Universidad de Concepción
Departamento de Ingeniería Informática y Ciencias de la Computación

DISEÑO E IMPLEMENTACIÓN DE SISTEMA DE CONTROL BASADO EN ICE EN RADIOTELESCOPIO 3M.

Pablo Javier Furet Pereira

Memoria presentada para la obtención del título de Ingeniero Civil Informático

Profesor Patrocinante: Gonzalo Rojas Durán

Marzo de 2023

Índice

Glosario

Alt/Az: Altitud y Azimut (equivalente a Az/El). También referido como “altazimutal”. Sistema de coordenadas horizontales.

Az/El: Azimut y Elevación. Nombre alterno para Alt/Az.

CORBA: Common Object Request Broker Architecture. Framework para facilitar la comunicación entre sistemas heterogéneos y distribuidos.

CSO: Caltech Submillimeter Observatory. Observatorio en Hawaii, decomisionado en 2015. Su radiotelescopio será usado en LCT .

H I: Hidrógeno neutro. “H” siendo el símbolo químico del hidrógeno e “I” siendo el número romano, usado a menudo en astronomía para átomos neutros.

Ice: Internet Communication Engine. Framework RPC, con orientación a objetos.

LCT: Leighton Chajnantor Telescope. Proyecto de nuevo observatorio en Chile.

ORB: Object Request Broker. Paradigma, orientado a objetos, de middleware.

RA/Dec: Right Ascension & Declination. Sistema de coordenadas ecuatoriales.

RPC: Remote Procedure Call. Acción en la cual un proceso invoca la ejecución de otro, en un sistema remoto, como si fuera local.

SDR: Software Defined Radio. Sistema de radio en el cual los componentes, tradicionalmente análogos, son reemplazados por software.

1. Introducción

El proyecto Leighton Chajnantor Telescope (LCT) consiste en el traslado, comisionamiento y refacción de un radiotelescopio de 10.4 metros de diámetro (anteriormente utilizado en el Caltech Submillimeter Observatory o CSO, ver Figura 1a). El radiotelescopio, anteriormente ubicado en Mauna Kea, Hawaii, será instalado en el norte de Chile (Llano de Chajnantor, San Pedro de Atacama). Se trata de un proyecto colaborativo desarrollado por la Universidad de Concepción, el Instituto de Tecnología de California y la Universidad Normalista de Shanghai.

El telescopio cuenta con una excelente precisión en la superficie del reflector principal y una gran flexibilidad en cuanto a la instrumentación que puede albergar, ya que cuenta con tres plataformas (dos Nasmyth y una Cassegrain). En cuanto al refaccionamiento, uno de los aspectos en los que se trabaja es en mejorar el Sistema de Control del telescopio, específicamente, en actualizar las interfaces de comunicación (tanto software como hardware).

Respecto a software, el actual Sistema de Control del CSO funciona en base a CORBA (Common Object Request Broker Architecture), estándar de interoperabilidad entre componentes heterogéneos, lo que facilita la comunicación entre distintos equipos y aplicaciones del observatorio. Sin embargo, la tecnología actualmente utilizada para su implementación (MICO y OmniORB) no ha sido actualizada en más de 20 años.

Por otro lado, en cuanto al hardware, el Sistema de Control del CSO no cuenta con un estándar de comunicación definido, mientras que parte de las interfaces de comunicación utilizadas están implementadas con tecnología actualmente discontinuada/obsoleta. Lo anterior genera un potencial problema en caso de presentarse fallas en alguno de los equipos, ya que esto podría dificultar o retardar la reparación o mantención de los mismos, dejando al telescopio, o parte de él, fuera de operación por un tiempo.

Debido a lo anterior, surge la necesidad de actualizar los componentes de hardware y software encargados de la comunicación entre los distintos equipos e instrumentos del telescopio.

Por otra parte, el Observatorio Wenulafken consiste en un radiotelescopio de 3 metros (ver Figura 1b), instalado a 2.9 km del campus central de la Universidad de Concepción. Este radiotelescopio tiene objetivos principalmente académicos, permitiendo introducir tanto a futuros astrónomos y profesionales de áreas técnicas e ingeniería en tópicos asociados a instrumentación astronómica. El radiotelescopio de 3m permite medir la línea de hidrógeno neutro a 1.42GHz, con lo que es posible observar y obtener características relevantes de la Vía Láctea.

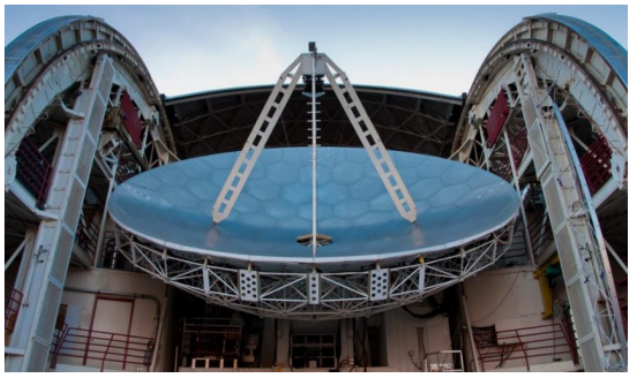

Figura 1a: Radiotelescopio LCT

Figura 1b: Radiotelescopio 3m

El actual sistema de control del observatorio de 3m implementa una arquitectura relativamente simple, con pocos componentes instalados. Sin embargo, debido a la futura integración de nuevos instrumentos y equipos, se propone implementar un modelo arquitectónico que permita la interoperabilidad entre distintos equipos y aplicaciones del observatorio.

Por consiguiente, se propone actualizar la capa de software encargada de la comunicación entre aplicaciones dentro del observatorio utilizando Ice (Internet Communications Engine), que es uno de los framework de llamada a procedimiento remoto (RPC) más utilizados y con más soporte en la actualidad, como prueba de esta tecnología y futuro escalamiento de esta solución al radiotelescopio LCT.

1.1. Objetivo General

Analizar empíricamente la implementación del framework RPC Ice como soporte al sistema de control de un radiotelescopio de 3m, para su futuro escalamiento en radiotelescopios de gran magnitud.

1.2. Objetivos Específicos

1. Comprender las características y la arquitectura del actual Sistema de Control del 3m.
2. Estudiar las principales características de Ice.
3. Diseñar e implementar un conjunto de APIs a través de Ice considerando los requerimientos del Sistema de Control del 3m.
4. Analizar factibilidad de implementación de plataforma basada en Ice para el LCT.

1.3. Metodología

Se realizó un análisis de la arquitectura actual del Sistema de Control del radiotelescopio de 3m, con revisión de documentación, instalaciones y funcionamiento, con foco en la comunicación entre sus distintos componentes software.

Se analizaron requerimientos de interoperabilidad entre los componentes actualmente instalados y los que se proyecta instalar.

En base al estudio del middleware Ice, se analizaron comparativamente sus prestaciones y soporte a los requerimientos solicitados con otras alternativas de frameworks RPC.

Se diseñó la arquitectura software del sistema en lenguaje de alto nivel de abstracción, y se implementó un prototipo de esta arquitectura en las instalaciones del Observatorio Wenulafken, basándose en el sistema de control de CSO, adoptando un método de desarrollo iterativo e incremental, con adaptaciones de propuestas ágiles a un entorno de desarrollo unipersonal.

Se puso a prueba el prototipo desarrollado con un caso de uso real, durante actividades de un curso de astronomía impartido por la UdeC.

1.4. Estructura del Informe

Los siguientes capítulos son el marco teórico (cap. 2), donde se resumen conceptos claves relacionados al trabajo de esta memoria, descripción de la propuesta (cap. 3), donde se resume la intención y lo realizado en este trabajo, detalle de la propuesta (cap. 4), donde explica en detalle la arquitectura e implementación del sistema propuesto, evaluación de la propuesta (cap. 5), donde discute el éxito alcanzado con el sistema propuesto, conclusiones (cap. 6), donde se discute la pregunta que podría resumir el propósito de este trabajo, “¿Es Ice adecuado para su uso como framework de soporte en radiotelescopios de gran escala?”, y trabajos futuros (cap. 7), donde se proponen ideas para expandir el trabajo de esta memoria.

2. Marco Teórico

Previamente se ha hecho mención a radiotelescopios, distintos observatorios y middlewares. En este capítulo se expandirá sobre estos conceptos y algunos más, útiles para la comprensión de este trabajo.

2.1. Radioastronomía

La radioastronomía solo difiere de la astronomía en el rango del espectro electromagnético sobre el cual trabaja y este, como su nombre lo insinúa, es el de las ondas de radio y un radiotelescopio es, por supuesto, un telescopio construido para captarlas. En la Figura 1 se muestra una fotografía del radiotelescopio utilizado en el observatorio Wenulafken (antes de ser trasladado a este). En la figura se resaltan algunos componentes importantes, estos son la bocina (captura las ondas reflejadas por el plato reflectante y las envía a un analizador de espectro o SDR), el plato reflectante (este refleja las ondas electromagnéticas y las enfoca en la bocina) y el disco central (para este radiotelescopio, es usado como referencia para juzgar la precisión de apuntamiento).

