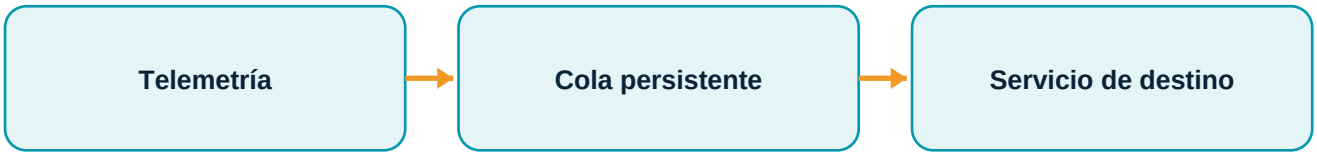
Capítulo 22 · Docker, almacenamiento SSD y búfer SQLite WAL

Conceptos clave

Usa almacenamiento persistente y conserva cada registro hasta recibir confirmación. Prueba el reinicio con registros pendientes.



GUÍA PASO A PASO DE LABORATORIO — MÓDULO 07

Persistencia en Series Temporales, Orquestación con Docker Compose, Node-RED, InfluxDB y Grafana

Curso: Internet de las Cosas (IoT) con Arduino y Raspberry Pi — Ingeniería Mecatrobótica

Módulo: 07 — Capa de Almacenamiento Histórico, Contenedores ARM64 y Observabilidad Industrial

---

1. Objetivos de Ingeniería del Laboratorio

- Montar una unidad de estado sólido externa (`SSD USB 3.0`) en la Raspberry Pi y configurar el reloj de tiempo real por hardware (`DS3231` en el bus `I2C1` dirección `0x68`) junto con el detector optoacoplado de corte eléctrico (`PC817` en `GPIO 17`).
- Desplegar mediante `Docker Compose` una arquitectura multicontenedor aislada sobre Linux ARM64 compuesta por `Eclipse Mosquitto 2` , `InfluxDB 2.7` , `Node-RED` y `Grafana OSS` .
- Programar y auditar el demonio en Python de ingesta resiliente **Store-and-Forward** (`ingestor_store_and_forward_sqlite_influxdb.py`) que respalda muestras en una cola `SQLite3` en modo `WAL` cuando la base de datos principal se encuentra en mantenimiento o desconectada.
- Configurar políticas de retención de datos (`30 días` a `1 Hz`) y tareas automáticas de agregación o *Downsampling* en lenguaje `Flux` (`300:1` hacia ventanas de `5 minutos` con retención de `365 días`).
- Construir paneles de supervisión industrial en `Grafana` cumpliendo la norma de ergonomía visual **ANSI/ISA-101** y flujos de alerta con histéresis en **Node-RED**.

---

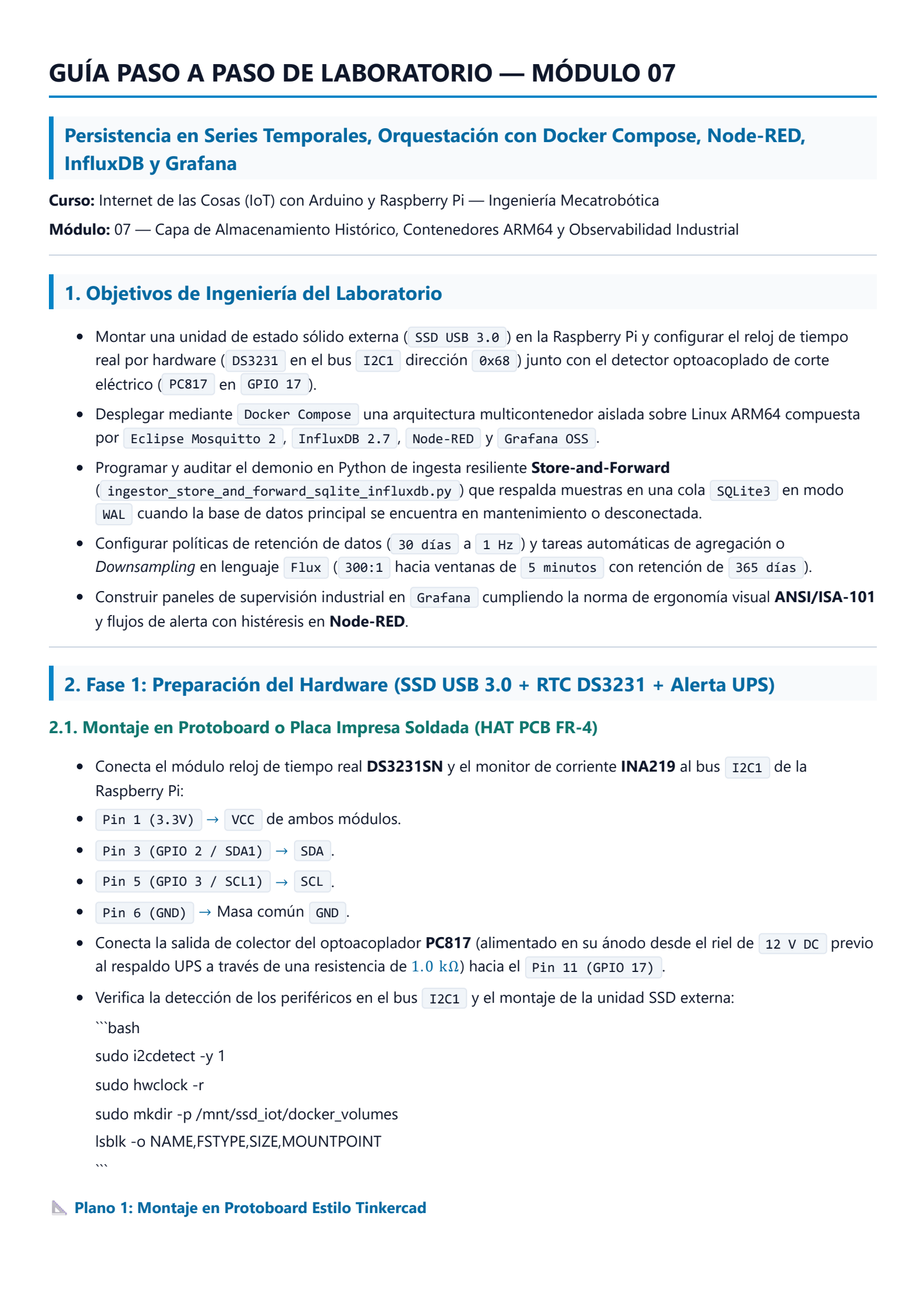
2. Fase 1: Preparación del Hardware (SSD USB 3.0 + RTC DS3231 + Alerta UPS)

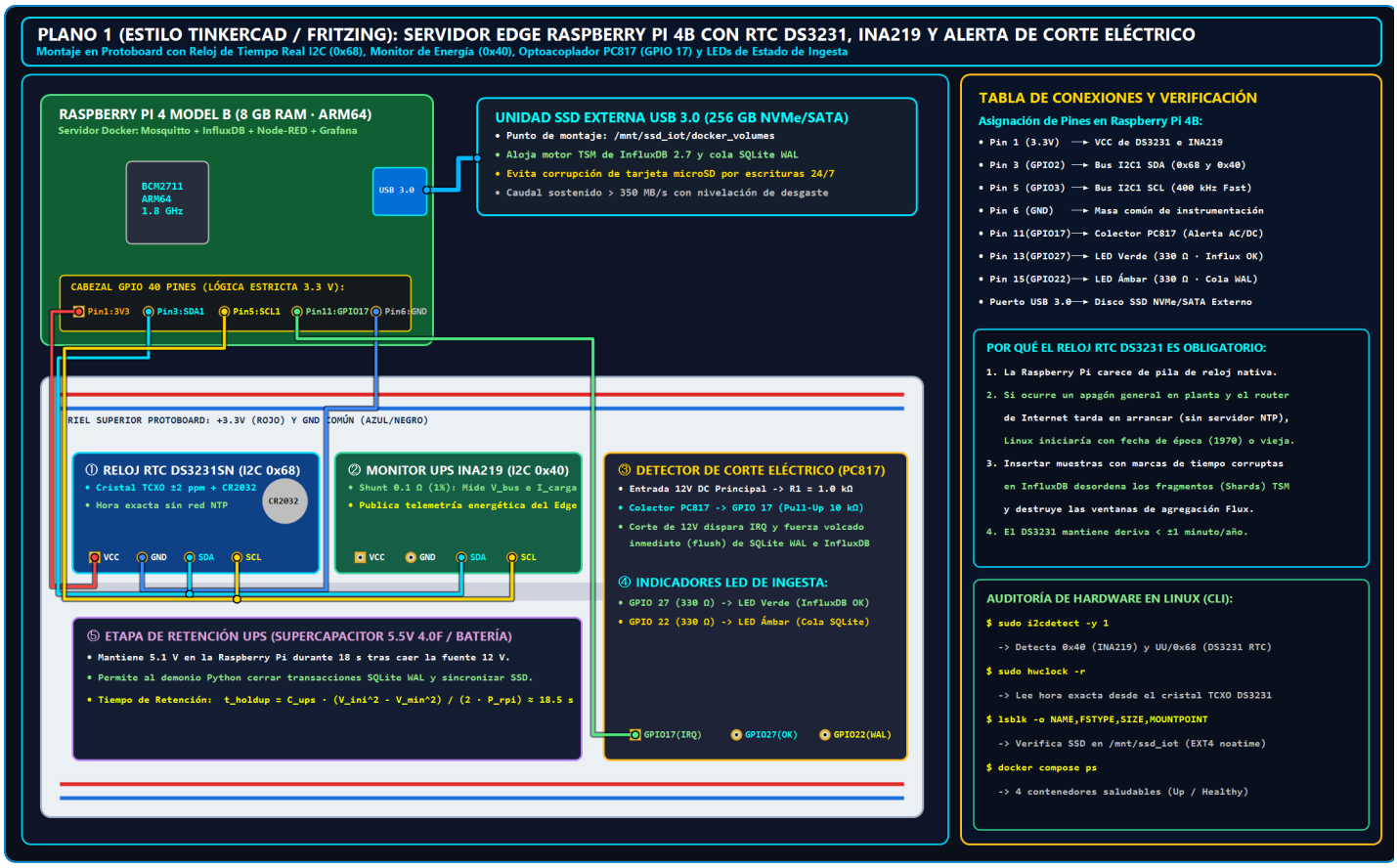
2.1. Montaje en Protoboard o Placa Impresa Soldada (HAT PCB FR-4)

