04 開發者 / 整合

連線與協議

> 📅 最後更新：2026-03-14 | 📌 負責人：KC

本文件涵蓋 MES I/O Gateway 使用的網路基礎設施與通訊協議。

01

1. 網路抽象層（`NetworkAdapter.h`）

Gateway 透過統一抽象層支援雙堆疊網路連線（Ethernet 和 Wi-Fi），確保高階服務（Web Server、MQTT、Modbus）與實體媒介解耦。

執行期模式選擇（有線優先）

Gateway 使用「有線優先（Wired-First）」偵測邏輯來降低現場配置複雜度：

- **Ethernet 優先**：開機時系統檢查實體 Ethernet 連結（`network.hasEthernetLink()`）。偵測到網線即初始化 Ethernet（DHCP/Static），同時**停用 Wi-Fi** 以減少干擾與電源雜訊。
- **Wi-Fi 備援**：若未偵測到 Ethernet 網線或初始化失敗，系統會嘗試連線已設定的 Wi-Fi 存取點。
- **介面狀態**：Web UI 的設定面板根據執行期偵測結果顯示目前使用的介面（"Ethernet" 或 "WiFi"）。
- **統一韌體 (v5.0)**：Ethernet 和 WiFi 堆疊編譯到同一個二進位檔（`env:opta`）。這實現了「部署中立性」——同一韌體可部署至任何 Opta 裝置，自動適應可用的實體基礎設施。

核心介面

- `NetClient` / `NetServer`：網路 socket 的共用型別定義（相容 Ethernet 和 WiFi）。
- `NetworkAdapter` **類別**：多媒介連線的統一處理器。

- `hasEthernetLink()`：硬體端檢查實體網線。
- `activeNetwork()`：回傳 `NET_ETHERNET`、`NET_WIFI` 或 `NET_NONE`。
- `beginEthernet(mac, ip, gw, sn)`：初始化有線靜態 IP。
- `beginEthernetDHCP(mac)`：初始化有線 DHCP。
- `setEthernetIP(ip, gw, sn)`：動態更新有線介面。
- `beginWiFi(ssid, pass)`：初始化無線連線（v5.4.7 從 `begin` 改名以避免歧義）。
- `disconnectWiFi()`：中斷無線連線（v5.4.7 從 `disconnect` 改名）。
- `setAPMode(enabled)`：標記系統進入存取點模式。
- `isAPMode()`、`isWiFi()`、`isEthernet()`：介面識別。

- **`NetworkConfig`**：持久化 IP 設定、DHCP 偏好、Wi-Fi 憑證。自 v5.1 起，**出廠預設改為 DHCP 啟用**以利無縫初始探索。

1.4 初始化順序 (Industrial Priority)

為支援韌性，`setup()` 順序優先載入配置與安全機制檢查，在任何網路流量開始前完成。自 **v5.4.24** 起，初始化順序經過重構，將 AP Mode 的資源與所有潛在的硬體衝突（DMA/SPI/中斷）隔離。

1. **`LocalConfig::begin()`**: High-priority mount of QSPI Flash. Sets the ground for all subsequent operations.
2. **Configuration Check**: `localConfig.loadLocal(config)` is called once.
3. **AP Mode Pre-emption (v5.4.24 Critical Fix)**: Checks `config->network.forceApMode`. If `true`:
   - **Resource Gating**: Skip initialization of Modbus RTU, I/O Manager, Virtual Devices, and MQTT.
   - **Isolation**: Call `WiFi.end()` followed by a `delay(2000)` to reset the radio module.
   - **QSPI Release (Final Blocker Fix)**: Call `localConfig.unmount()` to explicitly close LittleFS and the QSPI BlockDevice. This is necessary because the Murata 1DX radio and QSPI Flash share internal bus resources on the Opta's STM32H747 core.
   - **Hotspot Start**: Call `WiFi.beginAP("MES-Gateway-Setup", "12345678")`.

- **Early Exit**: Start `server.begin()` and **return** from `setup()`.
- *Result*: This multi-stage isolation prevents the `WiFi.beginAP()` hang by ensuring no other high-load hardware managers are holding onto shared resources during the radio's sensitive initialization phase.

4. **Standard Mode Path**: If not in AP mode, proceed with:
   - `checkSafeModeButton()`: Component-level fail-safe.
   - I/O Manager, Modbus RTU, MQTT initialization.
   - `network.hasEthernetLink()` detection and network stack start.
   - Main loop execution.

