Software for the operation of a sounding rocket experiment with a *jamming gripper*

Aleksander Bojda

February 7, 2019

Intermediate Project (instructed by Witold Paluszyński Ph.D)
Department of Cybernetics and Robotics, Wrocław University of Science and Technology

Abstract

The aim of the project was to design and implement the embedded software for the operation of the TRACZ experiment testing *jamming gripper* in microgravity and vacuum conditions. The experiment will be launched in march 2019 in the Rexus26 rocket as a part of Rexus/Bexus program organized i.a. by European Space Agency, German Aerospace Center and Swedish National Space Agency.

The nature of a rocket experiment requires software to handle both autonomous operation during the flight and asynchronous telecommands in ground tests. Also, in order to carry out the post-flight analysis all gathered data must be (redundantly) stored on SD cards. The device consists mainly of a pneumatic system, a linear motor (with various closed-loop control principles) and numerous sensors (i.a. force, pressure, current, temperature, humidity, IMU, encoders). Electronics is placed on three PCBs (Powerboard, Mainboard and Sensorboard) containing two STM32F405 MCUs – one for the main algorithm and sensor operation, the other used for the operation of additional sensors and redundant storage of data.

All the above assumptions and the requirements related to safety, reliability and thorough testing form a complex and challenging system.

All main goals of the project were achieved – resulting software i.a. handles operation of various sensors, acquires data, stores measurements on SD cards and enables two-way communication between device and ground (telemetry) station. Gripper's movements and gripping is possible thanks to the motor control (with dynamic switching between position, force and current feedback) and the pressure control (two valves – inlet and outlet) with the feedback from the differential pressure sensor. Also additional equipment including cameras with TV Channel transmission, electromagnetic lock, illumination and cameras synchronization system is implemented.

Software was thoroughly tested both during the implementation and verification phases of the project and during the rocket payload integration tests. During the verification phase minor problems were eliminated and during the integration tests device was operating nominally.

1 Introduction

TRACZ[1] project is a result of a collaboration born in October 2017 between students from two student research groups at Wroclaw University of Science and Technology – KoNaR and Space is More. The primary goal of the project is to test the catching of the object in the space-like conditions (vacuum and microgravity) using jamming gripper. The project is realized within Rexus/Bexus programme[2]. The REXUS/BEXUS programme[1] is organized under a bilateral Agency Agreement between the German Aerospace Center (DLR) and the Swedish National Space Agency (SNSA). 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). Experts from DLR, SSC[3], ZARM[4] and ESA provide technical support to the student teams throughout the project.

Jamming gripper is particular case of pneumatic gripper which is based on jamming transition in granular material. Pressurized membrane (i.e. neoprene in latex form) filled with granular medium (i.e. ground coffee) is adjusting to object, then depressurizing of membrane is performed and granular medium jams - the object gets gripped (figure 1).

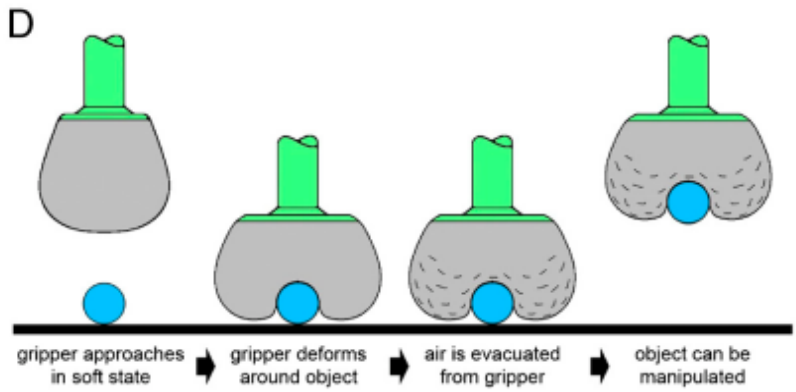

Figure 1: Operation of the jamming gripper

TRACZ aims to investigate the possibility of application of such gripper in space, where negative differential pressure is impossible to obtain and lack of gravitation may cause the granular substance inside the gripper to behave in an unpredictable manner. During REXUS flight, in microgravity and vacuum conditions series of catches will be performed on an single object and the force with which the object is held will be measured. The results will be compared with an on-ground experiment and the utility of the aforementioned gripper in the space applications will be discussed.

In order to conduct the experiment the mechanical, pneumatic and electronics systems were designed from scratch to fulfill the requirements of the experiment, rocket and environment. Resulting construction is presented in the figures 2 and 3.

