SOFTWARE Y COMPUTADOR EMBARCADO BASADO EN COTS PARA EL EXPERIMENTO TASEC-LAB

Ángel Grover Pérez Muñoz,* Juan Zamorano Flores,* David González Bárcena,** Juan Antonio de la Puente*
* *Information Processing and Telecommunications Center (IPTC-UPM)*
** *Instituto Universitario de Microgravedad "Ignacio da Riva"*
Universidad Politécnica de Madrid

Resumen

TASEC-Lab es un proyecto desarrollado por estudiantes, con la ayuda de profesores, de los grupos IDR y STRAST de la Universidad Politécnica de Madrid (UPM), que será lanzado a bordo de un globo estratosférico. Su objetivo es caracterizar la transferencia de calor por convección en misiones de este tipo y realizar un estudio del entorno térmico durante las fases de ascenso y flote de la misión. En este artículo se describen los dispositivos y el computador embarcado basados en componentes comerciales (COTS) que han sido empleados en el experimento TASEC-Lab, así como también la arquitectura, metodología y herramientas elegidas para el desarrollo y validación del software de a bordo.

Palabras clave: Computadores y control, software de vuelo, desarrollo de software basado en componentes, sistemas empotrados heterogéneos.

1. INTRODUCCIÓN

El proyecto TASEC-Lab *(Thermal Analysis Support and Environment Characterization Laboratory)* nace de la experiencia del Instituto Universitario de Microgravedad "Ignacio da Riva" (IDR) de la UPM en el Control Térmico de misiones de globos estratosféricas con el objetivo de caracterizar mejor la transferencia de calor por convección, el entorno térmico y la dinámica en este tipo de plataformas[1, 2]. El IDR y el grupo de Sistemas de Tiempo Real y Arquitectura de Servicios Telemáticos (STRAST) han propuesto el experimento TASEC-Lab, que ha sido desarrollado íntegramente por estudiantes del Grado en Ingeniería Informática, del Máster en Sistemas Espaciales y del doctorado en Ingeniería Aeroespacial de la Universidad Politécnica de Madrid, con el apoyo del personal docente e investigador de ambos grupos.

TASEC-Lab será lanzado en un globo estratosférico desde el Aeropuerto de León en el mes de julio de 2021. La compañía B2Space[1] será la encargada de operar el lanzamiento dentro del programa *Fly your CubeSat*, en el que estudiantes de diferentes universidades diseñan y fabrican un experimento tipo CubeSat. Concretamente, TASEC-Lab tiene unas dimensiones de 330x130 mm, aproximándose al estándar de CubeSat, con un peso inferior a 3 kg y una estructura de aluminio (Figura 1).

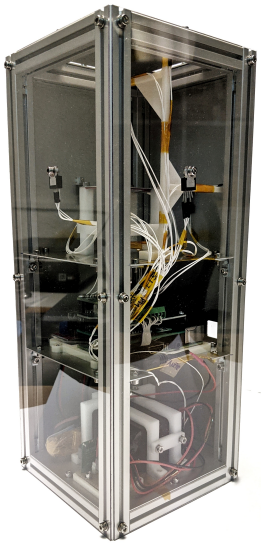

Figura 1: Vista general de la estructura de TASEC-Lab.

El grupo STRAST se encarga del desarrollo del software de vuelo y de la elección de los componentes de hardware, tanto del computador embarcado como de algunos dispositivos. El proceso de diseño y construcción de este experimento se realizó en unos seis meses, y por eso los sensores, actuadores, y el computador embarcado (OBC, *On-Board Computer*) elegidos son de tipo COTS *(Commercial Off-The-Shelf)*. En este caso las condiciones ambientales que tienen que soportar estos componentes no son tan extremas como las que se tendrían en una órbita baja.

