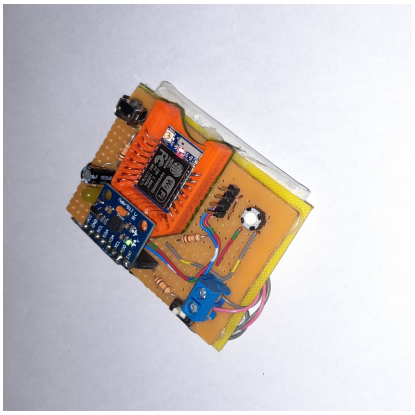
UNIVERSIDAD NACIONAL DE CUYO
Facultad de Ingeniería

Red de nodos de adquisición de datos sismográficos.

Informe de avance - Versión 0.1



CARRERA Ingeniería Mecatrónica

SEMESTRE Segundo Semestre 2025

CÁTEDRA Proyecto Tipo C

ALUMNOS Barrios Retta Federico , 14101
Cano Francisco , 12487
Pappeti Juan Ignacio , 11807

FECHA 03 de noviembre de 2025

Contents

1 Resumen ejecutivo

Alcance acordado: red de nodos sismográficos IoT con adquisición triaxial, backend MQTT+MySQL, dashboard.

1.1 Avances clave

- Diseño de placa de desarrollo v0.0.1 (THT) y v0.0.2 (SMD).
- Prototipo físico para dimensionamiento de dimensiones.
- Base de datos MySQL operativa.
- Análisis de energía y batería, con dificultades detectadas.
- Prototipo de firmware con timers/ISR para muestreo periódico (FreeRTOS).

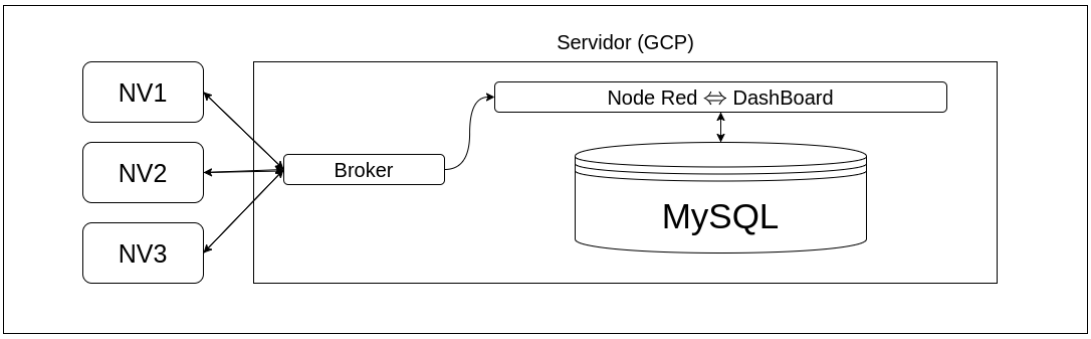
1.2 Riesgos activos

- Autonomía *real vs objetivo*
- Deep sleep con reset
- Tensión de la pila de Li-ion difiere con V_{cc} de alimentación del microcontrolador
- Reconexión robusta tras sleep.

1.3 Próximos hitos

- Validación energética en banco
- Prototipo físico de la placa de desarrollo v0.0.2 (SMD)
- Inserción bulk y particionado en DB.

1.4 Diagrama de bloques (visión actual)



2 Trazabilidad de requerimientos actuales

Tabla simple: **Requerimientos clave** → **Artefactos** → **Pruebas** → **Resultado**

Req	Artefacto	Prueba	Estado
Muestreo estable (100–200 Hz)	FW timers/ISR	Jitter $< 1\%$ en 30 min	En curso
Persistencia	Esquema MySQL	Inserciones 1k msg/min 10 min	OK/Parcial
Autonomía mínima N horas	Modelo energía	Medición I y duty cycle	En riesgo

Req	Artefacto	Prueba	Estado
Reconexión tras deep sleep	FW estados	Publicación QoS1 en ≤3 s	Pendiente

3 Problematica actual: la bateria de Li-ion

La red de nodos requiere **autonomía eléctrica** para operar sin red fija. La fuente elegida es una celda Li-ion de factor de forma compacto ($\leq 50 \times 50$ mm, espesor < 7.5 mm). El ESP8266 opera entre **2.5~3.6 V** y presenta cuatro modos de potencia (activo, modem-sleep, light-sleep y deep-sleep). Según el datasheet del ESP8266EX , el consumo típico es: deep $-$ sleep $\approx 20\mu$A, light $-$ sleep ≈ 0.9mA, modem $-$ sleep ≈ 15mA, y en activo con radio el promedio ronda **decenas de mA** con picos de **>100 mA** en TX/RX (ver Tabla 3-4 "Power Consumption by Power Modes" y Tabla 1-1 "Specifications").

El consumo del módulo MPU6050 se debe suponer hasta realizarse mediciones. **Supuesto**: la IMU **MPU6050** añade **≈ 3–4 mA** en adquisición continua y **< 10 µA** en modo low-power accel. Estos valores se validarán en banco.

Implicancia: la autonomía no la determina el consumo de 20μA en deep-sleep, sino el **duty cycle real** entre medir, transmitir y dormir. Además, el **rango de tensión de la celda Li-ion** ($\approx 4.2 \rightarrow 3.0V$) no es directamente compatible con la ventana 2.5–$3.6V$ del ESP8266 si no se condiciona la tensión.

3.1 Restricciones de diseño

1. **Tensión**
   - Batería Li-ion: 4.2 V (cargada) a ~3.0 V (descargada).
   - ESP8266: 2.5–3.6 V.
   - **Conclusión:** se requiere regulación.
     - Un **diodo en serie** (≈ 0.7 V) reduce 4.2 $\rightarrow$ 3.5 V, pero **colapsa** a ~2.3 V cuando la celda cae a 3.0 V, **bajo el mínimo** del ESP8266. No es solución de producción.
     - Un **LDO** a 3.3 V desaprovecha la cola de la batería (cuando Vin < 3.3 V entra en dropout).
     - Un **buck** a 3.3 V es eficiente por arriba de 3.3 V, pero **no regula** cuando Vin < 3.3 V.
2. **Ciclo de operación** Definimos un ciclo con tres estados:
   - **Activo (adquisición + TX)**: ESP8266 activo con radio, IMU activa.
   - **Idle/light-sleep**: CPU o periféricos pausados, radio apagado.
   - **Deep-sleep**: solo RTC operativo; en ESP8266 el **wake implica reset** del MCU.
3. **Recuperación post-sleep** El **deep-sleep reinicia** el ESP8266: se debe **reconstruir estado** (NTP, MQTT, `seq`, reconfiguración IMU) de forma **idempotente**.

3.2 Marco de cálculo de autonomía

Para un ciclo de duración $T = t_a + t_l + t_d$ con corrientes I_a, I_l, I_d:

Considerando:

- T: duración total de un ciclo.

- t_a: tiempo en **estado activo** (adquisición + TX/RX).
- t_l: tiempo en **light-sleep** (CPU/periféricos en pausa, radio apagada).
- t_d : tiempo en **deep-sleep** (solo RTC activo).
- I_a: **corriente media** en estado activo.
- I_l: **corriente media** en light-sleep.
- I_d: **corriente media** en deep-sleep.

$$I_{\text{prom}} = \frac{I_a t_a + I_l t_l + I_d t_d}{T} \quad \Rightarrow \quad \text{Autonomía (h)} \approx \frac{C_{\text{mAh}} \cdot \eta}{I_{\text{prom}}}$$

- **Capacidad efectiva** C_{mAh}: nominal $\times$ **factor de eficiencia** $\eta \in [0.8, 0.9]$ para pérdidas de regulador y temperatura.
- **Supuestos iniciales** (a validar): $ I_a=120 mA $ promedio durante TX y lectura, $I_l = 0.9$ mA , $I_d = 20$ μA. IMU activa suma ~4 mA en I_a y ~10 μA en bajo consumo.

