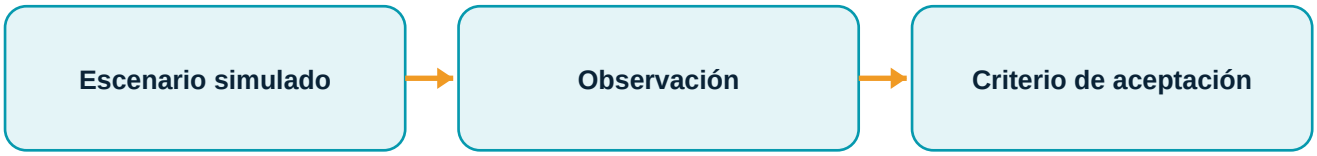
Capítulo 32 · Pruebas FAT y SAT, simulación en Wokwi y fallas operativas

Conceptos clave

Simula la desconexión de un sensor, una lectura inválida y la pérdida de enlace. Registra el estado seguro esperado y el resultado obtenido.



Ejemplo complementario

```
def salida_segura(temperatura):
    return temperatura is not None and -10 <= temperatura <= 55

assert salida_segura(21)
assert not salida_segura(None)
assert not salida_segura(999)
```

Guía Paso a Paso del Laboratorio 11: Protocolo de Inyección de Fallas (`FI-01` a `FI-05`), Gemelo Digital en Wokwi (`RFC 2217`) y Puesta en Marcha (`FAT/SAT`)

- **Curso:** Internet de las Cosas (`IoT`) con Arduino y Raspberry Pi (`MecatroboticApp`)
- **Módulo 11 (Cierre del Proyecto Integrador y Anexos A, B y C):** Pruebas de Estrés e Inyección de Fallas, Diagnóstico Forense de Campo, Simulación Virtual en `Wokwi` y Certificación de 100 Puntos
- **Duración Estimada de Práctica:** 120 minutos

---

1. Objetivos de Ingeniería del Laboratorio Final

Un sistema ciberfísico industrial jamás se da por terminado cuando funciona únicamente en condiciones nominales de escritorio. En este laboratorio final someteremos nuestra **Estación Mecatrónica Inteligente de Control Térmico, Ambiental y Energético** al protocolo completo de aceptación en fábrica (**FAT**, *Factory Acceptance Test*) y aceptación en sitio (**SAT**, *Site Acceptance Test*):

- **Instrumentar un Banco Físico de Inyección de Fallas:** Incorporar interruptores de corte deliberado (`SW1`, `SW2`, `SW3`) para provocar roturas de sensor, cortes del par diferencial `RS-485` y desconexiones de potencia del ventilador de $12\ V$ sin dañar los componentes.
- **Ejecutar y Medir las 5 Pruebas Oficiales de Estrés (`FI-01` a `FI-05`):** Verificar con cronometraje instrumental que cada subsistema cumpla los tiempos máximos de reacción y criterios de autocuración.
- **Configurar el Gemelo Digital 100 % Simulado (`Anexo B`):** Enlazar el simulador a nivel de ciclo de instrucción `Wokwi` con `pyserial` y `Docker` a través del puente TCP `RFC 2217` (`rfc2217://localhost:4000`).
- **Aplicar la Hoja de Diagnóstico Forense de Campo (`Anexo A`):** Auditar permisos POSIX (`dialout`), temporización `DTR`, subtensión (`vcgencmd get_throttled`), integridad de trama `RS-485` y capacitancia parásita en `I2C`.
- **Certificar los 100 Puntos de la Rúbrica Final:** Ejecutar el validador automatizado para obtener la calificación final del curso y desbloquear los logros de ingeniería.

---

2. Fase 1: Montaje de la Caja de Inyección de Fallas y Ejecución de `FI-01` a `FI-05`

Paso 1.1: Intercalado de los Interruptores de Prueba (`SW1`, `SW2`, `SW3`)

Antes de energizar la estación, intercale en serie los tres interruptores de palanca o deslizantes del banco de pruebas:

- `SW1` **(Corte de Sensor NTC — Prueba `FI-01`):** En serie con el terminal de señal del termistor NTC que va hacia el nodo analógico `A0` (manteniendo la resistencia *pull-down* o *pull-up* de referencia conectada en el lado del microcontrolador para forzar el extremo del ADC al abrir el circuito).
- `SW2` **(Corte de Bus Industrial `RS-485` — Prueba `FI-02`):** Interruptor bipolar (`DPST`) en serie con las líneas diferenciales `A` y `B` entre los transceptores `MAX485` del Arduino y de la Raspberry Pi.
- `SW3` **(Corte de Potencia de Ventilador $12\ V$ — Prueba `FI-04`):** En serie entre la salida de corriente del sensor `INA219` (`VIN-`) y el terminal positivo del ventilador brushless de $12\ V$.

