Manual Tecnico
“Mikrotik-Redes-Automatizacion-Scada”

Versión 1.0 – Agosto 2025

índice **Pag**

Diagramas

Figuras 01–02 · Topología del sistema

Qué muestran:

- Figura 01 (Esquemático): arquitectura lógica y conexiones entre equipos.
- Figura 02 (Con fotos): la misma topología con imágenes reales de los dispositivos.

Elementos y enlaces (usados):

- Raspberry Pi 3 → ejecuta program2.py.
    - Función: hace ping cada 5 s al host 10.0.0.20.
    - Si el ping OK envía ‘G’ por USB /dev/ttyUSB0 (serial 9600 8N1). Si falla, envía ‘R’.
    - Enlace al core por Ethernet.
- Core MikroTik (CRS) → switch/router central; de él salen los enlaces hacia:
    - hAP lite #1, #2 y #3 (uplinks Troncales/Access).
- Arduino MEGA → recibe por USB el byte ‘G’/‘R’ de la RPi.
    - Salida D8 conectada a IN del módulo de relés (con 5 V y GND comunes).
- Módulo de relés 4 canales → se utiliza CH1 en NO (Normalmente abierto).
- Bloque de 6 luces piloto 110 VAC → conectadas en serie; su alimentación de línea pasa por el contacto NO del relé.

Camino de señal y energía (resumen):
RPi (ping/USB) → Arduino (D8) → Relé CH1 (NO) → Línea 110 V → Serie de 6 pilotos → Retorno a N.

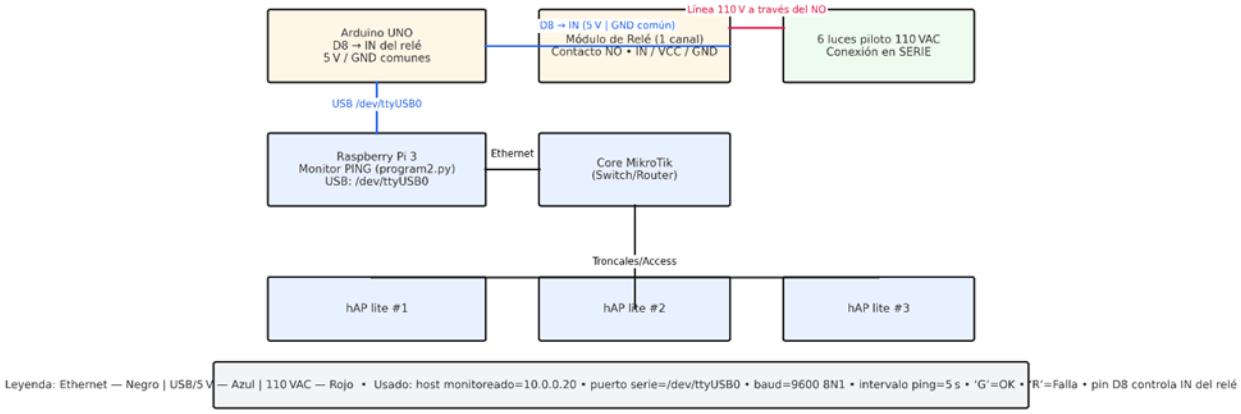
Arduino UNO
D8 → IN del relé
5 V / GND comunes
D8 → IN (5 V | GND común)
Módulo de Relé (1 canal)
Contacto NO • IN / VCC / GND
Línea 110 V a través del NO
6 luces piloto 110 VAC
Conexión en SERIE
USB /dev/ttyUSB0
Raspberry Pi 3
Monitor PING (program2.py)
USB: /dev/ttyUSB0
Ethernet
Core MikroTik
(Switch/Router)
Troncales/Access
hAP lite #1
hAP lite #2
hAP lite #3
Leyenda: Ethernet — Negro | USB/5 V — Azul | 110 VAC — Rojo • Usado: host monitoreado=10.0.0.20 • puerto serie=/dev/ttyUSB0 • baud=9600 8N1 • intervalo ping=5 s • 'G'=OK • 'R'=Falla • pin D8 controla IN del relé

Fig 1.

