Informe sobre proyecto implementado "Sistema de configuración inicial para dispositivosESP32 vía Wi-Fi".
Código, conexiones y todo lo necesario para este proyecto se encuentra en https://github.com/SimonAulet/Configurador-ACySE

Sobre el proyecto

Se trata del prototipo de un dispositivo de interface de configuración de máquinas usando como base una placa de desarrollo ESP32.

Principio de funcionamiento

Pines específicos de la placa de desarrollo se conectan a actuadores, leds, sensores, etc. de la máquina, según la aplicación. El estado de cada pin determina un tipo de configuración. Por ejemplo; en el caso de un horno, pin_13 en alto puede significar luz interior encendida por defecto.
El usuario se conecta a una red WiFi emitida por el microcontrolador, el cual redirige todas las peticiones de DNS a un captive portal; una página web almacenada en la memoria del dispositivo. De esta manera, el usuario al conectarse a la red es dirigido a una interface web desde la cual puede ver la configuración del aparato y modificarla. Luego, al almacenarla, la configuración se graba en la memoria flash y se mantiene una vez que el usuario se desconecta e incluso cuando el dispositivo se re-inicia.

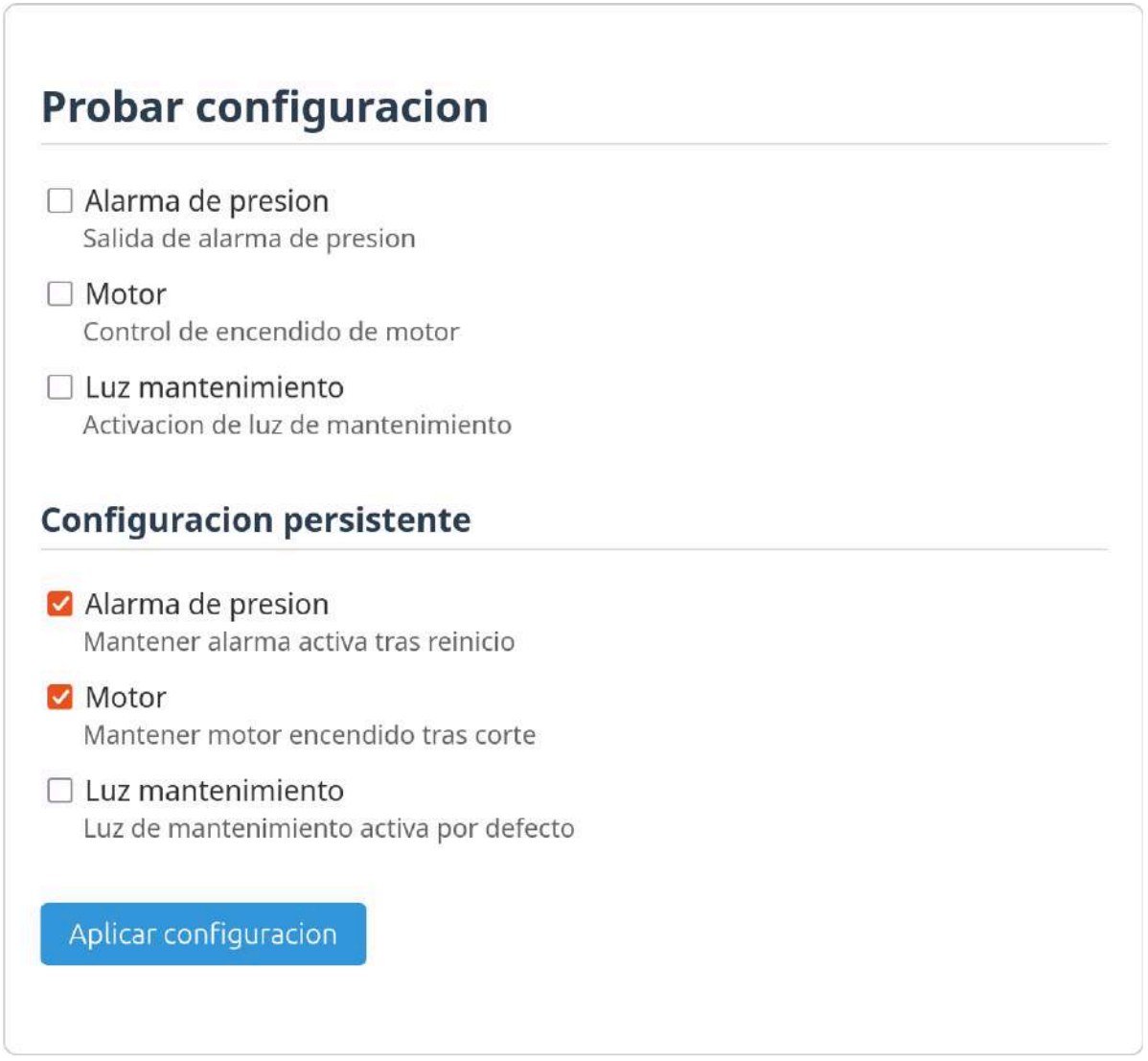
Motivación

La configuración de ciertos dispositivos suele se limitada o incómoda de hacer. Es muy común que las máquinas ofrezcan un LCD de 16x2 con 3 botones mediante los cuales el usuario tiene que, manual en mano, hacer configuraciones muy complejas. Esto genera propensión a

errores y mucha frustración a la hora de configurar máquinas. Por nombrar algunos ejemplos, esto pasa en heladeras comerciales, cámaras de frío, hornos, deshidratadoras, calderas, etc.