1.5 AP Mode Web Server（安全區域）

當系統處於 AP Mode 時，Web Server 以最小路由集初始化以避免記憶體壓力。技術人員可透過 **192.168.3.1** 存取 dashboard 進行 WiFi SSID 掃描與憑證輸入。

1.6 建置系統需求（v5.4.1 發現）

要在統一韌體中啟用 WiFi 管理子樹（AP Mode、掃描、靜默配置），需在建置環境配置（如 `platformio.ini`）中定義以下巨集：

- **`USE_WIFI`**：啟用 `main.cpp` 中 `/api/wifi/*` 和 `/api/network/apmode` 端點的路由。

若未定義此旗標，Gateway 將預設為「Ethernet-Primary」狀態，WiFi 管理邏輯不會編入二進位檔，以在非無線部署中節省記憶體。

02

2. Wi-Fi 支援 (v4.9.5)

Arduino Opta Wi-Fi 變體已完整支援，具備執行期配置能力。

功能特性

- **統一邏輯**：程式碼中 50+ 個 network client/server 實例已重構為使用 `NetClient` / `NetServer` 抽象。
- **Dashboard UI**：設定 tab 中有專用的「Wi-Fi 設定」卡片，支援 SSID 掃描、憑證輸入和連線管理。
- **API Support**:
  - `GET /api/wifi/status`: Reports connection state, SSID, IP, and RSSI.
  - `GET /api/wifi/scan`: Triggers a hardware scan for available networks (Note: Blocked while Ethernet is active).
  - `POST /api/wifi/save`: (v5.3 New) Updates credentials in Flash without initiating a connection. Optimized for silent provisioning while on Ethernet.
  - `POST /api/wifi/connect`: Updates credentials and attempts a heartbeat connection.

2.1 「控制路徑」挑戰 (v5.2/v5.3)

實際工業測試發現一個關鍵硬體限制：Arduino Opta 在 Ethernet 連結啟用時無法可靠地掃描 WiFi。

- **硬體限制**：在高負載多媒介情境下，掃描常回傳錯誤。
- **控制路徑風險**：若系統主動解除 Ethernet 以執行 WiFi 掃描，會同時中斷 Web UI 管理介面，對操作員形成「死路」。
- **v5.3 解法：靜默配置**：
  - **「僅儲存 WiFi」UI**：允許透過 Ethernet 輸入憑證但不立即連線。
  - **AP Mode 備援**：若未偵測到 Ethernet 且未設定 WiFi，裝置自行託管 SSID（192.168.3.1）供設定。
  - **重啟驗證**：引導式工作流觸發裝置重置以驗證無線連線，確保操作員知道設定是否成功。
- **v5.4 手動 AP Mode 切換**：當 Ethernet 啟用但使用者需要掃描 WiFi 時，系統實作手動 `POST /api/network/apmode` 觸發。無論當前連結狀態，強制裝置進入 AP 設定狀態（MES-Gateway-Setup），啟用被活躍 Ethernet 堆疊阻擋的完整硬體掃描功能。

2.2 WiFi 模組韌體依賴 (v5.4.7 重大)

實際部署發現 Arduino Opta 的 WiFi 無線電模組 (Nina-W10) 需要儲存在專用 QSPI 分區的特殊韌體檔案系統。

- **症狀**：Serial 錯誤 `Failed to mount the filesystem containing the WiFi firmware`。
- **原因**：無線電韌體不是標準應用程式二進位檔的一部分，可能在批量格式化或恢復出廠設定時遺失。
- **重要警告 (v5.4.12)**：即使 `WiFiFirmwareUpdater` 回報成功，若存在 **QSPI 分區衝突**，錯誤仍可能持續。應用程式的 LittleFS 資料儲存可能與 WiFi 分區重疊。
- **解法**：使用 Arduino IDE 的 **`WiFiFirmwareUpdater`** sketch。若錯誤持續，先使用 **`QSPIFormat`** 範例 sketch 重設分區表。
- **重要**：`QSPIFormat` 會清除無線電韌體。您**必須**在 `QSPIFormat` *之後立即*再次執行 `WiFiFirmwareUpdater`，且在*重新上傳專案之前*。**注意**：PlatformIO 目前尚未原生支援 Opta 核心的此特殊分區管理。

---

3. MQTT 整合

MQTT 提供事件驅動、低延遲的遙測功能，用於 MES/SCADA 系統。

實作：`MQTTClientManager` (v5.9)

- **Library**: `arduino-mqtt` (256dpi) [取代 PubSubClient]。
  - **原因**: 原生支援 QoS 0/1/2 publish 與 subscribe，支援重要控制指令的 Exact-delivery。
- **架構優化 (Resilience)**:
  - **Header 命名衝突迴避 (Critical)**: 為了避免與 library 內部的 `MQTTClient.h` 衝突，Gateway 的包裝類別已重命名為 **`src/MesMQTT.h`**。
  - **陷阱**: `arduino-mqtt` 的 `MQTT.h` 會 `#include "MQTTClient.h"`。若專案中存在同名檔案，編譯器會優先載入本地檔案而非 library 內部的類別定義，導致 `'MQTTClient' does not name a type` 錯誤。
- **靜態 Callback (Static Handler)**: 使用靜態函式 `mqttMessageHandler` 接收進站訊息，避免了 Lambda 在 `onMessageAdvanced` 重載 (std::function vs function pointer) 時引發的

編譯歧義。

- **全域預設 QoS**: 預設採用 **QoS 2** (Exactly Once)，大幅提升控制指令的可靠性。
- **Topic Hierarchy**:
  - `mes/gateway/status`: LWT (online/offline)。
  - `mes/gateway/di/{index}`: 狀態變化推送。
  - `mes/gateway/do/{index}`: 狀態變化推送。
  - `mes/gateway/rules/{index}/trigger`: 規則觸發通知。
  - `mes/gateway/{clientId}/cmd/#`: 遠端指令訂閱。