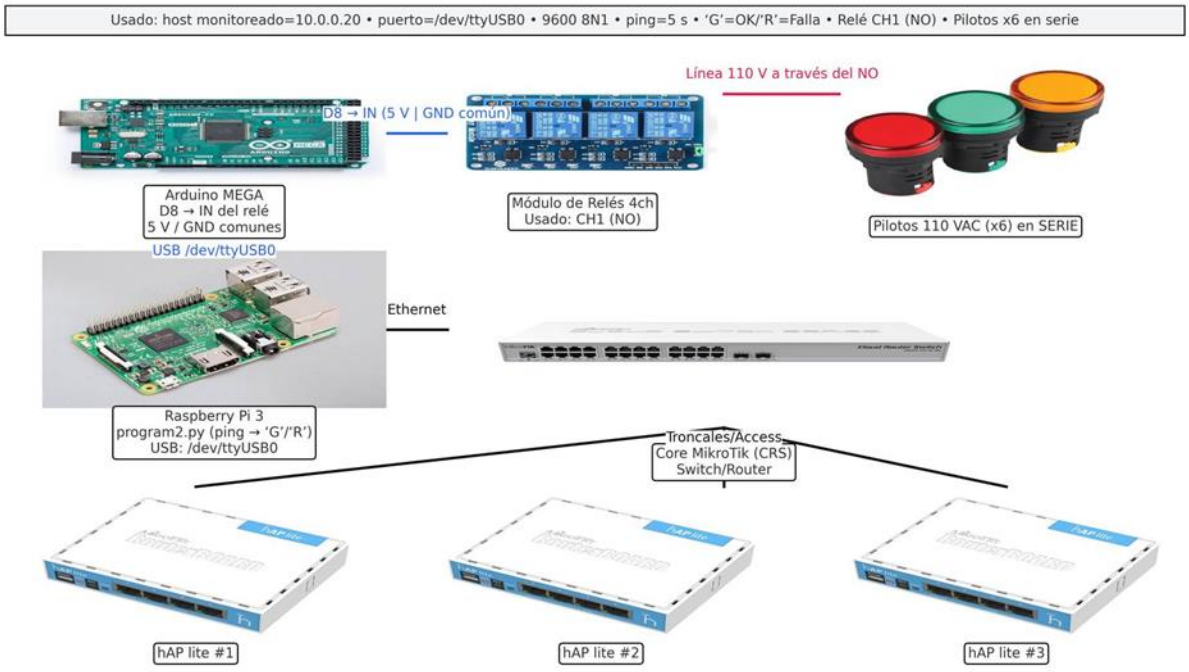
Usado: host monitoreado=10.0.0.20 • puerto=/dev/ttyUSB0 • 9600 8N1 • ping=5 s • 'G'=OK/'R'=Falla • Relé CH1 (NO) • Pilotos x6 en serie
Línea 110 V a través del NO
D8 → IN (5 V | GND común)
Arduino MEGA
D8 → IN del relé
5 V / GND comunes
Módulo de Relés 4ch
Usado: CH1 (NO)
Pilotos 110 VAC (x6) en SERIE
USB /dev/ttyUSB0
Ethernet
Raspberry Pi 3
program2.py (ping → 'G'/'R')
USB: /dev/ttyUSB0
Troncales/Access
Core MikroTik (CRS)
Switch/Router
hAP lite #1
hAP lite #2
hAP lite #3

Fig. 2

Figuras 03–04 · Cableado de potencia y control

Qué muestran:

- **Figura 03 (Esquemático):** trazado eléctrico de **L/N** y del control a 5 V.
- **Figura 04 (Con fotos):** el mismo cableado sobre las imágenes de **relé** y **pilotos**.

Potencia 110 VAC (usado):

- **L (fase) → Fusible → Relé CH1 (NO) → 6 pilotos en SERIE** → retorno a **N (neutro)**.
- El **NO** se representa con un **gap** (trazo discontinuo); al cerrar, la **L** alimenta la cadena completa.

Control 5 V (usado):

- **Arduino MEGA → D8 → IN** del módulo de relés.
- **VCC = 5 V** y **GND** comunes entre Arduino y módulo de relé (módulo **opto-aislado**).
- **USB** entre **RPi 3** y **Arduino** en **/dev/ttyUSB0** (**9600 8N1**).

Relación con el código:

- program2.py en la RPi decide **‘G’** (OK) o **‘R’** (Falla) según el ping a **10.0.0.20** cada **5 s**.
- El sketch del Arduino lee el byte por serie y pone **D8=HIGH** (si ‘G’) o **D8=LOW** (si ‘R’), conmutando CH1.

