UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías de Telecomunicación

Simulador simple de OBDH (On Board Data Handling) basado en software defined radio

Autor: Fernando García Arias
Tutor: Víctor P. Gil Jiménez
23/09/2015

(página en blanco)

Título: SIMULADOR SIMPLE DE OBDH (ON BOARD DATA HANDLING) BASADO EN SOFTWARE DEFINED RADIO

Autor: FERNANDO GARCÍA ARIAS

Director: DR. VÍCTOR P. GIL JIMÉNEZ

EL TRIBUNAL

Presidente: VANESSA GÓMEZ VERDEJO

Vocal: PEDRO JOSÉ MUÑOZ MERINO

Secretario: LUIS JORGE ORBE NAVA

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día 8 de Octubre de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle una calificación de:

VOCAL

SECRETARIO PRESIDENTE

(página en blanco)

Agradecimientos

Gracias. A mi familia en general y a toda la gente que para bien o para mal ha pasado por mi vida durante estos 23 años y 95 días de vida. Porque este *TFG* para mí no es un simple trabajo sino el reflejo de una forma de ser de la que tan orgulloso estoy y, la dedicación y el trabajo durante todos estos años. Sin toda esa gente no habría llegado hasta aquí.

A mi mami María Dolores, mi papi Fernando y a mi hermana Lydia. A los pachunchis, ukachaka… y todas las horas de juego de niño con mi hermanísima. Te quiero hermanita. Gracias a la convivencia con ellos y, a su magnífica e inmejorable educación soy lo que soy. La vida es contínuo movimiento y las cosas vienen como vienen. El cariño y la educación como pilares básicos, eso es lo que me han enseñado. Espero ser el fiel reflejo de lo que ellos son. Eternamente agradecido.

También a mis abuelos Teresa y Antonio, a los que agradezco la dedicación y el cariño que siempre me han mostrado. Mi trabajo también está formado por el que ellos hicieron en Francia y desde esa base del esfuerzo forman parte. A los que ya no están, Josefa y Manuel, con sus caramelos de café con los que inicié esta carrera y que ya no podré recibir de sus manos. Por y para ellos, descansen en paz.

Agradecimiento también a mis segundos padres Magda y Paco, y a mi hermanita Nayara. Durante largos y malos períodos en estos últimos años su casa ha sido un refugio, los partidos de mi querido Real Madrid con Paco un alivio, las charlas con Magda un consuelo y las horas de juego con mi niña la alegría de cada día al llegar de la universidad.

A toda mi familia, tíos y primos en general quiero darle las gracias, pero a mi tita Ana en especial. Por ella estoy aquí, ella me aconsejó meterme en este grado cuando yo no sabía si meterme en arquitectura, en caminos… Y me dijo que esto de las telecomunicaciones tenía futuro. Le hice caso y no puedo estar más contento con ese consejo y mi decisión. Como dijo John Lennon, *"Life is what happens to you while you're busy making other plans"*. A toda la familia de Húetor Vega y a mi prima Amanda con la que viví mis veranos de infancia y a la que deseo la mejor de las suertes en la vida porque se lo merece. Si me acuerdo de esos veranos también lo hago de mi tito Repe al que, aunque casi no veo, tengo que decir que mis recuerdos de esos veranos con él son maravillosos, por lo tanto, gracias.

A mis amigos, en especial a Agencia GdM: Gutmor, Mariano, Edu, Raúl y Rober. Aunque sean unos pendencieros, me han ayudado tantísimo en algunos

momentos que se merecen que respete sus costumbres. Hemos vivido tanto juntos en estos últimos años que han hecho cambiar en parte mi forma de ver la vida. Los fifas, las pizzas, los trolleos, las risas, las grabaciones… eso me ha dado vida durante los últimos años. Les deseo lo mejor de lo mejor. Igual que a mis amigos de la universidad en especial a Irene, David y Campillo, al pádel y tantos momentos juntos. Y los que están por venir. También a JF, a mis vecinos, a mi her, a mis amigos de la infancia, a los del fútbol, a pipiolo, a Laura, Javiete, Adrianete, Benítez, Alex, a las chicas, a MN… No me quiero dejar a nadie así que todos, estén o no mencionados, están presentes.

En total, con estas 253 páginas, 49.400 palabras, 291 figuras y 51 tablas espero haber conseguido plasmar lo que ya no es un simple *TFG* sino también el trabajo de una vida académica. En este aspecto quiero tener presente centros como los Escolapios de Monforte de Lemos, Escolàpies de Figueres, el Pedro Antonio de Alarcón y, como no, el I.E.S. Maestro Matías Bravo de Valdemoro. En especial en este último se forjó mi formación como ciudadano de un lugar llamado mundo e indudablemente mi base académica con dos de mis referentes a nivel docente, Daniel Pescador y Manuel Abella. Sus consejos siempre los tengo presentes. A todos los profesores que me han ayudado a llegar a este punto, gracias.

Ya en la Universidad Carlos III de Madrid, gracias a todos los profesores de todas las asignaturas, en especial a Víctor que me propuso hacer este trabajo cuando yo no tenía aún ni idea qué hacer. Espero no haber sido muy pesado con él. A Daniel y Félix, compañeros en este último trámite, y a las largas horas en transmisión y recepción. Ha hecho falta muchísimo trabajo y mucho esfuerzo pero el resultado final y todo el proceso ha merecido la pena.

Sin toda esa educación de calidad que me ha ofrecido la enseñanza pública no sería lo que me considero ya, ingeniero de telecomunicaciones. Por muchos planes que cambien, muchas tasas que suban y muchas subvenciones que recorten, la enseñanza pública de calidad seguirá presente y hay que defenderla a capa y espada. Espero que por fin los que manejan nuestro dinero se den cuenta de lo que queremos hacer con él. Ya va siendo hora.

En último lugar, gracias al destino. Por ponerme en esta familia, con estos amigos, en este país y en esta época. En segundo quise dejar la carrera cuando iba con 7 de 11 a Junio. Me mantuve firme y al final todo se saca. Durante estos cinco últimos años he pasado la peor época de mi vida y al final, una de las mejores. Manteniendo la cabeza fría y aprendiendo cómo se juega a esto de la vida al final todo es posible.

(página en blanco)

Resumen

Históricamente las telecomunicaciones siempre han sido una parte fundamental del desarrollo en numerosos ámbitos (social, defensa, tecnológico, económico…). Con el inicio de la era espacial, las comunicaciones dieron un salto cualitvo en las comunicaciones a nivel mundial. Toda esa tecnología se basa en el desarrollo de los satélites y en ese proceso son parte fundamental los simuladores. En ese punto se enmarca este *Trabajo Fin de Grado*.

En este documento se presenta una solución de un simulador de manejo de datos de a bordo (*On-Board Data Handling, OBDH*) que consiste en la implementación de once servicios que emulan el funcionamiento de algunas funciones del satélite. Se llevan a cabo funciones de asentimiento y reporte del estado de ejecución, simulación de datos de *housekeeping* y monitorización (frecuencia y temperatura del microprocesador, y estado de los instrumentos), generación de reporte de eventos con distintos niveles de gravedad, simulación y gestión de la memoria, sistema de tiempos del satélite, ejecución de telecomandos de manera automática de la lista de planificación, reporte del estado de la conexión y reinicio del sistema.

Además, se simulan dos instrumentos: un sensor de temperatura y un instrumento de imágenes. Se pueden modificar distintos parámetros referentes a los instrumentos y, reportar los datos de temperatura y las imágenes. Para la implementación del programa se emplea *LabVIEW* como herramienta de programación visual que permite emplear periféricos de comunicación radio.

Este proyecto emula el funcionamiento de un satélite y se complementa con otros dos que simulan la transmisión de telecomandos y la recepción de telemetrías en la estación base. Para la comunicación con la estación base se emplea la tecnología *SDR (Software Defined Radio)* junto con los periféricos *USRP-2920* de *National Instruments*.

Palabras clave: *telecomunicaciones, satélites, OBDH, simulador, LabVIEW, telecomandos, telemetrías, SDR, USRP.*

(página en blanco)

Abstract

Historically, telecommunications has always been a fundamental part of development in many areas (social, defense, technological, economic...). With the start of the space age, communications took a quantum leap in communications worldwide. All this technology is based on the development of the satellites and in this process simulators are fundamental. At that point this *Final Degree Work* is framed.

In this paper, a solution of an *On-Board Data Handling (OBDH)* consisting of eleven implementation services that emulate the operation of some functions of the satellite is presented. Functions are performed as assent and execution status report, simulation and monitoring data housekeeping (frequency and temperature microprocessor and instrument), events report generation with different gravity levels, simulation and management memory, time satellite system, automatic telecommand generation of the schedule list, report connection status and reboot.

In addition, two instruments are simulated: a temperature sensor and an image instrument. It can modify different parameters relating to the instruments and report temperature data and images. For the program implemetation, *LabVIEW* is used as a visual programming tool that allows the use of radio communication peripherals.

This project emulates the operation of a satellite and is complemented by two others Final Degree Works that simulate remote telecommand transmission and telemetry reception in the ground station. For communication with the base station *SDR (Software Defined Radio)* technology is used together with the *National Instruments USRP-2920* peripherals.

Keywords: *telecommunications, satellites, OBDH, simulator, LabVIEW, telecommands, telemetry, SDR, USRP.*

Extended abstract

This *Final Degree Work* consists on the development of a simulator software of an on-board satellite system including the communication with the ground(s) station(s) using *Software Defined Radio (SDR)*.

This project is related with the *On Board Data Handling (OBDH)* of the spacecraft and its software simulation. This allows receiving signals, processing them and generating a response to the ground station. The general operation of the system consists in the reception of one telecommand (*TC*) transmitted by the ground system and its processing [2], [3]. These telecommands include orders for the *On-Board System* and in some cases is necessary generate a response (telemetry –*TM*-). This process is shown in the figure below:

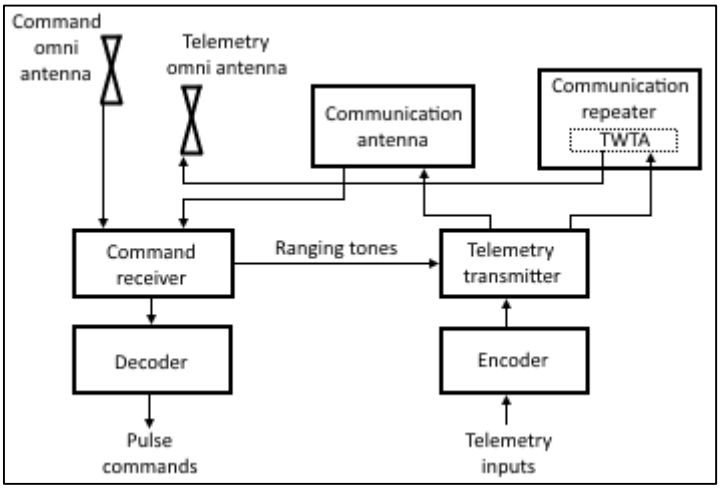


Figure I. Telemetry, tracking and ranging subsystem [1]

When a telecommand packet is received, it's necessary to decode and verify the information. Then, it runs the process depending on the value of the *Sevice Type* and *Subservice Type*. In some cases is necessary generate additional telemetry.

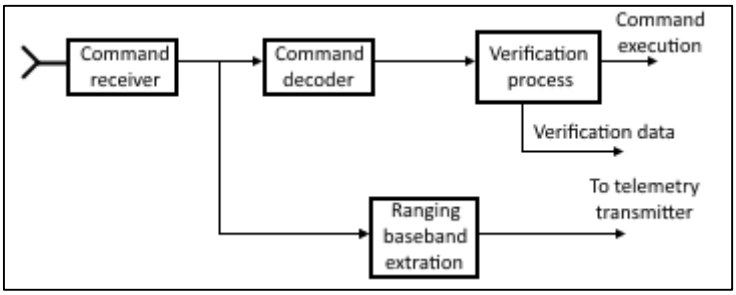


Figure II. The command system of a satellite [2]

[1] ***Based on reference [3], Page 112, Figure 4.17 Telemetry, tracking and ranging subsystem***

[2] ***Based on reference [3], Page 114, Figure 4.18 The command system of a satellite***

In this project, the simulation of the satellite receives from the same ground station that sends (theoretically) but the physic implementation of the ground system is separated. This *TGF* is complemented by other two *TFG*s. On one hand, Daniel Rodríguez Mogollón has implemented the part of the ground system that sends the TC ("*Simulador de EGSE (Electric Ground Supportment Equipment) [TELECOMANDOS] mediante plataforma Software Defined Radio"*). On the other hand, Félix Jiménez Monzón has made the reception part ("*Simulador de EGSE (Electric Ground Supportment Equipment) [TELEMETRÍA] mediante plataforma Software Defined Radio"*). The system structure is shown in the figure ***Figure 1.2.***

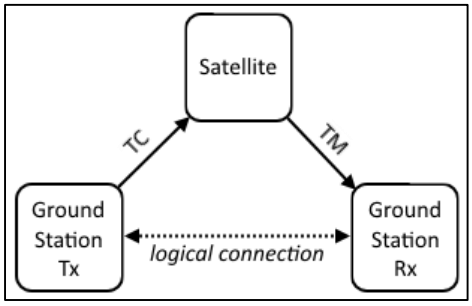


Figure III. System structure

In particular, it is intended to design a program simulation data management software from a satellite following the standard "*Ground systems and operations - Telemetry and telecommand packet utilization"* [1] of the *European Space Agency (ESA)* for the design of the frame formats exchanged between satellite and ground station. To achieve this, we have established a number of features and tools that owns the satellite, and are shown in the following illustration:

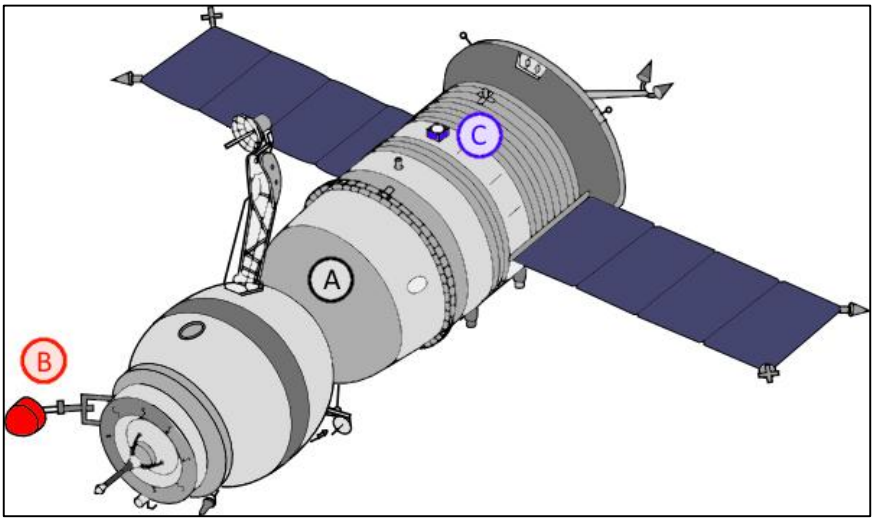


Figure IV. Simulated spacecraft structure[3]

[3] ***Based on reference [24]***

As shown in the previous figure, it will simulate a satellite with a main module (A), a temperature sensor instrument (B) and an image instrument (C).

Inside the main module include:

- ***Receiving module*.** This part of the system allows uplink communication. It is the module in which the synchronization sequences (receiving packets) are identified. For communication from the base station *modulation Binary Phase Shift Keying (BPSK)* is used to achieve a realistic simulation. The frequency used for the simulation is 650 MHz for the uplink.

- ***Transmission module.*** This part of the system allows communication in the downlink. Generate synchronization sequences and thus the transmission of telemetry. The frequency used for the downlink simulation is 600 MHz.

- ***Processing module.*** It is included within the receiving system and serves as verification of remote controls. It is the core of satellite processing. Within this module implements the *Service 1*. Maintains communication with the base station about the process of acceptance and execution of the remote received (failure acceptance and execution simulation).

- ***Housekeeping and diagnostics data module*.** It allows report monitoring data of the satellite. These reports can be generated automatically by the module itself or be requested by the ground station. This module complies with the functionalities for managing *Service 3* reporting diagnostic data (deletion, activation and deactivation of the reported data, request reports and change the period of the automatic report generation).

- ***Event module.*** As the system is designed, you can simulate event report generation with four severity levels (normal progress, low gravity, medium severity and high severity). Within this module includes *Service 5* which allows the generation of simulated events reports, clearing the event log, activation and deactivation of the report generation, and request the status of numerous events.

- ***Storage system (memory).*** There are three memories: Internal memory, *RAM memory* and *ROM memory*. Fulfills the functions of Service 6 that enables various operations with the simulated satellite memories (load data into *RAM*, memory fault simulation, load

predefined data of the *ROM memory* in the Internal memory...). You can also request the report of the data contained in *RAM*, and ultimately in the internal memory.

- ***Temporal satellite system.*** Within this module a record of the time it takes. The day, month, year, hour, minute and second sets. This module serves as the *Service 9* that enables exchange of the temporary system of satellite and request temporary reference report of the spacecraft. This module is essential for the proper operation of other modules, such as schedule TC or image instrument.

- ***Telecommand Schedule system*.** With this module it is possible to program remote controls running independently in an instant of time. This module complies with the duties of management of the *Service 11*. Management functions of planning allows to enable and disable the planned execution of remote controls, erasing all the planning list, inserting and deleting remote controls are implemented on the list, changing the scheduled execution time of the remote controls of the list and the report of the schedule list.

- ***On-Board Monitoring system*.** This module is closely related to housekeeping and allows monitoring simulates satellite data (temperature and frequency of the microprocesor, operating status of the instruments...). Fulfills the functions of the *Service 12*. The implemented functions are the activation and deactivation of the monitoring data, erasing the full list monitoring, insertion and deletion of parameters monitoring, and the report of the monitoring list.

- ***Connection test system*.** Fulfills the functions of the *Service 15*. It is one of the most important because it lets to check the status of the connection between the satellite and the ground station.

- ***Reboot module.*** Like its predecessor, this module is critical as it allows to return to the initial state.

Moreover, within the main module include an operations log and a summary. The option of connecting is also allowed by the TCP transport protocol with an Android device for recording the main system and interact with the program. It has been created an application to achieve this additional function.

Another module is the temperature sensor instrument. A virtual instrument sampling space temperature it is included. This instrument fulfills the functions of the *Service 32*. It allows to control the instrument on and off, change the sample

acquisition time and its report, the report of the state of the instrument, reporting temperature data stored and the interruption of the report.

Image instrument is also included. A virtual instrument that simulates taking pictures of space is implemented. This instrument fulfills the functions of the *Service 64*. It allows to control the instrument on and off, changing the time of acquisition of the photographs and its report, the instrument status report, the report of the stored image and the transmission abort of the image.

The operating diagram of the system is shown on ***Figure VI. Logical operation diagram***. In the picture you can see the complexity and magnitude of the system.

First, the system starts and is initialized (in case you want to display the tracking information at the mobile TCP connection is expected).

Then the core program starts working. Many processes are performed in parallel, reception and processing of remote controls, interactions between different parts, temporal dependences bewtween the time System and other modules, telemetry transmission, interaction with the logging module operations, data exchange with the Mobile (if the TCP connection has been created the connection)...

If a reboot telecommand is received, the operation of all modules stops and returns to the initialization.

For a better understanding of the ***Figure VI. Logical operation diagram*** and how the system is structured there is a caption below:

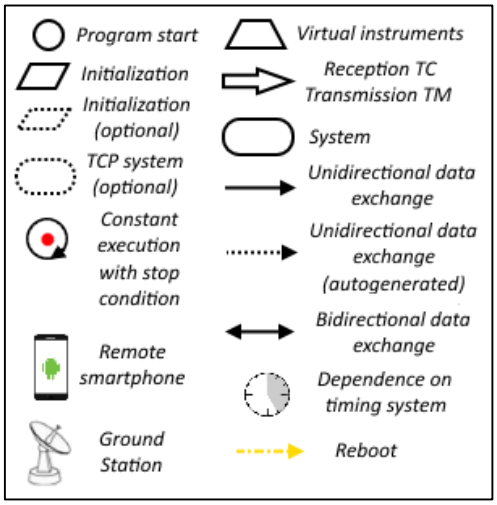


Figure V. Caption of the Figure 1.6

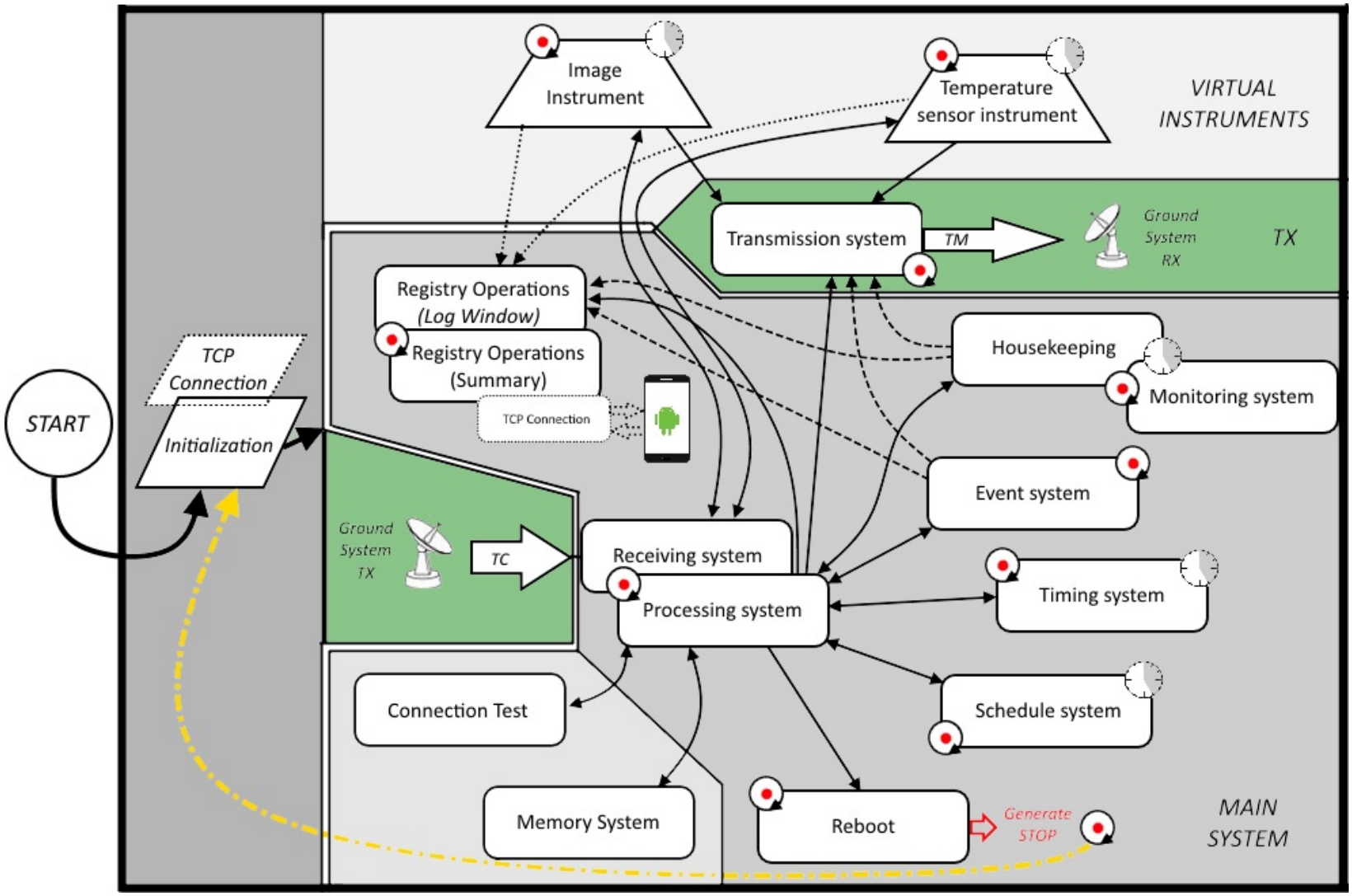


Figure VI. Logical operation diagram

Having described the operation of the system, it proceeds to the description of the chosen programming environment and communication hardware elements that are used.

For the emulator implementation is used *LabVIEW*, visual programming software that lets to use external hardware components with specific libraries of *National Instruments* for radio communication. The devices used for wireless communication are the *USRP-2920* of *National Instruments*.

Figure VII. NI USRP-2920[4]

Definitly, it is used *SDR (Software Defined Radio)* as a radio communication system where components that have been typically implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system [26].

In the National Instruments website there are some characteristics about this radio comunication System:

"NI USRP-292x software defined radio (SDR) transceivers are designed for wireless communications research and education. Programmable with NI LabVIEW software, NI USRP™ (Universal Software Radio Peripheral) hardware is an affordable and easy-to-use RF platform for rapidly prototyping applications" software defined radio" [10], [11].

"Software defined radios (SDRs) have emerged as a viable prototyping option for next generation wireless research by enabling researchers to quickly prototype a system, characterize it's performance, and iterate on the design." [10], [11].

[4] ***Figure obtained from [11]***

(página en blanco)

Índice

(página en blanco)

Índice de figuras

(página en blanco)

Índice de tablas

(página en blanco)

1. Introduction *[ENGLISH]*

This section puts in historical, socioeconomic and regulatory context the project and presents the logical structure of the system. Based on this points, it exposes the motivation and object of the project (goals).

1.1. Introduction

Since the beginning, the Human have always been fascinated by the universe and stars. First of all, in the following lines it is briefly described the space age and the satellite history [4], [7], [8], [9].

In 1952, the International Council of Scientific Unions decided to establish as the *International Geophysical Year (IGY)* between the 1957 and 1958, because in this time period, the cycles of solar activity would be at its high point. In October 1954, the council adopted a resolution calling for artificial satellites to be launched during the *IGY* to map the Earth's surface. In July 1955, the White House decided to launch an Earth-orbiting satellite for the *IGY*.

The October, 4th 1957, the Soviet Union successfully launched the first artificial satellite placed into Earth-orbit. It weighed 83.6 kg and its diameter was 58 cm. It took about 98 minutes to orbit the Earth on its elliptical path. This event marked the start of the space age and the space race between US and USSR The *space race* was a competition between the two Cold War rivals for the supremacy in spaceflight capability. It has left a legacy of Earth communications satellites. The *Sputnik* launch changed everything a cause of it was the most technical achievement in the space history and caught the world's attention. This event forced the United States to react. But on November 3, the Soviets struck again with the launch of the *Sputnik II* and including a dog named Laika.

On January 31, 1958, the U.S. successfully launched Explorer I. This satellite discovered the magnetic radiation Van Allen belt around the Earth. The 1st October, was created the *National Aeronautic and Space Administration (NASA).* The Soviet space program launch Luna 2, the first space probe to hit the moon in 1959. After two years, in April 1961, the Soviet cosmonaut Yuri Gagarin become the first person to orbit Earth. He travelled in the Vostok 1. The reaction of the USA was on May with the launch of Alan Shepard, the first American in space, and in 1962 John Glenn became the first American to orbit Earth.

On July 20, 1969, Neil Amstrong, on the *Apollo 11* space mission, became the first man to walk on the moon' surface. He famously called the moment "one

small step for man, one giant leap for mankind". With this gesture, the United States definitely "won" the space race.

Referring to the satellite communications, in 1962, the first active communications satellite was launched: *Telstar 1*. It was a simple single-transponder *Low Earth Orbit (LEO)* satellite but it could receive radio signals from the ground station and, amplify and retransmit them back over a large portion of the earth' surface. During seven months, Telstar 1 allowed the image transmission of TV programs.

During the 60s and 70s, the development of space communications technology came quickly. Satellites were mainly used at first for international and long-haul telephone traffic and distribution of select television programming using satellite broadcasting. By the 90s, satellite communications would be the primary means of distributing TV programs around the world. Today, the satellite communications allow to handle huge volume of data and it's vital for the technological development.

Considering the operation of the system discussed above, *LabVIEW* with the *NI USRP* has been considered the appropriate tool for the simulator system development. This tool allow the transmission and reception of signals using the modulation toolkit in *LabVIEW*. In the *National Instruments* website there are explained some characteristics of the *USRP*:

- *"Tunable center frequency from 50 MHz to 2.2 GHz covering FM radio, GPS, GSM, radar, and ISM bands."* [11].

- *"NI USRP-292x transceivers provide relevant, hands-on laboratory learning in RF and communications as part of an affordable teaching solution."* [11].

- *"NI USRP hardware and LabVIEW software offer students a unique and relevant opportunity to experiment with a wide range of real-world signals in introductory communications and digital communications laboratories."* [11].

All these features are suitable for this project.

1.2. Regulatory framework

In reference to the regulatory framework [19] - [20], this project conforms to the rules regulating electromagnetic transmissions. In the *Cuadro Nacional de*

Atribución de Frecuencias, *CNAF* [33], allocations to radio services and applications of the different frequency bands listed in Spain.

In this project, there is used the *UHF band* (470 MHz-862 MHz) for the accomplishment of the tests. In this frequency band is emitted the television in Spain *Televisión Digital Terrestre (TDT)* [33]. Internationally, spectrum use is regulated by the *Radio Regulations* of the *ITU* and by the decisions of international bodies such as the *European Conference of Post and Telecommunications (CEPT)* or the EU. Nationally, responsible for managing the use of spectrum and licensing is the *Ministerio de Industria, Turismo y Comercio*, through the *Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información (SETSI)*.

In addition, to the theoretical approach of the system and frame formats, it is taken as a reference the standard *ECSS-E-70-41A* of the *European Space Agency (ESA)*, named *Ground systems and operations – Telemetry and telecommand packet utilization* [1], and developed by the *European Cooperation for Space Standardization (ECSS*).

1.3. Socioeconomic environment

Concerning the socioeconomic status, the *Universidad Carlos III de Madrid* encourages the cultural and technological development. Each year many research projects are carried out and projects are developed internationally.

The transmission of knowledge and the contribution to society of technological advances developed at university are good for development at all levels of society (advances in medicine, handling large volumes of information and analysis -*Big Data*-, industry, telecommunications, responsible energy consumption, environment, education...).

Technological development provided by these projects impacts the economic development. With advances in the different branches of knowledge that requires a technological development of both public and private economic investment it is generated. This development allows the generation of jobs and stimulates economic funding at local and national level.

Receiving grants allows project work as presented in this document, which requires tools and software licenses that are given by agreements with leading companies such as National Instruments. The cost of the agreement to obtain the licenses with the total cost of the project is included in the section 19.

1.4. Motivation

The motivation of the project lies mainly in the rise of wireless and large data management, especially multimedia telecommunications. For years it is essential for technological development the use of satellite communications [4], [5], [6]. Many public institutions and private companies employ satellite communications for mobile communications, positioning systems (like *GPS*, *Galileo*…), streaming media, space research, weather forecasting, study of the Earth... Currently, the situation of the information society is a plausible business opportunity.

Simulation programs are essential today for programming the system on board the satellite because it is vital to the success of the mission that the satellite works perfectly. If not, repair costs become unviable. The satellite programming errors can cause the loss of millions of dollars for the companies. This fact makes simulation programs for radio communications charged importance in the development of the satellites and it's a chance.

As seen in the introduction, *LabVIEW* creates a simulation method with development of wireless communication between multiple devices (*USRP*). As mentioned in its website: *"With graphical programming syntax that makes it simple to visualize, create, and code engineering systems, LabVIEW is unmatched in helping engineers translate their ideas into reality, reduce test times, and deliver business insights based on collected data"* [23] . In addition, there is an agreement between *National Instruments* and the *Universidad Carlos III de Madrid* for obtaining the necessary project development licenses.

On a personal level of academic development, this tool involves learning visual programming and reveals the functioning of the systems simulation and radio communication systems. In essence, the development of this work allows to deepen the knowledge of on-board data handling and know in depth the process of communication with the ground station. It also allows to know the process of design communication models and working with standards of institutions such as *ESA*.

1.5. Goals

The main objective of this *Final Degree Work* (hereinafter *TFG*) is to develop a software simulation of a simple and realistic solution to the problem described. It is also describing the operation of an on-board data handling of the spacecraft. It aims to show the process of communication between the ground station and the satellite. Moreover design decisions both frequency distribution and the design of the on-board satellite system operation is performed.

A communication protocol between the base station and the satellite will be developed, with the design and implementation of numerous frame formats. Error handling both execution and communication is also managed. The communication process is implemented in accordance with applicable law and in compliance with the *ESA* standard. Tests will also be developed to verify quality simulator designed. With the result, they are going to analyze the data and will present a budget.

Finally, this *TFG* will serve as support for the development of a more complex system. This project will show prospective students the complexity of communication systems, especially satellite communications. It could be used in demonstrations and can be the basis for future university projects. The tool is very visual so any telecommunications student can understand how the system works. On a personal level, it helps me to better understand the operation of communication systems with the use of teaching tools. It also allows me to work with international standards and learn about designing such as complex systems.

2. Estado del arte y comparativa

En esta sección se realiza una comparación entre las soluciones teniendo en cuenta todo lo descrito en los puntos anteriores. El planteamiento del problema es el mostrado en dichos puntos. Los medios empleados en la elaboración de este *TFG* son los presentes en la sección *3.2. Medios empleados* del presente documento y, en general, a lo largo del mismo. El marco legal y el entorno socioeconómico del proyecto están descritos en las secciones *1.2. Regulatory framework* y *1.3. Socioeconomic environment* correspondientemente.

A continuación se realiza una comparativa de los medios empleados (*LabVIEW* como programa y el *NI USRP-2920* como hardware de transmisión radio) usando *SDR*. También se presentan varios proyectos similares que emulan el manejo de datos de a bordo de un satélite. También se muestran programas de simulación parecidos que cumplen con algunas de las funciones que se implementan en este *TFG*.

En la Universidad de Manitoba, Winnipeg (Canadá) se ha llevado a cabo el diseño de un simulador de una estación base cuyo funcionamiento se describe en el artículo "*SATELLITE GROUND STATION EMULATOR: AN ARCHITECTURE AND IMPLEMENTATION PROPOSAL*" [12]. Incluye el diseño de la arquitectura y su implementación, tal y como se muestra en la siguiente figura:

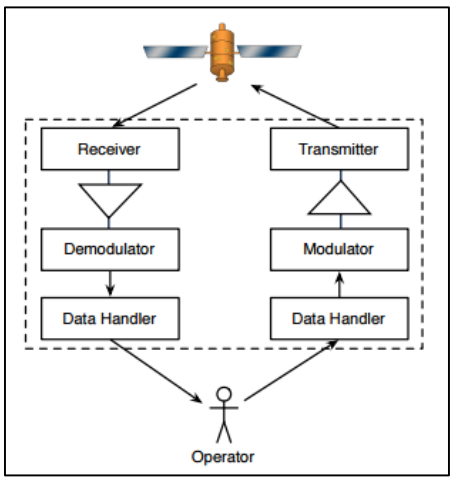


***Fig. 2.1. Diagrama de bloques de una estación base*[5]**

Se simula la transmisión de telemetría desde el satélite a la estación de tierra desde una localización remota con equipamiento físico. Se introduce ruido

5 Referencia [12], Fig. 1. Block diagram of a ground station.

y fallos en la comunicación para que el operario encargado de la estación base detecte anomalías. El propósito de este simulador es el entrenamiento de los operarios en situaciones críticas como es la recepción de la telemetría del satélite y su análisis. Se dice en el artículo que se emplea la simulación como una herramienta de entrenamiento para evitar los problemas que supondría un fallo en el caso de un satélite real.

En cuanto a la simulación del sistema de a bordo se simula el manejo de datos de a bordo (con un sistema de tiempos) y se simulan diferentes reportes acerca del estado del satélite como las baterías, temperatura, comprobaciones del propio satélite, etc… También se tienen en cuenta parámetros de posicionamiento del satélite, baterías…

Se propone una estructura con tres nodos, la estación base, un controlador y un simulador tal y como se muestra a continuación:

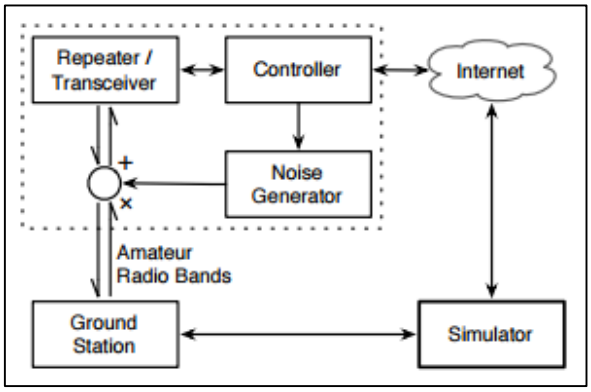


Fig. 2.2. Arquitectura del simulador propuesta[6]

La estación de trabajo se conecta al simulador a través de internet con la generación de ruido. El nodo repetidor está formado por una antena, un transmisor y un receptor. Este módulo recibe la señal radio de la estación base y se la manda al simulador, y viceversa (con la telemetría). En el nodo de simulación también se tienen en cuenta aspectos físicos como la posición orbital del satélite. Se ha programado para el envío de datos con el formato correcto y para la generación de telemetría. La emisión de las señales se realiza en las bandas de frecuencia para radioaficionados.

Como se ha descrito, el simulador anterior está dirigido a la simulación de recepción de telemetría con el fin de ayudar a los operarios a conocer el funcionamiento del sistema y prevenir errores. En este caso, toda la comunicación no se realiza a través del espectro radioeléctrico y se emplea Internet para la comunicación. En el caso de este *TFG* toda la comunicación se

6 Referencia [12], Fig. 3. Proposed emulator architecture.

realiza usando un medio inalámbrico con lo que la simulación se acerca más a la realidad (no se tiene en cuenta la posibilidad de la conexión TCP puesto que no interviene en la comunicación entre el satélite y la estación base). Además, no se lleva a cabo una interacción de telecomandos con el sistema de a bordo como en la solución de este documento. Por el contrario, en la solución del simulador descrito se manejan parámetros físicos como la órbita y se tiene en cuenta una unidad de potencia (simulación de baterías). También realiza test autogenerados por el satélite y los reporta. Estas características no están implementadas en la solución de este documento.

Otra propuesta de arquitectura de implementación es la del Consorcio Nacional para las Telecomunicaciones de Italia de la unidad de investigación de la Universidad de Génova. Está recogida en el artículo "*A Packet-Switching Satellite Emulator: A Proposal about Architecture and Implementation"* [13]. En este artículo se describe la arquitectura e implementación de un emulador de un sistema de a bordo.

Se plantea el uso de *Linux* como sistema operativo y la utilización de herramientas virtuales para modular las interfaces y el mapeo de la capa de red. Se emplean elementos software y hardware que emulan el comportamiento real de un satélite. Además se tiene muy en cuenta la calidad del servicio (*QoS*) y el tiempo de comunicación. La conexión se realiza empleando el protocolo IP en la capa de red. También se emplean *Gateways* y módems para conectar los distintos elementos del sistema cableado.

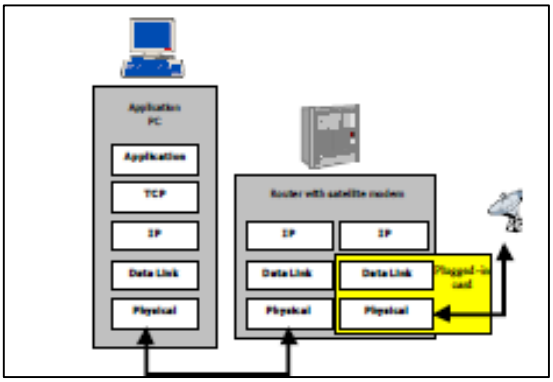


Fig. 2.3. Posible arquitectura de un sistema real y conexión de la tarjeta de red[7]

Se crea un instrumento virtual para la interfaz entre el módem en la estación base y los protocolos de capas superiores. Se emplean adaptadores *Ethernet* para la conexión. El escenario es el mostrado en la siguiente figura.

7 Referecia [13], Fig. 1. Possible architecture of the real system - plugged-in card.

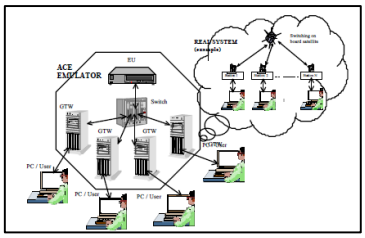


Fig. 2.4. Arquitectura real del emulador y sistema real[8]

En este caso el fin del proyecto es conseguir una comunicación estable usando *Packet-Switching*. Este proyecto va más encaminado a la conexión, a la calidad y al tiempo por lo que supera los resultados obtenidos en este *TFG* en cuanto a transmisión de datos. El medio que emplea es cableado que ofrece indudablemente mejores prestaciones en cuanto a tiempo y en cuanto a calidad de servicio que la transmisión radio. Sin embargo, la solución planteada en este documento es mucho más realista y ofrece una interfaz de usuario más estética y más centrada en la simulación de un sistema de a bordo. El hecho de que se trabaje sobre protocolos de red supone que el sistema sea menos realista.

Otro simulador relacionado con la solución de este *TFG* es la propuesta en el artículo "*A MICROPROCESSOR BASED COMMUNICATIONS SATELLITE EMULATOR"* [14]. Esta solución consiste en la simulación de un sistema de comunicación de un satélite dentro de una constelación de satélites *LEO (Low Earth Orbit),* y en definitiva de un ordenador de a bordo.

Se establecen funciones de comunicación incluyendo recepción y *buffering*. También se tienen en cuenta parámetros como la posición del satélite y el cálculo del resto de satélites de la red.

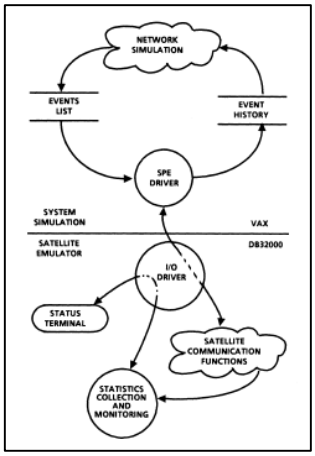


Fig. 2.5. Descripción general del emulador[9]

8 Referencia [13], Fig. 2. Overall Emulator Architecture and Real System.
9 Referencia [14], Figure 2. Overview of Emulation

La interfaz de usuario que emplea es muy vistosa según el artículo y muestra unos gráficos muy sofisticados. Además, como en este *TFG*, se lleva un historial con los paquetes transmitidos y recibidos, así como un registro de operaciones como (lista de planificación, sistema temporal, procesado de datos, eventos…). En este caso, la implementación tiene un componente físico con el desarrollo de los elementos mediante hardware. En cuanto al software se coordinan las funciones de administración de tareas, control del tiempo, memoria, interrupción del servicio y procesado de telecomandos.

En cuanto a la comparativa con la solución de este *TFG*, generalmente la implementación física siempre es mejor que por software. Además, según se expone en el artículo la interfaz de usuario es muy visual. En el caso de la implementación propuesta en este documento también pero está limitada por los elementos de *LabVIEW*. Además, se implementan muchas más funciones que las descritas en la solución anterior. El proyecto anterior es un poco particular porque además de controlar parámetros como la posición, también se tiene en cuenta la posición del resto de satélites dentro de la constelación. En la implementación de este documento no se tienen en cuenta estos parámetros y no se incluye el satélite dentro de ninguna constelación.

Otra solución más es la descrita en el artículo *"On the Emulation of Geostationary Earth Orbit Satellite Systems"* [15]. En él se describe una arquitectura que simula la comunicación con un simulador de satélite *GEO (Geostationary Earth Orbit)* desarollado en el *CNIT Radio Communication Laboratory* en *Politecnico di Torino.*

La implementación realizada emplea el protocolo IP para la transmisión de tramas usando enlaces radio *TDM (Time Division Multiplexing) / TDMA (Time Division Multiple Access)*. La arquitectura del simulador es:

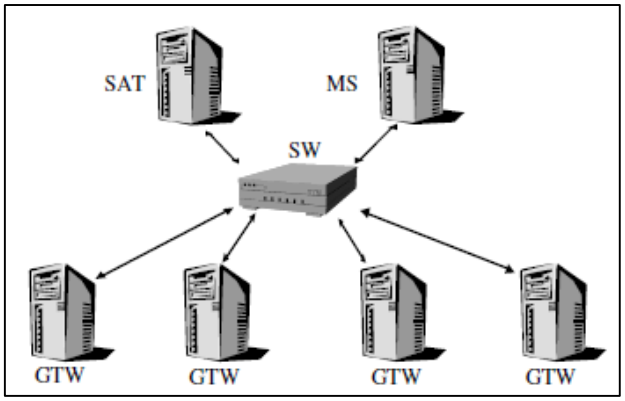


Fig. 2.6. Estructura del emulador[10]

10 Referencia [15], Figure 1 – Emulator Structure

La estructura mostrada presenta varios ordenadores conectados entre sí en una *Fast Ethernet LAN.* Se emplea como protocolo de transporte TCP y *Linux* como sistema operativo. Si comparamos esta solución con la mostrada en este documento, la conexión por cable es mejor pero menos realista. Lo mismo ocurre con el empleo de protocolos de red y transporte. Sin embargo, en la implementación descrita se tienen en cuenta parámetros de la comunicación y se obtiene una mejor calidad de servicio.

En el artículo "*A design of on-board dual-channel data handling method based on two FPGAs"* [16] se presenta una solución con implementación real de un sistema de a bordo de doble canal de manejo de datos empleando dos *FPGAs (Field Programmable Gate Array).* Es empleado en el satélite meteorológico *FY4* y por tanto es una implementación real pero que ha sido anteriormente simulada. Se emplean dos *FPGAs* y permite la comunicación en dos canales.

Fig. 2.7. Sistema de procesado de datos[11]

Sin lugar a dudas una implementación real hardware siempre es mucho mejor que cualquier simulador pero se ha introducido para ilustrar la necesidad de simular el manejo de datos del sistema de a bordo. Este sistema ha sido simulado antes de su realización física.

Otro artículo que describe el manejo de datos del sistema de a bordo de un satélite es "*A New On-Board Data Handling System for Spacecraft Flight Control Simulator"* [17]. En este caso se simula el manejo de datos para el control de vuelo de un satélite.

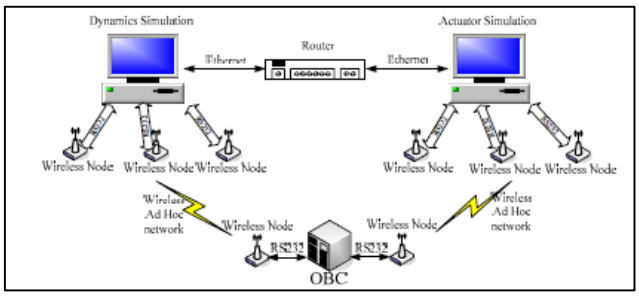


Fig. 2.8. Arquitectura del simulador[12]

11 Referencia [16], Fig. 1. Data-Processing System
12 Referencia [17], Fig. 1. Architecture of the simulator.

Como se puede observar, el simulador consta de seis partes que alternan conexiones por cable y conexiones inalámbricas con el ordenador de a bordo. El simulador se ejecuta en tres ordenadores y la conexión por cable se realiza mediante *Ethernet*. Como sistema operativo se emplea *Linux*.

Comparativamente, esta implementación está muy enfocada al control de vuelo y a técnicas aplicadas a ese mismo ámbito. Además se emplean tanto conexiones por cable como inalámbricas empleando protocolos de transmisión de redes inalámbricas. En el caso de este *TFG* no se emplea ninguno de esos protocolos para la transmisión y no se implementa la simulación de un control de vuelo.

Por último, se presenta la solución del artículo *"Wireless Telecommand and Telemetry System for Satellite"* [18]. Se trata de un sistema de transmisión de telemetría y recepción de telecomandos de manera inalámbrica.

Para lograrlo emplea la flexibilidad de tecnologías como *ZigBee* (conjunto de protocolos de comunicación inalámbrica de alto nivel para su utilización con radiodifusión digital de bajo consumo) y *Bluetooth*. La interfaz de comunicación con el sistema de a bordo es punto a punto. El sistema es el mostrado a continuación:

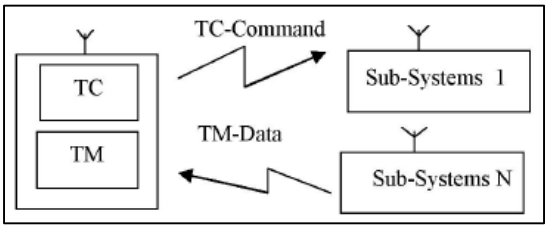


Fig. 2.9. Conexión de la interfaz inalámbrica del sistema de TC-TM con otros subsistemas[13]

En este sistema se plantea la comunicación pero no se implementa un sistema completo como el presentado en este *TFG*. La transmisión en ambos proyectos es inalámbrica pero la empleada por la solución anteriormente presentada está más definida lo que es una ventaja como es el uso de *Bluetooth*.

13 Referencia [18], Fig. 2 Wireless TC-TM sysrem interface to other subsystem in Satellite

3. Aspectos generales del proyecto

Como aclaración referente a la considerable extensión de este documento, esto es debido a que es programación visual y se ha querido exponer con detalle el desarrollo y la funcionadidad del simulador, pero gran parte se encuentra en los anexos, que son de lectura opcional.

3.1. Desarrollo del proyecto

Para la realización del *TFG* se han seguido las siguientes fases:

- **Planteamiento y estudio del problema.** En primer lugar, se lleva a cabo un análisis minucioso del caso de estudio, estableciendo los elementos software y hardware empleados. Del mismo modo se establece el estándar de referencia y el protocolo de comunicación del sistema a grandes rasgos.

- **Establecimiento de los formatos de trama y del funcionamiento del sistema.** Durante esta fase se han analizado los once servicios, entendiendo el funcionamiento y concretando los límites del sistema. Además para cada telecomando y telemetría se establece el formato de trama en concreto. Se constituye el sistema de funcionamiento general del sistema, incluyendo las partes de recepción y transmisión, así como la parte visual del programa.

- **Desarrollo del sistema.** A lo largo de esta fase se ha implementado el simulador de *OBDH* programando todo lo planteado en el punto anterior. Además se han ido realizando pruebas de simulación por módulos de manera independiente sin la parte de comunicación, es decir, se ha probado el correcto funcionamiento de los procesos del sistema. Además, han sido modificados algunos módulos teniendo en cuenta las limitaciones encontradas durante la implementación del simulador.

- **Pruebas.** En esta fase se han empleado los módulos de recepción y transmisión junto con los sistemas implementados para la estación base para comprobar de manera definitiva el funcionamiento del sistema al completo teniendo en cuenta los módulos de comunicación radio.

- **Redacción de la memoria.** Por último, se ha escrito la memoria basándose en la implementación del simulador y documentando todo lo referente al trabajo realizado.

Estas fases están descritas con más detalle en la sección 20 del presente documento.

3.2. Medios empleados

Durante la realización del simulador *OBDH* se han empleado:

- **Elementos software:**
    - **Sistema Operativo.** Se ha utilizado Windows 7 como sistema operativo tanto los ordenadores portátiles como el pc de sobremesa del laboratorio.
    - **Software de Desarrollo de Sistemas NI LabVIEW.** En el ordenador de sobremesa se ha empleado *LabVIEW 2010* (64 bits) mientras que en los portátiles está instalada la versión *LabVIEW 2011* (32 bits). Esta diferencia de versiones hace que los proyectos realizados con la versión 2011 tengan que ser guardados para la versión de 2010.
- **Elementos hardware:**
    - **Ordenadores.** Para la implementación del sistema al completo se ha usado un ordenador portátil ASUS UX32VD con procesador Intel(R) Core(TM) i5-3317u CPU @ 1.70GHz y 4GB de memoria RAM. Además se emplea un adaptador *Gigabyte Ethernet* conectado al puerto USB del portátil para la comunicación con el *USRP*. También se ha empleado otro ordenador portátil para la estación base y el ordenador de sobremesa del laboratorio para las pruebas del satélite.
    - **Dispositivo móvil Android.** Se ha empleado un s*martphone* marca Sony Z3 modelo D6603 con versión de Android 5.1.1. La MAC del terminal ha sido dada de alta en la subred *TSCWLANG* del departamento (se encuentra en la misma subred que el ordenador de sobremesa del laboratorio).
    - **Elementos de comunicación radio.** Se han empleado dos *NI USRP-2920* con frecuencia ajustable desde 50 MHz hasta 2.2 GHz. Junto con él se han usado dos cables *Gigabyte Ethernet* y dos antenas, uno por dispositivo.

3.3. Estructura de la memoria

De manera general, la estructura de la memoria es la mostrada a continuación:

- Introducción, entorno socioeconómico, marco legal, motivación y objetivos del proyecto (sección 1).
- Estado del arte y comparativa con otras soluciones (sección 2).
- Aspectos generales del proyecto como las fases de desarrollo, los medios empleados y esta sección, la descripción de la estructura del presente documento (sección 3).
- Descripción del entorno de programación empleado (sección 4).
- Presentación de la estructura del programa de manera general (sección 5).
- Descripción del módulo de recepción de telecomandos y su procesado (sección 6).
- Establecimiento de manera teórica de la estructura de telecomandos y telemetría (sección 7).
- Descripción de los servicios implementados de manera detallada (secciones 8 a 18).
- Muestra de las pruebas realizadas y conclusiones acerca del funcionamiento general del sistema (sección 19).
- Planteamiento de la planificación del proyecto y presentación de un presupuesto para su desarrollo (sección 20).
- Conclusiones acerca de la solución final del proyecto (sección 21).
- Exposición acerca de posibles líneas de desarrollo de continuación del proyecto (sección 22).
- Al final del documento se incluyen nueve anexos para entender mejor el funcionamiento del sistema.

4. Entorno de LabView

En esta sección se describe el funcionamiento general del sistema y cómo se ha implementado con *LabVIEW* [10], [22], [23].

En primer lugar es necesario conocer el funcionamiento del programa así como los elementos que ofrece y que han sido empleados para la implementación del sistema al completo.

LabVIEW se compone por dos paneles interconectados entre sí. A continuación mostramos en la parte izquierda el *Panel Frontal* (*Front Panel*) en el que se insertan los elementos del sistema tanto de interacción con el usuario como los que son necesarios emplear como variables locales. En la parte derecha se muestra el *Diagrama de Bloques* (*Block Diagram*) en el que se lleva a cabo en sí la programación tanto con elementos que se pueden insertar directamente (constantes, *arrays*…) como con los elementos insertados en el *Panel Frontal*.

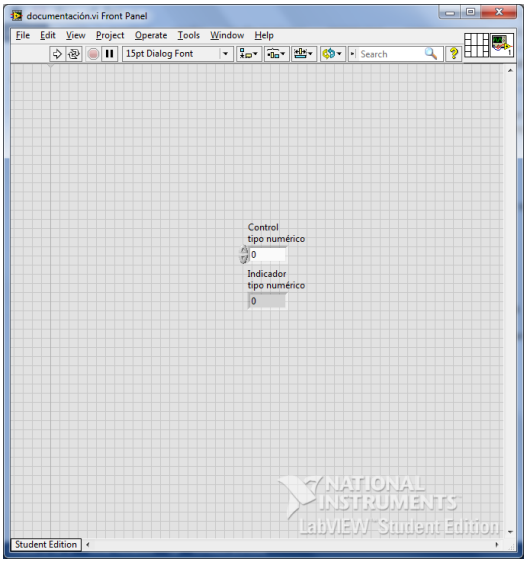


Fig. 4.1. Panel Frontal

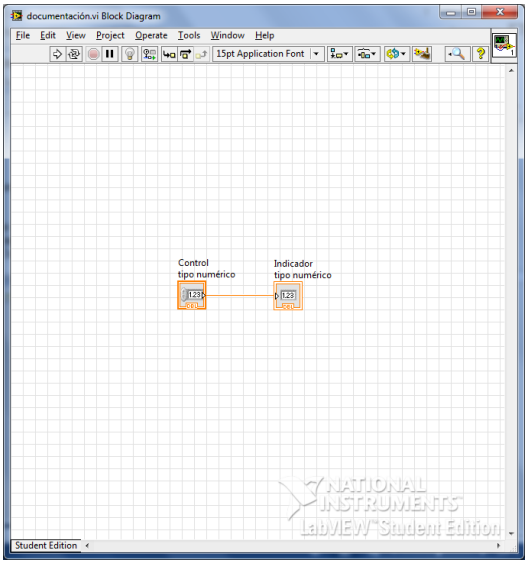


Fig. 4.2. Diagrama de Bloques

En las dos imágenes anteriores se ilustra cómo en el *Panel Frontal* se han insertado un *Controlador Numérico* y un *Indicador Numérico*, y en han aparecido en el *Diagrama de Bloques* donde se han conectado. Si se inserta un número en el controlador en la parte izquierda y se ejecuta el programa en el indicador aparece dicho número.

Toda la información referente al funcionamiento del programa y los elementos más empleados en la realización del proyecto se encuentra en *Anexo A: Entorno de LabVIEW.*

5. Estructura del programa

En el presente capítulo se describe el funcionamiento general del sistema, los bloques que lo componen y la interacción entre los mismos.

5.1 Estructura del proyecto *LabVIEW*

De manera general, la estructura de carpetas y archivos del proyecto es la siguiente:

```
/Heisenberg
  /Controls
      analog_clock
  /Files
      image_intro.txt; image_intro_summary.txt
  /Images
      IID0.jpg; IID1.jpg; IID2.jpg; IID3.jpg; IID4.jpg; logo.jpg
  /Sound
      satellite_init.wav; satellite_reboot.wav
  /subVI
      /Rx
          [archivos de recepción *.vi]
      /Tx*
          [archivos de transmisión *.vi]
```

Fig. 5.1. Estructura de carpetas y archivos del programa

A continuación se describe la estructura de archivos del proyecto de *LabVIEW*. La ventana del explorador de archivos del proyecto es:

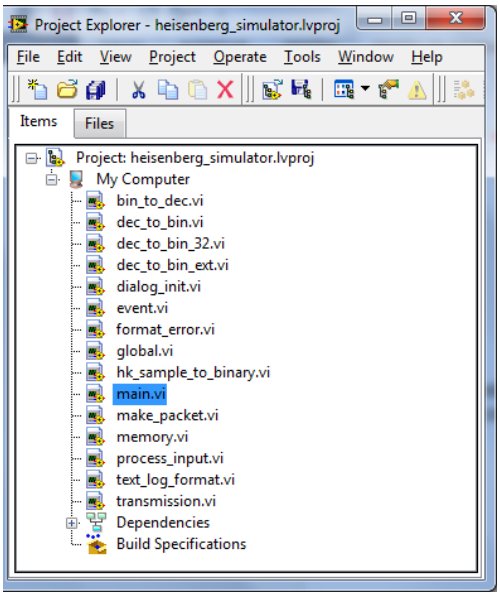


Fig. 5.2. Ventana del explorador del proyecto

Para la implementación del sistema se ha creado un proyecto que contiene catorce instrumentos virtuales (archivos con extensión *".vi"*), los cuales se listan a continuación:

- *bin_to_dec.vi*
- *dec_to_bin.vi*
- *dec_to_bin_32.vi*
- *dec_to_bin_ext.vi*
- *dialog_init.vi*
- *event.vi*
- *format_error.vi*
- *global.vi*
- *main.vi*
- *make_packet.vi*
- *memory.vi*
- *process_input.vi*
- *text_log_format.vi*
- *trasnmission.vi*

Además, para el módulo de transmisión se han incluido ocho instrumentos virtuales en el directorio *"/subVI/Tx",* tal y como se describe en *Anexo F: Módulo de transmisión*. Para el módulo de recepción se ha creado el directorio *"/subVI/Rx",* en el que se incluyen los archivos listados en *Anexo E: Módulo de recepción*. La transmisión y recepción está adaptada de un programa ya creado del foro de *National Instruments* [25].

5.2 Estructura del sistema

La implementación llevada a cabo está pensada de la forma más modular posible. El sistema completo de manera teórica está formado por once servicios. Además, la parte de comunicación está formada por dos módulos, uno de recepción y otro de transmisión. Todo ello se muestra en la siguiente figura:

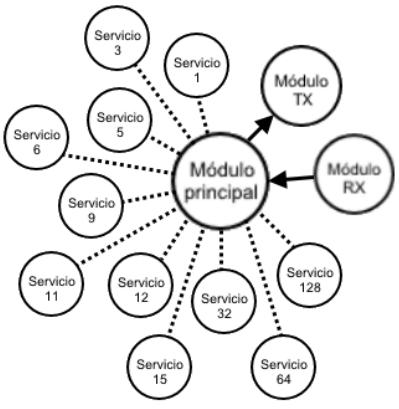


Fig. 5.3. Estructura principal teórica del programa

Servicio	Nombre
1	Verificación TC
3	*Housekeeping* y diagnóstico de datos
5	Eventos
6	Gestión de memoria
9	Gestión del tiempo
11	Gestión del planificador de TC
12	Monitorización de a bordo
15	Test de conexión
32	Instrumento sensor de temperatura
64	Instrumento imágenes
128	Reboot

Tabla 5.1. Identificación de los servicios

Como puede observarse, la mayor parte del sistema se implementa en el módulo principal que comunica con otros módulos dependiendo del grado de independencia que se pueda lograr. Una vez visto el sistema de manera teórica, pasamos a analizar de manera concreta la implementación llevada a cabo. La estructura del módulo principal es la siguiente:

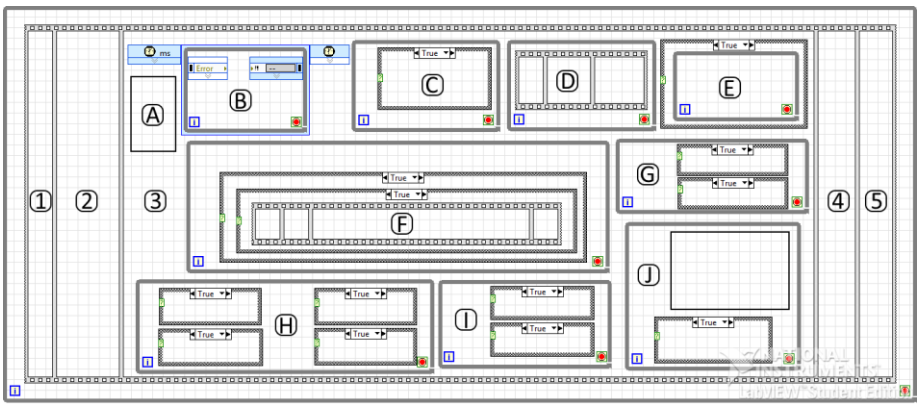


Fig. 5.4. Estructura del programa en LabVIEW

Como se puede observar, todo el programa se engloba dentro de un bucle *while* en el que se ejecuta dentro una estructura secuencial (*Flat Sequence Structure*) con cinco pasos:

1. *Inicialización visual y auditiva.* Se inicia el sistema de manera visual para una mejor experiencia del usuario tal y como se describe en el *Anexo G: Inicialización visual y auditiva*.

2. *Inicialización del sistema. S*e inicializan todos los parámetros del sistema, tanto del módulo principal como las variables globales de los módulos de memoria, tiempo y transmisión. Hasta que este paso no es completado no se ejecuta el programa. Su implementación está descrita en la sección *5.3 Inicialización del sistema* del presente documento.

3. *Sistema principal.* En esta subsecuencia se ejecuta el programa en sí de manera normal. Está constituido por la inicialización del tiempo (A), ocho módulos principales (B, C, D, F, G, H, I y J) y un módulo secundario (E). El funcionamiento de este módulo se explica más adelante de manera general y más en concreto en sus respectivas secciones.

4. *Ejecución de reinicio de manera visual y auditiva.* Esta subsecuencia es ejecutada cuando se ha producido un reinicio del sistema (*reboot*) y su funcionamiento se describe en el capítulo *18. Servicio 128: Reboot*.

Referente a la subsecuencia del *sistema principal*, en el programa principal se encuentra la inicialización del reloj y su correspondiente bucle (*Timed Loop*). Además, hay siete bucles *while* funcionando en paralelo de modo que el programa distribuye los procesos de igual manera. Adicionalmente hay un bucle más en el que se controla la recepción de una conexión TCP opcional. Dentro de cada uno de estos módulos se realiza una función, descritas a continuación:

A. ***Inicialización del reloj***. Su funcionamiento se explica más adelante en el capítulo *12. Servicio 9: Gestión del Tiempo*.

B. ***Reloj.*** Simula el reloj del satélite. Su funcionamiento se explica más adelante en el capítulo *12. Servicio 9: Gestión del Tiempo*.

C. ***Módulo TC Schedule****.* Es el encargado del control y ejecución de los telecomandos del sistema de planificación. Su funcionamiento se explica más adelante en el capítulo *13. Servicio 11: Gestión del planificador de TC*.

D. ***Módulo 1****.* En este bucle se trabaja con los registros y de manera opcional con el envío de la conexión TCP como se explica en el *Anexo I: Conexión TCP y aplicación Android.*

E. ***Módulo de recepción TCP.*** En caso de activarse la opción de conexión TCP se ejecuta este módulo para la recepción de los datos, tal y como se describe en el *Anexo I: Conexión TCP y aplicación Android*.

F. ***Módulo de recepción*****.** Constituye la parte recepción de telecomandos por parte de la estación base. Su funcionamiento se explica más adelante en el capítulo *6. Módulo de recepción y procesado*. Además, dentro de este módulo se incluye el procesado de los telecomandos. Adicionalmente, asociado a la ejecución de los telecomandos, se generan las telemetrías (empleando el módulo de transmisión) en el caso de que sea necesario. Su funcionamiento se explica más adelante en el *Anexo F: Módulo de transmisión*.

G. ***Módulo de generación de datos de instrumentos.*** Esta parte del programa se simulan la toma de fotografías del instrumento de imágenes y la toma de muestras de temperatura del sensor del satélite. Su funcionamiento se explica más adelante en los capítulos *16. Servicio 32: Instrumento sensor de temperatura* y *17. Servicio 64: Instrumento de imágenes*.

H. ***Módulo de housekeeping***. En esta parte del programa se simulan la generación los parámetros de *housekeeping* referentes al microprocesador (frecuencia y temperatura) y al estado de los instrumentos (imagen y sensor de temperatura). Además, este módulo es el encargado de la generación de reportes de los datos de *housekeeping*. Su funcionamiento se explica más adelante en los capítulos *9. Servicio 3: Housekeeping y diagnóstico de datos* y *14. Servicio 12: Monitorización de a bordo.*

I. ***Módulo de generación de reportes de datos de instrumentos.*** En este bucle se lleva a cabo la generación de las telemetrías de los datos obtenidos por los instrumentos (imagen y sensor de temperatura). Su implementación está recogida los capítulos *16. Servicio 32: Instrumento sensor de temperatura* y *17. Servicio 64: Instrumento de imágenes*.

J. ***Módulo 2.*** Es en esta parte del sistema en la que se simulan los eventos del satélite y se activa el reinicio del sistema. Su funcionamiento se explica más adelante en los capítulos *10. Servicio 5: Eventos* y *18. Servicio 128: Reboot.*

Como se ha visto en el apartado anterior, todo lo programado en la pantalla de *Diagrama de Bloques* se muestra de manera visual en el *Panel Frontal*. La pantalla principal muestra el siguiente aspecto:

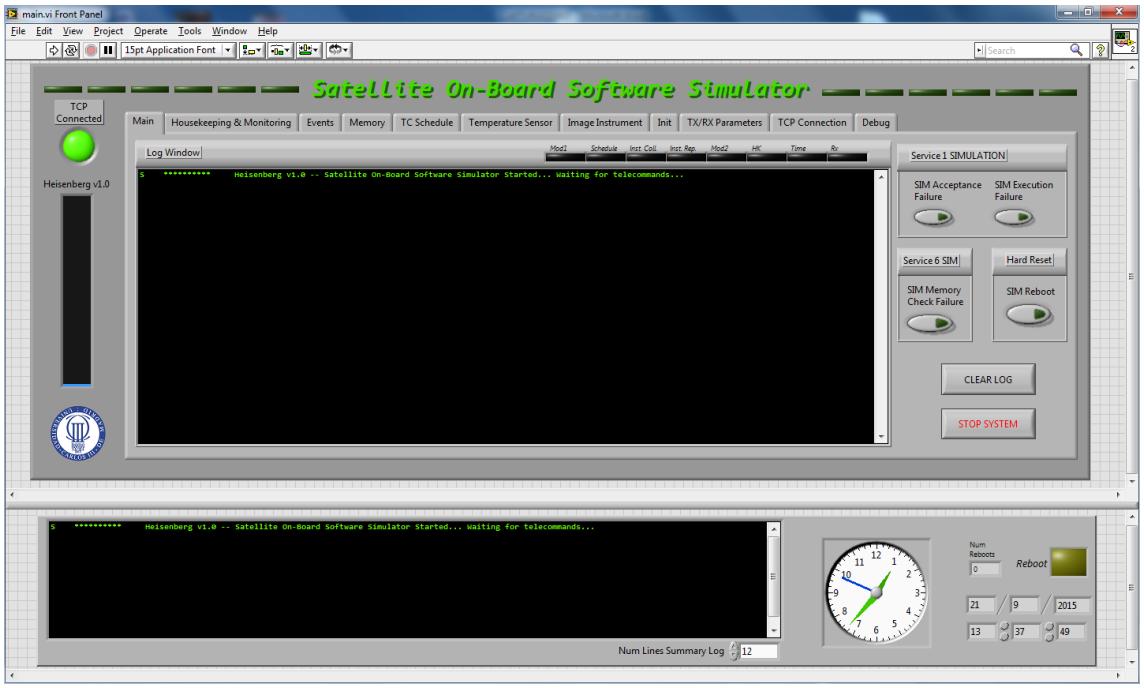


Fig. 5.5. Pantalla principal del programa

Se han establecido once pestañas en las que se agrupan diferentes servicios o funciones que serán explicados en sus respectivas secciones. Son las listadas a continuación.

- *Main*
- *Housekeeping & Monitoring*
- *Events*
- *Memory*
- *TC Schedule*
- *Temperature Sensor*
- *Image Instrument*
- *Init*
- *TX/RX Parameters*
- *TCP Connection*
- *Debug*

En cuanto a la pestaña *Debug*, simplemente se muestran algunas variables locales que son empleadas en el programa pero que no son mostradas porque o bien se tratan de pasos intermedios o bien carecen de importancia de cara al usuario. Además se incluyen sucesivos botones de parada de todos los bucles del sistema.

5.3 Inicialización del sistema

En lo referente al *Panel Frontal* la pestaña *Init* contiene los parámetros del sistema que pueden ser modificados salvo los de recepción y transmisión. Su aspecto es el siguiente:

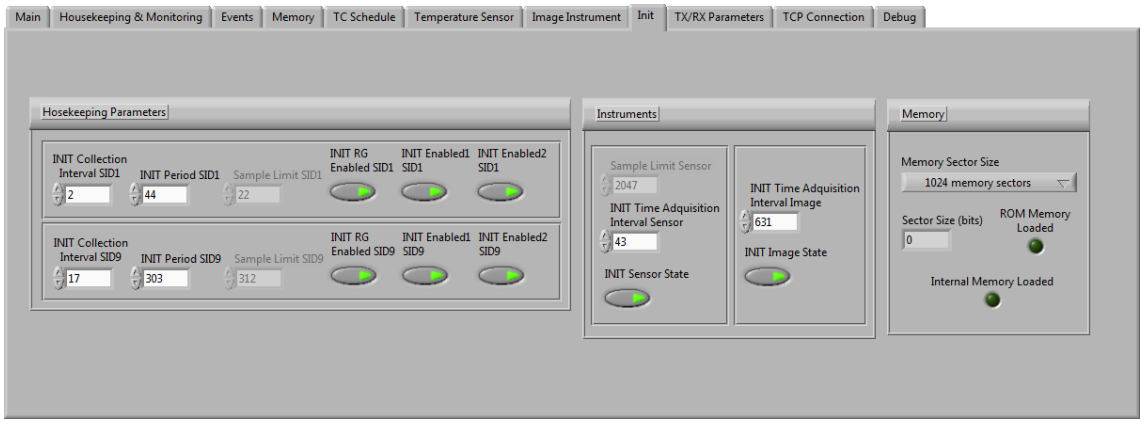


Fig. 5.6. Pestaña Init

Como se puede observar en la figura anterior, en el *Panel Frontal* se pueden inicializar tres bloques:

- ***Housekeeping & Monitoring Parameters.*** Referentes a los bloques de *housekeeping* y monitorización, se pueden inicializar, tanto para *SID1* como para *SID9*, el intervalo de recolección de datos, el período de reporte, el estado de generación de reporte y los estados de monitorización de datos.

- ***Instruments.*** Contiene los parámetros referentes a los instrumentos: el límite de muestras, los tiempos de adquisición de ambos instrumentos así como sus respectivos estados de funcionamiento.

- ***Memory*.** Permite establecer el tamaño de la memoria (tanto los sectores de memoria como el tamaño de los mismos). Además, informa del estado de carga de la memoria.

Relacionado con lo anterior, en la pantalla del *Diagrama de Bloques* se realiza la inicialización del sistema, en la primera subsecuencia como ha descrito, tomando datos del *Panel Frontal*.

La siguiente ilustración muestra la composición del *array* de *housekeeping* y monitorización (*Housekeeping Parameters*) según los datos obtenidos del *Panel Frontal*.

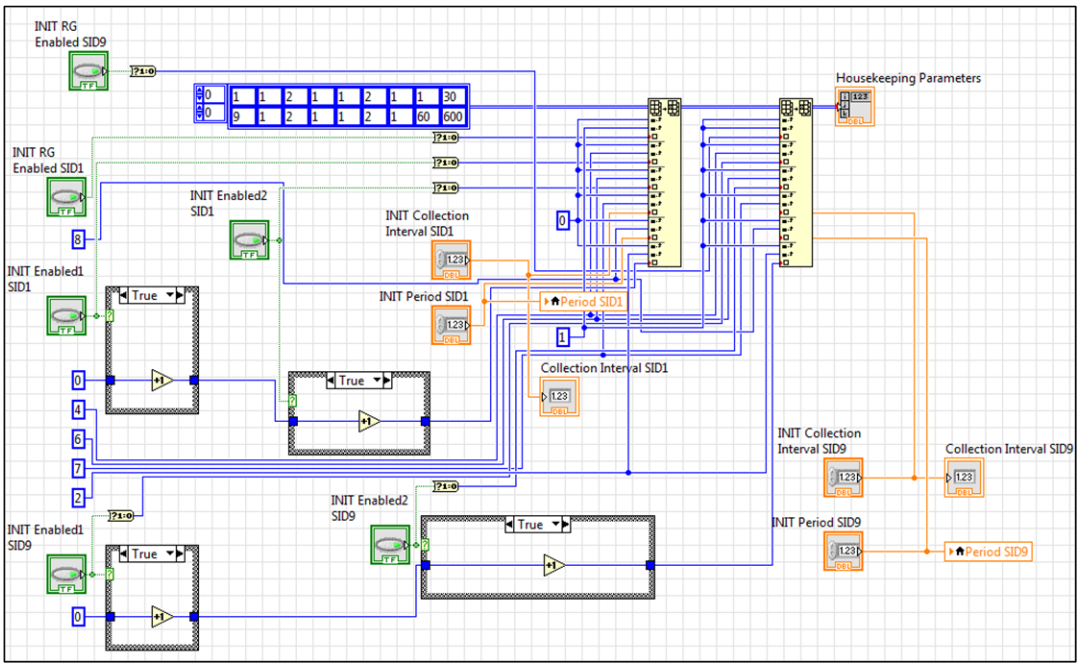


Fig. 5.7. Inicialización de la tabla de housekeeping

Como se puede ver se emplea un *Array Constant* con los valores iniciales (todos los parámetros están activados, tanto reporte como monitorización) y se insertan los obtenidos. En el caso de los botones es necesario hacer la transformación *Boolean To (0,1).* Además, en el caso de los períodos y los intervalos de recolección también se introduce el valor predefinido de inicio en su respectiva variable local.

Para la inicialización de la memoria se ejecuta el siguiente código:

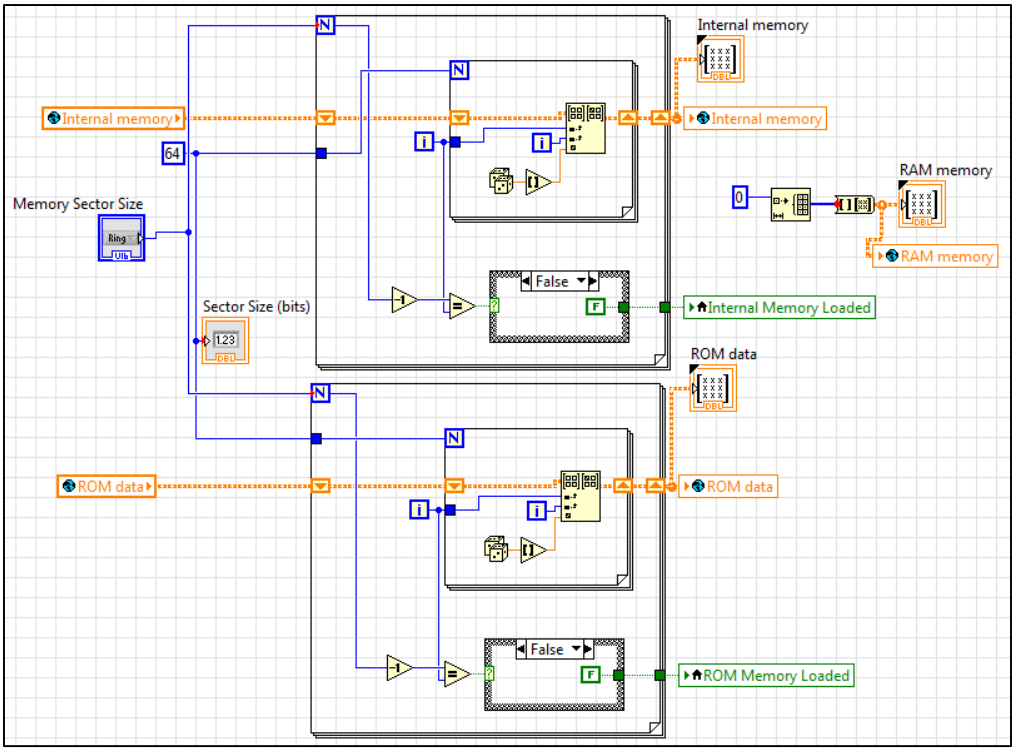


Fig. 5.8. Inicialización de la memoria

Como se observa en la figura anterior, se forman la *memoria interna* (*Internal Memory*) y la *memoria ROM* (*ROM Memory*). Con las dimensiones elegidas en los *Menu Ring* del *Panel Frontal* se recorre un bucle *for* en el que se inserta un número aleatorio devuelto por la función *Random Number (0-1)* y se redondea al más cercano. Posteriormente se inserta en el *array* y se van concatenando valores. De este modo se obtienen dos memorias con el mismo tamaño formadas por sendos '0' y '1'. De este modo, cada vez que se inicie el sistema habrá una memoria diferente lo que le otorga cierto realismo al sistema. Una vez concluida la generación de estas dos memorias se encienden los indicadores de *Internal Memory Loaded* y *ROM Memory Loaded*. Se asignan tanto a la variable global como al indicador local (mostrados en la pestaña *Memory* de *Main Tab*). Para la variable global y su análoga en *main.vi* se inicializan a vacío.