[1] **T**esting **R**obotic **A**pplications for **C**atching in **Z**ero-g – `www.tracz-rexus.pl`
[2] Rocket and Balloon Experiments for University Students – `www.rexusbexus.net`
[3] Swedish Space Corporation – operator of Esrange Space Center
[4] The Center of Applied Space Technology and Microgravity in Bremen

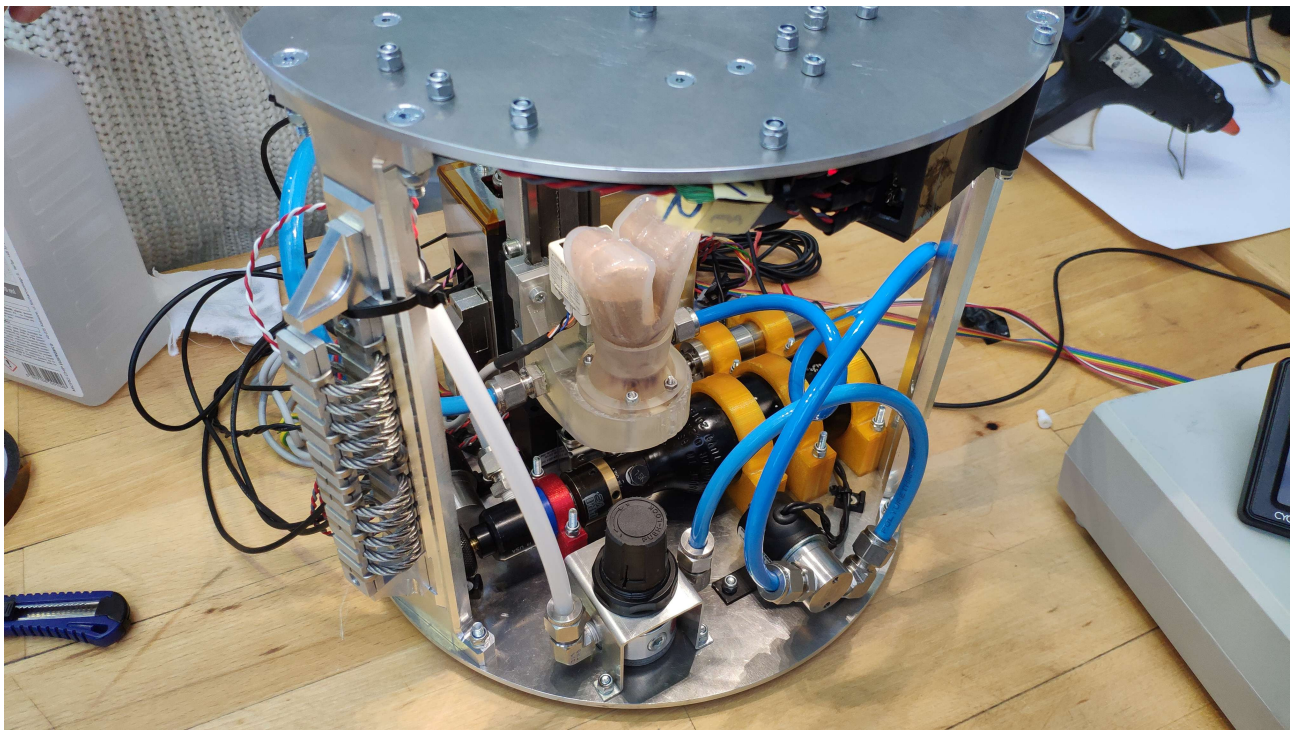

Figure 2: Overview of the hardware

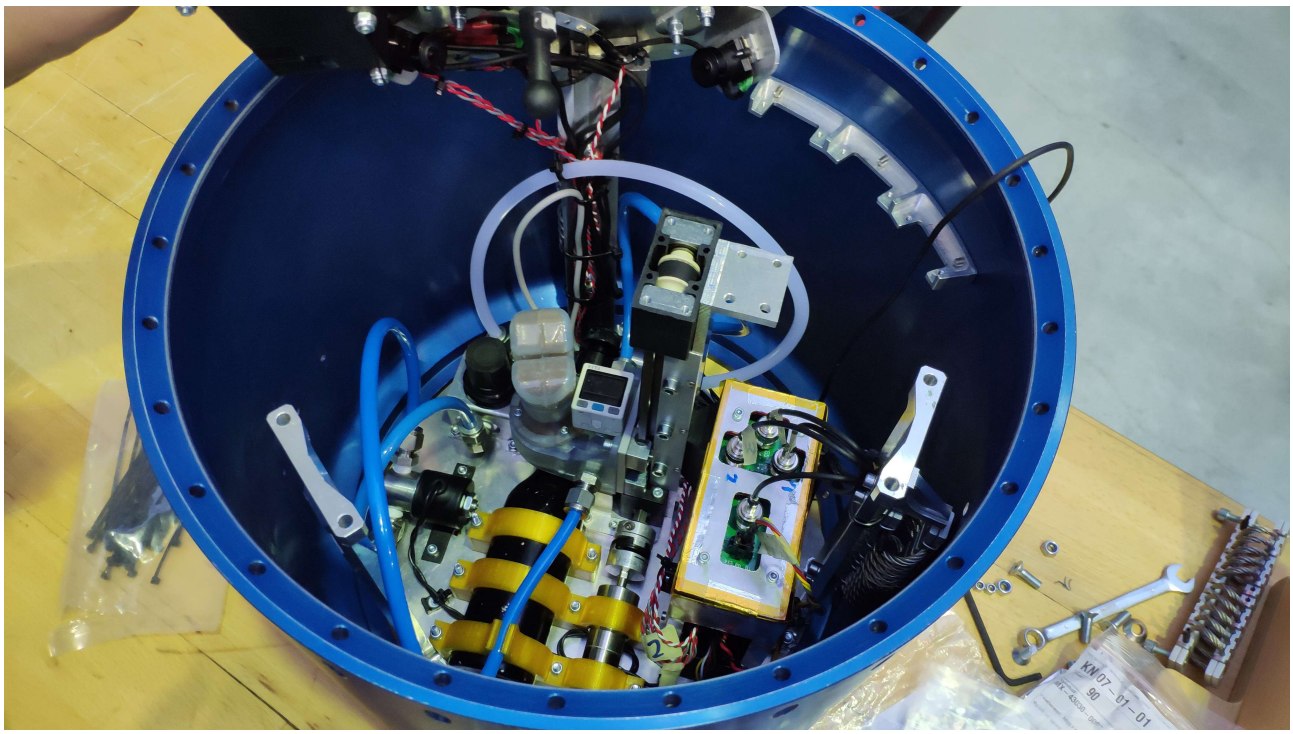

Figure 3: Device during the integration with the rocket module

The device consists of:
Mechanical subsystem:

- Top and bottom plates connected with beams

- Anti-vibration dampers
- Linear guide
- Motor with the coupling
- Mountings and housings

Pneumatic subsystem:

- Bottle with the pressurized air
- Two pressure regulators
- Two valves
- Membrane with the ground coffee inside
- Pneumatic tubes

Electronics subsystem:

- *PowerBoard*:
  - 5V and 12V DC/DC converters
  - 3.3V and 5V linear regulators
  - RS-422 transceiver
- *MainBoard*:
  - Differential pressure sensor
  - Load cell (force sensor)
  - DC motor driver
  - Encoder input
  - Motor current sensor
  - Valve controlling circuits
  - Rexus Service Module signals input
  - Electromagnetic lock output
  - Cameras power output
  - TV channel selection output
  - Illumination output
  - SD Card
  - UART connection with the Rexus Service Module
  - UART connection with the SensorBoard
- *SensorBoard*:
  - Absolute pressure sensor
  - Temperature sensor
  - Hygrometer
  - 3-axis accelerometer
  - 3-axis gyroscope
  - Vision system synchronization LED
  - Rexus Service Module signals input

  - SD Card
  - UART connection with the MainBoard

- *Vision System*:
  - Three camera-lenses
  - Three camera-boards
  - TV Channel connection to the Rexus Service Module
  - TV Channel switching board

All above subsystems require comprehensive control system in order to operate in accordance to the requirements. **Embedded software for the TRACZ experiment is the topic of this Intermediate Project and is described in the following sections**.

2 Project description

2.1 Software overview

2.1.1 Tools used

Software for the project was developed for the hardware platform with two STM32F405RGT6 microcontrollers. Software is based on the FreeRTOS system and makes use of STMicroelectronics' HAL libraries[2].

Simple microcontroller-based systems are usually implemented just as a single loop program with optional interrupts. However complex systems often handle multiple sensors, actuators or peripherals and also require various precisely timed control loops. It is usually still possible to implement such a system with a simple (single loop) software architecture, however often the better solution is to use real time operating system adapted for microcontrollers and small microprocessors. For these purposes, FreeRTOS was selected as it is lightweight, reliable, simple and well documented.

