DOI: 10.13009/EUCASS2019-772

8TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)
Conference on "Reusable Systems for Space Access". July 1–4, 2019, Madrid, Spain.

Auto-Coded Flight Software for the GNC VTVL Demonstrator EAGLE

Michael Dumke[1*†], *Stephan Theil*[2†],

[†] *German Aerospace Center (DLR), Institute of Space Systems*
Robert-Hooke-Str. 7, DE–28359 Bremen, Germany

[*] Corresponding author

Abstract

The paper will present the simulation and on-board software design for the Vertical Take-Off, Vertical Landing (VTVL) demonstrator EAGLE (Environment for Autonomous GNC Landing Experiments) developed by the Department of Guidance, Navigation and Control Systems at the DLR Institute of Space Systems. The simulation and on-board software primarily relies on auto-coding, easing the effort of the iteration between simulation, preliminary hardware tests, and real flight software. EAGLE demonstrates Guidance, Navigation, and Control (GNC) functionality for planetary landers and reusable launch vehicles. The implemented design approach allows to successively implement or enhance algorithms through simulation (SiL, PiL, and HiL), quickly reaching a level of confidence for a real test flight.

After the first tethered flight tests, this development cycle was tested to the full extend to implement a Mission and Vehicle Management system, that was required for automated lift-off and landing. The next feature to be added to the demonstrator is the capability to rely on a GNSS receiver to fly safely in a larger envelope. The satellite-based navigation is typically aided with correction data for precise positioning, but shall also work in absence of aiding information and under sub-optimal conditions. The full Hardware-in-the-Loop setup comprises the on-board computer, a dSPACE real-time system, as well as the Ground Station as part of the Electronic Ground Support Equipment (EGSE). This provides the opportunity to safely train the operations of EAGLE for developing and testing procedures for ultimately demonstrating free (untethered) flight.

1. Introduction

Developing a new complex system requires substantial testing. Not only for safety but also because it is too complex to know beforehand how and if the planned system will work. In the aerospace domain, this process generally begins in simulation. For the design of EAGLE a concept for development was selected of always keeping real tests in the focus, to react early if some idea is not as easily implementable as expected, or in the rare case that its easier to handle than expected, requiring less test effort.

Already during early stages of the project sensors and actuators were put into operation. For example the full navigation system for the vehicle was integrated into a casing to test the performance even before the main structure was designed. To reduce the effort to switch between simulation and real tests an early decision was taken to base the development strategy on automatic code generation. Extensive experience within this domain was available. The *Mathworks* toolchain (MATLAB/Simulink/Coder) was already in use for embedded systems in different laboratories[3][4], and a *dSPACE* real-time simulation system, based on the same toolchain, was utilized on a regular basis in the department. In most cases the *MATLAB Coder*, and the *Simulink Coder* were utilized for generating code from MATLAB source code, and Simulink models. The *Embedded Coder* that allows to optimize the generated code is not being used for the On-board software (OBS) of EAGLE.

From the beginning all hardware tests and software tests were conducted using a Simulink model. This means, that if a new sensor was available the data interface was tested and verified by auto-coding and cross-compiling a Simulink model to a target platform, either based on the the operation system *QNX* or based on *dSPACE* systems. The interface block of the new sensor was placed in a library to be immediately used, e. g., to feed real measurements into the

[1]E-Mail: michael.dumke@dlr.de, ORCID: 0000-0002-5587-7884
[2]E-Mail: stephan.theil@dlr.de, ORCID: 0000-0002-5346-8091
[3]`https://gnc.dlr.de/s/labs/FACE`
[4]`https://gnc.dlr.de/s/labs/TEAMS`

DOI: 10.13009/EUCASS2019-772

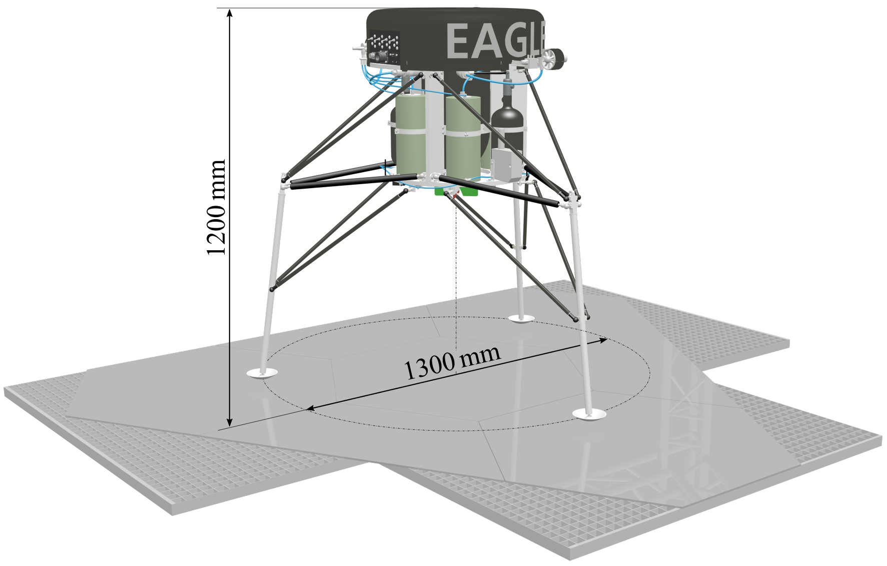


Figure 1: Main Dimensions of EAGLE

navigation filter under development. The same was done for actuator interfaces with the benefit that, e. g., the identical jet engine interface is available for dSPACE (necessary for the engine identification) and QNX (for the flight software of the On-board computer (OBC)). All software functions, like navigation filter, controllers, and guidance functions are put into a library that can be used without modification for simulation, real-time tests, and flight demonstrations.

The defined development environment showed benefits almost from the beginning. Only minor additional work was necessary to bring new hardware or software into the (real-time) Simulink environment, but when further advancing in the project the prior work could be reused without additional steps to take. For further supporting this approach a parameter system was implemented for configuration management that allows to run the same navigation filter on different systems, like a standalone navigation system (NavSys), or EAGLE itself that differs in sensor alignment or sensor calibration.

The process of developing software for the EAGLE project is described in detail within this paper. The VTVL vehicle, for which the OBS is developed, is introduced in Section 2. The OBS architecture itself is presented in Section 3 and Section 4 introduces the available test hardware and how it is used during the development process. Two examples are selected to show in Section 5 the benefits of developing software for EAGLE with the given tools, at the end concluding with an outlook on the GNC demonstrator's future.

2. VTVL Demonstrator EAGLE

EAGLE is an experimental VTVL vehicle developed as a demonstrator for GNC technologies. A rendering with the main dimensions is given in Figure 1.

