Capítulo 31 · Proyecto integrador II: despliegue Edge, Docker y SLA

Conceptos clave

Mide latencia, disponibilidad y recuperación de acuerdo con el SLA. Conserva la configuración respaldada y prueba la restauración.



GUÍA PASO A PASO DE LABORATORIO — MÓDULO 10

Proyecto Integrador Final (*Capstone*) — Especificación de Ingeniería, Interconexión Maestra y Despliegue de la Estación Inteligente

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

Módulo: 10 — Integración Full-Stack de las 4 Capas (Hardware Determinista, Bus RS-485, Servidor Edge Cifrado y Supervisión Analítica)

---

1. Objetivos de Ingeniería del Proyecto Integrador (Parte 1)

- Ensamblar el hardware completo de la **Estación Mecatrónica Inteligente de Control Térmico, Ambiental y Energético** (en doble protoboard estilo Tinkercad o sobre la Placa Portadora Madre soldada `MECATROBOTICA-CAPSTONE-CARRIER-V10`).
- Compilar y cargar en el controlador de campo Arduino el firmware integral de tiempo real estricto (`firmware_maestro_estacion_capstone.ino`) que unifica muestreo a $20\ \text{Hz}$ con filtro `EMA`, control `PID discreto` ($10\ \text{Hz}$) sobre ventilador `PWM`, histéresis sobre calefactor, máquina de 5 estados finitos (`FSM`), rearme físico, verificación `CRC-16` y *Watchdog* a $2\ \text{s}$.
- Interconectar el nodo Arduino con la pasarela **Raspberry Pi 4B** mediante el bus diferencial industrial **`RS-485`** (`MAX485` con terminación de $120\ \Omega$) y desplegar la estructura maestra `/opt/mecatrobotica/capstone/`.
- Levantar la pila de contenedores de producción (`docker_compose_capstone_produccion.yml`) con cifrado **`MQTTS TLS 1.3 (:8883)`**, listas `ACL`, persistencia en unidad `SSD` (`InfluxDB 2.7` + cola `SQLite WAL`) y paneles **ANSI/ISA-101** en `Grafana`.

---

2. Fase 1: Ensamblaje de la Estación Mecatrónica en Protoboard o Placa Portadora PCB

- **Subsistema de Percepción (Sensores de Campo en Arduino):**
- Conecta la sonda térmica `NTC 10k` / `DS18B20` a `A0`, el sensor de iluminación `LDR` a `A1`, y el bus `I2C` (`A4/SDA`, `A5/SCL`) al sensor ambiental **`BME280`** **(`0x76`)** y al monitor de potencia **`INA219`** **(`0x40`)**.
- **Subsistema de Actuación Dual y Seguridad Local:**
- Conecta el pin `D9 (PWM Timer1)` a la compuerta del **`MOSFET IRLZ44N`** (con resistencias de $100\ \Omega$ y $10\ \text{k}\Omega$ y diodo *flyback* `1N4007`) para modular el ventilador axial de $12\ \text{V DC}$.
- Conecta el pin `D7` a la entrada del **módulo de relé optoacoplado** para el elemento calefactor, el pin `D8` al zumbador de alarma local y el pin `D2` (`INPUT_PULLUP`) al botón físico de rearme.
- **Subsistema de Transporte Industrial `RS-485`:**
- Conecta `TX/RX` y `D4 (DE/RE)` del Arduino al transceptor **`MAX485`** y tiende el par trenzado diferencial (`A` y `B` con resistencias terminadoras de $120\ \Omega$) hacia el adaptador industrial `USB-RS485` de la Raspberry Pi.

