Famico

Bárbaro Leonardo Domingo, Martínez Brenda Carolina, Rosano Matias
42375507, 43815223,43570765
Lunes, L2

Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

Resumen. El propósito de esta aplicación para Android es proporcionar una interfaz de usuario intuitiva para controlar y modificar en tiempo real una matriz de valores, la cual representa una secuencia musical o de control (similar a un secuenciador MIDI). La comunicación con el dispositivo hardware (un ESP32) se realiza de forma inalámbrica utilizando el protocolo MQTT, permitiendo una interacción remota y eficiente.

Palabras claves: Emulador de sonido, NES, Android, MQTT.

Introducción

La aplicación sirve como una interfaz de control remoto para un dispositivo de hardware externo (presumiblemente un microcontrolador como el ESP32) encargado de generar o interpretar secuencias musicales. Su utilidad principal es permitir al usuario visualizar, modificar y disparar secuencias de datos organizados en una matriz de 16x4, de manera intuitiva y en tiempo real, sin necesidad de conexión física.

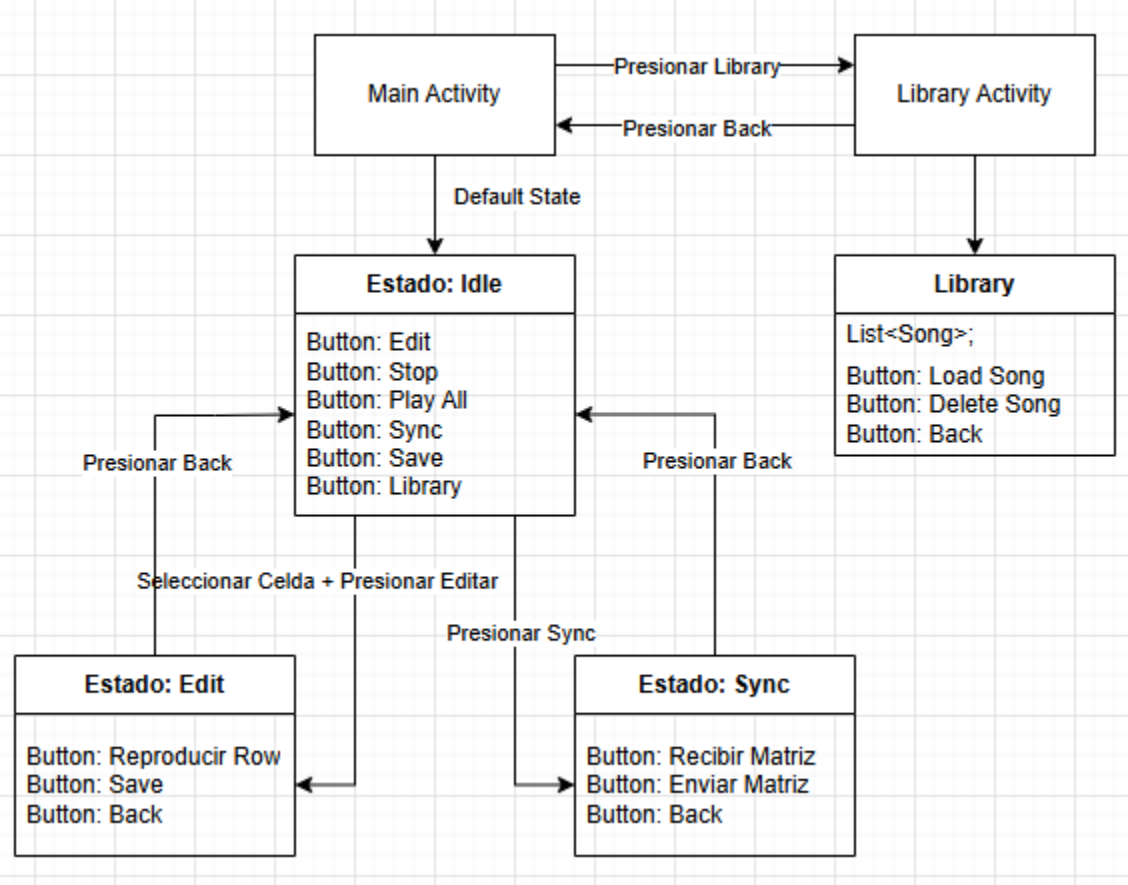
Funcionalidades Clave:

1. Control de Matriz Musical: La aplicación presenta una matriz de 16 filas por 4 columnas. Cada celda contiene un valor numérico que puede representar parámetros de una nota o evento musical (ej. nota, velocidad, duración, canal MIDI). El usuario puede ver el estado completo de la secuencia de un solo vistazo.
2. Edición Remota de Parámetros: El usuario puede seleccionar cualquier celda de la matriz y, a través de un modo de "Edición", cambiar su valor numérico directamente desde la interfaz del teléfono. Al guardar, este nuevo valor se envía al dispositivo hardware a través del protocolo de comunicación MQTT, actualizando el comportamiento de la secuencia de forma instantánea.
3. Disparo de Secuencias: La aplicación permite ejecutar la secuencia de dos maneras:
    a. Reproducción Completa ("PlayAll"): Envía un comando para que el dispositivo hardware reproduzca toda la secuencia almacenada en la matriz.
    b. Reproducción por Fila ("PlayRow"): Permite disparar únicamente la fila (paso de la secuencia) que está actualmente seleccionada, facilitando la prueba de cambios específicos.
4. Monitorización de Estado: La interfaz proporciona una retroalimentación constante al usuario, mostrando tanto el estado interno de la aplicación ("Idle" o "Editando") como los mensajes de estado que el propio dispositivo hardware envía, lo que permite saber en todo momento qué está ocurriendo en ambos extremos del sistema.

Desarrollo

Enlace al Repositorio del código Android
https://github.com/UNLAM-SOA/2025-SOA-Q2-L2

Diagrama de Funcionalidad



Manual de Usuario – FAMICO

1. Descripción General

FAMICO es una aplicación Android diseñada para **simular sonido estilo NES** mediante el control de un dispositivo embebido **ESP32**.

La app permite visualizar, editar, almacenar y reproducir una matriz musical de **16 filas × 4 columnas**, donde cada fila representa un patrón y cada columna un parámetro sonoro.

La comunicación entre la aplicación y el ESP32 se realiza a través del protocolo **MQTT**, utilizando tópicos específicos para:

- Enviar órdenes al ESP32 (play, stop, editar, sincronizar).
- Recibir estados del sistema embebido.
- Editar valores individuales de la matriz.
- Transferir/Recibir la matriz completa.

La aplicación funciona mediante **modos de operación**, que modifican qué acciones puede realizar el usuario:

- **Modo Idle** (principal)
- **Modo Edición**
- **Modo Sincronización**
- **Menú de Librería**

La aplicación incluye soporte para sensores del teléfono para funciones especiales:

- **Acelerómetro:** Detecta sacudidas ("shake") para iniciar la reproducción.
- **Giroscopio:** Permite modificar valores de celdas girando el teléfono durante la edición.

2. Requisitos del Usuario

Para utilizar FAMICO se requiere:

- Teléfono Android con conexión a Internet, acelerómetro y giróscopo
- Un ESP32 encendido y conectado al broker MQTT configurado.
- Que ambos utilicen el mismo servidor MQTT (por ejemplo, Ubidots).
- Configuración correcta de tópicos y credenciales MQTT en la aplicación.

3. Pantalla de Inicio

Al abrir la aplicación se muestran dos botones:

- **Play:** Accede al menú principal e inicia la aplicación en estado *Idle*.
- **Help:** Muestra una pantalla de ayuda con información básica de uso.

4. Pantalla Principal (Modo Idle)

Al ingresar al menú principal, la interfaz se divide en tres secciones:

A. Encabezado: Estados

- **Estado App**: muestra si la aplicación está en *Idle*, *Edit* o *Sync*.
- **Estado ESP32**: refleja el último estado publicado por el microcontrolador.
- **Último mensaje MQTT**: indica el comando o dato más reciente enviado/recibido.

B. Matriz 16×4

- Conformada por 16 filas y 4 columnas.
- Cada celda muestra un valor numérico.
- **Restricciones de valores**:
    - **Columna 0:** valores entre **0 y 15**.
    - **Columnas 1, 2 y 3:** valores entre **0 y 127**.
- Para editar una celda, esta debe primero **seleccionarse** y luego presionar **Editar**.

