Avionics for HyMOVE
Scalable Core Avionics Architecture for an Orbit Transfer Vehicle *

Sergio Montenegro[1], Emilio Miranda[1], Michael Strohmeier[1], Paola Breda[2], and Michael Vogel[2]

[1]Department of Aerospace Computer Science, Würzburg University
[2]HyImpulse Technologies GmbH

June 30, 2023

Contact: sergio.montenegro@uni-wuerzburg.de
Presentation will be performed by: Sergio Montenegro

1 Introduction

Within HyMOVE ("Hybrid Multipurpose Orbital Vehicle") , HyImpulse Technologies GmbH (HIT) and Julius-Maximilians-University Würzburg (JMUW) are developing an orbital vehicle, which can be adapted to a large scope of different missions, thanks to its modular architecture. This modularity will allow for a possible standardization of parts and thus to a reduction of cost by allowing a more complete adoption of procedures from general industrial practices like serial production into the space industry.

Next to this flexibility, one of the core technologies of the HyMOVE vehicle lies in the utilization of hybrid rocket propulsion, which is defined as the type of rocket propulsion that uses both solid and liquid components. This yields several advantages compared to both liquid and solid propulsion, key among which are safety and simplicity at high Isp-performance [1]. Unlike solid propulsion, hybrid rocket motors are controllable, so that more complex missions can be executed based on this type of engine. Hybrid propulsion has historically been plagued by low thrust densities, an issue which has been overcome due to the advent of liquefying fuels. Among these, paraffin has been the fuel of choice for HIT's propulsion developments, but the baseline choice of oxidizer used in HIT's other projects (liquid oxygen) is unfit for an in-space application due to its cryogenic nature. Instead, alternatives for oxidizers are currently under investigation, especially "green" oxidizers like hydrogen peroxide (H_2O_2) or nitrous oxide (N_2O), to complement the already non-toxic and soon to be carbon-neutral paraffin fuel.

JMUW develops the corresponding highly modular, scalable, and reliable avionics systems. HyMOVE is intended to have a modular design, so that the technology can be adapted for a wide variety of applications. For HyMOVE, a very high dependability is needed and if the system stays in orbit, then also very high reliability: Fault tolerant control (software and hardware), radiation resistant hardware and real-time critical tasks/control. The software includes the real-time kernel, software infrastructure, communications middleware from our building blocks execution platform RODOS, and key applications for satellite and kick-stage operations (HyMOVE combines both). Our fault tolerance mechanism is based on an "ultra-fast recovery" system. The goal is for each computer to take less than a second to reboot and start all applications, so that it can very quickly rejoin the network after a crash and take over control tasks. Important functions are replicated in the network several times in different nodes so that a crash does not jeopardize the flight. The crashed computer will then be ready to take over tasks again in a second. We are confident to implement our "ultra-fast recovery" concept as described, yet this is a novelty.

*Thanks to the financial support from Bayerischen Staatsministeriums für Wirtschaft

2 State of the Art

Today, conventional launch vehicle avionics, while very reliable, are also very heavy and expensive when compared to small and micro launcher requirements. In this context, a crucial issue in the design of a small launcher is the development of a lightweight avionics system at an affordable cost. According to Di Sotto et al. the following criteria are important [2]:

1. Scalability and the associated complexity
2. Modularity and integrability
3. Cost, especially reduction by using COTS
4. Reliability and the associated safety

In modern carrier systems, high reliability at sub component to subsystem level is implemented by means of redundancy. A reliable communication system forms the "backbone" of all Distributed Integrated Modular Avionic (DIMA) systems [3]. The communication system used must, above all, enable simple integration of various components and guarantee reliable data exchange in real time. In addition, a high data rate, usually exceeding 1 Mbit/s, is required in particular for communication between hot redundant subsystems.

