Universidad Nacional
de Educación a Distancia

Escuela Técnica Superior de
Ingeniería Informática

Universidad
Complutense de Madrid

Facultad de Informática

Sistema de control y monitorización de cúpula de gran telescopio con detección absoluta de posición de azimut mediante lectura de etiquetas RFID

Igor Gómez Gil de San Vicente

Director: José Moreno Sanchez
Co-director: David Salinas Moreno

Trabajo de Fin de Máster

Máster Universitario
en Ingeniería de Sistemas y de Control

6 de junio de 2025 / Convocatoria ordinaria de junio 2025

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado:

Igor Gómez Gil de San Vicente.

Firma del alumno

Agradecimientos

A Asunción Santamaría y Gonzalo Montañéz por ofrecerme la oportunidad de participar en este proyecto.

A Edgar Saavedra por su trabajo, consejos e ideas aportados al proyecto.

A mis compañeros del CeDint por su apoyo en los detalles y gestiones relacionados con el proyecto.

A todo el equipo del Observatorio de Sierra Nevada por su acogida en nuestras estancias en el observatorio.

A toda mi familia y a mi compañera Alicia por su apoyo incondicional y ánimo continuo.

A José Sánchez y David Salinas, tutores de este proyecto, por sus comentarios y el cuidado y rigor en su revisión.

Resumen

Este proyecto tiene como objetivo la renovación integral del sistema de control de uno de los telescopios del Observatorio de Sierra Nevada, perteneciente al Instituto Astrofísico de Andalucía - CSIC. Entre las tareas principales se encuentra el desarrollo e implementación del sistema de control de movimiento de la cúpula y la monitorización de parámetros operativos clave, tales como la posición en azimut, consumo energético del motor, temperatura, y estado de apertura o cierre del portón.

Para determinar la posición de la cúpula, el sistema emplea una combinación de tecnologías: etiquetas RFID distribuidas a lo largo de la pared de la cúpula rotatoria y un lector RFID fijo que detecta las etiquetas al girar la cúpula, proporcionando así la posición absoluta del azimut. Este sistema se complementa con un sensor magnético encargado de registrar los pasos relativos de movimiento mediante la detección de los dientes de la rueda dentada que acciona la cúpula, lo que permite una actualización continua y de alta resolución de la posición.

El control del movimiento se lleva a cabo mediante un variador que genera la señal trifásica necesaria para accionar el motor de giro. La comunicación con el variador se establece a través del protocolo MODBUS sobre una línea RS-485.

El sistema central de control está basado en una placa de circuito que integra un SBC (“Single Board Computer”) BeagleBone Black con sistema operativo Debian Linux. Esta placa centraliza la lectura de sensores (lector RFID, sensor magnético, entre otros) y gestiona la comunicación con el variador. El diseño y fabricación del hardware corre a cargo de la empresa líder del proyecto, mientras que la implementación del software de control y comunicaciones forma parte del alcance de este proyecto.

El dispositivo se conecta a la red local mediante Ethernet e implementa dos interfaces principales: un API REST compatible con el protocolo ASCOM - Alpaca, estándar en dispositivos astronómicos, y una interfaz MQTT para el control manual y la transmisión de datos de monitorización. Esta última permite la interacción mediante una aplicación web basada en JavaScript y WebSockets.

Finalmente, un servidor central con sistema operativo Ubuntu Linux opera como broker MQTT (Mosquitto) y almacena los datos de monitorización en una base de datos de series temporales (Graphite DB). Estos datos se visualizan mediante el software Grafana, permitiendo un análisis eficiente de las métricas de operación del sistema.

Este proyecto representa una modernización tecnológica significativa que mejorará la precisión y fiabilidad del telescopio, facilitando futuras observaciones astronómicas de alta calidad.

Palabras clave: Observatorio astronómico, cúpula de telescopio, tecnología RFID, protocolo ASCOM-Alpaca, protocolo MQTT, control de azimut, monitorización de datos.

Abstract

This project aims to achieve the renewal of the control system for one of the telescopes at the Sierra Nevada Observatory, part of the Instituto Astrofísico de Andalucía – CSIC. Among the main tasks is the development and implementation of the dome motion control system and the monitoring of key operational parameters, such as azimuth position, motor energy consumption, temperature, and the open or closed status of the shutter.

To determine the dome's position, the system employs a combination of technologies: RFID tags distributed along the rotating wall of the dome and a fixed RFID reader that detects the tags as the dome rotates, thus providing the absolute azimuth position. This system is complemented by a magnetic sensor responsible for recording relative movement steps by detecting the teeth of the gear driving the dome, enabling continuous and high-resolution position updates.

The motion control is carried out using a variable frequency drive (VFD) that generates the three-phase signal required to operate the rotation motor. Communication with the VFD is established using the MODBUS protocol over an RS-485 line.

The central control system is based on a circuit board that integrates a BeagleBone Black Single Board Computer running the Debian Linux operating system. This board centralizes sensor readings (RFID reader, magnetic sensor, among others) and manages communication with the VFD. The hardware design and manufacturing are handled by the project's leading company, while the implementation of the control and communication software falls within the scope of this project.

The device connects to the local network via Ethernet and implements two main interfaces: a REST API compatible with the ASCOM-Alpaca protocol, standard for astronomical devices, and an MQTT interface for manual control and data monitoring transmission. The latter enables interaction through a web application based on JavaScript and WebSockets.

Finally, a central server running Ubuntu Linux operates as an MQTT broker (Mosquitto) and stores monitoring data in a time-series database (Graphite DB). These data are visualized using the Grafana software, allowing efficient analysis of the system's operational metrics.

This project represents a significant technological modernization that will enhance the telescope's precision and reliability, facilitating high-quality astronomical observations in the future.

Keywords: Astronomical observatory, telescope dome, RFID technology, ASCOM-Alpaca protocol, MQTT protocol, azimuth control, data monitoring.

Índice general

Índice de figuras

Índice de tablas

Índice de listados

Tabla de acrónimos

Acrónimo	Significado	Descripción
ASCOM	*Astronomy Common Object Model*	Estándar de interfaces de software que permite la interoperabilidad entre diferentes dispositivos astronómicos (telescopios, cúpulas, enfocadores, etc.).
API	*Application Programming Interface*	Interfaz que permite la comunicación entre aplicaciones o dispositivos. Se usa en ASCOM-Alpaca (REST API) y en comunicación MQTT.
MQTT	*Message Queuing Telemetry Transport*	Protocolo ligero de mensajería para la comunicación máquina a máquina, ideal para aplicaciones de monitorización y control.
RFID	*Radio Frequency Identification*	Tecnología de identificación por radiofrecuencia usada para conocer la posición de la cúpula mediante etiquetas.
MODBUS	(Nombre propio de protocolo)	Protocolo de comunicación serie usado comúnmente en automatización industrial. Se usa sobre RS-485 para comunicar con variadores de frecuencia.
RS-485	*Recommended Standard 485*	Estándar de comunicación serial diferencial utilizado en entornos industriales.
SBC	*Single Board Computer*	Ordenador completo en una sola placa, como el BeagleBone Black usado en este proyecto.
OSN	*Observatorio de Sierra Nevada*	Observatorio astronómico donde se desarrolla el proyecto.
IAA-CSIC	*Instituto de Astrofísica de Andalucía - Consejo Superior de Investigaciones Científicas*	Organismo responsable del observatorio OSN.
USB	*Universal Serial Bus*	Estándar de conexión universal para dispositivos.
GPIO	*General Purpose Input/Output*	Pines de propósito general en sistemas embebidos utilizados para leer sensores o controlar actuadores.
UDP	*User Datagram Protocol*	Protocolo de comunicación sin conexión usado en redes IP.
REST	*Representational State Transfer*	Estilo de arquitectura para APIs, usado en el protocolo ASCOM Alpaca.
DB	*Database*	Base de datos. En este caso, se refiere a Graphite DB, usada para almacenar series temporales.
NTP	*Network Time Protocol*	Protocolo utilizado para sincronización horaria en redes (mencionado implícitamente en algunos contextos).
PLC	*Programmable Logic Controller*	Controlador lógico programable usado en automatización industrial (mencionado como alternativa en el documento).
IAC	*Instituto de Astrofísica de Canarias*	Institución que aparece como referencia en el estado del arte.
EPICS	*Experimental Physics and Industrial Control System*	Plataforma de software usada en observatorios para control remoto.

Acrónimo	Significado	Descripción
RTS2	*Remote Telescope System, 2nd Version*	Sistema de control remoto de telescopios utilizado en entornos automatizados.
GUI	*Graphical User Interface*	Interfaz gráfica de usuario, mencionada indirectamente como parte de la interacción web.
SSL	*Secure Sockets Layer*	Protocolo de seguridad usado en conexiones seguras como MQTT con autenticación.

Capítulo 1

Introducción

1.1. Motivación

El Observatorio de Sierra Nevada (OSN)[1] es un observatorio astronómico de alta montaña situado dentro de la estación de esquí de Sierra Nevada (Granada, España). El OSN es operado, desarrollado y gestionado por el Instituto de Astrofísica de Andalucía (IAA-CSIC) y cuenta con dos telescopios de 1,50 m y 0,90 m de apertura. Los telescopios actuales del observatorio fueron inaugurados en octubre de 1993 y funcionan con un sistema de control basado en un diseño electrónico realizado a finales de los años 80 del siglo XX. Con el paso del tiempo la tecnología del sistema de control ha quedado obsoleta y algunos de sus componentes ya no están disponibles en el mercado por lo que el equipo de mantenimiento depende de los componentes que todavía tienen en stock para realizar reparaciones. En la Figura 1.1 se muestra una imagen desde el exterior de las instalaciones del OSN. La cúpula que se ve a la izquierda tiene 8 metros de diámetro y alberga el telescopio de 1,50 metros de apertura (T-150). La cúpula de la derecha es de 6,5 metros de diámetro y alberga el telescopio de 90 centímetros de apertura (T-90).

Figura 1.1: Instalaciones del OSN desde el exterior. (Fuente: elaboración propia)

Por esta razón el IAA-CSIC ha puesto en marcha un proyecto para la renovación del sistema de control del telescopio de 90 cm. de apertura (T90) que posteriormente se replicará en el telescopio de 150 cm. La actualización del sistema de control utilizando tecnología actual tiene como principales objetivos disminuir los tiempos en los que el telescopio está fuera de servicio por mantenimiento, facilitar las reparaciones y tareas de mantenimiento, mejorar la monitorización del estado y funcionamiento del telescopio y permitir futuras ampliaciones y mejoras en las funcionalidades del mismo. En la Figura 1.2 se muestra una imagen de la vista general del telescopio T-90. En la parte superior de la imagen se aprecia la cúpula y el portón a medio abrir.

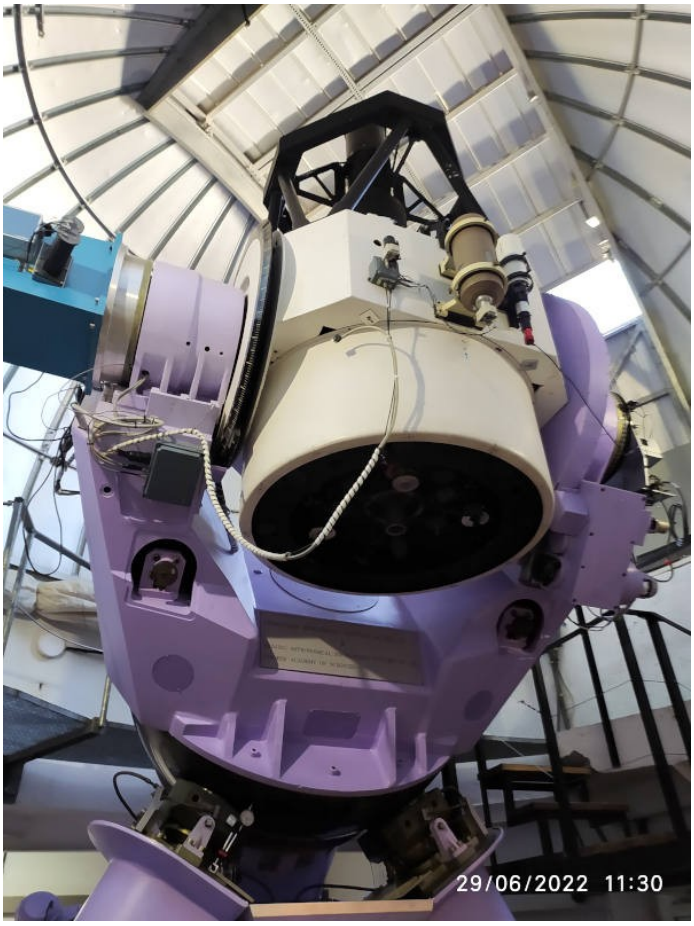

Figura 1.2: Vista general del telescopio T-90. (Fuente: elaboración propia)

Una de las tareas principales del proyecto es el desarrollo e implementación del sistema de control de movimiento de la cúpula y la monitorización de parámetros operativos clave, tales como la posición en azimut, consumo energético del motor, temperatura, y estado de apertura o cierre del portón. El presente trabajo de fin de Máster se ocupa de la implementación del software de control y comunicaciones así como el sistema de monitorización de datos de la cúpula del telescopio T90. El diseño y fabricación del hardware corre a cargo de la empresa que lidera el proyecto.

1.2. Propuesta y objetivos

Uno de los problemas fundamentales a la hora de controlar el movimiento de la cúpula de un telescopio es conocer en todo momento su posición de azimut. Para este fin el presente TFM propone un sistema de lectura de posición basado en etiquetas con tecnología RFID situadas a lo largo de la pared de la cúpula que rota, y un lector de RFID fijo por el que van pasando las etiquetas cuando la cúpula gira. De esta manera cada vez que lector lee una etiqueta obtenemos la posición absoluta del azimut asociado a la etiqueta leída. Además para actualizar la posición entre detecciones de etiquetas y aumentar la resolución se utiliza un sensor magnético que lee el paso de los dientes de la rueda dentada que mueve la cúpula obteniendo pasos de movimiento relativos.

El dispositivo hardware que centraliza la lectura de sensores (lector RFID, sensor magnético, etc) es una placa de circuito que incluye un “Single Board Computer” con sistema operativo Linux donde se ejecuta el software de control y las interfaces de comunicación. Este hardware también controla la actuación sobre el movimiento de la cúpula que se realiza a través de un variador que genera la señal trifásica necesaria para mover los motores de giro de la cúpula. La comunicación con el variador se realiza mediante protocolo MODBUS sobre línea RS-485. El diseño y fabricación del hardware lo realiza la empresa que lidera el proyecto.

La placa se conecta a la red local mediante Ethernet y se implementan dos interfaces de comunicación principales, un API REST para el protocolo ASCOM - Alpaca, que es un estándar para la comunicación entre dispositivos astronómicos y aplicaciones de observación astronómica, y una interfaz MQTT para el control manual mediante una aplicación web basada en JavaScript y websockets y para la comunicación de datos de monitorización.

Por último, un servidor central con sistema operativo Ubuntu Linux ejecuta el broker de MQTT (Mosquitto) y almacena en una base de datos de series temporales (Graphite DB) los datos de monitorización que se visualizan utilizando el software de visualización Grafana.

Los objetivos principales del proyecto son:

- Diseñar un sistema de lectura de la posición de la cúpula fiable y robusto sin partes móviles y sin detectores ópticos que necesiten fuentes de luz.

- Implementar un sistema de control para la cúpula basado en una arquitectura modular y en estándares de comunicación abiertos y bien establecidos como ASCOM – Alpaca[2] para la comunicación con software de observación astronómica y MQTT[3] para el control manual y monitorización de datos.

- Desarrollar un sistema de monitorización y visualización de los principales parámetros de funcionamiento de la cúpula.

1.3. Estructura del documento

En el Capítulo 2 “Estado del Arte” se describen los sistemas más utilizados en el control de cúpulas de telescopio.

En el Capítulo 3 “Materiales y métodos” se detalla el diseño de la solución implementada en este trabajo. Este capítulo se divide en varios puntos según la estructura del sistema: sensores, actuadores, sistema de control y comunicaciones y módulo de mando y monitorización.

En el Capítulo 4 “Resultados” se explican los problemas encontrados durante la realización del trabajo, las pruebas realizadas y los resultados obtenidos para los objetivos del proyecto.

En el Capítulo 5 “Conclusiones y trabajos futuros” se resumen las conclusiones a partir de los resultados obtenidos y las posibles mejoras y ampliaciones futuras.

En el apartado de Bibliografía y referencias se incluyen los materiales y contenidos consultados para la realización del proyecto.

Capítulo 2

Estado del arte

El control de la cúpula del telescopio es esencial para garantizar que la apertura de la cúpula esté siempre alineada con el telescopio y permita observaciones astronómicas sin obstrucciones. En el observatorio donde se realiza la actuación el sistema de control debe ser capaz de coordinarse con el movimiento del telescopio unicamente en el eje de azimut pues la apertura de la cúpula consiste un un portón alargado verticalmente que en el eje de elevación deja al descubierto una franja de cielo que va desde los cero grados de elevación hasta más de 90 grados.

Evolución tecnológica

La evolución de los sistemas de control y movimiento tanto de telescopios como de cúpulas en grandes observatorios astronómicos ha estado estrechamente ligada al desarrollo de la ingeniería mecánica, la electrónica y la informática. En los primeros grandes telescopios del siglo XIX y principios del XX, como el refractor del Observatorio de Yerkes[4] o el telescopio Hooker del Monte Wilson[5], el movimiento del instrumento se realizaba mediante mecanismos puramente mecánicos y eléctricos básicos. La montura del telescopio, usualmente ecuatorial, estaba equipada con grandes engranajes de precisión accionados por motores de corriente continua que seguían la rotación terrestre de forma continua, permitiendo el seguimiento del cielo. Estos motores eran controlados manualmente o mediante sistemas analógicos simples, como reguladores de velocidad ajustados con diales físicos. Las cúpulas, por su parte, se movían de forma independiente, muchas veces con manivelas o motores independientes que requerían supervisión directa del operador para mantener la apertura alineada con el telescopio.

A partir de mediados del siglo XX, se introdujeron los primeros sistemas de automatización analógica más complejos. Los telescopios comenzaron a integrar sistemas eléctricos con sensores de posición y temporizadores que permitían un mejor seguimiento del cielo. Las cúpulas también empezaron a incorporar motores eléctricos controlados desde una consola central, aunque todavía no existía una sincronización automática entre el telescopio y la cúpula. Esto significaba que los operadores debían ajustar manualmente la posición de la apertura de la cúpula a medida que el telescopio se desplazaba, lo que limitaba la eficiencia durante largas sesiones de observación o cuando se realizaban movimientos rápidos entre objetos.

La verdadera revolución llegó en los años 80 y 90 con la llegada de la electrónica digital y los primeros sistemas de control por microprocesador. Los telescopios comenzaron a emplear encóderes ópticos de alta resolución para medir con precisión la posición angular de sus ejes. Estos sensores, junto con controladores digitales, permitieron implementar sistemas de seguimiento mucho más precisos, incluyendo corrección de errores periódicos en los engranajes (PEC) y guías automáticas basadas en cámaras CCD. Paralelamente, las cúpulas se empezaron a equipar con encóderes absolutos y software de sincronización que permitían mantener automáticamente alineada la apertura de la cúpula con la dirección del telescopio.

A principios del siglo XXI, con la proliferación de los ordenadores personales y el software especializado, surgieron plataformas estandarizadas como ASCOM (Astronomy Common Object Model)[2] que facilitaron la integración de diferentes componentes mediante una interfaz común. Estas plataformas permitieron que tanto telescopios como enfocadores, ruedas de filtros, cámaras, monturas y cúpulas pudieran ser controlados desde una única aplicación, incluso remotamente. A nivel técnico, los sistemas de control pasaron a basarse en arquitecturas cliente-servidor, donde cada dispositivo puede responder a comandos estandarizados a través de interfaces COM en Windows, o más tarde mediante protocolos de red. La cúpula, en este contexto, se controla como un dispositivo más dentro del sistema, con funciones de sincronización, apertura, cierre y verificación de estado implementadas mediante comandos estructurados.

En años más recientes, el desarrollo de protocolos basados en red como Alpaca[6], extensión de ASCOM para entornos multiplataforma y comunicación RESTful sobre TCP/IP, ha ampliado las posibilidades de interoperabilidad. Los dispositivos ya no requieren controladores específicos para cada sistema operativo, sino que pueden ser accedidos mediante solicitudes HTTP estándar, lo cual resulta ideal para entornos distribuidos, sistemas embebidos y observatorios robotizados.

Motores y actuadores

Los actuadores más utilizados para accionar el giro en azimut de la cúpula son motores eléctricos de corriente continua o alterna.

Originalmente la cúpula sobre la que se realiza este proyecto disponía de dos motores eléctricos de corriente continua cuya activación se realizaba suministrando o cortando la alimentación directamente con un sistema de relés. Al ser una actuación de todo o nada sin posibilidad de variar la velocidad del motor, la puesta en marcha y detención del movimiento de la cúpula era muy brusca generando golpes que aceleraban el desgaste de los engranajes del sistema mecánico.

Recientemente el equipo técnico de mantenimiento ha sustituido los motores de corriente continua por dos motores de corriente alterna alimentados mediante un variador de frecuencia

con la posibilidad de programar una rampa de aceleración al iniciar el movimiento de la cúpula y una rampa de frenado al detenerlo. De esta manera la operación de la cúpula es mucho más suave con menos ruido y menos desgaste del sistema mecánico. En la Figura 2.1, se puede ver uno de los dos motores instalados actualmente para mover la cúpula. La Figura 2.2 muestra el variador de frecuencia utilizado para generar la alimentación trifásica de los motores y controlar su velocidad.

Figura 2.1: Uno de los dos motores trifásicos que mueven la cúpula. Se puede ver la caja reductora con el eje de salida y el piñón que engrana en la cremallera de la cúpula. (Fuente: elaboración propia)

Figura 2.2: Variador de frecuencia que genera la alimentación trifásica de los motores que mueven la cúpula. (Fuente: elaboración propia)

Sistemas de detección de posición

La posición de orientación de azimut de la cúpula generalmente se realiza mediante encoders rotativos acoplados mecánicamente al sistema mecánico de giro de la cúpula. Existen dos tipos de encoders, los relativos y los absolutos.

Los encoders relativos miden cambios en la posición y necesitan una referencia inicial mientras que los absolutos proporcionan la posición angular exacta, incluso después de un apagado. Inicialmente la cúpula contaba con un encoder rotativo relativo y un sensor magnético que detectaba la posición de un imán fijado a la cúpula. Cada vez que empezaba a operar la cúpula primero era necesario rotarla hasta que el sensor magnético detectaba el imán fijando así la posición de referencia para el encoder relativo.

Posteriormente se sustituyó el encoder relativo por un encoder rotativo absoluto que proporcionaba en todo momento la posición angular absoluta eliminando la necesidad de mover la cúpula a la posición de referencia cada vez que se iniciaba un periodo de observación. En la Figura 2.3 se puede ver el encoder absoluto del fabricante Hohner[7]. Es un encoder de 360 pasos por vuelta con una caja reductora cuya relación de reducción está calculada para hacer corresponder una vuelta de la cúpula con una vuelta del encoder de forma que las salida del encoder, una vez calibrado, proporciona directamente el azimut de la cúpula con una resolución de un grado. En la Figura 2.4 se ve el piñón del encoder que engrana en la cremallera de la cúpula.

Figura 2.3: Encoder absoluto del fabricante Hohner[7]. En la parte superior izquierda se aprecia el imán que proporcionaba la posición de referencia cuando se utilizaba el encoder relativo. (Fuente: elaboración propia)

Figura 2.4: Piñón del encoder absoluto engranado en la cremallera de la cúpula. Se observan un par de etiquetas RFID del nuevo sistema de detección de azimut pegadas a la pared de la cúpula. Para que se puedan leer correctamente las etiquetas tienen que estar un poco separadas de la pared metálica de la cúpula. La separación se ha ajustado de forma que no rocen con el piñón del encoder. (Fuente: elaboración propia)

La solución que se propone para este proyecto es utilizar un sistema de etiquetas RFID que se fijan a lo largo del perímetro de la cúpula de forma que al girar esta, las etiquetas van pasando delante de un lector de RFID estático. Cada vez que el lector detecta una etiqueta, el azimut asociado al identificador único de la etiqueta indica la posición de la cúpula. Además, para incrementar la resolución entre etiquetas se utiliza un sensor magnético que cuenta el paso de los dientes de la rueda dentada de la salida del motor que engrana con la cremallera de la cúpula. En caso de ocurrir un error en la lectura de un tag, el azimut se sigue actualizando utilizando la cuenta de pasos de dientes del sensor magnético. Tanto la lectura de las etiquetas RFID como el paso de la rueda dentada se realizan con sensores sin contacto de forma que se evita el uso del elemento mecánico de la rueda dentada en contacto con la cremallera de la cúpula necesaria en el caso de utilizar encoders rotativos.

Esta solución es bastante novedosa y no conocemos ninguna instalación donde la lectura de la posición de la cúpula se realice de esta manera. El control de movimiento de cúpulas de

observatorios astronómicos de más de 5 metros de diámetro es una aplicación de nicho y los sistemas utilizados son muy específicos de cada instalación en particular, utilizándose principalmente sistemas basados en encoders relativos o absolutos.

Dos de las empresas más importantes de fabricación e instalación de cúpulas para observatorios astronómicos son Ash Dome[8], fabricante de las dos cúpulas del OSN, y Baader Planetarium[9]. Esta última compañía, dispone de cúpulas de entre 3,2 m. y 8,5 m. de diámetro en dos versiones[10], automática y avanzada, equipadas con encoder relativo y absoluto respectivamente para la lectura del azimut de la cúpula. Baader Planetarium tiene instaladas cúpulas en observatorios de todo el mundo. En España, por ejemplo, dos observatorios significativos donde ha participado son el observatorio de Yebes del IGN/CNIG[11] y el observatorio del instituto Max Planck para la investigación del sistema solar[12] situado en el emplazamiento de Izaña en Tenerife.

Controladores electrónicos

Para la lectura de sensores y generación de las órdenes de actuación a los motores se suelen utilizar controladores programables industriales de tipo PLC o sistemas electrónicos a medida basados en microprocesador o microcontrolador.

En el proyecto que nos ocupa se va a utilizar una placa electrónica basada en un “Single Board Computer” (SBC) modelo Beaglebone Black[13] con sistema operativo Linux Debian. En la placa se implementa la electrónica necesaria para realizar la interfaz entre los sensores y actuadores con el sistema Linux. El sistema de control se ejecuta en la SBC y consiste en una aplicación implementada en el lenguaje de programación C.

El diseño del sistema de control actual del telescopio data de finales de los años ochenta del siglo XX. Está organizado de forma modular e implementado en varias placas electrónicas conectadas mediante backplanes y montadas en un armario de electrónica. La placa de procesamiento principal está basada en un microprocesador de 16 bits modelo Motorola 68000. En la Figura 2.5 se muestra una imagen del sistema actual. En la parte trasera del armario se aprecia un ventilador que debe encenderse cuando funciona el telescopio para disipar el calor generado por las fuentes de alimentación que proporcionan la energía para mover los motores de la montura del telescopio.

Figura 2.5: Sistema de control actual del telescopio T-90. (Fuente: elaboración propia)

Integración de software

Originalmente se utilizaban sistemas de comunicación y protocolos propietarios para conectar el software de observación astronómica con el sistema de control de la cúpula. Esto encarecía y complicaba el mantenimiento y actualización de sistemas al limitar las opciones a las ofrecidas por el fabricante del sistema.

