

UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA

Ingeniería en Sistemas

Curso: Arquitectura de Computadoras II

Catedrático: Alejandro Alberto García

INFORME TÉCNICO DE IMPLEMENTACIÓN

Mantenimiento Predictivo con Edge Computing para Editorial Piedrasanta

Informe técnico de implementación, pruebas y operación

Caso de uso: monitoreo de temperatura y aceleración de una impresora Canon G1110

Integrantes

Nombre	Carné
Didier Javier Mejía Anleu	0900-21-905
Jonathan José Pablo López Palacios	0900-20-25390
David Alejandro Mayen Garcia	0900-23-947
Richard Antonio Valle Monterroso	0900-10-2612

Guatemala, 16 de octubre de 2026

Versión 1.0 – cierre técnico inicial

ÍNDICE

PROPÓSITO DEL DOCUMENTO	3
1. RESUMEN TÉCNICO.....	3
2. OBJETIVO Y ALCANCE.....	3
2.1. Objetivo general.....	3
2.2. Alcance implementado	4
2.3. Fuera de alcance actual	4
3. ARQUITECTURA Y FLUJO DE DATOS.....	4
3.1. Secuencia de operación.....	5
3.2. Interfaces principales	5
4. HARDWARE E INTEGRACIÓN FÍSICA	5
4.1. Formato de telemetría	6
4.2. Criterio de montaje	6
5. SOFTWARE Y DESPLIEGUE	6
5.1. Servicios esperados	7
6. INTELIGENCIA ARTIFICIAL Y FASE 2.....	7
6.1. Modelo inicial	7
6.2. Fase 2 con datos físicos	7
7. PRUEBAS Y RESULTADOS	8
7.1. Resultado de la prueba de clúster	8
8. INCIDENCIAS, RIESGOS Y MITIGACIÓN	9
8.1. Riesgo de interpretación	9
9. OPERACIÓN Y MANTENIMIENTO	9
9.1. Inicio de demostración.....	9
9.2. Captura de Fase 2.....	10
9.3. Control de versiones	10
10. CONCLUSIONES Y PRÓXIMOS PASOS.....	10
10.1. Conclusiones	10
10.2. Próximas actividades	10
10.3. Repositorio.....	11

PROPÓSITO DEL DOCUMENTO

Este documento describe la arquitectura, los componentes, el despliegue, las pruebas y las limitaciones del prototipo. Su función es permitir que cada integrante comprenda el estado real del proyecto y pueda reproducir la demostración sin depender de explicaciones informales.

1. RESUMEN TÉCNICO

Se implementó una arquitectura de Edge Computing para adquirir temperatura y aceleración cerca de un equipo de impresión, conservar las lecturas en una Raspberry Pi Master y distribuir solicitudes de clasificación entre dos Workers. El prototipo opera en la red local: no necesita una API de inteligencia artificial ni un servicio de inferencia en Internet.

La adquisición física usa una ESP32, una sonda DS18B20 y un acelerómetro ADXL345. La ESP32 publica registros JSON por USB serial. Un servicio de la Raspberry consume ese puerto, valida las lecturas y expone el estado al panel web. El Master mantiene historial, consulta la disponibilidad de los Workers y envía las solicitudes por turnos.

Tabla 1

Estado de los resultados del prototipo

Resultado	Estado
Lectura física DS18B20 y ADXL345	Validada
Transmisión ESP32 a Raspberry por USB serial	Validada
Panel web local y API de sensor	Validado
Worker local y Worker remoto	Validados
Alternancia en cuatro solicitudes	Validada: local, remoto, local, remoto
Modelo inicial de clasificación	Validado con datos sintéticos; exactitud reportada 94%
Modelo reentrenado con datos físicos	En progreso: captura real iniciada

2. OBJETIVO Y ALCANCE

2.1. Objetivo general

Implementar un prototipo local que capture señales físicas, realice clasificación ligera en el borde y distribuya la carga de inferencia entre Workers de Raspberry Pi.

