IAC–21–B4,IP,20,x66551

TECHNOLOGIES AND FACILITIES FOR THE AIV/AIT PHASE OF NANOSATELLITES ADCS SUBSYSTEMS: HERMES AS AN APPLICATIVE CASE

Mr Lorenzo Colaninno
Politecnico di Milano, Italy, lorenzo.colaninno@mail.polimi.it

Dr. Andrea Colagrossi
Politecnico di Milano, Italy, andrea.colagrossi@polimi.it

Dr. Paolo Lunghi
Politecnico di Milano, Italy, paolo.lunghi@polimi.it

Prof. Michèle Lavagna
Politecnico di Milano, Italy, michelle.lavagna@polimi.it

Today's space sector is experiencing an increasing demand of small satellite platforms, ranging in the class of nanosatellites. These space systems, despite their dimensions, are now technological mature and they are entering the market of complex space missions with stringent system requirements. In fact, their system peculiarities allow to easily implement distributed space architectures to boost the mission return. Among all the subsystems of these advanced nanosatellites, the Attitude Determination and Control Subsystem (ADCS) is particularly stressed with complex pointing and maneuvering profiles. The algorithms and the software are continuously required with new and computationally relevant on-board functions, the hardware is loaded with relevant duty cycles and frequent demanding actuations. In these regards, the AIV/AIT phase is becoming more and more crucial also for small spacecraft. However, the required time-to-space and the cost caps of such missions prevent to apply all the standards and to access the facilities of large space systems. Thus, there is an emerging request of technologies and facilities to support the AIV/AIT phase of the ADCS for this class of spacecraft. The paper presents the technologies and the facilities that are being implemented at Politecnico di Milano, Department of Aerospace Science and Technology, by ASTRA research group. These have been designed and realized to support the AIV/AIT phase of the ADCS of HERMES (High Energy Rapid Modular Ensemble of Satellites) project. HERMES is an astrophysics mission for high-energy rapid transient localization and detection composed of 3U nanosatellites, distributed in a constellation, to perform sky monitoring and localization through triangularization. The paper describes the AIV/AIT procedures for the ADCS subsystem of the platform. The facilities where these are carried out are reported, highlighting the requirements that have been applied for their implementation. The technological architecture to conduct Model, Software, Processor, Hardware-In-The-Loop (MIL, SIL, PIL, HIL) testing activities is described, discussing the verification steps that are possible with the available laboratory facilities. The PIL facility is described in details, highlighting the rationale that led to the selection of the proposed architecture among a set of available solutions. The discussion also reports the main outcomes of the software qualification plan, discussing all the key points that prove the possibility for the HERMES system to fulfil its mission requirements in any operational model.
Keywords: Testing and Verification, AIV/AIT, ADCS, PIL/HIL, Test-Bench, Cubesats

1. Introduction

During the last twenty years the cubesat technology had a significant impact on the space industry, making access to space easier and cheaper. The resulting increase in the demand of small satellite platforms, particularly in the nanosatellite range, boosted the development of the related technologies and their miniaturization.

However, despite the continuous improvements in the technology and the gained experience from past missions, cubesats reliability still represents a major issue. According to the Swartwout database of 112 cubesats launched between 2003 and 2012 [11], almost half of all the failures arises from issues in the functional integration process and could have been avoided if a proper system-level functional test campaign had been performed. More recent studies [9, 12] report the same trend, with a high infant mortality (defined as a failure during the spacecraft commissioning or early stages), in particular for the university-developed cubesats, which often tend to cut out or reduce the test campaign time and efforts, due to time or budget constraints. Moreover, the use of Commercial Off-The-Shelf (COTS) components introduces the risk of additional failures in the space missions, if the importance of a proper system-level test campaign is underestimated. These considerations highlight the need of a proper Assembly, Integration and Verification/Test (AIV/AIT) campaign for cubesat missions. A complete cubesat verification and testing campaign shall include thermal, mechanical and environmental tests, but also functional, interface and performance tests. In particular, the following facilities shall be developed and implemented:

- calibration and testing facilities for sensors and actuators functional, performance and interface tests;
- SIL/PIL testing facility for software verification;
- complete HIL test facility with Sun simulator, Helmoltz cage, multi-DOF platforms.

Moreover, the PIL and HIL facilities shall employ real-time machines or Field Programmable Gate Arrays (FPGA) to guarantee the reliability of the results. However, cubesat missions often lack the necessary time, budget and resources to implement the above mentioned AIV/AIT facilities and technologies. For this reason, there is a double need in the small spacecraft industry: on one side the definition of cubesat dedicated AIV/AIT standards, on the other the availability of test-benches characterised by low-cost, easiness of design and quick implementation. The challenging task is the identification of a test-bench compliant with the low-budget characteristics and, at the same time, capable to guarantee the required performances and results reliability.

In this context, the technologies and facilities that are being implemented to support the AIV/AIT phase of the Attitude Determination and Control System (ADCS) of HERMES are presented. The HERMES mission and the ADCS design and characteristics are summarised in Section 2, while the AIV/AIT overview is reported in Section 3 and Section 4. Section 5 describes the requirements of the PIL/HIL test facility for ADCS software verification. The trade-off process that led to the selection of the PIL/HIL test-bench solutions is reported in Section 6, while Section 7 describes the implementation of the facility.

2. HERMES Mission and ADCS Overview

HERMES is an astrophysics mission to perform sky monitoring for high-energy rapid transient detection and localization through triangularization. The space segment is composed of 3U nanosatellites, distributed in a six elements constellation. The HERMES mission is realized by a consortium of universities, research institutions and enterprises, the main contributors being the Italian Ministry of University and Research (MUR), the Italian Space Agency (ASI) and the European Commission. The platform design and AIV/AIT are in charge of the Politecnico di Milano, whereas the National Institute of Astrophysics (INAF) and the Cagliari University lead the science operations and the payload design and AIV/AIT. ASI acts as the project coordinator and the launch service provider.

