Obtención de datos críticos de forma remota para optimización y automatización de mantenimiento y supervisión para sistemas *VHF Omnidirectional Range* de navegación aérea

Luis José Archila Madrid

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Obtención de datos críticos de forma remota para optimización y automatización de mantenimiento y supervisión para sistemas *VHF Omnidirectional Range* de navegación aérea

Trabajo de graduación presentado por Luis José Archila Madrid para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Obtención de datos críticos de forma remota para optimización y automatización de mantenimiento y supervisión para sistemas *VHF Omnidirectional Range* de navegación aérea

Trabajo de graduación presentado por Luis José Archila Madrid para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2024

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

El presente trabajo se desarrolló con el propósito de sentar las bases para la automatización y optimización de los procesos internos de COCESNA, guiado por la necesidad de adaptarse a la nueva era tecnológica. A través de este proyecto, se busca liberar al talento humano de tareas repetitivas, permitiéndole enfocarse en el análisis profundo y la mejora continua de los procesos, así como en la mejora de la calidad de los servicios ofrecidos por la corporación asegurando una mayor eficiencia y excelencia operativa.

A lo largo de este trabajo, he tenido el privilegio de contar con el incondicional apoyo de diversas personas e instituciones. En primer lugar, quisiera expresar mi más profundo agradecimiento a mis padres, quienes han sido mi mayor fuente de motivación y apoyo en todos los aspectos de la vida. Ellos me inculcaron desde siempre que, con perseverancia, determinación y esfuerzo, todo es posible. No solo me enseñaron esos valores, sino que, gracias a su respaldo constante, han hecho posible que llegue hasta este momento. También agradezco especialmente a mi familia y amigos, quienes me han apoyado a lo largo de mi carrera y durante el desarrollo de este trabajo.

También expreso mi agradecimiento a COCESNA por permitirme ser parte de este importante cambio y por confiarme un proyecto con una visión tan ambiciosa. Agradezco a todo el personal que me brindó su apoyo y facilitó el desarrollo de este trabajo, en especial al Lic. Manuel Polanco, Ing. Rony Mon-

tenegro, Lic. Guillermo Cruz, Ing. Gerberth Mancia e Ing. Raúl Calo, por su invaluable colaboración. Asimismo, deseo agradecer a Irtri Simone, cuya asesoría fue fundamental en el desarrollo del software y en el enlace con las radioayudas de COCESNA.

Finalmete, expreso mi gratitud a la Universidad Del Valle de Guatemala institución que me ha brindado las herramientas académicas y profesionales necesarias para afrontar este desafío. Asimismo, agradezco a todos los profesores que con su dedicación y conocimientos me han guiado a lo largo de este camino, fomentando en mí el pensamiento crítico, la innovación y el desarrollo integral.

Este proyecto ha sido una experiencia enriquecedora tanto en el ámbito académico como personal. Espero que los resultados obtenidos contribuyan de manera significativa a la implementación de nuevas tecnologías en COCESNA, y que este documento sirva de referencia para futuros estudios.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este trabajo está enfocado en la automatización y optimización de los procesos de mantenimiento y supervisión para los sistemas de navegación aérea VOR (VHF Omnidirectional Range). El proyecto, desarrollado en colaboración con COCESNA, responde a la necesidad de modernizar los procesos internos de la organización, pasando de la recolección manual de datos a un enfoque de monitoreo remoto automatizado. Aprovechando tecnologías como el protocolo SNMP (Protocolo Simple de Administración de Red), PostgreSQL y Grafana, la solución propuesta permite el monitoreo en tiempo real y la visualización de datos, minimizando el error humano y mejorando la eficiencia operativa. La fase de implementación del estudio se centra en el sistema DVOR432 del Aeropuerto Internacional La Aurora como una iniciativa piloto, con el potencial de extenderse a otros equipos de navegación aérea. Este trabajo sienta las bases para la adaptación tecnológica de COCESNA, permitiendo que el personal se concentre en el análisis de datos y la mejora continua en lugar de tareas repetitivas, contribuyendo a la modernización y eficiencia de los servicios de navegación aérea en Centroamérica.

Abstract

This work aims at automating and optimizing the maintenance and supervision processes for VOR (VHF Omnidirectional Range) air navigation systems. The project, developed in collaboration with COCESNA, addresses the need for modernization within the organization's internal processes, shifting from traditional manual data collection to a remote, automated monitoring approach. Leveraging technologies such as SNMP (Protocolo Simple de Administración de Red) protocol, PostgreSQL, and Grafana, the proposed solution enables real-time monitoring and data visualization, minimizing human error and enhancing operational efficiency. The study's implementation phase focuses on the DVOR432 system at La Aurora International Airport as a pilot initiative, with the potential to extend to other air navigation equipment. This work sets the foundation for COCESNA's technological adaptation, allowing personnel to focus on data analysis and continuous improvement rather than repetitive tasks, contributing to the modernization and efficiency of air navigation services in Central America.

CAPÍTULO 1

Introducción

La aviación moderna depende cada vez más de sistemas de navegación precisos y confiables o Radioayudas, como el VOR (VHF Omnidirectional Range). Estos sistemas son esenciales para garantizar la seguridad y eficiencia en la gestión del tráfico aéreo, especialmente en regiones con alta densidad de vuelos. En este contexto, la Corporación Centroamericana de Servicios de Navegación Aérea (COCESNA) ha asumido el desafío de modernizar sus procesos de mantenimiento y supervisión para adaptarse a las demandas actuales.

El presente proyecto se centra en la automatización y optimización del monitoreo y generación de reportes de mantenimiento del equipo DVOR modelo 432, ubicado en el Aeropuerto Internacional La Aurora, mediante la implementación de una solución que permita la obtención y monitoreo remoto de datos críticos. Actualmente, las rutinas de mantenimiento y obtención de parámetros de estos equipos requieren la intervención manual de técnicos en campo, lo que aumenta el riesgo de errores y limita la eficiencia operativa. La propuesta planteada utiliza SNMP (Protocolo Simple de Administración de Red) y bases de datos en PostgreSQL para recopilar datos en tiempo real, que luego son visualizados a través de la herramienta Grafana, facilitando así la supervisión continua y la detección temprana de posibles fallas. Asimismo, se hace

uso de alertas en inmediatas para notificación de eventos de riesgo o parámetros fuera de tolerancias.

El objetivo de este trabajo es reducir la carga operativa y los costos asociados, permitiendo que el personal técnico se enfoque en tareas de mayor valor agregado, como el análisis y mejora de los sistemas de navegación. Este proyecto piloto sienta las bases para futuras implementaciones en otros equipos de COCESNA y contribuye a la transición hacia una estructura de mantenimiento más automatizada y eficiente.

CAPÍTULO 2

Antecedentes

2.1. Rutinas de mantenimiento establecidas por la unidad de inspección y evaluación de COCESNA

Las rutinas de mantenimiento son documentos cuyo propósito es registrar el estado de todos los equipos de aeronáutica en posesión de COCESNA. Estos documentos son completados manualmente por un técnico o ingeniero durante su turno, requiriendo la presencia física en el equipo para la toma de mediciones y/o parámetros. Es necesario verificar que todas las mediciones estén dentro de las tolerancias establecidas, anotar la información a mano y posteriormente llevar a cabo un análisis y diagnóstico del equipo con base en dicha información.

En el caso específico del DVOR432 del Aeropuerto Internacional La Aurora (y la mayoría de equipos de COCESNA), se llevan a cabo dos tipos de rutinas de mantenimiento: mensuales y trimestrales. La complejidad de estas rutinas y la cantidad de parámetros a medir son los elementos diferenciadores entre ambos tipos de rutinas.

COCESNA — RUTINA DE MANTENIMIENTO — Código: MTTO-GT-RNAV-001 — Edición: 010 — **DVOR 432, ALCATEL** — **MENSUAL** — FORMATO: **1**

GERENCIA DE ESTACIÓN: — SITIO: — SISTEMA / EQUIPO:

CORPORACIÓN CENTROAMERICANA DE SERVICIOS DE NAVEGACIÓN AÉREA

FECHA (dia-mes-año)	HORA (UTC) hora de inicio	No. de ODM / No. Evento	EQUIPO	MONITORES	Lecturas de Monitores: LSB Distorsión en la antena central	Nivel de RF del Dipolo 1	Azimuth Dipolo 1	Index FM Dipolo 1	Nivel de RF del Dipolo 2	Azimuth Dipolo 2	Index FM Dipolo 2	Nivel promedio de RF / Nivel Interno de RF	Profundidad de Modulación de 30Hz	Profundidad de Modulación de 9960Hz	Indice de Modulación 30 Hz FM	Profundidad de modulación de la identificación	Frecuencia de Portadora (MHz)	Frecuencia de la SBA (MHz)	Frecuencia de la SBB (MHz)	Distorsión en 9960Hz AM	Distorsión en la det. USB-LSB	Distorsión en la LSB detectada	Nivel de LSB	Nivel de USB/LSB	Presencial o Remota	FIRMA O INICIALES TÉCNICO
REFERENCIA (MANUAL)					ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	ADRACS	N/A	
VALOR NOMINAL					<3%	100 %	VREF 23.7°	16	100 %	VREF 233.7°	16	100 %	30%	30%	16	10% ó NORMAL	VPROG 116700.0	VPROG 116710.0	VPROG 116690.0	<45%	VREF 99.9%	VREF 3.5%	VREF 80.0	VREF 42.0	P o R	
TOLERANCIA					<3%	±15%	± 1°	± 1.	±15%	± 1°	± 1.	±15%	28 a 32	28 a 32	15 a 17	±1%	± 1KHz	± 1KHz	± 1KHz	N/A	N/A	<20%	+/- 4	+/- 4	N/A	
			1	1																						
				2																						
			2	1																						
				2																						
			1	1																						
				2																						
			2	1																						
				2																						
			1	1																						
				2																						
			2	1																						
				2																						
			1	1																						
				2																						
			2	1																						
				2																						
			1	1																						
				2																						
			2	1																						
				2																						
			1	1																						
				2																						
			2	1																						
				2																						
			1	1																						
				2																						
			2	1																						
				2																						
			1	1																						
				2																						
			2	1																						
				2																						

NOTA: ANOTAR LAS OBSERVACIONES EN LA PARTE DE ATRÁS

VREF: Valor de referencia de la Estacion valor personalizado por sitio

VPROG: Dato programado en el equipo y cuyo valor debe ser personalizado por sitio

** Actualizar según dato brindado por ADRACS, puede ser "NORMAL" o puede ser un dato númerico

*** Actualizar ya que dato cambia según sea Far Fiel o Next Field

Figura 1: Ejemplo de formato de rutinas de mantenimiento para el DVOR432 Alcatel [1]

2.2. Sistema de control y monitoreo (MCS)

El Grupo Thales se destaca como una empresa francesa especializada en soluciones digitales y el desarrollo de sistemas de información y servicios, dirigidos a los mercados aeroespacial, de defensa, transporte, seguridad y aeronáutico. Según la información disponible en su sitio web oficial [2], su primera incursión significativa en la industria aeronáutica data de 1965, con la implementación de su primer radar destinado al control del tráfico aéreo. Desde entonces, Thales ha enfocado sus esfuerzos no solo en la fabricación de equipos de radioayudas y radares, sino también en la concepción de interfaces y sistemas destinados al control y monitoreo de dichos dispositivos. Este sistema, conocido por su acrónimo en inglés MCS (*Monitoring and Control System*), se vale de una red privada

y del protocolo SNMP (Protocolo Simple de Administración de Red) de TCP/IP (Transmission Control Protocol/Internet Protocol) para ofrecer al usuario información detallada, en una interfaz gráfica, sobre el estado de cada equipo bajo su supervisión.

COCESNA hace uso del MCS desde la implementación de los equipos de Thales en Guatemala y Centroamérica, que se remonta a inicios de la déacada de los 2000. Desde entonces, se ha convertido en un pilar escencial en la parte de la calidad de servicio y monitoreo de la infraestrucutra de navegación de la corporación.

CAPÍTULO 3

Justificación

Como se menciona en el nombre de la Corporación Centroamericana de Servicios de Navegación Aérea (COCESNA), se trata de un proveedor de servicios por lo que la calidad de servicio y la satisfacción de los usuarios debe ser prioridad. Las rutinas de mantenimiento de los equipos de navegación de COCESNA desempeñan un papel sumamente importante en cuanto a la calidad del servicio ofrecido por la corporación. No obstante, el formato o plantilla de estas rutinas fue elaborado en el año 2003 por el Gerente Técnico, Gabriel Quiros. Con el transcurso de los años, dicho formato ha llegado a ser considerablemente obsoleto, y su eficacia depende en gran medida de la experiencia del técnico encargado de llevar a cabo las mediciones, lo que compromete la fidelidad de los informes generados. Como era de esperar, esta situación no siempre se maneja de manera óptima, resultando en ocasiones en la presencia de datos incoherentes o fuera de las tolerancias establecidas para el equipo, los cuales no son reportados para su corrección. Además, es importante resaltar que estas rutinas requieren un tiempo considerable para completarse, lo que a menudo conduce a que no se les otorgue la debida atención que merecen. Asimismo, se persigue que los técnicos dediquen su tiempo al análisis de datos, interpretación y a la implementación de mejoras en el rendimiento del equipo, en lugar de invertirlo en la recopilación de datos, buscando lograr un tra-

bajo más eficiente y de mayor calidad. Al generar estas rutinas de manera automática, obteniendo la información en tiempo real directamente desde el equipo de interés, se logra aumentar la fidelidad de estos reportes y se abren nuevas posibilidades de automatización y monitoreo.

El proyecto se enfocará inicialmente en el DVOR, sirviendo como una prueba piloto para la corporación. Su éxito sentará las bases para la implementación de sistemas similares en el resto de los equipos de navegación. Desde una perspectiva académica, el proyecto adquiere relevancia debido a la aplicación de conceptos abarcados a lo largo de la carrera de Ingeniería Electrónica, enfatizando áreas como la programación, las telecomunicaciones, la teoría electromagnética y la electrónica analógica y digital. Por ende, este proyecto posee un impacto significativo en el desarrollo tanto académico como profesional, además de demostrar un dominio integral de los temas mencionados anteriormente.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Reducir la carga laboral asociada al monitoreo y generación de informes de mantenimiento, promoviendo así una disminución significativa en los costos operativos y error humano; permitiendo una gestión más eficaz del tiempo y cumpliendo con las normativas de la unidad de inspección y evaluación de COCESNA.

4.2. Objetivos específicos

- Recopilar información de mantenimiento del equipo VOR del Aeropuerto Internacional La Aurora de manera remota mediante el protocolo SNMP y el Sistema de Control y Monitoreo (MCS (*Monitoring and Control System*)).
- Desarrollo de un programa para análisis de los datos recopilados de forma automática y generar informes periódicos que describan detalladamente el estado operativo del equipo VOR.
- Implementar alarmas para notificar de inmediato ante cualquier desviación de los parámetros establecidos en uno o

varios aspectos del equipo.

- Presentación de los datos de manera accesible para un análisis efectivo mediante correos electrónicos automáticos con destinatario a las personas interesadas dentro de la corporación.
- Realizar la instalación desde cero del sistema operativo Thalix OS y de una versión actualizada del software MCS para el monitoreo y control de sistemas de navegación aérea, asegurando un ambiente libre de configuraciones e implementaciones previas que puedan afectar el correcto funcionamiento y la escalabilidad del sistema.

CAPÍTULO 5

Alcance

El objetivo principal de este proyecto es la automatización y optimización del proceso de mantenimiento y supervisión del sistema de navegación aérea VOR (VHF Omnidirectional Range); *VHF (Very High Frequency) Omnidirectional Range*, específicamente del equipo DVOR modelo 432 en el Aeropuerto Internacional La Aurora. El proyecto, comprendido entre enero y noviembre de 2024, se centrará exclusivamente en este equipo, sin involucrar ninguna otra radioayuda en Guatemala o Centroamérica.

Se implementará un sistema de monitoreo para la recopilación automática de datos críticos del DVOR432, que permitirá la creación de un programa capaz de analizar estos datos y generar informes periódicos de mantenimiento de manera automática y eficiente. Además, se integrará un sistema de monitoreo visual con acceso remoto, diseñado para notificar de manera inmediata cualquier desviación de los parámetros establecidos y/o tolerancias en el funcionamiento del equipo.

El análisis de datos se realizará a través de un programa desarrollado específicamente para este fin, que obtendrá los parámetros necesarios utilizando el protocolo SNMP, aprovechando los recursos proporcionados por el servidor del Sistema de Control y Monitoreo (MCS), el cual estará soportado por

una computadora industrial Advantek. Asimismo, se utilizará un Servidor Titan, donde se correrán los dockers dedicados al proyecto.

Adicionalmente, se implementarán alarmas y notificaciones automáticas mediante correos electrónicos dirigidos a los técnicos e ingenieros responsables.

Se espera que este proyecto reduzca significativamente la carga laboral asociada al monitoreo y generación de informes de mantenimiento, mejorando la calidad y precisión de los mismos. También se anticipa una disminución en los costos operativos y una reducción de errores humanos.

CAPÍTULO 6

Marco teórico

6.1. Corporación Centroamericana de Servicios de Navegación Aérea (COCESNA)

Es un Organismo Internacional de Integración Centroamericana, sin fines de lucro y de servicio público, con estatus legal y autonomía financiera. Fue fundado el 26 de febrero de 1960. La corporación establecida con objetivos y propósitos específicos para cumplir de manera conjunta con los compromisos internacionales en materia de aviación civil. Estos compromisos surgieron como resultado de la adhesión de los países signatarios al Convenio sobre Aviación Civil Internacional de 1944, conocido como el Convenio de Chicago.

En este contexto, COCESNA cuenta con derechos exclusivos para ofrecer servicios de Tránsito Aéreo, Telecomunicaciones Aeronáuticas y Radioayudas en los territorios de los Estados Miembros. Además, tiene la responsabilidad de proporcionar otros servicios establecidos en los planes regionales que le hayan sido confiados por las Partes Contratantes a través de Acuerdos Internacionales. Las operaciones de COCESNA se guían principalmente por las Normas y Métodos recomendados por la OACI (Organización de Aviación Civil Internacional) (Organización de Aviación Civil Internacional).

[3]

6.2. Organización de Aviación Civil Internacional (OACI) y anexo 10 de OACI

La Organización de Aviación Civil Internacional (OACI) es una agencia especializada de las Naciones Unidas que coordina y regula la aviación civil internacional. Su función principal es establecer normas y reglamentos necesarios para la seguridad, protección, eficiencia y regularidad de la aviación, así como para la protección del medio ambiente en este ámbito [4]. Dentro de las normas, se encuentra el Anexo 10; en el cual se describen todas las regulaciones referentes a las Telecomunicaciones Aeronáuticas. Se enfoca en los estándares y métodos recomendados para las comunicaciones, ayudas a la navegación (el VOR dentro de ellas, descrito más adelante) y sistemas de vigilancia que son esenciales para la navegación aérea. Este anexo busca garantizar la seguridad, regularidad y eficiencia en la navegación aérea global. Por lo tanto, el comportamiento de todos los VOR se rige mediante esta regulación. Las tolerancias de los parámetros, calidad de las señales y, en general, todo lo que se monitoreará en este proyecto está basado en esta regulación global.

6.3. Radioayudas *(Navids: Navigational Aids)*

"Las radioayudas son dispositivos físicos en tierra que proveen información de navegación aérea"[5]. Estos dispositivos desempeñan funciones esenciales para garantizar que cualquier aeronave pueda operar con seguridad en situaciones de baja visibilidad y funcionan como un "método de respaldo"para los sistemas de navegación satelital, como el GPS. Se podrían calisificar a las radioayudas en dos categorías dependiendo de su función: *Instrument Landing System (ILS)* y *Very High*

Frecuency Omnidirectional Range (VOR).

El Sistema de Aterrizaje por Instrumentos (ILS *(Instrument Landing System)*) está concebido para proporcionar una trayectoria de aproximación que permita la alineación exacta y el descenso preciso de una aeronave en la fase final de aproximación a una pista de aterrizaje. Los componentes fundamentales de un ILS incluyen el localizador y la pendiente de planeo (*Glideslope*) en conjunto con el equipo de medición de distancia (*DME (Distance Measuring Equipment): Distance Measuring Equipment*). Funcionalmente, el sistema puede dividirse en tres partes: información de orientación, que abarca el localizador y la pendiente de planeo; información de alcance, que involucra el DME; e información visual, que comprende las luces de aproximación, las luces de toma de contacto y de la línea central, y las luces de la pista [6]. Dado que la atención y los objetivos del proyecto se centrarán en el VOR, en la sección subsiguiente se profundizará de manera más detallada en esta radioayuda.

6.4. *VHF Omnidirectional Range* (VOR)

Según la referencia [7] proporcionada por COCESNA, la cual corresponde a un programa base de entrenamiento para inspectores de vuelo, en el cual se abordan los conceptos, funcionamiento y operación de cada uno de los equipos de navegación.

En el Sistema Estadounidense (y en gran mayoría de países del mundo) de Espacio Aéreo, las rutas aéreas se basan principalmente en el Sistema de *VHF Omnidirectional Range* (VOR). Además, el sistema de rutas aéreas puede utilizar DME (Equipo de Medición de Distancia, por sus siglas en inglés) como apoyo para brindarle más información a las aeronaves.

Un VOR es un dispositivo electrónico de navegación, posicionado en tierra, que transmite señales de navegación de VHF en un rango de 360 grados en azimut, es decir, horizontalmente con respecto al suelo y orientado desde el norte magnético (lo que corresponde al Sur geográfico del globo terrestre). Estos sistemas operan en un rango de frecuencia que va desde 108MHz hasta 118MHz, correspondiente al espectro de (Very High Frequency) (VHF). De acuerdo con las Designaciones Estándar de Letras para Bandas de Frecuencia de Radar establecidas por la IEEE en 1976 [8], este rango abarca desde los 30MHz hasta los 300MHz (consultar la Figura 2 para más información acerca del estándar de frecuencias). Un curso VOR es el resultado de la comparación de fase entre dos señales de la misma frecuencia, siendo ambas moduladas en la señal principal portadora transmitida por el VOR. La medición de la diferencia de fase se realiza en el receptor a bordo del avión, en el cual existen circuitos para medir esta diferencia de fase e indicarla visualmente en el instrumento VOR dentro de la aeronave.

International table				
Band designation	**Nominal frequency range**	**Specific frequency ranges for radar based on ITU assignments** (see Notes 1, 2)		
		Region 1	**Region 2**	**Region 3**
HF	3–30 MHz	(Note 3)		
VHF	30–300MHz	None	138 –144 MHz 216– 225 MHz (See Note 4)	223-230 MHz
UHF	300–1000 MHz (Note 5)	420–450 MHz (Note 4) 890–942 MHz (Note 6)		
L	1–2 GHz	1215–1400 MHz		
S	2–4 GHz	2300–2500 MHz		
		2700–3600 MHz	2700–3700 MHz	
C	4–8 GHz	4200– 4400 MHz (Note 7)		
		5250–5850 MHz	5250–5925 MHz	
X	8–12 GHz	8.5–10.68 GHz		
Ku	12–18 GHz	13.4–14 GHz		
		15.7–17.7 GHz		
K	18–27 GHz	24.05–24.25 GHz	24.05–24.25 GHz 24.65–24.75 GHz (Note 8)	24.05–24.25 GHz
Ka	27–40 GHz	33.4–36 GHz		
V	40–75 GHz	59–64 GHz		
W	75–110 GHz	76–81 GHz		
		92–100 GHz		
mm (Note 9)	110–300 GHz	126–142 GHz		
		144–149 GHz		
		231–235 GHz 238–248 GHz (Note 10)		

Figura 2: Nomenclatura estándar de bandas de frecuencia de radar [8]

Las señales moduladas en cuestión corresponden al campo de referencia y al campo variable, de manera que la diferencia de fase, expresada en grados, indica la posición relativa de la aeronave en relación con el VOR. La rotación a lo largo del campo variable, partiendo desde el norte, genera una diferencia de fase de un grado eléctrico por cada grado de rotación por lo que el azimut correcto se determina mediante la diferencia de fase entre la fase de la señal de referencia y la fase de la señal variable. Para las aeronaves ubicadas al norte de la estación VOR, la señal variable está en fase con la señal de referencia, lo que implica una diferencia de fase de 0 grados. En consecuencia, el indicador de la aeronave muestra 0 grados cuando la aguja está centrada, y esto se repite para cada Radial.

