SensorGrid Labs S. de R.L.

"Monitoreo inteligente, decisiones informadas"

PROPUESTA DE DESARROLLO

Sistema de Monitoreo Ambiental IoT

Cocinas de Cafetería Central

Instituto Tecnológico y de Estudios Superiores de Occidente

Materia:	Laboratorio de Desarrollo de Soluciones Tecnológicas
Profesor:	Jorge Higinio Ochoa de León
Equipo:	Renata, Rodrigo, Robert, Fernando
Duración:	12 semanas
Fecha:	27 de enero 2026
Versión:	1.0

Índice

1. Resumen Ejecutivo

1.1 Objetivo del Proyecto

Desarrollar e implementar un sistema de monitoreo IoT para las cocinas de la Cafetería Central del ITESO que permita:

• Monitorear temperatura de refrigeradores y congeladores en tiempo real

• Detectar y alertar sobre puertas abiertas por tiempo prolongado

• Medir calidad del aire (COVs, PM2.5, temperatura ambiente, humedad)

• Generar alertas automáticas vía Telegram ante anomalías

• Visualizar datos históricos en dashboard web con autenticación

• Generar reportes para cumplimiento de normativas sanitarias (NOM-251-SSA1-2009)

1.2 Propuesta de Valor

El sistema SensorGrid permite a los administradores de la cafetería tomar decisiones informadas basadas en datos objetivos, reduciendo el riesgo de pérdida de alimentos por fallas en refrigeración, mejorando las condiciones de trabajo del personal de cocina, y generando evidencia documentada para auditorías sanitarias.

1.3 Diferenciadores Clave

Aspecto	Ventaja SensorGrid
Conectividad	Red LoRaWAN independiente de infraestructura ITESO, con cobertura de largo alcance y bajo consumo energético
Arquitectura	100% cloud-native, sin servidor físico local, máxima escalabilidad y disponibilidad 99.9%
Costo operativo	Servicios cloud en tiers gratuitos, solo costo de SIM de datos (~$200 MXN/mes)
Escalabilidad	Arquitectura preparada para expandirse a otras cocinas, campus, o clientes comerciales

2. Alcance del Proyecto

2.1 Incluido en el Alcance

• Monitoreo de temperatura de refrigeradores y congeladores

• Detección de estado de puertas (abierta/cerrada)

• Medición de calidad del aire en áreas de cocina (COVs, PM2.5, PM10)

• Medición de condiciones ambientales (temperatura, humedad)

• Dashboard web con autenticación para visualización de datos

• Sistema de alertas en tiempo real vía Telegram

• Retención de datos históricos por 90 días

• Documentación técnica y manual de usuario

2.2 Excluido del Alcance

• Monitoreo del área de comensales (asignado a otro equipo)

• Integración con sistemas HVAC existentes

• Monitoreo de consumo energético de equipos

• Aplicación móvil nativa (se usa dashboard web responsive)

2.3 Variables a Monitorear

#	Variable	Unidad	Propósito	Sensor
1	Temperatura Refrigerador	°C	Cadena de frío, seguridad alimentaria	DS18B20 Waterproof
2	Estado de Puerta	Binario	Alertas de puerta abierta	Reed Switch
3	COVs / TVOC	Índice	Humos de cocina, productos limpieza	SGP40
4	PM2.5	µg/m³	Partículas finas de humo	PMS5003
5	PM10	µg/m³	Partículas gruesas, polvo	PMS5003
6	Temperatura Ambiente	°C	Confort térmico del personal	BME280
7	Humedad Relativa	% RH	Condensación, confort	BME280

3. Stack Tecnológico

3.1 Resumen del Stack

Capa	Tecnología	Justificación
Microcontrolador	ESP32 + LoRa (TTGO LoRa32)	WiFi+BLE+LoRa integrado, bajo costo, gran comunidad
Comunicación Local	LoRa 915 MHz	Largo alcance (1-2 km), bajo consumo, sin licencia
Protocolo IoT	LoRaWAN 1.0.3	Estándar industrial, seguridad AES-128
Network Server	The Things Network v3	Gratuito, bien documentado, comunidad activa
Gateway	Dragino LPS8	8 canales, económico, compatible TTN
Conectividad WAN	Router 4G LTE + SIM	Independiente de red ITESO, modular
Backend/Ingesta	Cloudflare Workers	TypeScript, serverless, 100k req/día gratis
Base de Datos	InfluxDB Cloud	Optimizada para time series, tier gratuito
Alertas	Telegram Bot API	Gratuito, notificaciones push, fácil
Dashboards	Grafana Cloud	Estándar industria, autenticación, gratis
DNS/SSL	Cloudflare	Gratuito, CDN, protección DDoS