3.2 連線生命週期與韌性 (v5.9)

為防止主應用程式迴圈在網路不穩定期間阻塞，MQTT 管理器實作了自動化生命週期：

1. **初始化**：`begin()` 綁定 broker 並註冊靜態 callback。此步驟在 `setup()` 早期執行。
2. **網路就緒觸發（延遲連線）**：`connect()` 被刻意從 `setup()` 移除，改為在 `loop()` 中偵測到有效的 Ethernet/WiFi IP 地址及 LinkUp 狀態後才觸發。
   - **原因**：防止在網路堆疊初始化過程中嘗試 TCP 連線導致的 Block 或 HardFault，並確保所有的「Connecting to...」日誌能完整輸出到已穩定的 Serial Console，方便現場除錯。
3. **迴圈與重試**：
   - 若斷線，系統使用**指數退避**（10s、20s、30s）重試。
   - **自動停用（重大）**：若連續 **3 次**連線失敗，明確設定 `enabled = false`。
   - **故障排除信號**：自動停用時，Serial 日誌印出：`MQTT: Auto-disabled after 3 failures. Fix config in Web UI.` 在透過 API 更新配置之前不再嘗試連線，節省 CPU 週期並防止網路堆疊飽和。

3.3 MQTT 故障排除指標

- **開機無日誌**：若 Serial monitor 未顯示「MQTT: Broker」訊息，透過 `GET /api/config` 確認 `config->mqtt.broker` 非空。
- **認證失敗**：檢查 Serial 日誌中的 `rc`（Return Code）：
  - `rc=4`：使用者名稱/密碼錯誤。

  - `rc=5`：未授權。
- **網路失敗**：`err=-1` 通常表示逾時或 broker IP 不可達。

3.4 MQTT TCP IO 通道 (v5.6.5+)

Gateway 可將特定 MQTT topic 視為邏輯層級的 IO 點。

- **入站**：訂閱已設定的感測器。對 payload/JSON 欄位與匹配值進行評估。
- **出站**：當觸發發生時發佈填空樣板。
- **詳細技術總覽**：見 `spec-tcp-io-channels.md`。

03

4. Modbus TCP 伺服器 (v4.5)

為了與傳統工業系統互通，Gateway 在 Port 502 作為 Modbus TCP Server 運作。

架構最佳化

為防止「Port 80 飢餓」（Modbus 流量阻塞 Web Dashboard），伺服器使用嚴格的非阻塞模式：

- **無延遲**：移除請求處理迴圈中的阻塞 `delay()` 呼叫。
- **機會性輪詢**：每個系統週期恰好處理一個封包。
- **Socket 效率**：明確清空並關閉 socket 以釋放資源給 HTTP 流量。

暫存器映射（v5.6.5 擴充）

- **FC 01/02**：讀取實體 DI (0-7) 和 DO (0-3)。
- **FC 03/04**：讀取縮放後的類比輸入、系統變數（`VAR[0..15]`）及 **TCP IO 輸入 (Registers 100-199)**。
- **FC 05/06**：寫入數位輸出、系統變數及 **TCP IO 輸出 (Registers 200-299)**。

4.1 Modbus TCP IO 通道 (v5.6.5)

Gateway 將內部 TCP IO 狀態映射到專用的 Holding Register 區塊：

- **輸入 (Registers 100-199)**：映射到 TCP IO 輸入通道的 `currentState`。
- **輸出 (Registers 200-299)**：寫入 `1` 到這些暫存器會觸發對應的 TCP IO 輸出動作（如發佈 MQTT 指令）。
- **自動分配**：配置 API 處理暫存器分配以防止與其他系統變數衝突。

04

5. 開發進度

- **網路抽象層**：完成。
- **MQTT (Pub/Sub)**：完成並驗證 (v4.9.3)。
- **Wi-Fi UI 與 API**：完成並驗證 (v4.9.5)。
- **v5.3 WiFi 配置**：UI 重構完成（舊版掃描改為引導式手動輸入）。
- **Modbus TCP**：邏輯已實作；Socket 管理已最佳化 (v4.5)。
- **Modbus RTU (RS485)**：Master/Slave 支援已整合 (v4.3)。

05

6. Modbus RTU (RS485) 管理器 (v4.3)

RTU 管理器利用 Opta 內建的 RS485 終端與傳統工業硬體介面。

6.1 雙角色能力

- **Master 模式**：實作非阻塞的循環輪詢迴圈，支援最多 8 個外部 slave（`MAX_MODBUS_DEVICES = 8`）。
- **Slave 模式**：透過 16-bit Holding Registers 的位元向外部 PLC/SCADA 暴露 Gateway 狀態。