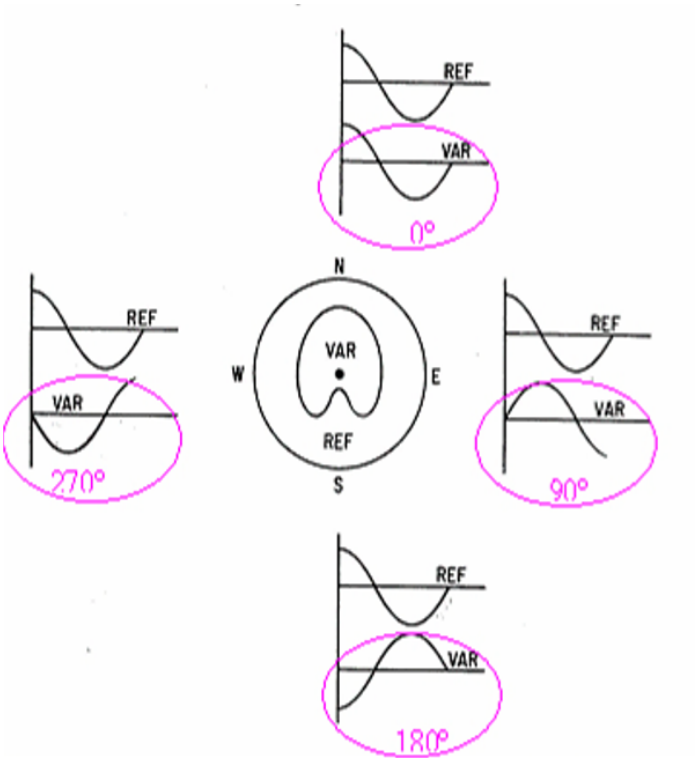


Figura 3: Relación entre la diferencia de fase entre el campo de referencia y de fase de acuerdo con los puntos cardinales [7]

Además de proporcionar información sobre el azimut en relación con la posición relativa de la aeronave, la señal portadora también transmite información de identificación en Código Morse y se puede incluir información de *Squitter*, un tipo de transmisión de datos automática que proporciona datos de vigilancia de Aeronaves en un entorno de tráfico aéreo.

6.4.1. *Doppler VOR* (DVOR)

El VOR Doppler representa una versión más avanzada del VOR convencional, diseñada para entornos donde los factores externos pueden causar interferencias o impidan la correcta propagación de las señales, como es el caso del Aeropuerto Internacional La Aurora. Esta variante del VOR se basa en el principio del efecto Doppler, cuya comprensión se facilita mediante una analogía con el funcionamiento de un faro en la navegación marítima: un faro emite un haz de luz rotativo

a una velocidad constante por lo que cuando este haz de luz pasa por el norte, emite un pulso característico, permitiendo notificar a los usuarios de dicho evento y, de esta manera, es posible determinar la posición relativa de una embarcación al calcular el tiempo transcurrido desde la emisión del pulso hasta que el haz de luz la alcanza.

Para lograr este efecto en el VOR Doppler, dado que la rotación mecánica no es factible, se recurre a una especeie de simulación en donde se dispone de un arreglo circular de antenas y se les alimenta secuencialmente con energía de banda lateral RF. Este proceso provoca que el centro de radiación aparente de la señal se desplace en círculo a una velocidad constante. [9]

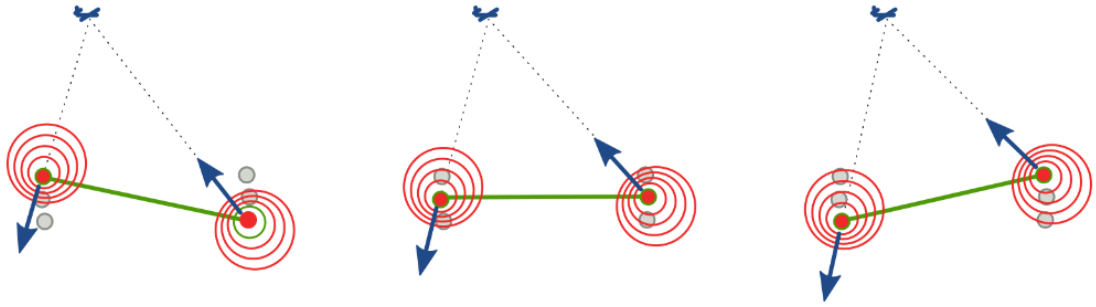


Figura 4: Principio de funcionamiento del DVOR y la generación de la señal variable. En el caso de la imagen se utilizan dos sistemas DVOR para realizar una triangulación de la aeronave [10]

6.4.2. *Distance Measuring Equipment* (DME)

El DME opera en conjunto con el VOR en la mayoría de ocasiones, este proporciona información precisa sobre la distancia en millas náuticas, entre la aeronave y la estación VOR; completando así el conjunto de datos necesarios para determinar la ubicación exacta de la aeronave en el espacio aéreo. El DME opera mediante la transmisión y recepción de pulsos emparejados desde la estación terrestre y se opera a una frecuencia aproximada de 1GHz. [7] El DME basa su funcionamiento en la medición del tiempo entre la transmisión y la recepción de pulsos de radiofrecuencia. En este proceso, la

aeronave transmite una serie de pulsos en pares hacia la estación DME, la cual los recibe y los retransmite. La aeronave, a su vez, mide el tiempo transcurrido desde la transmisión inicial de los pulsos hasta que son recibidos nuevamente. Haciendo uso de este intervalo de tiempo, es posible determinar la distancia entre la aeronave y la estación terrestre.

6.5. THALiX OS

THALiX OS es un sistema operativo con Kernel Linux basado en la distribución Fedora de Redhat. Este SO está creado y desarrollado por Thales Group exclusivamente para uso y compatibilidad con el Sistema de Control y Monitoreo (MCS) para equipos de navegación aérea. [11]

6.6. Virtualización

La virtualización es una tecnología que permite crear versiones virtuales o simuladas de recursos físicos como servidores, almacenamiento, redes o sistemas operativos. Al abstraer los recursos físicos, la virtualización permite que múltiples sistemas operativos y aplicaciones se ejecuten en un solo servidor físico, mejorando la eficiencia, optimizando el uso de los recursos y reduciendo costos operativos [12]. El responsable de que la ejecución de cualquier tipo de virtualización sea posible es el hipervisor; un software dedicado a la asignación y separación de recursos físicos a los enotrnos virtuales que lo necesiten [13].

La forma de virtualización con la que, por lo general, existe mayor familiaridad es una máquina virtual; de acuerdo a [14], son una emulación completa de un sistema físico que permite ejecutar un sistema operativo independiente y sus aplicaciones en un entorno completamente aislado. Cada máquina virtual incluye su propio kernel, drivers, y recursos de hardware vir-

tualizados, lo que permite que funcione como un equipo físico separado, incluso en el mismo servidor físico que otras máquinas virtuales.

6.6.1. Contenedores y Dockers

Los contenedores son otro tipo de tecnología de virtualización que permite empaquetar y ejecutar aplicaciones y sus dependencias en un entorno aislado, pero compartiendo el mismo sistema operativo subyacente. A diferencia de las máquinas virtuales, los contenedores no requieren un sistema operativo completo, lo que los hace más ligeros y eficientes, permitiendo un despliegue rápido y la ejecución de múltiples instancias en un solo servidor físico [15]. Dado que no se virtualiza un sistema operativo completo, los contenedores por lo general tienen un solo objetivo específico, por ejemplo, en este proyecto se utilizarán tres tipos contenedores: Python, PostgreSQL y Grafana.

Docker es una herramienta o plataforma de creación, despliegue y gestión de contenedores, permitiendo empaquetar varios contenedores en una sola aplicación incluyendo todas sus dependencias [16], en el caso de este proyecto, se tendrán los tres tipos de Dockers mencionados anteriormente en una sola aplicación que los ejecute de manera simultánea y coordinada. En la Figura 5 es posible observar la diferencia entre arquitecturas/estructuras y diferencias entre máquinas virtuales y contenedores.

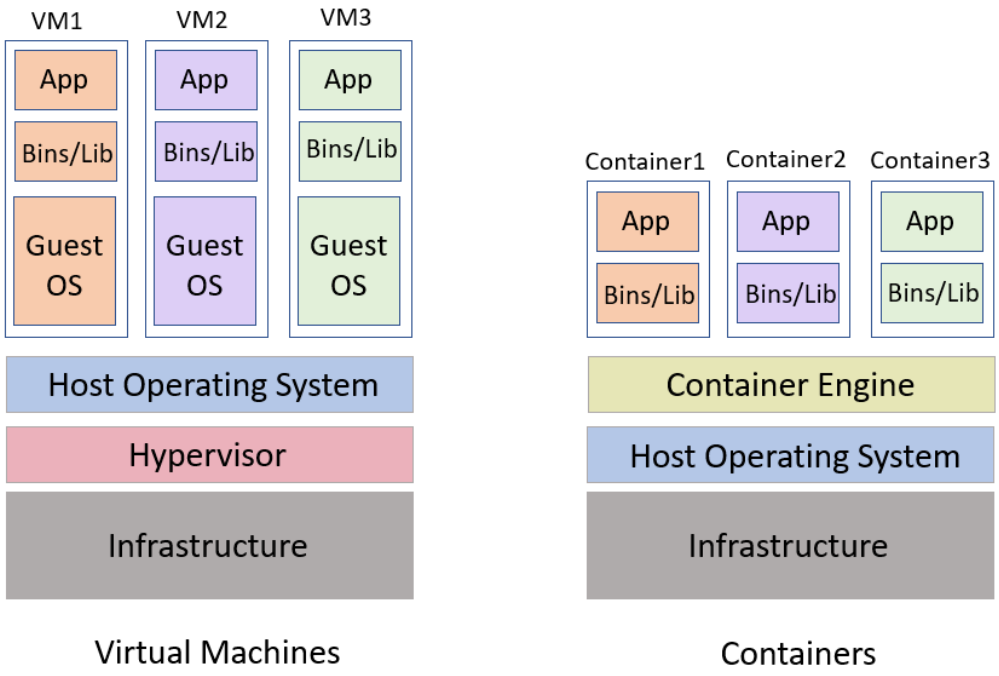


Figura 5: Comparación y representación de arquitecturas de virtualización: máquina virtual y contenedores [17]

6.7. PostgreSQL

Según el sitio web oficial [18], PostgreSQL surgió del proyecto POSTGRES en la Universidad de California, Berkeley en 1986. Se trata de un sistema de base de datos de código abierto con la característica de ser objeto-relacional, basado en el lenguaje SQL. Demuestra ser un sistema de alta compatibilidad, robustez, seguridad y fiabilidad, además de su facilidad de aprendizaje de uso; razones por las cuales se eligió para el desarrollo de este proyecto.

6.8. SNMP

El Protocolo Simple de Administración de Red (SNMP (Protocolo Simple de Administración de Red), por sus siglas en in-

glés) constituye un stándar de Internet empleado para la administración y supervisión de dispositivos de red, según [19]. Estos dispositivos pueden abarcar Routers, Switches, Servidores, impresoras y, en el contexto específico, equipos de navegación aérea. Un servidor SNMP representa un *software* dedicado a recopilar y almacenar información referente al estado y la configuración de tales dispositivos, proporcionando una interfaz para que otros sistemas de gestión de red accedan a dicha información. Los servidores SNMP posibilitan el monitoreo y la administración de dispositivos de red de manera centralizada, facilitando así la detección y resolución de problemas, la optimización del rendimiento y la planificación de capacidad de la red o sistema de enfoque.

6.8.1. Arquitectura

De acuerdo con [20], el protocolo se basa en una arquitectura de cliente-servidor, siendo los componentes principales:

- **Gestores**
  Se trata de sistemas centralizados encargados de gestionar la extracción de la información y el control de los equipos en la red. Algunos ejemplos de estos serían Solar Winds, Zabbix, etc. En el contexto del proyecto, se estará realizando un gestor personalizado y dedicado con funciones solamente de monitoreo.
- **Agentes**
  Son procesos de software que, por lo general, residen en los dispositivos gestionados. Sin embargo, en el caso particular de MCS, tanto el gestor (nativo) como los agentes (en forma de proxies) correspondientes a cada equipo corren en el mismo dispositivo (servidor). Los agentes responden a las instrucciones del gestor, ya sea proporcionando la información requerida o ejecutando acciones sobre el equipo.
- ***Management Information Base*** **(MIB)**
  La MIB es una base de datos jerárquica con una estructu-

ra de árbol que contiene información acerca de todos los objetos gestionables del equipo. Cada uno de estos objetos cuenta con un identificador único llamado OID *(Object Identifier)*.

SNMP se vale de una red basada en TCP/IP, utilizando el protocolo UDP (User Datagram Protocol) como medio de transporte de PDUs (*Protocol Data Units*). Lo cual establece la ventaja de baja latencia y poca sobrecarga de la red [21], ideal para protocolos de constante comunicación como SNMP, a cambio de la desventaja de que desde UDP no se tiene garantizada la entrega de paquetes [21] ni control de congestión. Los PDU son un tipo de paquete estándar utilizado por SNMP para la comunicación entre gestores y agentes. Existe una lista de tipos de PDU que, básicamente, son de extracción de información, establecimiento de acciones y alertas o Traps.

La comunicación SNMP utilza por defecto el puerto lógico 161, a excepción de los Traps, que utilizan el puerto lógico 162 [20].

6.8.2. SNMPv3

SNMPv3 se refiere a la tercera versión del mencionado protocolo, solucionando el problema más notorio de las versiones anteriores: la seguridad. [19]. Esta nueva arquitectura induce el modelo de seguridad basado en usuario, garantiza que los datos no se modifican durante la transmisión, confirma que la solicitud/respuesta provenga del origen correcto y protege los datos contra intrusiones [22]. En adición, en SNMPv3 se emplean distintos métodos de encriptación tanto para la autenticación como para la privacidad, es decir, para cada sesión de SNMPv3 se necesita un usario y dos çontraseñasz cada una de ellas tiene un protocolo distinto de encriptación que debe ser descrito al momento de establecer la sesión. Para autenticación se puede usar MD5 o DES y para privacidad puede ser SHA o AES.

6.9. *Application Programming Interface:* API (Interfaz de Programación de Aplicaciones)

De acuerdo con el artículo [23], una API es un conjunto de protocolos que permiten la interacción entre dos o varias aplicaciones de software para intercambio de información o características. Su gran ventaja es la facilitación de desarrollo de software ya que pueden aprovecharse capacidades de software ya existente y se pueden implementar en una aplicación propia. Un ejemplo sencillo es el mencionado en [24], la aplicación de clima de cualquier telélono inteligente moderno hace uso de una API para comunicarse con el sitio oficial de meteorología de la región. De esta manera, la aplicación no tiene la çarga"de recopilación de la información sino solamente de desplegarla, teniendo la ventaja de utilizar información oficial, ser una aplicación más ligera, un desarrollo más eficiente, etc.

6.9.1. Tipos de APIs

Según el motivo de creación de las APIs, pueden tener la siguiente clasificación [24]:

- **SOAP**: Es el tipo de API que primero se desarrolló, se encuentra en las aplicaciones más antiguas y se emplea el Protocolo Simple de Acceso a Objetos, donde el cliente y el servidor intercambian mensajes utilizando XML.
- **RPC**: Llamadas a procedimientos remotos; consisten en que se ejecuta un proceso de manera remota en el servidor y solamente se retorna el resultado al cliente.
- **WebSocket**: Se hace uso de objetos JSON para la transmisión de datos, es posible la comunicación bidireccional entre aplicaciones de cliente y servidor. Utilizada (como se menciona en el nombre) para aplicaciones web.
- **REST**: Transferencia de estado representacional. Se realiza el intercambio de información mediante el protocolo

HTTP. Es el tipo de API más empleado actualmente y tiene la característica de que el servidor no guarda ningún tipo de información acerca de las solicitudes y las respuestas al cliente son datos simples, lo que resulta conveniente en aplicaciones de desarrollo de software dedicado.

CAPÍTULO 7

Actualización y estandarización de MCS

7.1. Actualización de Sistema Operativo y software MCS

Como punto de partida en este proyecto, se consideró necesario llevar a cabo una actualización integral del sistema de control y monitoreo de radioayudas perteneciente a COCESNA. Esta actualización inició con la renovación del servidor, en el cual se implementó la versión más reciente del software THALiX (versión 13.1) disponible a la fecha, junto con la instalación de la versión más reciente del sistema MCS (versión 3.0).

El proceso completo de actualización se encuentra debidamente documentado en la Guía de instalación de THALiX 13.1 y MCS 3.0, donde se detallan los pasos seguidos, las configuraciones realizadas y las recomendaciones para futuras implementaciones. Este procedimiento constituye la base tecnológica que soporta el resto de las actividades desarrolladas en el marco del presente trabajo.

Esta etapa del proyecto representó un desafío significativo,

dado que no se disponía de guías o manuales de instalación para los componentes de software involucrados. En consecuencia, fue necesario invertir una cantidad considerable de tiempo en el proceso de instalación, así como en la identificación y corrección de errores inherentes al software en su versión inicial. Este esfuerzo fue fundamental para garantizar el correcto funcionamiento y la estabilidad del sistema.

7.2. Estandarización del monitoreo de radioayudas Guatemala

Con el objetivo de garantizar la escalabilidad del sistema en todo momento, se decidió llevar a cabo una estandarización integral del diseño, la nomenclatura y el sistema operativo de todo el MCS. Esta estandarización permite una mejor organización, facilita el mantenimiento futuro y asegura la interoperabilidad en diferentes entornos de implementación.

A continuación, se presentan los diversos mapas configurados hasta la fecha en el territorio guatemalteco, cada una de las imágenes geográficas se obtuvo utililizando la herramienta Google Earth. Para facilitar la comprensión del lector, es importante mencionar que el MCS utiliza un esquema de codificación por colores para representar el estado de cada equipo o sitio configurado, según se detalla a continuación:

- Verde: Indica que el equipo opera en condiciones normales.
- Amarillo: Representa que el equipo está en mantenimiento o en condiciones de pre-alarma.
- Rojo: Señala que el equipo se encuentra en condiciones de alarma debido a la desviación de uno o más parámetros críticos.
- Celeste: Denota que el equipo está fuera de rango o que la sesión de comunicación no se ha establecido.
- Gris: Indica que el equipo no aplica para condiciones de estado o alarmas.

Este sistema de codificación por colores es una herramienta fundamental para el monitoreo, ya que permite una interpretación rápida y eficiente del estado operativo de los equipos, mejorando la capacidad de respuesta ante cualquier eventualidad. Adicionalmente, se destaca que cada radioayuda está vinculada a un equipo denominado RCSE (identificado en la interfaz gráfica como RCSx), cuyo propósito y función serán detallados en la sección Estudio de topología del sistema. En términos generales, este dispositivo funciona como un enlace clave para habilitar la comunicación efectiva entre el servidor MCS y las radioayudas.

El proceso para configurar una radioayuda en el MCS se desarrolla siguiendo los pasos descritos a continuación:

1. Crear un agente proxy para el RCSE: Utilizando el menú superior de configuración, se debe crear un agente proxy (explicado y descrito en Estudio de topología del sistema) correspondiente al dispositivo RCSx (RCSE) y se le asigna un puerto SNMP específico. Este paso establece la comunicación inicial entre el sistema y el RCSE.
2. Generar una instancia gráfica del agente proxy: Una vez creado el agente proxy, se debe generar su representación en la interfaz gráfica del sistema mediante el puerto SNMP configurado para facilitar su administración.
3. Configurar la radioayuda ligada al RCSE: Se accede a la instancia gráfica del agente proxy correspondiente al RCSx, y desde ahí se selecciona la radioayuda vinculada. Posteriormente, se crea un agente proxy adicional para la radioayuda específica.
4. Generar una instancia gráfica de la radioayuda: Finalmente, se crea la representación gráfica de la radioayuda en la interfaz del MCS, lo que permite monitorear y gestionar su estado y configuración.

Figura 6: Pantalla de inicio de nueva versión de MCS estandarizada. Mostrando el mapa de Centroamérica con los estados miembros de COCESNA.

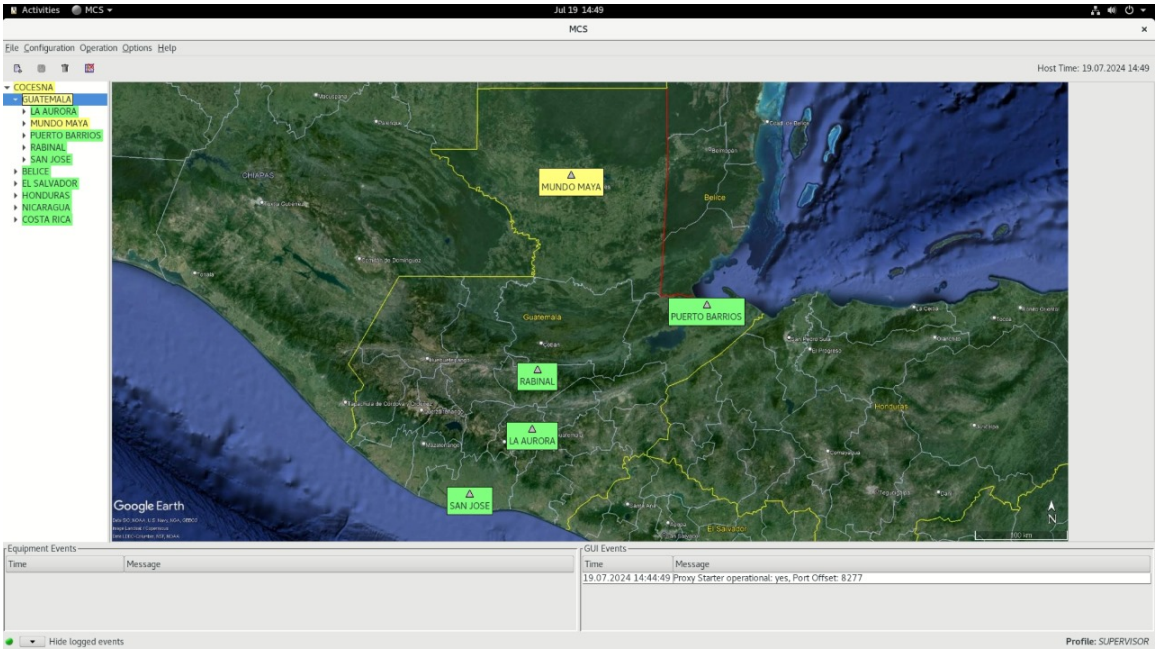


Figura 7: Mapa de Guatemala mostrando el resumen de cada sitio donde se cuenta con al menos una radioayuda.

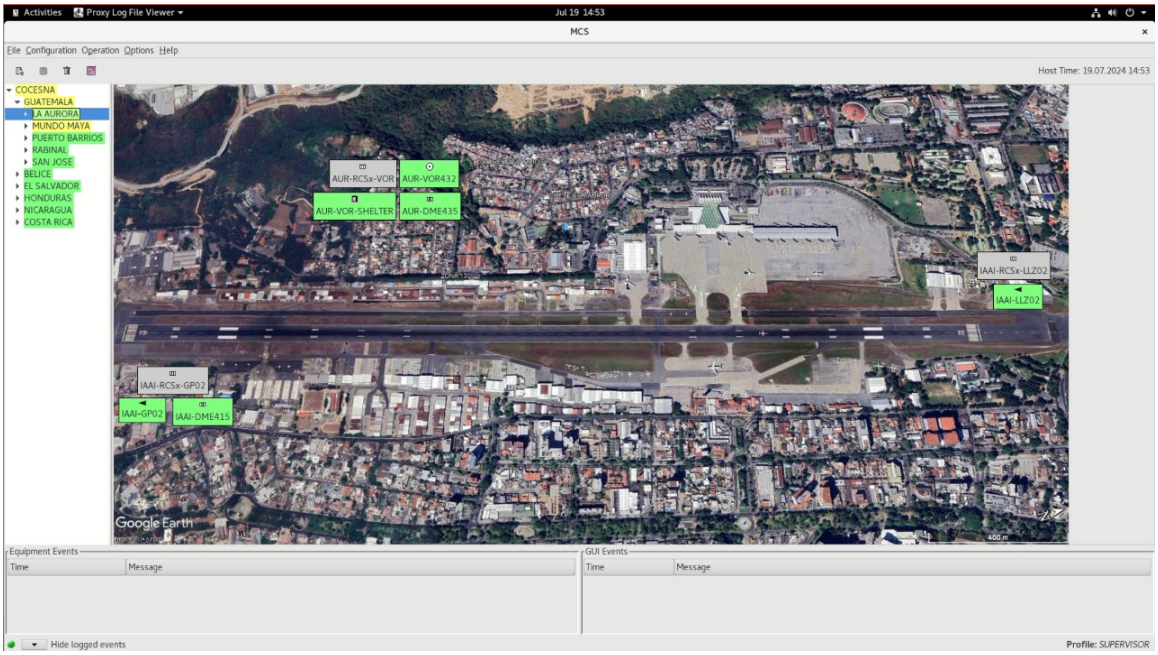


Figura 8: Aeropuerto Internacional La Aurora, Ciudad de Guatemala.

Se observan en posiciones geográficas aproximadas cada una de las radioayudas con las que cuenta la pista de despegue y aterrizaje.