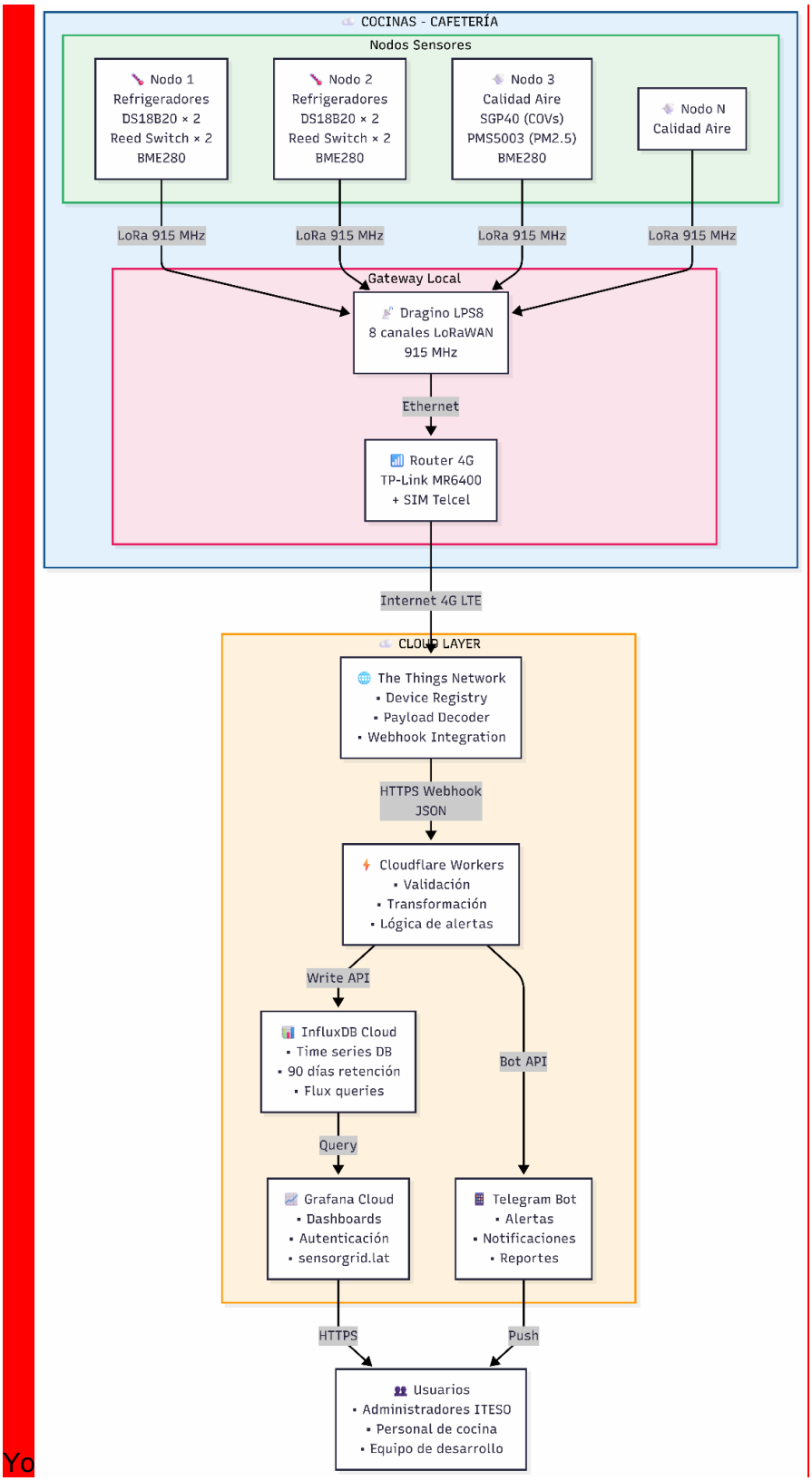
3.2 Lenguajes y Frameworks

Componente	Lenguaje	Framework/Herramientas
Firmware (Nodos)	C++ (Arduino)	PlatformIO, LMIC LoRaWAN, OneWire, Adafruit libs
Backend (Workers)	TypeScript	Cloudflare Workers, Wrangler CLI, Hono (router)
Queries DB	Flux	InfluxDB Query Language
Dashboards	N/A (visual)	Grafana Panels, Variables, Alerting
Infraestructura	YAML	Wrangler.toml, GitHub Actions

4. Arquitectura del Sistema

4.1 Diagrama de Arquitectura

La arquitectura sigue un modelo cloud-native sin servidor físico local:



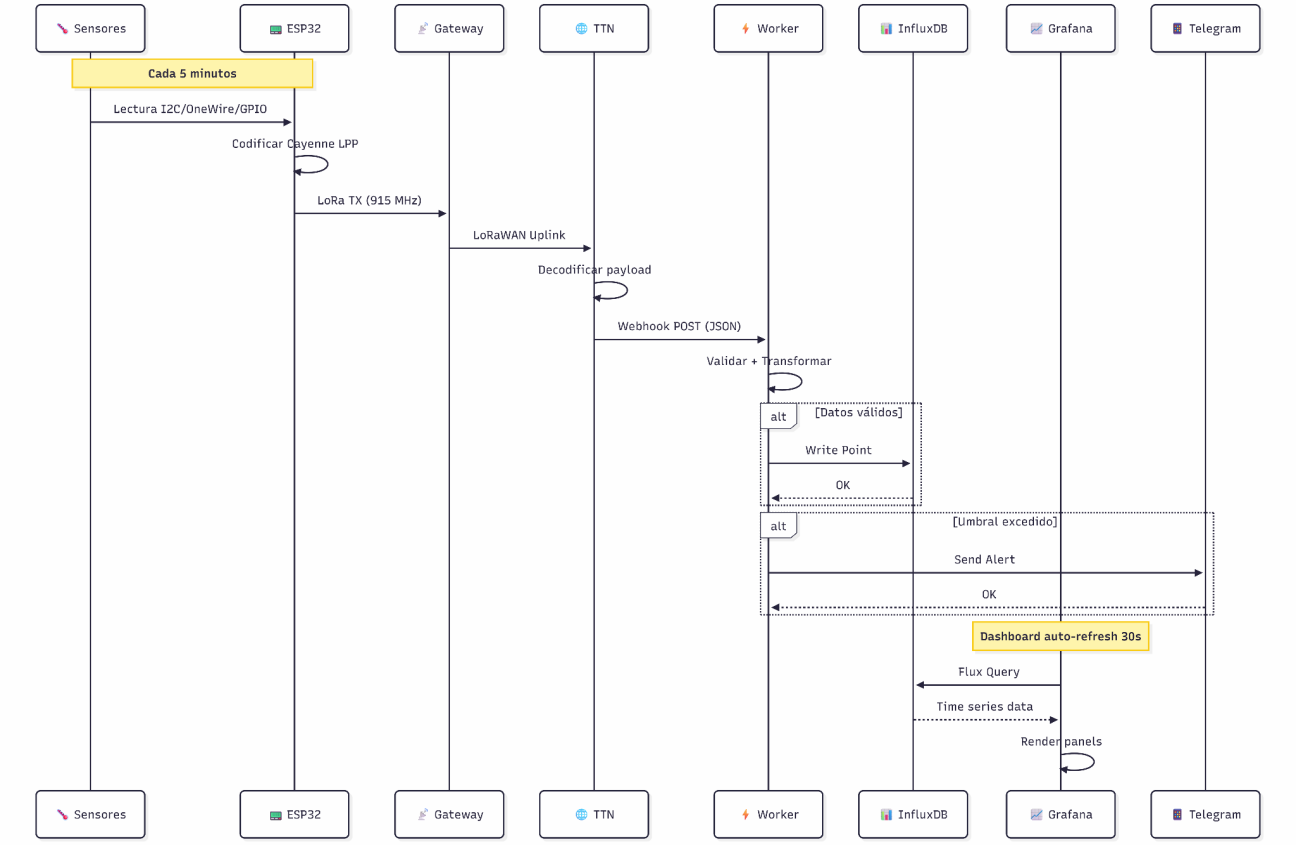
Yo

4.2 Flujo de Datos

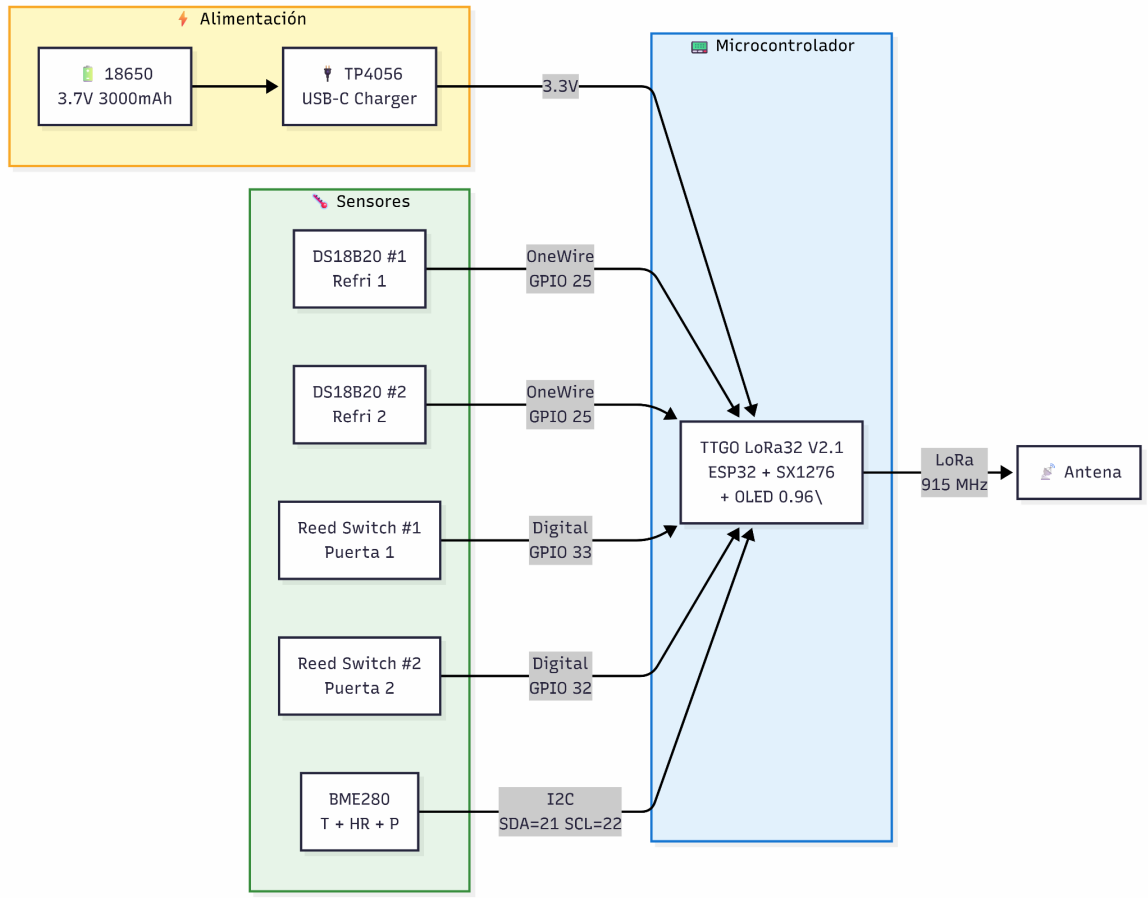
1. CAPTURA: Sensores envían datos al ESP32 cada 5 minutos via I2C/OneWire/GPIO