A continuación se muestra el estado inicial de algunas de las variables más importantes:

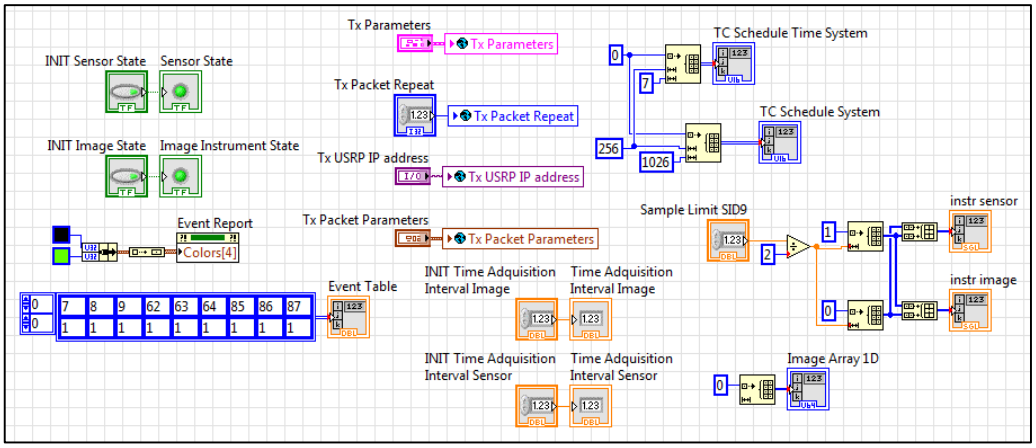


Fig. 5.9. Inicialización de algunas variables

En la figura anterior se muestra la asignación del valor *false* a las variables locales *RID* así como a los botones de simulación de generación de reportes del bloque de eventos. Toman el mismo valor el estado de carga de las memorias, los indicadores de ejecución de reporte del instrumento de imagen y el sensor (*Image Report EXECUTING* y *Sensor Report EXECUTING*) y los de interrupción de la transmisión correspondientes. Toman el valor *true* el estado del módulo de planificación y el estado de generación de reportes de *housekeeping*.

Se activan o desactivan los instrumentos (*Sensor State* e *Image Instrument State*) según el valor preestablecido en el *Panel Frontal,* así como sus respectivos tiempos de adquisición (*Time Adquisition Interval Image/Sensor*). También se inicializan los *arrays* del sistema de planificación. El primero, *TC Schedule System,* se dimensiona con 256 *arrays* unidimensionales con tamaño 1026 cada uno de ellos (equivalente a una matriz de 256 x 1026). El segundo, *TC Schedule Time System,* se asocia con el anterior y contiene el tiempo de ejecución de cada uno de los telecomandos. Más adelante se explicará su funcionamiento.

En adición, se ponen a cero varias variables locales que serán empleadas para el control de inserción de datos y el número de *TC Schedule Orders* (la lista de planificación está inicialmente vacía). También se compone el *array* bidimensional del módulo de eventos (en la primera fila se colocan los *RID* predefinidos y en la segunda el estado de reporte de cada uno de ellos). Se desactivan todos los botones de reporte de eventos (se ponen a falso). Por último, se modifican las variables globales referentes al módulo de transmisión.

Además, para el instrumento sensor de temperatura se inicializa el *array* de igual forma que se hace luego en el servicio tal y como se describe en la sección 16, representada a continuación:

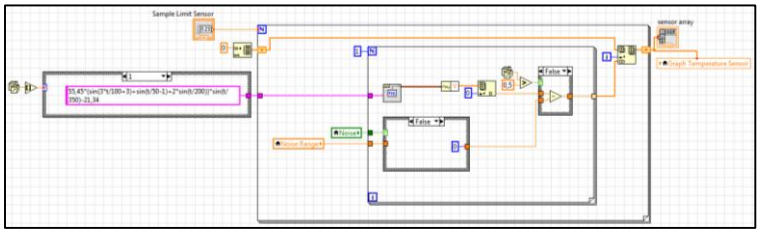


Fig. 5.10. Inicialización del array de temperatura del sensor

Por último, también se realizan una serie de operaciones de carácter estético de cara al usuario referentes a los registros (*Log Window, TC Schedule Log, Event Log* y *Summary Log*) y la barra de progreso. Además se realiza el proceso de conexión TCP en caso de estar activado. Todo este proceso se describe con detalle en el *Anexo I: Conexión TCP y aplicación Android*.

6. Módulo de recepción y procesado

En el presente capítulo se describe el proceso de recepción en el que se incluye el de procesado de los telecomandos recibidos. Dentro de la tercera subsecuencia de la estructura principal se encuadra este bucle de recepción tal y como se ha descrito en el capítulo *5.2 Estructura del sistema*. De manera general, el bucle de recepción se representa como se muestra a continuación:

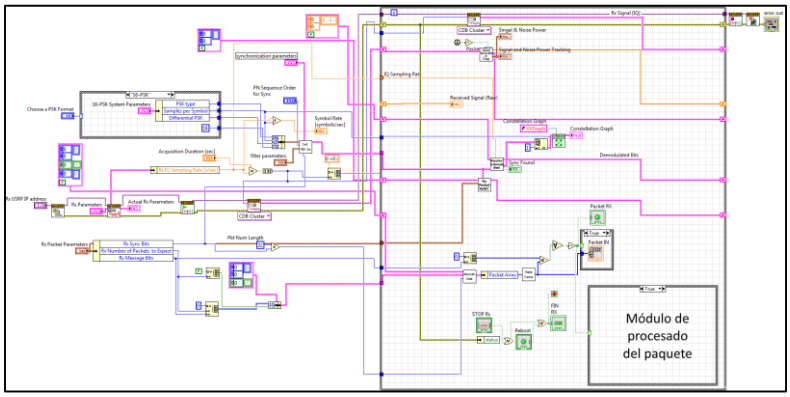


Fig. 6.1. Estructura general del módulo de recepción y procesado

Como se observa, se ejecuta una recepción constante y cuando se recibe un paquete se procesa en el módulo de procesado del paquete. El código empleado para la recepción se explica en el *Anexo E: Módulo de recepción*. A continuación se describe el módulo se procesado de manera más detallada.

El proceso de recepción se ilustra en el siguiente diagrama de flujo:

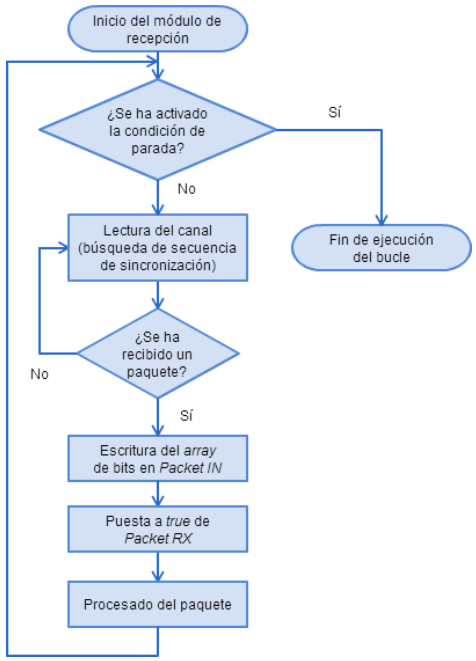


Fig. 6.2. Funcionamiento del módulo de recepción y procesado

Para realizar el procesado del paquete se ejecuta el código ilustrado con el siguiente diagrama de flujo:

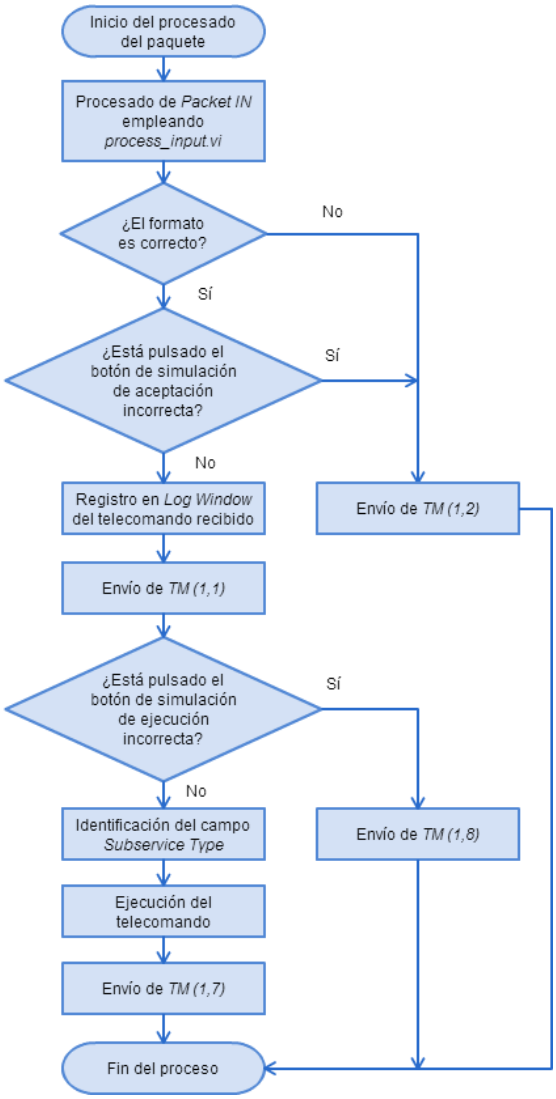


Fig. 6.3. Procesado del paquete

De manera general, la estructura de procesado del paquete es la siguiente:

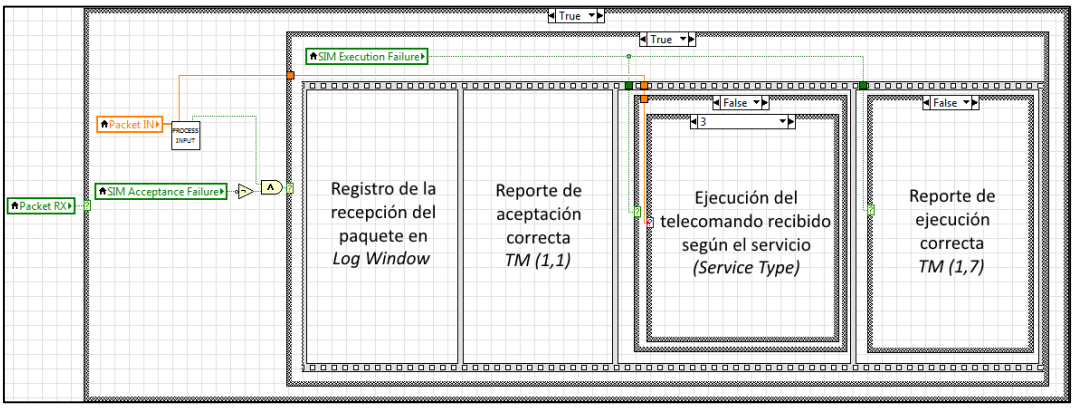


Fig. 6.4. Estructura del código LabVIEW del procesado de paquetes

En primer lugar, cuando se recibe un paquete, el indicador *Packet RX* se pone a *true*. El primer paso es procesar el paquete para obtener todos los campos según la estructura de telecomando descrita en el capítulo *7. Estructura de telecomando y telemetría* con la función *process_input.vi* cuyo funcionamiento está explicado con detalle en el *Anexo D: Procesado de la entrada de datos*. Si el paquete tiene un formato correcto por la salida *Format* se obtiene un booleano con valor *true*. Este valor se hace pasar por una puerta *AND* junto con el valor negado del botón *SIM Acceptance Failure* (simula error de aceptación si está pulsado). En caso de cumplirse ambas condiciones se ejecuta el caso verdadero. Todo lo descrito hasta el momento se representa en la siguiente figura:

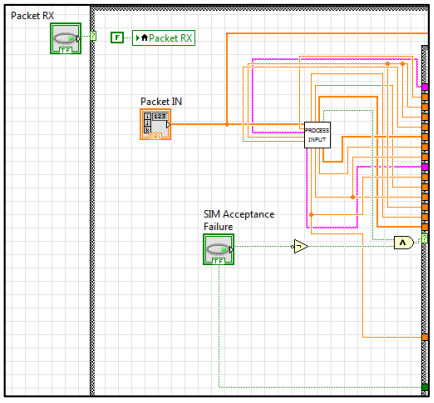


Fig. 6.5. Recepción y procesado del paquete

El siguiente paso es registrar los datos más significativos del telecomando entrante en *Log Window* tal y como se ilustra en la siguiente figura:

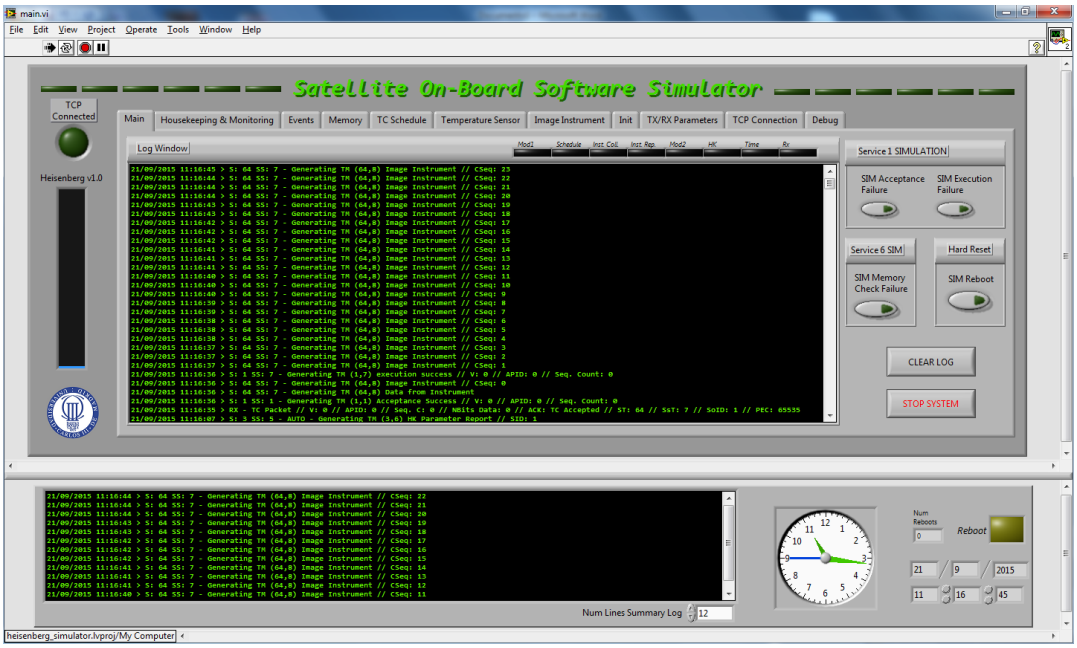


Fig. 6.6. Registros Log Window y Summary Log durante la ejecución del programa

Su funcionamiento es simple. Se van concatenando diferentes constantes de texto con el identificador de los diferentes campos numéricos pasados a *string* con la función *Number To Decimal String.* Luego se concatenan al registro siempre en la parte superior de modo que la última línea siempre se muestra en la parte de arriba. Esta característica se emplea para todos los registros del proyecto y se debe a que así se evita la constante operación de *autoscroll* del *string*. A continuación se ilustra dicho proceso:

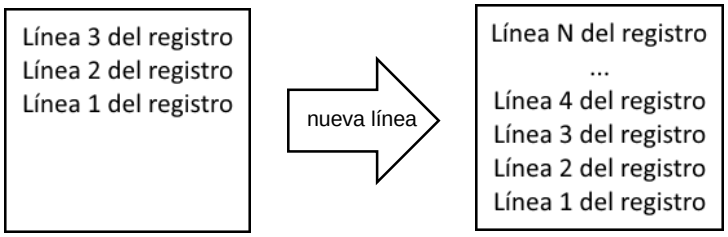


Fig. 6.7. Modo de registro de las operaciones en los log

Las siguientes tres subsecuencias y el caso *false* (aceptación incorrecta) se describen en sus respectivas secciones. Las telemetrías de aceptación y ejecución se explican en el capítulo *8. Servicio 1: Verificación TC*. El bloque de ejecución del telecomando según el tipo de servicio está presente en todas las secciones de los servicios puesto que al recibir un telecomando se comprueba el campo de tipo de servicio y se ejecuta un código de los descritos en las secciones referentes a los servicios implementados.

7. Estructura de telecomando y telemetría

A lo largo de este capítulo se establecen las estructuras de los telecomandos y las telemetrías. En primer lugar establecemos el formato de representación de trama de este documento, las estructuras del telecomando y telemetría de manera global.

El listado con todos los telecomandos y telemetrías se encuentra en el *Anexo B: Lista de telecomandos y telemetrías* del presente documento.

7.1 Estructura de telecomando (TC)

El formato de trama para los telecomandos es el que sigue:

Packet Header (48)						
Packet ID (16)				Packet Sequence Control (16)		Packet Length (16)
Version Number (3)	Packet Type (1)	Data Field Header Flag (1)	APID (11)	Sequence Flags (2)	Sequence Count (14)	
‘000’	‘1’	‘1’	‘X’	‘11’	‘X’	‘X’

Packet Data Field (Variable)		
Data Field Header* (32)	Application Data (variable)	Packet Error Control (16)
	‘X’	‘X’

Tabla 7.1. Estructura general de telecomando

Data Field Header* (32)					
CCSDS Secondary Header Flag (1)	TC Packet PUS Version Number (3)	ACK (4)	Service Type (8)	Service Subtype (8)	Source ID (8)
‘0’	‘001’	‘X00X’	‘X’	‘X’	‘X’

Tabla 7.2. Estructura general del campo Data Field Header para telecomando

Cada telecomando está formado por el *Packet Header* y el *Data Field Header*, cuya estructura se desglosa a continuación:

- ***Packet Header* (48 bits)**
  - ***Packet ID* (16 bits)**
    - *Version Number*. Indica la estructura del paquete tipo telecomando. Siempre toma valor ‘000’.

- *Packet Type*. Distingue entre paquetes tipo telecomando o telemetría. Para paquetes tipo telecomando siempre toma valor ‘1’.
- *Data Field Header Flag*. Indica la presencia o ausencia del campo *Data Field Header*. Todos los paquetes tipo telecomando presentan este campo por lo que siempre toma valor ‘1’.
- *Application Process ID (APID)*. Se corresponde de manera inequívoca con un proceso de aplicación de a bordo destino del paquete tipo telecomando.

- ***Packet Sequence Control*** **(16 bits)**
  - *Sequence Flags*. Se emplea para realizar un seguimiento de una serie de paquetes. Puede tomar los siguientes valores:
    - ‘01’ significa que es el primer paquete de una secuencia;
    - ‘00’ significa que es un paquete intermedio;
    - ‘10’ significa que es el último paquete de una secuencia;
    - ‘11’ significa que es un único paquete individual.

    Como se indica en la especificación, en la implementación del sistema siempre se enviarán paquetes individuales en telecomandos (en ningún caso se mandarán secuencias de paquetes) por lo que se empleará el valor ‘11’.
  - *Sequence Control*. Identifica un paquete tipo telecomando en particular. Servirá para llevar un control numérico de secuencia cuando se manda información en varios paquetes. En el caso de los telecomandos siempre se mandará únicamente un paquete por lo que siempre tomará el valor uno.

- ***Packet Length*****.** Especifica el número de octetos contenidos el campo *Packet Data Field*. Este campo contiene un entero sin signo “C” donde:

  C = (número de octetos del campo *Packet Data Field*) – 1

  Este campo está formado por 16 bits. Por tanto, el rango de valores del campo *Packet Data Field* es de 0 a 65542 octetos. Este valor es de manera teórica, en realidad en recepción se ha establecido el límite de 300 bits de datos.

- ***Packet Data Field*** **(variable)**
  - ***Data Field Header*** **(32 bits)**
    - *CCSDS Secondary Header Flag*. Siempre toma el valor '0' para indicar que el campo *PUS Data Field Header* es "cabecera CCSDS secundaria no definida".
    - *TC Packet PUS Version Number*. Siempre toma el valor '001' porque no se introducen variaciones de versión de la estructura de los telecomandos.
    - *Acknowledgement (ACK)*. Sirve para el asentimiento (aceptación y verificación de ejecución) de paquetes tipo telecomando a la estación de tierra. Se establecen los siguientes estados:
      - 'XXX1' (bit 3 a nivel alto) indica el asentimiento de la aceptación de paquete por el proceso de aplicación;
      - 'XX1X' (bit 2 a nivel alto) indica el asentimiento de comienzo de ejecución;
      - 'X1XX' (bit 1 a nivel alto) indica el asentimiento de progreso de la ejecución;
      - '1XXX' (bit 0 a nivel alto) indica el asentimiento de ejecución completada (cualquiera sea el resultado –éxito o fallo-).

      En esta implementación, siempre toma el valor 'X00X'. Por tanto, únicamente se asienten la aceptación del paquete (al comienzo) y la finalización de la ejecución (al final).

    - *Service Type*. Indica el tipo de servicio al que pertenece el paquete tipo telecomando. Toma los valores definidos en el *Anexo B: Lista de telecomandos y telemetrías*.
    - *Service Subtype*. Identifica de manera inequívoca la petición de servicio del telecomando. Está relacionado con el campo *Service Type* y puede tomar los valores definidos en el *Anexo B: Lista de telecomandos y telemetrías*.
    - *Source ID*. Indica la fuente del paquete tipo telecomando.

  - ***Application Data*** **(variable).** Constituye los datos de las peticiones de servicio del usuario.

  - ***Packet Error Control (PEC).*** Está formado por dos octetos y establece un código de detección de errores para verificar la integridad del paquete. Aunque está presente en la estructura, no se implementa en esta versión.

5.2 Estructura de telemetría (TM)

Los paquetes de telemetría siguen el siguiente formato:

Packet Header (48)						
Packet ID (16)				Packet Sequence Control (16)		Packet Length (16)
Version Number (3)	Type (1)	Data Field Header Flag (1)	APID (11)	Grouping Flags (2)	Sequence Count (14)	
‘000’	‘0’	‘1’	‘X’	‘11’	‘X’	‘X’

Packet Data Field (Variable)		
Data Field Header* (32)	Source Data (variable)	Packet Error Control (16)
	‘X’	‘X’

Tabla 7.3. Estructura general de telemetría

Data Field Header* (32)					
Spare (1)	TM Packet PUS Version Number (3)	Spare (4)	Service Type (8)	Service Subtype (8)	Destination ID (8)
‘0’	‘001’	‘0000’	‘X’	‘X’	‘X’

Tabla 7.4. Estructura general del campo Data Field Header para telemetría

En el caso de loa telemetría cada paquete también está formado por *el Packet Header* y el *Data Field Header*, cuya estructura se desglosa a continuación:

- ***Packet Header* (48 bits)**
  - ***Packet ID* (16 bits)**
    - *Version Number*. Indica la estructura del paquete tipo telecomando. Siempre toma valor ‘000’.
    - *Type*. Distingue entre paquetes tipo telecomando o telemetría. Para paquetes tipo telemetría siempre toma valor ‘0’.
    - *Data Field Header Flag*. Indica la presencia o ausencia del campo *Data Field Header*. Todos los paquetes tipo telemetría presentan este campo por lo que siempre toma valor ‘1’.
    - *Application Process ID (APID)*. Se corresponde de manera inequívoca con un proceso de aplicación de a bordo que es fuente de este paquete.

- ***Packet Sequence Control* (16 bits)**
  - *Grouping Flags*. Se emplea para realizar un seguimiento de una serie de paquetes. Se empleará siempre el valor '11'.
  - *Sequence Control*. Identifica un paquete tipo telecomando en particular. Servirá para llevar un control numérico de secuencia cuando se manda información en varios paquetes.

- ***Packet Length*.** Especifica el número de octetos contenidos el campo *Packet Data Field*. Este campo contiene un entero sin signo "C" donde:

  C = (número de octetos del campo *Packet Data Field*) – 1

  Este campo está formado por 16 bits. Por tanto, el rango de valores del campo *Packet Data Field* es de 0 a 65542 octetos. Este valor es de manera teórica, en realidad en transmisión se ha establecido el límite de 776 bits de datos.

- ***Packet Data Field* (variable)**
  - ***Data Field Header* (32 bits)**
    - *Spare* siempre toma el valor '0' para hacer de relleno y completar un número entero de octetos.
    - *TM Packet PUS Version Number*. Siempre toma el valor '001' porque no se introducen variaciones de versión de la estructura de la telemetría.
    - *Spare* siempre toma el valor '000' para hacer de relleno y completar un número entero de octetos.
    - *Service Type*. Indica el tipo de servicio al que pertenece el paquete de telemetría. Toma los valores definidos en el *Anexo B: Lista de telecomandos y telemetrías*.
    - Service Subtype. indica el tipo de servicio al que pertenece el paquete de telemetría. Toma los valores definidos en el *Anexo B: Lista de telecomandos y telemetrías*.
    - *Destination ID*. Indica el identificador de la estación base destino.

  - ***Source Data* (variable).** Constituye los datos de los reportes de servicio a la estación base.

  - ***Packet Error Control (PEC).*** Formado por dos octetos, establece un código de detección de errores para verificar la integridad del paquete. Sigue el mismo modelo que en TC.

8. Servicio 1: Verificación TC

En el presente capítulo se describe el proceso de asentimiento tanto de aceptación del telecomando como de su ejecución. Para entender mejor el proceso de comunicación a continuación se muestran una serie de diagramas.

Tras la recepción de un telecomando se comprueba que el formato del mismo sea correcto. En caso de serlo se genera un reporte de *aceptación correcta TM (1,1)*. Si tras su ejecución todo se ha realizado de manera correcta se genera una telemetría de *ejecución correcta TM (1,7),* lo cual se muestra en la figura de la izquierda. En caso de que se produzca un error mientras se ejecuta, se reporta con *TM (1,8),* tal y como se muestra en la figura de la derecha.

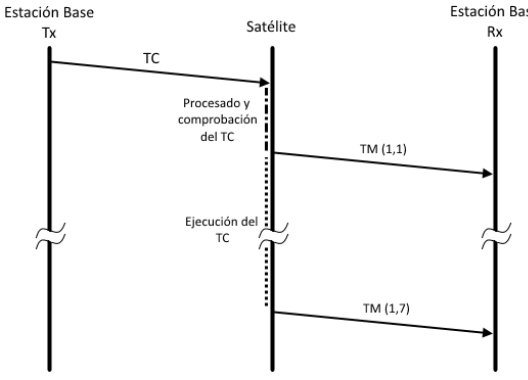


Fig. 8.1. Aceptación y ejecución correctas

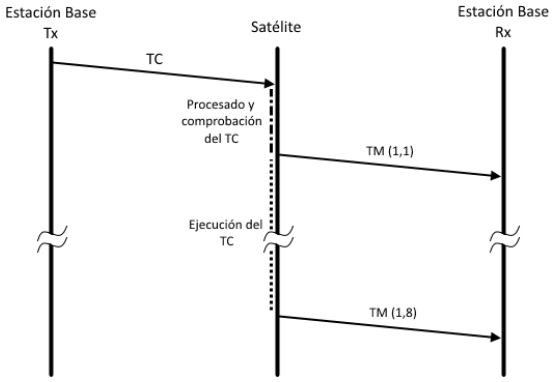


Fig. 8.2. Aceptación correcta y ejecución incorrecta

En el caso de que se compruebe el formato y la integridad del telecomando y se encuentre algún error se genera una telemetría *TM (1,2)* de *aceptación fallida*. En ese caso no se ejecuta nada más, tal y como se ilustra a continuación:

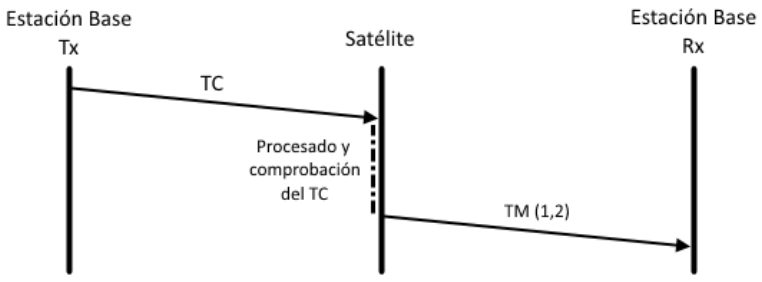


Fig. 8.3. Aceptación incorrecta

8.1 TC acceptance success report (1,1) [TM]

Basado en telemetría (1,1) del estándar [1]. Página 52.

Este comando de telemetría reporta la aceptación de un telecomando cuando la aceptación es llevada a cabo con éxito. Es mandado en cuanto se

verifica la integridad del telecomando recibido y se comprueba que el formato es correcto. Tiene el formato mostrado en la siguiente tabla:

Telecommand Packet ID (16)	Packet Sequence Control (16)
'X'	'X'

Tabla 8.1. Formato de trama de TM (1,1)

El campo *Telecommand Packet ID* está formado por dos octetos y es una copia del campo de mismo nombre de la cabecera del paquete que se quiere asentir. El campo *Packet Sequence Control* también es una copia del campo de mismo nombre de la cabecera del paquete que se quiere asentir.

En lo referente a su implementación en *LabVIEW*, tras la recepción del paquete, la comprobación de que es correcto y la información mostrada en *Log Window*, se ejecuta el siguiente código:

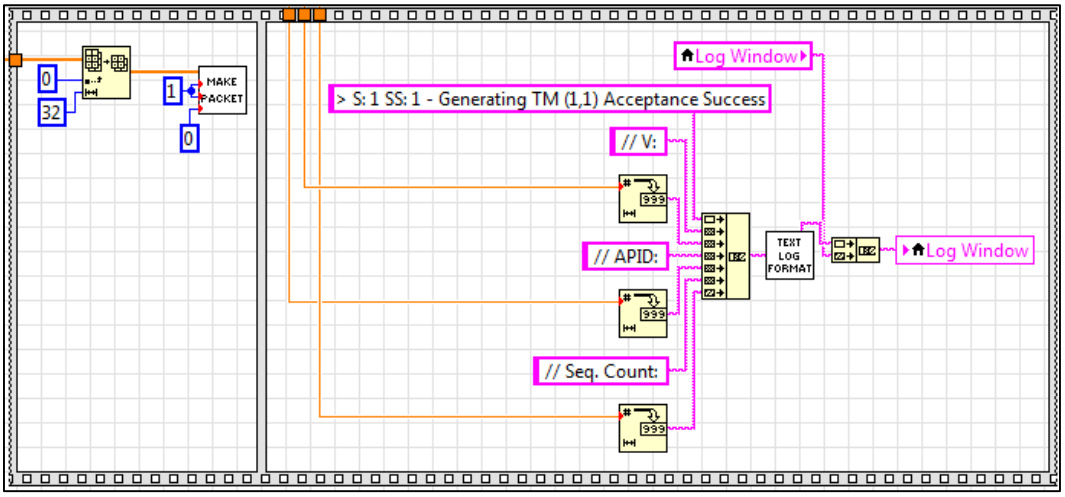


Fig. 8.4. Ejecución de "aceptación correcta"

En primer lugar se envía el paquete de telemetría correspondiente asintiendo al telecomando recibido. Lo único que se realiza es pasar por parámetro a la función *make_packet.vi* los primeros cuatro octetos del telecomando recibidos como *Source Data*, y, *Service Type* y *Subservice Type* con valor 1. En el presente documento el valor de *Sequence Number* siempre es 0 salvo que se diga lo contrario. Luego se transmite.

En segundo lugar, se informa mostrando el envío de la *TM (1,1)*, así como información importante del telecomando recibido (versión, *APID* y *Sequence Count*), cuyos valores numéricos son transformados a *string* con la función *Number To Decimal String*. Se concatenan los *strings* con la función *Concatenate*

Strings y se le pasan como parámetro a la función *text_log_format.vi* que introduce la marca temporal y un salto de línea. Luego se concatena la variable local *Log Window* que almacena todo el *log* anterior con la nueva información y se introduce de nuevo en dicha variable local. Este proceso será empleado constantemente a lo largo del programa.

8.2 TC acceptance failure report (1,2) [TM]

Basado en telemetría (1,2) del estándar [1]. Página 53.

Este comando de telemetría reporta la aceptación de un telecomando cuando se reporta error en la aceptación. Es mandado en cuanto se verifica la integridad del telecomando recibido y se comprueba que el formato es incorrecto. Tiene el formato mostrado en la siguiente tabla:

Telecommand Packet ID (16)	Packet Sequence Control (16)	Code (8)
'X'	'X'	'X'

Tabla 8.2. Formato de trama de TM (1,2)

El campo *Telecommand Packet ID* está formado por dos octetos y es una copia del campo de mismo nombre de la cabecera del paquete que se quiere asentir. El campo *Packet Sequence Control* también es una copia del campo de mismo nombre de la cabecera del paquete que se quiere asentir. Por último, en el campo *Code* (ocho bits) se indica la razón de fallo del telecomando al que refiere durante el proceso de aceptación del mismo.

Los valores que puede tomar el campo *Code* son los siguientes:

- 0 = *APID* ilegal;
- 1 = tamaño del paquete incompleto o inválido;
- 2 = error de *checksum*;
- 3 = tipo de paquete (*Packet Type*) inválido;
- 4 = subtipo de paquete (*Packet Subtype*) inválido;
- 5 = datos de aplicación (*Application Data*) inválido o inconsistente;
- 6 = error en algún otro campo de la cabecera (*Format Header Error*);
- 8 = simulación de error.

En cuanto a la implementación de este reporte, se ha implementado de manera parecida al anterior como se ve en la siguiente figura.

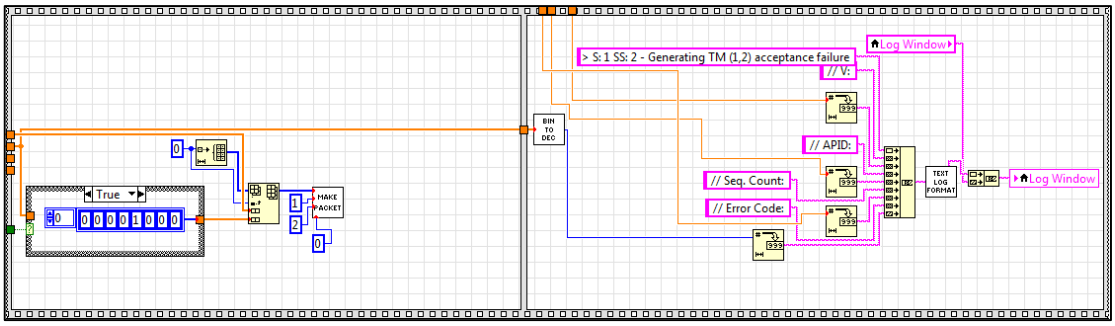


Fig. 8.5. Ejecución de "fallo de aceptación"

Como se observa se inicializa un *array* dinámico en el que se inserta primero los 32 bits iniciales del telecomando que se asiente y un código de error. Dicho código de error vale 8 si ha sido simulado (se ha pulsado el botón *SIM Acceptance Failure* en la pestaña *Main* del *Panel Frontal*). En caso contrario, se inserta el código de error devuelto anteriormente por la función *process_input.vi*. Ese *array* formado se pasa como *Source Data,* junto con *Service Type* y *Subservice Type* con valores 1 y 2 respectivamente, a la función *make_packet.vi*.

Luego se informa del envío de esta telemetría de igual manera que en el apartado anterior pero incluyendo el código de error, es decir, se registra la operación en *Log Window*.

8.3 TC execution success report (1,7) [TM]

Basado en telemetría (1,7) del estándar [1]. Página 54.

Este comando de telemetría reporta la finalización exitosa de la ejecución de un telecomando. Es mandado en cuanto se concluye dicha ejecución. El formato de trama es el mismo que en la telemetría TC acceptance success report (1,1).

La implementación de esta telemetría está asociada a la del siguiente apartado, de modo que ambos comparten una *Case Structure.* Si no se ha pulsado el botón *SIM Execution Failure* en la pestaña *Main* del *Panel Frontal,* se ejecutará el caso *false* y se enviará esta telemetría. En caso de ser pulsado, se simula error por lo que se manda la telemetría del apartado siguiente. Como se puede ver en la siguiente figura se pasa por parámetro a la función *make_packet.vi* los primeros 32 bits del telecomando recibidos como *Source Data*, y, *Service Type* y *Subservice Type* con valores 1 y 7 respectivamente.

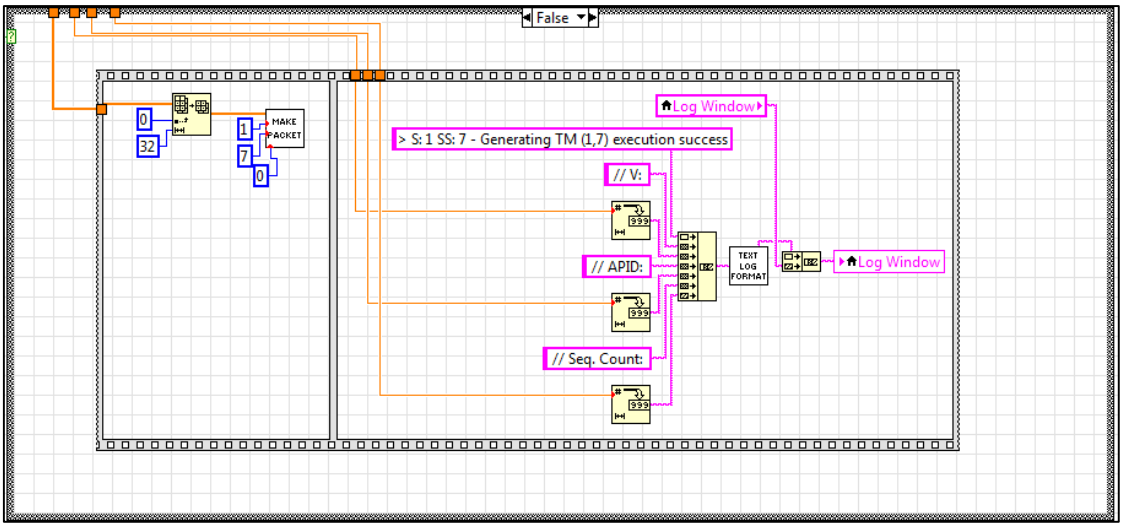


Fig. 8.6. Ejecución de "ejecución correcta"

Tras la realización del paso anterior, se registra dicho envío exactamente del mismo modo que en la telemetría TC acceptance success report (1,1) salvo por el *string* inicial.

8.4 TC execution failure report (1,8) [TM]

Basado en telemetría (1,8) del estándar [1]. Página 54.

Este comando de telemetría reporta la finalización de un telecomando cuando se ha producido algún fallo. Es mandado si se simula un error en ejecución tal y como se explicaba en la sección anterior. El formato de trama es el mismo que el de la telemetría TC acceptance failure report (1,2). En el caso de ejecución siempre se mandará *Code* con valor 8 puesto que siempre será simulado. Esto se debe a que nunca se va a producir un error de ejecución en *LabVIEW*.

Para su implementación, tal y como se ha visto en la sección anterior, se entra en el caso *true*. En primer lugar se despulsa el botón poniendo su variable local a *false* y se genera el reporte pasando por parámetro a la función *make_packet.vi* los primeros cuatro octetos del telecomando recibido, *Service Type* con valor 1 y *Subservice Type* con valor 8. A continuación se registra dicho envío en *Log Window* de manera análoga a la telemetría TC acceptance failure report (1,2) salvo por el *string* inicial. Todo ello se ilustra en la siguiente figura de la siguiente página.

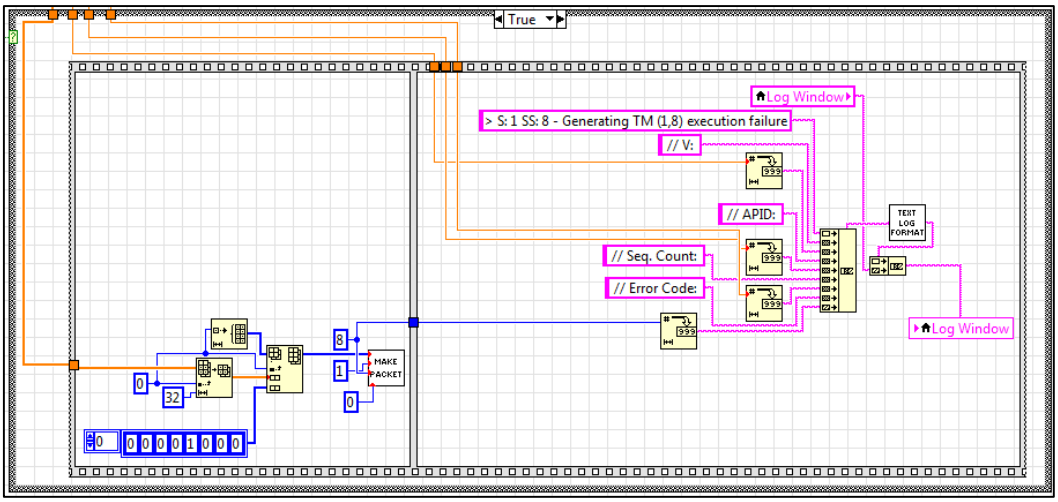


Fig. 8.7. Ejecución de "ejecución incorrecta"

9. Servicio 3: *Housekeeping* y diagnóstico de datos

Este servicio consiste en la toma de datos de una serie de parámetros de *housekeeping* predefinidos que son reportados cada cierto tiempo por el sistema o bien cuando son solicitados por la estación de tierra. Este servicio está muy relacionado con el servicio de monitorización de a bordo (*Servicio 12* del presente documento).

9.1 Funcionamiento del módulo de *Housekeeping*

De manera interna, en el ordenador de a bordo, se han definido dos parámetros de reporte de *housekeeping* en una tabla con la siguiente estructura:

SID (8)	Enabled SID RG (1)	NPAR1 (3)	Param1 (3)	Enabled1 (1)	Param2 (3)	Enabled2 (1)	Collection interval (18)	Period (24)
1	‘0’/’1’	2	1	‘0’/’1’	2	‘0’/’1’	2	44
9	‘0’/’1’	2	1	‘0’/’1’	2	‘0’/’1’	17	303

Tabla 9.1. Tabla de housekeeping y monitorización

Tabla basada en el telecomando (3,1) del estándar [1]. Páginas 64 y 65.

Donde, *Collection Interval* es cada cuánto tiempo toma una muestra (segundos) y *Period* es el período de reporte (segundos).

Los parámetros definidos con *Structure Identification (SID)* con valor 1 están relacionados con los del microprocesador. El *parámetro 1* hace referencia a la temperatura del microprocesador del satélite. El *parámetro 2* se refiere a la frecuencia media de funcionamiento del mismo. Al ser datos de importancia el intervalo de recolección de los datos es de dos segundo y se reportan datos cada 44. Los parámetros definidos con *SID* con valor 9 contienen información sobre el correcto funcionamiento de los instrumentos del satélite de temperatura e imágenes. El *parámetro 1* hace referencia al funcionamiento del sensor de temperatura y el *parámetro 2* al instrumento de imágenes. Toma muestra cada 17 segundos y envía reporte cada 303 segundos.

A continuación se muestra la presencia de este módulo en el *Panel Frontal*:

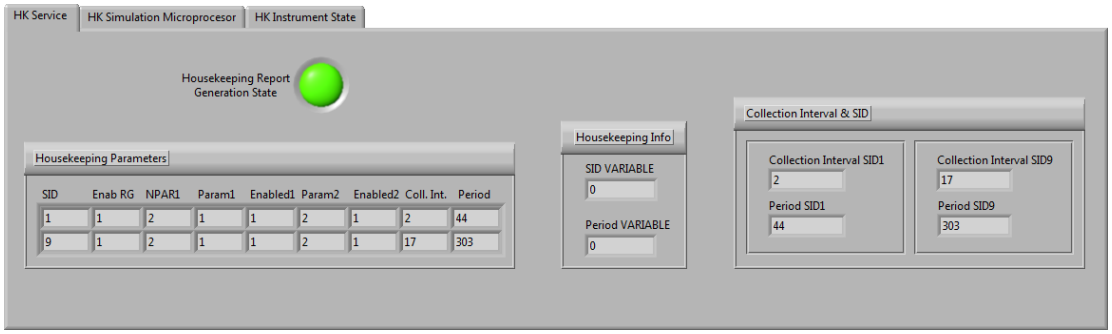


Fig. 9.1. Contenido de la pestaña Housekeeping & Monitoring

En la figura anterior se puede ver la presencia de la tabla de *housekeeping,* un indicador con el estado del módulo, información de los períodos y *SID* variables, así como la identificación en particular del período de recolección de datos y el período de reporte de los dos *SID*.

De manera general, referente a la implementación de este módulo, este servicio está presente tanto en el módulo de recepción y procesado como en el módulo de *housekeeping*. En el primero de ellos se realizan las acciones recibidas por telecomando y en el segundo se realizan dos acciones: recolección de datos de *housekeeping* y reporte (solicitado o automático) de los mismos. La generación de reportes de este módulo es automática, es decir, cada *Period* se genera de manera automática un reporte con los datos de *housekeeping* de cada uno de los *SID*, teniendo cada uno de ellos un tiempo diferente.

A continuación se representa la estructura del bucle de *housekeeping.*

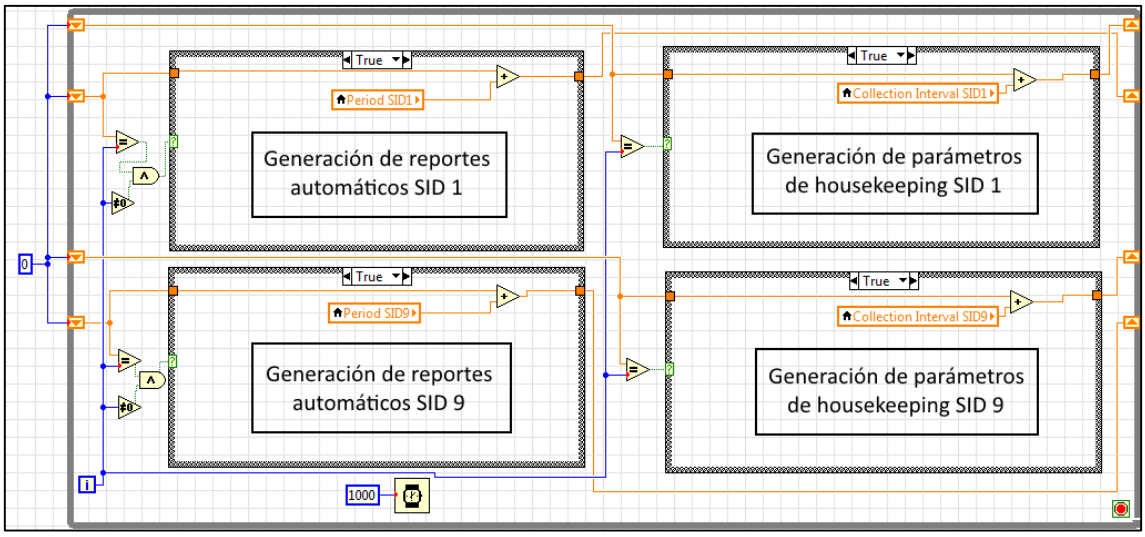


Fig. 9.2. Estructura del módulo de housekeeping

Como se observa, se ejecuta el bucle *while* una vez por segundo y hay cuatro *Case Structure,* dos por cada *SID* referentes a la generación de los datos de *housekeeping* y a la generación de los reportes automáticos cada *Period*. Si la iteración *i* es igual al registro correspondiente se ejecuta el código que está dentro del caso verdadero. En caso contrario, simplemente se mantiene el valor del registro de desplazamiento. En caso de que sea verdadero se suma al registro la variable local correspondiente (*Period SID1*, *Period SID9*, *Collection Interval SID1* y *Collection Interval SID9*), esto es, el número de segundos que deberán pasar hasta la siguiente ejecución de este caso (el tiempo que ha de pasar hasta la adquisición de datos de *housekeeping* o de generación de un reporte). Para la ejecución cada un segundo se emplea el elemento *Wait (ms)* al que se le pasa por parámetro el valor 1000 (milisegundos). Este bloque también es empleado por el *Servicio 12* de monitorización.

A continuación se describe la implementación llevada a cabo para la generación de datos de *housekeeping.*

En el caso del *SID 1* se generan dos funciones con cierta aleatoriedad ya que se permite introducir "ruido aleatorio". Su implementación es la mostrada en la figura ***Fig. 9.18. Generación de datos de housekeeping SID 1***. Para la generación de la función de temperatura se emplean dos funciones: *Sawtooth Waveform.vi* y *Triangle Waveform.vi* a las que se les pasa por parámetro la frecuencia, el *offset,* la amplitud y la fase (valores modificables para cada una de las funciones de manera independiente). Dichos valores pueden ser modificados en tiempo real en la pestaña *Parameters* dentro de la pestaña *HK Simulation Microprocesor* tal y como se muestra a continuación:

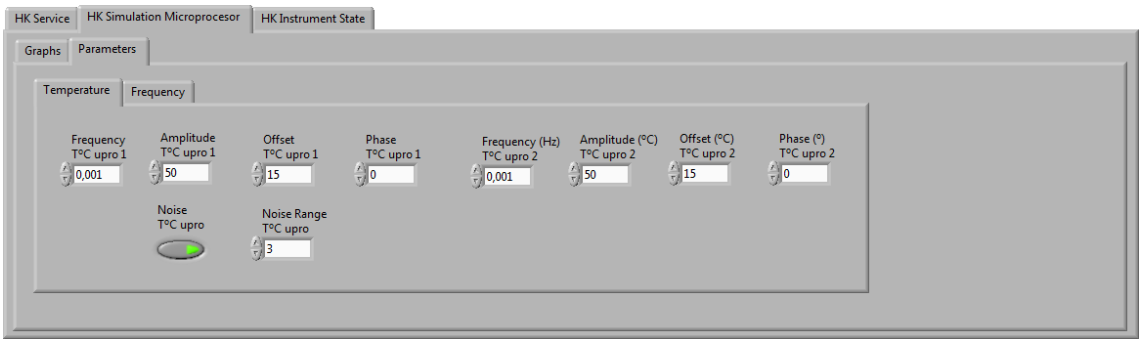


Fig. 9.3. Parámetros de generación de la temperatura

Para el caso de la generación de la función de frecuencia del microprocesador se emplean tres funciones: *Square Waveform.vi, Sine Waveform.vi* y *Sawtooth Waveform.vi*. De igual modo se pueden modificar sus parámetros de generación en la pestaña *Frequency* tal y como se muestra en la siguiente figura.

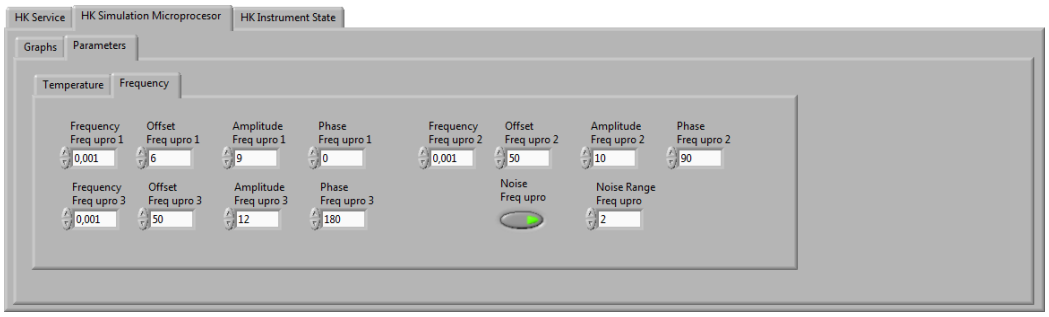


Fig. 9.4. Parámetros de generación de la frecuencia

En ambos casos la generación de la función es idéntica. En primer lugar, se ejecuta un *for* en el que se van indexando los valores que se van a introducir en el *array* del sensor. Se generan las funciones y se suman obteniendo los datos en formato *Waveform(DBL).*

Luego se selecciona el *array* de datos empleando la función *Get Waveforms Components* y con la función *Index Array* se selecciona el primer valor. El siguiente paso es coger ese valor y, en caso de estar pulsado el botón *Noise* se coge un valor aleatorio entre 0 y *Noise Range,* y luego se le suma o resta de manera aleatoria al valor de la función. Si el valor aleatorio es mayor de 0.5 se suma, en caso contrario se resta. Una vez obtenido el valor definitivo se introduce en el *array* correspondiente (*upro temp* o *upro frequency,* que tienen tamaño 2048) en la posición almacenada en la variable local correspondiente (*upro temp pos* o *upro frequency pos*). En el caso de que dicha posición llegue al límite establecido se inicializa de nuevo a cero de manera que se van sobrescribiendo los valores.

En el momento de insertar el nuevo valor en el *array* se comprueba que en la tabla de *housekeeping* en las posiciones 4 y 6 del *array* en la fila cero (que se corresponde con *SID 1*) el valor de generación de parámetros de *housekeeping* valga uno. En caso contrario no se inserta el nuevo valor para ese parámetro.

Luego se representan las funciones en sus respectivos gráficos, que pueden ser vistos en la pestaña *Graphs*, como se muestra en la siguiente figura:

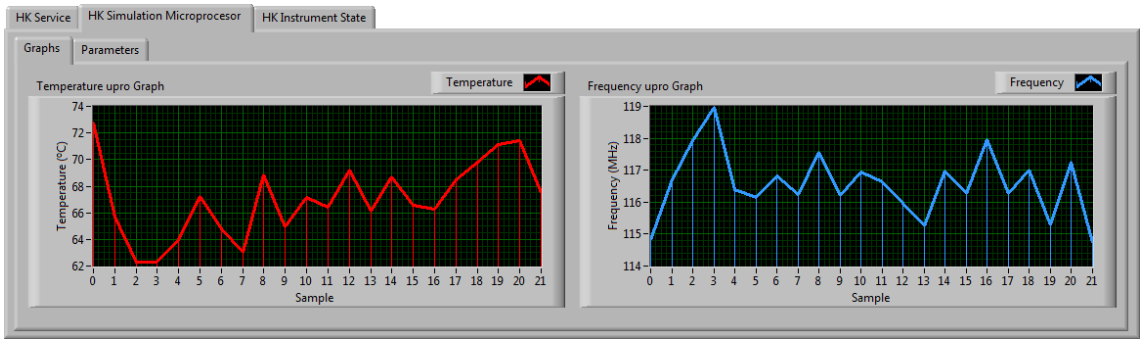


Fig. 9.5. Muestra de las gráficas de los parámetros de temperatura y frecuencia

Para el caso del *SID 9* se realiza algo similar salvo porque no se genera ninguna función sino que se mira el estado de los dos instrumentos (sensor de temperatura e imagen). En caso de estar activado en el momento de tomar la muestra se introduce un uno en el *array* correspondiente (*instr image* o *instr sensor*) y en caso contrario un cero. En ambos casos se introducen en la posición de su respectiva variable local (*instr image pos* o *instr sensor pos*), de modo que si se llega la última posición del *array* (mismo tamaño de *arrays* que para el *SID 1*) se reinicia a cero por lo que se sobrescriben las muestras. Para pasar de *booleano* a *numeric* se emplea la función *Boolean To (0,1)* y ara insertar en el *array* se emplea *Insert Into Array*. Todo el código descrito es el mostrado a continuación:

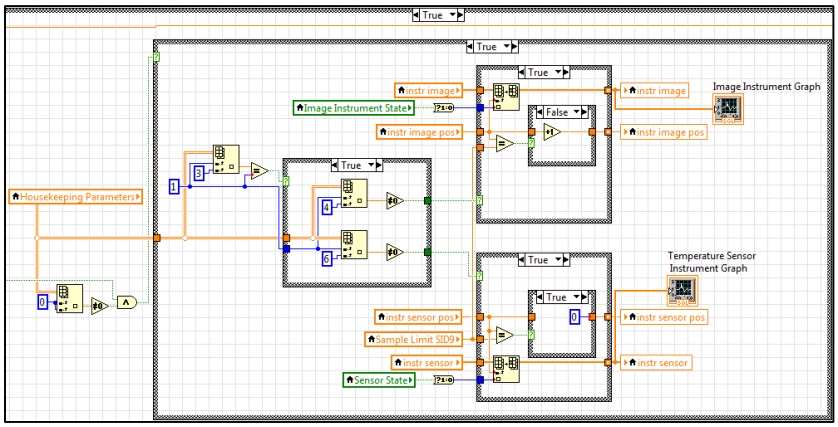


Fig. 9.6. Generación de parámetros de SID 9

Luego se representan las funciones en sus respectivas gráficas dentro de la pestaña *HK Instrument State* tal y como se representa en la siguiente figura:

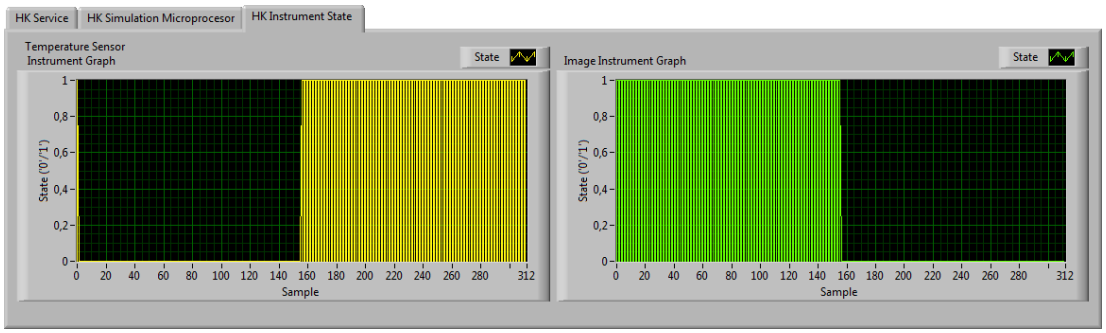


Fig. 9.7. Muestra de las gráficas del estado de los instrumentos

A menos que se diga lo contrario todo lo descrito en las siguientes el código descrito se ejecuta en el módulo de recepción y procesado.

9.2 Clear Housekeeping Parameter Report definitions (3,1) [TC]

Basado en el telecomando (3,3) del estándar [1]. Página 65.

Se mandan como telecomandos de peticiones de borrado de las definiciones establecidas en la tabla de parámetros de *housekeeping* predefinidos del apartado anterior. Una vez borrado, se elimina completamente de la memoria. Se establece la opción de borrar una *SID*. La estructura es la siguiente:

SID (8)
1 ó 9

Tabla 9.2. Formato de trama de TC (3,1)

El campo *SID* consta de ocho bits y establece el parámetro de reporte de *housekeeping* que debe ser borrado. En la implementación llevada a cabo sólo puede ser borrado un *SID* por telecomando recibido.

Para implementarlo se comprueba el subservicio y en caso de valer uno se ejecuta el código de la siguiente figura en el que simplemente se ponen a cero todos los campos de la variable local *Housekeeping Parameters*.

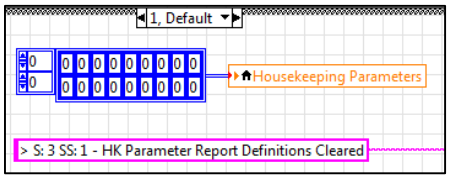


Fig. 9.8. Ejecución de borrado de la tabla de housekeeping

Luego se registra la operación en el *string Log Window* introduciendo en la parte superior siempre el nuevo texto formateado con la función *text_log_format.vi* y concatenado el texto anterior de la variable local. Este proceso será el empleado en el resto de servicios del presente documento para registrar las operaciones. El código ejecutado es el que se muestra en la siguiente figura:

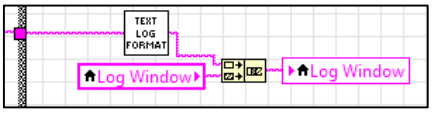


Fig. 9.9. Registro en Log Window de las operaciones

9.3 Enable Housekeeping Parameter Report definitions (3,3) [TC]

Basado en el telecomando (3,5) del estándar [1]. Página 65.

Este telecomando activa el reporte de datos de *housekeeping* de un determinado *SID*. Tiene el mismo formato que el telecomando Clear Housekeeping Parameter Report definitions (3,1).

Para lograr el funcionamiento descrito se ha implementado el código de la siguiente ilustración.

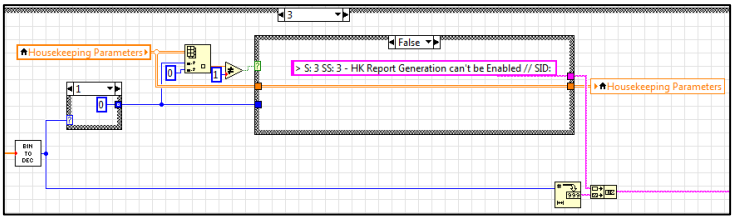


Fig. 9.10. Activación del reporte de parámetros de housekeeping

Como se puede observar en primer lugar se comprueba en el *Case* del bucle de recepción y procesado que su valor sea tres. En ese caso se cogen los datos de aplicación del telecomando y se pasan a decimal con la función *bin_to_dec.vi*, con lo que se obtiene el *SID*. Este valor se le pasa por parámetro a un *Case* del que se extrae la fila dentro de la variable local *Housekeeping Parameters* correspondiente al *SID* (si el *SID* es el uno se mira la fila cero, si el *SID* es el nueve se mira la fila uno). En caso de que no esté a uno se pone el campo *Enabled SID RG* a nivel alto (‘1’). Luego se registra la operación mostrando el *SID* en cuestión (se transforma de *numeric* a *string* con la función *Numeric To Decimal String*). En caso *false* simplemente se ejecuta lo siguiente:

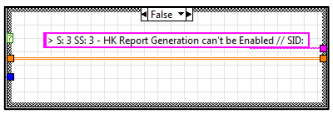


Fig. 9.11. Case false referente a Fig. 9.10

9.4 Disable Housekeeping Parameter Report definitions (3,4) [TC]

Basado en el telecomando (3,6) del estándar [1]. Página 65.

Este telecomando desactiva el reporte de datos de *housekeeping* de un determinado *SID*. Tiene el mismo formato que el telecomando Clear Housekeeping Parameter Report definitions (3,1).

Para la implementación de este telecomando se ha implementado un código exactamente igual al anterior pero poniendo el campo *Enabled SID RG* a nivel bajo (‘0’) y cambiando los mensajes de registro, tal y como se muestra a continuación:

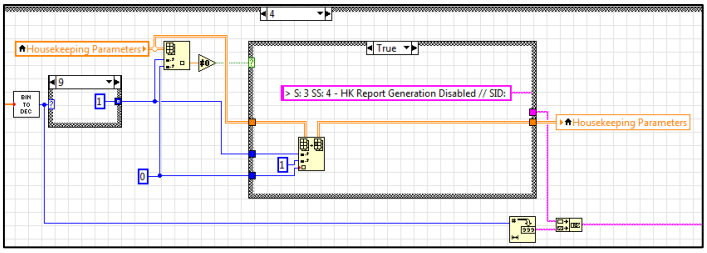


Fig. 9.12. Desactivación del reporte de parámetros de housekeeping

9.5 Request Housekeeping Parameter Report generation (3,5) [TC]

Basado en el telecomando (3,6) del estándar [1]. Página 65.

Este telecomando sirve para pedir un reporte con la recolección de datos de *housekeeping* de un determinado *SID*. Tiene el mismo formato que el telecomando Clear Housekeeping Parameter Report definitions (3,1).

9.6 Housekeeping Parameter Report (3,6) [TM]

Basado en telemetría (3,25) del estándar [1]. Páginas 69 y 70.

Este paquete de telemetría envía la recolección de datos de *housekeeping* de un determinado *SID*. Puede ser generado por el sistema cada cierto tiempo de reporte o por solicitud del telecomando Request Housekeeping Parameter Report generation (3,5). Tiene el formato de la siguiente tabla:

HK Parameter Report Header (variable)			Parameter Data (múltiplo de 8 bits)		
			repetido NPAR1 veces		
SID (8)	NPAR1 (3)	Spare (variable)	ParamN (3)	ParamN Length (16)	Data ParamN (múltiplo de 8 bits)
1 ó 9	0 ó 1 ó 2	‘0’	1 ó 2	‘X’	‘X’

Tabla 9.3. Formato de trama de TM (3,6)

El campo *SID* está formado por ocho bits e identifica a los datos que reportados desde el satélite. El campo *NPAR1* establece el número de

parámetros de los que se manda información de reporte. *Spare* simplemente establece los bits de relleno (definidos a nivel bajo) para completar el paquete. Según el número de parámetros que se envían tiene una longitud:

- Si *NPAR1* es 0, se introducen 5 bits de relleno;
- Si *NPAR1* es 1, se introducen 2 bits de relleno;
- Si *NPAR1* es 2, se introducen 7 bits de relleno.

El campo *Parameter Data* está formado por un múltiplo de 8 bits y contiene:

- *ParamN Length*. Es el número de muestras de *ParamN*. En esta implementación siempre van a ser 2047 muestras de *Param1* y/o 2047 muestras de *Param2*.

- Data ParamN. Contiene los datos de *housekeeping*. Según el tipo de datos se ha establecido para cada *SID* un formato diferente teniendo en cuenta la naturaleza de los mismos. Según el *SID*:

  - *SID 1*. Cada muestra se representa de la siguiente manera:

Decimal Sample Part (10)	Fractional Sample Part (4)
'X'	'X'

Tabla 9.4. Representación de muestra de SID 1

La parte entera de la muestra se almacena en el campo *Decimal Sample Part* y la parte decimal es representada con *Fractional Sample Part*.

  - *SID 9*. Cada muestra se representa como se muestra en la siguiente tabla:

State (1)
'0'/'1'

Tabla 9.5. Representación de muestra de SID 9

Se ha establecido que si en el momento en que se toma la muestra el instrumento está funcionando se inserta un '1'. En caso contrario, cuando el instrumento no está en funcionamiento, se coge como valor un '0'.

Para la implementación de este subservicio se realiza lo siguiente. Se pueden generar reportes automáticos dentro del bucle de housekeeping o bien ser solicitados mediante el presente telecomando y generarse el reporte correspondiente. En primer lugar se va describir el segundo caso, ilustrado en la figura ***Fig. 9.17. Generación de reportes de housekeeping***.

Una vez comprobado el valor del subservicio se ejecuta el código del *Case*, en el que hay una estructura secuencial en la que primero se genera el *array* de datos que se van a transmitir y luego se mandan con *make_packet.vi* y se registra la operación. En la primera subsecuencia se identifica el *SID* solicitado pasándolo a decimal. Luego se conectan los *arrays* de datos mediante el *Case* para obtener la fila correspondiente del *array*. El siguiente paso es concatenar todos los datos del reporte: *SID* (el solicitado), *NPAR1* (parámetros de los que se reporta información y que puede ser 0, 1 ó 2), *Spare* (*array* de ceros con tamaño variable dependiendo del valor de *NPAR1*) y luego tantos *Parameter Data* como valga *NPAR1*. En el caso de *SID 1* además se deberá cambiar el formato tomando el descrito anteriormente en representación binaria para lo que se emplea la función *hk_sample_to_binary.vi* descrita en el *Anexo C: Funciones de conversión*.

Para la generación de reportes automáticos se genera la misma telemetría por lo que es proceso es casi idéntico salvo porque se genera de manera automática por lo que no se identifica el *SID* (no es solicitado explícitamente). Su implementación se encuentra dentro del bucle de housekeeping y se muestra en las siguientes dos figuras.

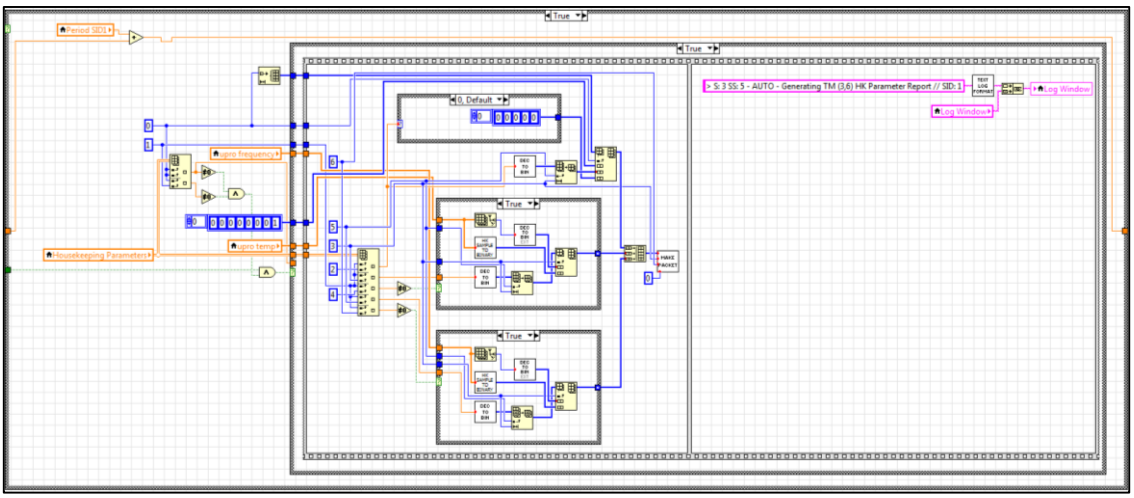


Fig. 9.13. Generación de reportes automáticos de SID 1

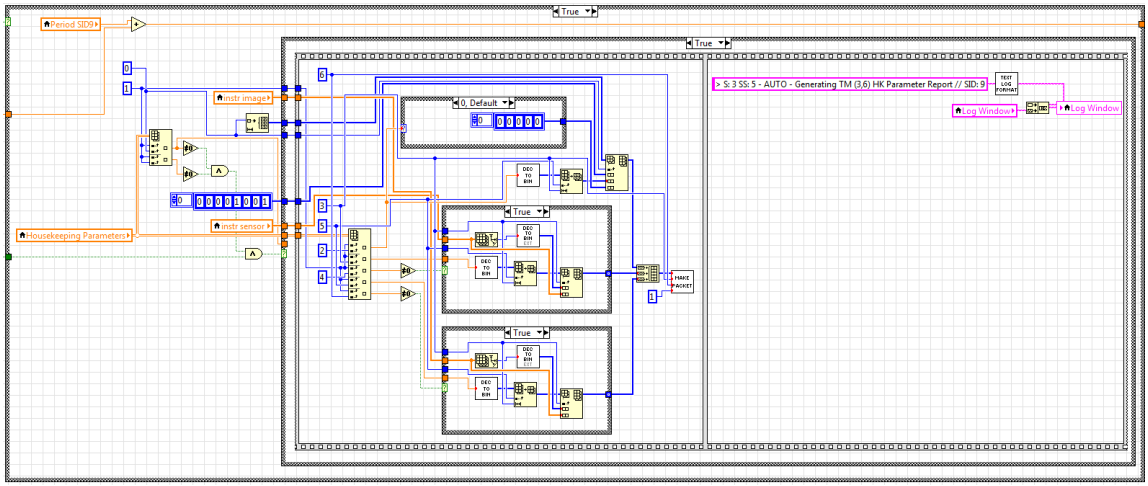


Fig. 9.14. Generación de reportes automáticos de SID 9

9.7 Update Housekeeping Report generation period (3,7) [TC]

Con este telecomando se actualiza el período de reporte de datos de *housekeeping* de un determinado *SID* a partir de uno recibido anteriormente con el telecomando Define Housekeeping Report interval (3,8). No tiene datos de aplicación.

9.8 Housekeeping report interval updated (3,9) [TM]

Con esta telemetría se informa acerca de la actualización del período de reporte de datos de *housekeeping* de un determinado *SID*. Es la respuesta al telecomando Update Housekeeping Report generation period (3,7). Tiene el siguiente formato:

SID (8)	Period (24)
'X'	'X'

Tabla 9.6. Formato de trama de TM (3,9)

El campo *SID* está formado por ocho bits e indica que su período de reporte (*Period*) de *housekeeping* ha sido actualizado. Este último campo está formado por 24 bits de modo que el rango de períodos es el siguiente de 0 a 16777215 segundos (4660.2275 horas; 194.18 días).

Para la implementación de este subservicio y su correspondiente telemetría se ejecuta el código de la figura de la siguiente página.

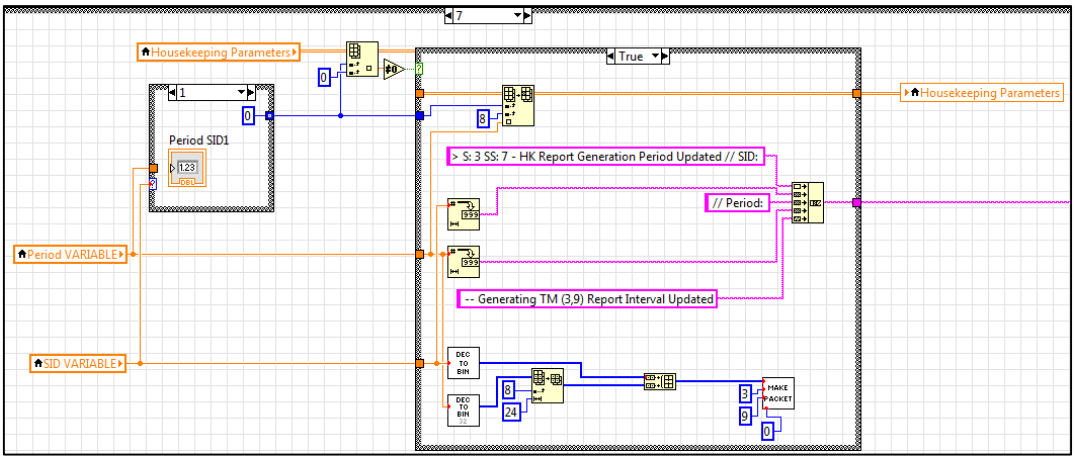


Fig. 9.15. Ejecución de TC (3,7)

Como se puede observar, se cogen el período y el *SID* variables recibidos anteriormente y se modifica el indicador correspondiente al *SID* en cuestión dentro del *Case Structure*. Luego se comprueba que no se haya borrado la tabla de *housekeeping* como en casos anteriores, en cuyo caso se modifica la columna de *Period* de dicha tabla con el nuevo valor. Por último se genera la telemetría insertando en representación binaria el *SID* del que se actualiza el período de reporte y el propio período. Para lograrlo se emplean las funciones *dec_to_bin.vi, dec_to_bin_32.vi* y *Array Subset* (cogiendo la parte menos significativa del *array*). Luego se registra la operación en *Log Window*.

9.9 Define Housekeeping report interval (3,8) [TC]

Con este telecomando se modifica el período de recolección de datos de *housekeeping* de un determinado *SID*. Se guarda en memoria hasta que se recibe el telecomando anterior Update Housekeeping Report generation period (3,7) -hasta entonces no se actualiza-. Tiene el mismo formato que la telemetría Housekeeping report interval updated (3,9).

Para la implementación de este telecomando en primer lugar se extrae con la función *Array Subset* el *SID* (primer octeto) y se comprueba que sea un valor válido (uno o nueve) en cuyo caso se ejecuta el caso verdadero del *Case*. Primero se actualiza el *SID VARIABLE* y luego se extraen los siguientes 24 bits correspondientes al período y se guardan en la variable local *Period VARIABLE.* En último lugar, se registra la operación incluyendo el *SID* y el período guardados. Todo lo descrito anteriormente se muestra en el código de la siguiente página.

