Universidad de Concepción

Facultad de Ciencias Físicas y Matemáticas

Software Distribuido de Control para Cargas de Calibración en Radio Astronomía

Tesis presentada por **Paulina Unanue Morales**
para obtener el titulo profesional de **Magister en Ciencias con Mención en Física** que otorga la Carrera Ciencias Físicas (y Astronómicas)
Profesor Guía: Rodrigo Reeves Díaz

Departamento de Física
Facultad de Ciencias Físicas y Matemáticas
Universidad de Concepción

Concepción, Chile
Fecha

Agradecimientos

Resumen

El software de control es una parte esencial en el diseño de un telescopio en radioastronomía. Se utiliza para coordinar y administrar sistemas y aplicaciones de software y hardware, con el fin de satisfacer los requisitos de alta precisión, que se requieren al momento de llevar acabó las observaciones radio astronómicas. Esto a llevado a la necesidad de contar con radio telescopios cada vez mayores, con capacidad para obtener imágenes con mejor resolución, avanzando hacia una nueva generación de telescopios extremadamente grandes. El desafío generado en materia de desarrollo de software, requiere del control y monitoreo de cientos de instrumentos en tiempo real y de forma coordinada. Además por lo general los instrumentos son desarrollos por diferentes proveedores en lugares geográficos distintos, requiriendo luego ser integrados a la plataforma de software del telescopio. Generando con esto que la mayoría de radiotelescopios opten por diseños de plataformas distribuidas y con un marco estándar para el desarrollo de software, pudiendo luego integrar el trabajo para operar en conjunto.
La colaboración de múltiples partes asociadas a grandes proyectos como ALMA, han ocasionado la colaboración de una comunidad de desarrolladores que ofrecen soporte a estas plataformas, produciendo arquitecturas más robustas y flexibles.

Esta tesis detalla el trabajo de desarrollo realizado utilizando el software de ACS, como una plataforma con un marco común y un sistema distribuido para el control y monitoreo de dispositivos, aplicado a las 3 cargas de calibración que han sido desarrolladas por el laboratorio Centro para la Instrumentación Astronomica (CePIA), por medio de un acuerdo establecido con el radiotelescopio Large Latin American Millimeter Array (LLAMA). Las Cargas de Calibración tienen como objetivos elemental, caracterizar el receptor de potencia total y la calibración de los datos obtenidos por

el radiotelescopio.

Según lo dispuesto por LLAMA los dispositivos deberán ser controlados utilizando ACS de forma remota a través de un protocolo de comunicación TCP/IP. Con este fin se ha desarrollado un controlador mediante comandos SCPI con todo los puntos de monitoreo y control para las tres cargas de calibración, donde ha sido necesario tener un conocimiento completo de los dispositivos y de su funcionalidad. El controlador se ha desarrollado utilizando una computadora de placa única que se incorpora al sistema de hardware de control de las cargas.
La segunda parte de esta tesis describe el diseño, construcción y las prueba del sistema de control de ACS para las cargas de calibración. Una parte fundamental para esto ha sido el desarrollo del componente/contenedor que encapsula las funciones y propiedades de los instrumentos, pudiendo manejar el ciclo de vida activo de los dispositivos, así mismo se ha implementado la integración de este componente de las cargas, a todas las funcionalidades que ofrece el software distribuido, como el sistema de alarma y el servicio de registro.
Además con el objetivo de generar una visualización simple del funcionamiento del hardware se ha desarrollado una interfaz gráfica para el control del cliente de ACS.
Las pruebas finales se han efectuado con los prototipo de cargas desarrolladas por el equipo de CePIA, demostrando el correcto funcionamiento del software, pudiendo verificar su precisión y un rendimiento adecuado. Por último, se describen los resultados del trabajo de tesis, utilizando el software descrito en esta tesis, lo que ilustra la flexibilidad y utilidad del enfoque de software distribuido, para la instrumentación en radioastronomía.

Abstract

Modern radio astronomy is inseparable from the advanced technology used in radio telescopes. The control system is an essential and important part of designing a radio astronomy telescope. It is used to monitor, control, coordinate and manage software and hardware systems in order to satisfy the requirements of high precision, which are required when carrying out radio astronomical observations. This has led to the need for increasingly larger radio telescopes, with the capacity to obtain images with better resolution, advancing towards a new generation of extremely large telescopes. Therefore, the control system is vital for successful observations .

The challenge generated in terms of software development requires the control and monitoring of hundreds and sometimes thousands of instruments in real-time and in a coordinated manner. In addition, the instruments are usually developed by different providers in different geographical locations. Generating that the majority of radio telescopes opt for common software development platform designs, which can be integrated to operate together. The collaboration of multiple parties associated with projects such as ALMA has led to collaboration in the development of these software platforms, producing more robust and flexible architectures.

This thesis details the work undertaken in the implementation of ACS software, as a platform with a common framework and a distributed system for the control and monitoring of devices, specifically applied to the 3 calibration loads that have been developed by CePIA, through an agreement established with the LLAMA radio telescope. The basic objectives of the Calibration Loads are to characterize the total power receiver and to calibrate the data obtained by the radio telescope .

According to provided by LLAMA, the devices must be controlled by ACS th-

rough a TCP/IP communication protocol. For this purpose, a driver has been developed using SCPI commands for all the monitoring and control points of the three calibration loads, For this, it was necessary to have complete knowledge of the devices and their functionalities. The driver has been developed using a single-board computer that will be incorporated into the hardware system of the loads.

The second part of this thesis describes the design, construction, testing, and use of the ACS control system in the calibration loads. A fundamental part of this has been the development of the component that is responsible to encapsulates the functions and properties of the instruments and is in charge of maintaining the active life cycle of the devices, as well as the integration of this component, into all the functionalities offered by distributed software.

In addition, with the aim of generating a simple visualization of the operation of the hardware, a graphical interface has been developed for its control

The final tests have been carried out with the prototype loads developed by the CePIA team, demonstrating the correct operation of the software, and being able to verify its precision and performance.

Finally, the results of the thesis work in progress are described, using the software suggested in this thesis, which illustrates the flexibility, modularity, and usefulness of the software approach for instrumentation in radio astronomy.

Índice general

Índice de figuras

Índice de cuadros

Lista de código

Lista de Abreviaturas

ACA Atacama Compact Array

ACS ALMA Common Software

ADC Analog to Digital Converter

AGN Active Galactic Nucleus

ALMA Atacama Large Millimeter Array

EHT Event Horizon Telescope

CAN Controller Area Network

CASA Common Astronomy Software Applications

CC Cargas de Calibración

CCAT Cerro Chajnantor Atacama Telescope

CDB Common Database

CePIA Centro para la Instrumentación Astronomica

CERN European Organization for Nuclear Research

CTA Cherenkov Telescope Array

CORBA Common Object Request Broker Architecture

DAC Digital to Analog Converters

DevIO Device Input Output

DHC Data handling system

ESO European Southern Observatory

EM Electromagnetic

FPBW Full Power Beam Width

FPGA Field Programmable Gate Array

FRB (Fast Radio Bursts)

GBT Green Bank Telescope

GNU General Public License

GW Gravitational Waves

GUI Graphical User Interface

HPBW Half Power Beam Width

HDL Hardware Description Language

ICD Control Interface Document

ICE Internet Communications Engine

ICS Telescope Control Instrument

IDL Interface Definition Language

IGM Intergalactic Gas Medium

IP Internet Protocol

LAMOST Large Area Mult -Object Fibre Spectroscopic Telescope

LGPL GNU Lesser General Public License

LLAMA Large Latin American Millimeter Array

LMT Large Millimeter Telescope

MinCyT Ministerio de Ciencia, Tecnología e Innovación

MSP Millisecond Pulsars

NAOJ National Astronomical Observatory of Japan

NRAO National Radio Astronomy Observatory

OCS Observatory control system

OFS Operations Support Facility

OMG Object Management Group

ORB Object Request Broker

RF Radio Frecuencia

SB Scheduler Block

SBC Singel Board Computer

SCPI Standard Commands for Programmable Instruments

SQL Structured Query Language

TAO The ACE ORB

TMCDB Telescope Monitor and Configuration Database

TCP Transmission Control Protocol

TCS Telescope Control System

TMT Thirty Meter Telescope

UDEC Universidad de Concepción

UI User Interface

UML Lenguaje Unificado de Modelado

USB Universal Bus System

USP Universidad de São Paulo

VLBA Very Long Baseline Array

VLBI Very Long baseline interferometry

XML Extensible Markup Language

Capítulo 1

Introducción

1.1. Motivación

1.1.1. Radioastronomía

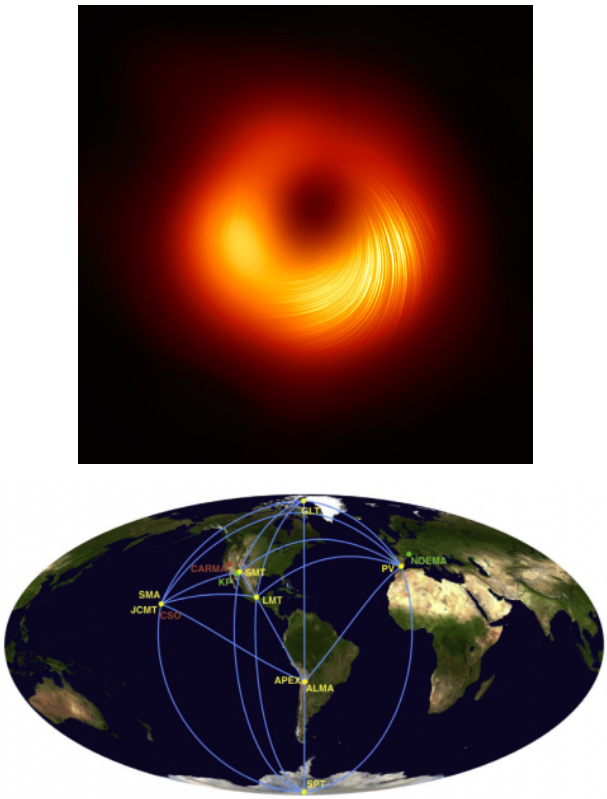
La investigación científica en el campo de la Radioastronomía se lleva a cabo gracias a los radio telescopios, estas infraestructuras técnicas complejas, permiten recolectar emisiones electromagnéticas en frecuencia de radio emitidas por objetos celestes, como resultado de algún proceso físico en el universo lejano.
Estos instrumentos han permitido a la comunidad astronómica realizar observaciones con una sensibilidad y resolución angular sin precedentes, ampliando nuestra comprensión de fenómenos físicos y astrofísicos. Algunos de los fenómenos más interesantes son: las radiogalaxias (por ejemplo, Jennison 1953), los cuásares (Hazard 1963), el fondo cósmico de microondas (CMB; Penzias 1965) y los pulsares. (Hewish 1968). Hoy en día, la radioastronomía tiene como objetivo una amplia gama de objetos astronómicos, incluidas estrellas, fenómenos de alta energía como pulsares y ráfagas rápidas de radio, planetas extrasolares, radiogalaxias, cuásares y fondos difusos, como las características en la linea de 21 cm del universo primitivo.

Los radiotelescopios modernos tienen diseños muy específicos, diseñados para superar los desafíos de obtener señales mas débiles de radio. Requiriendo muchas veces el esfuerzo de distintas organizaciones internacionales, trabajando en conjunto, para lograr sacar adelante estos proyectos.

Algunos instrumentos notables construidos en este último tiempo incluyen el Telescopio Green Bank (GBT), el Telescopio Esférico de Apertura de Quinientos metros (FAST, por sus siglas en inglés) Nan 2011, el Very Large Array (VLA), Hydrogen Epoch of Reionization Array (HERA) DeBoer 2016), Karoo Array Telescope (MeerKAT) Jonas 2018 y el Square Kilometer Array (SKA) Koopmans 2015, Dewdney 2017).

Recientemente el proyecto Event Horizon Telescope (EHT), una Matriz global Very Long baseline interferometry (VLBI) de observatorios de longitud de onda milimétrica y submilimétrica ha conseguido obtener las primera imágenes a escala de Horizonte de eventos de un agujero negro, especificamente del centro de la galaxia en Messier 87 (M87) Antxon Alberdi 2019, proporcionando nuevas pruebas a la relatividad general y al estudio extraordinariamente detallado de la física central de un AGN. Esto ha significado un premio Nobel de física.

Constantemente se realizan nuevas actualizaciones y propuestas más ambiciosas de observación, generando nuevos desafíos en el diseño y desarrollo en los radiotelescopios.



(b)

Figura 1.1: (a) La primera imágen, lanzada en 2019, se muestra el agujero negro del objeto masivo en el centro de la galaxia Messier 87 (M87).Esta imagen muestra la vista polarizada del agujero negro en M87. Las líneas marcan la orientación de la polarización, que está relacionada con el campo magnético alrededor de la sombra del agujero negro, créditos imagen EHT Collaboration. (b) La segunda imagen muestra el mapa global del EHT. Las estaciones activas en 2017 y 2018 se muestran con líneas de conexión y etiquetadas en amarillo, los sitios en comisión están etiquetados en verde y los sitios heredados están etiquetados en rojo.crédito imagen Antxon Alberdi 2019.

Otro ejemplo, son los interferómetros de sondeo rápido útiles para realizar mapas de grandes áreas del cielo , como el VLA Sky Survey (VLASS) Condon 1998 y la reciente matriz de antenas Deep Synoptic Array 2000 (DSA-2000) G. Hallinan1 2021: que detectará y localizará simultáneamente $\sim 10^4$ ráfagas rápidas ((FRB) de radio para la generación de catalos de imágenes cada año.

También brindará la capacidad de realizar observaciones VLBI en combinación con la infraestructura VLBI preexistente, como el Very Long Baseline Array (VLBA),

o con algún conjunto asociado de antenas DSA.
Esto se utilizará para proporcionar nuevas observaciones VLBI de eventos de ondas gravitacionales (Gravitational Waves (GW)) y de eventos de púlsares de milisegundos (Millisecond Pulsars (MSP)) con mediciones que permitan obtener distancias y así caracterizar las GW de agujeros negros supermasivos binarios e individuales.

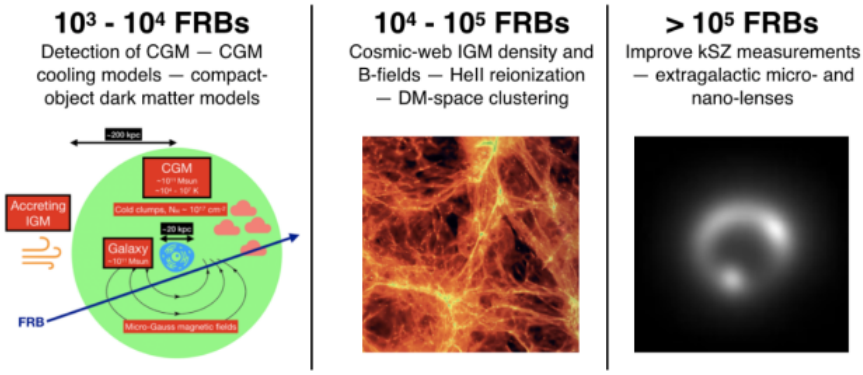


Figura 1.2: Resumen del potencial científico para un gran número de FRB que se localizarán interferométricamente. Las objetivos científicos se centran en la materia oscura para modelos de objetos compactos (por ejemplo, un agujero negro primordial), la caracterización del medio intergaláctico (IGM y micro y nano lentes extragalácticos. Crédito de la imágen G. Hallinan1 2021.

Con requerimientos de diseño de software automático, programación y respuestas anticipadas a los posibles errores del sistema, y así mantener las observaciones de forma continua.
La nueva generación de radiotelescopios, presentan infraestructuras altamente distribuidas que deben llevar acabo el control y sincronización en las monturas motorizadas de las antenas, con capacidad de elevación, azimut y seguimiento continuo de fuentes radioastronomicas de alta precisión , integrando múltiples interfaces de instrumentos heterogéneos y tecnologías de backends digitales que admitan sistemas de adquisiciones y transferencias masivas de datos.

1.1.2. Desafíos en Software de Control para Radiotelescopios

La arquitectura de software del sistema de control de un observatorio (OCS) esta ligada estrechamente a la instrumentación del telescopio, con enfoque en la entrega de productos de datos listos para los objetivos científicos, por lo que representa un pilar en el desarrollo de la astronomía moderna. Actuando como componente maestro encargado de programar, ordenar y supervisar las observaciones radioastronomícas. El OCS es responsable de las operaciones del observatorio de alto nivel, incluida la secuenciación de las interfaces de usuario, el procesamiento de datos, así como también del control en la asignación de recursos y la supervisión y el mantenimiento del sistema para la obtención de los datos.Es decir de todas las operaciones durante el ciclo activo de las observacion .
Si bien en un principio, los radiotelescopios contaban con estructuras de equipos simples, y funcionalidades de software basadas esencialmente en una lógica única y G. Juen. 1983, Jessner. 1995, con el tiempo y debido al aumento en los requerimientos científicos, se género un avance continuo de nuevas tecnologías, enfocadas en el desarrollo interno o back-end del telescopio y en la integración de grandes cantidades de equipos auxiliares de control y de medición, que incluyen principalmente el control activo de la superficie de la antena, ajustes de subreflectores, interferometría láser, calibración de instrumentos y mediciones meteorológicas con monitoreo del entorno electromagnético.

Los sistemas de software se ven constantemente desafiados por un alto nivel de distribución, en tareas relacionadas con el control y los diversos tipos de patrones de diseño para el flujo de datos del telescopio, en conjunto con el procesamiento de amplios rangos de muestreo de frecuencias (mediante correlación cruzada). Debido a esto, el sistema de control requiere del monitoreo y detección de errores y fallas de manera efectiva, impidiendo su propagación y mitigándolos con precisión, en el menor tiempo posible, para permitir restablecer los servicios de observación.
Esto a generado una transformación en la lógica del diseño de software en los radiotelescopios, pasando de arquitectura centralizadas con estructuras simple y ciclos de desarrollo cortos a arquitectura distribuidas extensible y flexibles, con sistemas automatizados en la programación de tareas, basadas en marcos de software inter-

medios o middleware Zhiyong Liu et al. Jun Li 2021.
Alan Bridger 2010 identifica un paradigma común en las arquitecturas de software de los observatorios modernos, funcionando a traves de un sistema principal, constituido por diversos subsistemas secundarios, como son: el sistema de control del telescopio (TCS), sistema de control de instrumentos (ICS),el sistema de manejo de datos (DHC) y el sistema de control del observatorio (OCS), cada uno con funcionalidades bien definidas y que a su vez pueden utilizar diversos tipos tecnologías.

Algunos ejemplos de software de control distribuidos modernos utilizados en radiotelescopios, son: Control System LBT (Fisher.,1998, The GBT Precision Telescope Control System (Ochsenbein y D. Egret, 2004), TMT Control Software (Corinne; Trinh Thang . Silva, 2004), ALMA Control System (ACS) (H.Sommera A.Caproniae et al. G.Chiozzi, 2004), LAMOST Observatory Control System (Yu Xiaoqi Huang Kun et al. Wang Jian, 2006) .

Figura 1.3: Radio telescopio ALMA.Crédito de la imagen, Observatorio Europeo Austral (ESO) [28].

La ventaja de los sistemas distribuidos es que proporcionan un diseño modular al sistema lógico, mejorando la reutilización del código y la eficiencia en el desarrollo; los módulos se pueden desarrollar, implementar y ejecutar de forma completamente independiente del resto de los componentes. Esto resulta de gran interés al momento de integrar diversas interfaces de intrumentos, pudiendo eliminar o modificar módulos o subsistemas del sistema principal. El marco del software intermedio, entrega

un soporte en la sincronización de las operaciones y trasferencia de datos de forma eficiente, otorgando una interfaz unificada dentro del sistema. Adicionalmente se encarga de ocultar los detalles del sistema operativo, la red o la base de datos, para que así los desarrolladores solo necesiten prestar atención a la lógica especifica del telescopio.

Dentro de este mismo contexto fue diseñado Alma Common Software (ACS) Gianni Raffi, 2002 para el proyecto The Atacama Large Millimeter/submillimeter Array (ALMA).El observatorio ALMA es un interferómetro que esta conformado por 66 antenas de 12 metros de diámetro y 12 antenas de 7 metros de diámetro .El radiotelescopio opera en longitudes de onda de 0.32 mm a 3.6 mm. ALMA es una asociación entre Europa (ESO), América del Norte (NRAO) y Japón (NAOJ), ubicado en el Llano de Chajnantor, en el desierto de Atacama en Chile, a una altura de 5000 msnm. El uso de una capa común de software, se introdujo en ALMA como la forma de hacer cumplir el uso de construcciones generales, en un equipo de desarrollo altamente distribuido geográficamente.

El marco de ACS consta de una colección de patrones, componentes y servicios comunes para el control y monitoreo de instrumentos. El corazón de ACS es un modelo de componentes basado en la Implementación de Objetos Distribuidos en programación enfocada a objetos, utilizando un agente intermedio como Common Object Request Broker Architecture (CORBA) Group 2002. ACS se basa en la experiencia acumulada de proyectos científicos similares dentro del contexto astronómico, como de proyectos relacionados con aceleradores de partículas, reutilizando conceptos y tecnologías aplicadas anteriormente, por ejemplo con el software común Very large Telescope (VLT) P.Sivera 2007 y el Sistema de alarma European Organization for Nuclear Research (CERN) LASER A.Capronib 2006. Aunque ACS está diseñado para ALMA, el software puede y está siendo utilizando en otros sistemas de control y proyectos de software distribuido, ya que se ha desarrollado para ser un marco de control común que utiliza patrones de diseño genéricos y componentes listos para ser utilizados Kim Gillies et al. Gianluca Chiozzi 2008. Mediante el uso y construcción de componentes estándar, facilitando a los desarrolladores la comprensión de la arquitectura de los módulos del software. Esto permite que el mantenimiento sea

accesible incluso en un proyecto de software muy grande como ALMA.
La reutilización de software se ha convertido en un factor clave en la industria debido a sus potenciales beneficios y ahorros. Gracias a que ACS es liberado periódicamente con una Licencia pública general (General Public License (GNU)) en su sitio web [8], a creado una comunidad de desarrolladores que dan soporte al software, alentado a una docena de instituciones ajenas a ALMA a que hagan uso de ACS, tanto para aplicaciones industriales como educativas Ibsen 2016, ver figura 1.4.
De esta forma abarca proyectos de telescopios amateur Mauricio A. Araya Joao S. López. odrigo J. Tobar 2008, como también en grandes proyectos de radioastronomía. Solo en estos últimos años, el telescopio Cerro Chajnantor Atacama (CCAT) Alan Bridger 2010, el Cherenkov Telescope Array (CTA) Oya 2021 y Large Latin American Millimeter Array (LLAMA), han adoptado ACS como su propio sistema de control.

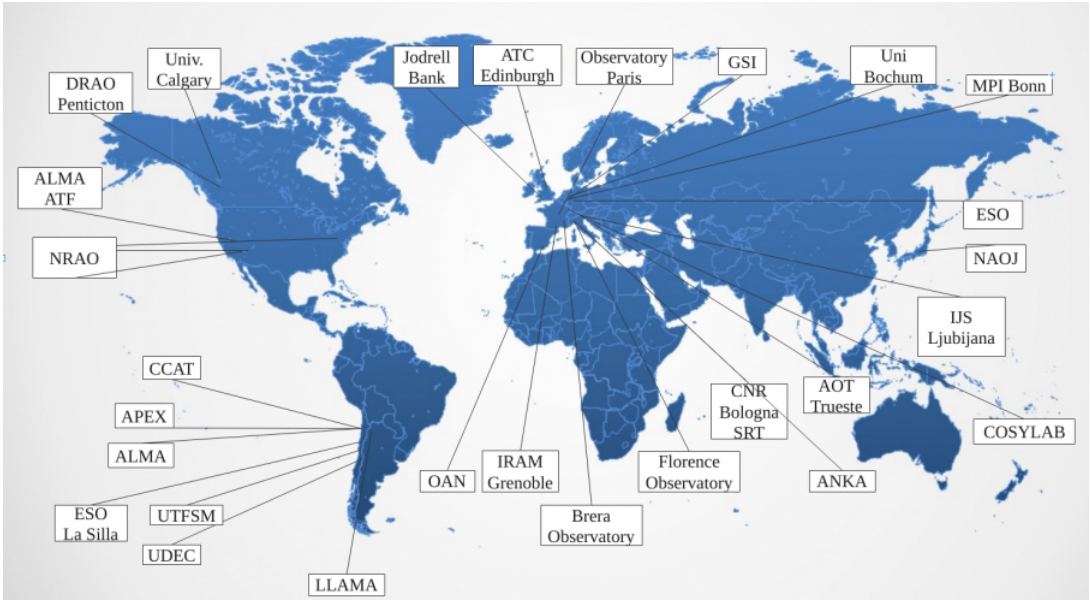


Figura 1.4: Sitios actuales con Instalaciones del software ALMA Commmon Software (ACS), desde 2006.

El proyecto LLAMA es una iniciativa del Ministerio de Ciencia e invocación argentino (MinCyT), y la Fundación de Apoyo a la Investigación del Estado de São Paulo en conjunto con la Universidad de São Paulo (USP) para la construcción de un radiotelescopio de 12 metros de diámetro ubicado en la región de Alto Chorrillos, en la provincia de Salta, Argentina a 4832.5 metros sobre el nivel del mar Romero

2018.

Figura 1.5: La imagen muestra la ubicación de Alto Chorrillos con respecto a ALMA. La distancia entre ambos se encuentra indicada. El punto en rojo indica el lugar de LLAMA. Créditos [56].

La antena de LLAMA tiene un diseño óptico con una configuración Cassegrain fabricada por Vertex Antennentechnik, con un rango de frecuencia de funcionamiento de 30 GHz a 950 GHz.Los objetivos científicos que pretenden ser estudiados por el radiotelescopio son: el Sol, Planetas, Objetos Estelares, Jets Astrofísicos y emisión Máser, Medio Interestelar Galáctico e Intergaláctico, Galaxias y Altas Energías. El radiotelescopio sera utilizado como parte del interferómetro intercontinental, Very Long Baseline Interferometry (VLBA) en latino américa, además se prevé que eventualmente LLAMA forme parte del EHT y otras redes milimétricas.

Gracias a un acuerdo de colaboración entre LLAMA y el Centro para la Instrumentación Astronomica (CePIA) en Departamento de Astrónomia de la Universidad de Concepción (UdeC) se ha resuelto la elaboración de tres Cargas de Calibración (CC), que tienen como objetivo caracterizar la temperatura de brillo del cielo para así poder relacionar sus datos a alguna fuente de observación en particular.

Figura 1.6: Antena melliza al proyecto LLAMA, que opera actualmente en Chile.Credito Foto [11].

Para esto se requiere establecer una temperatura de ruido mínima detectable por el receptor y así conseguir una calibración secuencial de este para lograr asociar una medida de potencia a un valor de temperatura de brillo en el cielo.

EL control de las cargas de calibración dentro de la antena debe efectuarse de forma remota siendo integradas al sistema de control del radiotelescopio, mediante su implementación en la versión del software requerido por ACS LLAMA.

1.2. Enfoque del problema

Los radio telescopios modernos deben visualizarse como verdaderos laboratorios de instrumentos, donde la flexibilidad y escalabilidad del sistema para integrar, coordinar hardware y aplicaciones de software es primordial. Debido a las nuevas necesidades científicas, las capacidades de observación de un telescopio deben ampliarse o cambiarse durante su ciclo de vida. Pudiendo requerir administrar una gran cantidad de dispositivos heterogéneos desarrollados por distinto tipo de proveedores. ACS ofrece una solución a esto como una plataforma de software común con licencia LGPL presentando portabilidad, mantenibilidad y configurabilidad, tanto para proyectos científicos de gran escala como para diversos proyectos académicos.

El radiotelescopio LLAMA, que representa en la actualidad el tercer radiotelescopio más importante de Latinoamérica , como miembro de la comunidad de ACS, ha integrado el software como sistema de control principal en su radio telescopio. Por lo que requiere que los dispositivos asociados al funcionamiento del observatorio sean integrados al software. El actual trabajo ofrece una solución al desarrollo del control y monitoreo de las cargas de calibración mediante ACS, como parte del proyecto de cooperación entre CePIA y LLAMA. El preciso control de los dispositivos es de vital importancia para la calibración del receptor y con esto poder llevar a cabo las observaciones de forma adecuada. Contribuyendo a la colaboración con el grupo de computación de LLAMA, para el desarrollo del software.

1.3. Objetivos

El objetivo principal de este trabajo es el desarrollo e implementación del software ACS para el control en tiempo real de dispositivos de hardware, aplicado a las cargas de calibración para el radio telescopio LLAMA. Este trabajo está guiado por las siguientes áreas clave de desempeño u objetivos específicos.

1. Desarrollo de la Interfaz de comunicación con las cargas de calibración.

    - Esto implica el desarrollo del controlador serial (desde la computadora de placa única con la tarjeta FPGA, para los puntos de monitoreo y control de las tres cargas de calibración.
    - Comunicación del instrumento por medio de un protocolo TCP/IP con ACS mediante comandos SCPI.

2. Instalación e implementación del software.Requiere incluir e integrar, las librerías necesarias para la correcta construcción de código en ACS LLAMA.

3. Crear esquemas Extensible Markup Language (XML) de la Common Database (CDB) para el componente y el contenedor. CDB es la base de datos de configuración, esta base de datos informa al marco sobre los componentes de software disponibles. La base de datos compara el componente con un esquema XML que define las propiedades del componente de software.

4. Desarrollo del componente. esto requiere la generación del código necesario para el ciclo de vida del componente y establece además las propiedades y funciones de los dispositivos de hardware que se requieren controlar, mediante uno de los lenguajes de programación admitidos por ACS, especificamente C++.

5. Desarrollo de una interfaz gráfica de usuario de control del componente, esta interfaz debe permitir una visualización simple de los puntos de monitoreo y control en tiempo real.

6. Evaluar su rendimiento, en relación a las necesidades de LLAMA a través de una conclusión.

1.4. Estructura del Documento

Con el fin de facilitar una comprensión adecuada del tema propuesto de tesis, se intenta presentar el material de manera concisa, comprensible y estructurada.

En primer lugar, la familiaridad con el título del tema de investigación y el planteamiento del problema es de suma importancia. Estos ya se han discutido en la portada y en la introducción respectivamente.
En el capitulo 2 se hace una revisión de las ideas básicas de radio astronomía y de tecnologías claves en los radiotelescopios, se analiza los concepto teórico físico y matemáticoutilizadas. También fundamentales para la función que cumplen las Cargas de Calibración desarrolladas y su utilidad en el radiotelescopio LLAMA.
En el capitulo 3 se presenta la tecnología de software utilizada por LLAMA, revisando la teoría detrás del sistema, así como los requerimientos para las aplicaciones utilizadas.También se discuten los diversos subsistemas y su funcionamiento, para la generación de código, que son la base de este trabajo de tesis, enfocada en el control de instrumentos.
En el capitulo 4 se presenta la tecnología de hardware utilizada para la comunicación con los instrumentos, y se describe la interfaz de todos los puntos de control y monitoreo, que definen a las tres cargas de calibración, igualmente se desarrolla la implementación del servidor de control para acceder a las cargas de forma remota a través del protocolo TCP/IP.

El Capitulo 5 se presenta la implementación del software, cubriendo la metodología que ha sido utilizada y se analiza el desarrollo de las funciones con más detalle, como también las clases de objetos utilizadas dentro del ciclo de vida del componente y su posterior despliegue. Adicionalmente se agrega el desarrollo de la interfaz gráfica de usuario al componente de los instrumentos.
En el Capitulo 6 se detalla la verificación de la solución propuesta mediantes pruebas directas con los prototipos de cargas de calibración, para la posterior validación del trabajo realizado. Se concluye con el Capítulo 7, al resumir el trabajo, discutiendo esfuerzos futuros y haciendo algunas recomendaciones.

Capítulo 2

Conceptos Generales de Radioastronomía

2.1. Introducción

Como cualquier herramienta científica, el diseño del instrumento radioastronómico y su control solo se puede lograr con éxito con una comprensión previa de la aplicación para la que está destinado. En consecuencia, para tener éxito en la tarea de desarrollar dispositivos como las cargas de calibración, que serán utilizadas en el radiotelescopio con rangos de frecuencias milimétricas y submilimétricas, es necesario tener un conocimiento general de radioastronomía y comprender los principios de funcionamiento de los telescopios.
Con este propósito, este capítulo presenta una introducción a la historia y la teoría de la radioastronomía y describe parte de la función y estructura de los receptores de microondas y ondas milimétricas más comunes y la necesidad de calibración de los datos obtenidos por estos instrumentos. La información presentada aquí proporciona una base para comprender los requisitos de diseño de los dispositivos de calibración desarrollados por CePIA.

2.2. Historia de la Radioastronomía

Las emisiones de radio del espacio exterior fueron detectadas por primera vez por Karl Jansky en 1931, cuando trabajaba para Bell Laboratories en Holmdel (Nueva

Jersey, EE. UU.). Jansky fue asignado para investigar posibles fuentes de interferencia que podrían afectar las comunicaciones de radio telefónicas transatlánticas. Para este propósito, construyó la antena primitiva que se muestra en la Figura 2.1 Weinreb. 2011.

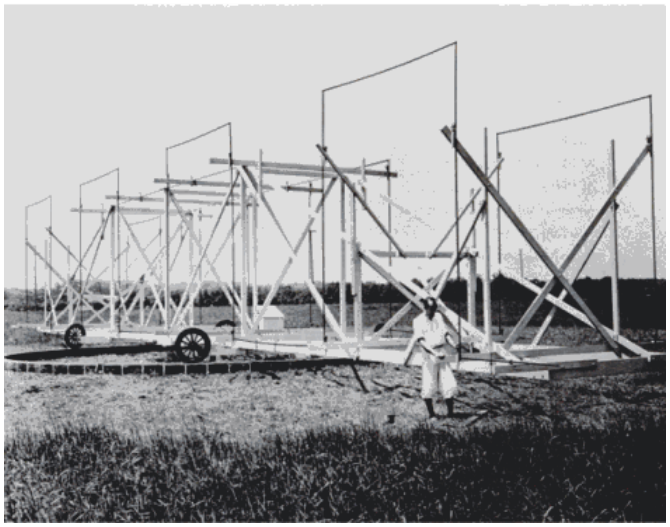


Figura 2.1: Karl Jansky con la Antena Jansky descubriendo las primeras ondas de radio de la Vía Láctea, (cortesía de Bell Telephone Laboratories).

La antena fue diseñada para recibir ondas de radio a 20,5 MHz, de esta forma, apunto la antena a todas las direcciones posibles desde el horizonte hasta el cenit. Después de varios meses de observaciones, detectó algunas emisiones de radio débiles de la Vía Láctea y estableció el campo de la radioastronomía. Después de su descubrimiento pionero, Jansky no pudo obtener los fondos para continuar la investigación en radioastronomía. Su trabajo fue continuado en 1937 por un ingeniero radioaficionado, Grote Reber, quien construyó un reflector parabólico de 10 metros en el patio trasero de su casa en Wheaton (Illinois, EE. UU). Reber mapeó porciones de la Vía Láctea en un rango de frecuencia más alto de 160 MHz y publicó los resultados de sus observaciones en una revista astronómica profesional Wilson. 2009, Weinreb 2012.
En los años siguientes, la tecnología de radio tuvo un gran progreso, debido al interés en el desarrollo de equipos de radar para la Segunda Guerra Mundial. Cuando terminó la guerra, las mejoras en la tecnología de los receptores permitieron la creación de varios observatorios de radio .
Durante las últimas tres décadas, la radioastronomía ha evolucionado significativamente hasta convertirse en una rama muy importante de la astrofísica moderna.

2.3. Necesidad de Bajo Ruido

Cada objeto en el Universo a una temperatura superior a 0 K irradia energía en forma de ondas electromagnéticas (Electromagnetic (EM)) Rybicki y Lightman. 1979.La potencia emitida por esta radiación se puede calcular por el brillo del objeto, B_f, que cuantifica la potencia radiada por ancho de banda Hz, por área de superficie en un ángulo sólido. Si se considera el caso idealizado de un cuerpo negro, es decir, un objeto que absorbe toda la radiación EM incidente independientemente de su frecuencia y ángulo de incidencia, en equilibrio termodinámico con su entorno, su brillo se puede expresar como:

$$B_f = \frac{2hf^3}{c^2} \frac{1}{e^{hf/\kappa T} - 1} \tag{2.1}$$

Esta es la ley de Planck para la radiación de Cuerpo Negro Wilson. 2009, donde h es la constante de Planck, c es la velocidad de la luz en el vacío y κ es la constante de Boltzmann. Con esto se puede observar que la distribución de brillo solo depende de la temperatura termodinámica del objeto (T), y de la frecuencia (f). Esto permite la representación que se muestra en la Figura 2.2 .