2. CODIFICACIÓN: ESP32 empaqueta datos en formato Cayenne LPP (Low Power Payload)
3. TRANSMISIÓN: Paquete se envía vía LoRa 915 MHz al Gateway
4. ROUTING: Gateway reenvía a TTN vía internet (Router 4G)
5. PROCESAMIENTO: TTN dispara webhook → Cloudflare Worker procesa
6. ALMACENAMIENTO: Worker escribe a InfluxDB Cloud
7. ALERTAS: Si se exceden umbrales → notificación Telegram
8. VISUALIZACIÓN: Grafana consulta InfluxDB y renderiza dashboards

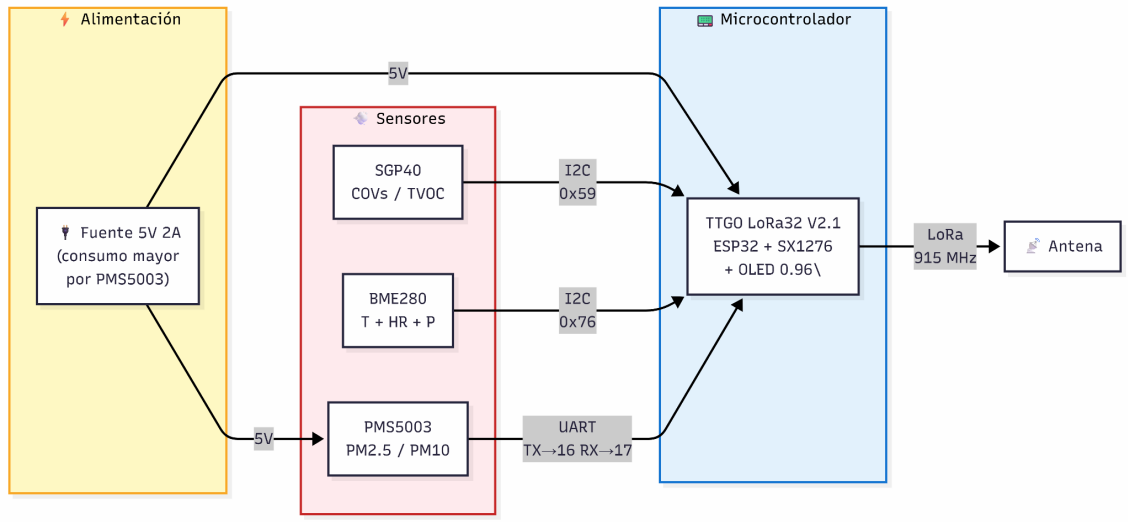
4.3 Diagrama de Secuencia Global



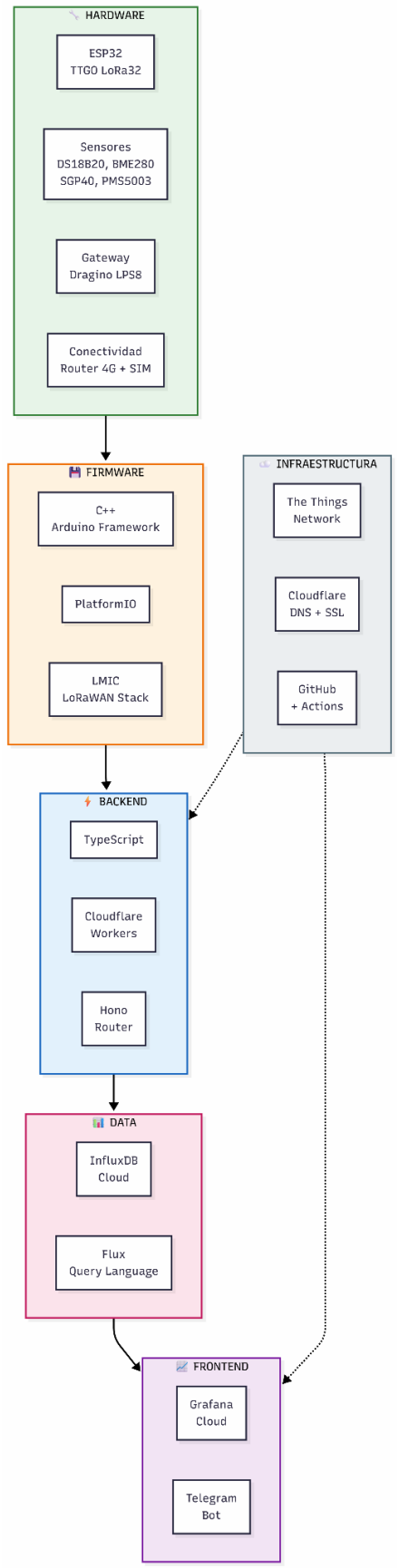
4.4 Arquitectura de Hardware - Nodo Refrigerador



4.4. Arquitectura de Hardware - Nodo Calidad del Aire



4.5 Arquitectura del Stack Tecnológico



5. Justificación de Decisiones Técnicas

Esta sección documenta el razonamiento detrás de cada decisión técnica importante, considerando las habilidades del equipo, los requisitos del proyecto, y la visión a largo plazo del producto.

5.1 ¿Por qué arquitectura 100% Cloud sin servidor local?

Factor	Justificación
Experiencia del equipo	El equipo tiene amplia experiencia en desarrollo cloud (Cloudflare Workers, Vercel, TypeScript) y limitada experiencia en administración de servidores físicos o Raspberry Pi. Aprovechar fortalezas existentes reduce riesgo y tiempo de desarrollo.
Formación académica	El equipo es de Ingeniería en Sistemas Computacionales, no de Ingeniería Electrónica. La programación de microcontroladores es nueva; agregar administración de servidor local aumentaría la complejidad innecesariamente.
Requisito de URL pública	El dashboard debe ser accesible desde internet para profesores, administradores y stakeholders. Con arquitectura cloud, esto viene incluido. Con servidor local requeriría configuración de IP pública, SSL, DNS dinámico, etc.
Escalabilidad comercial	Si el proyecto evoluciona a producto comercial, 'no requiere servidor local' es un diferenciador importante. Los clientes solo necesitan instalar hardware y conectar a internet.