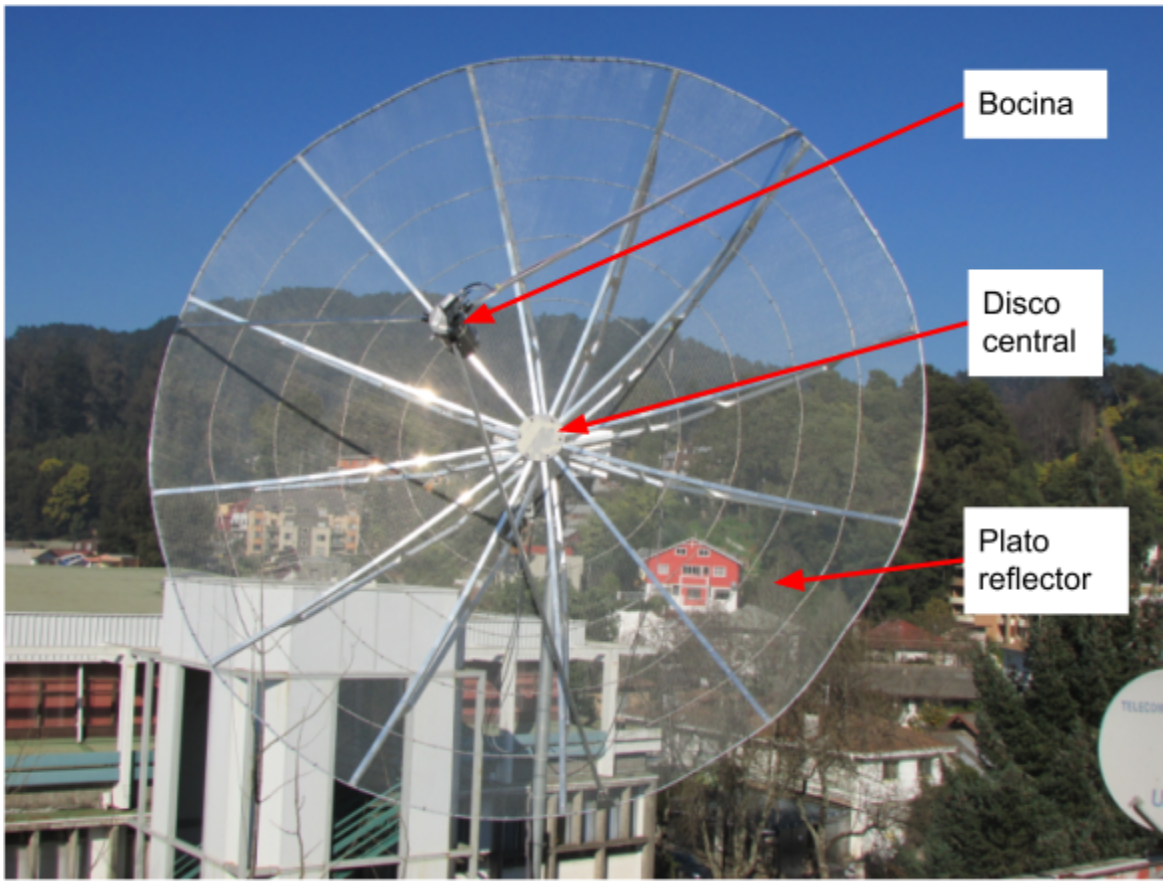

Figura 1: Fotografía del radiotelescopio usado en el observatorio Wenulafken, resaltando algunos componentes de interés.

2.1.1. Ondas de Radio

Representando el rango de entre 3 kHz y 300 GHz (correspondientes a longitudes de onda de 100 km y 1 mm aprox.), estas ondas además de posibilitar el estudio de distintos fenómenos, tienen la ventaja de ser capaces de atravesar nubes de polvo, permitiendo la observación de objetos invisibles para telescopios ópticos convencionales [13]. Adicionalmente, nuestra atmósfera es muy transparente para frecuencias entre 18 MHz y 40 GHz (longitudes de onda entre 16 metros y 7 mm aprox), como se muestra en la Figura 2.

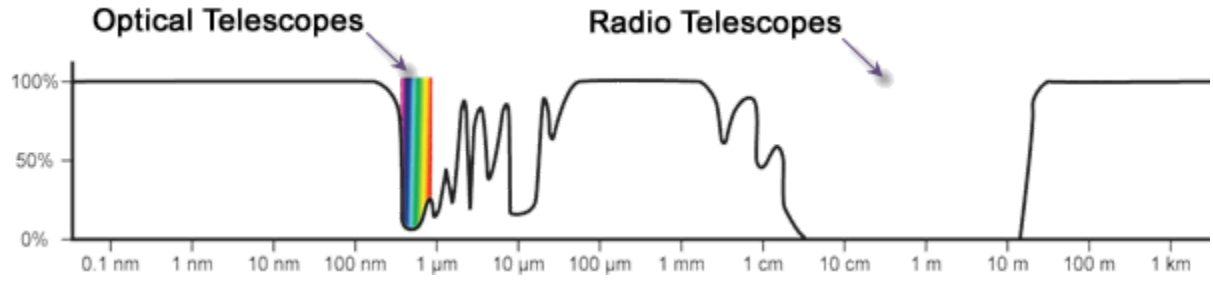

Figura 2: Transparencia de la atmósfera terrestre para distintas longitudes de onda del espectro electromagnético. (Hardhack [13])

2.1.2. Frecuencia de interés

Todo radiotelescopio tiene un rango de frecuencias para cuya observación está construido. En el caso del observatorio de Wenulafken, las ondas de interés son aquellas emitidas por el hidrógeno neutro, con aprox 1.42 GHz de frecuencia (longitud de onda de ~21 cm). Este se encuentra presente en todo el universo, pero en ciertas regiones (como dentro de galaxias) se encuentra en mucho mayor densidad. Por otro lado, la larga longitud de onda de estas emisiones significa que se ven poco afectadas por nubes de polvo, lo que nos permite estudiar regiones mucho más lejanas (véase [8, 9]).

2.1.3. Sistemas de Coordenadas

Existen múltiples sistemas de coordenadas para describir la posición de objetos en el cielo, cada uno con sus propios puntos de referencia. En este proyecto se utilizan principalmente Ascensión Recta & Declinación (RA/Dec, por sus siglas en inglés), y Altitud & Azimut (Alt/Az, a veces referido como Azimut y Elevación, Az/El). Vale notar que en ambos sistemas coordenados, cuando se usan grados, suelen utilizarse subdivisiones sexagesimales, es decir; grados, arcominutos (un arcominuto = 1‘ = 1/60 grados) y arcosegundos (un arcosegundo = 1’’ = 1/60 arcominutos).

2.1.3.1. Coordenadas RA/Dec

Las coordenadas RA/Dec tienen como referencia el ecuador y polos celestes (estos son la proyección del ecuador y polos terrestres sobre la esfera celeste). Usando este sistema de referencia, las coordenadas de los objetos en el cielo son las mismas para cualquier observador en la tierra y los objetos están estáticos en la esfera celeste (o se mueven muy, muy lentamente). Los componentes de estas coordenadas se muestran en la Figura 3a. La Ascensión Recta (RA) son los grados en el plano ecuatorial desde un punto cero. Estos se miden en horas, minutos y segundos de tiempo. El cielo parece girar 360° en 24 hrs, o 15° en una hora, por lo que una de ascensión recta equivale a 15° de rotación [14]. En la Figura 3a este punto está representado como γ y se define como uno de los puntos donde el ecuador celeste intersecta el plano de la eclíptica terrestre (el plano definido por la órbita de la tierra alrededor del sol), en particular el punto donde el sol pasa a estar sobre el hemisferio norte (momento que se conoce como el equinoccio de otoño para aquellos en el hemisferio sur, y equinoccio de primavera para aquellos en el norte). La Declinación son los grados de altura por sobre o por debajo del plano ecuatorial, de +90° a -90°, respectivamente.

2.1.3.2. Coordenadas Alt/Az

También llamadas altazimutales, tienen como referencia el plano del horizonte de la ubicación del observador, y como polos el Cenit y Nadir (puntos directamente sobre y debajo del observador, respectivamente). Este sistema de coordenadas usa la Tierra como referencia y es el cielo el que se mueve alrededor de esta. Consecuentemente, los objetos en el cielo se mueven a una velocidad comparable a la rotación de la Tierra, y sus posiciones dependen en gran medida de la ubicación geográfica del observador sobre el planeta. Los componentes de este sistema se muestran en la Figura 3b. El Azimut (Az) se mide en grados (0 a 360°) respecto al norte, y la Altitud (Alt) son los grados de altura por sobre el horizonte (de +90° a -90°, aunque coordenadas por debajo del horizonte no suelen ser directamente visibles).

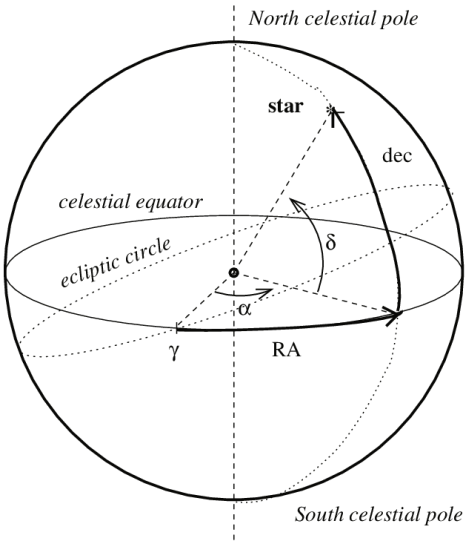

Figura 3a: Sistema de coordenadas ecuatoriales (RA/Dec). (J. Patris 2010 [11])

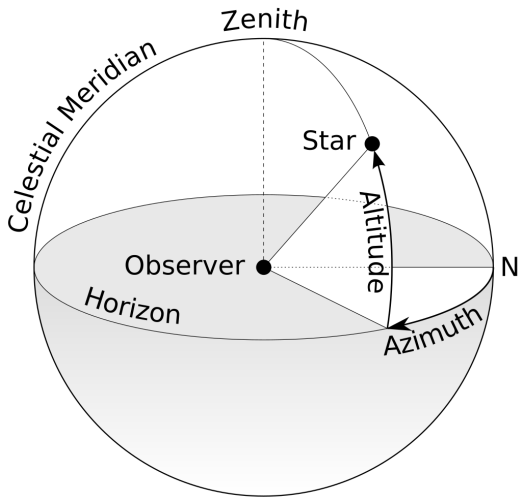

Figura 3b: Sistema de coordenadas horizontales (Alt/Az). (Wikipedia [10])

2.1.4. Caracterizando la Vía Láctea

Es posible identificar qué región de la Vía Láctea estamos observando mediante el análisis de las velocidades de los objetos observados.

