UNMANNED AIRCRAFT EXPERIMENTAL SYSTEM: THE FLYING LAB FOR APPLIED FLIGHT CONTROL AND FLIGHT MECHANICS

Janik Hopf, Jakob Dommaschk, Nikolai Block, Richard Reinfeld, Mara Krachten,
Paul Worrmann, Daniel Cracau, Alexander Köthe
AlphaLink Engineering GmbH, Ackerstraße 76/ACK384, 13355 Berlin

Abstract

During flight-mechanical experiments with unmanned aircraft, scientists and students are often faced with the challenge of configuring the aircraft system completely. However, their focus should only be on the testing of new flight control laws or flying qualities, not on setting up or programming flight control computers. Although these are valuable experiences, they cost a lot of time in the development process. With the Unmanned Aircraft Experimental System, a flying laboratory is available, which is completely preconfigured for flight tests. In addition to the physical aircraft, a flight mechanical model and a MATLAB/SIMULINK model for flight control law implementation are part of the system.
This paper presents the capability of the system. An overview of the overall system is provided and possible applications for flight mechanical experiments (system identification) and the implementation of flight controllers are demonstrated.

Nomenclature

p	Roll rate	α	Angle of attack
q	Pitch rate	η	Elevator deflection
r	Yaw rate	η_L	Left rudder at the V-tail deflection
TL	Thrust lever	η_R	Right rudder at the V-tail deflection
t	Time	γ	Flight path angle
V_{IAS}	Indicated airspeed	ξ	Aileron deflection
x	State Variable	ζ	Rudder deflection

1 INTRODUCTION

In recent years, the market for commercial unmanned aircraft has grown enormously. In order to meet the needs of the market, the range of possible applications for such drones (both fixed-wing aircraft and copters) is constantly increasing. Today, drones are used for example for the inspection of ground infrastructure, aerial photography or the transport of goods. To fulfil these tasks, flight control is becoming increasingly important. Drones must be easily controllable by pilots and able to fly trajectories with high precision. For this purpose, the flight mechanics of unmanned aircraft must be understood and flight control laws must be implemented with appropriate approaches.

The training of engineers in flight mechanics and flight control usually takes place only in the classroom. Practical experiments, where flight dynamic models are validated or flight controllers are tested, are rarely used. Most of the time, the associated high workload is the underlying reason. An aircraft, a flight control system, sensors, and partially actuators have to be selected and purchased; moreover, the systems have to be integrated into the aircraft. Then, extensive tests and the development of a flight dynamic model are necessary. Only after a long time, flight tests can be finally conducted. In addition to the time involved, lecturers and students often have to learn complex programming and to develop code by themselves. Especially for aerospace engineers, this can sometimes be a big barrier blocking effective work or research output. Once the system has been developed, staff turnover can further lead to gaps in knowledge that have to be closed later on at great expense.

AlphaLink Engineering GmbH has developed the Unmanned Aircraft Experimental System (UAXS) to address these abovementioned challenges. The UAXS is a system that comes ready to fly. Flight control system, sensors and actuators are already integrated in the aircraft, the ZOHD Nano Talon (see Fig. 1). The aircraft is delivered with i) linear state space models for different flight states and ii) a Simulink template for the integration of controllers. The Simulink model contains all available sensors as inputs and all actuators as outputs. The Direct Law is already implemented in the Simulink template. Lecturers and students can now remain in their domain. They develop the control laws and test them with the linear models. These control laws are then integrated into the Simulink template. Code is generated from this model, which is then copied directly to the flight control system without modifications. Then,

doi:10.25967/530237

the flight tests can be performed. No code development is necessary. The data analysis can also be done directly in MATLAB using the provided MATLAB scripts.

The UAXS does not only open up new possibilities for teachers and students. Scientists also have the opportunity to test new approaches in flight control quickly and at low cost with this system, thus increasing validity and bringing publications to a higher level.

Figure 1: ZOHD Nano Talon.

Table 1: Properties of Talon Nano Evo.

Property	Dimension
Wingspan	0.86 m
Wing Area	0.148 m^2
Aspect Ratio	5.1
Length	0.57 m
Take-Off Mass	0.65 kg
Battery Capacity	1550 mAh
Servomotors	3 × 9 g metal gear
Motor	SunnySky 2204-1870KV
Propeller	6×3 (inch)
ESC	30 A, 5 V 2 A BEC

2 HARDWARE SETUP

The UAXS consists of the aircraft, the flight control system, sensors and actuators, and other peripherals. These components are presented below.