Para resolver este problema y promover la flexibilidad e interoperabilidad en el sector de sistemas para telescopios, desde hace algunos años han surgido varias iniciativas para estandarizar los protocolos de comunicación entre el software de observación y el equipamiento de control.

Por ejemplo EPICS (Experimental Physics and Industrial Control System)[14] es un entorno de software libre diseñado para desarrollar sistemas de control distribuidos a gran escala, especialmente en instalaciones científicas complejas como aceleradores de partículas, telescopios, reactores nucleares y laboratorios de física experimental.

En astronomía, EPICS se utiliza ampliamente en telescopios e instalaciones como el Very Large Telescope (VLT)[15], el Atacama Large Millimeter/submillimeter Array (ALMA)[16], o el futuro Telescopio Extremadamente Grande (ELT)[17], donde se requiere coordinar de manera robusta subsistemas complejos como monturas, óptica adaptativa, espectrógrafos y cúpulas.

Otra iniciativa es el sistema RTS2 (Remote Telescope System, 2nd version)[18], un sistema de software libre diseñado para la automatización completa de observatorios astronómicos, con un enfoque especial en la operación remota y autónoma. Fue desarrollado inicialmente a principios de los años 2000 por Petr Kubánek y ha evolucionado hasta convertirse en una solución robusta utilizada tanto en observatorios profesionales como en instalaciones de investigación y educación.

El objetivo principal de RTS2 es permitir que un telescopio funcione de manera completamente automática, sin intervención humana, durante sesiones de observación completas. Para ello, el sistema gestiona de forma integrada todos los componentes del observatorio: telescopios, monturas, cúpulas, cámaras, enfocadores, ruedas de filtros, sensores meteorológicos y sistemas de alimentación eléctrica.

Entre sus características destacadas se encuentran el planificador automático de observaciones, la integración con catálogos astronómicos, el soporte para respuestas rápidas ante eventos transitorios (como estallidos de rayos gamma), y la compatibilidad con protocolos como INDI, ASCOM y EPICS.

RTS2 ha sido implantado en múltiples observatorios alrededor del mundo, especialmente en aquellos que requieren operaciones robotizadas o desatendidas, como los dedicados al seguimiento de satélites, detección de asteroides o vigilancia del cielo.

Un proyecto menos complejo es el protocolo INDI (Instrument-Neutral Distributed Interface)[19], un protocolo abierto diseñado para el control y automatización de instrumentación astronómica. Fue desarrollado originalmente por Elwood C. Downey en el Observatorio Clear Sky Institute a principios de los años 2000, y hoy en día es mantenido activamente por una comunidad de desarrolladores, especialmente dentro del entorno de software libre, con fuerte presencia en plataformas Linux y sistemas embebidos.

Una característica destacada de INDI es su extensibilidad y neutralidad: no está ligado a una plataforma ni a un tipo específico de hardware, lo que permite integrar fácilmente nuevos dispositivos mediante la definición de propiedades y comandos personalizados.

INDI es ampliamente utilizado en proyectos de astronomía amateur avanzada, observatorios educativos, y sistemas de control de telescopios, especialmente en entornos Linux, aunque también se han desarrollado versiones compatibles con otros sistemas operativos.

Otro de los estándares más implantados en grandes observatorios es el protocolo ASCOM-Alpaca[2].

ASCOM (Astronomy Common Object Model)[2] es un conjunto de interfaces de software que permite que los diferentes componentes de un sistema astronómico (como telescopios, enfocadores, ruedas de filtros, cúpulas, cámaras, etc.) puedan interoperar entre sí independientemente del fabricante. Es un estándar de software abierto que define cómo deben comunicarse los controladores de dispositivos astronómicos con las aplicaciones de control y automatización.

ASCOM fue desarrollado inicialmente por Bob Denny a finales de los años 90 y ha evolucionado gracias a una comunidad de desarrolladores, bajo la coordinación de la ASCOM Initiative, una organización sin ánimo de lucro, con el objetivo de resolver la falta de interoperabilidad entre dispositivos astronómicos de diferentes fabricantes.

Este estándar se utiliza para controlar el movimiento y apuntado de telescopios con monturas motorizadas, manejar cúpulas motorizadas y sincronizarlas con el telescopio, operar ruedas de filtros, enfocadores, cámaras CCD/CMOS y otros accesorios, automatizar observatorios remotos o robóticos, e integrar diversos dispositivos y programas en un entorno común (p. ej., con software como Maxim DL, PHD2, TheSkyX, NINA o Sequence Generator Pro).

Entre los beneficios que aporta se pueden destacar la interoperabilidad, al permitir que software de control se comunique con hardware de distintos fabricantes de forma uniforme, y el tratarse de un estándar abierto y gratuito con amplia compatibilidad con software astronómico que evita dependencias propietarias o de controladores específicos.

El núcleo del estándar ASCOM son las interfaces estándar consistentes en conjuntos de métodos y propiedades para controlar y conocer el estado de un dispositivo astronómico. Para que un dispositivo sea compatible con ASCOM, el driver que lo controla debe implementar un interfaz de acceso que ofrezca los métodos y propiedades especificados en el estándar para ese dispositivo. Las principales interfaces definidas en el estándar son: telescopio (ITelescopeV3), enfocador (IFocuserV3), cúpula (IDomeV2), cámara (ICameraV2) y rueda de filtros (IFilterWheelV2).

La iniciativa ASCOM también desarrolla y publica como software abierto y gratuito la plataforma ASCOM (ASCOM Platform), un conjunto de herramientas, bibliotecas, documentación y ejemplos que facilitan el desarrollo de nuevos drivers y permiten simular, testear, instalar y probar dispositivos nuevos.

ASCOM es muy utilizado en todo tipo de observatorios astronómicos, desde observatorios personales, pasando por observatorios universitarios o semiprofesionales, hasta observatorios avanzados profesionales. Su gran aceptación se debe a las ventajas explicadas con anterioridad y

a que está implementado en las principales herramientas software de observación astronómica como Maxim DL o TheSkyX.

Aunque inicialmente el estándar ASCOM se desarrolló basándose en tecnología de componentes COM/.NET limitando su compatibilidad a entornos con sistema operativo Windows, en los últimos años se ha desarrollado la extensión ASCOM Alpaca[6] que especifica los métodos y propiedades de las interfaces ASCOM en forma de API REST permitiendo la utilización del estándar en entornos multiplataforma, sistemas basados en Linux y sistemas embebidos.

Actualmente la comunicación con el controlador de cúpula se realiza mediante un protocolo propietario implementado por el IAA-CSIC para enlazar el software de observación astronómica con el movimiento de la cúpula.

Uno de los trabajos realizados en este proyecto ha sido implementar el API REST correspondiente al interfaz de control de cúpula especificado en ASCOM Alpaca con el objetivo de que el software de observación utilizado en el OSN pueda controlar directamente la cúpula utilizando un driver ASCOM a través de la red ethernet del observatorio. La implementación del interfaz se ha realizado en un sistema Linux embebido y la documentación y herramientas ofrecidas por la iniciativa ASCOM han facilitado su desarrollo y testeo.

Los sistemas de control de grandes telescopios son sistemas de control de tipo industrial muy específicos y son muy heterogéneos por la necesidad de adaptarse a la diversidad de características y equipamientos de este tipo de instalaciones. En los últimos años se encuentran algunos trabajos de investigación orientados a implementar sistemas de control de observatorio basados en tecnologías estandarizadas.

Por ejemplo, el Instituto de Astrofísica de Canarias (IAC) ha desarrollado un nuevo sistema de control para los telescopios nocturnos[20], que permite manejar de forma remota y eficiente tanto el telescopio como la cúpula, integrando sensores meteorológicos y otros dispositivos auxiliares.

En entornos extremos como la Antártida, se han implementado sistemas de control autónomos utilizando EPICS y RTS2. Estos permiten la operación remota y autónoma de telescopios, incluyendo el control de la cúpula, cámaras y adquisición de datos meteorológicos[21].

Algunos observatorios han desarrollado soluciones personalizadas utilizando microcontroladores como Arduino para el control de la cúpula. Por ejemplo, el observatorio ATVS en India implementó un sistema de control de cúpula basado en Arduino, integrándolo con RTS2 para operaciones autónomas[22].

El observatorio OARPAF en Italia ha desarrollado un marco de control estructurado en capas, que incluye una API REST para el control remoto de dispositivos, facilitando la robotización del observatorio[23].

En este proyecto se utilizan protocolos de comunicación estándar, desde RS-485, Wifi y Ethernet para el nivel de enlace, MQTT[3] a nivel de aplicación para comunicación de datos o Ascom-Alpaca como protocolo estándar para comunicar directamente el software de observación con los dispositivos del observatorio. Además, para la visualización de datos de monitorización se utiliza el software Grafana[24], ampliamente utilizado en aplicaciones de monitorización.

Esta combinación de tecnologías y la arquitectura del sistema se han adoptado pensando en la facilidad de mantenimiento, adaptación y ampliación del sistema utilizando componentes tanto software como hardware estándar y de fácil disponibilidad.

Capítulo 3

Materiales y métodos

En este capítulo se describe en detalle la arquitectura del sistema y el funcionamiento de cada uno de sus componentes. En primer lugar se describe la arquitectura general para pasar luego a detallar la implementación de cada componente.

3.1 Arquitectura general

Para el control de la cúpula el elemento más importante del sistema es el dispositivo de control. Este dispositivo consiste en una placa de circuito que integra un SBC BeagleBone Black[13] con sistema operativo Debian Linux donde se centraliza la lectura de sensores (lector RFID, sensor magnético y sensor de apertura de portón), se ejecuta el sistema de control de la cúpula y gestiona la comunicación con el variador para actuar sobre el movimiento de la cúpula. En la Figura 3.1 se muestra la placa de control en la caja de protección donde se instala.

Figura 3.1: Placa de control. A la izquierda se puede ver la SBC Beaglebone Black conectada a la placa con el cable USB del lector RFID en la parte superior y el cable Ethernet en la parte inferior. (Fuente: elaboración propia)

Este dispositivo gestiona los siguientes parámetros de funcionamiento de la cúpula:

Tabla 3.1: Parámetros de funcionamiento y tipo de comunicación.

Parámetro	**Sensor o actuador**	**Comunicación con la placa**
Velocidad y sentido de giro del motor que mueve la cúpula	Variador de frecuencia LS Electric iG5A controlando motor de alterna trifásico	Protocolo Modbus sobre línea serie RS-485
Posición absoluta de azimut	Lector RFID Eccel Chilli USB B1	Protocolo serie sobre USB proporcionado por el fabricante del lector.
Posición relativa de azimut	Sensor magnético.	GPIO
Consumo eléctrico de motor de azimut	Variador de frecuencia LS Electric iG5A	Protocolo Modbus sobre línea serie RS-485
Temperatura de motor de azimut.	Sensor NTC	Conversor ADC a SPI.

Además, en este dispositivo se implementan dos interfaces de comunicación para interactuar con el sistema de la cúpula. Por un lado se implementa el estándar ASCOM – Alpaca para permitir controlar la cúpula directamente desde aplicaciones de observación astronómica con soporte para este estándar de comunicaciones para equipos de observación astronómica. Por otro lado se implementa un API basado en MQTT que se utiliza para publicar los parámetros de monitorización del estado de la cúpula, ofrecer un medio para actualizar los parámetros de control de la cúpula no incluidos en el protocolo ASCOM – Alpaca, y para ofrecer una interfaz de control alternativa a la del protocolo ASCOM – Alpaca que facilite la implementación del control de la cúpula desde otros dispositivos como aplicaciones web o aplicaciones móviles.

El dispositivo de control se conecta a la red de área local de uso general del observatorio a través de Ethernet asignándole la dirección IP 10.10.0.50 de forma fija. Para los dispositivos del nuevo sistema de control se han reservado 20 direcciones IP desde la 10.10.0.50 hasta la 10.10.0.69 de la red de uso general. Las estaciones de observación están en otra subred conectada a la red general a través de un router. La Figura 3.2 muestra un esquema de la red.

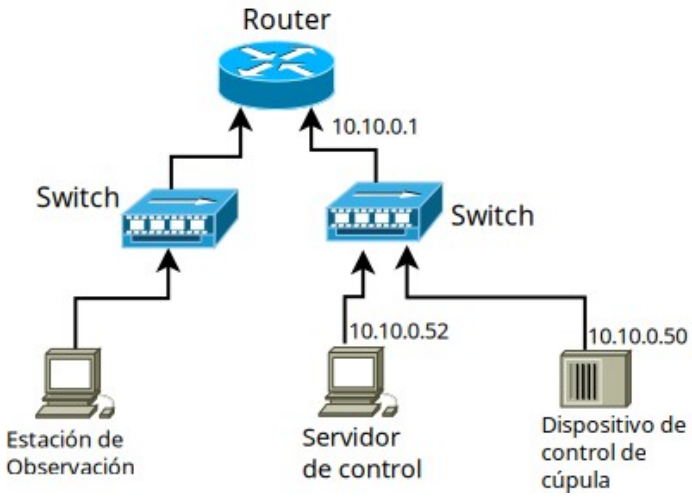

Figura 3.2: Esquema general de la red de área local del observatorio. (Fuente: elaboración propia)

Además de los elementos sensores y actuadores y del dispositivo de control, el último elemento importante del sistema es el servidor central, basado en un PC con sistema operativo Linux Ubuntu, que alberga la base de datos de series temporales Graphite DB[25] y el sistema de visualización de datos Grafana[24] utilizados para visualizar los parámetros de monitorización de la cúpula. Este servidor también alberga el broker MQTT encargado de centralizar las comunicaciones MQTT y un servidor web para servir la aplicación web de control manual de la cúpula.

En la Figura 3.3 se muestra un esquema general de los componentes del sistema y la comunicación entre ellos. Los elementos con fondo gris son aplicaciones de software ya existentes. Todas las aplicaciones utilizadas disponen de licencias permisivas de uso libre. Los elementos con fondo rosa son herramientas de software desarrolladas para este proyecto. Las flechas indican la conexión entre componentes y en su caso el protocolo utilizado en la comunicación.

En el esquema se muestra a la izquierda el servidor central con el broker MQTT (Mosquitto) que centraliza las comunicaciones por MQTT. En el servidor se ejecutan varios servicios que leen del broker MQTT información de monitorización del dispositivo para almancenarla en la base de datos Graphite que luego se visualizará a través del servidor de Grafana. También se muestra, en la misma máquina el servidor web lighttpd[27] utilizado para servir la aplicación web de control manual de la cúpula.

A la derecha se muestra el esquema del dispositivo de control de la cúpula donde se ejecuta el sistema de control implementado en el servicio “control-device.service” con el interfaz de comunicación ASCOM-Alpaca, que puede responder a peticiones de cualquier cliente que soporte este estándar, y conectado también al broker MQTT del servidor central, con una línea con doble flecha indicando que el dispositivo puede tanto enviar datos de monitorización como

recibir comandos de actuación por este interfaz. Se muestra también la conexión del servicio de control con los sensores para leer el azimut de la cúpula y con el variador de frecuencia que controla los motores de la cúpula. Además se puede ver que en el dispositivo también se ejecuta un servicio de monitorización (bbbmonitor-mqtt.service) que envía por MQTT información sobre el estado del dispositivo (carga de CPU, memoria libre, tráfico de datos, etc) al servidor central.

Por último, en la parte inferior se observa el esquema de una posible estación de trabajo desde la que se puede acceder mediante cualquier navegador web tanto al servidor de Grafana para visualizar los datos de monitorización del sistema como al interfaz web de control, además de poder controlar la cúpula desde cualquier software de observación astronómica compatible con el estándar ASCOM-Alpaca.

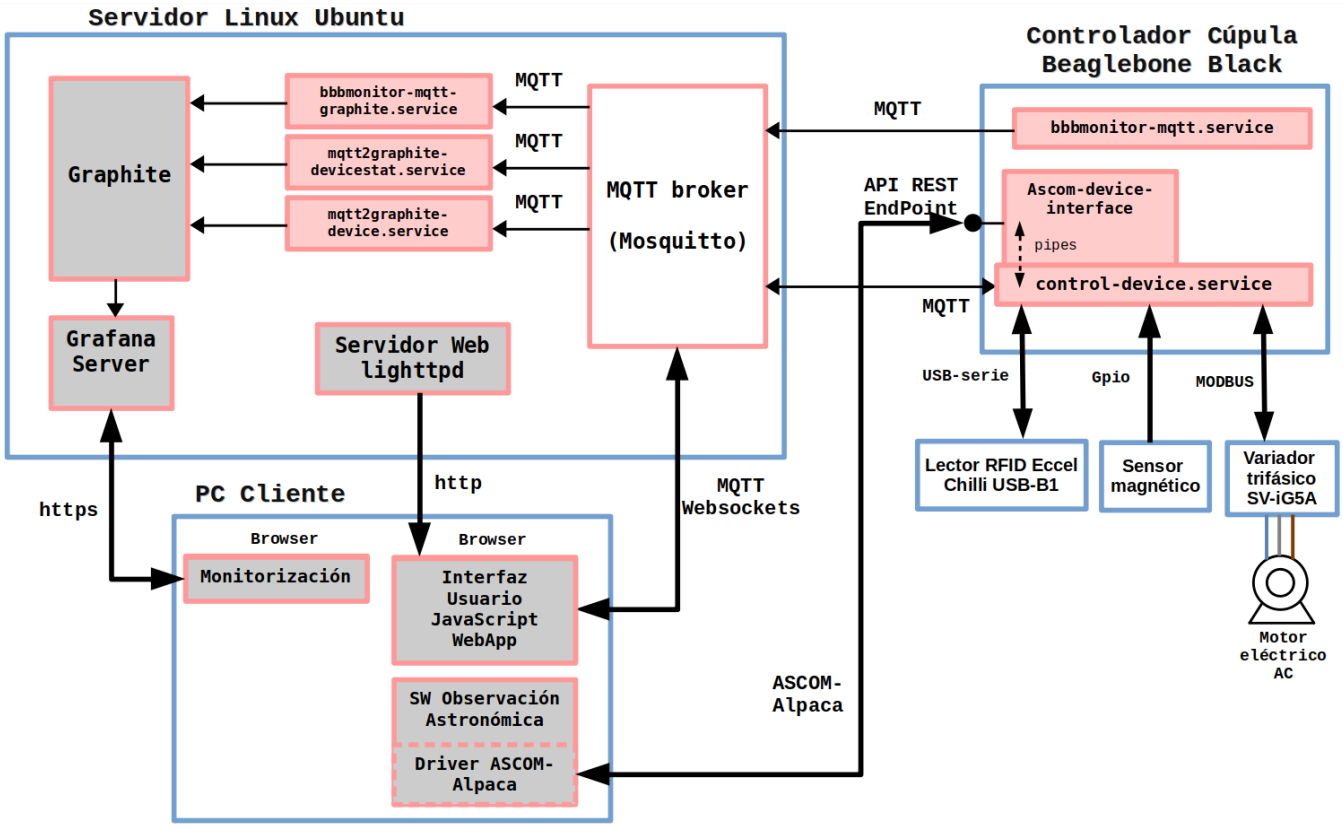

Figura 3.3: Arquitectura de comunicaciones. (Fuente: elaboración propia)

3.2 Sensores

3.2.1 Posición absoluta: sensor RFID

Para detectar la posición absoluta del azimut de la cúpula se han instalado 96 tags RFID equiespaciados a lo largo de la circunferencia de la cúpula. Cada tag tiene asociado un número de identificador único que se relaciona en una tabla con la posición de azimut en la que está colocado en la cúpula. De esta forma un tag RFID, que se mueve con la cúpula, pasa por delante del lector, al leer el identificador del tag y buscarlo en la tabla se conocerá la posición absoluta de la cúpula. La tecnología RFID usada cumple la norma ISO 14443 Tipo A[28] de 13.56 MHz.

Pruebas de selección del tipo de tag y lector RFID

Inicialmente se realizaron pruebas en laboratorio con tags RFID autoadhesivos circulares de 25 mm de diámetro basados en un integrado RFID con tecnología NTAG 213 de NXP, y un lector RFID con antena integrada de 40 mm x 40 mm de tamaño y basado en el integrado de NXP MFRC522[29] gestionado directamente a través de I2C desde una aplicación en la Beaglebone Black. Con esta configuración la lectura fiable del tag sin movimiento se conseguía a partir de una distancia de unos 20 mm entre el lector y el tag. En la Figura 3.4 se puede ver el lector RFID y uno de los tags utilizados en las primeras pruebas.

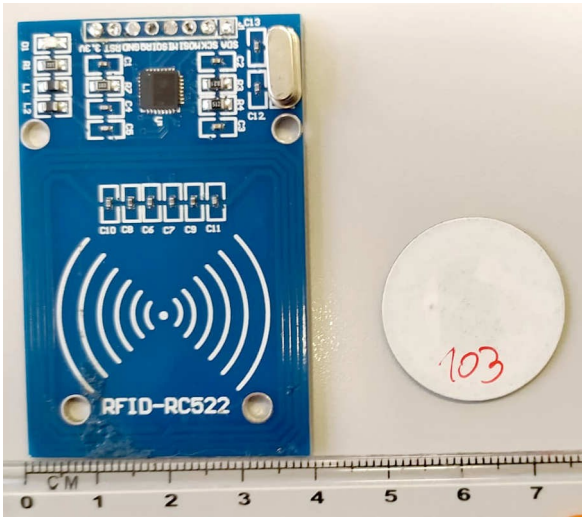

Figura 3.4: Lector RFID y tags iniciales. (Fuente: elaboración propia)

En las primeras pruebas de lectura se observó que la distancia de lectura disminuía al hacer pasar el tag frente al lector con una cierta velocidad. Por esta razón, para simular el movimiento de paso de los tags frente al lector se montaron varios tags en la circunferencia exterior de una rueda de bicicleta que se hacía girar frente al lector RFID. Teniendo en cuenta que la velocidad de movimiento lineal de la cúpula es aproximadamente de 120 mm/s, se impulsaba la rueda para

obtener una velocidad lineal de aproximadamente el doble, ajustando la distancia del lector a los tags hasta conseguir una detección fiable en todas las pasadas sin ninguna perdida de detección. De esta forma se comprobó que la distancia para una lectura fiable en movimiento era de un máximo de 15 mm entre el tag y el lector.

Con estos datos se instaló el sistema en la cúpula ajustando la distancia entre el lector y los tags a 10 mm. Sin embargo, en las pruebas realizadas en la instalación no se consiguió una detección fiable pues aproximadamente en el 10% de los pasos de tags frente al lector no se conseguía una lectura correcta del identificador del tag.

Uno de los motivos de los fallos se debe a que el eje de rotación de la cúpula no es totalmente concéntrico con la circunferencia sobre la que gira, por lo que según rota, la distancia entre el lector fijo y la pared de la cúpula puede variar algunos milímetros. Esto provocaría un error de lectura sistemático de los tags que quedaran más alejados del lector durante la rotación. Sin embargo, aunque los tags más alejados del lector presentaban un mayor porcentaje de fallos, también se producían errores de lectura de forma aleatoria en tags que siempre pasaban frente al lector a menos de los 15 mm de distancia que se habían medido en las pruebas.

Con estos resultados se concluyó que el sistema eléctrico de los motores de movimiento de la cúpula podría estar interfiriendo en el proceso de lectura de los tags y que la distancia máxima de lectura de 15 mm entre el lector y los tags en movimiento era un margen muy pequeño.

Para aumentar la distancia de lectura se decidió utilizar tags más grandes y un lector RFID con una antena mayor. Se repitieron las pruebas en laboratorio con tags autoadhesivos de forma cuadrada con un tamaño de 45 x 45 mm y con protección anti metal. En la Figura 3.5 se muestra uno de los tags RFID utilizados finalmente comparado con uno de los tags probados inicialmente.

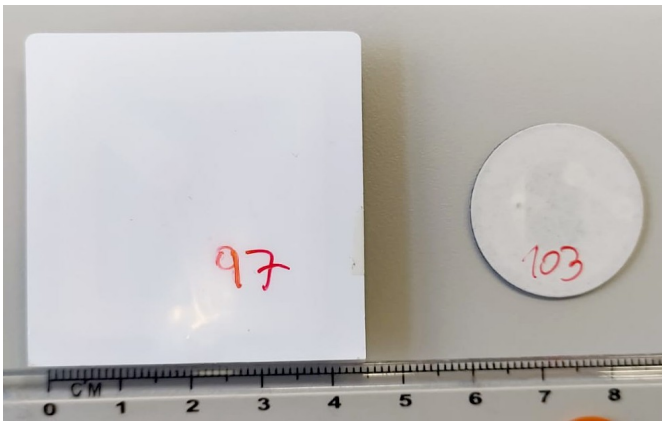

Figura 3.5: Comparación de tamaños del tag inicial (redondo) con el finalmente utilizado (cuadrado). (Fuente: elaboración propia)

El nuevo lector de RFID es el modelo RFID NFC Chilli USB-B1[30] de la empresa ECCEL y se conecta al dispositivo de control mediante un interfaz de comunicación USB a serie. Este lector dispone de un microcontrolador encargado de gestionar el proceso de lectura mediante el integrado de NXP MFRC522. Se ha utilizado una antena RFID externa de forma cuadrada con un tamaño de 80 mm x 80 mm. En la Figura 3.6 se puede ver el nuevo lector junto con la antena externa utilizada en la instalación final.

Figura 3.6: Lector RFID ECCEL Chilli USB-B1[30] con antena externa de 80 x 80 mm. (Fuente: elaboración propia)

Tabla 3.2: Comparación de distancias de lectura entre el lector y tags probados inicialmente con los utilizados finalmente.

Tags y Lector	**Distancia sin movimiento**	**Distancia en movimiento**
Tags redondos de 25 mm de diametro y lector con antena de 40 x 40 mm	20 mm.	15 mm.
Tags cuadrados de 45 x 45 mm y lector con antena de 80 x 80 mm	70 mm.	60 mm.

Las pruebas sin movimiento del tag mostraron un alcance de detección fiable de hasta 70 mm entre el lector y el tag. Sin embargo, al poner el tag sobre una superficie metálica no se consiguieron detecciones correctas lo que mostraba que la lámina de protección anti-metal que

incorpora el tag es muy poco efectiva. Para solucionar este problema se realizaron pruebas separando el tag del metal con diferentes grosores de foam autoadhesivo colocado entre la superficie metálica y el tag.

En la Figura 3.7 se ve una muestra de los dos tipos de tags y diferentes distancias de separación a una chapa metálica utilizada en las pruebas de distancia de lectura.

Figura 3.7: Diferentes tags RFID con varias distancias de separación a una chapa metálica para realizar pruebas de lectura. (Fuente: elaboración propia)

Separando el tag 10 mm de la superficie metálica se consiguió de nuevo una distancia de lectura de 70 mm. Sin embargo en la instalación de la cúpula el espacio libre entre la rueda dentada que engrana en la cremallera para mover la cúpula, y la superficie metálica de esta, también es de unos 10 mm y con esa separación muchos tags chocaban con la rueda dentada. Por este motivo, se ha utilizado una separación de 6 mm entre la superficie metálica y los tags, con la que se consigue una distancia de lectura de unos 40 mm en movimiento.

Tabla 3.3: Distancia de lectura de tags RFID en movimiento con diferentes separaciones a una superficie metálica. Tags cuadrados de 45 x 45 mm y lector con antena de 80 x 80 mm.

Separación entre el tag y la superficie metálica	**Distancia de lectura**
Sin separación	No hay lecturas
6 mm	40 mm.
10 mm	60 mm.

