ISTNanosat-1 Heart
Processing and Digital Communications unit

João André Henriques Ferreira
joaoahferreira@ist.utl.pt

Under supervision of:
Prof. Doutor Rui M. Rocha
Prof. Doutor Moisés Simões Piedade

Instituto Superior Técnico / TULisbon
October 2012

Abstract—The ISTNanosat-1 is a double CubeSat which is being developed by students and teachers from IST/TULisbon. The Heart unit, whose software was developed in this dissertation, presents a solution to manage the remote interactions with the Ground Station through the space-link. This unit is also responsible for satellite housekeeping. The solution comprises two different subsystems: Digital communications and Command & Data Handling.

The digital communication subsystem implements the required network functionalities e.g. allowing a ground operator to invoke remote commands in a reliable way onto a specific subsystem or in the satellite as a whole. This subsystem also provides the satellite general health status information (telemetry) periodically in the downlink taking into account maximum Ground Stations compatibility. Since the ISTNanosat will carry a tiny camera aboard, the Communications solution also implements a transport protocol that enable the imagery transmission from the spacecraft, abstracting all the details on this process. All the network functionalities were implemented taking into account the intrinsic characteristics typically found on Low Earth Orbit space-links such as: intermittent and disruptive connections; low and very asymmetric throughputs.

The Command & Data Handling is a critical subsystem due to its responsibilities as the on-board information orchestrator. To enhance its operation correctness it relies on a Real-Time Operating System to implement the satellite housekeeping and internal communications tasks when it enters in the safe-mode profile.

Finally, the Heart unit solution was deployed using a AT91RM9200 and the moteISTs5++ as prototype platforms. The final solution was submitted to a set of performance and quality tests highlighting the solution compact design, low power consumption and ARM processor under-utilization even in stressful cases.

Index Terms—ISTNanosat, Cubesat, embedded systems, space communications, redundant subsystems

1 Introduction

SPACE exploration is always related with large investments from governments and/or private institutions. This represents a limitation to technological advances, since the development is confined to some entities. In mid-1960 the Orbiting Satellite Carrying Amateur Radio (OSCAR) group was created, having as a major goal the construction and launch of amateur satellites. Two years later, the first amateur satellite called OSCAR I was launched. In 1969, the OSCAR project merged with *COMSAT Amateur Radio Club*, forming the *Radio Amateur Satellite Corporation* - also known as AMSAT - thus enabling the OSCAR 5 launch [1]. In 1981 the UO-9 - UoSAT-OSCAR 9 - or *University of Surrey's UoSAT-1* was launched. This satellite marked the beginning of a new era in which universities are the main sources of funding of these amateur projects [1] [2]. Since then, dozens of projects have been developed within the academic community. It is estimated that, on average, 12 satellites are launched by universities per year [2].

In 1999 the CubeSat project was started. This collaborative project, initially between the *California Polytechnic State University* (Cal Poly) and the *Standford University's* Space Systems Development Laboratory (SSDL), aims at the standardization of pico-satellites design. This standardization increases the space accessibility by allowing cost reduction, development time decrease, and keeping frequent launches. A Cubesat is a cube with 10cm - (10x10x10cm) with up to 1.33kg - or 1U [3]. These cubes, can be grouped easily in order to form larger satellites, for example 2U (10x10x20cm) or 3U (10x10x30cm). It is estimated that in 2010, 250 CubeSats were built in 1U, 2U and 3U formats [4]. The satellites developed under this specification - CubeSat Design Specification (CDS) - in addition to carry all necessary technology to its orbit operation, can be designed to transport more specific scientific experiments. One clear example is the NASA GeneSat-1, which has a bacteria miniature laboratory inside with the capacity of detecting proteins - products of specific genetic activity [5]. CubeSats are commonly delivered in LEO[1], defined as 160-2000km above the Earth's surface [6].

Apart from the CubeSat community developments, other working groups have been formed in order to solve more generic problems. For example, to address the problems associated with long distance communications, keeping in mind the lack of efficiency of terrestrial protocols in such scenarios, new communication paradigms were forced to emerge; a very significant example is the Delay Tolerant Network (DTN). This new communication approach, has been catapulted by the *Delay Tolerant Networking Research Group*[2]. The DTN proposed architecture is an evolution from the initial proposed (InterPlanetary Network) IPN Internet architecture [7]. "The IPN is a member of a family of emerging Delay Tolerant Networks" [8]. The *Internet Society IPN Special Interest Group*[3] is responsible for IPN developments. The IPN is already included in the *NASA Mars mission program* [9].