2.1 Aircraft ZOHD Nano Talon

The Talon Nano Evo is a fixed-wing aircraft with V-tail. Table 1 lists the most important aircraft parameters. Three aerodynamic rudders and thrust are available as input variables. Left and right aileron are actuated by a common servo motor. Left and right V-tail rudder have different servo motors and, hence, can be deflected independently. A control allocation is made in consideration of conventional control surfaces and is already implemented in the Direct Law model. It can be changed as required. Using one elevator input (η) and one rudder input (ζ), the following applies to the left (η_L) and right rudder (η_R) at the V-tail:

$$\begin{aligned} \eta_R &= \eta - \zeta \\ \eta_L &= \eta + \zeta \end{aligned} \quad . \tag{1}$$

The aerodynamic rudders are limited within $\pm 25^\circ$.

2.2 Flight Control System

The flight control system consists of the flight control computer, sensors and actuators. The flight control computer is the Pixhawk 4 mini (Pixhawk). The flight control system is already configured and installed in the aircraft. The flight control computer has three RS-232 interfaces, two I2C interfaces, one CAN interface, two analog inputs, one input for RC commands (S.Bus), one USB interface, and 8 PWM outputs. The three RS-232 interfaces are intended for i) the GPS module, ii) the telemetry module and iii) the LiDAR altimeter. An I2C interface is used for measuring dynamic pressure. The CAN interface is required to connect the system to the Hardware-in-the-Loop (HiL) Simulator. Alternatively, the open lightweight protocol UAVCAN can be used with the CAN interface. An analog input is used for the measurement of the battery state of charge. Currently, only RC receivers with S.BUS interface are supported. Suitable transmitters and receivers are included in the set. The USB interface is used to upload software and download log data. Four PWM outputs are required for thrust and aerodynamic surfaces.

Four sensors are already integrated in the Pixhawk. The ICM-20689 and BMI055 sensors measure the acceleration and rotation rates. The magnetic field is measured with the IST8310 sensor and the barometric pressure is measured with the MS5611 sensor. An STM32F765 is used as the microcontroller. In addition, an SD card can be used with the flight control computer to record the flight data.

2.3 Sensors & Peripherals

In addition to the standard on-board sensors of the Pixhawk, a dynamic pressure sensor and GPS are included in the UAXS set. The dynamic pressure sensor is of model MS4525DO. A pitot tube is already built in the airframe and connected to the sensor. The sensor is connected to the Pixhawk via I2C. The maximum measuring range is 6894.76 Pa. The resolution is 0.74 Pa. The accuracy is in the range of ± 17.23 Pa (at 25 °C). The dynamic pressure sensor is pre-calibrated by the original manufacturer. The GPS module is of type UBLOX M8N. It is connected to the Pixhawk via the UART interface. The GPS sensor provides the geographical position with a frequency of 2 Hz. The measurement of the height above ground is conducted with the LiDAR Benewake TFmini Plus Micro. The maximum measurable height is 12 m, the lowest measurable height is 30 cm.

A telemetry modem is used for communication with the ground station (QGroundControl software). It is already installed in the aircraft and connected to the flight control computer. The PX4 software (c.f. Sec. 3) uses the MAVLink protocol for data transmission. The transmission frequency is 433 MHz and the maximum transmission

power is 100 mW. According to the original manufacturer, distances of up to 300 m are possible. A second telemetry modem must be connected to the user's computer via USB (FT230X Basic UART to USB).

3 SOFTWARE CONFIGURATION

There are several already configured software solutions available for the flight control computer Pixhawk 4 mini. For the UAXS, the PX4 flight stack is used. This flight stack is based on the NuttX operating system and allows not only the interaction with sensors and actuators using preconfigured drivers, but also the communication between different program modules, the scheduled execution of individual program modules, and the communication to the external devices. Different modules, for example an extended Kalman filter or autopilots, are already included in the software. Users of the flight stack can develop their own modules in C++ and then integrate them into the software system. For the UAXS, the flight stack was modified so that the end-user does not have to write code anymore, but can create links between sensors and actuators via the modeling language Simulink.

3.1 Flight Stack and Ground Station

In order to enable the integration of software designed in Simulink into the PX4 flight stack, the latter one is modified for the UAXS. The high-level software architecture, consisting of the driver, external connectivity, storage, and flight control modules, has been maintained. Only the flight control module is modified. The existing software for the calculation of flight control laws, consisting of state machine, autonomous flight, position controller, and attitude & rate controller are removed from the flight stack. Instead of these modules, a C++ framework model was integrated and the code generated by Simulink is implemented there. The frame model reads the sensor data from the sensor hub and generates commands for the output drivers, which set the position of the aerodynamic surfaces and the thrust. In addition, the framework module uses the estimated flight attitude (quaternions and Euler angle) of the position & attitude estimator module. This means that the flight attitude is also available as an input in the Simulink model. The framework model works with 100 Hz. The execution time is monitored by a hardware timer.

The advantage of maintaining the core of the flight stack is that very well established software such as QGroundControl can be used with the UAXS. Because the MAVLink protocol is still available in the flight stack, the telemetry module is able to establish communication between the flight control computer and QGroundControl. This allows the visualization of flight parameter during the flight tests, but also the configuration of the system, e.g. the calibration of sensors or the download of log files.