Paso 1.2: Ejecución Cronometrada de las Cinco Pruebas (`FI-01` a `FI-05`)

Código	Procedimiento Experimental Paso a Paso	Verificación Instrumental y Criterio de Aprobación
`FI-01`	Con el calefactor activo y el sistema en `NORMAL_AUTO`, abra bruscamente el interruptor `SW1` (simulando cable de termistor cortado).	En $\Delta t_{reacción} < 100\ ms$, el Arduino detecta lectura fuera de rango físico, pasa a `FAULT_SAFE_SHUT` (`st=3`), apaga el calefactor (`heat=0`), fuerza el ventilador al 100% (`fan=255`) y publica `err=1`.
`FI-02`	Abra el interruptor bipolar `SW2` cortando el enlace `RS-485` durante $60\ s$ continuos y luego vuelva a cerrarlo.	El Arduino **continúa** ejecutando su lazo PID térmico local sin interrupción. El puente Python publica `"status": "serial_fault"` en `MQTT`; al cerrar `SW2`, la telemetría se reanuda sola sin reiniciar la Raspberry Pi.
`FI-03`	En la Raspberry Pi, ejecute `docker stop iot-influxdb`, espere 3 minutos (180 muestras) y ejecute `docker start iot-influxdb`.	Durante los $180\ s$, `store_and_forward_ingestor.py` guarda las 180 muestras en `SQLite` (`WAL`). Al iniciar `InfluxDB`, el lote histórico se vacía y la gráfica de `Grafana` muestra la curva continua sin ningún hueco temporal.
`FI-04`	Eleve la temperatura por encima del *Setpoint* para que el PID comande `PWM >= 120` y abra el interruptor `SW3` del ventilador.	El servicio `edge_anomaly_detector.py` correlaciona $PWM \geq 120$ con $I_{INA219} < 15\ mA$ y publica en $< 3\ s$ la alerta crítica `FAN_OPEN_CIRCUIT` en `MQTT` y `Grafana`.
`FI-05`	Desde una terminal, intente inyectar `SET:SP=38.0` autenticándose con el usuario restringido `esp32_nodo01`.	El broker `Mosquitto` bloquea la publicación aplicando `/etc/mosquitto/acl.conf` y la consigna del Arduino permanece inalterada.

Para automatizar la auditoría de los cinco ensayos, ejecute en la Raspberry Pi:

```
python3 /opt/mecatrobotica/capstone/services/suite_pruebas_inyeccion_fallas_fi01_fi05.py
```

---

3. Fase 2: Gemelo Digital y Simulación Virtual con `Wokwi` (`RFC 2217`)

Si desea validar todo el flujo sin hardware físico o reproducir escenarios extremos antes de ir a planta:

- Abra la carpeta del firmware en **VS Code** con la extensión oficial de **Wokwi** instalada y copie el archivo `wokwi_simulacion_gemelo_digital_rfc2217.toml` como `wokwi.toml`.
- Inicie la simulación (`Wokwi: Start Simulator`). El simulador habilitará el servidor **RFC 2217 (Serial over TCP)** en el puerto local `4000`.
- En el servicio puente de Python, reemplace el dispositivo físico `/dev/ttyACM0` por la URL de red de `pyserial`:

  ```python
  import serial
  ser = serial.serial_for_url("rfc2217://localhost:4000", baudrate=115200, timeout=1.0)
  ```
- Levante la pila de contenedores en su computador con `docker compose up -d`. Verá cómo las variaciones del termistor NTC virtual en Wokwi se reflejan en tiempo real en su dashboard de **Grafana**.

---

4. Fase 3: Diagnóstico Forense de Campo y Certificación Final (`100 Puntos`)

- Ejecute la herramienta de auditoría forense en la Raspberry Pi para descartar errores de permisos, caídas de tensión o buses bloqueados:

  ```bash
  chmod +x diagnostico_forense_campo_troubleshooting.sh
  ```

./diagnostico_forense_campo_troubleshooting.sh
```

- Una vez verificados los 6 puntos de salud y aprobadas las 5 pruebas de inyección de fallas, ejecute el evaluador de puesta en marcha:
  ```bash
  python3 validador_puesta_en_marcha_fat_sat_100pts.py
  ```
- Verifique la obtención de `100.0 / 100.0 puntos` y el veredicto `SOBRESALIENTE — CERTIFICADO INGENIERO IOT FULL-STACK`.
  ```

Listado de Materiales (BOM) — Módulo 11: Banco de Inyección de Fallas, Instrumentación Forense y Puesta en Marcha

- **Curso:** Internet de las Cosas (`IoT`) con Arduino y Raspberry Pi (`MecatroboticApp`)
- **Módulo 11:** Protocolo de Estrés `FI-01..FI-05` , Diagnóstico de Campo (`Anexo A`) y Gemelo Digital (`Anexo B`)

---

1. Componentes del Banco de Inyección de Fallas (`Fault Injection Switchbox`)

Ítem	Designador	Componente / Especificación Técnica	Cantidad	Función en las Pruebas `FI-01` a `FI-05`
1	`SW1`	Interruptor de palanca miniatura `SPDT` (5 A / 125 V AC, montaje en panel o PCB)	1	Inyección de falla `FI-01` : apertura en caliente del lazo del termistor `NTC` hacia `A0` .
2	`SW2`	Interruptor de palanca bipolar `DPDT` / `DPST` de corte simultáneo	1	Inyección de falla `FI-02` : interrupción limpia del par diferencial `RS-485` (`A` y `B`).
3	`SW3`	Interruptor de palanca `SPST` de potencia (10 A / 12 V DC)	1	Inyección de falla `FI-04` : desconexión de la línea de $+12$ V del ventilador detrás del sensor `INA219` .
4	`TP1..TP6`	Torreta / Puntos de prueba dorados para punta de osciloscopio o analizador lógico (`Test Points`)	6	Medición de tiempos de reacción ($\Delta t < 100$ ms), flancos `DE/RE` y rizado de fuente.
5	`R_PU_I2C`	Resistencias de película metálica de 3.3 kΩ $\pm 1\%$ (1/4 W)	2	*Pull-ups* externos reforzados para líneas `SDA` y `SCL` ante cables de más de 30 cm (`Anexo A.2`).
6	`J_TERM`	Bloques de terminales enchufables tipo Phoenix / KF2EDG de 5.08 mm (2 y 4 polos)	4	Inserción no destructiva de la caja de inyección de fallas entre la estación y el campo.