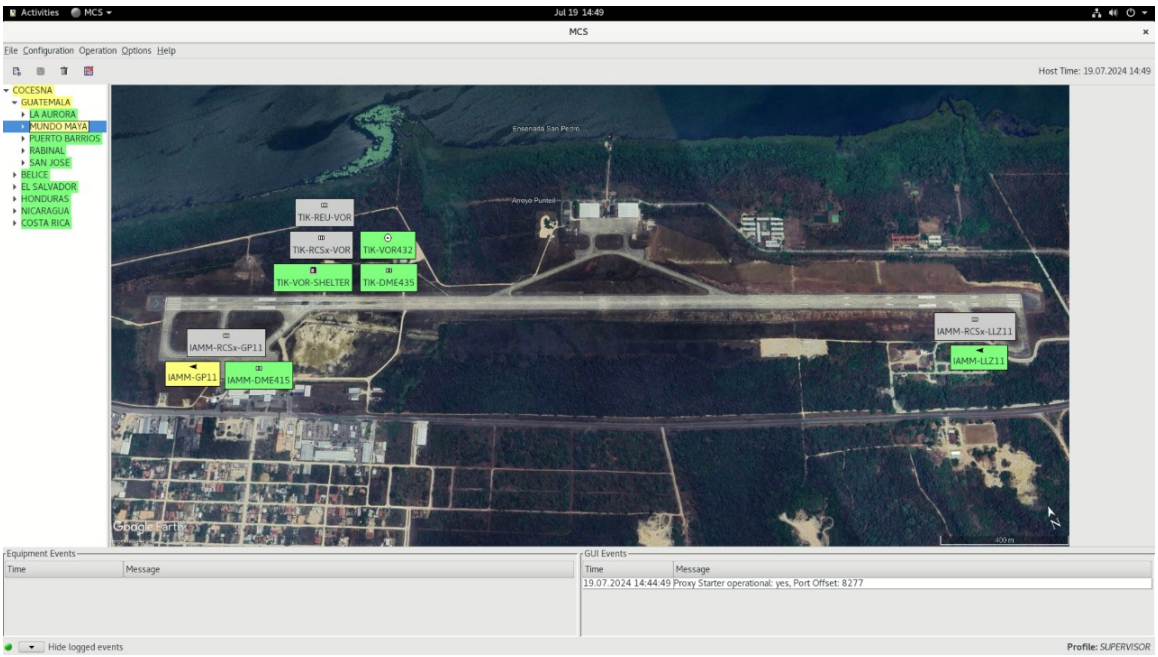


Figura 9: Aeropuerto Internacional Mundo Maya, Petén.

Se observan en posiciones geográficas aproximadas cada una de las radioayudas con las que cuenta la pista de despegue y aterrizaje. Se observa al indicador del Glideslope en amarillo debido a una rutina de mantenimiento activa en el momento de captura de estas imágenes, lo cual no representa fallos ni mal funcionamiento de la radioayuda.

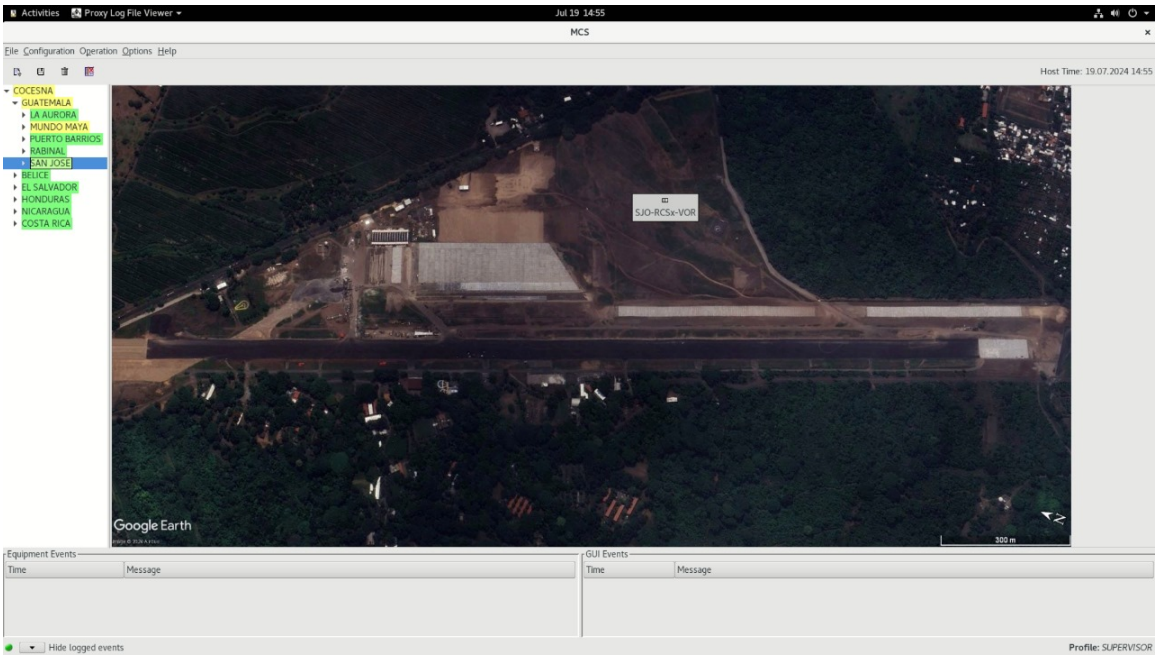


Figura 10: Aeródromo de San José, Escuintla.

En este caso no se observan radioayudas activas ya que el aeródromo se encuentra demolido debido a remodelaciones y reestructuración. Este sitio cuenta solamente con un DVOR/DME.

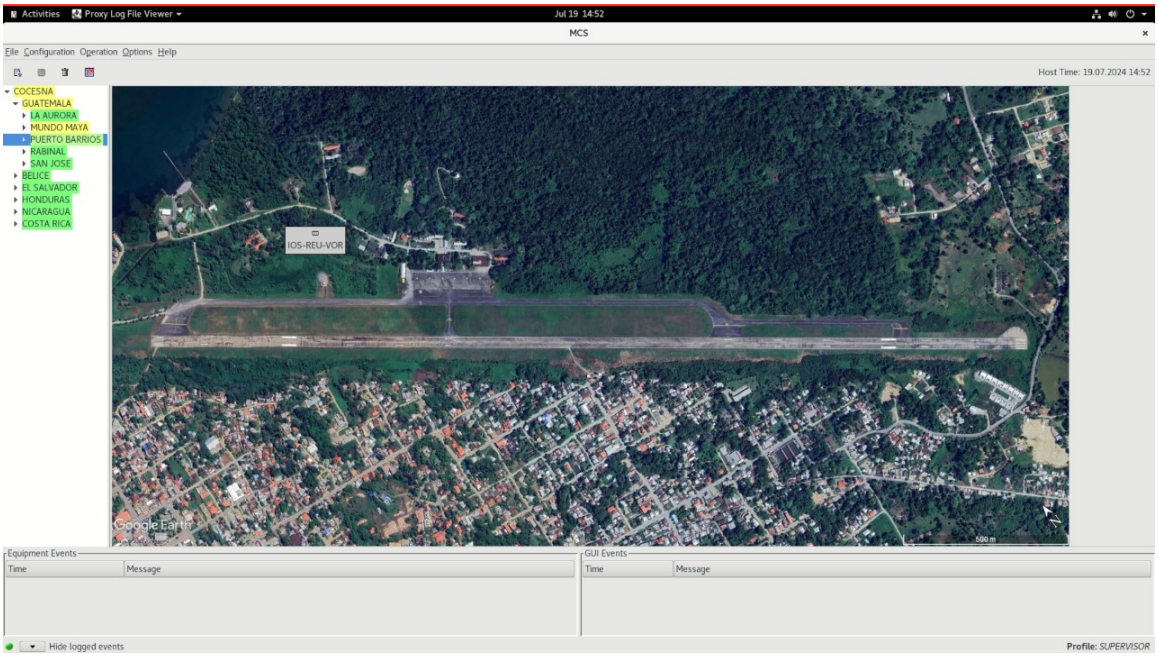


Figura 11: Aeropuerto de Puerto Barrios, Izabal.

En este caso no se observan radioayudas activas ya que actualmente se está trabajando en un proyecto de restauración del enlace de red Guatemala-Izabal, sin embargo, el equipo es monitoreado de manera local por personal de COCESNA de manera provisional. Este sitio cuenta solamente con un DVOR/DME.

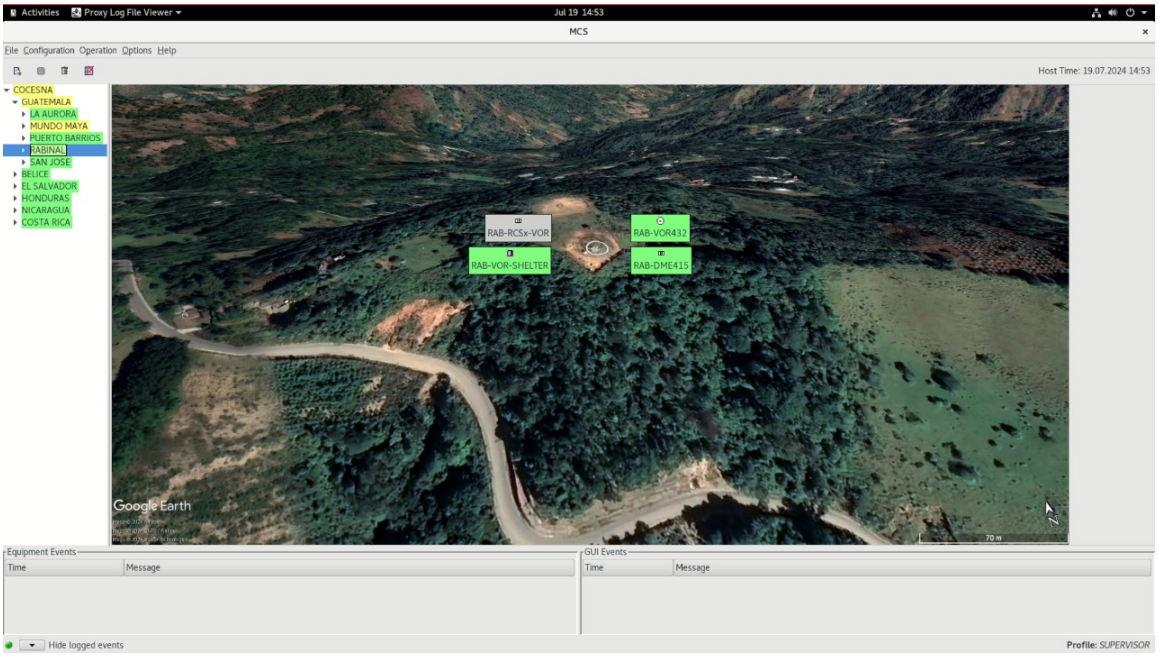


Figura 12: Sitio alto en Rabinal, Baja Verapaz.

Este sitio cuenta solamente con un DVOR/DME de ruta dado que no se cuenta con una pista de despegue/aterrizaje, tratándose de un punto geográficamente estratégico en el que se cubren varias rutas en el espacio aéreo guatemalteco.

CAPÍTULO 8

Monitoreo y obtención de parámetros críticos

8.1. Estudio de topología del sistema

8.1.1. Conocimientos iniciales

La instalación optimizada y la actualización del sistema operativo del servidor MCS, junto con el software de control y monitoreo, fueron realizadas no solo como parte del proceso de modernización, sino también con el propósito de estudiar y comprender el mecanismo de intercambio de información entre el sistema y las radioayudas. Además, este proceso permitió analizar a fondo la topología e infraestructura subyacente, dado que no se disponía de documentación suficiente al respecto. Esta labor fue esencial para adquirir un entendimiento integral del funcionamiento y las interconexiones del sistema.

La información disponible al inicio del proyecto era de carácter general y carecía de detalles específicos. En la Figura 13 se presenta el diagrama de la topología inicial, elaborado con base en la información proporcionada en esa etapa. Si bien se

conocía que el protocolo SNMP era utilizado para el intercambio de información entre el servidor MCS y el sistema DVOR, se identificaron ciertas discrepancias que requerían ser aclaradas para garantizar una comprensión completa y precisa del funcionamiento del sistema.

De manera nativa, el DVOR432 no dispone de una interfaz de red, siendo la comunicación serial el único medio para acceder a la información del equipo. Por ello, se requiere la intervención de un agente que actúe como traductor entre la comunicación serial y los protocolos de red. En este contexto, el RCSE o *Remote Control and Status Equipment* es un dispositivo especializado utilizado en los sistemas de navegación aérea para supervisar y controlar de forma remota las radioayudas instaladas en diversos puntos geográficos. Este equipo actúa como un intermediario entre las radioayudas y los sistemas centralizados de monitoreo y control, como el MCS, permitiendo la transmisión bidireccional de información crítica. Este dispositivo se asemeja a un *switch* de red, ya que centraliza las conexiones de todas las interfaces, tanto del DVOR y el DME, como de los sensores auxiliares instalados en el sitio. Entre estos sensores se incluyen aquellos destinados a medir temperatura, detectar humo, intrusión y obstrucción.

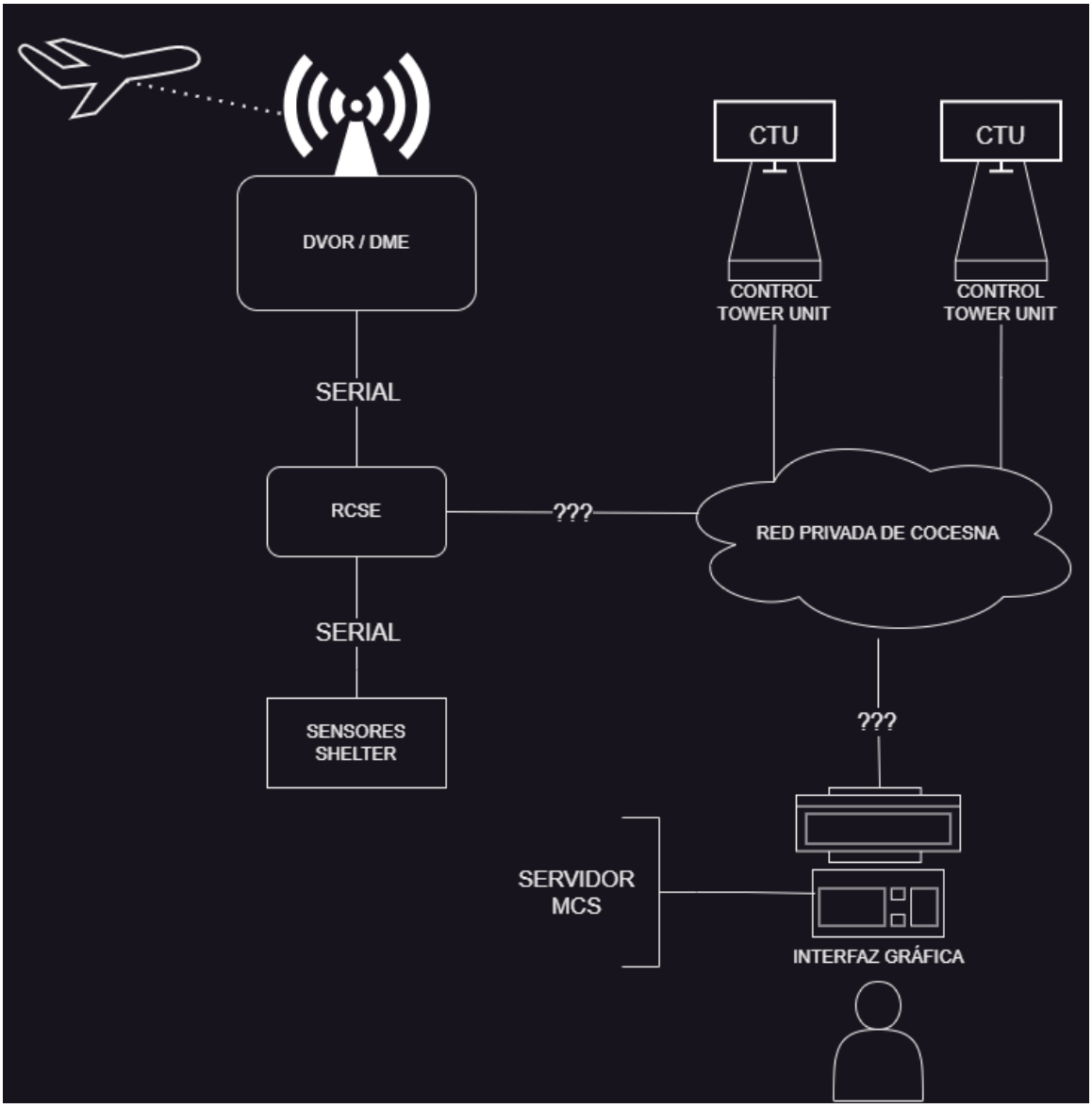


Figura 13: Topología al inicio del proyecto, limitada en cuanto a detalles técnicos.

En esta etapa, se desconocían los protocolos específicos y los tipos de sesiones que se establecían entre los distintos equipos. Esta representación inicial sirvió como punto de partida para el análisis y comprensión del sistema, aunque fue necesario realizar investigaciones adicionales para clarificar estas incertidumbres y completar el modelo de comunicación e interacción entre los componentes.

Al acceder al RCSE y revisar su configuración, se evidencia que este dispositivo dispone de cinco puertos lógicos para

conexiones TCP/IP, pero carece de capacidades SNMP. Dado que el equipo DVOR tampoco posee capacidades de red y no es posible establecer sesiones SNMP directamente con el RC-SE, surge el principal desafío de este proyecto: comprender y determinar el método mediante el cual se establece la sesión SNMP para el intercambio de información. Este obstáculo pone de manifiesto una de las interrogantes más críticas del sistema, ya que el protocolo SNMP es fundamental para la integración del MCS con las radioayudas.

Adicionalmente, en la parte superior de la Figura 13 se muestran dos elementos denominados CTU (*Control Tower Unit*), los cuales son equipos simples y dedicados específicamente a mostrar el estado de las radioayudas tanto a los controladores en la torre de control como al centro de control del Aeropuerto Internacional La Aurora. Estas unidades operan como una versión limitada del MCS, diseñada exclusivamente para el monitoreo. No permiten el acceso a información detallada más allá del estado de los equipos. En la interfaz de las CTU, únicamente se visualizan los símbolos que representan las radioayudas, acompañados de su correspondiente código de colores, facilitando así una rápida interpretación del estado operativo para los usuarios en estos puntos estratégicos.

8.1.2. Comprensión del sistema

Tras invertir un tiempo considerable en la exploración exhaustiva de los directorios y archivos del backend del MCS, el análisis de múltiples configuraciones de radioayudas dentro del sistema, y la realización de varias visitas presenciales al DVOR, fue posible descifrar y documentar la topología que se presenta en la Figura 14. Este proceso resultó esencial para comprender las interconexiones y flujos de comunicación entre los distintos componentes del sistema, lo que permitió superar las limitaciones iniciales de información y establecer una base sólida para la implementación y optimización del proyecto.

El MCS se compone esencialmente de la interacción entre

dos componentes principales: el entorno gráfico y los binarios del sistema. Los binarios son archivos compilados en lenguaje C, diseñados para cumplir la función de emular un agente SNMP. En este contexto, estos binarios actúan como representaciones virtuales de las radioayudas, adoptando el nombre de **Agentes Proxy**. La función principal de los Agentes Proxy es intermediar en la comunicación entre el sistema de monitoreo y las radioayudas físicas, permitiendo al MCS interpretar y gestionar la información como si interactuara directamente con un agente SNMP. Este diseño modular y distribuido facilita la implementación y el monitoreo, al tiempo que optimiza la gestión de los recursos en el sistema.

El intercambio de información entre la radioayuda y el servidor MCS se realiza a través del RCSE mediante una sesión TCP. El RCSE dispone de un número 'n' de puertos lógicos, comúnmente cinco, denominados Usuarios, que permiten mantener 'n' sesiones activas simultáneamente. Estos puertos lógicos están reservados para la comunicación con diferentes componentes del sistema, tales como las CTU, el servidor MCS y, en algunos casos, las instancias virtuales del MCS. En el contexto de este proyecto, la sesión establecida para la comunicación se encuentra configurada en el puerto 2075 con la IP correspondiente al RCSE del DVOR AILA (Aeropuerto Internacional La Aurora) que es 172.20.12.3.

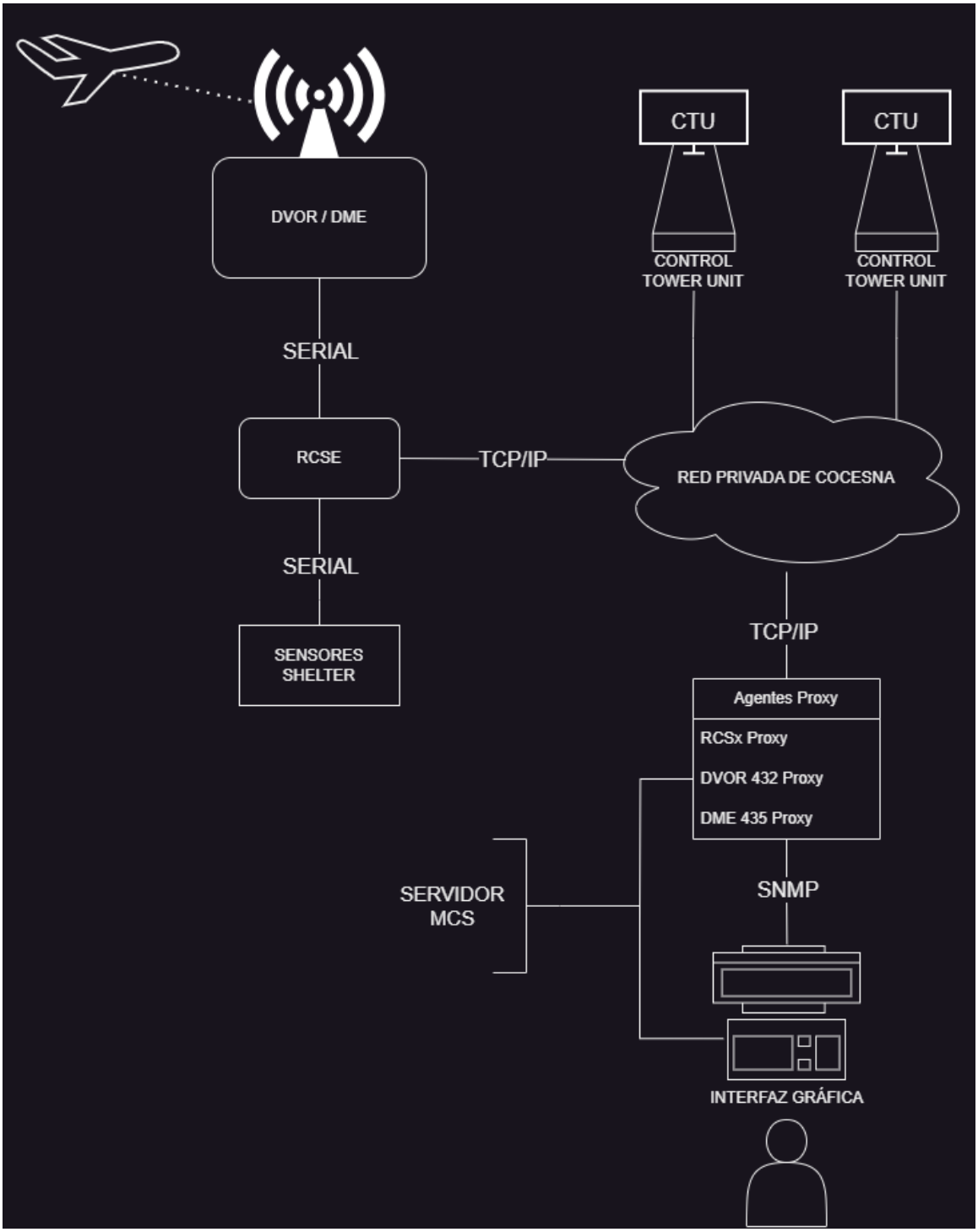


Figura 14: Topología definida despúes del proceso de investigación.

Este análisis permitió descubrir los protocolos empleados, así como la arquitectura y el esquema general del funcionamiento del MCS. Gracias a estos hallazgos, fue posible do-

cumentar de manera precisa las interacciones entre los componentes del sistema y comprender en detalle la estructura subyacente, lo que constituyó un avance significativo para la implementación y optimización del proyecto.

Una vez que la información proveniente de la radioayuda llega al servidor MCS, esta es procesada e interpretada por el Agente Proxy correspondiente. El agente desglosa los datos en un formato compatible con el protocolo SNMP, lo que permite su integración con el sistema de monitoreo. Dado que los agentes proxy son virtuales y comparten la misma interfaz de red que el servidor físico, su acceso se realiza utilizando la IP del servidor y el puerto SNMP configurado durante la creación del agente proxy. En este proyecto, la IP del servidor MCS es 172.20.12.126, y el puerto asignado al agente proxy del DVOR es 8225. Por lo tanto, el propio MCS, que opera en el mismo entorno que los agentes proxy, accede a la información de la radioayuda simulada a través de la dirección localhost:8225. Este esquema asegura que el sistema pueda interactuar con las radioayudas como si estuvieran directamente integradas en un entorno SNMP estándar.

En términos generales, los Agentes Proxy constituyen el componente encargado de la extracción y manejo de la información proveniente de las radioayudas. Por otro lado, el software con entorno gráfico del MCS tiene la función de consultar directamente a los agentes proxy, obtener los datos procesados y presentarlos de manera accesible y comprensible para el usuario. Sin embargo, este enfoque carece de escalabilidad y es poco eficiente ya que requiere de intervención humana constante para la detección de eventos y generación de informes de monitoreo. El MCS resulta útil para el control de las radioayudas, establecimiento de parámetros, conmutación de transmisores, etc. Pero es poco eficaz para el monitoreo.