- Conecta el módulo reloj de tiempo real **DS3231SN** y el monitor de corriente **INA219** al bus `I2C1` de la Raspberry Pi:
- `Pin 1 (3.3V)` → `VCC` de ambos módulos.
- `Pin 3 (GPIO 2 / SDA1)` → `SDA` .
- `Pin 5 (GPIO 3 / SCL1)` → `SCL` .
- `Pin 6 (GND)` → Masa común `GND` .
- Conecta la salida de colector del optoacoplador **PC817** (alimentado en su ánodo desde el riel de `12 V DC` previo al respaldo UPS a través de una resistencia de $1.0\ \text{k}\Omega$) hacia el `Pin 11 (GPIO 17)` .
- Verifica la detección de los periféricos en el bus `I2C1` y el montaje de la unidad SSD externa:

  ```bash
  sudo i2cdetect -y 1
  sudo hwclock -r
  sudo mkdir -p /mnt/ssd_iot/docker_volumes
  lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT
  ```

📐 Plano 1: Montaje en Protoboard Estilo Tinkercad





---

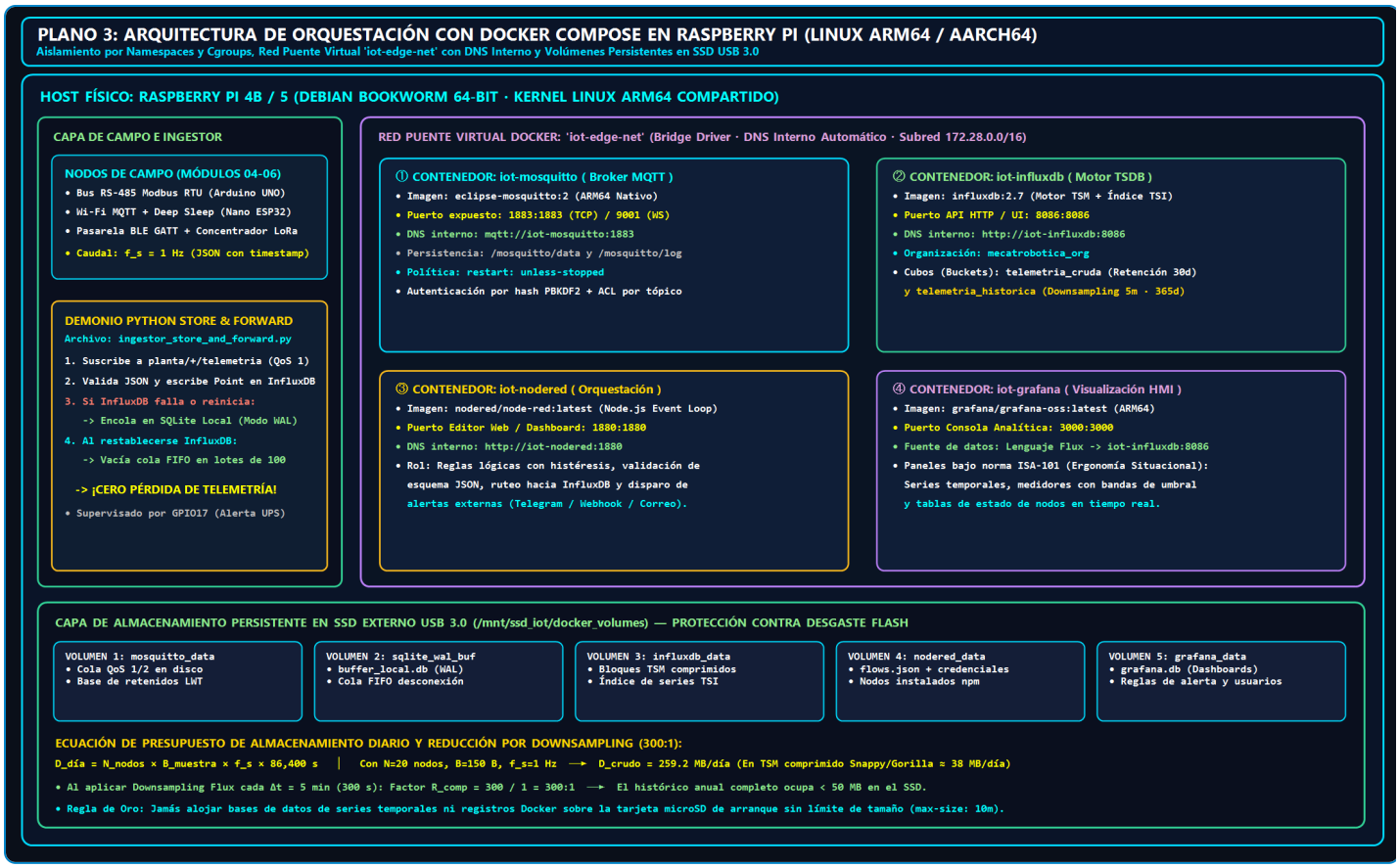
3. Fase 2: Despliegue de la Pila con Docker Compose en ARM64

- Copia el archivo `docker-compose.yml` incluido en la carpeta `codigo_fuente/` hacia `/opt/mecatrobotica/stack_v07/docker-compose.yml`.
- Levanta los cuatro servicios en segundo plano y verifica su estado de salud:

  ```bash
  cd /opt/mecatrobotica/stack_v07
  docker compose up -d
  docker compose ps
  ```

- Comprueba que los cuatro contenedores (`iot-mosquitto`, `iot-influxdb`, `iot-nodered` e `iot-grafana`) reporten estado `Up` y que `iot-influxdb` alcance el estado `healthy` en el puerto `8086`.

📐 Plano 3: Arquitectura Docker Compose en Raspberry Pi ARM64



---

4. Fase 3: Prueba de Fuego del Ingestor Store-and-Forward (SQLite WAL + InfluxDB)

- Instala las bibliotecas cliente en el entorno virtual de la Raspberry Pi:

  ```bash
  pip install paho-mqtt influxdb-client
  ```

- Ejecuta el demonio `ingestor_store_and_forward_sqlite_influxdb.py`.
- **Simulación de caída de base de datos**: Detén temporalmente el contenedor de InfluxDB mientras los nodos continúan publicando telemetría:

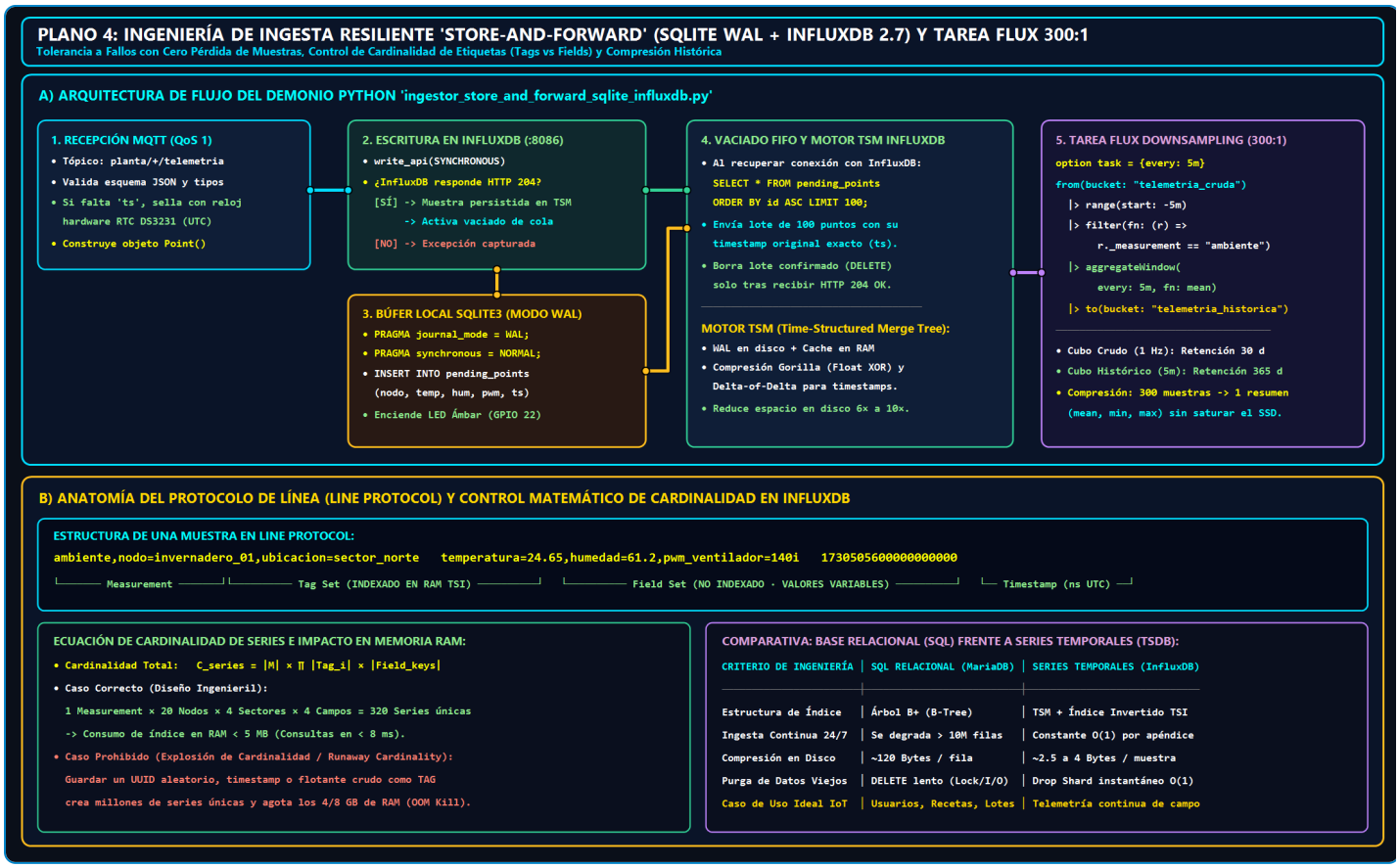
  ```bash
  docker stop iot-influxdb
  ```

