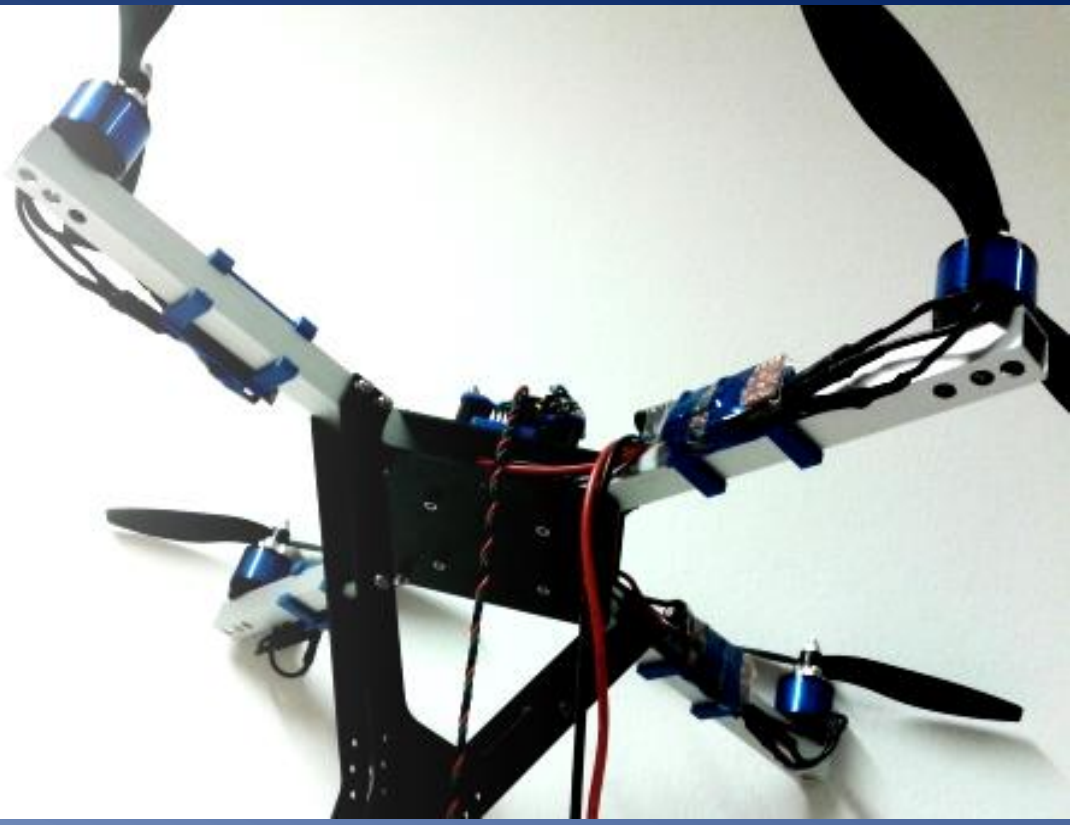
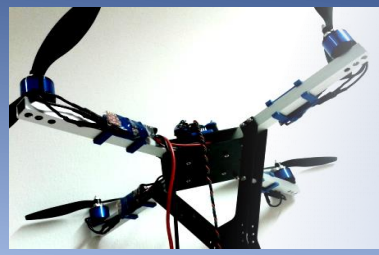
Matlab

