NOVEL ON-BOARD DATA PROCESSING STRATEGIES ON NANOSATELLITE SONATE-2

Andreas Maurer[1], Oleksii Balagurin[2], Tobias Greiner[3], Tobias Herbst[4], Tobias Kaiser[5], Hakan Kayal[6], Tobias Schwarz[7]

Julius-Maximilians-Universität Würzburg, Computer Science VIII, Space Technology
Emil-Fischer-Str. 32, 97074 Würzburg, Germany
[1]Phone: +49-931-31-83241, Mail: andreas.maurer@uni-wuerzburg.de
[2]Phone: +49-931-31-88637, Mail: oleksii.balagurin@uni-wuerzburg.de
[3]Phone: +49-931-31-89582, Mail: tobias.greiner@uni-wuerzburg.de
[4]Phone: +49-931-31-84591, Mail: tobias.herbst@uni-wuerzburg.de
[5]Phone: +49-931-31-85780, Mail: tobias.kaiser@uni-wuerzburg.de
[6]Phone: +49-931-31-86649, Mail: hakan.kayal@uni-wuerzburg.de
[7]Phone: +49-931-31-82010, Mail: tobias.schwarz@uni-wuerzburg.de

ABSTRACT

Reliable on-board data handling is the backbone of every successful satellite mission. New scientific experiments can only be validated, and results obtained, if the recorded scientific data reaches the ground completely and correctly. For this, several special techniques were applied during the implementation of the SONATE-2 mission. Generic communication protocols were defined between all payloads, subsystems, and the On-board data handling (OBDH) of SONATE-2 to reduce the overall complexity of the data management system. With the help of these protocols, the development effort could be reduced, and the data transfers simplified. In addition, this allowed the communication interfaces to be simulated and quickly tested during development with a desktop computer using an OBDH-Simulator.
Another important advantage of the developed solution is flexibility and abstraction of the content of the data to be transferred. A practical example of this capability is the reception of software updates. Usually, even small changes require completely new software images or individual applications to be transferred to the satellite. This takes a lot of time and moreover demands a stable radio connection, which is not always available, especially with CubeSats. For this reason, SONATE-2 offers the additional option of uploading delta updates. The following paper describes the developed data handling strategies for the reliable and bidirectional transfer of mission data via one of the three available radio communication links.

1 OVERVIEW of SONATE-2

SONATE-2 is a 6U+-CubeSat which was successfully launched into a 527x505 km SSO on SpaceX's Transporter 10 mission on March 4th, 2024. The satellite was developed by the University of Würzburg. An image of the flight model of SONATE-2 is depicted on the left side of Figure 1. The main goal of the mission is to verify a novel AI processing platform in LEO. Experiments on this platform cover a wide range of applications, including lightning detection, image segmentation, object detection and anomaly detection. Two wide-angle cameras and two narrow-angle cameras are integrated at each of the two AI payloads to carry out the experiments. Images recorded by these cameras are used directly for the training and execution of neural networks on board SONATE-2. Other mission objectives include the operation of an amateur radio payload, the testing of a pulse

plasma thruster PETRUS [1] and the qualification of MultiView [2], a multi-sensor star sensor system.

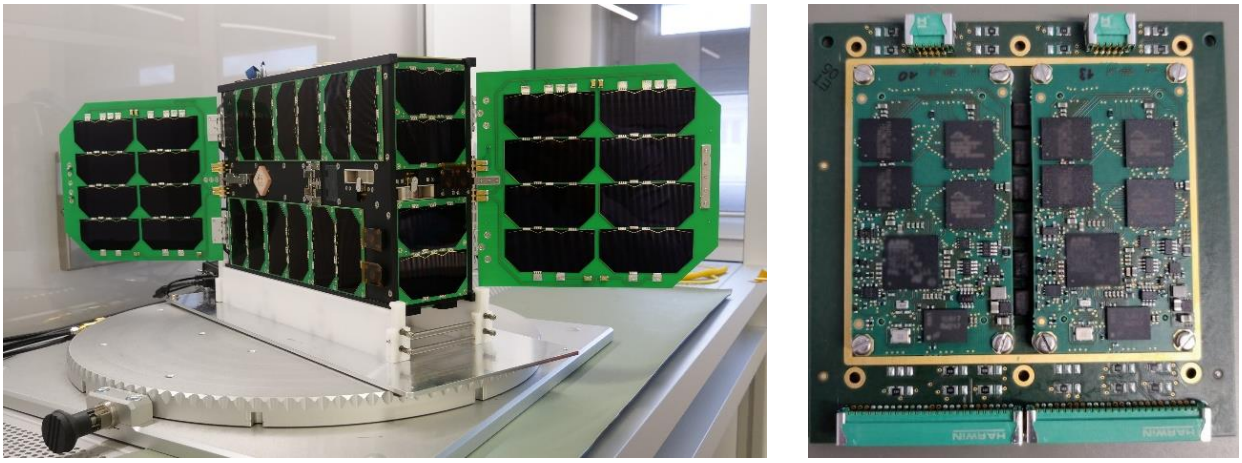

Figure 1: Flight Model of SONATE-2 (left) and two OBDHs of SONATE-2 (right)

SONATE-2 features a redundant bus architecture to achieve the mission goals. The satellite bus consists of two VHF, two UHF and two S-band transceivers, which all can be used for telecommand reception and telemetry transmission. All subsystems are interconnected with each other via both CAN busses, which also acts as the main communication medium between individual subsystems. All telecommands and telemetry is transferred via these two busses. A total number of four On-board data handling computers (OBDH) are integrated into the satellite. The OBDHs are designed to ensure that at least two of them are always switched on and the other two serve as a cold redundant reserve. Each of the OBDH has access to three external NOR flash devices and uses an Arm Cortex M4 as a sole processor. The right side of Figure 1 shows two OBDHs mounted on their interface card. SONATE-2's OBDH use a modified version of the real-time operating system RODOS which was developed mainly for satellite missions [3]. Novell generic data processing strategies needed to be developed at the OBDH to achieve the mission objectives mentioned before. This paper summarizes the implemented strategies, but also gives some insight into the first weeks of operations from the OBDH's perspective. A detailed overview of the first results of the mission can be found in [4].

