## POLITECNICO DI TORINO
## Repository ISTITUZIONALE

Modularity and Reliability in Low Cost AOCSs

(Article begins on next page)

23 April 2024

# Modularity and Reliability in Low Cost AOCSs

Leonardo M. Reyneri[1], Danilo Roascio[1], Claudio Passerone[1],
Stefano Iannone[1], Juan Carlos de los Rios[1], Giorgio Capovilla[1],
Antonio Martínez-Álvarez[2] and Jairo Alberto Hurtado[1]

*[1]Department of Electronics and Telecommunications, Politecnico di Torino, Torino,*
*[2]Department of Computer Technology, University of Alicante, Alicante,*
*[1]Italy*
*[2]Spain*

## 1. Introduction

The use of off the shelf electronic components is now common in university and low cost satellites. The advantage against space qualified parts consists in a significant cost reduction, a wider range of components selection, better second sourcing capabilities and an effective reuse of existing technologies, devices, circuits and systems from other engineering domains.

Commercial Off The Shelf Components (**COTS**) are sometimes subject to reliability requirements which are often tougher than those applied to space devices, as they have to be used in markets (e.g., automotive) where safety concerns and the huge number of systems manufactured set demanding constraints on the components. Yet, the drawbacks of COTS components and other low cost design methods, mostly in space missions, remain the higher sensitivity to radiation-induced effects and the reduced system level tolerance to faults.

Compensation of these weak points is possible with the development of appropriate design techniques and their proper application throughout the lifecycle of the system, from system level design down to manufacturing. The use of COTS components has thus enormous capabilities and benefits in, but not limited to, small satellite missions.

Low cost spacecraft design not only refers to COTS devices but to several other aspect of the design of a spacecraft.

A low-cost approach to spacecraft design will have a huge impact on the development of space technology in the future, provided that system-level approaches are applied to the design in order to contemporarily reduce cost and increase reliability.

This chapter will analyze several aspects in the low cost and high reliability design of a specific spacecraft subsystem, namely an Attitude and Orbit Control System (**AOCS**), developed at Politecnico di Torino as part of the AraMiS modular architecture for small satellites. We will show how an appropriate mixture of innovative techniques can produce a high reliability and high performance, low cost AOCS.

The topics will be covered at both system and subsystem level, with references to commercially available devices (sensors, actuators, drivers and microcontrollers) and

following the object-oriented modeling used for both system management, subsystem development and programming.

This chapter will therefore present several technical solutions (at different design levels) which the authors have developed and applied to the development of the AraMiS built-in AOCS system. In particular, after analyzing the **Requirements on small satellites AOCS**, the following items will be discussed in detail: i) **Attitude and Orbit Codetermination**, a mean to substitute the high-cost space-grade GPS with an appropriate analysis of the images acquired by our solar and horizon sensor; ii) **Fine-grain Modularity**, a mean to cut down design, development, testing and assembly costs, while maintaining a high level of design flexibility; iii) **Sensor Fusion and Reconfigurability**, a system-level mean to increase reliability of low-cost sensors and actuators by processing the data from the large network of sensor intrinsic in the architecture of the AraMiS spacecraft; iv) **Latchup Protection**, achieved by a hybrid anti-latchup protector developed by the authors; v) **Open hardware/software plug and play architecture,** allowing a simple and straightforward integration of several sensors and actuators into the SW architecture of the spacecraft, while minimizing the risk of errors and guaranteeing a high level of reliability, also thanks to the pervasive use of the proposed vi) **Hardening techniques against radiation of SW code and HW interfaces HW/SW**; vii) the application of an innovative **micropropulsor** found in literature is considered for future developments; viii) the aspects of **Sensor and Actuator Calibration** will be analyzed and it will be described how these will be integrated into the HW/SW architecture of AraMiS.

## 2. System overview

The proposed AOCS has been conceived as a part of the AraMiS modular architecture for small satellites developed at Politecnico di Torino (Reyneri et al., 2010). The basic idea behind AraMiS is the concept of **tile**, that is, a standardized building block to be used to build small spacecrafts according to the specific requirements. As many tiles as required can be assembled to build spacecrafts of virtually any size and shape, as shown in figure 1.
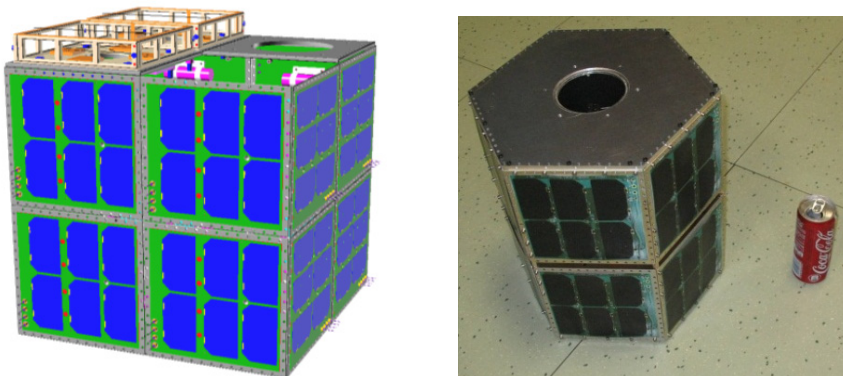


Fig. 1. Mechanical sketch of a 2x2x2 cubic configuration using 22 tiles (left). Photograph of the mockup of ah hexagonal telescope made using 12 tiles.

Each tile contains a number of attitude and housekeeping sensors and actuators, such that a spacecraft made of a number of tiles incorporates all the standard subsystems needed,

among which the AOCS which is being described in this chapter. Note that, although the AOCS will be described as a whole, in practice its subsystems are distributed among all the tiles, that is, throughout the spacecraft. Several functions are therefore replicated multiple times, offering a relevant degree of redundancy evaluated in detail later in the chapter.

Figure 2 shows a simplified overview of the proposed AOCS (internally called 1B2 Attitude and Orbit Subsystem), seen in its entirety. It is made of five major blocks.
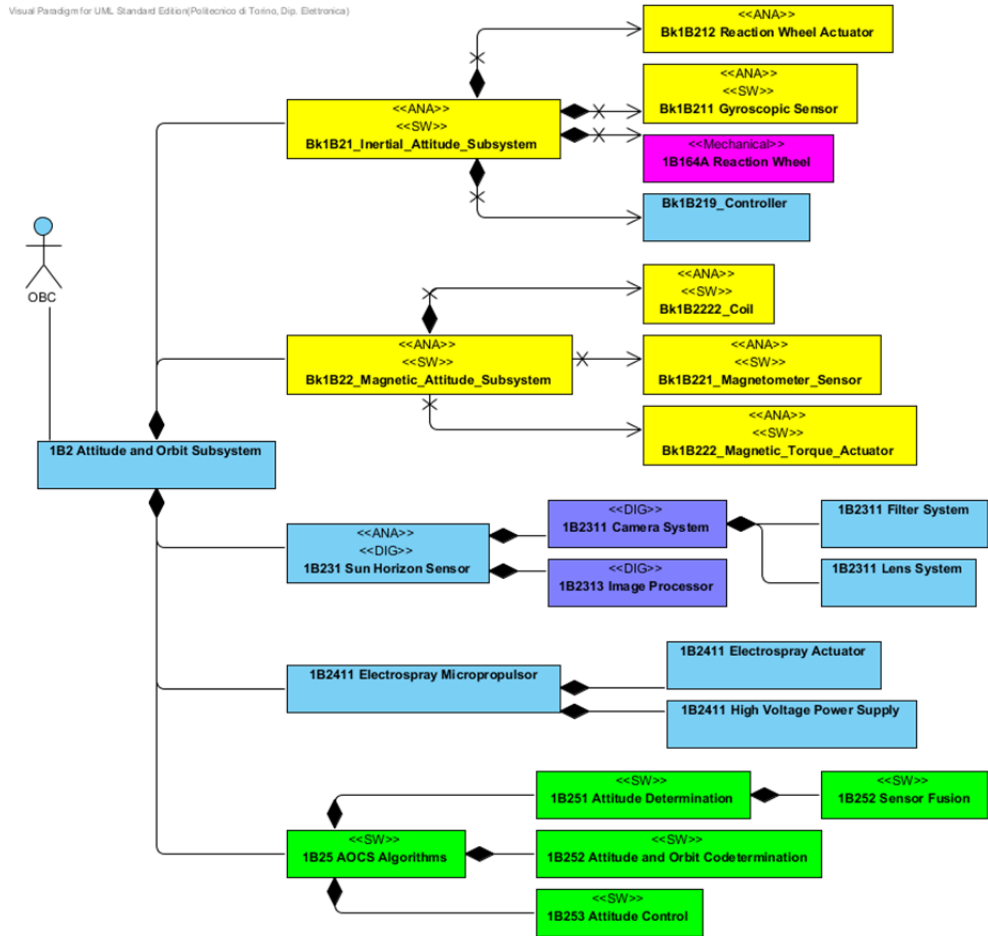


Fig. 2. Overview of the proposed Attitude and Orbit Control System.

