HIGH PERFORMANCE ON-BOARD IMAGE PROCESSING USING CANOPEN FOR EARTH OBSERVATION SATELLITES

Yago Isasi Parache [*1], **Dr. Pablo Ghiglino** [†2], **and Nicolas Perzo** [‡1]

[1]*LuxSpace Sàrl, Luxembourg*
[2]*Klepsydra Robotics, GmbH. Zurich, Switzerland*

ABSTRACT

Earth observation (EO) has been and still is a very complex topic in the Space Engineering field. Yet there is a growing number of new requirements and needs for new EO nano and microsatellites, working either standalone or in synchronized constellations. Higher download data rates, improved image quality and enhanced onboard computing capabilities are only a few examples of the challenges to be faced in this discipline. Furthermore, the need to speed up the time to market and reduce costs of development, has sparkled the use of modular solutions over monolithic ones for payload computer design, where purpose specific components - hardware and software - are reused and assembled for different spacecrafts and missions. In this work, we present a modular approach to on-board image processing based on a distributed computing setup with functionality specific computers connected to an on-board CAN-BUS using CANopen protocol. Optimal use of the resources in the network is achieved by a two-fold design consisting in using FPGAs for data processing and compression and in integrating a high performance software library data handling and data communications. The feasibility of this approach is tested in a two-node network for an EO payload with an image capture and compress node and an Earth Link node. Results are presented based on image size and performance parameters.

Key words: OBDH, OBDP, Earth Observation, CAN-BUS CANopen, flight software.

1. INTRODUCTION

1.1. Trends of small satellites

Since the year 2011, an increasing number of small satellites (<100Kg) has been registered. In particular 70% of the value is in the area of micro satellites in the range of 25Kg to 100Kg. According the NSR [1], the increasing trend of launches will continue in the following years to reach a $630M value by 2025.

While in the decade of the 80's 70% of the small satellites where dedicated to telecommunication applications the trend seems to indicate that by 2025 Earth Observation(EO) and situation awareness applications will cope most of the satellite market.

Table 1: Satellite global market payload application evolution

Payloads	1980s-1990s	2005
Earth observation	1.4%	47%
Situation awareness	2.3%	10%
Communications	69.2%	12%
In-orbit Demonstrators	11%	18%
Others	16.1%	13%

The global commercial satellite imaging market is a large growth area, with increasing demand for high resolution imagery. The market was valued at US $1.6Bn in 2014 and is projected to reach US$3.5Bn by 2024 [1]. The use of CubeSats and Microsatellites in satellite imaging, particularly EO, is rapidly increasing with the small satellite EO market value expected to be US$350M over the next six years. Sectors making use of EO satellite imagery include disaster monitoring, agriculture, civil engineering and energy as well as defence and security. Options for commercial off-the-shelf imaging systems that fit for a wide range of EO, are more and more available for satellite integrators to purchase and integrate into a satellite. It is also true, that the market is not fully developed yet at lower costs.

1.2. CAN in space

The controller area network (CAN) protocol is a highly reliable communication system for harsh environments and has been used in automotive and industrial applications since its presentation in 1986. The CAN two-wire bus multi-master/multi-drop topology makes it easy to add additional functionality to a system while significantly reducing the number of wires associated with

