Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Prof. Dr.-Ing. Hakan Kayal
Professur für Raumfahrttechnik
Lehrstuhl Informatik VIII

BACHELOR THESIS

Development and Post-Flight-Analysis of HORACE the Horizon Acquisition Experiment

submitted by

Thomas Rapp

to the Department of Mathematics and Computer Science
in Partial Fulfilment of the Requirements of the Degree of
Bachelor of Science
with the Major of Aerospace Information Technology

Würzburg, 20th August 2014

Thomas Rapp
thomas.rapp@stud-mail.uni-wuerzburg.de

c/o
Prof. Dr. Hakan Kayal
Dipl.-Inf. Gerhard Fellinger
Chair of Computer Science VIII
Aerospace Information Technology
Department of Computer Science
Faculty of Mathematics and Computer Science
Julius-Maximilian University Würzburg
Am Hubland, D-97074 Würzburg

ABSTRACT

This bachelor thesis systematically analyzes and discusses both the findings of the launch campaign and flight of the Horizon Acquisition Experiment (HORACE) and the main steps and decisions of its development. HORACE was part of the sixth cycle of the REXUS programme and was launched on May 28 2014 13:30 (UTC+2) from Esrange Space Center (Kiruna/Sweden) on REXUS 16. It was the first basic step of research towards a new sensor system for attitude determination based on horizon acquisition using image data of ordinary cameras and image processing technologies.

After an introductory chapter, which gives an overview of the experiment in general (the project frame, objectives and concept), both the main problems experienced during the campaign and flight and the main achievements of the experiment are systematically analyzed, using the payload data, as well as the experiment documentation, results of simulations, data of further investigation and recorded flight data of the sounding rocket. The main problems are: corruption of the file-system of the Measurement Unit; shifted electrical ground; interferences with the GPS-system of the RXSM and overexposure of the payload cameras. For all problems the observed behavior and performance of a specific setup is compared to the expected one, before the methods and results of the conducted analysis are presented and possible solutions outlined.

To evaluate the achievements, the performance of all subsystems is compared to the corresponding requirements and the main steps and decisions of the development are explained and discussed. The subsystems leading to achievements are: electronics & electrical interfaces; thermal design; mechanical design; OBDH-software of the core systems; algorithm for horizon acquisition; communication middleware; measurement unit and ground station software.

Summarizing both the problems and achievements leads to an overall mission evaluation of an 80% partial success, as the experiment design and implementation was very good, but several of the questions posed in the Experiment Objectives could only partially or indirectly be answered.

In a last part an outlook on possible follow-up experiments is given and the most appealing directions for further development are outlined. Those are: re-flight on REXUS; upgrade to full attitude determination; miniaturization for satellite mission; porting to existing hardware.

Keywords:

horizon, acquisition, Horizon Acquisition Experiment, HORACE, REXUS 16, evaluation, post-flight, development, analysis, discussion, flight performance, problems, mission evaluation

CONTENT

INTRODUCTION & TECHNICAL BACKGROUND

"Non omnis moriar" – "I will not wholly die" (Rudd, 2014), a phrase of the ancient Roman lyric poet Quintus Horatius Flaccus, who lived in the first century BC and is also known as "Horace", fits quite well the scientific and technical background of the HORACE-Project – the Horizon Acquisition Experiment – since it was the first basic step towards a new sensor system for attitude determination of satellites during emergencies.

Up to today many satellite missions still fail – or "die" – due to a malfunction of the attitude determination and control system (ADCS), which is an essential sub-system for most satellites. The attitude – not to be confused with the position – must be determined and controlled more or less precisely to direct solar panels to the sun or communication antennas to the ground stations. If a satellite fails this essential task, it will run out of energy or cannot be commanded anymore and eventually will probably be completely lost. Therefore the ADCS must always work properly – not only during nominal phases of the mission, but also in emergency cases, when the satellite is switched to the so called safe mode. This might happen when the satellite is tumbling and spinning uncontrolled, for example after a collision with space debris or a malfunction of the main ADCS. To prevent the satellite from "wholly dying", a new sensor system for attitude determination is envisioned, which is capable to (re)acquire a satellite's attitude also during those emergency cases autonomously. An autonomous system is not only required to face the general trend towards more and more autonomous satellites but also because it is improbable to have a proper communication link to the satellite during an emergency.

Among all possible types of attitude determination sensors a horizon sensor is considered to be the best approach for this scenario. No high accuracy is required, but it shall work while the satellite might spin with high rates (unlike star-trackers) and also during eclipse (unlike sun-sensors). As the central body (in most cases the earth) can be distinguished from the outer space also during eclipse and as motion modes in which the central body is never visible are very improbable, a horizon sensor seems to be a fair choice.

Unlike existing horizon sensors the new sensor shall not work in the infrared spectrum but the visible spectrum, capturing images with an ordinary camera which are evaluated by powerful image processing algorithms for two reasons. Firstly, to reduce costs and therefore to make the sensor system also available for smaller missions. Secondly, the experiment works in the visible spectrum to artificially worsen the initial conditions and thus emphasizing the capabilities of the software components of the system. Hence, in an even later version the software will probably be that generic that it can process image-data from any camera. As many satellites already carry payload cameras therefore the whole "sensor system" may only be a software-package and does not require own hardware, making it even more suitable as a system for emergencies.

The aim of the Horizon Acquisition Experiment was to prove or disprove the very basic feasibility of this approach for a new sensor system with a flight on a sounding rocket. Such a flight was considered suitable as it would be possible to capture realistic, space-like images and the motion of the unguided and uncontrolled vehicle would be similar to the motion of a tumbling satellite. (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 7-8)

After one and a half year of development HORACE was conducted during the flight of REXUS 16 which was launched on May 28 2014 13:30 UTC+2 from Esrange Space Center, Kiruna/Sweden.

Therefore, this thesis shall not only describe the design and development process but also evaluate the experiment and its flight performance, regarding both the main problems experienced during the launch campaign and the flight and the main achievements. After an introductory part, which provides a broad but short overview of the experiment, in the second chapter the main problems are analyzed and discussed thoroughly and possible solutions are outlined in separate subchapters by following a systematic approach. In the third chapter the main achievements, separated by the subsystems, are analyzed and discussed as well, following again a systematic approach and also regarding the development process. The findings of those both chapters are summarized to an evaluation of HORACE in general before finally in the last chapter an outlook on possible follow-up experiments on the way to a full sensor system is given.

1 EXPERIMENT & PROJECT OVERVIEW

In this introductory chapter a broad overview over the Horizon Acquisition Experiment regarding the project frame of the REXUS programme, the objectives of HORACE and its basic concept is given, which includes all necessary information to understand the evaluation of the experiment conducted in the main part.

As most of this fundamental information was already part of the experiment documentation, the corresponding chapters of the documents (in most cases (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014) and (Mawn, Kaczmarczik, & Schmidt, 2014)) are often directly cited and where appropriate explained or summarized. Therefore, readers who are used to the HORACE-project (its backgrounds, objectives, concept and nomenclature) and the REXUS programme may skip some of the subchapters and only refer to them if questions arise, while others seeking for more detailed information may directly refer to the underlying cited documents.

1.1 The REXUS Programme

The Rocket (respectively Balloon) Experiment for University Student (REXUS/ BEXUS) programme is described by its organizers as following:

> "The REXUS/BEXUS programme allows students from universities and higher education colleges across Europe to carry out scientific and technological experiments on research rockets and balloons. Each year, two rockets and two balloons are launched, carrying up to 20[1] experiments designed and built by student teams. REXUS experiments are launched on an unguided, spin-stabilised rocket powered by an Improved Orion Motor with 290 kg of solid propellant. It is capable of taking 40 kg of student experiment modules to an altitude of approximately 90 km. The vehicle has a length of approx. 5.6 m and a body diameter of 35.6 cm. (…)
>
> The REXUS/BEXUS programme is realised under a bilateral Agency Agreement between the German Aerospace Center (DLR) and the Swedish National Space Board (SNSB). The Swedish share of the payload has been made available to students from other European countries through a collaboration with the European Space Agency (ESA). EuroLaunch, a cooperation between the Esrange Space Center of SSC and the Mobile Rocket Base (MORABA) of DLR, is responsible for the campaign management and operations of the launch vehicles. Experts from DLR, SSC, ZARM and ESA provide technical support to the student teams throughout the project." (EuroLaunch, 2014)

Unlike other larger university projects with student participation, during which students usually only take part in single steps or phases of the project, the relatively short project lifecycle of REXUS, nominally 1.5 years (Mawn, Kaczmarczik, & Schmidt, 2014, p. 20), makes it possible that the students take part in the whole project – from the first rough sketch for a proposal to the post-flight evaluation.

[1] *in total for REXUS and BEXUS; each REXUS-rocket typically carries 4-5 experiments, thus 8-10 experiments are performed on REXUS each cycle.*

Thus, after having been initiated by five Bachelor students of Aerospace Information Technology from University of Würzburg in October 2012 HORACE became part of cycle 6 of the REXUS programme after the selection in December 2012. The experiment then was designed and implemented throughout 2013 and the campaign preparations and testing of the full experiment were carried out in spring 2014. Finally, the launch campaign took place from May 19 2014 to June 02 2014 at Esrange Space Center in Kiruna/Sweden and HORACE was successfully launched as payload of REXUS 16 on May 28 2014 13:30 UTC+2 reaching an altitude of 87km during its 14-minutes-long parabolic flight.

1.2 Mission Statement

The Mission Statement of HORACE was defined at a very early stage of the project and is given in (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, p. 8):

> "HORACE on REXUS 16 is a technology demonstration mission for autonomous earth detection on satellites. The aim is to prove or disprove the general technical feasibility of the outlined approach.
>
> During the mission the functionality and robustness of the general approach is tested under realistic, space-like conditions, by means of the HORACE Flight Segment. After post flight evaluation it shall be determined whether the approach of autonomous horizon acquisition with a camera in conjunction with image processing algorithms running on an embedded system connected to the camera is indeed apt to (re)acquire a satellite's attitude under nominal or stress conditions."

Of this Mission Statement one has to especially keep in mind that the aim was not to develop a full sensor system for attitude determination via horizon acquisition and its demonstration, but only to "prove or disprove" the very basic feasibility of that general approach. As the results could not have been predicted (HORACE was the first step of research) it was important to keep the question between both options (proving or disproving) open. Otherwise, if one had aimed for example at proving the approach and had failed because the approach had not been suitable in general, the mission success would have been reduced although the acquired technical and scientific knowledge had been equivalent.

1.3 Experiment Objectives

The Experiment Objectives were defined early as well and are directly based on the Mission Statement:

> "With HORACE, whose development will be part of the mission, the following **primary objectives** shall be reached:
>
> - Investigate whether horizon acquisition can be performed accurately enough for attitude determination.
> - Determine whether the very dynamic and time-critical problem can be solved with an embedded system with reasonable time resolution and power consumption.

Secondary objectives are:

- to show physical or systematic limits and problems of the general approach.
- to determine, if a future attitude determination system following the general approach would be applicable also for small satellites."

(Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, p. 9)

One has to note again that all posed questions are kept open ("investigate whether…", "determine whether…", "show…") and not limited to a single option for the same reason as for the more abstract Mission Statement (cf. 1.2). Furthermore, the development of the experiment itself – with only indirect link to the questions to be answered – is explicitly defined as part of the overall mission, which is important for the overall evaluation of the mission success (cf. 4).

1.4 Experiment Concept

This chapter summarizes chapter 1.4 (pp 9-10) and 4.1 (pp 32-34) of (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014) and *hence often combines information from both chapters in the same sentence. Please note that therefore and to increase simplicity and readability not every single piece of information is explicitly referenced.*

The complete Horizon Acquisition Experiment (HORACE) consists of the Flight Segment (FS), carrying out the actual experiment onboard of REXUS 16, and the Ground Segment. Within the FS there are five subsystems: the core system (CS), the power distribution unit (PDU), the camera, the measurement unit (MU) and the mechanical structure, whereas the Ground Segment consists of the ground station and both the electrical and mechanical ground support equipment (EGSE and MGSE) (cf. Figure 1-1).

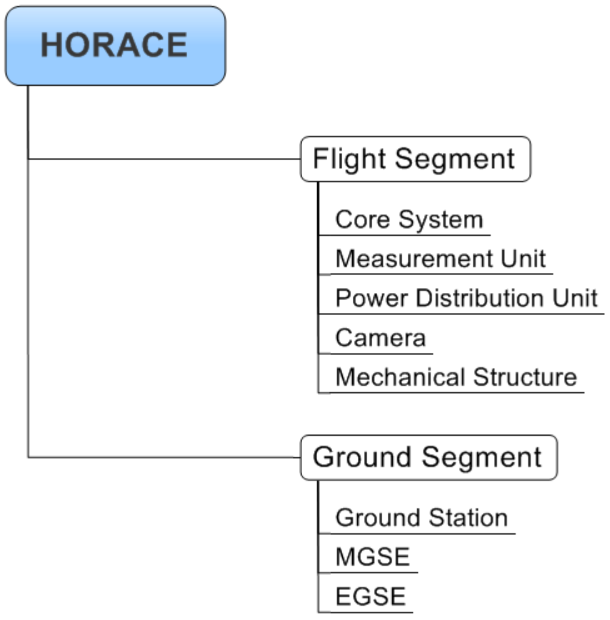

Figure 1-1: Hierarchy of HORACE (Rapp, et al., 2014, p. 9)

Within the Flight Segment (cf. Figure 1-2) the PDU conditions the unregulated power, which is provided by the REXUS Service Module (RXSM), to the voltages needed by each component and implements the hardware for the data interface to the RXSM (signals, telemetry and telecommand; optocoupler-circuits and RS-422-to-RS-232-converter).

The camera observes the outer environment of the REXUS-rocket and passes its image-data via the standardized GigE-Vision-Interface to the core system, a micro-ITX embedded PC, which first saves the image-data via SATA-II to a mass memory device (SSD) and then processes it for the actual horizon acquisition. The results of the image-processing are also stored to the SSD and parts of them are provided for downlink. Furthermore, the core system implements the protocols for telemetry (TM) and telecommand (TC) and is connected to the hardware-interface of the PDU via RS-232.

The measurement unit, an Arduino Leonardo extended with a SD-card-shield and RS-232-interface, collects health data (temperatures and currents) at distinct points of the Flight Segment and stores them to its internal microSD-card or provides them via the RS-232-connection to the core system for downlink, according to the time-line of the experiment and the software-modes. Via this RS-232-connection to the core system the two components involved in data-handling are time-synchronized to the mission elapse time (reset at LO) to ensure that all collected data can be matched for the post-flight evaluation.

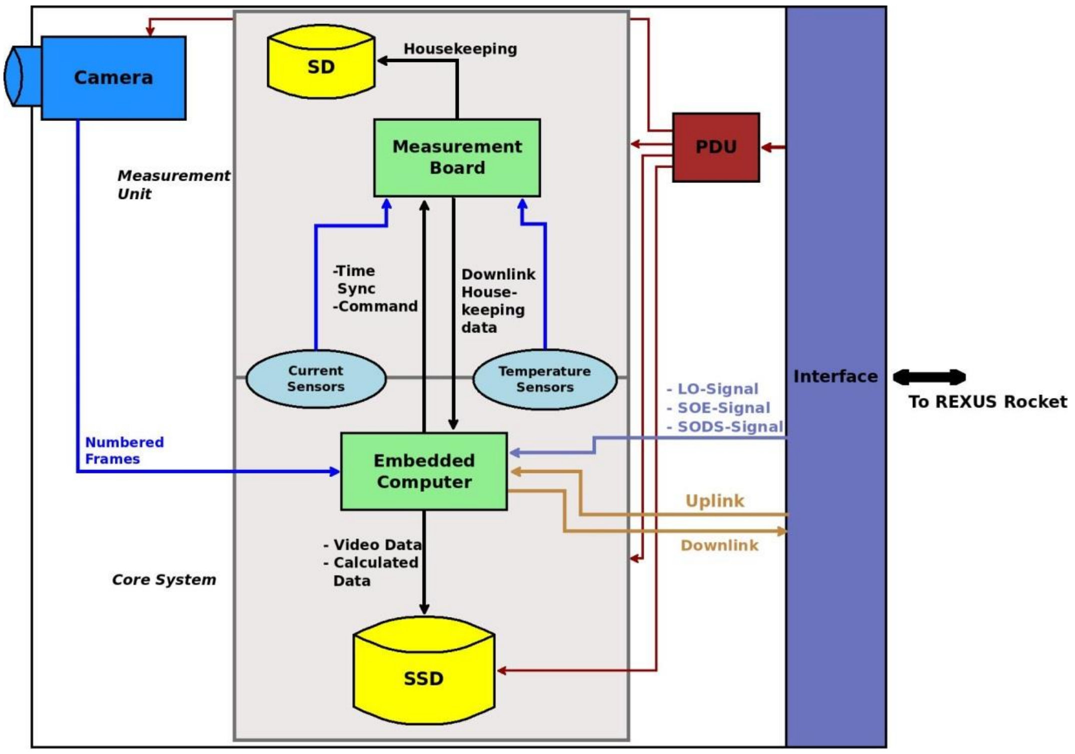

Figure 1-2: Flight Segment - experiment setup (Rapp, et al., 2014, p. 32)

The mechanical structure simply mounts all components properly to the vehicle and protects the electronics from small pieces like screws possibly floating through the rocket during micro-gravity and thus causing short circuits.