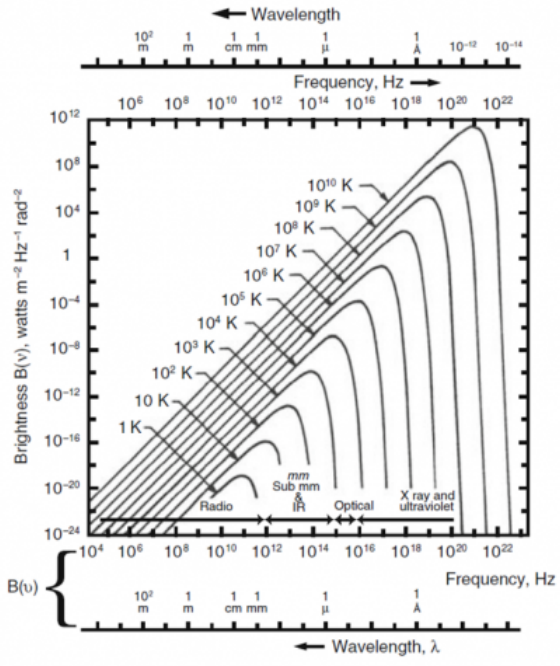


Figura 2.2: Espectro de Planck para cuerpos negros a diferentes temperaturas termodinámicas del cuerpo.Créditos imagen Wilson. 2009.

Dependiendo de la relación entre $h\nu$ y κT, la fórmula de Planck para la radiación de cuerpo negro puede aproximarse mediante expresiones más simples. En particular, en las ventanas de radio y microondas del espectro EM, se aplica a la llamada región de Rayleigh-Jeans [1] . Dentro de esta región el término hf es mucho menor que κT (hf $<<$ κT) Kraus 1986, y se puede utilizar la siguiente aproximación :

$$e^{hf/\kappa T} - 1 \approx 1 + \frac{hf}{\kappa T} - 1 = \frac{hf}{\kappa T} \tag{2.2}$$

Con esta aproximación, la temperatura de brillo se puede reescribir como la ley de radiación de Rayleigh-Jeans :

$$B_f = \frac{2hf^3}{c^2}\frac{\kappa T}{hf} = \frac{2\kappa T}{\lambda^2} \tag{2.3}$$

Cuando un radiotelescopio apunta al cielo y observa una fuente, recibe una densidad espectral de potencia (w), dada por :

$$w = SA_e = B_f \Omega_a A_e = \frac{2\kappa T}{\lambda^2}\lambda^2 = 2\kappa T \tag{2.4}$$

donde S es la densidad de flujo, A_e es el área colectora del radiotelescopio y Ω_A es el ángulo sólido del haz del telescopio. En [62] [46] se demuestra que $A_e \Omega a = \lambda^2$ (suponiendo que $\Omega_S > \Omega_A$, donde Ω_S es el ángulo sólido de la fuente; es decir, la fuente es más grande que el haz del telescopio). La potencia total (P) recibida por el telescopio en una fracción de ancho de banda B se puede calcular como:

$$P = wB = 2\kappa T B \tag{2.5}$$

Las fuentes de radioastronomía suelen no estar polarizadas o parcialmente polarizadas Condon y Ransom 2016 . Para mejorar aún más la eficiencia general del sistema, los receptores de radioastronomía suelen incluir un polarizador de entrada con una rejilla de alambre de espacio libre u transductor ortomodo (OMT), para medir ambos componentes de polarización de la señal. Para el caso más simple de una señal no polarizada, cuando llega al polarizador, la mitad de la potencia lo atraviesa

[1] La ubicación de este región depende de la temperatura: cuanto más caliente es el objeto, más se extiende la región de Rayleigh-Jeans hacia longitudes de onda más cortas.

y la otra mitad es rechazada. En este caso, cada canal recibe la siguiente potencia

$$P_{canal} = wB = \kappa TB \tag{2.6}$$

En el caso de que T >> $\frac{hf}{\kappa}$ (como suele ocurrir en el rango de ondas milimétricas, donde 1,44 K < $\frac{hf}{\kappa}$ < 14,40 K para 30 GHz < f < 300 GHz), la aproximación de Rayleigh-Jeans no es apropiada, en cuyo caso la Ecuación 2.4 define una temperatura equivalente T_{RJ}, que está relacionada con la temperatura de la fuente T a través de:

$$T_{RJ} = \frac{hf}{\kappa} \frac{1}{e^{hf/\kappa T} - 1} \tag{2.7}$$

En radioastronomía la unidad de densidad de flujo es el Jansky (Jy) Wilson. 2009. Esta unidad cuantifica la potencia de la señal disponible en el telescopio y se define como:

$$1J = 10^{-26} \frac{W}{m^2 \cdot Hz}(SI) = 10^{-23} \frac{erg}{s \cdot cm^2 \cdot Hz}(cgs) \tag{2.8}$$

Los radioastrónomos suelen realizar observaciones que producen niveles de flujo de mili-Janskies o decenas de micro-Janskies.Para dar una estimación del brillo producido por una fuente astronómica y entender la importancia del ruido del sistema, se va a considerar un objeto que produce un flujo de 1 Jy. Si este objeto se observa con una antena Vertex de ALMA de 12 metros, que tiene un efecto de área de recolección de aproximadamente 113 m^2, la densidad de la potencia recibida por el telescopio es:

$$w(1Jy, 12m^2 ALMA) = 1{,}13 \times 10^{-24} W/Hz \tag{2.9}$$

y de la ecuación 2.6 la densidad de potencia en cada canal del receptor es la mitad de ese valor:

$$w_{canal}(1Jy, 12m^2 ALMA) = 0{,}565 \times 10^{-24} W/Hz \tag{2.10}$$

El ruido del sistema se expresa normalmente en términos de su temperatura equivalente (T_{SYS}). Si se utiliza un sistema con una temperatura de ruido de 30 K para observar esta fuente, la densidad de potencia de ruido en la entrada sería:

$$w_n = \kappa T_{SYS} = 4{,}14 \times 10^{-22} W/Hz \tag{2.11}$$

que es superior al nivel de potencia calculado para el caso de flujo de 1 Jy. Sin embargo, debido a que el ruido es una señal aleatoria con un nivel medio de cero, si la señal de salida del receptor es integrada en el tiempo, el ruido se puede promediar y la señal de la fuente se revela encima de él. Además, dado que las señales recibidas suelen tener un espectro razonablemente plano, la sensibilidad del sistema también se puede mejorar aumentando el ancho de banda de integración. Esto se refleja en la siguiente relación Wilson. 2009:

$$\triangle T = \frac{T_{SYS}}{\sqrt{B \times \tau}} \tag{2.12}$$

donde $\triangle T$ es la raíz cuadrada media (rms) de la fluctuación del ruido, τ es el tiempo de integración y B es el ancho de banda de integración. Las señales detectables más débiles son unas pocas veces (típicamente cinco) la fluctuación del ruido en rms de salida [9] . De la Ecuación 2.12 queda claro que minimizar el ruido del sistema es una prioridad, particularmente porque el tiempo de integración y el ancho de banda son a menudo parámetros establecidos por restricciones de observación, como la necesidad de resolver características espectrales o la escala de tiempo de la estabilidad atmosférica o instrumental.

El ruido del sistema es la suma del ruido de la antena y el ruido del receptor, y ambos se expresan típicamente en términos de su temperatura de ruido equivalente (T_A) y (T_{RX}):

$$T_{SYS} = T_A + T_{RX} \tag{2.13}$$

El ruido de la antena se debe a varias fuentes, como el ruido del suelo, el ruido atmosférico o las pérdidas en la alimentación electrónica.

La temperatura de ruido de un receptor corresponde a la temperatura de una carga perfectamente adaptada en su entrada que da lugar a una cantidad equivalente de ruido en un receptor sin ruido. Se puede calcular con la ecuación de trasmisión de Friis:

$$T_{RX} = T_1 + \frac{T_2}{g_1} + \frac{T_2}{g_1 g_2} + ... + \frac{T_2}{g_1 g_2 ..... g_{n-1}} \tag{2.14}$$

donde T_n y g_n son la temperatura de ruido y la ganancia de cada componente (n) en la cadena del receptor.
De la Ecuación 2.14 queda claro que los primeros elementos en la cadena del receptor, que corresponden a valores más bajos de n, generan mayor influencia en el ruido del receptor.

Por lo general lo que se busca para minimizar la temperatura de ruido del receptor, es que estos componentes presenten bajo ruido y una alta ganancia.

2.4. Calibración de Receptor

La estructura de un receptor frontal en radioastronomía tiene una gran influencia en el ruido del sistema. En un sistema típico, la señal de radiofrecuencia (RF) entrante es recolectada por el telescopio y acoplada a la bocina de alimentación en la entrada del receptor a través de la disposición óptica del telescopio. Como se describe en la sección anterior, esta señal es muy débil y comparable al ruido del receptor.

El procedimiento de calibración es una parte importante del análisis instrumental. Ya que una calibración incorrecta puede causar un sesgo significativo de los resultados analíticos del valor real de los datos.

Para la calibración del receptor se desea obtener la temperatura empírica de ruido asociada a este, y encontrar una relación entre potencia y temperatura, esto permitirá calibrar los datos obtenidos.

2.4.1. Método del Factor-γ

Una técnica utilizada en los radio telescopios para medir la temperatura del sistema es el método del "Factor γ". Este método emplea una carga fría y otra carga caliente de temperaturas efectivas conocidas y no incluye ningún efecto de la antena. Para obtener el ruido del receptor se utiliza una carga fría, que se asemeja a un cuerpo negro, a una temperatura conocida. Luego, esta carga se coloca sobre el receptor de elección, o se acopla de otro modo con él, y se mide el nivel de potencia.Se procede de la misma forma con la carga caliente, obteniendo los niveles de potencia a una temperatura conocida previamente configurada. Los valores de potencia medidos en cada carga se asocian a temperaturas físicas por medio de la ecuación 2.5. La

medición de la temperatura del sistema, utilizando una cargas frías y una calientes es considerablemente más confiable que otros método utilizados menos precisos.
Si la carga uno tiene una temperatura efectiva conocida T1 y la carga dos tiene una temperatura efectiva conocida T2, la relación de la potencia medida entre las dos cargas será:

$$Y = \frac{T_1 + T_{RX}}{T_2 + T_{RX}} \tag{2.15}$$

donde recordemos que T_R a estos efectos es la temperatura del sistema sin ningún efecto de la antena. Esta relación (el factor γ) se puede usar para obtener T_R a través de:

$$T_{RX} = \frac{T_1 - YT_2}{Y - 1} \tag{2.16}$$

De esta forma se puede obtener un calculo preciso de la temperatura del sistema y luego con esto obtener la sensitividad del receptor.

2.4.2. Calibración de Datos

La calibración de los datos obtenidos por el radio telescopio, se lleva acabo estableciendo una relación entre la potencia del receptor y la temperatura de las cargas de calibración.
La aproximación de Rayleigh-Jeans, para longitudes de onda de radio, proporciona una relación lineal simple entre la potencia medida y la temperatura de la superficie de los dispositivos de calibración.
Para esto se requiere ajustar una ecuación lineal entre los datos de potencia y los de temperatura.Lo que se puede representar como:

$$P = a * T + b \tag{2.17}$$

donde los parametros a y b representan la pendiente y la ordenada de la recta respectivamente . El objetivo de la calibración sera determina estos dos parametros y así encontrar la ecuación de la recta, para la cual se necesitan al menos dos puntos.

En la práctica se obtienen los registros de la lectura de potencia del receptor

cuando incide radiación de dos cargas de calibración con temperaturas físicas conocidas . Las cargas se colocan a la entrada del receptor, bloqueando toda emisión de fuentes externas. La lectura a la salida del receptor corresponderá entonces a la temperatura del cuerpo negro más la del sistema receptor.
Haciendo esto con diferentes temperaturas (ver figura 2.3) se puede definir exactamente los parámetros a y b de la ecuación 2.17.

La temperatura de las cargas utilizadas para calibrar el receptor suelen estar a temperatura ambiente, ya que se utiliza el punto de ebullición del nitrógeno, la carga que se considera una carga caliente y otra a una temperatura mas fría,pues se utiliza el punto de ebulliciń del nitrógeno líquido o del helio, que son 78 K y 4.2 K respectivamente Basoalto. 2018.

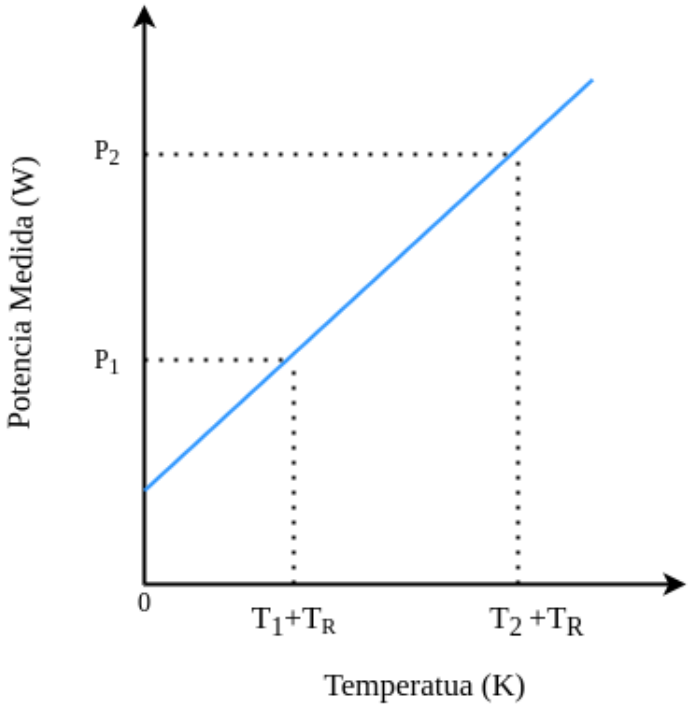


Figura 2.3: Relación Lineal de temperatura adquirida versus potencia.

En el radio telescopio LLAMA se utilizaran dos cuerpos negros para poder realizar el proceso descrito anteriormente.Una carga fría que estará a temperatura ambiente y una carga caliente que estará operando a una temperatura alrededor de 60 °C.

2.5. Cargas de Calibracion para LLAMA

El desarrollo de los prototipos de calibración para el radiotelescopio ha involucrado la participación de distintas partes, cada una abocada a alguna tarea en particular, necesaria para el desarrollo de los instrumentos y para efectuar las pruebas radiometricas y de simulación necesarias para su fabricación.

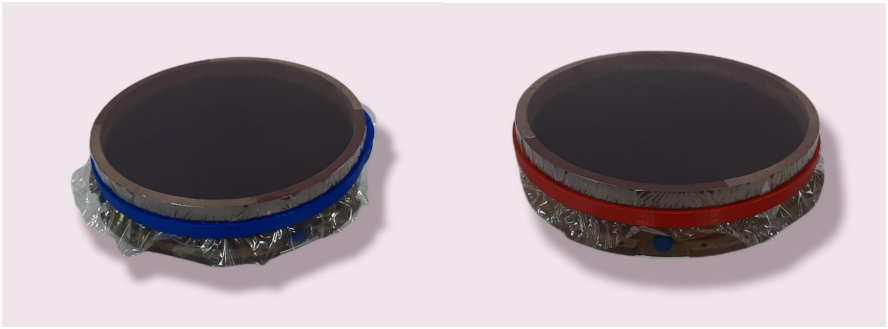
Dada la configuración dradiotelescopio de LLAMA, que cuenta con un amplio rango de frecuencias para la operacion de sus receptores, se requiere de tres tipos distintos de Cargas de Calibración:

- 1 carga fabricada con tecnología TK-RAM grandes trabajando a una temperatura de 70 °C que cubrirá las bandas B1 a B3 (31 GHz a 116 GHz).

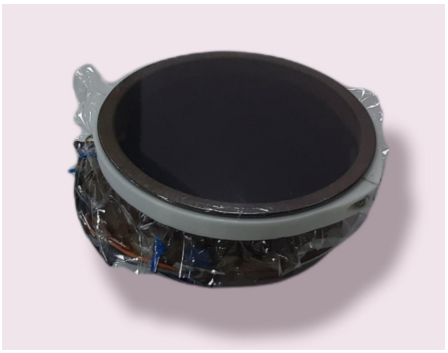
- 1 carga fabricada de TK-RAM pequeñas operando a una temperatura de 70 °C en las bandas B3 a B9 (84 GHz a 720 GHz).

- 1 carga fabricada de TK-RAM grandes operando a temperatura ambiente que cubrirá ambas frecuencias.

La tecnología TK-RAM fabricada de polipropileno con la cual se han fabricado las cargas, permite trabajar en un rango de frecuencia entre los 50 GHz a 1 T Hz, con una reflexión de dispersión muy baja Basoalto. 2018, pudiendo ser utilizadas como cuerpos negros en las frecuencias del milimétrico/submilimétrico.



(a) Carga Grande Fria. (b) Carga Grande Caliente.



(c) Carga Chica Caliente.

Figura 2.4: Prototipos de Cargas de Calibración desarrolladas por CePIA.

2.5.1. Requerimientos Operacionales

Desde el punto de vista de los requerimientos operacionales y de control de las CC, se consideran algunos aspectos exigidos por LLAMA, que son presentados a continuación:

1. **Despliegue de Dispositivos en ACS**
   Las CC deben ser controladas por medio del software de ACS. Esto implica que el sistema de control de las cargas, se encuentre integrado al software y pueda ser desplegado remotamente por medio de la lógica de comunicación distribuida del software.

2. **Base de Datos**
   Los datos de temperatura de los dispositivos deben estar integrados a la base de datos común del software de ACS.

3. **Sistema de Monitoreo**
   Es importante que la superficie de cada carga de calibración mantenga una uniformidad de temperatura sobre las piezas RAM, con una desviación estándar menor a 1°C. Cada carga cuenta con tres sensor distribuidos uniformemente sobre la placa de aluminio de cada carga.
   El sistema debe proporcionar un monitoreo de los tres sensores por separado de cada carga, en tiempo real y en caso de que algun sensor falle, alertar sobre las fallas en el software del telescopio.

4. **Sistema de Control**

   El sistema debe permitir establecer un punto de temperatura para cada carga, dentro de un rango de temperatura de 0 °C a 70 °C, además de poder activar y desactivar cada instrumento. Incluyendo una señal de parada, que permita desactivar el sistema por completo.

5. **Visualización en Tiempo Real**. Incluir una interfaz gráfica para controlar los tres dispositivos de calibración.
   Proporcionando gráficos de temperatura versus tiempo para cada uno de los sensor.

Capítulo 3

Descripción Tecnología de Software

El sistema de software ALMA está diseñado para transformar la entrada del usuario en una propuesta de observación, realizando los procedimientos específicados para entregar datos calibrados (imágenes y espectros) al usuario y a un archivo del observatorio. A partir de la información retenida en este archivo, no solo el observador original, si no también otros astrónomos interesados, pueden realizar más investigaciones y análisis de los datos.
El software gestiona y administrar la preparación de propuestas, las operaciones en los instrumentos y el uso de archivos para la calibración del proyecto científico.

La primera parte de este capitulo presenta la física dentro de las antenas de ALMA, utilizadas para captar señales de onda milimétricas y submilimétricas de radio con objetivos científicos, y como esto marcan el diseño lógico del sistema para relacionar el flujo de datos de extremo a extremo, que se utiliza en las instalaciones del telescopio .

Se describe la arquitectura funcional del software de ACS dentro del telescopio LLAMA, explicando los subsistemas involucrados en las operaciones de observación. Esto para definir la abstracción del software en términos de interacciones comunes entre los componentes y los elementos estructurales del software.

La segunda parte de este capitulo presenta una descripción general de los con-

ceptos fundamentales del subsistema de control centrándose en los criterios que se utilizan al momento de integrar instrumentos en el software.
Con una descripción de la arquitectura técnica, enfocada en un modelo de capas, constituido por diversas tecnologías de software como aplicaciones e interfaces de datos, bibliotecas y clases de objetos para la comunicación (transmisión, subprocesamiento, activación y transacciones, acceso remoto) , que interactúan entre si como base para el desarrollo de ACS LLAMA.

3.1. Introducción

3.1.1. El Telescopio ALMA

El Atacama Large Millimeter/Submillimeter Array (ALMA) es un radio telescopio, que consta de 66 antenas de alta precisión (54 antenas de 12 mt y un arregle de 12 antenas de 7 mt, el Atacama Compact Array (ACA)), esta diseñado para funcionar como un interferómetro utilizando técnicas de síntesis de apertura, mediante configuraciones dinámicas en su arreglo de antenas, con líneas de base (distancias entre dos antenas) que van desde 15 m hasta $\sim$ 16 km.
El brillo o la intensidad especifica I_ν captada por una antena puede definirse como la potencia δP dentro de un rango de frecuencia (ancho de banda) captada dentro de un ángulo solido $\delta\Omega$, que es interceptado por una superficie de área δA . Por lo tanto la potencia recibida, considerando un receptor polarizado, de área efectiva A_e, es la siguiente:

$$P_{rec} = \frac{1}{2} I_\nu A_e \delta\Omega \tag{3.1}$$

Las variables angulares θ y φ corresponden a direcciones ortogonales o coordenadas del cielo, $I_\nu(\theta, \varphi)$ y $P_N(\theta, \varphi)$ pueden definirse como las funciones direccionales del brillo del cielo y la potencia de la antena normalizada respectivamente. La potencia total recibida de una antena en un punto dado, es la integración sobre el cielo del producto de la distribución del brillo del cielo y la respuesta de potencia de la

antena:

$$P_{rec} = \frac{1}{2} A_e \int_{4\pi} I_\nu(\theta, \varphi) P_N(\theta, \varphi) \delta\Omega. \tag{3.2}$$

Mientras que el ángulo solido de la antena, queda definido por:

$$\Omega_A = \int_{4\pi} P_N(\theta, \varphi) \delta\Omega. \tag{3.3}$$

El ancho de potencia media (Full Power Beam Width (FPBW)) se considera la resolución de Rayleigh[19] de la antena, es decir, su capacidad para distinguir objetos en el cielo separados por cierta distancia angular. Las longitudes de onda de radio recibidas en forma de un frente plano y paralelo, están esencialmente limitadas por la difracción generada por la geometría de la antena.
El disco angular o FPBW para una apertura circular de diámetro D es $\sim 1{,}22\lambda/D$, es por esto que la resolución en ondas milimétricas captadas por una antena, puede llegar a ser mucho menor que la de un telescopio óptico.
Alternativamente, varios pares de antenas, considerando cada antena como un elemento, pueden distribuirse en posiciones a distancias mucho mayores de lo que es posible construir con un solo telescopio de apertura total, donde las señales recibidas por las antenas pueden combinarse en fase para aproximar el poder de resolución total al deun solo telescopio de mayor diámetro, mediante técnicas como la sintesis de apertura . La figura 3.1 muestra un esquema geométrico de un interferómetro con dos antenas separadas por una distancia b, apuntando hacia una misma fuente de brillo *so* en el cielo . La emisión recibida por la antena 1 experimenta un retraso geométrico con respecto a la antena 2, con un tiempo de retraso igual a $\tau_g = bs_o/c$.

Para el caso más general en el que el frente electromagnético incide fuera del eje de la antena, con un ángulo α, tendrá una posición en el cielo dada por $l = \sin\alpha$.

Para compensar el retraso geométrico, se puede insertar un retraso artificial (electrónicamente) en la ruta de la señal de la antena 2, para que las señales de ambas antenas puedan ser correlacionadas con la misma fase.

La diferencia de fase puede ser caracterizada considerando la longitud de la trayectoria adicional como $x = u \sin\alpha = ul$ y relacionando la respuesta de la antena 2,

V_2 en termino del voltaje de la antena 1.El V_1 tiene un factor de retardo de fase que varia sinusoidalmente en función del ángulo α:

$$V_2 = V_1 e^{2\pi i(ul)} \tag{3.4}$$

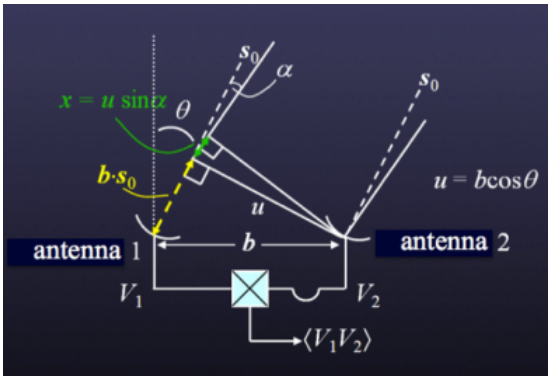


Figura 3.1: Un interferómetro en una dimensión ideal que consta de dos antenas, 1 y 2, separadas por una distancia física (es decir, una línea de base) b. Ambas antenas apuntan hacia una ubicación en el cielo dada por s_o , que está en un ángulo desde el meridiano. La distancia proyectada entre las dos antenas en esa dirección es por lo tanto $u = b\cos$. Las dos antenas están conectadas a un correlador donde se combinan los voltajes detectados de cada una. Crédito de imagen Allen Farris 2009.

Considerando una fuente en el cielo con un sistema de dos dimensiones, se introduce β con una dirección ortogonal a α , esto genera una línea de base que corresponde a un vector con componentes en ambas dimensiones, es decir, b_1 y b_2 .Definiendo $u = b_1 \cos\theta$ y $v = b_2 \cos\varphi$, donde φ es el ángulo de la posición s_o en el cielo en un sistema de referencia ortogonal a θ.Se puede considerar la longitud de trayectoria adicional introducida en esta nueva dirección, con unidades de longitud de onda de emisión, $y = v \sin\beta = vm$, con una respuesta de potencia igual a:

$$V_2 = V_1 e^{2\pi i(ul+um)} \tag{3.5}$$

Donde u y v se identifican como componentes de frecuencia espacial específicos de la sinusoide en las direcciones Este-Oeste y Norte-Sur.
Las señales trasmitidas a la supercomputadora que se encarga de correlacionar las señales, el correlador de ALMA, actúa como un dispositivo multiplicador y promediador de tiempo para las señales entrantes de las antenas 1 y 2. Por lo tanto, su

salida es:

$$\langle V_1 V_2 \rangle = \langle \int\int V_1(l,m) dl dm \int\int V_2(l,m) dl dm \rangle \tag{3.6}$$

Bajo el supuesto de que las señales que emanan de diferentes partes del cielo son incoherentes, es decir, no tienen similitudes en fase, los promedios de tiempo de la correlación de esas señales serán cero. Por lo tanto, el producto de las integrales en la ecuación 3.6 se puede simplificar a:

$$\begin{aligned} \langle V_1 V_2 \rangle = & \quad \langle \int\int V_1(l,m) V_2(l,m) dl dm \rangle \\ \langle V_1 V_2 \rangle = & \quad \int\int \langle V_1(l,m) V_2(l,m) \rangle dl dm \\ \langle V_1 V_2 \rangle = & \quad \int\int \langle V_1(l,m)^2 \rangle e^{2\pi i(ul+um)} dl dm \end{aligned} \tag{3.7}$$

Como $V^2 \propto P$ y $P \propto I_v$ (ver Ecuación 3.2),

$$\langle V_1 V_2 \rangle \propto \int\int I(l,m) e^{2\pi i(ul+um)} dl dm \tag{3.8}$$

donde I(l,m) es la distribución de intensidad en el cielo. Por lo tanto, el correlador mide una cantidad conocida como visibilidad compleja, ν, que es formalmente la transformada de Fourier de la distribución de intensidad en el cielo:

$$\nu(u,v) \propto \int\int I(l,m) e^{2\pi i(ul+um)} dl dm = A e^{i\phi} \tag{3.9}$$

En la práctica, el HPBW del haz principal sirve como campo de visión de la imagen interferométrica en un solo punto. Además, A(l, m), la función de Airy, está incluida formalmente en el salida del correlador:

$$\nu(u,v) = \int\int A(l,m) I(l,m) e^{-2\pi i(ul+vm)} dl dm \tag{3.10}$$

De este modo el interferómetro mide la transformada de Fourier de la distribución del brillo del cielo, multiplicada por la respuesta de potencia de la antena.

Las señales interferidas corresponden a distribuciones de brillo, que pueden ser muestreadas en escalas angulares muy pequeñas.Con imágenes con resoluciones an-

gulares extremadamente detalladas y finas, llegando a los 0.6 arco-segundos .
Los datos producidos por el correlador son del orden de un Gbyte por segundo y deben ser trasmitidos a una velocidad promedio de 4/40 mbyte/s. La reducción final de las imágenes obtenidas por el telescopio aumenta esta cifra casi en un 50 % alrededor de 6/60 Mbytes/s, esto implica $\sim$ 180 Tbyte por año de datos que deben ser procesados y archivados Schwarz 2004 .

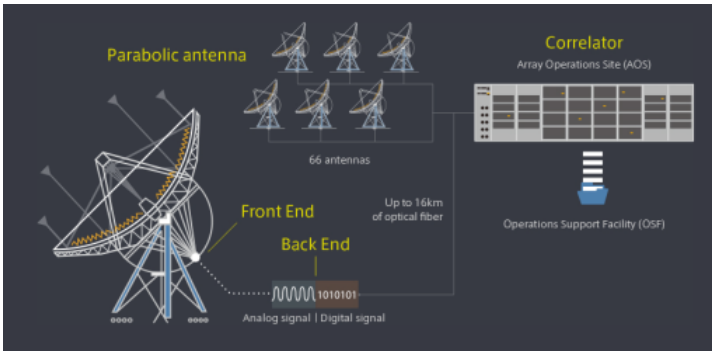


Figura 3.2: Una vista esquemática muestra el camino seguido por una señal astronómica una vez que ingresa a una antena de ALMA. La señal primero es recolectada por la antena, después de lo cual es enfriada criogénicamente en el Front End, luego digitalizada por el Back End y transmitida a través de fibra óptica hacia el edificio central donde el correlador combina la señal de todas las antenades ALMA para simular un radiotelescopio del tamaño de la matriz entera. Crédito de imagen NAOJ [10].

Las operaciones de control y monitoreo de las antenas, requiere de una calibración de datos en tiempo real, es decir se debe calcular la corrección en la orientación, el enfoque en la corrección de fase y el ruido de fase promedio, para luego enviar estos resultados al proceso de observación en $\sim$ 0.5 s, realizando ajustes .Además ALMA utiliza sofisticados modelos atmosféricos, estaciones de monitoreo climático y radiómetros para medir la cantidad de vapor de agua presente en la línea de visión de cada antena en tiempo real, con el fin de corregir efectos atmosféricos no deseados.
El sistema de ALMA maneja alrededor de 4000 dispositivos o ensamblajes cada uno asociado a distintos puntos de monitoreo y control Allen Farris 2009.

Otro de los grandes desafíos del observatorio es que su funcionamiento es altamente distribuido, con instalaciones asociadas al proyecto que se encuentran en dis-

tintas locaciones geográficas. Desde las instalaciones del interferómetro, en el valle de Chajnantor (5000 mt), hasta la instalación de apoyo a las operaciones (Operations Support Facility (OFS) por sus siglas en ingles) en San Pedro de Atacama o las oficinas centrales en Santiago de Chile y los centros de ALMA en América del norte y Europa. Cada locación se encarga de llevar acabo distintas labores, contribuyendo al funcionamiento del observatorio.

Los sistemas informáticos distribuidos con cómputo paralelo masivo de datos a gran tamaño como el utilizado por ALMA, son por lo general inisuales y requieren de arquitecturas modulares y flexibles que puedan adaptarce a distintas tecnlogias. .

3.1.2. Software de ALMA

El software de ALMA, debe encargarse de resolver todas las fases que un proyecto de observación pueda requerir, esto incluye: preparación de propuestas, planificación dinámica, control de instrumentos, manejo y formateo de datos, archivo y recuperación de datos y la calibración, además de dar soporte en las operaciones relacionadas con el observatorio.
El flujo de datos dentro del observatorio una vez iniciado un proceso de observación, puede visualizarse en la figura 3.3.
Los astronomos inician el ciclo creando y enviando una propuesta en tiempo de observación a ALMA en la Fase I .Para esto se utiliza la Herramienta de Preparación de Observación, que puede ser ejecuta en el escritorio de cada astronomo, definiendo las configuraciones de parametros físicos, en términos de las líneas deseadas o las frecuencias de observación, con los anchos de banda de la ventana espectral y sus resoluciones.

Por lo tanto,se consideran para la etapa de preparación de la observación una configuración de paquetes con instrucciones, que interactúan de forma estrecha con la herramienta de observación que se encuentra en el centro,de esta unidad lógica siendo independiente del resto del sistema de software.
En un modo de operación automatico cada bloque de observación que sea generado sera clasificado de forma dinámica por el planificador o programador, que se encargara de ejecutar los procedimientos contenidos en el bloque y de enviar los comandos a los subsistemas de control de la antena y el receptor para realizar las observaciones .

Las propuesta revisadas y aceptadas, se convierte en un Programas de Observación y pasan a la Fase II , donde las observaciones reales se configuran completamente, a través de las especificaciones que hayan sido incluidas en el Bloque de Planificación (Scheduler Block (SB)). Una vez que se ha definido cada SB, se almacena en el Archivo, para luego llevar acabo el proceso descrito por el bloque (siempre que las condiciones de observación estén disponibles).
Luego de ser inicializado el subsistema de Operaciones de Instrumentos de ALMA se elegirá y ejecutará/observará el SB que este listo para funcionar . La ejecución de una SB genera un conjunto de datos sin procesar, que se guardan en el Archivo y se envían al Canal de calibración (en el caso de los datos que se utilizarán para la calibración) y/o al Canal de reducción para su procesamiento.

Los procesos del canal de Calibración, entre otras cosas, se utilizan para corregir el enfoque, la orientación y datos del calibrador de fase, y así enviar los resultados en primer lugar al sistema ALMA para modificar los parámetros de observación casi en tiempo real y en segundo lugar el programador dinámico para permitirle seleccionar el próximo SB a observar.

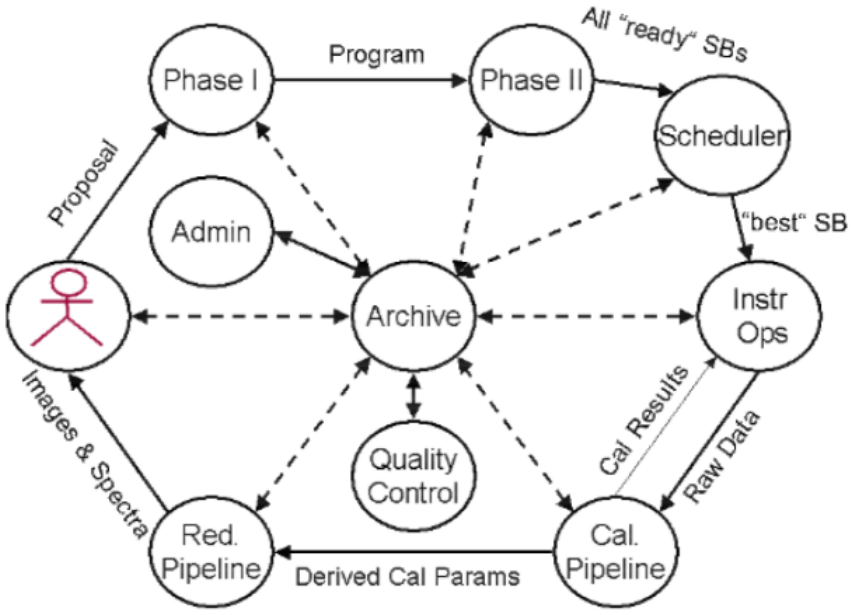


Figura 3.3: Flujo de datos del sistema ALMA (esquema). Las líneas continuas exteriores muestran el flujo de datos lógicos; las líneas discontinuas dirigidas hacia/desde el Archivo indican que a) todos los datos se guardan y se pueden recuperar desde el Archivo; y b) que el flujo de datos lógicos puede, cuando corresponda, ser manejado por el archivo .Creditos a ALMA [1].

Los conjuntos de datos interferométricos de radio modernos a menudo incluyen cientos o miles de observaciones distintas.Estos resultados son guardados en el Archivo para ser utilizados por la cadena de reducción de datos (Reduction Pipeline). Donde se produce la vista rápida y las imágenes y/o espectros finales, aplicando las correcciones y filtros que sean necesarios.