As FreeRTOS[3][4] system is used, software is split into separate, quasi-parallel tasks, each responsible only for the operation of the part of the system. Tasks' access to the processor time is controlled by the scheduler and based on tasks' priorities. In order to control access to the peripherals, synchronize operations and enable inter-task communication FreeRTOS mechanisms such as Queues, Notifications and Event Groups are used. These mechanisms are also used to plan sequences of events, conditions and actions to operate autonomously.

HAL (Hardware Abstraction Layer) library is the solution developed by ST company (manufacturer of the STM32 microcontrollers). Main goal of the library is the simplification and acceleration of the development process by providing the set of unified APIs for all families of ST microcontrollers. Because of this HAL library was used in the TRACZ project together with STM32CubeMX – tool used to generate peripherals configuration code and Atollic TrueSTUDIO for STM32 – IDE with built-in compiler. Some low level functions not supported by HAL library were programmed through operations on the registers.

2.1.2 Algorithm

Algorithm is divided into three main parts – Init State, Test Mode and Flight Mode and implemented as a state machine. On power-up, system starts in Init State. After initialization of all peripherals the software switches to the Flight Mode. Normally, this mode automatically operates the experiment starting from initialization, through consecutive stages (controlled by SODS[5], LO[6] and SOE[7] signals[8])

[5] Start of Data Storage
[6] Liftoff
[7] Start of Experiment
[8] These signals are issued by the Rexus Service Module based on the flight timeline

until end of the experiment. However, if after power-up one of the signal lines (SODS, LO or SOE) is active, the system is automatically put in the proper phase of the flight mode in order to (possibly) recover from a temporary loss of power or accidental reset during the flight.

Test Mode allows to run subsystems' tests both before (software and electronics implementation tests) and during flight campaign (final tests before liftoff). Testing procedure is simplified by use of ground station with control and monitoring software. Around 50 telecommands are available allowing i.e. to move gripper to the desired position, turn on/off any of the cameras or set the parameters of any of PID controllers. Test Mode routine is presented in the figure 4.

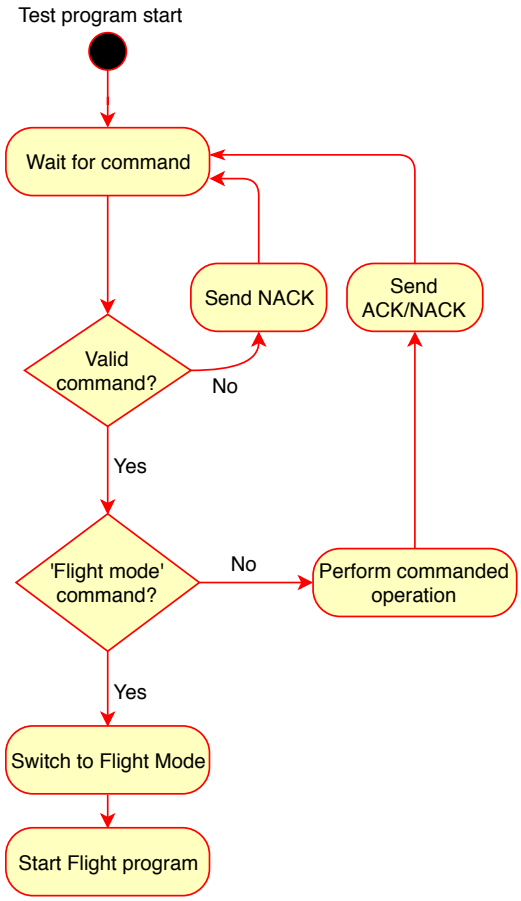

Figure 4: Test Mode algorithm

Flight Mode is used during flight. It can be separated into two sub-modes: pre-experiment operation (RXSM signals handling, start of data storage and gripper's initial state maintenance – presented in figure 5) and actual experiment (gripper and valves operation, pulling force tests and preparation for landing – presented in figure 6).