- A Bk1B21_Inertial_Attitude_Subsystem, aimed at attitude determination and control with inertial techniques. It is composed of a Bk1B212 Reaction Wheel Actuator (namely, a brushless micromotor to operate a small 1B164A Reaction Wheel), a Bk1B211 Gyroscopic Sensor to measure satellite spin, plus a software Bk1B219_Controller to interpret user and attitude commands from OBC and drive the other subsystems accordingly.

- A Bk1B22_Magnetic_Attitude_Subsystem, aimed at attitude determination and control using magnetic techniques. It is composed of a Bk1B222_Magnetic_Torque_ Actuator (namely, a power driver driving an appropriate magnetic Bk1B2222_Coil), a Bk1B221_Magnetometer_Sensor to measure satellite attitude with respect to earth magnetic field; an embedded software controller (not shown) interprets user and attitude commands from OBC and drive the other subsystems accordingly.
- An optical 1B231 Sun Horizon Sensor to measure satellite attitude with respect to sun and earth horizon. It is made basically of a 1B2311 Camera System (incorporating a 1B2311 Filter System plus a 1B2311 Lens System) and a 1B2313 Image Processor which interpret images taken from the camera. It returns (to the OBC) the satellite attitude, whenever either sun or earth is visible.
- An 1B2411 Electrospray Micropropulsor, not yet implemented as its possible usage is currently under analysis. An appropriate micropropulsor has been taken from literature as a driver example to evaluate its applications to the proposed small satellite.
- A set of appropriate 1B25 AOCS Algorithms to gather and fuse magnetic, attitude and optical sensor measurements and to drive magnetic and inertial attitude actuators.

## 3. Applications and requirements of small satellites AOCS

The goal of the AOCS is to determine the position of the satellite along the orbit and its attitude, i.e. where pre-defined axes of the satellite are pointing to. A detailed mathematical treatment of the AOCS problem will be considered in the next sections, while here the focus will be on its requirements and use.

The ability to determine the satellite position and attitude is of paramount importance in most space missions, as there are only a few and special cases where these capabilities are not needed. Satellite applications can be divided in a number of broad classes, among which the most common are: i) Telecommunication, ii) Earth observation, iii) Science and iv) Navigation. In each of these cases the AOCS plays a role to achieve the mission goals, but requirements with respect to several metrics can be extremely different, even within missions belonging to the same class. The obvious consequence is that methods and technologies vary considerably among various cases, which makes the choice of an AOCS architecture a difficult task when designing a satellite.

Typical applications of an AOCS subsystem can be summarized as follows.

- **Despinning** of a satellite after deployment. The launcher generally releases the satellite with an unknown spin, which must be compensated prior to starting the mission. An initial coarse attitude determination is sufficient, while orbit control can often be neglected, as the focus is on stabilizing satellite attitude. Despinning may take a long time (e.g., several orbits), but is applied only once during the life of the satellite.
  Despinning can be useful in other situations, as well. Light pressure or atmospheric drag can impose unknown spins to the satellite, so this procedure might be occasionally applied to recover from them. However, the magnitude of the correction is typically much smaller with respect to the initial deployment case.
- **Pointing** of the satellite towards a specific direction. This task has a number of uses depending on the application. In telecommunication missions, it is needed to accurately direct the satellite antennas lobe to the ground stations to improve the radio signal

strength. In Earth observation, instruments such as cameras shall be pointed towards those locations that are of interest for the application. Science mission often need to know the attitude of the satellite, as well.

Pointing can be used in different ways. Some satellites need to always point to the Nadir, regardless of their position in the orbit. Other satellites need to constantly track a given point on the Earth surface, and thus require a constant attitude correction as the satellite moves along the orbit. In other cases, inertial pointing is necessary, i.e. the ability to always point towards the same direction with respect to a fixed reference frame.

The applications of pointing are too many to list them all here, and strongly depend on the satellite mission. In many cases, pointing is used for different goals, even within the same mission. For instance, many satellites will also use pointing to steer the solar panel arrays in order to increase the generated electrical power, or to avoid forbidden attitudes that may endanger the spacecraft.

- **Positioning** is also needed in many different cases. When coarse orbit determination is sufficient, the orbit position can be computed using a mathematical model starting from ephemerides, which can be periodically updated from ground. However, if the satellite requires to autonomously determine its position, other techniques based on several kinds of sensors are used.

  The ability to know the satellite position is extremely important in navigation missions, as these satellites are the base for many other applications that require accurate position determination, on the Earth or in orbit. However, positioning is also important in Earth observation and telecommunication satellites, especially when coupled with pointing: a typical example is that of pointing a specific location on Earth, which requires both abilities to work together to reach the final goal.

- **Deorbiting**, namely the capability to move to a low enough orbit (at the end of mission) to guarantee the spacecraft destruction impacting Earth atmosphere. This is desirable (and soon it may become compulsory) to keep orbit pollution by space debris within reasonable limits.

- **Formation Flight and Docking**, where sets of satellites cooperate to reach a common goal and where their position must be carefully controlled. The AOCS must provide the capability to slightly correct the orbit, to keep the relative distance among a set of spacecrafts within given bounds, or to control the distance among them accurately to perform a safe docking.

All the applications listed above, and many others as well, use the AOCS subsystem with different requirements, and several metrics shall be considered in order to evaluate its performance. One of the most important is the **accuracy**, both of pointing and of positioning. Antennas pointing, especially for small satellites in LEO orbit which don't have sophisticated radio systems, does not usually require a fine accuracy, as the transmission lobe is generally broad enough to allow for large errors. Typical values are in the order of several degrees. Similarly, a small camera with a short focal length objective can image a wide area on the Earth surface, thus not needing high accuracy in order to frame a specific target within the picture. Longer focal lengths coupled with small image sensors impose more stringent requirements on the pointing accuracy, as other applications, especially in science missions, also do.

Another important aspect of pointing is its **stability**, i.e. the ability to keep the satellite pointing towards a specific direction for a given amount of time. Consider again a small camera with a short focal length objective: while the absolute pointing requirements might not be very tight due to the large area swath, a single pixel will only cover a very small angle, which should not change during the acquisition of the frame. For example, the angle covered by a 5µm pixel coupled with a 50mm objective is around 20 arcsec, corresponding to approximately 60m on ground as seen from a 600km high orbit. To avoid smearing of the picture due to movements of the satellite, a very high precision in pointing should be guaranteed for the entire period of time needed to take a shot. While this can be a short time (1/100s or less) for pictures in visible light taken during the day, it can be significantly longer for night pictures, in other bands of the electromagnetic spectrum, or when imaging stars or other celestial objects in astronomical related missions.

Performance of the AOCS system is also related to the **pointing time**, namely the time needed to complete a given pointing or positioning command. Higher accuracy usually requires a longer time, but this has a negative impact on the number of activities that the satellite can carry out during the lifespan of the mission. Large changes in pointing or positioning also require a longer time, due to more complex maneuvers, so a careful scheduling of the satellite tasks can minimize these kinds of overheads.

Finally, **cost**, **size** and **mass** should also be considered when designing an AOCS subsystem. These metrics deal more with the technologies and the kind of sensors and actuators that are involved in the satellite. Small satellites often require several trade-offs that may sacrifice accuracy and performance to the sake of decreasing costs; larger satellites can accommodate better sensors and actuators, thus achieving better metrics. Modularity can significantly increase the efficiency and availability of the AOCS, allowing the satellite to get good accuracy and performance at moderate costs, as will be detailed in the rest of this chapter.

## 4. Attitude and orbit codetermination

A satellite in the free space, corresponds to a model of an object with six degrees of freedom, rotation in three dimensions and translation in a three dimensional space.

The attitude of a body, in our case a satellite, is its angular position or orientation with respect to a defined frame of reference. Attitude in a rigid body can be represented by Euler angles, heading (rotation about Z-axis, $\psi$), elevation (rotation about Y-axis, $\theta$) and bank (rotation about X-axis, $\phi$). A diagram of this is shown in Figure 3.