In 2010, the ISTNanosat-1 project was born, presenting itself as a candidate to be the first Portuguese satellite entirely made inside an academic context. In this project, minimum use of Commercial Off-The-Shelf (COTS) components is planned or, in other words, preference will be given to academic developed technology. This nano-satellite[4] will be built under CubeSat specifications in a 2U structure. Some technology has already been engineered by researchers and students from IST. In this context, there arises the need for an orchestrator which is designated as Heart unit.

The Heart unit is responsible for for both digital communication processing and central decision logic orchestration. The whole solution encompasses two different subsystems, the Command and Data Handling (C&DH) and the Digital Communications (COM). The C&DH subsystem is responsible for processing data gathered from on-board components, such as magnetometers, sun sensors, gyroscopes etc., while the COM subsystem is responsible for digital communications through the space-link - connecting the satellite with the Ground Station (GS).

Taking into account the large budget involved in a spatial project, it is necessary to develop very robust and reliable systems, in order to avoid jeopardizing the entire investment due to a particular component fault. With this requirement in mind, the Heart module architecture needs to be redundant. The entire solution, needs to be

1. Low Earth Orbit
2. http://www.dtnrg.org accessed on 15-09-2012
3. http://www.ipnsig.org accessed on 15-09-2012
4. Satellite with a wet mass between 1 and 10 kg

adaptable enough to allow different faulty scenarios, in order to keep the general satellite performance as good as possible, avoiding that specific faults can turn into global failures. Beyond the imposed constraints to this space navigation device, it is important to endue the communications with tolerance to delay, disruptions, lack of bi-directionality or highly asymmetric links, etc..

In this article we start by detailing the conceptual view of our Heart unit architecture. Next, we address the issues related with hardware and software implementations followed by the discussion on the results obtained through experimental evaluation. We close by drawing some conclusions.

2 Heart Design

ISTNanosat-1 Heart is the satellite central intelligence unit. It is responsible for onboard data processing and handling the communications with Earth Ground Stations. It may be able to run the necessary procedures for satellite housekeeping and other functional tasks. These procedures can be, for example, image gathering, collect internal system performance information, or inject well formatted telemetry information into the radio link.

The Heart unit architecture comprises two processor boards and the corresponding software running on top of them. The COM subsystem is responsible for processing all in/output space-link digital data and to route some information to another subsystems like the C&DH. The C&DH subsystem is responsible for satellite housekeeping tasks and for the safety-critical operational decisions as well as the remaining subsystems management. This subsystem must have an ultra low power consumption profile to allow basic satellite operation when facing unfavourable scenarios.

Both subsystems are connected to the Analog COM subsystem which is responsible for the following functions: Data Framer, which implements a basic network stack in hardware that conceals from the remaining subsystems the intrinsic protocol mechanisms like handshakes, signalling, retransmissions, etc.; and the Modem. The C&DH subsystem being less powerful is connected to the Data Framer avoiding network stack processing, when the satellite enters in the safe mode. The satellite runs in this profile when the spacecraft suffers some problem, or simply when it needs to save more energy. Also, when such scenario occurs, the COM subsystem is disconnected. When the satellite operates with no problems and enough energy, it works in the normal-mode profile, where both subsystems are connected. Within this profile, the COM is directly connected to the Modem, implementing by its own the network functionalities and the C&DH can process its housekeeping tasks without worry about space-link communications.

2.0.1 Digital communication subsystem

The first design issue regarding this subsystem concerns the necessary processing power to support the demanding functionalities, such as receive the GS commands and ensure its reliability, send the telemetry beacon and allow the imagery transmission to Earth. These functionalities require interaction with different types of interfaces and processing space-link network protocols. Fortunately, all these functionalities share a common behaviour: a heavy but sporadic energy consumption. From all the available power allocated to the Heart unit (950mW), the COM subsystem will have $\sim 80\%$ of it. This 80% is a rough estimation based on the foreseen high demanding requirements in terms of processing power, e.g. space-link network stack processing and remote command execution. This power budget allows the utilization of a general purpose MCU. Such MCU permits software flexibility in terms of code reutilization, e.g. from open-source community, and a more wide spectrum choice of possible OS types and operational philosophies. The selected CPU architecture was a 32-bit ARM due to its power efficiency characteristics and global support from many embedded systems vendors. The ARM also incorporates a MMU, which allows the use of popular OS such as Linux.

