



DIVISIÓN DE ESTUDIOS POSGRADO E INVESTIGACIÓN

Opción 1.- TESIS

TRABAJO PROFESIONAL

**SISTEMA ELECTRÓNICO DE MONITOREO DE VARIABLES DE
VOLTAJE, CORRIENTE Y TEMPERATURA PARA LA PREDICCIÓN
DE FALLAS Y OPTIMIZACIÓN DE MANTENIMIENTO A
CONGELADORES INDUSTRIALES**

**QUE PARA OBTENER EL GRADO DE:
MAESTRO EN INGENIERÍA ELECTRÓNICA**

PRESENTA:

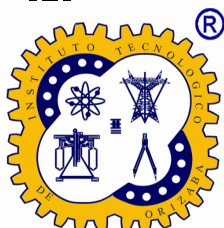
ING. RENÉ GARCÍA ROBLES

DIRECTOR DE TESIS:

DRA. BLANCA ESTELA GONZÁLEZ SÁNCHEZ

CODIRECTOR DE TESIS:

DR. OSCAR OSVALDO SANDOVAL GONZALEZ





Instituto Tecnológico de Orizaba
Departamento de División de Estudios de Posgrado e Investigación

Orizaba, Veracruz, **2/octubre/2026**
Dependencia: **División de Estudios de
Posgrado e Investigación**
Asunto: **Autorización de Impresión**
OPCION: I

C. RENÉ GARCÍA ROBLES

Candidato(a) a Grado de Maestro(a) en:
INGENIERÍA ELECTRÓNICA
P R E S E N T E.-

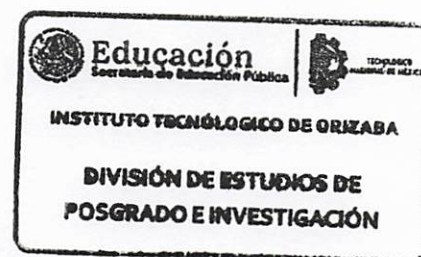
De acuerdo con el Reglamento de Titulación vigente de los Centros e Institutos Tecnológicos Federales del Tecnológico Nacional de México, de la Secretaría de Educación Pública, y habiendo cumplido con todas las indicaciones que la Comisión Revisora le hizo respecto a su Trabajo Profesional titulado:

**“SISTEMA ELECTRÓNICO DE MONITOREO DE VARIABLES DE VOLTAJE, CORRIENTE
Y TEMPERATURA PARA LA PREDICCIÓN DE FALLAS Y OPTIMIZACIÓN DE
MANTENIMIENTO A CONGELADORES INDUSTRIALES”**

Comunico a Usted que este Departamento concede su autorización para que proceda a la impresión del mismo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
CIENCIA – TÉCNICA - CULTURA®

DR. OSCAR BÁEZ SENTÍES
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN



OG-13-F06



2026
año de
Margarita
Maza





Instituto Tecnológico de Orizaba
Departamento de División de Estudios de Posgrado e Investigación

Orizaba, Veracruz, **30/septiembre/2026**
Asunto: **Revisión de trabajo escrito**

C. DR. OSCAR BÁEZ SENTÍES
JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
P R E S E N T E.-

Los que suscriben, miembros del jurado, han realizado la revisión de la Tesis del (la) C.

RENÉ GARCÍA ROBLES

La cual lleva el título de:

**“SISTEMA ELECTRÓNICO DE MONITOREO DE VARIABLES DE VOLTAJE, CORRIENTE Y
TEMPERATURA PARA LA PREDICCIÓN DE FALLAS Y OPTIMIZACIÓN DE MANTENIMIENTO A
CONGELADORES INDUSTRIALES”**

Y concluyen que se acepta.

A T E N T A M E N T E

Excelencia en Educación Tecnológica®
CIENCIA – TÉCNICA - CULTURA®

PRESIDENTE: DRA. BLANCA ESTELA GONZÁLEZ
SÁNCHEZ

FIRMA

SECRETARIO: DR. OSCAR OSVALDO SANDOVAL
GONZÁLEZ

FIRMA

VOCAL: DR. GERARDO ÁGUILA RODRÍGUEZ

FIRMA

VOCAL SUP.: DR. IGNACIO HERRERA AGUILAR

FIRMA

TA-09-18



2026
año de
**Margarita
Maza**



CONTENIDO

RESUMEN	9
ABSTRACT	10
INTRODUCCIÓN	11
PLANTEAMIENTO DEL PROBLEMA	14
JUSTIFICACIÓN	15
HIPÓTESIS	16
OBJETIVOS	17
OBJETIVOS ESPECÍFICOS	17
CONTRIBUCIÓN AL CONOCIMIENTO	18
CAPÍTULO 1: ESTADO DEL ARTE	19
CAPÍTULO 2: MARCO TEÓRICO	25
2.1 Sistemas de refrigeración	25
2.1.1 Funciones principales	25
2.2 ¿Qué es IoT?	26
2.2.1 IoT Industrial (IIoT)	26
2.2.2 Arquitectura de IoT	27
2.2.3 Beneficios del IoT	27
2.2.4 Computación en la nube	27
2.2.5 Plataformas en la nube para IoT	28
2.3 Sistemas embebidos y microcontroladores	29
2.3.1 Arquitectura de un sistema embebido	29

2.4	Microcontroladores.....	29
2.4.1	Características del microcontrolador ESP32.....	30
CAPÍTULO 3: METODOLOGÍA		31
3.1	Desarrollo.....	38
3.1.1	Placa de circuito impreso	38
3.1.2	Diseño de la caja para Protección y Montaje	45
3.1.3	Firmware del ESP32	46
3.1.3.1	Variables calculadas	47
3.1.4	Lógica de detección de fallas	48
3.1.5	Almacenamiento local en microSD	50
3.1.6	Selección de sensores	50
3.1.7	ZMPT101B	51
3.1.8	ACS712 30A	55
3.1.9	DS18B20.....	57
3.1.10	SD APDAPTER ESP32	60
3.1.10.1	Código de prueba con BLYNK	61
3.2	Especificaciones del congelador monitoreado	64
3.2.1	Instalación de sensores dentro del congelador	65
3.2.2	Conexión con AWS.....	66
3.2.3	Arquitectura de servicios AWS	71
3.2.3.1	Enrutamiento de mensajes	72
3.2.3.2	DynamoDB	74

3.2.3.3	AWS Lambda	76
3.2.3.4	Función procesar_alertas_congelador	78
3.2.4	IAM.....	81
3.2.5	SNS	81
3.2.6	EventBridge	83
3.2.7	CloudWatch	84
3.2.8	Grafana	84
3.2.9	Creación de dashboard	84
3.2.9.1	Configuración de panel	85
3.2.10	Data source en la query	85
3.2.11	Diseño final de panel	86
CAPÍTULO 4: RESULTADOS		87
4.1	PCB Ensamblada	87
4.1.1	Sistema de montaje durante pruebas	88
4.1.2	PCB dentro de Case	88
4.1.3	Diseño final del case	89
4.1.4	Unidades fabricadas	90
4.2	Estructura de pruebas por Software	90
4.2.1	Verificación de la transmisión de datos	91
4.3	Resultados del sistema eléctrico	92
4.3.1	Análisis de ciclos del compresor.....	93
4.3.2	Resultados de temperatura.....	93

4.3.3	Resultados del COP	94
4.3.4	Verificación del sistema de Alarmas	95
4.4	Grafana	97
4.4.1	Panel 1	97
4.4.2	Panel 2	97
4.4.3	Panel 3	98
4.4.4	Panel 4	99
CONCLUSIONES		100
RECOMENDACIONES		103
REFERENCIAS BIBLIOGRÁFICAS		105

LISTA DE FIGURAS

Figura 1.	Ciclo de refrigeración por compresión de vapor	25
Figura 2.	Elementos fundamentales de un sistema de refrigeración.	26
Figura 3.	Representación gráfica de la metodología	31
Figura 4.	Etapas de la metodología y su correspondencia con los objetivos específicos (OE). Las flechas punteadas indican los ciclos de retroalimentación del proceso.	33
Figura 5.	Esquema del circuito	39
Figura 6.	Prototipo uno	40
Figura 7.	Prototipo 2	41
Figura 8.	Prototipo 3	41
Figura 9.	Layout cara inferior.....	42
Figura 10.	Layout cara superior	43
Figura 11.	Modelo 3D del sistema embebido.....	43
Figura 12.	Vista superior PCB	44
Figura 13.	Vista inferior.....	44
Figura 14.	instalando componentes en PCB	44
Figura 15.	Sistema bajo pruebas	44
Figura 16.	Case vista lateral.....	45
Figura 17.	Case vista lateral izquierda.....	46
Figura 18.	Case final	46
Figura 19.	Configuración interna	51
Figura 20.	Gráfica del sensor.....	52
Figura 21.	Gráfica no calibrada.....	53
Figura 22.	Simulación Lámparas en paralelo.....	55
Figura 23.	Modelo de corriente.....	56
Figura 24.	Sensor de temperatura conectado al ESP32	58
Figura 25.	Dashboard en Blynk	61
Figura 26.	Perforación a la cámara	65
Figura 27.	Sensores adentro de cámara.....	65
Figura 28.	Termostato vs sensor.....	66
Figura 29.	Sensor dentro de cámara.....	66
Figura 30.	Consola Principal de AWS IoT Core Región US East (Ohio)	67

Figura 31.	Registro de dispositivo congelador_ito en AWS IoT Core	67
Figura 32.	Asignación de la política al certificado del dispositivo	68
Figura 33.	Descarga del certificado de dispositivo, llave privada y Amazon Root CA ...	69
Figura 34.	Verificación de la conexión MQTT mediante el cliente de prueba de AWS IoT Core.....	70
Figura 35.	Cliente MQTT con mensaje JSON real.....	71
Figura 36.	Acción de la regla IoT.....	72
Figura 37.	Propiedades completas de la regla de enrutamiento hacia DynamoDB	73
Figura 38.	Mapeo de claves de partición y ordenación desde el mensaje MQTT.....	73
Figura 39.	Tabla monitoreo_congeladores activa en Amazon DynamoDB.....	74
Figura 40.	Estructura de la tabla con clave de partición dispositivos y clave de ordenación timestamp	75
Figura 41.	Registros almacenados en la tabla monitoreo_congeladores con datos reales del congelador_ito.....	75
Figura 42.	Funciones Lambda desplegadas en AWS, región US East (Ohio).....	76
Figura 43.	Función dynamo_a_cloudwatch con desencadenador EventBridge.....	76
Figura 44.	Código de la función dynamo_a_cloudwatch.....	77
Figura 45.	Testeo a la función dynamo_a_cloudwatch.....	77
Figura 46.	Permisos para EventBridge	78
Figura 47.	Función procesar_alertas_congelador con desencadenador AWS IoT Core ..	78
Figura 48.	Código de la función.....	79
Figura 49.	Configuración de evento de prueba	79
Figura 50.	Permiso de invocación otorgado a AWS IoT Core.....	80
Figura 51.	Configuración detallada del permiso de invocación de IoT Core.....	80
Figura 52.	Roles dentro de IAM	81
Figura 53.	Tema SNS alertas_congeladores creado en la región US East (Ohio).....	81
Figura 54.	Suscripción de correo electrónico confirmada al tema alertas_congeladores.	82
Figura 55.	Ejemplo de envío de alerta vía EMAIL.....	82
Figura 56.	Funciones dentro de EventBridge	83
Figura 57.	Regla para Eventbridge a función.....	83
Figura 58.	Agregar panel en Grafana	84
Figura 59.	Cambio de nombre al Panel en Grafana	85

Figura 60.	Configuración de Query	86
Figura 61.	Panel de potencia	86
Figura 62.	Ensamble de PCB con sensores y microcontrolador	87
Figura 63.	Sistema embebido colocado junto a congelador CHP-68	88
Figura 64.	Case con sistema embebido integrado	89
Figura 65.	Diseño Placa enclosure.....	89
Figura 66.	Unidades armadas	90
Figura 67.	Monitor Serie	91
Figura 68.	Correo de alerta generado automáticamente ante una condición de ciclo largo en el compresor.....	95
Figura 69.	Alerta sensor fallando	96
Figura 70.	Panel 1 Corriente y Voltaje.....	97
Figura 71.	Temperatura de salida y COP	98
Figura 72.	Voltaje	98
Figura 73.	Panel de Potencia	99

LISTA DE TABLAS

Tabla 1.	Etapas de la metodología, productos y criterios de cumplimiento.....	34
Tabla 2.	Variables Calculadas	48
Tabla 3.	Diagnóstico de fallas por fases	49
Tabla 4.	Valores de calibración	53
Tabla 5.	Voltaje comparada con el multímetro y el sensor	54
Tabla 6.	Tabla de conexiones modulo SD Adapter	60
Tabla 7.	Especificaciones CHP-68	64
Tabla 8.	Resultados del análisis de corriente de arranque	91
Tabla 9.	Estadísticas del sistema eléctrico	92
Tabla 10.	Análisis de ciclos del compresor	93
Tabla 11.	Estadísticas de temperatura de la cámara	94
Tabla 12.	Análisis del coeficiente de rendimiento (COP).....	94

RESUMEN

En los congeladores comerciales e industriales, el mantenimiento suele ser correctivo: se repara el equipo cuando ya falló. Si la falla no se detecta a tiempo, el producto almacenado puede perderse. En este trabajo se diseñó, fabricó y validó un sistema electrónico de bajo costo que monitorea de forma continua el voltaje, la corriente, la temperatura y la vibración de equipos de refrigeración por compresión de vapor. Su propósito es detectar a tiempo condiciones anómalas de operación y sentar las bases para un mantenimiento predictivo. El sistema se montó en una placa de circuito impreso propia, desarrollada en cuatro iteraciones. La placa integra un microcontrolador ESP32, un sensor de voltaje ZMPT101B, un sensor de corriente de efecto Hall ACS712, tres sensores de temperatura DS18B20, una unidad inercial MPU6050 y almacenamiento local en microSD. Los sensores eléctricos se calibraron contra un multímetro de referencia. El firmware toma lecturas cada 2 s y caracteriza la corriente de arranque del compresor a 1 kHz. Con esos datos aplica reglas de diagnóstico sobre la duración de los ciclos, la corriente y la temperatura. La información se envía a AWS IoT Core mediante MQTT sobre TLS 1.2, con autenticación por certificados X.509. Después se almacena en DynamoDB y se visualiza en Grafana, y las alertas llegan por correo electrónico a través de AWS Lambda y SNS. Para validar el sistema, se instaló en un congelador de 113 L con refrigerante R-134a durante 61.3 h de operación continua. En ese periodo se registraron 110,422 muestras y 101 ciclos del compresor. Con el análisis de los ciclos se ajustaron los umbrales de alerta hasta eliminar las falsas alarmas, y la cadena de notificación se probó con eventos reales. Además, se detectó un sobrevoltaje sostenido en el suministro: 134.8 V en promedio y picos de 152.4 V, por encima de la tolerancia de $\pm 10\%$ sobre los 127 V nominales.

Palabras clave: monitoreo de condición, mantenimiento predictivo, refrigeración comercial, Internet de las Cosas, sistemas embebidos.

ABSTRACT

Commercial and industrial freezers are usually maintained correctively: the equipment is repaired only after it has failed. If a failure goes unnoticed, the stored product can be lost. This work presents the design, fabrication, and validation of a low-cost electronic system that continuously monitors voltage, current, temperature, and vibration in vapor-compression refrigeration equipment. The goal is to detect abnormal operating conditions early and to lay the groundwork for predictive maintenance. The system was built on a custom printed circuit board, developed over four iterations. The board integrates an ESP32 microcontroller, a ZMPT101B voltage sensor, an ACS712 Hall-effect current sensor, three DS18B20 temperature sensors, an MPU6050 inertial measurement unit, and local microSD storage. The electrical sensors were calibrated against a reference multimeter. The firmware samples the variables every 2 s and characterizes the compressor's inrush current at 1 kHz. With this data, it applies diagnostic rules based on cycle duration, current, and temperature. Measurements are sent to AWS IoT Core over MQTT with TLS 1.2 and X.509 certificate authentication. They are then stored in DynamoDB and displayed in Grafana, and alerts are delivered by email through AWS Lambda and SNS. To validate the system, it was installed on a 113 L freezer using R-134a refrigerant for 61.3 h of continuous operation. During this period, 110,422 samples and 101 compressor cycles were recorded. Analyzing the cycles made it possible to adjust the alert thresholds until false alarms were eliminated, and the notification chain was tested with real events. In addition, a sustained overvoltage was detected in the power supply: an average of 134.8 V with peaks of 152.4 V, exceeding the $\pm 10\%$ tolerance on the 127 V nominal value.

Keywords: condition monitoring, predictive maintenance, commercial refrigeration, Internet of Things, embedded systems.

INTRODUCCIÓN

La industria alimentaria, la farmacéutica y buena parte de los servicios dependen de la refrigeración, aunque casi nunca se piense en ella. La cadena de frío funciona solo si cada eslabón mantiene el producto dentro de un intervalo de temperatura acotado. Basta con que falle un equipo para comprometer la calidad, la inocuidad o el valor comercial de todo lo que guarda. En los establecimientos pequeños y medianos esta dependencia es todavía más delicada. Ahí los congeladores comerciales trabajan sin parar durante años, suele haber uno solo o muy pocos y no existe respaldo si alguno falla. Además, casi nadie revisa cómo están operando más allá de mirar el termostato de vez en cuando.

Por eso, el mantenimiento de estos equipos suele ser correctivo: se atiende el equipo cuando la falla ya apareció. Este enfoque sale caro por dos lados. Primero está la reparación, que cuesta más cuando la falla avanzó hasta dañar componentes como el compresor. Después está el producto que se pierde mientras el equipo funcionó fuera de especificación sin que nadie se diera cuenta, y que muchas veces vale más que la propia reparación. El mantenimiento preventivo, con intervenciones programadas en calendario, reduce parte de este riesgo. El problema es que no toma en cuenta el estado real del equipo, así que la intervención puede llegar antes de tiempo o demasiado tarde.

El mantenimiento predictivo ofrece otra opción: intervenir cuando la condición del equipo lo pide, con base en la medición continua de variables que reflejan su estado. Todo parte del monitoreo de condición, que consiste en adquirir sistemáticamente variables físicas y compararlas con un comportamiento de referencia para detectar desviaciones. En los sistemas de refrigeración por compresión de vapor, buena parte de los modos de falla importantes se concentra en el compresor. Muchos de ellos dejan huellas en variables fáciles de medir: la corriente que consume, la duración de sus ciclos de trabajo, la temperatura de descarga y la evolución térmica de la cámara. Una fuga lenta de refrigerante, un protector térmico que se dispara una y otra vez o una pérdida de capacidad de enfriamiento modifican estas señales mucho antes de que el producto almacenado resulte afectado. Hoy existen microcontroladores

baratos con conectividad inalámbrica integrada, sensores accesibles y plataformas de cómputo en la nube. Gracias a eso, este tipo de monitoreo ya puede llevarse más allá de la gran industria. Con el Internet de las Cosas, un dispositivo instalado junto al equipo puede enviar sus mediciones de forma continua, que luego se almacenan y consultan a distancia. El mismo sistema puede generar avisos automáticos cuando detecta algo anómalo. En la literatura hay muchas arquitecturas de este tipo aplicadas a maquinaria industrial, equipos hospitalarios, sistemas de climatización y flotas vehiculares. Mucho menos se ha trabajado con congeladores comerciales de baja capacidad validados en equipos reales en operación y con las condiciones de suministro eléctrico propias del país.

Ese último punto importa. La calidad del suministro eléctrico afecta directamente el desempeño y la vida útil de los compresores herméticos. Cuando el voltaje se aleja de su valor nominal, puede aumentar la corriente de operación, la temperatura del devanado y el esfuerzo mecánico durante el arranque. Si un sistema registra al mismo tiempo las variables eléctricas y térmicas del equipo, puede detectar fallas del ciclo de refrigeración. También puede revelar problemas en el suministro que deterioran el equipo y que, sin esa medición, nadie notaría.

Con este planteamiento, en este trabajo se desarrolla un sistema electrónico que monitorea de forma continua el voltaje, la corriente, y la temperatura de congeladores comerciales. El sistema funciona sobre una placa de circuito impreso diseñada para esta aplicación, con un microcontrolador ESP32 y los sensores necesarios para medir las variables eléctricas y térmicas del equipo. El firmware aplica localmente reglas de diagnóstico que identifican ciclos de trabajo prolongados, una posible activación del protector térmico, indicios de pérdida de refrigerante y fallas de sensores. Las mediciones se envían de forma segura a una arquitectura de servicios en la nube que las almacena y las muestra en un tablero de visualización. Cuando detecta una condición de alerta, avisa por correo electrónico al responsable del equipo. Además, el sistema guarda una copia local de todos los registros para no perder datos si se cae la conexión.

Para validarlo, el sistema se instaló en un congelador comercial en operación continua. Con los datos recogidos se caracterizó el comportamiento normal del equipo, se ajustaron los umbrales de diagnóstico y se comprobó que toda la cadena de notificación respondiera ante eventos reales. Esos mismos datos sirven, además, como base para desarrollar más adelante modelos de predicción de fallas.

El documento está organizado así. El Capítulo 1 revisa el estado del arte de los sistemas de monitoreo basados en Internet de las Cosas y su uso en el mantenimiento predictivo. El Capítulo 2 presenta el marco teórico: los fundamentos de la refrigeración por compresión de vapor, el Internet de las Cosas, las plataformas de cómputo en la nube y los sistemas embebidos. El Capítulo 3 explica la metodología de desarrollo, que abarca el diseño de la placa de circuito impreso y del gabinete, la selección y calibración de los sensores, el firmware con su lógica de diagnóstico y la arquitectura en la nube. El Capítulo 4 muestra los resultados de la validación en el equipo de prueba. Al final se presentan las conclusiones y las recomendaciones para continuar el trabajo.

PLANTEAMIENTO DEL PROBLEMA

Los congeladores industriales desempeñan un papel esencial en la preservación de la calidad de los productos; sin embargo, no están exentos de fallas. En la actualidad, no existen sistemas que permitan anticipar posibles problemas, por lo que el mantenimiento que se realiza es de tipo reactivo, es decir, se lleva a cabo únicamente después de que ocurre una avería. Esta situación provoca pérdidas en los productos almacenados y un incremento en los costos de mantenimiento.

La ausencia de un sistema que pueda monitorear variables críticas como el voltaje, la corriente y la temperatura dificulta la implementación de estrategias de mantenimiento predictivo. En consecuencia, cuando se presenta una falla, el producto almacenado puede verse afectado, generando pérdidas económicas y retrasos operativos.

Con el propósito de atender esta problemática, el presente proyecto propone el desarrollo de un sistema electrónico capaz de monitorear en tiempo real las condiciones de operación del congelador y predecir posibles fallas. La implementación de dicho sistema permitirá optimizar la gestión del mantenimiento, reduciendo costos, tiempos de inactividad y riesgos asociados a la pérdida de producto.

JUSTIFICACIÓN

La implementación de un sistema de monitoreo IoT en tiempo real en congeladores es fundamental para abordar varios desafíos críticos en el mantenimiento y operación de equipos de refrigeración. Los congeladores, al ser dispositivos esenciales para la conservación de productos perecederos, requieren un monitoreo constante para garantizar su funcionamiento óptimo. Las fallas en estos sistemas, como variaciones anómalas en el voltaje, corriente, temperatura o vibración, pueden generar pérdidas económicas significativas debido a la destrucción de productos y a la necesidad de reparaciones costosas.

La medición en tiempo real de estas variables permite identificar desviaciones en el comportamiento normal de los equipos y, mediante algoritmos de predicción, anticipar posibles fallas antes de que se materialicen. De esta forma, se pueden realizar intervenciones preventivas de manera más precisa, evitando daños mayores y reduciendo la necesidad de mantenimiento correctivo, que suele ser más costoso. Además, el uso de tecnologías IoT para este monitoreo permite una vigilancia continua, sin la necesidad de intervención humana constante, y proporciona acceso remoto a datos clave desde cualquier lugar, lo que mejora la capacidad de respuesta ante problemas inminentes. Esto no solo incrementa la fiabilidad del sistema de refrigeración, sino que también reduce tiempos de inactividad no planificados, los cuales pueden ser perjudiciales para la operación continua, especialmente en industrias como la alimentaria o farmacéutica, donde la cadena de frío es crítica.

Por otro lado, la prolongación de la vida útil de los congeladores, al prevenir fallas graves, contribuye a una reducción de costos a largo plazo, ya que se maximiza el rendimiento de los equipos sin necesidad de reemplazos frecuentes. Este enfoque proactivo en el mantenimiento y la operación de los congeladores representa una inversión estratégica que optimiza los recursos, mejora la eficiencia operativa y reduce riesgos asociados a las fallas imprevistas, beneficiando tanto a las empresas como a los consumidores al asegurar la calidad y seguridad de los productos almacenados.

HIPÓTESIS

La implementación de un sistema de monitoreo IoT en tiempo real para la medición simultánea de voltaje, corriente y temperatura en congeladores comerciales, permite identificar de manera oportuna desviaciones anómalas en el patrón de operación, optimizando el mantenimiento preventivo y reduciendo el riesgo de fallas catastróficas y pérdidas de producto.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar e implementar un sistema embebido de bajo costo para el monitoreo continuo de variables eléctricas (voltaje y corriente) y térmicas en congeladores comerciales, que permita caracterizar su comportamiento operativo, detectar de forma oportuna condiciones anómalas mediante reglas de diagnóstico en tiempo real y notificar alertas remotas para la prevención de fallas.

OBJETIVOS ESPECÍFICOS

- Diseñar y fabricar una placa de circuito impreso que integre el microcontrolador, los sensores y la etapa de alimentación desde la red eléctrica, con protección contra transitorios de voltaje, para su instalación junto al equipo de refrigeración.
- Calibrar los sensores de voltaje, corriente y temperatura contra instrumentos de referencia, cuantificando el error de medición de cada uno.
- Desarrollar el firmware del sistema para la adquisición periódica de las variables, la caracterización de la corriente de arranque del compresor, la evaluación de reglas de diagnóstico de fallas y el almacenamiento local de los registros.
- Implementar una arquitectura de servicios en la nube que permita la transmisión segura de los datos, su almacenamiento, su visualización en un tablero de monitoreo y el envío de notificaciones automáticas ante condiciones de alerta.
- Validar el sistema en un congelador comercial en operación continua, caracterizando su comportamiento operativo, ajustando los umbrales de diagnóstico a partir de datos reales y verificando la detección y notificación de eventos anómalos.

CONTRIBUCIÓN AL CONOCIMIENTO

El proyecto contribuye al conocimiento científico en el ámbito del mantenimiento preventivo de equipos industriales, desarrollando un sistema embebido basado en el Internet de las Cosas (IoT), sensores y una placa de circuito impreso (PCB) personalizada. Mediante una programación avanzada, el sistema transmite en tiempo real los datos recolectados a la plataforma Amazon Web Services (AWS), lo que posibilita el monitoreo continuo de las condiciones de los congeladores y la anticipación de necesidades de mantenimiento. Dado que actualmente no existe un sistema automatizado que prediga el momento óptimo para realizar el mantenimiento en estos equipos, esta investigación presenta una metodología innovadora y escalable que mejora la eficiencia operativa y optimiza el mantenimiento preventivo en la industria alimentaria.

CAPÍTULO 1: ESTADO DEL ARTE

El siguiente capítulo contiene la información más actualizada sobre el tema, tomando en cuenta todos los avances tecnológicos para la implementación de sistemas de monitoreo en tiempo real

El artículo propone un sistema de mantenimiento predictivo basado en sensores IoT. Siguiendo un algoritmo básico que es conectar sensores, tarjetas electrónicas y dispositivos IoT a una máquina para poder monitorear su comportamiento y detectar señales de fallo antes de que algo se detenga o se dañe. La metodología incluye la recolección de datos en tiempo real, la clasificación automática de fallos y lo que le da utilidad práctica la posibilidad de adaptar el sistema a diferentes tipos de maquinaria. De acuerdo al artículo este sistema no está atado a ningún tipo de maquinaria particular por lo que se puede aplicar dependiendo de las necesidades.

