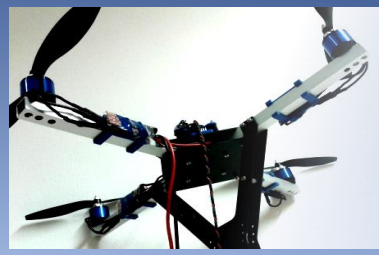
# Attitude Control

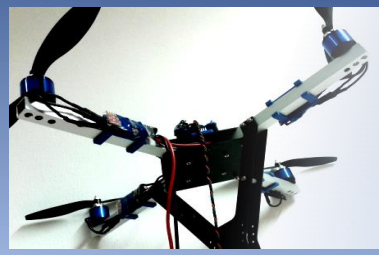
## Actuators and Attitude Control



# Content

**Time scope: 2-4h**

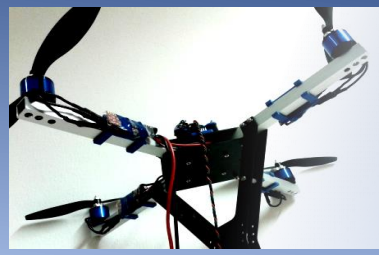
- Attitude Control
- Brushless motors
- Brushless controllers
- EMQ Framework
- Safety Instructions
- Control Theory
- Exercises & Hints



# Attitude Control

**Definition Attitude Control:**
Attitude Control is the core of the quadrotor and keeps it up in the air. In contrast to other aircraft systems (e.g. planes, helicopters), quadrotors are instable systems and require permanent control of the roll and pitch axes:

-> 2 attitude controllers control the attitude in the z-plane
-> corresponding to the problem of seesaw control: 2 motor control



# Brushless motors

**Brushless Motors (Bl-Motors):**

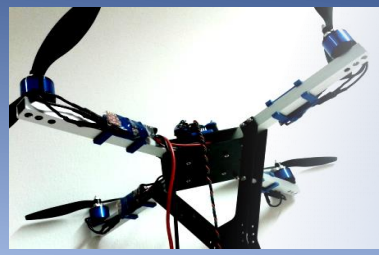
We use  brushless motors which work with electronic commutation in contrast to brush motors. Despite their name, the functional principal is comparable to three-phase-motors with permanent magnets and three phases: (+/-/0)

Advantages of Bl-Motors:

- High power density (power/mass)
- High turning rates
- Low wearing (Bearings, no brushes)
- Very popular with high variety

Disadvantages:

- Requires a brushless controller to generate the synchronous rotating (commutation) for the motors over 3 phases
- Costs
- Difficult / expensive to realize low turning rates



# Brushless Controllers

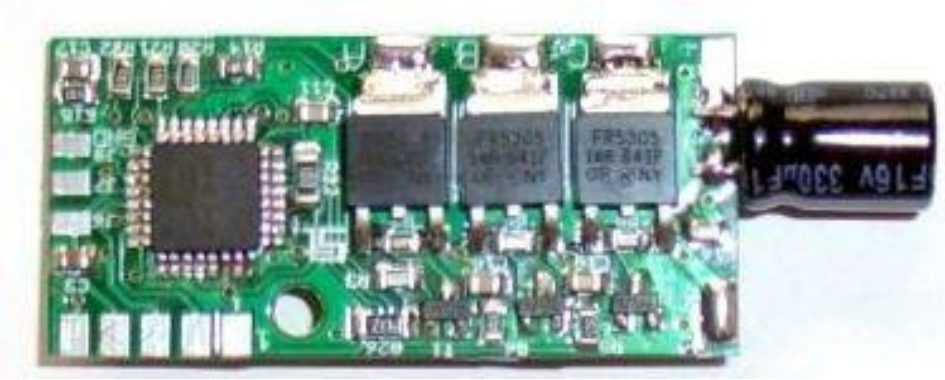
**Brushless Controllers:**

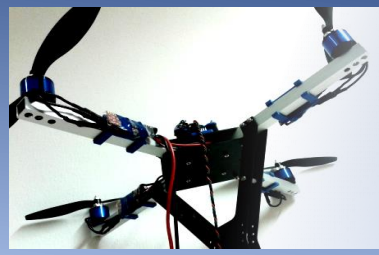
The Brushless Controller (BlCtrl), Electronic Speed Controller (ESC)) drives the motors and is controlled via I²C. It is responsible for the commutation, this means the switching of the phases in the optimal moment. The input voltage to the Bl-motor (normally controlled via PWM) defines the rotation speed, whereas the switching is done in the moment of the zero passing to guarantee a continuous rotation of the motor.