The framework for the Simulink model is stored in the autostart of the flight stack. Whenever the system is started, the flight control laws implemented in the Simulink model are executed immediately.

3.2 Simulink Model

The development of flight control laws for the UAXS follows the model-based approach in Simulink. Figure 15 shows the template for the development (last page of this paper). After the blocks are integrated in the Simulink model and connected to the inputs and outputs, source code is generated by the embedded coder. With the predefined settings in the Simulink model, the generated code can be copied directly into a directory of the flight stack. The entire flight stack has to be compiled with the supplied software (one command) and finally uploaded to the flight control computer (another command). After rebooting the system, the designed control law can be used directly.

The Simulink model has 46 inputs. These are 9 inputs belonging to the IMU (three accelerations, three rotation rates, and three magnetic flux densities), one input from LiDAR (height above ground), three inputs from the static pressure sensor (static pressure, temperature, air density), four inputs from the dynamic pressure sensor (dynamic pressure, indicated airspeed, true airspeed, temperature), and 8 inputs from GPS (longitude, latitude, altitude above sea level, altitude above the ellipsoid, orbit speed in all three axes, course), 9 inputs from the extended Kalman filter (four quaternions, three Euler angles, wind speed in north and east direction), 10 inputs from the RC receiver (four axes, four switches, fail-safe mode, signal strength) and two inputs from the battery (discharge and percentage of remaining battery). The Simulink Model has 24 outputs. Four of these outputs are for aerodynamic surfaces and thrust, and the other 20 outputs are custom logs. These custom logs can be used to store values calculated within the model and evaluate them in post processing. In the Simulink model, the signals are consistently assigned their unit and data type. Signals that represent a floating point number have an `r` as prefix, integers have the prefix `i`, unsigned integers have the prefix `u`, and boolean values have the prefix `b`. The prefix is followed by the memory size of the signal in bytes. In the Simulink template, the direct law is already implemented. It converts the input from the RC receiver into commands for the aerodynamic surfaces (especially the conversion at the V-tail) and thrust.

In principle, it is possible to use all Simulink toolboxes, especially StateFlow. Nevertheless, some development guidelines have to be respected. It is not possible to work with continuous blocks. Therefore, a discrete description has to be used. Custom created MATLAB blocks are supported for code generation. If MATLAB functions are used, a maximum number of 60 lines shall not be exceeded. Due to rounding errors, most floating-point numbers end up being slightly imprecise. This leads to equality tests failing and therefore floating point comparison for equality/inequality must be avoided. Algebraic loops are problematic for code generation. If the algebraic loop cannot be avoided, a `Delay` block shall be used to break up the loop. It is not possible to use the `If` blocks without `else` condition. By using the ModelAdvisor, design errors can be detected early in the development process.

3.3 Flight Data Analysis

The Pixhawk stores data from the flight tests in ULog file format. These files can be uploaded to a website and used to visualize the flight test. For scientists and students, however, further processing of the data in MATLAB is essential. Therefore, a MATLAB script was developed that reads the ULog files into MATLAB and converts them into a MAT file. Using a synchronization script, the time stamps of different sensor topics are synchronized to a common time. This allows an easy evaluation and analysis of flight tests.

3.4 Flight Dynamic Models

The UAXS is delivered with four linear state spaces for the design of flight controllers. In these models, the state variables pitch rate, angle of attack, indicated airspeed, pitch angle, yaw rate, sideslip angle, roll rate, and bank angle are used. In addition, flight path angle, ground speed and vertical acceleration are available as output variables. As control inputs thrust, elevator, ailerons, and rudder are used. The wind speeds in the geodetic system in all three axes and the wind induced rotation rates about all three axes can be used as disturbance variables. The flight conditions are listed in Tab. 2. For each flight state, state spaces are provided for i) longitudinal motion, ii) lateral motion, and iii) longitudinal and lateral motion.

Table 2: Available Trim Points.

Nr	V_{IAS}	γ	α	TL	η
1	$17\,\mathrm{m\,s^{-1}}$	$0°$	$1.68°$	0.91	$-7.87°$
2	$12\,\mathrm{m\,s^{-1}}$	$0°$	$5.83°$	0.66	$-17.01°$
3	$12\,\mathrm{m\,s^{-1}}$	$10°$	$5.56°$	0.77	$-16.45°$
4	$12\,\mathrm{m\,s^{-1}}$	$-10°$	$5.86°$	0.52	$-17.11°$

4 FURTHER APPLICATIONS

Testing the controllers is essential for a successful flight test. Ideally, this is done with the entire system on the ground. A hardware-in-the-loop simulator is available for this purpose. In order to get direct impressions for the flight test, a digital twin of the aircraft has been created, which can be tested in the so-called Virtual Flight Test Environment.