In traditional launch systems and space vehicles (e.g. Ariane5, SpaceShuttle, ISS), a field bus based on the MIL-STD-1553b network standard introduced by the US Air Force in 1973 is used as the central communication system [4, 5]. Due to the increasing number of small satellites, especially for commercial tasks, the demand for micro and mini launchers is also growing. In this context, the development costs incurred can be significantly reduced by using COTS. Modern launchers (e.g. Ariane 6, VEGA) already use commercial network solutions based on IEEE 802 network standards, but with deterministic network behavior due to data prioritization [6, 7, 8]. In particular, Time-Triggered Ethernet (TTEthernet) [9], which is standardized by SAE AS6802 and meets stringent transmission requirements by using different network mechanisms simultaneously (time-triggered, rate-constraint, and best-effort traffic).

3 HyMOVE Avionics

3.1 Requirements

When designing according to the principles of safety engineering, several key considerations come into play. Firstly, a modular distributed system and modular software approach should be adopted. This involves breaking down the system and software into building blocks that can be independently developed, tested, and maintained. By doing so, it becomes easier to identify and isolate potential issues, leading to improved safety.

Additionally, incorporating spare units is crucial for ensuring system reliability. Hot redundancy can be implemented for launchers, meaning that backup units are constantly active and ready to take over in the event of a failure. For satellites, warm and cold redundancy can be employed, where spare units are kept in a state of readiness or powered off, respectively, for deployment when needed.

To address transient failures, a high-performance computer can be utilized, taking advantage of time redundancy. This involves executing vital applications three times on each node and comparing the results. In the event of a failure, control can be transferred to a spare unit, triggering a reboot and recovery process. This approach enhances fault detection and provides an effective means of masking and overcoming transient failures.

By incorporating these principles into the design process, safety engineering can be effectively applied, improving the overall reliability and robustness of the system.

3.2 Hardware

All Hardware is implemented as dual-X (X: controller, router, connections, switches, etc.) for fault tolerance reasons. Everything is duplicated. Each basic hardware unit is a dual-X as shown in Figure 1. We have two parallel running Time Triggered Ethernet networks. Every controller, e.g. from

the dual-board computer or dual-payload computer or dual-GNC Computer has two interfaces to the network, one for each router. Hence, each Dual-X has in total four links to the dual-network.