Los laboratorios o experimentos de la misión se resumen en el apartado 2. En el apartado 3 se describen los componentes de hardware del sistema, el OBC, los sensores y actuadores que lo conforman, y su disposición en la estructura. Posteriormente, en la sección 4 se expone la metodología de desarrollo basado en componentes (CBD) y el conjunto de herramientas TASTE empleadas pa-

[1] `https://b2-space.com`

https://doi.org/10.17979/spudc.9788497498043.724

ra la implementación del software embarcado. La arquitectura de software, es decir los componentes de software y la comunicación entre ellos, se describe en el apartado 5. Finalmente, en el apartado 6 se presenta un resumen y las conclusiones de este trabajo.

2. DESCRIPCIÓN DE LOS EXPERIMENTOS

Para cumplir con el objetivo del experimento, TASEC-Lab cuenta con tres laboratorios conformados por una serie de sensores y actuadores que se resumen a continuación:

- *Attitude Determination Lab.* El laboratorio de determinación de actitud tiene como objetivo conocer la dinámica del vuelo (inclinación y orientación de la góndola) durante toda la misión. Para ello es necesario contar con una IMU (Unidad de Medición Inercial) y un dispositivo GPS (Sistema de Posicionamiento Global).
- *Environmental Lab.* Su objetivo es determinar el entorno térmico, es decir las variables o parámetros que afectan a la transferencia térmica, como la velocidad del aire o la presión atmosférica. Para ello el experimento cuenta con dos sensores de presión, un anemómetro y un calefactor para controlar la temperatura del anemómetro.
- *Heat Transfer Lab.* En este laboratorio se realiza un estudio de la transferencia de calor por convección. Para cumplir con su objetivo, el experimento está equipado con cinco sensores de temperatura digitales, seis termistores, y un calefactor.

Estos laboratorios no son independientes, ya que algunos sensores contenidos en unos laboratorios complementan la funcionalidad de otros. Por ejemplo, el *Heat Transfer Lab* necesita los valores obtenidos por los sensores de presión del *Environmental Lab* para las transiciones entre sus modos de funcinamiento.

La Figura 2 muestra el diagrama de contexto del sistema TASEC-Lab. En dicha figura se aprecia la relación entre el sistema y sus laboratorios.

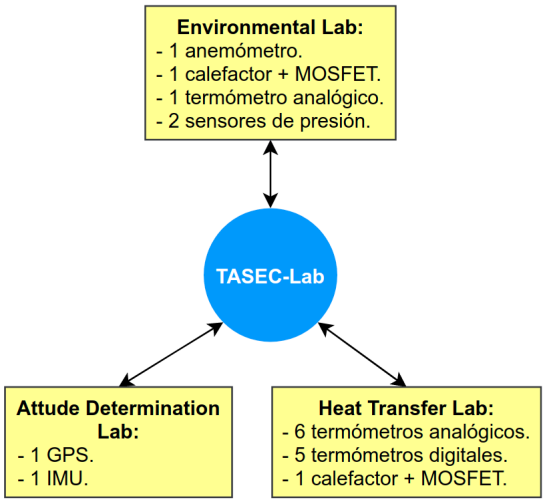

Figura 2: Diagrama de contexto de TASEC-Lab.

3. COMPONENTES DE HARDWARE DE LA MISIÓN

3.1. COMPUTADOR EMBARCADO

En cuanto al OBC, tras analizar los requisitos de la misión, teniendo en cuenta que las condiciones ambientales (niveles de radiación, temperatura, etc.) no son extremas, se eligió un computador Raspberry Pi (RPi) modelo 3B+, con las siguientes características:

- Procesador Dual-core Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz.
- 1 GB de LPDDR2 SDRAM.
- Puerto RJ45 Gigabit Ethernet.
- 40-GPIO header, 26 pines digitales GPIO (*General-purpose input/output*).