[1] El proyecto habla sobre el monitoreo de equipos hospitalarios a usando IoT y algoritmos automatizados. El objetivo es poder prevenir fallas usando tecnología y dispositivos IoT, con el fin de garantizar la integridad, exactitud, fiabilidad y funcionamiento de los dispositivos hospitalarios. El sistema recolecta datos y los envía en tiempo real a un servidor en la nube, donde un algoritmo entra en juego y logra anticipar fallos. Para que después estos datos se categorizan en cuatro estados: funcionando, apagado, en reparación o en necesidad de ser reemplazado. Los resultados indican que se ha mejorado el monitoreo de los dispositivos en un 94.25 %. [2] El artículo analiza como de IA logra mejorar el mantenimiento predictivo en las redes industriales de IoT. La idea principal es en el cambio de enfoque reactivo a uno proactivo. En lugar de que falle o hacer un mantenimiento previamente programado, el sistema actúa antes, basando únicamente en los sensores lo que estén leyendo en ese momento. Para eso se usan datos en tiempo real de sensores de vibración, temperatura, corriente y presión. Para que finalmente se le apliquen algoritmos de IA que en este caso fueron algoritmos de machine learning y deep learning. como resultado existieron menos tiempos muertos no programados y menos intervenciones innecesarias. [3]

La investigación busca explicar como es que funcionan correctamente los sensores cuando se combinan con IA y de igual manera explica como no hacerlo bien, para el mantenimiento

preventivo, apoyándose en casos reales y estudios empíricos. Para eso utiliza una Revisión Sistemática de la Literatura (SLR), que básicamente consiste en buscar y filtrar de forma ordenada todo lo que ya se ha investigado sobre el tema. El proceso empieza con definir preguntas de investigación concretas y criterios claros para decidir que estudios entran y cuales no sirven para este caso en específico, evitando así que la selección sea injusta. [4] El artículo presenta un sistema de control de temperatura para microcontroladores de bajo costo. El diseño tiene dos fases: primero, un controlador difuso opera el sistema y registra sus propias decisiones; luego, una red neuronal se entrena con esos datos para aprender la dinámica térmica del sistema. Después de eso, la red asume el control y el controlador difuso ya no es necesario el sistema corre offline. Los resultados muestran mejor precisión y menor tiempo de establecimiento que un PID clásico. El sistema también maneja perturbaciones sin perder estabilidad. Y dado que todo corre en hardware y software accesible, el costo de implementación se mantiene bajo. Lo que llama la atención del enfoque es la secuencia: usar el controlador difuso como generador de datos de entrenamiento, no como solución final. Es una forma práctica de arrancar sin datos históricos del proceso. [5] El sistema monitorea corriente y voltaje de forma continua. Si detecta un cortocircuito o una sobrecarga, el Arduino reacciona de inmediato: corta la carga a través de relés sin intervención del usuario. Una pantalla LCD muestra los valores actuales para que el operador pueda ver qué está pasando en todo momento. La lógica es sencilla pero efectiva el sistema no alerta para que alguien actúe, sino que actúa solo y notifica después y así se puede tener un sistema de monitoreo confiable. [6]

El estudio presenta un medidor de consumo eléctrico basado en el microcontrolador ESP32. Utilizando sensores no invasivos lo que simplifica la instalación y sobre todo mantiene un costo bajo, el sistema es capaz de leer los datos de voltaje y corriente en tiempo real y transmite estos datos para su análisis o visualización. El artículo muestra que el rango de aplicaciones es bastante grande ya sea desde una simple facturación residencial hasta una integración en redes eléctricas inteligentes. El autor resalta que esto lo hace interesante no sólo como prototipo académico sino también como punto de partida para un producto real. [7] Zikria 2021 realizó un recorrido por el estado del IoT y hacia dónde apunta su crecimiento. No es solo que más dispositivos se conecten a internet, es que esa masa crítica cambia cómo se conciben los productos y servicios desde el diseño. Los autores documentan aplicaciones en

salud, agricultura, monitoreo industrial y análisis de datos. Lo más útil del trabajo, sin embargo, es que no se queda en listar usos: identifica los cuellos de botella reales que frenan la adopción a gran escala, como la nula operación entre plataformas, la seguridad de los nodos y la gestión del ancho de banda en redes densas. [8] El proyecto desarrolló un sistema IoT para monitorear y reducir el consumo eléctrico residencial. El hardware central es un ESP32 NodeMCU, que recibe las lecturas del sensor SCT013-30 a través del convertidor ADS1115 esta combinación permite medir corriente y potencia sin intervenir directamente en el cableado. Los datos llegan a ThingSpeak y el usuario los consulta en tiempo real desde cualquier dispositivo. El control remoto de cargas está disponible, aunque en condiciones normales el sistema no lo necesitan los relés actúan solos según el consumo y el horario programado. El enfoque apunta a dos objetivos concretos: bajar la factura eléctrica y reducir el impacto ambiental del hogar. La automatización por horario es lo que hace viable el segundo objetivo en el uso cotidiano. [9] El estudio describen NGSPantOne, un sistema IIoT diseñado para monitorear maquinaria en entornos industriales. El hardware son 33 nodos de bajo costo alimentados por batería, instalados en una planta eléctrica para medir temperatura y vibraciones de forma continua. Lo más distintivo del diseño es dónde corre la lógica: directamente en los dispositivos, no en un servidor central. La arquitectura IPv6 resuelve el acceso remoto sin capas intermedias. En las pruebas, los 33 nodos respondieron con latencias compatibles con mantenimiento predictivo ninguno quedó fuera de alcance. Como conclusión demuestra que un sistema de monitoreo industrial no requiere hardware costoso ni conexión centralizada para funcionar a escala.[10] El artículo trabaja sobre un problema concreto: monitorear el consumo energético de una instalación de bombeo subterráneo. Para eso, sensores distribuidos envían datos por MQTT a una base de series temporales. La pasarela Simatic IoT2040 de Siemens hace de puente entre el equipo físico y el sistema de análisis. El trabajo incluye el desarrollo de algoritmos para el cálculo de indicadores de eficiencia. Lo relevante es que no se quedan en simulación: los resultados se compararon contra mediciones en una instalación real, lo que da peso a la propuesta como herramienta de control y mantenimiento predictivo. [11] Raparathi 2023 abordó un problema frecuente en entornos IoT: predecir cuándo un dispositivo va a necesitar mantenimiento antes de que falle. Para eso, combinan análisis de series temporales con modelos de deep learning entrenados con datos reales lecturas de sensores, registros de errores

e historial de intervenciones anteriores. Los modelos identifican patrones en esos datos y generan alertas anticipadas. Según los autores, la precisión mejoró respecto a enfoques previos, lo que se traduce en menos tiempo de inactividad y mejor uso de los recursos de mantenimiento. [12] El artículo hace una revisión sistemática de la literatura sobre mantenimiento predictivo con IoT y machine learning. El foco no está en un sistema propio, sino en mapear qué modelos se usan, cómo se entrenan y qué tan bien funcionan en la práctica. Los tres algoritmos más documentados en la revisión son ANN, SVM y Random Forest. Los tres se nutren de datos capturados por sensores IoT lecturas en condiciones normales y en falla para aprender a distinguir cuándo una máquina está a punto de fallar. Según los autores, los resultados reportados en la literatura muestran alta precisión predictiva, aunque el rendimiento varía según el tipo de maquinaria y la calidad de los datos disponibles. [13] En 2020, Killeen presentan COSMO, un sistema de mantenimiento predictivo diseñado para flotas de autobuses urbanos. El nombre responde a su lógica interna: modelos de machine learning que se auto-organizan por consenso para predecir fallas antes de que ocurran. La fuente de datos es el bus J1939, el protocolo estándar de comunicación entre dispositivos en vehículos comerciales. Desde ahí, el sistema recopila lecturas de los sensores del autobús y las procesa para detectar patrones anómalos. Una parte del trabajo se dedica específicamente a mejorar la calidad de esas lecturas, ya que un sensor que mide mal entrena mal. [14] En el libro parten de un dato concreto: una parte de los accidentes de tráfico en España tiene origen en fallas mecánicas o de sensores. Desde ahí, diseñan una arquitectura IoT para detectar esas fallas antes de que se vuelvan peligrosas. El sistema no se probó en vehículos reales, trabaja con datos simulados generados por una API, transmitidos mediante MQTT y AMQP. A partir de esos flujos, analiza patrones de eventos para estimar el estado de salud del vehículo. La implementación está en Python con FastAPI. Los autores reconocen que es una simulación, pero argumentan que los resultados justifican una prueba en condiciones reales. [15] La investigación revisan el estado del machine learning aplicado al mantenimiento predictivo en sistemas HVAC. El argumento de fondo es práctico: estos sistemas fallan de formas predecibles, y los algoritmos entrenados con datos históricos pueden anticiparlas mejor que los esquemas de mantenimiento por calendario. La revisión clasifica los algoritmos más usados, analiza estudios de caso reales y discute cómo llevar estos modelos a la práctica. Lo que distingue el trabajo es

que no se queda en el inventario técnico intenta responder cuándo y por qué un algoritmo funciona mejor que otro en contextos HVAC específicos. [16] Calabrese (2020) construyó una arquitectura IoT-ML para una empresa maderera. El sistema lee el estado de cada máquina y, si algo sale de rango, actúa sin esperar intervención humana. Los modelos, DRF, GBM y XGBoost, todos entrenados en H2O.AI, alcanzaron precisiones de 98.9 %, 99.6 % y 99.1 %. Los números son altos, pero lo más relevante es el contexto: se validaron sobre maquinaria real en operación, no en datasets de laboratorio. Los autores argumentan que la arquitectura es transferible a otros tipos de equipo industrial, aunque eso queda pendiente de validación. [17] El artículo construye un sistema de mantenimiento predictivo de extremo a extremo: sensores de temperatura, vibración y consumo energético transmiten datos por MQTT a ThingSpeak, donde los modelos procesan la información y deciden si algo está fuera de lo normal. El sistema evalúa varios algoritmos Random Forest, SVM, Gaussian Process y Deep Learning para ver cuál predice mejor el estado de las máquinas. El mecanismo de alerta es directo: si una variable supera el umbral calculado por desviación estándar, el equipo de mantenimiento recibe un correo. Sin dashboards complejos, sin intermediarios. De los modelos probados, Deep Learning fue el que mejor funcionó, con 96 % de precisión. Los autores atribuyen esto a su capacidad para capturar relaciones no lineales en los datos de los sensores. [18] el artículo desarrolla un sistema de mantenimiento predictivo para máquinas BCF de hilo continuo, con hardware de bajo costo. Cada máquina lleva tres sensores: el MPU6050 para vibración en tres ejes, el SCT013-100 para consumo de corriente y el MLX90640 con IR-Pt100 para temperatura. Todo llega a ThingSpeak vía MQTT cada 20 segundos. El preprocesamiento fue cuidadoso: se eliminaron valores ausentes, se normalizaron los datos y se descartaron registros corruptos. Del total recolectado, solo el 15 % se reservó para validación, cuidando que ambas clases "saludable" y "fallo" estuvieran bien representadas. Se entrenaron cinco modelos con los mismos datos y validación cruzada: Random Forest, SVR, Decision Tree, Gaussian Process y una red neuronal profunda. El DNN obtuvo 0.96 de exactitud, pero su validación cayó a 0.86 una brecha que vale la pena observar. Los demás modelos quedaron por debajo en exactitud, aunque el trabajo también midió tiempos de entrenamiento, donde los modelos más simples tienen ventaja obvia. El sistema quedó integrado en ThingSpeak y genera alertas por correo cuando alguna variable supera los umbrales definidos [19] El siguiente artículo

consistió en la implementación de un sistema de monitoreo mediante una arquitectura IoT en un congelador de la empresa Rico Pollo. Con el fin de registrar la temperatura y el estado de funcionamiento en tiempo real, mejorar la precisión de las mediciones y acelerar la detección de fallas. Se siguió un enfoque experimental estructurado en cinco fases (1) identificación del problema; (2) diseño del sistema (hardware ESP32, sensores SB-70, módulo ADS1115 y protocolo MQTT); (3) construcción e implementación del prototipo; (4) pruebas y validación en un entorno real comparando con el método manual y por último (5) análisis de costos y beneficios. Los resultados mostraron una precisión mejorada ya que el margen de error pasó de $\pm 5^{\circ}\text{C}$

Del análisis de los trabajos revisados se desprenden tres tendencias. En primer lugar, la mayoría de los sistemas de mantenimiento predictivo reportados se orientan a maquinaria industrial de gran escala, como líneas de producción, motores, turbinas o sistemas de climatización centralizados, donde el costo del equipo justifica infraestructuras de monitoreo complejas. En segundo lugar, los enfoques basados en aprendizaje automático, aunque ofrecen una capacidad predictiva superior, dependen de grandes volúmenes de datos históricos de operación, y en particular de registros de fallas etiquetados, que rara vez existen para equipos pequeños y que solo pueden obtenerse después de un periodo prolongado de monitoreo. En tercer lugar, los trabajos aplicados a equipos de refrigeración comercial tienden a centrarse en la temperatura como variable principal, lo que permite advertir que el producto está en riesgo, pero no identificar la causa de la desviación ni anticiparla a partir del comportamiento del compresor. En conjunto, en los trabajos revisados son escasos los sistemas que integran la medición simultánea de variables eléctricas y térmicas en un dispositivo de bajo costo, diseñado específicamente para congeladores comerciales de pequeña capacidad y validado sobre un equipo real en operación, y ninguno considera de forma explícita el efecto de la calidad del suministro eléctrico sobre el desempeño del equipo. Esta brecha motiva el presente trabajo, que propone un sistema de monitoreo continuo y diagnóstico por reglas para este tipo de equipos, cuyos registros constituyen además la base de datos necesaria para el desarrollo posterior de modelos de predicción de fallas.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Sistemas de refrigeración

Los sistemas de refrigeración por compresión constan, básicamente, de cuatro elementos que se consideran fundamentales a través de los cuales circula un fluido refrigerante. La [figura 1](#), muestra el ciclo de refrigeración por compresión de vapor.

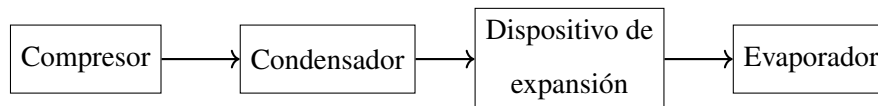


Figura 1: Ciclo de refrigeración por compresión de vapor

2.1.1 Funciones principales

La función principal se muestra en la [figura 2](#) como también se describe la función de cada uno de los componentes:

- **Compresor:** Aspira el fluido refrigerante a la presión de baja establecida y lo comprime elevando su presión y temperatura hasta unos valores tales que se pueda efectuar la condensación. La descarga la efectúa al condensador.
- **Condensador:** Es el elemento de la instalación que se encarga de pasar el estado de vapor del fluido refrigerante a estado líquido. El fluido refrigerante entra en el condensador en estado de gas (vapor recalentado) y sale en estado líquido a la temperatura que se condensó o incluso a una temperatura menor si se produce enfriamiento. El fluido refrigerante cede su calor al agente condensante (aire o agua).
- **Dispositivo de expansión:** Hace que el fluido, que entra en estado líquido, sufra una caída de presión (y temperatura) hasta la necesaria en el evaporador. También controla la cantidad de fluido refrigerante que debe entrar en el evaporador.
- **Evaporador:** Se encarga de enfriar o acondicionar la cámara. Puede estar dentro o fuera de la misma. Su misión es que el fluido refrigerante, que entra a baja presión y temperatura, efectúe el enfriamiento de la cámara

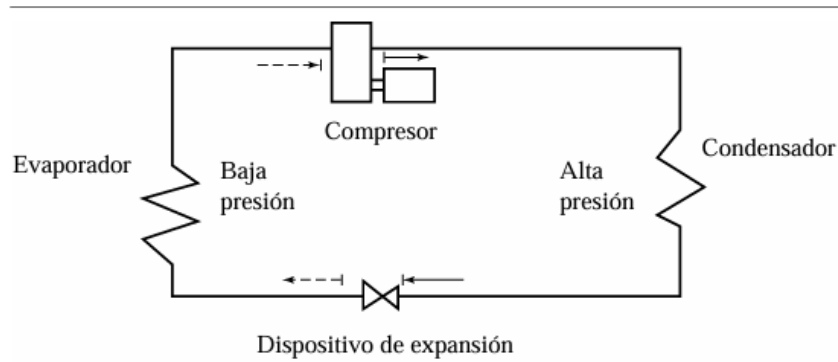


Figura 1.1 Elementos fundamentales de un sistema de refrigeración.

Figura 2: Elementos fundamentales de un sistema de refrigeración.

[20]

2.2 ¿Qué es IoT?

El término IoT, o Internet de las cosas, se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos. El Internet de las Cosas integra las "cosas" de uso diario con internet.

2.2.1 IoT Industrial (IIoT)

IoT industrial (IIoT) se refiere a los dispositivos inteligentes utilizados en la fabricación, el comercio minorista, el sector de la salud y otras empresas para generar eficiencias empresariales. Los dispositivos industriales, que van desde los sensores hasta los equipos, proporcionan a los empresarios datos detallados y en tiempo real que pueden utilizarse para mejorar los procesos empresariales.

Algunos ejemplos de IoT en la industria son:

- **Fabricación :** El IoT empresarial en la fabricación utiliza el mantenimiento predictivo para reducir el tiempo de inactividad no planificado y la tecnología de dispositivos que forman parte de la ropa para mejorar la seguridad de los trabajadores.
- **Automóviles :** Los análisis y la robótica controlados por sensores aumentan la eficiencia en la fabricación y el mantenimiento de automóviles.

- Logística y transporte : Los dispositivos de IoT comerciales e industriales son útiles para administrar la cadena de suministro, incluida la gestión de inventarios, las relaciones con los proveedores, la gestión de flotas y el mantenimiento programado. [21]

2.2.2 Arquitectura de IoT

La arquitectura de IoT es el plan estructural que define cómo todos estos dispositivos de iot, plataformas, y las aplicaciones interactúan. Muestra cómo fluyen los datos.

Una arquitectura se divide a través de componentes que cada uno de estos conforman la arquitectura de IoT.

- Componente de seguridad y gestión
- Componente de aplicaciones y análisis
- Componente de integración [22]

2.2.3 Beneficios del IoT

La importancia del IoT se justifica en que es un habilitador de la eficiencia sistémica. Al poner en red objetos y equiparlos con la capacidad de comunicarse y analizar datos, el IoT va más allá de la automatización para optimizar en base a datos. Esto se traduce en una seguridad mejorada con respuestas automatizadas a incidentes y una gestión más eficiente de recursos gracias al monitoreo y análisis constante de datos. [23]

2.2.4 Computación en la nube

La computación en la nube o "Cloud Computing" es una tecnología mediante la cual las empresas pueden tener acceso remoto a espacios de almacenamiento de archivos digitales, diferentes programas de software e incluso el procesamiento de datos.[24]

La computación en la nube es la disponibilidad a pedido de recursos de procesamiento (como infraestructura y almacenamiento), como servicios a través de Internet. Elimina la necesidad

de que las personas y las empresas administren ellos mismos los recursos físicos y solo paguen por lo que usan . [25]

2.2.5 Plataformas en la nube para IoT

Según Gartner, en su directorio de plataformas industriales, se listan mas de 140 productos solo en esa categoría. [26]

Algunas de las plataformas en la nube mas utilizadas son las siguientes en la cual también se presenta su definición de cada una de ellas.

- Microsoft Azure: Es una plataforma de software que permite a las organizaciones conectarse, monitorear y manejar dispositivos del internet de las cosas y tareas a escala. El software provee herramientas tales para la provisión de dispositivos, colección de datos y análisis, y asegurar comunicación entre dispositivos y la nube. Soporta la integración con sistemas de negocio existentes y ofrece capacidad para el mantenimiento predictivo[27]
- Amazon Web Services : AWS es un proveedor de servicios en la nube, permite disponer de almacenamiento, recursos de computación, aplicaciones móviles, base de datos y una extensa modalidad de cloud computing.[28] Amazon Web Services (AWS) es la nube más adoptada y completa en el mundo, que ofrece más de 200 servicios integrales de centros de datos a nivel global.[29]
- IBM Watson IoT : Es un software diseñado para conectar IoT con dispositivos con capacidad de computación en la nube, haciendo posible a las organizaciones la colección de datos, el manejo y el análisis de datos provenientes de sensores, equipos, y dispositivos. También apoya funciones como visualización en tiempo real, manejo de tiempo de vida en dispositivos, alertas, procesamiento de eventos basados en regla y la integración con sistemas empresariales.[30]
- Google Cloud IoT Platform: El Google Cloud IoT Platform es una suite de servicios de Google Cloud diseñada para conectar, procesar, almacenar y analizar datos procedentes

de dispositivos IoT a escala global, tanto en la nube como en el borde (edge).[31]. Se centra en ofrecer soluciones seguras, escalables y gestionadas para gestionar millones de dispositivos conectados. [32]

- Oracle IoT Cloud: Oracle Internet of Things (IoT) Cloud Service es una plataforma gestionada como servicio (PaaS) en la nube que permite conectar dispositivos, analizar datos en tiempo real y integrarlos con aplicaciones empresariales. Facilita la toma de decisiones estratégicas al ofrecer una arquitectura robusta para gestionar grandes volúmenes de datos generados por dispositivos IoT.[33]

2.3 Sistemas embebidos y microcontroladores

Los sistemas embebidos son dispositivos de computación especializados diseñados para realizar funciones dedicadas dentro de sistemas más grandes, requiriendo optimización de código eficiente y programación específica de hardware para asegurar un rendimiento confiable dentro de las limitaciones de recursos. [34]

2.3.1 Arquitectura de un sistema embebido

De acuerdo a la definición de Liora la arquitectura está diseñada para optimizar el rendimiento, la fiabilidad y la eficiencia energética para tareas específicas.

2.4 Microcontroladores

Un microcontrolador es una pequeña computadora en un solo circuito integrado que esta diseñado para realizar tareas específicas con sistemas electrónicos. Combina las funciones de una unidad de procesamiento central conocida como "CPU", memoria, interfaces de entrada y salida y todo en un solo chip. Un microcontrolador es una computadora independiente que puede ser utilizada en sistemas embebidos para ejecutar tareas específicas. Lo que incluye un microcontrolador es un procesador que corre instrucciones, una memoria que almacena los programas y los datos, puertos de entrada y salida para conectarse con botones, sensores, pantallas, etc. y por último puertos de comunicación para poder conectarse a otros dispositivos.

2.4.1 Características del microcontrolador ESP32

Procesador y Memoria El ESP32 cuenta con uno o dos núcleos Xtensa 32-bit LX6 a hasta 240 MHz, 448 KB de ROM, 520 KB de SRAM, y 16 KB de SRAM en RTC. También soporta flash y RAM externa via QSPI. Conectividad: Wi-Fi 802.11 b/g/n con hasta 150 Mbps, y Bluetooth 4.2 BR/EDR + Bluetooth LE dual mode con potencia de transmisión de hasta +9 dBm y sensibilidad de -94 dBm en BLE. Periféricos: Dispone de 34 GPIOs programables, ADC de 12 bits con hasta 18 canales, 2 DAC de 8 bits, 10 sensores táctiles capacitivos, 4 interfaces SPI, 2 I2C, 2 I2S, 3 UART, controlador Ethernet MAC con soporte IEEE 1588, interfaz TWAI (compatible con CAN 2.0), RMT, y hasta 16 canales de LED PWM. Gestión de Energía :Cinco modos de energía: Activo, Modem-sleep, Light-sleep, Deep-sleep e Hibernación. En Deep-sleep el consumo baja a 10 μ A, y en Hibernación a solo 5 μ A. Incluye coprocesador ULP para operación en bajo consumo. Seguridad: Secure boot, cifrado de flash, 1024 bits OTP, y aceleración criptográfica por hardware para AES, SHA-2, RSA y generador de números aleatorios (RNG). Alimentación: Voltaje de operación de 2.3 V a 3.6 V, con 3.3 V recomendados y corriente mínima de 500 mA. Fabricación:Fabricado en tecnología TSMC de 40 nm de bajo consumo, disponible en encapsulados QFN 5×5 mm y QFN 6×6 mm.[35]

CAPÍTULO 3: METODOLOGÍA

El propósito de esta investigación es el desarrollo de un sistema de monitoreo inteligente con IoT y algoritmos de IA, basado en la medición en tiempo real de las variables de voltaje, corriente, temperatura.

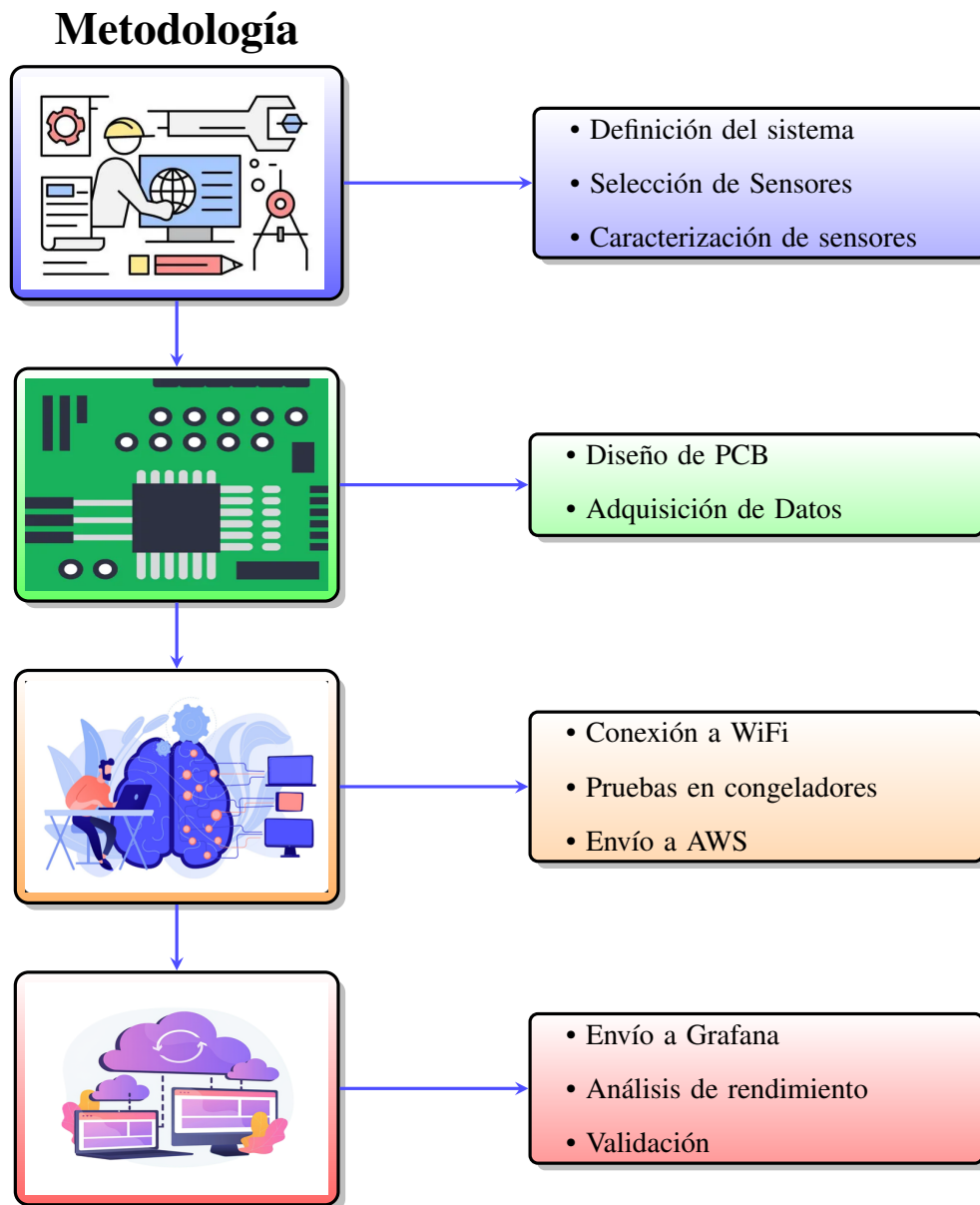


Figura 3: Representación gráfica de la metodología

El presente trabajo corresponde a una investigación aplicada de desarrollo tecnológico, con enfoque cuantitativo y validación experimental en campo. Su propósito fue diseñar, construir y evaluar un sistema electrónico para el monitoreo continuo de las variables eléctricas y térmicas de un congelador comercial, capaz de detectar condiciones anómalas de operación y notificarlas de forma remota.

El desarrollo se organizó en seis etapas secuenciales, derivadas de los objetivos específicos de la investigación. El proceso no fue lineal: el diseño del hardware se resolvió mediante prototipos sucesivos, en los que los resultados de cada versión orientaron las modificaciones de la siguiente, y los datos obtenidos durante la validación en campo se utilizaron para ajustar los umbrales de diagnóstico del firmware. La Figura 4 muestra las etapas, su correspondencia con los objetivos específicos y los dos ciclos de retroalimentación que caracterizaron el proceso.