[*] isasi@luxspace.lu
[†] pablo.ghiglino@klepsydra.org
[‡] perzo@luxspace.lu

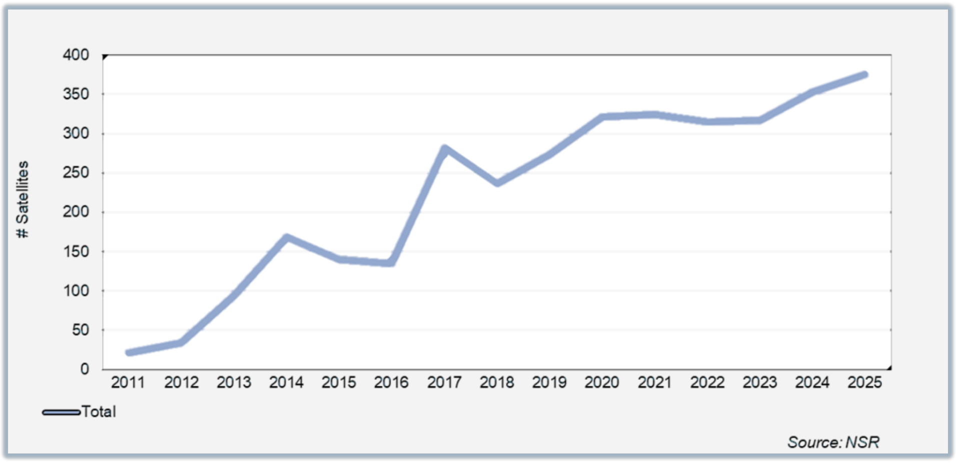


Figure 1: Trend of small satellite launches from 2011 to 2025.

the wire intensive point-to-point topologies used in the past. The space community was aware of the CAN protocols many benefits and advantages, and the need for it to replace traditional spacecraft bus architectures, however CAN was not fully used in space systems. With the emergence of of the QML-V qualified radiation tolerant CAN-Bus transceivers [2], and the qualification of several CAN-enable rad hard microcontrollers, design and implementing a CAN bus network for spacecraft onboard communications with the characteristics and benefits of terrestrial CAN-based embedded system is perfectly possible. The use of CAN-Bus for internal communications onboard spacecraft is becoming more and more common ground. The general use of CAN-Bus communication in spacecrafts is shown in e.g. [3], The use of CAN-Bus in CubeSats is presented in [4]. Furthermore, CANopen protocol is also used extensively in space. [5] presents the industrial use of CANopen in space. More importantly, [6] is the official specification provided by ESA for CANopen for onboard communications.

1.3. CANopen at LuxSpace

At LuxSpace the use of CAN-Bus systems was introduced during the development of the E-SAIL project. It has allowed engineers to replace older architectures with more complex wiring communications, with a CAN two-wire bus network. Reducing the number of wires and weight along with lower power consumption and easier testability that resulted in important cost savings due to the easier testability. In particular, during the design, implementation and testing of the E-SAIL project, we used a tailored implementation of the ECSS-E-ST-50-15C standard. LuxSpace vision was to reuse the existing and well proven parts of CANopen, as needed by the E-SAIL project, removing the parts which were not explicitly required. Comparing networks in the automotive and space domains, in a typical CAN network in an automotive environment a big advantage of CANopen is that device profiles exist and nodes can be easily replaced by products from other manufacturers. Each node, sharing the same device profile, provides a set of functions to identify it and configure it online to exchange information with the existing network. This allows to be very flexible and agile when setting up new networks. In the case of space, the network is well defined and all nodes and most parameters are known a priori. Moreover, the replacement of nodes after launch is indeed not possible. In E-SAIL, that was exactly the case, having a physical architecture based on a multi-drop topology, with two independent buses; one nominal and one redundant. Only one bus is active at a time (cold redundancy). The bus ends are terminated external to the units in order to allow flexibility for the arrangement of the unit. ISO11898-2:2003 transceivers have been used for both nominal and redundant bus. Additionally, Several tools and methods commonly used by automotive industry in design, development, production and maintenance of CAN bus networks for safety critical applications have been used successfully used during the implementation and testing of E-SAIL, as the CANoe tool. As a direct result of the E-SAIL development, the CANOpen modules developed in the frame of the project are expected to be reused as libraries in the future microsatellite projects in the company. Due to the market explosion on EO and Situation awareness satellites, it was necessary to step forward on extending the use-cases covered by the E-SAIL project. LuxSpace worked with Klepsydra on novel modular approach to on-board image processing based on a distributed computing setup with functionality specific computers connected to an on-board CAN-Bus using CANopen protocol. The performance of the network solution is a critical aspect for the future platform at LuxSpace, specially when high volumes of data will need to be processed through it. The following sections describe the system, the software design and the results of our study where optimal use of the hardware resources in the network is achieved by, from one side using FPGAs for data processing and compression and by integrating the Klepsydra software, a high-performance software library for data handling and data communications.