2 TELEMETRY and TELECOMMAND HANDLING

On SONATE-2 all subsystems, except of the two S-Band transceivers, are directly connected to one or both CAN busses. For this reason, the decision was made to implement generic telemetry and telecommand protocols, simplifying the implementation effort at the OBDH. Instead of implementing a specific interface for every subsystem only one interface needs to be implemented. This significantly reduced the development time and decreased the potential for critical errors in the overall software design.

2.1 TELEMETRY HANDLING

The design of SONATE-2's telemetry interface is largely based on the CCSDS standards of the TM Space Data Link Protocol (CCSDS 132.0-B-3) and of the Space Packet Protocol (CCSDS 133.0-B-2) [5, 6]. Telemetry frames transmitted by SONATE-2 to the ground segment are packed by the OBDH into CCSDS source packages and these then into telemetry transfer frames. In accordance with the standard, various application process identifiers (APIDs) have been defined on SONATE-2. Each APID contains data from one or several different subsystems. A unique CAN ID is assigned to each APID used on SONATE-2 via the telemetry database. By broadcasting this CAN ID through the

OBDH on the satellite bus, all subsystems are requested to provide their telemetry data for the corresponding APID. Upon reception of the CAN ID each subsystem sends out one or multiple CAN frames for this APID. The OBDH assembles the individual responses into one complete source package. The individual telemetry parameters of each response are assigned to one corresponding bit position in the source package via the telemetry database. An export tool was written that generates source code from the definition of the telemetry database to simplify and automate the mapping process. The export defines which bit in the CAN response frame is to be copied to which bit position in the source package. The generated telemetry frames are stored onboard in a ring buffer in the flash. During ground station contacts, both online and historical data stored in the buffer can be sent to Earth via UHF or S-band.
Due to the simplicity of the protocol, an OBDH-Simulator was implemented on the PC. This reads the current definition of the telemetry database at startup and uses the information contained therein to create the telemetry frames. The OBDH-Simulator can be connected directly to the satellite's FlatSat via a USB-to-CAN connection and can therefore be used for the development of subsystems. This tool played a key part in the success of the mission because problems at the telemetry interfaces were identified at an early stage. End-to-end tests with the subsystems could thus be completed quickly and successfully. During testing and integration of the engineering and flight models no issues in the telemetry interfaces were found.

2.2 TELECOMMAND HANDLING

In addition to telemetry processing, telecommand processing has also been greatly simplified in comparison with the predecessor mission SONATE-1. Each subsystem has been assigned one or more specific CAN IDs which the subsystem uses to receive telecommands. One CAN message contains up to eight data bytes. The first byte in the message is used as a command identifier (CMD CODE) for the subsystem and the following 7 Byte in the CAN message can be used as parameters of the telecommand if necessary. All telecommands can be sent to one or both redundant components. The CAN ID, CMD CODE as well as all telecommand parameter are managed in a dedicated telecommand database. It is possible to transmit any of the telecommands to the satellite via VHF, UHF or S-band. The successful reception of the telecommand is controlled on the satellite via a command counter. A command is only executed if its command counter matches the value of the command counter at the satellite. The OBDH of SONATE-2 supports immediate as well as time-tagged telecommands. When time-tagged telecommands are uploaded, they are first saved with an absolute or relative timestamp in a passive time-tagged list. All commands in this list must have relative or absolute timestamps. It is not possible to mix the timestamp type in time-tagged lists. As soon as all the commands for a procedure have been uploaded, the passive list is copied to an active list using an immediate telecommand. For relative timestamps in the passive list, the execution time is also specified. This ensures that only absolute timestamps are used in the active time-tagged list. The OBDH then executes the individual commands in the active list at the specified timestamps.
The OBDH-Simulator also supports the execution of immediate or time tagged telecommands with the previously described format. During the development of a subsystem, it is quickly possible to enter telecommands into the database and send them via the OBDH-Simulator. Instead of implementing own interfaces to support the development of the subsystem, flight interfaces can be used, which reduces the development time. Another advantage of this approach is that errors in the telecommand definition are already identified at subsystem level, prior to complete end-to-end test with the real OBDH hardware.

3 SOFTWARE UPLOADS

In the case that software of one of the satellite components needs to be replaced or modified, the OBDH of SONATE-2 also has the task of receiving software uploads for all the embedded

microprocessors of the satellite and applications of the AI payload. For redundancy reasons, each of the software images was stored three times on each of the four OBDHs. There are two different options for the upload process. One option is to upload the complete new software image. This process is referred to in the following as a *classic software update*. Alternatively, it is also possible to upload a delta between the software image stored on board and a new software image created on the ground. Such updates are further referred to as *delta software updates*.

3.1 CLASSIC SOFTWARE UPDATE

Once the new software was successfully tested on the ground, it must be transferred to the satellite. For this purpose, the complete software image is divided into smaller packages. These packages are then transmitted as individual command blocks via one of the three available communication links to the satellite. The OBDH receives these individual packages and saves them in an external flash memory. Upon completion of this upload process the OBDH checks whether the checksum of the uploaded image corresponds to the checksum previously pre-calculated on the ground. If this is the case, the component or subsystem can be reprogrammed with this new software image. To do this, the complete software image is transferred to and installed at the subsystem via the satellite's CAN bus. Figure 2 shows this process.

