# Flight Controller basic concept and structure

## What is a Flight Controller?

The heart of every UAV is the flight controller. This unit is responsible for keeping the aircraft stable in the air while executing the commands, the UAV gets from the user via the Remote Control (RC). There are several commercial flight controllers available that could be mounted on the XCopter as described in section **“****Commercial Flight control”.** For the purpose of this project an own, non-commercial flight controller is implemented. It’s basic idea or concept and structure as well as the dataflow is described in the following subchapters and figures.

## Basic concept of a flight controller

For every flight controller the basic concept is more or less the same: The UAV has sensors such as a gyroscope, an accelerometer and (sometimes) a compass on board. The main control loop does the following as long the UAV is in the air: With the data of the sensors, the pitch, yaw and gear orientation (see REF Figure 1) of the UAV is calculated.

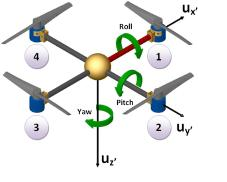


Figure 1: Roll, Pitch and Yaw orientation [https://xplorenlearn.files.wordpress.com/2014/11/yaw\_pitch\_explain.png]

The actual orientation and the desired position, which is transmitted by the user via the remote control, are the inputs for the PID regulators. (For a description of what a PID is, see **“****PID Regulator”.**) Depending of the output of these PID’s the individual motors have to run at different speeds to stabilize the UAV in the air.

## Structure of the XCopter flight controller

In the XCopter project, the flight controller is separated in three tasks. One task handles the remote control input, the second task is responsible for gathering the sensor data, and the third task holds the main controlling algorithm. The three tasks are synchronized and share their data with mutexes and semaphores. The detailed properties and timings of the tasks are described in a later section **“Tasks and Timing”**. You can see the program flow in more detail in the chapter REF **“Program flow”**.

In REF Figure 2 a component diagram, that shows the structure of the flight controller, can be seen. Please note that the position of the components has no meaning, it’s just for a better readability. All the hardware drivers, which are providing the interfaces to communicate with the sensors and motors, are colored in white. The three parts, mentioned above, are highlighted with different colors. The ‘Sensor Data Manager’ is colored in pink and its details are described in section **“Sensor Data Manager”.** The ‘RC Receiver’ is colored in green and is elucidated in section **“RC Receiver”.** The main controlling part contains multiple components, is colored in yellow. Its particular components are described separately in later sections, whereas the functionality will be pointed out in the following section.

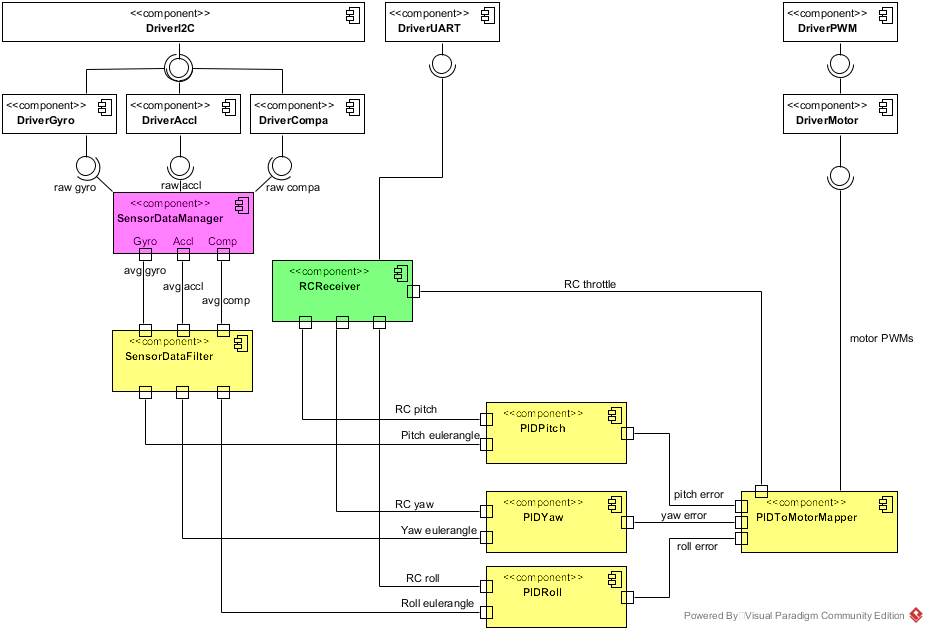


Figure 2: Flight Controller structure

## Dataflow through the flight controller

Basically, what a flight controller does, is getting data from the sensors and the remote control and translate this into motor revs. Or the other way round: The motors are running at different velocities depending on the data it gets from the user (remote control) and the sensors. Therefore, understanding the dataflow is very important because it is also big error source. In the following an overview of the dataflow in the XCopter flight controller is given. For more details of the data, e.g. the ranges, in which the values are, please see section REF **“Interfaces and Ranges”.** More details about the different components can be found in their respective chapters.

In Figure 3 an overview of the dataflow through the flight controller can be seen.

At first, the Sensor Data Manager collects the raw data from the accelerometer, the gyroscope and the compass.

Then the raw data gets averaged and handed over to the Sensor Data Filter. This component then filters the data and computes the current orientation of the XCopter as euler angles (pitch, yaw, roll) by combining the data from the sensors.

When the euler angles are calculated, the second data source, the remote control comes into play. The remote control also provides euler angles depending on the orientation of its joysticks.

The euler angles from the Sensor Data Filter and the Remote Control are then fed into the particular PIDs. Each of those PIDs computes an error value for each axis.