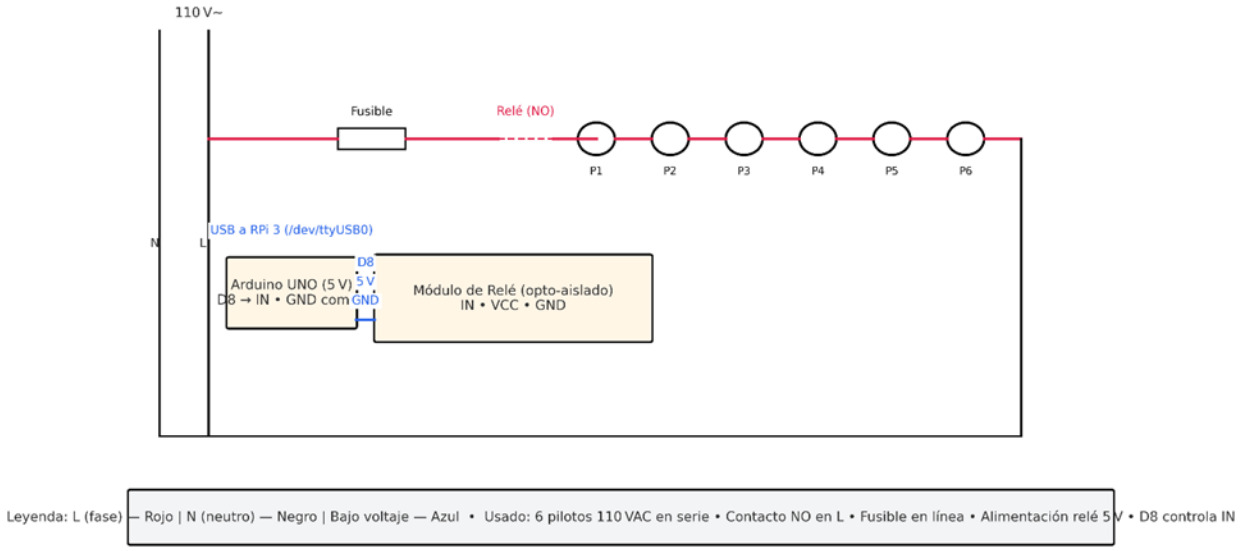
110 V~
Fusible
Relé (NO)
P1
P2
P3
P4
P5
P6
N
L
USB a RPi 3 (/dev/ttyUSB0)
Arduino UNO (5 V)
D8 → IN • GND com
D8
5 V
GND
Módulo de Relé (opto-aislado)
IN • VCC • GND
Leyenda: L (fase) — Rojo | N (neutro) — Negro | Bajo voltaje — Azul • Usado: 6 pilotos 110 VAC en serie • Contacto NO en L • Fusible en línea • Alimentación relé 5 V • D8 controla IN