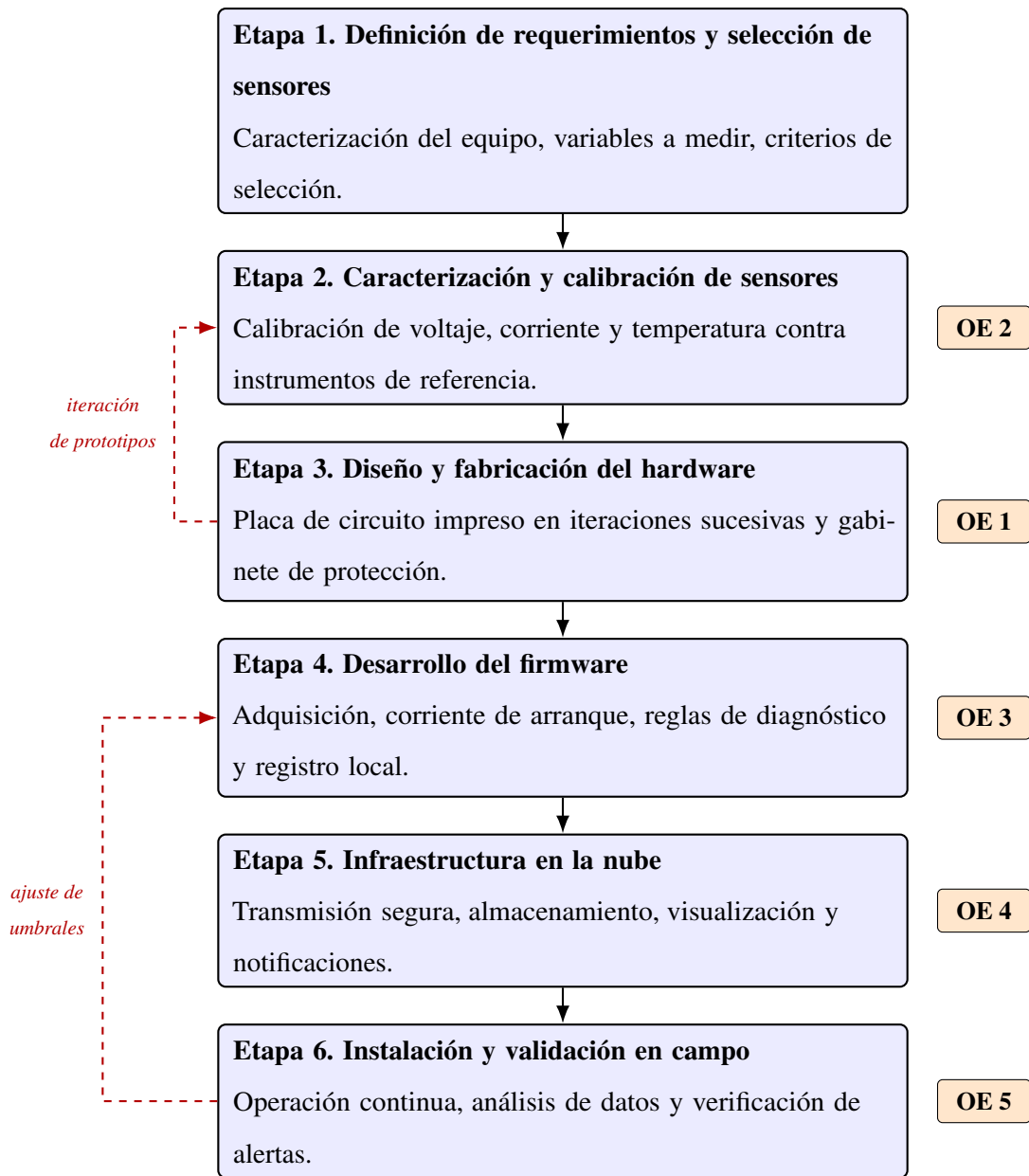


Figura 4: Etapas de la metodología y su correspondencia con los objetivos específicos (OE). Las flechas punteadas indican los ciclos de retroalimentación del proceso.

La Tabla 1 resume, para cada etapa, el objetivo específico al que responde, su producto principal y el criterio con el que se consideró concluida.

Tabla 1: Etapas de la metodología, productos y criterios de cumplimiento.

Etapas	OE	Producto	Criterio de cumplimiento
1. Requerimientos	–	Especificación de variables y sensores seleccionados	Cada variable requerida cuenta con un sensor compatible con el ESP32.
2. Calibración	2	Constantes de calibración y error de medición por sensor	Error cuantificado contra instrumento de referencia en varios puntos del rango.
3. Hardware	1	PCB fabricada y ensamblada; gabinete impreso	Operación autónoma desde la red eléctrica con todos los sensores integrados.
4. Firmware	3	Programa del ESP32 con lógica de diagnóstico	Adquisición estable, detección de las condiciones definidas y registro en microSD.
5. Nube	4	Arquitectura en AWS y tablero en Grafana	Datos visibles en el tablero y notificación recibida ante una alerta de prueba.
6. Validación	5	Base de datos de operación y análisis de resultados	Monitoreo continuo sin pérdida de datos y verificación de alertas ante eventos reales.

A continuación se describe cada etapa. Los detalles técnicos de su implementación se desarrollan en las secciones posteriores de este capítulo.

Etapas 1. Definición de requerimientos y selección de sensores

El punto de partida fue la caracterización del equipo que se pretendía monitorear. Se tomó como caso de estudio un congelador comercial Ojeda CHP-68 de 113 L, con refrigerante R-134a, alimentación nominal de 127 V a 60 Hz, corriente nominal de 2.0 A y temperatura de conservación de -18°C . A partir de estas características y de los modos de falla más frecuentes en compresores herméticos, se definieron las variables a medir: el voltaje de

alimentación, la corriente consumida por el compresor, incluida su corriente de arranque, la temperatura en dos puntos de la cámara y la temperatura de descarga del compresor.

Con base en estas variables se establecieron los requerimientos funcionales del sistema: operar de manera autónoma alimentado desde la misma toma del congelador, conservar los registros ante pérdidas de conectividad, transmitir los datos a una plataforma remota y notificar automáticamente al responsable del equipo cuando se detectara una condición anómala, todo ello con componentes de bajo costo y amplia disponibilidad comercial.

Los sensores se seleccionaron según su rango de medición, su compatibilidad eléctrica y de protocolo con el microcontrolador ESP32, su disponibilidad y su costo. Se eligieron el transformador de voltaje ZMPT101B, que proporciona aislamiento galvánico respecto a la red; el sensor de corriente de efecto Hall ACS712; tres sensores digitales de temperatura DS18B20, que comparten un único bus 1-Wire.

Etapas 2. Caracterización y calibración de sensores

Antes de su integración definitiva, cada sensor se caracterizó y calibró de forma individual para garantizar la validez de las mediciones sobre las que opera la lógica de diagnóstico.

Para el sensor de voltaje se ajustó primero la ganancia del módulo hasta obtener una señal senoidal sin recortes ni distorsión, y posteriormente se determinó la constante de calibración conectándolo a la red de 127 V. El error resultante se evaluó comparando las lecturas del sensor contra un multímetro Fluke 101 en cuatro niveles de voltaje obtenidos mediante un variador.

Para el sensor de corriente se construyó un banco de pruebas con lámparas incandescentes conectadas en paralelo y controladas por interruptores independientes, que permitió aplicar cargas resistivas conocidas en escalones de aproximadamente 1 A. El circuito se verificó previamente mediante simulación en Multisim. La calibración consistió en determinar el voltaje de reposo del sensor en ausencia de carga, que se resta en cada lectura para eliminar el error de desplazamiento, y en comparar la corriente calculada contra un instrumento de referencia.

Los sensores de temperatura se identificaron por su dirección única en el bus 1-Wire y se compararon contra un instrumento de referencia

Etapas 3. Diseño y fabricación del hardware

El hardware se desarrolló mediante prototipos sucesivos. Cada versión se evaluó en funcionamiento y sus limitaciones definieron las mejoras de la siguiente: la primera dependía de un puerto USB para su alimentación; la segunda incorporó una fuente AC/DC integrada, barrenos de sujeción y capacitores de desacoplamiento; la tercera integró todos los sensores en la placa para eliminar el cableado entre módulos, y la versión final redujo las dimensiones de la tarjeta e incorporó un varistor para la supresión de transitorios provocados por el arranque del compresor. El diseño consideró criterios de la norma IPC-2222, la separación de las etapas de potencia, analógica y digital, y la minimización de la longitud de las pistas de los buses de comunicación.

Las placas se fabricaron externamente a partir de los archivos Gerber y se ensamblaron en el laboratorio. De forma paralela, se diseñó en Fusion 360 un gabinete impreso en PLA que aloja la placa, facilita la ventilación de la fuente de alimentación, concentra las conexiones de corriente alterna y permite el paso ordenado del cableado de los sensores hacia el congelador.

Etapas 4. Desarrollo del firmware

El firmware del ESP32 fue estructurado en tres fases de operación. Al ser encendido el sistema, durante la primera fase fueron registrados durante 10 s los valores de corriente de arranque del compresor, con una frecuencia de muestreo de 1 kHz, y fueron calculadas sus principales estadísticas. Posteriormente, fueron ejecutados ciclos de monitoreo en los que las variables fueron adquiridas cada 2 s, fueron acumuladas 900 muestras en ventanas de 30 minutos y, a partir de sus promedios, fue evaluado un conjunto de reglas de diagnóstico. De forma paralela, la corriente instantánea fue supervisada durante una tercera fase, mediante la cual fueron identificados los eventos de encendido y apagado del compresor.

A partir de las señales adquiridas, fueron calculadas las variables derivadas del sistema: voltaje

y corriente eficaces, temperatura media de la cámara, gradiente térmico entre los sensores de la cámara, diferencial de temperatura entre la cámara y la descarga del compresor, potencia aparente y una estimación del coeficiente de rendimiento basada en el ciclo de Carnot. Las reglas de diagnóstico fueron definidas a partir de la relación entre estas variables y los modos de falla del ciclo de refrigeración. Entre las condiciones consideradas fueron incluidos los ciclos de trabajo prolongados, la activación del protector térmico, los indicios de pérdida de refrigerante, la pérdida de capacidad de enfriamiento y las fallas de sensores. Para cada condición, fue implementado un mecanismo mediante el cual se evitó el reenvío de una misma alerta mientras la condición permaneciera activa.

Todos los registros fueron almacenados localmente en una tarjeta microSD, independientemente del estado de la conexión a internet.

Etapas 5. Infraestructura en la nube

La comunicación con la nube se implementó sobre AWS IoT Core, que actúa como intermediario MQTT. La conexión emplea TLS 1.2 con autenticación mutua mediante certificados X.509, de modo que solo el dispositivo registrado puede publicar datos. El ESP32 publica cada 30 s en dos tópicos: uno para las variables de operación y otro, que se activa únicamente ante una condición de falla, para las alertas.

Sobre este intermediario se configuró una cadena de servicios: reglas de enrutamiento que dirigen las mediciones a una tabla de DynamoDB y las alertas a una función de AWS Lambda; una segunda función que traslada periódicamente las métricas a CloudWatch, programada mediante EventBridge; el servicio SNS para el envío de notificaciones por correo electrónico, y un tablero en Grafana Cloud, conectado a CloudWatch, para la visualización de las variables. Los permisos entre servicios se administraron mediante roles de IAM. La arquitectura se verificó por componentes, utilizando el cliente MQTT de prueba de AWS y eventos de prueba en Lambda, antes de integrarla con el firmware definitivo.

Etapla 6. Instalación y validación en campo

El sistema se instaló junto al congelador de prueba en el laboratorio. Con el apoyo de un técnico en refrigeración, se realizaron perforaciones en la cámara sin afectar la tubería del equipo para introducir los sensores de temperatura, que se ubicaron en las zonas superior e inferior de la cámara, y el tercer sensor se fijó en la línea de descarga del compresor. Entre la toma de corriente y el equipo se conectó un supresor de picos.

La validación consistió en la operación continua del sistema durante 61.3 h, en condiciones normales de uso del congelador. Durante este periodo se verificó la integridad de la cadena de transmisión y se recolectaron los registros de la tarjeta microSD, para obtener las estadísticas de las variables eléctricas y térmicas, identificar los ciclos del compresor y determinar su duración.

Los resultados de este análisis se utilizaron para ajustar los umbrales de las reglas de diagnóstico, en particular el tiempo máximo de ciclo del compresor, que en su definición inicial generaba falsas alarmas. Finalmente, se verificó el funcionamiento de la cadena completa de detección y notificación ante eventos reales: un ciclo de trabajo prolongado del compresor, la desconexión de un sensor de temperatura y la apertura prolongada de la puerta del equipo.

3.1 Desarrollo

3.1.1 Placa de circuito impreso

Para el diseño de circuito impreso se consideraron en cuenta diversos factores técnicos que pudieran afectar al buen funcionamiento. Uno de los aspectos más importantes y prioritarios fue la prevención de cortocircuitos ya que estos dañan no solo la placa sino también otros componentes interconectados a la tarjeta madre, en este caso el compresor del congelador se puede ver afectado por lo que este aspecto es crucial. Asimismo, se tomó en cuenta otra de las consideraciones la cual fue el ruido electrónico, este puede ser proveniente de los propios componentes incorporados a la tarjeta. Por esta razón, la correcta distribución física de los sensores y componentes electrónicos fue un paso importante en la realización de esta tarjeta, con el objetivo de eliminar interferencias y asegurar la funcionalidad del sistema.

Todo el diseño electrónico de la placa fue siguiendo la norma IPC -2222, como también, siguiendo con esta norma se colocaron tornillos de sujeción para garantizar una adecuada fijación mecánica y evitar desplazamientos que pudieran afectar el funcionamiento del sistema embebido. Considerando todos estos aspectos se realizó el esquema del circuito el cual permita desempeñar la función de hacer que cada uno de los sensores y componentes electrónicos puedan trabajar de manera simultanea y correcta. La figura 5 muestra el diagrama eléctrico.

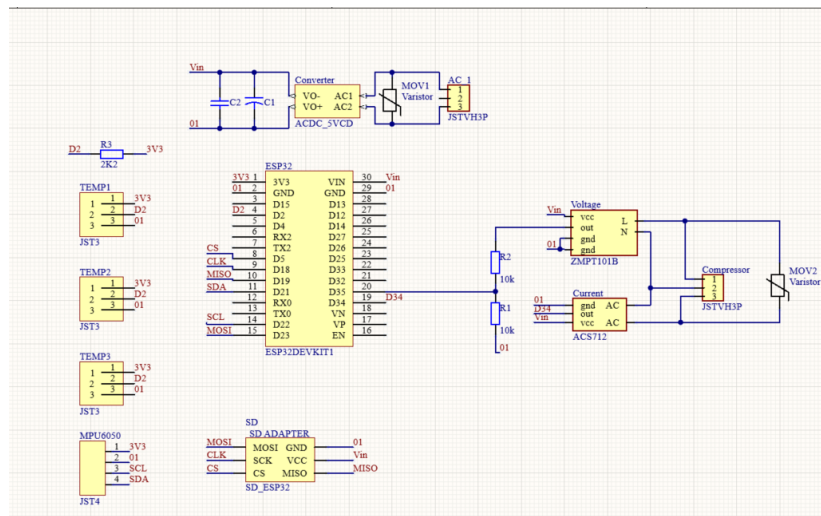


Figura 5: Esquema del circuito

El objetivo del diseño esquemático trascendió la mera conectividad, enfocándose en garantizar la integridad de señal, la estabilidad del riel de alimentación y la minimización del ruido eléctrico. Para lograrlo, se implementaron estrategias específicas. Primero, se definió un plano de tierra (GND) sólido y continuo para la etapa digital del microcontrolador ESP32, replicando un esquema similar para las etapas analógicas de los sensores con el fin de segregar las tierras y aislar las señales sensibles del ruido digital. Segundo, se incorporaron filtros de desacoplamiento (capacitores cerámicos de 100 nF y electrolíticos de 10 μ F) en la entrada de alimentación de cada circuito integrado para suprimir transitorios y garantizar un voltaje estable. Tercero, en la transición del esquema al layout, se priorizó minimizar la longitud de las trazas críticas, especialmente las del bus I2C (SDA, SCL), para reducir la inductancia parásita. El esquema resultante, que actúa como el mapa de interconexión definitivo, se muestra en la Figura 5. Este documento fue la base directa para la etapa de ruteo físico (layout) de la PCB,

cuyos resultados finales. El diseño en 2D y una representación 3D de la placa ensamblada— se presentan en las Figuras 67. El layout final fue precedido por un proceso iterativo de diseño y revisión. La primera versión (figura 6) presentó una tarjeta con la capacidad de medir el voltaje la corriente y temperatura con el impedimento de tener que estar conectado al puerto USB de la computadora.

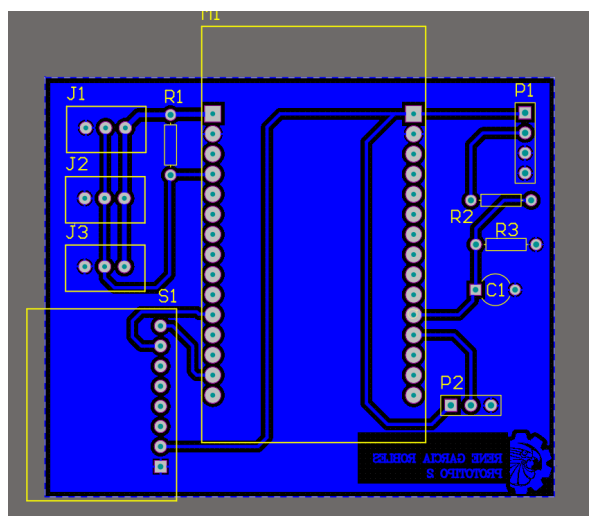


Figura 6: Prototipo uno

La segunda versión introdujo cuatro mejoras críticas:

Un módulo convertidor AC/DC integrado, que permitió una alimentación autónoma del sistema desde la red eléctrica, eliminando la dependencia del puerto USB. Barrenos de sujeción para garantizar una fijación mecánica robusta. Una red de capacitores de desacoplamiento en los rieles de alimentación, para filtrar el ruido de conmutación y asegurar la estabilidad del voltaje. Un ventilador de 5 VCD con control térmico pasivo, para disipar el calor generado por el ESP32 y prevenir el throttling por sobrecalentamiento. Figura 7

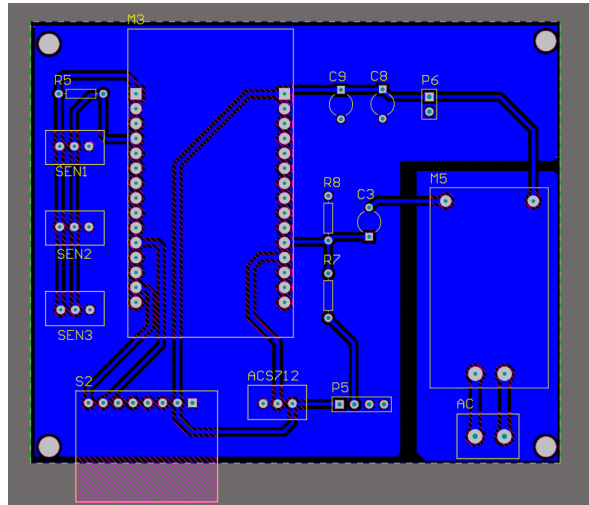


Figura 7: Prototipo 2

La tercera versión del diseño tuvo como objetivo principal la integración completa de los sensores en el layout de la PCB. Esto eliminó la necesidad de conexiones cableadas externas entre módulos, consolidando todas las interconexiones en el propio sustrato para mejorar la confiabilidad, reducir puntos de falla y minimizar el ruido introducido por cables largos. Figura

8

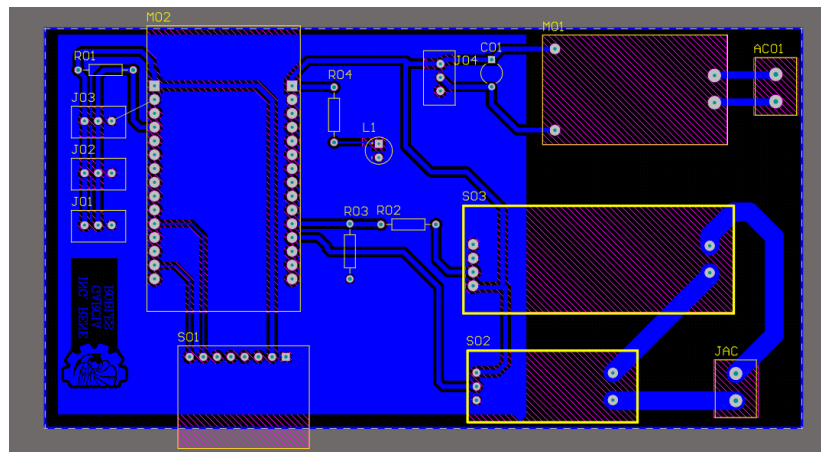


Figura 8: Prototipo 3

La desventaja de esta modificación fue el tamaño de la placa, por lo que se redujo nuevamente las dimensiones, y una re configuración completa a un layout de doble cara. Esta última medida, junto con el cambio de lugares de algunos sensores, permitió aislar físicamente las

redes críticas y eliminar los puntos de falla potenciales.

Tras este proceso iterativo, se consolidó el diseño final del sistema embebido. Este prototipo integra de manera óptima ya que, todos los sensores (voltaje, corriente, temperatura) están interconectados en la PCB. La robustez mecánica se logró con la inclusión de tornillos de sujeción para un montaje seguro. Y por último se incorporó un varistor (MOV) para supresión de transitorios de voltaje y se expandieron los planos de tierra (L y N). Esta medida incrementa la capacidad de disipación de corriente, protegiendo al sistema contra los picos de encendido (inrush current) generados por el arranque del compresor. La figura 9 , 10 y 11 muestran el diseño final de la PCB

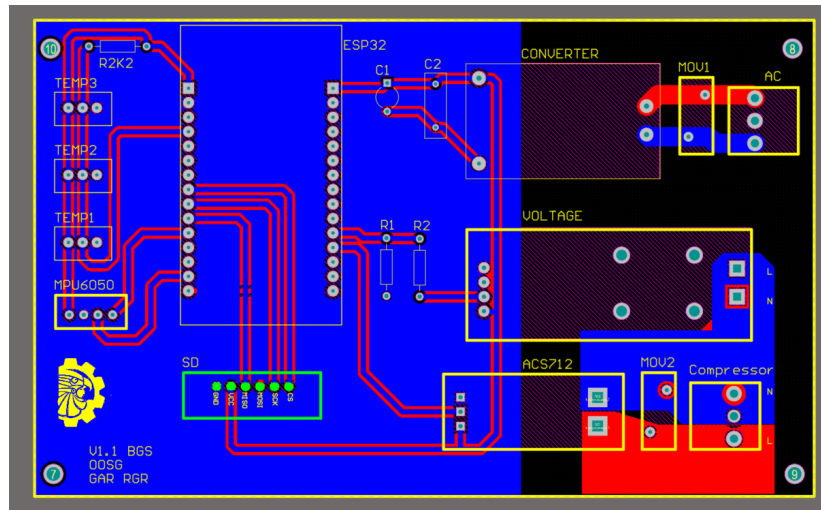


Figura 9: Layout cara inferior

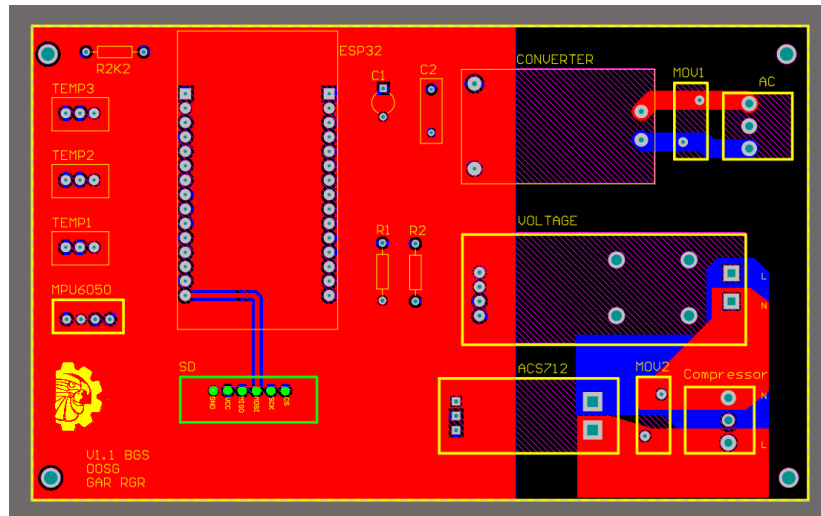


Figura 10: Layout cara superior

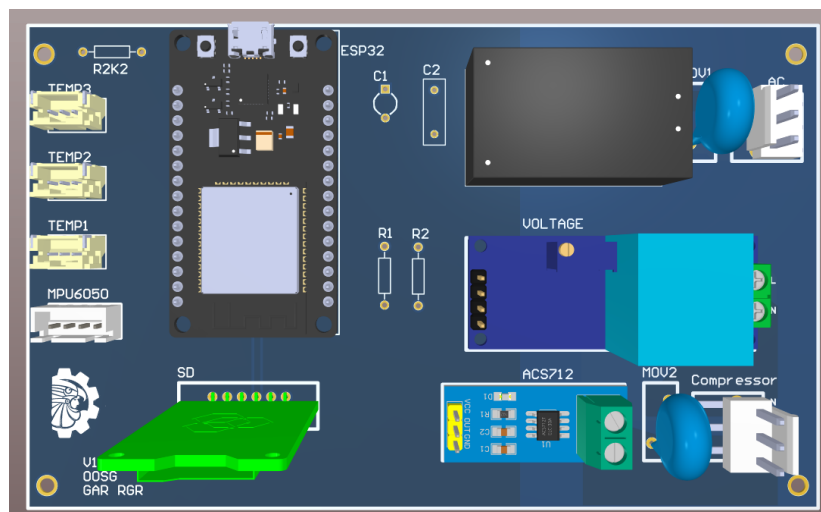


Figura 11: Modelo 3D del sistema embebido

La etapa de la manufactura se realizó con la fabricación externa al servicio de manufactura JLCPCB. La cual a a partir de los archivos "Gerber" y de "Drill" se solicitó la fabricación de un pequeño lote de 5 placas especificando los parámetros estándar. La disponibilidad de múltiples unidades permite realizar pruebas en campo, como también, hacer mejoras para una versión más equipada y realizar pruebas simultaneas a diferentes congeladores con diferentes problemas para corroborar la eficiencia del sistema lo cual es crucial en la fase experimental. las figuras 12 , 13, 14, y 15 muestran la fabricación de la PCB.