Actualmente los observatorios, especialmente ALMA, han logrado grandes avances hacia la calibración automatizada y de alta calidad de datos interferométricos.Las tuberías interferométricas y de energía total de ALMA , se basan en el proyecto de software CASA (Common Astronomy Software Applications, que permite la calibración y deconvolución de estos datos para producir imágenes correctas del cielo. Gracias a estos esfuerzos, ALMA entrega datos de visibilidad (u- v) bien calibrados a sus usuarios.

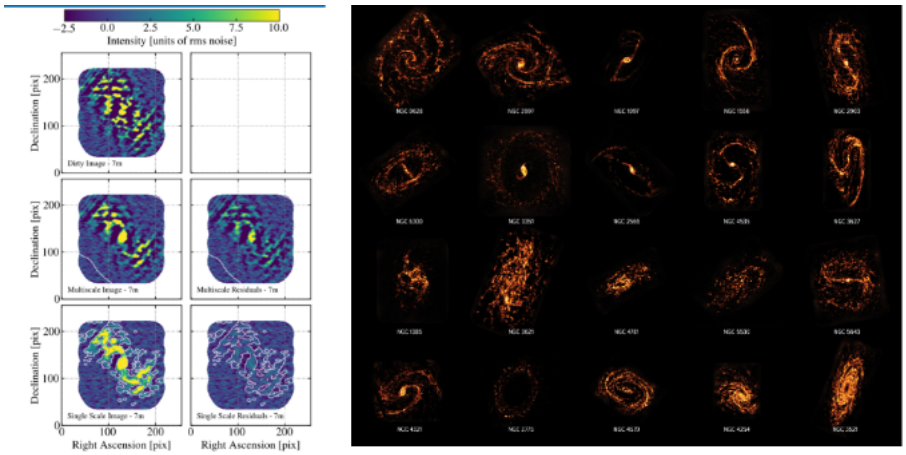


Figura 3.4: Se presenta un ejemplo de desconvolucion de imágenes conseguidas a través de la matriz de antenas de 7 m de ALMA con PHANGS (Física a Alta Resolución Angular en Galaxias Cercanas), para la reducción de datos utilizando CASA , con el objetivo de obtener mapas de CO (2-1) en galaxias cercanas.La imagen de la izquierda muestra dos columnas en distintas etapas de limpieza y filtrado, la primera columna se puede visualizar las imágenes limpias y la segunda el residuo del proceso de filtrado.La segunda imagen muestra el producto final de calibración y reducción de un grupo de galaxias.Créditos Adam K. Leroy 2016.

Por ultimo, el astrónomo recibe los resultados del proyecto de observación en forma de imágenes y/o espectros calibrados junto con la información asociada (p. ej., registros).
La estructura general del sistema de software ALMA se centra en 5 subsistemas de nivel superior con las siguientes responsabilidades:

1. Preparación de la observación
2. Operación del instrumento
3. Reducción de datos científicos
4. Archivo
5. Administración

Estos subsistemas utilizan propiedades del sistema ALMA que marcan en profundidad el diseño del software, en términos de tasas de transacción, incluyendo:

- Tasas de adquisición de datos muy altas. El correlador de línea de base de ALMA produce del orden de 1 Gbyte/s de datos, el subsistema del correlador debe encargarce de reducir esto en tiempo real, dejando que el resto del sistema se ocupe de los datos a velocidades de 6 a 60 Mbyte/s más pequeñas, pero aún significatibas.

- Gran volumen de datos archivados (180 Tbyte/año).

- Líneas con canales de datos para, la reducción fuera de línea, la Calibración del telescopio y el Archivo (velocidad de datos: 10 MB/s - 300 TB/año).

- Sincronización en tiempo real de hardware a grandes distancias. Las antenas de ALMA se pueden extender en más de 10 a 15 km. El propio correlador puede estar ubicado a 50-80 km del sitio de la antena.

Debido a la masiva cantidad de datos que manejan los distintos subsistemas y por su naturaleza altamente distribuida, se emplean las especificaciones del servicio de trasmisión de Audio/Vídeo del software intermedio CORBA, que facilita la comunicación y otorga servicios distribuidos mediante especificaciones de la Object Request Broker (ORB) a los distintos componentes y subsistemas el software. Cumpliendo un rol fundamental para la sincronización de procesos internos del sistema.

3.2. Software de LLAMA

El software de LLAMA se basa en ALMA, con pequeñas modificaciones, particularmente considerando que LLAMA funciona como un radiotelescopios de plato único, por lo que, el módulo correspondiente al correlador no se considera.
La imagen 3.5, entrega una descripción general de los subsistemas que proporcionan el flujo de datos en el observatorio LLAMA. El esquema muestra las principales rutas funcionales dentro del sistema, para comprender de forma específica el papel del subsistema de control.

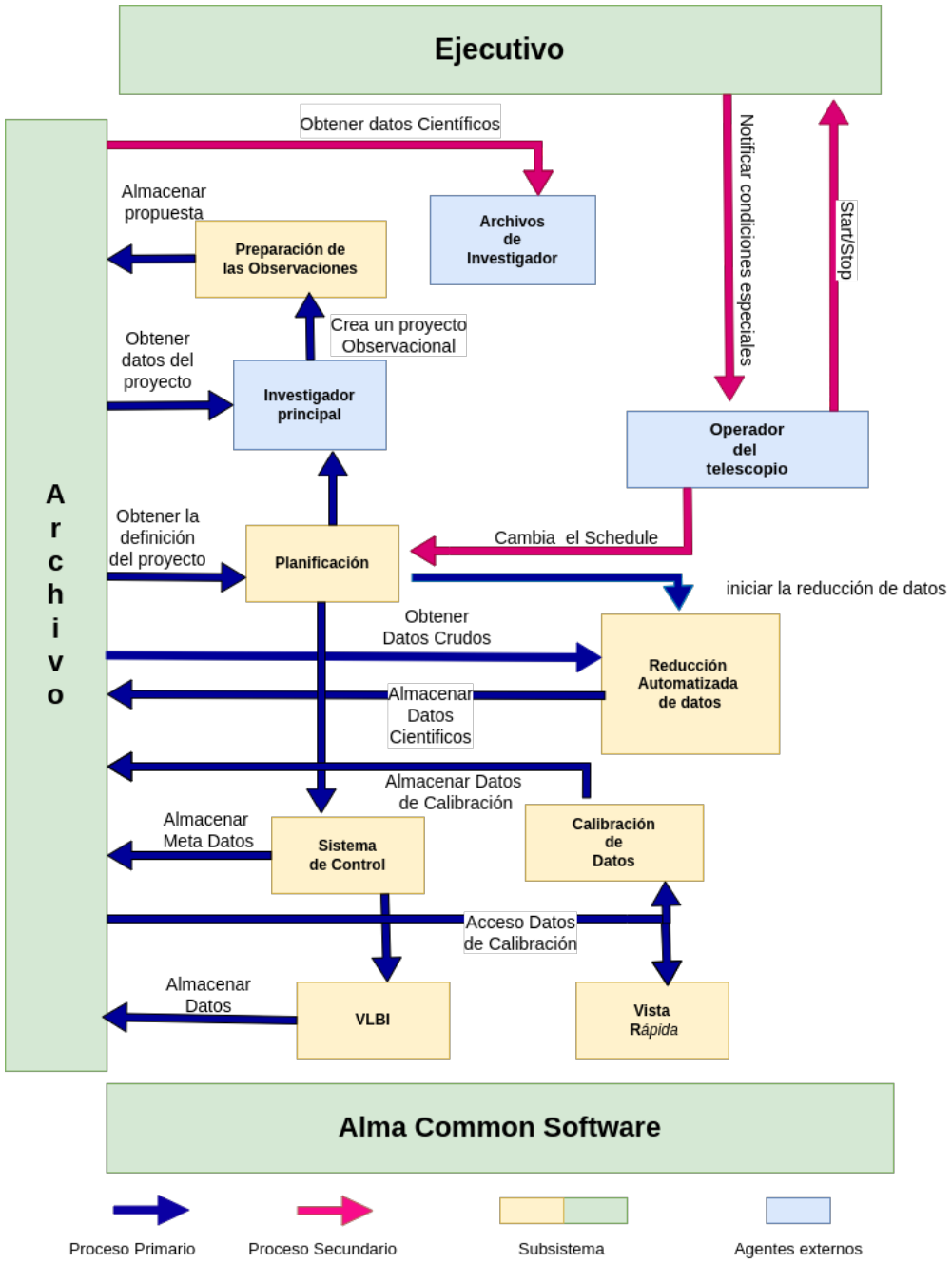


Figura 3.5: Sistema de Software de LLAMA desde la perspectiva de Control.

El proceso de observación real de LLAMA se implementa en las Operaciones de Instrumentos, que integra al subsistema de control.

El propósito del sistema de control en el radiotelescopio LLAMA es monitorear y controlar el telescopio, ejecutando bloques de programación de proyectos de obser-

vación científica, que puedan ser aprobados.Para Monitorear el telescopio se requiere de un seguimiento de su estado, poder diagnosticar problemas y reportar anomalías. Mientras que el control del telescopio, se encarga de inicializar y detener el software, ejecutando trabajos útiles, como observaciones, calibraciones y diagnósticos. El seguimiento y control del telescopio son actividades continuas, y pueden funcionar independientemente de si el telescopio este siendo ejecutando por alguna observación científica.
Toda la planificación de operación de instrumentos, depende de las condiciones del sistema, por lo que el planificador debe obtener información constantemente sobre las condiciones climáticas y el rendimiento de los instrumentos utilizados, múltiples interfaces que representan la lógica física de cada dispositivo. Una parte crucial del subsistema es el paquete de calibración, que proporcionará toda la funcionalidad para una calibración adecuada de telescopio/instrumento/datos.

El subsistema de software común (ACS) proporciona la unión entre todos los demás subsistemas. basándose en el middleware CORBA, pero ofrece muchas más funciones que facilitan la comunicación entre subsistemas, el manejo de errores, la recopilación de datos, etc.

El subsistema de Archivo contiene todos los servicios de datos e información, siendo necesario para muchos paquetes de otros subsistemas. Por lo tanto, las interacciones se proporcionan mediante consultas de archivos, catálogos, repositorios, bases de datos, recuperación de datos, etc.
Cada subsistema está representado por un Componente Maestro, que es responsable de la interacción con otros subsistemas, encargado de iniciarlos y detenerlos reportando su estado.

El flujo de datos se dirige desde el dominio del telescopio a través de mensajes en forma de eventos CORBA, al dominio científico con la interfaz de captura de datos, como puente entre los dos dominios . Esta interfaz es clave para separar estos dominios y todos sus subsistemas. Dentro del telescopio, se generan todos los datos obtenidos en las operaciones de control de las antenas. Dentro del dominio de la ciencia, el telescopio es una abstracción.Aquí se ajustan todos los datos, como los desplazamientos de orientación y enfoques, para ser utilizados como producto de la

calibración en el proceso de observación, en la vista rápida y posteriormente en el almacenamiento, como imágenes FITS en el archivo científico de LLAMA.

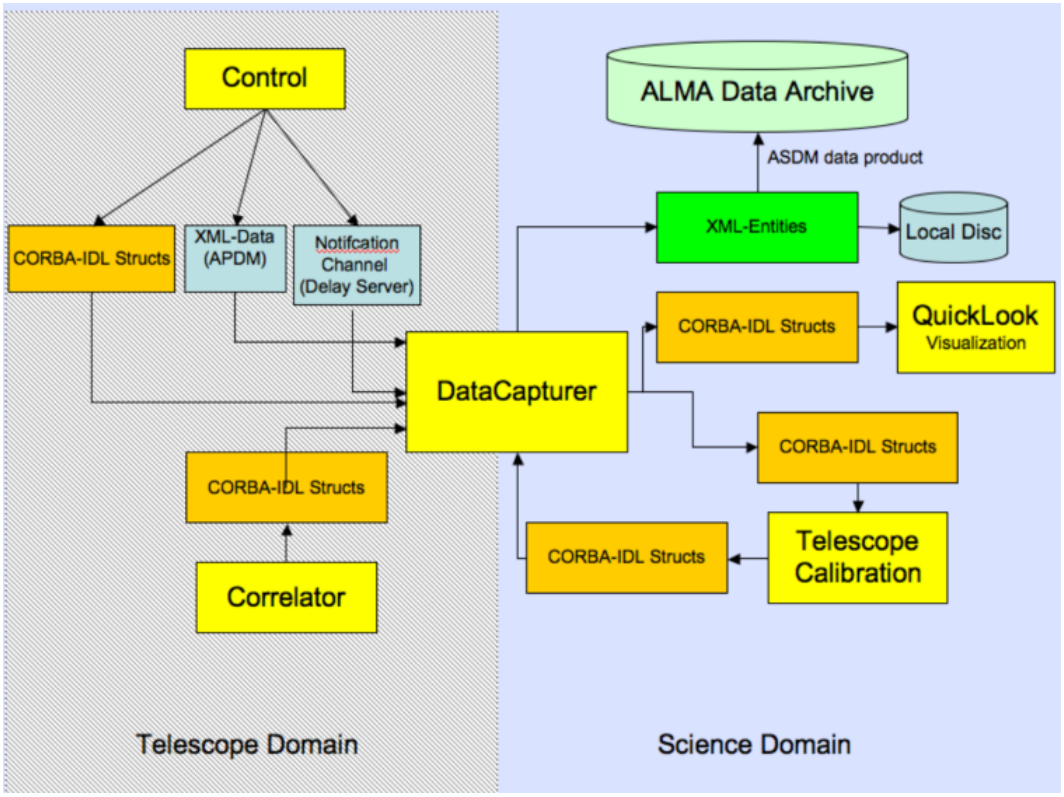


Figura 3.6: Representación gráfica del intercambio de datos mediante la interfaz de captura de datos, entre el dominio del telescopio y el de ciencia.Créditos de la imagen [19].

Para el subsistema de Control, el Componente Maestro es especialmente complejo. Su principal responsabilidad es iniciar y detener todo el hardware que conforma el telescopio. Dentro del sistema de Control, los datos generalmente se recopilan y estructuran en función de la perspectiva del instrumento para que puedan ser utilizados en las observaciones.
Al iniciar la configuración de la base de datos para la configuración del Monitor del telescopio (Telescope Monitor and Configuration Database (TMCDB)). Esta base de datos, se encarga del despliegue de las operaciones, conteniendo todos los datos de instanciación de los instrumentos y del sistema de alarma. Contiene una lista de hardware que deben ser desplegados y puestos en línea. De esta manera, el sistema

de control descubre qué hardware está realmente disponible para ser administrado.

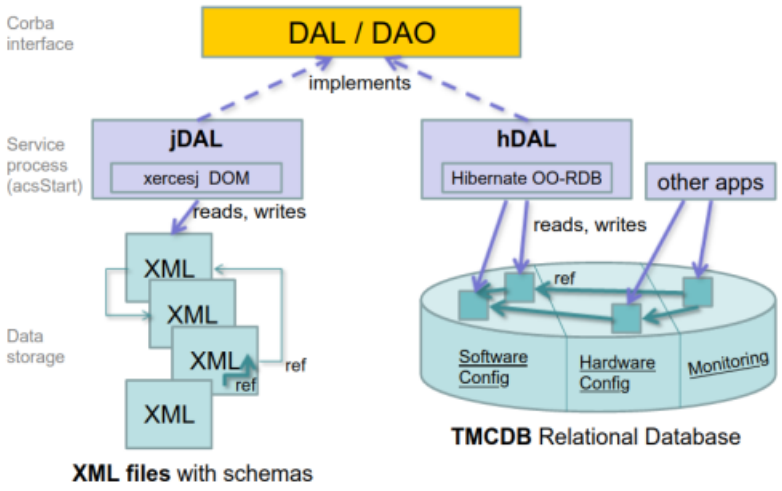


Figura 3.7: Se muestra la interacción de la TMCDB con el proceso de inicio del telescopio , utilizando CORBA como medio para la trasferencia de información.Créditos Sommer 2004.

El propósito de la base de datos de configuración y monitor del telescopio ALMA es proporcionar un sistema que:

- Almacene datos monitoreados durante la vida útil del telescopio ALMA.
- Almacene datos necesarios para configurar e inicializar todos los módulos de hardware y software necesarios para operar el telescopio.
- Rastrear la configuración del telescopio a lo largo del tiempo, incluido el almacenamiento de atributos de hardware y software planeados para ser incorporados.Proporcionar todos los datos necesarios para iniciar el telescopio.

También se encarga de determina si otros sistemas como la Calibración y los sistemas de Archivo están disponibles al momento deseado de ejecutar las operaciones, interactuando principalmente con el operador del telescopio .
De este modo el sistema de control, funciona bajo la supervisión de un operador que puede monitorear el telescopio, proporcionando una colección de dispositivos que funcionen de forma controlada, y ejecutando una secuencia de comandos de alto nivel que representen la intención para la observación científica .

3.3. Arquitectura ACS

ACS proporciona un marco para la arquitectura de objetos distribuidos, con clases de análisis fusionadas en paquetes que se utilizan desde, aplicaciones de más alto nivel, hasta el dominio de los subsistemas que controlan las antenas, los receptores y otros instrumentos.

El diseño de la arquitectura de software considera una separación técnica funcional, haciendo uso de un modelo de contenedor/componente, que facilita la infraestructura para las aplicaciones y la encapsulación del trabajo de desarrollo, y así permitir solo tener que concentrarse en el código específico de los componentes, como son los detalles de hardware y los algoritmos de la física que existe detrás del telescopio. Mientras la infraestructura técnica se encarga de la comunicación, la base de dato, la seguridad del sistema y otros aspectos técnicos.

La arquitectura de desarrollo de ACS, se basa en el concepto de programación de Objeto Distribuido (DO). Presentando un diseño de patrones comunes en capas, desarrollado especificamente para el mapeo de dispositivos físicos (mediante la serialización de objetos), en el caso del software de control o en otro tipo de objetos mas abstractos con aplicaciones distintas.

Los paquetes desarrollados dentro del software puede usar servicios proporcionados por otros paquetes, en capas inferiores como también de la misma capa, pero no en capas superiores.
Cada paquete proporciona un conjunto básico de servicios y herramientas que son utilizados por las aplicaciones de ACS. La figura 3.8 muestra en detalle el modelo para las herramientas de software.

La arquitectura para el desarrollo de ACS consta de 4 capas: la capa de presentación (interfaces de usuario), la capa de aplicación (que proporciona las tareas principales), la capa de dominio (objetos de uso común) y la capa del sistema (servicios del sistema y comunicación entre subsistemas).Un quinto grupo contiene paquetes de software que utilizan muchos subsistemas de ALMA, pero que no utilizan otros paquetes de ACS. Por conveniencia, estos paquetes están integrados y distribuidos junto con ACS, pero no son partes integrales de ACS.

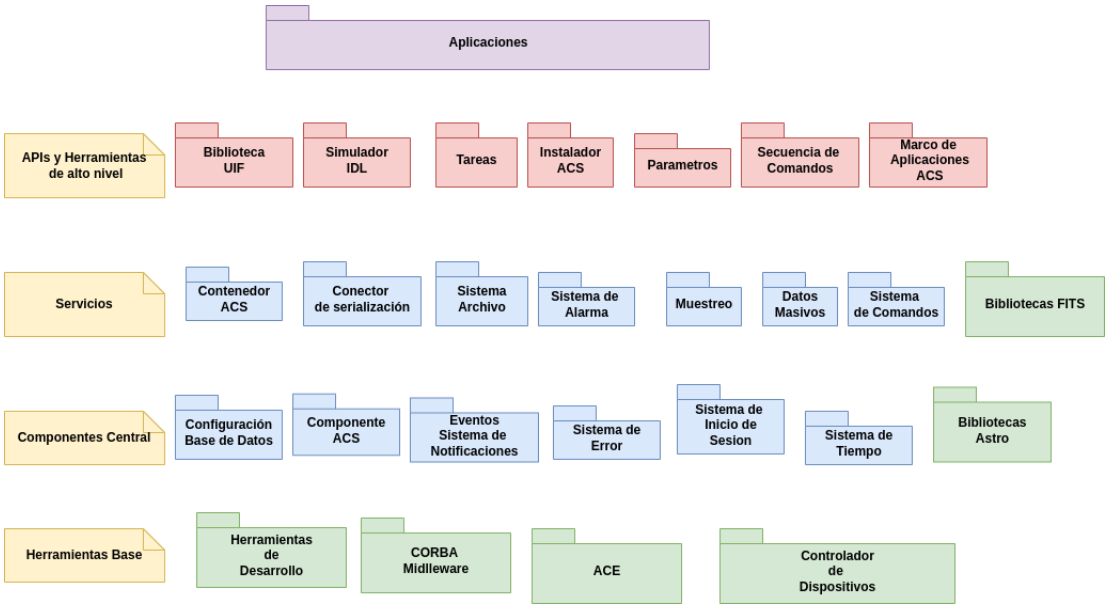


Figura 3.8: Paquetes de ACS.

La capa inferior contiene herramientas básicas que se distribuyen como parte de ACS para proporcionar un entorno de desarrollo común y tiempos de ejecución uniformes con el sistema operativo, con todas las otras capas y aplicaciones superiores. Estos son esencialmente componentes listos para usar, ACS en sí mismo, simplemente brinda soporte de paquetes, instalación y distribución. Esto asegura que todas las instalaciones del software tendrán el mismo conjunto básico de herramientas.
Esta segunda capa proporciona componentes esenciales que son necesarios para el desarrollo de cualquier aplicación. La tercera capa implementa servicios que no son estrictamente necesarios para el desarrollo de prototipos y aplicaciones de prueba o que están destinados a permitir la optimización de las prestaciones del sistema.
La cuarta y última capa proporciona aplicaciones (API) y herramientas de alto nivel. El objetivo principal de estos paquetes es ofrecer un camino claro para la implementación de aplicaciones, con el objetivo de obtener una conformidad implícita con los estándares de diseño y software mantenible.

Las estructuras de datos en ACS por lo general se representa como archivos XML para facilitar el almacenamiento y definir el contenido y la estructura de los objetos que se pasan entre subsistemas, para asi poder generar automáticamente las clases necesarias para acceder a datos serializados contenidos en estos objetos, y ermitir su

trasmisión, a través de la red.

El sistema divide claramente el sistema de control de observación y otros sistemas, y adopta una combinación de modularización con interfaces genéricas para cada nivel, logrando así la escalabilidad y portabilidad del sistema.

3.3.1. Modelo Contenedor/Componente

ACS está estructurado en forma de Componentes, que se ejecutan dentro de Contenedores y pueden ser administrados bajo llamadas entre Cliente/Servidor dentro de una red.
Este modelo es el principal instrumento para la organización y el desarrollo de las funcionalidades técnicas del software, permitiendo una estructura uniforme en todos los subsistemas .
Los Componentes, básicamente objetos CORBA, se definen en términos del Lenguaje de definición de interfaz (Interface Definition Language (IDL)) (este permite que un programa u objeto escrito en un lenguaje de programación se comunique con otro programa escrito en un lenguaje desconocido), pudiendo modelar un dispositivo físico, como las Cargas de Calibración, o una entidad abstracta (por ejemplo, un algoritmo, un objeto utilizado para acceder a una base de datos o cualquier cosa).
La Implementación de un Componente se establece por medio de la interfaz de Control de Acceso Básico (BACI) M. Plesko 2005 proporcionada por ACS, para entregar los conceptos de Objeto Distribuido, propiedades y características como también las herramientas para el desarrollo .
El acceso al componente se divide en dos interfaces de desarrollo:

- Interfaz de servicio: Para definir los métodos y propiedades mediante los cuales los servicios del componente están disponibles a los clientes.
- Ciclo de Vida : define un conjunto de métodos a los que solo puede acceder el contenedor del componente. Estos son : en el orden en que normalmente se invocarían: Iniciar() ,Ejecutar(), Limpiar(), Abortar().

EL Contenedor, por otro lado, se encarga de administrar los componentes.La implementación se realiza por medio de herramientas para la gestión y control de la

Interfaz de acceso, que proporciona los protocolos y metaservicios para centralizar el acceso a los Componentes y gestionar el ciclo de vida de estos, coordinando las solicitudes de acceso y los servicios remotos.

EL entorno Contenedor/Componente se muestra en la figura 3.9. Las interfaces publicadas por los Componentes 1,2 desarrollados utilizando diferentes lenguajes de programación C++ y Java, exponen sus servicios declarando explicitamente sus dependencias a otros Componentes o servicios remotos, a través de el Contenedor.

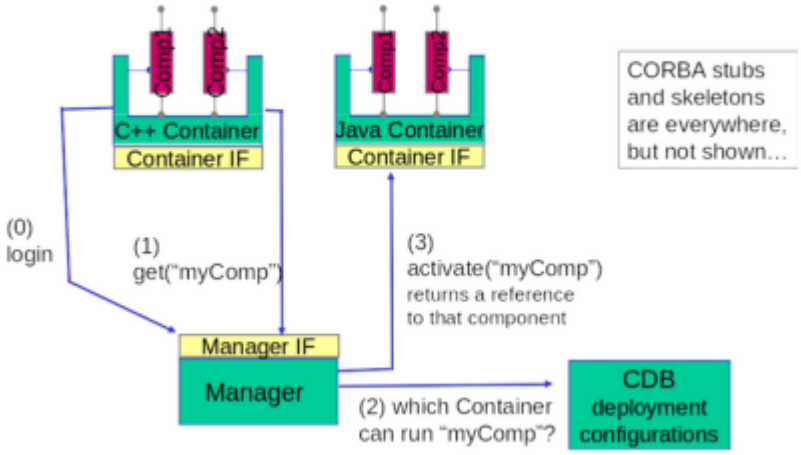


Figura 3.9: Un componente se implementa como un servidor CORBA, mientras que el contenedor se implementa como una capa de intercepción entre el agente de solicitud de objetos (ORB) de CORBA y el servidor.Créditos ALMA [14] .

La división de responsabilidades entre los Componentes y los Contenedores permite que la implementación de un Componente se pueda desarrollar de forma independiente, pudiendo definir su configuración al momento de ser ejecutado por parte del Contenedor, que se encarga de leer la información de archivos desde la base de datos .

3.3.2. Servicios Distribuidos del Sistema

El middleware es una parte integral de todos los sistemas de control modernos.En el pasado, el término middleware se usaba para referirse a la primera capa de abs-

tracción entre la capa de acceso de hardware de nivel inferior y la capa de aplicación de nivel superior. A menudo introducido para ocultar los protocolos de comunicación de la red . Este concepto se ha desarrollado mucho más allá de esto. Los sistemas de control modernos ahora pueden tener múltiples capas de middleware que se encargan de la red, pero también, cada vez más, de los aspectos de modelación del sistema de control, basándose en estructuras de datos bien definidas. Los nuevos modelos de componentes de middleware proporcionan la tecnología para organizar el sistema de control desde el nivel más alto hasta el nivel de acceso al hardware (ver Figura 3.10).

La base de ACS es CORBA un estándar desarrollado por Object Management Group (OMG) para proporcionar interoperabilidad entre objetos distribuidos. Permitiendo el intercambio de información, independientemente de las plataformas de hardware, los lenguajes de programación y los sistemas operativos. CORBA es esencialmente una especificación de diseño para un Object Request Broker (ORB) J. P. Almeida 2003, donde un ORB proporciona el mecanismo necesario para que los objetos distribuidos se comuniquen entre sí, ya sea localmente o en dispositivos remotos, escritos en diferentes idiomas o en diferentes ubicaciones .

La estructura de datos que CORBA utiliza se basa en el servicio de distribución de datos DDS de OMG, que permite definir interfaces y estructuras con el lenguaje de definición de interfaz (IDL). Proporcionando un compilador que genera las estructuras de datos resultantes que se transportarán a través de la red. con patrones de comunicación estandarizados (como publicación-suscripción centrada en datos con pares desacoplados basados en productores y consumidores de datos y el patrón de solicitud-respuesta.) sobre los cuales se construyen todas las aplicaciones.

Si bien la mayoría de los servicios distribuidos de alto nivel en ACS son administrados por CORBA, y dependen de la estructura de datos utilizada por este, no proporciona una interfaz para el control de los dispositivo de hardware.Es por esto con el tiempo ACS ha ido integrado nuevas tecnología de middlware para el monitoreo de bajo nivel como Internet Communications Engine (ICE) Y. L. Ding y Yang 2009 y Tango A. Senchenko 2018 , que se basan en la misma lógica de solicitud de objetos que CORBA.Para conectar y administrar las diferentes capas (nivel alto, nivel bajo y la capa de dispositivo G. Chiozzi 2001).

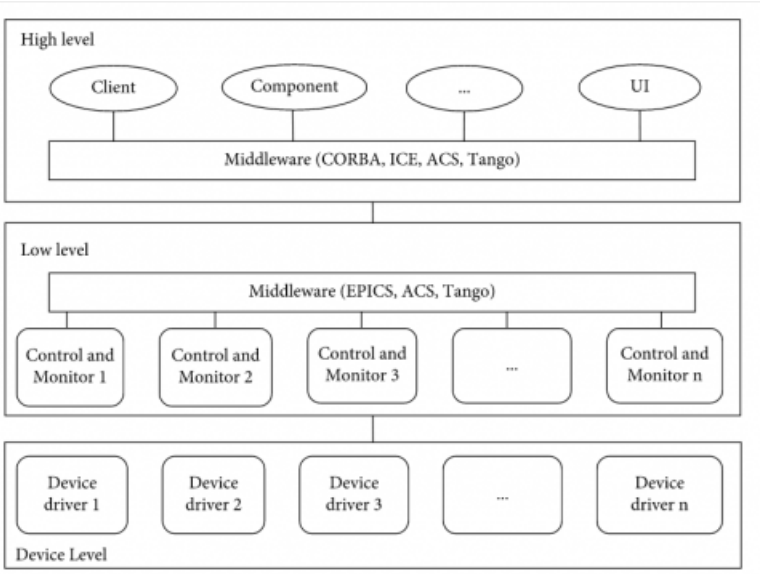


Figura 3.10: Arquitectura de middleware utilizada en un sistema de control para Radiotelescopio. Créditos de la imagen Zhiyong Liu et al. Jun Li 2021.

Algunos de los servicios enfocados a la computación distribuida que brinda ACS son :

- Repositorio de interfaz para la invocacion de objetos remotos de forma transparente..
- Servicio de notificación de eventos distribuido.
- Servicio de registro.
- Base de datos de configuración.
- Sistema de alarmas errores.
- Administrador.

3.3.3. Interfaz de los dispositivos de Hardware

La capa inferiores del software se encarga de la interfaz de los dispositivos de hardware.
ACS proporciona una abstracción genérica de acceso al hardware, con puntos de supervisión de forma independiente del software subyacente. Esta tecnología se basa en el patrón de diseño Componente/Propiedad/Característica (ver BACI en et al. 2002) introducida al desacoplar la implementación de las clases de propiedad del acceso al hardware .

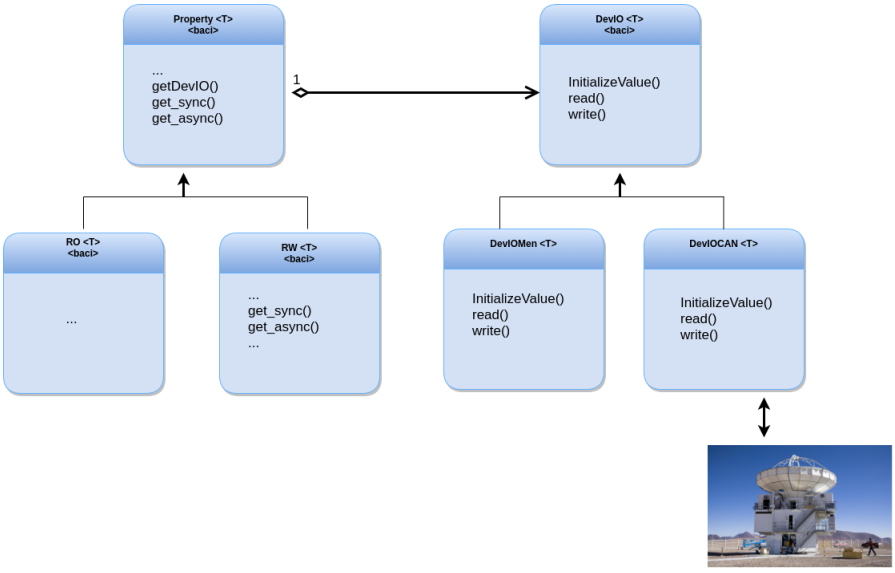


Figura 3.11: Diseño de clases DevIO para la implementación de dispositivos de hardware.

Las clases de objetos Device Input Output (DevIO) corresponden a plantillas que manejan los detalles de la comunicación del hardware, permitiendo uno o varios métodos de lectura y/o escritura, que descienden hasta los métodos en el nivel del dispositivo.
Las capas subyacentes a las DevIOs, corresponden a controladores de dispositivos, algunos de estos ya se encuentran implementados y a disposición de los desarrolladores, como los DevIo para el bus CAN y para la comunicación serial. Por lo tanto, el código de las Propiedades del dispositivo es totalmente genérico y proporciona acceso síncrono y asíncrono al valor de los puntos de monitoreo y control, permitiendo funciones de registro y alarmas.

3.3.4. Marco de Aplicaciones de alto nivel

La cuarta y última capa proporciona aplicaciones de herramientas a nivel superior. El objetivo principal de estos paquetes es ofrecer una forma clara para la implementación de aplicaciones de interfaz gráfica, con el fin de adecuarse a los estándares de diseño. Entre estos, se encuentran:

- **bibliotecas para la Interfaz de Usuario** Las Herramientas y aplicaciones de desarrollo y bibliotecas de widgets para la implementación de la interfaz de usuario como secuencias de comandos, marco de aplicaciones, bibliotecas FITS, etc.

- **ACS C++ y marco de aplicación de Java** La implementación de interfaces de usuario Java se basan en la biblioteca ABeans 3.12 que envuelve objetos CORBA, utilizando herramientas visuales y de manipulación de datos.

Ademas el back-end de ACS a traves de su base de datos, facilita la sincronización de las propiedades de los instrumentos con tecnologias enfcadas a interfaces gráficas. En el caso de la GUI prototipo del telescopio Cherenkov su front-end esta basada en tecnología Web Oya et al. 2017 que almacenan en búfer la información recopilada de varios componentes del software y la bases de datos, para su visualizacion.

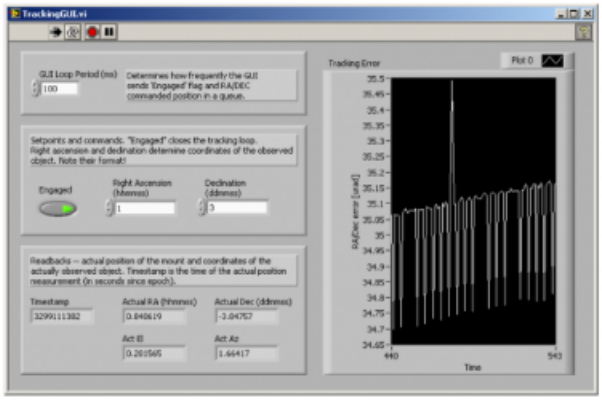


Imagen (a)

Imagen (b)

Imagen (c)

Figura 3.12: Ejemplos de interfaces gráficas desarrolladas con ACS, la imagen (a) corresponde a una GUI desarrollada por Labview integrando ACS al control de instrumentos K. Žagar 200. Las imágenes (b) y (c) muestran el panel de operaciones utilizado para el arreglo de antenas del telescopio CTA Oya et al. 2017.

Capítulo 4

Interfaz de Control para Cargas de Calibración

Este ccapítulo describe y analiza el desarrollo de la interfaz de control desde las CC, que permite que los dispositivos puedan ser integrados al software de control del radiotelescopio, según los requerimientos de LLAMA.

El sistema de control y adquisición de datos de los instrumentos de calibración se implementa integrando herramientas de hardware y software, que permiten controlar los componentes principales de cada carga y mantener una temperatura estable en cada una de ellas.
Las distintas opciones de configuración y secuencias de comandos, quedan definidas por medio de los puntos de monitoreo y control del instrumento, que permitirán adquirir datos del sistema.

El desarrollo de la interfaz de control, requiere de un conocimiento profundo del funcionamiento del sistema de control de las CC. Es por esto, que parte del trabajo se realizó con conocimiento adquirido en conjunto, con la parte involucrada en el desarrollo del control electrónico de las cargas. Produciendo una interfaz lógica que le permita al cliente establecer una comunicación con las CC.