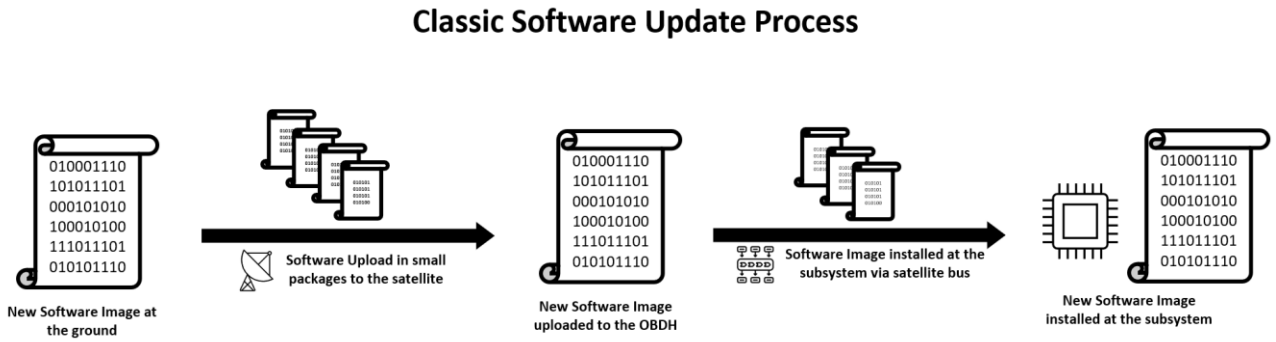

Figure 2. Steps of a classic software update

3.2 DELTA SOFTWARE UPDATE

Contrary to classic software updates, delta updates, require the difference (delta) between the currently installed software image and the new software image to be uploaded to the satellite. This has certain advantages if the affected software image is large, and the available communication bandwidth for an upload is low. Delta updates are very widespread nowadays and are for example used to update Android apps on smartphones and tablets but are still rarely used on embedded systems. [7]

For the calculation of delta updates for binary executable applications, a strategy called sequential patching is used. The overall software patch is divided into multiple repetitive sequences each consisting of a difference, extra and adjustment section. Figure 3 shows this general structure. The difference section contains the changes between the old and the new software image. Information that was not available in the old software image is included in the extra section. For data that has not changed in the new software, the adjustment section is used. Since addition in embedded software mostly only affect the addresses of the existing software, the difference section in particular consists of highly repetitive data. Compression algorithms are therefore highly effective to reduce the overall size of the software patch. [8]

Diff 1	Extra 1	Adj 1	Diff 2	Extra 2	Adj 2	…

Figure 3: Structure of a sequential patch [8]

For the compression of the individual patches the open-source library “heat-shrink” was used. Heat-shrink is a compression library especially designed for the unique constrains of embedded systems having limited memory and working without dynamic memory allocation. [9]
Making use of these algorithms a delta image is computed on ground and uploaded to the satellite. This delta image is usually much smaller in size, if the changes in the software are not significant, compared to uploading the entire new software image. After the upload is complete the information contained in the delta patch as well as the on board saved old software image are used to create the new software image. Therefore, after decompression, the individual sections in the delta patch are sequentially looped until all changes are applied. Thereby, the data contained in the difference section is simply added byte by byte to the old image in order to compute the new image. The bytes in the extra section are copied afterwards to the new image. The adjustment section contains the number of bytes which are unchanged, so these bytes are directly copied from the old image to the new image. After all these steps the new image is reconstructed at the satellite. After verifying the integrity of the software image using a CRC, the subsystem can be reprogrammed. Figure 4 shows the individual steps graphically.