El MCS y los Agentes Proxy se comunican utilizando el protocolo SNMPv3, lo que garantiza un intercambio de datos seguro mediante el uso de credenciales y métodos avanzados de autenticación y encriptación. Sin embargo, se descubrió que la obtención de parámetros es posible únicamente si la con-

sulta se realiza localmente del servido al agente proxy virtual, utilizando las credenciales configuradas no fue posible realizar consultas desde una máquina externa. Dado que se trata de información confidencial y sensible, las credenciales específicas y los procedimientos detallados de configuración no pueden ser revelados. Sin embargo, es fundamental resaltar que la correcta configuración de estos parámetros de seguridad es un paso crucial en el desarrollo del proyecto, ya que asegura la integridad y protección de los datos transmitidos entre los componentes del sistema, además, representa el último paso necesario para la extracción de parámetros críticos del DVOR AILA de manera automática y sin utilizar la interfaz gráfica del MCS.

CAPÍTULO 9

Desarrollo de solución

El primer paso en el desarrollo de la solución a la problemática planteada consistió en un estudio detallado de las rutinas de mantenimiento realizadas en el DVOR AILA. Este análisis incluyó la identificación de todos los parámetros de interés, su relevancia dentro del sistema y la justificación de su monitoreo. Como parte de este proceso, se llevó a cabo una rutina de mantenimiento en compañía de un técnico especializado. Este ejercicio tuvo como objetivo principal la familiarización con el procedimiento, lo cual permitió comprender las necesidades operativas y técnicas del sistema, así como identificar áreas clave para la optimización y automatización del mantenimiento.

Adicionalmente, debido a que en ningún momento se proporcionaron los archivos MIB correspondientes al DVOR432 (modelo instalado en el AILA), fue necesario realizar un análisis manual dentro del entorno del MCS. Este procedimiento consistió en recorrer todos los parámetros de interés utilizados durante las rutinas de mantenimiento, identificarlos, obtener sus respectivos OIDs y determinar los factores de conversión aplicables. Este proceso fue fundamental para garantizar la

precisión en la interpretación de los datos y para habilitar la integración del monitoreo automático de estos parámetros en la solución propuesta.

Los parámetros de interés están clasificados en la rutina protocolaria como lecturas de Monitores, Transmisores y CSB-LRCI (*Central Storage Battery* y *Low Ripple Current Indicator*, refiriéndose a los sistemas de suministro de energía del DVOR).

Descripción	**OID (Terminación .1 y .2 según el ID de monitor)**	**Factor de conversión**	**Unidad**	**Tolerancia**	**Valor normal**
Nivel RF Dipolo 1	1.3.6.1.4.1.30909.99.12.4.23.1	1	%	± 15 %	100 %
Azimuth Dipolo 1	1.3.6.1.4.1.30909.99.12.4.24.1	1	°	± 1°	263.5°
Índice de Modulación Frecuencia Dipolo 1	1.3.6.1.4.1.30909.99.12.4.25.1	1	Adimensional	± 1 %	16.0
Nivel RF Dipolo 2	1.3.6.1.4.1.30909.99.12.4.26.1	1	%	± 15 %	100 %
Azimuth Dipolo 2	1.3.6.1.4.1.30909.99.12.4.27.1	1/100	°	± 1°	175.0°
Índice de Modulación Frecuencia Dipolo 2	1.3.6.1.4.1.30909.99.12.4.28.1	1	Adimensional	± 1 %	16.0
Profundidad de Modulación 30Hz AM	1.3.6.1.4.1.30909.99.12.4.19.1	1/10	%	± 2 %	30 %
Profundidad de Modulación 9960Hz AM	1.3.6.1.4.1.30909.99.12.4.20.1	1/10	%	± 2 %	30 %
Frecuencia de Portadora	1.3.6.1.4.1.30909.99.12.4.19.1	1/10	kHz	± 1kHz	114900.0
Frecuencia USB (*Upper Sideband*)	1.3.6.1.4.1.30909.99.12.4.15.1	1/10	kHz	± 1kHz	114910.0
Frecuencia LSB (*Lower Sideband*)	1.3.6.1.4.1.30909.99.12.4.16.1	1/10	kHz	± 1kHz	114890.0
Distorsión USB-LSB	1.3.6.1.4.1.30909.99.12.4.14.1	1/10	%	-	>40 %

Cuadro 1: Información de parámetros de monitores

Descripción	**OID**	**Factor de conversión**	**Unidad**	**Tolerancia**	**Valor normal**
Igualdad de voltaje en celdas de batería	1.3.6.1.4.1.30909.99.12.6.13.2	1/100	V	± 2	0.00
Voltaje total de la batería	1.3.6.1.4.1.30909.99.12.6.13.3	1/100	V	± 2	54V
Corriente de batería	1.3.6.1.4.1.30909.99.12.6.13.4	1/100	A	[-1,20]	-
Corriente VOR	1.3.6.1.4.1.30909.99.12.6.13.5	1/100	A	[4,29]	-
Estado BCPS (*Battery Charging Power Supply*)	1.3.6.1.4.1.30909.99.12.1.73	-	-	-	1 (ON)
CSL VDD	1.3.6.1.4.1.30909.99.12.6.13.16	1/100	V	± 1	15.0
CSL VSS	1.3.6.1.4.1.30909.99.12.6.13.17	1/100	V	± 1	-15.0
Referencia CSL	1.3.6.1.4.1.30909.99.12.6.13.15	1/100	V	± 0.25	-2.40
Estado de la batería	1.3.6.1.4.1.30909.99.12.6.12.88	-	-	-	0 (Normal)

Cuadro 2: Información de parámetros del CSB y LRCI

Descripción	**OID (Terminación .1 y .2 según el ID de transmisor)**	**Factor de conversión**	**Unidad**	**Tolerancia**	**Valor normal**
Fase RF USB (*Upper Sideband*)	1.3.6.1.4.1.30909.99.12.5.13.13	1/10	°	± 4°	160°
Estado BIT (*Built-in-Test*) ADC1	1.3.6.1.4.1.30909.99.12.5.17.12	-	-	-	0 (OK)
Estado BIT (*Built-in-Test*) ADC2	1.3.6.1.4.1.30909.99.12.5.17.13	-	-	-	0 (OK)

Cuadro 3: Información de parámetros de transmisores

9.1. Estructura de la aplicación de contenedores Docker

Con base en el trasfondo de la investigación, se propone el sistema mostrado en la figura 15. Esta solución se compone de una aplicación implementada en tres contenedores Docker, los cuales están alojados en un servidor físico Titán ubicado en la sala de equipos de la estación de Guatemala, con la dirección IP 172.20.12.129. Además, se incluye un servidor SMTP para el envío de alertas y reportes de monitoreo que está alojado en la sede central de Honduras con IP 172.16.11.28 y puerto 25.

Se realizó la integración de tres tipos de contenedores Docker: Python, PostgreSQL y Grafana.

```
# Use an official Python runtime as a parent image
FROM python:3.10

# Set the working directory in the container
WORKDIR /app

# Instalar texlive-latex-base y dependencias
 RUN apt-get update && \
    apt-get install -y texlive-latex-base && \
    apt-get install -y texlive-latex-extra && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Copiar los archivos necesarios a /app
COPY ./dataPoller.py /app/dataPoller.py
COPY ./reportGenerator.py /app/reportGenerator.py
COPY ./requirements.txt /app/requirements.txt
COPY ./start.sh /app/start.sh
```

```
19
20 # Instalar las dependencias de Python
21 RUN pip install --no-cache-dir -r /app/
     requirements.txt
22
23 # Dar permisos de ejecucion al script start.sh
24 RUN chmod +x /app/start.sh
25
26 # Usar el script de shell como comando de inicio
27 CMD ["/app/start.sh"]
```

Listing 9.1: Dockerfile para el contenedor de Python.

```
1 paramiko==2.12.0    # For SSH connections
2 apscheduler==3.9.1  # For scheduling tasks
3 psycopg2-binary==2.9.7    # For PostgreSQL
     connections
```

Listing 9.2: Archivo de dependencias. Enumera las bibliotecas necesarias para los scripts del contenedor Python.

```
1 #!/bin/bash
2 python /app/dataPoller.py &
3 python /app/reportGenerator.py &
4
5 # Mantener el contenedor en ejecucion
6 wait
```

Listing 9.3: Archivo encargado de ejecutar ambos scripts dentro del contenedor de Python.

```
1 version: '3.8'
2
3 services:
4   postgres:
```

```
    image: postgres:16
    container_name: postgres
    environment:
      POSTGRES_USER: monitoreogt
      POSTGRES_PASSWORD: ***# Censurado por
        motivos de confidencialidad
      POSTGRES_DB: aur-dvor432
      PGDATA: /var/lib/postgresql/data/pgdata
    ports:
      - "5432:5432"
    volumes:
      - ./postgres_data:/var/lib/postgresql/data/
        pgdata
      - ./db/create-schema.sql:/docker-entrypoint-
        initdb.d/create-schema.sql
    networks:
      - app-network

  python-app:
    build:          # Use the Dockerfile to build
      the image
      context: .    # Path to the Dockerfile (
        current directory)
    container_name: python-app
    volumes:
      - .:/app      # Mount the entire project
        directory inside the container
    depends_on:
      - postgres
```

```
    environment:
      DB_HOST: postgres
      DB_PORT: 5432
      DB_USER: monitoreogt
      DB_PASSWORD: ***# Censurado por motivos de
        confidencialidad
      DB_NAME: aur-dvor432
    networks:
      - app-network

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_USER=monitoreogt_admin
      - GF_SECURITY_ADMIN_PASSWORD=*** # Censurado
        por motivos de confidencialidad
      - GF_SMTP_ENABLED=true
      - GF_SMTP_HOST=172.16.11.28:25
      - GF_SMTP_SKIP_VERIFY=true
      - GF_SMTP_FROM_ADDRESS=alertas.
        mcsguatemala@cocesna.org
      - GF_SMTP_FROM_NAME="Alertas MCS Guatemala"
      - GF_SMTP_EHLO_IDENTITY=cocesna.org
      - GF_SMTP_STARTTLS_POLICY=NoStartTLS
    volumes:
      - ./grafana_data:/var/lib/grafana
```

```
54      depends_on:
55        - postgres
56      networks:
57        - app-network
58
59  networks:
60    app-network:
61      driver: bridge
```

Listing 9.4: Archivo responsable de la estructura del *stack* de Dockers y servicios relacionados

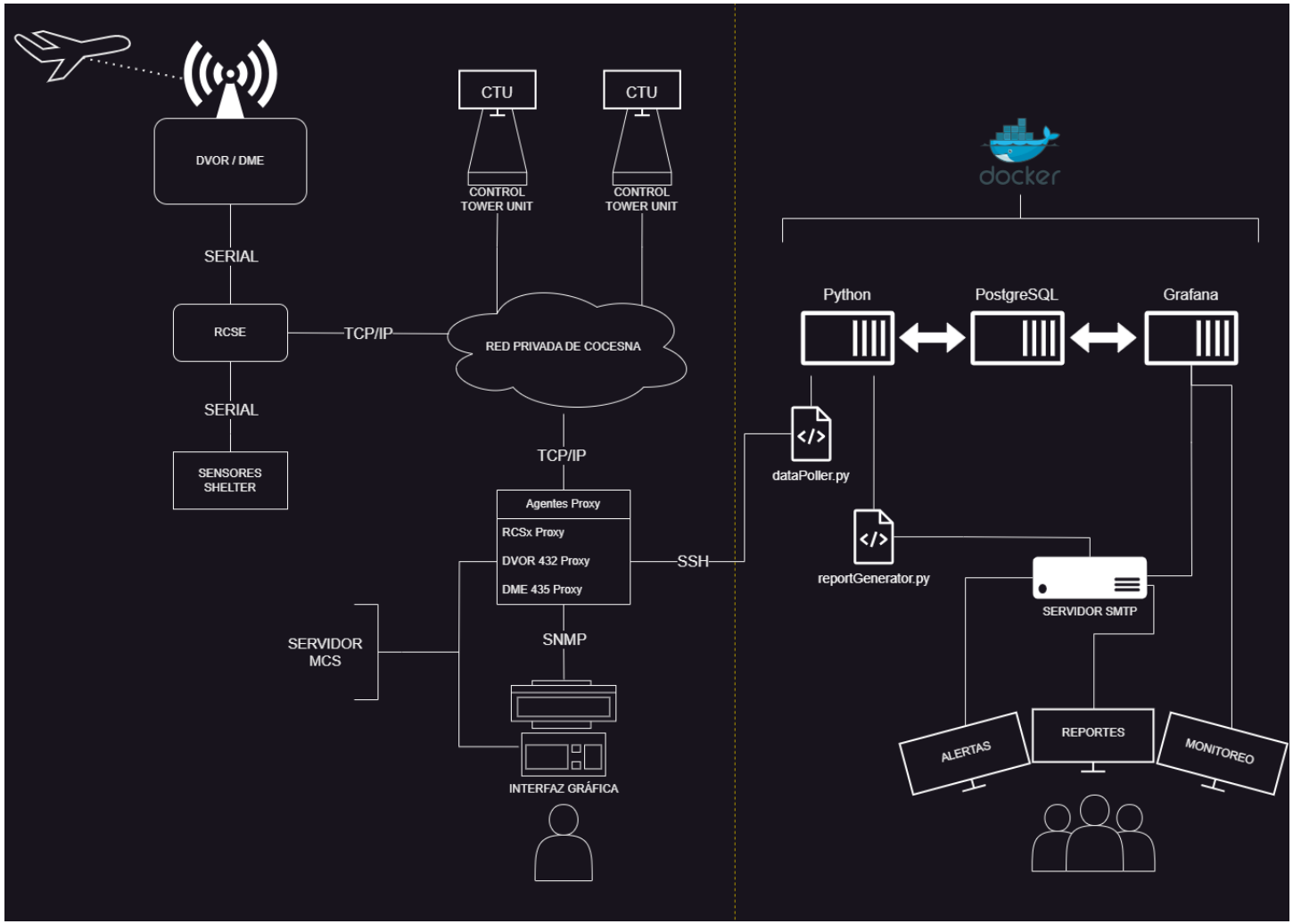


Figura 15: Topología final con la implementación de la solución propuesta en este proyecto. La integración es representada por el conjunto de elementos en la parte derecha de la topología.

9.1.1. Python

Este Docker está diseñado para gestionar los scripts de Python esenciales para el correcto funcionamiento del sistema. En su configuración, se implementan dos scripts principales que trabajan en conjunto para cumplir con los requerimientos del proyecto.

Polling de información y almacenamiento en base de datos

A través de una conexión SSH con el servidor MCS, el script utiliza SNMP local para extraer datos en tiempo real sobre parámetros críticos del DVOR AILA. Estos datos son procesados y almacenados automáticamente en una base de datos PostgreSQL, facilitando su análisis y generación de reportes. El script incluye un sistema de agendamiento con apscheduler para ejecutar tareas periódicas y utiliza bibliotecas como paramiko para establecer la conexión SSH y psycopg2 para interactuar con la base de datos. Adicionalmente, implementa un manejo robusto de errores y genera logs detallados, asegurando la trazabilidad y confiabilidad del sistema. Esto contribuye a optimizar procesos manuales y mejorar la eficiencia en la supervisión de equipos de navegación aérea.

Generación de reportes de monitoreo de manera automática

Genera reportes mensuales basados en datos extraídos del último registro realizado por `dataPoller.py` en la base de datos. Utilizando apscheduler, programa una tarea recurrente el primer día de cada mes para obtener la información más reciente, evaluar parámetros críticos y asignar estados de operación (NORMAL o ALARMA) según umbrales establecidos. Luego, elabora un reporte en formato LaTeX, lo compila en PDF, y lo envía automáticamente por correo electrónico a los destinatarios definidos mediante un servidor SMTP. Además,