Advantages of the used BlCtrl:

- Wiring (I²C)
- Very fast set point positioning time < 0,5ms

Set point: 1 Byte U8
Range of values: 0 to 255





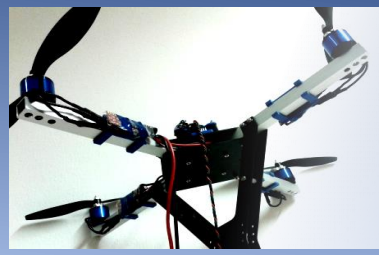
# EMQ Framework

EMQ Framework

- Delivered as a library with open source parts
- Developed exclusively for the quadrotor lab
- Contains drivers with functions for:
  - Hardware: CPU, IRQ, TC, Remote, ADC, USART, TWI, Delay, motors, LED
  - Software:
    - Basics for easy start
    - Function templates
- For details check software documentation: EMQ_Framework.pdf
- The EMQ Framework does not use an operating system
- The whole codes runs in an endless loop (while)

EMQ Data types

- The file EMQ_Interface_Data.h contains several available EMQ data types
- Data types including dummies are meant to be filled with content later on.
- All other data types should **not** be changed!
- The system time is available in the variable tc_ticks [ms].



# EMQ Framework

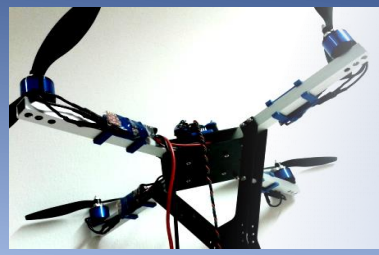
Motor driver:

The framework contains two functions to simplify motor control:

- Motors_Init
  Initializes the motor driver together with the communication protocol. This function has to be called once before the motors can be used.
- Motors_run
  Starts the motors accordingly to the passed set point values of the specified struct motor_data (do not change the struct!). The set point values are passed on as one byte (character) for each motor. The motor set point values have to be transmitted continuously. Otherwise the motors will turn off.
- steering.engine_on
  The motors will start only if steering.engine_on is true (e.g. > 0). The struct steerData must not be modified!

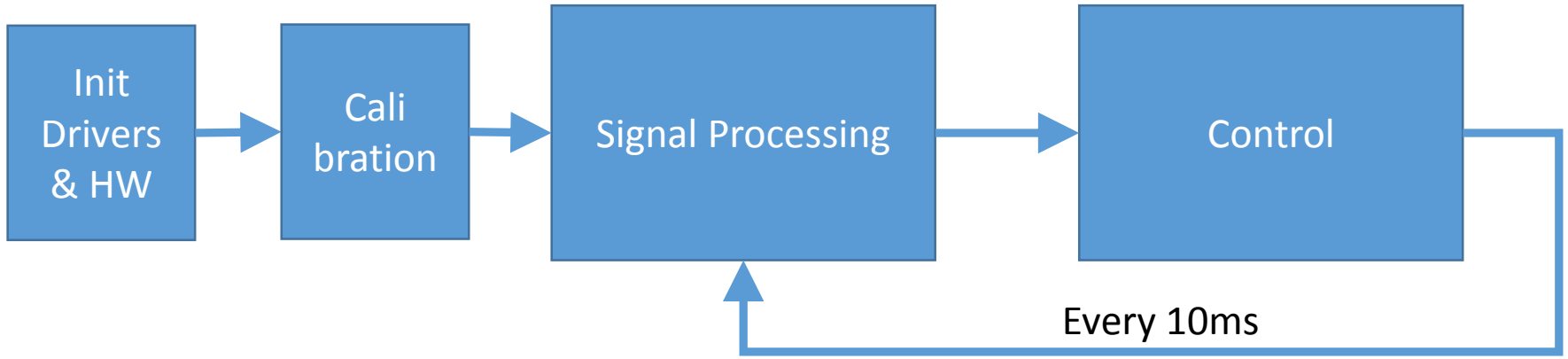
```
// Initialize motor
communication packages
void Motors_Init();

// Run 4 Motors with set
points of motor_data
inline void
Motors_run(motor_data *m);
```

```
// Motor Values
typedef struct {
unsigned char M1;// Motor 1 (left)
unsigned char M2;// Motor 2 (rear)
unsigned char M3;// Motor 3 (right)
unsigned char M4;// Motor 4 (front)
} motor_data;
```



# EMQ Framework

**Overview:**

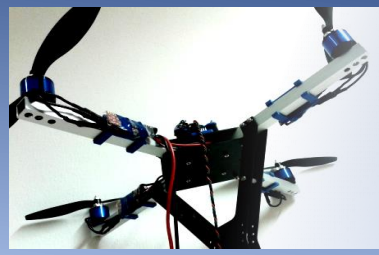



Signal processing:

- Reading the sensor values
- Conditioning the sensor values (processing)
- Data Fusion (e.g. Kalman Filter)

Control:

- Implementation of the controller (PID/PD)
- Driving the motors with the new motor set point values
- Important: The sample time has to be adhered to!