Fig. 3

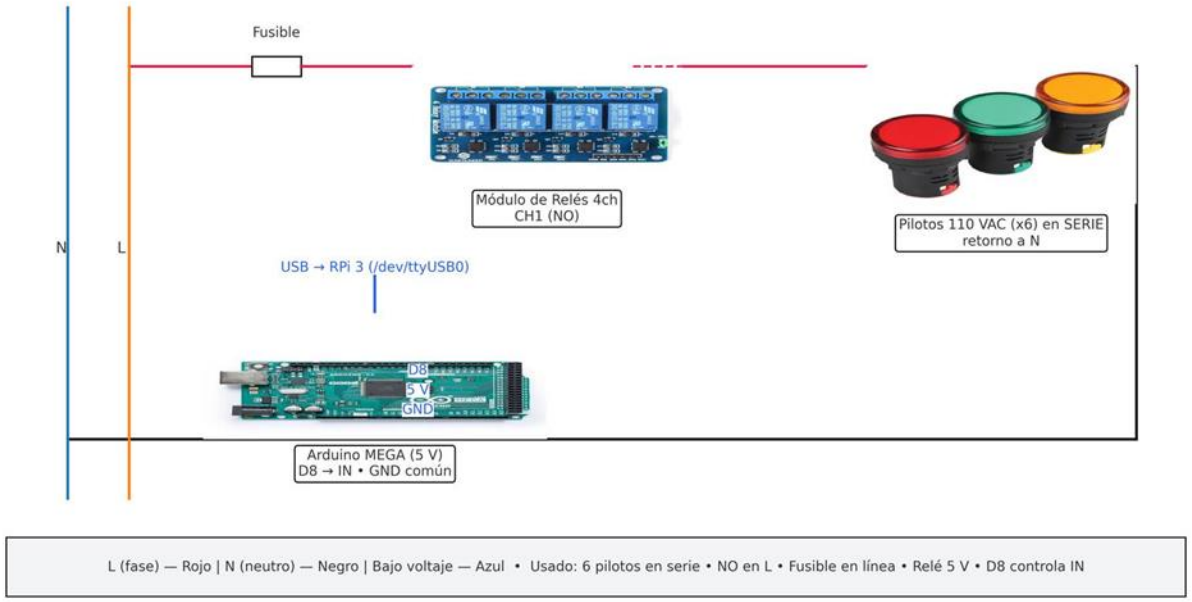
Fusible
Módulo de Relés 4ch
CH1 (NO)
Pilotos 110 VAC (x6) en SERIE
retorno a N
N
L
USB → RPi 3 (/dev/ttyUSB0)
D8
5 V
GND
Arduino MEGA (5 V)
D8 → IN • GND común
L (fase) — Rojo | N (neutro) — Negro | Bajo voltaje — Azul • Usado: 6 pilotos en serie • NO en L • Fusible en línea • Relé 5 V • D8 controla IN

Fig. 4

Figuras 05–06 · Flujo de estados RPi → Arduino → Relé → Pilotos

Qué muestran:

- Figura 05 (Esquemático): la lógica de estados con líneas verde (OK) y roja (Falla).
- Figura 06 (Con fotos): el mismo flujo sobre imágenes reales de RPi, Arduino, relé y pilotos.

Secuencia de operación (usada):

1. Ping OK
    - RPi envía ‘G’ por /dev/ttyUSB0 → Arduino pone D8=HIGH → el relé cierra el NO → la L queda conectada → pilotos encendidos.
2. Ping falla
    - RPi envía ‘R’ → Arduino pone D8=LOW → el relé abre el NO → la L queda abierta → pilotos apagados.

Parámetros operativos (comunes a los 3 pares):
Host monitoreado 10.0.0.20 · Intervalo 5 s · Puerto /dev/ttyUSB0 · Serie 9600 8N1 · Protocolo ‘G’/‘R’ · D8 controla IN de CH1 (NO) · 6 pilotos a 110 VAC en serie con fusible en la línea.

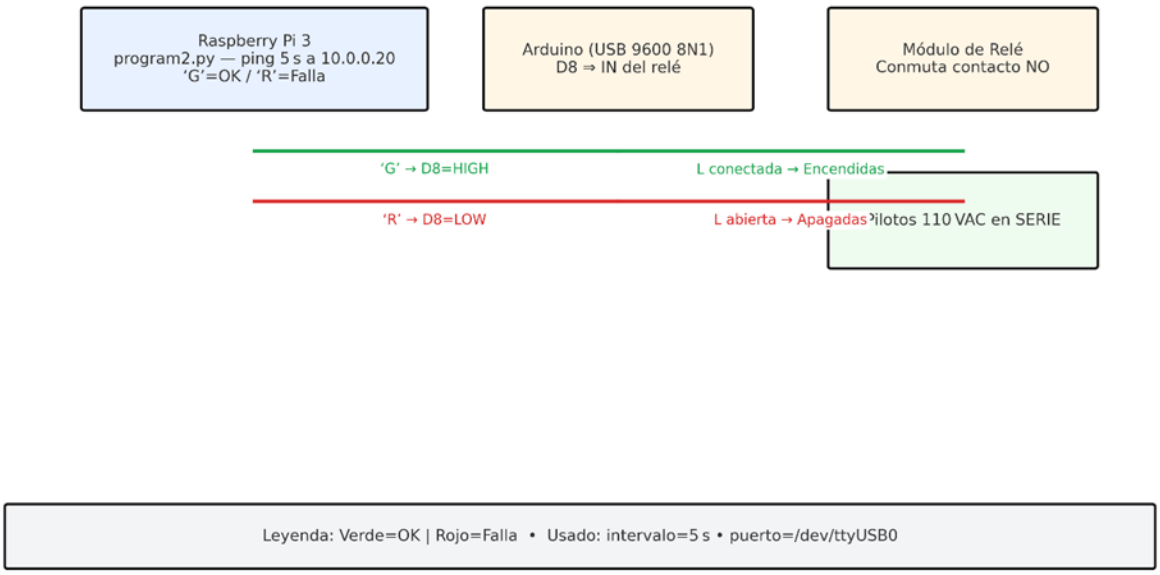
Raspberry Pi 3
program2.py — ping 5 s a 10.0.0.20
'G'=OK / 'R'=Falla
Arduino (USB 9600 8N1)
D8 ⇒ IN del relé
Módulo de Relé
Conmuta contacto NO
'G' → D8=HIGH
L conectada → Encendidas
'R' → D8=LOW
L abierta → Apagadas
Pilotos 110 VAC en SERIE
Leyenda: Verde=OK | Rojo=Falla • Usado: intervalo=5 s • puerto=/dev/ttyUSB0