📐 Plano 1: Montaje Integral en Protoboard de la Estación Inteligente Capstone



---

3. Fase 2: Despliegue de la Estructura `/opt/mecatrobotica/capstone/` en Raspberry Pi

- Ejecuta el script de aprovisionamiento automatizado en la Raspberry Pi:

  ```bash
  chmod +x despliegue_automatizado_capstone.sh
  sudo ./despliegue_automatizado_capstone.sh
  ```

- Verifica que los cuatro contenedores de producción estén saludables y escuchando únicamente en los puertos autorizados (`8883` y `3000` hacia LAN, `1883` , `8086` y `1880` confinados a `127.0.0.1`):

  ```bash
  docker compose -f /opt/mecatrobotica/capstone/docker_compose_capstone_produccion.yml ps
  ```

- Verifica la operación en tiempo real del puente demonizado `orquestador_edge_capstone_bridge.py` :

  ```bash
  journalctl -u capstone-bridge.service -f -n 30
  ```

LISTADO DE MATERIALES (BOM) Y PRESUPUESTO GLOBAL — MÓDULO 10

Estación Mecatrónica Inteligente Completa (Proyecto Integrador Final *Capstone*)

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

Módulo: 10 — Proyecto Integrador Final (*Capstone*): Arquitectura y Despliegue

---

1. Tabla Maestra de Materiales de la Estación Integradora (BOM Consolidado)

Subsistema	Designador	Componente / Especificación Técnica	Cant.	Rol en la Estación Capstone
Control de Campo (Capa 1)	`MCU1`	**Arduino UNO R3 o Arduino Nano (ATmega328P, 16 MHz)**	1	Ejecución determinista (20 Hz / 10 Hz) de sensores, lazo PID, histéresis, FSM y *Watchdog* de 2 s.
Sensórica Multimodal	`S1..S4`	**Sonda Térmica** `DS18B20` **/** `NTC 10k` **+** `BME280` **(** `0x76` **) +** `INA219` **(** `0x40` **) +** `LDR`	4	Medición de temperatura de proceso, humedad/presión, corriente/potencia de actuador ($0.1\ \Omega$) e iluminación.
Etapa de Potencia Dual	`Q1, D1, K1`	**MOSFET Nivel Lógico** `IRLZ44N` **+ Disipador + Diodo** `1N4007` **+ Módulo Relé Optoacoplado + Ventilador** 12 V **+ Calefactor**	1 kit	Enfriamiento continuo por `PWM` (490 Hz) y calentamiento por banda de histéresis con aislamiento galvánico.
Seguridad Física Local	`SW1, BZ1`	**Pulsador Industrial de Rearme (** `D2` **) + Zumbador Piezoeléctrico (** `D8` **) + Indicadores LED**	1 kit	Alarma acústica/visual inmediata y rearme manual verificado tras estado seguro `FAULT_SAFE_SHUT`.
Transporte Industrial (Capa 2)	`U_485`	**2× Transceptores** `MAX485` **+ Par Trenzado Blindado + 2× Resistencias** $120\ \Omega$ **+ Adaptador USB-RS485 FTDI**	1 kit	Enlace diferencial inmune a ruido electromagnético con verificación polinómica `CRC-16` (`0xA001`).
Servidor Edge (Capa 3)	`SBC1, SSD1`	**Raspberry Pi 4B (4 GB/8 GB) + SSD USB 3.0 (256 GB) + RTC** `DS3231` **+ Cámara CSI + Gabinete con Tamper**	1 kit	Alojamiento de `Mosquitto TLS 8883`, `InfluxDB 2.7`, `Node-RED`, `Grafana`, `SQLite WAL` e `Isolation Forest`.
Placa Portadora Soldada	`PCB_CAP`	**Placa Madre Industrial Doble Capa FR-4 (** `MECATROBOTICA-CAPSTONE-CARRIER-V10` **)** con zócalos y borneras `KF301`	1	Integración permanente libre de cables sueltos con partición de planos de masa `AGND`, `DGND` y `PGND`.