Disponibilidad	Servicios cloud ofrecen SLA de 99.9%+ uptime. Un Raspberry Pi local depende de la estabilidad eléctrica y de red de ITESO.
Mantenimiento	Sin servidor físico = sin actualizaciones de sistema operativo, sin backups locales, sin riesgo de falla de microSD, sin preocuparse por seguridad del servidor.
Costo	Eliminar Raspberry Pi + accesorios ahorra ~$3,230 MXN. Los servicios cloud usados tienen tier gratuito suficiente para el proyecto.

5.2 ¿Por qué LoRaWAN en lugar de WiFi directo?

Aspecto	WiFi Directo	LoRaWAN
Alcance	30-50 metros indoor	1-2 km, penetra paredes
Consumo energía	~100-300 mA activo	~20-50 mA TX, µA sleep
Red requerida	Acceso a WiFi ITESO (negado)	Red propia independiente
Escalabilidad	Limitada por APs	1 gateway = miles de nodos
Expectativas académicas	Básico	Alineado con alcance del proyecto de la materia

5.3 ¿Por qué Router 4G separado en lugar de Gateway con LTE integrado?

Aspecto	Gateway LTE Integrado	Router 4G + Gateway
Disponibilidad MX	Difícil, importación 2-4 sem	Inmediata, tiendas locales
Costo si falla	Reemplazar todo: $5,000+	Solo router: $1,800
Debugging	Limitado a logs internos	Conectar laptop, aislar problemas
Flexibilidad	Solo gateway	Puertos extra para futuro
Experiencia equipo	Interfaz propietaria nueva	Configuración de router conocida

5.4 ¿Por qué Cloudflare Workers para el backend?

• Experiencia previa: El líder del equipo tiene experiencia desplegando Workers en producción

• TypeScript nativo: Lenguaje principal del equipo, sin curva de aprendizaje

• Serverless: Sin servidores que administrar, escala automáticamente

• Tier gratuito generoso: 100,000 requests/día, suficiente para proyecto y más allá

• Latencia global: Edge computing, respuesta rápida desde cualquier ubicación

• Integración Cloudflare: DNS, SSL, KV storage, todo en un ecosistema

5.5 ¿Por qué Telegram para alertas?