2.2. Alcance implementado

- Lectura real de temperatura con DS18B20 y aceleración en tres ejes con ADXL345.
- Comunicación serial USB entre ESP32 y Raspberry Pi Master.
- Servicios persistentes para Master, Worker local, Worker remoto y lector de sensores.
- Panel web con estado de sensores, historial de lecturas, Workers y pruebas manuales.
- Modelo de árbol de decisión exportado a JSON para inferencia local.
- Registro de evidencias y pipeline de Fase 2 para datos físicos.

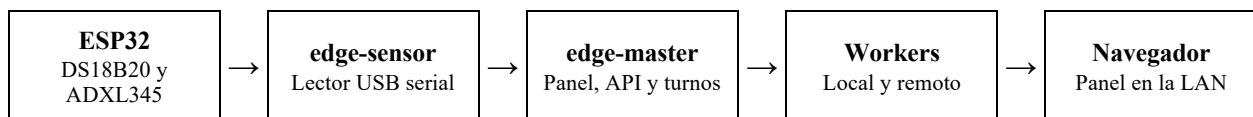
2.3. Fuera de alcance actual

- Control de energía, movimiento o seguridad de la impresora.
- Diagnóstico certificado por fabricante.
- Predicción de vida útil o fecha exacta de falla.
- Modelo entrenado con fallas reales de la Canon G1110.
- Alta disponibilidad del Master mediante un nodo de respaldo.

3. ARQUITECTURA Y FLUJO DE DATOS

Figura 1

Flujo de datos del prototipo



Fuente: elaboración propia.

Tabla 2*Capas y componentes de la arquitectura*

Capa	Componente	Función
Adquisición	ESP32	Lee sensores y publica JSON secuencial
Sensores	DS18B20 y ADXL345	Temperatura y aceleración X, Y, Z
Ingesta	edge-sensor.service	Lee USB serial y actualiza estado local
Coordinación	edge-master.service	Expone panel/API, historial y selección de Worker
Inferencia	edge-worker.service local y remoto	Carga árbol JSON y devuelve clasificación
Visualización	Navegador	Consulta el panel dentro de la LAN
Evidencia	Archivos CSV/JSON	Conserva pruebas y sesiones de Fase 2

3.1. Secuencia de operación

1. La ESP32 toma temperatura y aceleraciones, agrega secuencia y uptime, y escribe un JSON por USB.
2. El lector serial valida el JSON y actualiza runtime/sensor_latest.json.
3. El panel solicita /api/sensor, /api/status e historial para mostrar información reciente.
4. Una prueba manual usa POST /api/analyze con temperatura y vibración.
5. El Master consulta Workers disponibles y aplica selección por turnos.
6. El Worker devuelve estado, disponibilidad del modelo y resultado de la clasificación.
7. El Master registra respuesta, Worker usado, latencia y código HTTP.

3.2. Interfaces principales

Tabla 3*Interfaces principales del sistema*

Ruta o interfaz	Método	Uso
USB serial	JSON por línea	ESP32 hacia Master
/api/sensor	GET	Últimas lecturas físicas y conectividad
/api/status	GET	Disponibilidad de Workers
/api/history	GET	Historial reciente de análisis
/api/analyze	POST	Clasificación manual de temperatura y vibración
/health	GET	Estado de cada Worker

4. HARDWARE E INTEGRACIÓN FÍSICA

Tabla 4

Elementos de hardware y validación realizada

Elemento	Interfaz	Validación realizada
ESP32 DevKit	USB serial	Publica JSON con secuencia, tiempo, temperatura y ejes
DS18B20	OneWire	Sonda detectada y lecturas válidas de temperatura
ADXL345	I2C	Dirección 0x53, DEVID 0xE5, POWER_CTL 0x08
Raspberry Pi Master	USB, red, almacenamiento	Recibe sensores, sirve panel y ejecuta Worker local
Raspberry Pi Worker	Red	Worker remoto con health y modelo listo
Switch/red local	Ethernet o Wi-Fi local	Conectividad entre nodos; validación cableada pendiente

4.1. Formato de telemetría

Cada lectura enviada por la ESP32 tiene el siguiente formato:

```
{ "tipo": "lectura", "dispositivo": "edge-sensor-01", "secuencia": 1485,
  "temperatura_ok": true, "temperatura_c": 33.56, "ax_m_s2": -0.314,
  "ay_m_s2": -0.392, "az_m_s2": 9.101 }
```

El uso de un número de secuencia permite identificar continuidad de lecturas. La aceleración se registra por eje para que, en la Fase 2, sea posible calcular magnitud, desviación estándar, RMS y otras características de vibración.