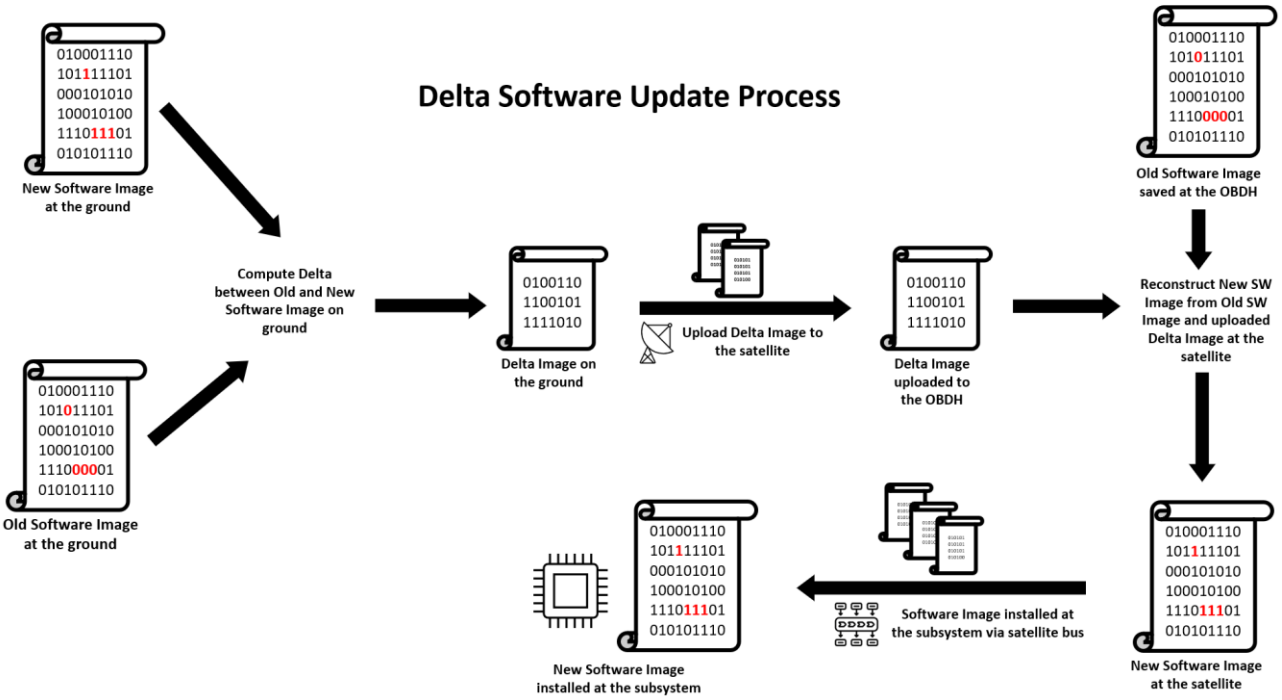

Figure 4. Steps of a delta software update

3.3 COMPARISION between DELTA and CLASSIC SOFWARE UPDATES

On CubeSat missions, UHF (either commercial or amateur radio frequencies) is typically used as the main communication channel for the downlink and uplink. This frequency range is subject to strong interference, especially over Europe, Asia, and North America, which contributes to the fact that not all transmitted telecommands are received by the satellite. In our predecessor mission SONATE, which also used UHF as the main radio link, mostly less than 10% of the commands sent to the satellite were received. In absolute figures, this translates into only around 20 – 30 commands per overpass being received by the satellite. Also, ESAs OPS-SAT mission encountered similar challenges in LEOP. [10 – 14]
A classical software update of a software image having for example 100 kByte of data would take at least one week, if only 3 kByte of data can be uploaded per overpass. In case of critical problems which need to be addressed with this update, a quick response is therefore not easily possible and depending on the severity of the problem a mission loss or significant mission delay likely. On ESAs

OPS-SAT mission for example 5000 telecommand were required to update the board computer software, which took about a week. [14]
On SONATE-2 we have therefore decided to additionally implement delta updates to be able to quickly update the affected subsystem in the event of software bugs being detected after the launch in orbit or a new feature is required in the flight software.
As an example, in the first overpasses over our ground station the number of decoded telemetry frames were lower than expected even though the measured signal strength appeared to be high enough. Possible solutions were investigated to increase the number of decoded frames. One idea was to increase the number of synchronization bytes the UHF radio is transmitting before a telemetry frame. Unfortunately, during ground testing it was noticed that this specific telecommand was not working as intended. After a deeper code analysis, we noticed that two lines of code were swapped around. Therefore, it was not possible to increase the number of synchronization bytes without a software update. The corrected software image has a size of 53704 Bytes. By using delta software updates only 1227 Bytes are required to be uploaded to the satellite. In this case the required bandwidth and upload duration could be reduced by 97.7%.
This example shows that delta updates are especially useful if you are working on limited bandwidth, e.g. if the satellite has a high spin rate, a poor communication link with the ground segment, or you are operating in deep space. The big advantage of delta updates compared to classical software updates is the small image size required for a software update. By small changes in the flight software only a few kilobytes need to be uploaded to the satellite. This saves time and costs if for example external commercial ground stations are used to speed up the upload process. Especially in LEOP, when the orbit data is preliminary, disruptions in communication with the satellite are not uncommon. Delta updates are particularly useful in this phase if problems that have not yet been considered in the software version installed on the satellite require a quick reaction by the operation team. Quite often either parameter updates or complete software updates are then required to fix or further debug the problems observed. The delta update process comes however with the disadvantage of a slightly higher memory demand, because the old software image, the delta update and the new software image must be saved on board. Also, the computation effort for delta updates is slightly higher than for classical updates because the new software image must be regenerated on board. Overall, it can be said that the advantages of uploading delta updates clearly outweigh the disadvantages. The following Table 1 summarizes the advantages and disadvantages of classical compared to delta software updates.

Table 1. Comparison of Delta Updates vs. Classical Updates (Pros: green/++; Cons: red/--)

	CLASSICAL UPDATES	DELTA UPDATES
UPLOAD BANDWIDTH	--	++
TIME CONSUMPTION	--	++
COSTS	-	+
MEMORY DEMAND	+	-
COMPUTATION EFFORT	+	-
POWER DEMAND	0	0

3.4 SOFTWARE UPDATES

SONATE-2 includes a total of 46 microprocessors, all of which can be reprogrammed in orbit. A standardized software update protocol was developed and implemented in the bootloaders of the microprocessors to achieve this goal. Each of the processors was assigned a unique identifier (Boot-ID) via the bootloader. This approach simplified the implementation efforts of software update process on the OBDH, because the same protocol is utilized for reprogramming. Only the size, memory address and checksum as well as the Boot-ID of the microprocessor to be updated must be transferred to the OBDH.

As all updates are carried out via the CAN busses and these are connected to the PC via the EGSE, the update process could also be carried out via a special EGSE-Software. In the EGSE-Software the same update protocol was implemented as in the OBDH's software. Software updates were therefore possible very quickly from the outside. Updates were even possible, when the debug ports of the subsystems were no longer accessible once the satellite was in flight-ready state.

4 PAYLOAD DATA HANDLING

Another important aspect of the OBDH on SONATE-2 include receiving experiment results from the various payloads and subsystems on board. It must also be possible to forward new configuration parameters to the subsystems, especially for the AI payload. The configuration parameters of the AI include for example IDs of images which should be used for the on-board training of neural networks and can therefore be very large. In this context, a dedicated generic payload data processing software module was implemented in the OBDH on SONATE-2.