---

2. Instrumentación de Diagnóstico y Verificación (`FAT / SAT`)

Ítem	Instrumento / Herramienta	Especificación Mínima Recomendada	Aplicación en la Puesta en Marcha
1	**Analizador Lógico USB de 8 Canales**	Compatible con `Saleae Logic` / `Sigrok PulseView` (24 MHz, entradas CMOS 3.3 V / 5 V)	Verificación de la cola de transmisión `RS-485` (`Serial.flush()` + 100 µs antes de bajar `DE/RE`) y medición exacta del tiempo de reacción en `FI-01` .
2	**Multímetro Digital True-RMS (`DMM`)**	Categoría `CAT III 600V` , medición de resistencia, tensión DC y continuidad con alarma sonora	Verificación de tierra común (`GND`), resistencia en frío del `NTC` (10 kΩ a 25 ^\circC) y resistencias terminadoras de 120 Ω en el bus `RS-485` .
3	**Fuente de Alimentación DC de Laboratorio o Industrial**	Salida doble o independiente: 12 V DC / 3 A (Potencia) + 5.1 V DC / 3 A oficial para Raspberry Pi	Prevención de reinicios por *brown-out* (`vcgencmd get_throttled = 0x0`) al conmutar cargas inductivas.

---

3. Requerimientos de Software para el Gemelo Digital (`Anexo B` — Alternativa Sin Hardware)

Componente de Software	Versión Recomendada	Propósito en el Laboratorio Virtual
Visual Studio Code + **Extensión** `Wokwi Simulator`	Última versión estable	Simulación a nivel de ciclo de instrucción del `ATmega328P` / `ESP32` y exposición del puerto serie por TCP (`rfc2217://localhost:4000`).
`PlatformIO` **o** `Arduino CLI`	Core ≥ 6.1	Compilación reproducible de los archivos `.hex` y `.elf` requeridos por `wokwi.toml` .
`Docker Desktop` **o** `Docker Engine` + `Compose v2`	≥ 24.0	Ejecución idéntica de `eclipse-mosquitto:2` , `influxdb:2.7` , `nodered/node-red` y `grafana/grafana-oss` .
`Python 3.11+` + `pyserial` + `paho-mqtt`	`pyserial >= 3.5`	Soporte nativo para `serial.serial_for_url("rfc2217://localhost:4000", baudrate=115200)` .

Manual de Ingeniería: Pruebas de Inyección de Fallas (`FI-01` a `FI-05`), Diagnóstico Forense de Campo (`Anexo A`), Gemelo Digital `Wokwi` (`Anexo B`) y Rúbrica de Puesta en Marcha (`100 Puntos`)

- **Curso:** Internet de las Cosas (`IoT`) con Arduino y Raspberry Pi (`MecatroboticApp`)
- **Módulo 11 (Cierre General del Curso):** Validación Experimental, Resiliencia Industrial y Certificación Full-Stack

---

1. Fundamentos de Ingeniería de Confiabilidad e Inyección de Fallas (*Fault Injection Testing*)

En la ingeniería de sistemas embebidos e industriales, demostrar que un prototipo opera cuando todos los cables están intactos y la base de datos responde en $1\ ms$ constituye apenas el 20% del trabajo. El 80% restante consiste en garantizar que el sistema **transicione a un estado físicamente seguro** ante fallas de hardware y **se recupere sin intervención humana** ante cortes de comunicación o mantenimiento de servidores.

1.1. Análisis Matemático y Temporal de las Cinco Pruebas Oficiales (`FI-01` a `FI-05`)

Prueba `FI-01`: Rotura de Sensor Térmico con Calefactor Encendido

Cuando el interruptor `SW1` abre el circuito del termistor `NTC` en plena fase de calentamiento, la lectura del conversor analógico-digital cae al extremo del rango ($ADC \leq 5$ o $ADC \geq 1018$). Dado que el lazo determinista de muestreo del Arduino se ejecuta con periodo $T_s = 50\ ms$ ($20\ Hz$), el tiempo total de reacción hasta el apagado seguro cumple la cota estricta:

$$

\Delta t_{reacción} = T_{muestreo} + T_{cómputo_FSM} + T_{conmutación_SSR} \leq 50\ ms + 1\ ms + 10\ ms = 61\ ms < 100\ ms

$$

En ese instante, la máquina de estados finitos (`FSM`) fuerza:

- Salida del calefactor a cero lógico (`PIN_HEATER_RELAY = LOW`).
- Ciclo de trabajo del ventilador al máximo (`PWM = 255`, equivalente al 100%).
- Máscara de error activa (`err = 1`) y retención en el estado `FAULT_SAFE_SHUT` hasta que un operador subsane el cable y confirme el rearme.