Ejemplo de orden de magnitud (no compromiso de diseño): Si $t_a = 2$ s, $t_l = 8$ s, $t_d = 50$ s por ciclo de 60 s, entonces $I_{\text{prom}} \approx \frac{(124)(2)+(0.9)(8)+(0.02)(50)}{60} \approx 4.5$ mA. Con una celda **2000 mAh** y $\eta = 0.85 \rightarrow \approx 377h$ teóricas. Pequeños cambios en t_a o I_a derrumban la cifra, de ahí la necesidad de medir.

3.3 Riesgos y mitigaciones

Riesgo	Efecto	Mitigación
Picos de TX > 200 mA	Reset brown-out	Capacitores cerca del SoC y del regulador, ruta de GND corta
Dropout en LDO	Apagado prematuro	Buck-boost o cutoff a Vbat que garantice 3.3 V estable
Overhead de reconexión Wi-Fi	Aumenta t_a	Persistir credenciales, aumentar DTIM, batch de publicaciones
Deriva de consumo IMU	Baja autonomía	Cambiar a modo low-power accel entre eventos

4 Diseños de placas de Desarrollo

4.1 Prototipo físico

4.1.1 Descripción general

Módulo de adquisición autónomo montado en placa perforada (protoboard FR-4) con **batería Li-ion plana** como sustrato mecánico. El conjunto integra **ESP8266 (ESP-01)**, **IMU MPU6050 (GY-521)**, **cargador/protección de batería Li-ion**, y **módulo DC-DC** para acondicionamiento de tensión. Se incorporan **interruptor**, **pulsador de reset/usuario**, **LED de estado** y **test-pads** para depuración.

4.1.2 Componentes principales

Conjunto	Función	Observaciones de montaje
ESP8266 ESP-01 con soporte impreso	MCU + Wi-Fi	Soporte 3D para alivio de tensión del conector; disipador pasivo pequeño.

Conjunto	Función	Observaciones de montaje
MPU6050 (GY-521)	Acelerómetro/giroscopio	Montado en el plano superior; acceso a SDA/SCL en header.
Cargador/protector Li-ion	Carga 5 V y protección de celda	Entrada micro-USB; ubicado en cara inferior junto a la batería.
DC-DC (ajustable)	Regulación a 3.3 V	Módulo comercial; ajuste en 3.30 ± 0.03 V.
Batería Li-ion tipo "flat"	Fuente	Fijación con cinta Kapton; cableado con alivio térmico.
Diodo serie (temporal)	Caída de ~0.7 V	Solución transitoria para compatibilidad de tensión.
Interfaz	Interruptor, pulsador, LED	Acceso lateral; útiles para pruebas de banco.

4.1.3 Interconexión eléctrica (alto nivel)

- **Batería → cargador/protección → DC-DC 3.3 V → rail lógica (ESP8266 + IMU)**.
- **GPIO/UA**: SDA/SCL expuestos; UART accesible para programación.

4.1.4 Funcionalidad validada en banco

- Encendido estable a 3.3 V regulados.
- Lectura de IMU.
- Carga de batería por micro-USB y operación en modo "cableado".

4.1.5 Limitaciones actuales

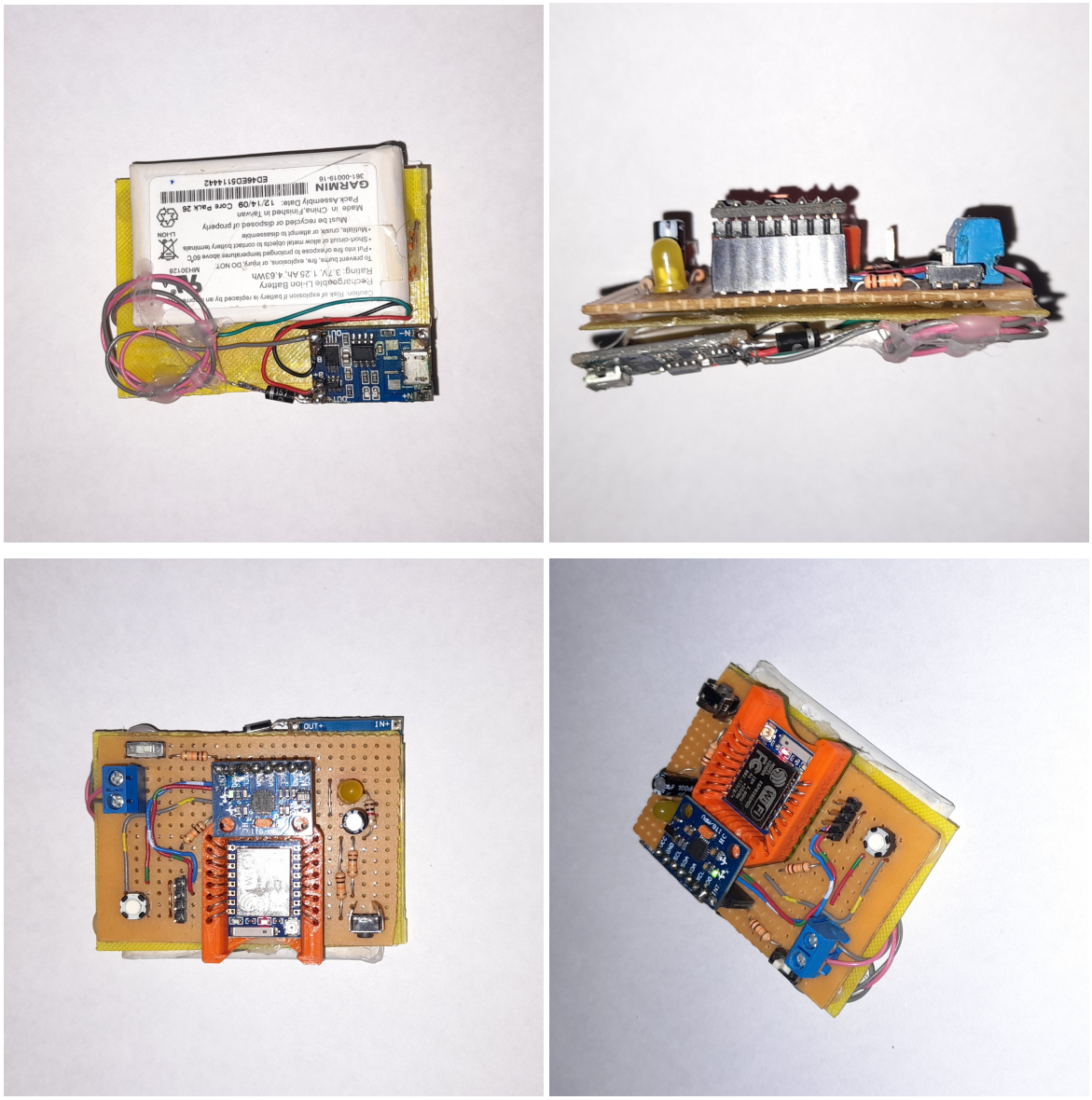
- **Regulación**: uso de diodo de caída como solución temporal; riesgo de subalimentación cuando Vbat $<$ 3.3 V + Vdrop.
- **Mecánica**: apilado con adhesivo caliente; se requiere placa SMD y separadores para rigidez y repetibilidad.
- **Medición de consumo**: sin shunt/INA dedicado; no hay perfil de corrientes por estado.
- **Protección**: sin TVS ni fusible rearmable en entrada; headers sin retención.

