Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería Electrónica y Eléctrica
Escuela Profesional de Ingeniería Electrónica

Diseño e implementación de un sistema inteligente de monitoreo con aprendizaje automático y arquitectura híbrida local-nube para gabinetes Tier 1

TESIS

Para optar el Título Profesional de Ingeniero Electrónico

AUTORES

Fernando Oren CARDOZA MARIN
Cristy Brigithe SAMAMÉ ROJAS

ASESOR

Mg. Ricardo YAURI RODRÍGUEZ

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Cardoza, F. & Samamé, C. (2026). *Diseño e implementación de un sistema inteligente de monitoreo con aprendizaje automático y arquitectura híbrida local-nube para gabinetes Tier 1*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería Electrónica y Eléctrica, Escuela Profesional de Ingeniería Electrónica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor 1	
Nombres y apellidos	Fernando Oren Cardoza Marin
Tipo de documento de identidad	DNI
Número de documento de identidad	73024697
URL de ORCID	https://orcid.org/0000-0002-7266-5197
Datos de autor 2	
Nombres y apellidos	Cristy Brigithe Samamé Rojas
Tipo de documento de identidad	DNI
Número de documento de identidad	71850838
URL de ORCID	https://orcid.org/0009-0001-7240-785X
Datos de asesor	
Nombres y apellidos	Ricardo Yauri Rodríguez
Tipo de documento de identidad	DNI
Número de documento de identidad	41900467
URL de ORCID	https://orcid.org/0000-0001-9884-9317
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Teresa Esther Nuñez Zuñiga
Tipo de documento	DNI
Número de documento de identidad	08006778
Miembro del jurado 1	

Nombres y apellidos	Jorge Elías Moscoso Sánchez
Tipo de documento	DNI
Número de documento de identidad	07206008
Miembro del jurado 2	
Nombres y apellidos	Juan Francisco Tisza Contreras
Tipo de documento	DNI
Número de documento de identidad	08596442
Datos de investigación	
Línea de investigación	Sistemas de monitoreo y gestión ambiental, Internet de las cosas IOT, Sistemas de alerta temprana y prevención de desastre
Grupo de investigación	No aplica.
Agencia de financiamiento	Sin financiamiento.
Ubicación geográfica de la investigación	Edificio: El Almirante, Oficina 1306 País: Perú Departamento: Lima Provincia: Lima Distrito: Lince Dirección: Av. Petit Thouars 1775 Latitud: -12.085 Longitud: -77.033
Año o rango de años en que se realizó la investigación	2025 – 2026
URL de disciplinas OCDE	Ingeniería eléctrica, Ingeniería electrónica https://purl.org/pe-repo/ocde/ford#2.02.01 Sistemas de automatización, Sistemas de control https://purl.org/pe-repo/ocde/ford#2.02.03

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, DECANA DE AMÉRICA)
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA
Teléfono 619-7000 Anexo 4226
Calle Germán Amezaga 375 – Lima 1 – Perú

ACTA DE SUSTENTACIÓN

TESIS

ACTA N°63/FIEE-PPP/2026

Los suscritos Miembros del Jurado, docentes permanentes de las Escuelas Profesionales de la Facultad de Ingeniería Electrónica y Eléctrica, reunidos en la fecha 03 de Septiembre del 2026, como presidente de Jurado la **DRA. TERESA ESTHER NUÑEZ ZUÑIGA,** integrado por el Miembro de Jurado el **DR. JORGE ELÍAS MOSCOSO SÁNCHEZ,** el Miembro de Jurado el **MG. JUAN FRANCISCO TISZA CONTRERAS** y Miembro Asesor el **MG. RICARDO YAURI RODRIGUEZ.**

Después de escuchar la Sustentación Presencial de la Tesis de la Bachiller **CRISTY BRIGITHE SAMAMÉ ROJAS** con código 20190071 que para optar el Título Profesional de Ingeniera Electrónica y del Bachiller **FERNANDO OREN CARDOZA MARIN** con código 20190177 que para optar el Título Profesional de Ingeniero Electrónico, sustentaron la Tesis titulada **DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTELIGENTE DE MONITOREO CON APRENDIZAJE AUTOMÁTICO Y ARQUITECTURA HÍBRIDA LOCAL-NUBE PARA GABINETES TIER 1.**

El jurado examinador procedió a formular las preguntas reglamentarias y, luego de una deliberación en privado, decidió otorgarle el calificativo de 16 (Dieciseis).

Ciudad Universitaria, 03 de Septiembre del 2026

DRA. TERESA ESTHER NUÑEZ ZUÑIGA
Presidente Jurado

DR. JORGE ELÍAS MOSCOSO SÁNCHEZ
Miembro Jurado

MG. JUAN FRANCISCO TISZA CONTRERAS
Miembro Jurado

MG. RICARDO YAURI RODRIGUEZ
Miembro Asesor

UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, **RICARDO YAURI RODRIGUEZ** en mi condición de asesor acreditado con el Dictamen N°65/FIEE-PPP/2026 de la Tesis titulada **DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INTELIGENTE DE MONITOREO CON APRENDIZAJE AUTOMÁTICO Y ARQUITECTURA HÍBRIDA LOCAL-NUBE PARA GABINETES TIER 1,** presentado por la bachiller CRISTY BRIGITHE SAMAMÉ ROJAS, para optar al título profesional de Ingeniera Electrónica y por el bachiller FERNANDO OREN CARDOZA MARIN, para optar al título profesional de Ingeniero Electrónico. CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 04% de similitud, nivel PERMITIDO para continuar con los trámites correspondientes y para su publicación en el repositorio institucional.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título de Ingeniera Electrónica y de Ingeniero Electrónico.

Lima, 03 de Septiembre del 2026

Firma del Asesor

DNI: 41900767

Nombres y apellidos del asesor:
RICARDO YAURI RODRIGUEZ

DEDICATORIA

A mi madre, Susana, por su amor incondicional, por ser mi mayor ejemplo de perseverancia y resiliencia, y por demostrarme que con esfuerzo siempre es posible alcanzar nuestros sueños.

A mi hermana, Anggie, por su compañía, por ayudarme a encontrar alegría y tranquilidad en este camino y por estar siempre a mi lado. Este logro también les pertenece.

Cristy Samamé

DEDICATORIA

A mis padres, Jaime y Magaly, por creer en mí incluso cuando yo dudé. Todo lo que soy y lo que logro lleva su nombre.

Fernando Cardoza

AGRADECIMIENTOS

A mi familia, en especial a mi tío Luis, por su generosidad y acogida durante mi formación profesional.

A Fernando, mi compañero, por su compromiso y dedicación, por compartir este proyecto conmigo y por la armonía que hemos construido juntos, demostrando que hacemos un gran equipo.

Al Mg. Ricardo Yauri Rodríguez, por su asesoría, dedicación y por brindarnos orientación durante el desarrollo de esta investigación.

Cristy Samamé

AGRADECIMIENTOS

Ante todo, a mi familia, por el apoyo brindado durante el desarrollo del trabajo de investigación.

A Cristy, mi compañera de tesis, de esfuerzo y de vida, gracias por cada momento dedicado a este trabajo. Sin ti, esto no hubiera sido lo mismo.

Asimismo, al Mg. Ricardo Yauri Rodríguez por sus consejos, apoyo y dedicación, esencial apoyo para poder culminar este trabajo.

Finalmente, a la empresa Cloud IT Peru S.A.C., por prestar sus instalaciones para lograr cumplir los objetivos de este trabajo de tesis.

Fernando Cardoza

INDICE GENERAL

INDICE GENERAL vi

LISTA DE FIGURAS xi

LISTA DE TABLAS xvii

RESUMEN xx

ABSTRACT xxi

CAPÍTULO I: INTRODUCCIÓN 1

I.1 Introducción 1

I.2 Planteamiento del problema 2

I.2.1 Determinación del problema 2

I.2.2 Formulación del problema general 4

I.2.3 Problemas específicos 5

I.3 Objetivos 6

I.3.1 Objetivo general 6

I.3.2 Objetivos específicos 6

I.4 Importancia y alcance de la investigación 7

I.5 Limitaciones de la investigación 8

CAPÍTULO II: REVISIÓN DE LA LITERATURA 10

II.1 Marco teórico 10

II.1.1 Centros de datos y salas de servidores 10

II.1.2 Internet de las Cosas (IoT) 11

II.1.3 Computación en el borde (Edge Computing) 13

II.1.4 Infraestructura híbrida 14

II.1.5 Detección de anomalías en sistemas de monitoreo 15

II.1.6 Umbrales de tiempo de respuesta en sistemas de monitoreo 20

II.2 Antecedentes del estudio 21

II.2.1 Antecedentes internacionales 21

II.2.2 Antecedentes nacionales 22

II.3 Bases teóricas 24

II.3.1 Hardware y electrónica 24

II.3.2 Protocolos de comunicación 34

II.3.3 Plataforma Gateway IoT 35

II.3.4 Software para monitoreo y control en el Gateway IoT 37

II.3.5 Infraestructura en la nube 38

II.3.6 Aprendizaje Automático 39

II.4 Definición de Términos 40

CAPÍTULO III: HIPÓTESIS Y VARIABLES 41

III.1 Hipótesis 41

III.1.1 Hipótesis general 41

III.1.2 Hipótesis específicas 41

III.2 Variables 42

III.2.1 Variables independientes 42

III.2.2 Variables dependientes 42

III.2.3 Variables externas 43

III.3 Operacionalización de variables 43

CAPÍTULO IV: MATERIALES Y MÉTODOS 45

IV.1 Área de estudio 45

IV.2 Diseño de investigación 45

IV.2.1 Tipo de investigación 45

IV.2.2 Nivel de investigación 46

IV.2.3 Enfoque metodológico 46

IV.2.4 Diseño de la investigación 46

IV.3 Población y muestra 46

IV.4 Procedimiento, técnicas e instrumentos de recolección de información 47

IV.4.1 Etapas de la investigación 47

IV.4.2 Técnicas e instrumentos por bloque del sistema 49

IV.4.3 Calibración de instrumentos 52

IV.4.4 Desarrollo e implementación de la propuesta 52

IV.5 Análisis estadístico 127

IV.5.1 Preparación de datos 127

IV.5.2 Estadística descriptiva 128

IV.5.3 Evaluación del modelo de detección de anomalías 128

CAPÍTULO V: RESULTADOS 129

V.1 Presentación y análisis de resultados 129

V.1.1 Hardware de la Unidad de Adquisición y Control (UAC) basado en ESP32 131

V.1.2 Firmware de la Unidad de Adquisición y Control 141

V.1.3 Modelo de Detección de Anomalías mediante Isolation Forest 151

V.1.4 Gateway IoT Basado en Raspberry Pi 159

V.1.5 Plataforma de Software para monitoreo y Control 166

V.1.6 Infraestructura Híbrida On-Premise y en la Nube 173

CAPÍTULO VI: DISCUSIÓN 177

VI.1 Discusión de la Hipótesis General 177

VI.2 Discusión de las Hipótesis Específicas 177

VI.2.1 Hardware de la Unidad de Adquisición y Control 177

VI.2.2 Firmware de la Unidad de Adquisición y Control 178

VI.2.3 Modelo de Detección de Anomalías mediante Isolation Forest 179

VI.2.4 Gateway IoT basado en Raspberry Pi 180

VI.2.5 Plataforma de Software para Monitoreo y Control 181

VI.2.6 Infraestructura Híbrida On-Premise y en la Nube 182

VI.3 Implicancias Técnicas y Económicas 183

VI.3.1 Implicancias técnicas 183

VI.3.2 Implicancias económicas 183

VI.4 Fortalezas y limitaciones 184

CAPÍTULO VII: CONCLUSIONES 185

VII.1 Conclusión General 185

VII.2 Conclusiones Específicas 185

CAPÍTULO VIII: RECOMENDACIONES 187

CAPÍTULO IX: BIBLIOGRAFÍA 188

CAPÍTULO X: ANEXOS 196

Anexo A 196

Anexo B 199

Anexo C 201

C.1. Configuración general del nodo y tópicos MQTT 202

C.2. Escaneo y selección de sensores mediante multiplexor I²C 202

C.3. Conexión WiFi, reconexión MQTT y suscripción a comandos 203

C.4. Construcción de la carga útil (payload) JSON y publicación MQTT 204

Anexo D 205

D.1. Configuración general del script de entrenamiento 206

D.2. Extracción y estructuración de lecturas históricas desde MariaDB 206

D.3. Generación del vector de características 208

D.4. Normalización, entrenamiento y almacenamiento del modelo 208

D.5. Configuración y carga del modelo en el microservicio de inferencia 209

D.6. Obtención de lecturas recientes y construcción de características 210

D.7. Inferencia, almacenamiento del resultado y ciclo autónomo 211

Anexo E 213

E.1. Servicios de red Access Point en Gateway IoT 213

E.2. Servicio de red uap0 para servicio DHCP 213

E.3. Servicio de mensajería MQTT BROKER 214

E.4. Unidades systemd 214

E.5. Archivo de configuración servicio de pantallas físicas 215

E.6. Nginx 216

E.7. Script de validación servicios Gateway IoT 217

Anexo F 220

Anexo G 223

Anexo H 229

Anexo I 230

LISTA DE FIGURAS

Figura 1 Diagrama de Ishikawa del problema central de investigación. 4

Figura 2 Arquitectura IoT de siete capas. 13

Figura 3 Tipos de anomalías en series temporales. 15

Figura 4 Principio de aislamiento del algoritmo Isolation Forest: las observaciones anómalas requieren menos particiones que las normales. 17

Figura 5 Interpretación del puntaje de anomalía del Isolation Forest. 18

Figura 6 Umbrales de tiempo de respuesta de las hipótesis específicas frente a los límites de percepción humana (Miller, 1968; Nielsen, 1993). 21

Figura 7 Diagrama de pines del ESP32-DevKit V. 24

Figura 8 Secuencia I²C: escritura de un byte al esclavo 0x47 (dato: carácter 'A'). 25

Figura 9 Diagrama funcional del multiplexor I²C TCA9548A. 27

Figura 10 Diagrama de bloques funcional del sensor SHT3x-DIS. 27

Figura 11 Diagrama de bloques funcional del sensor de iluminancia BH1750FVI. 28

Figura 12 Diagrama de bloques funcional del sensor de distancia por tiempo de vuelo VL53L0X. 29

Figura 13 Diagrama de bloques funcional de la unidad de medición inercial MPU-6050. 30

Figura 14 Principio de operación del reed switch como sensor de estado de puerta en gabinete. 31

Figura 15 Estructura interna y diagrama de conexión del LED SMD5050 RGB. 32

Figura 16 Dimensiones de referencia para rack de 19 pulgadas según EIA-310-D. 33

Figura 17 Diagrama esquemático del conector GPIO J8 de 40 pines de la Raspberry Pi 4 Model B. 36

Figura 18 Vista integral del sistema: las seis etapas de desarrollo convergentes en el equipo físico. 53

Figura 19 Diagrama arquitectónico del sistema de la UAC basado en ESP32. 55

Figura 20 Diagrama del sistema electrónico de la UAC. 58

Figura 21 Asignación de pines RJ45 para la conexión I²C mediante cable UTP CAT5e. 59

Figura 22 Sistema de alimentación del hardware de la UAC. 60

Figura 23 Prototipo PCB: Primera iteración del hardware de la UAC (grabado químico). 61

Figura 24 Distribución funcional del PCB. 62

Figura 25 Diagrama de distribución del chasis utilizado para el hardware. 63

Figura 26 Visión general del firmware ESP32 y su integración con el sistema. 64

Figura 27 Arquitectura del firmware ESP32. 67

Figura 28 Diagrama de flujo del firmware ESP32. 68

Figura 29 Salida monitor serial firmware UAC basado en ESP32. 71

Figura 30 Arquitectura del módulo de detección de anomalías basado en Isolation Forest. 72

Figura 31 Flujo de procesamiento (pipeline) de inferencia del modelo de detección de anomalías. 77

Figura 32 Análisis exploratorio de los datos de entrenamiento: distribución de variables continuas y matriz de correlaciones entre características. 80

Figura 33 Muestra de cinco observaciones del conjunto de datos de reentrenamiento con el vector completo de 11 características. 82

Figura 34 Convergencia del ensemble Isolation Forest sobre el vector promedio de entrenamiento, en función del número de árboles utilizados. 84

Figura 35 Proceso de despliegue del modelo Isolation Forest en el Gateway IoT. 86

Figura 36 Diagrama de flujo del script entrenar_modelo.py. 87

Figura 37 Diagrama de flujo del microservicio ml_service.py. 88

Figura 38 Salida del log de ml_service.py. 92

Figura 39 Arquitectura de red del Gateway IoT. 92

Figura 40 Separación de interfaces de red del Gateway: eth0 en LAN y UAP como AP dedicado. 97

Figura 41 Estado operativo de los servicios del Gateway verificado mediante comando systemctl. 99

Figura 42 Estado operativo de los servicios complementarios del Gateway IoT verificado mediante comando systemctl. 99

Figura 43 Arquitectura general de la plataforma de software para monitoreo y control. 100

Figura 44 Diagrama de flujo de módulos de la base de datos. 104

Figura 45 Diagrama de flujo de módulos del backend. 105

Figura 46 Diagrama de flujo de módulos del frontend. 108

Figura 47 Mockup de la interfaz principal dashboard. 109

Figura 48 Mockup gestión de actuadores y equipos de gabinete. 110

Figura 49 Mockup gestión de parámetros de red del Gateway IoT. 110

Figura 50 Mockup de apartado de configuración de alertas. 111

Figura 51 Mockup pestaña de ultimas incidencias detectadas por el modelo ML Isolation Forest. 112

Figura 52 Mockup Estadísticas del modelo de detección ML Isolation Forest. 112

Figura 53 Mockup Apartado de gestión de usuarios. 113

Figura 54 Mockup apartado de auditoría del sistema. 113

Figura 55 Mockup Configuración visual para los temas de la página web. 114

Figura 56 Mockup de la configuración visual de identidad de marca. 115

Figura 57 Mockup apartado de configuración visual sobre las pantallas físicas del equipo. 115

Figura 58 Mockup de apartado de backup y restauración del sistema. 116

Figura 59 Frontend desplegado en producción. 117

Figura 60 Mockup presentado en el diseño previo a implementación. 118

Figura 61 Arquitectura general de la infraestructura híbrida: plano local, Cloudflare y Azure. 119

Figura 62 Flujo de acceso remoto al backend mediante Cloudflare Tunnel. 123

Figura 63 Flujo de despliegue continuo del frontend: GitHub - Azure Static Web Apps. 124

Figura 64 Estado activo del servicio túnel Cloudflared. 125

Figura 65 Configuración DNS del dominio del sistema en Cloudflare. 126

Figura 66 Repositorio del sistema en GitHub. 126

Figura 67 Despliegue del frontend en Azure Static Web Apps. 127

Figura 68 PCB ensamblado para el hardware de la UAC (Iteración final). 132

Figura 69 Captura del monitor serie de muestra de dos ciclos consecutivos. 136

Figura 70 Gráfico de barras expresando los resultados de las pruebas. 136

Figura 71 Hardware final integrado en chasis 1RU: componentes internos (superior), vista frontal (centro) y vista general (inferior). 138

Figura 72 Salida validación en el Monitor Serial: lecturas válidas por sensor al término de la prueba de estabilidad. 138

Figura 73 Equipo instalado en rack. 141

Figura 74 Diagrama de flujo del firmware del ESP32. 142

Figura 75 Fragmento de configuración general del firmware del ESP32. 143

Figura 76 Salida del Monitor Serie durante ciclo de publicación: lecturas por sensor y confirmación MQTT. 144

Figura 77 Latencia registrada desde el Monitor Serial. 145

Figura 78 Latencia de publicación MQTT por ciclo (n = 30). 146

Figura 79 Composición de la latencia de publicación MQTT: lectura de sensores frente a mqtt.publish() (media, n = 30). 147

Figura 80 Reconexión WiFi del ESP32 tras reinicio del servicio en el Gateway. 148

Figura 81 Reconexión al bróker MQTT del ESP32 tras reinicio del servicio en el Gateway. 149

Figura 82 Tiempos de reconexión automática por ciclo - WiFi vs. MQTT (n = 30). 150

Figura 83 Lógica de inferencia del modelo Isolation Forest desplegado en el Gateway IoT. 151

Figura 84 Puntuación del Isolation Forest sobre datos reales: evolución temporal de la puntuación y eventos de anomalía detectados. 153

Figura 85 Matriz de confusión del Isolation Forest - evaluación (n = 1 200). 154

Figura 86 Tasa de detección por escenario de anomalía (120 muestras por escenario). 155

Figura 87 Distribución del decision_function por clase: normal vs. Anomalía. 156

Figura 88 Diagrama de caja del score de anomalía por clase: mediana, rango intercuartílico y valores atípicos. 159

Figura 89 Gateway IoT basado en Raspberry Pi 4B integrado en el chasis rack 1U. 160

Figura 90 Verificación de la presencia del Gateway IoT en la red LAN mediante Advanced IP Scanner. 161

Figura 91 Ciclo DHCP completo registrado por dnsmasq: la UAC recibe IP automáticamente al conectarse al AP. 161

Figura 92 Conexión de la UAC al AP del Gateway confirmada desde el Monitor Serie del ESP32: SSID conectado e IP asignada. 162

Figura 93 Sincronización verificada: mensaje MQTT publicado y registro correspondiente en MariaDB con marca temporal. 163

Figura 94 Latencias del Gateway IoT: broker MQTT y sincronización con MariaDB (media ± desviación estándar). 166

Figura 95 Plataforma web en operación: visualización en tiempo real de variables del gabinete con todos los sensores activos. 167

Figura 96 Dashboard principal con visualización de dashboard de variables ambientales del gabinete en tiempo real. 168

Figura 97 Configuración de regla de alerta con umbral de iluminancia en la plataforma web. 169

Figura 98 Correo de notificación recibido tras superación del umbral configurado, con detalle del evento. 170

Figura 99 Tiempos de respuesta de la plataforma: actualización del dashboard y notificación de alerta (media ± desviación estándar). 171

Figura 100 Módulos disponibles para el perfil administrador. 172

Figura 101 Módulos disponibles para el perfil visualizador. 172

Figura 102 Estado operativo del servicio cloudflared verificado mediante systemctl en el Gateway IoT. 173

Figura 103 Acceso remoto al backend vía HTTPS a través del túnel Cloudflare verificado desde red externa. 174

Figura 104 Interfaz de inicio de sesión del sistema accesible desde red externa bajo el dominio público web.tesis-monitoring.xyz. 174

Figura 105 Estado del túnel tesis-backend en el panel Cloudflare Zero Trust: estado HEALTHY y funcionamiento continuo. 175

LISTA DE TABLAS

Tabla 1 Diferencias entre detección de anomalías y predicción de fallas. 19

Tabla 2 Operacionalización de las variables de la investigación 43

Tabla 3 Resumen de etapas de implementación del sistema 48

Tabla 4 Técnicas e instrumentos de recolección por bloque del sistema 49

Tabla 5. Técnicas estadísticas aplicadas para el análisis y validación de resultados. 50

Tabla 6 Arquitectura por capas del sistema de monitoreo 54

Tabla 7 Requisitos del hardware de la UAC basado en ESP32 56

Tabla 8 Componentes seleccionados 57

Tabla 9 Niveles de tensión del sistema de alimentación 60

Tabla 10 Requisitos del firmware de la Unidad de Adquisición y Control (UAC) 65

Tabla 11 Selección de tecnologías 66

Tabla 12 Tópicos MQTT implementados en el firmware 69

Tabla 13 Constantes de red y MQTT definidas en el firmware 69

Tabla 14 Requisitos del módulo de detección de anomalías 73

Tabla 15 Tecnologías seleccionadas para el módulo de detección de anomalías 74

Tabla 16 Eventos habituales anómalos operacionales y variables asociadas para el modelo de detección 75

Tabla 17 Fases CRISP-DM aplicadas al módulo de detección de anomalías 78

Tabla 18 Características del vector de entrada al modelo (FEATURES) 81

Tabla 19 Hiperparámetros del modelo Isolation Forest 83

Tabla 20 Estructura de la tabla anomalias_ml 89

Tabla 21 Endpoints REST del módulo de detección de anomalías 90

Tabla 22 Comparación de hiperparámetros: modelo inicial vs. modelo reentrenado 91

Tabla 23 Requerimientos funcionales del Gateway IoT 93

Tabla 24 Resumen de componentes para el Gateway IoT 94

Tabla 25 Servicios principales del Gateway IoT basado en Raspberry Pi 95

Tabla 26 Capacidades complementarias del Gateway IoT 97

Tabla 27 Requerimientos de la plataforma de software 101

Tabla 28 Resumen de tecnologías de la plataforma de software 102

Tabla 29 Resumen de roles de la plataforma de software 106

Tabla 30 Módulos API REST utilizado por el backend 106

Tabla 31 Tópicos utilizados en la plataforma de software 107

Tabla 32 Requerimiento de la infraestructura híbrida 120

Tabla 33 Componentes de la infraestructura híbrida 121

Tabla 34 Escenarios de operación arquitectura híbrida 124

Tabla 35. Resumen de los indicadores de desempeño obtenidos por etapa. 129

Tabla 36 Resultados de la prueba de estabilidad multisensorial 132

Tabla 37. Muestra aleatoria de registros por ciclo (10 ciclos consecutivos) 133

Tabla 38 Condiciones de carga utilizadas para la evaluación de las tensiones del PCB 139

Tabla 39 Resumen de las tensiones medidas en el PCB bajo diferentes condiciones de carga 140

Tabla 40 Resultados de latencia de publicación MQTT (30 ciclos) 146

Tabla 41 Muestra ilustrativa de 10 ciclos de publicación MQTT. 147

Tabla 42 Tiempos de reconexión automática 149

Tabla 43 Parámetros del modelo Isolation Forest desplegado 152

Tabla 44 Métricas de clasificación del modelo Isolation Forest (n = 1 200) 154

Tabla 45 Tiempo de inferencia del modelo Isolation Forest 156

Tabla 46 Estadísticos descriptivos del score de anomalía por clase (n = 1 200). 157

Tabla 47 Prueba de significancia estadística y tamaño del efecto sobre la separabilidad de clases. 158

Tabla 48 Latencia del broker MQTT Mosquitto (n = 30 muestras, QoS 1) 162

Tabla 49 Latencia de sincronización MQTT → MariaDB (n = 20 mediciones). Indicador: tasa de eficiencia en procesamiento de mensajes MQTT 164

Tabla 50. Muestra ilustrativa de 10 mediciones de latencia del broker MQTT. 164

Tabla 51 Muestra ilustrativa de 10 mediciones de sincronización MQTT→MariaDB. 165

Tabla 52 Tiempo de actualización del tablero de monitoreo (dashboard) (n = 10 mediciones) 168

Tabla 53 Tiempo de notificación de alerta (n = 10 mediciones) 170

Tabla 54 Muestra ilustrativa de 10 mediciones de tiempo de respuesta de la plataforma. 170

Tabla 55 Eventos de auditoría registrados por tipo de acción (n = 15, sesión del 23 de marzo de 2026). 176

RESUMEN

Los centros de datos Tier 1 (nivel básico según el estándar Uptime Institute) carecen de supervisión a microescala dentro de sus gabinetes que detecte de forma autónoma, continua y temprana las anomalías ambientales y de acceso. Ante esta brecha, se diseñó e implementó un sistema inteligente de monitoreo de bajo costo, basado en sensado distribuido, un Gateway de Internet de las Cosas (IoT) con aprendizaje automático y arquitectura híbrida, para la supervisión y gestión remota de sus gabinetes.

La solución se desarrolló en seis etapas: hardware, firmware de comunicación, modelo de detección de anomalías, Gateway IoT, plataforma de software e infraestructura híbrida. Una Unidad de Adquisición y Control (UAC) mide temperatura, humedad, apertura de puerta, vibración, iluminancia y presencia, y transmite los datos al Gateway mediante el protocolo MQTT (Message Queuing Telemetry Transport), donde se centraliza la recepción y se ejecuta la detección de anomalías con Isolation Forest. La plataforma web ofrece visualización en tiempo real, gestión de usuarios, supervisión de equipos mediante SNMP y alertas por correo electrónico mediante SMTP, y una infraestructura híbrida con Azure y Cloudflare Tunnel garantiza acceso seguro y operación continua ante conectividad limitada. Las pruebas en un gabinete real validaron el funcionamiento del sistema y el cumplimiento de los indicadores definidos.

Palabras clave: Internet de las Cosas (IoT), Gateway IoT, monitoreo ambiental, detección de anomalías, centros de datos, Isolation Forest, MQTT

ABSTRACT

Tier 1 data centers (basic-level infrastructure under the Uptime Institute standard) lack micro-scale monitoring inside their cabinets capable of autonomously, continuously, and early detecting environmental and access anomalies. To address this gap, a low-cost intelligent monitoring system was designed and implemented, based on distributed sensing, an Internet of Things (IoT) gateway with machine learning, and a hybrid architecture, for the remote supervision and management of these cabinets.

The solution was developed in six stages: hardware, communication firmware, anomaly-detection model, IoT gateway, software platform, and hybrid deployment infrastructure. An Acquisition and Control Unit (ACU) measures temperature, humidity, door opening, vibration, illuminance, and presence, and transmits the data to the gateway via the MQTT (Message Queuing Telemetry Transport) protocol, where reception is centralized and anomaly detection is performed with Isolation Forest. A web platform provides real-time visualization, user management, network-device monitoring via SNMP, and email alerts via SMTP, and a hybrid infrastructure with Azure and Cloudflare Tunnel ensures secure access and continuous operation under limited connectivity. Testing on a real cabinet validated the system's operation and its compliance with the defined indicators.

Keywords: Internet of Things (IoT), IoT gateway, environmental monitoring, anomaly detection, data centers, Isolation Forest, MQTT

CAPÍTULO I:
INTRODUCCIÓN

I.1 Introducción

La creciente dependencia de las organizaciones respecto a su infraestructura tecnológica ha convertido a los centros de datos en activos críticos cuya operación continua resulta indispensable para la prestación de servicios esenciales. En entornos clasificados como Tier I, donde la ausencia de redundancia expone al sistema ante cualquier falla no anticipada, la supervisión de las condiciones ambientales y de acceso físico de los gabinetes constituye una necesidad operativa de primer orden. Sin embargo, la mayoría de las organizaciones con infraestructura de esta categoría carece de mecanismos de monitoreo a microescala que operen de forma autónoma, continua y a bajo costo, dependiendo en su lugar de revisiones manuales o alertas reactivas que solo se activan cuando el daño ya ha ocurrido.

Frente a esta problemática, el presente trabajo propone el diseño e implementación de un sistema inteligente de monitoreo para gabinetes en centros de datos Tier I, basado en sensado distribuido, comunicación IoT mediante el protocolo MQTT, detección autónoma de anomalías mediante aprendizaje automático y una arquitectura híbrida que combina procesamiento local con acceso remoto seguro a través de servicios en la nube. El sistema fue desarrollado sobre componentes de bajo costo y validado en un gabinete en operación real durante siete días continuos en las instalaciones de la empresa Cloud IT Perú S.A.C., demostrando su viabilidad técnica en un entorno productivo.

El documento se organiza en diez capítulos. El Capítulo I presenta el planteamiento del problema, los objetivos, la importancia y las limitantes de la investigación. El Capítulo II desarrolla la revisión de literatura, incluyendo el marco teórico, los antecedentes y las bases teóricas que sustentan las decisiones de diseño. El Capítulo III define las hipótesis, variables e indicadores de evaluación. El Capítulo IV

describe la metodología, el área de estudio y el desarrollo completo del sistema por etapas. El Capítulo V presenta los resultados obtenidos por indicador. El Capítulo VI discute los resultados en relación con los objetivos e hipótesis planteados. El Capítulo VII expone las conclusiones; el Capítulo VIII, las recomendaciones derivadas de la investigación; el Capítulo IX, la bibliografía; y finalmente el Capítulo X, los Anexos de la investigación.

I.2 Planteamiento del problema

I.2.1 Determinación del problema

Los centros de datos constituyen la base física de servicios críticos, como sistemas bancarios, plataformas de salud, redes de gobierno y comunicaciones. En particular, los centros de datos Tier I presentan una disponibilidad de 99,671 % y no cuentan con componentes redundantes, por lo que una falla en sus subsistemas puede afectar directamente la continuidad del servicio. En este tipo de infraestructura, los sistemas de monitoreo basados en IoT permiten supervisar variables operativas en tiempo real y reducir la dependencia de inspecciones manuales (Witczak & Szymoniak, 2024).

Dentro de estos entornos, el gabinete de servidores concentra riesgos operativos como acumulación térmica, variaciones de humedad, accesos físicos no autorizados y manipulaciones no registradas. Aunque las guías de ASHRAE establecen rangos recomendados de temperatura y humedad para proteger los equipos de procesamiento de datos (ASHRAE Technical Committee 9.9, 2021), el monitoreo general de sala no siempre permite detectar condiciones localizadas dentro de cada gabinete. Los problemas térmicos pueden presentarse a nivel de rack, especialmente ante fallas o cambios de carga (Lu et al., 2025). Por ello, el monitoreo continuo a microescala resulta necesario para detectar desviaciones antes de que generen fallas.

La brecha principal se encuentra en la falta de observabilidad interna del gabinete. En diversos cuartos de comunicaciones, gabinetes técnicos y pequeños centros de datos que operan con baja redundancia y condiciones similares a las de una

infraestructura Tier I, no se dispone de sistemas autónomos, continuos y de bajo costo para supervisar estas condiciones. En consecuencia, dependen de revisiones manuales, registros aislados o alertas reactivas. Las arquitecturas IoT con procesamiento distribuido ofrecen una alternativa de solución, al permitir capturar, transmitir y analizar datos desde dispositivos cercanos al entorno monitoreado (Witczak & Szymoniak, 2024). Además, los esquemas edge-cloud permiten combinar procesamiento local y escalabilidad en la nube, favoreciendo soluciones flexibles para monitoreo continuo (Andriulo et al., 2024).

Otra limitación es la ausencia de inteligencia en el tratamiento de los datos. Un sistema que solo registra valores ambientales no reduce de forma significativa la carga del operador, ya que este sigue interpretando manualmente los datos. Frente a ello, los modelos de detección de anomalías permiten identificar comportamientos atípicos en sistemas de sensores ambientales, incluso cuando no existen datos previamente etiquetados (Potharaju et al., 2025). Asimismo, los enfoques de Edge AI permiten ejecutar análisis cerca de la fuente de datos, reduciendo la dependencia de una conexión permanente con la nube (Reis, 2025).

También se identifica la falta de una plataforma integrada de gestión y acceso remoto. Sin una interfaz centralizada, alertas automáticas, registro histórico y trazabilidad, la supervisión fuera del horario presencial o desde otra ubicación se vuelve limitada. La implementación de plataformas centralizadas de monitoreo de infraestructura ha demostrado mejorar la supervisión y el control de los recursos al unificar la adquisición y la visualización de datos en un solo entorno (Aguirre et al., 2022). En entornos IoT industriales, el procesamiento en el borde permite detectar eventos en tiempo real y reducir la carga sobre sistemas centralizados (Zhukabayeva et al., 2025). Por ello, una solución de monitoreo debe integrar adquisición de datos, comunicación, análisis y gestión en una misma plataforma.

En conjunto, el problema que motiva esta investigación es la ausencia de un sistema integrado, autónomo, intuitivo, de bajo costo y con acceso remoto para supervisar de manera continua las condiciones ambientales y de acceso dentro de gabinetes ubicados en centros de datos Tier I. Esta carencia limita la detección oportuna de variaciones que podrían afectar la disponibilidad, seguridad y funcionamiento de los equipos tecnológicos instalados. La solución requiere dos

dimensiones complementarias: una capa de adquisición y procesamiento inteligente, basada en hardware de sensado distribuido de bajo costo construido sobre microcontroladores y computadoras de placa única (Alatoom & Smadi, 2025); y una capa de gestión, detección de anomalías y acceso remoto, basada en una plataforma de software con infraestructura híbrida, coherente con las arquitecturas IoT por capas (Choudhary, 2024). Estas dimensiones se descomponen en seis problemas particulares abordados en esta investigación. Los tres primeros problemas corresponden a la capa de adquisición y procesamiento. Los tres problemas restantes se vinculan con la capa de gestión y acceso remoto cuya relación causal con el problema central se presenta en la Figura 1.

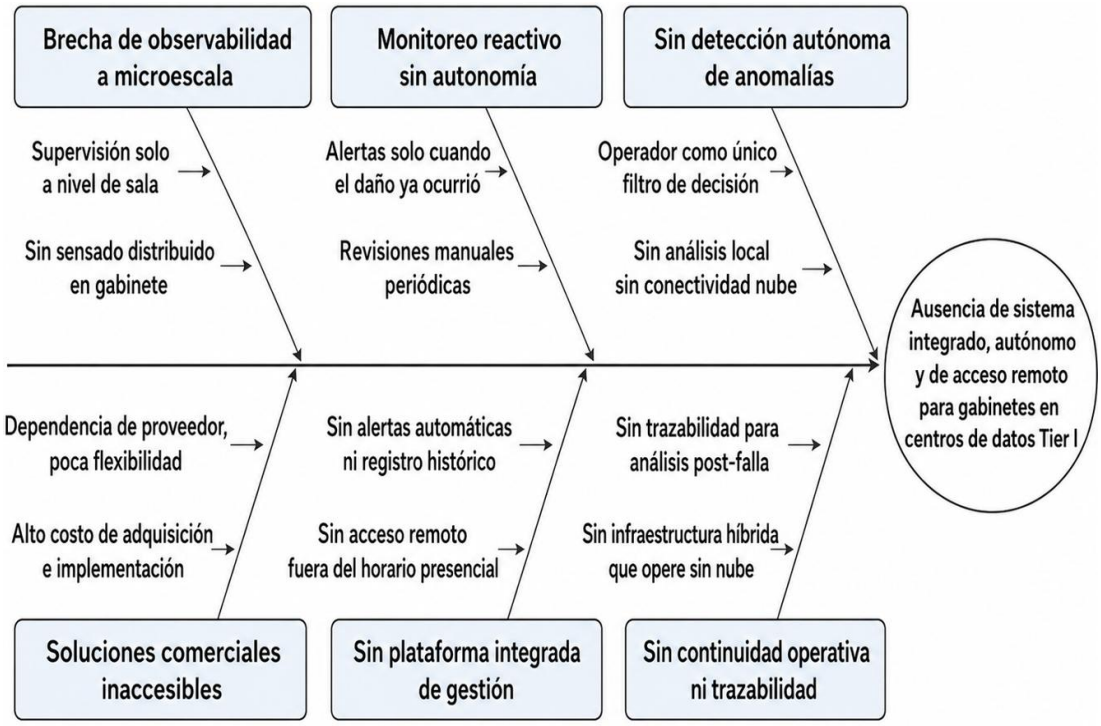


Figura 1 **Diagrama de Ishikawa del problema central de investigación.** Fuente. Elaboración propia.

I.2.2 Formulación del problema general

¿Cómo diseñar e implementar un sistema inteligente de monitoreo, basado en tecnologías de sensado y comunicación distribuida, un Gateway IoT y técnicas de

aprendizaje automático, sobre una arquitectura híbrida (nube–local), que permita la supervisión y gestión remota de gabinetes en centros de datos Tier 1?

I.2.3 Problemas específicos

- ¿Cómo diseñar e implementar el hardware de la Unidad de Adquisición y Control (UAC) basada en ESP32, que integre la adquisición de variables ambientales mediante sensores y el control de iluminación RGB, garantizando operación precisa y estable en gabinetes de centros de datos Tier 1?
- ¿Cómo desarrollar el firmware de la Unidad de Adquisición y Control (UAC) basada en ESP32, integrando la adquisición de datos de sensores, el control de actuadores y la transmisión mediante el protocolo MQTT, validando la confiabilidad de la comunicación y la transmisión en condiciones de operación continua?
- ¿Cómo incorporar un modelo de aprendizaje automático Isolation Forest en una Raspberry Pi para el análisis local de condiciones ambientales y estructurales, con capacidad de identificar patrones anómalos en gabinetes de centros de datos?
- ¿Cómo diseñar e integrar un Gateway IoT basado en Raspberry Pi que opere como nodo central del sistema, integrando la gestión de red inalámbrica dedicada para la UAC, el almacenamiento y procesamiento local de datos, y la sincronización de información hacia la plataforma de software, verificando la disponibilidad continua de sus servicios en condiciones de red local?
- ¿Cómo desarrollar una plataforma de software para la gestión, monitoreo, control y visualización dinámica de las variables internas de los gabinetes, validando su disponibilidad mediante pruebas de endpoints, flujo de datos MQTT y tiempo de actualización de la interfaz?
- ¿Cómo desplegar una infraestructura híbrida que combine entorno on-premise y servicios en la nube mediante Azure y Cloudflare para habilitar la administración remota y garantizar la operación continua en escenarios

con conectividad limitada, validando el acceso externo, disponibilidad y funcionamiento del despliegue continuo?

I.3 Objetivos

I.3.1 Objetivo general

Diseñar e implementar un sistema inteligente de monitoreo, basado en tecnologías de sensado y comunicación distribuida, un Gateway IoT y técnicas de aprendizaje automático en una arquitectura híbrida nube-local, para la supervisión y gestión remota de gabinetes en centros de datos Tier 1.

I.3.2 Objetivos específicos

- Diseñar e implementar el hardware de la Unidad de Adquisición y Control (UAC) basada en ESP32, que integre la adquisición de variables ambientales mediante sensores y el control de iluminación RGB, garantizando operación precisa y estable en gabinetes de centros de datos Tier 1.
- Desarrollar el firmware de la Unidad de Adquisición y Control (UAC) basada en ESP32, integrando la adquisición de datos de sensores, el control de actuadores y la transmisión mediante el protocolo MQTT, validando la confiabilidad de la comunicación y la medición de latencia de transmisión en condiciones de operación continua.
- Incorporar un modelo de aprendizaje automático Isolation Forest en una Raspberry Pi para el análisis local de condiciones ambientales y estructurales, con capacidad de identificar patrones anómalos en gabinetes de centros de datos.
- Diseñar e implementar un Gateway IoT basado en Raspberry Pi que opere como nodo central del sistema, integrando la gestión de red inalámbrica dedicada para la UAC, el almacenamiento y procesamiento local de datos,

y la sincronización de información hacia la plataforma de software, verificando la disponibilidad continua de sus servicios en condiciones de red local.

- Desarrollar una plataforma de software para la gestión, monitoreo, control y visualización dinámica de las variables internas de los gabinetes, validando su disponibilidad mediante pruebas de endpoints, flujo de datos MQTT y tiempo de actualización de la interfaz.
- Desplegar una infraestructura híbrida que combine entorno on-premise y servicios en la nube mediante Azure y Cloudflare para habilitar la administración remota y garantizar la operación continua en escenarios con conectividad limitada, validando el acceso externo, disponibilidad y funcionamiento del despliegue continuo.

I.4 Importancia y alcance de la investigación

Desde el punto de vista técnico-científico, esta investigación contribuye al campo del IoT industrial al proponer una arquitectura de monitoreo que integra sensado distribuido, procesamiento en el borde y gestión híbrida nube-local en un sistema unificado orientado a infraestructura crítica asimismo, la incorporación de un modelo de aprendizaje automático no supervisado que es operando localmente sin dependencia de conectividad permanente aporta evidencia empírica sobre la viabilidad de aplicar detección autónoma de anomalías en entornos Tier I de pequeña escala, donde la escasez de datos etiquetados dificulta el uso de enfoques supervisados (Rafique et al., 2024).

Desde el punto de vista práctico, el sistema desarrollado ofrece a los operadores de centros de datos Tier I una alternativa de bajo costo frente a soluciones comerciales de gestión de infraestructura cuya adopción resulta inviable en organizaciones con recursos limitados, debido al costo de licenciamiento, hardware propietario y/o soporte especializado (Alatoom & Smadi, 2025). La capacidad del sistema para generar alertas automáticas y mantener operación local continua ante pérdida de conectividad reduce la dependencia de supervisión humana constante y mejora la capacidad de respuesta ante incidentes ambientales y de acceso. Por ello, estas características lo hacen

pertinente para instituciones públicas, pequeñas empresas y entornos académicos con infraestructura Tier I crítica, pero sin presupuesto para plataformas DCIM comerciales.

En cuanto al alcance, la investigación se limita al diseño, implementación y validación del sistema en un único gabinete de centro de datos que cumple las características mínimas de la categoría Tier I según el estándar TIA-942, evaluado en un entorno real de operación. No se contempla la evaluación en un centro de datos certificado ni la operación simultánea de múltiples nodos de adquisición; ambos aspectos constituyen líneas de trabajo futuro. El período de validación abarca las sesiones realizadas entre febrero y mayo de 2026, asimismo los resultados son representativos en el contexto de aplicación propuesto mas no generalizables a entornos Tier II o superiores.

I.5 Limitaciones de la investigación

El sistema fue evaluado durante siete días continuos en un gabinete de comunicaciones instalado en un centro de datos real en operación, el cual, si bien no cuenta con certificación Tier 1, presenta características propias de una infraestructura básica de continuidad y comunicaciones. En este contexto, se validó el comportamiento del sistema bajo condiciones reales de funcionamiento. No obstante, los resultados de latencia en la transmisión MQTT y disponibilidad del acceso remoto podrían variar en entornos con mayor carga de tráfico, mayor interferencia electromagnética o políticas de firewall corporativas más restrictivas.

El modelo de detección de anomalías Isolation Forest fue entrenado con datos recolectados durante el período de prueba en el gabinete, los cuales representan un perfil térmico y ambiental específico correspondiente al entorno de evaluación. Al no disponer de registros etiquetados de condiciones anómalas reales, la evaluación del modelo se realizó mediante inyección controlada de escenarios de prueba. En consecuencia, la tasa de falsos positivos y la sensibilidad del modelo pueden variar al desplegarse en gabinetes con perfiles térmicos, patrones de acceso o cargas de trabajo distintos a los registrados durante el entrenamiento.

La arquitectura del sistema fue validada con un único nodo de adquisición UAC basado en ESP32. La operación simultánea de múltiples nodos publicando datos al mismo broker MQTT no fue evaluada experimentalmente, por lo que el comportamiento del sistema en cuanto a latencia, estabilidad del broker y capacidad de procesamiento del Gateway IoT ante múltiples fuentes de datos concurrentes no puede garantizarse a partir de los resultados obtenidos.

La disponibilidad del acceso remoto depende parcialmente de la infraestructura de Cloudflare Tunnel, servicio gestionado por un tercero cuya continuidad operativa está sujeta a condiciones externas al control del sistema desarrollado. La verificación de disponibilidad remota se realizó durante una ventana continua de seis horas dentro del período general de evaluación de siete días y no constituye una medición de disponibilidad a largo plazo. Asimismo, las pruebas de los módulos de firmware, plataforma de software e infraestructura híbrida se realizaron en un entorno de prueba controlado que simuló la operación conjunta de todos los sensores, por lo que los resultados reflejan el comportamiento del sistema bajo condiciones ideales de conectividad y carga.

CAPÍTULO II:
REVISIÓN DE LA LITERATURA

II.1 Marco teórico

II.1.1 Centros de datos y salas de servidores

Un centro de datos es una instalación física que aloja sistemas de cómputo, almacenamiento y comunicaciones junto con la infraestructura de soporte necesaria como alimentación eléctrica, refrigeración y conectividad, para garantizar la disponibilidad continua de los servicios digitales que sustenta (Uptime Institute, 2022). La clasificación más difundida es el esquema de cuatro niveles (Tier I–IV) del Uptime Institute: el Tier I garantiza disponibilidad del 99,671 % sin redundancia de componentes; el Tier II incorpora componentes redundantes; el Tier III añade múltiples rutas de distribución con mantenimiento concurrente; y el Tier IV exige tolerancia a fallos con disponibilidad del 99,995 %. El presente proyecto se orienta a entornos Tier I, donde la ausencia de redundancia hace crítica la detección temprana de anomalías ambientales.

El monitoreo ambiental de gabinetes de servidores se sustenta en dos aspectos complementarios: a) las variables que resultan determinantes para la fiabilidad operativa, y b) los estándares internacionales que definen los rangos aceptables para dichas variables.

a) Variables críticas de monitoreo

Las variables ambientales determinantes para la fiabilidad operativa de un gabinete incluyen: temperatura del aire en la entrada de los equipos, humedad relativa, estado de acceso físico y disponibilidad del suministro eléctrico. La temperatura es la variable de mayor impacto: incrementos sostenidos aceleran la degradación de componentes electrónicos y reducen su vida útil media (ASHRAE Technical Committee 9.9, 2021). La humedad relativa excesiva genera condensación; la

deficiente favorece descargas electrostáticas. El acceso físico no autorizado constituye un vector de incidente de seguridad documentado en entornos de infraestructura crítica.

b) Estándares relevantes

ASHRAE TC 9.9 define rangos recomendados (A1–A4) para temperatura y humedad en equipos de procesamiento de datos; la quinta edición establece para la clase A1 un rango recomendado de 18–27 °C y humedad relativa de 8–80 % sin condensación (ASHRAE Technical Committee 9.9, 2021). El estándar ANSI/TIA-942-C especifica los requerimientos de infraestructura para centros de datos, definiendo cuatro niveles de fiabilidad alineados con la clasificación Tier del Uptime Institute (Telecommunications Industry Association, 2024).

II.1.2 Internet de las Cosas (IoT)

El Internet de las Cosas puede entenderse como un ecosistema de objetos físicos dotados de sensado, identificación y conectividad capaces de intercambiar datos y habilitar servicios inteligentes sobre infraestructuras interoperables (Choudhary, 2024). De forma complementaria, el estándar IEEE 2413-2019 formaliza un marco arquitectónico de referencia que organiza los sistemas IoT en dominios (dispositivo, conectividad, aplicación y usuario) con el fin de facilitar la interoperabilidad entre implementaciones heterogéneas (IEEE, 2020). Su estudio requiere abordar tanto su estructura funcional, representada comúnmente mediante el modelo de siete capas, como sus principales ámbitos de aplicación.

a) Modelo de siete capas

El modelo de siete capas de IoT permite comprender la arquitectura de un sistema IoT desde sus componentes físicos hasta su uso dentro de un proceso organizacional. Este modelo organiza el funcionamiento de una solución IoT en siete niveles: dispositivos físicos y controladores, donde se ubican sensores, actuadores y unidades de control; conectividad, encargada de la transmisión de datos; Edge Computing, orientada al procesamiento cercano al entorno físico; acumulación de datos, relacionada con la recepción y almacenamiento de información; abstracción de datos, donde los datos son procesados e interpretados; aplicación, enfocada en la

visualización, alertas y control; y colaboración y procesos, que representa la integración del sistema con las actividades y decisiones de los usuarios. Este esquema continúa siendo una referencia útil para describir sistemas IoT de monitoreo ambiental de escala reducida como el propuesto en la presente tesis (Choudhary, 2024), tal como se aprecia en la Figura 2.

Este modelo sirve como base conceptual para ubicar los principales componentes desarrollados en la tesis dentro de una arquitectura de referencia IoT de siete capas, permitiendo evidenciar cómo cada elemento del sistema propuesto cumple una función específica dentro del flujo de datos, desde la captura física hasta la generación de valor aplicativo. El hardware de la UAC basado en ESP32 se vincula con la capa de dispositivos físicos y controladores, al ser responsable de la captura directa de las variables ambientales mediante los sensores instalados en los gabinetes del centro de datos. El firmware de la UAC, por su parte, se asocia con la capa de conectividad, ya que se encarga de la lectura periódica de los sensores, la estructuración de los datos en un formato estandarizado y su transmisión hacia el gateway mediante el protocolo MQTT, garantizando una comunicación ligera y confiable. El Gateway IoT basado en Raspberry Pi corresponde a la capa de Edge Computing, pues concentra el procesamiento local de los datos recibidos antes de su envío hacia niveles superiores, reduciendo la latencia y la dependencia constante de la nube. La plataforma de software para monitoreo y control se relaciona con las capas de acumulación de datos y aplicación, al encargarse tanto del almacenamiento estructurado de la información como de su visualización y uso por parte del usuario final. El modelo de detección de anomalías basado en Isolation Forest se ubica en la capa de abstracción de datos, dado que su función es interpretar las mediciones ya procesadas para identificar comportamientos atípicos que podrían representar riesgos operativos. La infraestructura híbrida on-premise y en la nube, en cambio, no se asocia con una sola capa, sino que actúa de forma transversal, soportando de manera simultánea la conectividad, el almacenamiento, la disponibilidad de la plataforma y el acceso remoto seguro a través de los distintos niveles del modelo. Finalmente, la capa de colaboración y procesos no cuenta con una etapa directa dentro del desarrollo de la tesis, ya que su alcance corresponde al uso organizacional y a la integración del sistema dentro de los procesos internos de la empresa, aspectos que exceden el desarrollo técnico principal de la investigación.

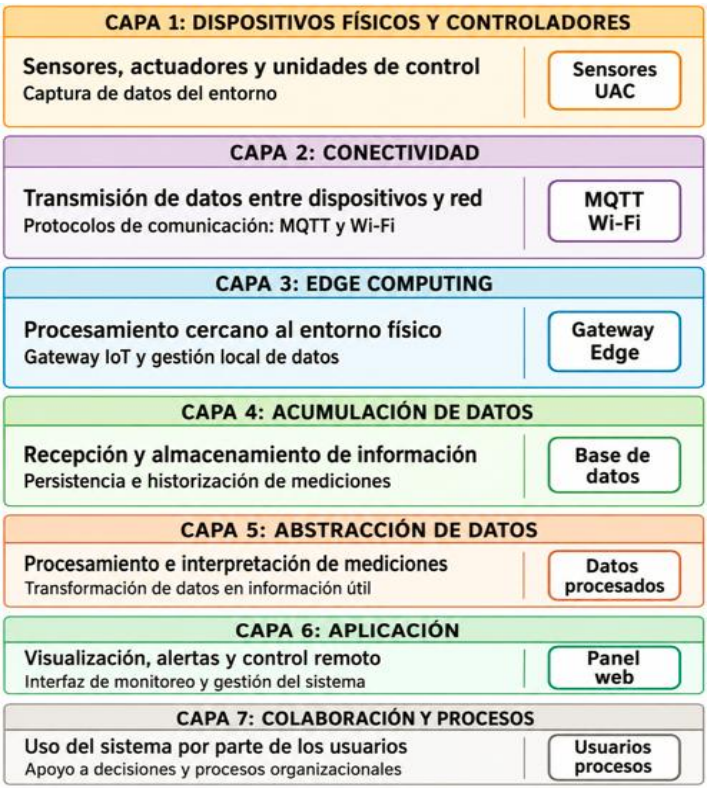


Figura 2 **Arquitectura IoT de siete capas.** Fuente. Adaptado de "Internet of Things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions" (Vol. 4, Núm. 31), por A. Choudhary, 2024, Springer Nature Link.

b) IoT en entornos industriales y de infraestructura crítica.

La aplicación de IoT en entornos de infraestructura crítica introduce requisitos adicionales de fiabilidad, latencia determinista y seguridad que no son estrictamente necesarios en aplicaciones de consumo. En el contexto de centros de datos, la instrumentación IoT permite la supervisión continua de variables ambientales con resolución temporal elevada, habilitando la detección de condiciones anómalas antes de que escalen a fallos de equipos (Cook et al., 2020).

II.1.3 Computación en el borde (Edge Computing)

La computación en el borde es un paradigma de procesamiento distribuido que desplaza capacidad de cómputo hacia nodos físicamente próximos a las fuentes de

datos, reduciendo la latencia de respuesta y el volumen de datos que deben transmitirse hacia infraestructuras centralizadas. En contraste con el cloud computing, donde el procesamiento se concentra en centros de datos remotos, el borde privilegia la respuesta inmediata, la continuidad operativa y el tratamiento local de la información; la nube, en cambio, prioriza escalabilidad y capacidad de almacenamiento y cómputo (Avasalcai et al., 2020).

En una arquitectura IoT, el gateway actúa como nodo de borde al concentrar los datos provenientes de múltiples dispositivos de campo, ejecutar procesamiento local filtrado, agregación, inferencia y gestionar la comunicación hacia la capa de aplicación o la nube. Avasalcai et al. (2020) identifican al gateway como el componente crítico que permite desacoplar la lógica de los dispositivos embebidos de la plataforma de servicios, incrementando la resiliencia del sistema ante interrupciones de conectividad. En el presente proyecto, la Raspberry Pi cumple este rol.

II.1.4 Infraestructura híbrida

Una arquitectura híbrida combina recursos de infraestructura on-premise con servicios de computación externos o en la nube, distribuyendo funciones según criterios de latencia, continuidad operativa, seguridad y costo. En sistemas IoT, este enfoque permite mantener la adquisición y el procesamiento crítico cerca del proceso físico, mientras que las funciones de acceso remoto, respaldo o visualización ampliada pueden trasladarse a entornos de mayor elasticidad computacional (Avasalcai et al., 2020).

Dentro de este modelo, la interconexión entre recursos locales y servicios externos constituye una decisión de diseño con impacto directo en la seguridad del sistema. El acceso remoto a servicios alojados en redes privadas sin apertura de puertos en el firewall puede implementarse mediante conectores salientes cifrados que enlazan el servicio interno con un proxy o plano de acceso externo.

Este enfoque reduce la superficie de exposición, evita publicar directamente direcciones o puertos de la red local y resulta coherente con los principios de arquitectura Zero Trust para acceso a aplicaciones privadas. En el presente proyecto,

dicho modelo se materializa mediante un túnel inverso gestionado desde el nodo local hacia el servicio externo (National Institute of Standards and Technology, 2023).

II.1.5 Detección de anomalías en sistemas de monitoreo

Una anomalía en una serie temporal es una observación o secuencia de observaciones que se desvía del comportamiento esperado bajo condiciones normales de operación. Cook et al. (2020) distinguen tres categorías mostradas en la Figura 3: anomalías puntuales, donde un único valor atípico se aparta claramente del resto de la serie sin considerar su entorno temporal; anomalías contextuales, en las que el valor no resulta atípico en términos absolutos, pero sí lo es respecto al contexto temporal en el que se presenta, por ejemplo, una temperatura que sería normal en cierto horario pero anómala en otro, lo que exige un análisis que considere la variable temporal y no solo su magnitud; y anomalías colectivas, donde una secuencia de valores en conjunto resulta anómala, aunque cada punto individual sea plausible dentro del rango esperado, evidenciando un patrón sostenido que solo se revela al observar el comportamiento conjunto de los datos. Por esta razón, un sistema de detección orientado a este tipo de entornos debe ser capaz de capturar no solo valores extremos, sino también tendencias y secuencias que, en conjunto, representen un riesgo real para la continuidad operativa.

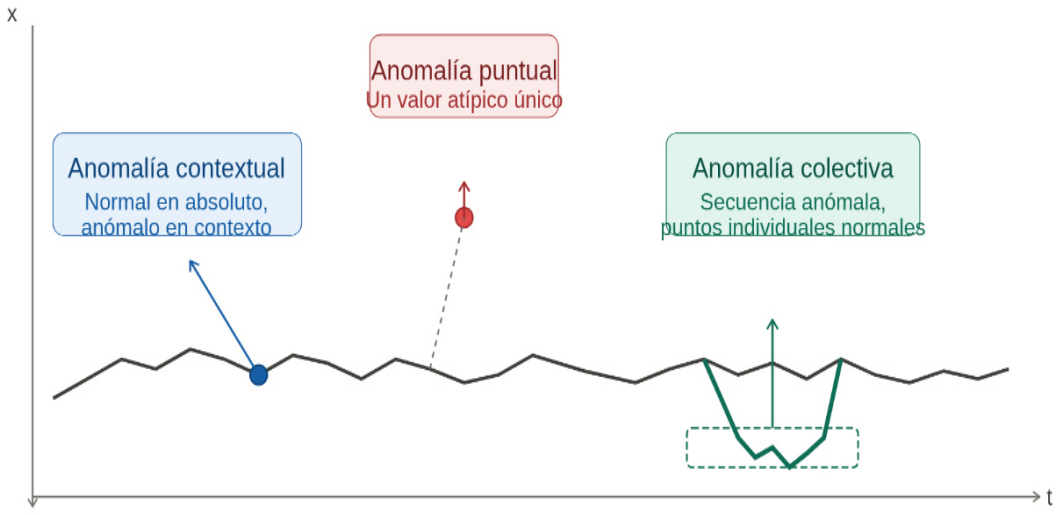


Figura 3 **Tipos de anomalías en series temporales.** Fuente. Adaptado de (Chandola et al., 2009) y (Cook et al., 2020).

La detección de estas anomalías puede abordarse mediante métodos supervisados o no supervisados, según la disponibilidad de datos etiquetados. Los métodos supervisados de detección de anomalías requieren conjuntos de datos etiquetados, condición difícilmente satisfecha en sistemas de monitoreo de infraestructura donde los fallos son infrecuentes. Los métodos no supervisados modelan la distribución del comportamiento normal a partir de datos históricos sin etiquetas, identificando como anómalas las observaciones que se alejan de dicha distribución (Chatterjee & Ahmed, 2022). Para el presente sistema, el enfoque no supervisado es el apropiado dado que no se dispone de registros etiquetados de condiciones anómalas previas.

a) Isolation Forest como mecanismo de detección de anomalías

Entre los métodos no supervisados disponibles, el presente sistema emplea el algoritmo Isolation Forest, propuesto por (Liu et al., 2008). Su principio fundamental se basa en una propiedad estadística de las anomalías: al ser pocas en número y presentar características alejadas del comportamiento mayoritario del sistema, son intrínsecamente más fáciles de aislar mediante particiones aleatorias sucesivas que las observaciones normales. Este principio de aislamiento permite prescindir de la estimación de densidad o de cálculos de distancia, haciendo al algoritmo eficiente tanto en entrenamiento como en inferencia, lo que resulta especialmente relevante para plataformas de borde con recursos computacionales limitados (Xu et al., 2023).

b) Construcción del árbol de aislamiento (iTree)

El algoritmo construye un conjunto de t árboles binarios aleatorios denominados árboles de aislamiento (*iTrees*). Cada árbol se genera a partir de una submuestra aleatoria de tamaño ψ extraída del conjunto de entrenamiento. En cada nodo interno del árbol se selecciona aleatoriamente una característica q y un valor de corte p dentro del rango $[\min(q), \max(q)]$, dividiendo las observaciones en dos subconjuntos. Este proceso se repite recursivamente hasta que cada punto queda aislado en una hoja o se alcanza la profundidad máxima del árbol. La Figura 4 ilustra este proceso para dos observaciones: una normal y una anómala. Se observa que la muestra normal requiere ocho particiones para ser aislada, mientras que la observación anómala se separa con solo dos, evidenciando su mayor facilidad de detección por el modelo.

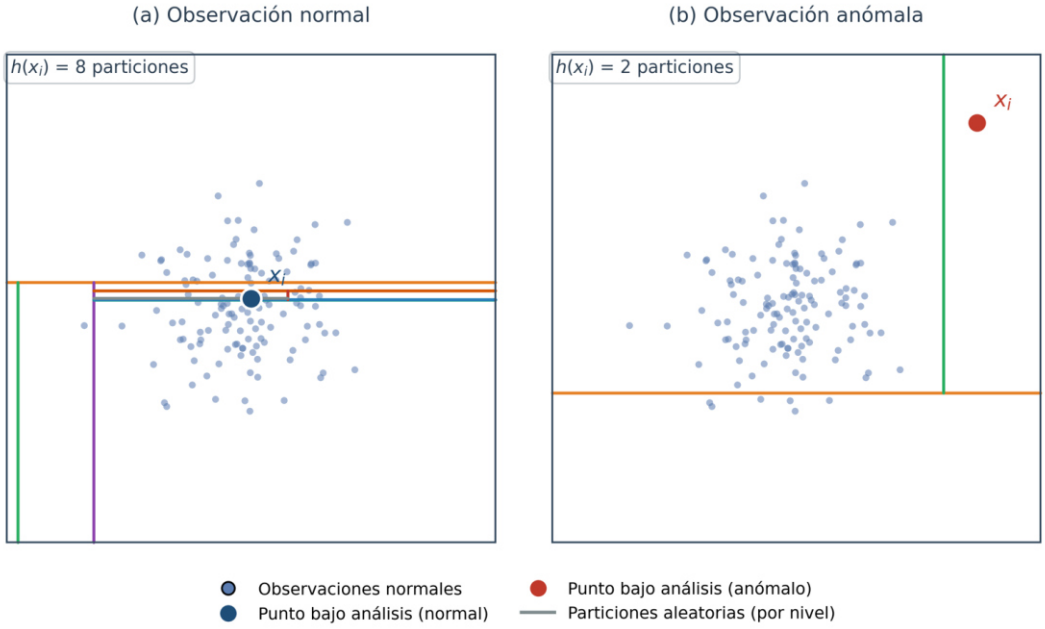


Figura 4 **Principio de aislamiento del algoritmo Isolation Forest: las observaciones anómalas requieren menos particiones que las normales.** Fuente. Elaboración propia, adaptado de (Liu et al., 2008).

c) Longitud de camino y puntaje de anomalía

La métrica fundamental del algoritmo es la longitud de camino *h(x)*, definida como el número de aristas recorridas en un iTree desde la raíz hasta el nodo donde la observación *x* queda aislada. Dado que las observaciones anómalas se aíslan en niveles superiores del árbol, su longitud de camino promedio *E[h(x)]*, calculada sobre todos los *t* iTrees, resulta considerablemente menor que la de las observaciones normales (Liu et al., 2008).

Para comparar longitudes de camino entre árboles de distinto tamaño, se normaliza mediante la longitud de camino promedio esperada para un árbol construido sobre *n* observaciones, definida en la **Ecuación 1**:

Ecuación 1:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

donde *H(i)* es el *i*-ésimo número armónico, aproximado como $H(i) = ln(i) + \gamma$, siendo $\gamma \approx 0{,}5772$ la constante de Euler-Mascheroni. Esta cantidad actúa como factor

de normalización y representa la longitud de camino esperada en un árbol de búsqueda binaria sobre *n* observaciones.

El puntaje de anomalía *s(x, n)* se define en el intervalo (0, 1) mediante la **Ecuación 2**:

Ecuación 2:

$$s(x,n) = 2^{-\frac{E[h(x)]}{c(n)}}$$

donde E[h(x)] es la media de las longitudes de camino de la observación x sobre todos los iTrees del conjunto. La interpretación del puntaje se resume en la Figura 5, descrita a continuación:

- *s(x, n) - 1:* la observación se aisló con muy pocas particiones - **alta probabilidad de anomalía**.
- *s(x, n) - 0:* la observación requirió muchas particiones para aislarse - **comportamiento normal**.
- *s(x, n) ≈ 0,5:* el aislamiento no fue ni rápido ni lento - **observación ambigua, sin señal definida**.

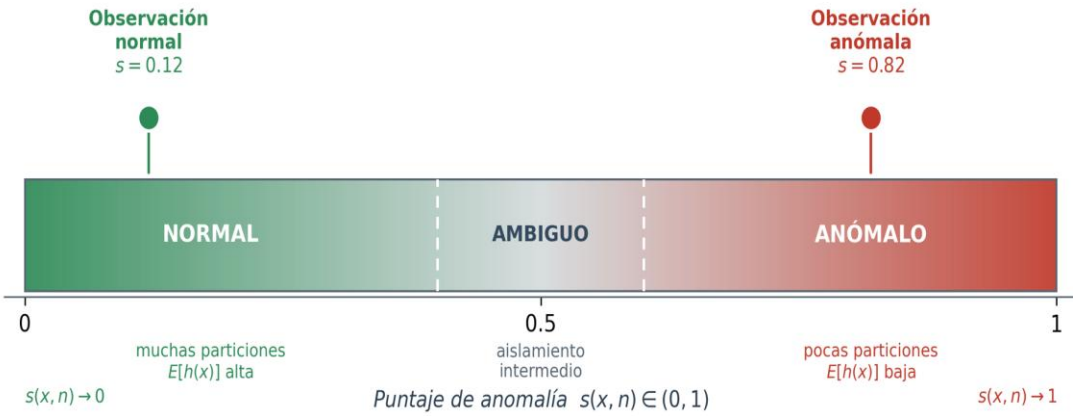


Figura 5 **Interpretación del puntaje de anomalía del Isolation Forest.** Fuente. Elaboración propia, adaptado de (Liu et al., 2008).

La función de decisión implementada por la librería scikit-learn (*decision_function*) devuelve una versión desplazada del puntaje, de modo que valores negativos corresponden a anomalías (clasificadas como −1 por la función *predict*) y

valores positivos a comportamiento normal (clasificados como +1), en coherencia con la convención de clasificadores de la librería (Geron, 2022).

d) Ventajas operacionales para el dominio de centros de datos

Las propiedades de Isolation Forest lo hacen especialmente adecuado para el monitoreo de gabinetes en centros de datos Tier 1: (i) no requiere supuestos sobre la distribución de los datos ni umbrales predefinidos, permitiendo adaptarse al perfil térmico y ambiental específico de cada gabinete; (ii) su complejidad computacional es lineal $O(n)$ en entrenamiento, frente a la complejidad cuadrática $O(n^2)$ de métodos basados en distancias, habilitando su ejecución en hardware de borde como la Raspberry Pi; y (iii) al ser no supervisado, opera sin etiquetas previas de falla, condición habitual en entornos donde los eventos anómalos son infrecuentes y raramente documentados . El detalle de la implementación práctica mediante scikit-learn, los hiperparámetros seleccionados y el proceso de entrenamiento y despliegue se describen en la sección IV.5.3 del presente documento.

e) Detección de anomalías frente a predicción de fallas

Es necesario distinguir la detección de anomalías, empleada en este trabajo, de la predicción o pronóstico de fallas. La detección identifica, a partir del estado presente, observaciones que se desvían del comportamiento normal aprendido (Chandola et al., 2009; Rafique et al., 2024), sin proyectar el futuro del sistema; la predicción estima cuándo ocurrirá una falla aún no manifestada, típicamente con series de tiempo o modelos de vida útil remanente sobre historiales etiquetados. El Isolation Forest implementado clasifica cada ventana de observaciones como normal o anómala, sin estimar cuándo ocurrirá una falla futura. La Tabla 1 resume las diferencias.

Tabla 1 **Diferencias entre detección de anomalías y predicción de fallas.**

Dimensión	**Detección de anomalías**	**Predicción de fallas**
Pregunta que responde	¿El estado actual es anómalo?	¿Cuándo ocurrirá una falla futura?

Dimensión	Detección de anomalías	Predicción de fallas
Datos de entrada	Estado presente de las variables monitoreadas	Historial temporal de variables y de fallas etiquetadas
Salida del modelo	Clasificación binaria o puntaje de anomalía sobre la observación actual	Probabilidad de falla u horizonte temporal estimado.
Requisito de etiquetado	No requiere fallas etiquetadas previas (no supervisado)	Generalmente requiere historial de fallas etiquetadas
Aplicación en esta investigación	Isolation Forest (Etapa 3)	No implementada

Fuente. Elaboración propia, con base en Chandola et al. (2009) y Rafique et al. (2024).

II.1.6 Umbrales de tiempo de respuesta en sistemas de monitoreo

Los umbrales temporales de las hipótesis específicas (500 ms en la red, 5 s en actualización de interfaz y reconexión, 60 s en alertas) se fundamentan en los límites de tiempo de respuesta percibidos por el usuario descritos por Miller (1968) y sistematizados por Nielsen (1993): 0,1 s se percibe como instantáneo, 1 s mantiene el flujo de pensamiento sin interrupción, y 10 s es el límite antes de que la tarea se perciba como interrumpida. Sobre esta base, las transmisiones internas (MQTT, sincronización con base de datos) se acotan en cientos de milisegundos, muy por debajo del límite de 1 s; la actualización de interfaz y la reconexión se acotan en segundos individuales, dentro del límite de 10 s; y la notificación por correo, sujeta a la latencia del protocolo SMTP, se acota en 60 s, un margen mayor pero aún compatible con una respuesta oportuna ante una condición crítica. La Figura 6 ejemplifica la explicación brindada.