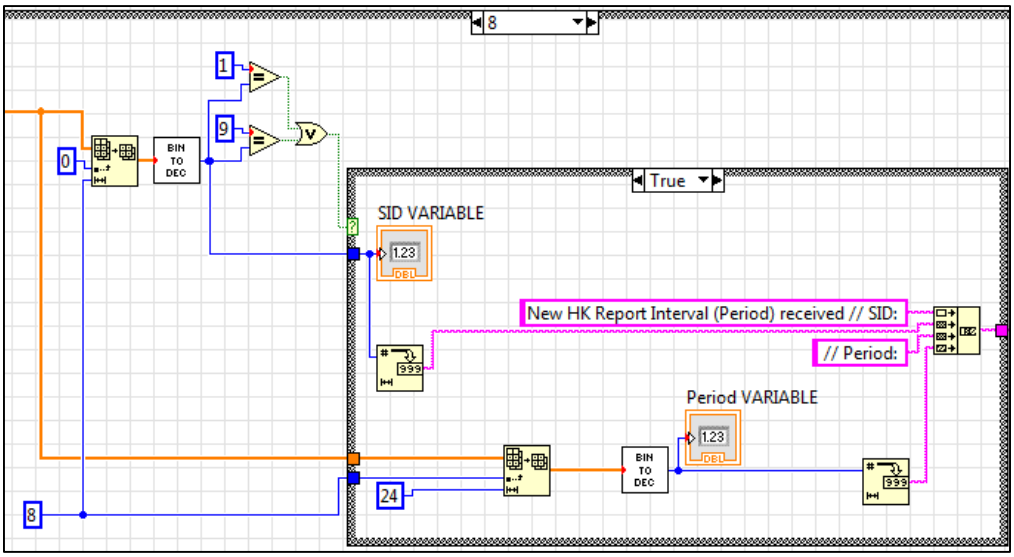


Fig. 9.16. Definición de un nuevo intervalo de reporte de housekeeping

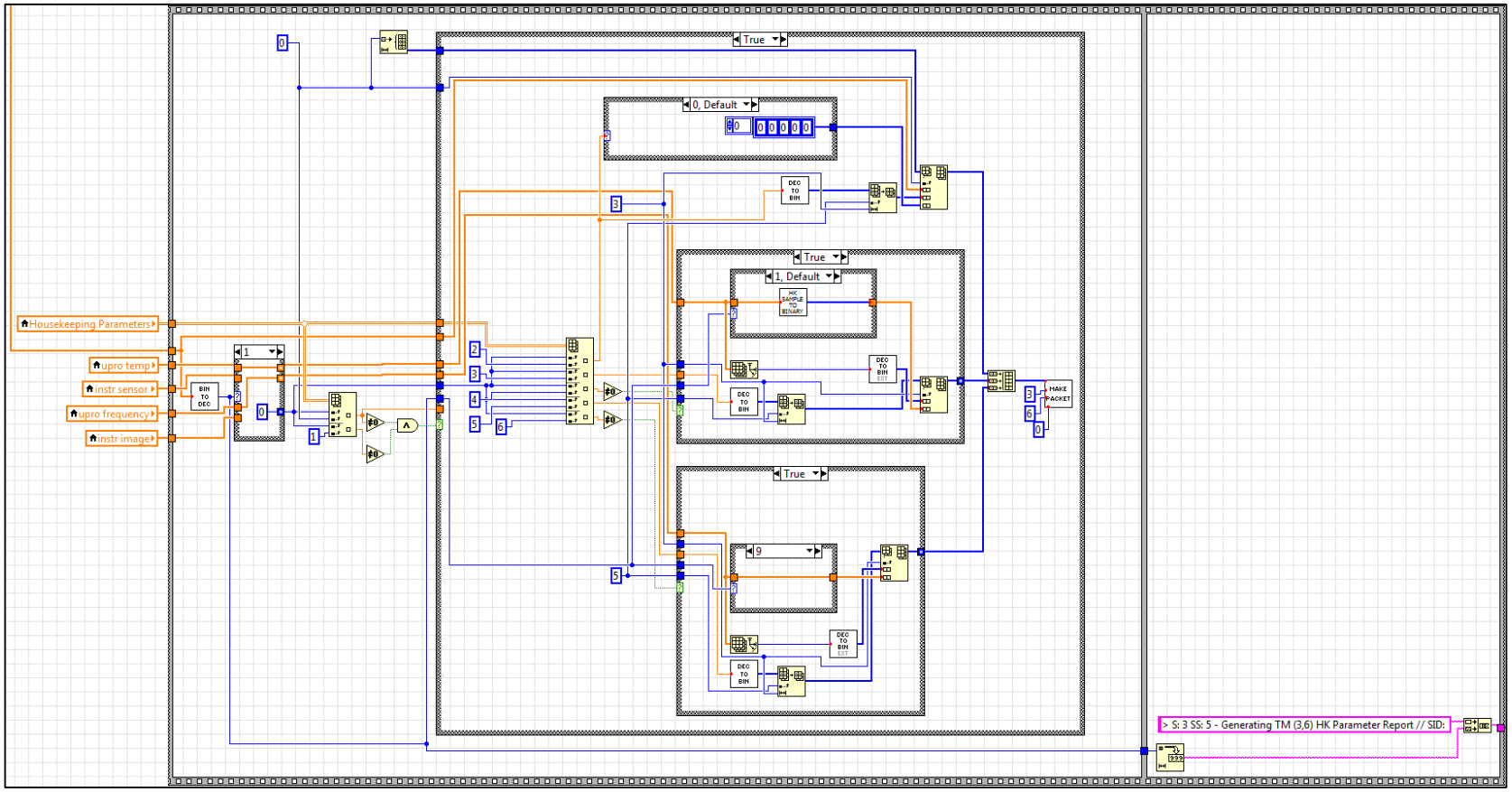


Fig. 9.17. Generación de reportes de housekeeping

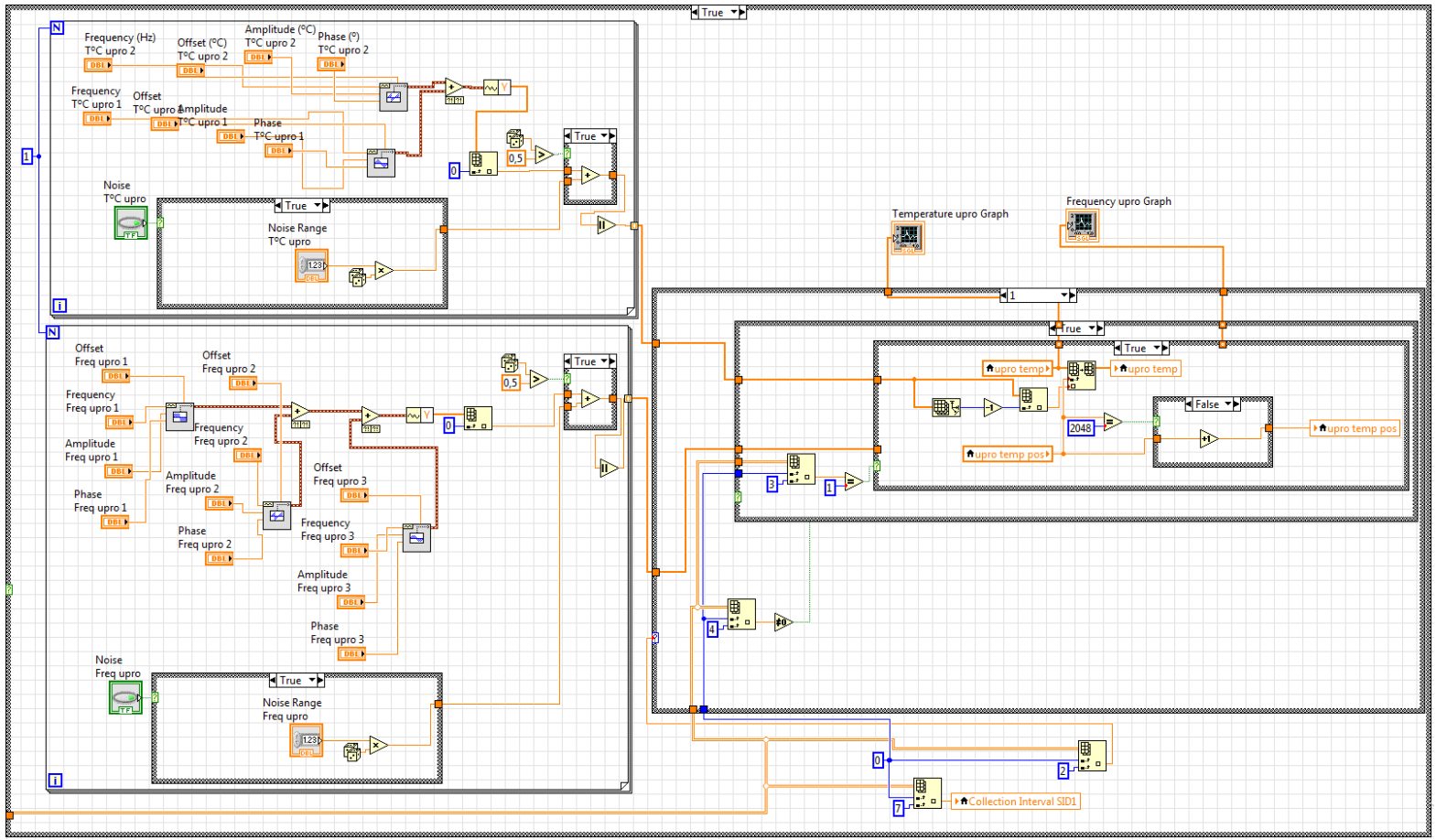


Fig. 9.18. Generación de datos de housekeeping SID 1

10. Servicio 5: Eventos

Para este servicio se establecen cuatro niveles de progreso de eventos, uno de procesado sin errores y tres de error con diferentes niveles de gravedad:

- Proceso normal;
- Error de gravedad leve (*warning*);
- Error de gravedad media (requiere acción desde la estación de tierra);
- Error de alta gravedad (requiere realizar una acción desde el propio ordenador de a bordo).

10.1 Funcionamiento del módulo de Eventos

La estructura que se mantiene en el ordenador a bordo acerca de los procesos es la siguiente:

RID (16)	Event RG State (1)
'X'	'0'/'1'

Tabla 10.1. Estructura de procesos de eventos

Cada proceso del que se pueden reportar eventos tiene un *Report ID (RID)* asociado, que junto con el *APID*, identifica dicho proceso. Los *RID* están predefinidos como se muestra más adelante. Cada uno de ellos tiene un campo estado que puede estar activado ('1') o desactivado ('0'). En caso de estar activado se generan reportes de eventos, mientras que se está en el estado opuesto no se genera ningún reporte. Además en el ordenador a bordo del satélite se lleva un registro de los eventos ocurridos en el mismo y que puede ser borrado.

A continuación se muestra una tabla con los nueve *RID* diferentes predefinidos por el sistema y su nivel gravedad:

RID	***Evento***	***Nivel de gravedad***
7	Cambio de orientación de panel solar	Leve
8	Bajo aumento de temperatura del sistema	
9	Borrado de sector de memoria RAM	
62	Fallo de un sector de la memoria interna	Media
63	Temperatura alta en el instrumento de imágenes	

64	Repliegue de panel solar necesario	
85	Fallo del microprocesador	Alta
86	Fallo en el sistema de alimentación	
87	Nivel de temperatura excesivo en el sensor	

Tabla 10.2. Tabla de definición de eventos predefinidos

De manera interna, la tabla de eventos queda como sigue:

RID	7	8	9	62	63	64	85	86	87
Event RG State	‘0’/’1’	‘0’/’1’	‘0’/’1’	‘0’/’1’	‘0’/’1’	‘0’/’1’	‘0’/’1’	‘0’/’1’	‘0’/’1’

Tabla 10.3. Estructura de la tabla de eventos

Para su implementación se emplea un *array* bidimensional formado por nueve columnas y cuyo nombre es *Event Table.* Para realizar el registro de los eventos se ha establecido un elemento *string,* con barra vertical mostrada, llamado *Event Log.* Para la generación de reportes de las telemetrías que son simuladas (*TM (5,1), TM (5,2), TM (5,3)* y *TM (5,4)*) se incluye un elemento *Tab Control* que las agrupa.

Con todo lo descrito anteriormente, la pestaña *Events* del *Panel Frontal* tiene el siguiente aspecto:

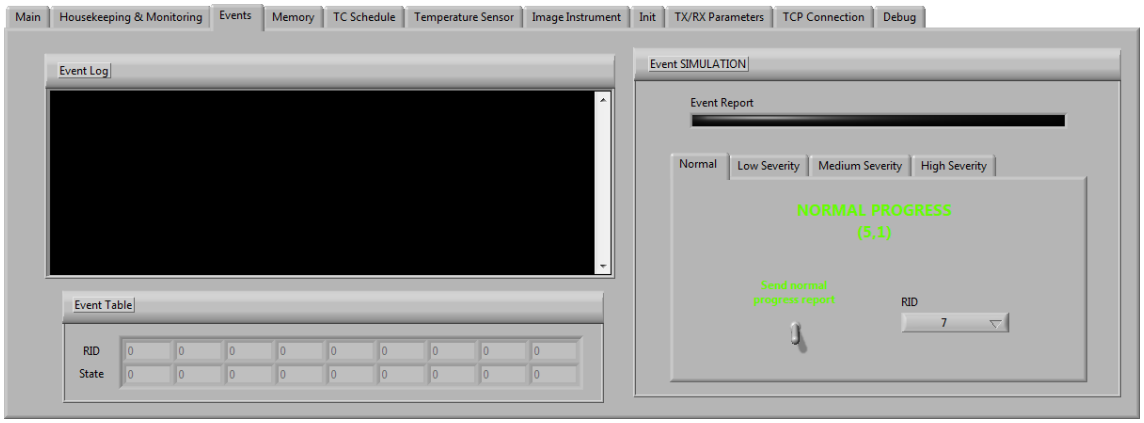


Fig. 10.1. Pestaña Events

De manera general, referente a la implementación de este servicio, su presencia está en el bucle de generación de reportes de eventos y *reboot* para las primeras cuatro secciones de este servicio, y en el bucle de recepción y procesado junto con la función *event.vi* para el resto de secciones del servicio.

10.2 Normal Progress Report (5,1) [TM]

Basado en la telemetría (5,1) del estándar [1]. Página 80.

Con esta telemetría se informa acerca del proceso normal de funcionamiento de un servicio. Tiene el siguiente formato:

RID (16)
'X'

Tabla 10.4. Formato de trama de TM (5,1)

El campo *RID* identifica, junto con el *APID*, el servicio que está funcionando correctamente.

En el *Panel Frontal* en la pestaña *Normal* se han incluido un botón con el que, al pulsarlo, se simulará el evento de progreso normal seleccionado en el elemento *Menu Ring* con nombre de variable *RID*. Todo ello se ilustra en la imagen anterior.

En cuanto su implementación, se ha generado el código de la figura ***Fig. 10.2. Generación de reporte de progreso normal de evento.***

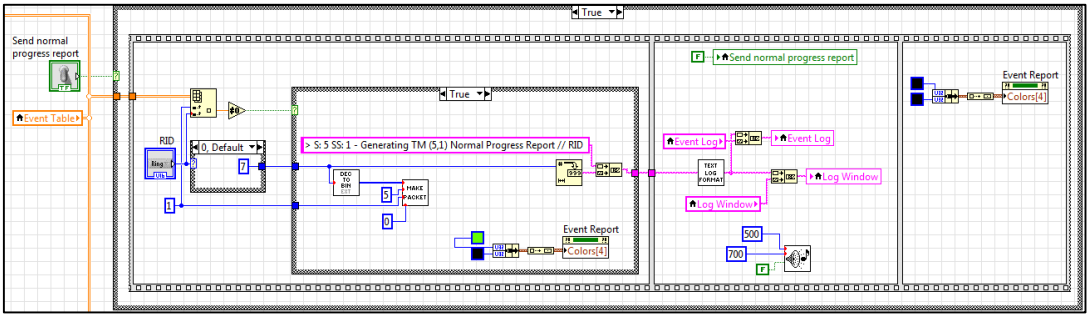


Fig. 10.2. Generación de reporte de progreso normal de evento

Como se puede observar, se incluye un *Case Structure* en el que en caso de que el botón sea pulsado se genera el reporte.

En primer lugar, se lee el elemento seleccionado en el menú desplegable y se identifica el *RID* escogido. Luego se mira en la fila 1 (en la que se indica el estado de reporte de cada *RID*) para comprobar que está a nivel alto ('1') con lo que se puede realizar dicho reporte. En caso de poderse, se le pasa por parámetro a la función *make_packet.vi* el *RID* pasado a binario con extensión de 16 bits (se emplea la función *dec_to_bin_ext.vi*), como *Service Type* el valor 5 y como *Subservice Type* el valor 1 *Sequence Count* toma valor cero en todo el

Servicio 5. Además se cambia el color del LED *Event Report* en este caso a verde.

En segundo lugar, se registra la operación realizada tanto en *Event Log* como en *Log Window*. En el caso de que se haya podido crear el reporte se muestra el texto de la imagen anterior incluyendo el *RID* correspondiente, en caso contrario se informa de que primero hay que poner el campo *Event RG State* a '1', tal y como se ilustra a continuación:

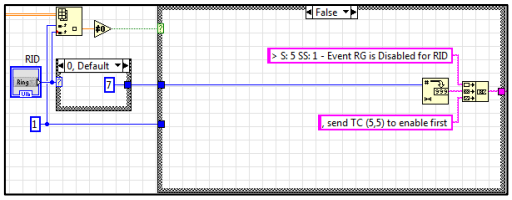


Fig. 10.3. Registro de operación fallida en Log Window

Además se pone a *false* en controlador (botón) con el que se generó el evento y se emite un sonido con la función *beep.vi* (a 500 Hz durante 700 ms). Una vez se concluye con todo, se pasa a poner el LED de nuevo en negro.

10.3 Error/anomaly report – low severity - warning (5,2) [TM]

Basado en la telemetría (5,2) del estándar [1]. Página 80.

Con esta telemetría se informa acerca de un error o anomalía de funcionamiento de un servicio. Simplemente informa acerca del *warning*. Puede ser un proceso rutinario o alguna función implementada que requiere realizar una acción de poca importancia en el sistema. Tiene el mismo formato que Normal Progress Report (5,1) [TM].

En la pestaña *Low Severity* del *Panel Frontal* se han incluido tres botones, uno por cada *RID* de este nivel de gravedad con los que, al pulsarlos, se simulan los eventos de baja gravedad. Su aspecto es el de la siguiente figura.

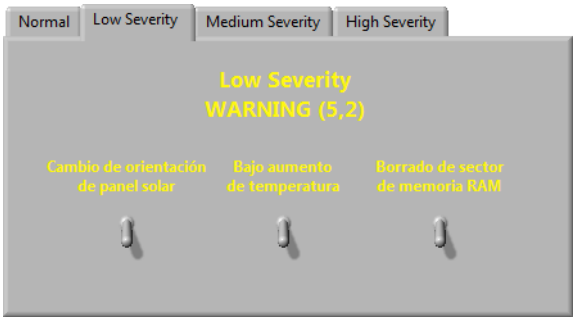


Fig. 10.4. Pestaña Low Severity

En lo referente al código generado para esta telemetría, por cada *RID* se ha empleado una *Case Structure* y se hace el mismo proceso cambiando únicamente el valor del *RID* y la posición del *array* de *Event Table* (la columna) en la que se mira si dicho *RID* tiene o no activada la generación de reportes de eventos.

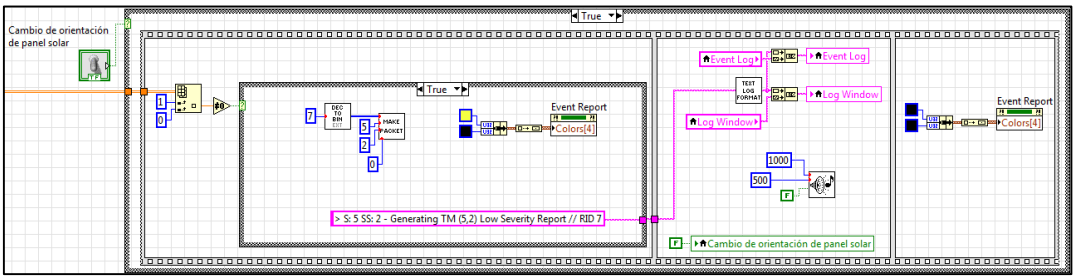


Fig. 10.5. Generación de reporte de evento de aviso

Como se ve en la figura anterior, simplemente se mira el estado de reporte asociado al *RID* seleccionado (fila 1, columna 0 de la tabla de eventos), que debe ser distinto de cero, en cuyo caso se crea un paquete con el *RID* como datos de aplicación, y con el tipo de servicio y el subservicio con valores 5 y 2 respectivamente. Luego se mandan a través de la función *make_packet.vi*. Se cambia el color del LED a amarillo, lo que indicará *warning*. Tras realizar esta serie de operaciones se registra en *Event Log* y en *Log Window.* Además se emite un sonido igual que en el caso anterior pero con frecuencia 1 kHz (más agudo) y durante medio segundo. Luego se vuelve a cambiar el LED de color a negro.

Para *RID 8* y *RID 9* se realizan las mismas operaciones pero se comprueban las columnas 1 y 2, y se muestra el *RID* correspondiente en el *log*.

10.4 Error/anomaly report – medium severity – ground action (5,3) [TM]

Basado en la telemetría (5,3) del estándar [1]. Página 80.

Con esta telemetría se informa acerca de un error de funcionamiento de un servicio. Informa acerca de un evento de gravedad media por lo que requiere una acción desde la estación de tierra. Tiene el mismo formato que Normal Progress Report (5,1) [TM].

A continuación se puede ver el aspecto de la pestaña *Medium Severity* que es empleada para esta telemetría. De nuevo se sitúan tres botones para cada uno de los *RID* de este nivel de gravedad.

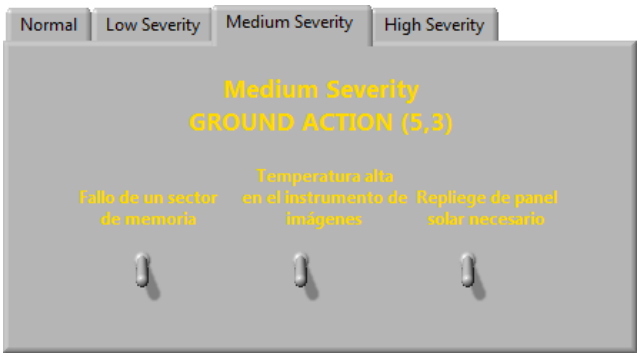


Fig. 10.6. Pestaña Medium Severity

Para implementar los reportes se realiza una operación análoga a la empleada en los casos de gravedad baja. Una vez se ha pulsado el botón, se despulsa y se comprueba que esté activada la generación de eventos para el *RID* seleccionado. En caso de estarlo, se crea el paquete con el *RID* como datos de aplicación, *Service Type* con valor 5 y *Subservice Type* con valor 3. Luego se introduce en la función *make_packet.vi* para ser enviados. En último lugar, se registra la operación de manera análoga a los casos anteriores. En este caso se cambia al color del LED a naranja y el sonido es más agudo (frecuencia a 1.5 kHz). Todo el proceso descrito se muestra en la siguiente figura:

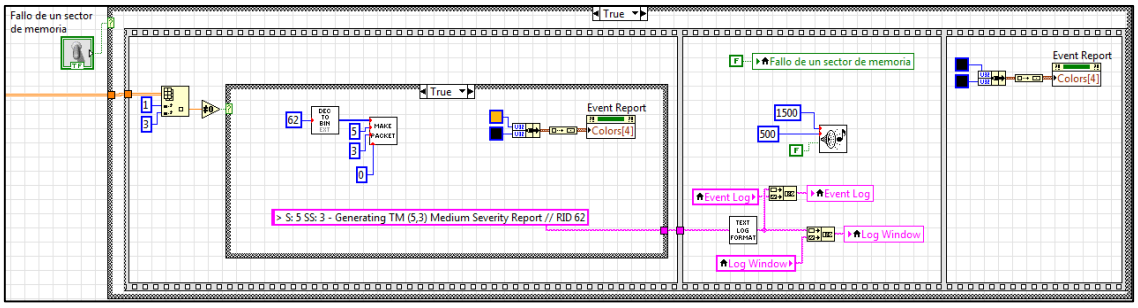


Fig. 10.7. Generación de reporte de evento de gravedad media

Para *RID 63* y *RID 64* se realizan las mismas operaciones pero se comprueban las columnas 4 y 5, y se muestra el *RID* correspondiente en el *log*.

10.5 Error/anomaly report – high severity – on-board action (5,4) [TM]

Basado en la telemetría (5,4) del estándar [1]. Página 80.

Con esta telemetría se informa acerca de un error de funcionamiento de un servicio. Se trata de un error de alta gravedad, por lo que se realiza una acción directamente desde el propio satélite. Por tanto, se realizan operaciones adicionales en el funcionamiento del satélite para subsanar dichos errores. Tiene el mismo formato que Normal Progress Report (5,1) [TM].

La visualización de esta telemetría en *Front Panel* es la de la figura ***Fig. 10.8. Pestaña High Severity.***

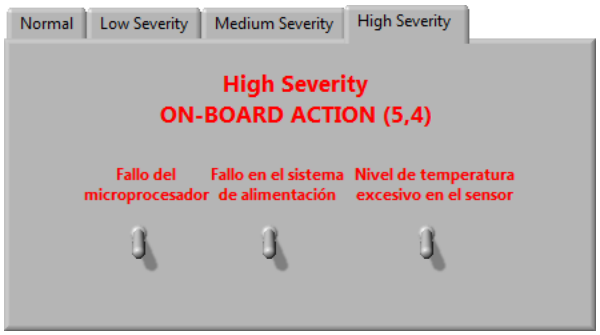


Fig. 10.8. Pestaña High Severity

De manera igual a las dos telemetrías anteriores hay tres botones, uno por cada *RID* de nivel alto de severidad. En el caso de que se pulse el botón asociado al *RID* con valor 86, se ejecuta el siguiente código:

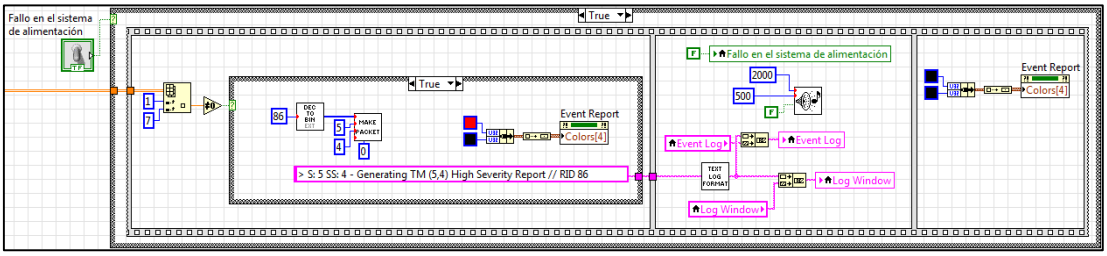


Fig. 10.9. Generación de reporte de evento de alta gravedad para RID 86

Su funcionamiento es el mismo que en las telemetrías anteriores salvo por el parámetro *Subservice Type* pasado para la creación del paquete a valer 4. Además el LED en este subservicio se pone en color rojo y el sonido es todavía más agudo (se le pasa frecuencia con valor 2 kHz). El registro de operaciones se realiza de igual manera.

En esta telemetría además, para el *RID 85*, al tratarse de un error que requiere acción en el propio satélite, se modifica la tabla de *housekeeping* referente al *SID 1* que es el que tiene que ver con el microprocesador. Es decir, se paran los reportes asociados y la monitorización de sus parámetros simulando que no pueden ser obtenidos. De este modo, la ejecución del programa es la mostrada en la figura de la siguiente página.

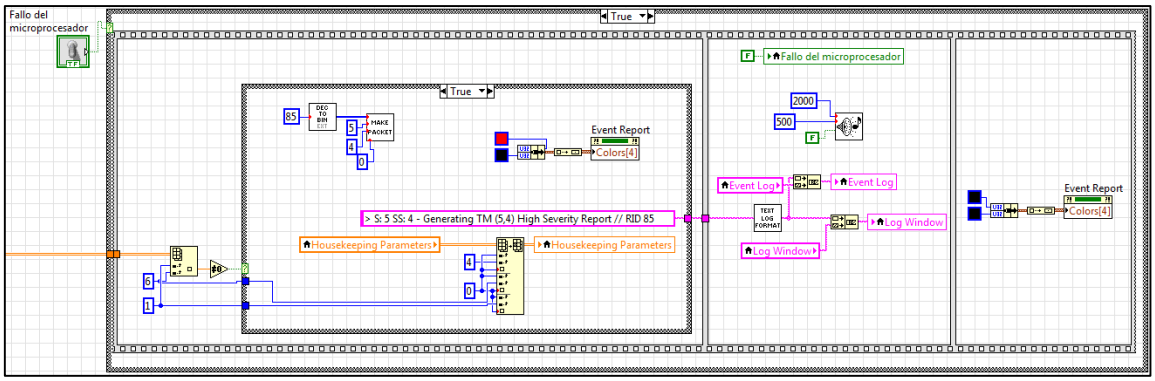


Fig. 10.10. Generación de reporte de evento de alta gravedad para RID 85

De igual modo, para el *RID 87* además del envío de la telemetría correspondiente se desactiva el módulo de ejecución del sensor (se pone a *false* la variable local *Sensor State*), por lo que el sensor de temperatura para su actividad, tal y como se muestra:

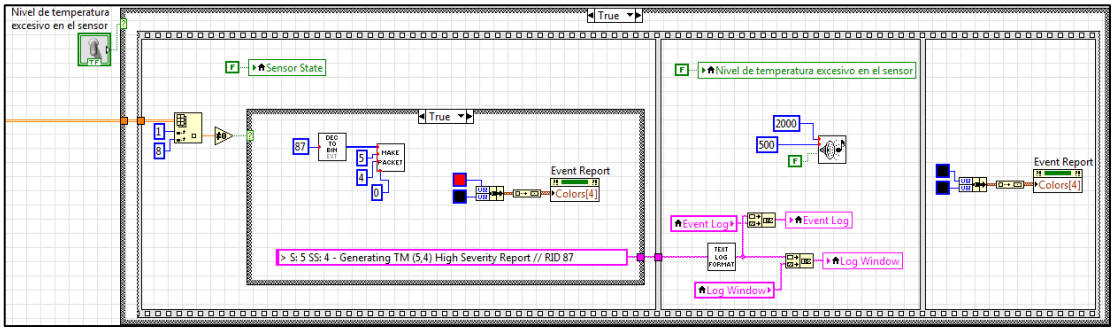


Fig. 10.11. Generación de reporte de evento de alta gravedad para RID 87

10.6 Enable Event Report Generation (5,5) [TC]

Basado en telecomando (5,5) del estándar [1]. Páginas 80 y 81.

Con este telecomando se activa la generación de reportes de eventos. De este modo solamente cuando esté activo se reportarán eventos, mientras que si está desactivado no se informa del evento con una *RID* en concreto. Tiene el mismo formato que Normal Progress Report (5,1) [TM].

En lo referente al programa, se comprueba que el *Service Type* tenga valor 5 y se ejecuta el código asociado al telecomando recibido con la función *event.vi* (más adelante se explica su funcionamiento) y se registra la operación tanto en *Log Window* como en *Event Log*. En segundo lugar, en caso de generarse telemetría, se inserta la función *make_packet.vi*. Se ejecuta código de la siguiente figura.

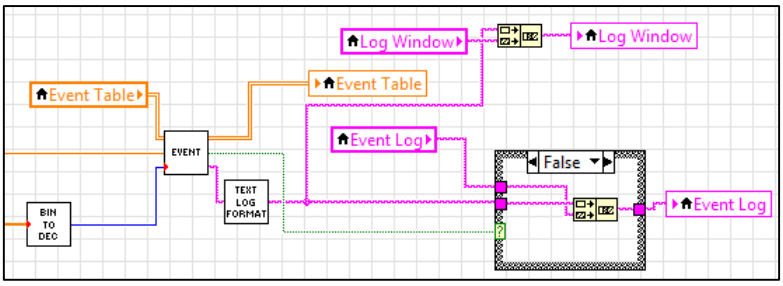


Fig. 10.12. Ejecución dentro del módulo de recepción y procesado

Para la implementación de este telecomando y de los cuatro posteriores del módulo de eventos se hace uso de un instrumento virtual creado con tal fin llamado *event.vi*, que tiene como entradas: la tabla de eventos (*Event Table IN*), un campo *RID* y *Subservice Type* para identificar el telecomando recibido. Como salidas se tiene la tabla de eventos actualizada (*Event Table OUT*), un registro de operaciones (*Log OUT*) y un indicador de borrado tipo booleano del registro de operaciones del módulo de eventos (*Event Log Clear*). A continuación se muestra dicho instrumento virtual:

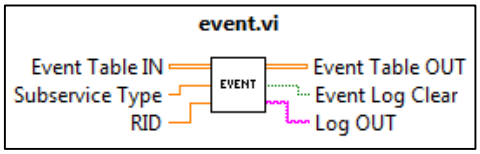


Fig. 10.13. Instrumento virtual event.vi

Todo lo descrito hasta el momento referente a la implementación de este módulo es común al resto de secciones de esta sección.

De manera particular, para esta telemetría se ejecuta el siguiente código:

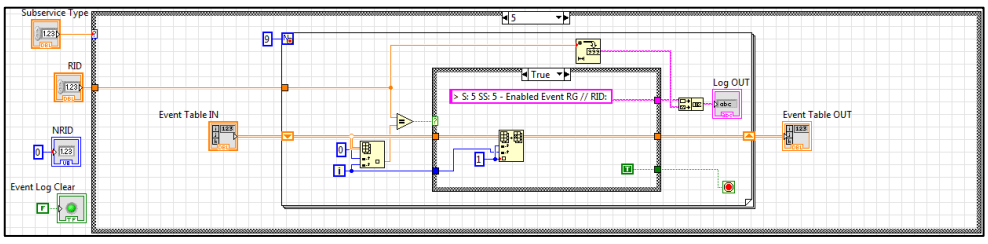


Fig. 10.14. Activación de la generación de reportes de eventos

Una vez identificado el *Service Type* con valor 5, se realiza un bucle *for* con 9 iteraciones (tantas como columnas tiene *Event Table IN*) en el que se busca en la fila 0 posición *i* el valor del *RID* y si coincide con el recibido en el telecomando se pone su correspondiente *Event RG State* a '1' y se registra la operación en el *log*. Además se para el bucle pasándole el valor *true* al *Conditional Terminal*. En caso de ser distintos, no se hace nada y se vuelve a

ejecutar el bucle. Se emplea un registro de desplazamiento (*Shift Register*) para la tabla de eventos para emplear siempre la misma.

10.7 Disable Event Report Generation (5,6) [TC]

Basado en telecomando (5,6) del estándar [1]. Páginas 80 y 81.

Con este telecomando se desactiva la generación de reportes de eventos. De este modo solamente cuando esté activo se reportarán eventos, mientras que si está desactivado no se informa del evento con una *RID* en concreto. Tiene el mismo formato que Normal Progress Report (5,1) [TM].

Este telecomando es análogo al anterior pero poniendo *Event RG State* a '0'. El código es el mostrado a continuación.

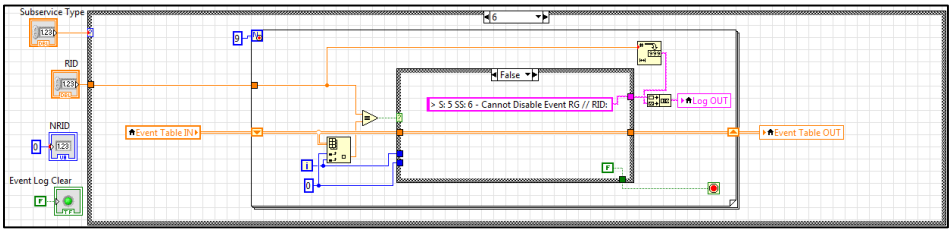


Fig. 10.15. Desactivación de la generación de reportes de eventos

10.8 Clear Event Log (5,16) [TC]

Con este telecomando se elimina por completo el historial de eventos que tiene el ordenador a bordo. No contiene datos de aplicación, por lo que simplemente se identifica a través de los campos de la cabecera *Service Type* y *Service Subtype*.

Dentro de la función *event.vi* se identifica *Service Type* con valor 5 y se ejecuta el código. Se indica que se borra *Event Log* poniendo *Event Log Clear* a *true* y registrando la operación. Además se pasa la tabla de entrada por la salida sin cambios para que no la borre en *main.vi*.

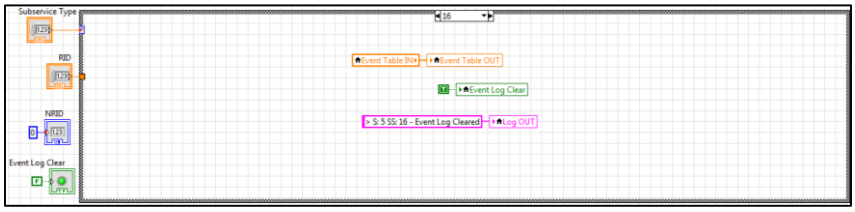


Fig. 10.16. Generación de la operación de borrado del registro de eventos

En este caso, además, se ejecuta el caso *true* por lo que en *Event Log* únicamente queda registrada la operación de borrado.

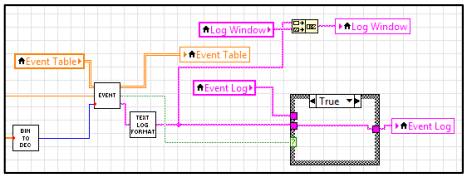


Fig. 10.17. Ejecución de la operación de borrado en main.vi

10.9 Report Enabled Event Packets (5,17) [TC]

Con este telecomando se solicita una lista con todos los procesos (identificados con su respectivo *RID*) cuyo estado de generación de reportes se encuentra activo. No tiene datos de aplicación.

10.10 Enabled Event Packets Report (5,18) [TM]

Se responde a dicho telecomando con la telemetría Enabled Event Packets Report (5,18). Por tanto, en él se incluyen todos los *RID* cuyo estado es activo. El paquete tiene el siguiente formato:

NRID (16)	repetido NRID veces
	RID (16)
'X'	'X'

Tabla 10.5. Formato de trama de TM (5,18)

El campo *NRID* indica el número de campos *RID* de los que se reporta su estado. Cada *RID* identifica al evento con estado activo ('1'). Ambos campos están formados por dos octetos cada uno.

Su implementación es la que sigue:

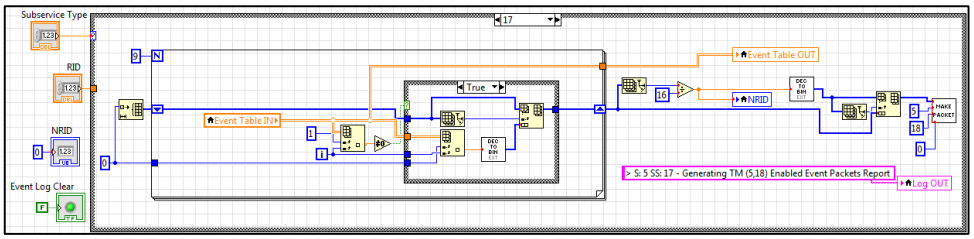


Fig. 10.18. Generación del reporte de eventos activos

Como puede verse, hay un bucle en el que se comprueba el estado del campo *Event RG State* asociado a cada uno de los nueve *RID*. En caso de valer '1' se introduce en un *array* con la función *Insert Into Array* en la posición devuelta por la función *Array Size.* Se introducen 16 bits que se corresponden con la representación binaria devuelta por *dec_to_bin_ext.vi* del *RID*. Una vez concluye el bucle, se obtiene el campo *NRID* con el cociente del tamaño del *array* anterior entre el número de bits de cada campo *RID*. Se pasa a binario y se introduce al principio del *array.* Este *array* definitivo se le pasa por parámetro a la función *make_packet.vi* junto con los valores 5 y 18 correspondientes a *Service Type* y *Service Subtype* respectivamente. Además se registra la operación en *Log OUT* y se pasa la tabla de entrada por la salida sin cambios.

10.11 Report Disables Event Packets (5,19) [TC]

Con este telecomando se solicita una lista con todos los procesos (identificados con un *RID*) cuyo estado de generación de reportes se encuentra desactivado. No tiene datos de aplicación.

10.12 Disables Event Packets Report (5,20) [TM]

Con la telemetría se responde al telecomando Report disables event packets (5,19). Por tanto, en él se incluyen todos los *RID* cuyo estado es desactivado. El formato de trama es idéntico al de la telemetría Enabled Event Packets Report (5,18).

La implementación de esta parte es análoga a la de la sección anterior. La única diferencia reside en la comprobación de *Event RG State,* que deberá valer '0'. Además, se pasa como *Service Subtype* el valor 20 para crear el paquete y el registro toma un valor *string* diferente acorde con esta sección. Todo se muestra a continuación:

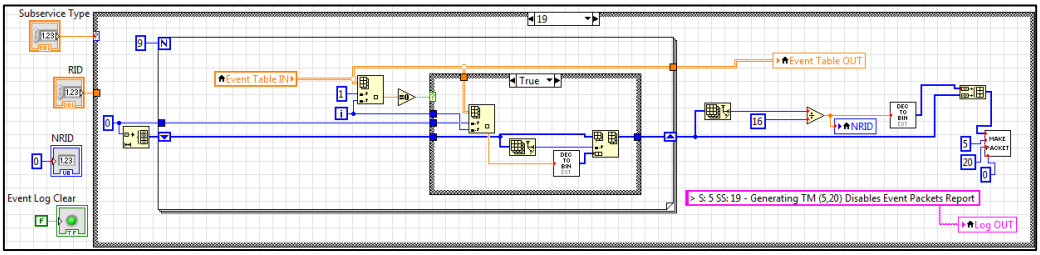


Fig. 10.19. Generación del reporte de eventos inactivos

11. Servicio 6: Gestión de memoria

Este servicio tiene como función el manejo de la memoria del satélite. En esta implementación se manejan la *memoria interna* permanente, la *memoria RAM (Random Access Memory)* y la *memoria ROM (Read-Only Memory)*. El formato de la memoria que se ha establecido es el que sigue:

	Bit 0	**Bit 1**	**Bit N**	**Bit 63**
Bloque 0	'0'/'1'	'0'/'1'	'0'/'1'	'0'/'1'
Bloque 1	'0'/'1'	'0'/'1'	'0'/'1'	'0'/'1'
Bloque 2	'0'/'1'	'0'/'1'	'0'/'1'	'0'/'1'
Bloque N	'0'/'1'	'0'/'1'	'0'/'1'	'0'/'1'
Bloque M	'0'/'1'	'0'/'1'	'0'/'1'	'0'/'1'

Tabla 11.1. Formato de memoria del sistema

Como se puede observar, se establecen bloques de memoria de 64 bits. En esta implementación se permite elegir el número de bloques de memoria que serán simulados. Por defecto se establecen 1024 bloques aunque también se permite seleccionar 128, 256, 512 y 65536 bloques. Esto se debe a que la creación de una matriz de 1024x64 puede ralentizar el procesamiento en equipos con bajas prestaciones y la selección de 65536 es inviable como memoria simulada. Dicha selección se realiza antes de iniciar el programa en la pestaña *Init* del panel principal y si se cambia no tendrá efecto hasta que o bien se vuelva a iniciar de nuevo, o bien se produzca un *Reboot*.

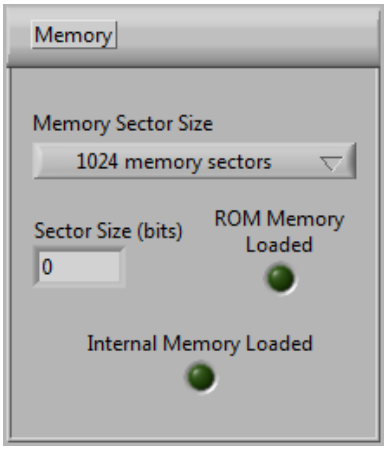


Fig. 11.1. Inicialización de la memoria

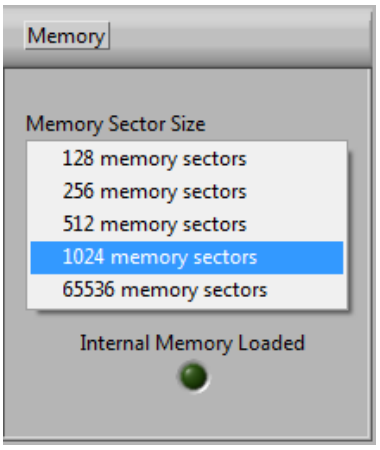


Fig. 11.2. Selección del tamaño de memoria

En el *Panel Frontal* en la pestaña *Memory* se pueden ver las tres memorias de las que consta el sistema tras su inicialización y los cambios que se van produciendo con llegada de telecomandos. Se representa una parte de las mismas. La inicialización de la memoria se describe en *5.3 Inicialización del sistema*.

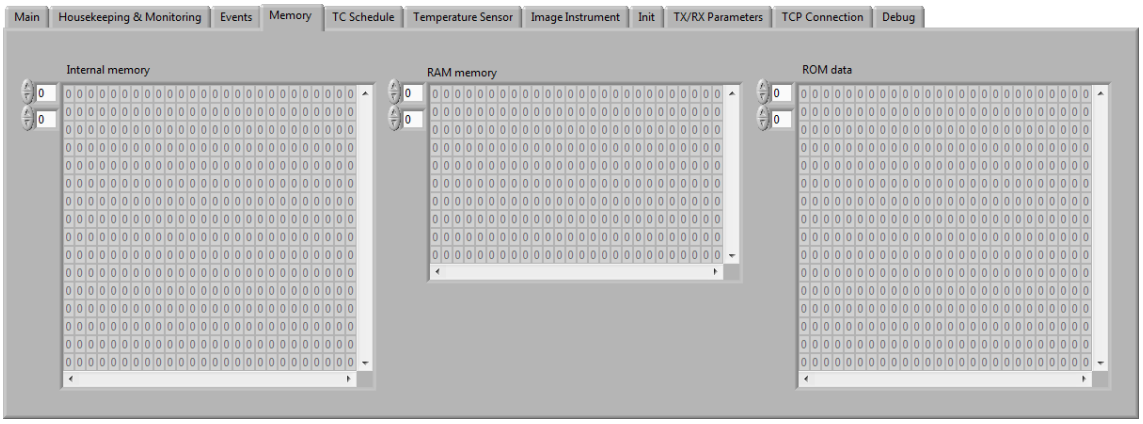


Fig. 11.3. Pestaña Memory

De manera general para todo este servicio, una vez identificado que se trata de un telecomando con *Servicio 6* se comprueba dicho valor en el *Case Structure* como en servicios anteriores y se ejecuta el código mostrado en la siguiente figura:

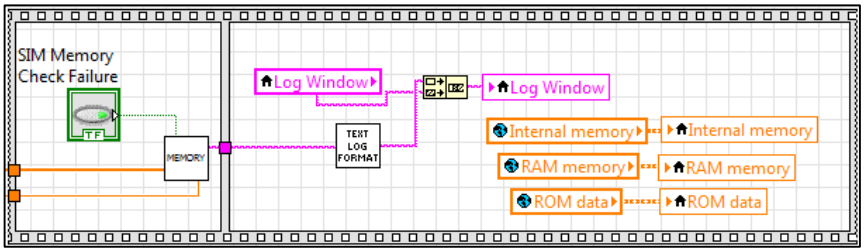


Fig. 11.4. Ejecución del módulo de memoria en main.vi

En primer lugar se ejecuta la función *event.vi*. Se introducen como parámetros un booleano que indica si se simula error de memoria (para cuando se reciba un Check memory (6,4)) el subservicio y los datos de aplicación. Se devuelve como parámetro un *string* con el registro de la operación realizada en el instrumento virtual. En las siguientes secciones se explica la implementación llevada a cabo de dicho *vi*.

En segundo lugar, se registran las operaciones en *Log Window* y se insertan en las variables locales (los indicadores de la pestaña *Memory*) las nuevas memorias para poder ser vistas.

Para este servicio se emplean tres variables globales: *Internal memory, RAM memory* y *ROM data* que servirán para comunicar el instrumento virtual *main.vi* con el *memory.vi.* Se emplean variables globales y no salidas del instrumento virtual porque el tamaño de las matrices es muy grande y el transvase de información en el bucle de recepción y procesado puede ser muy

pesado. A continuación se muestran dichas variables globales que se encuentran dentro del fichero *global.vi*.

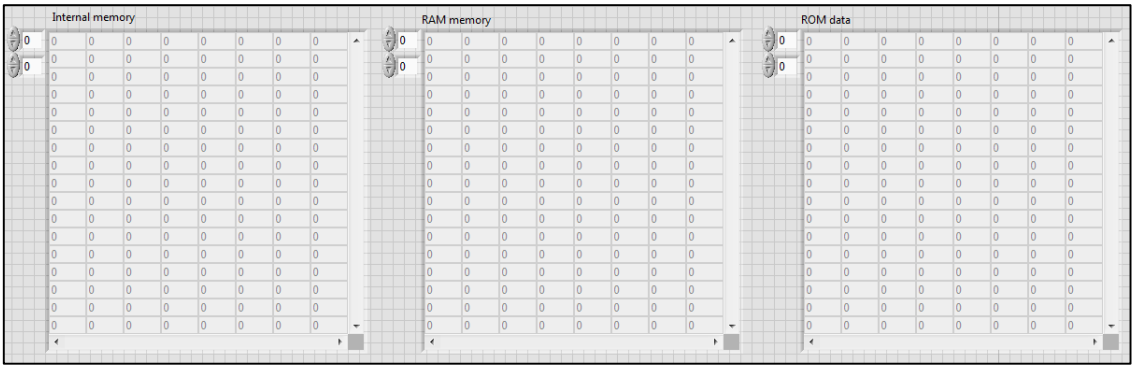


Fig. 11.5. Variables globales de memoria

11.1 Load data into memory (6,1) [TC]

Basado en el telecomando (6,2) del estándar [1]. Página 85.

Este telecomando permite la carga de los datos en memoria dinámica a partir de una dirección de memoria absoluta, de modo que se guarda en la *memoria RAM* hasta que se recibe el telecomando Dump memory area (6,2) para cargar de forma permanente los datos en la memoria. Tiene el siguiente formato:

Start Address (16)	Length (4)	Spare (4)
'X'	'X'	'0000'

Tabla 11.2. Formato de trama de TC (6,1)

El campo *Start Address*, formado por 16 bits, indica la posición inicial de memoria en la que se van a cargar los datos y tiene rango de 0 a 1023 porque es el máximo de la memoria en el sistema. Se ha establecido un tamaño de 16 bits para futuras implementaciones. El siguiente campo, *Length*, muestra el tamaño de los datos que se quieren cargar en memoria. El campo *Spare* hace la función de relleno o *padding* para obtener un número entero de octetos entre los dos campos anteriores.

Referente a su implementación, en primer lugar se comprueba que el subservicio toma valor 1. Se extraen de los datos de aplicación los campos de dirección de inicio y longitud (*Start Address* y *Length*) y se pasan de binario a decimal con la función *bin_to_dec.vi.* Luego se comprueba que el campo *Start Address* y *Length* estén dentro del rango de la memoria creada (se suman ambos y si es mayor que el número de filas de la memoria interna no se vuelca la

memoria). En caso de ser válido se emplea la función *Get Submatrix* en la que se cogen todas las columnas (desde la posición 0 hasta la 63) desde la posición *Start Address* con tamaño *Length*, y se insertan en *RAM Memory.* Luego se almacenan en una variable local tanto la dirección de inicio como el tamaño almacenado a la espera de la llegada del telecomando de volcado, y se registra dicha operación. Todo lo descrito se muestra en la siguiente figura:

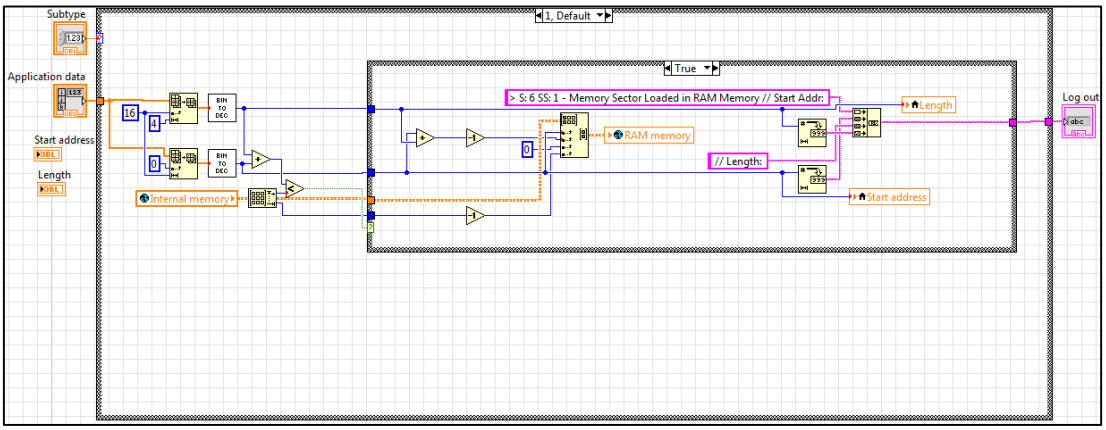


Fig. 11.6. Carga de datos en la memoria RAM

En caso de que el campo de dirección no sea válido se ejecuta lo siguiente:

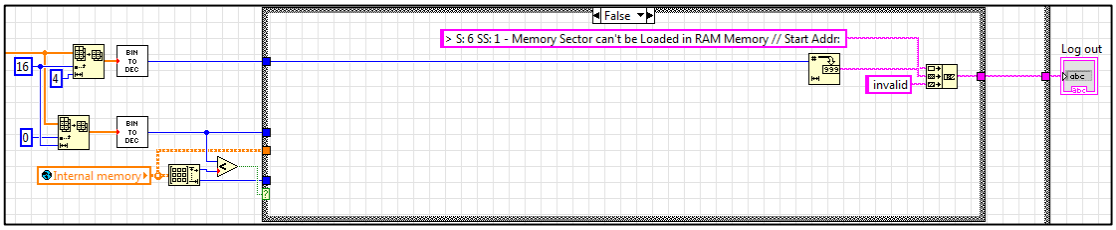


Fig. 11.7. Fallo al cargar los datos en la memoria RAM

11.2 Dump memory area (6,2) [TC]

Basado en el telecomando (6,5) del estándar. Página 87.

Con este telecomando se solicita el envío de los datos cargados en la *memoria RAM* con el telecomando anterior Load data into memory (6,1). Cuando se recibe este telecomando se mandan los bloques de memoria solicitados con la telemetría Memory dump (6,3). No tiene datos de aplicación.

11.3 Memory dump (6,3) [TM]

Esta telemetría es enviada como respuesta al telecomando anterior Dump memory area (6,2). Sirve para mandar los datos solicitados anteriormente. Tiene el formato mostrado en la siguiente tabla.

Start Address (16)	Length (4)	Spare (4)	Data (múltiplo de octeto)
'X'	'X'	'0000'	'X'

Tabla 11.3. Formato de trama de TC (6,3)

El campo *Start Address,* formado por 16 bits, indica la posición inicial del bloque de memoria del campo *Data*. El campo *Length* muestra el tamaño del bloque que quiere ser enviado. El campo *Spare* hace la función de relleno o *padding* para obtener un número entero de octetos entre los dos campos anteriores. El campo *Data* contiene los datos de memoria solicitados. Se mandan tantos como *Length x 64 bits*.

En lo referente al código, tras la recepción del telecomando se produce la llamada a la función *memory.vi.* En dicho instrumento virtual se comprueba que el valor del subservicio sea 2. En caso de serlo, se cogen las variables locales almacenadas en con el telecomando Load data into memory (6,1) y se comprueba que el valor haya sido modificado alguna vez, es decir, sea distinto de 65535 que es el valor al que se inicializa cuando se ejecuta el programa por primera vez. En caso de que no se haya mandado un Load data into memory (6,1) anteriormente se ejecuta el código del caso *false* en el que simplemente se informa del error como se muestra en la siguiente figura:

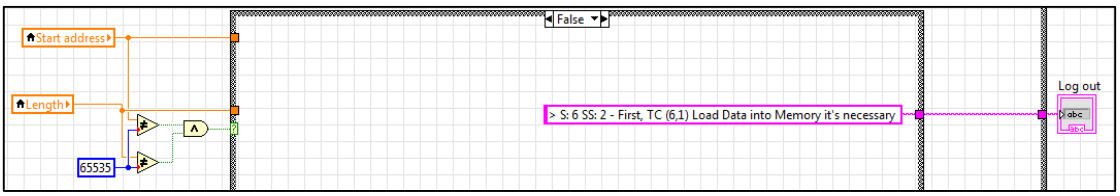


Fig. 11.8. Fallo en la generación de TM (6,3)

En caso contrario, se genera la telemetría respuesta con el formato anteriormente descrito. Para ello se emplea la función *Initialize Array* para crear la secuencia que se manda y los cuatro bits de relleno. Se introducen los 16 bits de la dirección de inicio (previa conversión con la función *dec_to_bin_ext.vi*), los cuatro bits del tamaño (se usa *dec_to_bin.vi* y con *Array Subset* se cogen los cuatro menos significativos) y el relleno. A continuación se muestra la implementación de esta sección.

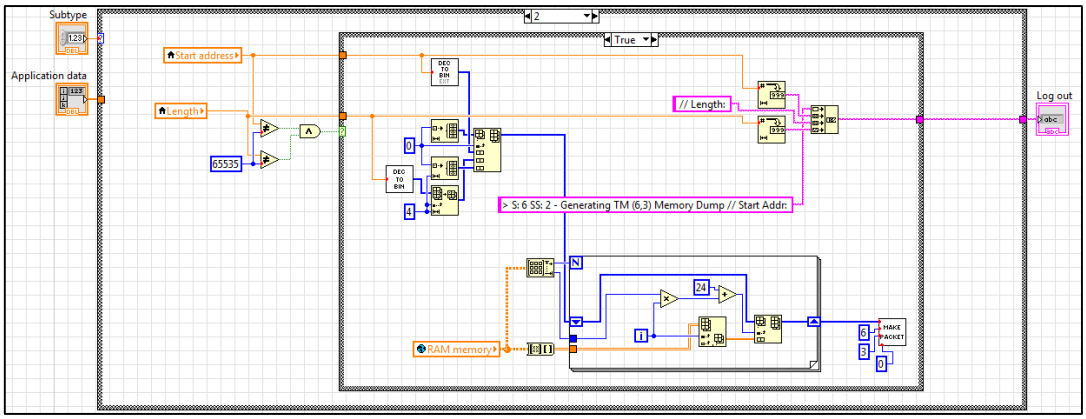


Fig. 11.9. Generación del reporte de memoria

Los datos anteriores se introducen en un bucle en el que se insertan los bits de memoria y se va concatenando con un *Shift Register* hasta completar los datos de la telemetría. Para realizarlo, se pasa de matriz a *array* bidimensional con la función de la siguiente figura:

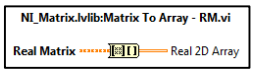


Fig. 11.10. Función de transformación de matriz a array

Dentro del bucle se va cogiendo fila a fila y se van introduciendo en el *array* unidimensional que luego se le pasa por parámetro a la función *make_packet.vi* con *Service Type* valiendo 6 y *Subservice Type* con valor 3. El valor de *Sequence Count* siempre es cero en este servicio. En último lugar se registra la operación con la concatenación del *string* visto en la figura.

11.4 Check memory (6,4) [TC]

Basado en el telecomando (6,9) del estándar [1]. Páginas 88 y 89.

Este telecomando solicita la comprobación de correcto formato de los datos almacenados en la memoria estática del sistema de manera simulada. Cuando es recibido se genera como respuesta la telemetría Memory Check Report (6,5). No tiene datos de aplicación.

11.5 Memory Check Report (6,5) [TM]

Basado en telemetría (6,10) del estándar [1]. Página 89.

Con esta telemetría se responde al telecomando anterior Check memory (6,4), tras realizar la comprobación solicitada de los datos en memoria estática. Tiene el formato descrito en la siguiente tabla.

Memory OK (8)
‘00000000’/’11111111’

Tabla 11.4. Formato de trama de TM (6,5)

Esta telemetría está formada únicamente por un campo formado por un octeto que puede tener dos estados:

- *Memory OK* (bits a ‘0’), lo que indica que hay un fallo en la memoria.
- *Memory OK* (bits a ‘1’), lo que indica que la memoria no contiene errores.

Para su implementación, tras la identificación del *Subservice Type,* se mira la entrada del booleano *SIM Memory Check Failure*, esto es, si está activado en *main.vi* el correspondiente botón, y se introduce en un C*ase Structure*. En caso de estar activado se ejecuta el siguiente código:

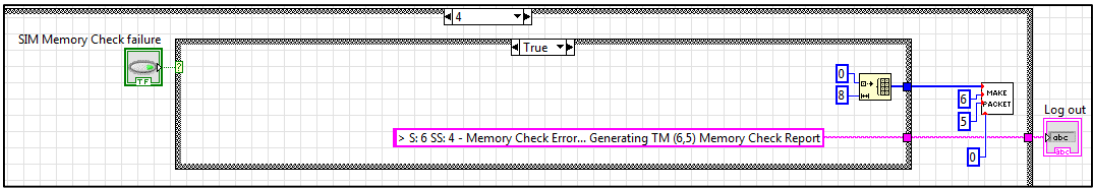


Fig. 11.11. Generación de reporte de memoria incorrecta

Como se observa, se emplea la función *Initialize Array* con valor 0 y tamaño un octeto. Luego se introduce en la función *make_packet.vi* junto con *Service Type* con valor 6, *Subservice Type* con valor 5 y *Sequence Count* con valor cero. Luego se registra la operación en el *string* de salida. Para el caso en que no se simule error, se realizan las mismas operaciones pero en *Initialize Array* se inicializa a 1, tal y como se ilustra a continuación:

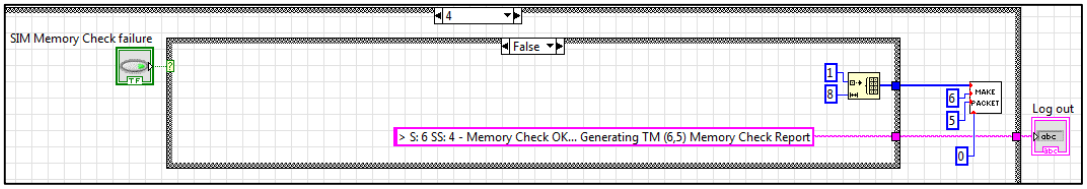


Fig. 11.12. Generación de reporte de memoria correcta

11.6 Preload ROM data (6,16) [TC]

Con este telecomando se solicita cargar los datos precargados en la *memoria ROM* (*Read-Only Memory*) en la memoria del sistema de datos. No tiene datos de aplicación.

Para la implementación de este telecomando, tras la identificación del servicio como en los telecomandos anteriores, se identifica que el subservicio es el 16 y se ejecuta el siguiente código:

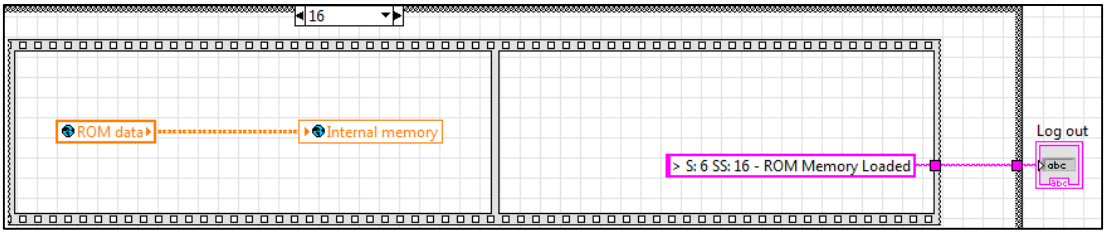


Fig. 11.13. Carga de la memoria ROM en la memoria interna

Se realiza una secuencia en la que en primer lugar se asigna la matriz *ROM data* a *Internal Memory* y después se registra dicha operación en *Log out*.

12. Servicio 9: Gestión del Tiempo

Con este servicio se permite controlar el sistema de tiempos del ordenador de a bordo. En esta implementación se ha tomado como sistema temporal el sistema gregoriano con el registro de día, mes, año, hora, minuto y segundo. Como aclaración, un año tiene 365 días a menos que sea bisiesto que tiene un día más en el mes de Febrero, esto es, pasa a tener 29. A continuación se muestra la posición del sistema de tiempo en el *Panel Frontal*, en la esquina inferior derecha. La fecha tiene formato *DD:MM:YY* y la hora *hh:mm:ss* y además se incluye un reloj analógico.

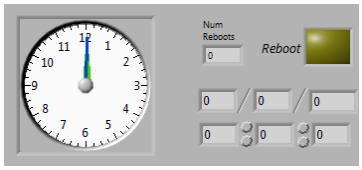


Fig. 12.1. Sistema temporal del satélite y reloj analógico

12.1 Reloj del sistema

El ordenador de a bordo cuenta con un módulo de tiempo en el que se implementa el sistema temporal del satélite.

Para su implementación, en primer lugar antes de entrar en el bucle de generación temporal, se toma como referencia de tiempo el que devuelve la función *Get Date/Time In Seconds* y se agrupa en un *cluster* de datos numéricos con la función *Seconds to Date/Time.* Luego se almacena cada valor en una variable local (*Day, Month, Year, Hour, Minute* y *Second*).

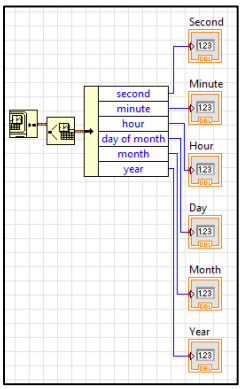


Fig. 12.2. Inicialización del sistema temporal

En segundo lugar, se emplea un bucle *Timed Loop* que se ejecuta una vez por segundo en la entrada *Period* pasándole como parámetro el valor 1000 *(ms)*.

Además, como condición de parada se emplea el botón *STOP Time*. De manera esquemática la estructura es la mostrada en la siguiente figura:

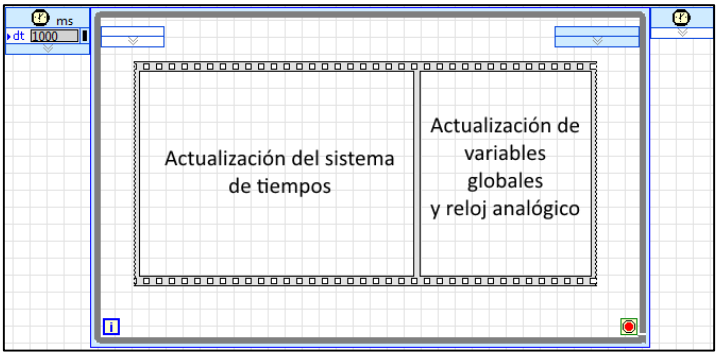


Fig. 12.3. Estructura del módulo temporal del satélite

Como se observa en la figura anterior en primer lugar se actualiza el sistema temporal del satélite en el primer bloque de secuencia.

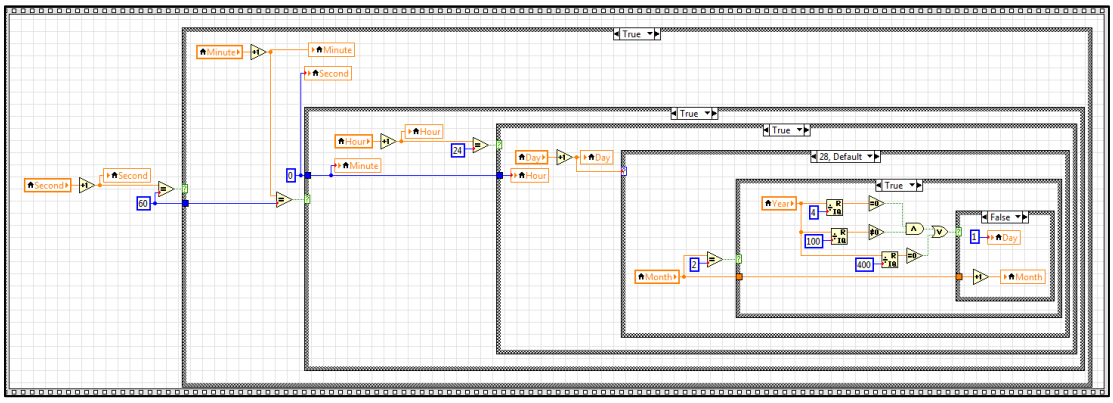


Fig. 12.4. Primer bloque de secuencia del módulo temporal

Su funcionamiento es sencillo, se aumenta el valor de la variable local *Second* una vez por cada iteración del bucle. Para las comprobaciones se emplean elementos *Case Structure*. Cuando se llega al valor 60, se suma una unidad al campo *Minute* y se pone a cero la variable de los segundos. Lo mismo ocurre cuando el valor de *Minute* llega a 60, se aumenta en una unidad *Hour* y se pone a cero los minutos. Cuando *Hour* alcanza el valor 24 se aumenta en una unidad *Day*. Se comprueba si el día es el 28 o menor, 29, 30 ó 31. En cada caso se realiza una acción dependiendo del mes en el que esté el sistema de tiempo en ese instante. Para cada caso:

- *Si es día 28 y es Febrero (Month = 2)*. En este caso se pasa a comprobar el año (*Year*) para saber si es bisiesto. Para ello se comprueba que el resto al dividir el año entre 4 sea 0 y el cociente entre 100 sea distinto de 0, o bien que al dividir el año entre 400 el resto sea 0. Si no es bisiesto se suma uno al mes y se pone el día a

1. En caso contrario, no se hace nada. A continuación se muestra la implementación en *LabVIEW*:

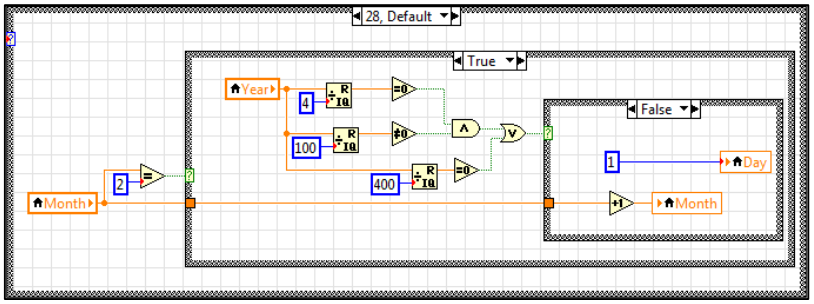


Fig. 12.5. Ejecución si es día 28 y es Febrero

- *Si es día 29 y es Febrero*. Sólo entrará en el *Case* si se ha cumplido anteriormente el punto anterior de modo que es año bisiesto; se suma uno al mes y se pone el día a 1. En caso contrario, no se hace nada.

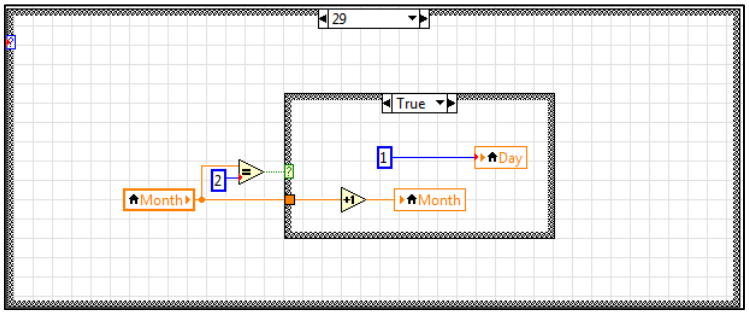


Fig. 12.6. Ejecución si es día 29 y es Febrero

- *Si es día 30 y es Abril, Junio, Septiembre o Noviembre* (*Month toma valor 4, 6, 9 u 11*) se suma una unidad al mes y se pone *Day* a 1. En caso contrario, no se hace nada.

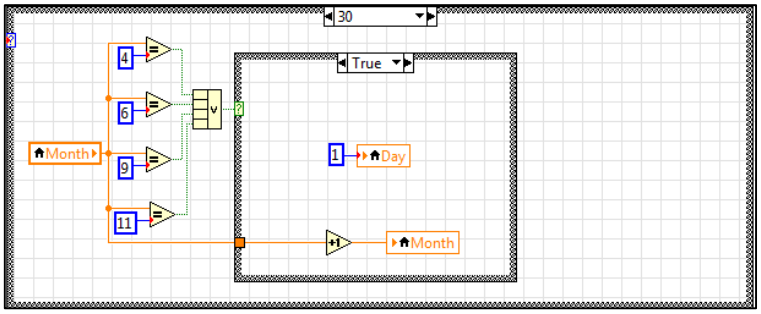


Fig. 12.7. Ejecución si es día 30 y es un mes con los mismos días

- *Si es día 31 y es Enero, Marzo, Mayo, Julio, Agosto u Octubre* (*Month toma valor 1, 3, 5, 7, 8 u 10*) se suma una unidad al mes y

se pone *Day* a 1. En el caso de que sea Diciembre, se ponen el día y el mes a 1, además de aumentar en una unidad el año, *Year*.

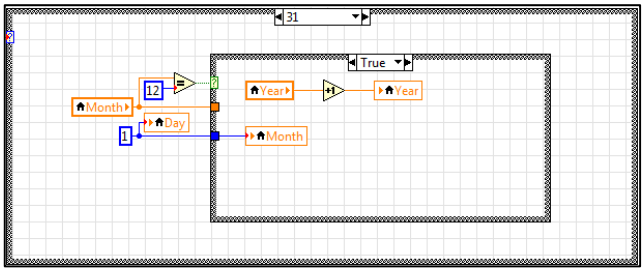


Fig. 12.8. Ejecución si es día 31 y es un mes con los mismos días

En segundo lugar, se actualizan las variables globales de tiempo y el reloj analógico en la segunda *frame* de la secuencia:

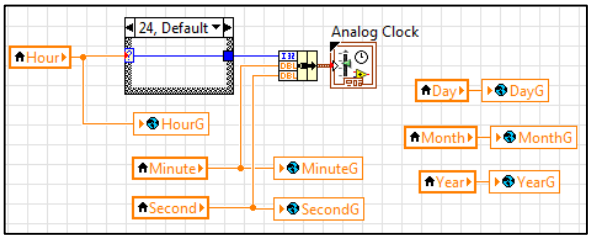


Fig. 12.9. Código de la segunda subsecuencia

Al reloj analógico es necesario pasarle un *cluster* de tres elementos con hora, minuto y segundo. En el caso de la hora es necesario hacer una conversión a un formato de 12 horas con un *Case Structure* como se muestra a continuación.

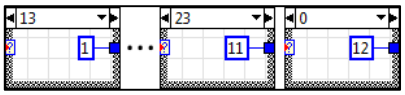


Fig. 12.10. Secuencia de la transformación del formato de hora

12.2 Set OBT (9,1) [TC]

Este telecomando permite reestablecer el sistema de tiempo en el satélite. Cuando se recibe se modifican el día, mes y año, así como los campos de hora, minutos y segundos. Tiene el formato de la siguiente tabla:

Day (5)	Month (4)	Year (12)	Spare (2)	Hour (5)	Minute (6)	Second (6)
'X'	'X'	'X'	'00'	'X'	'X'	'X'

Tabla 12.1. Formato del sistema temporal para reportes de tiempo

El campo *Day* está formado por 5 bits y hace referencia al día del sistema de tiempo. *Month* es un campo formado por 4 bits y establece el. El campo *Year* (12 bits) establece de igual modo el año. El campo *Spare* hace la función de relleno o *padding* para obtener un número entero de octetos y está formado por 2 bits a '0'. El campo *Hour* (5 bits) hace referencia a la hora del reloj del satélite. *Minute* es el campo que hace referencia a los minutos del reloj y está formado por 6 bits. El campo *Second*, con el mismo número de bits que el campo anterior, refiere a los segundos del reloj del satélite.

Se ha elegido ese número de bits teniendo en cuenta el rango de valores que pueden tomar, que son:

Campo	***Nº Bits***	***Rango***	***Rango necesario***
Día (Day)	5	0 - 31	1 - 31
Mes (Month)	4	0 - 15	1 - 12
Año (Year)	12	0 - 4095	2015 - 4095
Hora (Hour)	5	0 - 31	0 - 23
Minuto (Minute)	6	0 - 63	0 - 59
Segundo (Second)	6	0 - 63	0 - 59

Tabla 12.2. Descripción detallada de los campos de tiempo

Para su implementación en *LabVIEW*, tal y como se ha descrito en el capítulo *6. Módulo de recepción y procesado*, tras la comprobación del formato de paquete, se obtienen diferentes campos del mismo entre los que se encuentran el *Service Type* y el *Subservice Type*. El primero de ellos deberá valer 9. El segundo debe tomar valor 1 tal y como se muestra en la siguiente figura:

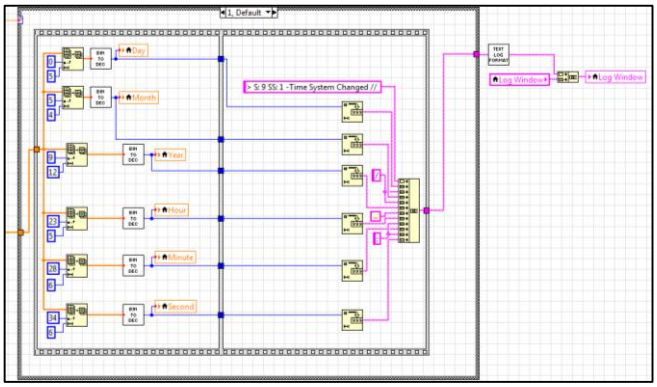


Fig. 12.11. Modificación del sistema temporal

Una vez dentro del *Case* se realizan dos bloques de operaciones de forma puramente secuencial.

En primer lugar, se toman los datos de aplicación del telecomando recibido. Para cada campo se obtiene desde una posición tantos bits como se le pasa por el parámetro *length,* se pasa de binario a decimal con la función creada para dicho propósito y se almacena en la variable local correspondiente. Hay que tener en cuenta el formato descrito al comienzo de esta sección, incluyendo que hay un salto de índice entre el año y la hora aumentado en dos unidades a causa del *Spare* presente entre ambos y que no se almacena en el sistema. Por ejemplo, para el caso de *Day* se cogen los 5 bits menos significativos desde la posición 0, se pasa a decimal y se almacena en su variable local. Dicho proceso se muestra con detalle en la siguiente figura:

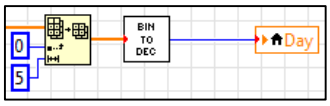


Fig. 12.12. Proceso de extracción del campo Day

En segundo lugar, se informa en *Log Window* que el sistema de tiempos ha cambiado y, se muestra la fecha y hora. Para ello se pasan los números a texto y se concatena el *string* definitivo.

12.3 Time Request (9,2) [TC]

Con el telecomando Time Request (9,2) se solicita el reporte del sistema de tiempo del satélite. No tiene datos de aplicación.

12.4 Time management (9,3) [TM]

Basado en el comando de telemetría (9,2) del estándar [1]. Página 96.

Con la telemetría Time management (9,3) se reporta la fecha y hora del satélite solicitada con el telecomando anterior. El formato de trama es el mismo que en el telecomando Set OBT (9,1).

Referente a su implementación, el proceso seguido es parecido al caso anterior pero a la inversa. En primero lugar, se crea un *array dinámico* en el que se van insertando los diferentes campos: *Day, Month, Year, Spare, Hour, Minute* y *Second*. En cada caso se extrae el valor numérico de la variable local, se pasa de decimal a binario, se cogen el número de bits pertinentes y se van concatenando para formar el *array*. Finalmente se emplea la función *make_packet.vi* a la que se le pasa el *array* formado, el *Service Type* con valor 9 y el *Subservice Type* con valor 3. Como *Sequence Count* se toma el valor 0. De esta forma se genera la telemetría Time management (9,3). Para los dos bits de

relleno se insertan directamente dos ‘0’ en el *array* final. Se debe tener en cuenta que se emplean diferentes funciones según la extensión que se necesite, es decir, para el caso del año se emplea la función para pasar de decimal a binario extendida (devuelve 16 bits en lugar de 8). Como ejemplo, se obtiene el valor numérico *Minute,* se pasa a binario y se cogen del bit 2 al bit 7 (parámetro *length* toma valor 6).

En segundo lugar, se informa de que se ha generado la telemetría en *Log Window.* Todo el proceso descrito se muestra en la figura ***Fig. 12.13. Generación del reporte del sistema temporal del satélite***.

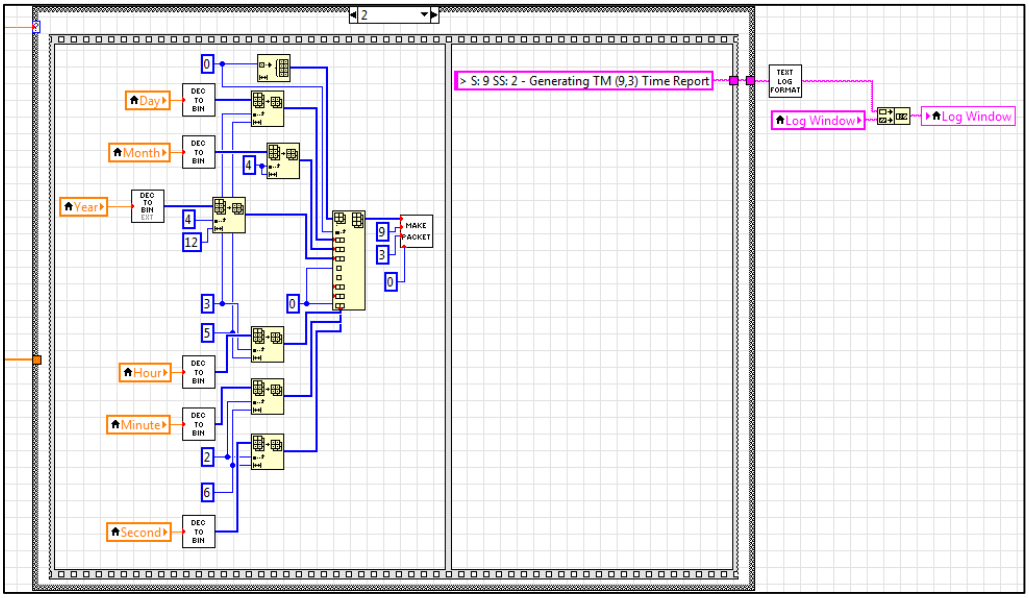


Fig. 12.13. Generación del reporte del sistema temporal del satélite

13. Servicio 11: Gestión del planificador de TC

Este servicio permite a la estación base establecer una lista de telecomandos en el sistema de planificación del satélite para que sean ejecutados en un determinado instante de tiempo de igual modo que si fueran mandados en tiempo real con la diferencia de que se ejecutan de manera automática.