2. SYSTEM DESIGN

Firstly, there are several works show the advantages of using a distributed architectures. For example, [7] presents a distributed architecture for small satellites, [8] presents a fault-tolerance solution based on distributed architectures. Secondly, one of the current trends in the on-board computer design, is the use of small boards that host FPGA. Example of the use of this approach can be found in [9] and [10]. For our particular system we have selected a Zynq Board, that is attracting some interest by its cost and extended features, in particular the support for CAN. [11] has already made some experiments with this board in the ISS. Also, [12] proposes a visual navigation solution based in the Zynq board.

In regards to image compression, most solutions for Earth Observation are based in discrete wavelet transform (DWT). [13], [14], [15], and [16] are research work in this type of compression applied to satellites. Furthermore, [17] provides this specification for DWT compression applied to Earth Observation and also [18] presents an implementation of DWT for FPGA.

Luxspace and Klepsydra decided to work on a distributed architecture that could be used in the frame of EO projects. The solution proposes a two-node solution, with one computer connected to the camera and performing capturing and compressing, and the second node that receive the image via CAN-Bus and send it to Earth. In our system, both nodes are Zynq boards.

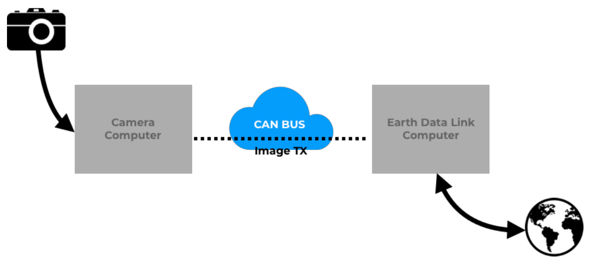


Figure 2: Distributed Earth Observation architecture.

3. SOFTWARE DESIGN

In terms of software, the different layers are described in Figure 3 .The so-called camera computer acts as a server application in charge of capturing the image, compressing it and transmitting it thought the CAN Bus. The image compression is performed in the FPGA. On the other side the client side receives the image though the CAN-Bus and downlinks to ground.

In terms of our specific implementation details, the proposed solution has the following features:

- DWT is used for image compression. In particular, we used CCSDS 122.0 and CCSDS 123.0
- CANopen as communication protocol to send the image from the camera node to the Earth data link node.

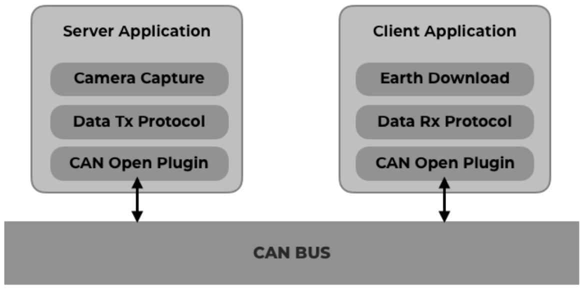


Figure 3: Software layer design.

- Klepsydra Space Software.

3.1. Klepsydra

The solution uses Klepsydra Space, a high performance software toolset for space flight software and payload solutions. Klepsydra is a test driven, agile oriented and event-driven software, inspired in cutting-edge software engineering tecniques used in the big technology companies (Netflix, Google, Amazon, etc) but also in the investment banking and electronic trading sectors. The main features of the Klepsydra software are:

- High performance and resource optimisation.
- Platform independent, high level and lightweight.
- User friendly and easy APIs.

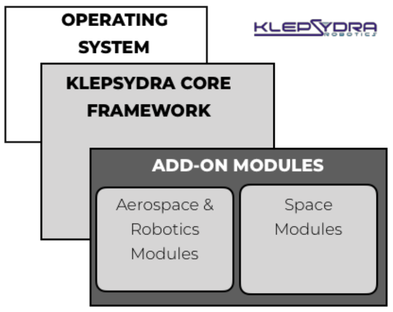


Figure 4: Klepsydra toolset overview

We have made use of the following features of Klepsydra:

- kpsr-core. High performance core library.
- kpsr-canopen. CANopen protocol connector.
- kpsr-admin. For performance monitoring.
- Custom module with a simple protocol for internal image transmission.
- Control service. To start and stop Earth Observation based on critical sensor data.