2.0.2 Command and Data Handling subsystem

The C&DH subsystem is responsible for satellite housekeeping tasks and for the safety-critical operational decisions as well as the remaining subsystems management. This subsystem must have an ultra low power consumption profile to allow basic satellite operation when facing unfavourable scenarios. For example, if Electrical Power Subsystem (EPS) batteries are defective or long eclipse periods occur, the C&DH can keep the basic satellite functions: maintain attitude and telemetry transmission and simple remote command reception. The C&DH gets the remain Heart unit power budget, roughly 15% (150mW). To meet this requirement, the ultra low power TI MSP430 architecture was selected. This MCU, one of the best options when low power constraints are envisaged, allows operation in different Low Power Modes (LPM) which brings a useful way to employ energy-aware software solutions.

2.0.3 Communication protocols

Taking into account the communications requirements, specially those concerned with the need for a reliable remote command operation, the possibility to transmit images from the satellite and the telemetry service, some conceptual decisions were made in the design of the communication protocols.

Fig. 1 describes the protocol architecture involved in the space-link. The beacon service directly uses the AX.25 layer 2 protocol [10], because it is a very simple service. The reliable remote command service uses the Cubesat Space Protocol (CSP), which is later encapsulated over AX.25 frames. To ensure the imagery transmission to the GS a new tolerant transport protocol designated as Tolerant-CSP (T-CSP), is used. All these services are implemented in a module called Primary Satellite Interface Software (PriSIS) which runs on top of the COM Operating System. This very important application serves as the satellite data gateway when the satellite runs on the Normal profile.

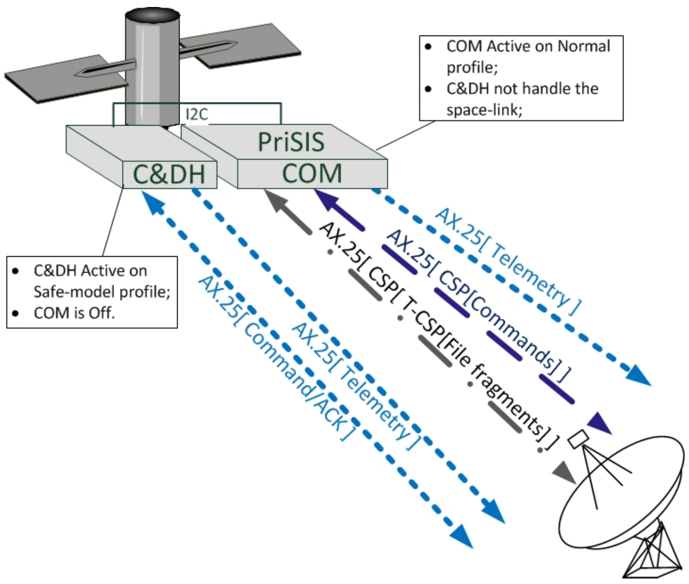
I2C
C&DH
PriSIS
COM
COM Active on Normal profile;
C&DH not handle the space-link;
C&DH Active on Safe-model profile;
COM is Off.
AX.25[Command/ACK]
AX.25[Telemetry]
AX.25[CSP[T-CSP[File fragments]]]
AX.25[CSP[Commands]]
AX.25[Telemetry]

Fig. 1. Communication options available in C&DH and COM.

Telemetry - The first protocol design decision had the specific objective of maximizing the number of potential GS on Earth that can receive the broadcast of this basic health satellite information (telemetry). To achieve this compatibility one widely deployed protocol must be used to ensure, as much as possible, the communication standardization. The protocol used for digital telemetry in the downlink (satellite to GS) is the AX.25, using exclusively Unnumbered Information (UI) frames.

The telemetry information is directly encoded into the AX.25 UI frame payload in a human-readable format (ASCII). Every piece of telemetry data is delimited by a special character ':'. Any kind of information with dynamic size can be encoded using this special character approach. The receiver does not need to know a priori each field length; it must know, however, the information format semantics.

Remote command reception and processing - The remote task invocation from GS operators on Earth raises a problem since each command can have other subsystems (rather than COM) as destination. To address this functionality the CSP protocol is used. In this solution, the CSP packets are encapsulated using AX.25 UI frames. The utilization of this layer 2 protocol under CSP enhances the error resilience by forcing a maximum CSP packet length (CSP have variable length packets) and providing error detection using its FCS mechanism.