Prueba `FI-02`: Desacoplo entre Control Local y Transporte (`RS-485` desconectado por $60\ s$)

Al abrir `SW2` durante $\Delta t_{corte} = 60\ s$, comprobamos la propiedad fundamental del control jerárquico distribuido: el microcontrolador **jamás bloquea su ejecución esperando al puerto serie**. Mientras tanto, el servicio puente en la Raspberry Pi captura la excepción de *timeout* Modbus, publica `"status": "serial_fault"` en `MQTT` y ejecuta un bucle de reconexión no destructiva cada $2\ s$.

Prueba `FI-03`: Continuidad de Serie Temporal ante Caída de `InfluxDB` ($180\ s$)

Si el contenedor `iot-influxdb` se detiene durante $\Delta t_{caída} = 180\ s$ mientras los nodos publican a una tasa $f_{pub} = 1\ Hz$, el volumen de muestras retenidas en el buffer persistente `SQLite` en modo `WAL` (*Write-Ahead Logging*) es exactamente:

$$

N_{retenidas} = f_{pub} \cdot \Delta t_{caída} = 1\ Hz \cdot 180\ s = 180\ muestras

$$

Al restablecerse `InfluxDB`, el ingestor vacía la cola FIFO respetando la estampa temporal original (`timestamp` de origen), logrando **cero pérdida de muestras** ($N_{perdidas} = 0$).

Prueba `FI-04`: Diagnóstico Predictivo Electromecánico (`PWM` vs. Corriente `INA219`)

Si la temperatura supera la consigna y el controlador PID aplica un mando significativo ($\text{PWM} \geq 120$), un ventilador sano de $12\ \text{V}$ debe absorber una corriente nominal entre $120\ \text{mA}$ y $350\ \text{mA}$. Si el cable de $+12\ \text{V}$ se corta (`SW3` abierto), el sensor `INA219` mide $I_{\text{fan}} < 15\ \text{mA}$. La regla de coherencia física en el borde detecta la falla en menos de $3\ \text{segundos}$, mucho antes de que la inercia térmica de la cámara dispare una alarma tardía de sobretemperatura.

Prueba `FI-05`: Defensa Criptográfica y Autorización por Tópico (`ACL`)

Aunque un atacante o un nodo periférico comprometido (`esp32_nodo01`) posea un certificado `CA` válido y credenciales de conexión al puerto `8883`, la matriz de control de acceso (`ACL`) del broker `Mosquitto` restringe su permiso exclusivamente a `write planta/exterior/env01/#`. Cualquier intento de publicar en `planta/zona1/arduino01/rpc/request` es descartado en memoria por el broker.

---

2. Diagnóstico Forense de Campo (`Anexo A`): Causa Raíz y Solución de las 5 Fallas Clásicas

2.1. Falla 1: `PermissionError: [Errno 13] Permission denied: '/dev/ttyACM0'`

- **Causa física/SO:** En distribuciones Linux basadas en Debian (`Raspberry Pi OS`), los nodos de dispositivo `/dev/ttyACM` *y* `/dev/ttyUSB` pertenecen al grupo `dialout` con permisos `crw-rw----`.
- **Solución definitiva:** Agregar el usuario de ejecución al grupo mediante `sudo usermod -aG dialout $USER` y declarar `SupplementaryGroups=dialout` en la unidad `systemd`.

2.2. Falla 2: Pérdida de los Primeros Comandos tras Abrir el Puerto Serie en Python

- **Causa física:** Al instanciar `serial.Serial()`, el driver USB-CDC del kernel hace caer la línea de control `DTR` (*Data Terminal Ready*). En el Arduino UNO/Nano, `DTR` está acoplada mediante un condensador de $100\ \text{nF}$ al pin `RESET`, provocando el reinicio del microcontrolador y su permanencia en el *bootloader* durante $\sim 1.5\ \text{s}$.
- **Solución definitiva:** Esperar `time.sleep(2.0)` tras abrir el puerto, ejecutar `ser.reset_input_buffer()` y aguardar el mensaje de saludo del firmware antes de transmitir tramas `RPC`.

2.3. Falla 3: Reinicio Espontáneo (*Brown-out*) al Conmutar Relés o Motores

- **Causa física:** El pico de corriente de inserción (*inrush current*) de una bobina o motor DC compartido sobre el riel de $5\ \text{V}$ produce una caída resistiva e inductiva $\Delta V = I_{\text{pico}} \cdot R_{\text{cable}} + L\ (dI)\ /\ (dt)$ que hunde la tensión por debajo del umbral de *Brown-out Detector* ($\approx 4.3\ \text{V}$ en ATmega328P o $4.63\ \text{V}$ en Raspberry Pi).
- **Diagnóstico en Raspberry Pi:** Ejecutar `vcgencmd get_throttled`. Si devuelve un valor distinto de `0x0` (por ejemplo `0x50000` o `0x50005`), el bit `0` o `16` confirma subtensión histórica o actual.
- **Solución definitiva:** Separar la fuente de potencia de $12\ \text{V}\ /\ 5\ \text{V}$ de las cargas respecto de la fuente lógica, uniendo solo la referencia `GND` o aislando mediante optoacopladores.