PID-Regler

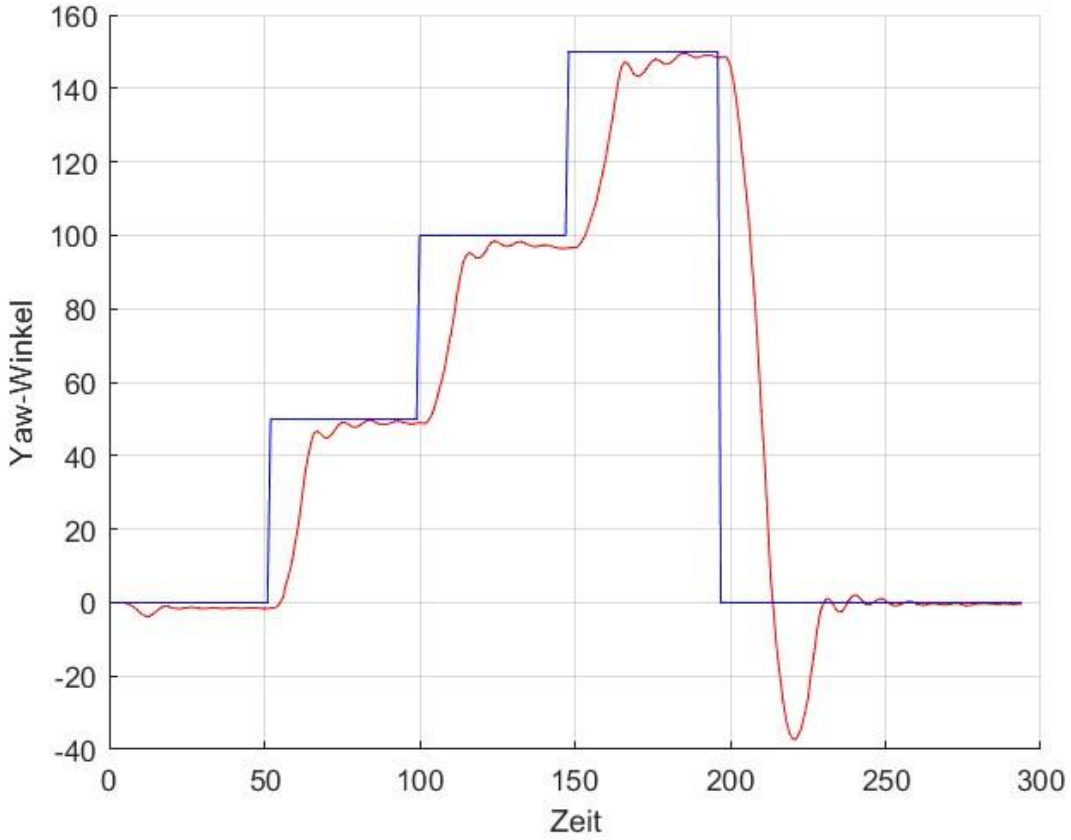


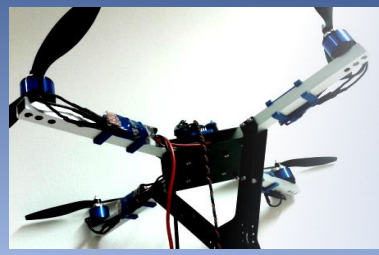


Inhalt

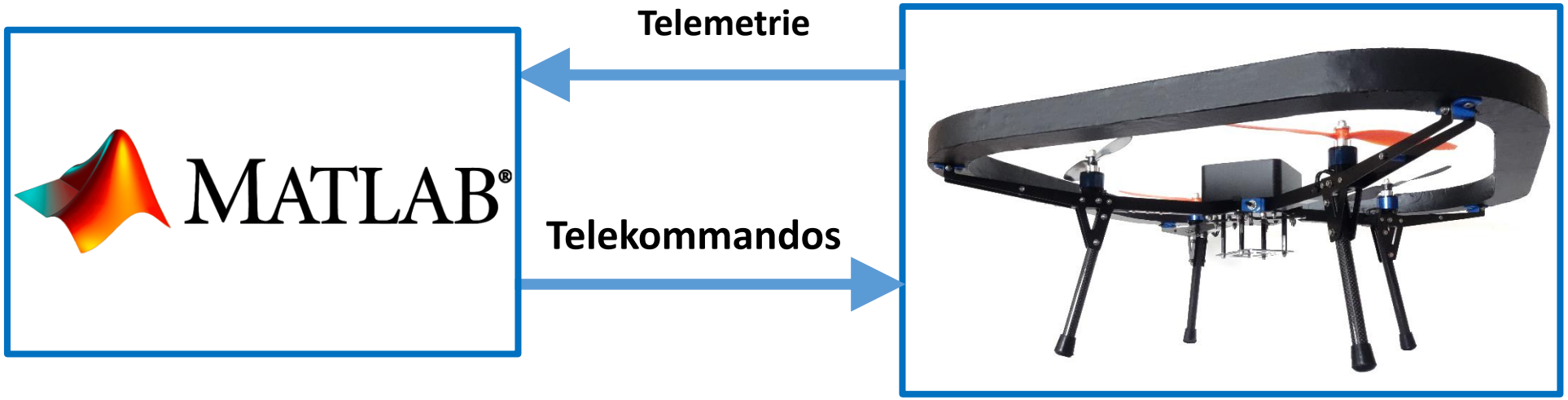
Umfang: ca. 2-3 Zeitstunden

- Matlab Telemetrie und Telekommand
- Yaw-Regelung
- Theorie und Regelung
- Superposition
- Aufgaben