6.2 技術規格

- **介面**：`Serial1`（RS485）。
- **預設配置**：19200 Baud、8N1、標準 CRC-16。
- **暫存器映射（Slave 模式）**：
  - Addr 0：DI 狀態位元遮罩。
  - Addr 1：DO 狀態位元遮罩。
  - Addr 2-9：縮放後 AI 電壓值（×100）。
  - Addr 10-11：32-bit 系統運行時間（秒）。

6.3 RS485 韌性模式 (v5.8.5)

為確保從高密度量測裝置（如 Finder 6M）可靠取得資料，系統實作了專用的輪詢策略。

分段讀取批次策略

某些 Modbus 裝置對單次批次可讀取的暫存器數量有限制。超過此限制（如請求 22 個暫存器但限制為 18）會觸發逾時。

- **實作**：`PowerMonitor` 管理器執行兩階段讀取：
  1. **主要（Registers 192-209）**：高頻資料（V、I、P、Hz）。
  2. **擴充（Registers 210-213）**：累計計數器（Energy±）。

displayMask 與資料擷取的解耦

為平衡 UI 效能與資料完整性，顯示偏好與背景擷取解耦。

- **displayMask**：由 Web UI 用來隱藏/顯示量測卡片。
- **背景完整性**：韌體始終從 slave 輪詢所有欄位，確保 Data Logger 和 API 取得連續資料流，無論 dashboard 上顯示什麼。

UI 防禦邏輯

RS485 配置 modal 附加到 `document.body` 以避免在巢狀 CSS 版面中被裁剪。RS485 區塊的可見性邏輯以資料存在與否為範圍，確保在主機和擴充模組 tab 之間切換時元素保持可見。

統一韌體建置 (v5.0)

自 v5.0 起，Ethernet 和 WiFi 的獨立建置環境已廢棄。單一的 `env:opta` 建置針對 Opta WiFi 硬體，動態管理兩個介面。

- **依賴**：`Ethernet.h` 和 `WiFi.h` 包含在同一個二進位檔中。
- **邏輯**：有線連線始終優先。WiFi 僅在開機時未偵測到 Ethernet 網線時才初始化。

建議的上線工作流程

1. **實體優先**：新裝置直接插入 Ethernet 網線。裝置會偵測連結、使用 DHCP，立即可被發現。
2. **配置**：透過 Ethernet 存取 dashboard 配置站點專屬的靜態 IP 或 Wi-Fi 憑證。
3. **無線轉換**：WiFi 配置完成後，拔除 Ethernet 網線觸發自動 WiFi 切換（就地切換，無需重啟）。
4. **簡化**：不需要獨立的「WiFi」或「Ethernet」韌體；單一「統一」韌體處理兩種硬體配置。

06

7. 網路容錯架構 (v5.5)

Gateway 實作 **雙向就地切換** Ethernet 和 WiFi，零重啟。

7.1 開機最佳化

Path	Time	Key Technique
Ethernet	~8s	5s PHY link check + DHCP
WiFi	~14s	`ENC_TYPE_CCMP` skip-scan + 2-stage retry

- **WiFi** `ENC_TYPE_CCMP`: Passing WPA2 encryption type as 3rd arg to `WiFi.begin()` skips `scanNetworks()` (saves ~10s).
- **2-Stage WiFi Retry**: Attempt 1 (3s warm-up, expected failure) + Attempt 2 (10s real connection).

- **Ethernet 5s PHY Timeout**: Cold-boot PHY (LAN8742A) needs >2s to detect cable. Default 60s is excessive; 5s is sufficient.

7.2 容錯狀態機

7.3 就地 Ethernet 回復探測

每隔 `FO_ETH_PROBE_INTERVAL` (10 秒)，系統**無需重啟**即檢查 Ethernet 網線是否存在：

1. `_onBeforeEthernet()` — Release WiFi server (port 80) + unmount QSPI
2. `WiFi.disconnect()` + `WiFi.end()`
3. `Ethernet.begin(mac, dummy_ip, 2000)` — 2s link check
4. **Cable found** → `Ethernet.begin(mac)` DHCP → switch to Ethernet → `_onEthernetRestored()`
5. **No cable** → `WiFi.begin(ssid, pw, ENC_TYPE_CCMP)` → return to WiFi

> [!IMPORTANT] Each probe causes ~~3s WiFi downtime. The 10s interval balances detection speed~~ (5s average) with ~70% WiFi availability.

7.4 回呼介面

Callback	Trigger	Responsibility
`_onBeforeWiFi`	Before Eth→WiFi switch	Release `EthernetServer` port 80 + unmount QSPI
`_onWiFiRestored`	WiFi connected	Set `wifiServerStarted = false` (loop starts server)
`_onBeforeEthernet`	Before WiFi→Eth probe	Release `WiFiServer` port 80 + unmount QSPI
`_onEthernetRestored`	Ethernet DHCP success	Restart `EthernetServer` on port 80

7.5 硬體限制

- **雙堆疊衝突**：Arduino Opta (STM32H747) 無法同時運作 Ethernet 和 WiFi。在 WiFi 啟用時呼叫 `Ethernet.begin()` 會導致 mbed 懸停。
- **PHY 暫存器存取**：`Ethernet.linkStatus()` 在 `Ethernet.end()` 之後會懸停——必須重新初始化後才能檢查連結。
- **QSPI 化流排共享**：Murata 無線電和 QSPI Flash 共享內部 SPI 化流排。在 WiFi 操作前必須卸載 LittleFS。