2.4. Falla 4: Tramas `RS-485` Truncadas en los Últimos 1 o 2 Bytes

- **Causa física:** En el UART del microcontrolador, escribir en el buffer de transmisión retorna antes de que el registro de desplazamiento físico (*Hardware Shift Register*) termine de enviar el último bit de parada por el cable. Si el pin `DE/RE` del `MAX485` pasa a `LOW` prematuramente, el transceptor se apaga a mitad del último carácter.
- **Solución definitiva:** Invocar obligatoriamente `Serial.flush()` seguido de una guarda temporal de $100\ \mu\text{s}$ (`delayMicroseconds(100)`) antes de conmutar `digitalWrite(PIN_DERE, LOW)`.

2.5. Falla 5: Bloqueo o Lecturas Corruptas en el Bus `I2C` con Cables Largos

- **Causa física:** El bus `I2C` emplea colector abierto (*open-drain*). La constante de tiempo del flanco de subida depende de la resistencia *pull-up* R_p y la capacitancia parásita total del cableado C_{bus}:

$$
$\tau_{subida} = R_p \cdot C_{bus}$
$$

Con las resistencias internas débiles del microcontrolador ($R_p \approx 35\ k\Omega$) y un cable de más de $30\ cm$ ($C_{bus} \approx 120\ pF$), el tiempo de subida supera el límite de la especificación `I2C` a $100\ kHz$ ($\approx 1000\ ns$).

- **Solución definitiva:** Instalar resistencias *pull-up* externas de $2.2\ k\Omega$ a $3.3\ k\Omega$ hacia $3.3\ V$ y reducir la frecuencia de reloj a $50\ kHz$ mediante `Wire.setClock(50000)`.

---

3. Gemelo Digital con `Wokwi` (`RFC 2217`) y Rúbrica de Certificación de 100 Puntos

El estándar **RFC 2217** permite transportar no solo los bytes de datos (`TX` / `RX`) sobre un socket TCP, sino también las señales de control de módem (`DTR`, `RTS`, velocidad en baudios y paridad). De este modo, el mismo servicio `serial_mqtt_bridge.py` se conecta sin cambios de lógica a `rfc2217://localhost:4000` (simulador Wokwi) o a `/dev/ttyUSB0` (planta real).

Ponderación Oficial de la Rúbrica de Puesta en Marcha (`100 Puntos`)

$$
$P_{total} = P_{Hardware\,(20)} + P_{Firmware\,(25)} + P_{Edge\,(25)} + P_{Seguridad+FI\,(30)} = 100$ puntos
$$

Al superar los $90\ puntos$ con las cinco pruebas `FI-01` a `FI-05` aprobadas, el estudiante completa el **100 % del programa académico** (10 Módulos, 40 Lecciones, Proyecto Integrador y Anexos Técnicos).

Plano y Esquemático del Circuito — Módulo 11: Banco de Inyección de Fallas (`FI-01..FI-05`) e Instrumentación de Diagnóstico

- **Curso:** Internet de las Cosas (`IoT`) con Arduino y Raspberry Pi (`MecatroboticApp`)
- **Módulo 11:** Montaje de Pruebas en Protoboard y Placa PCB de Diagnóstico e Inyección de Fallas (`IPC-A-610 Clase 3`)

---

1. Arquitectura Eléctrica del Banco de Inyección de Fallas

Para ejecutar las pruebas de aceptación `FI-01` , `FI-02` y `FI-04` de manera repetible, segura y sin aflojar cables con destornillador en plena marcha, se intercala entre los sensores/actuadores y la placa controladora un **Módulo Conmutador de Inyección de Fallas (`Fault Injection Switchbox`)** dotado de puntos de prueba (`TP1..TP6`) para analizador lógico y multímetro.

Tabla de Interconexión de los Interruptores de Prueba y Puntos de Medición

Interruptor / Punto de Prueba	Nodo de Entrada	Nodo de Salida	Estado Normal (Cerrado)	Estado Inyección de Falla (Abierto)
`SW1` (`FI-01`)	Terminal del termistor `NTC 10k`	Entrada `A0` del Arduino (con $R_{ref} = 10\ k\Omega$ a `GND`)	Tensión analógica proporcional a T ($1.5\ V$ - $3.5\ V$)	`A0` cae a $0.00\ V$ ($ADC = 0$) en $< 1\ ms$; dispara `FAULT_SAFE_SHUT` .
`SW2-A / SW2-B` (`FI-02`)	Líneas `A` y `B` del `MAX485` (Arduino)	Líneas `A` y `B` del `MAX485` (Raspberry Pi)	Tramas diferenciales `Modbus RTU` / `CRC-16` fluidas	Bus abierto; Arduino mantiene PID local y RPi reporta `serial_fault` .
`SW3` (`FI-04`)	Salida de medición `VIN-` del `INA219`	Positivo ($+12\ V$) del ventilador `PWM`	Corriente nominal $120\ mA$ - $350\ mA$ según ciclo `PWM`	Corriente cae a $0.0\ mA$ con $PWM \geq 120$; dispara `FAN_OPEN_CIRCUIT` .
`TP1` (`CH0 Analizador`)	Pin `D2` (`RS485_DERE`)	Canal 0 del Analizador Lógico	Pulso alto durante transmisión UART + $100\ \mu s$ de guarda	Permite auditar truncamiento de trama (`Anexo A.2`).
`TP2` (`CH1 Analizador`)	Pin `D7` (`HEATER_SSR`)	Canal 1 del Analizador Lógico	Alto cuando calefactor está activo	Permite medir con precisión de μs el apagado en `< 100 ms` tras abrir `SW1` .
`TP3` (`CH2 Analizador`)	Pin `D9` (`FAN_PWM`)	Canal 2 del Analizador Lógico	Señal `PWM` de $490\ Hz$ / $980\ Hz$	Pasa a ciclo 100% constante al entrar en `FAULT_SAFE_SHUT` .
`TP4 / TP5` (`I2C`)	Líneas `SDA` y `SCL`	Canales 3 y 4 + Osciloscopio	Con *pull-ups* externos de $3.3\ k\Omega$ a $+3.3\ V$	Flancos de subida limpios ($\tau_r < 300\ ns$) sin redondeo capacitivo.