- **Colocar un display grande táctil para configurar el aparato**: No siempre es factible; principalmente por los costes de poner un display de estas condiciones. Un display táctil es muy propenso a romperse y no sobrevive en todas las condiciones (uso exterior, dentro de cocinas con alta temperatura y harina volando, vibraciones de motores en ciertas máquinas, etc.) lo cual complica muchísimo.
- **Diseñar una app personalizada** Hubo una gran moda, principalmente en electrodomésticos, a diseñar apps para controlarlos. El problema de estas apps es, principalmente, el coste de mantenimiento. Mantener una app a lo largo de los años para distintas versiones de Android e IOS y adaptarla a los permanentes cambios de cada ecosistema es extremadamente caro, lo cual lleva a que, normalmente, las aplicaciones de este estilo estén muy descuidadas y sean muy propensas a fallas. Además, el publico general esta harto de tener que instalar una app para cada cosa, perdiéndose la simplicidad de operar una máquina.

- **Limitar las opciones a configuraciones pre-establecidas**:
  Suele ser la más usada y aunque funciona, limita el potencial de las máquinas.
  Con esto en mente, mi solución propone que la configuración se haga a través de la página web servida por el microcontrolador mediante protocolos http simples y fiables. A la configuración no se accede cada vez que se quier usar el aparato, solo cuando se va a modificar algo. Por ejemplo, configurar una sonda exterior que se añade a una caldera para aumentar su eficiencia. Mediante interface web se puede incluso dar ayuda e instrucciones sobre el uso del aparato y la funcionalidad de las configuraciónes



Alcance

Este sistema esta pensado para aparatos principalmente domésticos o comerciales, operados por personas con conocimientos técnicos medios

/ bajos de manera esporádica.