Al observar una región de nuestra galaxia, estaremos simultáneamente observando varias regiones de varios brazos. La Figura 4 muestra un diagrama de esto. En ella se ilustra como nuestro cono de observación incluirá información de estrellas y objetos de múltiples brazos distintos. Cada una de estas regiones se mueve a una velocidad distinta a las otras, respecto a nuestra posición. Podemos calcular estas velocidades al analizar el corrimiento hacia el rojo o el azul de las líneas de hidrógeno captadas. La combinación de estas velocidades generan un perfil que es propio de la región observada. La Figura 5a muestra un ejemplo de tales perfiles, en este caso de una región de nuestra galaxia de coordenadas (RA/Dec) 12h 26m 32s -63.09°. Como comparación se muestra el perfil del centro de nuestra galaxia (Figura 5b), donde se puede observar una única velocidad distinguible de casi 0 km/s. Esto se debe a que al mirar en dirección al centro de nuestra galaxia, todos los objetos de todos los brazos incluidos en nuestro cono de observación se están moviendo de forma casi perfectamente perpendicular a la dirección de la observación, lo que implica que sus movimientos tienen un componente casi nulo en esta dirección y por ende, corrimientos hacia el rojo/azul casi imperceptibles. Los perfiles fueron extrapolados utilizando la herramienta provista por Argelander Institut für Astronomie [7].

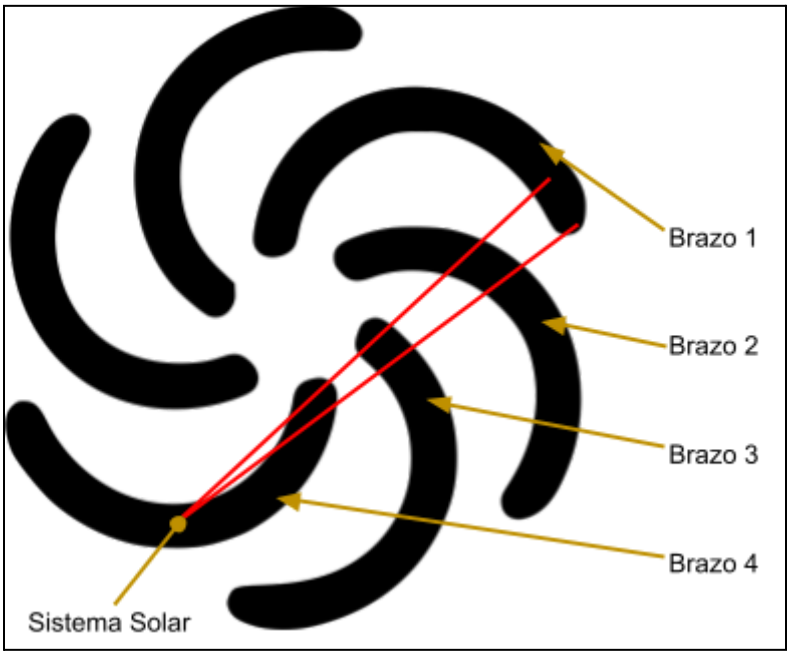

Figura 4: Al observar una región del cielo en el plano galáctico recibimos luz proveniente de múltiples brazos.

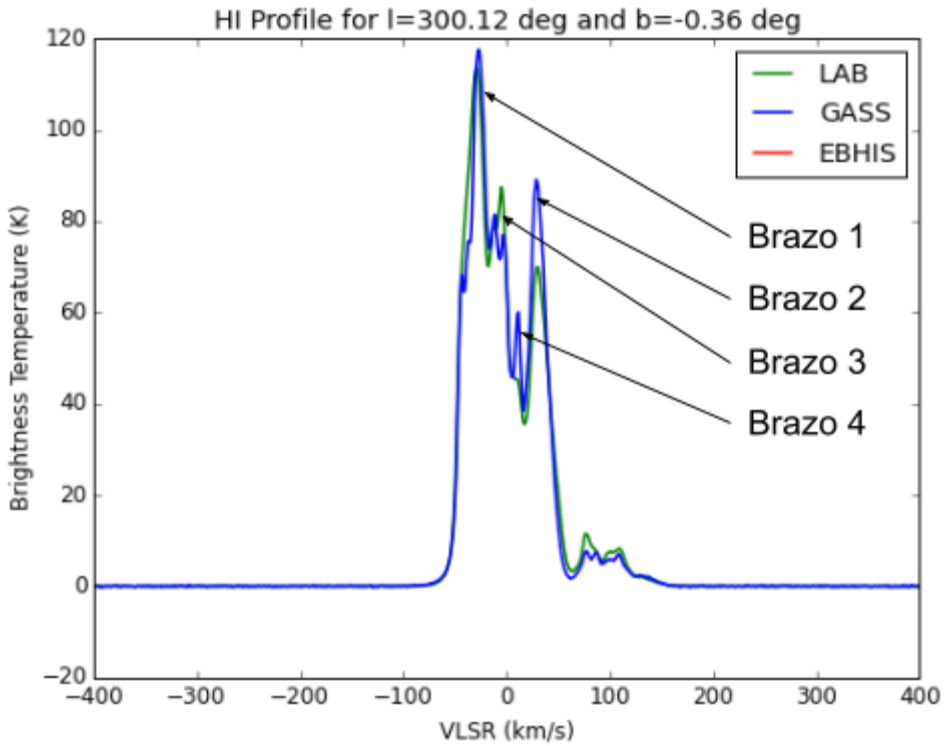

Figura 5a: Ejemplo de un perfil de hidrógeno neutro (H I) de una región arbitraria de nuestra galaxia. Coordenadas RA/Dec: 12h 26m 32s/-63° 05’ 44’’.

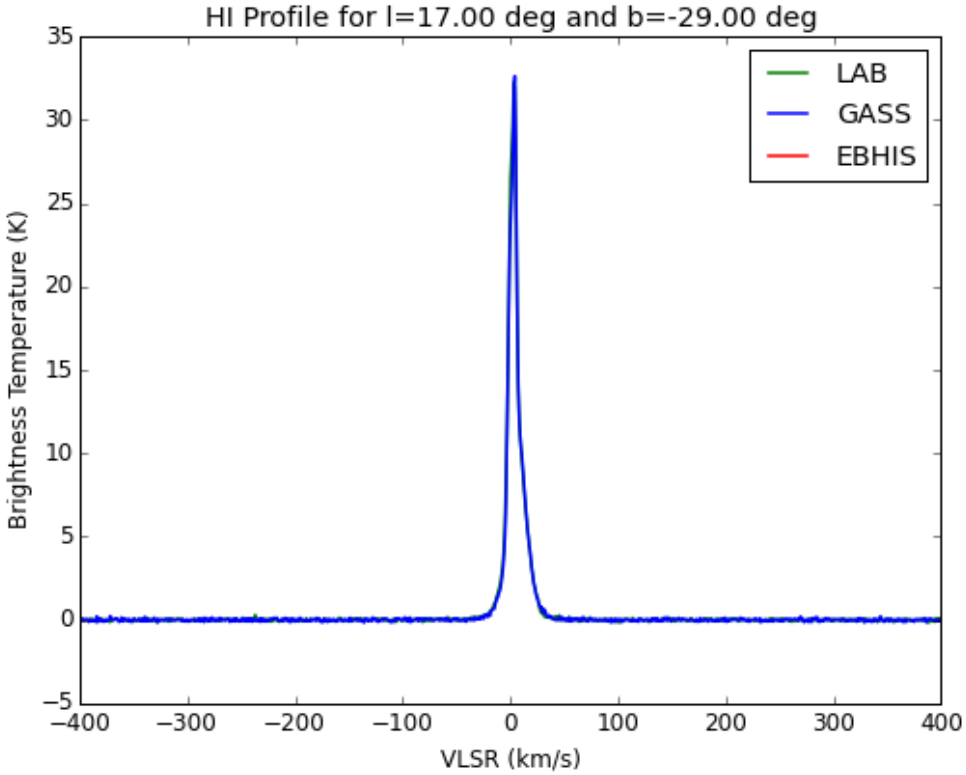

Figura 5b: Perfil de hidrógeno neutro (H I) del centro de nuestra galaxia. Coordenadas RA/Dec: 17h 45m 39s/-29° 00’ 26’’.

2.2. CSO, LCT y Wenulafken

El Caltech Submillimeter Observatory (CSO) fue un observatorio ubicado en Hawai destinado a la astronomía submilimétrica, desmantelado en el año 2015.
Este observatorio contaba con una de las (inicialmente) 4 antenas de ∅10.4 m diseñadas por R. Leighton [16].

