
# Laboratorio 2

## Diseño IOT y Sistemas Embebidos

DIEGO BERGARA
CARLOS GONZALEZ
FELIPE CABRERA

DOCENTES:

PABLO ALONSO
AGUSTÍN DERRIGIBUS

# Primera Parte

## 2.a

- Volume up: Sube el volumen del parlante conectado a la placa.
- Volume down: Baja el volumen del parlante conectado a la placa.
- Photo: Sirve para sacar una foto con la cámara que se puede conectar a la placa.
- Play/Pause: Reproduce o pausa audio usando el parlante conectado a la placa.
- Record: Empieza a grabar audio usando algún micrófono conectado a la placa.
- Network: Permite a la placa conectarse a una red.

Todos estos botones son programables para cambiar su función.

## 2.b

Hay un conector tanto en la placa base como en la de extensión touch, al conectarlas usando la lámina metálica que viene con el touchpad ya debería ser posible el uso del mismo. La placa que utilizamos viene con dos placas de extensión ya integradas, y por lo tanto hay algunos “known issues” de compatibilidad de las mismas con el touchpad, limitando así nuestro posible uso de botones, como el de foto y el de network. Por lo tanto decidimos desarmar la placa para no tener problemas de compatibilidad, siendo así el touchpad la única placa de extensión.

## 2.c

Pseudocódigo:

Constante TOUCH_BUTTON_NUM como 7

button [
  - Botón 1: Volumen arriba
  - Botón 2: Reproducir/Pausar
  - Botón 3: Volumen abajo
  - Botón 4: Sensor de seguridad
  - Botón 5: Grabar
  - Botón 6: Tomar foto
  - Botón 7: Conexión de red ]

read_task()
  touch_values = tamaño TOUCH_BUTTON_NUM
  last_button = -1

```
    while(true):
        Para cada botón i en el arreglo button:
            touch_values[i] = botón[i]
            Si touch_values[i] >30000:
                last_button = i
            Fin Si

        Esperar 100 milisegundos

Definir procedimiento principal app_main():
    Inicializar el sensor táctil

    Para cada botón i en el arreglo button:
        Configurar el botón[i] del sensor táctil

    Configurar la configuración de reducción de ruido del sensor táctil

    Habilitar la reducción de ruido del sensor táctil

    Configurar el modo de funcionamiento del sensor táctil en modo de temporizador

    Iniciar el funcionamiento del sensor táctil

    Crear una tarea llamada "read_task" que ejecute el procedimiento read_task
```

# Segunda Parte

## 1.b

ESP-NETIF provee una capa de abstracción para la aplicación por encima del stack de TCP/IP, permitiendo así que las aplicaciones elijan entre stacks de IPs. También sus APIs tienen seguridad en hilos, aun si el stack de TCP/IP no tiene. La función esp_netif_init() es necesaria para su uso ya que con la misma lo inicializamos. Si vamos a iniciar una estación wifi esto va a ser necesario para facilitar la comunicación con la red y asegurar que sea seguro.

## 1.c

La librería event loop permite que componentes declaren eventos en los cuales otros componentes pueden registrar código de handler, el cual se va a ejecutar cuando esos eventos ocurran. Esto permite que distintos componentes adhieran su comportamiento deseado a cambios de estado de otros componentes sin involucrar a la aplicación.

Un uso común es si una librería de alto nivel usa la librería de wifi puede suscribirse a eventos producidos por el subsistema de wifi directamente y actuar sobre esos eventos. El event loop por defecto es un tipo especial usado para eventos de sistemas, por ejemplo los de wifi. El handle de este tipo de loop está oculto al usuario. La función esp_event_loop_create_default() crea un event loop de tipo default. Considerando que suelen ser usados para eventos de wifi nos es útil al implementar una estación wifi.

## 2.a

- Modo estación, sería el modo STA o cliente de wifi, la ESP32-S2 se conecta al AP
- Modo AP, en el cual las estaciones se conectan al ESP32-S2 ya que este se comporta como AP
- Modo combinado AP-STA, la ESP32-S2 está funcionando como un AP y al mismo tiempo es una estación conectada a otro AP.
- Distintos modos de seguridad (WPA, WPA2, WEP, etc)
- Escaneando por APs, tanto activa como pasivamente
- Modo promiscuo para monitorear paquetes wifi del protocolo IEEE 802.11