- Observa en la consola cómo el demonio captura la excepción de conexión y desvía inmediatamente cada muestra entrante hacia la tabla `pending_points` de `buffer_local_wal.db` (`PRAGMA journal_mode=WAL`).
- **Restauración y vaciado por lotes**: Vuelve a iniciar el contenedor:

  ```bash
  docker start iot-influxdb
  ```

- Verifica cómo en el siguiente ciclo de recepción el demonio vacía automáticamente la cola FIFO en lotes de `100 muestras` (`ORDER BY id ASC LIMIT 100`), insertando toda la serie histórica con sus marcas de tiempo originales intactas.

📐 Plano 4: Arquitectura Store-and-Forward y Tarea Flux 300:1



5. Fase 4: Tarea de Downsampling en Flux y Paneles ISA-101 en Grafana

- Accede a la interfaz web de InfluxDB (`http://<IP_RASPBERRY>:8086`), crea el segundo cubo llamado `telemetria_historica` con retención de `365 días` y registra el script `tarea_downsampling_flux_300a1.flux` en la sección **Tasks**.
- Accede a **Node-RED** (`http://<IP_RASPBERRY>:1880`), importa el archivo `flujo_orquestacion_alertas_nodered.json` y despliega el flujo con validación de esquema JSON e histéresis térmica ($T_{on} = 30.0\ ^\circ C$, $T_{off} = 28.5\ ^\circ C$).
- Accede a **Grafana** (`http://<IP_RASPBERRY>:3000`), configura la fuente de datos **InfluxDB** seleccionando el lenguaje **Flux**, la URL interna `http://iot-influxdb:8086` y el token de administración.
- Diseña el panel de supervisión aplicando la jerarquía de tres niveles y la semántica de colores de la norma **ANSI/ISA-101**.

📐 Plano 5: Ergonomía Visual ISA-101 en Grafana y Flujo Node-RED

PLANO 5: DISEÑO DE INTERFACES HMI BAJO NORMA ISA-101 EN GRAFANA Y FLUJO DE REGLAS EN NODE-RED

Conciencia Situacional para Operadores de Planta, Jerarquía de 3 Niveles, Histéresis de Alerta y Disponibilidad Operativa (SLA)

A) PANEL DE SUPERVISIÓN INDUSTRIAL GRAFANA (:3000) DISEÑADO BAJO ESTÁNDAR ANSI/ISA-101.01

TEMPERATURA PROCESO (NODO 01)

24.6 °C

Banda Nominal: [18.0 .. 28.0 °C] (OK)

HUMEDAD RELATIVA Y PRESIÓN

61.2 %RH | 1013 hPa

Sensor Bosch BME280 (I2C 0x76)

PRE-ALARMA PREVENTIVA (ADVERTENCIA)

29.4 °C (Sector Sur)

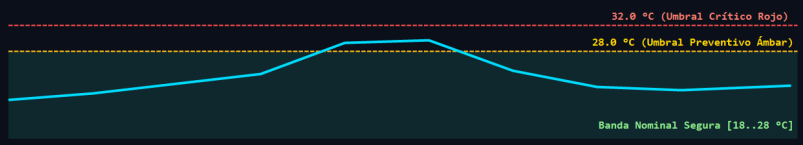
Supera umbral preventivo > 28.0 °C

ALARMA CRÍTICA ISA-18.2 (PRIORIDAD 1)

0 ALARMAS ACTIVAS

Disponibilidad SLA: A = 99.982% (Online)

CURVA DE SERIE TEMPORAL CON BANDA OPERATIVA SOMBREADA (CONSULTA FLUX aggregateWindow)



REGLAS DE ORO DE ERGONOMÍA VISUAL ISA-101:

1. Prohibido el efecto 'Árbol de Navidad': Si todo brilla en rojo, verde y amarillo saturado, el operador sufre fatiga visual y no detecta fallos.
2. Fondos oscuros mate (#0A101C / #121E36) y rejillas de bajo contraste para descanso ocular en turno 24/7.
3. Reserva cromática estricta:
   - Cian / Esmeralda -> Operación dentro de consigna
   - Ámbar (#FFBE1E) -> Desviación preventiva
   - Rojo (#FF5050) -> Alarma crítica que exige acción
4. Incluir siempre unidades físicas (°C, %RH, mA, dBm).

B) FLUJO DE ORQUESTACIÓN EN NODE-RED (:1880) CON HISTÉRESIS DE ALARMA Y CÁLCULO DE DISPONIBILIDAD (SLA)

① MQTT In
planta/+/telemetria
(QoS 1 · JSON)

② JSON Parser &
Schema Validator
(Verifica llaves)

③ Function Node
Filtro Histéresis
(T_on=30, T_off=28.5)

④ RBE / Filter
Report by Exception
(Bloquea spam)

⑤ Salida Dual:
• InfluxDB Batch Out
• Alerta Telegram/MQTT

LÓGICA JAVASCRIPT CON MEMORIA DE CONTEXTO E HISTÉRESIS EN NODE-RED:

```
const T_ALARMA_ON  = 30.0; // Umbral de disparo (°C)
const T_ALARMA_OFF = 28.5; // Umbral de reposición (Banda ΔT = 1.5 °C)
let enAlarma = context.get('enAlarma') || false;
if (!enAlarma && msg.payload.temp >= T_ALARMA_ON) {
    enAlarma = true;  context.set('enAlarma', true);
    return [msg, { payload: { alerta: 'ALTA_TEMP', valor: msg.payload.temp } }];
} else if (enAlarma && msg.payload.temp <= T_ALARMA_OFF) {
    enAlarma = false; context.set('enAlarma', false);
}
```

INGENIERÍA DE CONFIABILIDAD Y DISPONIBILIDAD DEL SISTEMA (MTBF / MTTR):

1. Ecuación de Disponibilidad Estacionaria (Availability):
   A = MTBF / (MTBF + MTTR) = T_operativo / T_total

2. Efecto de Docker 'restart: unless-stopped' + Healthchecks:
   Reduce el Tiempo Medio de Reparación (MTTR) de horas (intervención humana manual) a menos de 3.5 segundos (reinicio automático).

3. Combinado con el búfer SQLite WAL (Store-and-Forward):
   La disponibilidad efectiva de datos alcanza el 100.0% frente a microcortes de red o reinicios de mantenimiento de contenedores.

LISTADO DE MATERIALES (BOM) Y PRESUPUESTO DE ALMACENAMIENTO — MÓDULO 07

Servidor Edge de Series Temporales, Respaldo UPS, Reloj RTC DS3231 y Placa HAT Soldada

Curso: Internet de las Cosas (IoT) con Arduino y Raspberry Pi — Ingeniería Mecatrobótica

Módulo: 07 — Persistencia en Series Temporales, Docker, Node-RED, InfluxDB y Grafana