Imagery gathering - In order to allow the correct transmission of onboard images, a layer 4 protocol must be employed. The rationale behind the use of such a transport protocol is the lack of available payload inside a CSP packet, due to the imposed AX.25 MTU restrictions. Every image with more than 253 bytes[5] needs to be fragmented into multiple CSP packets. This segmentation process also needs to take into account an efficient mechanism for missing packet retransmission, to overcome the issues imposed by a particular segment loss in the downlink. Since the CSP protocol does not have one transport layer implementation that performs fragmentation and packet recovery, a new solution called T-CSP was engineered. The Tolerant prefix has to do with the underlying asynchronous transmission mode provided by the AX.25 UI logic. This UI mode is very useful here as it does not have any time-based constraints. Thus, the segment transmission process can proceed continuously without any acknowledgement. The T-CSP rely on a Selective Negative ACKnowledgment (SNACK) logic to ensure packet recovery. The SNACK approach is adequate for this particular scenario because the space-link may suffer from high throughput asymmetry and delay. This asymmetry tends to be favourable (more throughput) for downlink, being important to use a parsimonious ACK mechanism to avoid the utilization of the uplink channel, as much as possible. This SNACK mechanism is implemented in the following conceptual way: After the PriSIS receives a request for file transmission, it starts sending each segment sequentially over the space-link. After each correct segment reception in the GS, the T-CSP checks if x previously segments were already correctly received; if not, it asks PriSIS for those missing segments. This process continues until all segments are correctly received allowing the image to be e.g. correctly displayed to the GS operator.

5. 256 bytes AX.25 MTU value and 4 bytes CSP header

In Fig 2 the T-CSP segment format is shown, where 8 bits are allocated for file identification, 16 for segment number and another 16 for total segments in the file being transmitted.

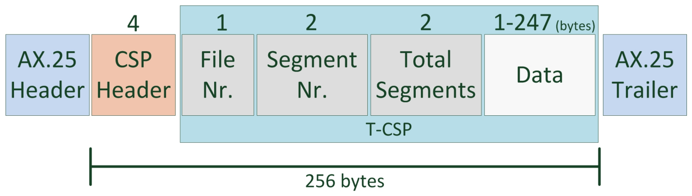

Fig. 2. T-CSP segment over CSP packet

3 Heart Implementation

Here we consider the different applications developed in each subsystem and the corresponding system's software that supports it.

To enable the software development for both subsystems, two different rapid prototyping boards were used. In what concerns the COM subsystem, an AT91RM9200 based board was utilized, namely the AT91RM9200-DK board. This board has an ARM920T CPU core running at 180MHz, and enough peripherals and interfaces for the COM, such as 10 Mbyte flash, I^2C and SPI. For the C&DH subsystem the software prototyping board used was the moteISTs5/1011. This homebrewed platform is equipped with a TI MSP430F5438A MCU. This ultra-low power micro-controller relies on a 16-bit CPU running with a 25 MHz system clock. It also has all the required interfaces such as I^2C, and provides different operational modes in terms of energy consumption.

3.1 COM base system

The COM operating system includes three different components: the boot-loader, the Linux Kernel and the userland environment, also known as root FileSystem (rootFS). The first implementation issue was what would be the best approach capable of minimizing the hardware resources and, at the same time, enabling a versatile platform for the PriSIS module. The system used was the Buildroot[6], which allows a complete embedded system parametrization through a set of meta-information files (e.g. Makefiles and patches). It also provides a simple approach to new developed software integration in a portable way. With this in mind, the buildroot seemed to be a reasonable choice for the rootFS.

The Linux kernel version used was the 2.6.38 vanilla source[7]. This version was used mainly because this is the last kernel version where it is possible to apply the AT91 support patch[8]. This patch is important because it enables/enhances the support of some of the required AT91RM9200 MCU hardware on Linux. Those features are related with e.g. DataFlash operation, I^{2}C and SPI bug corrections. The Linux kernel provides an important feature for the COM subsystem which is the the native AX.25 support as a loadable kernel module. This kernel module implements all the required AX.25 framing functions.

