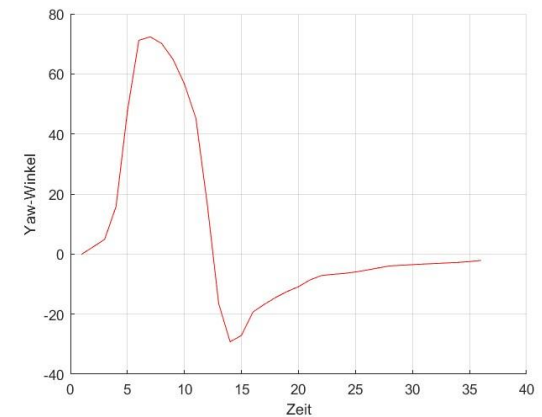
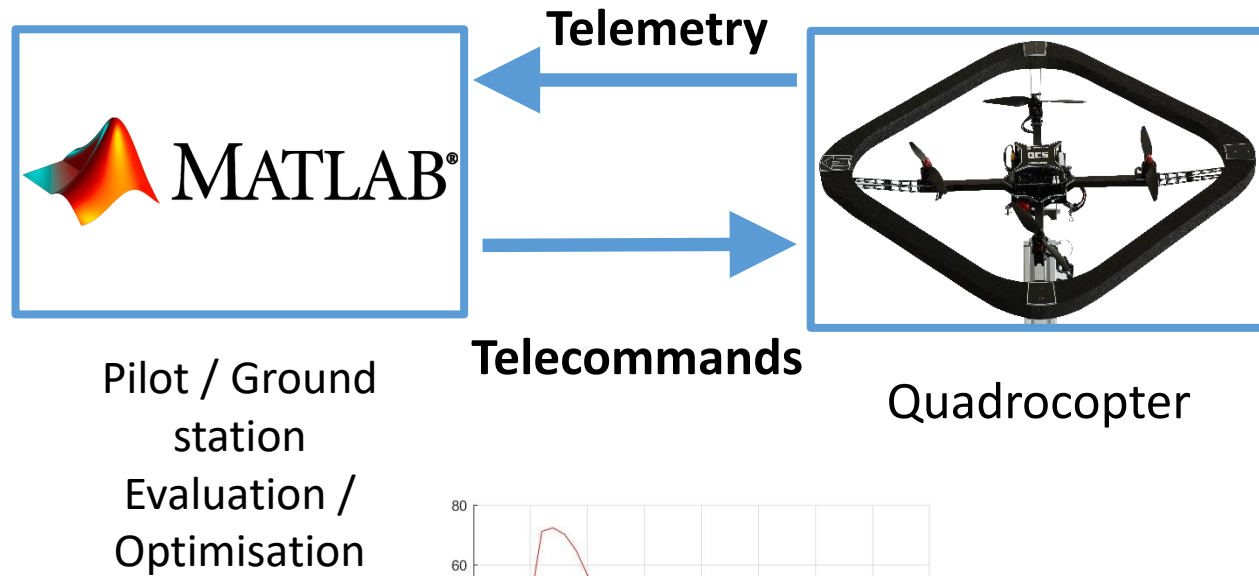