Fig. 5

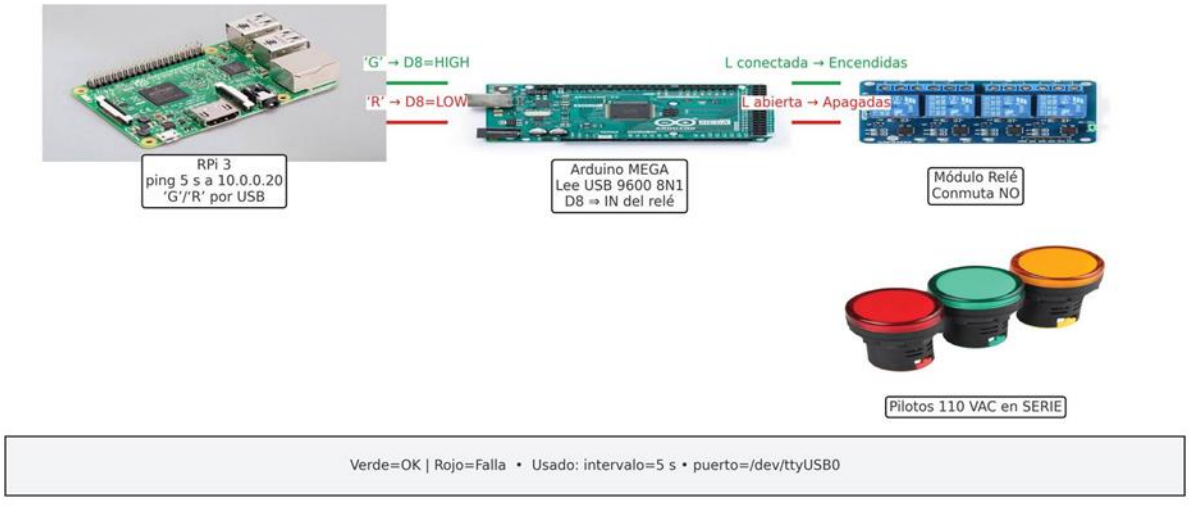
'G' → D8=HIGH
'R' → D8=LOW
L conectada → Encendidas
L abierta → Apagadas
RPi 3
ping 5 s a 10.0.0.20
'G'/'R' por USB
Arduino MEGA
Lee USB 9600 8N1
D8 ⇒ IN del relé
Módulo Relé
Conmuta NO
Pilotos 110 VAC en SERIE
Verde=OK | Rojo=Falla • Usado: intervalo=5 s • puerto=/dev/ttyUSB0

Fig. 6

Software

Este programa es una herramienta gráfica en Python para administrar un router MikroTik de forma directa y en tiempo real usando SSH y una interfaz con ttkbootstrap (tema oscuro).

Su objetivo es permitir que un usuario, sin usar la consola del router, pueda realizar tareas comunes de configuración y monitoreo mediante botones y formularios.

Este código convierte tareas que normalmente se harían en la terminal del MikroTik en una aplicación gráfica amigable, capaz de configurar y monitorear el router en tiempo real, además de integrarse con hardware externo para señalización de estado.

Esta es la vista principal de nuestro programa

Fig. 7

run_ssh(cmd: str) -> str

Propósito en el programa:

Es la función central de comunicación SSH.