4.1 Hardware-in-the-Loop Simulator

The hardware-in-the-loop simulator can test the entire system in a nonlinear simulation. For this purpose, the sensors and actuators of the aircraft were identified and their dynamic behavior was integrated into the nonlinear simulation of the flight dynamics. The nonlinear Simulink model is coupled with CANoe software from *Vector Informatik* [4]. The CANoe software establishes a CAN connection to the flight control computer. For this purpose, the flight stack was further modified, since this type of CAN communication is not provided as standard. The CAN interface is used to send sensor values from the Simulink model to the flight control computer. The flight stack has been modified to use CAN messages instead of sensor values. The emulated sensor values are used to calculate the flight control laws, deflect the control surfaces accordingly, and transmit the deflections back to the Simulink model via the CAN interface. With this setup, it is possible to use the ground station (e.g. QGroundControl) during the test. Using a UDP connection, the behavior of the aircraft can be directly visualized in the Virtual Flight Test Environment. This makes it possible to perform the planned flight test in a complete offline test and derive improvements for the controller directly from the hardware-in-the-loop tests. Figure 2 shows the flight test vehicle that is connected to the CANoe software via CAN interface and whose flight attitude is displayed in the Virtual Flight Test Environment.

Figure 2: Hardware-in-the-Loop Simulator for ZOHD Nano Talon.

For users without access to hardware and software from Vector Informatik, an alternative hardware can be provided. This hardware establishes the CAN connection to the Pixhawk and communicates with the Simulink model via a UDP interface.

4.2 Virtual Flight Test Environment

For the visualization of manned aircraft there are many visual software solutions available that can be used for the three-dimensional visualization of the aircraft. The flight simulation programs *X-Plane* and *FlightGear*, for example, allow the connection to a Simulink model via a UDP interface. However, both the Simulink model and the software for visualization have to be installed at the end user's site.

To overcome these vulnerabilities, the Virtual Flight Test Environment is a web-based solution implemented with HTML, CSS and JavaScript. It can be executed on any end device that has a web browser. The nonlinear model of the Talon's flight dynamics was created in Simulink and then converted into C++ code using the embedded coder. Afterwards, this code is converted into JavaScript code using Emscripten. The visualization itself is done using Three.js - a framework for WebGL. The geometric model of the talon, the propeller and the control surfaces as well as the landscape (the Magdeburg Water Bridge) were created using *Blender* and imported as objects in Three.js. The nonlinear model can be controlled via the keyboard. Via a Graphical User Interface, it is currently possible to set the control coefficients for pitch, roll and yaw damper. In the future, entire controllers will

be designed in a web-based development and then tested immediately. If the control results are satisfactory, source code is generated directly from the online modeled controller, which can be copied into the flight stack. This will make the Virtual Flight Test Environment independent of both Simulink and the user's end device. Figure 3 shows a scene from the Virtual Flight Test Environment. Besides the flight dynamics, the most important flight parameters are visualized by instruments.

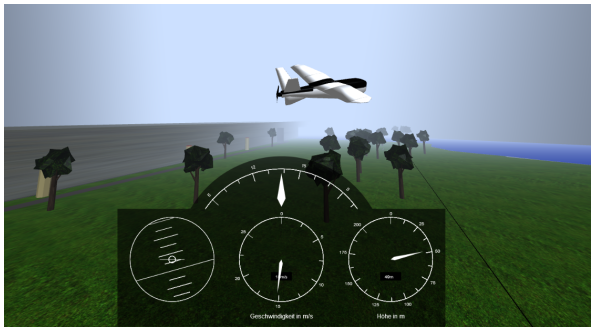

Figure 3: Illustration of the Web-Based Virtual Flight Test Environment.

The Virtual Flight Test Environment can also be used in combination with the hardware-in-the-loop simulator. For this purpose, a web server is set up with NODE.JS. This web server receives the data of the Simulink model of the nonlinear simulation, which communicates with the flight control computer via the CAN interface, through a UDP interface and replaces the module of the nonlinear simulation in the Virtual Flight Test Environment. This allows to visualize not only the behavior of the aircraft, but at the same time the movement of the control surfaces and the behavior of the entire flight control system on the aircraft.

5 FLIGHT TEST APPLICATION

Two application examples will be used to describe the use if the system. Here, i) an identification of the short-period mode and phugoid is performed and ii) a stability augmentation system with the UAXS is used.