C. Barra Inferior – Botones en modo Idle

1. **Edit:** Cambia el estado a *Edición* para modificar una celda seleccionada.
2. **Play All:** Envía el comando PlayAll al ESP32 para reproducir toda la matriz.
3. **Stop:** Envía Idle al ESP32 para detener la reproducción.
4. **Sync:** Cambia el estado a Sincronización.
5. **Save:** Guarda la matriz actual en la librería interna y permite ingresar un nombre.
6. **Library:** Abre la pantalla de Librería con las canciones guardadas.

5. Funcionalidad por Sensores

FAMICO incorpora dos sensores del dispositivo móvil para mejorar la interacción:

5.1 Acelerómetro – Detección de Shake

La app detecta un movimiento brusco de sacudida ("shake"). Esta función **solo está disponible cuando la app está en modo Idle**.

Comportamiento:

- Si se detecta un shake:
    - La app envía automáticamente al ESP32:

topic: state
mensaje: "PlayAll"
Equivale a presionar el botón **Play All**.

Uso recomendado:

- Activar reproducción sin tocar la pantalla.
- Función útil cuando el usuario está utilizando la app como instrumento interactivo.

5.2 Giroscopio – Control de Valor en Edición

Mientras la app está en **Modo Edición**, el giroscopio se habilita para permitir la modificación del valor de la celda seleccionada mediante la orientación del teléfono.

Comportamiento:

- **Girar el teléfono a la derecha → Incrementa el valor**
- **Girar el teléfono a la izquierda → Decrementa el valor**

La app ajusta el valor dentro de los rangos permitidos según la columna:

- Columna 0 → 0 a 15
- Columna 1-3 → 0 a 127

Ventajas:

- Edición más rápida y precisa.
- No requiere teclado táctil.

6. Edición de Celdas

Cómo editar una celda

1. Tocar una celda en la matriz.
2. La app mostrará: **“Celda (X, Y) seleccionada. Presiona EDITAR.”**
3. Presionar el botón **Edit**.
4. La app entra en **Modo Edición**.
5. Modificar el valor numérico directamente.

7. Modo Edición

En este modo aparecen **tres botones**:

1. **Back:** Vuelve al modo Idle y envía al ESP32: **{topic: state, mensaje: "Idle"}**
2. **Save:** Guarda el valor de la celda editada y lo envía: **{topic: playRow, mensaje: <row column value>}**
3. **Play Row:** Reproduce únicamente la fila seleccionada. **MQTT enviado: {topic: playRow, mensaje: <número de fila>}**

8. Modo Sincronización

Este modo sirve para transferir la matriz completa entre el teléfono y el ESP32.

Botones disponibles:

1. **Back:** Regresa al modo Idle.
2. **Recibir Matriz:** La app se suscribe al tópico correspondiente para recibir la matriz del ESP32.

   ⚠ **Advertencia:**
   a. La app **no queda en espera permanente**.
   b. **Si la matriz llega cuando la app NO está en modo Sync**, se generará un error.
3. **Enviar Matriz:** Envía la matriz completa al ESP32 en texto plano, es decir se mandan todos los valores de la matriz juntos: “15 127 127 127 15 127 127 127...”.

9. Menú de Librería

La Librería permite gestionar canciones guardadas.

Contenido de la pantalla:

- Lista de canciones guardadas.
- Cada item contiene un **radio button** para seleccionar una sola canción.

Botones:

1. **Load:** Carga en la matriz la canción seleccionada.
2. **Delete:** Elimina la canción de la librería.
3. **Back:** Vuelve al menú principal.

10. Interacción MQTT

Mensajes enviados por la app

Función	Tópico	Descripción
Cambiar el estado del ESP	**topicState**	Idle / Edit / PlayAll
Enviar celda editada	**topicEdit**	En formato JSON
Reproducir una fila	**topicPlayRow**	envía número de fila
Enviar matriz completa	topicSendMatrix	depende de configuración
Recibir matriz completa	topicReceiveMatrix	depende de configuración

Mensajes recibidos del ESP32

La aplicación se suscribe al tópico:

- **TopicStatus (simulator/status)**
- **TopicReceiveMatrix (simulator/celval)**

Ejemplos de mensajes recibidos:

```
{"value": "Idle"}
{"value": "Edit"}
{"value": "PlayAll"}
{"value": "PlayRow"}
```

{“topic”: ‘simulator/celval’ "value": "15 127 127 127 15 127 127 127..."}

Estos se muestran automáticamente en *Estado ESP*.