Funcionamiento

A continuación se detallan las funciones relevantes del código

1. Se parte de el ejemplo provisto por Espressif de un captive portal(https://github.com/espressif/esp-idf/tree/v5.4.1/examples/protocols/http_server/captive_portal) el cual ya tiene configurado una redirección de dns robusta y bien mantenida.
2. Se genera un tipo de dato que representa un ítem de configuración mediante `llave - valor`

```
typedef struct {
    char key[16];
    int32_t val;
} pin_config_T;
```

3. Se genera una configuración de pines por defecto. Si no hay nada grabado en la flash se van a usar estos valores. En este caso simplemente se usaron dos pines en bajo y uno en alto. Notese que esta configuración únicamente se va a tener en cuenta cuando la flash permanente es borrada. Luego por más que se apague el aparato se usa la ultima configuración guardada y no la default

```
const pin_config_T defaults[] = {
 {"pin_13", 0},
 {"pin_12", 0},
 {"pin_14", 1}};
```

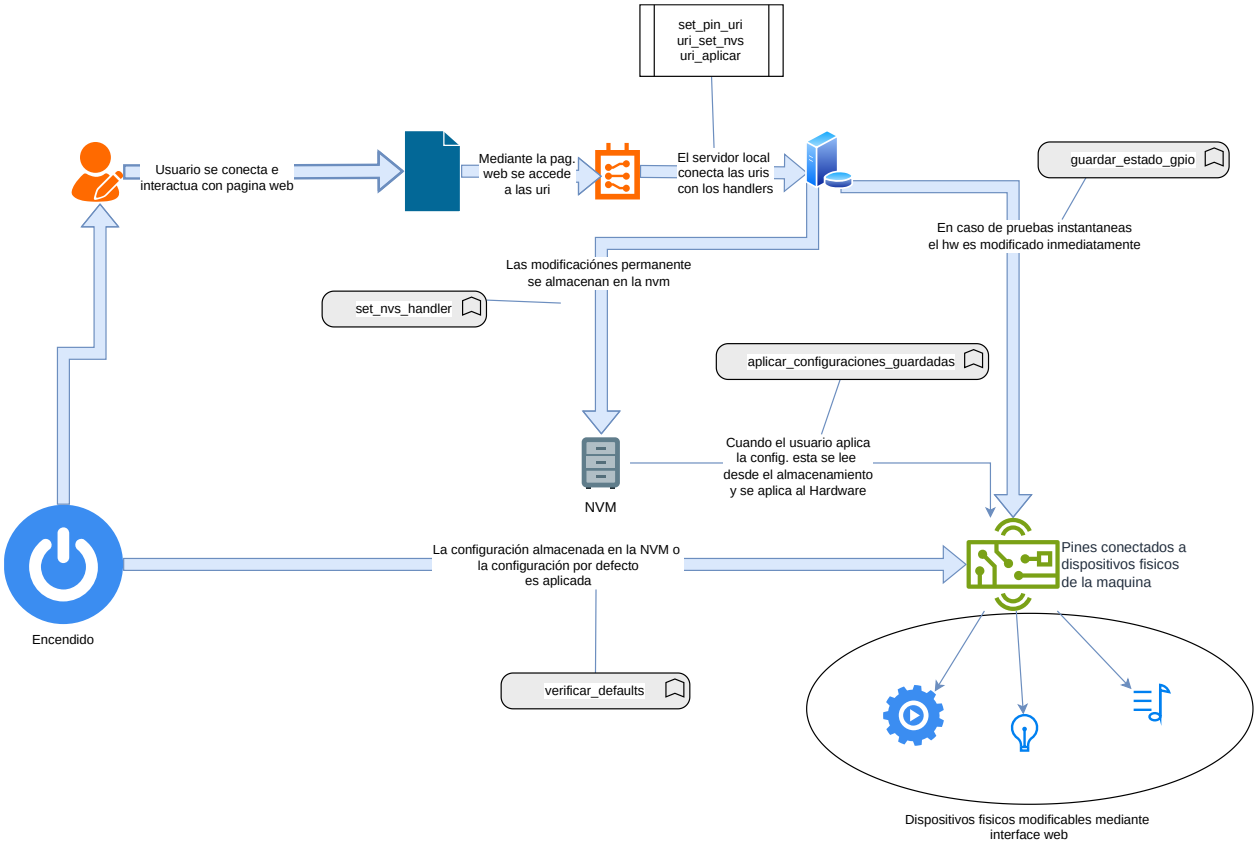
4. Se generan las siguientes tres funciones (los cuerpos completos están en el código)
    1. `void verificar_defaults(const pin_config_T *defaults, int cantidad);`
       Verifica que todas las opciones de configuración existan en la nvm. Si alguna no existe, la crea y le asigna el valor default
    2. `void aplicar_configuraciones_guardadas(const pin_config_T *defaults, int cantidad);`
       Aplica las configuraciones guardadas en la nvm a los pines físicos del dispositivo. Esto se hace en el encendido o modificación de configuraciones.
    3. `void guardar_estado_gpio(int pin, int value);`
       Asigna un estado específico a un gpio específico ignorando lo que esté guardado de manera permanente. El propósito de esta función es probar funcionalidades de la máquina sin necesidad de afectar su configuración
5. Como la interface va a ser web, se generan handlers para conectar el servidor web con las funciones internas del microcontrolador:
    1. `esp_err_t set_pin_handler(httpd_req_t *req)`
       Handler que simplemente llama a `guardar_estado_gpio` para almacenar un estado
    2. `esp_err_t set_nvs_handler(httpd_req_t *req)`
       Este handler tiene dentro la función que almacena una `key` y su respectivo `value` en la memoria persistente. No afecta la configuración, solo la memoria
    3. `esp_err_t aplicar_config_handler(httpd_req_t *req)`
       Aplica la configuración almacenada en la nvm mediante el llamado a `aplicar_configuraciones_guardadas`
6. Definidos los handlers, se definen las uri (del tipo `httpd_uri_t`) que van a vincular las acciones realizadas en la web con los handlers y funciones definidas previamente. Estas son: set_pin_uri (vincula a set_pin_handler), uri_set_nvs (vincula a set_nvs_handler) y uri_aplicar (vincula a