---

2. Presupuesto de Tiempos de Ejecución y Determinismo por Capa

Tarea / Bucle del Sistema	Dispositivo	Periodo Nominal (T)	Tiempo de Cómputo (C)	Factor de Utilización ($U = C/T$)
Adquisición ADC + Filtro EMA ($\alpha=0.20$) + Chequeo de Sonda	Arduino (`MCU1`)	50 ms (20 Hz)	0.42 ms	0.84%
Ley de Control PID Discreto + Histéresis + Máquina FSM	Arduino (`MCU1`)	100 ms (10 Hz)	0.35 ms	0.35%
Empaquetado JSON + Cálculo `CRC-16` + Envío `RS-485`	Arduino (`MCU1`)	1000 ms (1 Hz)	8.40 ms	0.84%
Inferencia `Z-Score` + `Isolation Forest` + Ingesta `SQLite/InfluxDB`	Raspberry Pi 4B	1000 ms (1 Hz)	3.10 ms	0.31%

MANUAL TÉCNICO DE INGENIERÍA — MÓDULO 10

Arquitectura Maestra de Cuatro Capas, Partición de Masas en Placa Portadora y Orquestación del Proyecto Integrador (*Capstone*)

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

---

1. Integración de las Cuatro Capas de Ingeniería Mecatrónica IoT

El Proyecto Integrador Final (*Capstone*) consolida en una sola estación ciberfísica todos los subsistemas desarrollados a lo largo del curso:

- **Capa 1 — Percepción, Actuación y Control Determinista (Arduino UNO / Nano):**
- Muestreo analógico/digital a $20\ \text{Hz}$ ($T_s = 50\ \text{ms}$) con filtro exponencial $\bar{x}_k = \alpha\, x_k + (1-\alpha)\bar{x}_{k-1}$.
- Regulador **PID discreto** a $10\ \text{Hz}$ ($T_s = 100\ \text{ms}$) con *Anti-Windup* por recorte y derivada sobre la medición para el ventilador `PWM`, coordinado con control por banda de histéresis para el calefactor a fin de evitar que ambos actuadores compitan térmicamente entre sí.
- Máquina de estados finitos (`FSM`) con transición en $< 100\ \text{ms}$ a `FAULT_SAFE_SHUT` ante rotura de sensor o sobrecorriente ($> 850\ \text{mA}$) y temporizador *Watchdog* de hardware activo a $2\ \text{s}$.
- **Capa 2 — Transporte Industrial Diferencial (`RS-485` + `CRC-16`):**
- Comunicación sobre par trenzado balanceado (A - B) con control preciso de dirección `DE/RE` tras `Serial.flush()` y verificación polinómica `CRC-16` (`0xA001`).
- **Capa 3 — Cómputo en el Borde, Ciberseguridad y Persistencia (Raspberry Pi 4B):**
- Demonios `systemd` en Python protegidos por cerrojo `threading.Lock` , broker `Mosquitto` con cifrado `TLS 1.3 (:8883)` y reglas `ACL` , ingesta resiliente *Store-and-Forward* con `SQLite WAL` hacia `InfluxDB 2.7` y mantenimiento predictivo con `Isolation Forest` .
- **Capa 4 — Supervisión Analítica y Mando Remoto (`Grafana ISA-101` + `Node-RED` + `WireGuard`):**
- Panel de alto desempeño con estado de la máquina `FSM` , curvas históricas, llamadas `RPC` correlacionadas con `req_id` y alertas instantáneas.