En la primera sección de este capítulo se presenta la configuración general de la interfaz del sistema, como son los puntos de monitoreo y control de los instrumentos

y la descripción de su uso en el software de ACS.
En la sección 4.2 se detallan los requerimientos necesarios para el desarrollo de la interfaz de control establecidas por LLAMA, además se presentan algunas limitaciones, dentro de las cuales se establece el diseño del sistema.En la sección 4.3 se detalla la tecnología y la configuración de hardware utilizada. La sección 4.4 describe el diseño de la arquitectura para el control y la adquisición de datos, presentando los detalles del flujo de datos hacia y desde los instrumentos. Por ultimo en la sección 4.5 se describe la implementación y el desarrollo del sistema general para comunicarse con las CC .

4.1. Descripción general de la Interfaz de Control

4.1.1. Interfaz y Protocolo de Comunicación

El control de instrumentos consiste, en poder comunicar una computadora con un dispositivo formado por uno o más sensores y actuadores. La computadora, a través de un programa de software, envía los mensajes de comando al dispositivo, el cual ejecuta eventos asignados a cada mensaje que puedan ser recibidos. Estos eventos pueden estar relacionados con cambios en los parámetros del instrumento, adquisición de datos de la temperatura en los sensores o información de estado de los dispositivos.

La interfaz de control de los instrumentos se define por medio de la comunicación establecida entre las Cargas de Calibración y la computadora central de la antena del radio telescopio, el bus maestro de la antena (ABM, por sus siglas en ingles) ver figura, 4.1. La computadora central contiene el software de ACS LLAMA instalado, que funciona como un contenedor para el control de distintos instrumentos dentro de la antena.

Figura 4.1: Computadora ABM de LLAMA.

La integración de diferentes protocolos de comunicaciones en los sistemas de instrumentación y control dentro del radiotelescopio, puede establecerse por medio de diferentes tipos de conexiones como; Ethernet, Universal Serial Bus (USB), RS-232 y Controller Area Network (CAN).
En el caso de los dispositivos de calibración, LLAMA ha dispuesto que los instrumentos utilicen una interfaz ethernet con un protocolo TCP/IP, para así habilitar la comunicación remota entre las CC y la computadora.
El desarrollo de la Interfaz y el control remoto, se ha implementado bajo un protocolo Standard Commands for Programmable Instruments (SCPI), que representa un estándar de lenguaje de programación, diseñado específicamente para controlar instrumentos. Los comandos SCPI automatizan las operaciones remotas con los dispositivos, mediante scripts o programas de control con el software o firmware del instrumento, para ejecutar pruebas o generar señales.
De igual forma, SCPI puede ser utilizado por la mayoría de las aplicaciones de medición populares (como LabView), y aplicaciones que pueden conectarse con instrumentos de medición para recuperar datos (es decir, Matlab).

El diseño de los dispositivos de calibración, incluye una caja de control con el sistema de hardware necesario para controlar y adquirir datos.El diseño de este equipo, cumple con las características específicas para ser ubicado dentro de la antena, y cuenta con un periférico ethernet que permite que las CC puedan conectarse a una red de equipos controlados por la ABM.

4.1.2. Puntos de Monitoreo y Control

Los puntos de monitoreo y control describen las propiedades y metodos de las CC, y representan el medio por el cual el software de ACS puede comandar los dispositivos. Es por esto que se requiere de una descripción clara de la interfaz de los instrumento, por lo general las propiedades y metodos queda descrito en el documento de control para la interfaz de cada instrumento o Control Interface Document (ICD).La estructura de los comandos incluye una dirección de nodo, que se usa para enrutar el mensaje al dispositivo adecuado y una dirección relativa que se utilizada para identificar un comando específico para cada dispositivo.

Los datos asociados con el monitoreo y los puntos de control están dictados por la estructura de cada instrumento. En el caso de las CC, la temperatura de los sensores, se encuentra restringida en ciertos límites por el diseño de los componentes de hardware; solo ciertos bits en una palabra de estado pueden tener significado. Si se envían datos no válidos a un punto de control, generalmente se devuelve un mensaje con el error asociado. Si un punto de monitor devuelve datos no válidos, frecuentemente indica un problema interno grave en el dispositivo. La especificación de las restricciones en los puntos de monitoreo y control es una parte esencial y necesario para la implementación dentro del software de ACS.

Otro aspecto muy importante de los puntos de monitoreo y control es cómo se relacionan con los eventos de tiempo dentro del software. Los comandos que se deben ejecutar en límites de tiempo precisos, diseñados específicamente para el tiempo requerido, con un manejo especializado por parte del software.
En particular, para la frecuencia de muestreo asociado a la temperatura de los sensores de cada carga de calibración, como se verá mas adelante, se ha establecido un muestreo de 0.4 segundos, que permite tener un seguimiento continuo en los valores de temperatura.

4.2. Analisis de los Requerimientos del sistema

El objetivo de este apartado es describir de forma completa y precisa las necesidades que cubrirá el sistema de las cargas y el alcance de esta solución, a partir de las requerimientos del radio telescopio LLAMA. Se detallan las características que se incluirán en el sistema y las que no se incluirán, definiendo claramente el alcance del desarrollo. El análisis de los requisitos del sistema se puede dividir en requisitos funcionales y no funcionales. Por un lado, los requisitos funcionales describen las funcionalidades esperadas del sistema y su comportamiento, el plan para implementar estos requisitos se detalla en el diseño del sistema. Por otro lado, los requerimientos no funcionales definen los requisitos a nivel de usuario que no están directamente relacionados con la funcionalidad y que se utilizan para juzgar el funcionamiento del sistema en lugar de un comportamiento específico. Algunos ejemplos de estos requisitos son usabilidad, mantenibilidad, escalabilidad, etc. El plan para implementar requisitos no funcionales se presenta con mas detalle en la arquitectura del sistema. Las siguientes subsecciones describen ambos tipos de requisitos para este trabajo.

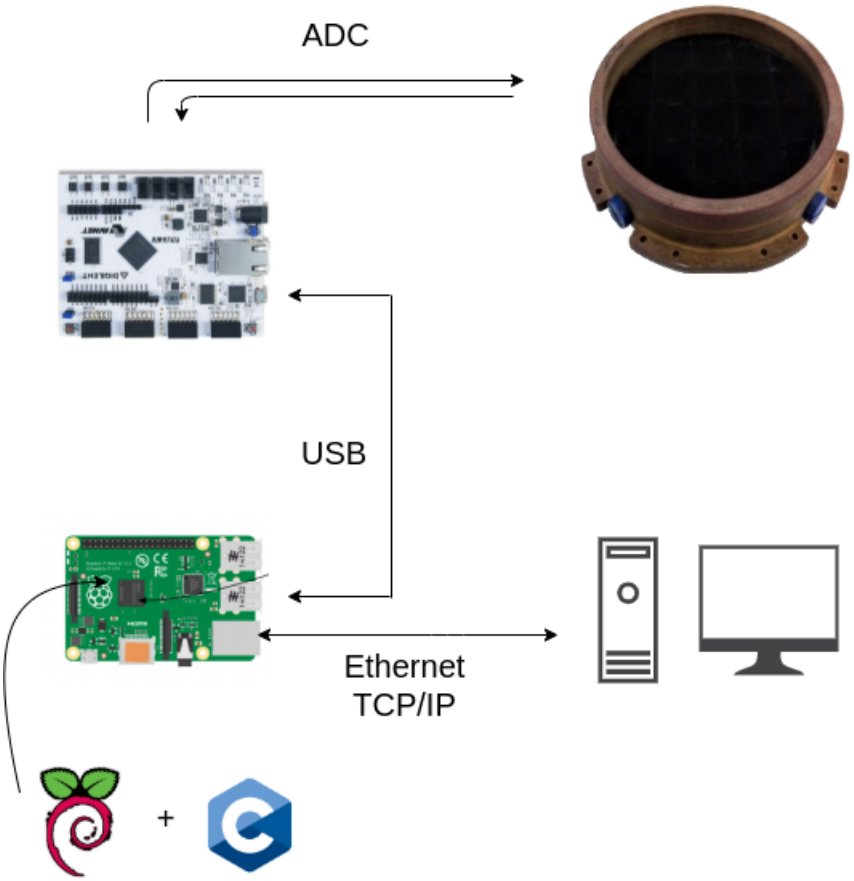


Figura 4.2: Esquema del sistema de comunicación con las CC.

4.2.1. Requerimientos Funcionales

Los requisitos funcionales consisten en la descripción de la funcionalidad de alto nivel del sistema y su comportamiento. Como ya se ha introducido en el apartado 4.1, el principal objetivo es controlar los instrumentos y adquirir datos de ellos, siendo capaces de adaptarse a diferentes situaciones. Para contextualizar mejor el sistema, la Figura 4.2 proporciona un diagrama de las conexiones y relaciones dentro del sistema.

Interfaz para crear la descripción del Instrumento

La interfaz de las CC debe proporcionar el acceso a todas las funcionalidades de los tres dispositivos, mediante mensajes estructurados a través de un protocolo de comunicación. La estructura de estos mensajes debe ser implementada mediante

comandos SCPI. Los comandos provocan dos tipos posibles de acciones: o hacen que el dispositivo de hardware envíe un cierto patrón de bits al solicitante (que no se supone que modifique el estado interno del dispositivo), o utilizan el contenido del comando para modificar el estado interno del dispositivo (que, a su vez, puede hacer que el dispositivo realice alguna acción, como establecer una temperatura). Las acciones del primer tipo, se denominan puntos de monitoreo; los segundos se denominan puntos de control.

Iniciar y detener la adquisición de datos

El sistema debe proporcionar una interfaz para que el usuario inicie y detenga el proceso de adquisición de un equipo.

Enviar comandos al dispositivo

El sistema debe poder enviar mensajes a cada instrumento. Estos mensajes incluyen comandos de configuración, comandos de selección del modo de operación y comandos para solicitar paquetes de datos específicos que se envíen desde el dispositivo.

Convertir valores a datos binarios

Siempre que se deba enviar un comando al dispositivo, se debe convertir a su forma binaria , generando una secuencia de bits con la estructura del paquete asociada con el comando y los valores de cada campo. Para transformar valores en secuencias de bits también se utiliza la descripción del instrumento mediante un bus serial.

Proporcionar acceso remoto al dispositivo

El sistema debe poder conectarse al usuario mediante un servidor socket para enviar y recibir datos. Esto incluye mantener un servidor activo para establecer una conexión, convertir paquetes en secuencias de datos binarios que se pueden transferir a través de la conexión, y el proceso inverso, convertir los datos de la conexión en una secuencia binaria que se puede procesar y los paquetes de datos se pueden extraer de ella para ser enviados de forma asíncrona a la tarjeta Field Programmable Gate Array (FPGA).

Convertir datos binarios en valores físicos

Siempre que se reciba información del dispositivo, es necesario convertirla en valores estructurados significativos. Para cada medición, el instrumento envía un paquete al sistema que debe ser detectado, identificado y analizado. Analizar un paquete consiste en, una vez que su inicio y final se han encontrado correctamente, unir bits de datos en bloques de bits que se pueden convertir a valores enteros, puntos flotantes, caracteres, y asignar un nombre a cada campo a partir de la descripción del instrumento.

Respuesta a los Errores

En caso de error, el sistema debe informar una descripción detallada del problema al sistema de registro e intentar recuperarse el error. Si el error es irrecuperable, por ejemplo, cuando no se puede establecer la conexión con el instrumento, la sesión de adquisición debe finalizar.

4.2.2. Requerimientos no funcionales

A diferencia de los requerimientos funcionales, los requerimientos no funcionales no funcionales describen otros aspectos importantes que pueden afectar la experiencia del usuario y el rendimiento del sistema. Para este proyecto, se han considerado los siguientes requisitos:

Fiabilidad

Dado que el objetivo de este sistema es adquirir datos de instrumentos que pueden estar en lugares de difícil acceso, la confiabilidad es un requisito esencial . Además, debido a que este software proporciona control remoto y monitoreo del instrumento, debe ser accesible en cualquier condición y debe poder recuperarse de errores.

Mantenibilidad

Una vez desarrollada la aplicación, en caso de que se encuentren errores, estos serán fácilmente reparables. El estilo de programación debe ser simple y claro y

codificado, bien documentado para que cualquiera pueda modificar y resolver un error sin depender del programador que escribió el código fuente original.

Portabilidad

Para que el sistema sea lo más genérico posible y se reutilice en diferentes proyectos, deberá ser flexible con los requisitos del entorno de trabajo. En concreto, el sistema funcionará en diferentes arquitecturas de hardware y los principales sistemas operativos, incluidos Microsoft Windows, GNU/Linux y Mac OS X, sin requerir muchos cambios. Por lo tanto, se basa en bibliotecas y lenguajes de programación multiplataforma para su implementación.

Legal

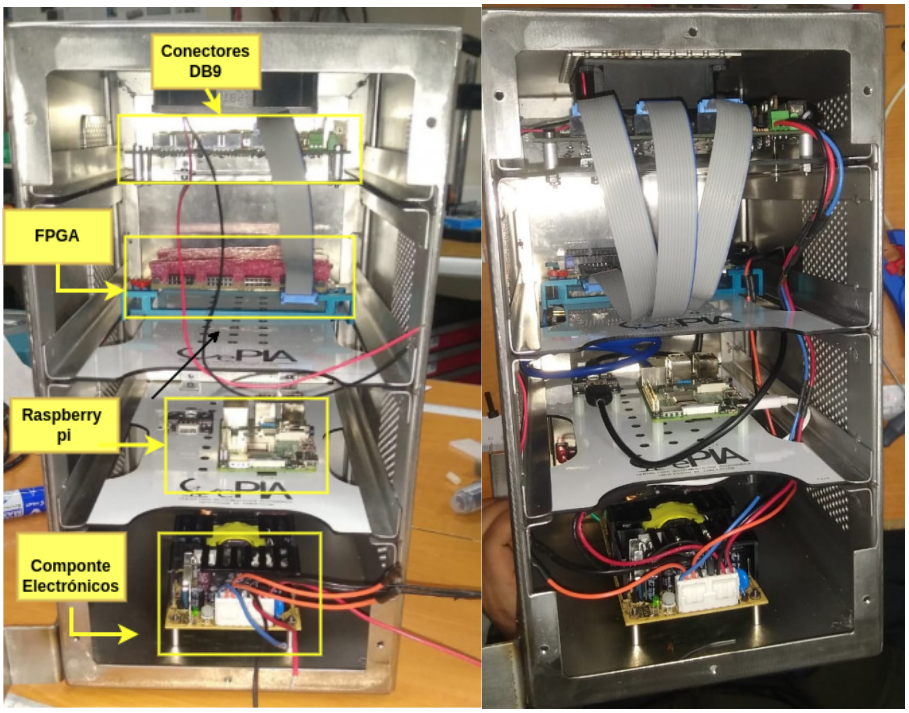
Dado que el sistema está diseñado como base para futuras aplicaciones de adquisición de datos y control de instrumentos, además de posibles actualizaciones,su licencia debería permitir que el código fuente se reutilice y modifique.

4.3. Descripción y Configuración de Hardware

Los sistemas de adquisición de datos, puede definirse como aquellos componentes que participan en el proceso de transformación de los fenómenos físicos, asociados a señales eléctricas que puedan ser medidas y transformadas a un formato digital para su recolección, procesamiento y posterior visualización.

El diseño del sistema de control se basa en las plataformas modulares de hardware utilizadas para medir o cuantificar, los fenómenos involucrados en el control de la temperatura de cada carga.
El sistema de control y la adquisición de datos, se compone de dos tarjetas electrónicas; la FPGA Arty A7 y una computadora de placa única, Raspberry pi 4 b+.

La figura 4.3 muestra la ubicación de cada una de las placas dentro de la caja de control, y las conexiones de hardware para la comunicación y control de los dispositivos.



(a) Modulos de hardare (b) Conexión entre los distintos módulos.

Figura 4.3: Caja de control de las CC,

Esta división se ha determinado por la funcionalidad de cada dispositivo de hardware, los requerimientos (4.2) y las herramientas necesarias para desarrollar cada parte del sistema.
La solución permite un alto nivel de flexibilidad, que se decidió utilizar en beneficio de la capacidad del sistema para futuras ampliaciones y modificaciones.
Mientras que la tarjeta FPGA se encarga del control, la Raspberry pi se ocupa del estándar para la interfaz de los instrumentos con el cliente.

La descripción de las arquitecturas de hardware de cada tarjeta, definen con mayor detalle el propósito específico que cumple cada dispositivo, dentro del sistema de control.

4.3.1. Tarjeta FPGA

La tarjeta utilizada en este proyecto es una Arty-7 de Xilinx, ver figura 4.4 . La Arty A7 es una plataforma de desarrollo creada por Digilent, que incluye un chip FPGA Artix-7 de Xilinx, con 1800 Kbits de BRAM (Block Ram), un convertidor analógico-digital incluido y programable mediante JTAG o la memoria Quad-SPI Flash de 16 MB que lleva incorporada. Como elementos externos tiene una memoria DRAM DDR3L de 256 MB, un puente USB-UART y un puerto Ethernet que soporta velocidades de 10 o 100 Mbits por segundo. Además, como periféricos, tiene 4 interruptores, 4 botones, 8 leds, 4 conectores Pmod para conectar cualquier periférico externo que se necesité y un adaptador para conectar una placa Arduino.

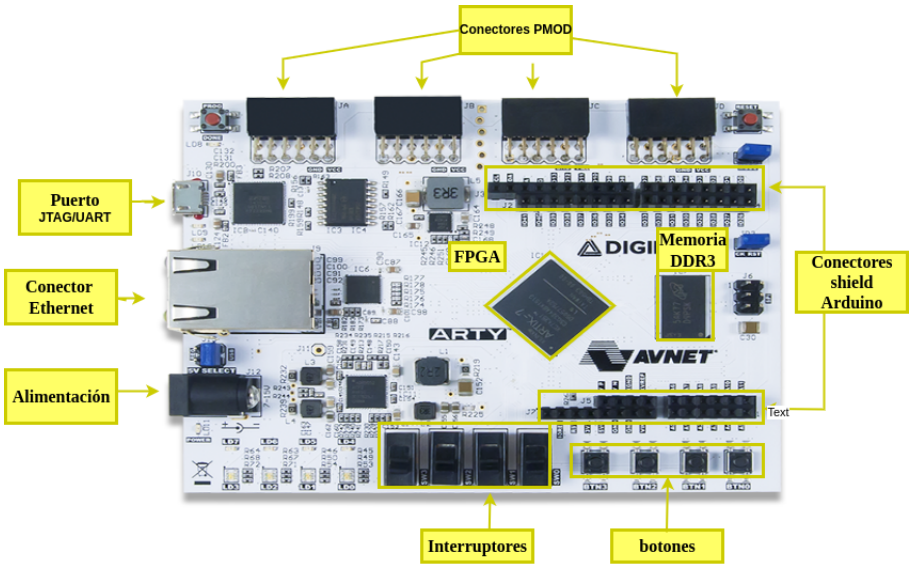


Figura 4.4: Tarjeta Artix-7.

El desarrollo en la tarjeta esta proporcionado por Xilinx, que distribuye su propio software para la sintesis y análisis de diseños Hardware Description Language (HDL), el entorno de desarrollo Vivado [13]. Este permite programar, depurar, reprogramar y repetir operaciones, pudiendo llevar a cabo completamente el desarrollo de circuitos específicos para el diseño de control. La FPGA puede mostrar sus capacidades y ventajas reflejadas principalmente en el procesamiento de senales digitales (DSP Digital Signal Processor) utilizadas para calculos, como cifrado y decifrado, modulación y demodulación.

4.3.2. Tarjeta Raspberry pi

La computadora de placa única (Singel Board Computer (SBC)) Raspberry Pi (RPI) construida por Fundación Raspberry Pi en Reino Unido, es una computadora muy económica con un que sistema operativo Linux, pero también incluye una serie de pines GPIO (entrada/salida de propósito general) que le permiten monitorear componentes electrónicos de computación física .

El modelo utilizado en este proyecto es Raspberry Pi 4 B+ (figura 4.5) .

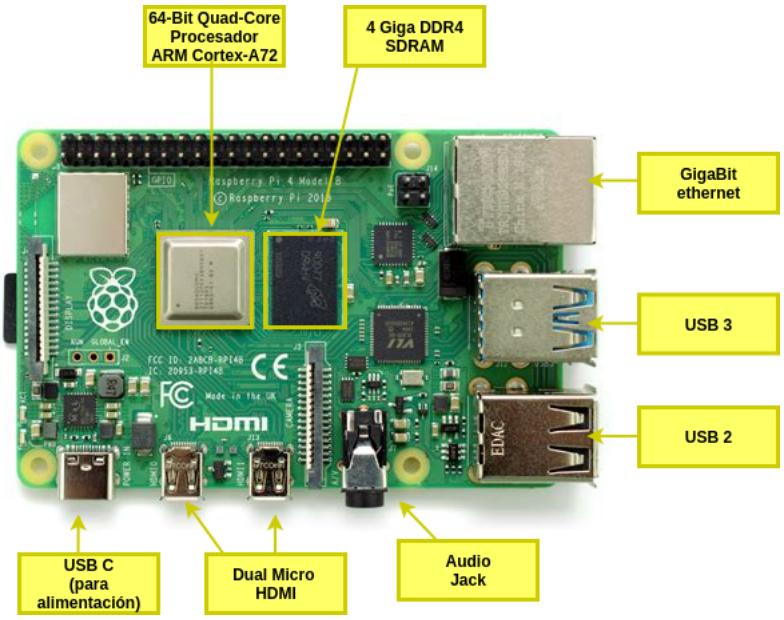


Figura 4.5: Tarjeta Raspberry Pi 4B+ .

Las principales especificaciones técnicas del modelo Raspberry Pi a utilizar en este proyecto son:

- Broadcom BCM2711, SoC de 64 bits Cortex-A72 (ARM v8) de cuatro núcleos a 1,5 GHz.
- SDRAM LPDDR4-3200 de 8GB.
- 2.4 GHz y 5.0 GHz IEEE 802.11ac inalámbrica, Bluetooth 5.0, BLE Gigabit Ethernet.
- 2 puertos USB 3.0; 2 puertos USB 2.0.
- 2 × puertos micro-HDMI (hasta 4kp60 compatible).
- Puerto de vídeo compuesto y audio estéreo .

La Raspberry pi utiliza un procesador de sistema de arquitectura ARM, que integra muchos recursos de hardware listos para usar y para ser programados. Permitiendo que se puedan realizar funciones control directamente el hardware a través de interruptores. De este modo, se puede utilizar para ejecutar interfaces y aplicaciones. Sus ventajas se reflejan principalmente, en una gran capacidad de procesamiento de datos y una mayor velocidad de funcionamiento.

4.4. Aplicación de Software para la Adquisición de datos

El desarrollo de la interfaz de los instrumentos, debe considerar la información obtenida por la tarjeta Arty A7, que se encarga de la adquisición simultánea de las señales análogas provenientes de los sensores y calefactores de cada carga (ver figura 4.7), por medio de los conversores análogo a digital (Analog to Digital Converter (ADC)) de 12 bits. Una vez que se realiza la conversión y se almacenan los datos en los registros internos de propósito especial, se utiliza un esquema de control derivado integral proporcional (PID por sus siglas en ingles), que funciona como un sistema de retroalimentación automatico en la señal de salida, permitiendo corregir el error entre el proceso de medición y la respuesta deseada, y así poder mitigar las variaciones de voltaje que se puedan producir al aplicar una tensión de entrada en la señal.
De esta forma el controlador desarrollado en la FPGA, define las funcionalidades del sistema y describe las propiedades de cada punto de monitoreo y control en los instrumentos.

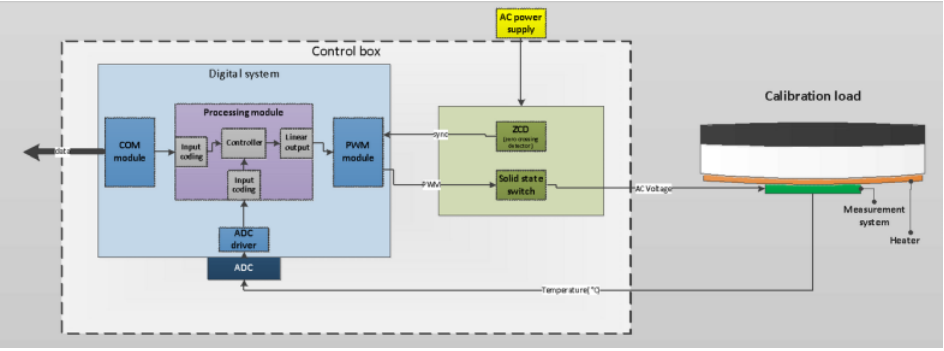


Figura 4.6: Esquema de la plataforma de Control para la FPGA, con la integración de los componentes necesarios para monitorear y controlar las CC.

La información desde la FPGA, es entregada a la SBC por medio de un protocolo USB/UART, que actualiza continuamente los datos recibidos, para así poder detectar cualquier tipo de falla o anomalía que se pudiese presentar en el sistema.
La figura 4.8 muestra la lógica de la aplicación para el sistema ARM. El cliente a

través de una red Ethernet, se comunica con los dispositivos por medio de una trasmisión de comandos al servidor socket, que se ocupa de analizar la información y de entregar una respuesta a cada solicitud.

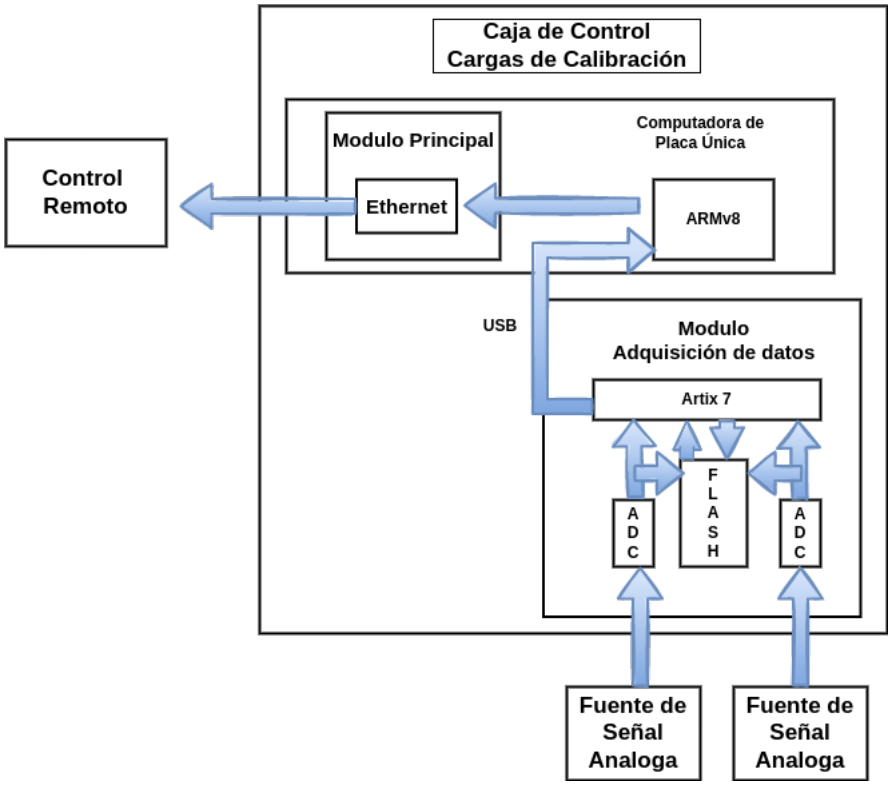


Figura 4.7: Flujo de Datos, dentro de la Caja de Control.

Una de las características con las que cuenta la SBC, es que posee una CPU principal con un sistema operativo con diferentes bibliotecas de código abierto, para acceder y controlar los distintos periféricos de la tarjeta, facilitando el desarrollo de procesos dentro del sistema operativo.

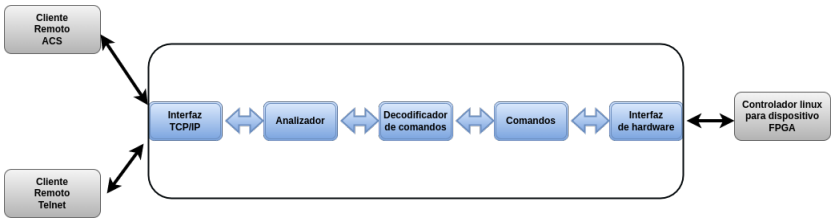


Figura 4.8: Flujo de Datos para la aplicacion para la interfaz de las CC.

El diseño de la aplicación de software para la interfaz de las CC, considera una arquitectura basada en el flujo de datos de la figura 4.8.
Para esto se ha diseñado un controlador que considere la lectura de los datos desde y hacia la FPGA, mediante una comunicación asíncrona . La transmisión asíncrona es aquella que se transmite o se recibe mediante un carácter, bit por bit, con tareas que pueden solicitar acciones de control, y pueden continuar su ejecución sin necesidad de esperar los resultados.Este tipo de controladores se utiliza cuando se requiere avanzar en alguna ejecución sin que se terminen las operaciones de entrada y salida. Un caso especial es cuando se realizan operaciones en varias etapas, aquí, la tarea puede encargarse de procesar los datos de la primera etapa, mientras el controlador obtiene los datos de la siguiente.Ademas de la implementación de la sintaxis para los comandos estándares SCPI.

4.5. Implementación Interfaz CC

La implementación de la aplicación de interfaz para las CC se realiza en la distribución y Configuración del Sistema Linux utilizado en la tarjeta Raspberry pi, queda descrito en el apéndice C.0.1.

4.5.1. Organización de Directorios

Para la estructura del código fuente requerido para construir la aplicación, se creó la siguiente jerarquía de carpetas:

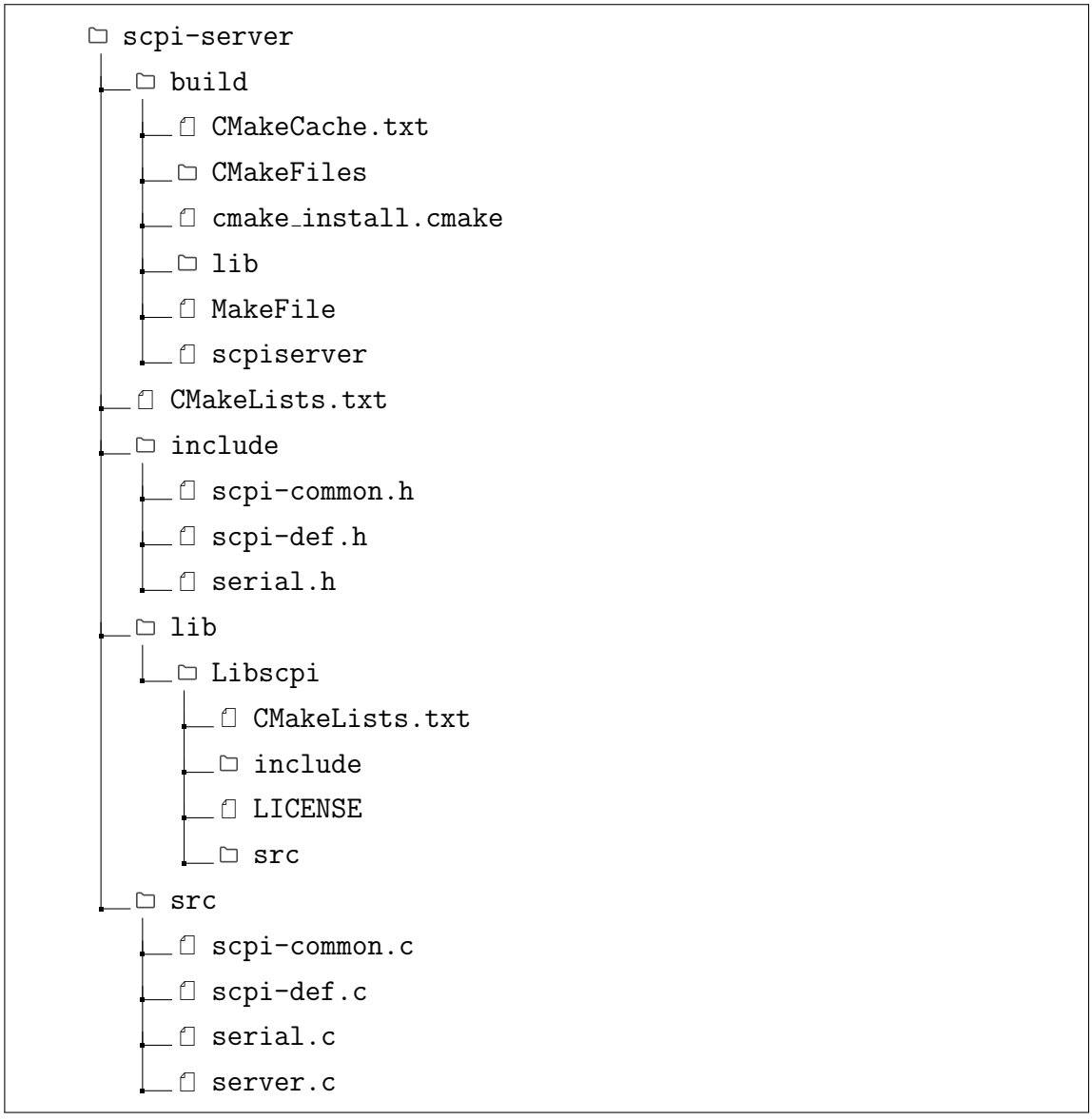


Figura 4.9: Organización de ficheros y directorios para el servidor SCPI.

Los detalles de cada directorio se deteallan en la siguiente taabla:

Cuadro 4.1: Organización de Directorios para servidor SCPI.

Directorios	Descripción
build	Directorio con archivos de configuración necesarios para la construcción del software.
include	Esta carpeta incluye todos los archivos de cabecera para cada archivo fuente.
lib	Carpeta con bibliotecas adicionales.Contiene los archivos fuentes del estandar de comandos scpi-paser.
src	Contiene los archivos fuentes para la implementación servidor SCPI y el controlador serial.

El sistema operativo ha sido configurado de forma que la aplicación de software comience automáticamente como un proceso interno del sistema Raspberry pi OS al ser inicializado.

4.5.2. Aplicación de Software

El desarrollo de la aplicación de software se basa en la lógica del modelo presentado en la Figura 4.10. El programa se divide en dos partes principales, un módulo de comunicación con la tarjeta FPGA y otro para el servidor encargado de analizar los comandos de control y monitoreo, que puedan ser solicitados de forma remota. El programa principal se encarga de crear un servidor **server.c**, que esperara por llamadas entrantes procedente de algún cliente o diferentes clientes, los detalles de la configuración del protocolo de comunicación TCP/IP se manejan dentro de las bibliotecas de Linux configuradas mediante el uso de socket.

Para ciertos eventos como nuevas solicitudes de conexión o cuando existan nuevos paquetes de datos disponibles, el servidor mediante la función `select` realizara un registro de estos archivos descriptivos de entrada, asegurándose que cada uno retorne con alguna respuesta asociada al comando recibido .

