Maturitätsarbeit an der Kantonsschule Zürich Nord

Regelungstechnik PID-Parameter anhand einem Quadrocopter erforschen

Charpoan Kong
M6d
Kantonsschule Zürich Nord

8. November 2019

Inhaltsverzeichnis

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.

Mit dieser Arbeit will ich mit dem Bau und dessen Regelung befasssen. Der Quadrocopter wird selbst gebau mit dem Hauptziel den Flightcontroller selbst zu designen, bauen und programmieren. So sollte die Regelung auch selbst implementiert werden. Dannach sollten Versuche gemacht werden, die bestimmen sollten, wie der PID-Regler stabiler gemacht werden kann und wie ungleichheiten, wie sie durch Vibration entstehen, auszugleichen.

2 Was ist ein Quadrocopter?

Ein Qadrocopter ist wie in der Einleitung beschrieben ein Flugobjekt. Man könnte meinen, dass man einen stabilen Flug erreichen kann, wenn man den vier Propellern den gleichen Schub gibt. 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 steuert das der Quadrocopter im Gleichgewicht ist oder dieser einen anderen bestimmten Winkel hält.

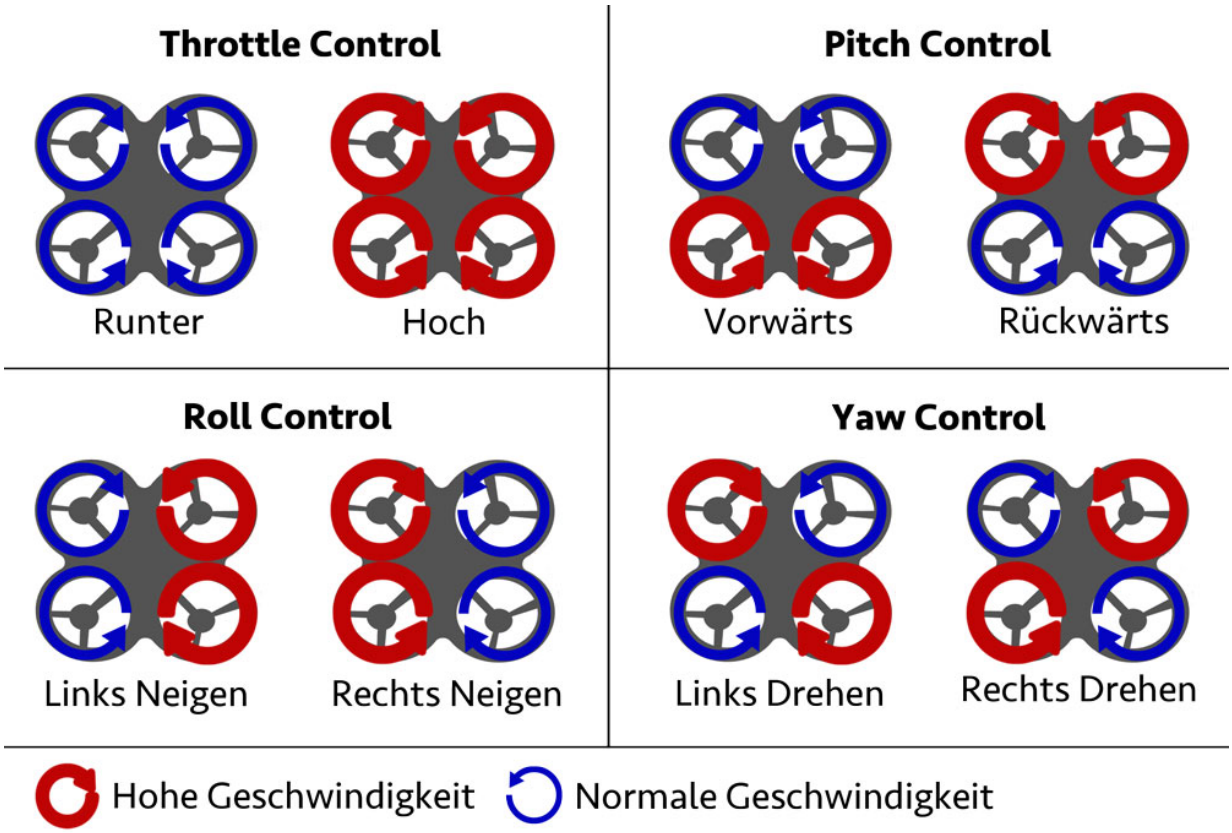

Abbildung 1: Drehrichtung und Schub bei verschiedenen Steuereingaben

Man sieht in dieser Abbildung welche Rotationsrichtung die Motoren haben. Die benachbarten Motoren müssen eine gegensätzliche Rotation haben, da sonst der Quadrocopter zum 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. Der FC hält dieses Gleichgewicht mit einem PID-Regler.