The HERMES ADCS is developed in Politecnico di Milano by the ASTRA research group and it is intended to provide full control authority on the attitude states (i.e. 3 DOF) of the spacecraft, and to ensure complete attitude and position states (i.e. 6 DOF) determination both in sunlight and in eclipse. The pointing requirements, dictated by the mission objectives, led to the definition of the ADCS performances reported in Tab. 1.

Table 1: ADCS performance requirements

Performance	Value
APE	$5°$
ARE	$0.1°/s$
AKE	$3°$
Positioning	300 m
Jitter	$< 1°$
Agility	$> 100°$ in 600 s
Gain margin	6 dB
Phase margin	$30°$

The HERMES ADCS software is composed by four main units:

- Attitude Determination Subsystem (ADS)
- Orbital Determination Subsystem (ODS)
- Attitude Control Subsystem (ACS)
- ADCS Mode Manager (ADCS-MM) and Finite State Machine (ADCS-FSM)

The HERMES ADCS is composed by the following actuators:

- 4 Reaction Wheels (RW);

- 3 magnetorquers (MAGTRQ).

The reaction wheels provide the main attitude control torque, while the magnetorquers are required for redundancy, reaction wheels desaturation, and detumbling maneuvers.

The HERMES ADCS is composed by the following sensors:

- 4 Fine Sun Sensors (FSS);
- 12 Coarse Sun Sensors (CSS);
- 3 magnetometers (MAGMTR);
- 2 gyroscopes (3-axis);
- 1 GPS sensor;
- 1 accelerometer.

2.1 *Hardware Components and Characteristics*

The ADCS hardware components are provided by GomSpace A/S, except for the Inertial Measurement Unit (IMU), which is provided by MemSense LLC. The components are here briefly presented, highlighting their data interfaces, which are fundamental for the implementation of the AIV/AIT Ground Support Equipment (GSE). A connection scheme of the ADCS hardware architecture is shown in Fig. 1.

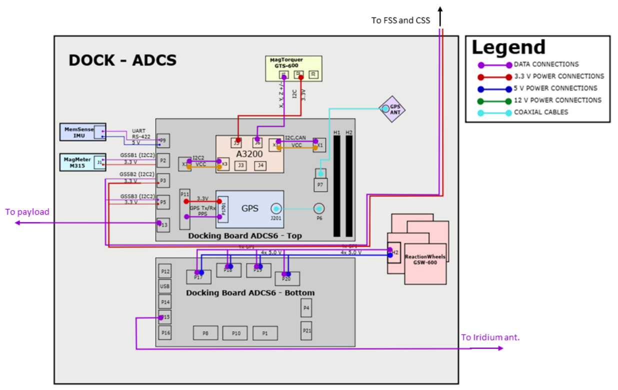

Fig. 1: HERMES ADCS connection scheme.

ADCS MOTHERBOARD: docking board compatible with ADCS hardware provided by GomSpace. Used to mount the OBC for the ADCS and to interface it with sensors and actuators. The magnetorquers are the only ADCS hardware component which are not connected to this motherboard, since they are directly interfaced with the OBC through an H-bridge. The motherboard features the following interfaces to connect the sensors and actuators: Inter Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), Controller Area Network (CAN) and RS422 Universal Asynchronous Receiver-Transmitter (UART).

OBC-ADCS: dedicated OBC for ADCS installed on the ADCS motherboard. It includes a 3-Axis gyroscope and a 3-Axis magneto resistive sensor which interface with the micro-controller unit of the OBC-ADCS via I^2C bus.

GPS: includes a GPS module and a GPS antenna. The GPS module communicates with the motherboard through UART RS422, in binary or ASCII.

MAGNETOMETER: magnetometer based on a geomagnetic sensor. The interface with the motherboard is through a I^2C bus.

FSS: 4 fine Sun sensors connected to the motherboard with a I^2C interface.

CSS: 12 coarse cosine type Sun sensors integrated in the solar panels, communicating with the motherboard through I^2C.

IMU: includes a gyroscope, an accelerometer and a magnetometer. The communication with the motherboard is driven by a UART RS422 interface exchanging binary data.

REACTION WHEELS: main actuation components for attitude control. The HERMES spacecrafts use an already mounted reaction wheels assembly in a 4-wheel pyramid setup. A SPI bus is used for data exchange between the RW and the motherboard.

MAGNETORQUERS: three axis magnetorquer designed for 3U and 6U cubesats. The magnetorquer control torque signal is transmitted by a Pulse Width Modulation (PWM) output of the microcontroller unit of the ADCS-OBC, through 3 H-bridge drivers.

3. AIV/AIT Plan

The AIV/AIT activities of the HERMES-TP spacecrafts are mainly carried out at Politecnico di Milano, following the specifications defined by the *European Cooperation for Space Standardisation* [3, 4]. Three main phases can be recognised in the AIV/AIT plan, and each phase includes tests which are different by level, models and facilities.

3.1 *AIV/AIT Phases*

The AIV/AIT activities can be divided in three high-level phases, which include the following sublevels of tests: Software Tests (SWT), Equipment Tests (ET), Subsystem Integration Tests (SSIT), System Integration Tests (SIT), Payload Integration

Tests (PLT) and Integrated System Tests (IST). The SWT cover all the three phases of the AIV/AIT plan, increasing in complexity, from unit tests of single software components (first phase), scaling up to end to end simulations of the whole system (last phase).

- ◦ First phase includes ET and SSIT. In this phase each subsystem is characterized, integrated, and tested in a flat-sat configuration.
- ◦ Second phase is the complete integration of the whole S/C and includes part of the SSIT, the SIT, and the PLT.
- ◦ Third phase includes the IST, i.e. all the test campaign performed on the fully integrated spacecraft, including environmental tests, end to end mission simulation, and characterization of overall mass and inertia properties.