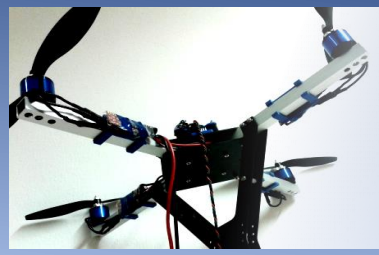


Matlab Telemetry & Telekommand



Pilot / Bodenstation
Auswertung / Optimierung

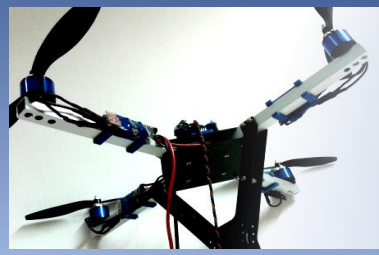
Quadrokopter



Yaw-Regelung

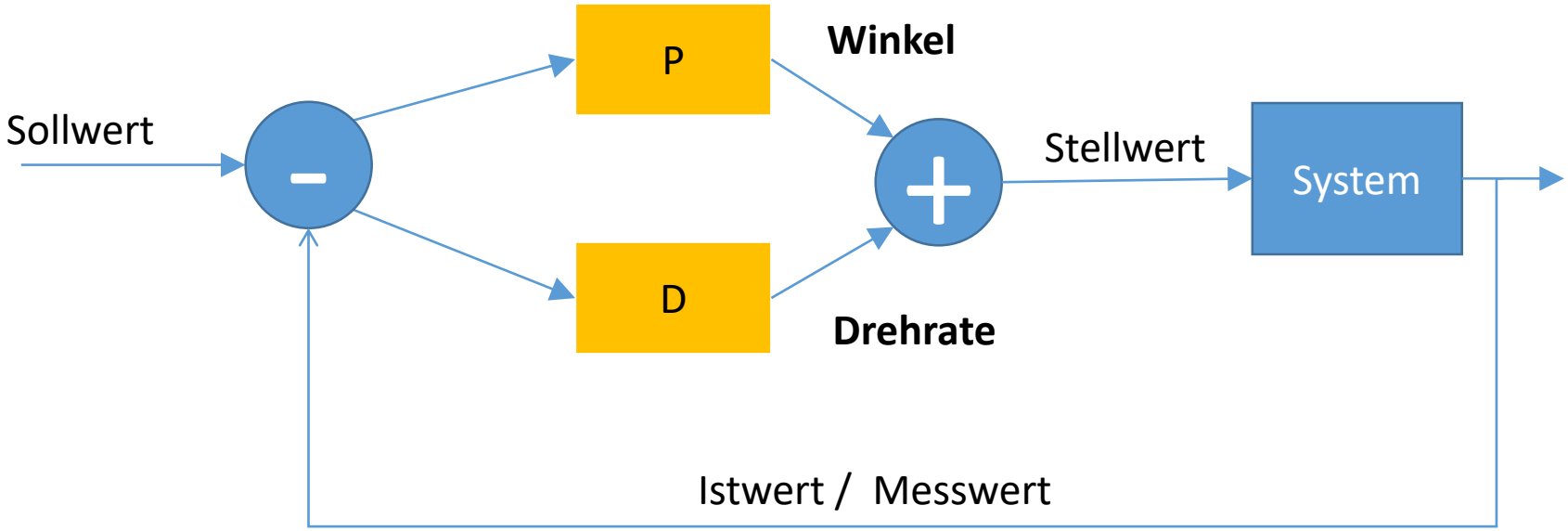
Definition Yaw-Regelung:
Das Drehmoment eines jeden Motors bewirkt, dass der Quadrokopter um die eigene Achse (z-Achse) giert. Deshalb drehen stets zwei Motoren links und zwei rechts ´rum. Damit dies mit der Lageregelung nicht kollidiert, drehen stets die gegenüberliegenden Motoren in die selbe Richtung.

Mit Hilfe eines Yaw-Reglers lassen sich dann Yaw-Manöver ausführen bzw. Störungen ausregeln. Verglichen mit der Lageregelung ist diese Regelung meist deutlich einfacher, weil das System nicht gegen die Gravitation ankämpft und somit sich gutwilliger verhält.

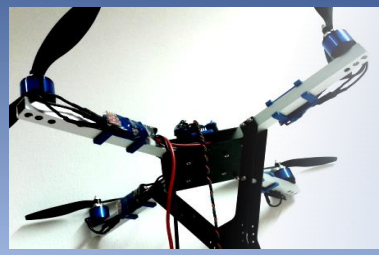


Theorie zur Regelung

Mit einem PD-Regler und einem Gyrosensor lässt sich das System regeln. Als Regelgröße dient der Yaw-Winkel.