Its intent is to demonstrate and verify Guidance, Navigation, and Control technologies for exploration vehicles which shall land on the Moon, planets and other celestial bodies such as asteroids. These types of mission often require a main engine for soft landing.

In addition, EAGLE is also suitable for researching navigation and control systems for space transportation. On the one hand, a vertical take-off is a fundamental characteristic of launch vehicles; on the other hand, research in the area of re-usable launch vehicles is currently a global topic for future launchers. One possible option for re-usability is to have the first stages of a launcher return to ground in a controlled manner, using their main thrusters for powered descent and landing. Disturbing effects like structural oscillations or sloshing of fuel and their effect on the GNC system can then be researched using EAGLE.

Contrarily to launchers and planetary landers, EAGLE is not powered by a rocket engine but by a small jet engine, which makes its operation safer and far less costly. Additionally, the vehicle is controlled with a thrust vectoring unit and a cold gas thruster system. For autonomous flight EAGLE is equipped with a set of navigation sensors combined using an advanced filtering algorithm.

The on-board avionics is shown in Figure 2. On the left side the available set of sensors is listed that is directly connected to the OBC, or relayed via an interface board. The actuator interfaces for the jet engine, Thrust Vector Control (TVC) system, and the cold gas Reaction Control System (RCS) are shown on the right side. For Telemetry/T-

DOI: 10.13009/EUCASS2019-772

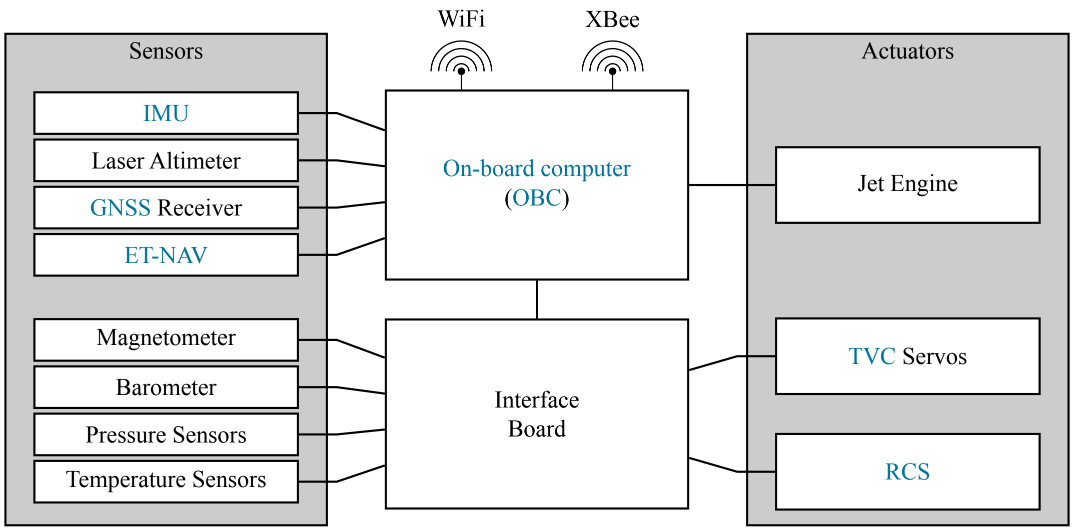


Figure 2: Avionics Overview of EAGLE

elecommand (TM/TC) operations two separate links are available as a reliable communication infrastructure. Further information regarding the vehicle can be found in the user manual[5] and in [1, 2, 3, 4].

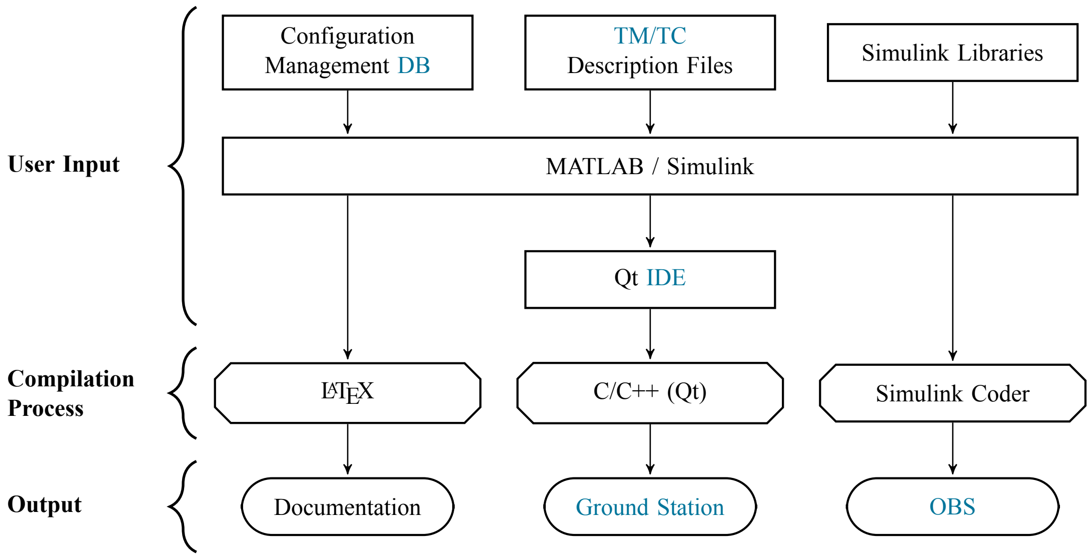


Figure 3: Software Tool Chain

3. Tool Chain and Software Architecture

The tool chain used within and partly developed for the EAGLE project is depicted in Figure 3. The developer can update or add functionality as input to the tool chain. The inputs can be provided to five different points that are marked as *User Input*. All parameters are put into a *Configuration Management Database (DB)* that can later be used within a Simulink model. Telecommunication data frames can be added or modified that generally are grouped by topics within individual Comma Separated Values (CSV) *Description Files*. Software parts that can be reused are given as a Simulink library as an input to the tool chain.