3.2 *AIV/AIT Models*

The HERMES testing activities employ different models of the spacecraft software components, subsystems, Service Module (SM) and assembled spacecraft:

- ◦ Virtual Models (VM) support the development of the On-Board Data Handling (OBDH) software and ADCS software. The behaviour of physical components is simulated and unit tests of separated software components, Model-In-the-Loop (MIL) and Software-In-the-Loop (SIL) tests are included in this category;
- ◦ Engineering Models (EM) are required for the OBDH. PIL and HIL verification and tests are going to be performed on the EM OBDH board.
- ◦ Structural-Thermal Model (STM) is required for the structure of the CubeSat.
- ◦ Prot-Flight Models (PFM) of the subsystems are functionally tested and their performances are verified, prior to integration in the complete PFM of the HERMES spacecraft. The PFM of the whole Service Module (SM) and of the fully integrated spacecraft will be subjected to environmental tests at qualification level.
- ◦ Flight Models (FM) of the whole SM and of the fully integrated spacecraft, including the subsystems FM, will undergo the acceptance tests. In case of successful acceptance test campaign, the FM will be ready for launch.

3.3 *AIV/AIT Facilities*

The HERMES test facilities consist in:

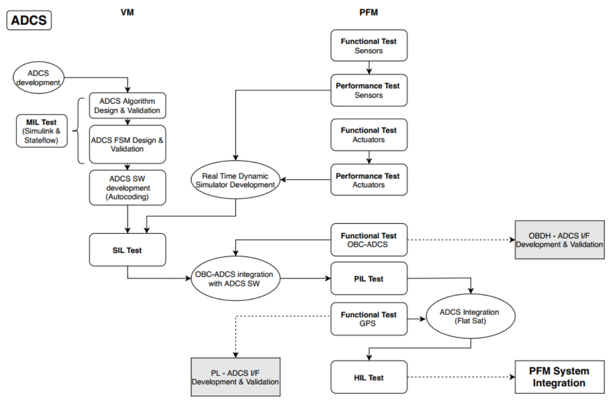
ADCS
VM
PFM
ADCS development
ADCS Algorithm Design & Validation
MIL Test (Simulink & Stateflow)
ADCS FSM Design & Validation
ADCS SW development (Autocoding)
Functional Test Sensors
Performance Test Sensors
Functional Test Actuators
Performance Test Actuators
Real Time Dynamic Simulator Development
SIL Test
Functional Test OBC-ADCS
OBDH - ADCS I/F Development & Validation
OBC-ADCS integration with ADCS SW
PIL Test
Functional Test GPS
ADCS Integration (Flat Sat)
PL - ADCS I/F Development & Validation
HIL Test
PFM System Integration

Fig. 2: ADCS AIV/AIT scheme

- ◦ clean room, ISO 8 with venting and filtration system used to integrate and assemble the satellites in a controlled environment;
- ◦ thermal vacuum chamber;
- ◦ vibration test facility;
- ◦ anechoic chamber;
- ◦ ADCS calibration and testing facility for the HERMES sensors and actuators;
- ◦ Electrical Ground Support Equipment (EGSE) facility, an hardware-software facility that will be placed inside the clean room. It comprises all the hardware and software tools needed to perform electrical and software tests on the satellite.

4. ADCS AIV/AIT Plan

The ADCS AIV/AIT plan follows the general plan, passing through the three high-level phases described in Section 3. The VM, PFM and FM are used to validate the ADCS, exploiting the clean room, the calibration and testing facility and the EGSE facility. A scheme of the ADCS AIV/AIT plan is shown in Fig. 2. As reported in the scheme, two processes proceed in parallel: the components testing and the software testing.

The COTS components are procured at a Technology Readiness Level (TRL) of 9, and they shall undergo functional, interface, performance, and mechanical/environmental tests at ET level, to check the compliance with what is declared in the datasheets. If each component passes the test campaign, it is accepted and the procedure continues with tests at subsystem levels.

The ADCS software is internally developed and it needs to be intensively tested. The complete testing and qualification campaign of the ADCS software represents the core of the HERMES AIV/AIT plan. All four units of the ADCS software shall be tested with MIL, SIL, PIL and HIL tests. The early development and testing of the subsystem is carried out on Virtual Models: the software components are checked for their correctness during the design process and, once fully developed, they are verified with SIL tests. If the SIL tests are successful, the ADCS software is integrated with the ADCS-OBC and the PIL and HIL test campaign begins at subsystem level.

At the end of the Performance tests, all the subsystems undergo a phase of qualification tests. Qualification is required for the in-house developed software. If the PFM passes the qualification tests, it is ready for the acceptance at system level.

5. ADCS PIL/HIL Facility: Design Requirements

After the VM tests on the ADCS software and its integration with the OBC-ADCS, the PIL and HIL tests are planned at subsystem integration level, in the first phase of the ADCS AIV/AIT plan presented in the previous Section. The main goal of the PIL/HIL facility is the verification of the ADCS software functionality and performances, which must be compliant with the HERMES requirements.

The basic idea is to provide simulated environment, dynamics and sensors/actuators data to the ADCS-OBC of the cubesat in real time. With a PIL/HIL facility, the designed software can be run on the target hardware in different modes, testing in depth all its functionalities while monitoring and checking the hardware-software compatibility.

The facility is composed by three main blocks:

- ADCS-OBC, with a complete deployment of the ADCS software, which is the Device Under Test;
- simulation computer running the Dynamical and environmental simulator, which include the VM of the sensors, if needed;
- interface modules, which allows the data exchange between the simulator and the ADCS-OBC.

In the PIL/HIL tests, the environment, the spacecraft dynamics, and possibly one or more sensors and actuators shall be simulated, exchanging mock data with the ADCS-OBC to feed the attitude determination and control algorithms in real time. The interface between the simulator and the ADCS-OBC needs the development of a suitable GSE, which has to be compliant with the required performances of the whole ADCS. In this regard, one of the fundamental points in the development of the test-bench is the update frequency of the ADCS system; the related HERMES requirements are here reported:

- ADCS shall control the 3 DOF rotation state with a control frequency $\geq$1 Hz.
- ADCS shall determine the 3 Rotational DOF state and the 3 Translational DOF state with a determination frequency $\geq$1 Hz.
- ADCS sensors shall be sampled at a frequency $\geq$1 Hz.
- ADCS actuators shall be commanded at a frequency $\geq$1 Hz.