Adicionalmente a las funciones de entrada/salida en los pines GPIO (3.3 V para nivel alto), la RPi soporta diversos protocolos de comunicación que se indican a continuación:

- **IIC (Inter-Integrated Circuit)**:
  - Cantidad: 1.
  - 7 bits para la dirección de los esclavos, configurable a 10 bits.
  - Hasta 127 esclavos con 7 bits.
  - Transmisión: half-duplex.
  - Ancho de banda: hasta 100 Kbps, y 400 kbps en *fast mode*.
  - No está disponible el *clock stretching*.
- **SPI (Serial Peripheral Interface)**:
  - Cantidad: 2.
  - SPI-0: hasta dos CS (*Chip Select*).
  - SPI-1: hasta tres CS.
  - Transmisión: full-duplex.
  - Su velocidad de reloj es igual a la velocidad del reloj central.

https://doi.org/10.17979/spudc.9788497498043.724

- **UART (Universal Asynchronous Receiver Transmitter)**:
  - Cantidad: 2.
  - Solo uno disponible a través de los pines GPIO.
  - Transmisión: full-duplex.

La ventaja que implica usar este tipo de computador es que en el mercado hay disponible una gran variedad de módulos y HATs (*Hardware Attached on TOP*) que soportan estos protocolos, y por consiguiente facilitan el montaje y la integración de nuevos elementos al sistema.

Asimismo, se han empleado con éxito Raspberry Pi como OBC en misiones similares a TASEC-Lab, como los experimentos [11] y [12] propuestos para el programa BEXUS (Balloon Experiments for University Students) de la ESA (Agencia Espacial Europea), con características similares al programa *Flight your Cubesat* en el que se lanzará TASEC-Lab.

3.2. SENSORES Y ACTUADORES

En este sistema hay dos tipos de sensores y actuadores conectados al computador embarcado (OBC) para interactuar con el entorno:

- Dispositivos conectados directamente al computador a través de los pines GPIO, ya sea empleando las funciones de E/S o mediante los protocolos de comunicación que soporta, como SPI, IIC, o UART.
- Dispositivos que no se pueden conectar directamente al GPIO (por ejemplo, sensores analógicos). En estos casos se precisan componentes adicionales, como convertidores ADC o DAC, o drivers MOSFET.

El cuadro 1 presenta una descripción de las conexiones entre los dispositivos y el OBC. En la segunda columna se muestra el número de dispositivos incluidos en el experimento, y en la tercera columna se indica el tipo de interfaz entre cada dispositivo y el computador o el HAT (*Hardware Attached on Top*) que se utiliza para ampliar la capacidad de entrada y salida del OBC.

En el apartado 2, se han presentado los laboratorios y los dispositivos que los componen.

La Figura 3 muestra el diagrama de composición del sistema. En ella se pueden apreciar todos los dispositivos conectados al OBC, así como también el tipo de interfaz o protocolo de sus conexiones.

Todos estos dispositivos son componentes de tipo COTS. La mayoría de ellos se han ensamblado

Cuadro 1: Dispositivos conectados al OBC.

Dispositivo	Cant.	Interfaz
IMU 9DOF click Mikrobus	1	IIC
IIC MUX pHAT	1	IIC
Sensor de temperatura digital TC74	5	IIC
IRF520 MOSFET Driver Module	2	GPIO.
Anemómetro A100L2	1	GPIO.
Sensor de presión MS5611-01BA03	2	SPI
MIKROE-1032 GPS (u-blox LEA-6s)	1	USB
Termistor PT1000	7	ADC
Calefactor de silicona	2	MOSFET

mediante un HAT de prototipos que se puede enganchar en la parte superior de la RPi, resultando en un diseño más cómodo y robusto.

En la Figura 4 se ilustra el HAT con el IMU, un GPS, los dos sensores de presión y algunas resistencias empleadas como divisores de tensión para el anemómetro y los termistores.