Con esta configuración se realizó una nueva instalación en la cúpula del telescopio situando el lector RFID a una distancia de 30 mm de los tags y protegido en el interior de una caja para equipamiento estándar con protección IP66. En esta ocasión en las pruebas realizadas no se detectó ningún fallo de lectura. En la Figura 3.8 se puede apreciar la instalación del lector RFID y cómo los tags, pegados a la superficie de la cúpula, pasan frente a el.

Figura 3.8: Instalación de lector RFID y tags en la cúpula. (Fuente: elaboración propia)

El lector RFID ECCEL Chilli USB-B1[30] proporciona un API de comandos serie para gestionar las operaciones de detección, lectura y escritura de los tags presentes en su campo de acción. Uno de los comandos es el modo denominado “polling” en el que el microcontrolador del lector que gestiona el transceptor RFID, lanza de forma periódica una operación de escaneo de los tags presentes en su rango de lectura. En caso de detectar un tag transmite por el puerto USB-serie el identificador (UID) del tag detectado.

Para configurar el lector en modo polling, se utiliza una herramienta de línea de comandos proporcionada por el fabricante del lector a la que se llama en el inicio del proceso encargado de leer los tags detectados. Con esta herramienta también se configura la periodicidad del escaneo a 10 escaneos por segundo, la máxima permitida por el lector, y el formato con el que se envía por el puerto USB-serie el UID del tag detectado. Se ha decidido utilizar el formato de cadena de caracteres de dígitos hexadecimales en código ASCII pues en este formato, al final de cada UID detectado el lector envía un carácter de salto de línea que sirve como delimitador para detectar cuando se ha recibido correctamente un UID completo.

Para recibir los UIDs de los tags detectados se utiliza un programa en C que se ejecuta como proceso independiente con la única tarea de configurar el lector al inicio, abrir el puerto USB-serie de comunicación con el lector y quedar a la escucha de los UIDs detectados. Cuando se recibe un UID completo con el tamaño correcto se envía este al proceso de control principal a través de una "cola" de Linux ("Linux FIFO" o "named Pipe").

Los "pipes" de Linux son una herramienta de intercomunicación de procesos que permite que un proceso productor envíe información a un proceso consumidor a través de un buffer de tipo FIFO o cola evitando problemas de sincronización en el acceso al recurso compartido que en este caso es el buffer de intercomunicación.

Utilizar un proceso independiente para leer los tags detectados descarga de esta tarea al proceso de control principal haciendo el sistema más modular. Además, en caso de utilizar lectores diferentes con diferentes formas de comunicación, como ha ocurrido durante el desarrollo y pruebas, sólo es necesario modificar el proceso de lectura de acuerdo al protocolo de comunicación ofrecido por el lector en cuestión.

3.2.2 Posición relativa: sensor magnético

La separación entre los tags RFID es de aproximadamente 3.75 grados. Para obtener una mayor resolución y actualizar la posición entre detecciones de tags consecutivos se utiliza un sensor magnético situado sobre la rueda dentada de salida de uno de los motores que engrana en la cremallera de la cúpula (Figura 3.9).

Figura 3.9: Sensor magnético instalado sobre el piñón de uno de los motores que mueven la cúpula de forma que al rotar el piñón, el sensor detecta el paso de sus dientes. (Fuente: elaboración propia)

El sensor magnético proporciona una señal digital que se activa cada vez que detecta el paso de un diente de la rueda dentada. Esta señal digital se conecta a una entrada digital de propósito general (GPIO) de la Beaglebone Black y se lee mediante un hilo del proceso de control dedicado únicamente a monitorizar la activación del sensor magnético utilizando el driver de gestión de GPIOs disponible en Linux.

Cuando el hilo de lectura del sensor magnético detecta una activación de este, incrementa un contador implementado en una variable con tipo de entero sin signo de 32 bits. Esta variable se declara como variable global y es accesible tanto desde el hilo del proceso de control principal como desde el hilo de lectura del sensor magnético. Al declarar esta variable en C como “volatile” y ser de tipo entero de 32 bits se garantiza que el acceso a ella se realiza de forma atómica haciendo que no sea necesario utilizar técnicas de sincronización para su acceso. En el hilo de lectura del sensor sólo se escribe en la variable mientras que el hilo principal sólo la lee.

De esta manera se consigue tener en el proceso de control un contador que se incrementa con cada detección de paso de los dientes de la rueda dentada que mueve la cúpula. Calculando el ángulo de movimiento de la cúpula correspondiente a cada paso de la rueda y conociendo el sentido de giro en el que se está moviendo el motor, el proceso principal puede calcular y actualizar la posición de la cúpula mientras se mueve entre detecciones de tags RFID.

En las pruebas realizadas se ha medido que el paso de cada diente corresponde a 0.38 grados de giro de la cúpula, mejorando el sistema actual que, con el encoder absoluto, proporciona una precisión angular de un grado. La mayoría de los sistemas de cúpula modernos miden el ángulo de azimut con una precisión mejor que medio grado, mientras que en observatorios de alta gama, esta precisión se reduce típicamente a una décima de grado.

3.3 Actuadores

Los principales actuadores del sistema son los dos motores eléctricos que mueven la cúpula. Se trata de motores eléctricos de corriente alterna trifásicos de las mismas características situados en la circunferencia de la cúpula en extremos opuestos de uno de sus diámetros. Cada motor dispone de una reductora que reduce las revoluciones por minuto del eje de salida final y aumenta el par. La salida de cada reductora cuenta con una rueda dentada que engrana en la cremallera de giro de la cúpula (Figura 3.10).

Figura 3.10: Montaje de uno de los motores de movimiento de la cúpula donde se puede apreciar el eje de salida con el piñón que engrana en la cremallera y el sensor magnético situado justo encima del piñón. (Fuente: elaboración propia)

Los motores se alimentan de una línea trifásica generada por un variador de frecuencia modelo SV-iG5A del fabricante LS Electric[31]. Este variador dispone de un canal de comunicaciones serie RS-485[33] que mediante protocolo Modbus RTU[32] ofrece una amplia variedad de comandos que permiten modificar parámetros de configuración del variador, actuar sobre el movimiento de los motores y consultar el estado de funcionamiento del variador.

Para comunicar con el variador a través del puerto RS-485, se utiliza un puerto serie de la Beaglebone black de tipo RS-232 con niveles TTL que se conecta a un integrado de adaptación de niveles (TI SN65HVD3085EDR[34]) para obtener el formato físico de señal utilizado en el estándar RS-485. La conexión al variador utiliza un par de cables con las señales A y B del estándar RS-485 que utiliza codificación diferencial que no permite comunicación full duplex con un sólo par de cables. Por este motivo el dispositivo de control actúa como maestro iniciando siempre las comunicaciones.

Para que el integrado de conversión a RS-485 pueda saber si se está realizando una transmisión del maestro hacia el esclavo, o si se va a esperar una recepción del esclavo hacia el maestro, dispone de una entrada digital a través de la cual el maestro le indica si va a realizar una transmisión o a esperar una recepción. Esta señal de dirección de comunicación se obtiene de una salida GPIO (P9_23) de la Beaglebone Black y es controlada directamente por el driver de comunicación RS-485 disponible en el sistema operativo Linux. Para utilizar este driver para el control de la señal de dirección ha sido necesario recompilar el kernel de Linux modificando un par de archivos de código fuente correspondientes al código del driver y activando en la configuración la utilización del driver de puerto serie con soporte para RS-485. Además se ha añadido un archivo de tipo “Device Tree Overlay” para indicar al núcleo de Linux la línea de GPIO a utilizar como señal de dirección para la comunicación RS-485. Para realizar estos

cambios se han seguido las indicaciones encontradas en la sección de temas abiertos de la página de GitHub de Robert C. Nelson, uno de los principales mantenedores del sistema Linux para la Beaglebone Black[35].

Una de las funcionalidades del variador de frecuencia es la posibilidad de activar una rampa de aceleración cuando se inicia el arranque del motor y una rampa de deceleración cuando se detiene. Esto permite que el motor no arranque ni se pare de golpe consiguiendo transiciones más suaves con menor desgaste de los componentes mecánicos. La duración de ambas rampas es configurable de forma independiente.

La implementación de las funciones de comunicación con el variador se ha realizado en forma de librería en un módulo del programa de control principal, concretamente en los archivos “motor-control.c” y “motor-control.h”. Para comunicar mediante el protocolo Modbus RTU se ha utilizado la librería de C libmodbus[36].

En la Tabla 3.4 se detallan los comandos de comunicación con el variador implementados en el módulo.

Tabla 3.4: Comandos de comunicación con el variador SV-iG5A.

Comando	Descripción	Parámetro
stop	Detiene el motor	-
EmergencyStop	Detiene el motor sin realizar la rampa de frenado	-
failReset	Reinicia el variador si está en modo de fallo	-
moveFwd	Iniciar movimiento hacia adelante	-
moveRev	Iniciar movimiento hacia atrás	-
setFreq	Escribe la frecuencia de movimiento	Escritura: frecuencia en centihercios (uint16_t)
getFreq	Lee la frecuencia de movimiento	Lectura: frecuencia en centihercios (uint16_t)
setAcctime	Escribe la duración de la rampa de aceleración	Escritura: duración en décimas de segundo (uint16_t)
getAcctime	Lee la duración de la rampa de aceleración	Lectura: duración en décimas de segundo (uint16_t)
setDecctime	Escribe la duración de la rampa de deceleración	Escritura: duración en décimas de segundo (uint16_t)

getDecctime	Lee la duración de la rampa de deceleración	Lectura: duración en décimas de segundo (uint16_t)
getState	Lee el registro de estado de funcionamiento del variador	Lectura: registro de estado (uint16_t)
getFail	Lee el registro de código de error	Lectura: registro de error (uint16_t)
getCurrent	Lee la intensidad de corriente RMS de salida del variador	Lectura: corriente en décimas de amperio (uint16_t)
getVoltage	Lee la tensión RMS de salida del variador	Lectura: tensión en décimas de voltio (uint16_t)
getPower	Lee la potencia activa de salida del variador	Lectura: potencia en centésimas de vatio (uint16_t)
getRpm	Lee la velocidad de giro en rpm del motor	Lectura: velocidad en revoluciones por minuto (uint16_t)

3.4 Sistema de control

3.4.1 Arquitectura general del sistema

El sistema de control se implementa mediante una aplicación escrita en lenguaje C que se ejecuta en la placa Beaglebone Black con sistema operativo Linux Debian. La aplicación se ejecuta como un servicio gestionado con el sistema de “systemd” de Linux.

El código fuente se organiza de forma modular en diferentes archivos de código clasificados según su funcionalidad. Los archivos .c con la implementación de las funciones se guardan en una carpeta de nombre “modules” mientras que los archivos de caberceras .h correspondientes se guardan en la carpeta “include”. En la carpeta raíz de la aplicación se localiza el archivo .c con la función principal (main), el archivo Makefile utilizado para gestionar la compilación de la aplicación mediante la herramienta make de Linux, y los archivos de estado y configuración del sistema de control.

La compilación de la aplicación se realiza en la misma placa Beaglebone Black pues al no tratarse de un software muy extenso es posible compilarlo directamente en la placa en unos segundos, evitando la utilización de herramientas de compilación cruzada.

En la Figura 3.11 se muestra un esquema de los componentes software y de comunicación alojados en el dispositivo de control.

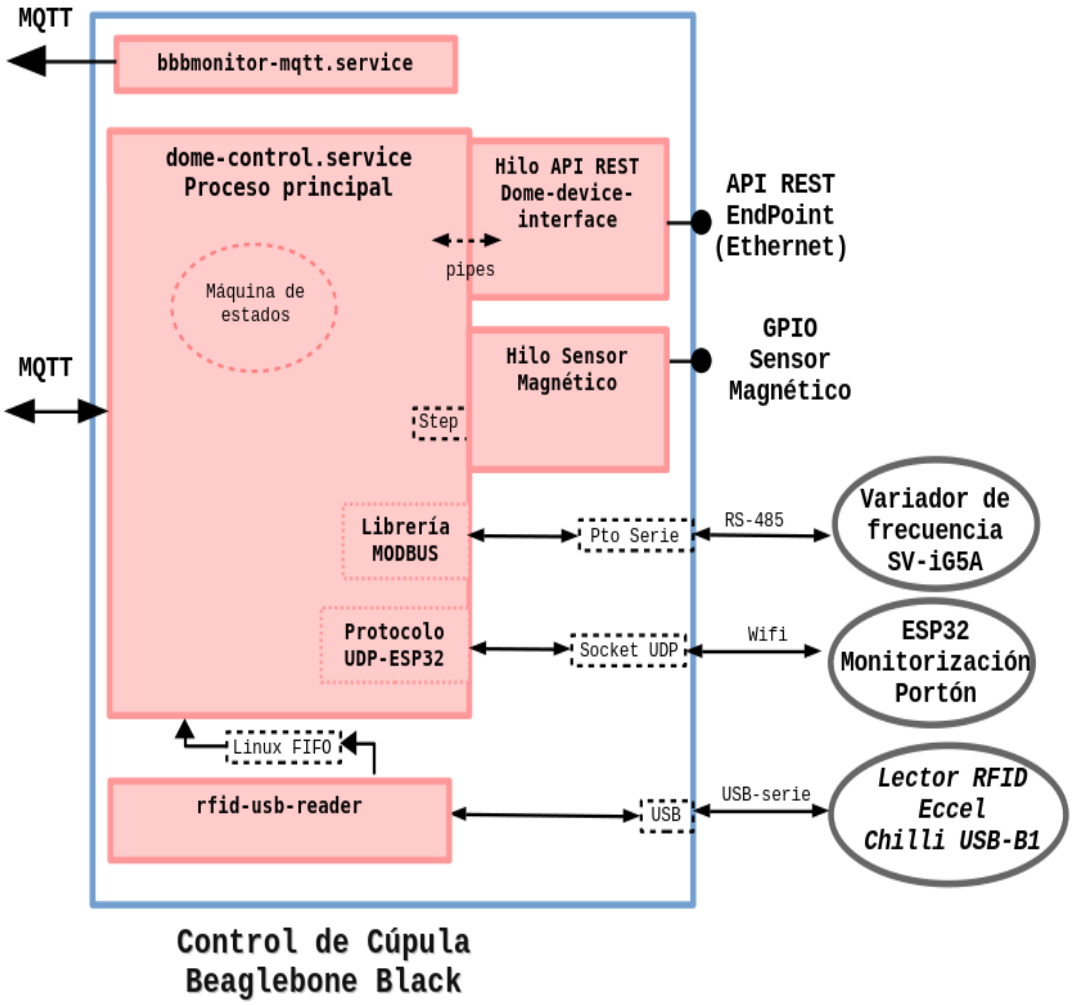

Figura 3.11: Componentes software y de comunicaciones del dispositivo de control. (Fuente: elaboración propia)

En la Figura 3.11 se puede ver el proceso del programa principal de control con sus dos hilos asociados, el que implementa el protocolo de comunicación ASCOM-Alpaca, y el hilo de lectura del sensor magnético. También se ve el proceso de lectura de tags RFID y un proceso de monitorización del estado del sistema que se explica en un apartado posterior.

En resumen, la función de la aplicación de control es realizar la lectura de sensores, actualizar el estado del sistema y controlar los actuadores según el estado del sistema y los comandos recibidos por los interfaces de comunicación.

3.4.2 Programa principal

El programa principal se implementa en el archivo “dome-controller.c” en el que se encuentra la función “main” donde se inicia la ejecución de la aplicación y encargada de inicializar las interfaces de comunicación y los otros hilos asociados al proceso.

En primer lugar se leen los argumentos incluidos en la línea de comandos al llamar al programa. Los argumentos aceptados por el programa se resumen en la pantalla de ayuda del programa mostrada en el Listado 3.1.

```
Formato de llamada: dome-controller [OPCIONES]

OPCIONES:

-V                // Muestra versión.
-v                // Activa trazas de depuracion.
-h                // Muestra esta ayuda.
-a                // Archivo de información de Alpaca Device
-d                // Archivo de información de Ascom Device
-p nº puerto      // Nº de puerto del broker MQTT. Por Defecto: 1883
-S direccion MQTT // Dirección IP o host del broker MQTT. Por defecto: 127.0.0.1
-U usuario MQTT   // Nombre de usuario para autenticar en broker MQTT. Si no se incluye, sin
                     usuario (por defecto)
-P passwd MQTT    // Password para autenticar en broker MQTT. Si no se incluye, sin password (por
                     defecto)
-T topic MQTT     // Topic base. Por defecto: MQTT
-I ClientID MQTT  // Client ID para la conexión con el broker MQTT. Si no se incluye se genera uno
                     automaticamente (por defecto)
-k MQTT keep alive// Intervalo en segudnos entre Keeps Alive de MQTT. Entre 1 y 7200 Por defecto:
                     60
-c certificado    // Path al certificado del broker MQTT si se utiliza SSL, si no se incluye no se
                    utiliza
```

Listado 3.1: Resumen de opciones del programa de control de cúpula.

Luego se inicializa el protocolo de comunicación ASCOM-Alpaca, se lanza el hilo que cuenta las detecciones del sensor magnético, se inicializa la interfaz de comunicación por MQTT y, finalmente, se lanza el bucle de eventos que se ejecuta de forma indefinida.

3.4.3 Bucle de eventos

El núcleo de lectura de sensores, cálculo de posición y actuación sobre los motores se implementa en el módulo “dome-control”.

Tras inicializar la comunicación con los sensores y actuadores, el proceso lee del disco los parámetros de configuración y los últimos datos disponibles de posición y estado y entra en un bucle infinito donde se realizan las siguientes acciones:

- Comunicación con sensores e interfaces de comunicación:

  - Comprobación de peticiones del API REST ALPACA.

  - Comprobación de peticiones del módulo de control.

  - Lectura de eventos del lector RFID.

  - Lectura del estado del variador de frecuencia.

- Actualización de posición y propiedades de estado de la cúpula: a partir de los datos de los sensores se calcula la posición actual de azimut de la cúpula y otras variables de estado.

- Ejecución de la máquina de estados de control: con la situación actual y los comandos recibidos se utiliza una máquina de estados para determinar las actuaciones requeridas y el nuevo estado de la cúpula.

- Envío de datos de monitorización y del estado del sistema.

Al final del bucle se realiza un retardo en el que el proceso se queda dormido dejando libre tiempo de CPU para otros hilos y procesos del sistema. De esta forma se consigue una ejecución periódica del bucle. El intervalo durante el que el proceso se queda dormido se configura a 10 milisegundos con lo que el bucle se ejecuta aproximadamente cien veces por segundo.

El controlador dispone de varias estructuras de datos para representar el estado de la cúpula. Por un lado se utiliza una estructura de datos asociada al interfaz de comunicación ASCOM en la que se implementan las propiedades definidas en la especificación ASCOM-Alpaca para el tipo de dispositivo cúpula. Esta estructura se muestra en el Listado 3.8 y se explica con más detalle en el punto 3.4.7 que trata sobre la interfaz de comunicación ASCOM Alpaca.

Además de las propiedades del protocolo ASCOM es necesario otra estructura con las principales variables necesarias para realizar el control. En el Listado 3.2 se incluye esta estructura implementada en C.

```
struct controller_data
{
    dome_controllerStates_t controllerState; // Estado actual de la máquina de estados
    double parkPosition;                     // Posición de parking asignada a la cúpula
    double homePosition;                     // Posición de home asignada a la cúpula
    double azimutTarget;                    // Objetivo de posición de Azimuth
    uint8_t slewDirection;                   // Sentido de giro de la cúpula: 0: clockwise.
                                             // 1: counterclockwise
    int    magnetCount;                      // Contador del sensor magnético
    uint8_t onTag;                           // 0: tag not detected. 1: tag detected
    struct rfidcomm_tag *tagDetected;        // último tag que se detectó o que se está detectando.
    int currentSpeed;                        // Velocidad actual a la que se está moviendo el motor

    // Parámetros de configuración
    double azimuthTolerance;                 // Tolerancia de Azimuth para comprobar si se está en el
                                             // azimuth Home o Park
    double azimuthDiffRDown;                 // Distancia del azimuth Target a la que se empieza a frenar
    double maxAzimuthDeviation;              // Desviación entre el azimuth absoluto leído de un tag y el
                                             // que estaba en memoria a partir de la que se recalcula el
                                             // movimiento necesario
    double stepGearsensor;                   // Cuantos grados de azimuth son cada cuenta del sensor
                                             // final de carrera de la cremallera
    int maxSpeed;                            // Frecuencia máxima de velocidad del motor con la que
                                             // configurar el variador (en centesimas de Hz)
    int accTime;                             // Tiempo de la rampa de aceleración que hace el variador en
                                             // décimas de segundo.
    int deccTime;                            // Tiempo de la rampa de desaceleración que hace el variador
                                             // en décimas de segundo.
    int countDecc;                           // Cuantas ejecuciones del loop de eventos dura la rampa de
                                             // parada
    int numTags;                             // Número de tags rfid en la cúpula
};
```

Listado 3.2: Estructura de datos que implementa las variables necesarias para el control de la cúpula.

En la estructura controller_data, además de variables de estado como el estado de la máquina de estados, el objetivo de azimut o la dirección de giro actual, también se guardan varios parámetros de configuración del sistema de control de movimiento.

Además, otra estructura de datos importante es la que guarda el estado de funcionamiento de los motores que proporciona el variador de frecuencia. Esta estructura se denomina motor_state y se muestra en el Listado 3.3.

```
struct motor_state
{
        uint8_t  connected;    // a 1: hay conexion con el variador
        uint8_t  stopped;      // a 1: el motor está parado
        uint8_t  movingFW;     // a 1: el motor se mueve hacia adelante
        uint8_t  movingBW;     // a 1: el motor se mueve hacia atrás
        uint8_t  error;        // a 1: el variador está en modo fallo
        uint8_t  accelerating; // a 1: el motor está acelerando (rampa subida)
        uint8_t  braking;      // a 1: el motor está frenando (rampa bajada)
        uint8_t  speedReached; // a 1: se ha alcanzado la velocidad fijada
        uint16_t failCode;     // código de error del variador
        uint16_t current;      // Corriente RMS proporcionada al motor en décimas de amperio
        uint16_t voltage;      // Tensión RMS proporcionada al motor en décimas de voltio
        uint16_t power;        // Potencia activa proporcionada al motor en centésimas de vatio.
        uint16_t freq;         // Frecuencia de la señal trifásica proporcionada al motor
        uint16_t rpm;          // Velocidad de giro del motor
};
```

Listado 3.3: Estructura de datos que implementa las variables de estado de funcionamiento de los motores de movimiento de la cúpula.

A continuación se explica con más detalle las acciones ejecutadas en el bucle de eventos.

Inicialización

Antes de entrar en el bucle de eventos se ejecuta una función en la que se lee el fichero de configuración de parámetros y el fichero con los datos de estado guardados en el último funcionamiento del sistema para inicializar las variables de las estructuras de datos con el estado del sistema.

Luego se inicializan el pipe de comunicación con el proceso de lectura de tags RFID, se lee de un archivo en disco (tagsdata.dat) el azimut correspondiente al identificador de cada tag y se lanza el proceso de lectura de tags.

Finalmente, se inicializa la librería de comunicación por Modbus, se comunica con el variador para configurar los parámetros de funcionamiento y se actualizan los datos de estado del variador. En caso de no poder comunicar con el variador se indica dejando la variable “connected” a cero.

Si la inicialización ha sido correcta se entra en el bucle infinito en el que se realizan las siguientes acciones.

Lectura de peticiones API ASCOM-Alpaca

Para comunicar con el hilo que implementa la interfaz de comunicación ASCOM-Alpaca se utiliza un pequeño protocolo de comunicaciones a través de pipes de Linux. Al iniciar el bucle se llama a la función que implementa este protocolo que recibe posibles peticiones del API a través de un pipe. Según el comando recibido se ejecuta la acción requerida que puede consistir en actualizar una variable de estado como por ejemplo el azimut objetivo y en modificar el estado de la máquina de estados para iniciar una acción cuando esta se ejecute, o en proporcionar al API alguna variable de estado como el azimut actual. Si se ha podido realizar la acción se le responde al API con la información adecuada. Si la lectura de las peticiones ha producido un cambio en el azimut objetivo, se publica el nuevo objetivo por MQTT.

Envío de datos por MQTT

Esta acción no se realiza en cada ejecución del bucle sino que se utiliza un contador para que se ejecute con una frecuencia menor que el bucle de eventos pues para ciertas tareas no es necesario una frecuencia muy alta de repetición. En este caso concreto, esta actualización de datos hacia el exterior por MQTT se ejecuta cada 2000 repeticiones del bucle principal, es decir cada 20 segundos aproximadamente. En este envío de datos se actualiza el azimut objetivo, el estado de funcionamiento correcto del controlador, y el estado de conexión del variador de frecuencia.

Lectura de tags RFID

Se llama a una función que se encarga de ver si se ha recibido algún identificador de tag en el pipe de comunicación con el proceso de lectura de tags. En ese caso, si el identificador recibido se encuentra en la lista de tags se recupera su azimut asociado y se anota que se ha recibido correctamente una lectura de tag.

Actualización de azimut

Se llama a una función encargada de actualizar la posición de azimut. Para ello primero calcula la diferencia en el contador de pasos del sensor magnético desde la última ejecución del bucle. Si se acaba de recibir un tag simplemente se actualiza el azimut con el asociado al tag recibido. Si no hay tag, y si en ese momento se está moviendo la cúpula, se calcula el cambio de azimut en base a la variación del contador de pasos y teniendo en cuenta el sentido de giro en el que se está moviendo la cúpula.

Envío de azimut por MQTT

Al igual que en el anterior envío de datos, esta actualización del azimut por MQTT no se realiza siempre sino que en este caso se envía cada 20 ejecuciones del bucle, es decir, unas cinco veces por segundo.

Actualización y envío por MQTT del estado del motor

En este caso, cada 100 ejecuciones del bucle o aproximadamente cada segundo, se actualiza la estructura de datos con el estado del variador y el motor y se publican por MQTT. Para ello se comunica con el variador de frecuencia para leer la información necesaria. En caso de no poder comunicar se indica por MQTT que el variador está offline. En caso de que el variador estuviera offline pero vuelve a comunicar se le envían de nuevo los parámetros de funcionamiento. También se comprueba si el variador está en estado de fallo. En ese caso se intenta reiniciarlo enviándole el comando de reset.

Ejecución de la máquina de estados

Si el variador de frecuencia está conectado se llama a la función que ejecuta la máquina de estados que en función del estado actual de la máquina realizará las acciones adecuadas para el control de la cúpula. La máquina de estados se explica con detalle en un apartado posterior.

Recepción de comandos por MQTT

Se comprueba si se ha recibido algún comando por MQTT en cuyo caso se llama a la función que los procesa y realiza la acción solicitada implementando así un API de comunicaciones por MQTT que se detalla en un apartado posterior.

3.4.4 Máquina de estados

La máquina de estados consta de 5 estados definidos en un enumerado de C (Listado 3.4).

```
typedef enum {ready, moveFwd, moveRev, slewing, slewingDecc} dome_controllerStates_t;
```

Listado 3.4: Enumerado en lenguaje de programación C con los posibles estados de la máquina de estados.

En la Figura 3.12 se muestra un diagrama con el esquema de la máquina de estados.