The ground station receives the downlinked data and displays them in an appropriate manner on a Graphical User Interface (GUI). Using the same GUI also telecommands can be sent to the Flight Segment via the infrastructure of the REXUS Service System. The EGSE and MGSE are used for integration, assembly, preparation and testing.

The Flight Segment works mostly autonomously already during countdown – except from only a few preparations and check-outs (clear memory etc.) – and it runs completely autonomously during flight. Nevertheless, TC can be received and executed throughout the complete timeline what was mostly used for testing.

To gather more scientific data, two nearly identical setups of the outlined experiment were flown within the same 120mm high experiment module with also two interfaces to RXSM and thus also two ground stations. The MU is the only component which was not needed twice, hence introducing the distinction between the two setups: the Master, which is connected to the MU and the Slave, which is not.

2 ANALYSIS & DISCUSSION OF MAIN PROBLEMS

In this chapter the main problems experienced before and during the launch campaign, especially the actual flight, are thoroughly analyzed, following a systematic approach: First the actual system setup (limited to the relevant subsystems for simplicity) which caused the problem is described before the expected behavior, respectively results are explained. In a third step the actually experienced and observed behavior or results are described. The fourth and fifth parts consist of the actual analysis of the discrepancy between expected and actual behavior, explaining firstly the methods and procedures of analysis with observations of the single steps and then drawing conclusions from this newly gained knowledge in the "discussion" part. As a last step possible solutions to face the issue in the future are outlined and explained.

2.1 Corruption of the File-System of the Measurement Unit

2.1.1 Setup of the Measurement Unit & the Experiment in General

The Cold Flight Simulation[2] was performed on May 27 2014 14:40 (UTC+2) and was one of the final tests before the hot countdown. Compared to all successful tests – especially to the Hot Flight Simulation[3], which was conducted most recently only three days before (May 24 2014 15:40 UTC+2) – nothing had changed regarding the Measurement Unit.

Moreover, the setup of the experiment in general, especially the Flight Segment, was the same setup as before, which also was intended to fly. A minor change compared to the successfully tested setup was that ferrite cores were placed around several cables of the Flight Segment to reduce the interference with the vehicle's GPS-system (cf. 2.3). During the troubleshooting procedure for the same issue the Flight Segment was powered on and off approximately one minute later for several times without the supervision of the team.

2.1.2 Expected Behavior

All tests before were successful regarding the MU, no direct changes for the MU were introduced and the minor hardware change with the ferrite cores was expected not to affect the MU at all. Therefore, perfectly nominal behavior, i.e. passing all self-checks before gathering and downlinking health data, was expected also for the Cold Flight Simulation.

2.1.3 Observed Behavior

Unlike literally all tests – except the ones to test the self-checks itself by provoking failures – during the Cold Flight Simulation the self-checks for the microSD-card failed reproducibly: The read-SD-self-check should have opened a prepared file on the microSD-card and should have read the content, which should have been equal to a predefined constant. The write-SD-self-check should have created a new file, written a constant content and closed the file. When reopened and reread, the content should have been equal to the written constant. Finally the

[2] Complete setup as for the Flight, complete time-line from T-600s to T+600s (including LO signal) conducted, but consumables (like pyro-cutters, scientific samples) not used.

[3] Same as Cold Flight Simulation, but consumables are used (pyros fired, samples deployed etc.). As HORACE does not have consumables, there is no difference between those simulations for HORACE (unlike for other experiments).

written file should have been deleted again. Both tests reproducibly returned via TM that they were not successfully completed, as well as the “mu clear” command (to delete old data) returned “no file to delete” although there should be old data from the previous test (Hot Flight Simulation).

Gathering and downlinking health data actually worked as expected, but without access to the microSD-card the MU would not have been able to store the data gathered during flight, i.e. it would have been lost. Thus, an issue with the microSD-card was assumed.

2.1.4 Analysis of the Issue

After disassembling the MU and replacing the microSD-card by a spare one, the self-checks were passed and the whole experiment performed perfectly nominally, confirming the assumption of a failure of the microSD-card.

The failing microSD-card was obviously electronically intact as it could be recognized with a USB-card-reader and an EGSE-laptop. But the files were neither correctly listed (some with a size of 0 bytes, another with a size larger than the capacity of the whole microSD-card), nor could any of them be opened with the EGSE-laptop, nor could new files be stored to the card (again with the EGSE-laptop).

Reformatting the card with FAT32 solved this problem and the card could be read again, respectively written to it via the USB-card-reader. Thus, a corruption of the FAT32-file-system was strongly assumed.

2.1.5 Discussion of the Issue

Further research has shown that it is a known issue that the file system may be corrupted when file-handles are still open and the files are not correctly flushed while the SD-card is removed (Arduino, 2014), (Gre, 2011). It is most likely that this issue also occurs when the whole Arduino with SD-shield and card is powered off while having file-handles open and files not being correctly flushed. That probably has been the case during the intense troubleshooting without supervision by the team for the issue with GPS (cf. 2.3.4) and its many short power-cycles. This seems plausible as during the first minute the self-checks are performed (during which file-handles are opened of course). Moreover, as soon as the experiment is put to flight-mode, the housekeeping-file stays open for the whole flight (until T+590s or manually switched to shut-down-mode via telecommand).

2.1.6 Solution of the Issue

Besides the replacement of the microSD-card, which was actually performed, there are some more possible solutions, which would have either prevented the issue a priori or would have minimized its impact.

With supervision during the troubleshooting by the team it could have been ensured (if necessary with sending corresponding existing telecommands) that all file-handles were closed properly before power-off. That quite simple operational measure unfortunately was not possible due to operational reasons.

Secondly, keeping the housekeeping-file only open, when data is actually written to it, may reduce the probability of the issue occurring. However, this was not suitable as opening and closing the file takes quite a long time. This would have even more worsened the trade-off regarding the sample-time (cf. 3.7.2).

Thirdly, as it is possible to reformat the SD-card directly via the Arduino and as there are even appropriate example-programs available, this functionality could have been implemented as telecommand to be remotely performed when necessary. But with the limited EEPROM (cf. 3.7.2) other – most likely more important – functionalities had to be traded for this security feature which was quite unlikely to be needed. Another similar possibility would have been a small utility program which reformats the card when being flashed to the MU, thus overwriting the actual flight-software. Therefore, it would not have been needed to disassemble the MU itself, whereas direct access to the experiment in general would still have been needed and further possibilities for failures due to the change of the software would have been introduced. Furthermore, this solution would have only worked because in fact solely the file system was corrupt and the microSD-card itself electrically intact.

Hence, among those several possible solutions, except from the supervision by the team, the conducted one – simply replacing the microSD-card following a detailed prepared procedure – is considered to be the best one.

2.2 Shifted Electrical Ground

2.2.1 Setup of the Flight Segment in General

As this issue was experienced first already during the Integration Week (ITW), but also during Bench Test and even the preparation during campaign and as the setup changed in between regarding minor aspects (replacement of data-storage devices or cables etc.), no exact setup is documented. All the slightly different setups, the issue was experienced with, have in common that the full Flight Segment was assembled as designed and all subsystems were working. Moreover, unlike during all tests before, during the tests on ITW the Flight Segment was electrically grounded via the structure (i.e. the experiment module).

2.2.2 Expected Behavior

Because the electrical design, especially the power system, was conceived according to the REXUS user manual (Mawn, Kaczmarczik, & Schmidt, 2014, p. 52) no issues with the electrical ground, but a fully nominal proceeding of the time-line-tests (from T-600s to T+600s) for both systems were expected.

2.2.3 Observed Behavior

Actually, during several of the time-line-tests the flight software of the master system completely crashed while the operating system stayed alive, requiring a reboot of the master core system or direct access to manually restart the flight software. However, the issue could not be reproduced in every test, but if it occurred, the master system was lost around T-30s.

Thorough troubleshooting both during ITW and during Bench Test revealed only very late, that the grounding via the structure was the critical factor. Thus, the issue could at least be reproduced in most (still not in all) tests with the actual flight-setup of the Flight Segment.

Nevertheless, although the issue could not be analyzed properly before campaign, the workaround solutions (cf. 2.2.6) reduced the impact of the issue to a level which would not have limited the experiment's performance during flight (though luckily it was not even experienced during flight).

2.2.4 Analysis of the Issue

With the knowledge of the grounding being the critical factor already during Bench Test basic analysis was performed. It was detected that the Flight Segment's structure had a higher electrical potential than the ground of the power supply. The ground was shifted by 150mV, causing a stray current of about 27mA when the structure was shortened with the ground of the power supply. However, during Bench Test it was not possible to identify a specific component which caused the ground being shifted as it was not possible to run only single components (at least the PDU was always needed to power the other components).

Furthermore, even in the meantime between the Bench Test and the campaign the issue could not be further investigated because the flight setup had to be delivered to EuroLaunch and it was not possible to reproduce the issue with a setup using spare parts.

Therefore, further analysis was only possible after the campaign. By running the single subsystems and components independently from the others but in the same setup as flown it was revealed that presumably the core systems caused the shifted electrical ground (see all measured values in Table 2-1).

Component	Potential
Master-CS: all interfaces connected	~50mV
Master-CS: c[4]: SATA, Ethernet, RXSM; n.c.[5]: MU	47mV
Master-CS: c: MU, Ethernet, RXSM; n.c.: SATA	48mV
Master-CS: c: MU, SATA, RXSM; n.c.: Ethernet	48mV
Master-CS: c: MU, SATA, Ethernet; n.c.: RXSM	47mV
Master-CS: no interfaces connected	48mV
Slave-CS: all interfaces connected	~100mV
Slave-CS: c: SATA, Ethernet; n.c.: RXSM	109mV
Slave-CS: c: SATA, RXSM; n.c.: Ethernet	111mV
Slave-CS: c: RXSM, Ethernet; n.c.: SATA	110mV
Slave-CS no interfaces connected	108mV
Master-SSD	4mV
Slave-SSD	4mV
Master-Camera	~10mV
Slave-Camera	~0mV
Master-PDU (without load)	2mV
Slave-PDU (without load)	2mV

Table 2-1: potential between structure & ground of power supply, only single comp. run

4 connected
5 not connected

2.2.5 Discussion of the Issue

Although apparently the core systems could be identified to cause the shifted electrical ground, it is not understood why it occurs despite the fact that the electrical design was conceived according to the REXUS user manual and all components including their cases should have been grounded via the appropriate lines of the interface cable to RXSM and furthermore why shortening the potential difference via the structure caused software crashes. This is due both to the lack of experience and knowledge of the team regarding electrical engineering and the complex electronics especially of the core system microITX-boards.

The most probable explanation is that the manufacturing accuracy of the custom-made heatsinks for the core systems (cf. 3.2.2) compared to the ones produced by the board's manufacturer, respectively their conductivity (bare aluminum vs. eloxed aluminum) causes some pins of the board setting a potential to the structure and that the stray currents cause some low-level hardware interrupts to fire and thus crash the flight software.

However, as the proposed solution (cf. 2.2.6) is very basic this lack of fully understanding the issue is not expected to impair the solution.

2.2.6 Solution of the Issue

As the issue was in fact already detected during ITW and analyzed with basic methods during Bench Test but could not be resolved completely, the team came up with two approaches of workaround solutions for the campaign to reduce the impact of the issue on the overall performance of the experiment and to fight its "symptoms".

Firstly, the master core system was replaced with a spare part as with that spare-board the issue could not be reproduced in the meanwhile between Bench Test and the campaign. As the post-flight analysis showed the potential difference to the ground of this spare-board was lower than of the Slave. Thus, probably the overall shift of the ground was reduced to a lesser harmful level, as the issue was still, but less often, experienced with the replaced master core system.

Secondly, as the shifted ground only caused the flight software but not the whole operating system to crash a software-watchdog on a lower application level (only a basic shell script) was implemented to restart the complex flight-software if it crashed (cf. 3.4.2)

With those two measures it was guaranteed that the overall performance of the experiment would not be reduced due to the shifted ground although in fact they did not face the basic issue unlike the next outlined solution. However, that solution could not be carried out before flight as it would have introduced too many changes of the Flight Segment.

The ground shifting causing harmful stray currents could simply be prevented by mounting all components (or at least the core systems) electrically insulated (e.g. with ceramic screws and varnishing the cases) to the bulkhead and explicitly grounding the cases via the power-cables. Hence, any potential put on the case by any component could not only flow off directly but also stayed localized to the single component and would not cause malfunctions of others due to uncontrolled stray currents.

2.3 Interferences with the GPS-System of the RXSM

2.3.1 Setup of the Rocket-Payload in General

The whole payload of REXUS 16 as it was intended to fly was fully assembled for the Hot Flight Simulation. This includes all experiment modules as well as the Recovery Module, the Service Module and the nosecone being mechanically mounted and all cabling connected and put through the cable-feed-through at 180°. Especially the GPS-antenna located in the nosecone was operated so close to the HORACE Flight Segment and the antenna-cable was put through said cable-feed-through for the very first time.

There were no other changes regarding HORACE, both hardware and software, introduced in this test compared to the other previous successful interference tests.

2.3.2 Expected Behavior

As all previous interference tests, like the Bench Test during campaign preparation with the same setup but not mechanically fully assembled, were successful without any interferences occurring, the same excellent behavior was expected during the Hot Flight Simulation.

2.3.3 Observed Behavior

Totally unexpectedly a reproducible interference with the GPS-system of RXSM occurred as soon as the experiments were powered on at T-600s of the timeline which induced such loud noise to all channels of the GPS-signal that the GPS-fix was lost and the position could not be determined any more.

2.3.4 Analysis of the Issue

The troubleshooting, conducted right after the test to identify the source of the interference by MORABA-staff but without the team's direct supervision (which presumably lead to the SD-card issue, cf. 2.1), showed that of the four experiments HORACE was the one which caused the interference.

During further investigation back home after the launch campaign the problem could be reproduced using an own GPS-receiver placed approximately as far from the experiment as in the rocket's actual configuration with the following results:

By removing (i.e. not to power) more and more components from the setup the SATA-connection between the SSDs and the core systems could be identified as the source of the interference.

Shielding all parts being involved in this connection (both SSD and CS, as well as the cabling) with a tin can eliminated the interference, whereas the induced noise remained loud with only partial shielding (e.g. SSD and CS in separate tin cans, cabling unshielded).

When signals of many GPS-satellites could be received with good quality, the GPS-fix remained stable despite of the interference.

Using the SSD not with the CS but with a laptop via an USB-to-SATA-adapter caused the same interferences, whereas there were no problems using the CS with a common SATA-HDD.

Grounding the SSD as described in (Mp3Car.com Inc., 2010) had no effects.

2.3.5 Discussion of the Issue

As some, even if only few, customers of other SSDs reported about similar issues (Mp3Car.com Inc., 2010) and as the interference did not occur with a common SATA-HDD, it is most probable that the SSD itself is the source of the noise in the GPS-frequencies. However, it cannot be explained why shielding both of CS and SSD but not the cable still caused interferences.

Because of both little knowledge within the team about high-frequency-electronics and no special high-frequency-equipment being available for the team, this conclusion drawn from the outlined very basic analysis may be questionable and shall be verified by experts and with better equipment before taking measures.

2.3.6 Solution of the Issue

With having identified the SSDs as the most probable source of interference, replacing them by another model would be the easiest measure to face the issue. However, as the conclusion this measure is based on is quite uncertain and other models obviously causing similar problems, it remains uncertain if that indeed solved the issue. Therefore, if this option is chosen, it shall be tested very carefully and earlier than just during campaign short before the scheduled launch.

The very basic measures taken during the campaign itself – placing ferrite cores around several cables hoping to thus reduce the interference – were absolutely ineffective.

Therefore the only effective measure would be to shield CS and SSD, including the cabling within the same case. To improve the shielding, either tin instead of aluminum shall be used for the case or special varnish for electromagnetic shielding (containing ferrites) shall be applied. But as placing both CS and SSD in the same box would affect not only the mechanical design but also the accessibility of the SSDs, which is important for operations, taking this measure should be thoroughly considered as well.

2.4 Overexposure of the Payload-Cameras

2.4.1 Setup of Cameras & Core Systems

Both cameras were run with the settings uploaded with “wxPropView(x64)” the manufacturer’s utility software for the camera during ITW regarding the appropriate procedure (Bergmann & Rapp, 2014). Those settings, having been determined to be the best suitable ones during Integration and exported to a XML-file with the same utility-software (called “default-settings” in the following), were tested at component level with high performance (outdoors on a sunny day) before ITW and with lower performance (indoors, artificial light sources) both on component and full system level before ITW as well as during ITW. After ITW the camera settings were never touched again – neither changed, nor rechecked.

Right at the beginning of the launch campaign a software update was installed on both core systems. However, as it did not introduce any changes regarding the usage of and communication with the cameras, it was equivalent to the successfully tested version, using the cameras via the “device specific” software interface.

2.4.2 Expected Behavior & Performance

It was expected that the cameras of both Master and Slave would provide not only valid but well exposed (as auto-adjustment of exposure time was enabled in the default-settings) image-data to the core systems, which would record them from T-60s to T+590s and pass it on to the algorithm for horizon acquisition during flight. Furthermore, it was expected that at least after the yoyo-despin of the rocket the horizon acquisition would be successful.

That there will possibly be problems with the exposure of the cameras during flight, when they alternately watch the dark space and bright earth and sun due to the rocket's rotation, was a known risk. Therefore, it was tested with the best effort possible for the team but no problems, both on full system and component level, were revealed. In contrast, there should not have been any problems before lift-off – as the cameras should have had enough time to adjust the exposure time automatically while standing still on the launcher.