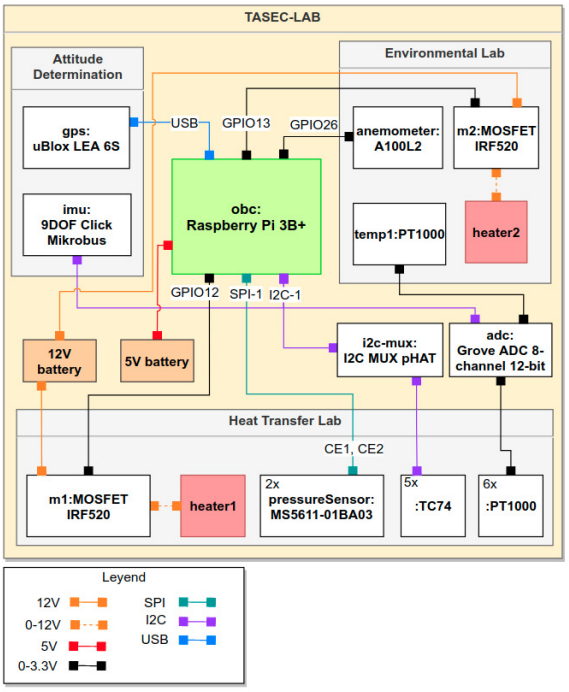

Figura 3: Diagrama de composición UML de TASEC-Lab.

https://doi.org/10.17979/spudc.9788497498043.724

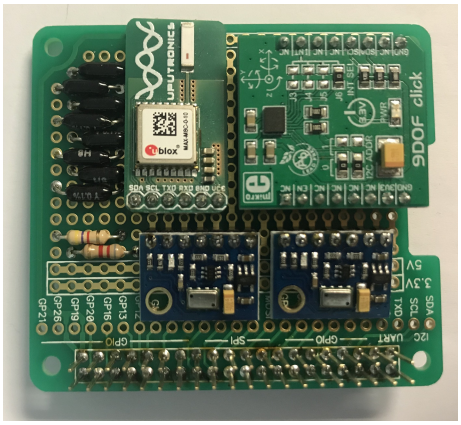

Figura 4: HAT de prototipos de TASEC-Lab.

4. METODOLOGÍA y HERRAMIENTAS PARA EL DESARROLLO DEL SOFTWARE

4.1. DESARROLLO BASADO EN COMPONENTES

El desarrollo basado en componentes (CBD, *Component-based development*), también conocido como ingeniería de software basada en componentes (CBSE, *Component-based software engineering*) es un enfoque de ingeniería de software cuya entidad de diseño más básico (*building block*) es el componente de software.

En la especificación UML 2.1 (*Unified Modelling Language*) [6], un componente de software se define como una unidad reemplazable del sistema que proporciona y requiere un conjunto de especificaciones. Asimismo, en la *Onboard Software Reference Architecture* (OSRA) de la iniciativa SAVOIR, un componente de software se define como una pieza de software más abstracta que las clases y los objetos del paradigma OOP (*Object Oriented Programming*), que presenta las siguientes propiedades [3]:

- Actúa como un proveedor de servicios, con o sin estado.
- Actúa como un consumidor de servicios implementados por otros componentes, formándose una relación de contrato entre el proveedor y el cliente. Es decir, el componente proveedor se compromete a ofrecer unos servicios específicos, y el componente cliente puede acceder a estos servicios tal como se especifica en el contrato.
- Puede ser visto como una caja negra que encapsula, oculta, y abstrae los detalles de implementación.
- Su implementación puede realizarse a partir de otros componentes existentes o desde cero en algún lenguaje de programación (visto como una caja blanca).

El desarrollo del software de este proyecto ha seguido este enfoque debido a los principios arquitectónicos sobre los que se asienta el CBD: encapsulamiento, consistencia, bajo acoplamiento, y reusabilidad. Además, gracias a dichos principios, la calidad del software se incrementará reduciendo el tiempo y los costes del futuro mantenimiento.

4.2. CONJUNTO DE HERRAMIENTAS TASTE

El conjunto de herramientas TASTE (*The ASSERT Set of Tools for Engineering*) tuvo su origen en el proyecto europeo ASSERT [13]. TASTE ha sido extendido y mantenido por la Agencia Europea del Espacio (ESA), con la colaboración de grupos universitarios e industriales. Estas herramientas están orientadas al desarrollo de sistemas de tiempo real empotrados heterogéneos con un enfoque de ingeniería de software basada en componentes (CBSE) y generación automática de código.