In a similar way, the attitude of a satellite can be represented by a rotation matrix, defined by Euler angles, corresponding to a series of positive right hand rotation. If a 1-2-3 Euler rotation is used, the rotation matrix will be:

$$Q = \begin{bmatrix} C\theta C\psi & C\theta S\psi & S\theta \\ S\phi S\theta C\psi + C\phi S\psi & -S\phi S\theta S\psi + C\phi C\psi & -S\phi C\theta \\ -C\phi S\theta C\psi + S\phi S\psi & C\phi S\theta S\psi + S\phi C\psi & C\phi C\theta \end{bmatrix} \tag{1}$$

where $\phi$, $\theta$, and $\psi$ are the Euler angles, while $C\theta = \text{Cos }\theta$, $S\Psi = \text{Sin }\Psi$.
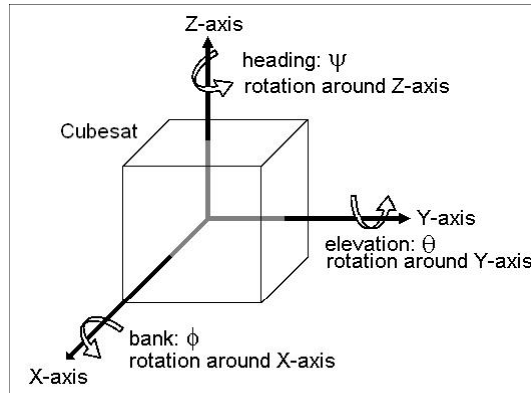
Fig. 3. Satellite coordinate system and Euler angles.

A part of the AOCS is the Attitude Determination and Control Subsystem (ADCS). It is a set of sensors, actuators and software that must be able to determine, change or keep the attitude of the satellite within a predefined range of values.

Attitude determination can be achieved through sensors such as gyroscopes, Sun, Earth and horizon sensors, orbital gyrocompasses, star trackers and magnetometers. A body sensor, such as Earth, Moon and Sun sensors, is a device that senses the direction to the respective object and returns the unit vector to the corresponding body.

A **Sun and Horizon sensor** can be based on solar cells, photodiodes, CCD cameras or CMOS cameras; these devices can determine the incidence angle (unit vector orientation) of the sun with respect to the normal to their plane. However, this information only gives a cone of possible sun positions and another sensor (e.g., a magnetometer) must be used to solve the ambiguity.

When a Sun sensor is used, the terrestrial albedo can generate errors especially in low Earth orbit (LEO), but it can be corrected using appropriate processing.

The second vector that will be used to estimate the attitude will be taken from a **Magnetometer**. It measures direction and magnitude of Earth's magnetic field and the measurement is compared to a simulated model or with a previously available map of the expected magnetic field. To avoid errors in the measurement of the magnetic field, all other (active and passive) parts inside the satellite must have a minimal residual magnetic field.

## 4.1 Attitude determination

To determine the attitude, a set of three sensor-measured vectors (with a non-null cross product between them) is used with another set of three corresponding reference vectors (based on models, simulations or data stored on memory) to find a rotation matrix Q, as shown in (1), that satisfies:

$$m = Q \cdot r \qquad (2)$$

where $m$ = measurement vector and $r$ = reference vector. The first vectors of the sets are obtained with:

$$\hat{a}_1 = \frac{m_1}{|m_1|}, \quad \hat{b}_1 = \frac{r_1}{|r_1|} \tag{3}$$

The second vector in each set ($a_2$, $b_2$), is given by the second pair (vector measurement and the corresponding reference, $m_2$ and $r_2$, respectively). The third vectors are the result of a cross product of the first two vectors in each set:

$$\hat{a}_3 = \hat{a}_1 \times \hat{a}_2, \quad \hat{b}_3 = \hat{b}_1 \times \hat{b}_2 \tag{4}$$

So, like these vectors, the sets satisfy:

$$A = Q \cdot B \tag{5}$$

Where A and B are the two matrices 3x3 made with the set of three vectors. Finally, the attitude matrix Q, is determined by:

$$Q = A \cdot B^{-1} \tag{6}$$

In some cases there may be singularities, for instance when the first two vectors are parallel to each other.

In this case, the two main vectors are given by the Sun sensor and a magnetometer; with these vectors the analysis can be completed. However, since the Sun sensor will not work properly in the eclipse phase of the orbit (near to 40 minutes in LEO), a third vector can be used. To have a set of three measurement vectors, the sensor suite is complemented with a gyroscope which senses satellite spin without any external reference object. MEMS gyroscopes (Micro Electro-Mechanical Systems) have a reduced size making them suitable to be used in AraMiS.

All measurements from sensors are processed by the microcontroller. Once the reading is done, the microcontroller applies the algorithms to estimate the attitude of the satellite.

## 4.2 Optical orbit determination

Another processing that must be done is to estimate the orbit of the satellite. Twenty years ago, tracking and processing in the ground stations was used for orbit determination. Later, some satellites (LEO in particular) computed their own positions using GPS receivers. It reduced the work that ground stations should do to collect and transfer data.

Now the idea is to have another option to estimate the orbit without the use of GPS while still reducing the work of the ground segment, and this could be feasible using image sensors. The satellite, capturing an image of the Earth, can use this information to estimate its own orbit.

The processing is based in several parts: image capture of the Earth, knowledge of the satellite attitude, shine-dark zones detection on Earth's surface, Sun position.

First step in the process is to detect when the image shows a day-night transition or vice versa. In each rotation the satellite could detect two kinds of transitions (day-night or night-day). See Figure 4.
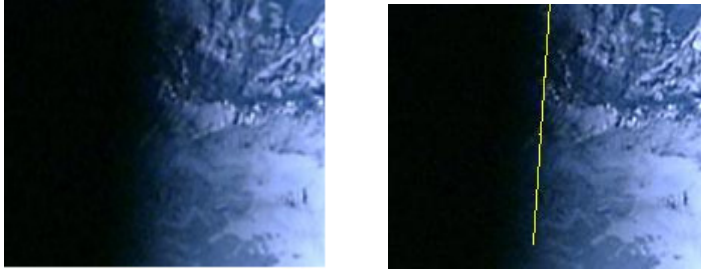
Fig. 4. Night-day transition (left). Inclination angle of the transition (right).

Once the transition is detected, the inclination angle of this transition in the image must be calculated, as shown in Figure 4. This inclination angle on the image must be corrected with the attitude angles of the satellite. Now, taken this angle as a reference, we can calculate the inclination angle of the orbit satellite with respect to the shine-dark angle.

From the image some reference points are taken and they are compared in the successive frames to estimate their movement in the sequence. With at least two different images, the trajectory of the points can be estimated as shown in Figure 5.



Fig. 5. Sequence of images tracing a point (left and center). The angle between the two lines (right).

The angle between the shine-dark line (calculated previously) and the points trajectory can then be estimated, as shown in Figure 5.

Finally, having the angle between the border in the image and the relative movement of the points on Earth, applying the satellite attitude and Earth's axial tilt correction factors it's possible to estimate the orbit angle of the satellite. This angle will have some error, but after several orbits this error will decrease.

With this process, the orbit inclination angle can be estimated. Trajectory prediction, instead, is still in development.

## 5. Fine-grain modularity

In electronics, a useful and alternative design methodology to reduce system complexity, among others, is the modularity; whereby, through the usage of standardized proper interfaces, it is possible to split the main functions of a complex system into some independent components (modules). Thanks to modularity, our system also becomes

flexible because it can be configured in several ways just plugging/unplugging few modules.

The main goal of this project is to integrate in an innovative way, for the development of nano and micro satellites, mechanical structure, harness, signal routing, and other basic and common functions like solar panels, into standard off-the-shelf modular panels. This new approach is expected to allow the development of new space missions with a mass, cost and time budget significantly lower than present missions.

The outcome will be a standardized modular platform, equipped with power management and attitude control subsystems, integrated harness and data routing capabilities, for low-cost nano and micro satellites.

One of the keys for achieving the goal is the creation of loosely coupled component designs by specifying standardized component interfaces that define functional, spatial, and other relationships between components. Once specified, this will allow all of the developers to design their own components in an independent way. In the AraMiS modular architecture, major bus functions are split over a number of properly placed and identical modules, so to simplify design, maintenance, manufacturing, testing and integration.

The modules interconnect and dynamically exchange data and power in a distributed and self configuring architecture, which is flexible because standardized interfaces between components are specified. Product variations can then take place substituting modular components without affecting the rest of the system. This design strategy offers, additionally, a high degree of built in redundancy and different configurations providing a larger number of system variations adaptable to different missions, spacecraft sizes and shapes.

AraMiS is characterized by its design reuse, which allows to effectively reduce as much as possible design and non-recurrent fabrication costs (e.g., qualification and testing costs), while reducing as well time-to-launch.

The basic architecture of AraMiS is based on one or more small intelligent modules (**tiles**), located on the outer surface of the satellite. The inner part of the satellite is mostly left empty to suit several kind of user-defined payload, which is the only part to be designed and manufactured ad-hoc for each mission.

Each tile is designed, manufactured and tested in relatively large quantities. There is an increased design effort to compensate for the lower reliability of COTS components, therefore achieving reasonable system reliability at a reduced cost.