Todo lo que el programa haga en el Mikrotik (consultar, cambiar nombre, agregar IP, crear DHCP, rutas, etc.) pasa por aquí.

- Recibe un comando RouterOS (cmd).

- Ejecuta sshpass para conectarse al router con las credenciales y ejecutar ese comando.

- Devuelve la salida del comando como texto o lanza un error si falla.

Importante: Sin esta función, ninguna de las demás podría modificar o leer datos del Mikrotik.

show_output(data: str)

Propósito:

Mostrar información en el área de salida de texto (ScrolledText) de la ventana principal.

- Limpia lo que había antes.

- Inserta el nuevo texto.

- Se usa para mostrar resultados de consultas (consulta_name, consulta_ip).

Aquí tenemos las funciones respectivas a cada apartado del programa:

save_name()

Propósito:

Cambiar el nombre del router.

- Obtiene el texto del campo var_name.

- Si está vacío, muestra advertencia.

- Si hay nombre, ejecuta:

system identity set name=<nuevo_nombre>
usando run_ssh.

- Muestra mensaje de éxito.

consulta_name()

Propósito:

- Consultar el nombre actual del router (identity).

- Llama a run_ssh("system identity print").

- Muestra el resultado con show_output.

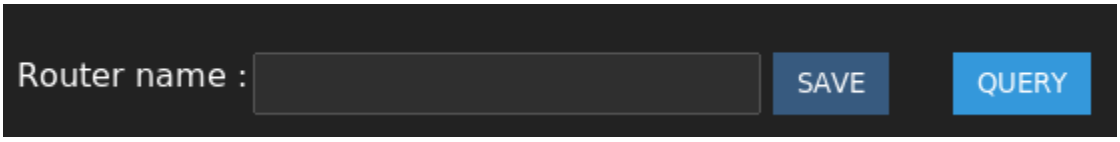

Fig. 8

save_addr()

Propósito:

Agregar una dirección IP al router.

- Obtiene IP, interfaz y comentario de los campos (var_ip, var_iface, var_comm).

- Si IP o interfaz están vacíos → advertencia.

- Ejecuta:

ip address add address=<ip> interface=<iface> comment='<comentario>'

- Muestra mensaje de éxito.

delete_addr()

Propósito:

Eliminar una dirección IP existente.

- Obtiene índice desde un Combobox (combo_del).

- Verifica que sea un número válido (1–5 en este diseño).

- Ejecuta:

ip address remove <índice>

- Muestra mensaje de éxito.

consulta_ip()

Propósito:

Consultar la lista de direcciones IP configuradas en el Mikrotik.

- Llama a run_ssh("ip address print").

- Muestra el resultado con show_output.

Fig. 9

ventana_dhcp()

Propósito:

- Abrir una ventana secundaria para crear un servidor DHCP.

- Pide datos como Pool, rangos de IP, nombre del servidor, interfaz, red, gateway, DNS.

- Al pulsar "SAVE" ejecuta en orden:

  - ip pool add ... (crear pool de direcciones).
  - ip dhcp-server add ... (crear el servidor DHCP en la interfaz).
  - ip dhcp-server network add ... (asociar red, gateway y DNS).

- Mensaje de confirmación al final.

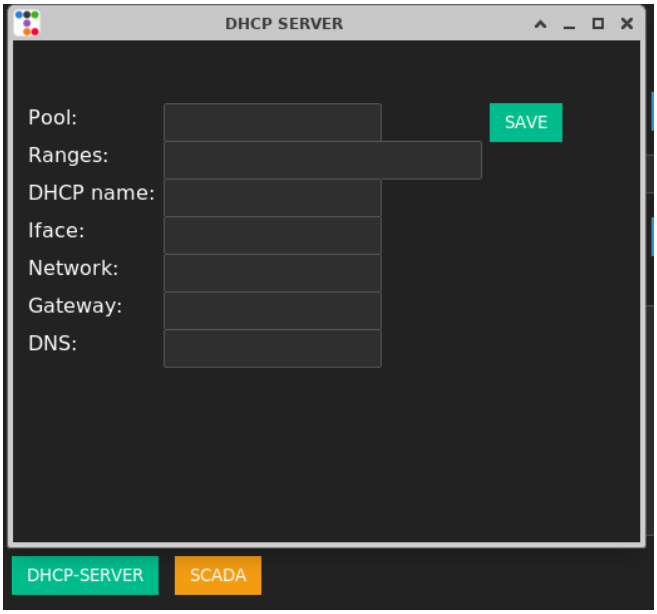

Fig. 10

ventana_scada()

Propósito:

Abrir una ventana para monitorear el estado de red (ping) con un indicador gráfico.

