Capítulo 13 · Servidor Edge Linux con udev, GPIO y systemd

Conceptos clave

Identifica el equipo por atributos estables. Ejecuta el servicio con los permisos indispensables y comprueba su recuperación tras reiniciar.



Guía Paso a Paso del Laboratorio 03 — Módulo 03 (Video 03)

Despliegue Headless, Reglas `udev`, Control GPIO Moderno y Servicio Resiliente `systemd`

> [!NOTE]

> **Archivo Complementario Referenciado en el Video 03:** Esta guía contiene el procedimiento completo paso a paso para reproducir en laboratorio o en casa la configuración íntegra del servidor Edge Linux explicada en el **Video 03**. Todos los scripts listos para usar se encuentran en la carpeta [`codigo_fuente/`] (file:///D:/_Mecatrobotica/Video_03_Modulo_03_Servidor_Edge_Linux_Raspberry_Pi/material_complementario/codigo_fuente/).

Paso 1: Grabación *Headless* de Raspberry Pi OS Lite (64-bit) y Llaves SSH `Ed25519`

- En tu computador de ingeniería (Windows, Linux o macOS), abre una terminal y genera tu par de llaves criptográficas `Ed25519` si aún no dispones de una:

  ```bash
  ssh-keygen -t ed25519 -C "ingenieria-iot@mecatrobotica"
  ```

- Abre **Raspberry Pi Imager**, selecciona tu modelo de placa (**Raspberry Pi 4** o **Raspberry Pi 5**), elige **Raspberry Pi OS (other) ➔ Raspberry Pi OS Lite (64-bit)** (basado en Debian 12 *Bookworm*) y selecciona tu tarjeta microSD Clase A2.
- Haz clic en **Edit Settings (Personalización del SO)**:
- **Hostname:** `edge-gateway-01` (accesible en red local mediante `mDNS` como `edge-gateway-01.local`).
- **Username:** `iotadmin` y una contraseña robusta.
- **Wireless LAN / Ethernet:** Configura el SSID si usarás Wi-Fi o conecta cable Ethernet Cat 6.
- **Services (Pestaña Servicios):** Activa **Enable SSH**, selecciona **Allow public-key authentication only** y pega el contenido de tu archivo `~/.ssh/id_ed25519.pub`.
- Inserta la microSD en la Raspberry Pi, energízala con la fuente oficial de $5.1\ \text{V}$ y conéctate remotamente tras $40\ \text{segundos}$:

  ```bash
  ssh iotadmin@edge-gateway-01.local
  ```

Paso 2: Habilitación de Buses de Hardware (`I2C`, `SPI`, `UART`, `1-Wire`)

Una vez dentro de la sesión SSH en la Raspberry Pi, habilita los periféricos del kernel y libera el puerto serie de la consola de login ejecutando:

```
# 1. Habilitar bus I2C (/dev/i2c-1) y bus SPI (/dev/spidev0.0)
sudo raspi-config nonint do_i2c 0
sudo raspi-config nonint do_spi 0

# 2. Desactivar consola de login en UART (1 = desactivar consola) y activar hardware UART (0 = activar)
sudo raspi-config nonint do_serial_cons 1
sudo raspi-config nonint do_serial_hw 0

# 3. Habilitar bus 1-Wire en GPIO 4 (Pin físico 7)
echo "dtoverlay=w1-gpio,gpiopin=4" | sudo tee -a /boot/firmware/config.txt
```

Paso 3: Grupos de Privilegio Mínimo y Reglas `udev` Deterministas

- Otorga acceso directo a los dispositivos de hardware (`/dev/ttyACM`, `/dev/i2c-1`, `/dev/spidev`, `/dev/gpiochip*`) al usuario `iotadmin` sin necesidad de ejecutar tus programas como `root`:

  ```bash
  sudo usermod -aG dialout,i2c,spi,gpio iotadmin
  ```