4.1 LOW-SPEED DATA TRANSFERS

Since all subsystems of SONATE-2 are connected to one or both CAN busses a generic payload data transfer protocol was developed. The protocol was designed in a way that it can be used for all subsystems which either require larger amount of configuration parameters not suitable for individual telecommands or need to transmit locally saved experiment data to the OBDH. The data flow of the protocol is depicted in Figure 5. The start of a transfer is initiated by a telecommand to the subsystem. This gives the subsystem the opportunity to carry out preparatory tasks such as clearing memory or calculating checksums. The actual data transfer begins with a subsequent command to the OBDH. In the first step, the size and checksum of the complete data block are exchanged via the transfer start messages. The actual data transfer is then initialized by a corresponding request or announcement from the OBDH. The transfer takes place in packets of 256 bytes, each of these packets is protected by its own checksum. In the event of errors during the transfer, a single packet can therefore be requested again. Transfer rates of up to 3300 Byte/s could be achieved with this protocol.

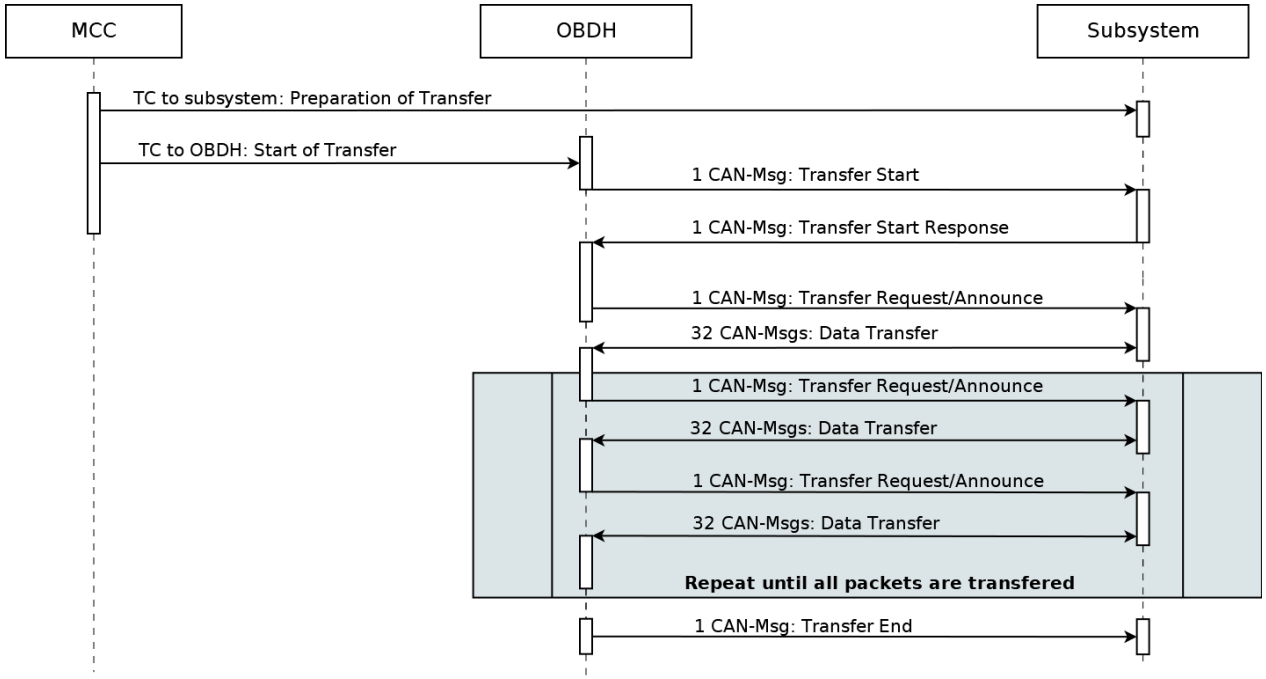

Figure 5. Data flow of the low-speed data transfer via the CAN-Bus

This protocol has already been extensively tested in orbit, with the first image data from the AI payload being transferred to the on-board computer. A subsequent transfer of one of the images from the OBDH to our amateur radio payload was also successful. The protocol also proved to be very useful when it was noticed in orbit that the direction of the control signals to the reaction wheels were partially inverted. In order to reverse the direction of the reaction wheels, it was necessary to verify the currently set direction, set in the ADCS' configuration. As these values were not included in the standard telemetry, the configuration of ADCS was transmitted as a binary to the OBDH and then to the ground via the generic protocol. With the help of this generic protocol the correct direction could be set. Furthermore, the developed protocol allows new experiments to be carried out on the individual subsystems in the future without having to make changes to the OBDH's flight software.

4.2 HIGH-SPEED DATA TRANSFERS

Besides data transfer via CAN, there is also the option of transferring experiment data from the AI payload via SPI to the OBDH. For this interface, a bidirectional communication protocol has also been implemented. The protocol again starts with a simple handshake were the size of the data in bytes, the CRC and the size of a transfer packet are exchanged between the two participants. Each package is identified by a packet number and protected with a CRC. After receiving a package an acknowledgment (ACK) message is returned if the CRC and packet number match the expected ones. In the case of an invalid CRC is received a not acknowledgment (NACK) is returned. The sending end then retransmits the same packet again up to a certain amount of retries. If all packets were transferred the receiving end calculates the checksum and the transfer is finished. This protocol is used for transferring large amount of data, e.g. to upload new applications or neural networks to the AI payload or download experiment results like images and classified scenes. Transfer rates of up to 220 kByte/s have been measured.
For development purposes a SPI-to-USB FTDI converter was used to integrate this protocol in the EGSE-Software. During development and initial system tests of the AI payload, direct transfers of experiment results to the PC were then possible using the same protocols as in the flight version. Moreover, it was also possible to transfer images without using radio communication after qualification and acceptance tests of our flight model to the PC for analysis purposes. This allowed us to quickly verify that our optical components survived the random vibration and shock tests without any visible defects.

