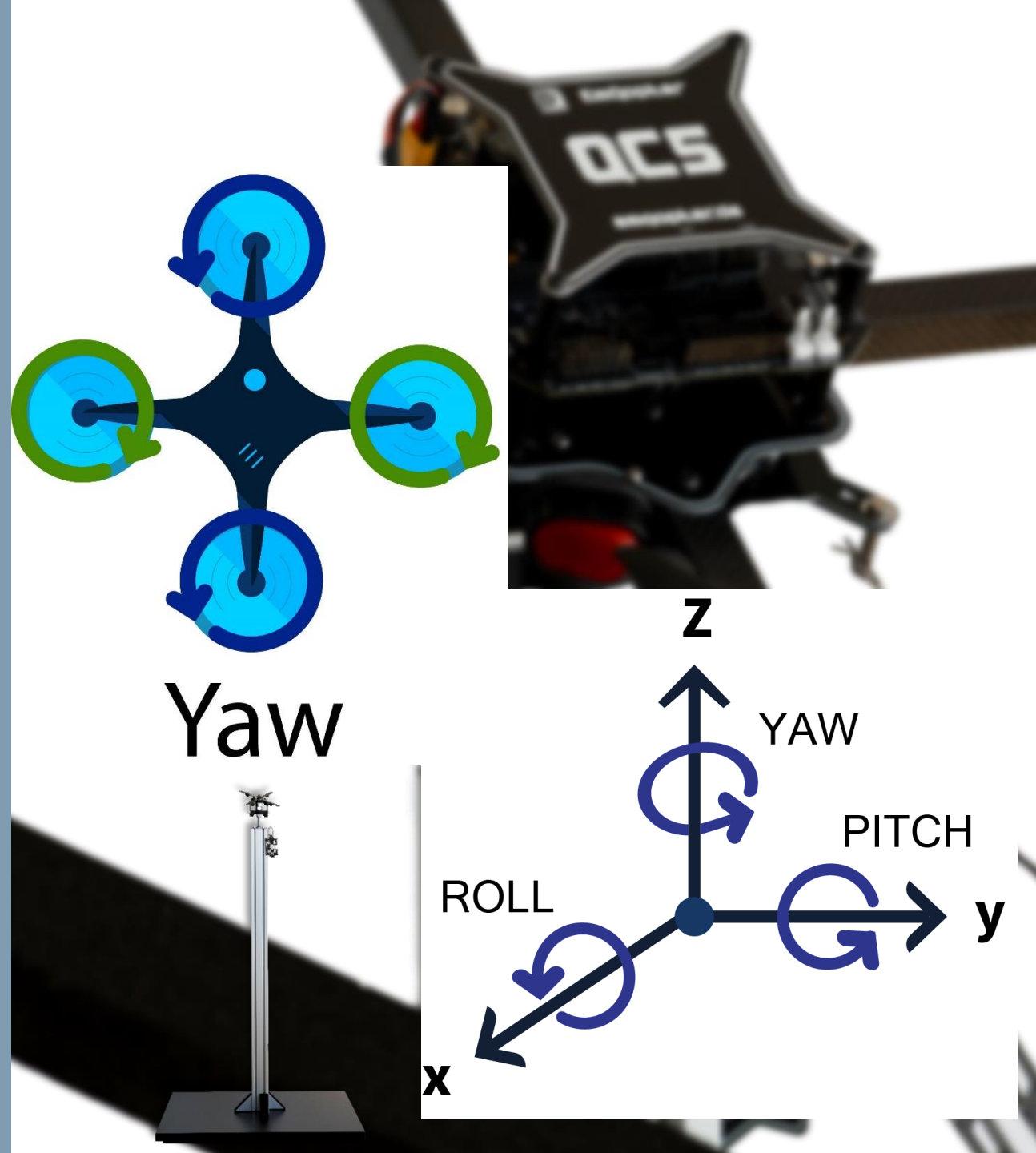


Yaw control

Lesson 8



Contents



- Yaw control
- Safety briefing
- Control theory
- Superposition
- Exercises & tips