The full OBS of EAGLE, as the software for Model-in-the-Loop (MiL)/Hardware-in-the-Loop (HiL) simulations, is a complex Simulink model. A developer can modify these models to include and test new functionality. Automatically, a specific configuration for the intended use case is loaded. For example selecting the correct parameters for the EAGLE

[5] `https://gnc.dlr.de/s/projects/eagle/user-manual`

DOI: 10.13009/EUCASS2019-772

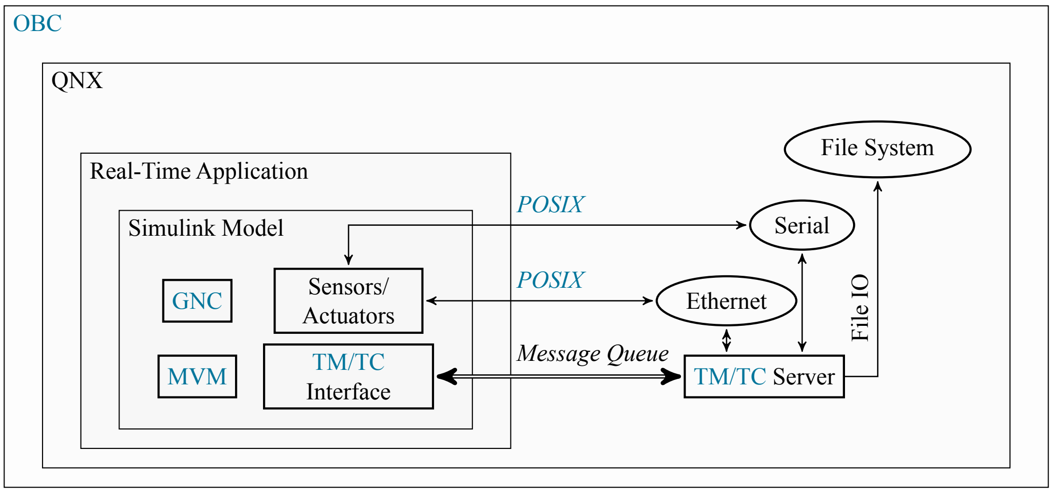


Figure 4: Software Architecture of EAGLE

lander, or for the NavSys, comprising a different set of sensors and calibration parameters. Additionally, the description files are parsed to provide an interface to the TM/TC communication directly within Simulink. At the same time the Ground Station source code is updated to include all modified data frames.

If in the last step a new or modified TM/TC frame was generated, it will be automatically decoded or encoded by the Ground Station software. For proper visualization of these information a developer might decide to put new Graphical User Interface (GUI) elements into the software that was developed with the Qt Integrated Development Environment (IDE). This concludes the inputs for which a developer can provide changes to the software.

After all the necessary input was given to the tool chain the compilation process from Figure 3 is started. The output is the documentation of the parameter set and the TM/TC data frames. During MiL simulations the model is directly executed within Simulink and the Simulink Coder is not necessary. If the model is intended to run in real-time two binaries are produced for the execution on the target platforms. The Ground Station binary is typically launched on a laptop as shown in Section 4.1.3. Two possible target architectures exist for the real-time OBS binary that is generated by the Simulink Coder. It depends on the Simulink model that was modified. Either it is for the execution on an OBC (EAGLE or the navigation system mock-up (NavSys)), or it is a binary file that executes on a dSPACE platform for a real-time HiL simulation. In both cases the process is identical and transparent for the developer.

When running the model in real-time on an OBC the Operating System (OS) *QNX* is used. Figure 4 shows the software architecture of the OBS that is necessary for a controlled flight of EAGLE. A special compilation target for the Simulink Coder handles the thread creation and the thread execution during runtime. This binary is the *Real-Time Application* that includes the converted *Simulink Model* and runs within QNX on the OBC.

The model philosophy is based on the usage of Simulink libraries, increasing the possibility to reuse by sharing the exact same functionality between the different real-time target platforms and simulations (Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), HiL).

The root level of the OBS Simulink model includes library blocks of systems that can be executed on any target platform, these blocks are:

- Mission and Vehicle Management (MVM)
- Guidance
- Navigation
- Control

Within these library functions further levels of libraries do exists, e. g. different guidance schemes, controllers, etc. Other high level blocks are explicitly chosen not to be within a library. These blocks are:

- TM/TC Interface
- Sensors
- Actuators

DOI: 10.13009/EUCASS2019-772

The reason for this is that different models for different use cases exist. For example the model running on the navigation system mock-up does not include actuators, and on the other hand it includes a different set of sensors, e. g., a different Inertial Measurement Unit (IMU) device.

The interfaces between these high level blocks are predefined *busses* which are shared across all models. The definition of these are kept as general as possible to allow to exchange devices, e. g., the bus requires an IMU with its typical measurements. The same bus shall be used no matter what specific IMU is integrated within the system, or which Global Navigation Satellite System (GNSS) receiver is utilized.

All communication to the sensors and actuators use the Portable Operating System Interface (POSIX) interface provided by QNX to access serial ports and Ethernet interfaces. A message queue handles the data exchange between an additional application, the *TM/TC Server*, and the real-time application. Due to the hard disk access of this application real-time violations cannot be precluded which is the reason why it is executed outside of the main application. The TM/TC server acts as relay between the Ground Station and the real-time application, it receives the telecommands from two different sources, WiFi (*Ethernet*) and XBee (*Serial*), and passes them to the model. Telemetry from the model is send to the Ground Station, also over two different wireless links. All data passed through the server is immediately written as log files to the hard disk for later post-processing.

3.1 Configuration Management

For system-wide parameters a configuration management system was designed. This allows to have the same navigation filter model running on different systems, that differ, e. g., in sensor calibration and alignment. The configuration management defines a parameter not only as a name and a value, but enforces to provide a storage type, a description, a symbol, and, if necessary, a C/C++ variable name that is available within MATLAB s-functions. MVM modes, available controllers, and guidance functions are defined by *enumerations* within the configuration set as the single authoritative source. Automatically generated reports are added to the EAGLE project documentation to keep it updated with, e. g., the current Mass, Center of Mass, Inertia (MCI) parameters.

With this configuration management system the main EAGLE flight software model is able to execute on different computers, either the real flight computer on EAGLE, or a spare computer connected to a dSPACE system for a full HiL demonstration, without having to modify the Simulink model.

3.2 Communication Beyond the System's Boundary

Protocol handlers interfacing sensors and actuators are generally programmed in C/C++ for decoding and encoding the byte stream exchanged with the devices over serial lines or Ethernet. The TM/TC communication is handled transparently for the user, even if new or different information needs to be displayed on the Ground Station display or needs to be logged for post-processing. A simple CSV file format is used to define TM/TC frames. An automated process parses these files, generates the C-code for decoding and encoding the byte stream, builds a Simulink library with a block for each packet, adds the new packets to the Ground Station software and log-file conversion program, and finally generates documentation for a full overview of the available TM/TC data. Also transparently to the user the communication between EAGLE and the Ground Station can be sent over two different links (WiFi and Xbee) with configurable transmitting frequencies.