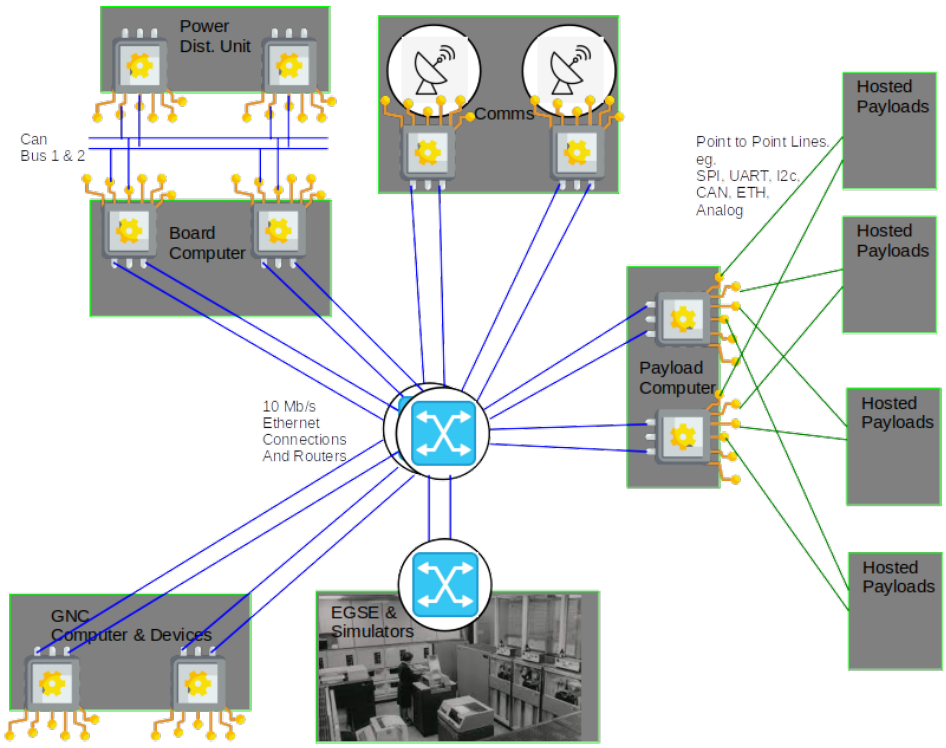

Figure 1: HyMOVE Avionic Hardware

Figure 2 shows the implementation of each dual computer (eg. dual board computer, dual payload computer and dual GNC computer). There are two aspects which shall be explained in more detail, the rest shall be clear:

1. IO Adapter

   There is only one dual computer implementation and this shall be used for different functions and for different payloads in different missions. The dual computer shall not be implemented for different IO Configurations. Therefore we route all IO lines to a big IO connector. For different IO Configuration we have just to re-implement the IO adapter which shall be a simple implementation.

2. Keep-off

   With the Keep-Off circuit we implement in a very simple and flexible way the redundancy management. The redundancy management is totally controlled by software, so we can switch very easily from cold to hot redundancy. As long as one computer (the worker) keeps sending a pulse to the keep off circuit of the other computer (spare), the spare computer will remain off. If the pulse does not arrive on time, the spare computer will be turned on and after a self check it will take control (become the worker) and send keep off pulses to the other computer which will be then turned off and become the spare computer. Each node has an application called Watchdog. The watchdog app checks timings and the correct behavior of all other applications and devices on the computer. If every thing seems to be working properly, it will send one pulse to the keep off circuit of the other computer. If the computer crashes of the watchdog-app considers some thing is wrong, then the keep off pulse will not come and the other computer will take over.

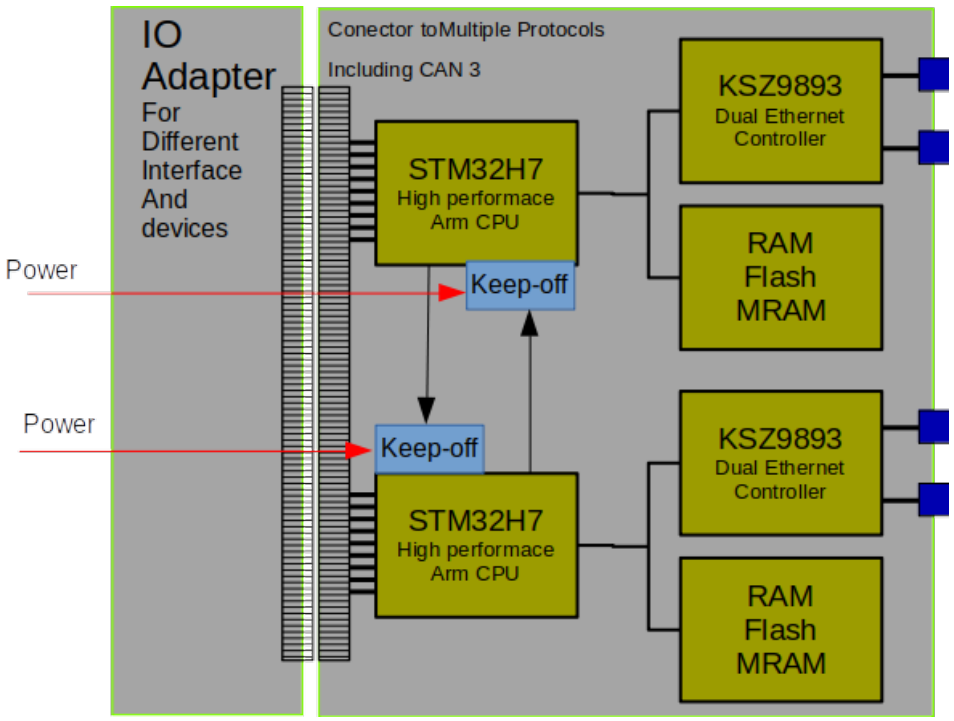

Figure 2: HyMOVE Dual Computer

3.2.1 Power Bus Distribution

Figure 3 shows the implementation of the power bus. The power lines are each controlled by exactly one switch and each switch is controlled separately. The power lines have a defined voltage and therefore, we must generate another voltage internally on any board if required. The power buses are completely independent and supply all the hardware. The dual-X components are connected to a different power bus, this way, even if a power bus fails, all the redundant system can continue with nominal operations.