Leighton Chajnantor Telescope (LCT) es un proyecto colaborativo entre la Universidad de Concepción (UdeC), el Instituto de Tecnología de California (Caltech) y la Universidad Normalista de Shanghai (ShNU) que busca reubicar el telescopio Leighton del CSO a la meseta de Chajnantor, Chile.

El observatorio Wenulafken pertenece a la Universidad de Concepción y es utilizado principalmente para enseñar a los estudiantes de astronomía el uso de instrumentación, en cursos como Fundamentos de Instrumentación Astronómica (FIA). Este observatorio cuenta con un simple radiotelescopio de 3m de diámetro, diseñado principalmente para observar las emisiones del hidrógeno neutro (H I), y se ha escogido para esta memoria como sustituto de LCT para poner a prototipar un sistema de control.

2.3. Sistemas distribuidos y Middleware

Un sistema distribuido es aquel que utiliza múltiples recursos computacionales (o nodos) para lograr un objetivo común. Usualmente esto requiere comunicación entre todos los nodos, lo que da cabida a los middlewares.
Middleware es el término utilizado para referirse a software que permite o facilita la comunicación entre procesos, conectados usualmente a través de redes.
En el caso de observatorios astronómicos, estos a menudo se ven haciendo uso de múltiples tecnologías y sistemas que frecuentemente no han sido diseñados para funcionar en conjunto. Estos son buenos ejemplos de sistemas distribuidos. ALMA (Atacama Large Millimeter/Submillimeter Array) es uno de estos ejemplos, utilizando ACS (ALMA Common Software) que a su vez está construido a base de CORBA [3].

2.3.1. ZeroC Ice

Ice es otro ejemplo de middleware y uno central para el desarrollo de este trabajo. Su filosofía de diseño orientado a objetos permite crear interacciones a través de la red usando un paradigma familiar para la mayoría de los programadores.
La ideología principal para trabajar con Ice se basa en modelar las funcionalidades de un sistema como *clientes* y *servidores*, siendo los clientes aquellos que solicitan de un servicio del servidor, y los servidores son aquellos que prestan dicho servicio al cliente. Vale notar que estos roles no necesariamente han de ser fijos para cada parte de la aplicación, pudiendo intercambiarlos tanto como sea necesario.
Los servidores alojan *objetos Ice*. Estos son las entidades responsables de responder a las solicitudes de los clientes. Los objetos Ice poseen una o más Interfaces. Una Interfaz es una colección de operaciones soportadas por el objeto. Un cliente puede realizar solicitudes al

invocar estas operaciones.
Por otro lado, los clientes deben crear un *proxy* para comunicarse con los objetos Ice. Un proxy es una representación local del (posiblemente remoto) objeto Ice.

La creación de los objetos y proxies es manejada por código generado en base a una definición *Slice*. Slice (Specification Language for Ice) es un lenguaje que permite abstraer la definición de las interfaces de los objetos de sus implementaciones. Estas definiciones se traducen al lenguaje requerido por el sistema usando un compilador específico para cada lenguaje (slice2cpp, slice2java, slice2py, slice2matlab, etc.), permitiendo generar código (usado para el proxy por parte del cliente y como esqueleto del objeto Ice por parte del servidor) que permite la integración de la implementación del objeto con los servicios que provee la aplicación. (véase Figura 6)

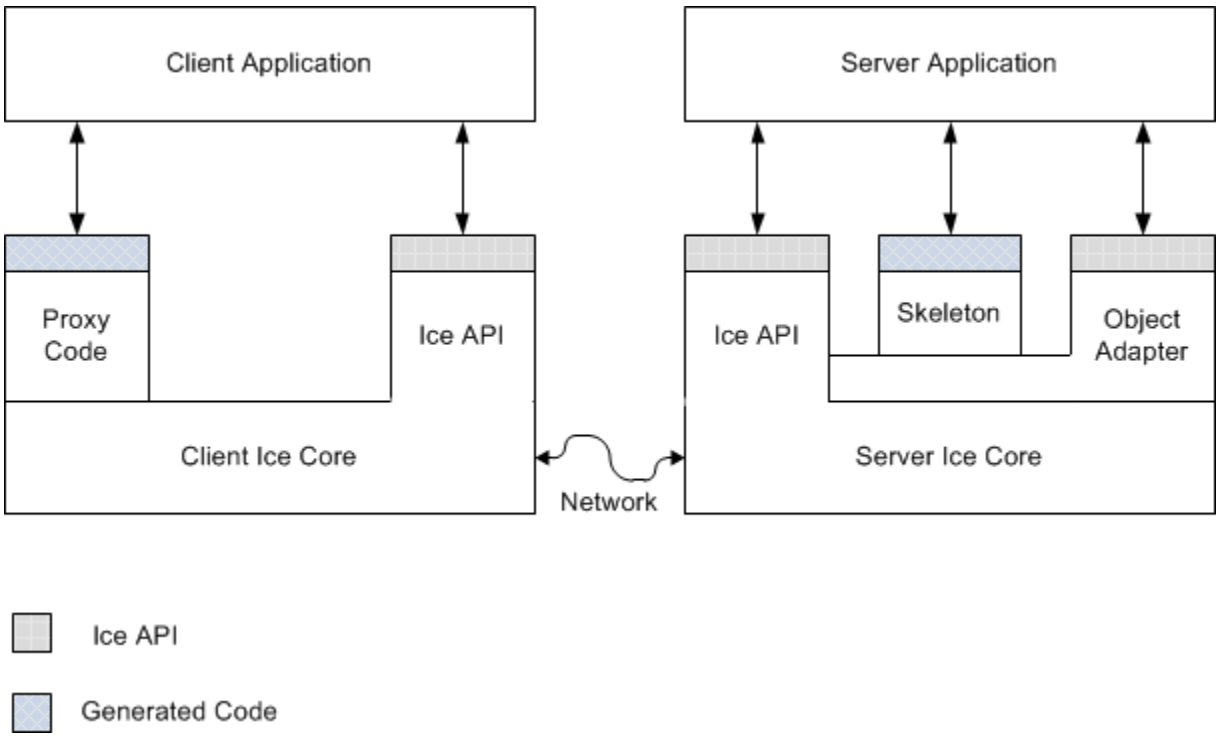

Figura 6: Estructura Cliente-Servidor de Ice. (ZeroC [21])

En la Figura 6 se observa un diagrama de la arquitectura Cliente-Servidor. En esta podemos observar los principales componentes:

- El *Ice Core* (núcleo) maneja la comunicación entre cliente y servidor durante la ejecución. Las librerías de Ice proveen de este núcleo para su uso en ambas partes.
- El acceso al Ice Core se realiza a través de la *Ice API*, es idéntica para los clientes y servidores y permite realizar tareas administrativas como iniciar o finalizar la ejecución de Ice.
- El *Código Proxy* es generado por nuestra definición Slice y permite al cliente invocar operaciones en un objeto remoto, además de la serialización y deserialización de datos requeridos por estas.

- El *Skeleton* es la contraparte del proxy, usado por el servidor. Permite la recepción de las solicitudes además de la serialización y deserialización de datos.
- El *Object Adapter* es una parte únicamente usada por los servidores y una de sus funciones es asistir al cliente en la creación del proxy, entre otras que menos pertinentes para este trabajo. (véase la documentación de Ice para información más detallada [21])

Ice ha sido considerado como una de las opciones en múltiples estudios [2, 3, 4] evaluando alternativas de middleware en software de control de telescopios además sugerido por la dirección del proyecto, por lo que fue elegido como el framework de soporte para este trabajo.

3. Descripción de la Propuesta

Para el modelado del nuevo sistema de control, debemos encontrar qué elementos de la arquitectura del CSO poseen análogos en el observatorio Wenulafken y reemplazar a CORBA (referido como ORB en la Figura 7) por nuestro middleware elegido, Ice.
Gracias al trabajo de B. Andler [20], tenemos un entendimiento del sistema de control del CSO. El observatorio de Caltech contaba con un gran número de sistemas y servicios que no tienen paralelos en el observatorio de Wenulafken, sin embargo se describen tres que podemos modelar. Tres servicios que, podemos presumir, son esenciales para cualquier telescopio. Estos son el “UIP” (User Interface Program), “Orrery Server” y “eSMA Telescope Server” (véase Figura 7), que son responsables de permitir al usuario controlar el sistema, indicar la posición de fuentes (objetos de interés) en el cielo y del control del telescopio, respectivamente.