TASTE permite describir formalmente un sistema mediante cuatro vistas [14]:

- *Datos (data view)*: describe los tipos de datos que se usan en el sistema y que intercambian los componentes como se especifica en la vista de interfaces.
- *Interfaces (interface view)*: describe los componentes del sistema y las interfaces que ofrecen o solicitan de otros componentes. Los componentes son heterogéneos, en el sentido de que pueden estar modelados con distintas herramientas y lenguajes de modelado o implementación.
- *Despliegue (deployment view)*: describe la aplicación en hardware de las funciones especificadas en la vista de interfaces. Esta vista permite especificar la comunicación entre elementos remotos, mediante elementos como puertos serie, sockets, etc.
- *Concurrencia (concurrency view)*: proporciona una descripción global del sistema, permitiendo simular su ejecución y realizar análisis temporal o de planificación.

La elección de TASTE para desarrollar el software del experimento TASEC-Lab se debe fundamentalmente a dos razones:

https://doi.org/10.17979/spudc.9788497498043.724

- Soporte a la implementación del software embarcado. El diseño basado en componentes y la abstracción que se consigue con las diferentes vistas permite encapsular detalles de implementación como el sistema operativo. drivers, mecanismos de comunicación, lenguajes de programación, etc.

- TASTE soporta el modelo computacional de Ravenscar [15], restringiendo el código generado de acuerdo con este modelo con objeto de garantizar un comportamiento temporal predecible, que permita analizar el cumplimento de los requisitos temporales del sistema.

5. SOFTWARE EMBARCADO

5.1. DESCRIPCIÓN GENERAL

El software embarcado del experimento TASEC-Lab interactúa con el entorno físico a través de una serie de sensores y actuadores (Figura 2) y, por tanto, es un sistema de tiempo real empotrado.

A grandes rasgos, el OBSW (*On-board software*, software embarcado) lleva a cabo las siguientes funciones:

- Control de los sensores y actuadores,

- implementación de todos los experimentos (apartado 2),

- almacenamiento en memoria persistente de todos los datos recogidos por los sensores de los experimentos.

La arquitectura de software está basada en capas jerárquicas y almacenes de datos, siguiendo los principios definidos en las referencias [7, 8]. Los requisitos funcionales del software sugieren la siguiente descomposición:

- *Gestión de dispositivos.* Esta capa se encarga de la adquisición de datos de los sensores y del control de los actuadores según un esquema cíclico o por petición (esporádico).

- *Almacén de datos.* Esta capa representa una estructura de datos que contiene todos los valores recogidos por la capa de gestión de dispositivos, actuando como mediador entre esa capa y la capa de aplicaciones.

- *Aplicaciones.* Esta capa realiza las funciones del *heat transfer lab*, el control térmico del anemómetro, y los componentes de registro de datos.

El sistema operativo y los manejadores de dispositivos conforman una cuarta capa situada en la parte baja de la arquitectura, que ofrece sus servicios a todos los componentes del OBSW. Esta capa interactúa con el OBC y el hardware conectado a él mediante los manejadores (drivers) de los distintos dispositivos, accesibles en el espacio de usuario por medio de bibliotecas de software esopecíficas.

5.2. PROCESO DE DESARROLLO

Como se ha dicho anteriormente, se ha seguido el enfoque CBD para el desarrollo e implementación del software de TASEC-Lab. A continuación, se muestra el proceso seguido, que está basado en los procesos de desarrollo e implementación definidos en la referencia [3]:

1. **Definición de los tipos de datos e interfaces**

   Este paso consiste en definir los tipos de datos y las interfaces. En TASTE, la definición de los tipos de datos se puede escribir en el lenguaje ASN1 dentro de la vista de datos.

