PITCH(); //<-- ALSO EDIT VALUES ON 0% Throttle
27     MENU_4:
28     print_PID_YAW(); //<-- ALSO EDIT VALUES ON 0% Throttle
29 }
```

Listing 2: Programmablauf Pseudocode

7 Versuche

Nach dem Bau des Quadrocopter und der Fernbedienung wurden tests in der Turnhalle der Schule gemacht, da der Quadrocopter schwerer ist als 0.5kg und dies gegen die Bestimmungen des Bundesamtes für Zivilluftfahrt ware(Quelle).

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 wurde der erste Flug probiert, der auch gescheitert ist. Es wurde dann auch probiert das D-Glied einzustellen, aber diese sind auch gescheitert. Später bemerkte ich den Fehler mit dem übersetzen der Winkel, der im Kapitel über den Inertialsensor, beschrieben wird. Als dieser Verbessert wurde, war das Verhalten des Quadrocopters immer noch nicht befriedigend und dieser regelte immer noch zu schwach.

Desshalb wurde das Verhalten des Quadrocopters mit einer funktionierenden Quadrocopters des Mechaniker verglichen. Dadurch wurde sichtbar das die Oszillation des P-Reglers gar nicht zu stark war sondern sogar viel zu schwach. Also wurden Versuche gemacht bei dem der P-Regler um das fünffache vergrößert worden ist und ein D-Glied gesetzt, das ein zehntel des P-Reglers entsprach. Diese Versuch war dann erfolgreich und der Quadrocopter machte einen instabilen oszillierenden Flug. Dennoch sah man das der Flightcontroller richtig regelte.

Zurück zu den ersten Versuchen. Bei der Gierachse wurde auch zuerst ein P-Regler Wert eingesetzt. Aber diese korrektoren führten immer zu einem Purzelbaum des Quadrocopters. Aber der Mechaniker hat dann gesagt, dass die Gierachse mit ca. 5 Hz läuft. So wurde auch der obengenannte Digitale Lowpass-Filter einprogrammiert, der die Werte der Gierachse auf 5 Hz filtert. Auch wurde ein kleines Python Programm geschrieben, der die aufgezeichneten Sensor Werte filtert und so die Effekte des Filters sichtbar wurden. Man muss dabei beachten, dass das Verhalten des Flightcontrollers nicht damit dargestellt werden können, da er die Werte noch ungefiltert verarbeitet hat. Dennoch waren die darauffolgenden Versuche nicht erfolgreich. So wurde das Augenmerk zuerst 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.

8 Schlussfolgerung

9 Quellenverzeichnis

9.1 Literaturverzeichnis

Literatur

- [1] elektronik-kompodium.de. *MEMS - Micro-Electro-Mechanical Systems*. Webseite. Oktober 2019. URL: <https://www.elektronik-kompodium.de/sites/bau/1503041.htm>.
- [2] fpvracing.ch. *Multicopter Komponenten*. Webseite. Oktober 2019. URL: <https://fpvracing.ch/de/content/8-multicopter-komponenten>.
- [3] Oscar Liang. *What is Oneshot ESC Protocol – Active Braking*. Webseite. Oktober 2019. URL: <https://oscarliang.com/oneshot125-esc-quadcopter-fpv/>.
- [4] Sparky256 (Fiktiver Name). *What is a PA/LNA?* Webseite. Oktober 2019. URL: <https://electronics.stackexchange.com/questions/237267/what-is-a-pa-lna/237278>.
- [5] Waste(Fiktiver Name). *Regelungstechnik*. Webseite. Oktober 2019. URL: <https://rn-wissen.de/wiki/index.php/Regelungstechnik>.
- [6] sauter-elektronik.de. *What is a PA/LNA?* Webseite. Oktober 2019. URL: <https://www.sauter-elektronik.de/wissenswertes/reflow-loeten>.
- [7] Wikipedia. *Low-pass filter*. Webseite. Oktober 2019. URL: https://en.wikipedia.org/wiki/Low-pass_filter#Simple_infinite_impulse_response_filter.
- [8] Wikipedia. *Quadrocopter*. Webseite. Oktober 2019. URL: <https://de.wikipedia.org/wiki/Quadrocopter#Entwicklung>.

9.2 Abbildungsverzeichnis

Abbildungsverzeichnis

1	https://fpvracing.ch/de/content/7-grundsatzliche-funktion-quadrocopter-multicopter	2
2	https://rn-wissen.de/wiki/images/2/25/Regelkreis1.png	3
3	https://rn-wissen.de/wiki/images/5/5d/Regelkreis2.png	3
4	aasdf	5
5	https://de.wikipedia.org/wiki/TDK	8
6	Screenshot Eagle FC brd file	11
7	http://www.comtec-crystals.com/service-5.php	12

10 Anhang 1

Auflösung 1:

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

Beweis 1:

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