5.1 Identification

To verify the flight dynamics in longitudinal motion, the short-period mode and phugoid are excited by an elevator double in the elevator [3]. The doublet is defined with

$$\eta_{\text{doublet}} = \begin{cases} +6.5^\circ & 0 \le t_{\text{rel}} < 0.5\,\text{s} \\ -6.5^\circ & 0.5\,\text{s} \le t_{\text{rel}} < 1\,\text{s} \\ 0 & \text{else} \end{cases} \tag{2}$$

with t_{rel} as relative time after starting the excitation. The total elevator deflection η results as suponation from the signal for the double η_{doublet} and the pilot input η_{pilot} with

$$\eta = \eta_{\text{pilot}} + \eta_{\text{doublet}}. \tag{3}$$

To implement the signal generator, two inputs of the Simulink model of Fig. 15 are required: one is a push button on the remote control (if pressed, state is 1; if released, state is 0) and the other is the pilot elevator command signal. The pilot signal can be taken from the Direct Law Block, which is already implemented in the model.

5.1.1 Implementation

The signal generator is implemented as a state machine, which has the push button of the remote controller as input and the doublet signal for the elevator as output. The state machine has the four states i) `Idle` ii) `Pos_Deflect` (positive deflection), iii) `Neg_Deflect` (negative deflection), and iv) `Stop`. As long as the remote control signal is at zero, the state machine remains in `Idle` state. Only when the signal on the remote control changes, the state machine changes to `Pos_Deflect`. In this state, a positive elevator deflection of 6.5° is commanded. A local variable is used to count up to 50. This corresponds to half a second, as the model runs at 100 Hz. After the value of 50 is reached, the system changes to the state `Neg_Deflect`. Here, the elevator deflection is -6.5°. The local variable is counted up to 100. After reaching this value, the system changes to the `Stop` state, in which a value of 0° elevator is commanded. The described behavior is shown in Fig. 4.

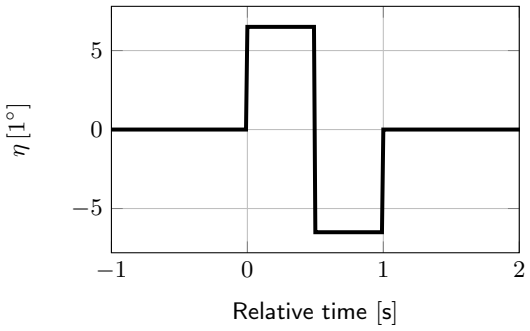

Figure 4: Commanded Elevator Deflection.

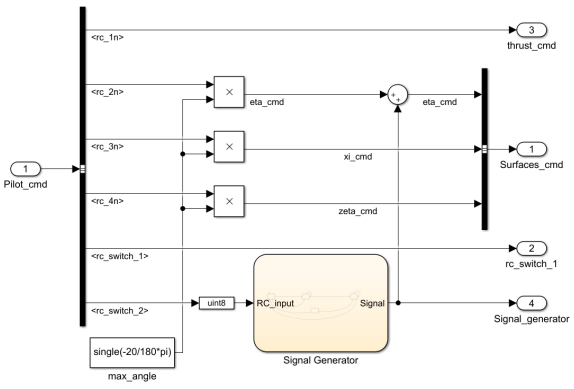

Figure 5: Integration of the Signal Generator (as a State Machine) into the existing DirectLaw Module of the Simulink Model.

When the push button on the remote control is released, the state machine immediately changes from any state

to the `Idle` state. On transition to this state, the commanded elevator is set to 0° and the local variable is set back to 0.

Figure 5 shows the implementation of the state machine into the existing Simulink model. Figure 6 shows the state machine itself. The output of the signal generator can also be saved as a custom log.

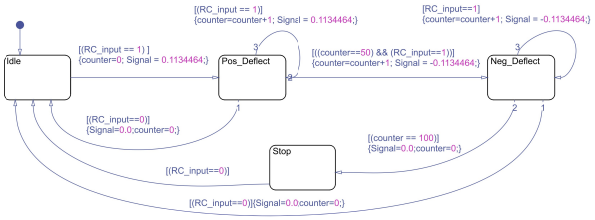

Figure 6: Realization of the Signal Generator as a State Machine.

5.1.2 Flight Test Results

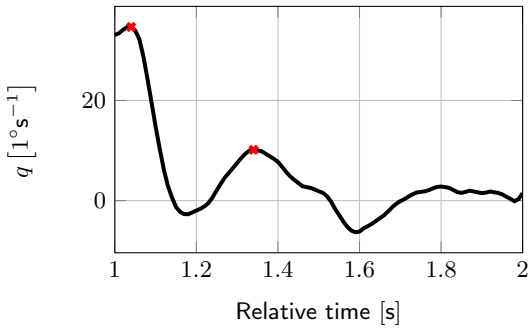

Figure 7: Flight Data for Identification of Short-Period Mode.

The excitation was performed in flight test. Figure 7 shows the output in the pitch rate, which allows a very good identification of the angle of attack oscillation. The excitation is completed after 1 s and the aircraft responds with its eigendynamics. Two negative amplitudes can be recognized, which follow each other in a time interval of 0.3 s. Thus, the frequency of the angle of attack oscillation is 3.33 Hz. The damping can be determined via the logarithmic decrement. This results in a damping ratio of 0.22.