3 PID-Regler und Filter

3.1 Was ist ein Regler?

Eine Regelung ist ein 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]

Der Regelkreis misst die Regelgrösse z.B. mit einem Sensor. Dann wird dieser mit dem Soll-Wert verglichen um die Regelabweichung zu bestimmen. So kann ein z.B. Motor gestellt werden damit dieser sich dem Soll-Wert nähert.

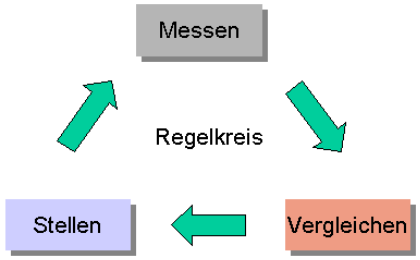

Abbildung 2: Der Regelkreis

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

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

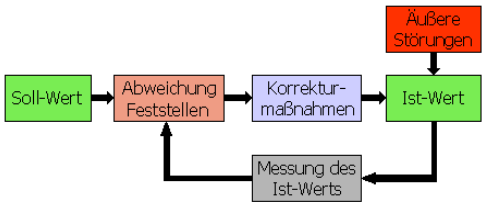

Abbildung 3: Wirkungsweise des Regelkreises

In der Abbildung 3 sieht man, wie vorher beschrieben, wie ein solcher Regelkreis funktionert. In der Regelungstechnik wird versucht einen solchen Regelkreis mathematisch zu modellieren. In dem Kapitel wird nur der PID-Regler und seine Komponenten besprochen, da dieser häufig bei Quadrocoptern benutzt wird.[5]

Der PID-Regler besteht aus mehreren Relgern und dem D-Glied. Untern 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. Das Problem dabei ist, dass diese Abweichung bleibend ist und somit den Soll-Wert über- bzw. unterschiesst. Dieser Regler wirkt mittelschnell.[5]

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

```
void P_Regler(float Kp, float error, float &P){
  P = kp * error;
}
```

Listing 1: P-Regler C++ Pseudocode

3.1.2 I-Regler

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

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

```
void I_Regler(float Ki, float error, float &I ){
  I += I+error * Ki;
}
```

Listing 2: I-Regler C++ Pseudocode

3.1.3 D-Glied

Das Differenzialglied schaut auf die Differenz der Regelabweichung zur vorherigen Regelabweichung und wird mit dem Faktor Kd verstärkt. Desshalb reagiert dieser sehr schnell und gibt den den beiden anderen Vorhaltezeit. Differenzialglied ist kein Regler da es alleine nichts regleln kann sondern nur auf Veränderungen in der Regelabweichung reagiert.[5]

$$y(t) = Kd \cdot e(t)\frac{d}{dt} \tag{4}$$

```
void D_Glied(float Kd, float error, float &previous_error, float dt,
    float &D){
  D = Kd * (error - previous_error);
  previous_error = error;
}
```

Listing 3: D-Regler C++ Pseudocode

3.1.4 PID-Regler

Beim PID-Regler werden die Eigenschaften der einzelnen Regler und dem D-Glied vereint.

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

```
void PID_Regler(float P, float I, float D, float &PID){
  PID = P + I + D;
}
```

Listing 4: D-Regler C++ Pseudocode

3.2 Digitaler Low-Pass Filter

Ein Low-Pass Filter filtert hohe Frequenzen aus. Man kennt diesen Filter meist aus der Elektronik oder der Tontechnik. Beim Quadrocopter wird ein solcher Filter für das unterdrücken des Rauschens der IMU verwendet. Dieses Rauschen wird durch die Vibrationen der Motoren erzeugt und liegt ca. bei 133Hz.