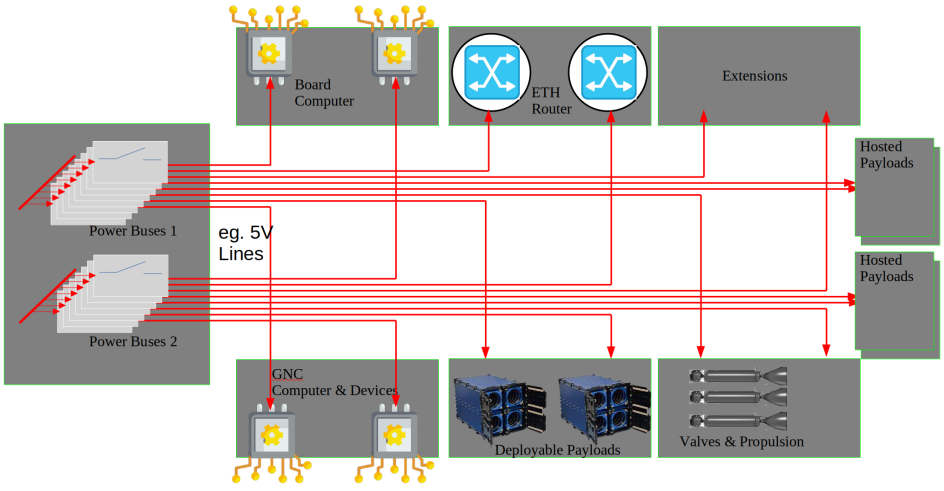

Figure 3: HyMOVE Power Bus Distribution

3.2.2 Power Distribution Unit

Figure 4 shows the power distribution unit diagram. There are different power lines and devices, and every device has its own power line. Each of this group of switches is controlled by a separate

microcontroller or FPGA. This group of switches is controller by a killswitch or deployment switch, independent from each other, with two completely independent power lines. It is important to mention, that if one of these microcontroller or FPGA fails, we could have a critical situation where one power line stays always on or always off. This could potentially drain the batteries. Therefore, the control of the switch is not only on and off, but we have to give a pulse to keep this line on or off. The circuit in Fig 4 shows a simple solution to implement this. We generate a pulse and this pulse allows a capacitor to charge. Once the capacitor is charged, the main bus switch changes from its default state to the active state. This switch can either be a keep on of a keep off switch. As long as the signal is present, the switch is on in the case of a keep on, or off in the case of a keep off signal. When the pulse is no longer present, the capacitor discharges, and the switch returns to its default state. For example, the transmitter should be off when the microcontroller goes off, then this component should also turn off. On the other hand, the board computer should be on by default when the control turns off. Then, the default of this line should be on, and this way the board computer will continue to live.