2.4.3 Observed Behavior & Performance

During flight no horizon could be detected, neither by the Master nor the Slave, although all systems worked in the nominal way and the video-footage of RXSM, which was downlinked live, showed that the horizon should be clearly visible. Thus, already at the end of the flight operations an issue with the payload-cameras of HORACE was expected. This turned out to be right, as the stored video-data both of the Master and Slave, which was accessible only after recovery, showed that all data was valid video-frames, but most of them were completely overexposed during flight as well as when the rocket was still on the launcher before lift-off. So, the frames were either completely white – when looking into the sun or down on earth –, completely black – when looking to the space –, or showed some blue color gradients – when the horizon should have been perfectly visible. Of course, with this raw-data no horizon acquisition was possible for the algorithm.

Especially the fact that the overexposure already occurred before lift-off led to the hypothesis that the auto- adjustment of the exposure time did not work at all or not sufficiently.

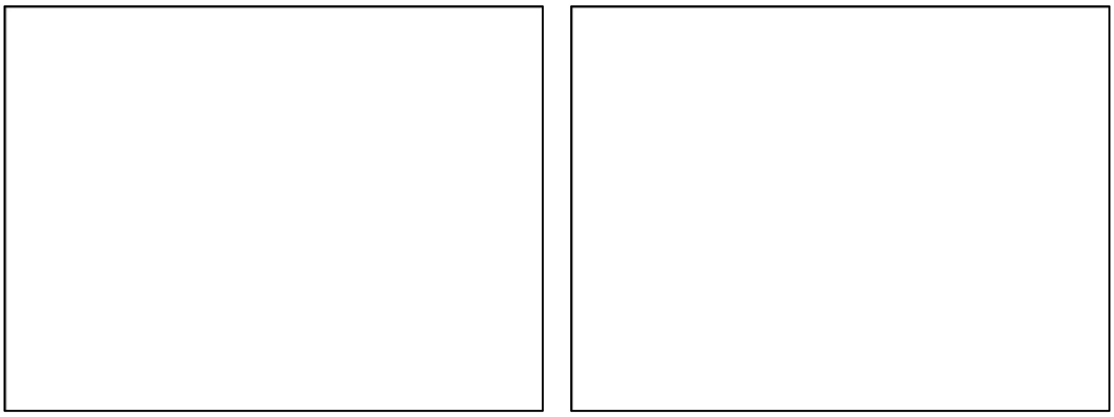

Figure 2-1: example images, T-50s; left: Master; right: Slave

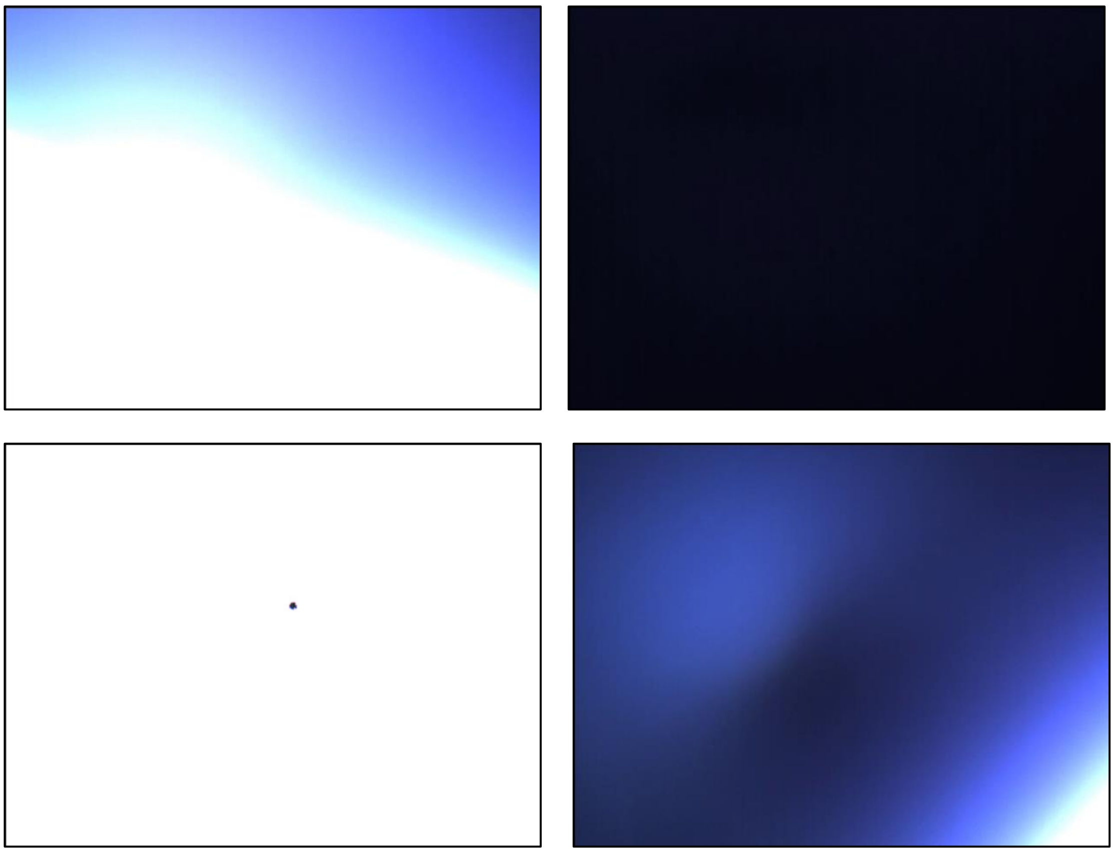

Figure 2-2: example images Master; clockwise from upper left: T+46s, T+79s, T+87s, T+97s

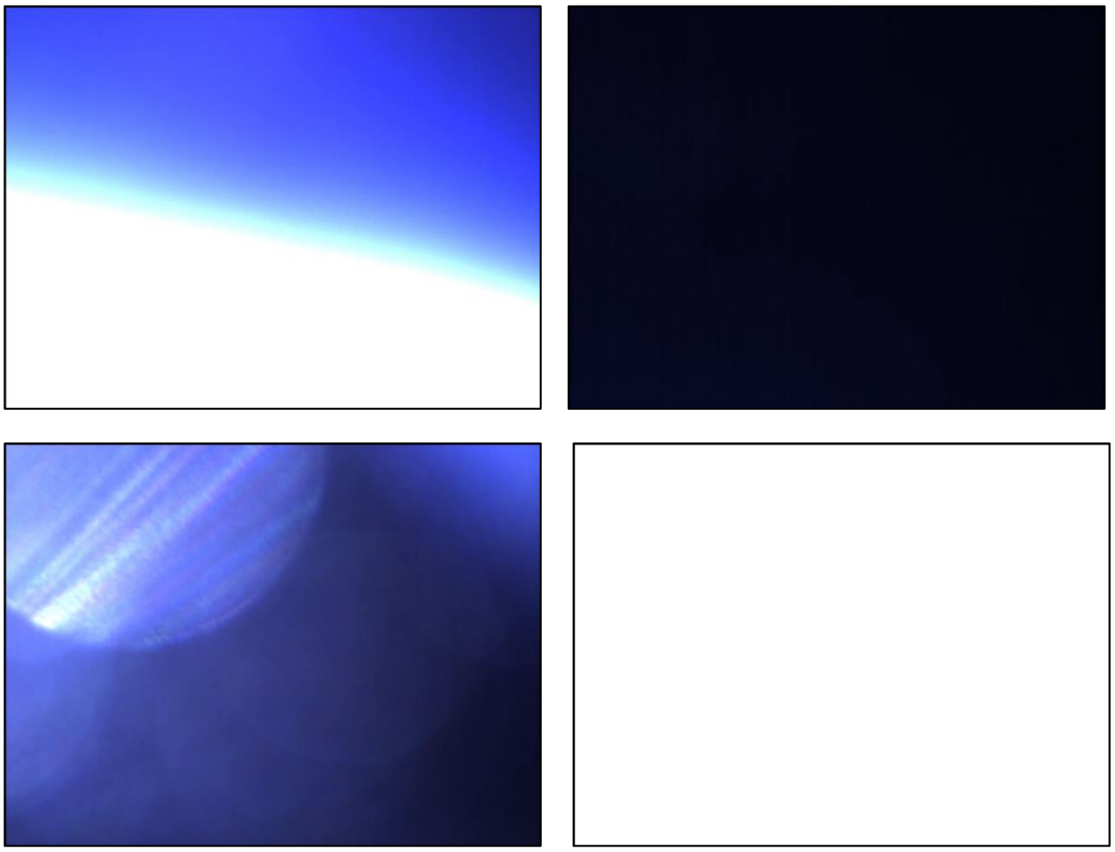

Figure 2-3: example images Slave; clockwise from upper left: T+57s, T+74s, T+87s, T+90s

2.4.4 Analysis of the Issue

The thorough analysis of this issue conducted during the post-flight evaluation back in Würzburg showed that in fact there was a problem with the auto-adjustment of the exposure time of the cameras. Although indoors with the same full system setup as flown the issue could not be reproduced, even looking out of the window (again with the same setup) on a sunny day or being outside caused at least partial or total overexposure of the images. Auto-adjustment of the exposure time did not work at all for that setup.

The spare camera, which was not flown, provided perfect images both in- and outdoors while it was not operated by a core system but with an EGSE-laptop and wxPropView with the camera settings "as they were". Moreover, even while looking directly into the sun color gradients and shapes were still visible. After having reimported the default-settings and uploaded to the spare camera (again with wxPropView), also the spare camera only provided overexposed images when being outdoors used with wxPropView.

Then the settings of all three cameras (Master, Slave, spare), which were currently set, were downloaded and the corresponding XML-files were compared line by line (with standard tools for difference-check). The comparison showed that in fact all settings were partly different and did not match the default-settings-file – even the settings of the spare camera, which was only programmed with the default-settings some steps of the analysis before. However, the entries which seemed to be relevant were identical: "auto-exposure" was enabled, the "area of interest" for auto-exposure (which part of the image shall be exposed best) set to "full" and the "auto average gray" value (desired average brightness of the auto-exposed image) set to reasonable 57%.

Additionally, it was discovered that outdated releases of wxPropView, which were shipped with the cameras, were used both on the EGSE-laptops (release 2.5.17 from Nov 12 2013) and the team's PC in the lab (updated on Jan 07 2014 from 2.5.17 to 2.5.18 released on Dec 12 2013) although a new version is released nearly every month. The default-settings were created with the lab PC (v2.5.18), whereas it was uploaded with the EGSE-laptops (v.2.5.17) during ITW. As the manufacturer's change-log of the software does not say anything about importing or exporting XML-files or uploading or downloading settings to the cameras, this difference of versions shall not be relevant (e.g. by introducing incompabilities) (Matrix Vision GmbH, 2014).

In contrast the entry in the change-log for version 2.9.0 released June 26 2014 – which was nearly one month after flight – says in the bug-fix section that "[l]oad[ing] settings from XML file now works properly" (Matrix Vision GmbH, 2014), implying that it did not in previous releases. Because this bug-fix – unlike others – does not tell which version introduced the bug, it has to be presumed that importing XML-files does not work correctly with the used versions (v.2.5.17 and v2.5.18). Thus, to determine whether the bug of wxPropView was the single reason for the problem, the latest release (v2.9.2) was installed to the lab PC, all settings manually set and exported to a XML-file. This file was not only drastically smaller in size than the other ones (created with v2.5.17/18), but also the relevant entries (auto-exposure enabled, area of interest for auto-exposure, auto average gray) were missing completely, while the problem of overexposure remained.

To make things even worse and less transparent, there are two software-interfaces/modes to use the camera with: “device specific” and “GenICam”. “device specific” is the manufacturer’s custom interface to that specific camera-model. In contrast, “GenICam” is a generic, standardized, but therefore also much more complex, interface used by various manufacturers for many cameras. wxPropView uses the “GenICam”-interface for customizing the camera settings, whereas in the flight-software of the core systems the cameras are used with the “device specific”-mode. Obviously, as unfortunately only discovered during this analysis, both modes share some of the camera settings, so that changing them via wxPropView with “GenICam” applies also to the “device specific” mode, but others – among the relevant ones for auto-adjustment of exposure time – are not shared. This means setting them via wxPropView with “GenICam” does not affect the same parameters of the “device specific” mode.

2.4.5 Discussion of the Issue

To summarize, quite a cascade of mistakes and failures led to the fatal issue of the overexposure of the cameras.

First of all, there was the absolute non-transparency and partial malfunction of wxPropView. Not only the import and export function for XML-files obviously did not work correctly, which was not sufficiently given warning of, but also the fixed version is probably not intended to import/export the relevant settings (as the entries are missing in the XML-files). Therefore, the complete procedure of uploading settings to the cameras, the team worked with, was obsolete and most probably did not even affect the relevant settings. Instead of importing settings-files, apparently the settings should have been manually adjusted for each camera and uploaded directly. Additionally, the fact that the two different modes to use the camera share some settings caused the team to assume that all settings apply for both modes and not to worry about uploading them in another mode than actually using the camera within the Flight Segment. That turned out to be a false assumption, which is again not completely the team’s mistake as it was not documented precisely and unambiguously enough.

Secondly, there is the insufficient testing. The tests which would have been sufficient (e.g. outdoors on a sunny day) were only performed at component level, that means with wxPropView, and, following the same false assumption as mentioned above, were not repeated on full system level – neglecting the fact that regarding the overall project schedule further full system tests would have only been possible during winter, when the low-cost option (testing outdoors on a sunny day) would not have been available, as well as there were no other sufficient facilities available for the team. Furthermore, the settings with fixed exposure time unfortunately happened to be good enough for the full system tests (indoors with less powerful light-sources), which obviously were not sufficient although performed with best possible effort.

2.4.6 Solution of the Issue

There are mainly two possibilities to face this issue. First, the flight software of the core system could be changed, using the camera in “GenICam”-mode instead of “device specific”. Thus the settings uploaded via wxPropView are valid also for the core system. But as the usage of the “GenICam”-interface is quite complex – the reason why it was not used in the first place – and one still has to rely on the functionality of wxPropView, which still might be risky, the second possibility is recommended: Via the “device specific”-interface all settings can be applied by any C++-program quite easily (only two lines of code for each setting). Therefore, it is possible to simply set all needed settings directly with the flight software of the core systems during the boot- and initialization-procedures, which was already tested successfully. The main disadvantage – and again the main reason why this possibility was not taken in the first place – is that with this solution any changes of the camera settings to be done would also affect the flight software of the core systems (in worst case requiring recompilation and a complete software-update), thus reducing the modularity and increasing the complexity of the experiment. Implementing the possibility to apply or change the settings via telecommand (additionally to autonomously loading some default-settings during initialization) would on the one hand totally ban the need of updates of the flight software, but on the other hand increase complexity even further – not only from a technical but also an operational point of view. Furthermore, as it is unlikely to have to change the settings when the best are found once, the impact of the disadvantage of software updates is quite small and possibly would not justify implementing the telecommands.

Additionally to this direct measure to face the issue, it is strongly recommended to improve the test procedures and performance of available facilities for the full system tests regarding the brightness of light sources. Possibly external partners with adequate expertise and facilities have to be found.

As another indirect measure, which probably could have prevented the failure at least in the last nick of time, it is suggested to implement the possibility to downlink some single video-frames via telemetry during stand-by-mode. Thus, at the latest during the hot countdown – if not even in one of the numerous test more or less before –, and not only after flight, the issue would have been recognized with the possibility to abort the countdown and fix the problem. Of course, this also increases the overall complexity, but as on the other hand so far the only way to access the image data requires direct access to the SSDs and is a quite lengthy procedure – and therefore was not performed during Bench Test or campaign preparation –, this trade-off seems to be fair.

3 ANALYSIS & DISCUSSION OF MAIN ACHIEVEMENTS

In this chapter the main achievements of HORACE both during flight and the overall project are thoroughly analyzed and discussed in an approach similar to the analysis and discussion of the main problems in chapter 2. Firstly, the requirements and constraints which are relevant for the treated subsystems are summarized and explained. Where applicable also the IDs of relevant requirements referring to (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 13-20) are given. Secondly, the concept for the design and the main steps of its implementation are outlined, focusing on the reasons for design decisions as well as on the “evolution” of the design and not the final design itself as the latter is already described in detail in the Student Experiment Documentation (especially chapter 4) (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014). In a last step the results, respectively the actual performance, are presented and discussed as well as evaluated regarding the relevant requirements.

3.1 Electronics & Electrical Interfaces

3.1.1 Relevant Requirements & Constraints

Although the focus of HORACE was on the software for the horizon acquisition, of course some hardware was needed to execute the software on. Proper communication infrastructure and a power system were also necessary. From those basic functional requirements for the electronics, a lot of performance and design requirements are derived – most of them depending on the algorithm’s performance (e.g. for data-transfer-speed or data storage size), the electronics itself (e.g. for the output-voltages and currents the PDU must provide) or the constraint of being a payload on REXUS 16 (e.g. regarding the interfaces or environment).

The significant lack of experience in electrical engineering, the limited financial budget as well as the relatively short project-life-cycle were additional severe constraints on the electrical design.

3.1.2 Design & Implementation

To face the team constellation with only little experience in electrical engineering and to keep costs low and development time short, it was decided already in a very early stage to use components off-the-shelf and standardized interfaces wherever possible. Whenever that was only partly possible, for example in the case of the PDU and the Arduino-shield, things were kept as simple as possible. This decision severely influenced the whole experiment design and development as the number of suitable models being available for each of the main components was quite limited and the requirements for intra-experimental interfaces introduced several dependencies and thus narrowed the choice even further.