Matlab

Telemetry & Telecommands

Lesson 11

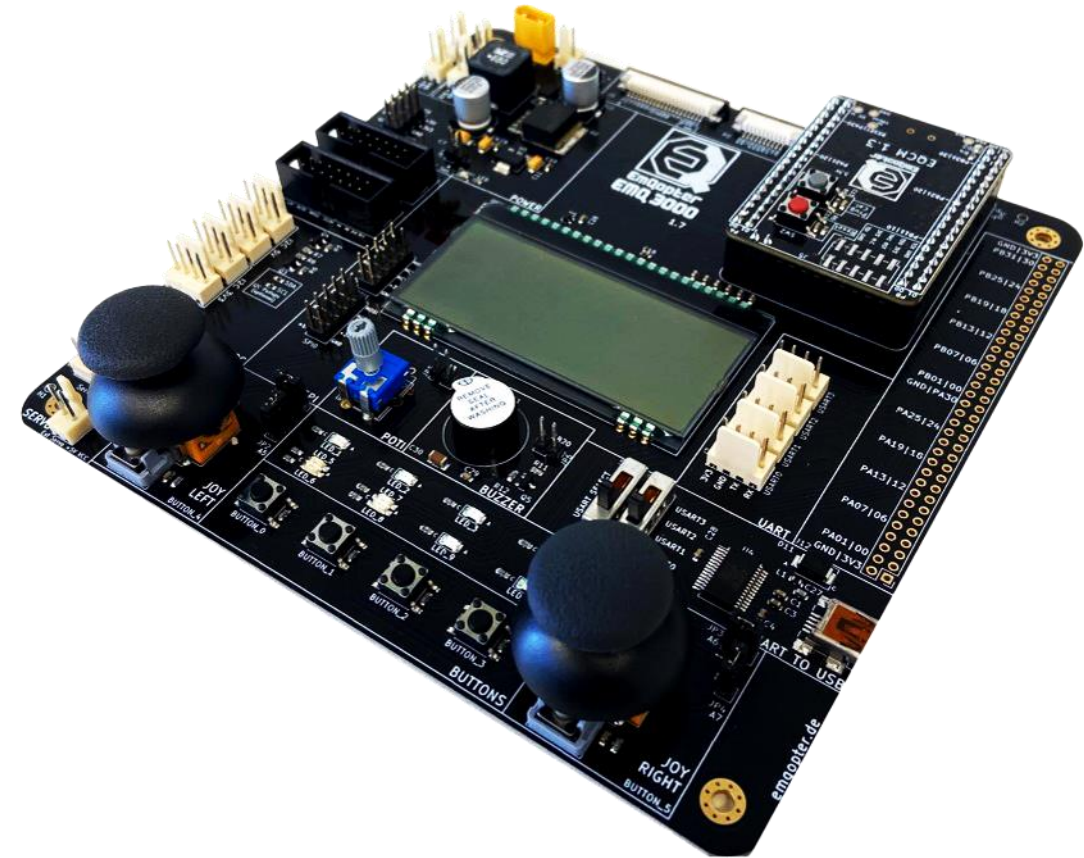


Content



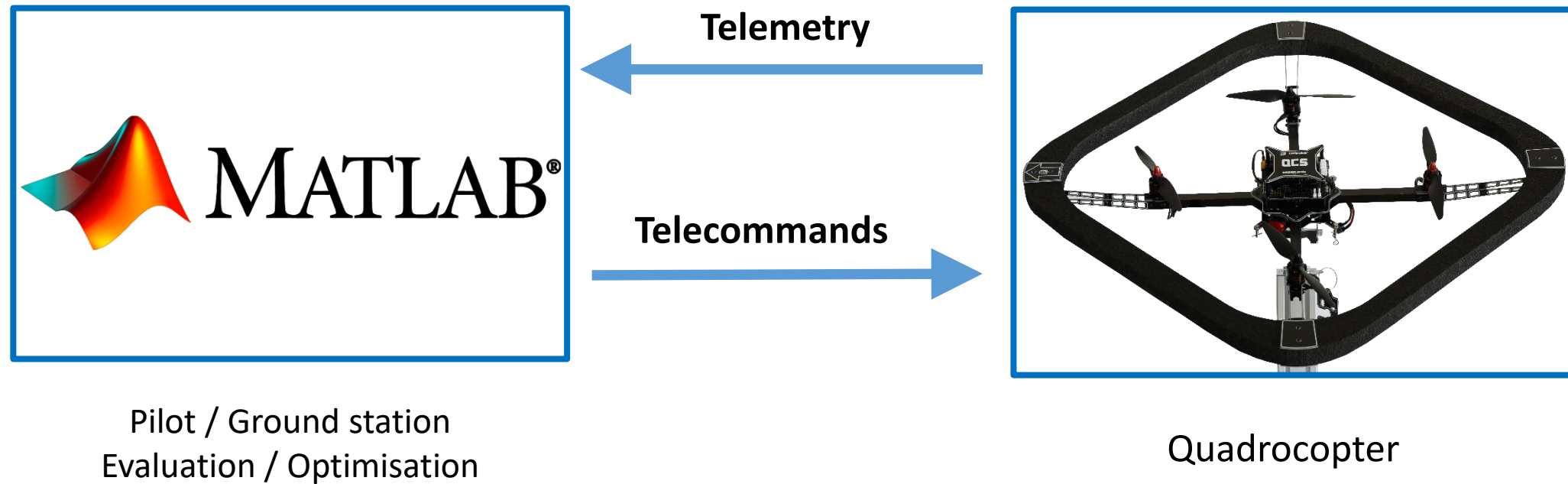
- Matlab: Telemetry & Telecommands
- Matlab: Serial communication
- Matlab: Plotting data
- Matlab: Creating own functions
- Exercises

Time scope: approx. 2-3 hours



Matlab

Telemetry & Telecommands



Matlab:

Telemetry & Telecommands



- The already known telemetry has been extended to be able to communicate with Matlab.
- The files are located in the COM folder of the „**EMQ_QCSF_Matlab**“ project:
 - ***matlab_protocol.h***: Defines the communication protocol
 - ***matlab_telemetry.c/.h***: Sends regularly activated data to Matlab
 - ***matlab_telecommandos.c/.h***: Reads and executes telecommands from Matlab
- In terms of functionality and their structure, they are similar to the already familiar telemetry. Only the syntax, i.e. the structur of the communication, has been adapted.