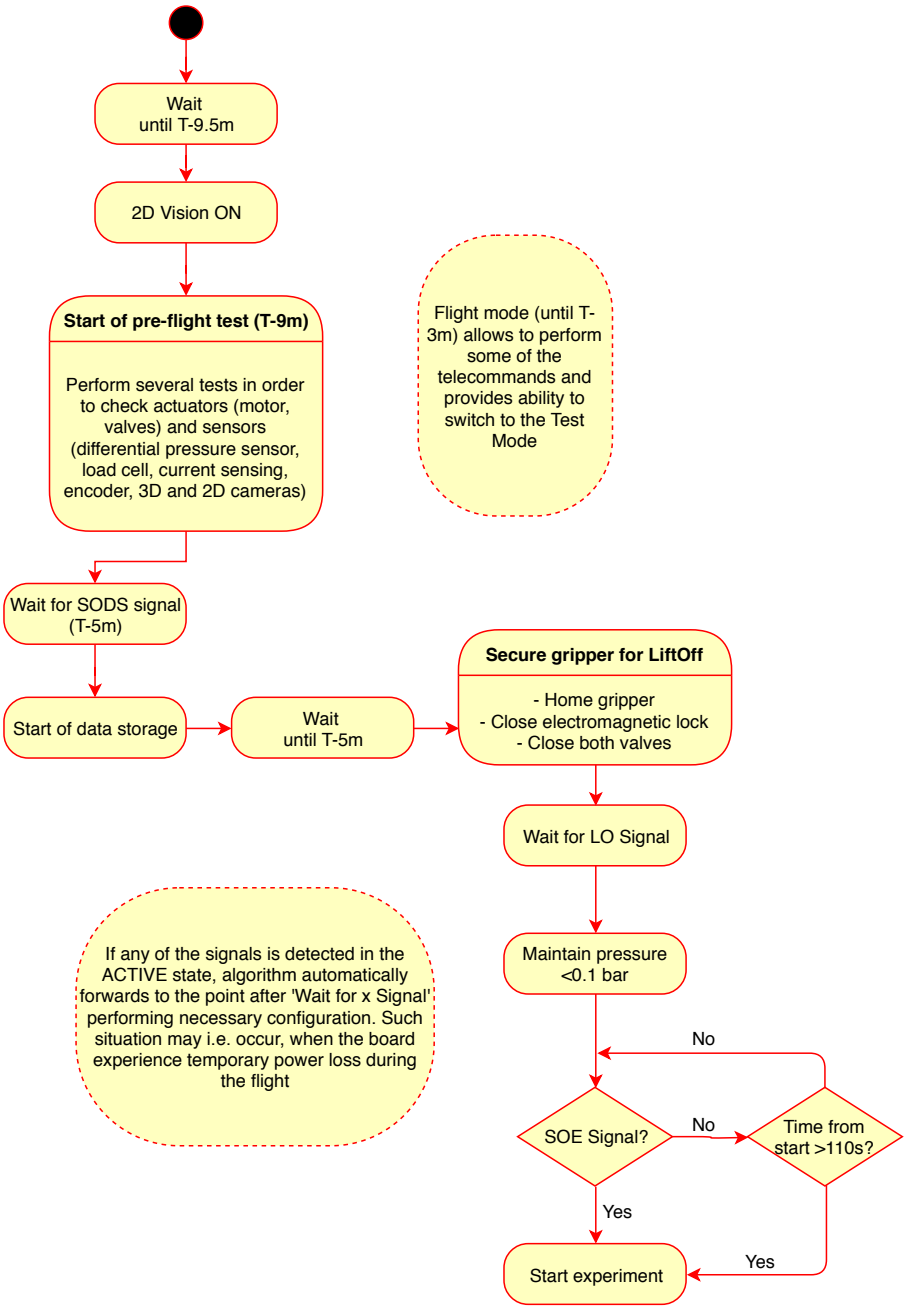

Figure 5: Flight Mode algorithm

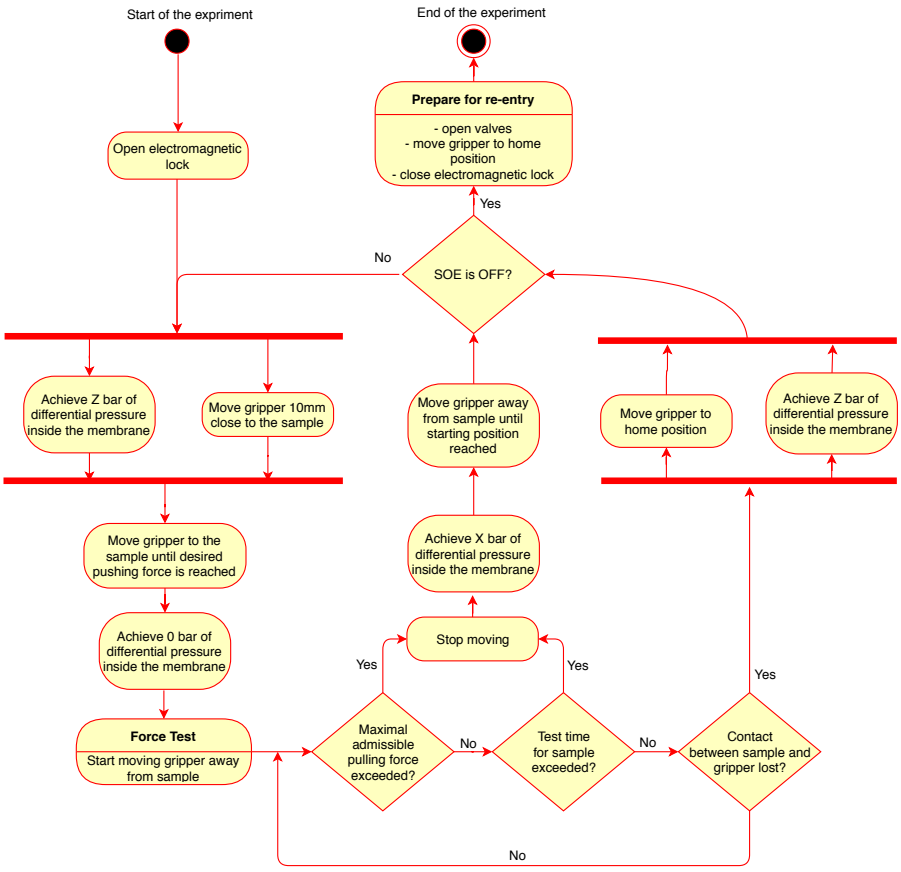

Figure 6: Experiment algorithm

2.2 Communication and data flow

General data flow of the experiment and used communication standards are presented in the figure 7. All data from the MainBoard and the SensorBoard (encoded in data frames) is transmitted to the ground station through RXSM downlink (using PowerBoard as a signal translator). Data is also saved locally on boards' SD Cards. Moreover the data from the MainBoard is sent to the SensorBoard and stored on SB's SD card. Also the data from SB is sent to the MB and stored on MB's SD card. This mechanism provides redundancy of the data acquisition.

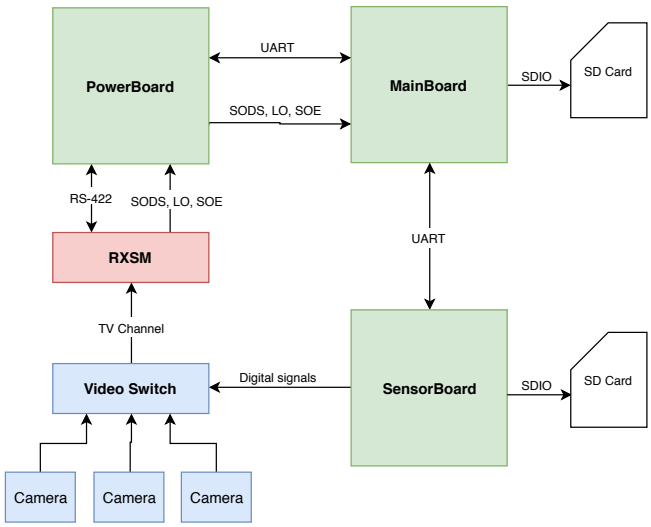

Figure 7: Data flow

Each data frame is encoded using Consistent Overhead Byte Stuffing algorithm and separated by

`0x00` byte. `0x00` bytes are used for data synchronization. The COBS algorithm replaces each zero data byte with a non-zero value so that no `0x00` data bytes will appear in the packet.

Several types of data frames were implemented: MainBoard, SensorBoard, Info and Telecommand data frames. The MainBoard, SensorBoard and Info data frames (fig. 8) consists of:

- `MSGID` byte
- `MSGCNT`
- `TIMESTAMP`
- `MSG`
- Two `CRC` bytes

Protocol Buffer[5] is used in order to serialize the data (green part of the frame). Nanopb is used on the side of the microcontroller, which is a small code-size Protocol Buffers implementation in ANSI C

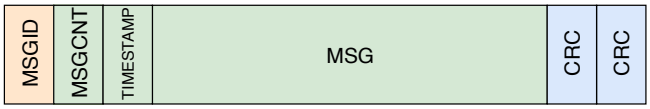

Figure 8: Data flow

Downlink data budget
Data rates of the synchronous downlink are presented in the below table. Total *pure data* rate is below 21kbit/s. Including start and stop bits overhead, rate is **26.2 kbit/s**, which is below allowed maximum (30kbit/s). It leaves enough space for asynchronous data handling (commands and commands' feedback frames), which has rates negligibly small compared to synchronous transmission.