4. Development Process

The testing and validation of software from the first tests within Simulink on a personal computer to a software running in real-time on the OBC can be done in a very efficient way. Depending on the software functionality to be tested, multiple test environments are available. Not necessarily all have do be used to prove flight readiness, at least not for the current tethered flight facility. For later free flights a clear process should state what and how a functionality should be tested before it is allowed to be used within the flight software on EAGLE.

A general depiction of such a process is given in Figure 5. It shows the succession of the typical steps of a simulation based design approach (MiL/SiL, PiL, HiL, and demonstration). The demonstration step is split into two in order to differentiate between *Tethered Flight* and *Free Flight*. The development path towards a free flight (not yet conducted and therefore drawn with dashed lines) should ultimately pass a tethered flight test before. Each development step after MiL/SiL closes the loop by providing feedback to the design, either by showing that it is not working or that a functionality is missing, or by showing that the reality deviates from the simulation results. These cycles help that the simulation converges to a close representation of the real system.

DOI: 10.13009/EUCASS2019-772

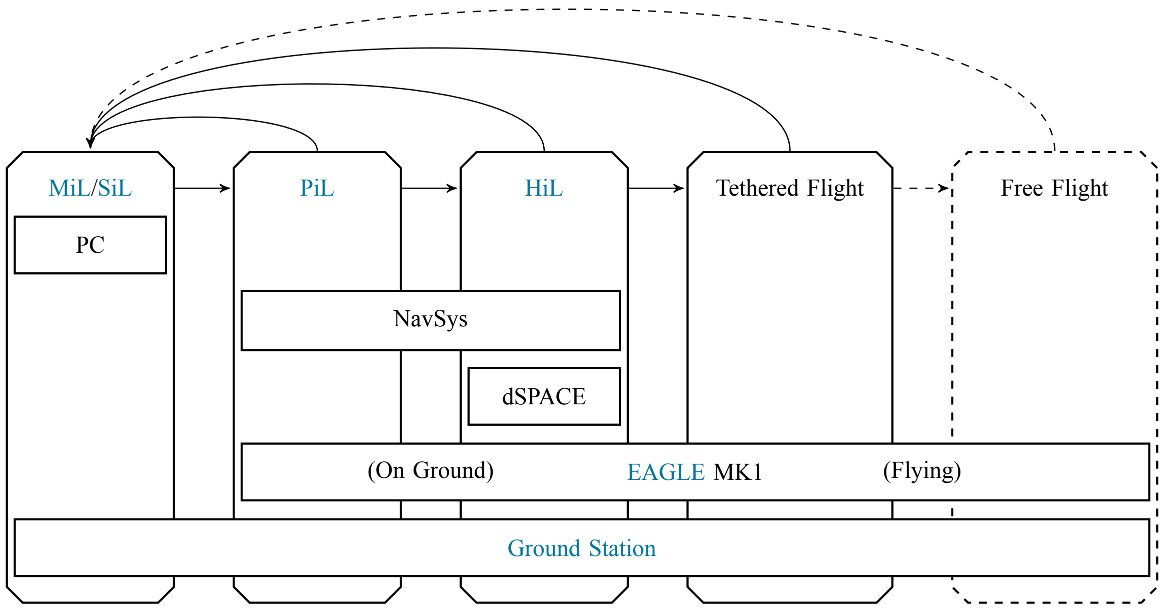


Figure 5: Development Process

The hardware setups that can be utilized in each development step is given as a vertical bar in Figure 5. These setups were used during the validation of the current flight software and are described in Section 4.1. The sections below describe the development steps and how the listed hardware can be used.

4.1 Test Hardware

During the course of the EAGLE project different test benches have been established. Most of these setups were used multiple times, with some modifications or enhancements over time. They are seen as the basic tools for testing and maturing the OBS for flight worthiness.

4.1.1 Navigation System Mock-Up (NavSys)

The mock-up of the navigation system for EAGLE was built to test the sensors and algorithms for the later implementation on the lander. Figure 6 shows the box with the appendages for, e. g., GNSS antenna, laser altimeter, and battery. Integrated within the box are a Microelectromechanical Systems (MEMS) IMU, GNSS receiver and an OBC for the auto-coded navigation filter. Additionally, a barometer and a magnetometer are available as inputs to the filter.

The NavSys has access to all the available measurements as the navigation system on board of EAGLE, but it is light enough to be carried around for quickly testing new algorithms. During development of the navigation filter, the system was used to record measurement data that was fed into a (offline) SiL simulation, later the navigation solution was computed online, monitored by the ground station.

Occasionally, the setup was modified, e. g., a navigation camera was mounted to the casing to validate the EAGLE Tag Navigator (ET-NAV) system [2]. For performance verification a second GNSS antenna and a GNSS-splitter was installed to compare the navigation solution to a ground-truth produced by a geodetic grade GNSS receiver with a correction service for centimeter level accuracy. With the second antenna the heading and the roll angle was available for comparison to the on-board solution [4].

The NavSys is intended to be used for real-time tests of auto-coded software, for validation of the navigation filter with real measurements, and as a test bed for a new sensor later to be integrated on EAGLE.

4.1.2 dSPACE System

The *dSPACE* real-time simulation system is available for testing new hardware interfaces, or for identification of actuators. Similar to auto coding a Simulink model for the OBC, a model is compiled for the dSPACE box that has extensive access to digital and analog interfaces. It was used during the identification of the jet engine and the TVC system [1].