aplicar_config_handler) y luego se registran dentro de la función `start_webserver` para que tomen efecto

7. Definidas las funciones y las conexiones entre las funciones y la web, se diseña la página web (root.html) la cual es iniciada por defecto por el webserver. Ahora todo el tráfico a que pase por la red wifi emitida por el microcontrolador es redirigido a esta página mediante la cual se configura el dispositivo

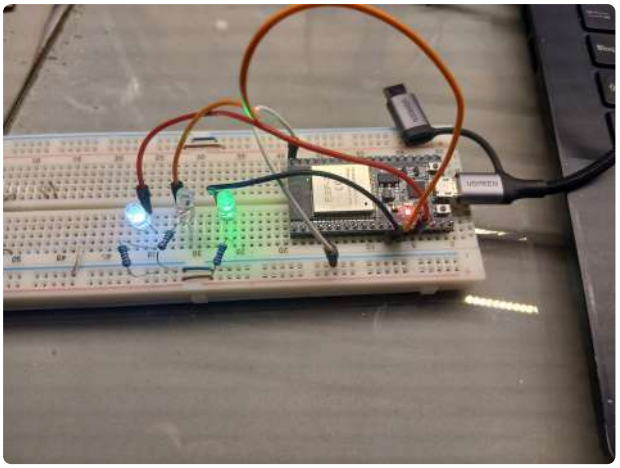
Diagrama de flujo de el uso

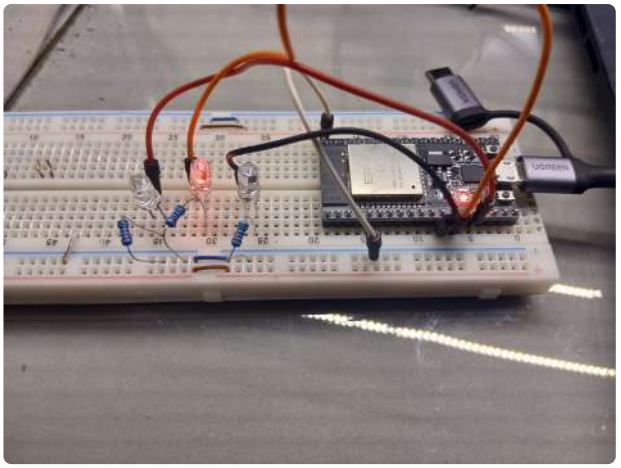
En el siguiente diagrama se muestra el uso normal del aparato. Notese que tras el encendido el dispositivo carga la configuración almacenada (o las defecto si no hay nada almacenado) y solo cuando el usuario se conecta se hacen modificaciones a la configuración