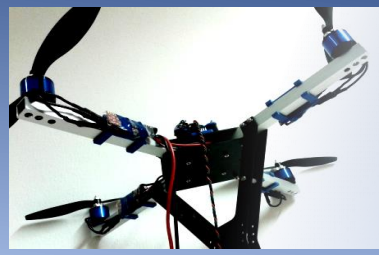
# Safety instructions

**Safety instructions:**

- Check the system and retighten all parts (screws/propellers/motors/arms/joints/bolts/nuts) if necessary before every start!
- Only connect and start the motors under the supervision of your teacher!
- Keep a safety distance of at least 50cm to any propeller.
- Take care for your neighbor and yourself! Be ready to turn the system off immediately in case of emergency!

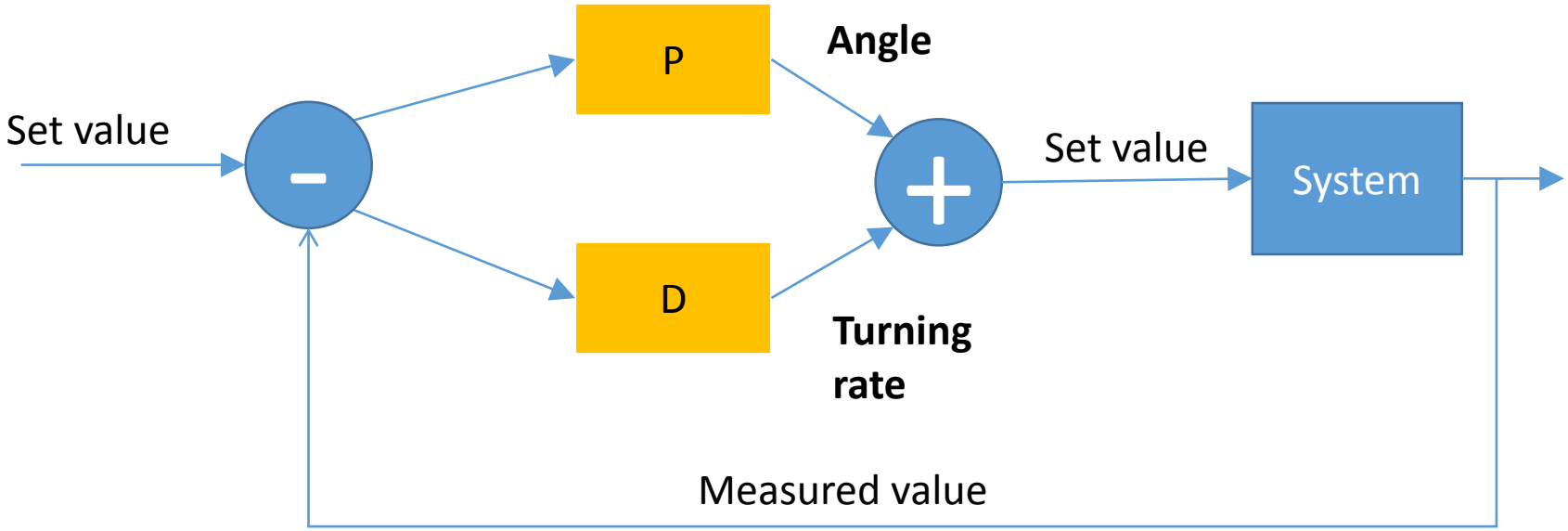
**Emergency Stop:**

- Unplug / Shut down Power
- Unplug TWI Cable
- Shut down MCU

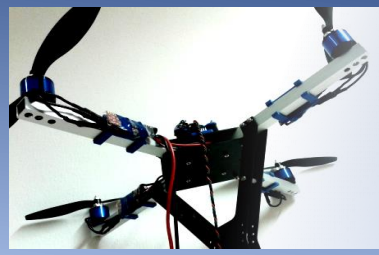


# Control Theory

The system can be controlled with an PD controller using a gyroscope. The control variable is the angle of inclination.