DOI: 10.13009/EUCASS2019-772

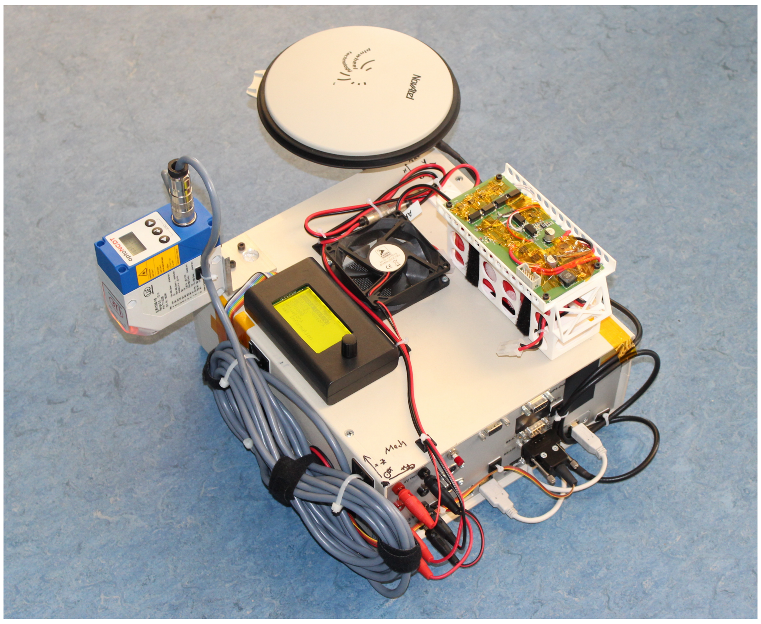
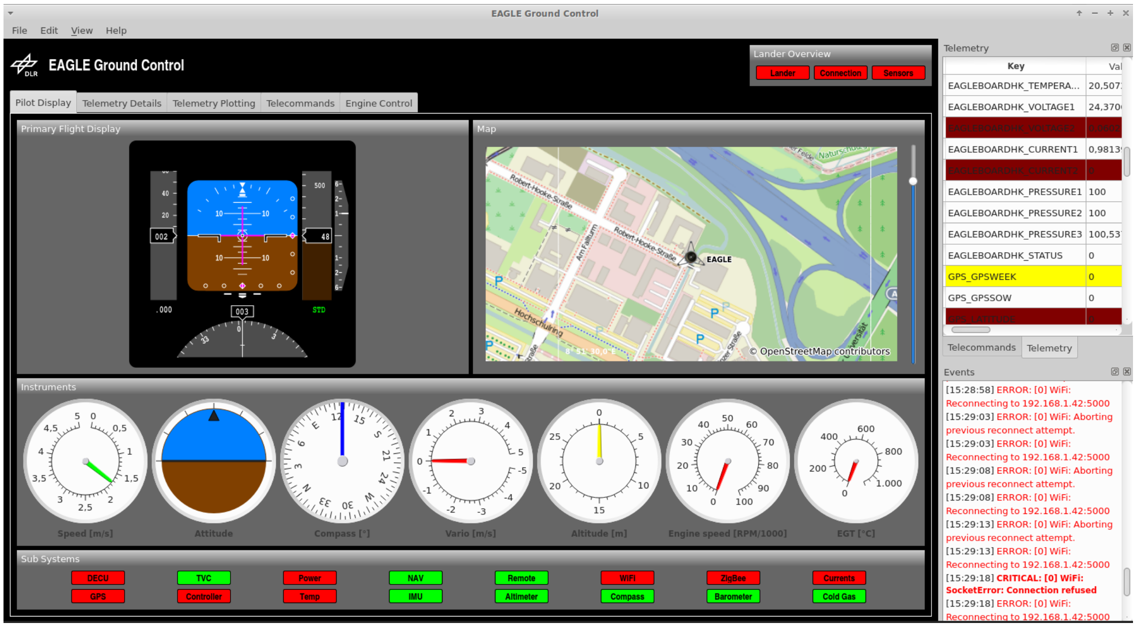


Figure 6: Navigation System Mock-Up (NavSys)

The driver software for the engine and the TVC servos is the same as used on the OBC of EAGLE, minimizing the effort for such a test setup.

4.1.3 Ground Station

For monitoring and commanding the OBS the TM/TC infrastructure can be accessed with the Ground Station software running on a *Linux* laptop. The software is developed in the *Qt* environment that is highly extensible. To evaluate the overall state of the vehicle it implements an overview panel, shown in Figure 7. Other specific panels show, e. g., the state of the engine, but also a general panel for plotting any TM/TC value is available.



Key	Val
EAGLEBOARDHK_TEMPERA...	20,507
EAGLEBOARDHK_VOLTAGE1	24,370
EAGLEBOARDHK_VOLTAGE2	0,0602
EAGLEBOARDHK_CURRENT1	0,9813
EAGLEBOARDHK_CURRENT2	0
EAGLEBOARDHK_PRESSURE1	100
EAGLEBOARDHK_PRESSURE2	100
EAGLEBOARDHK_PRESSURE3	100,53
EAGLEBOARDHK_STATUS	0
GPS_GPSWEEK	0
GPS_GPSSOW	0
GPS_LATITUDE	0

```
[15:28:58] ERROR: [0] WiFi:
Reconnecting to 192.168.1.42:5000
[15:29:03] ERROR: [0] WiFi: Aborting
previous reconnect attempt.
[15:29:03] ERROR: [0] WiFi:
Reconnecting to 192.168.1.42:5000
[15:29:08] ERROR: [0] WiFi: Aborting
previous reconnect attempt.
[15:29:08] ERROR: [0] WiFi:
Reconnecting to 192.168.1.42:5000
[15:29:13] ERROR: [0] WiFi: Aborting
previous reconnect attempt.
[15:29:13] ERROR: [0] WiFi:
Reconnecting to 192.168.1.42:5000
[15:29:18] CRITICAL: [0] WiFi:
SocketError: Connection refused
[15:29:18] ERROR: [0] WiFi:
Reconnecting to 192.168.1.42:5000
```

Figure 7: Ground Station User Interface

DOI: 10.13009/EUCASS2019-772

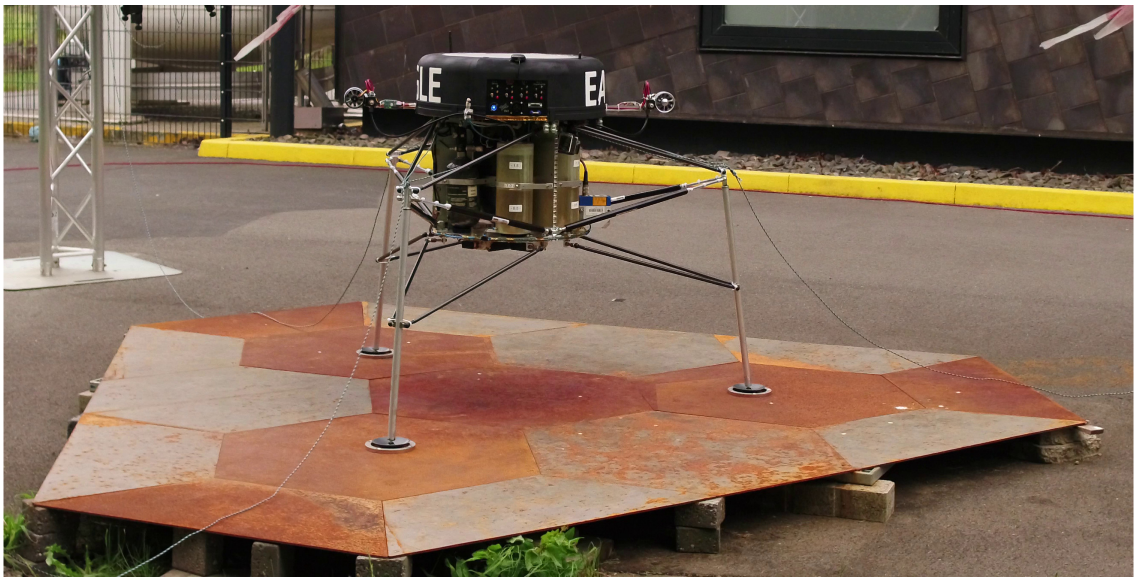


Figure 8: Lift-Off and Landing Test of EAGLE within NEST

4.1.4 Navigation System for HiL Simulation (NavSys HiL)

The NavSys from Section 4.1.1 can be used in combination with the dSPACE box from Section 4.1.2 as a full HiL test bench. The software running on the NavSys's OBC can be the exact same as the software running on EAGLE during a test flight, by only changing the parameter set stored in the configuration management system introduced within Section 3.1. A harness connects both system for all the communication links of the sensors and actuators utilized in the test.