---

1. Tabla Maestra de Componentes de Hardware (BOM)

Ítem	Designador	Componente / Especificación Técnica	Cantidad	Función en la Arquitectura del Módulo 07
1	`SBC1`	**Raspberry Pi 4 Model B (4 GB / 8 GB RAM)** o Raspberry Pi 5 con disipador de aluminio anodizado	1	Servidor Linux ARM64 para orquestación de los 4 contenedores Docker (`Mosquitto` , `InfluxDB` , `Node-RED` , `Grafana`).
2	`SSD1`	**Unidad de Estado Sólido Externa USB 3.0 (256 GB NVMe/SATA)** con soporte UASP y TRIM	1	Almacenamiento de alta durabilidad para bloques TSM de InfluxDB y cola SQLite WAL (evita destruir la tarjeta microSD).
3	`U1 / BT1`	**Módulo Reloj en Tiempo Real DS3231SN (`I2C 0x68`)** con oscilador TCXO compensado (± 2 ppm) y pila `CR2032`	1	Garantiza marcas de tiempo exactas en nanosegundos UTC tras cortes eléctricos aunque aún no exista sincronización NTP.
4	`U2`	**Sensor de Corriente y Tensión High-Side INA219 (`I2C 0x40`)** con resistencia Shunt Kelvin de $0.100\ \Omega$ (1%)	1	Monitorea en tiempo real la tensión del riel de $5.1\ V$ y el consumo de corriente del servidor Edge y del SSD.
5	`U3`	**Optoacoplador Fototransistor PC817C (DIP-4)** con aislamiento galvánico de $2500\ V_{rms}$	1	Detecta instantáneamente la caída del riel primario de $12\ V\ DC$ y dispara la interrupción en `GPIO 17` para volcado limpio (`flush`).
6	`C_UPS`	**Banco Supercapacitor de Respaldo** $5.5\ V$ $4.0\ F$ (o módulo UPS de celdas `18650` de $5.1\ V$ $3\ A$)	1	Sostiene la tensión de $5.1\ V$ durante 15 a $30\ s$ tras un apagón para cerrar archivos SQLite WAL e InfluxDB sin corrupción.
7	`SW1`	**Switch Ethernet Industrial de 5 Puertos (10/100/1000 Mbps)** para riel DIN con cables Cat6 blindados	1	Interconexión de baja latencia entre la pasarela Raspberry Pi, los puntos de acceso inalámbricos y las consolas de operador.
8	`PCB1`	**Placa de Circuito Impreso FR-4 Doble Capa (`MECATROBOTICA-EDGE-HAT-V07`)** con cabezal hembra de 2×20 pines y bornera `KF301`	1	Montaje definitivo por soldadura libre de falsos contactos para operación continua $24/7$.
9	`R1..R5, D1..D2`	Resistencias de película metálica ($1\ k\Omega$, $10\ k\Omega$, $2\times\ 330\ \Omega$) + Diodos LED de $3\ mm$ Verde y Ámbar + Capacitor `100 nF X7R`	1 kit	Polarización del optoacoplador `PC817` , indicadores visuales de estado de ingesta (`GPIO 27` y `GPIO 22`) y desacoplo local.

---

2. Cálculo de Ingeniería: Presupuesto de Almacenamiento y Factor de Compresión

Para dimensionar científicamente la unidad SSD externa, calculamos el volumen bruto diario de telemetría $D_{día}$ generado por una planta con $N_{nodos} = 20$ nodos enviando tramas de $B_{muestra} = 150\ Bytes$ a una frecuencia de muestreo $f_s = 1.0\ Hz$:

$$D_{día} = N_{nodos} \cdot B_{muestra} \cdot f_s \cdot 86\,400\ s/día = 20 \times 150 \times 1.0 \times 86\,400 = 259.2\ MB/día$$

Gracias al motor **TSM (Time-Structured Merge Tree)** de InfluxDB 2.7 (que aplica codificación *Delta-of-Delta* en marcas de tiempo y compresión XOR *Gorilla* en punto flotante), el volumen físico en disco se reduce aproximadamente un factor de $6.8\times$, ocupando solo $\approx 38\ MB/día$ ($\approx 1.14\ GB$ para la ventana de retención cruda de $30\ días$).

Al ejecutar nuestra tarea automática de **Downsampling** en lenguaje `Flux` hacia ventanas de $\Delta t_{agregado} = 5\ minutos = 300\ s$, el factor de reducción temporal es:

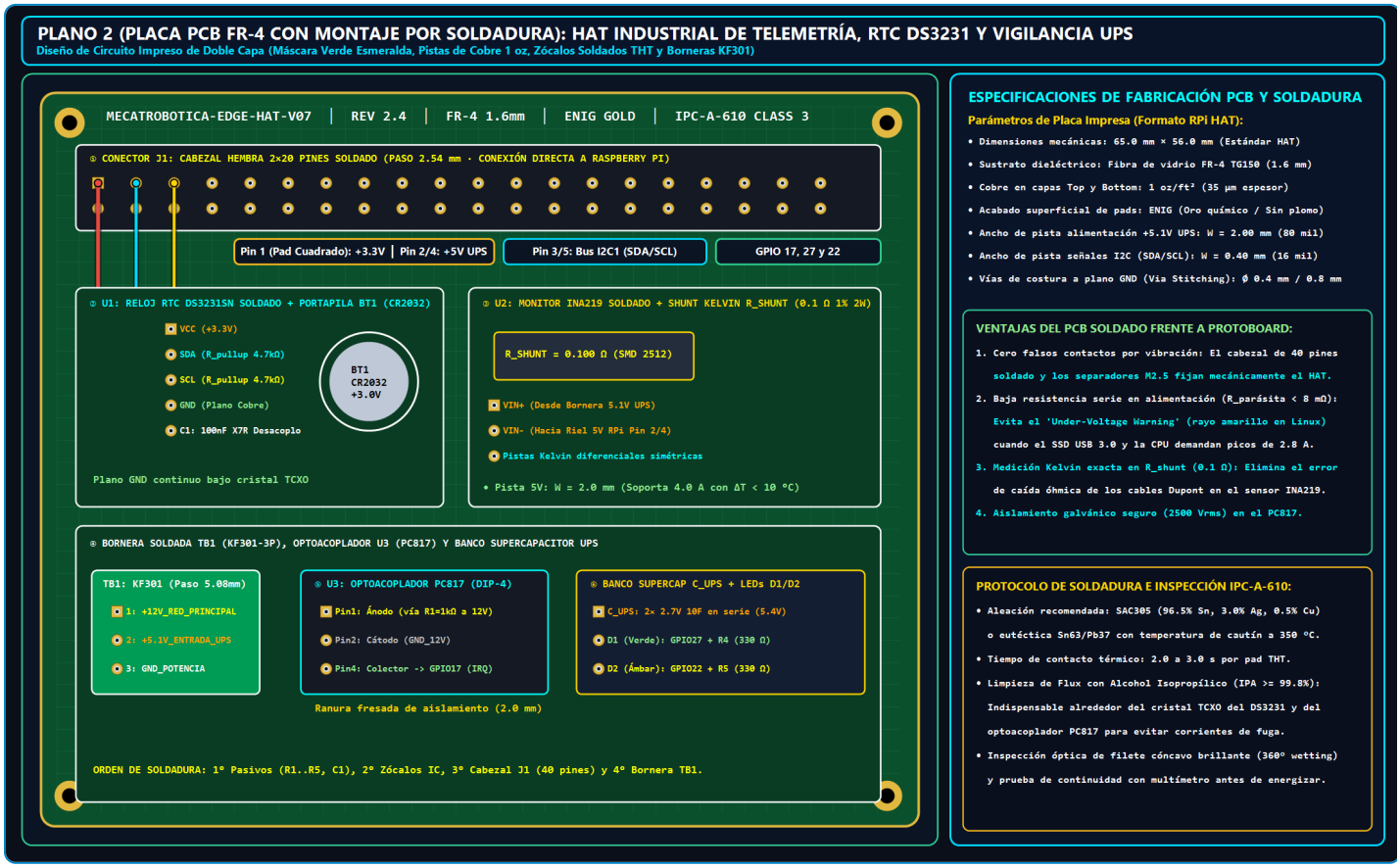
$$R_{comp} = \frac{\Delta t_{agregado}}{\Delta t_{crudo}} = (300\ s) / (1\ s) = 300 : 1$$

Por consiguiente, un año completo ($365\ días$) de historial analítico resumido (media, mínimo y máximo cada $5\ minutos$) requiere menos de $50\ MB$ de espacio en disco, garantizando consultas gráficas instantáneas en **Grafana**.

---

3. Comparativa de Montaje: Protoboard (Tinkercad) vs. Placa Impresa Soldada (PCB HAT)

📐 Plano 2: Placa PCB FR-4 con Montaje por Soldadura



MANUAL TÉCNICO DE INGENIERÍA — MÓDULO 07

Arquitectura de Contenedores Docker en ARM64, Series Temporales en InfluxDB 2.7, Resiliencia Store-and-Forward y Ergonomía Visual ISA-101

Curso: Internet de las Cosas (IoT) con Arduino y Raspberry Pi — Ingeniería Mecatrobótica

---

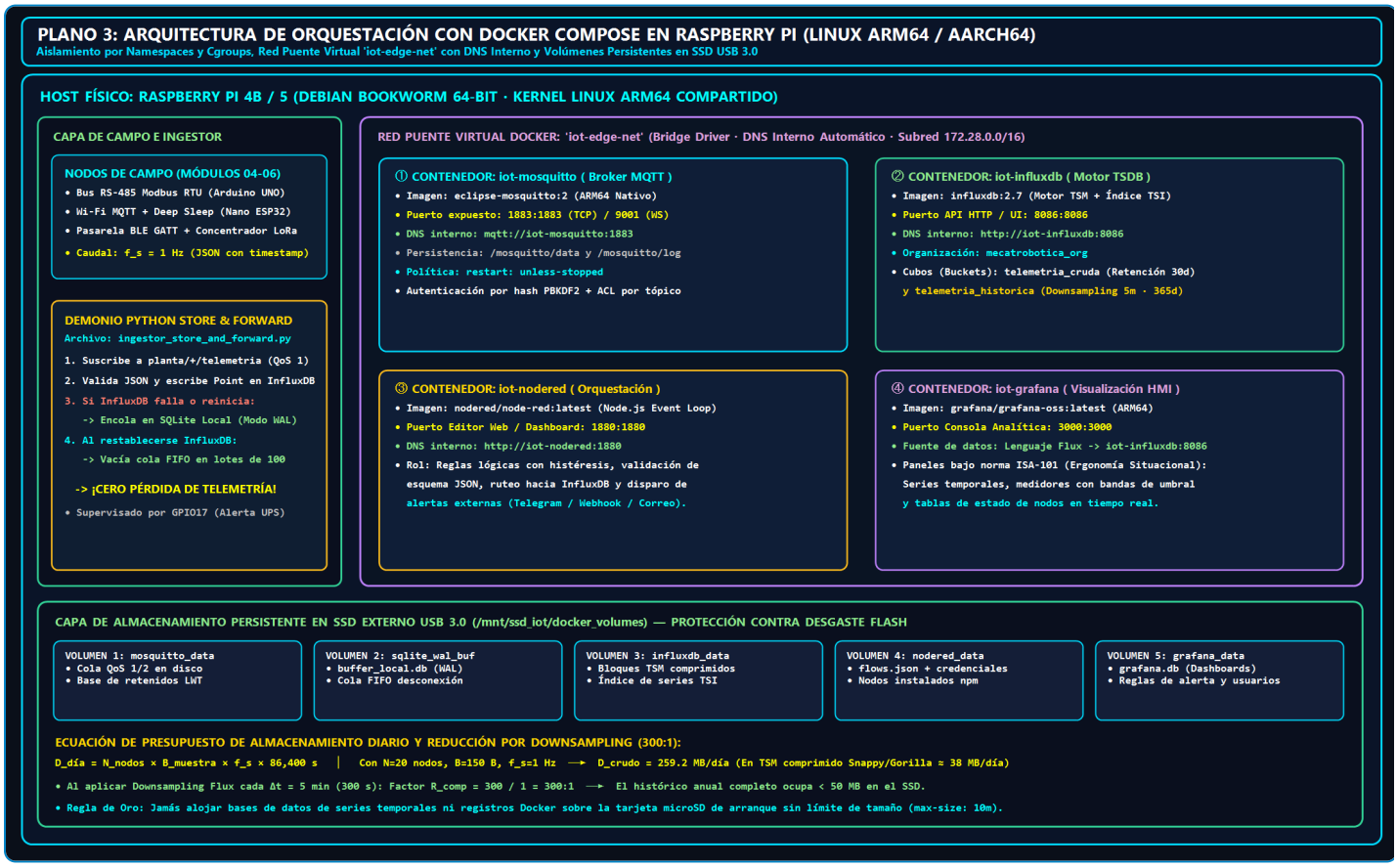
1. Fundamentos de Contenedorización con Docker en Linux ARM64