Frame type	Frame size	Transmission frequency	Data rate
Mainboard frame	∼260 bits	60Hz	∼15.25 kbit/s
Sensorboard frame	∼420 bits	10Hz	∼4,10 kbit/s
Info frame	∼165 bits	10Hz	∼1,6 kbit/s
		Total	**∼20.95 kbit/s**

2.3 Sensors

List of the sensors with data acquisition rates and interface types are presented in the table below.

Mainboard		
Sensor	**Interface**	**Data rate [Hz]**
Load cell	ADC	5000
Differential pressure sensor	I^2C	250
Motor current sensor	I^2C	250
Encoder	Quadrature	-

Sensorboard		
Sensor	**Interface**	**Data rate [Hz]**
IMU (accelerometer + gyroscope)	I^2C	100
Absolute pressure sensor	I^2C	100
Temperature sensor	I^2C	2
Humidity sensor	I^2C	2

2.4 Motor control

DC motor was controlled with the use of the DRV8871 motor driver chip. Microcontroller generates two PWM signals at the frequency of 8400Hz at the inputs of the driver. Different motor velocities and directions of rotation can be achieved by changing of the signals' duty cycles.

Three different control feedback principles and loops were implemented:
Position control
Microcontroller's encoder input provides hardware support for the quadrature signal reception. Number of received signals is stored in one of the registers. Proper sampling of this value and calculations taking into account i.a. encoder resolution and gear ratio, allow to obtain the actual position and velocity linear guide trolley. This measurement is the feedback principle for the position control, which is implemented as a 1000Hz control loop with PID controller. Manipulation of the desired position with the reference to the actual position allows to move the gripper.

Force control
In this case load cell force measurement is used as a feedback principle. Gripper can be moved with the constant velocity in one of the directions until the desired pulling or compression force is measured on the force sensor.

Current control
In this case motor current measurement is used as a feedback principle. Gripper can be moved with the constant velocity in one of the directions until the desired current is measured on the motor.

The advantage of the system is that control loops can be switched dynamically – current control is used to home the gripper at the beginning of the experiment, control is then switched to the position control and the gripper is moved to the position 15mm away from the sample. From this position gripper is moved towards the sample until the compression force of i.e. 13N is measured. Gripper is then moved again to the position 15mm away from the sample and then again moves with the force feedback towards the sample. When the experiment time is exceeded gripper is once again homed with the current control feedback. All the positions, forces and currents are parameters, which can be easily adjusted in the initialization function in order to improve or change experiment routine (i.e. when the size of the sample or gripper will be changed).

2.5 Pressure control

Pressure control inside the membrane must be controlled within a certain tolerance range. Gripper's principle of operation requires that during the inflation phase pressure inside the membrane is kept above around 80% of the determined optimal pressure for the given membrane. On the other hand, pressure can't be too high as it would cause the membrane to tear apart. Initial idea for the pressure regulation was based on the low pressure regulator, which lowers the pressure from the bottle and delivers it to the inlet valve and therefore to the membrane. This regulator has an adjustable output, which could be set to the optimal value. Simple double setpoint controller was sufficient to maintain the desired pressure.
However during vibration tests, due to the vibrations, regulator output readjusted itself to the different value disturbing the experiment. Appropriate steps were taken to mechanically prevent next failure, but also the pressure control was redesigned to be doubly secured.

New pressure control is still based on the double setpoint controller. However instead of relying directly on readings from the pressure sensor, which were subjected to the inertia, and opening the valve until desired pressure was reached, new control algorithm opens valves only for few milliseconds, let the pressure readings to stabilize and only then decides if next correction is necessary. In this way, pressure is controlled by short pulses changing its value gradually – this phenomenon can be observed in the fig. 9 presenting reaching the pressure of 20kPa during one of the tests. This solution is generally slower

than previous one (because of the stabilization period), however it is reliable, allowing to achieve stable pressure even if the regulator readjusts itself.