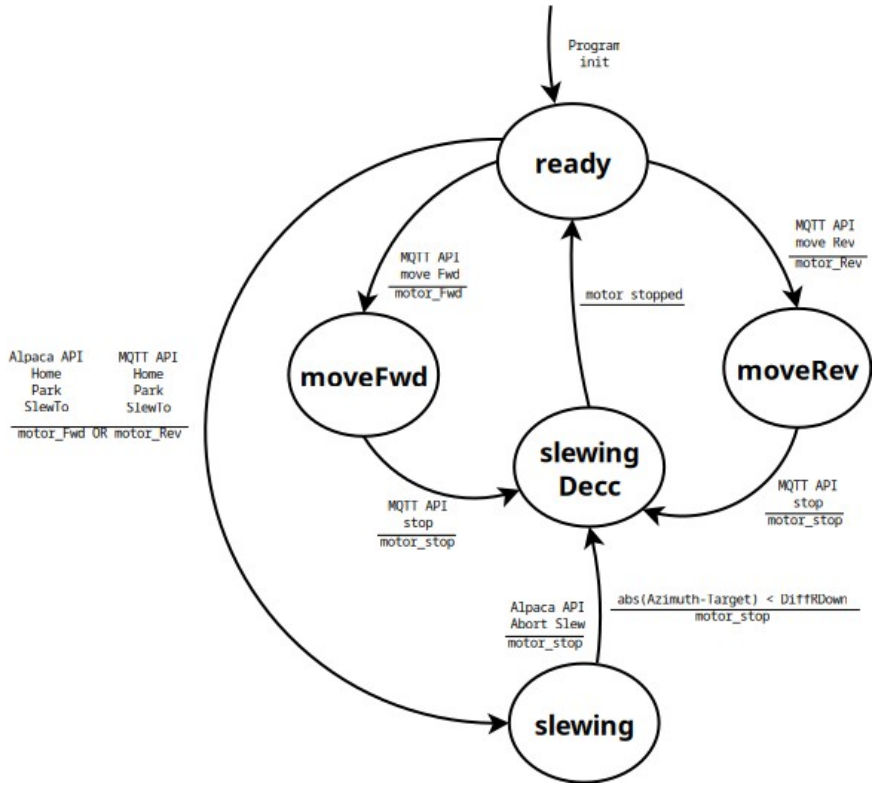

Figura 3.12: Diagrama esquemático de la máquina de estados. (Fuente: elaboración propia)

En el diagrama, las flechas entre dos estados representan una transición de estado. Si la transición conlleva una operación de activación o parada del motor, esta se indica bajo una línea horizontal. Sobre la línea horizontal se indica el evento que ha desencadenado la transición.

A continuación se detalla el funcionamiento de la máquina explicando cada uno de los estados.

- ready:

En este estado los motores de movimiento están parados. No es necesario realizar ninguna comprobación y la cúpula permanece detenida y preparada para recibir comandos. A este estado se llega desde el inicio del programa y desde el estado *slewingDecc* cuando se ha detenido la cúpula.

- moveFwd:

Se activa cuando a través del API MQTT se recibe el comando de activación manual de movimiento hacia adelante. Al activarse se envía al variador de frecuencia el comando para mover los motores hacia adelante. En este estado el movimiento de la cúpula se realiza de forma manual sin especificar una posición objetivo por lo que no es necesario realizar ningún control continuando el movimiento actual hasta que se reciba el comando de parada.

- moveRev:

Se activa cuando a través del API MQTT se recibe el comando de activación manual de movimiento hacia atrás. Al activarse se envía al variador de frecuencia el comando para mover los motores hacia atrás. En este estado el movimiento de la cúpula se realiza de forma manual sin especificar una posición objetivo por lo que no es necesario realizar ningún control continuando el movimiento actual hasta que se reciba el comando de parada.

- slewing:

Se activa cuando se reciben a través de los interfaces de comunicación MQTT o Alpaca los comandos de mover a la posición de parking, mover a la posición home, o mover al azimut objetivo. Cuando se activa, teniendo en cuenta la posición actual y la posición objetivo, calcula la dirección de giro adecuada para llegar al objetivo por el lado más corto y envía al variador de frecuencia el comando para mover los motores en la dirección adecuada.

En este estado se comprueba si la diferencia entre el azimut actual y el azimut objetivo es menor que el parámetro de configuración *AzimuthDiffRDown*. En ese caso se envía al variador el comando para parar el motor y se pasa al estado *slewingDecc*. Cuando se envía el comando de paro, el variador inicia la rampa de frenado o deceleración por lo que la cúpula todavía se mueve durante el tiempo que dura la rampa. Por este motivo, para que la cúpula se detenga del todo justo en el azimut objetivo es necesario enviar el comando de paro un poco antes de llegar a él. Para ello se utiliza el parámetro *AzimuthDiffRDown* que indica la distancia al objetivo a la que empezar a decelerar la cúpula.

- slewingDecc:

Se activa cuando se recibe el comando “stop” por el API MQTT, el comando “abort Slew” por el API ASCOM-Alpaca, o desde el estado *slewing* cuando se está cerca de llegar al azimut objetivo. Al activarse se envía al variador de frecuencia el comando para detener los motores. Se permanece en este estado mientras dura la rampa de frenado aplicada por el variador de frecuencia. Cuando la cúpula se detiene por completo se guarda en un archivo la posición actual y se pasa al estado *ready*.

3.4.5 Parámetros de configuración.

Los parámetros de configuración más importantes que afectan al funcionamiento del control de la cúpula se guardan en el archivo de nombre “*configdata.dat*”. Este archivo se lee al iniciar el controlador para cargar los parámetros guardados con anterioridad. En caso de no encontrar el archivo, no poder leerlo correctamente, o encontrar algún parámetro fuera de su rango permitido de valores, los parámetros se inicializan con un valor por defecto definido como constante en el

código fuente. Los parámetros se pueden modificar editando directamente el archivo desde un terminal de línea de comandos de Linux, o a través del API de comunicación MQTT que incluye comandos tanto para consultar los parámetros actuales como para modificarlos y actualizarlos en el archivo de configuración.

En la Tabla 3.5 se detallan los parámetros de configuración disponibles.

Tabla 3.5: Parámetros de configuración.

Parámetro	**Descripción**	**Rango permitido**	**Valor por defecto**
AzimuthTolerance	Para las posiciones de *parking* y *home*, si la posición actual está dentro de este margen de tolerancia se considera que la cúpula está en la posición de parking o home. Para los comandos de movimiento a un azimut objetivo, si la posición actual se encuentra dentro del objetivo más el margen de tolerancia, se considera que ya se está en el objetivo y no se inicia el movimiento.	0 a 30	0.5
AzimuthDiffRDown	Distancia en grados al ángulo objetivo a la que iniciar la deceleración del motor para que se detenga lo más cerca posible de la posición objetivo.	0 a 30	1
StepGearSensor	Grados de azimut correspondientes a cada cuenta del sensor magnético de la rueda dentada que mueve la cremallera de la cúpula.	0 a 20	0.35
maxSpeed	Frecuencia máxima de velocidad del motor con la que configurar el variador (en centesimas de Hz).	0 a 9000	5000
accTime	Duración de la rampa de aceleración que hace el variador en décimas de segundo.	0 a 300	15
deccTime	Duración de la rampa de frenado que hace el variador en décimas de segundo.	0 a 300	20
numTags	Número de tags RFID instalados en la cúpula.	0 a 180	180

Además del archivo de parámetros de configuración se utiliza otro archivo de nombre "*initdata.dat*" para guardar el estado actual de posición de azimut así como la posición de *parking* y *home.* Cada vez que se detiene el giro de la cúpula, ya sea de forma manual o por haber llegado al azimut objetivo, se actualiza este archivo con el azimut alcanzado. De esta forma, en caso de perdida de alimentación o del apagado del sistema, cuando se vuelve a

encender, al iniciar el sistema se lee de este archivo la última posición registrada del azimut de la cúpula.

En la Tabla 3.6 se detallan los parámetros guardados en este archivo.

Tabla 3.6: Parámetros del archivo *initdata.dat*.

Parámetro	Descripción
Azimuth	Última posición de azimut registrada. Se guarda cada vez que se detiene el giro de la cúpula.
HomePos	Posición de *Home* configurada en el sistema. Posición predefinida en la que se sitúa la cúpula como referencia antes de comenzar una serie de observaciones. Esta propiedad es requerida por la especificación del API de comunicaciones ASCOM-Alpaca.
ParkPos	Posición de *Parking* configurada en el sistema. Posición predefinida en la que se sitúa la cúpula cuando se va a dejar de operar durante un tiempo con ella. Esta propiedad es requerida por la especificación del API de comunicaciones ASCOM-Alpaca.
AtHome	Si es igual a 1 indica que la cúpula se encuentra en la posición de *Home*. En caso contrario su valor es 0.
AtPark	Si es igual a 1 indica que la cúpula se encuentra en la posición de Parking. En caso contrario su valor es 0.

3.4.6 Interfaz de comunicación por MQTT

MQTT[3] es un protocolo ligero de tipo publicación, suscripción, en el que un cliente productor publica información en un canal o topic concreto al que se pueden subscribir uno o más clientes consumidores que deseen recibir la información de ese canal o topic. De esta forma cuando el productor publica un dato en el topic, todos los clientes subscritos a ese topic recibirán en ese momento el dato publicado.

Este modelo de comunicación proporciona una gran flexibilidad pues se desacopla la producción de información de su consumo y posibilita añadir nuevos clientes consumidores de un topic sin modificar la arquitectura de datos permitiendo añadir fácilmente nuevas funcionalidades o modificar las ya existentes.

Este desacoplo entre consumidores y productores se consigue gracias a un nodo centralizado llamado broker que gestiona el envío y recepción de datos entre los clientes a través de los canales o topics. El broker actúa como servidor mientras que las aplicaciones que quieren publicar o subscribirse a información se conectan a el como clientes.

El broker utilizado en este sistema es una implementación de código abierta llamada Mosquitto[26] y se ejecuta en el servidor central. El protocolo MQTT se diseñó originalmente para aplicaciones de comunicación de telemetría y gracias a sus principales características de

sencillez, ligereza, baja latencia y alta velocidad se utiliza ampliamente en el área de “Internet of Things”.

En la Figura 3.13 se puede ver un ejemplo del tipo de comunicaciones utilizado en MQTT.

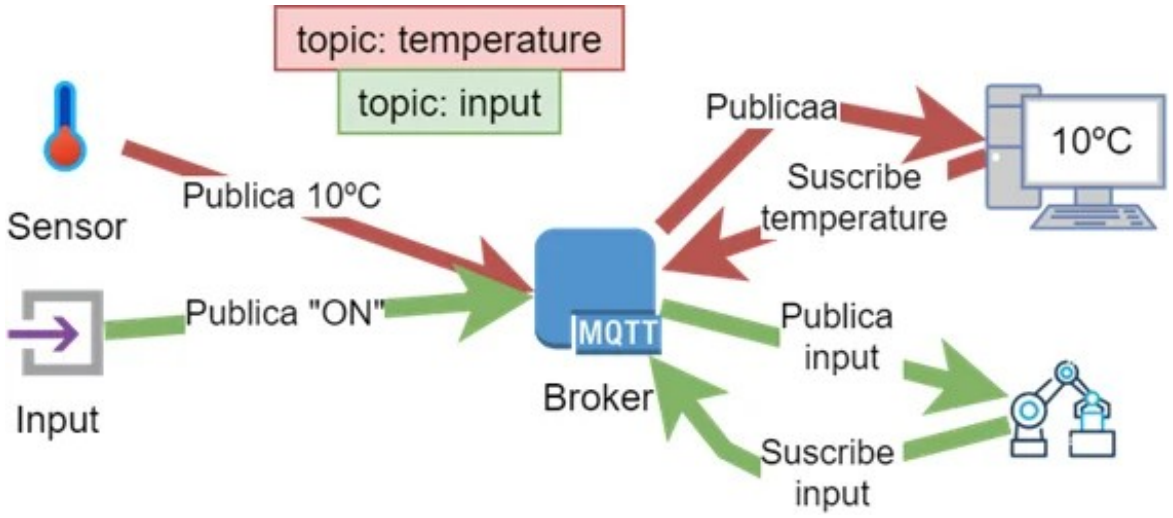

Figura 3.13: Ejemplo de esquema de comunicaciones por MQTT (Fuente: https://iberasync.es/protocolo-mqtt-en-iot/)

Las funciones básicas de comunicación por MQTT se implementan en el módulo definido por los archivos *mqtt-comm.c* y *mqtt-comm.h*. En este módulo a su vez se utiliza la librería de cliente MQTT para C del proyecto “*paho*” de la fundación eclipse[37].

En MQTT el canal básico para enviar (publish) o recibir (subscribe) información es el “topic”. En un sistema con varios productores y consumidores la estructura del sistema se puede organizar en forma de árbol jerárquico utilizando una estructura de topics y subtopics separados por el carácter “/” del mismo modo en el que se estructura la información en carpetas y subcarpetas en un sistema de archivos.

De esta manera se organiza la información en forma de árbol de sistemas y subsistemas encontrando en las hojas del árbol los elementos finales correspondientes a cada parámetro medido o a cada actuador a controlar.

En la tabla 3.7 se detalla la estructura de topics utilizada para la cúpula.

Tabla 3.7: Topics MQTT de publicación de parámetros de la cúpula.

Tópic Raiz	Topic	Descripción
/T90/dome/0	/stat	El programa de control publica el valor “1” en este topic cuando se inicia y de forma periódica en las tareas de mantenimiento para indicar que la aplicación de control está en funcionamiento y con conectividad por MQTT. Este topic se utiliza en la funcionalidad “Will” de MQTT mediante la cual el broker MQTT publica un 0 en este topic cuando detecta que el cliente que ha activado el “Will” ha perdido la conexión. De esta forma consultando este topic se puede saber si la aplicación de control está funcionando y conectada o no.
	/azimuth	El programa de control publica periodicamente la posición de azimut de la cúpula.
	/azimuthObjetivo	Se publica el azimut objetivo de forma periódica y cuando es modificado.
	/variador	Se publica el valor “1” cuando el sistema de control tiene conectividad con el variador de frecuencia. En caso de no comunicar con el variador se publica un “0”
	/motor	Se publican periodicamente los datos de funcionamiento del motor proporcionados por el variador de frecuencia.
	/control	El programa de control se subscribe a este tópic para recibir comandos de control.
	/respuesta	El programa de control publica en este topic el resultado de la ejecución del último comando recibido en el topic /control.

Mediante los topics “control” y “respuesta” se implementa un sistema para ejecutar comandos a través de MQTT.

En la Tabla 3.8 se detallan los comandos disponibles.

Tabla 3.8: Comandos de control por MQTT

Comando	Parámetro	Descripción
slewTo	azimut Objetivo	Se inicia el movimiento de la cúpula para alcanzar el azimut objetivo indicado en el parámetro.
home		Mueve la cúpula a la posición de “home”
Park		Mueve la cúpula a la posición de “parking”
moveFwd		Inicia el movimiento de la cúpula hacia adelante.
moveRev		Inicia el movimiento de la cúpula hacia atrás.
stop		Detiene la cúpula.
azimuthTolerance	Tolerancia de azimut (opcional)	Si se incluye el parámetro se actualiza el parámetro de configuración azimuthTolerance con el valor indicado. Si no se incluye parámetro se muestra el valor actual del parámetro azimuthTolerance.
azimuthDiffRDown	Inicio de Frenado (opcional)	Si se incluye el parámetro se actualiza el parámetro de configuración azimuthDiffRDown con el valor indicado. Si no se incluye parámetro se muestra el valor actual del parámetro azimuthDiffRDown.
stepGear	Resolución sensor magnético (opcional)	Si se incluye el parámetro se actualiza el parámetro de configuración stepGear con el valor indicado. Si no se incluye parámetro se muestra el valor actual del parámetro stepGear.
maxSpeed	Frecuencia máxima (opcional)	Si se incluye el parámetro se actualiza el parámetro de configuración maxSpeed con el valor indicado. Si no se incluye parámetro se muestra el valor actual del parámetro maxSpeed.
accTime	Rampa de aceleración (opcional)	Si se incluye el parámetro se actualiza el parámetro de configuración accTime con el valor indicado. Si no se incluye parámetro se muestra el valor actual del parámetro accTime.
deccTime	Rampa de frenado (opcional)	Si se incluye el parámetro se actualiza el parámetro de configuración deccTime con el valor indicado. Si no se incluye parámetro se muestra el valor actual del parámetro deccTime.

3.4.7 Interfaz de comunicación ASCOM - Alpaca

El estándar ASCOM[2] define interfaces estándar consistentes en conjuntos de métodos y propiedades para controlar y conocer el estado de un dispositivo astronómico. ASCOM define interfaces para varios tipos de dispositivos como puede ser telescopios, enfocadores o cúpulas. Para que un dispositivo sea compatible con ASCOM, el driver que lo controla debe implementar un interfaz de acceso que ofrezca los métodos y propiedades especificados en el estándar.

Es importante mencionar que, aunque para que un dispositivo sea compatible con ASCOM debe responder a todos los métodos especificados, existen algunos métodos cuya funcionalidad es de implementación opcional. Los dispositivos que no implementen la funcionalidad opcional deben responder al método correspondiente con un mensaje de error indicando que no disponen de esa funcionalidad.

El protocolo ASCOM Alpaca[6] especifica estos métodos utilizando un API REST a través de HTTP. El estándar define una serie de métodos comunes para todos los dispositivos compatibles y un conjunto de métodos específicos para cada dispositivo. En este proyecto se ha implementado un interfaz REST que cumple con los métodos especificados en ASCOM Alpaca. En la Figura 3.14 se muestran los métodos comunes a todos los dispositivos que se han implementado.

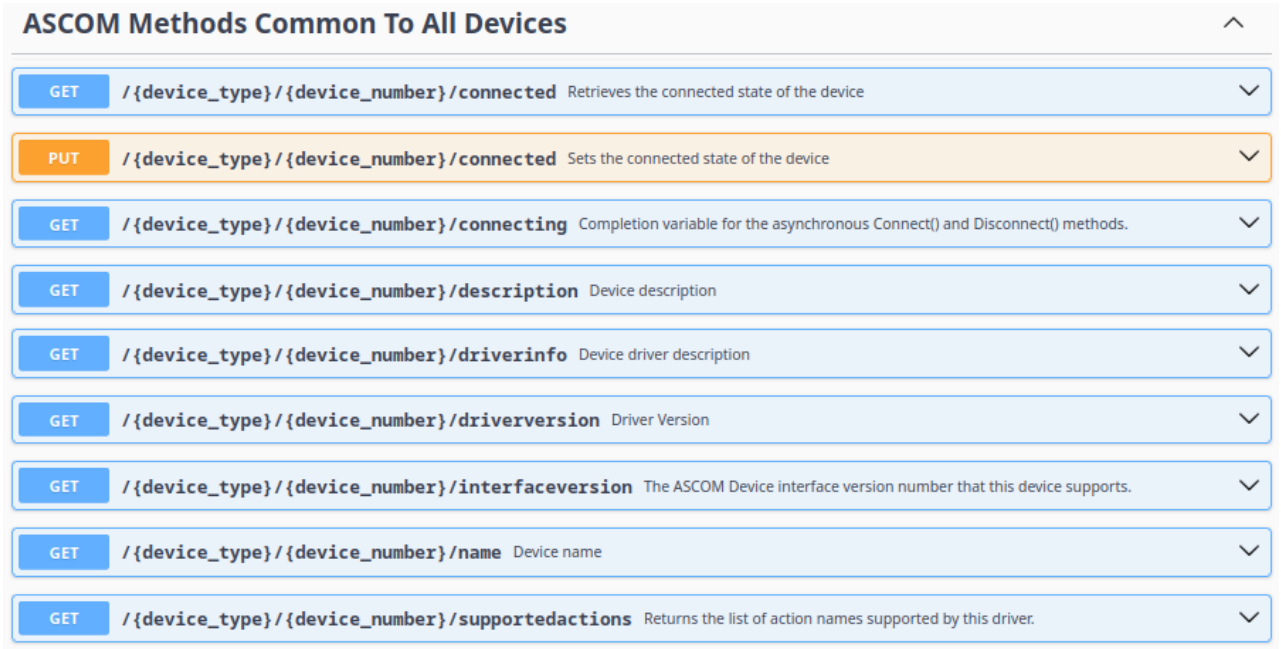

Figura 3.14: Métodos REST de la especificación ASCOM Alpaca, comunes a todos los tipos de dispositivos, que se han implementado en el controlador de la cúpula. (Fuente: https://ascom-standards.org/api/#/)

En la Figura 3.15 se presentan los métodos específicos para el dispositivo de control de cúpula que se han implementado.

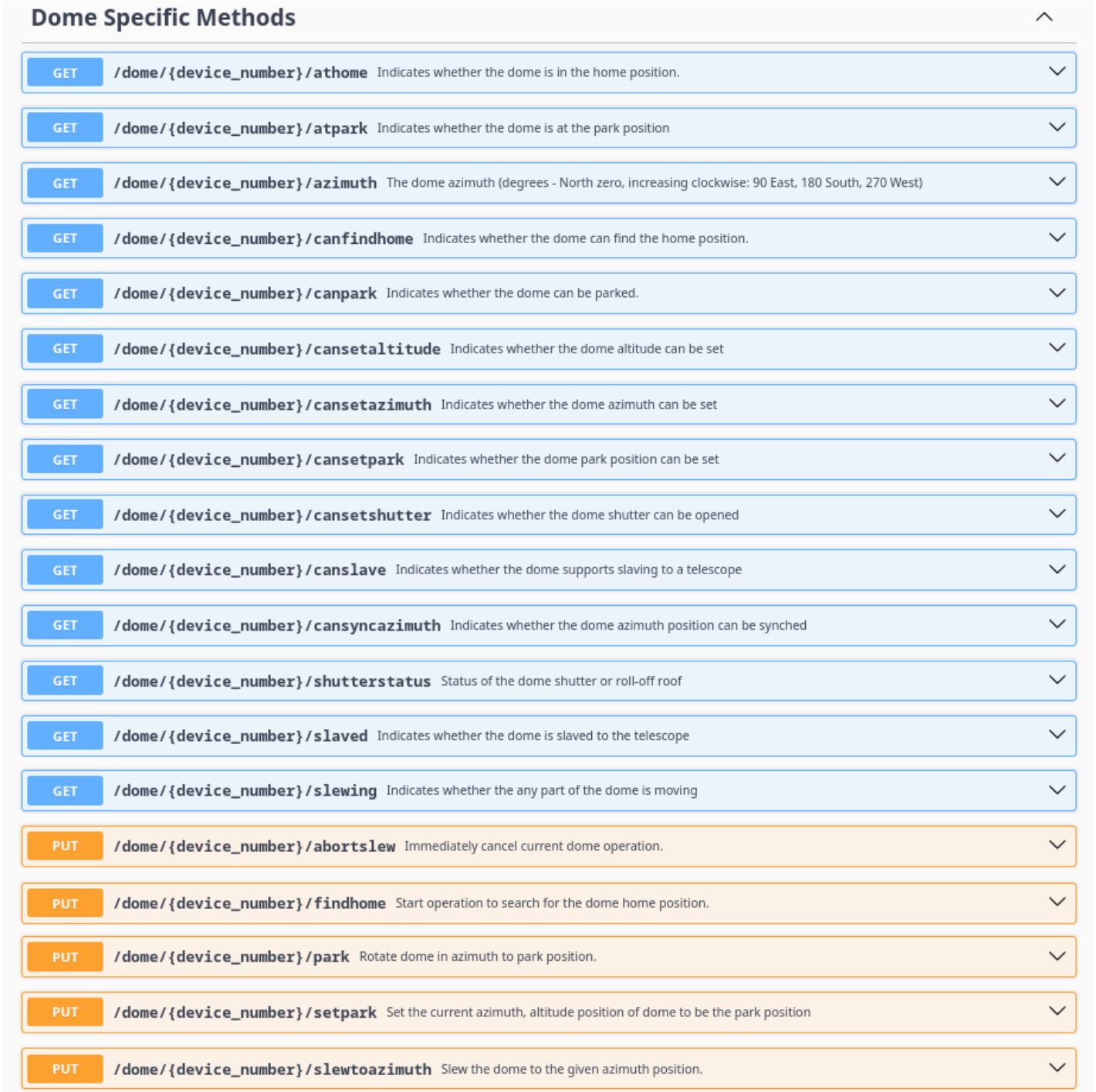

Figura 3.15: Métodos REST de la especificación ASCOM Alpaca, específicos para el dispositivo del tipo cúpula, que se han implementado. (Fuente: https://ascom-standards.org/api/#/)

Los métodos que no se han implementado son opcionales y responden con un mensaje indicando que el dispositivo no implementa esa funcionalidad.

El API REST del estándar ASCOM Alpaca se puede consultar en la web de documentación del API de la iniciativa ASCOM[38].

Además de detallar los métodos del API REST, para entender la implementación del estándar ASCOM Alpaca también es necesario conocer los conceptos de Alpaca Device y ASCOM Device definidos en la especificación.

Un ASCOM Device es una implementación de una interfaz de dispositivo ASCOM, como ITelescope o IDome, a la que se puede acceder a través del IP endpoint del Alpaca Device.

Un Alpaca Device es un dispositivo hardware o servicio software que ofrece uno o más ASCOM Devices mediante el protocolo Alpaca y se comunica con los clientes a través de la red TCP/IP mediante su IP endpoint.

El Alpaca Device más sencillo presentará un único ASCOM Device (por ejemplo, un enfocador o una cúpula) y estará dedicado a esa única tarea. Un dispositivo Alpaca más complejo puede presentar varios ASCOM Devices de distintos tipos a través de un único IP endpoint. En la Figura 3.16 se puede ver un esquema representativo de esta estructura.

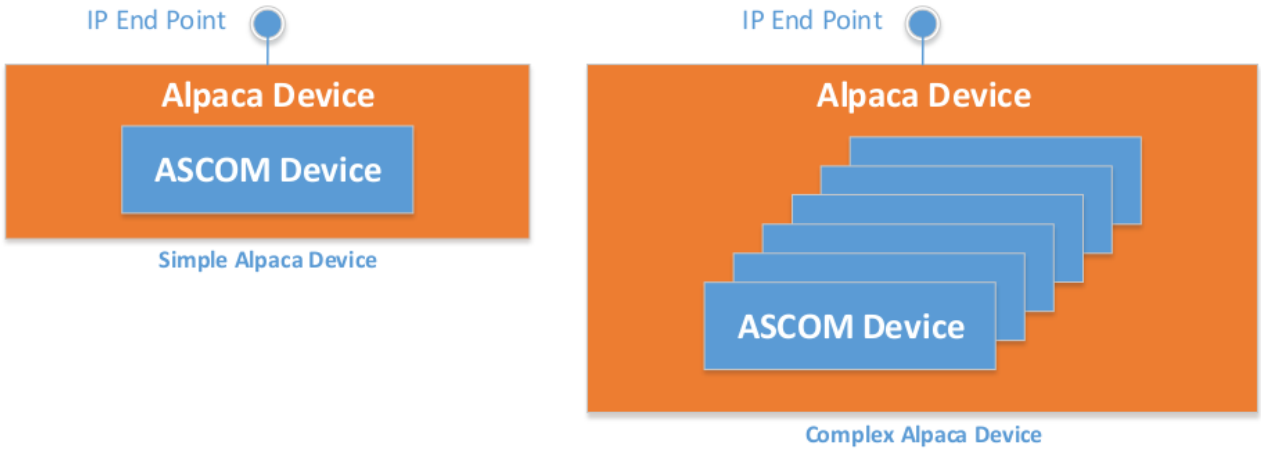

Figura 3.16: Esquema de organización de dispositivos ASCOM y Alpaca. (Fuente: ASCOM ALPACA API Reference - Version 9[39])

En la Tabla 3.9 se muestra un resumen de los conceptos principales de organización del protocolo.