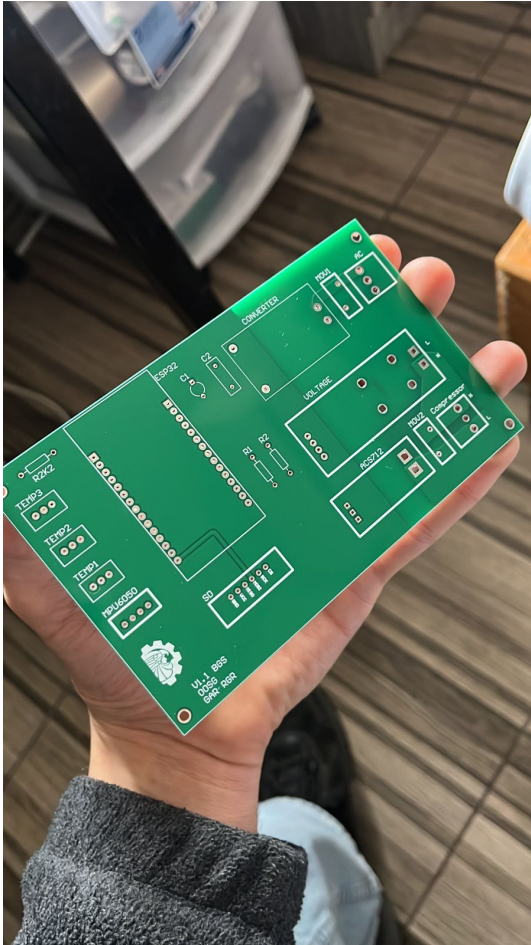


Figura 12: Vista superior PCB

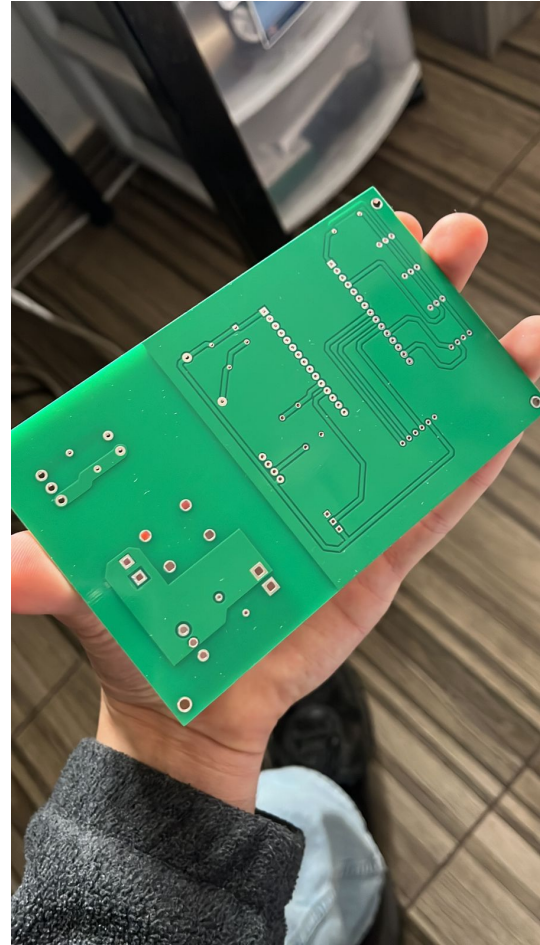


Figura 13: Vista inferior

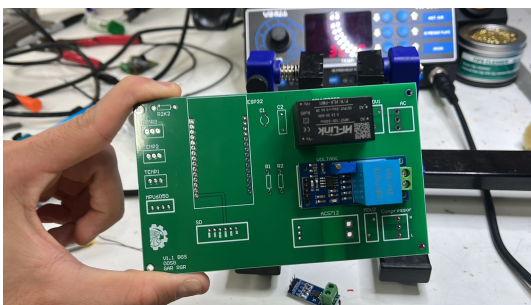


Figura 14: instalando componentes en PCB

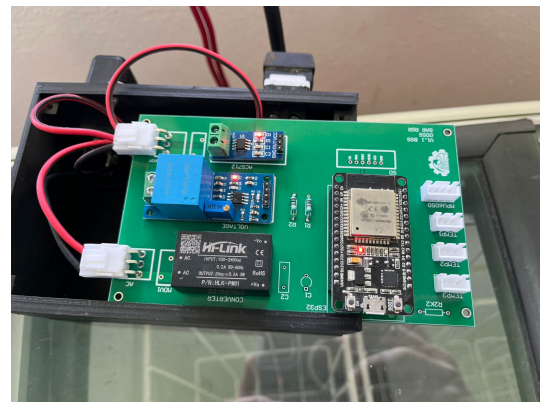


Figura 15: Sistema bajo pruebas

3.1.2 Diseño de la caja para Protección y Montaje

La etapa de diseño mecánico fue de suma importancia, ya que permitió albergar el sistema en un contenedor que cumpliera con los objetivos de accesibilidad, para poder conectar el cable USB de programación, ventilación para evitar el sobrecalentamiento en la fuente de voltaje como también en el ESP32, punto de fijación que permitiera el montaje seguro y fácil de colocar en el congelador ya sea en la misma cámara del compresor o externamente , un diseño compacto y ligero que no alterara la dinámica de vibración. Para realizar todos estos criterios se optó por un diseño que estuviera formado por 2 etapas, la base y la tapa esto permite que el sistema sea fácil de manipular al solo quitar la tapa del case y poder modificar o cambiar algún componente del sistema. Para realizar el diseño CAD se utilizó el software "FUSION360" el cual permitió realizar el diseño de manera sencilla.

El diseño también cuenta con una base para anclarse a la pared por si el sistema es colocado afuera del congelador y también por si se requiere colocar en una superficie rígida como por ejemplo una pared, una madera, concreto, etc. A continuación las figuras 16, 17 y 18 muestran el resultado final del diseño CAD

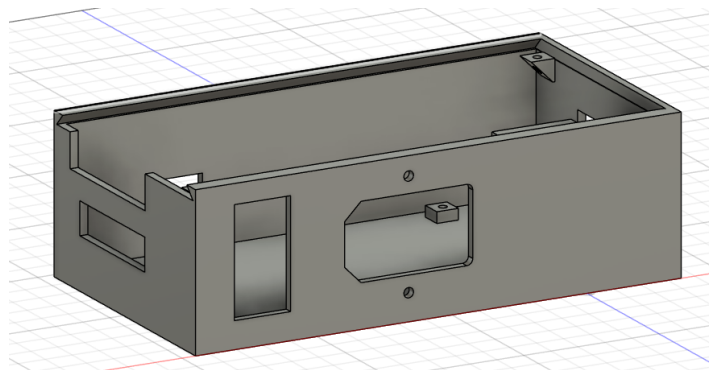


Figura 16: Case vista lateral

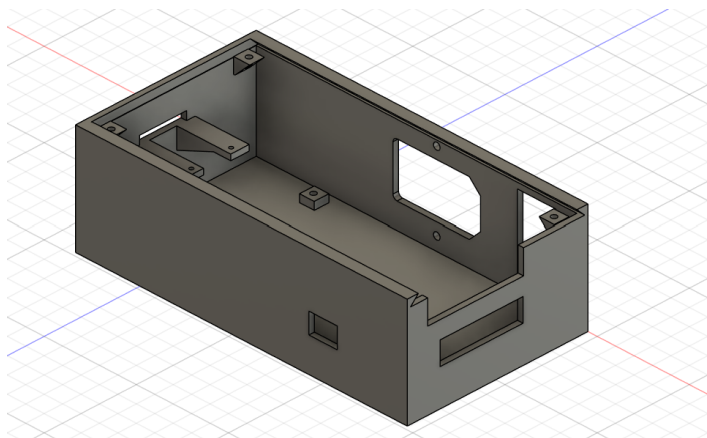


Figura 17: Case vista lateral izquierda

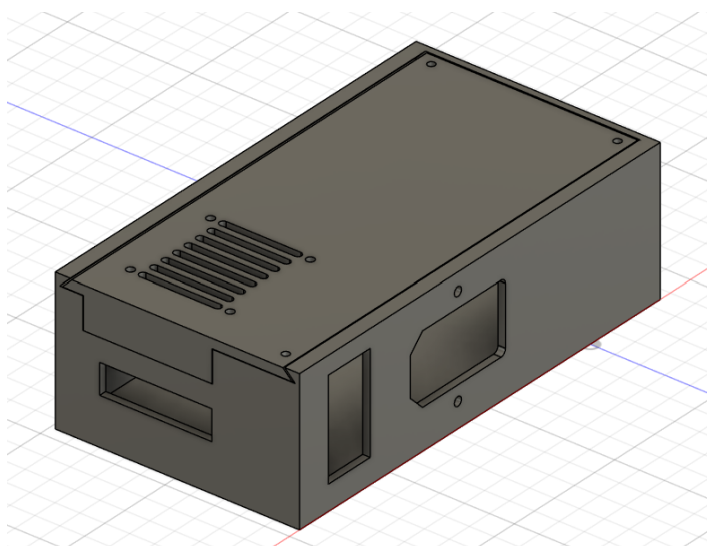


Figura 18: Case final

3.1.3 Firmware del ESP32

El firmware opera en tres fases secuenciales. Al encender el ESP32, la primera fase registra durante 10 segundos el perfil de corriente de arranque del compresor a 1 kHz, sin guardar las 10,000 muestras completas — en su lugar calcula estadísticas acumuladas como el máximo, la suma de cuadrados y la suma absoluta. Esto permite caracterizar el arranque del equipo específico sin saturar la RAM.

Una vez completada la captura inicial, el sistema entra en ciclos de monitoreo de 30 minutos. Cada 2 segundos acumula lecturas de voltaje, corriente y temperatura hasta completar 900

muestras, momento en que calcula promedios y ejecuta la lógica de diagnóstico. La tercera fase corre en paralelo: monitorea la corriente instantánea para detectar los eventos de encendido y apagado del compresor en tiempo real.

3.1.3.1 Variables calculadas

El firmware calcula nueve variables a partir de las señales crudas de los sensores. El voltaje RMS se obtiene directamente del sensor ZMPT101B a través de su librería. La corriente RMS se calcula tomando 1,000 muestras a 1 kHz y aplicando la raíz cuadrada de su media cuadrática, lo que permite medir con precisión tanto la corriente de arranque como la de operación continua.

La temperatura de la cámara se obtiene promediando las lecturas de T1 y T2, los dos sensores ubicados en la parte inferior y superior respectivamente. T3 mide directamente la temperatura de descarga del compresor. A partir de estas tres lecturas se calcula el diferencial térmico entre la cámara y la descarga, variable clave para diagnosticar el estado del ciclo de refrigeración.

La potencia se calcula multiplicando voltaje por corriente. Con este valor y las temperaturas, el firmware estima el Coeficiente de Rendimiento (COP) usando la fórmula de Carnot ajustada con un factor de eficiencia real del 45 %, valor típico para compresores herméticos de pequeña capacidad. La tabla 2 contiene las fórmulas de cada variable

Variable	Fórmula y Descripción
Voltaje RMS	Lectura directa del sensor ZMPT101B vía librería
Corriente RMS	Raíz cuadrada de la media de los cuadrados de 1,000 muestras proveniente del ACS712
Temp. Cámara	Promedio de T1 y T2 : $(T1 + T2) / 2$
Temp Descarga	Lectura directa de T3
Diferencial T	Temperatura cámara menos temperatura de descarga
Potencia	Producto de voltaje por corriente: $V \times I$
COP Carnot	$T_{evapK} / (T_{condK} - T_{evapK})$
COP estimado	COP Carnot $\times$ 0.45 (eficiencia real estimada 45 %)
Gradiente T	Diferencia entre T2 y T1

Tabla 2: Variables Calculadas

3.1.4 Lógica de detección de fallas

El sistema identifica siete condiciones de alerta. Cada una tiene asociada una variable booleana que se activa al detectar la falla y bloquea el reenvío de alertas mientras la condición persiste, evitando saturar el correo del responsable con notificaciones repetidas. La bandera se resetea automáticamente cuando la condición desaparece.

La primera condición detecta ciclos de trabajo excesivos: si el compresor lleva encendido más tiempo del permitido sin alcanzar los $-22\text{ }^{\circ}\text{C}$ de setpoint, se genera una alerta. El tiempo máximo no es fijo — se calcula según la temperatura al inicio del ciclo para evitar falsas alarmas durante el primer arranque o después de una carga de producto. Para temperaturas mayores a $0\text{ }^{\circ}\text{C}$ el límite es 90 minutos; entre $0\text{ }^{\circ}\text{C}$ y $-10\text{ }^{\circ}\text{C}$ baja a 60 minutos; entre $-10\text{ }^{\circ}\text{C}$ y $-18\text{ }^{\circ}\text{C}$ a 40 minutos; y para operación normal por debajo de $-18\text{ }^{\circ}\text{C}$, el umbral es de 35 minutos, valor calibrado con datos reales del equipo.

La segunda condición identifica una posible activación del protector térmico: si el compresor permaneció apagado más de 50 minutos después de un ciclo de más de 40 minutos, es probable que el protector haya cortado la alimentación por sobrecalentamiento. La tercera condición detecta fugas lentas de gas refrigerante monitoreando la corriente: si el compresor

está encendido pero la corriente se mantiene por debajo de 0.5A durante más de una hora, indica que el compresor trabaja sin carga hidrostática, condición compatible con un nivel bajo de R-134a. Las condiciones cuatro y cinco se evalúan al cerrar cada ciclo de 30 minutos, sobre los promedios acumulados. Se verifican cuatro situaciones: compresor apagado si la corriente promedio es menor a 0.30A; falta de gas si el compresor está encendido pero la diferencia entre la temperatura de descarga y la temperatura ambiente es menor 5°C con la cámara por encima de -15°C compresor dañado si la corriente supera los 3A sin caída de temperatura; y pérdida de enfriamiento si la corriente está entre 0.3A y 1.0A con la temperatura de cámara en aumento.

La última condición monitorea el estado de los sensores DS18B20: cuando cualquiera devuelve -127°C , valor que el protocolo 1-Wire usa para indicar desconexión, el sistema registra el error en la tarjeta SD y envía una alerta única a AWS. Una vez que el sensor vuelve a responder correctamente, la bandera se resetea de forma automática. La tabla 3 muestra las fallas por fases

Tabla 3: Diagnóstico de fallas por fases

Fase	Código	Condición detectada
Fase 1 (0–10 seg)	R001	Corriente de arranque
	R002	Voltaje de arranque
Fase 2 (30 min)	R005	Baja compresión
	R006	Falta de gas
	R009	Sobrecarga de refrigerante
	R008	Pérdida de enfriamiento con baja corriente
Fase 3 (90 min – 2 h)	R003	Sobrecalentamiento en marcha
	R004	Falla de termostato/protector
	R007	Falta de gas (fuga lenta)
	R008	Pérdida de enfriamiento con baja corriente
	R009	Sobrecarga de refrigerante
	R010	Degradación del aislamiento/fugas de aire

3.1.5 Almacenamiento local en microSD

El sistema guarda una copia local de todos los datos en una tarjeta microSD, independiente de la conexión a AWS. Esto garantiza que no se pierda información si el WiFi falla o la conexión a la nube se interrumpe temporalmente.

Se generan cinco archivos con funciones distintas. El archivo `datos.csv` es el registro principal: guarda todas las variables calculadas cada 2 segundos con encabezados, lo que facilita su análisis posterior en cualquier herramienta de hoja de cálculo. El archivo `resumen_30min.csv` complementa al anterior con un resumen estadístico de cada ciclo de 30 minutos, incluyendo promedios, máximos y mínimos de voltaje y corriente.

Para el seguimiento de fallas, el sistema genera dos archivos de texto: `alertaCompresor.txt` y `alertaCiclo.txt`, donde se registra cada alerta detectada con su timestamp. El archivo `análisis_arranque.txt` guarda el resultado del análisis de corriente de arranque que se ejecuta cada vez que el ESP32 se reinicia, lo que permite comparar el comportamiento del compresor a lo largo del tiempo. Por último, `error_log.txt` registra los errores de sensores con su marca de tiempo para facilitar el diagnóstico de problemas de hardware.

Esta estructura de archivos permite que el sistema funcione de forma autónoma incluso sin conexión a internet: los datos siguen registrándose localmente y pueden recuperarse conectando la microSD directamente a una computadora.

3.1.6 Selección de sensores

Los sensores se utilizaron por su funcionalidad, sus modos de operación y su complejidad. Se caracterizaron con la hoja de datos dada por el fabricante para proseguir con la etapa de calibración de cada sensor, probar de manera individual asegurando que cumplan con los rangos de medición, precisión y compatibilidad con el microcontrolador ESP32.

3.1.7 ZMPT101B

Para la calibración del sensor de voltaje (ZMPT101B) se realizó una serie de pasos para obtener su calibración que implica una serie de códigos que se ajustaron conforme a los resultados obtenidos. Internamente (figura 19) este sensor esta conformado por un transformador ZMPT101B lo cual permite aislar el circuito de potencia con la parte del circuito que va al microcontrolador.

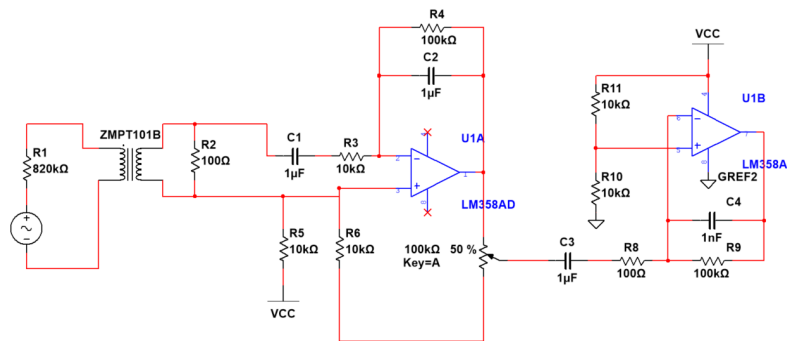


Figura 19: Configuración interna

La configuración del sensor consta de 2 bornes en la entrada que permite la conexión de la corriente alterna y de 4 pines que permiten conectarse al microcontrolador, en este caso permite la conexión al microcontrolador ESP32. Estos pines son de alimentación al sensor (VCC), Voltaje de salida (VOUT), y dos pines para tierra (GND). Es importante mencionar que la salida de este sensor puede variar desde los 2.5 voltios hasta los 5 voltios y la tarjeta ESP32 solo permite un máximo de 3.3 voltios de entrada en cada pin, por lo que si se conecta sin antes un arreglo de resistencias se puede dañar el microcontrolador. En este caso se utilizó un divisor de voltaje con ambas resistencias iguales para disminuir el voltaje a la mitad, y poder así asegurar la protección de la tarjeta ESP32.

Para generar la onda sinusoidal se implementó el código (3.1) que gráfica los datos obtenidos por el sensor.

Listing 3.1: Calibración ZMPT101B

```
1 void setup() {  
2   Serial.begin(115200);  
3   analogReadResolution(12);  
4 }  
5 void loop() {  
6   Serial.println(analogRead(35));  
7   delayMicroseconds(500);  
8 }
```

Al ejecutar el código en el software de Arduino se obtiene la siguiente respuesta la cual se ajustó el potenciómetro que tiene el sensor hasta obtener una forma de onda limpia ya que si la onda tiene buena forma (figura 20), mayor precisión en la lectura.

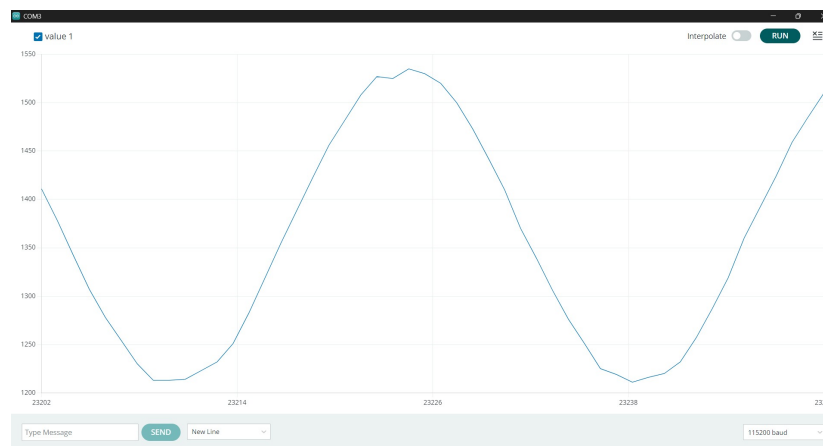


Figura 20: Gráfica del sensor

Al momento de la calibración sucedieron dos fenómenos que se debe de evitar durante la calibración de este sensor en los cuales la gráfica se mostró con picos y/o entre cortada(figura 21). En ambos casos indica una mala calibración .

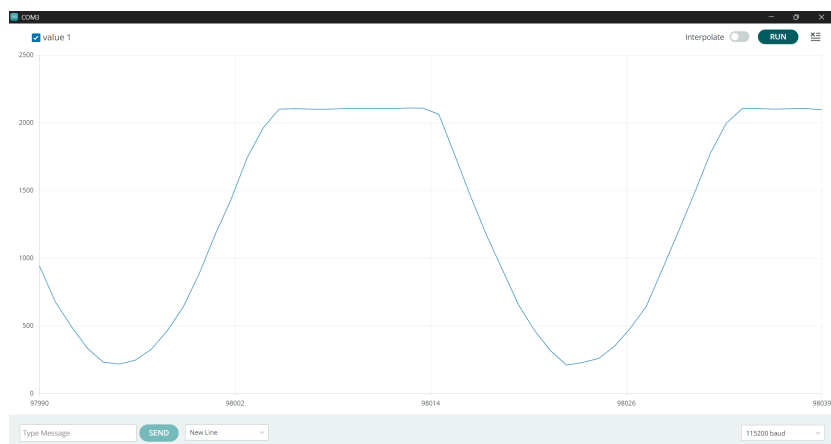


Figura 21: Gráfica no calibrada

Una vez que se configuró la resistencia del sensor y la gráfica nos muestra una onda sinusoidal clara y sin la presencia de picos en ella, se prosiguió al siguiente paso donde consistió en subir la segunda parte del código para la obtención de la constante de calibración, para este paso se utilizó la tarjeta previamente manufacturada para la facilidad de las pruebas; La constante de calibración es esencial para el código final. Para esta segunda se conectó el sensor a la línea de voltaje (127 Voltios en alterna) para tener una referencia de que valor el sensor tiene que buscar. Este proceso se repitió 5 (tabla 4) veces para obtener un promedio del valor de calibración.

Tabla 4: Valores de calibración

No.	Valor de calibración
1	300.50
2	299.50
3	301.00
4	300.50
5	301.25

Después de hacer 5 pruebas el promedio resultó ser de 300.55 por lo tanto el valor de sensibilidad equivale a 300.55 para este específico sensor, sin embargo, las lecturas que mostraba

el sensor eran precisas pero no exactas, para el caso de 127V en corriente alterna el sensor detectaba 124V en corriente alterna y de acuerdo con la formula (ecuación 3.1) para obtener el porcentaje de error se tuvo lo siguiente:

$$\frac{|ValorMedido - ValorReal|}{|ValorReal|} * 100 \% \quad (3.1)$$

Sustituyendo y dando como resultado:

$$\left| \frac{124 - 127}{127} \right| * 100 \% = 2.36 \% \quad (3.2)$$

Por lo que se modificó el valor de calibración utilizando un multímetro "Fluke 101", que permitió hacer la comparativa de las lecturas arrojadas por el sensor y el multímetro. Se realizó los cambios aumentando el valor de calibración de 5 en 5 hasta obtener lecturas de voltajes exactas en ambos dispositivos (sensor y multímetro), dando como resultado un cambio de valor de calibración de 300.55 a 315.55. Con esto el sensor de voltaje no solo era preciso sino también exacto. Por último se modificó el valor de voltaje con la ayuda de un variador de voltaje obteniendo 4 diferentes casos de voltaje y leyendo cada uno de ellos 5, tanto con el multímetro como con el sensor de voltaje.

Se dieron los siguientes casos de lectura, todos los valores estan dado por las unidades de voltio en corriente alterna (VCA).

Tabla 5: Voltaje comparada con el multímetro y el sensor

Voltaje	Multímetro	ZMPT101B
125	125.5	125.62
100	100.7	100.62
50	50.6	50.56
20	20.9	20.34

3.1.8 ACS712 30A

Para la calibración del sensor primero se creó un prototipo que permitiera aumentar la corriente de manera controlada. Se usó un arreglo de focos incandescentes conectados en paralelo y de interruptores para la creación de este prototipo (se muestra en la siguiente figura 22 y 23), lo que permite controlar cada uno de manera independiente permitiendo así el aumento controlado de la corriente.

Para poder fabricar este prototipo se simuló primero y se verificó con en el software de "Multisim de National Instruments".

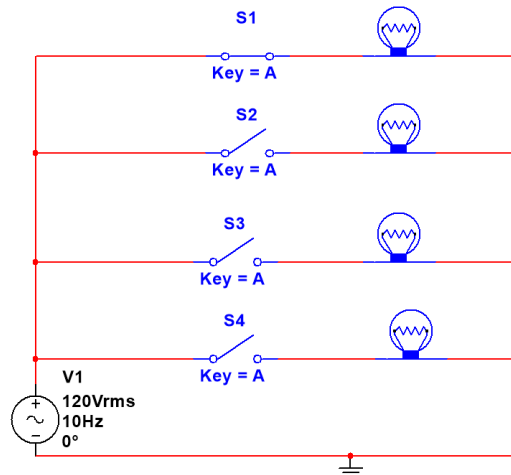


Figura 22: Simulación Lámparas en paralelo

La razón principal de la cual se utilizaron los focos incandescentes se basó en 3 criterios principales. Cada lámpara presenta una carga resistiva pura y consume aproximadamente 1 amperio. En segundo lugar al tener 4 o 5 focos conectados en paralelo se obtiene una corriente considerable y perfecta para no solo calibrar el sensor si no también simular cambios bruscos en el valor de corriente. Finalmente, la naturaleza de la carga resistiva permite simular la corriente de arranque del compresor, haciendo un ambiente ideal para corroborar el comportamiento del sensor de corriente.

Una vez validado la simulación se prosiguió a la construcción del prototipo físico que incluyera los 4 focos y interruptores. para la fabricación de este prototipo físico se implementó sobre

una base de madera, añadiendo los 5 sockets para colocar focos, cable de calibre adecuado, sus respectivos interruptores individuales, una toma corriente, focos y una caja que se fabricó para mitigar el deslumbramiento durante las pruebas, los sockets se colocaron dentro de esta caja atornillados a la base de madera.

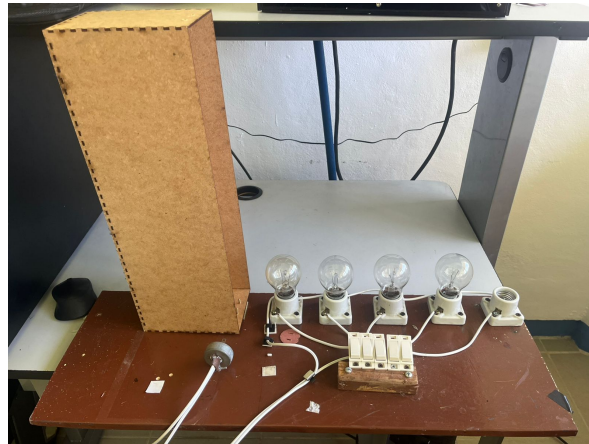


Figura 23: Modelo de corriente

EL sensor ACS712 se puede encontrar de diferentes tipos en el mercado, lo puede haber para medir 5 Amperios, 20 Amperios y 30 Amperios. Para este caso se utilizó el de 30 Amperios ya que existen diferentes tipos de sistemas de refrigeración en los cuales hay diferentes tamaños y por lo tanto las características eléctricas no son las mismas, por lo que es mejor tener un mayor rango de corriente a no poder utilizar el sensor ya que la corriente que demanda el sistema es más que la que este puede soportar.

De acuerdo con la hoja de especificaciones del fabricante este sensor en particular tiene una sensibilidad de 66mV/A . Este constituye la constante de calibración teórica, sin embargo, para lograr una alta precisión en las mediciones, es necesario realizar cambios hasta obtener un valor exacto comparándolo con otro instrumento que en este caso es un amperímetro.

Para la calibración del sensor se implementó un algoritmo cuyo flujo consta de determinar cual es el voltaje de offset (punto cero), donde consiste realizar una lectura de voltaje de salida cuando el sensor tiene una carga de cero amperios. el objetivo es obtener la lectura de voltaje para tener una referencia la cual se resta sistemáticamente para eliminar el error de offset del sensor.

El segundo paso de este algoritmo consiste en el filtrado de la señal ya que esta presenta ruido y oscilaciones, esto permite obtener una lectura de señales mas limpia que permite una visualización de datos más estable.

Una vez pasado por el filtrado de datos el algoritmo promedia las primeras 200 muestras (código 3.2) ya que al tener más datos para promediar más exacto sera el valor final.

Listing 3.2: Calibración ACS71230

```
1 // Medicion de corriente AC (valor RMS)
2 float getCurrentAC() {
3     float sumSq = 0;
4     int samples = 200;
5     for (int i = 0; i < samples; i++) {
6         float voltage = analogRead(ACS712_PIN) * (VREF /
7             ADC_RESOLUTION) - zeroCurrentVoltage;
8         sumSq += voltage * voltage;
9         delay(1);
10    }
11    float rmsVoltage = sqrt(sumSq / samples);
12    float rmsCurrent = rmsVoltage / SENSITIVITY;
13    return rmsCurrent;
14 }
```

Al terminar este proceso de calibración, el sensor se incluyó en la tarjeta madre con la seguridad de que la función de leer corriente se encuentra calibrada.

3.1.9 DS18B20

La calibración del sensor de temperatura no presenta una alta complejidad, dado que se basa únicamente en dos etapas principales para la obtención de la temperatura. En primer lugar, se determina el valor de la resistencia de pull-up. Para ello, se midió la distancia a la que se ubicarían los sensores respecto de los distintos puntos estratégicos dentro del sistema

de refrigeración, con el fin de asegurar una adecuada integridad de la señal y un correcto funcionamiento de la línea de datos.