integra un sistema de logs para el monitoreo de su ejecución en Docker.

```
'''
Codigo elaborado por Luis Jose Archila Madrid
para proyecto de graduacion: "Optimizacion de rutinas de mantenimiento y supervision para
sistemas VHF Omnidirectional Range de navegacion aerea" de la Universidad Del Valle de Guatemala
en colaboracion con COCESNA.

Integracion de conexion ssh hacia server MCS
Datamining utilizando snmp local dentro de server MCS
Almacenamiento de informacion en PostgreSQL DB

Inicio de desarrollo: Julio 2024
'''

import re # Para filtrado de stdout ssh y obtener la informacion necesaria
import time # Aux de agendador
import sys # Para parar el script si ocurre algun error (envia comandos al SO)
```

```
import logging # Para generacion de logs en Docker
import psycopg2 # Para conexion con PG DB
from paramiko import SSHClient, AutoAddPolicy, AuthenticationException,
  SSHException # Para sesion ssh con el server mcs
from apscheduler.schedulers.background import BackgroundScheduler #
  Agendador
from datetime import datetime # Formato de tiempo

# Configure logging
logging.basicConfig(
    level=logging.INFO, # Set the log level
    format='%(asctime)s - %(levelname)s - %(message)s', # Log message
      format
    handlers=[
        logging.StreamHandler(sys.stdout), # Output logs to stdout
    ]
)

'''
```

```
33 ------------------------------------------------------------------------------
34    DATA POLLING CLASS
35 ------------------------------------------------------------------------------
36 '''
37 class MCS_DataPoller:
38     #
          ============================================================================
39     # INIT
40     #
          ============================================================================
41     def __init__(self, ip_server, user_server, psswd_server, snmp_user,
          snmp_auth, snmp_priv, snmp_port, db_config):
42         self.ip_server = ip_server
43         self.user_server = user_server
44         self.psswd_server = psswd_server
```

```
45          self.snmp_user = snmp_user
46          self.snmp_auth = snmp_auth
47          self.snmp_priv = snmp_priv
48          self.snmp_port = snmp_port
49          self.db_config = db_config
50
51          self.oid_pos = 1
52          self.type_pos = 2
53          self.descr_pos = 3
54          self.unit_pos = 4
55          self.scal_pos = 5
56          self.llim_pos = 6
57          self.ulim_pos = 7
58
59          self.mon_columns = """timestamp, nivel_rf_dipolo_1,
              azimuth_dipolo_1, index_fm_dipolo_1, nivel_rf_dipolo_2,
              azimuth_dipolo_2,
60                              index_fm_dipolo_2, depth_modulation_30hz_am,
                                  depth_modulation_9960hz_am, carrier_frequency
```

```
                          , usb_frequency,
61                       lsb_frequency, distortion_usb_lsb"""
62      self.tx_columns = """timestamp, usb_rf_phase, adc1_bit_status,
          adc2_bit_status"""
63      self.csblcri_columns = """timestamp, battery_cell_equality,
          total_battey_voltage, battery_current, vor_currente, bcps_status,
           csl_vdd,
64                           csl_vss, csl_reference, battery_status"""
65      self.params_buffer = []
66
67      # Inicializacion de Scheduler
68      self.scheduler = BackgroundScheduler()
69
70      # Se obtiene la lista de parametros y su info
71      try:
72          (self.mon1_params_list, self.mon2_params_list, self.
              tx1_params_list, self.tx2_params_list, self.
              csblcri_params_list) = self.get_params_from_db()
73      except Exception as e:
```

```
74              self.params_list = None
75              logging.error(f"No es posible acceder a la informacion de la
                  base de datos Postgres: {e}")
76
77          # Periodicidad de la obtencion de datos
78          self.scheduler.add_job(self.data_polling, 'cron', minute='*', args
                =[self.mon1_params_list, self.mon2_params_list, self.
                tx1_params_list, self.tx2_params_list, self.csblcri_params_list])
79
80          # Start para scheduler
81          self.scheduler.start()
82          logging.info("El scheduler esta ejecutandose en segundo plano...")
83
84
85      #
            ============================================================================
86      # DB GET PARAMETERS
87      #
```

```
    ================================================================================

88  def get_params_from_db(self):
89      # Conexion y obtencion de lista de parametros de la base de datos
90      conn = None
91      cursor = None
92      try:
93          conn = psycopg2.connect(**self.db_config)
94          cursor = conn.cursor()
95          cursor.execute("SELECT * FROM parameters WHERE
              type_of_measurement = 'Monitor 1' ORDER BY id")
96          mon1_params_list = cursor.fetchall()
97          cursor.execute("SELECT * FROM parameters WHERE
              type_of_measurement = 'Monitor 2' ORDER BY id")
98          mon2_params_list = cursor.fetchall()
99          cursor.execute("SELECT * FROM parameters WHERE
              type_of_measurement = 'Transmisor 1' ORDER BY id")
100         tx1_params_list = cursor.fetchall()
101         cursor.execute("SELECT * FROM parameters WHERE
```

```
            type_of_measurement = 'Transmisor 2' ORDER BY id")
        tx2_params_list = cursor.fetchall()
        cursor.execute("SELECT * FROM parameters WHERE
            type_of_measurement = 'CSB (LCRI)' ORDER BY id")
        csblcri_params_list = cursor.fetchall()
        cursor.close()
        return (mon1_params_list, mon2_params_list, tx1_params_list,
            tx2_params_list, csblcri_params_list) # Se retorna una lista
            de tuplas, en donde cada tupla es un registro de la db (un
            parametro con toda su info)
    except Exception as e:
        logging.error(f"Error conectandose a la base de datos Postgres:
            {e}")
        return None
    finally:
        # Cerrar el cursor y la conexion
        if cursor:
            cursor.close()
        if conn:
```

```
115                 conn.close()
116
117     def save_param_in_db(self, table_name, columns, params):
118         conn = None
119         cursor = None
120         try:
121             conn = psycopg2.connect(**self.db_config)
122             cursor = conn.cursor()
123
124             insert_query_mon = f"""
125             INSERT INTO {table_name} ({columns})
126             VALUES (NOW(), {', '.join(['%s'] * len(params))})
127             """
128             cursor.execute(insert_query_mon, params)
129             conn.commit()
130             logging.info(f"Se ha guardado el registro correctamente en {
                table_name}")
131         except Exception as e:
132             logging.error(f"Error conectandose a la base de datos Postgres,
```

```
            no ha sido posible guardar el parametro: {e}")
    finally:
        # Cerrar el cursor y la conexion
        if cursor:
            cursor.close()
        if conn:
            conn.close()

#
    ================================================================================

# SHH CONNECTION, DATA POLLING AND DB STORAGE
#
    ================================================================================

def data_polling(self, mon1_params_list, mon2_params_list,
    tx1_params_list, tx2_params_list, csblcri_params_list):
     # Conexion SSH hacia el servidor MCS
     try: # Intentar establecer conexion con el servidor
```

```
146         sshclient = SSHClient()
147         sshclient.set_missing_host_key_policy(AutoAddPolicy())
148         sshclient.load_system_host_keys()
149         sshclient.connect(self.ip_server, username=self.user_server,
              password=self.psswd_server)
150
151         # Establece la fecha actual - Utilizada principalmente para los
              reportes
152         #current_time = datetime.now().strftime("%H:%M:%S %Y-%m-%d")
153
154         ''' ---------- Manejo de informacion, respuestas SNMP y
              almacenamiento en DB ----------'''
155         '''Monitor 1'''
156         for element in mon1_params_list:
157             # Enviar comando SNMP a travez de la terminal del servidor
158             stdin, stdout, stderr = sshclient.exec_command(f'snmpwalk -l
                  AuthPriv -u {self.snmp_user} -A {self.snmp_auth} -X {self
                .snmp_priv} {self.ip_server}:{self.snmp_port} {element[
                self.oid_pos]}')
```

```
159
160             # Recepcion y manejo de la respuesta SNMP
161             if stdout.channel.recv_exit_status() == 0: # Espera a que
                  termine la recepcion
162                 output = stdout.read().decode("utf8") # Decodificacion de
                      la informacion
163                 match = re.search(r'INTEGER: (-?\d+)', output) # Filtra
                      la respuesta por solamente el valor numerico
164                 if match: # Si se obtiene un valor con sentido/esperado
165                     value = int(match.group(1)) * (element[self.scal_pos]
                          if element[self.scal_pos] is not None else 1) # Se
                          convierte de str a num y se multiplica por su
                         factor de escala
166                     self.params_buffer.append(value) # Se llena el buffer
                          con todos los valores de los parametros
167                     logging.info(f'Valor extraido: {value if int(match.
                        group(1)) != -2147483648 else "N/A"}{element[self.
                        unit_pos] if element[self.unit_pos] is not None
                        else ""}, correspondiente a {element[self.descr_pos
```

```
                ]} de {element[self.type_pos]}')
            else:
                # Guardar valor como NULL en db
                self.params_buffer.append('NULL')
                logging.error(f"No fue posible obtener un valor
                  correctamente para {element[self.descr_pos]} en {
                  element[self.type_pos]}")
        else:
            self.params_buffer.append('NULL')
            logging.error(f'Algo salio mal en la respuesta SNMP:\n{
              stderr.read().decode("utf8")}')
    # Guardar en DB
    self.save_param_in_db('monitor_1', self.mon_columns, self.
      params_buffer)
    self.params_buffer.clear() # Vaciar el buffer de parametros

    '''Monitor 2'''
    for element in mon2_params_list:
        # Enviar comando SNMP a travez de la terminal del servidor
```

```
182             stdin, stdout, stderr = sshclient.exec_command(f'snmpwalk -l
                    AuthPriv -u {self.snmp_user} -A {self.snmp_auth} -X {self
                    .snmp_priv} {self.ip_server}:{self.snmp_port} {element[
                    self.oid_pos]}')
183
184             # Recepcion y manejo de la respuesta SNMP
185             if stdout.channel.recv_exit_status() == 0: # Espera a que
                    termine la recepcion
186                 output = stdout.read().decode("utf8") # Decodificacion de
                        la informacion
187                 match = re.search(r'INTEGER: (-?\d+)', output) # Filtra
                      la respuesta por solamente el valor numerico
188                 if match: # Si se obtiene un valor con sentido/esperado
189                     value = int(match.group(1)) * (element[self.scal_pos]
                            if element[self.scal_pos] is not None else 1) # Se
                            convierte de str a num y se multiplica por su
                          factor de escala
190                     self.params_buffer.append(value) # Se llena el buffer
                            con todos los valores de los parametros
```

```
191                 logging.info(f'Valor extraido: {value if int(match.
                      group(1)) != -2147483648 else "N/A"}{element[self.
                      unit_pos] if element[self.unit_pos] is not None
                      else ""}, correspondiente a {element[self.descr_pos
                      ]} de {element[self.type_pos]}')
192             else:
193                 # Guardar valor como NULL en db
194                 self.params_buffer.append('NULL')
195                 logging.error(f"No fue posible obtener un valor
                      correctamente para {element[self.descr_pos]} en {
                      element[self.type_pos]}")
196         else:
197             self.params_buffer.append('NULL')
198             logging.error(f'Algo salio mal en la respuesta SNMP:\n{
                  stderr.read().decode("utf8")}')
199     # Guardar en DB
200     self.save_param_in_db('monitor_2', self.mon_columns, self.
          params_buffer)
201     self.params_buffer.clear() # Vaciar el buffer de parametros
```

```
202
203         '''Transmisor 1'''
204         for element in tx1_params_list:
205             # Enviar comando SNMP a travez de la terminal del servidor
206             stdin, stdout, stderr = sshclient.exec_command(f'snmpwalk -l
                    AuthPriv -u {self.snmp_user} -A {self.snmp_auth} -X {self
                    .snmp_priv} {self.ip_server}:{self.snmp_port} {element[
                    self.oid_pos]}')
207
208             # Recepcion y manejo de la respuesta SNMP
209             if stdout.channel.recv_exit_status() == 0: # Espera a que
                    termine la recepcion
210                 output = stdout.read().decode("utf8") # Decodificacion de
                        la informacion
211                 match = re.search(r'INTEGER: (-?\d+)', output) # Filtra
                        la respuesta por solamente el valor numerico
212                 if match: # Si se obtiene un valor con sentido/esperado
213                     value = int(match.group(1)) * (element[self.scal_pos]
                            if element[self.scal_pos] is not None else 1) # Se
```

```
                convierte de str a num y se multiplica por su
                factor de escala
214             self.params_buffer.append(value) # Se llena el buffer
                con todos los valores de los parametros
215             logging.info(f'Valor extraido: {value if int(match.
                group(1)) != -2147483648 else "N/A"}{element[self.
                unit_pos] if element[self.unit_pos] is not None
                else ""}, correspondiente a {element[self.descr_pos
                ]} de {element[self.type_pos]}')
            else:
216
217             # Guardar valor como NULL en db
218             self.params_buffer.append('NULL')
219             logging.error(f"No fue posible obtener un valor
                correctamente para {element[self.descr_pos]} en {
                element[self.type_pos]}")
        else:
220
221         self.params_buffer.append('NULL')
222         logging.error(f'Algo salio mal en la respuesta SNMP:\n{
            stderr.read().decode("utf8")}')
```

```
223             # Guardar en DB
224             self.save_param_in_db('transmitter_1', self.tx_columns, self.
                  params_buffer)
225             self.params_buffer.clear() # Vaciar el buffer de parametros
226
227             '''Transmisor 2'''
228             for element in tx2_params_list:
229                 # Enviar comando SNMP a travez de la terminal del servidor
230                 stdin, stdout, stderr = sshclient.exec_command(f'snmpwalk -l
                      AuthPriv -u {self.snmp_user} -A {self.snmp_auth} -X {self
                     .snmp_priv} {self.ip_server}:{self.snmp_port} {element[
                     self.oid_pos]}')
231
232                 # Recepcion y manejo de la respuesta SNMP
233                 if stdout.channel.recv_exit_status() == 0: # Espera a que
                      termine la recepcion
234                     output = stdout.read().decode("utf8") # Decodificacion de
                          la informacion
235                     match = re.search(r'INTEGER: (-?\d+)', output) # Filtra
```

```
            la respuesta por solamente el valor numerico
        if match: # Si se obtiene un valor con sentido/esperado
236         value = int(match.group(1)) * (element[self.scal_pos]
237             if element[self.scal_pos] is not None else 1) # Se
                convierte de str a num y se multiplica por su
                factor de escala
238         self.params_buffer.append(value) # Se llena el buffer
                con todos los valores de los parametros
239         logging.info(f'Valor extraido: {value if int(match.
                group(1)) != -2147483648 else "N/A"}{element[self.
                unit_pos] if element[self.unit_pos] is not None
                else ""}, correspondiente a {element[self.descr_pos
                ]} de {element[self.type_pos]}')
        else:
240         # Guardar valor como NULL en db
241         self.params_buffer.append('NULL')
242         logging.error(f"No fue posible obtener un valor
243           correctamente para {element[self.descr_pos]} en {
              element[self.type_pos]}")
```

```
244                 else:
245                     self.params_buffer.append('NULL')
246                     logging.error(f'Algo salio mal en la respuesta SNMP:\n{
                          stderr.read().decode("utf8")}')
247             # Guardar en DB
248             self.save_param_in_db('transmitter_2', self.tx_columns, self.
                  params_buffer)
249             self.params_buffer.clear() # Vaciar el buffer de parametros
250
251
252             '''CSB/LCRI'''
253             for element in csblcri_params_list:
254                 # Enviar comando SNMP a travez de la terminal del servidor
255                 stdin, stdout, stderr = sshclient.exec_command(f'snmpwalk -l
                      AuthPriv -u {self.snmp_user} -A {self.snmp_auth} -X {self
                      .snmp_priv} {self.ip_server}:{self.snmp_port} {element[
                      self.oid_pos]}')
256
257                 # Recepcion y manejo de la respuesta SNMP
```

```
258                 if stdout.channel.recv_exit_status() == 0: # Espera a que
                      termine la recepcion
259                     output = stdout.read().decode("utf8") # Decodificacion de
                          la informacion
260                     match = re.search(r'INTEGER: (-?\d+)', output) # Filtra
                        la respuesta por solamente el valor numerico
261                     if match: # Si se obtiene un valor con sentido/esperado
262                         value = int(match.group(1)) * (element[self.scal_pos]
                            if element[self.scal_pos] is not None else 1) # Se
                            convierte de str a num y se multiplica por su
                          factor de escala
263                         self.params_buffer.append(value) # Se llena el buffer
                            con todos los valores de los parametros
264                         logging.info(f'Valor extraido: {value if int(match.
                          group(1)) != -2147483648 else "N/A"}{element[self.
                          unit_pos] if element[self.unit_pos] is not None
                          else ""}, correspondiente a {element[self.descr_pos
                          ]} de {element[self.type_pos]}')
265                     else:
```

```
266                     # Guardar valor como NULL en db
267                     self.params_buffer.append('NULL')
268                     logging.error(f"No fue posible obtener un valor
                          correctamente para {element[self.descr_pos]} en {
                          element[self.type_pos]}")
269                 else:
270                     self.params_buffer.append('NULL')
271                     logging.error(f'Algo salio mal en la respuesta SNMP:\n{
                          stderr.read().decode("utf8")}')
272             # Guardar en DB
273             self.save_param_in_db('csb_lcri', self.csblcri_columns, self.
                  params_buffer)
274             self.params_buffer.clear() # Vaciar el buffer de parametros
275
276         # Si ocurre algun error al establecer la sesion ssh
277         except AuthenticationException:
278             logging.error("Error en la autenticacion de la conexion del
                  servidor MCS")
279         except SSHException as ssh_exception:
```

```
280             logging.error(f"No es posible establecer conexion con el
                  servidor MCS: {ssh_exception}")
281         except Exception as e:
282             logging.error(f"Ha ocurrido un error desconocido al intentar
                  establecer sesion con el servidor MCS: {e}")
283         finally:
284             # Cerrar conexion SSH
285             if sshclient:
286                 sshclient.close()
287             if stdin:
288                 stdin.close()
289             if stdout:
290                 stdout.close()
291             if stderr:
292                 stderr.close()
293
294
295
296
```

```
297
298
299     def run(self):
300         # Mantiene el script corriendo indefinidamente
301         try:
302             while True:
303                 time.sleep(1)
304         except (KeyboardInterrupt, SystemExit):
305             self.shutdown()
306
307     def shutdown(self):
308         # Apaga el scheduler
309         self.scheduler.shutdown()
310         logging.info("Scheduler detenido.")
311
312 '''
313 ------------------------------------------------------------------------------
314    MAIN
```

```
--------------------------------------------------------------------------------------
'''
if __name__ == "__main__":
    logging.info("Programa inicializado")

    # Definicion de credenciales MCS
    IP_SERVER = '172.20.12.126'
    USER_SERVER = '***' # Censurado por motivos de confidencialidad
    PSSWD_SERVER = '***' # Censurado por motivos de confidencialidad
    SNMP_USER = 'mcs'
    SNMP_AUTH = '***' # Censurado por motivos de confidencialidad
    SNMP_PRIV = '***' # Censurado por motivos de confidencialidad
    SNMP_PORT = '8225'

    # Definicion de credenciales DB
    DB_CONFIG = {
        'dbname': 'aur-dvor432',
        'user': 'monitoreogt',
```

```
        'password': '***', # Censurado por motivos de confidencialidad
        'host': 'postgres',
        'port': '5432'
    }

    # Crea y ejecuta la instancia de la clase para monitoreo
    poller_instance = MCS_DataPoller(IP_SERVER, USER_SERVER, PSSWD_SERVER,
      SNMP_USER, SNMP_AUTH, SNMP_PRIV, SNMP_PORT, DB_CONFIG)
    poller_instance.run()
```

Listing 9.5: Script de obtención de datos críticos para DVOR432 La Aurora y almacenamiento en base de datos


```
'''
Codigo elaborado por Luis Jose Archila Madrid
para proyecto de graduacion: "Optimizacion de rutinas de mantenimiento y
  supervision para
sistemas VHF Omnidirectional Range de navegacion aerea" de la Universidad
  Del Valle de Guatemala
en colaboracion con COCESNA.

Generacion automatica de rutinas de mantenimiento DVOR432 AUR
```

```
Obtencion mensual de datos en DB
Generacion de documento PDF a partir de LaTeX

Inicio de desarrollo: Septiembre 2024
'''

from apscheduler.schedulers.blocking import BlockingScheduler # Agendador
from apscheduler.triggers.cron import CronTrigger # Trigger Agendador
import logging # Para generacion de logs en Docker
import psycopg2 # Para conexion con PG DB
from datetime import datetime # Formato de tiempo
import re # Para filtrado y llenado de documento LaTeX
import subprocess # Abrir subproceso para
import os # generacion de pdf usando la terminal de SO
# Para envio de correos electronicos:
import smtplib
from email.message import EmailMessage
from email.mime.application import MIMEApplication

```

```
27
28 # Configurar el logging
29 logging.basicConfig(level=logging.INFO)
30 logger = logging.getLogger(__name__)
31
32 '''
33 ------------------------------------------------------------------------------
34    REPORT GENERATOR CLASS
35 ------------------------------------------------------------------------------
36 '''
37
38 class generadorRtunas:
39     #
          ============================================================================
40     # INIT
41     #
```

```
    ================================================================================

42  def __init__(self, db_config):
43
44      self.db_config = db_config
45
46      # Para generacion de reportes
47      self.timestamp = None
48      self.template_file = "templateRutinas/FormatoRutina.txt"
49      self.output_directory = "Rutinas-DVOR432-AUR"
50      if not os.path.exists(self.output_directory):
51          os.makedirs("Rutinas-DVOR432-AUR")
52
53      # Para ejecucion periodica
54      self.scheduler = BlockingScheduler()
55      self._configure_jobs()
56
57      self.remitente = 'reportes.mcsguatemala@cocesna.org'
58      self.destinatarios = ["***"] # Censurado por motivos de
```

```
59          confidencialidad
            self.contenido = 'Este en un correo autogenerado, por favor no
              responder. \n\n'
60          self.smtp_server = '172.16.11.28'
61          self.smtp_port = 25
62
63          #self.generate_pdf(self.report_name)
64
65
66
67      #
          ============================================================================
68      # CONFIGURACION DE PERIODICIDAD
69      #
          ============================================================================
70      def _configure_jobs(self):
71          cron_trigger = CronTrigger(day='1', hour='8', minute='0') # Primer
```

```
        dia de cada mes a las 08:00 horas UTC-6
72      #cron_trigger = CronTrigger(hour='*')
73      self.scheduler.add_job(self.rutinaMensual, trigger=cron_trigger, id
          ="monthly_task", replace_existing=True)
74      logger.info("Tarea mensual programada para ejecutarse el primer dia
           de cada mes.")
75
76  #
      ============================================================================
77  # GENERACION Y ENVIO DE RUTINA MENSUALMENTE
78  #
      ============================================================================

79  def rutinaMensual(self): # ----- La funcion principal -----
80      #self.queryDB()
81      self.fechaActual = datetime.now()
82      self.fechaActualstr = self.fechaActual.strftime("%d-%m-%Y")
83      self.report_name = f"Reporte de Monitoreo DVOR432-AUR {self.
```

```
        fechaActualstr}"
        self.generate_pdf(self.report_name)
        logger.info("Reporte de monitoreo generado.")

    #
    ================================================================================

    # OBTENER ULTIMO REGISTRO DE CADA TABLA DE DB
    #
    ================================================================================

    def queryDB(self):
        # Conexion y obtencion de lista de parametros de la base de datos
        conn = None
        cursor = None
        try:
            conn = psycopg2.connect(**self.db_config)
            cursor = conn.cursor()
            cursor.execute("SELECT * FROM monitor_1 ORDER BY timestamp DESC
```

```
            LIMIT 1") #Cambiar query para obtener el ultimo registro
98          monitor1 = cursor.fetchall()
99          cursor.execute("SELECT * FROM monitor_2 ORDER BY timestamp DESC
              LIMIT 1")
100         monitor2 = cursor.fetchall()
101         cursor.execute("SELECT * FROM transmitter_1 ORDER BY timestamp
              DESC LIMIT 1")
102         transmisor1 = cursor.fetchall()
103         cursor.execute("SELECT * FROM transmitter_2 ORDER BY timestamp
              DESC LIMIT 1")
104         transmisor2 = cursor.fetchall()
105         cursor.execute("SELECT * FROM csb_lcri ORDER BY timestamp DESC
              LIMIT 1")
106         csblcri = cursor.fetchall()
107         cursor.close()
108         #return (monitor1, monitor2, transmisor1, transmisor2, csblcri)
              # Se retorna una lista de tuplas, en donde cada tupla es un
              registro de la db (un parametro con toda su info)
109     except Exception as e:
```

```
110             logging.error(f"Error conectandose a la base de datos Postgres:
                  {e}")
111             return None
112         finally:
113             # Cerrar el cursor y la conexion
114             if cursor:
115                 cursor.close()
116             if conn:
117                 conn.close()
118 
119             # Diccionario con los valores a reemplazar en la plantilla
120             self.timestamp = csblcri[0][1]
121             lecturas = {
122                 "mon1_nivelRFDipolo1":   round(monitor1[0][2], 2),
123                 "mon1_azimuthDipolo1":   round(monitor1[0][3], 2),
124                 "mon1_indexFMDipolo1":   round(monitor1[0][4], 2),
125                 "mon1_nivelRFDipolo2":   round(monitor1[0][5], 2),
126                 "mon1_azimuthDipolo2":   round(monitor1[0][6], 2),
127                 "mon1_indexFMDipolo2":   round(monitor1[0][7], 2),
```

```
128             "mon1_modulacion30HzAM": round(monitor1[0][8], 2),
129             "mon1_modulacion9960HzAM": round(monitor1[0][9], 2),
130             "mon1_carrierFreq":       round(monitor1[0][10], 2),
131             "mon1_USBFreq":           round(monitor1[0][11], 2),
132             "mon1_LSBFreq":           round(monitor1[0][12], 2),
133             'mon1_distorsionUSBLSB': round(monitor1[0][13], 2),
134
135             "mon2_nivelRFDipolo1":    round(monitor2[0][2], 2),
136             "mon2_azimuthDipolo1":    round(monitor2[0][3], 2),
137             "mon2_indexFMDipolo1":    round(monitor2[0][4], 2),
138             "mon2_nivelRFDipolo2":    round(monitor2[0][5], 2),
139             "mon2_azimuthDipolo2":    round(monitor2[0][6], 2),
140             "mon2_indexFMDipolo2":    round(monitor2[0][7], 2),
141             "mon2_modulacion30HzAM": round(monitor2[0][8], 2),
142             "mon2_modulacion9960HzAM": round(monitor2[0][9], 2),
143             "mon2_carrierFreq":       round(monitor2[0][10], 2),
144             "mon2_USBFreq":           round(monitor2[0][11], 2),
145             "mon2_LSBFreq":           round(monitor2[0][12], 2),
146             'mon2_distorsionUSBLSB': round(monitor2[0][13], 2),
```

```
147
148     "nivelRFDipolo1_state":   '{009901} NORMAL' if (85<=round(
          monitor1[0][2], 2)<=115 and 85<=round(monitor2[0][2], 2)
          <=115)      else '{CB0000} ALARMA',
149     "azimuthDipolo1_state":   '{009901} NORMAL' if (262.5<=round
          (monitor1[0][3], 2)<=264.5 and 262.5<=round(monitor2
          [0][3], 2)<=264.5) else '{CB0000} ALARMA',
150     "indexFMDipolo1_state":   '{009901} NORMAL' if (15.84<=round
          (monitor1[0][4], 2)<=16.16 and 15.84<=round(monitor2
          [0][4], 2)<=16.16) else '{CB0000} ALARMA',
151     "nivelRFDipolo2_state":   '{009901} NORMAL' if (85<=round(
          monitor1[0][5], 2)<=115 and 85<=round(monitor2[0][5], 2)
          <=115)      else '{CB0000} ALARMA',
152     "azimuthDipolo2_state":   '{009901} NORMAL' if (174<=round(
          monitor1[0][6], 2)<=176 and 174<=round(monitor2[0][6], 2)
          <=176)    else '{CB0000} ALARMA',
153     "indexFMDipolo2_state":   '{009901} NORMAL' if (15.84<=round
          (monitor1[0][7], 2)<=16.16 and 15.84<=round(monitor2
          [0][7], 2)<=16.16) else '{CB0000} ALARMA',
```

```
154         "modulacion30HzAM_state": '{009901} NORMAL' if (28<=round(
              monitor1[0][8], 2)<=32 and 28<=round(monitor2[0][8], 2)
              <=32)        else '{CB0000} ALARMA',
155         "modulacion9960HzAM_state": '{009901} NORMAL' if (28<=round(
              monitor1[0][9], 2)<=32 and 28<=round(monitor2[0][9], 2)
              <=32)      else '{CB0000} ALARMA',
156         "carrierFreq_state":      '{009901} NORMAL' if (114899<=
              round(monitor1[0][10], 2)<=114901 and 114899<=round(
              monitor2[0][10], 2)<=114901) else '{CB0000} ALARMA',
157         "USBFreq_state":          '{009901} NORMAL' if (114909<=
              round(monitor1[0][11], 2)<=114911 and 114909<=round(
              monitor2[0][11], 2)<=114911) else '{CB0000} ALARMA',
158         "LSBFreq_state":          '{009901} NORMAL' if (114889<=
              round(monitor1[0][12], 2)<=114891 and 114889<=round(
              monitor2[0][12], 2)<=114891) else '{CB0000} ALARMA',
159         'distorsionUSBLSB_state': '{009901} NORMAL' if (40<round(
              monitor1[0][13], 2) and 40<round(monitor2[0][13], 2))
                            else '{CB0000} ALARMA',
160
```

```
    # ----------------------------------------------------------------

    "tx1_usbRFphase":         round(transmisor1[0][2], 2) if (round(transmisor1[0][2], 2) != -2147483648) else 'Tx OFF',
    "tx1_ADC1bitStatus":      'OK' if(transmisor1[0][3] == 0) else 'ERROR' if transmisor1[0][3] == 1 else 'Tx OFF',
    "tx1_ADC2bitStatus":      'OK' if(transmisor1[0][4] == 0) else 'ERROR' if transmisor1[0][4] == 1 else 'Tx OFF',

    "tx2_usbRFphase":         round(transmisor2[0][2], 2) if (round(transmisor2[0][2], 2) != -2147483648) else 'Tx OFF',
    "tx2_ADC1bitStatus":      'OK' if(transmisor2[0][3] == 0) else 'ERROR' if transmisor2[0][3] == 1 else 'Tx OFF',
    "tx2_ADC2bitStatus":      'OK' if(transmisor2[0][4] == 0) else 'ERROR' if transmisor2[0][4] == 1 else 'Tx OFF',

    "usbRFphase_state":       '{009901} NORMAL' if (156<=round(transmisor1[0][2], 2)<=164 and 156<=round(transmisor2[0][2], 2)<=164) else '{009901} NORMAL' if (round(
```

```
    transmisor1[0][2], 2) == -2147483648 or round(transmisor2
    [0][2], 2) == -2147483648) else '{CB0000} ALARMA',
"ADC1bitStatus_state":   '{CB0000} ALARMA' if (transmisor1
    [0][3]==1 and transmisor2[0][3]==1) else '{009901} NORMAL'
    ,
"ADC2bitStatus_state":   '{CB0000} ALARMA' if (transmisor1
    [0][4]==1 and transmisor2[0][4]==1) else '{009901} NORMAL'
    ,

# ------------------------------------------------

"csblcri_CellEquality":  round(csblcri[0][2], 2),
"csblcri_batteryVoltage": round(csblcri[0][3], 2),
"csblcri_batteryCurrent": round(csblcri[0][4], 2),
"csblcri_VORcurrent":    round(csblcri[0][5], 2),
"csblcri_BCPSstatus":    'ON' if (csblcri[0][6] == 1) else '
    OFF',
"csblcri_cslVDD":        round(csblcri[0][7], 2),
"csblcri_cslVSS":        round(csblcri[0][8], 2),
```

```
184             "csblcri_cslRef":          round(csblcri[0][9], 2),
185             "csblcri_batteryStatus": 'OK' if (csblcri[0][10] == 0) else
                  'ERROR',
186
187             "CellEquality_state":   '{009901} NORMAL' if (-2 <= round(
                  csblcri[0][2], 2) <= 2) else '{CB0000} ALARMA',
188             "batteryVoltage_state": '{009901} NORMAL' if (52 <= round(
                  csblcri[0][3], 2) <= 56) else '{CB0000} ALARMA',
189             "batteryCurrent_state": '{009901} NORMAL' if (-1 <= round(
                  csblcri[0][4], 2) <= 20) else '{CB0000} ALARMA',
190             "VORcurrent_state":     '{009901} NORMAL' if (4 <= round(
                  csblcri[0][5], 2) <= 29) else '{CB0000} ALARMA',
191             "BCPSstatus_state":     '{009901} NORMAL' if (csblcri[0][6]
                  == 1) else '{CB0000} ALARMA',
192             "cslVDD_state":         '{009901} NORMAL' if (14 <= round(
                  csblcri[0][7], 2) <= 16) else '{CB0000} ALARMA',
193             "cslVSS_state":         '{009901} NORMAL' if (-16 <= round(
                  csblcri[0][8], 2) <= -14) else '{CB0000} ALARMA',
194             "cslRef_state":         '{009901} NORMAL' if (2.15 <= round(
```

```
                csblcri[0][9], 2) <= 2.65) else '{CB0000} ALARMA',
            "batteryStatus_state": '{009901} NORMAL' if (csblcri[0][10]
                == 0) else '{CB0000} ALARMA',

            # ------------------------------------------------

            "timestamp": csblcri[0][1],
            "date": self.fechaActualstr
        }

        return lecturas

#
    ================================================================================

# GENERADOR DE PDF
#
    ================================================================================
```

```
208     def load_template(self): #
          ------------------------------------------------------------------------------------------

209         # Lee el contenido de la plantilla LaTeX en FormatoRutina.txt
210         with open(self.template_file, 'r') as file:
211             return file.read()
212
213     def replace_values(self, template_content, replacements): #
          ------------------------------------------------------
214         #Reemplaza los valores dentro de la plantilla LaTeX usando
              expresiones regulares.
215         # Los valores a reemplazar aparecen como <<valor>> en la plantilla
              latex
216         for placeholder, value in replacements.items():
217             template_content = re.sub(rf"<<{placeholder}>>", str(value),
                  template_content)
218         return template_content
219
220     def save_tex_file(self, tex_content, tex_filename): #
```

```
    ------------------------------------------------------------
221     #Guarda el contenido de LaTeX en un archivo .tex.
222     tex_path = os.path.join(self.output_directory, f"{tex_filename}.tex
          ")
223     with open(tex_path, 'w') as file:
224         file.write(tex_content)
225     return tex_path
226
227 def compile_pdf(self, tex_filename): #
    ------------------------------------------------------------------------

228     #Ejecuta pdflatex para compilar el archivo .tex a PDF.
229
230     # Asegurarse de que el nombre de archivo tenga la extension .tex
231     tex_path = os.path.join(f'../{self.output_directory}', tex_filename
          + ".tex")
232     try:
233         subprocess.run(["pdflatex", tex_path], cwd=self.output_directory
              , check=True)
```

```
234             print(f"PDF generado exitosamente: {tex_filename}.pdf")
235             return tex_filename
236         except subprocess.CalledProcessError as e:
237             print(f"Error al compilar el archivo LaTeX: {e}")
238 
239     # -----------------------------
240     # ENVIO DE CORREOS ELECTRONICOS
241     # -----------------------------
242     def send_email(self, ruta_pdf, destinatarios, asunto, cuerpo, remitente
        , smtp_server, smtp_port, smtp_user=None, smtp_password=None): #
        ************************************************************************

243         # Crear el mensaje de correo electronico
244         mensaje = EmailMessage()
245         mensaje['From'] = remitente
246         mensaje['To'] = ", ".join(destinatarios)
247         mensaje['Subject'] = asunto
248         mensaje.set_content(cuerpo)
249 
```

```
250         # Adjuntar el archivo PDF
251         with open(ruta_pdf, 'rb') as pdf_file:
252             pdf_data = pdf_file.read()
253             mensaje.add_attachment(pdf_data, maintype='application', subtype
                  ='pdf', filename=ruta_pdf.split('/')[-1])
254 
255         # Configurar y enviar el correo
256         try:
257             with smtplib.SMTP(smtp_server, smtp_port) as server:
258                 server.starttls() # Si el servidor SMTP usa TLS
259                 if smtp_user and smtp_password:
260                     server.login(smtp_user, smtp_password)
261                 server.send_message(mensaje)
262             print("Correo enviado exitosamente a todos los destinatarios.")
263         except Exception as e:
264             print(f"Error al enviar el correo: {e}")
265 
266 
267     def generate_pdf(self, output_filename="output"): #
```

```
-----------------------------------------------
    # Genera un PDF a partir de la plantilla LaTeX con los valores
      reemplazados.

    # Obtener valores desde DB
    replacements = self.queryDB()

    # Cargar plantilla
    template_content = self.load_template()

    # Reemplazar valores en la plantilla
    modified_content = self.replace_values(template_content,
      replacements)

    # Guardar el archivo .tex modificado
    tex_path = self.save_tex_file(modified_content, output_filename)

    # Compilar el archivo .tex a PDF
    file_name = self.compile_pdf(output_filename)
```

```
284
285
286
287         # Extensiones de archivos auxiliares
288         auxiliares = [f'{self.output_directory}/{file_name}.aux', f'{
                self.output_directory}/{file_name}.log',
289                       f'{self.output_directory}/{file_name}.out', f'{self
                          .output_directory}/{file_name}.tex']
290         # Eliminar los archivos auxiliares generados por LaTeX
291         for file in auxiliares:
292             if os.path.exists(file):
293                 os.remove(file)
294                 print(f"Archivo eliminado: {file}")
295
296         self.send_email((f'{self.output_directory}/{file_name}.pdf'),
                self.destinatarios, file_name, self.contenido, self.remitente
                , self.smtp_server, self.smtp_port)
297
298
```

```
299
300
301
302
303     def start(self):
304         #Inicia el scheduler para que comience a ejecutar las tareas.
305         logger.info("Iniciando el scheduler...")
306         self.scheduler.start()
307
308 # Punto de entrada principal
309 if __name__ == "__main__":
310
311     # Definicion de credenciales DB
312     DB_CONFIG = {
313         'dbname': 'aur-dvor432',
314         'user': 'monitoreogt',
315         'password': '***', # Censurado por motivos de confidencialidad
316         'host': 'postgres',
317         'port': '5432'
```

```
318     }
319
320     task_scheduler = generadorRtunas(DB_CONFIG)
321     task_scheduler.start()
```