Tabla 3.9: Conceptos principales de ASCOM Alpaca.

Término	Significado
Alpaca Device	Hardware o software que admite los protocolos de la API de Gestión Alpaca y de la API de Dispositivos Alpaca para proporcionar acceso a uno o más Dispositivos ASCOM.
ASCOM Device	Una implementación de una interfaz de dispositivo ASCOM, como ITelescope o IDome, que puede ser accedida a través del protocolo de la API de Alpaca Device.
Tipo de Dispositivo ASCOM	Uno de los tipos de hardware compatibles con ASCOM, p. ej. telescopios, enfocadores, cúpulas y cámaras.
IP endpoint	El host / dirección IP y número de puerto en el que opera el dispositivo Alpaca.

Para implementar este API se utiliza la librería de código abierto Ulfius, un framework de programación de aplicaciones REST HTTP para lenguaje C.

Para implementar el API se ha desarrollado una estructura de código modular siguiendo los conceptos de Alpaca Device (Dispositivo Alpaca) y Ascom Device (Dispositivo Ascom) definidos en la especificación.

El código se ha estructurado de forma que una parte de los módulos que implementan los conceptos de la especificación es común para todos los dispositivos y se puede reutilizar en forma de librería. Por otro lado, existe una parte específica para cada dispositivo en la que se implementan los métodos del protocolo específicos para cada dispositivo.

Módulos comunes de implementación de ASCOM

A continuación se detallan los módulos que forman la parte común de implementación del protocolo.

El módulo “**ascom-defs**” es un archivo de definiciones de códigos comunes para el protocolo ASCOM como por ejemplo códigos de tipos de dispositivo, códigos y mensajes de error, etc. Este archivo se utiliza en otros módulos de la implementación.

El módulo “**alpaca-device**” implementa la estructura de datos y funcionalidad correspondiente a lo que en la especificación ASCOM se define como Alpaca Device. El Alpaca Device implementa el punto de conexión para recibir peticiones ASCOM y puede contener uno o más Ascom Devices. Este módulo implementa la estructura de datos correspondiente a un Alpaca Device y se encarga de crear una instancia de la librería Ulfius y de crear los puntos de entrada o endpoints REST para implementar todas las entradas a la interfaz ASCOM definidas

para un Alpaca Device.

La estructura de datos principal de un Alpaca Device programada en C se presenta en el Listado 3.5.

```
struct alpaca_device
{
     // Informacion y descripción.
     char    serverName[LEN_NAME_DESCR];
     char    manufacturer[LEN_NAME_DESCR];
     char    manufacturerVersion[LEN_NAME_DESCR];
     char    location[LEN_NAME_DESCR];

     // Instancia de Ulfius
     struct _u_instance     ulfInstance;
     // Número de puerto de entrada
     int     portNumber;

     // Dispositivos Ascom que tiene este Alpaca Device
     struct ascom_device    *devices[MAX_DEVICES];
     int     numDevices;
};
```

Listado 3.5: Estructura de datos con las propiedades de un Alpaca Device.

En la estructura se definen las propiedades del Alpaca Device, la instancia de la librería Ulfius, el número de puerto en el que escucha el API REST, y un array con los ASCOM Devices implementados en el Alpaca Device.

El módulo "**ascom-device**" implementa la estructura de datos y funcionalidad correspondiente a lo que en la especificación ASCOM se define como Ascom Device (Dispositivo Ascom). Un "Ascom Device" implementa la interfaz del API correspondientes a uno de los dispositivos de control especificados en el protocolo ASCOM. Todos los dispositivos tienen una serie de propiedades comunes que se implementan en este módulo. En la estructura de datos principal también se incluye una lista de referencias a todos los métodos del API implementados para el dispositivo. La principal función de este módulo, dado un nombre de método del API, es buscarlo en la lista de métodos del dispositivo y ejecutarlo. En caso de no encontrarlo devuelve un error.

La estructura de datos principal de un Ascom Device programada en C se presenta en el Listado 3.6.

```
struct ascom_device
{
  // PROPERTIES COMMON TO ALL DEVICES
  char        deviceName[LEN_DEV_DESCR];   // short name of the driver, for display purposes
  char        deviceType[LEN_DEV_DESCR];
  int         deviceNumber;
  char        uniqueID[LEN_DEV_DESCR];
  char        description[LEN_DEV_DESCR];  // description such as manufacturer and model number
  char        driverInfo[LEN_DRIVER_INFO]; // Descriptive and version information about this
                                           // driver.
  char        driverVersion[LEN_DEV_DESCR];// string containing only the major and minor version
                                           // of the driver. must be in the form "n.n"

  int         interfaceVersion;            // interface version number that this device supports

  uint8_t     connected;                   // connected state of the device

  char        management_htmlSetupdevice_url[64];
  int (*management_htmlSetupdevice_callback)(const struct _u_request * request,
                                             struct _u_response *  response,
                                             void * user_data);

  // PROPERTIES Specific to the specific device
  void    *devProp;

  struct api_interface *device;
};
```

Listado 3.6: Estructura de datos con las propiedades de un Ascom Device.

La estructura define las propiedades comunes a todos los tipos de dispositivo. El elemento de la estructura devProp es un puntero que apunta a otra estructura, explicada más adelante, con las propiedades específicas del ASCOM Device que en este caso se trata de la cúpula. Por último el elemento device es un puntero a una estructura que define los metodos del api implementados en el ASCOM Device y que se define en el módulo “api-interface”.

El módulo “api-interface” implementa la interfaz del API con la lista de métodos que soporta el dispositivo. Cada método se define con un nombre de método y una referencia a una función manejadora (“handler”) que implementa la funcionalidad del método.

Las estructuras de datos que implementan esta interfaz se presentan en el Listado 3.7.

```
struct api_method
{
  char *method_name;
  int16_t (*method_handler)(void *device,
                            char *returnValue,
                            returnType_t *retValType,
                            struct _u_map * inputParams);
};

struct api_interface
{
  struct api_method methods[API_MAX_METHODS];
  uint8_t num_methods;
};
```

Listado 3.7: Estructuras de datos para implementar el acceso a los métodos de la interfaz del API.

En la estructura se ve que el “api_interface” es un array de métodos y que cada método consta de un nombre y un puntero a una función (method_handler) que implementa su funcionalidad.

El módulo “**api-control-comm**” implementa el protocolo de comunicación entre el hilo que implementa el protocolo ASCOM y el proceso principal. El hilo que gestiona el protocolo ASCOM se comunica con el proceso principal mediante un sencillo protocolo de petición - respuesta a través de “pipes” de Linux.

El módulo “**json-response**” implementa las funciones para generar las respuestas en formato Json a las peticiones del API REST de Alpaca. Para generar el formato Json se utiliza la librería jansson[40].

Estos módulos son comunes a todos los dispositivos y se reutilizan en cada uno incluyéndolos como librería.

Módulos específicos de dispositivo

Además, cada dispositivo implementa varios módulos con la funcionalidad concreta de ese dispositivo.

En el caso de la cúpula, en el módulo “**device-dome**” se implementan las estructuras de datos y funciones específicas correspondientes al interfaz del dispositivo Ascom del tipo “Dome”.

En el Listado 3.8 se muestra la estructura de datos con las propiedades específicas de un dispositivo Ascom de tipo “Dome”.

```
struct device_props
{
    // PROPERTIES SPECIFIC TO THE DEVICE
    uint8_t AtHome;    // Boolean. Indicates whether the dome is in the home position
    uint8_t AtPark;    // Boolean. true if the dome is in the programmed park position
    double Azimuth;    // The dome azimuth (degrees, North zero and increasing clockwise, i.e,
                       // 90 East, 180 South, 270 West). North is true north and not magnetic
                        // north.
    uint8_t CanFindHome;    // true if driver can perform a search for home position.
    uint8_t CanPark;        // true if the driver is capable of parking the dome.
    uint8_t CanSetAltitude; // true if driver is capable of setting dome altitude
    uint8_t CanSetAzimuth;  // true if driver is capable of rotating the dome in order to observe
                            // at a given azimuth.
    uint8_t CanSetPark;     // true if the driver can set the dome park position
    uint8_t CanSetShutter;  // true if the driver is capable of opening and closing the shutter
    uint8_t CanSlave;       // true if the dome hardware supports slaving to a telescope
    uint8_t CanSyncAzimuth; // true if the driver is capable of synchronizing the dome azimuth
                            // position using the SyncToAzimuth(Double) method.
    shutterState_t ShutterState;  // status of the dome shutter or roof structure.
    uint8_t Slaved;     // true if the dome is slaved to the telescope in its hardware, else
                        // false.
    uint8_t Slewing;    // true if any part of the dome is currently moving, false if all dome
                        // components are stationary.
};
```

Listado 3.8: Propiedades específicas de un dispositivo Ascom de tipo Dome.

De la estructura device_props, las variables más importantes y que se utilizan directamente en el controlador son la de Azimuth, que guarda la posición de azimut en la que se encuentra la cúpula, y la de Slewing, que indica si la cúpula está en movimiento o no.

Esta estructura se inicializa al arrancar el programa en la función "**dome_device_init**".

Las propiedades que indican las capacidades del driver (empiezan por "Can") se inicializan como se indica en el Listado 3.9.

```
CanFindHome    = 1;
CanPark        = 1;
CanSetAltitude = 0;
CanSetAzimuth  = 1;
CanSetPark     = 1;
CanSetShutter  = 0;
CanSlave       = 0;
CanSyncAzimuth = 0;
```

Listado 3.9: Inicialización de las capacidades del driver de control de cúpula.

Los métodos del API REST del dispositivo ASCOM "Dome" se implementan en el módulo "dome-api".

En la Figura 3.17 se muestra un esquema de los principales módulos que implementan el interfaz ASCOM Alpaca y sus relaciones.

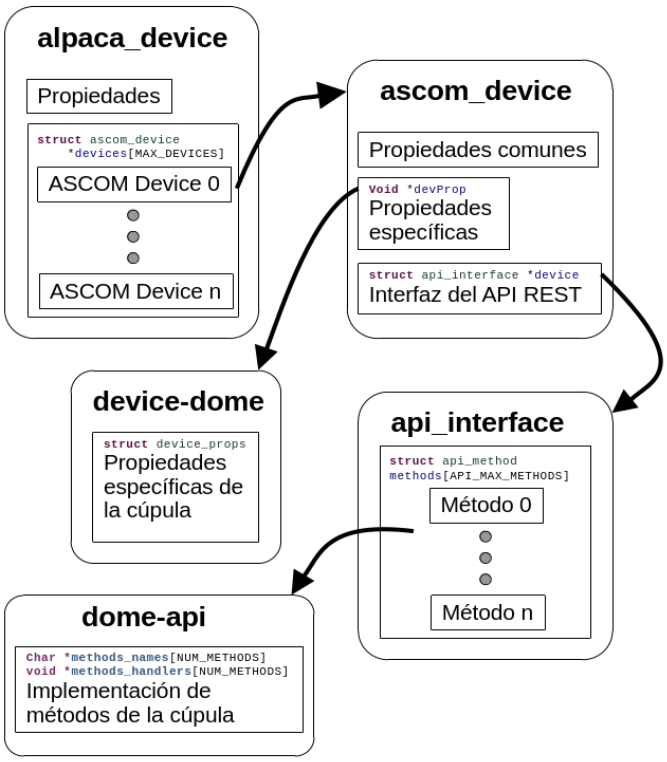

Figura 3.17: Esquema de organización de módulos de implementación de la interfaz ASCOM Alpaca. (Fuente: elaboración propia)

Comunicación entre hilo ASCOM y proceso principal

El proceso principal del controlador, explicado en el punto 3.4.2, se ejecuta en un hilo de ejecución independiente del hilo de la librería Ulfius que implementa el estándar ASCOM Alpaca y atiende las peticiones recibidas por el interfaz REST.

Cuando se arranca el programa principal, una de sus tareas de inicialización es lanzar la instancia de la librería Ulfius que pone en marcha el servidor que atiende las peticiones REST recibidas por el IP endpoint de la interfaz. Este servidor se ejecuta en uno o varios hilos independientes del hilo del programa principal.

La estructura de datos con las propiedades de un dispositivo ASCOM reside en una zona de memoria compartida tanto por el proceso principal como por el hilo de comunicación ASCOM asociado a dicho dispositivo.

Algunas de las propiedades de la estructura de datos de un dispositivo son datos de configuración que se inicializan al principio del programa y luego ya no vuelven a cambiar. Puesto que estas propiedades son estáticas no presentan problemas de concurrencia en el acceso a ellas y el hilo ASCOM accede a ellas directamente referenciando el campo correspondiente de la estructura de datos.

Sin embargo, hay otra serie de propiedades dinámicas que varían durante la ejecución y constituyen el estado de funcionamiento del dispositivo. Para evitar problemas de concurrencia, el acceso a estas propiedades desde el hilo ASCOM se realiza a través de un protocolo de comunicación de tipo petición – respuesta utilizando la herramienta de comunicación entre procesos de “pipes” proporcionada por el sistema operativo Linux.

Cuando el hilo ASCOM recibe una petición para consultar una de estas propiedades, el hilo envía un mensaje a través de pipe al proceso principal indicando la propiedad a leer. El proceso principal al recibir el mensaje envía al hilo una respuesta con el valor de la propiedad solicitada.

Del mismo modo si la petición recibida por ASCOM implica un comando de control, el hilo ASCOM envía un mensaje al proceso principal con el comando a ejecutar y los posibles parámetros. Cuando el hilo principal recibe el comando, lo ejecuta y envía una respuesta con el resultado al hilo ASCOM.

De esta manera se evitan problemas de concurrencia en el acceso a recursos compartidos por varios hilos o procesos de ejecución, centralizando en el proceso principal la actualización y control de dichos recursos.

En la Figura 3.18 se muestra un esquema explicativo de los hilos de ejecución del programa y la comunicación con pipes.

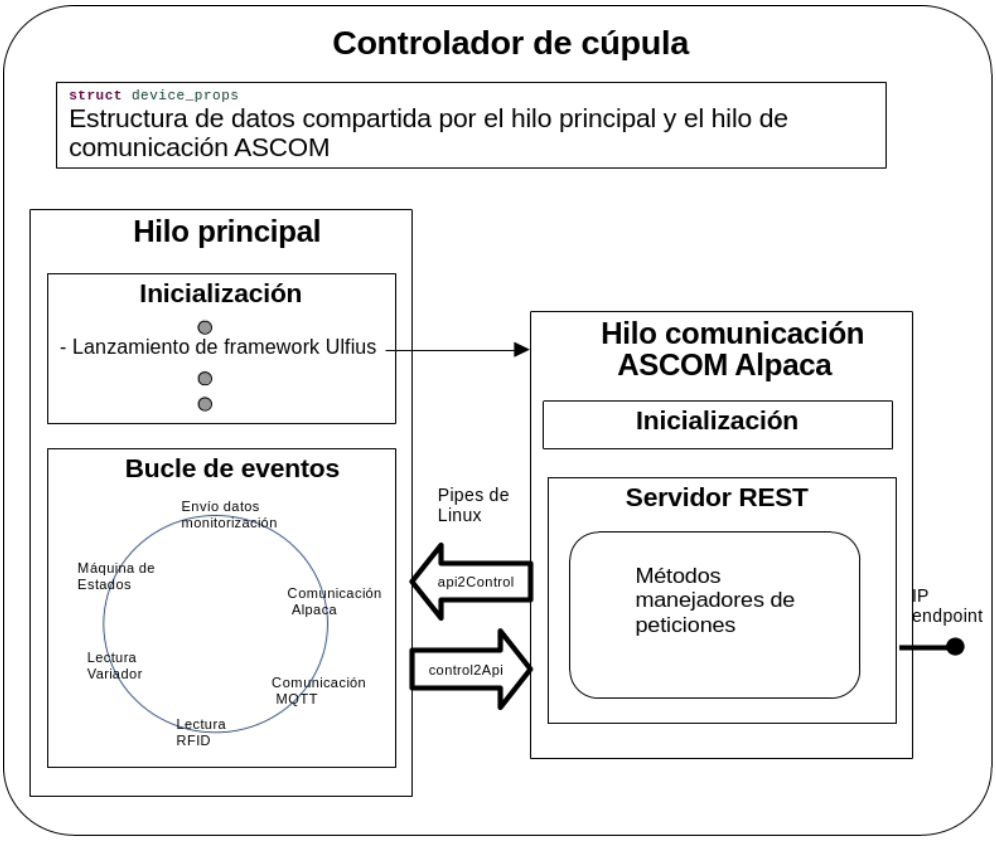

Figura 3.18: Esquema de intercomunicación de hilos mediante pipes de Linux. (Fuente: elaboración propia)

Funcionamiento general

En la Figura 3.19 se puede ver el formato de las peticiones REST en el estándar ASCOM Alpaca.

Element Number	Element	Description
1	api	Fixed lower-case text denoting the root of the API path
2	vversion_number	Integer API version number prefixed with a lower-case v
3	device_type	ASCOM device type e.g. camera, telescope, focuser etc.
4	device_number	Integer device number of the required device
5	command	Command to be processed by the device in lower-case

Figura 3.19: Documentación de la especificación ASCOM-Alpaca para el formato de las peticiones REST. (Fuente: ASCOM ALPACA API Reference - Version 9[39])

En la Tabla 3.10 se muestra un ejemplo de cómo se construye la URL de una petición al API.

Tabla 3.10: Ejemplo de URL para petición al API REST de ASCOM Alpaca.

Elemento	Protocolo	Host	“api”	N.º Versión	Tipo de dispositivo	Número de dispositivo	Comando
Ejemplo	http://	api.peakobservatory.com	api	v1	dome	0	atpark
Url final	http://api.peakobservatory.com/api/v1/dome/0/atpark						

Cuando llega una petición al API REST del dispositivo se ejecuta el handler principal del “endpoint” del API, implementado en el módulo “alpaca-device” (función alpaca_device_callback).

Este handler extrae los parámetros del path de la petición (número de versión, tipo de dispositivo, número de dispositivo). Si los parámetros son correctos busca el dispositivo Ascom indicado en ellos en la lista de dispositivos Ascom implementados en el “Alpaca Device”.

En caso de encontrarlo construye el nombre del método invocado y llama a la función “ascom_device_run_method” implementada en el módulo “**ascom-device**”. Esta función busca el método en la lista del API del dispositivo y si lo encuentra llama al handler que lo implementa.

La ejecución del método genera un resultado que se incluye en la llamada a la función “buildJsonResponse” (implementada en el módulo “**json-response**”) para generar el Json de respuesta que se envía como resultado a la petición del API.

En el Listado 3.10 se muestra un ejemplo de petición al API y respuesta JSON:

```
GET /api/v1/telescope/0/canslewasync?ClientID=1&ClientTransactionID=20

{
  "Value":true,
  "ClientTransactionID":20,
  "ServerTransactionID":168,
  "ErrorNumber":0,
  "ErrorMessage":""
}
```

Listado 3.10: Ejemplo de petición al API y respuesta JSON:

Si no se encuentra el dispositivo solicitado o el método invocado, se envía la respuesta de error especificada en el protocolo ASCOM. En la Figura 3.20 se muestran los tipos de error.

Condition	Alpaca Error Number	COM Exception Number
Successful transaction	0x0 (0)	N/A
Property or method not implemented	0x400 (1024)	0x80040400
Invalid value	0x401 (1025)	0x80040401
Value not set	0x402 (1026)	0x80040402
Not connected	0x407 (1031)	0x80040407
Invalid while parked	0x408 (1032)	0x80040408
Invalid while slaved	0x409 (1033)	0x80040409
Invalid operation	0x40B (1035)	0x8004040B
Action not implemented	0x40C (1036)	0x8004040C

Figura 3.20: Documentación de la especificación ASCOM-Alpaca para los códigos de error. (Fuente: ASCOM ALPACA API Reference - Version 9[39])

Alpaca Discovery

La especificación ASCOM / ALPACA define un mecanismo de descubrimiento automático de dispositivos desde aplicaciones clientes.

En base a la especificación se ha desarrollado un servicio de descubrimiento llamado "**discovery-service**" que se ejecuta como proceso de servicio en la misma máquina donde se ejecuta el dispositivo Alpaca. Gracias a este servicio, las aplicaciones externas que cumplan con el protocolo ASCOM pueden realizar una búsqueda automática de los dispositivos Alpaca presentes en la red local.

En la Figura 3.21 se muestra un esquema del funcionamiento del servicio de descubrimiento automático.

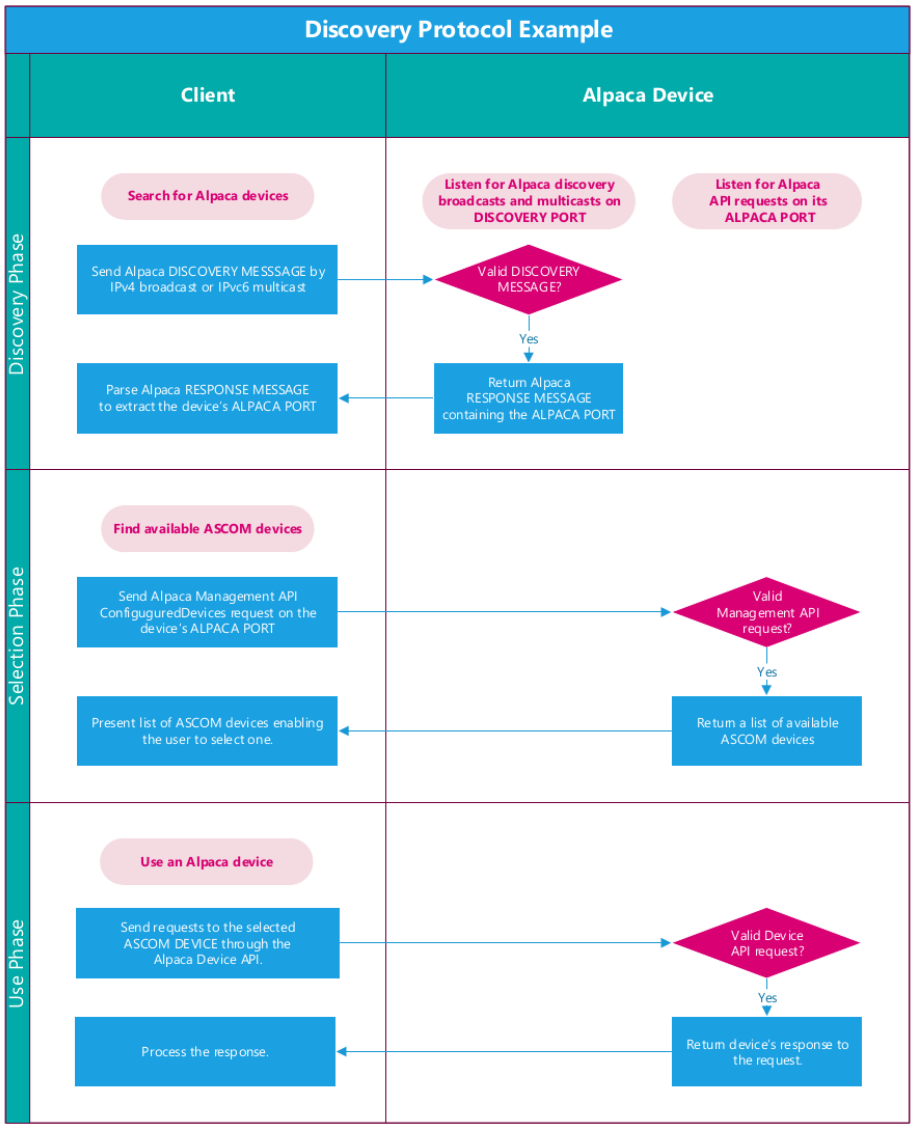

Figura 3.21: Documentación de la especificación ASCOM-Alpaca con el esquema del funcionamiento del servicio de descubrimiento automático. (Fuente: ASCOM ALPACA API Reference - Version 9[39])

Comprobación de funcionamiento de la interfaz Ascom

Para comprobar el funcionamiento de la implementación del driver del dispositivo Ascom, se modificó el código del módulo “dome-control” para simular el movimiento de la cúpula actualizando cada cierto tiempo la posición de azimut según los comandos de movimiento.

Para comprobar que la implementación cumple con la especificación Ascom se ha utilizado la aplicación de test oficial publicada por la iniciativa ASCOM: “ASCOM Universal Device Conformance Checker”[41].