Problem: 2 Motorpaare / Stellwerte , 1 Messwert (Winkel)
Lösung: Vereinfach durch Regelung der Differenz der Stellwerte beider Motorpaare



Superposition

Superposition:
Bei der Superposition werden die Regler einfach additiv überlagert. Die Superposition geht dabei davon aus, dass die Regler unabhängig voneinander sind.

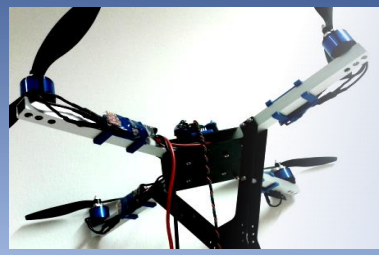
Ziel dieser Übung ist die Regelung eines Quadrokopters. Alle dazu benötigten Regler werden einzeln entwickelt. Zu Regeln sind die Freiheitsgrade RPY, damit der Quadrokopter in der Luft bleibt. Dies wird durch 2 (3) Regler realisiert:

- Attitude-Controller (x2 = Roll + Pitch)
- Yaw-Controller

Am Ende werden die beiden (drei) Regler einfach superpositioniert. Für den zweidimensionalen Fall (1 Gierregler, 1 Lageregeler) ergibt sich somit:

M1 = (**unsigned char)(START_GAS + roll + yaw);**
M2 = (**unsigned char)(START_GAS - yaw);**
M3 = (**unsigned char)(START_GAS - roll + yaw);**
M4 = (**unsigned char)(START_GAS - yaw);**

Beachte:
Die Superposition ist eine Vereinfachung. In der Realität beeinflussen sich die Regler. Daher ist bei der Realisierung noch mehr zu tun!



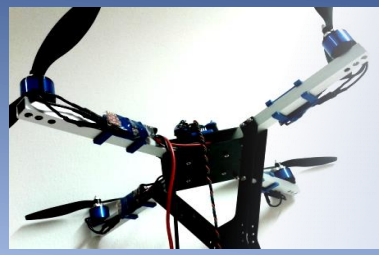
Aufgaben

Notwendige Hardware:

- EMQ3000
- QCS / QCSF
- Mini USB Kabel für USART-Verbindung und zum Flashen

Notwendige Software:

- Matlab Version >= R2019b, keine zusätzlichen Toolboxen
- AVR Studio 32 installiert (mit Tool Chain und FLIP Treiber)
- Projekt-Code und Library:
  EMQ_QCSF_Matlab



Aufgaben

Hilfe und Hintergrund:

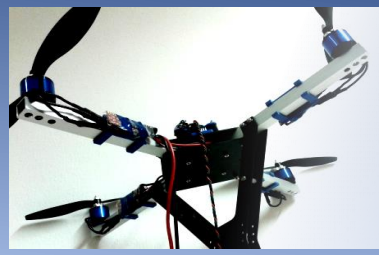
Die Aufgaben sind mit Hilfe des Projekts EMQ_QCSF_Matlab zu lösen. Legen Sie sich zusätzlich einen eigenen Dateipfad für Ihre Matlab-Dateien an und fügen Sie diesen Ihrem Matlab Workspace hinzu.

Aufgabe 1:

Aufbauend auf die Aufgabe „Matlab Telemetrie und Telekommand“ soll in dieser Aufgabe die Funktion set_Yaw_angle.m implementiert werden, mit der der Yaw-Winkel durch Matlab eingestellt werden kann. Rufen sie diese Funktion in ihrem Programm (aus Aufgabe 4 in der letzten Einheit) in einer Schleife mit Winkeln von 0 – 150 Grad auf und speichern sie den Plot inklusive Soll-Winkel. Beachten Sie dabei, dass der Regler genügend Zeit hat, um den entsprechenden Winkel zu regeln und erst danach der nächste Winkel geregelt wird.

Aufgabe 2:

Im nächsten Schritt werden nun die Funktionen set_P_Yaw, set_I_Yaw und set_D_Yaw implementiert, mit denen die P-,I- und D-Parameter für den Yaw-Winkel eingestellt werden können. Variieren Sie die PID-Parameter und vergleichen Sie das Regelverhalten des Systems. Versuchen Sie die PID-Parameter so einzustellen, dass das System möglichst schnell reagiert, aber nicht anfängt zu schwingen. Da my_Attitude_Contol auch roll und pitch regelt, müssen zusätzlich die PID-Paramter für roll und pitch gleich Null gestzt werden, damit das System stabil geregelt wird.