7.6 容錯時間常數

Constant	Value	Purpose
`FO_LINK_DEBOUNCE`	3s	Ethernet disconnect debounce
`FO_WIFI_TIMEOUT`	20s	WiFi connection timeout
`FO_ETH_PROBE_INTERVAL`	10s	Ethernet recovery probe interval
`FO_RECOVERY_DELAY`	5s	Ethernet recovery debounce
`FO_RETRY_INTERVAL`	60s	WiFi retry after total failure

IP 分配注意事項

在生產環境（如 `192.168.50.*`）中，靜態 IP 通常在站點專屬設定時逐一分配（`.101`、`.105`、`.9`）。Gateway 的預設靜態 IP 應視為「已知入口」而非永久分配；最終 IP 應透過 Web UI 設定並持久化到 Flash。

07

9. 網路韌性與安全機制 (v5.1)

為防止「網路隔離」（裝置因無效靜態 IP 或網路變更而無法連線），整合了兩個安全機制：

9.1 硬體觸發 DHCP 重設

在開機前 3 秒內按住 **USER 按鈕**（在 Opta 核心中符號定義為 `BTN_USER`）將覆寫所有儲存設定並強制裝置進入 DHCP 模式。

- **硬體特性**：按鈕為低電位有效（內部 pull-up）。偵測需要 `pinMode(BTN_USER, INPUT)`。
- **消除彈跳**：開機時使用 **100ms 延遲**確保按鈕狀態穩定後才確認進入 Safe Mode。
- **持久化**：此變更自動透過 `LocalConfig.saveLocal()` 儲存到 QSPI Flash 上的持久化 `config.json`。

9.2 自動連線備援

若裝置嘗試以靜態 IP 初始化但未能在定義的逾時時間內（WiFi 15 秒）建立連結，它會：

1. 將作用中配置物件的 `network.dhcpEnabled` 切換為 `true`。
2. **持久化到 QSPI Flash**：執行 `LocalConfig.saveLocal()` 確保裝置在後續重啟時保持 DHCP 模式。
3. **重啟堆疊**：以 DHCP 模式再次呼叫 `network.begin()`。

此「黏性備援」確保裝置被移動到新網路且連線失敗後，持續透過 DHCP 可過，直到使用者明確重新配置，避免重複連線失敗的迴圈。

08

10. 現場除錯與故障排除

當 Web Dashboard 在部署後無法連線時：

10.1 Serial 監控

- **工具**：`pio device monitor --baud 115200`
- **啟動日誌**：尋找 `Ethernet... OK` 或 `WiFi... OK` 以及印出的 IP 地址。
- **重設技巧**：若裝置無回應或 Serial 無輸出，切換 DTR/RTS 或使用 1200-baud 重設技巧：`stty -f /dev/cu.usbmodemXXXX 1200; sleep 2`（強制 Opta 進入 bootloader/重設模式）。

10.2 命令列監控（非 TTY 環境）

在無互動終端的環境中（如 CI/CD 日誌、AI Agent、受限 SSH），使用 `stty` 和 `cat` 直接存取：

1. **識別 Port**：`ls /dev/cu.usbmodem*`（macOS）或 `ls /dev/ttyACM*`（Linux）。
2. **初始化與讀取**：

```bash
stty -f /dev/cu.usbmodemXXXXX 115200 raw -echo && cat /dev/cu.usbmodemXXXXX
```

此方法繞過對 TTY 監控器的需求，允許背景式日誌記錄。

主機端準備清單

- **子網對齊**：確保電腦的 IP 與 Opta 在同一範圍。
  - **預設情境**：若 Opta 預設為 `192.168.0.99`，您的 PC 必須在 `192.168.0.X`。
  - **已配置情境**：若設定 Opta 為 `192.168.50.9`，您的 PC 必須在 `192.168.50.X`。
  - **子網不匹配**：使用 `ifconfig`（macOS/Linux）或 `ipconfig`（Windows）驗證自己的 IP。
- **實體連結**：檢查 Opta 的 RJ45 埠 LED 指示燈。無燈通常表示網線故障或缺少電源/交換器連線。
- **Port 衝突**：確保網路上沒有其他裝置使用相同的靜態 IP（`.99`、`.101`、`.9`）。

- **探索（ARP 掃描）**：若因持久化不匹配而不知 IP，使用 ARP 表尋找裝置：

  ```bash
  arp -a | grep -E "192\.168\."
  ```

  尋找以 `aa:bb:cc` 開頭的 MAC 地址（或 Opta 標籤上印的製造商前綴）。

瀏覽器快取與 Socket 狀態

- **IP 變更延遲**：瀏覽器常「掛在」對前一個 IP（如 `.99`）的逾時連線。明確關閉分頁或執行強制重新整理（`Ctrl/Cmd + Shift + R`）。
- **Socket 重設**：某些情況下，IP 變更後需要對 Opta 進行實體電源重置以清除內部網路狀態機。