4.2. Criterio de montaje

El sensor debe fijarse por fuera de la impresora, en una superficie estable, sin abrir circuitos, bloquear ventilación, interferir con piezas móviles ni cruzar la ruta del papel. Cada sesión debe registrar ubicación, actividad y duración.

5. SOFTWARE Y DESPLIEGUE

Tabla 5

Componentes de software

Componente	Tecnología	Responsabilidad
Firmware	Arduino/C++	Lectura de DS18B20 y ADXL345, generación de JSON
Lector serial	Python y pyserial	Reconexión, validación y estado local
Master	Python, Flask y requests	API, panel, historial y balanceo de Workers
Worker	Python y Flask	Carga de tree.json e inferencia
Modelo	DecisionTreeClassifier exportado a JSON	Clasificación ligera reproducible
Servicios	systemd	Inicio automático, reinicio y registro
Versionamiento	Git y repositorio privado	Código, documentos y evidencias

5.1. Servicios esperados

En la Raspberry Master, el estado de los servicios y de la API se verifica con los siguientes comandos:

```
systemctl is-active edge-master.service edge-worker.service \
    edge-sensor.service
curl -sS http://127.0.0.1:8000/api/status
curl -sS http://127.0.0.1:8000/api/sensor
```

En la Raspberry Worker remota se verifica edge-worker.service y la ruta `http://127.0.0.1:5000/health`. El Master mantiene un Worker local en loopback y un Worker remoto configurado en `config/nodes.json`.

6. INTELIGENCIA ARTIFICIAL Y FASE 2

6.1. Modelo inicial

La primera versión fue entrenada con 500 lecturas sintéticas controladas. Utiliza un árbol de decisión para las clases Normal, Advertencia y Crítico. La evaluación inicial reportó 94% de exactitud sobre datos de prueba sintéticos. Este resultado valida la construcción y despliegue del modelo, pero no representa precisión industrial ni entrenamiento sobre fallas reales.

Tabla 6*Propiedades del modelo inicial*

Propiedad	Valor
Algoritmo	DecisionTreeClassifier
Entrada del modelo inicial	temperatura_c y vibracion_mm_s
Salida	Normal, Advertencia o Crítico
Artefacto	model/tree.json
Ejecución	Cada Worker realiza inferencia local

6.2. Fase 2 con datos físicos

La Fase 2 crea sesiones etiquetadas con la telemetría física. Se inició con dos sesiones normal_reposo y 431 lecturas acumuladas. Faltan sesiones normal_impresion para construir un conjunto balanceado. Solo al contar con ambas condiciones se construyen características y se entrena un modelo real para comparación de reposo versus impresión normal. Las actividades previstas son:

1. Capturar al menos tres sesiones normal_reposo y tres normal_impresion.
2. Registrar montaje, fecha, duración, actividad y etiqueta de cada sesión.
3. Construir características de temperatura y aceleración por ventana.
4. Separar datos de entrenamiento y prueba antes de medir desempeño.
5. Documentar métricas y límites antes de sustituir el modelo académico.

7. PRUEBAS Y RESULTADOS

Tabla 7

Pruebas realizadas y resultados obtenidos

Prueba	Evidencia	Resultado
Sonda DS18B20	Detección y lectura serial	Temperatura válida
ADXL345	Diagnóstico I2C	0x53, DEVID 0xE5 y modo medición activo
ESP32	JSON por serial	Campos de temperatura, ejes y secuencia válidos
Lector USB	20 lecturas verificadas	20 de 20 con campos válidos
Servicios Master	systemctl y rutas API	Master, Worker y sensor activos
Worker remoto	/health	model_ready=true y status=ok
Balanceo	Cuatro POST /api/analyze	Local, remoto, local, remoto
Modelo inicial	Prueba de entrenamiento	94% sobre datos sintéticos

7.1. Resultado de la prueba de clúster

Las cuatro solicitudes enviadas al Master produjeron la siguiente salida:

```
Worker: edge-worker-local | Estado: Normal
Worker: edge-worker1      | Estado: Normal
Worker: edge-worker-local | Estado: Normal
Worker: edge-worker1      | Estado: Normal
```

La alternancia prueba el algoritmo de distribución por turnos. No debe interpretarse como una prueba de rendimiento máximo ni como tolerancia ante caída del Master.