• Gratuito y sin límites: Bot API no tiene costo

• Notificaciones push: Llegan instantáneamente al celular

• Grupos: Se pueden crear grupos para diferentes roles (admin, cocina, mantenimiento)

• Fácil implementación: HTTP POST con JSON, desde cualquier lenguaje

• Escalable a WhatsApp: Si se requiere en el futuro, la lógica es similar

6. Diseño de Hardware

6.1 Tipos de Nodos

Se diseñan dos tipos de nodos especializados:

Nodo Tipo A: Monitoreo de Refrigeradores

Función: Monitorear 2 refrigeradores (temperatura + puerta) y ambiente de la zona

Sensores: 2× DS18B20 (temperatura), 2× Reed Switch (puerta), 1× BME280 (ambiente)

Alimentación: Batería 18650 + cargador TP4056

Cantidad estimada: 2-4 nodos (según número de refrigeradores)

Nodo Tipo B: Calidad del Aire

Función: Monitorear calidad del aire en área de cocina

Sensores: 1× SGP40 (COVs), 1× PMS5003 (PM2.5/PM10), 1× BME280 (T+HR)

Alimentación: Fuente 5V (consumo mayor por ventilador del PMS5003)

Cantidad estimada: 1-2 nodos (uno por cocina activa)

6.2 Conexiones Eléctricas

Componente	Pin ESP32	Protocolo	Notas
DS18B20 (×2)	GPIO 25	OneWire	Mismo bus, resistor 4.7kΩ pull-up
Reed Switch 1	GPIO 33	Digital	Pull-up interno habilitado
Reed Switch 2	GPIO 32	Digital	Pull-up interno habilitado
BME280	SDA=21, SCL=22	I2C	Dirección 0x76
SGP40	SDA=21, SCL=22	I2C	Dirección 0x59, mismo bus que BME280
PMS5003	TX→16, RX→17	UART	Serial2, 9600 baud
LoRa SX1276	Interno	SPI	Ya conectado en TTGO
OLED 0.96"	Interno	I2C	Ya conectado en TTGO

7. Diseño de Software

7.1 Firmware (Nodos ESP32)

Lenguaje: C++ con framework Arduino

IDE: PlatformIO (VS Code)

7.2 Backend (Cloudflare Workers)

Lenguaje: TypeScript

Framework: Hono (router minimalista)

Deploy: Wrangler CLI + GitHub Actions

7.3 Repositorio GitHub

Organización: Monorepo con workspaces

Branches: main (producción), dev (integración), feature/* (desarrollo)

CI/CD: GitHub Actions para lint, test, y deploy automático

Protección: main requiere PR con al menos 1 aprobación

7.4 Estructura del Monorepo

```
sensorgrid/
│
├── README.md
├── LICENSE
├── .gitignore
├── .github/
│   └── workflows/
│       ├── firmware-ci.yml
│       └── backend-deploy.yml
│
├── docs/
│   ├── arquitectura.md
│   ├── manual-usuario.md
│   ├── configuracion-cloud.md
│   └── troubleshooting.md
│
├── firmware/                    # Código para nodos ESP32
│   ├── platformio.ini
│   ├── include/
│   │   ├── config.h             # Pines, intervalos, umbrales
│   │   └── credentials.h          # Keys LoRaWAN (gitignored)
│   ├── lib/
│   │   ├── SensorManager/
```

```
│   │   │   ├── SensorManager.h
│   │   │   └── SensorManager.cpp
│   │   ├── LoRaManager/
│   │   │   ├── LoRaManager.h
│   │   │   └── LoRaManager.cpp
│   │   └── PowerManager/
│   │       ├── PowerManager.h
│   │       └── PowerManager.cpp
│   ├── src/
│   │   ├── main.cpp              # Entry point
│   │   ├── node_refri.cpp        # Lógica nodo refrigerador
│   │   ├── node_airq.cpp         # Lógica nodo calidad aire
│   │   └── payload.cpp           # Codificación Cayenne LPP
│   └── test/
│       └── test_sensors.cpp
│
├── backend/                      # Cloudflare Workers
│   ├── wrangler.toml             # Configuración Cloudflare
│   ├── package.json
│   ├── tsconfig.json
│   ├── src/
│   │   ├── index.ts              # Entry point, router Hono
│   │   ├── handlers/
│   │   │   ├── webhook.ts        # Recibe datos de TTN
│   │   │   ├── api.ts            # Endpoints REST
│   │   │   └── health.ts         # Health check
│   │   ├── services/
│   │   │   ├── influx.ts         # Cliente InfluxDB
│   │   │   ├── telegram.ts       # Bot de alertas
│   │   │   └── decoder.ts        # Decodifica Cayenne LPP
│   │   ├── utils/
│   │   │   ├── validation.ts     # Validación de payloads
│   │   │   └── thresholds.ts     # Lógica de umbrales/alertas
│   │   └── types/
│   │       └── index.ts          # TypeScript types
│   └── test/
│       ├── webhook.test.ts
│       └── thresholds.test.ts
│
├── dashboards/                   # Configuración Grafana
│   ├── provisioning/
│   │   └── dashboards/
│   │       ├── overview.json
│   │       ├── refrigeradores.json
│   │       └── calidad-aire.json
│   └── alerts/
│       └── alert-rules.yaml
│
└── hardware/                     # Documentación de hardware
    ├── esquematicos/
    │   ├── nodo-refri.png
    │   └── nodo-airq.png
    ├── carcasas/
    │   ├── carcasa-nodo.stl
```

```
│   └── soporte-gateway.stl
└── BOM.xlsx                    # Bill of Materials
```