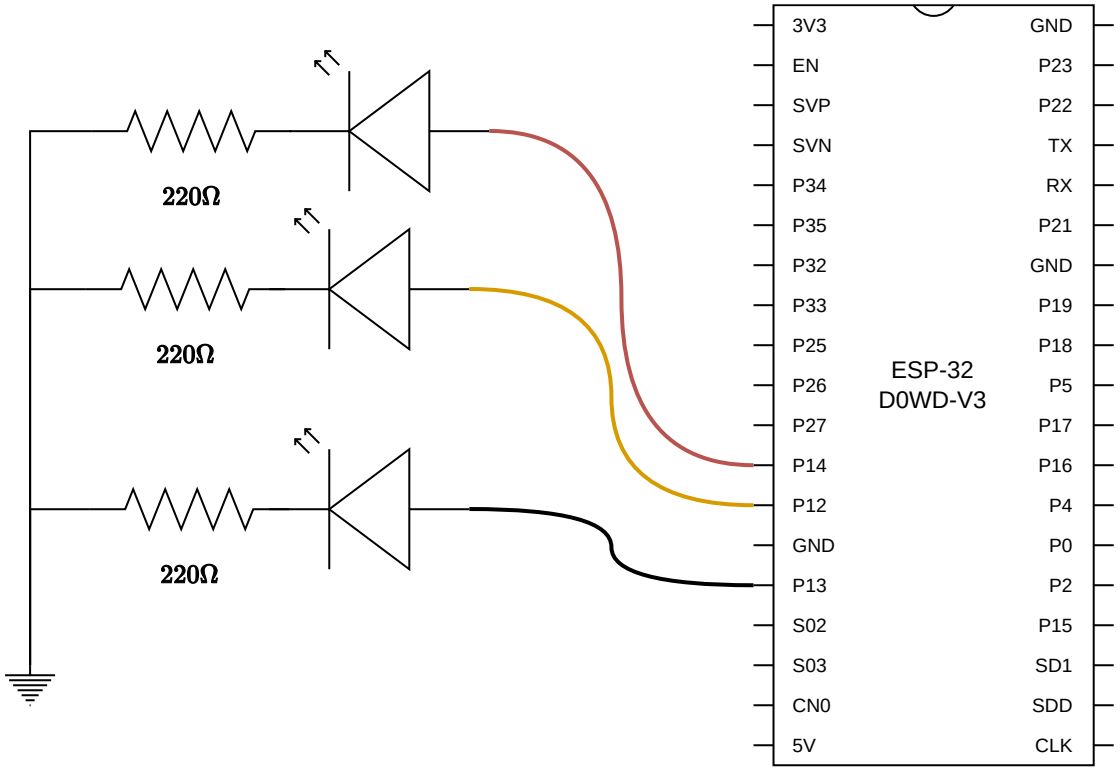
Ejemplos de uso

Al tratarse de una interface web interactiva que almacena información de manera permanente, se hace difícil mostrar con imágenes su funcionamiento. Estas son dos fotos de la placa de desarrollo en un protoboard con dos configuraciones diferentes aplicadas. Cada led representa una configuración física a aplicar a una máquina real





Luego el diagrama de conexiones para probar este mismo ejemplo con una placa ESP32 de 38 pines



La totalidad del código y este informe se puede encontrar en el github del proyecto: https://github.com/SimonAulet/Configurador-ACySE