4.1.6 Próximos pasos de mejora

1. Migrar a **PCB v0.0.2 SMD** con plano GND continuo bajo IMU y "via-stitching".
2. Incorporar **montajes mecánicos**: tornillería M2.5, separadores, y casquillos para desacoplar vibraciones.
3. Añadir **protecciones**: TVS en entrada USB, PTC 500 mA, y filtro LC en rail 3.3 V.

4.1.7 Registro fotográfico (referencias)

- **Vista superior**: ubicación de ESP-01, GY-521, LED y controles.
- **Vista inferior**: batería y cargador/protección con cableado.
- **Vista lateral**: apilado de módulos, disipador en ESP-01, accesos a headers.
- **Vista isométrica**: vista general de la disposición del prototipo.



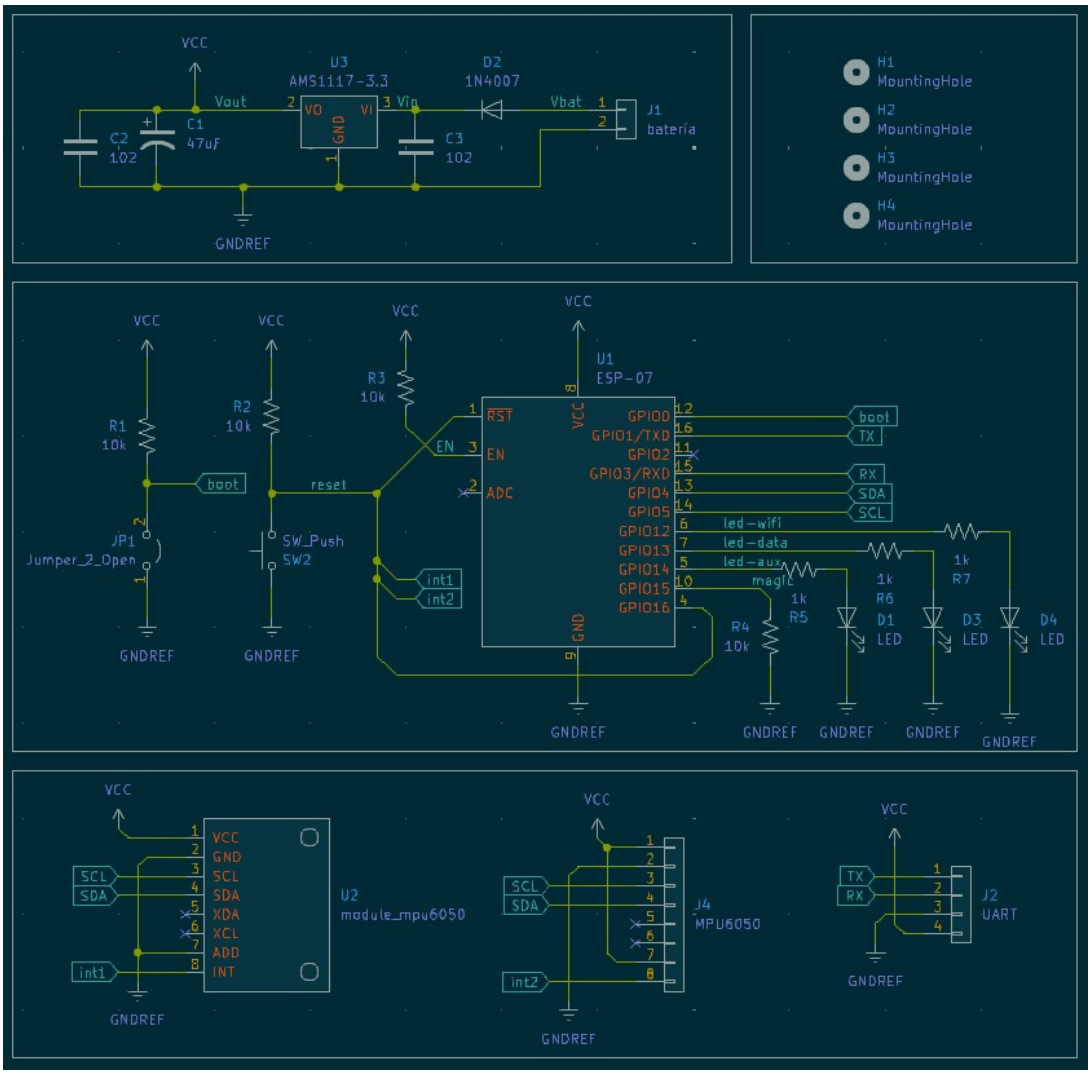
4.2 Placa de Desarrollo v0.0.1 (THT)

4.2.1 Descripción General

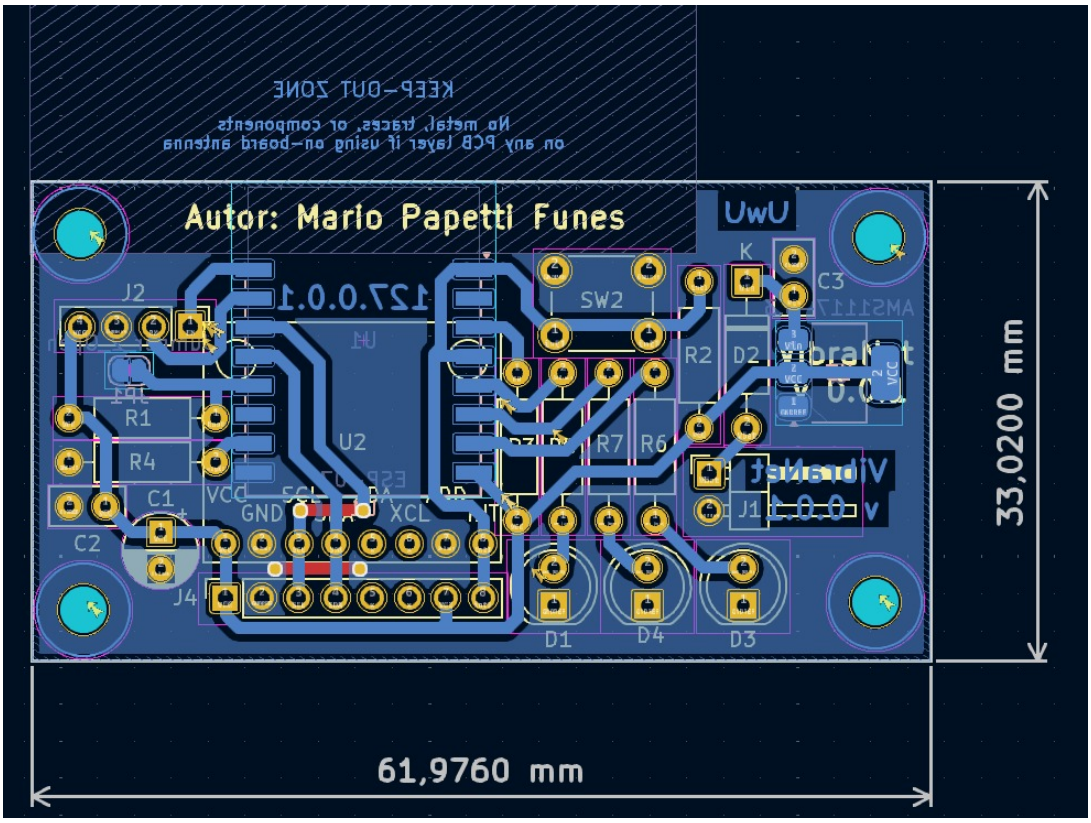
Se diseñó una primera revisión orientada a verificación funcional. La alimentación se resolvió con batería Li-ion, diodo serie y LDO AMS1117-3.3, suficiente para banco pero limitada en eficiencia y margen de caída. La lógica de arranque incluye pulsadores de BOOT y RESET con EN por pull-up. La IMU MPU6050 se conecta por I2C y expone INT al MCU. Se dispone de señalización mediante cuatro LEDs y de headers para UART y MPU6050.