4.1.5 EAGLE MK1

The first fully integrated lander *EAGLE MK1* is the final step for qualifying software for flight. For qualification tests this has to be done within the NEST Environment for Suspended flight Tests (NEST) facility, which allows to fail a test without damaging the vehicle. Reasonably well tested GNC software can then also be tested under relaxed safety restrictions, like the lift-off and landing test shown in Figure 8. All actuator tests were conducted with the real lander. The roll control system was tested in a lab even before EAGLE was fully integrated. By disassembling the landing gear, the current version of EAGLE can be mounted to a force-torque sensor for, e. g., actuator identification. [1]

4.2 Test Environments

Newly developed or modified software has to be qualified before it can run on EAGLE. The qualification process is not fixed to a precise procedure but it should be ensured to expect a, first of all, safe but also positive outcome of a flight test. A flight test within the NEST can be seen as a further validation step towards free flight, but also as the final demonstration of a milestone. E. g., the lift-off and landing milestone[6] was achieved this way.

For each possible validation environment the utilized hardware setups from Section 4.1 are referenced. The core software is able to run in every of theses environments, only differences of interfaced hardware must be taken into account. For example, the SiL environment does not provide access to the real actuator hardware. Only a simulated actuator performance model is used, but the controller and MVM are the same no matter in which environment it is executed.

4.2.1 Model-in-the-Loop and Software-in-the-Loop

The first step during development of a new software functionality always starts with a model running in Simulink. The differentiation between MiL and SiL is not seen as a clear line. For example, a new device driver might be written in C/C++ directly, whereas a new controller most probably is a model. The important part of this step is that it can be executed on a normal workstation.

[6]`https://gnc.dlr.de/s/yt/eagle-liftoff-landing`

DOI: 10.13009/EUCASS2019-772

During the development of a navigation filter two general concepts can be distinguished. Either the filter is tested and tuned with real data, may be recorded by the NavSys (cf. Section 4.1.1), or the sensors are simulated, producing synthetic data. The simulation of a closed loop system necessitates sensor models in any case, but crosschecking the filter with real recorded data is a recommended approach.

Within such a closed loop simulator the controllers can be developed and tuned. For EAGLE four distinct controllers work to stabilize the lander:

1. Roll controller for the RCS,
2. engine controller commanding the jet engine,
3. vane controller steering the servo motors, and
4. position controller varying the desired attitude to track a reference position/velocity.

With a full working set of available controllers, a new single controller design is relatively easy to handle within simulation. The vehicle is brought into stable flight and only then the controller under testing is activated by the MVM. The approach of starting with a simpler plant model, e. g., a one dimensional model for the roll axis, is of course valid, but the step towards the ready-to-use 6-Degree of Freedom (DoF) simulator is small, and the benefit of testing a controller design immediately in a verified simulation environment is high.

The outcome of a SiL simulation campaign should result in a tested Simulink library block that can be directly executed in a PiL/HiL environment. To ensure that new functionality, e. g., GNC or MVM, works in reality very close as in simulation, the simulation itself must be fine tuned to the data fed back from a real flights. The tuning can be done by performing specific flight test as was done for inertia estimation[5, 4], or, e. g., by comparing the closed loop behavior of a hover flight between simulation and reality.

Another valuable way to utilize the available simulation is operator training and procedure testing. Running the model in real-time allows to connect the Ground Station to monitor the telemetry and command the simulated vehicle. This allows to operate EAGLE in a safe environment and already evaluate if it is realistic to perform an intended procedure during a real flight test.

4.2.2 Processor-in-the-Loop

A PiL test is a pre-stage of a HiL test with relaxed focus on the interface to the avionic. Because immediate HiL campaigns are possible (cf. Section 4.2.3), going first through only processor centered investigation is generally not seen as beneficial for developing software for EAGLE. For some simple investigations, like general run time tests of executing an auto-coded model, performance tests of complex algorithms, checking for real-time violations, or execution time measurements it is not necessary to use the full HiL setup.

4.2.3 Hardware-in-the-Loop

Multiple HiL test benches were used during the development of EAGLE. A full closed loop simulations is available with the NavSys from Section 4.1.4 connected to a dSPACE real-time simulation system. Both systems are connected over the same interfaces as on the real system (E. g., serial link for IMU, Global Positioning System (GPS) receiver, and Ethernet for the data stream from ET-NAV). The setup includes a live visualization of the lander and its surrounding, and the Ground Station with the full TM/TC chain. This setup is used to validate the software before flying it in a test (tethered) flight, and to be reasonably sure that a test will work as in simulation.

Another important HiL test bench was built for testing and identifying the jet engine and the TVC system[1]. The tests were not performed in a closed loop fashion, but the actuator commanding and the reaction force (and torque) measurements were recorded synchronously by the dSPACE system.

4.2.4 Demonstration

The test environments described in the previous sections do not interface with the real world except the HiL setups for identification of actuators. The NavSys from Section 4.1.1 is a full navigation system that does not need a plant model or sensor simulation. It is a complete system that provides position and attitude based on real measurements (IMU, GPS receiver, magnetometer, laser altimeter, etc). Its purpose is to allow the testing of the navigation system, and to record real measurements used in the SiL environment for testing the navigation filter offline. The final step of the software qualification process is to prove that the developed software executes in real-time on board of EAGLE and is able to control the vehicle in the intended way. This should be done in a safe environment, like the tethered flight facility NEST.

DOI: 10.13009/EUCASS2019-772

Results gained during such a tests are then fed back to the SiL framework from Section 4.2.1 increasing the fidelity and confidence of the simulation.

In the future a free flight without a ”safety net“ will finally prove the overall functioning of the OBS in situations that cannot be achieved in such a safe environment. Hovering could be fully proven within NEST, as lift-off and landing was done, but if the purpose is to demonstrate, e. g., a powered descent maneuver from a significant altitude, then the here described software development process will limit the probability of a failed demonstration. By having the simulation model updated with new findings from real tethered flights, the simulation has been developed to an adequate tool that fully represents a free flight.