13.1 Funcionamiento del sistema de planificación

En el *Panel Frontal*, dentro de *Main Tab,* se encuentra todo lo referente al servicio de planificación del satélite (pestaña *TC Schedule*), tal y como se muestra en la siguiente figura:

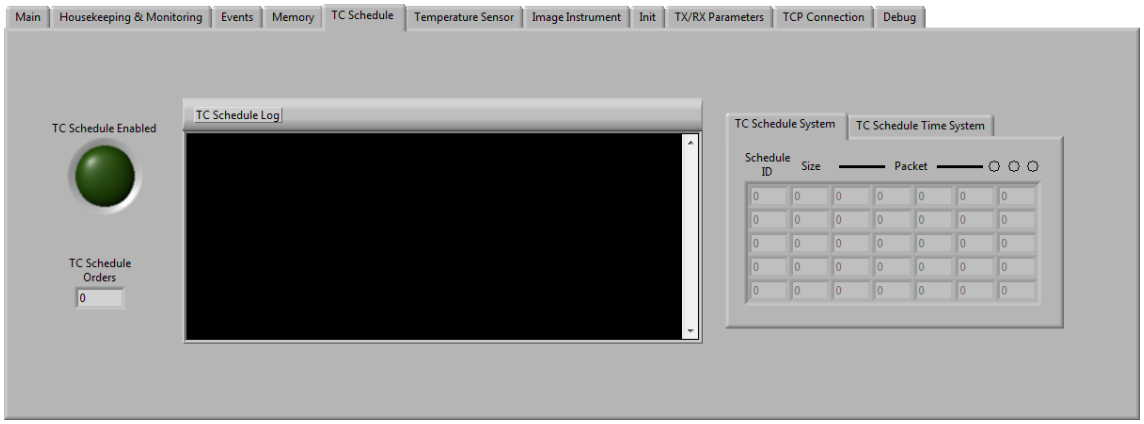


Fig. 13.1. Pestaña TC Schedule

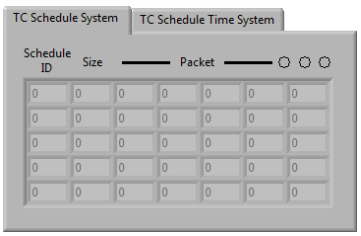


Fig. 13.2. Lista de planificación

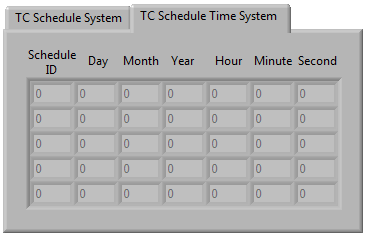


Fig. 13.3. Lista de tiempo asociada

De manera general, para la implementación de este módulo se han establecido dos tablas: *TC Schedule System* y *TC Schedule Time System*. Su estructura, en el mismo orden es la mostrada en las siguientes tablas.

Schedule ID	***Size***	***Packet***
Schedule ID 1	Packet Size 1	Telecomando 1
…		
Schedule ID N	Packet Size N	Telecomando N

Tabla 13.1. Estructura del sistema de planificación

Schedule ID	*Day*	*Month*	*Year*	*Hour*	*Minute*	*Second*
Schedule ID 1	Day 1	Month 1	Year 1	Hour 1	Minute 1	Second 1
…						
Schedule ID N	Day N	Month N	Year N	Hour N	Minute N	Second N

Tabla 13.2. Estructura temporal del sistema de planificación

El campo *Schedule ID* en ambas tablas es el identificador del *Schedule* y puede tomar valor desde 0 hasta 255. El campo *Size* contiene el tamaño del telecomando almacenado y ocupa la segunda columna de la tabla. El campo *Packet* está formado por tantas columnas como indica *Size* y contiene el telecomando al completo en representación binaria.

En lo referente a la segunda tabla, el *Schedule ID* en cada fila es el mismo que el *Schedule ID* la misma fila correspondiente a la otra tabla. Los seis campos siguientes indican en notación decimal el tiempo en que se debe ejecutar el telecomando en concreto.

En cuanto a la implementación de este servicio, está presente en el bucle de recepción y en el bucle de ejecución de telecomandos del sistema de planificación. En el primero se procesan las acciones asociadas al servicio recibidas desde la estación base y en el segundo se ejecutan los telecomandos programados en el sistema de planificación. A continuación se describe el funcionamiento del segundo de ellos, cuya estructura general es la mostrada a continuación:

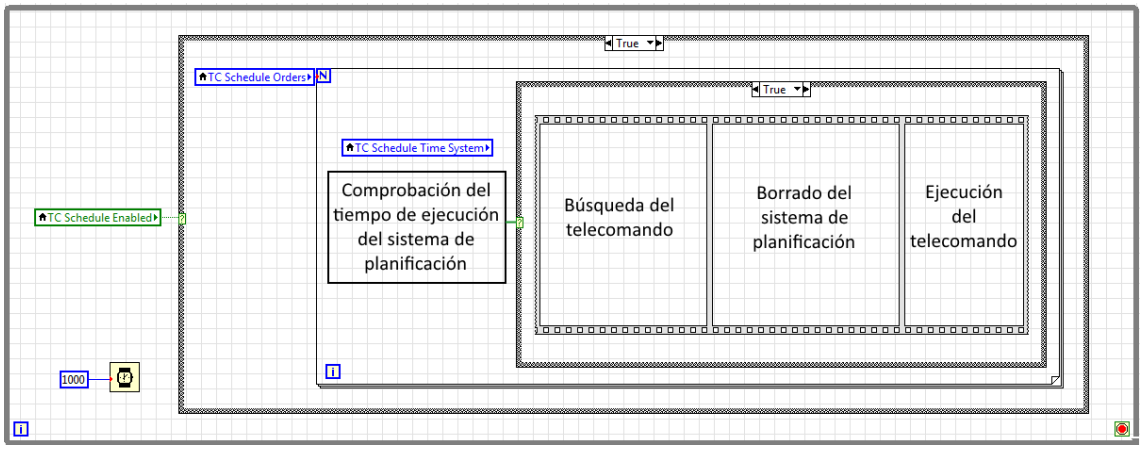


Fig. 13.4. Estructura general de ejecución de telecomandos de la lista de planificación

Como se observa, se emplea un bucle *while* que se ejecuta una vez por segundo (con la función *Wait (ms)*) comprobando en primer lugar el estado del módulo de planificación mediante la variable *TC Schedule Enabled*, que deberá estar a *true*. En ese caso se ejecuta un bucle *for* en el que se van comprobando todos los registros de tiempo de la tabla *TC Schedule Time System,* desde 0

hasta *TC Schedule Orders.* Para realizar esta operación se ejecuta el siguiente código:

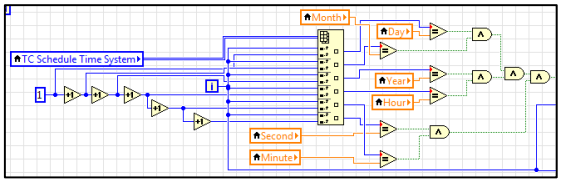


Fig. 13.5. Comprobación del tiempo de ejecución

En la figura anterior se extrae del *array* bidimensional de tiempos cada uno de los registros así como el campo *Schedule ID* asociado de manera que se comprueba que sea igual cada uno de los registros temporales a las variables locales del reloj del satélite. Si todos coinciden se ejecuta el código del caso verdadero del *Case*.

Como se ha ilustrado anteriormente dentro del *Case Structure* tenemos una estructura de secuenciación en la que primero se busca y se extrae del *array* bidimensional *TC Schedule System* el telecomando con *Schedule ID* la misma que en la que ha coincidido el tiempo. Para ello se mira la segunda columna en la misma posición *i* que en la que ha habido coincidencia de tiempo y se extrae dicho valor que es el tamaño del telecomando almacenado (*Size*). Luego se ejecuta un bucle *for* de 0 hasta *Size* y se van concatenando todos los bits del telecomando en cuestión, extrayéndolos desde la posición 2 hasta la posición *Size* más dos del *array TC Schedule System* y se introducen en la variable local *Packet IN*. Todo este proceso se puede ver en la siguiente figura.

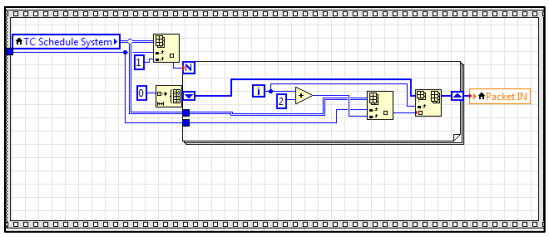


Fig. 13.6. Primera subsecuencia de la ejecución de TC

En la siguiente parte de la estructura secuencial se borran los datos asociados al telecomando que se va a ejecutar del sistema de planificación. Para lograrlo se extraen las filas siguientes a la fila correspondiente a la que se ha ejecutado el telecomando y se insertan en dicha fila, es decir, se baja una posición el resto de filas del *array* bidimensional *TC Schedule System*. Para ello se emplean las funciones *Array Subset* y *Replace Array Subset*. También se extrae las dimensiones de dichos *arrays* mediante las funciones *Array Size* e *Index Array* para posicionar correctamente los nuevos *arrays*. Este mismo proceso de hace para *TC Schedule Time System.* Además, se decrementa en

una unidad el valor del número de elementos del sistema de planificación. En último lugar, en la última parte de la secuenciación se pone a *true* el indicador de recepción de paquete de modo que se ejecuta. Todo lo descrito se ilustra en la siguiente figura:

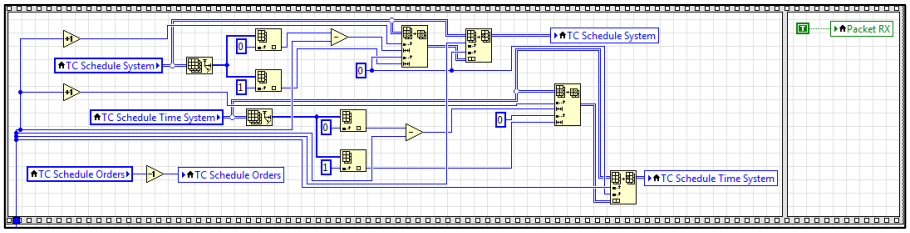


Fig. 13.7. Segunda y tercera subsecuencias de la ejecución de TC

En las siguientes secciones se describe el resto del módulo de planificación.

13.2 Enable TC schedule (11,1) [TC]

Basado en el telecomando (11,1) del estándar [1]. Página 104.

Con este telecomando permitimos la ejecución de la lista de telecomandos del sistema de planificación. Cuando se recibe se activa la ejecución de la lista de telecomandos del sistema de planificación. No tiene datos de aplicación.

Para la implementación de este telecomando se mira dentro del bucle de recepción el tipo de servicio y el subservicio, cuyos valores deben valer 11 y 1 respectivamente. En caso de ser así se ejecuta el siguiente código en el que se pone a *true* el indicador *TC Schedule Enabled* y se registra la operación en *Log Window*.

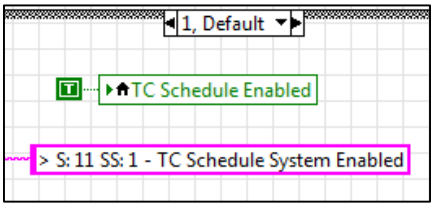


Fig. 13.8. Activación del sistema de planificación

13.3 Disable TC schedule (11,2) [TC]

Basado en el telecomando (11,2) del estándar [1]. Página 104.

Con este telecomando no permitimos la ejecución de la lista de telecomandos del sistema de planificación. Cuando se recibe se desactiva la

ejecución de la lista de telecomandos del sistema de planificación, por lo que no se sigue con su ejecución. No tiene datos de aplicación.

La implementación de este telecomando es idéntica al caso anterior, el subservicio debe valer dos y se pone a *false* el indicador *TC Schedule Enabled.* Luego se registra la operación en *Log Window*, tal y como se muestra a continuación:

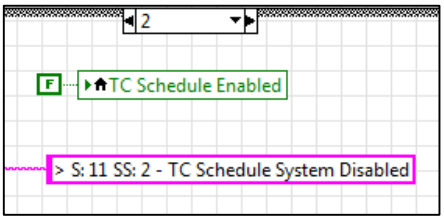


Fig. 13.9. Desactivación del sistema de planificación

13.4 Reset TC schedule (11,3) [TC]

Basado en el telecomando (11,3) del estándar [1]. Página 105.

Con este telecomando se eliminan todos los telecomandos de la lista de telecomandos del sistema de planificación. Cuando se recibe se elimina la lista por completo de manera permanente. La lista queda completamente vacía. No tiene datos de aplicación.

Referente a la implementación de este telecomando, se asignan dos nuevos *arrays* con las mismas dimensiones que los de las variables locales con todos los campos a cero a las variables locales *TC Schedule System* y *TC Schedule Time System*. Luego se registra la operación. Se ejecuta el código de la siguiente figura:

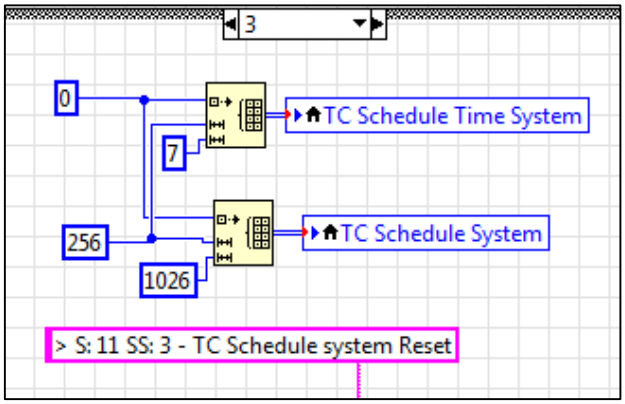


Fig. 13.10. Reinicio del sistema de planificación

13.5 Insert TC in command schedule (11,4) [TC]

Basado en el telecomando (11,4) del estándar [1]. Páginas 105, 106 y 107.

Con este telecomando se inserta un telecomando en la lista del sistema de planificación. Cuando se introduce, pasa a estar integrado en la lista con lo que se ejecuta cuando está definido en el telecomando de inserción del telecomando. Tiene el siguiente formato:

Schedule ID (8)	Abs Time Tag (40)	Telecommand Packet (múltiplo de octeto)
'X'	'X'	'X'

Tabla 13.3. Formato de trama de TC (11,4)

Con este telecomando se inserta otro telecomando que será ejecutado en un cierto tiempo, para lo que se identifica con un *Schedule ID* (8 bits) para poder ser identificado en todo momento. Por lo tanto, se podrá tener en lista del sistema de planificación hasta 256 telecomandos. El campo *Abs Time Tag* contiene un tiempo absoluto (referenciado al sistema de tiempo del satélite) con la estructura del telecomando Set OBT (9,1). Por último el campo *Telecommand Packet* está formado por un número de bits múltiplo de ocho y tiene el formato del telecomando en cuestión (se almacena el paquete al completo).

En cuanto a la implementación de este telecomando, se ejecuta el código de la siguiente figura:

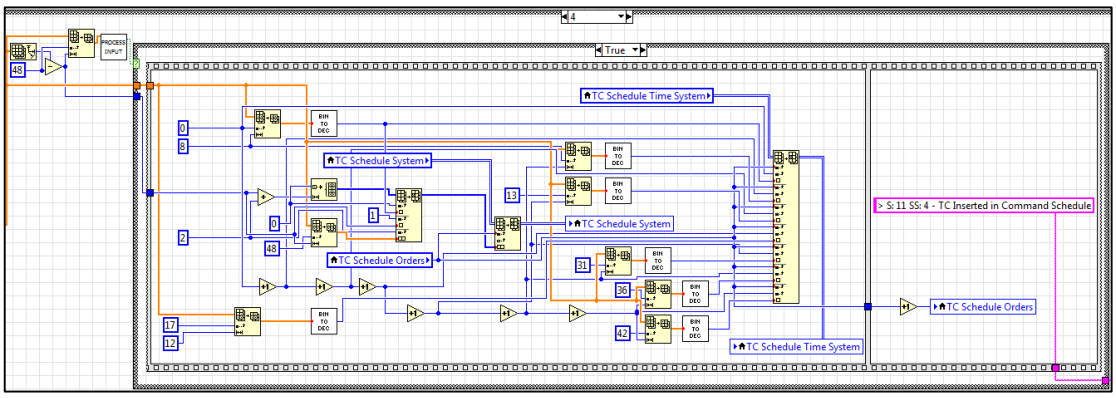


Fig. 13.11. Inserción de un TC en el sistema de planificación

La entrada de los datos de aplicación en este caso contiene un *Schedule ID*, una marca de tiempo con el formato del *Servicio 9* y un telecomando. Tras la comprobación de que el subservicio tiene valor 4, se extrae el telecomando (desde la posición 48 hasta el final de los datos de aplicación) y se le pasa a la función *process_input.vi* para comprobar que el telecomando recibido tiene un

formato correcto. En caso de serlo, se introduce en el sistema de planificación tanto en *TC Schedule System* como en *TC Schedule Time System.* Para lograrlo se concatenan datos en dos *arrays* diferentes y se introducen a sus respectivas variables locales.

En ambos casos primero se introduce el *Schedule ID* cogiendo los 8 primeros bits de *Application Data*. En el caso del sistema temporal se van cogiendo los campos y pasando a decimal con las funciones de transformación *bin_to_dec.vi* y *bin_to_dec_ext.vi,* cogiendo la parte menos significativa con el número de bits correspondiente a cada campo (igual que en el *Servicio 9*). En el caso de *TC Schedule System* en la segunda columna se introduce el tamaño del telecomando calculado anteriormente y a partir de la tercera columna el telecomando al completo. En ambos casos toda la información se introduce en la posición *TC Schedule Orders*, que en el siguiente paso de la estructura secuencial se ve incrementada en una unidad. Luego se registra la operación de inserción en *Log Window*.

13.6 Delete TC in command schedule (11,5) [TC]

Basado en el telecomando (11,5) del estándar [1]. Páginas 107 y 108.

Este telecomando permite borrar de la lista de telecomandos del sistema de planificación un telecomando identificado con su correspondiente *Schedule ID*. Cuando se recibe se borra de forma permanente de la lista. Tiene el siguiente formato:

Schedule ID (8)
'X'

Tabla 13.4. Formato de trama de TC (11,5)

Con este telecomando se elimina por completo de la lista de telecomandos del sistema de planificación el telecomando y tiempo de ejecución asociado a un determinado *Schedule ID* (8 bits). Si está en ejecución, ya no puede ser borrado por lo que se concluye con su ejecución.

Para la implementación del borrado de un determinado *Schedule ID* y sus datos asociados del sistema de planificación se realiza el proceso descrito a continuación que se implementa dentro de una estructura secuencial de tres subestructuras. En primer lugar, se pone la variable *Schedule ID found* a *false*. En la segunda subestructura es donde se borra. Se ejecuta un bucle *for* con condición de parada *Schedule ID found.* Se va extrayendo de cada fila la primera columna que contiene el *Schedule ID* y si coincide con el recibido en el

telecomando de borrado entonces se ejecuta el código del *Case* en el caso verdadero. Para borrar la información asociada se cogen las filas posteriores a las del *Schedule ID* en el que ha habido coincidencia y se bajan una posición. El proceso seguido es el mismo que el explicado al comienzo del servicio en el bucle de ejecución de telecomandos del sistema de planificación de la sección primera de este servicio. A continuación se muestra el proceso descrito:

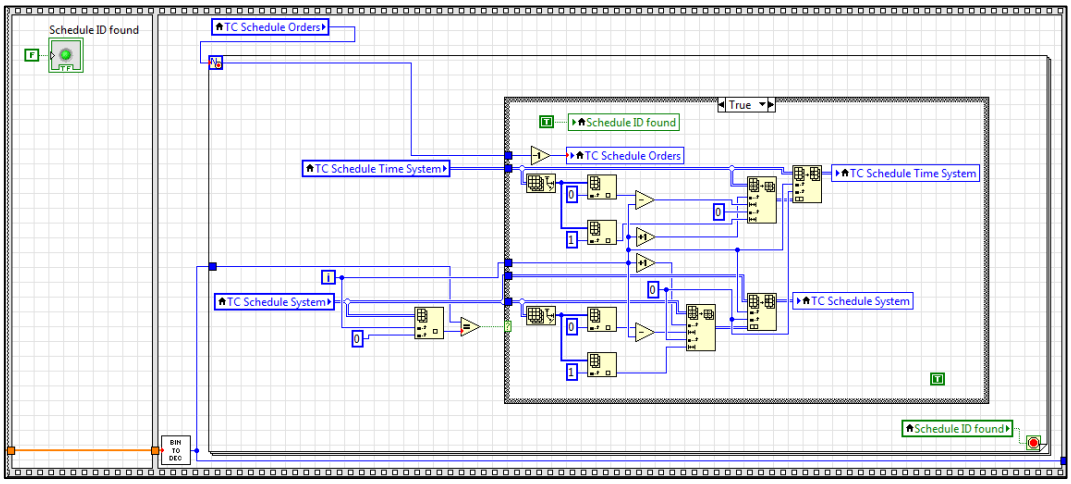


Fig. 13.12. Borrado de un TC del sistema de planificación (primera subsecuencia)

En último lugar si no ha habido coincidencia se muestra un mensaje indicando tal hecho. En caso contrario se muestra la operación llevada a cabo, tal y como se muestra en las siguientes dos figuras.

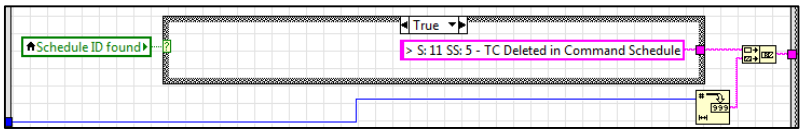


Fig. 13.13. Informe de la operación exitosa (segunda subsecuencia)

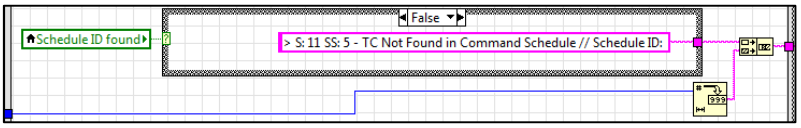


Fig. 13.14. Informe de la operación fallida (segunda subsecuencia)

13.7 Time shift to selected TC (11,6) [TC]

Basado en el telecomando (11,7) del estándar [1]. Páginas 109 y 110.

Este telecomando permite cambiar el tiempo planificado que está almacenado en el satélite de un determinado *Schedule ID* de la lista de planificación. Cuando se recibe este telecomando se modifica el tiempo definido

en el paquete de manera inmediata del tiempo almacenado en la lista de planificación del satélite. Tiene el siguiente formato:

Schedule ID (8)	Abs Time-Shift (40)
'X'	'X'

Tabla 13.5. Formato de trama de TC (11,6)

El campo *Schedule ID* (8 bits) identifica el telecomando de la lista de planificación que es retrasado. El campo *Abs Time-Shift* contiene el tiempo que será retrasado el tiempo almacenado asociado en el sistema de planificación (referenciado al sistema de tiempo del satélite) con la estructura del telecomando Set OBT (9,1).

Para la implementación de este subservicio se sigue una estructura idéntica a la del anterior pero cambiando en la tercera subestructura de la estructura de secuenciación el texto mostrado en el *log*. En la segunda subsecuencia de la estructura secuencial se ejecuta el código de la siguiente ilustración:

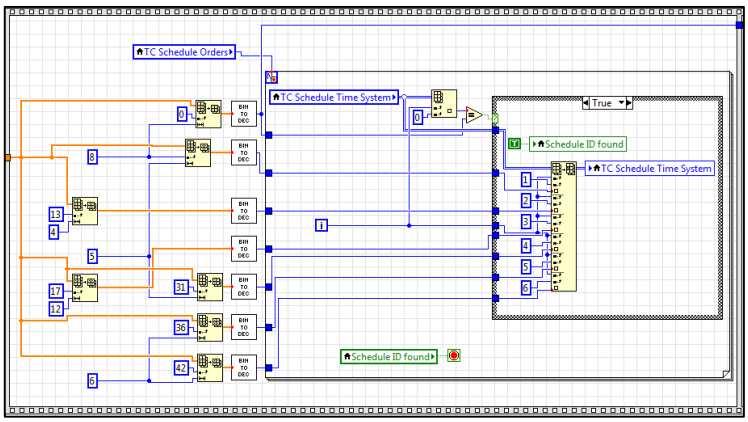


Fig. 13.15. Cambio del tiempo de ejecución de un TC del sistema de planificación

Como se observa, en primer lugar se extraen tanto el *Schedule ID* del que se quiere cambiar el tiempo como los seis campos del sistema temporal. Para ello se emplean las funciones *Array Subset* y *bin_to_dec.vi*. En segundo lugar se ejecuta un bucle *for* con condición de parada *Schedule ID found* un número de iteraciones *TC Schedule Orders.* Se va extrayendo de cada fila de *TC Schedule Time System* la primera columna que contiene el *Schedule ID* y si coincide con el recibido en el telecomando entonces se encuentra coincidencia y en dicha fila *i* se cambian las seis columnas correspondientes al tiempo por sus nuevos valores.

13.8 Report subset of command schedule (11,7) [TC]

Basado en el telecomando (11,9) del estándar [1]. Página 112.

Este telecomando solicita el reporte de un determinado subconjunto de telecomandos de la lista de planificación del satélite. Cada telecomando almacenado en dicha lista tiene asociado un identificador, por lo que se solicitan tantos telecomandos como *Schedule ID* hay presentes en este telecomando (*N*). Tiene el siguiente formato:

N (8)	repetido N veces
	Schedule ID (8)
'X'	'X'

Tabla 13.6. Formato de trama de TC (11,7)

Con el campo *N* se define el número de telecomandos de la lista de la lista de planificación del satélite que quieren ser reportados. El campo *Schedule ID* (8 bits) identifica cada uno de los *N* telecomandos de la lista de planificación que quieren ser reportados.

13.9 Detailed schedule report (11,8) [TM]

Basado en la telemetría (11,10) del estándar [1]. Página 113.

Esta telemetría responde al telecomando anterior Report subset of command schedule (11,7). Se reportan todos los telecomandos de la lista de planificación del sistema con formato válido y existente que hayan sido solicitados con el telecomando Report subset of command schedule (11,7). Tiene el formato mostrado en la siguiente tabla:

N (8)	repetido N veces
	TC Schedule Definition (múltiplo de octeto)
'X'	'X'

Tabla 13.7. Formato de trama de TC (11,8)

El campo *TC Schedule Definition* (múltiplo de 8 bits) contiene un telecomando de la lista de planificación del sistema y tiene el formato mostrado en la ***Tabla 13.8. Formato del campo TC Schedule Definition.***

Schedule ID (8)	Abs Time Tag (40)	TC Packet Length (16)	Service Type (8)	Subservice Type (8)
'X'	'X'	'X'	'X'	'X'

Tabla 13.8. Formato del campo TC Schedule Definition

Para su implementación, de manera general, se sigue la siguiente estructura:

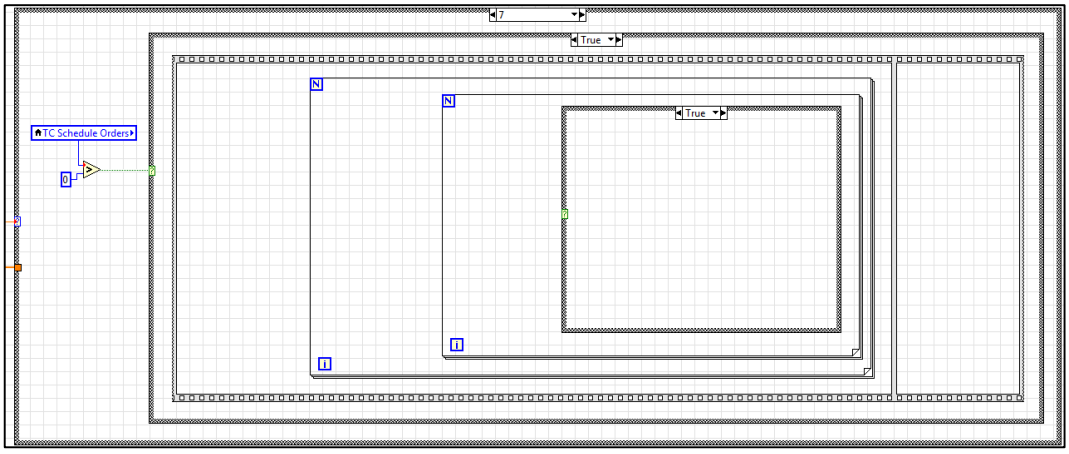


Fig. 13.16. Estructura general del reporte de la lista de planificación

En primer lugar, se comprueba que el valor de *Subservice Type* sea siete. Luego, se comprueba que la lista de planificación no esté vacía (*TC Schedule Orders* mayor que cero). En caso de serlo, se ejecuta una estructura secuencial en dos pasos. Primero se extrae el valor de *N* del telecomando recibido, se pasa a decimal con *bin_to_dec.vi* y se pone *position* a 8. Luego se ejecuta el primer bucle *for* un número de iteraciones igual a *N* en el que se pone a false *report schedule found* y se extrae el valor del *Schedule ID* primero. En sucesivas iteraciones se van extrayendo los siguientes.

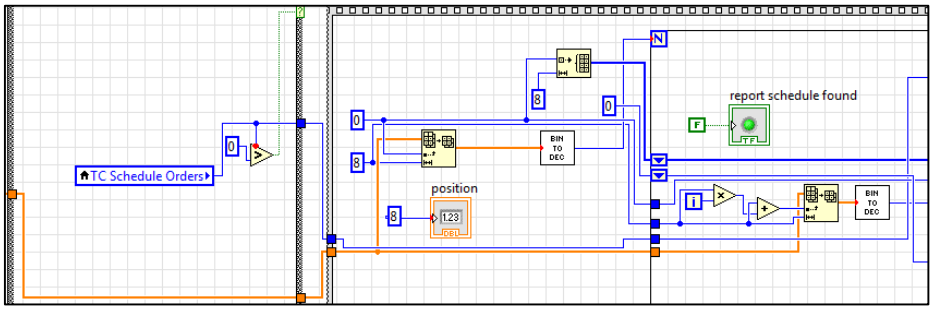


Fig. 13.17. Ejecución del TC (11,7) en la primera subsecuencia

Una vez cogido el *Schedule ID* correspondiente, se ejecuta un bucle *for* para encontrarlo. Se va comparando la columna 0, fila a fila, del *array TC Schedule System* y si se encuentra coincidencia se ejecuta el caso verdadero del *Case*. Como en casos similares se van concatenando los valores del sistema de tiempos del módulo se planificación asociados al *Schedule ID* del que se ha encontrado coincidencia, se pasan a binario con las funciones implementadas y se insertan en el *array* siguiendo el formato de trama anteriormente descrito.

En último lugar, se cogen los dos octetos correspondientes al servicio y subservicio desde la posición 58 del *array* de *TC Schedule System* y se añaden. Todo ese *array* se va concatenando tantas veces como *N Schedule ID* se hayan solicitado y de los que se encuentre coincidencia. Además, se lleva un registro que comienza en cero en el bucle exterior y que va incrementando en uno por coincidencia encontrada para luego ser añadido como *N* al comienzo de la telemetría.

Todo lo descrito se ilustra en la siguiente figura:

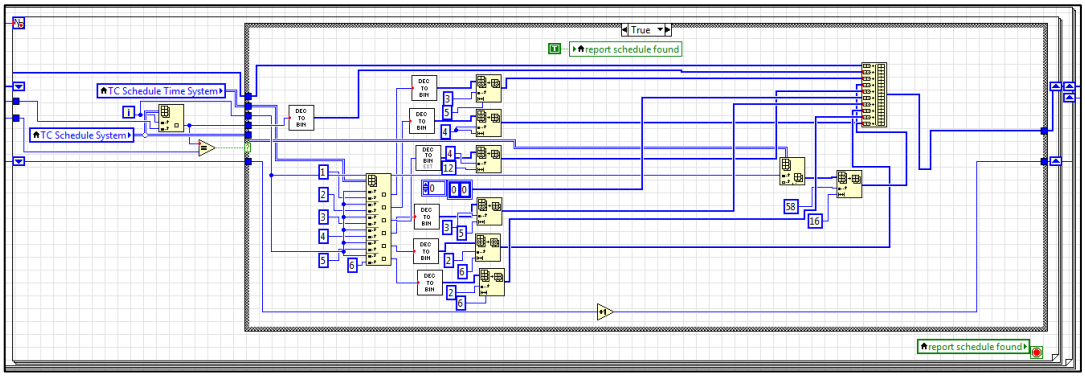


Fig. 13.18. Búsqueda de las estructuras solicitadas (primera subsecuencia)

Por último, se emplea la función *Replace Array Subset* para modificar el primer octeto del paquete introduciendo el valor *N* obtenido como se ha descrito anteriormente. Una vez concluida esta subsecuencia se emplea la función *make_packet.vi* para generar la telemetría pasándole los valores de *Service Type* igual a 11 y *Subservice Type* igual a 8. Luego se registra la operación en *Log Window*.

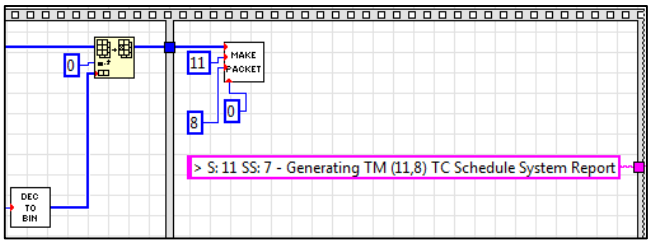


Fig. 13.19. Generación del reporte de la lista de planificación (segunda subsecuencia)

14. Servicio 12: Monitorización de a bordo

La monitorización de abordo realizada con este servicio permite el control de diferentes parámetros de los distintos elementos que se monitorizan que componen el satélite (explicados en el *Servicio 3*) para su posterior análisis en la estación de tierra. Este servicio está estrechamente relacionado con el de *housekeeping* (*Servicio 3* del presente documento). De hecho comparten la tabla definida en dicho servicio para la monitorización y reporte de los parámetros definidos en la tabla.

En lo referente a la implementación de este servicio de manera general, este servicio tiene presencia igual que el servicio tres, en el módulo de recepción y en el de *housekeeping.* La generación de los parámetros referentes al microprocesador y al estado de los instrumentos es el descrito en dicha sección.

14.1 Enable monitoring of parameters (12,1) [TC]

Basado en el telecomando (12,1) del estándar [1]. Páginas 123 y 124.

Con este telecomando se activa la monitorización de un parámetro. En esta implementación se permite activar un parámetro a la vez. Tiene el siguiente formato de trama:

SID (8)	Parameter (3)	Spare (5)
1 ó 9	1 ó 2	'00000'

Tabla 14.1. Formato de trama de TC (12,1)

El campo *Parameter* está constituido por tres bits y representa los parámetros de monitorización que son activados (pone el campo *Enabled* a '1' en la tabla de parámetros de *housekeeping* predefinidos).

Para cumplir la función descrita en este telecomando se ejecuta el código de la siguiente figura:

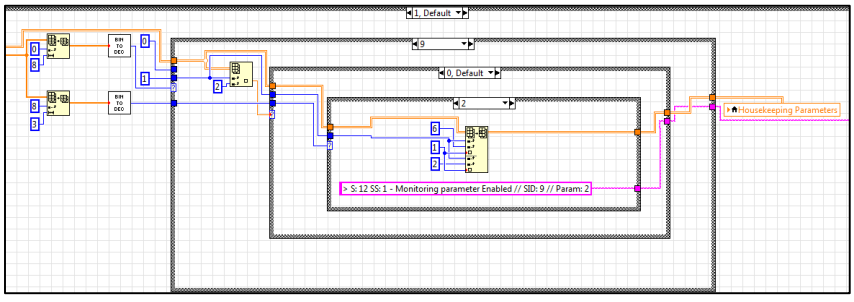


Fig. 14.1. Activación de la monitorización del parámetro 2 (SID 9)

Una vez comprobado el valor de *Subservice Type* se extraen de los datos de aplicación el *SID* (primer octeto) y el parámetro (tres bits siguientes). Para ello se emplea la función *Array Subset* y luego se pasa a decimal con *dec_to_bin.vi.* El siguiente paso es, según el *SID* recibido ejecutar una cláusula del *Case* (1 ó 9) y luego comprobar el valor de *NPAR1* (número de parámetros de monitorización) que tienen el campo *EnabledN* a uno. Dependiendo del número de dicho campo se ejecuta una cláusula distinta. En definitiva, con la función *Array Subset* se cambia el estado de monitorización del parámetro que corresponda. Luego se registra la operación en *Log Window*.

Fuera del Case del subservicio encontramos el siguiente código en el cual se concatena el registro de la operación correspondiente (con formato el devuelto por la función *text_log_format.vi*) a la variable local con el registro total como se ilustra en la siguiente figura:

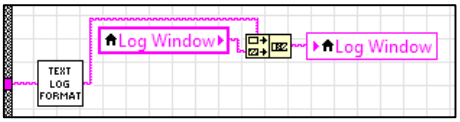


Fig. 14.2. Registro de las operaciones en Log Window

14.2 Disable monitoring of parameters (12,2) [TC]

Basado en el telecomando (12,2) del estándar [1]. Páginas 123 y 124.

Con este telecomando se desactiva la monitorización de un parámetro. En esta implementación se permite desactivar un parámetro a la vez. La estructura es la misma que en el telecomando anterior Enable monitoring of parameters (12,1).

La implementación de este telecomando es idéntica al anterior pero poniendo un cero en el estado de monitorización correspondiente, tal y como se muestra a continuación:

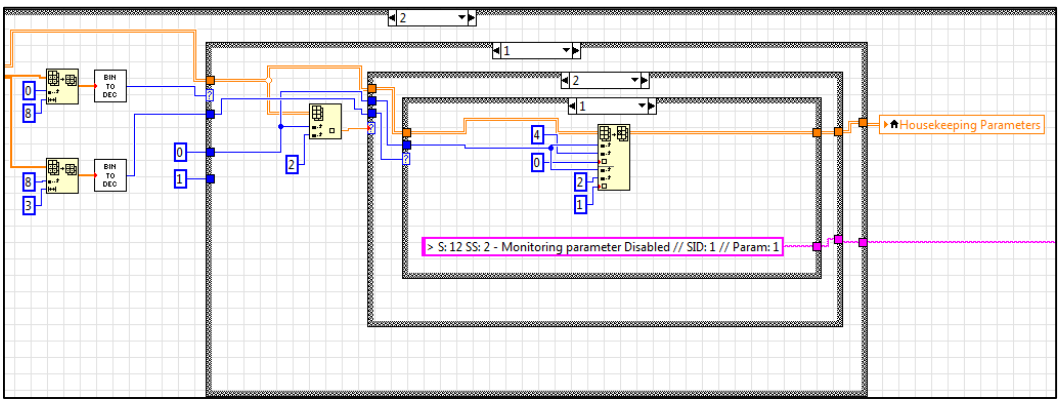


Fig. 14.3. Desactivación de la monitorización del parámetro 1 (SID 1)

14.3 Clear monitoring list (12,4) [TC]

Basado en el telecomando (12,4) del estándar [1]. Página 124.

Con este telecomando se eliminan todos los parámetros de la lista así como sus estados de monitorización (campo *Enabled* en la tabla de parámetros de *housekeeping* predefinidos). No tiene datos de aplicación.

La implementación de este telecomando es sencilla. Se coge la variable local *Housekeeping Parameters* y se ponen las columnas *Enabled SID RG, NPAR1, Param1, Enabled1, Param2, Enabled2* y *Period* a cero para ambos *SID*, y se registra la operación.

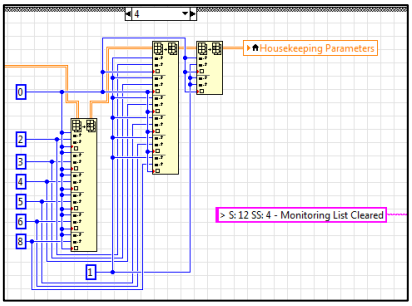


Fig. 14.4. Reinicio del bloque de definiciones de monitorización

14.4 Add parameter to monitoring list (12,7) [TC]

Basado en el telecomando (12,5) del estándar [1]. Páginas 125, 126 y 127.

Con este telecomando se añade un parámetro a la lista de monitorización (campos *Parameter* de la tabla de parámetros de *housekeeping* predefinidos). En esta implementación se permite añadir un parámetro a la vez y están predefinidos. La estructura es la misma que en el telecomando Enable monitoring of parameters (12,1).

En lo referente a la implementación de este telecomando, se ejecuta el código mostrado en la siguiente figura.

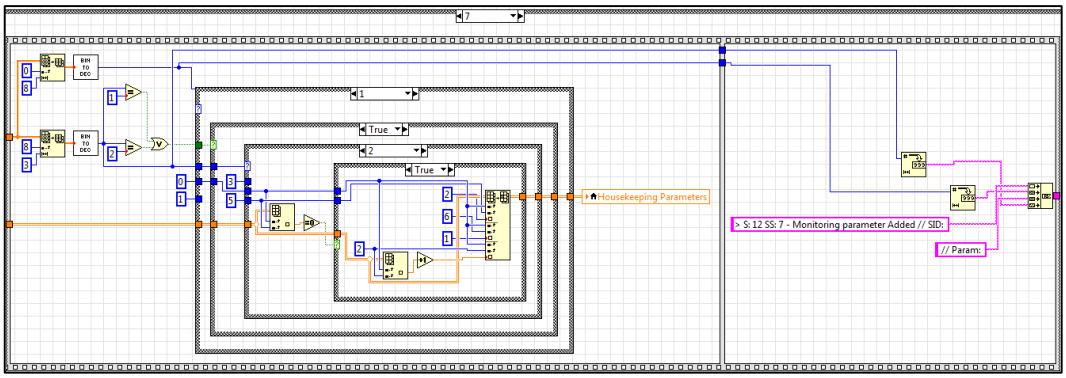


Fig. 14.5. Inserción de parámetro en la lista de monitorización

Como puede observarse en primer lugar se extraen de los datos de aplicación el *SID* (primer octeto) y el parámetro (tres bits siguientes). Para ello se emplea la función *Array Subset* y luego se pasa a decimal con *dec_to_bin.vi.* Se comprueba que el *SID* sea válido y luego se entra en una sucesión de *Case Structure* en los que se coloca en la fila correspondiente al *SID* recibido en la columna de *ParameterN* (según el parámetro recibido) el *monitoring parameter*. Además, se pone a uno la columna de *EnabledN* correspondiente por lo que se comienzan a generar los datos de monitorización. También se suma una unidad a la columna de *NPAR1* por la activación de generación de datos de monitorización de dicho *SID*. En último lugar se registra la operación.

14.5 Delete parameter to monitoring list (12,8) [TC]

Basado en el telecomando (12,6) del estándar [1]. Página 127.

Con este telecomando se elimina un parámetro a la lista de monitorización (campos *Parameter* de la tabla de parámetros de *housekeeping* predefinidos). En esta implementación se elimina un parámetro a la vez y están predefinidos. La estructura es la misma que en el telecomando Enable monitoring of parameters (12,1).

El proceso de ejecución para este telecomando es exactamente igual que en el caso anterior salvo por la diferencia de que se ponen a cero las columnas pertinentes (se decrementa en una unidad *NPAR1*, y se ponen a cero el *ParameterN* y el *EnabledN* correspondientes). El código es el mostrado en la figura ***Fig. 14.6. Borrado de parámetro en la lista de monitorización*.**

14.6 Report current monitoring list (12,9) [TC]

Basado en el telecomando (12,8) del estándar [1]. Página 129.

Con este telecomando se solicita el reporte de la lista de monitorización. No tiene datos de aplicación.

14.7 Current monitoring list report (12,10) [TM]

Basado en la telemetría (12,9) del estándar [1]. Páginas 130 y 131.

Esta telemetría responde al telecomando anterior Report current monitoring list (12,9). En ella se incluye la tabla de parámetros de *housekeeping* predefinidos.

NS (8)	Spare (6)	Repetido NS veces			
		SID (8)	NPAR1 (3)	Parameter 1 (3)	Enabled1 (1)
‘X’	‘000000’	1 ó 9	0, 1 ó 2	‘X’	‘X’

Repetido NS veces		
Parameter2 (3)	Enabled2 (1)	Collection interval (18)
‘X’	‘X’	‘X’

Tabla 14.2. Formato de trama de TM (12,10)

El primer campo *NS*, formado por 7 bits, indica el número de identificadores de estructura (*SID*) con sus respectivos *NPAR1*, parámetros, estados e intervalos de recolección asociados. El campo *SID* está formado por un octeto e identifica la fuente de la que se realiza la monitorización. Los siguientes 3 bits indican el número de parámetros y estados asociados que se envían de la lista de reporte. Los dos siguientes campos se repiten *NPAR1* veces e incluyen, los primeros 3 bits el identificador del parámetro reportado y el bit siguiente su estado asociado. Los siguientes 18 bits establecen el periodo de recolección de datos referente al *SID*.

En lo referente a la implementación de esta sección su implementación es la mostrada en la figura ***Fig. 14.7. Reporte de la lista de monitorización***. Es muy parecida a sucesivos casos de este documento. Se coge la lista de *housekeeping* (*array* bidimensional *Housekeeping Parameters*) y se van pasando todos los campos a binario y cogiendo el número de bits pertinente. Luego, al inicio del *array* se reemplaza el primer octeto por la representación binaria de *NS* obtenida mediante un registro de desplazamiento en el bucle. Una vez formado el *array* binario se introduce en la función *make_packet.vi* junto con *Service Type* con valor 12, *Subservice Type* con valor 10 y *Sequence Number* con valor cero. Por último se registra la operación en *Log Window*.

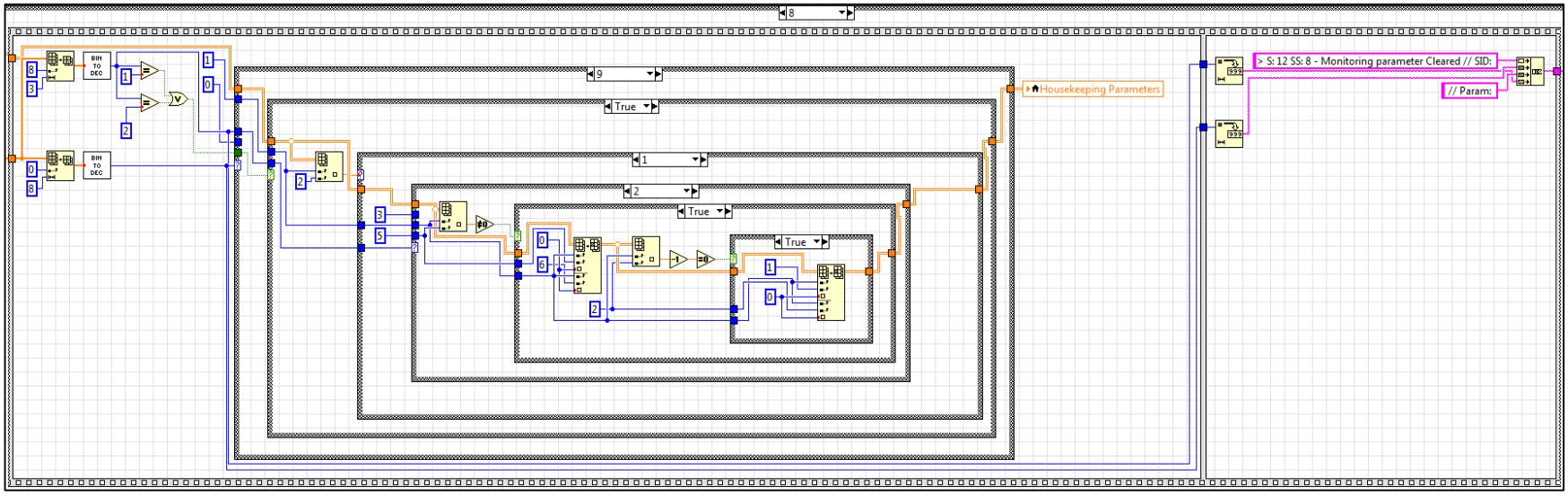


Fig. 14.6. Borrado de parámetro en la lista de monitorización

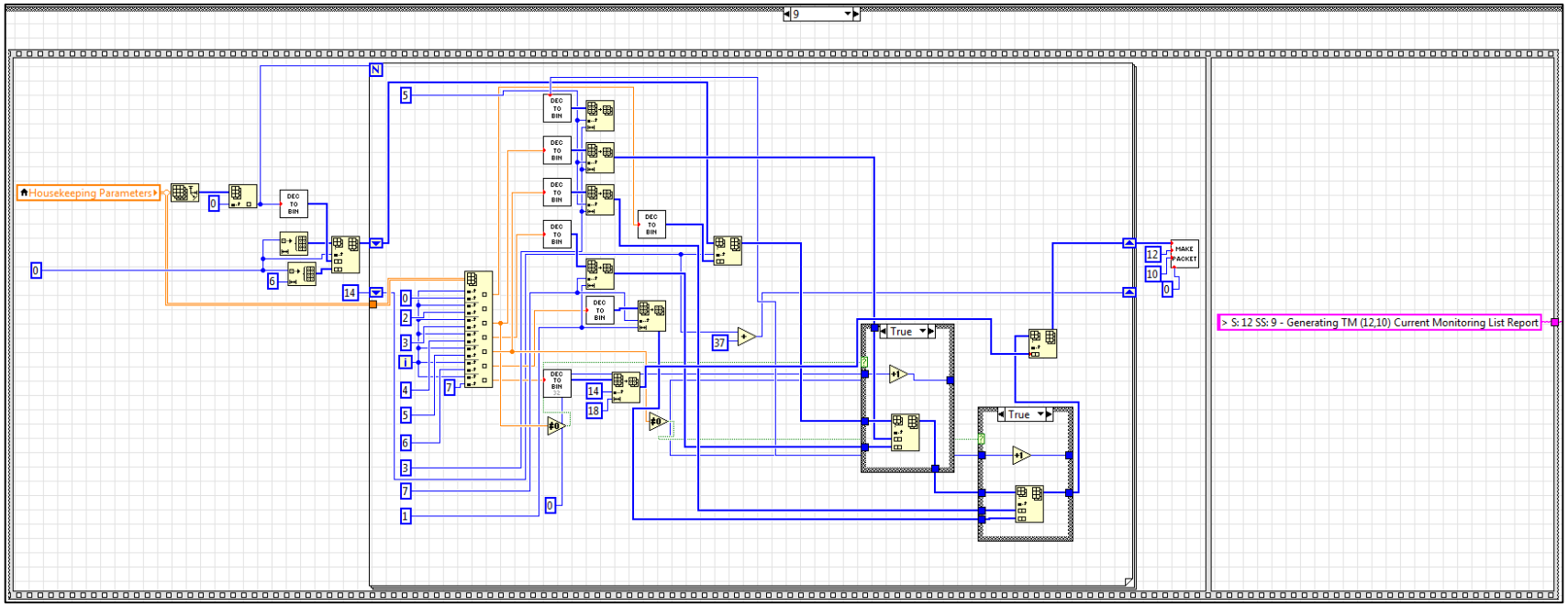


Fig. 14.7. Reporte de la lista de monitorización

15. Servicio 15: Test de conexión

Este servicio es el más elemental y uno de los más importantes de todos los implementados porque permite conocer el estado de la conexión entre estación base y satélite.

15.1 Request Connection Test (15,1) [TC]

Basado en el telecomando (17,1) del estándar [1]. Página 167.

Con este telecomando se solicita un test de conexión. Será inmediatamente respondido con la telemetría Connection Test Report (15,2). Ninguno tiene datos de aplicación.

15.2 Connection Test Report (15,2) [TM]

Basado en el comando de telemetría (17,2) del estándar [1]. Página 167.

La implementación de este módulo es sencilla. En primer lugar, se comprueba el correcto formato del telecomando así como el valor de los campos de *Service Type* y *Subservice Type* (deben valer 15 y 1 respectivamente). En caso de cumplirse se genera el reporte de test de conexión solicitado con la función *make_packet.vi* pasándole como parámetros *Service Type* con valor 15 y *Subservice Type* con valor 2, y Sequence Count valiendo cero. Luego se registra dicho envío en *Log Window* de manera idéntica a casos anteriores. El código es el mostrado en la figura siguiente:

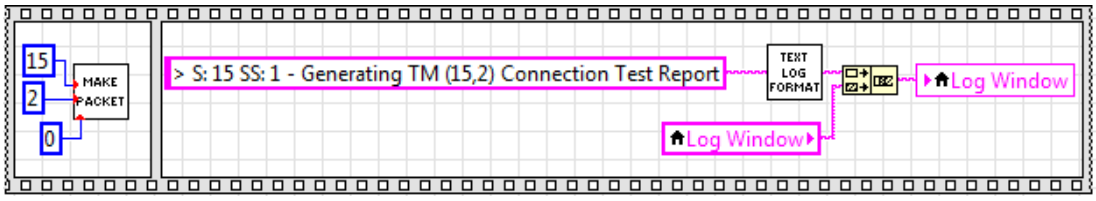
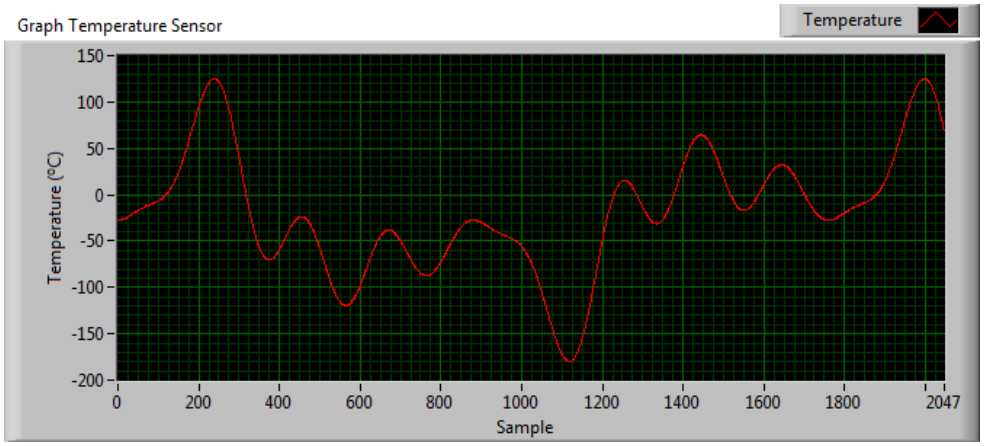


Fig. 15.1. Generación de la telemetría de test de conexión

16. Servicio 32: Instrumento sensor de temperatura

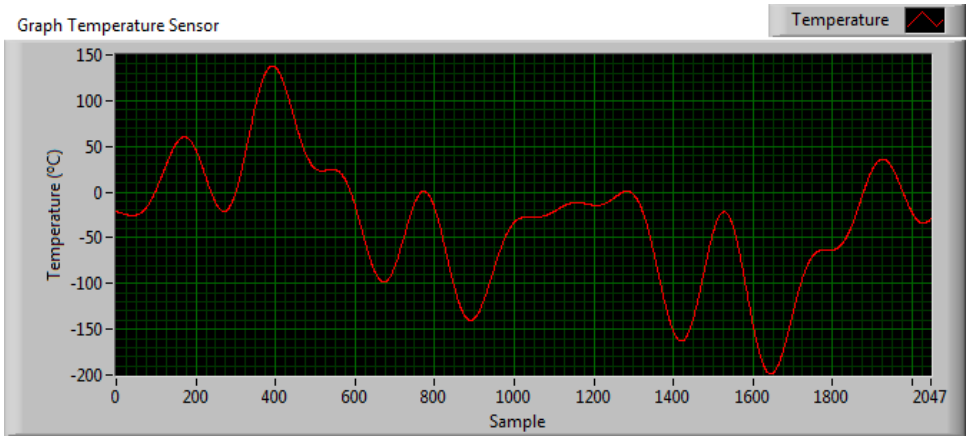
Este servicio se corresponde con el manejo de uno de los instrumentos del satélite: el sensor de temperatura. Con este instrumento se simula la toma de muestras de temperatura en el espacio, permitiendo encender y apagar dicho instrumento, modificar el tiempo de adquisición de muestras, el reporte de los datos obtenidos y la interrupción de la transmisión, además de poder comprobar el funcionamiento del instrumento.

Para la simulación de la temperatura se emplean dos funciones en las que se simulan períodos de exposición solar y de ocultación. Como se describe más adelante, durante la toma de muestras luego se genera una función aleatoria siguiendo el mismo modelo y con la posibilidad de introducir ruido. Las funciones empleadas de manera inicial son dos, mostradas a continuación:



$$55{,}45 \times sin(\frac{2 \times t}{70} + 1) + sin(\frac{t}{70} - 1) + 2 \times sin(\frac{t}{140}) \times sin(\frac{t}{280}) - 27{,}5$$

Fig. 16.1. Función de temperatura número uno y su representación



$$55{,}45 \times \sin\left(\frac{3 \times t}{100} + 3\right) + \sin\left(\frac{t}{50} - 1\right) + 2 \times \sin\left(\frac{t}{200}\right) \times \sin\left(\frac{t}{350}\right) - 21{,}34$$

Fig. 16.2. Función de temperatura número dos y su representación

La temperatura en el sensor se toma en grados centígrados y para las funciones empleadas se ha establecido una oscilación de entre -200ºC y 150ºC en el período de un día. En condiciones normales la temperatura en el espacio oscila entre normalmente entre -180ºC y 122ºC [27].

16.1 Funcionamiento del instrumento

Dentro del *Panel Frontal*, el instrumento sensor de temperatura se encuentra en la pestaña *Temperature Sensor* que tiene el siguiente aspecto:

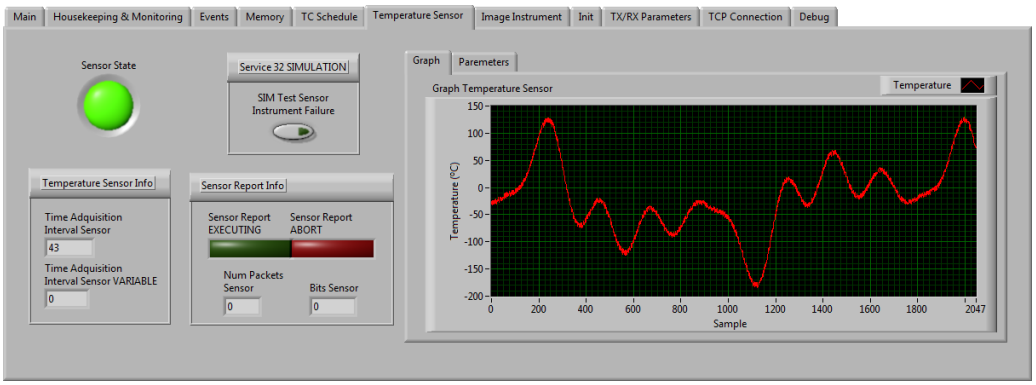


Fig. 16.3. Pestaña Temperature Sensor

En esta pestaña se incluye un LED con el estado de actividad (indica si el instrumento está en funcionamiento o no), información del instrumento (tiempo de adquisición de las imágenes y del provisional como se explica más adelante), un botón para simular fallo en el instrumento y un elemento *Tab Control* con una estaña en la que se incluye un gráfico con la evolución de la temperatura y otra con la posibilidad de introducir ruido. También se presenta información del proceso de creación de paquetes de reporte (un LED como que se ilumina en verde si el reporte está en ejecución, otro que se ilumina en rojo en caso de abortar el reporte, el número de paquetes necesarios para reportar la temperatura y el número de bits que ocupan las 2047 muestras de temperatura). Tanto el tamaño total de bits que se mandan como el número de paquetes no se actualizan hasta que se empiezan a mandar los datos.

En lo referente a la implementación de la generación de las imágenes, dentro del módulo de generación de datos de instrumentos encontramos la estructura de la siguiente figura.

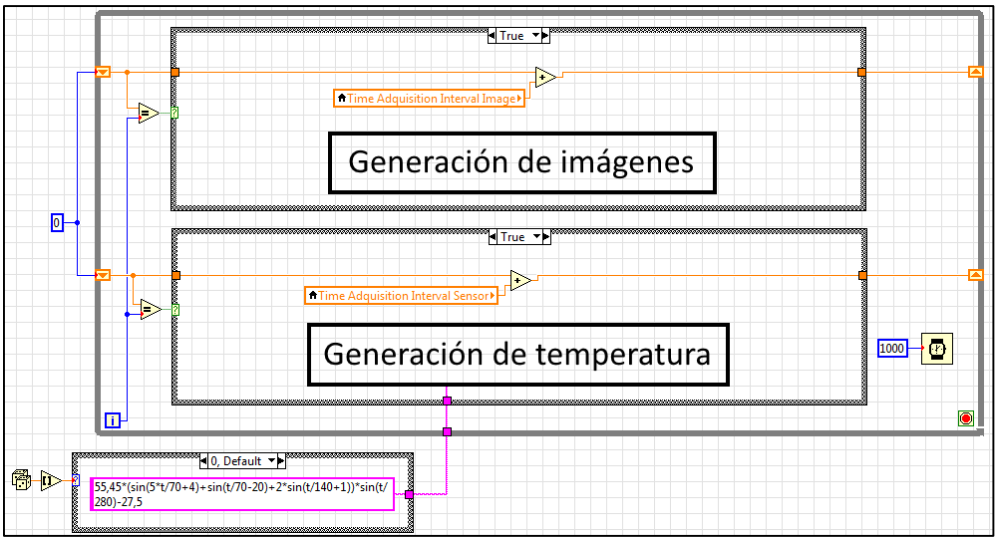


Fig. 16.4. Estructura del módulo de generación de datos de instrumentos

Como se observa, se ejecuta el bucle una vez por segundo de modo que si la iteración *i* es igual al registro se ejecuta el código que está dentro del caso verdadero. En dicho caso se suma al registro la variable local *Time Adquisition Interval Image (s)*, esto es, el número de segundos que deberán pasar hasta la siguiente ejecución de este caso (el tiempo que ha de pasar hasta la adquisición de una nueva imagen). En el caso falso, simplemente se vuelve a registrar el mismo valor de la entrada. Para la ejecución cada un segundo se emplea el elemento *Wait (ms)* al que se le pasa por parámetro el valor 1000 (milisegundos). Este mismo proceso es el empleado en el bloque de generación de imágenes.

Además, para la generación de las muestras de temperatura se conecta un *string* con la función que será generada durante la ejecución del programa y que puede ser una de las vistas al comienzo de esta sección. Para su elección se coge un valor aleatorio entre 0 y 1 y se redondea, con la funciones *Random Number 0-1* y *Round To Nearest*. Luego se introduce en un *Case* en el que se incluyen en cada caso una de las dos funciones.

Dentro del bloque de generación de temperatura se ejecuta lo siguiente:

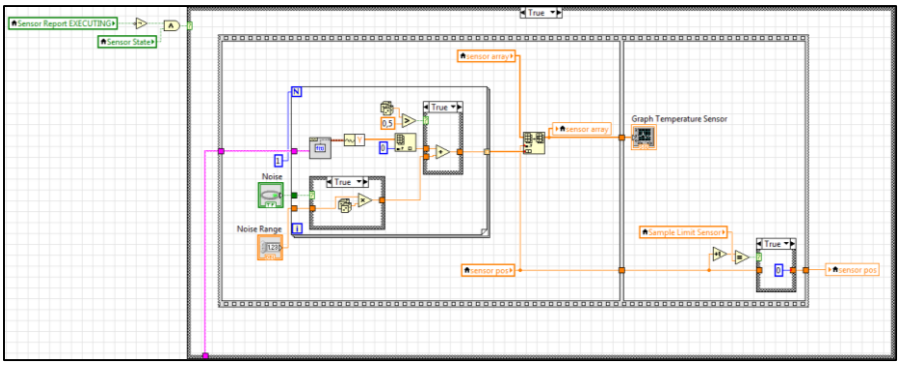


Fig. 16.5. Generación de la función de temperatura

Como se observa hay una estructura *Case* en la que se comprueba que el estado del instrumento sea encendido (*Sensor Instrument State* con valor *true*) y que no se está reportando la temperatura en ese momento (*Sensor Report EXECUTING*).

Su funcionamiento dentro de la estructura secuencial es el siguiente. En primer lugar, se ejecuta un *for* en el que se van indexando los valores que se van a introducir en el *array* del sensor. Se genera la función que se pasa por parámetro desde el *string* exterior empleando *Formula Waveform.vi*, la cual devuelve los datos en formato *Waveform(DBL).*

Luego se selecciona el *array* de datos empleando la función *Get Waveforms Components* y con la función *Index Array* se selecciona el primer valor. El siguiente paso es coger ese valor y, en caso de estar pulsado el botón *Noise* se coge un valor aleatorio entre 0 y *Noise Range* y luego se le suma o resta de manera aleatoria al valor de la función. Si el valor aleatorio es mayor de 0.5 se suma, en caso contrario se resta. Una vez obtenido el valor definitivo se introduce en el *array* que contiene los datos del sensor (*sensor array*) en la posición almacenada en la variable local *sensor pos.*

En último lugar, se representan los datos en el gráfico, se le suma una unidad a *sensor pos* y se comprueba si sigue dentro del límite de muestras (2047). En caso de llegar al límite, se pone *sensor pos* a cero de modo que los datos se van sobrescribiendo constantemente.

16.2 Switch on instrument (32,1) [TC]

Con este telecomando se enciende el instrumento sensor de temperatura. No tiene datos de aplicación.

Su implementación es muy sencilla. En primera instancia, dentro del módulo de recepción se comprueba el correcto formato del telecomando entrante. Luego se comprueban los valores de *Service* Type y *Subservice Type* que deben valer 32 y 1 respectivamente. Dentro del *Case* que identifica el subservicio se pone el estado *true* del indicador *Sensor Instrument State* y se registra en *Log Window.* El código es el mostrado en la siguiente figura:

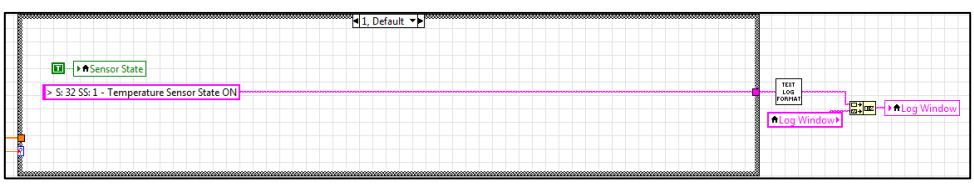


Fig. 16.6. Encendido del instrumento sensor de temperatura

16.3 Switch off instrument (32,2) [TC]

Este telecomando apaga el instrumento sensor de temperatura. No tiene datos de aplicación.

La implementación de este telecomando es casi idéntica a la del telecomando anterior. Se ejecuta el código del caso en el que *Subservice Type* toma valor 2, en cuyo caso se introduce un *false.* A continuación se muestra dicha implementación:

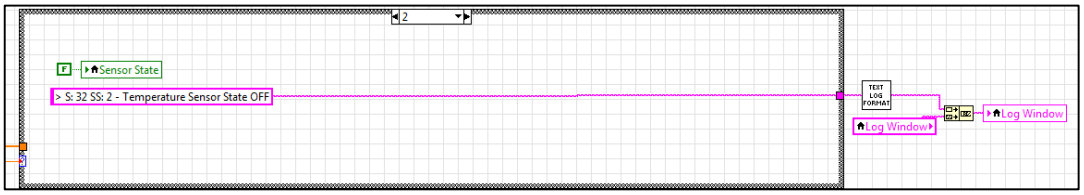


Fig. 16.7. Apagado del instrumento sensor de temperatura

16.4 Change time acquisition interval (32,3) [TC]

Con este telecomando se guarda un nuevo intervalo de adquisición de datos de temperatura del instrumento en memoria, que no será actualizado hasta recibir el telecomando Update time acquisition interval (32,6). El formato de trama es el siguiente:

Time acq interval (16)
'X'

Tabla 16.1. Formato de trama de TC (32,3)

El campo *Time acq interval* está formado por dos octetos cuyo rango va desde 1 hasta 65535 segundos. Constituye el tiempo de adquisición de las muestras de temperatura del sensor.

Referente a su implementación, tras la comprobación del valor del subservicio, se pasan los 16 bits de datos de aplicación a decimal con la función *bin_to_dec.vi* y se almacenan en la variable local *Time Adquisition Interval Sensor VARIABLE* a la espera de recibir el telecomando de actualización del periodo de adquisición. Luego se registra la operación de igual modo que en casos anteriores. El código es el mostrado en la siguiente figura.

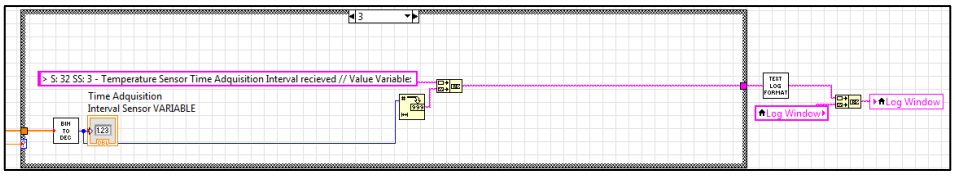


Fig. 16.8. Guardado de tiempo de adquisición temporal del sensor

16.5 Request time acquisition interval (32,4) [TC]

Con este telecomando se solicita el reporte del intervalo de adquisición de datos de temperatura del instrumento. No tiene datos de aplicación.

16.6 Time acquisition interval report (32,5) [TM]

El telecomando anterior se responde con la telemetría Time Acquisition Interval Report (64,5), incluyendo el intervalo de adquisición de imágenes del instrumento. El formato de trama es idéntico al del telecomando Change Time Acquisition Interval (64,3).

Este servicio se implementa como se describe a continuación. Tras la comprobación de que los valores de *Service* Type y *Subservice Type* valen 64 y 4 respectivamente, se coge el valor de la variable local *Time Adquisition Interval Sensor* y se pasa a binario con la función *dec_to_bin_ext.vi.* Esta representación binaria se introduce en la función *make_packet.vi* junto con los valores de servicio igual a 32 y subservicio igual a 5. De este modo se crea la telemetría respuesta. El valor de *Sequence Number* es cero. Luego, se registra la operación.

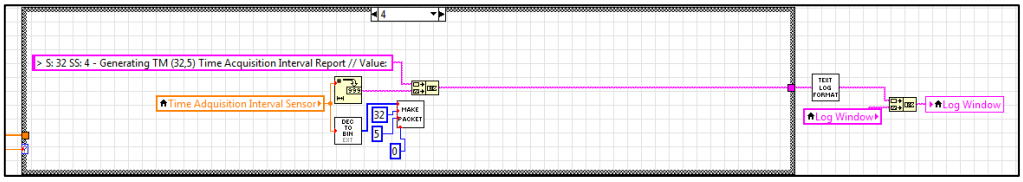


Fig. 16.9. Generación del reporte del tiempo de adquisición del sensor

16.7 Update time acquisition interval (32,6) [TC]

Con este telecomando se actualiza el intervalo de adquisición de datos del instrumento sensor de temperatura que previamente se había mandado con el telecomando Change time acquisition interval (32,3). No tiene datos de aplicación.

La implementación de este telecomando es simple. Una vez comprobado el valor del subservicio, se almacena el valor de la variable local *Time Adquisition Interval Sensor VARIABLE* en la variable local *Time Adquisition Interval Sensor* y se registra la operación en *Log Window*.

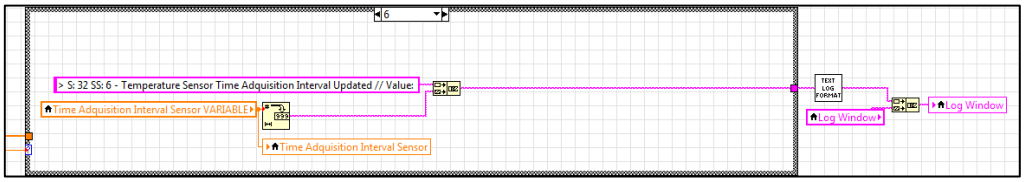


Fig. 16.10. Actualización del tiempo de adquisición del sensor

16.8 Request data (32,7) [TC]

Con este telecomando se solicitan los datos tomados por el instrumento sensor de temperatura. No tiene datos de aplicación.

16.9 Data from instrument (32,9) [TM]

Con esta telemetría se responde al telecomando anterior, Request data (32,7), incluyendo el intervalo de adquisición de datos de temperatura del instrumento y los datos de temperatura sensados. El formato de trama es el siguiente:

Time acq interval (16)	Samples Number (11)	Spare (5)	Data instrument (14)
'X'	'X'	'00000'	'X'

Tabla 16.2. Formato de trama de TM (32,9)

El campo *Time acq interval* contiene el tiempo de adquisición de las muestras y está formado por dos octetos. *Samples Number* indica el número de muestras que van en el paquete. El campo *Spare* hace la función de relleno (todo a cero). Con 11 bits de *Samples Number* podríamos mandar hasta 2047 muestras de temperatura en un mismo paquete. En la práctica se enviarán hasta 46 en un mismo paquete. El campo *Data instrument* contiene las medidas de los sensores, siendo cada catorce bits una medida. Cada muestra de temperatura tiene el siguiente formato:

Sign (1)	Decimal Sample Part (9)	Fractional Sample Part (4)
'0'/'1'	'X'	'X'

Tabla 16.3. Representación de muestra de temperatura del sensor

Donde el primer bit es de signo (si es positivo valdrá ‘0’; en caso de ser negativa la temperatura, valdrá ‘1’). *Decimal Sample Part* contiene la parte entera de la muestra y el campo *Fractional Sample Part* representa la parte fraccional. Según los límites establecidos anteriormente para la temperatura en el espacio con 9 bits se pude representar un rango de temperaturas (en grados centígrados) de [-511, 511]. En ningún caso se baja de -273,1ºC y el rango empleado oscilará entre -200ºC y 175ºC.

Para la ejecución de esta sección, se debe tener en cuenta que en primer lugar se recibe el telecomando en el módulo de recepción y procesado y se activa la ejecución del reporte del módulo de generación de reportes de datos de instrumentos. En el bucle de recepción se ejecuta el siguiente código:

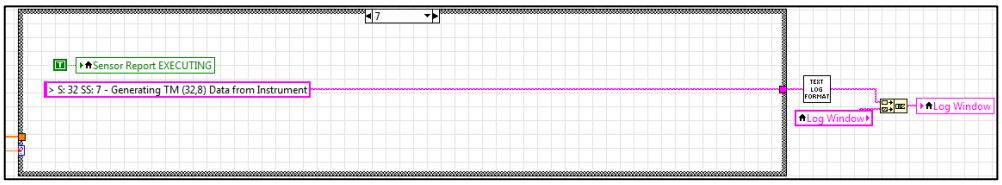


Fig. 16.11. Activación del reporte de datos del sensor

Como se observa, únicamente se pone a *true* la variable local *Sensor Report EXECUTING* y se informa de que se procede a la generación del reporte.

Dentro del módulo de generación de reportes de datos de instrumentos se implementa la estructura de la siguiente figura:

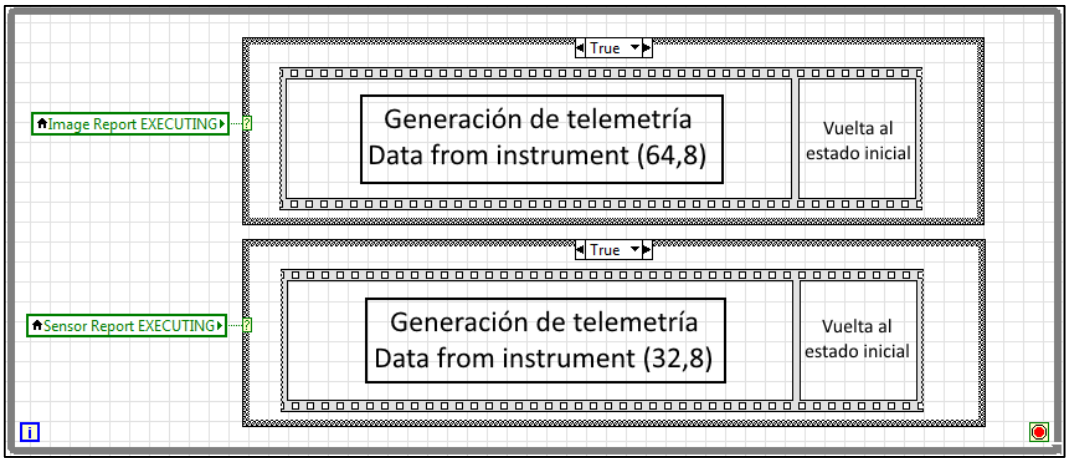


Fig. 16.12. Estructura del módulo de generación de reportes de datos de instrumentos

Como se puede observar, una vez activado el reporte de imágenes se procede a la ejecución del caso *true,* dentro del cual hay una estructura secuencial. En primer lugar se generan todos los paquetes y se mandan, y en

segundo lugar se ponen a *false* las variables locales *Sensor Report EXECUTING* y Sensor *Report ABORT*, tal y como se muestra a continuación:

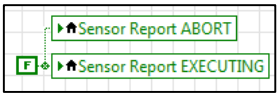


Fig. 16.13. Segunda subsecuencia del reporte de temperatura

Anteriormente se han mandado todos los paquetes salvo que haya llegado un telecomando Abort Data transfer (32,9), en cuyo caso se interrumpe la transmisión. La implementación de dicho proceso se puede ver en la página 114.

Dentro de la estructura de secuenciación, en primer lugar se calcula el número de bits de todas las muestras de temperatura (se multiplica el tamaño de *sensor array*, que por defecto es 2047, por 14 -número de bits de representación de cada muestra-) y se divide por 644 que es tamaño máximo de bits que se pueden incluir en cada paquete y se redondea hacia arriba. De este modo se calcula el número de paquetes totales del reporte de temperatura.

El siguiente paso es obtener un *array* con todas las muestras pero en representación binaria para lo cual se ejecuta un *for* en el que se realiza la conversión muestra a muestra. Se van extrayendo los datos del *array* con la función *Index Array* y se van concatenando el *array* en representación binaria, el signo de la siguiente muestra, la parte entera y la parte decimal. Para la obtención del signo se comprueba si el valor es mayor o igual a cero. En caso de serlo se introduce un cero y en caso contrario un uno. Para la parte entera se pone la muestra en valor absoluto y se redondea hacia abajo. Luego se pasa a binario con la función *dec_to_bin_ext.vi* y se cogen los 9 bits menos significativos. Para obtener la parte decimal se le resta al valor absoluto de la muestra la parte entera, se multiplica por 10 y se redondea al más cercano.

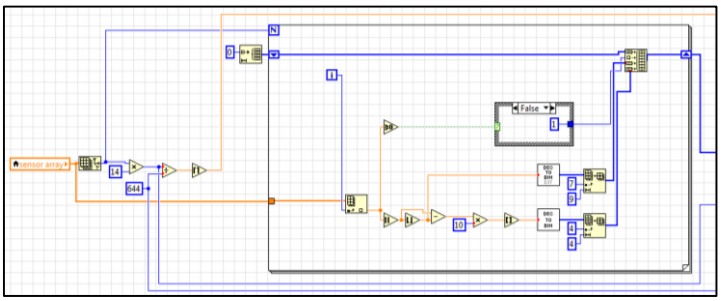


Fig. 16.14. Obtención de las muestras en formato binario

Una vez ejecutado el código anterior se procede al envío de los datos. Para realizarlo se ejecuta un bucle *for* en el que en cada iteración se comprueba el estado del indicador de interrupción de reporte del sensor. En caso de que no se

haya solicitado dicha interrupción se ejecuta el código del caso falso. El siguiente paso es concatenar el paquete de salida introduciendo primero el tiempo de adquisición de las muestras pasado a binario con la función *dec_to_bin_ext.vi*. En segundo lugar, se inserta el número de muestras que van en el paquete. En caso de que no sea la última iteración del bucle se selecciona 46 y en caso contrario 23. Se pasa a binario y se cogen los 11 bits menos significativos. Luego se introducen los cinco bits de relleno y en último lugar se insertan las muestras correspondientes en representación binaria. Para ello se van sumando el número de bits de datos del sensor en cada iteración y se toma como posición de inicio de *Array Subset* con longitud la correspondiente al valor de salida del *Case*.