En el Listado 3.11 se incluyen los resultados del test.

```
12:21:42.652 ASCOM Universal Device Conformance Checker Version 0.9.0.0, Build time: Vie. 18 marzo 2022 13:54:36
12:21:42.654
12:21:42.655 Alpaca device: DomeController (192.168.0.198:8080 Dome/0)
12:21:42.655
12:21:42.675 CreateDevice      INFO     Creating Alpaca device: IP address: 192.168.0.198, IP Port: 8080, Alpaca device number: 0
12:21:42.725 CreateDevice      INFO     Alpaca device created OK
12:21:42.725 CreateDevice      INFO     Successfully created driver
12:21:43.328 ConformanceCheck  OK       Driver instance created successfully
12:21:43.477 ConformanceCheck  OK       Connected OK
12:21:43.478
12:21:43.483 Common Driver Methods
12:21:43.554 InterfaceVersion  OK       2
12:21:43.557 Connected         OK       True
12:21:43.565 Description       OK       ALPACA compatible Dome Controller. Manufacturer: T6000
12:21:43.573 DriverInfo        OK       Dome Controller ALPACA protocol compliant.
                                        Manufacturer: T6000
12:21:43.580 DriverVersion     OK       0.1
12:21:43.588 Name              OK       DomeController
12:21:43.588 Action            INFO     Conform cannot test the Action method
12:21:43.599 SupportedActions  OK       Driver returned an empty action list
12:21:43.599
12:21:43.599 Can Properties
12:21:43.614 CanFindHome       OK       True
12:21:43.621 CanPark           OK       True
12:21:43.627 CanSetAltitude    OK       False
12:21:43.634 CanSetAzimuth     OK       True
12:21:43.640 CanSetPark        OK       True
12:21:43.646 CanSetShutter     OK       False
12:21:43.653 CanSlave          OK       True
12:21:43.659 CanSyncAzimuth    OK       False
12:21:43.659
12:21:43.659 Pre-run Checks
12:21:45.735 DomeSafety                 Open shutter check box is unchecked so shutter not opened
12:21:45.736
12:21:45.736 Properties
12:21:45.737 Altitude          INFO     You have configured Conform not to open the shutter so the following test may fail.
12:21:45.769 Altitude          OK       Optional member threw a PropertyNotImplementedException exception.
12:21:46.735 AtHome            OK       False
12:21:47.735 AtPark            OK       False
12:21:48.736 Azimuth           OK       0
12:21:49.736 ShutterStatus     OK       Closed
12:21:50.736 Slaved Read       OK       False
12:21:52.737 Slaved Write      OK       Slave state changed successfully
12:21:53.738 Slewing           OK       False
12:21:53.738
12:21:53.739 Methods
12:21:55.739 AbortSlew         OK       AbortSlew command issued successfully
12:21:55.747 SlewToAltitude    OK       Optional member threw a MethodNotImplementedException exception.
12:21:58.740 SlewToAzimuth 0   OK       Synchronous slew OK
12:22:09.743 SlewToAzimuth 0   OK       Reached the required azimuth: 0,0 within tolerance ±1 degrees. Reported azimuth: 0,0
12:22:56.766 SlewToAzimuth 45  OK       Asynchronous slew OK
12:23:07.769 SlewToAzimuth 45  OK       Reached the required azimuth: 45,0 within tolerance ±1 degrees. Reported azimuth: 45,0
12:23:54.792 SlewToAzimuth 90  OK       Asynchronous slew OK
12:24:05.794 SlewToAzimuth 90  OK       Reached the required azimuth: 90,0 within tolerance ±1 degrees. Reported azimuth: 90,0
12:24:52.818 SlewToAzimuth 135 OK       Asynchronous slew OK
12:25:03.821 SlewToAzimuth 135 OK       Reached the required azimuth: 135,0 within tolerance ±1 degrees. Reported azimuth: 135,0
12:25:50.845 SlewToAzimuth 180 OK       Asynchronous slew OK
12:26:01.847 SlewToAzimuth 180 OK       Reached the required azimuth: 180,0 within tolerance ±1 degrees. Reported azimuth: 180,0
12:26:48.871 SlewToAzimuth 225 OK       Asynchronous slew OK
12:26:59.874 SlewToAzimuth 225 OK       Reached the required azimuth: 225,0 within tolerance ±1 degrees. Reported azimuth: 225,0
12:27:46.897 SlewToAzimuth 270 OK       Asynchronous slew OK
12:27:57.900 SlewToAzimuth 270 OK       Reached the required azimuth: 270,0 within tolerance ±1 degrees. Reported azimuth: 270,0
12:28:44.923 SlewToAzimuth 315 OK       Asynchronous slew OK
12:28:55.926 SlewToAzimuth 315 OK       Reached the required azimuth: 315,0 within tolerance ±1 degrees. Reported azimuth: 315,0
12:28:56.928 SlewToAzimuth     OK       COM invalid value exception correctly raised for slew to -10 degrees
12:28:57.927 SlewToAzimuth     OK       COM invalid value exception correctly raised for slew to 370 degrees
12:28:57.935 SyncToAzimuth     OK       Optional member threw a MethodNotImplementedException exception.
12:28:57.943 CloseShutter      OK       Optional member threw a MethodNotImplementedException exception.
12:28:57.952 OpenShutter       OK       Optional member threw a MethodNotImplementedException exception.
12:29:55.954 FindHome          OK       Dome homed successfully
12:30:29.969 Park              OK       Dome parked successfully
12:30:40.972 SetPark           OK       SetPark issued OK
12:30:40.973
12:30:40.973 Post-run Checks
12:30:40.974 DomeSafety        INFO     Open shutter check box is unchecked so close shutter bypassed
12:30:40.975 DomeSafety        INFO     Attempting to park dome...
12:30:44.976 DomeSafety        OK       Dome successfully parked
12:30:44.976
12:30:44.980 Conformance test complete
12:30:44.980
12:30:44.981 No errors, warnings or issues found: your driver passes ASCOM validation!!
```

Listado 3.11: Resultados del test de prueba positivo del API REST ASCOM Alpaca para el dispositivo cúpula realizados con la herramienta de testeo “ASCOM Universal Device Conformance Checker”[41]

3.4.8 Servicio de monitorización de Beaglebone Black

El sistema de control de la cúpula está basado en una placa Beaglebone Black. Se ha implementado un servicio de monitorización llamado *bbbmonitor-mqtt.service* que se conecta como cliente MQTT al broker MQTT alojado en el servidor central. Este servicio se utiliza para monitorizar el estado de conectividad de la Beaglebone y el estado de los principales recursos del sistema operativo (carga de cpu, memoria libre, tiempo de funcionamiento, etc). Publica en dos topics siendo *NombreHost*, el nombre del host asignado a la Beaglebone:

- *NombreHost*/BBB-Data/stat: el servicio publica un “1” en este topic cuando establece una conexión con el broker MQTT del servidor central. Este topic se usa con la función de “last will” de MQTT con "0" como mensaje de “last will” y el flag de “retain” activado. De esta forma, si se pierde la conexión, el broker MQTT publica automáticamente un "0" en el topic. De esta manera, el estado de conectividad del servicio siempre está disponible en este topic: si su valor es "1", el servicio está en línea, lo que significa que la Beaglebone puede comunicarse con el broker MQTT, si su valor es "0", el servicio no está conectado, lo que significa que tal vez la Beaglebone esté apagada o no tenga conectividad con el servidor central.

- *NombreHost*/BBB-Data/data: cada minuto el servicio publica un mensaje en formato JSON con los campos indicados en el Listado 3.12.

```
{
  "AppString": "bbbmonitor-mqtt-pub",
  "Data_Vers": "002",
  "Host_Name": "BLK02-0055",
  "TimeStamp": 1650293380,
  "LoadAvrg5": 2,
  "DiskFreeP": 30,
  "Mem_FreeP": 82,
  "UpTimeSec": 5986068,
  "ethTxTotK": 1029599,
  "ethTxThro": 141,
  "ethRxTotK": 1371121,
  "ethRxThro": 198
}

- AppString: application identifier.
- Data_Vers: data structure version.
- Host_Name: Beaglebone host name.
- TimeStamp: message time stamp in epoch format.
- LoadAvrg5: Beaglebone cpu load last minute average in percentage.
- DiskFreeP: Beaglebone free disk space in percentage.
- Mem_FreeP: Beaglebone free ram memory space in percentage.
- UptimeSec: number of seconds elapsed since last system reboot.
- ethTxTotK: number of Kilobytes of data transmited through the eth0 network interface.
- ethTxThro: eth0 interface last minute average transmitted data throughput in bytes per second.
- ethRxTotK: number of Kilobytes of data received through the eth0 network interface.
- ethRxThro: eth0 interface last minute average received data throughput in bytes per second.
```

Listado 3.12: Ejemplo de mensaje JSON con la información de estado del sistema linux y explicación de los campos.

En el servidor central se implementa el servicio bbbmonitor-mqtt-graphite.service que se conecta al broker MQTT y se subscribe a estos topics para almacenar la información recibida en la base de datos Graphite.

Con estos datos se incluye en Grafana[24] un Dashboard (Figura 3.22) para monitorizar de forma visual la conectividad y estado de funcionamiento de todas las Beaglebones utilizadas en el sistema con la posibilidad de generar alertas y avisos en caso de que alguno de los parámetros supere los umbrales de funcionamiento adecuado.

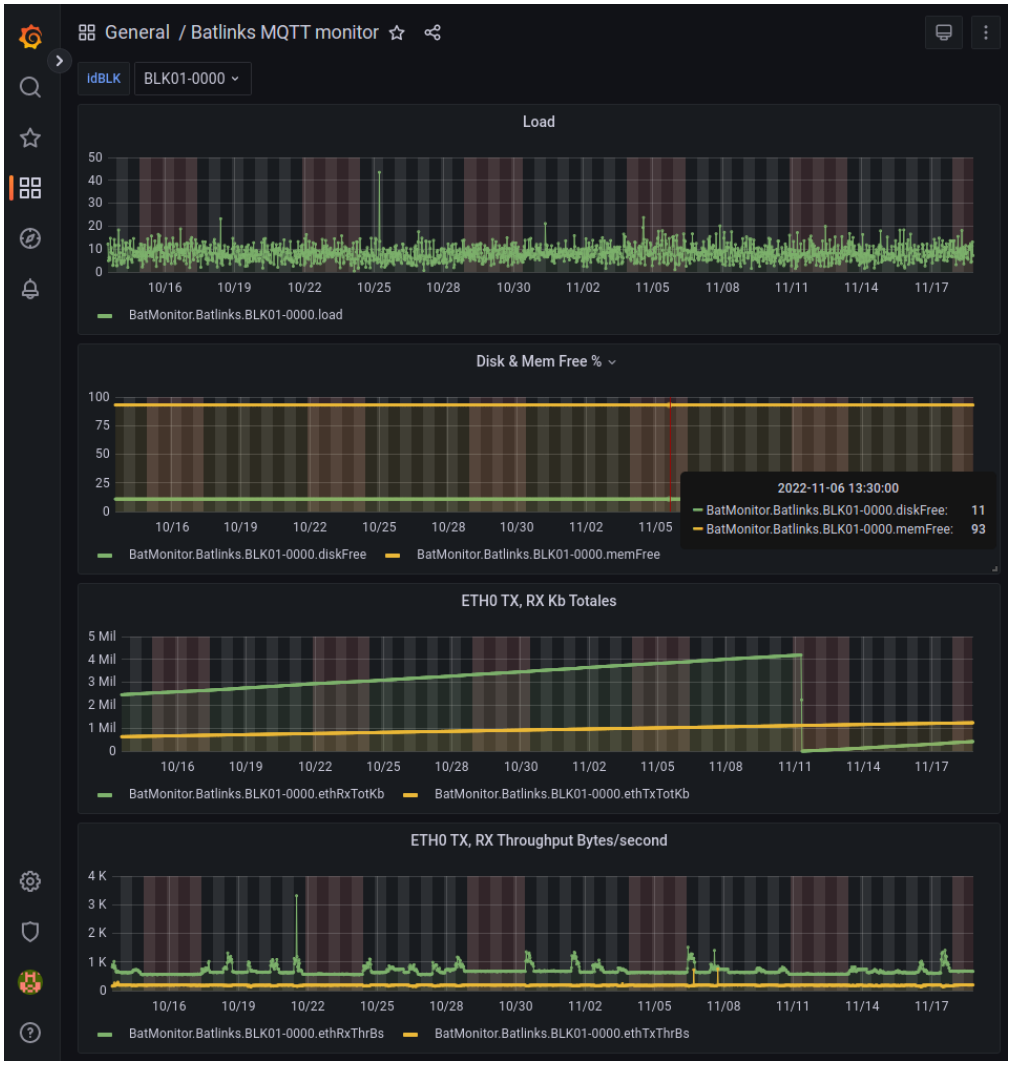

Figura 3.22: Ejemplo de dashboard de Grafana para visualizar el estado de funcionamiento de la placa Beaglebone Black. (Fuente: elaboración propia)

3.5 Servidor central

Las principales funciones del servidor central se pueden resumir en los siguientes puntos.

- Centralización de las comunicaciones de los diferentes dispositivos de control, entre ellos el controlador de cúpula. Con este propósito se ejecuta en el servidor un broker MQTT (Mosquitto[26]) que centraliza todas las comunicaciones realizadas mediante este protocolo.

- Almacenamiento y visualización de datos de monitorización de los diferentes sistemas del telescopio incluido el sistema de la cúpula. Para almacenar los datos de monitorización se utiliza la base de datos de series temporales Graphite DB[25], y como herramienta de visualización se utiliza el software Grafana[24].

- Albergar la aplicación web de control manual. Para servir la aplicación se utiliza el software de servidor web ligero Lighttpd[27].

Entre los motivos para centralizar estas funcionalidades en un servidor destacan la posibilidad de utilizar un hardware estandarizado y con gran capacidad de cómputo, la simplificación de las tareas de gestión de versiones, actualización y mantenimiento así como facilidad para realizar copias de seguridad de todo el sistema.

Además, el servidor se instala en una sala de control, aledaña a la sala del telescopio y la cúpula, que no está expuesta a la intemperie y dispone de unas mejores condiciones de temperatura, humedad y ruido electromagnético. En general en entornos industriales es conveniente que los dispositivos situados en la instalación de campo, expuestos a condiciones adversas de temperatura, humedad y ruido electromagnético, estén diseñados para realizar su función de la forma más sencilla posible para conseguir una mayor fiabilidad y minimizar las tareas de mantenimiento como posibles actualizaciones.

El equipo hardware que se utiliza como servidor es un PC de torre del fabricante HP con microprocesador Intel Core i7, 16 Gigabytes de memoria DRAM y un Terabyte de almacenamiento permanente en disco duro mecánico. Sin embargo, gracias a que el servidor se implementa como imagen de sistema en el entorno de virtualización basado en contenedores LXD[42], la migración del sistema a otro hardware con la misma arquitectura de procesador (x86_64) es muy sencilla. El sistema operativo que se utiliza en el host es Linux Ubuntu 22.04.5 LTS[43], el mismo que se utiliza en el sistema virtualizado.

Para poder acceder remotamente tanto al servidor central como a los dispositivos de control basados en Linux, se utiliza el servicio de VPN ligera Tailscale.

Tailscale[44] es un servicio de VPN de malla (mesh VPN) que permite crear una red privada segura entre dispositivos distribuidos, sin necesidad de configurar manualmente puertos,

firewalls o direcciones IP públicas. Utiliza el protocolo WireGuard[45] para el cifrado punto a punto, y gestiona automáticamente la autenticación, la asignación de direcciones IP privadas y el descubrimiento entre nodos. Está diseñado para ser fácil de usar, con clientes para Linux, Windows, macOS, iOS, Android y dispositivos embebidos como Raspberry Pi o Beaglebone Black. Con Tailscale, todos los dispositivos conectados se comportan como si estuvieran en la misma red local, permitiendo el acceso remoto seguro a servicios internos, desarrollo distribuido, sincronización de archivos, etc.

La posibilidad de un acceso remoto de forma segura al sistema es de gran importancia pues permite realizar remotamente tareas de soporte, mantenimiento, actualización, configuración, corrección de posibles errores de software y firmware e incluso desarrollo y realización de pruebas en coordinación con el personal técnico del observatorio. Gracias a esto, en muchas ocasiones se pueden evitar desplazamientos no imprescindibles facilitando mucho el desarrollo del proyecto.

Hay que tener en cuenta que al tratarse de una instalación de alta montaña, en determinadas épocas del año el acceso al observatorio es complicado. En temporada de ski, el acceso se realiza en telecabina desde el pueblo de Pradollano, al que se llega en coche, para luego subir hasta el observatorio en moto de nieve. En verano, cuando no hay nieve, el acceso se realiza en vehículo 4x4. Las temporadas en las que es más difícil llegar al observatorio son en primavera y otoño, cuando la estación de ski está cerrada pero todavía hay o empieza a haber nieve suficiente como para impedir el acceso en 4x4.

3.5.1 Virtualización con LXD

El uso de un sistema de virtualización en un servidor ofrece una serie de ventajas significativas. Una de ellas es la flexibilidad y escalabilidad que proporciona. Las máquinas virtuales pueden crearse, modificarse, duplicarse o eliminarse con mucha facilidad, lo que permite adaptar el entorno rápidamente a nuevas necesidades, realizar pruebas sin comprometer el sistema en producción o replicar configuraciones para tareas de mantenimiento, desarrollo o respaldo.

En términos de mantenimiento y fiabilidad, la virtualización también simplifica considerablemente la gestión. Las actualizaciones, migraciones y copias de seguridad pueden realizarse de manera más segura y sin afectar directamente al hardware físico.

En este proyecto se utiliza el sistema de virtualización basado en contenedores LXD[42]. La virtualización basada en contenedores, a diferencia de la basada en hipervisores (como VMware ESXi, Microsoft Hyper-V o KVM), no emula sistemas operativos completos, sino que utiliza el mismo núcleo (kernel) del sistema anfitrión para ejecutar múltiples entornos aislados. Cada

contenedor empaqueta solo las aplicaciones y sus dependencias, lo que los hace mucho más ligeros y rápidos de arrancar en comparación con una máquina virtual. Al compartir el mismo sistema operativo base, los contenedores consumen menos memoria y almacenamiento, permitiendo ejecutar muchas más instancias con los mismos recursos físicos.

Las principales ventajas de los sistemas basados en contenedores incluyen su eficiencia en el uso de recursos, su rapidez de despliegue, y su gran portabilidad, ya que los contenedores pueden ejecutarse en cualquier entorno que soporte el motor de contenedores, garantizando que funcionarán de forma idéntica independientemente de la plataforma.

LXD es un entorno de virtualización basado en contenedores, diseñado para ofrecer una experiencia similar a la de las máquinas virtuales tradicionales, pero con la eficiencia y rapidez de los contenedores. Fue desarrollado inicialmente por Canonical, la empresa detrás de Ubuntu, y está orientado a ejecutar contenedores de tipo sistema completo, es decir, entornos que simulan una máquina completa con su propio sistema operativo de usuario (por ejemplo, Ubuntu, CentOS, etc.), pero compartiendo el núcleo del sistema anfitrión.

LXD funciona sobre la tecnología LXC (Linux Containers), que proporciona el aislamiento a nivel de sistema operativo. Sin embargo, LXD añade una capa de gestión más avanzada y fácil de usar, incluyendo una API REST, herramientas de línea de comandos, gestión remota, y funciones como instantáneas, copias de seguridad, redes definidas por el usuario y almacenamiento flexible. Con LXD, los contenedores se comportan más como máquinas virtuales ligeras que como simples entornos para ejecutar aplicaciones.

Aunque las herramientas de línea de comandos de LXD son muy potentes y flexibles, también ofrece una aplicación web como interfaz gráfica de usuario que facilita su uso. En este proyecto se utiliza una imagen de LXD con sistema operativo Linux Ubuntu 22.04.5 LTS. En la Figura 3.23 se muestra una captura de pantalla de la interfaz gráfica de LXD en la que se puede ver las características principales de la imagen utilizada.

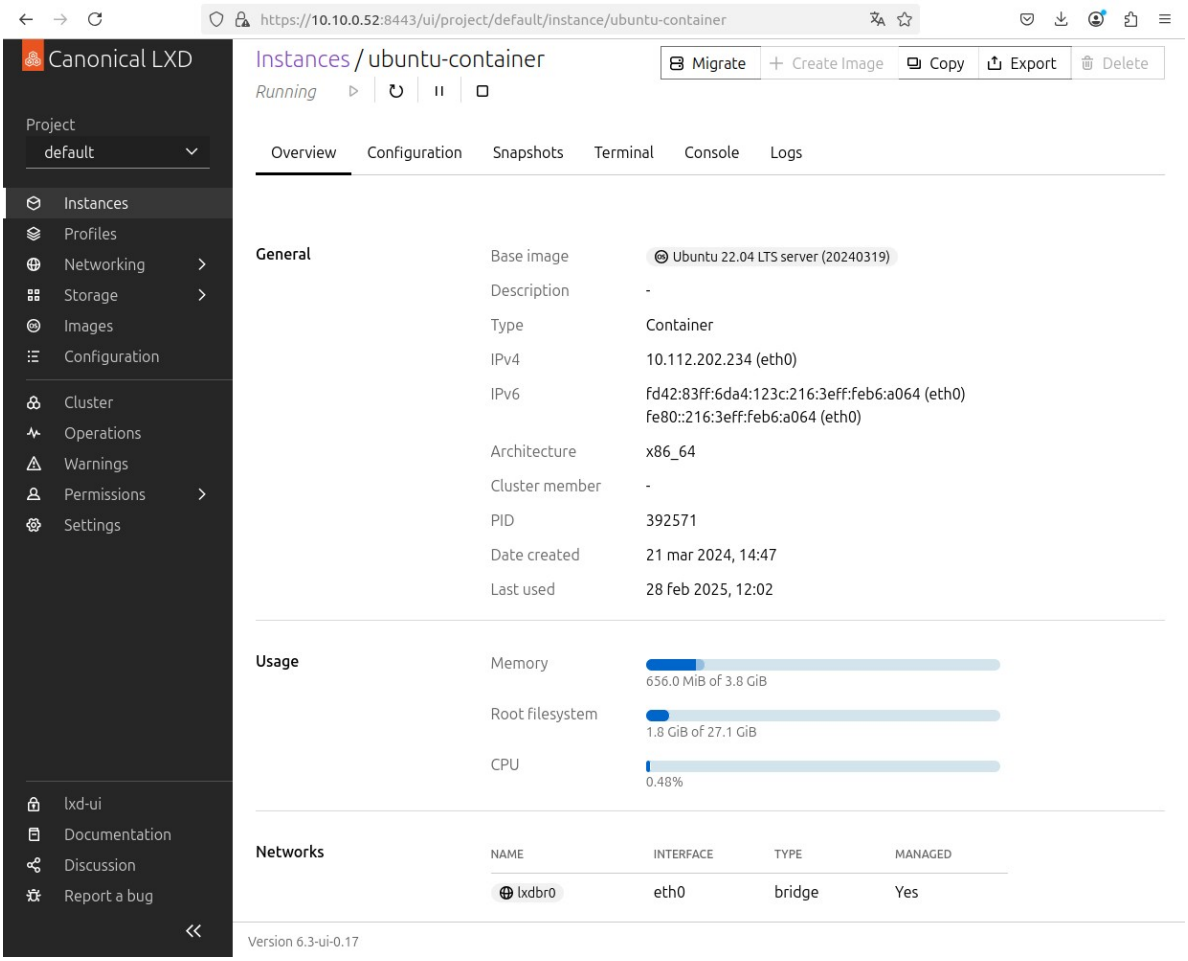

Figura 3.23: Interfaz gráfica del entorno de virtualización LXD. (Fuente: elaboración propia)

3.5.2 Broker MQTT

En las comunicaciones por MQTT[3] un broker actúa como intermediario entre los dispositivos (clientes) que publican y suscriben mensajes. En lugar de que los dispositivos se comuniquen directamente entre sí, todos envían sus mensajes al broker, que luego los redistribuye a los clientes que están suscritos a esos temas o canales específicos. Esta arquitectura publish/subscribe (publicar/suscribirse) permite una comunicación desacoplada, escalable y eficiente.

El broker se encarga de gestionar las conexiones, almacenar los mensajes si es necesario (según la calidad del servicio definida), distribuir los mensajes solo a quienes los soliciten y mantener el control sobre la persistencia y seguridad de los datos.

En este proyecto se utiliza Mosquitto[26], un broker MQTT de código abierto, ligero y eficiente. Es uno de los brokers más populares y utilizados en proyectos de Internet de las Cosas

(IoT) debido a su sencillez, fiabilidad y bajo consumo de recursos. Está desarrollado y mantenido por la Eclipse Foundation, lo que garantiza un desarrollo activo y una comunidad sólida.

Entre sus características principales, Mosquitto ofrece compatibilidad total con las versiones MQTT 3.1, 3.1.1 y 5.0, lo que permite interoperar con una amplia variedad de clientes y dispositivos. Soporta múltiples niveles de calidad de servicio (QoS), permite la persistencia de mensajes, autenticación y autorización mediante archivos o integraciones con bases de datos y sistemas externos (como LDAP), y es capaz de funcionar tanto en sistemas pequeños como en servidores más potentes.

Mosquitto destaca por su ligereza y eficiencia, lo que lo hace ideal para dispositivos embebidos o entornos con recursos limitados. Su facilidad de configuración, el bajo consumo de CPU y memoria, y su capacidad para escalar desde pequeñas redes locales hasta infraestructuras distribuidas hacen que sea una opción muy versátil.

Mosquitto también permite utilizar el protocolo MQTT sobre conexiones con websockets. Este tipo de comunicación se usa en el proyecto para implementar las comunicaciones por MQTT en la aplicación web de control manual.

El acceso al broker se ha configurado con autenticación por nombre de usuario y contraseña y se han habilitado dos interfaces de acceso, una a través del puerto 1883 para conexiones con TCP, y otra a través del puerto 9001 para conexiones con websockets. Aunque se puede configurar Mosquitto para usar el protocolo de encriptación TLS, actualmente no se está utilizando este tipo de encriptación pues el acceso al broker no está abierto al exterior y todo el tráfico MQTT se realiza dentro de la red de área local del observatorio.

En el apartado 3.4.6 se explica la estructura de topics utilizada en el proyecto para el controlador de la cúpula con cada topic detallado en la Tabla 3.7.

En la Figura 3.24 se incluye una captura de pantalla de la herramienta MQTT Explorer[46] conectada al broker Mosquitto del servidor central donde se puede ver, en la parte izquierda de la imagen la estructura de topics del sistema, y en la parte derecha los datos recibidos en el topic de azimut.

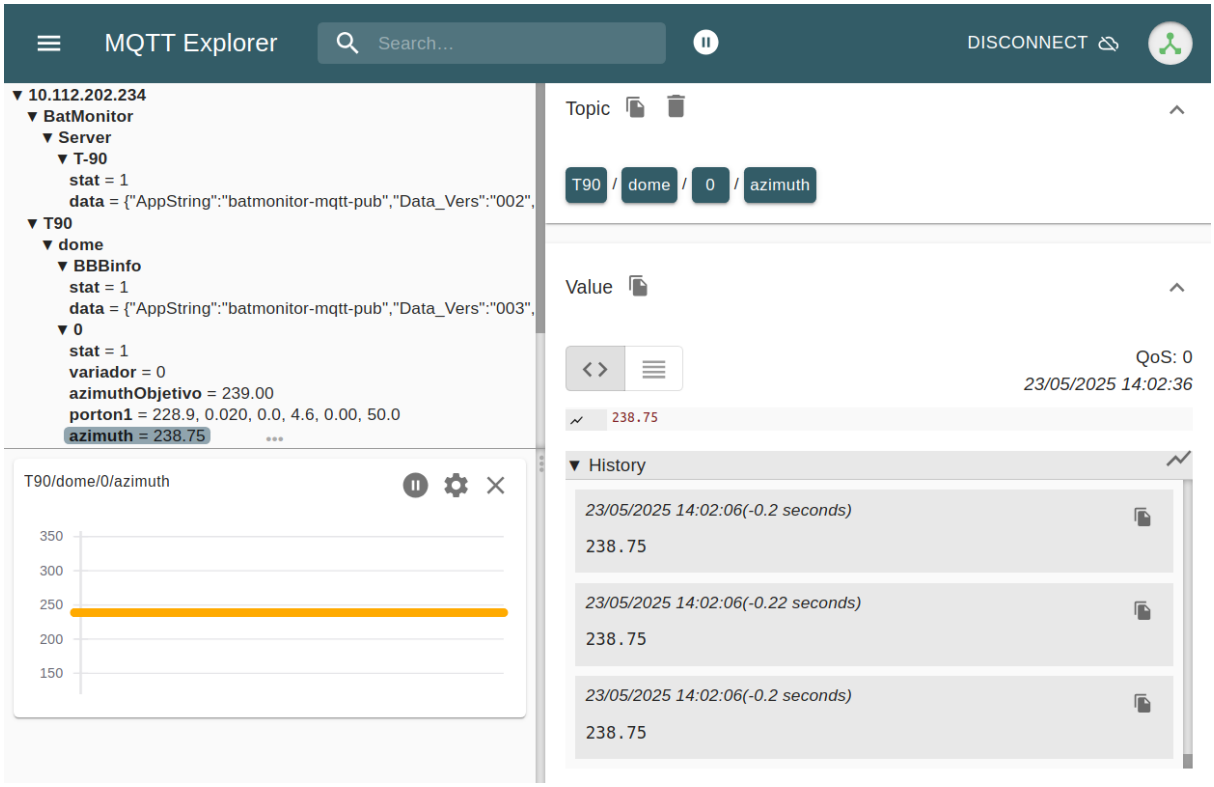

Figura 3.24: Captura de pantalla de la herramienta MQTT Explorer conectada al broker Mosquitto del servidor central. (Fuente: elaboración propia)

3.5.3 Almacenamiento y visualización de datos

Uno de los requisitos del proyecto es la toma de datos de monitorización y visualización de diversos parámetros de funcionamiento de los sistemas del observatorio incluido el de la cúpula.

Almacenamiento de datos

Para almacenamiento de datos históricos se utiliza la base de datos de series temporales Graphite DB[25].

Graphite es una herramienta de software diseñada para recopilar, almacenar y visualizar datos de series temporales, es decir, datos que varían a lo largo del tiempo y que suelen provenir de sistemas de monitorización, métricas de rendimiento, sensores o registros de actividad en sistemas informáticos y de red. Se utiliza principalmente para monitorizar infraestructuras TI, aplicaciones, servidores y redes, aunque también puede aplicarse en contextos científicos e industriales donde se requiere el seguimiento continuo de valores numéricos.

