Universidad Católica del Uruguay

Proyecto Final

3 de Julio de 2025

Diseño de IoT y Sistemas Embebidos

Integrantes del equipo:

Gonzalo Soto

Jerónimo Ventos

Santiago El Ters

Contenido

Introducción

Como cierre del curso Diseño de IOT y sistemas embebidos se nos pidió realizar un programa, que permita al dispositivo ESP-IDF Kaluga reproducir una playlist de canciones con el parlante integrado ESP-LyraT-8311A v1.2. Dicha playlist, debe ser manipulable por un sitio web o un servidor Mqtt, donde también se permita el control del dispositivo reproductor (Pausar/reproducir audio, subir/bajar volumen, pedir/saltar un audio de la lista de reproducción).

Desarrollo

Preparación

Para realizar este trabajo de forma óptima usamos de base parte de los laboratorios que se desarrollaron previamente en el curso. Del Laboratorio 2 se sacaron una serie de funcionalidades que facilitan el armado del proyecto, ya que en este ya teníamos un librerías reutilizables en el proyecto actual, como por ejemplo, librería para levantar un servidor y página web. También introducimos partes del laboratorio 3 para poder implementar el encendido de la luz led del kit para que parpadee cuando se conecte a internet, y el uso del touchpad para controlar el reproductor. Con esas tareas ya solucionadas nos dividimos lo que quedaba de esta manera.

- Módulo: **audio_embebido**
  Prioridad: Alta
  Objetivo principal: Reproducir canciones desde memoria usando ES8311 + I2S
  Tareas claves: Inicializar ES8311, configurar I2S, leer datos de audio, implementar comandos
  Depende de: Ninguno
  Responsable: Gonzalo

- Módulo: **mqtt_embebido**
  Prioridad: Media
  Objetivo principal: Control remoto vía MQTT y envío de logs
  Tareas claves: Conectar al broker, suscribirse a tópicos, enviar logs en JSON con QoS 1
  Depende de: wifi_embebido, event_logger, config_embebido
  Responsable: Gonzalo

- Módulo: **servidor_embebido**
  Prioridad: Alta
  Objetivo principal: Servir una web para configurar red, MQTT y controlar comandos
  Tareas claves: Iniciar servidor HTTP, levantar página web, manejar endpoints de red, MQTT, control, canciones, etc.
  Depende de: wifi_embebido, config_storage
  Responsable: Gonzalo

- Módulo: **config_embebido**
  Prioridad: Alta

Objetivo principal: Guardar y recuperar configuración desde NVS
Tareas claves: Leer y escribir SSID, datos de MQTT y canciones; manejar persistencia y reinicio.
Depende de: Ninguno
Responsable: Gonzalo

- Módulo: **tiempo_embebido**
  Prioridad: Alta
  Objetivo principal: Sincronizar hora del sistema con NTP
  Tareas claves: Configurar NTP, obtener timestamp actual
  Depende de: wifi_embebido
  Responsable: Santiago

- Módulo: **event_logger**
  Prioridad: Media
  Objetivo principal: Registrar eventos (play, pausa, etc.) con timestamp en NVS
  Tareas claves: Implementar buffer circular, guardar eventos, leer en orden cronológico.
  Depende de: ntp_sync, config_storage
  Responsable: Santiago

- Módulo: **queue_embebido**
  Prioridad: Alta
  Objetivo principal: Coordinar comandos de múltiples fuentes y encolarlos.
  Tareas claves: Crear queue, tarea consumidora, manipulación de comandos.
  Depende de: audio_embebido
  Responsable: Jerónimo

Librerías creadas para el proyecto

Audio Embebido

Es la función principal para todo lo que concierne a la manipulación del audio y configuración de la extensión de audio ESP-LyraT-8311A v1.2 del proyecto. Esta librería incializa las conexiones del parlante (Librería driver/i2c/gpio/i2c master) para que funcione con la placa, a la par que maneja la todas las acciones de la placa en lo que reproducción del audio refiere. Define el máximo/mínimo de volumen, de 100 a 0 con saltos de 10, y el máximo de canciones en cola que en este caso es 5.

Usan un semáforo mutex para controlar cuando pasa y reproduce, o mismo para la cola de audios, utilizan un mutex para controlar qué archivo está corriendo, de esta manera no se rompe ninguna función por intentar hacer la acción contraria a la vez.