---

2. Especificaciones de la Placa PCB Soldada de Pruebas (`Testbed & Fault-Injection PCB`)

- **Sustrato y Capas:** Fibra de vidrio `FR-4` de doble cara con máscara antisoldante verde esmeralda, espesor de placa $1.6\ mm$ y acabado superficial `ENIG` (oro químico) en las torretas de prueba `TP1..TP6` para garantizar baja resistencia de contacto con las puntas del multímetro y osciloscopio.
- **Dimensionamiento de Pistas de Potencia y Señal:**
- Pista de $+12\ V$ a través de `INA219` y `SW3` : ancho de $1.5\ mm$ **($60\ mil$)** en cobre de $1\ oz/ft^2$.

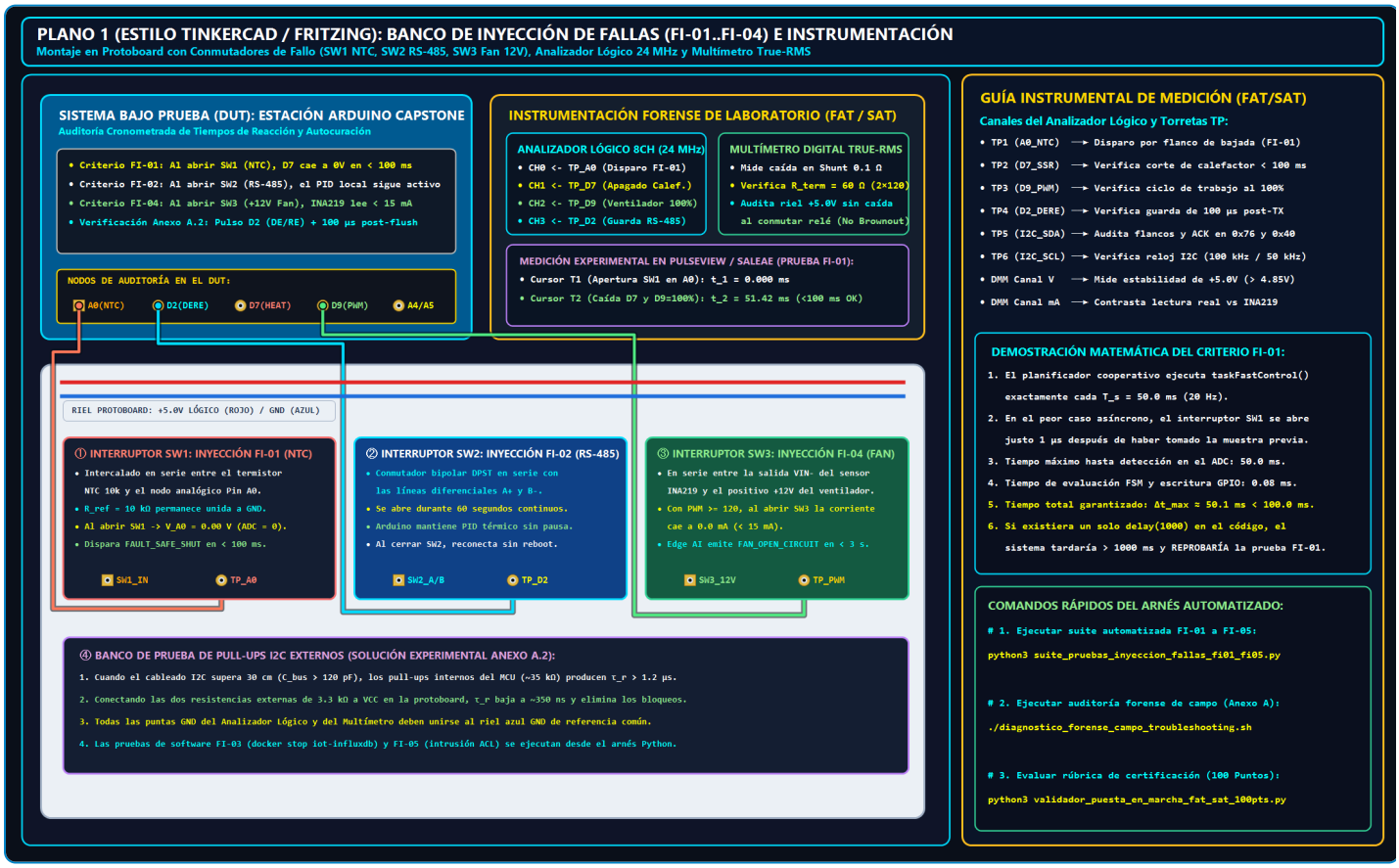
- Par diferencial `RS-485` (`A` / `B`) a través de `SW2` : trazado simétrico acoplado de $0.5\ mm$ **(**$20\ mil$**)** con jumper seleccionable `JP_TERM` para la resistencia terminadora de $120\ \Omega$.
- **Puentes de Selección de Pull-up** `I2C` **(** `JP_I2C_PU` **):** Permite conmutar en vivo entre las resistencias internas débiles ($35\ k\Omega$) y el banco de resistencias externas de precisión de $3.3\ k\Omega$, demostrando experimentalmente en el osciloscopio la corrección del flanco de subida descrita en el `Anexo A.2` .