「LocalConfig 覆寫」陷阱

問題：即使上傳了包含修改後靜態 IP（如從 `192.168.0.99` 改為 `192.168.50.9`）的新韌體，裝置可能仍然以**舊** IP 開機。**原因**：`LocalConfig` 載入優先級（Flash Storage）高於硬編碼韌體預設。若裝置先前已配置並儲存，重啟時會還原舊配置，覆寫新的硬編碼值。**偵測**：檢查 Serial 日誌中的 `Config: Loaded from local storage` 後面的 IP 地址。

「API 覆寫」解法

若裝置在網路上可達但回報不正確的資料或舊 IP，可透過 REST API 強制更新持久化的 `LocalConfig`，無需重新燒錄：

```bash
curl -s -X POST "http://[CURRENT_IP]/api/config" \
  -H "Content-Type: application/json" \
  -d '{"network":{"ip":"192.168.50.9","gateway":"192.168.50.1","subnet":"255.255
```

只要裝置在*某個* IP 上可達，即可繞過實體 Serial 存取就能清除 Flash。

09

11. 部署與韌體上傳錯誤

DFU 故障排除（`Error 74`）

症狀：`dfu-util: No DFU capable USB device available`。**原因與解法**：

1. **實體連線**：確保 USB-C 線完全插入且支援資料傳輸。
2. **偵測檢查（macOS）**：執行 `ls /dev/cu.usbmodem*`。若出現如 `/dev/cu.usbmodem12201` 的 port，裝置已連線但處於 **Serial 模式**而非 DFU 模式。
3. **觸發 DFU 模式**：
   - **快速雙擊 Reset**：快速按 **Reset** 按鈕兩次。
   - **長按**：按住 **Reset** 按鈕約 2 秒。
   - **視覺確認**：Opta 的 LED（通常 L1/L2）會開始有節奏的「心跳」或閃爍模式。
4. **驅動/Port 衝突**：確保沒有 Serial Monitor 正在使用。若在心跳模式中仍然錯誤，嘗試不同的 USB 埠或線。

10

12. WiFi 無線電韌體救援 (v5.4.13)

在工業環境中，Arduino Opta 的 WiFi 無線電模組（Nina-W10）可能因 QSPI 分區表損壞或應用程式資料（LittleFS）與無線電韌體區域重疊而進入「Mount Failed」狀態。

12.1 「Failed to mount filesystem」錯誤

- **症狀**：Serial monitor 在開機時重複印出 `Failed to mount the filesystem containing the WiFi firmware`。
- **原因**：QSPI Flash 開頭的 1MB 專用分區已被覆寫或 Master Boot Record（MBR）偏移。

12.2 強制救援順序

恢復無線電需要使用 Arduino IDE（v2.2+）的特定順序。**勿跳過第 2 步。**

1. **第一階段：QSPI 分區重設**
   - **工具：**`Examples > STM32H747_System > QSPIFormat`。
   - **動作：**上傳到 **M7 Core**。開啟 Serial Monitor（115200 baud）。
   - **結果：**清除整個 16MB Flash 並恢復出廠 MBR。**注意：此操作會刪除所有使用者設定。**
2. **第二階段：無線電韌體重新佈建（重大）**
   - **工具：**`Examples > WiFi > WiFiFirmwareUpdater`。
   - **動作：**QSPIFormat 完成後立即上傳。依照 Serial Monitor 提示操作。
   - **結果：**將 Nina-W10 無線電檔案系統和 TLS 憑證重新安裝到 1MB 分區。**跳過此步驟會導致持續的 mount 失敗。**
3. **第三階段：專案恢復**
   - **動作：**返回 PlatformIO 重新上傳 **MES Gateway** 專案韌體。
   - **結果：**裝置應能正確開機、找到無線電韌體，並允許啟用 WiFi AP 模式。

12.3 資料復原說明

由於 `QSPIFormat` 會清除配置，若裝置半可運作，請在執行前記下以下設定：

- **裝置名稱**
- **靜態 IP 憑證**
- **MQTT Broker 地址**
- **自訂信號規則**

第三階段完成後，裝置將預設為 DHCP。存取 UI 即可恢復個人化設定。

11

13. 非同步 RS485 掃描 (v5.6)

由於 Modbus 掃描的長時間（最長 2.5 分鐘），同步架構會觸發 HTTP 逾時並阻塞 Gateway 的 Web Server。v5.6 引入了非同步模式。

13.1 非同步模式實作

- **Stage 1 (POST)**: Triggering a scan via `/api/modbus/scan` returns `{"success":true,"scanning":true}` immediately.
- **Stage 2 (Loop)**: The main `loop()` detects the `g_scanPending` flag.
  - **Cache Reset (v5.6.1)**: To ensure results reflect the current bus state, the logic explicitly calls `powerMonitor.clearScanResults()` and `localConfig.deleteScanResults()` before starting.
  - **Discovery**: Performs a round-robin poll of all slave IDs.
- **Stage 3 (GET)**: The frontend polls the endpoint.
  - **Configuration Awareness (v5.6.1)**: The backend iterates through the discovered devices and compares their `slaveId` against the active `DeviceConfig`. It injects an `alreadyAdded` boolean field into the JSON response for each device, enabling the UI to distinguish between new and existing equipment.