- Muestra imágenes ON/OFF (ethernetON.gif, ethernetOFF.gif).

- Tiene un Checkbutton para habilitar el monitor.

- Cambia color y texto según haya ping al Mikrotik.

- Si hay conexión con Arduino, envía "G" (verde) o "R" (rojo).

hay_ping(ip)

Propósito:

Probar si un host responde al ping.

- Usa subprocess.run con ping para 1 paquete.

- Devuelve True si responde, False si no.

actualizar_estado_ping(ok)

Propósito:

Actualizar la interfaz gráfica para reflejar si hay conexión.

- Cambia el texto ON/OFF y color verde/rojo.

- Cambia la imagen a ON/OFF.

- Actualiza el Checkbutton.

hilo_ping()

Propósito:

Hacer el monitoreo de conexión en segundo plano.

- Ciclo infinito cada 5 segundos:

- Verifica ping.
- Si hay Arduino, envía señal G/R.
- Llama a actualizar_estado_ping para refrescar GUI.

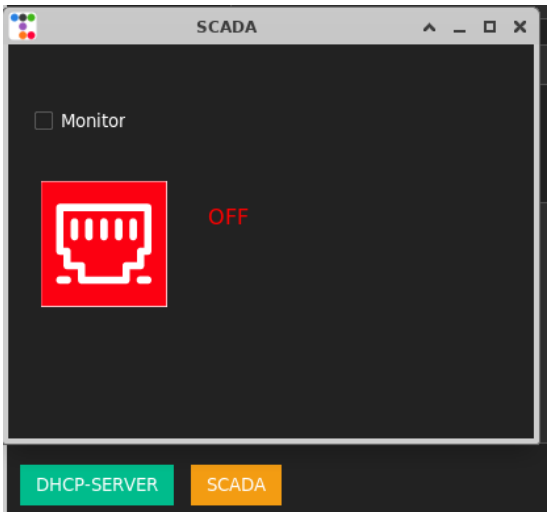

Fig. 11

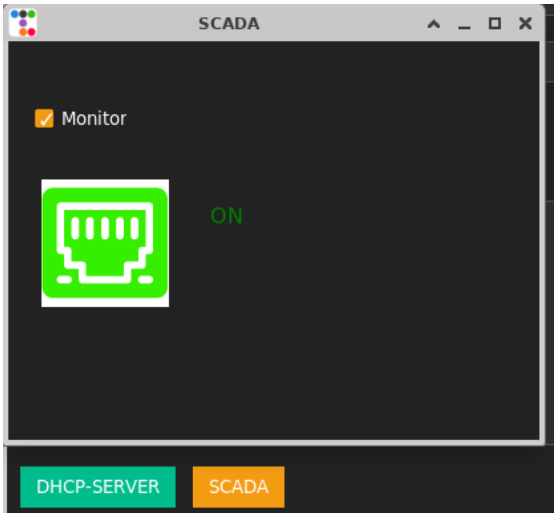

Fig. 12

Bloque de conexión a Arduino

Propósito:

Intentar abrir conexión serie (/dev/ttyUSB0) para enviar señales ON/OFF.

- Si falla, deja arduino = None y el programa sigue.

Resumen de funciones por tipo:

- Gestión de nombre/IP: save_name, save_addr, delete_addr, consulta_name, consulta_ip

- Gestión avanzada: ventana_dhcp, ventana_rutas

- Monitoreo: ventana_scada, hilo_ping, hay_ping, actualizar_estado_ping

- Base de comunicación: run_ssh

- Interfaz: show_output

Flujo General De La Aplicación:

- El usuario abre la aplicación y ve la ventana principal.

- Introduce datos (nombre del router, IP, interfaz, etc.) y pulsa botones.

- La aplicación traduce esas acciones a comandos RouterOS.

- Los comandos se envían por SSH al MikroTik.

- El resultado o confirmación se muestra en pantalla.

- Paralelamente, un hilo en segundo plano verifica si hay conexión y actualiza el estado visual y/o envía señal al Arduino.