Bei diesem Digitalen Low-Pass Filter(DLPF) wird ein analoger RC-Low-Pass Filter emuliert.[7]

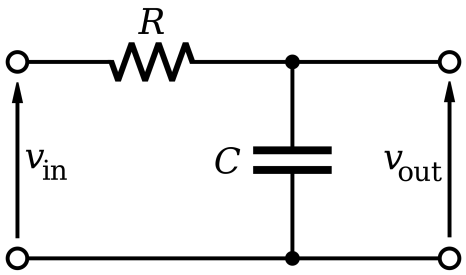

Abbildung 4: RC Low-Pass Filter

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

Wie man in 6 sieht, ist die Eingang-und Ausgangsspannung vom Wiederstand und vom Strom abhängig.

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

$$I(t) = \frac{dQ_c}{dt} = C \cdot \frac{dv_{out}}{dt} \tag{8}$$

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

$$v_{in}(t) - v_{out}(t) = RC \cdot \frac{dv_{out}}{dt} \tag{9}$$

So kann man die Differenzialgleichung 6 wie 8 darstellen.

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

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

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

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.

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

RC kann man 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).

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

Vibrationen verursachen 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 so starke Ausschläge verhindert.

4 Äusserer Aufbau

4.1 Motoren

Die verwendeten Motoren sind sogenannte bürstenlose Motoren. Diese werden mit einem Electronic Speed Controller(ESC) angesteuert, diese wandeln 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 meinem Quadrocopter handelt es sich um ReadyToSky RS2205 Motoren mit einem Kv von 2300. Ich habe diese Motoren ausgesucht, da sie für genug Leistung bringen für einen Quadrocopter mit einer Rahmenlänge von 250mm.(Bild l)

4.2 Akku

Der Akku versorgt alle Kompionenten, 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. Es gibt noch das Entladerate, die als C-Rate angegeben wird. Mit mit der Kapazität multipliziert gibt es den maximalen Entladestrom an.[2]

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

(Bild r)

Man benutzt in Quadrocoptern meist LiPo Akkus, da sie eine hohe Energiedichte besitzen und viel Leistung abgeben können. Das 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 bringen. Desshalb wird der Akku auch an einem Akkuüberwacher angeschlossen, der piepst, wenn der Akku verbraucht ist. Auch braucht man ein Balancerladegerät, das kontrolliert, dass alle Zellen gleichmässug 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 Propgrammierprozess 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 einzelen 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 lauft 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, diese werden Direkt auf den Siliziumwaver der Chips realisiert. (Bild) Man ätzt die Sensorstrukturen und deren auslese Schaltung auf einen Chip. MEMS bedeutet microelectromechanical systems. Das bedeutet, dass diese Sensoren auf einer mechanischen Basis funktionieren.[1]

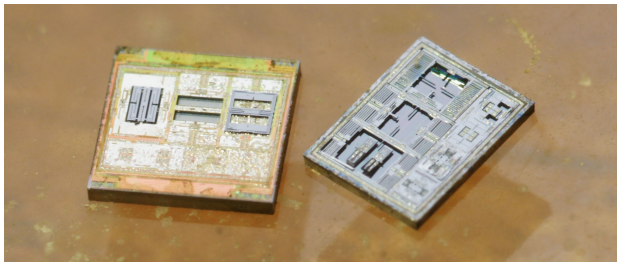

Abbildung 5: Dice des MPU6050

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.

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

Wenn man den Absoluten Winkel alleine mit dem Gyrosensor misst hat dieser über die Zeit einen Drift. 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 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.

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

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. (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.)

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

Dem Winkel aus dem Beschleunigungssensor wird 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. (Barometer ?)

5.1.3 Empfänger

Als Empfänger wir dein 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 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 beseutet, dass schwache Empfangssignale verstärkt werden.[4] Vom Hersteller wird eine Reichweite vom bis zu einem Kilometer versprochen aber in einer Wohnsiedlung kommt man mit Sichtkontakt bis zu 300m, was ausreichend genug ist.