📐 **Plano 3: Arquitectura Maestra de 4 Capas de la Estación Inteligente Capstone**



---

2. Coordinación Térmica Dual: PID de Enfriamiento + Histéresis de Calefacción con Zona Muerta

Cuando una cámara térmica cuenta tanto con un calefactor resistivo (controlado por relé electromecánico) como con un ventilador axial de extracción (controlado por `PWM` mediante `MOSFET`), es fundamental establecer una **banda muerta de exclusión mutua** alrededor del *Setpoint* $r[k]$ para evitar oscilaciones y derroche energético:

- **Región de Enfriamiento Proporcional-Integral-Derivativo ($T_{medida} > r[k]$):** El relé del calefactor permanece estrictamente apagado (`HEAT = 0`) y el controlador PID modula el ciclo útil `PWM` del ventilador entre 0 y 255.
- **Zona Muerta de Equilibrio Térmico ($r[k] - 0.3\,^\circ C \leq T_{medida} \leq r[k]$):** Ambos actuadores permanecen en reposo o con el integrador estabilizado sin conmutar el relé.
- **Región de Calefacción con Histéresis ($T_{medida} < r[k] - 1.5\,^\circ C$):** Se activa el relé del calefactor (`HEAT = 1`) con el ventilador apagado (`PWM = 0`) hasta alcanzar $r[k] - 0.3\,^\circ C$.

📐 Plano 4: Máquina de Estados Finitos Integrada y Coordinación de Lazos en el Capstone

PLANO 4: MÁQUINA DE ESTADOS FINITOS (FSM) DETERMINISTA Y FLUJO DE DATOS BIDIRECCIONAL

Transiciones con Histéresis Térmica, Apagado Seguro en < 100 ms (FAULT_SAFE_SHUT) y Ciclo de Solicitud-Respuesta RPC

1. MÁQUINA DE ESTADOS FINITOS (FSM) EN EL MICROCONTROLADOR

ESTADO 0: BOOT_SELF_TEST (Autodiagnóstico de Arranque)

- Inicializa ADC, bus I2C (BME280 + INA219) y transceptor RS-485.
- Verifica que la lectura inicial del NTC esté dentro de [-20 °C, +110 °C].
- **Si los sensores son válidos -> Transiciona a ESTADO 1 (NORMAL_AUTO).**

ESTADO 1: NORMAL_AUTO (Regulación Térmica PID + Histéresis)

- Ventilador gobernado por ley PID discreta $u[k] \in [0, 255]$ cada 50 ms.
- Calefactor gobernado por histéresis alrededor del Setpoint $r[k]$.
- **Si T_actual >= T_warn + ΔT_hyst (32.8 °C) -> Pasa a ESTADO 2.**

ESTADO 2: WARNING_HIGH_TEMP (Alerta por Sobretemperatura)

- Enclavamiento de seguridad: Calefactor forzado a OFF (heat=0).
- PID aplica enfriamiento activo reforzado + Buzzer intermitente.
- **Solo regresa a ESTADO 1 cuando T_actual <= T_warn - ΔT_hyst (31.2 °C).**

ESTADO 3: FAULT_SAFE_SHUT (Apagado Seguro por Fallo Crítico < 100 ms)

- Se activa si ADC <= 5 o ADC >= 1018 (rotura o cortocircuito de NTC).
- Acción inmediata: Calefactor = OFF, Ventilador = 100 % (PWM 255), err = 1.
- **Requiere subsanar el sensor y recibir confirmación física (D4) o RPC ACK.**

2. TUBERÍA DE DATOS Y PROTOCOLO RPC CORRELACIONADO

FLUJO ASCENDENTE DE TELEMETRÍA (CAMPO -> BORDE -> BASE DE DATOS)

1. Muestreo y Filtrado Local (t = 0 ms):

$$y_{ema}[k] = \alpha \cdot x_{adc}[k] + (1 - \alpha) \cdot y_{ema}[k-1] \quad \text{con } \alpha = 0.25$$

2. Empaquetado y Firma CRC-16 Modbus (cada 1000 ms):

Arduino calcula CRC-16 (polinomio 0xA001) y transmite por RS-485.

3. Validación y Publicación MQTTS en Raspberry Pi (t = +8 ms):

serial_mqtt_bridge.py verifica CRC-16 y publica en puerto 8883 (QoS 1):
Tópico: planta/zona1/arduino01/telemetry

4. Persistencia Dual y Detección de Anomalías (t = +18 ms):

- store_and_forward_ingestor.py -> InfluxDB (o SQLite WAL si hay caída).
- edge_anomaly_detector.py -> Evalúa [Temp, PWM, mA] en tiempo real.