3.2. Transmission Protocol

The custom protocol developed our EO solution, consists of splitting the image in maximum size SDO segments (each segment is 889 SDO blocks), and using a dictionary object to control the flow that includes the status, the number of blocks and the current block index. The statuses of the solution are NEW_IMAGE, NEW_BLOCK, BLOCK_READ, LAST_BLOCK, IMAGE_READ. This information is all sent into one 8-byte block to save back and forth from with CANopen. Both nodes, transmitter and received, are constantly polling the CANopen Object Dictionary for changes in the control object.

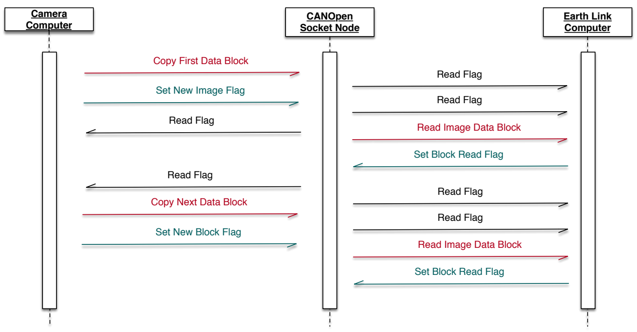


Figure 5: Data transmission protocol.

As part of our solution, we opted for using CAN Socket combined with CANopen. CAN Socket is a convenient solution proposed by several automotive vendors to allow share access to the CAN bus by different process running in the same host computer. One of the main implementation of this specification is CANopenSocket [19]. CANopenSocket is based on CANopenNode, which is an open source CANopen Stack. Klepsydra has developed a modified version of this implementation that makes better use of the resources and performs faster than the open source version.

3.3. Experimental Setup

The testing of this solution was perform in two different environments. First in Zynq boards with Petalinux [20] installed, and secondly in Zedboard with [21] installed. The image dimensioning is: 256KB before compression and 32KB after compression. Images were transmitted with a 0.1Hz and the protocol used busy wait of 477KHz, which was the optimal rate empirically found.

4. RESULTS

The results presented here are for the Petalinux setup which performs substantially better than Xilinux. The reason behind is that Petalinux is optimised for Zynq boards, while Xilinux is more oriented to development and prototyping. The throughput of data was approximately 100Kbps with an approximately 8% deviation. This throughput remained constant for different sizes of images we tested on the range from 64KB to 256KB. This can be seen in figure 6.

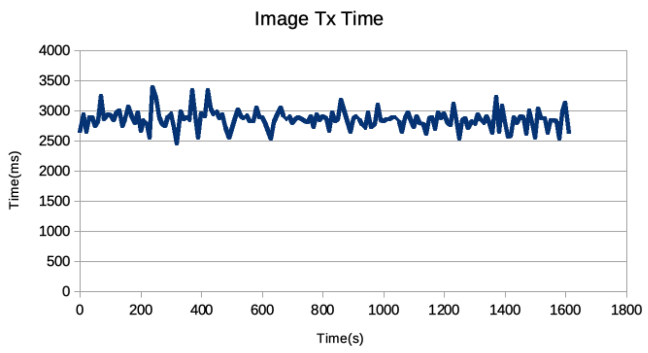


Figure 6: Image Tx Publication rate.

This rate is roughly consistent with the CAN-Bus rate specification. CANopen usually runs at 250 Kbps, out of which, approximately 50% is protocol overhead. Furthermore, our custom design introduce another extra overhead of roughly 20% depending on the busy wait rate, being finally the effective rate of 100Kbps. ($R_{effective} = 250Kbps \times 0.5 \times 0.8 = 100Kbps$).

Figures 8 show RAM and CPU consumption for server (transmitter) and client (receiver) processes. In these figures it can be seen that the memory consumption of both processes is constant, and the CPU in both cases is limited and also roughly constant. This is due to the use of Klepsydra, which one of its best features besides resource optimisation, is its stability and predictability.