En despliegues industriales de borde (*Edge Computing*), instalar manualmente demonios de bases de datos, intérpretes de Node.js y servidores web directamente sobre el sistema operativo anfitrión conduce a conflictos de bibliotecas dinámicas y dificulta la recuperación ante desastres.

Mediante **Docker Engine** y **Docker Compose** sobre arquitecturas `ARM64 (aarch64)`, cada servicio opera aislado mediante dos primitivas nativas del kernel de Linux (sin la sobrecarga de emular hardware de una máquina virtual):

- **Espacios de nombres (`Namespaces`)**: Aíslan el árbol de procesos (`PID`), la pila de red (`NET`) y los puntos de montaje (`MNT`) de cada contenedor.
- **Grupos de control (`Cgroups v2`)**: Limitan y auditan el uso de memoria RAM y ciclos de CPU por servicio.
- **Red puente definida por el usuario (`iot-edge-net`)**: Provee resolución DNS interna automática por nombre de servicio (`iot-mosquitto`, `iot-influxdb`, `iot-nodered`, `iot-grafana`), evitando depender de direcciones IP estáticas frágiles.

📐 Plano 3: Arquitectura Docker Compose ARM64



---

2. Bases de Datos Relacionales (SQL) frente a Series Temporales (TSDB)

Una señal de telemetría mecatrónica es una sucesión discreta ordenada cronológicamente:

$$S = \{(t_1, \mathbf{v}_1), (t_2, \mathbf{v}_2), \dots, (t_k, \mathbf{v}_k)\} \quad \text{con } t_1 < t_2 < \dots < t_k$$

2.1. Anatomía del Protocolo de Línea (*Line Protocol*) en InfluxDB

Cada registro en InfluxDB se compone de cuatro elementos:

```
ambiente,nodo=invernadero_01,ubicacion=sector_norte temperatura=24.65,humedad=61.2,pwm_ventilador=140i
1730505600000000000
```

- **Medición (`Measurement`)**: Contenedor lógico de la magnitud (`ambiente`).
- **Conjunto de Etiquetas (`Tag Set`)**: Pares clave-valor de metadatos **indexados en memoria RAM** (`nodo`, `ubicacion`). Se utilizan en cláusulas de filtrado rápido.
- **Conjunto de Campos (`Field Set`)**: Valores numéricos variables **no indexados** (`temperatura`, `humedad`, `pwm_ventilador`).
- **Marca de Tiempo (`Timestamp`)**: Entero de 64 bits en nanosegundos UTC desde la época Unix.

2.2. Control Matemático de la Cardinalidad de Series

La cardinalidad total de series C_{series} en un cubo de InfluxDB viene dada por el producto cartesiano de los valores únicos de cada etiqueta:

$$C_{series} = |M| \times \prod_{i=1}^{m} |Tag_i| \times |Field_{keys}|$$

> **Regla Crítica de Diseño**: Jamás almacenes valores continuos (temperaturas con decimales), marcas de tiempo ni identificadores únicos por mensaje (`UUID`) dentro de un **Tag**. Hacerlo provoca una explosión de cardinalidad (*Runaway Cardinality*) que agota la memoria RAM de la Raspberry Pi (`OOM Kill`). Los valores variables van siempre en **Fields**.

---

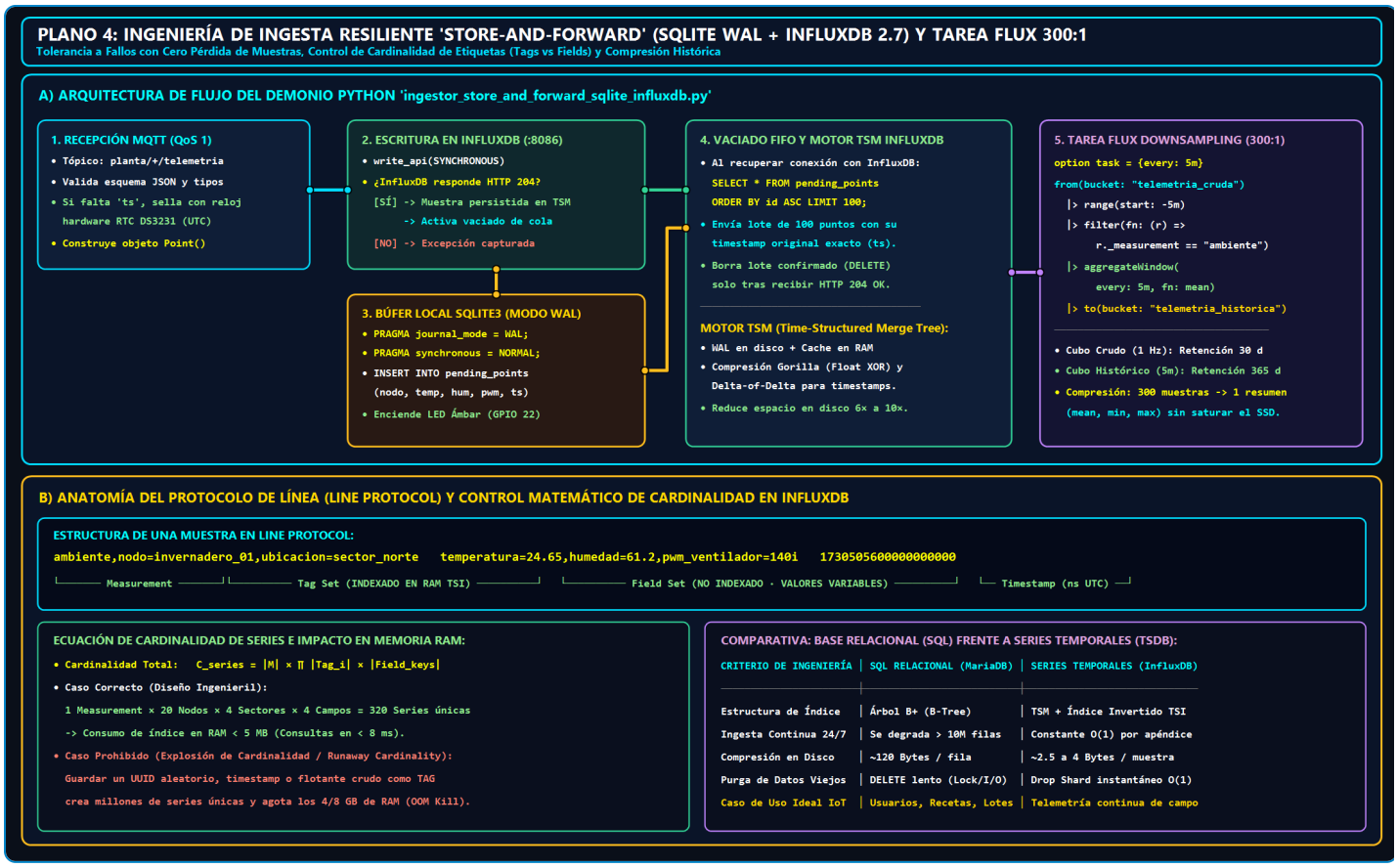
3. Patrón de Resiliencia *Store-and-Forward* con SQLite3 en Modo WAL

Para alcanzar una pérdida de datos igual a cero ($\text{Data Loss} = 0\%$) cuando el contenedor de InfluxDB se reinicia por mantenimiento o el enlace hacia un servidor central cae, nuestro demonio en Python implementa un búfer local en **SQLite3** activando la directiva:

```
PRAGMA journal_mode = WAL;
PRAGMA synchronous = NORMAL;
```

El modo **Write-Ahead Logging (`WAL`)** escribe las transacciones entrantes en un archivo secuencial de apéndice (`buffer_local_wal.db-wal`) sin bloquear las lecturas concurrentes y reduciendo drásticamente las operaciones de escritura aleatoria sobre las celdas Flash del SSD.