Listing 9.6: Script de generación de reportes de monitoreo DVOR432 La Aurora

9.1.2. Postgres

Este Docker centraliza la gestión de datos al alojar una base de datos PostgreSQL donde se almacenan todos los parámetros monitoreados y extraídos por el contenedor de Python encargado de la recopilación. La base de datos también actúa como fuente de información para el despliegue visual en Grafana, permitiendo un monitoreo en tiempo real y facilitando el análisis de tendencias. Además, el Docker de Python automatiza la generación de reportes mensuales en PDF basados en estos datos, integrando visualización, análisis y documentación en una solución cohesiva.

```
1
2  -- ===== Parametros generales =====
3  CREATE TABLE IF NOT EXISTS parameters (
4      id SERIAL PRIMARY KEY,
5      oid VARCHAR,
6      type_of_measurement VARCHAR(255),
7      description VARCHAR(255),
8      unit_of_measure VARCHAR(255),
9      scaling_factor FLOAT,
10     lower_limit FLOAT,
11     upper_limit FLOAT
12 );
13
14 -- ===== Lecturas Monitores =====
15 CREATE TABLE IF NOT EXISTS monitor_1 (
16     id SERIAL PRIMARY KEY,
17     timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
18     nivel_rf_dipolo_1 FLOAT,
19     azimuth_dipolo_1 FLOAT,
20     index_fm_dipolo_1 FLOAT,
```

```
    nivel_rf_dipolo_2 FLOAT,
    azimuth_dipolo_2 FLOAT,
    index_fm_dipolo_2 FLOAT,
    depth_modulation_30hz_am FLOAT,
    depth_modulation_9960hz_am FLOAT,
    carrier_frequency FLOAT,
    usb_frequency FLOAT,
    lsb_frequency FLOAT,
    distortion_usb_lsb FLOAT
);

CREATE TABLE IF NOT EXISTS monitor_2 (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
    nivel_rf_dipolo_1 FLOAT,
    azimuth_dipolo_1 FLOAT,
    index_fm_dipolo_1 FLOAT,
    nivel_rf_dipolo_2 FLOAT,
    azimuth_dipolo_2 FLOAT,
    index_fm_dipolo_2 FLOAT,
    depth_modulation_30hz_am FLOAT,
    depth_modulation_9960hz_am FLOAT,
    carrier_frequency FLOAT,
    usb_frequency FLOAT,
    lsb_frequency FLOAT,
    distortion_usb_lsb FLOAT
);

-- ===== Lecturas Transmisores =====
```

```
CREATE TABLE IF NOT EXISTS transmitter_1 (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
    usb_rf_phase FLOAT,
    adc1_bit_status FLOAT,
    adc2_bit_status FLOAT
);

CREATE TABLE IF NOT EXISTS transmitter_2 (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
    usb_rf_phase FLOAT,
    adc1_bit_status FLOAT,
    adc2_bit_status FLOAT
);

-- ===== Lecturas CSB/LCRI =====
CREATE TABLE IF NOT EXISTS csb_lcri (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
    battery_cell_equality FLOAT,
    total_battey_voltage FLOAT,
    battery_current FLOAT,
    vor_currente FLOAT,
    bcps_status FLOAT,
    csl_vdd FLOAT,
    csl_vss FLOAT,
    csl_reference FLOAT,
    battery_status FLOAT
```

```
79 );
80
81
82 -- Inserta datos de todos los parametros
83 INSERT INTO parameters (oid, type_of_measurement,
     description, unit_of_measure, scaling_factor,
     lower_limit, upper_limit) VALUES
84     -- MONITORES
         ================================================

85     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.23.1', '
         Monitor 1', 'Nivel RF Dipolo 1', '%', 1.0,
         85.0, 115.0),
86     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.23.2', '
         Monitor 2', 'Nivel RF Dipolo 1', '%', 1.0,
         85.0, 115.0),
87     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.24.1', '
         Monitor 1', 'Azimuth Dipolo 1', 'deg', 0.01,
          262.5, 264.5),
88     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.24.2', '
         Monitor 2', 'Azimuth Dipolo 1', 'deg', 0.01,
          262.5, 264.5),
89     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.25.1', '
         Monitor 1', 'Index FM Dipolo 1', NULL, 0.1,
         15.84, 16.16), --***
90     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.25.2', '
         Monitor 2', 'Index FM Dipolo 1', NULL, 0.1,
         15.84, 16.16), --***
91
```

```
92      ('1.3.6.1.4.1.3090.99.12.4.1.6.1.26.1', '
          Monitor 1', 'Nivel RF Dipolo 2', '%', 1.0,
          85.0, 115.0),
93      ('1.3.6.1.4.1.3090.99.12.4.1.6.1.26.2', '
          Monitor 2', 'Nivel RF Dipolo 2', '%', 1.0,
          85.0, 115.0),
94      ('1.3.6.1.4.1.3090.99.12.4.1.6.1.27.1', '
          Monitor 1', 'Azimuth Dipolo 2', 'deg', 0.01,
           262.5, 264.5),
95      ('1.3.6.1.4.1.3090.99.12.4.1.6.1.27.2', '
          Monitor 2', 'Azimuth Dipolo 2', 'deg', 0.01,
           262.5, 264.5),
96      ('1.3.6.1.4.1.3090.99.12.4.1.6.1.28.1', '
          Monitor 1', 'Index FM Dipolo 2', NULL, 0.1,
          15.84, 16.16), --***
97      ('1.3.6.1.4.1.3090.99.12.4.1.6.1.28.2', '
          Monitor 2', 'Index FM Dipolo 2', NULL, 0.1,
          15.84, 16.16), --***
98
99      ('1.3.6.1.4.1.3090.99.12.4.1.6.1.19.1', '
          Monitor 1', 'Depth Modulation 30Hz AM', '%'
          , 0.1, 28.0, 32.0),
100     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.19.2', '
          Monitor 2', 'Depth Modulation 30Hz AM', '%'
          , 0.1, 28.0, 32.0),
101     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.20.1', '
          Monitor 1', 'Depth Modulation 9960Hz AM', '
          %', 0.1, 28.0, 32.0),
102     ('1.3.6.1.4.1.3090.99.12.4.1.6.1.20.2', '
```

```
    Monitor 2', 'Depth Modulation 9960Hz AM', '
    %', 0.1, 28.0, 32.0),
103
104 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.9.1', '
    Monitor 1', 'Carrier Frequency', 'kHz', 0.1,
     114899.0, 114901.0),
105 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.9.2', '
    Monitor 2', 'Carrier Frequency', 'kHz', 0.1,
     114899.0, 114901.0),
106 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.15.1', '
    Monitor 1', 'USB (SBA) Frequency', 'kHz',
    0.1, 114909.0, 114911.0),
107 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.15.2', '
    Monitor 2', 'USB (SBA) Frequency', 'kHz',
    0.1, 114909.0, 114911.0),
108 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.16.1', '
    Monitor 1', 'LSB (SBB) Frequency', 'kHz',
    0.1, 114889.0, 114891.0),
109 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.16.2', '
    Monitor 2', 'LSB (SBB) Frequency', 'kHz',
    0.1, 114889.0, 114891.0),
110 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.14.1', '
    Monitor 1', 'Distortion USB-LSB', '%', 0.1,
     40.0, NULL),
111 ('1.3.6.1.4.1.3090.99.12.4.1.6.1.14.2', '
    Monitor 2', 'Distortion USB-LSB', '%', 0.1,
     40.0, NULL),
112
113 -- CSB/LCRI
```

```
    ================================================================

114 ('1.3.6.1.4.1.3090.99.12.6.1.3.1.2', 'CSB (
      LCRI)', 'Battery Cell Equality', 'V', 0.01,
      -2.00, 2.00),
115 ('1.3.6.1.4.1.3090.99.12.6.1.3.1.3', 'CSB (
      LCRI)', 'Battery Voltage', 'V', 0.01, 52.00,
       56.00),
116 ('1.3.6.1.4.1.3090.99.12.6.1.3.1.4', 'CSB (
      LCRI)', 'Battery Current', 'A', 0.01, -1.00,
       20.00),
117 ('1.3.6.1.4.1.3090.99.12.6.1.3.1.5', 'CSB (
      LCRI)', 'VOR Current', 'A', 0.01, 4.00,
      29.00),
118 ('1.3.6.1.4.1.3090.99.12.6.1.2.1.73', 'CSB (
      LCRI)', 'BCPS Status', NULL, NULL, NULL,
      NULL),
119 ('1.3.6.1.4.1.3090.99.12.6.1.3.1.6', 'CSB (
      LCRI)', 'CSL VDD', 'V', 0.01, 14.00, 16.00),
120 ('1.3.6.1.4.1.3090.99.12.6.1.3.1.7', 'CSB (
      LCRI)', 'CSL VSS', 'V', 0.01, -16.00,
      -14.00),
121 ('1.3.6.1.4.1.3090.99.12.6.1.3.1.8', 'CSB (
      LCRI)', 'CSL Reference', 'V', 0.01, 2.15,
      2.65),
122 ('1.3.6.1.4.1.3090.99.12.6.1.2.1.88', 'CSB (
      LCRI)', 'Battery Status', NULL, NULL, NULL,
      NULL),
123
```

```
124     -- Transmisores
          ================================================

125     ('1.3.6.1.4.1.3090.99.12.5.1.6.1.3.1', '
          Transmisor 1', 'USB RF Phase', 'deg', 0.1,
          156.0, 164.0),
126     ('1.3.6.1.4.1.3090.99.12.5.1.6.1.3.2', '
          Transmisor 2', 'USB RF Phase', 'deg', 0.1,
          156.0, 164.0),
127     ('1.3.6.1.4.1.3090.99.12.5.1.9.1.16.1', '
          Transmisor 1', 'ADC1 BIT Status', NULL, NULL
          , NULL, NULL),
128     ('1.3.6.1.4.1.3090.99.12.5.1.9.1.16.2', '
          Transmisor 2', 'ADC1 BIT Status', NULL, NULL
          , NULL, NULL),
129     ('1.3.6.1.4.1.3090.99.12.5.1.9.1.17.1', '
          Transmisor 1', 'ADC2 BIT Status', NULL, NULL
          , NULL, NULL),
130     ('1.3.6.1.4.1.3090.99.12.5.1.9.1.17.2', '
          Transmisor 2', 'ADC2 BIT Status', NULL, NULL
          , NULL, NULL);
```

Listing 9.7: Configuración inicial y esquema de la base de datos PostgreSQL

9.1.3. Grafana

Grafana se utiliza como la herramienta principal para visualizar de manera clara y efectiva los datos extraídos de los sistemas de monitoreo y transmitidos al contenedor PostgreSQL. Grafana permite crear paneles personalizados donde se despliegan parámetros críticos. Estos paneles no solo ayudan a

supervisar el estado en tiempo real, sino que también permiten analizar tendencias históricas para detectar anomalías o predecir posibles fallos. El Docker de Grafana se configura para interactuar directamente con la base de datos PostgreSQL, estableciendo una fuente de datos constante y confiable. Desde este contenedor, Grafana extrae los valores de los parámetros, los organiza y los presenta de manera visual mediante gráficos, tablas y alertas configurables.

CAPÍTULO 10

Resultados de la extracción de parámetros críticos automatizada

10.1. Tablero de monitoreo

Se llevó a cabo una distribución de paneles y tableros de monitoreo con el objetivo de lograr un equilibrio entre la presentación detallada de la información y la facilidad y rapidez para su interpretación. En este enfoque, se empleó un código de colores que facilita la identificación inmediata del estado de los parámetros monitoreados:

- Verde: Indica que el parámetro se encuentra en condiciones normales.
- Amarillo: Actúa como una pre-alarma, señalando que los valores de un parámetro están próximos a salir de las tolerancias establecidas.
- Rojo: Señaliza que el parámetro ha excedido los límites de tolerancia, requiriendo intervención para su corrección.

Particularmente en las lecturas relacionadas con los transmisores, se integraron colores adicionales para facilitar la in-

terpretación de su estado en tiempo real y en el historial. Estos colores son:

- Gris: Indica que el transmisor asociado a ese parámetro se encuentra apagado en tiempo real.
- Naranja: En el historial de estado de los transmisores, señala que el transmisor estuvo apagado en esa instancia de tiempo.
- Azul: En el historial de estado de los transmisores, indica que el transmisor estuvo activo en esa instancia de tiempo.

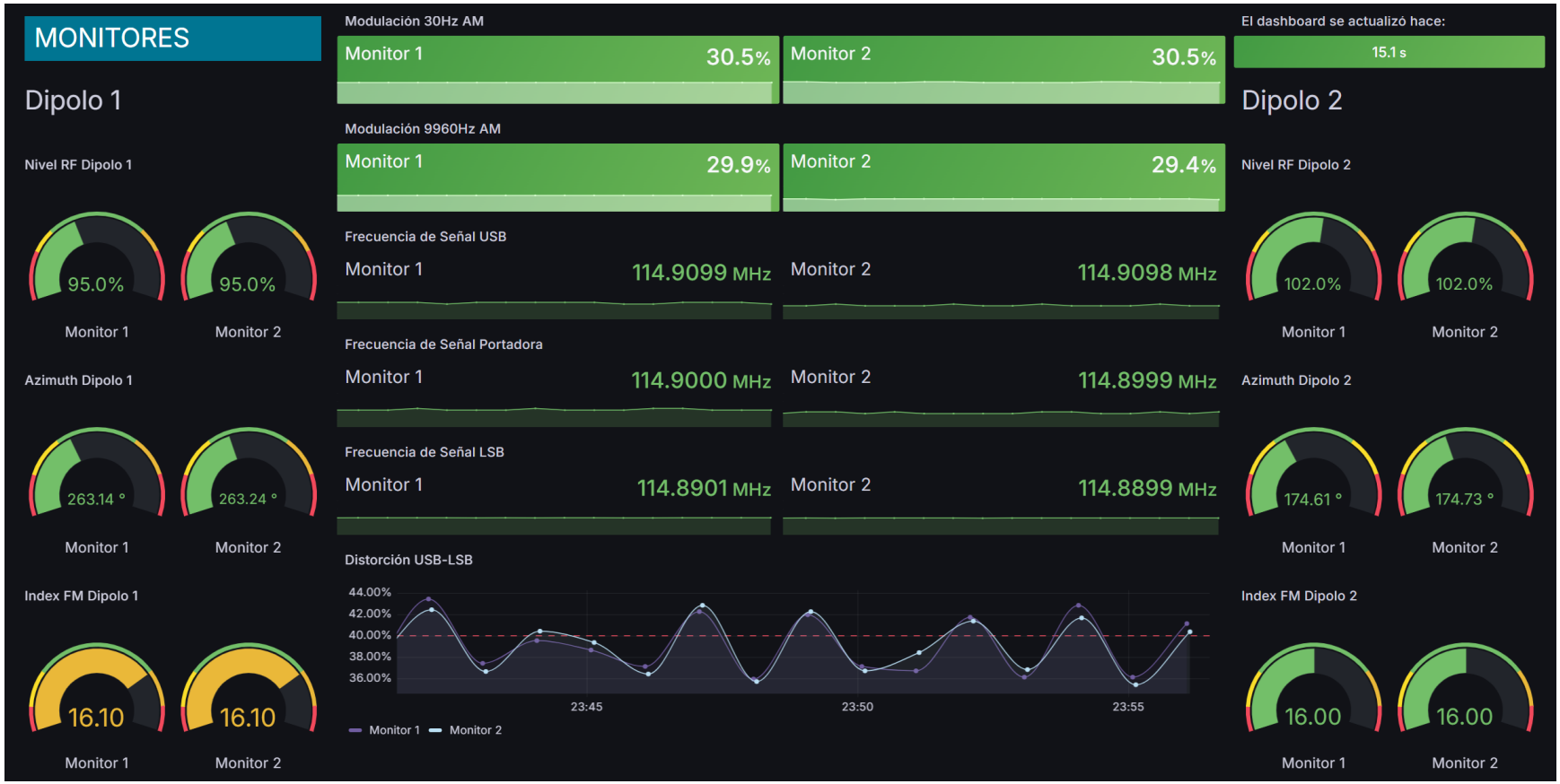


Figura 16: Tablero de lecturas de monitores

Figura 17: Tablero de lecturas de transmisores

Figura 18: Tablero de lecturas de CSB y LRCI

10.2. Alertas

Se implementó un sistema de alertas en Grafana para supervisar en tiempo real los parámetros críticos del sistema DVOR. Este sistema permite detectar y notificar automáticamente cuando los parámetros monitoreados, como las señales de los transmisores o los valores de frecuencia, exceden ciertos umbrales preestablecidos. Las alertas están configuradas para escalar según la persistencia del problema: inicialmente se envían a los técnicos, luego se escalan a los coordinadores si persisten por más de una hora, y finalmente se notifican a los superiores si la alerta dura más de ocho horas. Esta estructura garantiza una intervención rápida y eficiente, mejorando la respuesta ante fallos y optimizando el mantenimiento del sistema.

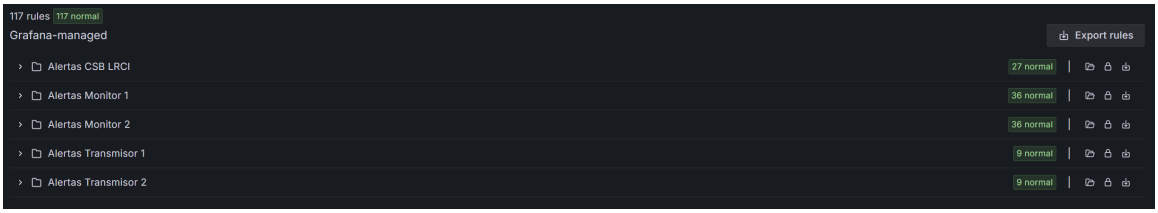


Figura 19: Listado de alertas configuradas en Grafana, estado y agrupación por tipo de parámetro.

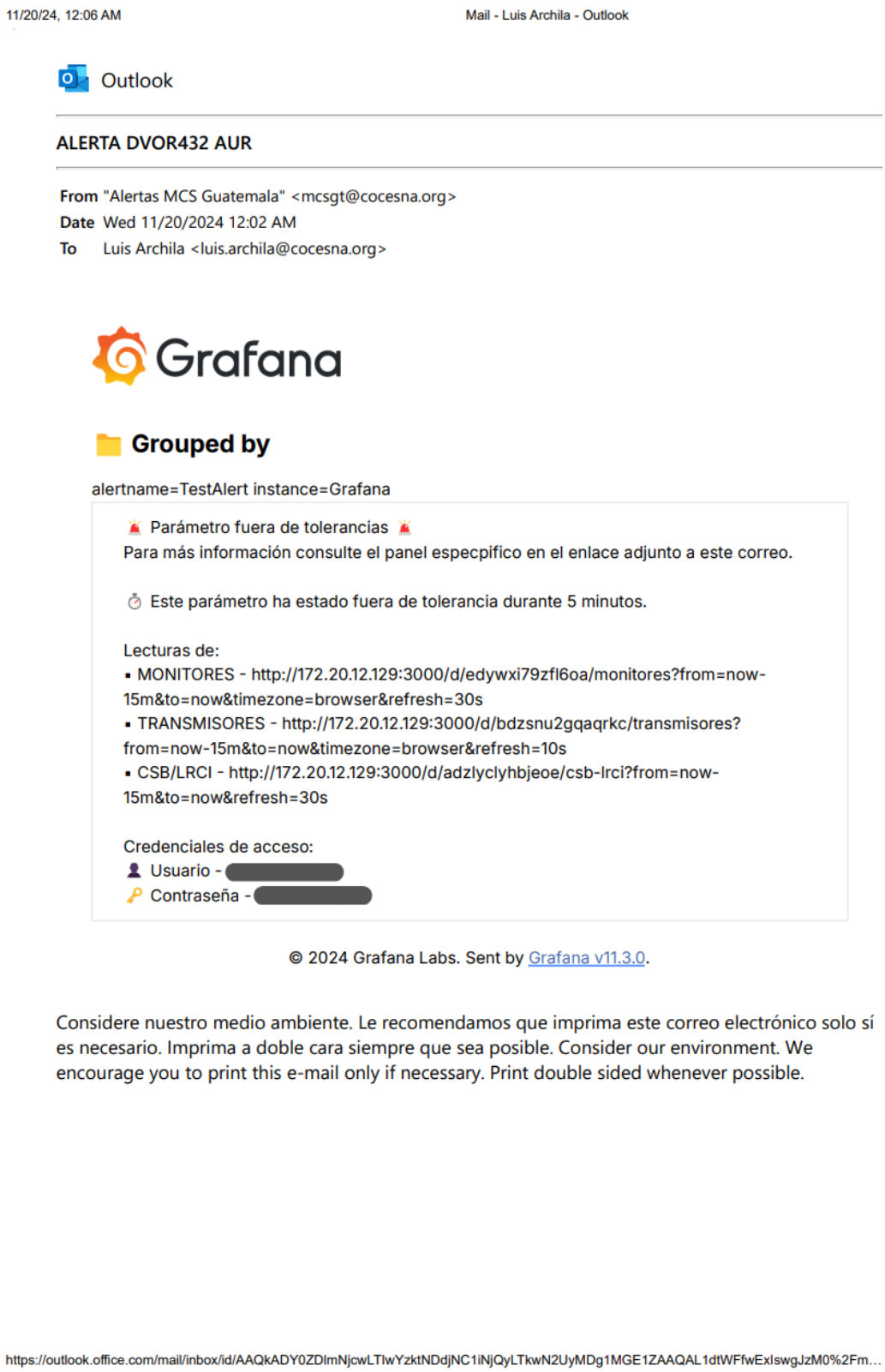


Figura 20: Ejemplo de formato de alertas por correo electrónico, no representa una alerta ni un parámetro real.