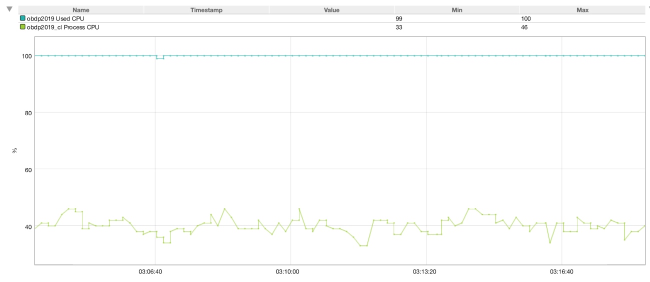
As Klepsydra bases all its usage in the publisher-subscriber pattern ([22]), in this section we show the performance of the publishers and subscribers in the client and server side. The use of busy wait can be seen to perform with great accuracy.

5. CONCLUSIONS

The Earth Observation solution based on distributed computing using CANopen as middleware presented together by LuxSpace and Klepsydra has shown not only a great throughput, but also stability and predictability that are needed in space conditions. Furthermore, this solution is modular and reusable, making time-to-market in Spacecraft much more attractive and affordable.

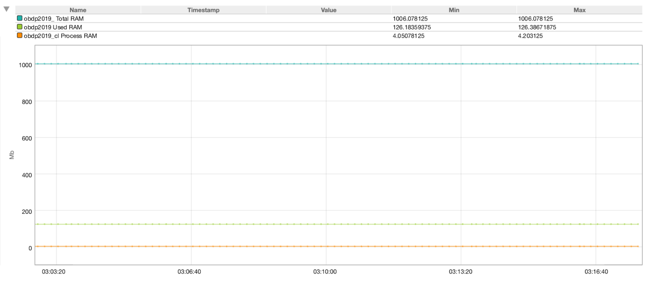
6. FUTURE WORK

Based on the same architectural principles described in this paper, a distributed visual navigation, where the image processing is done in the FPGA, while the host drives the navigation control can provide great benefits to the space community. There are already some works done in the field ([12], [23], [24]), which are the base for further



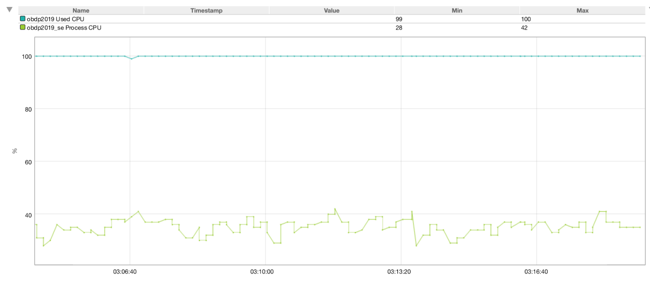
Name	Timestamp	Value	Min	Max
🟩 obdp2019 Used CPU			99	100
🟩 obdp2019_cl Process CPU			33	46

(a) Client CPU



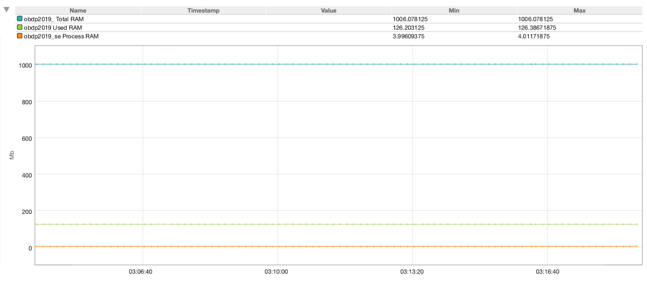
Name	Timestamp	Value	Min	Max
🟩 obdp2019_ Total RAM			1006.078125	1006.078125
🟩 obdp2019 Used RAM			126.18359375	126.38671875
🟧 obdp2019_cl Process RAM			4.05078125	4.203125

(b) Client Memory



Name	Timestamp	Value	Min	Max
🟩 obdp2019 Used CPU			99	100
🟩 obdp2019_se Process CPU			28	42