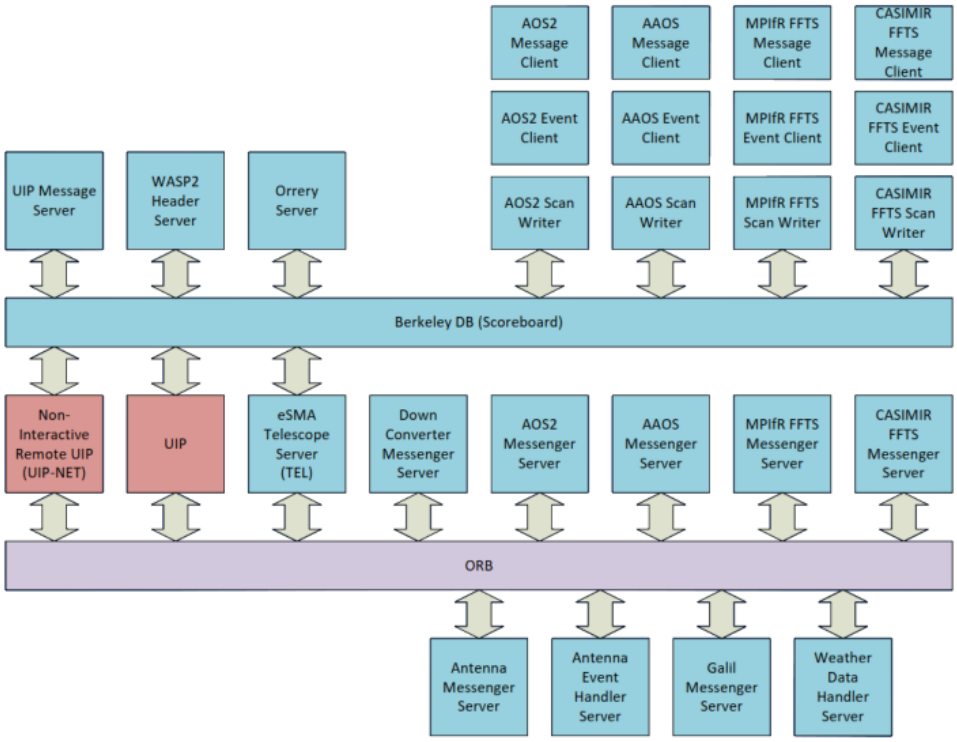

Figura 7: Componentes de software en el sistema de CSO (B. Andler [20])

En el sistema propuesto en este trabajo, Stellarium será utilizado simultáneamente como Orrery Server y UIP. Stellarium es un planetario de código abierto, soportado en múltiples plataformas y con una gran comunidad soportándolo. Por otro lado, el control del telescopio será manejado con una implementación personalizada del protocolo de comunicación de los rotores, con Ice reemplazando a CORBA y permitiendo la comunicación entre los componentes. La arquitectura del sistema propuesto se explica en más detalle en el siguiente capítulo.

El sistema propuesto se pondrá a prueba usándolo para actividades del curso de Fundamentos de Instrumentación Astronómica impartido por la facultad de Astronomía de la UdeC.

4. Detalle de la propuesta

4.1. Descripción de la arquitectura actual

El radiotelescopio del observatorio Wenulafken utiliza EME System para manejar el trackeo de objetos celestes además de la comunicación con los controladores de los rotores utilizando el protocolo Yaesu GS-232.
Vale notar que este tiene la limitación de sólo poder transmitir ángulos enteros entre 0° y 360°.
La información capturada por el receptor es recibida por una tarjeta SDR que se comunica con un software de radio (Airspy o GNURadio). Un diagrama del sistema de control se muestra en la Figura 8.

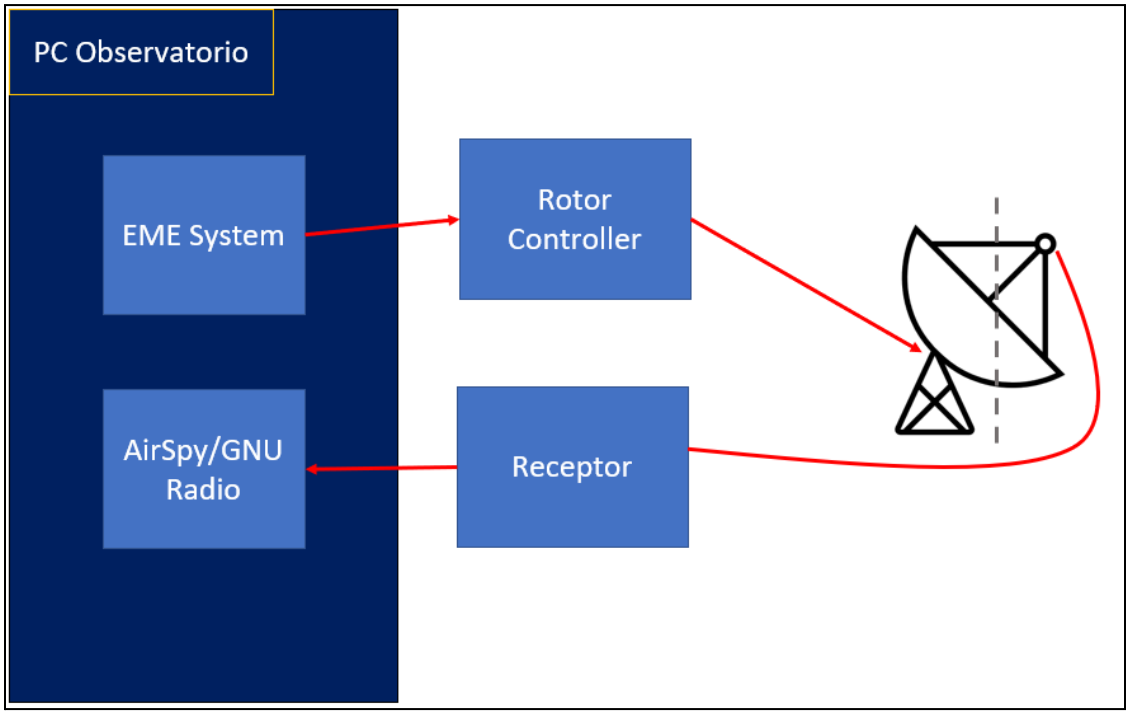

Figura 8: Diagrama del sistema de control inicial del observatorio Wenulafken

4.2. Requerimientos para el nuevo sistema

El proyecto busca modelar funciones del radiotelescopio del observatorio Wenulafken que tengan análogos en radiotelescopios de mayor escala (basándose en CSO y proyectándose a LCT) y coordinar estas funciones utilizando el framework de Ice como middleware.
Como descrito en el capítulo anterior, las principales funciones modeladas son la capacidad de obtener la posición de un objeto celeste en tiempo real (o seguimiento) y la capacidad de controlar los rotores del radiotelescopio utilizando esta información. Al mismo tiempo, se espera que el desarrollo del prototipo, en particular la implementación del framework de Ice, sea simple, versátil y mantenible.

4.3. Características deseables del nuevo sistema

Por otra parte, se desea que el prototipo sea capaz de satisfacer el o los casos de uso actualmente encontrados en el uso rutinario del radiotelescopio de Wenulafken, con la esperanza de que usuarios de este observatorio puedan poner a prueba el sistema para ayudar a evaluar su desempeño y usabilidad.
En conversaciones con el equipo de CePIA (Centro Para la Instrumentación Astronomica) se determinó que el principal caso de uso es el seguimiento de un objeto celeste (o fuente). Ademas se sugirieron funcionalidades adicionales que el equipo desearía tener en el observatorio:

- Capaz de realizar trayectorias de escaneo (e.g. lissajous trajectory) y modos alternos de observación (e.g. observación “on & off”)
- Capaz de acelerar y desacelerar suavemente el movimiento de los rotores para reducir vibraciones durante observación
- Capaz de realizar movimientos con precisión <1°
- Contar con una interfaz de usuario más usable

4.4. Sistema propuesto

El sistema propuesto ha sido diseñado como un sistema distribuido, recordando lo mencionado en el punto 3, donde separamos cada función del telescopio en distintos módulos (o servicios) capaces de comunicarse con una aplicación central a través de Ice.

Las principales funciones del sistema modeladas son el módulo de “Tracking” o “Seguimiento”, el módulo de “Rotor Control” y la comunicación entre ellos.
El módulo de Tracking posee la información de la posición de los objetos celestes que se desean seguir. Para este caso se utilizó Stellarium para obtener dicha información. Su amplia base de datos de objetos estelares y amigable interfaz de usuario y soporte para plugins lo hacen un buen candidato para este prototipo.
El módulo de Rotor Control envía instrucciones a los rotores de la montura del radiotelescopio.
La Aplicación Central (referida como “Central App” en las Figuras 8a y 8b) provee una interfaz básica, además de coordinar y comunicar a los distintos servicios para cumplir las tareas requeridas por el usuario.

En las Figuras 8a y 8b se ilustra conceptualmente el sistema propuesto. La idea principal es crear un sistema capaz de funcionar en sistemas distribuidos, donde distintos servicios pueden ser provistos por distintos equipos que no han sido diseñados para trabajar juntos. En particular, la Figura 9a muestra el caso de uso más frecuente que este sistema tiene en el observatorio Wenulafken, mientras que la Figura 9b ilustra el funcionamiento del mismo sistema distribuido en múltiples (dos) ordenadores. Mientras se provea de la IP y los puertos a través de los cuales cada módulo proveerá sus servicios, cualquier distribución es funcional.