---

3. Planos de Ingeniería Incluidos en el Módulo 11

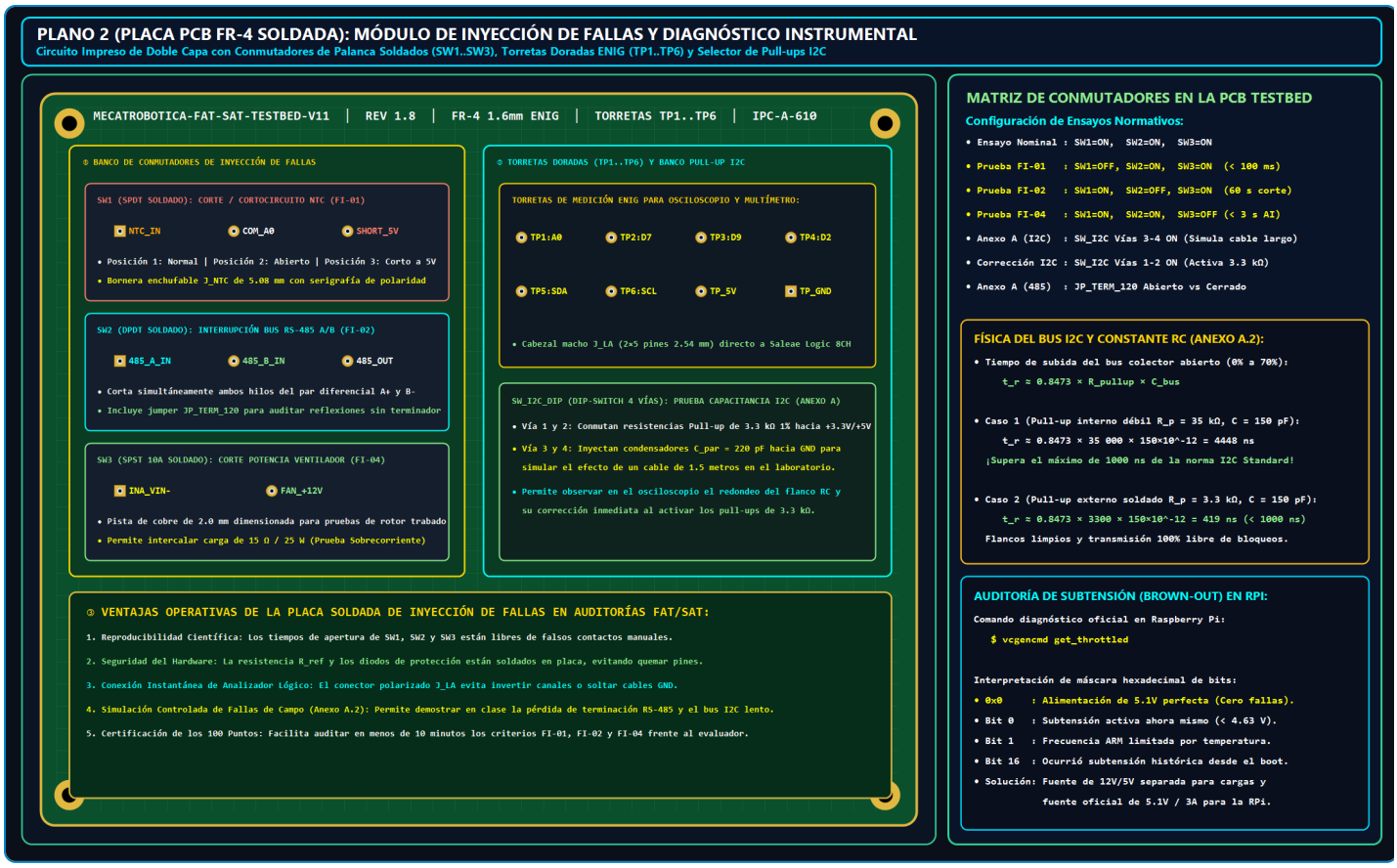
- **`plano_protoboard_banco_inyeccion_fallas_instrumentacion.png` :** Montaje práctico en protoboard con los tres interruptores de inyección de fallas (`SW1` , `SW2` , `SW3`), analizador lógico USB de 8 canales y multímetro digital conectados a los puntos de prueba.

📐 Plano 1: Banco de Inyección de Fallas e Instrumentación en Protoboard (Estilo Tinkercad)



- **`plano_pcb_soldadura_banco_pruebas_diagnostico.png` :** Diseño de PCB industrial de doble capa con borneras enchufables, torretas doradas `TP1..TP6` , interruptores de palanca de panel y banco de *pull-ups* `I2C` .