FLUJO DESCENDENTE DE MANDO BIDIRECCIONAL (PATRÓN RPC CON ACK)

1. Operador cambia consigna a 28.5 °C en el Dashboard:

Node-RED genera un UUID corto req_id = "c4a9" y publica en:
Tópico: planta/zona1/arduino01/rpc/request
Payload: `{"req_id":"c4a9","cmd":"SET_SP","value":28.5}`

2. Autorización Criptográfica en Broker Mosquitto:

Mosquitto verifica que el usuario nodered_service posea permiso 'write' en acl.conf sobre planta/+/+/rpc/request.

3. Ejecución y Confirmación desde el Arduino (< 1500 ms):

El puente envía la orden con CRC-16 por RS-485; el Arduino valida que 28.5 °C esté dentro del rango seguro [15.0, 40.0], actualiza r[k] y devuelve:
Tópico: planta/zona1/arduino01/rpc/response
Payload: `{"req_id":"c4a9","status":"OK","applied_sp":28.5}`

---

3. Árbol de Despliegue en Producción (`/opt/mecatrobotica/capstone/`) y Orquestación

📐 Plano 5: Estructura de Producción, Contenedores Docker y Servicios Systemd del Capstone

PLANO 5: ESTRUCTURA DE DIRECTORIOS DE PRODUCCIÓN Y DESPLIEGUE AUTOMATIZADO (LECCIÓN 10.2)

Árbol Estándar /opt/mecatrobotica/capstone, Orquestación Docker Compose + Unidades systemd y Aprovisionamiento en 6 Fases

1. ÁRBOL DE DIRECTORIOS EN LA RASPBERRY PI

```
/opt/mecatrobotica/capstone/
├── certs/                          # Permisos 0750 (root:mosquitto)
│   ├── ca.crt                      # Certificado Raíz CA X.509 (0644)
│   ├── server.crt                  # Certificado Broker con SAN (0644)
│   └── server.key                  # Clave Privada RSA-2048 (0600)
│
├── docker-compose.yml              # Pila declarativa ARM64 / x86_64
├── .env                            # Secretos y Tokens (chmod 0600)
│
├── firmware_arduino/               # Proyecto PlatformIO / Arduino CLI
│   ├── platformio.ini              # Configuración ATmega328P
│   └── src/
│       ├── main.cpp                # Lazo determinista 20 Hz + FSM
│       └── DiscretePid.h           # PID con Anti-Windup y Derivada -y[k]
│
├── mosquitto/
│   └── config/
│       ├── mosquitto.conf          # Listener TLS 8883 + Cero anónimos
│       ├── passwd                  # Usuarios con hash SHA-512 + sal
│       └── acl.conf                # Matriz de mínimo privilegio
│
└── services/                       # Demonios systemd (Python .venv)
    ├── serial_mqtt_bridge.py       # Puente RS-485 <-> MQTTS 8883
    ├── store_and_forward_ingestor.py  # Ingestor SQLite WAL -> InfluxDB
    └── edge_anomaly_detector.py    # Detector INA219 + IsolationForest
```

2. PIPELINE DE DESPLIEGUE AUTOMATIZADO (despliegue_automatizado_capstone.sh)

FASE 1: Preparación de Sistema Operativo, Grupos POSIX y Entorno Virtual Python

- Verifica grupo 'dialout', instala dependencias ARM64 y crea /opt/mecatrobotica/capstone/.venv.

FASE 2: Aprovisionamiento Criptográfico PKI X.509 con Extensión SAN (OpenSSL)

- Genera ca.crt, server.crt y server.key incluyendo DNS local e IP estática en subjectAltName.