As expected, the development and manufacturing of the PDU needed several iterations of prototyping and nearly complete redesign to finally have a highly efficient PDU with a fully functional data- and signal-interface between CS and RXSM. The team even had to make use of supervision by ZARM for the dimensioning of the LC-circuits to reduce the feedback-ripple to the required level. With this experience and today’s point of view, the decision, not to use own auxiliary power units but to draw all needed power from RXSM, seems to have been fair. This was a crucial decision as the PDR-panel actually requested to implement auxiliary power units as the high and conservative preliminary calculations for the power budget of HORACE potentially would have exceeded the total available power provided by RXSM. To keep simplicity the team instead offered to forego the chance to fly two systems (Rapp, et al., HORACE Student Experiment Documentation - v2.0, 2013, pp. 30-31).

The power budget was reduced drastically from worst case 95.66W (respectively 16Wh for the complete flight, incl. 50% margin; (Rapp, et al., HORACE Student Experiment Documentation - v1.0, 2013, p. 34)) after PDR to worst case 67.16W (11.2Wh, incl. 50% margin; (Rapp, et al., HORACE Student Experiment Documentation - v1.1, 2013, p. 38)) due to various design changes. Additionally, it would have been possible to deploy a low-power mode for the slave system. Eventually, with the complete setup on the table it was even lower (49.1W, respectively 8.2Wh, measured, incl. 50% margin; (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, p. 96)). Therefore, EuroLaunch approved to fly two systems although no auxiliary power units were implemented for HORACE.

3.1.3 Discussion of the Performance during Flight

The actual power consumption, nominally regulated and distributed during flight by the PDUs and logged by RXSM, was for both systems nearly constant. The slave system drew 0.5352A from the unregulated power (28-32V), which equals a total power consumption of 2.74Wh, whereas the master system consumed a bit more (0.5315A, 2.76Wh) as the MU was also powered via the Master-PDU (cf. Figure 3-1).

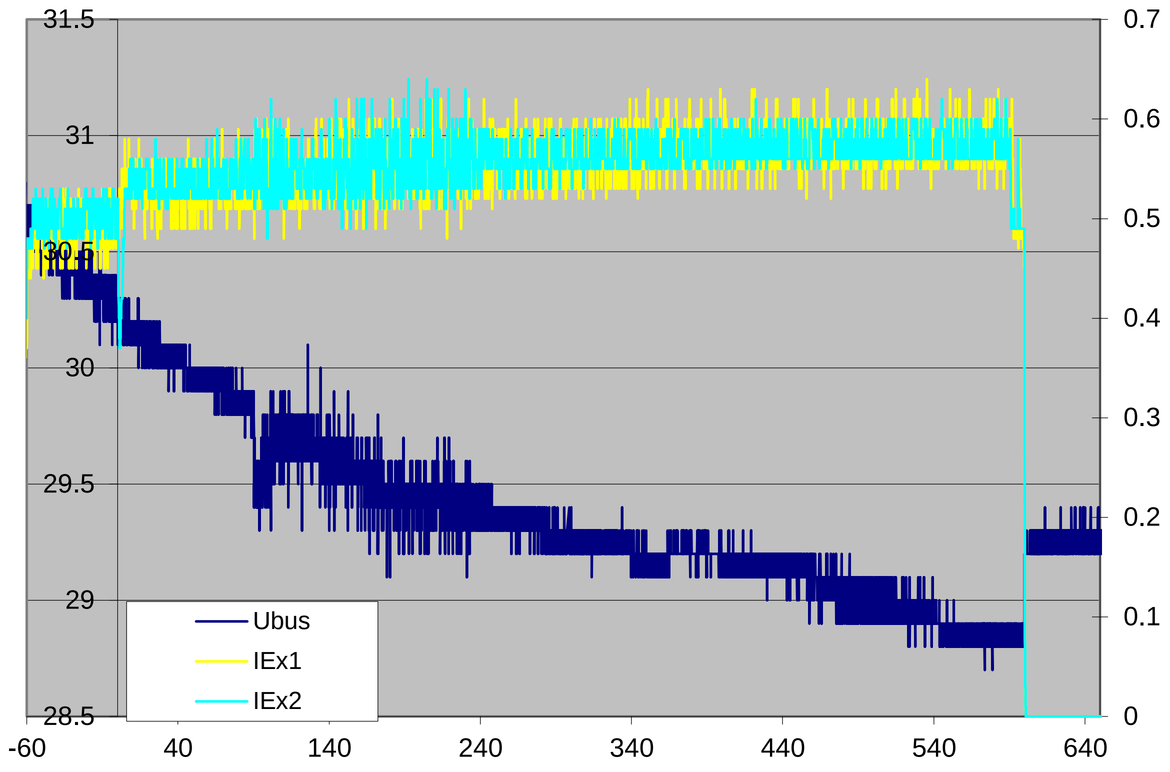

Figure 3-1: HORACE total power consumption recorded by RXSM; Ubus - voltage of the provided power supply [V], IEx1 – current Master [A], IEx2 – current Slave [A]

The measurements of currents, collected during flight by the MU with an average sample time of 46ms and 10th-degree average filter (thus equivalent to average 210Hz of single samples) for each CS, also shows nearly constant power consumption (cf. Figure 3-2). The Master-CS drew 0.77A @12V (1.56Wh), which is more than half of the total power consumption, whereas the Slave-CS, according to the logged data, averagely consumed 0.89A (1.82Wh).

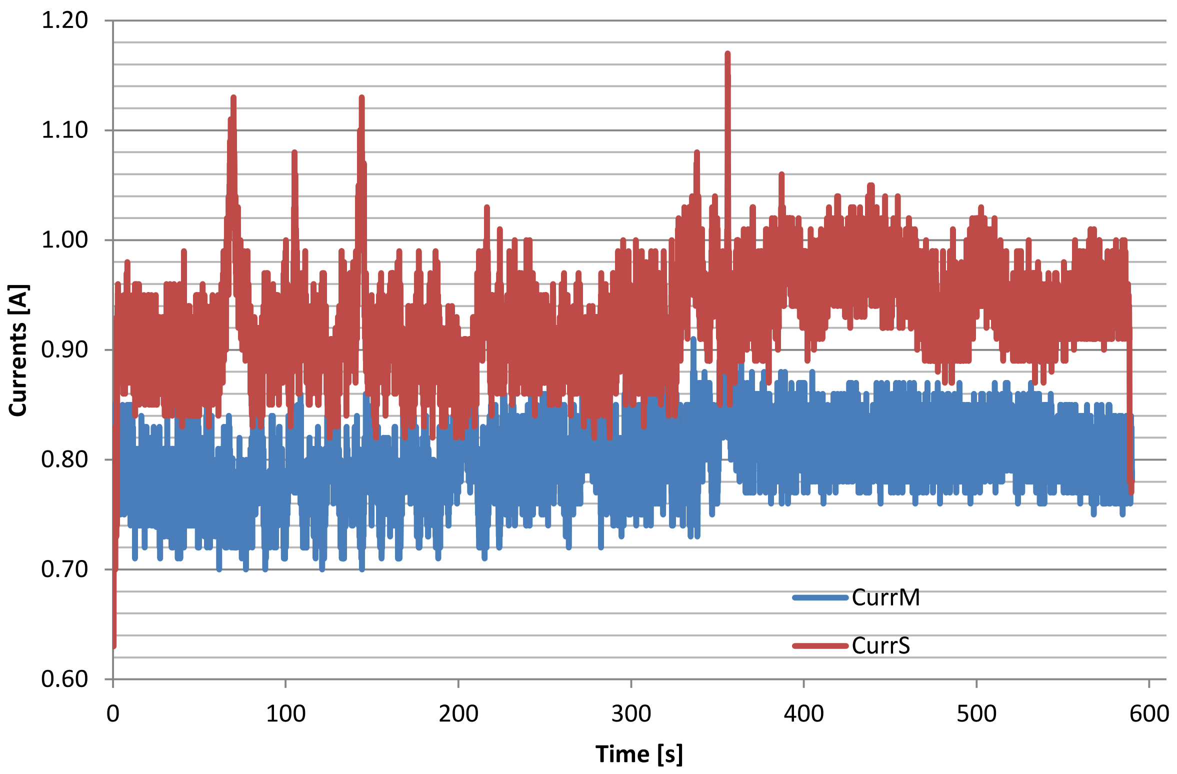

Figure 3-2: power consumption core systems; CurrM - Master, CurrS - Slave

This data clearly shows no correlation between the actual algorithmic workload for the horizon acquisition and the power consumption(cf. 1.3 – 2nd primary objective), which is mainly caused by the core systems working to full capacity the whole time due to the massive overhead (e.g. operating system, complete experiment control, TM/TC, cf. 3.4.3). Moreover, the question whether horizon acquisition for attitude determination is also applicable for small satellites (cf. 1.3 – 2nd secondary objective) can only partly be answered: The given power consumption is above the capabilities of small satellites but exceeds them not too much although the flown experiment was not optimized for low power consumption at all. Hence, the chances that a future operational sensor-system is applicable for small satellites regarding the power consumption are good.

The discrepancy between the power consumption of the core systems – both had nearly the same workload and therefore are supposed to consume the same power budget – can be explained by a static error which was caused by a static magnetic field within the rocket falsifying the measurements of the sensitive analog hall-sensor. It was neither possible to calibrate the sensor before flight, as the error only occurred while the rocket was fully assembled, nor could it be reproduced afterwards to determine the exact value of the offset. Also the louder noise, which exceeds the required accuracy of ±100mA, and the clearly visible peaks, are supposed to be caused by external magnetic fields. This is plausible as the cable between the analog sensor and the ADC of the MU led through the same cable feedthrough as all cables of the rocket (experiment power, experiment data-interfaces, GPS etc.) and the peaks can be correlated with some events causing high currents (switching motors, pyro-cutter-events) in those cables, inducing dynamic magnetic fields and thus inducing voltages in the sensor-cable.

Also regarding the electrical interfaces, it was a good decision to “keep it simple” and to only use standard interfaces, and as well only polarized standard connectors, as all interfaces worked perfectly nominal and the risk of major damage of components due to changed polarity was banned.

Except from the issues of the overexposure of the payload-cameras (cf. 2.4; P-E-10, P-E-12), the interference with GPS (cf. 2.3; D-E-01, D-E-02) and the shifted Ground (cf. 2.2), which caused the appropriate requirements not to be met during flight, all other requirements for the electronics were perfectly met, thus creating a mostly functional electrical platform for the actual experiment.

3.2 Thermal Design

3.2.1 Relevant Requirements & Constraints

Actually, for the performance of HORACE, i.e. derived from the Mission Statement, there are no direct requirements on thermal aspects, but as during the preflight preparations and during flight itself the experiment may be exposed to harsh temperature conditions from -30°C up to +50°C within the module (Mawn, Kaczmarczik, & Schmidt, 2014, p. 60), a proper thermal design was necessary to guarantee the full functionality of all components and the experiment as a whole. This is reflected in D-M-04 and D-M-05 for the most critical parts, which are directly derived from the constraint of being payload on REXUS.

3.2.2 Design & Implementation

Especially the components off-the-shelf, like the camera or the embedded PC, selected due to the design decision explained in 3.1.2, led to discrepancies between the expected temperature profile and the temperature ranges specified in the datasheets, while for other even more common components (SD-card, RAM) no temperature ranges were specified at all. The discrepancies of low temperatures during start up as well as high temperatures during the vacuum of flight were considered most critical, but manageable. Extremely low temperatures during start up might be harmful when electronic parts fail to power on due to the low temperatures, whereas once running the situation would improve due to the power consumption and heat dissipation of the components itself. High temperatures, even within the specified ranges, during flight may lead to partial overheating as in vacuum passive cooling is possible only via the structure and not by convection. Other discrepancies were not considered to be critical as they either would not affect the experiment’s performance or could be faced quite easily, for example by properly packaging and insulation during shipment (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 56-58).

The discrepancies of low temperatures during start up were shown to be uncritical by testing. In a combined thermal test all components were cooled to -40°C and successfully powered on several times showing full functionality (Scharf, Test #2.5 Report, 2013).

Before testing the discrepancies of high temperatures in vacuum conditions potential hotspots within the experiment were identified with a basic method: Based on the relative heat dissipation of all components while both idling and under full stress with ambient temperature of 21°C and normal pressure, the heat distribution within the module was numerically simulated for vacuum conditions using special software packages. The heat caused by friction during reentry on the outer surface was neglected as the holes for the cameras were closed by protective windows and as another numerical simulation showed that the heat would barely transfer to the components because they were thermally decoupled from the module itself. The core systems and PDUs turned out to be hotspots and thus, it was decided to use heatsinks, which are thermally coupled to the rest of the structure to dissipate as much heat as possible, to passively cool those parts (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 58-59). Whereas for the PDU the heatsinks had to be designed from scratch, originally it was planned to use standard heatsinks provided by the manufacturer of the embedded PCs for the core systems. However, thermal vacuum tests unveiled that the standard heatsinks are not suitable for vacuum conditions, as under full stress with convection missing the heat dissipated by the CPU causes overheating of the nearby southbridge, which withstands only much lower temperatures than the CPU (Scharf, Test #1.2 Report, 2013).Therefore, a modified heatsink for the embedded PCs with a groove, which thermally decouples CPU and southbridge, was designed.

3.2.3 Discussion of the Performance during Flight

The evaluation of the temperature data, collected at six points of the Flight Segment (camera, PDU, core system for each setup) with a sample-rate of 1Hz and an accuracy of ±0.5°C and logged by the MU throughout the flight, proves that the thermal design was both sufficiently and reasonably conceived and dimensioned. As seen in Figure 3-3 all temperatures were perfectly nominal within the ranges specified in the components' datasheets although the maximum temperatures (between 52°C and 57°C) are a bit higher than expected (50°C). This difference is simply caused by the ambient temperature (ca. 15°C) during the countdown being much higher than expected (0°C or below) as the launch campaign took place not until end of May although it was originally scheduled for February or March like all previous campaigns.

The temperature of the master core system was constantly 0°C with only a single peak to 85°C (both not in scale) as the electrical connection to the sensor was lost during preflight preparations and no time was left to replace the sensor. Probably the connection was unexpectedly regained during flight for a short moment (less than 2sec) as 85°C is the nominal return of the sensor when reading the temperature during the sensor-initialization, thus causing the peak in a single sample.

All other temperatures drastically increase from around T+20s on, as until this time the pressure logged by the REXUS Service Module reached its minimum and so cooling of the components by convection was no longer possible.

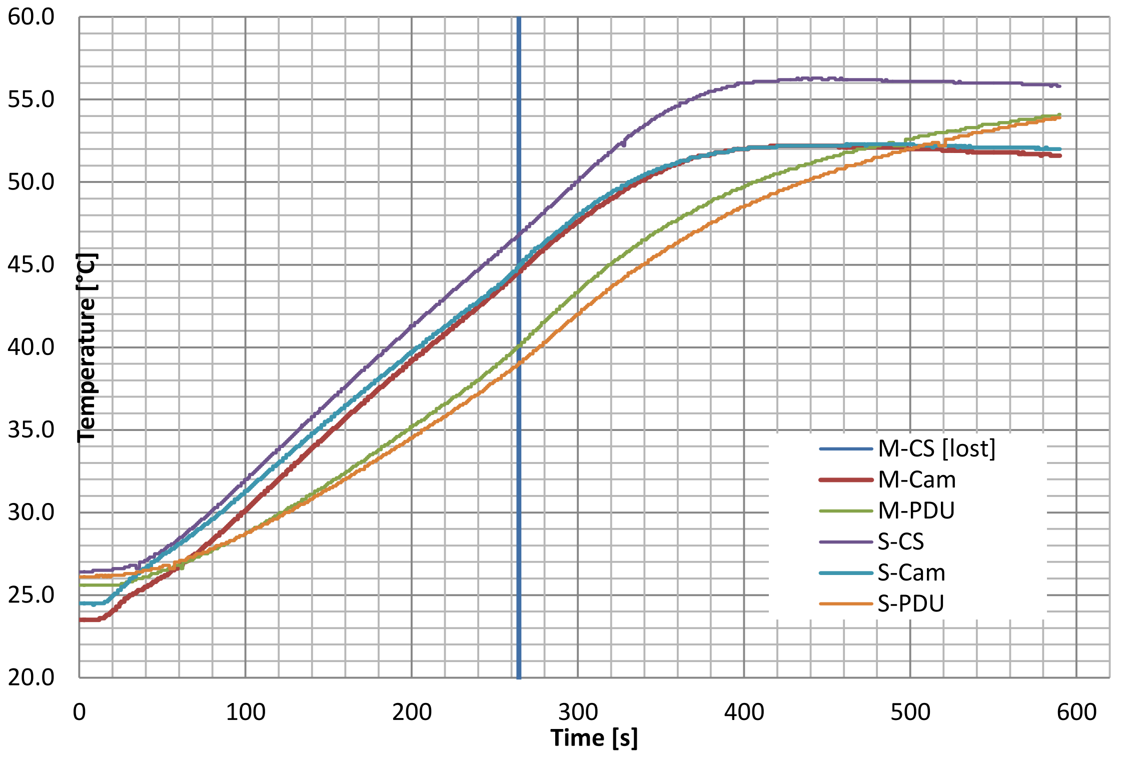

Figure 3-3: MU-data: temperature vs time

According to the RXSM-data the module was pressurized again with 0.5bar around T+350s and hot air (estimated ca. 50°C) rushed in. Thus, cooling by convection – but due to the little temperature difference between the air and the components only little, if any – was possible again causing the temperatures of both cameras and the Slave-CS to settle around T+400s and to even decrease a bit. In contrast, the temperatures of both PDUs kept increasing and only started to settle around T+600s. This can be explained as the PDUs were significantly cooler than the other parts and thus unlike the other components they probably were heated by the inrushing hot air. Additionally, the heatsinks of the PDUs are coupled to the main structure only indirectly and have only small surface itself. Hence, the cooling both by convection and dissipation was worse than for the other components.