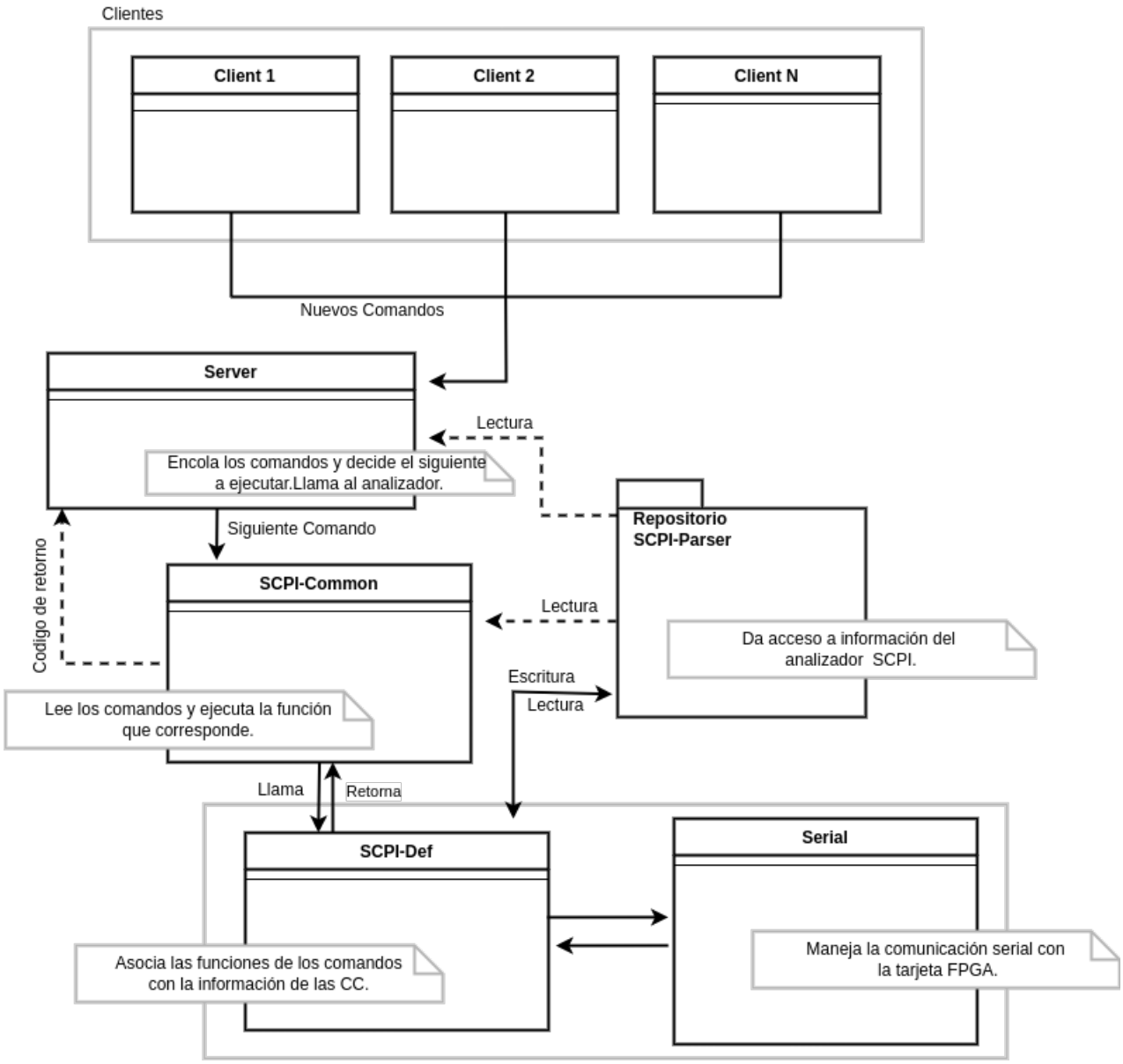


Figura 4.10: Arquitectura del flujo de datos para el Servidor de comandos SCPI.

De esta forma los paquetes de datos aceptados, se traducen desde la interfaz de red en uno o mas comandos internos de mas bajo nivel, donde son almacenados por la biblioteca del analizador, para luego ser asociados con alguna función de llamada.

Las definiciones comunes del estadar SCPI se encuentran en el archivo `scpi-common.c` con una lista de comandos básicos para el control digital de instrumentos, aquí se incluyen los comandos específicos de cada instrumento que puedan no encontrarse

en la lista por defecto.

Las funciones de llamada se asocian a la biblioteca del analizador, para leer los parametros y escribir los resultados de cada comando. La mayoría de los comandos se traducen en actualizaciones de registros de valores, asociados a la temperatura de algún sensor o actualizaciones de estados de las CC.

Cada vez que se encuentre una cadena de comandos que es coincidente, se llama a la función de devolución de llamada correspondiente en `scpi-def.c`. Si corresponden los parámetros del comando, se leerá la función de devolución de llamada antes de que se procese el comando. En caso contrario, de no coincidir se devolverá un mensaje para cada tipo de error establecido dentro del estándar.

Para la comunicación con la tarjeta FPGA se utilizan bibliotecas de subprocesos estándar del sistema operativo Linux. La función **serial.c** realizara una lectura continua de los datos dentro de un bucle de control, mediante un proceso paralelo a la interfaz de red.
La información con datos es transferida a una estructura de código global, para luego sera decodificada y concatenada, transformando los valores digitales a valores de temperatura física.

Cada función de devolución de llamada en `scpi-def`, apunta a valores determinados en el arreglo de la estructura global de los datos.

De esta forma se asegura una actualización continua de datos hacia el cliente .

4.5.3. Implementación Controlador

Las operaciones que el controlador debe realiza son las siguientes:

- Esperar a que lleguen mensajes a la cola.
- Obtener el mensaje y entregar la información a la tarea.
- Continuar con nuevas operaciones de entrada o salida.

Mientras que en la rutina de atención para las interrupciones se realiza lo siguiente:

- Obtener la información desde (o enviar hacia) el dispositivo de hardware.
- Empaquetar la información en un mensaje.
- Agregar el mensaje a la cola.

La configuración del protocolo UART, define la transferencia de información entre la comunicacion de las tarjetas Raspberry pi con la tarjeta FPGA . La función de la UART es convertir los datos entrantes y salientes en un flujo binario en serie. Los datos en serie de 8 bits son recibidos por dispositivo periférico . Estos datos están presentes en forma de modulación y se transmiten a una tasa de baudios.
La secuencia de bits enviada tiene una estructura definida, añadiéndole bits de inicio y bits que indican el término de un paquete, para separar así los caracteres que se van enviando/recibiendo. La configuración del protocolo UART, para generar el canal para la trasmisión de información con la FPGA, es el siguiente:

Cuadro 4.2: Configuración de Bits UART.

Velocidad en Baudios	Bits Datos	Paridad	bit de parada	Control de flujo
19200	8	sin paridad	un bit	sin flujo de control

La lectura de bytes se genera en un ciclo de 19200 bits por segundo, cada paquete recibido corresponde a un tipo de dato entero.
La lectura continua de bytes es monitoreada por la funcion select, para luego apuntar a una estructura global, de variables que representan cada tipo de dato. El siguiente pedazo de código muestra a lectura de bytes en el puerto serial:

Código 4.1: Código Comunicación Serial

```
1  while(1)
2   {FD_ZERO(&rset);
3   FD_SET(s_serial->serial_fd, &rset);
4   tv.tv_sec = 1;
5   tv.tv_usec = 0;
6   count = select(s_serial->serial_fd + 1, &rset, NULL, NULL, &tv);
7
8   if(count > 0)
9   {
10  // memset(s_serial->recv_temp, 0,_SERIAL_BUFFER_SIZE_);
11  s_serial->recv_cnt = read(s_serial->serial_fd,&s_serial->recv_data,1);
12  s_serial->data.data=s_serial->recv_data;
13  if (s_serial->callback_function != NULL)
14  {
15          (*s_serial->callback_function)(&s_serial->data);
16  }
17          } else {if(s_serial->serial_fd < 0)
18                  { close(s_serial->serial_fd); } }
19
```

Los tipos de datos enviados, contienen las distintas etiquetas con la información correspondiente para cada punto de monitoreo.La estructura global permite que se pueda acceder a estos datos, mediante un proceso que apunte a la función de llamada global.

Código 4.2: Codigo Comunicacion Serial

```
1  typedef struct
2  {
3   uint32_t serial_fd;
4          pthread_t pt_recv;
5
6          int32_t recv_cnt;
7          int32_t recv_buff;
8          unsigned char recv_data;
9          int32_t (*callback_function)(void *);
10
11         unsigned char send_buff[SERIAL_BUFFER_SIZE];
12
13         t_serial_data data;
14
15 }t_serial;
```

4.5.4. Implementación Servidor SCPI

La biblioteca scpi-parser, puede definir patrones de comando y devoluciones de llamadas, mediante una estructura de definición de comandos SCPI. La descripcion general de la estructura de para el desarorollo de comados SCPI, se muestra en el apendice C.0.1 . La biblioteca SCPI-Parser desarrollada por Jan Breuer [22] (basada en los estadares SCPI-99 [2] y IEEE 488.2-2004 [5]), tiene como objetivo proporcionar la capacidad de análisis de los comandos SCPI en el lado del instrumento.
Los códigos fuente se publican con licencia BSD de código abierto, en lenguaje de programación c.
La configuración del puerto de escucha TCP se establece en el puerto SCPI estándar (5025), permitiendo que cualquier cliente Telnet similar, pueda enviar comandos al dispositivo a través de un modo sin formato, es decir, texto ASCII.
La descripción de interfaz para cada punto de monitoreo y control de las CC se encuentra completamente descrito en el apéndice A.
Para la implementación de la biblioteca libscpi se ha desarrollado un servidor socket que espere por llamadas remotas, esto queda definido en el siguiente código:

Código 4.3: Tabla con Comandos SCPI

```
SCPI_Init(&scpi_context,
scpi_commands,
&scpi_interface,
scpi_units_def,
SCPI_IDN1, SCPI_IDN2, SCPI_IDN3, SCPI_IDN4,
scpi_input_buffer, SCPI_INPUT_BUFFER_LENGTH,
scpi_error_queue_data, SCPI_ERROR_QUEUE_SIZE);

scpi_context.user_context = &clifd;

while(1) {
        rc = waitServer(clifd);
        if (rc < 0) { /* failed */
                perror("recv() failed");
                break;  }
        if (rc == 0) { /* timeout*/
                SCPI_Input(&scpi_context, NULL, 0);
        }
        if (rc > 0) { /* something to read */
                rc = recv(clifd, smbuffer, sizeof (smbuffer), 0);
                if (rc < 0) {
                        if (errno != EWOULDBLOCK) {
                                perror("recv() failed");
                                break;}
                }else if (rc == 0) {
                        printf("Connection closed\r\n");
                        break;
                }else {
                        SCPI_Input(&scpi_context, smbuffer, rc); } } }
close(clifd);
}
```

La ejecución de la biblioteca, se genera llamando a SCPI_Init, esta estructura contiene las devolución de llamadas y se encarga de juntar los distintos parametros de la biblioteca. Los siguientes métodos, también han sido implementados en el código:

Código 4.4: Implementacion de metodos SCPI

```
SCPI_Write(scpi_t* context, const char* data, size_t len)
SCPI_Flush(scpi_t* context)
SCPI_Error(scpi_t* context, int_fast16_t err)
SCPI_Control(scpi_t* context, scpi_ctrl_name_t ctrl, scpi_reg_val_t val)
SCPI_Reset(scpi_t* context)
```

Como es posible determinar, SCPI_Write permite que libscpi escriba en la salida que se elija, pudiendo asociar archivos descriptivos a la función y SCPI_Flush se usa para vaciar cualquier búfer de salida que pueda existir. SCPI_Error se encarga de imprimir los mensajes de error de libscpi, SCPI_Reset reinicia el dispositivo y SCPI_Control se usa para escribir en el canal de control . Para hacer que libscpi analice cualquier cadena entrante nueva desde el socket de lectura, su código llama a SCPI_Input o, en el caso de comandos singulares, SCPI_Parse también se puede usar directamente.

La implementación de los comandos, llamada scpi_commands define todos los comandos asociados al instrumento en un formato de patrón con devolución de llamada, incluyendo los comandos obligatorios de IEEE, como por ejemplo, *CLS (estado claro):

.patrón = "*CLS", .devolución de llamada = SCPI_CoreCls,

El '*'(asterisco) delante de un comando significa que es un comando SCPI común requerido para todos los dispositivos y debe implementarse. Los más importantes han sido implementados son *IDN?, que consulta al dispositivo sobre su identificación, *RST para ordenar que el dispositivo se reinicie y *WAI que le dice al dispositivo que espere para ejecutar cualquier comando nuevo, hasta que los comandos que preceden a este comando hayan finalizado.Estos tres comandos básicos han sido implementados en el código scpi-common.c con los comandos tipos de las CC, como se muestra en la siguiente tabla:

Código 4.5: Tabla con Comandos SCPI

```
const scpi_command_t scpi_commands[] = {
        /* IEEE Mandated Commands (SCPI std v19999.0 4.1.1) */
{ .pattern = "*IDN?", .callback = SCPI_CoreIdnQ,},
{ .pattern = "*OPC" , .callback = SCPI_CoreOpc,},
{ .pattern = "*OPC?", .callback = SCPI_CoreOpcQ,},
{ .pattern = "*RST" , .callback = SCPI_CoreRst,},
{ .pattern = "*SRE" , .callback = SCPI_CoreSre,},
{ .pattern = "*SRE?", .callback = SCPI_CoreSreQ,},
{ .pattern = "*STB?", .callback = SCPI_CoreStbQ,},
{ .pattern = "*TST?", .callback = SCPI_CoreTstQ,},
{ .pattern = "*WAI" , .callback = SCPI_CoreWai,},

//* Required SCPI Commnads (SCPI std V1999.0 4.2.1) */
{.pattern = "SYSTem:ERRor[:NEXT]?", .callback =SCPI_SystemErrorNextQ,},
{.pattern = "SYSTem:ERRor:COUNt?",  .callback = SCPI_SystemErrorCountQ,},
{.pattern = "SYSTem:VERSion?",      .callback = SCPI_SystemVersionQ,},

{.pattern = "STATus:QUEStionable[:EVENt]?", .callback = SCPI_StatusQuestionable
                    EventQ,},
{.pattern = "STATus:QUEStionable:ENABle",   .callback = SCPI_StatusQuestionable
                    Enable,},
{.pattern = "STATus:QUEStionable:ENABle?",  .callback = SCPI_StatusQuestionable
                    EnableQ,},
{.pattern = "STATus:PRESet",                .callback = SCPI_StatusPreset,},

{.pattern = "SYSTem:COMMunication:TCPIP:CONTROL?",.callback =SCPI_SystemComm
                                TcpipControlQ,},

 /* Calibration Loads Commands */
 {.pattern = "CALibration:LOAd#:SENsor#?", .callback = SCPI_GetLoadSensor,},
 {.pattern = "CALibration:LOAd#:HEAter",   .callback = SCPI_SetHeater,},
 {.pattern = "CALibration:FEEDback:LOAd#",  .callback = SCPI_SetFeedback,},
 {.pattern = "CALibration:CONTrol:LOAd#",  .callback = SCPI_SetControl,},
 {.pattern = "CALibration:STOp",           .callback = SCPI_SetStop,},

        SCPI_CMD_LIST_END
};

```

La razón del uso mixto de mayúsculas y minúsculas en los comandos se relaciona con el aspecto de "patrón": en SCPI solo se requiere la parte mayúscula de la plantilla, y la sección minúscula de un comando se puede omitir por brevedad. Como se ha señalado anteriormente, un comando seguido de un signo de interrogación es una pregunta. El uso de dos puntos es para separar los niveles de la jerarquía del árbol que define la interfaz SCPI.

Capítulo 5

Implementación Software de Control

Se presenta el trabajo de implementación de control para las CC usando el marco de ACS para el radiotelescopio LLAMA, este es el objetivo principal del actual trabajo de Tesis.
En principio se presenta la instalación y configuración del software, además de las aplicaciones necesarias para poder trabajar dentro de un ambiente de desarrollo, acorde a los requerimientos de LLAMA.
Segundo, se describen los pasos, involucrados en el desarrollo del componente, intentando brindar información mas allá del alcance de los componente, demostrando el uso del marco para el desarrollo de subsistemas concretos en el software.El modelo de contenedor/componente de C++ está totalmente integrado en el software. Por lo que puede ser utilizado por los subsistemas de LLAMA o las partes de ellos que no tienen requisitos de tiempo real y no controlan directamente los dispositivos de hardware.
La sección tres describe la construcción de la GUI para las CC.Principalmente los científicos (que se encuentren en Proyectos de Observación o investigando los archivos), así como los operadores de ACS, serán los que utilicen el clientes GUI. ACS proporciona un marco de GUI opcional llamado Abeans la cual se encuentra integrada con la version original de ACS, diseñado para escribir GUI de aplicaciones de control. Sin embargo, se ha decidido desarrollar una GUI para las CC con los requisitos especificos de visualización acordados con LLAMA.

5.0.1. Descripción de archivos y directorios

La construcción e instalación del software de ACS, como también la configuración del componente pueden visualizarse en el Apéndice D.
Los archivos y directorios creados donde se aloja el código fuente del componente, se describen en la siguiente tabla:

Cuadro 5.1: Descripción de directorios del componente

Directorios	**Descripción**
idl	Aquí se colocan los scripts y archivos binarios que requieran ser ejecutados.
Changelog	Un archivo que podría usarse para registrar cambios del módulo. Usamos JIRA y "git log"para esto, pero esto podría usarse como un lugar adicional para esta información.
config	Directorio para configuraciones de la CDB del módulo. Puede usarse con la variable de entorno ACS_CDB para configurar una CDB y probar el módulo.Los archivos XML Schema (XSD) tambien se colocan en este directorio.
doc	Directorio para colocar documentación.
idl	Directorio para el archivos IDL de las CC (.idl, .midl), definiciones de errores XML, archivo de enlace de esquema XML.
include	Directorio para encabezados de C++, archivos en línea y de plantilla.
src	Directorio para archivos fuentes.
object	Directorio para archivos fuente con el archivo principal para definir los objetivos que se compilarán, instalarán e implementarán el módulo .
test	Directorio que contiene todos los archivos necesarios para la pruebas del Código fuente, Makefile, archivos de configuración TAT (estos archivos son utilizados para generar simulaciones del componente, sin necesidad de contar con el hardware), archivos de referencia, etc.

5.0.2. Estructura de Desarrollo del Componente

El detalle del código fuente desarrollado para el componte CalibrationLoads se muestra en la figura 5.1.

El repositorio de ACS cuenta con archivos de construcción del proyecto o Makefiles, que son creados automáticamente al momento de crear un componente . El archivo Makefile se ha reformulado agregando los ficheros descritos anteriormente, esto es agregar las bibliotecas para las clases C++, y el uso de excepciones.

Código 5.1: Makefile para la construccion del componente de las CC.

```
1  # Includes (.h) files (public only)
2  # --------------------------------
3  INCLUDES        = Socket.h CLSocket.h CalibrationLoadsImpl.h
4
5  # Libraries (public and local)
6  # ----------------------------
7  LIBRARIES       =  socks CLSocket CalibrationLoadsImpl
8  #LIBRARIES_L    =
9
10 # <brief description of socks library>
11 socks_OBJECTS   = Socket
12 socks_LIBS =  socketError
13 # <brief description of CLSocket library>
14 CLSocket_OBJECTS   = CLSocket
15 CLSocket_LIBS      = cloadsError socketError socks
16 #
17 # <brief description of CalibrationLoads library>
18 CalibrationLoadsImpl_OBJECTS   = CalibrationLoadsImpl
19 CalibrationLoadsImpl_LIBS      = CLSocket cloadsError CalibrationLoadsStubs baci
20 #
21 # Configuration Database Files
22 # ----------------------------
23 CDB_SCHEMAS = CalibrationLoads
24 # IDL Files and flags
25 #
26 IDL_FILES = CalibrationLoads
27 TAO_IDLFLAGS =
28 USER_IDL =
29 CalibrationLoadsStubs_LIBS = baciStubs cloadsError acscomponentStubs
```

Los archivos Makefile indican dos tipos de instrucciones:

- las necesarias para generar las bibliotecas de funciones de cada clase.
- las necesarias para instalar en el directorio `$INTROOT`, las bibliotecas de funciones,los archivos IDL, los archivos XML y los ficheros de cabecera.

Las entradas precedidas por la etiqueta: `INCLUDES` especifican los archivos de cabecera que serán instalados en `$INTROOT/include` y que por lo tanto son públicos y pueden ser incluidos por otros componentes en caso de ser necesario. Las entradas correspondientes a la etiqueta `LIBRARIES` indican las bibliotecas que se van a generar y que se van a instalar en `$INTROOT/lib`.A continuación para cada biblioteca se escriben las etiquetas con el nombre de cada una de estas, seguida de `_OBJECTS` y `_LIBS` que indican los objetos y las librerías de las que dependen y que por tanto son necesarias para construirse.
Durante el proceso de compilación se genera la biblioteca «CalibrationLoads» del componente, que luego serán cargadas en el contenedor `cepiaContainer`.

Figura 5.1: Conjunto de ficheros desarrollados en el componente `CalibrationLoads`.

5.0.3. Descripción del componente de ACS para los dispositivos de Calibración

Los ficheros que contienen el código para la comunicación se han denominado CLSocket.h y CLSocket.cpp, en ellos están, respectivamente, las declaraciones y definiciones de las variables y funciones utilizadas para el control remoto del componente caracteristico `CalibrationLoads` (figura 5.2).

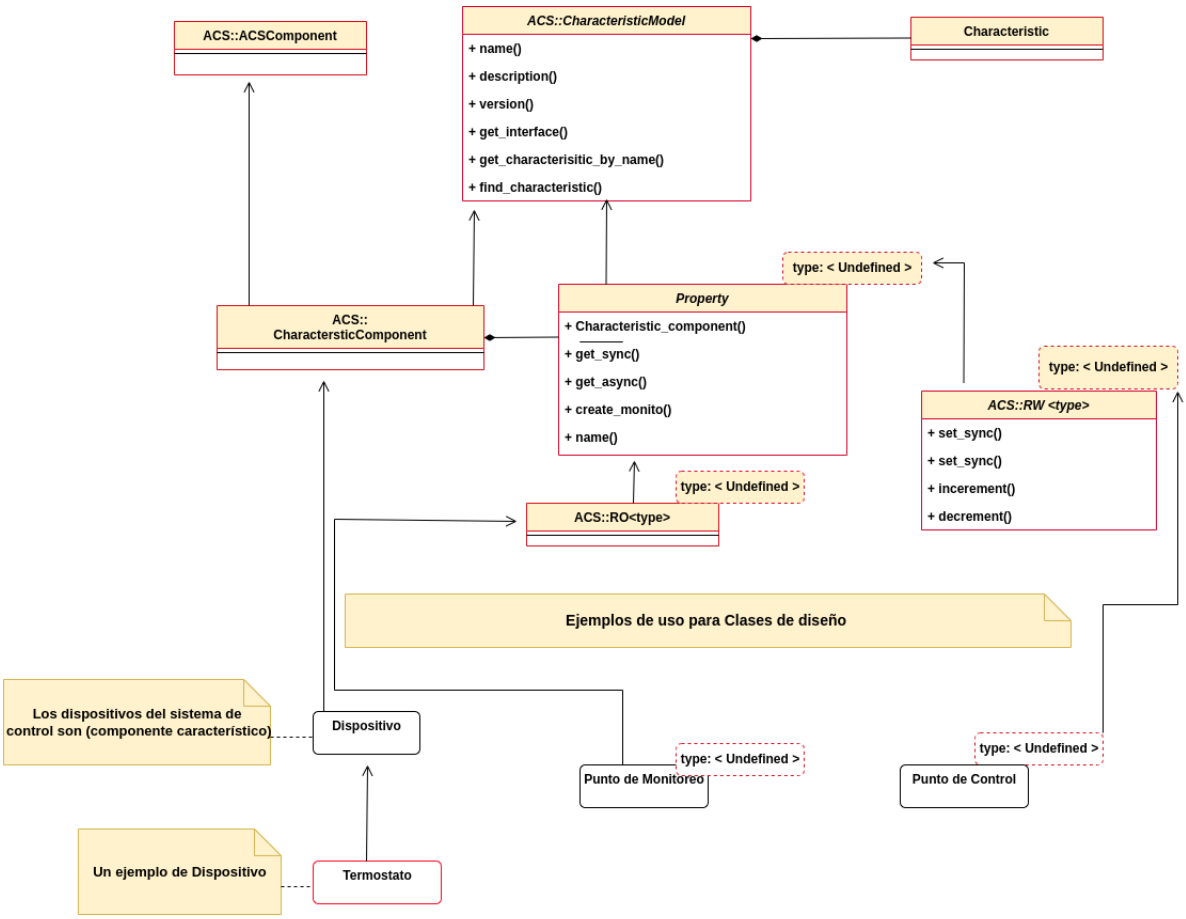


Figura 5.2: Diagrama de clases utilizado por el Componente Característico.EL diagrama muestra las clases de escritura y lectura para las propiedades del isntrumento.

Dado que los instrumentos se controlan a través de un cable ethernet mediante un protocolo TCP/IP, es necesario configurar el puerto y la IP para la comunicación con los dispositivos, estos parámetros han sido incluidos como atributos en la Base de datos, en el archivo CDB.
También es posible definir estos parámetros como propiedades del componente, lo que permitiría que los valores puedan ser modificados directamente desde el inicio de la implementación .
El constructor de la clase CLSocket por medio del método `connect` llama a los valores previamente configurados, para poder establecer la comunicación con los instrumentos, si es que existe algún problema al momento de conectarse, se enviará una excepción alertando de algún error en la construcción del socket, como se puede ver en la siguiente descripción:

Código 5.2: Construccion Clase CLSocket()

```
1  ACS_SHORT_LOG((LM_DEBUG, "Creating the socket to address %s and port %d"ip, port));
2  try {
3          sock = new Socket(ip, port);
4  }
5  catch (socketError::cannotOpenExImpl &ex) {
6  throw cloadsError::cannotConstructExImpl(ex,__FILE__,__LINE__,"CLoads::CLoads");
7  }
8  ACS_SHORT_LOG((LM_DEBUG, "Connecting to socket..."));
9  try {
10 sock->Connect();
11 }
12 catch (socketError::cannotConnectExImpl & ex) {
13 ACS_SHORT_LOG((LM_ERROR, "Error conecting to the socket"));
14 throw cloadsError::cannotConstructExImpl(ex,__FILE__,__LINE__,"CLoads::CLoads");
15 }
```

Las funciones síncronas de escritura y lectura operan mediante los metodos `Read()` y `Write()`, cada vez que se requiera enviar algún comando, se enviará el comando a través del socket y serán recibidos por el servido scpi que se encargará de leer la respuesta desde el lado del instrumento.

Código 5.3: Lectura y Escritura de la clase CLSocket().

```
1  const char * CLSocket::queryIdentification() throw
2  (cloadsError::writeErrorExImpl &, cloadsError::readErrorExImpl &)
3  sprintf(cmd, "*IDN?");
4  try {
5   sock->Write(cmd,strlen(cmd),0);
6   usleep(time_config);
7  }
8  catch (socketError::cannotWriteExImpl & ex) {
9  throw cloadsError::writeErrorExImpl(ex,_FILE_,_LINE_,"CLoads::queryIdentification");
10 } try {
11         sock->Read(response,30,0);
12 }
13 catch (socketError::cannotReadExImpl & ex) {
14 throw cloadsError::readErrorExImpl(ex,_FILE_,_LINE_,"CLoads::queryIdentification");}
```

Los metodos de la clases definen las llamadas de comandos de consulta y los comandos de ejecución . Existe un método para cada comando definido.

En los ficheros del código desarrollado, se requiere indicar la biblioteca en la que están definidas las rutinas, funciones y variables propias del socket, mediante la la clase `Socket()`.

La figura 5.3 muestra un esquema de la clase CLSocket y sus dependencias :

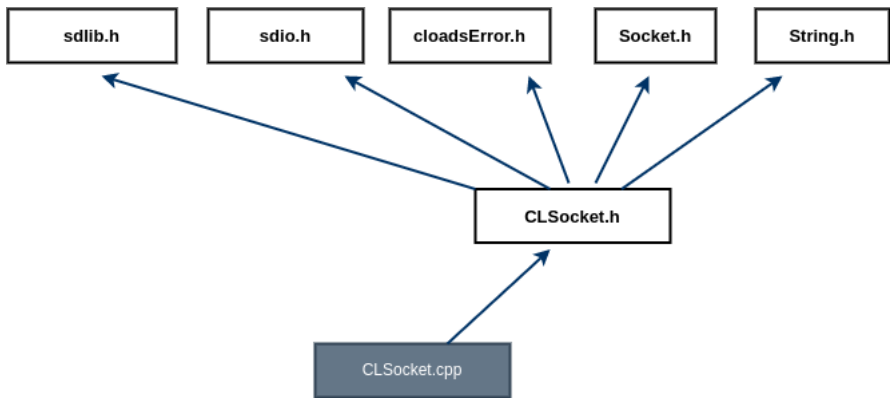


Figura 5.3: Archivos de cabecera de las clases utilizadas por la clase CLSocket.

5.0.4. Errores. Lanzamiento de excepciones

Las excepciones son situaciones anómalas que tienen lugar durante la ejecución de un programa debido, por ejemplo, a la ocurrencia de algún tipo de error. El sistema de errores de ACS precisa que los errores se definan en un fichero xml con el código de cada error, su nombre y descripción. En el componente de ACS del sistema de CC se han definido un conjunto de excepciones en el fichero `cloadsError.xml` y se han identificado con enumeradores .

Cuadro 5.2: Excepciones componente CalibrationLoads.

Exepciones	Descripcion
`cloadsCorrect`	Request sucessfully procesed.
`cannotConstruct`	Error de constructor .
`paramOutOfLimits`	"Parameters sent to Calibration Loads out of limits".
`busyError`	"Busy integrating".
`writeError`	«Error sending data to CalibrationLoads». Esta excepción se genera a partir de una excepción en la escritura sobre un objeto de la clase Socket.
`readError`	«Error receiving data from CalibrationLoads». Esta excepción se genera a partir de una excepción en la lectura de un objeto de la clase Socket.

El Makefile genera automáticamente el código necesario para el manejo de errores y excepciones tanto para el IDL como para todos los lenguajes de programación soportados por ACS.
Tras ejecutar el Makefile se crearán dos nuevos ficheros ACSErrTypeCommon.idl y cloadsError.idl. Ambos se incluirán en el idl del componente (CalibrationLoads.idl) para utilizar las excepciones.

Las excepciones se lanzan desde las clases C++, como de las clase de los DevIOs

y la clase de implementación del componente.

5.0.5. Definición del archivo IDL

El archivo IDL contiene la declaración de las propiedades y los métodos del componente. Esta interfaz es la única forma en que un componente le diga a los clientes los métodos que exponen los dispositivos. Del mismo modo, un componente que conoce la interfaz del otro componente puede utilizar sus métodos. Este archivo se encuentra en el directorio idl y se ha denominado CalibratonLoads.idl.

En el archivo IDL se ha definido un módulo llamado `LOADS_MODULE` y una interfaz denominada CalibrationLoads.idl que hereda de `ACS:CharacteristicComponent`.En el inicio del módulo se define un canal de notificaciones para las propiedades de temperatura de cada sensor. Para esto se ha implementado un mecanismo de suscripción utilizando el servicio de notificación The ACE ORB (TAO) de ACS con eventos estructurados.
Los valores asociados a cada propiedad pueden transmitirse de forma asincrónica y distribuirse a varias aplicaciones del software.

Cada carga de calibración cuenta con 3 propiedades y 7 métodos propios, además en general se cuenta con un método para detener el sistema completo en caso de que se pudiese presentar algún problema y fuese necesario detener el sistema. Además de un método para devolver una cadena que identifica de forma exclusiva a las CC, descrito por el estándar SCPI.

Métodos

Cada dispositivos disponen de un método de encendido/apagado pudiendo ejecutarse de forma individual o a mas de una carga al mismo tiempo . En caso de que algún sensor pudiese fallar es posible cambiar el modo de retroalimentación en los sensores de cada carga, con un modo de retroalimentación para obtener el valor de un sensor especifico. El método para parar el sistema funciona como un interruptor, por lo que debe ser desactivado para que pueda ser posible volver a utilizar el sistema nuevamente.

Propiedades

Cada carga de calibración cuenta con tres sensores. La temperatura de cada sensor se considera una propiedad real (variables tipo double) de solo lectura. Los rangos de valores pueden tomar valores de 0.0 °C hasta 70.0 °C.

Código 5.4: Propiedades del Archivo idl.

```
1        /** Indicates the actual temperature from sensor 1 load 1. */
2        readonly attribute ACS::ROdouble sen1load1;   // CORBA::Double
3
4        /** Indicates the actual temperature from sensor 2 load 1. */
5        readonly attribute ACS::ROdouble sen2load1;   // CORBA::Double
6
7        /** Indicates the actual temperature from sensor 3 load 1. */
8        readonly attribute ACS::ROdouble sen3load1;   // CORBA::Double
9
10       /** Indicates the actual temperature from sensor 1 load 2. */
11       readonly attribute ACS::ROdouble sen1load2;    // CORBA::Double
12
13       /** Indicates the actual temperature from sensor 2 load 2. */
14       readonly attribute ACS::ROdouble sen2load2;    // CORBA::Double
15
16       /** Indicates the actual temperature from sensor 3 load 2. */
17       readonly attribute ACS::ROdouble sen3load2   //CROBA::Double
18
19       /** Indicates the actual temperature from sensor 1 load 3. */
20       readonly attribute ACS::ROdouble sen1load3;   //CORBA::Double
21
22       /** Indicates the actual temperature from sensor 2 load 3. */
23       readonly attribute ACS::ROdouble sen2load3;   //CORBA::Double
24
25       /** Indicates the actual temperature from sensor 3 load 3. */
26       readonly attribute ACS::ROdouble sen3load3;   //CORBA::Double
```

5.0.6. Implementación Componente: servidor

La implementación del componente se hace a través de los métodos de lectura y escritura de las clases de los DevIO y de los método síncronos. Los DevIOs se asocian a las propiedades y permiten establecer u obtener los valores de cada propiedad. La

funcionalidad del DevIO se define e implementa en un fichero de cabecera . Todos los DevIOs definidos heredan de la clase DevIO$< T >$ una plantilla que permite reimplementar sus métodos read y write, encargados de obtener o establecer valores en las propiedades respectivamente. Mediante el modelo de la clase heredada se indica el tipo de archivo DevIO. Los métodos de lectura y escritura crean excluciones mutuas (mutex) para impedir la modificación de las variables privadas.
Las propiedades de solo lectura están asociadas a la temperatura de los sensores y su valores se obtienen consultando las funciones privadas correspondientes a la clase CLSocket(), si los modos de encendido en cada carga se encuentran desactivados, esto generará una excepción , si es que los modos se encuentran activados las instrucciones serán enviadas a los dispositivos.

Se ha desarrollado una clase C++ `CalibrationLoadImpl` que implementa todas la funcionalidad del sistema de las CC. El ciclo de vida del componente en ACS también queda definido en esta clase, este requiere ser definido y desarrollado a través de las funciones: `initialize()`, `executer()`, `clean()` y `stop()`.Estas funciones son manejadas por el contenedor y definen el estado en el que se encuentra el componente.

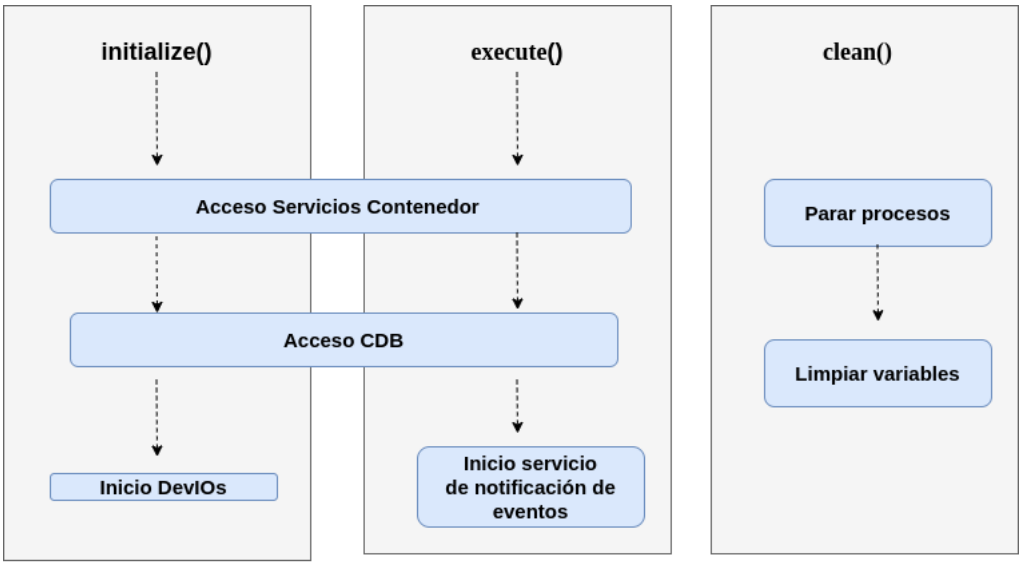


Figura 5.4: Ciclo de vida del componente de las CC.

El método initialize() de la clase CalibrationLoadsImpl() instancia los DevIOs y crea las propiedades. Las propiedades obtienen su valor predeterminados tras consultar la base de datos (CalibrationLoads.xml) empleando el método write() de los DevIO. La implementación de los métodos definidos en el IDL también se realiza en esta clase. Igualmente este método crea un objeto de la clase CLSocket() para tener acceso a todas sus funciones públicas. El método execute() iniciara un hilo de ejecución que pone en marcha el método getData() de la clase C++, este proceso lee las cadenas que envía el sistema constantemente. Mientras el hilo principal queda a la espera de las peticiones de los clientes. Por ultimo los métodos clean()/stop() detendrán todo lo procesos relacionados con el componente limpiando las variables utilizadas y destruyendo todo los objetos creados .