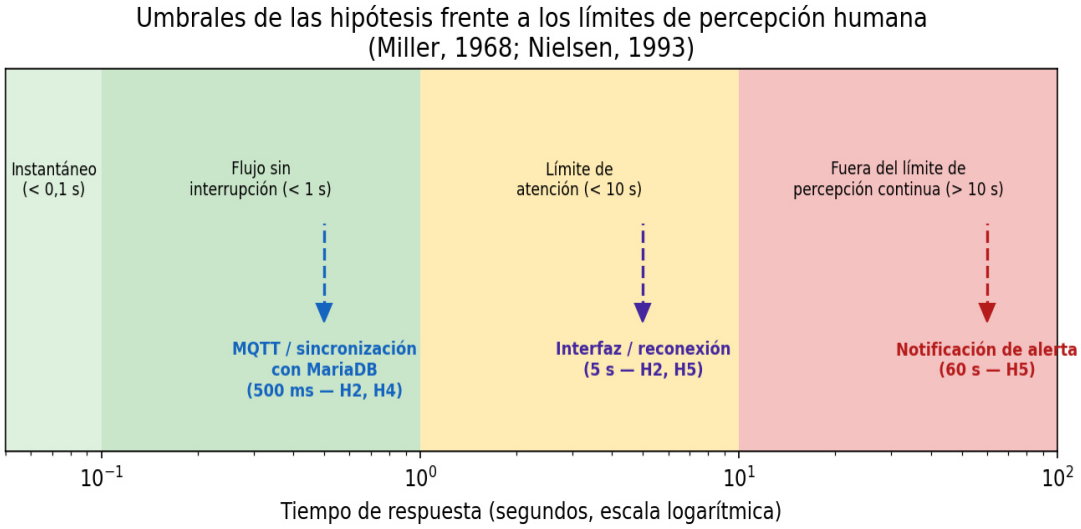


Figura 6 **Umbrales de tiempo de respuesta de las hipótesis específicas frente a los límites de percepción humana (Miller, 1968; Nielsen, 1993).** Fuente. Elaboración propia, con base en Miller (1968) y Nielsen (1993).

II.2 Antecedentes del estudio

II.2.1 Antecedentes internacionales

Jiang et al. (2021) en su artículo proponen un sistema de control predictivo tolerante a fallos basado en datos para la regulación de temperatura en centros de datos con arquitectura de enfriamiento por rack, publicado en la revista Mechatronics, identificaron como problema la inestabilidad térmica en gabinetes de centros de datos ante fallos en los sistemas de enfriamiento, situación que compromete la disponibilidad de los equipos al no existir mecanismos de control capaces de anticipar y compensar dichas fallas de manera autónoma. Su objetivo fue desarrollar un modelo de control predictivo basado en datos capaz de mantener la regulación de temperatura dentro de rangos seguros aun en presencia de fallos en el sistema de enfriamiento por rack.

Reis (2025) en su artículo propuso un marco ligero de procesamiento de señales y Edge AI para la detección de anomalías en tiempo real en redes IoT de sensores. El autor identificó como problema que muchos enfoques tradicionales de detección dependen de procesamiento en la nube, lo que genera limitaciones de latencia, ancho

de banda, escalabilidad y eficiencia energética. Su objetivo fue desarrollar un enfoque de detección de anomalías basado en extracción de características y modelos ligeros de aprendizaje automático ejecutados cerca de la fuente de datos. Este trabajo aporta sustento técnico para incorporar procesamiento local y detección autónoma de anomalías en sistemas IoT, aspecto central en la arquitectura propuesta.

Elnaggar (2025) en su artículo publicado en el IAR Journal of Engineering and Technology ha propuesto un sistema de monitoreo ambiental IoT con alertas en tiempo real y analítica predictiva basado en MQTT. Ha identificado como problema la dependencia de supervisión humana constante en los sistemas de monitoreo ambiental convencionales, que carecen de capacidad para generar notificaciones automáticas ante condiciones críticas. Su objetivo ha sido desarrollar un sistema IoT de alta confiabilidad con generación automática de alertas en tiempo real durante operación continua prolongada.

Rafique et al. (2024) en su artículo proponen una revisión sistemática de técnicas de aprendizaje automático para detección de anomalías en redes IoT, publicado en la revista Sensors de MDPI, identificaron como problema la incapacidad de los sistemas basados en reglas fijas para adaptarse a patrones de comportamiento anómalo variables en entornos IoT dinámicos. Su objetivo fue evaluar y comparar el rendimiento de algoritmos supervisados y no supervisados aplicados a la detección de anomalías sobre conjuntos de datos reales de redes IoT. La revisión contribuye a esta investigación al justificar el uso de técnicas de aprendizaje automático para identificar comportamientos anómalos en entornos IoT dinámicos.

II.2.2 Antecedentes nacionales

Santillán (2026) en su tesis presentada en la Universidad Nacional Mayor de San Marcos ha propuesto el diseño de un invernadero inteligente mediante Internet de las Cosas para la agricultura. Ha identificado como problema la ausencia de control automatizado de variables ambientales en cultivos bajo invernadero en zonas rurales con conectividad limitada, lo que generaba pérdidas productivas por condiciones ambientales no detectadas oportunamente. Su objetivo ha sido diseñar e implementar un sistema IoT de bajo costo capaz de monitorear y controlar la temperatura y la

humedad de forma autónoma y continua. La propuesta integró un microcontrolador ESP32, sensores de humedad del suelo, temperatura y luminosidad, además de actuadores para el control automático del riego, la ventilación y la iluminación. integrando un dashboard web para supervisar el estado del cultivo.

Dulanto (2025) en su tesis presentada en la Universidad Nacional Mayor de San Marcos ha propuesto el diseño de un sistema IoT basado en computación de borde para el monitoreo de la calidad del aire en el distrito de Chancay. Ha identificado como problema la ausencia de un sistema de monitoreo de calidad del aire en un distrito con actividades económicas relevantes y presencia de infraestructura portuaria. Su objetivo ha sido diseñar un sistema IoT con arquitectura de tres capas: capa IoT, capa Edge y capa de solución de negocios. La propuesta ha integrado sensores para contaminantes gaseosos, material particulado y variables meteorológicas, transmisión mediante LoRa, procesamiento local y visualización mediante panel (dashboard).

Zevallos (2024) en su tesis presentada en la Universidad de Lima ha propuesto un modelo de monitoreo de calidad del aire en espacios interiores mediante tecnología IoT. Ha identificado como problema la inexistencia de mecanismos de medición continua y asequible de contaminantes en ambientes cerrados de una institución educativa, lo que impedía conocer las condiciones reales del aire que respiraban los ocupantes. Su objetivo ha sido construir un modelo IoT capaz de adquirir, transmitir y visualizar datos de calidad del aire en tiempo real con registro histórico. Este trabajo ha aportado al enfoque de monitoreo continuo, pues plantea la adquisición, transmisión, visualización y registro histórico de variables ambientales en espacios interiores.

Garcia (2023) en su tesis presentada en la Pontificia Universidad Católica del Perú ha propuesto el diseño de una plataforma de control y supervisión remota de equipos de laboratorio mediante IoT. Ha identificado como problema la imposibilidad de operar equipos críticos de infraestructura académica desde ubicaciones externas, lo que afectaba la continuidad de actividades que requerían presencia física obligatoria. Su objetivo ha sido implementar una plataforma IoT con acceso remoto seguro para el monitoreo y control de equipos de laboratorio desde cualquier ubicación con conectividad a Internet.

II.3 Bases teóricas

II.3.1 Hardware y electrónica

a) Microcontrolador ESP32-WROOM-32

El ESP32-WROOM-32 es un módulo de sistema en chip (SoC) diseñado por Espressif Systems que integra un procesador de doble núcleo a 240 MHz, conectividad Wi-Fi y Bluetooth, junto con periféricos que incluyen buses I²C, SPI, UART, conversor analógico-digital de 12 bits y salidas PWM (Espressif Systems, 2025).

En este contexto, el módulo DOIT ESP32 DevKit V1 integra el SoC ESP32-WROOM-32 tal como se observa en la Figura 7, donde la placa incluye múltiples pines para distintas interfaces y, en la parte superior, una antena integrada que permite la comunicación Wi-Fi y Bluetooth, lo que simplifica el diseño y reduce costos en aplicaciones IoT (Mohanan, 2022). Por su relación capacidad/consumo y la disponibilidad de un ecosistema de desarrollo maduro lo posicionan como la plataforma de referencia para nodos IoT de bajo costo como el presente proyecto.

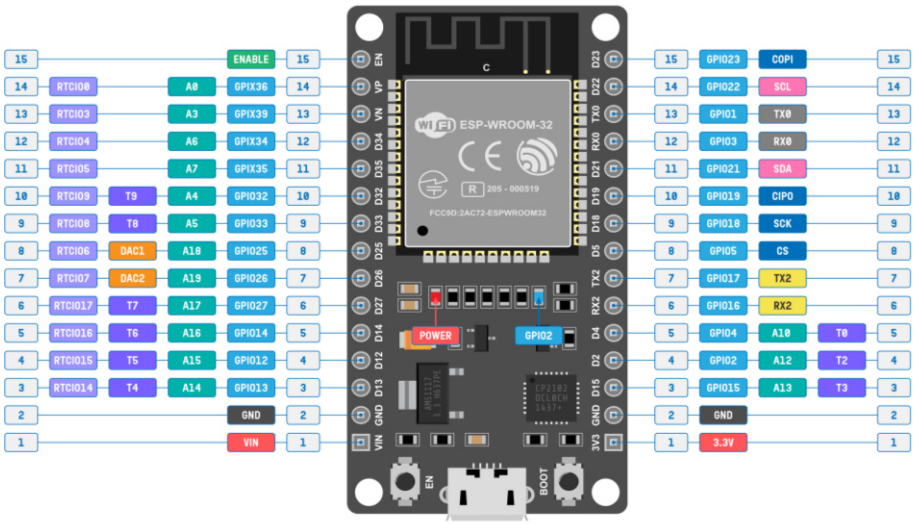


Figura 7 **Diagrama de pines del ESP32-DevKit V.** Fuente. Tomado de DOIT ESP32 DevKit V1 Wi-Fi Development Board – Pinout Diagram & Arduino Reference (Rev. 0.1), por Mohanan (2022), CIRCUITSTATE Electronics.

b) Protocolo de comunicación I²C

El bus I²C (Inter-Integrated Circuit) es un protocolo serial sincrónico de dos hilos SDA y SCL que permite la comunicación entre un maestro y múltiples esclavos mediante direccionamiento de 7 o 10 bits. Opera en modos estándar (100 kbit/s), rápido (400 kbit/s) y de alta velocidad (3,4 Mbit/s), características prescindibles para la elección como protocolo de comunicación en este proyecto. La Figura 8 muestra la secuencia básica de transmisión en el bus I²C, donde se ilustran las condiciones de inicio (START) y parada (STOP), el envío de la dirección del dispositivo junto con el bit de lectura/escritura (R/W), así como la transmisión de datos en bloques de 8 bits seguidos de un bit de reconocimiento (ACK). Esta secuencia define el mecanismo mediante el cual el dispositivo maestro controla la comunicación y sincroniza el intercambio de información con los dispositivos esclavos. (NXP Semiconductors, 2021).

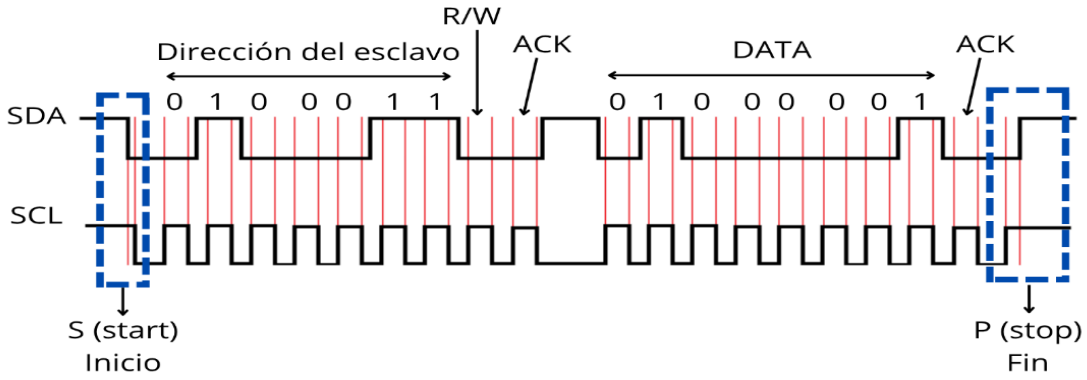


Figura 8 **Secuencia I²C: escritura de un byte al esclavo 0x47 (dato: carácter 'A').** Fuente. *Adaptado de* UM10204: I²C-bus Specification and User Manual *(Rev. 7.0, p. 9), por NXP Semiconductors, 2021.*

c) Multiplexor I²C TCA9548A

El TCA9548A es un conmutador I²C de 8 canales de Texas Instruments que permite conectar hasta ocho buses I²C independientes a un único maestro, resolviendo el

conflicto de direcciones que surge al utilizar múltiples sensores con la misma dirección fija. La selección de canal se realiza mediante escritura en el registro de control del dispositivo, que ocupa la dirección 0x70–0x77 según el estado de sus pines de configuración. La Figura 9 muestra el diagrama funcional del multiplexor I²C TCA9548A donde se observa cómo las líneas principales SCL y SDA ingresan al bloque de control interno, encargado de conmutar la comunicación hacia los ocho canales disponibles, definidos como SC0–SC7 y SD0–SD7. Asimismo, se identifican los pines de alimentación, reinicio y configuración de dirección (VCC, GND, RESET, A0, A1 y A2). Esta estructura permite seleccionar por software el canal I²C requerido, facilitando la conexión de múltiples sensores al mismo bus sin generar conflictos de dirección. (Texas Instruments, 2012)

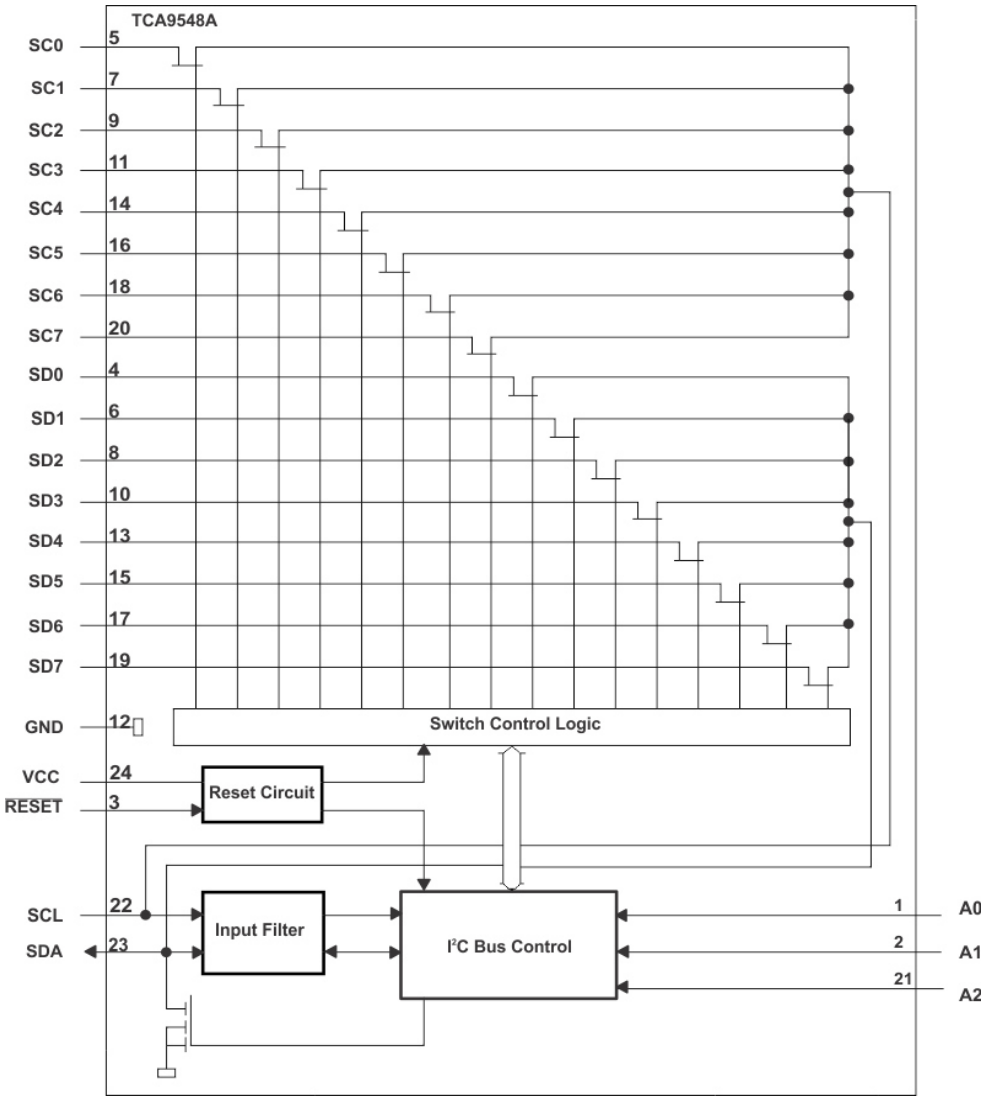


Figura 9 **Diagrama funcional del multiplexor I²C TCA9548A.** Fuente. Tomado de TCA9548A Low-Voltage 8-Channel I²C Switch with Reset Datasheet (SCPS207H), por Texas Instruments (2012).

f) Sensor de temperatura y humedad SHT31

El SHT3x-DIS de Sensirion es un sensor digital de temperatura y humedad relativa que opera sobre bus I²C con una precisión típica de ±0,3 °C en temperatura y ±2 % HR en humedad. Su rango de operación de –40 a +125 °C cubre holgadamente el entorno de un gabinete de centro de datos. La Figura 10 muestra el funcionamiento interno del sensor SHT31. Las señales de temperatura y humedad sensadas ingresan a un conversor ADC de 16 bits y posteriormente son procesadas mediante etapas de calibración, linealización y tratamiento digital de datos. El resultado se entrega al microcontrolador mediante la interfaz I²C, usando las líneas SDA y SCL. Además, se observan los pines de alimentación (VIN y GND), selección de dirección (ADDR), reinicio (RST) y alerta (ALERT), así como el módulo físico del sensor en encapsulado compacto DFN de 2.5 × 2.5 mm. (Sensirion, 2022).

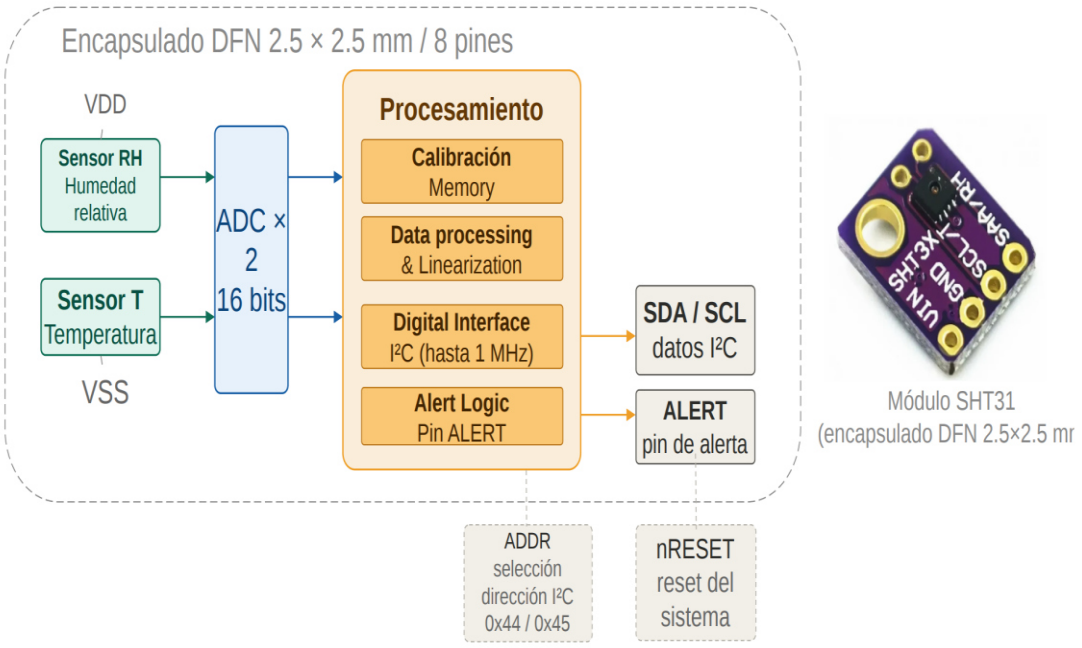


Figura 10 **Diagrama de bloques funcional del sensor SHT3x-DIS.** Fuente. Adaptado de Datasheet SHT3x-DIS (Fig. 1, p. 1), por Sensirion (2022)

g) Sensor de iluminancia BH1750

El BH1750FVI es un sensor digital de luz ambiental con interfaz I²C que mide iluminancia en lux con rango de 1 a 65 535 lx, caracterizado por su bajo consumo e integración sencilla, lo que resulta útil para discriminar condiciones de oscuridad y exposición directa en recintos cerrados. La Figura 11 muestra el funcionamiento interno del sensor BH1750: la luz captada por el fotodiodo (PD) se transforma en una señal eléctrica, que luego es acondicionada por un amplificador integrador (AMP) y convertida a un valor digital mediante un ADC de 16 bits para que finalmente el bloque de control interno transmita la medición al microcontrolador mediante la interfaz I²C, usando las líneas SDA y SCL. En la figura también se observan las conexiones físicas de alimentación (VIN y GND), selección de dirección (ADDR) y el módulo BH1750 en DFN. (ROHM Co., 2011).

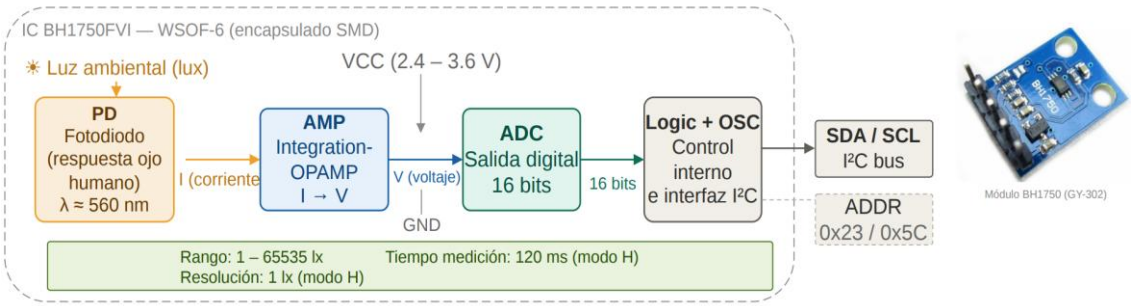


Figura 11 **Diagrama de bloques funcional del sensor de iluminancia BH1750FVI.** Fuente. Adaptado de BH1750FVI Datasheet (Block Diagram, p. 3), por ROHM Co., Ltd., 2011.

h) Sensor de distancia por tiempo de vuelo VL53L0X

El VL53L0X es un sensor de distancia basado en tiempo de vuelo (ToF), diseñado para medir hasta 2 000 mm mediante un emisor VCSEL de 940 nm, reduciendo la influencia del color y la reflectividad del objeto. La Figura 12 resume su funcionamiento interno, en el que el pulso infrarrojo emitido por el VCSEL se refleja sobre el objeto y retorna al arreglo SPAD, permitiendo que el microcontrolador embebido determine la distancia mediante ToF, compense interferencias ópticas internas o crosstalk, y comunique el valor obtenido al sistema a través de la interfaz

I²C por las líneas SDA y SCL. Además, la figura identifica los pines GPIO1 y XSHUT, la dirección I²C 0x52 y las dimensiones físicas del módulo, correspondientes a 4.4 × 2.4 × 1.0 mm utilizado durante el desarrollo posterior en esta investigación .(STMicroelectronics, 2024)

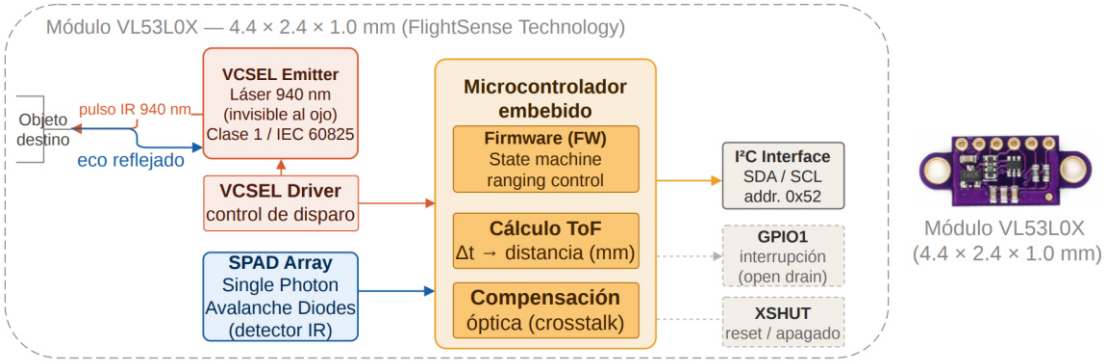


Figura 12 **Diagrama de bloques funcional del sensor de distancia por tiempo de vuelo VL53L0X.** Fuente. Adaptado de VL53L0X Datasheet (Fig. 1, p. 6), por STMicroelectronics (2024).

i) Sensor de medición inercial MPU6050

El MPU-6050 es una unidad de medición inercial que reúne un acelerómetro triaxial, un giroscopio triaxial y un procesador digital de movimiento (DMP) dentro de un encapsulado QFN-24, característica que facilita su uso en sistemas IoT orientados a la detección de vibración y movimiento, así como en aplicaciones que requieren un monitoreo compacto y de bajo consumo energético. La Figura 13 resume su arquitectura interna, en la que las mediciones de los ejes X, Y y Z, junto con la señal del sensor de temperatura, son acondicionadas y convertidas a formato digital mediante ADC de 16 bits, procesadas por el procesador digital de movimiento (DMP) y organizadas mediante registros de configuración, lógica de interrupción y un búfer FIFO de 1024 bytes, lo que permite almacenar temporalmente los datos capturados y reducir la carga de lectura continua sobre el microcontrolador, para finalmente comunicarse con el microcontrolador por la interfaz I²C principal mediante las líneas SDA y SCL, con dirección seleccionable por AD0 como 0x68 u 0x69, lo cual facilita la conexión de múltiples unidades MPU-6050 en un mismo bus sin generar conflictos

de direccionamiento. En la misma figura, también se identifica la salida de interrupción (INT), que permite notificar al microcontrolador la disponibilidad de nuevos datos sin necesidad de sondeo constante, el bus I²C auxiliar (AUX_SDA y AUX_SCL), empleado para la conexión de sensores adicionales como magnetómetros externos, y el encapsulado QFN-24 de 4 × 4 × 0.9 mm, dimensiones que favorecen su integración en dispositivos de reducido tamaño físico (InvenSense Inc., 2013)

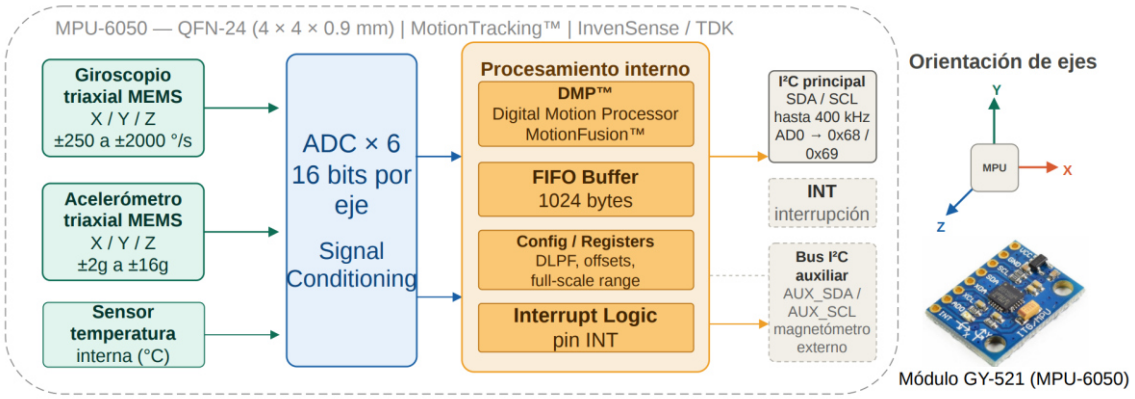
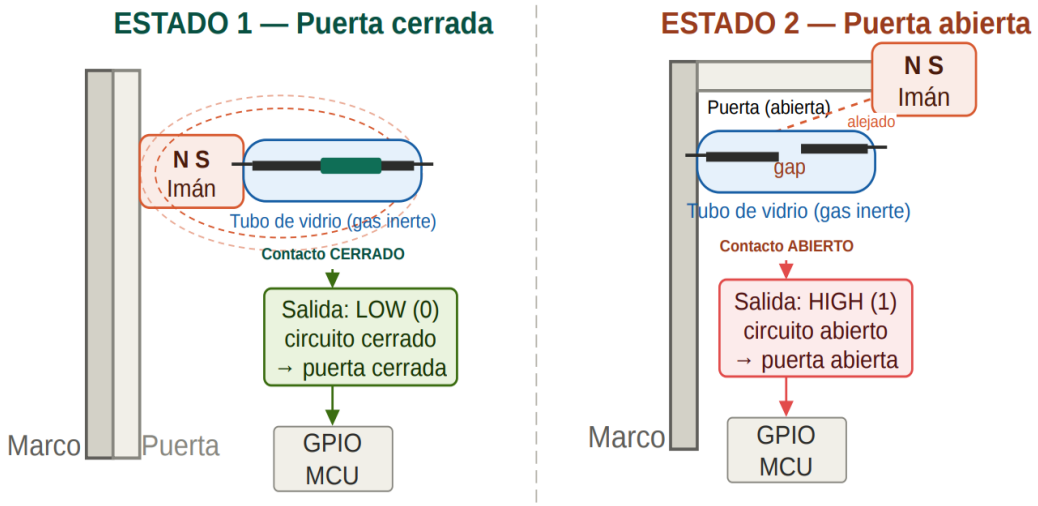


Figura 13 **Diagrama de bloques funcional de la unidad de medición inercial MPU-6050.** Fuente. Adaptado de MPU-6000 and MPU-6050 Product Specification (Rev. 3.4, p. 7), por InvenSense Inc. (2013).

j) Reed switch como sensor de estado

Un reed switch es un interruptor magnético cuyo estado depende del campo producido por un imán próximo, lo que le permite operar al sensor sin necesidad de contacto físico directo entre las partes que detecta. Debido a su salida binaria, bajo consumo y ausencia de partes mecánicas expuestas, se emplea como una solución práctica y de bajo costo para supervisar la apertura y cierre de puertas en gabinetes, especialmente en entornos donde se busca minimizar el desgaste y la necesidad de mantenimiento frecuente como es el caso en esta investigación. Figura 14 describe de manera visual la operación del sensor reed en el sistema, donde la puerta cerrada mantiene el imán cerca del sensor, conserva los contactos internos cerrados por efecto del campo magnético y produce una señal LOW (0) en el GPIO, mientras que la apertura de la puerta aleja el imán, elimina la influencia del campo sobre los contactos,

provoca su apertura y cambia la salida a HIGH (1). Este comportamiento binario permite que el microcontrolador interprete de forma inmediata y confiable el estado físico de la puerta sin requerir procesamiento adicional. La tabla inferior a la Figura 14 resume la relación entre el estado de la puerta, la presencia del campo magnético, el estado de los contactos reed y la salida digital entregada al microcontrolador utilizado, consolidando de forma clara la lógica de funcionamiento del sensor ante los distintos escenarios posibles en el desarrollo del sistema propuesto en esta investigación (Standex Detect, 2025).



Estado de puerta	Campo magnético	Contactos reed	Salida GPIO
Cerrada	Presente (imán próximo)	Cerrados (NO)	LOW (0)
Abierta	Ausente (imán alejado)	Abiertos	HIGH (1)

Figura 14 **Principio de operación del reed switch como sensor de estado de puerta en gabinete.** Fuente. Elaboración propia basada en el principio de operación descrito por Standex Detect (2025).

i) Tira LED direccionable SMD5050

El LED RGB SMD5050 integra tres emisores independientes para los colores rojo, verde y azul dentro de un encapsulado compacto de 5.0 × 5.0 × 1.6 mm, lo que permite lograr una amplia gama de colores mediante la combinación de sus canales sin requerir componentes adicionales de mezcla óptica. Su configuración de cátodo

común, junto con entradas separadas para cada canal, permite controlar la intensidad de cada color mediante señales PWM, lo que facilita su uso como indicador visual de estado en sistemas embebidos e interfaces físicas compactas, siendo especialmente útil para representar de forma inmediata distintos niveles de alerta o condiciones operativas del sistema. La Figura 15 presenta la estructura superior y el diagrama de conexión eléctrica del componente, donde se identifican los emisores R, G y B con sus respectivas longitudes de onda, información relevante para calibrar la respuesta cromática percibida por el usuario. En el diagrama de conexión se observa que cada canal posee una entrada independiente, lo que permite su activación individual o combinada según la lógica de control implementada, mientras que todos comparten un cátodo común (GND), simplificando el cableado hacia el microcontrolador. Además, la figura muestra el encapsulado SMD de 5.0 × 5.0 mm, cuyas reducidas dimensiones favorecen su montaje directo en placas de circuito impreso de espacio limitado. (Everlight Electronics, 2023).

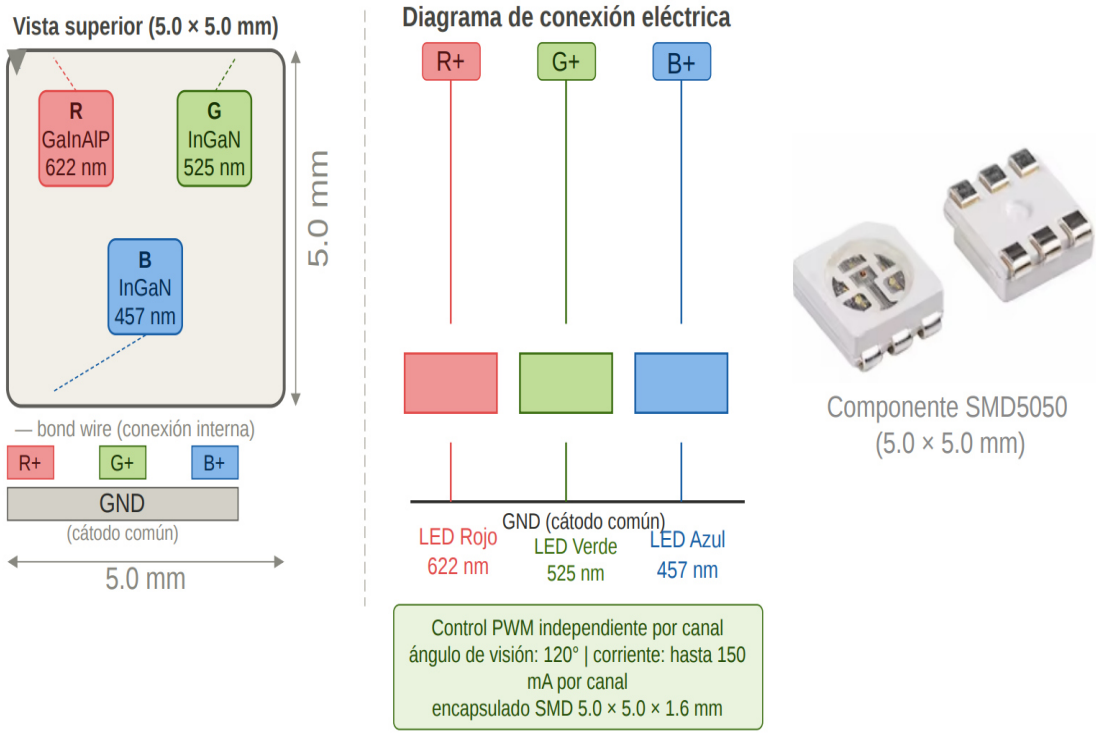


Figura 15 **Estructura interna y diagrama de conexión del LED SMD5050 RGB.** Fuente. Elaboración propia basada en las especificaciones descritas en 5050 Full Color SMT LED Datasheet, por Everlight Electronics (2023).

j) Estándar de rack EIA-310-D (19")

La especificación EIA-310-D establece la normalización dimensional de las estructuras mecánicas de 482,6 mm (19 pulgadas) empleadas en equipos electrónicos y de telecomunicaciones. Esta estandarización asegura la compatibilidad física entre gabinetes, bandejas, paneles y equipos montables en rack, constituyendo la referencia vigente para la integración mecánica de soluciones en salas de comunicaciones y centros de datos. La Figura 16 presenta de manera gráfica todas estas dimensiones de, mostrando la relación entre el patrón de orificios y las medidas principales necesarias para la compatibilidad mecánica del rack que se utilizará como parte de las pruebas físicas de esta investigación, información que resulta útil al momento de diseñar o adaptar componentes destinados a integrarse físicamente dentro de este tipo de estructuras (Intel Corporation, 2005).

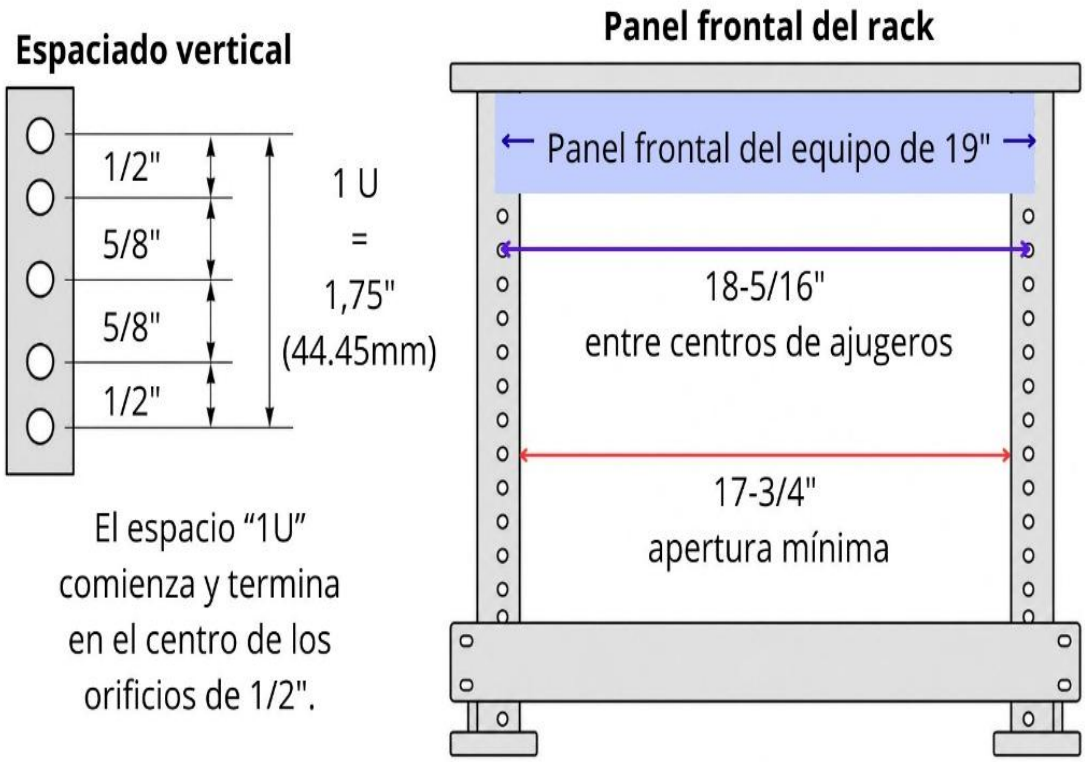


Figura 16 **Dimensiones de referencia para rack de 19 pulgadas según EIA-310-D**. Fuente. Elaboración propia basada en las dimensiones de compatibilidad para gabinetes de servidores descritas por Intel Corporation *(2005)*.

k) Regulación de tensión en sistemas embebidos digitales

En sistemas embebidos digitales, la estabilidad de la tensión de alimentación es un factor determinante para la operación confiable de los componentes activos. Los reguladores conmutados de topología buck permiten obtener niveles de tensión estables a partir de una fuente primaria con alta eficiencia de conversión. El regulador LM2596 empleado en el presente sistema garantiza una tolerancia máxima de ±5% sobre la tensión de salida bajo condiciones especificadas de tensión de entrada y carga (Texas Instruments, 2023)

Este margen fue verificado como compatible con los rangos de alimentación declarados por los fabricantes de todos los componentes del sistema, incluyendo el microcontrolador ESP32-WROOM-32 (Espressif Systems, 2025), el multiplexor TCA9548A (Texas Instruments, 2012), los sensores SHT31 (Sensirion, 2022), BH1750 (ROHM Co., 2011), VL53L0X (STMicroelectronics, 2024) y MPU6050 (InvenSense Inc., 2013), todos los cuales admiten variaciones superiores a dicho margen sobre su valor nominal de operación.

II.3.2 Protocolos de comunicación

a) Protocolo MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligera basado en el patrón publicador/suscriptor y orientado a intercambios eficientes sobre TCP/IP en escenarios con recursos limitados o conectividad variable. Su funcionamiento mediado por un broker y la disponibilidad de niveles de calidad de servicio lo han consolidado como uno de los protocolos más utilizados en soluciones IoT e IIoT contemporáneas (Radwan & Alves-Foss, 2024).

***b)* Protocolo WiFi 802.11 y modo punto de acceso (AP)**

El estándar IEEE 802.11-2020 consolida las especificaciones para redes de área local inalámbricas (WLAN) en las bandas de 2,4 GHz y 5 GHz. En la arquitectura implementada, el Gateway IoT opera como punto de acceso WiFi mediante la interfaz uap0 y el ESP32 funciona en modo estación (STA), conectándose automáticamente a

la red dedicada del gabinete. Esta separación permite mantener el enlace de telemetría local independiente de la red LAN principal.(IEEE, 2021)

c) Protocolo SNMP

El Protocolo Simple de Gestión de Red (SNMP) es un estándar ampliamente utilizado para supervisar y administrar dispositivos de red mediante el intercambio de mensajes de consulta y respuesta sobre UDP. Su modelo de operación se basa en agentes, estaciones gestoras y una base de información de gestión (MIB), lo que permite recuperar variables operativas como el estado de las interfaces, el uso de recursos y las condiciones generales del equipo; en sus versiones más recientes, además, incorpora mecanismos de autenticación y protección del intercambio de información. Asimismo, permite generar notificaciones automáticas mediante mensajes trap o inform, facilitando la detección oportuna de eventos sin depender exclusivamente de consultas periódicas. Debido a su interoperabilidad y bajo consumo de recursos, SNMP es especialmente adecuado para integrarse en plataformas centralizadas de monitoreo y supervisión remota (Espinel et al., 2022)

II.3.3 Plataforma Gateway IoT

a) Raspberry Pi 4 Modelo B como plataforma embebida

La Raspberry Pi 4 Modelo B es una computadora de placa única (SBC) basada en el sistema en chip (System-on-chip, SoC) Broadcom BCM2711 con CPU ARM Cortex-A72 de cuatro núcleos a 1,8 GHz, disponible en configuraciones de 2, 4 u 8 GB de RAM LPDDR4. El SoC integra un procesador gráfico VideoCore VI y un subsistema PCIe Gen 2 que habilita los puertos USB 3.0 de alta velocidad. Como se muestra en la Figura 17, muestra el conector J8 de 40 pines de la Raspberry Pi, donde se identifican las líneas de alimentación (3V3 y 5V), tierra (GND) y pines de propósito general (GPIO) para la conexión de periféricos. También se observan los pines ID_SD e ID_SC, reservados para la detección automática de módulos HAT mediante EEPROM, por lo que no se recomiendan para uso general. Esta distribución permite integrar sensores, actuadores y buses de comunicación como I²C, SPI o UART dentro del nodo gateway (Raspberry Pi Ltd, 2024). Su capacidad de ejecutar Linux completo la convierte en la plataforma idónea para el nodo gateway del presente sistema.

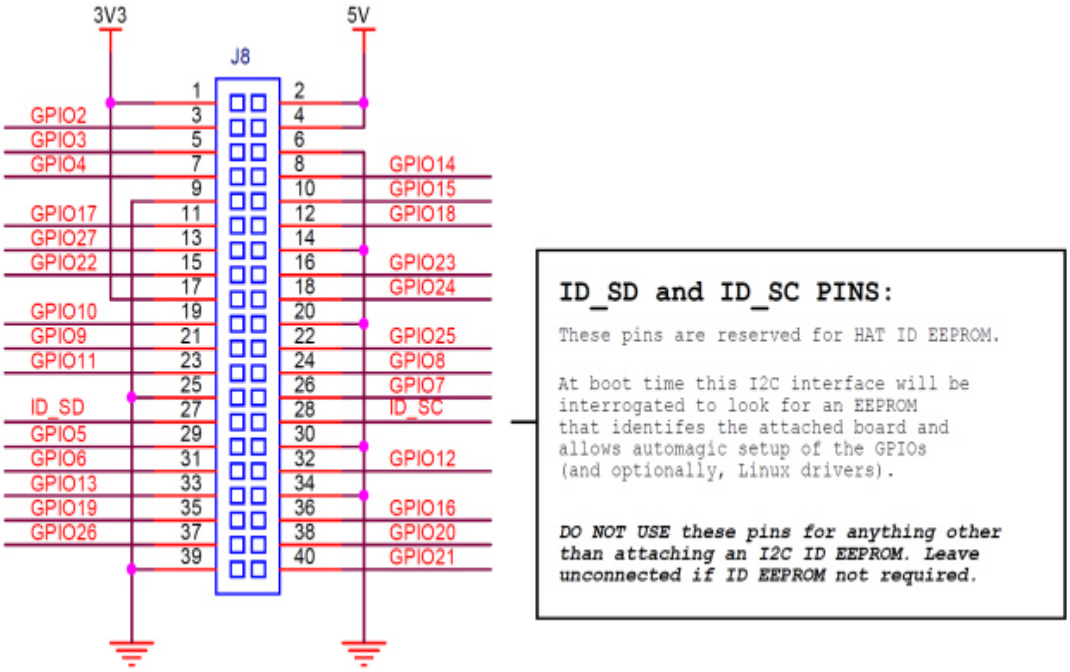


Figura 17 **Diagrama esquemático del conector GPIO J8 de 40 pines de la Raspberry Pi 4 Model B**. Fuente. Adaptado de Raspberry Pi Ltd (2024).

b) Broker MQTT Mosquitto

Eclipse Mosquitto es uno de los brokers MQTT de código abierto más difundidos en entornos IoT por su ligereza, facilidad de despliegue y compatibilidad con clientes y dispositivos heterogéneos. Su adopción resulta especialmente conveniente en plataformas de borde o gateways de recursos moderados, donde se requiere una capa de mensajería estable y de bajo consumo operativo (Radwan & Alves-Foss, 2024)

c) hostapd y dnsmasq

hostapd es un demonio de espacio de usuario para Linux que implementa los modos de punto de acceso IEEE 802.11 y autenticador IEEE 802.1X, permitiendo que una interfaz WiFi actúe como AP sin hardware dedicado (Malinen, 2024). dnsmasq es un servidor ligero de DNS y DHCP ampliamente utilizado en sistemas embebidos para resolver nombres locales y asignar direcciones IP a los clientes de la red (Kelley, 2025). La combinación de ambas herramientas implementa un punto de acceso WiFi autónomo sobre Linux con mínimos recursos del sistema.

II.3.4 Software para monitoreo y control en el Gateway IoT

a) Framework Flask (Python)

Flask es un microframework para Python basado en WSGI que privilegia un núcleo ligero y extensible, adecuado para construir APIs y aplicaciones web modulares. Su enfoque minimalista, junto con el uso de módulos independientes (blueprints) y un ecosistema maduro de extensiones, lo vuelve especialmente apropiado para backend de tamaño medio con necesidad de control explícito de rutas, lógica y servicios utilizados durante el desarrollo de software en esta investigación (Grinberg, 2018).

b) React como librería frontend

React es una biblioteca JavaScript orientada a la construcción de interfaces de usuario mediante componentes reutilizables y composición declarativa. Su enfoque basado en estado y árbol de componentes favorece el desarrollo de frontends modulares y mantenibles; en sus versiones recientes, además, incorpora mecanismos que mejoran la respuesta percibida en interfaces con actualizaciones frecuentes (Santana Roldan, 2023).

c) MySQL como sistema de gestión de bases de datos relacionales

MySQL es un sistema de gestión de bases de datos relacionales que implementa el modelo SQL y utiliza InnoDB como motor transaccional por defecto. En la rama 8.0 ofrece soporte para transacciones ACID, claves foráneas, roles, tipos JSON e indexación avanzada, características pertinentes para el almacenamiento estructurado de telemetría y eventos operativos que resultan de suma importancia para el desarrollo de esta investigación en el apartado de desarrollo de software (Oracle Corporation, 2026).

d) paho-mqtt como cliente MQTT en Python

Paho MQTT es una biblioteca cliente ampliamente utilizada en Python para integrar aplicaciones con brokers MQTT mediante un modelo de callbacks y bucles de red síncronos o asíncronos. Su uso es frecuente en implementaciones de pasarela, adquisición y mensajería IoT por la sencillez con que permite publicar, suscribirse y gestionar sesiones MQTT desde procesos Python (Light, 2024).

e) pysnmp para consultas SNMP en Python

PySNMP es una biblioteca para Python empleada en tareas de supervisión de red y automatización, al ofrecer soporte para operaciones SNMP sobre las versiones 1, 2c y 3 del protocolo. Su utilidad práctica radica en que permite construir rutinas de consulta, sondeo e integración con sistemas de monitoreo desde aplicaciones Python sin depender de herramientas externas complejas (Etingof et al., 2026).

II.3.5 Infraestructura en la nube

a) Cloudflare: DNS, proxy inverso y tunneling

Cloudflare actúa en esta arquitectura como servicio de resolución y publicación segura de aplicaciones, mientras que el acceso remoto al backend se implementa mediante un túnel saliente cifrado desde la infraestructura local hacia el perímetro externo. Desde el punto de vista arquitectónico, este esquema responde al patrón de publicación de aplicaciones privadas mediante conectores salientes, evitando la exposición directa de la red local y reduciendo la necesidad de abrir puertos entrantes en el firewall (National Institute of Standards and Technology, 2023).

b) Azure Static Web Apps

Azure Static Web Apps es un servicio administrado orientado al despliegue de aplicaciones web estáticas y frontends compilados en la nube pública del fabricante Microsoft, con integración nativa con flujos de automatización y distribución global del contenido. Su adopción resulta adecuada para aplicaciones desarrolladas con React debido a que simplifica el ciclo de compilación, publicación y exposición segura del frontend en Internet como es el caso de esta investigación a nivel de desarrollo de software frontend (Cashmore, 2026).

c) Integración y despliegue continuos (CI/CD)

La integración continua (CI) comprende la automatización de tareas de compilación, verificación y validación del software cada vez que se incorporan cambios al repositorio, mientras que el despliegue continuo (CD) extiende dicho flujo hacia la entrega automatizada de artefactos a entornos de prueba o producción. En proyectos modernos, plataformas como GitHub Actions permiten materializar este

proceso mediante flujo de trabajo reproducibles, versionados y vinculados al repositorio del sistema (Faqih et al., 2024).

II.3.6 Aprendizaje Automático

a) Aprendizaje automático no supervisado

El aprendizaje automático no supervisado comprende los algoritmos que identifican estructura, regularidades o agrupamientos en datos sin recurrir a etiquetas previas de clase. En contextos de monitoreo, este enfoque permite modelar el comportamiento normal del sistema a partir de observaciones históricas y detectar desviaciones relevantes aun cuando no se dispone de registros etiquetados de falla, situación habitual en entornos reales de infraestructura (Geron, 2022;Chatterjee & Ahmed, 2022).

b) Isolation Forest como algoritmo de detección de anomalías

Isolation Forest es un método no supervisado de detección de anomalías que parte de la idea de que las observaciones anómalas suelen aislarse con menos particiones aleatorias que los datos normales. Su aplicación en sistemas IoT y redes de sensores resulta atractiva por su bajo costo computacional, su escalabilidad y su buen desempeño sobre conjuntos de datos multivariados, por lo que es una alternativa pertinente para plataformas de borde con recursos limitados (Xu et al., 2023).

c) Scikit-learn como biblioteca de machine learning en Python

Scikit-learn es una biblioteca de aprendizaje automático para Python que reúne implementaciones consolidadas de algoritmos de clasificación, regresión, agrupamiento y detección de anomalías bajo una API homogénea. En la práctica, facilita el entrenamiento, la evaluación y el despliegue de modelos como Isolation Forest dentro de flujos de trabajo reproducibles sobre datos tabulares y temporales. Además, integra herramientas para el preprocesamiento de datos, la selección de características y la validación del rendimiento mediante métricas estandarizadas. Su compatibilidad con otras bibliotecas del ecosistema científico de Python permite incorporarla fácilmente en aplicaciones de análisis y monitoreo automatizado (Geron, 2022).

II.4 Definición de Términos

A continuación, se presentan las definiciones de los términos técnicos empleados a lo largo de esta investigación. Los términos cuyo desarrollo conceptual profundo se encuentra en el Marco Teórico o en las Bases Teóricas son referenciados brevemente aquí para evitar repetición; la remisión explícita a la sección correspondiente orienta al lector hacia el tratamiento completo.

- Anomalía. Valor o secuencia que se aparta del comportamiento normal del sistema. Puede ser puntual, contextual o colectiva, siendo estas últimas relevantes en monitoreo de infraestructura (Cook et al., 2020)
- Broker MQTT. Componente que recibe mensajes publicados por los clientes MQTT y los distribuye a los suscriptores del tópico correspondiente. En este sistema se usa Eclipse Mosquitto (Radwan & Alves-Foss, 2024).
- Centro de datos Tier 1. Infraestructura con disponibilidad anual de 99,671 %, sin redundancia de componentes críticos ni rutas alternativas (Uptime Institute, 2022)
- Firmware. Programa embebido que controla el hardware, adquiere datos y gestiona comunicaciones. En la UAC se ejecuta sobre el ESP32.
- Gabinete o rack de servidor. Estructura normalizada de 19 pulgadas para alojar equipos electrónicos y de comunicaciones. Su unidad de altura estándar es 1U = 44,45 mm (Intel Corporation, 2005).
- Gateway IoT. Nodo de borde que concentra datos de dispositivos de campo, ejecuta procesamiento local y comunica el sistema con la aplicación o la nube (Avasalcai et al., 2020)
- Latencia de publicación MQTT. Tiempo entre la lectura del sensor y la entrega del mensaje al broker MQTT; mide la respuesta del sistema de transmisión.
- Unidad de Adquisición y Control (UAC). Módulo basado en ESP32 que integra sensores I^2C/GPIO, actuadores RGB y transmisión MQTT hacia el Gateway IoT. (Definición de elaboración propia)
- Arquitectura de Confianza Cero. Enfoque de seguridad que exige autenticación y autorización explícita en cada acceso, sin confiar en la ubicación de red (Rose et al., 2020).

CAPÍTULO III:
HIPÓTESIS Y VARIABLES

III.1 Hipótesis

III.1.1 Hipótesis general

El Sistema Inteligente de Monitoreo implementado, que integra tecnologías de sensado, comunicación distribuida, un Gateway IoT y técnicas de aprendizaje automático en una arquitectura híbrida (nube-local), alcanza un desempeño que satisface los indicadores técnicos de supervisión y gestión remota de gabinetes en centros de datos Tier 1, evidenciando capacidad de detección temprana de fallas y operación continua.

III.1.2 Hipótesis específicas

- El hardware basado en ESP32, con sensores conectados por multiplexores I^2C y pines GPIO, logra la adquisición de variables ambientales en gabinetes Tier 1: tasa de éxito del 100 % en los ciclos de lectura y tensión de alimentación dentro de ±4 % del valor nominal.
- El firmware del ESP32 opera como nodo de adquisición, control y transmisión MQTT, con latencia de publicación menor a 500 ms y reconexión automática menor a 5 s ante pérdida de conectividad.
- El modelo Isolation Forest integrado en la Raspberry Pi identifica patrones anómalos en las condiciones del gabinete, con recall superior al 95 %, F1-score superior al 90 % y tiempo de inferencia menor a 5 s.
- El Gateway IoT basado en Raspberry Pi centraliza la recepción, el almacenamiento y la sincronización de la información de la UAC, con latencia del broker MQTT y

de sincronización con la base de datos menor a 500 ms, tasa de entrega del 100 % y servicios persistentes.

- La plataforma de software permite gestionar usuarios, visualizar variables en tiempo real y controlar remotamente los gabinetes, con actualización del panel de monitoreo menor a 5 s y entrega de alertas menor a 60 s.
- La infraestructura híbrida que combina el entorno on-premise con Azure y Cloudflare Tunnel permite la accesibilidad remota con certificado SSL válido.

III.2 Variables

III.2.1 Variables independientes

La variable independiente es el Sistema Inteligente de Monitoreo para Gabinetes en Centros de Datos Tier 1, entendido como la propuesta tecnológica que constituye la intervención de esta investigación. El sistema se estructura en seis dimensiones de implementación: Hardware UAC, Firmware, Modelo de Aprendizaje Automático, Gateway IoT, Plataforma de Software e Infraestructura Híbrida. El diseño, la implementación y la configuración de estas seis dimensiones constituyen los componentes que se someten a evaluación en la presente investigación; su detalle se presenta en la tabla de operacionalización. Cada dimensión comprende indicadores específicos orientados a verificar su funcionamiento, nivel de integración y aporte al cumplimiento de los objetivos planteados. En conjunto, su evaluación permite determinar la capacidad del sistema para realizar el monitoreo autónomo, detectar anomalías y facilitar la gestión remota de los gabinetes (Tabla 2, sección III.3).

III.2.2 Variables dependientes

La variable dependiente es la Supervisión y gestión remota de gabinetes en centros de datos Tier 1, entendida como la capacidad efectiva del operador de observar, controlar y auditar el estado del gabinete desde cualquier ubicación con acceso a

Internet, en tiempo real y de forma continua. Los indicadores específicos de cada dimensión se detallan en la Tabla 2, sección III.3.

III.2.3 Variables externas

Factores externos mantenidos constantes durante la evaluación: condiciones de la red LAN, tipología y modelo de sensores, características físicas del gabinete y configuración de los servicios de nube utilizados

III.3 Operacionalización de variables

En la Tabla 2 se presenta el detalle de la operacionalización de las variables que serán observadas y medidas dentro del presente estudio

Tabla 2 **Operacionalización de las variables de la investigación**

Variable	Dimensión	Indicador
VI: Sistema Inteligente de Monitoreo	Hardware UAC	Tasa de éxito de lectura I²C/GPIO (%); tensión de alimentación respecto al valor nominal (%).
	Firmware	Latencia de publicación MQTT (ms); tiempo de reconexión ante pérdida de conectividad (s).
	Modelo ML	Recall (%); F1-score (%); tiempo de inferencia por muestra (ms).
	Gateway IoT	Latencia del broker MQTT y de sincronización con la base de datos (ms); tasa de entrega (%).
	Plataforma de software	Tiempo de actualización del panel de monitoreo (s); tiempo de entrega de alertas (s).

	Infraestructura híbrida	Validez del certificado SSL en el acceso remoto vía túnel Cloudflare.
VD: Supervisión y gestión remota de gabinetes en centros de datos Tier 1	Desempeño de la plataforma web	Visualización en tiempo real de variables del gabinete mediante dashboard web
		Recepción de alertas automáticas ante condiciones anómalas
		Consulta del historial de eventos y accesos desde la interfaz web
	Continuidad del acceso híbrido	Acceso remoto seguro a la plataforma desde cualquier ubicación
		Auditoría de sesiones de acceso remoto al sistema

Fuente. Elaboración propia

CAPÍTULO IV:
MATERIALES Y MÉTODOS

IV.1 Área de estudio

El presente trabajo de investigación se desarrolló en dos entornos complementarios. El desarrollo e integración del sistema se llevó a cabo en un entorno de prueba controlado donde se replicó la operación simultánea de todos los sensores y componentes del sistema. La validación final se realizó en las instalaciones de la empresa Cloud IT Perú S.A.C. (Av. Petit Thouars 1775, Oficina 1306, Edificio El Almirante, distrito de Lince, Lima, Perú; Latitud: -12.085, Longitud: -77.033), empresa especializada en infraestructura de centros de datos y telecomunicaciones. El sistema fue conectado a un gabinete en operación continua que aloja equipos de red y seguridad activos, donde operó durante siete días consecutivos recolectando datos reales sin interferir con la operación del entorno productivo.

IV.2 Diseño de investigación

IV.2.1 Tipo de investigación

La investigación es de tipo aplicada, dado que su propósito es resolver un problema técnico concreto mediante el diseño e implementación de un sistema tecnológico, apoyándose en conocimientos del campo del IoT, los sistemas embebidos y el aprendizaje automático. Su desarrollo se orienta a transformar estos fundamentos teóricos en una solución funcional que pueda ser implementada en gabinetes de centros de datos Tier 1. Asimismo, busca generar resultados prácticos y verificables relacionados con la supervisión ambiental, la detección de anomalías y el acceso remoto a la información del sistema (Hernandez-Sampieri & Mendoza Torres, 2018).

IV.2.2 Nivel de investigación

El nivel es descriptivo, dado que su propósito es caracterizar y medir el desempeño del sistema implementado bajo condiciones de operación definidas, documentando el valor alcanzado por cada indicador técnico establecido en el Capítulo III, sin analizar relaciones de causalidad entre variables. En consecuencia, las hipótesis formuladas son de carácter descriptivo: pronostican los valores o estados esperados de los indicadores del sistema (Hernandez-Sampieri & Mendoza Torres, 2018).

IV.2.3 Enfoque metodológico

El enfoque es cuantitativo, orientado a la medición objetiva de variables técnicas mediante instrumentos estandarizados. Los datos recolectados tales como: latencia de transmisión, precisión de adquisición, tasa de detección de anomalías y disponibilidad del sistema, son numéricos y analizados mediante estadística descriptiva para evaluar el cumplimiento de los indicadores definidos (Hernandez-Sampieri & Mendoza Torres, 2018).

IV.2.4 Diseño de la investigación

El diseño adoptado es no experimental, dado que no se manipulan deliberadamente las variables independientes, sino que se observa y mide el comportamiento del sistema en condiciones controladas que simulan el entorno real de operación. Las variables técnicas son registradas tal como se presentan durante las sesiones de prueba estructuradas, sin introducir modificaciones al sistema durante el proceso de medición (Hernandez-Sampieri & Mendoza Torres, 2018).

IV.3 Población y muestra

La unidad de análisis del presente estudio es el gabinete de centro de datos como entorno físico de monitoreo. La población está conformada por gabinetes de

centros de datos que satisfacen los requerimientos mínimos de la categoría Tier I según el estándar TIA-942, caracterizados por su arquitectura básica sin sistemas redundantes. Para efectos de validación, la muestra estuvo conformada por el gabinete en operación de la empresa Cloud IT Peru S.A.C., el cual cumple dichas características y constituyó el entorno idóneo para la adquisición, el monitoreo y la validación de los indicadores técnicos propuestos.

IV.4 Procedimiento, técnicas e instrumentos de recolección de información

La recolección de datos se realizó mediante dos técnicas complementarias aplicadas sobre cada bloque funcional del sistema: la observación directa estructurada y las pruebas de funcionamiento técnico. La primera consistió en el seguimiento sistemático del comportamiento del sistema durante las sesiones de prueba, registrando eventos de detección de anomalías, activación de alertas y respuesta ante escenarios de fallo controlados. La segunda comprendió pruebas cuantitativas sobre cada bloque para evaluar parámetros de desempeño mediante procedimientos estandarizados y repetibles. Los bloques evaluados, los instrumentos empleados y las variables medidas se presentan en la Tabla 4.

IV.4.1 Etapas de la investigación

El desarrollo del sistema se organiza en seis etapas secuenciales que siguen un modelo incremental por capas, donde cada etapa construye sobre los resultados de la anterior. Este enfoque permite verificar el funcionamiento de cada componente de manera independiente antes de su integración con el conjunto, reduciendo el riesgo de errores sistémicos. La Tabla 3 presenta el resumen de las etapas, su componente principal, el resultado esperado y el mecanismo de validación correspondiente. En la Etapa 3, el término “detección” se emplea en su sentido técnico de identificación de condiciones anómalas en el estado actual del sistema, y no como sinónimo de predicción o pronóstico de fallas futuras.

Tabla 3 **Resumen de etapas de implementación del sistema**

Etapa	Componente principal	Resultado esperado	Validación
1	Hardware de la UAC basado en ESP32	Módulo de sensado integrado en chasis rack 1U capaz de adquirir variables ambientales y controlar actuadores	Pruebas de lectura de sensores y respuesta de actuadores
2	Firmware y protocolo MQTT	Comunicación bidireccional estable entre el módulo IoT y el broker con transmisión de datos en formato JSON	Verificación de publicación y suscripción de mensajes en tópicos definidos
3	Modelo de aprendizaje automático	Modelo Isolation Forest entrenado capaz de identificar condiciones ambientales anómalas en el gabinete	Evaluación de métricas de detección sobre datos de validación
4	Gateway IoT (Raspberry Pi)	Nodo central operando con broker MQTT, servicio SNMP y punto de acceso WiFi de forma simultánea y persistente	Verificación de servicios activos y consulta SNMP a equipos registrados
5	Plataforma de software (backend y frontend)	Sistema web funcional con dashboards, alertas, control de actuadores y gestión de usuarios	Pruebas de endpoints, flujo MQTT y despliegue del frontend
6	Infraestructura híbrida	Acceso remoto seguro al sistema desde Internet mediante Cloudflare Tunnel y Azure Static Web Apps	Verificación de acceso externo, certificado SSL y despliegue continuo

Fuente. Elaboración propia

IV.4.2 Técnicas e instrumentos por bloque del sistema

Se ejecutaron pruebas controladas sobre cada bloque del sistema para evaluar parámetros cuantitativos de desempeño mediante procedimientos estandarizados y repetibles. Los bloques evaluados, los instrumentos empleados y las variables medidas se presentan en la Tabla 4.

Tabla 4 **Técnicas e instrumentos de recolección por bloque del sistema**

Bloque	**Técnica**	**Instrumento / Indicador**	**Variable evaluada**
Hardware (ESP32 + sensores)	Prueba de funcionamiento técnico	Lectura de sensores SHT31, VL53L0X y MPU6050 comparadas con instrumentos de referencia	Precisión de adquisición de datos
Comunicación MQTT	Observación directa estructurada	Registro de mensajes publicados y recibidos en el broker con marca temporal	Confiabilidad y latencia de transmisión
Software (backend y frontend)	Prueba de funcionamiento técnico	Verificación de endpoints API, visualización de paneles de control y tiempo de actualización de la interfaz	Disponibilidad y tiempo de respuesta del sistema
Infraestructura en la nube	Observación directa estructurada	Acceso remoto al sistema desde red externa, verificación de certificado SSL y despliegue continuo	Disponibilidad del acceso remoto

Modelo ML (Isolation Forest)	Prueba de funcionamiento técnico	Evaluación de métricas de detección: tasa de verdaderos positivos, falsos positivos y tiempo de inferencia	Precisión del modelo de detección de anomalías

Fuente. Elaboración propia

En las etapas de hardware, firmware, Gateway IoT, plataforma de software e infraestructura híbrida, las hipótesis establecen criterios técnicos de aceptación definidos previamente; por ello, los resultados se resumieron en la Tabla 5 mediante estadística descriptiva y se compararon directamente con el umbral o condición correspondiente. En estos casos no se busca comparar poblaciones o distribuciones, sino verificar el cumplimiento de una especificación de desempeño. Para el modelo Isolation Forest, en cambio, se compararon las distribuciones de puntajes de las clases normal y anómala; por esta razón se evaluó la normalidad con Shapiro-Wilk y, al no cumplirse, se aplicó la prueba U de Mann-Whitney, complementada con una medida del tamaño del efecto.

Las pruebas de Shapiro-Wilk, Mann-Whitney y el tamaño del efecto se aplicaron únicamente en la Etapa 3 (sección V.1.3), correspondiente a la evaluación del modelo Isolation Forest. Las demás etapas se validaron mediante estadística descriptiva y comparación directa de la media observada con el criterio técnico fijado para cada hipótesis.

Tabla 5. **Técnicas estadísticas aplicadas para el análisis y validación de resultados.**

Técnica	Qué cuantifica	Aplicación en esta investigación
Media aritmética	Valor central de un conjunto de mediciones.	Valor de referencia de cada indicador de desempeño (latencias, tasas de éxito, tiempo de inferencia) en las seis etapas,

Técnica	Qué cuantifica	Aplicación en esta investigación
		contrastado contra el umbral de su hipótesis específica.
Mediana	Valor central robusto ante asimetría o valores atípicos.	Estadístico del score del modelo Isolation Forest (Etapa 3, hipótesis específica 3), cuya distribución presenta asimetría marcada.
Desviación estándar y varianza	Dispersión de los datos respecto a la media.	Reportadas junto a la media en las mediciones con muestreo repetido; en la Etapa 1 se emplearon para describir la variabilidad de las tensiones medidas bajo las condiciones de carga evaluadas.
Cuartiles y rango intercuartílico (IQR)	Dispersión robusta y forma de la distribución.	Estadísticos del score de anomalía por clase (Etapa 3) y base del diagrama de caja usado para visualizar la separación entre clases normal y anómala.
Prueba de normalidad (Shapiro-Wilk)	Determina si una distribución se ajusta a una distribución normal.	Aplicada exclusivamente sobre los scores del Isolation Forest (Etapa 3) para evaluar el supuesto de normalidad antes de comparar las clases normal y anómala.
Prueba U de Mann-Whitney	Compara dos distribuciones independientes sin asumir normalidad.	Aplicada exclusivamente a los scores de las clases normal y anómala del Isolation Forest (Etapa 3), al no cumplirse el supuesto de normalidad.
Tamaño del efecto (d de Cohen)	Magnitud de una diferencia entre grupos,	Cuantifica la magnitud de la separación observada entre las clases normal y anómala del Isolation Forest

Técnica	Qué cuantifica	Aplicación en esta investigación
	independiente del tamaño de muestra.	(Etapa 3), complementando la prueba de significancia.

Fuente. Elaboración propia.

IV.4.3 Calibración de instrumentos

Los sensores integrados en el hardware UAC operan dentro de los rangos definidos por sus hojas de datos. El sensor SHT31 presenta una precisión de ±0.3°C en temperatura y ±2% en humedad relativa(Sensirion, 2022); el BH1750 opera con resolución de 1 lux en el rango de 1 a 65 535 lux; y el VL53L0X presenta un error típico de ±3% en su rango de operación. Las lecturas fuera del rango físico esperado son descartadas por el firmware y registradas como eventos de error en el log del sistema.

IV.4.4 Desarrollo e implementación de la propuesta

El desarrollo del sistema de monitoreo inteligente para gabinetes en centros de datos se llevó a cabo mediante un enfoque incremental por etapas, en el que cada componente fue diseñado, implementado y verificado de forma independiente antes de su integración progresiva con el conjunto. Este modelo de desarrollo permite detectar y corregir errores en capas tempranas sin comprometer los componentes ya validados, reduciendo la complejidad de la integración final y facilitando la trazabilidad de cada decisión de diseño a lo largo del proceso.