8. INCIDENCIAS, RIESGOS Y MITIGACIÓN

Tabla 8

Incidentes, riesgos y medidas de mitigación

Situación	Causa o interpretación	Mitigación
DS18B20 sin lectura inicial	Conexión física incompleta	Revisar cableado y confirmar detección antes de integrar
ADXL345 con ceros	I2C inestable o modo de medición no activado	Validar 0x53, DEVID y POWER_CTL
USB intermitente	Alimentación inestable de Raspberry/USB	Fuente directa estable y cable de calidad
Worker remoto con desconexiones	Subvoltaje indicado por throttling	Cambiar fuente y cable; confirmar valor 0x0
Falla Worker remoto	Nodo no disponible	Master usa Worker local mientras exista
Falla Master	Punto único de coordinación	Fase futura: Master de respaldo y réplica de historial
Datos físicos insuficientes	Fase 2 incompleta	Completar sesiones normales antes de entrenar

8.1. Riesgo de interpretación

El panel muestra telemetría real, pero los estados del modelo inicial provienen de datos sintéticos. Durante la demostración se debe comunicar esta diferencia: la adquisición física está validada; la recalibración del modelo con datos de la impresora está en proceso.

9. OPERACIÓN Y MANTENIMIENTO

9.1. Inicio de demostración

1. Encender Raspberry Master con fuente estable y conectar la ESP32 por USB.
2. Verificar servicios Master, Worker local y lector de sensores.
3. Encender la Raspberry Worker remota y confirmar /health.
4. Abrir el panel desde un equipo de la misma red.
5. Comprobar que /api/status muestre ambos Workers disponibles.
6. Demostrar lectura real y cuatro solicitudes de análisis para mostrar alternancia.

9.2. Captura de Fase 2

La captura de una sesión de impresión normal se realiza con los siguientes comandos:

```
cd ~/edge_predictive_maintenance
.venv/bin/python training/phase2_real_data.py capture \
    --label normal_impresion --duration 180
```

La captura se ejecuta únicamente cuando el sensor esté seguro y la impresora trabaje de forma normal. No se deben provocar daños para generar datos de falla.

9.3. Control de versiones

Cada cambio de código o configuración debe revisarse con git status y git diff antes de confirmar. Se agregan únicamente archivos deliberados: código, documentación y evidencias seleccionadas. No se publican runtime, entornos virtuales, respaldos automáticos ni secretos.

10. CONCLUSIONES Y PRÓXIMOS PASOS

10.1. Conclusiones

1. El prototipo demuestra una ruta completa desde sensores físicos hasta visualización y clasificación local.
2. La arquitectura separa adquisición, coordinación e inferencia y permite usar dos Workers.
3. La prueba de alternancia confirmó que ambos Workers pueden atender solicitudes del Master.
4. La estabilidad de la alimentación fue determinante para el USB y debe corregirse también en el Worker remoto.
5. El modelo académico permite demostrar el flujo de IA local, mientras que la Fase 2 permitirá evaluar un patrón real de la impresora.

10.2. Próximas actividades

1. Completar las sesiones normal_impresion y documentar el montaje externo.
2. Construir el dataset de Fase 2, entrenar y registrar métricas reales.
3. Instalar fuente/cable adecuados en el Worker remoto y validar throttling 0x0.

4. Conectar ambos nodos al switch/router local, verificar IP y repetir salud y balanceo por Ethernet.
5. Etiquetar en Git la versión demostrada y actualizar evidencias finales.

10.3. Repositorio

Repositorio privado del proyecto: <https://github.com/DidierMejia/edge-predictive-maintenance>