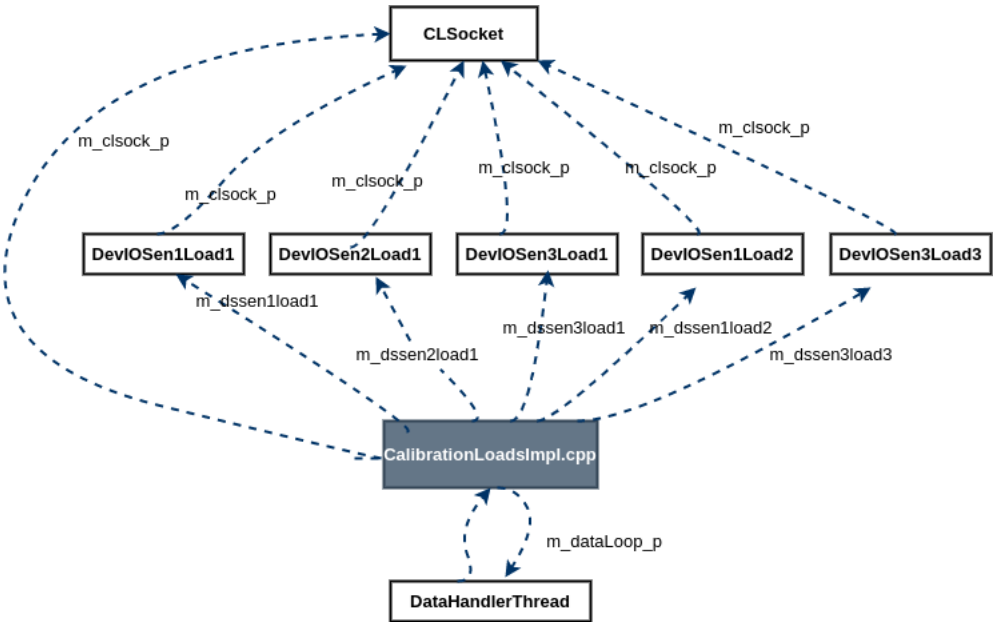


Figura 5.5: Diagrama de clases para CalibrationLoaddImpl.

5.0.7. Configuración inicial de las instancias.Archivos de la Base de Datos

El componente CalibrationLoads puede crear varias instancias, aunque como solo disponemos de un solo sistema de cargas de calibración solo existirá una instancia en este caso.

Los valores predefinidos con los que se caracteriza el componente se guardan en el

archivo `CalibrationLoads/CALIBRATIONLOADS.xml` . En este directorio se ubica la base de datos de todos los componentes del radiotelescopio LLAMA. La aplicación `cdbBrowser` permite navegar a través de los distintos componentes y sus propiedades .

Al momento de inicializar el componente, se requiere especificar y configurar el contenedor como parámetros asociados al servicio de notificación de CORBA.Todos los ficheros desarrollados para poder instanciar los dispositivos de calibración desde el software se detallan en la siguiente figura:

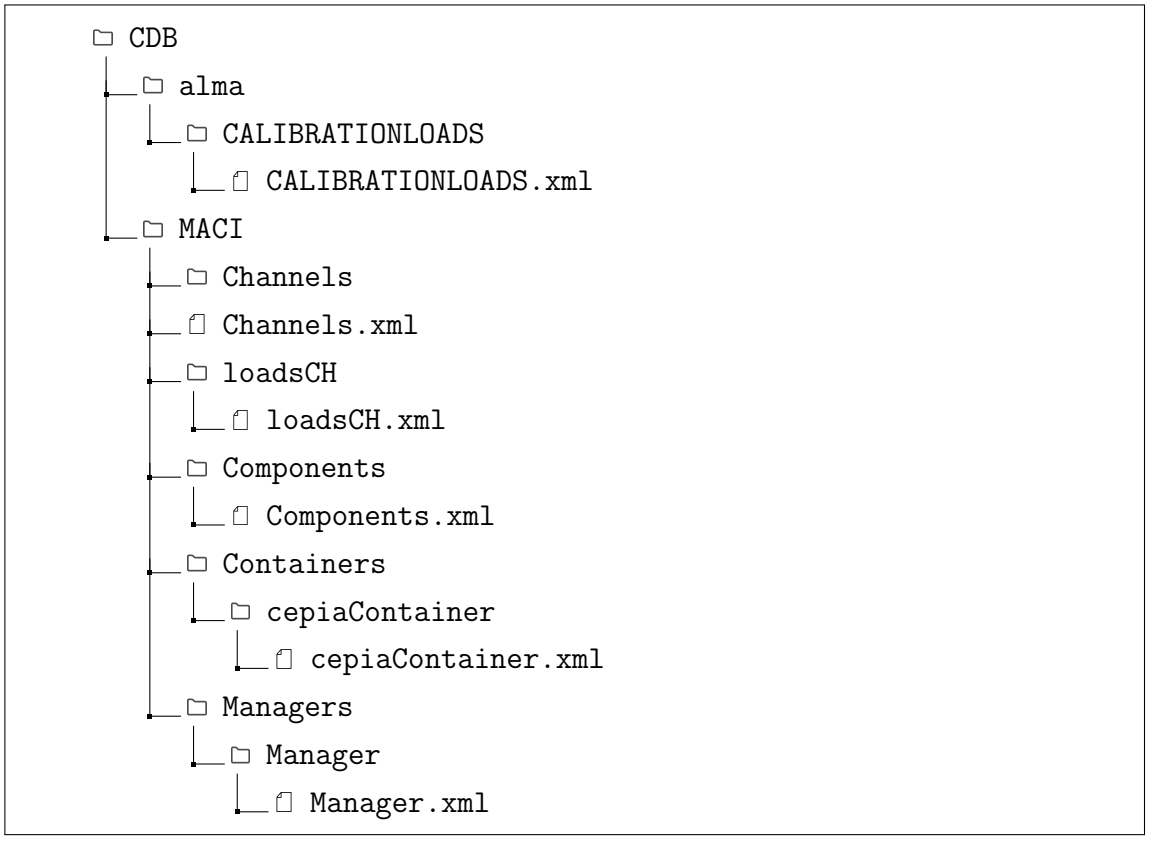


Figura 5.6: Conjunto de ficheros desarrollados para instaciar el componente.

La figura 5.7 muestra el contenido de los valores para la propiedades de la instancia CALIBRATIONLOADS .

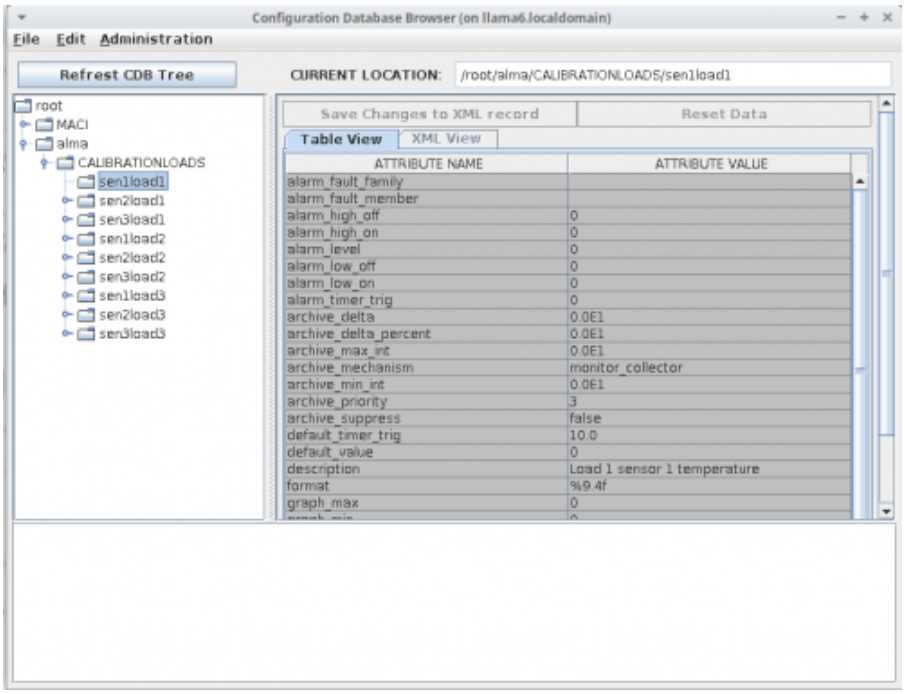


Figura 5.7: Configuración de los valores de las propiedades en cdbBrowser.

La aplicación cdbBrowser es una Interfaz gráfica de ACS que permite modificar los valores asociados a cada propiedad, de una forma simple seleccionando dos veces sobre cada campo, y pulsando después sobre «Save Changes to XML Record». Los principales campos a tener en cuenta para la temperatura de cada sensor son:

- description . Descripción de la propiedad.
- units . Unidades, que en el caso de la temperatura se utiliza grados Celsius.
- format . Formato de los datos.
- alarm_high_on y alarm_high_off . Por encima de 70 °C se activa una alarma y se desactiva por debajo de 70 °C.
- alarm_high_on y alarm_high_off . Por debajo de 0 °C se activa una alarma y se desactiva por encima de 0 °C.
- default_timer_trig, min_timer_trig, delta_timer_trig . En el caso de utilizar una propiedad de monitoreo y no fijar el intervalo del cronómetro este vale 60 segundos y el mínimo aceptado es de: 10 segundos.

5.1. Implementación Cliente

EL software de ACS proporciona clientes genéricos, permitiendo el desarrollo en lengajes de programación como python, Java y C++ independiente del lenguaje que se haya utilizado para la implementación del servidor.El cliente puede acceder a las propiedades y métodos de las CC como también a propiedades y métodos de otros componentes como ya se ha comentado con anterioridad.

En el caso de las CC el cliente gráfico se ha implementado utilizando C++, con el objetivo de facilitar el control y la visualización de los valores de cada propiedad en los dispositivos de calibración.

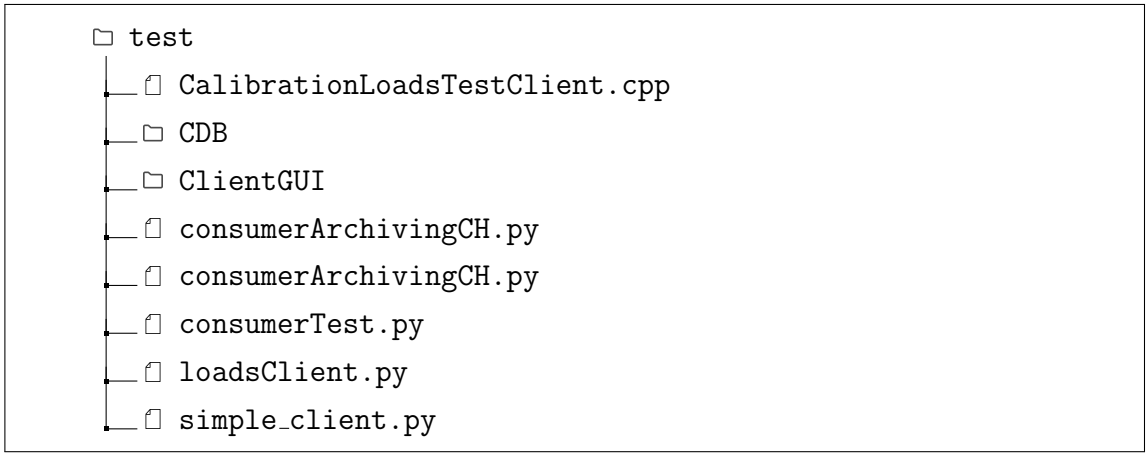


Figura 5.8: Organización de ficheros y directorios de la carpeta test.

La figura 5.8 muestra la estructura de directorios con los archivos de implementación del cliente, la carpeta ClientGUI contiene al cliente gráfico C++ .
Adicionalmente se desarrollaron algunas pruebas utilizando el lenguaje python para evaluar el despliegue del componente y de algunas funcionalidades de ACS, como por ejemplo el servicio de eventos y el canal de notificaciones. EL directorio CDB contiene la configuración de la instancias las cuales han sido descritas en la sección anterior.

5.1.1. Cliente C++ para Cargas de Calibración

Se ha creado el directorio `ClientGUI`, siguiendo el formato de estructura de ACS para los directorios.El desarrollo de las clases y los objetos necesarios para construir la GUI se muestran a continuación:

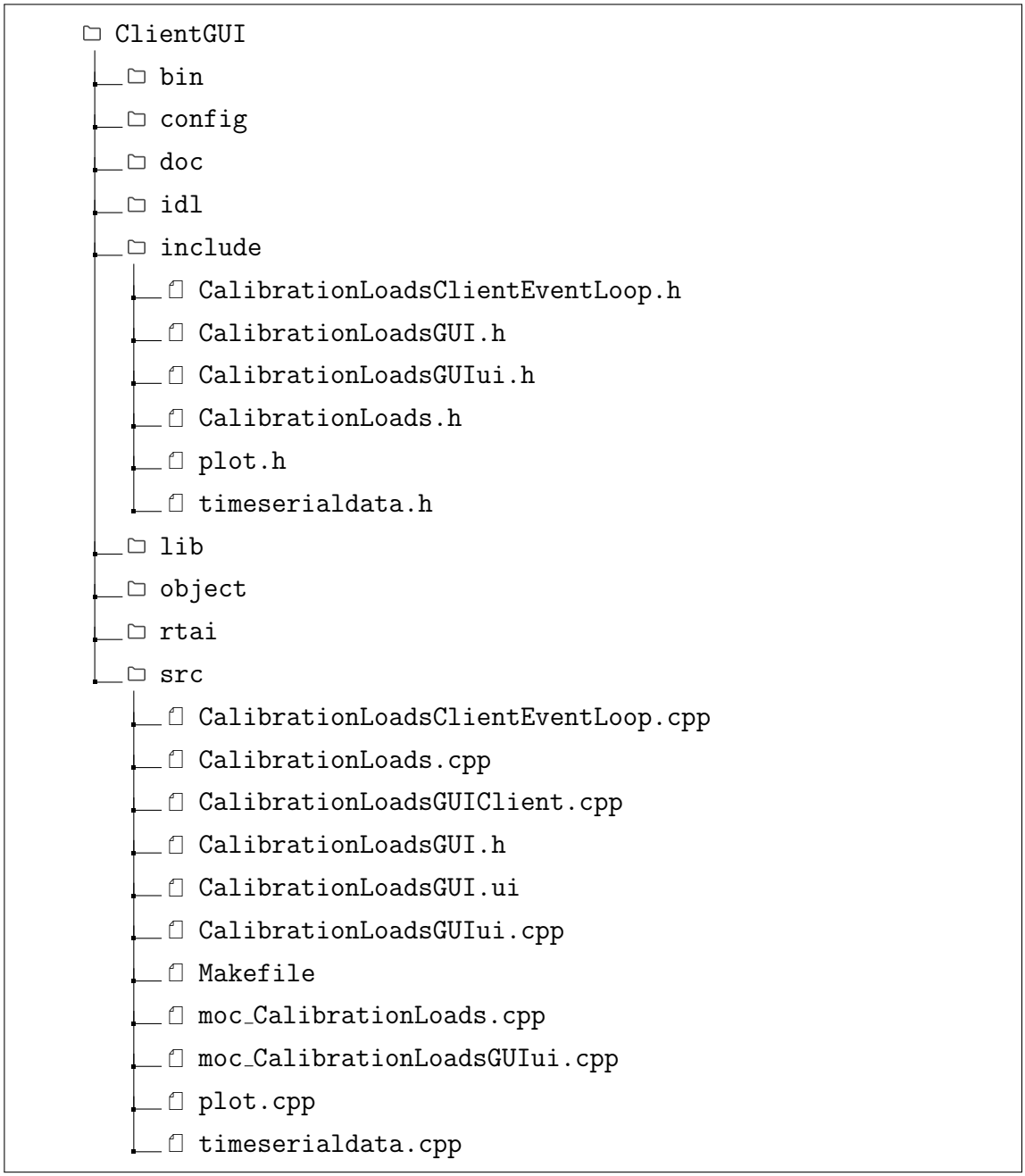


Figura 5.9: Ficheros desarrollados para la implementación de la GUI.

Como se ha comentado, para el desarrollo del cliente se ha seleccionado el lenguaje de programación C++ y el diseñador gráfico de Qt versión 4.6.2, pues una de sus principales ventajas es la facilidad en la instalación de las bibliotecas, y la construcción de las aplicaciones dentro del sistema Centos 6.6.10 en relación a las bibliotecas de ACS LLAMA.

En primer lugar es necesario generar las clases C++ asociadas al diseño gráfico . El archivo generado por QT Designer [3], utiliza un formato UI descrito con un esquema XML y que se denomina `CalibrationLoadsGUI.ui` .
A continuación se emplea la herramienta pyuic que trasforma los archivos de interfaz de usuario en una clase C++ utilizando las bibliotecas de Qt enlazadas al lenguaje de programación C++ especificamente.
En el archivo Makefile, correspondiente al cliente se ha incluido una entrada para realizar automáticamente esta operación:

```
UItoC++: ../CalibrationLoads/test/ClientGUI/src/focuser.ui
pyuic ../CalibrationLoads/test/ClientGUI/src/CalibrationLoadsGUI.ui -o
../CalibrationLoads/test/ClientGUI/src/CalibrationLoadsGUIui.h
```

El cliente implementa una interfaz gráfica que permite interactuar con las cargas de calibración de un modo sencillo y con una visualización de todas las características de las CC. El desarrollo de la interfaz, mostrada en la figura 5.10, incluye un panel de control para cada carga donde es posible controlar la temperatura de cada calefactor, igualmente se incluye un gráfico por carga que muestra las curvas de temperatura por sensor, también se considera el promedio de temperatura de los tres sensores.Los demás botones corresponden a los métodos de retroalimentación, mientras que los valores de las propiedades de temperatura pueden ser visualizados por medio de los widget LCD .

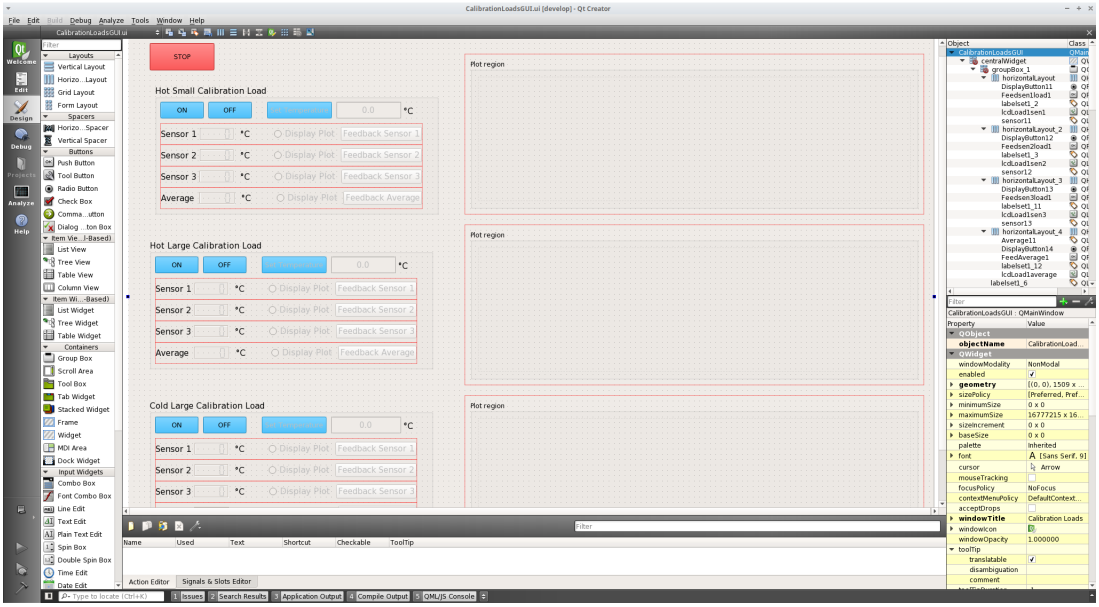


Figura 5.10: Aplicación de software QT Creator para el diseño de GUI.

5.1.2. Clases de Objetos

Para el cliente se han desarrollado los siguientes archivos, que se describen a continuación:

- `CalibrationLoadsGUIClient.cpp` corresponde al código principal, donde se crea el objeto de la clase SimpleClient() (para hacer referencia al cliente de ACS), y el objeto para la clase del component «CalibrationLoads», la instancia de tipo QApplication es pasada al cliente para que pueda inicializar la interfaz .

Código 5.5: Inicializacion de la aplicacion QT para el cliente de ACS.

```
1  QApplication a(argc,argv);
2  CalibrationLoadsGUI* myCalibrationLoadsGUI = new CalibrationLoadsGUI;
3  SimpleClient* client= new SimpleClient;
4
5  if(client->init(qApp->argc(),qApp->argv())==0)
6  {
7      /*if no arguments are given */
8          ACE_DEBUG((LM_DEBUG,"Cannot init client"));
9          delete client;
10         goto CloseLabel; }
```

- `CalibrationLoadsClientEventLoop.cpp` Esta clase proporciona una forma de administrar los subprocesos de control, actualizando los valores de las propiedades que serán monitorizadas.De forma predeterminada, el método run() inicia el bucle de eventos llamando a exec(), y stop para parar el bucle de eventos que se este ejecutando en el hilo.

- `CalibrationLoads.cpp` La clase CalibratioLoads hereda de la clase CalibrationLoadsClientEventLoop actualizando los valores de entrada de las temperaturas y obteniendo el valor del tiempo asociado en nanosegundos. Adicionalmente en esta clase se definen los métodos para fijar y leer las propiedades.Como por ejemplo:

Código 5.6: Metodo para obtener valores de temperatura en el tiempo.

```
1 void CalibrationLoads::appendSen2Load1(CORBA::Double sensor12, int time)
2 {
3         nwval12 = sensor12;
4         time12 = time;
5         emit appendsen2load1Value();
6         }
```

- `plot.cpp` y `TimeSerial.cpp` El archivo plot.cpp contiene una clase con todos los atributos de los gráficos y las curvas de temperaturas. El archivo TimeSerial.cpp contiene la clase TimeSerialData() que se encarga de generar la linea de tiempo del eje x en cada gráfico.

- `CalibrationLoadsGUIui.cpp` Finalmente la clase CalibrationLoadsGUIui(), se asocian los eventos gráficos, «signals» o senales a cada método de la clase CalibrationLoads() que se han considerado en la implementación como botonones, inicializando los valores de los «widgets» tras leer el estado del componente a través de los métodos de lectura de las propiedades definidas en CalibrationLoadImpl del servidor.

Para la confección de la interfaz se han hecho múltiples modificaciones al archivo makefile clásico que entrega ACS, para poder integrar el compilador de metaobjetos (moc) encargado de maneja las extensiones C++ de Qt.

Código 5.7: Configuracion archivo Makefile para el cliente grafico utlizando Qt4

```
1  # TARGETS
2  $(QT_UI_FILES_H): ../include/%.h: %.ui $(QT_UI_FILES_UI)
3  /usr/lib64/qt4/bin/uic-qt4 -o $@ $<
4
5  #$(QT_UI_FILES_CPP): %.cpp: ../include/%.h $(QT_UI_FILES_UI)
6  #       $(QTDIR)/bin/uic -o $@ $< $(subst .cpp,.ui,$@)
7  #$(QT_UI_FILES_MOC): moc_%.cpp: ../include/XClient.h
8  #       $(QTDIR)/bin/moc -o $@ $<
9  $(QT_UI_FILES_EXTERN_MOC_H): moc_%.cpp: ../include/%.h $(QT_UI_FILES_EXTERN_H)
10 /usr/lib64/qt4/bin/moc -o $@ $<
11
12 $(QT_UI_FILES_EXTERN_MOC_CPP): moc_%.cpp: %.cpp $(QT_UI_FILES_EXTERN_CPP)
13 /usr/lib64/qt4/bin/moc -o $@ $<
14
15 #qt:     $(QT_UI_FILES_H) $(QT_UI_FILES_CPP) $(QT_UI_FILES_MOC)
16 $(QT_UI_FILES_EXTERN_MOC_H) $(QT_UI_FILES_EXTERN_MOC_CPP)
17
18 qt:     $(QT_UI_FILES_H) $(QT_UI_FILES_MOC)
19  $(QT_UI_FILES_EXTERN_MOC_H) $(QT_UI_FILES_EXTERN_MOC_CPP)
20 @echo " . . . 'qt' done"
```

El makefile generará el ejecutable de la GUI, que se muestra en la figura 5.11.
La imagen muestra el cliente gráfico definitivo funcionando con valores aleatorios en cada sensor. que se han dispuesto para probar la aplicación .

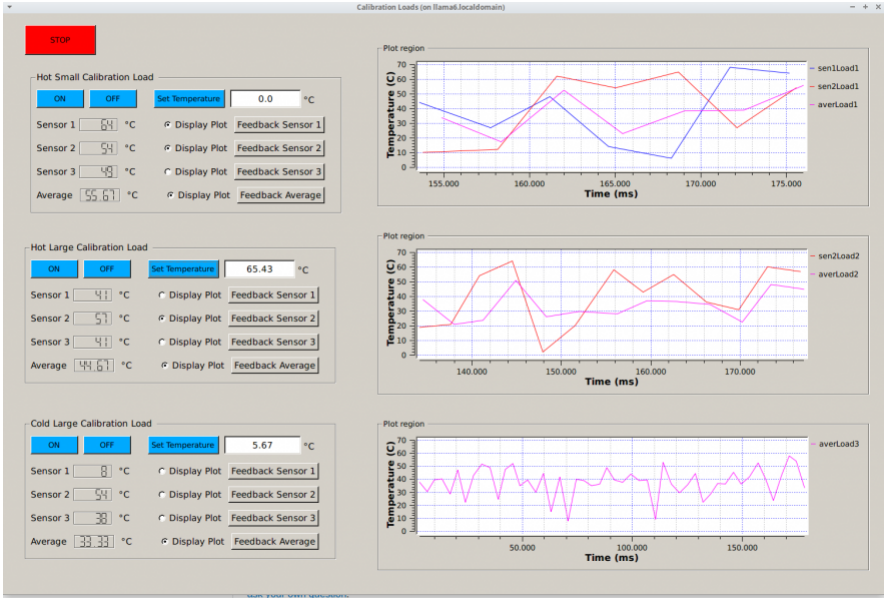


Figura 5.11: Captura de prueba del cliente gráfico.

Capítulo 6

Pruebas y Resultados

Los resultados obtenidos en la implementación del sistema en su conjunto, se detallan y discuten en este capítulo. Se analiza el funcionamiento de las Cargas de Calibración, con sus subsistemas básicos y se revisa el cumplimiento de las funcionalidades esperadas para los instrumentos en términos de control. Las pruebas se ejecutan sobre las versión 1.1 del sistema de control. Por último se determina si la solución es adecuada para los requerimientos del radiotelescopio.

6.1. Pruebas de Integración

Con las pruebas de integración del sistema, se busca determinar el cumplimiento de los requerimientos operacionales del proyecto, especialmente aquellas partes que dependen directamente de la implementación del software. Para esto, se han integrado los dispositivos de calibración con el sistema de control de los instrumentos y el software, haciendo que funcione durante un tiempo prolongado para poder analizar las respuestas del sistema y los servicios de ACS.

6.1.1. Configuración

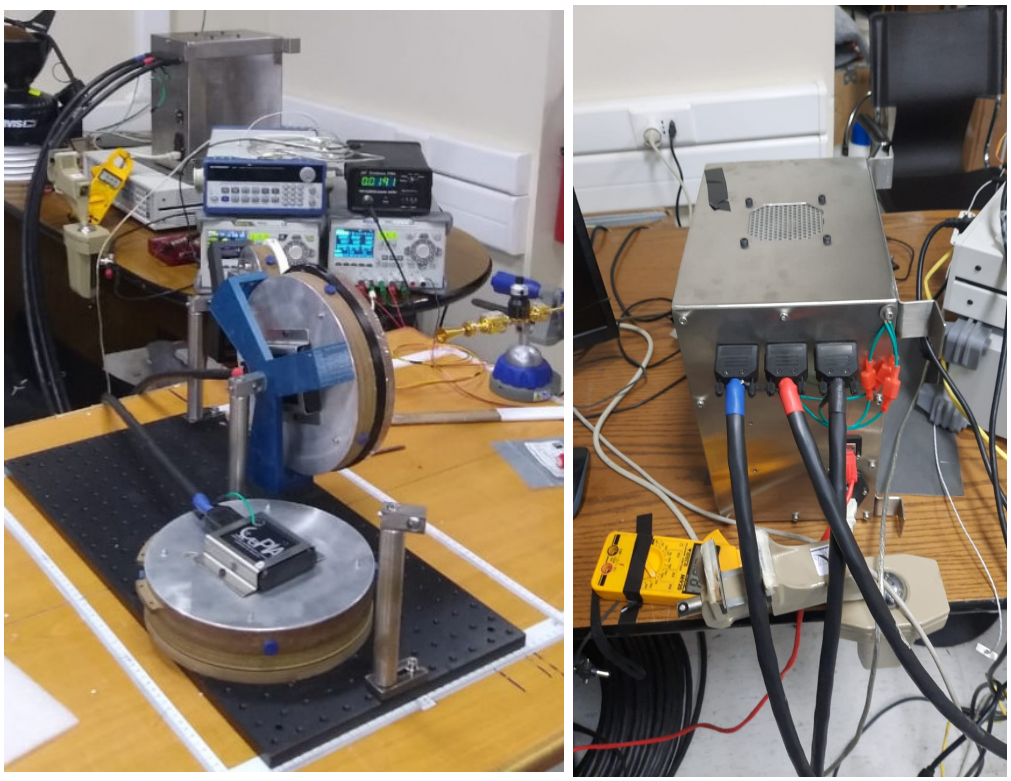
Las herramientas utilizadas para el desarrollo de la prueba consisten en:

- El módulo central, con las tres cargas de calibración,Carga grande fría, carga caliente grande, carga caliente chica.
- La caja de control, compuesta por la tarjeta FPGA, la tarjeta raspberry pi 4b y los componente electrónicos.
- Un cable de alimentación que se conecta a la caja de control.
- 3 cables DB9 .
- Radiómetro de Vapor de Agua en 183 GHz.
- Computadora con VirtualBox 6.6 instalado.

El montaje de las pruebas, como se muestra en la figura 6.2, consiste en mantener las cargas funcionando para estabilizar su temperatura siendo controladas y monitoreadas de forma remota, por medio del software de ACS.Para corroborar el contro lde temperatura se utiliza el Radiómetro de 183 GHz.
Las cargas son conectadas a través de cables DB9 hacia la caja de control 6.1, la cual contiene las tarjeta FPGA y la tarjeta Raspberry pi ya configuradas .El servidor SCPI comenzará a escuchar en el puerto seleccionado (en este caso 5025), reaccionando a los comandos SCPI que puedan ser recibidos.

El sistema operativo Centos 6 corre de forma virtualizada en una computadora anfitriona, simulando la computadora central de la antena la ABM, para replicar el escenario original, en el que correrá ACS .

El software funciona en la configuración de red de VirtualBox , donde puede conectarse a los dispositivos que se encuentren en la misma red local.



(a) Cables DB9 conectados a las CC. (b) Caja de control de las CC.

Figura 6.1: La figura a y b muestran las conexión de las CC con la caja de control, mediante cables DB9.

Las primeras pruebas llevadas acabo, se ejecutan con el fin de verificar la respuesta del servidor de las cargas y la correcta información recibida desde el sistema de control.

EL protocolo de red para la comunicación entre el controlador y los dispositivos a través de una red TCP/IP, permite que una aplicación (cliente) pueda llamar algún procedimiento en los instrumento de medición, de forma remota (servidor), accediendo a su dirección IP.

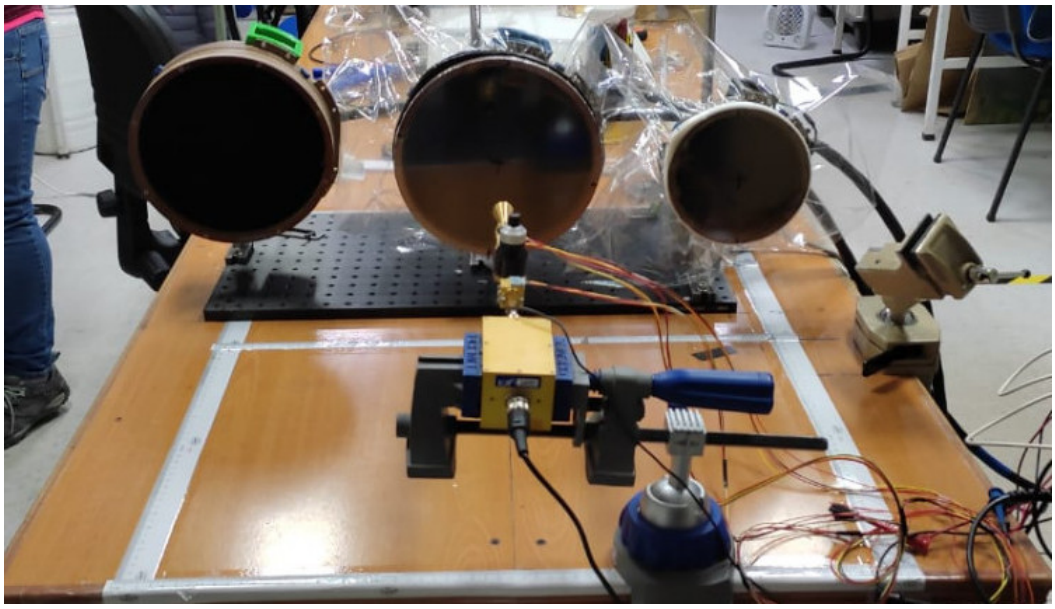


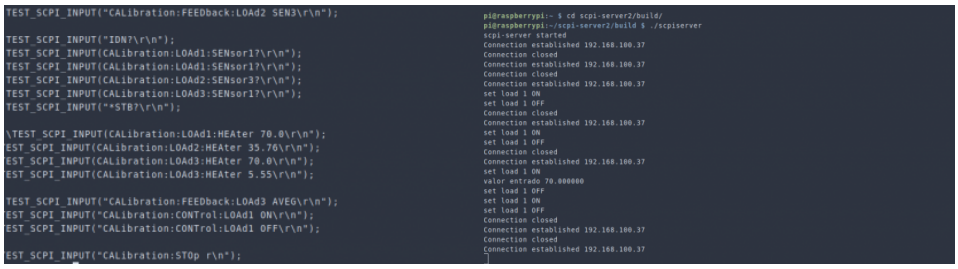
Figura 6.2: Montaje de las tres cargas de calibración.

6.1.2. Pruebas y Resultados

Los resultados obtenidos son utilizados para verificar los requerimientos hecho por LLAMA con respecto a las CC, especificamente, aquellos que dependen directamente del software de ACS.

Comunicación con el Instrumento.

Las primeras pruebas se establecen mediante una conexión socket con los instrumentos, enviando comandos SCPI desde una terminal de forma interactiva para verificar la respuesta en cada caso.
Se puede observar en el registro de la terminal, en la figura 6.3 , donde se indica la ejecución del comando «`CALibration::LOAd2:HEAter1 70.0`» para controlar la temperatura de la carga caliente grande. Así mismo se ejecuta el comando de lectura para el sensor 1 de esta misma carga «`CALibration:LOAd2:SENsor1?`» obteniendo el valor esperado.
Estas pruebas se ejecutan con todos los comandos de monitoreo y control de los dispositivos de calibración. También se verifican los mensajes de error del estándar en caso de que algún parametro no corresponda con los configuración establecida o la sintaxis del mensaje sea erronea .



(a) Comandos SCPI para las CC. (b) Ejecución de servidor para comandos SCPI.

Figura 6.3: Prueba de comunicación con los instrumentos por medio de comandos SCPI con el servidor TCP/IP SCPI de los instrumentos.

La correcta ejecución de estos comandos, como se muestra en las imágenes 6.3 implica una comunicación activa con los dispositivos, así como una configuración adecuada de los comandos SCPI descritos en el servidor.