Por último se crea el paquete y se manda con la función *make_packet.vi* a la que se le pasa por parámetro los valores 32 y 8 como valores de servicio y subservicio correspondientemente. El valor de *Sequence Number* es *i* yendo desde cero hasta el número de paquetes total. En última instancia se registra la operación en el *log*.

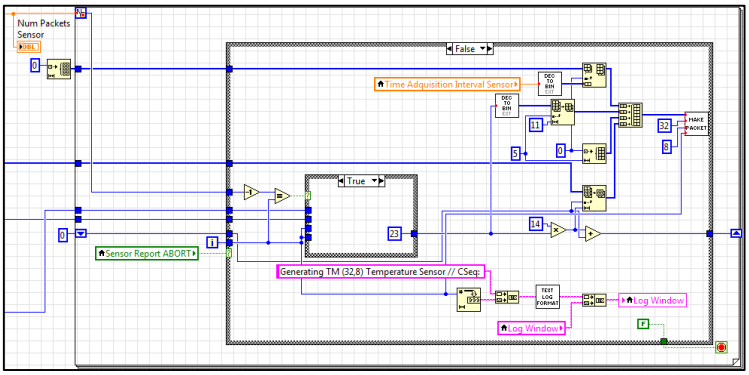


Fig. 16.15. Creación de la secuencia de paquetes de reporte del sensor

16.10 Abort Data transfer (32,9) [TC]

Con este telecomando se solicita la interrupción de la transmisión de los datos de temperatura que se estén transmitiendo en ese mismo instante de tiempo. No tiene datos de aplicación.

16.11 Transmission Aborted (32,20) [TM]

Con la telemetría Transmission Aborted (64,20) se informa de la interrupción de la transmisión. No tiene datos de aplicación.

Referente a la implementación de este subservicio primero se comprueba dicho valor y se procede a la ejecución del código de la siguiente figura en el cual

se pone a *true* el indicador de transmisión de sensor de temperatura abortada y se registra la operación.

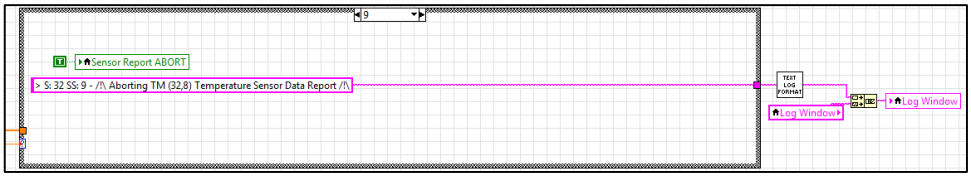


Fig. 16.16. Activación de la interrupción de la transmisión de los datos del sensor

En segundo lugar, relacionado con la sección anterior, se procede a la parada de la transmisión dentro del bucle de generación de reportes de datos de instrumentos como se muestra en la siguiente figura:

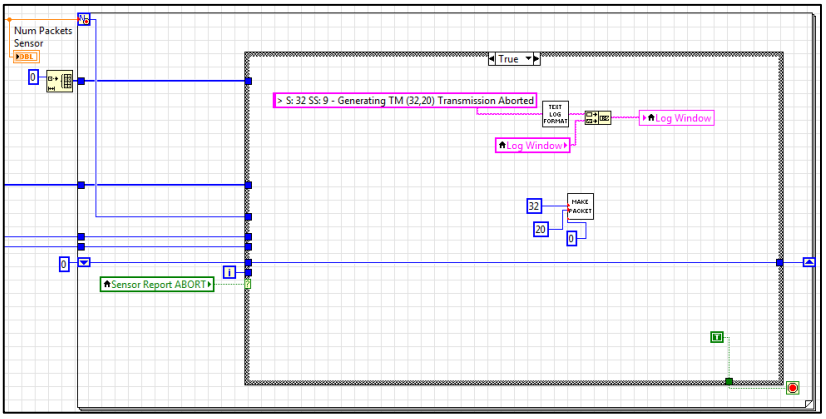


Fig. 16.17. Interrupción del reporte y generación de TM (32,20)

Se pone a *false* la ejecución, se crea la telemetría que indica que se ha abortado la transmisión de los datos de temperatura (con la función *make_packet.vi* pasándole el subservicio con valor 20) y se registra la operación. Además, se detiene el bucle *for* pasándole el valor *true* a la condición de parada.

16.12 Test instrument (32,16) [TC]

Con este telecomando se solicita el reporte del correcto funcionamiento del instrumento sensor de temperatura. No tiene datos de aplicación.

16.13 Instrument test report (32,17) [TM]

Con esta telemetría se responde al telecomando anterior, Test instrument (32,16), para la comprobación del correcto funcionamiento del instrumento. No tiene datos de aplicación.

Sensor Instrument Error (8)
‘00000000’/’11111111’

Tabla 16.4. Formato de trama TM (32,17)

El campo *Sensor Instrument Error* informa acerca del estado del instrumento sensor de temperatura del satélite. Si se simula error dicho campo está formado por todo ‘1’ si no se pone todo a ‘0’.

Para la implementar este subservicio, tras la comprobación del mismo se comprueba el estado del botón *SIM Test Sensor Instrument Failure*. En el caso de que no se encuentre pulsado (se simule error del instrumento) se ejecuta el caso false en el que se crea un *array* de un octeto con todos sus elementos puestos a 0. En caso de que se simule error se crea una *array* del mismo tamaño pero con todos los bits a 1. Luego se introducen los 8 bits en la función *make_packet.vi* junto con *Service Type* con valor 64 y *Subservice Type* con valor 17. De este modo se crea la respuesta. Luego se registra la operación en *Log Window*. Todo lo descrito se ilustra a continuación.

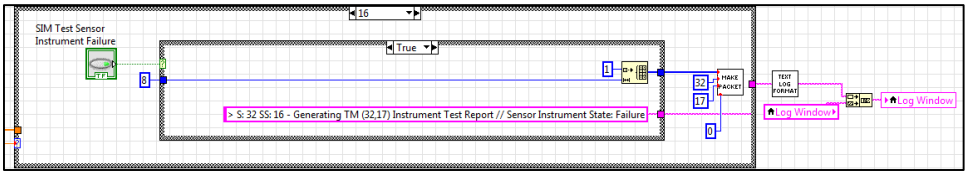


Fig. 16.18. Generación de error del sensor

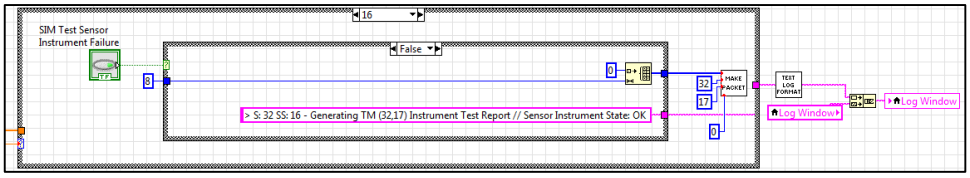


Fig. 16.19. Generación de estado correcto del sensor

17. Servicio 64: Instrumento de imágenes

Con este servicio se gestiona el instrumento de imágenes. Cada imagen tomada por dicho sensor tendrá una resolución de 200 píxeles por 200 píxeles (200x200). Se han escogido ocho imágenes relacionadas con el espacio con dichas medidas y son las que se simula que toma el instrumento de imágenes (con extensión *.jpeg*). Se muestran a continuación junto con su IID (*Image Identification*):

IID	***Imagen***	***Nombre del archivo***	***Referencia***
0		IID0.jpeg	[28]
1		IID1.jpeg	[29]
2		IID2.jpeg	[30]
3		IID3.jpeg	[31]
4		IID4.jpeg	[32]

Tabla 17.1. Tabla de imágenes del instrumento con identificador

En lo referente a la estandarización de la forma, se toma como largo de la imagen (*Long*) la dirección horizontal y como ancho de la imagen (*Width*) la vertical, tal y como se muestra a continuación:

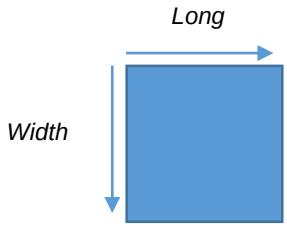


Fig. 17.1. Formato de forma de imagen

En cuanto a la implementación de este servicio, dentro del *Diagrama de Bloques* está presente en tres bucles, el de recepción, el de generación de datos de instrumentos y en el de generación de reportes de datos de instrumentos. En primer lugar se explica cómo se lleva a cabo la generación de las imágenes.

17.1 Funcionamiento del instrumento

Referente al Panel Frontal, el instrumento de imágenes se encuentra en la pestaña *Image Instrument* que tiene el siguiente aspecto:

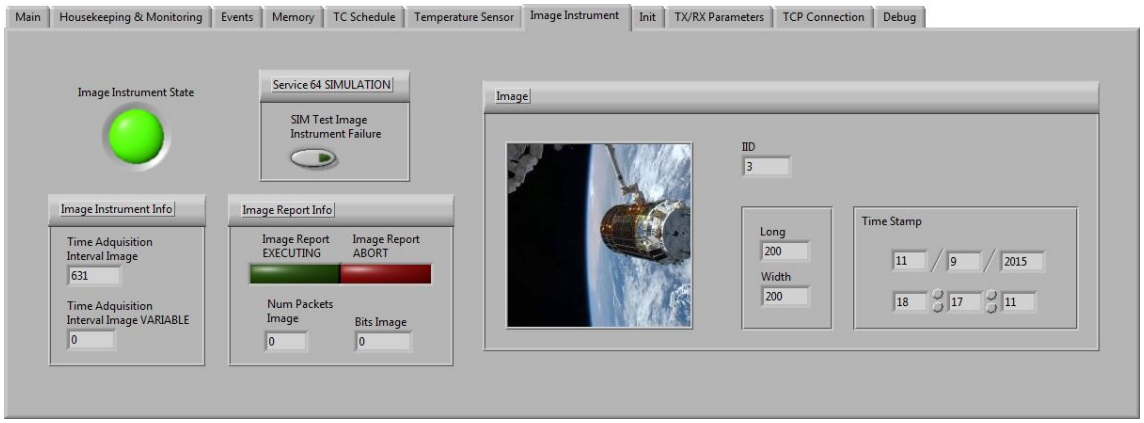


Fig. 17.2. Pestaña Image Instrument

En ella se incluyen información del instrumento (tiempo de adquisición de las imágenes y del provisional como se explica más adelante), un botón para simular fallo en el instrumento, información de la imagen (la propia imagen que es tomada en la marca temporal mostrada, su identificador de imagen –*IID*- y las dimensiones de la misma en píxeles), información del proceso de creación de paquetes de reporte (un LED como que se ilumina en verde si el reporte está en ejecución, otro que se ilumina en rojo en caso de abortar el reporte, el número de paquetes necesarios para reportar la imagen y el número de bits que ocupa la imagen en sí misma) y un LED con el estado de actividad (indica si el

instrumento está en funcionamiento o no). Tanto el tamaño total de bits que se mandan como el número de paquetes no se actualizan hasta que se empiezan a mandar los datos.

En lo referente a la implementación de generación de las imágenes, ésta se lleva a cabo dentro del bucle de generación de datos de instrumentos tal y como se describe en la sección *16.1 Funcionamiento del instrumento.*

Dentro del bloque de generación de imágenes encontramos el siguiente código:

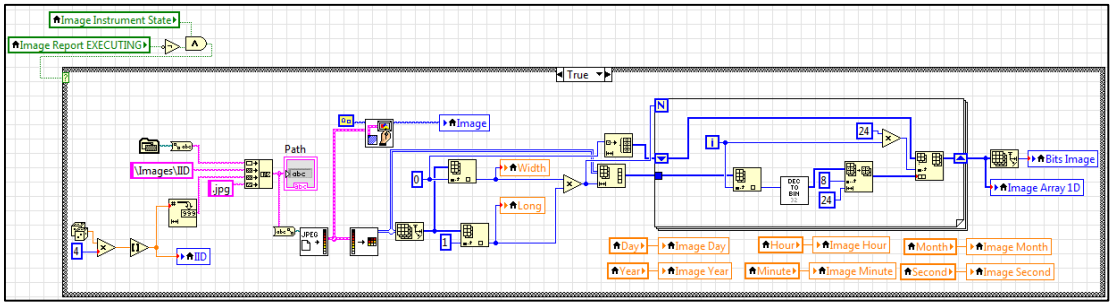


Fig. 17.3. Generación de las imágenes

Como se ilustra, hay un *Case Structure* en el que se comprueba que el estado del instrumento sea encendido (*Image Instrument State* debe valer *true*) y que no se está reportando una imagen en ese momento (*Image Report EXECUTING*).

El funcionamiento es el descrito a continuación. Primero, se registra la marca temporal (día, mes, año, hora, minuto y segundo) en que se toma la fotografía usando los datos del reloj del sistema. Por cada uno de los seis valores del reloj se crea una variable local pero que guarda la marca temporal para el instrumento de imágenes.

En segundo lugar, se obtiene un número aleatorio entre 0 y 1, que se multiplica por 4 (representa el número de imágenes disponibles) y se redondea. De este modo se obtiene el identificador de la imagen (*IID*) que se convierte a *string*. Se obtiene el directorio del proyecto (tipo *path*) y se convierte a *string* a lo que se concatena el subdirectorio donde están las imágenes y el nombre del archivo (“*/Images/IID*”) junto con el número de imagen y la extensión (“.jpg”). Una vez cosntituido el directorio definitivo en formato *string* se convierte al tipo *path*.

En tercer lugar, se introduce por parámetro el *path* obtenido en la función *Read JPEG File* que extrae datos de imagen. Con estos datos se muestra la imagen en el indicador Image del *Panel Frontal* empleando la función *Draw Flattened Pixmap.* También se le pasan a la función *Unflatten Pixmap* que

devuelve un *array* bidimensional en el que cada píxel toma un valor de 24 bits (*24-bit pixmap*). Se modifican los valores de dimensión de la imagen tomando los datos de dicho *array* de dos dimensiones con la función *Array Size.* La multiplicación del ancho y el largo arroja el número de píxeles que tiene la imagen.

Por último, lo que se hace es dejar lista la imagen para su envío. Para ello se almacenan en la variable local *Image Array 1D* el valor de cada píxel en representación binaria de veinticuatro bits. El tamaño de dicho *array* unidimensional representa el número de bits de imagen que serán enviados y se muestra en el indicador *Bits Image*. Para formar dicho *array* se ha programado un bucle *for* que recorre el *array* bidimensional, extrae el valor en la posición *i*, lo pasa a un binario de 32 bits con la función *dec_to_bin_32.vi* de cuyo resultado se cogen los 24 últimos y se introducen en un *array* unidimensional que se va concatenando con un registro de desplazamiento en el bucle.

17.2 Switch on instrument (64,1) [TC]

Con este telecomando se enciende el instrumento de imágenes. No tiene datos de aplicación.

Su implementación es muy sencilla. En primera instancia, dentro del bucle de recepción se comprueba el telecomando entrante, así como los valores de *Service Type* y *Subservice Type* que deben valer 64 y 1 respectivamente. Luego simplemente se pone el estado *true* del indicador *Image Instrument State* y se registra la operación tal y como se muestra:

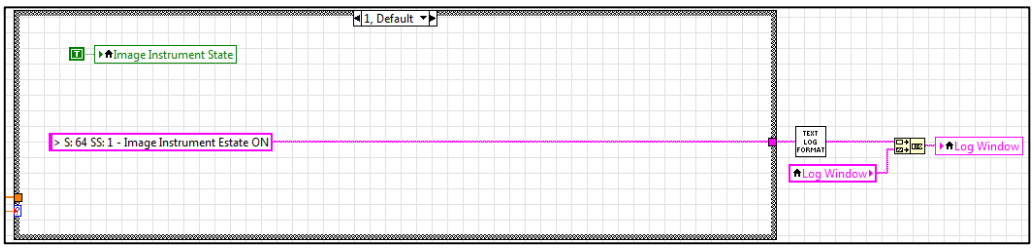


Fig. 17.4. Encendido del instrumento de imágenes

17.3 Switch off instrument (64,2) [TC]

Con este telecomando se apaga el instrumento de imágenes. No tiene datos de aplicación.

La implementación de este telecomando es parecida a la del anterior. Se ejecuta el código de *Case Structure* con valor de selector 2 (*Subservice Type*). En este caso se introduce un *false:*

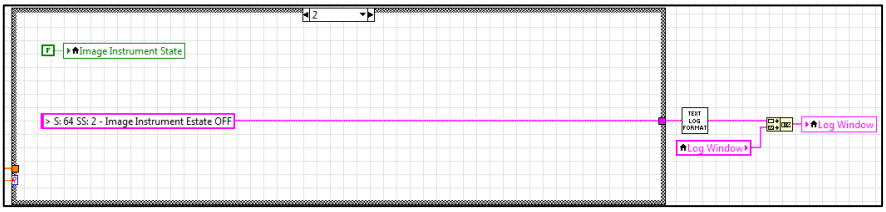


Fig. 17.5. Apagado del instrumento de imágenes

17.4 Change Time Acquisition Interval (64,3) [TC]

Con este telecomando se guarda un nuevo intervalo de adquisición de imágenes del instrumento en memoria, que no será actualizado hasta recibir el telecomando Update time acquisition interval (64,6). El formato de trama es el siguiente:

Time Acq Interval (16)
'X'

Tabla 17.2. Formato de trama de TC (64,3)

El campo *Time acq interval* está formado por dos octetos cuyo rango va desde 1 segundo hasta 65535 segundos, igual que en el instrumento sensor de temperatura. Constituye el tiempo de adquisición de las imágenes.

En lo referente a su implementación, tras la comprobación de que se trata del subservicio 3, se pasan los 16 bits de datos de aplicación a decimal con la función *bin_to_dec.vi* y se almacenan en la variable local *Time Adquisition Interval Image VARIABLE* a la espera de recibir el telecomando de actualización del periodo de adquisición. Después, se registra la operación de igual modo que en casos anteriores. El código es el mostrado en la siguiente figura.

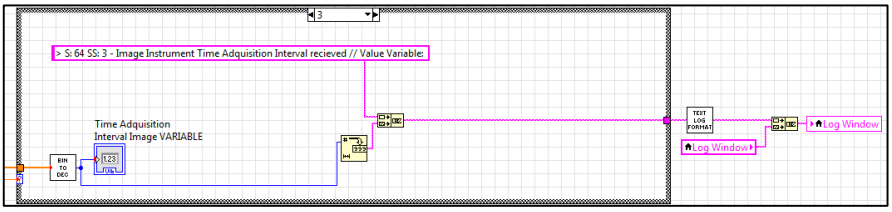


Fig. 17.6. Guardado de tiempo de adquisición temporal del instrumento de imagen

17.5 Request Time Acquisition Interval (64,4) [TC]

Con este telecomando se solicita el reporte del intervalo de adquisición de imágenes del instrumento. No tiene datos de aplicación.

17.6 Time Acquisition Interval Report (64,5) [TM]

El telecomando anterior se responde con la Time Acquisition Interval Report (64,5), incluyendo el intervalo de adquisición de imágenes del instrumento. El formato de trama es idéntico al del telecomando Change Time Acquisition Interval (64,3).

Para su implementación, tras la comprobación de que los valores de *Service* Type y *Subservice Type* valen 64 y 4 respectivamente, se coge el valor de la variable local *Time Adquisition Interval Image* y se pasa a binario con la función *dec_to_bin_ext.vi.* Esta representación binaria se introduce en la función *make_packet.vi* junto con los valores de servicio igual a 64 y subservicio igual a 5. De este modo se crea la telemetría respuesta. El valor de *Sequence Number* es cero. Después, se registra la operación.

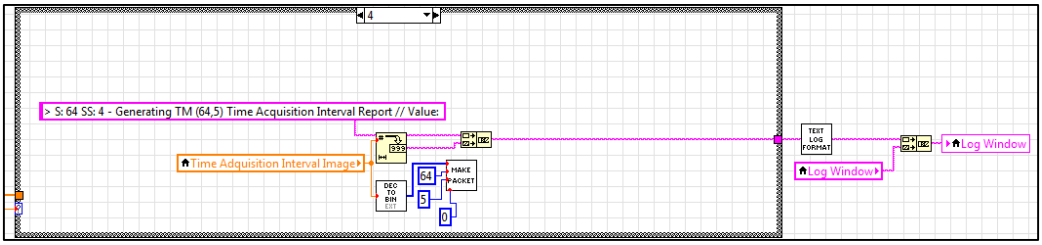


Fig. 17.7. Generación del reporte del tiempo de adquisición del instrumento de imagen

17.7 Update time acquisition interval (64,6) [TC]

Con este telecomando se actualiza el intervalo de adquisición de imágenes que previamente se había mandado con el telecomando Change time acquisition interval (64,3). No tiene datos de aplicación.

Para la implementación de esta sección se ejecuta el siguiente código:

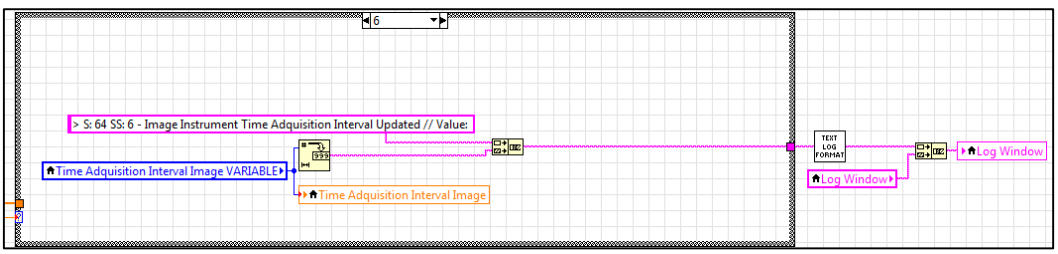


Fig. 17.8. Actualización del tiempo de adquisición del instrumento de imagen

Su funcionamiento es sencillo. Una vez comprobado el valor del subservicio simplemente se almacena el valor de la variable local *Time Adquisition Interval Image VARIABLE* en la variable local *Time Adquisition Interval Image* y se registra la operación.

17.8 Request data (64,7) [TC]

Con este telecomando se solicitan los datos tomados por el instrumento de toma de imágenes. No tiene datos de aplicación.

17.9 Data from instrument (64,8) [TM]

Con esta telemetría se responde al telecomando anterior, Request data (64,7), incluyendo el intervalo de adquisición de imágenes del instrumento y las imágenes tomadas. El formato de trama es el siguiente:

IID (16)	Time Stamp (40)	Long (11)	Width (11)	Spare (2)	Data instrument (múltiplo de octeto)
'X'	'X'	'X'	'X'	'00'	'X'

Tabla 17.3. Formato de trama de TM (64,8)

El campo *IID (Image Identification)* está formado por dos octetos y contiene un identificador de imagen para poder discernir los paquetes que corresponden a cada imagen en concreto. El campo *Time Stamp* contiene el tiempo en que fue hecha la fotografía en el sistema de planificación (referenciado al sistema de tiempo del satélite) con la estructura del sistema de tiempo del ordenador a bordo descrita en el *Servicio 9* del presente documento. Lo campos *Long* y *Width* representan las dimensiones de la imagen tal y como se ha descrito al inicio del presente servicio. Los dos bits siguientes hacen la función de relleno (*Spare*). En último lugar se mandan los datos del instrumento.

Para la ejecución de esta sección, se debe tener en cuenta que en primer lugar se realiza la recepción en el bucle de recepción y se activa la ejecución del reporte del bucle de generación de reportes de datos de instrumentos. En el bucle de recepción se ejecuta el siguiente código:

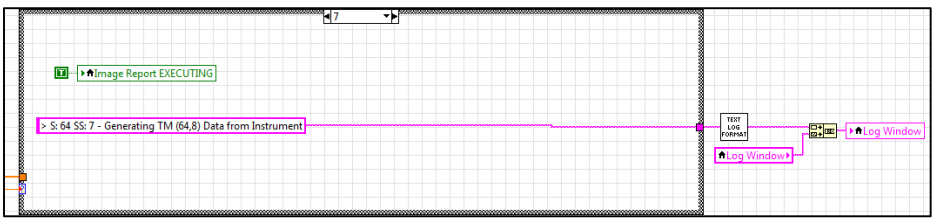


Fig. 17.9. Activación del reporte de imagen

Como se observa, simplemente se pone a *true* la variable local *Image Report EXECUTING* y se informa de que se procede a la generación del reporte.

Dentro del bucle de generación de reportes de datos de instrumentos se implementa la estructura vista en las secciones *16.8 Request data (32,7)* y *16.9 Data from instrument (32,9)*.

Una vez activado el reporte de imágenes se procede a la ejecución del caso *true,* dentro del cual hay una estructura secuencial. En primer lugar se generan todos los paquetes y se mandan, y en segundo lugar se ponen a false las variables locales *Image Report EXECUTING* e *Image Report ABORT*, tal y como se muestra a continuación:

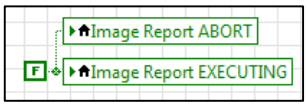


Fig. 17.10. Segunda subsecuencia del reporte de imágenes

Anteriormente se han mandado todos los paquetes salvo que haya llegado un telecomando Abort Data transfer (64,9), en cuyo caso se aborta la transmisión. La implementación de dicho proceso se puede ver en la figura ***Fig. 17.15. Creación de la secuencia de paquetes de reporte del instrumento de imágenes.***

.

En primer lugar se cogen los datos del tamaño de la imagen (variables locales *Long* y *Width*) para calcular el número de paquetes que van a transmitir (indicador *Num Packets Image*). Para ello se divide el número de píxeles total (multiplicación del ancho por el largo) entre el número de píxeles que se introducen por paquete. En esta primera versión se ha limitado el tamaño de paquete en transmisión del satélite a 776 bits por lo que el número de píxeles máximo es de 25 por paquete. Luego se redondea hacia arriba y este valor es el que se emplea como número de iteraciones del bucle.

Una vez calculado el número de paquetes, dentro del bucle *for* se realiza una comprobación por iteración para saber si se ha recibido la orden de abortar la transmisión de la imagen. En caso de que no se aborte se ejecuta el código del interior del *case false*. El proceso es el mismo de concatenación de *arrays*.

En primer lugar se introduce la IID en representación binaria para lo que se emplea la función *dec_to_bin_ext.vi*. Luego se introduce la marca de tiempo de la imagen (variables locales *Image Day, Image Month, Image Year, Image Hour, Image Minute* e *Image Second*, junto con los bits de relleno siguiendo con el formato de tiempos del *Servicio 9*). El proceso es exactamente el mismo que en la generación del reporte de tiempo del *Servicio 9*. Luego se introducen las dimensiones de la imagen (11 bits cada campo) y dos bits de relleno. En último

lugar insertan los datos de la imagen (los 25 píxeles en formato binario) y cada píxel en binario está representado por 24 bits. De este modo se van cogiendo de 600 en 600 bits de la variable local *Image Array 1D* y se van insertando.

Luego se envía el paquete con la función *make_packet.vi* con *Service Type* valiendo 64 y *Subservice Type* valiendo 8. Por cada paquete que se genera se le pasa un valor de *Sequence Count* en aumento desde 0 hasta 44. En última instancia se va registrando la operación en *Log Window*.

17.10 Abort Data transfer (64,9) [TC]

Con este telecomando se solicita abortar la transmisión de la imagen que se esté transmitiendo en ese mismo instante de tiempo. No tiene datos de aplicación.

17.11 Transmission Aborted (64,20) [TM]

Con la telemetría Transmission Aborted (64,20) se informa de la interrupción de la transmisión. No tiene datos de aplicación.

En cuanto a su implementación en primer lugar se comprueba el subservicio y se procede a la ejecución del código de la siguiente figura en el cual se pone a *true* el indicador de transmisión de imagen abortada y se registra la operación. Este proceso es el mostrado en la figura ***Fig. 17.11. Activación de la interrupción de la transmisión de los datos del instrumento de imagen.***

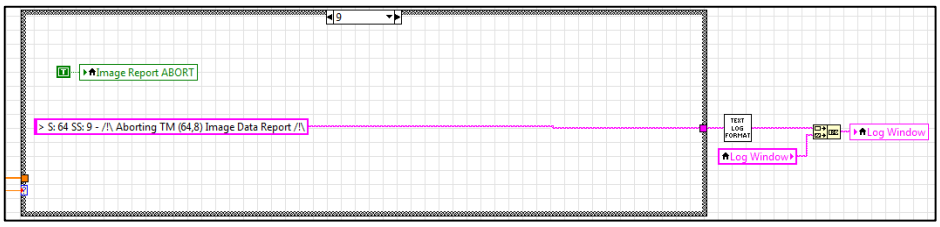


Fig. 17.11. Activación de la interrupción de la transmisión de los datos del instrumento de imagen

En segundo lugar, relacionado con la sección anterior, se procede a la parada de la transmisión dentro del bucle de generación de reportes de datos de instrumentos como se muestra a continuación:

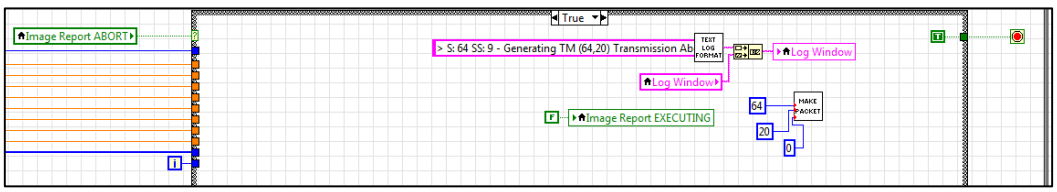


Fig. 17.12. Interrupción del reporte y generación de TM (64,20)

Se pone a *false* la ejecución, se crea la telemetría que indica que se ha abortado la transmisión de la imagen (con la función *make_packet.vi* pasándole el subservicio con valor 20) y se registra la operación. Además se para el bucle *for* pasándole el valor verdadero a la condición de parada.

17.12 Test instrument (64,16) [TC]

Con este telecomando se solicita el reporte del funcionamiento del instrumento de imágenes. No tiene datos de aplicación.

17.13 Instrument test report (64,17) [TM]

Con esta telemetría se responde al telecomando Test instrument (64,16), Test instrument (64,16), para la comprobación del correcto funcionamiento del instrumento. Tiene el siguiente formato:

Image Instrument Error (8)
'00000000'/'11111111'

Tabla 17.4. Formato de trama de TM (64,17)

El campo *Sensor Instrument Error* informa acerca del estado del instrumento de imágenes del satélite. Si se simula error dicho campo estará activo a nivel alto ('1'). En caso contrario, estará a nivel bajo ('0').

Para la implementación de este subservicio, tras la comprobación del mismo se comprueba el estado del botón *SIM Test Image Instrument Failure*. En el caso de que no se simule error del instrumento (no pulsado) se ejecuta el caso false en el que se crea un *array* de 8 bits con todos sus elementos a 0. En caso de que se simule error se crea una *array* del mismo tamaño pero con todos los bits a 1. Luego se introduce dicho octeto en la función *make_packet.vi* junto con *Service Type* con valor 64 y *Subservice Type* con valor 17. De este modo se crea la telemetría respuesta. Luego se registra la operación en *Log Window*. A continuación se muestra el código descrito.

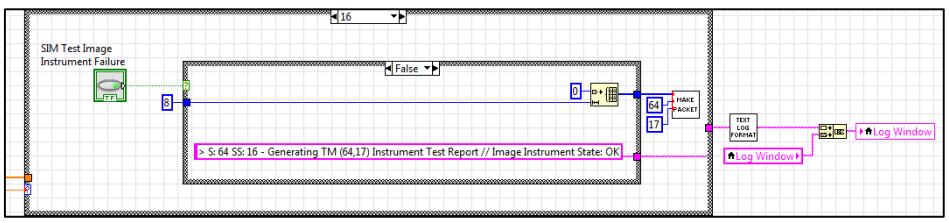


Fig. 17.13. Generación de error del instrumento de imagen

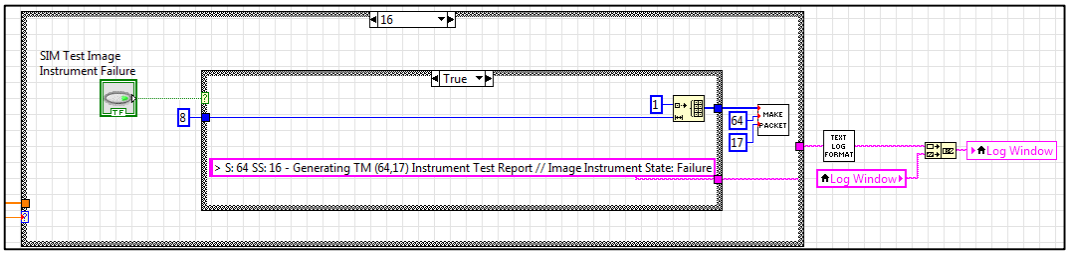


Fig. 17.14. Generación de estado correcto del instrumento de imagen

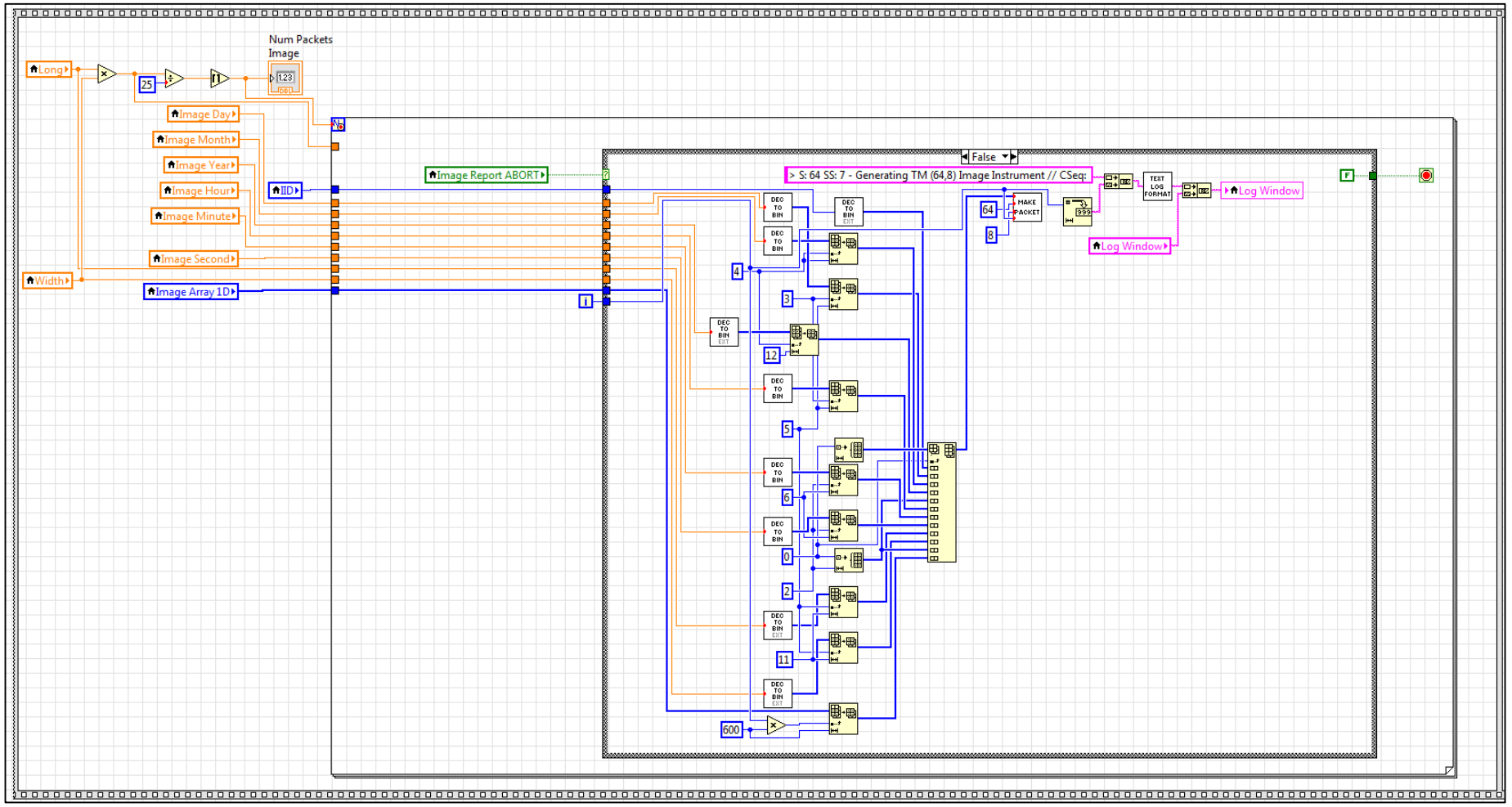


Fig. 17.15. Creación de la secuencia de paquetes de reporte del instrumento de imágenes

18. Servicio 128: Reboot

Este servicio es uno de los más básicos y a la vez más importantes del sistema porque reinicia por completo el sistema.

18.1 Reboot (128,1) [TC]

Con este telecomando se solicita el reinicio del sistema de a bordo por completo. Por lo tanto, con la recepción de este telecomando se inicializan todos los parámetros del sistema en su estado inicial de funcionamiento. No tiene datos de aplicación.

Este servicio está presente en el *módulo de recepción y procesado* y en el *módulo 2.* El sistema se reiniciará cuando o bien se reciba este telecomando o bien cuando se pulse en el panel frontal el botón *SIM Reboot*. Los elementos empleados en este servicio son los destacados en la siguiente figura:

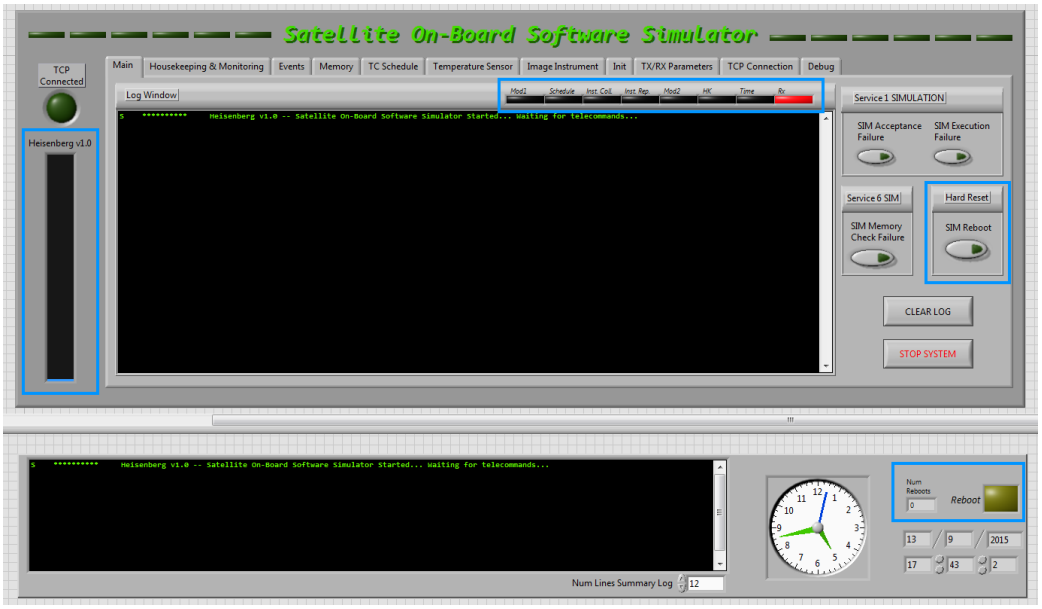


Fig. 18.1. Elementos del módulo de reinicio en el Panel Frontal

Como se observa, para cada uno de los bucles principales presentes en *main.vi* hay un indicador (LED en negro) encima de *Log Window,* que indica que el bucle se ha parado cambiando su color a rojo. Además, a la izquierda de la figura se llenará la barra de progreso en color amarillo y abajo a la derecha el indicador *Reboot* parpadeará en color amarillo. También hay un indicador numérico con el número de reinicios del sistema.

En el caso de que el reinicio se produzca por recepción de este telecomando se pone a *true* el controlador de reinicio del bucle de recepción.

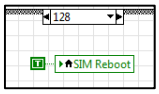


Fig. 18.2. Activación del reinicio del sistema

En cualquier caso, una vez activado el reinicio se ejecuta el código de la figura ***Fig. 18.3. Reinicio del sistema.***

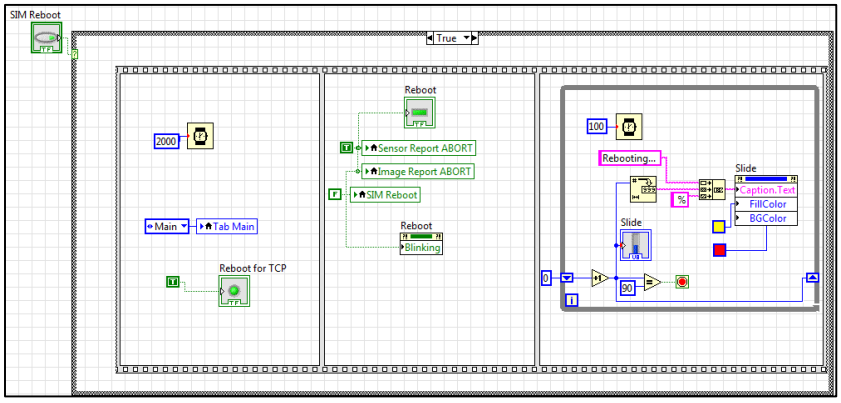


Fig. 18.3. Reinicio del sistema

Se realiza una espera de dos segundos para que en el caso de que se haya establecido conexión TCP le dé tiempo a cerrarla y se fija el panel en la pestaña *Main*. En segundo lugar se aborta la transmisión de los instrumentos en caso de que estuvieran reportando datos para no esperar a su finalización y se activa la propiedad de parpadeo del LED *Reboot*. Además, se pone a *false* el controlador de reinicio. Luego se rellena la barra (*Slide*) indicando que se está reiniciando el sistema en color amarillo hasta llegar a 90.

Una vez activada la variable *Reboot*, en todos los bucles está implementada la parada de los mismos mediante el respectivo botón de parada o la puesta a *true* de la variable de reinicio (empleando una puerta *OR*). Una muestra es la siguiente imagen:

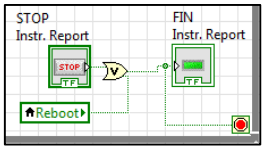


Fig. 18.4. Parada de los bucles del programa

Una vez que se han parado todos los bucles se pasa a la siguiente subsecuencia de la estructura de secuenciación principal del sistema en la que se completa la barra de progreso y se reproduce el sonido *satellite_reboot.wav* de la misma manera que el sonido de inicialización (véase *Anexo G: Inicialización*

visual y auditiva). Luego se realiza una espera de cinco segundos. Dicha implementación es la que se muestra en la siguiente imagen:

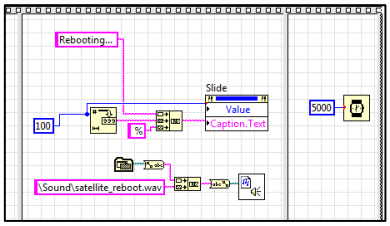


Fig. 18.5. Ejecución de las subsecuencias cuatro y cinco

Al finalizar todo el proceso se ejecuta otra iteración del bucle principal del sistema y se actualiza el número de reinicios cogiendo el valor *i* de dicho bucle. De esta manera se vuelve a la subsecuencia inicial del sistema, a la inicialización y de nuevo a la ejecución del programa.

19. Pruebas y validación

En este anexo se documentan las pruebas realizadas tanto de simulación del código sin los módulos de transmission inalámbrica como de las llevadas a cabo para el sistema al completo incluyendo la comunicación con la estación base (recepción y transimisión).

En primer lugar, para comprobar el correcto funcionamiento del sistema sin el módulo de recepción, a lo largo de la implementación se han realizado numerosas pruebas empleando indicadores y controladores para obtener los resultados esperados. Así, por ejemplo, se ha simulado la recepción de paquetes con el uso de un controlador *array* en lugar de insertar en *Packet IN* los bits recibidos y un botón que al pulsarlo ponía a *true* la variable local *Packet RX*. Con esto se simula la entrada de un paquete y se ve que la salida en el array de *make_packet.vi* (con la transmisión deshabilitada) es la que se tiene que generar. De este modo, se ha conseguido aislar lo que es la comunicación del procesado y funcionamiento general del sistema.

Una vez comprobado el correcto funcionamiento del programa, se procede a la inserción de la recepción de paquetes. Se ha ido comprobando todo el funcionamiento del sistema y modificando a medida que se realizaban las pruebas. Por ejemplo, para el caso del reporte de la imagen en un principio no se realizaban repeticiones de paquetes en la transmisión pero ante la alta tasa de pérdidas se ha ajustado dicho parámetro a cuatro para que se envíe cuatro veces un mismo paquete. Con esta técnica se ha conseguido una tasa de pérdidas de entorno al 1,71%.

El proceso para comprobar el funcionamiento del sistema al completo ha sido el de envío de telecomandos al satélite y la comprobación de la tasa de paquetes recibidos. Para la telemetría el proceso es exactamente el mismo, tanto de paquetes generados a petición de la estación base como los generados automáticamente por el sistema como ocurre con el *Servicio 3*. A continuación se muestra la tabla realizada a tal efecto:

	TC/TM	Paquetes Tx	Paquetes Rx	Porcentaje de acierto (%)	Porcentaje de fallo (%)
Servicio 1	*TM(1,1)*	315	309	98,10	1,90
	TM(1,2)	24	21	87,50	12,50
	TM(1,7)	315	305	96,83	3,17
	TM(1,8)	24	22	91,67	8,33
Servicio 3	*TC(3,1)*	23	21	91,30	8,70
	TC(3,3)	33	28	84,85	15,15
	TC(3,4)	33	29	87,88	12,12

	TC(3,5)	41	36	87,80	12,20
	TM(3,6)	36	32	88,89	11,11
	TC(3,7)	38	36	94,74	5,26
	TC(3,8)	40	38	95,00	5,00
	TM(3,9)	36	35	97,22	2,78
Servicio 5	*TM(5,1)*	60	57	95,00	5,00
	TM(5,2)	60	56	93,33	6,67
	TM(5,3)	60	57	95,00	5,00
	TM(5,4)	60	58	96,67	3,33
	TC(5,5)	50	47	94,00	6,00
	TC(5,6)	50	45	90,00	10,00
	TC(5,16)	30	28	93,33	6,67
	TC(5,17)	40	31	77,50	22,50
	TM(5,18)	31	24	77,42	22,58
	TC(5,19)	40	29	72,50	27,50
	TM(5,20)	29	22	75,86	24,14
Servicio 6	*TC(6,1)*	35	30	85,71	14,29
	TC(6,2)	30	28	93,33	6,67
	TM(6,3)	28	26	92,86	7,14
	TC(6,4)	50	47	94,00	6,00
	TM(6,5)	47	45	95,74	4,26
	TC(6,16)	35	33	94,29	5,71
Servicio 9	*TC(9,1)*	60	54	90,00	10,00
	TC(9,2)	60	56	93,33	6,67
	TM(9,3)	56	54	96,43	3,57
Servicio 11	*TC(11,1)*	60	48	80,00	20,00
	TC(11,2)	60	47	78,33	21,67
	TC(11,3)	40	37	92,50	7,50
	TC(11,4)	65	55	84,62	15,38
	TC(11,5)	30	24	80,00	20,00
	TC(11,6)	35	32	91,43	8,57
	TC(11,7)	30	25	83,33	16,67
	TM(11,8)	25	20	80,00	20,00
Servicio 12	*TC(12,1)*	40	37	92,50	7,50
	TC(12,2)	40	36	90,00	10,00
	TC(12,4)	30	28	93,33	6,67
	TC(12,7)	45	39	86,67	13,33
	TC(12,8)	45	40	88,89	11,11
	TC(12,9)	35	30	85,71	14,29
	TM(12,10)	30	28	93,33	6,67
Servicio 15	*TC(15,1)*	58	52	89,66	10,34
	TM(15,2)	52	50	96,15	3,85
Servicio 32	*TC(32,1)*	40	37	92,50	7,50

	TC(32,2)	40	37	92,50	7,50
	TC(32,3)	30	26	86,67	13,33
	TC(32,4)	28	26	92,86	7,14
	TM(32,5)	26	24	92,31	7,69
	TC(32,6)	27	26	96,30	3,70
	TC(32,7)	35	28	80,00	20,00
	TM(32,8)	1.170	1.080	92,31	7,69
	TC(32,9)	32	28	87,50	12,50
	TC(32,16)	45	44	97,78	2,22
	TM(32,17)	44	42	95,45	4,55
	TM(32,20)	28	25	89,29	10,71
Servicio 64	*TC(64,1)*	40	38	95,00	5,00
	TC(64,2)	40	36	90,00	10,00
	TC(64,3)	30	27	90,00	10,00
	TC(64,4)	30	28	93,33	6,67
	TM(64,5)	28	26	92,86	7,14
	TC(64,6)	25	23	92,00	8,00
	TC(64,7)	23	20	86,96	13,04
	TM(64,8)	32.000	31.748	99,21	0,79
	TC(64,9)	20	17	85,00	15,00
	TC(64,16)	45	43	95,56	4,44
	TM(64,17)	43	41	95,35	4,65
	TM(64,20)	25	23	92,00	8,00
Servicio 128	*TC(128,1)*	48	47	97,92	2,08
Total		**36.531**	**35.907**	**98,29**	**1,71**

Tabla 19.1. Registro de las pruebas realizadas

En la tabla anterior hay que tener en cuenta que se comprueba que al darle al botón de enviar en la estación base, llegue al satélite y que al ser el satélite el que transmite, lleguen los paquetes a la estación base, sin tener en cuenta la repetición de paquetes. Esta repetición se produce en enlace descendente con la repetición de cuatro veces por paquete, de modo que en la tabla anterior se registra como un paquete enviado, y en caso de llegar alguno de esos cuatro se apunta como un paquete recibido en la estación base.

En total para las pruebas se han enviado 36531 paquetes de los que se han recibido 35907. Por tanto la tasa media de recepción correcta de paquetes es del 98,29%. De manera complementaria la tasa de pérdida y error es del 1,71%.

Como conclusiones, se puede comprobar que los paquetes del asentimiento (verificación y ejecución), el reporte de imágenes y el reinicio son los que mejor tasa de recepción correcta tienen. En general, el envío de telemetrías funciona mejor que la transmisión de telecomandos. Esto se debe a

que el envío de telecomandos no se repite y el de telemetrías sí por lo que es difícil que se pierda un paquete de telemetría.

Además, hay que tener en cuenta que el tiempo de procesado de la recepción de datos en *LabVIEW* a través del cable *Gigabyte Ethernet* ralentiza el proceso y que los *USRP* al calentarse pueden funcionar un poco peor. En general, una vez se estabiliza el sistema se obtienen los resultados anteriormente mostrados.

20. Planificación del trabajo y presupuesto

En este anexo se describe en primer lugar el proceso seguido para la obtención de la solución final y en segundo lugar su presupuesto.

20.1. Fases del Trabajo Fin de Grado

Las actividades para la realización del *TFG* son las descritas en la siguiente tabla:

Actividad	***Nombre***	***Descripción***
A	*Análisis del problema*	En primer lugar se ha analizado el problema planteado y las posibles soluciones. Esta actividad ha durado en torno a 10 horas.
B	*Análisis del estándar* [1]	Durante un total de 40 horas se ha leído el estándar entendiendo el funcionamiento del sistema.
C	*Diseño de los servicios y los formatos de trama*	Tras la lectura del estándar, se llevó a cabo el diseño de los servicios, tanto su funcionamiento como los formatos de trama asociados a cada servicio, lo que llevó en torno a 40 horas.
D	*Estructuración del sistema principal y sus módulos*	Paralelamente a la anterior actividad, se estructuró el programa en los diferentes módulos y se definieron los objetivos de cada uno de ellos de manera teórica. El tiempo estimado de esta actividad fue de 20 horas.
E	*Implementación de los módulos de recepción y transmisión*	Una vez se resuelta la actividad anterior se procedió a la implementación de la comunicación radio. Incluyendo búsqueda, diseño de la solución de comunicación, e implementación de la transmisión y la recepción, esa actividad tuvo una duración total de 45 horas.
F	*Pruebas y conclusiones*	De forma paralela a varias actividades (E, G, H, I, J y K) se realizaron sendas pruebas para comprobar el funcionamiento de la comunicación en un total de 120 horas.
G	*Implementación de los once servicios principales*	La implementación en *LabVIEW* del programa principal tuvo una duración total de 310 horas.
H	*Mejora visual del programa*	Una vez finalizada la implementación principal se llevó a cabo una mejora estética del programa. La duración estimada fue de 12 horas.

I	*Diseño del módulo de conexión TCP*	De manera adicional se llevó a cabo el diseño de la conexión TCP en un tiempo de 7 horas.
J	*Desarrollo del servidor TCP*	Para el desarrollo en *LabVIEW* del servidor fue necesario en torno a 14 horas.
K	*Desarrollo del cliente TCP*	Paralelamente a la actividad anterior se desarrolló la aplicación Android en un período de desarrollo de 14 horas.
L	*Documentación y realización de la memoria*	En último lugar, se ha escrito la memoria del *TFG* incluyendo la búsqueda de información. El tiempo estimado ha sido de 120 horas.

Tabla 20.1. Fases de desarrollo del TFG

Actividad	***Predecesora***	***Fecha Inicio***	***Fecha Finalización***	***Tiempo (horas)***
A	-	27/11/14	5/12/14	10
B	A	5/12/14	28/1/15	40
C	B	28/1/15	20/2/15	40
D	B	7/2/15	20/2/15	20
E	C, D	21/2/15	22/4/15	45
F	C, D	21/2/15	9/7/15	120
G	C, D	6/3/15	9/6/15	310
H	G	10/6/15	13/6/15	12
I	H	14/6/15	15/6/15	7
J	I	16/6/15	24/6/15	14
K	I	16/6/15	24/6/15	14
L	E, F, J, K	10/7/15	23/9/15	120
			Total (horas)	752

Tabla 20.2. Relación de precedencia y horas de las actividades

Con los datos de la ***Tabla 20.2.*** se ha creado un *diagrama de Gantt* (con *Excel*).

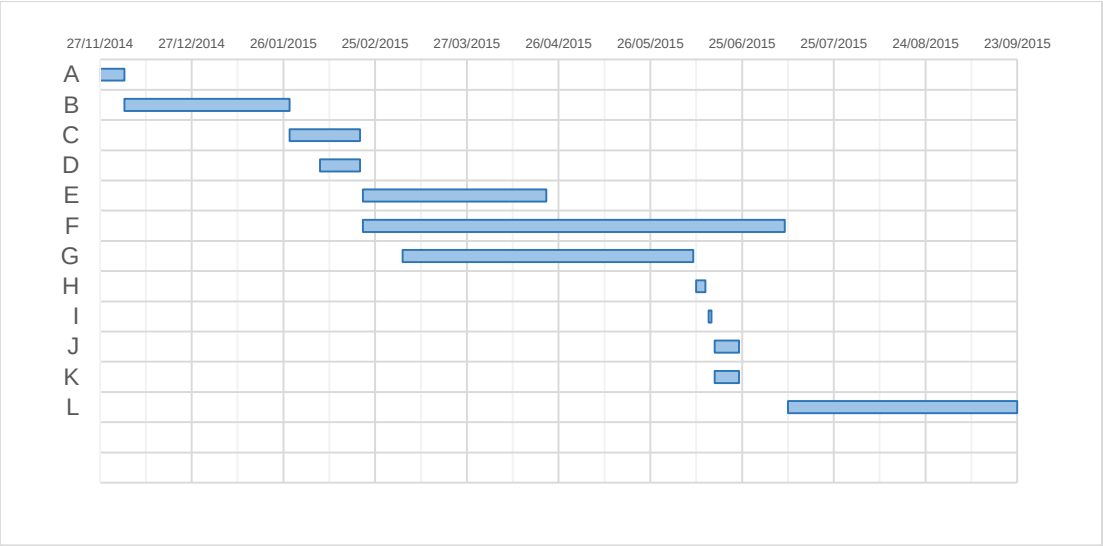


Fig. 20.1. Diagrama de Gantt

El tiempo total necesario para la realización del proyecto es de en torno a 188 días laborables a una media de 4 horas al día, con un total de 752 horas. Las actividades críticas son: *actividad C, actividad E, actividad F, actividad G* y *actividad L.*

20.2. Presupuesto de realización del proyecto

Para la realización del proyecto se han empleado tanto recursos humanos como materiales. A continuación se desglosa el gasto necesario para la obtención de la solución propuesta en el presente documento.

Persona	*Horas trabajadas*	*Coste por hora (€)*	*Coste unitario (€)*
Víctor P. Gil Jiménez	75	80	6.000
Fernando García Arias	752	40	30.080
Félix Jiménez Monzón	752	40	30.080
Daniel Rodríguez Mogollón	752	40	30.080
		Total (€)	**96.240**

Tabla 20.2. Coste de personal

Como aclaración, se ha establecido un coste por hora aproximado de *ingeniero junior* para los alumnos y de *ingeniero sénior* para el tutor. El total de los gastos de personal asciende a 96.240€.

Material	*Unidades*	*Coste por unidad (€)*	*Nº meses de utilización*	*Amortización [] (%)*	*Coste unitario (€)*
Software LabVIEW	1	50.000	3,5	26	3791,67
Equipo de sobremesa	1	800	6	26	110,5
Equipo portátil	2	700	6	26	166,4
USRP-2920	3	2.870	3,5	26	652,92
				Total (€)	**4.721,49**

Tabla 20.3. Coste de los recursos materiales

En cuanto a los recursos materiales empleados, el coste total de *LabVIEW* es de en torno a 50.000€ porque la Universidad Carlos III de Madrid tiene un contrato por todas las licencias de *LabVIEW* empleadas por la Universidad. Para el cálculo del coste final de los recursos empleados se tiene que tener en cuenta el grado de utilización. En lo referente a la valoración de los tres ordenadores y los *USRP* se tiene también en cuenta el uso que se hace. En todos los casos la amortización es del 26% tal y como se indica en la Agencia Tributaria [24]. El programa *Eclipse* y todo lo necesario para la realización de *Android* tiene coste cero. En total, el coste de estos recursos ha sido de 4721,49€.

Para el cálculo del coste total final, hay que añadir a los costes de personal y materiales (directos) los costes indirectos que suponen en torno a un 20 % de los anteriores. Además, hay que añadirle un 21 % de *Impuesto al Valor Agregado (I.V.A.).*

	Coste (€)
Personal	96.240
Recursos Materiales	4721,49
Total costes directos	**100.961,49**
Costes indirectos	20.192,3
Total (sin I.V.A)	**121.153,79**
I.V.A.	25.442,3
Total (con I.V.A)	**146.596,09**

Tabla 20.4. Coste total del proyecto

Por último, el coste total del proyecto realizado con el *I.V.A.* asciende a 146.596,09€.

21. Conclusions *[ENGLISH]*

Today's satellite communications are vital to the technological, economic and social development. The *OBDH emulator* is an essential part of the satellite design process.

The design of communication systems is very complex and for the realization of the project has had many problems due to radio communication. This shows the complexity of the existing communication systems and the need for investment in the development of satellite technology. Undoubtedly, the processing core design of a satellite as shown in this paper is hard and get a full communication process is very complicated. Currently, the major world countries and the leading companies of television and communication technology used. A functional level is essential to develop simulation software before using it on the satellite. Many global institutions such as *NASA* or the *ESA* make use of such simulations in its space operations.

Personally, I learned a visual programming language such as *LabVIEW* that allows many possibilities and that is increasingly demanding. In addition, more and more development of aesthetic and visual part of the applications and programs that *LabVIEW* is handled from the start and offers many possibilities in this regard.

At the teaching level, the ability to see how a satellite on-board works is instructive. The tools used help the learning of such systems and wireless communication. *SDR* offers the possibility to simulate scenarios as described in this work without the need to develop the hardware part. It is flexible and offers many opportunities to work with the radio emission. As shown in this paper, it is possible to perform a complete development from scratch of a complex system in this type of platform.

With this *TFG* I have learned to use standards and how the process of developing such programs. I have also learned how to frame formats such communications and the need for some fields are structured in frames. I have learned and really found the difficulties of a satellite communications system. Also the need for communication protocols to improve service with packet retransmission or repetition. It is difficult to work with wireless communication systems and are numerous problems that must be solved.

As telecommunications student I have been really see what I've learned along the degree over the years. All the problems have been studied related with wireless communications, network protocols, signal processing or satellite communications, I could handle in this project.

22. Futuras líneas de desarrollo

Como cualquier programa, siempre se pueden realizar mejoras respecto a la implementación original y añadir nuevas funcionalidades que mejoren y amplíen el servicio ofrecido.

En primer lugar se podrían realizar numerosas mejoras del funcionamiento y optimización del simulador implementado en este *TFG*. Primero, se podrían mejorar algunos servicios como:

- En el *Servicio 1* se puede optimizar la comunicación con la estación de tierra para lo que primero habría que unir los simuladores de la estación base. Además, se puede mejorar considerablemente el reporte del estado de ejecución de los telecomandos con la introducción de telemetrías intermedias que informen acerca del estado.
- En el *Servicio 3,* junto con el *Servicio 12,* se pueden simular muchos más parámetros de monitorización y ampliar considerablemente la tabla de *housekeeping*. Además, en el reporte de los datos se puede ampliar el número de datos reportados usando la secuenciación de paquetes.
- En el *Servicio 5*, al igual que en el caso anterior, se puede ampliar el número de eventos simulables.
- En el *Servicio 6* se puede ampliar el número de bloques de memoria y simular una estructura más real que se asemeje a la empleada por los satélites. También se puede ampliar el número de bloques de memoria reportados con el aumento del tamaño de la *memoria RAM* y usando secuenciación de paquetes. Además, se debería permitir la interrupción del reporte de memoria.
- En el *Servicio 11* se puede mejorar el reporte de la lista de planificación y optimizar el código para poder ampliarla
- En el *Servicio 15* se pueden añadir más telecomandos y telemetrías para obtener el reporte de datos acerca de la conexión en el satélite y del estado del enlace.
- En el *Servicio 32* se plantea el aumento del tamaño de las muestras tomadas y añadir marcas temporales para poder situar las muestras en el tiempo.
- En el *Servicio 64* y en general en el instrumento de imágenes se debe ampliar el tamaño y de las imágenes para que se pueda tener una versión más realista del funcionamiento de este tipo de elementos.
- En el *Servicio 128* se debería reportar que el reinicio del sistema ha sido completado con éxito.

De manera general, en la transmisión y en la recepción debería haber una comunicación con la estación base para permitir solictar paquetes perdidos y realizar la secuenciación de paquetes de manera automática para cualquier servicio para lo que habría que redefinir algunos formatos de trama.

A un nivel más físico, se puede mejorar la comunicación entre la estación base y el satélite con el empleo de una modulación más densa y una mejora en la sincronización. Todo ello requiere del empleo de ordenadores más potentes que los empleados para la realización de este *TFG* y que soporten *Gigabyte Ethernet*.

También se podrían ampliar el número de servicios simulados que ofrece el programa para obtener una solución más realista y completa del funcionamiento del manejo de datos de a bordo de un satélite. Por ejemplo, se podría implementar un servicio de transferencia de archivos tanto en el enlace ascendente como en el descendente. Además, a nivel de implementación es recomendable introducir unos *buffers* de entrada y salida.

Como este *TFG* no deja de ser un simulador, además muy enfocado al ámbito didáctico, se puede mejorar la estética del programa dentro de las limitaciones que tiene *LabVIEW*. Para lograrlo, se podrían añadir imágenes y botones personalizados, más indicadores a modo de *LED* y sonidos, ventanas emergentes…

Otra posibilidad sería la de introducir más instrumentos que pudieran ser seleccionados al inicio de la ejecución de manera que se puedan simular más tipos de satélites con diversas aplicaciones (meteorología, comunicaciones móviles, sistemas de posicionamiento, exploración espacial…).

En cuanto a la posibilidad de conectar mediante TCP con una aplicación móvil en *Android*, se puede mejorar notablemente. En este proyecto la versión realizada está aún en fase de test y, la comunicación y el manejo de errores no es del todo completo. Se podría añadir a la aplicación el mismo panel que el mostrado en el programa principal para que se pudieran conectar varios dispositivos móviles usando la aplicación para que de esta manera todos los móviles conectados pudieran ver la pantalla e interactuar con el programa. A nivel educativo sería una opción muy interesante.

(página en blanco)

Glosario

Sigla/Acónimo	*Definición*
OBDH	On Board Data Handling
ESA	European Space Agency
NASA	National Aeronautics and Space Administration
SDR	Software Defined Radio
CNAF	Cuadro Nacional de Atribución de Frecuencias
TC	Telecomando (Telecommand)
TM	Telemetría (Telemetry)
ECSS	European Cooperation for Space Standardization
ITU	International Telecommunication Union
COIT	Colegio Oficial De Ingenieros De Telecomunicación
GPS	Global Positioning System
EGSE	Universal Software Radio Peripheral
RAM	Random Access Memory
ROM	Read Only Memory
BPSK	Binary Phase Shift Keying
VI	Virtual Instrument
SID	Structure Identification
RID	Report ID
CCSDS	The Consultative Committee for Space Data Systems
ACK	Acknowledgement
OBT	On-Board Time
TFG	Trabajo Fin de Grado
IGY	International Geophysical Year
US	United States
USSR	Union of Soviet Socialist Republics
LEO	Low Earth-Orbit
GEO	Geostationary Earth Orbit
FM	Frequency Modulation
GSM	Global System for Mobile communications
ISM	Industrial, Scientific and Medica
TDT	Televisión Digital Terrestre
SETSI	Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información
CEPT	European Conference of Post and Telecommunications
UHF	Ultra High Frequency
TCP	Transmission Control Protocol
QoS	Quality of Service

CNIT	Consozio Nazionale Interuniversitario per le Telecomunicazioni
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
LAN	Local Area Network
FPGA	Field Programmable Gate Array
MAC	Media Access Control
APID	Application Process ID
PEC	Packet Error Control
RG	Report Generation
SIM	Simulate
LED	Light-Emitting Diode
Abs	Absolute
IID	Image ID
JPEG	Joint Photographic Experts Group
Acq	Adquisition
IVA	Impuesto al Valor Agregado

Bibliografía y referencias

[1] European Space Agency (ESA), “ECSS-E-70-41A - Ground systems and operations – Telemetry and telecommand packet utilization”, 30 January 2003.

[2] B.G. Evans, “Satellite Communication Systems”, 3rd Edition, IET, 1999. [E-book] Available: Google Books.

[3] Dharma Raj Cheruku, “Satellite Communication”, I. K. International Pvt Ltd, 2010. [E-book] Available: Google Books.

[4] S. Tirró, “Satellite Communication Systems Design”, Springer Science & Business Media, 6 dic. 2012. [E-book] Available: Google Books.

[5] Rosado Rodríguez, Carlos, “Comunicación por satélite: principios, tecnología y sistemas“, 2ª ed., Mexico : Limusa-Wiley, 2008.

[6] Francisco Sacristán Romero, “Satélites de comunicación”, Revista Latinoamericana de Comunicación Chasqui, ISSN-e 1390-1079, Nº. 91, 2005, págs. 64-71. [Online]. Available: http://dialnet.unirioja.es/

[7] NASA website, “Sputnik and The Dawn of the Space Age”, October 10, 2007. Available: http://history.nasa.gov/sputnik/

[8] Telesat, “Brief History of Satellite Communications”. Available: https://www.telesat.com/about-us/why-satellite/brief-history

[9] History.com, “The Space Race”, A+E Networks, 2010. Available: http://www.history.com/topics/space-race

[10] National Instruments website. http://www.ni.com/

[11] National Instruments website, NI USRP. http://sine.ni.com/nips/cds/view/p/lang/en/nid/212995

[12] Schor, D.; Kinsner, W.; Thoren, A., "Satellite ground station emulator: An architecture and implementation proposal," in Electrical and Computer Engineering, 2009. CCECE '09. Canadian Conference on , vol., no., pp.868-873, 3-6 May 2009. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5090253&isnumber=5090078

[13] Marchese, Mario; Perrando, M., "A packet-switching satellite emulator: a proposal about architecture and implementation," in Communications, 2002. ICC 2002. IEEE International Conference on , vol.5, no., pp.3033-3037 vol.5, 2002. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=997396&isnumber=21518

[14] Pateros, Charles N.; Cartwright, Jeffrey R., "A Microprocessor based Communications Satellite Emulator", in Military Communications Conference - Crisis Communications: The Promise and Reality, 1987. MILCOM 1987. IEEE, vol.1, no., pp. 0156-0160, 19-22 Oct. 1987. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4795175&isnumber=4795134

[15] Albertengo, G.; Petroianni, S., "On the emulation of geostationary Earth orbit satellite systems," in Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on , vol., no., pp.91-96, 2002. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1021663&isnumber=21983

[16] Zou Yaopu; Jiang Jiayou; Han Changpei, "A design of on-board dual-channel data handling method based on two FPGAs," in SpaceWire Conference (SpaceWire), 2014 International, vol., no., pp.1-4, 22-26 Sept. 2014. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6936260&isnumber=6936203

[17] Wu Jinjie; Hu Min; Gao Yudong, "A new on-board data handling system for spacecraft flight control simulator," in Cybernetics and Intelligent Systems (CIS), 2011 IEEE 5th International Conference on , vol., no., pp.19-23, 17-19 Sept. 2011. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6070295&isnumber=6070290

[18] Sharma, S.; Ravichandran, P.N.; Sunil; Lakshminarsimhan, P.; Seshaiah, R., "Wireless Telecommand and Telemetry System for Satellite," in Recent Advances in Space Technologies, 2007. RAST '07. 3rd International Conference on , vol., no., pp.551-555, 14-16 June 2007. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4284054&isnumber=4283956

[19] Ministerio de Trabajo y Asuntos Sociales de España. “NTP 234: Exposición a radiofrecuencias y microondas (I). Evaluación”.

[20] Colegio Oficial De Ingenieros De Telecomunicación, “Emisiones Radioeléctricas: Normativa, Técnicas de Medida y Protocolos de Certificación”, Cuaderno 01/2008. Available: https://www.coit.es/descargar.php?idfichero=2469

[21] API de Android. http://developer.android.com/guide/index.html

[22] National Instruments Forum. http://www.ni.com/community/

[23] LabVIEW tutoriales. http://www.ni.com/labview/

[24] Agencia Tributaria, “Estimación Directa Simplificada”. Available: http://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresarios_individuales_y_profesionales/Rendimientos_de_actividad

des_economicas_en_el_IRPF/Regimenes_para_determinar_el_rendimiento_de_las_actividades_economicas/Estimacion_Directa_Simplificada.shtml

[25] Sam Shearman, “Packet-based Digital Link”, NI Community (forum). Available: https://decibel.ni.com/content/docs/DOC-18801

[26] Markus Dillinger, Kambiz Madani, Nancy Alonistioti, “Software Defined Radio: Architectures, Systems and Functions”. Page xxxiii (Wiley & Sons, 2003, ISBN 0-470-85164-3).

[27] Fraser Cain, “How Cold is Space?”, July 2, 2013.Available: http://www.universetoday.com/77070/how-cold-is-space/

[28] http://www.listnerd.com/list/best-nebula-off-all-time

[29] http://www.wired.com/tag/space-tourism/

[30] https://es.pinterest.com/gtasa14/iphone-stuff/

[31] http://www.japantimes.co.jp/tag/international-space-station/

[32] http://waynefarleyaviation.com/2015/07/india-launches-satellite-based-navigation-system-gagan/

[33] Mohamed Ibrahim, “Satellite Clip Art”, 7 de Abril de 2009 Available: http://www.clker.com/clipart-29149.html

(página en blanco)

Anexos

Índice

Índice de figuras

Índice de tablas

Anexo A: Entorno de *LabVIEW*

A.1. *Panel Frontal*

Referente al *Panel Frontal*, si se hace *click derecho* en el *Panel Frontal* aparece el menú para insertar los diferentes elementos (*Controls*): controles numéricos (*Num Ctrls*), botones (*Buttons*), controles de texto (*Text Ctrls*), controles de usuario (*User Ctrls*), indicadores numéricos (*Num Inds*), *LEDs*, indicadores de texto (*Text Inds*), indicadores gráficos (*Graph Indicators)…* También se han empleado elementos como *Array* en el que se inserta un controlador o un indicador para crear una *array* de ese tipo (además de poder elegir el número de dimensiones del mismo). Otro elemento empleado para el instrumento de imágenes ha sido una imagen en dos dimensiones (*Image 2D*) para mostrar las fotografías tomadas por el instrumento.

Además han sido empleados elementos adicionales para la recepción y la transmisión de datos como son *MT Constellation Graph* (perteneciente al grupo de *RF Communications*) para mostrar la constelación. También se han usado elementos como *Tab Control* para ordenar el panel principal, *Real Matrix* para el servicio de memoria, *Cluster* para agrupar elementos o *Menu Ring* para seleccionar un elemento de los que se despliegan en el selector. El menú de elementos es el siguiente:

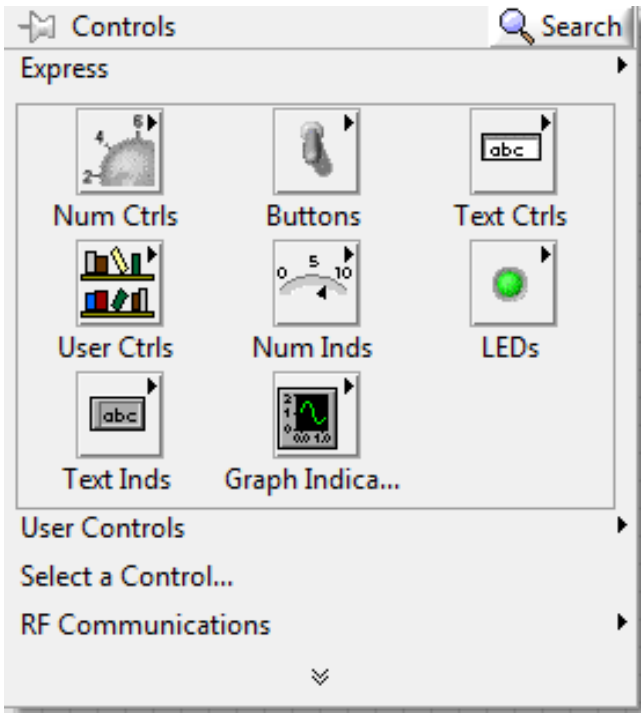


Fig. A.1. Menú de selección del Panel Frontal

A.2. *Diagrama de Bloques*

En lo referente al *Diagrama de Bloques*, al hacer *click* sobre la pantalla con el botón derecho se muestra el siguiente menú de selección de elementos:

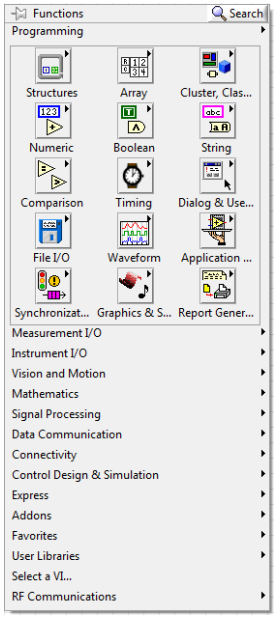
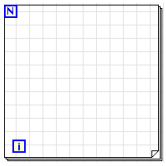


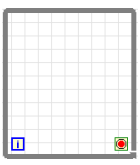
Fig. A.2. Menú de selección del Diagrama de bloques

Como puede observarse existen diferentes elementos agrupados por categorías. A continuación se describen los elementos empleados para la implementación del simulador. Durante el resto del anexo se describen las más empleadas.

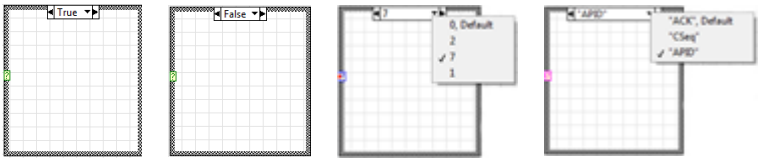
- *Structures*. A continuación se muestran las estructuras más importantes:
  - *For Loop.* Este bucle se ejecuta desde 0 hasta N-1.



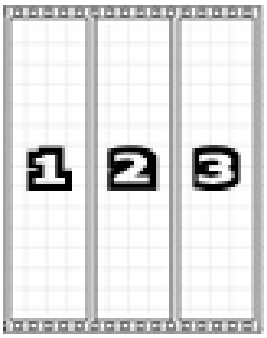
  - *While Loop*. Se ejecuta hasta que se cumple la condición de parada.



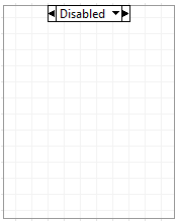
- *Case Structure.* Es una estructura muy empleada y similar a los *if* y *switch* de otros lenguajes de programación. Dependiendo del tipo de dato que se pase por el selector se modifican los casos de la estructura, es decir, si por ejemplo se le pasa un selector tipo booleano, en la estructura se programa el caso de que la condición sea *true* y el caso de que la condición sea *false.* En otro caso que se le pase al selector un dato tipo numérico en la estructura se insertan los casos numéricos y para cada uno de ellos se programa su función.



- *Flat Sequence Structure.* Se emplea para asegurar que no se ejecuta un bloque de código hasta que no haya terminado de ejecutar el bloque anterior.



- *Diagram Disable Structure.* Sirve para depurar el programa a modo de comentar bloques de código.



- *Local Variable.* Sirve para comunicar datos entre las diferentes estructuras de un mismo archivo. Los controles e indicadores son empleados de esta forma. Pueden ser escritas o leídas.

- *Global Variable.* Se usan igual que las variables globales con la salvedad de que pueden comunicar datos entre diferentes archivos de un mismo proyecto. Para usarlas es necesario crear un *Virtual Instrument* especial que sólo cuenta con *Panel Frontal*.

- *Array*. Es uno de los elementos base y tiene asociadas numerosas funciones tales como:
  - *Initialize Array.* Inicializa un *array* con elementos del tipo que se le pasa y con una dimensión determinada pasada también por parámetro. En caso de estar vacío se crea un *array* dinámico.

  - *Array Size.* Devuelve las dimensiones del *array*.

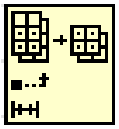
  - *Index Array.* Devuelve un elemento o *subarray* en la posición indicada.

  - *Replace Array Subset.* Modifica los elementos en la posición solicitada.

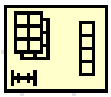
  - *Insert Into Array.* Inserta elementos en la posición pasada por parámetro.

  - *Build Array.* Concatena tantos *arrays* como se le pasen por parámetro.

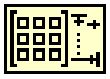
- *Array Subset.* Devuelve un sector del *array* desde una cierta posición y con la longitud indicada.



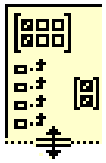
- *Reshape Array.* Modifica las dimensiones del *array*.



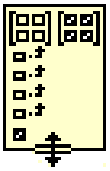
- *Matrix*. Este módulo será empleado en el servicio de memoria. A continuación, se muestran las funciones empleadas:
  - *Matrix Size.* Devuelve las dimensiones de la matriz (filas y columnas).



  - *Get Submatrix.* Esta función obtiene una *submatriz* de la matriz dada por parámetro comenzando en una fila y columna dadas y finalizando por otras pasadas también por parámetro.