All in all, with the conceived thermal design, a perfect – sufficiently for the harsh temperature conditions but also not over-dimensioned and thus not causing an unnecessary increase of weight of the Flight Segment – thermal platform was created for the experiment.

3.3 Mechanical Design

3.3.1 Relevant Requirements & Constraints

Besides the only functional requirement for the mechanical design – to ensure the visibility of the horizon during flight – and the derived performance and design requirements (e.g. specific percentages of visibility, to mount the cameras in certain ways or to use two cameras), like for the thermal design (cf. 3.2) the main part of the mechanical requirements was based on the constraint of being payload of REXUS 16. Therefore, the experiment shall not harm or interfere neither with the vehicle, nor the launcher, nor the Service System, nor other experiments. Furthermore the Flight Segment had to fit to the defined mechanical interface and had to survive the expected mechanical stress (vibrations, pressure difference, g-forces etc., (Mawn, Kaczmarczik, & Schmidt, 2014, p. 38)) to guarantee a fully functional experiment and protect sensible electronic parts properly during flight. From the operational point of view easy access both to connectors and data storage devices was a soft requirement.

Another constraint put on the team was, like for the electrical design (cf. 3.1.1), the lack of knowledge and experience about mechanical engineering especially CAD-drawings as well as only limited possibilities for manufacture (e.g. no access to the services of the University's workshop).

3.3.2 Design & Implementation

To face the lack of experience and knowledge a new team member, Matthias Bergmann, was recruited shortly after PDR, which was strongly recommended by the PDR-panel. Matthias had also little experience, as because of the lack of proper engineering faculties at the University of Würzburg nearly everyone. But as explained in 3.3.1 the requirements for the mechanical design were quite basic and as for the beginning it was his only main responsibility he had enough time to read up on mechanics and the used CAD-software.

For the preliminary design the components were placed on two bulkheads – one bottom-mounted and one wall-mounted – and several PCBs were not even boxed. Those two aspects of the mechanical design were identified as improvable during PDR. Therefore and caused by changes of the electrical design (cf. 3.1.2) the mechanical design was nearly completely changed between PDR and CDR. Instead of two bulkheads, one single standard bulkhead was wall-mounted approximately in the middle of the module with the standard brackets. This bulkead was then populated with the components from both sides in a highly symmetrical manner, to utilize the volume of the only 120mm high module best and to keep the center of gravity close to the rockets roll-axis. Moreover, all electronic parts which were not properly protected by own housings were placed into aluminum boxes. Those boxes were mainly meant to protect the electronic components from possibly loosened parts like screws floating through the rocket during micro-gravity causing shorts on PCBs. The boxes were, unlike few other parts which required high manufacturing accuracy, manufactured by the team at one team-member's personal home workshop to reduce costs (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 47-48).

As it was expected that during descent phase hot gases may rush into the module through the camera holes, if they were not closed and as tests have shown that the cameras themselves would probably not withstand those temperatures, with a very late design change (right after CDR) the holes were closed with protective windows although it was tried to avoid windows in the first place due to their possible optical impact on the captured images (Rapp, et al., RID Report - Post-CDR: Thermal Protection of Optical System, 2013).

Despite all efforts the final design exceeded the required maximal total height within the module (a gap of 10mm is required to avoid mechanical interferences) as eight nuts reached into the forbidden zone by 4mm each. However, that was no severe issue as HORACE was on top of the payload stack without another expe-riment in the nosecone thus there was nothing to interfere with and an exception for the required height was approved by EuroLaunch (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, p. 34).

Because the vibration tests performed during ITW with the complete experiment setup were perfectly successful without any screw being loosened, in contrast to the original plan it was decided not to lock the screws with locktite as that would have complicated the access to the components at a late stage (as indeed it was needed due to the SD-card failure, cf. 2.1.4) or after recovery.

3.3.3 Discussion of the Performance during Flight

The flight proved the mechanical design to be perfectly suitable for its basic function of protecting the experiment from the high mechanical stress. The experiment returned in an excellent state without any component being broken or any loosened screw.

Because the exception for the interference requirement was approved long prior to launch, also all requirements regarding mechanical interfaces and interferences are met.

As far as one could tell from the bad image data and as calculations were made in advance (Scharf, Analysis Report, 2014) also the mechanical requirements regarding the visibility of the horizon are considered to be met.

3.4 OBDH-Software of the Core System

3.4.1 Relevant Requirements & Constraints

In fact, there were several mostly independent functional requirements which were meant to be fulfilled by the OBDH-Software of the core systems. Besides the execution of the horizon acquisition itself (which is discussed in detail in 3.5) the core system software was in charge of saving all scientific data and implementing the data interfaces to RXSM both for up- and downlink as well as the data inter-face to the MU. Furthermore, for operational reasons as an uplink was not available during flight, the CS-software had to autonomously control the whole experiment and its proper conduct during flight and partly also during countdown.

The main constraint put on the software development was the decision made for the electrical design (cf. 3.1.2) to use an off-the-shelf micro-ITX computer as core system. Especially the limitation of electrical data interfaces and the necessity to run the actual OBDH-software on top of an operating system were caused by this constraint.

3.4.2 Design & Implementation

To ensure the least possible impact of the operating system on the actual OBDH-software, the minimalistic Linux-distribution “Arch-Linux” was chosen and only needed features were activated and customized manually. Thus, the OBDH-software itself was developed as a “normal” C/C++-application for Linux and was started by a small shell-script, which was executed directly after booting the OS. The shell-script additionally implemented the functionality of a software-watchdog and would restart the flight-application after a fatal crash (as it was experienced during testing several times due to the shifted ground, cf. 2.2)

Because the different required functions, defined in (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 62-70) as software-tasks, should be kept as independent from each other as possible for security and reliability (so that errors of single tasks would not lead to a complete crash), they were implemented in four separate threads (based on POSIX threads), which only share protected buffers where necessary.

The Control-Thread initializes all other threads and the buffers for data-exchange and controls the correct proceeding of the time-line according to the software-modes (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, p. 62) by activating and deactivating the other threads. Furthermore, it performs the system-self-checks during stand-by and executes telecommands provided as package-objects (cf. 3.6.2) by the Communication-Thread in a shared RXSM-receive-buffer. Moreover, it pushes command-packages (both for telecommands and for time-sync) to the MU-send-buffer as well as interprets MU-data-packages (popped out of the MU-receive-buffer), reformats it and pushes to the RXSM-send-buffer, as it also does with own data to be downlinked (like self-check-results).

The VideoSaver-Thread fetches the image data provided by the camera via the GigE-Vision-interface, saves them to the SSDs and provides them for the Algorithm-Thread via a shared buffer.

The Algorithm-Thread is only activated during flight. It processes the image data provided by the VideoSaver, writes its results to the SSDs and provides them for downlink by pushing appropriate packages to the RXSM-send-buffer.

The Communication-Thread has control over both RS-232 interfaces, receives the uplink-data-stream from ground, respectively the downlink-data-stream from the MU, parses them and provides the data-packages in the appropriate buffers. Moreover, it builds the byte-stream of the packages which are to be sent and are provided in the buffers by other threads. This applies both for telecommands from ground and time-sync-packages, which shall be sent to the MU, and health-data from the MU, which shall be forwarded to ground. Finally, the Communication-Thread also sends the data via the RS-232-interfaces. In addition, the thread captures signals (LO, SOE, SODS), which also arrive at the RS-232-interface connected to RXSM.

Those threads are dynamically scheduled by the operating system, what caused heavy losses of the overall speed. However, as the whole software was not optimized regarding speed at all but regarding reliability due to the limited development-time, there shall be plenty possibilities of speed-optimization.

3.4.3 Discussion of the Performance during Flight

Despite experienced crashes of the master core system’s OBDH-software during testing at ITW and Bench Test, which were correlated to the shifted ground (cf. 2.2), the OBDH-software ran perfectly stable on both core systems during countdown and flight. Thus, the timeline was conducted nominally, all scientific data was stored correctly and the communication via the RS-232-interfaces was stable – meeting all functional requirements.

But the usage of an OS, its dynamic scheduling and the fact that all those functionalities were implemented in the OBDH-software of the core systems introduced so much overhead compared to the important functions (gathering and processing image data) that especially P-S-03 (required a frame-rate of processed images of 30fps) was severely missed: The frame-rate of captured images was around 10fps (Master: 9.8fps; Slave: 9.6fps) and the frame-rate of processed images was even lower (cf. 3.5.3).

On the other hand, as reliable but slow software is in case of a REXUS-experiment better than fast but unstable software and as the software was not speed-optimized, the overall performance of the OBDH-software for the core systems during flight is considered to be excellent.

3.5 Algorithm for Horizon Acquisition

3.5.1 Relevant Requirements & Constraints

The main objective of HORACE was to investigate whether horizon acquisition using image-data from an ordinary camera is applicable for attitude determination. Therefore, of course, several software requirements aiming at this objective were defined. The main functional requirement for the part of the overall software which is called “algorithm” and which is basically independent from the rest of the OBDH-software was to detect the horizon line in image data provided by a camera and whenever this step was successful to calculate from this information the position of the 2D projection of the center of the earth.

Along came several performance requirements for this functionality: The 2D-vector to the earth center was required to be accurate in the sub-pixel-scale. The system should have processed 30fps and percentages of successful horizon acquisitions depending on the quality of the processed image were set. A horizon acquisition was defined to be successful when both $0.9 \leq r/R \leq 1.1$ and $e/R \leq 0.1$ hold, where r is the calculated earth radius, R the real earth radius in the image and e the Euclidian distance between the calculated and the real earth center of the 2D projection. Those criteria as well as the percentages for successful acquisitions were set very early, even before the performance of the algorithm could be at least roughly estimated, and therefore turned out to be both unlikely to be met and also little informative to actually evaluate the algorithm’s performance.

As for the rest of the OBDH-software, the decision for the electrical design (cf. 3.1.2) to use a microITX-board with operating system as core system, which also executed the algorithm for horizon acquisition, was the main constraint being put on the algorithm-design.

3.5.2 Design & Implementation

Although in the very first place there were two possible branches to detect the horizon line – either an approach following the detection of edges and contours between parts differing in the brightness or segmentation between parts differing in the colors – only the edge detection was pushed forward, as early tests had shown that it would be the simpler to implement, faster and more efficient method. (Rapp, et al., HORACE Student Experiment Documentation - v1.1, 2013, pp. 42-43)

After preprocessing steps, which skip images because of their overall brightness (either too dark or too bright) and binarize the image with a threshold filter, outer edges (which are not encircled by other edges) are detected. Out of those outer edges the longest is assumed to be the horizon and extrapolated to a circle by a least-square method whose center and radius are the wanted results. The threshold filter is dynamically set proportional to the average brightness of the image.

If the image is processed correctly and not prematurely skipped, either the results are returned or, if the results do not match the expected ones (being out of predefined bounds due to image disturbances),– following the divide-and-conquer-principle – the image is divided into sub-images, which then are processed independently following the same method. For more detailed information about the algorithmic approach itself refer to (Barf, 2014).

The fact that with the given electrical design the software had to run on top of an operating system was in contrast to the effects on the algorithm’s performance an advantage for the development and implementation. Therefore, it was possible to use openCV, a very powerful and fast open-source library for image processing, which led to reduced development effort. Additionally, thanks to the usage of openCV it was easy to first implement a stand-alone software of the algorithm to make early testing (e.g. with simulation videos or footage of former REXUS-teams) and experimental determination of parameters (like threshold or bounds for the algorithm’s results) possible. As finally also the algorithm was meant to be one of the mostly independent threads of the core system software (cf. 3.4.2), the effort to port the stand-alone version and implement the algorithm as part of the OBDH-software stayed small as well.

3.5.3 Discussion of the Performance during Flight

Due to the overexposure of the cameras (cf. 2.4) not a single horizon was detected both by the Master and Slave during the whole flight. However, all images provided by the cameras, which were valid though containing no information for the horizon acquisition, were correctly processed, i.e. skipped, by the fully operational algorithm. Thus, both the rate of false positives (detected horizon although there was none) and the rate of false negatives (no detected horizon although horizon was visible) of both systems are 0%. Furthermore, as the performance requirements on successful horizon acquisitions are only related to images in which the horizon is indeed visible, technically they are met although no conclusion about the algorithm’s performance can be drawn.

During flight the Master processed 1719 frames in total (first processed frame #849 at T+0.599s, last processed frame #6589 at T+585.777s) resulting in an average frame-rate of processed 2.9fps. The Slave’s performance was similar: 1656 processed frames (from #812 at T+3.740s to #6423 at T+586.379s, 2.0fps in average). So the requirement of 30fps of processed frames was severely missed. Closer evaluation of the time needed for the processing of each frame, which was saved by the algorithm via two timestamps (start-of-calculation and stop-of-calculation), showed that indeed the algorithm took only between 50ms to 70ms (equivalent to 14-20fps) for most of the frames with only few outliers (cf. Figure 3-4). Of course, those calculation-times do not properly reflect the average performance of the algorithm, as nearly all images presumably were not fully processed but prematurely skipped, but that does not explain the discrepancy between the calculation-times of single frames and the average frame rate of processed images during flight.

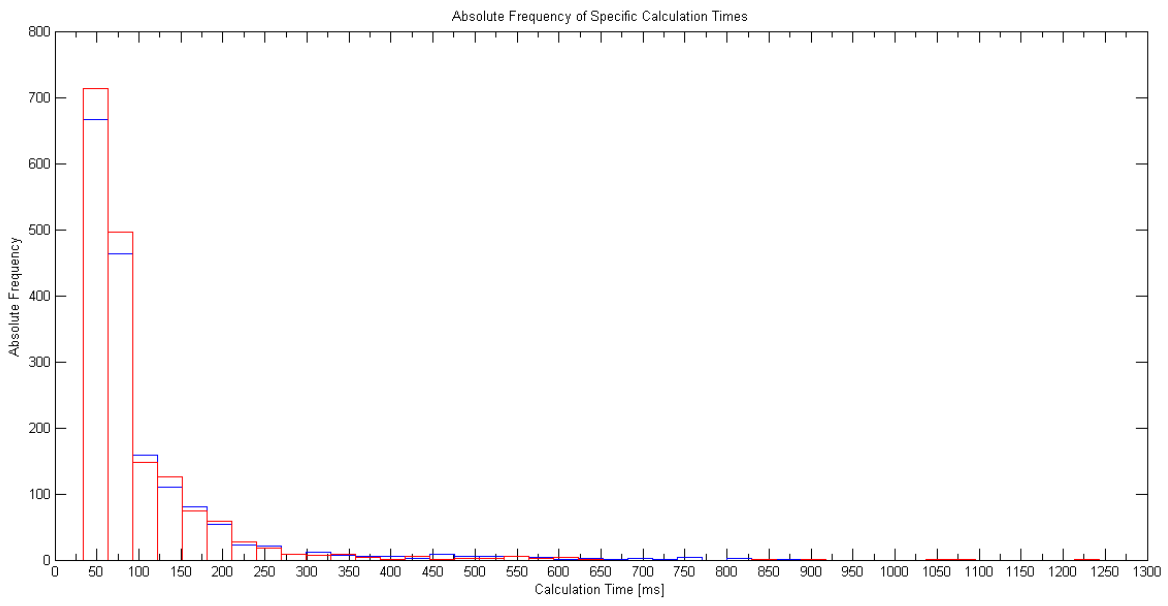

Figure 3-4: histogram of calculation times; red - Master; blue - Slave

As it can be seen regarding the IDs of the first and the last and the total numbers of processed images, apparently the algorithm did not even look at a lot of recorded images. This can be explained with the massive workload-overhead of the core systems, which was caused by the operating system, its dynamic scheduling and the bundle of functionalities the core system software had to fulfill (cf. 3.4.3). Hence, the Algorithm-Thread was probably suspended by the operating system that often and that long, sometimes even during the procession of an image causing the heavy outliers of the calculation time, that the algorithm could only process every 4th or 5th recorded image or even the buffer between the VideoSaver-Thread and the algorithm, which was limited in size, flew over.

However, in general the performance of the algorithm during flight was excellently nominal and would have resulted in valuable scientific data, if the provided input-images had not been overexposed.

3.5.4 Discussion of the Performance in Simulation

As the data gathered during the flight provided only little information (cf. 3.5.3) about the algorithm's actual performance, it was decided to thoroughly evaluate not only the flight data but also of a simulation carried out after the campaign with the stand-alone version of the algorithm (cf. 3.5.2). Using Cinema4D a model of a satellite tumbling at a height of 200km above the earth carrying a camera with nearly the same properties as the payload-cameras of the actual experiment setup was created and rendered from the point of view of the satellite's camera. The resulting footage was then processed by the algorithm.

The advantage of the simulation is that the complete model (especially the relative position of the camera to the earth) is known and thus an evaluation using mathematical methods is possible and not every single image has to be manually evaluated (as also the flight data would have had to be evaluated, if it had contained usable information (Rapp, et al., HORACE Student Experiment

Documentation - v4.0, 2014, p. 99)). On the other hand, the model, of course, does not exactly describe the reality – for example the atmosphere could only roughly be modeled.