El sensor de temperatura dispone de tres conductores: uno de alimentación (cable rojo), uno de referencia a tierra, GND (cable negro), y uno de transmisión de datos (cable amarillo). El protocolo de comunicación empleado es del tipo “one-wire”, lo que permite la conexión de múltiples sensores utilizando un único pin del microcontrolador. En la figura 24 se ilustra la configuración empleada para la conexión del sensor con el microcontrolador ESP32.

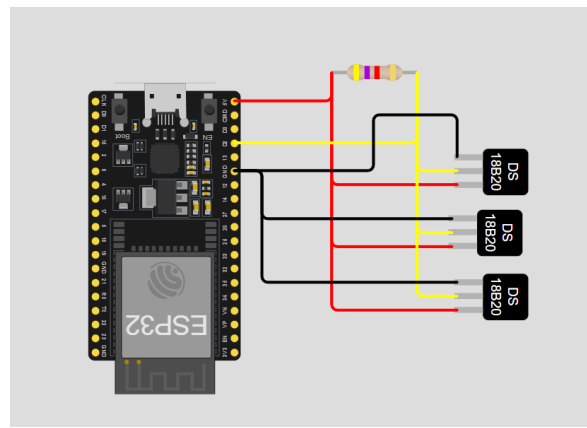


Figura 24: Sensor de temperatura conectado al ESP32

Se utilizaron 3 sensores de temperatura ya que dentro del sistema de refrigeración se colocarán 2 dentro de la cámara de enfriamiento y uno más en la cámara del compresor para obtenerla gradiente de temperatura del sistema de refrigeración.

Como siguiente paso se programó en Arduino un código que permitiera conocer las direcciones de cada uno de estos sensores, esto con la finalidad de obtener los datos provenientes de estas direcciones.

```

1      #include <OneWire.h>
2  #include <DallasTemperature.h>
3  #define ONE_WIRE_BUS 2  // Pin del ESP32 donde conectaste los
    sensores DS18B20
4  OneWire oneWire(ONE_WIRE_BUS);
5  DallasTemperature sensors(&oneWire);
6
7  void setup() {
8      Serial.begin(115200);
9      sensors.begin();
10     Serial.println("Buscando sensores DS18B20...");
11     int numSensores = sensors.getDeviceCount();
12     Serial.print("Sensores encontrados: ");
13     Serial.println(numSensores);
14
15     DeviceAddress address;
16     for (int i = 0; i < numSensores; i++) {
17         if (sensors.getAddress(address, i)) {
18             Serial.print("Direccion del sensor ");
19             Serial.print(i + 1);
20             Serial.print(": ");
21             printAddress(address);
22             Serial.println();
23         } else {
24             Serial.print("No se pudo obtener la direccion del sensor ")
25             ;
26             Serial.println(i + 1);
27         }
28     }

```

Una vez calibrado el primer sensor de temperatura, se aplicó el mismo algoritmo para llevar a cabo la calibración de los otros dos sensores de temperatura restantes. Finalmente se concluyó el proceso de la calibración de los sensores de temperatura.

3.1.10 SD APDAPTER ESP32

El módulo no requiere una calibración previa. Para su operación, cuenta con 6 pines. Los cuales se muestran en la tabla 6 Vcc, GND, MISO, MOSI, SCK, y CS. Este modulo permite la lectura y la escritura de una MicroSD mediante el protocolo de comunicación SPI.

Tabla 6: Tabla de conexiones modulo SD Adapter

Pin SD	Conexión a ESP32
VCC	5V
GND	GND
CS (chip Select)	GPIO5
MOSI	GPIO23
SCK (Clock)	GPIO8
MISO	GPIO19

Es importante señalar que el módulo lector de tarjetas Micro SD utilizado únicamente es compatible con el sistema de archivos FAT32. Este formato es el estándar para este tipo de dispositivos embebidos debido a su amplia compatibilidad y baja sobrecarga de procesamiento. Dado que muchas tarjetas Micro SD vienen formateadas en otros sistemas (como exFAT o NTFS), fue necesario verificar y, en su caso, reformatear las tarjetas a FAT32 antes de su uso en el sistema. Para llevar a cabo esta tarea de formateo se empleó el software EaseUS Partition Master, una herramienta que permite gestionar particiones y cambiar el sistema de archivos de manera sencilla y confiable.

3.1.10.1 Código de prueba con BLYNK

Una vez que se obtuvo todos los sensores calibrados, la PCB final y el diseño final del contenedor CAD. se procedió a programar un código que sea capaz de incluir todos los sensores y hacer que funcionen de manera simultanea. Como objetivo principal es que mande todos los valores obtenidos por los sensores por WIFI, el microcontrolador ESP32 tiene la capacidad de poder hacerlo, como también, se utilizó la plataforma de IoT "BLYNK" (figura 25) que tiene como función conectar dispositivos hardware como lo es Arduino, ESP32, Raspberry pi y otros a la nube para controlarlos y supervisarlos de forma remota mediante aplicaciones móviles (iOS y Android).

Se creó un panel o "dashboard" en el cual visualiza todos los valores obtenidos por los sensores en "widgets". Estos permiten darle formato, color y estilo al dashboard. la siguiente figura muestra como se diseñó el dashboard en la aplicación móvil en iOS,



Figura 25: Dashboard en Blynk

Blynk genera un "Aut Token" (token de autenticación), una cadena de caracteres única que funciona para permitir que solo ese dispositivo específico se conecte y envíe datos a su panel. Este token se incorporó dentro del firmware del microcontrolador.

```
1 // --- Blynk Configuración ---
2 #define BLYNK_TEMPLATE_ID "TMPL2nP4-KCfX"
3 #define BLYNK_TEMPLATE_NAME "PM IoT Refrigeration system"
4 #define BLYNK_AUTH_TOKEN "XXXXXXXXXXXXXX"
```

Para la conectividad de la red, el código implementa 2 librerías claves las cuales son "WiFi.h" la cual gestiona el protocolo de comunicación inalámbrica, y por otro lado la librería "Wi-FiManager.h" agrega una capa de gestión dinámica. Esta última genera un punto de acceso temporal (AP) al iniciar el dispositivo, permitiendo al usuario seleccionar manualmente la red WiFi disponible y almacenar las credenciales en la memoria (EEPROM) del ESP32.

```
1 //--Librerías--
2 #include <Wi-FiManager.h>
3 #include <ZMPT101B.h>
4 #include <OneWire.h>
5 #include <DallasTemperature.h>
6 #include <WiFi.h>
7 #include <BlynkSimpleEsp32.h>
8 #include <Wire.h>
9 #include <math.h>
10 #include <SD.h>
11 #include <SPI.h>
```

Para consolidar el sistema, se integró de manera modular el código previamente desarrollado para la calibración y lectura de los sensores (MPU6050). Se realizaron las modificaciones necesarias en la estructura del bucle principal y en la gestión de los datos para que la adquisición de vibraciones, el procesamiento local y el envío de datos a la nube Blynk funcionaran de manera coordinada y sin interferencias.

La siguiente etapa del algoritmo fue recopilar todos los datos obtenidos y ejecutar operaciones

matemáticas, funciones y el envío de datos hacia Blynk.

```
1 void enviarDatos() {
2   float v = readVoltage();           //Lee voltaje
3   float c = obtener_corriente_rms(); //Lee corriente RMS
4   float t1, t2, t3;
5   readTemperatures(t1, t2, t3); //Lee las 3 temperaturas
6   float vibracion = leerVibracionMPU6050(); // Lee nivel de
   vibracion
7   float DftE= ((t1+t2)/2) - t3;      //Calcula diferencial de
   temperatura
8   float Pot = v* c; // Calcula potencia (V*I)
9   float Gradiente= t2 - t1; //Calcula gradiente termico
```

Con los datos obtenidos se enviaron por Blynk con la siguiente instrucción:

```
1 // Enviar datos a Blynk
2 Blynk.virtualWrite(V0, v);           // Envia voltaje
3 Blynk.virtualWrite(V1, c);           // Envia corriente
4 Blynk.virtualWrite(V2, t1);          // Envia temp 1
5 Blynk.virtualWrite(V3, t2);          // Envia temp 2
6 Blynk.virtualWrite(V4, t3);          // Envia temp 3
7 Blynk.virtualWrite(V5, vibracion);   // Envia vibracion
8 Blynk.virtualWrite(V6, DftE);        // Envia diferencial
9 Blynk.virtualWrite(V7, Pot);         // Envia potencia
10 Blynk.virtualWrite(V8, COP_estimado); // Envia COP
11 Blynk.virtualWrite(V9, Gradiente);   // Envia gradiente
```

Cabe mencionar que el código manda cada 2 segundos una cadena de caracteres en la cual contiene toda la información recopilada por los sensores por lo que para evitar fallos o saturación de información se programó la siguiente instrucción

```

1  void loop() {
2  Blynk.run();           // Ejecuta Blynk
3  timer.run();           // Ejecuta
   temporizador
4  }

```

Finalmente el código permitió enviar información por medio de IoT lo que asegura el funcionamiento de este mismo como también la conectividad a internet.

3.2 Especificaciones del congelador monitoreado

El congelador CHP-68 de Ojeda fue el que se utilizó para realizar la pruebas necesarias para probar el sistema. La tabla 7 muestra las especificaciones técnicas.

Tabla 7: Especificaciones CHP-68

Parámetro	Valor
Marca / Modelo	Ojeda CHP-68
Capacidad	4 pies cúbicos (113 litros)
Refrigerante	R-134a
Voltaje nominal	127V / 60Hz
Corriente Nominal	2.0 A
Consumo energético	2.29kWh/ día
Temperatura de conservación	-18°C
Evaporador	Paredes frías

3.2.1 Instalación de sensores dentro del congelador

Para la instalación de los sensores fue con apoyo de un técnico profesional con la finalidad de no dañar ninguna tubería identificar la mejor ruta para los sensores de temperatura y vibración. Con ayuda de herramientas de taladradoras se perforó la cámara del congelador sin dañar ninguna tubería. La figura 26 muestra el proceso a seguir para la instalación de los sensores de temperatura. Una vez realizadas las perforaciones, se introdujeron los sensores de temperatura a través de ellas. Se colocaron dos sensores en el interior de la cámara, en posiciones distintas, para obtener lecturas comparativas en dos zonas del espacio. Los sensores se fijaron en dos puntos opuestos de la cámara: uno en la parte inferior y otro en la superior. La prueba se realizó con el congelador vacío y la tapa cerrada durante 15 minutos, para no alterar las condiciones internas mientras se tomaban las lecturas. Al comparar con el termostato externo, la diferencia fue de $\pm 3^{\circ}\text{C}$, lo cual se consideró aceptable para una aplicación de monitoreo. Más interesante fue la diferencia entre los dos sensores: $\pm 5^{\circ}\text{C}$ entre la parte superior e inferior. El aire frío se acumula en la zona baja, por lo que era esperable. Pero verlo en los datos confirmó que un solo sensor no habría sido suficiente para representar el comportamiento térmico real de la cámara. Las figuras 26, 27, 28 y 29 muestran el proceso de instalación.



Figura 26: Perforación a la cámara



Figura 27: Sensores adentro de cámara



Figura 28: Termostato vs sensor



Figura 29: Sensor dentro de cámara

3.2.2 Conexión con AWS

Para poder conectar el microcontrolador a la nube se utilizó el servicio de AWS "AWS IoT Core" como broker MQTT central. La comunicación corre sobre TLS 1.2 la cual es una versión del protocolo de seguridad Transport Layer Security que se utiliza para cifrar y autenticar la comunicación entre clientes y servidores en Internet[36]. Con autenticación mutua por certificados, esto quiere decir que el broker solo acepta conexiones de dispositivos que presenten credenciales válidas emitidas por Amazon(ver figura 30), por lo que se descarta conexiones no autorizadas desde el origen.

Antes que nada se modificó en Arduino donde se le agrego las librerías de internet, para poder conectarse a una red , este paso es crucial ya que si queremos mandar datos a la nube primero se debe de conectar a internet. Después se prosiguió con acceder a la consola de AWS IoT Core en la región US East (Ohio) y registrar el dispositivo dentro de la plataforma

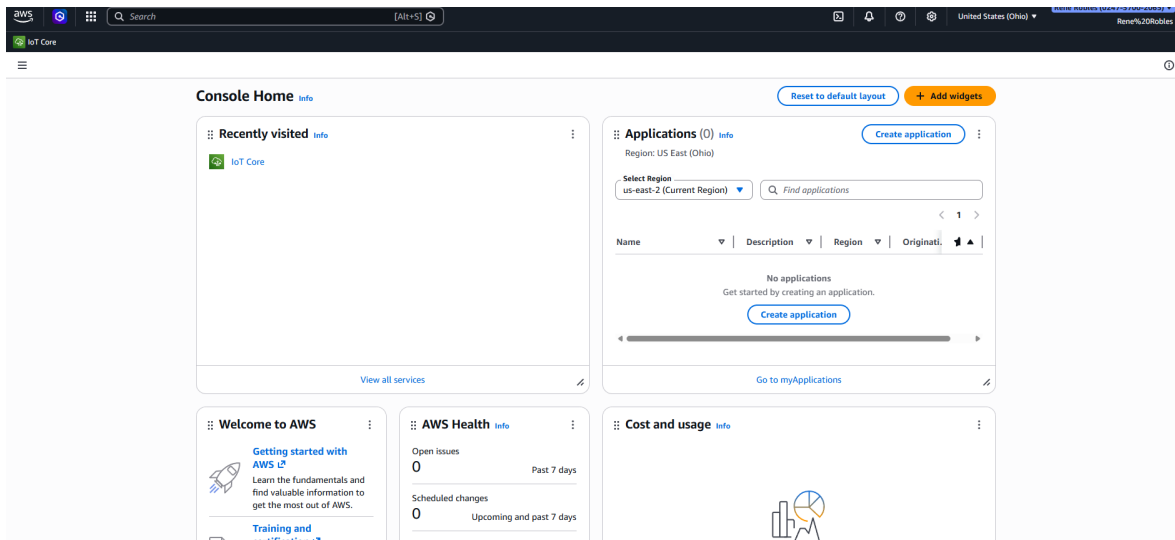


Figura 30: Consola Principal de AWS IoT Core Región US East (Ohio)

Una vez dentro de la consola de AWS IoT Core se creó una "Thing" (Figura 31) con el nombre de `congelador_ito`. Cabe recalcar que en AWS una Thing es la representación digital del dispositivo físico dentro de la plataforma, es el indicador que vincula los certificados X.509, (certificados son un formato estándar de clave pública que asocia de forma segura una clave criptográfica con la identidad de un usuario, organización o dispositivo[37]) las políticas y los mensajes MQTT a un equipo en específico

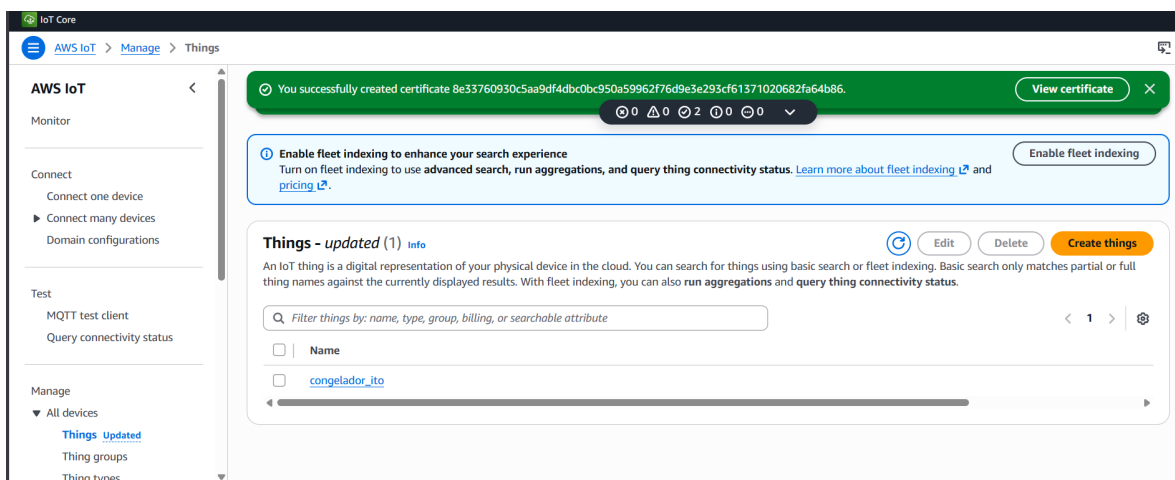


Figura 31: Registro de dispositivo congelador_ito en AWS IoT Core

Al crear la Thing, AWS generó automáticamente el certificado X.509 del dispositivo junto con

su llave privada (Figura 32). Esta pantalla aparece una sola vez, por lo que si no se descargan los archivos en ese momento, la llave privada no puede recuperarse y hay que generar un certificado nuevo. Se descargaron tres archivos: el certificado del dispositivo, la llave privada y el Amazon Root CA 1.

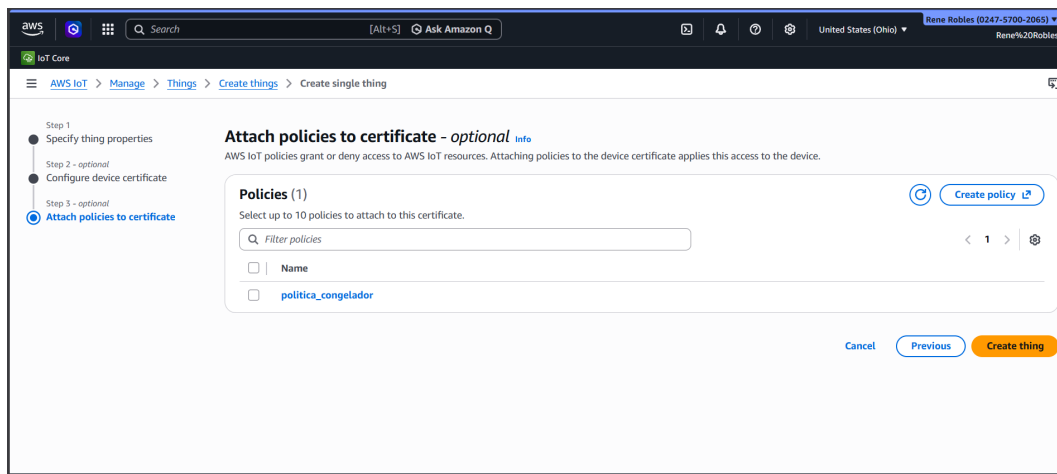


Figura 32: Asignación de la política al certificado del dispositivo

El último paso de la fase 1 fue vincular la política `politica_congelador` al certificado. Esta política (ver figura 33) define los permisos del dispositivo dentro de AWS: conectarse al broker, publicar mensajes, suscribirse a tópicos y recibirlos. Sin esta vinculación, el certificado existe pero el dispositivo no tiene autorización para hacer algo.

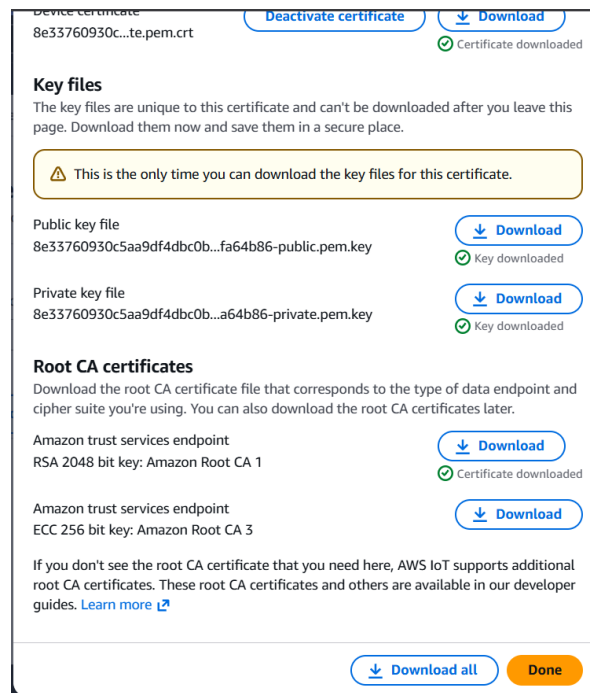


Figura 33: Descarga del certificado de dispositivo, llave privada y Amazon Root CA

Antes de integrar la conexión al firmware, se verificó que todo funcionara correctamente usando el cliente MQTT de prueba de AWS IoT Core. Se abrieron suscripciones a los dos topics del sistema, congeladores/ito/datos y congeladores/ito/alertas, y se confirmó la recepción de mensajes en tiempo real. Esta prueba descartó problemas de certificados o permisos antes de cargar el código definitivo al ESP32. (Ver Figura 34) Al poder corroborar el envío de datos se dio por concluido la fase 1 para el envío de datos por medio de AWS + ESP32, en la fase 2 consistió en la subsección de las librerías dentro del firmware y la estructura de los mensajes JSON.

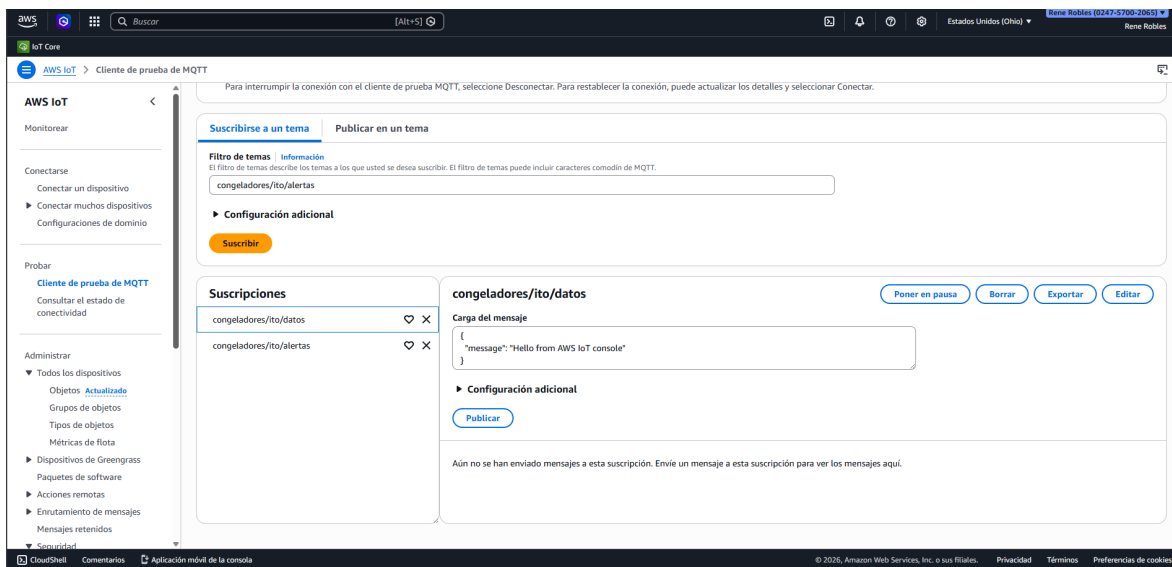


Figura 34: Verificación de la conexión MQTT mediante el cliente de prueba de AWS IoT Core

El ESP32 publica mensjaes en dos "topics" (Figura 35) distintos cada 30 segundos, (intervalo elegido de momento con finalidad de pruebas pero que puede ser modificado a nivel de firmware) para mantenerse dentro del nivel gratuito de AWS. El topic `congeladores/ito/datos` transporta las variables del sistema, y `congeladores/ito/alertas` se activa únicamente cuando el firmware detecta una condición de falla previamente preestablecida. La captura del cliente MQTT de prueba muestra un mensaje real recibido con fecha 17 de Abril de 2026 a las 15:02:50 UTC-6, con lecturas de voltaje corriente, temperatura en los tres sensores, vibración, potencia y COP.

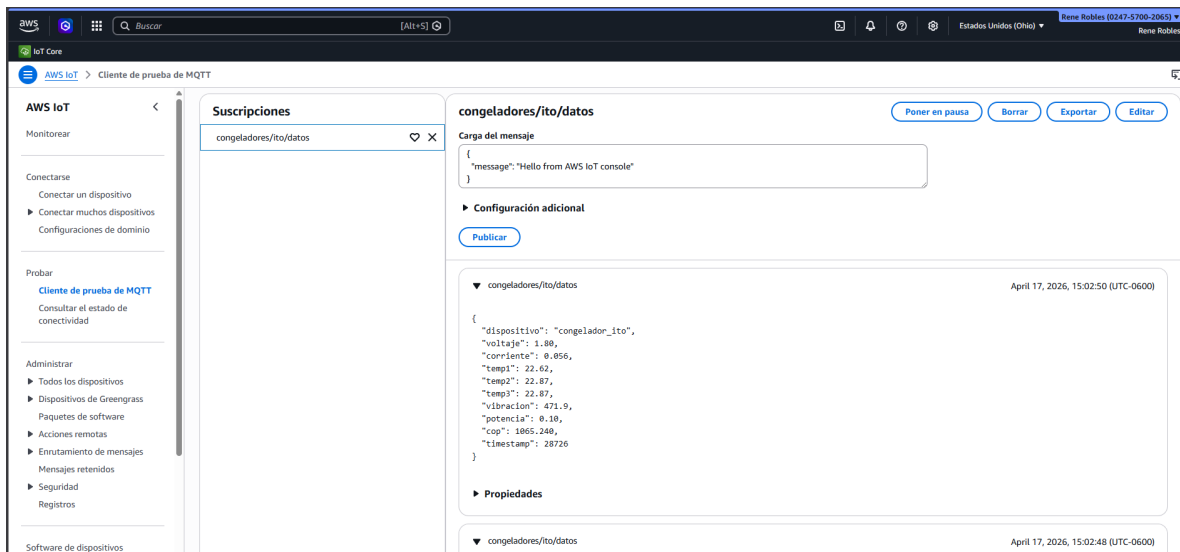


Figura 35: Cliente MQTT con mensaje JSON real

El campo `timestamp` corresponde a un Unix timestamp obtenido mediante sincronización NTP (La sincronización NTP Network Time Protocol es el proceso mediante el cual los sistemas informáticos ajustan sus relojes internos para mantener una hora precisa y coherente con una referencia externa [38] con el servidor `pool.ntp.org`, configurado en UTC-6 para Orizaba, Veracruz. Esto garantiza que cada registro en DynamoDB tenga una marca de tiempo real e incremental, independientemente de los reinicios del ESP32. El topic de alertas sigue la misma estructura pero con tres campos: el tipo de alerta, un detalle textual de la condición detectada y el timestamp. Esto permite que Lambda procese el mensaje y genere un correo descriptivo sin necesidad de consultar registros adicionales.

3.2.3 Arquitectura de servicios AWS

La nube no es un solo servicio es una cadena de seis componentes que trabajan en secuencia. Cada mensaje que sale del ESP32 recorre esa cadena completa antes de aparecer en el dashboard de Grafana o en el correo del responsable del equipo.

3.2.3.1 Enrutamiento de mensajes

IoT Core no solo recibe los mensajes del ESP32, también decide qué hacer con ellos. Se configuraron dos reglas SQL: la primera toma cada mensaje del topic de datos y lo escribe directamente en DynamoDB; la segunda escucha el topic de alertas y dispara la función Lambda de notificaciones. Sin estas reglas, los mensajes llegarían al broker y no irían a ningún lado. Para que los mensajes del ESP32 lleguen a DynamoDB sin intervención manual, se configuró la regla `regla_guardar_congeladores` en IoT Core. La instrucción SQL que ejecuta es:

```
SELECT *, timestamp() as ts FROM 'congeladores/ito/datos'
```

Esta regla toma cada mensaje del topic de datos, le agrega un timestamp generado por AWS y lo escribe directamente en la tabla. La acción apunta a `monitoreo_congeladores` usando dispositivo como clave de partición y timestamp como clave de ordenación, los valores se extraen directamente del JSON que envía el ESP32. Las figuras 36, 37 y 38 muestran el procedimiento llevado a cabo.

Asociar acciones de reglas Información
Una acción dirige los datos a un servicio de AWS específico.

Instrucción SQL Atrás
`SELECT *, timestamp() as timestamp FROM 'congeladores/ito/datos'`

Acciones de reglas Información
Seleccione una o varias de las acciones que se deben producir cuando la regla anterior coincida con un mensaje entrante. Las acciones definen actividades adicionales que se producen cuando llegan mensajes, como su almacenamiento en una base de datos, la invocación de funciones de la nube o el envío de notificaciones. Puede agregar hasta 10 acciones.

Acción 1
▼ **DynamoDB**
Insertar un mensaje en una tabla de DynamoDB Eliminar

Nombre de la tabla Información
 Ver Crear una tabla de DynamoDB

Clave de partición
La clave de partición (también denominada clave hash) debe coincidir con la clave de partición de la tabla de DynamoDB que ha creado.