Graphite funciona mediante tres componentes principales:

- Carbon, que es el servicio que recibe las métricas enviadas por otros sistemas mediante el protocolo plaintext (o pickle) y las almacena.

- Whisper, que es la base de datos de series temporales propia de Graphite. Almacena los datos en archivos circulares optimizados para consultas rápidas, y permite consolidar los datos antiguos para ahorrar espacio.

- Graphite Web, que proporciona una interfaz web para consultar, visualizar y crear gráficos de las métricas almacenadas, con capacidades de filtrado, agregación y análisis. Este componente también proporciona un API REST para la consulta y recuperación de datos almacenados en Whisper lo que permite utilizar otras herramientas de visualización como Grafana que es compatible con esta base de datos y es la herramienta de visualización que se utiliza en este proyecto.

La herramienta está diseñada para ser altamente eficiente y escalable, capaz de manejar grandes volúmenes de datos con una baja latencia de acceso.

En Graphite DB, y en otras bases de datos de series temporales, la únidad básica de almacenamiento se denomina métrica y consiste en una serie de valores numéricos que se miden periódicamente almacenándose cada medida acompañada con una marca de tiempo o timestamp que indica el instante temporal en el que se ha recibido la medida. Cada métrica se corresponde con un único parámetro del sistema que se está monitorizando y cuya magnitud varía con el tiempo.

Cada métrica de Graphite se almacena en un único archivo cuya principal característica es que tiene un tamaño fijo. Esto se debe a que los datos se almacenan en forma de buffer circular. Es decir, cuando el archivo se llena, los nuevos datos se guardan desde el principio del archivo sobreescribiendo los datos más antiguos. La periodicidad de guardado de datos y el tiempo durante el que almacenan se puede configurar para todas las métricas recibidas o por grupos de métricas organizados según el nivel que ocupan en la jerarquía. Por ejemplo, se podría configurar Graphite para que todas las métricas por debajo de los niveles “Sistema1.datos-servidor” se guardaran con una periodicidad de un minuto durante un año, mientras que las métricas por debajo de los niveles “Sistema1.datos-sensores” se guardaran con una periodicidad de un segundo durante un mes.

Además, para optimizar el espacio de almacenamiento, otra de las características de las métricas es que se pueden definir diferentes resoluciones temporales de guardado de datos para un mismo archivo. Por ejemplo, se podría configurar cierto grupo de métricas para que se almacenaran con una periodicidad de un segundo durante un día, un minuto durante un mes, y una hora durante un año. Cada vez que se pasa de una resolución temporal más alta a una más

baja, los datos para la resolución baja se calculan mediante una función de agregación que suele ser una media aritmética aunque se puede configurar. En el ejemplo anterior, los datos que se guardan con resolución temporal de un minuto se obtendrían calculando la media aritmética de los sesenta segundos correspondientes a cada minuto. De esta forma, se puede configurar un esquema de almacenamiento donde los datos recientes tengan una alta resolución temporal mientras que los más antiguos van perdiendo resolución, optimizando así el espacio de almacenamiento. Además, conociendo el esquema de almacenamiento de cada métrica se puede saber de antemano el tamaño que ocupará el archivo donde se almacena.

En este proyecto el esquema de almacenamiento para las métricas del sistema se ha configurado para guardar los datos con una periodicidad de un segundo durante un día, un minuto durante treinta días, cinco minutos durante noventa días, treinta minutos durante ciento ochenta días y una hora durante un año. En la Tabla 3.11 se resume este esquema de almacenamiento.

Tabla 3.11: Esquema de almacenamiento de métricas configurado en Graphite DB

Periodicidad de almacenamiento en segundos	**Duración de almacenamiento en días**
1	1
60	30
300	90
1800	180
3600	365

Las métricas en Graphite DB se estructuran de manera jerárquica con un sistema de nombrado que separa cada nivel de la jerarquía con un punto. Esta forma de organización de la información es similar a la que se utiliza en el protocolo MQTT para estructurar los topics de mensajes. Esta similitud se puede aprovechar para utilizar la misma estructura jerárquica utilizada en las comunicaciones por MQTT en la organización de las métricas de Graphite consiguiendo una organización sencilla y clara.

Por ejemplo, si el topic en el que se comunica el azimut de la cúpula es **/T90/dome/0/azimuth**, simplemente sustituyendo las barras por puntos y añadiendo al principio la raiz “Observatorio”, obtenemos la métrica **Observatorio.T90.dome.0.azimuth**, donde se guarda la evolución de la posición de azimut de la cúpula.

Para los topics en los que se publica más de un parámetro en formato de lista de valores separados por comas, el nombre de la métrica de cada parámetro se compone de igual forma que en el ejemplo anterior, añadiendo al final de la métrica el nombre correspondiente a cada parámetro. Por ejemplo, en el topic **/T90/dome/0/motor**, correspondiente a los datos de funcionamiento de los motores proporcionados por el variador, se publican varios parámetros, como la tensión, corriente y potencia consumida, en una lista de valores separados por comas. Las métricas en este caso son **Observatorio.T90.dome.0.motor.voltage**, **Observatorio.T90.dome.0.motor.current**, **Observatorio.T90.dome.0.motor.power**, etc.

Recopilación de datos

Los datos que se monitorizan en el proyecto se pueden clasificar en tres áreas principales según la parte del sistema del que provienen.

Por un lado se monitorizan los principales datos de funcionamiento del servidor central para detectar posibles problemas como exceso de carga de CPU, saturación de memoria, congestión de comunicaciones, etc. Las métricas con los datos de funcionamiento del servidor monitorizados se detallan en la Tabla 3.12.

Tabla 3.12: Parámetros monitorizados del servidor

Nombre de métrica	Descripción
BatMonitor.Server.T90.load	carga media del procesador durante el último minuto, en porcentaje.
BatMonitor.Server.T90.diskFree	espacio libre en disco, en porcentaje.
BatMonitor.Server.T90.memFree	espacio libre de memoria RAM, en porcentaje.
BatMonitor.Server.T90.upTime	número de segundos transcurridos desde el último reinicio del sistema.
BatMonitor.Server.T90.ethRxTotKb	número de Kb de datos recibidos a través de la interfaz de red.
BatMonitor.Server.T90.ethTxTotKb	número de Kb de datos transmitidos a través de la interfaz de red.
BatMonitor.Server.T90.ethRxThrBs	promedio de velocidad de recepción de datos durante el último minuto, en bytes por segundo.
BatMonitor.Server.T90.ethTxThrBs	promedio de velocidad de transmisión de datos durante el último minuto, en bytes por segundo.

De forma similar a los datos del servidor, y tal y como se explica en el apartado 3.4.8, también se monitorizan los datos de los dispositivos de control basados en sistema operativo Linux. En este caso, se monitorizan los datos del sistema Linux que se ejecuta en la SBC Beaglebone Black del dispositivo de control de la cúpula. Los datos monitorizados son los mismos que los del servidor pero en este caso corresponden al sistema de la Beaglebone Black. En la Tabla 3.13 se detallan las métricas utilizadas.

Tabla 3.13: Parámetros monitorizados del sistema Linux del dispositivo de control de cúpula

Nombre de métrica	Descripción
Observatorio.T90.dome.BBBinfo.BLK02-0012.load	carga media del procesador durante el último minuto, en porcentaje.
Observatorio.T90.dome.BBBinfo.BLK02-0012.diskFree	espacio libre en disco, en porcentaje.
Observatorio.T90.dome.BBBinfo.BLK02-0012.memFree	espacio libre de memoria RAM, en porcentaje.

Nombre de métrica	Descripción
Observatorio.T90.dome.BBBinfo.BLK02-0012.upTime	número de segundos transcurridos desde el último reinicio del sistema.
Observatorio.T90.dome.BBBinfo.BLK02-0012.ethRxTotKb	número de Kb de datos recibidos a través de la interfaz de red.
Observatorio.T90.dome.BBBinfo.BLK02-0012.ethTxTotKb	número de Kb de datos transmitidos a través de la interfaz de red.
Observatorio.T90.dome.BBBinfo.BLK02-0012.ethRxThrBs	promedio de velocidad de recepción de datos durante el último minuto, en bytes por segundo.
Observatorio.T90.dome.BBBinfo.BLK02-0012.ethTxThrBs	promedio de velocidad de transmisión de datos durante el último minuto, en bytes por segundo.

Para guardar los datos publicados en MQTT en las métricas correspondientes de Graphite DB, tanto en el caso de los datos del servidor como en el de los de la Beaglebone, se utiliza la herramienta denominada “monitorbatnet-mqtt-graphite”. Esta herramienta, desarrollada anteriormente en lenguaje C por el autor, se ha reutilizado en este proyecto y se ejecuta en el servidor como un servicio de Linux.

Por último, se monitorizan los datos propios de funcionamiento del sistema que se está controlando. En este caso los datos de funcionamiento del sistema de la cúpula. En la Tabla 3.14 se detallan los datos monitorizados del sistema de la cúpula y las métricas donde se guardan.

Tabla 3.14: Parámetros monitorizados del sistema Linux del dispositivo de control de cúpula

Nombre de métrica	Descripción
Observatorio.T90.dome.0.stat	a 1: el controlador de cúpula está funcionando
Observatorio.T90.dome.0.variador	a 1: hay conexion con el variador
Observatorio.T90.dome.0.azimuth	azimut actual de la cúpula
Observatorio.T90.dome.0.azimuthObjetivo	Objetivo de posición de azimut
Datos de los motores proporcionados por el variador de frecuencia	
Observatorio.T90.dome.0.motor.current	Corriente RMS proporcionada al motor en décimas de amperio
Observatorio.T90.dome.0.motor.voltage	Tensión RMS proporcionada al motor en décimas de voltio
Observatorio.T90.dome.0.motor.power	Potencia activa proporcionada al motor en centésimas de vatio.
Observatorio.T90.dome.0.motor.temperature	Temperatura medida en el motor.
Observatorio.T90.dome.0.motor.stop	a 1: el motor está parado
Observatorio.T90.dome.0.motor.fw	a 1: el motor se mueve hacia adelante

Nombre de métrica	Descripción
Observatorio.T90.dome.0.motor.bw	a 1: el motor se mueve hacia atrás
Observatorio.T90.dome.0.motor.error	a 1: el variador está en modo fallo
Observatorio.T90.dome.0.motor.failCode	código de error del variador
Observatorio.T90.dome.0.motor.freq	Frecuencia de la señal trifásica proporcionada al motor
Observatorio.T90.dome.0.motor.rpm	Velocidad de giro del motor

En el caso de los datos de la cúpula, para guardar los datos que se publican por MQTT en las métricas correspondientes de Graphite, se utiliza una herramienta desarrollada en lenguaje C a la que se ha llamado **mqtt2graphite**. Esta herramienta, de carácter genérico, aprovecha la forma de nombrar las métricas explicada en el apartado anterior para suscribirse por MQTT a los topics indicados en línea de comando y guardar los datos recibidos en las métricas correspondientes.

En los topics donde se publica más de un parámetro, en formato de lista de valores separados por comas, la herramienta parsea la lista de valores y genera cada métrica a partir del nombre del topic añadiendo al final el nombre de cada parámetro que se debe indicar en la línea de comandos al llamar al programa.

En el Listado 3.13 se muestra el formato de llamada al programa por línea de comandos y las opciones de línea de comandos.

```
Formato de llamada: mqtt2graphite [OPCIONES]
    OPCIONES:
        -v                // Activa trazas de depuracion.
        -V                // Muestra versión y termina.
        -p nº puerto      // Nº de puerto del broker MQTT. Por Defecto: 1883
        -S direccion MQTT // Dirección IP o host del broker MQTT. Por defecto: 127.0.0.1
        -U usuario MQTT   // Nombre de usuario para autenticar en broker MQTT. Si no se incluye,
                          // sin usuario (por defecto)
        -P passwd MQTT    // Password para autenticar en broker MQTT. Si no se incluye, sin
                          // password (por defecto)
        -T topic MQTT     // Topic al que se subscribe en el broker MQTT. Por defecto: #
        -I ClientID MQTT  // Client ID para la conexión con el broker MQTT. Si no se incluye se
                          // genera uno automaticamente (por defecto)
        -k MQTT keep alive// Intervalo en segudnos entre Keeps Alive de MQTT. Entre 1 y 7200 Por
                          // defecto: 60
        -C constant       // Se ignora el contenido del mensaje, siempre se guarda el valor
                          // constant (nº entero) en la métrica de graphite
        -M contenido      // Si está activada la opción '-C constant', la constante sólo se guarda
                          // si el contenido del mensaje coincide con el parámetro indicado.
        -L n1,n2,-,n3,... // Parsea el mensaje como una lista de valores separados por coma, a
                          // cada valor se le asigna el final de métrica indicado (n1,n2,n3).
                          // Los campos para los que no se indica nombre (con un guion entre dos
                          // comas: ,-,) se ignoran.
        -G dir Graphite   // Dirección IP del servidor de graphite. Por defecto: 127.0.0.1
        -R path Graphite  // Inicio de path para la métrica de graphite
        -E fin metrica    // Final de métrica a añadir al path de graphite (debe empezar por '.')
                          // en el caso de utilizar '-C constant' o value (por defecto))
```

Listado 3.13: Formato de llamada y opciones de línea de comandos para la herramienta mqtt2graphite.

La herramienta mqtt2graphite, desarrollada anteriormente por el autor, se ha reutilizado en este proyecto y se ejecuta en el servidor como un servicio de Linux.

Visualización de datos

Para visualizar los datos guardados en Graphite DB se utiliza la herramienta de visualización Grafana.

Grafana[24] es una plataforma de software de código abierto diseñada para la visualización y análisis de datos en tiempo real. Se utiliza principalmente para crear paneles interactivos (dashboards) que representan visualmente datos provenientes de múltiples fuentes, como bases de datos, servicios web, sistemas de monitoreo o aplicaciones industriales.

Grafana funciona como una interfaz web que se conecta a fuentes de datos externas, como InfluxDB, Prometheus, MySQL, PostgreSQL, Elasticsearch, Graphite DB o incluso archivos CSV, entre muchas otras. A través de consultas configurables, obtiene los datos y permite presentarlos en gráficos, tablas, medidores y otras visualizaciones personalizadas. También admite alertas basadas en condiciones definidas por el usuario, que pueden notificarse por correo electrónico, Slack, Telegram, entre otros canales.

Se utiliza ampliamente en monitorización de sistemas y redes, observabilidad de infraestructuras, IoT, análisis industrial, aplicaciones científicas y todo tipo de entornos donde es importante entender grandes volúmenes de datos mediante visualizaciones claras y actualizadas en tiempo real. Su interfaz intuitiva, flexibilidad para integrar múltiples fuentes y capacidad para personalizar dashboards lo convierten en una herramienta muy potente tanto en entornos profesionales como académicos.

En este proyecto en principio se han creado tres dashboards para presentar los datos de cada una de las áreas explicadas en el apartado anterior. En la Figura 3.25 se muestra una captura de pantalla del dashboard correspondiente a los datos de monitorización del servidor central donde se puede observar la evolución en el tiempo de los parámetros de monitorización del servidor especificados en el apartado anterior.

Figura 3.25: Dashboard de Grafana para visualizar los datos de monitorización del estado del servidor. (Fuente: elaboración propia)

En la Figura 3.26 se muestra una captura de pantalla del dashboard correspondiente a los datos de monitorización del sistema Linux del dispositivo de control de cúpula donde se puede observar la evolución en el tiempo del estado de funcionamiento del dispositivo.

Figura 3.26: Dashboard de Grafana para visualizar los datos de monitorización del estado del dispositivo de control de cúpula. (Fuente: elaboración propia)

En la Figura 3.27 se muestra una captura de pantalla del dashboard correspondiente a los datos de monitorización del sistema de la cúpula.

En la columna de la izquierda, en la parte superior, se ve un panel con los datos del azimut de la cúpula en línea verde así como el azimut objetivo en línea amarilla punteada. Inmediatamente debajo, se aprecian dos paneles indicando el estado de activación del controlador de la cúpula y el estado de comunicación con el variador respectivamente. Justo debajo se ve otro panel donde se muestra el estado de funcionamiento del variador, indicando si los motores están parados o en movimiento hacia adelante o atrás o si está en estado de fallo y en ese caso el código de error.

En la columna central se ven tres paneles mostrando los parámetros de funcionamiento de los motores: tensión y corriente eléctrica, potencia y temperatura, y frecuencia y velocidad de giro. Por último, en la columna de la derecha se ven los parámetros eléctricos del motor que abre y cierra el portón de la cúpula: tensión, corriente y potencia. Estos parámetros se están midiendo con un dispositivo comercial de medida de consumo en cuadro y por ello no se han especificado antes en este proyecto.

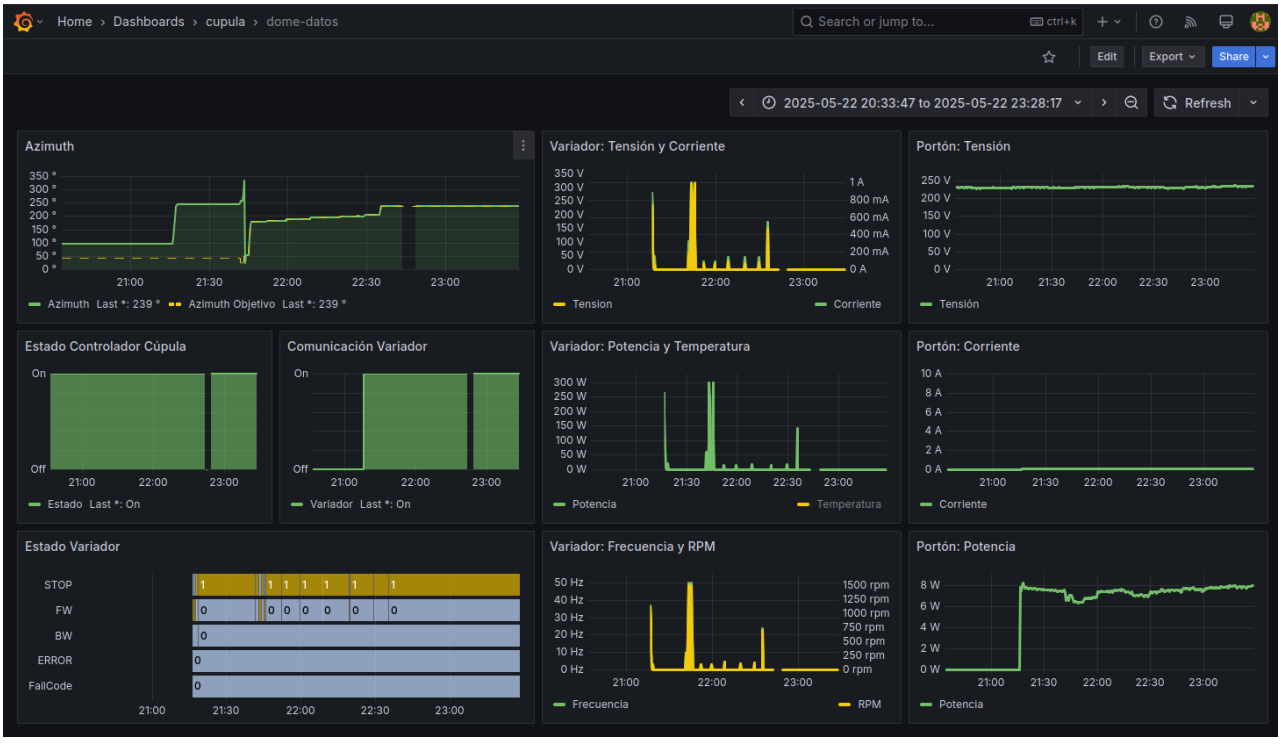

Figura 3.27: Dashboard de Grafana para visualizar los datos de monitorización del sistema de control de cúpula. (Fuente: elaboración propia)

Gracias a la facilidad de creación y edición de dashboards que ofrece Grafana, estos dashboards se pueden modificar o añadir otros nuevos de forma muy sencilla según las preferencias y necesidades de los usuarios finales del sistema.

3.5.4 Aplicación web de control

Además de poder operar la cúpula utilizando una aplicación de observación astronómica compatible con el protocolo ASCOM Alpaca, el proyecto también requiere la posibilidad de controlar los sistemas a través de un centro de control centralizado.

Con este objetivo se ha desarrollado una aplicación web utilizando HTML, CSS y el lenguaje de programación web javascript. Para facilitar el diseño de los componentes de la aplicación se utiliza el framework de CSS Bootstrap[47]. También se utiliza la librería de javascript jQuery[48].

La aplicación se comunica con el dispositivo de control utilizando el API MQTT implementado en el dispositivo, conectándose como cliente al broker MQTT mosquito a través del interfaz de websockets. Para implementar MQTT se utiliza la librería javascript de cliente MQTT del proyecto Paho desarrollado como código abierto por la fundación Eclipse[49].

En la Figura 3.28 se muestra una captura de pantalla de la pantalla principal de la aplicación.

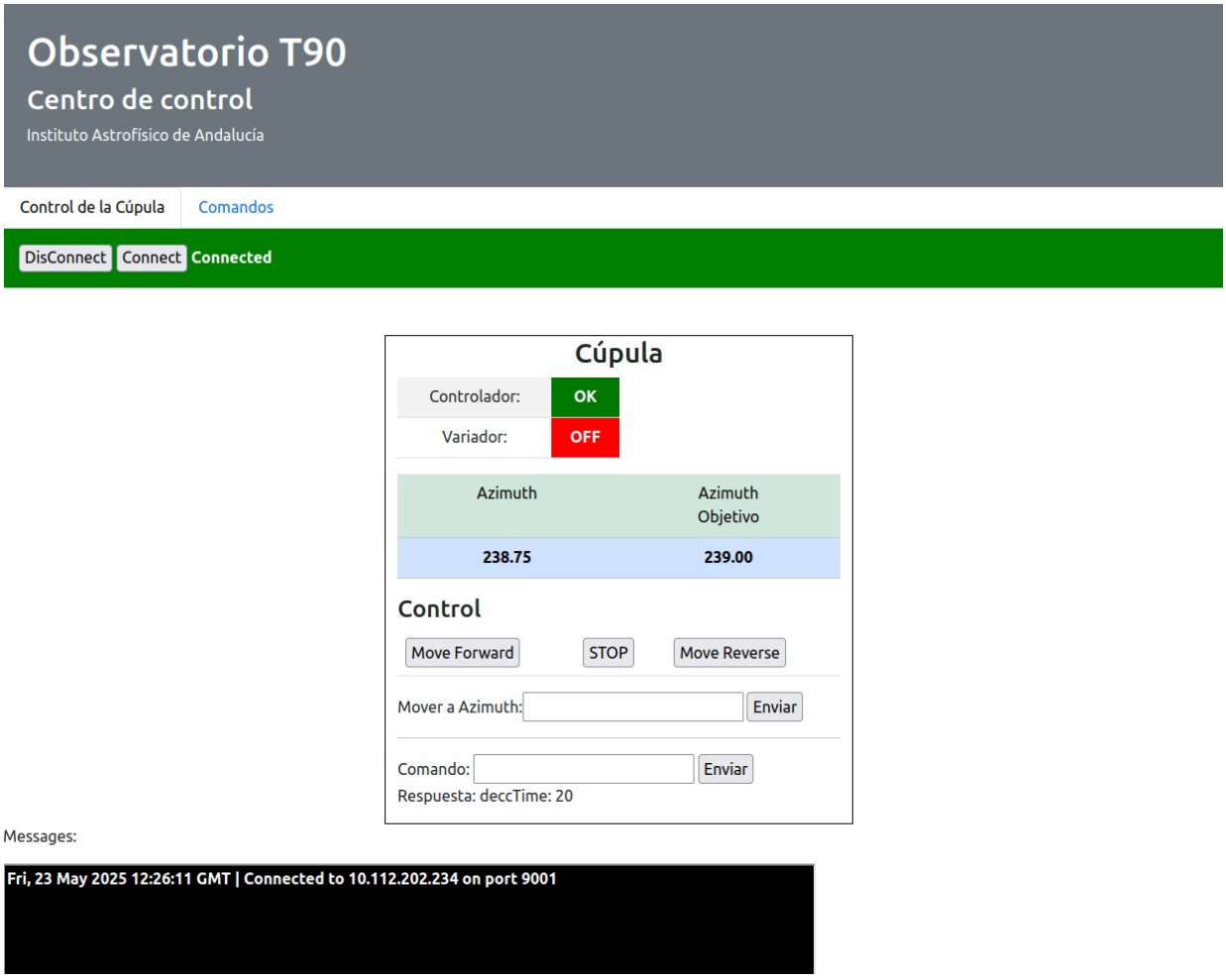

Figura 3.28: Pantalla principal de la aplicación web de control de la cúpula. (Fuente: elaboración propia)

A continuación se explican los contenidos de la aplicación empezando por la parte superior y en sentido descendente.

En primer lugar se encuentra la franja de título. Justo debajo aparece el menú para pasar de la pantalla principal a la de comandos o viceversa.

Luego hay una franja de color que se utiliza para conectar o desconectar la aplicación al broker MQTT utilizando los botones Connect y Disconnect respectivamente. Además, si la aplicación está conectada al broker se indica con el texto Connected y el color de fondo de la franja en verde, mientras que si no está conectada, aparece el texto Disconnected con el color de la franja en rojo. Como las comunicaciones con el dispositivo de control se realizan a través de MQTT, para que la aplicación se pueda utilizar, esta franja debe aparecer con el color de fondo en verde. En caso contrario, todos los datos y estados mostrados en la aplicación no son válidos.

Más abajo está el cuadro de control de la cúpula. Comienza con dos indicadores que muestran el estado de funcionamiento del controlador y la conectividad del variador de frecuencia. Si el indicador "*Controlador*" muestra el mensaje OK con fondo verde, indica que el controlador de la cúpula está funcionando y comunicando correctamente. En caso contrario se muestra el mensaje OFF con fondo rojo indicando que el controlador no funciona. En este caso los datos y estados mostrados a continuación no son válidos.

Del mismo modo, si el indicador "*Variador"* muestra el mensaje OK con fondo verde, indica que el controlador tiene comunicación con el variador de frecuencia. En caso contrario se muestra el mensaje OFF con fondo rojo indicando que el controlador no tiene comunicación con el variador. En este caso, los comandos de movimiento de la cúpula no surtirán efecto.

Más abajo se muestra una tabla que indica los valores actuales tanto de la posición de azimut de la cúpula como del azimut objetivo establecido en el último comando de posicionamiento.

A continuación aparecen los tres botones de movimiento manual de la cúpula. Al hacer click en los botones Move Forward o Move Reverse, la cúpula comienza a girar hacia adelante o atrás respectivamente mientras se mantenga pulsado el botón. Al soltar el botón la cúpula se detiene. Si la cúpula se está moviendo debido a un comando de posicionamiento, se puede detener haciendo click en el botón Stop.

Luego se encuentra el cuadro para enviar un comando de posicionamiento al azimut introducido en el cuadro "Mover a Azimuth". Al pulsar el botón Enviar la cúpula girará hacia el azimut indicado.

Después aparece un cuadro que permite enviar directamente comandos del API MQTT del dispositivo. Al enviar un comando pulsando el botón Enviar, la respuesta aparece justo debajo.

Por último en la parte inferior se muestra un panel con un log de mensajes de información y posibles errores.

En la Figura 3.29 se muestra una captura de la pantalla de comandos de la aplicación. Esta pantalla se utiliza para facilitar el envío de comandos del API MQTT y la consulta de parámetros de configuración del controlador de la cúpula.