13.2 化流排韌性與鎖定

- **手動化流排鎖定**：掃描期間使用 `lockBus(true)` 暫停正常電力監控輪詢。
- **ZOMBIE 鎖定韌性**：為防止瀏覽器重新整理期間化流排被永久鎖定，`WebPage::init()` 例程和後端啟動序列會發出明確的 `lockBus(false)`。
- **逾時最佳化**：`MODBUS_TIMEOUT` 從 500ms 降低到 200ms 以加速掃描，同時維持工業相容性。

13.3 裝置專屬整合（Finder 6M）

- **同位對齊**：Finder 6M 的出廠預設為 **8N1**（SERIAL_8N1）。透過修正 `PowerMonitor` 管理器中的同位設定，穩定了高效能輪詢。
- **半雙工硬體**：Arduino Opta 內建的 RS485 收發器需要正確設定 `halfDuplex` 旗標（`_uart->begin()` 的第 3 個參數，目前為 `true`）以允許共用 TX/RX 接腳正確切換角色。

13.4 Power Monitor 讀取最佳化（Finder 6M 批次限制）

v5.8.5 中發現 RS485 裝置（特別是 Finder 6M）在單次 Modbus 批次讀取超過 18 個暫存器時會回報「Offline」。

- **根本原因**：Finder 6M 有硬體/韌體限制，單次批次讀取超過 18 個暫存器（如 192-213，共 22 個）會觸發 Modbus 逾時或錯誤。
- **解法（兩階段讀取）**：`PowerMonitor` 邏輯重構為分兩次獨立的 Modbus 呼叫：
  1. **批次 1**：Registers 192-209（18 個）涵蓋電壓、電流、功率和頻率。
  2. **批次 2**：Registers 210-213（4 個）涵蓋正向和負向能量計數器。
- **顯示與擷取解耦**：系統使用 `displayMask` 控制 UI 中顯示的量測項目。此機制與實際資料擷取解耦，確保 Data Logger 無論使用者顯示偏好如何，都持續接收所有欄位的更新。

13.5 非同步出站連線模式 (v5.9+)

§13.1 的非同步模式解決了「長時間操作阻塞 HTTP handler」的問題。v5.9 的授權啟用發現了更深層的限制：**入站 HTTP 連線期間無法建立出站 TCP 連線**。

根本原因

STM32H747 的 Mbed OS 網路堆疊（lwIP）資源有限。當設備正在處理入站 HTTP 請求時（EthernetServer socket 已佔用），嘗試建立出站 TCP 連線（`EthernetClient.connect()`）會因為 socket 資源爭用而失敗（回傳 `0`）。

通用非同步出站連線模式

此模式適用於所有需要在 HTTP handler 中觸發出站 TCP 連線的場景：

```
HTTP Handler（入站 socket 佔用中）          loop()（入站 socket 已釋放）
──────────────────────────────────          ──────────────────────────────
1. 收到 POST 請求                           4. 偵測 pending flag
2. 設定 g_pending* flag + 參數              5. 執行出站 TCP 連線
3. 回傳 202 Accepted                        6. 寫入結果到全域變數
   {"status":"activating"}                  7. 清除 pending flag

前端 polling（每 2 秒）
──────────────────────
8. GET 查詢結果 → 回傳最終狀態
```

實際應用：授權啟用

階段	實作位置	說明
排程	`handlePostLicenseActivate()` in `ApiHandlers_Connectivity.cpp`	設定 `g_pendingLicense.pending = true` + 回傳 `202`
執行	`doLicenseActivateAsync()` in `loop()`	建立出站 TCP 到 Config Server
輪詢	`activateLicense()` in `web/index.html`	前端每 2 秒 GET `/api/license/status`，最多 5 次

注意事項

- **不可**在 HTTP handler 內直接呼叫 `EthernetClient.connect()`
- 前端必須處理 `202` 狀態碼並啟動 polling
- 全域 pending 結構應包含超時保護（避免 flag 永久卡住）

12

14. OTA Bundle 機制（規劃中）

為確保韌體和 Web UI（LittleFS 資產）在遠端更新期間保持同步，實作了「Bundle」方法。

14.1 Bundle 架構

Gateway 使用單一 `.mesb` 容器而非獨立更新：

- **Header (32B)**：包含專案魔術字（`MESB`）、韌體大小、Web payload 大小及兩者的 CRC32。
- **韌體二進位檔**：標準二進位檔，用於 Internal Flash。
- **Web Payload**：壓縮的 `index.html`，用於 QSPI Partition 4 LittleFS。