Thanks to the features of compatibility, design reuse, integration and expandability, while keeping the low-cost and COTS approach of CubeSats, the AraMiS architecture extends its modularity in two directions:

1.  Possibility to reconfigure the system according to the needs of the payload, to target different satellite shapes and sizes (from 5 to 100 kg and even more) based on one of the following design options: Smallest cubic shape; larger cubic (or prismatic) shape; small hexagonal / octagonal satellites.
2.  Functional modularity, achieved using smart tiles (or Panel Bodies) which have, at the same time, thermo-mechanical, harness, power distribution, signal processing and communication functionalities.

Power and data handling capabilities are embedded in each tile (*Power Management tile*), which incorporates solar panels on the external side and basic power and data routing capabilities on the internal side, and can host a number of small sensors, actuators or payloads (up to 16 for each tile). An AraMiS tile is illustrated in Figure 6. Each tile offers a power and data standardized interface with mechanical support for small subsystems.
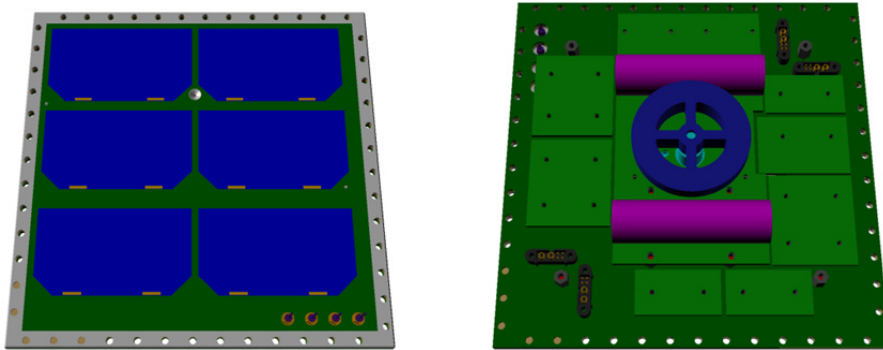


Fig. 6. Mechanical drawing of external (left) and internal (right) side of an AraMiS tile.

Figure 6 also shows about twelve standard interconnection points for small and light-weighted subsystems, among which a reaction wheel (centre) and a small power storage subsystems (batteries).

Figure 7 shows a photograph of the outer side of a tile, bearing solar cells, and a detail of the honeycomb structure of one of the possible AraMiS configurations:

Each subsystem is housed in small daughter boards which can be connected with spring-loaded connectors, like the one shown in Figure 8. Connections are electrically and mechanically modular, so they are suitable for a large range of systems, from those with just 1 or 2 analog channels up to larger systems with up to 8 analog channels, 16 digital I/O and possibly a CPU with I2C and RS232 communication.
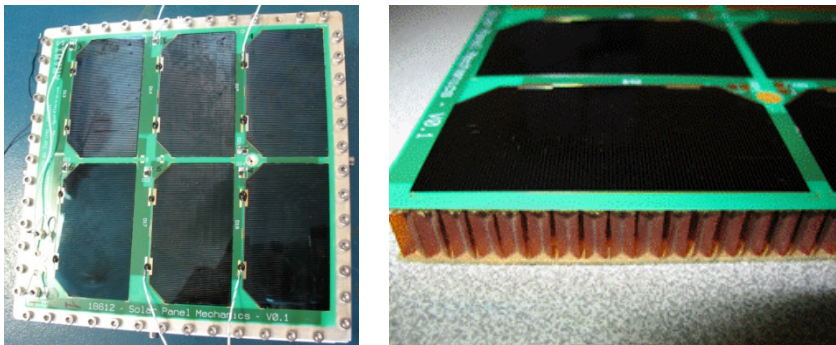


Fig. 7. Photograph of the outer side of a tile (left). Detail of honeycomb structure (right).
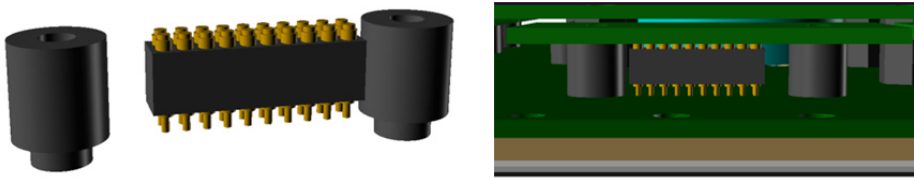
Fig. 8. Detail of a spring-loaded connector (left). Detail of modules interconnection (right).

Figure 9 shows examples of a single connector board and a double connector board respectively. The bottom side of the small modules is also shown, where the pads for spring-loaded connectors are clearly visible.



Fig. 9. Top view of single module (left); double module (center); bottom view (right).

The basic idea is that the outer tiles must also possess structural properties, i.e. that they may become an important part of mechanical subsystems. This results in reducing both weight and costs and the simplification of processes and assembly and testing times.

Therefore, the basic Power Management tile is composed of solar cells, control electronics with a dual CPU and A/D converters for handling sensors, storage, signal processing, housekeeping functions, rechargeable batteries and the Al-alloy panel which holds everything together and encloses the satellite. All of these modules are connected with a power and data distribution bus to share power and to exchange system information.

The power generation system is consequently extremely modular, since it is composed by the Power Management tile replicated as many times as needed to get the desired power. All these tiles work in parallel and offer a good level of redundancy, making the system able to tolerate several faults and allowing graceful performance degradation.

## 6. Sensor fusion and reconfigurability

An AraMiS satellite is built using several replicated modules connected together and capable of sharing information. The module in charge of attitude and orbit control is located on the power management tile and can send the measurements, acquired by its sensors and stored in a housekeeping vector, only to the on-board computer (OBC).

Having several tiles with the same structure and equipment, means that also the housekeeping vectors contain the same information. This solution offers a high degree of redundancy but also a more accurate and less error prone readout; in order to exploit the characteristics of the AraMiS architecture a sensor fusion algorithm was developed.

The sensor fusion algorithm is included in a module of the OBC which allows system level management. Its main task is to check for the status of the tile, that includes also the AOCS, in order to detect any anomalies; if some anomalous conditions are identified the software starts the self configuration.

At regular intervals, the OBC reads from each tile in the satellite the housekeeping and the configuration vectors. From the housekeeping vector analysis, the OBC obtains a view of the general status of the satellite, while the configuration vector shows the mechanical status. Then the sensor fusion algorithm analyzes these vectors in order to detect anomalies or non-working sensors. The word "analyze" is a bit vague, so from now on the discussion will be about the operations which supervise the system and reconfigure the parameters.

## 6.1 Fault detection

A way to detect faulty condition and anomalies is to do some calculation on the housekeeping vectors; in fact, comparing the telemetry information of each tile can be a very effective method to find errors, especially if the dimensions of the satellite are small. For each type of information there will be an appropriate operation to detect faulty conditions and the choice of what to do depends on the characteristics of the information.

The intrinsic subsystem redundancy of the AraMiS architecture can be effectively exploited as a fault detection mechanism. Subsystems compatible both in the measured quantity and in its reference points can be critically compared to detect anomalies and faults. Comparison on indirect measurements (e.g., on attitude parameters obtained from sun and magnetic sensors) is to be considered risky mainly due to the possible ambiguities in the comparison.

As a fault detection example, the magnetic field measured by two parallel tiles should be similar in absolute values. With the magnetic field, the arithmetic mean of the values can be an effective detector for anomalies. Also, comparing the newly acquired information and the previous one helps the detection of anomalies in slowly varying measurements like temperature.

Also, it's possible to use a range detection method. There are some kinds of information that can't exceed a certain range, like the minimum battery voltage. If the measured value exceeds the threshold it's easy to detect an anomaly.

## 6.2 Self reconfiguration

When a fault is detected, for example if a sensor stops working, the system needs to check if the parameters are still valid or become obsolete with respect to the new configuration of the system. The critical parameters to update are all those related to vector information, such as magnetic field or spin. In these cases, the parameter is a pseudo matrix so the calculation is more complex; in the following example is considered the calculation of the magnetic field, but the same process can be used for other fields.

The computation of the magnetic field vector with respect to the centre of the satellite is a very important step in order to determine the maneuvers to control attitude. All values read from magnetometers are referred to the coordinate system of their tile, which means that, in order to compute the magnetic vector in the satellite's coordinate system, the algorithm needs to apply a roto-translation to all measures finding the solution of a linear system. For

each tile are available two magnetometers, which measure the component $x$ and $y$ with respect to the coordinate system of the tile.

This operation is done using the director cosine matrix of each tile, which represents the relative position of the tile's coordinate system with respect to the coordinate system of the entire satellite. Using the first two rows of the director cosine matrix of each tile, the algorithm builds a $2T * 3$ matrix $A$, where $T$ indicate the number of the tiles with working sensors. This matrix represents the roto-translation needed to switch from the tile coordinate system to the one in the centre of the satellite. In order to find the magnetic field vector from the measures of the magnetometer, matrix $A$ needs to be pseudo inverted (the normal inversion works only on square matrix).

