Regelungstechnik - PID-Parametrisierung anhand eines selbstgebauten Quadrocopters

Betreuer: Christian Prim, mit besonderem Dank an Bruno Thurnherr
Kantonsschule Zürich Nord
Charpoan Kong

Leitfragen

Wie muss die Regelung des Flightcontrollers implementiert werden, damit diese funktioniert? Welche Faktoren stören diese Regelung? Wie schnell müssen die Daten verarbeitet werden?

Spezifikationen:
- Controller: STM32F722RET6,ATMega328P
- IMU: ICM20689
- Barometer: BMP 280
- Schnittstellen: USB, UART, I2C

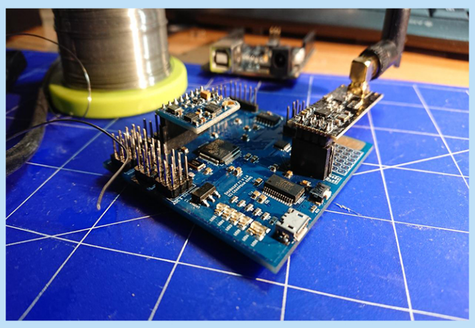

Abbildung 1: Fertig gelötetes Board

Die SMD Bauteile wurden mit dem Reflow-Verfahren gelötet. Zuerst wurde mithilfe der Schablone Lötpaste auf die einzelnen Lötstellen aufgegeben, anschliessend wurde die Leiterplatte mit einer bestimmten Temperaturkurve im Ofen verlötet. Die Rückseite hingegen wurde von Hand gelötet.

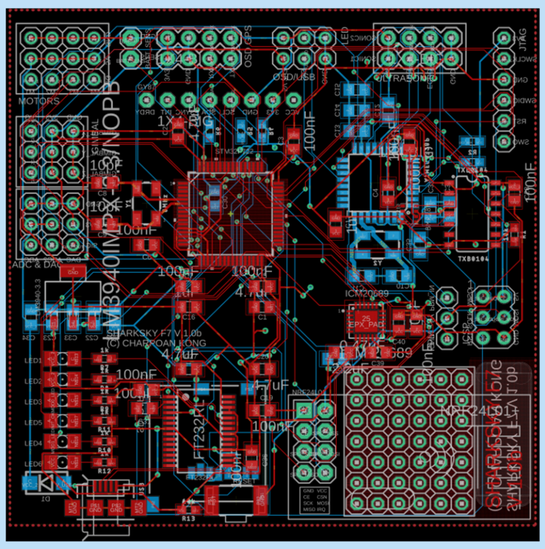

Abbildung 2: Board in CAD-Software

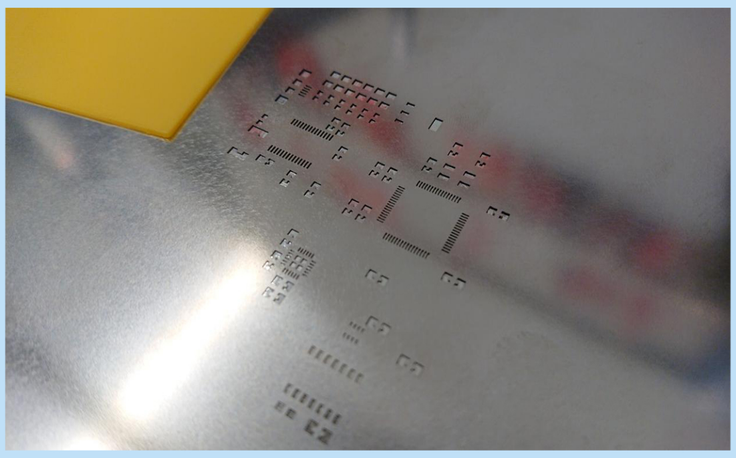

Abbildung 3: SMD-Stencil

Beim Design der PCB wurde darauf geachtet, dass das Board nicht zu gross wurde. Weiter lag der Fokus darauf, dass sich keine bzw. kaum Fehler im Design versteckten, da sonst alles hätte neu gekauft und gelötet werden müssen. Trotzdem ist der microSD-Slot verkehrt reindesignt worden. Die meisten Komponenten sind auf der Vorderseite, damit vieles nicht per Hand verlötet werden musste.

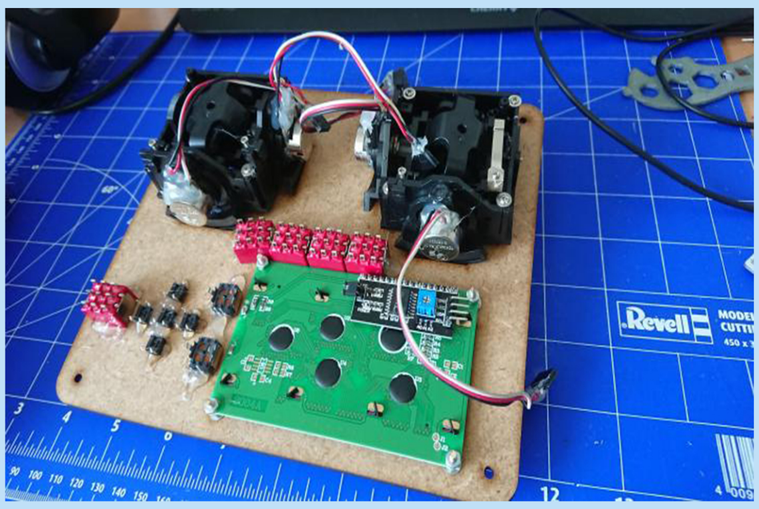

Abbildung 4: Rückseite der Fernbedienung

Die Fernbedienung ist sehr schlicht gehalten und es wurde kein Board dafür designt. Manche Teile stammen aus einer alten Fernbedienung. Diese kann Werte zum Neigungswinkel empfangen, es können aber auch die PID-Parameter gesetzt werden, wenn der Schub auf Null ist.