The test reliability can be guaranteed if the PIL/HIL facility is compliant with the HERMES ADCS performances and functionalities. Therefore, the basic requirements for the facility can be formuated as follows:

- update rate from 1 Hz to 10 Hz;
- sensors sampling at a frequency higher than 10 Hz;
- soft real-time performances;
- data exchange with low corruption and losses;
- UART, SPI, I^2C interface, with slave behaviour for I^2C and SPI.

In addition to the performance and functional constraints, the PIL/HIL facility must be tailored for small spacecrafts. The development, procurement and implementation of the test-bench shall be possible with low budget and in short time, exploiting technologies which do not require advanced specific knowledge with respect to the medium system engineering skills. The cubesat oriented approach is a key element in the test-bench development, making the selected architecture a trade-off between the functional and performance requirements, and the implementation time, budget and efforts

6. ADCS PIL/HIL Facility: Design Trade-Off

The considered alternative solutions are presented in this Section, justifying the selected test-bench. A criteria matrix is reported in Tab. 3 (with cost parameter on Tab. 2), to visualize the trade-off result.

Three main solutions have been identified:

- single-board microcontrollers or single-board computers as interface boards, with non real-time simulator computer;
- real-time machines;
- Field Programmable Gate Arrays for sensors/actuators simulation.

Table 2: Cost values used in the criteria matrix

Cost [€]	< 120	120 to 200	200 to 400	400 to 4000	> 4000
Value	5	4	3	2	1

6.1 *Single-Board Microcontroller/Computer with non Real-Time Simulators*

A first possible solution exploits hardware interface boards between the Simulation computer, running the dynamical and environmental simulator, and the ADCS-OBC. The interface boards can either be single-board microcontrollers, such as the Arduino ATmega328P board family, or single-board computers, such as Raspberry Pi boards. In this architecture, a standard non-real time computer runs the dynamical and environmental simulation in soft real-time mode, exploiting the Simulink Desktop Real Time (SDRT) framework, described in details in Subsection 7.1. The performances and functionalities of this set-up are compliant with the facility requirements, both for the interface boards and the simulation computer:

- Arduino boards with ATmega328P microcontroller feature a 16 MHz clock frequency and a 32 kb flash memory. The required interfaces can be implemented with the available I/O peripherals on the Arduino Uno board: USB-Serial, 14 digital pins and 6 analog pins with available PWM;
- Raspberry Pi single-board computers feature a 1.5 GHz clock speed CPU, with 8 Gb LPDDR4 SDRAM, 4 USB ports and 40 GPIO pins which can be used to reproduce all the hardware interfaces specified in the facility requirements. The Raspberry Pi Operating System runs on the board.
- simulation computer with SDRT framework can reach performances up to 20 kHz, with the possibility to monitor if the real-time performances have been lost and at which point in the simulation.

The number of boards used, and the function they fulfill in the test-bench, introduce additional sub-solutions. A possible architecture implements the sensors/actuators models on the simulation computer, while the interface boards are only used to exchange the sensors/actuators mock data between the ADCS-OBC and the simulation computer. Another approach consists in the implementation of the sensors/actuators models directly on the interface boards, which receive the simulation data from the dynamical and environmental simulator, generate the mock data and forward them to the ADCS-OBC. The latter solution is possible, provided that the boards feature enough memory and performance resources.

On the number of boards needed, the simplest architecture would use a total of 15 boards, one for each sensor and actuator. However, a solution with high number of interface boards presents several disadvantages: it is difficult to manage, maintain and update and it needs a high number of serial ports on the simulator computer side, which is not a common feature for standard computers. The number of required boards can be decreased to 5 if the interface boards are configured to act as multi-slave devices in the SPI and I^2C buses. In this solution, one board for each sensors group is needed, i.e. GPS, IMU, sensors using I^2C bus (CCS, FSS, magnetometer, OBC magnetometer, OBC gyroscope), sensors using SPI bus (4 reaction wheels) and magnetorquers analog communication. A comparison between the architectures

Table 3: Criteria Matrix for PIL/HIL test bench.

			M&S compatibility	Technical skills	RT Performances	I2C/SPI slave	Efficiency	Developing time	Cost	Score
Weight			3	3	4	5	3	4	5	
Arduino	5 boards	Run sensor models	2.5	4	3	4.5	4	3.5	5	105
		Receive sensor data	3.5	4	3.5	4.5	5	4	5	115
	15 boards	Run sensors models	2.5	4.5	3.5	4.5	3.5	5	3	103
		Receive sensors data	3.5	4.5	4	4.5	3	5	3	106.5
Raspberry	5 boards	Run sensor models	2.5	3.5	3	3.5	4	4	4	95.4
		Receive sensor data	4	3.5	3	3.5	3	4	4	97
	15 boards	Run sensors models	2.5	4	3	3.5	2.5	5	2	86.5
		Receive sensors data	4	4	3	3.5	2	5	2	89.5
Speedgoat			5	4.5	5	5	2.5	4	1	102
Typhoon			4	3.5	5	5	2.5	3.5	1	94
FPGA			3	1	5	5	2.5	2	2	82.5

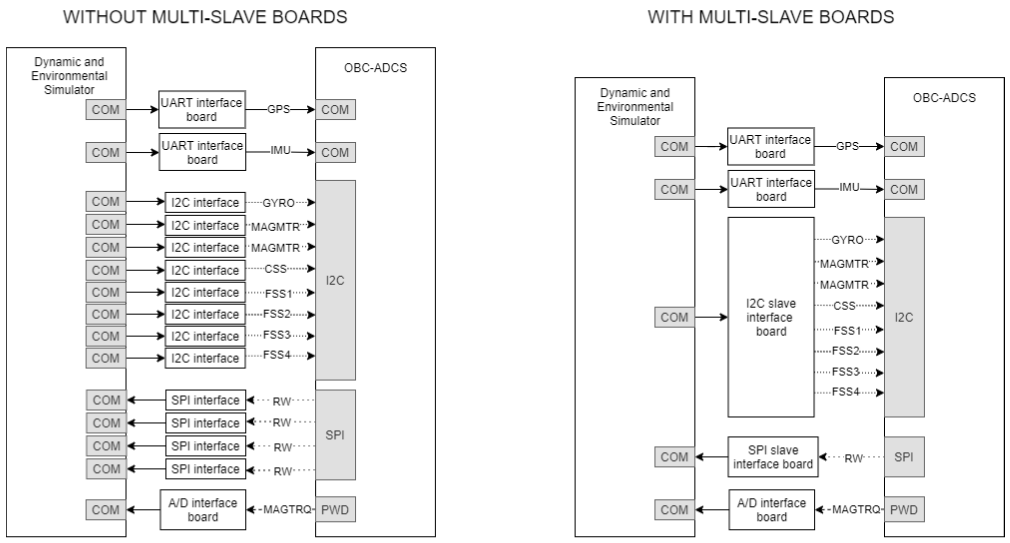

Fig. 3: Comparison between the PIL/HIL testing facility with and without multi-slave boards for the I^{2}C and SPI buses.