5. Examples Applying the Development Strategy

The development process and the available tools from Section 4 and 3 have been created during the course of the project. Software development and testing as described above was exercised for several functionalities. Two examples will be presented in this section that successfully facilitated the process to reach quickly a level of confidence in the software to advance to a stage that allows real flight.

5.1 Implementation of the MVM

Only when planning to lift-off from ground, it was inevitable to think of a Mission and Vehicle Management system that differentiates between flight phases to be able to automatically (de-)activate actuators and that handles the transitions between GNC functions. This was not required during the beginning of tethered flights. The early flights were fully controlled by an operator, manually switching on and off controllers. Because the whole system was never flown before the risk of automatically handling this procedure was too high and a step-by-step approach was chosen. Each controller was tested in the simplest possible way possible. E. g., the roll controller could be tested alone directly within a laboratory, but the thrust vector controller needed a running engine, and the engine controller was first activated after the thrust vector controller proofed to be working.

Later flights were still operated this way, but due to the already gained experience the operator could switch on all controllers in a fast succession.

The envisaged sequence for an automatic lift-off required to switch the controllers in a fast succession, or during landing to deactivate them and automatically putting the jet engine to idle. Experience drawn from prior flights showed that this is not feasible to be manually performed by an operator.

The MVM system was planned as a simple state machine with a limited number of states, that can handle automatic mode transitions and is always accepting (abort) commands from the Ground Station. It was decided that automatic transitions based on measurements will only be based on laser altimeter measurements. The device proved to be very reliable and accurate. The transitions triggered by the laser altimeter are only basic thresholds used to detect the lift-off and landing.

The main task of this state machine is to provide the vehicle mode to individual managers, also simple state machines, for each single vehicle mode. Figure 9 shows the design of the MVM system of EAGLE. The output of these managers is a (changing) set of controllers and a guidance scheme. In this context, controllers can be as simple as a constant value, or a ramp up of the engine.

This system was implemented into the SiL simulation and tested in a step-by-step approach. The individual mode manager was first as simple as possible, e. g., landing was performed by switching off the engine. The designs were later refined to perform the given task in a more suitable manner and that commands from the (virtual) Ground Station allow to abort a flight during any point in time. This early testing helped to identify an additional vehicle mode for emergency landings that was necessary for quicker landings that might still be able to save the vehicle in case of unforeseen dangerous events.

The laser altimeter’s measurement are filtered to allow for some errors in measurement, e. g., a swirled-up object in in the Line of Sight (LoS) of the laser.This modified measurement is called *Feet2Ground* and it is accessible for the mode managers that need to detect the lift-off and landing, or some intermediate height threshold within their respective phase. After the SiL simulation gave the anticipated results, the model was auto-coded to be deployed on the HiL test bench for immediate testing. On the one hand, the operations could already be prepared by modifying the Ground Station for commanding the vehicle state and testing what information an operator needs to conduct flight tests. On the other hand extensive tests of lifting off and landing, aborting a maneuver, and performing emergency landings, helped to be sure to have a robust implementation of the intended MVM function. This was then quickly verified with the real hardware by pulling EAGLE up and letting it down with a crane.

During the next real tethered flight this functionality was tested on the real vehicle with a running engine. EAGLE was still hanging from the tether and the MVM software was tested by lifting of from this position. In case of an aborted

DOI: 10.13009/EUCASS2019-772

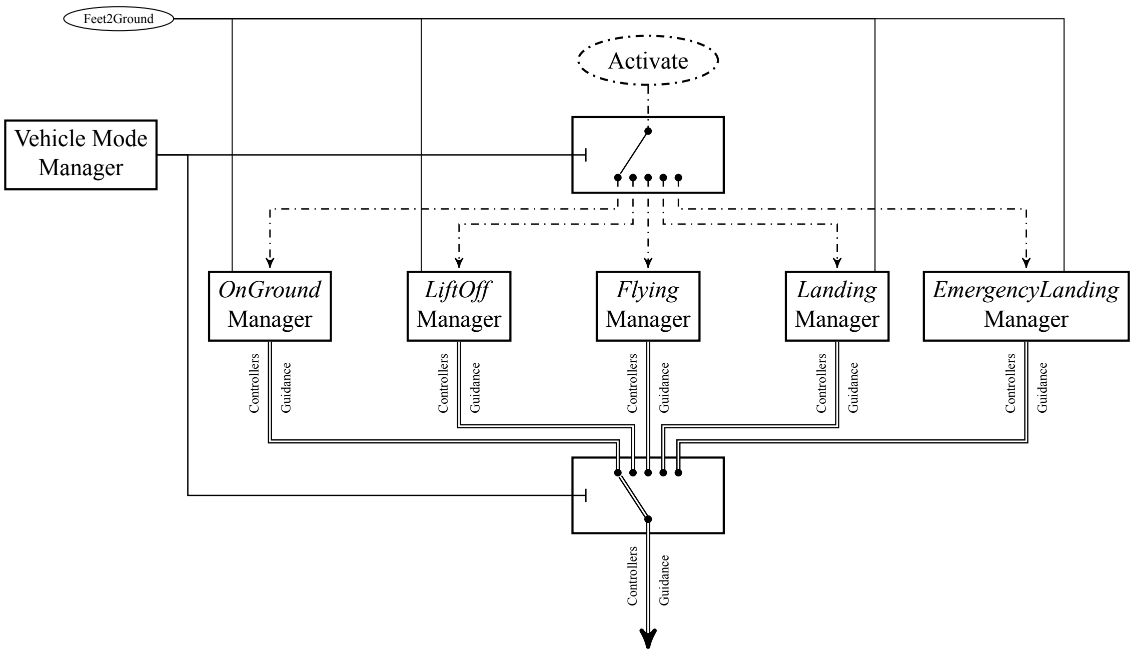


Figure 9: MVM Architecture of EAGLE

maneuver it was safely possible for the operator to decide to abort by stopping the engine, letting EAGLE fall into safety. The system worked as intended and lifting off and landing into the tether was a success, the concept of the operator training during the HiL campaign proved valuable to allow to observe the correct sequence of the MVM system. The only minor change for the following test was to tune the position controller to be more aggressive, to really stay away from the truss structure of the NEST during the initial lift-off (this was tested in SiL in advanced). The lift-off and landing test worked flawlessly, post-processing of the data showed the correct behavior as seen in simulation [4].