One of the methods that can be used for the calculation of a pseudo inverse matrix is the Singular Values Decomposition (SVD). Using this method the starting matrix $A$ can be decomposed in the matrices $U$, $V^*$ and $\Sigma$, which represent, respectively, the left-singular vectors, the right-singular vectors and the singular values of $A$. With $U$, $V^*$ and $\Sigma$, the pseudo-inverse of $A$ is equal to

$$A^+ = V\Sigma^+U^*$$

The only problem related to this method is that it's a numerical approximation one, so it needs to use floating point variables, which grant a greater dynamic range, in order to return acceptable values. But considering that the self reconfiguring algorithm is, hopefully, used only few times (or never used at all) the floating point computation is not a major concern.

## 7. Latchup protection

Single event latchup (SEL) in silicon devices is the most damaging condition to be considered during the design of space-based electronic systems. Latchup in CMOS gates is a condition where a parasitic SCR thyristor connecting the supply rails is activated by a spurious event. This creates a low-resistance path which draws an excess current, possibly leading to the destruction of the affected device (Sexton, 2003). This condition is self-sustaining.

The latchup condition can be triggered in several ways, e.g. from power supply and input signal glitches or from interaction of the substrate with ionizing particles, heavy ions, etc. (Voldman, 2007). Commonly found in the space environment, ionizing particles and heavy ions pose then a serious threat to CMOS electronic devices since the particles cannot be easily stopped from interacting with the silicon substrate. The use of commercial, non rad-hard, components in space requires then proper mitigating measures to avoid damages and recover proper operation of the affected systems.

The hold voltage for self-sustainment of thyristor activation is on the order of 1 V. To extinguish the latchup, the supply voltage has to be reduced below this threshold, i.e., by means of a simple power cycle of the affected ICs. At system level, a trade-off needs to be established since the protection of every CMOS device cannot satisfy the low-complexity requirements. System partitioning is needed to isolate vulnerable sections that will be protected by a limited number of adjustable protection modules.

The latchup is detected when an excess current draw is measured in a circuit section and the power supply of multiple devices will be disconnected at once. Before power supply

re-activation however, stored energy within the section circuitry (e.g., in bypass capacitors) needs to be depleted through an explicit short circuiting of the interested supply rails. This ensures that discharge transients complete and latchups are extinguished before the timed re-activation of the power supply.

During the system partitioning, also the power supply requirements of the protected devices have to be taken into account. On one hand, impulsive loads may be erroneously detected as latchup conditions, causing periodic and rather systematic reboots in the protected section. On the other hand, some components may be rather sensitive to the high current draw of a real latchup and may need a quicker/more sensitive setting on the current consumption threshold. Also, a sensitive component should not be grouped in a high power consumption section since its latchup condition may become difficult to detect. The different devices should then be grouped in sections based on the expected power consumption profile. Each section will be overseen by a single protection module and its intervention parameters will have to be tuned accordingly.

The central part of the latchup protection subsystem can then be isolated in a replicated latchup protection module. The module needs to be both radiation tolerant (to be immune from the problems affecting the protected circuits), flexible (to adapt to different power consumption profiles) and simple (in terms of basic functionalities, circuit complexity, occupied board space and cost). We have then decided to develop the **1B127** (1B127 Datasheet, 2009), an appropriate ad-hoc anti-latchup and over current protection device manufactured using Neohm's hybrid technology.

For the protection circuit to be radiation tolerant and not being affected by SELs and total dose induced phenomena, a proper component selection is needed. The electronic components will have to be readily available and based on a latchup-free technology. The obvious choice is a bipolar process that may also be dielectrically isolated for improved SEL tolerance. The use of dielectric trenches and oxide isolation layers, allows for a complete isolation of the single transistors, avoiding resistive paths and parasitic junctions and capacitances (National Semiconductor, 2000). Available at the NSRE Components Database (IEEE Radiation Effects Data Workshop, 2010) is also a list of radiation tolerances of selected devices that may provide valuable information. The accurate selection of commercial devices will contain costs while providing a design stage guarantee over the expected radiation tolerance of the subsystem. This early guarantee will have to be complemented by real world radiation exposure to reach the target tolerance levels.

The flexibility in adapting the protection circuit to the needs of the different sections is centered on a limited number of parameters. The *turn-off* time controls the time needed for the protection to trigger, allowing control of the circuit tolerance to spikes and inrush currents. The *turn-on* time controls the time needed for the output of the module to reach (almost linearly due to an intrinsic output current limitation) the nominal value after supply re-activation; the slow ramp limits the inrush current of the protected load avoiding instability. The *low-pass filtering* of the current reading allows masking high frequency noise that may trigger the protection and contributes, with the *turn-off* time, to protection stability.

An excessive level of flexibility however may hinder circuit adoption within the system or module re-use in future applications. The need of external components adds to the used board space, requires additional design and manufacturing time, and increases cost. The

module is then designed to work with a minimal configuration needing, essentially, only an external sense resistor. The parameters are internally tuned to achieve optimal performances in average subsystem conditions.

The module block diagram is shown in figure 10. A high-side current sense is implemented with the use of a specialized *current sense* (or *shunt*) *amplifier*. This solution avoids low-side sense resistors that would introduce interruptions and extraneous resistances on ground planes. The specialized amplifier helps avoiding common mode rejection issues typical of differential amplifiers used in high-side applications. A precision voltage reference allows for predictable triggering of the protection over an extended temperature and supply voltage range. The comparison result is then fed into a monostable circuit controlling the delay before load re-activation and a slew rate controller that introduces the necessary asymmetry to differentiate *turn-off* and *turn-on* times. Two MOS transistors are finally used (exclusively) to connect the load rails to the supply or to short circuit them to extinguish latchups during protection activation. Additionally, the protection module outputs the current reading for telemetry purposes and allows for the use of external higher power transistors to switch high current loads.
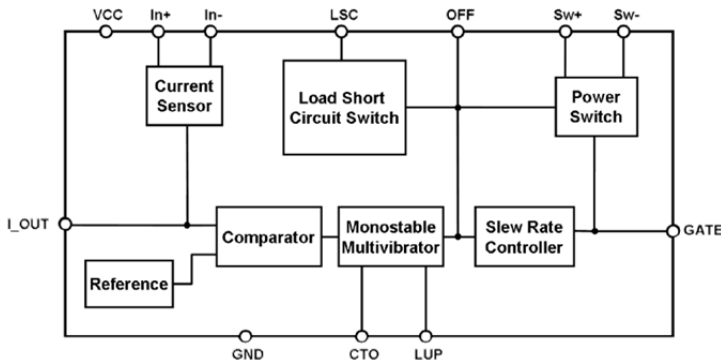


Fig. 10. Internal block diagram of the latchup protection module.

The intrinsic flexibility of this approach also allows for more advanced approaches like the one depicted in figure 11, where two modules are used in parallel to provide some degree of module fault tolerance.

## 8. Open hardware/software plug and play architecture

The modularity at the hardware level which has been described previously requires an open software architecture, in order to really be effective.

We originally aimed at having a kind of **plug and play** approach, similar to that available in many other terrestrial applications and also in the Space Plug and Play (SPA, 2011) but we soon pointed out that this was posing excessive constraints on the capabilities of the supporting CPU and its tolerance to radiations. It simply could not be afforded by a low cost, high reliability approach. Furthermore, a real-time, full plug and play solution was not required, as the flexibility of plug and play was only needed during the integration phase and not later.
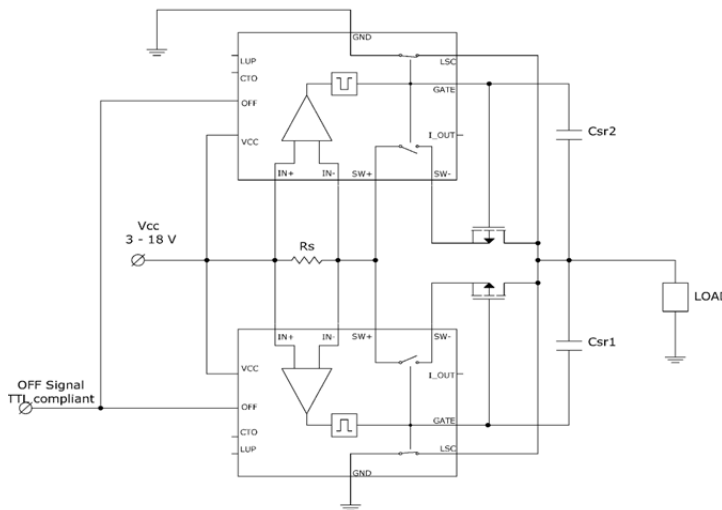
Fig. 11. Advanced protection stage with redundancy of protection modules and external transistors. A simple system can use just one device plus an external resistor.

We therefore decided to develop an alternative approach, nearly as flexible as our original aim, but which requires significantly lower CPU capabilities and power and which could offer better tolerance to radiation. This solution can be run on the MSP430 family of processors, a standard COTS from Texas Instruments.