with the multi-slave boards and without them is represented in the scheme in Fig. 3. Technical details on the implementation of the multi-slave behaviour of the interface boards are reported in Subsection 7.2.

6.2 *Real-Time Machines*

Real-time computers, such as Speedgoat or typhoon machines, represent one of the best solutions for testing and rapid prototyping of engineering systems. They are fully compatible with the Matlab & Simulink environment and their performances are extremely high with respect to the previous solutions, reaching up to 2 GHz clock frequency in real-time. Real-time machines feature a complete set of peripherals and they do not require the development of any interface between the OBC-ADCS and the simulation computer.

6.3 *Field Programmable Gate Arrays*

Field Programmable Gate Arrays (FPGA) have been used in the aerospace industry to design PIL and HIL test benches, with real-time performances up to the nano-second level [2, 6]. FPGA are versatile devices which can be perfectly adapted to the specific desired architecture and configuration, thus representing a good alternative for the implementation of interfaces and sensors models in the PIL/HIL facility. They are usually programmed using Hardware Description Language.

6.4 *Architecture Selection*

Comparing the characteristics of the alternative solutions with the PIL/HIL facility requirements, the Raspberry Pi, the real-time machines and the FPGA solutions have been discarded.

The Raspberry Pi computers feature enough power and memory to actually run even the more demanding sensors simulators. However, the presence of the Raspberry Pi operating system could downgrade the real-time performances with respect to Arduino. Moreover they are more expensive and they provide less software support with respect to Arduino, when used as slave and multi-slave devices in a I^{2}C or SPI bus.

Real-time machines, such as SpeedGoat and Typhoon, represent a valid solution, allowing to perform PIL and HIL tests with high performances. However, as discussed in Section 5, such performances are not needed for the testing of the HERMES spacecrafts and, in general, of the cubesat family. The high cost of these machines makes them not appropriate to cubesats: small satellites are often designed and built with low budget and they need a testing process compliant with their intrinsic low-cost and affordable nature.

FPGA have been discarded, mainly due to the highly specific technical skills required to program and set up them and the high cost.

The solution with microcontroller boards, used as data interfaces, with a non real-time simulation computer exploiting the SDRT framework, has been selected to realise the PIL/HIL architecture. Given the low frequency required to the ADCS, as reported in Section 5, a non fully real-time test-bench has been considered sufficient to perform the HERMES PIL/HIL tests. Indeed, even if the real-time performances of such test-bench cannot be guaranteed a priori, the system update rate, and in general its performances, can be measured once the test data have been acquired. If the test-bench is fast enough to potentially reach the required update rate, it is highly probable that the whole test can be considered valid.

The sensors/actuators models will run on the simulation computer, together with the dynamical and environmental simulator, since the Arduino board 32 kb flash memory is not enough to store the complete sensors simulators implementation. Moreover, VM of the sensors and actuators are already implemented on Simulink: an auto-coding procedure to implement the simulators in the interface boards would need additional verification on the generated code. Lower real-time performances are also expected by an Arduino board running a more complicated software with respect to the simple protocol handling.

As a final consideration, the family of PIC Microchip microcontrollers represents a valid and possibly better alternative to Arduino AVR chips, being more performing and flexible for application specific products [8]. The selection of the Arduino boards has been dictated by their simplicity, hiding the complexity of embedded systems. A natural step for future advancements of the test-bench is to exploit the PIC microcontrollers to substitute the Arduino boards.

7. PIL Facility Implementation

The general overview of the PIL test-bench, developed for the HERMES ADCS software verification, is reported in figure 4. The three main modules required to realise the facility are shown in the figure: the simulator computer, the interface units and the OBC running the ADCS software.

The HERMES PIL test-bench employs the virtual models already developed for the early MIL and SIL verification of the ADCS software, to simulate the environment, the dynamics and the sensors/actuators data. The virtual models are developed in the Matlab & Simulink environment and run on a standard desktop computer. The Simulink Desktop Real-Time kernel and its I/O library are used to reach soft real-time performances on the simulation computer, as explained in the following Subsections.

The hardware and software interface of the instruments have been developed exploiting the

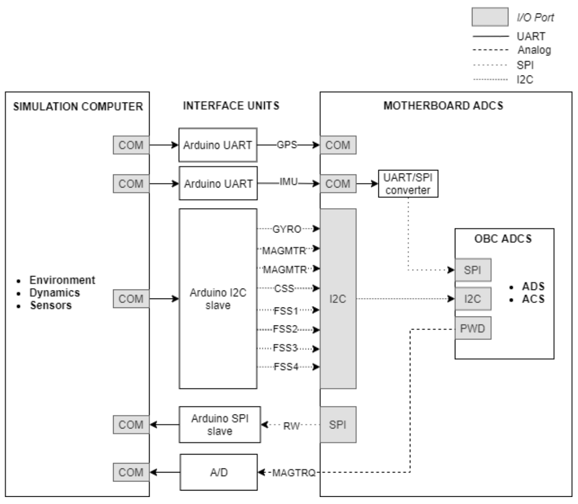

Fig. 4: PIL test architecture.

ATmega328P microcontroller of the Arduino Uno boards.