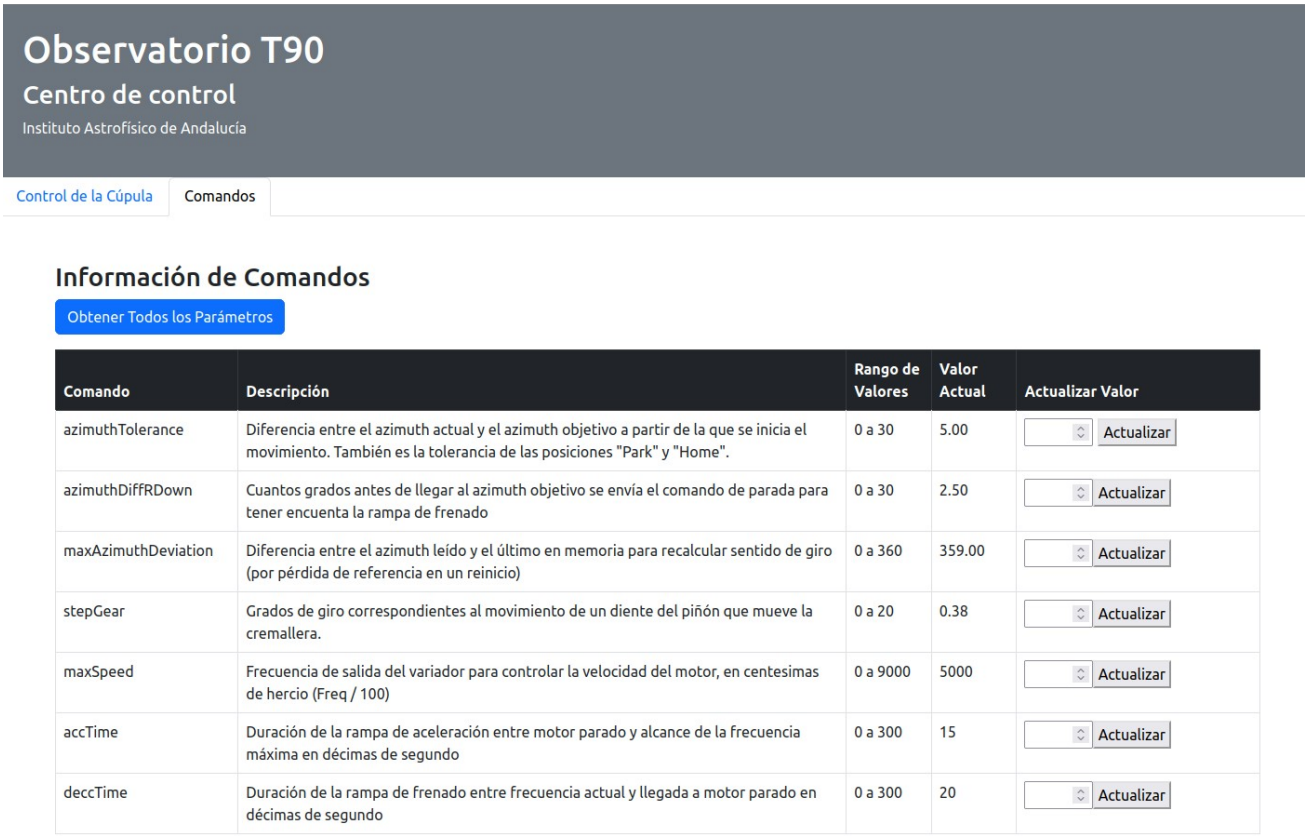

Figura 3.29: Pantalla de comandos de la aplicación web de control de la cúpula.

En la pantalla se muestra una tabla con los comandos de configuración, disponibles, su descripción, el rango de valores que aceptan, su valor actual y un cuadro para modificarlos. Pulsando el botón “Obtener Todos los Parámetros”, situado en la parte superior de la tabla, se actualiza el valor actual de todos los parámetros.

3.6 Coste de materiales

En la Tabla 3.15 se muestra el coste de los materiales utilizados en el proyecto.

No se incluye el coste del variador de frecuencia ni de los motores pues estos componentes ya estaban instalados y se utilizan en el sistema de control actual.

Tampoco se incluyen los costes del tiempo de trabajo dedicado al diseño, desarrollo y pruebas del sistema ni de los viajes al observatorio para los trabajos de instalación y pruebas.

Tabla 3.15 : Coste de materiales.

Componente	Descripción	Precio Unitario (€)	Número de unidades	Coste total (€)
Lector RFID inicial	Lector RFID con el que se realizaron las primeras pruebas	2,25	2	4,5
Etiquetas RFID iniciales	Etiquetas RFID con las que se realizaron las primeras pruebas	1,25	100	125
Lector RFID definitivo	Lector RFID finalmente utilizado	31,17	2	62,34
Antena RFID	Antena de 80 x 80 mm para el lector RFID	11,95	2	23,9
Etiquetas RFID definitivas	Etiquetas RFID finalmente utilizadas	1,92	100	192
Caja de protección lector RFID	Caja de protección para el lector RFID	24,21	2	48,42
Espuma de foam autoadhesiva	Material para pegar las etiquetas RFID a la cúpula	0,16	300	48
Beaglebone Black	Placa SBC donde se implementa el sistema de control	57,86	4	231,44
Fabricación placa de control	Coste de fabricación de la placa de control	157	2	314
Caja de protección placa de control	Caja de protección para la placa de control	56,16	1	56,16
Cable de red	Bobina de 100 metros de cable de red CAT 5e	33,95	1	33,95
Estación de trabajo.	Estación de trabajo basada en equipo de sobremesa HP.	664,7	1	664,7
TOTAL:				**1804,41**

Capítulo 4

Resultados

El proceso de desarrollo y pruebas del sistema ha sido largo por la dificultad para realizar pruebas en la instalación debido a la complicación de acceso y a que cada visita supone un viaje de varias horas de desplazamiento.

La implementación del protocolo de comunicación por Modbus con el variador de frecuencia se realizó en laboratorio realizando pruebas con el mismo variador que luego se instalaría en la cúpula, y un motor de corriente alterna trifásico. Gracias a disponer de estos elementos en el laboratorio se pudo desarrollar y depurar la comunicación con el variador sin necesidad de viajar al observatorio.

En la primera visita al observatorio, que por motivos operativos sólo fue de un día, se instalaron el primer tipo de tags probados en la cúpula (Figura 3.4) realizando alguna prueba de lectura con el primer lector utilizado.

En la segunda visita, de 4 días de duración, se instaló la placa de control de cúpula, el lector RFID, el cableado de red y el cable de par trenzado para comunicar con Modbus sobre RJ-45 con el variador. Con los tags y el lector ya instalados y con la posibilidad de girar la cúpula, se hicieron pruebas de lectura de tags más exhaustivas comprobando que la tasa de errores en la lectura de los tags era de aproximadamente del 10%. Por este motivo se decidió utilizar tags más grandes y un lector RFID con una antena de mayor tamaño. Con el sistema inicial la distancia entre el lector y el tag con la que se conseguían lecturas fiables era de menos de 15 milímetros. Uno de los objetivos era aumentar la distancia de lectura.

Tal y como se explica en el apartado 3.2.1, se realizaron en laboratorio pruebas de lectura de diferentes tipos de tags, con diferentes superficies y separaciones y con pruebas de lectura tanto con los tags estáticos como con los tags en movimiento.

Para simular el movimiento del paso de los tags frente al lector y medir la distancia de lectura fiable con el tag en movimiento se realizó un montaje pegando varios tags alrededor de la llanta de una rueda de bicicleta que giraba libre al lado de una mesa donde se colocaba el lector. (Figura 4.1)

Figura 4.1: Pruebas de lectura de tags RFID en movimiento con rueda de bicicleta. (Fuente: elaboración propia)

Con este sistema se realizaron pruebas hasta conseguir, con los nuevos tags más grandes y el nuevo lector, una distancia fiable de lectura de 40 milímetros entre los tags y el lector.

En el tercer viaje al observatorio se instalaron los nuevos tags y el nuevo lector y se realizaron las primeras pruebas de lectura sin encontrar ningún error de lectura en varias vueltas de la cúpula en ambos sentidos de rotación. Sin embargo, por falta de tiempo, no se pudo calibrar el sistema ni realizar pruebas de comunicación por ASCOM. En la Figura 4.2 se muestra una imagen tomada durante las pruebas de lectura de tags. A la izquierda se aprecia el lector RFID con una tablet a su derecha donde se ejecuta la aplicación web de control, y en la parte inferior la placa de control de cúpula en su caja de protección.

Figura 4.2: Pruebas de lectura de tags RFID en la cúpula. (Fuente: elaboración propia)

De vuelta en el laboratorio se realizaron pruebas satisfactorias de comunicación mediante ASCOM Alpaca entre el software de observación astronómica utilizado en el observatorio y el dispositivo de control. Se redactó un manual de configuración para que los observadores del OSN pudieran configurar la conexión con el dispositivo y se realizaron en remoto en el observatorio pruebas de conexión mediante ASCOM Alpaca entre el software de observación y el dispositivo de la cúpula que se había dejado instalado.

En el cuarto viaje se calibró el sistema de lectura de azimut. Para ello, gracias a que estaba funcionando el sistema de lectura de azimut basado en encoder absoluto, se grabó un vídeo en el que se podía ver a la vez un portátil con una terminal ejecutando el controlador de cúpula en modo de depuración, y la pantalla del módulo de mando actual que muestra el azimut de la cúpula leído por el encoder. En el portátil, cada vez que el controlador de cúpula leía un tag, mostraba su identificador en el terminal. De esta forma, analizando el vídeo se podía anotar en qué posición de azimut, leída por el encoder, se encontraba la cúpula cada vez que se leía un determinado tag por el lector RFID.

Con los datos extraídos del vídeo se compuso el archivo que asocia los identificadores de los tags RFID con su correspondiente posición de azimut.

Una vez con el sistema calibrado, tras probar de nuevo la conexión del software de observación astronómica con el sistema de control mediante el protocolo ASCOM Alpaca, se realizaron pruebas durante la noche con el astrofísico observador comprobando que el software de observación era capaz de mover la cúpula a través de ASCOM y posicionarla en el azimut adecuado al posicionamiento del telescopio. En la Figura 4.3 se muestra el portón de la cúpula alineado correctamente con la posición del telescopio tras realizar una operación de apuntado del telescopio.

Figura 4.3: Pruebas de alineamiento de la cúpula con el telescopio. (Fuente: elaboración propia)

También se comprobó que el software de observación podía mover la cúpula en intervalos regulares de tiempo para seguir el movimiento del telescopio a lo largo de una observación normal.

Actualmente los astrofísicos observadores están realizando las primeras pruebas de control de la cúpula en situaciones reales de observación. En la Figura 4.4 se muestra una captura del sistema de visualización de datos donde se puede ver cómo, durante el intervalo de una observación, el azimut de la cúpula (línea verde) se va ajustando a la posición de azimut objetivo (línea punteada amarilla) comandada por el software de observación.

Figura 4.4: Pruebas de control de la cúpula en el transcurso de una observación astronómica. (Fuente: elaboración propia)

Capítulo 5

Conclusiones y trabajos futuros

En este trabajo se ha presentado el desarrollo e implementación de un sistema de control de cúpula de gran telescopio basado en la lectura de identificadores RFID para medir la posición de azimut de la cúpula.

El proyecto abarca todos los sistemas necesarios para una solución de control de cúpula completa. Desde la lectura de sensores para medir el azimut de la cúpula, pasando por el sistema de control que determina la actuación necesaria sobre los motores que la mueven, hasta el sistema de monitorización y visualización de los datos de funcionamiento de la cúpula.

El proyecto ha logrado renovar el sistema de control de la cúpula del telescopio T-90 del Observatorio de Sierra Nevada (OSN)[1] con un sistema basado en estándares abiertos y tecnologías actuales, lo que mejora la fiabilidad y mantenibilidad del sistema de la cúpula.

Para la detección de posición se ha propuesto una solución innovadora implementada con un sistema híbrido que combina un sistema de lectura de tags RFID con 96 etiquetas distribuidas en la cúpula proporcionando una medida de posición absoluta con una resolución de aproximadamente 3.75°, y un sistema basado en un sensor magnético que detecta el paso de los dientes de la rueda dentada que mueve la cremallera de la cúpula para actualizar de forma relativa la posición de azimut, entre lecturas de etiquetas RFID, con una resolución de 0.38°.

Esta solución evita partes móviles y sensores ópticos, siendo robusta frente a interferencias eléctricas y condiciones ambientales.

Otra característica destacada del proyecto es la implementación en el dispositivo de control de una interfaz de comunicación compatible con el protocolo estandarizado ASCOM Alpaca[2]. Esto permite controlar el sistema de la cúpula a través de cualquier aplicación que soporte este estándar como son las aplicaciones de observación astronómicas más utilizadas en observatorios profesionales.

Además se ha implementado otra interfaz de comunicación basada en MQTT[3], un protocolo ligero de tipo publicación, suscripción, ampliamente adoptado en entornos donde se requiere transmitir datos en tiempo real entre múltiples dispositivos con una infraestructura de red limitada o variable.

El sistema se ha diseñado siguiendo una arquitectura modular basada en software y estándares abiertos proporcionando una gran flexibilidad y escalabilidad.

El dispositivo de control está basado en una placa SBC BeagleBone Black[13] con sistema operativo Linux conectado al servidor central a través de la red local Ethernet del observatorio.

En la monitorización del sistema se utiliza la base de datos de series temporales Graphite DB[25] para mantener un histórico de datos de funcionamiento, utilizando como herramienta de visualización el software Grafana[24], una plataforma de software de código abierto diseñada para la visualización y análisis de datos en tiempo real que se utiliza ampliamente en monitorización de sistemas y redes, observabilidad de infraestructuras, IoT, análisis industrial y aplicaciones científicas.

También se ha desarrollado una aplicación web basada en tecnologías estándar (HTML, CSS, JavaScript y websockets) para el control manual y la configuración del sistema de la cúpula.

Las pruebas realizadas en el observatorio han confirmado una lectura 100% fiable de las etiquetas RFID, tras optimizar el tamaño de los tags y del lector.

También se ha probado con resultados positivos el control del sistema de la cúpula mediante el software de observación astronómica utilizado en el observatorio utilizando la comunicación con el protocolo ASCOM-Alpaca[6].

Actualmente se están realizando las primeras pruebas de control de la cúpula en situaciones reales de observación.

A continuación se describen posibles mejoras y trabajos futuros relacionados con el proyecto.

En la aplicación del sistema de control, el bucle de eventos se ejecuta de forma periódica cada 10 milisegundos comprobando en cada ejecución si se han generado eventos de los diferentes sistemas. Una posible mejora en este sentido sería utilizar las herramientas proporcionadas por el sistema operativo Linux para despertar al proceso de control en el momento en el que se produce un evento. Aunque en las pruebas realizadas hasta el momento no se han detectado retardos y la responsividad del sistema es buena, esto podría mejorar el tiempo de respuesta.

En la interfaz de comunicación por MQTT podría ampliarse el API de comandos para permitir nuevas funcionalidades. Otra una posible mejora en este aspecto sería la utilización de cifrado TLS para las comunicaciones por MQTT.

Respecto al sistema de monitorización, una posible actualización sería cambiar la base de datos de series temporales para utilizar Influx DB. Aunque Graphite DB es una buena opción

debido a su facilidad de uso, ligereza y escalabilidad, Influx DB es una solución más actual y con mucho soporte dada su gran implantación.

En este aspecto también cabe destacar la posibilidad de modificaciones y creación de nuevos dashboard de visualización en Grafana adaptados a las preferencias y necesidades de los usuarios finales, así como la creación de posibles alarmas y avisos en caso de que ciertos parámetros medidos se salgan de los umbrales de seguridad que se puedan definir. Gracias a la facilidad de uso de Grafana este tipo de modificaciones se pueden realizar de forma sencilla sin necesidad de grandes desarrollos.

Además, la aplicación web de control manual también está sujeta a posibles modificaciones o ampliaciones según las necesidades y sugerencias que propongan los usuarios finales del sistema.

Respecto al hardware del servidor central, cabría la posibilidad de instalar un sistema host con varios discos duros montados en un sistema RAID (redundant array of independent disks). RAID[51] es un sistema de almacenamiento que utiliza varios discos para distribuir o replicar los datos. Aunque esto no evitaría la necesidad de realizar periódicamente puntos de restauración y copias de seguridad de la imagen del sistema virtualizado, proporcionaría una mayor integridad y tolerancia frente a fallos.

Por último, como trabajo futuro, una vez que el sistema haya sido probado durante un tiempo y se haya demostrado su fiabilidad y robustez, cabría la posibilidad de replicarlo en el telescopio T-150 del OSN.

La modernización del sistema no solo extiende la vida útil del telescopio, sino que también facilita observaciones astronómicas de mayor precisión, consolida al OSN como un observatorio competitivo y sienta las bases para futuras actualizaciones tecnológicas. La combinación de RFID, estándares abiertos y herramientas de código abierto lo convierte en un modelo replicable para otros observatorios.

Bibliografía y referencias

1. IAA-CSIC. Observatorio de Sierra Nevada. [En línea]. Disponible en: https://www.osn.iaa.csic.es/ . [Accedido: 30-may-2025].

2. The ASCOM Initiative. ASCOM - Standards for Astronomy. [En línea]. Disponible en: https://ascom-standards.org/ . [Accedido: 30-may-2025].

3. MQTT.org. MQTT: The Standard for IoT Messaging. [En línea]. Disponible en: https://mqtt.org/ . [Accedido: 30-may-2025].

4. Yerkes Future Foundation. Homepage - Yerkes Observatory. [En línea]. Disponible en: https://yerkesobservatory.org/ . [Accedido: 30-may-2025].

5. Mount Wilson Institute. Mount Wilson Observatory. [En línea]. Disponible en: https://www.mtwilson.edu/ . [Accedido: 30-may-2025].

6. The ASCOM Initiative. 1. About ASCOM and Alpaca. [En línea]. Disponible en: https://ascom-standards.org/About/Overview.htm . [Accedido: 30-may-2025].

7. Hohner Automation. Hohner Automation | Incremental & Absolute Encoders. [En línea]. Disponible en: https://www.encoderhohner.com/ . [Accedido: 30-may-2025].

8. Ash Manufacturing Company. Home - Ash-Dome. [En línea]. Disponible en: https://ashdome.com/ . [Accedido: 30-may-2025].

9. Baader Planetarium GmbH: Zur Sternwarte. Baader Planetarium GmbH – Aiders in Astronomy - English. [En línea]. Disponible en:https://www.baader-planetarium.com/ . [Accedido: 30-may-2025].

10. Baader Planetarium GmbH: Zur Sternwarte. Baader Slit-Domes (Advanced) – 3.2 to 8.5 Meter - English. [En línea]. Disponible en: https://www.baader-planetarium.com/en/baader-slit-domes-advanced-3-2-to-8-5-meter.html . [Accedido: 30-may-2025].

11. Baader Observatories. YEBES Observatory (IGN/CNIG) . [En línea]. Disponible en: https://www.baader-observatories.com/de/mapped-posts/yebes-observatory-ign-cnig/ . [Accedido: 30-may-2025].

12. Baader Observatories. MPS (Max-Planck Institut für Sonnensystemforschung). [En línea]. Disponible en: https://www.baader-observatories.com/de/mapped-posts/mps-max-planck-institut-fuer-sonnensystemforschung/ . [Accedido: 30-may-2025].

13. BeagleBoard.org Foundation. BeagleBone® Black - BeagleBoard. [En línea]. Disponible en: https://www.beagleboard.org/boards/beaglebone-black . [Accedido: 30-may-2025].

14. EPICS Controls. EPICS - Experimental Physics and Industrial Control System. [En línea]. Disponible en: https://epics-controls.org/ . [Accedido: 30-may-2025].

15. European Southern Observatory. Very Large Telescope | ESO. [En línea]. Disponible en: https://www.eso.org/public/teles-instr/paranal-observatory/vlt/ . [Accedido: 30-may-2025].

16. ALMA Observatory. In search of our cosmic origins | ALMA Observatory. [En línea]. Disponible en: https://www.almaobservatory.org/en/home/ . [Accedido: 30-may-2025].

17. European Southern Observatory. Extremely Large Telescope brochure. [En línea]. Disponible en: https://www.eso.org/public/archives/brochures/pdf/brochure_0054.pdf . [Accedido: 30-may-2025].

18. P. Kubánek. RTS2: open source standard and package for autonomous observatory. [En línea]. Disponible en: https://rts2.org/ . [Accedido: 30-may-2025].

19. INDI Library. Introduction | INDI Technical Documentation. [En línea]. Disponible en: https://docs.indilib.org/ . [Accedido: 30-may-2025].

20. Instituto de Astrofísica de Canarias • IAC. Nuevo sistema de control para los Telescopios Nocturnos. [En línea]. Disponible en: https://www.iac.es/es/proyectos/nuevo-sistema-de-control-para-los-telescopios-nocturnos . [Accedido: 30-may-2025].

21. G. Zhang et al., "An Autonomous Observation and Control System Based on EPICS and RTS2 for Antarctic Telescopes", *MNRAS* (January 11, 2016) 455 (2): 1654-1664. Disponible en: https://arxiv.org/pdf/1511.06950 . [Accedido: 30-may-2025].

22. A. Kumar, A. Sengupta, S. Ganesh, "Autonomous Dome For Robotic Telescope", *Revista Mexicana de Astronomia y Astrofisica Conference Series*, Vol. 48 29-30 (2016). Disponible en: https://arxiv.org/pdf/1604.08370 . [Accedido: 30-may-2025].

23. D. Ricci, L. Cabona, S. Tosi, S. Zappatore, "Toward the remotization and robotization of the OARPAF telescope", en SPIE Astronomical Telescopes + Instrumentation, 2022, Montréal, Québec, Canada. Disponible en: https://arxiv.org/pdf/2501.08016 . [Accedido: 30-may-2025].

24. Grafana Labs. Grafana: The open and composable observability platform. [En línea]. Disponible en: https://grafana.com/ . [Accedido: 30-may-2025].

25. Chris Davis. Graphite. [En línea]. Disponible en: https://graphiteapp.org/ . [Accedido: 30-may-2025].

26. Eclipse Foundation AISBL. Eclipse Mosquitto. [En línea]. Disponible en: https://mosquitto.org/ . [Accedido: 30-may-2025].

27. Lighty team. Home - Lighttpd - fly light. [En línea]. Disponible en: https://www.lighttpd.net/ . [Accedido: 30-may-2025].

28. Fundación Wikimedia. ISO 14443 - Wikipedia, la enciclopedia libre. [En línea]. Disponible en: https://es.wikipedia.org/wiki/ISO_14443 . [Accedido: 30-may-2025].

29. NXP Semiconductors. Standard performance MIFARE® and NTAG® frontend. [En línea]. Disponible en: https://www.nxp.com/products/rfid-nfc/nfc-hf/nfc-readers/standard-performance-mifare-and-ntag-frontend:MFRC52202HN1 . [Accedido: 30-may-2025].

30. Eccel Technology. RFID NFC Chilli USB-B1. [En línea]. Disponible en: https://eccel.co.uk/product/chilli-usb-b1/ . [Accedido: 30-may-2025].

31. LS ELECTRIC Co., Ltd. Product -iG5A. [En línea]. Disponible en: https://www.ls-electric.com/products/view/Smart_Automation_Solution/AC_Drive_-*VFD*-/Low_Voltage_VFD/iG5A . [Accedido: 30-may-2025].

32. Modbus Organization, Inc. Modbus Technical Resources. [En línea]. Disponible en: https://www.modbus.org/tech.php . [Accedido: 30-may-2025].

33. Fundación Wikimedia. TIA-485 - Wikipedia, la enciclopedia libre. [En línea]. Disponible en: https://es.wikipedia.org/wiki/TIA-485 . [Accedido: 30-may-2025].

34. Texas Instruments Incorporated. SN65HVD3085E data sheet, product information and support | TI.com. [En línea]. Disponible en: https://www.ti.com/product/SN65HVD3085E . [Accedido: 30-may-2025].

35. Thomas Kuschel, “@kzoltaan. Thanks for sharing your idea, I finally got my circuit to work properly with the RS485 on my UART2 and the RS-485-DE signal switched automatically.”, 30-ene-2021. [Foro]. Disponible en: https://github.com/RobertCNelson/bb-kernel/issues/38#issuecomment-770122621 . [Accedido: 30-may-2025].

36. Stéphane Raimbault. libmodbus. [En línea]. Disponible en: https://libmodbus.org/ . [Accedido: 30-may-2025].

37. Eclipse Foundation AISBL. Eclipse Paho, “MQTT C Client for Posix and Windows”. [En línea]. Disponible en: https://eclipse.dev/paho/clients/c/ . [Accedido: 30-may-2025].

38. ASCOM Initiative. ASCOM Alpaca API. [En línea]. Disponible en: https://ascom-standards.org/api/#/ . [Accedido: 30-may-2025].

39. ASCOM Initiative. ASCOM ALPACA API Reference - Version 9. [En línea]. Disponible en: https://ascom-standards.org/AlpacaDeveloper/ASCOMAlpacaAPIReference.html . [Accedido: 30-may-2025].

40. Petri Lehtinen. Jansson Documentation — Jansson 2.14.1 documentation. [En línea]. Disponible en: https://jansson.readthedocs.io/en/latest/ . [Accedido: 30-may-2025].

41. ASCOM Initiative. ASCOMInitiative/ConformU: ConformU is a cross-platform tool to validate that Alpaca Devices and ASCOM Drivers conform to the ASCOM interface specification. It supersedes the original Windows based Conform application. [En línea]. Disponible en: https://github.com/ASCOMInitiative/ConformU#readme . [Accedido: 30-may-2025].

42. Canonical Ltd. Canonical LXD, Lightweight open source virtualisation with LXD. [En línea]. Disponible en: https://canonical.com/lxd . [Accedido: 30-may-2025].

43. Canonical Ltd. Ubuntu releases, Ubuntu 22.04.5 LTS (Jammy Jellyfish). [En línea]. Disponible en: https://releases.ubuntu.com/jammy/ . [Accedido: 30-may-2025].

44. Tailscale Inc. Tailscale · Best VPN Service for Secure Networks. [En línea]. Disponible en: https://tailscale.com/ . [Accedido: 30-may-2025].

45. Jason A. Donenfeld. WireGuard: fast, modern, secure VPN tunnel. [En línea]. Disponible en: https://www.wireguard.com/ . [Accedido: 30-may-2025].

46. Thomas Nordquist. MQTT Explorer | An all-round MQTT client that provides a structured topic overview. [En línea]. Disponible en: https://mqtt-explorer.com/ . [Accedido: 30-may-2025].

47. Bootstrap team. Bootstrap · The most popular HTML, CSS, and JS library in the world. [En línea]. Disponible en: https://getbootstrap.com/ . [Accedido: 30-may-2025].

48. OpenJS Foundation and jQuery contributors. jQuery, a fast, small, and feature-rich JavaScript library. [En línea]. Disponible en: https://jquery.com/ . [Accedido: 30-may-2025].

49. Eclipse Foundation AISBL. Eclipse Paho JavaScript Client. [En línea]. Disponible en: https://eclipse.dev/paho/clients/js/ . [Accedido: 30-may-2025].

50. InfluxData Inc. InfluxDB Time Series Data Platform. [En línea]. Disponible en: https://www.influxdata.com/ . [Accedido: 30-may-2025].

51. Fundación Wikimedia. RAID - Wikipedia, la enciclopedia libre. [En línea]. Disponible en: https://es.wikipedia.org/wiki/RAID . [Accedido: 30-may-2025].