5 TIME MANAGEMENT

Initial task of any OBDH is to provide all subsystems and payloads with an accurate timestamp. These timestamps are especially required for the SGP4, sun and magnetic field reference models of the ADCS to determine the correct attitude at the correct time over predefined target areas to accomplish the mission objectives. For the analysis of the image data of the AI payload and the performance of the ADCS, it is also important to know exactly when an image was taken to compare any pointing errors of the ADCS between the predicted and the observed scene.

5.1 TIME KEEPING

The on-board reference time is provided by a real-time clock (RTC) integrated in the OBDH processor, which is driven by an external quartz crystal. The advantage of RTCs compared to timers, which are often used in embedded software development, is the possibility to correct the natural frequency error of the quartz crystal and thus the time drift by manually setting special hardware registers. Another essential advantage is that the time continues to be incremented even if the processor is reset. Should the RTC malfunction during the planned one-year operating phase of the satellite, the time is also propagated via a normal timer for safety reasons.

5.2 TIME UPDATE

There are two different ways to set the reference time on board of SONATE-2. On the one hand, the onboard time can be set via a telecommand, whereby the current timestamp of the telecommand client is transmitted to the satellite. This telecommand can be sent via the three available frequency bands VHF, UHF and S-band. The disadvantage of this method is an unknown time offset between command transmission until the reference time is set on the OBDH.
On the other hand, the reference time can also be taken from one of the two GNSS receivers. This allows a very precise synchronization of the time but requires a stable GNSS fix. In the first GNSS experiments, as the satellite was not detumbled (under 1°/s), stable GNSS fixes could not be obtained. Since SONATE-2 was successfully detumbled, a stable GNSS fixes was achieved. The GNSS position and time were successfully verified. In the future it is planned to use the measured GNSS time to update the OBDH's reference time.
The following Figure 6 shows all possibilities to update the reference time of the OBDH of SONATE-2 in orbit. The violet, yellow and orange solid arrows symbolize the path of the reference time from mission control center (MCC) via ground segment via radio transceivers via the satellite bus to the OBDH. Whereas the green dotted lines represent the path from the GNSS receiver to the OBDH.

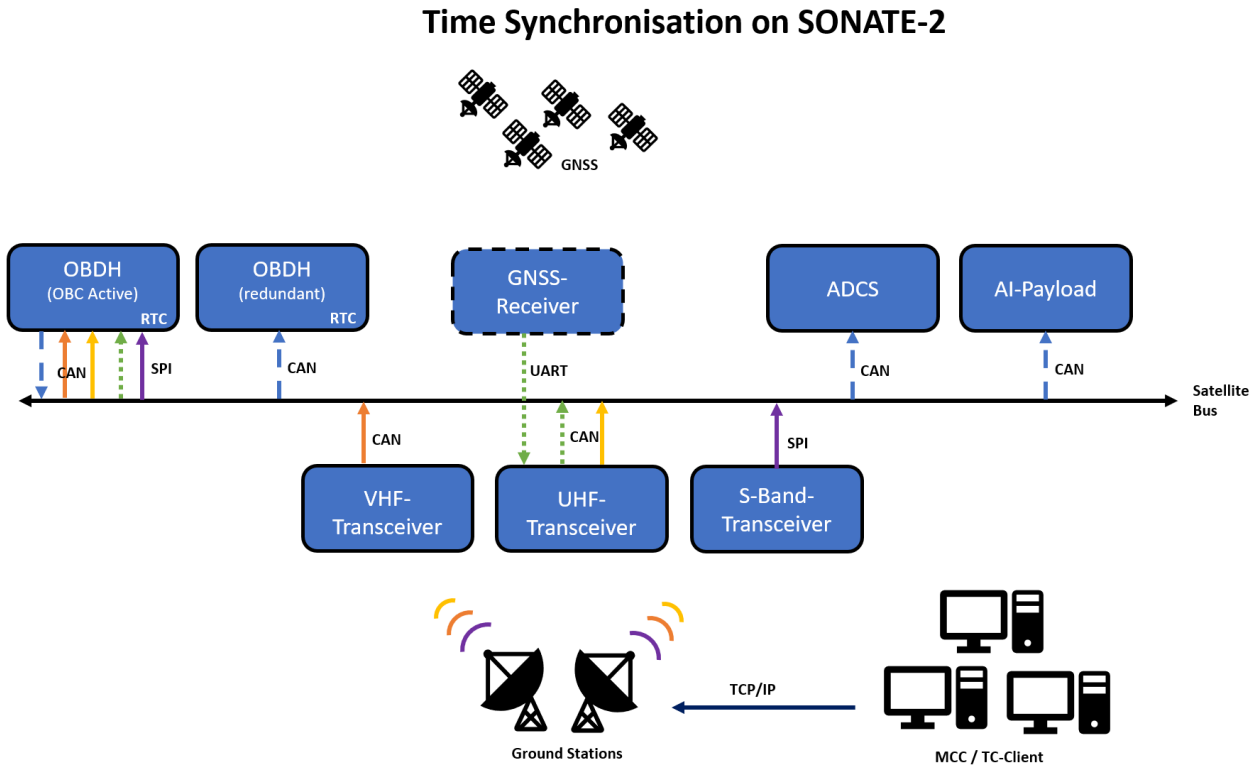

Figure 6.Overview of the available time synchronization paths on SONATE-2

5.3 TIME DISTRIBUTION

For the distribution of the reference time on the satellites, the OBDH periodically transmits the current onboard time in millisecond resolution on both CAN buses every 5 seconds. Subsystems that require this time can listen to the CAN message and thus correct their time to the broadcasted reference time. These paths are also shown with blue stitched arrows in Figure 6. Between the individual CAN messages, each subsystem must propagate its own time in order to remain up to date.