10. Estados del Sistema

Estado de la App

- **Idle:** Modo principal, permite seleccionar celdas, reproducir, sincronizar o guardar.
- **Edit:** Solo disponible para modificar una celda.
- **Sync:** Exclusivo para transferir matrices.

Estado del ESP32

Ejemplos:

- “Idle”
- “Edit”
- “PlayAll”
- “PlayRow”

11. Errores Comunes y Soluciones

1. No se actualiza el estado del ESP32

- Verificar la conexión a Internet.
- Confirmar que el ESP32 esté conectado al mismo broker MQTT.
- Revisar credenciales MQTT.

2. No puedo editar la matriz

- Asegurarse de seleccionar una celda antes de presionar **Edit**.

3. No llega el mensaje al ESP

- Revisar el broker MQTT.
- Validar la configuración en **ConfigMQTT**.

4. Error al recibir matriz

- La aplicación debe estar en modo **Sync** durante la recepción.

12. Resumen General del Funcionamiento

1. El usuario selecciona una celda.
2. Presiona **Edit.**
3. Cambia el valor y presiona **Save** → se envía el mensaje correspondiente.
4. Puede reproducir la fila (Play Row) o toda la matriz (Play All).
5. Puede sincronizar matrices completas.
6. Puede guardar canciones y cargarlas desde la Librería.
7. El ESP32 responde mediante mensajes MQTT que la aplicación muestra en pantalla.

3. Conclusiones

Recaudos tomados para lograr tolerancia a fallos

Durante el desarrollo de la aplicación fue necesario implementar una serie de recaudos para asegurar que el sistema fuese robusto y tolerante a fallos, tanto en la comunicación con el ESP32 como en la interacción del usuario con la interfaz:

- **Validación del estado de la aplicación antes de ejecutar acciones.**
  La app solo permite editar una celda cuando se seleccionó correctamente una posición y se ingresó al modo *Editando*. Esto evita inconsistencias o envíos erróneos de datos.
- **Control de estados del sistema embebido.**
  Se incorporó la visualización del *Estado ESP* recibido via MQTT, permitiendo detectar rápidamente desconexiones, demoras o mensajes inesperados.
- **Manejo controlado de MQTT.**
  Se implementaron try/catch en cada publicación para evitar que caídas o demoras del broker provoquen cierres de la aplicación.
  Además, se define un comportamiento seguro si la conexión no está disponible.
- **Prevención de acciones inválidas.**
  La app no permite pulsar “Play Row”, “Guardar” o “Editar” cuando no corresponde, evitando errores lógicos y protegiendo al microcontrolador de recibir comandos incompletos.

- **Sincronización estricta entre App y ESP32.**
  Cada cambio de modo (Idle / Edit) se publica al microcontrolador evitando desfasajes entre la interfaz y el estado real del sistema.

Problemas encontrados y soluciones aplicadas

Durante el desarrollo surgieron diversos inconvenientes que debieron resolverse:

- **Sincronización incorrecta entre la selección de celdas y el modo de edición.**
  → *Solución:* Se creó la variable selectedRow y selectedCol y se actualiza solo al hacer click en la celda; después se valida antes de editar.
- **El ESP32 no recibía correctamente la edición de la matriz.**
  → *Solución:* Se envió el JSON en un formato estandarizado (row, col, value) usando JSONObject, evitando errores de parseo.
- **Pérdidas de conexión temporal al broker MQTT.**
  → *Solución:* Se agregó control de reconexión y manejo suave de errores para evitar que la app se cierre.
- **Confusión del usuario al cambiar entre modos.**
  → *Solución:* Se agregaron indicadores visuales ("APP STATE: IDLE/EDITING") para dejar claro qué acciones están disponibles.
- **Problemas iniciales con el GridView al mostrar valores incorrectos.**
  → *Solución:* Se convirtió la matriz a una estructura ArrayList<String> y se usa un adaptador personalizado que refresca de forma estable.

Referencias

1. — SOA-UNLaM: "PUBLICO: Android", disponible en: https://www.soa-unlam.com.ar/wiki/index.php/PUBLICO:Android