- Copia el archivo complementario [`codigo_fuente/99-iot-arduinos.rules`] (file:///D:/_Mecatrobotica/Video_03_Modulo_03_Servidor_Edge_Linux_Raspberry_Pi/material_complementario/codigo_fuente/99-iot-arduinos.rules) a `/etc/udev/rules.d/` y recarga el gestor `udev`:

  ```bash
  sudo cp 99-iot-arduinos.rules /etc/udev/rules.d/99-iot-arduinos.rules
  ```

sudo udevadm control --reload-rules

sudo udevadm trigger

```

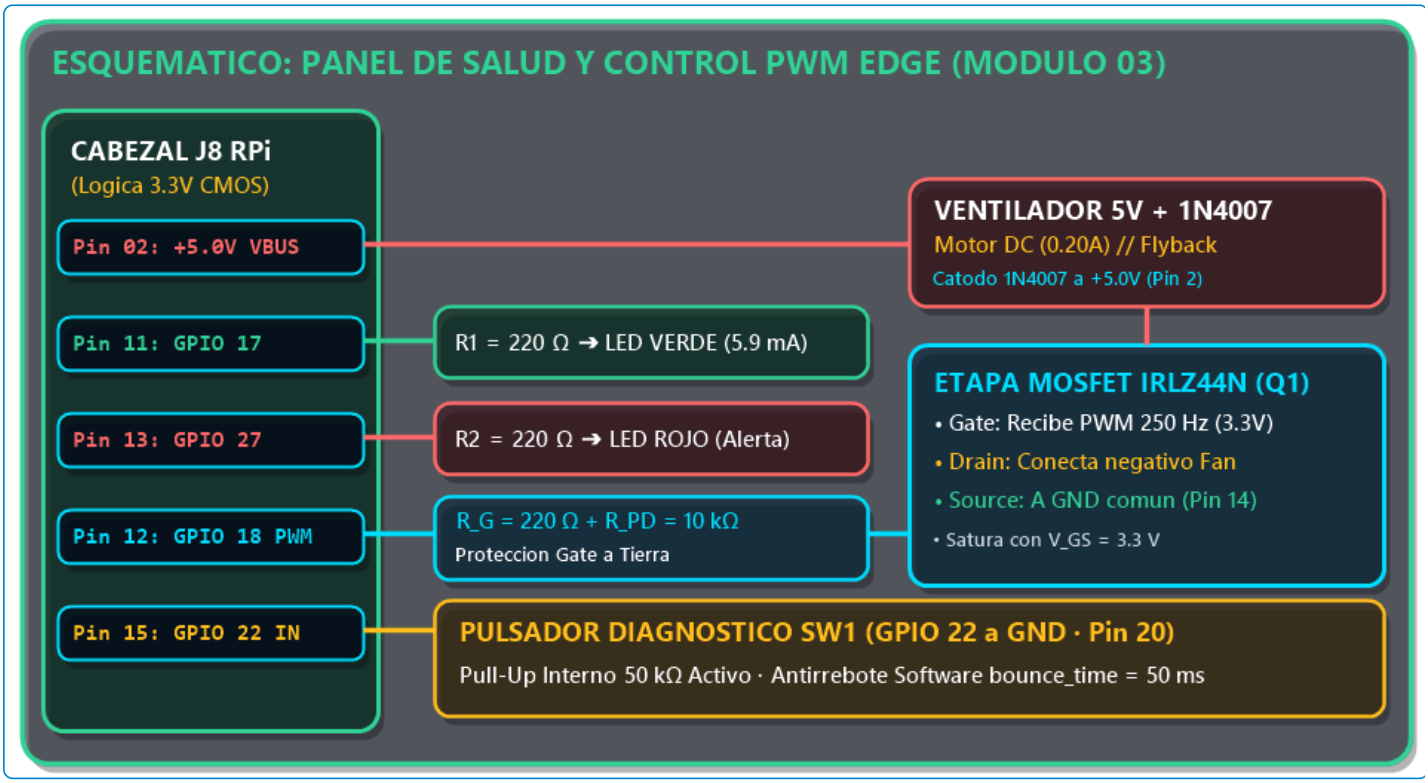
- Verifica que al conectar un Arduino UNO R3 aparezca el enlace simbólico fijo `/dev/arduino_clima`:

```bash

ls -l /dev/arduino_*

```

Paso 4: Armado del Circuito de Supervisión GPIO en Protoboard



- Conecta `GPIO 17` (**Pin físico 11**) ➜ Resistencia $220\ \Omega$ ➜ Ánodo LED Verde ➜ Cátodo a `GND` (**Pin físico 9**).
- Conecta `GPIO 27` (**Pin físico 13**) ➜ Resistencia $220\ \Omega$ ➜ Ánodo LED Rojo ➜ Cátodo a `GND` (**Pin físico 14**).
- Conecta `GPIO 18` (**Pin físico 12**) ➜ Resistencia $220\ \Omega$ ➜ Compuerta (*Gate*) del MOSFET `IRLZ44N`, añadiendo la resistencia *pull-down* de $10\ \text{k}\Omega$ entre *Gate* y `GND`.
- Conecta el ventilador de $5\ \text{V}$ entre el **Pin físico 2 ($+5.0\ \text{V}$)** y el Drenador (*Drain*) del MOSFET, colocando el diodo `1N4007` en antiparalelo (cátodo hacia $+5.0\ \text{V}$).
- Conecta el pulsador normalmente abierto `SW1` entre `GPIO 22` (**Pin físico 15**) y `GND` (**Pin físico 20**).

Paso 5: Despliegue del Demonio en Python y Servicio `systemd`

Puedes ejecutar el aprovisionamiento automático con [`codigo_fuente/provision_edge_rpi.sh`] (file:///D:/_Mecatrobotica/Video_03_Modulo_03_Servidor_Edge_Linux_Raspberry_Pi/material_complementario/codigo_fuente/provision_edge_rpi.sh) o realizar los pasos manualmente:
```

```
# 1. Instalar dependencias del sistema operativo (backend lgpio para RPi 4 y RPi 5)
sudo apt-get update
sudo apt-get install -y python3-venv python3-gpiozero python3-lgpio gpiod i2c-tools

# 2. Crear directorio de producción y entorno virtual heredando librerías C del sistema
sudo mkdir -p /opt/mecatrobotica/edge
sudo chown -R iotadmin:iotadmin /opt/mecatrobotica
python3 -m venv --system-site-packages /opt/mecatrobotica/edge/.venv

# 3. Copiar el script del demonio y otorgar permisos de ejecución
cp edge_health_daemon.py /opt/mecatrobotica/edge/edge_health_daemon.py
chmod +x /opt/mecatrobotica/edge/edge_health_daemon.py

# 4. Instalar y habilitar la unidad de servicio systemd
sudo cp edge-health.service /etc/systemd/system/edge-health.service
sudo systemctl daemon-reload
sudo systemctl enable --now edge-health.service
```

Paso 6: Pruebas de Auditoría y Resiliencia (Inyección de Fallas)

- **Inspección en Tiempo Real de `journald`:**

  ```bash
  journalctl -u edge-health.service -f
  ```

  Presiona el botón físico conectado a `GPIO 22` (Pin 15) y verifica que aparezca instantáneamente la línea `DIAGNÓSTICO MANUAL [GPIO 22] -> Temp SoC: ...`.

- **Prueba de Auto-Recuperación ante Fallo Crítico (`Crash Test`):**

  Obtén el `PID` del proceso y envíale una señal letal `SIGKILL (-9)` simulando un cuelgue abrupto:

  ```bash
  sudo kill -9 $(systemctl show --property MainPID --value edge-health.service)
  ```

  Observa en `journalctl -u edge-health.service -f` cómo `systemd` detecta la caída, espera exactamente `RestartSec=5` segundos y vuelve a levantar el servicio automáticamente restaurando el latido del LED verde.

Listado de Materiales (BOM) y Especificaciones Eléctricas — Módulo 03 (Video 03)

Preparación y Administración de Raspberry Pi como Servidor Edge Linux

> [!IMPORTANT]

> **Archivo Complementario Referenciado en el Video 03:** Este documento detalla todos los componentes electrónicos, herramientas e insumos requeridos para construir el panel local de supervisión y enfriamiento activo del **Servidor Edge Linux (** `edge-health.service` **)** explicado en la **Diapositiva 08, 09, 10 y 13** del **Video 03**.

