LABORATORIO FINAL IoT

11 DE JULIO DE 2025

MARCOS GIOMBINI
GASTON MASSAFERRO
FERNANDO LARRICA

Contenido

1. Introducción

El presente informe documenta la implementación y validación funcional del laboratorio final del curso de IoT. La solución desarrollada integra múltiples tecnologías clave: control de reproducción de audio en una placa ESP32-S2 Kaluga, control de audio vía touchpad y MQTT, conectividad WiFi dinámica en modo AP+STA, interfaz web responsiva, persistencia de configuración vía NVS, sincronización horaria por NTP y publicación de eventos vía MQTT. Todo el sistema está basado en FreeRTOS y emplea una arquitectura modular orientada a tareas.

Este documento se estructura en torno al cumplimiento de los requisitos de la letra del laboratorio, identificando las estrategias de implementación, problemas enfrentados y soluciones adoptadas.

2. Implementación por funcionalidades

2.1 Reproductor de Audio

La funcionalidad principal del proyecto es la reproducción de archivos de audio. Para ello se utilizó el códec ES8311, conectado a la placa Kaluga a través de las interfaces I2C e I2S. El módulo encargado de gestionar la reproducción se organiza como una tarea independiente dentro del sistema operativo FreeRTOS, que opera sobre una cola de comandos compartida.

Esta cola recibe instrucciones desde diferentes fuentes: los botones táctiles, la interfaz web y el protocolo MQTT. Cada comando (PLAY, STOP, PAUSE, NEXT, etc.) se interpreta y ejecuta de forma secuencial, permitiendo que múltiples fuentes de entrada interactúen con el reproductor sin interferencias. La arquitectura orientada a tareas, junto con el uso de semáforos y colas, permitió una sincronización efectiva y segura entre los distintos módulos del sistema.

Los archivos .wav son cargados a SPIFFS desde la web, gestionados en una playlist que puede ser actualizada dinámicamente.

2.2 Control por Touchpad

El sistema incluye un panel táctil capacitivo implementado con siete canales disponibles en la placa ESP32-S2 Kaluga. Cada canal fue asignado a una acción específica: reproducción/pausa, detener, siguiente, anterior, subir volumen y bajar volumen. Para garantizar una detección precisa, se aplicó un proceso de calibración inicial que determina el valor de referencia de cada canal en reposo.

Este valor de referencia se actualiza de manera dinámica y se combinó con filtros IIR y técnicas de eliminación de ruido (denoise), lo que permitió obtener una lectura estable incluso en entornos eléctricos ruidosos. El control táctil resultó ser una alternativa eficiente y confiable para operar el sistema sin necesidad de acceso remoto.

2.3 Interfaz Web

La interfaz web fue desarrollada en el archivo index.html, cargado directamente desde el sistema de archivos SPIFFS de la placa, permite a los usuarios controlar el reproductor, subir nuevos archivos, descargarlos y borrar desde la lista de audio, configurar la red WiFi y definir el broker MQTT.

Además de los controles básicos, la interfaz permite borrar credenciales almacenadas y reiniciar el sistema desde el navegador. Se implementaron mecanismos para evitar el envío duplicado de comandos, así como validaciones en los formularios para garantizar la integridad de los datos ingresados. La página también consulta periódicamente el estado actual del sistema, ofreciendo una visualización en tiempo real del funcionamiento del reproductor.

2.4 WiFi AP + STA con FSM

La conectividad del dispositivo fue abordada mediante una máquina de estados finitos (FSM), que permite gestionar de manera estructurada y reactiva las distintas fases de conexión de la placa ESP32-S2. Esta FSM representa el comportamiento del sistema en términos de estados bien definidos y transiciones disparadas por eventos externos, como la llegada de credenciales, la pérdida de señal o el éxito en la conexión.

El flujo comienza en el estado INIT, donde se realiza una inicialización básica y se verifica si existen credenciales de red almacenadas en NVS. En caso afirmativo, el sistema transita hacia el estado CONNECTING, donde se inicia el intento de conexión en modo estación (STA) utilizando los datos proporcionados. Si la conexión es exitosa, se alcanza el estado CONNECTED, habilitando la funcionalidad completa del sistema, incluyendo la activación del cliente MQTT y la sincronización horaria NTP.