FASE 3: Endurecimiento de Red (UFW + Fail2ban) y Supervivencia Flash (log2ram + WDT)

- Aplica reglas de cortafuegos LAN, monta /var/log en RAM y activa RuntimeWatchdogSec=15.

FASE 4: Despliegue de Contenedores Docker Compose y Salud de InfluxDB v2

- Ejecuta 'docker compose up -d' y espera HTTP 200 en http://127.0.0.1:8086/health.

FASE 5: Compilación y Carga del Firmware Maestro en el Nodo Arduino

- Compila main.cpp + DiscretePid.h y verifica recepción de la primera trama con CRC-16 válido.

FASE 6: Registro y Arranque de Unidades systemd con Auto-Reinicio

- Habilita serial-mqtt-bridge, store-and-forward-ingestor y edge-anomaly-detector.

PLANOS Y ESQUEMÁTICOS DE CIRCUITO E INGENIERÍA — MÓDULO 10

Planos en Protoboard (Estilo Tinkercad), Placa Portadora Madre PCB FR-4 con Montaje por Soldadura y Arquitectura Capstone

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

Módulo: 10 — Proyecto Integrador Final (*Capstone*) — Estación Inteligente de Monitoreo y Control

---

1. Plano 1 (Estilo Tinkercad / Fritzing): Montaje Integral en Protoboard de la Estación Mecatrónica Completa

En este primer plano maestro se detalla el conexionado completo sobre protoboard de todos los subsistemas de la Estación Inteligente:

- **Sensores analógicos y digitales (`NTC/DS18B20`, `LDR`, `BME280` e `INA219`):** Alimentados y desacoplados con condensadores cerámicos de $100\ nF$.
- **Etapa de potencia dual:** Transistor `MOSFET IRLZ44N` en `D9` con diodo *flyback* `1N4007` para el ventilador `PWM` de $12\ V$ y módulo de relé optoacoplado en `D7` para el calefactor, alimentados desde la fuente externa de $12\ V/3\ A$ con retorno en estrella.
- **Enlace diferencial `RS-485` (`MAX485`):** Conectado a `TX/RX/D4` del Arduino y enlazado por par trenzado con resistencias terminadoras de $120\ \Omega$ hacia la **Raspberry Pi 4B** con unidad `SSD USB 3.0`.