Time scope: approx. 1-4 hours



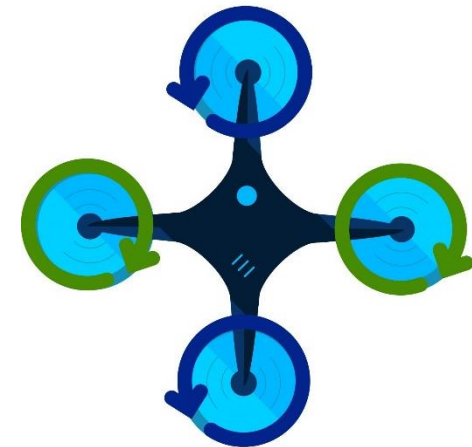
Yaw control



Definition Yaw-Control:

The rotation of each motor causes the quadrocopter to yaw around its own axis (z-axis). For this reason, two motors always rotate to the left and two to the right. To ensure that this does not affect the position control, the opposite motors always rotate in the same direction.

A yaw controller can then be used to carry out yaw maneuvers or compensate for disturbances. Compared to position control, this control is usually much simpler because the system does not have to fight gravity and therefore behaves more favorably.

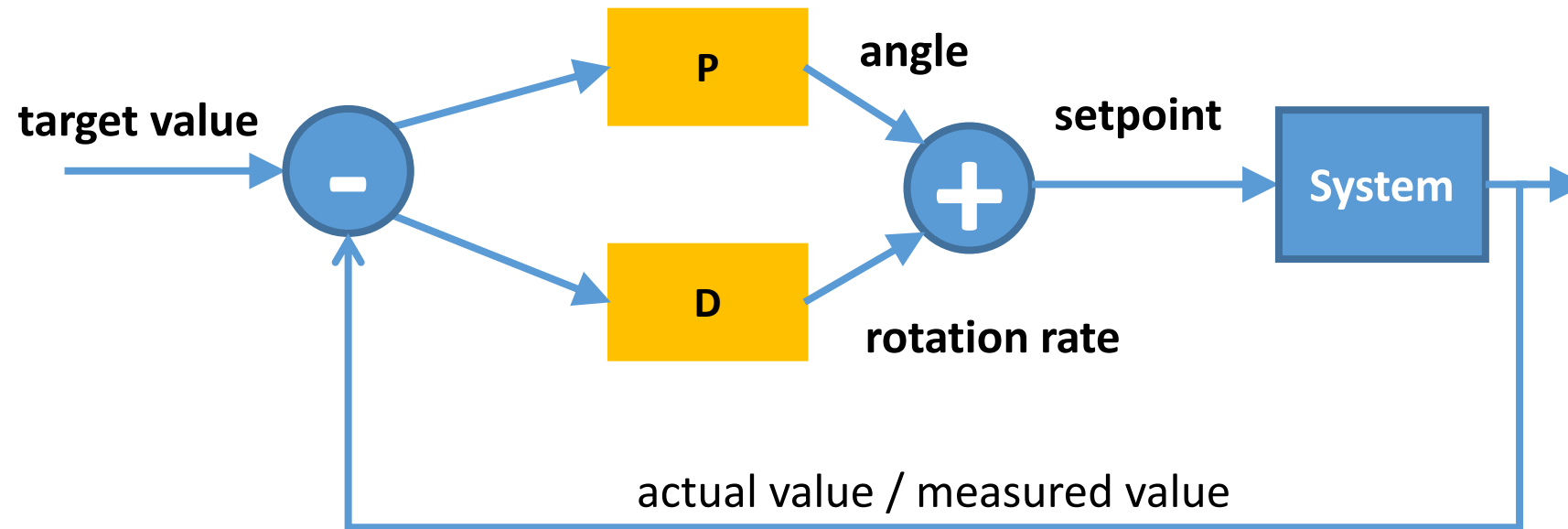


Yaw

Control theory



The system can be controlled using a PD controller and a gyro sensor. The inclination angle serves as the control variable.