We found that this capability was straightforward if a proper design approach was used. We therefore developed an appropriate system-level approach based on the extensive use of UML and all its capabilities, strongly supported by the Visual Paradigm VP Suite tool.

With the proposed approach, every subsystem is composed of: i) a HW part (namely, each of the small module boards described in section 5 and shown in figure 9); ii) a corresponding SW support, properly hardened using the techniques which are discussed in section 9; iii) an appropriate interface to the sensor fusion and reconfiguration algorithms described in section 6 and the AOCS algorithms which are shown in figure 2.

An **UML class** is associated with each of our subsystems, as shown in figure 2 (overview), while figure 12 shows a detailed (although simplified) view of one of those subsystems. From that, we can see a few important items:

- The internal hierarchical architecture of the module: the "root" class (the left box) is composed of three major blocks (the three boxes at the right); in turn, each block is made of smaller and simpler blocks (not shown);
- The electromechanical parameters of the module, which are described by appropriate **class attributes**, properly documented (documentation not shown). For instance, attribute **float SENS_MAGNETIC_FIELD_RAW** indicates the sensitivity (in V/T) of the pin measuring magnetic field. Its value is given by a formula **1.0 / (sensor. SENS_MAGNETIC * adc.SENS_ADC)**, and therefore it is given in term of the subsystem parameters (including for instance also the sensitivity of the ADC which is inherited by

the hosting CPU). Any change of its subsystem (or, similarly, any element of its subsystems) or any change in the ADC is automatically inherited by the overall system and, from here, to the sensor fusion algorithm, which requires this value to properly interpret values received from the module. This inheritance is obtained at compile time, that is, during HW/SW system integration: when the integrator chooses to assemble a given module, he will bring the whole UML class inside the UML model of the spacecraft under construction and, from here, the correct SW is automatically generated, both for the tile CPUs and for the OBC supporting the sensor fusion and AOCS algorithms.

- Some template parameters (included in the dashed box above the root class), which allow to configure, for instance, in which physical slot in each tile the module is plugged (see figure 6) and in which positions of the overall housekeeping, status and control vectors the relevant data of the module shall be stored by the tile CPU.

This is a form of UML-based Object Oriented Design (OOD) approach which allows building either a spacecraft or, in this case, one of its subsystems according to specific mission requirements, with very limited risks, effort and design time.
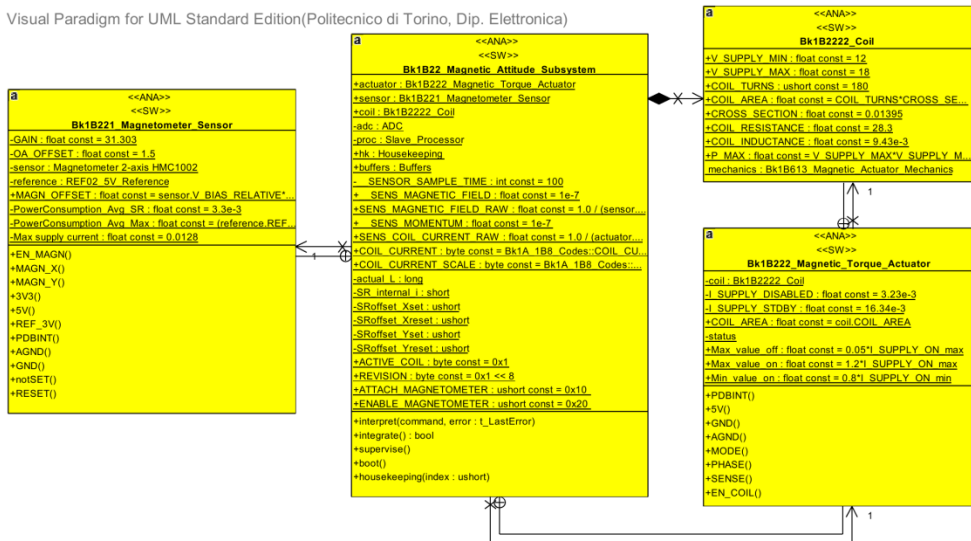


Fig. 12. A detailed (although not complete) UML view of the architecture and the parameters of our 1B22 Magnetic Attitude Subsystem.

## 9. Hardening techniques against radiation of SW code and HW interfaces

This section presents a general methodology depicting how to address the overall protection of code to mitigate transient faults induced by radiations. The code is supposed to be written in either C/C++ or assembly language. COTS devices running this code are supposed to suffer from all possible Single Event Effects (SEE): single event upsets (SEU), both single (SBU) and multiple (MBU), single event functional interrupts (SEFI) and possibly others. They are also supposed to: survive a certain desired total ionization dose

(TID) present in its operational environment (given orbit and mission lifetime) and be either latchup-free or protected from disruptive radiation-induced effects as shown in section 7.

Having in mind the mitigation of the harmful effect of soft errors, we have determined a number of use cases to design an effective radiation hardening of a COTS processor-based system. Although no hardware redundancy can be applied within the processor, preventing us from applying a Triple Modular Redundancy (TMR) over a set of critical registers from the register file, we can think of a rich set of software-side fault-tolerant strategies. Moreover, exploring the software hardening design space reveals not only the importance to take into account the type and physical location of code/data, but also the need to take into consideration the hardening of the interface between processor and outer elements such as ports, DACs, etc.

We describe the following software hardening targets and the selected hardening action:

### 9.1 Data storage

With independence of its physical location, data can be labeled as having an infinite, medium or short lifetime:

- **Infinite lifetime**, if lifetime of data is much higher than most other variables in the system. In this case, the proposed hardening method is performing periodic refreshing (compatible with system time constraints) using software DMR or TMR (Double/Triple Module Redundancy) when applicable.
- On the other hand, having a **medium** or **short lifetime**, that is, a lifetime comparable or shorter than storage time of most variables in the system, led us to propose DMR or TMR carrying out its action (fault detection or correction) every time data is accessed.

### 9.2 Data driven routines

This use case consists of programs without state automata, whose execution flow does not depend on past events and system states. A data-driven routine is supposed to enter, get input data (either from the calling program, input devices or data storage), execute in a predefined time and with a predefined algorithm, outputs results (either to the calling program, the output devices or data storage) and either exit or repeat execution endlessly.

A data-driven routine can either be **flat** (that is, without internal calls to other routines) or **hierarchical** (that is, with internal calls to subroutines). This characterization implies two different hardening processes:

- **Flat code**: A data-driven flat code is defined as a software routine which either has no input and output data (so all data are and remain inside the routine itself, as for example a delay function), or input and output data come from or go to the calling routine, but they need not be hardened. The code can either have or not have calls to internal routines or accesses to external devices which either requires no parameter passing or parameter passing need not be secured. We have evaluated two possible hardening strategies. At low level (i.e., at assembler level) we propose using the Software Hardening Environment (SHE) as described in (Antonio Martínez-Álvarez et al., 2012). SHE is a proven compiler-directed soft error mitigation system especially designed to harden a certain software code against radiation-induced faults. SHE makes a special emphasis on flexibility and selectivity, that is, different hardening techniques

can be carried out on different subsets microprocessor resources. At a higher level, for example, hardening a C++ code, we propose using a library with hardened-types. This library must declare dual hardened types for every C++ native type such as char, int, and so on. So we can choose at C++ level either using char or Hchar, the hardened version for char. Each H-type ensures redundancy (based on TMR) by overloading every possible operation on it.

- **Hierarchical code**: a software routine which requires securing the exchange of data from one element to another. All data transfers must be hardened. We have focused our attention to the process of hardening parameter passing.

  Different hardening sub-cases can be described depending of the nature of sender/receptor:

- **Hardware to hardware**: data is passing from a hardware device to another hardware device, usually via one or more wires (bus). If the bus handles critical information, bus-TMR can be applied; otherwise a higher level software solution (like using a hardened data packet exchange protocol) is a good solution where time overhead doesn't exceed design constraints.

- **Hardware to software**: data is passing from a hardware device to a software routine. It usually implies a read operation by the code from a microprocessor port or an internal device register. It does not include interrupts.

- **Software to hardware**: data is passing from a software routine to a hardware device. It usually implies a write operation by the code to a microprocessor port or an internal device register.

- **Software to software**: hardens parameter passing between two software routines. This case encloses hardening of different lifetime variables, data memory and register file.

The last three sub cases are supported by the SHE tool so we propose its use to ensure predictable fault coverage.

## 9.3 Program flow

The hardening against Control Flow Errors (CFE) is a mandatory hardening task. Without the possibility of a hardware hardening of special involved flags nor registers (e.g. carry, zero, overflow flags presented in conditional jump/calls), we take into consideration two possible alternatives. The first one consists of the classic use of a watchdog (see 9.6 harden code memory). The later one consists of **control-flow checking by software signatures** (Oh et al., 2002b). This technique is based on adding compiled-time calculated signatures on every node from Control-Flow Graph (CFG). These signatures will be checked during execution of code. The best choice depends on the time-constraints of the system (e.g. keep time overhead less or equal than a certain limit).