To compare both the known data of the simulation and the algorithm's results, some transformations are necessary, which are explained in the following and conducted by a Matlab-Script (see appendix for the raw data and script).

The global coordinate system of the simulation is earth centered and fixed. The camera-position is given in km by vector $v0$ and its rotation to the global coordinate system by the Cartesian matrix $(v1, v2, v3)$, where $v3$ is the camera's view axis (normal to the image-plain), $v1$ lies horizontally in the image-plain starting at the center of the image and respectively $v2$ vertical.

Therefore, the projection of the earth center to the image-plain in global coordinates is the dot-product of $v0$ and $v3$ multiplied by $v3$ being normalized.

$$f_g = \frac{\langle v0|v3\rangle * v3}{\|v3\|}$$

To get the 2D vector from the image center to the projection, $v0$ and f_g are subtracted and transformed to the camera-coordination system by left-multiplication with the inverse matrix of the camera orientation and disregarding the third component of the result:

$$f_c = (v1, v2, v3)^{-1} * (f_g - v0)$$

The angular direction is hence the arcus-tangent of the ratio between the coordinates of f_c (which have to be converted to a right handed coordinate system by multiplying the second coordinate with -1):

$$\alpha = \tan^{-1} \frac{\langle f_c|\binom{0}{-1}\rangle}{\langle f_c|\binom{1}{0}\rangle}$$

The given results of the algorithm a_c in pixels are already in the camera-coordinate-system and right handed, but have to be centered to the image center by subtracting offsets (half the image-resolution each component):

$$a_{c,o} = a_c - \binom{512}{384}$$

The angular direction is again the arcus-tangent between both components of $a_{c,o}$

$$\beta = \tan^{-1} \frac{\langle a_{c,o}|\binom{0}{1}\rangle}{\langle a_{c,o}|\binom{1}{0}\rangle}$$

The angular difference $\Delta = \alpha - \beta$ is calculated for every of the n successful horizon acquisitions and the results plotted to a histogram summarizing the values for Δ to $\sqrt{n}$ classes and counting their occurrence/absolute frequency. Figure 3-5 shows the results with a Gaussian fit to the data. The offset of the mean of only 0.03° as well as the outliers are most likely to be caused by the dynamic adjustment of the threshold filter which leads to larger errors in mostly dark images (but ensures that the horizon can be detected at all also with bad illumination).

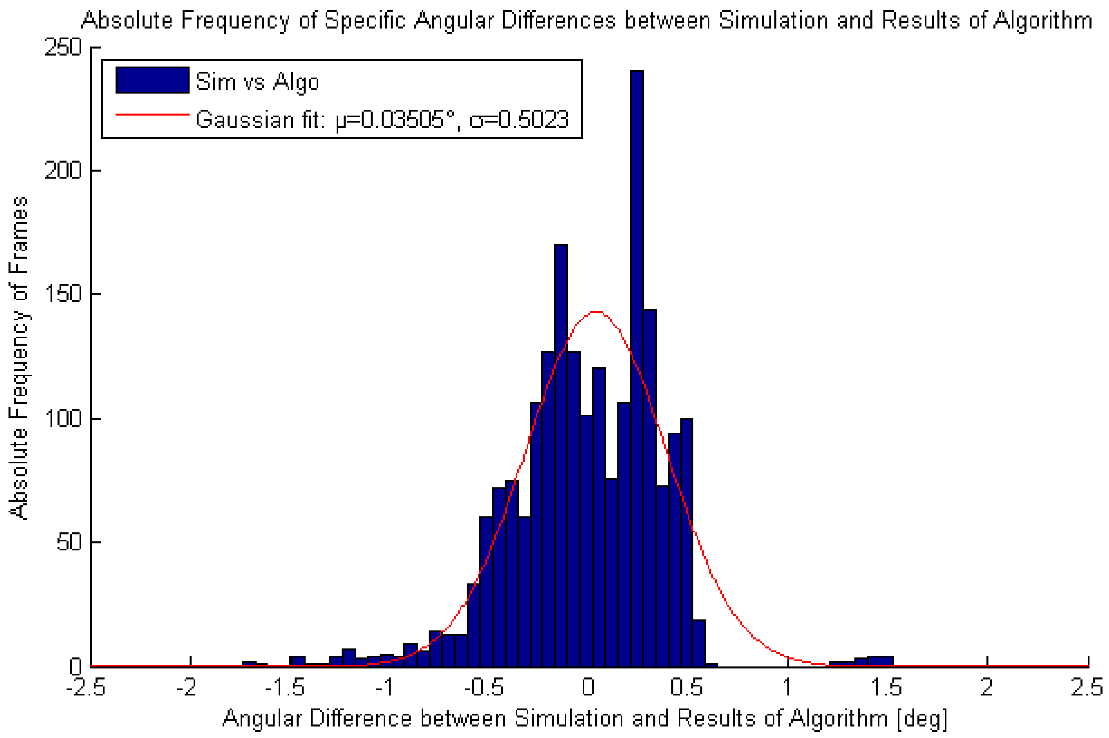

Figure 3-5: histogram of angular difference for simulation

As no mathematical solution to convert the vector f_c from km to pixels or $a_{c,o}$ vice versa was found, the norm of both vectors, unlike the angular direction, cannot be directly compared (e.g. to calculate a difference). However, plotting them for each image (cf. Figure 3-6) already shows a clear correlation, which fits to the camera movement observed in the simulation-video, and the calculation of the correlation-coefficient reveals a real high correlation ($r = 0.7;\ p = 2.3 * 10^{-295}$), suggesting that not only the direction but also the actual position of the projected earth center was calculated quite accurate.

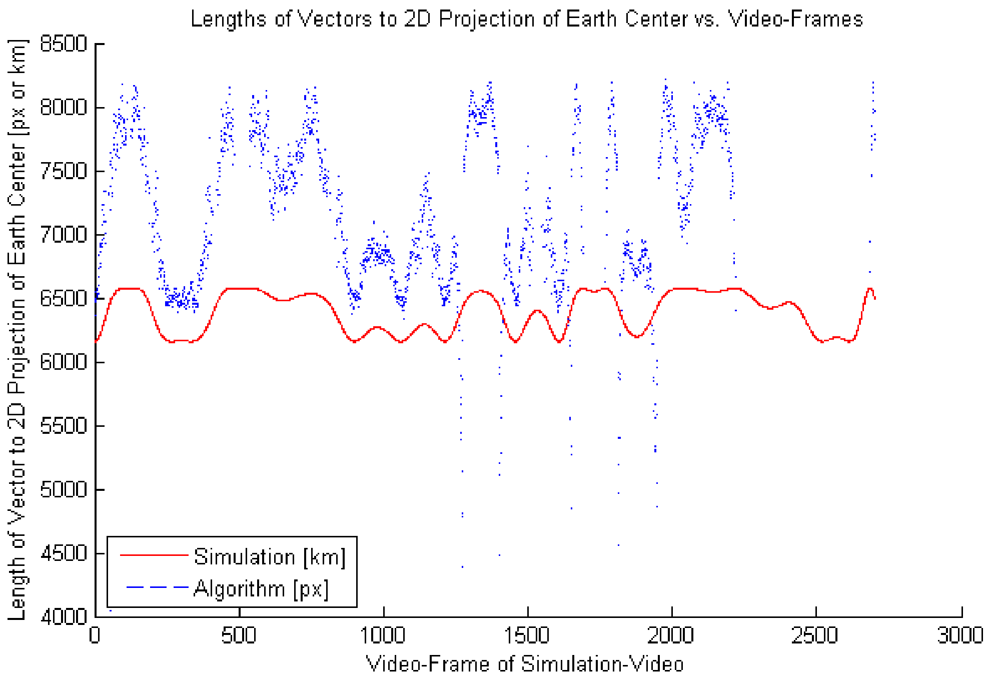

Figure 3-6: comparison of vector-lengths between algorithm results & simulation

Also, the criterion to determine whether the horizon is visible in the image was too complicated to be solved in a close form. Therefore, the image-frames which do not show the horizon were simply manually counted (horizon always visible except from #2257 to #2673) and thus decided whether a false positive or false negative occurred. For the whole simulation not a single false positive (horizon detected where is none) occurred and the rate of false negatives (10.29%) was only slightly above the required value (10%). Most of the false negatives occurred when the horizon was nearly out of the image, thus the line presumably was too short for proper extrapolation and therefore skipped. In other cases the horizon, also for the human eye, seemed very straight, so that small errors due to the dynamic threshold filter probably caused an infinite (or at least very large) radius for the least-square-fit exceeding the defined upper bounds.

All those results show that the algorithm for the horizon acquisition works excellently although it was not possible to prove that properly under space-like conditions during flight due to the overexposure of the cameras (cf. 2.4). As the calculated results especially for the angular direction with an error of only ±0.6° (offset of mean + standard derivation) are even stronger than the criteria actually required (e.g. $e/R \leq 0.1$ equals in worst case a difference of ±5.7° for the angular direction) an explicit evaluation regarding those criteria is waived.

3.6 Communication Middleware

3.6.1 Relevant Requirements

One of the functional requirements for the software was to downlink specific data for two reasons: Downlinking a minimum set of scientific data during flight should ensure that at least some evaluation of the experiment would be possible if the payload data could not be recovered (e.g. if complete payload was lost or data storage devices broken). Additionally, during countdown the downlinked data should help to check out the experiment and to be sure that the experiment is ready to fly.

Several design requirements specifying which data shall be downlinked in which states come along with those functional requirements.

Moreover, it was required to provide a system-wide unique timestamp all sub-systems are synchronized with to ensure that data from different subsystems can be joined.

3.6.2 Design & Implementation

Actually those requirements do not directly lead to the stringent necessity of a communication middleware or framework, but as the relevant design requirements imply and the definitions of all data interfaces and different data packages to be exchanged or stored (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 72-73) clearly show, the communication infrastructure of HORACE was quite complex.

To avoid ending with a total mess of the protocols being implemented anywhere within the implementation of the software of the involved subsystems, it was decided to bundle all functionality related to storing data to data storage devices or exchanging data between different subsystems in a special “communication”-namespace. Furthermore, as for example the definition of data interfaces (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 41-44) show, between different interfaces and packages there are many similarities (like beginning with a two-byte long sync-word). Therefore, it seemed to be fair to follow an object-oriented approach and to systematically group similar functionalities into a class hierarchy (cf. Figure 3-7):

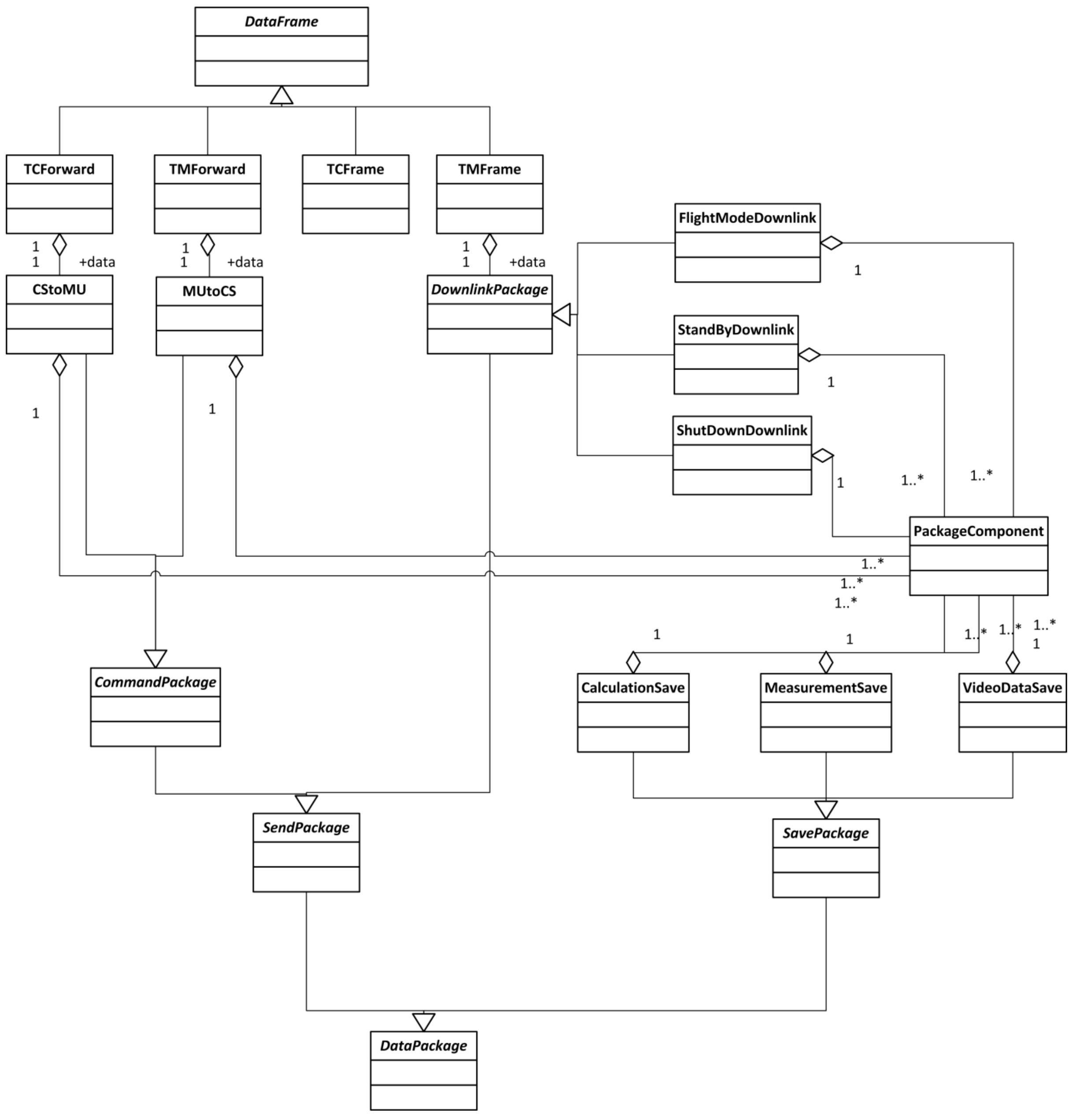

Figure 3-7: UML-class-diagram of the communication-middleware, showing inheritance and associations; member-variables & -methods not shown for simplicity

DataFrame is a wrapper class for the protocols between different subsystems (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 41-44) and is split into four child-classes (TMFrame, TCFrame, TMForward, TCForward) according to the interfaces (either RXSM ↔ CS or CS ↔ MU) and the direction of communication.

A DataPackage is either a SendPackage and is wrapped into a corresponding DataFrame-Object or a SavePackage and is meant to be directly written to a data storage device.

All DataPackages consist of a set of PackageComponents, which represent the different defined fields (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 72-73) and convert and store the passed data in char-arrays with the appropriate length.

Therefore, building for example a TMFrame creates a char-vector, appends the class-specific values like the sync-word and calls the building-method of the wrapped DownlinkPackage, which itself again only iterates over all its PackageComponents and appends their data (char-arrays with different length) to the returned char-vector.

Parsing received data goes exactly the other way round: First the data is unwrapped from the overhead of a DataFrame and by means of this overhead it is decided which child-class' (of DataPackage) parse-method is called. This method again constructs the specific PackageComponent-objects, which eventually hold the actual values.

With this structured hierarchy of all specified data-packages and protocols and a SerialHandler-class, which takes control of the buffers and the actual electrical data interfaces (RS-232 in most cases), all functionalities regarding exchange or storage of data could be standardized and thus easily used by all subsystems except from the Measurement Unit, whose EEPROM was too small to import the whole communication middleware (cf. 3.7.2). Especially the development effort for the ground station software was drastically reduced by this reusability (cf. 3.8.2).

3.6.3 Discussion of the Performance during Flight

Not only the advantage of saving development time but also the performance of the communication middleware during countdown and flight proved that it has been a fair decision to implement it although it was not directly required.

During countdown no irregularities were experienced at all. Health data from the MU as well as all expected status reports from both core systems were received and all telecommands correctly uplinked as the expected responses were received at the ground station.

Also during shutdown-mode the Master received valid health data with reasonable values as well as no corruption of packages saved by the Measurements could be detected.

As only the algorithm's results were both stored and downlinked, the correspon-ding packages are the only ones which can be directly compared and thus make numerical statements about the performance of the communications of REXUS 16 in general but also of the communication middleware of HORACE possible.

Of more than 1650 downlinked frames only 1 frame received by the Slave was corrupted and the following 4 completely lost. As this loss of downlink data took place between T+296s and T+299s and can be time-correlated to the loss of 4 downlinked frames of the Master between T+295s and T+297s, it is most likely that during this time the radio-link in general was bad. Two more frames of the Master were corrupted (at T+252s and T+278s) and one pair of successive received frames was swapped compared to the transmitted ones (at T+396s).

Whereas the corrupted frames may simply be transmission errors, the swapping might have occurred as the first frame was “stuck” in the ground station’s receive-buffer (cf. 3.8.3).

Both on the Master and Slave the last 13 packages, which should have been saved to the SSD, were empty (file-size 0 bytes), which was probably caused as not all internal buffers of the SSDs were flushed when the power was switched off. This was a known issue but not further pushed to be solved as those last frames were expected to be completely useless for the scientific evaluation (no more horizon visible at that altitude). But – as a fun fact – 12 of them could properly be downlinked before power-off. Hence, if they had contained any relevant data, they would not have been lost. The same happened to the last several (Master: 35; Slave: 30) images which should have been saved to the SSDs (but were completely lost as image-data was not downlinked), backing the thesis that it was an issue with the SSD-buffers.