La propuesta se estructuró en seis etapas de desarrollo, cada una asociada a uno de los objetivos específicos de la investigación y orientada a un componente funcional del sistema. La Figura 18 presenta una vista integral del sistema desplegado, en la que las seis etapas convergen sobre el equipo físico en formato rack 1U que las contiene e integra: la Etapa 1 corresponde al diseño del hardware de la Unidad de Adquisición y Control basada en ESP32; la Etapa 2 al firmware de comunicación de la UAC; la Etapa

3 al modelo de detección de anomalías mediante Isolation Forest; la Etapa 4 al Gateway IoT implementado sobre Raspberry Pi; la Etapa 5 a la plataforma de software para monitoreo y control; y la Etapa 6 a la infraestructura híbrida que extiende el alcance del sistema hacia la nube. Esta representación evidencia que, aun cuando cada etapa fue desarrollada y validada de forma independiente, todas confluyen en una única solución integrada físicamente en el gabinete del centro de datos.

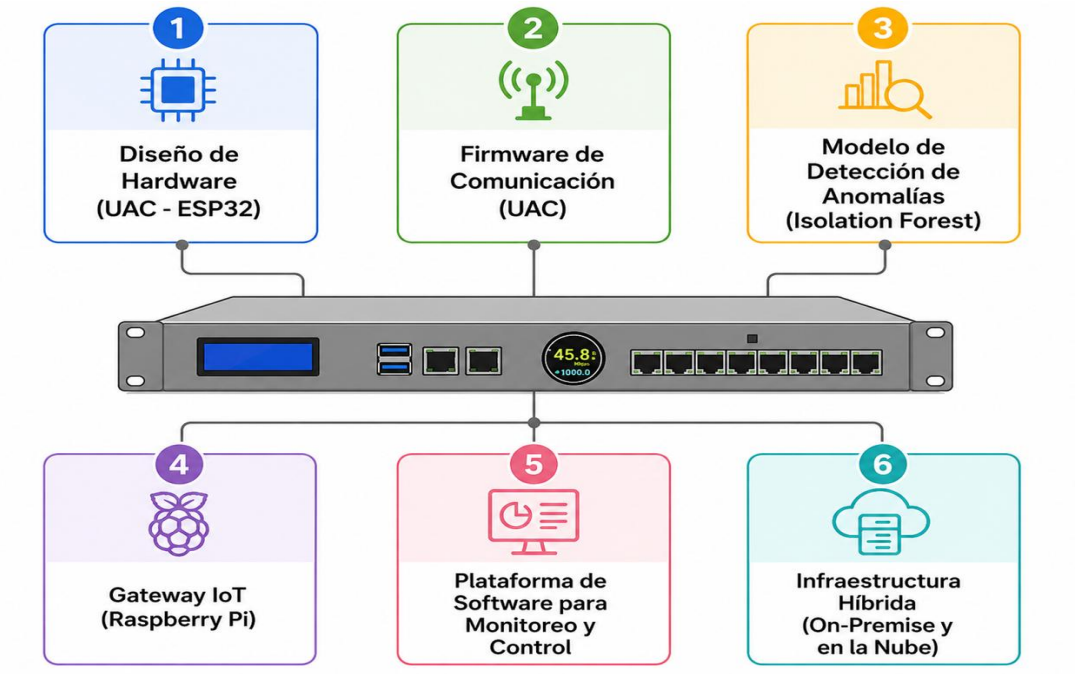


Figura 18 **Vista integral del sistema: las seis etapas de desarrollo convergentes en el equipo físico**. Fuente. Elaboración propia.

Desde una perspectiva funcional, estas seis etapas se organizan a su vez en cinco capas arquitectónicas, cada una con un rol técnico delimitado dentro del flujo de datos del sistema: desde la adquisición física de variables ambientales en la capa de sensado, hasta la publicación remota del sistema en la capa de nube, pasando por las capas intermedias de comunicación, procesamiento y aplicación. Esta organización por capas permite asociar cada etapa de desarrollo con su rol dentro de la arquitectura general y delimitar la frontera tecnológica entre los componentes que la integran. La Tabla 6 sintetiza esta correspondencia, indicando para cada capa su denominación, los componentes que la integran y la etapa o etapas de desarrollo en las que fue construida.

Tabla 6 Arquitectura por capas del sistema de monitoreo

Capa	Denominación	Componentes del sistema	Etapa de desarrollo
1	Capa de sensado	Hardware de la UAC basado en ESP32, sensores SHT31, VL53L0X, MPU6050, interruptor magnético (*reed switch*), actuadores LED RGB.	Etapa 1
2	Capa de comunicación	Firmware MQTT en ESP32, *broker MQTT*, red WiFi *softAP*, tópicos de sensores y actuadores	Etapas 2 y 3
3	Capa de procesamiento	Gateway IoT Raspberry Pi, servicio SNMP, API intermediaria, modelo *Isolation Forest*	Etapas 3 y 4
4	Capa de aplicación	*Backend Flask*, base de datos MariaDB, API REST, *frontend* ReactJS, motor de alertas SMTP	Etapa 5
5	Capa de nube	Azure Static Web Apps, Cloudflare Tunnel, dominio tesis-monitoring.xyz, certificados SSL	Etapa 6

Fuente. Elaboración propia

A continuación, se presentan las etapas de desarrollo correspondiente a lo presentado anteriormente:

a) Etapa 1: Hardware de la Unidad de Adquisición y Control (UAC) basado en ESP32

La primera etapa comprende el diseño e implementación del módulo electrónico encargado de la adquisición de variables ambientales, la supervisión estructural y el control del gabinete. Este módulo constituye la capa de sensado del sistema y la base física sobre la cual operan todas las etapas superiores. La Figura 19 muestra la arquitectura general del sistema en un formato de 1RU, separando los

planos de alimentación, adquisición y control, sensado, actuadores, visualización y Gateway IoT. El sistema parte de una fuente principal de 12 V / 10 A, desde la cual se generan las líneas de 5 V y 3.3 V mediante convertidores DC-DC. La adquisición se concentra en el ESP32-WROOM-32 y el multiplexor I²C, conectados a sensores ambientales y estructurales, mientras que el control de salida incluye el actuador LED RGB. Además, se integra una Raspberry Pi como Gateway IoT y pantallas locales de estado, representadas por un display LCD y una pantalla circular. La leyenda resume los tipos de conexión eléctrica y lógica empleados en el diseño.

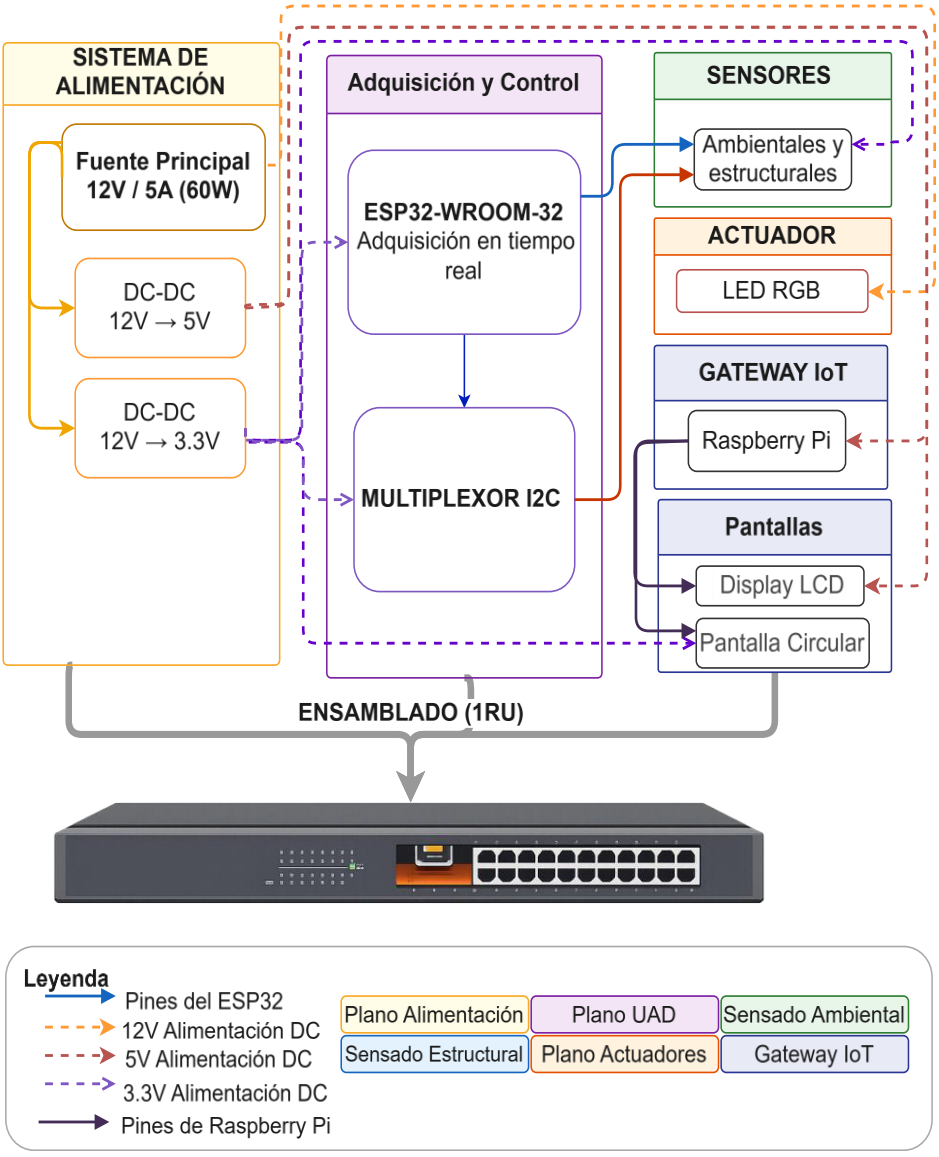


Figura 19 **Diagrama arquitectónico del sistema de la UAC basado en ESP32.** Fuente. Elaboración propia.

a.1) Requerimientos del hardware de la Unidad de Adquisición y Control (UAC) basado en ESP32

Se identificaron requerimientos funcionales: medición ambiental en zonas del gabinete, detección de acceso, transmisión MQTT y control de actuadores. La Tabla 7 presenta la clasificación completa.

Tabla 7 **Requisitos del hardware de la UAC basado en ESP32**

ID	Tipo	Descripción
RF-01	Funcional	Medir temperatura y humedad en tres zonas verticales del gabinete. Alertar si temperatura > 27 °C o humedad > 60 %.
RF-02	Funcional	Detectar apertura/cierre de puerta y signos de manipulación no autorizada (vibración, apertura de puerta, cambio de iluminación).
RF-03	Funcional	Procesar datos localmente y transmitirlos al sistema central mediante protocolo MQTT vía WiFi.
RF-04	Funcional	Controlar actuadores de señalización (LED RGB)
RNF-02	No funcional	Disponibilidad mínima del 99,9 %.
RNF-03	No funcional	Diseño modular que permita agregar y retirar sensores sin rediseñar la plataforma base.
RNF-04	No funcional	El hardware debe adaptarse al formato físico rack 19" (44,45 mm de altura).
RNF-05	No funcional	Los niveles de tensión del sistema de alimentación (3.3V, 5V y 12V) deben mantenerse dentro del margen de ±4% sobre el valor nominal.
R-01	Restricción	Integración en chasis metálico rack 1U, compatible con el estándar EIA-310 de 19".
R-02	Restricción	Sensores con interfaz digital estándar (I^2C, GPIO); sin acondicionadores analógicos adicionales.

Fuente. Elaboración propia

a.2) Selección de tecnologías

Los componentes fueron seleccionados con base en compatibilidad con interfaces I²C y GPIO, bajo consumo energético y disponibilidad comercial. El microcontrolador ESP32-WROOM-32 actúa como unidad central de procesamiento y comunicación; el multiplexor TCA9548A gestiona el bus I²C compartido entre los sensores; y el conjunto de sensores cubre las variables ambientales y estructurales definidas en los requerimientos. Tabla 8 lista los componentes seleccionados.

Tabla 8 **Componentes seleccionados**

Componente	**Modelo**
Microcontrolador	ESP32-WROOM-32
Multiplexor I²C	TCA9548A
Sensor Temperatura/Humedad	SHT31 (×3)
Sensor iluminancia	BH1750
Sensor distancia	VL53L0X
IMU / Vibración	MPU6050
Reed switch	Genérico
LED RGB	SMD 5050

Fuente. Elaboración propia

a.3) Diseño electrónico del sistema

El hardware se organizó bajo una topología centralizada en el ESP32, con los sensores I²C conectados mediante el multiplexor TCA9548A para evitar conflictos de direccionamiento, y los sensores digitales conectados directamente a pines GPIO en el ESP32. La comunicación con el Gateway IoT se realiza de forma inalámbrica

mediante WiFi y el protocolo MQTT. Para el control de la tira LED RGB se implementó una etapa de potencia con MOSFET controlados por señales PWM desde el ESP32. El detalle del esquema de diseño electrónico completo se presenta en la Figura 20. El ESP32 se comunica con el Gateway IoT mediante WiFi/MQTT y concentra los sensores I²C a través del multiplexor TCA9548A, usando las líneas (SDA y SCL). A este bus se conectan sensores como SHT31 para temperatura y humedad, MPU6050 para vibración y BH1750 para luminosidad. Asimismo, se observan entradas GPIO para el reed switch de puerta y el MQ-2 de humo, además de salidas PWM GPIO de 3.3 V hacia la etapa MOSFET IRLZ44N, encargada de controlar los canales R, G y B de la tira LED RGB. La figura también muestra las pantallas locales asociadas al Gateway, como la pantalla circular y el display LCD.

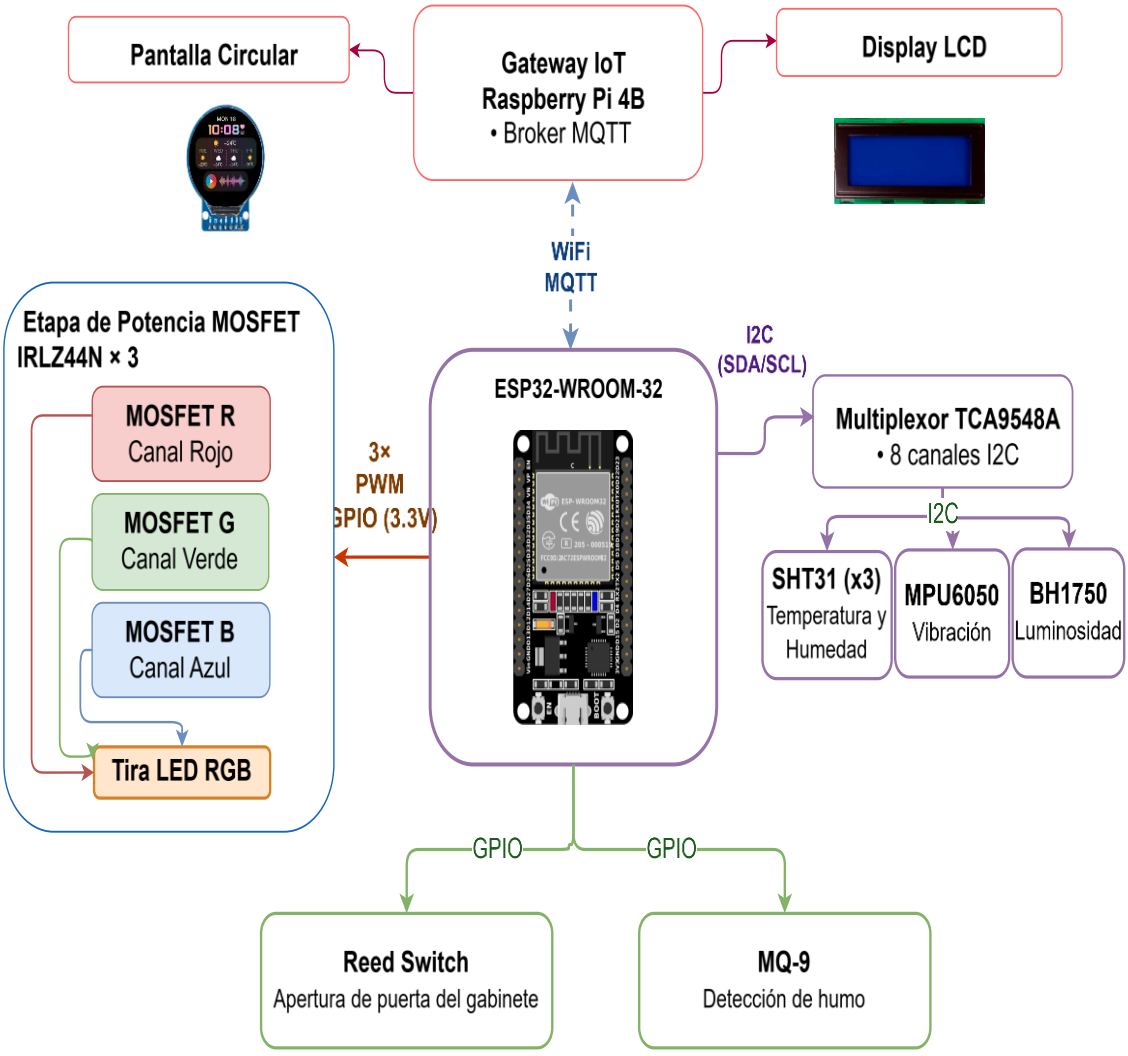


Figura 20 **Diagrama del sistema electrónico de la UAC.** Fuente. Elaboración propia.

Para la conexión de los sensores I²C del gabinete hacia el multiplexor TCA9548A se empleó cables UTP CAT5e (~50 pF/m) con conectores RJ45, asignando cada señal del bus con su referencia GND como se muestra en la Figura 21 para reducir el ruido inducido por los equipos activos en el rack. Esta decisión permite extender el bus hasta la distancia máxima de 2 m por sensor manteniendo la capacitancia del bus alrededor de 100 pF por debajo del límite de 400 pF permitido para I²C estándar a 100 kHz, además de facilitar una conexión firme y el reemplazo físico de los sensores sin modificar el PCB principal.

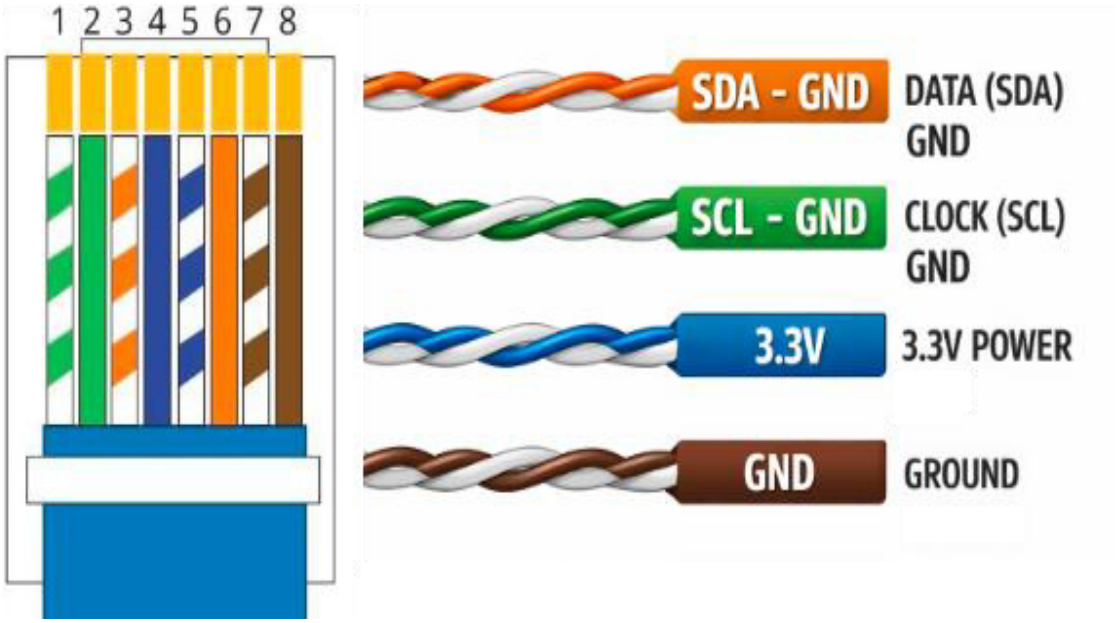


Figura 21 **Asignación de pines RJ45 para la conexión I²C mediante cable UTP CAT5e.** Fuente. Elaboración propia

La Figura 22 presenta el sistema de alimentación del hardware. La energía proveniente de la red AC de 220 VAC a 60 Hz ingresa a una fuente conmutada AC/DC, que entrega un bus principal de 12 VDC para la tira LED RGB SMD5050. A partir de este bus, dos convertidores reductores buck DC-DC generan los niveles regulados de 5 VDC y 3,3 VDC. El nivel de 5 VDC alimenta al Gateway IoT, al display LCD y al sensor MQ-2, mientras que el nivel de 3,3 VDC suministra energía al ESP32-WROOM-32, al multiplexor TCA9548A, a los sensores SHT31, MPU6050, BH1750 y VL53L0X, y a la pantalla circular. De esta forma, el diseño separa la alimentación

de actuadores, gateway, periféricos y sensores según sus requisitos eléctricos. La Tabla 9 resume las características de cada nivel bajo condiciones de operación nominal.

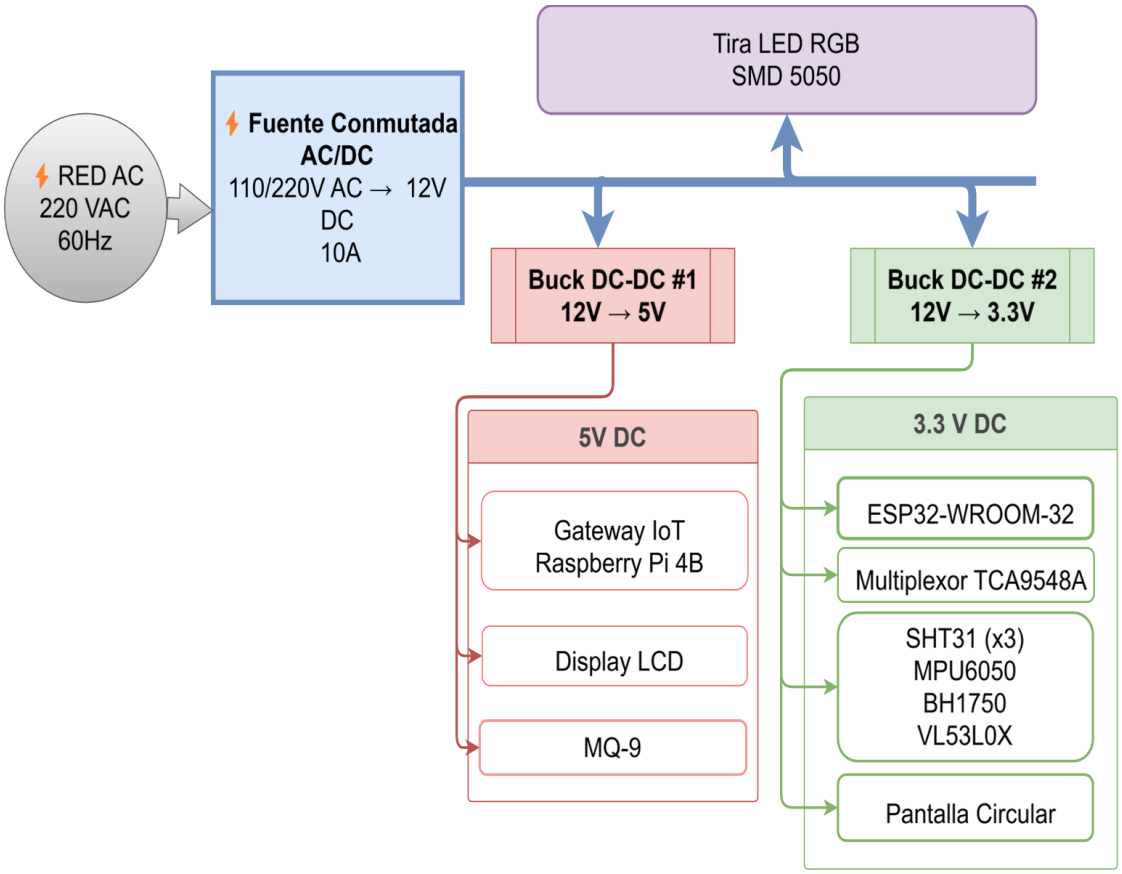


Figura 22 **Sistema de alimentación del hardware de la UAC.** Fuente. Elaboración propia.

Tabla 9 **Niveles de tensión del sistema de alimentación**

Nivel de tensión	Fuente	Cargas alimentadas	Corriente	Módulo
12 V DC	Fuente AC/DC	LED RGB SMD 5050, bus DC-DC	≤ 3,0 A	Fuente conmutada 12V, 10A
5 V DC	Buck DC-DC #1	Raspberry Pi 4B, Display LCD	≤ 3,0 A	LM2596 #1

3,3 V DC	Buck DC-DC #2	ESP32-WROOM-32, TCA9548A, SHT31 ×3, MPU6050, BH1750, Pantalla Circular LH128R-IG01	≤ 0,3 A	LM2596 #2

Fuente. Elaboración propia

a.4) Implementación del hardware del sistema

La primera iteración del PCB se realizó mediante grabado químico para validar conexiones y funcionamiento general. La Figura 23 presenta la distribución funcional de la primera iteración del PCB de la Unidad de Adquisición y Control. En la placa se identifican los bloques principales del diseño: el ESP32, el multiplexor I²C, el conversor DC-DC y la fila de conectores RJ45 hacia los sensores. Esta organización permitió ordenar físicamente los elementos de adquisición, alimentación e interconexión en una misma tarjeta, sirviendo como base para la validación inicial del diseño electrónico.

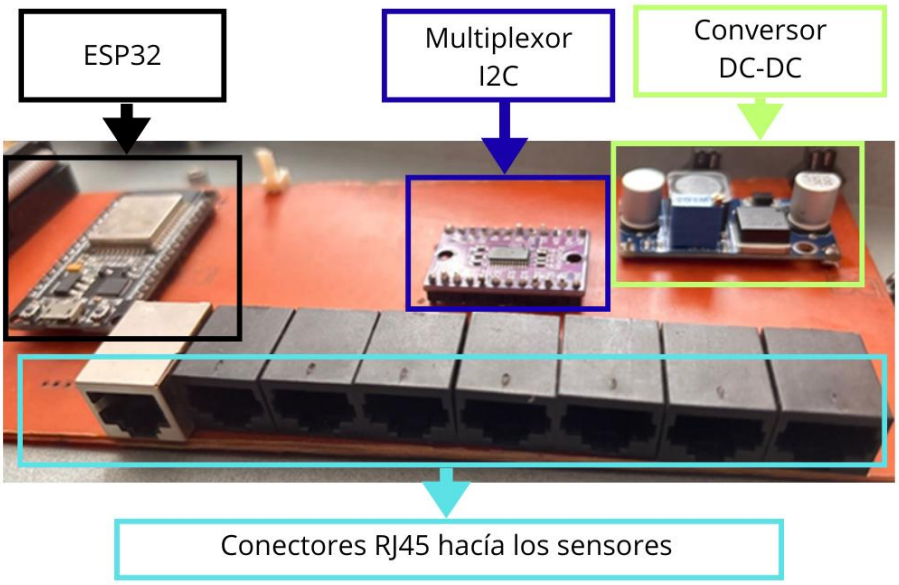


Figura 23 **Prototipo PCB: Primera iteración del hardware de la UAC (grabado químico).** Fuente. Elaboración propia.

A partir de los resultados obtenidos en la primera iteración, se desarrolló una versión optimizada del PCB con una distribución más ordenada de los bloques funcionales, mejor separación entre etapas de alimentación, adquisición y control, y una integración más limpia de los conectores destinados a sensores externos. Esta segunda versión permitió consolidar el diseño electrónico en una tarjeta más robusta y adecuada para su montaje dentro del gabinete 1U, manteniendo la compatibilidad con la arquitectura definida para la Unidad de Adquisición y Control, en la Figura 24 se presenta la distribución funcional del PCB de la Unidad de Adquisición y Control. En la tarjeta se identifican los bloques principales del diseño: la entrada de alimentación de 12 V, los conversores DC-DC de 5 V y 3,3 V, el módulo ESP32, el multiplexor I²C TCA9548A, la etapa de control PWM mediante MOSFETs, el conector hacia el Gateway IoT y los conectores RJ45 para sensores.

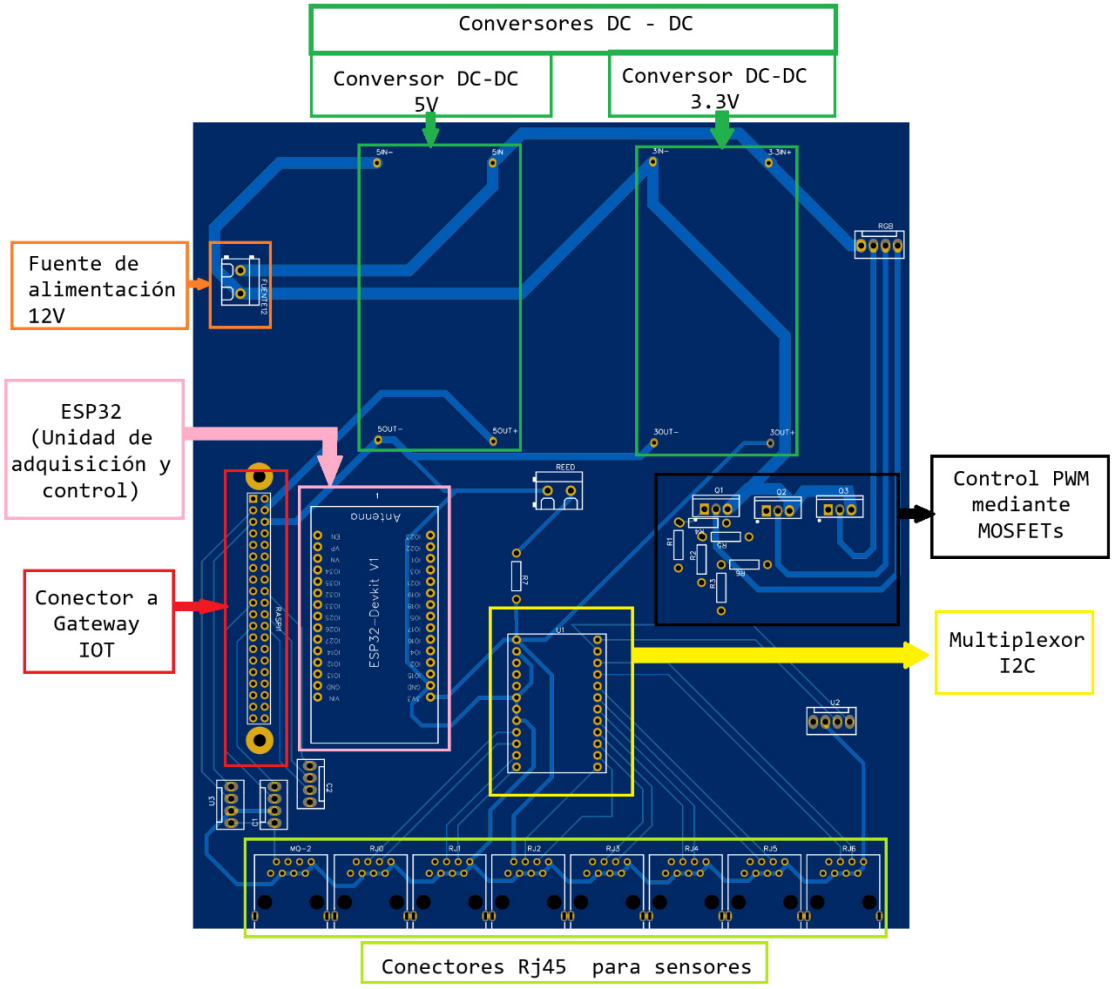


Figura 24 **Distribución funcional del PCB.** Fuente. Elaboración propia.

Una vez definida la distribución funcional del PCB, el siguiente paso consistió en integrar la tarjeta dentro de una estructura física que permitiera su instalación ordenada en el gabinete. Para ello, se consideró la disposición de la fuente de alimentación, el Gateway IoT y los puntos de conexión hacia sensores y actuadores, con el fin de mantener una arquitectura compacta, accesible y compatible con montaje en rack. La Figura 25 muestra la distribución final del sistema, incluyendo la tapa superior del chasis, la ubicación interna de los bloques, las pantallas LCD, las entradas RJ45 para sensores, la salida hacia la tira RGB y el reed switch asociado al monitoreo de apertura.

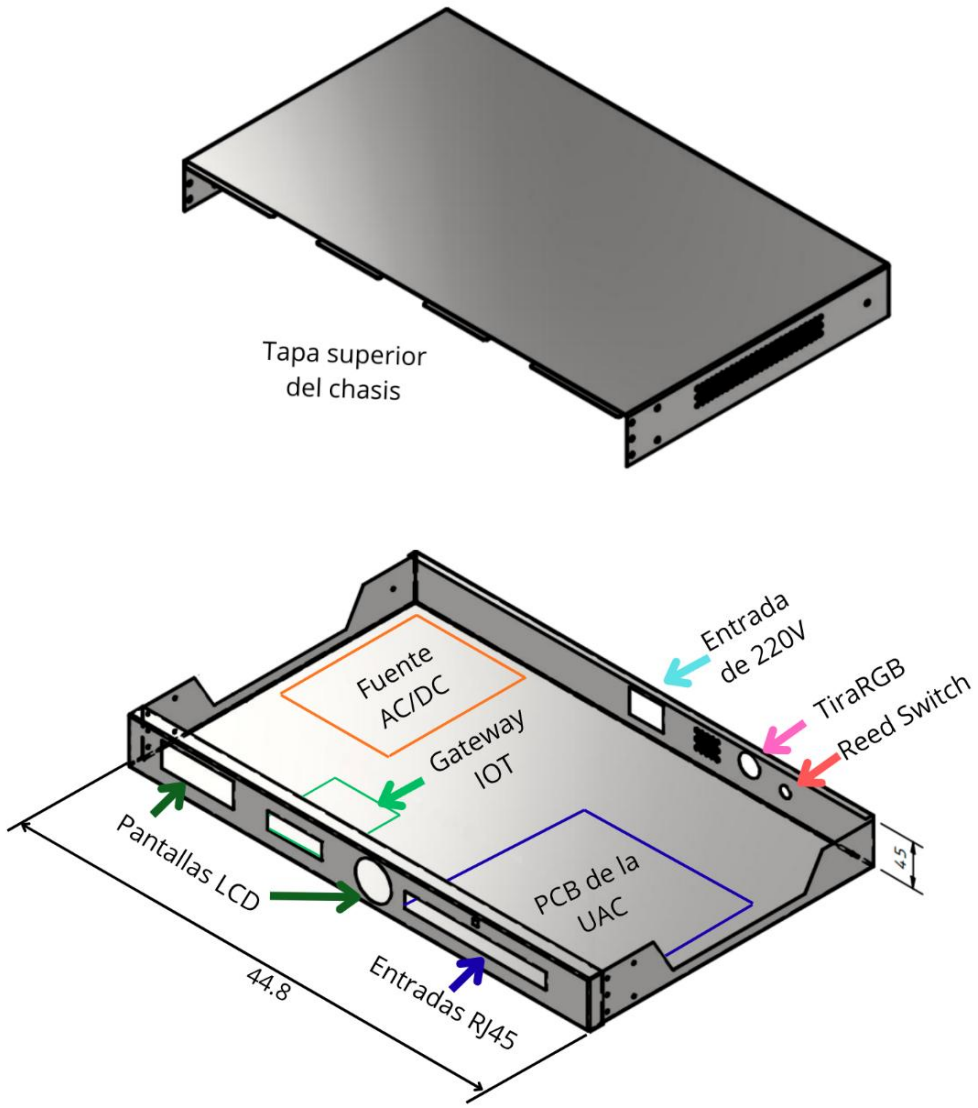


Figura 25 **Diagrama de distribución del chasis utilizado para el hardware.** Fuente. Elaboración propia.

b) Etapa 2: Firmware de la Unidad de Adquisición y Control

Esta etapa comprende el desarrollo del firmware responsable de adquirir las lecturas de los sensores de la UAC, gestionar el control de los actuadores y transmitir los datos al Gateway IoT mediante el protocolo MQTT, funcionando como el núcleo lógico que articula el comportamiento del nodo ESP32 dentro del sistema de monitoreo. Tal como se muestra en la Figura 26, el firmware adquiere las lecturas provenientes de los sensores, procesa y serializa los datos en un formato adecuado para su transmisión, y los envía al Gateway IoT mediante WiFi/MQTT para su publicación en el bróker MQTT, garantizando que la información llegue de manera estructurada y oportuna a las etapas posteriores del sistema. Asimismo, recibe comandos de control desde el Gateway IoT y ejecuta acciones sobre los actuadores, como la iluminación RGB, permitiendo que el nodo no solo cumpla una función de reporte, sino también de respuesta ante instrucciones externas. De esta manera, el alcance de esta etapa se centra exclusivamente en el comportamiento interno del nodo ESP32 y en su intercambio de datos y comandos con las demás etapas del sistema, sin abarcar el procesamiento, almacenamiento o interpretación de la información una vez que esta ha sido transmitida fuera del dispositivo.

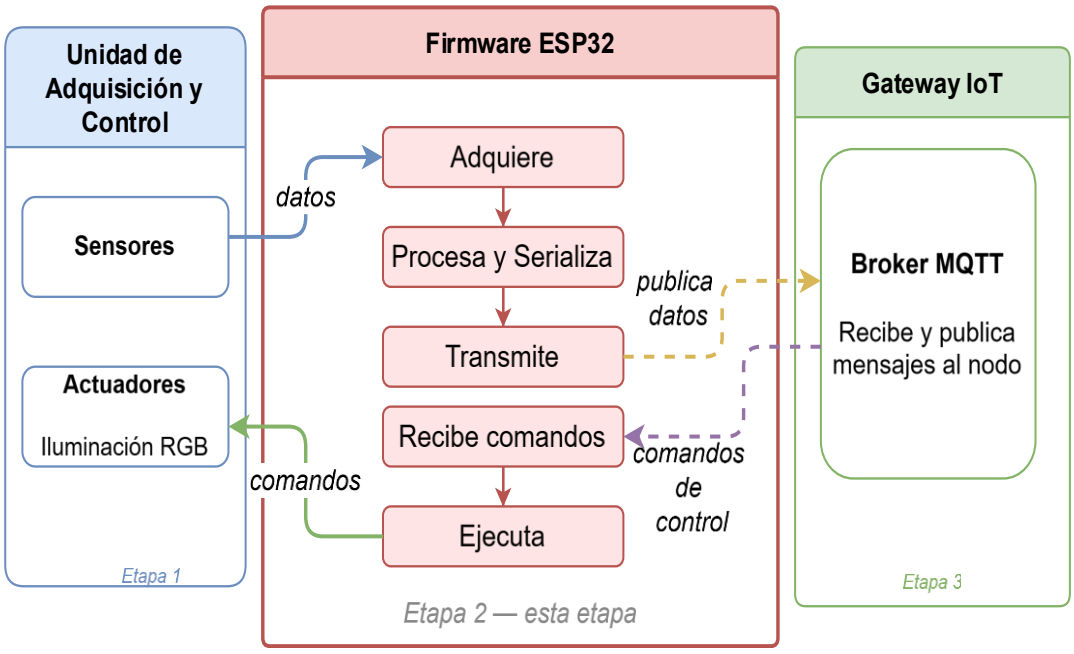


Figura 26 **Visión general del firmware ESP32 y su integración con el sistema.** Fuente. Elaboración propia.

b.1) Requerimientos del firmware

Se identificaron cinco requerimientos funcionales: conexión WiFi, publicación periódica de sensores, suscripción a tópicos de control, reconexión automática y publicación de estado, tres requerimientos no funcionales vinculados a latencia máxima de 500 ms, disponibilidad del 99,9 % y escalabilidad de tópicos, y tres restricciones que acotan el entorno tecnológico, la Tabla 10 presenta la clasificación completa.

Tabla 10 **Requisitos del firmware de la Unidad de Adquisición y Control (UAC)**

ID	Tipo	Descripción
RF-01	Funcional	Conectar el ESP32 a la red WiFi del Gateway IoT con SSID y credenciales fijas en firmware.
RF-02	Funcional	Publicar las lecturas de todos los sensores activos en un único tópico MQTT cada 3 segundos, en formato JSON plano, organizadas por canal del multiplexor TCA9548A.
RF-03	Funcional	Suscribirse a tópicos de control para ejecutar comandos sobre actuadores (LED RGB) de forma asíncrona mediante una función de respuesta (*callback)*.
RF-04	Funcional	Implementar reconexión automática ante pérdida de conexión WiFi o desconexión del *broker*, sin intervención manual.
RF-05	Funcional	Publicar el estado general del nodo (operativo/desconectando) en el tópico de estado correspondiente.
RNF-01	No funcional	Latencia máxima entre lectura de sensor y publicación MQTT inferior a 500 ms en condiciones normales de red.
RNF-02	No funcional	Disponibilidad continua del nodo ≥ 99,9 % en condiciones de red estable.

RNF-03	No funcional	La estructura de tópicos y el formato JSON deben ser escalables para integrar múltiples nodos sin modificar la lógica.
R-01	Restricción	Comunicación exclusiva vía MQTT sobre la red WiFi interna del Gateway.
R-02	Restricción	La marca de tiempo de cada mensaje es asignada por el Gateway al recibir el mensaje; el ESP32 no la genera.
R-03	Restricción	Las credenciales WiFi (SSID/clave) y MQTT (usuario/contraseña/IP del broker) se definen como constantes en el firmware.

Fuente. Elaboración propia

b.2) Selección de tecnologías

La selección se basó en compatibilidad directa con el hardware de la UAC en la Etapa 1, madurez de las bibliotecas disponibles para el entorno Arduino IDE y soporte nativo para el protocolo MQTT, la integración de PubSubClient sobre WiFi nativo del ESP32 con mensajes en JSON, permite una transmisión de datos con un bajo consumo de recursos, amplia disponibilidad de documentación y coherencia con arquitecturas IoT basadas en comunicación ligera. La Tabla 11 resume las decisiones tecnológicas adoptadas.

Tabla 11 **Selección de tecnologías**

Componente / Biblioteca	**Selección**
Biblioteca MQTT	PubSubClient
Protocolo de comunicación	MQTT
Formato de mensajes	JSON
Framework de desarrollo	Arduino IDE

Fuente. Elaboración propia

b.3) Diseño del firmware

El ESP32 opera como cliente MQTT bajo el modelo de publicación/suscripción sobre la red WiFi interna del Gateway IoT, sin exposición directa a redes externas. Como se muestra en la Figura 27, el firmware se organiza en cuatro capas funcionales: la capa de adquisición, encargada del escaneo del multiplexor TCA9548A y de la lectura cíclica de sensores como SHT31, BH1750, VL53L0X, MPU6050 y reed switch; la capa de serialización, donde las lecturas y estados se estructuran en formato JSON por canal; la capa de transporte, responsable de publicar la telemetría mediante MQTT con la librería PubSubClient; y la capa de control, que recibe comandos mediante un callback MQTT para accionar los LEDs RGB. La figura muestra los tópicos empleados para la publicación de datos del gabinete y la suscripción a comandos de actuadores, esta organización modular facilita el mantenimiento del firmware y permite incorporar nuevos sensores o funciones sin alterar de manera significativa el flujo principal del sistema.

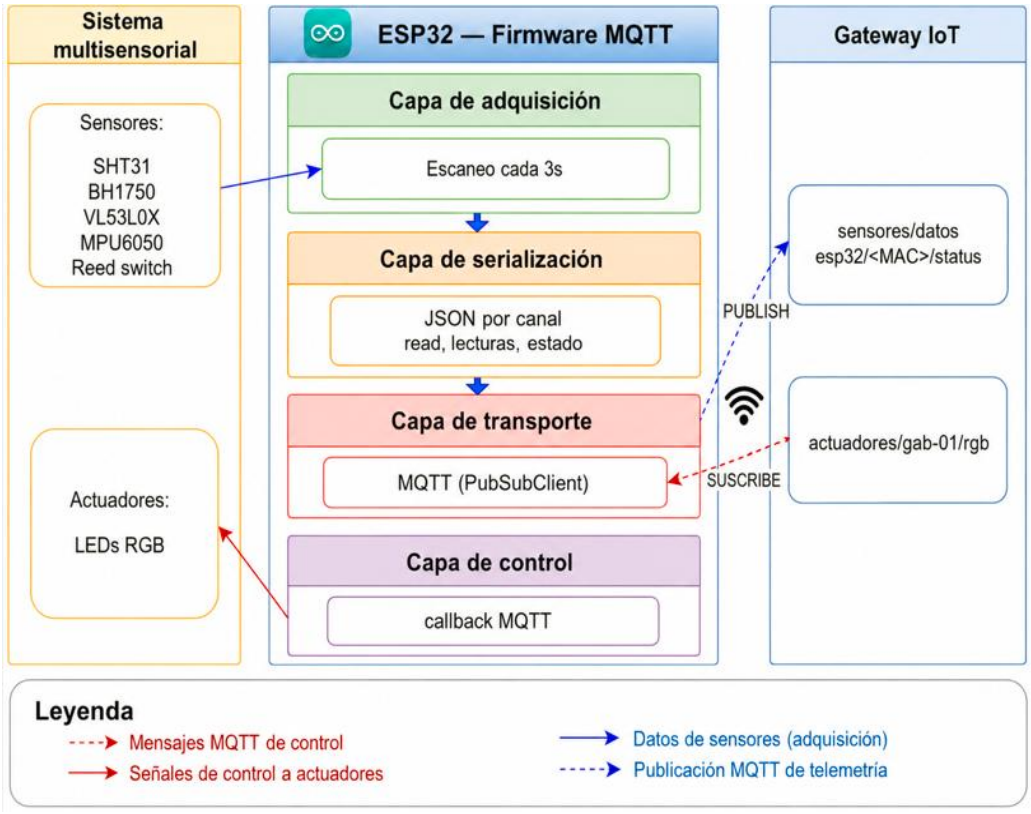


Figura 27 **Arquitectura del firmware ESP32.** Fuente. Elaboración propia.

El flujo de ejecución del firmware se organiza en dos fases principales, como se muestra en la Figura 28. En la fase de inicialización (setup), el ESP32 inicializa los pines GPIO, obtiene su dirección MAC e inicia la conexión WiFi. En el ciclo principal (loop), el firmware verifica el estado de la conexión WiFi y MQTT, ejecutando reconexión automática cada 1 y 3 segundos respectivamente en caso de interrupción. Una vez establecida la comunicación, procesa los mensajes entrantes y ejecuta tres tareas temporizadas: lectura y publicación de sensores cada 3 segundos, reescaneo del multiplexor cada 30 segundos, y publicación de estado de red y latencia cada 30 segundos. Además, la figura incluye el flujo del callback MQTT asíncrono, encargado de validar el tópico actuador/rgb e interpretar los datos en formato JSON para ejecutar el comando correspondiente sobre el LED RGB WS2812B.

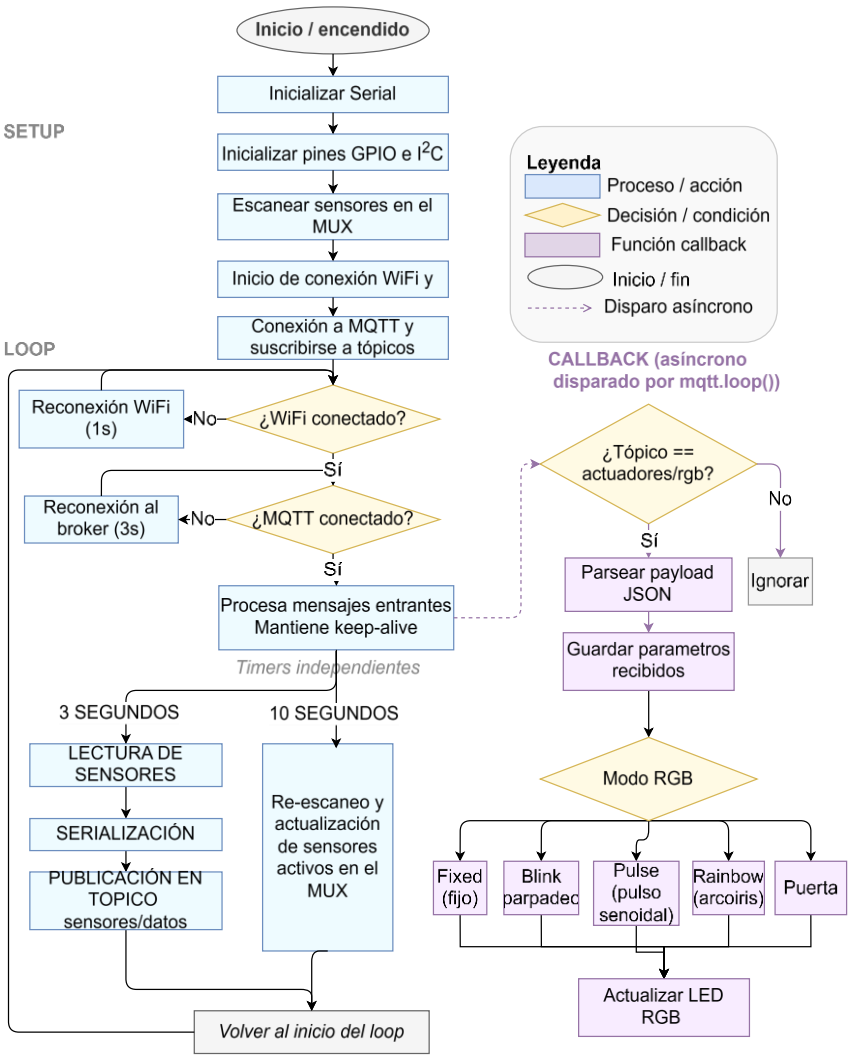


Figura 28 **Diagrama de flujo del firmware ESP32.** Fuente. Elaboración propia.

b.4) Implementación del firmware

El firmware publica un único mensaje JSON por ciclo con todos los sensores activos, organizados por canal del multiplexor y puerto utilizado, en lugar de un tópico separado por sensor. Esta decisión reduce el número de publicaciones y simplifica la correlación de datos en el backend (Etapa 5). La Tabla 12 describe los tres tópicos implementados con su dirección y contenido.

Tabla 12 **Tópicos MQTT implementados en el firmware**

Tópico	Dirección	Descripción
Sensores/datos	Publicación	Mensaje en formato JSON único con lecturas de todos los sensores activos organizadas por modelo del sensor. Publicado cada 10 s.
actuadores/gab-01/rgb	Suscripción	Comandos de control de iluminación LED RGB (SMD 5050): color, intensidad, modo.
esp32/<MAC>/status	Publicación	Estado general del nodo IoT identificado por MAC: operativo/desconectado.

Fuente. Elaboración propia

Además, los parámetros de conectividad tales como: credenciales WiFi, dirección del broker y constantes de configuración, se definen como constantes en el firmware, garantizando operación autónoma desde el encendido; las constantes se presentan en la Tabla 13.

Tabla 13 **Constantes de red y MQTT definidas en el firmware**

Constante	Valor (ejemplo)	Descripción
WIFI_SSID	"GabiNet-IoT"	SSID de la red WiFi del Gateway definido en firmware. El nodo se conecta

		automáticamente al encendido.
WIFI_PASS	" ClaveSegura2024"	Contraseña WPA2 de la red del Gateway. Definida como constante.
MQTT_BROKER	"192.168.4.1"	IP del Gateway en la red IoT interna. El ESP32 siempre apunta a esta dirección fija.
MQTT_PORT	1883	Puerto estándar MQTT del *broker* en el Gateway.
MQTT_USER / MQTT_PASS	"esp32" / "****"	Credenciales de autenticación ante el *broker*. Deben coincidir con las configuradas en el Gateway (Etapa 3).
GABINETE_ID	"GAB-01"	Identificador del nodo; se incluye en cada mensaje JSON y en la ruta del tópico MQTT.
PERIOD_MS	10000ms	Intervalo de publicación en milisegundos (10 s). Equilibrio entre resolución temporal y consumo de ancho de banda.

Fuente. Elaboración propia

Previo a la integración con el Gateway IoT de la Etapa 4, se implementó el firmware de la UAC basado en ESP32. La configuración consideró tres sensores SHT31 conectados en canales independientes del multiplexor TCA9548A, permitiendo la lectura organizada de variables ambientales desde distintos puntos del gabinete. La Figura 29 muestra la salida del Monitor Serial, donde se observa la identificación del ESP32 mediante su dirección MAC, la conexión a la red WiFi, la asignación de IP, el enlace con el broker MQTT, el reconocimiento de los sensores conectados y la generación de datos en formato JSON. Esta implementación establece la base de adquisición, estructuración y comunicación de datos requerida para su posterior envío al Gateway IoT.

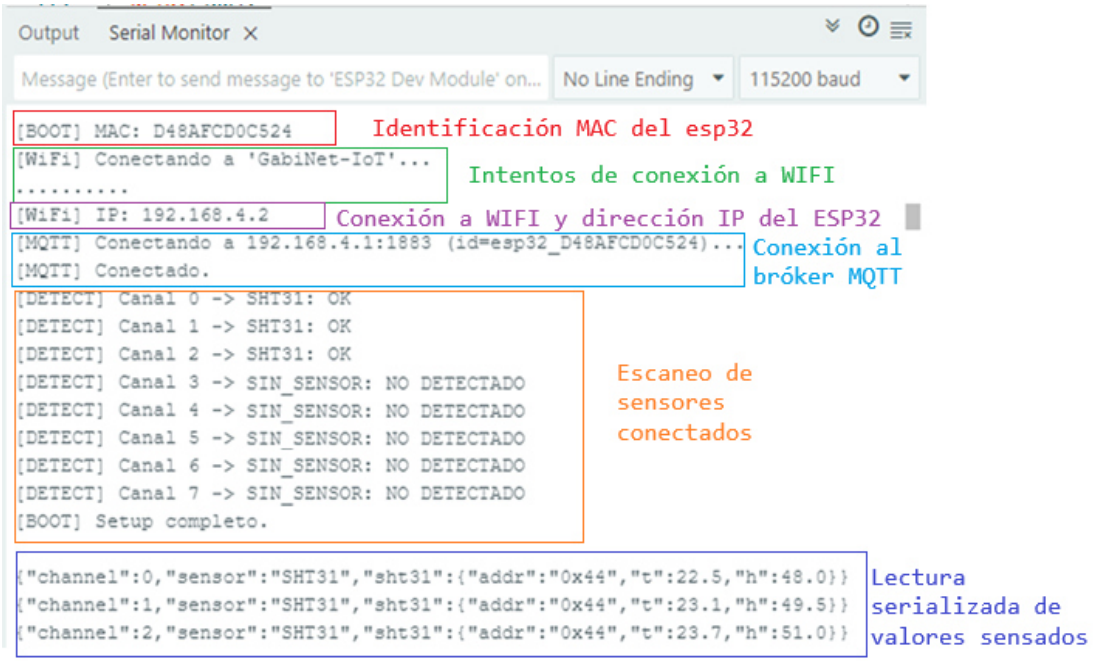


Figura 29 **Salida monitor serial firmware UAC basado en ESP32.** Fuente. Elaboración propia.

c) Etapa 3: Modelo de Detección de Anomalías mediante Isolation Forest

Esta etapa incorpora un modelo de aprendizaje automático no supervisado para la detección de anomalías en el entorno físico del gabinete, implementado como microservicio autónomo en el Gateway IoT, lo que permite que el análisis se ejecute de forma local sin depender de un procesamiento externo constante. El desarrollo siguió la metodología CRISP-DM, que estructuró el proceso en seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue, brindando un marco de trabajo ordenado que facilitó la trazabilidad de las decisiones tomadas en cada etapa del desarrollo. El algoritmo seleccionado fue Isolation Forest, que opera sin requerir etiquetado previo de datos, propiedad característica en el dominio de monitoreo de centros de datos donde los eventos anómalos son infrecuentes y raramente están catalogados, lo que dificulta la aplicación de enfoques supervisados tradicionales. Esta característica resulta especialmente ventajosa en el contexto de la tesis, ya que permite identificar comportamientos atípicos a partir del comportamiento normal observado, sin necesidad de contar con ejemplos previos de fallas o condiciones anómalas. En la Figura 30 se presenta la arquitectura de la etapa, donde se evidencia la integración del

modelo dentro del flujo de datos del Gateway IoT y su relación con las demás etapas del sistema.

Cabe precisar que el modelo implementado realiza detección y no predicción de fallas, en el sentido descrito en la sección de Bases teóricas: clasifica el estado actual del gabinete a partir de las lecturas más recientes, sin proyectar la ocurrencia de fallas futuras.

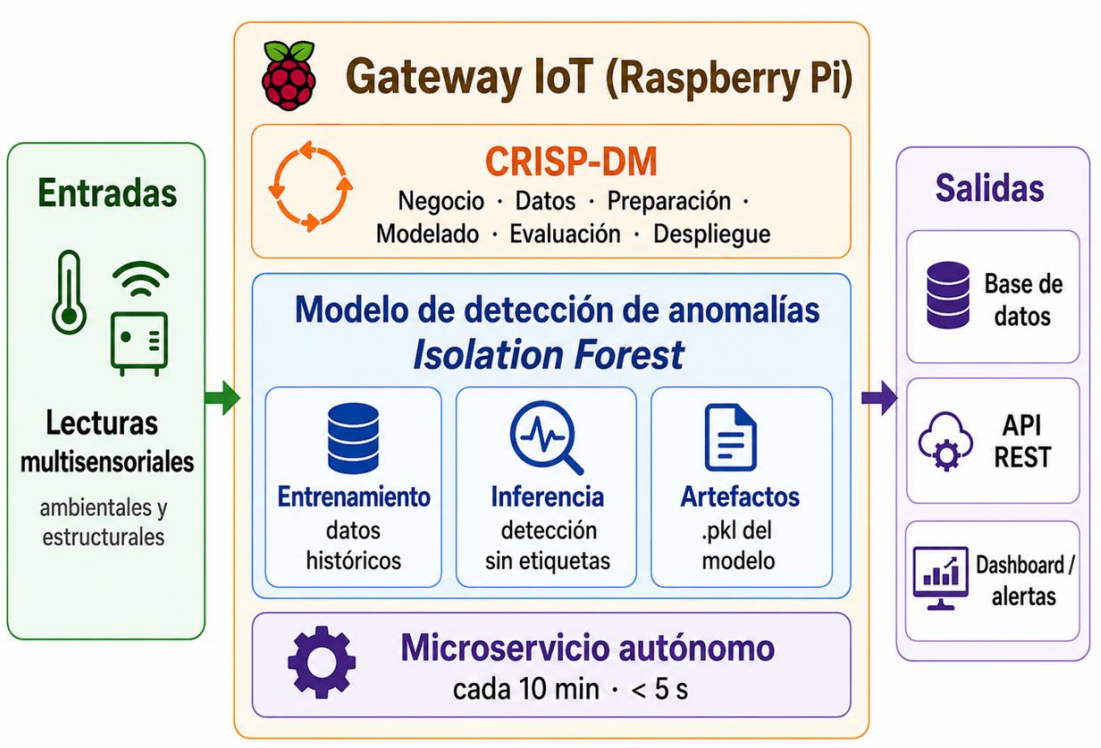


Figura 30 **Arquitectura del módulo de detección de anomalías basado en Isolation Forest.** Fuente. Elaboración propia.

c.1) Requerimientos del modelo de detección de anomalías

Los requerimientos de esta etapa se derivan de la necesidad de detectar condiciones inusuales en el gabinete de forma autónoma, sin intervención manual y sin etiquetado previo de datos. La Tabla 14 consolida los requisitos funcionales y no funcionales identificados para el modelo de detección de anomalías, que abarcan desde la obligación de analizar las lecturas de todos los sensores activos, hasta la restricción

de completar cada ciclo de inferencia en menos de cinco segundos, lo que garantiza que el microservicio no compita con otros procesos críticos del Gateway IoT.

Tabla 14 **Requisitos del módulo de detección de anomalías**

ID	Tipo	Descripción
RF-01	Funcional	El modelo debe analizar las últimas lecturas de todos los sensores activos del gabinete.
RF-02	Funcional	Cada resultado de inferencia debe persistirse con score, clasificación y vector de características.
RF-03	Funcional	Los resultados deben estar disponibles para el frontend a través de la API REST del backend.
RF-04	Funcional	El operador debe poder marcar una anomalía como revisada desde la interfaz web.
RNF-01	No funcional	El microservicio debe ejecutar inferencia de forma autónoma cada 10 minutos.
RNF-02	No funcional	El módulo debe operar exclusivamente con datos y artefactos locales, sin dependencia de red externa.
RNF-03	No funcional	La inferencia de un ciclo debe completarse en menos de 5 segundos.

Fuente. Elaboración propia

c.2) Selección de tecnologías

La Tabla 15 detalla las tecnologías y bibliotecas seleccionadas para el módulo de aprendizaje automático. Destaca la elección de scikit-learn como núcleo del módulo por su implementación optimizada de Isolation Forest con paralelismo nativo, y de

joblib como mecanismo de serialización de artefactos, que es el formato nativo de scikit-learn y resulta eficiente para ensembles de árboles del tamaño utilizado en este sistema. La gestión mediante systemd, al igual que en los demás servicios del Gateway, garantiza arranque automático y trazabilidad de logs.

Tabla 15 **Tecnologías seleccionadas para el módulo de detección de anomalías**

Tecnología	Versión	Rol	Justificación
Python	3.9.2	Lenguaje principal del módulo.	Compatibilidad directa con scikit-learn; misma versión que el backend Flask, eliminando conflictos de entorno.
scikit-learn	1.3.x	Provee Isolation Forest y Standard Scaler.	Implementación optimizada con paralelismo nativo (n_jobs=−1); API estable y ampliamente documentada.
pandas	2.x	Manipulación y pivotado del conjunto de datos.	Facilita la transformación de series temporales multisensor en formato ancho, con manejo nativo de valores faltantes.
NumPy	1.x	Cómputo vectorial en inferencia.	Bajo sobrecarga para operaciones matriciales en la Raspberry Pi; dependencia directa de scikit-learn.
joblib	1.x	Serialización de artefactos (.pkl).	Formato nativo de scikit-learn; eficiente para objetos de gran tamaño como ensembles de árboles.
pymysql	1.x	Conexión a MariaDB desde Python.	Driver puro Python sin dependencias de sistema; compatible con Raspberry Pi OS 64-bit sin instalación adicional.
matplotlib / seaborn	3.x	Gráficas de análisis exploratorio.	Herramientas estándar para inspección visual del dataset durante la fase de comprensión de datos.

systemd	—	Gestión del microservicio.	Garantiza arranque automático, reinicio ante fallo y trazabilidad de registros, coherente con los demás servicios del Gateway.

Fuente. Elaboración propia

c.3) Definición de eventos anómalos para el modelo de detección de anomalías

En el dominio de monitoreo de gabinetes de centros de datos con características Tier 1, un evento anómalo representa una condición que puede comprometer el funcionamiento seguro y continuo de la infraestructura. Es importante precisar que el modelo propuesto, al ser un algoritmo no supervisado, aprende el perfil de operación normal del gabinete y señala como anómala cualquier observación que se desvíe significativamente de dicho perfil, sin eventos predefinidos en su configuración. En consecuencia, los eventos descritos a continuación constituyen un conjunto de escenarios de riesgo representativos que sirven como referencia para justificar el monitoreo y para construir el conjunto de validación del modelo.

Se definieron cinco escenarios de relevancia operacional, resumidos en la Tabla 16 que detalla cada evento, su descripción operacional y las variables físicas habitualmente involucradas; algunas anomalías se manifiestan mediante una sola variable, como la presencia de humo, mientras que otras emergen de la combinación de varias señales, como el estado de la puerta (reed), la iluminancia, la distancia, la vibración y la hora de ocurrencia.

Tabla 16 **Eventos habituales anómalos operacionales y variables asociadas para el modelo de detección**

#	Evento	Descripción operacional	Variables activadas
1	Sobrecalentamiento	Temperatura sostenida por encima de 27 °C (límite ASHRAE clase	temperatura, temp_rate

		A1) o tasa de incremento anormal	
2	Acceso fuera de horario	Apertura de puerta en horario no operativo o sin correlación con patrón histórico	reed, luz, distancia, hora_sin, hora_cos
3	Presencia de humo o gases	Concentración de gases combustibles superior al umbral de operación normal	humo
4	Acceso no autorizado	Combinación de puerta abierta + vibración + cambio brusco de distancia sin patrón registrado	reed, mpu6050, distancia
5	Vibración anómala	Movimiento del gabinete sin correlación con apertura de puerta	mpu6050, reed

Fuente. Elaboración propia

Estos cinco escenarios se definieron como escenarios de validación representativos de los riesgos operativos del gabinete porque representan los riesgos más frecuentes en entornos con características Tier 1 sin redundancia activa, donde la ausencia de mecanismos de respaldo automático incrementa la probabilidad de que estas condiciones deriven en afectaciones directas sobre la disponibilidad del servicio; son detectables mediante las variables físicas adquiridas por los sensores implementados, lo que garantiza que el modelo cuente con información suficiente para identificar cada escenario sin depender de fuentes de datos adicionales; y generan patrones de comportamiento multivariado que se desvían del perfil de operación normal, condición necesaria para su identificación mediante Isolation Forest, algoritmo cuya efectividad se sustenta precisamente en la capacidad de aislar observaciones cuyo comportamiento conjunto difiere del resto del conjunto de datos. Esta selección permitió, además, cubrir un espectro variado de condiciones anómalas, evitando que la validación del modelo se centrara únicamente en un tipo específico de falla y asegurando que el sistema fuera puesto a prueba frente a situaciones con

distintos orígenes y niveles de severidad. La Figura 31 muestra cómo los valores de las 10 características son evaluados conjuntamente en cada ciclo de inferencia una vez desplegado en el Gateway IoT, evidenciando la forma en que el modelo integra la información proveniente de múltiples variables para emitir una evaluación consolidada del estado del gabinete en cada iteración.

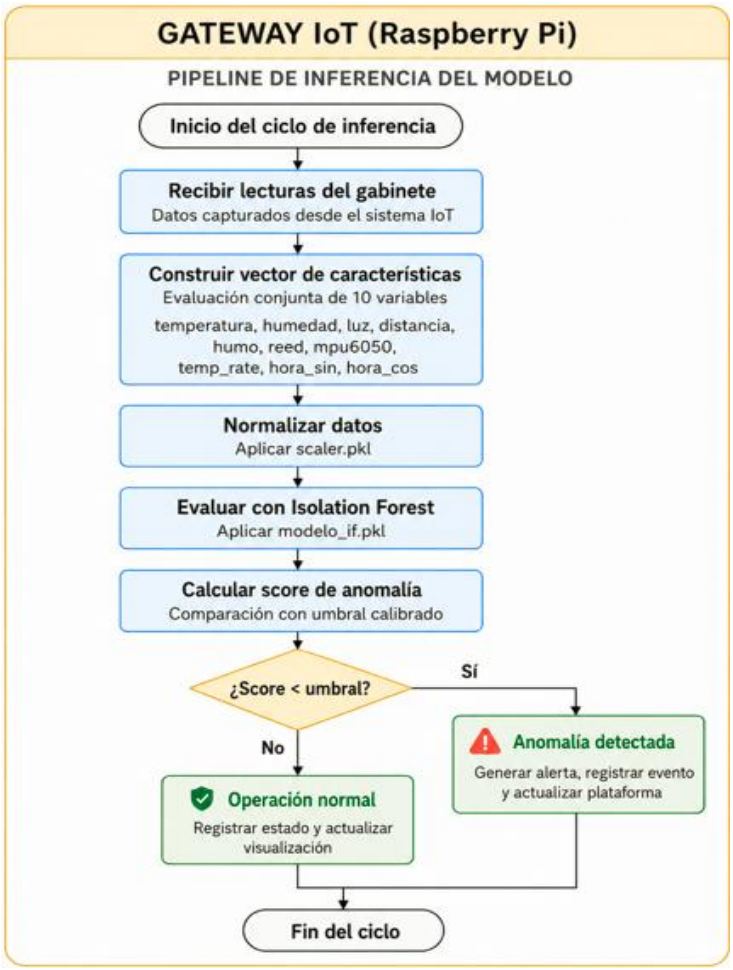


Figura 31 **Flujo de procesamiento (pipeline) de inferencia del modelo de detección de anomalías.** Fuente. Elaboración propia.

c.4) Diseño del módulo de detección de anomalías

El diseño aplicó las seis fases del proceso CRISP-DM, en la Tabla 17 se resume cada fase con su entrada, proceso y salida correspondiente. El ciclo inicia con

la comprensión del negocio, donde se definió el objetivo de detección sin umbral fijo a partir de los requisitos del sistema, y culmina con el despliegue del microservicio ml_monitoreo en el Gateway IoT, todo el detalle de código completo del proceso de extracción, transformación y entrenamiento se presenta en el

Anexo D.

Tabla 17 **Fases CRISP-DM aplicadas al módulo de detección de anomalías**

Fase	**Entrada**	**Proceso**	**Salida**
1. Comprensión del negocio	Requisitos del sistema	Identificación del objetivo de detección sin umbral fijo	Objetivo de minería de datos definido
2. Comprensión de los datos	Tabla lecturas en MariaDB	Extracción de 8 combinaciones sensor-medida; análisis exploratorio	Conjunto de datos crudo
3. Preparación de los datos	Conjunto de datos crudo	Pivotado, limpieza, imputación por mediana; derivación de 3 características (features)	Vector de 11 características
4. Modelado	Conjunto de datos preparado (n = 1 202)	Normalización Standard Scaler; entrenamiento Isolation Forest	modelo_if.pkl, scaler.pkl, features.pkl
5. Evaluación	Modelo entrenado + 1 200 muestras de evaluación	Evaluación frente a conjunto de prueba etiquetado mediante	Recall 99.33 %, F1 97.47 % (Cap. V)

		inyección de 5 escenarios	
6. Despliegue	Artefactos .pkl	Transferencia SCP a Raspberry Pi; registro como servicio systemd	Microservicio ml_monitoreo activo

Fuente. Elaboración propia

Presentadas las seis fases de forma resumida, cabe señalar que la ejecución práctica del ciclo no comenzó con la fase 1, sino con la fase 2 (Comprensión de los datos). Esto se debe a que el objetivo de negocio ya estaba determinado por los requisitos del sistema definidos en la sección IV.4.4.C.c.1: detectar condiciones anómalas en el gabinete de forma autónoma y sin etiquetado previo, por lo mencionado, la fase 1 quedó resuelta en la etapa de requerimientos.

i. Comprensión de los datos

Los datos de entrenamiento inicial se extrajeron de la tabla lecturas de la base de datos monitoring_iot, entre el 17 y el 29 de marzo de 2026, filtrando siete combinaciones sensor-medida: temperatura y humedad relativa (SHT31), iluminancia (BH1750), distancia (VL53L0X), estado del acelerómetro-giroscopio (MPU6050), concentración de gases (MQ-2) y estado de puerta (reed). Tras el pivotado a formato ancho, se aplicó una limpieza en entrenar_modelo.py: eliminación de filas con menos de seis de los ocho canales presentes (dropna, thresh=6) e imputación de valores faltantes con la mediana de cada columna. La Figura 32 presenta cuatro subgráficas del análisis exploratorio. En las distribuciones: la temperatura se concentra entre 23–26 °C con cola derecha leve; la humedad presenta sesgo positivo entre 70–75 %; la iluminancia se concentra cerca de cero lux, con picos por aperturas de puerta; y la distancia refleja la posición estable del objeto más cercano al sensor. La matriz de correlación muestra valores de Pearson generalmente bajos ($|r| < 0{,}30$). La serie temporal de temperatura muestra un comportamiento cíclico con variaciones diurnas de ≈1–2 °C y oscilaciones ligadas al ciclo de ventilación del gabinete.