Das Programmieren umfasste den Hauptteil meiner Arbeit. Es wurde darauf geachtet, dass die Funktionen bzw. der Code schnell genug laufen und dass alles redundant ist, damit der Quadrocopter bei gewissen Winkeln keine falschen Kalkulationen macht. Einige mathematische Funktionen, wie beispielsweise der Sinus, wurden mit Näherungsverfahren kalkuliert, um kürzere Berechnungszeiten zu erreichen. Auch wurden für einige Komponenten Bibliotheken von anderen Personen benutzt, um Zeit zu sparen.

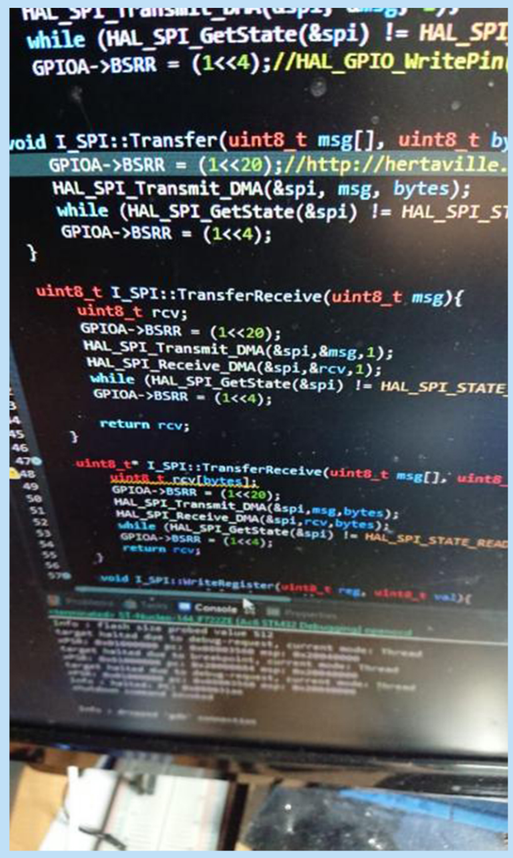

Abbildung 5: Kleines Beispiel aus der Software

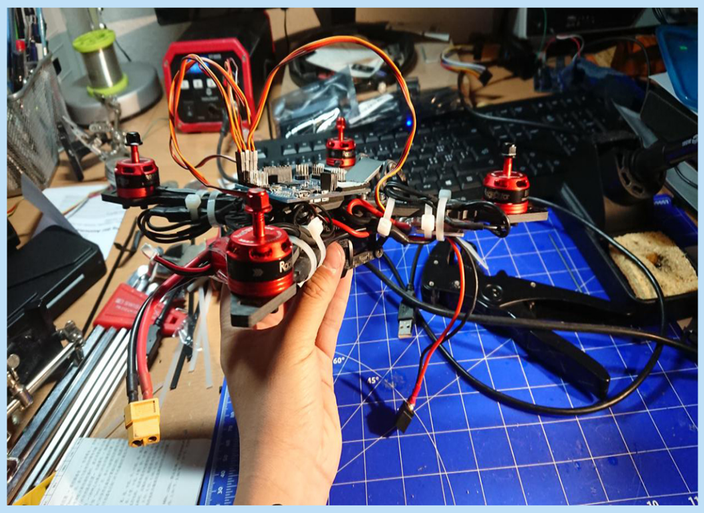

Abbildung 6: Zusammengesteckter Quadrocopter

Der Flight Controller reagiert richtig auf die Änderungen der Fluglage, jedoch stören die Vibrationen der Motoren den Neigungssensor. Deshalb wurde ein digitaler Tiefpass eingefügt, um die Werte zu glätten. Auch wurde das Programm optimiert, indem etwa für das Auslesen des Neigungssensors ein Direct Memory Access (DMA) verwendet wird, der Prozessorzeit spart. In zahlreichen Ver suchen wurden die korrekten Parameter gesucht und weiter eruiert, ob der Flight Controller schnell genug regelt, damit der Quadrocopter fliegt.

Fazit. Auch wenn der Quadrocopter richtig geflogen ist, müssen weitere Verbesserungen vorgenommen werden: der Einbau eines besseren Filters oder eine Stabilisierung des Programms. Zudem zeigte der letzte Versuch, dass der Quadrocopter fähig ist, senkrecht abzuheben, jedoch wurde die Fernbedienung dabei durch das schuleigene WLAN gestört, sodass der Quadrocopter mit der Decke kollidierte.