All in all the communication middleware provided a stable software infrastructure for all functionalities regarding data transfer and data storage of HORACE and thus contributed to the corresponding requirements and hence the experiment in general although the design and implementation of the middleware was not directly required.

3.7 Measurement Unit

3.7.1 Relevant Requirements & Constraints

The main functionality which was carried out by the Measurement Unit was to gather health- and housekeeping data of the experiment (F-E-06), downlink them during stand-by (F-S-16) to check out the experiment before lift-off and store them to own data storage during flight (D-S-01). Additionally to the experiment checkout, with the gathered health data some of the questions stated in the Experiment Objectives (cf. 1.3 – 2nd primary objective & 2nd secondary objective) should be answered (cf. also 3.1.3). Several other performance and design requirements both regarding the hard- and software, (e.g. for sensor-accuracy and sample rates, write speed and size of data storage) come along with that functionality.

The main constraints introduced for this subsystem originate from the selected component (due to the design decision of the electrical design, cf. 3.1.2) – an Arduino Leonardo with micro-SD-shield and additional RS-232 interface – and its limited hard- and software capabilities.

3.7.2 Design & Implementation

As the measurement of health data by any subsystem for itself would falsify the results – especially regarding the power consumption, which was relevant most for the Experiment Objectives –, it was decided at a very early stage of design to carry out this task using a special subsystem – the Measurement Unit. According to the decision for the electrical design to use components off-the-shelf (cf. 3.1.2) the Arduino Leonardo extended with a micro-SD-shield and native RS-232 interface was selected as it met all hardware requirements, was cheap and considered to be easy to use as the Arduino platform is especially known for its low entry level and lively community.

As there is only one data interface to RXSM available for each of the two quasi identical setups of the experiment, but the Measurement Unit also has to be controlled (both by the signals of RXSM and telecommand) and needs access to the telemetry-interface, it was decided to control it via the core system's RS-232-interface. The CS basically forwarded all signals and commands as they arrived, independently from the actual content of the message, to the MU as well as the data to be downlinked provided by the MU. The CS also provided a time-synchronization via the same RS-232 interface to ensure that the data from both systems could be matched for the evaluation. The fact that the MU was the only subsystem which was not redundant in the setup of two identical experiments led to the distinction between those two into a Master, which controlled the MU, and a Slave. Despite being controlled by the CS the MU basically worked autonomously: After power-on and initialization self-checks (try to read and write the SD-card) were performed before gathering sensor data following a static scheduling and either downlinking or saving the data according to the current state of a simple state-machine.

In fact, thanks to many open source libraries and example programs it was really easy to get started on the platform and implement the basic functions like gathering sensor data, use the SD-card and serial interfaces. But the simplified programming language (based on C++) with special language structs and modified compiler and linker turned out to be a crux for advanced purposes as it severely complicated the modularization of the software. Hence, the final application (with around 1000 lines of code) is barely modularized, what makes it hard to debug and maintain.

But besides the software platform also the hardware was quite limited. For example, the EEPROM was very limited in size and mainly consumed by needed libraries for the serial interfaces and sensor access. Therefore, it was impossible to directly import the communication middleware (cf. 3.6), but the needed parts had to be manually adapted. Furthermore, security features (e.g. to remotely format the SD-card to solve the SD-card failure (cf. 2.1)) could not be implemented. Also regarding the security of gathered data and sample rate a trade-off had to be made as flushing the write buffers after every single sample exceeded the desired sample rate.

3.7.3 Discussion of the Performance during Flight

Because the issue of the SD-card failure could be solved before the hot countdown and flight (cf. 2.1.6), the general performance of the Measurement Unit was excellent as all tasks were conducted nominally and the whole software flow through the different states during countdown and flight was nominal as well. However, as already discussed in 3.2.3, a temperature sensor was lost during assembly and no time was left to replace it and the required measurement error of one current sensor was exceeded due to interferences with external magnetic fields (cf. 3.1.3). Thus the corresponding performance requirements were only partly met.

Although the desired sample time was exceeded every second while flushing the write buffers (ca. 140ms instead of ca. 40ms), the average sample time of 46ms was sufficient to meet the corresponding performance requirements and all other requirements were also met.

A minor discrepancy was detected in the saved data regarding the time synchronization with the core system, which only slightly affected the quality of the saved data. Throughout the flight the time-offset, which holds the time difference between the MU's internal and the global clock of the CS (see (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, p. 43) for the definition of the time-sync-protocol), was decreased by ca. 790ms and increased by ca. 800ms again one second later for nine times in total with a quite constant rate of once per minute. This is most probably caused by the dynamic scheduling of the operating system of the CS and its insufficient real-time capabilities. If one command-package from the CS to the MU was delayed (e.g. by unexpected suspension by the operating system) that much that in the meantime a second time-pulse was sent, this would cause the MU to turn back its clock from the time of the second pulse to the time received with the first delayed package and simply ignore the second received package (as no new time-pulse was received in the meanwhile). The time-sync would be regained causing the MU to turn its clock forward again when the next time-pulse and its corresponding command-package arrive, which is nominally the case one second later. Thus, the timestamps logged by the MU are ambiguous at nine points for one second each, which – as nothing interesting or relevant happened during those periods – only slightly affects the overall quality and integrity of the gathered data.

To sum, up the Measurement Unit worked nearly perfectly, meeting only two requirements only partly, and thus provided valuable data not only for checking out the experiment during countdown but also for the evaluation of the thermal and electrical design as well as to answer two main questions of the Experiment Objectives.

3.8 Ground Station Software

3.8.1 Relevant Requirements

As the Flight Segment was required to downlink selected data both for scientific and operational reasons (to at least have a minimal set of scientific data if the payload was lost, respectively to check out the experiment during countdown), of course also a ground station was needed to receive the data. For the operations it was important that the ground station would not only properly save all received data but also display them in real-time and in an appropriate manner to thoroughly monitor the experiment. Also the capability to send telecommands and not only to receive telemetry was required for operations of both countdowns and tests.

More detailed and soft requirements, especially for the GUI and regarding usability, were worked out during design and implementation, mostly by feedback of the operators.

3.8.2 Design & Implementation

Originally, it was planned to implement the ground station software in Python as it is a quite easy script-language, some team-members already had experience with regarding GUI-design and implementation. But as it was realized that the powerful communication middleware (cf. 3.6) could completely be reused for the ground station – thus nearly the complete data and most of the functional layer (receiving and interpreting telemetry, formatting and sending telecommands) were already implemented –, it was decided to implement the ground station software in C++ as well. This decision not only saved a lot of development time but also minimized the risk of bugs (the middleware was already thoroughly tested and even if there were bugs, for example regarding formatting and interpreting packages, they were likely to be neutralized).

To minimize the impact of possible bugs, the backend directly saved the received byte-stream before interpreting it to display the contained information and to store the values to comma-separated files. Hence, even with malfunctions of the frontend all raw-data was saved for later evaluation.

The same software was used both for the Master and Slave as the behavior of both systems was, except from the availability of health data (cf. 3.7), nearly the same. Therefore, to avoid distraction of the operator, simply some functions were disabled or hidden when using the ground station in slave-mode. The GUI-frontend was designed in a clearly structured and intuitive manner (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, pp. 76-77) with different views for different modes and colored indications so that the operator could monitor all important data simultaneously.

Furthermore, several security features were implemented to prevent accidental actions of the operator during stress situations (as the hot countdown for sure was one). For example, double confirmation was required for critical actions; it was impossible to quit the whole software, if the serial connection was established and the telecommands, needed during a nominal countdown, were provided as short-cut-buttons.

3.8.3 Discussion of the Performance during Flight

The ground station software ran absolutely stable and behaved nominally throughout the whole campaign including all test countdowns, the hot countdown and the flight.

The only issue was that sometimes – not really reproducible – probably due to some problems with the driver of the RS232-interface of the EGSE-laptops single telemetry frames got “stuck” in the buffers. But as the frames were not lost but only displayed belatedly (in most cases, latest when the next frame arrived) and as the issue was known, it did not narrow the performance of the ground station software at all.

Thus, the ground station software met all requirements and helped to gather scientific data and to check out the experiment during countdown and testing. So it contributed to reach full functionality of the experiment in general terms in an excellent way.

4 SUMMARY & GENERAL EVALUATION

With the given scientific data gathered during flight, which is deficient due to the overexposure of the cameras of both systems (cf. 2.4), it is neither possible to "prove [n]or disprove the general technical feasibility" (cf. 1.2) of horizon acquisition with ordinary cameras and powerful image processing algorithms for attitude determination (cf. 3.5.3). However, the simulation showed that the algorithm itself with an accuracy of ±0.6° (cf. 3.5.4) is likely to be sufficient for attitude determination (cf. 1.3 – 1st primary objective) – especially in emergency cases during which also only roughly determined attitude information would be better than none – although that could not be proven under "realistic, space-like conditions" (cf. 1.2).

The actually achieved time resolution of approximately 2fps of processed image-frames (cf. 3.5.3) is much lower than required and is not considered to be "reasonable" (cf. 1.3 – 2nd primary objective). But the further evaluation showed that this was mainly due to the massive overhead to be executed by the core systems. So, the time resolution of particularly the algorithm was much higher (about 20fps, cf. 3.5.3) and was close to the required value. Therefore, the time resolution is considered to be "reasonable" in terms of the Experiment Objective despite of the low actual value. The same applies to the overall power-consumption: Of course, the power consumption of both the core systems and the total Flight Segment (cf. 3.1.3) exceed a satellite's power budget (especially in emergency cases). But regarding again the functional overhead, which also increased the power-consumption, and furthermore the fact that the experiment was not optimized towards low-power-consumption, the achieved performance seems to be "reasonable" in terms of the Experiment Objective.

Regarding the secondary objectives (cf. 1.3) as systematical limits, the camera and the threshold filter could be identified. Both the quality of the provided images and the parameters of the threshold filter severely affect the quality of the results (cf. 3.5.3 and 3.5.4). Whereas for the camera simply a better component could be selected if necessary (respectively the issue fixed), the parameters for the threshold filter must either be carefully experimentally determined or dynamically set by a higher autonomous logical unit. The applicability of a future complete sensor system for small satellites can be answered only indirectly, as HORACE was not optimized towards that. Several other projects (like STELLA (Balagurin, Wojtkowiak, & Kayal, 2011)) have proven that image processing applications can be run on hardware which is applicable for small satellites. As during the development of HORACE no severe increase of complexity of the image processing software compared to the existing applications was experienced, it can be expected that a full sensor system would also be applicable for small satellites in the future.

Due to the deficient scientific data gathered during flight and the design itself, as pointed out, several of the questions posed in the Mission Statement and Experiment Objectives could only be partially or indirectly answered. That of course decreases also the overall mission success.

On the other hand, the experiment development, which explicitly is defined as "part of the mission" (cf. 1.3), has been nearly perfectly successful. During the project lifecycle all challenging milestones were excellently and always in time reached. The overall design and implementation of the experiment in general was profound (cf. 3), meeting most requirements and leaving only few issues open, which despite of their severe impact on the experiment performance are from the technical point of view only minor issues. Thus, the experiment could be delivered in time to EuroLaunch and it was successfully launched and fully operational throughout the whole flight.

Regarding all those factors, all remaining issues but also all achievements, which were obtained, the whole mission “HORACE on REXUS 16” is considered to be an 80% partial success.

5 OUTLOOK ON FUTURE EXPERIMENTS

As HORACE was only the very first basic study and unfortunately left several of the posed questions partially open or could only indirectly answer them (cf. 4), it will for sure take several more steps until the final aim of an operationally used sensor system for emergency cases, which is also applicable for small satellites, will be reached.

The results of HORACE and the experiences gathered throughout the project suggest several possible directions for follow-up experiments. The four most appealing possibilities are outlined in the following raising no claim to completeness which may – respecting resources like manpower, budget and time – also be combined to some extend although they are presented in closed subchapters.

5.1 Re-Flight on REXUS

The obviously easiest way to close the gap between the posed questions and the answers available from HORACE would be a re-flight of the experiment on a sounding rocket. Of course, the issues which limited the performance of HORACE (cf. 2) shall be eliminated preferably following the outlined solutions (cf. 2.x.6).

But as firstly a generally identical experiment is unlikely to be selected twice for the REXUS-programme and secondly the full two year life-cycle must be accomplished leaving much time for adjustments, some more improvements are suggested.

Especially the flight-software of the core systems should be optimized regarding the overall speed (cf. 3.4.3) so that the overhead is reduced and the ratio between saved image-frames (ca. 10fps, cf. 3.4.3) and actual processed images (ca. 2fps, cf. 3.5.3) is balanced and thus more scientific data collected.

Furthermore, the evaluation methods could be significantly improved. As outlined in the evaluation plan of HORACE – if proper scientific data would have been collected as expected – the available data would have required to “manually” evaluate every single frame regarding the success of the horizon acquisition. (Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014, p. 99) This would have been necessary because neither the RXSM collected appropriate reference data with the needed accuracy, nor resources were left to develop another subsystem performing reference measurements. (Rapp, et al., HORACE Student Experiment Documentation - v2.0, 2013, p. 30) This could probably be done throughout the development and integration phase of another REXUS-cycle as other subsystems could be directly adapted. For instance, it is suggested to replace the Measurement Unit, whose benefit would only be little as the thermal design was proven to be good (cf. 3.2.3), with a subsystem accurately determining the attitude of the vehicle using Inertial Measurement Units (IMU).

To handle both the drift of IMUs and the wide required range (rotation: 0-4Hz, acceleration: 0-20g (Mawn, Kaczmarczik, & Schmidt, 2014, p. 38)), probably sensor fusion methods must be applied to reach the desired accuracy. Additionally, the comparison of the results of the horizon acquisition with the recorded attitude is probably a challenging mathematical problem, thus this improvement would surely take some effort. The advantage would be that – except from the changes required to eliminate the experienced issues – the already flight-proven experiment setup including most hardware-parts could be reused, causing a drastic saving of time both for development and implementation as well as financial budget.

5.2 Upgrade to Full Attitude Determination

The conversion of the vector resulting from the horizon acquisition from pixels to (kilo)meters is a key factor towards a full attitude determination via the horizon acquisition but could not be solved in the scope of this thesis and only a correlation was shown (cf. 3.5.4) as it was not the focus of this thesis.

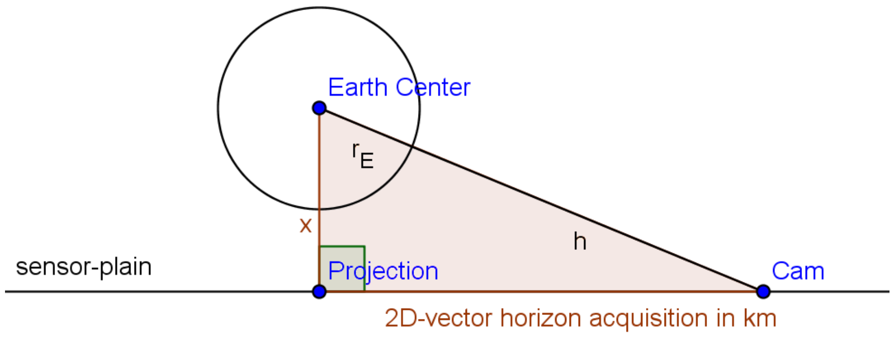

Figure 5-1: side view, projection of earth center to sensor-plain

If this mathematical problem was solved together with the height h of the satellite above ground, which can be calculated from the results of the horizon acquisition as shown in (Barf, 2014), the triangle of Figure 5-1 would be fully described. Thus, the distance x from the Earth Center to the sensor-plain could be easily calculated and the projection of the Earth Center to the sensor-plain (cf. 3.5.4) could be inverted. The result would be a 3D-vector (instead of 2D in the sensor-plain) from the satellite to the Earth Center within the camera-frame. If this vector is also known in a reference-frame, the full attitude information relatively to the reference-frame could easily be calculated. The most suitable reference-frame would be the ECI-frame and the vector to the Earth Center would be given by the satellites position, which, if not determined directly, can also be very well predicted using mathematical models like SGP4.

If adapted for a satellite, this improvement of basically the software shall implement the functionality to either receive position-information from the OBDH-computer or to predict the position itself using SGP4, being regularly updated with NORAD-Two-Line-Elements.

If adapted for REXUS, the position shall be either determined directly within the experiment (e.g. using GPS) or be matched with the data from RXSM after the flight. Similar to the pure re-flight (cf. 5.1) only few changes of the design and setup in general would be required and thus time and money would be saved.

5.3 Miniaturization for Satellite Mission

One would have to make much more effort, of course, if the experiment was miniaturized and ported to real embedded hardware to fly it as payload of a small satellite. From the point of view of the horizon acquisition this step is already prepared, as in (Barf, 2014) a profound application for the horizon acquisition was developed. As it uses only standard C++-libraries and follows coding directives for software of space projects, the porting of the horizon acquisition itself shall be no problem whereas the complete hardware and basic software (control, communication-interfaces etc.) has to be redesigned.

Furthermore, if one takes no intermediate step, the requirements for the miniaturized experiment will be much more challenging not only regarding size and power consumption but also regarding redundancy, space conditions, possibility for on-orbit software updates or adjustment of parameters, etc. Additionally, as outlined in 5.1, a possibility to compare the results of the horizon acquisition payload with reference data must be guaranteed, as due to bandwidth limitations it is unlikely that all image-data can be downlinked for evaluation.