   Hay que tener en cuenta que, con objeto de preservar el modelo de Ravenscar, TASTE solo admite una operación por interfaz. Lo más cercano a un conjunto de operaciones está definido como un *grupo de conexiones* en la nueva versión gráfica de TASTE (*Space Creator*). Sin embargo, su único objetivo es visual (no abrumar la interfaz gráfica con muchas conexiones).

2. **Definición de los tipos de componentes**.

   En este paso se definen los tipos de componentes. Un tipo de componente está compuesto de: (i) una o más interfaces proporcionadas (PI), (ii) cero o más interfaces requeridas (RI), y (iii) un conjunto de atributos accesibles en el componente con una serie de modificadores (de solo escritura, de lectura, o ambos).

   Este paso se puede realizar en la vista de interfaz de TASTE. En dicha vista, se pueden definir los tipos de funciones (*function types*) del sistema, y las PIs y RIs que implementa el componente. La diferencia entre un tipo de componente y una *function type* en TASTE es que la segunda también contiene la implementación del componente (es tipo e implementación al mismo tiempo).

3. **Implementación de los tipos de componentes**.

   En este paso se implementan los tipos de componentes definidos en el paso anterior. Este

https://doi.org/10.17979/spudc.9788497498043.724

paso se realiza en la vista de interfaz de TASTE.

4. **Creación de ejemplares de los componentes**.

   Este paso consiste en crear ejemplares (*instances*) de los tipos de componentes implementados en el paso anterior.

5. **Enlazado de los componentes**.

   Una vez creados los ejemplares de componentes, en este paso se realizan las conexiones entre ellos a través de sus interfaces. Aquí también se definen los atributos no funcionales de las interfaces como por ejemplo: el tamaño de la cola para una interfaz esporádica, el tiempo de ejecución en el peor caso (WCET), el tiempo mínimo entre activaciones (MIAT), el periodo de una interfaz cíclica, etc.

   Este paso se puede realizar en la vista de interfaz de TASTE mediante la creación de ejemplares de funciones tipo. Además, TASTE también permite establecer los atributos no funcionales antes mencionados.

6. **Definición de la arquitectura física**.

   En este paso se realiza un modelo del hardware relevante del sistema, como procesadores (CPU), dispositivos, instrumentos, buses que interconectan estos elementos, etc.

   Este paso se puede realizar en TASTE ya que la arquitectura física se puede definir en la vista de despliegue. En dicha vista se pueden modelar los siguientes elementos:

   - nodos,
   - particiones,
   - plataformas o entornos de ejecución,
   - drivers y buses para comunicar varios nodos.

7. **Aplicación de los componentes en la arquitectura física**.

   Por último, en este paso se aplican los componentes de software a los componentes de hardware definidos en la arquitectura física. Las comunicaciones que se realizan entre componentes ubicados en nodos remotos también se deben asignar a un bus de comunicación concreto. Todo esto se puede realizar en la vista de despliegue de TASTE.

6. CONCLUSIONES

Se ha presentado la estructura y metodología de desarrollo del computador embarcado del experimento TASEC-Lab. Para su desarrollo se han empleado componentes comerciales, en el caso del hardware, y las herramientas TASTE para el software. Gracias a las facilidades de abstracción de TASTE y a su proceso de generación de código compatible con el perfil de Ravenscar se ha conseguido desarrollar todo el sistema en el plazo establecido y pasar las pruebas funcionales sin contratiempos.

Agradecimientos

Los autores quieren manifestar su agradecimiento a todos los miembros, estudiantes y profesores, que conforman el proyecto TASEC-Lab.

English summary

ONBOARD SOFTWARE AND COMPUTER BASED ON COTS FOR THE TASEC-LAB EXPERIMENT

Abstract

TASEC-Lab is a project developed by students, with the help of teachers, of the IDR and STRAST research groups from the Technical University of Madrid (UPM, Universidad Politécnica de Madrid), that will be launched from a stratospheric balloon. Its main purpose is to characterize the heat transfer by convection in this type of mission, and to study the thermal environment during the ascent and float phases of the mission. The following article describes the equipment and onboard computer based on COTS selected for the TASEC-LAB mission, as well as the architecture, methodology, and toolset used during the development and validation of thg on-board software system.