5.4 TIME DRIFT CORRECTION

In orbit, we observed a slight drift in the reference time during daily operation. In LEOP, the reference time was thus set by telecommands during each overpass over our ground station in Würzburg. A

more precise analysis of the received timestamp to the timestamp generated on board in the telemetry frames showed a linear drift of approx. 800 ms per day. This correlation can be clearly seen in Figure 1, where the time difference between the received and generated time was plotted over several contact phases.

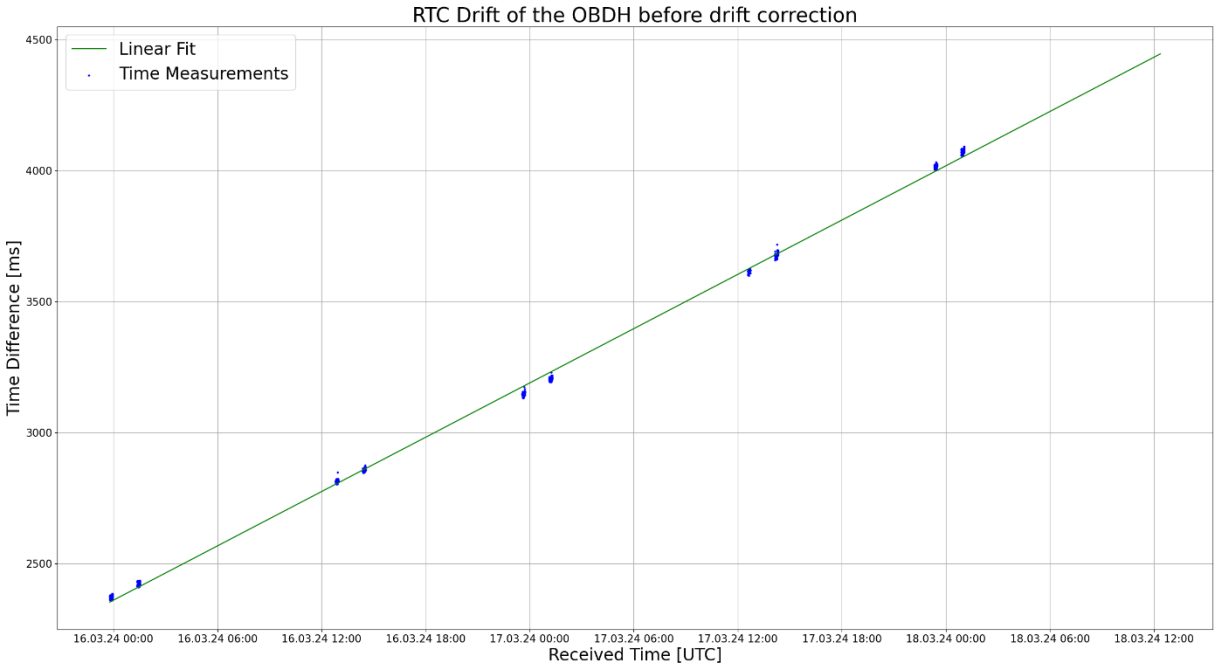

Figure 7. Drift of the reference time before the drift correction

A correction value for the RTC was then calculated from the slope of the linear fit and sent to the satellite. In the following days, the time difference between the received and generated time was recorded again. The result of this experiment is shown in Figure 8. After the correction was applied, no signification drift of the reference time was observed anymore, and the time error stayed always less than a second. The slight variation in the order of 30 ms between the individual overpasses are likely caused by temperature variation at the crystal. In summary, it can be said that drift correction has significantly reduced the time drift and thus allows in the future a more precise execution time of the planned experiments.