5.2 Integration of a Differential GNSS Receiver System

The integration of a precise satellite navigation into the EAGLE system is still a work in progress. Over the whole project life span the topic of an aided GNSS system was pursued. From the beginning it was planned to implement a phase-based differential solution. The Real Time Kinematic (RTK) approach is still under investigation, but tests showed the feasibility and the centimeter-level precision is unmatched.

The differentially enhanced version of the overall GNSS system, including EAGLE and the EGSE, is shown in Figure 10.

The Ground Station was modified to send a RTCM (Radio Technical Commission for Maritime Services) data stream from a stationary GNSS receiver to EAGLE that relays it to the on-board receiver. Additional functions within the Ground Station were added to monitor the stationary receiver and the data stream. Only minor modifications were necessary for the OBS not necessitating "re-qualification", the important difference with a forwarded data stream of correction data is a position measurement with an improved accuracy send by the on board GNSS receiver.

The modification was tested with the NavSys, proving that centimeter-level accuracy is possible, but also that it can take time until the system converges, and that RTK can loose the phase-lock, resulting in a reduced accuracy of the provided measurements.

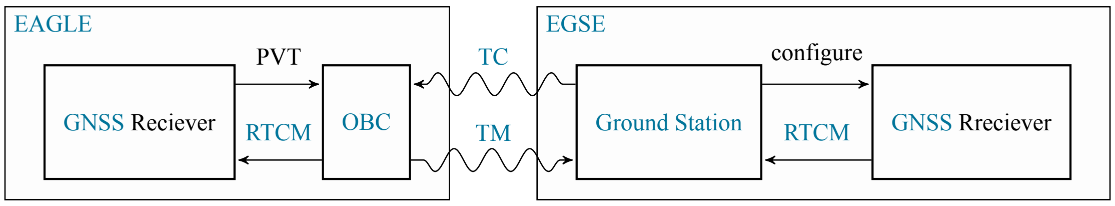


Figure 10: Differential GNSS Receiver Architecture

DOI: 10.13009/EUCASS2019-772

The housekeeping data of the receiver provides information about the current differential state of the solution. Without the correction from the Ground Station the on-board receiver computes a *single* solution, with an activated stream it switches to *float*, with highly reduced noise, but with a slowly varying bias in the order of magnitude of the single solution error. When the receiver ”locks“ onto the phase of the satellite signals the status switches to *fix*, giving a position solution with very little noise and the expected centimeter level accuracy.

The important part of the validation strategy is to be sure that EAGLE can safely fly with a varying position accuracy. Under *fix* conditions the difference as with flying with the ET-NAV camera should be negligible. What is necessary to prove is that EAGLE can fly with the meter-level accuracy provided by the single on-board receiver and without attitude aiding by ET-NAV. Obviously the positioning accuracy will degrade significantly, but the laser altimeter will stabilize the vertical channel, only resulting in a horizontal influence that has to be investigated.

The investigation starts within the SiL environment. The reduced accuracy is, as expected, a problem for the position controller, which performed well under the past conditions. The jumps in the estimated position by the GNSS receiver, already smoothed by the Kalman filter, lead to a vehicle behavior that is not stabilized anymore. Tuning of the gains of this controller show that a stable flight is possible, but as expected, EAGLE will drift horizontally according to the GNSS error.

The implementation of a gain scheduling based on the accuracy of the current navigation solution needs only minor modifications of the OBS. A simulation test campaign shows that the full flight envelope is still possible. It focused on the transition between all phases under changing navigation performance, for example, if landing is still possible with a single solution that leads to higher horizontal velocities without tipping the vehicle over during touch down. The navigation switch-over planned during real flight tests is a critical situation that could already be tested. At least first free flights are planned to include the well tested optical tag navigation (ET-NAV), that is turned on and off during flight to test the satellite based navigation. Because the tag navigation measurement includes attitude information, the absence of these information will influence the overall controlled system.

The next steps planned are the adaption of the Ground Station to allow to command the navigation filter what measurements to use, and testing and training the operations within the HiL real-time test bench. During the next tethered flights, the differential GNSS infrastructure needs to be tested within NEST, but it is assumed that the signal reception, and multi-pathing effects within that environments, do not allow for a GNSS based test. Conducting a tethered flight with the navigation performance of a single receiver solution is most probable not possible. Either an enlarged test facility is necessary, that is placed where an unobstructed GNSS reception is ensured, or a free flight on an ”open field“ with tag based navigation.

The approach for conducting a safe flight described within this section already shows with a high confidence that a GNSS based flight is possible. Many tests show, based on the verified simulation environment, that no show-stoppers were identified and that the project can advance as planned. The ultimately prove is pending and can only be provided by a real demonstration flight.

6. Conclusion

This paper presented the current approach of developing software for the VTVL vehicle EAGLE. An overview was given for the available hardware environments for testing and validation. The tool chain set up for this project was described and the advantages of the auto-code based development cycle was illustrated by showing how it was exercised for enhancing the system.

Especially, the self-developed tools (Configuration Management and automatic TM/TC handling) allow for future modifications of the system. A second vehicle with a different parameter set, or more and different sensors, can be easily included into the development of new software.

The process of testing and validating software functionality was conducted multiple times from first simulations to finally flying in a tethered environment. It showed a very mature simulation environment that delivers results in line with real tests.

The next big step is to perform the presented process in a full cycle, including a free flight without a safety net. For this the discussed GNSS based navigation must be fully integrated and validated, but also operational and legal processes have to be fulfilled.

References

[1] Marco Sagliano, Michael Dumke, and Stephan Theil. Simulations and Flight Tests of a New Nonlinear Controller for the EAGLE Lander. *Journal of Spacecraft and Rockets*, pages 1–14, Oct 2018.

DOI: 10.13009/EUCASS2019-772

[2] W. C. Leite Filho, M. Dumke, and Theil S. Control System Design of the Lander Demonstrator EAGLE. EUCASS, 2015.

[3] Michael Dumke, Marco Sagliano, Piyapat Saranrittichai, Guilherme Fragoso Trigo, and Stephan Theil. EAGLE - Environment for Autonomous GNC Landing Experiments. In *10th International ESA Conference on Guidance, Navigation and Control Systems*, Juni 2017.

[4] Michael Dumke, Guilherme F. Trigo, Marco Sagliano, Piyapat Saranrittichai, and Stephan Theil. Design, Development, and Flight Testing of the Vertical Take-Off and Landing GNC Testbed EAGLE. *CEAS Space Journal*, 2019. (Under Review).

[5] M. Sagliano, A. Heidecker, and M. Dumke. Spacecraft Inertia Matrix Identification via Convex Optimization. *AIAA Journal of Spacecraft and Rockets*, 2019. (Under Review).