Por otro lado, si no se encuentran credenciales al inicio, o si el intento de conexión falla tras varios reintentos, la FSM deriva al estado WAIT_CREDENTIALS. En este estado, el dispositivo permanece operando como punto de acceso (AP), ofreciendo una red propia a la que los usuarios pueden conectarse para configurar la red WiFi mediante la interfaz web. Una vez ingresadas y validadas las nuevas credenciales, el sistema vuelve al estado CONNECTING y reinicia el proceso.

Este enfoque mediante FSM no solo ordena el comportamiento del sistema ante eventos asincrónicos, sino que también facilita la depuración y mejora la resiliencia frente a condiciones adversas de red. El diseño modular de los estados permite agregar funcionalidades específicas a cada uno, como feedback visual con LEDs, limpieza de recursos o bloqueo de acciones incompatibles. Gracias a esta arquitectura, el sistema puede adaptarse dinámicamente a distintos entornos de conectividad sin requerir intervención manual o reinicios forzados.

Además, la FSM contempla eventos de desconexión inesperada durante la operación. En tales casos, el sistema transita automáticamente a un estado de espera activa (WAITING_USER), donde permanece en modo AP hasta que el usuario decida reconfigurar la red o reiniciar el intento de conexión. Esta capacidad de autogestión y recuperación ante fallos fue fundamental para garantizar la estabilidad del dispositivo a lo largo del uso.

2.5 NVS: Persistencia

Uno de los objetivos clave del proyecto fue lograr que el sistema pueda conservar su configuración y estado incluso tras un reinicio o corte de energía. Para ello, se utilizaron las funcionalidades de NVS (Non-Volatile Storage) del ESP32. Se definieron particiones separadas para almacenar las configuraciones de red, la dirección del broker MQTT, los archivos de audio y los eventos del logger.

Antes de cada escritura en NVS, se realiza una validación para evitar errores por corrupción o datos malformados. Esto permitió lograr un sistema capaz de recuperar automáticamente su configuración al encenderse, sin necesidad de ser reflasheado o reconfigurado por el usuario.

2.6 MQTT

El sistema de mensajería MQTT fue diseñado para activarse únicamente cuando la conexión STA sea exitosa. Una vez conectado a un broker, el sistema comienza a publicar eventos de reproducción en tiempo real, y además transmite todos los eventos registrados previamente en el logger.

Para cumplir con la confiabilidad requerida, los mensajes se envían utilizando el nivel de calidad de servicio QoS 1, que asegura que cada mensaje se entregue al menos una vez y se confirme mediante un ACK. Los datos se empaquetan en formato JSON, utilizando la biblioteca cJSON, lo que facilita su interpretación por cualquier cliente MQTT externo.

2.7 Sincronización horaria (NTP)

El sistema necesita conocer la hora exacta para registrar los eventos del logger de forma precisa. Por este motivo, se implementó la sincronización horaria mediante el protocolo NTP, utilizando como fuente el servidor pool.ntp.org. La sincronización se considera válida únicamente cuando se obtiene una fecha posterior al año 2024, para evitar registros inválidos en el arranque.

Una vez sincronizada la hora, todos los eventos del sistema incorporan un timestamp correcto, lo que resulta esencial para la trazabilidad de los datos enviados por MQTT..

2.8 Logger de eventos

El logger registra todos los eventos de reproducción relevantes: inicio, pausa, avance, retroceso y detención. Cada uno de estos eventos se guarda junto con

su marca de tiempo en un buffer circular de tamaño limitado (20 eventos), almacenado en una partición específica de NVS.

Cuando el sistema se conecta exitosamente a un broker MQTT, este buffer se recorre y cada evento es publicado en orden cronológico. Si la conexión se pierde en algún momento, los eventos continúan almacenándose y se envían una vez que la conexión se restablece. Esta mecánica garantiza la integridad y persistencia de los eventos sin importar el estado de la red.