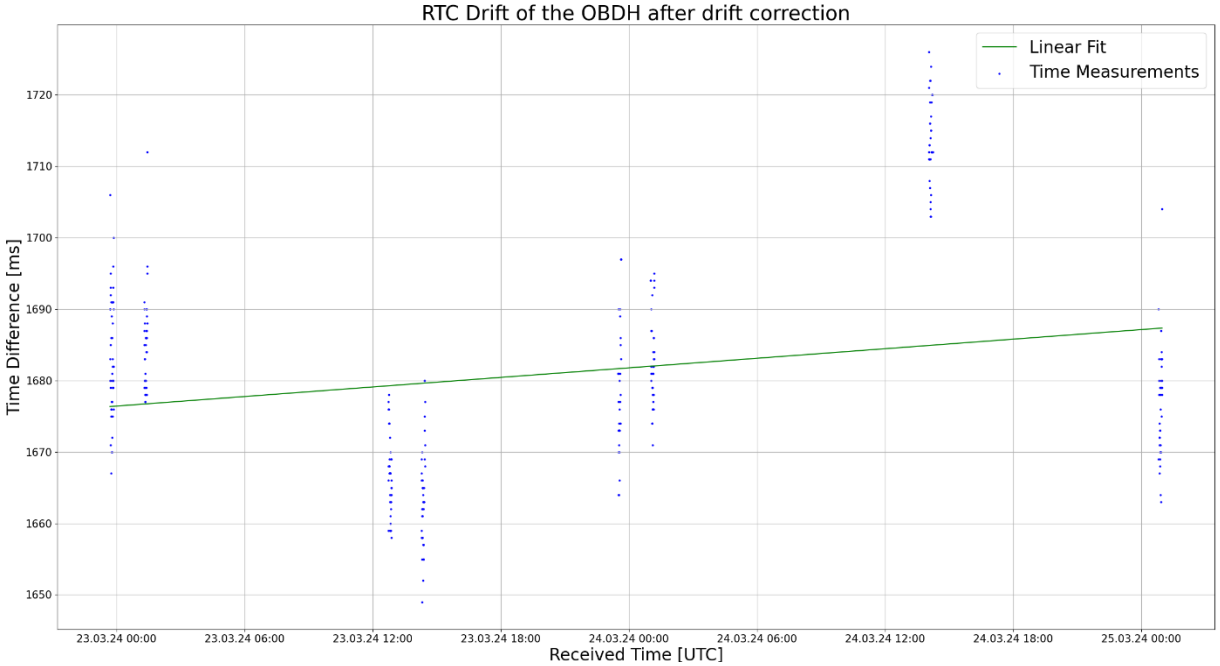

Figure 8. Drift of the reference time after the drift correction

6 CONCLUSION

This paper gave a brief overview about the generic data processing strategies implemented at the OBDH of SONATE-2. By using these generic protocols more than 3000 immediate and time-tagged telecommands were successfully received and executed by SONATE-2 in the first month of operation. During this time also more than 150000 unique source packages have been transmitted and received in Würzburg by our own ground station. The OBDH was running thereby continuously without any problem. After 37 days in orbit the first reset of the OBDH is still to come. Already the first images of the AI payload were transferred to the OBDH and later to the ground using the generic protocols. One of these images was also forwarded to our radio amateur payload and transmitted to Earth as a SSTV image over Easter. It was furthermore possible to correct the OBDH's time drift, allowing a more precise execution timestamp of the planned experiments.
Additionally, a new software update technique was introduced. With the help of delta updates the required bandwidth and upload duration of software updates can be reduced significantly by more than 95%. Delta updates are especially useful for future interplanetary CubeSat missions where the communication bandwidths are much lower than in LEO.
All of these points clearly demonstrate the stability and reliability of the implemented generic data processing interfaces. The developed protocols together with the implemented OBDH-Simulator and EGSE-Software enabled errors in the database, and software to be identified and corrected at an early stage. This could already be done in the development phase before complete end-to-end tests had to be carried out with the OBDH. It is important to emphasize that the use of the discussed protocols together with stable time management contributed significantly to the successful execution of the satellite project.

7 ACKNOWLEDGEMENTS

SONATE-2 is funded by the German Federal Ministry of Economic Affairs and Climate Action, represented by the German Space Agency DLR (FKZ 50RU2100), on the basis of a decision by the German Bundestag.

8 REFERENCES

[1] Schäfer F., et al. *In-Orbit Testing of PETRUS Pulsed Plasma Thruster on the GREENCUBE 3U Cubesat*, 8th Edition of the Space Propulsion Conference (SPC), Estoril, Portugal, 2022.

[2] Neumann, T. et al., *MultiView: Koordinierter Sternsensorverbund im TunaCan-Format für CubeSats*, in Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., Bonn, 2022.

[3] Montenegro S. and Dannemann F., *RODOS - real time kernel design for dependability*, Data Systems in Aerospace, no. 669, p. 66, 2009.

[4] Schwarz T., et al. *Early results and In-Flight Experience of the 6U-Misssion SONATE-2*, The 4S Symposium 2024, Palma de Mallorca, Spain, 2024.

[5] The Consultative Committee for Space Data Systems (CCSDS), Space Packet Protocol, CCSDS 133.0-B-2, June 2020.

[6] The Consultative Committee for Space Data Systems (CCSDS), TM Space Data Link Protocol, CCSDS 132.0-B-3, October 2021.

[7] Morris A. and Hayden A., *Improvements for smaller app downloads on Google Play*, 22 July 2016. Available: https://android-developers.googleblog.com/2016/07/improvements-for-smaller-app-downloads.html. [Accessed 05 April 2024].

[8] Lindh L., *Delta Updats for Embedded Systems,* Chalmers University of Technology, Gothenburg, Sweden, 2021.

[9] Atomic Object, *Heatshrink,* GitHub repository, 2015. Available: https://github.com/atomicobject/heatshrink. [Accessed 05 April 2024]

[10] NASA Ames Research Center, *State-of-the-Art of Small Spacecraft Technology*, 2023. Available: https://www.nasa.gov/wp-content/uploads/2024/03/soa-2023.pdf?emrc=7affa4. [Accessed 05 April 2024].

[11] Busch S., et al., *UWE-3, In-orbit performance and lessons learned of a modular and flexible satellite bus for future pico-satellite formations,* Acta Astronautica, no. 117, pp. 73-89, 2015.

[12] Quintana-Diaz G., et al. *Detection of radio interference in the UHF amateur radio band with the Serpens satellite*, Advances in Space Research, Vol. 69, 1159-1169, 2021.

[13] Schwarz T., et al., *In-Flight Experience and Operations of the 3U-Mission SONATE*, in Small Satellites Systems & Services (4S) Symposium, Vilmoura, Portugal, 2022.

[14] Evans D., et al., *OPS-SAT LEOP and Comissioning: Running a Nanosatellite Project in a Space Agengy Context*, in Small Satellite Conference, Utah State University, Logan, UT, 2022.