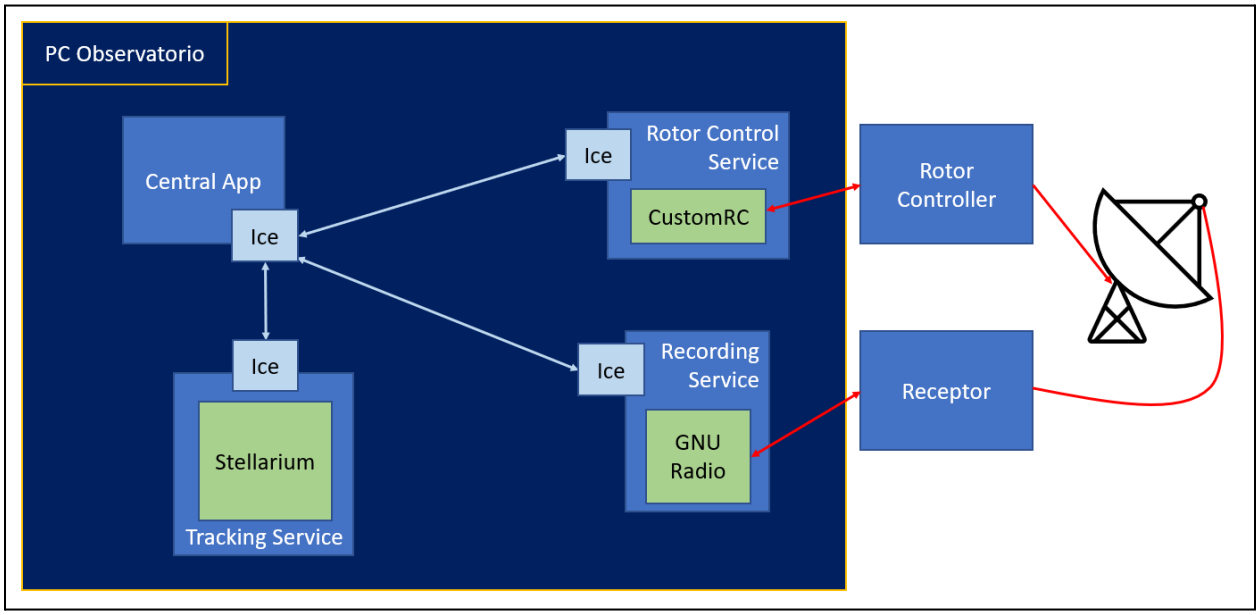

Figura 9a: Diagrama del sistema propuesto, con múltiples módulos funcionando en un único ordenador.

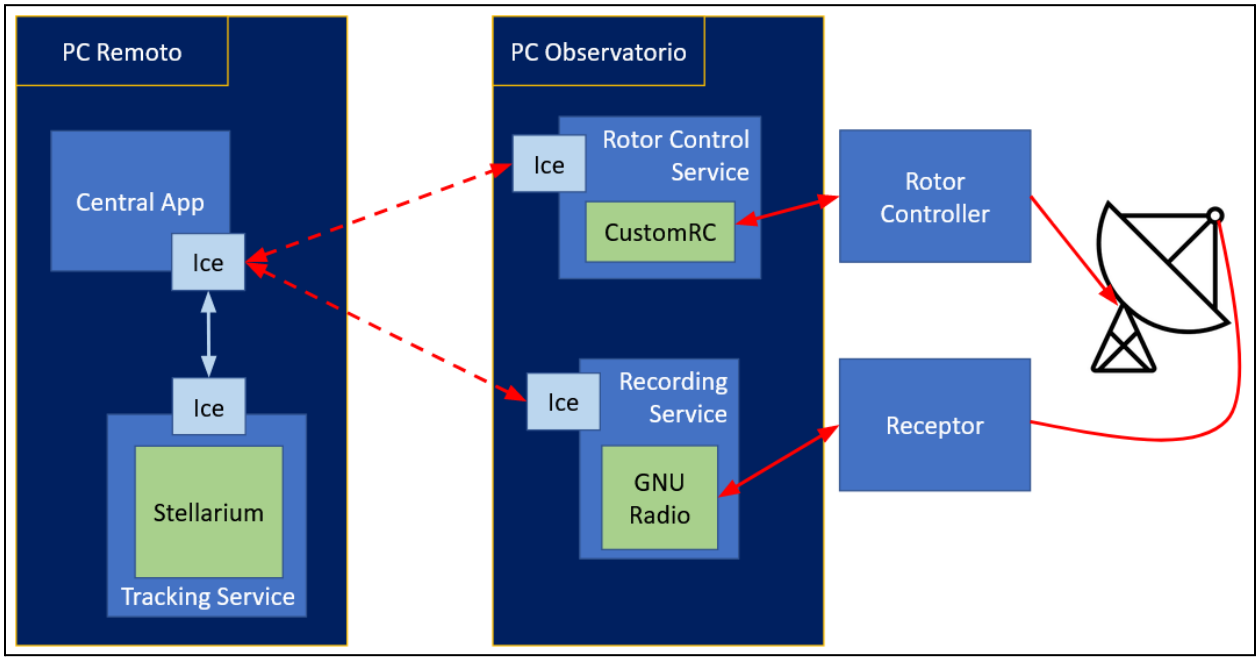

Figura 9b: Comunicación de los módulos distribuidos en múltiples ordenadores facilitada por Ice.

4.5. Implementación

Nótese que lo descrito en esta sección no es un instructivo de como crear una aplicación con Ice y se han dejado de lado algunos pasos y líneas innecesarias para la comprensión (pero requeridas para el correcto funcionamiento) de esta implementación. Para más detalles sobre cómo escribir una aplicación de Ice, sugiero comenzar con el tutorial provisto en la documentación oficial [15].

4.5.1. Módulo de Tracking

El primer paso para construir una aplicación basada en Ice es escribir una definición Slice incluyendo las interfaces usadas por dicha aplicación. En el caso del servicio de Tracking, sólo se requiere que éste entregue las coordenadas Alt/Az de un objeto (o "source"). Así, la definición slice (`Tracker.ice`) se describe como a continuación:

```
# Filename: Tracking.ice

module TrackingModule {
    interface Tracker {
        string getAziAlt(string source);
    };
};
```

Con esta, cuando la Central App desee conocer la posición de un objeto, solo ha de invocar la función `getAziAlt()` del servidor de Tracking.

El servicio de Tracking utiliza Stellarium para conocer las posiciones de un gran catálogo de objetos celestes. Para esto, hacemos uso de un plugin incluido en Stellarium (aunque no activado por defecto) llamado "Remote Control". Este permite controlar Stellarium usando una interfaz de servidor web. Utilizandolo, la clase `TrackerI` implementa la interfaz definida en `Tracker.ice` como se muestra a continuación:

```
# Filename: Tracking_Server.py

class TrackerI(TrackingModule.Tracker):
    def getAziAlt(self, source, current=None):
        if source != '':
            source = f'name={source}&'
        req =
requests.get(f'http://localhost:{STEL_RC_PORT}/api/objects/info?{source}format=
```

```
json')
        if req.ok:
            data = req.json()
            return f'{data["azimuth"]:.10f} {data["altitude"]:.10f}
{data["name"]}'
        else:
            return 'NoSourceSelected'

status = 0
ic = None
try:
    ic = Ice.initialize(sys.argv)
    adapter = ic.createObjectAdapterWithEndpoints("SimpleTrackerAdapter",
f"default -p {TRACKER_PORT}")
    object = TrackerI()
    adapter.add(object, ic.stringToIdentity("SimpleTracker"))
    adapter.activate()
    ic.waitForShutdown()
except:
    traceback.print_exc()
    status = 1

if ic:
    # Clean up code for exit procedure
sys.exit(status)
```

`TrackerI.getAziAlt()` crea una solicitud HTTP para comunicarse con el plugin de Remote Control, incluyendo en esta el nombre del objeto de interés. En caso de no especificar un nombre, la solicitud retornará la información de cualquier objeto que esté seleccionado en Stellarium. De la respuesta de esta solicitud (recibida como un objeto JSON) se extrae la información de interés (coordenadas Alt/Az y nombre del objeto), se concatena y se retorna en una única string.
Además de implementar la interfaz definida en `Tracker.ice`, `Tracking_Server.py` también crea los objetos y adaptadores requeridos para inicializar el servidor (en el bloque try/except).

4.5.2. Módulo de Rotor Control

Similarmente, comenzamos con la definición slice de la interfaz. Esta vez, las funciones requeridas de este módulo son: mover los rotores a coordenadas especificadas e interrumpir el movimiento de los rotores (particularmente útil por motivos de seguridad). Con esto, la definición de la interfaz se describe en `RotorController.ice` como sigue.

```
# Filename: RotorController.ice
```

```
module RotorModule {
    interface Rotor {
        void gotoAziAlt(float azi, float alt);
        void stop();
    };
};
```

La comunicación con el controlador de los rotores se realiza a través de un puerto COM (RS-232) y utilizando alguno de los protocolos de comunicación soportados por el controlador. Estos protocolos son Yaesu GS-232A y Rot2Prog. El protocolo Rot2Prog tiene la ventaja de soportar ángulos con décimas de grado de precisión, a diferencia de Yaesu, que solo permite el envío de números enteros. Sin embargo, se escogió utilizar el protocolo Yaesu para esta iteración del prototipo por dos motivos. Primero, el protocolo Yaesu es utilizado por EME System, el sistema de control previamente utilizado en el observatorio Wenulafken, dejando antecedente de que sus capacidades son suficientes, y segundo, no se ha determinado si las ventajas de poder mover los rotores a pasos más pequeños superan las desventajas que vienen con las vibraciones causadas por necesitar realizar un mayor y más frecuente número de pasos. En `Rotor_Server.py` se implementa la interfaz previamente definida en slice.

```
# Filename: Rotor_Server.py

class RotorI(RotorModule.Rotor):
    def gotoAziAlt(self, azi, alt, current=None) -> None:
        azi = int(azi)
        alt = int(alt)
        command = f'W{azi:03d} {alt:03d}\r\n'
        ser.reset_input_buffer()
        ser.write(command.encode('UTF-8'))
        response = ser.readline().decode().strip()

    def stop(self, current=None) -> None:
        command = f'S\r\n'
        ser.reset_input_buffer()
        ser.write(command.encode('UTF-8'))
        pass

status = 0
ic = None
try:
    ser = serial.Serial('COM6', 9600, timeout=0)
    # Generic Ice Server Initialization
except:
```

```
    traceback.print_exc()
    status = 1

if ic:
    # Clean up code for exit procedure

sys.exit(status)
```

Los comandos se construyen como strings en conformidad con el protocolo Yaesu, se codifican al estándar UTF-8 y se envían a través de un puerto COM. La inicialización del servidor es análoga a la mostrada en `Tracking_Server.py`, con la adición de necesitar abrir una conexión serial con el puerto COM mencionado.