Tipo de clave de partición
El tipo de clave de partición (también denominada clave hash) puede ser STRING o NUMBER. El valor predeterminado es STRING.

Valor de clave de partición
El valor de clave de partición (también denominada clave hash) admite plantillas de sustitución que proporcionan datos en tiempo de ejecución.

Figura 36: Acción de la regla IoT

Nombre de la tabla **Información**

monitoreo_congeladores

Ver

Crear una tabla de DynamoDB

Clave de partición

La clave de partición (también denominada clave hash) debe coincidir con la clave de partición de la tabla de DynamoDB que ha creado.

dispositivo

Tipo de clave de partición

El tipo de clave de partición (también denominada clave hash) puede ser STRING o NUMBER. El valor predeterminado es STRING.

STRING

Valor de clave de partición

El valor de clave de partición (también denominada clave hash) admite plantillas de sustitución que proporcionan datos en tiempo de ejecución.

\${dispositivo}

Clave de ordenación: *opcional*

La clave de ordenación (también denominada clave de rango) debe coincidir con la clave de ordenación de la tabla de DynamoDB que ha creado.

timestamp

Tipo de clave de rango

El tipo de clave de ordenación (también denominada clave de rango) puede ser STRING o NUMBER. El valor predeterminado es STRING.

NUMBER

Valor de clave de rango

El valor de clave de ordenación (también denominada clave de rango) admite plantillas de sustitución que proporcionan datos en tiempo de ejecución.

\${timestamp}

Figura 37: Propiedades completas de la regla de enrutamiento hacia DynamoDB

Propiedades de la regla

Nombre

regla_guardar_congeladores

Descripción

-

Paso 2: Instrucción SQL

Editar

Instrucción SQL

Versión de SQL

2016-03-23

Consulta SQL

SELECT *, timestamp() as timestamp FROM 'congeladores/ito/datos'

Paso 3: Acciones de reglas

Editar

Acciones

DynamoDB

Insertar un mensaje en una tabla de DynamoDB

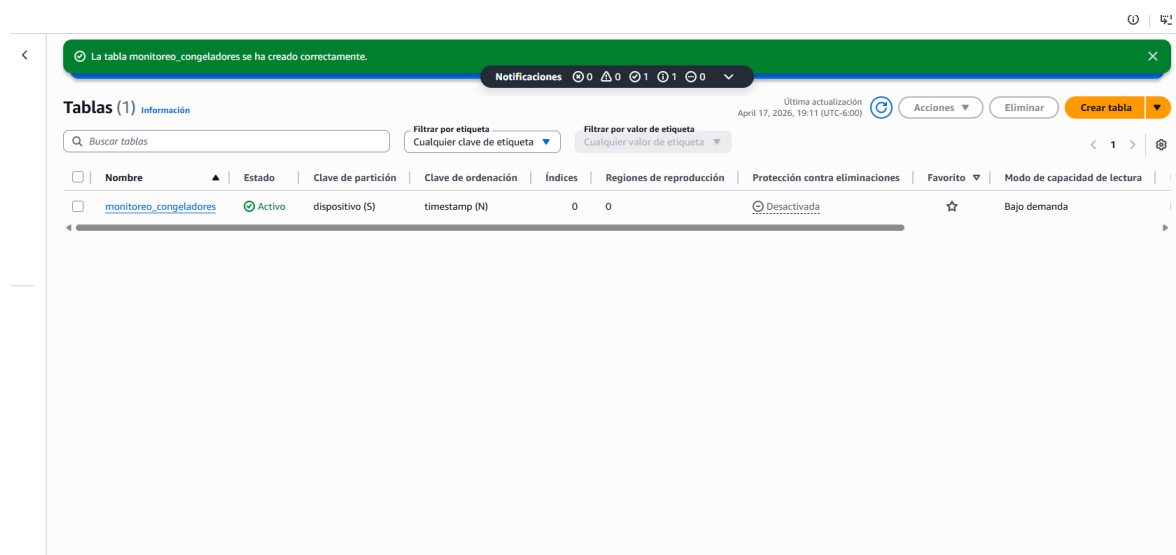
Nombre de la tabla	Clave de partición	Tipo de clave de partición
monitoreo_congeladores	dispositivo	STRING
Valor de clave de partición	Clave de ordenación: <i>opcional</i>	Tipo de clave de rango
\${dispositivo}	timestamp	NUMBER
Valor de clave de rango	Escritura de datos del mensaje en esta columna: <i>opcional</i>	Operación: <i>opcional</i>
\${timestamp}	-	-

Figura 38: Mapeo de claves de partición y ordenación desde el mensaje MQTT

73

3.2.3.2 DynamoDB

Los datos se almacenan en una tabla llamada `monitoreo_congeladores`. El diseño de la tabla tiene dos claves: el nombre del dispositivo como clave de partición, y el timestamp como clave de ordenación. Esta combinación permite consultar el historial completo de un congelador específico en un rango de tiempo determinado, sin escanear toda la tabla. El timestamp real, obtenido por NTP en el ESP32, es lo que hace útil esta estructura. Sin él, los registros llegarían desordenados cada vez que el microcontrolador se reinicia y las consultas por rango de tiempo no funcionarían.



La tabla `monitoreo_congeladores` se ha creado correctamente.

Notificaciones: 0 0 1 1 0 0

Última actualización: April 17, 2026, 19:11 (UTC-6:00)

Acciones: Eliminar, Crear tabla

Filtrar por etiqueta: Cualquier clave de etiqueta, Cualquier valor de etiqueta

Nombre	Estado	Clave de partición	Clave de ordenación	Índices	Regiones de reproducción	Protección contra eliminaciones	Favorito	Modo de capacidad de lectura
<code>monitoreo_congeladores</code>	Activo	dispositivo (S)	timestamp (N)	0	0	Desactivada	☆	Bajo demanda

Figura 39: Tabla `monitoreo_congeladores` activa en Amazon DynamoDB

La tabla `monitoreo_congeladores` se creó el 17 de abril de 2026 en modo de capacidad bajo demanda, lo que significa que AWS escala automáticamente sin necesidad de configurar capacidad fija. La tabla quedó activa desde el primer momento. (Ver Figura 39 y Figura 40) El diseño de claves es el que hace útil la tabla. La clave de partición es `dispositivo` de tipo `STRING`, y la clave de ordenación es `timestamp` de tipo `NUMBER`. Con esta combinación es posible consultar todos los registros de un congelador específico en cualquier rango de tiempo, sin necesidad de escanear registros de otros equipos.

Asociar acciones de reglas
[Información](#)

Una acción dirige los datos a un servicio de AWS específico.

Instrucción SQL

Atrás

SELECT *, timestamp() as timestamp FROM 'congeladores/ito/datos'

Acciones de reglas
[Información](#)

Seleccione una o varias de las acciones que se deben producir cuando la regla anterior coincida con un mensaje entrante. Las acciones definen actividades adicionales que se producen cuando llegan mensajes, como su almacenamiento en una base de datos, la invocación de funciones de la nube o el envío de notificaciones. Puede agregar hasta 10 acciones.

Acción 1

DynamoDB

Insertar un mensaje en una tabla de DynamoDB

Eliminar

Nombre de la tabla
[Información](#)

monitoreo_congeladores

Ver

Crear una tabla de DynamoDB

Clave de partición

La clave de partición (también denominada clave hash) debe coincidir con la clave de partición de la tabla de DynamoDB que ha creado.

\$(dispositivo)

Tipo de clave de partición

El tipo de clave de partición (también denominada clave hash) puede ser STRING o NUMBER. El valor predeterminado es STRING.

STRING

Valor de clave de partición

El valor de clave de partición (también denominada clave hash) admite plantillas de sustitución que proporcionan datos en tiempo de ejecución.

\$(timestamp)

Figura 40: Estructura de la tabla con clave de partición dispositivos y clave de ordenación timestamp

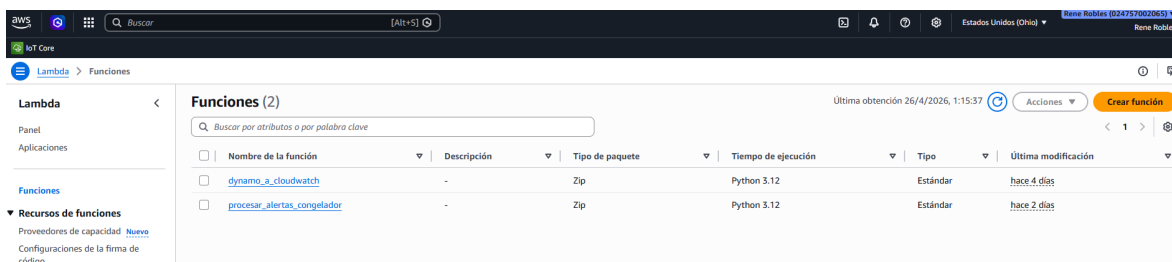
Una vez activa la regla de enrutamiento, los datos del ESP32 comenzaron a llegar a la tabla de forma automática ver figura 42. Al 26 de abril de 2026, la tabla monitoreo_congeladores contaba con 300 registros del dispositivo congelador_ito, cada uno con su timestamp incremental y el payload completo con las variables del sistema.

dispositivo (Key)	timestamp (Número)	payload
congelador_ito	30754	{ "potencia": { "W": "248.27", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "2...
congelador_ito	31206	{ "potencia": { "W": "7.55", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "2...
congelador_ito	31312	{ "potencia": { "W": "8.21", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "2...
congelador_ito	31323	{ "potencia": { "W": "260.21", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "489.9", "temp": { "W": "...
congelador_ito	60754	{ "potencia": { "W": "247.24", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "...
congelador_ito	61525	{ "potencia": { "W": "254.05", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "482.2", "temp": { "W": "...
congelador_ito	63228	{ "potencia": { "W": "7.96", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "2...
congelador_ito	90754	{ "potencia": { "W": "241.53", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "...
congelador_ito	91354	{ "potencia": { "W": "250.74", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "450.7", "temp": { "W": "...
congelador_ito	95206	{ "potencia": { "W": "8.14", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "2...
congelador_ito	120788	{ "potencia": { "W": "250.08", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "...
congelador_ito	123335	{ "potencia": { "W": "250.4", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "472.2", "temp": { "W": "...
congelador_ito	133208	{ "potencia": { "W": "7.99", "dispositivo": { "S": "congelador_ito", "vibracion": { "W": "1.3", "temp": { "W": "2...

Figura 41: Registros almacenados en la tabla monitoreo_congeladores con datos reales del congelador_ito

3.2.3.3 AWS Lambda

Se implementaron dos funciones en Python 3.12, ambas visibles en la consola de Lambda. La función `dynamo_a_cloudwatch` procesar alertas congelador lo que corresponde al orden en que se construyó el sistema: primero el pipeline de datos, luego el de alertas. como lo muestra la figura 42 muestra las funciones.



The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with 'Lambda' and 'Funciones' (Functions) selected. The main area displays a table of functions. Two functions are listed: 'dynamo_a_cloudwatch' and 'procesar_alertas_congelador'. Both are using Python 3.12 and are in the 'Estándar' (Standard) state. The last modification for 'dynamo_a_cloudwatch' was 4 days ago, and for 'procesar_alertas_congelador' it was 2 days ago.

Nombre de la función	Descripción	Tipo de paquete	Tiempo de ejecución	Tipo	Última modificación
dynamo_a_cloudwatch	-	Zip	Python 3.12	Estándar	hace 4 días
procesar_alertas_congelador	-	Zip	Python 3.12	Estándar	hace 2 días

Figura 42: Funciones Lambda desplegadas en AWS, región US East (Ohio)

La función `dynamo_a_cloudwatch` tiene un solo desencadenador: EventBridge como lo muestra la figura 43. Eso significa que no depende de mensajes entrantes, corre cada minuto por reloj, lee el registro más reciente de DynamoDB y escribe las métricas en CloudWatch.



Figura 43: Función dynamo_a_cloudwatch con desencadenador EventBridge

El código de la función consulta DynamoDB con `Limit=1` y `ScanIndexForward=False` esto devuelve directamente el registro más reciente sin recorrer toda la tabla (Ver figura 44). Luego extrae las ocho variables del payload y las escribe en CloudWatch mediante `cloudwatch.put_metric_data`. Si la tabla está vacía, la función termina limpiamente sin generar errores.



Figura 44: Código de la función dynamo_a_cloudwatch

Antes de integrar la función al sistema completo se realizaron pruebas directamente desde la consola de Lambda usando eventos JSON de prueba como muestra la figura 45. Esto permitió verificar que la función leía correctamente de DynamoDB y escribía las métricas en CloudWatch sin necesidad de esperar el trigger de EventBridge.

Probar evento Información CloudWatch Logs Live Tail Guardar Probar

Para invocar la función sin guardar un evento, configure el evento JSON, y luego elija Probar.

Acción de evento de prueba

☒ Crear un nuevo evento ☐ Editar evento guardado

Tipo de invocación

☒ Sincrónico
Ejecuta la función de Lambda y bloquea hasta recibir la respuesta de la función, con un tiempo de espera máximo de 15 minutos. Devuelve el resultado de la función o los detalles del error directamente a la aplicación que llama.

☐ Asincrónico
Pone en cola la función de Lambda para su ejecución y la devuelve inmediatamente con un ID de solicitud. La función se procesa de forma independiente y, si lo desea, los resultados se envían a un destino configurado, como SQS, SNS o EventBridge.

Nombre del evento

Máximo de 25 caracteres compuestos por letras, números, puntos, guiones y guiones bajos.

Configuración de uso compartido de eventos

☒ Privado
Este evento solo está disponible en la consola de Lambda y para el creador del evento. Puede configurar un total de 10. [Más información](#)

☐ Compartible
Este evento está disponible para los usuarios de IAM de la misma cuenta que tienen permisos para obtener acceso y utilizar eventos compartibles. [Más información](#)

Plantilla - Opcional

Evento JSON Formato JSON

1	{
2	{

Figura 45: Testeo a la funcion dynamo_a_cloudwatch

El permiso que permite a EventBridge invocar la función está definido en la política basada en recursos (Ver Figura 46). La entidad principal es `events.amazonaws.com` con la condición `ArnLike` esto restringe la invocación únicamente a la regla de EventBridge configurada, sin abrir el acceso a otros servicios.

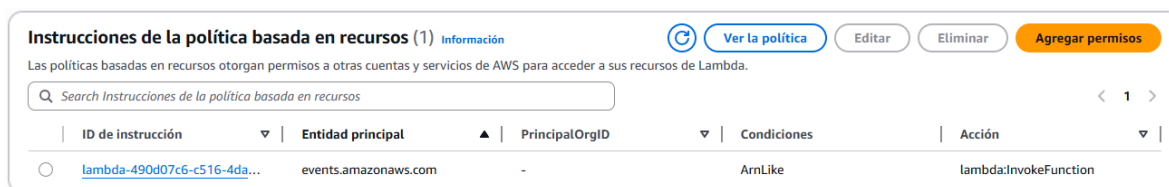


Figura 46: Permisos para EventBridge

3.2.3.4 Función procesar_alertas_congelador

A diferencia de `dynamo_a_cloudwatch`, esta función no corre por reloj, esta se activa únicamente cuando AWS IoT Core recibe un mensaje en el topic de alertas (Figura 47). El desencadenador es el propio servicio de IoT, lo que garantiza que cada falla detectada por el ESP32 genera una notificación sin demora.

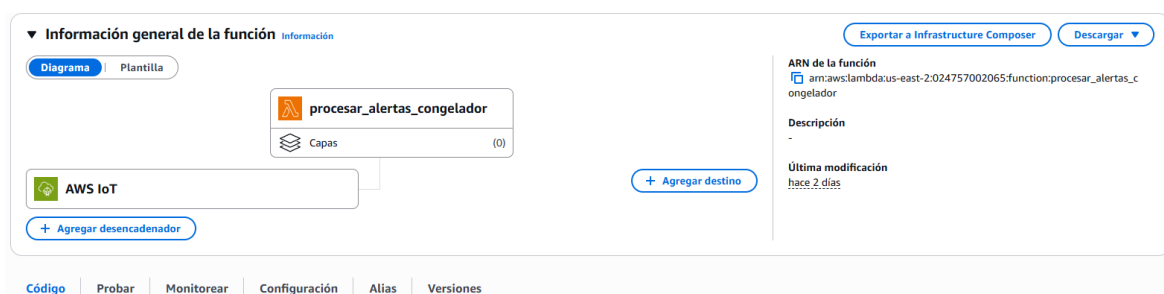


Figura 47: Función procesar_alertas_congelador con desencadenador AWS IoT Core

El código define dos diccionarios: uno con la descripción técnica de cada tipo de falla y otro con las acciones recomendadas específicas para esa condición. Cuando llega una alerta, la función identifica el tipo `CICLO_LARGO`, `GAS_LENTO`, `SOBREVOLTAJE`, entre otros, recupera el mensaje y las recomendaciones correspondientes, y los publica en SNS para el envío del correo (Figura 48). Cada tipo de falla tiene instrucciones distintas: una fuga lenta de gas lleva a contactar a un técnico, un sobrevoltaje lleva a verificar el regulador y reportar a CFE.

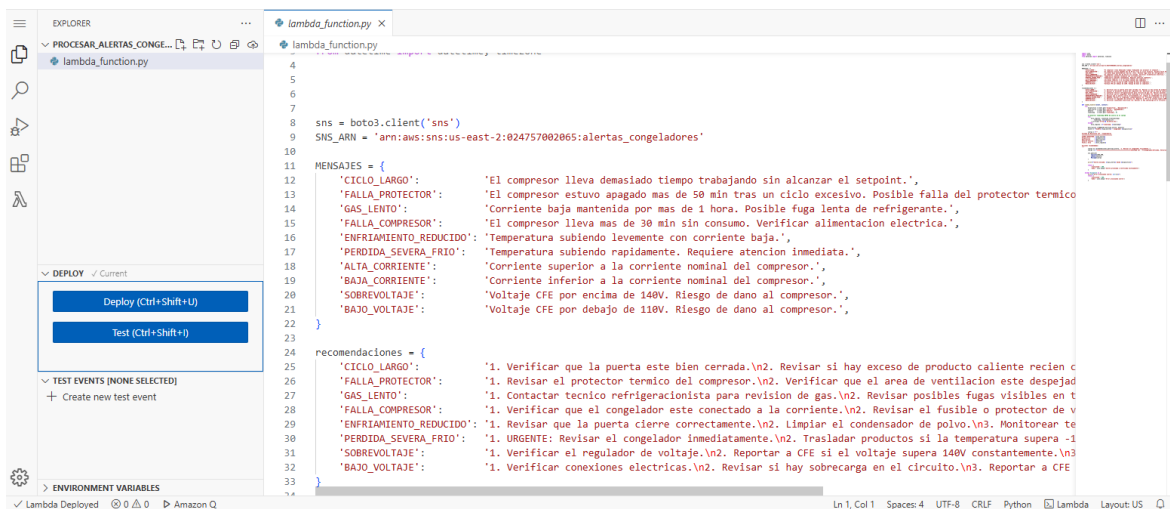


Figura 48: Código de la función

Al igual que con la otra función, se realizaron pruebas desde la consola de Lambda usando eventos JSON antes de activar el sistema completo como lo muestra la figura 49. Esto permitió verificar que los mensajes de alerta se formateaban correctamente y llegaban a SNS sin errores.

Probar evento Información

Para invocar la función sin guardar un evento, configure el evento JSON, y luego elija Probar.

Acción de evento de prueba

☒ Crear un nuevo evento ☐ Editar evento guardado

Tipo de invocación

☒ Sincrónico
Ejecuta la función de Lambda y bloquea hasta recibir la respuesta de la función, con un tiempo de espera máximo de 15 minutos. Devuelve el resultado de la función o los detalles del error directamente a la aplicación que llama.

☐ Asincrónico
Pone en cola la función de Lambda para su ejecución y la devuelve inmediatamente con un ID de solicitud. La función se procesa de forma independiente y, si lo desea, los resultados se envían a un destino configurado, como SQS, SNS o EventBridge.

Nombre del evento

MyEventName

Máximo de 25 caracteres compuestos por letras, números, puntos, guiones y guiones bajos.

Configuración de uso compartido de eventos

☒ Privado
Este evento solo está disponible en la consola de Lambda y para el creador del evento. Puede configurar un total de 10. [Más información](#)

☐ Compartible
Este evento está disponible para los usuarios de IAM de la misma cuenta que tienen permisos para obtener acceso y utilizar eventos compartibles. [Más información](#)

Plantilla - Opcional

Hello World

Evento JSON

```

1 {
2   "message": "Hello World"
3 }

```

[Formato JSON](#)

Figura 49: Configuración de evento de prueba

El permiso que autoriza a IoT Core a invocar esta función está registrado bajo la instrucción `regla_alertas_lambda`, con `iot.amazonaws.com` como entidad principal y condición ArnLike (Figura 50). Esto restringe la invocación a la regla específica configurada en IoT Core, sin abrir el acceso a otros servicios de AWS.

Resumen de recursos

Para ver los recursos y las acciones a los que la función tiene permiso de acceso, elija un servicio.

Amazon CloudWatch Logs
3 acciones 2 recursos

Por acción
Por recurso

Recurso	Acciones
arn:aws:logs:us-east-2:024757002065:*	Allow: logs:CreateLogGroup
arn:aws:logs:us-east-2:024757002065:log-group:/aws/lambda/procesar_alertas_congelador:*	Allow: logs:CreateLogStream Allow: logs:PutLogEvents

Lambda ha obtenido esta información de las siguientes declaraciones de política:

- Política administrada AWSLambdaBasicExecutionRole-6836a16a-2007-438a-823d-385ecc121ff9, declaración 0
- Política administrada AWSLambdaBasicExecutionRole-6836a16a-2007-438a-823d-385ecc121ff9, declaración 1

Instrucciones de la política basada en recursos (1) [Información](#)

Las políticas basadas en recursos otorgan permisos a otras cuentas y servicios de AWS para acceder a sus recursos de Lambda.

< 1 >

ID de instrucción	Entidad principal	PrincipalOrgID	Condiciones	Acción
regla_alertas_lambda	iot.amazonaws.com	-	ArnLike	lambda:InvokeFunction

Figura 50: Permiso de invocación otorgado a AWS IoT Core

El detalle de la instrucción confirma la configuración: servicio AWS IoT, entidad principal `iot.amazonaws.com`, ARN de origen apuntando específicamente a la regla `regla_alertas_lambda`, y acción `lambda:InvokeFunction` como se puede mostrar en la figura 51. Este nivel de especificidad evita que cualquier otra regla de IoT Core (presentes o futuras) pueda invocar esta función sin autorización.

Editar política basada en recursos

Editar instrucción de política

☐ Cuenta de AWS
Conceda permisos a otra cuenta, usuario o rol de AWS.
 ☒ Servicio de AWS
Conceda permisos a otro servicio de AWS.
 ☐ URL de la función
Conceda permisos para invocar la función de a través de la URL de la función.

Servicio
El servicio de AWS al que se conceden permisos.

AWS IoT

ID de instrucción
Ingrese un ID de instrucción único para diferenciar esta instrucción dentro de la política.

regla_alertas_lambda

Entidad principal
La entidad principal del servicio de este servicio de AWS. [Más información](#)

iot.amazonaws.com

ARN de origen
El ARN de un recurso. Busque el ARN en la consola de servicio relacionada.

arn:aws:iot:us-east-2:024757002065:rule/regla_alertas_lambda

Acción
Elija la acción que desea permitir.

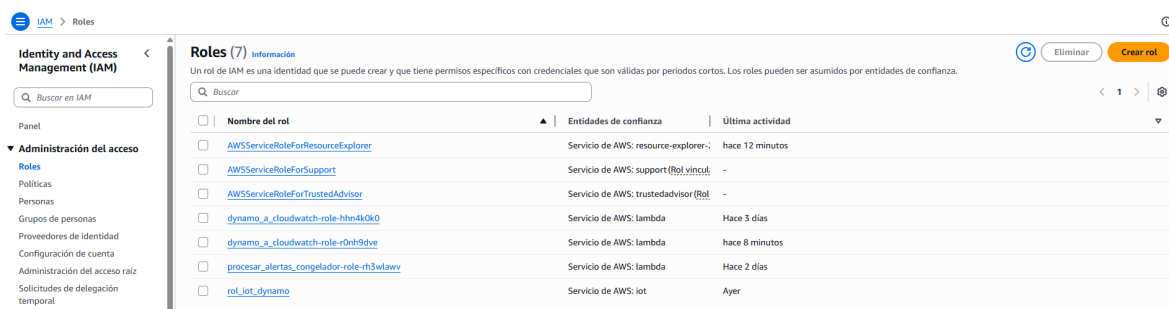
lambda:InvokeFunction

Cancelar
Guardar

Figura 51: Configuración detallada del permiso de invocación de IoT Core

3.2.4 IAM

IAM (Identity and Access Management) es el servicio de AWS que gestiona los permisos entre servicios y usuarios dentro de una cuenta. En lugar de dar acceso total a cada componente, IAM permite definir exactamente qué puede hacer cada servicio: leer una tabla, escribir una métrica, invocar una función [39]. La consola de IAM muestra los roles creados para el proyecto (figura 52). Cada función Lambda tiene su propio rol de ejecución, y la regla de IoT Core que escribe en DynamoDB tiene un rol independiente `rol_iot_dynamo` que le otorga únicamente el permiso de insertar registros en la tabla. Esta separación de roles sigue el principio de mínimo privilegio: cada servicio puede hacer solo lo que necesita y nada más.



Nombre del rol	Entidades de confianza	Última actividad
AWSServiceRoleForResourceExplorer	Servicio de AWS: resource-explorer-	hace 12 minutos
AWSServiceRoleForSupport	Servicio de AWS: support (rol vincu...	-
AWSServiceRoleForTrustedAdvisor	Servicio de AWS: trustedadvisor (Rol	-
dynamo_a_cloudwatch-role-hhn4k0k0	Servicio de AWS: lambda	Hace 3 días
dynamo_a_cloudwatch-role-r0nh9dye	Servicio de AWS: lambda	hace 8 minutos
procesar_alertas_congelador-role-rh3wlaav	Servicio de AWS: lambda	Hace 2 días
rol_iot_dynamo	Servicio de AWS: iot	Ayer

Figura 52: Roles dentro de IAM

3.2.5 SNS

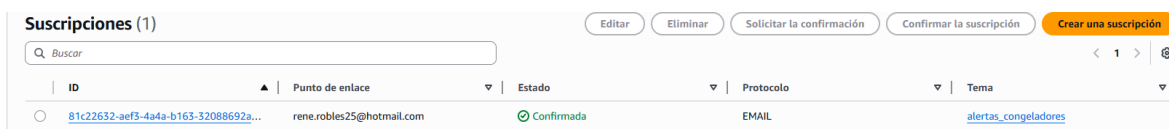
Amazon SNS (Simple Notification Service) es el servicio que se encarga de entregar las alertas al responsable del equipo [40]. Se creó un tema estándar llamado `alertas_congeladores` al que la función Lambda (Figura 53) publica cada vez que el ESP32 detecta una falla.



Nombre	Tipo	ARN
alertas_congeladores	Estándar	arn:aws:sns:us-east-2:024757002065:alertas_congeladores

Figura 53: Tema SNS `alertas_congeladores` creado en la región US East (Ohio)

El tema tiene una suscripción de tipo EMAIL (ver la figura 54) con estado confirmado, lo que significa que el destinatario aceptó recibir las notificaciones antes de que el sistema empezara a operar. Sin esa confirmación, SNS no entrega los mensajes.



Suscripciones (1)					
ID	Punto de enlace	Estado	Protocolo	Tema	
81c22632-ae3f-4a4a-b163-32088692a...	rene.robles25@hotmail.com	Confirmada	EMAIL	alertas_congeladores	

Figura 54: Suscripción de correo electrónico confirmada al tema alertas_congeladores

La figura 55 muestra un correo real generado por el sistema el 21 de abril de 2026. La alerta es de tipo CICLO_LARGO, disparada por el dispositivo congelador_ito cuando el compresor llevaba demasiado tiempo trabajando sin alcanzar el setpoint. El correo incluye la descripción técnica de la falla, el tiempo de sistema al momento del evento y tres acciones recomendadas específicas para esa condición: verificar que la puerta esté bien cerrada, revisar si hay producto caliente recién cargado y, si persiste, revisar el nivel de gas refrigerante. Todo esto se genera automáticamente sin intervención manual desde el momento en que el firmware detecta la condición hasta que el correo llega al destinatario.