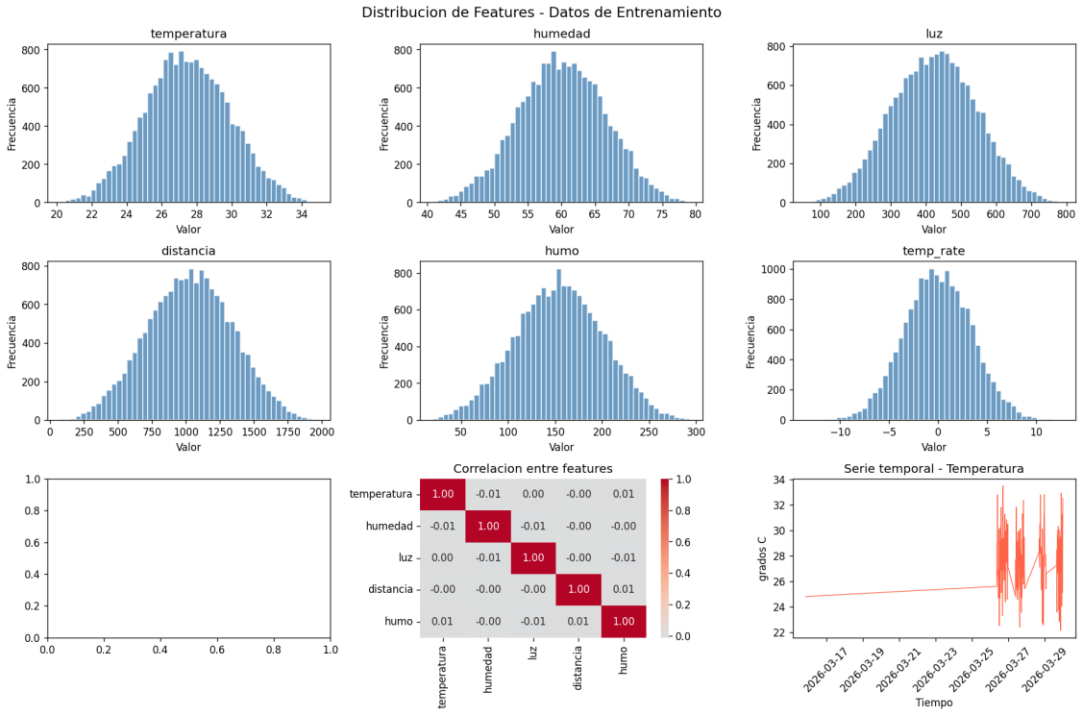


Figura 32 **Análisis exploratorio de los datos de entrenamiento: distribución de variables continuas y matriz de correlaciones entre características.** Fuente. Elaboración propia.

ii. Preparación de los datos

El flujo de procesamiento (pipeline) de preparación, implementado en entrenar_modelo y esquematizado en las figuras siguientes, comprende cuatro etapas secuenciales. Primero, las lecturas en formato largo fueron pivotadas a formato ancho agrupando por intervalos de 10 segundos mediante pivot_table con aggfunc='mean', generando una columna por combinación sensor-medida, de modo que cada fila representa el estado simultáneo de todos los sensores en un instante dado. Segundo, se aplicó la limpieza descrita en la fase anterior: eliminación con dropna(thresh=6) e imputación de faltantes con fillna (mediana). Tercero, se derivaron tres características adicionales: temp_rate, tasa de cambio de temperatura entre lecturas consecutivas; hora_sin y hora_cos, codificación cíclica de la hora del día. El vector final de 11 características se detalla en la Tabla 18 e incluye ocho variables directamente medidas por los sensores y tres características derivadas.

Tabla 18 **Características del vector de entrada al modelo (FEATURES)**

N°	Característica	Sensor	Descripción
1	temperatura	SHT31 (0x44)	Temperatura del gabinete en °C.
2	humedad	SHT31 (0x44)	Humedad relativa en %.
3	luz	BH1750 (0x23)	Iluminancia en lux.
4	distancia	VL53L0X (0x29)	Distancia al objeto más cercano en milímetros.
5	humo	MQ-2	Concentración de gases combustibles (valor analógico escalado).
6	reed	Reed switch	Estado de la puerta (binario).
7	mpu6050	MPU6050 (0x68)	Estado del acelerómetro/giroscopio (binario en fase actual).
8	temp_rate	Derivada	Diferencia de temperatura entre lecturas consecutivas (°C/ciclo).
9	hora_sin	Derivada	Componente seno de la hora del día - codificación cíclica.
10	hora_cos	Derivada	Componente coseno de la hora del día - codificación cíclica.

Fuente. Elaboración propia

Para ilustrar la composición práctica del vector de características, la Figura 33 presenta cinco observaciones representativas extraídas del conjunto de datos de reentrenamiento (30 de abril - 8 de mayo de 2026), valores de temp_rate, hora_sin y hora_cos fueron calculados mediante las transformaciones descritas en la Preparación de los datos.

Intervalos de 10 s con todos los sensores presentes: 71,254

```
============================================================
  71,254 observaciones  |  2026-04-30 – 2026-05-08
============================================================
```

MUESTRA DE 5 OBSERVACIONES (vector de 11 caracteristicas):

timestamp	temperatura	humedad	luz	distancia	humo	movimiento	reed	mpu6050	temp_rate	hora_sin	hora_cos
2026-04-30 00:00:00	25.61	66.58	1.30	265.8	80.30	0	0	0	0.0000	0.0000	1.0000
2026-05-02 01:28:50	26.07	68.03	17.47	356.7	15.06	0	0	0	-0.0100	0.3780	0.9258
2026-05-04 02:57:40	25.84	64.96	0.97	451.0	8.50	0	0	0	-0.0300	0.6999	0.7143
2026-05-06 04:26:30	25.90	66.37	7.91	219.2	20.37	0	0	0	0.0200	0.9179	0.3967
2026-05-08 05:56:00	25.50	67.74	0.00	205.9	77.33	0	1	0	-0.1300	0.9998	0.0175

Figura 33 **Muestra de cinco observaciones del conjunto de datos de reentrenamiento con el vector completo de 11 características**. Fuente. Elaboración propia.

iii. Modelado

Se empleó Isolation Forest, que detecta anomalías midiendo cuántas particiones aleatorias son necesarias para aislar cada observación: las anómalas se aíslan con pocas particiones, las normales con muchas. El fundamento matemático del algoritmo, incluyendo la definición del puntaje de anomalía y la función de decisión, se desarrolla en la sección II.1.5 del Marco Teórico. Previo al entrenamiento, cada característica j del vector de entrada fue normalizada mediante la transformación estándar definida en la Ecuación 3:

$$z_j = (x_j - \mu_{j)}/\sigma_{j)}$$

Donde z_j es el valor normalizado de la característica j, x_j es el valor original, $\mu_{j)}$ es la media de esa característica calculada sobre el conjunto de entrenamiento completo, y $\sigma_{j)}$ es su desviación estándar. Esta normalización garantiza que diferencias

de escala entre variables, por ejemplo, distancia en milímetros frente a variables binarias como el estado de la puerta, no introduzcan sesgos en el proceso de aislamiento del modelo. Los hiperparámetros fijados se presentan en la Tabla 19; su selección buscó equilibrar la estabilidad del puntaje con la baja frecuencia de anomalías esperada en operación normal de un gabinete de centro de datos.

Tabla 19 **Hiperparámetros del modelo Isolation Forest**

Hiperparámetro	**Valor**	**Justificación**
n_estimators	100	Suficiente para estabilizar el puntaje en el conjunto de datos de entrenamiento.
contamination	0.02	Refleja la baja frecuencia de eventos inusuales esperada en operación normal.
random_state	42	Garantiza reproducibilidad del entrenamiento.
n_jobs	−1	Paralelismo sobre todos los núcleos del procesador durante el entrenamiento.

Fuente. Elaboración propia

El algoritmo Isolation Forest fue implementado utilizando la clase IsolationForest de la librería scikit-learn. El entrenamiento se realizó mediante modelo.fit(X_train) sobre el vector de 11 características normalizado por StandardScaler; durante la inferencia en el microservicio, modelo.decision_function(X) retorna un puntaje continuo, donde los valores menores que cero se clasifican como anomalías, mientras que modelo.predict(X) devuelve la clasificación binaria (−1 = anomalía, +1 = normal). El umbral de decisión depende del

offset calculado a partir del hiperparámetro contamination, que fija la proporción esperada de anomalías en el conjunto de entrenamiento.(Liu et al., 2012)(Geron, 2022)

Isolation Forest no optimiza una función de pérdida de forma iterativa como los modelos basados en descenso de gradiente; cada árbol se construye de forma independiente. Su equivalente a una curva de entrenamiento es la estabilidad del score según el número de árboles del ensemble. Al evaluar modelo_if.pkl con subconjuntos crecientes de sus 200 árboles sobre el vector promedio de entrenamiento, el score se estabiliza a partir de aproximadamente 20 árboles, confirmando que n_estimators = 200 opera con margen sobre el punto de convergencia tal como se evidencia la convergencia grafica del ensemble en la Figura 34.

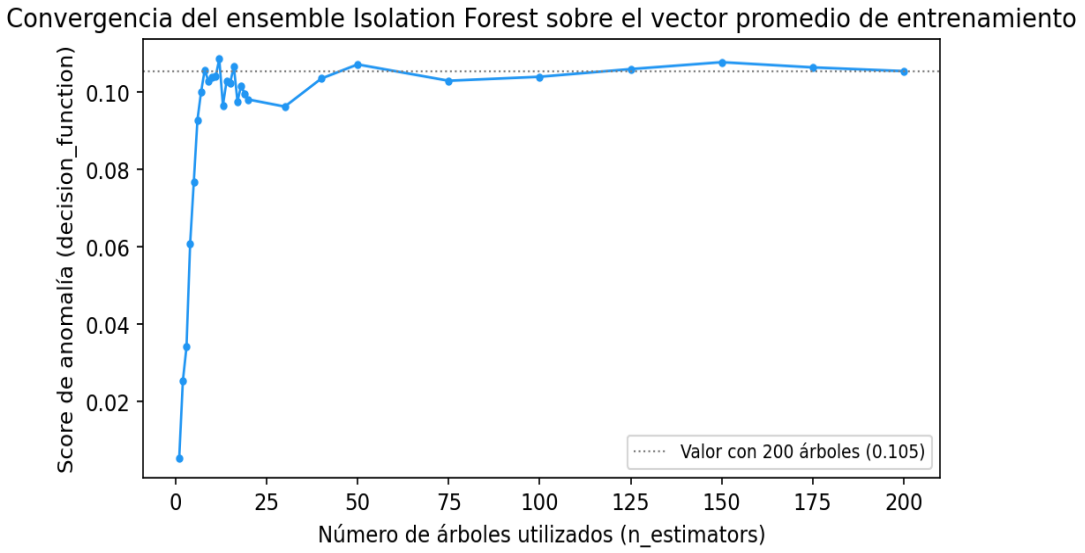


Figura 34 **Convergencia del ensemble Isolation Forest sobre el vector promedio de entrenamiento, en función del número de árboles utilizados.** Fuente. Elaboración propia, a partir de la evaluación del modelo modelo_if.pkl.

Como resultado del entrenamiento, el script entrenar_modelo.py genera automáticamente tres artefactos serializados mediante joblib: modelo_if.pkl, que contiene el modelo Isolation Forest entrenado; scaler.pkl, con los parámetros de normalización del StandardScaler; y features.pkl, con el orden de las 11 características

del vector de entrada. El detalle del código que ejecuta este proceso se presenta en el Anexo D.

iv. Evaluación

Dado que los datos de producción no disponen de etiquetas verificadas, la evaluación de desempeño se realizó mediante un conjunto de prueba construido ad hoc: 600 muestras normales reales del gabinete y 600 anomalías sintéticas que representan 5 escenarios operacionales de riesgo real. Esta estrategia permitió disponer de una referencia controlada para identificar correctamente verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos, sin modificar la naturaleza no supervisada del modelo basado en Isolation Forest. A partir de este conjunto de evaluación se calcularon métricas cuantitativas de desempeño, obteniéndose un recall de 99.33 %, un F1-score de 97.47 % y un tiempo de inferencia de 0.057 ms por muestra, evidenciando una elevada capacidad de detección y una respuesta computacional rápida. Los resultados detallados, incluyendo la matriz de confusión y el análisis desagregado por cada escenario de anomalía, se presentan en el Capítulo V, sección V.1.3.

v. Despliegue

Una vez entrenado y evaluado el modelo, los tres artefactos generados por entrenar_modelo.py (modelo_if.pkl, scaler.pkl y features.pkl) fueron transferidos desde el equipo de desarrollo hacia el Gateway IoT mediante SCP sobre SSH, garantizando una transferencia íntegra y segura.

En la Raspberry Pi, los artefactos se almacenaron en el directorio del servicio y el microservicio de inferencia se registró como unidad systemd, que es el gestor de servicios de Linux que administra procesos en segundo plano, por lo que el servicio queda bajo el nombre ml_monitoreo.service; su habilitación dejó el servicio con arranque automático al inicio del sistema y política de reinicio ante fallo (Restart=always), en coherencia con la gestión adoptada para los demás servicios del Gateway IoT. La Figura 35 esquematiza este proceso, desde la generación de los artefactos hasta la activación del servicio en la Raspberry Pi.

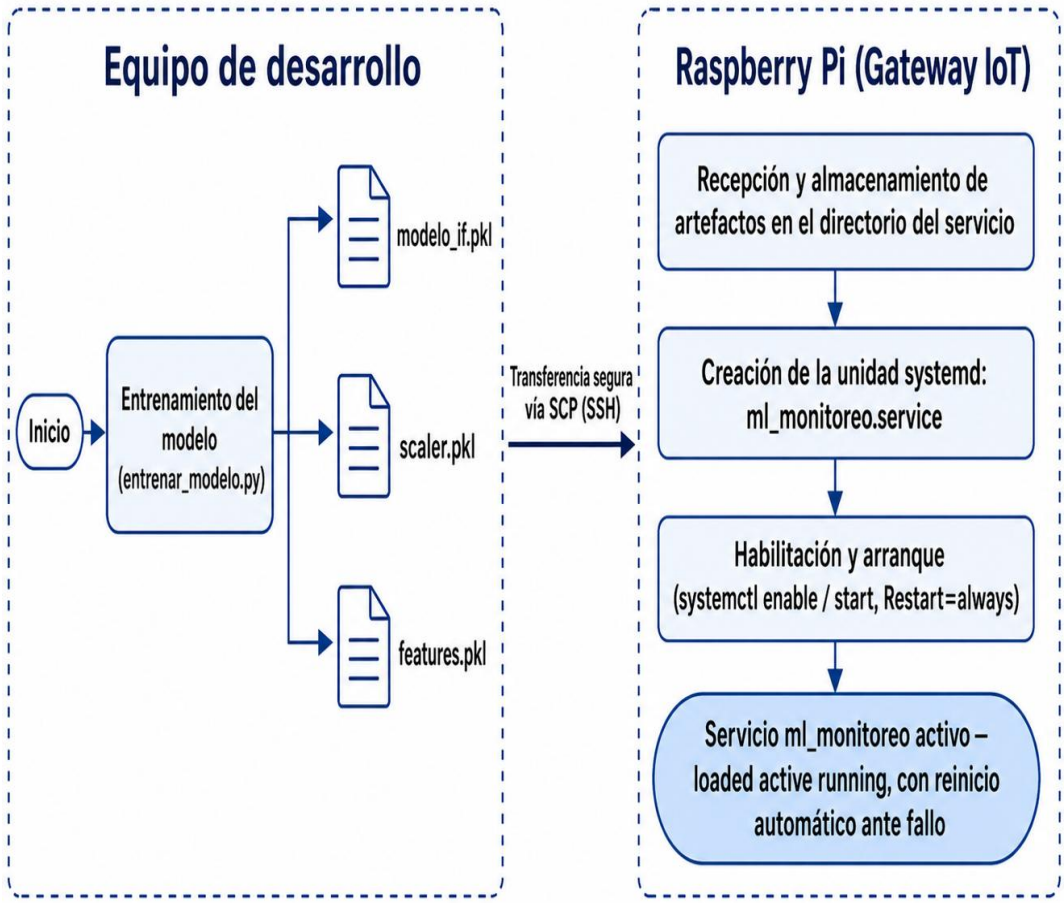


Figura 35 **Proceso de despliegue del modelo Isolation Forest en el Gateway IoT.** Fuente. Elaboración propia.

c.5) Implementación del módulo de detección de anomalías

La implementación del módulo comprende dos componentes principales: el script de entrenamiento entrenar_modelo.py, ejecutado en el equipo de desarrollo, y el microservicio de inferencia ml_service.py, desplegado en el Gateway IoT. Ambos comparten los mismos artefactos serializados (modelo_if.pkl, scaler.pkl y features.pkl), lo que garantiza coherencia entre la etapa de entrenamiento y la operación en producción. El código completo de ambos scripts se presenta en el Anexo D; a continuación, se describe la lógica de operación de cada componente mediante sus respectivos diagramas. La Figura 36 presenta el diagrama de flujo del script entrenar_modelo.py, el cual genera como resultado el modelo Isolation Forest; el diagrama se muestra detallando las etapas de extracción de datos desde MariaDB,

preparación del vector de características, normalización, entrenamiento del modelo y serialización de artefactos.

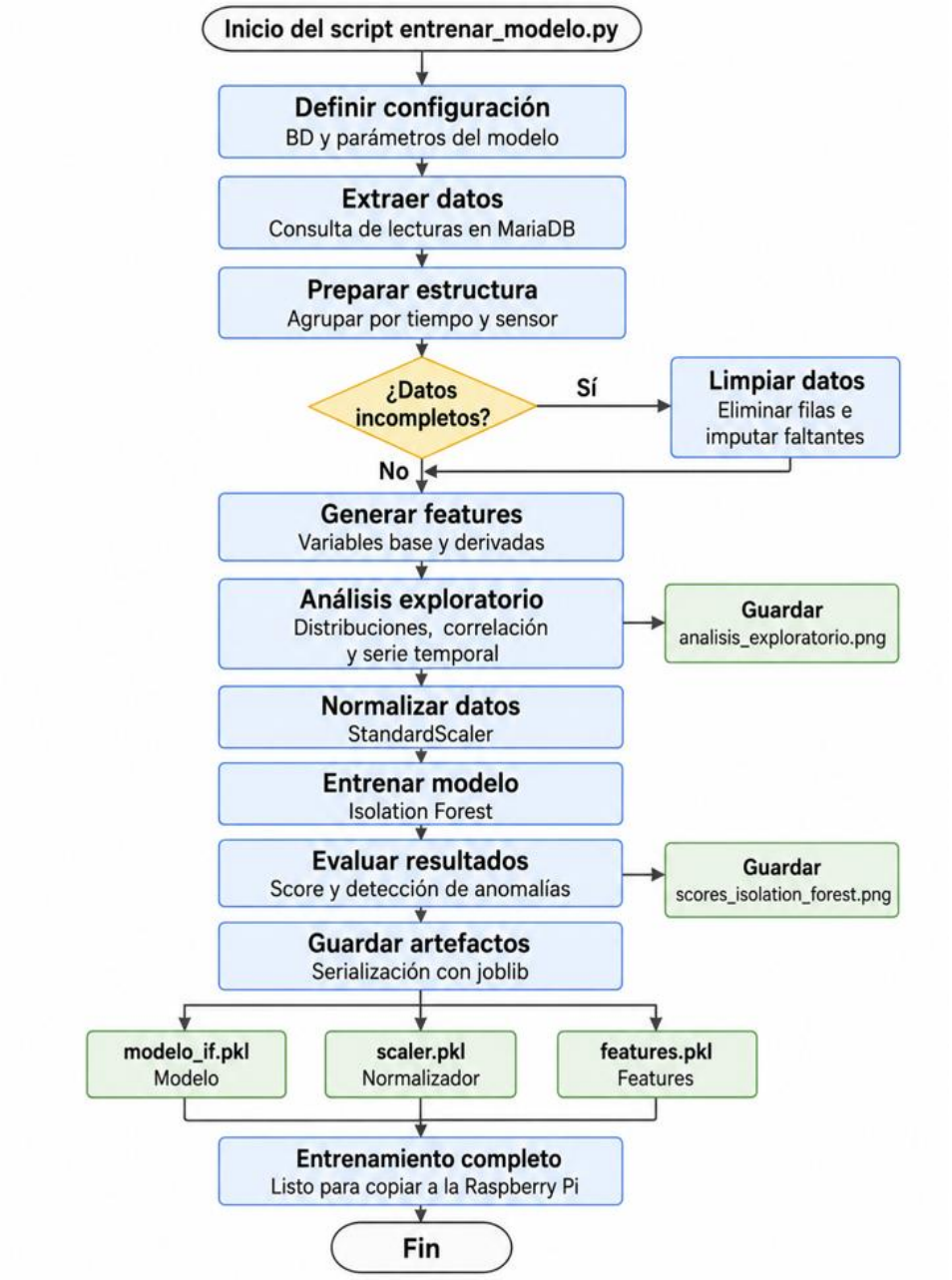


Figura 36 **Diagrama de flujo del script entrenar_modelo.py.** Fuente. Elaboración propia.

La Figura 37 presenta el diagrama de flujo del microservicio ml_service.py. Al iniciar, el servicio carga los artefactos serializados y registra su arranque en el log. En

cada iteración del bucle, consulta en MariaDB el promedio de las lecturas de los últimos 10 minutos; si el resultado contiene menos de tres filas, el ciclo se omite. En caso contrario, construye el vector de 11 características, reemplaza valores vacíos (NaN) por 0.0 de ser necesario, normaliza la entrada mediante scaler.transform(X) y ejecuta la inferencia con decision_function() y predict(). Si predict devuelve -1, se registra una anomalía con nivel WARNING; si devuelve 1, se registra un ciclo normal con nivel INFO. En ambos casos el resultado se persiste en la tabla anomalias_ml y el servicio espera 600 segundos antes de iniciar la siguiente iteración.

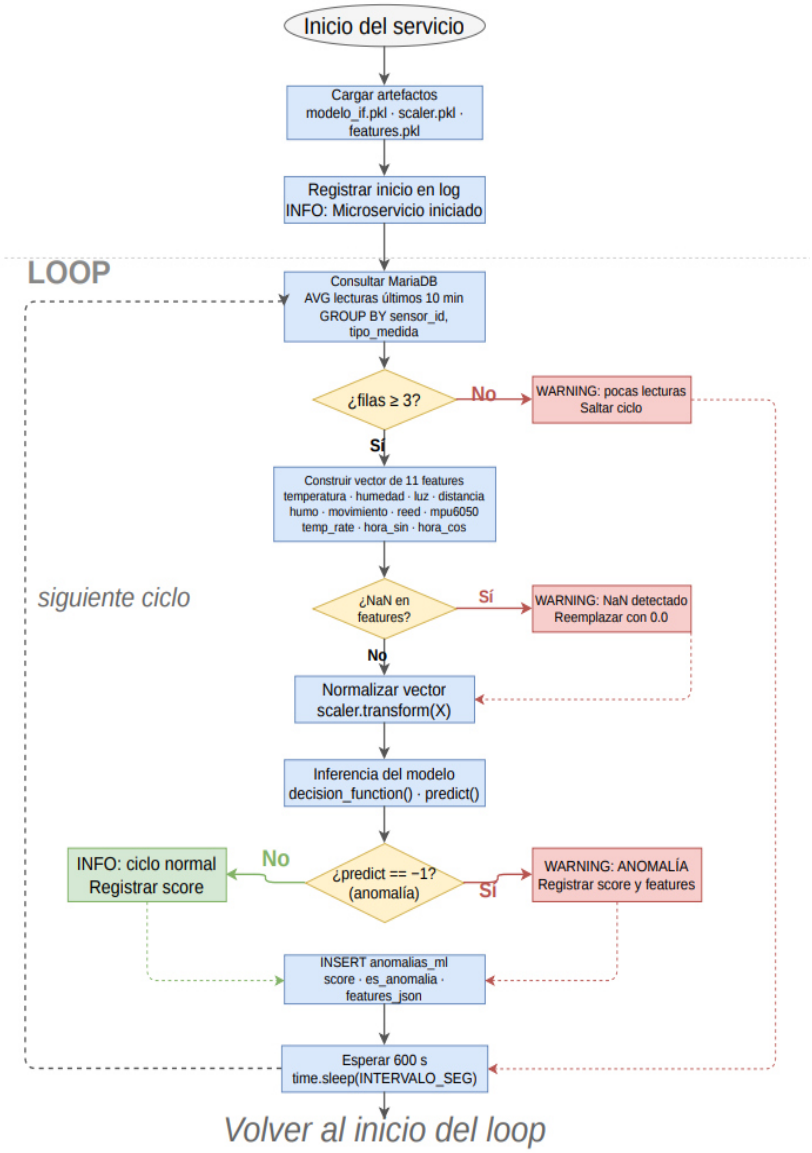


Figura 37 **Diagrama de flujo del microservicio ml_service.py.** Fuente. Elaboración propia.

Los resultados de inferencia se almacenan en la tabla *anomalias_ml*, cuya estructura se describe en la Tabla 20. Esta tabla centraliza toda la información de cada ciclo de evaluación: el identificador del gabinete, la marca temporal de la inferencia, el puntaje continuo generado por el modelo (negativo para anomalías, positivo para comportamiento normal), la clasificación binaria resultante, el vector de características utilizado serializado como JSON para trazabilidad, y el estado de revisión por parte del operador. Esta última columna (revisado) permite al personal de monitoreo confirmar desde la interfaz web qué alertas han sido atendidas, cerrando el ciclo entre detección automática e intervención humana.

Tabla 20 **Estructura de la tabla anomalias_ml**

Campo	Tipo	Descripción
id	INT AUTO_INCREMENT	Identificador único del registro.
gabinete_id	INT	Referencia al gabinete monitoreado.
timestamp	DATETIME	Marca temporal del instante de inferencia.
score_anomalia	FLOAT	Puntaje continuo: negativo = anomalía, positivo = normal.
es_anomalia	TINYINT(1)	Clasificación binaria: 1 = anomalía, 0 = normal.
features_json	TEXT	Vector de características utilizado, serializado como JSON.
revisado	TINYINT(1)	Estado de revisión por el operador: 0 = pendiente, 1 = revisado.

Fuente. Elaboración propia

El módulo ml_routes.py expone tres endpoints REST para consumo desde el frontend, detallados en la Tabla 21 Estos endpoints cubren los tres flujos de interacción definidos en los requisitos: consulta de los registros más recientes con opción de filtrar

solo anomalías pendientes (RF-03), recuperación de la serie temporal de puntajes para el gráfico de tendencia en el panel de monitoreo (dashboard), y actualización del estado de revisión por parte del operador (RF-04).

Tabla 21 **Endpoints REST del módulo de detección de anomalías**

Método	Ruta	Parámetros	Descripción
GET	/api/anomalias	gabinete_id, limit, solo anomalías	Retorna los últimos registros de inferencia. Si solo_anomalias=true, filtra solo anomalías sin revisar.
GET	/api/anomalias/historico	gabinete_id, horas, limit	Serie temporal de puntajes para el gráfico de tendencia (máx. 288 puntos).
PUT	/api/anomalias/<id>/revisar	Body JSON: {"revisado": true/false}	Actualiza el estado de revisión del registro.

Fuente. Elaboración propia

Una vez acumulado un volumen representativo de lecturas en condiciones de operación real, se ejecutó un ciclo de reentrenamiento del modelo utilizando los datos del período 30 de abril - 8 de mayo de 2026. Este ciclo constituye la continuación natural de la fase de Despliegue del proceso de metodología adoptada para la investigación: CRISP-DM y permitió ajustar el modelo a los patrones térmicos, de iluminación y de acceso propios del gabinete físico instalado. La Tabla 22 compara los hiperparámetros del modelo inicial respecto al modelo reentrenado; los ajustes más relevantes son el incremento de n_estimators de 100 a 200 para responder al mayor volumen de datos reales, y el aumento de contamination de 0.02 a 0.05, dado que la tasa de eventos inusuales observada en operación resultó superior a la estimada inicialmente.

Tabla 22 **Comparación de hiperparámetros: modelo inicial vs. modelo reentrenado**

Hiperparámetro	Modelo inicial (marzo)	Modelo reentrenado (mayo)	Justificación del ajuste
n_estimators	100	200	Mayor volumen de datos reales requiere más árboles para estabilizar el puntaje.
contamination	0.02	0.05	La tasa de eventos inusuales observada en operación real fue superior a la estimada inicialmente.
random_state	42	42	Sin cambio - garantiza reproducibilidad.

Fuente. Elaboración propia

c.6) Pruebas de verificación del modelo de detección de anomalías

Previo a la integración con la plataforma de software de la Etapa 5, se verificó el funcionamiento del microservicio ml_service.py utilizando datos reales del gabinete, con el fin de asegurar su estabilidad antes de ponerlo en contacto con los demás componentes del sistema. La Figura 38 muestra la salida del archivo ml_service.log durante la operación, donde se observan ciclos normales registrados con nivel INFO y su respectivo puntaje positivo, así como eventos registrados con nivel WARNING correspondientes a condiciones clasificadas como anomalías, identificadas por puntajes negativos. Estos registros confirman que el microservicio ejecuta correctamente el ciclo de inferencia, diferencia entre comportamiento normal y anómalo, y mantiene trazabilidad completa de cada evaluación, lo cual resulta esencial para poder auditar posteriormente el comportamiento del modelo ante distintos escenarios operativos. Adicionalmente, la evaluación visual del comportamiento del modelo sobre datos reales se presenta en el Capítulo V, Figura 85, donde se profundiza en los resultados obtenidos durante esta etapa de verificación.

```
admin@monitor-local: ml_service.log
$ grep -E 'INFO|WARNING' /home/admin/ml_monitoreo/ml_service.log | tail -20
2026-05-07 22:00:00 [INFO] Normal - score: 0.0812
2026-05-07 22:10:00 [INFO] Normal - score: 0.0595
2026-05-07 22:20:00 [INFO] Normal - score: 0.0763
2026-05-07 22:30:00 [INFO] Normal - score: 0.0508
2026-05-07 22:40:00 [INFO] Normal - score: 0.0863
2026-05-07 22:50:00 [INFO] Normal - score: 0.0668
2026-05-07 23:00:00 [INFO] Normal - score: 0.0923
2026-05-07 23:10:00 [INFO] Normal - score: 0.0415
2026-05-07 23:20:00 [INFO] Normal - score: 0.0439
2026-05-07 23:30:00 [INFO] Normal - score: 0.0729
2026-05-07 23:40:00 [WARNING] ANOMALIA DETECTADA - score: -0.1394
2026-05-07 23:50:00 [INFO] Normal - score: 0.0600
2026-05-08 00:00:00 [INFO] Normal - score: 0.1054
2026-05-08 00:10:00 [INFO] Normal - score: 0.0520
2026-05-08 00:20:00 [WARNING] ANOMALIA DETECTADA - score: -0.1785
2026-05-08 00:30:00 [INFO] Normal - score: 0.0677
2026-05-08 00:40:00 [INFO] Normal - score: 0.0856
2026-05-08 00:50:00 [INFO] Normal - score: 0.0438
2026-05-08 01:00:00 [WARNING] ANOMALIA DETECTADA - score: -0.1305
```

Figura 38 **Salida del log de ml_service.py.** Fuente. Elaboración propia.

d) Etapa 4: Gateway IoT Basado en Raspberry Pi

Esta etapa establece la infraestructura del Gateway IoT sobre la Raspberry Pi, configurando la conectividad y los servicios necesarios para operar como nodo central de comunicaciones del gabinete, sin depender de infraestructura externa. Como se muestra en la Figura 39, el Gateway habilita una red WiFi con SSID propio para la comunicación con la UAC mediante WiFi/MQTT, e incorpora un broker MQTT en 192.168.4.1, puerto 1883. Asimismo, mantiene conexión con la red local mediante Ethernet o WiFi para el acceso de los equipos internos. La leyenda diferencia la comunicación WiFi/MQTT de la UAC y la comunicación hacia la red local.

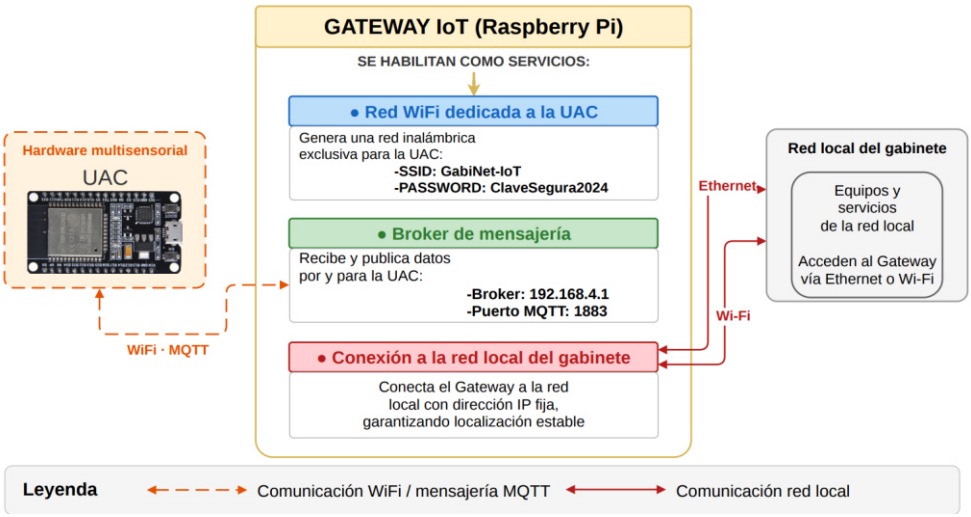


Figura 39 **Arquitectura de red del Gateway IoT.** Fuente. Elaboración propia.

d.1) Requerimientos del Gateway IoT

Los requerimientos se formularon a partir del rol central que cumple el Gateway IoT como nodo de comunicaciones entre la UAC, los servicios locales y la red LAN principal. En esta etapa, la restricción transversal es que las funciones esenciales del sistema deben operar de forma autónoma, independientemente de la disponibilidad de una infraestructura de red externa. Por ello, se consideraron requerimientos asociados a la generación de una red WiFi propia, asignación automática de direccionamiento IP, ejecución local del broker MQTT, configuración de la interfaz Ethernet y recuperación automática de servicios ante reinicios. Asimismo, se incluyeron criterios no funcionales orientados a la segregación de red y al control de acceso mediante cifrado. La Tabla 23 presenta los requerimientos funcionales y no funcionales identificados para el Gateway IoT.

Tabla 23 **Requerimientos funcionales del Gateway IoT**

ID	Tipo	Descripción
RF-01	Funcional	El Gateway IoT debe generar una red WiFi propia a la que la UAC se conecte automáticamente.
RF-02	Funcional	El Gateway IoT debe asignar dinámicamente una dirección IP a la UAC.
RF-03	Funcional	El Gateway IoT debe ejecutar un *broker* MQTT accesible desde la red WiFi dedicada a la UAC.
RF-04	Funcional	La interfaz Ethernet del Gateway IoT debe operar con una dirección estática y/o automática por DHCP.
RF-05	Funcional	Los servicios de red y mensajería deben recuperar su estado operativo automáticamente ante cualquier reinicio del Gateway IoT.
RF-06	Funcional	El Gateway IoT debe ejecutar una base de datos relacional local (MariaDB) que

		almacene las lecturas de sensores, los registros de anomalías y las configuraciones del sistema, accesible internamente por los módulos de las Etapas 3 y 5 mediante TCP/3306.
RNF-01	No Funcional	La red WiFi del Gateway IoT debe estar separada de la red LAN principal
RNF-02	No Funcional	La red AP dedicada a la UAC debe operar con cifrado WPA2-PSK para restringir el acceso mediante credenciales exclusivas del sistema.

Fuente. Elaboración propia

d.2) Selección de tecnologías

La plataforma seleccionada es la Raspberry Pi 4 Model B ejecutando Raspberry Pi OS de 64 bits (Debian Bookworm), por su compatibilidad nativa con el hardware ARM64, soporte directo para la gestión de interfaces de red y disponibilidad de paquetes estables para los servicios requeridos en los repositorios oficiales de Debian. Alternativas como Ubuntu Server para ARM fueron descartadas por introducir capas que dificultan el control directo sobre las interfaces de red virtuales requeridas. La Tabla 24 resume las especificaciones adoptadas.

Tabla 24 **Resumen de componentes para el Gateway IoT**

Componente	**Especificación**
Plataforma	Raspberry Pi 4 modelo B
Almacenamiento	microSD 64 GB
Arquitectura	ARM64
Sistema operativo	Raspberry Pi OS 64-bit (Debian Bookworm)
Interfaz LAN	eth0 - IP estática/dinámica

Interfaz física WI-FI	wln0 - IP estática/dinámica
Interfaz AP	uap0 - modo punto de acceso

Fuente. Elaboración propia

d.3) Diseño de los servicios del Gateway IoT

El Gateway IoT ejecuta tres servicios base de forma simultánea y persistente: hostapd con dnsmasq, que crean la red inalámbrica dedicada WPA2 para la UAC y asignan dinámicamente su dirección IP dentro del rango 192.168.4.10-192.168.4.50; Mosquitto, que centraliza la recepción de la telemetría publicada por la UAC y la distribuye a los suscriptores locales; y MariaDB, que actúa como capa de persistencia central, almacenando las lecturas de sensores, los registros de anomalías y las configuraciones del sistema, accedida por los módulos de las Etapas 3 y 5 mediante el conector pymysql. Los cuatro servicios se registran como unidades systemd con directiva Restart=always, de modo que ante un fallo del proceso el sistema operativo lo relanza sin intervención manual cumpliendo de esta manera también los requerimientos de la etapa. La Tabla 25 resume cada servicio con su interfaz de escucha y los parámetros de operación definidos.

Tabla 25 **Servicios principales del Gateway IoT basado en Raspberry Pi**

Servicio	Interfaz / Puerto	Función
hostapd	Uap0	Gestiona el punto de acceso WiFi WPA2. Define el SSID y las credenciales de acceso a la red dedicada del módulo IoT.
dnsmasq	uap0 - DHCP UDP/67	Asigna dinámicamente una dirección IP a la

		UAC dentro del rango 192.168.4.2–192.168.4.20 al momento de su conexión al AP.
Mosquitto	0.0.0.0 - TCP/1883	Broker MQTT que recibe y envía las publicaciones de la UAC y las distribuye a los suscriptores locales.
MariaDB	127.0.0.1 - TCP/3306	Base de datos relacional local. Almacena lecturas, anomalías y configuraciones; accedida internamente por los módulos de ML (Etapa 3) y el backend Flask (Etapa 5).

Fuente. Elaboración propia

Con los servicios definidos, se estableció la configuración de red del Gateway IoT en dos planos de comunicación independientes, como se muestra en la Figura 40. El primero es la interfaz cableada eth0, asignada con dirección estática o dinámica según el entorno LAN del cliente, a través de la cual el Gateway se integra con la red local y accede a Internet cuando está disponible. El segundo es la interfaz virtual uap0, que opera como punto de acceso WiFi dedicado a la UAC con cifrado WPA2-PSK, dirección IP fija 192.168.4.1 y rango DHCP exclusivo 192.168.4.10-192.168.4.50; el ESP32 se conecta a este AP de forma automática al arrancar, sin intervenir en el segmento LAN principal. Esta separación garantiza que el tráfico de telemetría IoT no atraviese la red del cliente y que el sistema opere con independencia de la disponibilidad de la LAN o de Internet y así mismo se logran cumplir parte de los

requerimientos funcionales de esta etapa. Los parámetros completos de configuración de interfaces, hostapd y dnsmasq se detallan en el Anexo E.

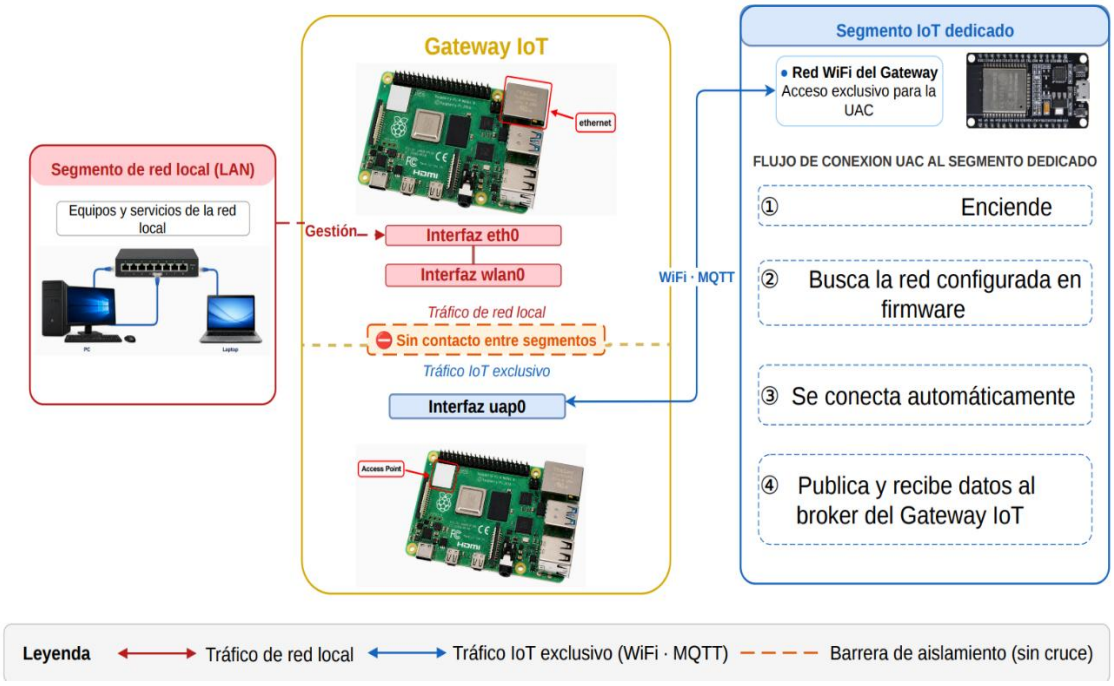


Figura 40 **Separación de interfaces de red del Gateway: eth0 en LAN y UAP como AP dedicado.** Fuente. Elaboración propia.

Como capacidades complementarias, el Gateway incorpora tres funciones de soporte que amplían su operación de forma modular: supervisión de equipos de red mediante SNMP, envío de notificaciones por correo electrónico mediante SMTP, e interacción con periféricos locales a través de GPIO, I²C y SPI. La Tabla 26 resume estas capacidades con su base de soporte y función.

Tabla 26 **Capacidades complementarias del Gateway IoT**

Capacidad	Base de soporte	Función
Monitoreo sistema	SNMP	Supervisión del estado del Gateway y equipos a través de red como UPS y PDU.

Notificación	SMTP	Envío de alertas y eventos hacia destinatarios configurados.
Interacción con periféricos	GPIO / I2C / SPI	Comunicación con dispositivos locales (LCD u otros).

Fuente. Elaboración propia

d.4) Implementación del Gateway IoT

La implementación se desarrolló en tres fases sucesivas. En la primera, se configuraron las dos interfaces de red en sus segmentos independientes: eth0 en la red LAN con dirección estática o dinámica según el entorno, y uap0 como punto de acceso WiFi dedicado a la UAC. En la segunda, se desplegaron los servicios de mensajería y persistencia: Mosquitto quedó a la escucha en TCP/1883 sobre todas las interfaces, y MariaDB fue inicializada con el esquema de base de datos que define las tablas de lecturas, anomalías, usuarios y configuraciones del sistema.

La ingesta de telemetría desde el broker hacia MariaDB es responsabilidad del backend Flask desarrollado en la Etapa 5; el Gateway provee la infraestructura como capa de soporte. En la tercera fase, se habilitaron las capacidades complementarias: el agente SNMP para supervisión de equipos de red, el servicio SMTP para alertas y la comunicación con periféricos locales. El detalle de los archivos de configuración de cada componente se presenta en el Anexo G.

La Figura 41 muestra la verificación del estado operativo de los tres servicios principales mediante el comando *systemctl list-units --type=service --state=running*, herramienta del gestor de servicios systemd de Linux que lista en tiempo real todas las unidades de servicio activas en el sistema. El filtro grep -E "mosquitto|dnsmasq|hostapd" acota la salida a los tres servicios de interés, confirmando que dnsmasq, hostapd y mosquitto se encuentran simultáneamente en

estado activo (*loaded active running*), lo que acredita la operación conjunta de la red AP dedicada, la asignación dinámica de IP y el broker MQTT sobre el Gateway IoT.

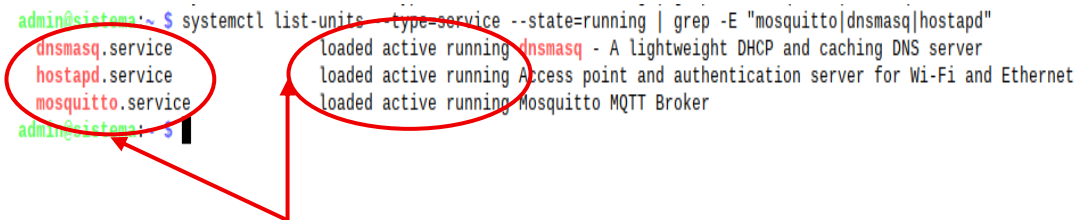


Figura 41 **Estado operativo de los servicios del Gateway verificado mediante comando systemctl.** Fuente. Elaboración propia.

La Figura 42 verifica el estado operativo de los servicios que sustentan las capacidades complementarias del Gateway. El comando *systemctl list-units* filtrado por los servicios de la capa de aplicación confirma que *scmp-api* que gestiona internamente las rutas SNMP y el envío de notificaciones SMTP, *scmp-tft*, encargado de la comunicación con el display TFT mediante SPI/I²C, *nginx* y *mariadb* se encuentran simultáneamente en estado *loaded active running*, acreditando la disponibilidad de las tres capacidades descritas en la Figura 42 La validación funcional de estos elementos se presenta junto con la Etapa 5.

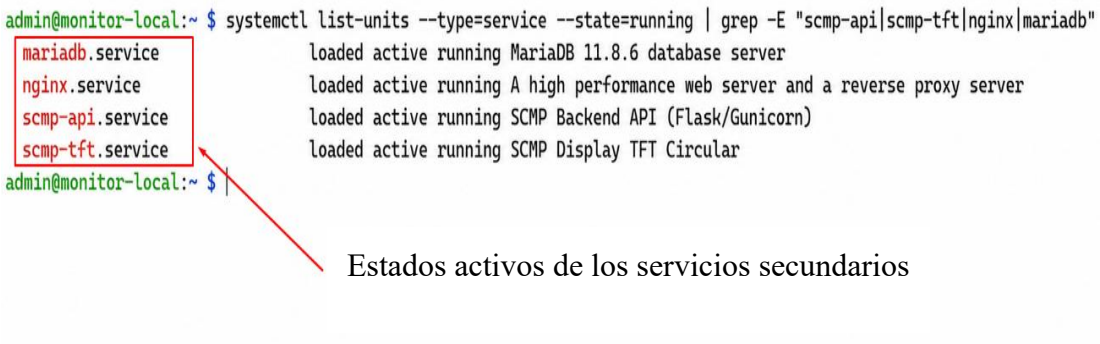


Figura 42 **Estado operativo de los servicios complementarios del Gateway IoT verificado mediante comando systemctl.** Fuente. Elaboración propia.

e) Etapa 5: Plataforma de Software para Monitoreo y Control

Esta etapa incorpora la capa de aplicación del sistema, responsable de exponer la información del gabinete al usuario mediante una base de datos relacional en MariaDB, un backend en Flask y un frontend en ReactJS, conformando así la interfaz principal entre el sistema técnico y el usuario final. El backend recibe telemetría vía MQTT desde la UAC, procesa las lecturas, evalúa reglas de alerta, gestiona usuarios por rol y emite comandos hacia los actuadores cuando corresponde, actuando como el núcleo lógico que articula la comunicación entre los dispositivos y la interfaz visible al usuario. Por otro lado, el frontend expone estas funciones mediante una interfaz web accesible desde el navegador, permitiendo la visualización de dashboards, la gestión de alertas y la administración de usuarios, facilitando así una interacción intuitiva sin requerir conocimientos técnicos adicionales. La implementación respetó el orden persistencia - backend - frontend, debido a que las decisiones de diseño de cada capa condicionan el comportamiento de la superior, evitando reprocesos y asegurando coherencia entre los datos almacenados y su representación final. La Figura 43 presenta esta arquitectura general, mostrando la relación entre la capa de persistencia, la capa de procesamiento y la capa de presentación.

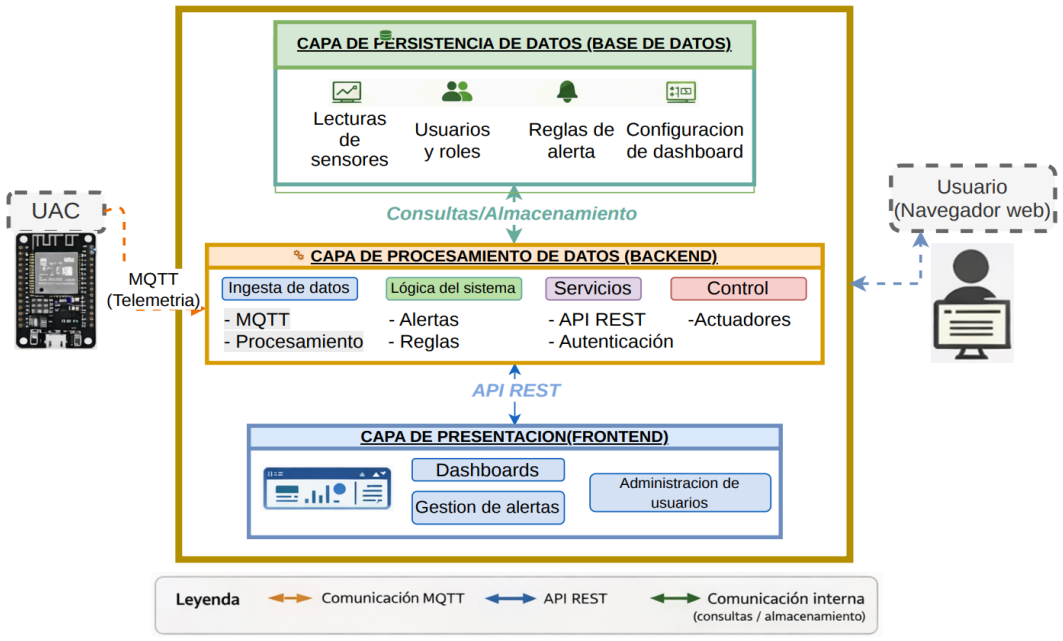


Figura 43 **Arquitectura general de la plataforma de software para monitoreo y control.** Fuente. Elaboración propia.

e.1) Requerimientos de la plataforma de software

Se identificaron cinco requerimientos funcionales que incluyen recepción de telemetría, visualización, gestión de usuarios por rol, generación de alertas y control de actuadores; cinco requerimientos no funcionales vinculados a disponibilidad local, escalabilidad, seguridad de credenciales, persistencia de servicios y usabilidad. Los requerimientos RNF-01 y RNF-04 son transversales al sistema por condicionar tanto el despliegue del backend como la configuración de la base de datos. La Tabla 27 presenta la clasificación completa.

Tabla 27 **Requerimientos de la plataforma de software**

ID	Tipo	Descripción
RF-01	Funcional	La plataforma debe recibir, procesar y almacenar datos de los sensores en tiempo real, garantizando trazabilidad temporal de cada registro.
RF-02	Funcional	La plataforma debe proveer una interfaz web con paneles configurables para la visualización de los datos sensados, accesible desde la red local e Internet.
RF-03	Funcional	Módulo de gestión de cuentas con perfiles diferenciados (administrador, operador y lector) y restricciones funcionales por rol.
RF-04	Funcional	Evaluación automática de condiciones sobre variables registradas, con generación de notificaciones por correo electrónico ante superación de umbrales configurados.
RF-05	Funcional	Emisión de comandos de control hacia actuadores del sistema desde la interfaz web, previa validación de permisos por rol.
RNF-01	No funcional	El sistema debe operar de forma continua dentro del Gateway IoT,

		manteniendo disponibilidad local ante fallos de conectividad externa.
RNF-02	No funcional	Arquitectura que permita incorporar nuevos puntos de monitoreo sin rediseñar la plataforma.
RNF-03	No funcional	Contraseñas almacenadas con mecanismo de cifrado irreversible. Autenticación sin estado mediante tokens firmados con expiración controlada.
RNF-04	No funcional	El *backend*, el *broker* MQTT y la base de datos deben iniciar automáticamente ante reinicios del Gateway IoT.
RNF-05	No funcional	La interfaz debe permitir la ejecución de las operaciones principales en un máximo de tres interacciones desde cualquier vista, sin requerir conocimientos de programación.

Fuente. Elaboración propia

e.2) Selección de tecnologías

La selección de tecnologías se orientó a garantizar compatibilidad entre componentes, disponibilidad de librerías para integración con MQTT y MariaDB, y facilidad de despliegue en el entorno Raspberry Pi OS del Gateway IoT. Las tecnologías seleccionadas y sus versiones se fundamentaron en la sección II.3.4 del marco teórico. La Tabla 28 resume las tecnologías adoptadas con su rol específico dentro del sistema.

Tabla 28 **Resumen de tecnologías de la plataforma de software**

Tecnología	Versión	Rol en el sistema
Python	3.9.2	Lenguaje base del *backend*

Flask	3.1.2	Framework del *backend*
MariaDB	10.5.29	Base de datos relacional
ReactJS	18.3.1	Framework del *frontend*
Mosquitto	2.0.11	Broker MQTT Fuente: Broker MQTT compartido con la Etapa 4. Se lista aquí por su rol como fuente de datos del backend Flask.
***Paho*-MQTT**	2.1.0	Cliente MQTT del *backend*
mysql-connector-python	9.4.0	Conector de base de datos
PyJWT + Flask-JWT-Extended	2.10.1 / 4.7.1	Autenticación por token

Fuente. Elaboración propia

e.3) Diseño de la plataforma de software

El diseño de la plataforma se desarrolló siguiendo el patrón de arquitectura en tres capas: persistencia (Base de datos), procesamiento (Backend) y presentación (Frontend). Cada capa se diseñó de forma independiente considerando sus dependencias con las capas siguientes.

i. Base de datos

La base de datos *monitoring_iot* se estructuró bajo un modelo relacional organizado en cuatro grupos funcionales: gestión de usuarios y acceso, alertas y notificaciones, datos operativos e historial, y seguridad y configuración de red. Esta

organización distingue tablas estructurales de escritura poco frecuente y tablas operativas de alta frecuencia, criterio que orientó las decisiones de indexación para equilibrar las escrituras continuas con las consultas históricas realizadas desde el frontend. Como se muestra en la Figura 44, cada grupo reúne tablas asociadas a una función específica del sistema; por ejemplo, usuarios, sesiones y dashboard_widgets para la gestión de acceso e interfaz; reglas_alerta, alertas y notificaciones para la gestión de eventos; lecturas, dispositivos, predicciones_ml y auditoria para el registro operativo; y redes_autorizadas, configuracion_sistema, tokens_api y logs_acceso para seguridad y configuración. Todas estas tablas se integran sobre la base de datos central *monitoring_iot*, implementada en MariaDB/InnoDB. El detalle técnico de cada tabla y las decisiones de integridad referencial se presentan en el Anexo E.

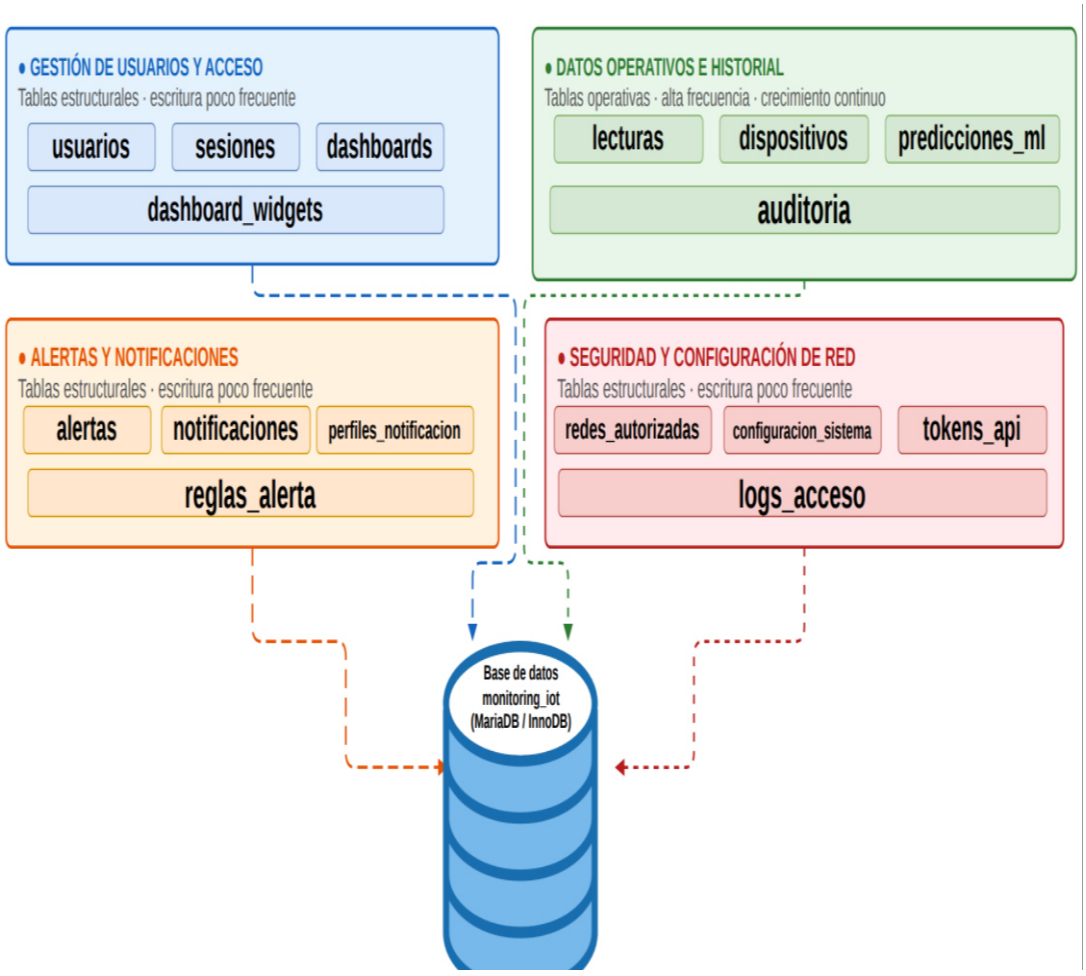


Figura 44 **Diagrama de flujo de módulos de la base de datos.** Fuente. Elaboración propia.

ii. Backend

El backend se organizó en módulos independientes con responsabilidades derivadas de los requerimientos funcionales: recepción y persistencia de telemetría, gestión de usuarios y control de acceso, motor de alertas y notificaciones y control de actuadores. El flujo principal sigue cinco pasos secuenciales: autenticación de cada solicitud, distribución por la API REST, recepción MQTT de telemetría, evaluación de reglas de alerta y supervisión periódica de equipos de red del gabinete. La Figura 45 presenta este flujo con sus fuentes de entrada, módulos internos y salidas.

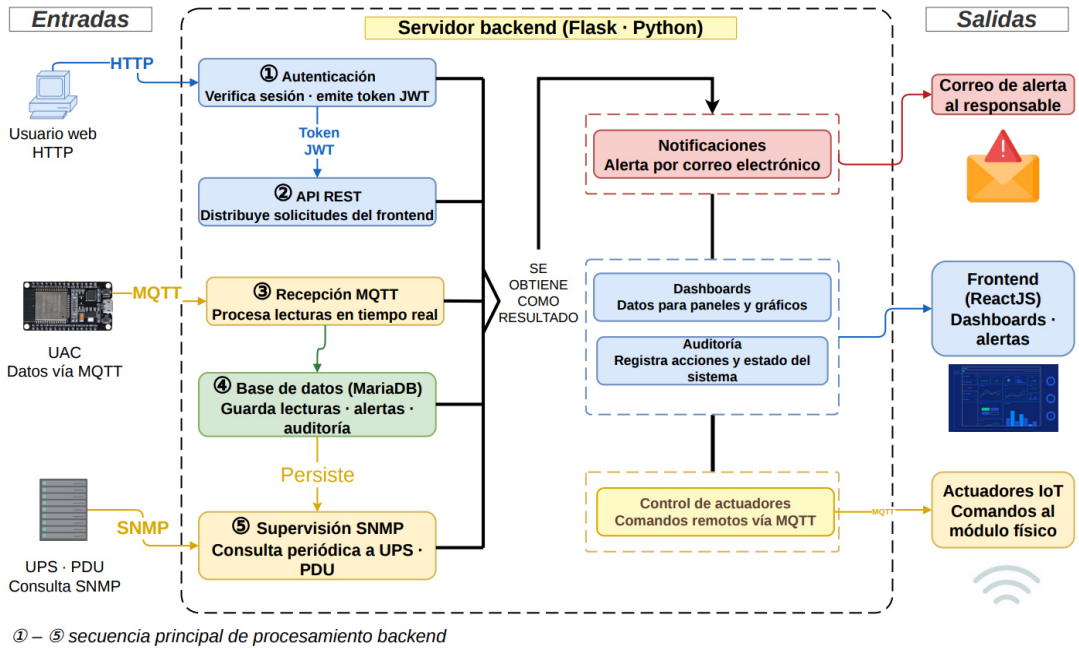


Figura 45 **Diagrama de flujo de módulos del backend.** Fuente. Elaboración propia.

El diseño del backend se complementa con tres definiciones transversales que condicionan el comportamiento del sistema en su conjunto, conformadas por el control de acceso por roles, orientado a regular la interacción del usuario con la plataforma; la especificación de la API REST, encargada de estructurar la comunicación entre el backend y el frontend; y la organización de los tópicos MQTT, destinada al intercambio de datos y comandos entre el backend y el hardware.

El control de acceso por roles se gestiona mediante tres perfiles. Tras un inicio de sesión exitoso, el backend emite un token firmado con la identidad y rol del usuario,

que debe adjuntarse en cada solicitud posterior para que el sistema restrinja o habilite funciones de forma automática. La Tabla 29 detalla las funcionalidades disponibles por perfil.

Tabla 29 **Resumen de roles de la plataforma de software**

Funcionalidad	Administrador	Operador	Visualizador
Visualización de dashboards	SÍ	SÍ	SÍ
Consulta de historial de sensores	SÍ	SÍ	SÍ
Configuración de alertas	SÍ	SÍ	NO
Control de actuadores	SÍ	SÍ	NO
Gestión de usuarios	SÍ	NO	NO
Configuración SNMP	SÍ	NO	NO

Fuente. Elaboración propia

La API REST expone los servicios del backend al frontend y los organiza por dominios funcionales, así todas las solicitudes pasan por la capa de autenticación antes de ser procesadas, en la Tabla 30 se presentan los ocho módulos de la API con las operaciones que expone cada uno.

Tabla 30 **Módulos API REST utilizado por el backend**

Módulo	Operaciones	Función
Autenticación y usuarios	Inicio de sesión, creación, modificación y eliminación de cuentas	Gestiona el acceso a la plataforma y la administración de usuarios
Sensores	Consulta de telemetría actual e histórica por rango de fechas	Entrega las lecturas almacenadas al *frontend* para su visualización

Actuadores	Envío de comandos al módulo IoT	Permite controlar remotamente los dispositivos físicos del gabinete
Alertas	Gestión de reglas de umbral y perfiles de notificación por correo	Configura las condiciones que activan las alertas automáticas del sistema
Seguridad	Registro de redes autorizadas y consulta del historial de acciones	Controla el acceso perimetral y mantiene trazabilidad de operaciones
Dashboard	Creación, consulta y eliminación de widgets por usuario	Permite personalizar los paneles de visualización de cada usuario
Estado del sistema	Consulta del estado operativo, disponibilidad y dirección IP del Gateway	Permite que el *frontend* verifique la disponibilidad del *backend* en tiempo real
Apariencia	Configuración visual y carga de recursos	Gestiona la personalización de la interfaz por usuario

Fuente. Elaboración propia

Finalmente, los tópicos MQTT definen el canal de comunicación entre el módulo IoT y el backend sobre el broker del Gateway. La Tabla 31 describe los tres tópicos implementados con su dirección de flujo y contenido.

Tabla 31 **Tópicos utilizados en la plataforma de software**

Tópico	Dirección	Contenido
sensores/datos	Gateway - *Backend*	Variables ambientales del gabinete: temperatura, humedad, estado de puerta, presencia, vibración y distancia
esp32/+/status	Módulo IoT - *Backend*	Estado de conectividad del nodo: red WiFi, conexión MQTT y última actualización

uad/+/net	Módulo IoT - *Backend*	Parámetros de red activos: SSID, dirección IP y nivel de señal WiFi

Fuente. Elaboración propia

iii. Frontend

El frontend fue diseñado como aplicación de página única (SPA), permitiendo mantener el estado de sesión y las suscripciones en tiempo real sin recargas de página. La autenticación opera como único punto de entrada con rutas protegidas por rol. La interfaz se organiza en módulos comunes accesibles a todos los roles tales como dashboards, sensores, gabinete, alertas, actuadores y módulos exclusivos del administrador como gestión de usuarios, seguridad, apariencia y configuración de red. La lógica de selección de endpoint evalúa en tiempo de ejecución si el acceso proviene de la red local o de Internet, conectándose al backend correspondiente de forma transparente. La Figura 46 presenta la estructura de navegación por rol.

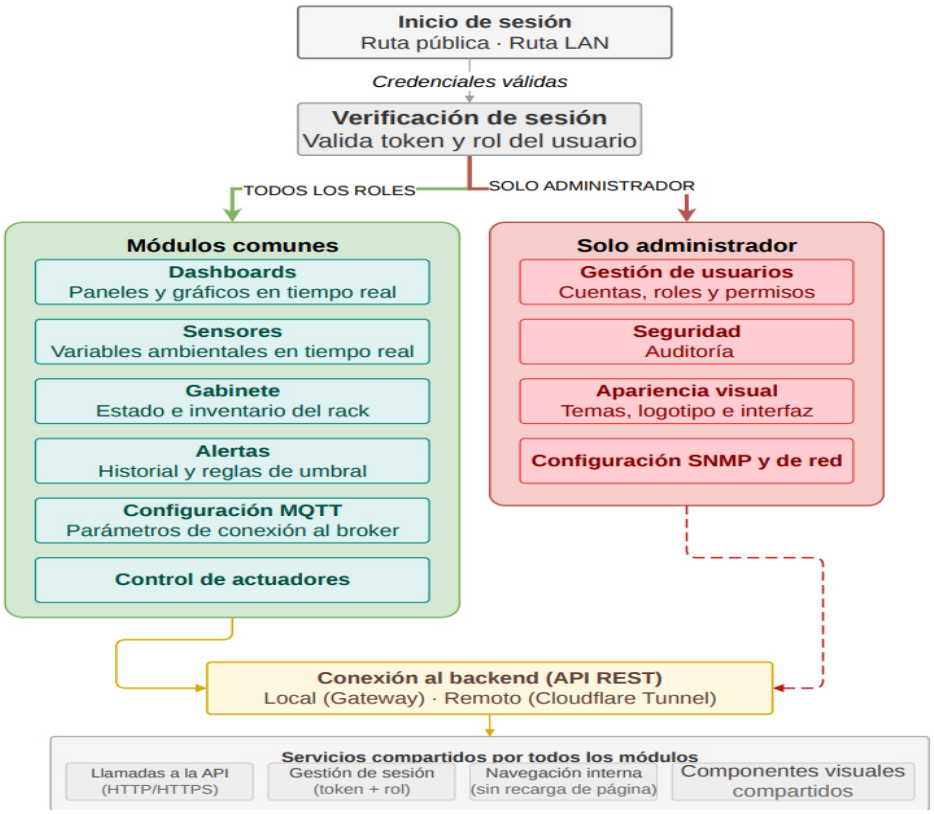


Figura 46 **Diagrama de flujo de módulos del frontend.** Fuente. Elaboración propia.

Las figuras siguientes presentan los mockups de las vistas principales del sistema, elaborados como referencia para la implementación, organizados según el esquema de acceso por roles, en el Anexo H se detalla el código HTML del mockup desarrollado para el diseño del frontend. La Figura 47 muestra el diseño propuesto para el tablero de monitoreo (dashboard) principal, el cual reúne en una sola vista los indicadores más relevantes del gabinete, como temperatura, humedad, estado de puerta, alertas activas y conexión del Gateway. También incorpora gráficos de comportamiento reciente y una tabla de últimas lecturas, facilitando la supervisión general del sistema desde la pantalla inicial.

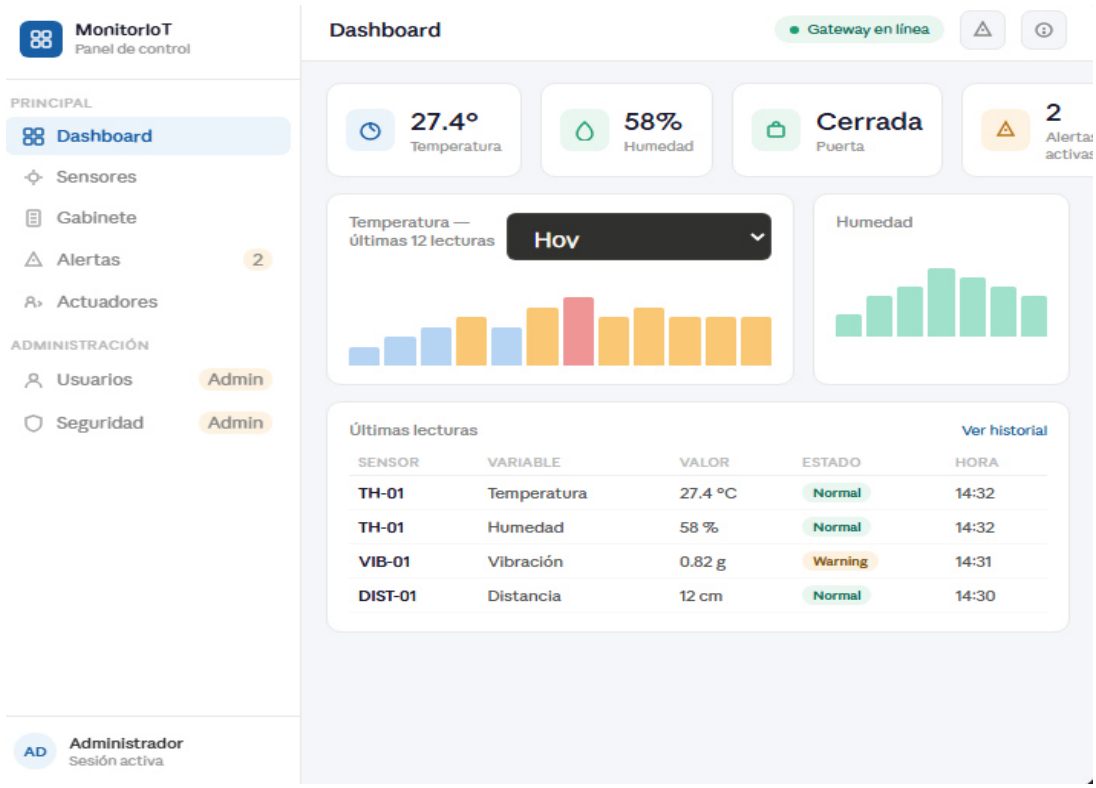


Figura 47 **Mockup de la interfaz principal dashboard.** Fuente. Elaboración propia.

La Figura 48 presenta la pestaña de Gabinete, en donde se configura la iluminación de la tira RGB, así como el apartado de SNMP, permitiendo personalizar en base a alertas el color de la tira RGB entre otros. Adicionalmente de una vista del rack del gabinete para poder documentar los sensores colocados.