📐 **Plano 4: Flujo Store-and-Forward y Tarea Flux 300:1**



---

4. Ergonomía Visual bajo Norma ANSI/ISA-101 en Grafana

Un panel de control industrial no se evalúa por cuántos colores brillantes exhibe, sino por la velocidad con la que un operador humano detecta una condición anormal a las tres de la madrugada:

- **Contexto neutro de bajo contraste**: Fondos oscuros mate (`#0A101C` y `#121E36`) que reducen el deslumbramiento ocular.
- **Reserva cromática para anomalías**: Las curvas normales utilizan tonos cian o verde esmeralda moderados; el color **ámbar (`#FFBE1E`)** indica advertencia preventiva y el color **rojo (`#FF5050`)** se reserva exclusivamente para alarmas críticas activas.
- **Jerarquía de tres niveles**: Nivel 1 (KPIs globales y estado de salud), Nivel 2 (tendencias por subsistema con bandas sombreadas) y Nivel 3 (diagnóstico detallado de tramas, RSSI y latencias).

📐 Plano 5: Ergonomía Visual ISA-101 y Flujo Node-RED

PLANO 5: DISEÑO DE INTERFACES HMI BAJO NORMA ISA-101 EN GRAFANA Y FLUJO DE REGLAS EN NODE-RED

Conciencia Situacional para Operadores de Planta, Jerarquía de 3 Niveles, Histéresis de Alerta y Disponibilidad Operativa (SLA)

A) PANEL DE SUPERVISIÓN INDUSTRIAL GRAFANA (:3000) DISEÑADO BAJO ESTÁNDAR ANSI/ISA-101.01

TEMPERATURA PROCESO (NODO 01)	HUMEDAD RELATIVA Y PRESIÓN	PRE-ALARMA PREVENTIVA (ADVERTENCIA)	ALARMA CRÍTICA ISA-18.2 (PRIORIDAD 1)
24.6 °C	61.2 %RH | 1013 hPa	29.4 °C (Sector Sur)	0 ALARMAS ACTIVAS
Banda Nominal: [18.0 .. 28.0 °C] (OK)	Sensor Bosch BME280 (I2C 0x76)	Supera umbral preventivo > 28.0 °C	Disponibilidad SLA: A = 99.982% (Online)

CURVA DE SERIE TEMPORAL CON BANDA OPERATIVA SOMBREADA (CONSULTA FLUX aggregateWindow)



REGLAS DE ORO DE ERGONOMÍA VISUAL ISA-101:

1. Prohibido el efecto 'Árbol de Navidad': Si todo brilla en rojo, verde y amarillo saturado, el operador sufre fatiga visual y no detecta fallos.
2. Fondos oscuros mate (#0A101C / #121E36) y rejillas de bajo contraste para descanso ocular en turno 24/7.
3. Reserva cromática estricta:
   - Cian / Esmeralda -> Operación dentro de consigna
   - Ámbar (#FFBE1E) -> Desviación preventiva
   - Rojo (#FF5050) -> Alarma crítica que exige acción
4. Incluir siempre unidades físicas (°C, %RH, mA, dBm).

B) FLUJO DE ORQUESTACIÓN EN NODE-RED (:1880) CON HISTÉRESIS DE ALARMA Y CÁLCULO DE DISPONIBILIDAD (SLA)

① MQTT In
planta/+/telemetria
(QoS 1 · JSON)

② JSON Parser &
Schema Validator
(Verifica llaves)

③ Function Node
Filtro Histéresis
(T_on=30, T_off=28.5)

④ RBE / Filter
Report by Exception
(Bloquea spam)

⑤ Salida Dual:
• InfluxDB Batch Out
• Alerta Telegram/MQTT

LÓGICA JAVASCRIPT CON MEMORIA DE CONTEXTO E HISTÉRESIS EN NODE-RED:

```
const T_ALARMA_ON  = 30.0; // Umbral de disparo (°C)
const T_ALARMA_OFF = 28.5; // Umbral de reposición (Banda ΔT = 1.5 °C)
let enAlarma = context.get('enAlarma') || false;
if (!enAlarma && msg.payload.temp >= T_ALARMA_ON) {
    enAlarma = true;  context.set('enAlarma', true);
    return [msg, { payload: { alerta: 'ALTA_TEMP', valor: msg.payload.temp } }];
} else if (enAlarma && msg.payload.temp <= T_ALARMA_OFF) {
    enAlarma = false; context.set('enAlarma', false);
}
```

INGENIERÍA DE CONFIABILIDAD Y DISPONIBILIDAD DEL SISTEMA (MTBF / MTTR):

1. Ecuación de Disponibilidad Estacionaria (Availability):
   A = MTBF / (MTBF + MTTR) = T_operativo / T_total

2. Efecto de Docker 'restart: unless-stopped' + Healthchecks:
   Reduce el Tiempo Medio de Reparación (MTTR) de horas (intervención humana manual) a menos de 3.5 segundos (reinicio automático).

3. Combinado con el búfer SQLite WAL (Store-and-Forward):
   La disponibilidad efectiva de datos alcanza el 100.0% frente a microcortes de red o reinicios de mantenimiento de contenedores.

PLANOS Y ESQUEMÁTICOS DE CIRCUITO E INGENIERÍA — MÓDULO 07

Planos en Protoboard (Estilo Tinkercad), Placa PCB FR-4 con Montaje por Soldadura y Diagramas de Arquitectura

Curso: Internet de las Cosas (IoT) con Arduino y Raspberry Pi — Ingeniería Mecatrobótica

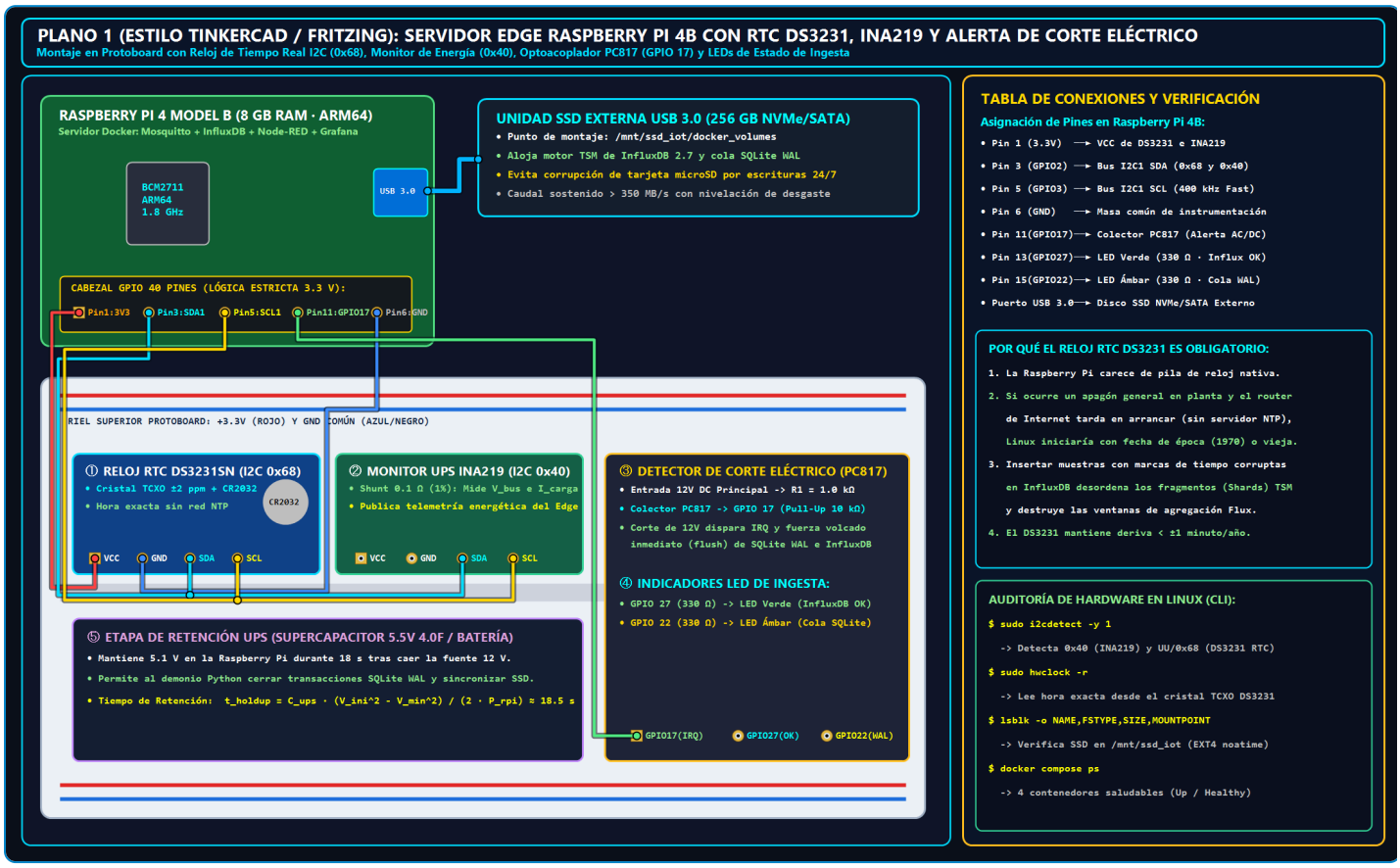
Módulo: 07 — Persistencia en Series Temporales, Docker, Node-RED, InfluxDB y Grafana

---

1. Plano 1 (Estilo Tinkercad / Fritzing): Montaje en Protoboard con RTC DS3231, INA219 y Alerta de Corte Eléctrico

En este primer plano de alta legibilidad se detalla el conexionado sobre protoboard del servidor Edge **Raspberry Pi 4B** acoplado a una unidad **SSD USB 3.0**, un reloj de tiempo real de alta precisión **DS3231SN (** `0x68` **)**, un monitor de potencia **INA219 (** `0x40` **)** y un detector optoacoplado de caída de alimentación principal **PC817** conectado al pin `GPIO 17`.