With the UAXS it is therefore possible to perform simple system identifications. Using the *System Identification Toolbox* from MATLAB, it is also possible to determine more complex connections with higher-order procedures.

5.2 Stability Augmentation System

To improve flying and handling qualities, a stability augmentation system is used [5]. It consists of four basic parts. In the longitudinal motion, the damping of the short-period mode can be improved by feeding back the pitch rate to the elevator [1] as shown in Fig. 8. Regarding the lateral motion, the roll and dutch-roll mode can be affected by feeding back the roll rate to the ailerons (Fig. 8) and the yaw rate to the rudder [1]. As part of the yaw damper, the yaw-rate is primarily filtered with a first order washout filter (Fig. 9). The gains are calculated for one reference state using MATLAB. Afterwards, the controller structure is integrated in the Virtual Flight Test Environment to validate its functionality within a nonlinear simulation. The tested controllers are then brought onto the UAXS for real world testing.

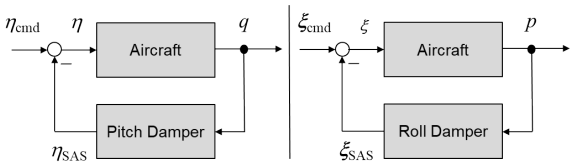

Figure 8: Block Diagram of a Pitch and Roll Damper.

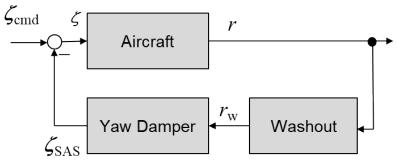

Figure 9: Block Diagram of the Yaw Damper.

5.2.1 Gain Calculation

In order to design the different controllers, the linear state-space for one reference state is used. In the specific situation, the aircraft is trimmed at an airspeed of 17 m/s and a path angle of 0°. To calculate the gains of the pitch damper, the short-period approximation considering the pitch rate and angle of attack is used. Figure 10 shows the root locus of the system for a feedback of the pitch-rate on the elevators. To achieve a damping of 0.7 in the short-period mode, a gain of $K_{\eta \mathrm{q}} = -0.13$ is used.

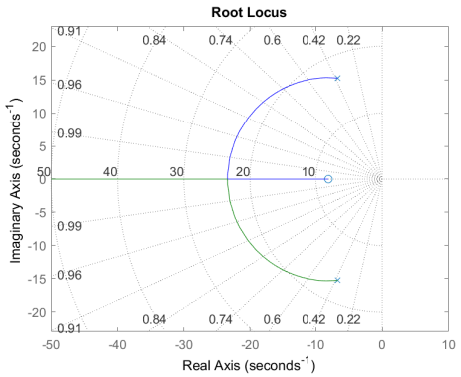

Figure 10: Root Locus of Short-Period Mode to Calculate $K_{\eta \mathrm{q}}$.

The UAXS bears a challenge as it is very agile. To improve its controllability, a roll damper is required. The roll time shall be decreased to 0.05 s. Thus, a negative gain is used. It is calculated using the root locus method as $K_{\xi \mathrm{p}} = -0.02$. The roll damper is then integrated in the

lateral system in order to calculate the gain of the yaw damper and to design the washout filter.

The implementation of a yaw damper requires a washout filter for only using the high frequency parts of the yaw rate. Otherwise, the controller would work against the constant yaw rate as an integral element of performing turns. The structure used is shown in Fig. 9. The time constant of the washout filter $F_{\text{washout}} = \frac{T_\text{w} s}{T_\text{w} s + 1}$ is chosen as $T_\text{w} = 2.5$ s. In order to integrate the washout filter as part of the discrete-time controller [2], its transfer function is described as a state-space

$$\dot{x} = -\frac{1}{T_\text{w}} x + r \tag{4}$$

$$r_\text{w} = -\frac{1}{T_\text{w}} x + r \ . \tag{5}$$

Thus, it can be integrated in SIMULINK using a discrete-time integrator block. Finally, the gain of the yaw damper is determined using the root locus method. A gain of $K_{\zeta_\text{r}} = -0.18$ is required to achieve a damping of 0.7 in the dutch-roll mode.

5.2.2 Integration in Virtual Environment

The Virtual Flight Test Environment is used to test the designed controller. Therefore, the structure as shown in Fig. 8 and 9 is integrated in the non-linear simulation. The value of gains and time constant are declared as inputs to the simulation. Hence, they can be changed dynamically while flying in the Virtual Flight Test Environment. This is shown in Fig. 11.