10.3. Reportes de monitoreo automatizados para DVOR432 AILA

A continuación, se presenta el resultado de la generación automática de reportes de mantenimiento, el cual corresponde al mismo archivo que los interesados reciben por correo electrónico. Este reporte contiene los últimos registros de los parámetros monitoreados. Dicho informe se genera de manera periódica y automatizada, asegurando que los datos estén siempre actualizados y sean fácilmente accesibles para los técnicos y coordinadores encargados de la gestión del mantenimiento.

CORPORACIÓN CENTROAMERICANA DE SERVICIOS DE NAVEGACIÓN AÉREA

Organismo Internacional de Integración Centroamericana

REPORTE DE MONITOREO DVOR432_2K

Aeropuerto Internacional La Aurora

20-11-2024

Fecha de edición original	Octubre 2024
Fecha de edición vigente	N/A
Versión	Prototipo
Unidad responsable	Gerencia Técnica

ELABORACIÓN

Nombre	Luis José Archila
Cargo	Pasante
Fecha	04/10/2024

REVISIÓN

Nombre	—
Cargo	—
Fecha	—

APROBACIÓN

Nombre	—
Cargo	—
Fecha	—

LECTURAS DE MONITORES

Parámetro	**Monitor 1**	**Monitor 2**	Valor nominal	Lim. inferior	Lim. superior	Estado
Nivel RF Dipolo 1	**95.0%**	**95.0%**	100%	85%	115%	NORMAL
Azimuth Dipolo 1	**263.1°**	**263.14°**	263.5°	262.5°	264.5°	NORMAL
Índice FM Dipolo 1	**16.1**	**16.1**	16.0	15.84	16.16	NORMAL
Nivel RF Dipolo 2	**102.0%**	**102.0%**	100%	85%	115%	NORMAL
Azimuth Dipolo 2	**174.62°**	**174.68°**	175.0°	174.0	176.0	NORMAL
Índice FM Dipolo 2	**16.0**	**16.0**	16.0	15.84	16.16	NORMAL
Modulación 30Hz AM	**30.5%**	**30.6%**	30%	28%	32%	NORMAL
Modulación 9960Hz AM	**29.8%**	**29.5%**	30%	28%	32%	NORMAL
Frecuencia de señal portadora	**114900.0 kHz**	**114899.8 kHz**	114900.0 kHz	114899 kHz	114901 kHz	NORMAL
Frecuencia de USB	**114910.0 kHz**	**114909.9 kHz**	114910.0 kHz	114909 kHz	114911 kHz	NORMAL
Frecuencia de LSB	**114890.1 kHz**	**114889.9kHz**	114890.0 kHz	114889 kHz	114891 kHz	NORMAL
Distorsión USB-LSB	**42.5%**	**42.8%**	N/A	40%	N/A	NORMAL

LECTURAS DE TRANSMISORES

Parámetro	**Transmisor 1**	**Transmisor 2**	Valor nominal	Lim. inferior	Lim. superior	Estado
Fase RF de USB	**161.0**	**Tx OFF**	160.0°	156.0°	164.0°	NORMAL
Estado BITE de ADC 1	**OK**	**Tx OFF**	OK	N/A	N/A	NORMAL
Estado BITE de ADC 2	**OK**	**Tx OFF**	OK	N/A	N/A	NORMAL

LECTURAS CSB/LRCI

Parámetro	**Lectura**	Valor nominal	Lim. inferior	Lim. superior	Estado
Igualdad de celdas de baterías	**0.3**	0.00 V	-2.00 V	2.00 V	NORMAL
Voltaje total de baterías	**53.85**	54.0 V	52.0 V	56.0 V	NORMAL
Corriente de baterías	**0.0**	N/A	-1 A	20 A	NORMAL
Corriente VOR	**13.09**	N/A	4 A	29 A	NORMAL
Estado BCPS	**ON**	ON	N/A	N/A	NORMAL
CSL VDD	**14.6**	15.0 V	14.0 V	16.0 V	NORMAL
CSL VSS	**-14.67**	-15.0 V	-16.0 V	-14.0 V	NORMAL
Referencia CSL	**2.47**	2.40 V	2.15 V	2.65 V	NORMAL
Estado general de la batería	**OK**	OK	N/A	N/A	NORMAL

Los datos mostrados anteriormente corresponden a la última actualización de la Base de Datos (UTC): 2024-11-20 05:59:12.029801+00:00

Enlaces para visualización de paneles de monitoreo

- Lecturas de Monitores
- Lecturas de Transmisores
- Lecturas de CSB LRCI

Se abrirá el entorno gráfico de *Grafana*, las credenciales de inicio de sesión para visualización son:

- Usuario: monitoreogt
- Contraseña: monitoreogt

CAPÍTULO 11

Interpretación de parámetros monitoreados y su importancia para el funcionamiento del equipo

11.1. Lecturas de monitores

Nivel RF dipolos

Mide la intensidad de la señal de radiofrecuencia captada por el dipolo de monitoreo del sistema DVOR. Este parámetro es crucial para evaluar la potencia de la señal transmitida por la antena. Un nivel de RF bajo o inestable puede indicar problemas en la antena o en la transmisión, lo que podría afectar la cobertura y precisión del sistema de navegación.

Azimut dipolos

Indica la dirección en la que se orienta el dipolo en el plano horizontal. Es fundamental para asegurar que la señal del sistema DVOR se dirija correctamente hacia las áreas de cober-

tura deseadas. Una desviación en el azimut puede resultar en una cobertura incorrecta o en la pérdida de señal en áreas críticas.

Índice FM

Está relacionado con el diseño físico del sistema, especialmente con el diámetro del arreglo de antenas. Un valor de 16.0 indica que el arreglo de antenas tiene un tamaño adecuado en relación con la frecuencia de operación, lo que asegura una correcta modulación de la señal de radiofrecuencia.

Profundidad modulación 30Hz AM

Indica la intensidad con la que se modula la señal a esta frecuencia. Un valor adecuado es necesario para asegurar que la señal sea lo suficientemente estable y clara, lo que facilita una recepción precisa por parte de los receptores del sistema de navegación.

Profundidad modulación 9960Hz AM

Se refiere a la modulación de la señal en una frecuencia específica. Este parámetro es crítico para garantizar que la señal transmitida no sufra distorsiones que puedan interferir con su capacidad para ser recibida correctamente.

Frecuencia portadora

Es la frecuencia básica sobre la que se modula la señal de radiofrecuencia. Este parámetro es esencial para asegurar que el sistema opere dentro de los límites de frecuencia estableci-

dos, evitando interferencias con otros sistemas de navegación o comunicación.

Frecuencia USB

Se refiere a la frecuencia de la portadora de banda superior (Upper Side Band). Es importante monitorearla para asegurar que la señal esté en el rango correcto y no se desplace fuera de las bandas asignadas, lo que podría afectar la operación del sistema DVOR.

Frecuencia LSB

Es la frecuencia de la portadora de banda inferior (Lower Side Band). Similar al parámetro USB, su monitoreo es esencial para garantizar que la señal no interfiera con otros sistemas de navegación o comunicaciones.

Distorsión USB/LSB

Mide las alteraciones en las señales de las bandas USB y LSB. Un valor alto de distorsión puede generar problemas de calidad de la señal, lo que afectaría la precisión del sistema de navegación. Mantener estos valores bajos es fundamental para un funcionamiento adecuado.

11.2. Lecturas de transmisores

Fase RF USB

Mide la fase de la señal RF en la banda USB. Este parámetro es esencial para asegurar la correcta alineación y sincroni-

zación de la señal transmitida. Desviaciones en la fase pueden generar interferencias que afecten la precisión del sistema de navegación.

Estado ADC

Se refiere a los bits del estado del convertidor analógico a digital (ADC). Este parámetro es crucial para verificar que la señal analógica esté siendo convertida correctamente a formato digital, asegurando que no haya errores en la transmisión y procesamiento de la señal.

11.3. Lecturas LRCI

Igualdad de celdas de batería

Mide la diferencia de voltaje entre las celdas de la batería. Mantener las celdas equilibradas es fundamental para maximizar la vida útil de la batería y evitar problemas de carga que puedan afectar el rendimiento del sistema de respaldo de energía.

Voltaje total de la batería

Es el voltaje combinado de todas las celdas de la batería. Este parámetro permite identificar problemas de carga o fallas en la batería, lo cual es crucial para garantizar el funcionamiento continuo del sistema DVOR en caso de un corte de energía.

Corriente de la batería

Indica la cantidad de corriente que fluye a través de la batería. Es importante monitorear este parámetro para detectar posibles problemas en el proceso de carga o descarga de la batería, asegurando que el sistema siempre tenga suficiente energía disponible.

Corriente VOR

Mide la corriente que fluye a través del sistema VOR. Este parámetro es esencial para asegurar que el sistema de navegación VOR esté funcionando adecuadamente, proporcionando información precisa y confiable.

Estado BCPS

Indica el estado del sistema de respaldo de energía BCPS. Es crucial monitorearlo para asegurar que el sistema esté listo para entrar en funcionamiento en caso de que se produzca una falla en la fuente principal de alimentación.

Voltaje CSL VDD

Es el voltaje de alimentación del sistema CSL. Este parámetro es necesario para garantizar que el sistema de monitoreo y control esté operando correctamente, sin fluctuaciones que puedan afectar su estabilidad.

Voltaje CSL VSS

Es el voltaje de tierra del sistema CSL. Al igual que el voltaje VDD, es importante para asegurar que el sistema funcione

de manera estable y que no haya problemas de interferencia o mal funcionamiento debido a fluctuaciones de voltaje.

Referencia CSL

Mide la referencia de voltaje utilizada por el sistema CSL. Monitorear este parámetro asegura que el sistema se mantenga calibrado y funcionando dentro de los límites establecidos.

Estado de la batería

Indica la condición general de la batería del sistema. Este parámetro es fundamental para identificar cuando la batería necesita mantenimiento o reemplazo, garantizando que el sistema siempre esté preparado para funcionar sin interrupciones.

CAPÍTULO 12

Conclusiones

- A través de la implementación de soluciones tecnológicas, como la automatización de la recolección y procesamiento de datos mediante protocolos de red, se ha logrado un avance significativo hacia la reducción de la intervención manual. Esto permite optimizar el uso de recursos y facilita un enfoque más estratégico por parte de los equipos de trabajo. Herramientas como PostgreSQL y Grafana han sido claves para almacenar y visualizar los datos de forma eficiente.
- Con el uso de LaTeX y expresiones regulares en Python para la generación de reportes de monitoreo, se ha conseguido automatizar otro aspecto clave del proyecto, mejorando la eficiencia en la creación de reportes detallados para análisis y auditorías internas.
- La recolección de parámetros mediante SNMP y su almacenamiento en bases de datos estructuradas ha permitido obtener una visión en tiempo real del rendimiento del sistema. Esta mejora en la supervisión posibilita una respuesta más rápida ante cualquier falla, contribuyendo directamente a la fiabilidad y seguridad de los sistemas de navegación aérea.
- La implementación de soluciones tecnológicas no solo ha optimizado los procesos de supervisión y mantenimiento, sino que también ha demostrado cómo los recursos hu-

manos y económicos pueden gestionarse de manera más eficiente. Al reducir el tiempo invertido en tareas repetitivas y mejorar el flujo de información, el equipo puede enfocarse en tareas de mayor valor agregado.

- Este proyecto, siendo una iniciativa piloto para el DVOR432, sienta las bases para la creación de un modelo replicable en otros equipos y sistemas de navegación de COCESNA. La estandarización y automatización de rutinas de monitoreo remoto representan una metodología adaptable que podría aplicarse a distintos equipos en la región, potenciando una infraestructura de navegación aérea más segura, moderna y sostenible.

CAPÍTULO 13

Recomendaciones

- Es fundamental que el personal técnico reciba capacitación continua en el uso del software, administración de bases de datos, y configuración de herramientas de monitoreo como Grafana y PostgreSQL. Esto garantizará la operación adecuada del sistema automatizado, así como una resolución de problemas eficiente y una escalabilidad a otros equipos de manera ordenada.
- Realizar un análisis exhaustivo de compatibilidad entre el equipo y los protocolos de monitoreo automatizado. Algunos equipos antiguos podrían requerir adaptaciones para integrarse completamente con el sistema de monitoreo. Se debe iniciar con una fase piloto que permita validar la funcionalidad y detectar problemas específicos en el entorno o equipo.
- Crear una guía técnica completa que incluya instrucciones detalladas de instalación, configuración, operación y solución de problemas. La documentación asegura coherencia en futuras expansiones y facilita el proceso de implementación en otros equipos.
- La colaboración con los fabricantes de equipos es crucial para obtener archivos, información y especificaciones técnicas que faciliten la integración y el monitoreo de nuevos dispositivos.

- La incorporación de herramientas de análisis predictivo permitirá prever posibles fallas y realizar mantenimientos preventivos. Buscando mejorar la fiabilidad del sistema, además de optimizar los costos y recursos a largo plazo.

CAPÍTULO 14

Bibliografía

[1] G. Quiros, *RUTINA DE MANTENIMIENTO DVOR 432, ALCATEL*, 2003.

[2] T. Group, *About Thales*, 2024. dirección: https://www.thalesgroup.com/en/global/group.

[3] COCESNA, *Nuestra Historia*, 2021. dirección: https://cocesna.org/site/historia/.

[4] OACI, *Organización de Aviación Civil Internacional*, 2024. dirección: https://www.icao.int/.

[5] «NAVAIDs: Navigational Aids,» U.S. Federal Aviation Administration, inf. téc., 2019. dirección: https://www.faa.gov/about/office_org/headquarters_offices/ang/offices/tc/library/storyboard/detailedwebpages/navaid.html.

[6] «Air Navigation: Navaids,» U.S. Federal Aviation Administration, inf. téc., 2023. dirección: https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap1_section_1.html.

[7] F. A. A. FAA, *VTAC General Principles*, Presented to: International Flight Inspection Crew Members, 2022.

[8] «IEEE Standard Letter Designations for Radar-Frequency Bands,» *IEEE Std 521-2002 (Revision of IEEE Std 521-1984)*, págs. 1-10, 2003. DOI: 10.1109/IEEESTD.2003.94224.

[9] G. Höfgen, «VOR y VOR-Doppler,» ver. Español, *Comunicaciones Eléctricas*, n.º 50/4, 1975. dirección: https://forohistorico.coit.es/index.php/sendas/tecnologica-mundial/item/vor-y-vor-doppler.

[10] *What is the difference between conventional VOR and Doppler VOR?* Aviation Stack Exchange Forum, 2016. dirección: https://aviation.stackexchange.com/questions/29879/what-is-the-difference-between-conventional-vor-and-doppler-vor.

[11] S. Kattamuri, *MCS SERVER CONFIGURATION GUIDE - GENERAL*.

[12] IBM, *Virtualization*, 2024. dirección: https://www.ibm.com/topics/virtualization.

[13] R. Hat, *¿Qué es la Virtualización?* 2024. dirección: https://www.redhat.com/es/topics/virtualization/what-is-virtualization.

[14] VMware, *What is a Virtual Machine?* 2024. dirección: https://www.vmware.com/topics/virtual-machine.

[15] IBM, *Containers vs. VMs: What's the difference?* 2024. dirección: https://www.ibm.com/think/topics/containers-vs-vms.

[16] IBM, *Docker: A Comprehensive Guide*, 2024. dirección: https://www.ibm.com/topics/docker.

[17] S. Pouladvand, *Containerization VS Virtualization*, https://www.linkedin.com/pulse/containers-vs-virtualization-sajjad-pouladvand/, 2022.

[18] *What is PostgreSQL?* 2024. dirección: https://www.postgresql.org/about/.

[19] K. J. S. Douglas R. Mauro, *Essential SNMP: Help for System and Network Administrators*. O'Reilly Media, 2005.

[20] M. Fedor, M. L. Schoffstall, J. R. Davin y D. J. D. Case, *Simple Network Management Protocol (SNMP)*, RFC 1157, mayo de 1990. DOI: 10.17487/RFC1157. dirección: https://www.rfc-editor.org/info/rfc1157.

[21] *User Datagram Protocol*, RFC 768, ago. de 1980. DOI: 10.17487/RFC0768. dirección: https://www.rfc-editor.org/info/rfc768.

[22] IBM, *SNMPv3*, https://www.ibm.com/docs/es/aix/7.1?topic=management-snmpv3, 2021.

[23] IBM, *API: Qué es y por qué es importante*, https://www.ibm.com/mx-es/topics/api, 2024.

[24] A. W. Services, *¿Qué es una API?* 2024. dirección: https://aws.amazon.com/es/what-is/api/.

CAPÍTULO 15

Anexos

Guía de instalación para THALiX 13.1 y MCS 3.0

CORPORACIÓN CENTROAMERICANA DE SERVICIOS DE NAVEGACIÓN AÉREA

GUÍA DE INSTALACIÓN DE THALiX 13.1 Y MCS 3.0

Fecha de edición original	Agosto 2024
Versión	v1.0
Autor	Luis José Archila Madrid

En esta guía de instalación se asume que el lector cuenta con conocimientos del funcionamiento básico de un sistema Linux, comandos y operación en una terminal.

1 Requisitos de instalación

Para la instalación de sistema operativo serán necesarios todos los archivos adjuntos en la carpeta comprimida de instalación (a la fecha actual en la versión 1.0 de esta guía de instalación, existe la posibilidad de que en versiones posteriores se cuente con más archivos o versiones actualizadas de los mismos).

- thalix-13.1.iso - imagen de instalación de sistema operativo
- 8314501301-301_-A.iso - imagen de instalación de MCS
- Carpeta de mapas de Centroamérica: En los archivos adjuntos se encontrarán los mapas de los países de Centroamérica comprendidos por COCESNA y de los sitios/aeropuertos de Guatemala.
  - Belice.jpg
  - CentroamericaCocesna.jpg
  - CostaRica.jpg
  - ElSalvador.jpg
  - GT-LaAurora.jpg
  - GT-MundoMaya.jpg
  - GT-PuertoBarrios.jpg
  - GT-Rabinal.jpg
  - GT-SanJose.jpg
  - Guatemala.jpg
  - Honduras.jpg
  - Nicaragua.jpg
- COCESNA.xml - Archivo con estructura MCS de centroamérica

En caso de instalación en máquina física:

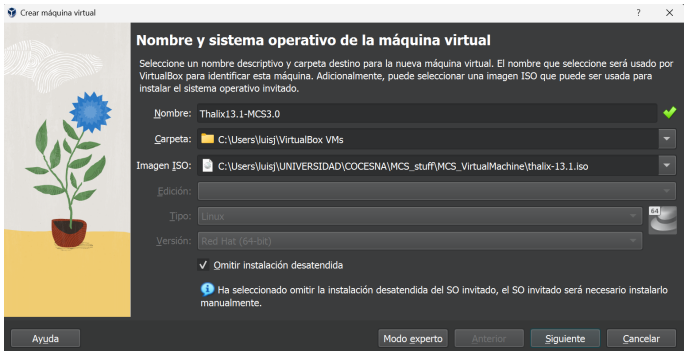
- Procesador de 64 bits
- Memoria RAM de 4GB
- USB booteable con imagen ISO de sistema operativo
- USB con todos los archivos adjuntos en la carpeta comprimida (puede ser la misma USB formateada nuevamente luego de la instalación del sistema operativo)

2 Preparación para instalación de sistema operativo

2.1 En caso de servidor virtual

2.1.1 Creación de la máquina virtual

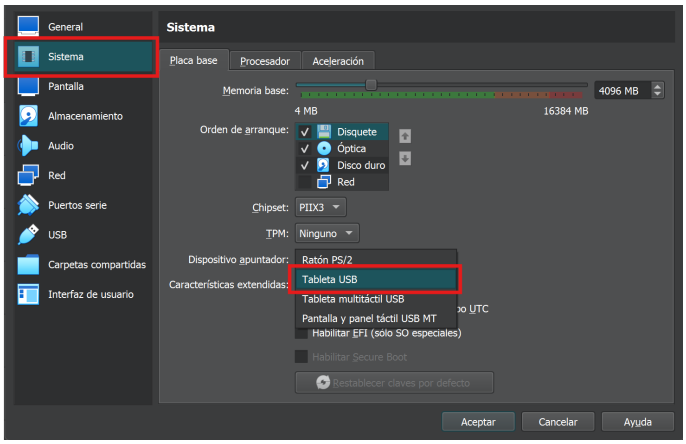
En esta guía se trabajará en el ambiente virtual Oracle VirtualBox.



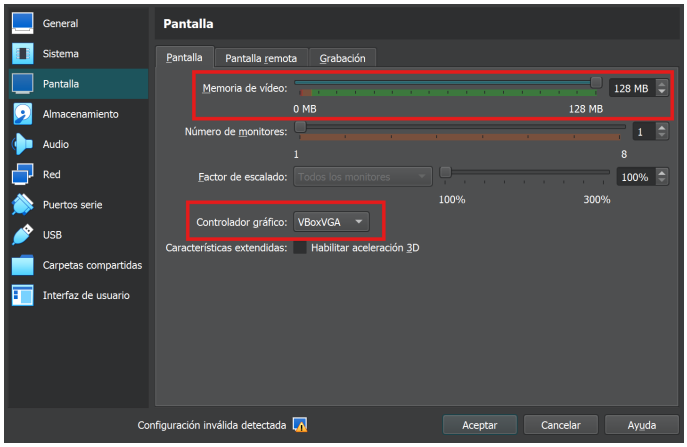
Se recomienda asignar 4GB (4096MB) de memoria RAM y 4 procesadores para un comportamiento óptimo de la máquina virtual, sin embargo, si se cuenta con recursos limitados será posible asignar 2GB de memoria y 1 procesador, apegándose a las limitaciones que esto significa. El tamaño de disco virtual de almacenamiento queda a discreción, en esta instalación se creará un disco virtual de 20GB.

2.1.2 Configuración de máquina virtual

Es probable que este paso no sea necesario en todas las instalaciones ya que depende de la máquina *Host*, sin embargo, para el correcto funcionamiento del cursor durante la ejecución de la máquina virtual. Colocar tipo de dispositivo apuntador "Tableta USB"

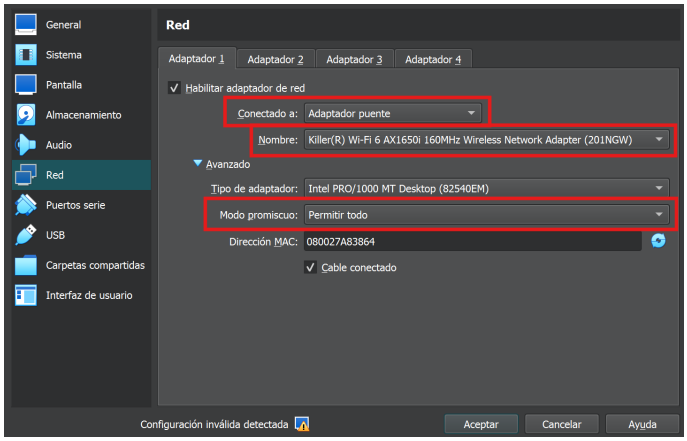


De nuevo, la configuración de video dependerá de las capacidades de la máquina *Host* pero es muy recomendable brindarle la mayor capacidad posible a la memoria de video y experimentar con el tipo de controlador gráfico y seleccionar uno que se adapte a las expectativas y necesidades.



Colocar un adaptador puente en configuración de red y seleccionar el tipo de adaptador según sea el caso (wireless en caso de conectarse a la red por medio de WiFi o Ethernet en caso de

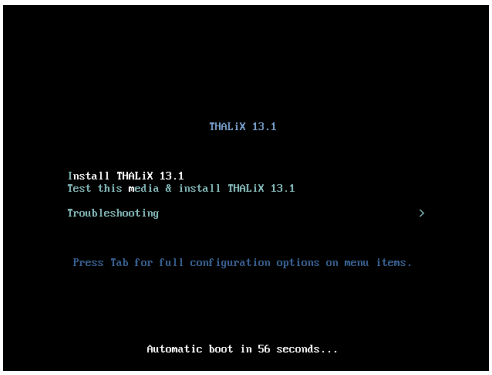
conectarse a la red mediante un cable de red). También permitir todo en modo promiscuo ya que es útil al momento de intercambio de archivos entre la máquina virtual y la máquina *Host*.