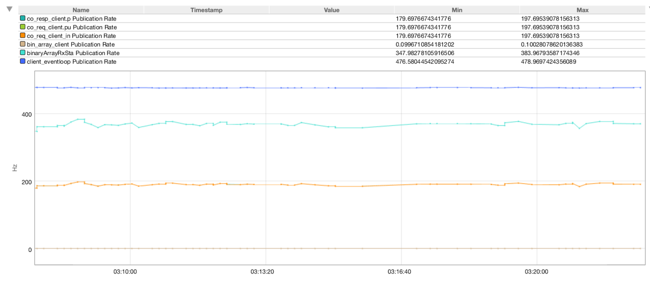
(c) Server CPU



Name	Timestamp	Value	Min	Max
🟩 obdp2019_ Total RAM			1006.078125	1006.078125
🟩 obdp2019 Used RAM			126.203125	126.38671875
🟧 obdp2019_se Process RAM			3.99609375	4.01171875

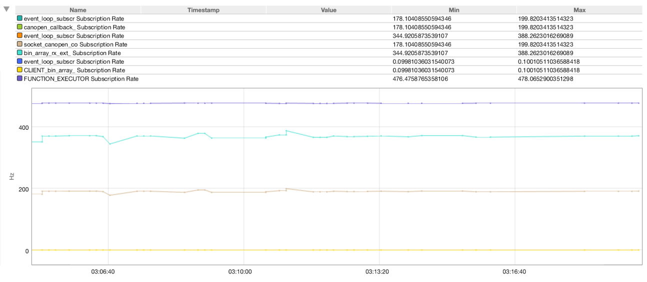
(d) Server Memory

Figure 7: Resources consumption in client and server side



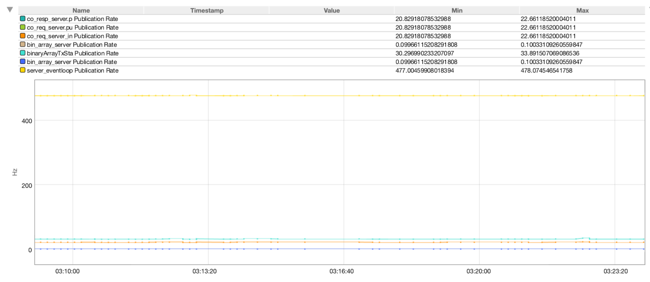
Name	Timestamp	Value	Min	Max
🟩 co_resp_client.p Publication Rate			179.6976674341776	197.69539078156313
🟩 co_req_client.pu Publication Rate			179.6976674341776	197.69539078156313
🟧 co_req_client_in Publication Rate			179.6976674341776	197.69539078156313
🟫 bin_array_client Publication Rate			0.0996710854181202	0.10028078620136383
🟦 binaryArrayRxSta Publication Rate			347.98278105916506	383.96793587174346
🟦 client_eventloop Publication Rate			476.58044542095274	478.9697424356089

(a) Client Publication Rate



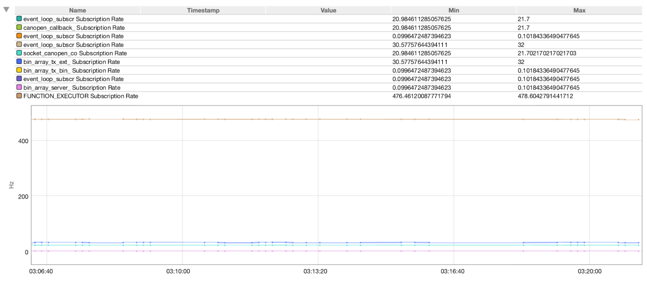
Name	Timestamp	Value	Min	Max
🟩 event_loop_subscr Subscription Rate			178.10408550594346	199.8203413514323
🟩 canopen_callback_ Subscription Rate			178.10408550594346	199.8203413514323
🟧 event_loop_subscr Subscription Rate			344.9205873539107	388.2623016269089
🟫 socket_canopen_co Subscription Rate			178.10408550594346	199.8203413514323
🟦 bin_array_rx_ext_ Subscription Rate			344.9205873539107	388.2623016269089
🟦 event_loop_subscr Subscription Rate			0.09981036031540073	0.10010511036588418
🟨 CLIENT_bin_array_ Subscription Rate			0.09981036031540073	0.10010511036588418
🟪 FUNCTION_EXECUTOR Subscription Rate			476.4758765358106	478.0652900351298