- **Formato:** 61.98 × 33.02 mm, 4 orificios de montaje.
- **Topología:** batería Li-ion → diodo serie → **LDO AMS1117-3.3** → rail 3V3 para **ESP-07** y **GY-521**.
- **Señalización:** cuatro LEDs (Wi-Fi, datos, aux1, aux2) con resistencias limitadoras.
- **Control:** pulsadores de **BOOT** y **RESET**; **EN** con pull-up (R3).
- **Interfaz:** header **UART** para programación y header **I2C**/INT para IMU.
- **Conectividad IMU:** `SCL/SDA`, `INT` y `AD0` cableados al conector J4 (módulo MPU6050).

4.2.2 3) Bloques y esquemas



1. **Alimentación**: AMS1117-3.3 con **C1=47 µF** de bulk y cerámicos de desacople (C2, C3). Diodo **1N4007** en serie desde batería como solución transitoria para reducir Vbat.
2. **Control de arranque**:
   - `BOOT` con R1=10 kΩ a GND y pulsador a VCC para forzar modo flash cuando se requiera.
   - `RESET` con R2=10 kΩ, pulsador a GND y **EN** con R3=10 kΩ a VCC.
3. **MCU + IO**: **ESP-07** con GPIOs mapeados a UART, I2C y LEDs (R4–R7).



Ruteo de una cara con puentes; headers dedicados para **UART** y **MPU6050**. Se señalan keep-out y leyendas (autor, versión).

4.2.3 Pinout efectivo (para trazabilidad de firmware)

Función	Pin ESP8266	Conector	Nota
UART TX	GPIO1/TXD0	J2-TX	Programación/registro
UART RX	GPIO3/RXD0	J2-RX	Programación/registro
I2C SCL	GPIO5	J4-SCL	Pull-ups en módulo GY-521
I2C SDA	GPIO4	J4-SDA	—
IMU INT	GPIO14 (int1)	J4-INT	Interrupciones de movimiento
LED Wi-Fi	GPIO12	D1 + R4	Estado radio
LED Datos	GPIO13	D2 + R5	Publicación/actividad
LED Aux 1	GPIO15	D3 + R6	Uso general
LED Aux 2	GPIO16	D4 + R7	Uso general
EN (chip-enable)	EN	—	Pull-up R3=10 kΩ
RESET	RST	SW2	Pulsador a GND
BOOT	GPIO0	SW1/JP1	Forzar modo programación

4.2.4 Riesgos y mitigaciones (esta revisión)

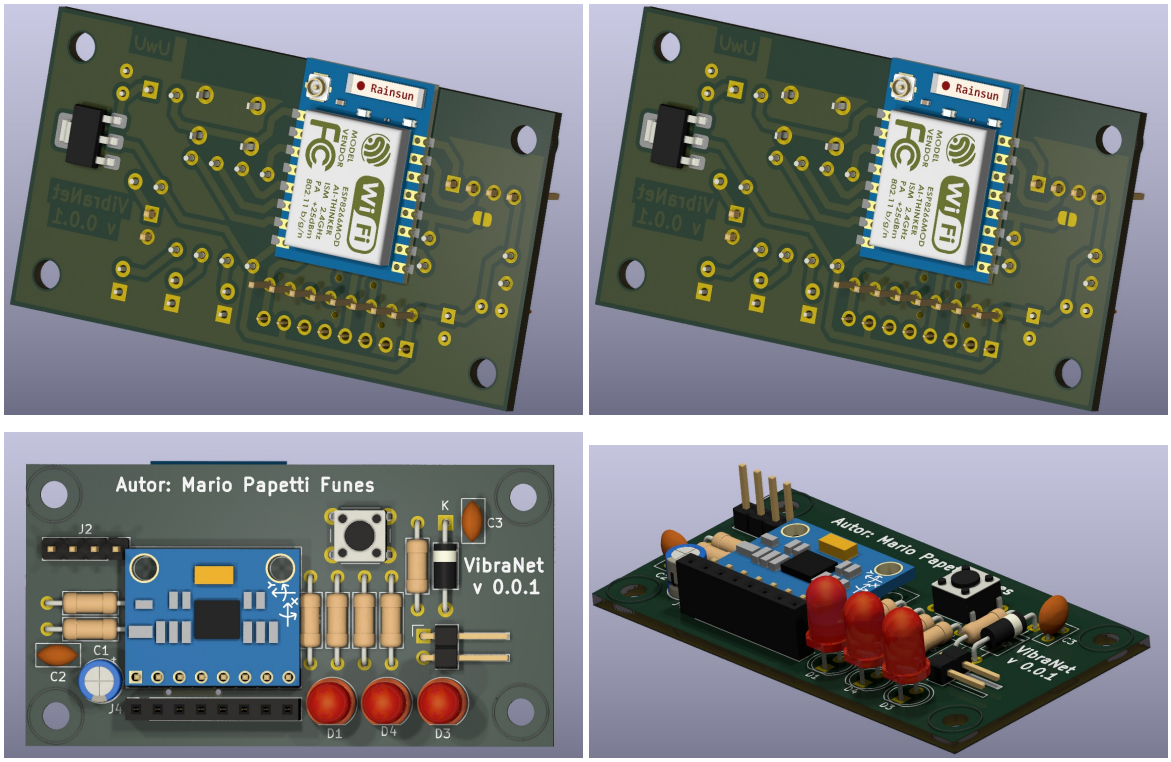
Riesgo	Impacto	Mitigación en v0.0.2
Dropout con LDO + diodo	Resets en TX	Buck-boost, cutoff Vbat
Falta de desacople correcto	Ruido en IMU/MCU	100 nF locales + 10 µF; trazas cortas

Riesgo	Impacto	Mitigación en v0.0.2
Sin medición de corriente	Autonomía incierta	Shunt/INA + pads Kelvin
Protección insuficiente	Sensibilidad a ESD/fallos	TVS, PTC, clearances y serigrafía

Con esto capitalizás las imágenes: el lector ve el esquema, entiende las rutas críticas, tiene tabla de pines para firmware y una lista clara de mejoras hacia v0.0.2.

4.2.5 Registro fotográfico (referencias)

- **Vista superior**: ubicación de ESP-01, GY-521, LED y controles.
- **Vista inferior**: batería y cargador/protección con cableado.
- **Vista lateral**: apilado de módulos, disipador en ESP-01, accesos a headers.
- **Vista isométrica**: vista general de la disposición del prototipo.