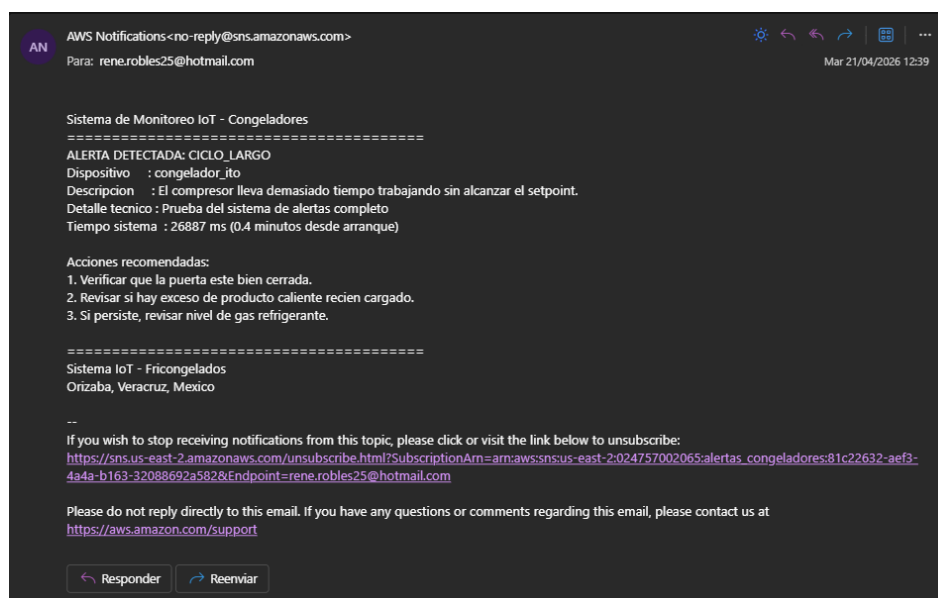


Figura 55: Ejemplo de envío de alerta vía EMAIL

3.2.6 EventBridge

Amazon EventBridge es un bus de eventos sin servidor y totalmente gestionado que facilita la creación de arquitecturas basadas en eventos, permitiendo que las aplicaciones se comuniquen entre sí mediante eventos casi en tiempo real.[41] EventBridge actúa como el reloj del sistema. Se configuró una regla programada llamada `cada_minuto_cloudwatch` (Figura 56) que corre de forma continua en el bus de eventos por defecto. Su estado es habilitado desde el momento en que se activó el sistema.

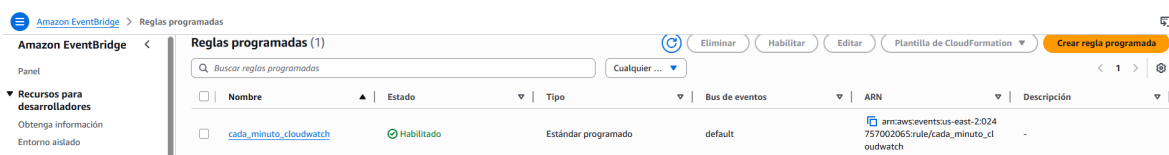


Figura 56: Funciones dentro de EventBridge

El detalle de la regla muestra los tres elementos que la definen: la programación fija de 1 minuto, el tipo de regla programada y el destino. Cada minuto, EventBridge invoca automáticamente la función Lambda `dynamo_a_cloudwatch`, que lee el registro más reciente de DynamoDB y escribe las métricas en CloudWatch como se muestra en la figura 57. No hay intervención manual mientras el sistema esté encendido, el ciclo corre solo.

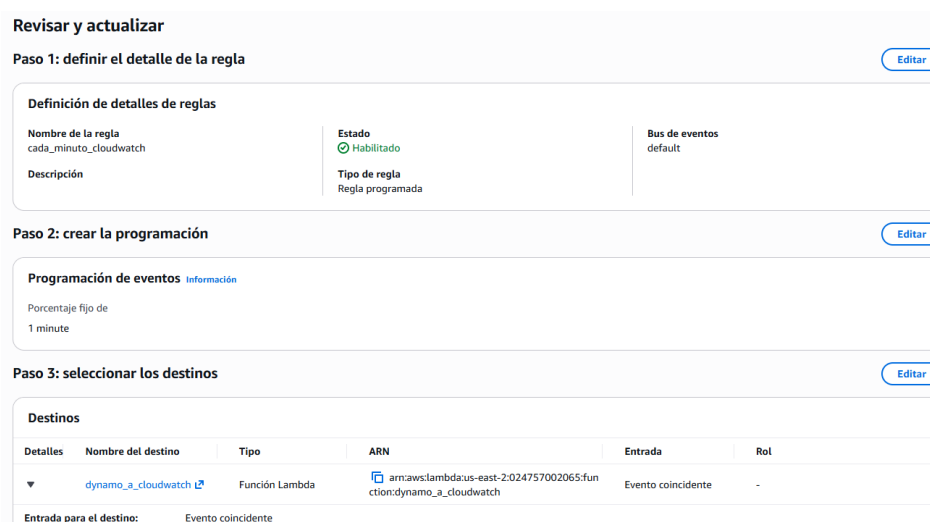


Figura 57: Regla para Eventbridge a función

3.2.7 CloudWatch

CloudWatch actúa como intermediario entre DynamoDB y Grafana Cloud. Cada minuto, la función Lambda `dynamo_a_cloudwatch` lee el registro más reciente de DynamoDB y escribe las ocho métricas en CloudWatch bajo el namespace `Congeladores/IoT` con la dimensión `Dispositivo=congelador_ito`. Esta arquitectura introduce un retraso máximo de dos minutos entre la medición real del sensor y su aparición en el dashboard, lo cual es aceptable para un sistema de monitoreo de este tipo. La dimensión `Dispositivo` además facilita escalar a múltiples congeladores sin modificar la estructura.

3.2.8 Grafana

La visualización del sistema se implementó en Grafana Cloud conectado a Amazon CloudWatch como fuente de datos. El dashboard agrupa seis paneles con datos en tiempo real del congelador, cubriendo variables térmicas, eléctricas y mecánicas. Cada panel actualiza su contenido cada uno o dos minutos conforme CloudWatch recibe nuevas métricas.

3.2.9 Creación de dashboard

Para la creación del dashboard donde se verán todos los paneles con las mediciones específicas, se deben de asociar por medio de cloudwatch a una variable en específico. La interfaz de Grafana incluye una opción para agregar paneles donde se le conectará una query para cada creación de estos mismos. La figura 58 muestra gráficamente como se visualiza un panel dentro de Grafana.

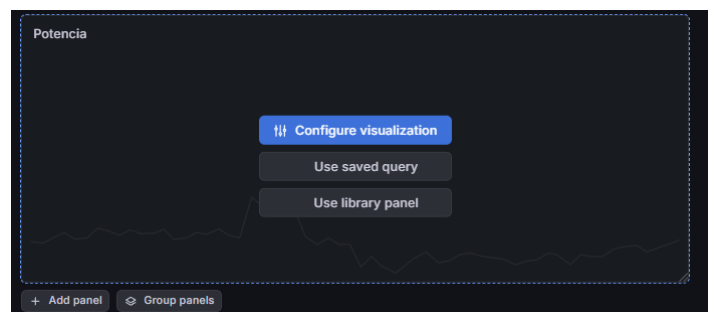


Figura 58: Agregar panel en Grafana

3.2.9.1 Configuración de panel

Para la configuración Grafana (Figura 59) permite cambiar el titulo del panel que en este caso se cambió por el titulo de "Potencia" ya que son estos los datos que se estarán mostrando.

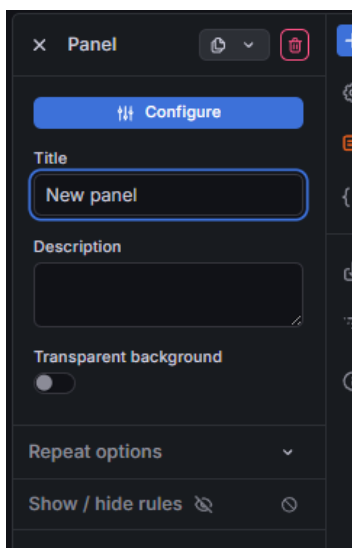


Figura 59: Cambio de nombre al Panel en Grafana

3.2.10 Data source en la query

Una vez creada el panel donde se estará visualizando la información, ahora es asignar la fuente de datos para poder visualizar. Al estar conectado con AWS por medio de Cloudwatch, se seleccionó "Cloudwatch" como lo muestra la figura 60. Como fuente de dato e identificador único (namespace) se buscó Congeladores_IoT, el nombre de la Métrica Potencia y el valor que se mostró fue un promedio de potencia por lo que en "Statistic" se seleccionó Average.

Por último la dimensión al seleccionar el objeto Congeladores_IoT habilita Dispositivo que esto sera igual al thing congelador_ito con la finalidad de que los datos los extraiga del JSON que previamente fue creado y el periodo fue de 60 esto con la razón de que el dashboard se actualiza cada 60 segundos (1 minuto).

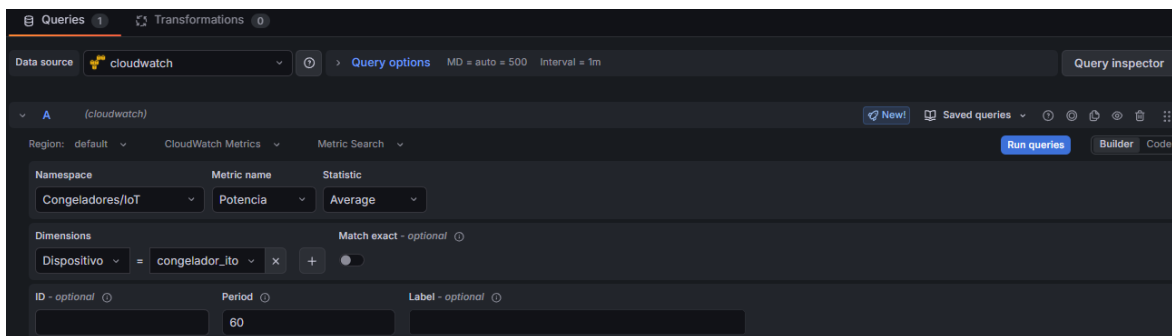


Figura 60: Configuración de Query

3.2.11 Diseño final de panel

Finalmente se seleccionó el tipo de panel solo fue cuestión de escoger el gráfico que mejor interpretara la potencia, se escogió el gráfico "Stats" ya que muestra el comportamiento durante varios ciclos. Figura 61. El mismo procedimiento se siguió para los paneles de voltaje, corriente, temperatura en la cámara, temperatura de salida y vibración.



Figura 61: Panel de potencia

CAPÍTULO 4: RESULTADOS

4.1 PCB Ensamblada

La figura 62 muestra la tarjeta ya ensamblada con todos los componentes soldados y montados. El ESP32 ocupa la parte central superior, a un costado izquierdo se encuentran conectores JST de tres pines los cuales son para los sensores de temperatura, colocados en esa posición para facilitar el manejo de cableado entre el sistema y el congelador. Justo debajo de los tres conectores JST de 3 pines se encuentra un conector mas tipo JST de 4 pines este es especial para el sensor MPU5060 (sensor para la vibración), también misma lógica se colocó en esa parte para facilitar el manejo de conexión con el congelador. En contra parte de la placa se encuentra ubicado la etapa de potencia junto con los sensores de corriente y voltaje. Y la fuente de alimentación Hi-Link HLK-PM01 que convierte los 127V de CA a 5V para alimentar el sistema.

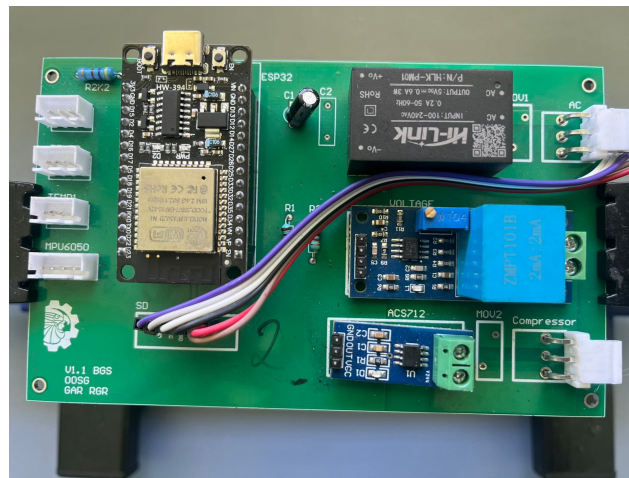


Figura 62: Ensamble de PCB con sensores y microcontrolador

4.1.1 Sistema de montaje durante pruebas

Durante el periodo de pruebas el sistema se instaló en el laboratorio junto al congelador Ojeda CHP-68. El supresor de picos se conectó entre la toma de corriente y el congelador para proteger el compresor de las variaciones de voltaje detectadas previamente. Los cables de los sensores de temperatura se introdujeron a través de las perforaciones realizadas en la cámara. Y la placa se colocó a un costado del congelador cerca de la cámara del compresor (figura 63) ya que el objetivo es que los cables no recorran tanto camino para llegar a los destinos designados para cada sensor.



Figura 63: Sistema embebido colocado junto a congelador CHP-68

4.1.2 PCB dentro de Case

El case fue diseñado en Fusion 360 e impreso en PLA. La figura 64 muestra la PCB alojada en su interior con el cableado de los sensores organizado a través de las entradas laterales. En la parte inferior del case se aprecia el módulo de tarjeta microSD conectado a la PCB. La tapa se dejó abierta durante las pruebas para facilitar el acceso a los conectores y al puerto USB del ESP32, necesario para monitorear el puerto serie y verificar el comportamiento del firmware en tiempo real.

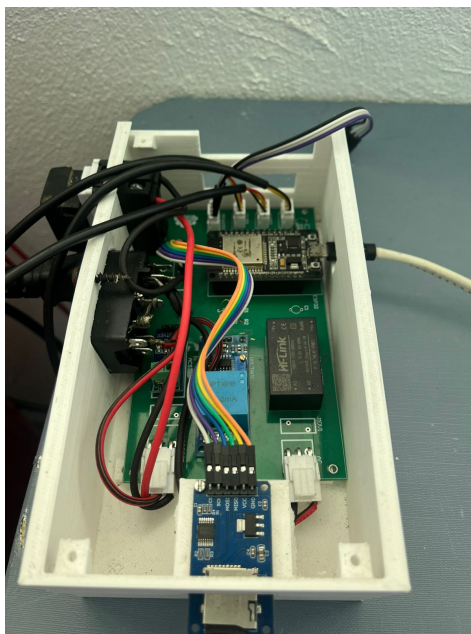


Figura 64: Case con sistema embebido integrado

4.1.3 Diseño final del case

El case (Figura 65) incluye una tapa la cual tiene ranuras de ventilación en la parte superior para disipar el calor generado por la fuente Hi-Link, y en la cara frontal se integraron directamente los conectores de CA: la entrada IEC con interruptor de encendido y la salida hacia el congelador. La base tiene tres entradas de cable con una abertura en la parte inferior para el paso ordenado del cableado de los sensores. El resultado es un enclosure funcional que concentra en una sola pieza la alimentación, el control y las conexiones del sistema.



Figura 65: Diseño Placa enclosure

4.1.4 Unidades fabricadas

Se fabricaron diez PCBs del mismo diseño. La figura 66 muestra cuatro placas sin ensamblar, identificadas con los números 0, 2, 3 y 4 que esta junto a la unidad 0 ya instalada dentro del case. Todas comparten el mismo diseño pero en esta etapa aún no tenían el ESP32 ni los módulos de sensores montados. Fabricar varias unidades desde el inicio permite tener placas de repuesto y facilita la escalabilidad del sistema a otros equipos de refrigeración sin necesidad de fabricar nuevamente.



Figura 66: Unidades armadas

4.2 Estructura de pruebas por Software

Al encender el ESP32, el firmware ejecuta automáticamente una captura de 10 segundos a 1 KHz antes de entrar al modo normal de operación. Esta función trabaja en conjunto del sensor ACS712 la cual registra la corriente en esos 10 primeros segundos. Las pruebas se realizaron en un congelador CHP-68 de la marca OJEDA. Cuando se conectó el congelador en total se registraron 9,995 muestras con un offset de 2.346V. La corriente máxima detectada fue de 3.267A, alcanzada a los 0.174 segundos del arranque. ese es el pico de inrush del compresor. La corriente RMS durante el arranque fue de 1.968A y la corriente promedio de 1.750A. El análisis quedó guardado (tabla 8) automáticamente en el archivo [análisis_arranque.txt](#) de la tarjeta microSD.

Tabla 8: Resultados del análisis de corriente de arranque

Parámetro	Valor
Muestras totales	9,995
Frecuencia de muestreo	1,000 Hz
Corriente máxima (inrush)	3.267 A
Corriente mínima	0.000 A
Corriente promedio	1.750 A
Corriente RMS	1.968 A
Tiempo del pico	0.174 s

4.2.1 Verificación de la transmisión de datos

La captura del monitor serie (Figura 67) muestra el arranque completo del sistema. A las 11:43:36 inició la captura de corriente de arranque; a las 11:43:47 completó las 9,995 muestras y guardó el análisis en la microSD. Inmediatamente después entró al modo normal de operación guardando datos en la tarjeta SD cada 2 segundos y detectó que el compresor estaba encendido con una temperatura de inicio de 17.88 °C.

```

11:43:36.666 -> === CAPTURA DE CORRIENTE DE ARRANQUE (10 seg) ===
11:43:37.158 -> Offset: 2.346 V
11:43:38.149 -> t=1 s | muestras: 999 | pico: 3.267 A
11:43:39.161 -> t=2 s | muestras: 1999 | pico: 3.267 A
11:43:40.161 -> t=3 s | muestras: 2999 | pico: 3.267 A
11:43:41.142 -> t=4 s | muestras: 3998 | pico: 3.267 A
11:43:42.143 -> t=5 s | muestras: 4998 | pico: 3.267 A
11:43:43.155 -> t=6 s | muestras: 5998 | pico: 3.267 A
11:43:44.157 -> t=7 s | muestras: 6997 | pico: 3.267 A
11:43:45.159 -> t=8 s | muestras: 7997 | pico: 3.267 A
11:43:46.162 -> t=9 s | muestras: 8997 | pico: 3.267 A
11:43:47.149 ->
11:43:47.149 -> Captura completa: 9995 muestras
11:43:47.186 -> Analisis guardado en analisis arranque.txt
11:43:47.186 ->
11:43:47.186 -> --- INICIANDO MODO NORMAL DE OPERACIÓN ---
11:43:47.186 -> Guardando datos en tarjeta SD cada 2 segundos
11:43:47.186 ->
11:43:48.977 -> >> Compresor ENCENDIDO. Temp inicio: 17.88 C
11:43:48.977 -> --- Iniciando nuevo ciclo de monitoreo (30 min) ---
11:43:49.024 -> Voltaje: 128.78 V | Corriente: 1.924 A
11:43:49.024 -> Temperaturas: 16.81, 18.94, 20.94
11:43:49.024 -> MQTT conectado: SI
11:43:49.024 -> WiFi estado: 3
11:43:50.989 -> Voltaje: 127.73 V | Corriente: 1.921 A
11:43:51.027 -> Temperaturas: 16.94, 18.94, 21.06
11:43:51.027 -> MQTT conectado: SI
11:43:51.027 -> WiFi estado: 3
11:43:53.007 -> Voltaje: 125.72 V | Corriente: 1.920 A
11:43:53.007 -> Temperaturas: 16.88, 18.94, 21.12
11:43:53.007 -> MQTT conectado: SI
11:43:53.007 -> WiFi estado: 3
11:43:55.025 -> Datos publicados a AWS
11:43:55.025 -> Voltaje: 127.76 V | Corriente: 1.919 A
11:43:55.025 -> Temperaturas: 16.88, 18.94, 21.19
11:43:55.025 -> MQTT conectado: SI
11:43:55.025 -> WiFi estado: 3
11:43:57.020 -> Voltaje: 127.64 V | Corriente: 1.913 A
11:43:57.020 -> Temperaturas: 16.88, 18.94, 21.25
11:43:57.020 -> MQTT conectado: SI

```

Figura 67: Monitor Serie

A partir de ese momento el sistema comenzó a publicar lecturas cada 2 segundos: voltaje entre 125V y 129V, corriente estable alrededor de 1.92A y tres temperaturas registradas simultáneamente. La línea "Datos publicados a AWS" confirma que el ciclo completo funcionó desde el primer momento. El ESP32 se conectó a WiFi, estableció la sesión MQTT con AWS IoT Core y publicó los datos sin intervención manual.

4.3 Resultados del sistema eléctrico

Durante 2.6 días de monitoreo continuo se registraron 110,422 muestras con envío cada 2 segundos, la tabla 9 muestra el resumen. La cual el voltaje promedio de CFE fue de 134.83V, valor que supera un poco el valor nominal de 127V. Entonces esto activo una alarma que el voltaje proveniente de CFE esta muy alto para la capacidad del compresor. El limite superior permitido por norma es de 140V. Este sobrevoltaje no es un evento aislado, aparece de forma recurrente en los datos y explica los picos de corriente de arranque observados.

Tabla 9: Estadísticas del sistema eléctrico

Parámetro	Valor
Duración del monitoreo	2.6 días (61.3 horas)
Total de registros	110,422 muestras
Voltaje promedio CFE	134.83 V
Voltaje mínimo	118.96 V
Voltaje máximo	152.37 V
Corriente promedio	1.215 A
Corriente máxima (arranque)	5.211 A
Corriente nominal (ficha técnica)	2.0 A

4.3.1 Análisis de ciclos del compresor

En 2.6 días se detectaron 101 ciclos completos. La duración promedio fue de 22.5 minutos y la mediana de 21.8 minutos, valores muy cercanos entre sí, lo que indica que el compresor opera con un patrón estable sin ciclos atípicos frecuentes. El 84.2 % de los ciclos duró entre 20 y 30 minutos, y ninguno superó los 30 minutos. Este análisis tuvo un gran impacto sobre todo a nivel de firmware porque permitió cambiar todos los tiempos de alertas ya que los tiempos que se establecieron al principios eran muy exigentes lo cual generaba muchas alertas falsas. Finalmente se ajustó a 35 minutos con base en los datos reales, las falsas alarmas desaparecieron sin comprometer la detección de fallas reales. La tabla 10 muestra el análisis del compresor.

Tabla 10: Análisis de ciclos del compresor

Parámetro	Valor
Ciclos detectados	101 ciclos
Duración mínima	8.8 min
Duración máxima	28.2 min
Duración promedio	22.5 min
Duración mediana	21.8 min
Ciclos menores a 20 min	16 (15.8 %)
Ciclos entre 20 y 30 min	85 (84.2 %)
Ciclos mayores a 30 min	0 (0 %)

4.3.2 Resultados de temperatura

La temperatura promedio (Tabla 11) de la cámara fue de -17.84°C , apenas por encima del setpoint que es de -18°C , esto no es un problema en cualquier sistema de refrigeración por ciclos la temperatura sube durante el reposo del compresor y baja durante el enfriamiento, por lo que el promedio siempre queda un poco más alto que el setpoint. la mínima de -20.31°C corresponde al fondo de los ciclos de enfriamiento y la máxima de -2.25°C fue porque la puerta se quedó abierta con la finalidad de obtener la lectura de cuando se da este caso.

Tabla 11: Estadísticas de temperatura de la cámara

Parámetro	Valor
Temperatura promedio	$-17.54^{\circ}C$
Temperatura mínima	$-20.31^{\circ}C$
Temperatura máxima	$-2.25^{\circ}C$
Setpoint del termostato	$-18^{\circ}C$

4.3.3 Resultados del COP

El COP promedio fue de 2.272, dentro del rango normal para un compresor hermético de 1/5 HP con R-134a. El máximo de 3.870 se da en arranques desde temperatura fría, cuando la diferencia entre evaporación y condensación es pequeña, el ciclo trabaja con menos esfuerzo y el rendimiento sube. El dato interesante es lo que podría mejorar: se estima que instalar un regulador de voltaje de 500 VA para corregir el sobrevoltaje de CFE llevaría el COP a un rango de 2.5 a 2.8 (tabla 12), una mejora directamente relacionada con el hallazgo eléctrico de la sección anterior.

Tabla 12: Análisis del coeficiente de rendimiento (COP)

Parámetro	Valor
COP promedio	2.272
COP máximo	3.870
COP mínimo	1.900
Rango normal de operación	1.8 – 3.5
COP estimado con regulador	2.5 – 2.8

4.3.4 Verificación del sistema de Alarmas

La prueba más directa del sistema fue esperar a que el congelador generara una condición real (Ver Figura 68). El 21 de abril de 2026 a las 12:39 ocurrió: el compresor llevaba demasiado tiempo trabajando sin bajar al setpoint y el firmware disparó la alerta CICLO_LARGO. Lo que siguió fue automático. El ESP32 publicó el mensaje en AWS, la regla de IoT Core lo capturó, Lambda procesó la alerta y SNS entregó el correo. Todo el recorrido desde que el firmware detectó la condición hasta que el correo llegó a la bandeja de entrada sin que nadie tocara nada, quiere decir que fue un proceso automático. El correo llegó con cuatro elementos concretos: el tipo de alerta y el dispositivo que la generó, la descripción técnica de lo que pasó, el tiempo de sistema al momento del evento y tres acciones específicas para esa falla como lo es: revisar que la puerta esté cerrada, verificar si hay producto caliente recién cargado y revisar el nivel de gas refrigerante. No es un mensaje igual para todas las alertas, cada tipo de falla tiene su propio conjunto de recomendaciones escritas en el código de Lambda.

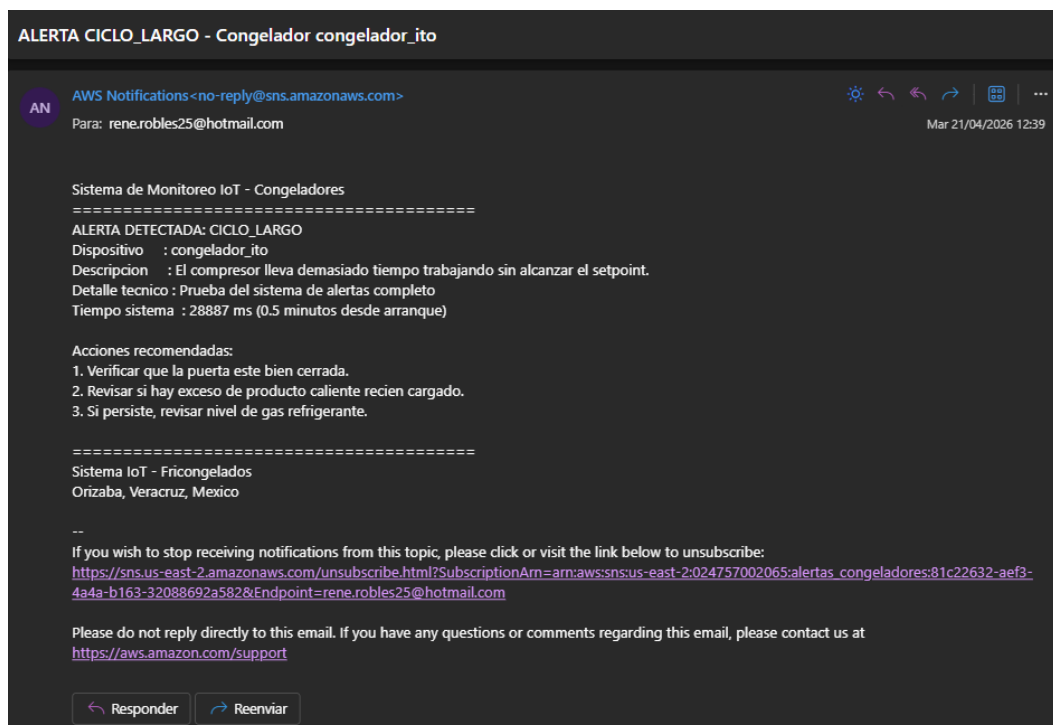


Figura 68: Correo de alerta generado automáticamente ante una condición de ciclo largo en el compresor

El mecanismo para que no se repitiera el mensaje y se llenara la bandeja de entrada con spam también respondió bien. Una vez enviada la alerta, el sistema no volvió a notificar mientras la condición seguía activa. Cuando el compresor alcanzó el setpoint y el ciclo terminó, la bandera se reinicia sola y esta lista para detectar el siguiente fallo.

Como también parte del firmware es detectar cuando algún sensor falla (Como muestra la figura 69), el ESP32 manda el mensaje a Lambda la cual activa una alerta por vía correo en la cual notifica que un sensor hay fallado y que se recomienda dar un chequeo visual al sensor de temperatura la figura muestra el ejemplo de la notificación.