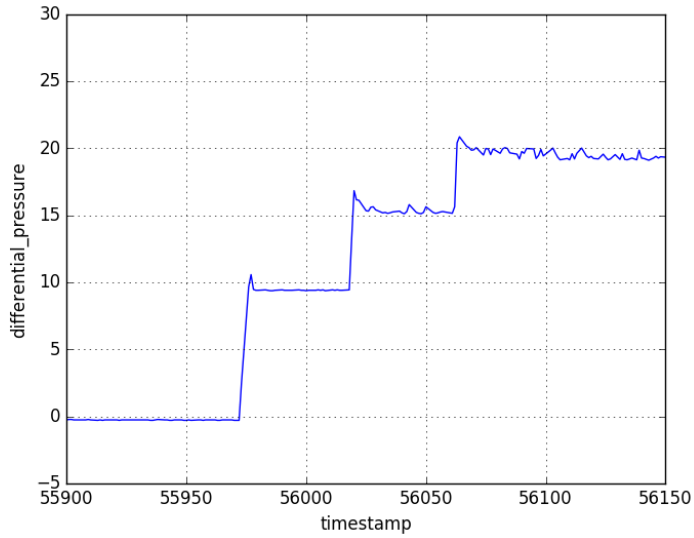

Figure 9: Example of the new pressure control algorithm

2.6 SD Cards

SD cards on both boards are using SDIO (Secure Digital Input Output) as a hardware interface. Instead of writing data binary directly to the cards, the FatFs[6] is used. FatFs is a lightweight FAT/exFAT filesystem for embedded systems written in compliance with ANSI C. It provides various functions allowing i.e. to mount volumes, create and delete directories and files or monitor the free space of the card. It also handles multiple open files and directories. This feature was used in the TRACZ software – after each power-up new directory is created allowing for storage of data from multiple sessions. Data is saved at a frequency of 100Hz in the form of UART frames as well as C structs containing more detailed information. Each type of frame/struct is stored in the different file. New files are created every 2 minutes to avoid data corruption. Cached data is flushed to the card every 5 seconds, so in case of power loss at most 5 last seconds of data are lost.

2.7 Other interfaces and features

A lot of smaller features had to be implemented in addition to the communication, data acquisition, motor and pressure control to handle the rest of the hardware and provide mechanisms necessary for the proper operation of experiment or later data analysis. They are presented below:

- Timestamp generation and synchronization between MainBoard and SensorBoard – timestamp is used to stamp frames for better data analysis. In order to obtain the same timestamps for both boards, MainBoard generates time base at the 1000Hz frequency (1ms resolution) and controls clock line for the SensorBoard timestamp.
- RXSM signals reading – SODS, LO and SOE signals are import for the flight mode algorithm, instead of relying on edge interrupts they are sampled at the frequency of 3000Hz (at least 5 identical consecutive samples are necessary to decide that the signal is active).
- Electromagnetic lock (Output) – used for immobilization of the gripper during the rapid ascent of the rocket, must be opened when the gripper moves from the initial position to avoid the blockage of the mechanism.
- Vision power (3x Output) – used to turn the cameras on/off (each one separately).

- TV Channel selection – (2x Output) switching between video transmission from the three cameras (only one can be transmitted at any given time), used i.a. to determine if the cameras were booted nominally
- Vision illumination (Output) – LED strip used as a lighting for the cameras.
- Vision system synchronization LED (PWM Output) – blinks blue LED every 5 seconds to allow the synchronization of the recording from the cameras in post-processing.

All the above features also had to be correctly integrated to operate both in test and flight (autonomous) mode.

3 Summary

Complexity of mechanical and pressure systems, extensive communication, data acquisition and reliability requirements formed the difficult software challenge. Not all the *Hope to do* requirements have been met, however the resulting system meets all the necessary requirements, was constantly improved on the basis of the conducted tests, successfully passed Rexus26 payload integration tests during Bench Tests at DLR Oberpfaffenhofen and seems ready for the upcoming flight in March 2019.

References

[1] REXUS User Manual. `rexusbexus.net/rexus/rexus-user-manual`.

[2] STM32 documentation for STM32F405RGT6 MCU and STM32F405 HAL Libraries. `www.st.com`.

[3] FreeRTOS V10.0.0 Reference Manual. `www.freertos.org/Documentation/RTOS_book.html`.

[4] Mastering the FreeRTOS Real Time Kernel – a Hands On Tutorial Guide. `www.freertos.org/Documentation/RTOS_book.html`.

[5] Protocol Buffers Reference. `developers.google.com/protocol-buffers/docs/reference/overview`.

[6] FatFS documentation. `elm-chan.org/fsw/ff/00index_e.html`.