8. Servicios Cloud

8.1 Configuración de Servicios

Servicio	Plan	Límites Tier Gratuito	Uso Estimado
The Things Network	Community (gratis)	Fair use, 30s uplink min	~300 msgs/día
Cloudflare Workers	Free (gratis)	100,000 req/día	~500 req/día
InfluxDB Cloud	Free (gratis)	30 días retención	Upgrade a pagado si >30 días
Grafana Cloud	Free (gratis)	10,000 series, 14 días	~50 series activas
Telegram Bot	Gratis	Sin límites prácticos	~10-50 alertas/día

8.2 Costos Operativos Mensuales

Concepto	Costo Mensual	Notas
SIM de datos (Telcel 6GB)	$200 MXN	Único costo recurrente
The Things Network	$0	Tier gratuito
Cloudflare Workers	$0	Tier gratuito
InfluxDB Cloud	$0 - $100 MXN	Gratis 30 días, ~$5 USD si más
Grafana Cloud	$0	Tier gratuito
Dominio sensorgrid.lat	~$25 MXN	~$300 MXN/año prorrateado
TOTAL MENSUAL	**$225 - $325 MXN**	

9. Metodología de Desarrollo

9.1 Framework: Scrum Adaptado

Se utilizará Scrum adaptado para un equipo académico de 4 personas:

• Sprints de 2 semanas (6 sprints en total)

• Daily standups asíncronos (mensaje en grupo de WhatsApp)

• Sprint Planning al inicio de cada sprint

• Sprint Review + Retrospectiva al final

• Tablero Kanban en Notion (https://www.notion.so/Laboratorio-de-Desarrollo-de-Soluciones-Tecnol-gica-2eed493bbf5a80069d4ef635295db72a?source=copy_link)

9.2 Ceremonias

Ceremonia	Frecuencia	Duración / Descripción
Sprint Planning	Cada 2 semanas	30-60 minutos. Definir objetivos y tareas del sprint.
Daily Standup	Diario	Asíncrono. Cada quien reporta: ¿Qué hice? ¿Qué haré? ¿Bloqueos?
Sprint Review	Cada 2 semanas	30-60 min. Demo de lo construido, feedback.
Retrospectiva	Cada 2 semanas	30 min. ¿Qué salió bien? ¿Qué mejorar? ¿Acciones?

9.3 Definición de Done

Una tarea se considera DONE cuando:

• El código está completo y funcional

• El código tiene tests (cuando aplica)

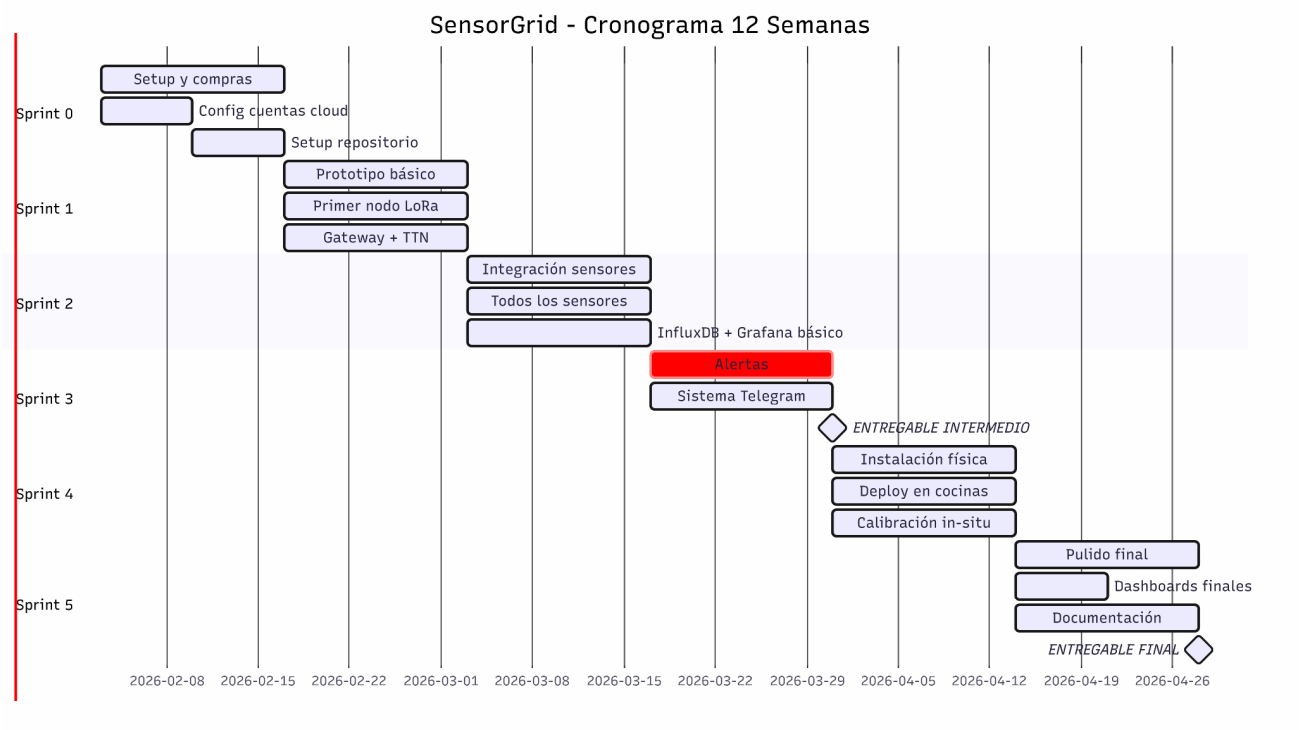
• El código pasó code review (PR aprobado)