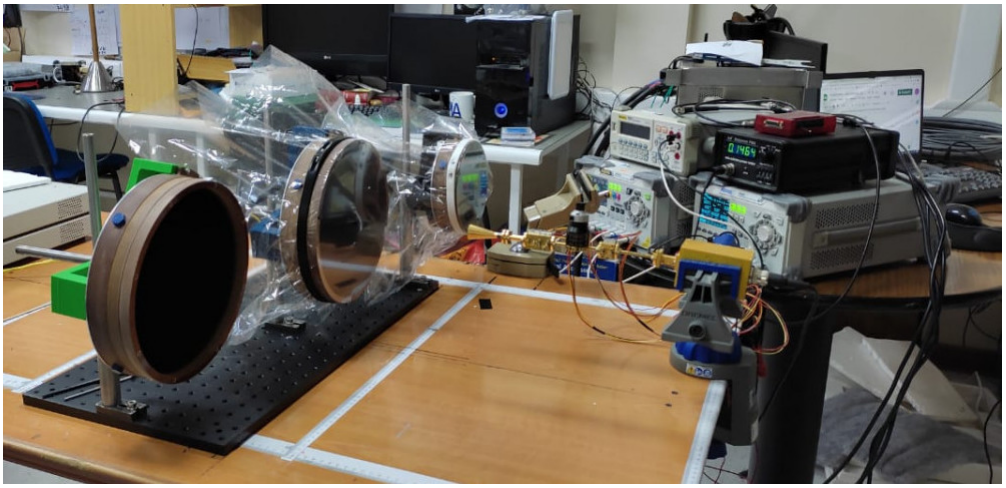


Figura 6.4: Montaje con las tres cargas de calibración.

6.1.3. Software de Control ACS

Para poder efectuar las pruebas con el software ACS, es necesario la instalación del componente en el directorio INTROOT y la configuración de la base de datos . Desde la carpeta del codigo fuente del componente se ejecuta los siguientes comandos :

```
almamgr@llama6:~$ make clean all install
almamgr@llama6:~$ cd llama-src/CONTROL/Device/HardwareDevice/
CalibrationLoads/test/
almamgr@llama6:~$ export ACS_CDB=$PWD
```

La integración del componente al sistema, permitirá el acceso a su interfaz con todas las propiedades y metodos de los dispositivos.
Una vez se haya ejecutado la copilación del componente con el sistema del software, se inicializa ACS. La forma de inicializar la aplicación puede ser por medio de la interfaz del Centro de Comandos o por medio de forma directa utilizando la terminal .
Para inicializar el software con la GUI se debe utilizar el comando `acscommandcenter`.
EL centro de comandos de ACS que se muestra en la siguiente figura :

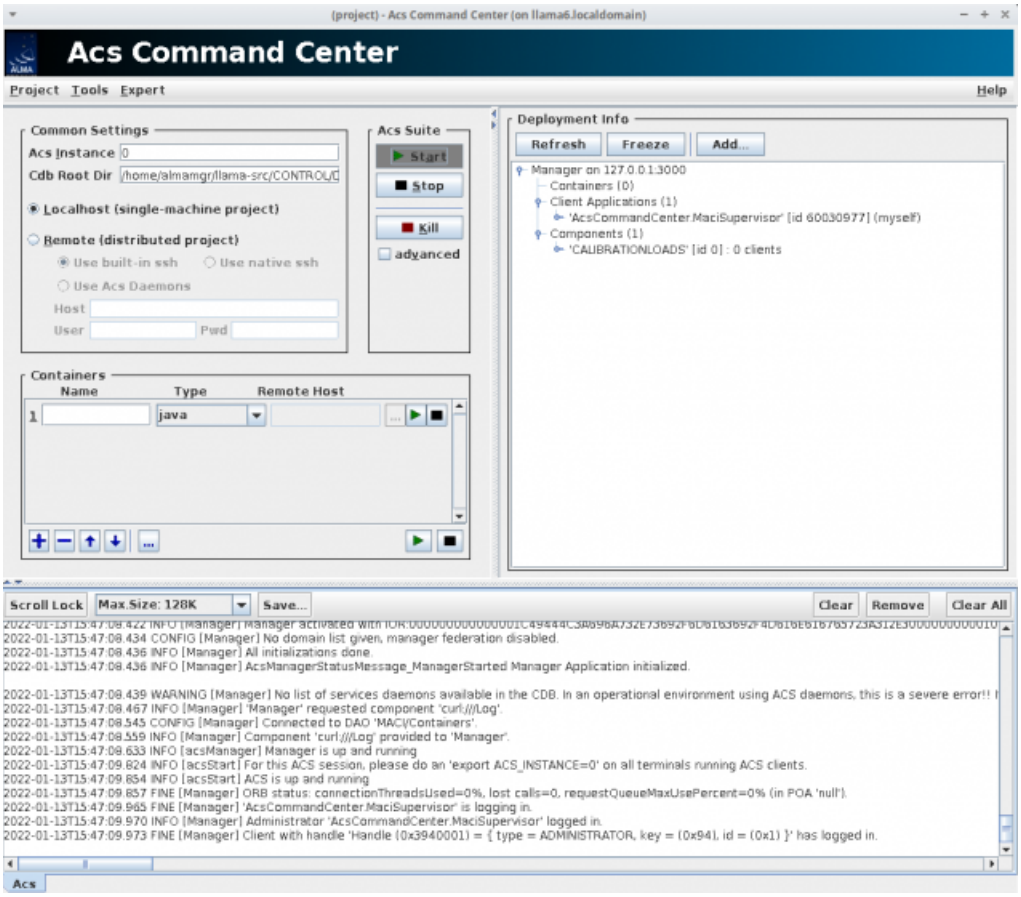


Figura 6.5: Interfaz del panel de Centro de Comandos de ACS.

Mientras la GUI este funcionando, el sistema de registro mostrará todas las operaciones que se ejecuten, esto es de gran ayuda en caso de que se presenten errores, por lo que es necesario mantener esta venta ejecutándose:

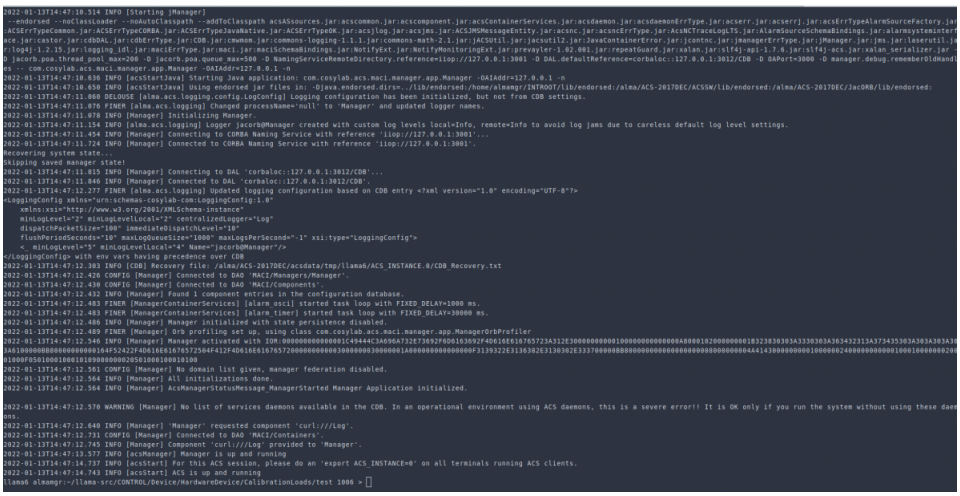
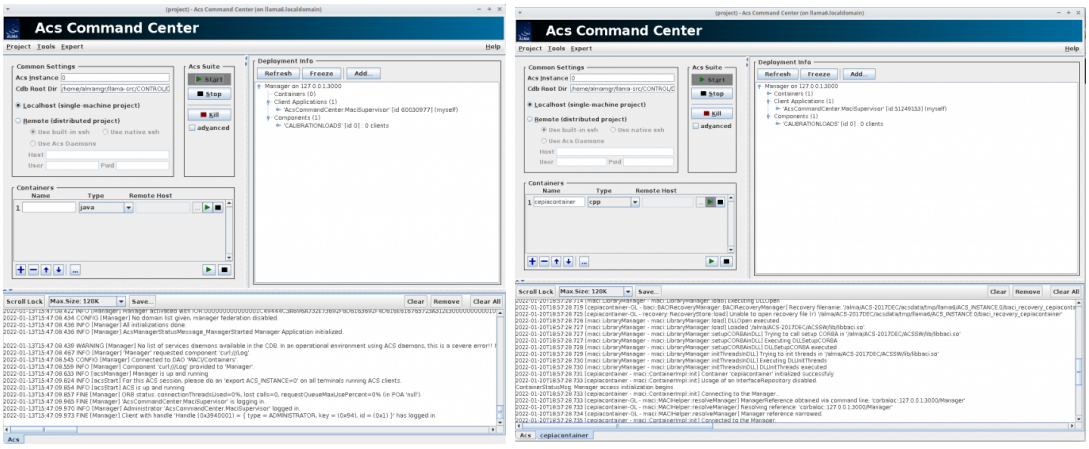


Figura 6.6: Terminal con el registro de la inicialización del Software

Con ACS corriendo, se configuran los campos asociados al contenedor, para que pueda ser desplegado:

- Nombre del contenedor.
- Lenguaje de Programación.



(a) Inicialización del software de ACS. (b) Despliegue del contenedor y componente.

Figura 6.7: Configuración del centro de comandos de ACS para inicializar el contenedor y el componente, asociado a las CC.

La inicialización del contenedor `cepiacontainer` en C++, creará las instancias desarrolladas para el componte. El contenedor se encarga de llamar a los métodos estándar del ciclo de vida del componente (inicializar, ejecutar, limpiar, abortar) para realizar un seguimiento de las condiciones de salud y activación . La figura 6.7 muestra la correcta inicialización y ejecución del componente, estableciendo la comunicación con los dispositivos de calibración y la ejecución del hilo de lectura para las propiedades de temperatura de cada sensor.En este determinado caso se ha utilizado la opción para correr un contenedor de forma local, pero esta opción también puede configurarse para correr algún contenedor de forma remota.

Una vez que el software de ACS está activo y el Contenedor este siendo ejecutando, es posible probar las aplicaciones para las cargas de calibración, utilizando los servicios más importantes proporcionados por la interfaz de ContainerServices, algunos de estos son:

1. Explorador de objetos.
2. Panel de registro.
3. Panel de alarma .

Es posible ver y manipular el componente de control con sus propiedades, utilizando el panel Explorador de objetos, para esto se selecciona la opción `Herramienta` → `Explorador de objetos`, esta muestra todos los componentes contenidos en el contenedor que desplegado, como se puede visualizar en la siguiente ventana :

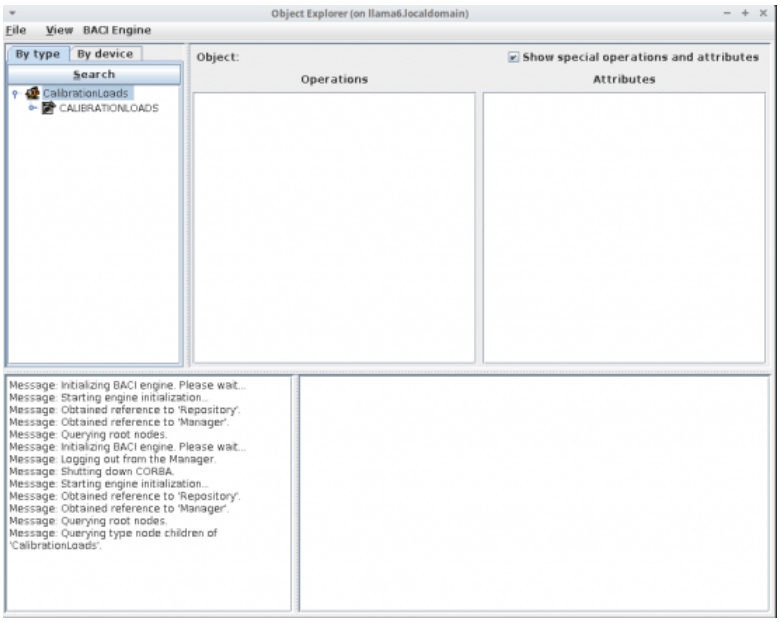


Figura 6.8: Panel Explorador de Objetos.

El contenedor desarrollado en este caso , solo contiene el componenete para los dispositivos de calibración.

El panel de registro se abre seleccionando `Herramienta` → cliente de registro (gráfico).El contenido muestra todos los registros generados tanto por el administrador de ACS como por el componente creado, un ejemplo se muestra en la figura 6.9 .

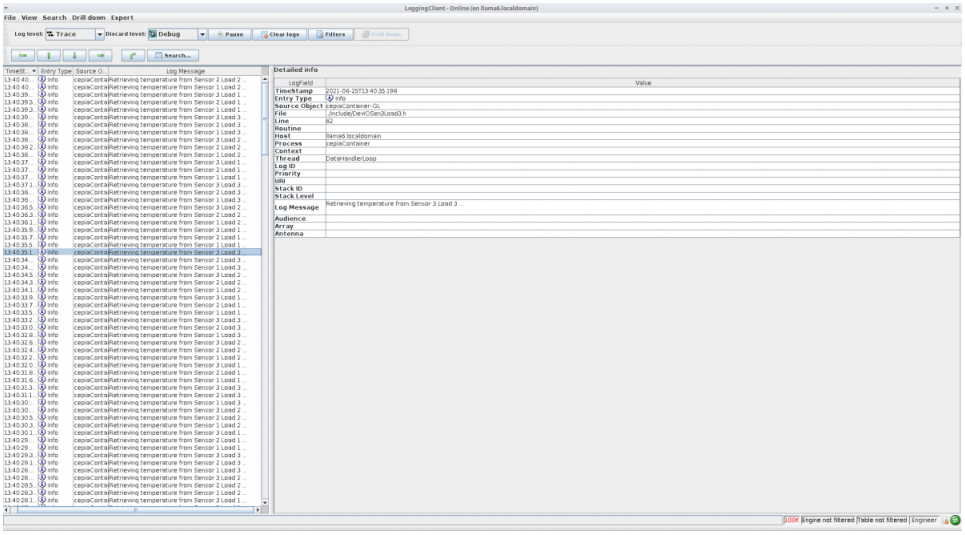


Figura 6.9: Ventana con el inicio de sesión. Detalla el registro para el ciclo de vida del componente.

La tabla del panel de registro, puede obtener y guardar las distintas operaciones y eventos realizados, informando la marca de tiempo en la que se crea el registro, el tipo, la fuente y el mensaje. El registro es muy útil para monitorear la ejecución del componente, pero es más importante que no se exceda, de lo contrario, el producto generará demasiados datos y muchos serán difíciles de manejar.

Por último, se prueba la GUI desarrollada con Qt. El cliente de ACS tiene acceso al componente por medio del despliegue del contenedor .

La aplicación se inicializa con el comando `./CalibrationLoadsTestClient`, que desplegara la interfaz del componente.
Para poder establecer el correcto funcionamiento de la aplicación se ha llevado acabo una prueba de forma prolongada, con una duración de aproximadamente 24 horas.

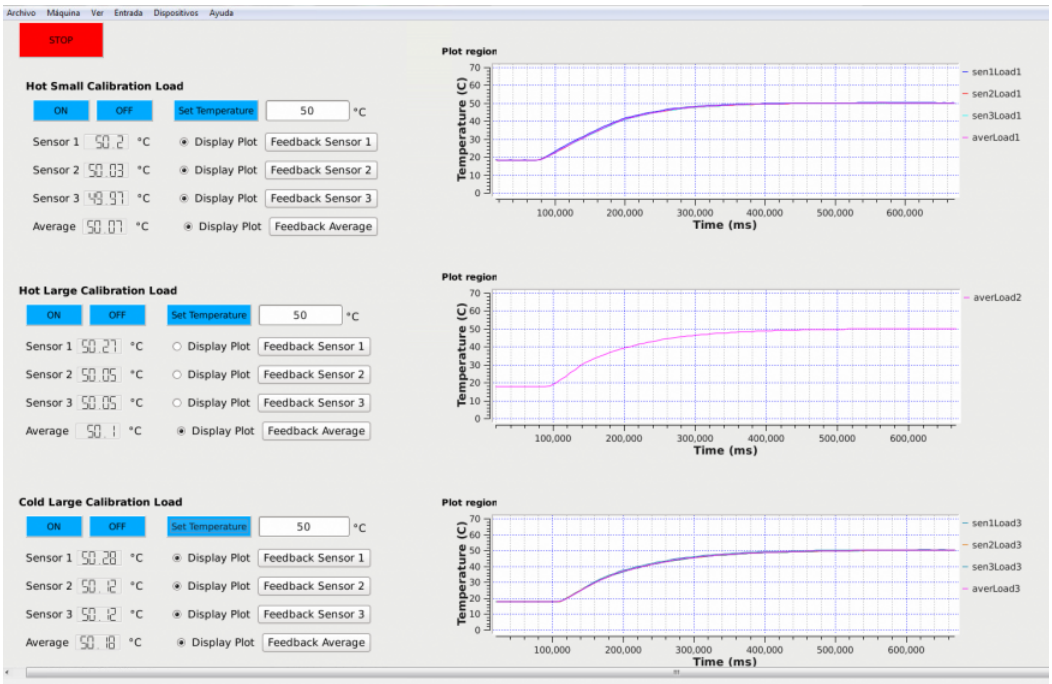


Figura 6.10: Control en tiempo real de las CC mediante la GUI desarrollada para el componente.Donde la temperatura seleccionada para cada carga, se ha establecido en 50 °C.

La figura 6.10 muestra el cliente gráfico conectado a las CC, con las cargas funcionando .
Las temperaturas han sido ajusta a 50 °C.Además para llevar acabo las pruebas y corroborar la temperatura de las cargas, se ha utilizado un Radiómetro de Vapor de Agua en 183 GHz para realizar pruebas.

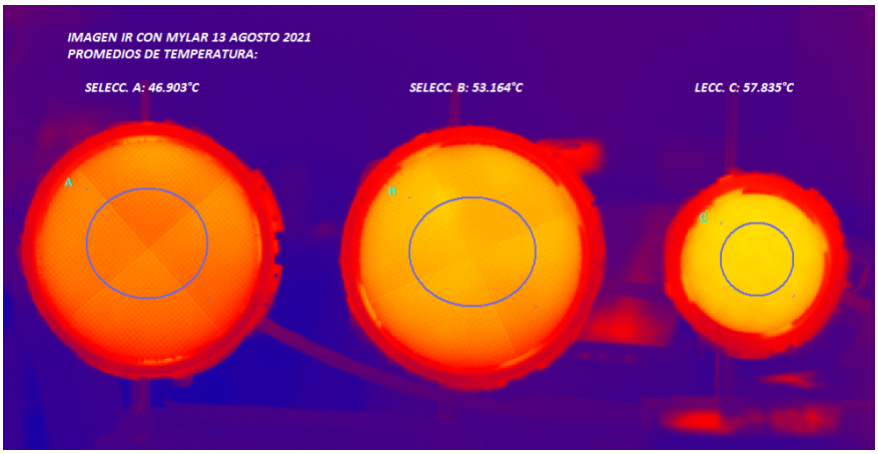


Figura 6.11: Imagen IR con temperatura constante promedio en grados Celsius, para cargas de calibración controladas a distintas temperaturas, de izquierda a derecha; Carga caliente grande a 46 °C , Carga ambiente grande a 53 °C y carga ambiente chica a 57 °C.

Los widget LCD permiten visualizar los valores de temperaturas por sensor y los gráficos presenta una vista detalla del cambio de temperatura en el tiempo.Los diferentes modos de retroalimentación pueden ser comandados por medio de los botones de `Feedback`.
Si el interruptor STOP es presionado las opciones de la interfaz quedan inhabilitadas hasta que el interruptor sea activado, nuevamente. El "back-end"de la GUI funciona con un cliente socket que ejecuta llamadas de forma asíncrona continuamente cada 0.3 segundos.
Las alarma serán activadas es caso de que se sobrepase la temperatura limite configurada (70 °C). Para efectos de esta prueba se ha configurado el componente de forma tal que pueda seguir funcionando incluso si la ventana de la interfaz es cerrada, para así poder probar otras aplicaciones de forma mas rápida sin necesidad de tener que desplegar el sistema cada vez que la interfaz se cierre, estas opciones están asociadas al cicl o de vida del componente y pueden configurarse en la CDB de las Cargas de

Calibración.

La adecuada ejecución de la prueba muestra una estabilización de la temperatura deseada, con una correcta sincronización del componente como servidor y del cliente gráfico de ACS.
Los datos de las propiedades obtenidos durante el ciclo activo del componente serán guardados en archivos XMl, a los cuales se puede acceder por medio de un lenguaje estándar SQL.Otra opción es mediante el panel de la base de datos para el monitoreo de instrumentos `tmcdb-explorer`, que permite exportar y visualizar los datos de los sensores según la fecha de registro.

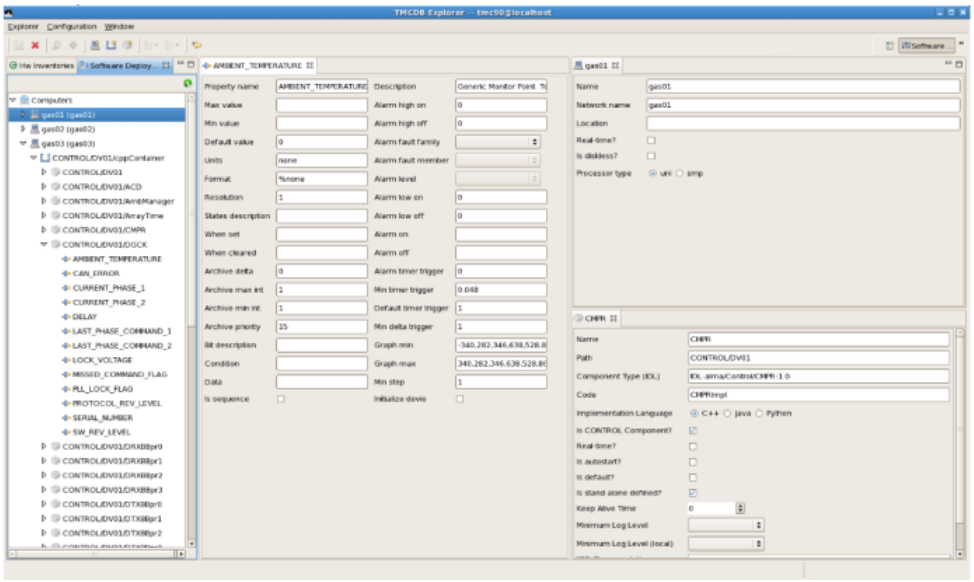


Figura 6.12: Panel de base de datos TMCDB Explorer.

Los dato exportados de temperatura para cada sensor serán utilizados para hacer los ajuste necesarios para así calibrar la potencia del receptor a distintas frecuencias.

Capítulo 7

Discusión y Conclusiones

7.1. Conclusiones Generales

Este trabajo ha producido un diseño y una implementación funcional del control para los dispositivos de calibración desarrollados por CePIA, utilizando el marco de ACS LLAMA.El proceso de desarrollo ha requirió del conocimiento y trabajo en conjunto de todas las partes involucradas en el proyecto de cargas de calibración. Además se trabajó de forma directa colaborativamente con el equipo de computación de LLAMA Y ALMA, para obtener experiencia práctica con el software y ejemplos reales, que hicieron evidentes las deficiencias y modificaciones necesarias en el diseño.

Durante el proceso de implementación se comenzó verificando el controlador para la comunicación de la tarjeta FPGA y la tarjeta la SBC, si bien en un principio LLAMA había dispuesto utilizar dispositivos SBC de tipo TS7200, con la intención de estandarizar los instrumentos de control dentro del telescopio, esto generó problemas, ya que la SBC no permitía la versión del sistema ARM Linux con los módulos necesarios para el controlador de la FPGA .Por esta razón se opto por trabajar con la tarjeta Raspberry que cumple con todos requerimientos necesarios, siendo además una plataforma de bajo costo .

Una vez se comenzó a trabajar con la tarjeta Raspberry pi se pudieron realizar las primeras pruebas sobre la plataforma de desarrollo, a la vez que se iban definiendo los métodos para el desarrollo de la capa superior de software.Esta separación de

capas, también es prueba de la característica de modularidad y la reusabilidad de la solución que se presenta en este trabajo.

La implementación del servicio de comandos SCPI con el sistema de control escrito en C se ha presentado como una excelente opción para la comunicación y control de instrumentos de forma estandarizada ya que no define el método de implementación de la capa física, y entrega uniformidad entre los diferentes buses de hardware de los instrumentos .

En lo que se refiere a la capa de software el valor del proyecto es haber desarrollado y aplicado el control y monitoreo de las cargas de calibración en tiempo real con el software de control ACS para el radiotelescopio LLAMA.Según lo analizado en el proceso de diseño, la implementación cumple con todos los requerimientos operacionales, lo cual se ha demostrado en las pruebas realizadas con los instrumentos.

El marco común de ACS ha demostrado ser general y flexible a la hora de controlar y monitorear instrumentos, entregando facilidad en los procesos de desarrollo al momento de hacer pruebas y modificaciones.El estándar CORBA permite que la comunicación se realice independientemente del lugar donde se ejecuten los procesos, pero es importante prestar atención a la configuración de los tiempo de sincronización del sistema.Si bien ACS tiene una gran dependencia en la estructura de datos para su desarrollo del software, gran partes de los software modernos de control para radioastronomía se basan en estructuras DDS, con el mismo concepto de definición de interfaces que puedan ser serializadas a través de la red pudiendo modelar las necesidades heterogéneas de los astrónomos e ingenieros independiente de la plataforma y lenguaje de programacion.Además la comunidad con el tiempo ha ido integrando otras estructuras a través de nuevas tecnologías de middlware.
La comunidad de ACS se encarga de desarrollar y actualizar el software, publicando las nuevas versiones de forma constante, generando un gran soporte para los usuarios y desarrolladores.Actualmente existen algunas aplicaciones para facilitar el desarrollo de control, mediante la generación de código automatico Nicolás Troncosoa 2013 facilitando la implementación de los componentes, aunque esta aplicación fue puesta a prueba, la compilación de los códigos requerían de varias clases de objetos

desarrolladas en lenguaje de programación JAVA, que se encontraban incompletas, dificultando el correcto uso de esta herramienta.

Se ha seguido la linea planteada por lo requerimientos operacionales y no operacionales a lo largo del proceso de diseño y de pruebas, produciendo una apropiada validación de la solución implementada. Enfatizar también que esto ha sido gracias a la correcta decisión de definir adecuadamente los requerimientos del proyecto cuando fue necesario, como un trabajo en conjunto del grupo de desarrollo de las cargas de calibración, para encontrar las soluciones adecuadas.

La arquitectura base del sistema ha sido licenciada según los términos de la LGPL y su código puesto a disposición en repositorios públicos. De esta manera el trabajo desarrollado se transforma en un aporte a la comunidad de ACS esperando también, contribuir en futuros proyectos de control de dispositivos con sistemas embebidos en general. El uso de una licencia LGPL, asegura que el código desarrollado se mantenga abierto en miras de la posibilidad de mejorar y extender este trabajo.

En relación a los objetivos planteados, se concluye lo siguiente:

- Se han implementado controladores de hardware para obtener la interfaz de las cargas de calibración, lo cual es la base de las funcionalidades de alto nivel de la aplicación. En específico, se han realizado pruebas de funcionamiento que implican el uso de periféricos de comunicación USB/UART y ethernet, propios de los sistemas embebidos.

- Se implementa y se prueba la arquitectura base del software de ACS LLAMA, instalando bibliotecas necesarias para la correcta ejecución del software y el desarrollo del componente de control.

- Se desarrolló el componente de control de las CC en base a la interfaz que define las propiedades de los instrumentos, además se creó el esquema de la CDB y las instancias con esquemas XML del contenedor `cepiacontainer` pudiendo inicializar el componente registrando todo su ciclo de vida y entregando las funcionalidades distribuidas del software.

- Se genero una interfaz gráfica de control y monitoreo en tiempo real para las CC utilizando C++ y QT.

- Con el sistema integrado se realizan pruebas prolongadas de funcionamiento para verificar los requerimientos operacionales, obteniendo resultados satisfactorios.

7.2. Limitaciones

Si bien los objetivos generales y específicos planteados se han cumplido, la solución puede presentar algunas limitaciones, que se describen a continuaion:

- Mientras que la aplicación de software para la comunicación con la tarjeta FPGA puede ser fácilmente re utilizada en una SBC distinta que cuente con periférico ethernet y USB, en el sistemas ARM linux, se requiere de una versión de Kernel igual o superior a la 2.6.9.

- La versión del sistema operativo seleccionado por LLAMA(Centos 6.6.10) presenta algunos problemas de compatibilidad con las bibliotecas del repositorio del marco ACS.La versión de ACS seleccionada funciona con un python 2.7, generando problemas con el sistema Centos que requiere de bibliotecas de python 2.6 para aplicaciones internas del sistema relacionadas con la interfaz gráfica del sistema operativo.Esto trajo varios problemas y no fue posible utilizar el lenguaje de python para el desarrollo de la GUI.En gran medida esto fue lo que género que la GUI de las CC se implementara en el lenguaje de programación C++.

- El desarrollo de GUIs en ACS esta bien implementado en el lenguaje Java utilizando ABeans con NetBeans [6], permitiendo que CORBA pueda trabajar con la GUI entre distintos contenedores de forma distribuida, sin embargo la GUI desarrollada para las cargas de calibración disenada con QT solo se utilizo con un contenedor local donde se ejecuto el componente desarrollado por lo que es necesario probar las características distribuidas de la GUI implementada.

7.3. Trabajos Futuros

El trabajo desarrollado es una de las primeras aproximaciones para el control de las CC en el radiotelescopio LLAMA, por lo tanto se desprenden una serie de posibles trabajos futuros:

- Agregar respuestas más específicas a los comandos SCPI, posibles errores de sintaxis o valores que se puedan presentar en relación a las funcionalidades de los instrumentos, ayudando a la depuración, en caso de presentarse algún error.
- La interfaz gráfica requiere de una optimización en el tiempo de actualización de las propiedades monitorizadas. Este problema de la actualización de los valores puede estar relacionado a la funcionalidad de ACS para publicar eventos o directamente al hilo QT implementado para esta tarea.
- Posible mejoras a la GUI, como agregar un cursor que muestre los valor de temperatura para la curvas en los gráficos, maximizar visualización de gráficos, etc.
- Agregar una propiedad de estatus para cada carga, para que pueda ser monitorizada no solo la temperatura pero además el estado de los instrumentos por el sistema distribuido de ACS.

Apéndice A

Puntos de Monitoreo y Control

A.0.1. Puntos de Monitoreo

Cuadro A.1: Comando para consultar identificación de instrumento.

Comando	*IDN?
Descripción	Cadena de caracteres con la descripcion serial para el sistema de CC.
Intervalo de Tiempo	Este valor no varia.
Dato	Cadena ASCII con 4 valores, separados por coma: 1. nombre del fabricante 2. Número de modelo: Cargas de calibración 3. Número de serie: NULL. 4. revisión de firmware.

A.0.2. Puntos de Control

Cuadro A.2: Comando obtiene Temperatura sensor 1, Carga chica caliente.

Comando	CALibration:LOAd1:SENsor1?
Descripción	Comando encargado de obtener la temperatura sensor 1, carga CCC.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 1 CCC.

Cuadro A.3: Comando obtiene Temperatura sensor 2, Carga chica caliente.

Comando	CALibration:LOAd1:SENsor2?
Descripción	Comando encargado de obtener la temperatura sensor 2, carga CCC.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 2 CCC.

Cuadro A.4: Comando obtiene Temperatura sensor 3, Carga chica caliente.

Comando	CALibration:LOAd1:SENsor3?
Descripción	Comando encargado de obtener la temperatura sensor 3, carga CCC.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 3 CCC.

Cuadro A.5: Comando obtiene Temperatura sensor 1, Carga Grande caliente.

Comando	CALibration:LOAd1:SENsor1?
Descripción	Comando encargado de obtener la temperatura sensor 1, carga CCG.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 1 CCG.

Cuadro A.6: Comando obtiene Temperatura sensor 2, Carga Grande caliente.

Comando	CALibration:LOAd2:SENsor2?
Descripción	Comando encargado de obtener la temperatural sensor 2, carga CCG.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 2 CCG.

Cuadro A.7: Comando obtiene Temperatura sensor 3, Carga Grande caliente.

Comando	CALibration:LOAd2:SENsor3?
Descripción	Comando encargado de obtener la temperatura sensor 3, carga CCG.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 3 CCG.

Cuadro A.8: Comando obtiene Temperatura sensor 1, Carga Grande Fria.

Comando	CALibration:LOAd3:SENsor1?
Descripción	Comando encargado de obtener la temperatura sensor 1, carga CGF.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 1 CGF.

Cuadro A.9: Comando obtiene Temperatura sensor 2, Carga Grande Fria.

Comando	CALibration:LOA3:SENsor2?
Descripción	Comando encargado de obtener la temperatura sensor 2, carga CGf.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 2 CGf.

Cuadro A.10: Comando obtiene Temperatura sensor 3, Carga Grande Fria.

Comando	CALibration:LOA3:SENsor3?
Descripción	Comando encargado de obtener la temperatura sensor3, carga CGf.
Intervalo de Tiempo	0.4 s
Dato	1 double (8 bytes): Salida sensor 3 CGf.

Cuadro A.11: Comando para detener el sistema completo.

Comando	CALibration:STOP<SP><Parameter>
Descripción	Detiene todo el sistema. Este comando debe ser desactivado para reiniciar el sistema nuevamente.
Intervalo de Tiempo	0.4 s
Dato	Cadena ASCII : Parametro: <ACTIVE> : Mantiene activo el modo STOP activo . Parametro: <DEActivate>: Desbloquea el modo STOP.

Cuadro A.12: Comando para controlar temperatura calefactor carga chica caliente.

Comando	CALibration:LOAd1:HEAter<SP><Data>
Descripción	Este comando permite cambiar la temperatura del calefactor de la carga CCC.
Intervalo de Tiempo	Según sea necesario.
Dato	Cadena numérica ASCII: Temperatura de control de la CCC en grados Celsius. Rango: 0 - 70 °C

Cuadro A.13: Comando para controlar temperatura calefactor carga Grande caliente.

Comando	CALibration:LOAd2:HEAter<SP><Data>
Descripción	Este comando permite cambiar la temperatura del calefactor de la carga CGC.
Intervalo de Tiempo	Según sea necesario.
Dato	Cadena numérica ASCII: Temperatura de control de la CGC en grados Celsius. Rango: 0 - 70 °C

Cuadro A.14: Comando para controlar temperatura calefactor carga grande fria.

Comando	CALibration:LOAd3:HEAter<SP><Data>
Descripción	Este comando permite cambiar la temperatura del calefactor de la carga CGF.
Intervalo de Tiempo	Según sea necesario.
Dato	Cadena numérica ASCII: Temperatura de control de la CGF en grados Celsius. Rango: 0 - 70 °C

Cuadro A.15: Comando para encender o apagar carga chica caliente.

Comando	CALibration:CONtrol:LOAd1:<SP><Parameter>
Descripción	Cambia el estado del la CCC . Posibles valores encendido o apagado.
Intervalo de Tiempo	0.4 s.
Dato	Cadena numérica ASCII: <ON> : Enciende la retroalimentación para la carga. <OFF> : Apaga la retroalimentación para la carga.

Cuadro A.16: Comando para encender o apagar carga grande caliente.

Comando	CALibration:CONtrol:LOAd2:<SP><Parameter>
Descripción	Cambia el estado del la CGC . Posibles valores encendido o apagado.
Intervalo de Tiempo	0.4 s.
Dato	Cadena numérica ASCII: <ON> : Enciende la retroalimentación para la carga. <OFF> : Apaga la retroalimentación para la carga.

Cuadro A.17: Comando para encender o apagar carga grande fría.

Comando	CALibration:CONtrol:LOAd3:<SP><Parameter>
Descripción	Cambia el estado del la CGf . Posibles valores encendido o apagado.
Intervalo de Tiempo	0.4 s.
Dato	Cadena numérica ASCII: <ON> : Enciende la retroalimentación para la carga. <OFF> : Apaga la retroalimentaciónn para la carga.

Cuadro A.18: Modo de retroalimentación para carga chica caliente.