📐 Plano 1: Montaje Integral en Protoboard de la Estación Capstone



---

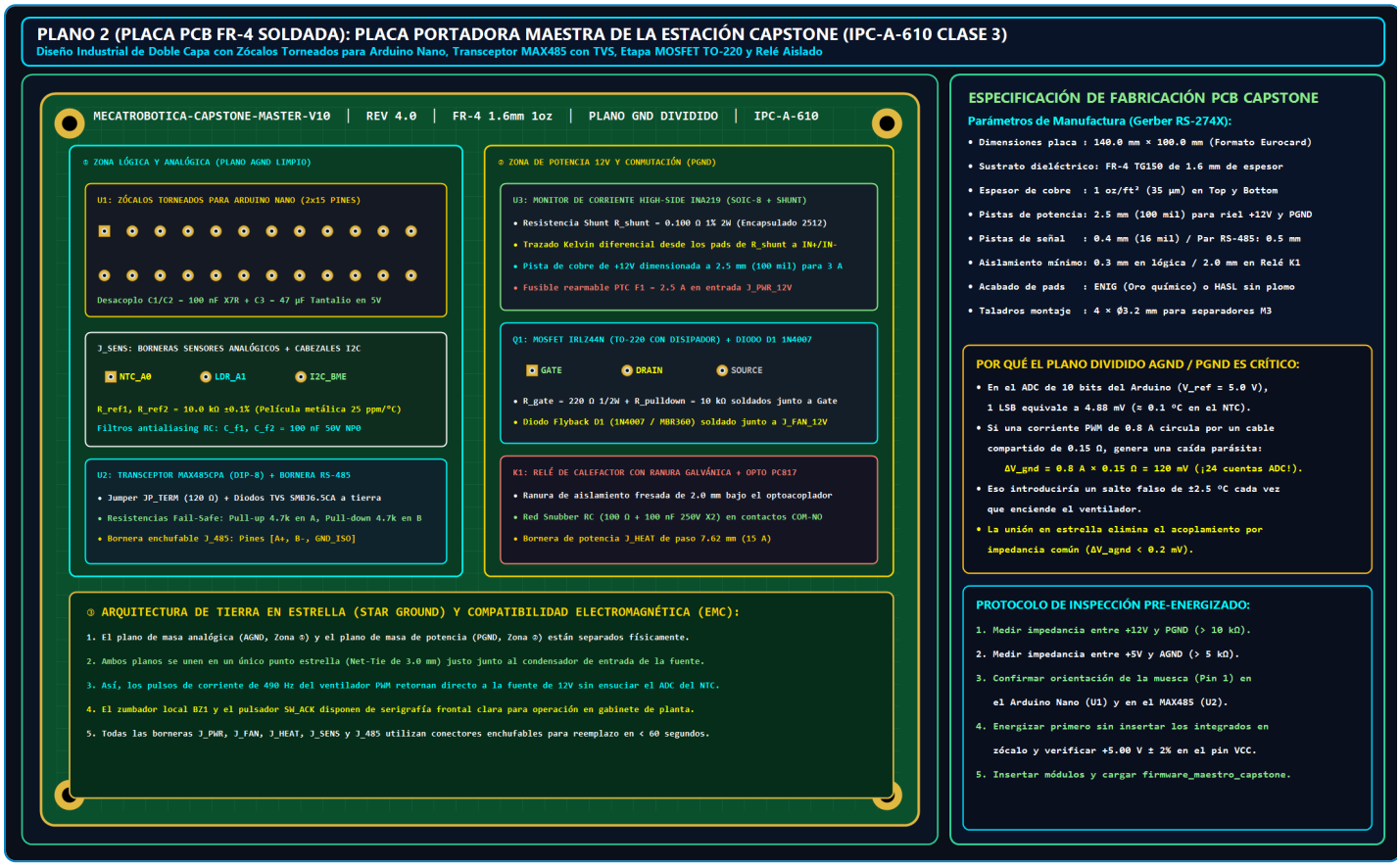
2. Plano 2 (Placa de Circuito Impreso PCB FR-4): Placa Portadora Madre con Montaje por

Soldadura (`MECATROBOTICA-CAPSTONE-CARRIER-V10`)

Para consolidar el proyecto final en un prototipo de grado industrial libre de la fragilidad de decenas de cables puente Dupont, este segundo plano presenta el diseño de la **Placa Portadora Madre en fibra de vidrio FR-4 de doble capa (`MECATROBOTICA-CAPSTONE-CARRIER-V10`)** con montaje por soldadura:

- **Zócalos hembra torneados soldados para el microcontrolador y módulos `I2C` :** Permiten insertar y reemplazar el controlador sin desoldar la placa base.
- **Partición física en tres zonas sobre el circuito impreso con unión en estrella (`Star Ground`):**
- **Zona Analógica y Sensórica (`AGND`):** Alejada de las trazas de conmutación para preservar la relación señal-ruido del ADC de 10 bits.
- **Zona Digital y Diferencial `RS-485` (`DGND`):** Con par diferencial simétrico para las líneas `A` y `B` del `MAX485` y diodos `TVS` hacia bornera `TB_RS485` .
- **Zona de Conmutación de Potencia (`PGND`):** Con pistas de cobre de $2.5\ mm$ de ancho para el `MOSFET IRLZ44N` , medición Kelvin en el shunt de $0.100\ \Omega$ del `INA219` y ranura fresada bajo el optoacoplador del relé.