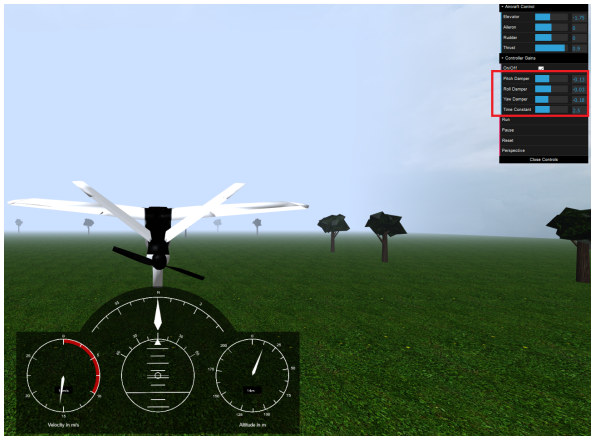

Figure 11: Panel for adjusting the controller gains in the VFTE.

5.3 Implementation

To test the designed controller in a real world application, it is integrated in a SIMULINK model, which defines the connections from sensor data and remote control inputs to the control surfaces. The used structure is shown in Fig. 12.

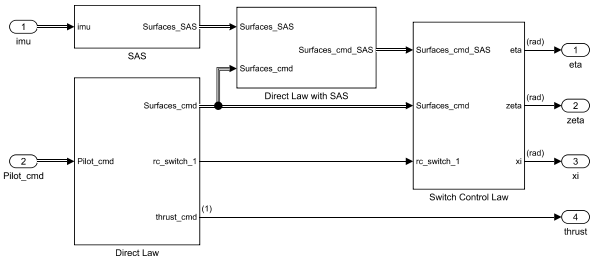

Figure 12: Block Diagram of the Yaw Damper.

The DirectLaw module is the same as shown before (Fig. 5). The controller is integrated in the SAS module as shown in Fig. 13. Afterwards, the controller commands are subtracted from the pilot commands in the DirectLaw with SAS module. The SwitchControlLaw module finally controls which signals feed through to the control surfaces. The remote control allows to toggle the controller at each point in time.

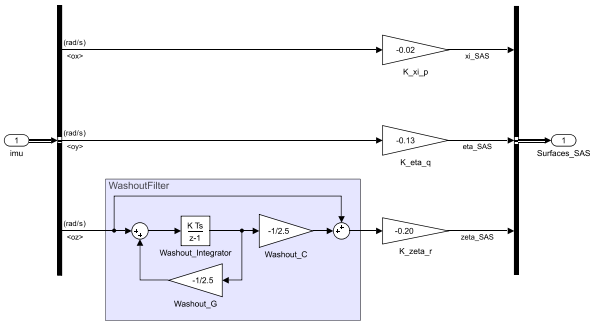

Figure 13: Implementation of the Gains and Filter in the SAS module.

5.4 Flight Test with Controller

The signal generator used for the identification of the short-period mode is used again to test the behaviour of the controlled aircraft. Figure 14 shows the pitch rate after an elevator doublet commanded by the signal generator. The effect of the controller can be clearly seen. Damping as well as frequency are increased compared to the uncontrolled aircraft.

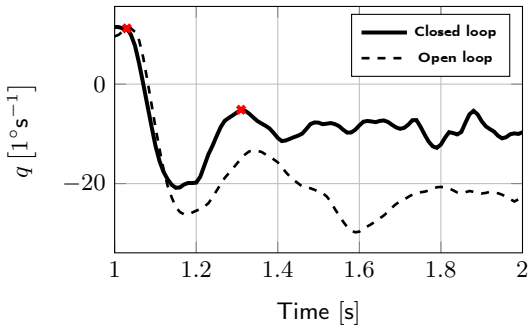

Figure 14: Flight Data of the Controlled Aircraft after an Elevator Doublet.

6 CONCLUSION & OUTLOOK

This paper presented an experimental set based on unmanned aerial vehicles for use in teaching and research. It described the basic structure of the hardware and software and discussed in detail the potential that such a flying lab offers to students, lecturers and researchers. The UAXS does not only contain a physical aircraft. As a digital twin in the Virtual Flight Test Environment, it contributes to the successful preparation of flight tests. Thanks to the CAN interface and the connection to the hardware and software of Vector Informatik, experiments can be carried out directly with the flight test vehicle. This gives engineers a better feeling whether control factors are too high or too low.

Two test cases were used to demonstrate the possibilities offered by the UAXS with regard to flight mechanics and flight control. The short-period mode could be identified from the measured data and the damping could be improved by using a pitch damper. Not only the pure measurement data, but also the subjective evaluation of the pilot showed an improvement of the flying characteristics. When critically analyzing the data, it has to be taken into account that in flight tests the airspeeds differ from the designed airspeeds of the controller. Therefore, the increase in damping is relatively small in the results.

The UAXS as a flying lab offers the possibility to implement any kind of controller. In flight tests, controllers for controlling the attitude have already been successfully tested. This makes it possible to successfully test new control concepts, e.g. for the control of aircraft swarms, by means of cost-effective experiments using the UAXS instaed of preparing otherwise expensive experiments with manned aircraft.