📐 Plano 1: Montaje en Protoboard Estilo Tinkercad



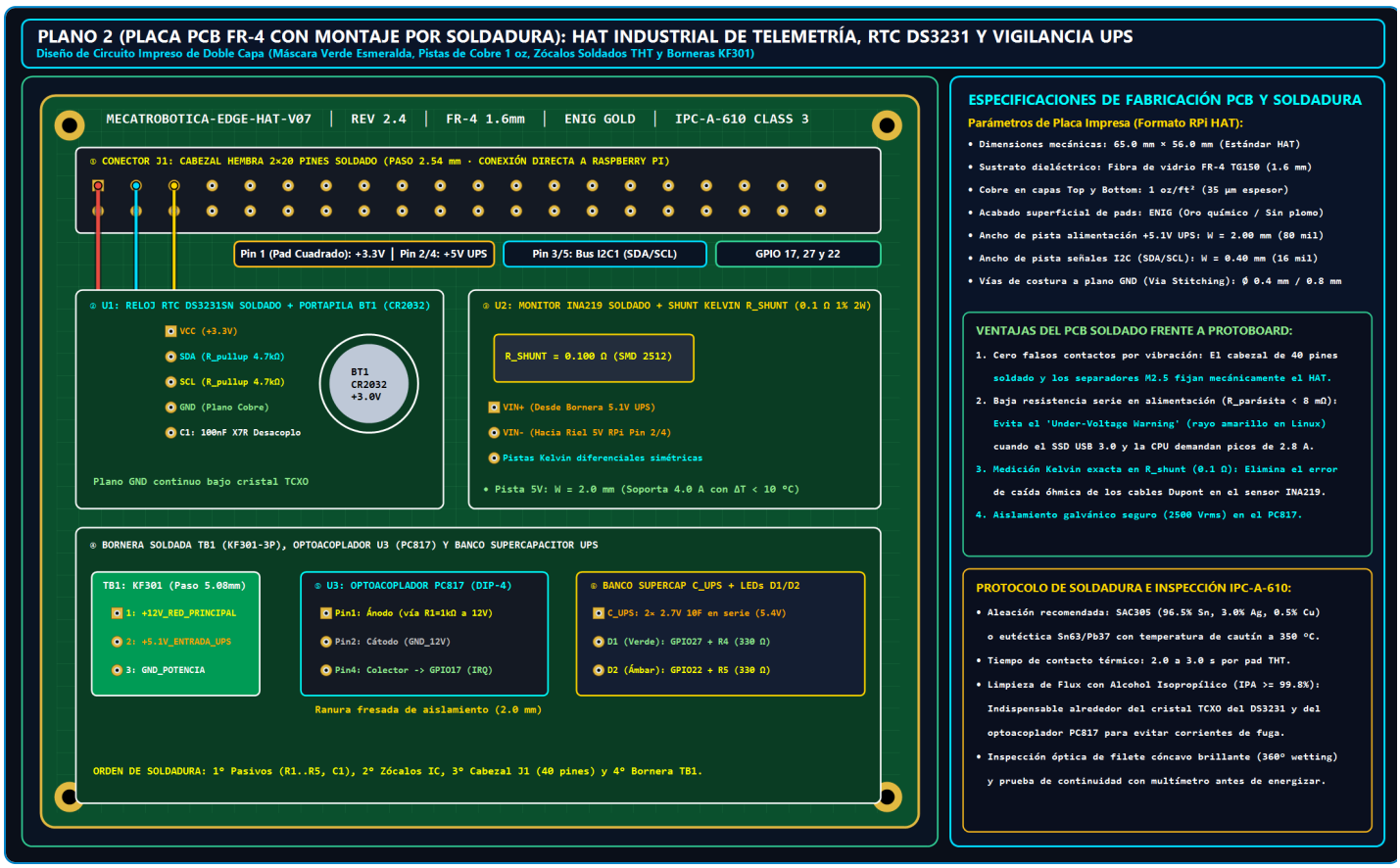
---

2. Plano 2 (Placa de Circuito Impreso PCB FR-4): Montaje por Soldadura de HAT Industrial para Raspberry Pi

Para servidores que operan las 24 horas de los 365 días del año en gabinetes sometidos a vibraciones de maquinaria, los cables puente Dupont sobre protoboard presentan riesgo de desconexión y caída óhmica en el riel de $5.1\ V$. Este segundo plano presenta el diseño de una **placa de circuito impreso de doble capa en fibra de vidrio FR-4 (** `MECATROBOTICA-EDGE-HAT-V07` **)** con montaje por soldadura (`IPC-A-610 Clase 3`):

- **Conector `J1` (2×20 pines hembra paso $2.54\ mm$ soldado)**: Acopla rígidamente el HAT sobre la Raspberry Pi junto con cuatro separadores mecánicos `M2.5`.
- **Pistas de potencia de $2.0\ mm$ de ancho y medición Kelvin en `R_SHUNT` ($0.100\ \Omega$)**: Soportan picos de corriente de hasta $4.0\ A$ con una caída resistiva despreciable.
- **Bornera de tornillo `TB1 (KF301)` soldada y ranura de aislamiento bajo el optoacoplador `PC817`**: Separan físicamente el riel industrial de $12\ V\ DC$ de la lógica de $3.3\ V$ del procesador Broadcom.

📐 Plano 2: Placa PCB FR-4 con Montaje por Soldadura

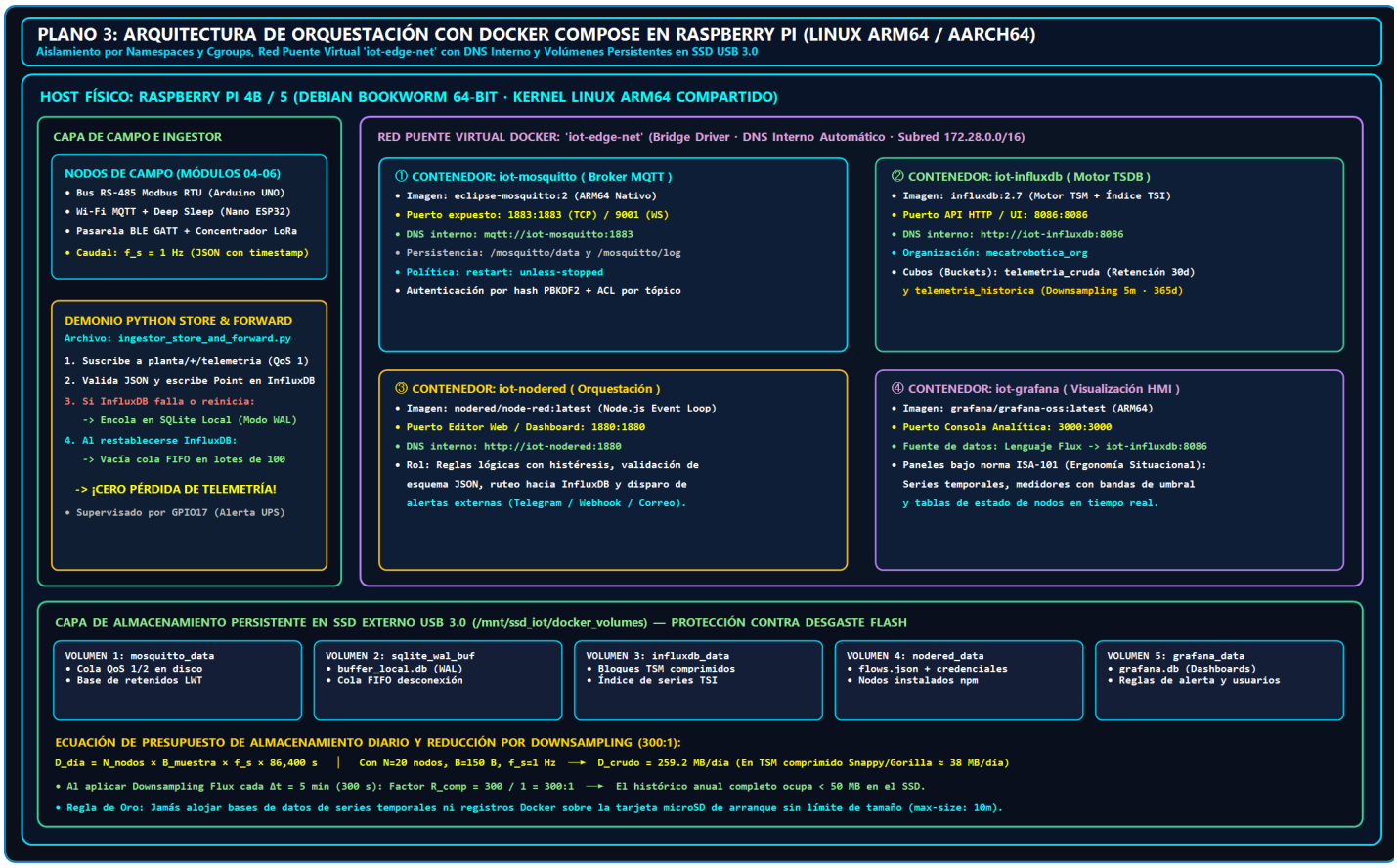


---

3. Plano 3: Arquitectura de Orquestación con Docker Compose en Linux ARM64

Representación detallada de la red puente aislada `iot-edge-net`, el mapeo de puertos de red (`1883`, `8086`, `1880` y `3000`) y la vinculación de los cinco volúmenes persistentes sobre la unidad externa **SSD USB 3.0** (`/mnt/ssd_iot/docker_volumes`).