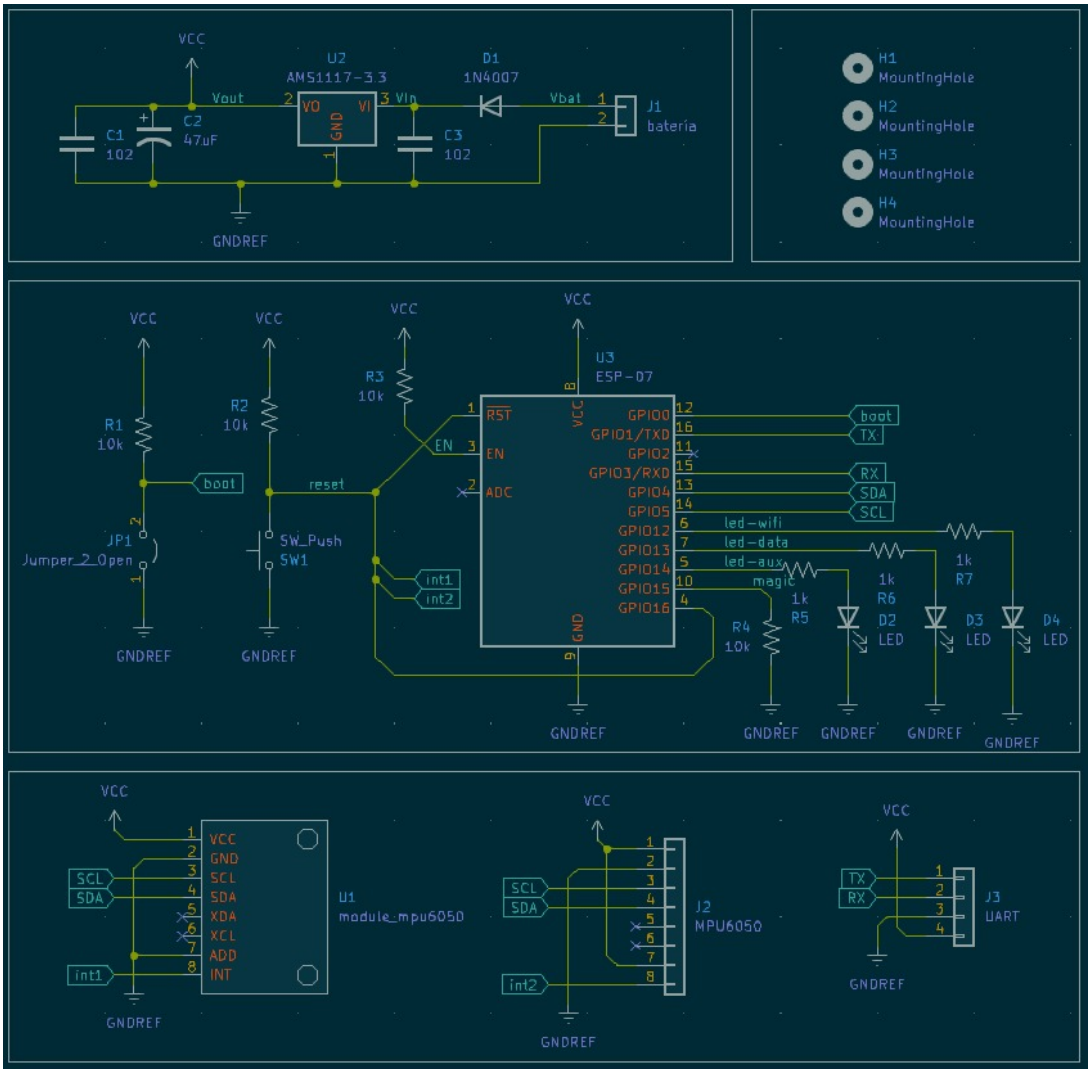
4.3 Placa de Desarrollo v0.0.2 (SMD)

4.3.1 Placa de Desarrollo v0.0.2 (SMD)

Descripción general Revisión orientada a **compactar** el diseño y mejorar **DFM/EMI** manteniendo la funcionalidad de la v0.0.1. Se migra a **SMD** con plano de masa continuo, keep-out para antena y headers accesibles para IMU y UART. La alimentación continúa con **LDO AMS1117-3.3 + diodo serie** como solución de banco, en espera de migrar a **buck-boost 3.3 V** en la siguiente iteración.

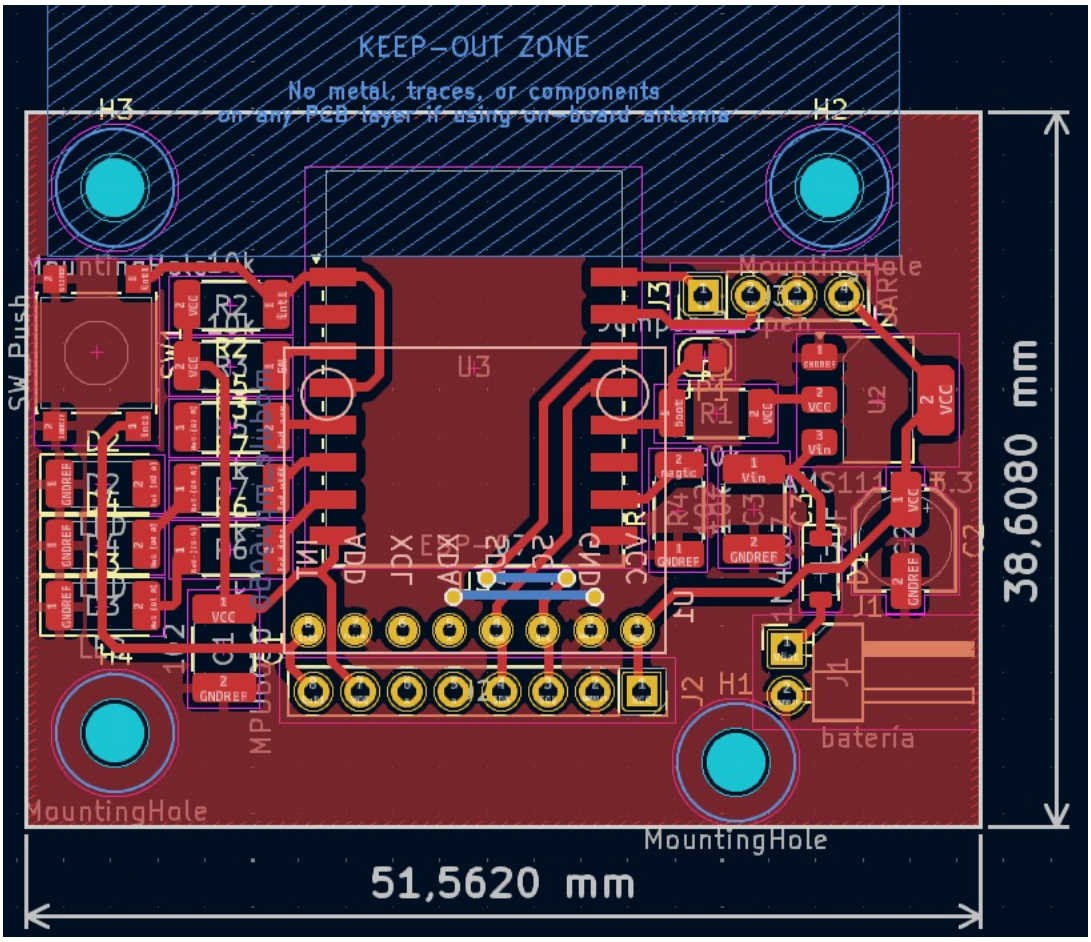
- **Formato: 51.56 × 38.61 mm**, 4 orificios de montaje.
- **Topología:** batería Li-ion → diodo serie → **LDO AMS1117-3.3** → rail 3V3 para **ESP-12** y conector **MPU6050**.

- **Keep-out RF:** zona superior libre de cobre y componentes sobre la antena integrada (serigrafía “KEEP-OUT ZONE”).
- **Control:** pulsador **RESET**, pad para **BOOT/FLASH** y `EN` con pull-up.
- **Interfaz:** header **UART** de programación y header **I2C/INT** para la IMU.
- **Señalización:** tres LEDs SMD para estado (Wi-Fi/datos/aux).



Bloques y esquemas

1. **Alimentación**: **AMS1117-3.3** con **C3 bulk** y cerámicos de desacople cercanos; **diodo 1N4007** serie desde batería como atajo temporal para ajustar Vbat.
2. **RF/MCU**: **ESP-12** (**ESP8266MOD**) con antena integrada Rainsun y keep-out superior; GPIO mapeados a UART, I2C y LEDs.
3. **Control de arranque**: `EN` con pull-up, `RESET` por pulsador; pads para `GPIO0` (modo flash).
4. **IMU**: conector para **MPU6050** con `SCL/SDA` y `INT`.