  - *Set Submatrix.* Realiza la misma función que el elemento anterior pero sustituyendo los elementos de la *submatriz*.

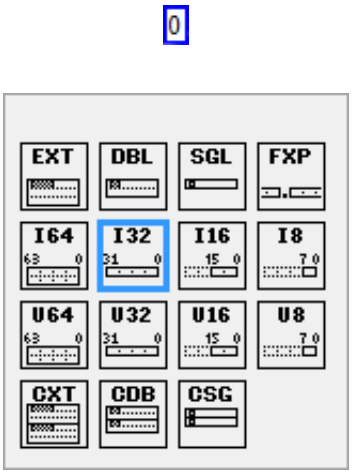


  - *Matrix To Array.* Convierte una matriz en un *array* de dos dimensiones.

- *Numeric*. Con este módulo se pueden realizar numerosas operaciones y funciones con valores de tipo numérico. Son mostradas a continuación:
  - *Operaciones aritméticas.* Se pasan dos parámetros los cuales se suman, restan, multiplican o dividen. Además hay dos funciones en particular para sumar y restar una unidad.

  - *Funciones de redondeo y truncamiento.* Se puede obtener el valor absoluto de un número, redondear al más cercano, truncar hacia el más próximo superior y al más próximo inferior.

  - *Constante numérica.* Todos los datos de tipo numérico tienen una representación en la que se controla la ocupación en memoria (en bits) así como su signo. Los iconos con una *I* son con signo y los que tienen una *V* son sin signo. La representación *DBL* es de doble precisión. A continuación se muestra dicha constante numérica y todas las posibles representaciones:



- *Boolean*. Los elementos con carácter booleano son los siguientes:
  - *Constantes.* Hay una constante *true* y otra *false*.

  - *Operaciones booleanas.* Se emplean las funciones *AND, OR* y *NOT*.

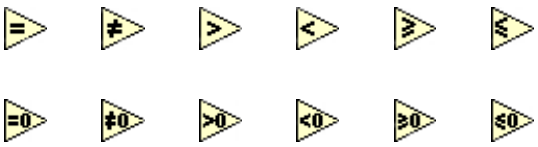
- *String*. A continuación se muestran todos los elementos de tipo texto:
  - *Constantes.* Se pueden establecer constantes textuales. Hay algunas predefinidas como son texto vacío o el cambio de línea.

  - *Concatenate Strings.* Esta función permite concatenar varios elementos de tipo texto.

  - *Number To Decimal String*. Permite convertir un número a texto.

  - *File To Path* y *Path To File*. Realizan la conversión de directorio a *string* y viceversa.

- *Comparasion*. Con este módulo se pueden realizar comparaciones y comprobaciones de distintos tipos de datos.
  - *Elementos de tipo numérico.* Se comparan dos números y devuelve un booleano para las comparaciones de igual, distinto, menos, mayor, menor o igual, y, mayor o igual. Como es frecuente comparar con el cero, también están predefinidas esas mismas funciones comparando con 0.



  - *Empty Array ?*. Comprueba si un *array* no contiene ningún elemento.

  - *Quotient & Remainder.* Realiza la división entre dos números y devuelve el resto y el cociente de dicha operación.

- *Time*. Permite controlar el tiempo de ejecución, esperas y obtener información del reloj del sistema.
    - *Wait (ms).* Ejecuta el código donde se encuentre cada cierto tiempo (medido en milisegundos).

    - *Get Date/Time In Seconds.* Devuelve una marca de tiempo del Sistema *LabVIEW* calculada a partir del número de segundos desde las *12:00 a.m. del Viernes, 1 de Enero de 1904 (Universal Time [01-01-1904 00:00:00])*.

    - *Seconds to Date/Time.* Convierte el valor devuelto por la función anterior en un *cluster* de valores numéricos.

- *Graphics*. Con este módulo se manejan los datos de imagen.
    - *Read JPEG File.* Se le pasa por parámetro el directorio de la imagen y devuelve todos los datos referentes a la misma (*cluster Image Data*).

    - *Unflatten Pixmap*. Convierte el *cluster* de la función anterior en un *array* de dos dimensiones.

    - *Draw Flattened Pixmap*. Dibuja una imagen a partir de un mapa de bits.

Anexo B: Lista de telecomandos y telemetrías

A continuación se muestra una tabla con los 75 telecomandos y telemetrías implementados así como el servicio al que pertenecen, el tipo de servicio y el subservicio.

Tipo de Paquete (TC/TM)	Tipo de Servicio (Service Type)	Tipo de Subservicio (Subservice Type)	Descripción
Servicio 1: Verificación TC			
TM	1	1	TC acceptance success report
TM	1	2	TC acceptance failure report
TM	1	7	TC execution success report
TM	1	8	TC execution failure report
Servicio 3: Housekeeping y diagnóstico de datos			
TC	3	1	Clear Housekeeping parameter report definitions
TC	3	3	Enable Housekeeping report generation
TC	3	4	Disable Housekeeping report generation
TC	3	5	Request Housekeeping parameter report generation
TM	3	6	Housekeeping parameter report
TC	3	7	Update Housekeeping report generation period
TC	3	8	Define Housekeeping report interval
TM	3	9	Housekeeping report interval updated
Servicio 5: Eventos			
TM	5	1	Normal progress report
TM	5	2	Error/anomaly report – low severity - warning
TM	5	3	Error/anomaly report – medium severity – ground action
TM	5	4	Error/anomaly report – high severity – on-board action
TC	5	5	Enable event report generation
TC	5	6	Disable event report generation
TC	5	16	Clear event log
TC	5	17	Report enabled event packets
TM	5	18	Enabled events packets report
TC	5	19	Report disables event packets
TM	5	20	Disables events packets report

Servicio 6: Gestión de memoria			
TC	6	1	Load data into memory
TC	6	2	Dump memory area
TM	6	3	Memory dump
TC	6	4	Check memory
TM	6	5	Memory Check Report
TC	6	16	Preload ROM data
Servicio 9: Gestión del tiempo			
TC	9	1	Set OBT
TC	9	2	Time request
TM	9	3	Time management
Servicio 11: Gestión del planificador de TC			
TC	11	1	Enable TC schedule
TC	11	2	Disable TC schedule
TC	11	3	Reset TC schedule
TC	11	4	Insert TC in command schedule
TC	11	5	Delete TC in command schedule
TC	11	6	Time shift to selected TC
TC	11	7	Report subset of command schedule
TM	11	8	Detailed schedule report
TC	11	16	Execution TC schedule
Servicio 12: Monitorización de a bordo			
TC	12	1	Enable monitoring of parameters
TC	12	2	Disable monitoring of parameters
TC	12	4	Clear monitoring list
TC	12	7	Add parameter to monitoring list
TC	12	8	Delete parameter to monitoring list
TC	12	9	Report current monitoring list
TM	12	10	Current monitoring list report
Servicio 15: Test de conexión			
TC	15	1	Request connection test
TM	15	2	Connection test report
Servicio 32: Instrumento sensor de temperatura (4 sensores)			
TC	32	1	Switch on instrument
TC	32	2	Switch off instrument
TC	32	3	Change time acquisition interval
TC	32	4	Request time acquisition interval
TM	32	5	Time Acquisition Interval Report
TC	32	6	Update time acquisition interval
TC	32	7	Request data
TM	32	8	Data from instrument
TC	32	9	Abort Data Transfer

TC	32	16	Test instrument
TM	32	17	Instrument test report
TM	32	20	Transmission Aborted
Servicio 64: Instrumento imágenes			
TC	64	1	Switch on instrument
TC	64	2	Switch off instrument
TC	64	3	Change time acquisition interval
TC	64	4	Request time acquisition interval
TM	64	5	Time acquisition interval report
TC	64	6	Update time acquisition interval
TC	64	7	Request data
TM	64	8	Data from instrument
TC	64	9	Abort Data Transfer
TC	64	16	Instrument test report
TM	64	17	Switch on instrument
TM	64	20	Transmission Aborted
Servicio 128: Reboot			
TC	128	1	Reboot

Tabla B.1. Lista de telecomandos y telemetría

Anexo C: Funciones de conversión

C.1. Función *bin_to_dec.vi*

Esta función pasa de un *array* de datos binario a un número entero. Tiene como entrada un *array* en representación binaria (*Binary*) y como salida un dato de tipo numérico (*Decimal*).

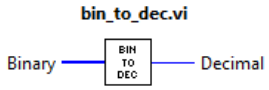


Fig. C.1.1. Función bin_to_dec.vi

En lo referente a su implementación, se pone a cero la salida y se realiza un *for* con un número de iteraciones el tamaño del *array*. En cada iteración se coge el bit correspondiente con la función *Index Array* y luego se introduce en la función *Scale By Power Of 2.* Luego se van sumando los valores obtenidos en cada iteración. El código es el mostrado en la siguiente figura:

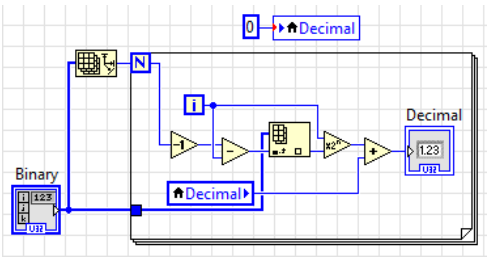


Fig. C.1.2. Transformación de binario a decimal

La representación del número de salida es *Unsigned Long (V32)*.

C.2. Función *dec_to_bin.vi*

Esta función pasa de un número entero a un *array* binario.

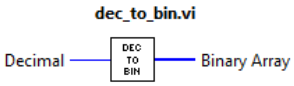


Fig. C.2.1. Función dec_to_bin.vi

En cuanto a su implementación, en primer lugar se pasa el número de entrada a un *array* booleano con la función *Number To Boolean Array*. Luego se pasan todos los valores de ese *array* a dato de tipo numérico (si es *true* se pasa a 1 y si es *false* se pasa a 0) con la función *Boolean To (0,1).* Por último, se le da la vuelta al *array* con la función *Reverse 1D Array* para poner los bits menos

significativos a la derecha siguiendo la representación binaria del estándar. La representación del número de salida es *Byte Long (V8)*.

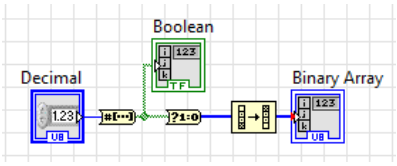


Fig. C.2.2. Transformación de decimal a binario con extensión 8 bits

C.3. Función *dec_to_bin_ext.vi*

El proceso es igual que el de la función anterior salvo por la representación del número de salida que es *Unsigned Word (V16)*.

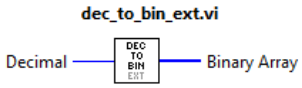


Fig. C.3.1. Función dec_to_bin_ext.vi

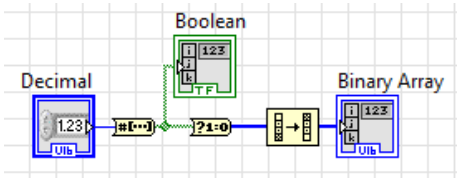


Fig. C.3.2. Transformación de decimal a binario con extensión 16 bits

C.4. Función *dec_to_bin_32.vi*

El proceso es igual que el de la función anterior salvo por la representación del número de salida que es *Long Word (V32)*.

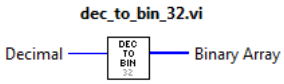


Fig. C.4.1. Función dec_to_bin_32.vi

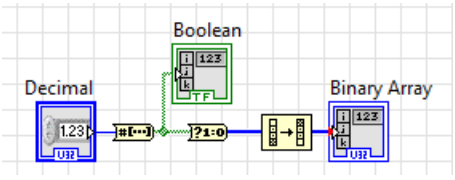


Fig. C.4.2. Transformación de decimal a binario con extensión 32 bits

C.5. Función *hk_sample_to_binary.vi*

Esta función pasa de un *array* de números en representación decimal con parte fraccional a un *array* binario con la representación descrita en la siguiente tabla:

Decimal Sample Part (10)	Fractional Sample Part (4)
'X'	'X'

Tabla C.1. Formato de muestra de datos de housekeeping SID 1

Para lleva a cabo dicha transformación se toma como entrada el *array* decimal y se ejecuta un bucle *for* desde cero hasta el tamaño total del *array*. Dentro del bucle se van extrayendo los datos del *array* con la función *Index Array* y se van concatenando el *array* en representación binaria, la parte entera y la parte decimal. Para la parte entera se pone la muestra en valor absoluto y se redondea hacia abajo. Luego se pasa a binario con la función *dec_to_bin_ext.vi* y se cogen los 9 bits menos significativos. Para obtener la parte decimal se le resta al valor absoluto de la muestra la parte entera, se multiplica por 10 y se redondea al más cercano. Todo el proceso descrito es el que se muestra en la siguiente figura:

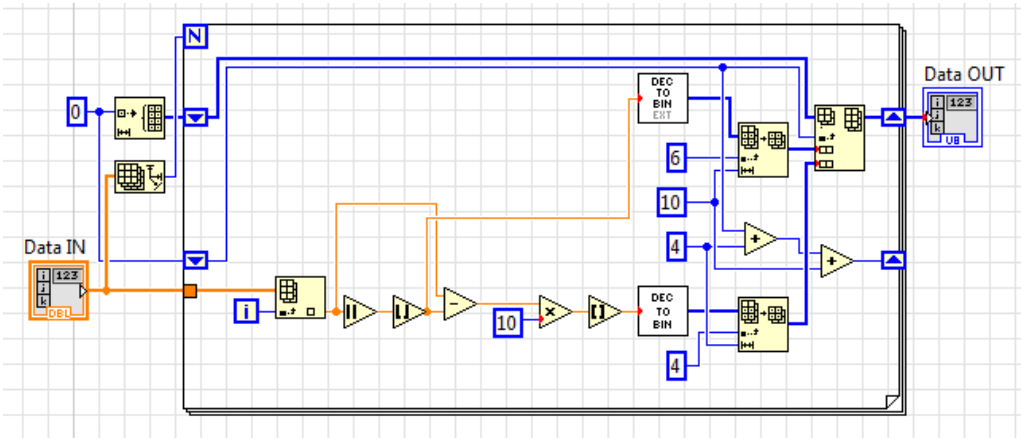


Fig. C.5.1. Transformación a binario de datos de housekeeping SID 1

Anexo D: Procesado de la entrada de datos

Para el procesado de una secuencia de bits y la comprobación de un formato de telecomando concreto se realiza una estructura secuencial con dos subsecuencias. En la primera de ellas se coge en *array* de bits entrante y, siguiendo la estructura de telecomando del capítulo *7. Estructura de telecomando y telemetría* se extraen cada uno de los campos (excepto los datos de aplicación) además de realizar el cálculo del tamaño de los datos de aplicación (*Application Data Length Bits*). Para extraer los bits correspondientes se emplean las funciones *Array Subset* e *Index Array*. A ambas se les pasa la posición del *array*. En el caso de la primera de ellas además se le pasa la longitud que se quiere extraer y devuelve un *array* de bits. En el caso de la segunda únicamente se obtiene el bit en dicha posición. El código es el mostrado en la siguiente figura:

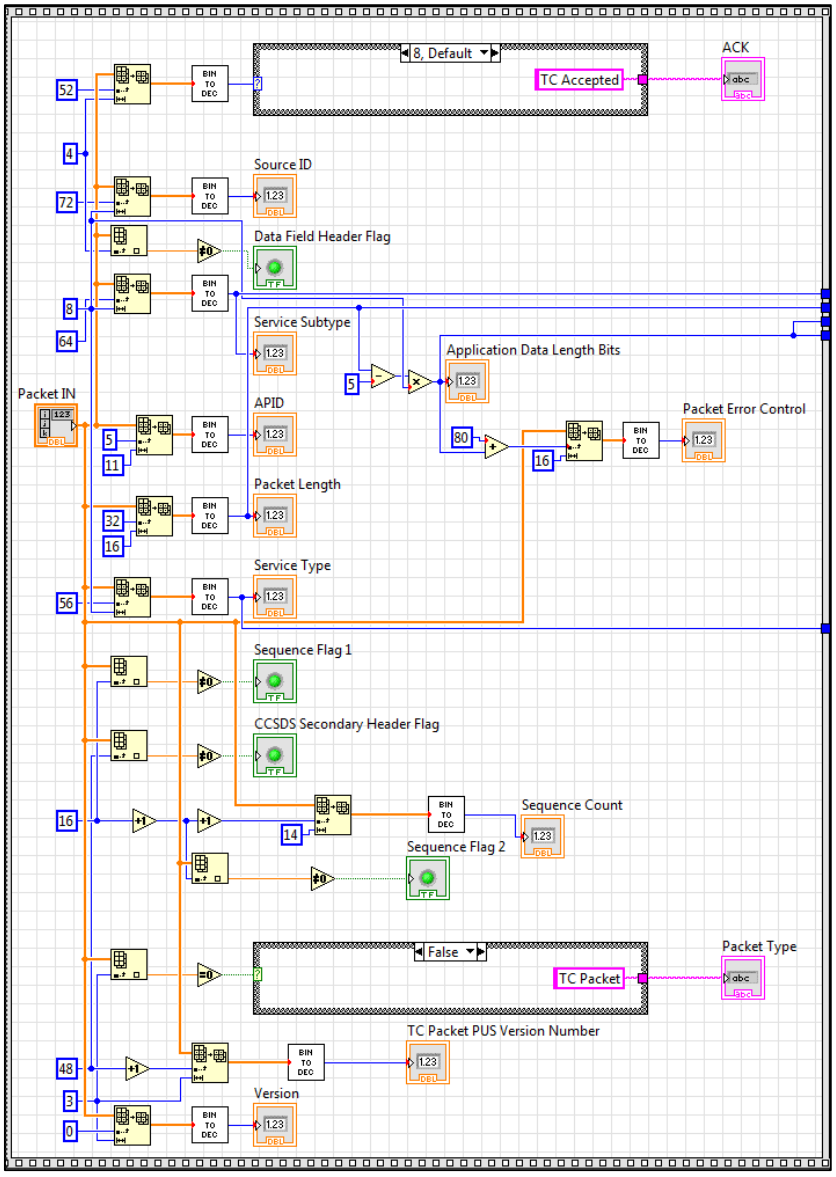


Fig. D.1. Primera subsecuencia del módulo de procesado

En la segunda subsecuencia se comprueba que todos los campos extraídos sean válidos de manera que se comprueba por un lado los campos que se mantienen constantes de la cabecera y por otro una serie de comprobaciones dentro de la función *format_error.vi*.

El funcionamiento general es el siguiente. Teniendo en cuenta el valor de los campos que se mantienen constantes de la cabecera del telecomando se emplean una serie de elementos de comparación (puertas *AND* y *NOT*, y función *Equal?*). En caso de que todo sea correcto se extraen los datos de aplicación desde la posición 80 del *array* entrante con longitud *Application Data Length Bits* y el código de error es el 7 (*no error*). En caso contrario se muestra el error 6 (indica que algún otro campo de la cabecera no es válido). El proceso descrito se muestra en la siguiente figura:

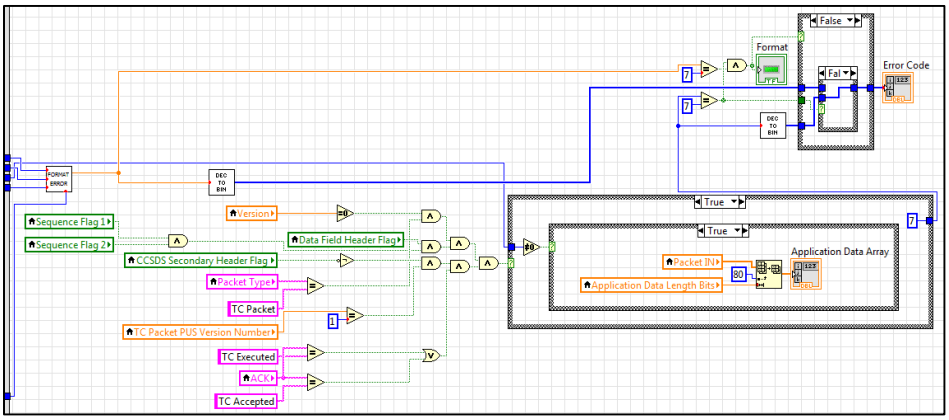


Fig. D.2. Segunda subsecuencia del módulo de procesado

Adicionalmente en *format_error.vi* se comprueba que el tamaño de los datos de aplicación sea el indicado, y que el tipo de servicio y el tipo de subservicio tengan un valor válido. El código es el siguiente:

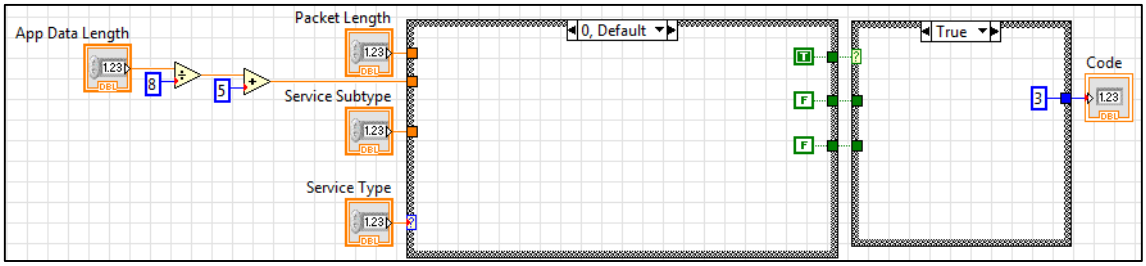


Fig. D.3. Comprobación del tipo de servicio

En caso de que el servicio no sea válido se ejecuta el código anterior devolviendo como código el 3. En caso de que el subservicio no sea válido se devuelve el código de error 4, tal y como se muestra en la siguiente figura.

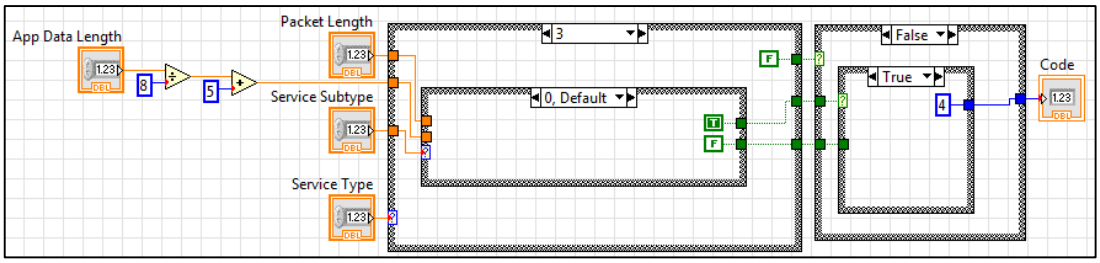


Fig. D.4. Comprobación del tipo de subservicio

En el caso de que el tamaño de los datos de aplicación calculado y el que se indica en el paquete sea distinto se devuelve el error 1, como se muestra en la siguiente figura.

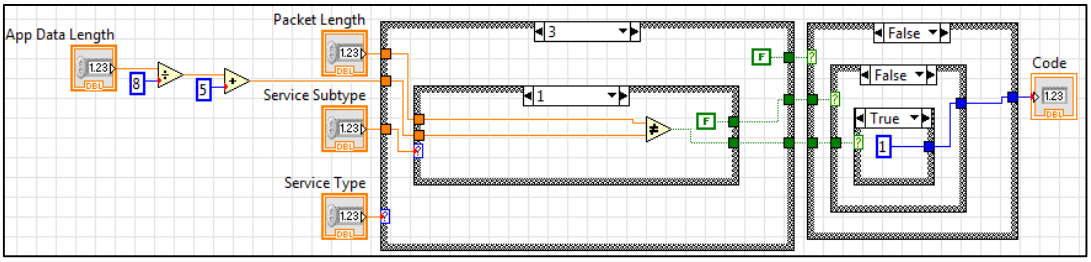


Fig. D.5. Comprobación del tamaño de los datos de aplicación

En el caso de que todos los campos sean correctos se devuelve el código de error 7 (*no error*), como se ilustra en la siguiente figura:

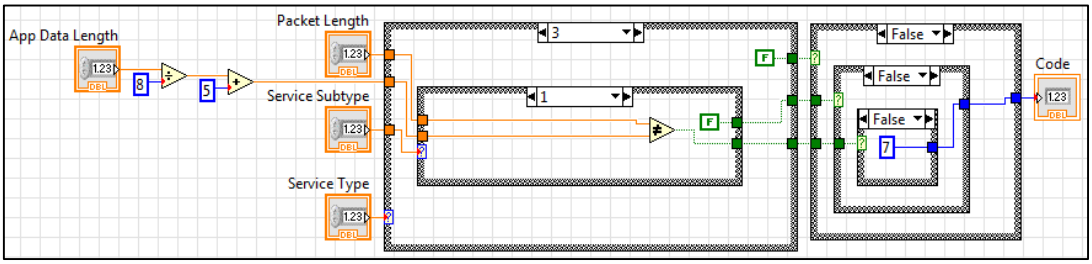


Fig. D.6. Devolución de código de error 7

Una vez obtenido el código de error, en caso de ser definitivamente el 7, se pone el indicador booleano de salida *Format* a verdadero. En caso de que el haya algún error en la cabecera se devuelve *false*.

Anexo E: Módulo de recepción

Para la recepción, dentro de *main.vi*, empleando el *USRP* se emplean los *subvi* listados a continuación:

- *sub_Ave_Power_And_Thresholding.vi*
- *sub_Check_Rx_Packet_Validity 2.vi*
- *sub_Chop_Packet.vi*
- *sub_est_noise_power - G1.vi*
- *sub_Extract_Packet_Boundaries - G.vi*
- *sub_Extract_Packets.vi*
- *sub_Init_PSK_At_Rx.vi*
- *sub_NoiseEst_And_Chop_Shell.vi*
- *sub_Reconstruct Data Message.vi*
- *sub_Resamp_Demod_Shell.vi*
- *sub_resample_and_demodulate.vi*
- *subCallConfigRx.vi*
- *subDCOffset.vi*
- *make_frame.vi**

El archivo con un asterisco ha sido creado para adaptar el funcionamiento del programa tomado como referencia [25] para el módulo de recepción.

En el *Panel Frontal* se configuran la IP del *USRP* (se emplea la misma que en transmisión), la antena de recepción (*RX1*), la frecuencia de transmisión (650 MHz), el factor de calidad (250 k símbolos/segundo), la ganancia (20 dB) y aspectos referentes a los paquetes (30 bits de guarda, 20 de sincronización y 300 bits de mensaje).

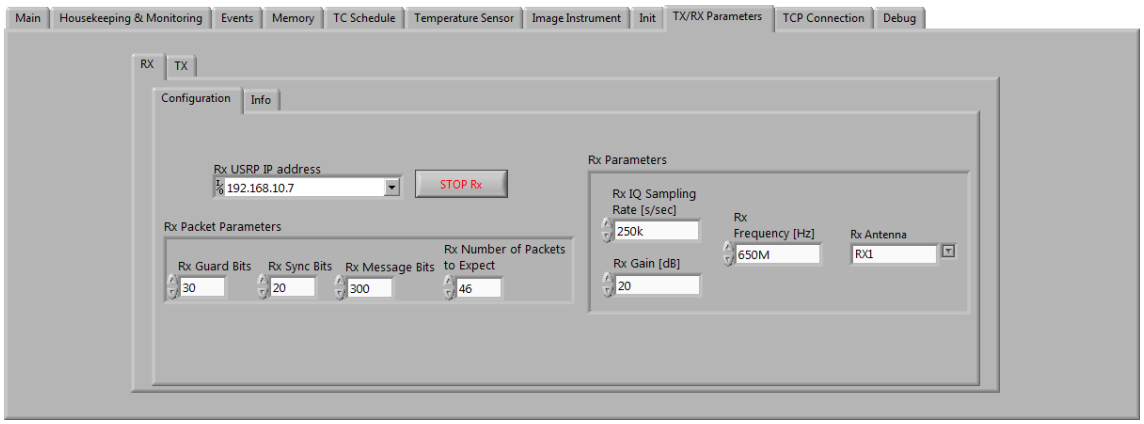


Fig. E.1. Pestaña de configuración de los parámetros de recepción

El tamaño de bits de mensaje se ha limitado a 300 porque nunca un telecomando recibido va a superar ese tamaño. En lo referente a la frecuencia

de recepción se emplea la más alta para la recepción debido a que es la estación base la que emplear mayor potencia en la transmisión debido a que el satélite tiene más limitada la generación de potencia. Todo ello se muestra en la figura ***Fig. E.2. Atribución de frecuencias.*** Como aclaración, las frecuencias empleadas son las expuestas para la realización de las pruebas, por lo que no simulan las frecuencias reales asignadas a los satélites.

En cuanto a la modulación se emplea una *BPSK (Binary Phase Shift Keying)* para simular el comportamiento real de transmisión y recepción de un satélite. Esta modulación consiste en el desplazamiento de fase de dos símbolos. Cada símbolo representa un único bit '0' o '1' y su diferencia de fase es máxima por lo que tiene mayor inmunidad al ruido y la tasa de bit es baja. Ambas características son muy importantes en las comunicaciones vía satélite, por lo que se ha decidido emplear esta modulación. En la figura ***Fig. E.3. Modulación BPSK*** se muestra el esquema de modulación general de la modulación *BPSK*.

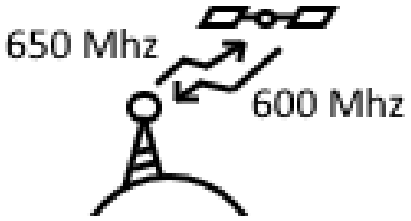


Fig. E.2. Atribución de frecuencias

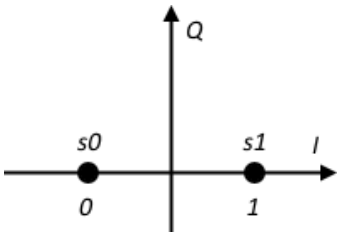


Fig. E.3. Modulación BPSK

El código empleado es el mostrado en la figura ***Fig. 6.1. Estructura general del módulo de recepción y procesado***. Para comprobar la llegada de un paquete se ejecuta lo siguiente:

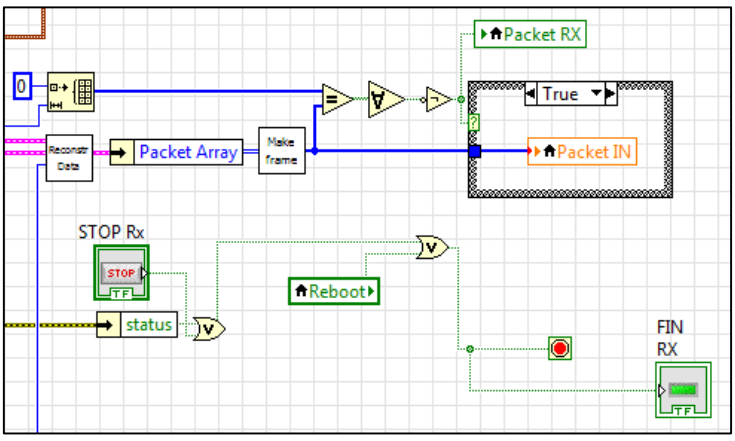


Fig. E.4. Detección de paquete e inserción en el módulo de procesado

En caso de que el *array* obtenido sea distinto de *todo ceros* (se comprueba bit a bit) se introduce el paquete en *Packet IN* y se pone a *true Packet RX.* Para

la conexión con el USRP se emplean las funciones mostradas en la siguiente figura:

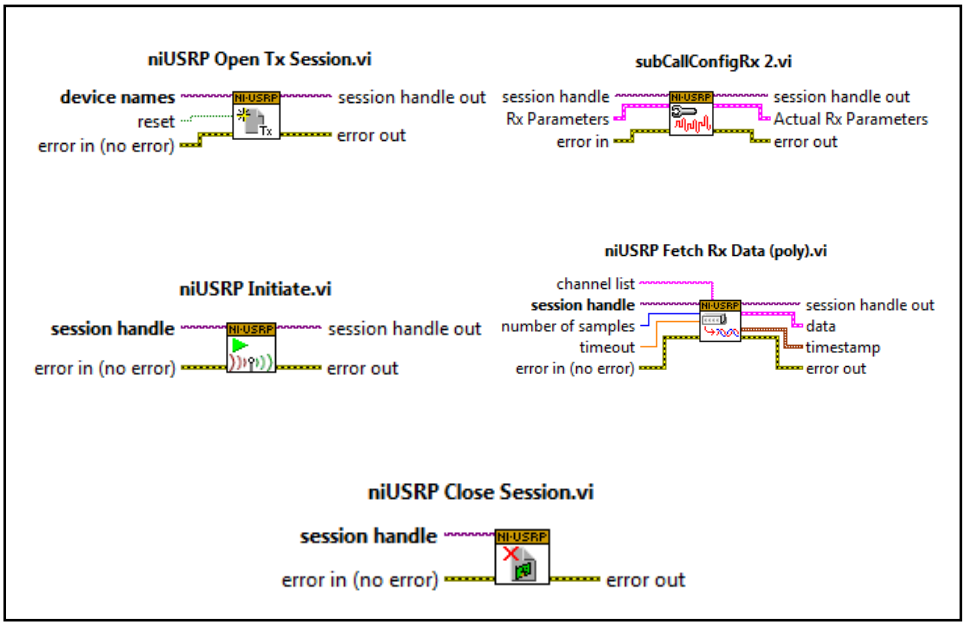


Fig. E.5. Funciones empleadas en la recepción

Anexo F: Módulo de transmisión

Para la transmisión empleando el *USRP* se ha creado un instrumento virtual llamado *transmission.vi* que emplea los *subvi* listados a continuación:

- *sub_calcpadding-symbolrate.vi*
- *sub_Create_Packet_Header.vi*
- *sub_Generate_Packet_Array.vi*
- *sub_padIQWF2.vi*
- *sub_PSKMod2.vi*
- *subCallConfigTx.vi*
- *gen_packets.vi**
- *num_packets.vi**

Los archivos con un asterisco han sido creados para adaptar el funcionamiento del programa tomado como referencia [25] para el módulo de transmisión.

En el *Panel Frontal* se configuran la IP del *USRP* (se emplea la misma que en recepción), la antena de transmisión (*TX1*), la frecuencia de transmisión (600 MHz), el número de repeticiones de cada paquete (cuatro), el factor de calidad (250 k símbolos/segundo), la ganancia (20 dB) y aspectos referentes a los paquetes (30 bits de guarda, 20 de sincronización y 776 bits de mensaje). Todo se muestra en la siguiente figura:

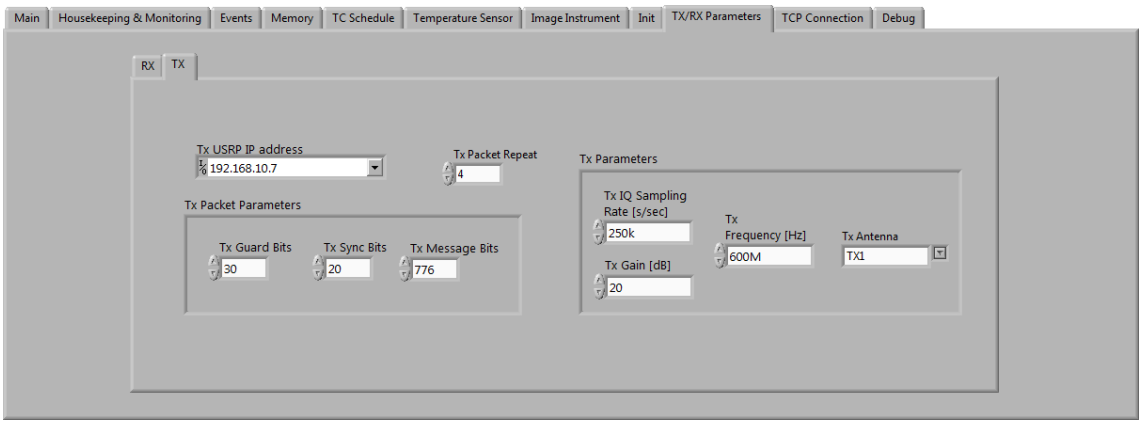


Fig. F.1. Pestaña de configuración de los parámetros de transmisión

Se emplean ese tamaño de bits de mensaje y ese número de repetición de paquetes porque en las pruebas más de ese valor se perdían bits de la cola y con menos repeticiones la imagen perdía muchos bloques de píxeles.

En lo referente a la frecuencia de transmisión empleada y de la modulación, sigue siendo válido lo expuesto en el *Anexo E: Módulo de recepción*.

El código empleado para la transmisión de los datos el de la figura ***Fig. F.4. Código del módulo de transmisión***. En él se cogen los valores de los indicadores del *Panel Frontal* descrito anteriormente, almacenados en sus respectivas variables globales. Además se introduce el *array* de bits que se quiere mandar en el controlador *Packet*. Para la conexión con el *USRP* se emplean las funciones mostradas en la siguiente figura:

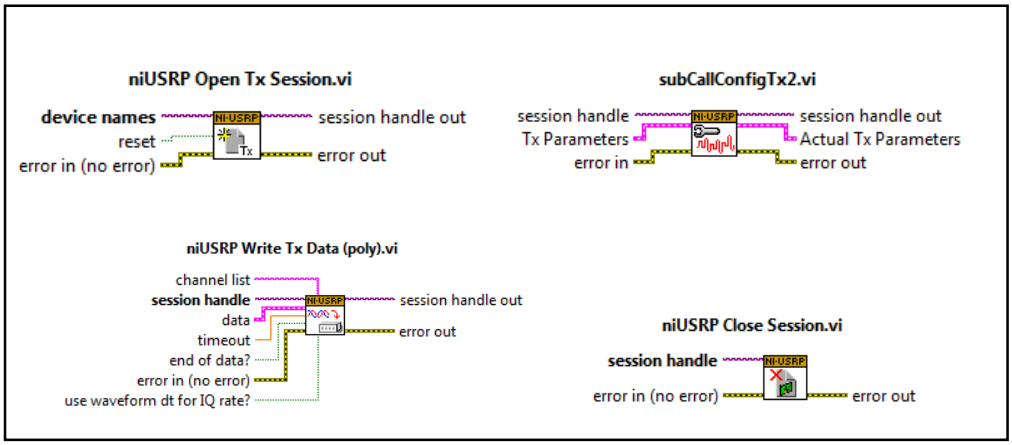


Fig. F.2. Funciones empleadas en la transmisión

Para la generación de los paquetes con secuencia de sincronización de *Galois* se emplea la función *gen_packets.vi*.

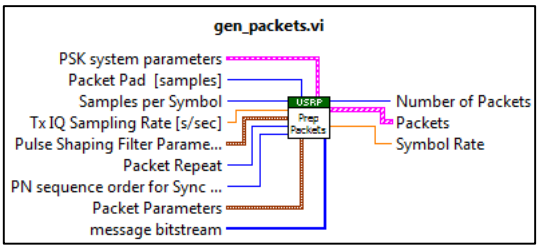


Fig. F.3. Función de generación de paquetes con sincronización

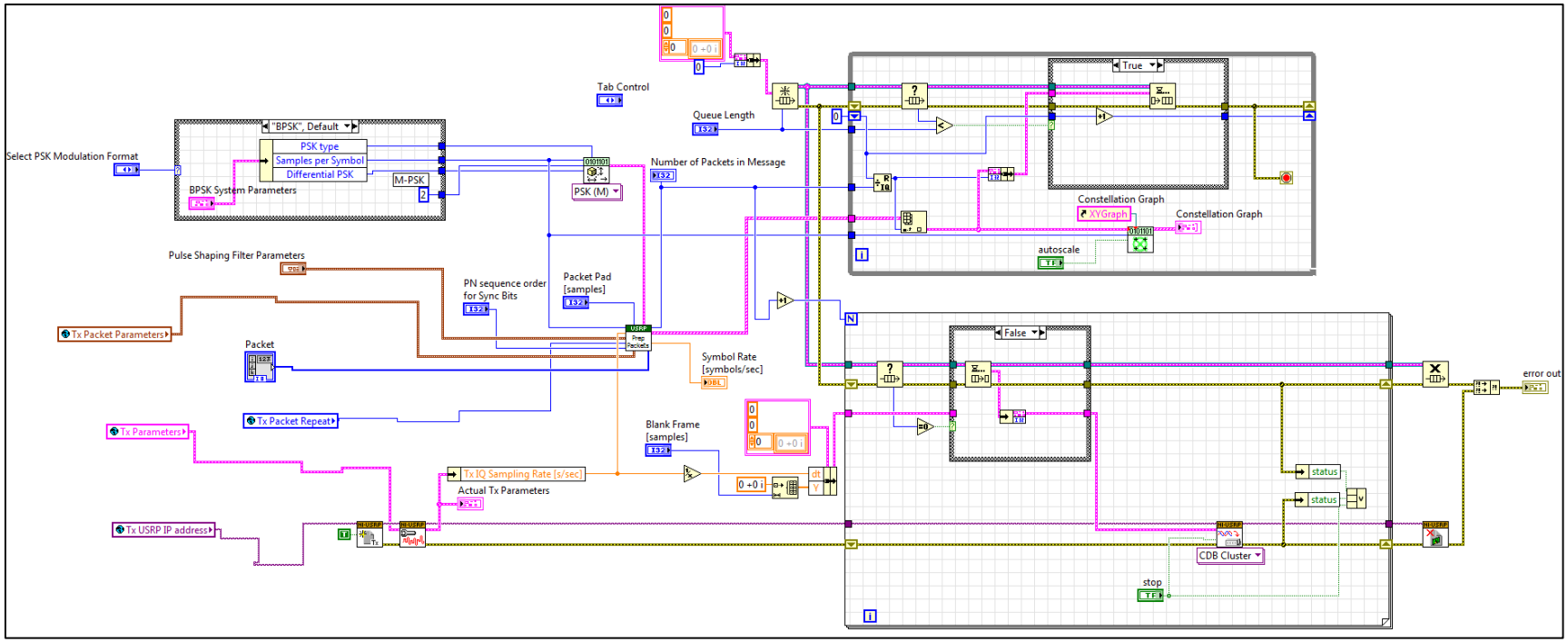


Fig. F.4. Código del módulo de transmisión

Anexo G: Inicialización visual y auditiva

En este anexo se explica la implementación llevada a cabo para el proceso de inicio del programa tanto en la primera ejecución como en las sucesivas a causa de los reinicios. El código que se ejecuta es el siguiente:

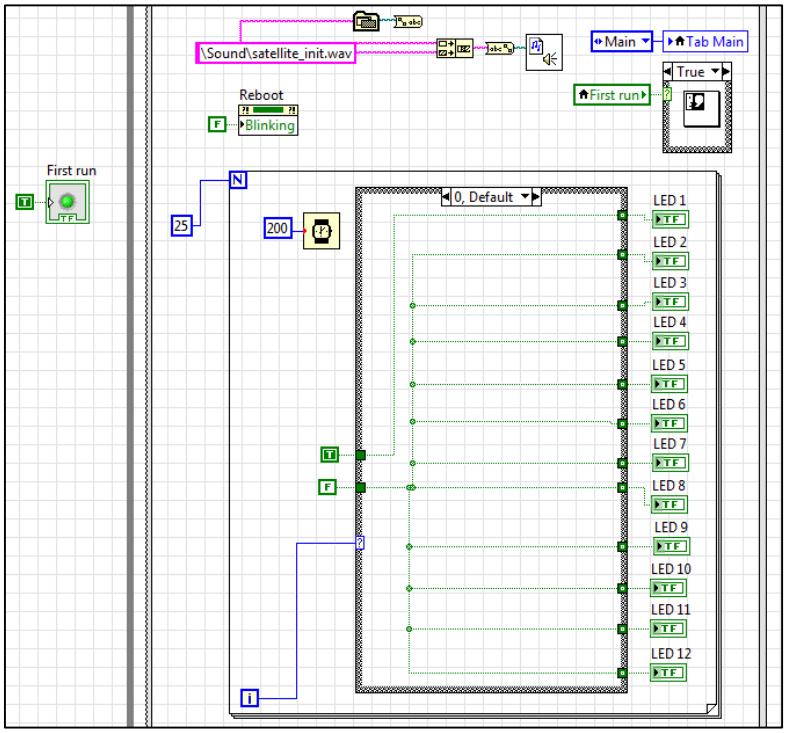


Fig. G.1. Código de la primera subsecuencia del programa principal

Como se puede observar en primer lugar se pone a *true* la variable local *First run* para indicar que el programa se ejecuta por primera vez. Luego se entra en el bucle principal del programa y más en concreto en la primera subsecuencia. En ella se cambia a la pestaña *Main* como la que se visualiza en *Tab Main*, se ejecuta la secuencia de LED de la parte superior del *Panel Frontal* y se pone a *false* la propiedad de parpadeo del indicador de *Reboot* (se ejecuta por si viene del estado de reinicio para que pare de parpadear y se apague indicando que se ha concluido con el reinicio). Además se reproduce un sonido y, en caso de ser la primera ejecución, es decir, que no venga de un reinicio, sale una ventana emergente con el logo y una barra de progreso.

Para llevar a cabo la secuencia de LED se ejecuta un caso según el valor de la iteración (desde 0 hasta 24). En cada caso se enciende un LED (poniendo su variable local a verdadero y el resto a falso) siguiendo la siguiente secuencia:

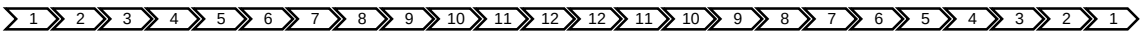


Fig. G.2. Secuencia de LED de inicio

Cada iteración se ejecuta pasados 200 milisegundos y tras la última iteración se produce una espera. En total, la ejecución de esta secuencia dura cinco segundos. Para lograrlo, dentro del bucle *for* se emplea la función *Wait (ms),* como se muestra en la figura:

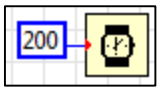


Fig. G.3. Función de espera de la secuencia de LED

En lo referente al sonido, para la obtención de la ruta del archivo de sonido se emplean las funciones *Application directory.vi* (obtiene el *path* del proyecto), *Path To String*, *Concatenate Strings* (para concatenar al directorio raíz el subdirectorio *Sound* y el nombre del archivo *satellite_init.wav*) y *String To Path.* Para la reproducción del sonido se emplea la función *Play Sound File.vi,* a la que se le pasa por parámetro la ruta obtenida anteriormente para que reproduzca de manera inmediata el sonido.

En último lugar, para mostrar la pantalla emergente se emplea el archivo *dialog_init.vi*. En caso de que sea la primera ejecución se muestra, en caso contrario no se vuelve a mostrar. Este archivo muestra un *Dialog Box* con el siguiente aspecto:

Fig. G.4. Pantalla de inicio emergente

El código asociado es el que se muestra a continuación:

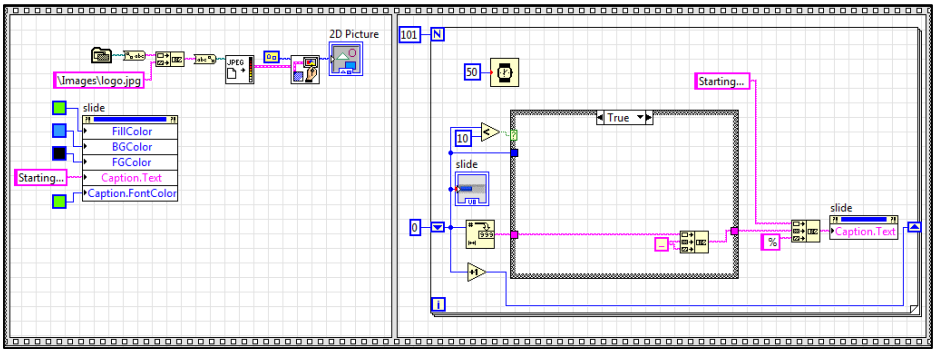


Fig. G.5. Ejecución de la barra de progreso de dialog_init.vi

Como se puede ver, en primer lugar se carga la imagen de fondo obteniendo el directorio de la misma manera que se ha explicado anteriormente. Luego se emplean las funciones *Read JPEG File.vi*, *Empty Picture.vi* y *Draw Flattened Pixmap.vi* para mostrar la imagen igual que se describe en el servicio de imágenes. Además se cambian los colores de fondo (negro) y progreso (verde) de la barra de progreso (*slide*), así como el texto y su color (verde).

En segundo lugar, se carga la barra de progreso inferior y se va cambiando el texto mostrando el porcentaje correspondiente ejecutado un bucle *for* desde 0 hasta 100 cada 50 milisegundos, por lo que el tiempo de carga total es de 5 segundos coincidiendo de esta manera con el tiempo de ejecución de la secuencia de LED. En el caso de esta barra, para que no se desplace el texto asociado en cada nueva ejecución se introducen uno o dos espacios en el texto en caso de que el valor numérico de carga sea menor a 10 y 100 respectivamente. Esto se ejecuta con una sucesión de *Case Structure* en el interior del bucle.

Anexo H: Proceso de inicialización de los registros y la barra de progreso

En la subsecuencia de inicialización del programa principal se encuentra la siguiente estructura secuencial referente a inicialización de los registros y la barra de progreso:

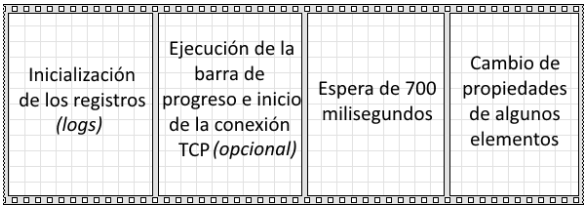


Fig. H.1. Estructura de a inicialización de los registros y la barra de progreso

En la pimera de las subsecuencias se ponen *Event Log* y *TC Schedule Log* vacíos y se les cambian algunas propiedades como el color de fuente (azul y verde respectivamente) empleando elmentos *Property Node.* El tipo de fuente empleado en todos los registros es *Consolas.* En el caso de *Summary Log* y *Log Window* se pone el tamaño de fuente a 2 para poder representar una imagen con texto. Dicha imagen se encuentra en un fichero de texto con la secuencia de caracteres correspondiente en el directorio *\Files.* Para *Log Window* se emplea el fichero *image_intro.txt* y para el de resumen se emplea el fichero *image_intro_summary.txt.* A continuación se muestra el código de esta subsecuencia:

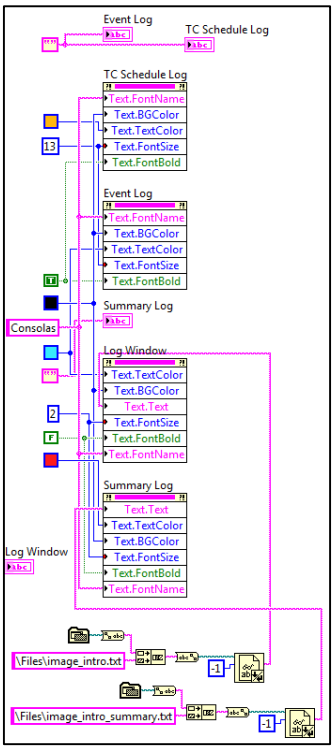


Fig. H.2. Inicialización de los registros

Para extraer la ruta se realiza la misma operación que en casos anteriores de trabajo con archivos exteriores y se introducen, junto con el campo *Count* a -1 para que lea todo el fichero, en la función *Read from Text File.vi*, la cual devuelve el *string* que se introduce en el nodo de propiedades correspondiente. A continuación se muestra el aspecto de los ficheros de texto.

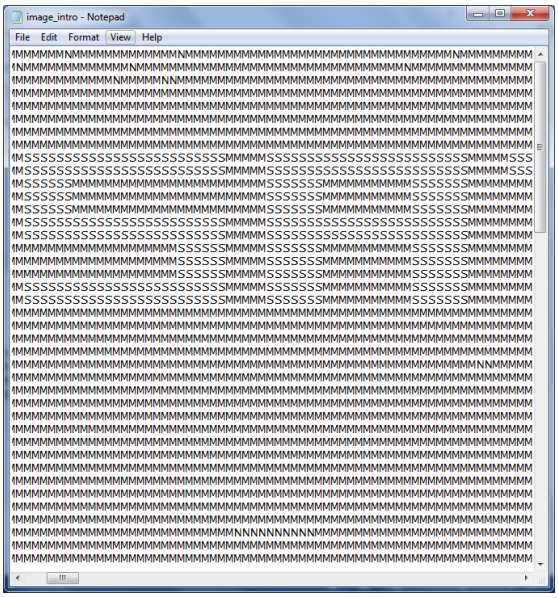


Fig. H.3. Archivo image_intro.txt

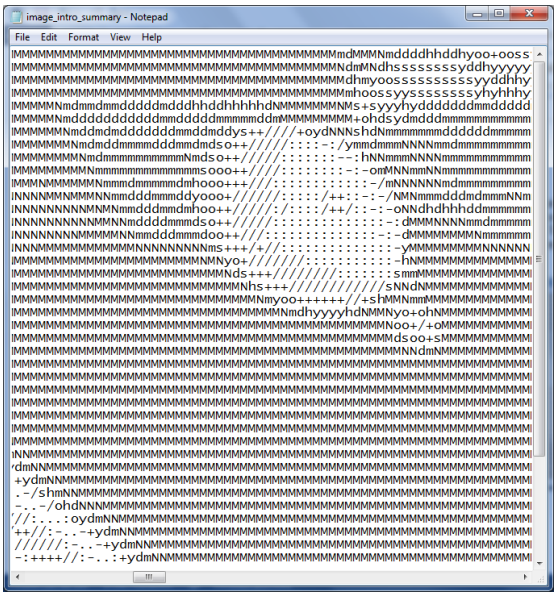


Fig. H.4. Archivo image_intro_summary.txt

Su representación en las ventanas de registro queda de la siguiente manera:

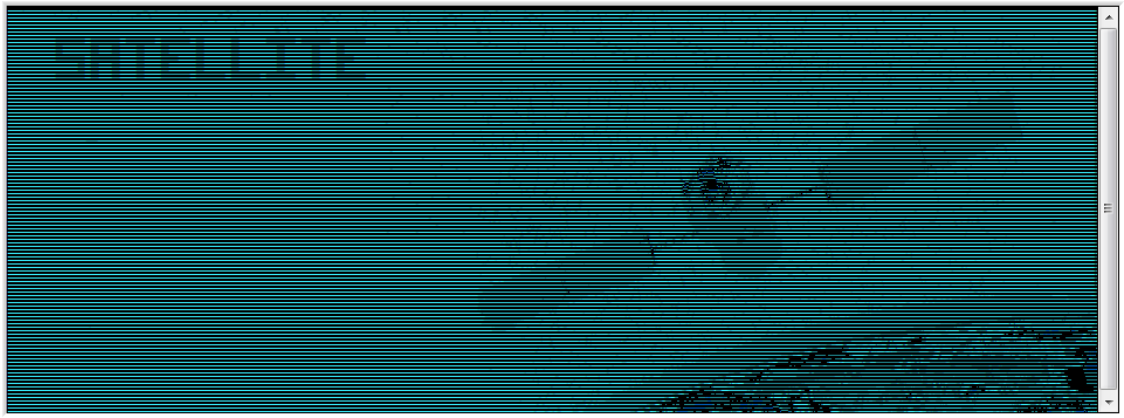


Fig. H.5. Representación inicial en Log Window

Fig. H.6. Representación inicial en Summary Log

En lo referente a la segunda subsecuencia, se carga la barra de progreso de 0 a 100, ejecutándose una vez cada 30 milisegundos de modo que tarda 3 segundos en cargarse. En el caso de que se quiera establecer la conexión TCP lo que se hace es tener la barra continuamente cargando y sin esperas hasta que se ha conectado con el dispositivo remoto (llega a 100 y se vuelve a poner la barra a 0, así sucesivamente). Además, se muestra el progreso en texto en forma de porcentaje y la barra en color verde. El proceso de conexión TCP está explicado en el *Anexo I: Conexión TCP y aplicación Android*. Ambos casos son los contemplados en la siguiente figura.

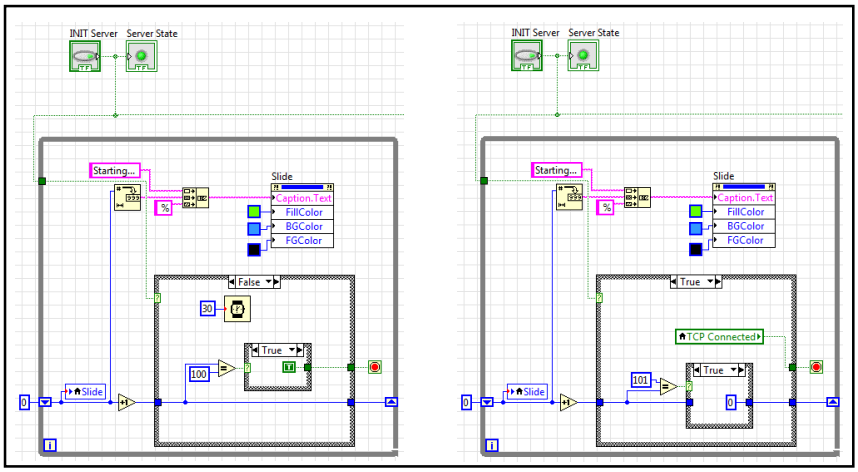


Fig. H.7. Ejecución de la barra de progreso

En la tercera subsecuencia se realiza una espera de 70 milisegundos y en la siguiente se vuelven a modificar algunas propiedades de los registros (el texto de inicio, el color de fuente a verde y el tamaño de texto a 12), tal y como se ilustra a continuación:

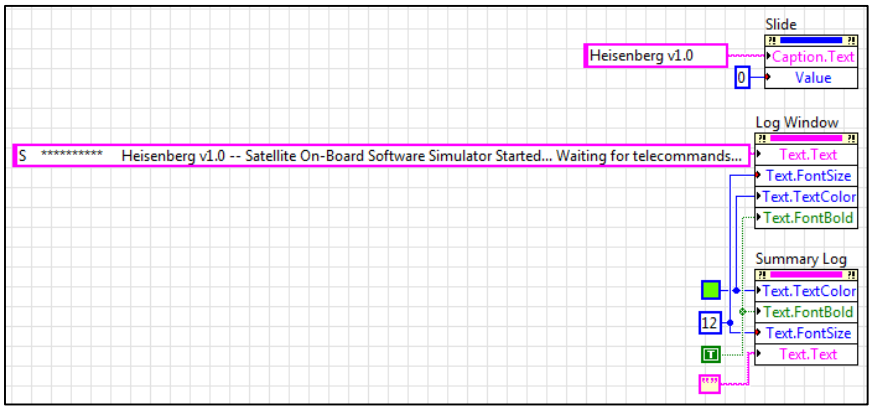


Fig. H.8. Cambio de propiedades de Slide, Log Window y Summary Log

Anexo I: Conexión TCP y aplicación Android

Adicionalmente, se ha implementado un servidor TCP en *main.vi* que conecta con un cliente Android. Como consideración general, para poder realizar la conexión, ambos dispositivos deben estar dentro de la misma subred. Como no es objetivo fundamental de este proyecto se va a explicar de manera muy general.

En primer lugar se describe el funcionamiento del servidor en *LabVIEW.* El primer paso es esperar a que el cliente solicite la conexión en la parte de inicialización del sistema. En el *Panel Frontal*, dentro del panel principal en la pestaña *TCP Connection* se pueden configurar los puertos de transmisión y recepción de la conexión TCP, tal y como se observa en la siguiente figura:

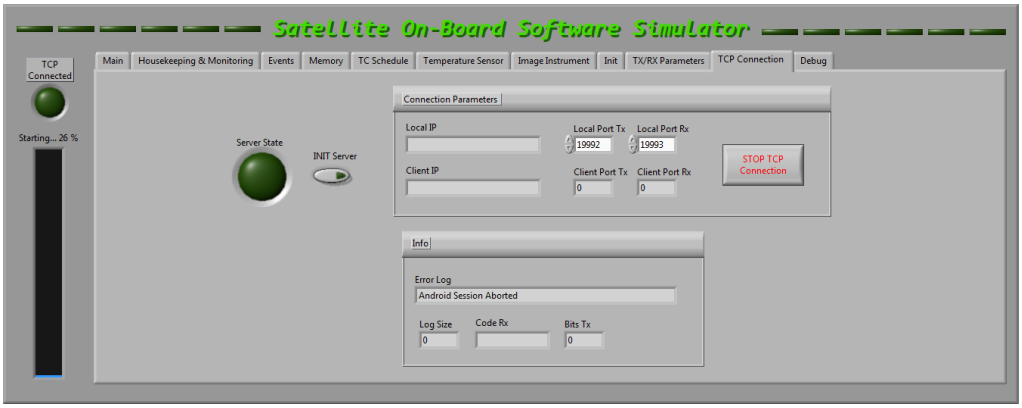


Fig. I.1. Pestaña TCP Connection

En el caso de que el botón *INIT Server* esté pulsado al iniciar el programa, el sistema se mantendrá esperando hasta que se reciba una conexión entrante para ambos puertos:

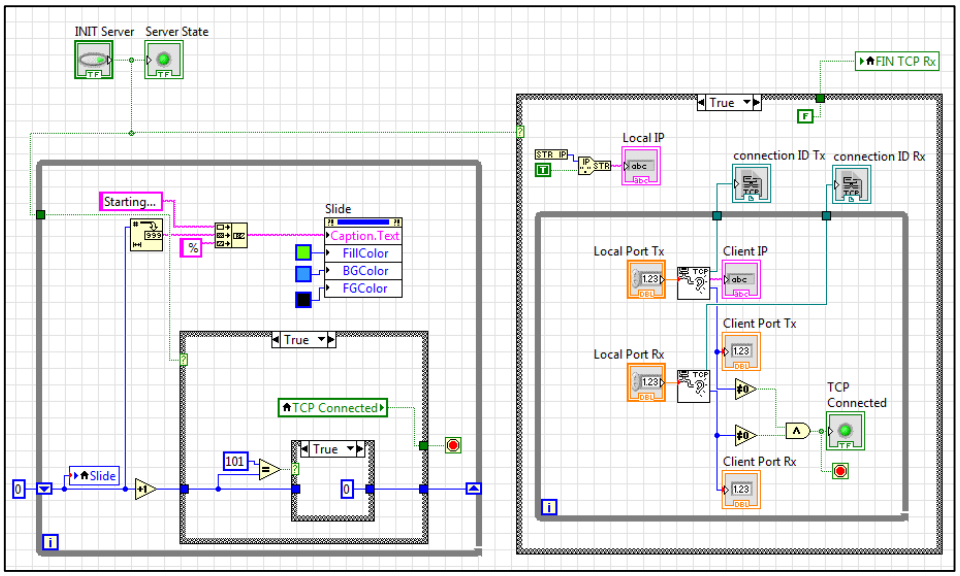


Fig. I.2. Ejecución de la espera de conexión TCP

Como se observa, se almacenan la referencia de ambas conexiones (una de recepción y otra de transmisión), obtenida mediante la función *TCP Listen.vi* que se mantiene escuchando en el puerto correspondiente de manera indefinida. Una vez que se ha establecido conexión el puerto cambia de cero al valor remoto. También se obtiene la IP remota. Una vez conectado se enciende el indicador *TCP Connected* situado encima de la barra de progreso en el *Panel Frontal*.

Una vez conectado el dispositivo, por un lado se manda el texto que se va introduciendo en *Log Window* al cliente así como la indicación de que se ha reiniciado el sistema en caso de producirse tal evento. Por otro lado, se reciben órdenes del cliente para modificar el estado de seis botones (pulsado o no pulsado), el indicador de cierre de la aplicación remota o solicitar el reporte de eventos del *Servicio 5* (*TM (5,1), TM (5,2), TM (5,3)* y *TM (5,4)).* Para la implementación de la transmisión se realiza lo el código de la siguiente figura, dentro del bucle del *Módulo 1* en la tercera subsecuencia (sólo se ejecuta si hay conexión TCP abierta).

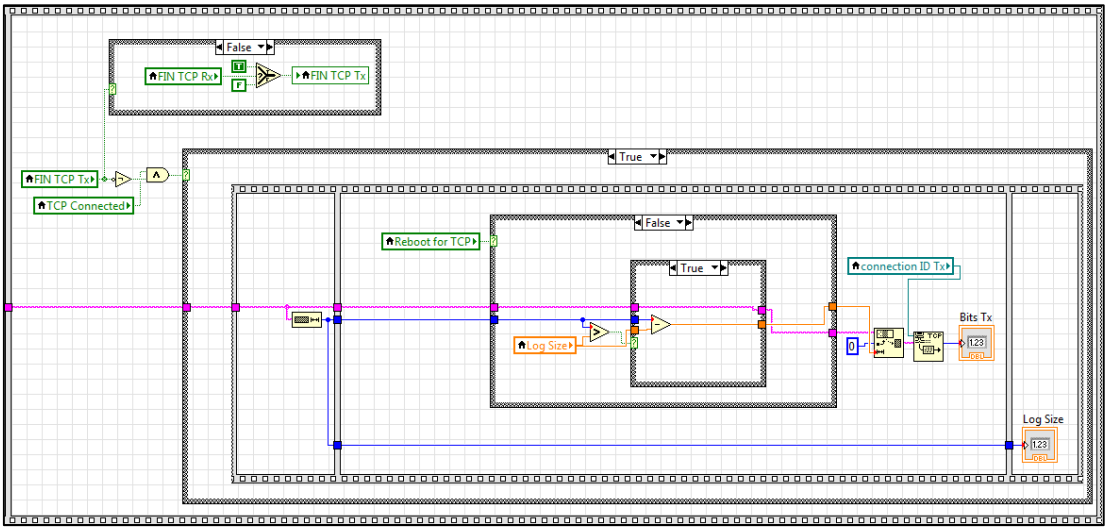


Fig. I.3. Transmisión TCP

Como se observa, se coge la variable local *Log Window* y se mira su tamaño. Si es distinto al registrado se envía el fragmento del *string* (se extrae con la función *String Subset* con *Offset* con valor cero y *Length* la diferencia entre el nuevo tamaño y el registrado anteriormente) empleando la conexión de salida con la función *TCP Write*. En caso de que se haya limpiado *Log Window* (con el botón *CLEAR LOG* de la pestaña Main) se envía dicho evento al dispositivo remoto para que realice la misma operación. El *string* mandado es “clear”. Para detectar dicho evento se mira si el tamaño del registro es mayor que el del tamaño de *Log Window.* Luego se modifica el tamaño del registro que lleva el tamaño mandado hasta ese momento en *Log Size*.

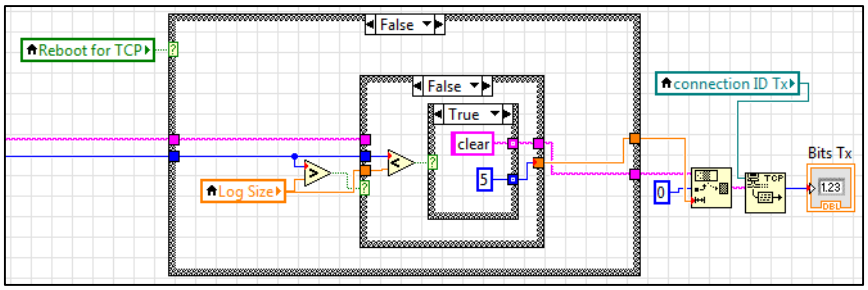


Fig. I.4. Transmisión del comando "clear"

En lo referente a la recepción el código implementado es el que se muestra a continuación:

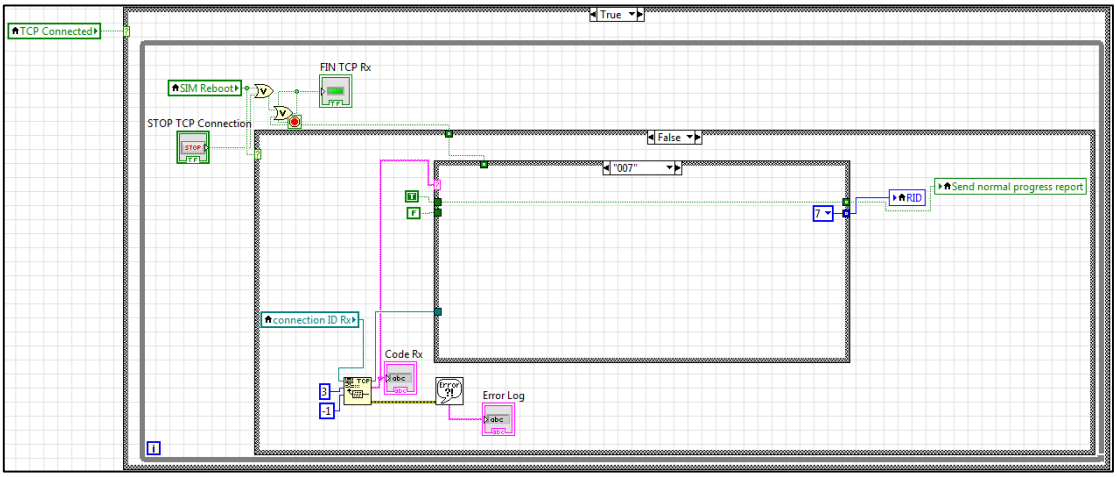


Fig. I.5. Recepción TCP

En el caso de que se haya establecido conexión, se ejecuta un bucle *while* que se mantiene constantemente escuchando en el puerto de recepción del sistema empleando la función *TCP Read* que emplea la referencia de conexión obtenida en la inicialización, escucha de manera indefinida hasta que recibe un mensaje "reboot" del sistema remoto y lee 3 bytes (tamaño de los códigos intercambiados). Dependiendo del mensaje recibido entra en una cláusula del *Case* y se ejecuta un código. En caso de que se reinicie el sistema se manda un mensaje de "close" y se cierra la conexión. En el nuevo inicio del sistema se vuelve a realizar el mismo proceso de conexión descrito al principio de este anexo. Los 30 códigos de acción son los descritos en la siguiente tabla:

Código	Acción
007	Reporte TM (1,1) RID 7
008	Reporte TM (1,1) RID 8
009	Reporte TM (1,1) RID 9
062	Reporte TM (1,1) RID 62
063	Reporte TM (1,1) RID 63
064	Reporte TM (1,1) RID 64

085	Reporte TM (1,1) RID 85
086	Reporte TM (1,1) RID 86
087	Reporte TM (1,1) RID 87
107	Reporte TM (1,2) RID 7
108	Reporte TM (1,2) RID 8
109	Reporte TM (1,2) RID 9
162	Reporte TM (1,3) RID 62
163	Reporte TM (1,3) RID 63
164	Reporte TM (1,3) RID 64
185	Reporte TM (1,4) RID 85
186	Reporte TM (1,4) RID 86
187	Reporte TM (1,4) RID 87
201	Pulsación de SIM Acceptance Failure
202	Pulsación de SIM Execution Failure
203	Pulsación de SIM Memory Check Failure
204	Pulsación de SIM Reboot
205	Pulsación de SIM Test Instrument Sensor Failure
206	Pulsación de SIM Test Instrument Image Failure
301	Des pulsación de SIM Acceptance Failure
302	Des pulsación de SIM Execution Failure
303	Des pulsación de SIM Memory Check Failure
304	Des pulsación de SIM Reboot
305	Des pulsación de SIM Test Instrument Sensor Failure
306	Des pulsación de SIM Test Instrument Image Failure
222	Cierre de conexión por parte del cliente
close	*Cierre de conexión por parte del servidor*
clear	*Borrado de Log Window*

Tabla I.1. Códigos de acción de la conexión TCP

El servidor sólo envía mensajes “close” y “clear”. El resto de mensajes son los mandados por el cliente.

En lo referente a la aplicación cliente se emplea Android como lenguaje de programación para ser empleada en un dispositivo móvil. Se ha usado el programa *Eclipse*, en el cual se ha desarrollado el proyecto. Como referencia se usa el *API* de Android [21]. La estructura del proyecto es la mostrada en la siguiente figura en la que se han creado dos clases principales *Splash.java* y *MainActivity.java*, el manifiesto de la aplicación *AndroidManifest.xml* (***Fig. I.33.***), un archivo *strings.xml* en el que se incluyen las constantes de texto empleadas en los *layouts* (***Fig. I.34.***) y dos archivos de tipo *layout* llamados *splash.xml* y *main.xml*.

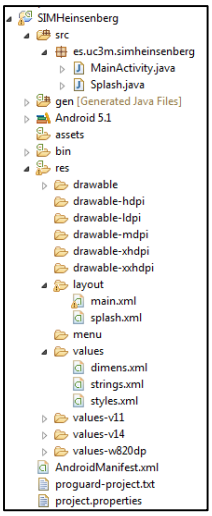


Fig. I.6. Estructura del proyecto en Eclipse

Además se han incluido varios archivos de imagen para representar los botones, las pestañas, el logo inicial y la imagen del icono de la aplicación. Todos ellos se incluyen en la carpeta *drawable* del proyecto. Los archivos son los siguientes:

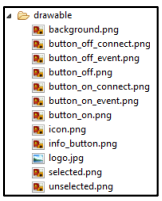


Fig. I.7. Recursos drawable

En primer lugar se muestra una pantalla de inicio durante tres segundos (esta parte se implementa en *Splash.java* junto con *splash.xml*). Luego se ejecuta la aplicación, implementada en *MainActivity.java* junto con *main.xml.*

En lo referente a la pantalla de inicio, se representa la imagen *logo.jpg.* A continuación se muestra una captura de dicha pantalla inicial:

Fig. I.8. Pantalla de inicio de la aplicación móvil

En cuanto a su implementación, se hereda de la clase *Activity* y dentro del método de creación se crea un *Thread* con la espera. Cuando concluye se crea un *Intent* que pasa inicia la actividad principal y se concluye con la ejecución de *Splash*. El código asociado es el mostrado en las figuras ***Fig. I.15.*** y ***Fig. I.16.***

En cuanto a la aplicación principal en la pantalla hay tres pestañas. En la primera se modifican los parámetros de conexión con el servidor (IP y puertos). En la segunda pestaña se puede ver el mismo log exactamente que en *Log Window* en el ordenador. En la tercera pestaña se pueden ejecutar varias acciones de manera remota pulsando los diez botones que componen dicha pestaña así como la selección del *RID* en el caso del servicio de eventos. En las figuras ***Fig. I.9.*** a ***Fig. I.14.*** se muestran algunas capturas de la aplicación durante su ejecución.

En lo referente al código, el proceso de creación de la *Activity* es el mismo que en la otra clase pero cargando el otro *layout*. Como característica reseñable se encuentra la creación de dos hilos de ejecución, uno para la recepción y otro para la transmisión. En cada uno de ellos se crea un objeto *Socket* al que se le pasa por parámetro la IP remota y el puerto remoto correspondiente. Para la escritura y lectura de los datos del canal de transmisión se emplean las clases *OutputStream* e *InputStream*. La lectura se ejecuta constantemente en un bucle *while* hasta que se cierra la conexión. La transmisión en cambio sólo se ejecuta cuando se produce un evento (se pulsa alguno de los botones). Además para una mejor experiencia del usuario se muestran elementos de la clase *Toast* que son los mensajes emergentes que aparecen en pantalla. En cuanto a la escritura en el *log* se emplea un hilo dentro de la planificación del sistema que se ejecuta cada medio segundo.

La generación del archivo con extensión *.apk* ha sido creado con *Eclipse* con validez de 25 años.