4.5.3. Central App

Finalmente, la Central App () es el módulo responsable de coordinar las tareas necesarias para cumplir las tareas requeridas que, en este caso, es mantener al radiotelescopio apuntando a un objeto que se desee observar. Vale notar que los siguientes segmentos de código son una versión altamente resumida y enfocada a demostrar la ejecución de la tarea de seguimiento, dejando de lado el manejo de la interfaz de usuario y otros detalles para facilitar el uso del sistema.

```
# Filename: Client.py

main():
    status = 0
    IC = None
    try:
        IC = Ice.initialize(sys.argv)

        # Creating Tracker Proxy
        tracker_base_prx = IC.stringToProxy(f"SimpleTracker:default -p
{MyEnv.TRACKER_PORT}")
        TRACKER_PRX = TrackingModule.TrackerPrx.checkedCast(tracker_base_prx)
        if not TRACKER_PRX:
            raise RuntimeError("Invalid tracker proxy")
        # Creating Rotor Proxy
        rotor_base_prx = IC.stringToProxy(f"SimpleRotor:default -p
{MyEnv.ROTOR_PORT}")
        ROTOR_PRX = RotorModule.RotorPrx.checkedCast(rotor_base_prx)
        if not ROTOR_PRX:
            raise RuntimeError("Invalid rotor proxy")

        trackingLoop()
```

```
    except:
        traceback.print_exc()
        status = 1

    if IC:
        # Clean up code for exit procedure
    sys.exit(status)
```

En `main()` se inicializa el comunicador de Ice y se crean los proxies (`TRACKER_PRX` y `ROTOR_PRX`) para los objetos definidos en los servidores de tracking y rotor control. Estos proxies permiten a la Central App interactuar con las funciones provistas por los objetos remotos como si existieran de forma local. Luego de esto podemos iniciar la secuencia de tracking definida en `trackingLoop()`.

```
# Filename: Client.py

def trackingLoop():
    source = ""
    while True:
        try:
            res = TRACKER_PRX.getAziAlt(source)
            if res != "NoSourceSelected":
                (azi, alt) , target_name = parseAziAlt(res)
                ROTOR_PRX.gotoAziAlt(azi, alt)
            else:
                ROTOR_PRX.stop()

            time.sleep(1) # arbitrary timer
        except Ice.UnknownException:
            print(f"[{source}] doesn't exist")
            break
```

El ciclo de seguimiento de un objeto consta de los siguientes pasos:

1. Solicitar la posición del objeto de interés al servicio de Tracking.
2. Parsear la respuesta recibida para extraer las coordenadas del objeto de interés.
3. Solicitar al servicio de Rotor Control el movimiento de los rotores a las coordenadas obtenidas.
4. En caso de recibir `NoSourceSelected` como respuesta, se comanda la interrupción del movimiento de los rotores.

La función `parseAziAlt()` es la responsable de parsear la string recibida del servicio de Tracking. Un tiempo de `sleep()` arbitrario (de 1 segundo) se incluye para no realizar demasiadas solicitudes por segundo considerando que la gran mayoría de objetos no se mueve a mucho menos de 1°/s (recordando la limitación del protocolo Yaesu).

La función `getAziAlt()` (definida en `Tracking_Server.py`) acepta una string source como argumento. Esta es para permitir buscar un objeto por nombre directamente a través de consola en vez de usar Stellarium en caso de que así lo desee el usuario, pero esta posibilidad no está soportada en esta versión del prototipo debido a su poco valor. En su lugar se entrega source como una string vacía, obteniendo información de cualquier objeto seleccionado en Stellarium.

Las Figuras 10a y 10b muestran screenshots del sistema en uso. En ellas se puede ver la aplicación de Stellarium y una consola de comando usada para ejecutar el sistema de control propuesto. Una vez iniciado el loop de seguimiento, “trackear” un objeto es tan simple como hacer click sobre él en Stellarium para seleccionarlo. Seguir un nuevo objeto se puede hacer en cualquier momento seleccionando un objeto distinto. Detener el seguimiento puede hacerse des-seleccionando el objeto en Stellarium haciendo click derecho en cualquier parte del cielo, o presionando Enter en la consola de comando para salir del loop de trackeo.

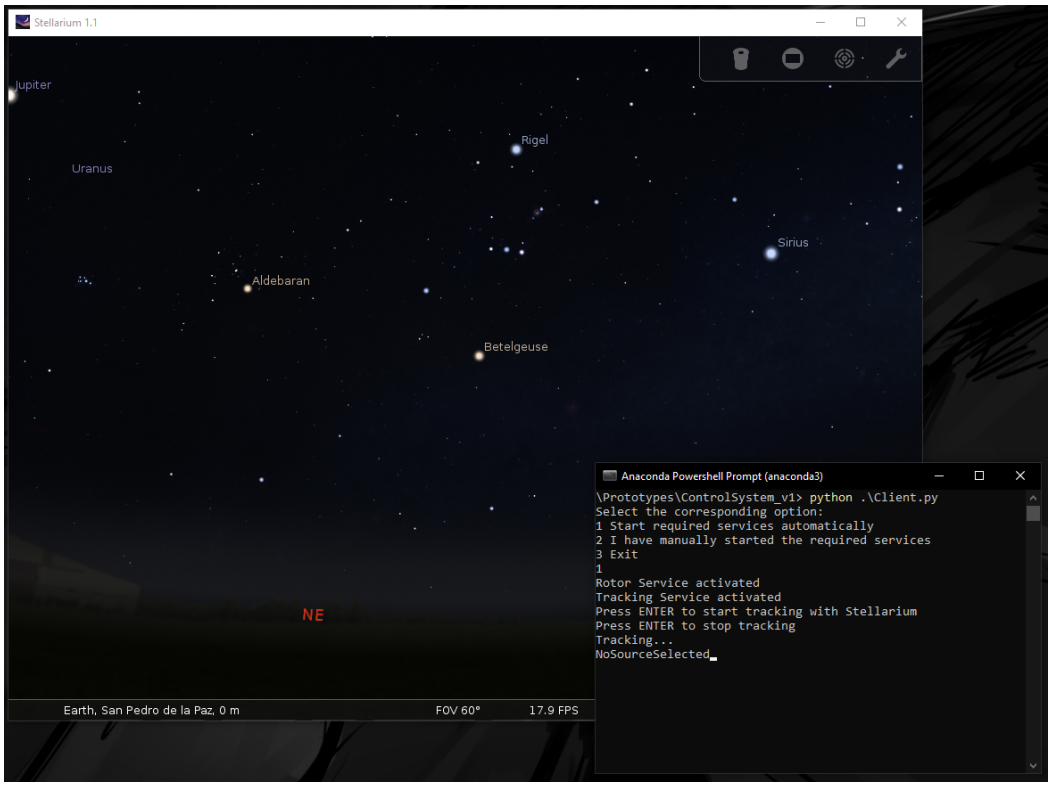

Figura 10a: Stellarium y consola de comandos. Ningún objeto seleccionado.

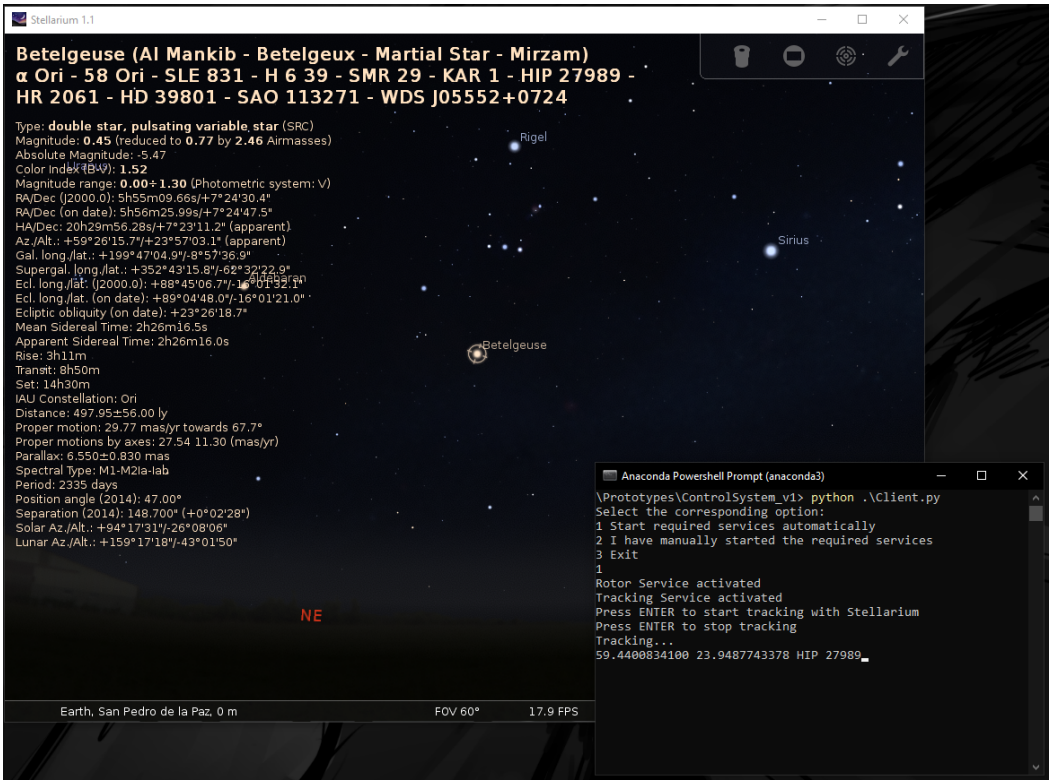

Figura 10b: Stellarium y consola de comandos, habiendo seleccionado un objeto.