Keywords: Computers and control, onboard software, Component-Based Software Development, heterogeneous embedded systems.

Referencias

[1] D. Gonzáles, A. Fernández, I. Pérez, y A. Sanz. Real data-based thermal environment definition for the ascent phase ofPolar-Summer Long Duration Balloon missions

https://doi.org/10.17979/spudc.9788497498043.724

from Esrange (Sweden). En *Acta Astronáutica 170 (2020)*, páginas 235-250, 20 de enero de 2021.

[2] I. Pérez, A. Sanz, N. Bezdenjnykh, A. Farrahi, P. Barthol, y R. Meller. Thermal control of SUNRISE, a balloon-borne solar telescope. En *Proc. IMechE Vol. 255*, páginas 1037–1049, febrero 2011.

[3] A. Jung, M. Panunzio, y J. Terraillon, *SAVOIR-FAIRE – On-Board Software Reference Architecture*, Savoir-Faire, 10-jun-2010 [En línea]. Disponible en: `https://essr.esa.int/project/osra-onboard-software-reference-architecture`.

[4] Alejandro Alonso, Emilio Salazar, y Juan A. de la Puente. Design of On-Board Software for an Experimental Satellite [En línea]. Disponible en: `https://www.dit.upm.es/~str/papers/pdf/alonso&13a.pdf`

[5] B2Space, "The Blue Jay Programme" [En línea]. Disponible en: `https://b2-space.com/blue-jay-programme/`.

[6] Grady Booch, James Rumbaugh, e Ivar Jacobson, *Unified Modeling Language User Guide, The (2nd Edition)*, Addison-Wesley Professional, 2005.

[7] J.A. de la Puente y J. Zamorano, OBDH_LABS, repositorio de GitHub, `https://github.com/STR-UPM/OBDH_LABS`

[8] Jens Eickhoff. *Onboard Computers, Onboard Software and Satellite Operations*, 2012.

[9] M. Perrotin, T. Tsiodras, J. Delange, y J. Hugues, *TASTE Documentation*, 27-ene-2012 [En línea]. Disponible en: `https://download.tuxfamily.org/taste/snapshots/doc/taste-documentation-current.pdf`.

[10] Universidad Politécnica de Madrid, "El IDR lanzará este verano un experimento en un globo estratosférico" 24-ene-2021 [En línea]. Disponible en: `https://www.upm.es/Investigacion?id=b0c70088bbe89710VgnVCM10000009c7648a____&prefmt=articulo&fmt=detail`.

[11] "Student Experiment Documentation - BEXUS 20. Cosmic Particle Telescope", Timo A. Stein *et al.*, 18-may-2016 [En línea]. Disponible en: `http://rexusbexus.net/wp-content/uploads/2016/05/BX20_CPT-SCOPE_SEDv5-0_18May16_reducedFileSize.pdf`

[12] "Student Experiment Documentation - BEXUS 19. Attitude determination system for a pico satellite based in a star tracker, a horizon sensor and Earth's magnetic field measurements", GranaSAT, 15-ene-2015 [En línea]. Disponible en: `http://rexusbexus.net/wp-content/uploads/2015/07/BX19_GRANASAT_SED_v5-0_15Jan15-reduced.pdf`

[13] Eric Conquet. ASSERT: a step towards reliable and scientific system and software engineering. *Embedded Real Time Software and Systems (ERTS2008)*, Jan 2008, Toulouse, France.

[14] Philippe Kruchten. The 4+1 View Model of Architecture. *IEEE Software*, Volume 12 Issue 6, November 1995.

[15] Alan Burns, Brian Dobbing, Tullio Vardanega, *Guide for the use of the Ada Ravenscar Profile in high integrity systems* April 2003 ACM SIGAda Ada Letters XXIV(2)

https://doi.org/10.17979/spudc.9788497498043.724