Matlab: Serial communication



- In Matlab, the class serial port has been available since version R2019b, which can be used to represent and control serial interfaces in Matlab.
- If an object of the class serial port, communication can be established with the QCSF via corresponding COM port of the PC using USART.

- **Required functions:**

- Init: `device = serialport(port, baudrate, timeout, value);`
- Send data: `writeline(device, data);`
- Receive data: `data = readline(device);`
- Flush buffer: `flush(device, „input/output“);`
- Disconnect: `clear device;`

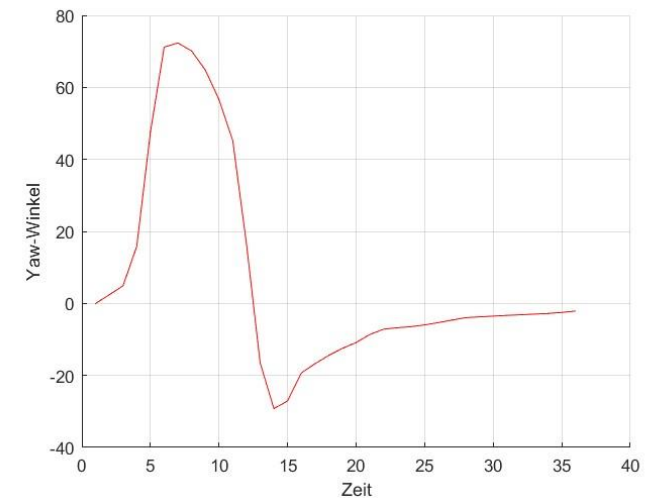
Matlab: Plotting data



Required functions:

- Create figure:
- Activate grid:
- Enter new data in the same figure:
- Axis labelling:
- Plot data:

```
figur;  
grid on;  
hold on;  
ylabel(„Y-labelling“);  
xlabel(„X-labelling“);  
plot(data, colour, etc.);
```



Matlab: Creating own functions



- Create a new Matlab file in the workspace with the same name as the function (e.g. **myfun.m**)

```
function [y1, y2] = myfun(x1,x2)
    y1 = x1;
    y2 = x2;
end
```

- At the end of the function, the return values (y1,y2) are automatically transferred to the calling programme.



Necessary hardware:

- EMQ3000
- QCS / QCSF
- Mini-USB cable for USART connection and flashing

Necessary software:

- Matlab version → R2019b, no additional toolboxes
- AVR Studio 32 installed (mit toolchain and FLIP driver)
- Project code and library: **EMQ_QCSF_Matlab**



Help and background:

The exercises are to be solved using the „**EMQ_QCSF_Matlab**“ project. In addition, create your own file path for your Matlab files and add this to your Matlab workspace.

Exercise 1:

Create a file **main.m** and initialise the interface to the **EMQ3000 Board** using the class serial port. Note that the **baud rate** is **57600** and **the timeout** is set to **0.5**. At the end of the programme, the serial port should be enabled again.

*Why does it make sense to select a timeout?

Exercise 2:

Familiarise yourself with the structure of Matlab telemetry and the Matlab telecommands. Which syntax does the respective communication structure follow?

Note:

*Have a look at the **matlab_protocol.h** file in the Com folder on the **AVR side**.*



Exercise 3:

Implement a MATLAB function **init_Telemetry.m** with which the telemetry for the PID parameters and the RPY data can be activated and deactivated. Activate the telemetry for the RPY data in **main.m**.

Note:

*Have a look at the methods **writeline** and **flush** of the class serial port and find out which values the **matlab_transcommand** method expects. To avoid errors in communication, the method should be executed twice in succession with a pause in between.*

*Why do these measures minimize errors?



Exercise 4:

Now that the telemetry for the RPY data has been activated, a real-time plot of the yaw angle is to be implemented. To read the yaw angle, either use the predefined function **read_Telemetry.m** or write your own (for advanced users). Now you can adjust the yaw angle of the QCS by hand and see the angle as a function of time in Matlab.

Note for your own implementation:

*Have a look at the **readline** method to read out data from the USART. To extract the telemetry data, the **split** method can be used. Familiarize yourself with global variables in Matlab and follow the instruction in the specified functions.*