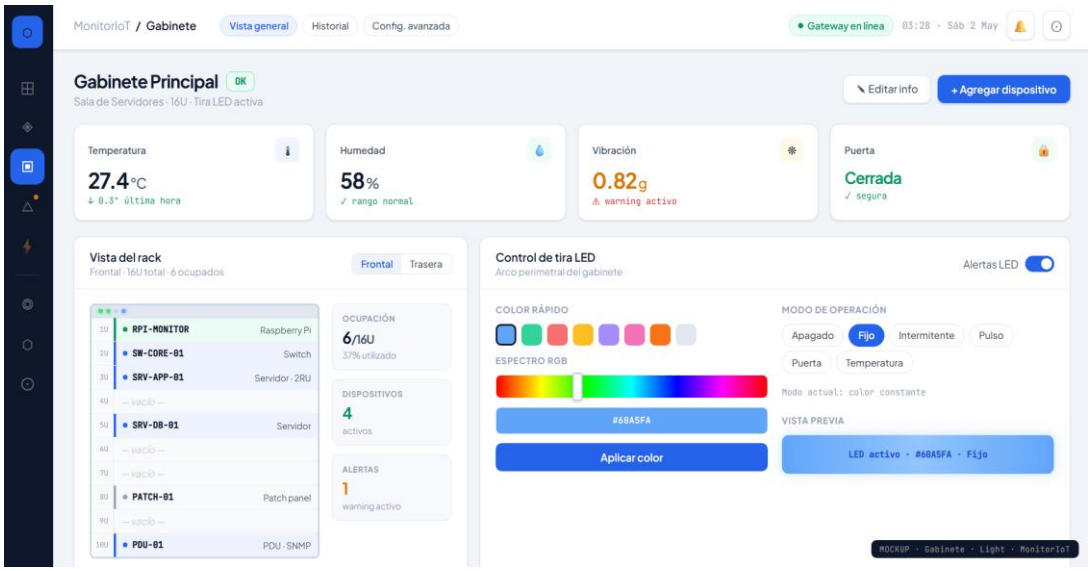


Figura 48 **Mockup gestión de actuadores y equipos de gabinete.** Fuente. Elaboración propia.

En la Figura 49 se presenta el apartado de configuración de red donde se configura tanto la tarjeta de red ethernet del Gateway IoT como de la tarjeta WiFi todo ello para la gestión del equipo, así como su conexión a internet para el acceso remoto.

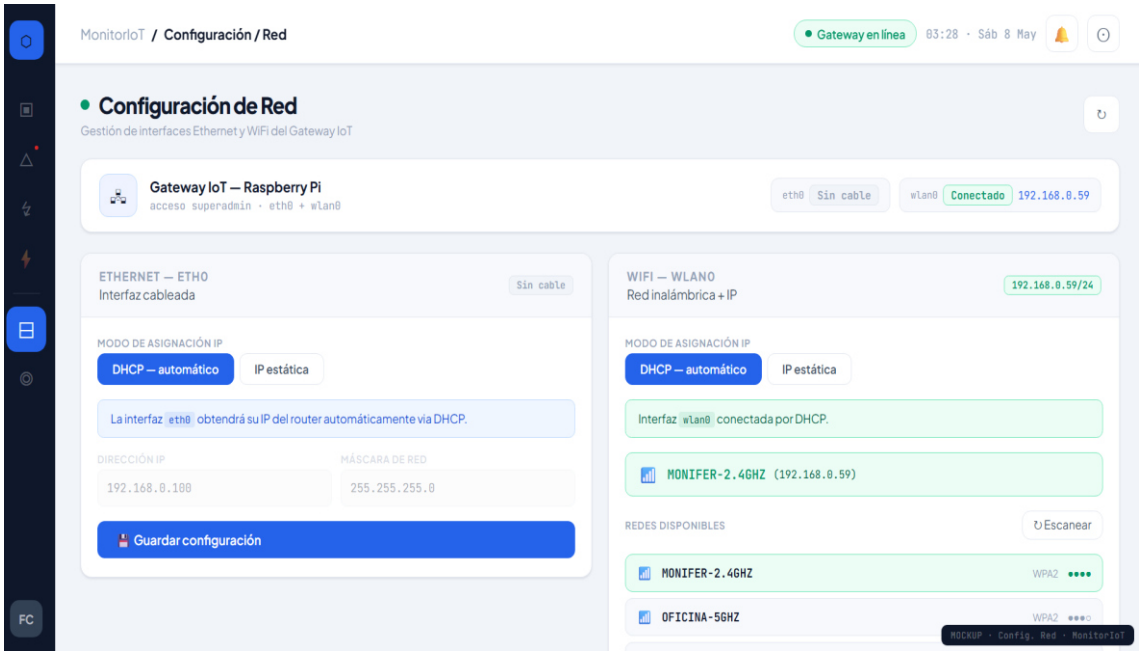


Figura 49 **Mockup gestión de parámetros de red del Gateway IoT.** Fuente. Elaboración propia.

La Figura 50 presenta el apartado de Alertas, donde se puede configurar perfiles SMTP para poder enviar correos de alertas, asimismo se presenta las alertas por sensores donde adicional a las alertas por el modelo Isolation Forest se puede configurar alertas manuales a los sensores.

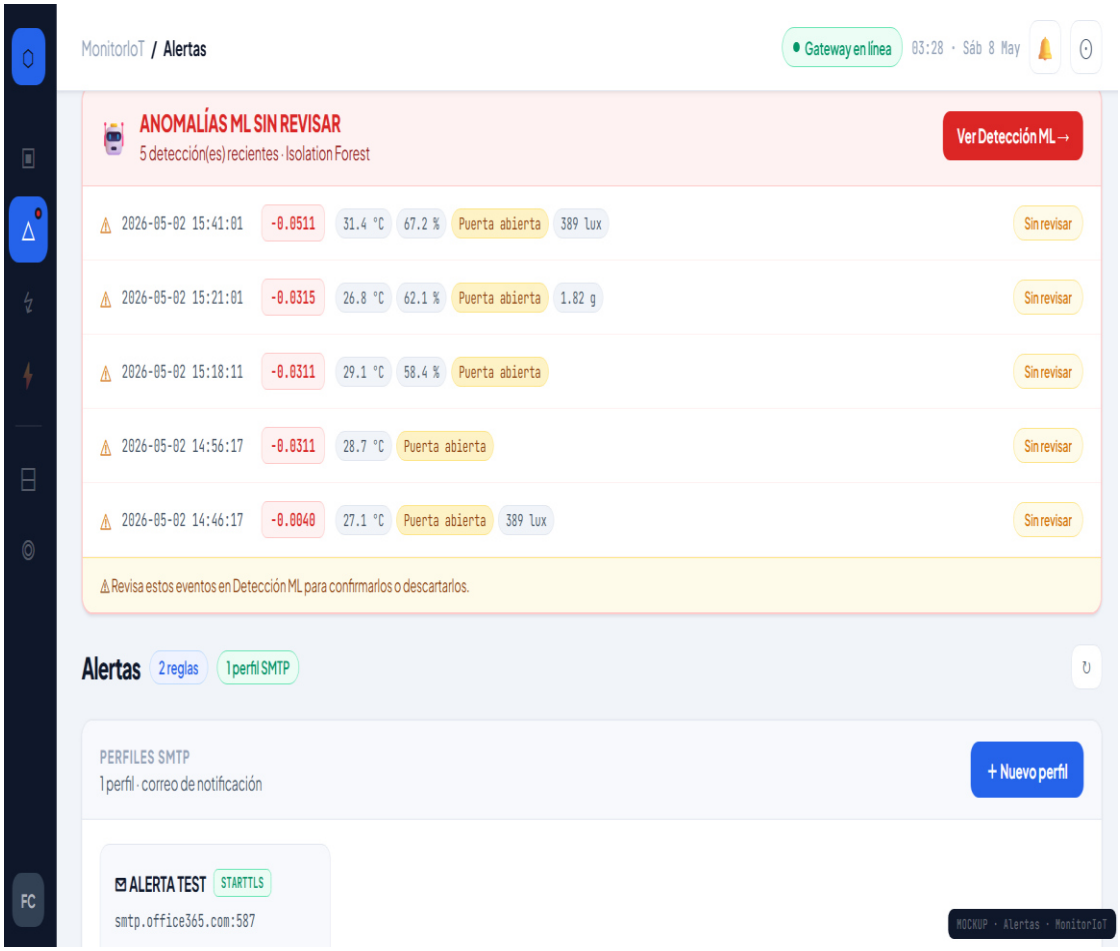


Figura 50 **Mockup de apartado de configuración de alertas.** Fuente. Elaboración propia.

En la Figura 51 se presenta el Mockup de la pestaña de Incidencias del frontend, en donde se puede visualizar todas las alertas que generaron los sensores únicamente activos en el sistema, estas alertas se generan ya fuese por el modelo de detección Isolation Forest como por alertas manuales creadas por el usuario operador/administrador, permitiendo en esta misma pestaña dar una alerta como revisada o marcada o dejarla como persistente.

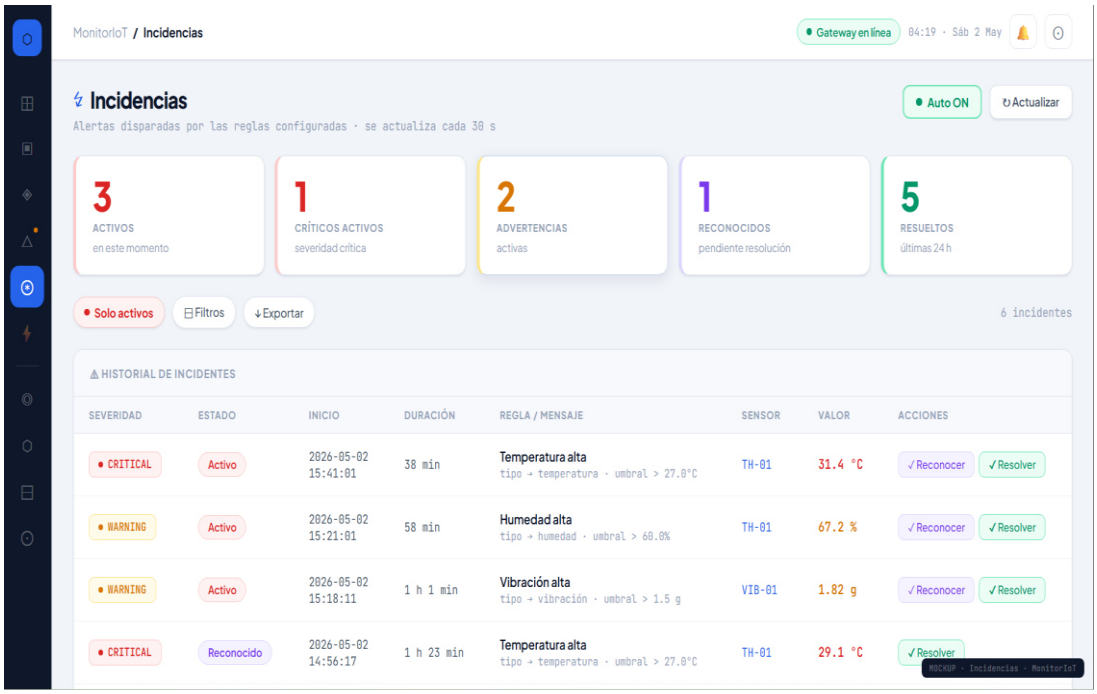


Figura 51 **Mockup pestaña de ultimas incidencias detectadas por el modelo ML Isolation Forest.** Fuente. Elaboración propia.

La Figura 52 presenta el apartado de Detección ML, apartado que muestra exclusivamente como está trabajando el modelo Isolation Forest.

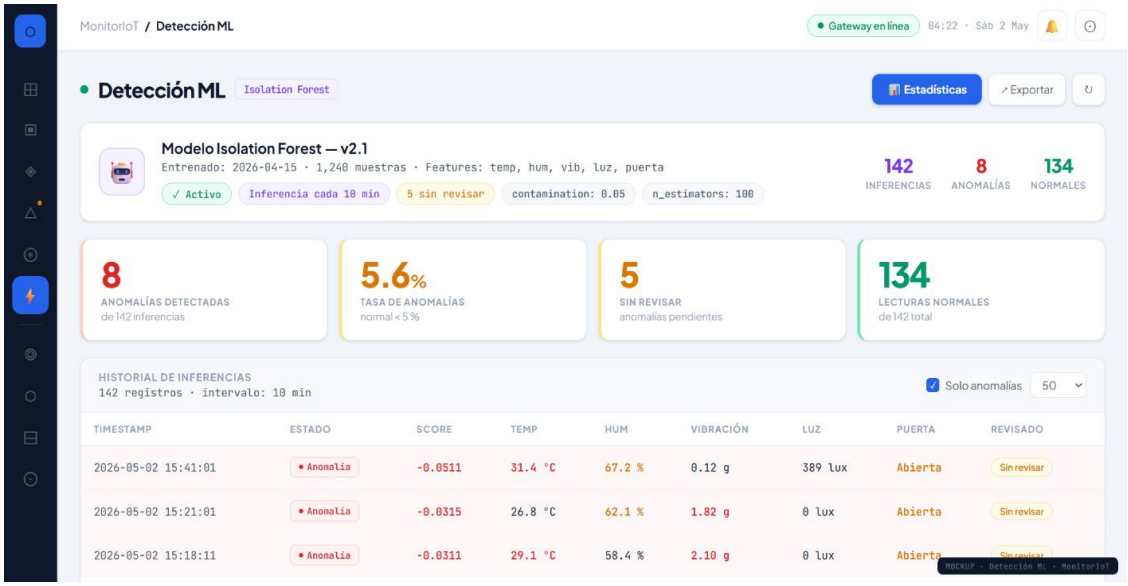


Figura 52 **Mockup Estadísticas del modelo de detección ML Isolation Forest.** Fuente. Elaboración propia.

En la Figura 53, está la pestaña de Gestión de Usuarios en la que se puede crear, editar y borrar usuarios a la plataforma asimismo también asignar roles para dichos usuarios.

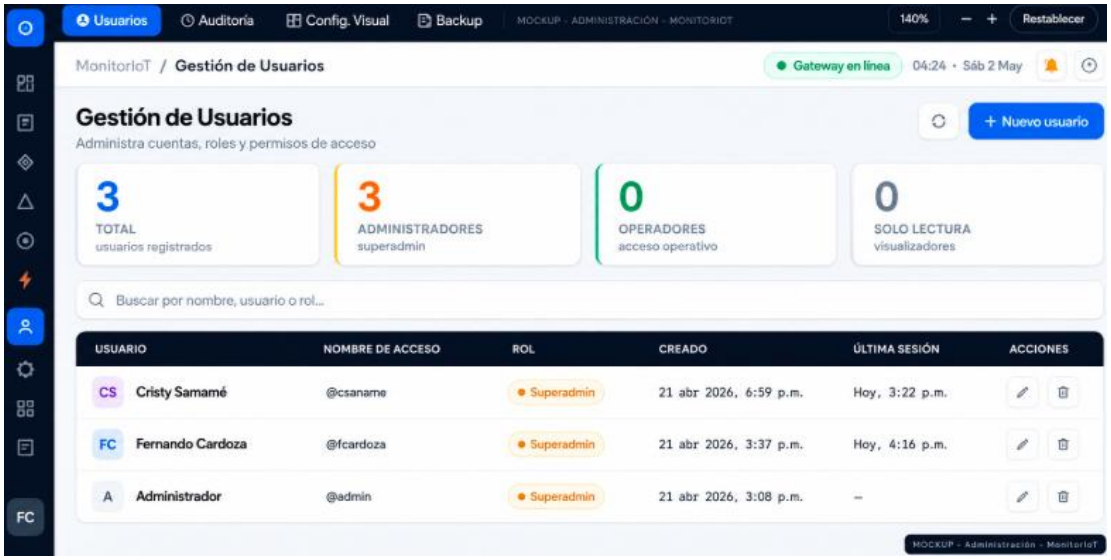


Figura 53 **Mockup Apartado de gestión de usuarios.** Fuente. Elaboración propia.

En la Figura 54, se muestra la pestaña Auditoría del Sistema, en donde todos los cambios y accesos a la plataforma se pueden evidenciar, accesos, creación de usuarios, dashboard y modificaciones.

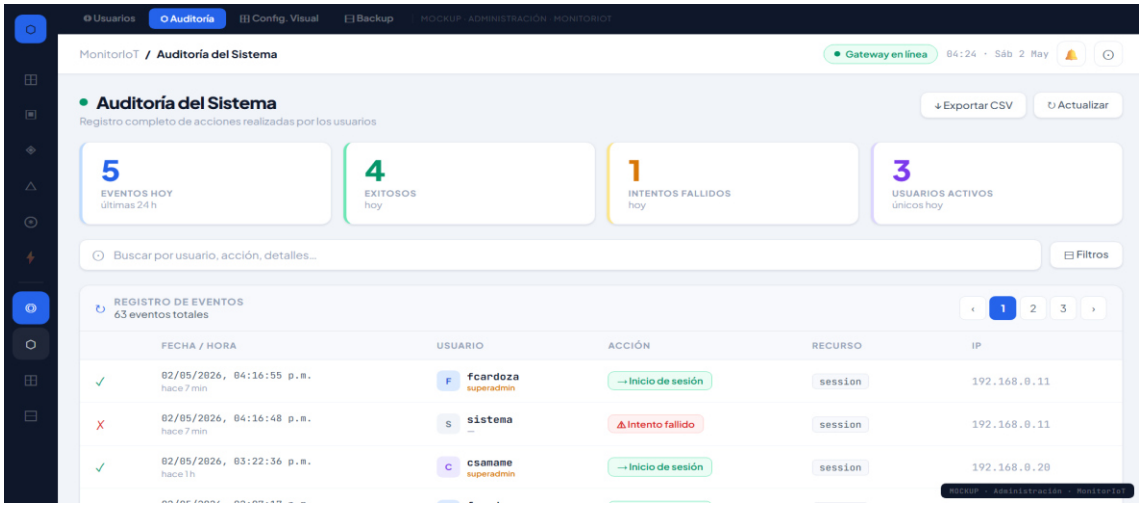


Figura 54 **Mockup apartado de auditoría del sistema.** Fuente. Elaboración propia.

En la Figura 55 se muestra la configuración de temas visuales, donde el usuario puede seleccionar entre distintos esquemas de color, como modo claro, oscuro o temas personalizados. Esta función permite adaptar la interfaz a las preferencias del operador o a la identidad visual de la organización.

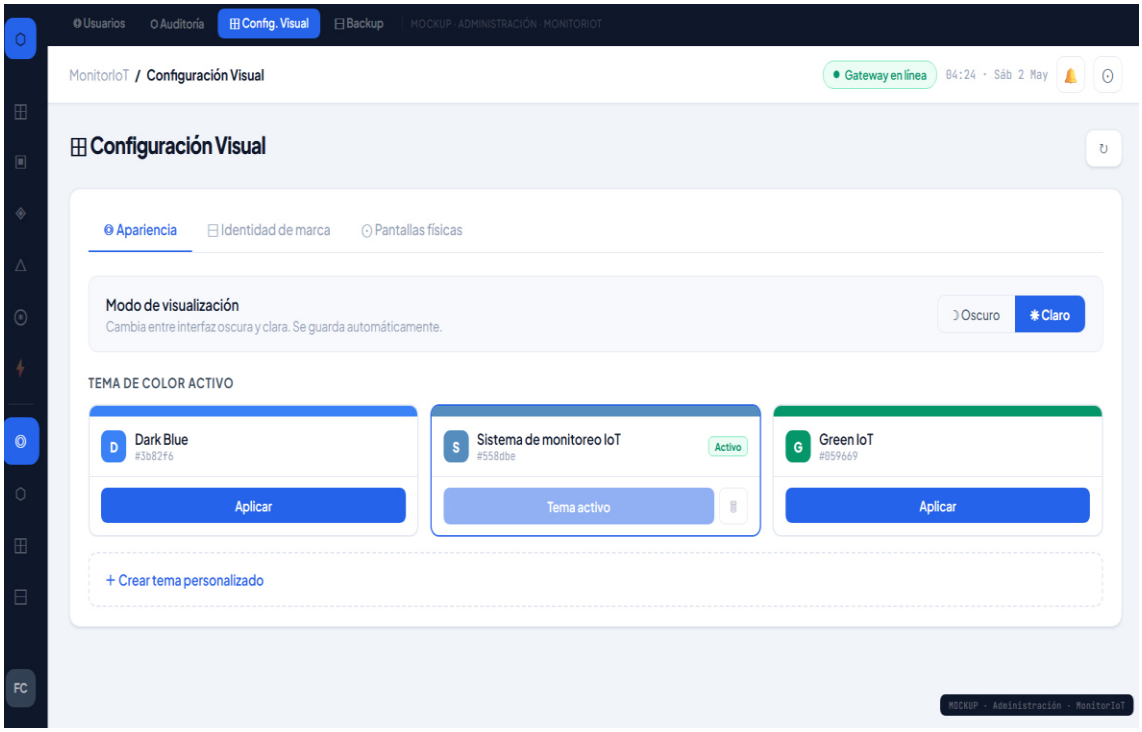


Figura 55 **Mockup Configuración visual para los temas de la página web.** Fuente. Elaboración propia.

La Figura 56 presenta el apartado de identidad de marca, orientado a la personalización de los elementos gráficos de la plataforma, como el nombre de la aplicación, el logo de inicio de sesión, el logo de la barra lateral y el favicon del sistema, agrupando en una sola sección los componentes visuales que definen la apariencia institucional del sistema. Con ello, la plataforma puede ajustarse visualmente al entorno institucional donde sea desplegada, permitiendo que distintas organizaciones adapten la interfaz a su propia identidad corporativa sin necesidad de modificar el código fuente de la aplicación. De esta manera, se favorece la reutilización del sistema en diferentes contextos, manteniendo la coherencia visual entre la plataforma y la imagen institucional de cada cliente.

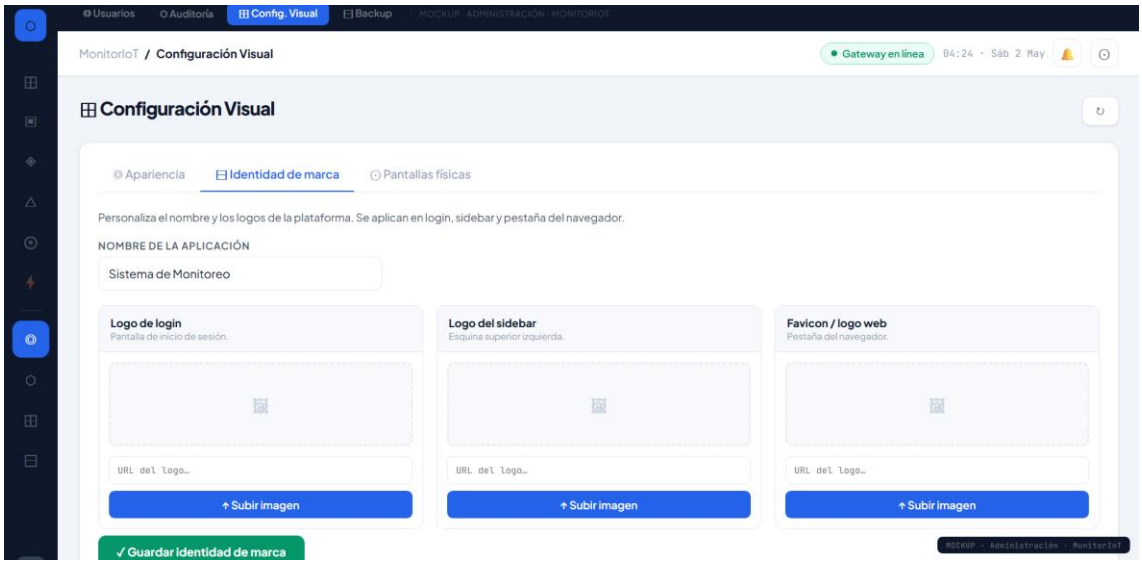


Figura 56 **Mockup de la configuración visual de identidad de marca.** Fuente. Elaboración propia.

La Figura 57 muestra la configuración de las pantallas físicas del equipo, desde donde se puede administrar el contenido mostrado en los displays integrados al prototipo. Este apartado permite definir parámetros visuales y funcionales de las pantallas, facilitando su gestión desde la plataforma web.

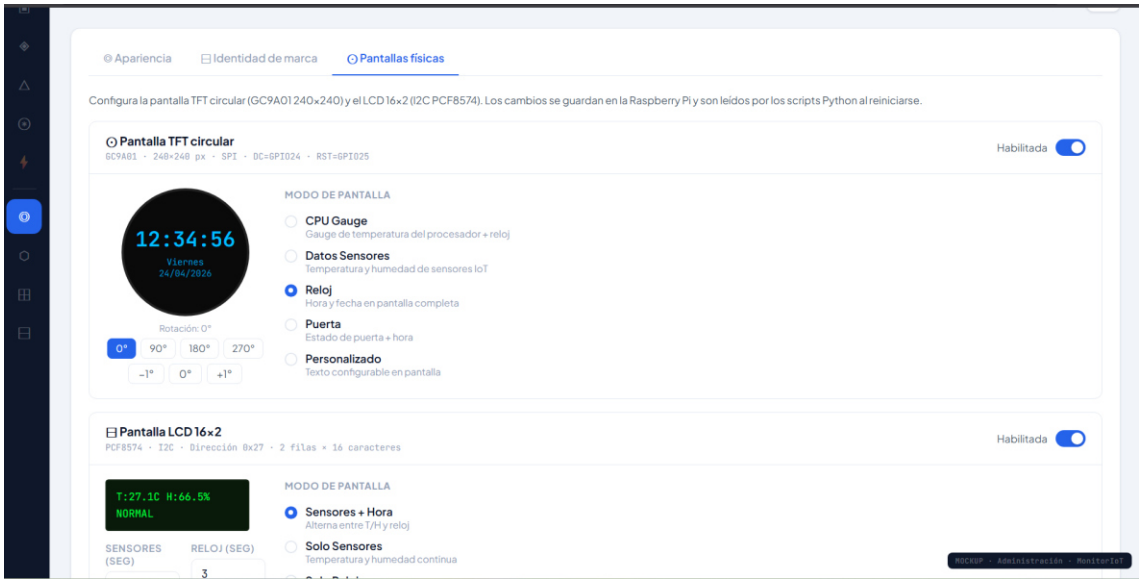


Figura 57 **Mockup apartado de configuración visual sobre las pantallas físicas del equipo.** Fuente. Elaboración propia.

En la Figura 58 esta presenta el módulo de Backup y Restauración del sistema, como parte de la función ante cualquier perdida o desastre se pueda recuperar toda la configuración de sistema.

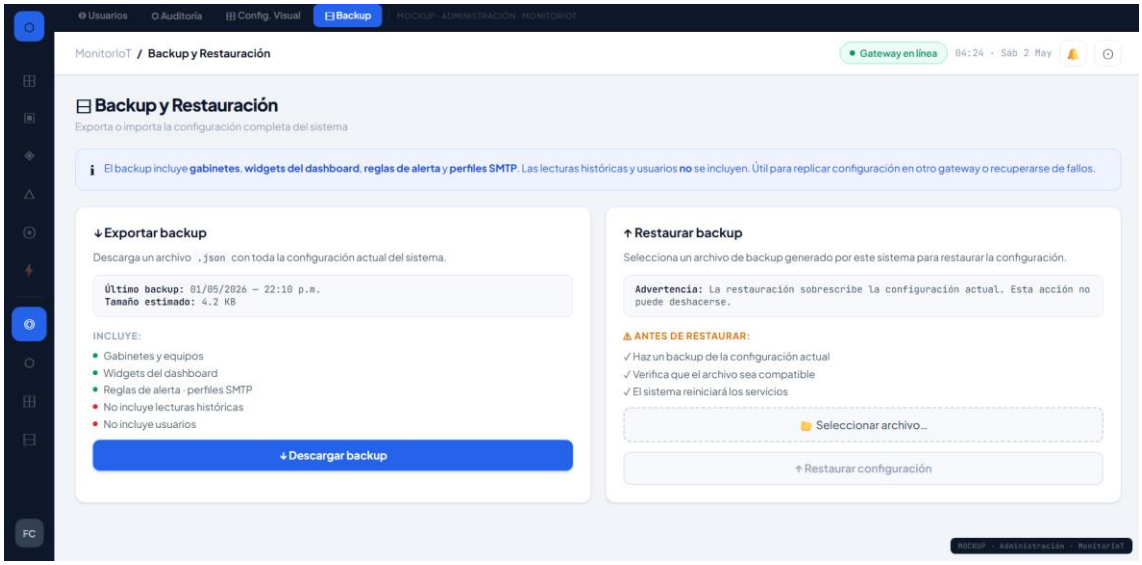


Figura 58 **Mockup de apartado de backup y restauración del sistema.** Fuente. Elaboración propia.

e.4) Implementación de la plataforma de software

La implementación siguió el orden establecido en el diseño. La base de datos *monitoring_iot* fue desplegada sobre el Gateway IoT con las tablas definidas, restricciones de unicidad en campos clave e índice sobre marca temporal en la tabla de lecturas para optimizar las consultas históricas. El backend Flask se desplegó con dos flujos concurrentes independientes, uno orientado al procesamiento de telemetría MQTT con evaluación inmediata de reglas de alerta, y otro destinado a la API REST con validación de token y control de acceso por rol. El frontend ReactJS fue implementado con todos los módulos definidos en el diseño, incluyendo la lógica de selección dinámica de endpoint local o remoto. Los tres componentes fueron registrados como servicios del sistema operativo con política de reinicio automático, satisfaciendo RNF-04. La Figura 59 muestra el frontend desplegado en producción,

donde se visualiza el tablero de monitoreo (dashboard) operativo con paneles de sensores, gráficas, valores actuales y registros de lectura.

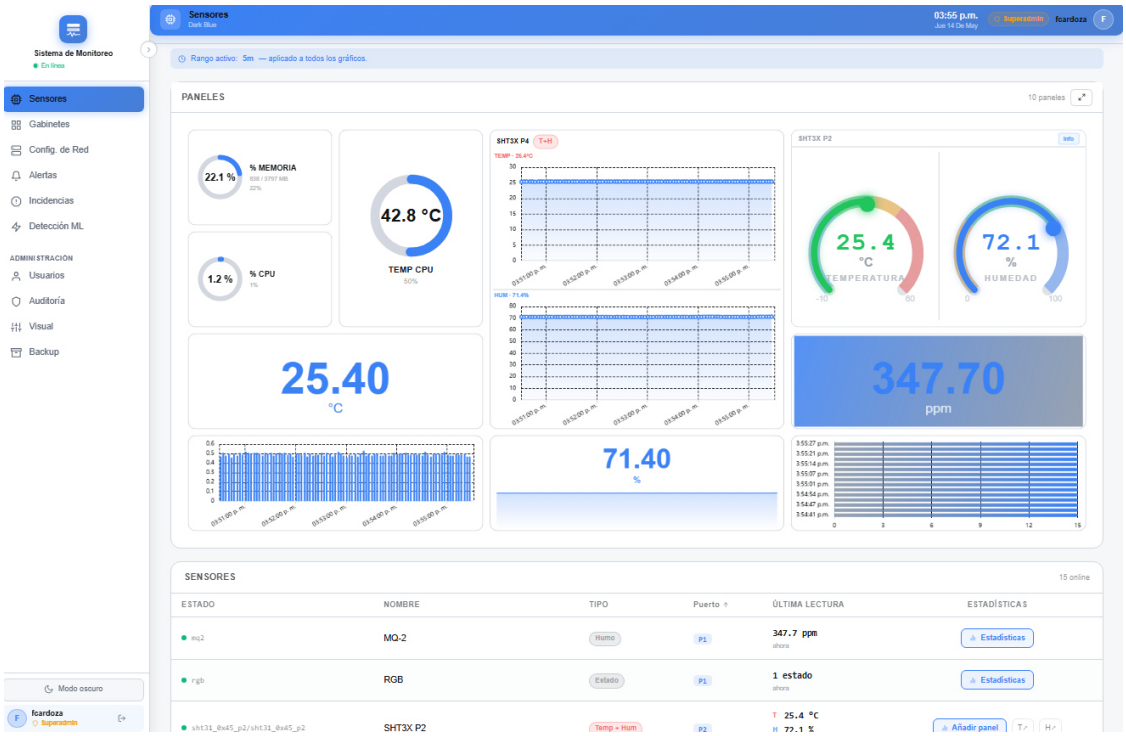


Figura 59 **Frontend desplegado en producción.** Fuente. Elaboración propia.

Por otro lado, la Figura 60 presenta el mockup elaborado durante la fase de diseño, el cual sirvió como referencia para la organización visual de los módulos, tarjetas de estado, gráficos y tabla de lecturas. Este mockup permitió establecer, de manera previa a la implementación, la distribución espacial de los distintos elementos de la interfaz, facilitando la toma de decisiones sobre jerarquía visual, agrupación de información y prioridad de los datos mostrados al usuario. A partir de este diseño preliminar, se definió la ubicación de las tarjetas de estado como elementos de rápida lectura, la disposición de los gráficos como recursos de análisis más detallado, y la tabla de lecturas como fuente de información granular para consultas específicas. De esta forma, el mockup no solo cumplió una función estética, sino que actuó como guía funcional para asegurar que la interfaz final resultara intuitiva, ordenada y coherente con las necesidades de monitoreo del usuario, reduciendo así ajustes posteriores durante la etapa de desarrollo del frontend.

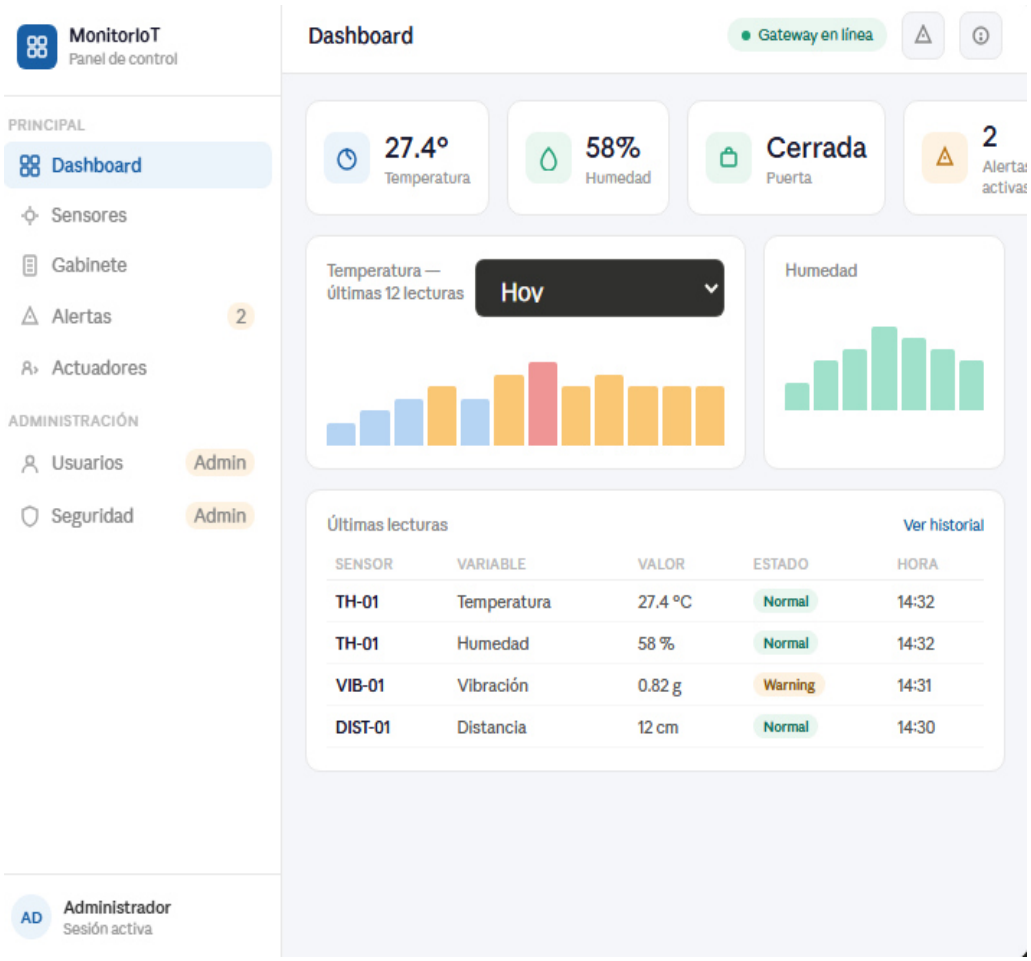


Figura 60 **Mockup presentado en el diseño previo a implementación.** Fuente. Elaboración propia.

f) Etapa 6: Infraestructura Híbrida On-Premise y en la Nube

Esta etapa extiende el alcance del sistema más allá de la red local del Gateway IoT, habilitando el acceso remoto al backend y la publicación del frontend en Internet mediante una arquitectura híbrida que combina infraestructura on-premise con servicios en la nube. La Figura 61 presenta la arquitectura de infraestructura híbrida del sistema, organizada en tres planos principales correspondientes al plano local, Cloudflare y Azure. En el plano local, el Gateway IoT mantiene la lógica operativa del sistema, la plataforma de software con Flask y MariaDB, y el acceso local continuo, incluso si la conexión a Internet falla. El plano Cloudflare actúa como capa de seguridad y acceso remoto, gestionando el DNS, el cifrado HTTPS, la protección perimetral y el túnel cifrado hacia el backend local. Finalmente, el plano Azure aloja

el frontend web mediante Azure Static Web Apps, permitiendo su publicación global, alta disponibilidad y despliegue automático. De esta manera, la arquitectura permite que el sistema opere localmente de forma autónoma, pero también pueda ser consultado de manera remota y segura desde Internet.

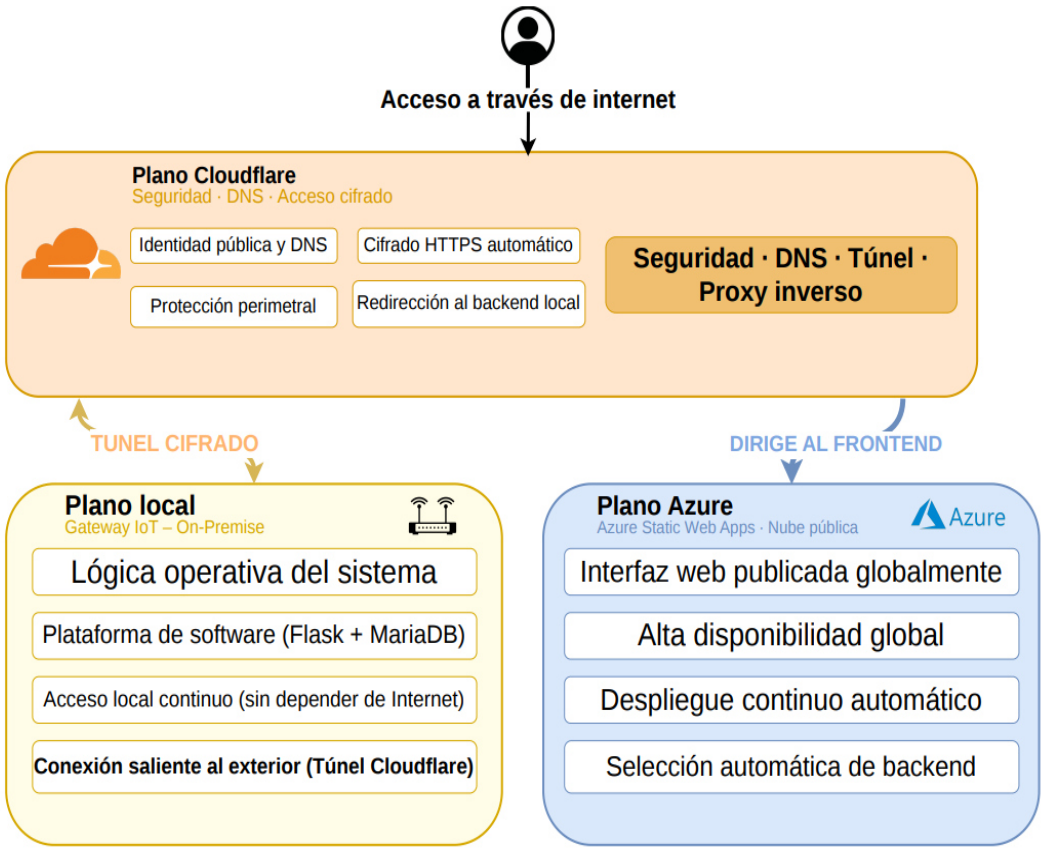


Figura 61 **Arquitectura general de la infraestructura híbrida: plano local, Cloudflare y Azure.** Fuente. Elaboración propia.

f.1) Requerimientos de la infraestructura híbrida

Los requerimientos se derivan de la necesidad de extender el acceso al sistema sin comprometer la seguridad perimetral del Gateway ni la continuidad operativa local. El requerimiento RF-01, junto con RNF-02, descarta cualquier enfoque basado en exposición directa de puertos entrantes, orientando el diseño hacia modelos de conexión saliente. La Tabla 32 presenta la especificación completa.

Tabla 32 **Requerimiento de la infraestructura híbrida**

ID	Requerimiento	Descripción
RF-01	Acceso remoto seguro al *backend*	El *backend* debe ser accesible desde Internet sin exponer puertos entrantes en la red del Gateway ni revelar su dirección IP.
RF-02	Publicación del *frontend* con alta disponibilidad	La interfaz web debe estar disponible mediante URL pública con certificado SSL válido, accesible desde cualquier ubicación geográfica.
RF-03	Selección automática de *backend* según entorno	El *frontend* debe determinar si opera en red local o remota y conectarse al *endpoint* correspondiente sin intervención del usuario.
RF-04	Gestión de dominio y subdominios	El sistema debe disponer de un dominio público con subdominios dedicados para *backend* y *frontend*, con resolución DNS administrada centralmente.
RF-05	Despliegue continuo del *frontend*	Toda actualización en el repositorio fuente debe producir automáticamente una nueva versión publicada, sin intervención manual.
RNF-01	Continuidad operativa local	El sistema debe mantener operación completa en red local independientemente del estado de los servicios en la nube.
RNF-02	Seguridad perimetral	Ningún servicio interno del Gateway IoT debe quedar accesible directamente desde Internet. El tráfico externo debe transitar por una capa intermedia de seguridad.
RNF-03	Cifrado de comunicaciones externas	Todas las comunicaciones entre el cliente y los servicios publicados deben realizarse bajo HTTPS con certificado SSL válido.
RNF-04	Escalabilidad del *frontend*	El servicio de publicación del *frontend* debe absorber incrementos de tráfico sin requerir intervención de configuración adicional.
RNF-05	Independencia entre capas	El *frontend* y el *backend* deben poder actualizarse y operar de forma independiente, sin requerir despliegue simultáneo.

Fuente. Elaboración propia

f.2) Selección de tecnologías

La selección se realizó evaluando alternativas bajo tres criterios: ausencia de exposición directa de la red local, certificados SSL gestionados automáticamente y viabilidad en nivel de servicio gratuito dada la naturaleza académica del proyecto. Cloudflare Tunnel resultó la opción más adecuada para el acceso remoto al backend, dado que el agente establece una conexión saliente persistente desde el Gateway IoT. Para el alojamiento del frontend, Azure Static Web Apps ofrece distribución global, SSL automático e integración nativa con GitHub para el flujo de procesamiento (pipeline) CI/CD. La Tabla 33 resume los componentes seleccionados con su justificación.

Tabla 33 **Componentes de la infraestructura híbrida**

Componente / Servicio	**Función en el sistema**	**Justificación de selección**
Cloudflare (DNS y proxy)	Gestión centralizada de dominio	Plataforma unificada que oculta la IP del Gateway, aplica cifrado SSL automático y ofrece protección perimetral sin costo operativo adicional.
Cloudflare Tunnel (cloudflared)	Acceso remoto al *backend* mediante conexión saliente cifrada.	Elimina la necesidad de abrir puertos entrantes. El túnel se establece desde el Gateway hacia Cloudflare, no a la inversa, cumpliendo RNF-02.
Azure Static Web Apps	Alojamiento del *frontend* React con distribución global.	Servicio administrado con SSL automático, integración nativa con GitHub y escalabilidad sin intervención, cubriendo RF-02, RF-05 y RNF-04.
Dominio tesis-monitoring.xyz	Identidad pública del sistema.	Punto de acceso estable e independiente de cualquier IP, con DNS administrados en Cloudflare para gestión centralizada de subdominios.

GitHub (repositorio *frontend*)	Control de versiones y disparador del pipeline de despliegue.	La integración con Azure Static Web Apps genera el flujo de despliegue continuo sin configuración adicional, cubriendo RF-05.

Fuente. Elaboración propia

f.3) Diseño de la infraestructura híbrida

El diseño organiza la infraestructura en tres planos con responsabilidades delimitadas, donde el plano local alberga el núcleo operativo con autonomía total; el plano de Cloudflare gestiona la identidad pública y actúa como intermediario del tráfico externo; y el plano de Azure aloja el frontend como aplicación estática distribuida globalmente. Esta separación permite que cada plano cumpla una función específica dentro del sistema, evitando la concentración de responsabilidades en un único punto y facilitando el mantenimiento independiente de cada componente sin afectar el funcionamiento de los demás.

La identidad pública del sistema se centraliza bajo un dominio propio con gestión DNS consolidada en Cloudflare, cuyos subdominios con proxy activo ocultan la IP real del Gateway y aplican cifrado SSL automático, reforzando así la seguridad perimetral del sistema frente a intentos de acceso directo o exposición innecesaria de la infraestructura local. Esta configuración permite además administrar de forma centralizada los distintos puntos de entrada del sistema, manteniendo una estructura ordenada y fácilmente escalable ante la incorporación de nuevos servicios. El detalle de la estructura DNS y subdominios se presenta en el Anexo I.

El acceso remoto adopta el modelo de túnel de conexión saliente, en el que es el Gateway quien inicia la conexión hacia Cloudflare, y no a la inversa, eliminando la necesidad de gestionar reglas de firewall entrantes y satisfaciendo RNF-02. Este enfoque reduce significativamente la superficie de ataque del sistema, ya que no se requiere exponer puertos hacia la red pública ni mantener configuraciones de acceso entrante en el Gateway local. La Figura 62 ilustra el flujo completo de una solicitud remota desde el usuario hasta el backend local, evidenciando cómo cada plano

interviene de forma coordinada para garantizar una comunicación segura y estable a lo largo de todo el recorrido

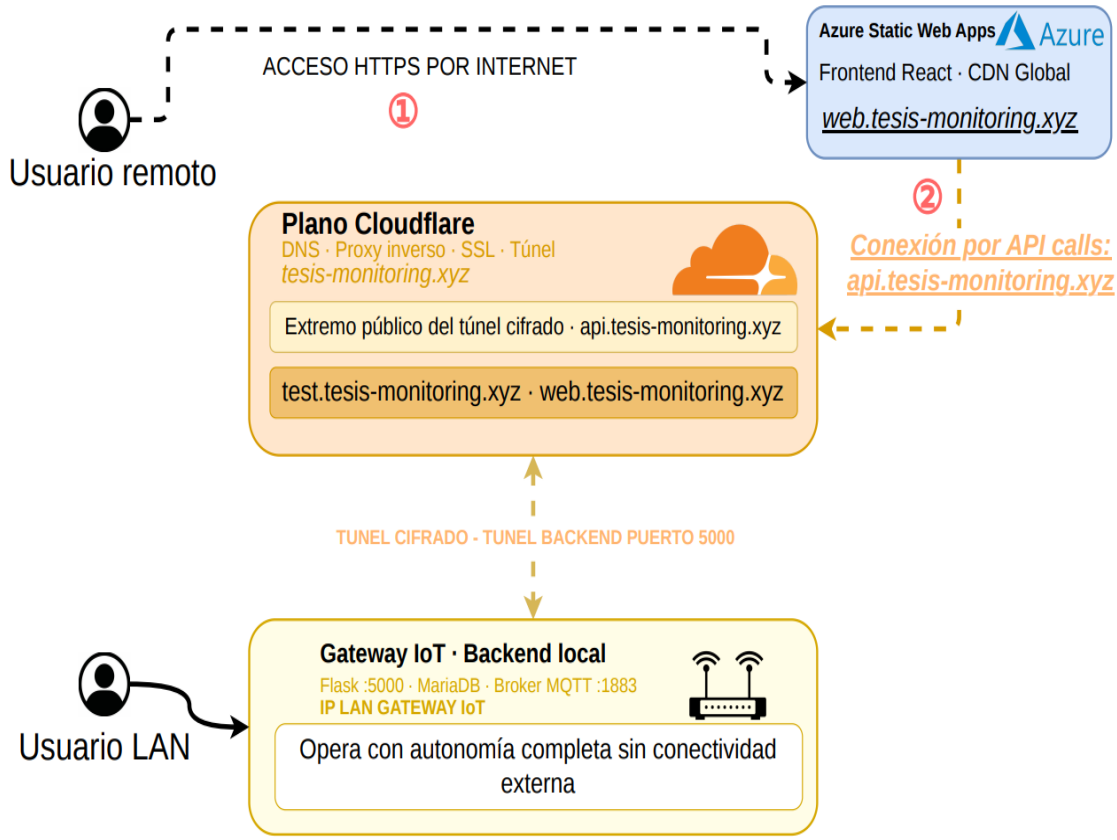


Figura 62 **Flujo de acceso remoto al backend mediante Cloudflare Tunnel.** Fuente. Elaboración propia.

El frontend opera en dos entornos posibles: red local e Internet, cuya dualidad se resuelve mediante variables de entorno inyectadas en tiempo de compilación. En tiempo de ejecución el frontend evalúa el origen de la sesión activa y selecciona el endpoint correspondiente de forma transparente, satisfaciendo RF-03. El despliegue se automatiza mediante CI/CD, de modo que, ante cada actualización en el repositorio, Azure Static Web Apps compila la aplicación, inyecta las variables de entorno y publica los artefactos automáticamente, satisfaciendo RF-05. La Figura 63 presenta el flujo completo de despliegue y la lógica de selección de endpoint en tiempo de ejecución.

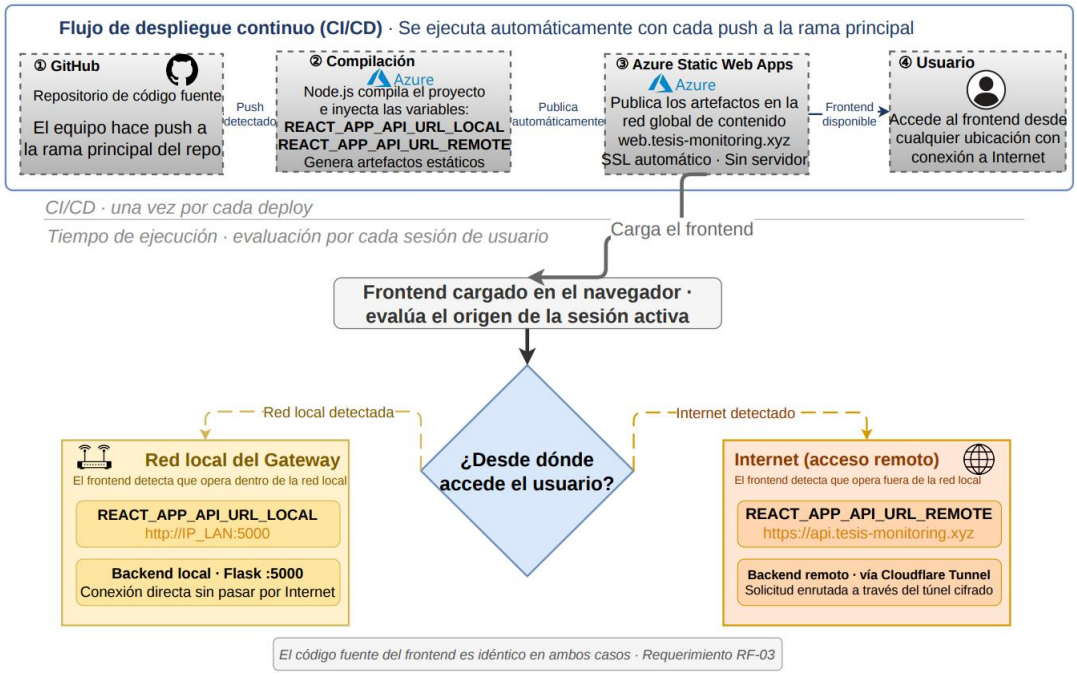


Figura 63 **Flujo de despliegue continuo del frontend: GitHub - Azure Static Web Apps.** Fuente. Elaboración propia.

La arquitectura fue evaluada frente a cuatro escenarios de conectividad para verificar que la dependencia de servicios externos se limite exclusivamente al acceso remoto, satisfaciendo RNF-01. La Tabla 34 presenta los resultados.

Tabla 34 **Escenarios de operación arquitectura híbrida**

Escenario	**Internet**	**Operación local**	**Acceso remoto**	**Componentes afectados**
Operación aislada	No disponible	Completa	No disponible	Cloudflare Tunnel - Azure Static Web Apps
Operación híbrida	Disponible	Completa	Completo	Ninguno
Fallo parcial de nube	Intermitente	Completa	Degradado	Acceso externo al *backend* y/o al *frontend*

Fallo de red local	Disponible	No disponible	No disponible	Todo el sistema (Gateway sin operación)

Fuente. Elaboración propia

f.4) Implementación de la infraestructura híbrida

La implementación siguió el orden dictado por las dependencias empezando por la habilitación del túnel de acceso remoto en el Gateway IoT, continuando con la configuración DNS como base de la identidad pública y finalizando con el despliegue del frontend con los endpoints ya definidos en Github para su publicación en Azure Static Web App. En la Figura 64 se muestra el servicio cloudflared ejecutándose en el Gateway IoT mediante systemd, con estado activo y habilitado para iniciar automáticamente con el sistema operativo. Esta configuración permite mantener la persistencia del túnel ante reinicios del equipo, satisfaciendo RNF-02

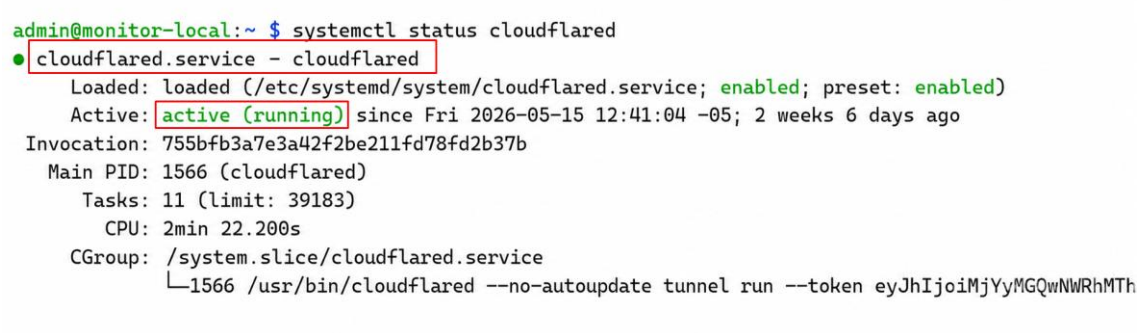


Figura 64 **Estado activo del servicio túnel Cloudflared.** Fuente. Elaboración propia.

En la Figura 65 se muestra la configuración DNS del dominio en Cloudflare, donde se observan los registros asociados al sistema y el estado del proxy aplicado a los subdominios correspondientes. Esta configuración permite que el acceso público sea gestionado mediante Cloudflare, incorporando proxy y SSL automático sin intervención manual en la administración de certificados, satisfaciendo RF-04 y RNF-03

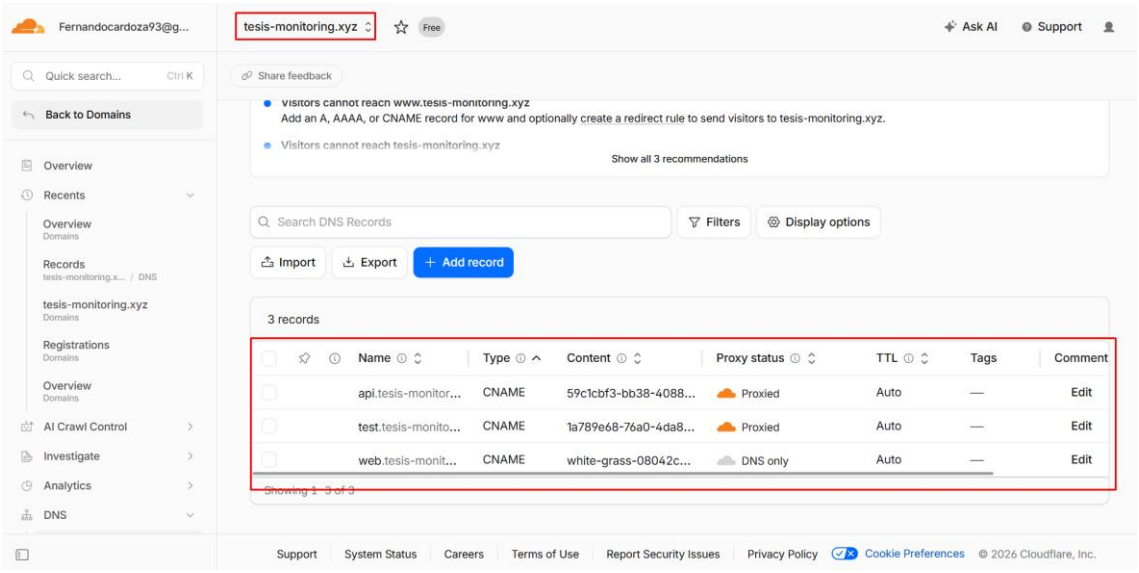


Figura 65 **Configuración DNS del dominio del sistema en Cloudflare.** Fuente. Elaboración propia.

En la Figura 66 se muestra el repositorio del proyecto en GitHub, donde se observa la estructura principal del sistema, incluyendo las carpetas correspondientes al backend, frontend, módulo de aprendizaje automático y flujo de trabajo de GitHub Actions. Esta organización permitió separar los componentes de software y facilitar su despliegue automatizado desde la rama principal.

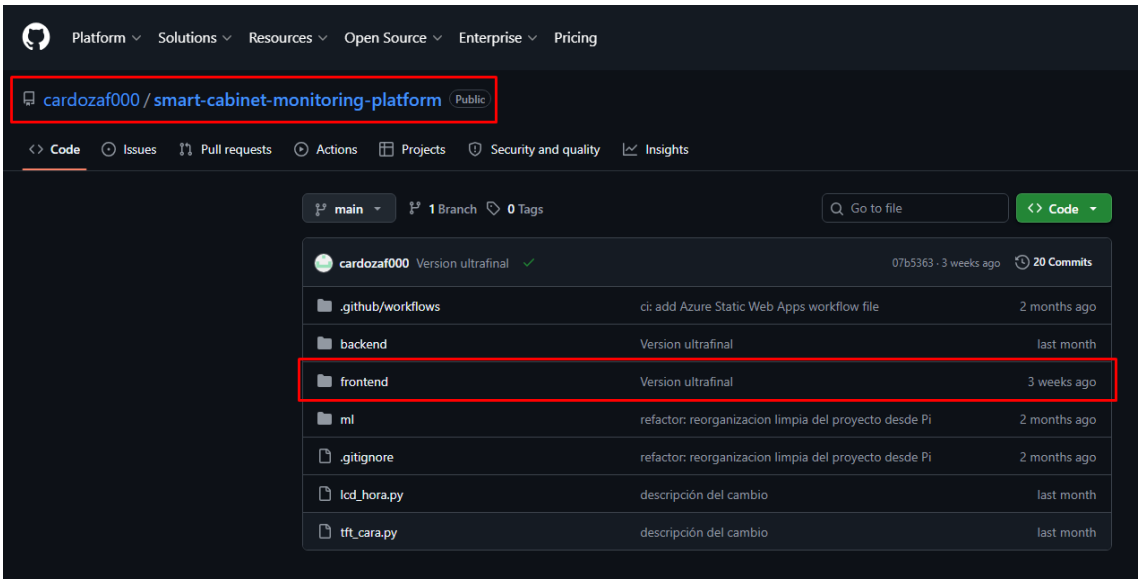


Figura 66 **Repositorio del sistema en GitHub.** Fuente. Elaboración propia.

El repositorio del frontend fue vinculado a Azure Static Web Apps con las variables de entorno configuradas directamente en la plataforma. En la Figura 67 se visualiza el recurso desplegado en Azure, donde se identifica la URL pública de la aplicación web estática y su vinculación con el flujo de implementación desde GitHub. De esta manera, cada commit en la rama principal desencadena automáticamente la compilación y publicación del frontend, satisfaciendo RF-05. El detalle de configuración del túnel, variables de entorno y estructura del flujo de procesamiento (pipeline) se presenta en el Anexo I.

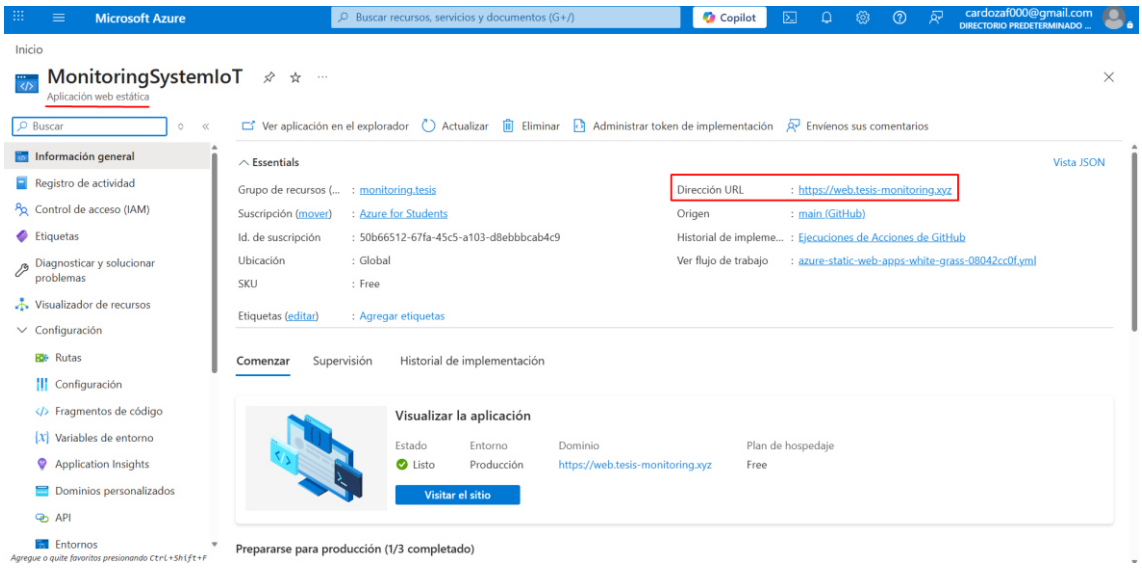


Figura 67 **Despliegue del frontend en Azure Static Web Apps.** Fuente. Elaboración propia.

IV.5 Análisis estadístico

IV.5.1 Preparación de datos

Las lecturas se depuraron descartando los valores fuera del rango físico definido para cada sensor; estos casos se registraron como eventos de error en el log del sistema. Para el modelo de detección de anomalías, las lecturas se agruparon en intervalos de 10 s mediante el promedio, se eliminaron las filas con menos de seis

canales presentes y los valores faltantes se imputaron con la mediana de cada columna. Finalmente, el vector de características se normalizó mediante estandarización (puntuación z) antes del entrenamiento y la inferencia.

IV.5.2 Estadística descriptiva

Se obtuvieron la media, el valor mínimo, el valor máximo y la desviación estándar de los indicadores de tiempo y latencia (publicación MQTT, latencia del broker, sincronización con la base de datos, actualización del tablero y notificación de alerta). Para los indicadores de confiabilidad se calcularon proporciones expresadas como porcentaje (tasa de éxito de los ciclos de adquisición y tasa de entrega de mensajes), y para la estabilidad eléctrica se determinó el error relativo de las tensiones de alimentación respecto de sus valores nominales, verificando su permanencia dentro del criterio de ±4 %.

IV.5.3 Evaluación del modelo de detección de anomalías

Sobre un conjunto de prueba etiquetado de 1 200 muestras (600 normales y 600 anomalías sintéticas), se calcularon la precisión, el recall, el F1-score y la exactitud a partir de la matriz de confusión, junto con el tiempo de inferencia por muestra.

CAPÍTULO V:
RESULTADOS

V.1 Presentación y análisis de resultados

Este capítulo presenta los resultados obtenidos durante la validación de las seis etapas de desarrollo e implementación del sistema (sección IV.4.4), cada una asociada a su hipótesis específica. Los resultados obtenidos permiten contrastar cuantitativamente el comportamiento real del sistema frente a dichos criterios y determinar el cumplimiento de las hipótesis específicas planteadas. La Tabla 35 siguiente consolida los indicadores de desempeño más relevantes junto con el criterio de aceptación de la hipótesis correspondiente; cada valor corresponde al promedio obtenido sobre el número de mediciones indicado entre paréntesis, no a una medición aislada.

Tabla 35. **Resumen de los indicadores de desempeño obtenidos por etapa.**

Etapa	Indicador	Valor promedio obtenido	Criterio de aceptación
Etapa 1 - Hardware (UAC)	Tasa de éxito de lectura I²C/GPIO (n = 1 200 ciclos/canal)	100 por ciento	≥ tasa de éxito esperada.
	Tensión de alimentación respecto al valor nominal	Dentro de ±4 por ciento	≤ ±4 por ciento.

Etapa 2 - Firmware	Latencia media de publicación MQTT (n = 30 ciclos)	409 milisegundos	< 500 milisegundos.
	Reconexión ante pérdida de WiFi	2,2 segundos	< 5 segundos
	Reconexión ante pérdida de MQTT	0,3 segundos	< 5 segundos
Etapa 3 - Modelo Isolation Forest	Recall (n = 1 200 observaciones)	99,33 por ciento	> 95 por ciento
	F1-score (n = 1 200 observaciones)	97,47 por ciento	> 90 por ciento
	Tiempo de inferencia por muestra	0,057 milisegundos	< 5 segundos
Etapa 4 - Gateway IoT	Latencia del broker MQTT (n = 30 muestras)	12 a 14 milisegundos	< 500 milisegundos
	Latencia de sincronización MQTT–MariaDB (n = 20)	13,3 milisegundos	< 500 milisegundos
Etapa 5 - Plataforma de software	Actualización del panel de monitoreo (n = 10)	209,6 milisegundos	< 5 segundos

	Notificación de alerta (n = 10)	135,9 milisegundos	< 60 segundos
Etapa 6 - Infraestructura híbrida	Latencia de acceso remoto (túnel Cloudflare)	640 milisegundos	Valor referencial

Fuente. Elaboración propia, a partir de las pruebas de funcionamiento técnico del Capítulo IV, desarrolladas en detalle en las secciones V.1.1 a V.1.6.

V.1.1 Hardware de la Unidad de Adquisición y Control (UAC) basado en ESP32

Los resultados de la Etapa 1 corresponden al primer objetivo específico, orientado al diseño e implementación del hardware de la UAC basada en ESP32. Como resultado principal, se obtuvo un módulo electrónico integrado en un chasis de 1RU, compuesto por el microcontrolador ESP32-WROOM-32, el multiplexor I²C TCA9548A, el conjunto de sensores ambientales y estructurales definidos en los requisitos, y un sistema de alimentación multitensión. El diseño final del PCB, descrito en el Capítulo IV, incorporó la reubicación térmica de la fuente AC/DC, la reorganización de los conectores RJ45 y la redistribución de los bloques funcionales.

La Figura 68 muestra el PCB ensamblado, en el que se identifican seis bloques principales: la fuente de alimentación de 12 V, los conversores DC-DC de 5 V y 3,3 V, el módulo ESP32, el multiplexor TCA9548A, la etapa de control PWM mediante MOSFET y los conectores RJ45 destinados a los sensores distribuidos. El multiplexor I²C permitió administrar varios sensores con direcciones coincidentes mediante canales independientes, ampliando la capacidad de adquisición del sistema. Por su parte, los conectores RJ45 facilitaron la instalación, identificación y sustitución de los sensores distribuidos dentro del gabinete. Estos resultados permiten iniciar la validación de la hipótesis específica 1, mediante la medición de la tasa de éxito de lectura por bus I²C/GPIO y la estabilidad de las tensiones de alimentación del PCB, presentada a continuación.

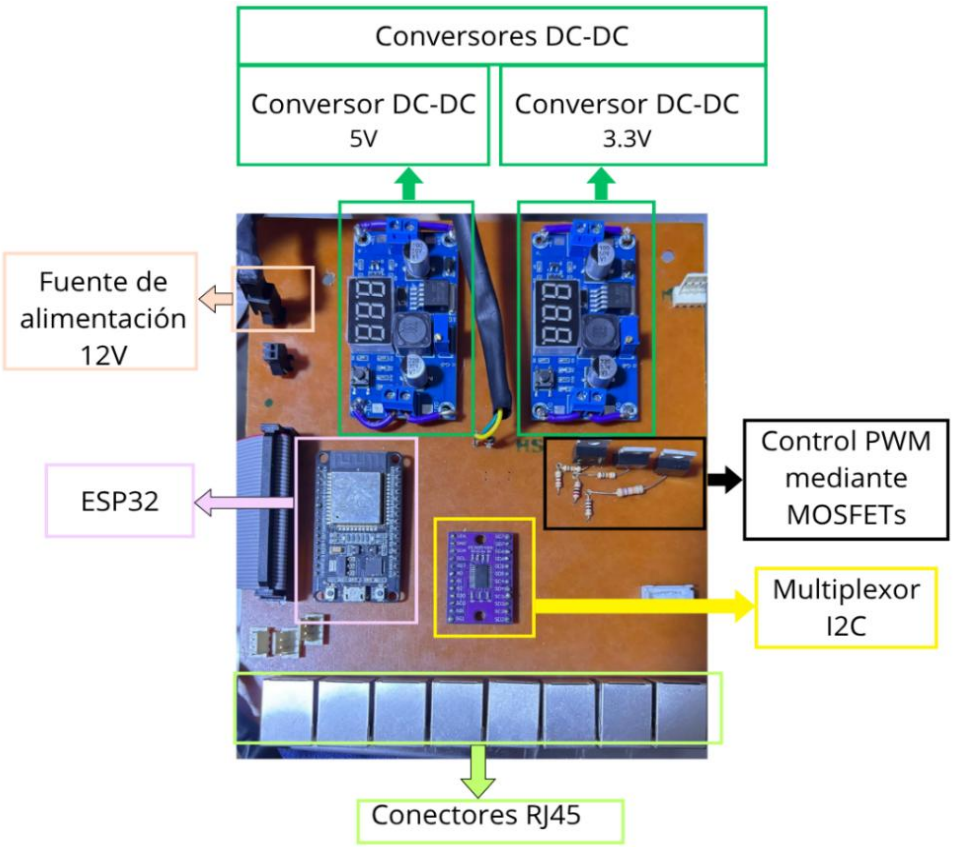


Figura 68 **PCB ensamblado para el hardware de la UAC (Iteración final).** Fuente. Elaboración propia.

La Tabla 36 resume los resultados obtenidos durante la prueba de estabilidad multisensorial realizada durante 60 minutos. Para cada sensor se presenta la tasa de éxito de lectura y lecturas exitosas totales, los valores máximos y mínimos registrados, la desviación estándar y el promedio de las mediciones, permitiendo evaluar tanto la continuidad de adquisición de datos como la variación de las lecturas durante el periodo de prueba.