Because of the high complexity it is strongly recommended to demonstrate the functionality of the miniaturized experiment on a flight with a sounding rocket before its launch on a satellite. That experiment shall then clearly separate the payload from the functions the satellite’s OBDH-computer would fulfill later. Hence, additionally to the horizon acquisition payload, which shall be exactly the same as intended to fly on the satellite, another subsystem is needed, which on the one side implements the interface to the sounding rocket and on the other side simulates the satellite-bus (both regarding data-handling and power distribution).

5.4 Porting to Existing Hardware

Another possible direction, which would save the effort of hardware development, would be to port the experiment’s software to an existing platform of embedded hardware which is used for similar operational scenarios and is available to the Chair of Aerospace Information Technology from other projects like ASAP (Wojtkowiak, Balagurin, Fellinger, & Kayal, 2013) or STELLA (Balagurin, Wojtkowiak, & Kayal, 2011). This would be very challenging like the miniaturization (cf. 5.3) but for other reasons. Firstly, the given hardware for data-handling would put severe constraints on the software and probably a totally different technology (FPGA instead of RISC-processor) is used so that even the more portable version of the horizon acquisition of (Barf, 2014) cannot be used. Secondly, especially the optical system probably is not appropriate for the application of horizon acquisition as it was not designed for that. Although the long term goal is generic software, this would require the algorithms for the horizon acquisition to be drastically improved already now.

As well as for the miniaturization (cf. 5.3) intermediate steps or at least the demonstration on a sounding rocket are strongly recommended due to the high level of complexity.

6 ABBREVIATIONS, LISTS OF FIGURES & REFERENCES

6.1 Abbreviations

Most of the abbreviations are standard-abbreviations of the REXUS-Programme and are therefore listed here although they might not be used in this thesis. Further abbreviations (either specific for HORACE or common in general but newly introduced to the context of REXUS by HORACE) are written in italics.

AIT	Assembly, Integration and Test
ADC	*Analog Digital Converter*
ADCS	*Attitude Determination and Control System*
BEXUS	Balloon Experiments for University Students
CAD	Computer Aided Design
CDR	Critical Design Review
COG	Centre of Gravity
CPU	Central Processing Unit
CRP	Campaign Requirement Plan
CS	*Core System*
DLR	Deutsches Zentrum für Luft- und Raumfahrt
EAT	Experiment Acceptance Test
EAR	Experiment Acceptance Review
ECI	*Earth Centered Inertial (coordinate system)*
ECTS	European Credit Transfer System
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
EGSE	Electrical Ground Support Equipment
EIT	Electrical Interface Test
EPM	Esrange Project Manager
ESA	European Space Agency
Esrange	Esrange Space Center
ESTEC	European Space Research and Technology Centre, ESA (NL)
ESW	Experiment Selection Workshop
FAR	Flight Acceptance Review
FAT	*File Allocation Table (file system)*
FPGA	*Field Programmable Gate Array*
FS	*HORACE Flight Segment*
FST	Flight Simulation Test
FRP	Flight Requirement Plan
FRR	Flight Readiness Review
GigE	*Gigabit Ethernet*
GPS	Global Positioning System
GSE	Ground Support Equipment
GUI	*Graphical User Interface*
HDD	*Hard Drive Disk*
HORACE	*Horizon Acquisition Experiment*

HK	House Keeping
H/W	Hardware
ICD	Interface Control Document
ID	Identifier
IMU	*Inertial Measurement Unit*
ITW	Integration Week
I/F	Interface
IPR	Integration Progress Review
JMU	*Julius-Maximilians-Universität Würzburg*
LO	Lift Off
LT	Local Time
LOS	Line of Sight
Mbps	Mega Bits per second
MFH	Mission Flight Handbook
MGSE	Mechanical Ground Support Equipment
MORABA	Mobile Raketen Basis (DLR, EuroLaunch)
MU	*Measurement Unit*
NORAD	*North American Aerospace Defense Command*
OBDH	*On-Board-Data-Handling*
OP	Oberpfaffenhofen, DLR Center
OS	*Operating System*
PCB	Printed Circuit Board (electronic card)
PDR	Preliminary Design Review
PDU	*Power Distribution Unit*
POSIX	*Portable Operating System Interface*
PST	Payload System Test
RAM	*Random Access Memory*
RBF	Remove Before Flight
REXUS	Rocket-borne Experiments for University Students
RID	Review Item Discrepancy
RISC	*Reduced Instruction Set Computing*
RXSM	REXUS Service Module
SATA	*Serial Advanced Technology Attachment*
SD-card	*Secure Digital Memory Card*
SED	Student Experiment Documentation
SGP4	*Simplified General Perturbations*
SNSB	Swedish National Space Board
SODS	Start Of Data Storage
SOE	Start Of Experiment
SSC	Swedish Space Cooperation
SSD	*Solid State Disk*
STW	Student Training Week
S/W	Software

T	Time before and after launch noted with + or -
TBC	To be confirmed
TBD	To be determined
TC	Telecommand
TM	Telemetry
USB	*Universal Serial Bus*
UTC	Coordinated Universal Time
WBS	Work Breakdown Structure
WP	work package
XML	*Extensible Markup Language*
ZARM	Zentrum für Angewandte Raumfahrttechnologie und Mikrogravitation

6.2 List of Figures & Tables

6.2.1 Figures

6.2.2 Tables

6.3 References

Arduino. (2014, 05 09). *Writing to SD Card after removing and replacing it - Arduino Forum*. Retrieved 08 03, 2014, from http://forum.arduino.cc/index.php/topic,239157.0.html

Balagurin, O., Wojtkowiak, H., & Kayal, H. (2011). STELLA - A New Small Star Tracker for Pico and Nano Satellites. *8th IAA Symposium On Small Satellites for Earth Observation, IAA-B8-1201.* Berlin.

Barf, J. (2014). *Development and Implementation of an Horizon Sensing Algorithm Based on Image Processing Technologies.* Würzburg.

Bergmann, M., & Rapp, T. (2014, 03 13). Procedure S#55 - Save settings onto camera. *(RX16_HORACE_PRO_S_55_saving_settings_to_camera_v1-0_13Mar14.docx)*. Würzburg.

EuroLaunch. (2014). *REXUS BEXUS*. Retrieved 08 04, 2014, from www.rexusbexus.net

Gre, B. (2011, 03 10). *Issue 18 - sdfatlib - Education - A FAT16/FAT32 Arduino library for SD/SDHC cards - Google Project Hosting*. Retrieved 08 03, 2014, from https://code.google.com/p/sdfatlib/issues/detail?id=18#c1

Matrix Vision GmbH. (2014, 07 07). *version info mvIMPACT Acquire.* Retrieved 07 18, 2014, from http://www.matrix-vision.com/mvvirtualdevice.html?file=tl_files/mv11/support/mvIMPACT_Acquire/01/versionInfo.txt

Mawn, S., Kaczmarczik, U., & Schmidt, A. (2014, 01 08). *REXUS User Manual.* (EuroLaunch, Ed.) Retrieved 07 22, 2014, from REXUS BEXUS: http://www.rexusbexus.net/images/rx_usermanual_v7-11_08jan14.pdf

Mp3Car.com Inc. (2010, 08 11). *OCZ SSD drive kills my GPS Signal !!! - Mp3car.com*. Retrieved 08 03, 2014, from http://www.mp3car.com/gps/145015-ocz-ssd-drive-kills-my-gps-signal.html

Rapp, T., Barf, J., Bergmann, M., Geiger, S., Scharf, A., & Wolz, F. (2013, 01 28). HORACE Student Experiment Documentation - v1.0. *(RX16_HORACE_SEDv1-0_28Jan13.docx)*. Würzburg.

Rapp, T., Barf, J., Bergmann, M., Geiger, S., Scharf, A., & Wolz, F. (2013, 03 27). HORACE Student Experiment Documentation - v1.1. *(RX16_HORACE_SEDv1-1_27Mar13.docx)*. Würzburg.

Rapp, T., Barf, J., Bergmann, M., Geiger, S., Scharf, A., & Wolz, F. (2013, 06 06). HORACE Student Experiment Documentation - v2.0. *(RX16_HORACE_SEDv2-0_06Jun13.docx)*. Würzburg.

Rapp, T., Barf, J., Bergmann, M., Geiger, S., Scharf, A., & Wolz, F. (2013, 08 08). RID Report - Post-CDR: Thermal Protection of Optical System. *(RX16_HORACE_CDR_RID_01_v1.1_08Aug13_final.docx)*. Würzburg.

Rapp, T., Barf, J., Bergmann, M., Geiger, S., Scharf, A., & Wolz, F. (2014, 01 27). HORACE Student Experiment Documentation - v4.0. *(RX16_HORACE_SEDv4-0_27Jan14.docx)*. Würzburg.

Rudd, N. (2014). *Odes and epodes / Horace ; edited and translated by Niall Rudd.* Cambridge, Mass: Harvard University Press.

Scharf, A. (2013, 10 24). Test #1.2 Report. *Vacuum Test of Core System(RX16_HORACE_TR1.2_v1.0_24Oct2013.docx)*. Würzburg.

Scharf, A. (2013, 09 07). Test #2.5 Report. *Analysis of Horizon Visibility and Mechanical Load during Flight (RX16_HORACE_TR2.5_v1.0_07Sep13.docx)*. Würzburg.

Scharf, A. (2014, 01 25). Analysis Report. *Analysis of Horizon Visibility and Mechanical Load during Flight (RX16_HORACE_AR_v1.3_25Jan14.docx)*. Würzburg.

Wojtkowiak, H., Balagurin, O., Fellinger, G., & Kayal, H. (2013, 06 14). ASAP: Autonomy through on-board planning. *Recent Advances in Space Technologies (RAST), 2013 6th International Conference on*, pp. 377-381.

APPENDIX

The appendix can be found on the enclosed DVD, which contains the following documents and files sorted in a directory-structure as outlined below.

Please note that all documents always remain property of the author(s)/issuing institution(s). Therefore, they may only be copied, distributed, licensed or changed with explicit permission, if not explicitly stated otherwise in the document.

Thesis

File-Name	BA-thomas-rapp_final_20Aug14.pdf
Description	this thesis
File-Format	Portable Document File (PDF)
Author	Thomas Rapp (Team HORACE)

Issue Investigation

File-Name	Issue-Investigation.zip
Description	all gathered data regarding the issues (cf. 2)
File-Format	ZIP-archive
Author	Thomas Rapp, Arthur Scharf, Florian Wolz (all Team HORACE)

Flight Data

File-Name	master_2014-05-28-12-30-38-flightmode.txt
Description	calculation data downlinked during flight, saved by ground station
File-Format	comma-separated text-file
Author	Team HORACE

File-Name	master_2014-05-28-12-30-38-housekeeping.txt
Description	health data downlinked during CD & flight, saved by ground station
File-Format	comma-separated text-file
Author	Team HORACE

File-Name	master_AFTER_LO.avi
Description	video footage of flight of M-CS , recorded raw-data converted to avi
File-Format	AVI-movie
Author	Team HORACE (conversion: Jochen Barf)

File-Name	master_BEFORE_LO.avi
Description	video footage of CD of M-CS , recorded raw-data converted to avi
File-Format	AVI-movie
Author	Team HORACE (conversion: Jochen Barf)

File-Name	master_calc_sorted.csv
Description	calculation-data of M-CS , recorded raw-data converted to csv
File-Format	comma-separated text-file
Author	Team HORACE (conversion: Sven Geiger)

File-Name	MU-data.xlsx
Description	health-data flight of MU , recorded raw-data converted to xlsx & evaluated
File-Format	MS Excel 2010 Spreadsheet
Author	Team HORACE (conv.: Sven Geiger, eval.: Thomas Rapp)

File-Name	RXSM_140528130652_RXS16_flight.xls
Description	flight data recorded by RXSM
File-Format	MS Excel 97-2003 Spreadsheet
Author	EuroLaunch

File-Name	slave_2014-05-28-12-30-41-flightmode.txt
Description	calculation data downlinked during flight, saved by ground station
File-Format	comma-separated text-file
Author	Team HORACE

File-Name	slave_AFTER_LO.avi
Description	video footage of flight of S-CS , recorded raw-data converted to avi
File-Format	AVI-movie
Author	Team HORACE (conversion: Jochen Barf)

File-Name	slave_BEFORE_LO.avi
Description	video footage of CD of S-CS , recorded raw-data converted to avi
File-Format	AVI-movie
Author	Team HORACE (conversion: Jochen Barf)

File-Name	slave_calc_sorted.csv
Description	calculation-data of S-CS , recorded raw-data converted to csv
File-Format	comma-separated text-file
Author	Team HORACE (conversion: Sven Geiger)

Simulation

File-Name	sim_evaluation.m
Description	Matlab-Script for evaluation of the simulation (cf. 3.5.4)
File-Format	m-file (Matlab-script/function)
Author	Thomas Rapp (Team HORACE)

File-Name	simulation_cinema4d_header.txt
Description	header information for the raw-data of Cinema4D
File-Format	comma-separated text-file
Author	Arthur Scharf (Team HORACE)

File-Name	simulation_cinema4d_rawdata.txt
Description	raw-data of Cinema4D (cf. 3.5.4)
File-Format	comma-separated text-file
Author	Arthur Scharf (Team HORACE)

File-Name	simulation_HORACE_header.csv
Description	header information for the raw-data of the algorithm
File-Format	comma-separated text-file
Author	Jochen Barf (Team HORACE)

File-Name	simulation_HORACE_results.csv
Description	raw-data of the algorithm (cf. 3.5.4)
File-Format	comma-separated text-file
Author	Jochen Barf (Team HORACE)

File-Name	simulation_video_with_results.avi
Description	simulation video, results of algorithm imprinted
File-Format	AVI-movie
Author	Jochen Barf, Arthur Scharf (Team HORACE)

References

This section includes all cited references which are not publicly available.

File-Name	RX16_HORACE_AR_v1.3_25Jan14.zip
Description	(Scharf, Analysis Report, 2014), including its appendix
File-Format	ZIP-archive
Author	see 6.3

File-Name	RX16_HORACE_CDR_RID_01_v1.1_08Aug13_final.pdf
Description	(Rapp, et al., RID Report - Post-CDR: Thermal Protection of Optical System, 2013)
File-Format	Portable Document File (PDF)
Author	see 6.3

File-Name	RX16_HORACE_PRO_S_55_saving_settings_to_camera_v1-0_13Mar14.docx
Description	(Bergmann & Rapp, 2014)
File-Format	MS Word 2010 Document
Author	see 6.3

File-Name	RX16_HORACE_SEDv1-0_28Jan13.pdf
Description	(Rapp, et al., HORACE Student Experiment Documentation - v1.0, 2013)
File-Format	Portable Document File (PDF)
Author	see 6.3

File-Name	RX16_HORACE_SEDv1-1_27Mar13.pdf
Description	(Rapp, et al., HORACE Student Experiment Documentation - v1.1, 2013)
File-Format	Portable Document File (PDF)
Author	see 6.3

File-Name	RX16_HORACE_SEDv2-0_06Jun13.pdf
Description	(Rapp, et al., HORACE Student Experiment Documentation - v2.0, 2013)
File-Format	Portable Document File (PDF)
Author	see 6.3

File-Name	RX16_HORACE_SEDv3-0_03Sep13.pdf
Description	HORACE Student Experiment Documentation – v3.0, not cited in the thesis but appended for completeness.
File-Format	Portable Document File (PDF)
Author	Team HORACE

File-Name	RX16_HORACE_SEDv4-0_27Jan14.pdf
Description	(Rapp, et al., HORACE Student Experiment Documentation - v4.0, 2014)
File-Format	Portable Document File (PDF)
Author	see 6.3

File-Name	RX16_HORACE_TR1.2_v1.0_24Oct13.zip
Description	(Scharf, Test #1.2 Report, 2013), including its appendix
File-Format	ZIP-archive
Author	see 6.3

File-Name	RX16_HORACE_TR2.5_v1.0_07Sep13.zip
Description	(Scharf, Test #2.5 Report, 2013), including its appendix
File-Format	ZIP-archive
Author	see 6.3

ACKNOWLEDGEMENTS

Finally, I would like sincerely thank everyone who made the valuable experience of leading a REXUS-team already during my Bachelor’s studies and thus also this thesis possible for me:

First, there are of course my dear team-mates Jochen Barf, Matthias Bergmann, Sven Geiger, Arthur Scharf and Florian Wolz – thank you for the team-work, the fun we had and the friendship, which grew throughout the last two years while we worked together on HORACE.

Prof. Dr. Hakan Kayal and Dipl.-Inf. Gerhard Fellinger, both our supervisors during the project and in particular mine for this thesis – for the trust that we would grow with the challenge and the freedom to gain our own experiences – as one cannot “study” them – but also the guidance whenever it was needed.

Moreover, I would like to thank all people of the institutes and organizations associated with the REXUS-programme – from DLR Agency, ESA, SNSB, DLR MORABA, ZARM and SSC – who provide the frame for that great unique educational programme, for the not only professional but also dedicated and cordial cooperation on a par with the participating students.

Last but not least my parents, who supported me throughout my whole studies wherever possible and in particular during this great adventure.

DECLARATION OF AUTHORSHIP

I declare that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Formulations and ideas taken from other sources are cited as such. This work has not been published before.

Würzburg, 20th August 2014

Signature