5.1.4 Schnittstellen

Damit der Mikrocontroller mit den Sensoren und der restlichen Pehpetrie kommunizeren kann braucht es Schnittstellen und Protokolle die diese Komponenten verbindedn. Zuerst wird

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 konstates Regelverhalten gezeigt hat. Einerseits lief ein Betriebsystem pararell zum Flightcontroller, adererseits war die Regelung nicht Echtzeit genug. Desswegen habe ich im Internet nachgeschaut was für Microcontroller auf den meisten Quadroopter verbaut sind. Dort werden meist MCU aus der STM32-Familie verwendet. So kam ich auf die Webseite von Oscar Liang[1] bei der die einzelnen MCU-Generationen aufgelistet waren mit den Vor-und Nachteilen. Ich entschied mich für die neuste Generation aus dieser MCU-Familie, einen STM32F7xx MCU zu benutzen. Nachher habe ich vorhandene Boards angschaut und gesehen, das die meisten den STM32F722RET6 benutzen aus dieser MCU-Generation. Als nächstes überlegte ich mir was für einen Gyro-und Beschleunigungssensor ich bentzen wolle. 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 benutzen[2] 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 standart 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 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 Span-

[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

nungsregler eigebaut werden, da die meisten IC's 3.3V braucchen, der Schulmechaniker Herr Thurnherr hat mir den LM3940 empfohlen.

Die Leiterplatte wird mit den Program Eagle von Autodesk designt. In dem Programm kann man die einzelnen Bauteile zusammensetzen und verbinden(Siehe Abbildung 5).Ich musste nach dem Datenblatt der einzelenen IC's, Konsendatoren und Wiederstände einbauen. Die Schaltbilder für die einzelnen IC's, die nicht schon in der Standartbibliothek 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 einzelenen Teile bei LCSC, Arrow, RS und Conrad bestellt. Dazu wurde noch eine Schablone für das SMD-löten bestellt.

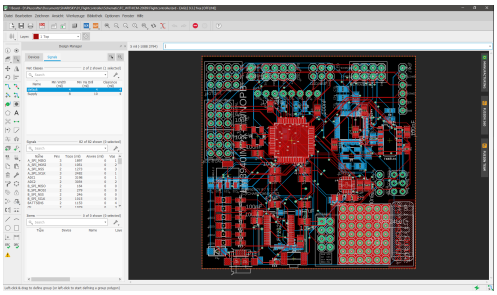

Abbildung 6: Autodesk Eagle

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

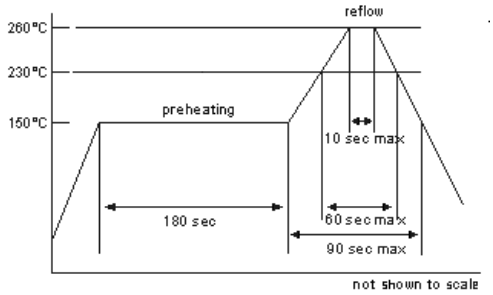

Abbildung 7: Temperaturkurve für Reflowlöten

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 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 plaziert. 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. In der unteren Abbildung sieht man den Ablauf des Programmes.(Bild)

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

  while(true){
    loop();
  }

  return 0;
}

void loop(){
  start = elapsedticks();

  calcTrueangle();
  writeSD() //every 200th loopcycle;
  calcError();
  calcPID();
  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 5: Programmablauf Pseudocode

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 Winkelgeschwindikeit implementieren kann, da keine Referenz besteht, z.b durch ein Magnetometer. Die Regelungsdifferenz und Regelungskorrektur wird jeden Loop-Zyklus berechnet, also ca. mit 500Hz.

6 Fernbedienung

7 Versuche

7.1 PID absoluter Winkel oder Winkelgeschwindigkeit

7.1.1 Absoluter Winkel

7.1.2 Winkelgeschwindigkeit

7.1.3 Ergebnisse

7.2 Filter

7.2.1 Ohne Filter

7.2.2 Filter 1

7.2.3 Filter 2

7.2.4 Ergebnisse

8 Schlussfolgerung

9 Glossar

10 Quellenverzeichnis

10.1 Literaturverzeichnis

Literatur

[1] elektronik-kompendium.de. *MEMS - Micro-Electro-Mechanical Systems*. Webseite. Oktober 2019. URL: https://www.elektronik-kompendium.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.

10.2 Abbildungsverzeichnis

Abbildungsverzeichnis

10.3 Personenverzeichnis

11 Anhang 1

Auflösung 1:

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

Beweis 1:

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