## 2.b

Pseudocódigo:

```
createAp()
   // Inicializar el subsistema de almacenamiento no volátil (NVS)
   esp_err_t ret = nvs_flash_init()

   // Verificar si no hay suficientes páginas libres en NVS o se encontró una nueva versión
   Si ret es igual a ESP_ERR_NVS_NO_FREE_PAGES o ret es igual a
ESP_ERR_NVS_NEW_VERSION_FOUND entonces
      // Borrar NVS y volver a inicializarlo
      ESP_ERROR_CHECK(nvs_flash_erase())
      ret = nvs_flash_init()
   Fin Si

   // Verificar y manejar cualquier error durante la inicialización de NVS
   ESP_ERROR_CHECK(ret)

   // Registrar un mensaje de registro informativo indicando el modo de Wi-Fi como modo
punto de acceso (AP)
   ESP_LOGI(TAG, "ESP_WIFI_MODE_AP")

   // Llamar al procedimiento wifi_init_softap() para inicializar el punto de acceso (AP) Wi-Fi
   wifi_init_softap()
```

## 2.c

Dirección MAC,nombre del dispositivo,dirección IP y tipo del dispositivo

## 2.d

Dependiendo de si se configura con o sin contraseña es necesario ingresarla o no al conectar otro dispositivo. Al hacer poner SSID oculto, ya no es inmediatamente visible la red generada como lo es cuando no está oculto.

## 3.a

Pseudocódigo:

```
Procedimiento createSTA()
    // Inicializar el subsistema de almacenamiento no volátil (NVS)
    esp_err_t ret = nvs_flash_init()

    // Verificar si no hay suficientes páginas libres en NVS o se encontró una nueva versión
    Si ret es igual a ESP_ERR_NVS_NO_FREE_PAGES o ret es igual a
ESP_ERR_NVS_NEW_VERSION_FOUND entonces
        // Borrar NVS y volver a inicializarlo
        ESP_ERROR_CHECK(nvs_flash_erase())
        ret = nvs_flash_init()
    Fin Si

    // Verificar y manejar cualquier error durante la inicialización de NVS
    ESP_ERROR_CHECK(ret)

    // Registrar un mensaje de registro informativo indicando el modo de Wi-Fi como modo
estación (STA)
    ESP_LOGI(TAG_STA, "ESP_WIFI_MODE_STA")

    // Llamar al procedimiento wifi_init_sta() para inicializar el modo estación (STA) Wi-Fi
    wifi_init_sta()
```

## 3.b

La dirección IP, el SSID, el estado de conexión, la fuerza de la señal, la velocidad de la conexión y direccion de MAC

## 3.c

Al poner con contraseña el STA de la placa puede acceder a todos los modos de seguridad, incluidos WEP/WPA, sin contraseña este no es el caso.

# Tercera Parte

## 1.a

Breve explicación de lo que haremos:

Procedimiento server_init()
    // Inicializa el servidor web embebido
    // Inicia el servidor web llamando a la función start_webserver

Función start_webserver() retorna httpd_handle_t
    // Inicia el servidor web con la configuración predeterminada
    // Registra los controladores de URI para "/hello", "/echo" y "/index"
    // Retorna el identificador del servidor web

Función http_auth_basic(const char *username, const char *password)
    // Crea una cadena con el formato "username:password"
    // Codifica la cadena en formato base64
    // Retorna la cadena codificada en base64

Procedimiento basic_auth_get_handler(httpd_req_t *req)
    // Maneja las solicitudes GET para la autenticación básica
    // Obtiene y verifica la información de autorización en la cabecera "Authorization"
    // Si la autenticación es exitosa, envía una respuesta 200 OK con los datos del usuario
    // Si la autenticación falla, envía una respuesta 401 UNAUTHORIZED

Procedimiento hello_get_handler(httpd_req_t *req)
    // Maneja las solicitudes GET para "/hello"
    // Obtiene y muestra los valores de las cabeceras "Host", "Test-Header-2" y "Test-Header-1"
    // Obtiene y muestra los parámetros de consulta de la URL
    // Envía una respuesta con encabezados personalizados y el cuerpo definido en el contexto de usuario

Procedimiento home_get_handler(httpd_req_t *req)
    // Maneja las solicitudes GET para "/index"
    // Envía el contenido de un archivo HTML como respuesta

Procedimiento echo_post_handler(httpd_req_t *req)
    // Maneja las solicitudes POST para "/echo"
    // Lee los datos recibidos en la solicitud y los muestra en la consola
    // Envía de vuelta los mismos datos como respuesta

Función start_webserver() retorna httpd_handle_t
    // Inicia el servidor web con la configuración predeterminada
    // Registra los controladores de URI para "/hello", "/echo" y "/index"

// Retorna el identificador del servidor web

Procedimiento connect_handler(void* arg, esp_event_base_t event_base, int32_t event_id, void* event_data)
// Maneja eventos de conexión
// Inicia el servidor web cuando se establece una conexión

## 1.e

Datos de sensores como el touchpad,registro de posibles eventos en el servidor, configuraciones que puedan llegar a ser implementadas y tal vez los datos de los usuarios que se conecten a dicho servidor.