To enable the interaction with the exported AX.25 kernel driver functionalities from userland, the AX.25 library and AX.25-tools[9] are required. Since this software is not currently automatically supported by the buildroot packages, it was necessary to extend the buildroot framework to support the AX.25 protocol by creating the required buildroot packages. A buildroot package is composed by a set of patches and meta-information. The package is later integrated into the buildroot package system to allow the required software to be compiled together with all the remaining rootFS software. The resulting final rootFS image takes $\approx$ 1.5 Mbyte of storage capacity where the Linux/ARM kernel image occupies around 1.55 Mbytes.

6. http://buildroot.uclibc.org/
7. Unmodified kernel source from kernel.org
8. http://maxim.org.za/at91_26.html
9. www.linux-ax25.org

3.2 Beacon software and Primary Satellite Interface Software

Here, we highlight the most important implementation strategies and decisions taken on both the beacon software and PriSIS. Fig. 3 shows an overview of the entire PriSIS solution. The discussion follows a bottom up approach in terms of network layers.

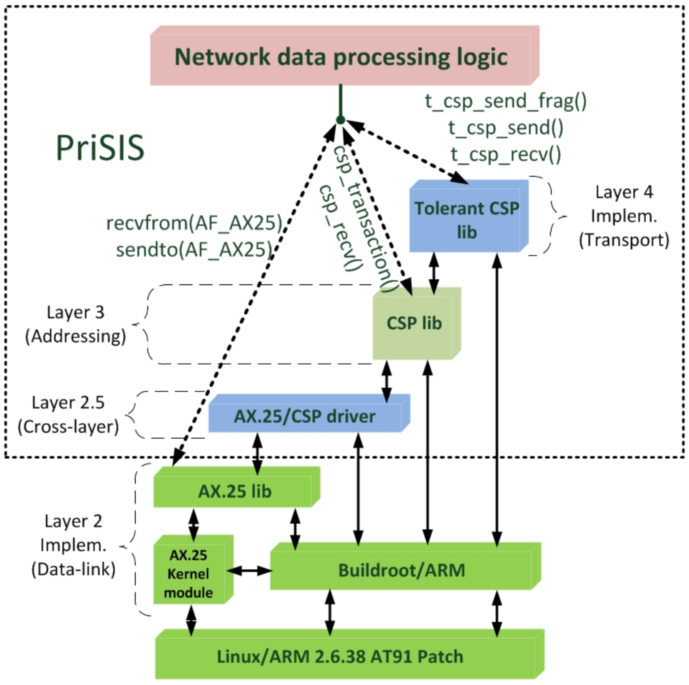

Fig. 3. Complete network stack overview

The onboard beacon software (designated as *pulsar*) is a daemon process that injects into the downlink, thanks to the AX.25-library, the available information about the current satellite health. The pulsar daemon uses the berkeley socket standard primitives to inject the collected telemetry into the AX.25 kernel module.

The PriSIS implements the command reception/response and imagery gathering functionalities, using the CSP and T-CSP protocols. It must implement the command reception/response mechanism over the CSP / AX.25 network stack. The CSP implementation does not support the AX.25 protocol. To overcome this lack of interoperability one CSP-AX.25/UI driver was created and added to the libcsp[10] implementation.

The native CSP implementation smooths the code compilation/configuration process through a waf script[11]. In addition to the new AX.25/CSP code that was integrated into the native csplib, the waf script was also extended to include this new AX.25 support. Therefore, no extra step is required when the csplib needs to be compiled with a AX.25 support.

One GS application was also developed to allow the interaction with PriSIS by a remote GS operator. This solution also implements the same CSP/AX.25 stack as PriSIS does. After being initialized, the GS software asks the operator for a command which will be properly encoded and sent to PriSIS.

The GS software also allows the operator to ask for imagery transmission. This function will be encapsulated as a normal CSP command. After this command is correctly received by PriSIS, it will trigger a T-CSP transmission, sending the onboard saved image to the GS. The T-CSP protocol is an extension to the already implemented CSP/AX.25 stack functions. It relies on the already developed primitives adding the new fragmentation and recovery functionalities.

3.3 C&DH Operating System and applications

Due to the real-time constraints associated with typical C&DH tasks, a RTOS (FreeRTOS) was considered in this case. The FreeRTOS for MSPGCC toolchain is being ported to the MSP430F5438A MCU by the FreeRTOS project. This common available port[12] was tailored to support the moteist++s5 platform. Since FreeRTOS code structure uses a Hardware Abstraction Layer (HAL) to implement the platform specific code, which is MCU independent, a new moteist++s5 HAL for FreeRTOS was created. This new FreeRTOS moteists5++ HAL contains the major platform configuration and the external peripherals functions implementation.