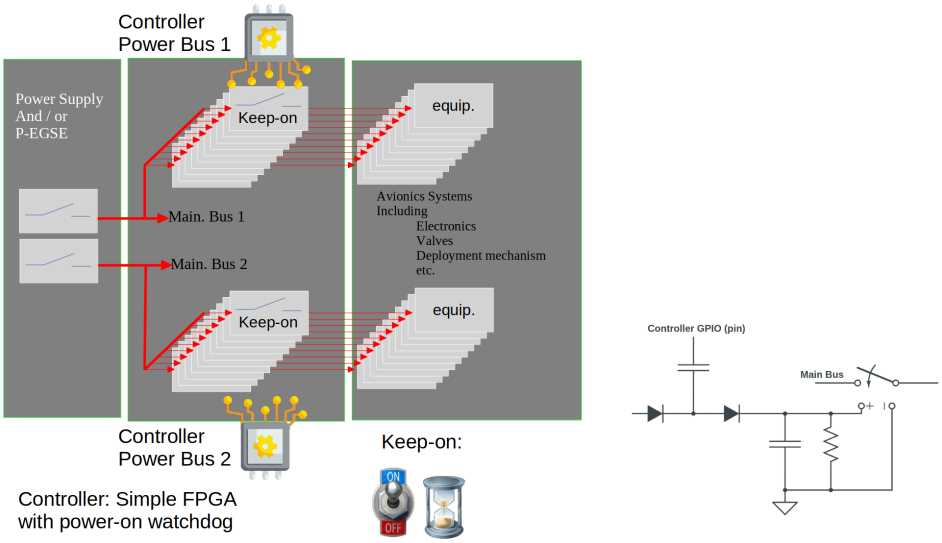

Figure 4: HyMOVE Power Distribution Unit

3.3 Software

We implement the control software "by construction based on building blocks (BB)" on the top of the BB-execution Platform RODOS. RODOS is a dependable distributed real time operating system. RODOS provides a real time micro kernel and a distributed Publisher-Subscriber Middleware. Applications distribute messages by publishing on topics. The middleware will distribute the message to all subscriber in the system, independently of its position. For the publisher it is transparent if the communication partner is running in the same computer or on other computer attached to the network. We can even build distributed systems which one part is running in the spacecraft and the other part on the ground segment. Even this physical barriers will be transparent for the publisher-subscriber communication. So we can distribute applications dynamically among computer. This simplifies the fault tolerance and redundancy management. Applications are then the Building Blocks (like chips on hardware) and they will be interconnected by the middleware (like on a printed circuit board) as depicted in Figure 5. To construct complex software, we "just" plug different BBs together and do not care about its internal structure.

3.3.1 HyMOVE Applications

1. Watchdog

   Every node has a watchdog application. The main task is to send periodically a signal to the hardware watchdog to avoid resetting the software when everything is working as expected. The application receives the iamAlive topic messages from the other applications. Every thread in