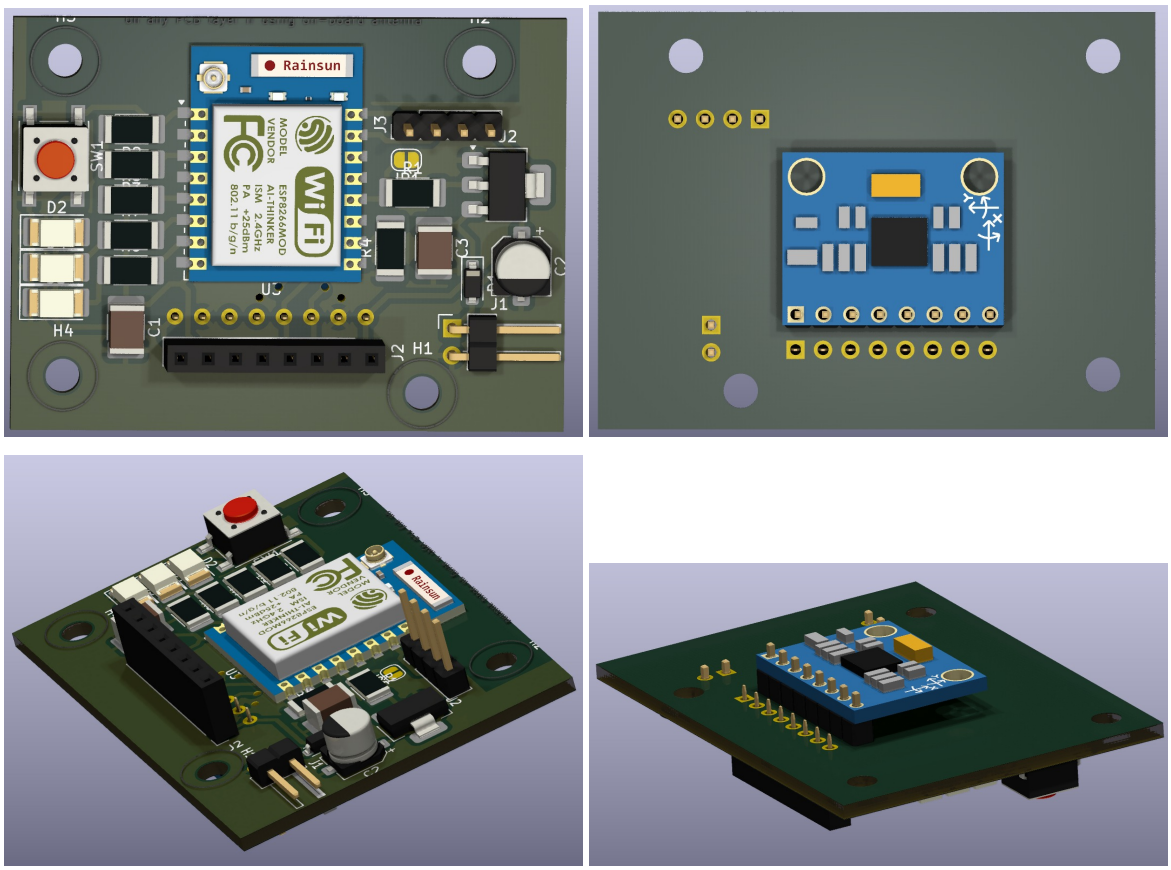


Ruteo con **plano GND** en top y vía-stitching perimetral; colocación SMD en bancos compactos para LEDs y RC de control; headers verticales para IMU/UART; keep-out RF claramente marcado.

Pinout efectivo (para trazabilidad de firmware)

Función	Pin ESP8266	Conector	Nota
UART TX	GPIO1/TXD0	J3-TX	Programación/registro
UART RX	GPIO3/RXD0	J3-RX	Programación/registro
I2C SCL	GPIO5	J2-SCL	Pull-ups en módulo IMU
I2C SDA	GPIO4	J2-SDA	—
IMU INT	GPIO14	J2-INT	Interrupciones de movimiento
LED Wi-Fi	GPIO12	D? + R?	Estado radio
LED Datos	GPIO13	D? + R?	Publicación/actividad
LED Aux	GPIO16	D? + R?	Uso general
EN (chip-enable)	EN	—	Pull-up
RESET	RST	SW1	Pulsador a GND
BOOT/FLASH	GPIO0	Pad/JP	Forzar modo programación

Registro render/fotográfico



4.4 Deep-sleep y "reset virtual" (ESP8266)

Es un **hecho técnico** que en el microcontrolador ESP8266 el modo **deep-sleep** no "despierta" el programa donde quedó: **reinicia** el chip. El despertar se logra con la unión **GPIO16 (D0) → RST**; al vencer el temporizador interno, **GPIO16 impulsa RST** y se produce un **reset por hardware** equivalente a un "software/virtual reset".

Implicancias de diseño

- **Re-hidratación de estado:** al iniciar, el firmware debe reconstruir NTP, MQTT, configuración de IMU y contadores (`seq`) de forma **idempotente**.
- **Persistencia mínima:** guardar en RTC/flash ligera: `seq`, último `ts`, flags de "shutdown limpio".
- **Tiempos de servicio:** medir y reportar `t_wake --> wifi --> publish (p95)`; los costos de asociación Wi-Fi dominan el consumo si el ciclo de sueño es corto.
- **Señalización y LWT:** publicar `status=boot/deepsleep_wakeup` y configurar **LWT** para recuperación en backend.

Requisito eléctrico

- Trazar **GPIO16 → RST** con retorno GND corto y mantener **pull-ups** de RST/EN según datasheet. No conectar GPIO16 a otros periféricos.

Prueba mínima

1. Programar `deepSleep(us)`, verificar pulso en RST desde GPIO16.
2. Comprobar re-inicio completo y publicación de `status` en ≤ 3 s tras el reset.