📐 Plano 2: Placa Portadora Madre PCB FR-4 Soldada de la Estación Capstone



---

3. Plano 3: Arquitectura Maestra de Cuatro Capas de la Estación Inteligente Capstone

📐 Plano 3: Arquitectura Maestra de 4 Capas del Capstone



---

4. Plano 4: Máquina de 5 Estados Finitos (`FSM`), Coordinación Térmica y Flujo Bidireccional

📐 Plano 4: Máquina de Estados Finitos y Coordinación de Lazos



5. Plano 5: Estructura de Producción `/opt/mecatrobotica/capstone/`, Contenedores Docker y Servicios `systemd`

📐 Plano 5: Despliegue en Producción de Contenedores y Servicios Systemd

PLANO 5: ESTRUCTURA DE DIRECTORIOS DE PRODUCCIÓN Y DESPLIEGUE AUTOMATIZADO (LECCIÓN 10.2)
Árbol Estándar /opt/mecatrobotica/capstone, Orquestación Docker Compose + Unidades systemd y Aprovisionamiento en 6 Fases

1. ÁRBOL DE DIRECTORIOS EN LA RASPBERRY PI

```
/opt/mecatrobotica/capstone/
├── certs/                          # Permisos 0750 (root:mosquitto)
│   ├── ca.crt                      # Certificado Raíz CA X.509 (0644)
│   ├── server.crt                  # Certificado Broker con SAN (0644)
│   └── server.key                  # Clave Privada RSA-2048 (0600)
│
├── docker-compose.yml              # Pila declarativa ARM64 / x86_64
├── .env                            # Secretos y Tokens (chmod 0600)
│
├── firmware_arduino/               # Proyecto PlatformIO / Arduino CLI
│   ├── platformio.ini              # Configuración ATmega32BP
│   └── src/
│       ├── main.cpp                # Lazo determinista 20 Hz + FSM
│       └── DiscretePid.h           # PID con Anti-Windup y Derivada -y[k]
│
├── mosquitto/
│   └── config/
│       ├── mosquitto.conf          # Listener TLS 8883 + Cero anónimos
│       ├── passwd                  # Usuarios con hash SHA-512 + sal
│       └── acl.conf                # Matriz de mínimo privilegio
│
└── services/                       # Demonios systemd (Python .venv)
    ├── serial_mqtt_bridge.py       # Puente RS-485 <-> MQTTS 8883
    ├── store_and_forward_ingestor.py  # Ingestor SQLite WAL -> InfluxDB
    └── edge_anomaly_detector.py    # Detector INA219 + IsolationForest
```

2. PIPELINE DE DESPLIEGUE AUTOMATIZADO (despliegue_automatizado_capstone.sh)

FASE 1: Preparación de Sistema Operativo, Grupos POSIX y Entorno Virtual Python

- Verifica grupo 'dialout', instala dependencias ARM64 y crea /opt/mecatrobotica/capstone/.venv.

FASE 2: Aprovisionamiento Criptográfico PKI X.509 con Extensión SAN (OpenSSL)

- Genera ca.crt, server.crt y server.key incluyendo DNS local e IP estática en subjectAltName.

FASE 3: Endurecimiento de Red (UFW + Fail2ban) y Supervivencia Flash (log2ram + WDT)

- Aplica reglas de cortafuegos LAN, monta /var/log en RAM y activa RuntimeWatchdogSec=15.

FASE 4: Despliegue de Contenedores Docker Compose y Salud de InfluxDB v2

- Ejecuta 'docker compose up -d' y espera HTTP 200 en http://127.0.0.1:8086/health.

FASE 5: Compilación y Carga del Firmware Maestro en el Nodo Arduino

- Compila main.cpp + DiscretePid.h y verifica recepción de la primera trama con CRC-16 válido.

FASE 6: Registro y Arranque de Unidades systemd con Auto-Reinicio

- Habilita serial-mqtt-bridge, store-and-forward-ingestor y edge-anomaly-detector.