Finally, the ADCS-OBC is interfaced with the Arduino peripherals, receiving the sensors data and sending the actuators command. For the test-bench validation, a Raspberry Pi computer with a deployment of the ADCS flight software is planned to be used, before proceeding with the actual hardware.

7.1 *Simulator Computer*

The simulator computer is a standard Windows or Mac Os workstation, with a Matlab & Simulink release. Two issues are related with simulations for a PIL test on a non real-time machine: the uncertain performances of the machine, and the loss of synchronization between the simulator and the hardware under test. As already described in Subsection 6.4, the performances can be checked after the test, given a machine which is theoretically able to reach them. However, the synchronization problem is harder to deal with: the Simulink Desktop Real-Time kernel can be exploited for this purpose. Simulink Desktop Real-Time is a framework for testing and rapid prototyping developed by The Mathworks and integrated in the Matlab & Simulink environment. It provides a real-time kernel and a library which together allow to perform real-time simulations on standard Windows or Mac Os machines. The real-time kernel interacts with the operating system and assigns the highest priority of execution to the simulation executable, allowing it to run without interference at the selected sample rate. During real-time execution, the kernel intervenes to give the model priority to use the CPU, executing each model update at the prescribed sample times. Once a model update completes, the kernel releases the CPU to run other operating system-based applications that need servicing.

Simulink Desktop Real-Time features two operating modes: a normal mode, in which the simulation still runs in Simulink, faster than the real-time, while the kernel forwards the simulation data to the hardware at the correct time; an external mode, in which the simulation is automatically coded, to produce an executable running in kernel mode, but still communicating data back to Simulink for visualization and tuning. In normal mode performances up to 1 kHz are reachable, whereas in external mode the simulation can run up to 20 kHz. For some time instants, the simulation cannot catch-up with the real-time clock, and the real-time performances are lost. The SDRT framework allows to identify when this happens exploiting a *missed ticks* counter which can be used later on to validate or discard the performed test. The performances of the Simulink Desktop Real-Time kernel are well beyond the required ADCS performances of the Hermes spacecrafts, justifying its choice. Moreover, exploiting this tool, the synchronization between the simulation and the hardware are guaranteed, since the simulation clock is synchronized with the real-time clock every time no missed ticks happened.

The dynamical and environment simulator and the sensors/actuators simulators have been developed on Matlab & Simulink by the ASTRA research group. The characteristics and performances reported in the datasheets of the physical sensors/actuators have been applied to accurately reproduce the output of the sensors.

7.2 *Interface Modules*

To allow the data communication between the simulator computer and the OBC-ADCS, hardware interfaces need to be developed. The required interfaces are UART, I^2C and SPI, and have been implemented exploiting the Arduino Uno board, which features the characteristics reported in Subsection 6.1.

The magnetorquers communicate with the ADCS-OBC through an analog signal, which can be forwarded to the simulation computer exploiting an analog to digital converter.

7.2.1 *UART Interface*

A UART (Universal Asynchronous Receiver-Transmitter) is a physical device for asynchronous, serial communication, in which the data format and transmission speeds are configurable. Two technical standards are used in this project: the single-ended RS-232 and the differential RS-422. Since each interface board must exchange serial data between the simulator and the ADCS-OBC, two serial ports are required on the board. The availability of a single port on the Arduino Uno boards determined the need for the Arduino *SoftwareSerial* library, to open a software serial port on two GPIO pins.

The UART interface is used by the IMU and by the GPS receiver. The IMU communicates through a UART RS-422 interface. A simple communication protocol is implemented by the sensor, in which data to be transmitted are preceded by a 4 bytes header and followed by a 2 bytes checksum. The IMU data messages can be directly built in Simulink: the acceleration, angular velocity and magnetometer data, generated by the dynamical and environmental simulator, are first converted to single precision floating point and then to a byte vector. Finally the message header and the checksum are included in the bytes to send, and the Simulink Desktop Real-Time I/O library functions are used to forward the data to a USB port on the simulation computer.

A USB to UART converter can be used to forward the message directly from the simulation computer to the Hardware OBC. However, data messages are not the only information exchanged between the IMU and the ADCS-OBC software: at the system startup, the ADCS-OBC sends a series of initialization commands, to which the IMU shall immediately reply. Since the simulation model advances by steps, several simulation steps would be required to receive and respond to initialization commands directly on Simulink. A more efficient solution to handle the initial communication, requires an intermediate interface to reply to initialization commands and exchange data messages. One Arduino board is used for this purpose. A solution which avoids the configuration messages of the ADCS-OBC is not feasible; indeed, it would modify the flight software, which is not allowed, since the tests shall verify the ADCS software in flight configuration.

The GPS module supports both RS-232 and RS-422 standards, with the possibility to send ASCII or binary messages. The GPS receiver features hundreds of logs and a more complex serial protocol with respect to the IMU. In particular, the receiver is capable of generating three type of logs: synchronous logs, in which data are generated on a regular schedule; asynchronous logs, which output the most current data as soon as it is available; polled logs, i.e.

generated on demand.

A complete reproduction of the GPS receiver interface is a difficult and laborious task, which, indeed, is not needed. The HERMES PIL test, only requires three synchronous logs from the GPS at a frequency of 10 Hz, to retrieve the following data and the related status flags: latitude, longitude, height, undulation, time, time offsets, position and velocity. GPS data are computed by the simulator and sent to the serial port, where an Arduino board receives and stores them in data structures. The GPS serial protocol is completely handled by the Arduino interface board in which a simple Finite State Machine (FSM) is implemented. The FSM scheme is reported in Fig. 5: the algorithm always passes from three states, in which it checks for available simulation data (from the simulator) and log requests (from the ADCS-OBC), and finally sends the requested synchronous logs.