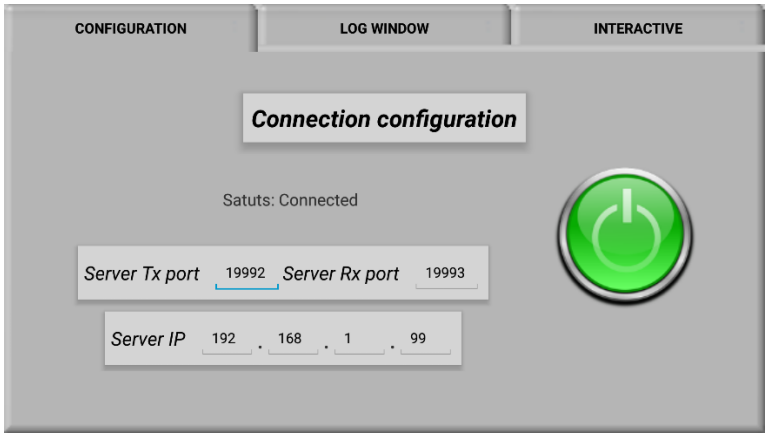


Fig. I.9. Conexión realizada con éxito

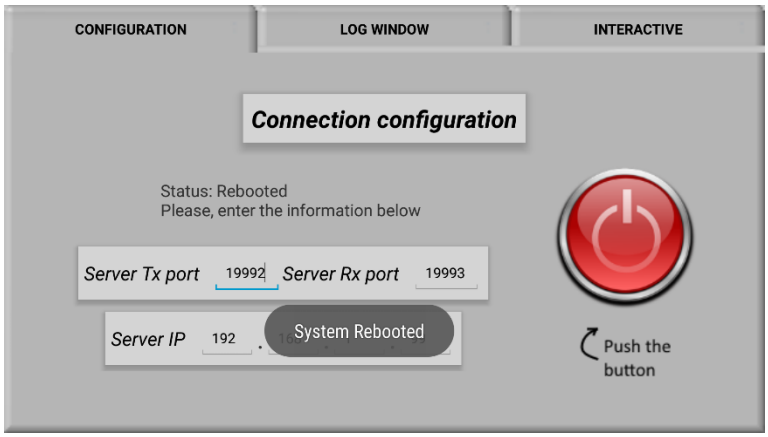


Fig. I.10. Reinicio del sistema

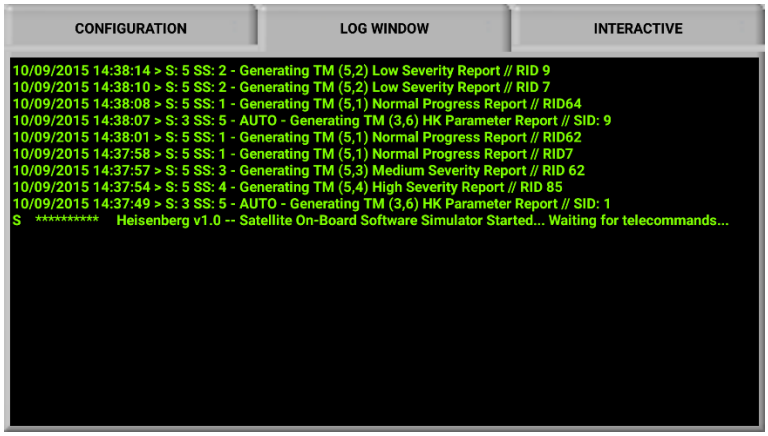


Fig. I.11. Pestaña de LOG WINDOW en ejecución

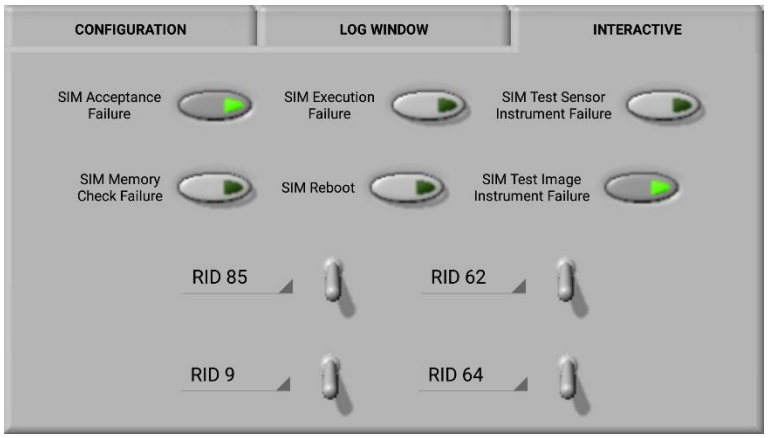


Fig. I.12. Pestaña de INTERACTIVE en ejecución

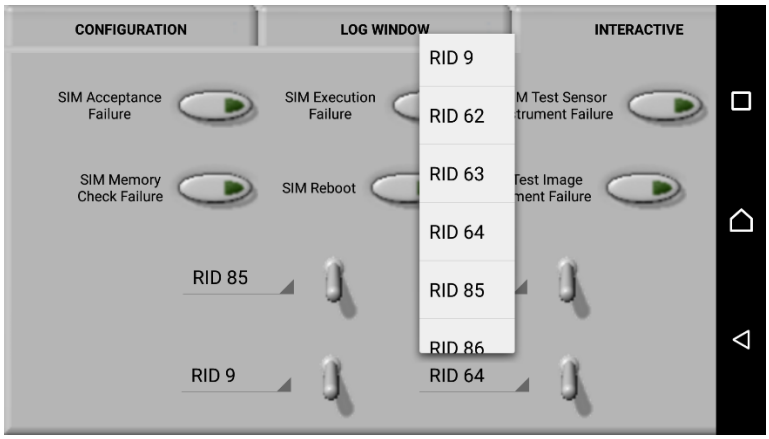


Fig. I.13. Selección de RID para generación de evento

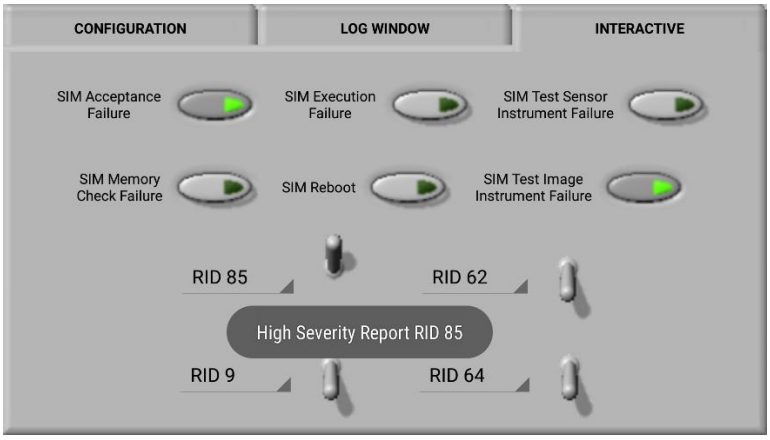


Fig. I.14. Toast de generación de reporte de RID 85

```java
package es.uc3m.simheinsenberg;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Splash extends Activity {

	protected boolean active = true;
	protected int splashTime = 3000;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		View decorView = getWindow().getDecorView();
		decorView.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_STABLE
				| View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
				| View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
				| View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
				| View.SYSTEM_UI_FLAG_FULLSCREEN
				| View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
		setContentView(R.layout.splash);

		Thread splashThread = new Thread() {
			@Override
			public void run() {
				try {
					int waited = 0;
					while (active && (waited < splashTime)) {
						sleep(100);
						if (active) {
							waited += 100;
						}
					}
				} catch (InterruptedException e) {
				} finally {
					openApp();
				}

			}
		};
		splashThread.start();
	}
```

Fig. I.15. Splash.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:background="@drawable/logo" >
```

Fig. I.16. splash.xml

En cuanto al manifiesto, se requiere la versión 19 de Android así como permisos de acceso a Internet y del estado de conexión durante la instalación. También se incluyen las actividades implementadas y como orientación de la pantalla, en apaisado.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.uc3m.simheinsenberg"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
    </uses-permission>

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".Splash"
            android:screenOrientation="landscape" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:screenOrientation="landscape" >
```

Fig. I.17. Archivo AndroidManifest.xml

A continuación se muestra el archivo con las constantes de texto empleadas en los *layout*. Se ha creado dicho archivo siguiendo el consejo de ser una buena práctica en el *API* de Android [21].