2.9 Funcionalidades adicionales implementadas

Además de cumplir con todos los requisitos obligatorios del proyecto, se implementaron varias mejoras opcionales. Entre ellas se destaca el uso del panel táctil como método alternativo de control, la incorporación de un LED de sistema que actúa como heartbeat para indicar el estado activo del reproductor, y el reporte por MQTT de eventos asociados a cortes o reanudaciones de energía.

Estas mejoras no solo enriquecieron la funcionalidad del sistema, sino que también reforzaron su confiabilidad y su adaptabilidad a situaciones reales de operación en campo.

3. Pruebas realizadas

Prueba	Resultado	Observaciones
Reproducción de audio desde web	Éxito	Tiempo de carga aceptable, sin glitches.
Control por touchpad	Éxito	Calibración automática evita falsos positivos.
Cambio de WiFi vía web	Éxito	FSM reacciona correctamente y guarda datos.
Conexión y reconexión a broker	Éxito	Se reconecta tras pérdida de IP.
Subida de WAVs desde web	Éxito	Archivos visibles e integrados en la playlist.
Publicación de eventos por MQTT	Éxito	JSON válido y actualizado.
Borrar credenciales desde web	Éxito	FSM vuelve a modo AP correctamente.
Sincronización NTP	Éxito	Validación robusta con reintentos.

Interacción simultánea	Éxito	Web, touch y MQTT operan sin conflictos.

4. Resultados obtenidos

Al finalizar la implementación del sistema, pudimos comprobar que se cumplían todos los requisitos definidos en la consigna del laboratorio, tanto en su forma funcional como estructural. El sistema se comportó de manera estable en todas las pruebas realizadas, permitiendo la reproducción de archivos de audio desde una lista local almacenada en SPIFFS, con controles accesibles desde la web, el protocolo MQTT y los botones táctiles de la placa.

Uno de los resultados más importantes fue lograr una integración fluida entre todos los módulos del sistema, lo que permitió una experiencia de uso coherente y sin interrupciones. Las distintas fuentes de control (touchpad, interfaz web y MQTT) pudieron operar de forma simultánea sin generar conflictos, y las órdenes enviadas desde cada una fueron correctamente encoladas, procesadas y ejecutadas por la tarea encargada de la reproducción.

Otro logro destacable fue la correcta gestión del almacenamiento persistente. La configuración del sistema —incluyendo red WiFi, dirección del broker MQTT y parámetros internos— se mantuvo incluso tras múltiples reinicios o cortes de energía. Esto permitió validar la funcionalidad de las particiones NVS y el diseño de recuperación automática al encender el dispositivo.

En cuanto a la conectividad, el comportamiento de la máquina de estados (FSM) superó nuestras expectativas. El sistema fue capaz de alternar entre modos AP y STA según las condiciones del entorno, ofreciendo siempre un punto de acceso cuando no se disponía de credenciales, y conectándose automáticamente cuando las configuraciones eran válidas. Además, ante desconexiones inesperadas, el sistema reaccionó adecuadamente, permitiendo al usuario recuperar el control sin intervención externa.

También pudimos validar satisfactoriamente el envío de eventos vía MQTT. Todos los comandos ejecutados fueron registrados correctamente en el logger circular, y al establecerse una conexión con el broker, el sistema publicó la información de forma ordenada, en formato JSON y con marca temporal precisa. La sincronización horaria mediante NTP funcionó sin inconvenientes y garantizó la validez de los registros generados.

El sistema demostró ser robusto incluso ante condiciones no ideales, como desconexiones de red, fallos temporales en la comunicación con el broker o reinicios abruptos. En todos los casos, el comportamiento fue consistente con el

diseño esperado: se preservó el estado interno, se mantuvo la integridad de la lista de reproducción y se retomó la operación sin pérdida de datos.

En síntesis, el proyecto entregado no solo alcanzó todos los objetivos planteados, sino que también incorporó funcionalidades adicionales que enriquecieron el sistema, como el uso del touchpad como entrada alternativa, la inclusión de un LED tipo heartbeat y el reporte de eventos críticos. Esto consolidó una solución completa, madura y lista para su uso real en un entorno IoT.