The throttle value, which also comes from the Remote Control, and the PID error values are the input of the PID to Motor Mapper. This component translates the error values in PWM signals that are then mapped on the motors by the motor drivers.

Sensors

Remote Control

Sensor Data Manager

Sensor Data Filter

PIDs

PID To Motor Mapper

Motors

averaged sensor data

raw sensor data

euler angles

PID error values

motor PWM signals

euler angles

throttle

Figure 3: Dataflow through the flight controller

Program flow of the flight controller

An overview of the program flow and algorithm of the flight controller can be seen in REF Figure 4 and is described in the following. More details on the implementation can be found in the REF **“Implementation of the Flight Controller”.**

When the XCopter is turned on, the program starts with an initialization progress. At first the, hardware device drivers, such as I²C and UART drivers for communication, the gyroscope, accelerometer and compass drivers, the motor drivers and the PWM driver, get initialized. After that, mutexes and semaphores to synchronize the tasks and protect the critical data sections are set up. Next, the timers, and finally the tasks are created and configured. When the initialization is done, the program gets divided into three tasks. The SensorDataManager and RCReceiver tasks are starting one second after the Main task to ‘warm up’, so they can provide the data that is needed in the Main task.

The SensorDataManager task, which is colored in pink again, is responsible for providing the averaged data from the gyroscope, accelerometer and compass sensors. In a loop, it reads all sensors as fast as possible. Then, when twenty data sets were read, the raw data sets are getting summed up and divided by twenty to get averaged. When the averaging is done, the main task gets notified by setting a flag, which indicates that the new averaged data is available. After notifying, the data counter gets reset and the process starts again.

The task which handles the commands the XCopter receives from the remote control is the RCReceiver task. The main loop of this task reads the incoming SUMD frames that the remote control transmits and CRC-checks them for validation. If the frame is valid, the RC values are scaled and converted into euler angles. When this is done, the RCReceiver task notifies the main task in the same way the SensorDataManager task does this – by setting a flag in the main task.

The major part of the flight controller algorithm is located in the Main task and is also running in an endless loop. At first it is checked, whether new averaged sensor data from the Sensor Data Manager is available. If there is, the main task gets and stores this data. After storing the averaged data, it is handed over to the sensor data filter to filter the data and compute the current pitch, yaw and roll euler angles of the XCopter. After storing the filtered data, it is checked if there are new remote control values available from the RCReceiver task. If there is new data available, it gets also stored. The euler angles from the filter and the remote control are then passed to the PIDs as parameters. The pitch PID gets the pitch angles, the roll PID gets the roll angles and the yaw PID gets the yaw angles. The particular PIDs then calculate an error value for each axis. Those error values are the input parameters for the motor mapper. There, the PWM signals, to run the motors at the requested velocities, are generated with the PID error values. After setting the motor speeds, the loop starts again. If there is no new sensor or remote control data available, the PIDs are fed with older data from a previous iteration. Anything else is just the same, as if there was new data available.

All tasks are running until the XCopter is turned off.

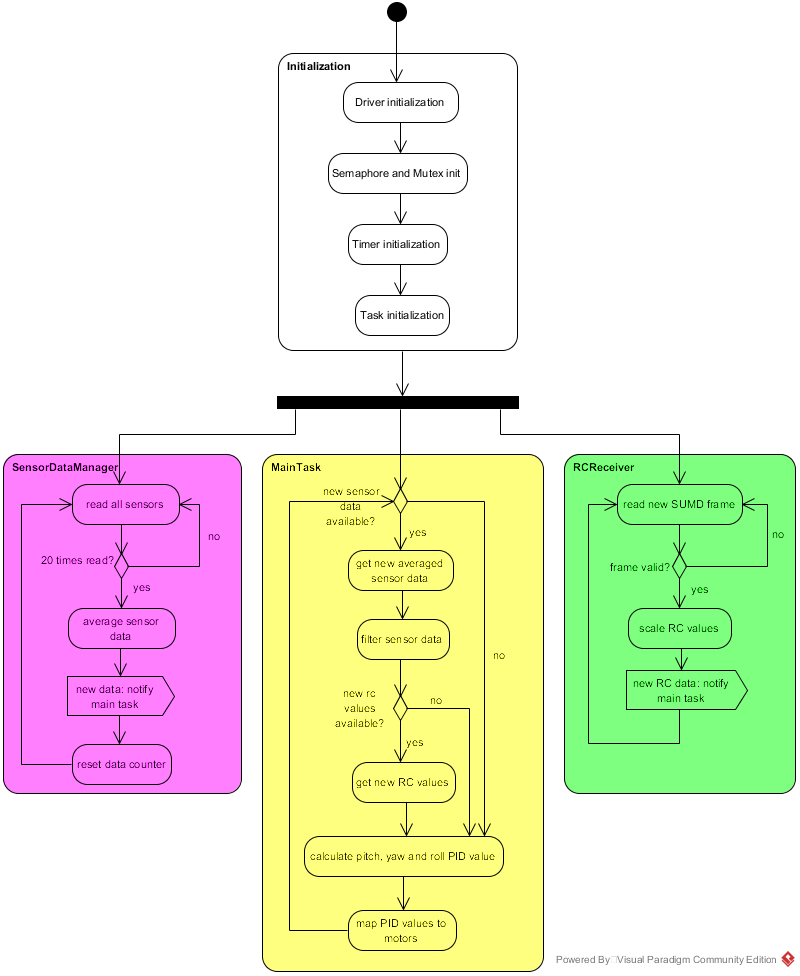


Figure 4: Program flow of the flight controller