---

1. Tabla Maestra de Materiales (Bill of Materials — BOM)

Ítem	Código / Designador	Componente	Especificación Técnica Recomendada	Cant.	Función en el Módulo 03
01	`SBC1`	**Raspberry Pi 4 Model B** o **Raspberry Pi 5**	2 GB / 4 GB / 8 GB RAM, SoC ARM64 (`BCM2711` o `BCM2712` + `RP1`), cabezal GPIO de 40 pines a $3.3\ V$	1	Computador monoplaca que actúa como Pasarela Edge Linux *headless*.
02	`PSU1`	**Fuente de Alimentación Oficial USB-C**	$5.1\ V\ DC$ estabilizados / $3.0\ A$ ($15.3\ W$ para RPi 4) o $5.0\ A$ ($27\ W$ PD para RPi 5)	1	Evita caídas de tensión ($< 4.63\ V$) y alertas de *undervoltage* (`0x50005`).
03	`SD1`	**Tarjeta microSD Industrial / Clase A2**	$32\ GB$ o $64\ GB$, estándar **Application Performance Class A2** (mínimo $4000\ IOPS$ lectura aleatoria)	1	Aloja `Raspberry Pi OS Lite (64-bit)` (Debian 12 *Bookworm*) y los diarios de `systemd-journald` .
04	`D_OK`	**LED Verde Difuso de 5 mm**	Tensión directa $V_F \approx 2.0\ V$, corriente de operación segura $I_F \approx 6\ mA$	1	Indicador de latido (*Heartbeat* a $0.5\ Hz$, $200\ ms$ encendido / $1800\ ms$ apagado) en `GPIO 17` (Pin físico 11).
05	`D_ALM`	**LED Rojo Difuso de 5 mm**	Tensión directa $V_F \approx 1.9\ V$, corriente de operación segura $I_F \approx 6.4\ mA$	1	Indicador de alerta por sobretemperatura del SoC ($T_{CPU} \geq 65\ ^\circ C$) en `GPIO 27` (Pin físico 13).
06	`R1, R2`	**Resistencias Limitadoras para LEDs**	$220\ \Omega$ (o $330\ \Omega$), película metálica $1/4\ W$, tolerancia $\pm 1\%$	2	Limitan la corriente extraída de los pines `GPIO 17` y `GPIO 27` muy por debajo del límite absoluto de $16\ mA$.
07	`Q1`	**Transistor MOSFET Canal N de Nivel Lógico**	`IRLZ44N` o `2N7000` / `BS170` (Alt. BJT NPN `2N2222A`), saturación plena con $V_{GS} = 3.3\ V$	1	Conmuta por modulación de ancho de pulso (`PWM` a $250\ Hz$) la corriente del ventilador desde `GPIO 18` (Pin físico 12).
08	`R_GATE`	**Resistencia de Compuerta / Base**	$220\ \Omega$ (para MOSFET) o $1\ k\Omega$ (para BJT `2N2222A`), $1/4\ W$, $\pm 1\%$	1	Protege el pin `GPIO 18` frente a transitorios de carga capacitiva durante la conmutación PWM.
09	`R_PD`	**Resistencia* Pull-Down *de Seguridad**	$10\ k\Omega$, $1/4\ W$, $\pm 1\%$	1	Mantiene la compuerta de `Q1` a $0\ V$ (`GND`) mientras la Raspberry Pi arranca el kernel Linux.
10	`D_FLY`	**Diodo Rectificador* Flyback***	`1N4007` ($1000\ V$, $1\ A$) o Schottky `1N5819`	1	Conectado en antiparalelo con el ventilador para absorber el pico inductivo ($V = -L\ (di)\ /\ (dt)$).
11	`FAN1`	**Ventilador Axial Brushless DC**	$5\ V\ DC$ ($0.15\ A$ - $0.25\ A$), $30\times 30\ mm$ o $40\times 40\ mm$ para gabinete Edge	1	Disipación térmica activa del procesador ARM gobernada por la curva proporcional en `edge_health_daemon.py` .
12	`SW1`	**Pulsador Táctil Normalmente Abierto (N.A.)**	$6\times 6\ mm$ o pulsador de panel industrial	1	Disparo manual de reporte de diagnóstico hacia `journald` conectado entre `GPIO 22` (Pin físico 15) y `GND` .
13	`C_DEB`	**Condensador Cerámico**	$100\ nF$ ($0.1\ \mu F$, código `104`), $50\ V$	1	Desacoplo de ruido y filtrado físico complementario al antirrebote por software (`bounce_time=0.05`).