5. Evaluación de la propuesta

La implementación del sistema propuesto es fácilmente expandible. Además, permite alojar cada uno de los módulos/servicios en servidores separados y heterogéneos con mínimos cambios gracias al lenguaje de definición de interfaz slice.

La usabilidad y correctitud (o precisión) del sistema de control fue puesto a prueba haciendo uso de él durante actividades del curso de Fundamentos de Instrumentación Astronómica (FIA), dirigido por investigadores del equipo de CePIA, Dr. Rodrigo Reeves como profesor y Brian Molina como ayudante. Para juzgar la precisión de apuntamiento utilizando los principios introducidos en 2.1.3. Los datos obtenidos fueron procesados y comparados con bases de datos de generación de perfiles [7].

Análisis exacto de la precisión de apuntamiento no fue realizado durante las pruebas debido a que estaba fuera del alcance del curso de FIA, pero se reportó que “fue suficiente” para las actividades de este. Pruebas gruesas de apuntamiento se realizaron con el básico método de observar la proyección de la sombra, proyectada por el sol, de la antena sobre el plato del telescopio y se confirmó que el telescopio apuntaba “suficientemente bien” al sol. El criterio para decidir esto es confirmar que la sombra de la bocina cae dentro del disco central del plato reflector.

6. Conclusiones

El objetivo general de esta memoria de título es evaluar la adecuación de Ice como framework de soporte para sistemas de control de telescopios, usando el observatorio de Wenulafken como banco de pruebas. Como respuesta a esta pregunta solo podemos decir que el radiotelescopio del anterior observatorio no es adecuado para poder extrapolar el uso de Ice en sistemas más complejos. El radiotelescopio en Wenulafken es un telescopio actualmente con propósitos meramente didácticos, con un az de observación muy ancho (~5°) y muy pocos sistemas que coordinar. Como tal, no ofrece el mejor escenario para poner a prueba las verdaderas capacidades que un framework como Ice ofrece. Sin embargo, sí podemos decir que Ice fué más que capaz de cumplir con los requerimientos del observatorio sin dificultar el desarrollo del sistema, además de impulsar una arquitectura distribuida fácilmente expandible. Combinando esto con estudios recientes (como [3, 4]) -donde se incluye a Ice como uno de las mejores alternativas- podemos, como mínimo, calificar a este proyecto como un prototipo exitoso con Ice como middleware en un sistema de control.

7. Trabajos Futuros

7.1. Integrar nuevos módulos/servicios

Probablemente, el mejor camino para expandir este trabajo es incluir nuevos módulos y/o servicios para el sistema de control que permitan evaluar más de las capacidades ofrecidas por Ice. Actualmente la integración y comunicación con un software de procesamiento de señal es una de las pocas posibilidades para el observatorio como se encuentra actualmente.Un candidato para dicho software de procesamiento de señal es el proyecto de código abierto, GNURadio. El diseño propuesto en el punto 4.4 se incluye un servicio de grabación (o “recording”) utilizando este software, sin embargo, se requiere de la experticia del equipo de astronomía para poder realizar el preprocesamiento de la señal con GNURadio antes de que integrarlo al sistema sea de valor.

7.2. Actualización de Firmware

El controlador de los rotores debería soportar otro protocolo de comunicación, Rot2Prog, y este posee ciertas funcionalidades (como el control en tiempo real del poder de los rotores) que no funcionan adecuadamente con el controlador actual. Contacto con el servicio de soporte de RF HAMDESIGN (el proveedor del kit de la antena usada en el observatorio) sugirió actualizar la versión del firmware del controlador.

7.3. Suavizar el movimiento de los rotores

El inicio y detención de cada movimiento es brusco y genera vibraciones en el telescopio. El protocolo Rot2Prog cuenta con funciones para manipular la potencia de los motores, lo que

permitiría suavizar los movimientos. Esta función no está funcionando adecuadamente, sin embargo, y puede requerir la actualización de firmware. La implementación del suavizado de movimientos, tampoco es trivial. Es más, se sospecha que la complejidad de suavizar los movimientos mientras se mantiene el telescopio siguiendo al objetivo es digno de un trabajo por sí solo.

7.4. Permitir el seguimiento de un punto celeste arbitrario

Una funcionalidad esperable de un sistema de control de telescopios es la habilidad de seguir un punto arbitrario en el cielo, es decir, proveer al sistema de un par de coordenadas RA/Dec y que este sea capaz de mantener al telescopio apuntando hacia dicha posición mientras se desplaza por el cielo, de manera precisa. Si bien esto no ha sido incluido en la implementación actual del sistema propuesto, una solución alterna es posible aprovechando la gran densidad de objetos trackeables con Stellarium y el gran beam de observación del radiotelescopio de Wenulafken. Estas características implican que para prácticamente cualquier punto del cielo que se desee observar, existirá un objeto en la base de datos de Stellarium suficientemente cerca para que al trackear dicho objeto, el punto deseado esté dentro del haz de observación. Esta, sin embargo, es una solución inexacta y sin garantías, por lo que una implementación apropiada de conversión de coordenadas RA/Dec a Alt/Az es deseable.

8. Referencias

[1] Z. Wang et.al. “RACS2: A Framework of Remote Autonomous Control System for Telescope Observation and its Application”. 2022.

[2] S. Shumko. Ice Middleware in the New Solar Telescope's Telescope Control System. 2009.

[3] R. Sanhueza. "EVALUATION OF DISTRIBUTED-SYSTEM TECHNOLOGIES FOR ALMA COMMON SOFTWARE". 2018.

[4] A. Dworak, P. Charrue, F. Ehm, W. Sliwinski, M. Sobczak. "Middleware Trends And Market Leaders 2011". 13th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2011). Vol. C111010. 2011.

[5] E. White et.al. “Green Bank Telescope: Overview and analysis of metrology systems and pointing performance”.2021.

[6] K. O'Neil. "Single dish calibration techniques at radio wavelengths." arXiv preprint astro-ph/0203001 2002.

[7] Argelander Institut für Astronomie. “HI Profile Search". Available: https://www.astro.uni-bonn.de/hisurvey/AllSky_profiles/index.php (Accessed 10-Aug-2023)

[8] M. Wilkinson, J. Kennewell. "HYDROGEN-LINE OBSERVATIONS OF THE GALAXY AND THE MAGELLANIC CLOUDS". Space Academy. Available: https://www.spaceacademy.net.au/spacelab/projects/hlineobs/hlineobs.htm. (Accessed: 10-Aug-2023)

[9] M. Olmo, R. Nave. “La Línea del Hidrógeno de 21-cm”. HyperPhysics. Available: http://hyperphysics.phy-astr.gsu.edu/hbasees/quantum/h21.html (Accessed 10-Aug-2023)

[10] "Horizontal coordinate system". Wikipedia. Available: https://en.wikipedia.org/wiki/Horizontal_coordinate_system. (Accessed 10-Aug-2023)

[11] J. Patris. "Preparing astronomical observations and observing with OHP facilities". EPJ Web of Conferences. 2010.

[12] T. Westmeier. “Tools - Useful equations for radio astronomy.” Homepage of Tobias Westmeier, Available: https://www.atnf.csiro.au/people/Tobias.Westmeier/tools_hihelpers.php. (Accessed 10-Aug-2023)

[13] R. Hart. "Radio Astronomy" Hardhack.org. 2008. Available: https://www.hardhack.org.au/radio_astronomy. (Accessed 17-Aug-2023)

[14] NASA."Basics of Space Flight". Solar System Exploration: NASA Science. Available: https://solarsystem.nasa.gov/basics/chapter2-2/. (Accessed 10-Aug-2023)

[15] ZeroC. "Hello World Application - Ice". ZeroC Documentation. Available: https://doc.zeroc.com/ice/3.6/hello-world-application. (Accessed 20-Aug-2023)

[16] R.B. Leighton. "Final Technical Report" (PDF). Caltech Library. 1977.

[17] B. Andler. "CSO Control System: Software understanding update". 2022.

[18] J.C. Guzman."PRELIMINARY DESIGN OF THE AUSTRALIAN SKA PATHFINDER (ASKAP) TELESCOPE CONTROL SYSTEM". 12th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2009). 2009.

[19] ZeroC. "Ice". ZeroC Ice. Available: https://zeroc.com/products/ice. (Accessed 21-Aug-2023))

[20] B. Andler. "CSO Control System: Software understanding update" (PDF). 2022. Available: https://tinyurl.com/bddz5waz. (Accessed 21-Aug-2023)

[21] ZeroC."Documentation for Ice". ZeroC.com. Available: https://doc.zeroc.com/ice/latest. (Accessed 31-Aug-2023))