Tabla 36 **Resultados de la prueba de estabilidad multisensorial**

Sensor	Lecturas exitosas / tasa de éxito de lecturas	Máximo valor medido	Mínimo valor medido	Desviación estándar	Promedio de lecturas
SHT31 #1	1200 / 100%	T: 23,8 °C / HR: 49,9 %	T: 23,2 °C / HR: 48,7 %	T: 0,17 / HR: 0,35	T: 23,5 °C / HR: 49,3 %

BH1750	1200 / 100%	153,4 lux	147,5 lux	1,70	150,45 lux
MPU6050	1200 / 100%	Mag: 0,11 m/s²	Mag: 0,05 m/s²	Mag: 0,017	normal: true (sin vibraciones)
SHT31 #2	1200 / 100%	T: 24,2 °C / HR: 50,8 %	T: 23,6 °C / HR: 49,6 %	T: 0,17 / HR: 0,35	T: 23,9 °C / HR: 50,2 %
SHT31 #3	1200 / 100%	T: 23,6 °C / HR: 50,3 %	T: 23,0 °C / HR: 49,1 %	T: 0,17 / HR: 0,35	T: 23,3 °C / HR: 49,7 %
VL53L0X	1200 / 100%	155 mm	149 mm	1,73	Distancia: 152 mm
MQ-2	1200 / 100%	102,1 u.a.	94,1 u.a.	2,31	98,1 u.a.
Reed Switch	1200 / 100%	0 (Cerrado)	0 (Cerrado)	0	0 (Cerrado durante la prueba)

Fuente. Elaboración propia

El conteo se registró mediante un contador incremental en el firmware de prueba: cada ciclo intenta una transacción I²C (vía multiplexor TCA9548A) o una lectura GPIO por sensor; se cuenta como válida si hay ACK y el valor está en rango físico, y como fallo si hay NACK o valor fuera de rango.

La Tabla 37 presenta una muestra de 10 ciclos consecutivos, seleccionados de forma aleatoria dentro del total de 1 200 ciclos registrados durante la prueba de 60 minutos, extraída directamente del monitor serie del sistema. Dado el volumen total de la prueba, se muestra este segmento como evidencia del formato de salida y del comportamiento de la prueba, cuyo resultado agregado fue del 100 % de lecturas válidas en todos los sensores.

Tabla 37. ***Muestra aleatoria de registros por ciclo (10 ciclos consecutivos)***

Ciclo	Lectura	Sensores leídos	Estado
1183	T1:23.7°C/H1:48.7% T2:24.1°C/H2:50.7% T3:23.5°C/H3:49.3% Lux:150.1 Vib:0.05 m/s²	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK

	Dist:153 mm Gas:97.0 u.a. Reed: CERRADO	Reed	Reed OK
1184	T1:23.7°C/H1:48.7% T2:24.1°C/H2:50.7% T3:23.5°C/H3:49.3% Lux:150.1 Vib:0.05 m/s² Dist:153 mm Gas:97.0 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1185	T1:23.7°C/H1:48.7% T2:24.1°C/H2:50.7% T3:23.5°C/H3:49.3% Lux:150.1 Vib:0.05 m/s² Dist:153 mm Gas:97.0 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1186	T1:23.6°C/H1:48.7% T2:24.1°C/H2:50.7% T3:23.5°C/H3:49.3% Lux:150.2 Vib:0.05 m/s² Dist:153 mm Gas:96.9 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1187	T1:23.6°C/H1:48.7% T2:24.2°C/H2:50.7% T3:23.5°C/H3:49.2% Lux:150.2 Vib:0.05 m/s² Dist:153 mm Gas:96.9 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1188	T1:23.6°C/H1:48.7% T2:24.2°C/H2:50.7% T3:23.5°C/H3:49.2% Lux:150.2 Vib:0.05 m/s² Dist:153 mm Gas:96.9 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1189	T1:23.6°C/H1:48.8%	SHT31 (x3)	SHT31 (x3) OK

	T2:24.2°C/H2:50.7% T3:23.6°C/H3:49.2% Lux:150.3 Vib:0.06 m/s² Dist:152 mm Gas:96.8 u.a. Reed: CERRADO	BH1750 MPU6050 VL53L0X MQ-2 Reed	BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1190	T1:23.5°C/H1:48.8% T2:24.2°C/H2:50.7% T3:23.6°C/H3:49.2% Lux:150.3 Vib:0.06 m/s² Dist:152 mm Gas:96.8 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1191	T1:23.6°C/H1:48.8% T2:24.2°C/H2:50.7% T3:23.6°C/H3:49.3% Lux:150.3 Vib:0.07 m/s² Dist:151 mm Gas:96.8 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK
1192	T1:23.6°C/H1:48.9% T2:24.3°C/H2:50.6% T3:23.7°C/H3:49.3% Lux:150.4 Vib:0.07 m/s² Dist:151 mm Gas:96.7 u.a. Reed: CERRADO	SHT31 (x3) BH1750 MPU6050 VL53L0X MQ-2 Reed	SHT31 (x3) OK BH1750 OK MPU6050 OK VL53L0X OK MQ-2 OK Reed OK

Fuente. Elaboración propia

La Figura 69 muestra la captura original del monitor serie correspondiente a dos de estos ciclos, como respaldo de la fuente de los datos presentados. Mientras que la Figura 70 presenta la cantidad de lecturas válidas obtenidas por cada sensor durante la prueba de estabilidad de 60 minutos, considerando 1 200 ciclos de medición por canal, con el propósito de verificar la continuidad de adquisición de datos y la estabilidad operativa de los sensores durante el periodo de prueba.

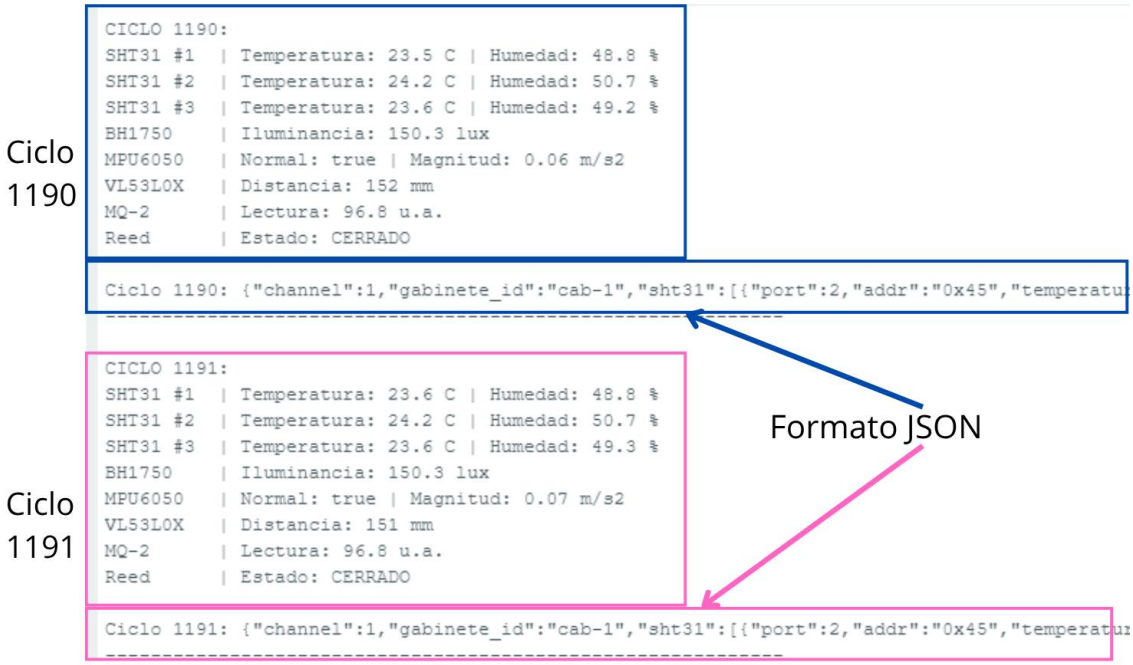


Figura 69 **Captura del monitor serie de muestra de dos ciclos consecutivos.** Fuente. Elaboración propia.

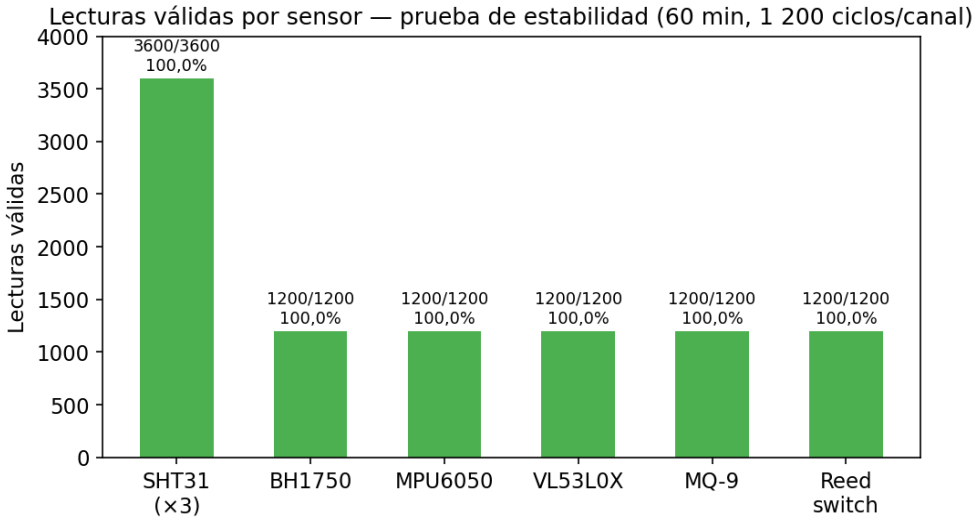
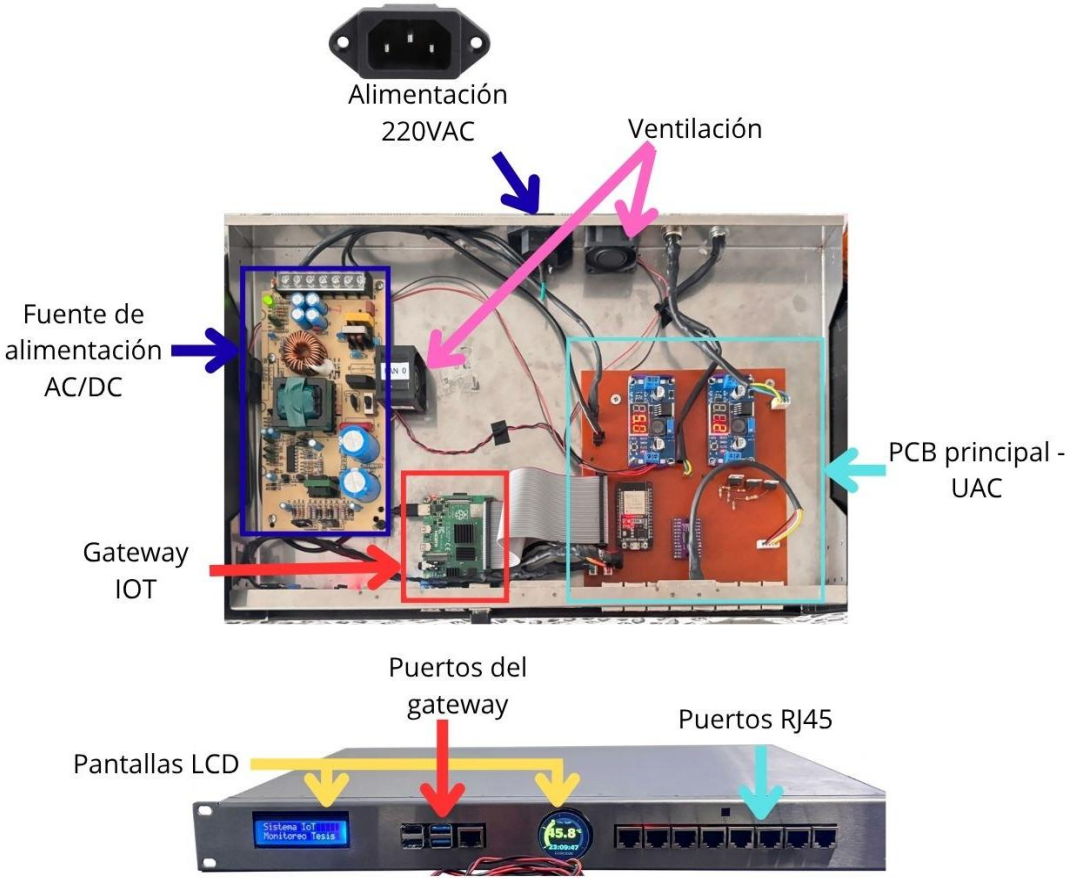


Figura 70 **Gráfico de barras expresando los resultados de las pruebas.** Fuente. Elaboración propia, a partir de los resultados de la Tabla 36.

Una vez verificado el correcto funcionamiento del PCB, el módulo fue integrado en un chasis metálico compatible con el estándar EIA-310 para racks de 19

pulgadas, en formato 1RU, con una altura de 44,45 mm. El chasis aloja el PCB principal de la UAC, la fuente de alimentación AC/DC y el Gateway IoT correspondiente a la Etapa 4.

En la Figura 71 se presenta el conjunto integrado, mostrando la distribución interna de los componentes y la disposición de los elementos del panel frontal. La organización interna permitió separar la etapa de alimentación, el procesamiento de la UAC y el Gateway IoT, favoreciendo una instalación ordenada y facilitando las labores de mantenimiento. Asimismo, se incorporó ventilación activa para contribuir a la disipación del calor generado por la fuente y los componentes electrónicos durante su operación continua. En el panel frontal se dispusieron las pantallas LCD, los puertos de comunicación del Gateway IoT y los conectores RJ45 destinados a los sensores distribuidos. Finalmente, el diseño modular permitió conectar externamente la tira RGB y los diferentes sensores sin necesidad de desmontar el equipo instalado en el rack.



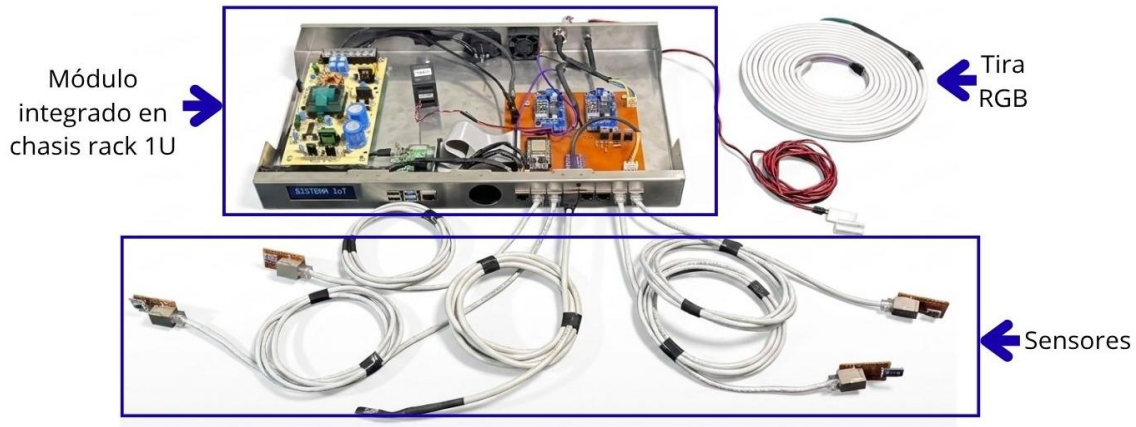


Figura 71 **Hardware final integrado en chasis 1RU: componentes internos (superior), vista frontal (centro) y vista general (inferior).** Fuente. Elaboración propia.

Con el hardware integrado y cableado, se realizó la verificación del sistema de adquisición multisensorial enfocada en comprobar la correcta conexión física de los sensores, la estabilidad del bus I²C y la operación de los canales habilitados mediante el multiplexor. Los sensores conectados por RJ45 fueron evaluados sobre el bus I²C a 100 kHz, mientras que los sensores de interfaz GPIO fueron verificados en paralelo. La Figura 72 presenta el reporte obtenido al finalizar la prueba de funcionamiento; la prueba se ejecutó durante 60 minutos, se acumularon 1 200 ciclos de lectura por sensor.

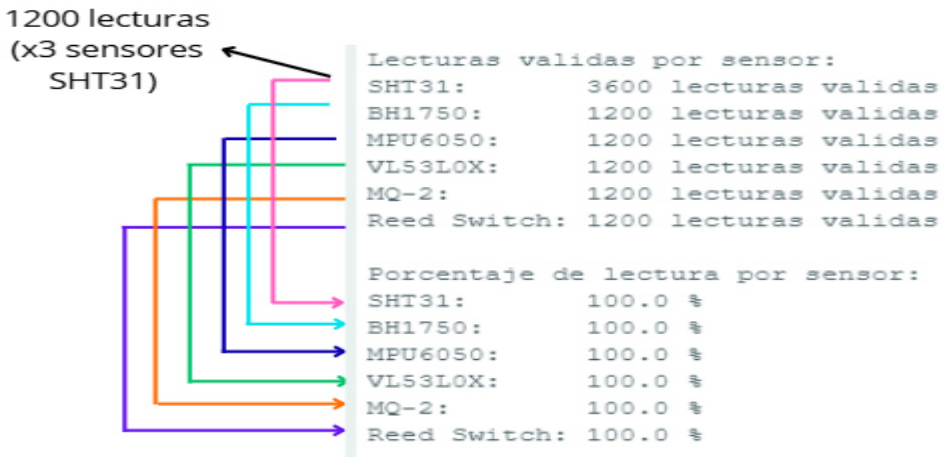


Figura 72 **Salida validación en el Monitor Serial: lecturas válidas por sensor al término de la prueba de estabilidad.** Fuente. Elaboración propia.

Como verificación complementaria del hardware, se midieron las tensiones de salida de las etapas de regulación del PCB bajo tres condiciones de carga: baja, nominal y alta, utilizando un multímetro digital en los puntos de prueba correspondientes a los rieles de 12 V, 5 V y 3,3 V. La configuración empleada en cada condición se muestra en la Tabla 38 y permitió evaluar el indicador RNF-05, considerando como criterio de aceptación una tolerancia máxima de ±4 % respecto al valor nominal.

Tabla 38 **Condiciones de carga utilizadas para la evaluación de las tensiones del PCB**

Condición	Elementos y estado de operación
Carga baja	Raspberry Pi, ESP32 y multiplexor TCA9548A en funcionamiento, con la tira LED RGB apagada y sin operación activa de los sensores. El riel de 12 V permanece prácticamente en reposo.
Carga nominal	Raspberry Pi, ESP32, multiplexor TCA9548A y sensores activos, con la tira LED RGB operando con un solo canal encendido.
Carga alta	Raspberry Pi, ESP32, multiplexor TCA9548A y sensores activos, con los tres canales de la tira LED RGB operando simultáneamente a máxima intensidad.

Fuente. Elaboración propia

Para cada riel se realizaron cinco mediciones con intervalos de 10 minutos en cada condición, obteniéndose 15 mediciones por riel y 45 mediciones en total. Los valores individuales se presentan en el Anexo B, mientras que la Tabla 39 muestra el

resumen estadístico mediante promedio, valores mínimo y máximo, desviación estándar y error relativo.

Tabla 39 **Resumen de las tensiones medidas en el PCB bajo diferentes condiciones de carga**

RIEL	Carga	Media (V)	Mín. (V)	Max. (V)	Desv. Estándar	Error relativo	Resultado
12,0 V	Baja	12.114	12.109	12.118	0.0034	0.95 %	✓
	Nominal	12.081	12.075	12.086	0.0041	0.67 %	✓
	Alta	12.035	12.029	12.041	0.0045	0.30 %	✓
5,0 V	Baja	5.100	5.098	5.103	0.0019	2.00 %	✓
	Nominal	5.079	5.076	5.082	0.0024	1.58 %	✓
	Alta	5.057	5.054	5.061	0.0027	1.15 %	✓
3,3 V	Baja	3.316	3.315	3.318	0.0011	0.50 %	✓
	Nominal	3.309	3.308	3.311	0.0011	0.28 %	✓
	Alta	3.301	3.300	3.303	0.0011	0.04 %	✓

Fuente. Elaboración propia

Como se observa en la Tabla 39, los tres rieles se mantuvieron dentro del margen de ±4 % en las tres condiciones evaluadas. Asimismo, las bajas desviaciones estándar evidencian una reducida variación entre las mediciones repetidas, incluso al incrementar la demanda eléctrica del sistema.

Finalmente, con el hardware funcional se realizó una prueba de funcionamiento en un entorno de producción controlado. La Figura 73 presenta el equipo instalado en el rack de 19 pulgadas, confirmando la compatibilidad mecánica del chasis con la

ranura estándar 1U, la ubicación de los sensores en el panel frontal y la integración de la iluminación LED RGB en el gabinete.



Figura 73 **Equipo instalado en rack.** Fuente. Elaboración propia.

V.1.2 Firmware de la Unidad de Adquisición y Control

Como resultado de la segunda etapa se obtuvo un firmware funcional para el ESP32 que integra en un ciclo continuo y autónomo la adquisición multisensorial, la serialización de datos en formato JSON, la publicación al broker MQTT y el control reactivo de la tira RGB mediante comandos recibidos por suscripción. La Figura 74 presenta la lógica de operación del firmware, donde se identifican las etapas de inicialización, el bucle principal de ejecución y el mecanismo de reconexión

automática ante pérdida de conectividad. Sobre esta base, se validó la hipótesis específica 2 midiendo la latencia de publicación MQTT y el tiempo de reconexión ante pérdida de WiFi y de enlace MQTT, presentados a continuación.

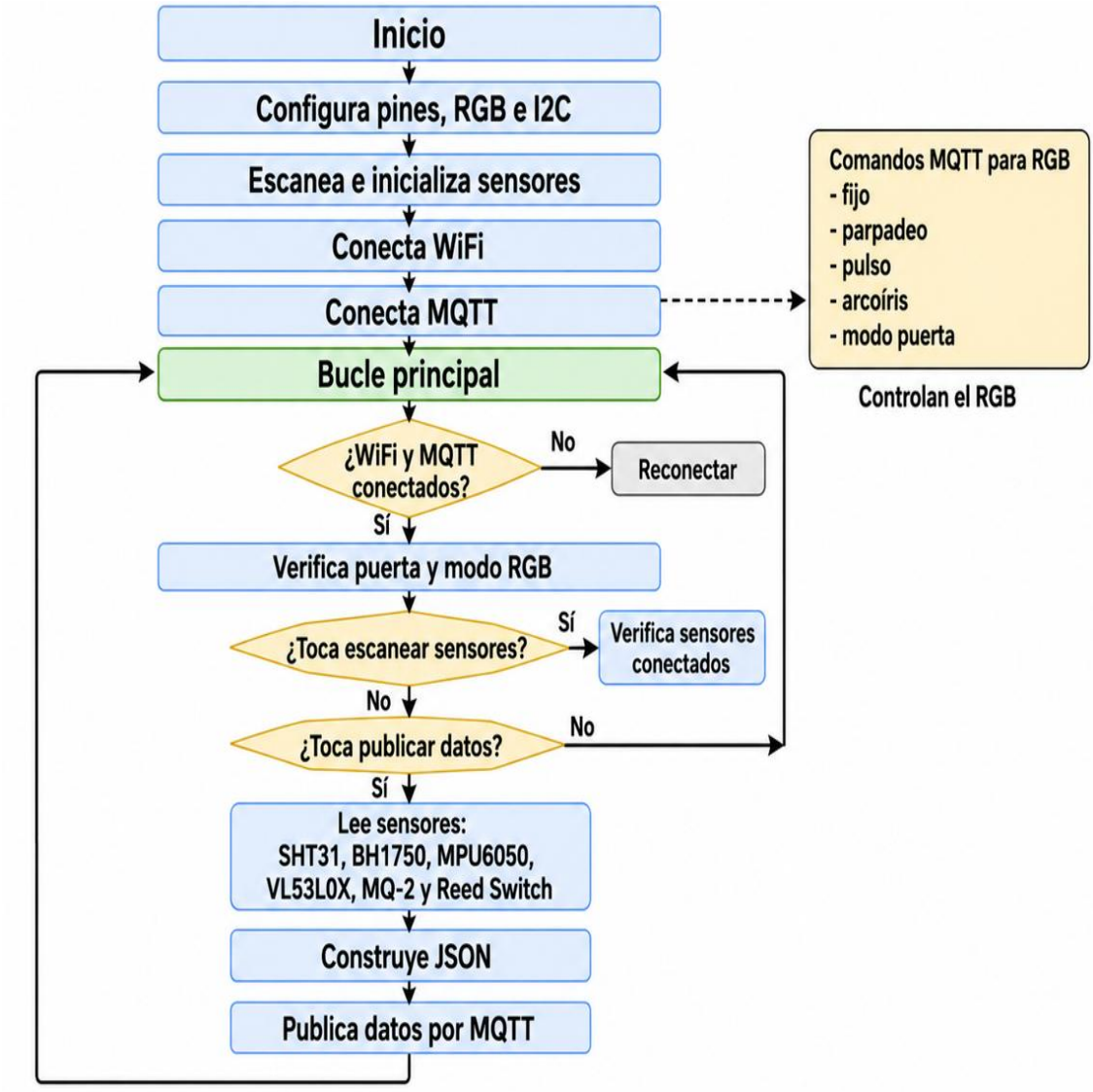


Figura 74 **Diagrama de flujo del firmware del ESP32.** Fuente. Elaboración propia.

Para complementar la descripción del firmware implementado, la Figura 75 presenta un fragmento representativo de la configuración inicial del firmware implementado en el ESP32. En este segmento se definen los parámetros principales

del nodo, como la red WiFi utilizada, la dirección del broker MQTT local, el puerto de comunicación, los tópicos de publicación y suscripción, el identificador del gabinete y los intervalos de operación para publicación y escaneo. Los fragmentos adicionales relacionados con la selección de canales I²C, reconexión WiFi/MQTT, suscripción a comandos y construcción del cuerpo JSON se presentan con mayor detalle en el Anexo C.

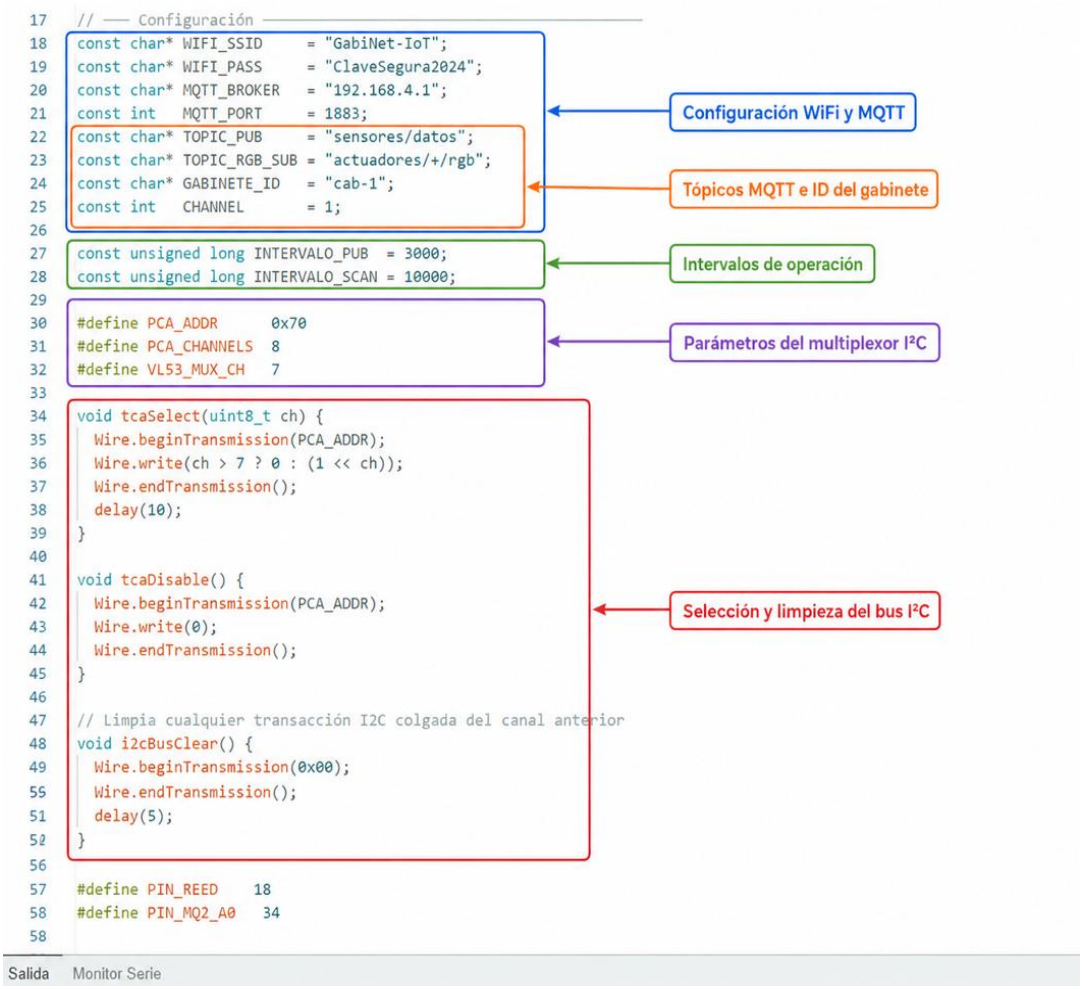


Figura 75 **Fragmento de configuración general del firmware del ESP32.** Fuente. Elaboración propia.

En cada iteración del bucle principal, el firmware verifica el estado de la conexión WiFi y MQTT antes de proceder; ante cualquier pérdida de enlace ejecuta el módulo de reconexión sin detener el ciclo, garantizando que el sistema se recupere de

forma autónoma sin requerir intervención manual. Si las conexiones están activas, lee todos los sensores activos, construye el paquete JSON y lo publica al broker cada 3 segundos, manteniendo un flujo constante y predecible de información hacia el Gateway IoT. La Figura 76 muestra el Monitor Serie durante ciclos consecutivos de operación, donde las marcas temporales confirman el intervalo de publicación y la línea pub=OK evidencia la transmisión exitosa en cada ciclo, permitiendo verificar de forma directa que el firmware cumple con la periodicidad esperada durante su funcionamiento continuo.

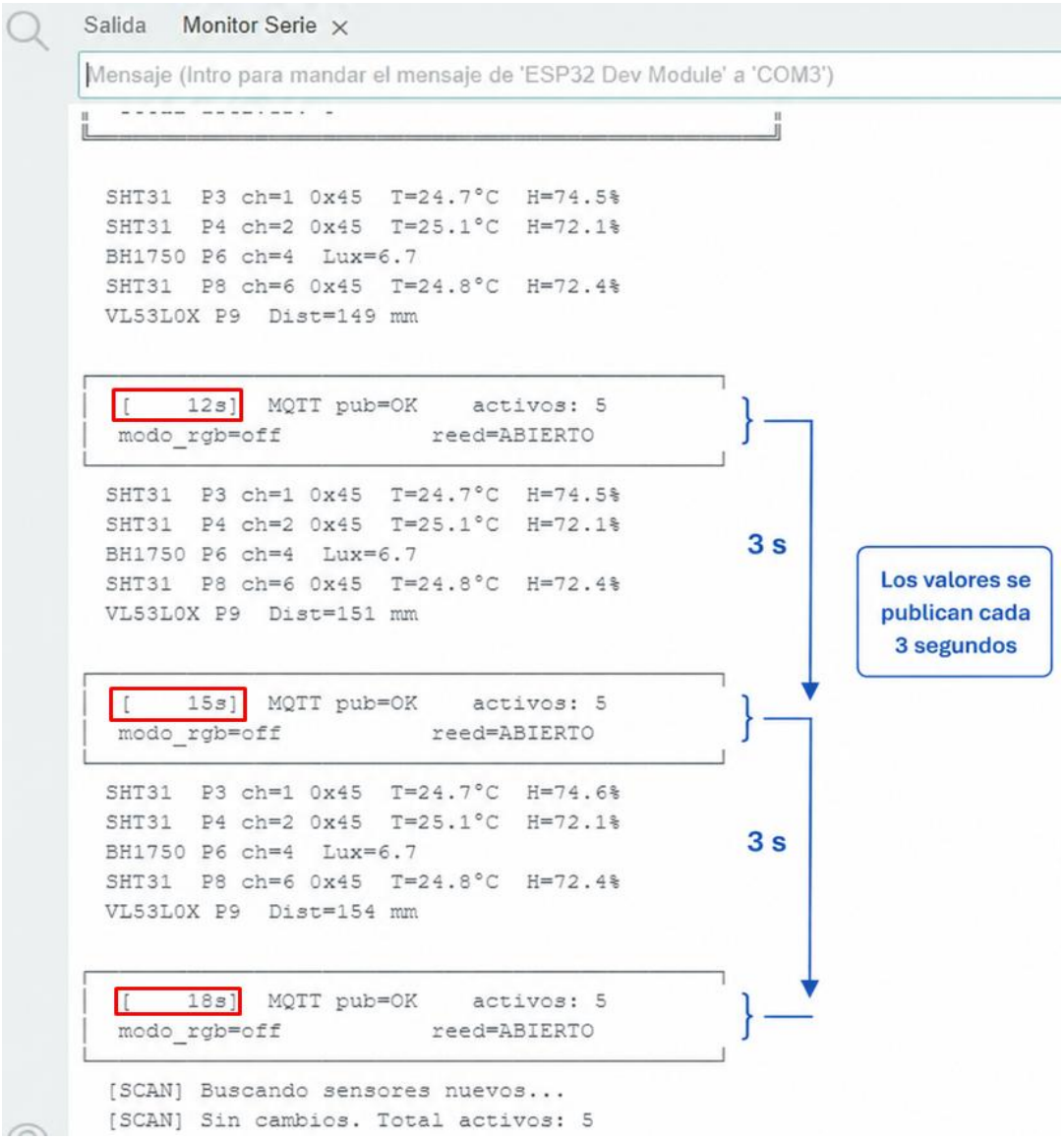


Figura 76 **Salida del Monitor Serie durante ciclo de publicación: lecturas por sensor y confirmación MQTT.** Fuente. Elaboración propia.

Para caracterizar el desempeño de la transmisión durante la etapa de validación, se incorporaron temporalmente en el firmware marcas temporales mediante la función millis(), con el fin de desglosar el tiempo de lectura de sensores, el tiempo de ejecución de mqtt.publish() y la latencia total por ciclo. Estas mediciones fueron añadidas como mecanismo de verificación para comprobar que el ciclo de adquisición y publicación cumpliera con los tiempos esperados. Como se observa en la Figura 77, la latencia de 409 ms se mantiene consistente entre ciclos sucesivos, con 406 ms correspondientes a la lectura de sensores y 3 ms a la publicación propiamente dicha Los estadísticos consolidados de los 30 ciclos se presentan en la Tabla 40, donde destaca que ningún ciclo superó el umbral de 500 ms y la tasa de publicaciones exitosas fue del 100 %.

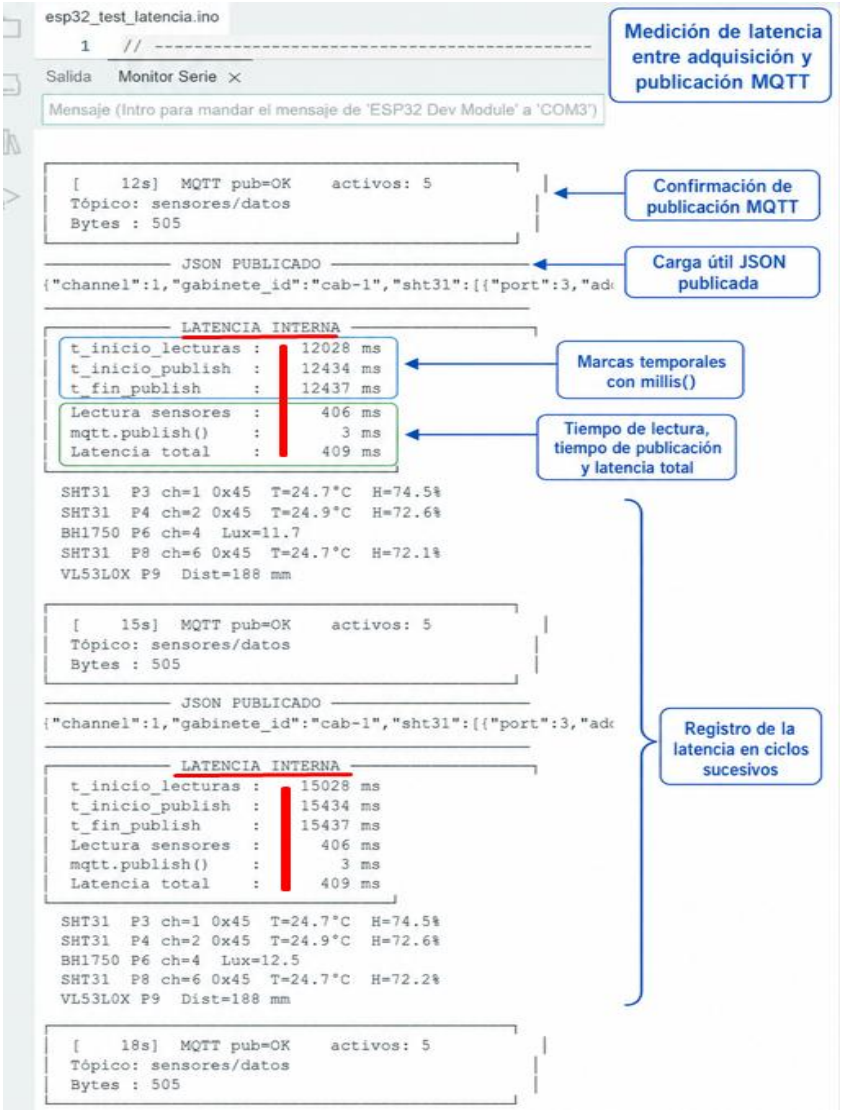


Figura 77 **Latencia registrada desde el Monitor Serial.** Fuente. Elaboración propia.

A partir de las mediciones realizadas durante los 30 ciclos de adquisición y publicación, se calcularon los principales estadísticos asociados al tiempo total desde la lectura de los sensores hasta la publicación del mensaje en el broker MQTT. La Tabla 40 resume los valores máximo, mínimo y promedio de latencia, así como su desviación estándar, el número de ciclos que superaron el umbral establecido y la tasa de publicaciones completadas correctamente.

Tabla 40 **Resultados de latencia de publicación MQTT (30 ciclos)**

Métrica	**Valor**
Latencia máxima	410,1 ms
Latencia mínima	407,9 ms
Latencia promedio (lectura - publicación)	409 ms
Desviación estándar	0,65 ms
Ciclos fuera de umbral (> 500 ms)	0 de 30
Tasa de publicaciones exitosas	100%

Fuente. Elaboración propia.

Con el fin de visualizar la estabilidad de la latencia a lo largo de toda la prueba, la Figura 78 grafica los 30 valores registrados en función del número de ciclo. Se observa que las mediciones se mantienen agrupadas alrededor del promedio de 409 ms, sin tendencias crecientes ni picos que se aproximen al umbral de 500 ms establecido por el indicador correspondiente.

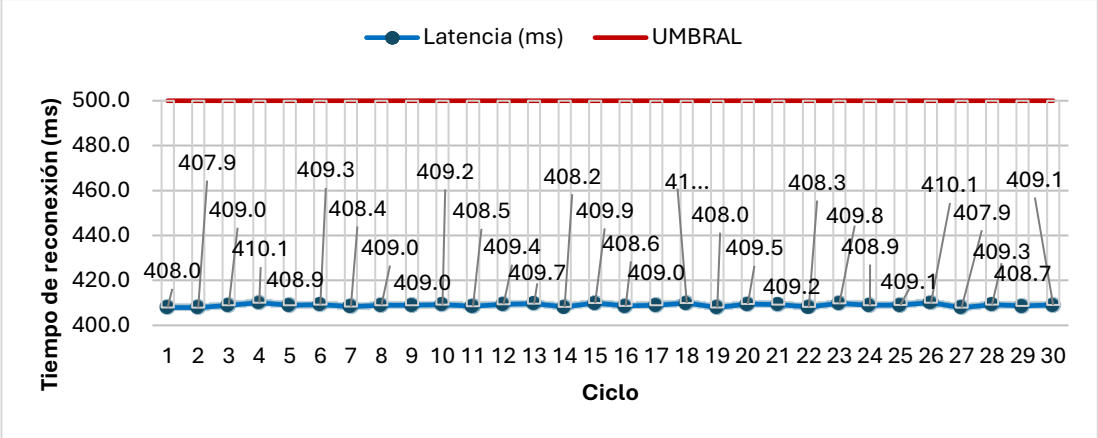


Figura 78 **Latencia de publicación MQTT por ciclo (n = 30).** Fuente. Elaboración propia.

La Tabla 41 siguiente ilustra el formato de los 30 ciclos individuales realizados durante la prueba. Asimismo, la Figura 79 muestra la composición de la latencia promedio de publicación MQTT, diferenciando el tiempo correspondiente a la lectura de sensores y al proceso mqtt.publish(), con el fin de identificar la contribución de cada etapa al tiempo total de publicación.

Tabla 41 **Muestra ilustrativa de 10 ciclos de publicación MQTT.**

Ciclo	Latencia (milisegundos)
1	408,0
2	407,9
3	409,0
4	410,1
5	408,9
6	409,3
7	408,4
8	409,0
9	409,0
10	409,2

Fuente. Elaboración propia. Muestra de datos.

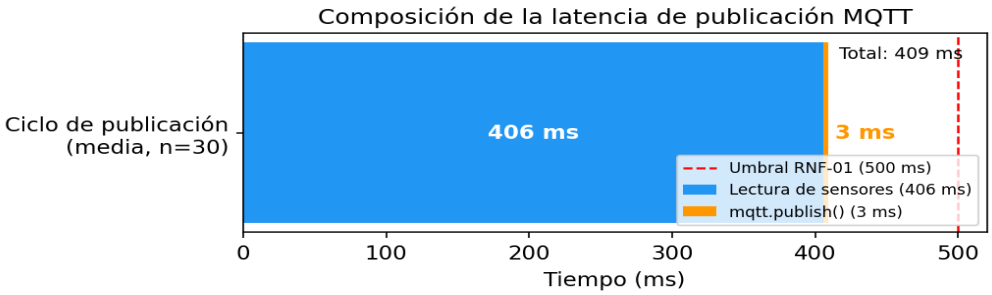


Figura 79 **Composición de la latencia de publicación MQTT: lectura de sensores frente a mqtt.publish() (media, n = 30).** Fuente. Elaboración propia.

Adicionalmente, se evaluó el comportamiento del firmware ante pérdida de conectividad durante el reinicio de servicios en el Gateway IoT provocando caída del punto de acceso WiFi e interrupción del enlace MQTT. La Figura 80 y Figura 81 muestran las secuencias registradas en el Monitor Serie para cada escenario, donde se aprecia que el firmware detecta la pérdida de enlace, ejecuta el proceso de reconexión y reanuda las publicaciones sin intervención manual. Los tiempos de recuperación promedio durante 30 ciclos de prueba son de 2,2 s ante caída de WiFi y 0,3s ante pérdida del enlace MQTT, tal como se resume en la Tabla 42.

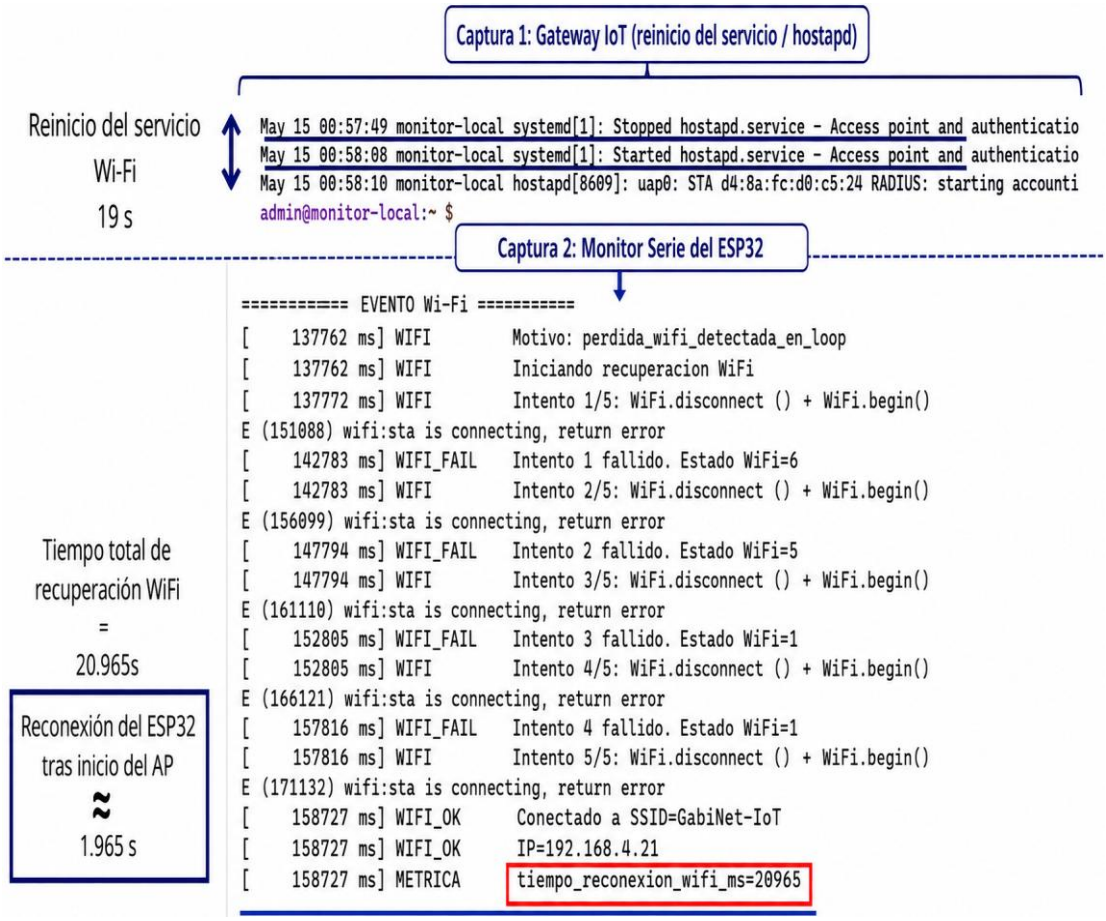


Figura 80 **Reconexión WiFi del ESP32 tras reinicio del servicio en el Gateway.** Registros de reconexión: arriba, el reinicio del servicio WiFi en el Gateway IoT; abajo, el Monitor Serie del ESP32 durante la detección de pérdida y posterior reconexión. Fuente. Elaboración propia.

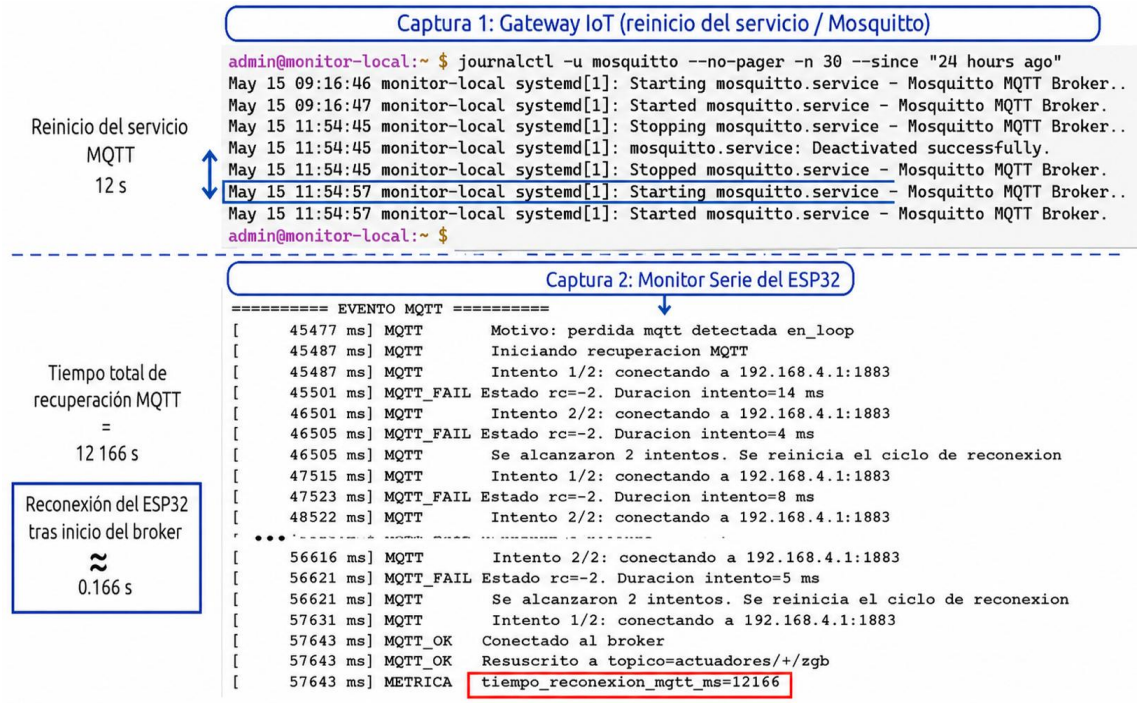


Figura 81 **Reconexión al bróker MQTT del ESP32 tras reinicio del servicio en el Gateway.** Fuente. Elaboración propia.

Tabla 42 **Tiempos de reconexión automática**

Escenario / Condición simulada	**Tiempo máximo de recuperación**	**Tiempo mínimo de recuperación**	**Desviación estándar de tiempo de recuperación**	**Tiempo promedio de recuperación**
Pérdida de WiFi: Reinicio del servicio WiFi	3,1s	1,6s	0,33s	2,2 s
Pérdida de enlace MQTT: Reinicio del servicio MQTT	0,5	0,2	0,09s	0,3 s

Fuente. *Elaboración propia.*

Con el fin de visualizar la consistencia del proceso de reconexión a lo largo de la prueba, la Figura 82 grafica los tiempos de recuperación registrados para ambos escenarios en función del número de ciclo. Se observa que los tiempos de reconexión WiFi varían hasta 0,9 s alrededor del promedio de 2,2 s, mientras que los de reconexión MQTT varían hasta 0,2 s en torno a los 0,3 s de tiempo promedio.

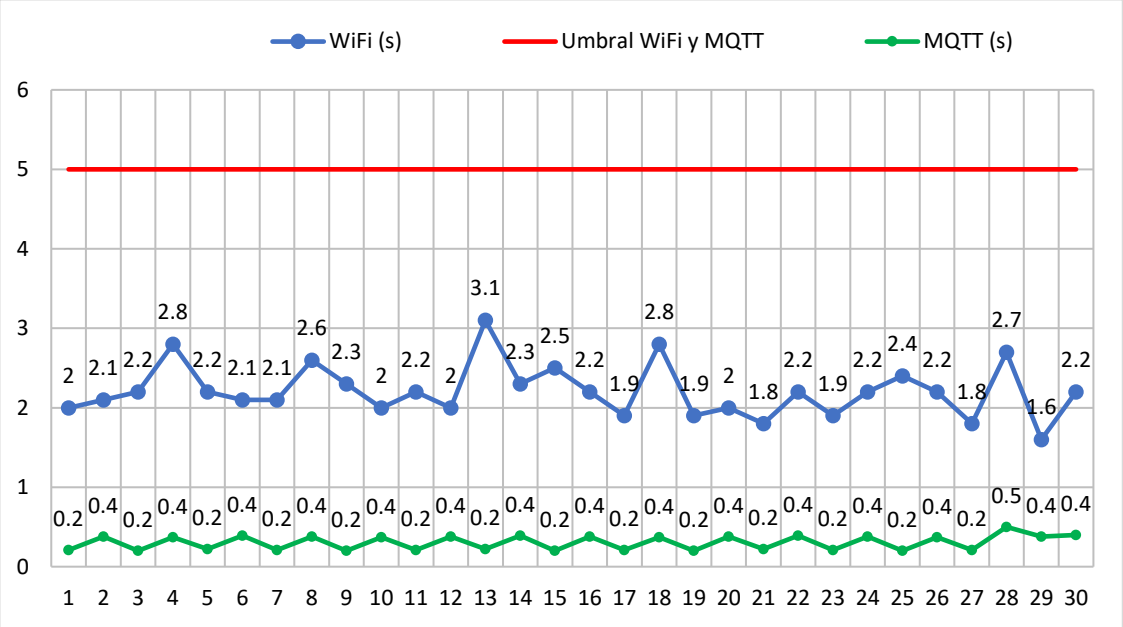


Figura 82 **Tiempos de reconexión automática por ciclo - WiFi vs. MQTT (n = 30).** Fuente. Elaboración propia.

Como resultado de esta etapa, se obtuvo un firmware operativo para el ESP32 con capacidad de adquisición multisensorial, publicación MQTT periódica, control de actuadores RGB por suscripción y reconexión automática ante pérdida de conectividad. El ciclo de operación quedó configurado con publicaciones cada 3 segundos hacia el tópico sensores/datos, usando un bloque de datos JSON consolidado con las lecturas de los sensores y el estado del actuador. Durante las pruebas de validación se registró una latencia promedio de publicación de 409 ms en 30 ciclos, sin ciclos fuera del umbral de 500 ms y con una tasa de publicaciones exitosas del 100 %. Además, se obtuvo un tiempo promedio de recuperación de 2,2 s ante pérdida de WiFi y de 0,3 s ante interrupción del enlace MQTT.

V.1.3 Modelo de Detección de Anomalías mediante Isolation Forest

Como resultado de esta etapa se obtuvo un modelo Isolation Forest entrenado y desplegado como microservicio autónomo en el Gateway IoT, capaz de clasificar en tiempo real las condiciones del gabinete como normales o anómalas a partir de las lecturas recibidas por MQTT. El modelo opera sobre una ventana de 10 minutos de lecturas, ejecutando un ciclo de inferencia cada 10 minutos de forma continua.

La Figura 83 presenta la lógica de inferencia del modelo desplegado, desde la recepción de datos hasta la generación de alerta, los parámetros del modelo resultante se consolidan en la Tabla 43, donde destacan las 11 características (features) de entrada que combinan variables físicas directas, una variable derivada de tasa de cambio térmico y dos componentes cíclicos de hora para capturar el comportamiento temporal del gabinete. Para validar la hipótesis específica 3, se evaluó el modelo mediante una matriz de confusión y estadística descriptiva e inferencial sobre los scores por clase, sobre un conjunto de 1 200 observaciones, presentados a continuación.

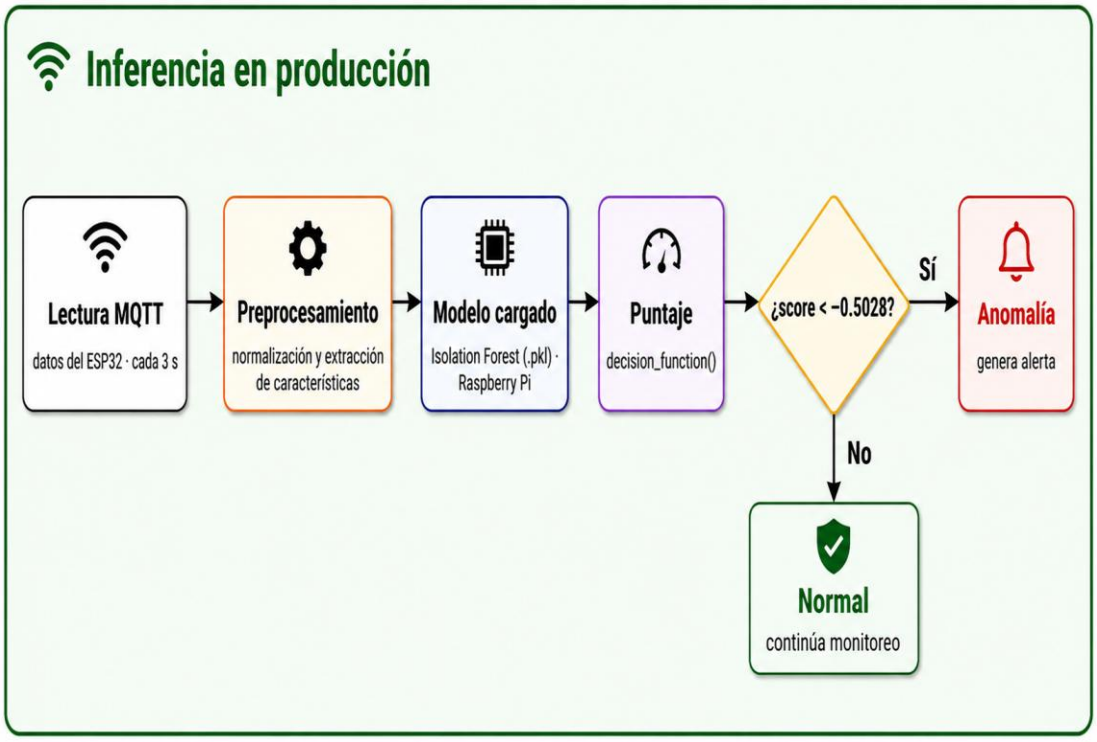


Figura 83 **Lógica de inferencia del modelo Isolation Forest desplegado en el Gateway IoT.** Fuente. Elaboración propia.

Tabla 43 **Parámetros del modelo Isolation Forest desplegado**

Parámetro	**Valor**
Algoritmo	Isolation Forest
n_estimators	200
contamination	0,05
random_state	42
Normalización	StandardScaler
Características (features) de entrada	11 (temperatura, humedad, luz, distancia, humo, movimiento, reed, mpu6050, temp_rate, hora_sin, hora_cos)
Umbral de clasificación	decision_function < 0 (equivalente a score_samples < −0,5028)
Intervalo de inferencia	10 min
Ventana de lectura	Últimos 10 min

Fuente. Elaboración propia

Con el modelo desplegado y sus parámetros definidos, la Figura 84 muestra la evolución del score del modelo desplegado. La línea azul representa el puntaje continuo, la línea punteada roja el umbral de clasificación en cero, y los puntos rojos los eventos clasificados como anomalías. Durante los primeros nueve días (16–24 de marzo) el score se mantiene estable y consistentemente por encima del umbral, sin ninguna anomalía detectada, lo que es coherente con la operación normal del gabinete en ese período. A partir del 25 de marzo el score comienza a descender y desde el 26 de marzo cruza el umbral de forma reiterada, con puntajes que alcanzan valores de hasta −0,05; los puntos de anomalía se agrupan en tres intervalos, alrededor del 26, 27 y 29 de marzo, intercalados con recuperaciones transitorias, lo que indica que las condiciones anómalas fueron episódicas y no continuas. Este patrón evidencia que el

modelo responde a desviaciones reales del entorno del gabinete y no genera detecciones arbitrarias.

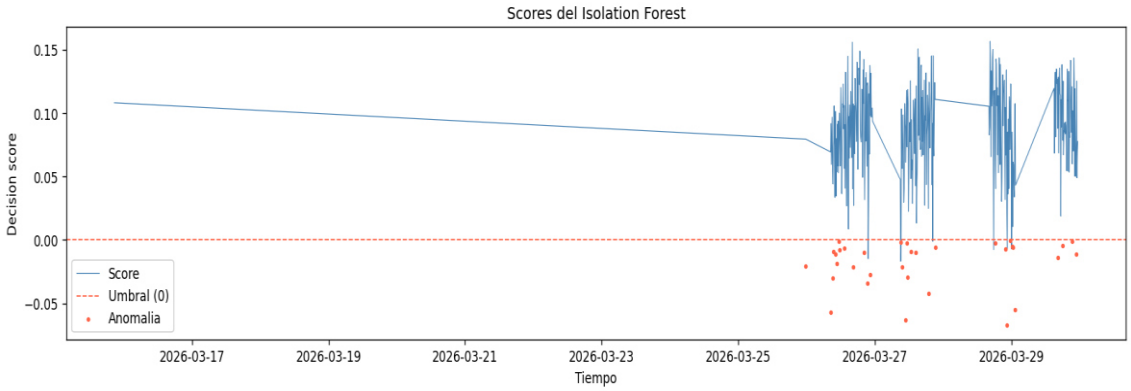


Figura 84 **Puntuación del Isolation Forest sobre datos reales: evolución temporal de la puntuación y eventos de anomalía detectados.** Fuente. Elaboración propia.

Para evaluar formalmente la capacidad de detección, se construyó un conjunto de 1 200 observaciones balanceadas compuesto por 600 muestras normales reales del período 30 de abril al 8 de mayo de 2026, y 600 anomalías sintéticas representando cinco escenarios operacionales relevantes en un gabinete Tier 1. La Figura 85 presenta la matriz de confusión obtenida sobre el conjunto de evaluación de 1 200 observaciones (600 normales reales + 600 anomalías sintéticas). El eje vertical indica la etiqueta real y el eje horizontal la etiqueta predicha por el modelo; la intensidad del azul en cada celda refleja su densidad relativa. El cuadrante superior izquierdo registra 573 verdaderos negativos correspondientes a observaciones normales correctamente clasificadas y que representan el 95,5 % de las muestras normales. El cuadrante inferior derecho registra 596 verdaderos positivos correspondientes a anomalías correctamente detectadas y equivalentes al 99,3 % de las 600 anomalías del conjunto. El cuadrante superior derecho muestra 27 falsos positivos, muestras normales clasificadas erróneamente como anomalías que constituyen el error principal del modelo con un 4,5 % sobre la clase normal. El cuadrante inferior izquierdo registra únicamente 4 falsos negativos, entendidos como anomalías no detectadas, todos provenientes del escenario de vibración anómala según el análisis de la Figura 86. La marcada asimetría entre ambos tipos de error, 27 falsos positivos frente a solo 4 falsos

negativos, es operacionalmente favorable en el dominio de monitoreo de infraestructura, donde no detectar una anomalía real tiene mayor costo que generar una alerta falsa. El resumen de las métricas de clasificación descritas se presenta en la Tabla 44.

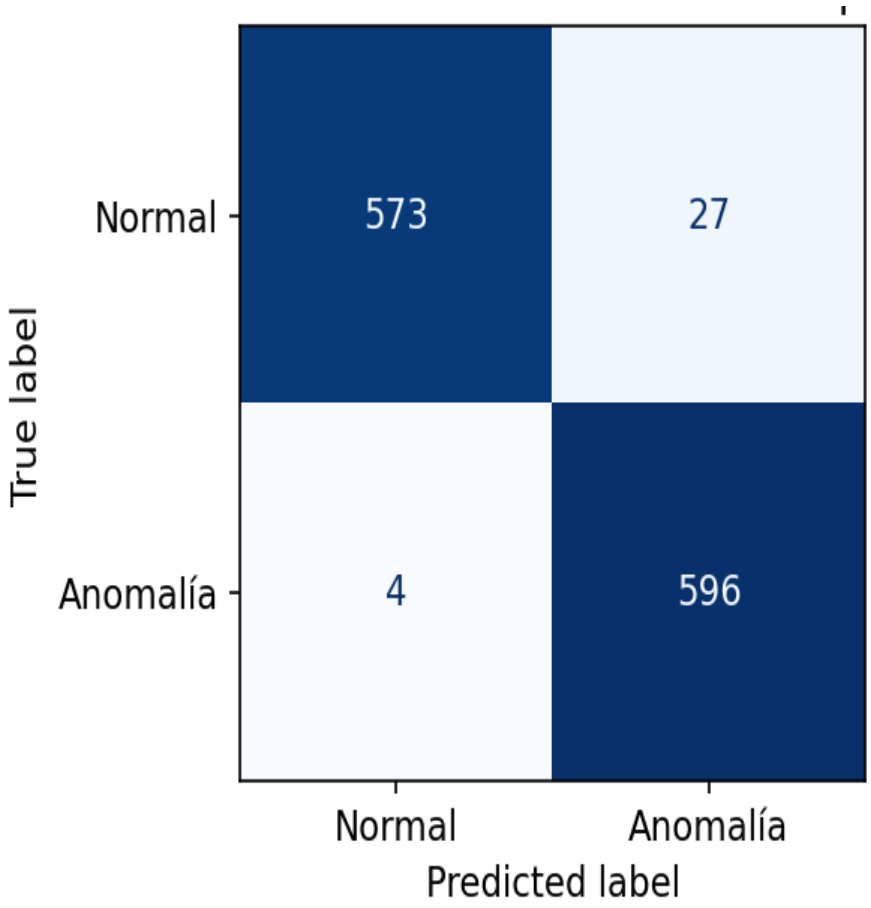


Figura 85 **Matriz de confusión del Isolation Forest - evaluación (n = 1 200).** Fuente. Umbral de clasificación: score < −0,5028. Elaboración propia.

Tabla 44 **Métricas de clasificación del modelo Isolation Forest (n = 1 200)**

Métrica	Valor
Verdaderos Positivos (VP)	596

Falsos Positivos (FP)	27
Verdaderos Negativos (VN)	573
Falsos Negativos (FN)	4
Precisión	95,67 %
Recall (sensibilidad)	99,33 %
F1-score	97,47 %
Exactitud (accuracy)	97,42 %

Fuente. Elaboración propia

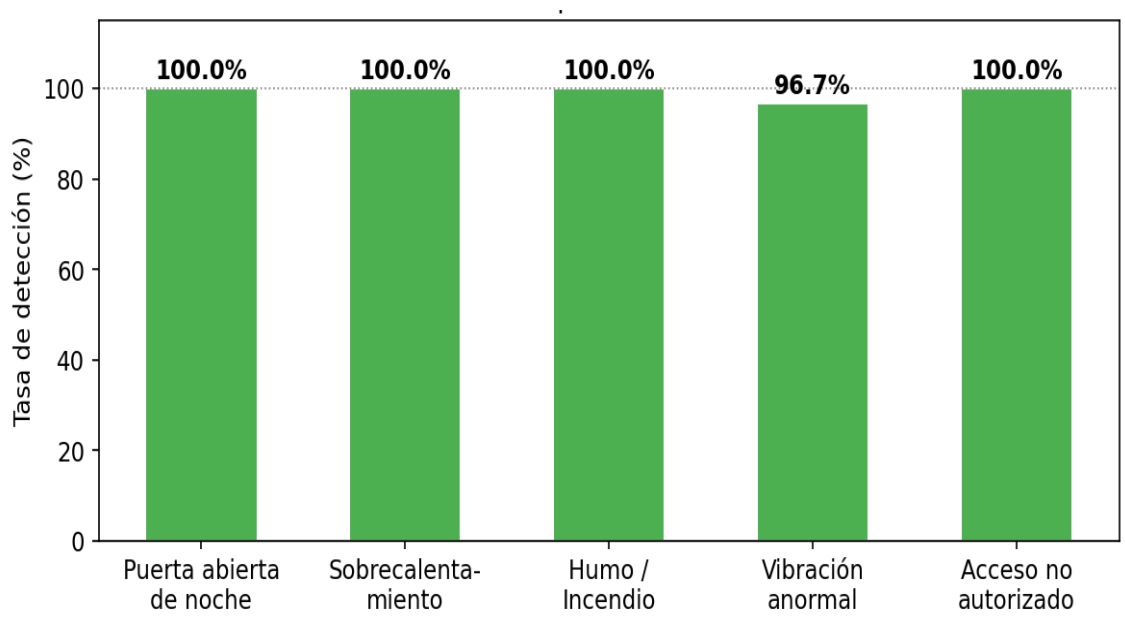


Figura 86 **Tasa de detección por escenario de anomalía (120 muestras por escenario).** Fuente. Elaboración propia.

Más allá de las métricas de clasificación, la Figura 87 permite visualizar que el modelo logra esa separación debido a que la distribución de puntajes muestra que las observaciones normales ($\mu = 0{,}054$, $\sigma = 0{,}049$) y las anómalas ($\mu = -0{,}138$, $\sigma = 0{,}040$)

forman dos nubes estadísticamente bien diferenciadas, con una diferencia entre medias de $\Delta\mu = 0{,}192$ equivalente a más de cuatro desviaciones estándar combinadas, confirmando que el modelo aprendió correctamente el patrón de operación normal del gabinete. Esta separabilidad estructural se complementa con el tiempo de inferencia consolidado en la Tabla 45 donde se registra un tiempo por muestra de 0,057 ms sobre las 1 200 muestras evaluadas, la latencia de inferencia no constituye un cuello de botella operacional en el Gateway IoT.

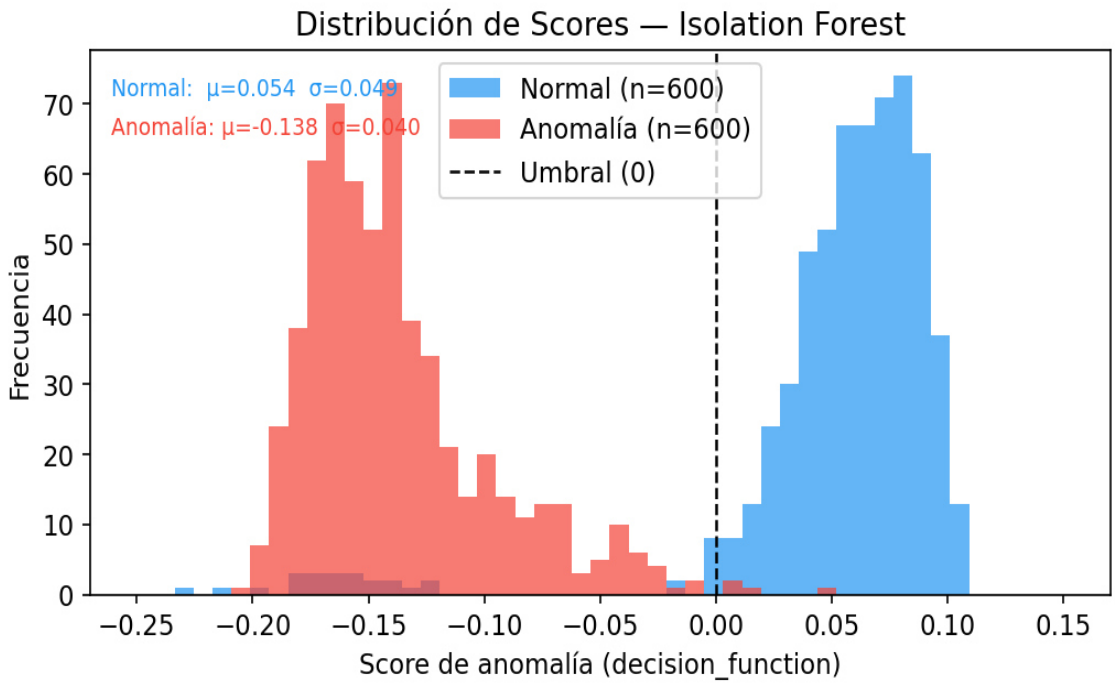


Figura 87 **Distribución del decision_function por clase: normal vs. Anomalía.** Fuente. Elaboración propia.

Tabla 45 **Tiempo de inferencia del modelo Isolation Forest**

Métrica	**Valor**
Tiempo total (1200 muestras)	68,83 ms

Tiempo por muestra		0,057 ms

Fuente. Elaboración propia

Para verificar cuantitativamente la hipótesis específica 3 más allá de la matriz de confusión, se aplicó estadística descriptiva e inferencial sobre los 1 200 scores del modelo: (i) media, mediana, desviación estándar, cuartiles y asimetría por clase; (ii) prueba de normalidad de Shapiro-Wilk; y (iii) prueba U de Mann-Whitney con tamaño del efecto (d de Cohen). La Tabla 46 resume los estadísticos por clase.

Tabla 46 **Estadísticos descriptivos del score de anomalía por clase (n = 1 200).**

Estadístico	**Normal (n = 600)**	**Anómala (n = 600)**
Media	0,054	−0,138
Mediana	0,063	−0,146
Desviación estándar	0,049	0,040
Mínimo	−0,232	−0,204
Cuartil 1 (Q1)	0,042	−0,166
Cuartil 3 (Q3)	0,080	−0,122
Máximo	0,109	0,044
Rango intercuartílico (IQR)	0,039	0,044
Asimetría (skewness)	−3,310	1,273

Fuente. Elaboración propia a partir de scores_detalle.csv.