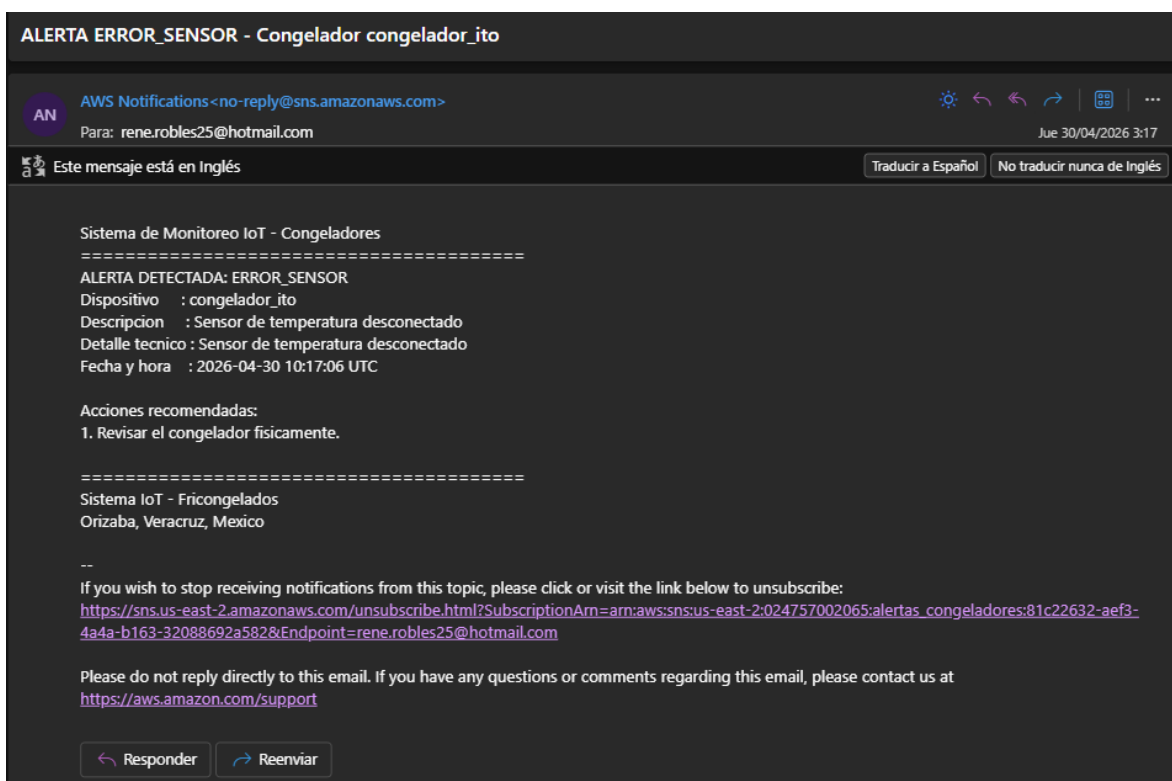


Figura 69: Alerta sensor fallando

4.4 Grafana

Los datos del congelador se visualizan en Grafana Cloud, conectado a CloudWatch como fuente de datos. El dashboard agrupa seis paneles que cubren las variables térmicas, eléctricas y mecánicas del equipo. Cada panel se actualiza solo, conforme CloudWatch recibe nuevas métricas cada minuto.

4.4.1 Panel 1

La gráfica de temperatura (figura 70) es la más legible del dashboard: muestra más de 20 horas de ciclos continuos entre -14°C y -18°C . Cada bajada es el compresor enfriando; cada subida, el compresor en reposo. La gráfica de corriente cuenta la misma historia en términos eléctricos: cae a 0A cuando el compresor se apaga y sube a 1.85A cuando arranca.

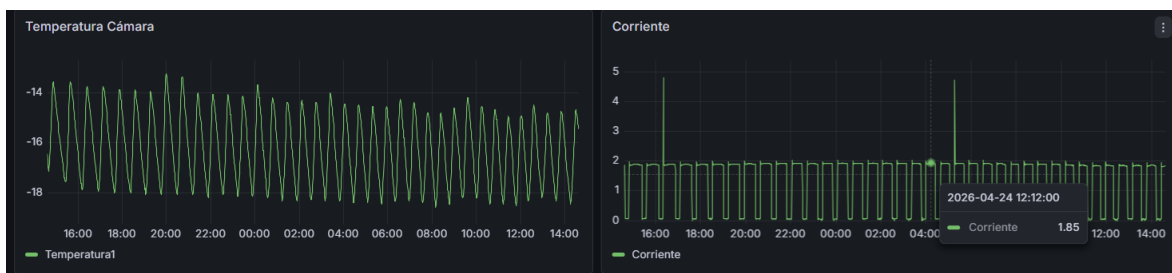


Figura 70: Panel 1 Corriente y Voltaje

4.4.2 Panel 2

La temperatura de descarga oscila entre 30°C y 50°C durante operación normal, con un pico inicial cerca de 50°C al arrancar desde temperatura ambiente (Figura 71). La línea cambia de azul a naranja-rojo según el valor, no hace falta leer los números para saber si el compresor está caliente. El gauge del COP marca 1.83 en esta captura, en zona roja. No es una falla: el compresor estaba arrancando desde temperatura alta y el COP mejora conforme la cámara se enfría.

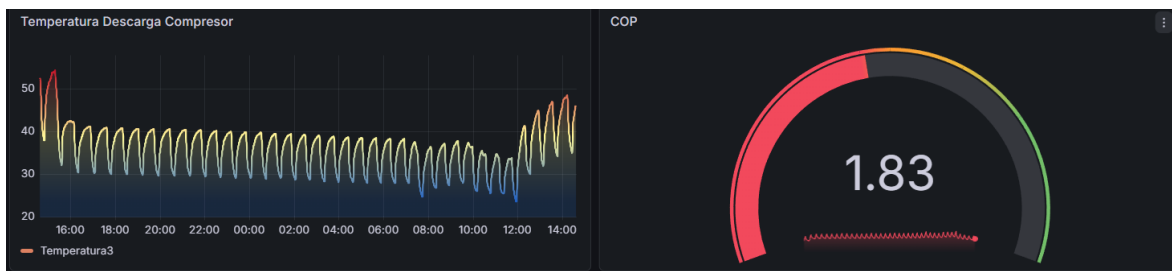


Figura 71: Temperatura de salida y COP

4.4.3 Panel 3

La vibración corre estable cerca de los 500 Hz durante todo el periodo registrado (Figura 72). El único evento que sale de ese rango es un pico que llega a 4 kHz y desaparece de inmediato, esto no indica una falla mecánica. El voltaje marca 128V en la captura, con variaciones visibles a lo largo del tiempo que reflejan el comportamiento normal del suministro de CFE en la zona.

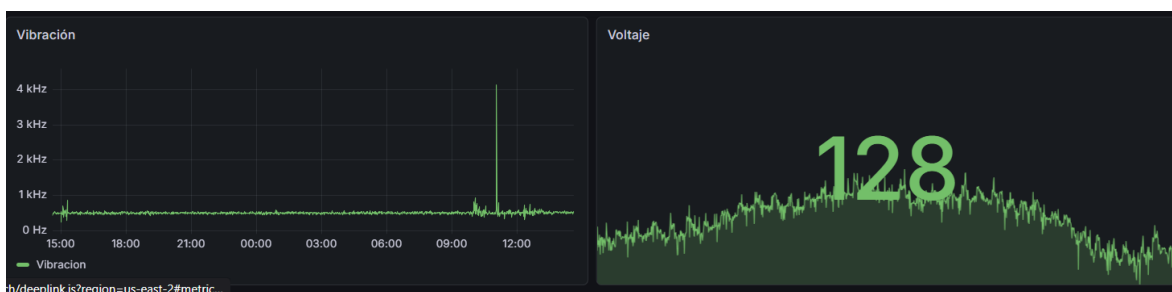


Figura 72: Voltaje

4.4.4 Panel 4

La potencia es directamente proporcional a la corriente y al voltaje ya que el consumo proviene de estas dos variable $P = V * I$. El panel 4 (figura 73) muestra el comportamiento de la potencia .



Figura 73: Panel de Potencia

CONCLUSIONES

En este trabajo se desarrolló y validó un sistema electrónico de bajo costo, basado en Internet de las Cosas, que monitorea de forma continua el voltaje, la corriente y la temperatura de congeladores comerciales. El sistema detecta automáticamente condiciones anómalas de operación y avisa de manera remota, con lo que se cumplió el objetivo general. Las conclusiones de cada objetivo específico se presentan a continuación.

Construir la placa de circuito impreso a partir de prototipos sucesivos funcionó bien. Cada versión dejó ver problemas concretos que se corrigieron en la siguiente: la dependencia de una alimentación externa, la fragilidad del cableado entre módulos y la vulnerabilidad ante los transitorios del arranque del compresor. La versión final reúne en una sola tarjeta el microcontrolador, los sensores, la fuente de alimentación y la protección contra sobretensiones. Además, se conecta a la misma toma que el equipo y opera de forma autónoma, así que puede instalarse en establecimientos que no cuentan con infraestructura especializada.

La calibración mostró que los sensores de bajo costo necesitan un ajuste individual para dar mediciones confiables. En el sensor de voltaje, bastó con ajustar su constante de calibración para que el error frente al instrumento de referencia quedara por debajo del 3 % en todo el intervalo evaluado. Esa precisión alcanza para detectar desviaciones del suministro respecto a su valor nominal. En el sensor de corriente fue indispensable eliminar el voltaje de desplazamiento en reposo; sin ese paso, las lecturas no correspondían a la carga aplicada.

En el firmware, dividir la operación en tres fases permitió tratar por separado fenómenos que ocurren en escalas de tiempo muy distintas. El transitorio de arranque del compresor dura apenas unos segundos y requiere muestrear a 1 kHz. Los ciclos de trabajo son del orden de decenas de minutos, y el comportamiento general del equipo se evalúa en ventanas de 30 minutos. Como los registros se guardan también en una tarjeta microSD, no se pierden aunque falle la conexión. Por su parte, el mecanismo de supresión de alertas repetidas evita que se acumulen notificaciones mientras una misma condición anómala continúa.

La infraestructura en la nube, montada sobre servicios de AWS, funcionó de extremo a extremo. La autenticación con certificados X.509 sobre TLS 1.2 asegura que solo el dispositivo

registrado pueda publicar datos. Al separar el flujo de mediciones del flujo de alertas, las notificaciones se procesan de inmediato, sin esperar al almacenamiento. Y el tablero en Grafana permite ver de un vistazo cómo está el equipo, sin tener que interpretar cifras.

El resultado central del trabajo es la validación en campo. En 61.3 horas de operación continua se registraron 110,422 muestras y 101 ciclos completos del compresor sin perder un solo dato. Los ciclos duraron en promedio 22.5 minutos y el más largo llegó a 28.2 minutos. Estos datos dejaron claro que el umbral inicial de 30 minutos para marcar un ciclo como prolongado estaba demasiado cerca del funcionamiento normal y provocaba falsas alarmas. Con base en ellos se subió a 35 minutos, y las falsas alarmas desaparecieron en el periodo analizado. La cadena de detección y notificación se probó con eventos reales: un ciclo prolongado del compresor, la desconexión de un sensor de temperatura y la puerta del equipo abierta durante un tiempo prolongado. En todos los casos, el responsable recibió la alerta por correo electrónico sin que nadie interviniera.

Por lo tanto, se acepta la hipótesis planteada. El sistema permitió caracterizar el comportamiento operativo del congelador y detectar automáticamente condiciones anómalas, notificándolas de forma remota y sin intervención humana. Una vez ajustados los umbrales, tampoco generó falsas alarmas durante la operación normal.

Uno de los hallazgos más importantes no estaba previsto. El monitoreo continuo del voltaje reveló que el equipo de prueba recibió de manera sostenida un suministro por encima de lo nominal: 134.8 V en promedio y picos de 152.4 V, cuando lo nominal son 127 V. Esos picos rebasan la tolerancia de $\pm 10\%$ establecida para el suministro. Una condición así aumenta el esfuerzo eléctrico y térmico sobre el compresor, y sin una medición permanente habría pasado desapercibida. Este resultado confirma que las variables eléctricas aportan información que la temperatura por sí sola no ofrece. También permiten identificar causas externas de desgaste del equipo, además de las fallas propias del ciclo de refrigeración.

El trabajo también tiene limitaciones. La validación se hizo en un solo equipo y durante 2.6 días, y los umbrales de diagnóstico se ajustaron con los mismos datos que sirvieron para evaluarlos. Por eso, para aplicarlos a otros equipos o a periodos más largos hace falta una

validación adicional. La potencia reportada es aparente, porque el sistema no mide el factor de potencia. El coeficiente de rendimiento se estimó a partir del ciclo de Carnot, así que no es una medición directa. La vibración sí se registró, pero con una frecuencia de muestreo demasiado baja para analizar su contenido espectral, y por eso quedó fuera de la lógica de diagnóstico. Ninguna de estas limitaciones impide cumplir los objetivos, pero sí marcan hacia dónde puede seguir el trabajo. La línea más clara es desarrollar modelos de predicción de fallas con la base de datos de operación que el sistema ya genera.

RECOMENDACIONES

A partir de los resultados obtenidos y de las limitaciones identificadas, se formulan las siguientes recomendaciones para la continuación del trabajo y para la aplicación práctica del sistema.

Ampliación de la validación experimental. Se recomienda extender el periodo de monitoreo a varias semanas e instalar el sistema en distintos equipos, de diferente capacidad, antigüedad y condiciones de uso, aprovechando las placas ya fabricadas. Esto permitiría determinar si los umbrales de diagnóstico ajustados en este trabajo son generalizables o si deben calibrarse para cada equipo. Asimismo, se recomienda evaluar el desempeño del sistema sobre un periodo de datos distinto al empleado para ajustar sus umbrales, y complementar la validación con fallas inducidas de forma controlada, como la obstrucción del condensador, la desconexión de sensores, la apertura prolongada de la puerta o la alimentación del equipo a voltajes fuera de norma mediante un variador. Con ello podría construirse una matriz de confusión del detector y cuantificar su tasa de falsas alarmas, de fallas no detectadas y su tiempo de respuesta, desde que ocurre el evento hasta que se recibe la notificación.

Desarrollo de modelos de predicción de fallas. La base de datos generada por el sistema constituye el punto de partida para incorporar técnicas de aprendizaje automático. En una primera etapa, se recomienda entrenar modelos de detección de anomalías no supervisados, como Isolation Forest, máquinas de vectores de soporte de una clase o autoencoders, sobre los registros de operación normal, y comparar su desempeño con el de la lógica de umbrales actual. A medida que el monitoreo prolongado permita registrar fallas reales y etiquetarlas, podrán desarrollarse modelos supervisados capaces de estimar la probabilidad de falla o el tiempo restante de operación del compresor, lo que haría posible pasar de la detección a la predicción. Una vez validados, estos modelos podrían ejecutarse directamente en el microcontrolador mediante técnicas de aprendizaje automático embebido, reduciendo la dependencia de la conectividad.

Mejora de la medición. Se recomienda incorporar la medición del factor de potencia mediante el muestreo síncrono de las señales de voltaje y corriente, lo que permitiría calcular la potencia

activa y el consumo real de energía del equipo, en lugar de la potencia aparente que se reporta actualmente. Para el sensor de corriente, conviene evaluar una versión de menor rango, como el ACS712 de 5 A, o un transformador de corriente, cuya resolución sea más adecuada para el consumo de un congelador de pequeña capacidad. Si se pretende monitorear equipos de mayor potencia, debe incorporarse una etapa de acondicionamiento que limite la señal al intervalo admisible por el convertidor analógico-digital del ESP32. En el caso de la vibración, se recomienda emplear el acelerómetro de la unidad inercial con una frecuencia de muestreo del orden de kilohertz y calcular su espectro de frecuencias en el propio microcontrolador, lo que permitiría identificar la firma mecánica del compresor e incorporarla al diagnóstico. Finalmente, se recomienda verificar los sensores de temperatura contra un termómetro patrón calibrado y caracterizar la no linealidad del convertidor del ESP32 en todo su intervalo de operación. Mejoras al firmware y a la infraestructura. Se recomienda implementar la actualización remota del firmware y la configuración remota de los umbrales de diagnóstico, por ejemplo mediante el servicio Device Shadow de AWS IoT, de modo que puedan ajustarse sin acceso físico al equipo. Para el despliegue en varios equipos, conviene automatizar el registro de dispositivos y la emisión de certificados, almacenar las credenciales en memoria cifrada y documentar el costo mensual de operación de la infraestructura en la nube en función del número de equipos monitoreados. Es igualmente recomendable integrar un temporizador de vigilancia que reinicie el sistema ante bloqueos del firmware y una alerta que notifique la pérdida prolongada de comunicación con el dispositivo.

Aplicación práctica y evaluación económica. Dado el sobrevoltaje sostenido identificado en el suministro, se recomienda a los usuarios de equipos de refrigeración en condiciones similares instalar un regulador de voltaje además del supresor de picos. Se sugiere, además, que su efecto se evalúe de manera experimental con el mismo sistema, comparando la corriente, la duración de los ciclos y el consumo de energía del equipo antes y después de su instalación. Por último, se recomienda realizar un análisis económico que considere el costo de fabricación del sistema, el costo de operación de la infraestructura y el valor del producto almacenado, con el fin de determinar el periodo de recuperación de la inversión y el tipo de establecimiento para el que su implementación resulta rentable.

REFERENCIAS BIBLIOGRÁFICAS

- [1] E. Zero, M. Sallak y R. Sacile, «Predictive Maintenance in IoT-Monitored Systems for Fault Prevention,» *Journal of Sensor and Actuator Networks*, vol. 13, 5 oct. de 2024, ISSN: 22242708. DOI: [10.3390/jsan13050057](https://doi.org/10.3390/jsan13050057).
- [2] M. Uppal, D. Gupta, S. Juneja et al., «Cloud-Based Fault Prediction for Real-Time Monitoring of Sensor Data in Hospital Environment Using Machine Learning,» *Sustainability (Switzerland)*, vol. 14, 18 sep. de 2022, ISSN: 20711050. DOI: [10.3390/su141811667](https://doi.org/10.3390/su141811667).
- [3] M. Abouelyazid, *Advanced Artificial Intelligence Techniques for Real-Time Predictive Maintenance in Industrial IoT Systems: A Comprehensive Analysis and Framework*, 2023.
- [4] H. Mudia, *Utilization of AI for Predictive Maintenance in IoT-Enabled Industrial Systems*, 2023. dirección: <https://creativecommons.org/licenses/by-sa/4.0/>.
- [5] C. M. C. Torres y J. F. V. Gómez, «Configuration of a temperature control in a low cost embedded system, using artificial intelligence and the internet of things tools,» *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, vol. 2019, págs. 68-84, 34 oct. de 2019, ISSN: 16469895. DOI: [10.17013/risti.34.68-84](https://doi.org/10.17013/risti.34.68-84).
- [6] H. M. Marhoon e I. A. Taha, «DESIGN AND IMPLEMENTATION-OF-INTELLIGENT-CIRCUIT-BREAKER-FOR-ELECTRICAL-CURRENT-SENSING-AND-MONITORING DESIGN AND IMPLEMENTATION OF INTELLIGENT CIRCUIT BREAKER FOR ELECTRICAL CURRENT SENSING AND MONITORING,» 2018, ISSN: 2348-9510. dirección: <https://www.researchgate.net/publication/324507273>.
- [7] P. S. Macheso y D. Thotho, «ESP32 Based Electric Energy Consumption Meter,» *International Journal of Computer Communication and Informatics*, vol. 4, págs. 23-35, 1 mayo de 2022. DOI: [10.34256/ijcci2213](https://doi.org/10.34256/ijcci2213).
- [8] Y. B. Zikria, R. Ali, M. K. Afzal y S. W. Kim, *Next-generation internet of things (IoT): Opportunities, challenges, and solutions*, feb. de 2021. DOI: [10.3390/s21041174](https://doi.org/10.3390/s21041174).

- [9] L. E. V. CANGA, *Sistema de Ahorro Energético para una vivienda Utilizando IoT en un Entorno Inteligente*, 2023.
- [10] F. Civerchia, S. Bocchino, C. Salvadori, E. Rossi, L. Maggiani y M. Petracca, «Industrial Internet of Things monitoring solution for advanced predictive maintenance applications,» *Journal of Industrial Information Integration*, vol. 7, págs. 4-12, sep. de 2017, ISSN: 2452414X. DOI: [10.1016/j.jii.2017.02.003](https://doi.org/10.1016/j.jii.2017.02.003).
- [11] R. José, C. Nebra y A. M. López, *Trabajo Fin de Grado Sistema de monitorización IoT de datos de una bomba de agua para su mantenimiento predictivo IoT data monitoring system of a water pump for predictive maintenance*, 2020.
- [12] M. Raparathi, S. B. Dodda y S. Maruthi, *Predictive Maintenance in IoT Devices using Time Series Analysis and Deep Learning*, 2023. dirección: <https://ballisticsjournal.com>.
- [13] G. G. Samatas, S. S. Moumgiakmas y G. A. Papakostas, *Predictive Maintenance- Bridging Artificial Intelligence and IoT*, 2021.
- [14] P. Killeen, B. Ding, I. Kiringa y T. Yeap, «IoT-based predictive maintenance for fleet management,» en *Procedia Computer Science*, vol. 151, Elsevier B.V., 2019, págs. 607-613. DOI: [10.1016/j.procs.2019.04.184](https://doi.org/10.1016/j.procs.2019.04.184).
- [15] J. Boubeta-Puig, J. J. Domínguez-Jiménez y J. Rosa-Bilbao, *Actas de las IV Jornadas de Investigación Predoctoral en Ingeniería Informática - Proceedings of the Fourth Doctoral Consortium in Computer Science (JIPII 2024)*. 2024, ISBN: 9788489867505.
- [16] V. Sharma y V. Mistry, «Machine Learning Algorithms for Predictive Maintenance in HVAC Systems,» *Journal of Scientific and Engineering Research*, vol. 2023, págs. 156-162, 11 2023, ISSN: 2394-2630. DOI: [10.5281/zenodo.11079979](https://doi.org/10.5281/zenodo.11079979). dirección: <https://www.researchgate.net/publication/381468614>.
- [17] M. Calabrese, M. Cimmino, F. Fiume et al., «SOPHIA: An event-based IoT and machine learning architecture for predictive maintenance in industry 4.0,» *Information (Switzerland)*, vol. 11, 4 abr. de 2020, ISSN: 20782489. DOI: [10.3390/INF011040202](https://doi.org/10.3390/INF011040202).

- [18] T. Akyaz y D. Engin, «Machine Learning-Based Predictive Maintenance System for Artificial Yarn Machines,» *IEEE Access*, vol. 12, págs. 125 446-125 461, 2024. DOI: [10.1109/ACCESS.2024.3454548](https://doi.org/10.1109/ACCESS.2024.3454548).
- [19] T. Akyaz y D. Engin, «Machine Learning-Based Predictive Maintenance System for Artificial Yarn Machines,» *IEEE Access*, vol. 12, 2024. DOI: [10.1109/ACCESS.2024.3454548](https://doi.org/10.1109/ACCESS.2024.3454548). dirección: <https://doi.org/10.1109/ACCESS.2024.3454548>.
- [20] J. M. Franco, *Manual de refrigeración*, Manual técnico en formato PDF, 2017. dirección: https://books.google.com.mx/books?id=F7-dmrTl0TAC&pg=PA55&lpg=PA55&dq=diametro+130&ots=HKT1MN0BQ_&sig=ACfU3U0xKhywHsaCNjCPCRCrcok001cuUTQ&hl=es&sa=X&ved=2ahUKEwiP5JHk75foAhUN16wKHcOMBp8Q6AEwEHoECAsQAQ#v=onepage&q=diametro%20130&f=false.
- [21] I. Amazon Web Services. «What is IoT (Internet of Things)?» Consultado: 5 Febrero 2026. (n.d.), dirección: <https://aws.amazon.com/what-is/iot/>.
- [22] Minew. «¿Qué es la arquitectura IoT? Una guía completa para las capas & componentes.» Accedido: 08 febrero 2026, Minew. (sep. de 2025), dirección: <https://www.minew.com/es/iot-architecture/>.
- [23] AODatacloud. «¿Qué es IoT y qué beneficios tiene?» Consultado: 07 de febrero de 2026, AODatacloud. (2023), dirección: <https://aodatacloud.es/blog/que-es-iot-y-que-beneficios-tiene/> (visitado 07-02-2026).
- [24] M. Gimenez. «Cloud Computing: qué es y para qué sirve.» Accedido: 08 febrero 2026, Hiberus. (feb. de 2023), dirección: <https://www.hiberus.com/crecemos-contigo/cloud-computing-que-es-y-para-que-sirve/>.
- [25] Google Cloud. «¿Qué es la computación en la nube?» Accedido: 08 febrero 2026, Google Cloud. (2025), dirección: <https://cloud.google.com/learn/what-is-cloud-computing?hl=es-419>.
- [26] Gartner. «Global Industrial IoT Platforms Reviews and Ratings.» (2025), dirección: <https://www.gartner.com/reviews/market/global-industrial-iot-platforms> (visitado 24-02-2026).

- [27] Gartner. «Global Industrial IoT Platforms Reviews and Ratings.» Gartner Peer Insights. (2025), dirección: <https://www.gartner.com/reviews/market/global-industrial-iot-platforms> (visitado 24-02-2026).
- [28] M. Gimenez. «Amazon Web Services (AWS): ¿qué es y qué ofrece?» Hiberus. (jul. de 2020), dirección: <https://www.hiberus.com/crecemos-contigo/amazon-web-services-aws-que-es-y-que-ofrece/> (visitado 24-02-2026).
- [29] Amazon Web Services. «¿Qué es AWS?» Accedido: 24-feb-2026, Amazon Web Services. (2024), dirección: <https://aws.amazon.com/es/what-is-aws/>.
- [30] Gartner. «IBM Watson IoT (Legacy) Reviews and Ratings.» Accedido: 26-feb-2026, Gartner. (2025), dirección: <https://www.gartner.com/reviews/product/ibm-watson-iot?marketSeoName=global-industrial-iot-platforms>.
- [31] ADLINK Technology. «AI at the Edge with Google Cloud.» Accedido: 26-feb-2026, ADLINK Technology. (2024), dirección: <https://www.adlinktech.com/en/google-cloud>.
- [32] NXP Semiconductors, *Google Cloud IoT Core*, Accessed: 2026-02-26, 2025. dirección: <https://www.nxp.com/design/design-center/software/i-mx-developer-resources/google-cloud-iot-core:GOOGLE-CLOUD-IOT-CORE>.
- [33] Oracle, *Oracle IoT Cloud Service Documentation*, Accessed: 2026-02-26, n.d. dirección: <https://docs.oracle.com/en/cloud/paas/iot-cloud/index.html>.
- [34] Unity Technologies, *¿Qué son los sistemas embebidos?* Accedido: 26-feb-2026, 2024. dirección: <https://unity.com/es/glossary/embedded-systems>.
- [35] «ESP32 Series Datasheet Version 5.2 2.4 GHz Wi-Fi + Bluetooth® + Bluetooth LE SoC Including,» inf. téc. dirección: www.espressif.com.
- [36] S. Ramallo. «Protocolo TLS 1.2: ¿Cómo funciona y por qué es esencial?» Accedido: 26-feb-2026, Prometeo. (ene. de 2025), dirección: <https://prometeoapi.com/blog/protocolo-tls-12>.

- [37] SSL.com Support Team. «What Is an X.509 Certificate?» Accedido: 26-feb-2026, SSL.com. (ago. de 2025), dirección: <https://www.ssl.com/faqs/what-is-an-x-509-certificate/>.
- [38] DeFelsko Corporation. «Sincronización automática de la hora NTP.» Accedido: 26-feb-2026, DeFelsko Corporation. (2025), dirección: <https://es.defelsko.com/resource-pages/automatic-ntp-time-synchronization>.
- [39] Amazon Web Services. «AWS Identity and Access Management (IAM).» Accedido: 26-feb-2026, Amazon Web Services. (2025), dirección: <https://aws.amazon.com/es/iam/>.
- [40] Amazon Web Services. «¿Qué es Amazon Simple Notification Service (Amazon SNS)?» Accedido: 26-feb-2026, Amazon Web Services. (2025), dirección: https://docs.aws.amazon.com/es_es/sns/latest/dg/welcome.html.
- [41] Amazon Web Services. «Amazon EventBridge.» Accedido: 26-feb-2026, Serverless Land. (2025), dirección: <https://serverlessland.com/eventbridge>.