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Heisenberg\nSimulator</string>
    <string name="string_acceptance_failure">SIM Acceptance\nFailure</string>
    <string name="string_execution_failure">SIM Execution\nFailure</string>
    <string name="string_sensor_failure">SIM Test Sensor\nInstrument Failure</string>
    <string name="string_memory_failure">SIM Memory\nCheck Failure</string>
    <string name="string_reboot">SIM Reboot</string>
    <string name="string_image_failure">SIM Test Image\nInstrument Failure</string>
    <string name="string_point">.</string>
    <string name="description_acceptance">Acceptance Failure Button</string>
    <string name="description_execution">Execution Failure Button</string>
    <string name="description_memory">Memory Failure Button</string>
    <string name="description_sensor">Sensor Instrument Failure Button</string>
    <string name="description_image">Image Instrument Failure Button</string>
    <string name="description_reboot">Reboot Button</string>
    <string name="description_high_report">High Report Button</string>
    <string name="description_low_report">Low Report Button</string>
    <string name="description_medium_report">Medium Report Button</string>
    <string name="description_normal_report">Normal Report Button</string>
    <string name="description_connect">Connect or Disconnect Button</string>
    <string name="description_info">Push the Button Image</string>
    <string name="string_log_window">Waiting for TCP Connection&#8230;</string>
    <string name="string_title">Connection configuration</string>
    <string name="string_info">Please, fill the information below</string>
    <string name="string_tx_port">Server Tx port</string>
    <string name="string_rx_port">Server Rx port</string>
    <string name="string_server_ip">Server IP</string>
    <string name="default_tx_port">19992</string>
    <string name="default_rx_port">19993</string>
    <string name="default_ip_1">192</string>
    <string name="default_ip_2">168</string>
    <string name="default_ip_3">1</string>
    <string name="default_ip_4">99</string>
    <string-array name="event_report_normal">
        <item>RID 7</item>
        <item>RID 8</item>
        <item>RID 9</item>
        <item>RID 62</item>
        <item>RID 63</item>
        <item>RID 64</item>
        <item>RID 85</item>
        <item>RID 86</item>
        <item>RID 87</item>
    </string-array>
    <string-array name="event_report_low">
        <item>RID 7</item>
        <item>RID 8</item>
        <item>RID 9</item>
    </string-array>
    <string-array name="event_report_medium">
        <item>RID 62</item>
        <item>RID 63</item>
        <item>RID 64</item>
    </string-array>
    <string-array name="event_report_high">
        <item>RID 85</item>
        <item>RID 86</item>
        <item>RID 87</item>
    </string-array>
</resources>
```

Fig. I.18. Archivo strings.xml

Anexo J: Manual de usuario

En este anexo se presenta un ejemplo de ejecución del programa. En primer lugar sin conexión TCP y luego conectando el móvil.

Primero, antes de arrancar el programa hay que configurar los parámetros de inicialización del sistema. Esta inicialización se lleva a cabo en la pestaña *Init*. Se recomienda dejar los valores por defecto.

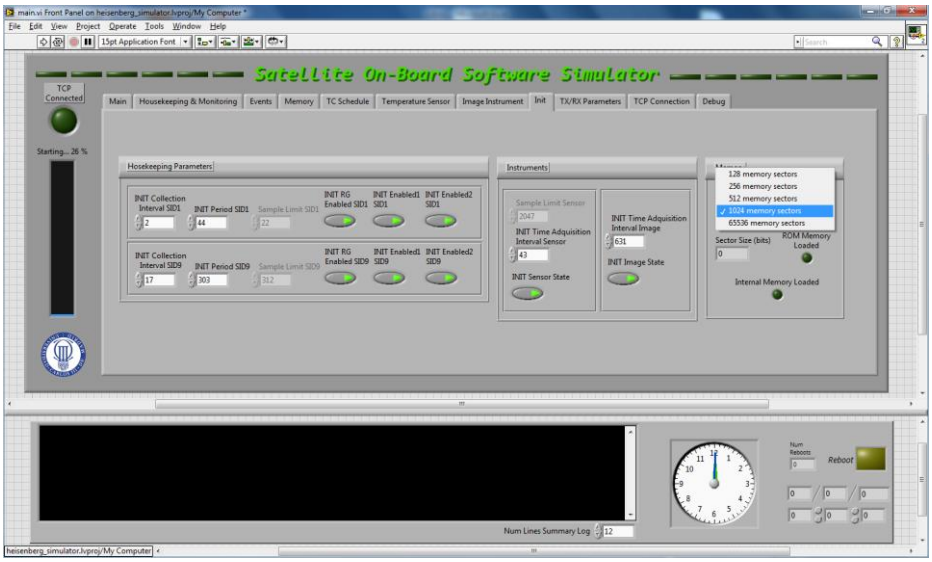


Fig. J.1. Inicialización de parámetros

Luego hay que configurar la conexión con los *USRP* y las frecuencias de conexión con las estaciones base. Se recomienda emplear la frecuencia por defecto y no modificar el resto de valores ni la modulación.

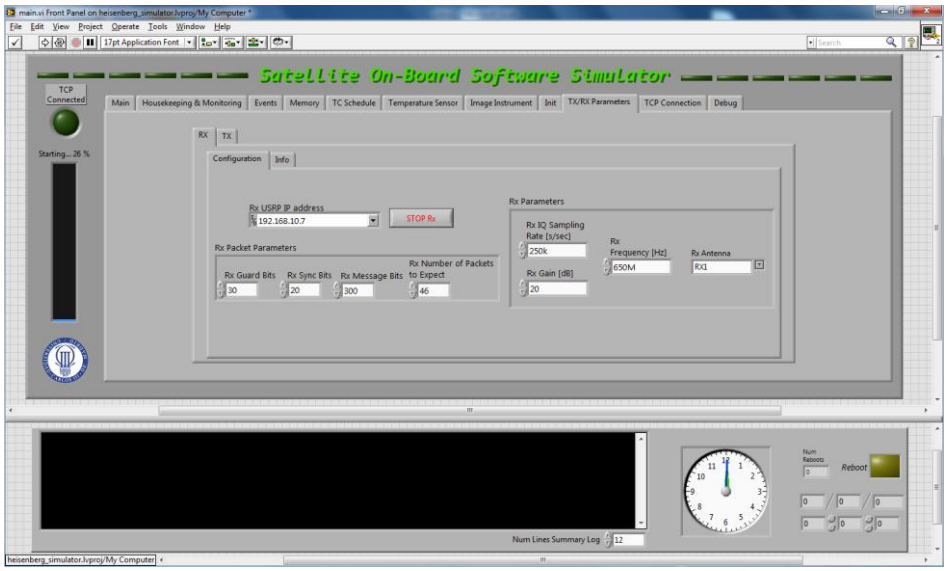


Fig. J.2. Configuración de la recepción radio

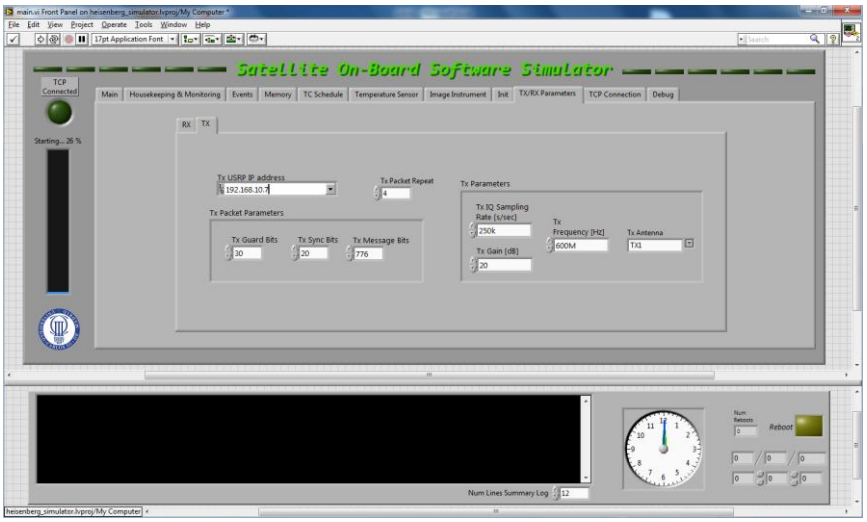


Fig. J.3. Configuración de la transmisión radio

Para obtener la IP del *USRP* en caso de no saberla se puede mirar en *Inicio >> National Instruments >> NI-USRP >> NI-USRP Configuration Utility*. De este modo se puede saber si está conectado el aparato.

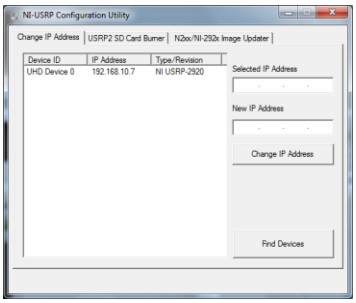


Fig. J.4. NI-USRP Configuration Utility

En este caso no se va conectar con el dispositivo móvil, luego se mantiene sin pulsar el botón *INIT Server* en la pestaña *TCP Connection*. Luego se inicia el programa pulsando *Run* en la parte superior.

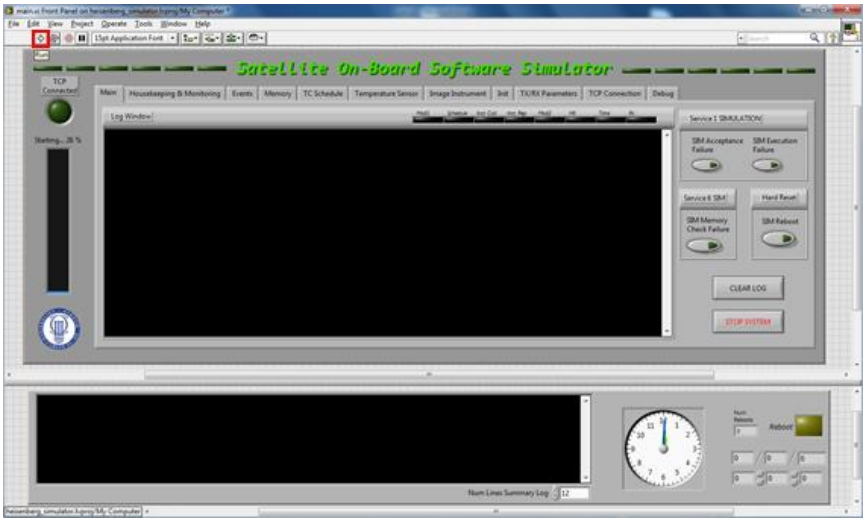


Fig. J.5. Inicio del programa

Una vez iniciado el programa sale una pantalla emergente con una barra de progreso y se muestran una serie de elementos visuales de carácter estético.

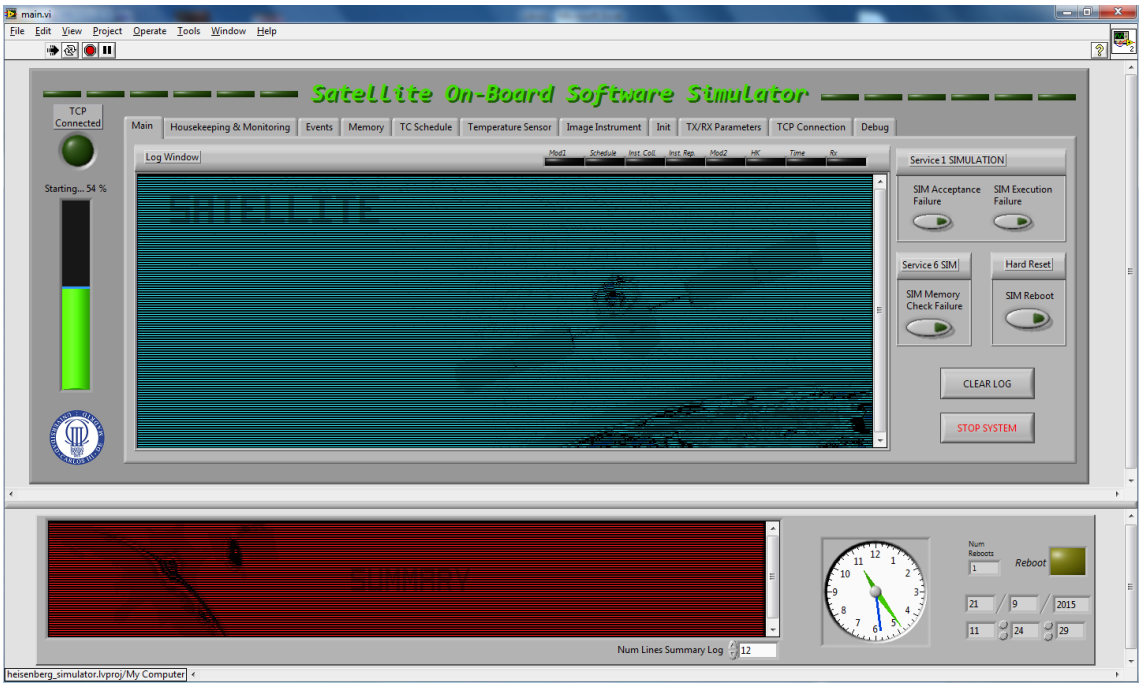


Fig. J.6. Arranque del simulador

El programa se considera iniciado cuando se visualiza la siguiente imagen:

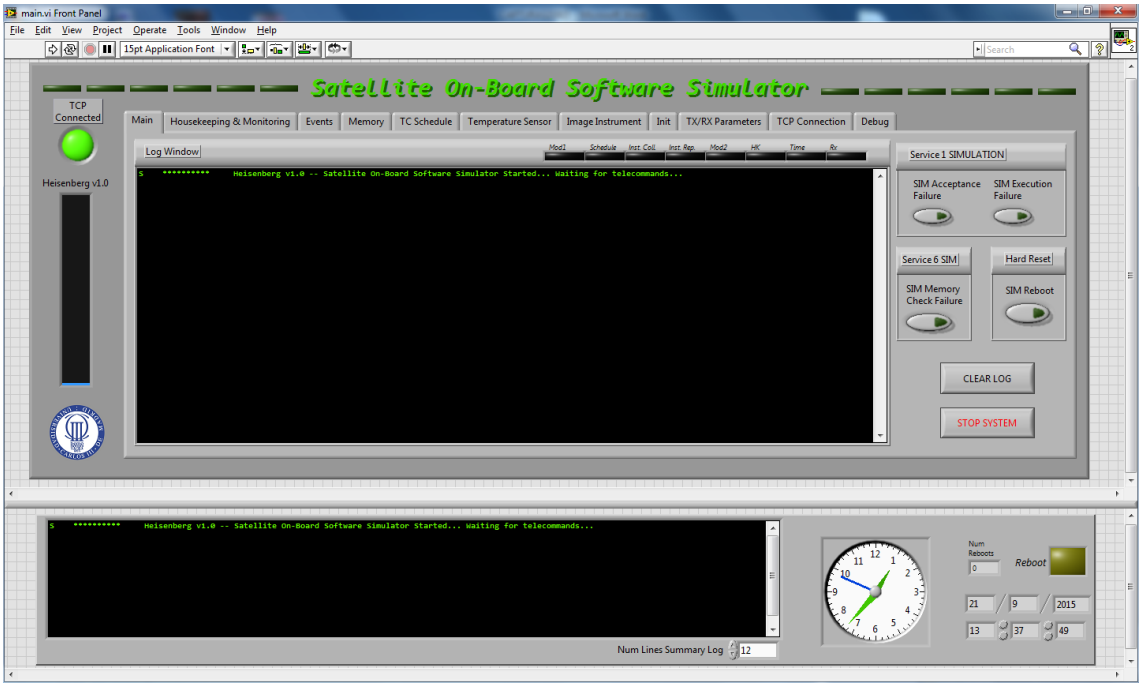


Fig. J.7. Programa iniciado

Ahora el programa se mantiene a la espera de telecomandos. Una buena práctica es mandar desde la estación base un test de conexión.

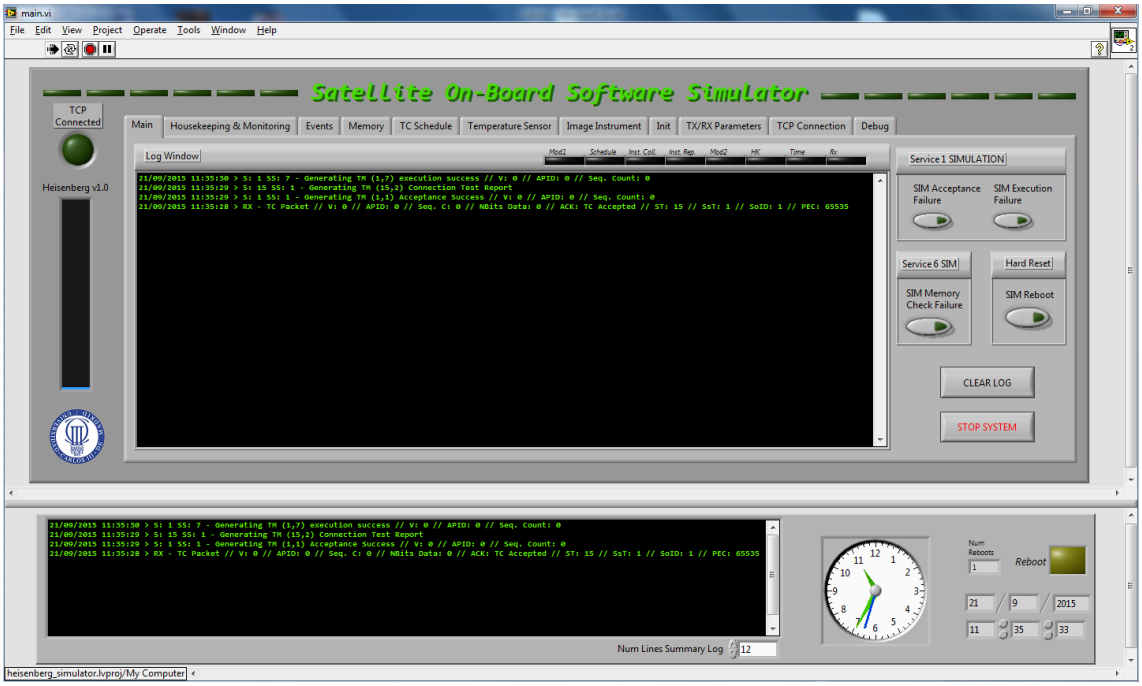


Fig. J.8. Recepción de test de conexión

Para saber si se está recibiendo o no se puede mirar si hay picos en la comparativa de señal-ruido en *TX/RX Parameters >> RX >> Info >> Debug*.

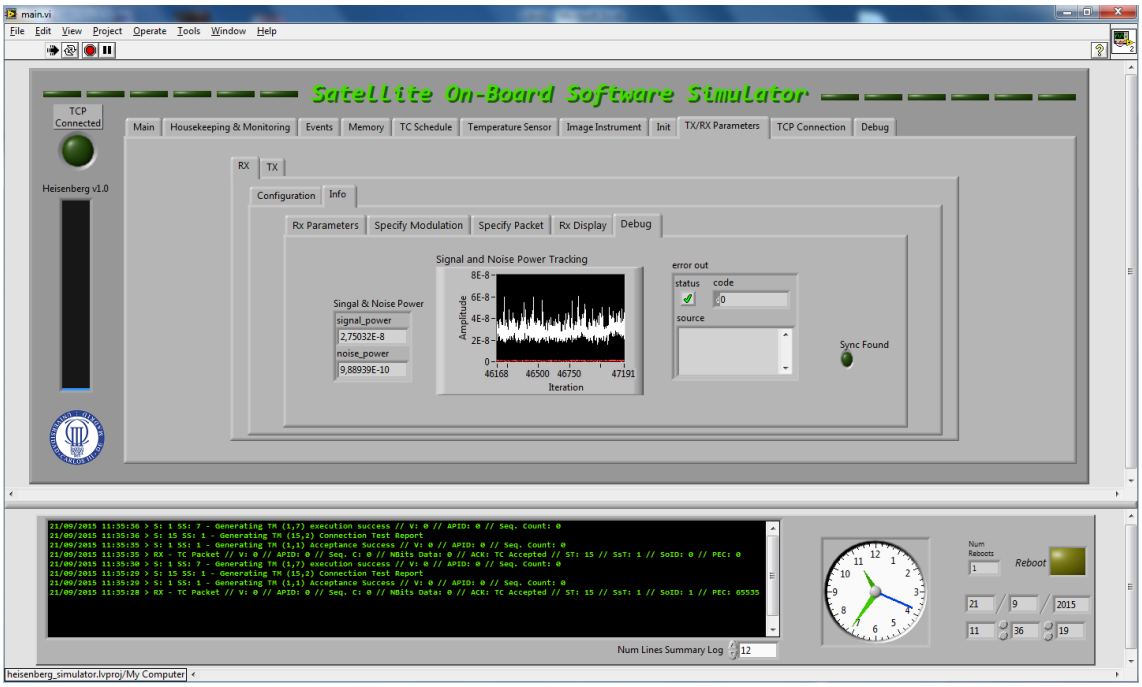


Fig. J.9. Recepción de ruido

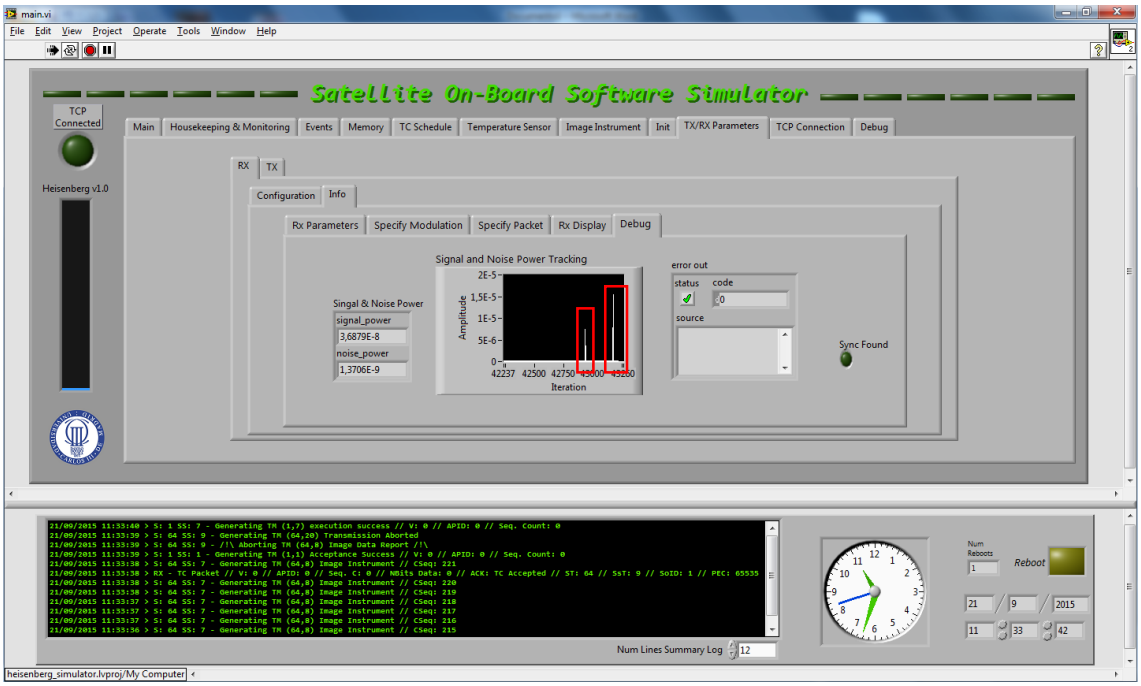


Fig. J.10. Recepción de paquetes

Para la simulación de errores de aceptación se pulsa el botón *SIM Acceptance Failure* y para el de ejecución *SIM Execution Failure*.

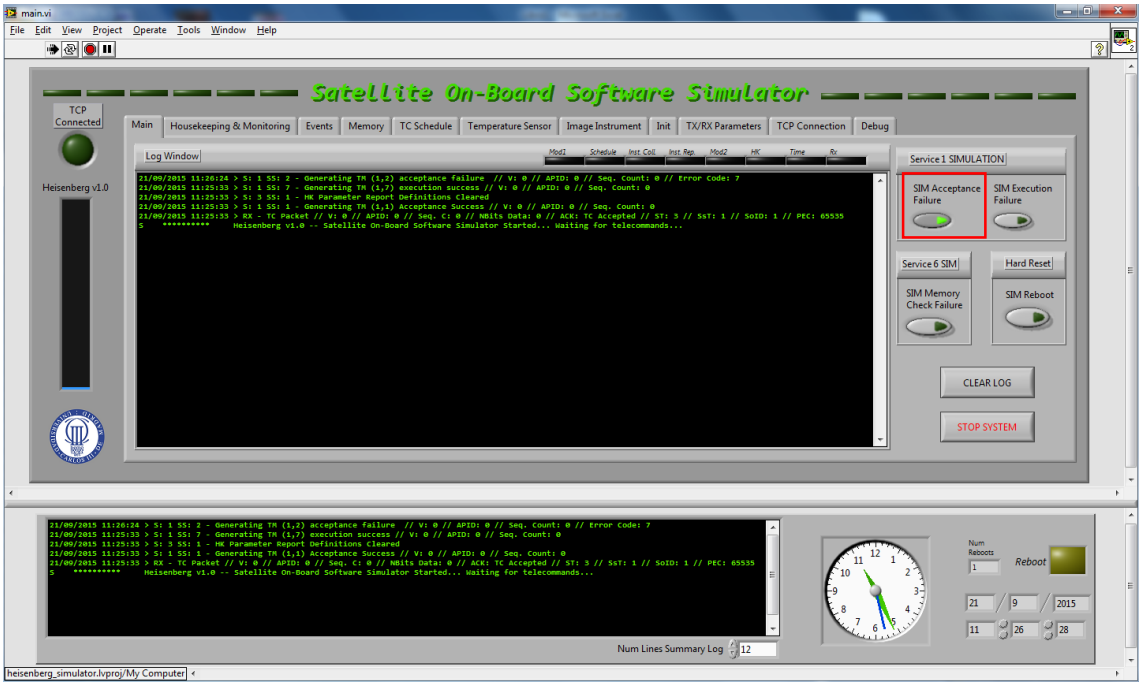


Fig. J.11. Simulación de error de aceptación

Mientras se mantengan pulsados no se eceptarán ni ejecutarán telecomandos recibidos desde la estación base. En el caso de la aceptación a veces se pueden generar reportes de aceptación fallida por un fallo en el formato recibido.

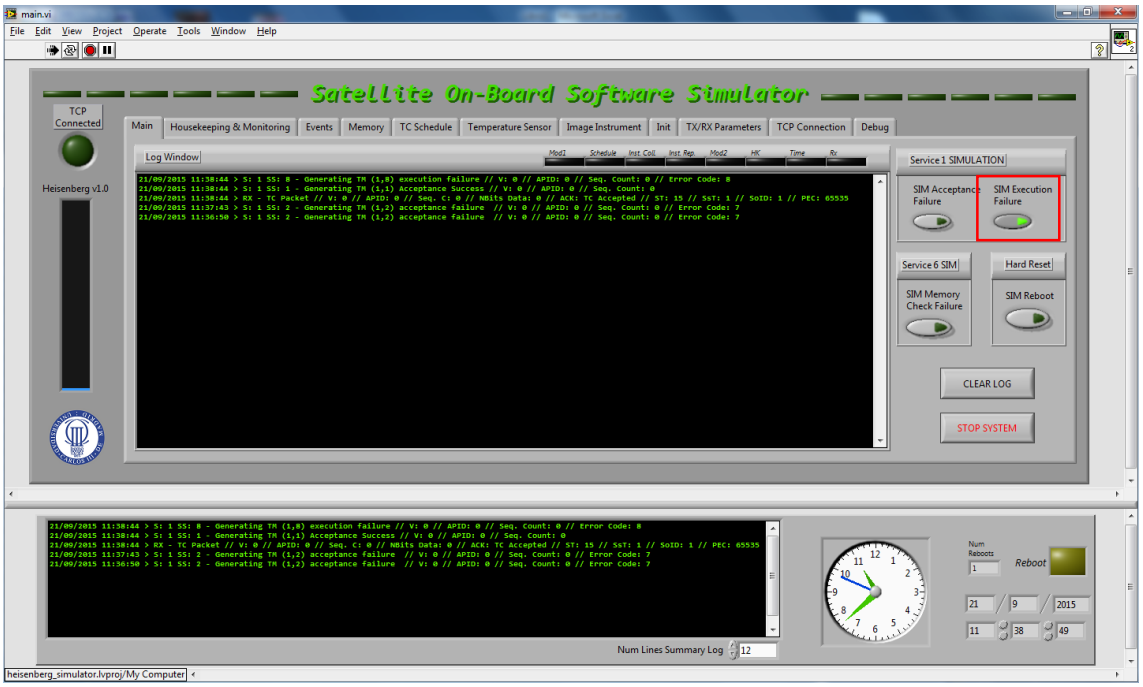


Fig. J.12. Simulación de error de ejecución

El estado inicial de la tabla de *housekeeping* es el mostrado en la siguiente figura en el que todos los campos están activados:

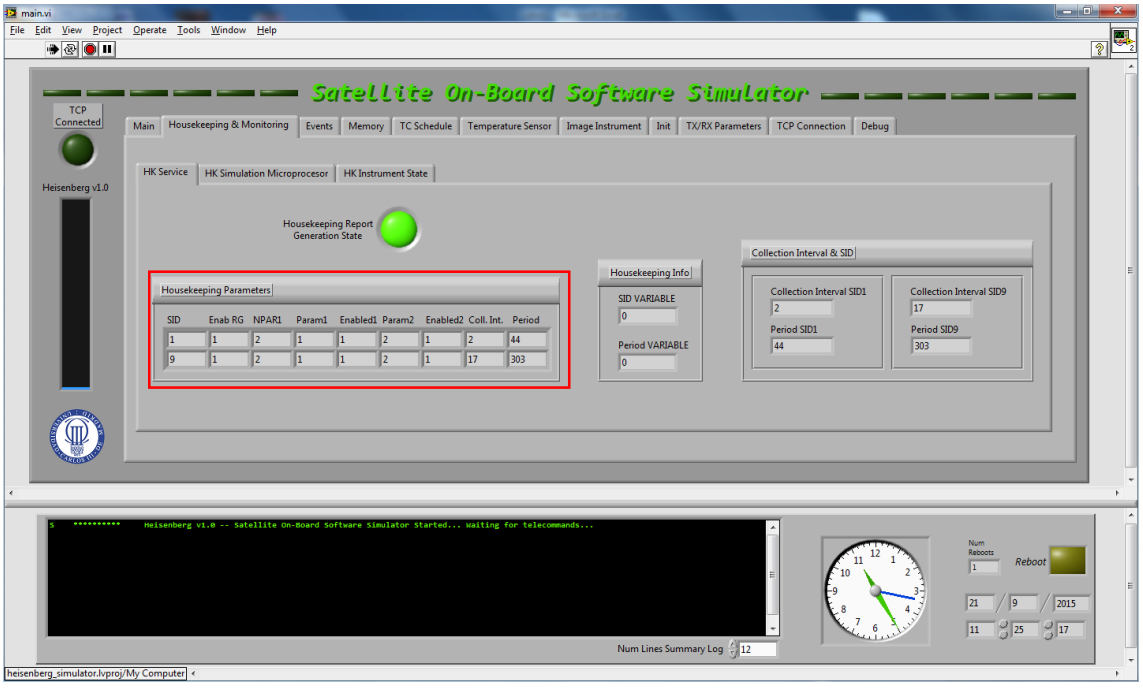


Fig. J.13. Estado inicial del módulo de housekeeping

Si se borra la tabla de *housekeeping* los *Servicios 3* y *12* quedan inutilizados por lo que se recomienda no emplearlo hasta haber finalizado con la ejecución del módulo. Si se quiere reiniciar hay que hacer uso del *Reboot*.

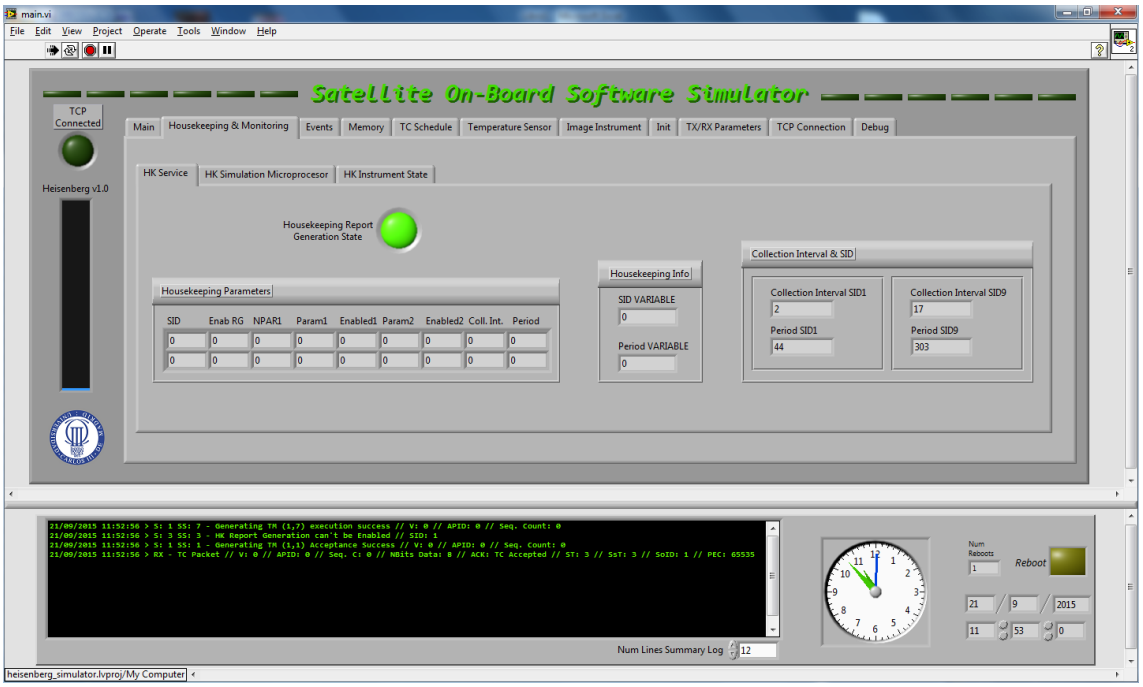


Fig. J.14. Borrado de la tabla de Housekeeping

Si se quiere parar el reporte de algún *SID* en la tabla se pone un cero:

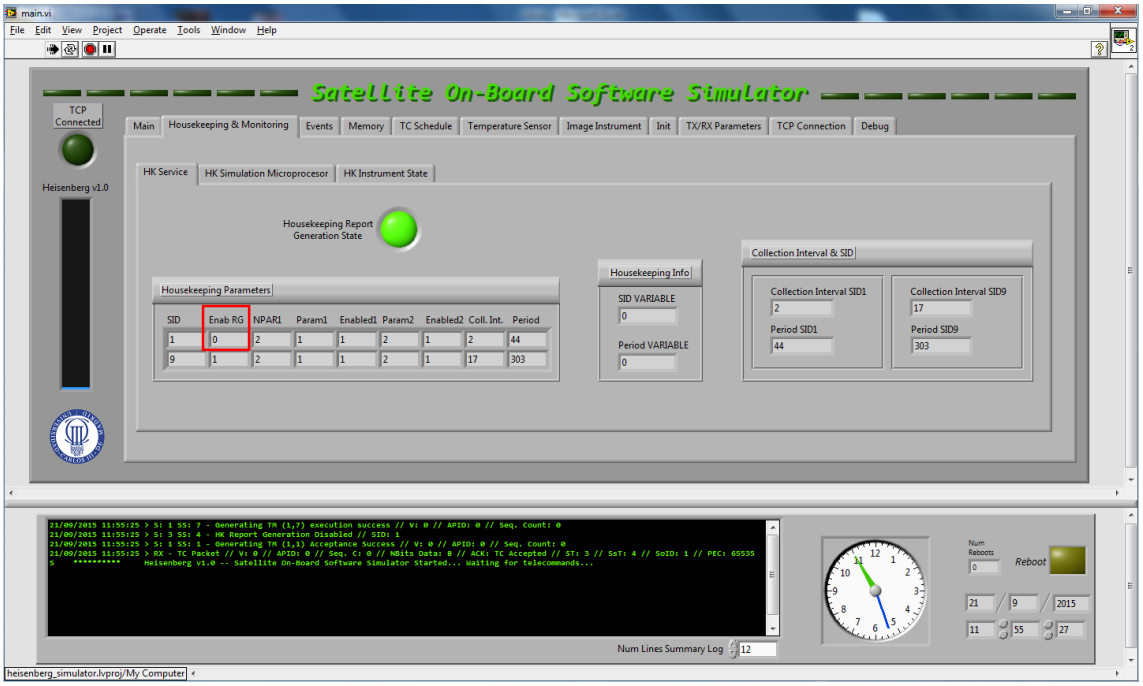


Fig. J.15. Deshabilitación de reporte de SID 1 (microprocesador)

También se puede ver la generación de datos de *housekeeping* y solicitar su reporte.

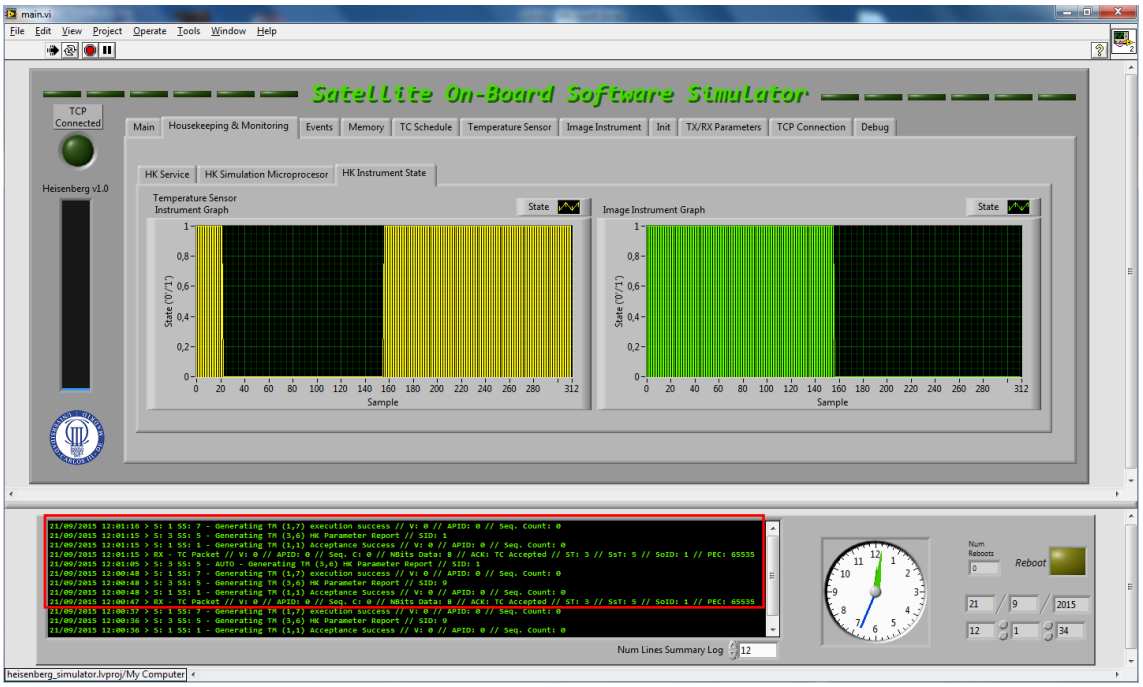


Fig. J.16. Parámetros de SID 9 (estado de instrumentos) y su reporte

Fig. J.17. Parámetros de SID 1 (microprocesador)

También se generan reportes automáticos como se muestra en la siguiente figura con el envío de los datos de *housekeeping* con *SID* 9.

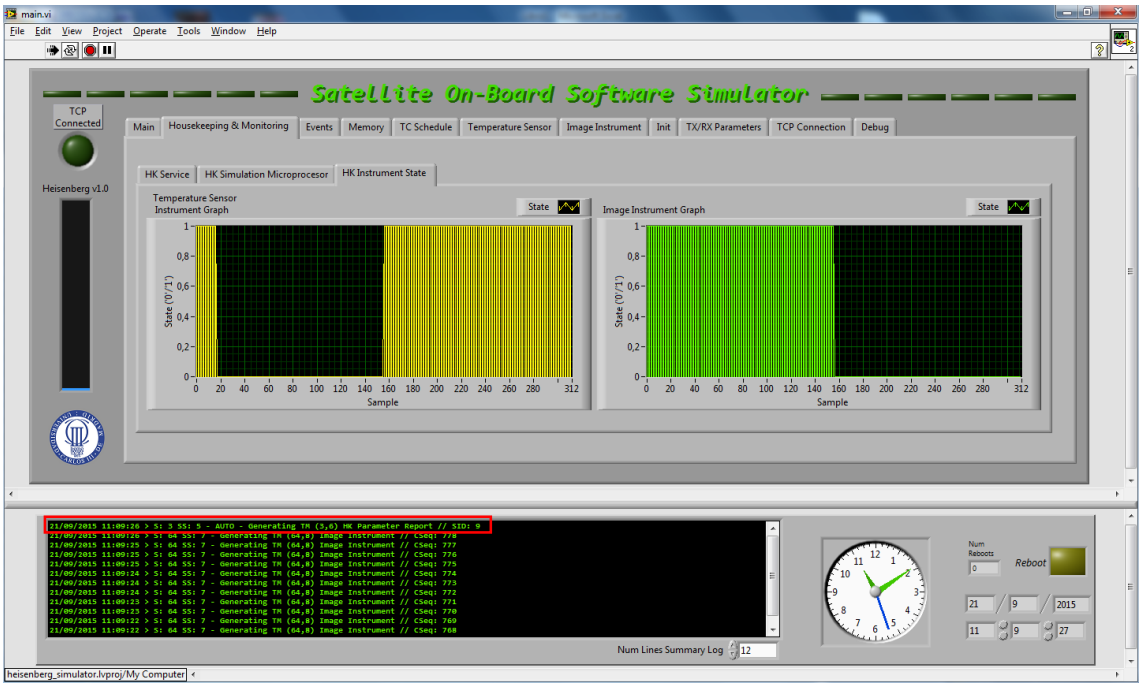


Fig. J.18. Envío automático de reporte SID 9

También se puede cambiar el período de reporte y mandar luego que se actualiza al valor temporal almacenado.

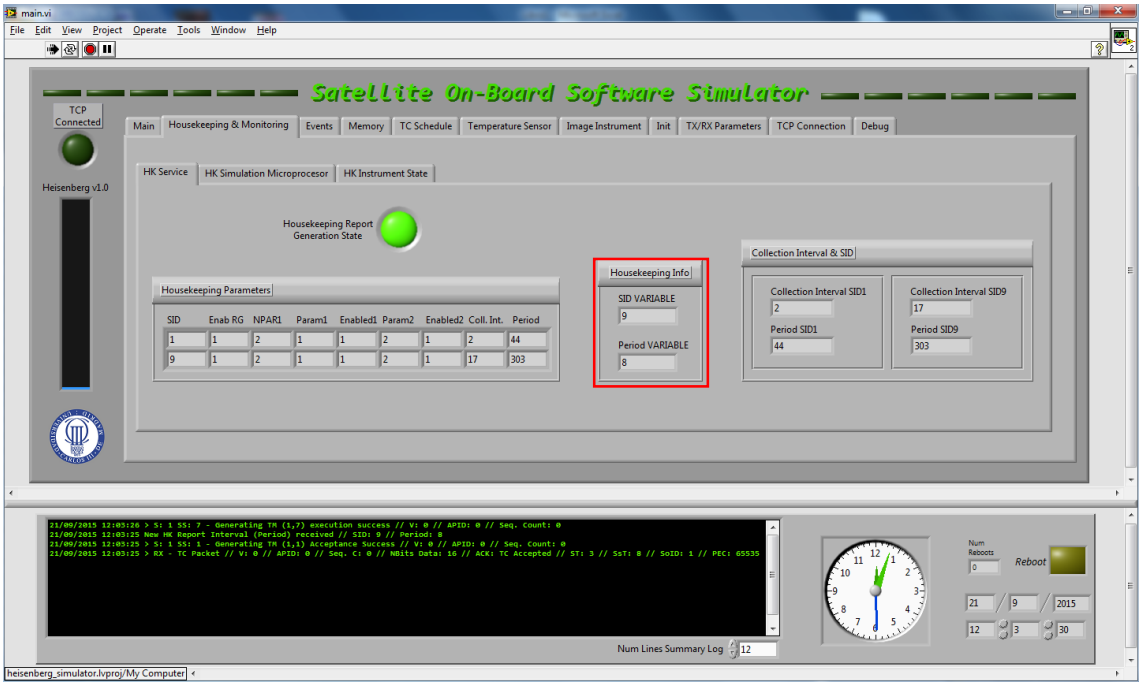


Fig. J.19. Recepción de un período de reporte para SID 9

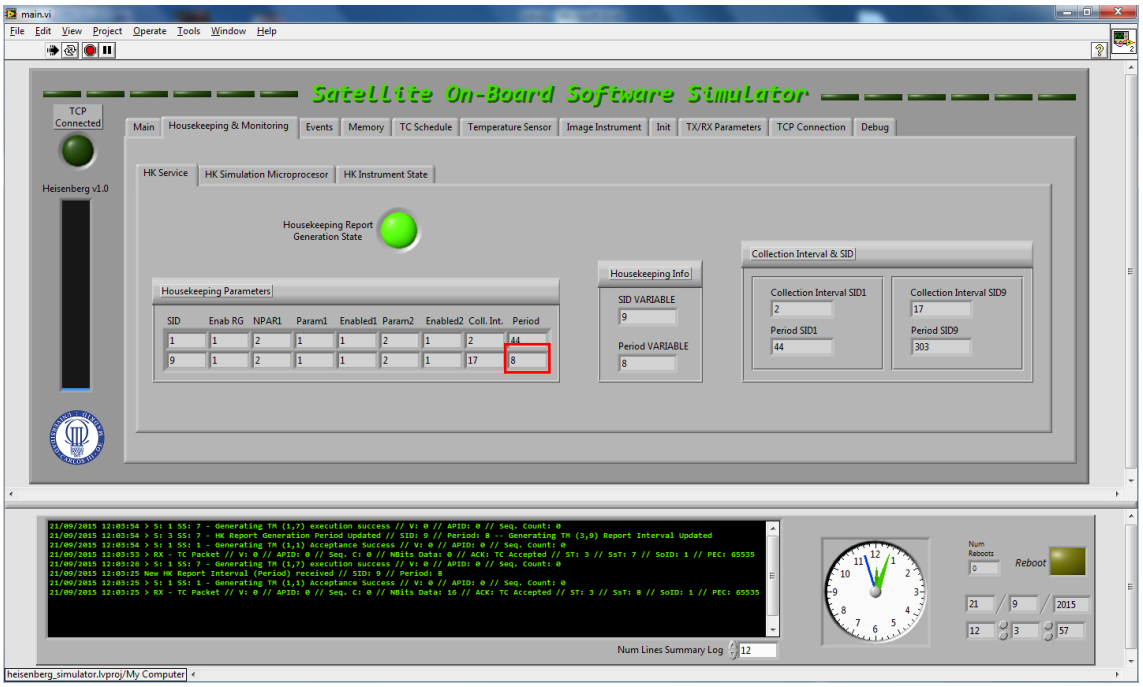


Fig. J.20. Actualización del período de reporte SID 9

También se puede activar y desactivar la monitorización de los datos poniendo un cero o un uno en la tabla.

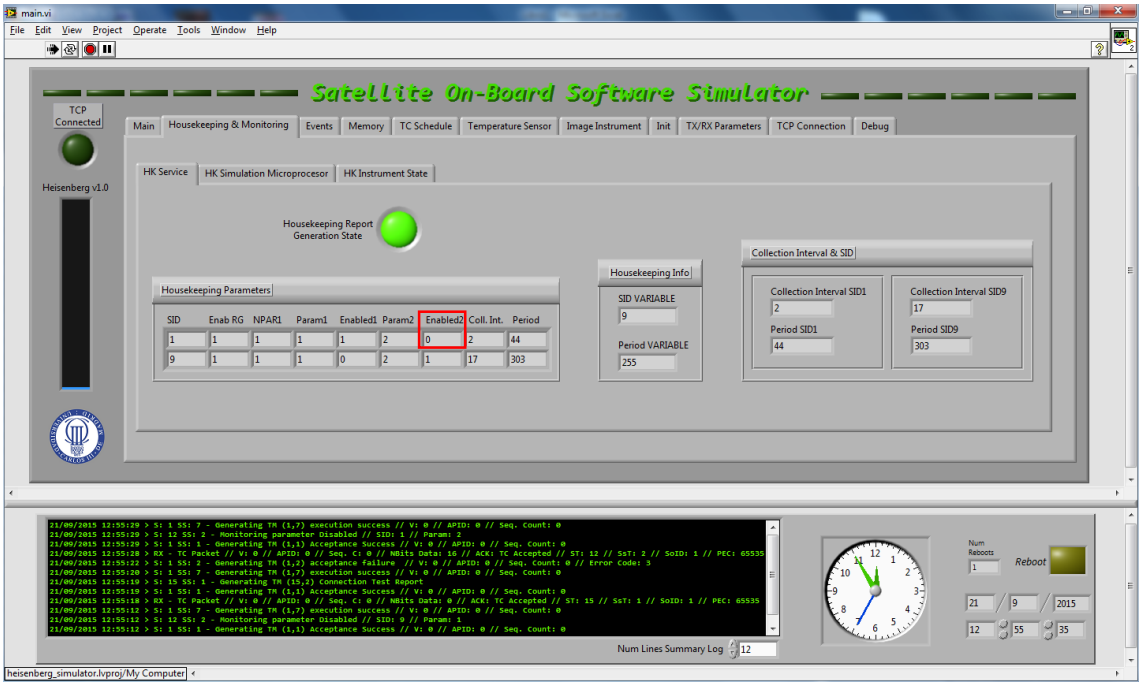


Fig. J.21. Desactivación de la monitorización Param 2 SID 1 (frecuencia del microprocesador)

Otra posibilidad es borrar todas las definiciones de monitorización, añadirlas o eliminarlas una a una.

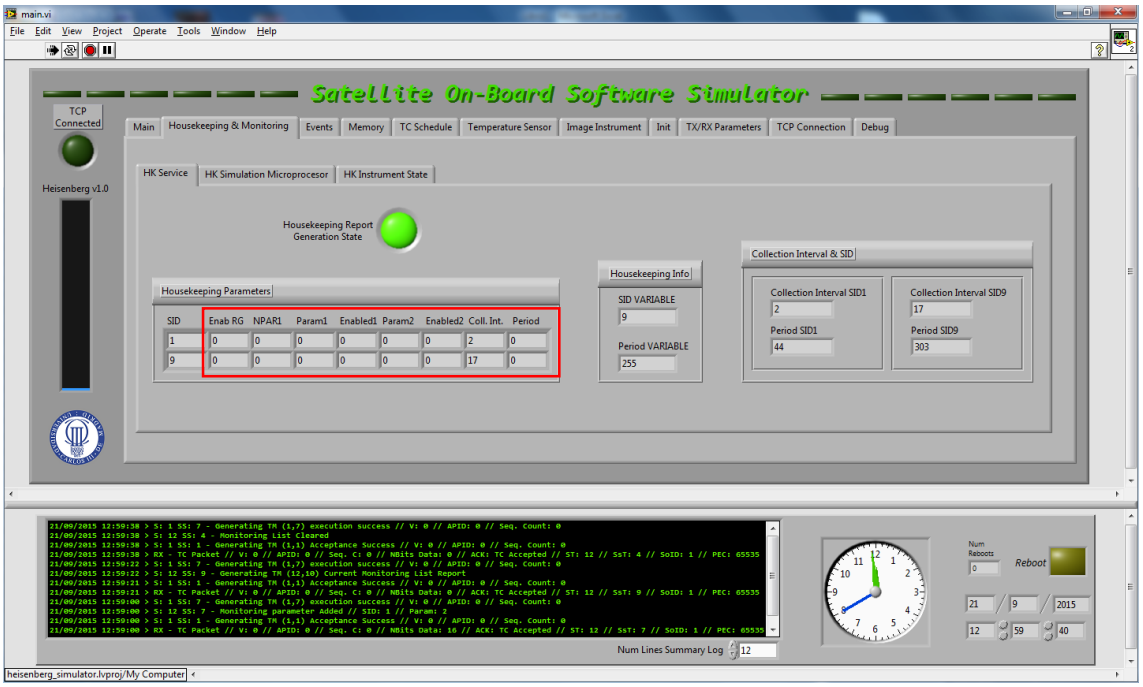


Fig. J.22. Borrado de todas las definiciones de monitorización

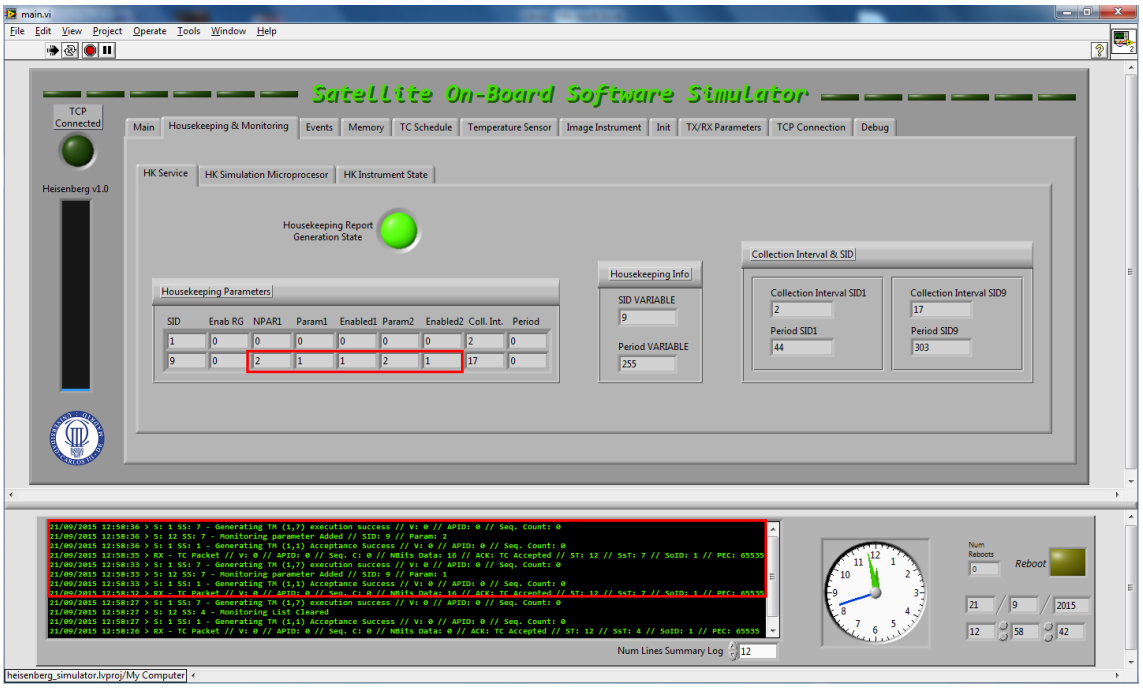


Fig. J.23. Inserción de Param 1 y Param 2 en SID 9

En cuanto al servicio de eventos se pueden simular eventos de cuatro niveles de gravedad. Se puede ir viendo en el *log* de eventos todas las operaciones realizadas de este servicio en concreto. Se enciende el LED de un color y suena un sonido en cada caso.

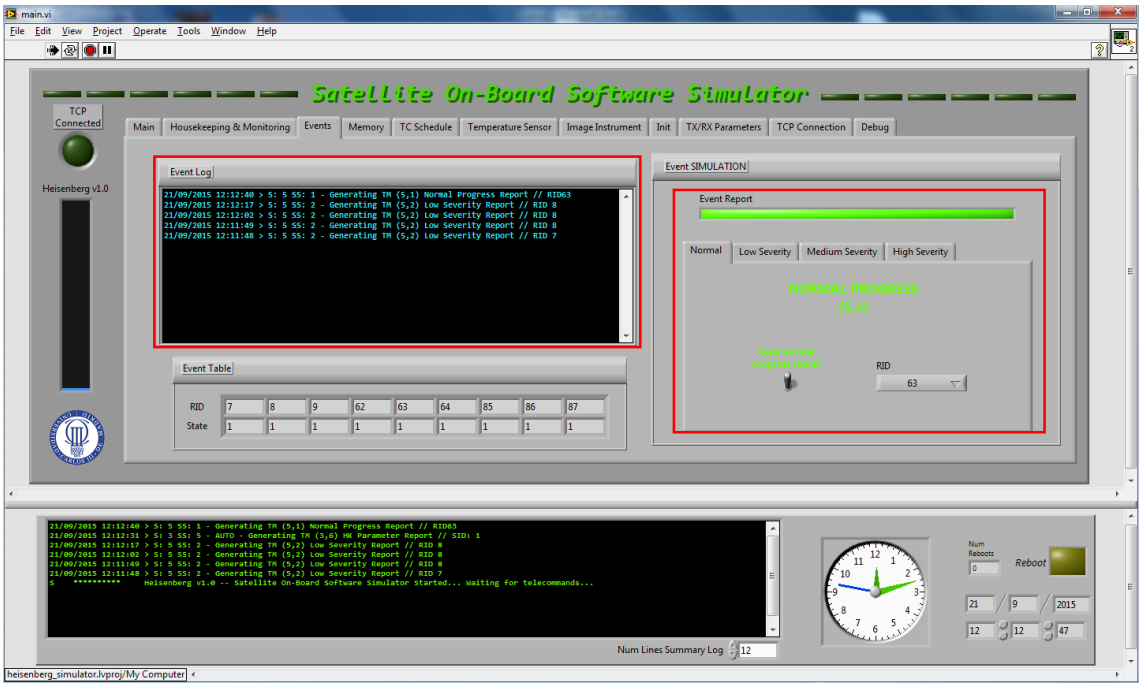


Fig. J.24. Reporte de funcionamiento normal

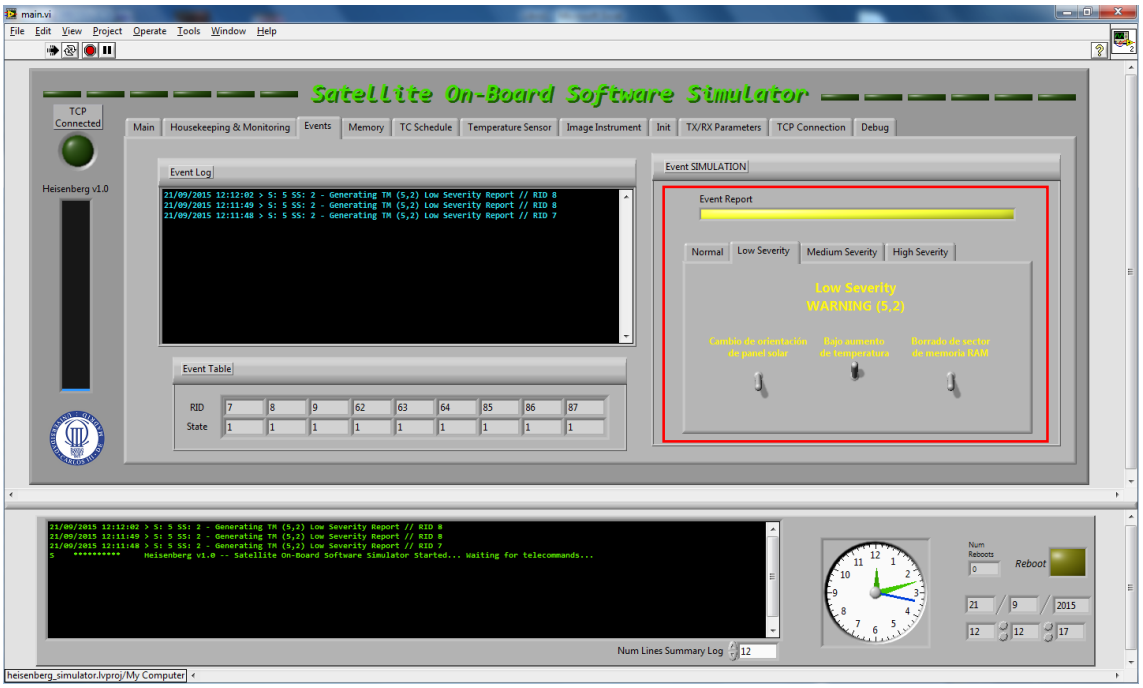


Fig. J.25. Reporte de evento de baja gravedad

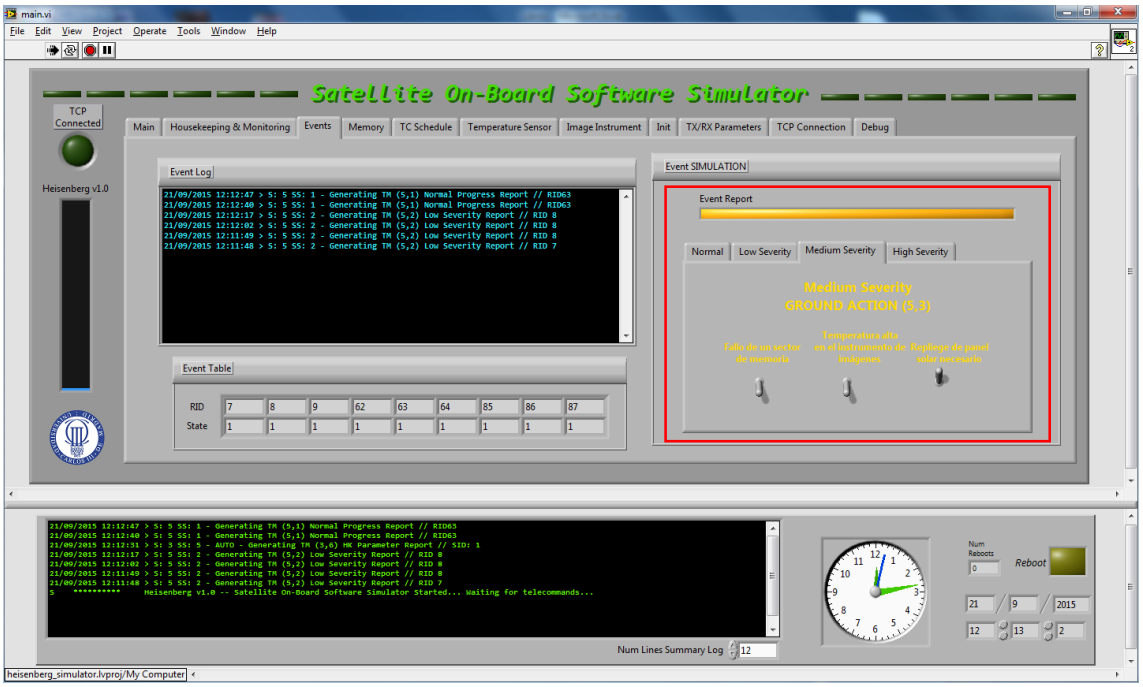


Fig. J.26. Reporte de evento de gravedad media

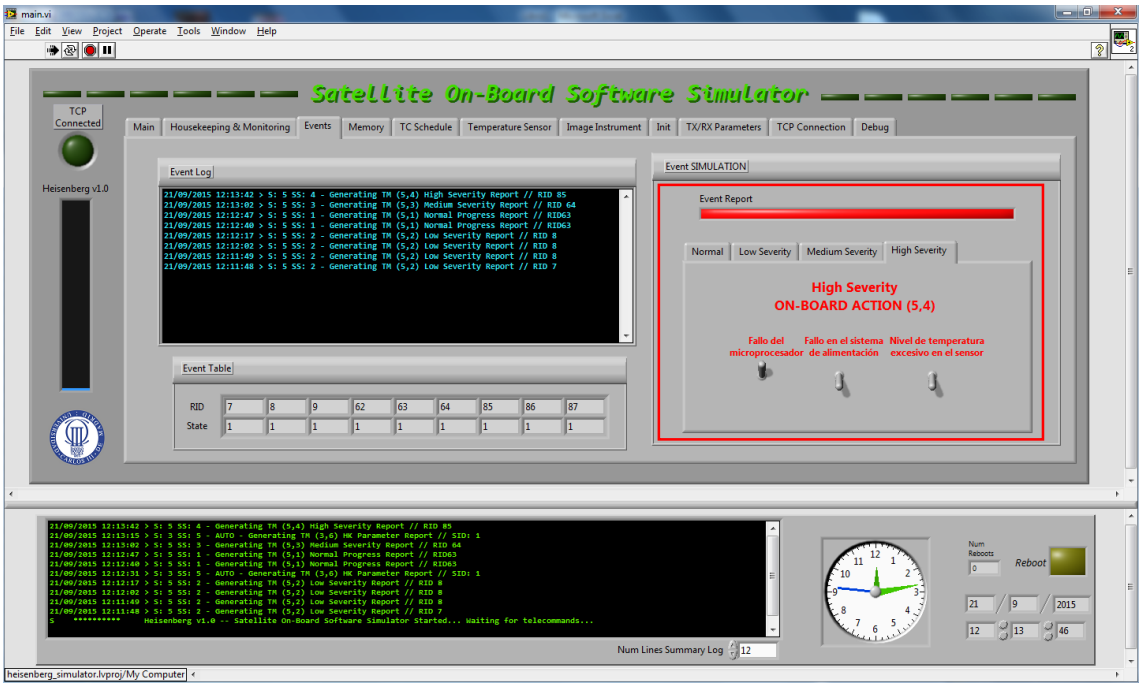


Fig. J.27. Reporte de evento de alta gravedad

Otras funciones son desactivar o activar la generación de reportes de evento para un *RID* en concreto, el reporte de los activos y no activos, y el borrado del *log* de eventos.

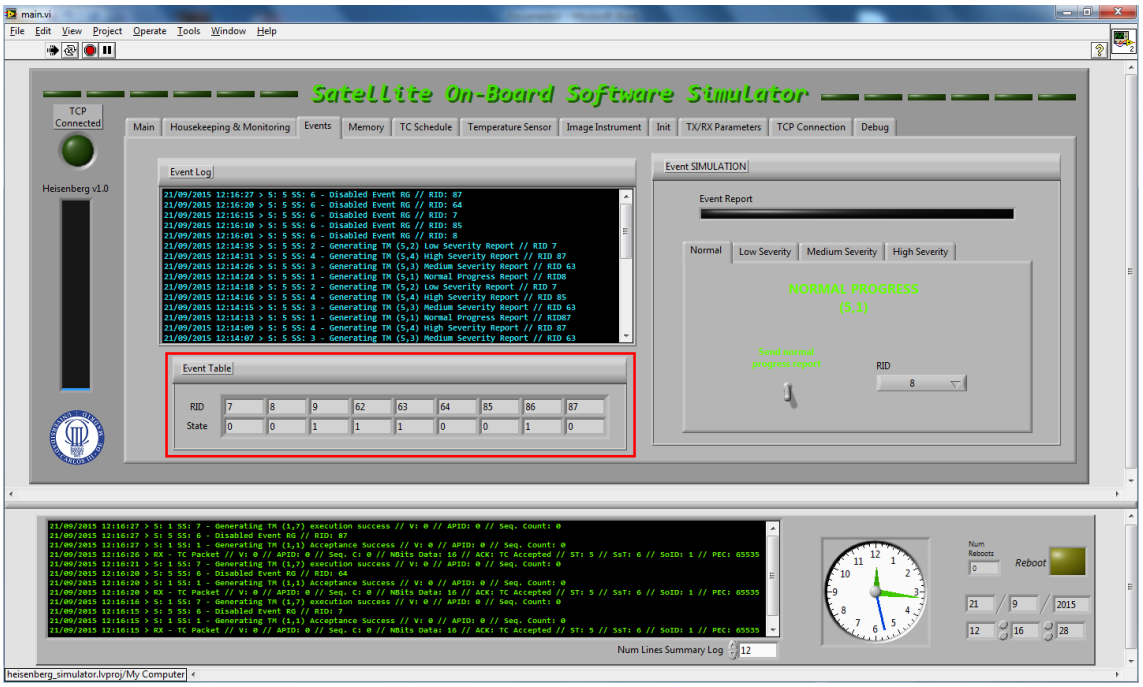


Fig. J.28. Desactivación de varios RID

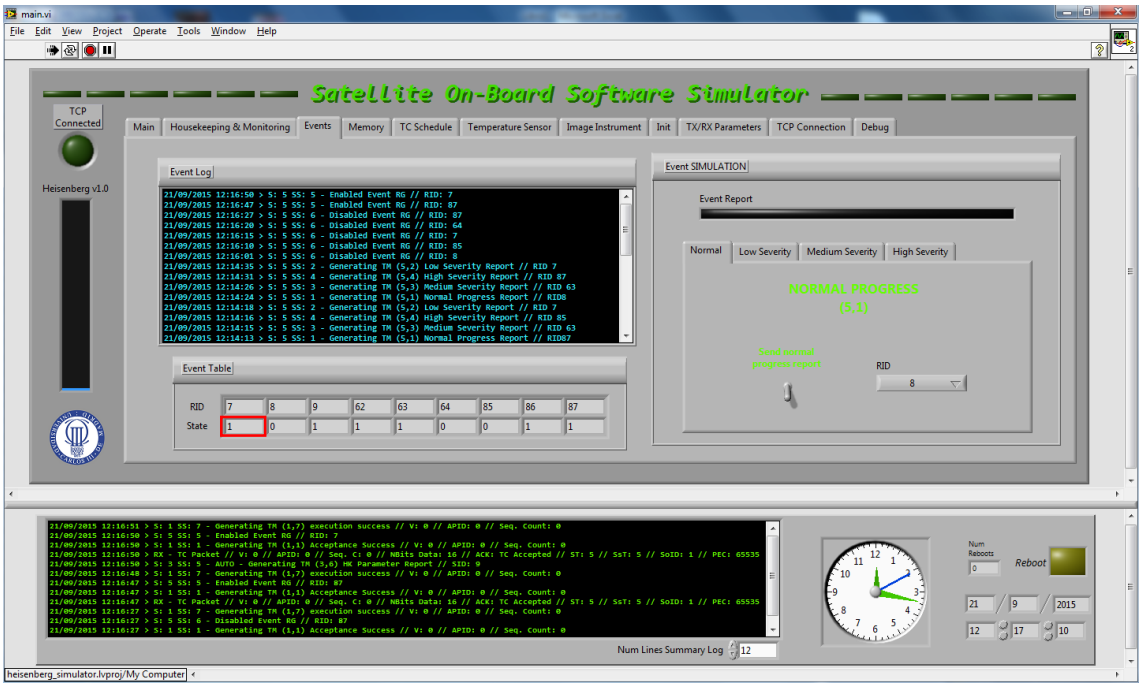


Fig. J.29. Activación del RID 7

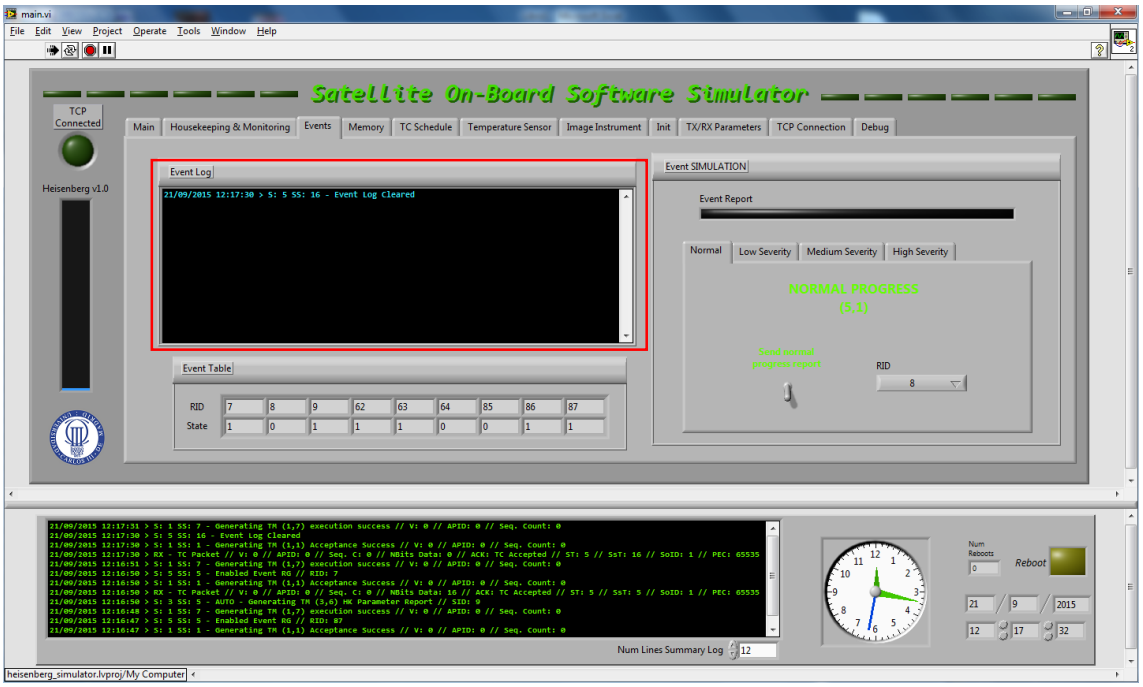


Fig. J.30. Borrado del log

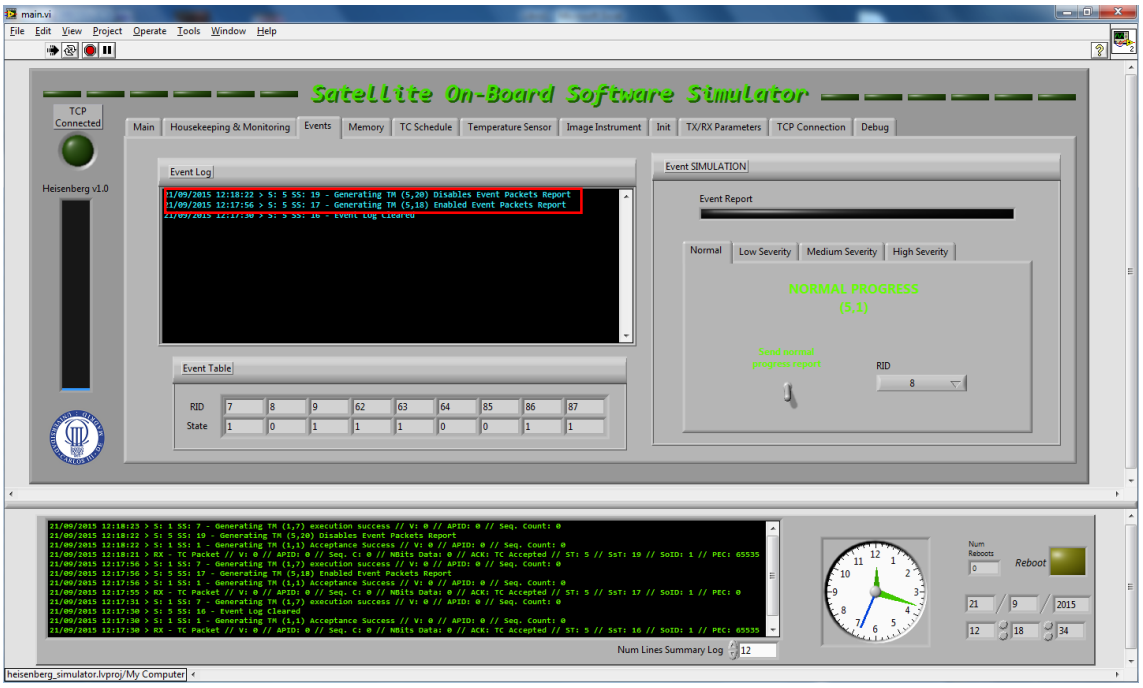


Fig. J.31. Reporte de los RID activos y no activos

En lo referente al servicio de gestión de la memoria, el estado inical es con la *memoria RAM* vacía. Se puede solicitar la carga de un sector de memoria interna en la *memoria RAM* y su reporte.

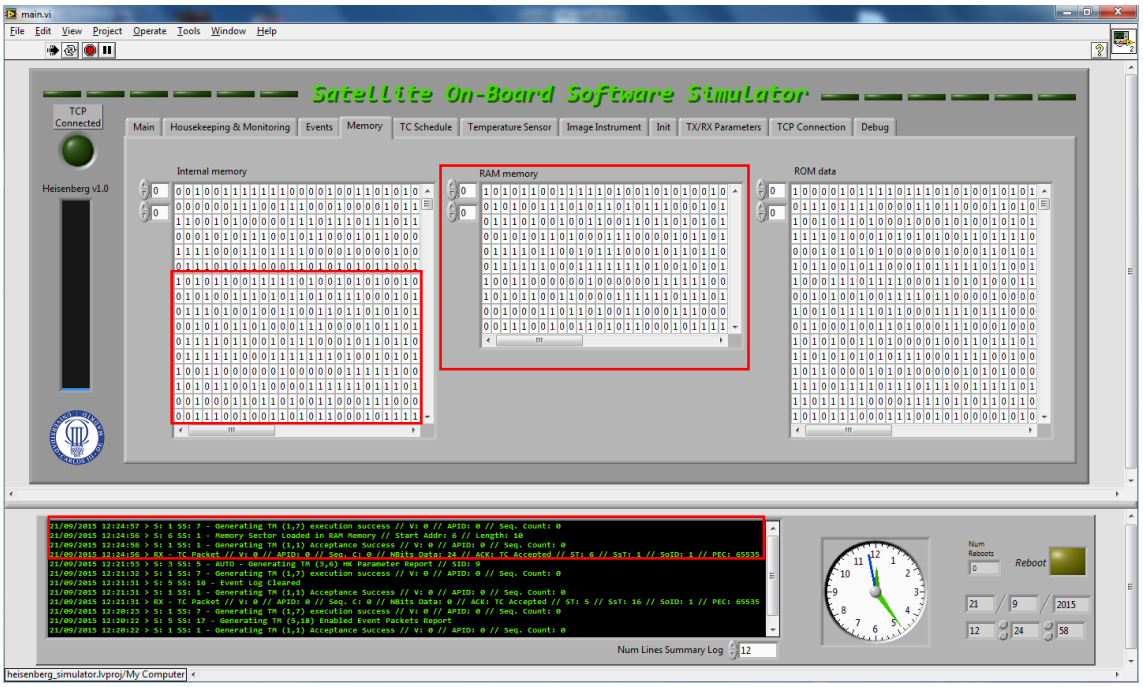


Fig. J.32. Carga en la RAM de sector de memoria interna desde la posición 6 con tamaño 10

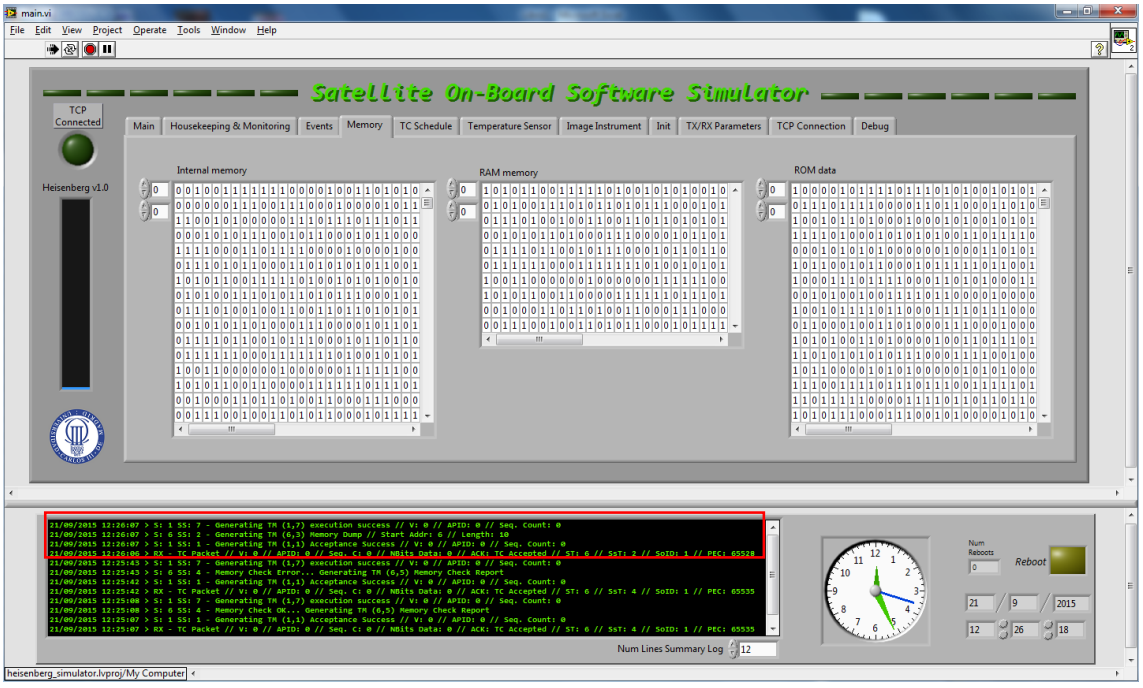


Fig. J.33. Envío de memoria RAM a la estación base

También se puede comprobar el estado de la memoria y en caso de querer simular fallo se pulsa el botón *SIM Memory Check* en la pestaña *Main*.

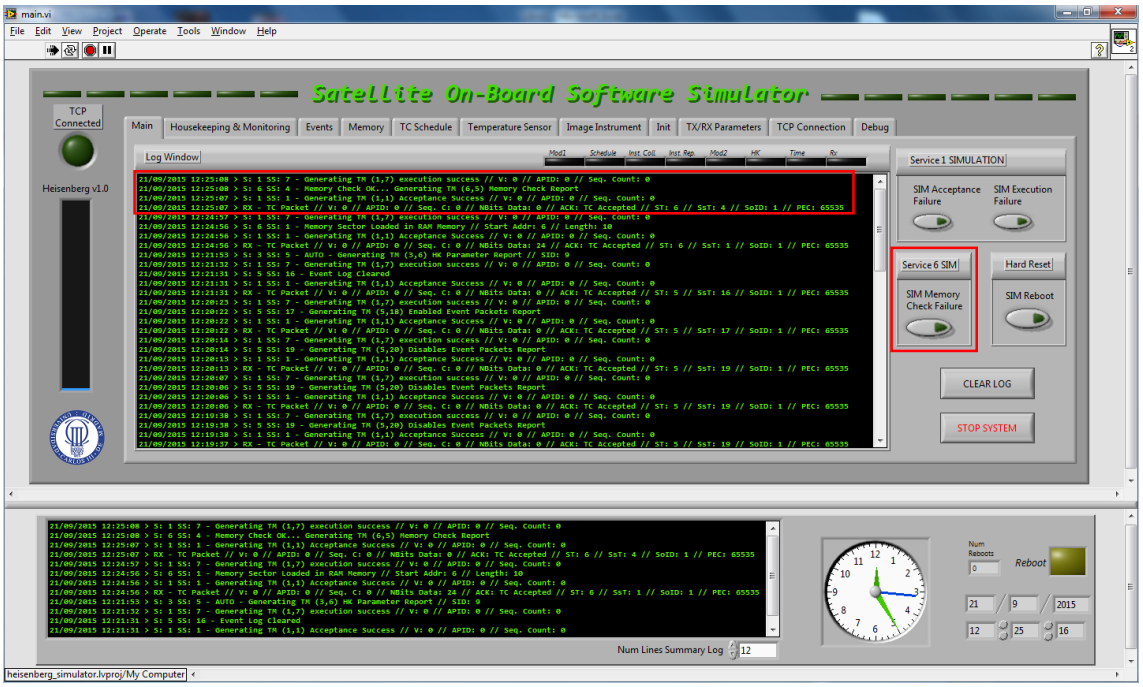


Fig. J.34. Reporte de estado de memoria correcta

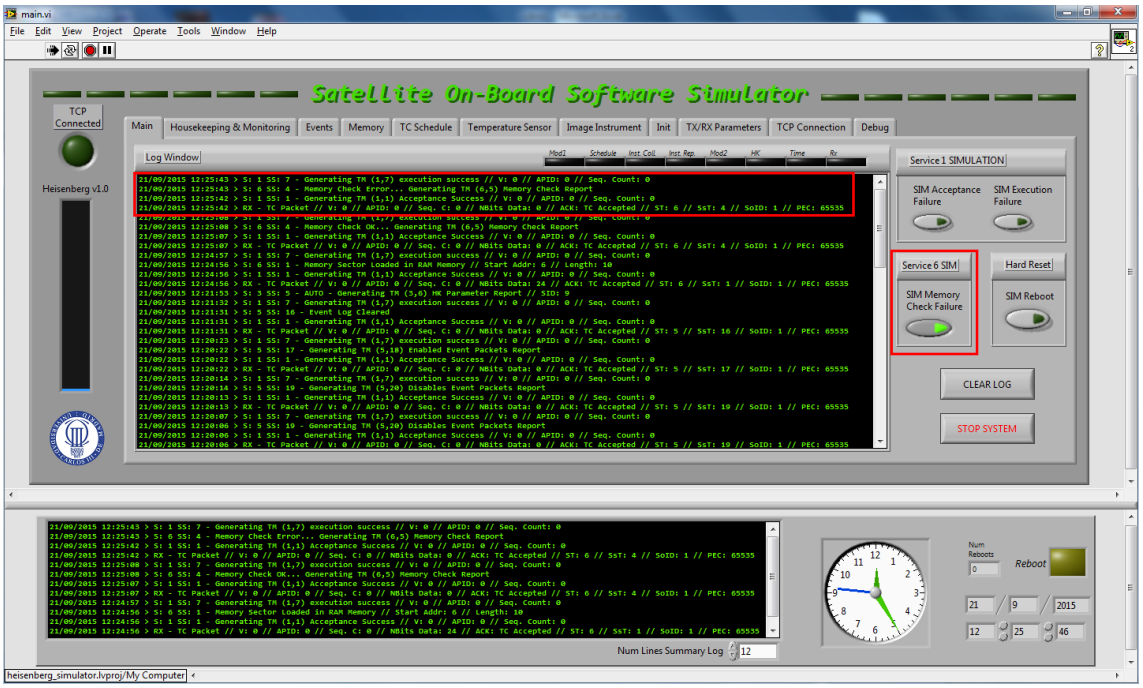


Fig. J.35. Simulación de error en la memoria

En último lugar para la memoria, también se puede cargar la memoria predefinida en la *memoria ROM* en la memoria interna.

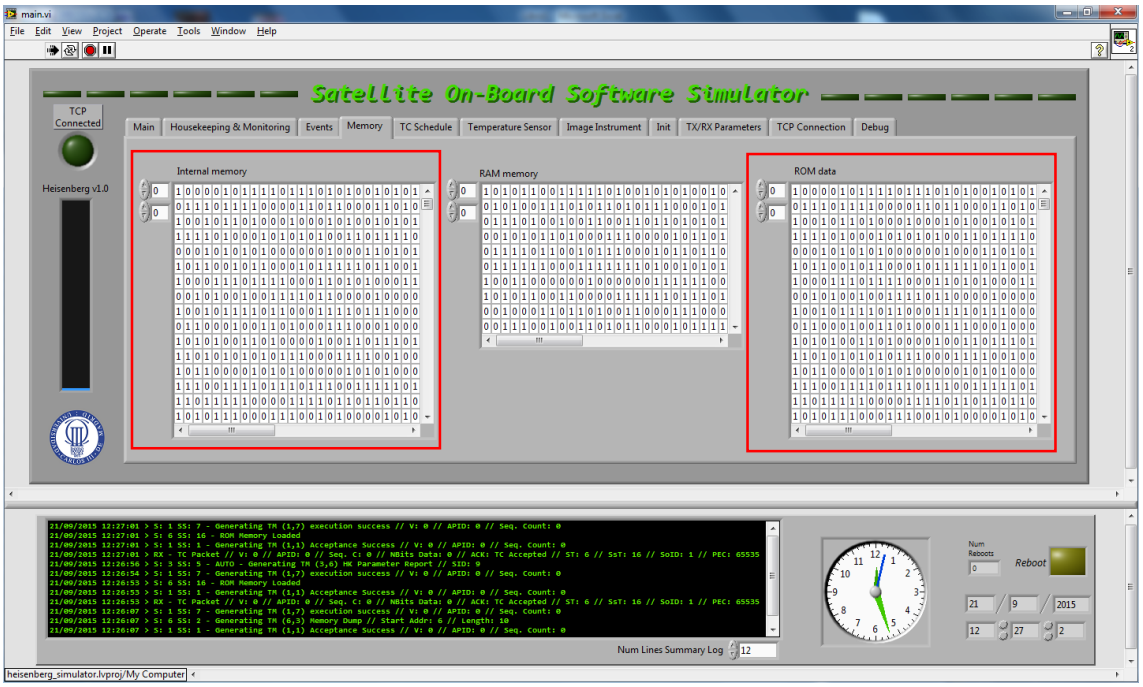


Fig. J.36. Carga de memoria ROM en memoria interna

Referente al sistema de de tiempos se puede reportar el tiempo del satélite y cambiarlo por uno desde la estación base.

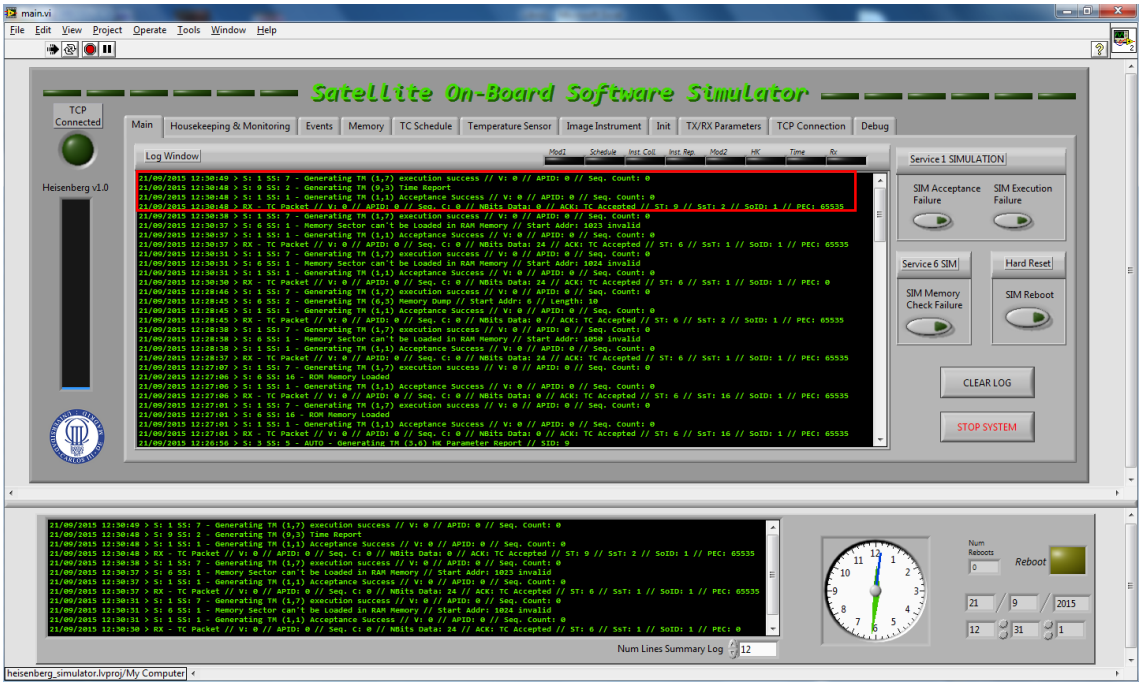


Fig. J.37. Reporte del sistema de tiempos del satélite

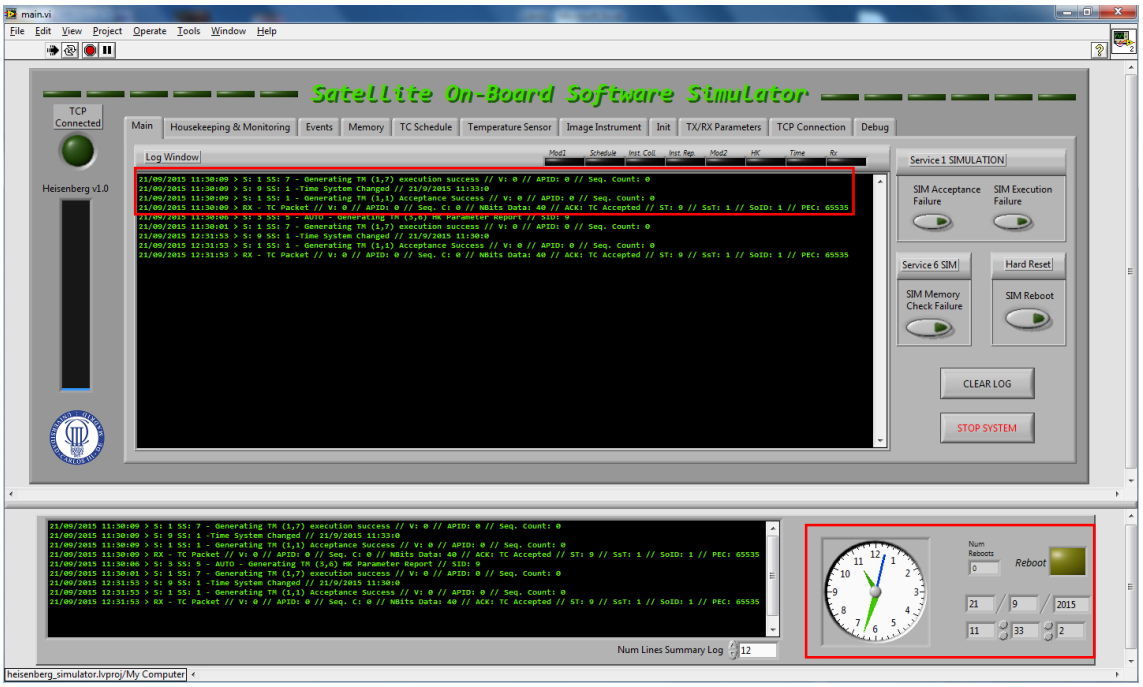


Fig. J.38. Cambio del sistema temporal del satélite (retraso de una hora)

Referente al sistema de planificación se puede activar o desactivar la ejecución de telecomandos de la lista cambiando el estado. Además, se puede ver un *log* con el registro de todas las operaciones de este módulo en concreto.

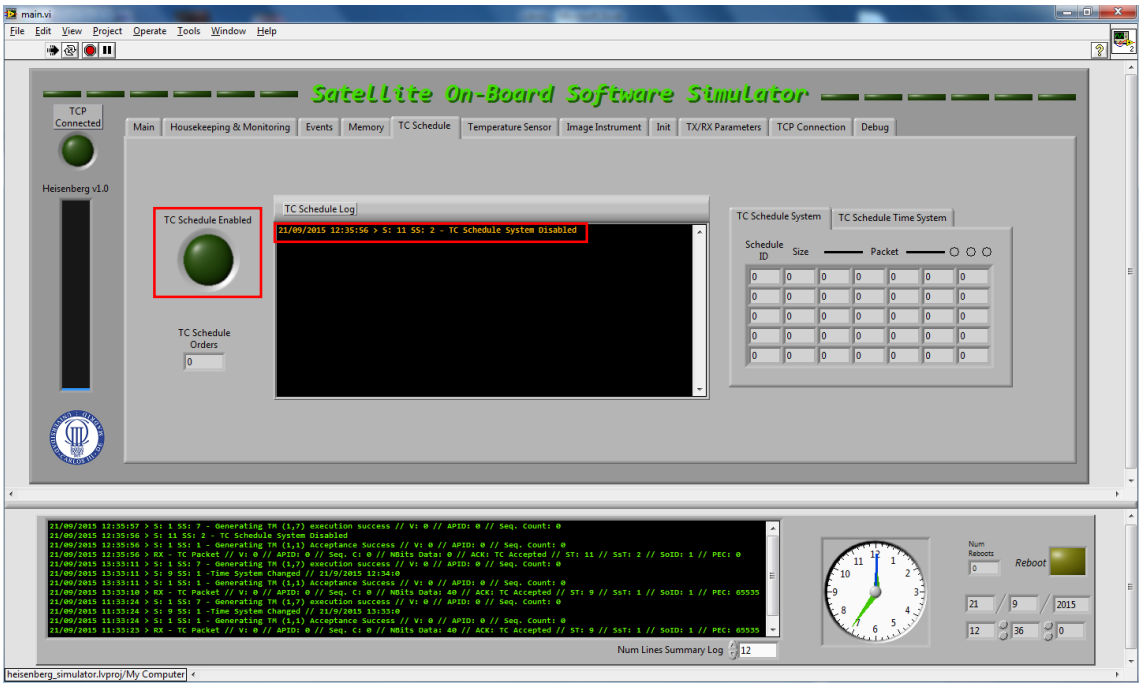


Fig. J.39. Desactivación de la ejecución de TC

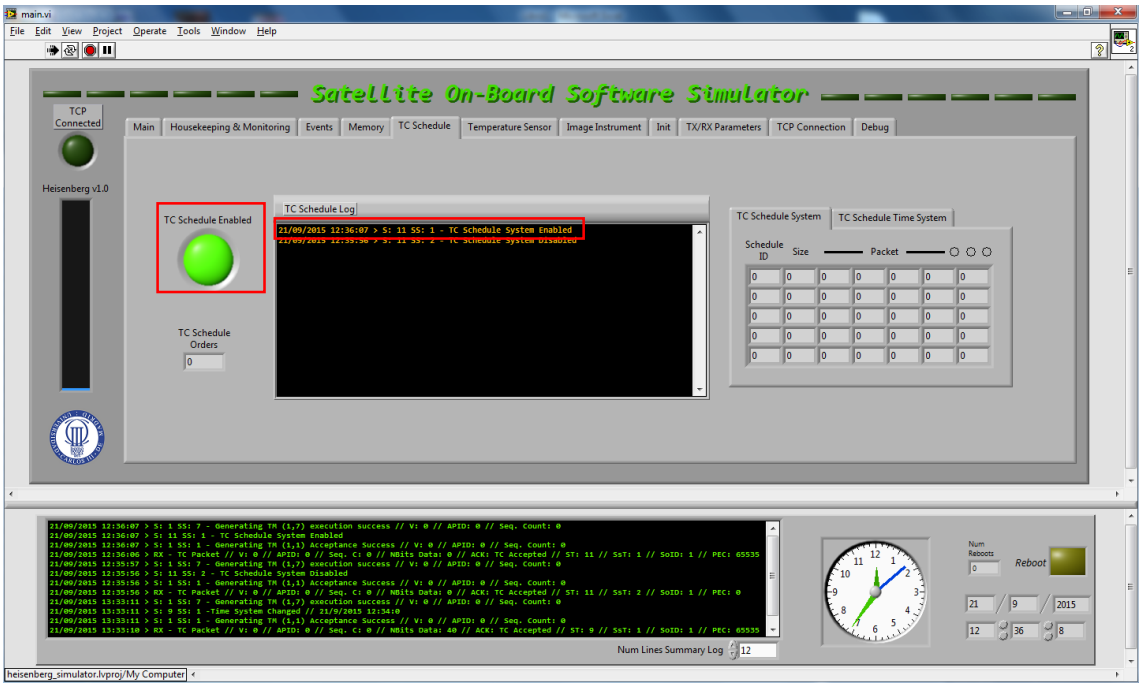


Fig. J.40. Activación de la ejecución de TC

También se pueden añadir TC a la lista y borrarlos. Cada TC tiene un tiempo de ejecución asociado.

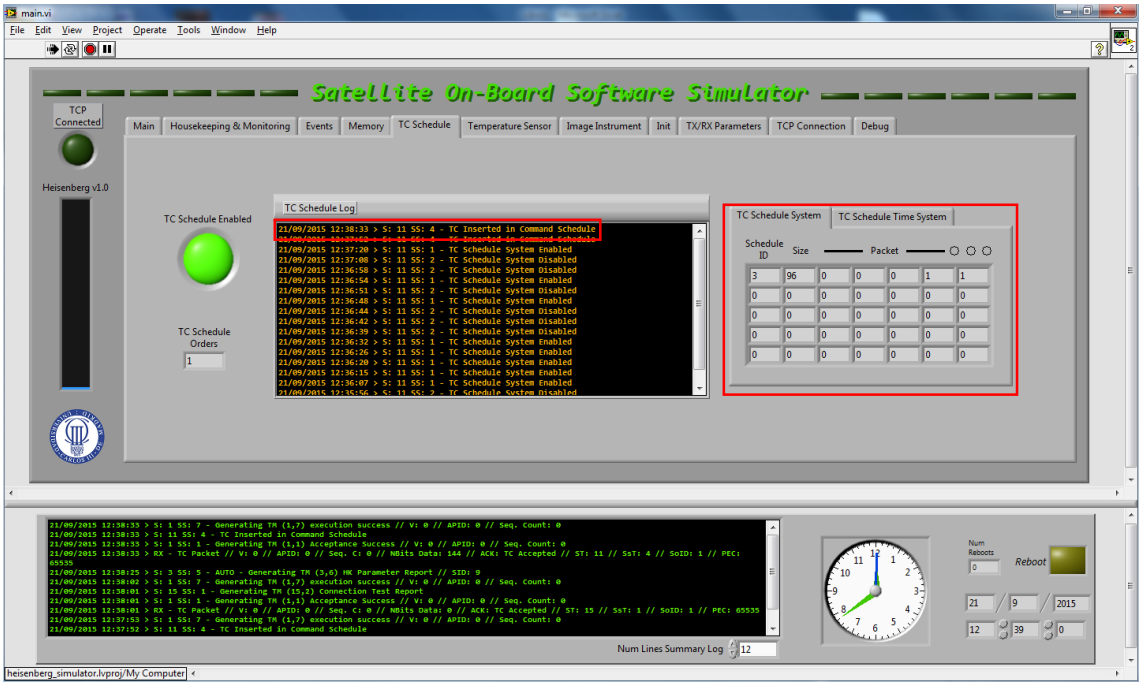


Fig. J.41. Inserción de TC en el sistema de planificación

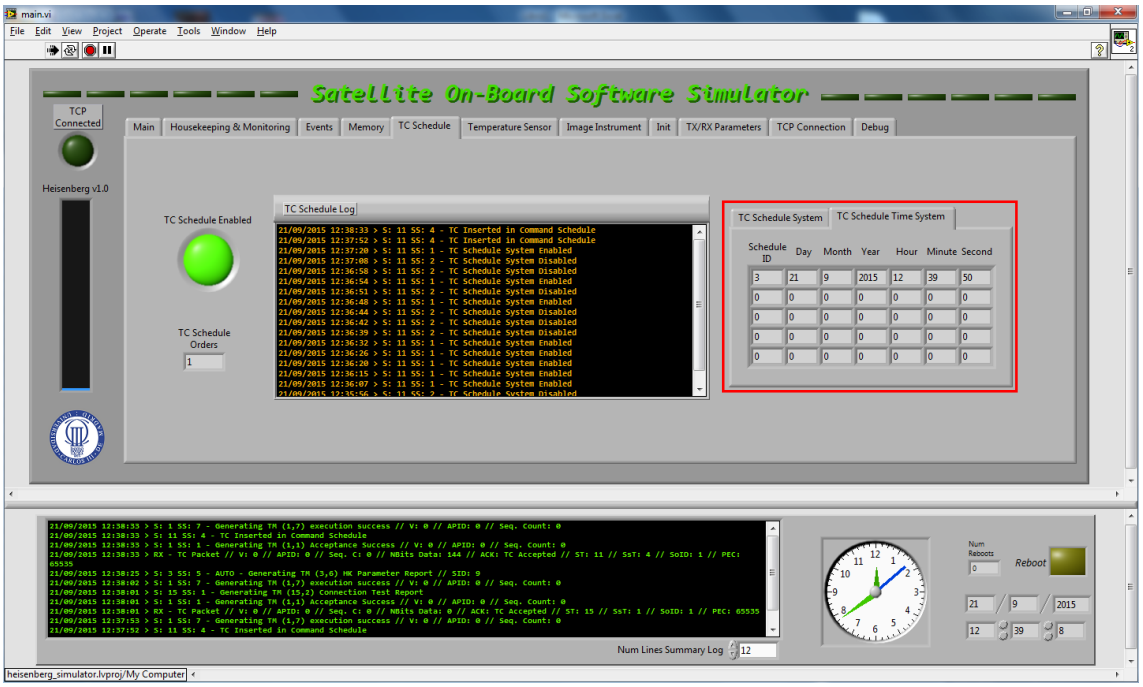


Fig. J.42. Tiempo de ejecución asociado al TC

También se puede cambiar el tiempo de ejecución de un TC de la lista en concreto.

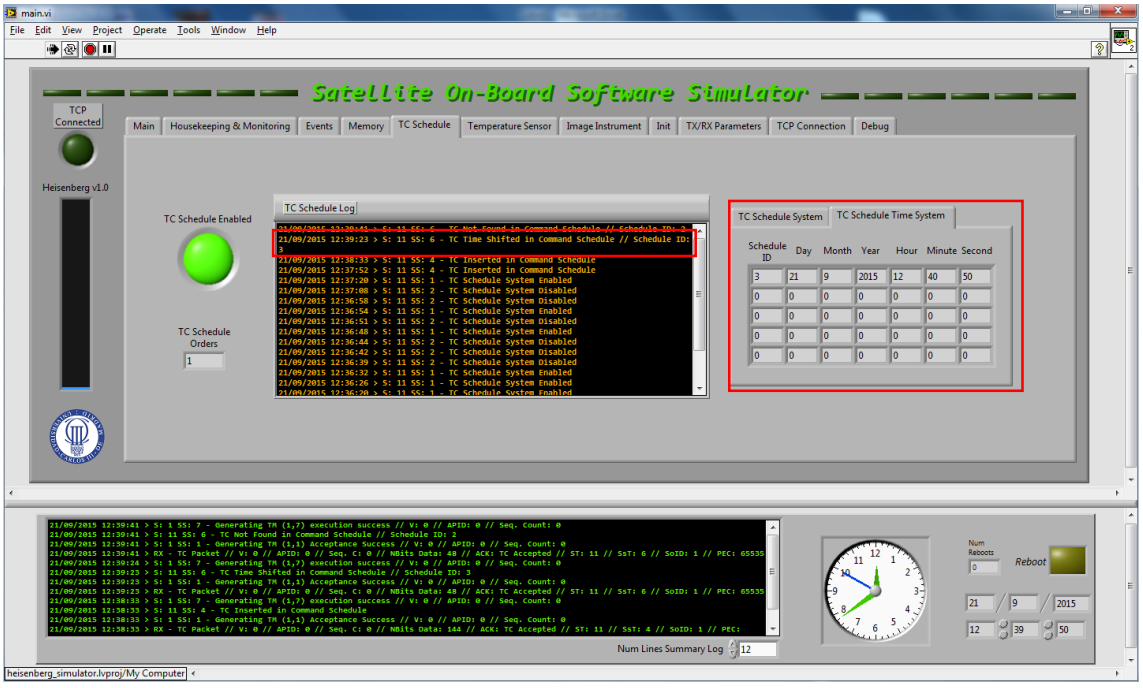


Fig. J.43. Cambio del tiempo de ejecución asociado al TC

Otra opción es resetear la lista de planificación al completo y solicitar el reporte de la lista del satélite.

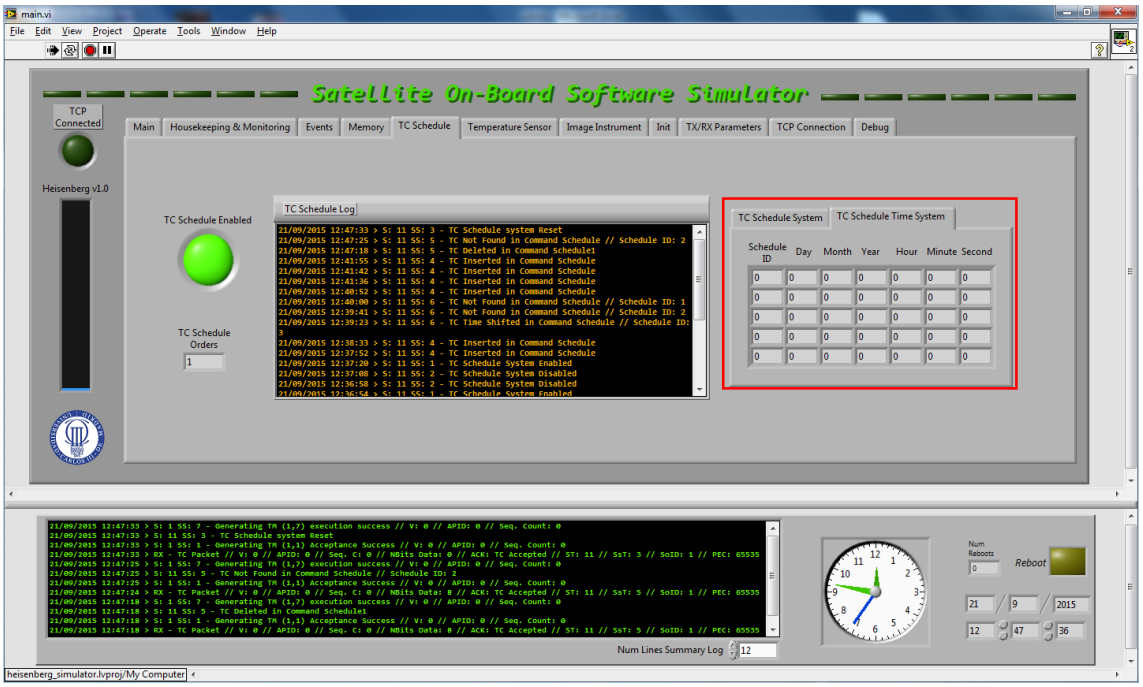


Fig. J.44. Reinicialización de la lista de planificación

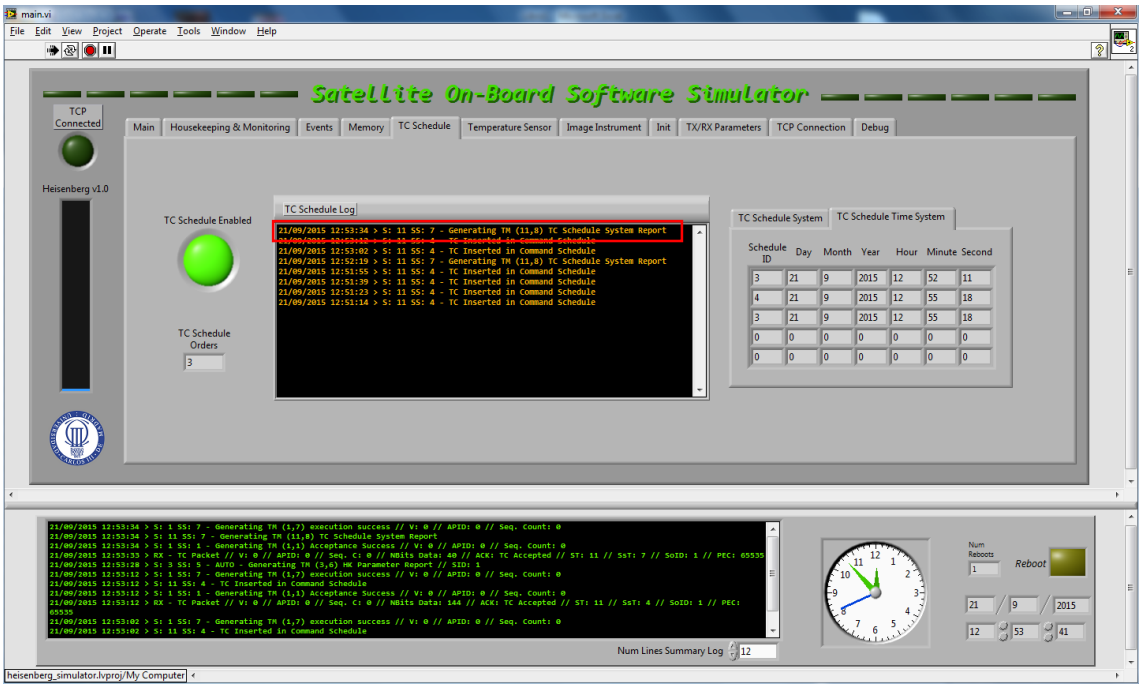


Fig. J.45. Reporte de la lista de planificación

En cuanto a los instrumentos (sensor de temperatura e instrumento de imágenes) su funcionamiento es similar: se puede encender y apagar, reportar el estado (correcto o fallido si se pulsa el botón), cambiar el tiempo de adquisición y actualizarlo, el reporte de datos y la interrupción de dicho reporte.

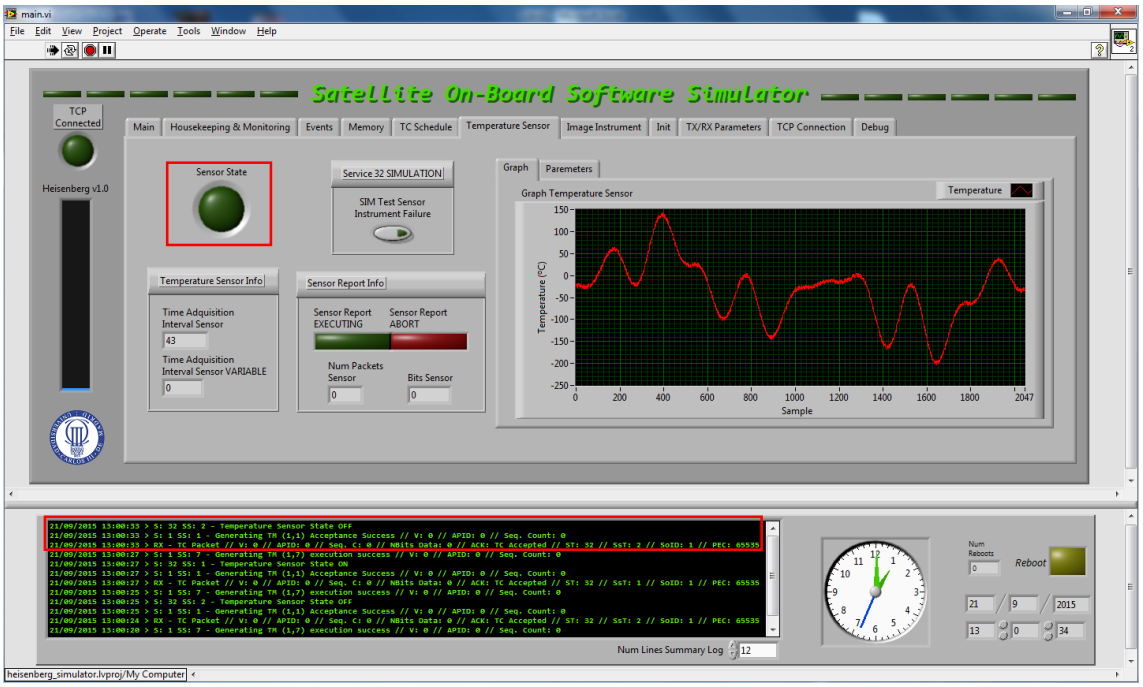


Fig. J.46. Apagado del sensor de temperatura

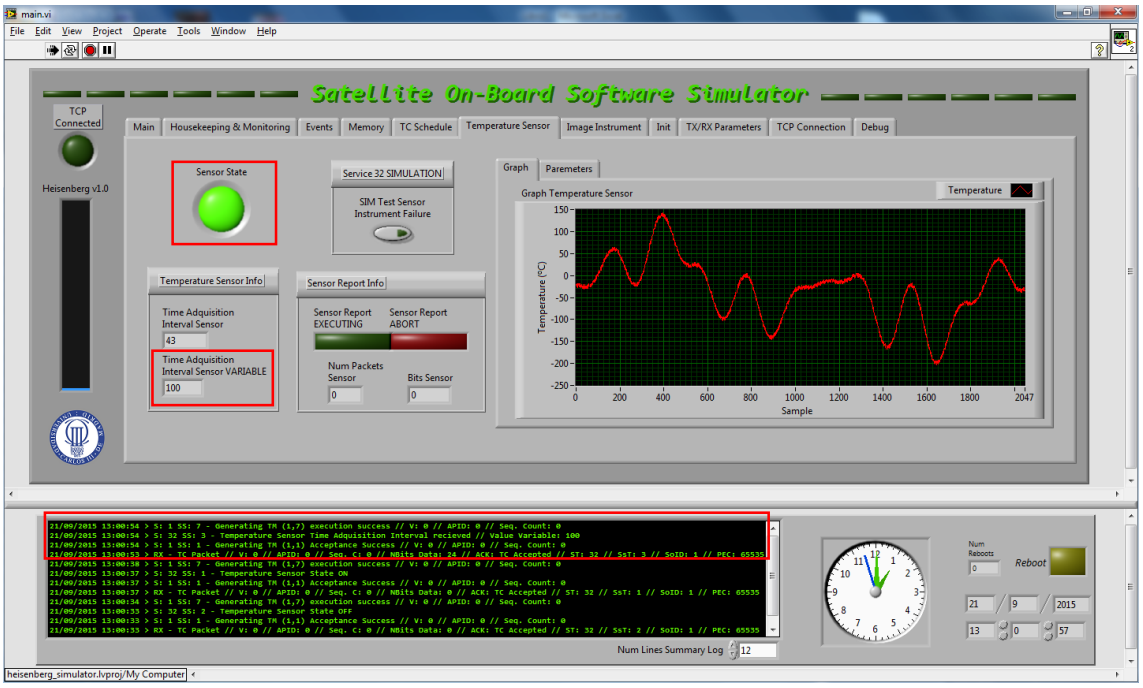


Fig. J.47. Encendido del sensor y cambio del tiempo de adquisición variable

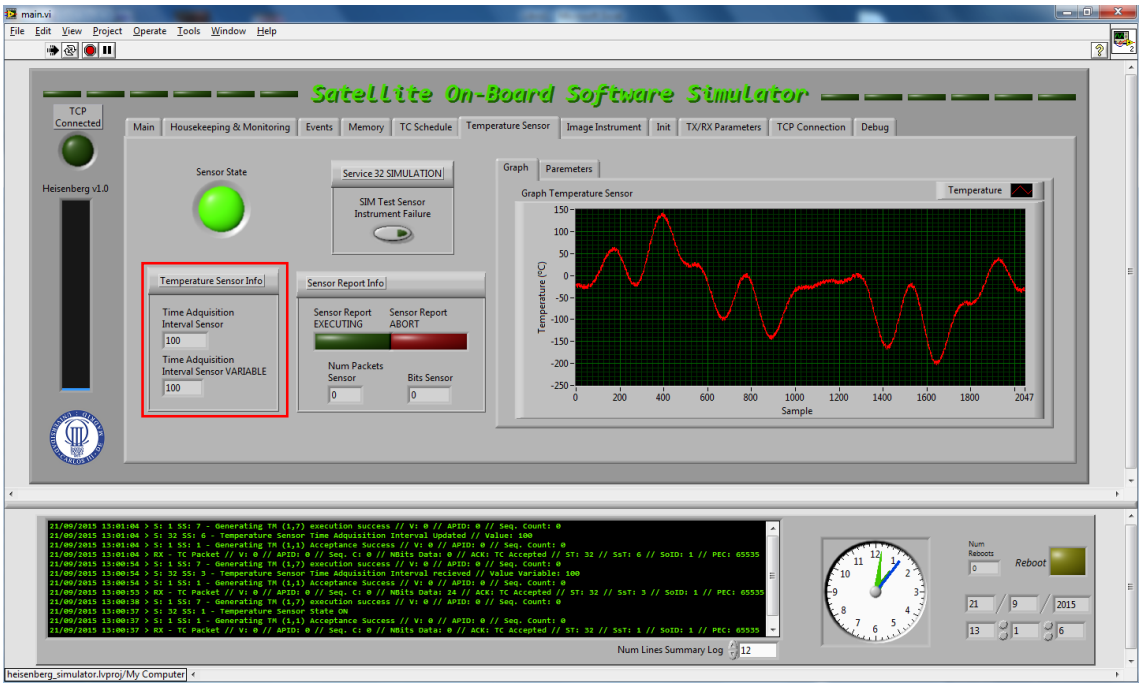


Fig. J.48. Actualización del tiempo de adquisición del sensor

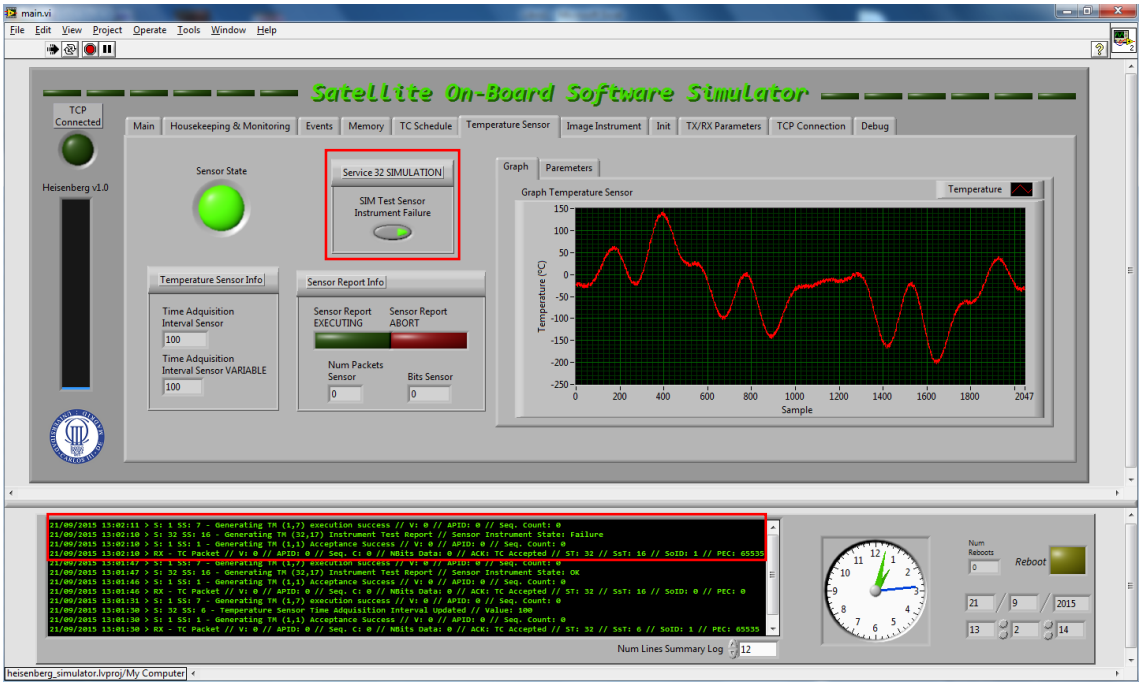


Fig. J.49. Simulación de error en el reporte del sensor

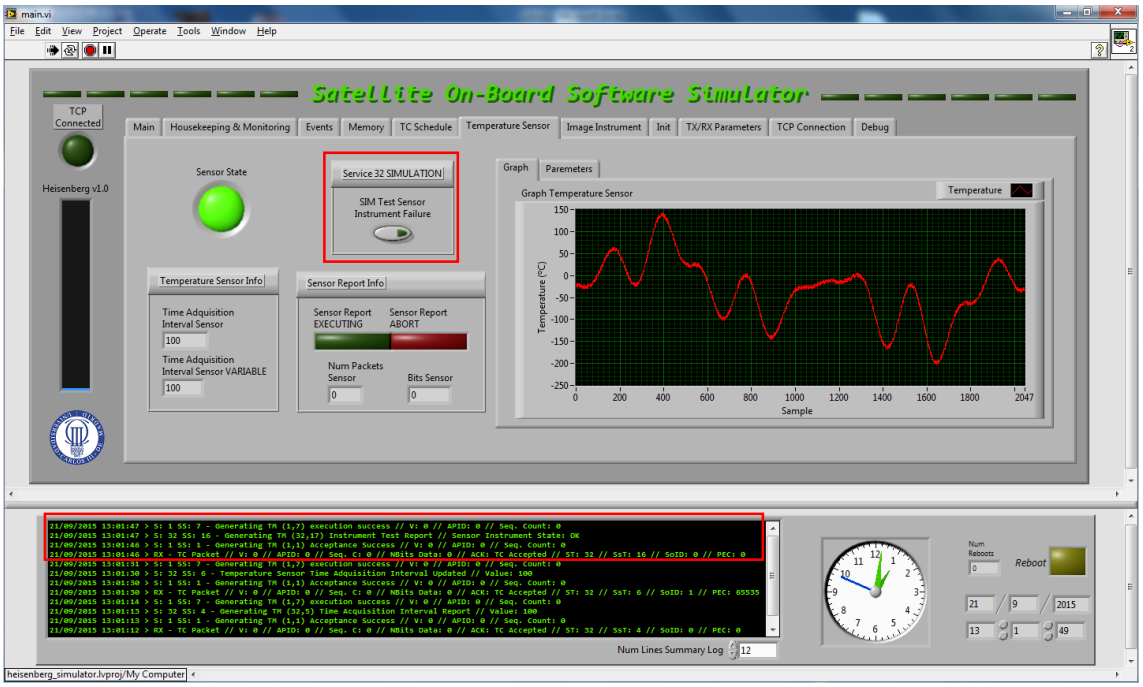


Fig. J.50. Reporte de estado correcto del sensor

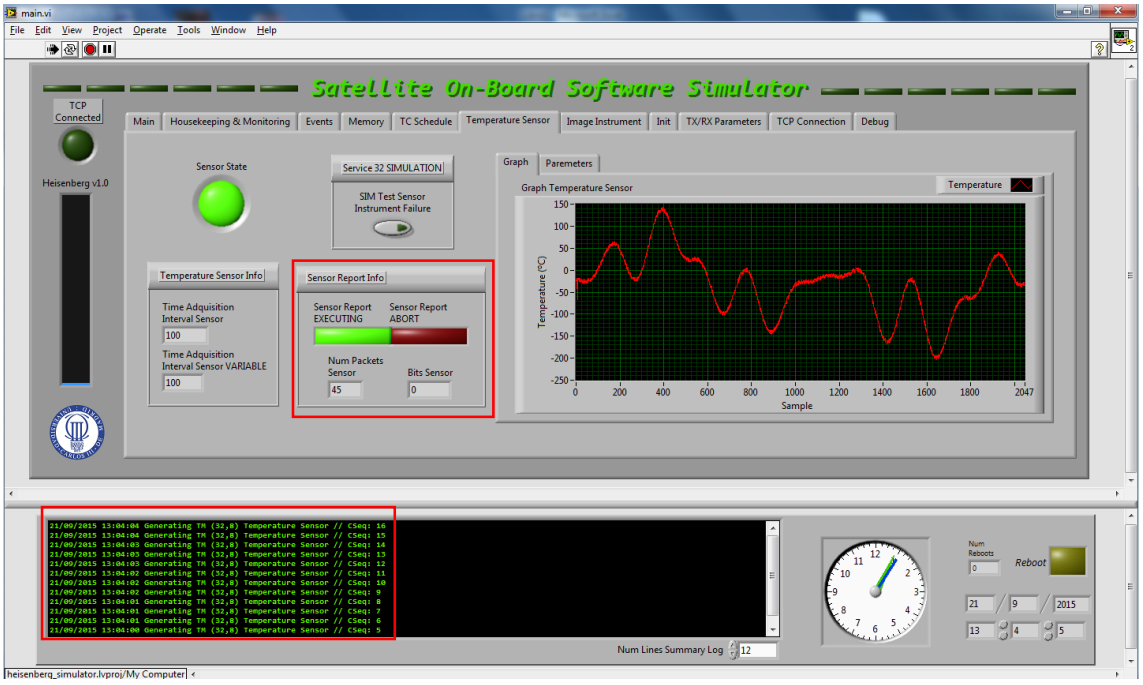


Fig. J.51. Reporte de los datos de temperatura

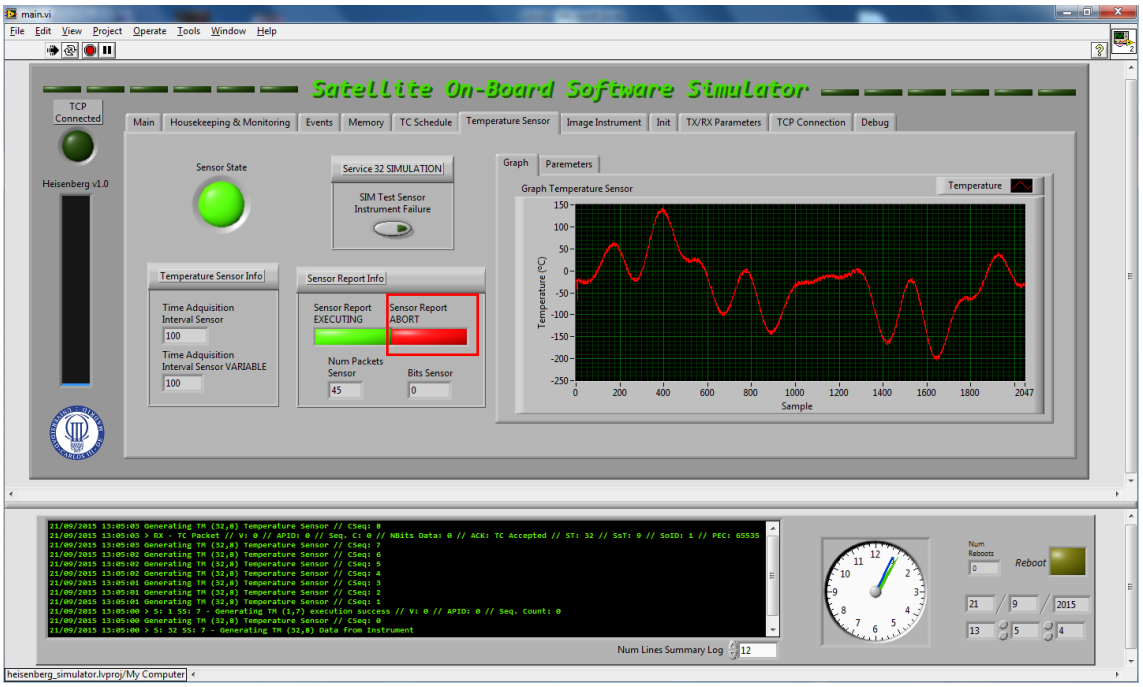


Fig. J.52. Interrupción de la transmisión de la temperatura

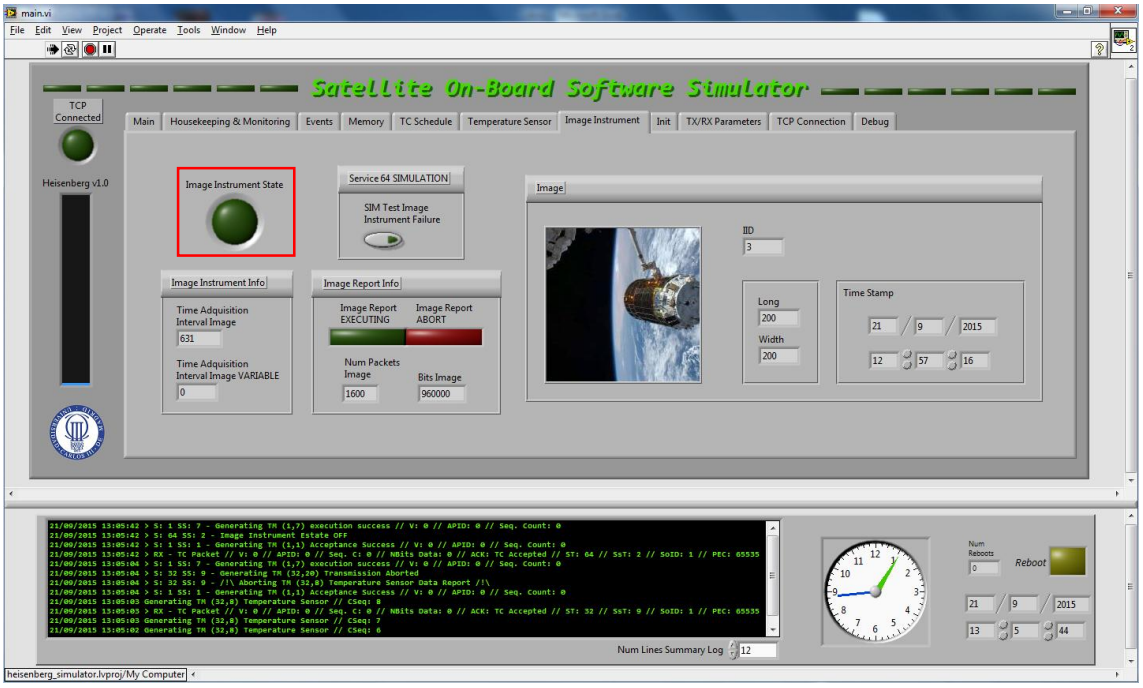


Fig. J.53. Apagado del instrumento de imágenes

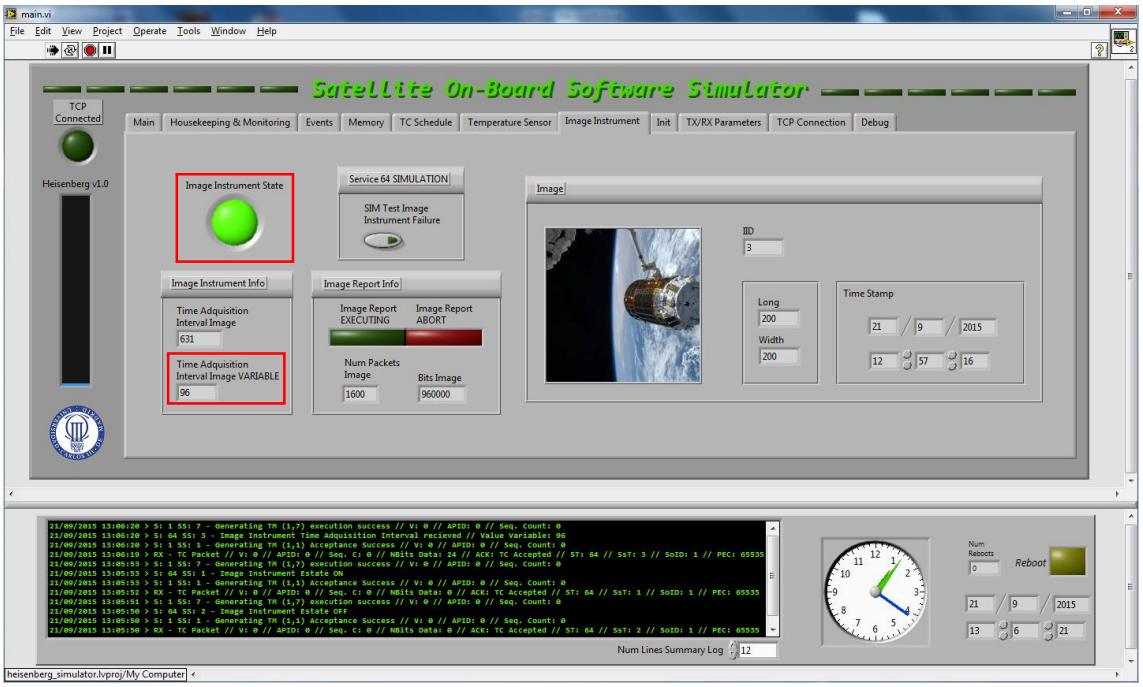


Fig. J.54. Encendido del instrumento de imágenes y cambio del tiempo de adquisición

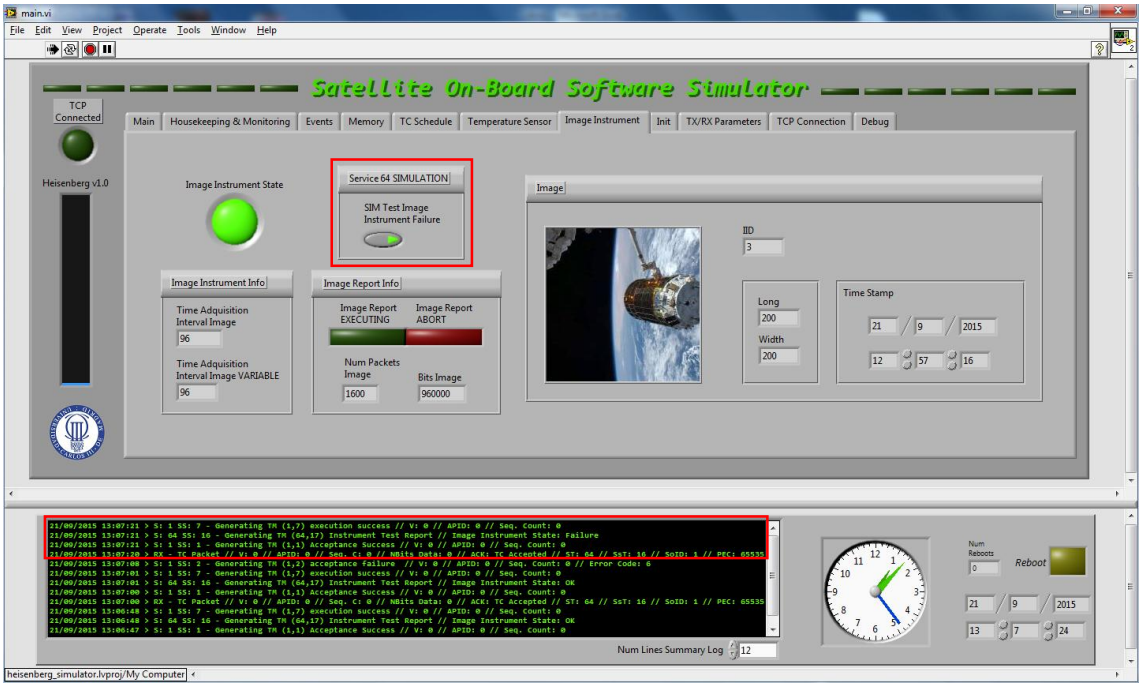


Fig. J.55. Simulación de fallo del instrumento de imágenes en el reporte

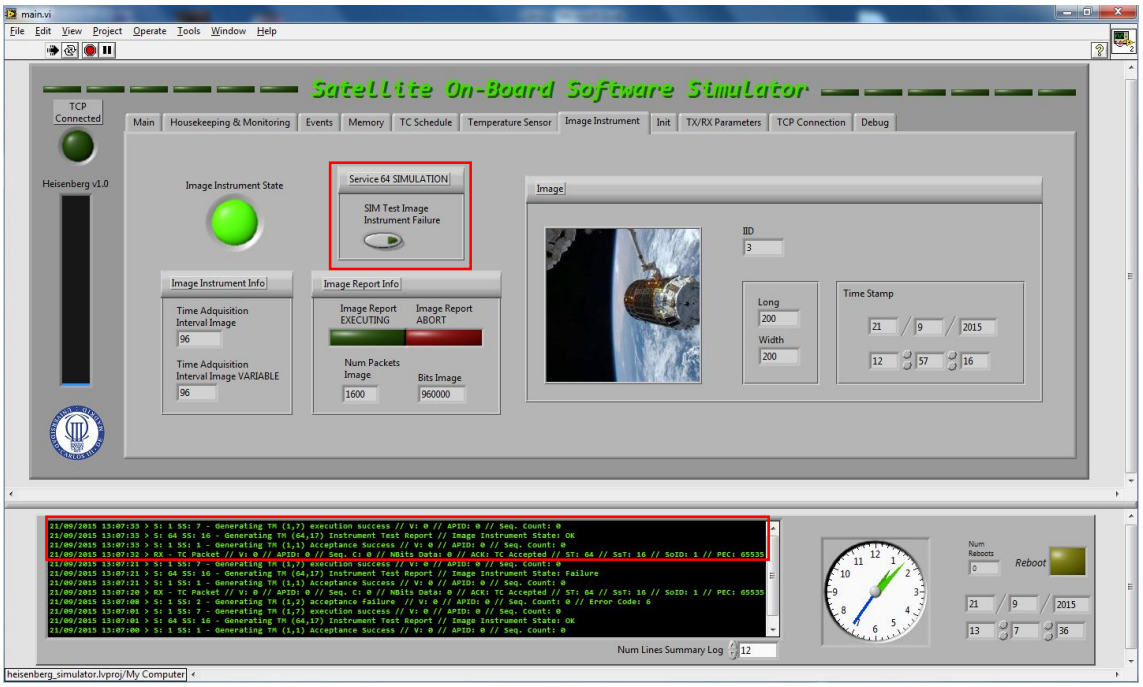


Fig. J.56. Reporte de estado correcto del instrumento de imágenes

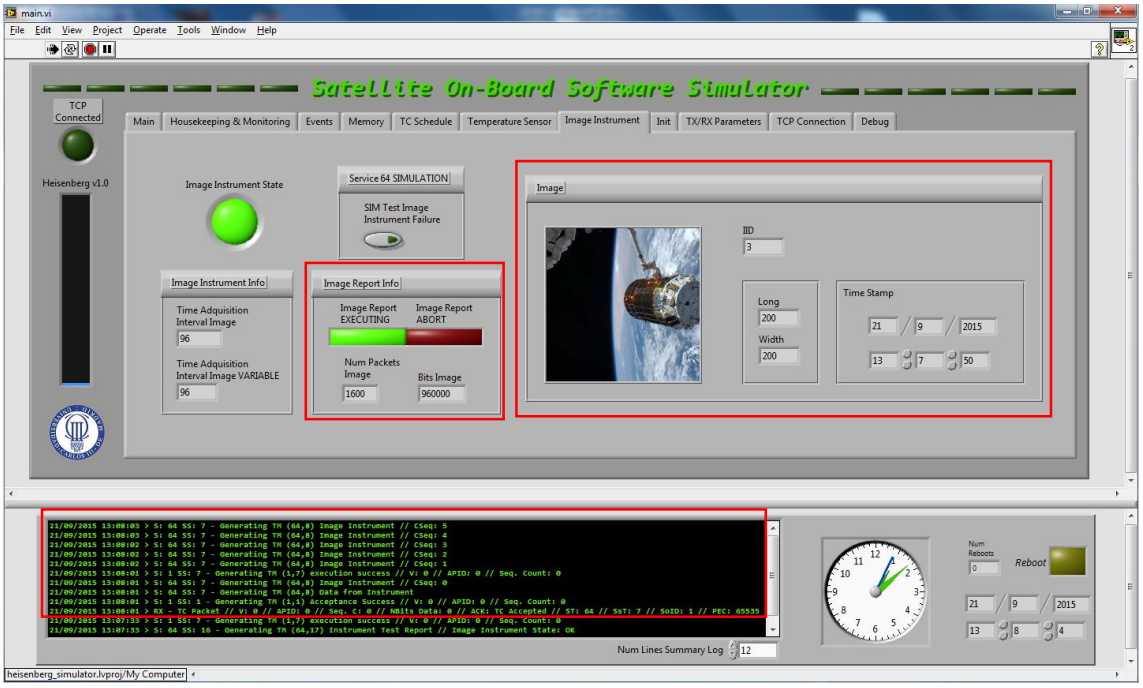


Fig. J.57. Reporte de la imagen

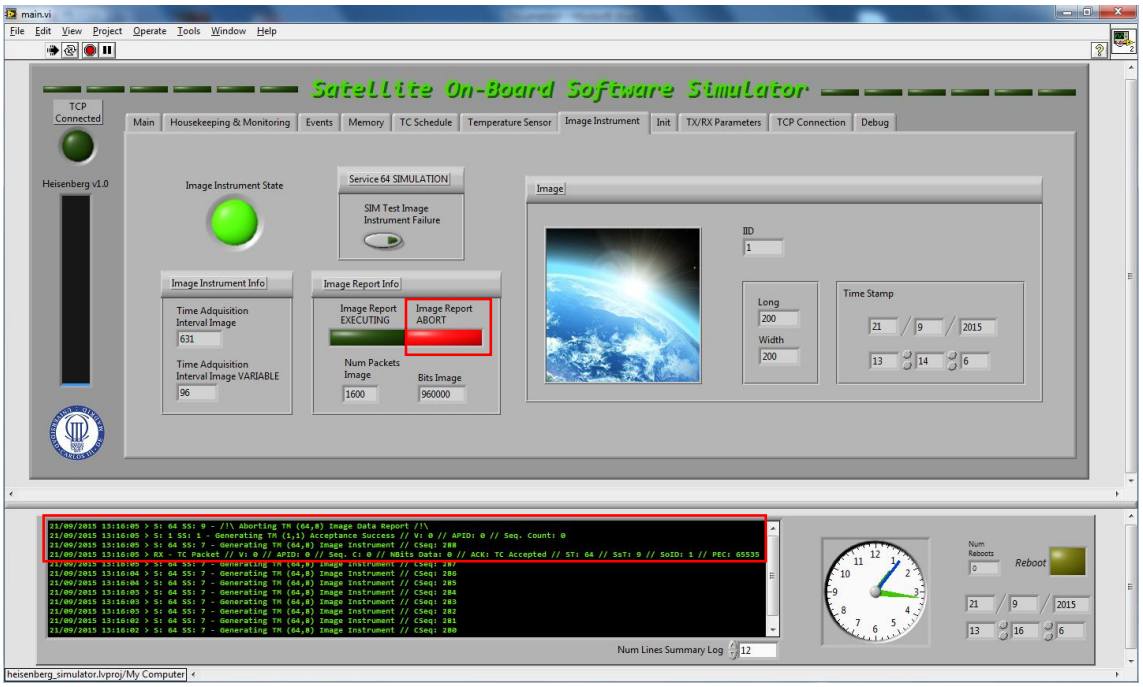


Fig. J.58. Interrupción de la transmisión de la imagen

Referente al reinicio del sistema, cuando se recibe un *TC(128,1)* o se simula el reinicio se activa la barra de progreso y según se van parando los distintos módulos se van encendiendo los LED en rojo. Una vez completado el reinicio se vuelve al estado inicial. Además cambia el inidcador de número de reinicios (al final del reinicio).

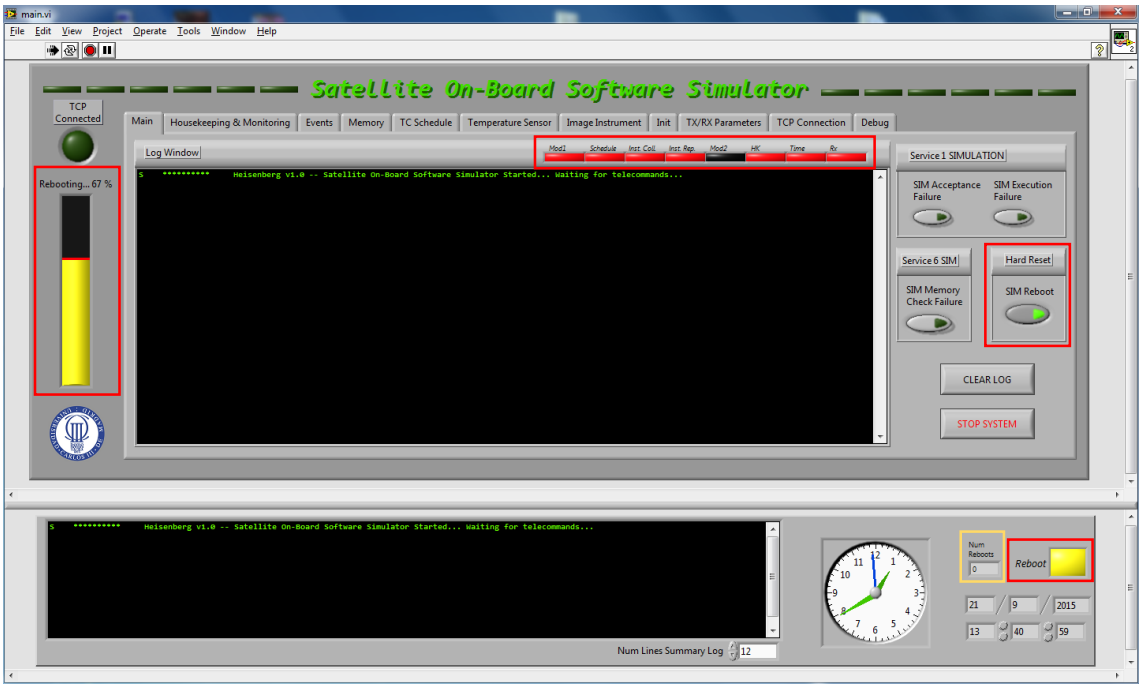


Fig. J.59. Reinicio simulado del sistema

También se pueden borrar los *log* principales pulsando el botón *CLEAR LOG* en la pestaña *Main*. Además en el *log* resumen se pueden ajustar el número de líneas mostradas.

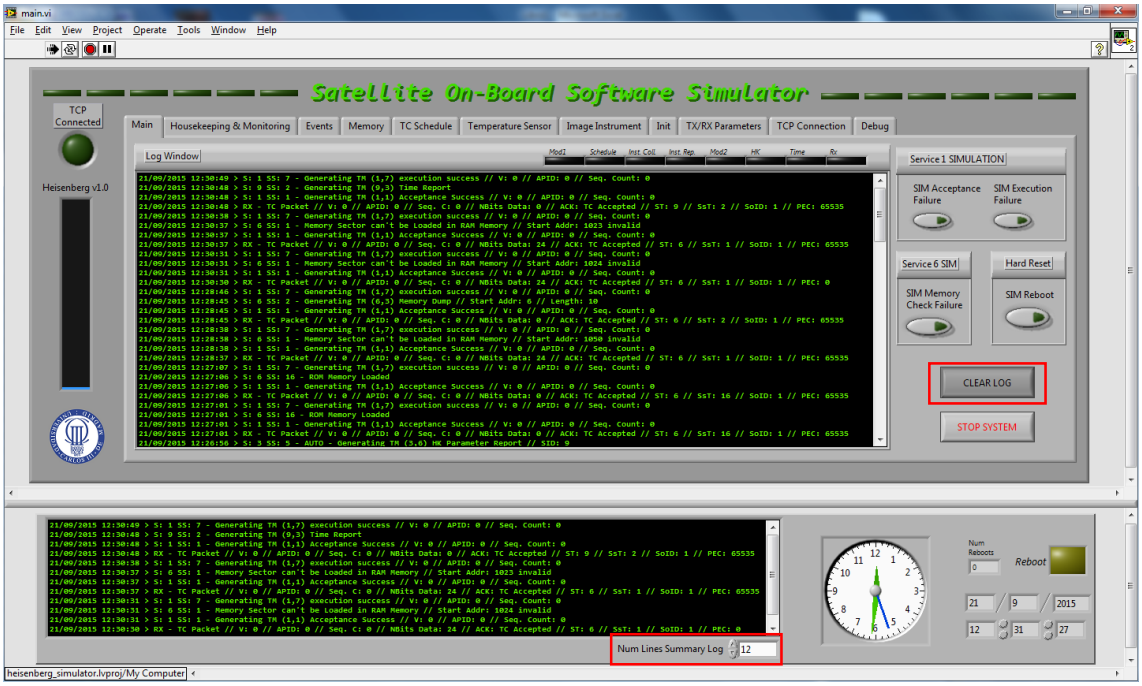


Fig. J.60. Pulsado de borrado de log

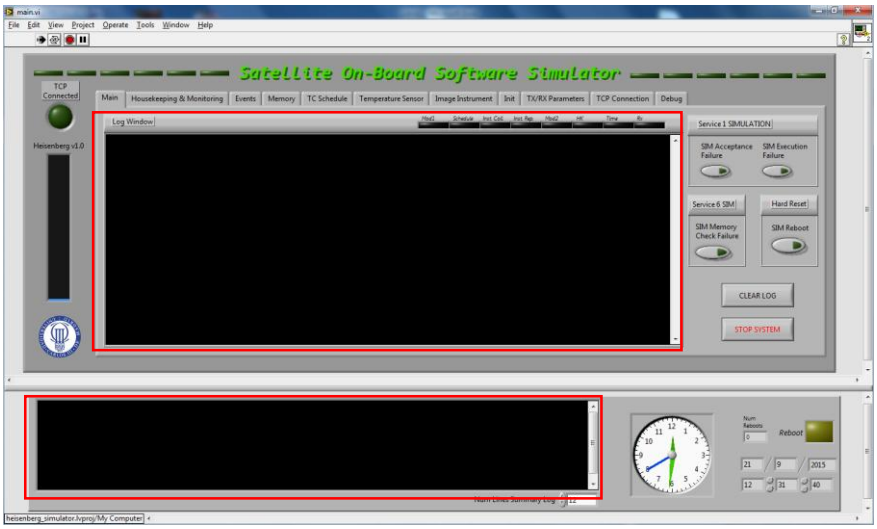


Fig. J.61. Aspecto después del borrado

En el caso de querer establecer una conexión TCP con el terminal móvil, primero se pulsa el botón *INIT Server,* se ajustan los parámetros de la conexión (serán los mismos datso que se introduzcan en el móvil) y se ejcuta el programa. Al principio la barra de progreso vuelve una y otra vez al principio hasta que se conecta con el móvil como se indica en el *Anexo I: Conexión TCP y aplicación Android*.

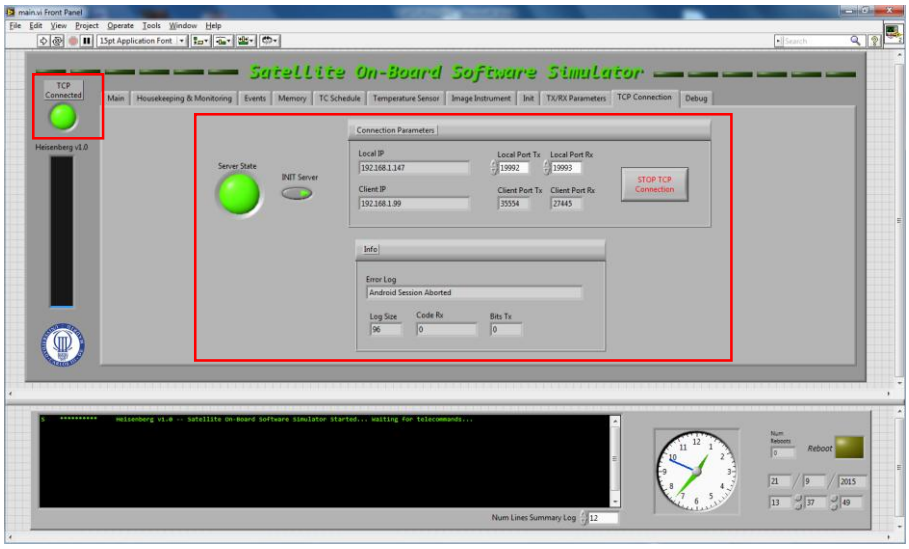


Fig. J.62. Conexión TCP realizada en el programa

El funcionamiento de la aplicación y la interaciión con el programa se describe de forma más detallada en el *Anexo I: Conexión TCP y aplicación Android.*