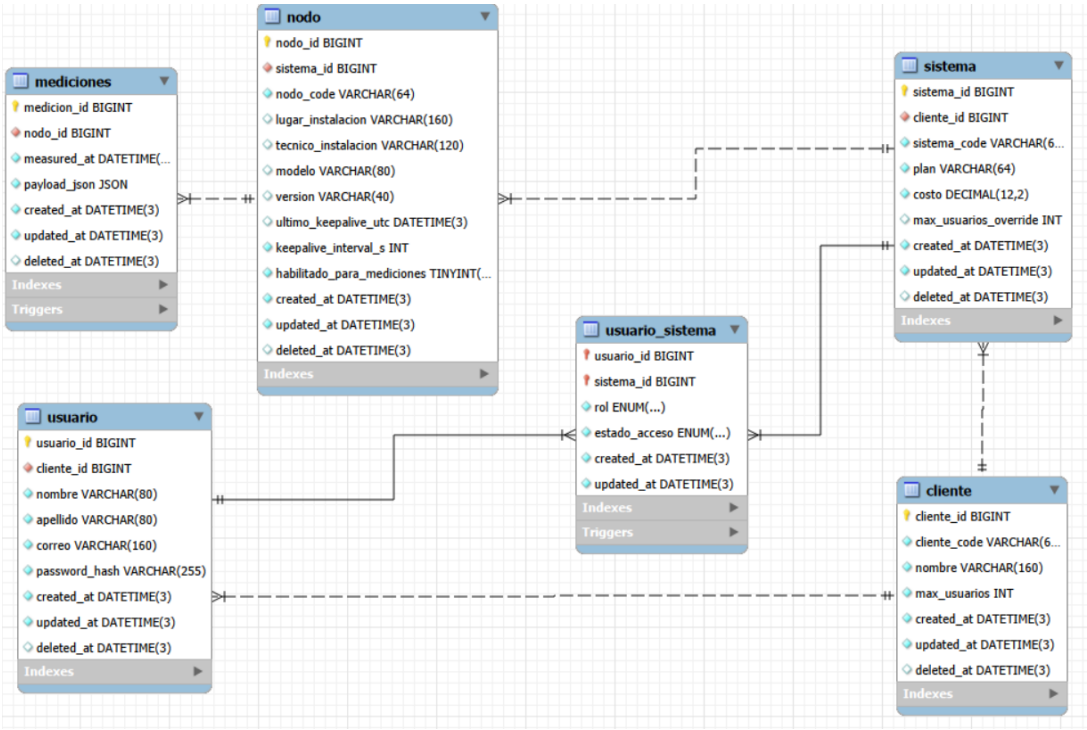
5 Base de datos MySQL

5.1 Objetivo

Almacenar mediciones por nodo con trazabilidad mínima y consultas rápidas por ventana temporal, usando UTC con milisegundos. El diagrama proporcionado es la referencia del modelo: cliente posee sistemas; cada sistema agrupa nodos; cada nodo genera mediciones; los usuarios acceden por rol mediante una tabla de relación.

5.2 Modelo lógico

Tablas de administración: `cliente`, `sistema`, `usuario`, `usuario_sistema`. Tablas operativas: `nodo` (metadatos, keep-alive, habilitación) y `mediciones` (serie temporal). Identificadores BIGINT autoincrementales y campos `created_at`, `updated_at`, `deleted_at` en `DATETIME(3)` para auditoría y soft-delete. Códigos legibles (`*_code`) únicos para direccionamiento desde flujos y UI.



5.3 Tabla central

`mediciones` guarda el instante de captura y el payload crudo en JSON, más columnas generadas para filtrar por secuencia, frecuencia de muestreo y tensión de batería. Se impone unicidad por `(nodo_id, measured_at, seq)` para lograr ingesta idempotente y evitar duplicados ante reintentos.

```
CREATE TABLE mediciones (
  medicion_id BIGINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
  nodo_id BIGINT UNSIGNED NOT NULL,
  measured_at DATETIME(3) NOT NULL,        -- UTC
  payload_json JSON NOT NULL,              -- ax, ay, az, fs_hz, seq, vbat...
  seq   INT      GENERATED ALWAYS AS (CAST(JSON_UNQUOTE(JSON_EXTRACT(payload_json,'$.seq')
  fs_hz SMALLINT GENERATED ALWAYS AS (CAST(JSON_UNQUOTE(JSON_EXTRACT(payload_json,'$.fs_hz
```

```
  vbat  DECIMAL(4,2) GENERATED ALWAYS AS (CAST(JSON_UNQUOTE(JSON_EXTRACT(payload_json,'$.v
  created_at DATETIME(3) NOT NULL DEFAULT CURRENT_TIMESTAMP(3),
  updated_at DATETIME(3) NOT NULL DEFAULT CURRENT_TIMESTAMP(3) ON UPDATE CURRENT_TIMESTAMP
  UNIQUE KEY uk_med (nodo_id, measured_at, seq),
  KEY k_nodo_ts (nodo_id, measured_at),
  CONSTRAINT fk_med_nodo FOREIGN KEY (nodo_id) REFERENCES nodo(nodo_id)
) ENGINE=InnoDB;
```

5.4 Ingesta (idempotente)

El flujo de Node-RED inserta usando `INSERT ... ON DUPLICATE KEY UPDATE`, con tres parámetros: `nodo_id` (identidad del dispositivo), `ts_ms` (marca de tiempo en milisegundos convertida a `DATETIME(3)` UTC) y `payload_json` (mensaje validado).

```
INSERT INTO mediciones (nodo_id, measured_at, payload_json)
VALUES (?nodo_id, FROM_UNIXTIME(?ts_ms/1000), ?payload_json)
ON DUPLICATE KEY UPDATE payload_json = VALUES(payload_json),
  updated_at = CURRENT_TIMESTAMP(3);
```

5.5 Consulta típica

Para graficar una ventana temporal por nodo se filtra por `nodo_id` y rango de `measured_at`, aprovechando el índice `k_nodo_ts`.

```
SELECT measured_at, payload_json
FROM mediciones
WHERE nodo_id = ? AND measured_at BETWEEN ?ini AND ?fin
ORDER BY measured_at;
```

5.6 Particionado y operación

Se recomienda particionar `mediciones` por rango mensual de `measured_at` para mantener índices compactos y facilitar la retención de crudos (p. ej., 180 días). El acceso desde Node-RED debe ir por TLS y con privilegios mínimos; la UI solo requiere SELECT.

Roles propuestos: - `ingestor` (INSERT/SELECT en `mediciones` y SELECT en `nodo`) - `dashboard` (SELECT) - `admin_db` (DDL/particiones).

6 Firmware

6.1 Objetivo y alcance

Adquirir datos crudos de la **MPU6050** con temporización estable y publicarlos por **MQTT** con marca de tiempo en milisegundos UTC. El microcontrolador **no filtra ni procesa**: solo mide y publica. Todo posprocesado (filtros, espectros, KPIs) corre en el servidor.

6.2 Justificación de frecuencias de muestreo

Las vibraciones sísmicas estructurales de interés se concentran típicamente entre **0.5 y 30 Hz**. Para capturarlas sin distorsión y con margen de análisis, se adopta un **factor de sobre-muestreo** $\geq \mathbf{5}\times$ respecto de la frecuencia máxima de interés:

$$F_s \geq k \cdot f_{\max} \quad \text{con} \quad k \in [5, 10]$$

Con $f_{\max} = 30,\text{Hz}$ y $k = 5$ resulta $F_s \geq 150\,\text{Hz}$. Se fija $F_s = 200\,\text{Hz}$ por las siguientes razones:

1. deja **banda útil hasta 100 Hz** (Nyquist) para cubrir picos transitorios o impactos por encima de 30 Hz;
2. mejora la estimación de espectros y tiempos característicos al contar con más muestras por ciclo;
3. aporta **robustez a jitter** del temporizador y a pequeñas derivas del bus I2C sin perder cobertura;
4. mantiene un **coste de red razonable** cuando se publica en **bloques** (sin filtrado en el nodo), dejando todo el posprocesamiento al servidor.

Parámetros operativos asociados:

- `FS_HZ = 200` define la frecuencia de adquisición en el microcontrolador.
- `PUB_BLOCK_SAMPLES` controla el tamaño del bloque publicado y, por ende, la **latencia** y el **ancho de banda** efectivo. Por ejemplo, `PUB_BLOCK_SAMPLES = 100` implica bloques de **0.5 s** a 200 Hz.

Este esquema garantiza resolución suficiente para la banda 0.5–30 Hz con margen de análisis y sin carga de cómputo en el firmware, que solo **mide y publica** datos crudos.

6.3 Temporización y ejecución

- **Temporizador periódico**: genera una interrupción cada $T = 1/FS_{\text{HZ}}$ para disparar la toma de muestra.
- **ISR mínima**: coloca un "trigger" en una cola.
- **Tarea de adquisición**: toma el "trigger", realiza **lectura burst I2C** del MPU6050, **aplica calibración** y escribe la muestra cruda en un **ring buffer**.
- **Tarea de publicación**: empaqueta **bloques de N muestras** y publica por MQTT (o los deja en cola local si no hay red).
- **Deep-sleep (ESP8266)**: `GPIO16 --> RST` produce reset; en `setup()` se rehidrata NTP/MQTT/IMU y se continúa la secuencia `seq`.