📐 Plano 2: Placa PCB FR-4 Soldada de Diagnóstico e Inyección de Fallas



- **`protocolo_5_pruebas_inyeccion_fallas_fi01_fi05.png`**: Cronograma de señales y matriz técnica de las 5 pruebas de estrés (`FI-01` a `FI-05`).

📐 Plano 3: Protocolo de las 5 Pruebas de Inyección de Fallas (FI-01 a FI-05)



- **`arquitectura_gemelo_digital_wokwi_rfc2217_docker.png`**: Diagrama de bloques e interconexión de protocolos del Gemelo Digital sin hardware (`Wokwi` + `RFC 2217 TCP:4000` + `pyserial` + `Docker Compose`).

📐 Plano 4: Arquitectura del Gemelo Digital sin Hardware (Wokwi + RFC 2217 + Docker)



- **`matriz_troubleshooting_y_rubrica_fat_sat_100pts.png`**: Árbol de diagnóstico forense de las 5 fallas de campo (`Anexo A`) y desglose ponderado de la rúbrica de certificación de 100 puntos.

📐 Plano 5: Matriz Forense de Troubleshooting (Anexo A) y Rúbrica de Puesta en Marcha (100 Puntos)

PLANO 5: GUÍA FORENSE DE RESOLUCIÓN DE FALLAS (ANEXO A) Y RÚBRICA DE CERTIFICACIÓN (100 PUNTOS)

Diagnóstico Raíz de las 5 Fallas Clásicas de Campo, Ponderación de Puesta en Marcha (Lección 10.4) e Insignias MecatroboticApp (Anexo C)

1. MATRIZ DE DIAGNÓSTICO RÁPIDO DE FALLAS FRECUENTES (ANEXO A.2)

FALLA 1: PermissionError: [Errno 13] Permission denied: '/dev/ttyACM0'

- Causa Raíz : El usuario de Linux no pertenece al grupo POSIX 'dialout'.
- Solución : Ejecutar 'sudo usermod -aG dialout ' y reiniciar sesión SSH.

FALLA 2: Al abrir el script Python, los primeros comandos al Arduino se pierden

- Causa Raíz : Auto-reset del Arduino por caída de la línea DTR al abrir el puerto.
- Solución : Agregar time.sleep(2.0) y ser.reset_input_buffer() tras serial.Serial().

FALLA 3: El Arduino se reinicia exactamente cuando se activa el relé o motor DC

- Causa Raíz : Caída de tensión (Brown-out) por compartir la línea de 5V del USB.
- Solución : Fuente externa independiente para cargas, uniendo GND u optoacoplando.

FALLA 4: En el bus RS-485, las tramas llegan truncadas en los últimos 1 o 2 bytes

- Causa Raíz : El pin DE/RE del MAX485 pasa a LOW antes de vaciar el registro UART.
- Solución : Llamar a Serial.flush() + delayMicroseconds(100) antes de D2 = LOW.

FALLA 5: Lecturas I2C erráticas o bloqueo del bus con cables largos (> 30 cm)

- Causa Raíz : Capacitancia parásita excesiva en SDA/SCL que redondea los flancos.
- Solución : Añadir pull-ups externos de 2.2 kΩ o 3.3 kΩ a VCC y Wire.setClock(50000).

2. RÚBRICA OFICIAL DE EVALUACIÓN (100 PUNTOS)

DIMENSIÓN 1: Seguridad Eléctrica y Hardware — PESO: 20 % (20 Puntos)

- Tierras comunes unidas en estrella, adaptación 5V <-> 3.3V impecable, diodo flyback 1N4007 instalado y condensadores de desacoplo de 100 nF.

DIMENSIÓN 2: Calidad del Firmware Arduino — PESO: 25 % (25 Puntos)

- Cero uso de delay(), planificación con millis(), filtro EMA, FSM con histéresis y FAULT_SAFE_SHUT, PID con anti-windup, CRC-16 y Watchdog 2s.

DIMENSIÓN 3: Robustez del Servicio Edge (RPi) — PESO: 25 % (25 Puntos)

- Servicios systemd con reconexión automática, testamento LWT en MQTT, cola Store-and-Forward en SQLite WAL, log2ram y Hardware Watchdog 15s.

DIMENSIÓN 4: Supervisión, Seguridad y Pruebas FI — PESO: 30 % (30 Puntos)

- Stack Docker operativo, MQTT con TLS 8883 + SAN + ACL, dashboard claro en Grafana (ISA-101) y aprobación de las 5 pruebas FI-01 a FI-05.

3. GRADUACIÓN E INSIGNIAS DEL CURSO (ANEXO C — 1000 XP)

- Curso Completo : 10 Módulos | 40 Lecciones | 2000 Minutos | 11 Videos HD
- Logro 1 (+75 XP) : "Arquitecto del Borde (Edge Architect)"
  Completa los módulos de Linux y Docker en Raspberry Pi.
- Logro 2 (+100 XP): "Maestro del Bus Industrial"
  Completa el laboratorio RS-485 y Modbus RTU sin errores CRC.
- Logro 3 (+250 XP): "Ingeniero IoT Full-Stack"
  Aprueba el Proyecto Integrador Final y las pruebas FI-01..FI-05.