Shapiro-Wilk rechazó la normalidad en ambas clases (Normal: W = 0,636; Anómala: W = 0,914; p < 0,001), por lo que la comparación se realizó con la prueba no paramétrica U de Mann-Whitney en vez de una prueba t. El resultado (U = 350 851; p < 0,001) confirma que la diferencia entre medianas no se debe al azar. El tamaño del efecto (d de Cohen = 4,28) corresponde a un efecto muy grande (Cohen, 1988), muy por encima del umbral convencional.

Tabla 47 **Prueba de significancia estadística y tamaño del efecto sobre la separabilidad de clases.**

Prueba / indicador	Estadístico	Valor	Interpretación
Normalidad Shapiro-Wilk - Normal	W	0,636 (p < 0,001)	No normal
Normalidad Shapiro-Wilk - Anómala	W	0,914 (p < 0,001)	No normal
Diferencia entre clases (U de Mann-Whitney)	U	350 851 (p < 0,001)	Diferencia significativa
Tamaño del efecto (d de Cohen)	d	4,28	Efecto muy grande (Cohen, 1988)

Fuente. Elaboración propia. Nivel de significancia α = 0,05.

La Figura 88 presenta esta distribución mediante diagramas de caja, considerando la mediana, el rango intercuartílico y los valores atípicos para las 600 observaciones normales y las 600 observaciones anómalas. En la clase normal, la mediana fue de 0,063, con un rango intercuartílico entre 0,042 y 0,080; mientras que, en la clase anómala, la mediana fue de −0,146, con un rango intercuartílico entre

−0,166 y −0,122. Esta representación permite observar la posición y dispersión de los scores de cada clase respecto al umbral de clasificación.

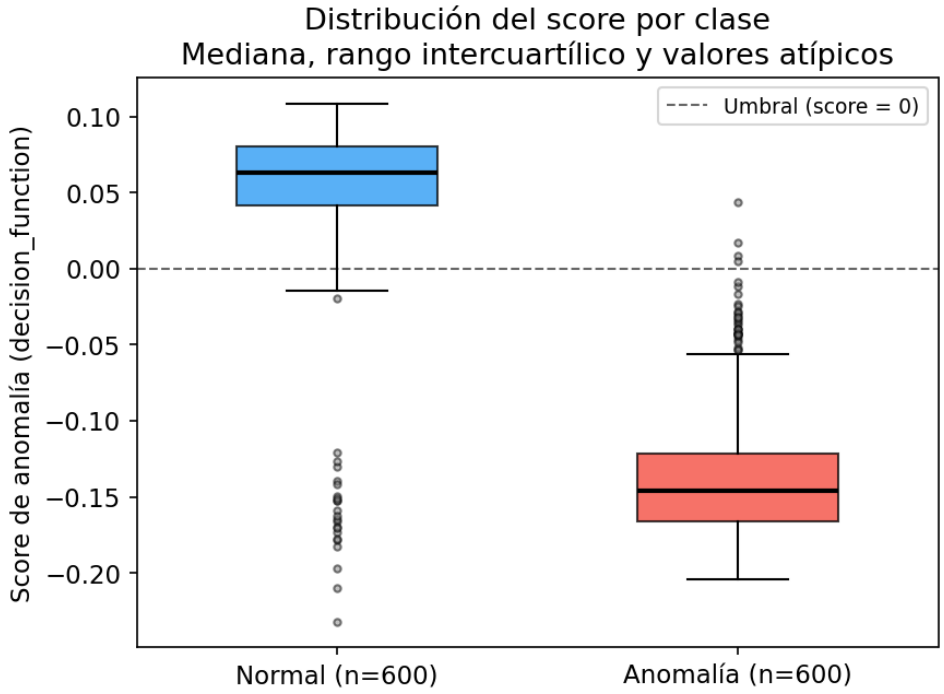


Figura 88 **Diagrama de caja del score de anomalía por clase: mediana, rango intercuartílico y valores atípicos.** Fuente. Elaboración propia

El modelo Isolation Forest desplegado alcanzó un recall de 99,33 % y un F1-score de 97,47 % sobre el conjunto de evaluación, con detección perfecta en cuatro de los cinco escenarios operacionales y un tiempo de inferencia de 0,057 ms por muestra, confirmando que el sistema detecta condiciones anómalas en el gabinete de forma precisa, autónoma y con latencia despreciable frente al ciclo de operación del sistema.

V.1.4 Gateway IoT Basado en Raspberry Pi

Como resultado de esta etapa se obtuvo un Gateway IoT como nodo central basado en Raspberry Pi 4B, integrado en el chasis rack 1U, capaz de generar una red WiFi dedicada para la UAC, gestionar el direccionamiento IP dinámico, operar el broker de mensajería MQTT y persistir la telemetría recibida en la base de datos local

MariaDB. La Figura 89 muestra el Gateway integrado en el chasis rack 1U en operación, donde se identifican sus bloques funcionales internos y su interconexión con el PCB principal de la UAC mediante ribbon cable. Para validar la hipótesis específica 4, se midió la latencia del broker MQTT y de la sincronización con MariaDB, sobre 30 y 20 muestras respectivamente, presentadas a continuación.

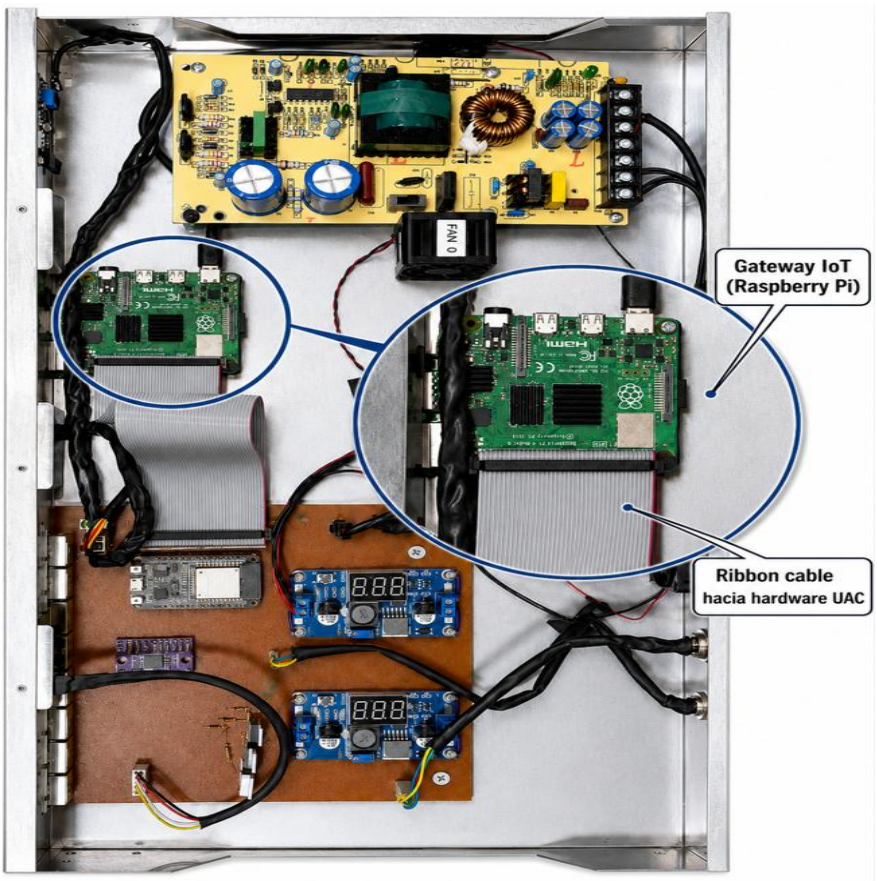


Figura 89 **Gateway IoT basado en Raspberry Pi 4B integrado en el chasis rack 1U.** Fuente Elaboración propia.

Con el Gateway operativo en la red LAN, la Figura 90 confirma su presencia mediante escaneo con el software Advanced IP Scanner, software que permite detectar equipos de un segmento de red especifico, donde se aprecia la dirección IP estática 192.168.0.25 asignada a la interfaz Ethernet eth0 y la MAC d8:3a:dd:49:70:19, verificando que el nodo es accesible y direccionable dentro de la infraestructura de red del centro de datos.

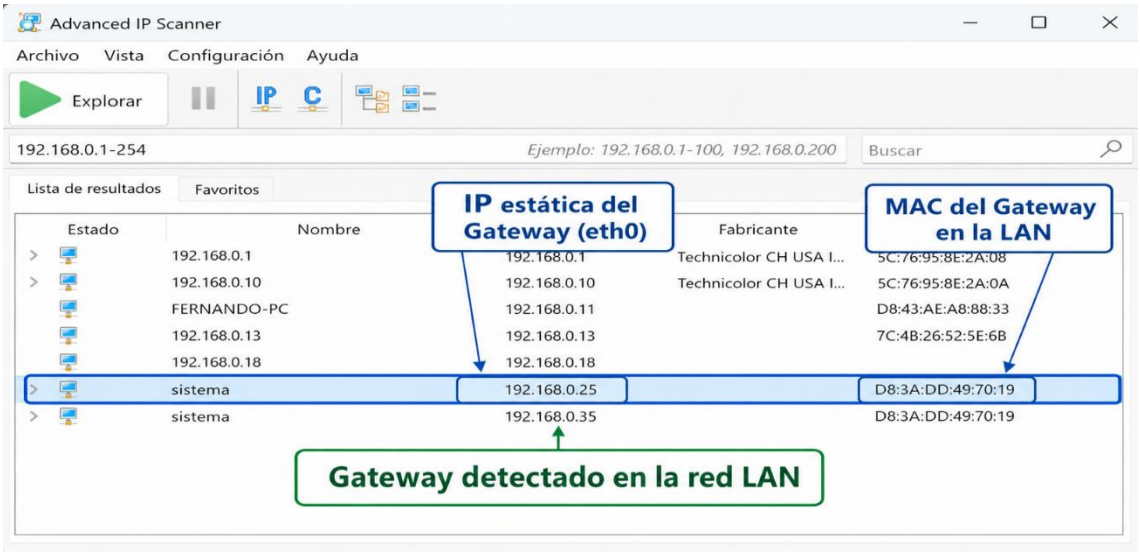


Figura 90 **Verificación de la presencia del Gateway IoT en la red LAN mediante Advanced IP Scanner.** Fuente. Elaboración propia.

La operación del punto de acceso WiFi dedicado a la UAC se evidencia en la Figura 91, que presenta la secuencia completa de negociación DHCP registrada mediante journalctl: los eventos DHCPDISCOVER, DHCPOFFER y DHCPACK confirman que la UAC recibió automáticamente la dirección 192.168.4.21 dentro del rango reservado 192.168.4.10-192.168.4.50. El ciclo se cierra desde el lado de la UAC en la Figura 92, donde el Monitor Serie del ESP32 confirma la conexión exitosa al SSID GabiNet-IoT con la IP asignada, completando el enlace extremo a extremo entre el punto de acceso y la unidad de adquisición.

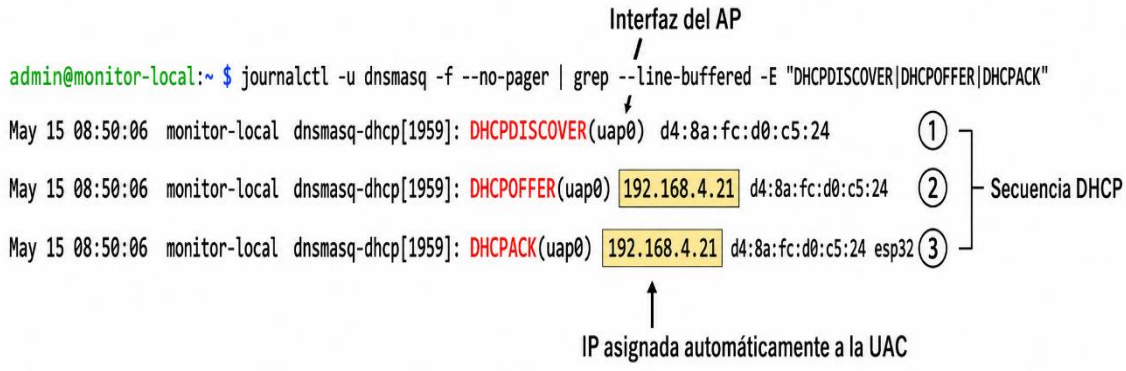


Figura 91 **Ciclo DHCP completo registrado por dnsmasq: la UAC recibe IP automáticamente al conectarse al AP.** Fuente. Elaboración propia.

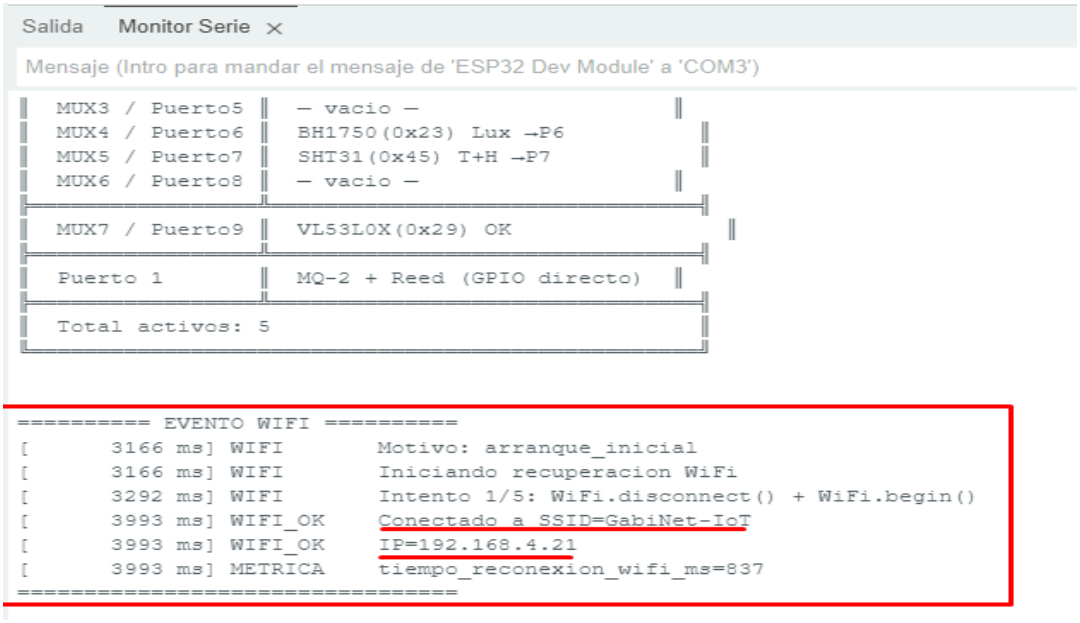


Figura 92 **Conexión de la UAC al AP del Gateway confirmada desde el Monitor Serie del ESP32: SSID conectado e IP asignada.** Fuente. Elaboración propia.

Para caracterizar el desempeño del broker, se cuantificó la latencia de Mosquitto mediante mosquitto_pub con QoS 1 y marca temporal en milisegundos sobre 30 muestras tanto en operación con un único sensor activo y operación con todos los sensores activos mediante variables en el firmware de la UAC. En ambos casos la latencia media se mantuvo por debajo de los 15 ms, muy por debajo del umbral de 500 ms establecido en los requisitos, con tasa de entrega del 100 % en las 30 mediciones realizadas, tal como se consolida en la Tabla 48.

Tabla 48 **Latencia del broker MQTT Mosquitto (n = 30 muestras, QoS 1)**

Escenario	Media	Mínima	Máxima	Desviación estándar	Resultado
Sin sensores adicionales (solo VL53L0X)	12 ms	9 ms	19 ms	3,6 ms	✓

Todos los sensores activos	14 ms	10 ms	22 ms	4,1 ms	✓

Fuente. Elaboración propia

La sincronización entre el broker y la base de datos se evidencia en la Figura 93, donde se presenta el mensaje publicado en el tópico sensores/datos y el registro correspondiente almacenado en MariaDB con su marca temporal. La latencia de esta sincronización, medida mediante el script de automatización mostrado en el Anexo E, sobre 20 muestras con intervalo de 400 ms, se consolida en la Tabla 49, donde la media de 13,3 ms y la ausencia de muestras fuera de umbral confirman la entrega consistente de cada mensaje publicado hacia la base de datos local. Estos resultados evidencian un comportamiento estable del sistema incluso bajo mediciones repetidas, lo que respalda la confiabilidad del mecanismo de sincronización implementado y su capacidad para sostener el flujo de datos requerido por la plataforma sin generar retrasos perceptibles para el usuario final.

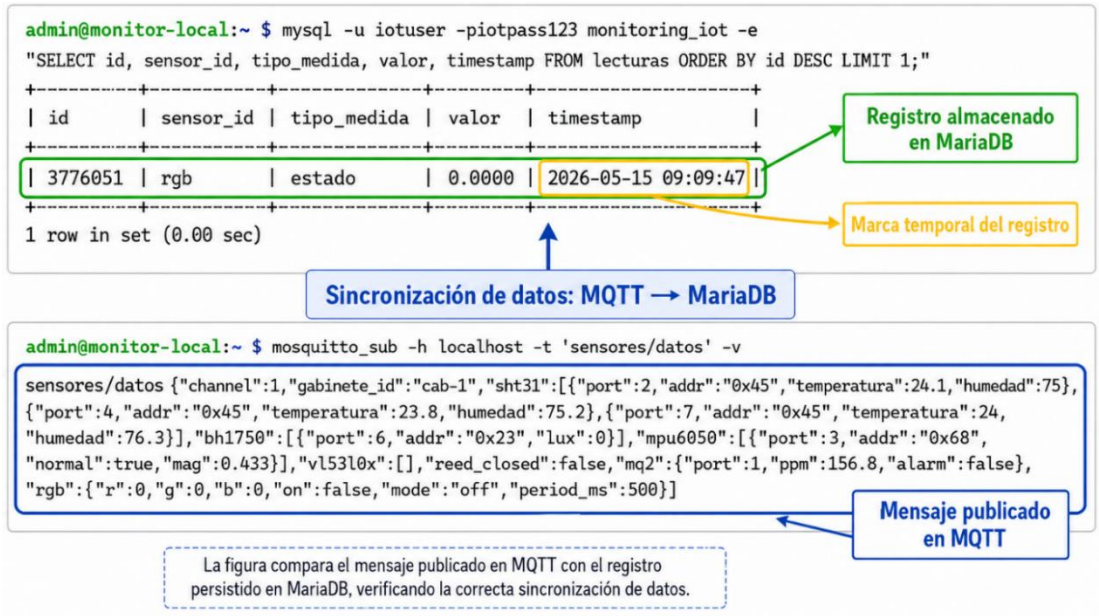


Figura 93 **Sincronización verificada: mensaje MQTT publicado y registro correspondiente en MariaDB con marca temporal.** Fuente. Elaboración propia.

Tabla 49 **Latencia de sincronización MQTT → MariaDB (n = 20 mediciones). Indicador: tasa de eficiencia en procesamiento de mensajes MQTT**

Estadístico	**Valor**
Media aritmética	13,3 ms
Valor mínimo	11 ms
Valor máximo	22 ms
Desviación estándar	3,7 ms
Muestras fuera de umbral (> 500 ms)	0 de 20

Fuente. Elaboración propia

La Tabla 50 y la Tabla 51 ilustran el formato de las mediciones individuales de ambos indicadores, generadas de forma consistente con la media, el mínimo, el máximo y la desviación estándar reportados.

Tabla 50. **Muestra ilustrativa de 10 mediciones de latencia del broker MQTT.**

Muestra	**Sin sensores adicionales (ms)**	**Todos los sensores activos (ms)**
1	9,0	10,0
2	13,9	10,5
3	14,4	10,0
4	9,0	12,6
5	11,1	13,9
6	15,4	15,5
7	9,0	12,4

8	10,9	11,5
9	12,7	13,6
10	19,0	22,0

Fuente. Elaboración propia. Muestra ilustrativa.

Tabla 51 **Muestra ilustrativa de 10 mediciones de sincronización MQTT→MariaDB.**

Muestra	Latencia (milisegundos)
1	11,0
2	12,7
3	11,0
4	16,7
5	14,2
6	12,0
7	15,8
8	12,8
9	19,9
10	22,0

Fuente. Elaboración propia. Muestra ilustrativa

La Figura 94 presenta las latencias promedio registradas en el Gateway IoT para la comunicación con el broker MQTT y la sincronización con MariaDB, incluyendo su desviación estándar, con el propósito de comparar el tiempo de

respuesta de cada proceso y verificar su comportamiento frente al umbral de 500 ms establecido para la prueba.

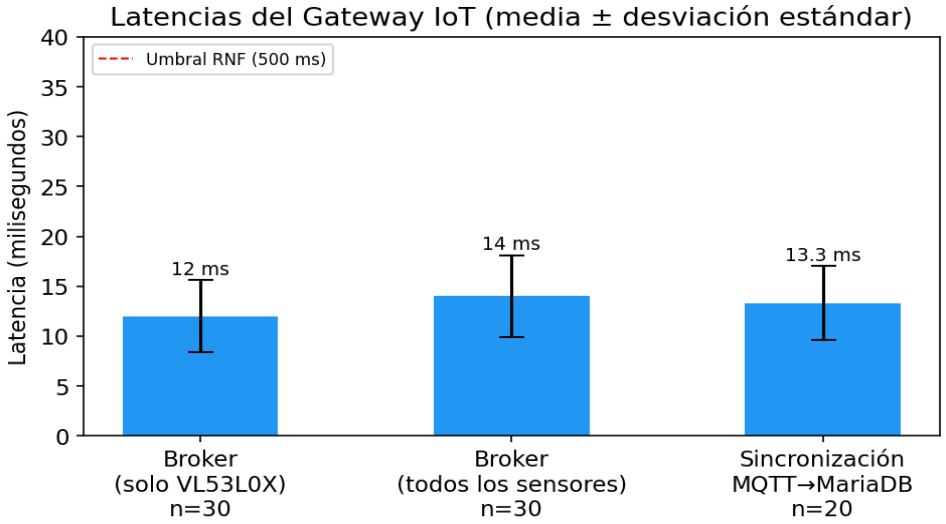


Figura 94 **Latencias del Gateway IoT: broker MQTT y sincronización con MariaDB (media ± desviación estándar).** Fuente. Elaboración propia.

El Gateway IoT basado en Raspberry Pi 4B opera como nodo central funcional dentro del sistema ya que gestiona la red WiFi dedicada con asignación DHCP automática hacia la UAC, procesa los mensajes MQTT con latencia promedio de 12-14 ms y persiste la telemetría en MariaDB con latencia promedio de 13,3 ms y tasa de entrega del 100 %.

V.1.5 Plataforma de Software para monitoreo y Control

Como resultado de esta etapa se obtuvo una plataforma web funcional, capaz de recibir telemetría en tiempo real, visualizar las variables del gabinete mediante tablero de monitoreo web, gestionar usuarios con perfiles diferenciados, generar alertas automáticas ante condiciones anómalas y controlar remotamente los actuadores. La Figura 95 muestra la plataforma en operación con lecturas activas de todos los sensores del gabinete, donde se aprecia el listado de dispositivos con su estado en línea, tipo de medida, puerto asignado y última lectura registrada. Para

validar la hipótesis específica 5, se midió el tiempo de actualización del panel de monitoreo y el tiempo de notificación de alerta, sobre 10 mediciones cada uno utilizando parte de la programación del software para medir el tiempo, presentados a continuación.

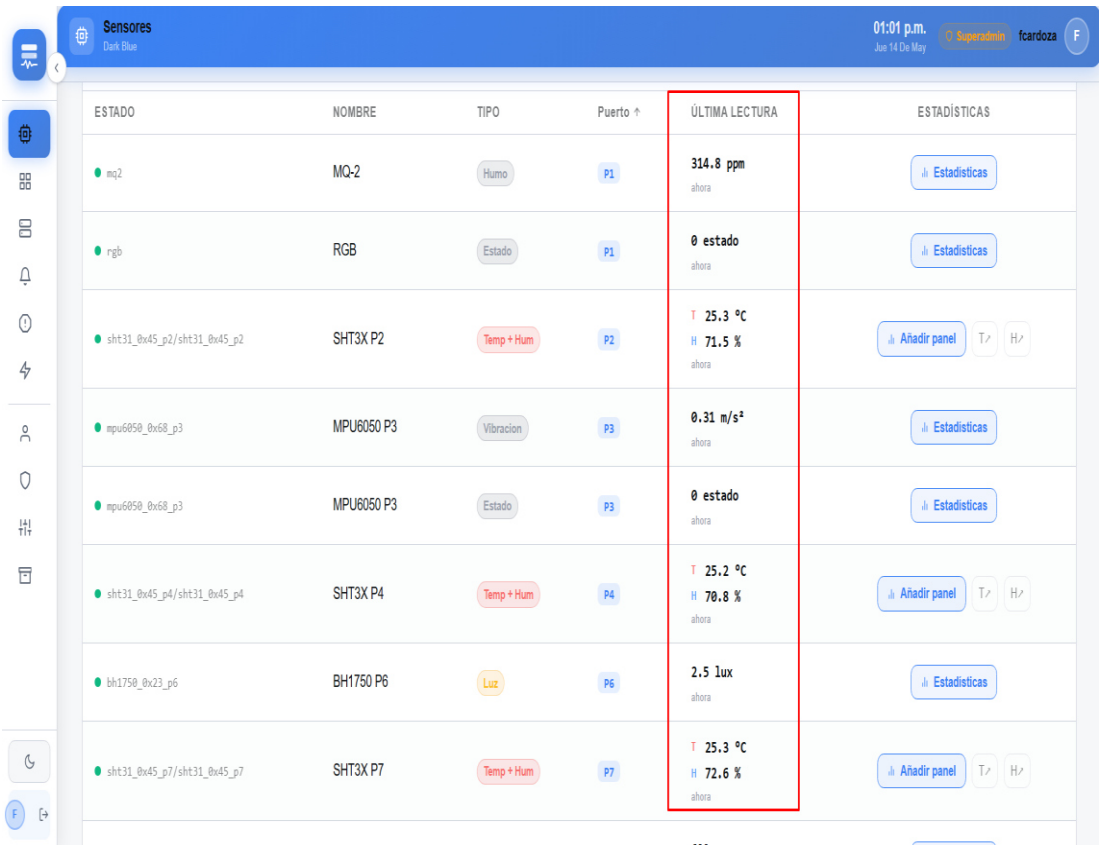


Figura 95 **Plataforma web en operación: visualización en tiempo real de variables del gabinete con todos los sensores activos**. Fuente. Elaboración propia.

La visualización en tiempo real opera mediante un mecanismo de consulta periódica *(polling)* al endpoint */datos_sensores* cada 3 segundos. La Figura 96 muestra el tablero de monitoreo (dashboard) principal con las variables ambientales del gabinete actualizándose en tiempo real, y la Tabla 52 consolida los tiempos de actualización medidos desde recepción del mensaje MQTT en el broker hasta respuesta completa del endpoint sobre 10 ciclos, donde el promedio de 209,6 ms garantiza que cada nueva lectura del ESP32 es visible en el navegador en menos de un segundo desde su publicación en el broker.



Figura 96 **Dashboard principal con visualización de dashboard de variables ambientales del gabinete en tiempo real.** Fuente. Elaboración propia

Tabla 52 **Tiempo de actualización del tablero de monitoreo (dashboard) (n = 10 mediciones)**

Estadístico	**Valor**
Promedio	209,6 ms
Mínimo	206,9 ms
Máximo	218,0 ms
Desviación estándar	4,1 ms

Fuente. Elaboración propia

El sistema de alertas opera mediante reglas configurables por umbral. Para evidenciar su funcionamiento, se configuró una regla con umbral de iluminancia sobre el sensor BH1750 (condición: luz > 10,0000 lux, gravedad CRITICAL) y se simuló su superación publicando un valor fuera de rango. La Figura 97 muestra la regla configurada en la interfaz y la Figura 98 presenta el correo de notificación recibido con los detalles del evento: regla, sensor, condición, valor medido y marca temporal

UTC. El promedio de 135,9 ms en los 10 eventos simulados confirma la entrega casi inmediata de la alerta desde el momento de detección del umbral utilizando el script de medición de tiempo de alertas desde su generación hasta la recepción del correo mostrado en el Anexo F y tal como se consolida en la Tabla 53.

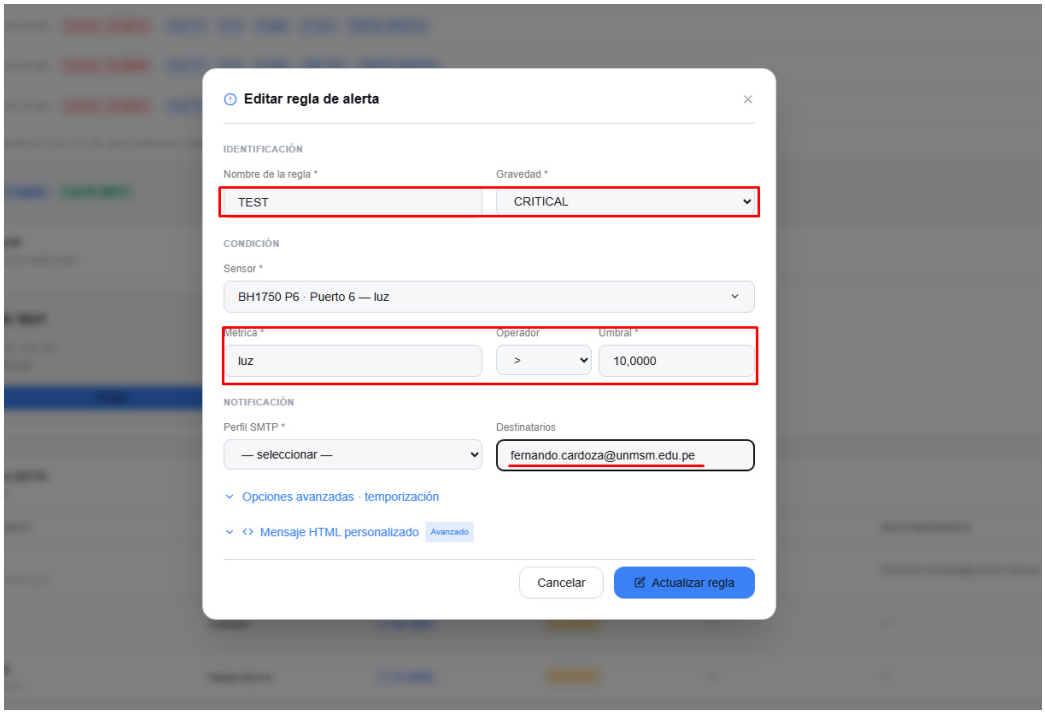


Figura 97 **Configuración de regla de alerta con umbral de iluminancia en la plataforma web.** Fuente. Elaboración propia**.**





Figura 98 **Correo de notificación recibido tras superación del umbral configurado, con detalle del evento**. Fuente. Elaboración propia.

Tabla 53 **Tiempo de notificación de alerta (n = 10 mediciones)**

Estadístico	**Valor**
Promedio	135,9 ms
Mínimo	132,7 ms
Máximo	146,5 ms
Desviación estándar	4,2 ms

Fuente. Elaboración propia

La Tabla 54 ilustra el formato de las mediciones individuales de ambos indicadores, generadas de forma consistente.

Tabla 54 **Muestra ilustrativa de 10 mediciones de tiempo de respuesta de la plataforma.**

Muestra	**Actualización del dashboard (ms)**	**Notificación de alerta (ms)**
1	206,9	132,7
2	209,0	143,3

3	207,9	140,0
4	218,0	138,0
5	207,7	140,4
6	209,2	132,7
7	206,9	138,5
8	213,5	141,9
9	207,2	141,7
10	218,0	146,5

Fuente. Elaboración propia. Muestra ilustrativa.

La Figura 99 presenta los tiempos promedio de respuesta de la plataforma para la actualización del dashboard y el envío de notificaciones de alerta, incluyendo su desviación estándar, con el propósito de evaluar el desempeño temporal de ambas funciones y verificar la rapidez de respuesta del sistema durante su operación.

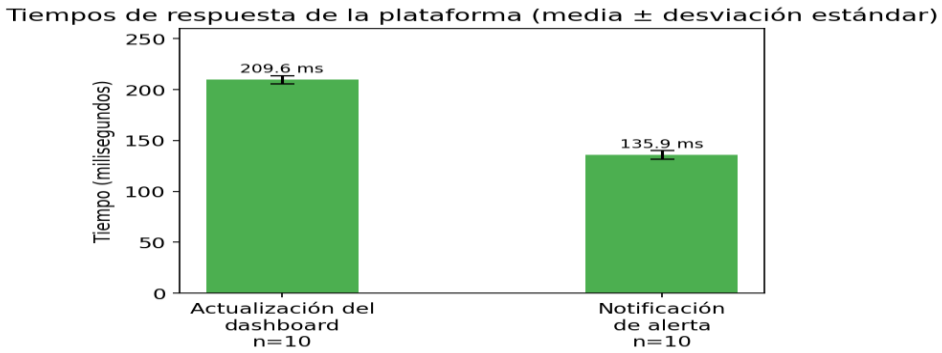


Figura 99 **Tiempos de respuesta de la plataforma: actualización del dashboard y notificación de alerta (media ± desviación estándar).** Fuente. Elaboración propia.

El control de acceso diferenciado por roles se evidencia en la Figura 100 y Figura 101, donde el perfil administrador accede a la totalidad de módulos del sistema, incluyendo gestión de usuarios, auditoría, configuración visual, backup y control de actuadores, mientras que el perfil visualizador accede únicamente a los módulos de consulta y monitoreo, sin opciones de configuración ni control disponibles en su menú lateral.

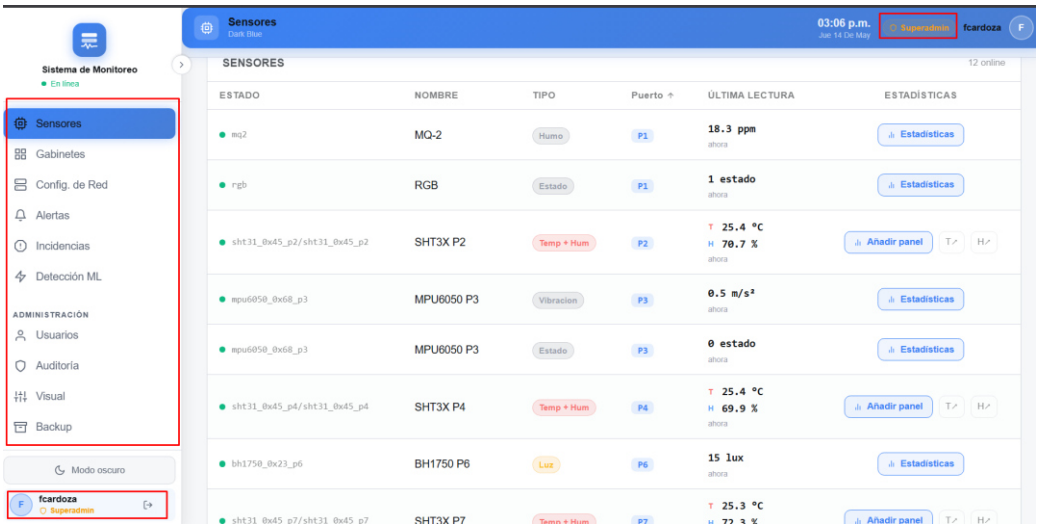


Figura 100 **Módulos disponibles para el perfil administrador**. Fuente. Elaboración propia.

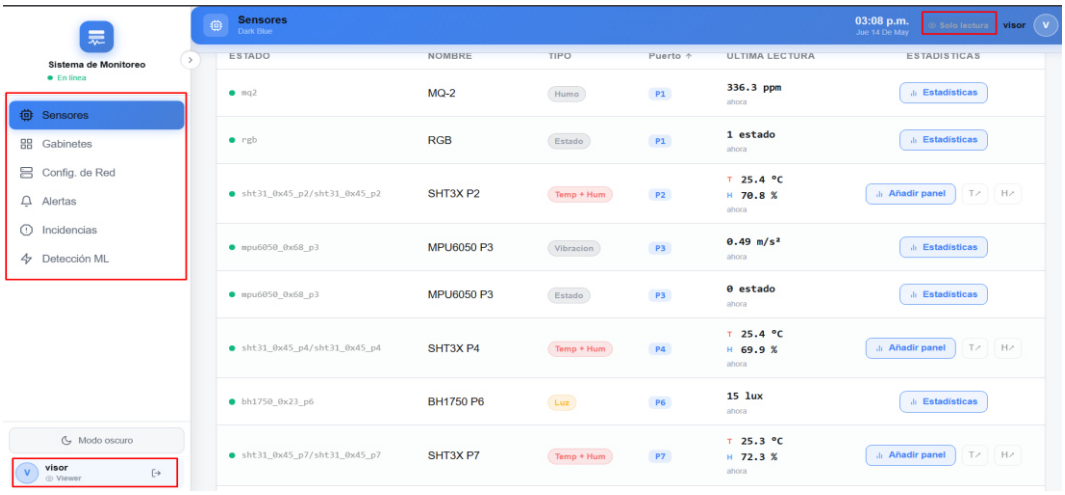


Figura 101 **Módulos disponibles para el perfil visualizador.** Fuente. Elaboración propia.

La plataforma web opera de forma funcional y continua: visualiza las variables del gabinete con un tiempo de actualización promedio de 209,6 ms, genera y despacha alertas por correo en 135,9 ms promedio ante superación de umbrales configurados, y controla el acceso mediante perfiles diferenciados que restringen las funciones disponibles según el rol del usuario autenticado.

V.1.6 Infraestructura Híbrida On-Premise y en la Nube

Como resultado de esta etapa se obtuvo una infraestructura híbrida operativa compuesta por un túnel Cloudflare que expone el backend Flask hacia Internet de forma segura, y un frontend ReactJS publicado globalmente mediante Azure Static Web Apps, ambos accesibles desde cualquier red externa bajo los dominios api.tesis-monitoring.xyz y web.tesis-monitoring.xyz respectivamente.

La Figura 102 muestra el estado del servicio cloudflared en el Gateway IoT verificado mediante systemctl, donde el estado active (running) con inicio automático configurado confirma que el túnel opera como servicio del sistema sin intervención manual. Para validar la hipótesis específica 6, se verificó el acceso remoto vía túnel Cloudflare, la disponibilidad del servicio y la continuidad de operación local ante desconexión de Internet, presentados a continuación.

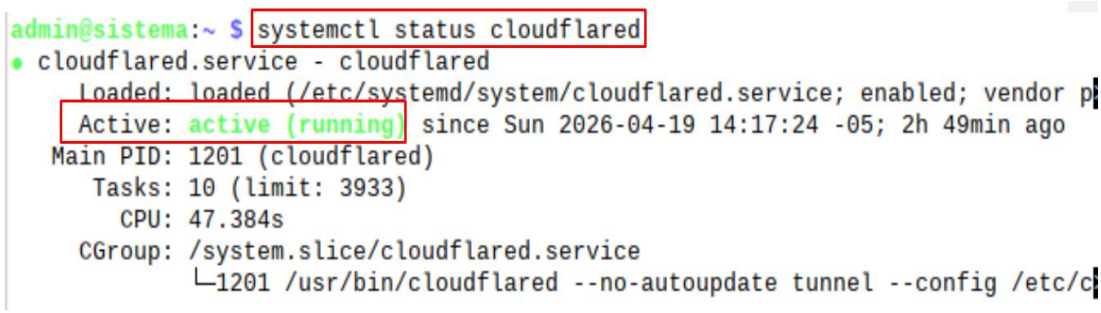


Figura 102 **Estado operativo del servicio cloudflared verificado mediante systemctl en el Gateway IoT.** Fuente. Elaboración propia.

Con el túnel activo, la Figura 103 confirma el acceso desde red externa al backend mediante solicitud HTTPS al dominio api.tesis-monitoring.xyz, donde la respuesta HTTP 200 con certificado SSL válido emitido automáticamente por Cloudflare evidencia que el backend es accesible de forma segura desde Internet con un tiempo de respuesta de 640 ms, correspondiente a la sobrecarga del túnel sobre la latencia base del endpoint.



Figura 103 **Acceso remoto al backend vía HTTPS a través del túnel Cloudflare verificado desde red externa.** Fuente. Elaboración propia.

El frontend fue desplegado en Azure Static Web Apps y su resolución DNS fue verificada mediante nslookup desde una red externa, confirmando que web.tesis-monitoring.xyz resolvía correctamente hacia la infraestructura de Azure con la dirección 20.22.31.128. Asimismo, la Figura 104 muestra el acceso a la interfaz de inicio de sesión mediante el dominio público y la conexión con el backend on-premise a través de api.tesis-monitoring.xyz, configurado en Cloudflare.



Figura 104 **Interfaz de inicio de sesión del sistema accesible desde red externa bajo el dominio público web.tesis-monitoring.xyz.** Fuente. Elaboración propia.

La disponibilidad de la infraestructura híbrida durante el período de evaluación se evidencia en la Figura 105, donde se presenta el panel de Cloudflare Zero Trust con el túnel tesis-backend en estado HEALTHY. Este estado confirma que el servicio de conectividad remota se mantuvo operativo durante la ventana de observación, registrando un tiempo de funcionamiento continuo de 6 horas sin interrupciones ni eventos de caída asociados al túnel. Asimismo, se observa que el conector permaneció en estado conectado, estableciendo comunicación activa desde los data centers lim y tpa, lo que demuestra la correcta operación del mecanismo de acceso seguro hacia el backend local.

En consecuencia, para la ventana analizada, el acceso remoto presentó una disponibilidad del 100 %, evidenciando que la arquitectura híbrida implementada mantiene continuidad operativa.

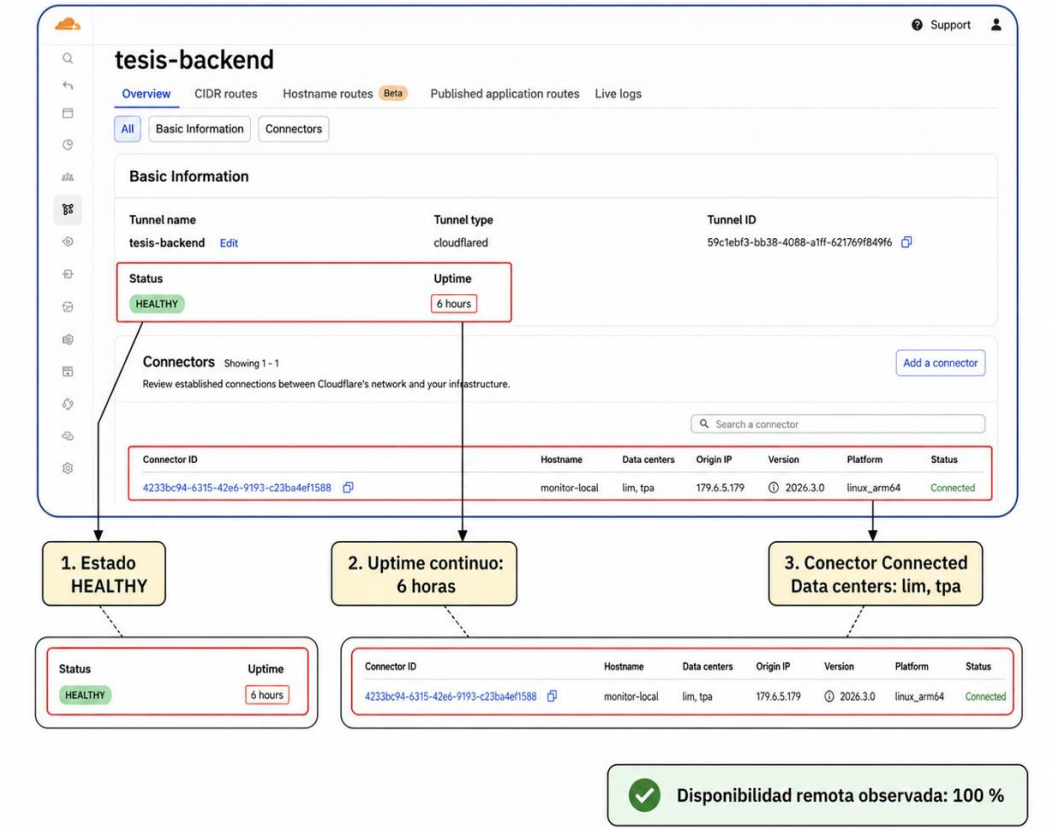


Figura 105 **Estado del túnel tesis-backend en el panel Cloudflare Zero Trust: estado HEALTHY y funcionamiento continuo.** Fuente. Elaboración propia.

La infraestructura híbrida opera de forma funcional y continua: el backend on-premise es accesible desde cualquier red externa mediante túnel Cloudflare con certificado SSL automático, el frontend está desplegado globalmente en Azure con resolución DNS verificada, y la disponibilidad del acceso remoto alcanzó el 100 % durante el período de evaluación, confirmando que el sistema es operable de forma remota sin depender de configuraciones de red específicas del centro de datos.

La Tabla 55 resume los eventos de la tabla auditoria en una sesión de prueba (23 de marzo de 2026, n = 15), todos desde IP de la LAN interna. El acceso externo vía túnel Cloudflare se verificó por separado en la Figura 103 (HTTP 200, SSL válido); al aplicar el backend el mismo mecanismo JWT en ambas rutas, la auditoría registra igual sin importar el origen de la sesión.

Tabla 55 **Eventos de auditoría registrados por tipo de acción (n = 15, sesión del 23 de marzo de 2026).**

Acción	Cantidad	Estado
LOGIN_SUCCESS	6	exitoso
LOGOUT	6	exitoso
DELETE_USER	3	advertencia

Fuente. Elaboración propia a partir de la tabla auditoria de monitoring_iot.

CAPÍTULO VI:
DISCUSIÓN

VI.1 Discusión de la Hipótesis General

La hipótesis general sostuvo que el sistema inteligente de monitoreo implementado alcanza un desempeño que satisface los indicadores de supervisión y gestión remota de gabinetes Tier 1, detectando tempranamente condiciones anómalas y garantizando operación continua ante fallos de conectividad externa. Los resultados del Capítulo V respaldan esta hipótesis. Las seis etapas del sistema operaron de forma integrada: el hardware de la UAC adquirió variables ambientales con estabilidad demostrada, el firmware las transmitió al Gateway mediante MQTT con latencia dentro del umbral establecido, el Gateway las persistió en MariaDB y ejecutó inferencia local con Isolation Forest, la plataforma las expuso en tiempo real con generación automática de alertas, y la infraestructura híbrida garantizó el acceso desde Internet sin comprometer la operación local. La supervisión remota se materializó mediante el acceso al tablero de monitoreo (dashboard) en web.tesis-monitoring.xyz con certificado SSL válido, y la gestión remota se verificó con la entrega de comandos al actuador RGB y la recepción de notificaciones automáticas por correo ante superación de umbrales. En conjunto, estas evidencias permiten aceptar la hipótesis general.

VI.2 Discusión de las Hipótesis Específicas

VI.2.1 Hardware de la Unidad de Adquisición y Control

La hipótesis específica 1 sostuvo que el hardware basado en ESP32, con sensores conectados mediante multiplexores I²C y pines GPIO, logra la adquisición de

variables ambientales en gabinetes Tier 1 con una tasa de éxito del 100 % en los ciclos de lectura y tensiones de alimentación dentro de ±4 % del valor nominal. Los resultados de la sección V.1.1 verifican cuantitativamente esta hipótesis: el bus I²C y los pines GPIO alcanzaron una tasa de éxito del 100 % sobre 1 200 ciclos por canal durante 60 minutos de operación continua, sin errores de NACK ni interferencias entre sensores, incluso con una extensión física de 2 m mediante cableado UTP CAT5e y conectores RJ45. Este comportamiento indica que la capacitancia acumulada se mantuvo dentro del límite de 400 pF establecido para I²C a 100 kHz y que el trenzado de las señales con su referencia de tierra contribuyó a reducir el ruido de modo común.

La estabilidad eléctrica también fue verificada bajo condiciones de carga baja, nominal y alta, manteniéndose los tres rieles de alimentación dentro del margen de ±4 %, con errores relativos máximos de 0,95 % para 12 V, 2,00 % para 5 V y 0,50 % para 3,3 V. Estos resultados evidencian que las etapas de regulación mantienen niveles adecuados de tensión ante las variaciones de carga evaluadas, permitiendo la operación de sensores y módulos de comunicación dentro de sus rangos de alimentación. Asimismo, la variación máxima de 0,4 °C entre los tres sensores SHT31 distribuidos verticalmente confirma la coherencia metrológica del arreglo y permite medir el gradiente térmico dentro del gabinete. En conjunto con el cumplimiento de la restricción de integración en un chasis rack 1U conforme a EIA-310, estos resultados permiten aceptar la hipótesis específica 1.

VI.2.2 Firmware de la Unidad de Adquisición y Control

La hipótesis específica 2 sostuvo que el firmware en el ESP32 que integrara adquisición de datos de sensores, control de actuadores y transmisión MQTT opera como nodo de adquisición y control con latencia de publicación inferior a 500 ms y reconexión automática en menos de 5 s ante pérdida de conectividad.

Los resultados de la sección V.1.2 respaldan esta hipótesis, obteniendo una latencia de publicación MQTT por debajo del umbral de 500 ms en la totalidad de los 30 ciclos evaluados, con una media de 409 ms y una tasa de publicaciones exitosas del 100 %. De esos 409 ms, 406 ms corresponden a la adquisición multisensorial y la serialización JSON, mientras que la operación mqtt.publish() en sí consume apenas 3

ms. Esto demuestra que el factor dominante del ciclo es la lectura de sensores y no la transmisión, lo cual es esperable dado que la comunicación ocurre en red local entre el ESP32 y el broker Mosquitto sin atravesar Internet; sin embargo, su cuantificación explícita es valiosa porque establece una línea base de rendimiento del nodo. Por otro lado, el mecanismo de reconexión automática restableció la operación en 2,2 s ante caída de WiFi y en 0,3 s ante pérdida del enlace MQTT, ambos dentro del umbral de 5 s. Estos tiempos son adecuados para el monitoreo de eventos de evolución lenta, como las variaciones térmicas, aunque podrían omitir eventos discretos de muy corta duración generados durante la ventana de reconexión, dado que el firmware actual no implementa un búfer de mensajes en memoria; esta observación delimita el alcance de la confiabilidad demostrada sin invalidar la hipótesis.

En contraste con Sultan (2025), quien validó un sistema IoT basado en MQTT con generación automática de alertas en operación continua reportando latencia y confiabilidad de transmisión, pero sin cuantificar los tiempos de reconexión y recuperación ante interrupciones, el presente trabajo aporta esa evidencia medible, lo que constituye una contribución adicional en términos de caracterización del desempeño. Por lo expuesto, se acepta la hipótesis específica 2.

VI.2.3 Modelo de Detección de Anomalías mediante Isolation Forest

La hipótesis específica 3 sostuvo que el modelo Isolation Forest en la Raspberry Pi identifica patrones anómalos en las condiciones del gabinete, habilitando la detección temprana de fallas con recall > 95 %, F1 > 90 % y tiempo de inferencia < 5 s.

Los resultados de la sección V.1.3 verifican esta hipótesis: El modelo Isolation Forest, de naturaleza no supervisada, se entrenó sin etiquetas a partir de los datos de operación normal del gabinete. Dado que un entorno Tier I en operación normal no genera registros etiquetados de fallas previas, su desempeño se validó frente a un conjunto de prueba construido mediante inyección controlada de anomalías: 600 muestras normales reales del gabinete y 600 anomalías sintéticas que representan cinco escenarios operacionales de riesgo real. Este esquema de entrenamiento no supervisado y validación contra un conjunto de prueba con muestras sintéticas de

forma ad hoc, permitió calcular métricas cuantitativas sin alterar la naturaleza no supervisada del algoritmo. Esta decisión metodológica es coherente con la literatura revisada: Rafique et al. (2024), en su revisión sistemática de técnicas de aprendizaje automático para detección de anomalías en redes IoT, concluyen que los métodos no supervisados resultan más adecuados precisamente en entornos donde no se dispone de etiquetas de falla verificadas, condición que describe el contexto de despliegue abordado.

Las métricas obtenidas fueron precisión de 95,67 %, recall de 99,33 %, F1-score de 97,47 % y exactitud de 97,42 %, con umbral de clasificación fijado en score $< -0{,}5028$. El indicador de mayor relevancia operacional es el recall: solo 4 de las 600 anomalías no fueron detectadas, todas correspondientes al escenario de vibración anómala, resultado consistente con el carácter binario de la característica mpu6050 en la implementación actual del firmware, que reduce la separabilidad de esa clase. Los otros cuatro escenarios con los valores sensados de mayor impacto operacional alcanzaron una tasa de detección del 100 %. La separabilidad estadística refuerza esta interpretación: las observaciones normales ($\mu = 0{,}054$; $\sigma = 0{,}049$) y las anómalas ($\mu = -0{,}138$; $\sigma = 0{,}040$) conforman dos distribuciones diferenciadas, con una diferencia entre medias de $\Delta\mu = 0{,}192$ equivalente a más de cuatro desviaciones estándar combinadas, lo que confirma que el modelo aprendió la estructura estadística del comportamiento normal del gabinete y no clasificó por umbral arbitrario.

Finalmente, el tiempo de inferencia de 0,057 ms por muestra ofrece un margen superior al 99 % respecto al umbral del requisito RNF-03, confirmando que la latencia de inferencia no constituye un cuello de botella en el Gateway. La limitación principal de esta evaluación radica en el uso de anomalías sintéticas en lugar de eventos de falla reales registrados en producción, aspecto que se retoma como recomendación de investigación futura. Por lo expuesto, se acepta la hipótesis específica 3.

VI.2.4 Gateway IoT basado en Raspberry Pi

La hipótesis específica 4 sostuvo que el Gateway IoT centraliza la recepción, almacenamiento y sincronización con latencia < 500 ms, tasa de entrega del 100 % y servicios persistentes.

Los resultados de la sección V.1.4 sustentan esta hipótesis de manera completa. La latencia del broker Mosquitto registró promedios de 12 ms y 14 ms en los dos escenarios evaluados, y la sincronización MQTT-MariaDB presentó una latencia promedio de 13,3 ms con tasa de entrega del 100% en las 20 mediciones. Ambos valores están muy por debajo del umbral de 500 ms esperado en las hipótesis planteadas, lo que indica que el Gateway no introduce cuellos de botella en el flujo de datos bajo la carga de un único gabinete. Un aspecto arquitectónico relevante es la decisión de diferenciar la red de adquisición de la red corporativa mediante una interfaz virtual dedicada (uap0) con punto de acceso WPA2 independiente. Esta separación implica que el tráfico generado por la UAC no atraviesa la LAN del entorno, reduciendo la superficie de ataque y eliminando la dependencia de la infraestructura de red del cliente para el funcionamiento del sistema de monitoreo, decisión que tiene implicancias directas en la seguridad operativa del despliegue y que refuerza el carácter autónomo del Gateway frente a la red institucional. El registro de todos los servicios (hostapd, dnsmasq, Mosquitto) como unidades systemd con política de reinicio automático garantiza la disponibilidad del nodo ante reinicios no planificados. Conviene señalar que la Raspberry Pi 4 Model B resultó suficiente para el contexto de un único gabinete; el escalamiento a múltiples gabinetes simultáneos podría requerir hardware con mayor capacidad de procesamiento o distribución de servicios entre nodos, aspecto que debe considerarse en futuras ampliaciones del sistema. Con esta acotación, la hipótesis específica 4 se acepta.

VI.2.5 Plataforma de Software para Monitoreo y Control

La hipótesis específica 5 sostuvo que la plataforma permite gestionar usuarios, visualizar en tiempo real y controlar remotamente el sistema de monitoreo, con actualización del panel (dashboard) < 5 s y alertas < 60 s.

Los resultados de la sección V.1.5 respaldan esta hipótesis. El tiempo de actualización del tablero de monitoreo (dashboard) registró una media de 209,6 ms (≈0,21 s) sobre diez mediciones, holgadamente dentro del umbral de 5 s, lo que garantiza que cada nueva lectura del ESP32 sea visible en el navegador en menos de un segundo desde su publicación en el broker. El tiempo de notificación de alerta promedió 135,9 ms (≈0,14 s) sobre diez eventos simulados, con la totalidad de las

alertas entregadas correctamente ante la superación de umbrales configurados, muy por debajo del umbral de 60 s. La diferenciación de roles operó conforme al principio de mínimo privilegio: el perfil administrador accedió a la totalidad de los módulos, mientras que el perfil visualizador quedó restringido a las funciones de consulta y monitoreo, sin opciones de configuración ni control en su menú.

Una observación pertinente, derivada de la operación del frontend mediante consulta periódica al endpoint correspondiente, es que el tiempo de respuesta de la autenticación es naturalmente superior al de las consultas de datos, debido al costo computacional del algoritmo de encriptación diseñado en el backend; este sobrecosto es inherente al diseño de seguridad y no representa una limitación operativa para el uso normal del sistema. Asimismo, las pruebas se realizaron con un único cliente en red LAN, por lo que el comportamiento bajo carga concurrente de múltiples usuarios simultáneos constituye una variable no evaluada en este estudio. Con esta acotación, se acepta la hipótesis específica 5.

VI.2.6 Infraestructura Híbrida On-Premise y en la Nube

La hipótesis específica 6 sostuvo que la infraestructura híbrida permite accesibilidad remota con certificado SSL válido. Los resultados de la sección V.1.6 no solo confirman esta condición, sino que evidencian además una disponibilidad del acceso remoto del 100 % y continuidad de la operación local ante desconexión de Internet, evidencia adicional que refuerza la solidez de la arquitectura híbrida propuesta más allá de lo estrictamente planteado en la hipótesis.

Los resultados de la sección V.1.6 validan esta hipótesis: el backend resultó accesible desde red externa mediante túnel Cloudflare con HTTP 200, certificado SSL válido y 640 ms de latencia; el frontend en Azure Static Web Apps resolvió su DNS público y quedó accesible bajo el dominio establecido; y la disponibilidad del acceso remoto alcanzó el 100 % durante el período observado, con el túnel en estado HEALTHY. El uso de un túnel saliente cifrado, en lugar de redirección de puertos o VPN con IP pública, evita exponer servicios internos del Gateway a Internet, y la prueba de continuidad confirmó que las funciones críticas permanecen operativas ante

la desconexión de Internet, independizando la operación local de la disponibilidad de terceros (Cloudflare, Azure). Con esta evidencia, se acepta la hipótesis específica 6.

VI.3 Implicancias Técnicas y Económicas

VI.3.1 Implicancias técnicas

El estudio demuestra la viabilidad de integrar sensado y control, mensajería MQTT, aprendizaje automático no supervisado y despliegue híbrido nube-local en una solución unificada para gabinetes Tier 1 construida con componentes de bajo costo y software libre. Tres aportes técnicos merecen destacarse. Primero, la combinación de detección por umbrales predeterminados en conjunto con la detección estadística mediante Isolation Forest ofrece una cobertura complementaria: la primera asegura respuesta inmediata ante condiciones críticas conocidas, mientras que la segunda identifica desviaciones contextuales no anticipadas que un sistema de reglas fijas no podría capturar. Segundo, la ejecución de la inferencia en el borde, sobre la Raspberry Pi, elimina la dependencia de la nube para el análisis crítico, reduce la latencia de detección y mantiene la confidencialidad de los datos operativos del gabinete. Tercero, el patrón de túnel saliente cifrado para el acceso remoto evita la exposición de la red local a Internet sin requerir IP pública ni configuración de firewall en la infraestructura del cliente.

VI.3.2 Implicancias económicas

El sistema se posiciona como una alternativa de bajo costo frente a las soluciones comerciales de gestión de infraestructura de centros de datos, cuya adopción en organizaciones con infraestructura Tier 1 limitada suele resultar inviable por el costo de licenciamiento, el hardware propietario y el soporte especializado que estas exigen. El costo referencial del hardware implementado se mantuvo en el orden de los S/ 1033, cifra considerablemente menor frente a la inversión que implicarían soluciones comerciales equivalentes, y el uso de planes gratuitos o de bajo consumo

como Cloudflare Tunnel, Azure Static Web Apps en nivel inicial y GitHub Actions con cuota gratuita, mantiene reducidos los costos recurrentes durante la operación inicial, sin perjuicio de migrar a planes pagados al escalar conforme crezcan las necesidades del despliegue. La capacidad del sistema para generar alertas automáticas sin supervisión humana constante reduce los costos operativos asociados a la inspección manual y a la respuesta tardía ante incidentes ambientales, cuyo impacto potencial como pérdida de servicios, daño a equipos por sobrecalentamiento y accesos no autorizados supera ampliamente la inversión inicial del prototipo. La continuidad operativa local ante fallos de conectividad protege adicionalmente esa inversión, dado que el sistema no queda inoperante durante interrupciones del enlace externo, preservando así la capacidad de monitoreo incluso en escenarios adversos donde otras soluciones dependientes de la nube podrían verse comprometidas.

VI.4 Fortalezas y limitaciones

Entre las fortalezas del estudio destacan la validación sobre un gabinete en producción real durante siete días consecutivos; la cobertura simultánea de cinco variables físicas del gabinete con un intervalo de muestreo de 3 s; la operación continua verificada ante la desconexión de Internet; y la modularidad del diseño arquitectónico, que permite incorporar nuevos sensores o gabinetes sin rediseño estructural.

Entre las limitaciones principales se identifican: la evaluación del modelo Isolation Forest mediante anomalías sintéticas y no mediante eventos de falla reales registrados en producción, por la ausencia de un conjunto de datos etiquetado de fallas; la ausencia de un búfer de mensajes en el firmware, que ocasiona la pérdida de lecturas generadas durante las ventanas de reconexión prolongadas; la no evaluación del comportamiento de la plataforma bajo carga concurrente de múltiples usuarios; la dependencia de servicios de terceros para el acceso remoto; y la validación sobre un único nodo de adquisición y un único gabinete, lo que limita la generalización de los resultados a despliegues con múltiples gabinetes simultáneos. Estas limitaciones restringen el alcance de las conclusiones sin comprometer la validez de los resultados dentro del contexto evaluado, y orientan directamente las recomendaciones de trabajo futuro.

CAPÍTULO VII:
CONCLUSIONES

VII.1 Conclusión General

El sistema inteligente de monitoreo integró sensado distribuido, comunicación MQTT, detección autónoma de anomalías y una arquitectura híbrida nube-local en una solución unificada de bajo costo y software libre, validada sobre un gabinete en operación real, que habilita la supervisión y la gestión remota de gabinetes en centros de datos con características Tier 1.

VII.2 Conclusiones Específicas

El hardware de la Unidad de Adquisición y Control basado en ESP32 registró las variables ambientales y estructurales del gabinete mediante lectura periódica por bus I²C y GPIO, con una tasa de éxito del 100 % sobre 1 200 ciclos por canal y tensiones de alimentación dentro del margen de ±4 % verificadas con multímetro digital bajo carga nominal, integrado en un chasis rack 1U conforme a la norma EIA-310.

El firmware del ESP32 operó de forma estable como nodo de adquisición y control, con una latencia media de publicación MQTT de 409 ms y reconexión automática en menos de 5 s ante pérdida de enlace, sin interrupción de la transmisión de datos.

El modelo Isolation Forest desplegado en la Raspberry Pi identificó de forma autónoma las condiciones anómalas del gabinete, alcanzando un recall de 99,33 % y un F1-score de 97,47 %, con un tiempo de inferencia de 0,057 ms por muestra que confirma su viabilidad para operación en el borde.

El Gateway IoT basado en Raspberry Pi 4B centralizó la recepción, el almacenamiento y la sincronización de la información con una latencia de procesamiento y sincronización inferior a 15 ms y una tasa de entrega del 100 %, manteniendo sus servicios en operación persistente sobre una red de adquisición segregada.

La plataforma de software basada en Flask, MariaDB y ReactJS habilitó la supervisión y el control remoto del gabinete, con un tiempo de actualización del tablero de monitoreo (dashboard) de 0,21 s y de notificación de alertas de 0,14 s, y un control de acceso por roles conforme al principio de mínimo privilegio.

La infraestructura híbrida garantizó el acceso remoto seguro con certificado SSL válido preservando la operación local completa ante la desconexión de Internet y el despliegue continuo automatizado del frontend.

CAPÍTULO VIII:
RECOMENDACIONES

En relación con el hardware, se recomienda extender la validación a múltiples unidades y a periodos de operación continua más prolongados, con el fin de confirmar la estabilidad del bus I²C y de las etapas de regulación de tensión bajo ciclos térmicos reales del centro de datos.

En relación con el firmware, se recomienda implementar un búfer circular de mensajes en memoria dentro del firmware del ESP32, con el fin de preservar las lecturas generadas durante las ventanas de reconexión.

En relación con el modelo de detección de anomalías, conviene asimismo revisar la representación de la característica de vibración (mpu6050), actualmente binaria, hacia una magnitud continua que mejore la separabilidad de ese escenario, el único que no alcanzó el 100 % de detección.

En relación con el Gateway IoT, se recomienda evaluar mecanismos de balanceo de carga y distribución de servicios entre nodos para el escenario de múltiples gabinetes, así como documentar los límites operacionales del nodo actual antes de planificar cualquier despliegue a escala.

En relación con la plataforma de software, se recomienda realizar pruebas de carga concurrente con múltiples usuarios simultáneos, a fin de caracterizar el comportamiento del backend Flask bajo condiciones de uso intensivo e identificar los cuellos de botella que requerirían optimización o la migración a un servidor de aplicaciones de mayor capacidad.

En relación con la infraestructura híbrida, se recomienda evaluar alternativas a los planes gratuitos de Cloudflare y Azure Static Web Apps para entornos de producción que requieran acuerdos de nivel de servicio formales, dado que dichos planes no garantizan disponibilidad contractual.

CAPÍTULO IX:
BIBLIOGRAFÍA

Aguirre, M., Vergara-Lozano, V., Lagos-Ortiz, K., & El, A. (2022). *Evaluation of User Interface (UI) and User Experience (UX) for Web Services of a Weather Data Monitoring Platform* (pp. 235–246). https://doi.org/10.1007/978-3-031-19961-5_17

Alatoom, Y. I., & Smadi, O. (2025). Embedded framework for low-cost pavement condition evaluation using microcontroller and single-board computer platforms. *Automation in Construction*, *178*, 106442. https://doi.org/10.1016/j.autcon.2025.106442

Andriulo, F. C., Fiore, M., Mongiello, M., Traversa, E., & Zizzo, V. (2024). Edge computing and cloud computing for Internet of Things: A review. *Informatics*, *11*(4), 71. https://doi.org/10.3390/informatics11040071

ASHRAE Technical Committee 9.9. (2021). *Thermal Guidelines for Data Processing Environments* (5th ed.). ASHRAE.

Avasalcai, C., Murturi, I., & Dustdar, S. (2020). Edge and fog: A survey, use cases, and future challenges. In *Fog Computing: Theory and Practice* (pp. 43–65). John Wiley & Sons. https://doi.org/10.1002/9781119551713.ch2

Cashmore, S. (2026). *Beginning Azure Static Web Apps: Building and Deploying Professional, Dynamic Web Applications* (2nd ed.). Apress.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 15. https://doi.org/10.1145/1541880.1541882

Chatterjee, A., & Ahmed, B. S. (2022). IoT anomaly detection methods and applications: A survey. *Internet of Things*, *19*, 100568. https://doi.org/10.1016/j.iot.2022.100568

Choudhary, A. (2024). Internet of Things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions. *Discover Internet of Things*, *4*(31). https://doi.org/10.1007/s43926-024-00084-3

Cook, A. A., Misirli, G., & Fan, Z. (2020). Anomaly detection for IoT time-series data: A survey. *IEEE Internet of Things Journal*, *7*(7), 6481–6494. https://doi.org/10.1109/JIOT.2019.2958185

Dulanto, S. (2025). *Diseño de un sistema IOT basado en la computación de borde para el monitoreo de la calidad del aire en el distrito de Chancay*.

Elnaggar, M. (2025). Enhanced IoT-based environmental monitoring system using MQTT with real-time alerts and predictive analytics. *IAR Journal of Engineering and Technology*, *6*(1), 1–5. https://iarconsortium.org/iarjet/191/2887/enhanced-iot-based-environmental-monitoring-system-using-mqtt-with-real-time-alerts-and-predictive-analytics-4950/

Espinel, R. I., Ardila, E., Zarate, H., & Ortiz, J. E. (2022). Diseño e implementación de un sistema de monitoreo de red para infraestructura de campus usando agentes de software. *Ingeniería e Investigación*, *42*(1), e87564. https://doi.org/10.15446/ing.investig.v42n1.87564

Espressif Systems. (2025). *ESP32-WROOM-32 Datasheet* (Version 3.6). https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

Etingof, I., Inc., L., & contributors, P. (2026). *PySNMP* (Version 7.1.27). Python Package Index. https://pypi.org/project/pysnmp/

Everlight Electronics. (2023). *Luminosity Full Color LED 5050ST/GRBC33-PU5/1T*. *5*.

Faqih, A. R., Taufiqurrahman, A., Husen, J. H., & Sabariah, M. K. (2024). Empirical analysis of CI/CD tools usage in GitHub Actions workflows. *Journal of Informatics and Web Engineering*, *3*(2), 251–261. https://doi.org/10.33093/jiwe.2024.3.2.18

Garcia, J. (2023). *Diseño e implementación de una plataforma IoT para el control y monitoreo remoto del banco de psicrometría del Laboratorio de Energía de la PUCP*. http://hdl.handle.net/20.500.12404/25760

Geron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (3rd ed.). O'Reilly Media.

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media.

Hernandez-Sampieri, R., & Mendoza Torres, C. P. (2018). *Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta*. McGraw-Hill Education.