Ítem	Código / Designador	Componente	Especificación Técnica Recomendada	Cant.	Función en el Módulo 03
		Multicapa (Opcional)			
14	`PB1`	**Protoboard y Cables DuPont**	Protoboard de 400 u 830 puntos + cables DuPont Hembra-Macho de $20\ \text{cm}$	1 kit	Interconexión limpia entre el cabezal de 40 pines de la Raspberry Pi y la etapa de supervisión.
15	`INST1`	**Multímetro Digital True-RMS**	Medición de continuidad, tensión DC y resistencia	1	Verificación obligatoria de ausencia de cortocircuitos entre los rieles de $5.0\ \text{V}$ (Pines 2/4) y $3.3\ \text{V}$ (Pin 1).

---

2. Cálculos de Dimensionamiento Eléctrico del Cabezal GPIO ($3.3\ \text{V}$)

2.1. Límite de Corriente del SoC Broadcom / Chip `RP1`

A diferencia del Arduino UNO (cuyos pines a $5.0\ \text{V}$ toleran hasta $20\ \text{mA}$ nominales), los buffers CMOS de $3.3\ \text{V}$ de la Raspberry Pi tienen restricciones estrictas:

- **Corriente máxima por pin individual:** $\text{I}_{\text{pin(máx)}} \leq 16\ \text{mA}$ (recomendado diseñar para $\text{I}_{\text{pin}} \leq 8\ \text{mA}$).
- **Corriente total combinada en todo el banco GPIO de $3.3\ \text{V}$:** $\text{I}_{\text{banco(máx)}} \leq 50\ \text{mA}$.

2.2. Cálculo de las Resistencias Limitadoras `R1` y `R2` ($220\ \Omega$)

Aplicando la Ley de Tensiones de Kirchhoff al lazo del pin `GPIO 17` ($\text{V}_{\text{GPIO}} = 3.3\ \text{V}$) con el LED verde ($\text{V_F} = 2.0\ \text{V}$):

$$
I_{LED} = \frac{V_{GPIO} - V_F}{R_1} = (3.3\ V - 2.0\ V) / (220\ \Omega) = (1.3\ V) / (220\ \Omega) \approx 5.91\ mA < 16\ mA
$$

La potencia disipada en la resistencia R_1 es:

$$
P_{R1} = I_{LED}^2 \cdot R_1 = (0.00591\ A)^2 \cdot 220\ \Omega \approx 7.68\ mW \ll 250\ mW\ (1/4\ W)
$$

Plano de Circuito y Mapa del Cabezal GPIO de 40 Pines — Módulo 03 (Video 03)

Interconexión Eléctrica del Demonio de Salud Edge (`edge_health_daemon.py`)

> [!IMPORTANT]



---

1. Mapa de Pines Críticos en el Cabezal J8 de 40 Pines (Numeración Física vs. BCM)

> [!CAUTION]

> **REGLA DE ORO DEL CABEZAL DE 40 PINES:** Los **Pines Físicos 2 y 4** entregan $5.0\ V\ DC$ **directos de la fuente USB-C** sin pasar por ningún limitador lógico. Jamás conectes los Pines 2 o 4 a ningún pin `GPIO`, ya que destruirás instantáneamente el SoC o el controlador `RP1` de la Raspberry Pi.

Pin Físico (Impar - Izquierda)	Función / Nombre BCM (Impar)	Función / Nombre BCM (Par)	Pin Físico (Par - Derecha)
Pin 1	`3.3V DC Power` (Referencia Lógica)	`5.0V DC Power` (Riel de Potencia para Ventilador)	**Pin 2**
Pin 3	`GPIO 2` (`I2C1 SDA` — Pull-up 1.8 kΩ interno)	`5.0V DC Power`	**Pin 4**
Pin 5	`GPIO 3` (`I2C1 SCL` — Pull-up 1.8 kΩ interno)	`GND` (Tierra Común Principal)	**Pin 6**
Pin 7	`GPIO 4` (`1-Wire` para sondas `DS18B20`)	`GPIO 14` (`UART0 TXD` @ 3.3 V)	**Pin 8**
Pin 9	`GND` (Retorno para LEDs de Estado)	`GPIO 15` (`UART0 RXD` @ 3.3 V)	**Pin 10**
Pin 11	`GPIO 17` **→ Salida LED Verde (** `Heartbeat` **)**	`GPIO 18` (`PWM0`) **→ Gate MOSFET Ventilador**	**Pin 12**
Pin 13	`GPIO 27` **→ Salida LED Rojo (** `Alerta Temp` **)**	`GND` (Retorno para Etapa MOSFET)	**Pin 14**
Pin 15	`GPIO 22` **→ Entrada Botón Diagnóstico (** `Pull-Up` **)**	`GPIO 23` (Libre)	**Pin 16**
Pin 17	`3.3V DC Power`	`GPIO 24` (Libre)	**Pin 18**
Pin 19	`GPIO 10` (`SPI0 MOSI`)	`GND` (Retorno para Botón `SW1`)	**Pin 20**