Problem: 2 motors / setpoints, 1 measured value (angle)

Solution: Simplification by controlling the difference between the setpoints of the two motors

Superposition



Superposition:

With superposition, the controllers are simply layered additively. The superposition assumes that the controllers are independent of each other.

The aim of this exercise is to control a quadrocopter. All the controllers required for this are developed individually. The degrees of freedom RPY are to be controlled so that the quadrocopter remains in the air. This is realized by 2 (3) controllers:

- ***Attitude controller (x2 = roll + pitch)***
- ***Yaw controller***

At the end, the two (three) controllers are simply superpositioned. For the two-dimensional case (1 yaw controller, 1 attitude controller), this results in the following for the motor setpoints:

$M1 = (\text{unsigned char})(\text{START_GAS} + \text{roll} + \text{yaw});$

$M2 = (\text{unsigned char})(\text{START_GAS} - \text{yaw});$

$M3 = (\text{unsigned char})(\text{START_GAS} - \text{roll} + \text{yaw});$

$M4 = (\text{unsigned char})(\text{START_GAS} - \text{yaw});$

Please note:

The superposition is a simplification. In reality, the controllers influence each other. Therefore, more needs to be done during implementation!



Necessary hardware:

- EMQ3000
- Mini-USB cable for power and flashing
- Mini-USB cable for communication: USART to PC
- QCS in 1DOF yaw control mode (later switchable to 2 DOF)

Necessary software:

- AVR Studio 32 installed (with toolchain and flip driver)
- EMQ_Framework (AVR-Code)
- Documents:
 - EMQ_Framework.pdf
- Qt Creator 2.0.1 and Qt 4.7.0 (32bit)
- EMQt_Framework\quatplay\quatplay.pro (Code)

Safety briefing



Safety briefing:

- Before starting the motors, always check that everything is tight:
This applies to all screws, motors, booms, Y-parts, knurled screws, etc.
- **ONLY** connect the power supply unit and start the motors with the authorisation of the supervisor.
- Keep your distance. Warn neighbours. Exercise caution. Have the on/off button ready (finger over it)!



Emergency stop options:

- On/Off button
- Switch off or disconnect the power source
- Disconnect **I2C** cable (pull out)
- Switch off **MCU** (disconnect USB cable)

Exercises



Exercise 1:

Implement a PD controller as a yaw controller that controls the yaw of the quadrocopter. Set the controller parameters empirically by trial and error. The controller must be able to compensate for disturbance pulses.

Note:

By using the **lock plates**, the quadcopter can be fixed in a horizontal position so that attitude control is not necessary. Use the Yaw configuration of the tripod for this task.

Exercise 2:

Now merge the two controls (3DOF configuration of the tripod). You should notice that the controllers influence each other. Nevertheless, the overall behavior must not deteriorate significantly. In particular, strong yaw disturbances should be compensated for without the system tilting. It is desirable if the system can continue to yaw quickly.



Notes on controller dimensioning:

The following parameters/specifications should help you: (only approximate values as a guide)

Sample-Time:	10ms		
Bias-Samples:	1000 bis 2000		
P	3	6	
D	0.5	2	(slows down P)
I (optional)	0.001	0.01	(vs. steady-state control deviation)

Procedure for controller parameterization:

Start with **D = 0** and **I = 0** and increase **P** until the system "**regulates**". If it becomes unstable, **P** is already too high. If the system never reaches the setpoint, **P** is too low. Normally, the system should oscillate, i.e. beat back and forth. Next, increase **D** to reduce overshoot. **D** should counteract the movement (change).

Idea:

The P component indicates how quickly/severely an error is reacted to. The D component can then be used to slow down the system (counter-control) if, for example, an error occurs but the system is already in motion.