Comando	CALibration:FEEDback:LOAd1:<SP><pARAMETER>
Descripción	Cambia el modo de retroalimentación para la carga CCC. Los diferentes parametros especifican el sensor al que se desea retroalimentar.
Intervalo de Tiempo	0.4 s.
Dato	Cadena ASCII : Modo de retroalimentación para carga CCC <SEN1> : Configura el modo de retroalimentación para el sensor 1. <SEN2> : Configura el modo de retroalimentación para el sensor 2. <SEN3> : Configura el modo de retroalimentación para el sensor 3. <AVErage> : Configura el modo de retroalimentación para el promedio de los sensores.

Cuadro A.19: Modo de retroalimentación para carga grande caliente.

Comando	CALibration:FEEDback:LOAd2:<SP><pARAMETER>
Descripción	Cambia el modo de retroalimentación para la carga CGC. Los diferentes parametros especifican el sensor al que se desea retroalimentar.
Intervalo de Tiempo	0.4 s.
Dato	Cadena ASCII : Modo de retroalimentación para carga CGC. <SEN1> : Configura el modo de retroalimentación para el sensor 1. <SEN2> : Configura el modo de retroalimentación para el sensor 2. <SEN3> : Configura el modo de retroalimentación para el sensor 3. <AVErage> : Configura el modo de retroalimentación para el promedio de los sensores.

Cuadro A.20: Modo de retroalimentación para carga grande fría.

Comando	CALibration:FEEDback:LOAd3:<SP><pARAMETER>
Descripción	Cambia el modo de retroalimentación para la carga CGF. Los diferentes parametros especifican el sensor al que se desea retroalimentar.
Intervalo de Tiempo	0.4 s.
Dato	Cadena ASCII : Modo de retroalimentación para carga CGF <SEN1> : Configura el modo de retroalimentación para el sensor 1. <SEN2> : Configura el modo de retroalimentación para el sensor 2. <SEN3> : Configura el modo de retroalimentación para el sensor 3. <AVErage> : Configura el modo de retroalimentación para el promedio de los sensores.

Apéndice B

Requerimientos de Instalación ACS

B.0.1. Paquetes Centos 6.6.10

```
# Install EPEL
yum install epel-release -y
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
# IUS
curl -s https://setup.ius.io | bash
# GIT LFS
curl -s https://packagecloud.io/install/repositories/
github/git-lfs/script.rpm.sh | bash

# Install rpmfusion
yum -y localinstall --nogpgcheck
https://download1.rpmfusion.org/free/el/
rpmfusion-free-release-6.noarch.rpm

https://download1.rpmfusion.org/nonfree/el/
rpmfusion-nonfree-release-6.noarch.rpm
```

```
# Additional packages
yum -y install binutils-devel \
man            \
time           \
ksh            \
emacs          \
procmail       \
vim-enhanced   \
xterm          \
blas-devel     \
lapack-devel   \
cfitsio-devel  \
pgplot         \
fftw3-devel    \
pgplot-devel   \
readline-devel \
libxslt        \
gcc-gfortran   \
gcc-c++        \
compat-libf2c-34 \
compat-gcc-34-g77 \
subversion     \
flex           \
byacc          \
bison          \
libxml2-devel  \
libxslt-devel  \
autoconf       \
automake       \
```

```
libtool        \
texinfo        \
expat-devel    \
db4-devel      \
boost-devel    \
bzip2-devel    \
openssl-devel  \
sqlite-devel   \
openldap-devel \
libpng-devel   \
freetype-devel \
libmemcached-devel \
java-1.8.0-openjdk \
java-1.8.0-openjdk-devel \
ncurses-devel \
xorg-x11-server-Xvfb \
xauth \
mysql-server  \
cmake \
rpm-build \
git2u \
git2u-gui \
git2u-gitk \
git-lfs \
qt-creator \
screen \
zsh \
ack \
meld
```

```
# install wcslib
wget http://www.aoc.nrao.edu/~atejeda/RPM/rhel6/x86_64/
wcslib-4.7-02.el6.x86_64.rpm
rpm -i wcslib-4.7-02.el6.x86_64.rpm

wget http://www.aoc.nrao.edu/~atejeda/RPM/rhel6/x86_64/
cfitsio-3.350-01.el6.x86_64.rpm
wget http://www.aoc.nrao.edu/~atejeda/RPM/rhel6/x86_64/
cfitsio-devel-3.350-01.el6.x86_64.rpm
rpm -Uvh cfitsio-3.350-01.el6.x86_64.rpm
cfitsio-devel-3.350-01.el6.x86_64.rpm
wget http://www.aoc.nrao.edu/~atejeda/RPM/rhel6/x86_64/
wcslib-devel-4.7-02.el6.x86_64.rpm
rpm -i wcslib-devel-4.7-02.el6.x86_64.rpm

echo '*      soft   nofile      10240' >> /etc/security/limits.conf
echo '*      hard   nofile      10240' >> /etc/security/limits.conf
# Create java link
mkdir /usr/java
ln -s /usr/lib/jvm/java /usr/java/default

# Create /alma
mkdir /alma
chown almamgr:users /alma

# Instalation of git-lfs for almamgr
su almamgr -c "
cd /home/almamgr
git lfs install
```

```
# Install a SSH key for root access without a password
su almamgr -c "
rm -f /home/almamgr/.ssh/id_rsa*
ssh-keygen -f /home/almamgr/.ssh/id_rsa -N ''
echo -n 'localhost ' >> /home/almamgr/.ssh/known_hosts
cat /etc/ssh/ssh_host_rsa_key.pub >> /home/almamgr/.ssh/known_hosts
"
mkdir -p /root/.ssh
cat /home/almamgr/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys

# this speeds up ssh login when you're offline
sed -i -e "s/#UseDNS yes/UseDNS no/" /etc/ssh/sshd_config

# install the llama_conf alias
su almamgr -c "
echo \"alias llama_conf=\\\"source /alma/ACS-2017DEC/ACSSW/config
/.acs/.bash_profile.acs\\\"\" >> /home/almamgr/.bashrc
echo \"export INTROOT=\\\"/home/almamgr/INTROOT\\\"\" >>
/home/almamgr/.bashrc
"

# set hostname
sed -i -e 's/HOSTNAME=.*/HOSTNAME=llama6.localdomain/'
/etc/sysconfig/network
#  required or acs will have issues with logging
sed -i -e 's/\(127.0.0.1.*\)/\1 llama6 llama6.localdomain/'
/etc/hosts
```

Apéndice C

Configuraciones Sistema SBC

C.0.1. Distribución y Configuración del Sistema Linux

El sistema GNU/Linux embebido utilizado en el proyecto corresponde a un sistema Raspberry pi OS, que se basa en el Kernel Linux 4.20. El cual se ejecuta mediante un dispositivo Micro SD, Este sistema proporciona unentorno de desarrollo integrado GNU C/C++, con los servicios básicos para llevar acabo el desarrollo de la aplicación de software .
Las bibliotecas del sistema, se relacionan con las capas de alto nivel que provee abstracciones comunes a todos los sistemas Linux, incluyendo procesos, archivos, sockets y señales. Algunas de las herramientas bibliotecas utilizadas, se describen en la siguiente tabla:

Cuadro C.1: Caracteristicas principales del sistema Raspberry pi OS.

Lenguaje	C,C++
OS	linux-4.20.8
Binutils	binutils-2.32
Copilador	CMake 3.16
Librerias C	glibc-2.29
Herramientas depuracion	gdb-8.2.1
Herramientas estra	libiconv-1.15 mpc-1.1.0 mpfr-4.0.2 ncurses-6.1 zlib-1.2.11.

Modulo de Kernel

Dentro del kernel, la interfaz de bajo nivel es específica para cada configuración de hardware, sobre la cual, el kernel se ejecuta y provee control directo de los recursos de hardware. Los servicios de bajo nivel manejan operaciones específicas de la CPU, operaciones de memoria específicas de la arquitectura, y provee interfaces básicas para los dispositivos.

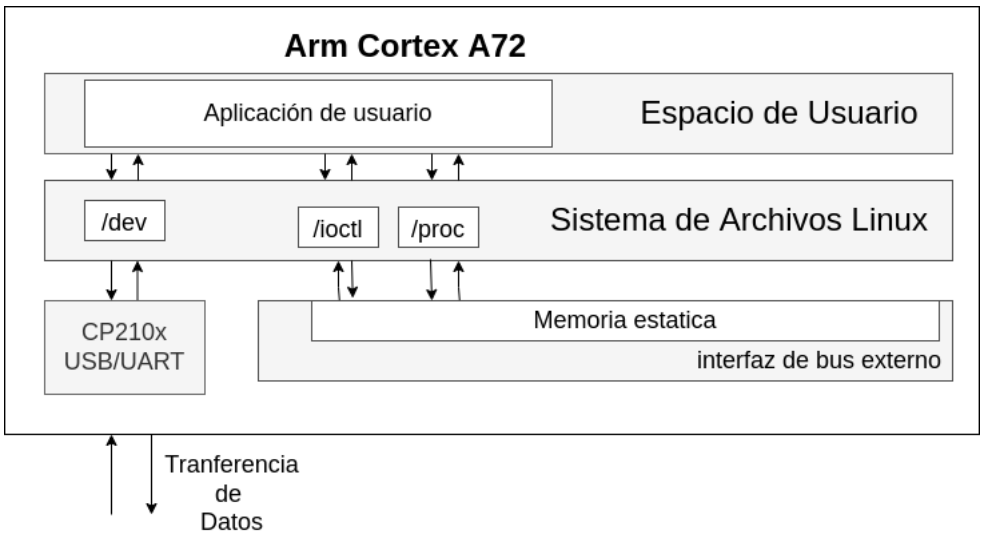


Figura C.1: Se muestra la interacción de la capa física de la comunicación con la capa lógica de usuario del sistema ARM Linux Raspberry.

El control del parifico USB/UART requiere de un controlador propio del dispositivo FPGA .Estos módulos tienen una extención .ko y en general son requeridos para establecer comunicación con otros dispositivos, el modulo utilizado con la tarjeta Artix 7 corresponde a un modulo CP210x.ko.

Para la integración del modulo (controlador) en el sistema operativo, es posible instalarlo con los siguientes e comando, como usuario root en la terminal del sistema Linux:

```
modprobe foo
modprobe -v CP210x.ko
insmod /lib/modules/3.5.0-30-generic/kernel/drivers/char/CP210x.ko
```

Este controlador de hardware se limita a la lectura y escritura de registros en la dirección de memoria adecuada y es fundamental para las aplicaciones de más alto nivel.

C.0.2. Descripción Comandos SCPI

Los Comandos estándar para instrumentos programables (SCPI) son cadenas ASCII, que se envían desde un programa en ejecución desde una computadora a un instrumento a través de alguna capa de comunicación física. El primer estándar SCPI fue lanzado en 1990 por el consorcio SCPI como una capa adicional para el estándar IEEE-488.2, que define las propiedades del protocolo de GPIB [4].
EL estándar especifica la estructura y la sintaxis de los comandos, sin definir el hardware ni el software subyacentes.
Los comandos SCPI consisten en comandos de configuración y comandos de consulta.De esta forma los comandos pueden cambiar la configuración del instrumento o realizar una acción específica.
Mientras que las consultas hacen que el instrumento devuelva datos o información sobre su estado. La estructura de consulta es la misma que la estructura de control, excepto que termina con un signo de interrogación. Un mensaje de comando corresponde al nombre de comando o consulta, seguido de cualquier información o parametro que el instrumento necesite ejecutar.
La estructura de mensaje, consta de cinco tipos de elementos, definidos en la Tabla C.2 y presentados en la Figura C.2.

Cuadro C.2: Elementos de los mensajes de comando.

Simbolos	**Significado**
Encabezado	Nombre del comando. El comando es una consulta si el encabezado termina con un signo de interrogación. Puede comenzar con un carácter de dos puntos (:).
Mnemotecnico	Una subfunción del encabezado. La mayoría de los encabezados consisten en muchos mnemotécnicos separados por dos puntos (:)
Argumento	Una cantidad, calidad, restricción o límite asociado con el encabezado. Algunos comandos no tienen argumentos, mientras que otros tienen múltiples argumentos. Los argumentos están separados del encabezado por un ¡Espacio¿. Varios argumentos están separados entre sí por ¡Coma¿
Espacio	Un carácter de espacio en blanco entre el encabezado del comando y el argumento.

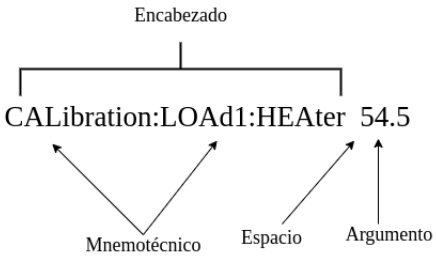


Figura C.2: Elementos de los mensajes de comando.

Algunos requisitos importantes del estandar SCPI con respecto a los nombres mnemotécnicos:

- Cada mnemotécnico tiene una forma larga y una forma corta. Un instrumento SCPI aceptará solo las formas corta exacta y larga exacta.
- El instrumento aceptará tanto mayúsculas como minúsculas sin distinción entre mayúsculas y minúsculas. Los comandos SCPI se basan en una estructura jerárquica.

La sintaxis que engloba a los comandos de las CC, se define directamente por medio de una estructura jerarquica también conocida como árbol. La siguiente figura

muestra el árbol jerárquico para los comandos de las Cargas de Calibración:

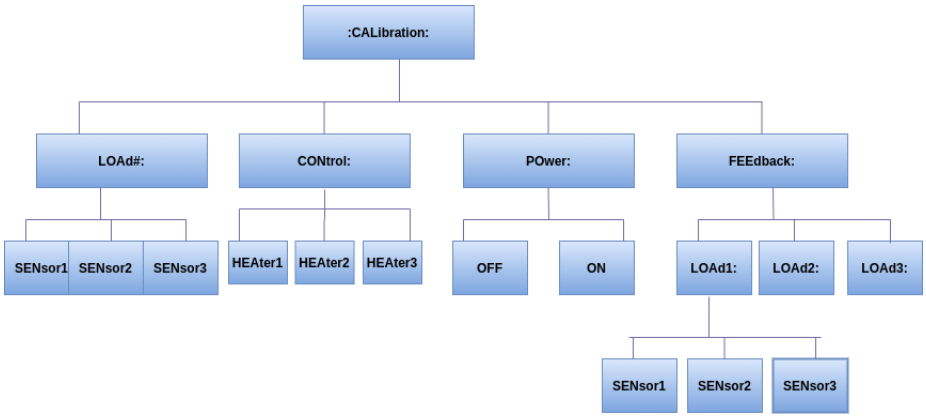


Figura C.3: Árbol jerárquico, para comandos SCPI asociados a las CC.

Los elementos del primer nivel en la estructura jerarquía y raíces de esta, son producto del modelo mismo de los instrumentos, y representan la palabra clave de cada agrupación de ordenes para las llamadas al subsistema.
De esta forma las órdenes asociadas con un subsistema en particular son agrupadas bajo un nodo común de la jerarquía, análogamente a ramas conectadas a otros nodos comunes. Cada nuevo nodo que origine una rama será caracterizado con un mnemotécnico, este nodo dará origen a nuevas ramas y estas a otras sucesivamente, de acuerdo a la funcionalidad que se desee cubrir con la orden en desarrollo. Así, la lectura de una orden se hará siguiendo los nodos de cada rama generada a partir de la raíz del árbol o estructura jerárquica.

Apéndice D

Construccion Software ACS

D.0.1. Configuración del Entorno Virtual

La virtualización es el proceso de crear una versión "virtual", basada en el software de una computadora, con una cantidad especifica de CPU, memoria y almacenamiento que se "toman prestadas"de una computadora anfitriona física, como una computadora personal, o un servidor remoto. Una máquina virtual se puede representar como un archivo, generalmente llamado imagen, que se comporta como una computadora real.
LLAMA ha proporcionado una imagen, con la configuración del sistema de hardware y software correspondiente a la computadora ABM, para el desarrollo del control de dispositivos, de forma que los distintos desarrolladores trabajen sobre una plataforma común.
Esta maquina corresponde a un sistema operativo (OS por sus siglas en ingles) GNU/Linux Centos 6.6 7 de 64 bits qu pude ser descargado desde la pagina oficial [12]. Algunos requerimientos de hardware de la maquina, pueden verse en la siguiente tabla :

Cuadro D.1: Requerimientos de hardware para el desarrollo.

Procesador	1 CP
Memoria Ram	4GB
Espacio en disco	8GB

La maquina anfitriona que se encarga de ejecutar la maquina virtual debe proporcionar a lo menos 4Gb RAM. Esto ya que el sistema es muy inestable con menos

de 4096 MB de RAM, pudiendo generar que el software presente varios problemas a la hora de ser ejecutado .
Además son importante algunas pequeñas configuraciones para que el software funcione sin problemas :

- El servidor dns debe apuntar a su DNS local para resolver IPs externas.
- Se han desactivado las herramientas de informes remotos para máquinas virtuales.
- Se deben aumentar algunos parámetros de inicio, como los tiempos de espera de ACS, ya que la operación en la maquina virtual es más lenta que en el anfitrión real. Este es el caso para la carga del repositorio IDL.

Oracle VirtualBox proporciona herramientas para generar un discos de maquina virtual, instalando el sistema operativo base para correr y poner en marcha el sistema .
La configuración en la maquina virtual, puede ser modificada fácilmente cambiando los parametros necesarios.

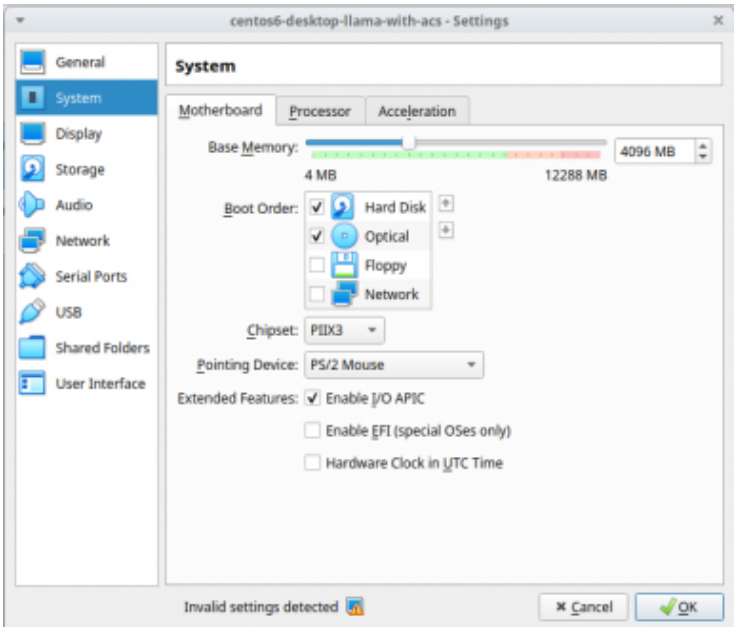


Figura D.1: Configuración de Imagen LLAMA en VirtualBox.

D.1. Construcción del Software ACS

ACS cuenta con sus propias librerías y archivos fuentes, que vienen incluidas en los paquetes de instalación del software, estos pueden ser descargados directamente desde el sitio oficial de ALMA SW [7], donde anualmente se van corrigiendo errores y se lanzan nuevas versiones del software. El proceso de instalación puede variar dependiendo de la versión escogida, ya que la versión anual principal a menudo contienen algunos cambios incompatibles con versiones anteriores. Incluso las versiones que no requieren cambios en el código generalmente conllevan cierto esfuerzo de portabilidad, esto ya que algunos resultados de registro que se evalúan en las pruebas modulares pueden cambiar y requieren una actualización de los archivos de referencia.

La versión del repositorio utiliza por LLAMA corresponde a la versión ACS-2017.0.0. El actual trabajo de tesis utiliza el repositorio git oficial de LLAMA para obtener una copia de la maquina virtual.

A continuación se describe La instalación del software y el proceso de configuración del sistema de LLAMA.El sistema de ACS utiliza bibliotecas básicas del sistema operativo para su ejecución aparte de las de uso general, es por esto que la instalación y configuración dependerán del sistema operativo utilizado.
En primer lugar se configura el sistema, generando los permisos de usuario necesarios en el directorio donde se instala ACS .La cuenta de usuario corresponde a `almamgr` y el grupo a alma .

```
sudo chown almamgr:users /alma
groupadd -g 335 alma
useradd g 335 u 3060 d /home/almamgr \ -m s /bin/bash almamgr
```

La versión del software se obtienen desde el repositorio oficial de LLAMA, y se instala y configura en el directorio antes mencionado, como un controlador de version git.

```
sudo mkdir /alma
sudo rm -rf /alma
```

```
git clone ssh://paulinaunanue@llamaobservatory.org/work/llama/
llama.git~/llama-src
```

Como las tareas de control generalmente necesitan soporte en tiempo real ACS requiere de la instalación de un núcleo linux 2.6.29.4-t3.7.1, que cuente con estas características:

```
source ~/llama-src/ACS/LGPL/acsBUILD/config/.acs/.bash_profile.acs
cd ~/llama-src/ACS/ExtProd/RTOS./buildRTOS -l
config/kernel-2.6.29.4-rtos-3.7.1-XVB601 -r config/
realtime-3.7.1-XVB601 -k 2.6.29.4-t3.7.1
exit
```

Para la instalación de las herramientas de propósito general (necesarias para diversas aplicaciones como compilaciones, pruebas de código, doxygen para manual de páginas, documentación, etc.) se utiliza el directorio llama-src, donde se desempaquetan todos los productos provistos como archivos binarios.

```
unset INTROOT
source ~/llama-src/ACS/LGPL/acsBUILD/config/.acs/.bash_profile.acs
cd ~/llama-src/ACS/ExtProd/INSTALL
./buildTools
exit
```

Una vez concretada la instalación, se debe verificar que los paquetes externos queden instalados correctamente :

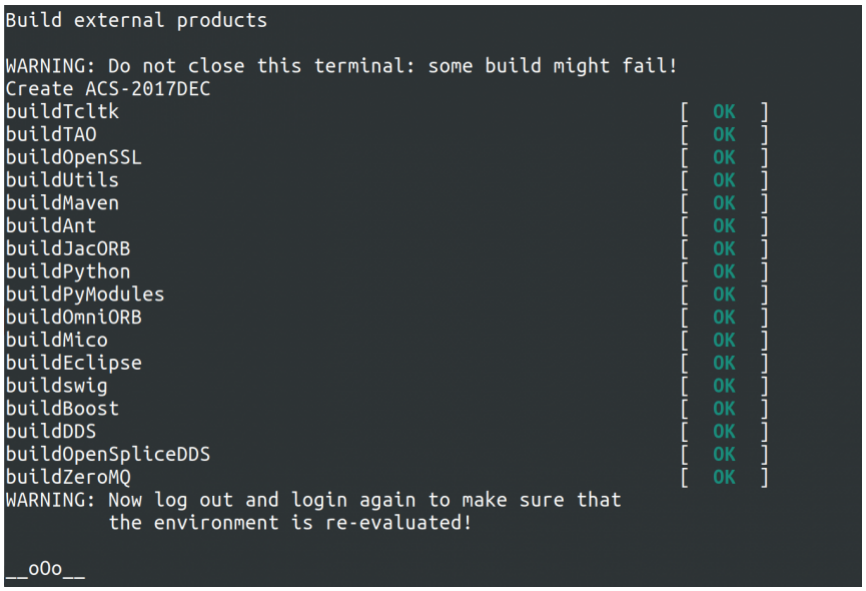


Figura D.2: Instalación paquetes externos de ACS.

D.1.1. Construcción Software especifico de LLAMA

ACS es un software con una arquitectura modular, es decir, divide su sistema en distintos subprogramas, que mantienen interfaces bien definidas capaces de interactuar entre si, con el fin de hacer el software más legible y manejable. Para la construcción de ACS LLAMA se instalan los módulos contenidos en el paquete por defecto de ACS, exceptuando el modulo correspondiente al correlador .

```
export MAKE_NOSTATIC=yes; export MAKE_NOIFR_CHECK=on
unset INTROOT
source /alma/ACS-2017.0/ACSSW/config/.acs/.bash_profile.acs
cd ~/llama-src/ARCHIVE
make build
exit
export MAKE_NOSTATIC=yes; export MAKE_NOIFR_CHECK=on
export INTROOT=$HOME/INTROOT
source /alma/ACS-2017.0/ACSSW/config/.acs/.bash_profile.acs
cd ~/llama-src/ICD
make build

export MAKE_NOSTATIC=yes; export MAKE_NOIFR_CHECK=on
```

```
export INTROOT=$HOME/INTROOT
source /alma/ACS-2017DEC/ACSSW/  config/.acs/.bash_profile.acs
cd ~/llama-src/CONTROL
make build
```

Preparación del INTROOT. Para la compilación y las pruebas, es necesario crear un directorio llamado INTROOT (raíz de integración"). Este directorio se completara con los resultados del proceso de compilación e instalación, es importante recalcar, que este directorio es de solo lectura y no se debe escribir directamente en él.

```
unset INTROOT
source ~/llama-src/ACS/LGPL/acsBUILD/config/.acs/.bash_profile.acs
export MAKE_NOSTATIC=yes; export MAKE_NOIFR_CHECK=on
cd ~/llama-src/ACS
make build
exit
```

Para continuar, se genera la configuración de bash que establece las variables de entorno que ACS usa para apuntar al directorio INTROOT recién creado .

```
unset INTROOT
source ~/llama-src/ACS/LGPL/acsBUILD/config/.acs/.bash_profile.acs
export MAKE_NOSTATIC=yes; export MAKE_NOIFR_CHECK=on
cd ~/llama-src/ACS
make build
exit
```

INTROOT permite instalar código sin interferir con el sistema de instalación de ACS . permitiendo desarrollar pruebas experimentales y también como sistema de parcheo, especialmente cuando se usa en combinación con INTLIST, que es una lista de INTROOTs, con un orden de periodización para elegir archivos binarios, bibliotecas, jars, código python, etc.

D.2. Generación del Componente

El proceso de generación del componente, sigue un patrón de diseño estándar como el resto de componentes de ACS generados en el sistema de control del radiotelescopio LLAMA, el diagrama de clases típico de un componente característico en ACS se representa en el apendice C.0.1.
Para esto es necesario especificar la ruta del modulo a crear. Para el caso del componente de las CC denominado `Calibrationloads`, se desarrollara dentro del modulo de CONTROL, donde se encuentran todos los dispositivos de hardware por defecto del sistema .

```
export HW_DEV_DIR=/home/almamgr/llama-src/CONTROL/Device/
HardwareDevice/CalibrationLoads
```

El software de ACS provee una función `getTemplate` para la generación de módulos, esta crea un script con distintos parametros para la estructura del directorio o para un código.Las opciones se ejecutan en este orden:

```
getTemplate
directoryStructure
createWS_MODROOTarea
directoryStructure
```

La opción MODROOT depende del tipo de modulo que se desee crear, se cuenta con tres opciones posibles que representan escenarios muy similares. Las opciones para crear un directorio de módulo, pueden ser ya sea para el desarrollo de control (MODROOTarea), para Unidad de control local (LCU) (MODROOT_LCU) o para ambos (MODROOT_WS_LCU).

Bibliografía

[1] Arquitectura de acs. URL https://confluence.alma.cl/display/ICTACS/ACSArchitecture.

[2] Comandos estandar para instrumentos programables (scpi), volumen 1: Estilo y sintaxis. . URL https://www.ivifoundation.org/docs/scpi-99.pdf.

[3] Estudio de diseño qt. URL https://www.qt.io/design.

[4] Interfaz digital estándar para instrumentación programable. URL https://webstore.iec.ch/publication/2246.

[5] Interfaz digital estándar para instrumentación programable - parte 2: Códigos, formatos, protocolos y comandos comunes. . URL https://ieeexplore.ieee.org/document/1352831.

[6] Java bean software. URL https://www.javatpoint.com/java-bean.

[7] Lanzamientos de versiones de alma common software . URL https://confluence.alma.cl/display/ICTACS/ACS+Releases.

[8] Links to acs release is available at. URL https://confluence.alma.cl/display/ICTACS/ACS+Releases.

[9] National radio astronomy observatory (charlottesville) website. URL http://www.cv.nrao.edu/course/astr534/Radiometers.html/.

[10] Observatorio astronómico nacional de japón . URL https://www.nao.ac.jp/en/.

[11] Servicio de informacion sobre ciencia tecnologia y politica cientifica argentina. URL https://nexciencia.exactas.uba.ar.

[12] Versión de centos 6.10. URL https://wiki.centos.org/Manuals/ReleaseNotes/CentOS6.10.

[13] Vivado design suite software. URL https://www.xilinx.com/products/design-tools/vivado.html.

[14] . En *Workshop ESO - Garching 08*. 2004.

[15] P. A. Selivanov et al. A. Senchenko, G. Fatkin. Tango based software of control system of lia-20. *Proceedings of the 16th International Conference on Accelerator and Large Experimental Physics Control Systems*, 2018.

[16] K.Zagar. A.Capronib, K.Sigerudc. Integrating the cern laser alarm system with the alma common software. *Proc. SPIE 6274, Advanced Software and Control for Astronomy.*, 627407, 2006.

[17] Daizhong Liu . et al. Adam K. Leroy, Annie Hughes. Phangs–alma data processing and pipeline. *The Astrophysical Journal .*, 255:54, 2016.

[18] Patrick Wallace. Alan Bridger, Dennis Kelly. D. L. Terrett. Choosing a control system for ccat. *SPIE Proceedings.*, 7740, 2010.

[19] Jeff Ker Allen Farris, Ralph Marson. *ALMA Project Control Subsystem Design Document*, 2009.

[20] Keiichi Asada Rebecca Azulay et al. Antxon Alberdi, Walter Alef. The event horizon telescope collaboration, kazunori akiyama1. first m87 event horizon telescope results. vi. the shadow and mass of the central black hole. *The American Astronomical Society.*, 2019.

[21] L. Basoalto. Desarrollo de fuentes de calibración para instrumentos de observación submilimétrica. 2018.

[22] Jan Breuer. Biblioteca del analizador scpi. URL https://www.jaybee.cz/scpi-parser/.

[23] J. J. Condon y S. M. Ransom. *Essential Radio Astronomy*. 2016.

[24] W. D.; Greisen E. W. et al. Condon, J. J.; Cotton. The nrao vla sky survey. *The Astronomical Journal*, 115, 1998.

[25] David R.; Boyer. Corinne; Trinh Thang . Silva. Object-oriented experiences with gbt monitor and control. *Proceedings of SPIE.*, 7019, 2004.

[26] Parsons A. R. Aguirre J. E. et al. DeBoer, D. R. Hydrogen epoch of reionization array (hera). ArXiv e-prints: 1606.07473, 2016.

[27] Braun R. Turner W. Dewdney, P. E. The mid-frequency telescope for the square kilometre array (ska-mid). *XXXIInd General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS).*, 2017.

[28] The European Southern Observatory (ESO). Pagina principal de el observatorio europeo austral. URL www.eso.org.

[29] G.Chiozzi et al. Corba-based common software for the alma project. *Advanced Telescope and Instrumentation Control Software II*, 4848:43–54, 2002.

[30] J. R. Fisher. Object-oriented experiences with gbt monitor and control. *Astronomical Data Analysis Software and Systems VII.*, 145, 1998.

[31] B. Goodrich et al. G. Chiozzi, K. Gillies. Trends in software for large astronomy projects. *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS, Knoxville, TN, USA.*, 2001.

[32] S. Weinreb et al. G. Hallinan1, V. Ravi1. The dsa-2000 - a radio survey camera. *American Astronomical Society meeting 237*, 53, 2021.

[33] Gianluca Chiozz. Gianni Raffi. The alma common software as a basis for a distributed software development. *Astronomical Data Analysis Software and Systems XI.*, 281, 2002.

[34] Object Management Group. tomo CORBA 3.0. 2002.

[35] M. M. B. S. A. J. Hazard, C. 1963.

[36] Bell S. J. Pilkington J. D. H. Scott P. F. Collins R. A. 1. Hewish, A. *Observation of a Rapidly Pulsating Radio Source.*, tomo 217. 1968.

[37] B.Jerama. H.Sommera A.Caproniae et al. G.Chiozzi. The alma common software: a developer friendly corba-based framework. *Proceedings of SPIE.*, 2004.

[38] Antognini Jonathan Avarias Jorgeet et al. Ibsen, Jorge. Building a world-wide open source community around a software framework. progress, dos, and don'ts. *Proceedings of the SPIE*, 9913:7, 2016.

[39] D. Quartel et al. J. P. Almeida, M. V. Sinderen. Interaction systems design and the protocol- and middleware-centred paradigms in distributed application development. *ECOOP 2003 Workshop on Communication Abstractions for Distributed Systems*, 2003.

[40] Das Gupta M. K. 1. Jennison, R. C. Fine structure of the extra-terrestrial radio source cygnus i. *Nature.*, 996:172, 1953.

[41] A. Jessner. Architecture of the effelsberg control system. *SPIE Proceedings.*, 2479, 1995.

[42] Justin L. Jonas. The meerkat radio telescope . *Proceedings.Science.*, 277:172, 2018.

[43] Cosylab et al. K. Žagar, A. Žagar. Integration of alma common software and national instruments labview. *Astronomical Telescopes + Instrumentation.*, 200.

[44] Alan Bridger Kim Gillies et al. Gianluca Chiozzi. Enabling technologies and constraints for software sharing in large astronomy projects. *Proceedings of SPIE.*, 7019, 2008.

[45] Pritchard J. Mellema G. et al. Koopmans, L. Advancing astrophysics with the square kilometre array. (AASKA14), 2015.

[46] J. D. Kraus. *Radio Astronomy*, tomo 2nd ed. 1986.

[47] M. Sekoranja M. Plesko, G. Chiozzi. *ACS Supported BACI Types*, 2005.

[48] Horst H. von Brand. Mauricio A. Araya Joao S. López. odrigo J. Tobar. An amateur telescope control system: toward a generic telescope control models. *Proceedings of the SPIE*, Advanced Software and Control for Astronomy II, 2008.

[49] Li D. Jin C. et al. Nan, R. *The Five-Hundred-Meter aperture spherical radio telescoper (FAST) project.*, tomo 20. 2011.

[50] Jorge Ibsenc Matias Moraa Victor Gonzaleza Nicolás Troncosoa, Horsth. Vonbr. A code generation framework for the alma common software. 2013.

[51] M. Allen Ochsenbein y eds. D. Egret. The gbt precision telescope control system. *Astronomical Data Analysis Software and Systems XIII.*, 314, 2004.

[52] Fübling M. Antonino P. O. Conforti V. et al. Oya, I. The software architecture to control the cherenkov telescope array. *Proceedings of the SPIE.*, 9913:15, 2021.

[53] I. Oya, Joseph Schwarz, y Dejan Deman. The graphical user interface of the operator of the cherenkov telescope. *Proceedings of the 16th International Conference on Accelerator and Large Experimental Control Systems*, 2017.

[54] Wilson R. W. 1. Penzias, A. A. A measurement of excess antenna temperature at 4080 mc/s. *Astrophysical Journal.*, 142, 1965.

[55] P.Sivera. Vlt common software overview. 2007. URL `http://www.eso.org/projects/vlt/sw-dev/wwwdoc/VLT2007/dockit.html`.

[56] Gustavo E. Romero. Large latin american millimeter array. *Science Reviews - from the end of the world.*, I:No.1, 2018.

[57] G. B. Rybicki y A. P. Lightman. *Radiative Processes in Astrophysics*, tomo ohn Wiley and Sons. 1979.

[58] Sommer H. Farris A. Schwarz, J. The alma software system. *Astronomical Data Analysis Software and Systems (ADASS) XIII.*, 314, 2004.

[59] Heiko Sommer. ACS Configuration Database. En *Workshop ESO - Garching 08*. 2004.

[60] S. Weinreb. Radio astronomy from jansky to the future - an engineer's point of view. ., Jansky Lecture, 2011.

[61] S. Weinreb. *An introduction to radio astronomy"*, tomo Sep. 2012.

[62] T. L. Wilson. *Tools of Radio Astronomy.*, tomo 5th ed. 2009.

[63] W. Brunswig J. Schraml y G. Juen. Design and software aspects for the control system of the 30 m. *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series.*, 444, 1983.

[64] L. Z. Gu Y. L. Ding y Y. Yang. Research of application architecture based on distributed middleware ice. *Journal of Computer Applications*, págs. 27–28, 2009.

[65] Jin Gel. Yu Xiaoqi Huang Kun et al. Wang Jian. The design of observatory control system of lamost. *Plasma Science and Technology.*, 8, 2006.

[66] Na Wang. Zhiyong Liu et al. Jun Li. Trends in architecture and middleware of radio telescope control system. *Advances in Astronomy.*, 10, 2021.