(b) Client Subscription Rate



Name	Timestamp	Value	Min	Max
🟩 co_resp_server.p Publication Rate			20.82918078532988	22.66118520004011
🟩 co_req_server.pu Publication Rate			20.82918078532988	22.66118520004011
🟧 co_req_server_in Publication Rate			20.82918078532988	22.66118520004011
🟫 bin_array_server Publication Rate			0.09966115208291808	0.10033109260559847
🟦 binaryArrayTxSta Publication Rate			30.296990233207097	33.891507069086536
🟦 bin_array_server Publication Rate			0.09966115208291808	0.10033109260559847
🟨 server_eventloop Publication Rate			477.00459908018394	478.074546541758

(c) Server Publication Rate



Name	Timestamp	Value	Min	Max
🟩 event_loop_subscr Subscription Rate			20.984611285057625	21.7
🟩 canopen_callback_ Subscription Rate			20.984611285057625	21.7
🟧 event_loop_subscr Subscription Rate			0.0996472487394623	0.10184336490477645
🟫 event_loop_subscr Subscription Rate			30.57757644394111	32
🟦 socket_canopen_co Subscription Rate			20.984611285057625	21.702170217021703
🟦 bin_array_tx_ext_ Subscription Rate			30.57757644394111	32
🟨 bin_array_tx_bin_ Subscription Rate			0.0996472487394623	0.10184336490477645
🟪 event_loop_subscr Subscription Rate			0.0996472487394623	0.10184336490477645
🟪 bin_array_server_ Subscription Rate			0.0996472487394623	0.10184336490477645
🟫 FUNCTION_EXECUTOR Subscription Rate			476.46120087771794	478.6042791441712

(d) Server Subscription Rate

Figure 8: Protocol performance

work in the field. In that sense, Klepsydra can help providing the optimisation and stability required for such a complex systems. LuxSpace has opted to continue with the development of microsatellites using CAN Networks and CANopen for spacecraft on-board communications and control. However, it is also desired to continue extending the use of the CANopen standard by enhancing the application layer protocol that operates in conjunction with the CAN Network data link layer providing a set of rich services and protocols useful to every device on the network. LuxSpace is currently developing its Triton-X platform, and evolution of the E-SAIL satellite, with 50 kg platform plus 30kg of payload. Triton-X is a satellite-based product line, designed with a high level of flexibility, which allows easy adaptation to different payloads and mission profiles (Earth observation, Space Situational Awareness (SAA), Communication In-orbit Test, Machine-to-machine (M2M) systems and for payload technology demonstration.), minimizing the associated non-recurring engineering cost.

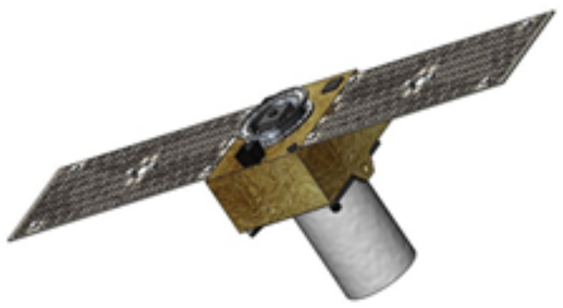


Figure 9: Triton-X EO design

The Triton-X bus will be based on CANopen and it is expected that fast integration of CAN compliant instrumentation will be needed to support the different client requirements. Is in here, where the presence of a software that isolates the low level drivers and protocols complexity such as Klepsydra could be of special benefit, bending the software development to focus on the application layer, enriching the functionality desired by the customers rather than in the problematic of the low level underworld. The first flight of a Triton-X satellite platform is scheduled for 2020.

REFERENCES

[1] Northern Sky Research. Northern sky research database. `https://www.nsr.com`, 2017.

[2] Cobham. Can-qmlv. `https://www.cobhamaes.com/news/2016/161006-CAN-QMLV.pdf`, 2016.

[3] C. Plummer, P. Roos, and L. Stagnaro. CAN Bus as a Spacecraft Onboard Bus. In *DASIA 2003 - Data Systems In Aerospace*, volume 532 of *ESA Special Publication*, page 51.1, 2003.