• El código está mergeado a dev

• La documentación está actualizada (si aplica)

10. Cronograma de 12 Semanas

Sprint	Semanas	Objetivos / Entregables
0	1-2	SETUP: Compra de materiales, configuración de cuentas cloud (TTN, Cloudflare, InfluxDB, Grafana), setup de repositorio GitHub, configuración de entorno de desarrollo PlatformIO
1	3-4	PROTOTIPO: Primer nodo funcional enviando datos de prueba via LoRaWAN a TTN. Gateway configurado y conectado. Worker recibiendo webhooks.
2	5-6	SENSORES: Integración de todos los sensores (DS18B20, Reed, BME280, SGP40, PMS5003). Datos reales llegando a InfluxDB. Dashboard básico en Grafana.
3	7-8	ALERTAS: Sistema de alertas Telegram funcionando. Umbrales configurables. Múltiples nodos operando. ENTREGABLE INTERMEDIO.
4	9-10	INSTALACIÓN: Despliegue físico en cocinas de ITESO. Carcasas impresas/montadas. Calibración de sensores in-situ. Pruebas de cobertura LoRa.
5	11-12	PULIDO: Dashboards finales, documentación técnica, manual de usuario, pruebas de estrés, corrección de bugs. ENTREGABLE FINAL.



11. Equipo y Roles

Nombre	Rol	Responsabilidades
Fernando	Team Lead / DevOps	Coordinación del equipo, arquitectura, desarrollo de Cloudflare Workers, integraciones cloud, code reviews
Renata	A definir	Asignar según fortalezas durante Sprint 0
Rodrigo	A definir	Asignar según fortalezas durante Sprint 0
Robert	A definir	Asignar según fortalezas durante Sprint 0

12. Riesgos y Mitigaciones

Riesgo	Prob.	Impacto	Mitigación
Demora en entrega de materiales	Media	Alto	Priorizar proveedores locales (ML, Steren). Pedir con margen de 2 semanas.
Dificultad con LoRaWAN (nuevo)	Media	Medio	Usar TTN que tiene excelente documentación. Empezar con ejemplo básico.
Cobertura LoRa insuficiente	Baja	Alto	Pruebas de cobertura tempranas. Antenas de mayor ganancia como backup.
Acceso restringido a cocinas	Baja	Alto	Coordinar con profesor. Reunión del jueves para establecer contactos.
Sensores defectuosos	Media	Bajo	Comprar 1-2 unidades extra de sensores críticos como respaldo.
Límites de tier gratuito	Baja	Bajo	Monitorear uso. Los límites son generosos para este proyecto.

13. Entregables

13.1 Entregable Intermedio (Semana 8)

• Sistema funcional con al menos 2 nodos transmitiendo datos reales

• Dashboard básico mostrando métricas en tiempo real

• Sistema de alertas Telegram operativo

• Documentación de arquitectura actualizada

13.2 Entregable Final (Semana 12)

• Sistema completo instalado en cocinas de ITESO

• Dashboard profesional con autenticación

• Alertas configuradas según umbrales de normativas

• Documentación técnica completa

• Manual de usuario para administradores

• Repositorio GitHub con código fuente

• Presentación final del proyecto

• Video demo del sistema funcionando

13.3 Documentación

• README.md con instrucciones de instalación y uso

• Documento de arquitectura técnica

• Diagramas de conexión de hardware

• Manual de configuración de servicios cloud

• Manual de usuario para dashboard

• Runbook de operaciones (troubleshooting)