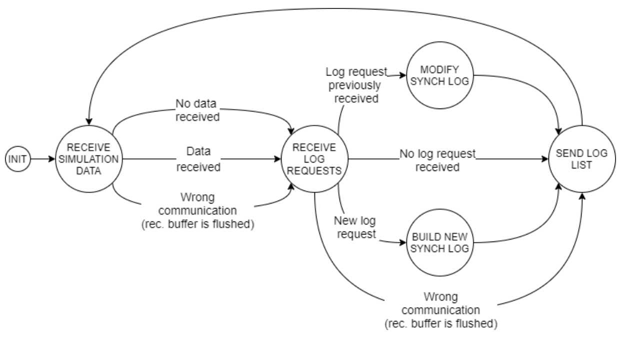

Fig. 5: Scheme of the interface FSM for GPS serial protocol handling

When a new log request is received, the list of the logs to be send is updated and the new log is sent at the specified time during the *send log list* state. If a log request which was already received is sent again, the parameters which set the sending time interval are updated, and the log is sent in the next state. In case of wrong serial data reception, the buffer is flushed and the code continues to follow the main line of the FSM.

7.2.2 *I2C Interface*

The I^2C (Inter Integrated Circuit) is a synchronous, multi-master, multi-slave, single ended serial communication bus. The bus uses a single data line: each slave device is identified by a unique 7-bit slave address which is called by the master device to send or receive data from it [10].

The I^2C interface is used by the OBC gyroscope, the OBC magnetometer, the main magnetometer, the CSS set and the FSS set. A simple USB-I^2C converter can be used to transmit the sensors data only, but not to simulate the behaviour of the hardware sensor interface; in particular, the need to receive data addressed to internal registers of the sensor and to respond consistently, led to the decision to use a microcontroller to handle the communication.

To avoid the use of one microcontroller unit for each sensor, a single board has been set up to behave as a *multi-slave* device. The *TwoWireSimulator* Arduino library, developed by Gaziello [5] has been used to let the board respond to master calls sent to multiple slave addresses, which allowed to simulate the I^2C interface of all the instruments with a single microcontroller unit. The emulation of multiple I2C slave devices with a single Arduino board requires the modification of the Arduino *Wire* library controlling the I^2C periferal. For this purpose, the Two Wire Interface (TWI) registers on the ATmega328P microcontrollers can be exploited: in particular, the chip features two TWI registers that can be used for this scope: the TWI Address Register (TWAR), which stores the device address, and the TWI Address Mask Register (TWAMR), which can be loaded with a 7-bit Slave Address mask. Each of the bits in TWAMR can mask (disable) the corresponding address bit in the TWAR: if the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWAR [1]. The working principle of the TWI registers is shown in Fig. 6.

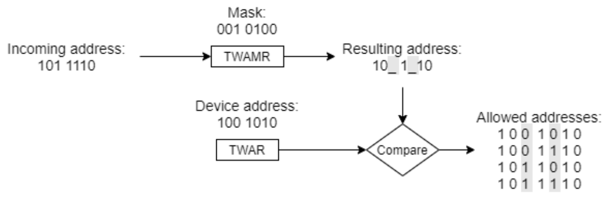

Fig. 6: TWI address register masking logic.

Exploiting the TWAMR register in the *TwoWireSimulator* custom Wire-based library, the Arduino interface can be programmed to mask the device address requested by the master call. This would cause the board to respond to master calls addressed to multiple I^2C devices. The incoming address can then be recovered from the Two Wire Data Register on the chip, which, in receive mode, stores the last byte received.

For the HERMES case, only 8 addresses need to be disabled, since the 12 CSS transmit their data as a single packet; the required slaves are therefore 2 magnetometers, 1 gyroscope, 1 CSS (including data of all

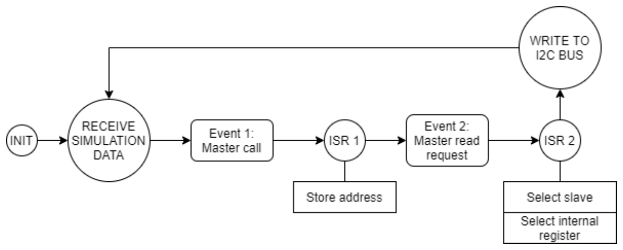

Fig. 7: I^{2}C interface software logic.

CSS) and 4 FSS. However, all the 7-bit addresses on the I^{2}C bus have been disabled. This was necessary since the slave addresses of the sensors are very different one from the other, and a mask to disable only the required 8 addresses, was unfeasible.

The overall I^{2}C software logic running on the Arduino board is extremely simple, and it is shown in Fig. 7: the microcontroller continuously checks the serial port to receive and store the sensors data from the simulation computer as soon as they are sent.

Once a I^{2}C master call is detected on the bus, two Interrupt Service Routines (ISR) are called: when the master writes the slave address on the bus, the first ISR saves it. Then the master device sends the data request to the slave, where a second ISR checks which slave address was received, reads the internal register requested by the master and responds consistently, exposing the data on the bus.

7.2.3 *SPI Interface*

The SPI (Serial Peripheral Interface) is a synchronous serial communication protocol using a minimum of four wires, for data transmission, data reception, clock and slave addressing. The slave addressing implements a totally different approach with respect to the I^{2}C protocol: a dedicated line, called Chip Select (CS) or Slave Select (SS) is triggered low any time the master starts the communication with the slave [7]. Every slave needs its own SS line, therefore the master device must have a number of SS pins equal to the number of slave devices it wants to communicate with.

The reaction wheels of the HERMES spacecrafts communicate with the ADCS-OBC through a SPI bus. Also in this case, Arduino boards are used to handle the communication between simulation data and the ADCS-OBC. To close the PIL test loop, the control parameter shall be transmitted to the actuators by the ADCS-OBC and forwarded to the simulation computer. A ISR is triggered on Arduino every time a byte is received through the SPI bus and once the full actuator message is recognised, it is immediately sent to the simulator.

For the HERMES project, one Arduino board for each reaction wheel need to be used. However, a method to use a single board as a multi-slave device, to respond to master calls to different slaves, has been considered. The Arduino board has one single SS pin, when used as a slave SPI device. The multi-slave behaviour can be achieved if two problems are solved:

- drive low the SS pin of the Arduino any time one of the four ADCS-OBC SS pins goes low;
- recognise which slave the master has called (or which SS pin of the master has been driven low).