5. Dificultades y aprendizajes

Durante el desarrollo de este proyecto nos enfrentamos a múltiples desafíos que no solo pusieron a prueba nuestra capacidad técnica, sino también nuestras habilidades de organización, colaboración y toma de decisiones.

Uno de los primeros problemas que se presentó fue la interacción entre la inicialización del módulo de audio y la configuración de la red WiFi. Descubrimos que si ambos se inicializaban en paralelo o en orden incorrecto, se generaban conflictos en los buses I2C e I2S, provocando fallos intermitentes difíciles de rastrear. Este comportamiento nos obligó a revisar cuidadosamente la secuencia de arranque del sistema y a diseñar una estructura modular y jerárquica, donde el audio se inicializa en una etapa anterior a la conexión de red. Esta decisión, aunque aparentemente menor, resolvió varios problemas de estabilidad que surgían de forma aleatoria durante las primeras pruebas.

El segundo gran aprendizaje vino del trabajo con los sensores táctiles. Al principio, los botones capacitivos respondían de forma errática y disparaban múltiples eventos por una única pulsación. Este comportamiento estaba influenciado por factores ambientales como el ruido eléctrico o la humedad. Para solucionarlo, implementamos un sistema de calibración dinámica que detecta el valor de reposo de cada canal y lo ajusta automáticamente, además de incorporar técnicas de filtrado digital como el uso de IIR y un algoritmo de denoise. Este proceso nos ayudó a comprender mejor cómo funcionan los sensores en entornos reales y cómo hacerlos confiables, incluso con recursos limitados.

También nos enfrentamos al desafío de manejar eventos asincrónicos desde múltiples fuentes —web, MQTT y touchpad— sin provocar condiciones de carrera ni bloqueos. Esto requirió una revisión profunda del uso de semáforos, colas y mutexes dentro de FreeRTOS. Fue necesario diseñar cuidadosamente las zonas críticas y las prioridades de las tareas para evitar bloqueos mutuos o pérdida de eventos. A partir de este problema, adquirimos una mayor comprensión de la programación concurrente en sistemas embebidos y de la importancia de aplicar buenas prácticas en el diseño multitarea.

Otro punto complejo fue el diseño y testeo de la máquina de estados para la conectividad WiFi. Implementar una FSM no fue simplemente definir estados y transiciones, sino comprender cómo se comporta el sistema bajo condiciones no ideales: redes que no responden, contraseñas incorrectas, pérdida de señal, etc. Esto nos llevó a incluir validaciones adicionales, timers de reintento, y retroalimentación al usuario, tanto visual como lógica, para evitar que el sistema quedara en estados inconsistentes. Al lograr que la FSM sea capaz de reconectar, esperar o reiniciarse sola según sea necesario, conseguimos un comportamiento robusto y profesional.

Finalmente, uno de los aprendizajes más valiosos fue comprender cómo diseñar una solución completa, capaz de sobrevivir a reinicios, cortes de energía y desconexiones de red, sin comprometer su integridad. Lograr persistencia mediante NVS, conservar la playlist, reanudar la operación tras reinicios, y sincronizar el tiempo con NTP fueron elementos que, más allá de lo técnico, nos obligaron a pensar en la experiencia real del usuario y en el ciclo de vida de un producto embebido en el mundo físico.

Este proyecto nos permitió integrar múltiples disciplinas —redes, almacenamiento, sensores, programación concurrente y diseño web— y entender cómo todas ellas deben coordinarse de manera sólida para lograr un sistema IoT funcional, confiable y extensible.

6. Conclusión

Este laboratorio permitió consolidar múltiples aspectos técnicos del ecosistema ESP-IDF, enfrentando desafíos reales de integración de hardware, conectividad dinámica, sincronización remota, persistencia y diseño de interfaces amigables. La solución implementada responde fielmente a la consigna y se comporta de forma robusta, incluso en condiciones no ideales. Se logró construir una arquitectura sólida, extensible y reusable, que podría ser base para futuros desarrollos reales de IoT.