## 9.4 Control driven programs

We find here programs with state automata, whose execution flow usually depend on past events and a number of system states. The most sensible parts within the hardening of these programs have to do with the hardening of parameter passing as described in 9.2 (**Hierarchical Code**) and the hardening of the state storage, which is also described in 9.1 (**Data Storage**).

## 9.5 Precompiled libraries

Although we have supposed the software to be given as a C/C++ or assembler code (allowing us to perform code transformations), it is worth assessing other possibilities. A binary code (statically linked or not) usually makes a number of subroutines calls that have to be under control. In the assumption that there is no operating system, the usual library calls are related to the compiler execution runtime (e.g. GNU-GCC **libgcc** support library), the classical C/C++ runtime libraries (e.g. GNU-GCC **libc/libstd++** support libraries) or the user libraries. Our primary choice to ensure a correct hardening of a precompiled library is writing in C/C++/assembler the subset of used calls and proceeding with its hardening as described in 9.2.

## 9.6 System configuration

In this use case, we are interested in hardening the processor configuration registers (clock configuration registers, peripheral configuration registers and interrupt configuration registers) and code memory. Our research group has already developed an FPGA-based **smart watchdog** able to assist the hardening of the next three cases of system configuration of a COTS system:

- **Harden code memory**. This code storage may contains either one or more software programs (for CPUs), as well as one or more fuse maps (or configuration files for FPGAs). Code memory has to be protected more than any other storage in the system, as any corruption forever affects the whole system functionality. A SEU in code memory always causes a SEFI. There are different types of code memory, and consequently three ways to harden them. In particular:
    - **Radiation-tolerant ROMs**, which can never be affected by SEUs. For instance, true ROMs and PROMs and several types of FLASH memories. In this subcase, no hardening tasks are needed.
    - **Radiation-tolerant ROM copied into radiation-sensitive RAM**, to speed up its execution. The master copy of the program is stored in a radiation-tolerant storage but the working copy is subject to SEU. Detecting this situation always requires an external agent (like our smart watchdog), since the code alone is not able to detect it in all cases. To correct this situation a reboot action is usually enough, as reboot reloads code RAM from code ROM
    - **Rewritable radiation-tolerant ROM**: there are situations where the code resides in a radiation-tolerant FLASH but the CPU has the capability to write onto it. Even if the program does not foresee rewriting the FLASH, a SEU-induced error might unexpectedly rewrite and corrupt the code memory. As in the previous situation, detecting this situation always requires and external agent (like our smart watchdog), while its correction requires and external agent able to reload the FLASH from a backup copy from an external radiation-tolerant ROM through an appropriate bootloader interface.
- **Harden static configuration**. Most systems configure their peripherals or I/O systems by means of appropriate configuration words. For instance, baud rate and modulation for UARTS; counting mode and period for timers/counters; acquisition modes for ADCs and DACs; pin direction for I/O ports. It requires an external agent like our smart watchdog.

- **Harden Dynamic configuration**. Hardening those processor or peripheral configuration registers which are occasionally modified during program execution. Depending on the modification rate, it can be necessary an external agent (smart watchdog) or a periodic refresh similar to the one described in 9.1.

### 9.7 Interrupt Service Routines (ISRs)

Hardening of interruptions masks is covered by hardening of system configuration (see 9.6) whereas the code itself is supposed to be a data driven routine which are covered in section 9.2. In relation to the expected calling rate, we can distinguish between periodic (non re-entrant) and occasionally called ISRs, which is also covered in section 9.6 (Harden static/dymanic configuration).

## 10. Micropropulsor

The most appropriate micropropulsor for a nano or micro satellite as AraMiS turns out to be an electrospray electric propulsor, such as that described by (Lozano and Courtney, 2010).

The propellant, which is an ionic liquid, is confined in a tank of porous metal, and by capillarity forces it can be driven into a pseudo-conical structure called emitter. Here the ions are accelerated by a strong electric field established among the emitter itself and a pair of electrodes, called extractor electrode and accelerator electrode. The global potential difference varies in the range of 1 – 2 kV. The thrust produced by a single emitter is too low to meet mission requirements, and therefore the emitters are grouped into arrays. Each propulsor houses a couple of arrays that emits ions of opposite species, to preserve the electrical neutrality of the satellite; in addition, the polarity of emitted ions alternates in time, at a frequency of about 1 Hz, to avoid accumulation of ions on the outer surface of the propulsor.

A set of achievable maneuvers are discussed below, considering propulsor performance similar to those reported by (Lozano and Courtney, 2010): a specific impulse of 3500 s and a thrust of 100 μN. A cubic configuration is adopted, with one tile per face and two propulsors per tile; to remain in the category of nano satellites, the mass varies between 1 kg and 10 kg. Two orbital maneuvers in LEO circular orbits (altitude between 120 km and 1000 km) are considered here: a deorbiting and a change of inclination (both with unchanged remaining orbital parameters), plus an attitude maneuver: a complete rotation about a body axis, in open loop. For each maneuver, propellant consumption and maneuver time are calculated. Other maneuvers have been considered although not reported here.

### 10.1 Orbital maneuvers

The used model involves ideal Keplerian orbits, i.e., it considers only the gravitational forces acting between the satellite and the Earth. The model refers to the so-called Edelbaum solution (Edelbaum, 1961), which provides the minimum velocity change $\Delta V$ to apply to the satellite to modify its altitude, and/or to provoke a change $i$ in its orbital inclination, between the initial orbit, marked by subscript $o$, and the final orbit. The orbital radius r (and therefore the altitude) is strictly related, in case of ideal circular orbits, to the orbital speed V by the relation $V = \sqrt{\frac{GM_T}{r}}$, where $G$ is the gravitational constant and $M_T$ the Earth mass. From Edelbaum solution follows:

$$\Delta V = \sqrt{V_0^2 - 2VV_0 \cos\left(\frac{\pi}{2}i\right) + V^2} \tag{7}$$

the consumption of fuel $\Delta m$ is obtained from $\Delta V$ through Tsiolkovsky's equation:

$$\Delta m = m_f \left(e^{\frac{\Delta V}{c}} - 1\right) \tag{8}$$

where $c$ is the speed of the expelled propellant and $m_f$ the mass of the satellite. The maneuver time $\Delta t$ is obtained thanks to the second law of dynamics:

$$\Delta t = \Delta V \frac{m_f}{T} \tag{9}$$

where $T$ is the overall thrust of two propulsors. The model therefore does not take into account the following phenomena:

- Gravitational interaction with other bodies;
- Other forces of different nature, such as aerodynamic drag;
- Attitude maneuvers to keep the thrust vector in the desired optimum direction, as the engines have no chance of movement when installed on an AraMiS tile.

The results for the deorbiting maneuver are collected in figures 13 and 14 (left graphs) and also apply in the case of an orbital climb. An increase in the final mass of the satellite is detrimental to maneuver characteristics, as it would imply that more mass has to be accelerated. This maneuver is regarded as practically feasible thanks to the results of the model.

The change of inclination (figures 13 and 14, bottom) is limited to 2 rad ≈ 114.59°, due to limitations of the Edelbaum solution; in a precautionary way a minimum altitude of 120 km is used, as it implies the maximum $\Delta V$, and therefore the worst characteristics. This maneuver would be much more challenging than the previous for the propulsion subsystem.

## 10.2 Attitude maneuvers

The open-loop rotation takes place in three phases:

1. *Angular acceleration*. A pair of engines, on opposite tiles, generates a pure moment on the satellite, up to a maximum angular speed;
2. *Drift*. The propulsors are turned off, and the satellite moves with uniform circular motion at the maximum angular velocity;
3. *Angular deceleration*. A torque is generated in the opposite direction, to allow the stopping of the satellite motion. Maneuver characteristics are the same of phase 1.