14.2 更新生命週期

1. **下載**：Bundle 下載到 **QSPI Partition 2**（OTA Buffer，5MB）。
2. **驗證**：系統驗證 header 和完整性（CRC32）。
3. **韌體更新**：韌體透過 `mbed::BlockDevice` 寫入 Internal Flash，系統重啟。
4. **UI 部署**：首次開機時，新韌體偵測 Partition 2 中的 Web payload，解壓縮並覆寫 Partition 4 中的 `/web/index.html`。
5. **完成**：OTA buffer 被清除，系統恢復正常運作。

14.3 版本鎖定

此機制防止「版本漂移」（新韌體期望特定的 API 結構但舊的快取 Web UI 不支援），確保工業介面可靠性。

13

15. TCP Converter 與雙向資料 (v5.9)

TCP Converter 將 MQTT/Modbus 通道轉換為進階資料管線，整合到規則引擎中。

15.1 核心邏輯：樣板替換（出站）

系統透過對使用者定義的 JSON 樣板執行純關鍵字替換來滿足複雜的 MES schema。

- **變數標籤**：支援 `${io.*}`、`${vc.*}`、`${rs485.*}` 和 `${sys.*}`。
- **工作流**：使用者將原始 JSON 範例貼入 UI 的 `<textarea>`，將靜態值替換為標籤，儲存到 `config.json`。
- **執行**：邏輯迴圈中的非阻塞非同步組裝。

15.2 入站 JSON 解析器（信號來源）

MQTT 訊息被解析以擷取系統信號：

Mode	Strategy	Use Case
Boolean	String match	cmd == "START"
Numeric	Direct extraction	speed == 1500
String	Raw text	job_id "A123"

- **值策略**:

  1. **最後值保持**:保留狀態直到下一則訊息到達。
  2. **過期逾時**:可選視窗(N 秒);若無更新則將信號標記為 'stale'。

15.3 資源限制

- **最大樣板數:16 個作用中樣板**(由 `#define MAX_CONVERTER_TEMPLATES` 控制)。
- **RAM 安全**:每個作用中樣板分配約 1KB 以防止 heap 破碎化。
- **協議範圍**:主要針對 **MQTT**(非同步)和 **Modbus TCP**(暫存器對暫存器橋接)。原始 TCP Sockets 已廢棄。

14

17. Modbus TCP Interface (Server/Client)

為提升工廠內 HMI、PLC 與 SCADA 的互操作性,Gateway 提供標準的 Modbus TCP 支援。

17.1 Server Role (Slave) - Port 502

Gateway 作為 Server 運行,可支援最多 4 個同時連線。目前採用 **0-based 固定位址對映 (Fixed Address Map)**。經過 v5.9 擴表後,現在擁有充足的暫存器空間能容納主機、擴充模組、虛擬通道、以及十數台的 Modbus 設備:

暫存器映射 (Register Map - 實作現況 v5.9 大容量版):

暫存器類型 (HMI 類型)	地址範圍	對應數據來源	權限	說明
Coil (0x)	`0 ~ 3`	主機 DO[0..3]	R/W	讀寫主機實體繼電器輸出
Coil (0x)	`16 + e*16`	擴充模組 DO	R/W	讀寫擴充模組輸出 (Exp0: 16-23, Exp1: 32-39...)
Discrete Input (1x)	`0 ~ 7`	主機 DI[0..7]	R	讀取主機實體輸入 High/Low
Discrete Input (1x)	`16 + e*16`	擴充模組 DI	R	讀取擴充模組輸入 (Exp0: 16-31, Exp1: 32-47...)
Input Register (3x)	`0 ~ 7`	主機 AI[0..7]	R	讀取類比輸入原始值 (經倍率轉換 0~65520)
Holding Register (4x)	`0 ~ 1023`	內部系統 Holding 區	R/W	提供高達 1024 個通用的 Holding 區 (`MODBUS_HOLDING_REG_COUNT=1024`)，可供外界讀寫。可作為 Rule Engine 與 TCP Converter 輸出目標的共通變數區。

推薦 Holding Register (4x) 應用區間規劃： 為避免設定混亂，建議 SCADA/HMI 規劃位址時遵循以下使用分佈：

- `0 ~ 99`: 系統內部保留與通用控制 VAR 變數。
- `100 ~ 199`: 虛擬通道 (Virtual Channels: Counter / Timer) 推送值。
- `200 ~ 499`: 實體 RS485 (Modbus RTU) 設備快取區 (支援 10+ 台設備，每台保留 20-30 Registers)。
- `500 ~ 999`: 其他 TCP IO 通道 (Parser) 所擷取提取的機台實時生產數據暫地。

(註：目前 Holding Register 僅作為大區塊供外界存取與系統對接，開發者可將 Parser 擷取到的值以 rule 行為自由倒入上述任一位址中。)

17.2 Client Role (Master)

Gateway 作為 Master 可以主動向外讀取數據（**目前 TCP 部分尚未實作**）。

- 若須收集其他 Modbus TCP 設備數據至系統，請透過第三方軟體(如 Node-RED)橋接為 MQTT Parser 輸入。
- 實體序列線路的 **Modbus RTU Master** 則已完整實作於自定義的 `ModbusManager` 中，支援 RS485 輪詢。