With the required moteIST I/O peripherals supported in FreeRTOS, the real-time application can be deployed. The developed application implements the redundant beacon function as well as the incoming I^{2}C data processing (e.g. from the COM subsystem). Beyond these functions, the application also receives simple serial commands from the Analog COM subsystem, namely the Data Framer component, replying with an ACK message if the command is correctly received.

4 Experimental evaluation

In order to validate the reliability, functionality and quality of the developed solution, one set of tests were performed for the Heart Unit. Since the Analog COM subsystem is currently under research and the intended radio link characteristics are difficult to emulate in the laboratory, the space link was abstracted using a serial connection from the GS computer to the COM subsystem. The same method was also applied to interconnect the C&DH subsystem with the GS computer. The lack of realistic physical conditions on this important link, makes the test accuracy hard to achieve. The test design phase had this fact into account together with the requirement specifications, specifically the safety-critical points, and produce two main test groups. On one hand, the first test group aims at the solution's performance evaluation, where both base systems (C&DH and COM Operating system) and intended network-based functionalities were evaluated. On the other hand, the second test group targets the solution quality validation for safety-critical environments. This second test group was formalized taking into account some directives found on *NASA Software Safety Guidebook* (GB-8719.13) [11].

10. https://github.com/GomSpace/libcsp
11. http://code.google.com/p/waf/
12. https://github.com/pabigot/freertos-mspgcc

4.1 COM subsystem

As Fig. 4 shows, the total required flash storage space is 3.6 Mbytes. This information was collected from the boot-loader memory organization. The COM software systems roughly takes 37 seconds to be launched when electrical energy becomes available.

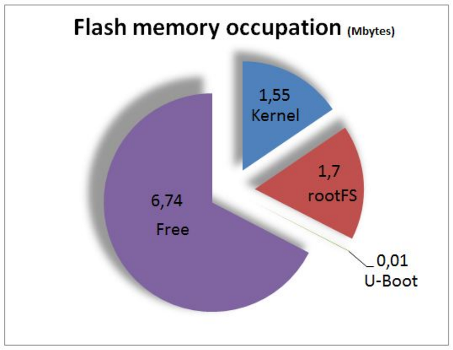

Fig. 4. Flash memory occupation

From all the SRAM available (roughly 32Mbytes) only 43% (12.6 Mbyte) is occupied when the subsystem is idling.

The next test consists in consecutive commands issued by the GS operator. Fig. 5 shows that SRAM utilization starts increasing but, at some point in time, seems to stabilize around the 55%. This stabilization gives a clue that there are no memory leaks on the PriSIS code.

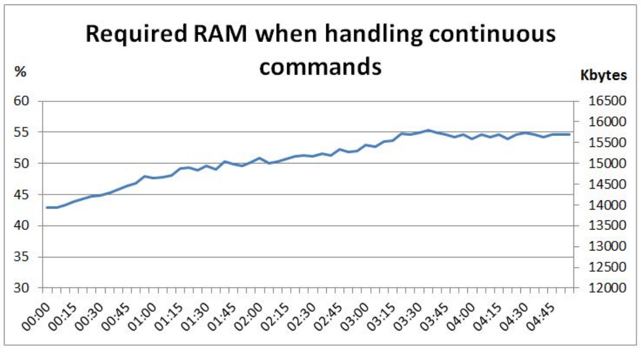

Fig. 5. RAM utilization over continuous command handling

The same behaviour is observed for the load values in Fig. 6. As long as more and more GS commands arrive at PriSIS, it requires more processing power to deal with them, thus raising the load values to around 65% of CPU utilization.

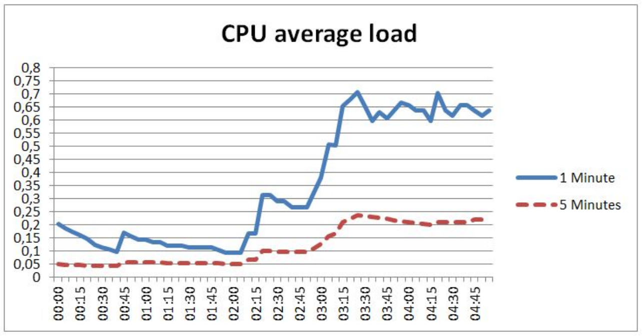

Fig. 6. CPU loads over continuous command handling

This high stressful scenario, which is unlikely to happen in a real-world setting, it is not yet the *killer application* in terms of COM processing power and memory characteristics.

