

# 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.

Se presenta el diseño e implementación de un secuenciador/emulador de sonido basado en ESP32, con interfaz gráfica en pantalla TFT ILI9341, controlado mediante dos encoders rotativos y comunicación MQTT hacia una aplicación Android. El sistema replica la lógica de generación de patrones estilo NES sobre una matriz de pasos de  $16 \times 4$ , donde cada fila representa un step y cada columna un canal sonoro: percusión/ruido NES y tres voces tonales (dos osciladores cuadrados y un oscilador triangular). Cada celda almacena un valor decimal que se interpreta como preset de percusión o nota MIDI, permitiendo construir patrones rítmico-melódicos.

La interfaz permite navegación y edición en tiempo real desde la pantalla y encoders, además de edición remota vía MQTT. La reproducción se implementa mediante síntesis digital en el ESP32 y salida de audio real por I<sup>2</sup>S hacia un módulo MAX98357A, a 44,1 kHz. La arquitectura separa captura de eventos de usuario (tarea FreeRTOS de entrada), comunicación MQTT (tarea dedicada) y la máquina de estados principal que corre en el loop junto al motor de audio, evitando bloqueos e integrando control en tiempo real del tempo en BPM. Se describe la máquina de estados (IDLE, EDIT, PLAY\_ALL, PLAY\_LINE), el manual de usuario y la integración con Android. Finalmente, se discuten extensiones posibles sobre el motor de síntesis y la interfaz.

Palabras claves: Emulador de sonido, NES, grid de patrones, FreeRTOS, ILI9341, I<sup>2</sup>S, MQTT

## Introducción

Los videojuegos clásicos (por ejemplo, la NES) generaban sonido mediante canales sencillos (square, triangle, noise, etc.) donde cada "paso" de una secuencia define parámetros de la onda. Este trabajo propone un prototipo embebido que simula esa idea: una matriz de pasos ( $rows \times cols$ ) donde cada celda contiene parámetros hexadecimales que representan la configuración sonora de ese paso. La interfaz permite editar los parámetros, preEscuchar la celda y reproducir secuencias completas con tempo variable. El objetivo educativo es entender la organización de datos para síntesis por pasos, la programación de interfaces en microcontrolador y la coordinación de tareas (UI vs reproducción) usando FreeRTOS.

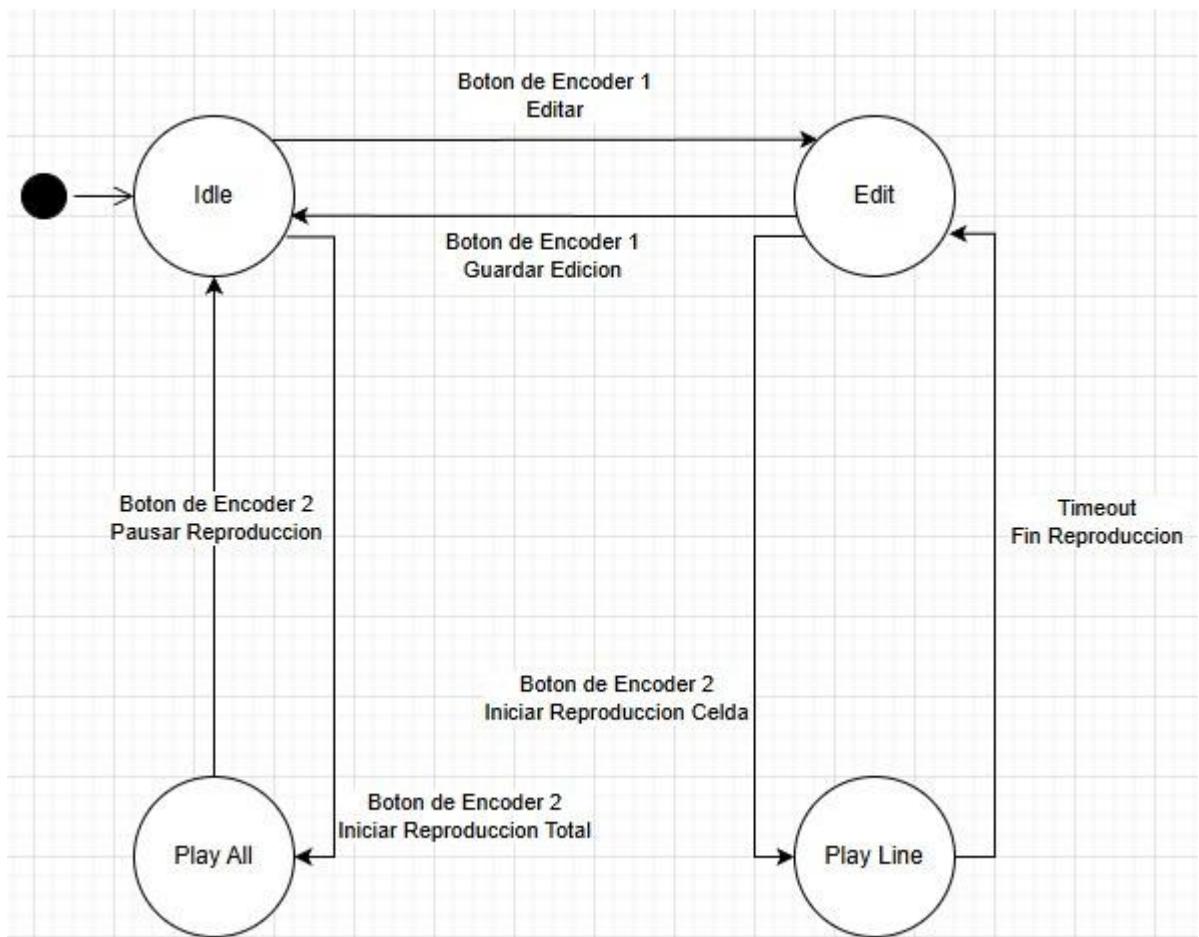
- Este trabajo propone un prototipo embebido que explora esa misma idea de síntesis por pasos, utilizando un ESP32 y una pantalla TFT ILI9341. El sistema organiza los datos en una matriz de 16 filas (steps) por 4 columnas (canales): la primera columna controla un generador de ruido estilo NES, y las tres restantes controlan osciladores tonales (dos cuadrados y uno triangular) mediante notas tipo MIDI. Cada celda almacena un valor entero que se interpreta como preset de percusión o como altura de nota.