Anexos

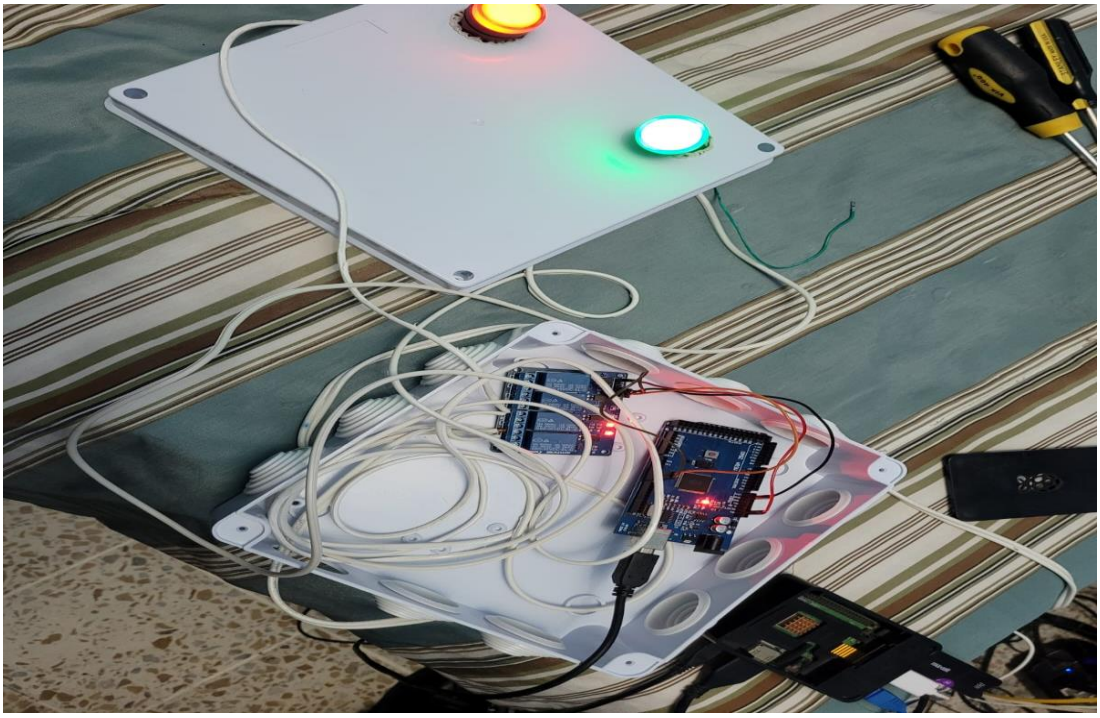

Fig. 13

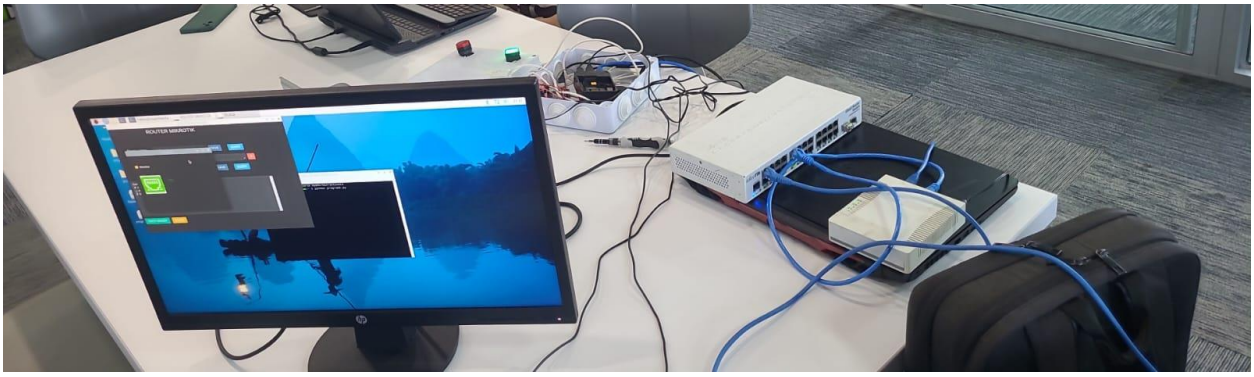

Fig. 14