The GPIO pins of the Arduino board can be used to identify the selected slave: each SS pin of the master device can be connected to an Arduino GPIO pin, whose reading allows to understand which slave the master wants to communicate with. Then a hardware logic gate can be implemented to drive the Arduino SS pin low any time even a single SS pin is driven low by the master. The SPI communication architecture, with the logic gate driving the input signal in the master SS pin, is shown in Fig. 8. The scheme of the logic gate is represented in Fig. 9 and the correspondent truth table is reported in table 4.

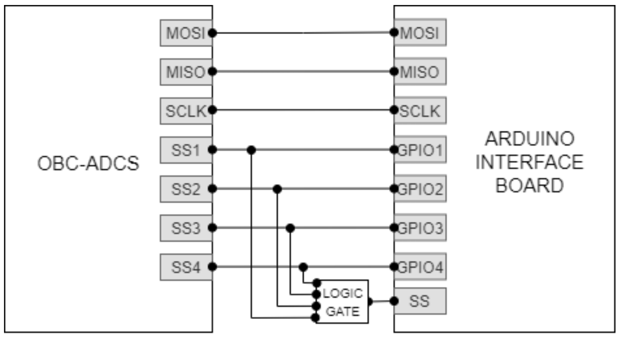

Fig. 8: SPI multi-slave architecture.

SS 1	SS 2	SS 3	SS 4	SS Out
1	1	1	1	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

Table 4: Truth table for the logic gate in Fig. 9

The logic gate is composed by four p-mos transistors connected in parallel, four n-mos transistors in series and a NOT gate. SS 1, SS 2, SS 3, and SS 4 are

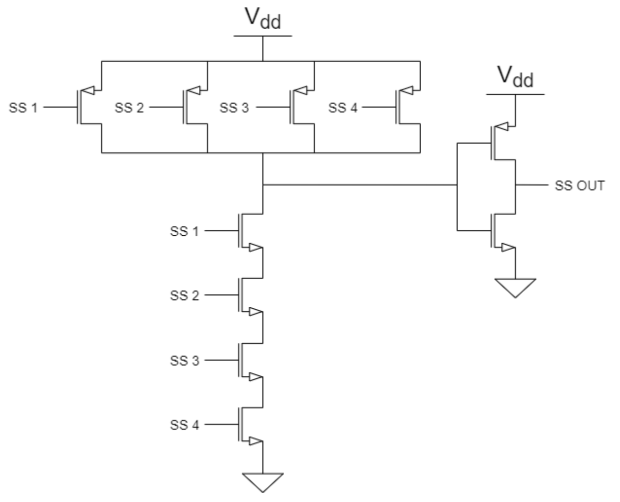

Fig. 9: Logic gate to drive the Arduino SS pin low if one of the master SS pin are driven low.

connected to the four SS chip of the master device, whereas SS Out feeds the Arduino SS pin.

8. Final Remarks

The AIV/AIT plan developed for the HERMES mission requires a complete testing campaign from equipment level to fully integrate system. The ADCS represents a crucial element from the performances point of view, and the AIV/AIT campaign for this subsystem involves three different testing models: VM, PFM and FM. functional, interface, performance, mechanical and environmental tests are planned at equipment level on the single COTS components. With the same purpose, tests at subsystem level shall be performed and a subsequent qualification campaign is planned on the PFM and FMs.

In this framework, a low-cost cubesat oriented test-bench for PIL/HIL validation of the ADCS software has been developed, exploiting the Matlab & Simulink soft real-time kernel and the capabilities of 8-bit microcontroller units. The interface modules between the simulation computer and the ADCS-OBC have been implemented using Arduino boards to reproduce the hardware behaviour and the communication protocols of sensors and actuators units. The number of microcontrollers units have been minimised, developing or exploiting tools which allow the use of the Arduino boards as multi-slave devices on I^2C and SPI buses.

Advancements of the presented PIL/HIL architecture move forward in two directions: include the use of high-performance and application-specific hardware (e.g. substitute the Arduino boards with Microchip PIC microcontrollers); standardise the procedures and the facility components to simplify the implementation of the architecture.

References

[1] Datasheet ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. Atmel corporation. *San Jose, CA*, 2015.

[2] Xunhua Dai, Chenxu Ke, Quan Quan, and Kai-Yuan Cai. Unified simulation and test platform for control systems of unmanned vehicles. *arXiv preprint arXiv:1908.02704*, 2019.

[3] European Cooperation for Space Standardization. ECSS-E-ST-10-02C rev.1, “space engineering – verification,” standard. Feb. 2018.

[4] European Cooperation for Space Standardization. ECSS-E-ST-10-03, “space engineering – testing,” standard. Jun. 2012.

[5] Alexis Gaziello. Twowiresimulator. *GitHub repository.* `https://github.com/alexisgaziello/TwoWireSimulator`, 2016.

[6] William Guareschi, José Azambuja, Fernanda Kastensmidt, Ricardo Reis, Otavio Durão, Nelson Schuch, and Gustavo Dessbesel. A configurable test bed platform for test and validation of payloads for nanosatellites. In *Workshop on the Radiation Effects on Electronics and Photonic Devices for Aerospace Applications*, volume 4, pages 18–21, 2012.

[7] SPI Block Guide. V03. 06.: Motorola. *Inc., February*, 2003.

[8] Rajratna Khadse, Nitin Gawai, and Bagwan M Faruk. Overview and comparative study of different microcontrollers. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, pages 311–315, 2014.

[9] Martin Langer and Jasper Bouwmeester. Reliability of cubesats-statistical data, developers' beliefs and the way forward. In *USU Conference on Small Satellites*, 2016.

[10] UM10204 NXP. I2c-bus-specification and user manual, apr. 4, 2014. *NXP Semiconductors, Revision*, 6.

[11] Michael Swartwout. The first one hundred cubesats: A statistical look. *Journal of small Satellites*, 2(2):213–233, 2013.

[12] Thyrso Villela, Cesar A Costa, Alessandra M Brandão, Fernando T Bueno, and Rodrigo Leonardi. Towards the thousandth cubesat: A statistical overview. *International Journal of Aerospace Engineering*, 2019, 2019.