6.4 Calibración del MPU6050 (compensación de offset)

Intención: remover sesgo de fábrica y de montaje.

1) Nodo inmóvil 10–20 s.

2) Capturar N muestras.

3) Calcular medias por canal. Acel: `bias_a[i]=mean(acc[i])` ajustando Z a $\pm$ 1 g; Giro: `bias_g[i]=mean(gyro[i])`. Guardar en NVS/EEPROM junto con `temp_ref`. En operación: `x_corr = x_raw - bias - k_temp*(temp - temp_ref)` si se calibra térmicamente. **Criterio de aceptación interno:** $\text{var}(\text{acc}) < 0.01$ g.

6.5 MQTT: tópicos y payload

- Tópicos:
  - `sismo/<site>/<nodeId>/data` (QoS1, no retenido)
  - `sismo/<site>/<nodeId>/status` (QoS1, retenido, LWT=`offline`)

– `sismo/<site>/<nodeId>/cfg` (QoS1, retenido)

`ts_ms` significa "timestamp en milisegundos desde época UNIX (UTC)". En modo por bloques, `ts_ms` es el **instante de la primera muestra** del bloque. La separación temporal entre muestras del bloque se infiere con `fs_hz` o con `dt_us` opcional.

Payload propuesto (bloques crudos, sin filtrado):

```
{
  "ts_ms": 1730158805123,      // UTC de la PRIMERA muestra del bloque
  "fs_hz": 200,                // frecuencia de adquisición en el MCU
  "seq": 15231,                // contador de bloque (monótono, persiste entre resets)
  "n": 100,                    // muestras por canal en este bloque
  "ax": [ ... 100 valores ... ],
  "ay": [ ... 100 valores ... ],
  "az": [ ... 100 valores ... ],
  "gx": [ ... 100 valores ... ],
  "gy": [ ... 100 valores ... ],
  "gz": [ ... 100 valores ... ],
  "temp_c": 27.5,              // temperatura IMU durante el bloque (media)
  "vbat": 3.92                 // tensión de batería al cerrar el bloque
}
```

Justificación de parámetros:

- `ts_ms` y `fs_hz` permiten reconstruir el **tiempo exacto de cada muestra** en servidor.
- `seq` asegura **idempotencia** extremo a extremo junto al `UNIQUE(nodo_id, measured_at, seq)` en DB.
- `n` define tamaño del bloque para decodificación eficiente.
- `temp_c` habilita compensaciones térmicas en análisis.
- `vbat` permite correlacionar eventos con caída de tensión.

6.6 Código de referencia (ESP32 FreeRTOS; crudo → bloque → MQTT)

Intención: ISR mínima, adquisición I2C con calibración, publicación por bloques, sin filtros en MCU.

```
// ---- Parámetros clave ----
constexpr uint16_t FS_HZ = 200;                 // adquisición cruda
constexpr uint16_t PUB_BLOCK_SAMPLES = 100;     // tamaño de bloque (0.5 s)
constexpr uint32_t T_US = 1000000UL / FS_HZ;    // periodo en microsegundos

struct Calib { float ax, ay, az, gx, gy, gz, t_ref; } gCal;
struct Sample { uint32_t t_us; float ax, ay, az, gx, gy, gz; float temp; };

static QueueHandle_t qTrig;
static hw_timer_t* timer0 = nullptr;

// --- ISR: solo notifica tiempo ---
void IRAM_ATTR onTimerISR() {
  BaseType_t hp = pdFALSE;
  uint32_t t = (uint32_t) esp_timer_get_time();
  xQueueSendFromISR(qTrig, &t, &hp);
  if (hp) portYIELD_FROM_ISR();
```

```
}

// --- Buffer de muestras crudas ---
static Sample rb[PUB_BLOCK_SAMPLES];
static volatile uint16_t rb_idx = 0;

// --- Tarea de adquisición: I2C + calibración ---
void taskAcquire(void*){
  for(;;){
    uint32_t t_us;
    if (xQueueReceive(qTrig, &t_us, portMAX_DELAY)) {
      int16_t ra[3], rg[3]; float temp;
      mpu6050_read_raw_burst(ra, rg, &temp);        // lectura atómica

      Sample s;
      s.t_us = t_us;
      s.ax = ra[0]*ACC_SCALE - gCal.ax;
      s.ay = ra[1]*ACC_SCALE - gCal.ay;
      s.az = ra[2]*ACC_SCALE - gCal.az;
      s.gx = rg[0]*GYR_SCALE - gCal.gx;
      s.gy = rg[1]*GYR_SCALE - gCal.gy;
      s.gz = rg[2]*GYR_SCALE - gCal.gz;
      s.temp = temp;

      rb[rb_idx++] = s;
      if (rb_idx >= PUB_BLOCK_SAMPLES) {
        rb_idx = 0;
        notify_block_ready();                       // señal a publicador
      }
    }
  }
}

// --- Tarea de publicación: empaqueta y envía bloque ---
void taskPublish(void*){
  static uint32_t seq = load_seq_from_nvs();
  for(;;){
    if (block_ready()){
      // construir JSON (o CBOR) con ts_ms de la primera muestra:
      uint64_t ts_ms = to_unix_ms(rb[0].t_us);   // NTP base + offset local
      JsonDoc doc;
      doc["ts_ms"] = ts_ms;
      doc["fs_hz"] = FS_HZ;
      doc["seq"]   = seq++;
      doc["n"]     = PUB_BLOCK_SAMPLES;
      append_arrays_from_rb(rb, PUB_BLOCK_SAMPLES, doc); // ax..gz, temp/vbat

      if (net_ok()) mqtt_publish_qos1("sismo/siteX/nodeY/data", doc);
      else spiffs_enqueue(doc);

      save_seq_to_nvs(seq);
```

```
    } else {
      vTaskDelay(pdMS_TO_TICKS(1));
    }
  }
}

void setup(){
  // NTP (UTC), MQTT con LWT retenido, carga de calibración y seq
  load_calibration(&gCal);

  qTrig = xQueueCreate(64, sizeof(uint32_t));
  timer0 = timerBegin(0, 80, true);           // 80 MHz / 80 = 1 MHz
  timerAttachInterrupt(timer0, &onTimerISR, true);
  timerAlarmWrite(timer0, T_US, true);
  timerAlarmEnable(timer0);

  xTaskCreatePinnedToCore(taskAcquire, "acq", 4096, nullptr, 3, nullptr, 0);
  xTaskCreatePinnedToCore(taskPublish, "pub",  6144, nullptr, 2, nullptr, 1);
}
```

ESP8266 (sin FreeRTOS): sustituir `hw_timer_t` por **Timer1**, y `notify_block_ready()/block_ready()` por una **cola circular** en "loop cooperativo". El resto del flujo se mantiene igual.

6.7 Deep-sleep y "reset virtual"

En ESP8266, `deepSleep(us)` despierta por **pulso de GPIO16 a RST**, equivalendo a un **reset por hardware**. El arranque debe: sincronizar NTP, reconectar MQTT publicando `status=deepsleep_wakeup`, restaurar `seq`, reconfigurar IMU y enviar/barrer cualquier bloque encolado en SPIFFS.

7 Cronograma y plan de la siguiente iteración

Objetivos verificables

- Crear un prototípo funcional de la placa de desarrollo v0.0.2 (SMD)
- Implementar cola local en firmware con reenvío y prueba de caída 10 min.
- Integrar TLS mutual en Mosquitto y validar ACLs por tópico.
- Completar dashboard con espectro en ventana móvil y alarmas.
- Ensayo de calibración de 2 nodos y reporte de error con incertidumbre.

8 Solicitudes al comité/directores (bloqueadores)

9 Bibliografia y Referencias