IEEE. (2020). *IEEE 2413-2019: IEEE Standard for an Architectural Framework for the Internet of Things (IoT)*. IEEE. https://ieeexplore.ieee.org/document/9032420

IEEE. (2021). *IEEE 802.11-2020: IEEE Standard for Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE. https://ieeexplore.ieee.org/document/9363693

Intel Corporation. (2005). *Server Rack Cabinet Compatibility Guide*. https://cdrdv2-public.intel.com/840778/server_rack_cabinet_compatibility_guide_24.pdf

InvenSense Inc. (2013). *MPU-6000 and MPU-6050 Product Specification Revision 3.4*. InvenSense Inc. https://www.cdiweb.com/datasheets/invensense/mpu-6050_datasheet_v3%204.pdf

Jiang, K., Kheradmandi, M., Hu, C., Pal, S., & Yan, F. (2021). Data-driven fault tolerant predictive control for temperature regulation in data center with rack-based cooling architecture. *Mechatronics*, *79*, 102633. https://doi.org/10.1016/j.mechatronics.2021.102633

Kelley, S. (2025). *Dnsmasq: Network services for small networks* (Version 2.91). https://thekelleys.org.uk/dnsmasq/doc.html

Light, R. (2024). *Eclipse Paho MQTT Python client (paho-mqtt)* (Version 2.1.0). Python Package Index. https://pypi.org/project/paho-mqtt/

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, 413–422. https://doi.org/10.1109/ICDM.2008.17

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2012). Isolation-Based Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, *6*(1), 1–39. https://doi.org/10.1145/2133360.2133363

Lu, Z., Di, Y., Di, H., Li, X., Chen, Z., Zhao, C., & Gao, Y. (2025). Optimization of data center thermal management performance under failure mode of rack-level heat pipe backplane system. *Building Simulation*, *18*, 1377–1392. https://doi.org/10.1007/s12273-025-1271-5

Malinen, J. (2024). *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/WPA3/EAP/RADIUS authenticator* (Version 2.11). https://w1.fi/hostapd/

Mohanan, V. (2022). *DOIT ESP32 DEVKIT V1 Pinout r0.1 - CIRCUITSTATE*. https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/https://www.circuitstate.com/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/

National Institute of Standards and Technology. (2023). A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments. In *NIST Special Publication 800-207A*. U.S. Department of Commerce. https://csrc.nist.gov/pubs/sp/800/207/a/final

NXP Semiconductors. (2021). *UM10204: I2C-bus specification and user manual* (Rev. 7.0). https://www.nxp.com/docs/en/user-guide/UM10204.pdf

Oracle Corporation. (2026). *MySQL 8.0 Reference Manual.* https://dev.mysql.com/doc/refman/8.0/en/

Potharaju, S., Tirandasu, R. K., Tambe, S. N., Jadhav, D. B., Kumar, D. A., & Amiripalli, S. S. (2025). A two-step machine learning approach for predictive maintenance and anomaly detection in environmental sensor systems. *MethodsX*, *14*, 103181. https://doi.org/10.1016/j.mex.2025.103181

Radwan, N. M., & Alves-Foss, J. (2024). MQTT in focus: Understanding the protocol and its recent advancements. *International Journal of Computer Science and Security*. https://www.cscjournals.org/library/manuscriptinfo.php?mc=IJCSS-1710

Rafique, S. H., Abdallah, A., Musa, N. S., & Murugan, T. (2024). Machine learning and deep learning techniques for Internet of Things network anomaly detection

— Current research trends. *Sensors*, *24*(6), 1968. https://doi.org/10.3390/s24061968

Raspberry Pi Ltd. (2024). *Raspberry Pi 4 Model B datasheet*. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

Reis, M. J. C. S. (2025). Lightweight signal processing and edge AI for real-time anomaly detection in IoT sensor networks. *Sensors*, *25*(21), 6629. https://doi.org/10.3390/s25216629

ROHM Co., Ltd. (2011). *Digital 16bit Serial Output Type Ambient Light Sensor IC BH1750FVI.*

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero Trust Architecture* (Number NIST Special Publication 800-207). https://doi.org/10.6028/NIST.SP.800-207

Santana Roldan, C. (2023). *React 18 Design Patterns and Best Practices: Design, build, and deploy production-ready web applications with React by leveraging industry-best practices* (4th ed.). Packt Publishing.

Santillán, R. (2026). *Diseño e implementación de un invernadero inteligente usando IOT para la producción continua y sostenible de cebolla china en la localidad de Taquia en el departamento de Amazonas*.

Sensirion, A. G. (2022). *Datasheet SHT3x-DIS: Humidity and Temperature Sensor* (Version 7). https://sensirion.com/media/documents/213E6A3B/63A5A569/Datasheet_SHT3x_DIS.pdf

Standex Detect. (2025). *Reed Switches & Sensors*. https://standexdetect.com/wp-content/uploads/sites/2/2025/10/brochure-product-line-reed-switches-sensors.pdf

STMicroelectronics. (2024). *VL53L0X: World's Smallest Time-of-Flight (ToF) Ranging Sensor*. https://www.st.com/resource/en/datasheet/vl53l0x.pdf

Sultan, M. (2025). Enhanced IoT-based environmental monitoring system using MQTT with real-time alerts and predictive analytics. *IAR Journal of Engineering and Technology*, *6*(1), 1–5. https://iarconsortium.org/iarjet/191/2887/enhanced-iot-based-environmental-monitoring-system-using-mqtt-with-real-time-alerts-and-predictive-analytics-4950

Telecommunications Industry Association. (2024). *ANSI/TIA-942-C: Telecommunications Infrastructure Standard for Data Centers*. TIA.

Texas Instruments. (2012). *TCA9548A Low-Voltage 8-Channel I2C Switch with Reset* (Rev. H). https://www.ti.com/lit/ds/symlink/tca9548a.pdf

Texas Instruments. (2023). *LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator*. Texas Instruments Incorporated. https://www.ti.com/lit/ds/symlink/lm2596.pdf

Uptime Institute. (2022). *Tier Standard: Topology*. https://uptimeinstitute.com/resources/asset/tier-standard-topology

Witczak, D., & Szymoniak, S. (2024). Review of monitoring and control systems based on Internet of Things. *Applied Sciences*, *14*(19), 8943. https://doi.org/10.3390/app14198943

Xu, H., Pang, G., Wang, Y., & Wang, Y. (2023). Deep Isolation Forest for Anomaly Detection. *IEEE Transactions on Knowledge and Data Engineering*, *35*(12), 12591–12604. https://doi.org/10.1109/TKDE.2023.3270293

Zevallos, J. (2024). *Modelo de monitoreo de calidad de aire interior basado en arquitectura IOT*. https://hdl.handle.net/20.500.12724/20543

Zhukabayeva, T., Ahmad, Z., Adamova, A., Karabayev, N., & Abdildayeva, A. (2025). An edge-computing-based integrated framework for network traffic analysis and intrusion detection to enhance cyber-physical system security in Industrial IoT. *Sensors*, *25*(8), 2395. https://doi.org/10.3390/s25082395

CAPÍTULO X:
ANEXOS

Anexo A

Tabla A1 **Matriz de c***onsistencia*

Problema	Objetivos	Hipótesis	Variables / Dimensiones	Metodología
General ¿Cómo diseñar e implementar un sistema inteligente de monitoreo, basado en sensado y comunicación distribuida, un Gateway IoT y aprendizaje automático sobre una arquitectura híbrida (nube–local), que permita la supervisión y gestión remota de gabinetes en centros de datos Tier 1?	**General.** Diseñar e implementar dicho sistema inteligente de monitoreo para la supervisión y gestión remota de gabinetes en centros de datos Tier 1.	**General.** El sistema implementado alcanza un desempeño que satisface los indicadores técnicos de supervisión y gestión remota, evidenciando detección temprana de fallas y operación continua.	**VI:** Sistema Inteligente de Monitoreo. **VD:** Supervisión y gestión remota de gabinetes Tier 1.	**Tipo:** aplicada. **Nivel:** descriptivo. **Enfoque:** cuantitativo. **Diseño:** no experimental. **Población:** gabinetes Tier I (TIA-942). **Muestra:** gabinete de Cloud IT Perú S.A.C. **Análisis:** estadística descriptiva.
PE1. ¿Cómo diseñar e implementar el hardware de la UAC basada en ESP32, que integre la adquisición de variables ambientales y	**OE1.** Diseñar e implementar el hardware de la UAC basada en ESP32 con adquisición de variables ambientales	**HE1. El hardware (ESP32, multiplexores I²C, pines GPIO) logra la adquisición de variables ambientales:**	Hardware UAC	(común a todas las dimensiones)

el control de iluminación RGB, garantizando operación precisa y estable?	y control de iluminación RGB, con operación precisa y estable.	**tasa de éxito del 100 % en los ciclos de lectura y tensión de alimentación dentro de ±4 % del valor nominal.**		
PE2. ¿Cómo desarrollar el firmware de la UAC (ESP32), integrando adquisición, control de actuadores y transmisión MQTT, validando la confiabilidad de la comunicación en operación continua?	**OE2.** Desarrollar el firmware de la UAC (ESP32) integrando adquisición, control de actuadores y transmisión MQTT, validando la confiabilidad de la comunicación.	**HE2. El firmware opera como nodo de adquisición, control y transmisión MQTT, con latencia de publicación menor a 500 ms y reconexión automática menor a 5 s.**	Firmware	(común a todas las dimensiones)
PE3. ¿Cómo incorporar un modelo Isolation Forest en una Raspberry Pi para el análisis local de condiciones ambientales y estructurales, capaz de identificar patrones anómalos?	**OE3.** Incorporar un modelo Isolation Forest en Raspberry Pi para el análisis local de condiciones ambientales y estructurales, capaz de identificar patrones anómalos.	**HE3. El modelo Isolation Forest identifica anomalías con recall superior al 95 %, F1-score superior al 90 % y tiempo de inferencia menor a 5 s.**	Modelo ML	(común a todas las dimensiones)
PE4. ¿Cómo diseñar e integrar un Gateway IoT basado en Raspberry Pi como nodo	**OE4.** Diseñar e implementar un Gateway IoT (Raspberry Pi) como	**HE4. El Gateway centraliza recepción, almacenamiento y sincronizaci**	Gateway IoT	(común a todas las dimensiones)

central, con red inalámbrica dedicada, almacenamiento y procesamiento local y sincronización a la plataforma?	nodo central: red inalámbrica dedicada, almacenamiento y procesamiento local y sincronización a la plataforma.	**ón, con latencia de broker y de base de datos menor a 500 ms, tasa de entrega del 100 % y servicios persistentes.**		
PE5. ¿Cómo desarrollar una plataforma de software para la gestión, monitoreo, control y visualización de variables, validando endpoints, flujo MQTT y tiempo de actualización?	**OE5.** Desarrollar una plataforma de gestión, monitoreo, control y visualización de variables, validando endpoints, flujo MQTT y tiempo de actualización.	**HE5. La plataforma de software actualiza el panel de monitoreo en menos de 5 s y entrega alertas en menos de 60 s.**	Plataforma de software	(común a todas las dimensiones)
PE6. ¿Cómo desplegar una infraestructura híbrida on-premise y nube (Azure y Cloudflare) para la administración remota y la operación continua con conectividad limitada?	**OE6.** Desplegar una infraestructura híbrida on-premise y nube (Azure y Cloudflare) para administración remota y operación continua con conectividad limitada.	**HE6. La infraestructura híbrida (on-premise, Azure, Cloudflare Tunnel) permite accesibilidad remota con certificado SSL válido.**	Infraestructura híbrida	(común a todas las dimensiones)

Fuente. Elaboración propia.

Anexo B

Tabla B1 **Lista de materiales y costo referencial del sistema implementado**

Componente		#	Precio unitario (S/)	Subtotal (S/)	Observación
Microcontrolador	ESP32-WROOM-32 / DevKit V1	1	30	30	Nodo principal de la UAC
Multiplexor I²C	TCA9548A	1	10	10	Expansión del bus I²C
Sensor temperatura/humedad	SHT31	3	30	90	Sensado distribuido
Sensor iluminancia	BH1750	1	7	7	Detección de luz
Sensor distancia	VL53L0X	1	16	16	Presencia/ distancia
Sensor inercial	MPU6050	1	14	14	Vibración/ movimiento
Sensor magnético	Reed switch	1	5	5	Estado de puerta
MOSFET canal N	IRLZ44N o equivalente lógico	3	3	9	Control RGB
Tira LED RGB	SMD5050 12 V	1	50	50	Señalización visual
Fuente AC/DC	220 VAC a 12 VDC	1	61	61	Alimentación principal
Convertidor DC/DC	Buck 12 V a 5 V	1	16	16	Alimentación auxiliar

Convertidor DC/DC	Buck 12 V a 3.3 V	1	16	16	Alimentación lógica
PCB	Placa fabricada	1	55	55	Integración electrónica
Conectores	RJ45 / borneras / headers	Varios	2	20	Interconexión
Cableado y accesorios	Cables, resistencias, terminales	Varios	2	50	Montaje
Chasis metálico 1RU	Corte y doblado metálico	1	120	120	Chasis para instalación física
Gateway IoT	Raspberry Pi 4B	1	450	450	Raspberry Pi + fuente de poder + memoria SD + chasis de plástico
Diseño de chasis metálico 1RU	Diseño para corte laser metálico	1	0	0	Elaboración propia
Diseño PCB	Diseño utilizado para la impresión de tarjeta PCB	1	0	0	Elaboración propia
Ventiladores para chasis metálico 1RU	Sistema de ventilación del chasis	2	5	10	Elementos de ventilación interna del chasis
Registro anual de dominio	web.tesis-monitoring.xyz	1	20	20	Dominio para el acceso público a

		la plataforma
TOTAL	1033	

Fuente. Los precios son referenciales y corresponden a valores de adquisición disponibles durante el periodo de desarrollo de la investigación.

Tabla B2 **Mediciones de tensión de los rieles del PCB en función del tiempo y condición de carga**

Riel	Condición	10 min	20 min	30 min	40 min	50 min
12,0 V	Baja	12.118 V	12.112 V	12.115 V	12.109 V	12.114 V
	Nominal	12.086 V	12.079 V	12.083 V	12.075 V	12.081 V
	Alta	12.041 V	12.034 V	12.038 V	12.029 V	12.035 V
5,0 V	Baja	5.103 V	5.099 V	5.101 V	5.098 V	5.100 V
	Nominal	5.082 V	5.078 V	5.081 V	5.076 V	5.079 V
	Alta	5.061 V	5.056 V	5.059 V	5.054 V	5.057 V
3,3 V	Baja	3.318 V	3.316 V	3.317 V	3.315 V	3.316 V
	Nominal	3.311 V	3.309 V	3.310 V	3.308 V	3.309 V
	Alta	3.303 V	3.301 V	3.302 V	3.300 V	3.301 V

Fuente. Elaboración propia.

Anexo C

Fragmentos representativos del firmware implementado

C.1. Configuración general del nodo y tópicos MQTT

El fragmento muestra los parámetros principales del firmware, incluyendo la red WiFi dedicada, el broker MQTT local, los tópicos de publicación y suscripción, el identificador del gabinete y los intervalos de operación del nodo.

```
const char* WIFI_SSID     = "GabiNet-IoT";
const char* WIFI_PASS     = "ClaveSegura2024";
const char* MQTT_BROKER   = "192.168.4.1";
const int   MQTT_PORT     = 1883;

const char* TOPIC_PUB     = "sensores/datos";
const char* TOPIC_RGB_SUB = "actuadores/+/rgb";
const char* GABINETE_ID   = "cab-1";
const int   CHANNEL       = 1;

const unsigned long INTERVALO_PUB  = 3000;
const unsigned long INTERVALO_SCAN = 10000;
```

Figura C1 **Configuración general del nodo y tópicos MQTT.** Fuente. Elaboración propia.

C.2. Escaneo y selección de sensores mediante multiplexor I²C

Se evidencia la lógica usada para seleccionar canales del multiplexor TCA9548A y verificar la presencia de sensores I²C mediante consulta a sus direcciones, lo que permite organizar la adquisición de datos desde sensores distribuidos en distintos puertos físicos.

```
34  #define TCA_ADDR     0x70
35  #define TCA_CHANNELS 8
36  #define VL53_MUX_CH  7
37
38  void tcaSelect(uint8_t ch) {
39    Wire.beginTransmission(TCA_ADDR);
40    Wire.write(ch > 7 ? 0 : (1 << ch));
41    Wire.endTransmission();
42    delay(10);
43  }

231
232 bool i2cProbe(uint8_t addr) {
233   for (int i = 0; i < 2; i++) {
234     Wire.beginTransmission(addr);
235     if (Wire.endTransmission() == 0) return true;
236     delay(20);
237   }
238   return false;
239 }
```

Figura C2 **Escaneo y selección de sensores mediante multiplexor I²C.** Fuente. Elaboración propia.

C.3. Conexión WiFi, reconexión MQTT y suscripción a comandos

El fragmento muestra la lógica de reconexión automática del nodo ESP32 ante pérdida de conectividad WiFi o MQTT. Una vez restablecida la conexión con el bróker, el firmware se suscribe al tópico de control RGB para recibir comandos desde la plataforma.

```
342  void connectWifi() {
343    WiFi.mode(WIFI_STA);
344    WiFi.begin(WIFI_SSID, WIFI_PASS);
345    Serial.printf("\n  WiFi: conectando a %s", WIFI_SSID);
346    unsigned long attempts = 0;
347    while (WiFi.status() != WL_CONNECTED) {
348      delay(500); Serial.print(".");
349      if (++attempts % 5 == 0) {
350        Serial.println("\n  [WiFi] reintentando...");
351        WiFi.disconnect(); delay(1000); WiFi.begin(WIFI_SSID, WIFI_PASS);
352      }
353    }
354    Serial.printf("\n  ✓ WiFi OK  IP=%s\n", WiFi.localIP().toString().c_str());
355  }

357  void connectMQTT() {
358    while (!mqtt.connected()) {
359      Serial.printf("  MQTT: conectando a %s:%d...", MQTT_BROKER, MQTT_PORT);
360      if (mqtt.connect("esp32-cab1")) {
361        mqtt.subscribe(TOPIC_RGB_SUB);
362        Serial.printf(" OK  pub=%s\n", TOPIC_PUB);
363      } else {
364        Serial.printf(" fallo rc=%d, reintento 3s\n", mqtt.state());
365        delay(3000);
366      }
367    }
368  }
```

Figura C3 **Lógica de reconexión automática ante pérdida de conexión WIFI y MQTT.** Fuente. Elaboración propia.

C.4. Construcción de la carga útil (payload) JSON y publicación MQTT

Se muestra la estructura del mensaje JSON consolidado publicado por el ESP32. En un solo bloque JSON se agrupan las lecturas de sensores ambientales, estado de puerta, medición del sensor MQ-2 y estado del actuador RGB, reduciendo publicaciones individuales hacia el broker MQTT

```
546
547    StaticJsonDocument<2048> doc;
548    doc["channel"]     = CHANNEL;
549    doc["gabinete_id"] = GABINETE_ID;
550
551    JsonArray shtArr = doc.createNestedArray("sht31");
552    JsonArray bhArr  = doc.createNestedArray("bh1750");
553    JsonArray mpuArr = doc.createNestedArray("mpu6050");
554    JsonArray vlArr  = doc.createNestedArray("vl53l0x");
555
```

```
654
655   doc["reed_closed"] = !reedStableState;  // true=cerrada, false=abierta
656
657   float mq2Ppm   = (float)analogRead(PIN_MQ2_AO) / 4095.0f * 1000.0f;
658   bool  mq2Alarm = (digitalRead(PIN_MQ2_DO) == HIGH);
659   JsonObject mq2 = doc.createNestedObject("mq2");
660   mq2["port"]  = PORT_MQ2;
661   mq2["ppm"]   = roundf(mq2Ppm * 10) / 10.0f;
662   mq2["alarm"] = mq2Alarm;
663
664   JsonObject rgb = doc.createNestedObject("rgb");
665   rgb["r"]       = rgbState.r;
666   rgb["g"]       = rgbState.g;
667   rgb["b"]       = rgbState.b;
668   rgb["on"]        = rgbState.on;
669   rgb["mode"]      = currentRgbMode;
670   rgb["period_ms"] = currentPeriodMs;
671
672   char buf[2048];
673   size_t n   = serializeJson(doc, buf);
674   bool pubOk = mqtt.publish(TOPIC_PUB, buf, n);
```

Figura C4 **Construcción de la carga útil (payload) JSON y publicación MQTT.** Fuente. Elaboración propia.

Anexo D

Fragmentos representativos del modelo de detección de anomalías basado en Isolation Forest.

El Anexo D se organiza en dos partes principales. Las secciones D.1 a D.4 corresponden al script entrenar_modelo.py, donde se realiza la extracción de datos históricos, preparación de características, normalización, entrenamiento del modelo Isolation Forest y generación de los artefactos modelo_if.pkl, scaler.pkl y features.pkl. Las secciones D.5 a D.7 corresponden al script ml_service.py, desplegado en la Raspberry Pi, encargado de cargar dichos artefactos, consultar las lecturas recientes desde MariaDB, ejecutar la inferencia de anomalías y registrar los resultados en la base de datos

D.1. Configuración general del script de entrenamiento

El fragmento presenta los parámetros principales empleados durante el entrenamiento del modelo, incluyendo la conexión a la base de datos MariaDB y los hiperparámetros configurados para el algoritmo Isolation Forest.

```
import pandas as pd
import numpy as np
import pymysql
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
# ──────────────────────────────────────
# CONFIGURACION
# ──────────────────────────────────────
DB_CONFIG = {
    "host": "192.168.0.25",
    "user": "ml_user",
    "password": "Focm1903$",
    "database": "monitoring_iot"
}
N_ESTIMATORS  = 100
CONTAMINATION = 0.02   # ~2% de datos anomalos asumidos en entrenamiento
RANDOM_STATE  = 42
```

Figura D1 **Configuración general del script de entrenamiento.** Fuente. Elaboración propia.

D.2. Extracción y estructuración de lecturas históricas desde MariaDB

El siguiente fragmento muestra la conexión a la base de datos, la extracción de lecturas históricas desde la tabla lecturas y su posterior reorganización mediante una operación de pivotado. Esta transformación permite convertir las lecturas individuales de cada sensor en una estructura tabular, donde cada fila representa un instante de tiempo y cada columna corresponde a una variable del sistema.

entrenar_modelo.py > ...

```
print("Conectando a MariaDB...")
conn = pymysql.connect(**DB_CONFIG)

query = """
SELECT timestamp, sensor_id, tipo_medida, valor
FROM lecturas
WHERE (sensor_id = '0x44' AND tipo_medida IN ('temperatura', 'humedad'))
   OR (sensor_id = '0x23' AND tipo_medida = 'luz')
   OR (sensor_id = '0x29' AND tipo_medida = 'distancia')
   OR (sensor_id = '0x68' AND tipo_medida = 'estado')
   OR (sensor_id = 'mq2'  AND tipo_medida = 'humo')
   OR (sensor_id = 'reed' AND tipo_medida = 'estado')
ORDER BY timestamp
"""

print("Cargando datos (puede tardar unos segundos)...")
df_raw = pd.read_sql(query, conn)
conn.close()

print(f"Registros cargados: {len(df_raw):,}")

# 2. PIVOTAR - una columna por sensor
# ─────────────────────────────────────
df_raw["feature"]    = df_raw["sensor_id"] + "_" + df_raw["tipo_medida"]
df_raw["timestamp"]  = pd.to_datetime(df_raw["timestamp"])
df_raw["ts_round"]   = df_raw["timestamp"].dt.round("10s")


df = df_raw.pivot_table(
    index="ts_round",
    columns="feature",
    values="valor",
    aggfunc="mean"
).reset_index()

df.columns.name = None
df = df.rename(columns={
    "ts_round":          "timestamp",
    "0x44_temperatura": "temperatura",
    "0x44_humedad":     "humedad",
    "0x23_luz":         "luz",
    "0x29_distancia":   "distancia",
    "0x68_estado":      "mpu6050",
    "mq2_humo":         "humo",
    "pir_movimiento":   "movimiento",
    "reed_estado":      "reed"
})

print(f"Registros despues de pivotar: {len(df):,}")
print("\nValores nulos por columna:")
print(df.isnull().sum())

df = df.dropna(thresh=6)
df = df.fillna(df.median(numeric_only=True))
print(f"Registros limpios: {len(df):,}")
```

Figura D2 **Extracción y estructuración de lecturas históricas desde MariaDB.** Fuente. Elaboración propia.

D.3. Generación del vector de características

El fragmento muestra la construcción del vector de características empleado por el modelo. Además de las variables medidas directamente por los sensores, se incorporan características derivadas como la variación de temperatura y la codificación cíclica de la hora.

```
83   # 3. FEATURES DERIVADAS
84   # ──────────────────────────────────────
85   df = df.sort_values("timestamp").reset_index(drop=True)
86
87   df["temp_rate"] = df["temperatura"].diff().fillna(0)
88
89   df["hora"]     = df["timestamp"].dt.hour + df["timestamp"].dt.minute / 60
90   df["hora_sin"] = np.sin(2 * np.pi * df["hora"] / 24)
91   df["hora_cos"] = np.cos(2 * np.pi * df["hora"] / 24)
92
93   FEATURES = [
94       "temperatura", "humedad", "luz", "distancia",
95       "humo", "movimiento", "reed", "mpu6050",
96       "temp_rate", "hora_sin", "hora_cos"
97   ]
98
99   X = df[FEATURES].copy()
100  print(f"\nShape del dataset de entrenamiento: {X.shape}")
```

Figura D3 **Generación del vector de característica. Fuente.** Elaboración propia.

D.4. Normalización, entrenamiento y almacenamiento del modelo

El siguiente fragmento resume la etapa principal del entrenamiento. Primero se normalizan las características mediante StandardScaler, luego se entrena el modelo Isolation Forest y finalmente se guardan los artefactos necesarios para su despliegue en la Raspberry Pi.

```
137  # 5. ENTRENAR ISOLATION FOREST
138  # ──────────────────────────────────────
139  print("\nNormalizando features...")
140  scaler   = StandardScaler()
141  X_scaled = scaler.fit_transform(X)
142
143  print(f"Entrenando Isolation Forest...")
144  modelo = IsolationForest(
145      n_estimators=N_ESTIMATORS,
146      contamination=CONTAMINATION,
147      random_state=RANDOM_STATE,
148      n_jobs=-1
149  )
150  modelo.fit(X_scaled)
151
152  df["score"]    = modelo.decision_function(X_scaled)
153  df["anomalia"] = modelo.predict(X_scaled)   # -1 = anomalía, 1 = normal
```

```
155  n_anomalias = (df["anomalia"] == -1).sum()
156  print(f"Anomalias detectadas: {n_anomalias} ({n_anomalias/len(df)*100:.1f}%)")
157
158  # Grafica de scores en el tiempo
159  plt.figure(figsize=(14, 4))
160  plt.plot(df["timestamp"].iloc[::50], df["score"].iloc[::50],
161           color="steelblue", linewidth=0.7, label="Score")
162  plt.axhline(y=0, color="tomato", linestyle="--", linewidth=1, label="Umbral (0)")
163  anom = df[df["anomalia"] == -1].iloc[::10]
164  plt.scatter(anom["timestamp"], anom["score"], color="tomato", s=5, zorder=5, label="Anomalia")
165  plt.title("Scores del Isolation Forest")
166  plt.xlabel("Tiempo")
167  plt.ylabel("Decision score")
168  plt.legend()
169  plt.tight_layout()
170  plt.savefig("scores_isolation_forest.png", dpi=120, bbox_inches="tight")
171  plt.show()
172  print("Guardada: scores_isolation_forest.png")
173
174  # ─────────────────────────────────────────
175  # 6. GUARDAR MODELO
176  # ─────────────────────────────────────────
177  joblib.dump(modelo,   "modelo_if.pkl")
178  joblib.dump(scaler,   "scaler.pkl")
179  joblib.dump(FEATURES, "features.pkl")
```

Figura D4 **Normalización, entrenamiento y almacenamiento del modelo.** Fuente. Elaboración propia.

D.5. Configuración y carga del modelo en el microservicio de inferencia

El microservicio ml_service.py realiza inferencias periódicas sobre las lecturas recientes del gabinete. El fragmento presenta la configuración principal del servicio y la carga de los artefactos generados durante el entrenamiento. El servicio trabaja con una ventana de análisis de 10 minutos y se ejecuta cada 600 segundos. La carga de modelo_if.pkl, scaler.pkl y features.pkl permite utilizar en producción el mismo modelo, normalizador y orden de características definidos durante el entrenamiento.

```
ml_service.py > ...
11   import logging
12   import json
13   import numpy as np
14   import pymysql
15   import joblib
16   from datetime import datetime
17   # ─────────────────────────────────────────
18   # CONFIGURACION
19   # ─────────────────────────────────────────
20   DB_CONFIG = {
21       "host":     "localhost",
22       "user":     "ml_user",
23       "password": "Focm1903$",
24       "database": "monitoring_iot"
25   }
26   MODELO_PATH   = "/home/admin/ml_monitoreo/modelo_if.pkl"
27   SCALER_PATH   = "/home/admin/ml_monitoreo/scaler.pkl"
28   FEATURES_PATH = "/home/admin/ml_monitoreo/features.pkl"
```

```
31
32  INTERVALO_SEG = 600   # 10 minutos
33  GABINETE_ID   = 1
34  VENTANA_MIN   = 10    # ultimos minutos de lecturas a considerar
35
36  logging.basicConfig(
37      level=logging.INFO,
38      format="%(asctime)s [%(levelname)s] %(message)s",
39      handlers=[
40          logging.FileHandler("/home/admin/ml_monitoreo/ml_service.log"),
41          logging.StreamHandler()
42      ]
43  )
44  log = logging.getLogger(__name__)
45
```

Figura D5 **Configuración y carga del modelo en el microservicio de inferencia.** Fuente. Elaboración propia.

D.6. Obtención de lecturas recientes y construcción de características

El siguiente fragmento muestra las funciones encargadas de consultar las lecturas recientes desde MariaDB y convertirlas en un vector compatible con el modelo. Para cada sensor se calcula el promedio dentro de la ventana temporal definida. Esta etapa permite convertir las lecturas recientes del gabinete en las mismas once características utilizadas durante el entrenamiento. En el microservicio, la variable temp_rate se fija en cero debido a que la inferencia se realiza sobre un promedio puntual de la ventana reciente.

```
56
57  def get_conn():
58      return pymysql.connect(**DB_CONFIG, cursorclass=pymysql.cursors.DictCursor)
59
60
61  def obtener_ultimas_lecturas(conn):
62      """Obtiene el promedio de los ultimos VENTANA_MIN minutos por sensor."""
63      query = """
64          SELECT sensor_id, tipo_medida, AVG(valor) as valor
65          FROM lecturas
66          WHERE timestamp >= NOW() - INTERVAL %s MINUTE
67            AND (
68                (sensor_id = '0x44' AND tipo_medida IN ('temperatura', 'humedad'))
69             OR (sensor_id = '0x23' AND tipo_medida = 'luz')
70             OR (sensor_id = '0x29' AND tipo_medida = 'distancia')
71             OR (sensor_id = '0x68' AND tipo_medida = 'estado')
72             OR (sensor_id = 'mq2'  AND tipo_medida = 'humo')
73             OR (sensor_id = 'pir'  AND tipo_medida = 'movimiento')
74             OR (sensor_id = 'reed' AND tipo_medida = 'estado')
75            )
76          GROUP BY sensor_id, tipo_medida
77      """
78      with conn.cursor() as cur:
79          cur.execute(query, (VENTANA_MIN,))
80          rows = cur.fetchall()
81      return rows
```

```python
def construir_features(rows):
    """Convierte filas de BD en vector de features para el modelo."""
    data = {}
    for r in rows:
        key = f"{r['sensor_id']}_{r['tipo_medida']}"
        data[key] = float(r['valor'])

    features = {
        "temperatura": data.get("0x44_temperatura", np.nan),
        "humedad":     data.get("0x44_humedad",     np.nan),
        "luz":         data.get("0x23_luz",         np.nan),
        "distancia":   data.get("0x29_distancia",   np.nan),
        "mpu6050":     data.get("0x68_estado",      0.0),
        "humo":        data.get("mq2_humo",         np.nan),
        "movimiento":  data.get("pir_movimiento",   0.0),
        "reed":        data.get("reed_estado",      0.0),
        "temp_rate":   0.0,   # no calculable en inferencia puntual

        "hora_sin":    np.sin(2 * np.pi * datetime.now().hour / 24),
        "hora_cos":    np.cos(2 * np.pi * datetime.now().hour / 24),
    }
    return features
```

Figura D6 **Obtención de lecturas recientes y construcción de características.** Fuente. Elaboración propia.

D.7. Inferencia, almacenamiento del resultado y ciclo autónomo

El último fragmento resume la ejecución principal del microservicio. En cada ciclo se obtienen las lecturas recientes, se construye el vector de entrada, se normaliza, se evalúa con el modelo Isolation Forest y se almacena el resultado en la tabla anomalias_ml. Este ciclo permite que la detección de anomalías opere de manera autónoma en el Gateway IoT. La inferencia se realiza de forma local, sin depender de procesamiento externo en la nube, y cada resultado queda registrado en la base de datos junto con el score de anomalía y las características evaluadas.

```python
def guardar_resultado(conn, score, es_anomalia, features_dict):
    """Inserta el resultado de inferencia en anomalias_ml."""
    query = """
        INSERT INTO anomalias_ml
            (gabinete_id, timestamp, score_anomalia, es_anomalia, features_json)
        VALUES (%s, NOW(), %s, %s, %s)
    """
    with conn.cursor() as cur:
        cur.execute(query, (
            GABINETE_ID,
            float(score),
            int(es_anomalia),
            json.dumps(features_dict)
        ))
    conn.commit()

# LOOP PRINCIPAL
# ─────────────────────────────────────────
log.info("Microservicio ML iniciado. Intervalo: %d segundos", INTERVALO_SEG)

while True:
    try:
        conn = get_conn()

        rows = obtener_ultimas_lecturas(conn)

        if len(rows) < 3:
            log.warning("Pocas lecturas en los ultimos %d min (%d filas). Saltando ciclo.",
            VENTANA_MIN, len(rows))
            conn.close()
            time.sleep(INTERVALO_SEG)
            continue


        features_dict = construir_features(rows)

        # Verificar que no haya NaN en features criticas
        valores = [features_dict[f] for f in FEATURES]
        if any(np.isnan(v) for v in valores):
            # Rellenar NaN con 0 antes de predecir
            valores = [0.0 if np.isnan(v) else v for v in valores]
            log.warning("NaN en features, rellenado con 0.")

        X = np.array(valores).reshape(1, -1)
        X_scaled = scaler.transform(X)

        score       = modelo.decision_function(X_scaled)[0]
        prediccion  = modelo.predict(X_scaled)[0]   # -1 anomalia, 1 normal
        es_anomalia = prediccion == -1

        # Limpiar NaN antes de serializar a JSON
        features_clean = {k: (0.0 if (isinstance(v, float) and np.isnan(v)) else v)
                          for k, v in features_dict.items()}
        guardar_resultado(conn, score, es_anomalia, features_clean)
        conn.close()

        if es_anomalia:
            log.warning("ANOMALIA DETECTADA - score: %.4f | features: %s", score, features_dict)
        else:
            log.info("Normal - score: %.4f", score)

    except Exception as e:
        log.error("Error en ciclo de inferencia: %s", e)

    time.sleep(INTERVALO_SEG)
```

Figura D7 **Inferencia, almacenamiento del resultado y ciclo autónomo.** Fuente. Elaboración propia.

Anexo E

E.1. Servicios de red Access Point en Gateway IoT

El archivo /etc/hostapd/hostapd.conf define el punto de acceso virtual que opera sobre la interfaz uap0. Configura el SSID GabiNet-IoT en el canal 11 de la banda 2,4 GHz con cifrado WPA2-PSK (CCMP). La configuración establece autenticación mediante clave compartida, permitiendo que únicamente los dispositivos con la contraseña definida puedan asociarse al punto de acceso. Esta interfaz funciona como red local dedicada para los dispositivos IoT, facilitando su comunicación con el gateway sin depender directamente de la red principal.

```
admin@monitor-local:~ $ cat /etc/hostapd/hostapd.conf
interface=uap0
driver=nl80211
ssid=GabiNet-IoT
hw_mode=g
channel=11
wmm_enabled=0
macaddr_acl=0
auth_algs=1
wpa=2
wpa_passphrase=ClaveSegura2024
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

Figura E1 **Servicios de red Access Point en Gateway IoT.** Fuente. Elaboración propia.

E.2. Servicio de red uap0 para servicio DHCP

La configuración efectiva de dnsmasq para el rango DHCP de la interfaz uap0 se ubica en /etc/dnsmasq.d/uap0.conf. El fragmento activo asigna direcciones IP

dentro del rango 192.168.4.10 - 192.168.4.50 exclusivamente sobre la interfaz uap0, con la máscara de red 255.255.255.0.

```
admin@monitor-local:~ $ ls /etc/dnsmasq.d/ && echo "----" && cat /etc/dnsmasq.d/*.conf
README  uap0.conf
----
interface=uap0
dhcp-range=192.168.4.10,192.168.4.50,255.255.255.0,24h
dhcp-option=3,192.168.4.1
dhcp-option=6,8.8.8.8
admin@monitor-local:~ $
```

Figura E2 **Servicio de red uap0 para servicio DHCP Fuente.** Elaboración propia.

E.3. Servicio de mensajería MQTT BROKER

El archivo /etc/mosquitto/mosquitto.conf configura el broker MQTT para escuchar en todas las interfaces (0.0.0.0) sobre el puerto TCP 1883, habilitando comunicación sin autenticación dentro de la red privada del Gateway.

```
admin@monitor-local:~ $ grep -v "^#" /etc/mosquitto/mosquitto.conf | grep -v "^$"
persistence true
persistence_location /var/lib/mosquitto/
log_dest file /var/log/mosquitto/mosquitto.log
include_dir /etc/mosquitto/conf.d
listener 1883 0.0.0.0
allow_anonymous true
admin@monitor-local:~ $
```

Figura E3 **Servicio de mensajería MQTT.** Fuente. Elaboración propia.

E.4. Unidades systemd

Fragmento de código de la unidad principal de la capa de aplicación. Levanta el backend Flask mediante Gunicorn con 2 workers sobre el puerto 5000. Gestiona internamente las rutas REST, el motor de alertas, las rutas SNMP y el envío de notificaciones SMTP. Además, la unidad queda definida para iniciar automáticamente junto con el sistema, después de que los servicios de red, MariaDB y Mosquitto estén disponibles. También se configuran archivos de log separados para accesos y errores, lo que facilita el seguimiento del comportamiento del backend y la revisión de fallas durante la operación.

```
admin@monitor-local:~ $ cat /etc/systemd/system/scmp-api.serte
[Unit]
  Description=SCMP Backend API (Flask/Gunicorn)
  After=network.target mariadb.service mosquitto.service

[Service]
  User=admin
  WorkingDirectory=/home/admin/monitoreo/backend
  Environment="PATH=/home/admin/monitoreo/backend/venv/bin"
  ExecStart=/home/admin/monitoreo/backend/venv/bin/gunicorn \
    --workers 2 \
    --bind 0.0.0.0:5000 \
    --timeout 120 \
    --access-logfile /home/admin/monitoreo/backend/access.log \
    --error-logfile /home/admin/monitoreo/backend/error.log \
    main:app
  Restart=always
  RestartSec=5

[Install]
  WantedBy=multi-user.target
admin@monitor-local:~ $
```

Figura E4 **Archivo de configuración de unidades systemd.** Fuente. Elaboración propia.

E.5. Archivo de configuración servicio de pantallas físicas

Fragmento de código de configuración de la unidad que ejecuta el script Python de control del display TFT circular conectado mediante SPI. Proporciona la interfaz de visualización local de datos de temperatura, humedad y estado del sistema sin requerir acceso a la interfaz web.

```
admin@monitor-local:~ $ cat /etc/systemd/system/scmp-api.service
[Unit]
Description=SCMP Backend API (Flask/Gunicorn)
After=network.target mariadb.service mosquito.service

[Service]
User=admin
WorkingDirectory=/home/admin/monitoreo/backend
Environment="PATH=/home/admin/monitoreo/backend/venv/bin"
ExecStart=/home/admin/monitoreo/backend/venv/bin/gunicorn \
    --workers 2 \
    --bind 0.0.0.0:5000 \
    --timeout 120 \
    --access-logfile /home/admin/monitoreo/backend/access.log \
    --error-logfile /home/admin/monitoreo/backend/error.log \
    main:app
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
admin@monitor-local:~ $
```

Figura E5 **Archivo de configuración servicio de pantallas físicas.** Fuente. Elaboración propia.

E.6. Nginx

Nginx actúa como proxy inverso que expone el frontend React (compilado en /frontend/build) y redirige las peticiones al prefijo /api/ hacia el backend Gunicorn en el puerto 5000. Esta configuración permite que el frontend y la API sean accesibles desde un único punto de entrada en el puerto 80, a continuación, se presenta el fragmento del archivo de configuración.

```
admin@monitor-local:~ $ cat /etc/nginx/sites-enabled/default
server {
    listen 80;
    server_name _;

    root  /home/admin/monitoreo/frontend/build;
    index index.html;

    location /api/ {
        proxy_pass http://127.0.0.1:5000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location / {
        try_files $uri $uri/ /index.html;
    }
}
admin@monitor-local:~ $
```

Figura E6 **Archivo de configuración NGINX para el proxy inverso en el Frontend.** Fuente. Elaboración propia.

E.7. Script de validación servicios Gateway IoT

```
ESCENARIO="${1:-sin_sensores}"
BROKER="localhost"
MUESTRAS_LATENCIA=30
MUESTRAS_CPU=60        # segundos de medición CPU/RAM
TOPIC="prueba/latencia/medicion$$"
FECHA=$(date +"%Y%m%d_%H%M%S")
ARCHIVO="resultado_${ESCENARIO}_${FECHA}.txt"

# ── Colores ──────────────────────────────────────────────
R='\033[0;31m'; G='\033[0;32m'; Y='\033[1;33m'; B='\033[0;34m'; NC='\033[0m'
log()  { echo -e "${G}[$(date +%H:%M:%S)]${NC} $1" | tee -a "$ARCHIVO"; }
warn() { echo -e "${Y}[AVISO]${NC} $1" | tee -a "$ARCHIVO"; }
sep()  { echo "──────────────────────────────────────────────────" | tee -a "$ARCHIVO"; }

# ── Cabecera ─────────────────────────────────────────────
{
echo "=================================================="
echo "  CARACTERIZACIÓN DEL GATEWAY IoT — $(date)"
echo "  Escenario : $ESCENARIO"
echo "  Broker    : $BROKER"
echo "  Duración  : ~$((MUESTRAS_CPU * 2 + 10)) segundos"
echo "=================================================="
} | tee "$ARCHIVO"
echo ""

# ── Verificar dependencias ───────────────────────────────
for cmd in mosquitto_pub free vmstat awk; do
    if ! command -v "$cmd" &>/dev/null; then
        warn "Comando '$cmd' no encontrado."
        if [[ "$cmd" == "mosquitto_pub" ]]; then
            warn "Instala con: sudo apt install mosquitto-clients"
        fi
        exit 1
    fi
done

# ── Función: uso promedio de CPU (vmstat, sin dependencias extra) ──
medir_cpu_ram() {
    local duracion=$1
    local label=$2

    log "Midiendo CPU y RAM — $label ($duracion segundos)..."
#   Función: uso promedio de CPU (vmstat, sin dependencias extra)
medir_cpu_ram() {
    local duracion=$1
    local label=$2

    log "Midiendo CPU y RAM — $label ($duracion segundos)..."

    # vmstat: columnas id=idle, wa=iowait
    # Toma una muestra cada segundo durante $duracion segundos,
    # descarta la primera línea (promedio desde boot) y promedia el resto
    # Escribe directamente en variables globales (evita eval con eval+log)
    CPU_USO=$(vmstat 1 "$duracion" | tail -n +4 | awk '
        {idle += $15; total++}
        END {
            uso = 100 - (idle / total)
            printf "%.1f", uso
        }
    ')
    RAM_TOTAL=$(free -m | awk '/^Mem:/ {print $2}')
    RAM_USADO=$(free -m | awk '/^Mem:/ {print $3}')
    RAM_DISPONIBLE=$(free -m | awk '/^Mem:/ {print $7}')
}
```

```
72
73  # ── FASE 1: Reposo (servicios activos, sin carga MQTT extra) ─
74  sep
75  log "FASE 1 — Reposo: midiendo recursos sin carga adicional"
76  log "(El ESP32 puede estar o no publicando según el escenario '$ESCENARIO')"
77  sep
78
79  medir_cpu_ram $MUESTRAS_CPU 'Reposo'
80  CPU_REPOSO=$CPU_USO
81  RAM_TOTAL_R=$RAM_TOTAL
82  RAM_USADO_R=$RAM_USADO
83  RAM_DISP_R=$RAM_DISPONIBLE
84
85  {
86  echo ""
87  echo "=== RESULTADO FASE 1 — Reposo ==="
88  echo "  CPU uso promedio : ${CPU_REPOSO} %"
89  echo "  RAM total        : ${RAM_TOTAL_R} MB"
90  echo "  RAM utilizada    : ${RAM_USADO_R} MB"
91  echo "  RAM disponible   : ${RAM_DISP_R} MB"
92  } | tee -a "$ARCHIVO"
93
94  # ── FASE 2: Latencia MQTT (30 muestras) ──────────────────
95  sep
96  log "FASE 2 — Latencia MQTT: $MUESTRAS_LATENCIA muestras con QoS 1"
97  sep
98
99  SUMA_LAT=0
100 MIN_LAT=99999
101 MAX_LAT=0
102 declare -a LISTA_LAT=()
103
104 for i in $(seq 1 $MUESTRAS_LATENCIA); do
105     T_INI=$(date +%s%3N)
106     mosquitto_pub -h "$BROKER" -t "$TOPIC" -m "ping_${i}" -q 1 2>/dev/null
107     T_FIN=$(date +%s%3N)
108
109     LAT=$((T_FIN - T_INI))
110     LISTA_LAT+=($LAT)
111     SUMA_LAT=$((SUMA_LAT + LAT))
112     [[ $LAT -lt $MIN_LAT ]] && MIN_LAT=$LAT
113     [[ $LAT -gt $MAX_LAT ]] && MAX_LAT=$LAT
114
115     printf "  Muestra %2d : %3d ms\n" "$i" "$LAT" | tee -a "$ARCHIVO"
116     sleep 0.1
117 done
118
119 MEDIA_LAT=$((SUMA_LAT / MUESTRAS_LATENCIA))
120
121 # Desviación estándar (usando awk)
122 DESV_LAT=$(printf '%s\n' "${LISTA_LAT[@]}" | awk -v media="$MEDIA_LAT" '
123     {suma += ($1 - media)^2; n++}
124     END {printf "%.1f", sqrt(suma / n)}
125 ')
126
127 {
128 echo ""
129 echo "=== RESULTADO FASE 2 — Latencia MQTT ==="
130 echo "  Media            : ${MEDIA_LAT} ms"
131 echo "  Mínima           : ${MIN_LAT} ms"
132 echo "  Máxima           : ${MAX_LAT} ms"
133 echo "  Desv. estándar   : ${DESV_LAT} ms"
134 } | tee -a "$ARCHIVO"
```

```
# ── FASE 3: CPU y RAM bajo carga MQTT generada ──────────────
sep
log "FASE 3 — Carga MQTT: publicando en background + midiendo recursos"
sep

# Lanzar publicaciones en segundo plano (simula carga sostenida)
(
    for i in $(seq 1 300); do
```

```
(
    for i in $(seq 1 300); do
        mosquitto_pub -h "$BROKER" -t "$TOPIC/carga" \
            -m "{\"n\":$i,\"ts\":$(date +%s%3N)}" -q 1 2>/dev/null
        sleep 0.2   # ~5 msg/s durante ~60 segundos
    done
) &
CARGA_PID=$!

medir_cpu_ram $MUESTRAS_CPU 'Bajo carga'
CPU_CARGA=$CPU_USO
RAM_USADO_C=$RAM_USADO
RAM_DISP_C=$RAM_DISPONIBLE

wait $CARGA_PID 2>/dev/null

{
echo ""
echo "=== RESULTADO FASE 3 — Bajo carga MQTT ==="
echo "  CPU uso promedio : ${CPU_CARGA} %"
echo "  RAM total        : ${RAM_TOTAL_R} MB"
echo "  RAM utilizada    : ${RAM_USADO_C} MB"
echo "  RAM disponible   : ${RAM_DISP_C} MB"
} | tee -a "$ARCHIVO"

# ── RESUMEN FINAL ──────────────────────────────────────────
sep
{
echo ""
echo "======================================================"
echo "  RESUMEN FINAL — Escenario: $ESCENARIO"
echo "======================================================"
echo ""
echo "  TABLA A — Recursos computacionales del Gateway"
printf "  %-35s %-10s %-10s\n" "Métrica" "Reposo" "Bajo carga"
printf "  %-35s %-10s %-10s\n" "CPU uso promedio (%)" "${CPU_REPOSO}" "${CPU_CARGA}"
printf "  %-35s %-10s %-10s\n" "RAM total (MB)" "${RAM_TOTAL_R}" "${RAM_TOTAL_R}"
printf "  %-35s %-10s %-10s\n" "RAM utilizada (MB)" "${RAM_USADO_R}" "${RAM_USADO_C}"
printf "  %-35s %-10s %-10s\n" "RAM disponible (MB)" "${RAM_DISP_R}" "${RAM_DISP_C}"
echo ""
echo "  TABLA B — Latencia broker MQTT (${MUESTRAS_LATENCIA} muestras, QoS 1)"
printf "  %-35s %-10s\n" "Media (ms)" "${MEDIA_LAT}"
printf "  %-35s %-10s\n" "Mínima (ms)" "${MIN_LAT}"
printf "  %-35s %-10s\n" "Máxima (ms)" "${MAX_LAT}"
printf "  %-35s %-10s\n" "Desv. estándar (ms)" "${DESV_LAT}"
echo ""
```

Figura E7 **Script de validación servicio MQTT-BASE DE DATOS**. Fuente. Elaboración propia.

Anexo F

```python
import paho.mqtt.client as mqtt
from paho.mqtt.client import CallbackAPIVersion
import requests
import time
import statistics

BROKER    = "localhost"
TOPIC     = "sensores/datos"
LOGIN_URL = "http://localhost:5000/login"
API_URL   = "http://localhost:5000/datos_sensores"
N         = 10

USERNAME = "fcardoza"
PASSWORD = "Focm1903$"

resultados = []
token      = None

def obtener_token():
    r = requests.post(LOGIN_URL, json={"username": USERNAME, "password": PASSWORD}, timeout=5)
    if r.status_code == 200:
        t = r.json().get("token")
        print(f"  Token JWT obtenido OK\n")
        return t
    else:
        print(f"  ERROR al obtener token: {r.status_code}")
        exit(1)

def on_message(client, userdata, msg):
    if len(resultados) >= N:
        return

    t1 = time.time() * 1000

    try:
        headers = {"Authorization": f"Bearer {token}"}
        r  = requests.get(API_URL, headers=headers, timeout=3)
        t2 = time.time() * 1000
        dt = round(t2 - t1, 1)
        resultados.append(dt)
        print(f"  [{len(resultados):2d}/10]  MQTT->API:  {dt} ms   (status {r.status_code})")
    except Exception as e:
        print(f"  Error: {e}")

    if len(resultados) >= N:
        print()
        print("╔═══════════════════════════════════════════════╗")
        print("║   Tabla 37 – Latencia MQTT → Dashboard        ║")
        print("╠═══════════════════════════════════════════════╣")
        for i, v in enumerate(resultados, 1):
            print(f"║  Medicion {i:2d}:          {v:7.1f} ms            ║")
        print("╠═══════════════════════════════════════════════╣")
        avg = statistics.mean(resultados)
        print(f"║  Promedio:           {avg:7.1f} ms            ║")
        print(f"║  Minimo:             {min(resultados):7.1f} ms            ║")
        print(f"║  Maximo:             {max(resultados):7.1f} ms            ║")
        print(f"║  Desv. estandar:     {statistics.stdev(resultados):7.1f} ms            ║")
        print("╚═══════════════════════════════════════════════╝")
        client.disconnect()

token = obtener_token()

client = mqtt.Client(CallbackAPIVersion.VERSION2)
client.on_message = on_message
client.connect(BROKER, 1883, 60)
client.subscribe(TOPIC)
print(f"Escuchando '{TOPIC}' – esperando {N} publicaciones del ESP32...\n")
client.loop_forever()

```

Figura F1 **Script de validación latencia dashboard**. Fuente. Elaboración propia.

```
import json
import statistics
import subprocess
import sys
import time

MUESTRAS     = 20
TOPIC        = "sensores/datos"
DB           = "monitoring_iot"
POLL_MS      = 2
TIMEOUT_MS   = 8000
RULE_ID      = 3
INTERVALO_S  = 2.0   # pausa entre muestras (> cooldown_sec=1)

# Mensaje con lux=5000 — muy por encima del umbral de 10
MSG_ALTO = json.dumps({
    "gabinete_id": "cab1",
    "channel": 4,
    "bh1750": [{"addr": "0x23", "canal": 4, "lux": 5000}]
})

# Mensaje con lux=0 — por debajo de threshold-hysteresis (10-2=8)
# fuerza la resolucion del incidente activo
MSG_RESET = json.dumps({
    "gabinete_id": "cab1",
    "channel": 4,
    "bh1750": [{"addr": "0x23", "canal": 4, "lux": 0}]
})

# ── Conexion DB ──────────────────────────────────────────────────────
try:
    import pymysql
    conn = pymysql.connect(
        host="127.0.0.1", port=3306,
        user="iotuser", password="iotpass123",
        database=DB,
        autocommit=True
    )
except Exception as e:
    print(f"ERROR conectando a MariaDB: {e}")
    sys.exit(1)

cur = conn.cursor()

# ── Bajar cooldown temporalmente ─────────────────────────────────────
print(f"Ajustando cooldown_sec=1 para regla id={RULE_ID}...")
cur.execute(f"UPDATE alert_rules SET cooldown_sec=1 WHERE id={RULE_ID}")
print("OK")

print("=" * 52)
print(f"  LATENCIA MQTT -> alert_incidents  (n={MUESTRAS})")
print(f"  Regla : TEST  luz > 10  (duration_sec=0)")
print(f"  Topic : {TOPIC}")
print("=" * 52)

latencias = []

try:
    for i in range(1, MUESTRAS + 1):

        # Reset: publicar lux=0 para resolver el incidente activo
```

```
        subprocess.run(
            ["mosquitto_pub", "-h", "localhost", "-t", TOPIC, "-m", MSG_RESET, "-q", "0"],
            capture_output=True
        )
        time.sleep(1.5)  # esperar que el motor resuelva el incidente + cooldown

        # ID maximo en alert_incidents antes del disparo
        cur.execute("SELECT COALESCE(MAX(id), 0) FROM alert_incidents")
        last_id = cur.fetchone()[0]

        # t0 justo antes de publicar el disparo
        t0 = time.perf_counter()

        subprocess.run(
            ["mosquitto_pub", "-h", "localhost", "-t", TOPIC, "-m", MSG_ALTO, "-q", "0"],
            capture_output=True
        )

        # Sondear alert_incidents hasta detectar nuevo registro
        found    = False
        deadline = t0 + TIMEOUT_MS / 1000.0

        while time.perf_counter() < deadline:
            time.sleep(POLL_MS / 1000.0)
            cur.execute(
                "SELECT id FROM alert_incidents WHERE id > %s AND rule_id = %s LIMIT 1",
                (last_id, RULE_ID)
            )
            row = cur.fetchone()
            if row:
                t1  = time.perf_counter()
                lat = round((t1 - t0) * 1000)
                latencias.append(lat)
                print(f"  Muestra {i:2d} : {lat:5d} ms")
                found = True
                break

        if not found:
            print(f"  Muestra {i:2d} : TIMEOUT (>{TIMEOUT_MS} ms)")

        time.sleep(INTERVALO_S)

finally:
    # ── Restaurar cooldown original ──────────────────────────────────
    print("\nRestaurando cooldown_sec=300...")
    cur.execute(f"UPDATE alert_rules SET cooldown_sec=300 WHERE id={RULE_ID}")
    print("OK")
    cur.close()
    conn.close()

# ── Estadisticos ─────────────────────────────────────────────────────
n = len(latencias)
if n == 0:
    print("Sin muestras exitosas. Revisa que alerts_engine este activo.")
    sys.exit(1)

media = statistics.mean(latencias)
desv  = statistics.stdev(latencias) if n > 1 else 0.0
mn    = min(latencias)
mx    = max(latencias)
```

```
136    print("")
137    print("=" * 52)
138    print(f"  RESULTADO ({n} muestras exitosas)")
139    print(f"  Media : {media:.1f} ms")
140    print(f"  Min   : {mn} ms")
141    print(f"  Max   : {mx} ms")
142    print(f"  Desv  : {desv:.1f} ms")
143    print("=" * 52)
```

Figura F2 **Script de validación latencia de alertas**. Fuente. Elaboración propia.

Anexo G

La base de datos monitoring_iot esta implementada en MariaDB/InnoDB y contiene 17 tablas organizadas en cuatro grupos funcionales. Las tablas operativas de alta frecuencia (lecturas, anomalias_ml, alert_incidents, audit_log) disponen de índices en los campos de consulta más frecuente (timestamp, sensor_id, status) para equilibrar las escrituras continuas con las consultas históricas del frontend. A continuación, se presenta la estructura de cada tabla con el tipo de dato, restricción de nulidad, clave y descripción funcional de cada campo.

TABLA G1 usuarios

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(255)	NO		NULL	
username	varchar(128)	NO	UNI	NULL	
password_hash	varchar(512)	NO		NULL	
rol	varchar(32)	NO		operador	
creado_en	timestamp	NO		current_timestamp()	

Fuente. Elaboración propia.

TABLA G2 user_preferences

Field	Type	Null	Key	Default	Extra
user_id	int(11)	NO	PRI	NULL	
theme_key	varchar(64)	NO		default	
theme_mode	varchar(16)	NO		dark	
custom_theme	longtext	YES		NULL	
updated_at	timestamp	NO		current_timestamp()	on update current_timestamp()

Fuente. Elaboración propia.

TABLA G3 dashboard_widgets

Field	Type	Null	Key	Default	Extra
id	varchar(64)	NO	PRI	NULL	
title	varchar(255)	YES			
chart_type	varchar(64)	YES		line	
measure	varchar(64)	YES			
sensor_scope	varchar(32)	YES		any	
sensor_id	varchar(128)	YES			
agg	varchar(16)	YES		none	
time_range	longtext	YES		NULL	
max_points	int(11)	YES		200	
decimals	int(11)	YES		2	
unit_override	varchar(32)	YES			
pinned	tinyint(1)	YES		0	
col_span	int(11)	YES		1	
gx	int(11)	YES		NULL	
gy	int(11)	YES		NULL	
gw	int(11)	YES		NULL	
gh	int(11)	YES		NULL	

Fuente. Elaboración propia.

TABLA G4 themes

Field	Type	Null	Key	Default	Extra
key_name	varchar(64)	NO	PRI	NULL	
name	varchar(255)	NO			
logo	varchar(255)	YES			
primary_color	varchar(32)	YES		#3b82f6	
bg_color	varchar(32)	YES		#0f172a	
accent_color	varchar(32)	YES		#64748b	

Fuente. Elaboración propia.

TABLA G5 alert_rules

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
enabled	tinyint(1)	NO		1	
scope	varchar(32)	NO		tipo	
selector	varchar(255)	NO			
metric	varchar(64)	NO			
op	varchar(4)	NO		>	
threshold	decimal(10,4)	NO		0.0000	
duration_sec	int(11)	NO		60	
hysteresis	decimal(10,4)	NO		0.0000	
cooldown_sec	int(11)	NO		600	
severity	varchar(32)	NO		warning	
channels_json	longtext	YES		NULL	

Fuente. Elaboración propia.

TABLA G6 alert_incidents

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
rule_id	int(11)	NO	MUL	NULL	
rule_name	varchar(255)	NO			
sensor_id	varchar(128)	YES		NULL	
metric	varchar(64)	YES		NULL	
severity	varchar(32)	YES		warning	
value	decimal(10,4)	YES		NULL	
threshold	decimal(10,4)	YES		NULL	
op	varchar(4)	YES		NULL	
status	varchar(32)	NO	MUL	active	
message	text	YES		NULL	
fired_at	datetime	YES	MUL	NULL	
resolved_at	datetime	YES		NULL	
ack_by	varchar(128)	YES		NULL	
ack_at	datetime	YES		NULL	

Fuente. Elaboración propia.

TABLA G7 smtp_profiles

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
host	varchar(255)	NO		NULL	
port	int(11)	NO		587	
tls_mode	varchar(16)	NO		starttls	
username	varchar(255)	YES		NULL	
password_enc	text	YES		NULL	
from_name	varchar(255)	YES		NULL	
from_email	varchar(255)	NO		NULL	
enabled	tinyint(1)	NO		1	
created_at	timestamp	NO		current_timestamp()	

Fuente. Elaboración propia.

TABLA G8 lecturas

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
sensor_id	varchar(128)	NO	MUL	NULL	
nombre_sensor	varchar(128)	YES		NULL	
tipo_medida	varchar(64)	YES	MUL		
valor	decimal(12,4)	YES		NULL	
unidad	varchar(32)	YES		NULL	
puerto	int(11)	YES		NULL	
timestamp	datetime	NO	MUL	current_timestamp()	

Fuente. Elaboración propia.

TABLA G9 anomalias_ml

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
gabinete_id	int(11)	NO	MUL	1	
timestamp	datetime	NO	MUL	current_timestamp()	
score_anomalia	float	YES		NULL	
es_anomalia	tinyint(1)	NO	MUL	0	
revisado	tinyint(1)	NO	MUL	0	
features_json	longtext	YES		NULL	

Fuente. Elaboración propia.

TABLA G10 gabinetes

Field	Type	Null	Key	Default	Extra
id	varchar(64)	NO	PRI	NULL	
name	varchar(255)	NO			
location	varchar(255)	YES			
status	varchar(32)	YES		OK	
created_at	timestamp	NO		current_timestamp()	

Fuente. Elaboración propia.

TABLA G11 sensor_aliases

Field	Type	Null	Key	Default	Extra
sensor_id	varchar(128)	NO	PRI	NULL	
alias	varchar(255)	NO			
updated_at	timestamp	NO		current_timestamp()	on update current_timestamp()

Fuente. Elaboración propia.

TABLA G12 audit_log

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(128)	YES		sistema	
user_id	int(11)	YES		NULL	
rol	varchar(32)	YES		NULL	
action	varchar(128)	NO	MUL	NULL	
resource_type	varchar(64)	YES		NULL	
resource_id	varchar(128)	YES		NULL	
details	longtext	YES		NULL	
ip_address	varchar(64)	YES		NULL	
status	varchar(32)	YES		success	
timestamp	timestamp	NO	MUL	current_timestamp()	

Fuente. Elaboración propia.

TABLA G13 wifi_networks

Field	Type	Null	Key	Default	Extra
user_id	int(11)	NO	PRI	NULL	
theme_key	varchar(64)	NO		default	
theme_mode	varchar(16)	NO		dark	
custom_theme	longtext	YES		NULL	
updated_at	timestamp	NO		current_timestamp()	on update current_timestamp()

Fuente. Elaboración propia.

TABLA G14 security_networks

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
cidr	varchar(64)	NO	UNI	NULL	
description	varchar(255)	YES		NULL	

Fuente. Elaboración propia.

TABLA G15 security_services

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(128)	NO		NULL	
port	int(11)	NO		NULL	
protocol	varchar(8)	YES		tcp	
system_name	varchar(128)	YES			

Fuente. Elaboración propia.

TABLA G16 service_acl

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
service_id	int(11)	NO	MUL	NULL	
network_id	int(11)	NO	MUL	NULL	
allowed	tinyint(1)	NO		1	

Fuente. Elaboración propia.

TABLA G17 security_audit

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
action	varchar(128)	YES		NULL	
details	text	YES		NULL	
ip_address	varchar(64)	YES		NULL	
created_at	timestamp	NO	MUL	current_timestamp()	

Fuente. Elaboración propia.

Anexo H

Fragmento de código HTML correspondiente al mockup desarrollado para el frontend del sistema. En este se define la estructura base de la interfaz web, incluyendo la organización inicial de los contenedores, secciones, encabezados, tarjetas informativas y elementos visuales que conforman el panel principal de monitoreo. Asimismo, se incorporan los estilos visuales principales, como la distribución de los componentes, la jerarquía de la información, el uso de colores, tipografías y espacios, con el objetivo de representar de manera preliminar la apariencia final de la plataforma.

Este mockup fue utilizado como una referencia visual y funcional durante la etapa de diseño.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>MonitorIoT — Plataforma completa</title>
<link href="https://fonts.googleapis.com/css2?family=Plus+Jakarta+Sans:wght@400;500;600;700;800&family=JetBrains+Mono:wght@400;500;600&display=swap" rel="stylesheet">
<style>
*,*::before,*::after{box-sizing:border-box;margin:0;padding:0}
:root{
  --bg:#F1F4F9;--surface:#fff;--surface2:#F7F9FC;--border:#E3E8F0;--border2:#C8D3E8;
  --accent:#2563EB;--accent-l:#EEF3FF;
  --green:#059669;--green-l:#ECFDF5;--green-b:#6EE7B7;
  --amber:#D97706;--amber-l:#FFFBEB;--amber-b:#FDE68A;
  --red:#DC2626;--red-l:#FEF2F2;--red-b:#FECACA;
  --purple:#7C3AED;--purple-l:#F5F3FF;--purple-b:#DDD6FE;
  --t1:#0F172A;--t2:#475569;--t3:#94A3B8;--t4:#CBD5E1;
  --sans:'Plus Jakarta Sans',sans-serif;--mono:'JetBrains Mono',monospace;
  --r:12px;--shadow:0 1px 3px rgba(15,23,42,.06),0 1px 2px rgba(15,23,42,.04);
}
body{font-family:var(--sans);background:var(--bg);color:var(--t1);min-height:100vh;display:flex;font-size:14px;}

/* — SIDEBAR — */
.sidebar{width:64px;flex-shrink:0;background:#0F172A;display:flex;flex-direction:column;align-items:center;padding:16px 0 20px;gap:4px;position:fixed;top:0;left:0;bottom:0;z-index:100;}
.sb-logo{width:38px;height:38px;border-radius:10px;background:var(--accent);display:flex;align-items:center;justify-content:center;font-size:18px;margin-bottom:22px;cursor:pointer;}
.sb-icon{width:44px;height:44px;border-radius:11px;display:flex;align-items:center;justify-content:center;font-size:19px;cursor:pointer;color:rgba(255,255,255,.3);transition:all .15s;position:relative;}
.sb-icon:hover{color:rgba(255,255,255,.75);background:rgba(255,255,255,.07);}
.sb-icon.active{background:var(--accent);color:#fff;}
.sb-dot{position:absolute;top:7px;right:7px;width:8px;height:8px;border-radius:50%;background:var(--red);border:2px solid #0F172A;}
.sb-dot.amber{background:var(--amber);}
.sb-divider{width:30px;height:1px;background:rgba(255,255,255,.07);margin:8px 0;}
.sb-bottom{margin-top:auto;}
.sb-avatar{width:36px;height:36px;border-radius:10px;background:#334155;display:flex;align-items:center;justify-content:center;font-size:12px;font-weight:700;color:rgba(255,255,255,.65);cursor:pointer;}

/* — MAIN WRAP — */
.main{flex:1;display:flex;flex-direction:column;min-width:0;margin-left:64px;}

/* — TOPBAR — */
.topbar{height:58px;background:var(--surface);border-bottom:1px solid var(--border);display:flex;align-items:center;padding:0 28px;gap:14px;flex-shrink:0;box-shadow:var(--shadow);position:sticky;top:0;z-index:50;}
.breadcrumb{display:flex;align-items:center;gap:7px;font-size:14px;color:var(--t3);}
.breadcrumb span{color:var(--t1);font-weight:700;}
.bc-sep{color:var(--t4);}
.topbar-right{margin-left:auto;display:flex;align-items:center;gap:10px;}
.online-pill{display:flex;align-items:center;gap:6px;background:var(--green-l);border:1px solid var(--green-b);color:var(--green);font-size:12px;font-weight:600;padding:5px 12px;border-radius:20px;}
.online-dot{width:7px;height:7px;border-radius:50%;background:var(--green);}
.time{font-family:var(--mono);font-size:12px;color:var(--t3);}
.icon-btn{width:36px;height:36px;border-radius:9px;border:1px solid var(--border);background:var(--surface);display:flex;align-items:center;justify-content:center;font-size:16px;cursor:pointer;color:var(--t2);}
```

Figura H **Fragmento de código HTML del mockup diseño para el frontend.** Fuente. Elaboración propia.

Anexo I

Se presenta la configuración realizada en Cloudflare para el túnel asociado al backend local del sistema. Mediante esta configuración, el servicio backend ejecutado en el Gateway IoT sobre el puerto 5000 queda expuesto de forma segura a través del dominio personalizado api.tesis-monitoring.xyz, sin necesidad de publicar directamente la IP pública ni abrir puertos adicionales en la red local.

El uso de Cloudflare Tunnel permite establecer un canal seguro entre la infraestructura local y la nube, reduciendo la superficie de exposición del sistema. De esta manera, se garantiza el acceso externo al backend de forma controlada, estable y segura, manteniendo protegida la red interna del Gateway IoT. Se presenta la configuración realizada en Cloudflare para el túnel asociado al backend local del sistema. Mediante esta configuración, el servicio backend ejecutado en el Gateway IoT sobre el puerto 5000 queda expuesto de forma segura a través del dominio personalizado api.tesis-monitoring.xyz, permitiendo el acceso externo sin publicar directamente la IP pública ni abrir puertos adicionales en la red local.

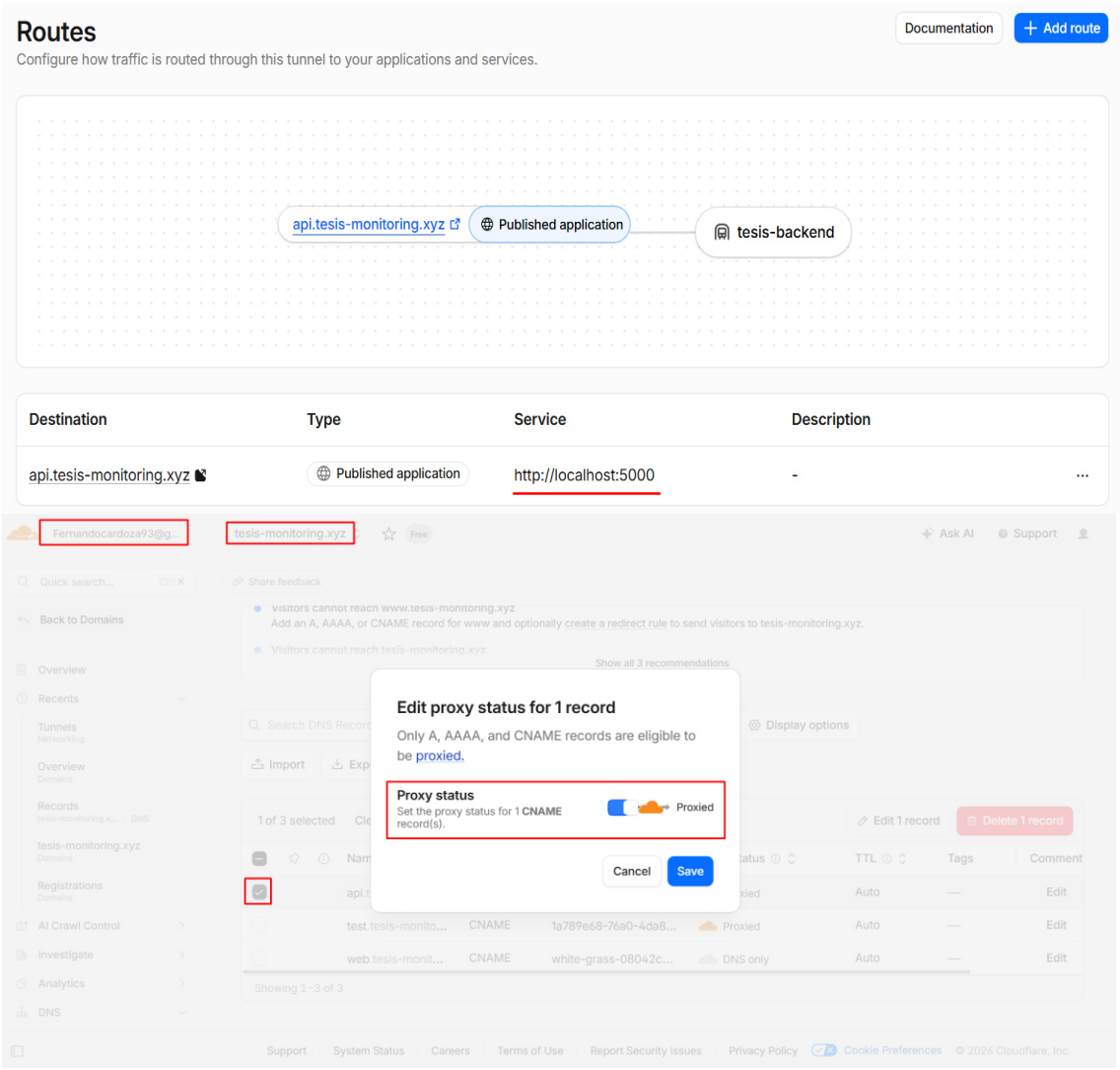


Figura I **Configuración web de Cloudflare para el dominio adquirido para el backend.** Fuente. Elaboración propia.