📐 Plano 3: Arquitectura Docker Compose ARM64

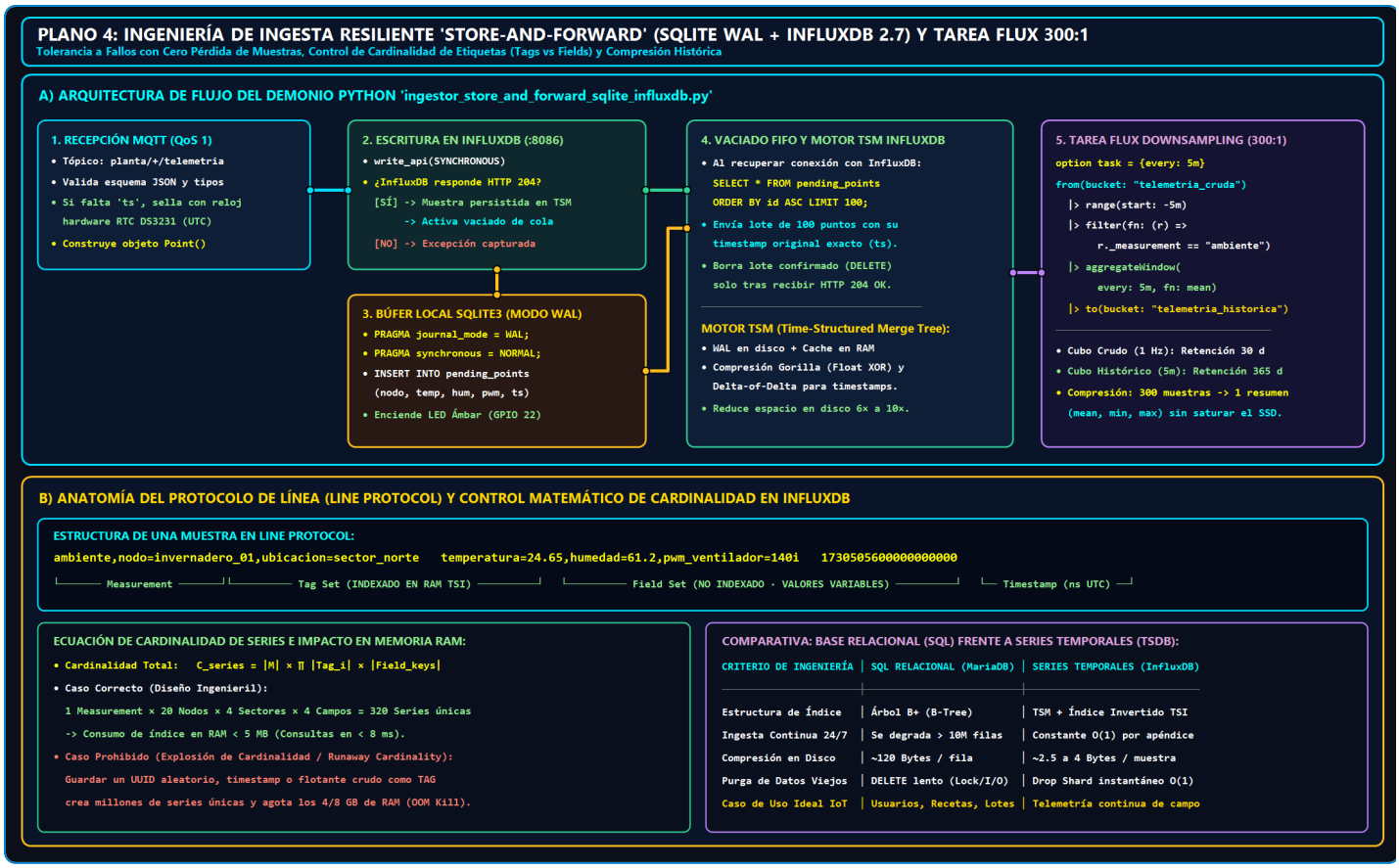


---

4. Plano 4: Flujo Store-and-Forward (SQLite WAL + InfluxDB 2.7) y Tarea Flux 300:1

Diagrama de ingeniería del demonio suscriptor en Python con conmutación automática a cola `SQLite3 WAL` ante caídas de red, recuperación por lotes FIFO (`LIMIT 100`), anatomía del *Line Protocol* y control de cardinalidad.

📐 Plano 4: Flujo Store-and-Forward y Tarea Flux 300:1



---

5. Plano 5: Ergonomía Visual ISA-101 en Grafana y Flujo de Reglas con Histéresis en Node-RED

Diseño de paneles de alto desempeño con banda operativa sombreada, semántica cromática estricta (cian/esmeralda nominal, ámbar preventivo y rojo crítico) y máquina de estados con histéresis en Node-RED.

📐 Plano 5: Ergonomía Visual ISA-101 y Flujo Node-RED

PLANO 5: DISEÑO DE INTERFACES HMI BAJO NORMA ISA-101 EN GRAFANA Y FLUJO DE REGLAS EN NODE-RED

Conciencia Situacional para Operadores de Planta, Jerarquía de 3 Niveles, Histéresis de Alerta y Disponibilidad Operativa (SLA)

A) PANEL DE SUPERVISIÓN INDUSTRIAL GRAFANA (:3000) DISEÑADO BAJO ESTÁNDAR ANSI/ISA-101.01

TEMPERATURA PROCESO (NODO 01)

24.6 °C

Banda Nominal: [18.0 .. 28.0 °C] (OK)

HUMEDAD RELATIVA Y PRESIÓN

61.2 %RH | 1013 hPa

Sensor Bosch BME280 (I2C 0x76)

PRE-ALARMA PREVENTIVA (ADVERTENCIA)

29.4 °C (Sector Sur)

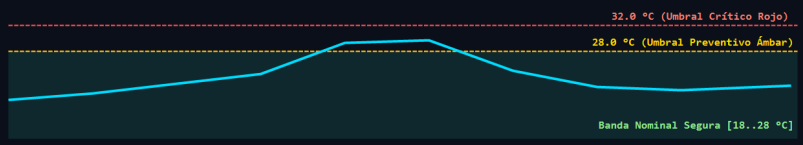
Supera umbral preventivo > 28.0 °C

ALARMA CRÍTICA ISA-18.2 (PRIORIDAD 1)

0 ALARMAS ACTIVAS

Disponibilidad SLA: A = 99.982% (Online)

CURVA DE SERIE TEMPORAL CON BANDA OPERATIVA SOMBREADA (CONSULTA FLUX aggregateWindow)



REGLAS DE ORO DE ERGONOMÍA VISUAL ISA-101:

1. Prohibido el efecto 'Árbol de Navidad': Si todo brilla en rojo, verde y amarillo saturado, el operador sufre fatiga visual y no detecta fallos.
2. Fondos oscuros mate (#0A101C / #121E36) y rejillas de bajo contraste para descanso ocular en turno 24/7.
3. Reserva cromática estricta:
   - Cian / Esmeralda -> Operación dentro de consigna
   - Ámbar (#FFBE1E) -> Desviación preventiva
   - Rojo (#FF5050) -> Alarma crítica que exige acción
4. Incluir siempre unidades físicas (°C, %RH, mA, dBm).

B) FLUJO DE ORQUESTACIÓN EN NODE-RED (:1880) CON HISTÉRESIS DE ALARMA Y CÁLCULO DE DISPONIBILIDAD (SLA)

① MQTT In
planta/+/telemetria
(QoS 1 · JSON)

② JSON Parser &
Schema Validator
(Verifica llaves)

③ Function Node
Filtro Histéresis
(T_on=30, T_off=28.5)

④ RBE / Filter
Report by Exception
(Bloquea spam)

⑤ Salida Dual:
• InfluxDB Batch Out
• Alerta Telegram/MQTT

LÓGICA JAVASCRIPT CON MEMORIA DE CONTEXTO E HISTÉRESIS EN NODE-RED:

```
const T_ALARMA_ON  = 30.0; // Umbral de disparo (°C)
const T_ALARMA_OFF = 28.5; // Umbral de reposición (Banda ΔT = 1.5 °C)
let enAlarma = context.get('enAlarma') || false;
if (!enAlarma && msg.payload.temp >= T_ALARMA_ON) {
    enAlarma = true;  context.set('enAlarma', true);
    return [msg, { payload: { alerta: 'ALTA_TEMP', valor: msg.payload.temp } }];
} else if (enAlarma && msg.payload.temp <= T_ALARMA_OFF) {
    enAlarma = false; context.set('enAlarma', false);
}
```

INGENIERÍA DE CONFIABILIDAD Y DISPONIBILIDAD DEL SISTEMA (MTBF / MTTR):

1. Ecuación de Disponibilidad Estacionaria (Availability):
   A = MTBF / (MTBF + MTTR) = T_operativo / T_total

2. Efecto de Docker 'restart: unless-stopped' + Healthchecks:
   Reduce el Tiempo Medio de Reparación (MTTR) de horas (intervención humana manual) a menos de 3.5 segundos (reinicio automático).

3. Combinado con el búfer SQLite WAL (Store-and-Forward):
   La disponibilidad efectiva de datos alcanza el 100.0% frente a microcortes de red o reinicios de mantenimiento de contenedores.