Modo 1: Temporizado Regresivo:

Este modo permite configurar el relé para que se active durante un período de tiempo determinado después de recibir una señal de activación.

Después de que se activa, el relé se desactiva automáticamente una vez que ha transcurrido el tiempo especificado.

Es útil en aplicaciones donde se requiere que el relé permanezca activo durante un tiempo limitado después de recibir una señal de activación.

### Modo 2: Programación Diaria:

En este modo, el relé se programa para activarse y desactivarse en horarios específicos todos los días.

Se pueden configurar múltiples horarios de activación y desactivación a lo largo del día, lo que permite una flexibilidad en la automatización de tareas programadas.

Es útil para automatizar dispositivos según un horario establecido, como encender luces por la noche o controlar sistemas de riego en horarios específicos.

### Modo 3: Enclavamiento:

Este modo proporciona un comportamiento de tipo interruptor de palanca, donde el relé cambia su estado (activo o inactivo) cada vez que recibe una señal de activación.

Una vez activado, el relé permanece en ese estado hasta recibir una nueva señal de activación, momento en el que cambia a su estado opuesto.

Es útil para aplicaciones donde se desea un control manual simple, como encender o apagar una luz con un interruptor de pared.

### Modo 4: Activación por Eventos Externos:

En este modo, el relé se activa o desactiva en respuesta a eventos externos, como cambios en el estado de un sensor, la detección de movimiento, señales recibidas de dispositivos remotos, etc.

Proporciona una respuesta flexible y dinámica a condiciones externas cambiantes.

Es útil en situaciones donde se requiere una activación basada en eventos específicos en lugar de un horario o temporizador predefinido.

# Clase ConfigManager

La clase **ConfigManager** proporciona métodos para cargar, guardar y manipular datos de configuración almacenados en un archivo JSON.

## Métodos Públicos

**void loadConfig()**

Carga la configuración desde el archivo JSON especificado en la ruta estática **configFilePath**.

**void saveConfig()**

Guarda la configuración actual en el archivo JSON especificado en la ruta estática **configFilePath**.

### Métodos para la configuración de Wifi

* **bool getWifiActive()**: Devuelve el estado de activación de la red Wifi.
* **void setWifiActive(bool active)**: Establece el estado de activación de la red Wifi.
* **String getWifiSSID()**: Devuelve el SSID de la red Wifi.
* **void setWifiSSID(String ssid)**: Establece el SSID de la red Wifi.
* **String getWifiPassword()**: Devuelve la contraseña de la red Wifi.
* **void setWifiPassword(String password)**: Establece la contraseña de la red Wifi.
* **String getWifiIP()**: Devuelve la dirección IP de la red Wifi.
* **void setWifiIP(String ip)**: Establece la dirección IP de la red Wifi.
* **String getWifiSubnet()**: Devuelve la máscara de subred de la red Wifi.
* **void setWifiSubnet(String subnet)**: Establece la máscara de subred de la red Wifi.
* **String getWifiGateway()**: Devuelve la puerta de enlace de la red Wifi.
* **void setWifiGateway(String gateway)**: Establece la puerta de enlace de la red Wifi.

### Métodos para la configuración del Dispositivo K1 (y similares)

* **bool getK1Active()**: Devuelve el estado de activación del dispositivo K1.
* **void setK1Active(bool active)**: Establece el estado de activación del dispositivo K1.
* **String getK1Mode()**: Devuelve el modo de funcionamiento del dispositivo K1.
* **void setK1Mode(String mode)**: Establece el modo de funcionamiento del dispositivo K1.
* **String getK1Name()**: Devuelve el nombre del dispositivo K1.
* **void setK1Name(String name)**: Establece el nombre del dispositivo K1.
* **bool getK1State()**: Devuelve el estado actual del dispositivo K1.
* **void setK1State(bool state)**: Establece el estado actual del dispositivo K1.
* **unsigned long getK1Timer()**: Devuelve el temporizador del dispositivo K1.
* **void setK1Timer(int timer)**: Establece el temporizador del dispositivo K1.
* **unsigned long getK1ActivationTime()**: Devuelve el tiempo de activación del dispositivo K1.
* **void setK1ActivationTime(unsigned long activationTime)**: Establece el tiempo de activación del dispositivo K1.
* **unsigned long getK1DeactivationTime()**: Devuelve el tiempo de desactivación del dispositivo K1.
* **void setK1DeactivationTime(unsigned long deactivationTime)**: Establece el tiempo de desactivación del dispositivo K1.

## Uso

Para utilizar la clase **ConfigManager**, crea una instancia de la misma y utiliza sus métodos para cargar, guardar y manipular la configuración según sea necesario.

# Clase WifiConfig

La clase **WifiConfig** proporciona métodos para configurar y administrar la conexión WiFi en dispositivos ESP32.

Constructores

* **WifiConfig()**: Constructor predeterminado.

Métodos Públicos

* **void initWifi(const char \*ssid, const char \*password)**: Conecta el dispositivo a una red WiFi especificada por un SSID y una contraseña.
* **void initAP(const char \*ssid, const char \*password)**: Inicia un punto de acceso (AP) con un SSID y una contraseña especificados.
* **void initWifiPlus(const char \*ssid, const char \*password, const char \*apSsid, const char \*apPassword)**: Conecta el dispositivo a una red WiFi y establece un punto de acceso simultáneo.
* **void searchNetworks()**: Escanea y lista las redes WiFi disponibles.
* **void configStaticIp(IPAddress ip, IPAddress gateway, IPAddress subnet)**: Configura una dirección IP estática.
* **String getConnectedSSID()**: Obtiene el SSID de la red WiFi a la que está conectado el dispositivo.
* **IPAddress getIPAddress()**: Obtiene la dirección IP asignada al dispositivo.
* **IPAddress getSubnetIP()**: Obtiene la dirección de subred del dispositivo.
* **IPAddress getGatewayIP()**: Obtiene la dirección IP del gateway.
* **String getMACAddress()**: Obtiene la dirección MAC del dispositivo.
* **int getSignalStrength()**: Obtiene la intensidad de la señal WiFi.
* **void disconnect()**: Desconecta el dispositivo del WiFi.
* **void stopAP()**: Detiene el punto de acceso (AP).
* **void reconnect()**: Reinicia la conexión WiFi.
* **void restartAP()**: Reinicia el punto de acceso (AP).