Knowing the geometric and inertial features of the satellite, the characteristics of the maneuver are calculated with simple considerations on the kinematics of circular motion. In this model only propulsive forces are considered. The results are described in figures 13 and 14 (right graphs). The graphs are restricted to a maximum angular speed of 0.19 rad/s, dictated by AraMiS features and by the considered thrust. As previously explained, an increase in mass worsens maneuver characteristics. The maneuver appears to be much more sustainable by the propulsive subsystem than the previous, even from a time extension point of view.
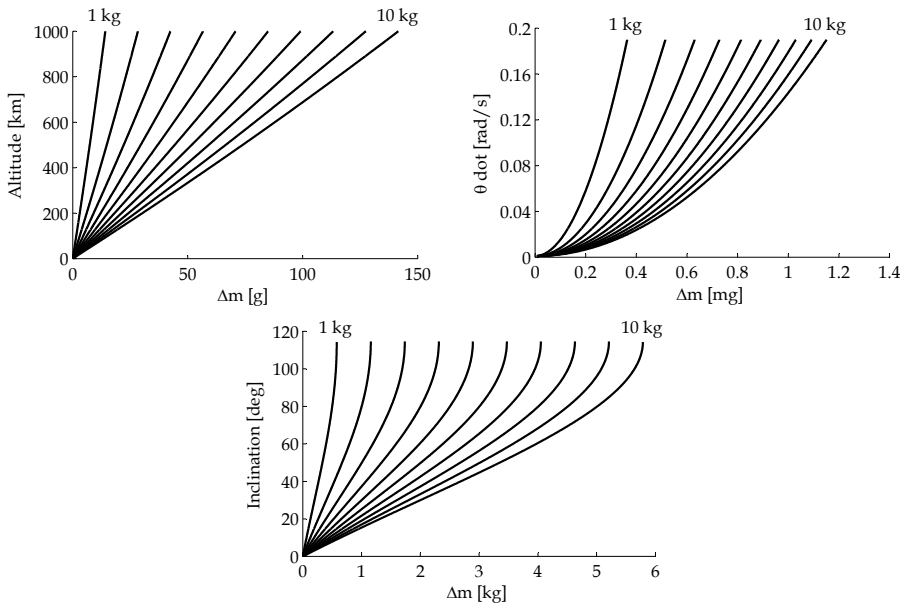
Fig. 13. Fuel consumption of chosen micropropulsor for three different attitude and orbit maneuvers: deorbiting (left); spin (right); orbital plane inclination (bottom). All plots are for satellite masses from 1 kg to 10 kg.
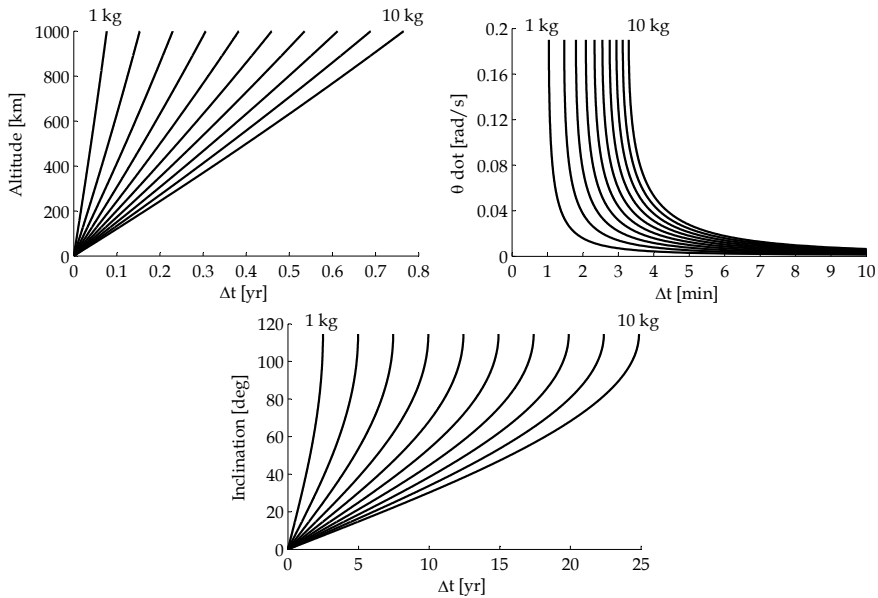
Fig. 14. Maneuver time of chosen micropropulsor for three different attitude and orbit maneuvers: deorbiting (left); spin (right); orbital plane inclination (bottom). All plots are for satellite masses from 1 kg to 10 kg.

Propellant consumption for deorbiting is of the order of tens of grams; therefore compatible with small satellites characteristics and maneuver time, even in the worst case of 10 kg of final mass, is less than one year, while fuel consumption and maneuver time for large changes of orbital inclination are too high for small spacecrafts and typical missions. Formation flight is still under consideration, while attitude maneuvers require little propellant, unless used frequently, therefore they can be occasionally used in support of the magnetic and inertial approaches.

## 11. Conclusion

This chapter has described a few aspects of the AOCS subsystem for small satellites which has been developed at Politecnico di Torino as a part of its AraMiS modular architecture for small satellites.

Several innovative aspects of the development of a low cost, high performance system have been considered in details, more than the complete description of the system, as these are the most technologically challenging aspects of the development and can find many applications in other low cost spaceborne system. In particular, the AOCS subsystem itself and all the innovative technological aspects described above can be used in other similar applications, like inside CubeSats or other small satellites.

## 12. References

Reyneri, L.M.; Sansoe, C.; Passerone, C.; Speretta, S.; Tranchero, M.; Borri, M.; Del Corso, D. (2010), Modular Design Solutions For Modular Satellite Architectures, *Book Chapter in Aerospace Technologies Advancements*, InTech Editor, 2010, pp. 165-188.

Hurtado, J.; Reyneri, L.; Saenz-Otero, A. (2011) CubeSat & NanoSat image simulator for a low cost model position and attitude sensor, *Proceedings of IEEE LARC & CCAC 2011*. October 2011, Bogota, Colombia.

Wertz, J.; Larson, W. (1999), Attitude Determination and Control, *Section 11.1 of Space Mission Analysis and Design*, 3rd edition, Space Technology Library, editor.

Sexton, F.W. (2003) Destructive single-event effects in semiconductor devices and ICs. *IEEE Transactions on Nuclear Science*, vol.50, no.3, pp. 603- 621, DOI 10.1109/TNS.2003.813137, June 2003.

Voldman, S.H. (2007) Latchup. John Wiley & Sons.

National Semiconductor (2000). National Semiconductor Develops New Complementary Bipolar Process.
<http://portal.national.com/nationaledge/aug01/860.html> (retrieved November 2nd, 2011).

IEEE. Radiation Effects Data Workshop (REDW) Record, 2010. <http://nsrec.com/redw/> (retrieved November 2nd, 2011).

De Los Rios, J.C.; Roascio, D.; Reyneri, L.M.; Sansoè, C.; Passerone, C.; Del Corso, D.; Bruno, M.; Hernandez, A.; Vallan, A (2011). ARAMIS: A fine-grained modular architecture for reconfigurable space missions. *Proceedings of 1st Conference on University Satellite Missions.* Rome, Italy.

Speretta, S.; Reyneri, L.M.; Sansoè, C.; Tranchero, M.; Passerone, C.; Del Corso, D. (2007). Modular Architecture for Satellites. *Proceedings of 58th International Astronautical Congress*. Hyderabad, India.

Del Corso, D.; Passerone, C.; Reyneri, L.M.; Sansoè, C.; Borri, M.; Speretta, S.; Tranchero, M. (2007). Architecture of a Small Low-Cost Satellite. *Proceedings of 10th Euromicro Conference on Digital System Design*, 428–431.

Schilling, M.A (2000). Towards a general modular systems theory and its application to inter-firm product modularity. Academy of Management Review, Vol 25: 312–334.

Baldwin, C. Y. & Clark, K. B (2000). Design rules, Volume 1: The power of modularity, Cambridge, MA: MIT Press.

Carliss Young Baldwin, Kim B Clark (2000). Design Rules: The power of modularity. *MIT Press*, 63-64.

Ron Sanchez, Joseph T. Mahoney (1996). Modularity, Flexibility, and Knowledge Management in Product and Organization Design. *Strategic Management Journal*, Vol 17: 63-76.

Orton, J. & Weick, K (1990). Loosely coupled systems: A reconceptualization. Academy of Management Review, 15: 203–223.

Loose Coupling. [Online]. Available: http://searchnetworking.techtarget.com/definition/loose-coupling

Lozano, P. & Courtney, D. (2010), On the Development of High Specific Impulse Electric Propulsion Thrusters for Small Satellites, Madeira, Portugal, June 2010

Edelbaum, T.N. (1961), Propulsion Requirements for Controllable Satellites, *ARS Journal*, August 1961

(Oh et al., 2002b) Oh, N.; Shirvani, P.; and McCluskey, E. (2002b). Control-flow checking by software signatures. *IEEE Transactions on Reliability*, 51(1):111-122.

Cuenca-Asensi, S.; Restrepo-Calle, F.; Palomo, F.R.; Guzmán-Miranda, H.; Aguirre, M.; (2011) Compiler-Directed Soft Error Mitigation for Embedded Systems, *IEEE Transactions on Dependable and Secure Computing*, 2011.

(1B127 Datasheet, 2009), 1B127 Datasheet, Neohm Components

(SPA, 2011), Space Plug-and-Play Architecture (SPA) Standard, System Capabilities, AIAA G-133-10-201X, Draft for Public Review, American Institute of Aeronautics and Astronautics 1801 Alexander Bell Drive, Reston, VA 20191.