Another aspect that was taken into account was the developed T-CSP time efficiency. Fig. 7 shows how much time T-CSP takes to transmit a set of segments when some disruption occurs in the space-link. This test was performed, using a 1200 bit/s serial connection for two possible different window verification sizes ($x = 1$ and $x = 5$[13]). These two sizes were chosen as a way to produce an evident contrast between these opposite (minimum and large) window sizes. The T-CSP receiver (GS on this case) timeout was configured to be 2,1 seconds since the frame time (with maximum payload) is about 1,8 seconds. The abscissa axis in Fig.7 represents the number of lost segments during multiple file transmissions. These lost segments were originated by unplugging the serial cable during multiple file transmissions, each transmission with different number of lost segments. The ordinate axis represents the time spent by T-CSP to transmit a sample image with $\approx$18 Kbytes which require 75 segments.

As Fig.7 illustrates, the lowest window verification size ($x = 1$) is more time efficient with these low bitrates. It is also possible to conclude, that the minimum time required to

13. x is the missing segment window size that reflects the maximum number of unreceived segments in a burst segment transaction and relative to the last correctly received segment.

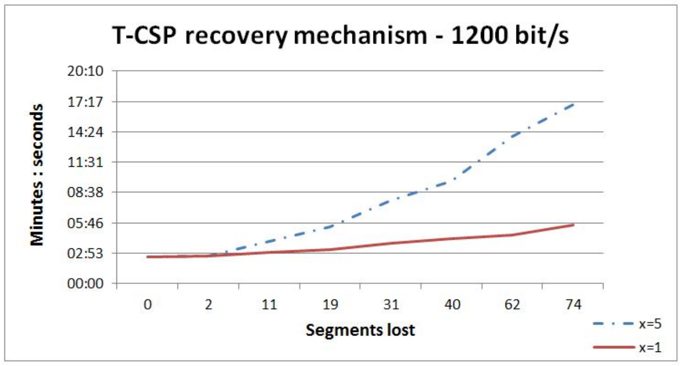

Fig. 7. File transmission at 1200 bit/s using T-CSP facing link degradation.

receive this 75 segment sequence at this bitrates is a little less than 3 minutes. It is also clear that for $x = 1$, in the worst case scenario (when almost all segments are lost - 74 in this case) is still advantageous the use of the T-CSP retransmission, rather than ask for the entire image retransmission. This efficiency gain regarding the $x = 5$ window as to do with less uplink utilization and fastest missing fragment verification on the GS software. With $x = 1$ the PriSIS buffer do not overflow, which also makes the entire retransmission faster. Finally, it is worthwhile to note that a typical satellite orbit with 20 minutes of communications opportunity will allow the transmission of about 500 segments (≈ 120 Kbyte of effective data) in good propagation conditions.

4.2 C&DH subsystem

The C&DH subsystem is difficult to test since it is a highly integrated platform. Unlike the COM, the C&DH does not have an intrinsic performance report mechanism (as the telemetry beacon on COM). The entire C&DH software solution (FreeRTOS OS, HAL functions and C&DH application code) uses 7 Kbytes (2.7%) of the 256 Kbyte available flash on the MSP430F5438A MCU. This utilization is acceptable and does not limit future C&DH software enhancements. Besides the Flash memory, the C&DH software RAM utilization is hard to evaluate in compile time. This C&DH solution utilizes at least $\approx 10.6 Kbytes$ of RAM. This represents 66.25% of all the MCU available RAM (16Kbytes).

4.3 PriSIS and beacon software safety/quality

The most important software applications developed for the Heart unit were the PriSIS and the beacon software aboard the COM subsystem. As such a larger test effort was employed on these software components to ensure its quality and reliability. The first test was the code visual inspection, taking into account the guidelines, precautions and standard verification processes found on [11, p. 214-218]. These guides state the limitations and problems with C language, programming C standards and the ten commandments for C programmers. This document also provides a ”Good Programming Practices Checklist” [11, p. 384-388].

Beyond the visual and conceptual code verification test, two group of tests were performed. These tests are grouped into statical analysis, which are done against the source code without executing the produced binaries, and dynamic analysis which are performed while executing the compiled binaries. The first static analysis tool used was the intrinsic GCC compiler warning enforcement. After this, the code was also inspected using a lint program called splint[14]. This static analysis tool reported some possible errors in the produced code. In the dynamic analysis test group, the valgrind tool was used to inspect about code runtime memory leaks and profiling.