Aufgaben

Aufgabe 3:

In dieser Aufgabe ist das Ziel einen eigenen PID-Regler in Matlab zu realisieren, der Motorstellwerte an AVR übergibt.

a) Zur Vorbereitung der Motorsteuerung über Matlab muss das AVR-Projekt konfiguriert werden.

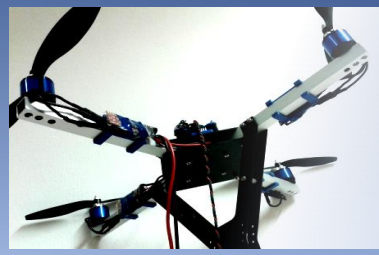
Öffnen Sie die Datei basics_qcsf.h und nehmen Sie folgende Änderungen vor:

#define USE_MY_ACTRL_FUNCTION	auskommentieren
#define USE_MY_MOTOR_STEERING	einkommentieren
#define ENGINE_ACTRL_ON	auskommentieren

b) Implementieren Sie die Funktion set_Motor_parameter.m, mit der für jeden Motor der entsprechende Stellwert per USART gesendet wird.

c) Nun gilt es mit Hilfe des Vorwissens aus der Lektion 8: „Gierregelung“ einen PID-Regler in Matlab zu entwickeln.

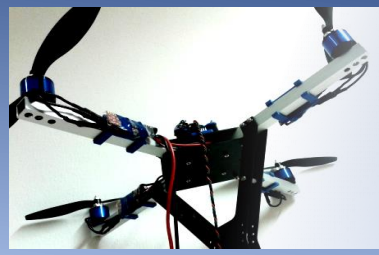
d) Vergleichen Sie die Reaktionen mit den in Aufgabe 1 abgespeicherten Plots. Sind Unterschiede zu erkennen? Woran könnten diese liegen?



Aufgaben

Hilfestellungen zu Aufgabe 3:

- Bei der Implementierung des PID-Reglers können Sie sich an der C-Funktion my_Attitude_Control in AttitudeControl.c im Ordner Control orientieren.
- Verwenden Sie für die Matlab Funktion des PID-Controllers globale Variablen und rufen Sie diese Funktion dann im Hauptprogramm in einer Schleife auf.
- Globale Variablen müssen in der Funktion, in der sie beschrieben werden zuerst aufgerufen werden und in jeder Funktion und jedem Skript, in dem sie verwendet werden zusätzlich als global deklariert werden.
- Stellen Sie sicher, dass der Regler mit den aktuellen Werten der IMU arbeitet, die über die Telemetrie ausgegeben werden.
- Wenn sie für den I-Parameter eine ähnliche Funktion wie die C-Funktion static verwenden möchten, kann das über die Funktion persistent realisiert werden. Genauere Infos zur Verwendung finden sich in der Matlab-Hilfe.
- Wenn der Controller zwar funktioniert, aber nicht gleichmäßig regelt, können die Pausezeiten reduziert werden. Jedoch sollte dabei berücksichtigt werden, dass die Pausen nicht so kurz gewählt werden, dass Fehler in der Kommunikation auftreten.



Aufgaben

Hinweise zur Regler-Dimensionierung:

Folgende Parameter/Vorgaben sollen Ihnen helfen: Beachten Sie, dass dies nur Richtwerte als Größenordnungshilfe sind, ohne Sample-Time oder Skalierung:

Sample-Time:	10ms		
Bias-Samples	1000 bis 2000		
P	3	| 6	
D	0.5	| 2	(bremst P ab)
I (optional)	0.001	| 0.01	(vs. stationäre Regelabweichung)

Vorgehen bei der Regler-Parametrisierung:

Fange mit D = 0 und I = 0 an und erhöhe P solange, bis das System „regelt“. Wenn es instabil wird, ist P bereits zu hoch. Wenn das System den Sollwert nie erreicht, ist P zu niedrig. Im Normalfall sollte das System oszillieren, d.h. hin und her schlagen. Als Nächstes erhöhe D um ein Überschwingen abzudämpfen. D sollte der Bewegung (Änderung) entgegenwirken.

Idee:

Der P-Anteil gibt an, wie schnell/heftig auf einen Fehler reagiert wird. Mit dem D-Anteil kann dann das System gebremst werden (Gegenregelung), wenn z.B. ein Fehler auftritt aber das System bereits in Bewegung ist.