Una de las principales complicaciones fue encontrar los pines adecuados en la placa para generar una salida de audio correcta y también por primera vez en el curso se hizo uso del archivo idf_component.yml ya que la es8311 no está por defecto en el esp y se tiene que

importar indicandolo en ese archivo. Una vez al compilar crea un directorio managed_components con todo lo necesario para su uso.

MQTT Embebido

Primero se definen las variables del broker, Id del usuario y tópico al que se suscribe, se crea el controlador de eventos, e intenta conectar con el broker.

Si no está conectado te suscribe automáticamente al topic definido junto con la id definida (las cuales son dadas por servidor_embedido).

Si el tópico al que está suscrito recibe un mensaje lo lee mediante el handler y lo pasa al JSON y si este contiene el formato adecuado, el cual es (“acción” + “Comando esperado”) lo transfiere al servidor_embedido para ser procesado.

Por temas de orden, posee tags de conexión/desconexión y mensaje recibido, en la terminal.

Queue Embebido

Este código es el intermediario entre el broker, el html y el touchpad traduce los comandos que recibe de las tres posibles fuentes a un comando estandarizado para manejar el reproductor. Permite un máximo de 10 órdenes en cola, una onceava no será leída por el kit hasta que se haya cumplido una anterior (para evitar saturar la memoria ram).

Cada vez que se inicializa imprime cuantos comandos quedan en la cola. Cada comando produce un tag al ejecutarse describiendo su acción y otro al sumarse una acción nueva.

Para funcionar se armó una estructura music_command con todas las posibles acciones del reproductor de audio embebido, primero define las órdenes en forma de string que ingresar el usuario y los pasa a variables para luego pasar esas variables a acciones de audio_embebido en una task que se repite infinitamente.

En un inicio no teníamos planeado el uso de esta task para pasar la información tanto el mqtt como el servidor de la página web se comunican directo con el reproductor.

Tiempo Embebido

Su función principal es sincronizar la hora del sistema utilizando el protocolo NTP (Network Time Protocol), conectándose a un servidor de tiempo en Internet (por defecto, "pool.ntp.org"). El módulo configura automáticamente la zona horaria para Uruguay y espera a que la sincronización se complete, con un tiempo máximo de espera configurable.

También provee funciones para verificar si la hora fue correctamente sincronizada, mostrar la hora actual por consola y obtener una marca de tiempo legible para su uso en registros o logs. Este sistema permite que el dispositivo mantenga un registro confiable del tiempo.

Event Logger

Guarda cada evento en un buffer circular y asegurando su persistencia mediante la NVS. el logger y recuperar el historial guardado en NVS, registrar nuevas acciones junto con su tiempo de ejecución y el nombre de la canción asociada, y mostrar todos los eventos almacenados a través de la consola. Cada vez que se registra un evento, este se agrega al buffer en la posición actual (sobrescribiendo los más antiguos cuando el buffer está lleno), y se guarda inmediatamente el buffer completo y el índice actual en NVS para asegurar que los eventos persistan tras reinicios o cortes de energía. El módulo también incluye una función para imprimir todos los eventos registrados, mostrando el número, la acción, la canción y la hora correspondiente.

Spiffs Embebido

El propósito principal de esta librería es inicializar y montar la partición de almacenamiento spiff permitiendo así al sistema guardar y acceder a archivos en la memoria flash del kit. El código configura los parámetros del sistema de archivos, como la cantidad máxima de archivos abiertos simultáneamente y la ruta base, e intenta registrar spiffs usando la función de la biblioteca de Espressif. Si el montaje falla, muestra un mensaje de error en el log; si tiene éxito, consulta y reporta la cantidad total y utilizada de espacio en la partición.

Para este módulo fue necesario crear un archivo partitions.csv con toda la información que tenia que levantar el SPIFFS del proyecto, el formato del archivo es el siguiente:
Name, Type, SubType, Offset, Size.
Además, con el menuconfig se extendió la capacidad a 4 Mb para poder almacenar correctamente todas las configuraciones necesarias de la flash.