5 Conclusion

The developed Heart unit presents a solution capable of handling the space-link communications between the ISTNanosat-1 and potential Ground Stations on Earth (Communications subsystem). It also presents an ultra low power solution capable of acting as the satellite central decision unit (C&DH subsystem). Both solutions were engineered taking into account the low space and power budget available on-board. Actually, these Cubesat intrinsic restrictions shaped the entire hardware and software developments. On one

14. http://splint.org/

hand, to meet the Heart unit requirements for the Communications subsystem it was necessary to fully parametrize a GNU/Linux Operating System trying to minimize as much as possible the scarce on-board resources utilization. On top of this GNU/Linux OS it was developed the communications software (Primary Satellite Interface Software) which implements the space-link network protocol details. The developed protocols allow a GS compatible telemetry beacon transmission, command reception from GS providing the required acknowledgements and imagery transmission from the CubeSat. This PriSIS software implements the developed Amateur X.25/Cubesat Space Protocol driver as well as the new transport protocol called Tolerant-CSP. On the other hand, to meet the C&DH subsystem requirements it was necessary to port the FreeRTOS version for TI MSP430F5 MCU with the MSPGCC toolchain for the moteists5++ platform. After the moteists5++ FreeRTOS port was done, it was developed an energy aware solution which is capable of communicating with the remaining subsystems using the system bus and implementing, on future projects, the intended decision making process, using the deployed real time scheduler.

One of the Heart major contributions was the development of the AX.25/CSP driver. This driver can easily be integrated into another CSP based projects since it was developed as a CSP extension without any extra software dependencies, except the AX.25-library. The T-CSP protocol was another achievement. This transparent API, abstracts all the underlying transport functionalities to the application. The included retransmission mechanism proved to be advantageous in disruptive scenarios and with common space-link bitrates.

References

[1] M. Davidoff, *The Radio Amateur's Satellite Handbook*, 1st ed. Newington: The American Radio Relay League, 2003.

[2] M. Swartwout, "The promise of innovation from university space systems: are we meeting it," in *Proceedings of the 23rd AIAA/USA Small Satellites Conference*, Logan, USA, 2009, pp. 1–6. [Online]. Available: http://www.usu.edu/ust/pdf/2009/october/itn10120930.pdf

[3] S. Lee, A. Hutputanasin, A. Toorian, W. Lan, and R. Munakata, "CubeSat Design Specification, Rev. 12," 2009. [Online]. Available: http://www.cubesat.org/images/developers/cds_rev12.pdf

[4] A. Bonnema, "ISIS Missions, Services and Technology Trends," in *CubeSat Summer Workshop*, Logan USA, 2011, p. 21.

[5] J. Bluck, "GeneSat-1 - Mission overview," 2007. [Online]. Available: http://www.nasa.gov/centers/ames/missions/2007/genesat1.html

[6] K. Woellert, P. Ehrenfreund, A. J. Ricco, and H. Hertzfeld, "Cubesats: Cost-effective science and technology platforms for emerging and developing nations," *Advances in Space Research*, vol. 47, no. 4, pp. 663–684, Feb. 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0273117710006836

[7] V. Cerf, Google/JPL, S. Burleigh, A. Hooke, L. Torgerson, NASA/JPL, R. Durst, K. Scott, The MITRE Corporation, K. Fall, Intel Corp, H. Weiss, and I. SPARTA, "rfc4838 - Delay-Tolerant Networking Architecture," 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt

[8] V. Cerf, "InterPlaNetary Internet," in *DARPA Proposers Day*, vol. 26, no. 6, Nov. 2004, p. 17. [Online]. Available: http://symoon.free.fr/scs/dtn/biblio/Cerf-IPN-DARPA.pdf

[9] V. Cerf and I. Society, "rfc3271 - The Internet is for Everyone," 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3271.txt

[10] W. A. Beech, D. E. Nielsen, and J. Taylor, "AX. 25 Link Access Protocol for Amateur Packet Radio," *Tucson Amateur Packet Radio Corporation, Tucson*, no. July, 1998. [Online]. Available: http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:AX+.+25+Link+Access+Protocol+for+Amateur+Packet+Radio\#0

[11] NASA - National Aeronautics and Space Administration, *NASA Software Safety Guidebook (GB-8719.13)*, nasa-gb-87 ed., 2004.