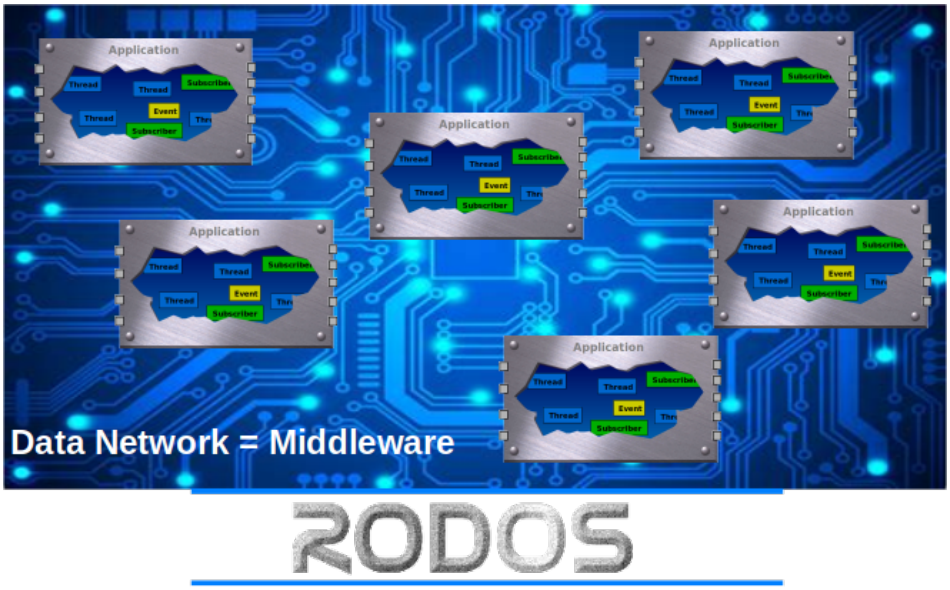

Figure 5: Building Blocks on RODOS BB-Execution Platform

every application is able to send an iamAlive message to the watchdog application. The watchdog stores the application id and its timeout of the imAlive messages and periodically checks if these timeouts have expired. Whenever a timeout has expired, it means that an application was not able to update its timeout and therefore that a problem occurred. The watchdog generates immediately an anomaly report and stops sending the reset signal to the hardware watchdog, which in turn will reset the node. The watchdog is also able to process commands such as to enable or disable its ability to self reset the hardware. Disabling this allows the node to continue working with faulty threads, but still generating anomaly reports.

2. Commander

   The commander application receives messages from the uplink manager and checks whether a command is an immediate command or a timed command and sends them through the respective topic.

3. Timed Commander

   The timed commander application stores the received commands with their execution time in UTC format. It checks periodically if the execution time corresponds to the actual UTC time to send the command through the immediate command topic.

4. Housekeeper

   The housekeeper application is subscribed to the standard and extended telemetry and stores these information in a list. The application asks the other applications periodically for their standard telemetry through the standard telemetry topic. The standard telemetry can be sent to the downlink manager immediately or once the spacecraft is in contact with the ground station. The ground station can command the housekeeper to also collect the extended telemetry, which will be collected the next time the housekeeper collects the standard telemetry.

5. Downlink Manager

   The downlink manager application is the bridge between the transmitter hardware and the avionics software. It checks periodically if the standard and extended telemetry queues are not empty. It can also process commands such as to pause the data transmission and clear the telemetry queues for example.

6. Uplink Manager

   The uplink manager application is the bridge between the receiver hardware and the avionics software. It checks the integrity of the received messages and them to the commander application.

7. Power Control

   The power control application is in charge of turning on and off the hardware. It does this either via ground station commands or after an anomaly has been generated.

8. Anomaly Manager

   Every application has the ability to check their performance. Whenever a verification fails, the application can publish an anomaly report and the anomaly manager will receive it and store the data. Some anomalies are not critical, they are simply registered and are sent through the telemetry in order to have a historic. However, with critical anomalies, an action is executed immediately.

9. Time Control

   There are several nodes and the ground station and each of them should have the same time. The time control application synchronizes the clocks between the different nodes and also with the ground station. The application tries to make the time difference between nodes in the order of microseconds and in the order of milliseconds with the ground station.

10. Payloads Deployer

    In the case of HyMOVE, there are two types of payloads. The ones that will be released and the ones that stay in the spacecraft. For the deployed, the application contains a list, which states at which time the payload should be released. These times can be modified via ground commands.

11. Hosted Payload Interfaces

    The application controls the different tasks that should be performed specifically for the payloads that stay in the spacecraft.

12. Mass Memory Manager

    The payloads could generate great amounts of information. The mass memory manager stores these data in Flash memory in the shape of a file system.

3.3.2 HyMOVE Generic Topics

1. anomalyReport

   Whenever any application has a failure or is not working as expected, its threads can publish an anomaly message to this topic

2. iamAlive

   Every thread can publish to this topic to allow the local watchdog timer to monitor its performance. The thread must publish a message periodically to let know the watchdog that it has a normal operation.

3. Standard Hk Request

   The housekeeper uses this topic to request all the applications for their most important telemetry. Every application must be subscribed to this topic.

4. Standard Telemetry

   The housekeeper uses this topic to collect the most important telemetry of every application. Every application has a set of its most important variables and flags.

5. External Telemetry

   Every application sends through this topic the telemetry with all the additional details from its parameters, data that is not as important as the standard telemetry, but rather complementary.

6. teleCommands

   Every application is subscribed to this topic. The commander application publishes to this topic to the respective application to execute a ground control operation.

Figure 6 and 7 shows a typical BB (applications) and topics configuration for a spacecraft.