Problem: 2 Motors / set values, 1 value of measurement (angle)
Solution: Simplification by controlling the difference of the motor set values



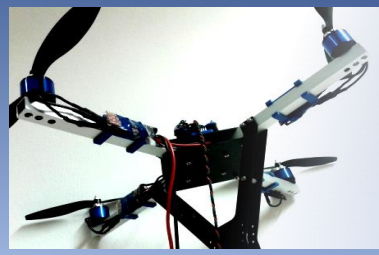
# Exercise

Required hardware:

- EVK1100
- Micro USB cable for power and flashing
- QCS in 1DOF Attitude Control Mode

Required software:

- AVR Studio 32 (with Tool Chain and FLIP Driver)
- EMQ Framework (Code)
- Documents:
  - EMQ_Framework.pdf



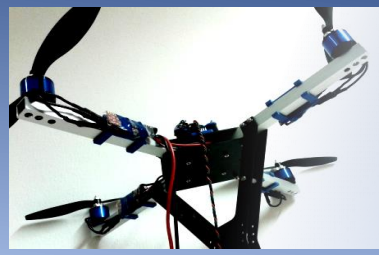
# Exercises

**Exercise 1a:**
Write a program to drive the brushless controllers and control the motors.

Implement a test function to turn each motor consecutively from set point 0 up to 50 in one run. Start with motor 0 and increment the motor set value so that it reaches the set point of 50 after about 5 seconds and stop the motor. Do the same for the other motors accordingly.

**Exercise 1b:**
Use the EVK 1100 to display the motor set point values of the 4 motors for debugging.

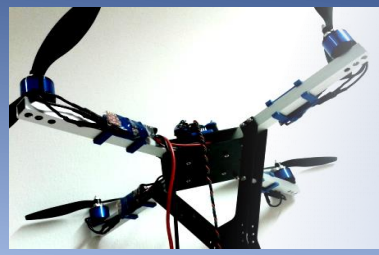
**!!! SAFETY FIRST !!!**
**Remember: Contact the teacher / supervisor and make sure that nobody can be hurt before you start the QCS!**



# Exercises

**Exercise 2:**

Implement a PD Controller for Attitude Control which keeps the QCS in horizontal position. Set the PD parameter values empirically by trial and error. The exercise is completed successfully when the controller stabilizes the system back to the horizontal position in a short time and compensates for small disturbances. You won't be able to reach a perfect solution with this approach yet (see next slides). Do use the gyroscope for control only!



# Exercises

**Hints for control parameter dimensioning:**

Stay within the following parameter ranges for a guideline (without sample time or scaling):

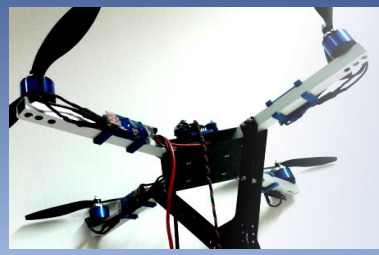
| | | | |
|---|---|---|---|
| Sample-Time: | 10ms | | |
| Bias-Samples | 1000 to 2000 | | |
| P | 2 | \| 4 | |
| D | 0.4 | \| 0.8 | (Stops P) |
| I (optional) | 0.01 | \| 0.02 | (To solve stationary control deviation) |

Approach for controller parametrization:

Start with D = 0 and I = 0 and increase P until the system starts to oscillate. If P is too high, the system is instable. If P is to small, the system will never reach its set point. Continue with increasing the D parameter to counteract an overshooting. D should work against the current movement of the system.

Idea:

P defines how strong the controller reacts to errors. D slows down the system movement (Counter Controlling).



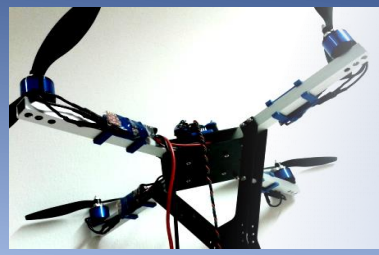
# Exercises

**Hints for control parameter dimensioning:**

Finding a stable system setting is a complex problem. If the sensor values are bad already, e.g. jumps due to sensor vibration, the best controller will fail all the same. This is why you should start setting the controller only with the gyroscope, since vibrations do directly effect the values from the accelerometer. It is much more difficult, to find the settings for the Kalman Filter (KF) and parameterize the controller at the same time. So start and use the angles, calculated by integration of the gyroscope values, to set the controller parameters. Later you can remove drifting effects by using a Kalman Filter. Moreover you have to make sure that the sample time is kept very precisely and does not vary (to much)!

When the system is instable (oscillating), the reason behind this is due to – as long as P is not set too high – a wrong D setting. The system will also not be stable if D controls in the wrong direction or is set too high or too low. Determine the sign of the D part first, so that it reacts against the movement. Then set D so that it damps P but is not too strong, which would cause (little) oscillating again.

An overlarge integrational part (I part) will lead to instability as well. The I part is used to solve stationary control deviations and should be added at the end with high caution. Stability can be reached without the use of the I part which is why the I part should be left at 0 at the beginning.



# Exercises

**Exercise 3:**

Use the Kalman Filter for data fusion of gyroscope und accelerometer. Parameterize the Kalman Filter so that vibrations do not falsify the orientation The system should not drift anymore. Add a small I part (if not done already) to overcome stationary control deviations.

Hint:

Exercise 3 is optional. Work on the exercises for the Kalman Filter before.