2. Diagrama Esquemático de Conexiones del Laboratorio 03

```
flowchart LR
    subgraph RPi["Cabezal J8 de 40 Pines — Raspberry Pi 4B / 5 (Lógica 3.3V)"]
        P2["Pin Físico 2<br/>+5.0V DC (V_BUS)"]
        P11["Pin Físico 11<br/>GPIO 17 (BCM)"]
        P13["Pin Físico 13<br/>GPIO 27 (BCM)"]
        P12["Pin Físico 12<br/>GPIO 18 (PWM)"]
        P15["Pin Físico 15<br/>GPIO 22 (Pull-Up 50kΩ)"]
        GND["Pines Físicos 6, 9, 14, 20<br/>GND (Tierra Común 0V)"]
    end

    subgraph StageLEDs["1. Indicadores de Estado (I_pin ≈ 6 mA)"]
        P11 --> R1["R1 = 220 Ω (1/4W)"] --> LED_G["LED Verde (Heartbeat 0.5 Hz)<br/>Ánodo (+) ➔ Cátodo (-)"] --> GND
        P13 --> R2["R2 = 220 Ω (1/4W)"] --> LED_R["LED Rojo (Alerta ≥ 65 °C)<br/>Ánodo (+) ➔ Cátodo (-)"] --> GND
    end

    subgraph StageFan["2. Etapa de Potencia PWM para Ventilador (250 Hz)"]
        P12 --> RG["R_GATE = 220 Ω"] --> GATE["Compuerta (Gate)<br/>MOSFET IRLZ44N (Q1)"]
        GATE --> RPD["R_PD = 10 kΩ (Pull-Down)"] --> GND
        P2 --> FAN["Ventilador DC 5V (+)"]
        P2 --> D_CATH["Cátodo Diodo 1N4007 (Flyback)"]
        FAN --> DRAIN["Drenador (Drain) Q1"]
        DRAIN --> D_ANOD["Ánodo Diodo 1N4007"]
        DRAIN --> SOURCE["Fuente (Source) Q1"] --> GND
    end

    subgraph StageBtn["3. Entrada de Diagnóstico con Antirrebote"]
        P15 --> SW1["Pulsador N.A. (SW1)<br/>bounce_time = 50 ms"] --> GND
    end
```

---

3. Tabla de Verificación Punto a Punto con Multímetro (Antes de Energizar)

- **Prueba de Aislamiento de Rieles ($5.0\ \text{V}$ vs. $3.3\ \text{V}$):** Medir resistencia entre el **Pin Físico 1 ($3.3\ \text{V}$)** y el **Pin Físico 2 ($5.0\ \text{V}$)** con la Raspberry Pi apagada. Debe marcar alta impedancia ($> 10\ \text{k}\Omega$).
- *Prueba de Polaridad del Diodo* Flyback *(`1N4007`): **Verificar que la** banda plateada (cátodo) **del diodo** `1N4007` **esté conectada al positivo de** $+5.0\ \text{V}$ (Pin 2) **y el ánodo al** Drenador (*Drain)* del MOSFET. Si se invierte, producirá un cortocircuito directo cuando el MOSFET conduzca.
- *Prueba de Resistencia* Pull-Down *de Compuerta: Medir entre la compuerta (*Gate*) del MOSFET y `GND` (Pin 14); el multímetro debe registrar $\approx 10.0\ \text{k}\Omega$.