Servidor Embebido

El host de la página web que maneja los comandos corre directamente en el kit La idea es que puedas manejar todo el dispositivo de forma remota, desde configurar el WiFi y el MQTT, hasta subir y borrar canciones en formato PCM (Formato de audio de mas comprimido que mp3) que quedan guardadas en la memoria flash. Cada función del servidor está conectada a una dirección web específica, por ejemplo, hay una ruta para la configuración de WiFi, otra para subir canciones, otra para listar archivos, y así. Antes de guardar algo nuevo, el sistema chequea que haya espacio en la memoria, y si cambias los datos de WiFi, se conecta automáticamente. Lo mismo si cambias el servidor MQTT: podés hacerlo sin tener que reiniciar todo el dispositivo.

Por último, hay funciones para prender y apagar el servidor web cuando sea necesario, asegurándose de que nunca haya dos instancias corriendo al mismo tiempo.

Se decidió que las canciones no tengan un tamaño superior a 200kb porque es el tamaño que consideramos permite una relación calidad/peso con el limitado recurso que tenemos en el kaluga, ya además que en el SPIFFS se configuró 1 Mb de espacio para las canciones, y como se prevé capacidad de 5 canciones, 200kb máximo para cada una de ellas.

Config Embebido

En este módulo se hace uso de la memoria no volátil ya que permite guardar y recuperar partes fundamentales del funcionamiento, como claves de WiFi, dirección y puerto de servidor MQTT, así como el estado de la canción actual, en que posición se encuentra y como lo identifica el sistema (song0, song1, etc..). Proporciona funciones para almacenar la configuración (config_guardar) y para cargarla (config_cargar), asegurando que el dispositivo conserve estos valores incluso después de un reinicio o apagado.

Librerías reutilizadas y ampliadas

Wifi Embebido

Una librería creada para el laboratorio 2 que permite al kit KALUGA funcionar como AP (punto de acceso) como STA (estación), y es necesaria para que el dispositivo se conecte a Internet y a su vez sea capaz de conectar varios otros dispositivos a sí mismo mediante la red, lo cual es útil a la hora de mantener el servidor en línea.

Por otro lado, se mejoró la capacidad de este módulo al fallo de errores, por ejemplo, si ya existía una conexión sta, primero se baja la actual, se libera el espacio y se levanta la nueva. Anteriormente no se tenía en cuenta este tipo de funcionamiento ya que el programa en cuestión no lo requería, pero como el actual tiene la capacidad de cambiar dichas credenciales durante su funcionamiento es clave poder reforzar este aspecto.

Led Strip y Led Embebido

Led strip es una librería que fue otorgada por parte de los profesores del curso, para no tener que programar el iniciado de la led desde 0. Mientras Led_embedido forma parte del tercer laboratorio y consta de una serie de tareas que operan con freeRTOS y se encargan de hacer parpadear una luz de un color rojo, verde o azul y cambiar entre los mismos.

Esta librería fue implementada en el servidor embebido con el fin de que cuando se reproduce un audio parpadee una luz verde, cuando Pause azul y cuando no haya más en la cola en rojo.

Touch Embebido

Esta librería se encarga de conectar el equipo ESP-IDF con un panel táctil que viene con el kit kaluga, ya se había usado esta librería en conjunto con la de led embebido para cambiar los colores de la led parpadeante.

Sin embargo en este caso se utilizó la librería de audio embebido para tener un control analógico sobre el parlante además de los comandos por Wi-fi.

Funcionamiento del Main

Primero monta los spiffs para poder usar la memoria flash del dispositivo y luego llama a la función de audio_embedido_iniciar(). Recurre a la memoria no volátil para cargar la configuración del reproductor y que todas las claves de Wifi y el broker de mqtt estén con las variables correctas. En caso de que no, se pueden alterar desde el propio main. Inicia la conexión a internet, carga el estado del audio, inicia el tiempo y por último crea una task del queue_embedido: music_player_task para leer los parámetros que le lleguen.

Conclusiones

Montar este proyecto requirió de todos los conocimientos previos dados en el curso. Se necesitaba un manejo íntegro de estos puesto que la falla de cualquiera de los componentes podía tirar el proyecto abajo, por lo interrelacionado que están todas las funciones del reproductor y se complejiza aún más con lo grande que terminó siendo la biblioteca de funciones que se acabó utilizando.