[4] Artur Scholz, Tian-Hao Hsiao, Jer-Nan Juang, and Claudiu Cherciu. Open source implementation of ecss can bus protocol for cubesats. *Advances in Space Research*, 62(12):3438 – 3448, 2018. Advances in Technologies, Missions and Applications of Small Satellites.

[5] RENESAS. Using can bus serial communications in space flight applications. `https://www.renesas.com/eu/en/doc/whitepapers/rad-hard/using-can-bus-in-space-flight-applications.pdf`, 2018.

[6] ESA. Can - controller area network bus. `http://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Computer_and_Data_Handling/CAN_-_Controller_Area_Network_Bus`, 2012.

[7] Bryan Palmintier, C Kitts, Pascal Stang, and Michael Swartwout. A distributed computing architecture for small satellite and multi-spacecraft missions. 08 2002.

[8] Muhammad Fayyaz and Tanya Vladimirova. Survey and future directions of fault-tolerant distributed computing on board spacecraft. *Advances in Space Research*, 58(11):2352 – 2375, 2016.

[9] L. Rockett, D. Patel, S. Danziger, B. Cronquist, and J. J. Wang. Radiation hardened fpga technology for space applications. In *2007 IEEE Aerospace Conference*, pages 1–7, March 2007.

[10] S. J. Visser, A. S. Dawood, and J. A. Williams. Fpga based real-time adaptive filtering for space applications. In *2002 IEEE International Conference on Field-Programmable Technology, 2002. (FPT). Proceedings.*, pages 322–326, Dec 2002.

[11] NASA. Stp-h5-center for high-performance reconfigurable computing (chrec) space processor (stp-h5 csp). `https://www.nasa.gov/mission_pages/station/research/experiments/1990.html`, 2019.

[12] George Lentaris, Konstantinos Maragos, Ioannis Stratakos, Lazaros Papadopoulos, Odysseas Papanikolaou, Dimitrios Soudris, Manolis Lourakis, Xenophon Zabulis, David Gonzalez-Arjona, and Gianluca Furano. High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. *Journal of Aerospace Information Systems*, 15(4):178–192, Feb 2018.

[13] I. Hacihaliloglu and M. Kartal. Dct and wavelet based image compression in satellite images. In *International Conference on Recent Advances in Space Technologies, 2003. RAST '03. Proceedings of*, pages 79–84, Nov 2003.

[14] I. Hacihaliloglu and M. Karta. Dct and dwt based image compression in remote sensing images. In *IEEE Antennas and Propagation Society Symposium, 2004.*, volume 4, pages 3856–3858 Vol.4, June 2004.

[15] Halah Saadoon Shihab, Suhaidi Shafie, Abdul Rahman Ramli, and Fauzan Ahmad. Enhancement of satellite image compression using a hybrid (dwt–dct) algorithm. *Sensing and Imaging*, 18(1):30, Nov 2017.

[16] Kristian Manthey. A new real-time architecture for image compression onboard satellites based on ccsds image data compression. 10 2014.

[17] ESA. Whitedwarf. `https://essr.esa.int/project/whitedwarf`, 2017.

[18] ESA. The use of reprogrammable fpgas in space. `https://www.esa.int/Our_Activities/Space_Engineering_Technology/Microelectronics/The_use_of_reprogrammable_FPGAs_in_space`, 2014.

[19] Canopensocket. `https://github.com/CANopenNode/CANopenSocket#canopensocket`.

[20] XILINX. Petalinux-sdk. `https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html`, 2017.

[21] Xillybus. Xillinux. `http://xillybus.com/xillinux`, 2019.

[22] Bruce Powel Douglass. Design patterns for embedded systems in c: An embedded software engineering toolkit. 2010.

[23] Mokhtar Aboelaze, Osama El-Debb, Ahmed El-Bayoumi Mansour, and Mohamed Ghazy. Fpga implementation of a satellite attitude control using variable structure control. 08 2014.

[24] Brent Edward Tweddle. Computer vision based navigation for spacecraft proximity operations. 08 2010.