2.2 En caso de servidor físico

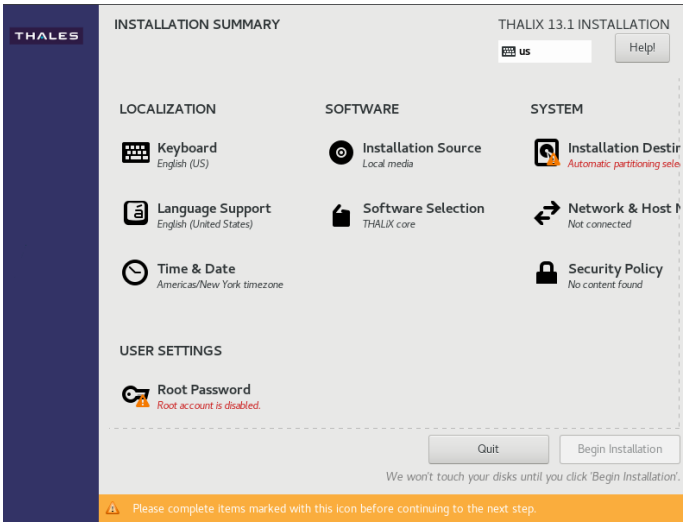
Será necesario crear un dispositivo *"booteable"* con la imagen ISO del sistema operativo thalix-13.1. Para ello, se recomienda el uso del *software* Rufus. Debe colocarse este dispositivo en un puerto USB y encender la máquina, al momento del arranque será necesario entrar a la BIOS y configurar para que el arranque se realice mediante este dispositivo.

3 Instalación de sistema operativo



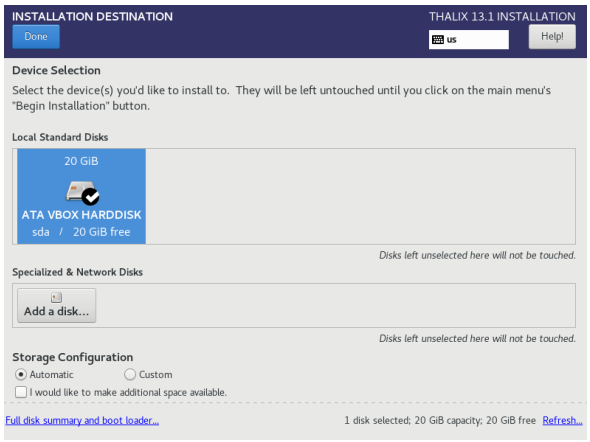
Una vez iniciada la máquina (ya sea virtual o física) aparecerá esta pantalla, se selecciona la primera opción y se inicia el instalador. Posteriormente se selecciona el idioma de preferencia.

Se desplegará el siguiente menú con todas las opciones de configuración, para tener seguridad de que todo se instale bien, será necesario revisar cada una de las secciones. Las opciones de la columna izquiera (Localization) pueden ser configuradas también una vez que se instale el sistema operativo (SO) y son opciones triviales por lo que se saltarán en la guía pero es necesario que se configuren para su correcto funcionamiento.

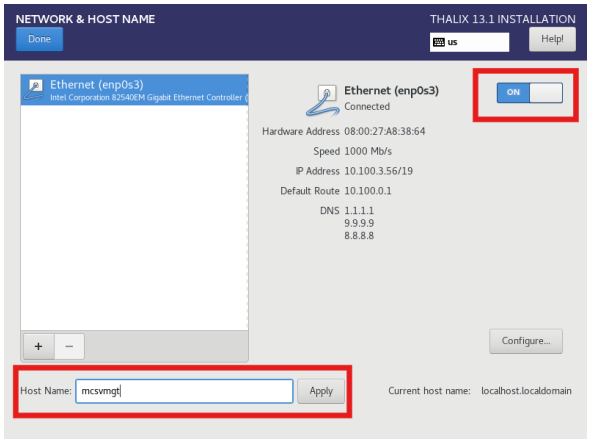


3.1 System: Installation Destination

En esta opción es necesario seleccionar el disco virtual que se creó al momento de la creación de la máquina virtual (o la partición del disco deseada en caso sea una máquina física). Al finalizar seleccionar "Done" y en el menú principal deberían reflejarse los cambios sin errores ni advertencias.

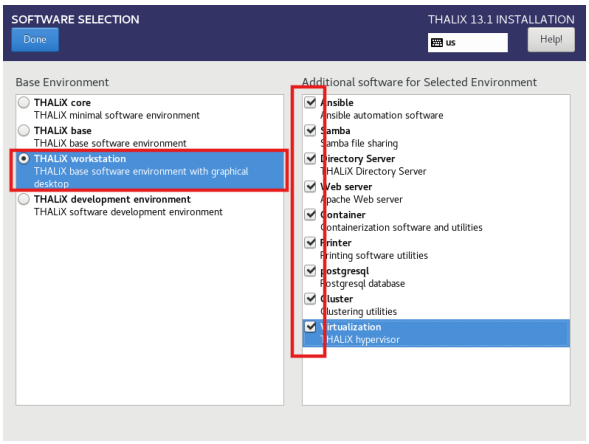


3.2 System: Network & Hostname



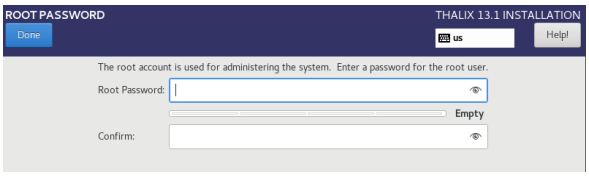
3.3 Software: Software Selection

Seleccionar THALiX Workstation ya que es la opción que brinda una interfaz gráfica y facilidades de uso intuitivas de cualquier sistema operativo convencional. Adicionalmente, se pueden instalar paquetes de software adicional en caso sea necesario, en este caso se instalaron todos.



3.4 User Settings: Root Password

En esta sección se configura la contraseña de usuario root, por convención/costumbre se utilizó "thales" como contraseña. Si se utiliza una contraseña sencilla o "de diccionario" será necesario confirmar dos veces en "Done".



NO se debe crear ningún usuario en la sección "User Creation" ya que estos se crearán al momento de la instalación del programa MCS.

Al finalizar todos estos pasos, seleccionar "Begin Installation" desde el menú principal y esperar a que el sistema operativo se instale y la máquina se reinicie.

4 Configuración básica del sistema operativo: Preparación para la instalación de MCS

Una vez el sistema operativo se haya instalado correctamente, será necesario ingresar con las siguientes credenciales temporales (una vez instalado el MCS esto cambiará):

- Usuario: root
- Contraseña: thales

NOTA: La contraseña dependerá de la que fue configurada como "root password" al momento de la instalación del SO.

4.1 Recomendaciones

- Configurar el "sleep" del sistema para que nunca se cierre, es decir, que ante inactividad el sistema no cierre sesión ni ponga ningún salvapantallas (Settings > Power).
- Luego de la instalación del MCS, instalar VBox Guest Additions (desde la configuración de la máquina virtual en la interfaz de VirtualBox) para lograr habilitar el portapapeles bidireccional y poder copiar/pegar texto de manera bidireccional entre máquina host y máquina virtual.

4.2 Importación de archivos para instalación

En caso de instalación en máquina física, será necesario montar una USB con todos los archivos adjuntos a esta guía de instalación o también por medio de SFTP o algún protocolo de red que permita la transferencia de archivos (la mejor opción para la transferencia en caso se instale en una máquina virtual).

Será necesario pasar los archivos a un directorio conveniente, se recomienda fuertemente apegarse al procedimiento mostrado en los pasos siguientes. Sin embargo, no es un proceso sumamente necesario para su funcionamiento, solamente ordenado y conveniente, siguiendo las buenas prácticas de una instalación en linux. El SO es una distribución de Red Hat (RHEL) por lo que su sistema funciona de manera similar a Fedora, Rocky, CentOS u openSUSE.

Ya que en esta guía de instalación se está trabajando en una máquina virtual, se presentará este procedimiento de transferencia de archivos, sin embargo, en una máquina física con USB es un procedimiento mucho más sencillo, aplicando comandos básicos de Linux.

4.2.1 Creación de directorios

```
mkdir /home/thalixMCS
mkdir /mnt/thalixISO
mkdir /mnt/mcsISO
```

4.2.2 Traslado de archivos

La transferencia de archivos se realizará con la ayuda del software WinSCP, se ingresa la IP de la máquina destino seguida de las credenciales del usuario root. Transferir los archivos a

`/home/thalixMCS`

```
[root@mcsvmgt ~]# mkdir /home/thalixMCS
[root@mcsvmgt ~]# mkdir /mnt/mcsISO
[root@mcsvmgt ~]# mkdir /mnt/thalixISO
[root@mcsvmgt ~]# cd /home/thalixMCS/
[root@mcsvmgt thalixMCS]# ll
total 3812720
-rw-r--r-- 1 root root  308645888 Jun 21 13:54 8314501301-301_-A.iso
-rw-r--r-- 1 root root       4557 Jun 25 14:07 COCESNA.xml
drwxr-xr-x 2 root root       4096 Aug 30 17:30 Mapas
-rw-r--r-- 1 root root 3595567104 Dec 19  2023 thalix-13.1.iso
[root@mcsvmgt thalixMCS]#
```

4.3 Montaje de imágenes ISO

```
mount -o loop /home/thalixMCS/thalix-13.1.iso /mnt/thalixISO
mount -o loop /home/thalixMCS/8314501301-301_-A.iso /mnt/mcsISO
```

Ya que la imagen ISO del SO thalix contiene los archivos de repositorios de yum/dnf, será algo demasiado útil realizar una configuración para que esta imagen se monte automáticamente en cada inicio de la máquina. Para ello, será necesario modificar el archivo /etc/fstab utilizando la herramienta "nano".

```
nano /etc/fstab
```

Agregar la línea:

```
/home/thalixMCS/thalix-13.1.iso /mnt/thalixISO iso9660 loop,defaults 0 0
```

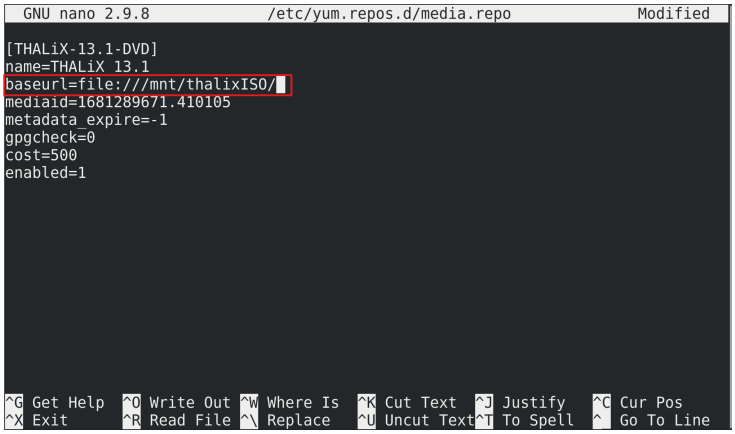
Una vez montadas ambas imágenes, acceder al directorio donde está montada la imagen de Thalix. Deberían observarse estos archivos/directorios:

```
[root@mcsvmgt ~]# cd /mnt/thalixISO/
[root@mcsvmgt thalixISO]# ll
total 429
dr-xr-xr-x 3 root   root     2048 Apr 12  2023 EFI
-rw-r--r-- 1 root   root    17989 Oct 19  2021 GPL
dr-xr-xr-x 3 root   root     2048 Apr 12  2023 images
drwxr-xr-x 2 nobody nobody   2048 Apr 12  2023 isolinux
-rw-r--r-- 1 nobody nobody    126 Apr 12  2023 media.repo
drwxrwxrwx 2   3509    122 405504 Apr 12  2023 Packages
drwxr-xr-x 2 nobody nobody   4096 Apr 12  2023 repodata
-rw-r--r-- 1   3509    122   3137 Apr  7  2023 RPM-GPG-KEY-THALIX
-r--r--r-- 1 root   root      888 Apr 12  2023 TRANS.TBL
[root@mcsvmgt thalixISO]#
```

Copiar el archivo **"media.repo"** al directorio **/etc/yum.repos.d**:

```
cp /mnt/thalixISO /etc/yum.repos.d
```

En adición a eso, será necesario modificar el archivo copiado para que yum/dnf tome como repositorio la dirección donde se tiene montada la imagen ISO. Utilizando la herramienta nano, modificar la línea mostrada en la figura de abajo, colocar el directorio antes mencionado tal y como se muestra.



Guardar el archivo y probar que haya funcionado con el siguiente comando:

```
dnf update
```

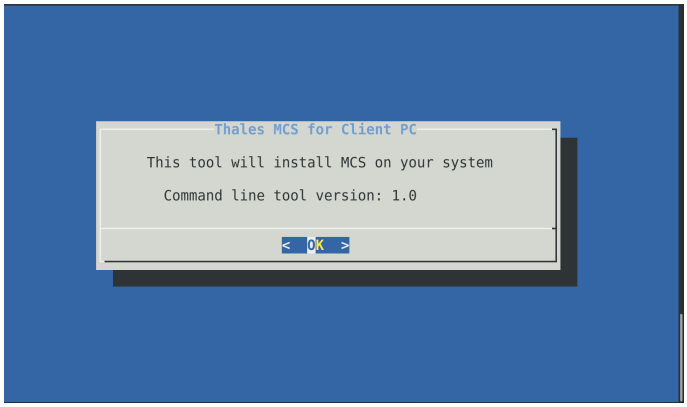
Debería actualizarse el gestor de paquetes yum/dnf sin errores.

5 Instalación MCS

Ir al directorio donde se encuentra montada la imagen ISO del software MCS y ejecutar el archivo **setup.sh**, todo puede lograrse con el siguiente comando:

```
/mnt/mcsISO/thalix/setup.sh
```

Se abrirá el siguiente wizard de instalación, se deberá presionar 'Enter' para pasar cada sección una vez se ejecute la instalación correspondiente a cada página, es un procedimiento intuitivo. Existirá una sección en donde se pregunta si se desea activar SSL para Postgres; se trata de una capa de seguridad extra para la base de datos, una encriptación de los datos, en términos simples. Queda a discreción del usuario si se activa o no.



Una vez terminado el wizard, será necesario reiniciar la máquina. Al iniciar nuevamente se encontrarán dos usuarios nuevos creados, ingresar como usuario **MCS Administrator** con la contraseña **mcsadmin**. Ya que ninguno de estos usuarios es privilegiado, será necesario ingresar como root a una terminal para ejecutar comandos de nivel privilegiado, para ello, el comando "sudo" no será de ayuda a menos de que se configure el usuario dentro de la lista de super-users. Para realizar acciones como root, será necesario ingresar el comando "su", se pedirá una contraseña tras darle a 'Enter', esta será la configurada como *root password* ("thales", en este caso).

5.1 Configuración inicial y solución de errores de ejecución

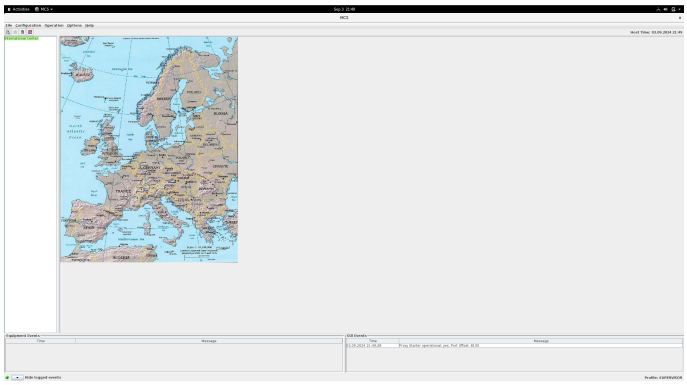
Ya que esta versión de MCS presenta errores al momento de la creación y acceso a los agentes proxy SNMP, será necesario copiar algunos archivos de librerías a un directorio general del sistema operativo, para ello, ejecutar los siguientes comandos como root:

```
rsync -av /usr/local/lib64 /lib64/
cp /usr/local/lib/libmisc.so /lib64/
cp /usr/local/lib/libACE.so.6.3.0 /lib64/
mount -o loop /home/thalixMCS/8314501301-301_-A.iso /mnt/mcsISO
dnf install /mnt/mcsISO/thalix/System/* -y
```

Una forma de verificar que todo lo anterior funcionó y se hizo de manera correcta es intentar ejecutar cualquier archivo Proxy encontrado en `/usr/local/bin/` y debería aparecer el siguiente cuadro de texto sin errores:

```
[root@mcsvmgt bin]# ./REU_Proxy -h
./REU_Proxy: /usr/pgsql-14/lib/libpq.so.5: no version information available (required by /lib64/libpqxx-4.0.so)
-----------------------------------------------------------------------
Usage:
./REU_Proxy<options>
possible options:
-h              print this short help and exit
-v              print information on the version of the proxy agent and exit
-d              run the proxy agent as daemon
-f <filename>   use the given file as configuration for the proxy
-D              start proxy in directConnect mode
-----------------------------------------------------------------------
```

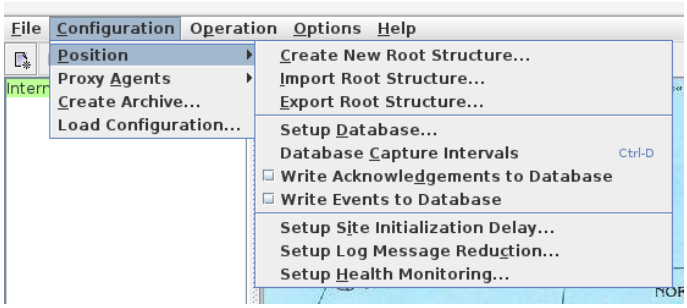
5.2 Configuración de interfaz MCS



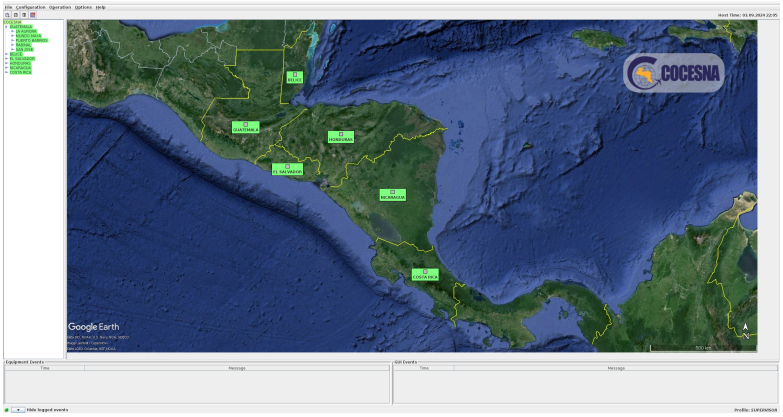
Al iniciar por primera vez el software MCS, se tendrá una pantalla así y sin ningún equipo configurado. Dentro de los archivos adjuntos se encuentra un archivo llamado `COCESNA.xml`, este contiene una estrucutra diseñada para COCESNA. Para ello, primero será necesario copiar los archivos de imágenes/mapas al directorio `/usr/local/MCS/images/`:

```
cp /home/thaliXMCS/Mapas/* /usr/local/MCS/images
```

Una vez se hayan copiado todas las imágenes, será necesario importar la estructura raíz. Para ello, seleccionar `Configuration > Position > Import root structure...`



Seleccionar el archivo `/home/thalixMCS/COCESNA.xml`



Una vez hecho este procedimiento (y cualquiera dentro de MCS), será necesario guardar los cambios con el comando `ctrl+s` o desde el botón arriba a la izquierda del GUI.

CAPÍTULO 16

Glosario

Advantek: Empresa especializada en soluciones de protección y empaque de componentes electrónicos, como cintas carrier y coberturas de película, utilizadas principalmente en las industrias de semiconductores y electrónica para proteger componentes sensibles durante el transporte y ensamblaje.

Aeronaves: Vehículos capaces de navegar en el aire, como aviones, helicópteros y drones, utilizados para transporte, vigilancia, y otras aplicaciones en la aviación civil y militar.

AES: Algoritmo de cifrado simétrico avanzado de 128, 192 o 256 bits, ampliamente adoptado como estándar de cifrado seguro para proteger información confidencial en diversas aplicaciones.

API (Interfaz de Programación de Aplicaciones): Conjunto de herramientas y protocolos que permiten que diferentes sistemas de software interactúen entre sí.

COCESNA: Corporación Centroamericana de Servicios de Navegación Aérea, encargada de proveer servicios de navegación en la región.

Código Morse: Sistema de codificación que representa letras y números mediante combinaciones de puntos y rayas, uti-

lizado en telegrafía y comunicaciones de emergencia para transmitir mensajes de forma sencilla y efectiva.

DES: Algoritmo de cifrado simétrico de 56 bits utilizado para la protección de datos. Aunque fue un estándar en criptografía, ha sido en gran medida reemplazado debido a su vulnerabilidad a ataques de fuerza bruta.

DME (Distance Measuring Equipment): Dispositivo que mide la distancia entre una aeronave y una estación de tierra mediante el uso de señales de radiofrecuencia.

DVOR: Doppler VHF Omnidirectional Range por sus siglas en inglés. Sistema de navegación aérea que utiliza el Efecto Doppler para proporcionar información precisa sobre la posición de una aeronave.

Efecto Doppler: Fenómeno físico en el que la frecuencia de una onda varía en función del movimiento relativo entre la fuente de la onda y el receptor.

Grafana: Herramienta de visualización de datos utilizada para monitorizar sistemas y generar gráficos en tiempo real.

ILS *(Instrument Landing System)*: Sistema de aterrizaje instrumental que guía a las aeronaves en su aproximación final a la pista mediante señales de radio, proporcionando orientación precisa tanto en el eje horizontal como en el vertical para aterrizajes seguros en condiciones de baja visibilidad.

Kernel Linux: Núcleo del sistema operativo Linux que gestiona recursos del hardware, controla procesos, y facilita la comunicación entre el software y el hardware, siendo fundamental en sistemas operativos basados en Linux.

MCS (*Monitoring and Control System*): Sistema utilizado para supervisar y controlar equipos de navegación aérea a través de una interfaz gráfica.

MD5: Algoritmo de hash criptográfico que genera un resumen de 128 bits para verificar la integridad de datos. Aunque

ampliamente utilizado en el pasado, es considerado inseguro para aplicaciones críticas debido a vulnerabilidades de colisión.

OACI (Organización de Aviación Civil Internacional): Agencia de las Naciones Unidas que establece normas y recomendaciones para la aviación civil internacional, promoviendo la seguridad, eficiencia y sostenibilidad en el transporte aéreo a nivel mundial.

OID *(Object Identifier)*: Identificador único utilizado en la gestión de objetos dentro de una MIB en redes SNMP.

PostgreSQL: Sistema de gestión de bases de datos relacional de código abierto que permite almacenar y consultar datos estructurados.

Radial: Dirección magnética que una aeronave sigue desde o hacia una estación VOR en un curso específico.

Radioayudas: Dispositivos en tierra que proporciona información esencial para la navegación aérea, facilitando a las aeronaves el seguimiento de rutas y la aproximación a pistas de aterrizaje, especialmente en condiciones de baja visibilidad. Las radioayudas, como el VOR (VHF Omnidirectional Range) y el ILS *(Instrument Landing System)*, actúan como sistemas de respaldo para la navegación satelital y son fundamentales para la seguridad y eficiencia en la gestión del tráfico aéreo.

Routers: Dispositivos de red que dirigen el tráfico de datos entre diferentes redes, permitiendo la comunicación eficiente y segura entre dispositivos conectados a internet o a redes privadas.

Servidor Titan: Plataforma de alto rendimiento utilizada para ejecutar contenedores Docker y manejar aplicaciones intensivas en procesamiento y almacenamiento, diseñada para soportar ambientes virtualizados y cargas de trabajo complejas en proyectos de desarrollo e investigación.

Servidores: Computadoras o sistemas dedicados que proporcionan recursos, servicios o datos a otros dispositivos en

una red, como almacenamiento de archivos, hosting de aplicaciones y manejo de bases de datos.

SHA: Familia de algoritmos de hash seguros diseñados para generar resúmenes únicos de datos, comúnmente utilizados en la verificación de integridad y en la criptografía moderna. Los más utilizados incluyen SHA-1, SHA-256 y SHA-3.

SNMP (Protocolo Simple de Administración de Red): Protocolo que facilita la administración y supervisión remota de dispositivos en redes.

Switches: Dispositivos de red que conectan múltiples dispositivos dentro de una misma red, facilitando el intercambio de datos y mejorando la gestión del tráfico local en redes de área local (LAN).

TCP/IP (Transmission Control Protocol/Internet Protocol): Conjunto de protocolos que facilita la comunicación en redes mediante la transmisión de datos de forma fiable.

Thalix OS: Sistema operativo basado en Linux utilizado para ejecutar el MCS de Thales Group.

Traps: Mensajes de notificación en el protocolo SNMP que se envían automáticamente desde un dispositivo de red al sistema de gestión cuando ocurre un evento específico o una anomalía, permitiendo una supervisión en tiempo real.

UDP (User Datagram Protocol): Protocolo de red que facilita la transmisión rápida de datos sin control de flujo ni garantías de entrega.

VHF (Very High Frequency): Rango de frecuencias de radio que oscila entre 30 MHz y 300 MHz, utilizado principalmente para comunicaciones aéreas.

VOR (VHF Omnidirectional Range): Sistema de radioayuda terrestre que permite a las aeronaves determinar su rumbo y ubicación.

XML: Lenguaje de marcado extensible utilizado para estructurar y almacenar datos de forma legible tanto para humanos como para máquinas, comúnmente empleado en el intercambio de información entre sistemas en aplicaciones web y servicios.