REFERENCES

[1] Brockhaus, R.; Alles, W.; Luckner, R.: *Flugregelung*. Springer Berlin Heidelberg, 2011.

[2] Graf, K.; Schulz, G.: *Regelungstechnik 2: Mehrgrößenregelung, Digitale Regelungstechnik, Fuzzy-Regelung*. De Gruyter, 2013.

[3] Mehra, R.: *Optimal inputs for linear system identification*: IEEE Transactions on Automatic Control Nr. 3, IEEE, S.: 192–200, 1974.

[4] N., N.: *Handbuch CANoe*. Vector Informatik GmbH, 2018.

[5] Stevens, B. L.; Lewis, F. L.: *Aircraft simulation and control*. John Wiley& Sons, New Jersey, USA, 2015.

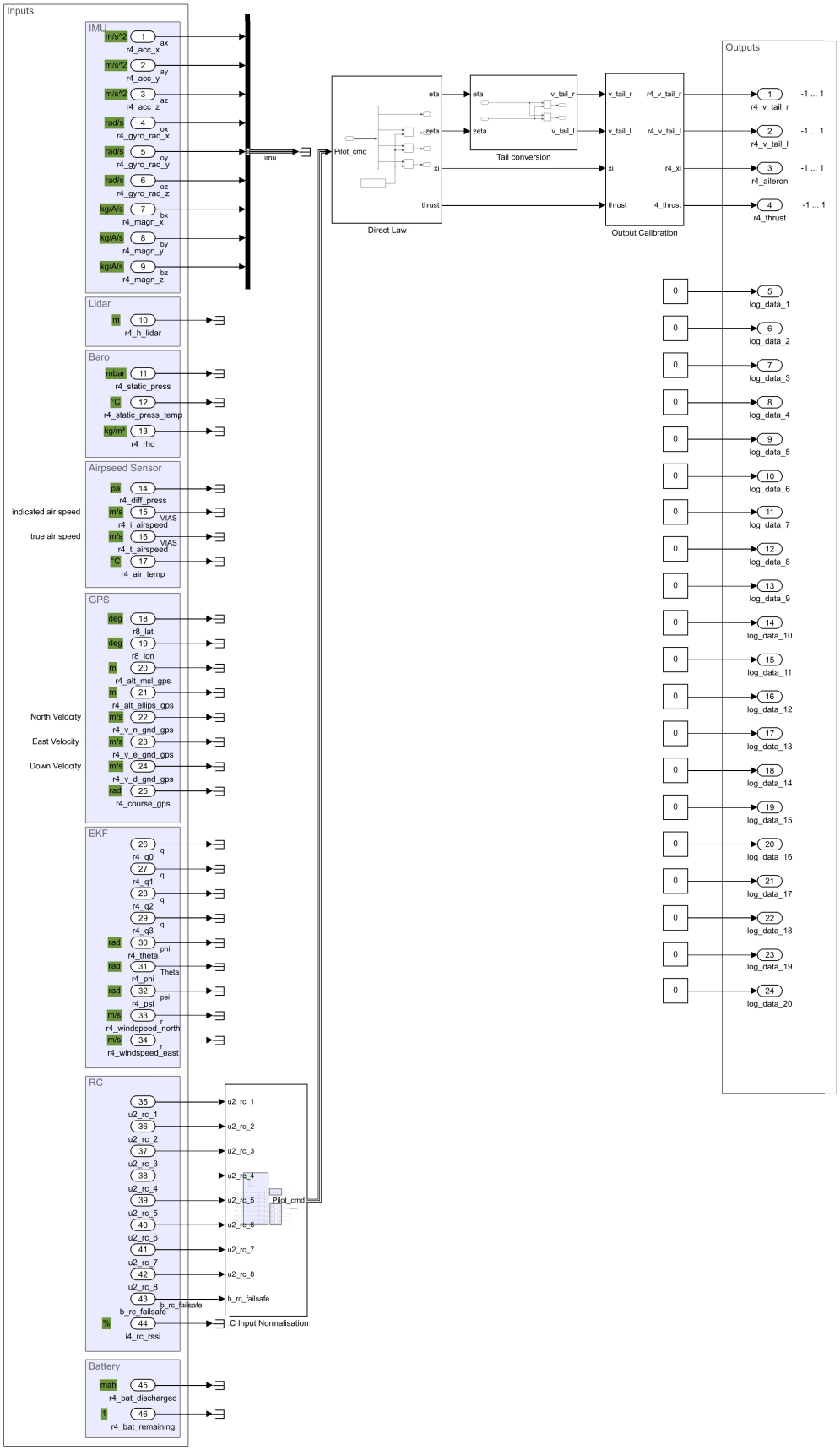

Figure 15: Simulink Template.