La interfaz gráfica muestra la grilla completa (con 6 filas visibles a la vez), permite navegar entre filas y columnas con encoders, editar los valores numéricos y ajustar el tempo en BPM. La reproducción recorre secuencialmente los steps, resintetizando audio en tiempo real y enviándolo al módulo MAX98357A por I<sup>2</sup>S. Además, el sistema se integra con una aplicación Android a través de MQTT, que puede leer y

sobrescribir la matriz, modificar el estado de la máquina (IDLE, EDIT, PLAY\_ALL, PLAY\_LINE) y ajustar el tempo. El objetivo educativo del trabajo es: por un lado, comprender la organización de datos y estados en un secuenciador por pasos; por otro, ejercitarse el diseño de firmware embebido con FreeRTOS, comunicación inalámbrica y síntesis de audio digital en un microcontrolador.

## Desarrollo

### Diagrama de Estados



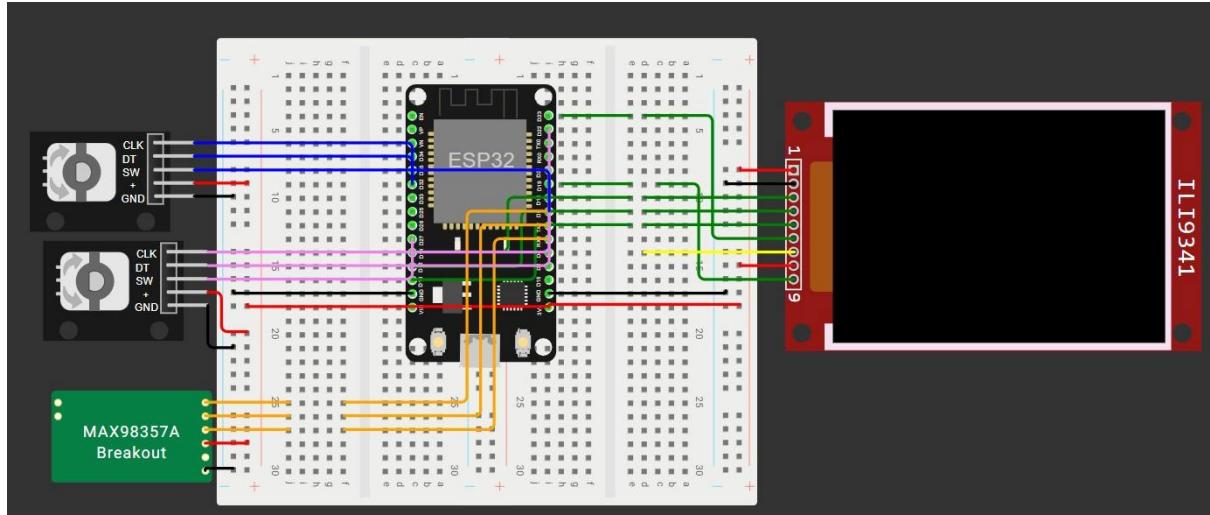
El firmware implementa una máquina de estados explícita, que se evalúa únicamente en el loop principal del ESP32. Los estados definidos son:

- IDLE: estado de reposo. Se permite navegar por las filas con el Encoder 1, sin edición ni reproducción activa.
- EDIT: estado de edición. Se modifican los valores de la celda seleccionada y el tempo en BPM.
- PLAY\_ALL: reproducción continua de todos los steps de la matriz, avanzando fila a fila.
- PLAY\_LINE: previsualización (preview) de una fila concreta durante un intervalo corto, tras el cual el sistema vuelve automáticamente a EDIT.

Las transiciones se disparan tanto desde los encoders (giro y pulsación) como desde mensajes MQTT en el tópico /simulator/state. Los eventos de hardware se capturan en una tarea FreeRTOS de entrada, que traduce los cambios

de los encoders a eventos simbólicos y los envía por una cola hacia la máquina de estados. De forma análoga, el callback MQTT traduce los mensajes recibidos a eventos como “EV\_MQTT\_TO\_PLAY\_ALL”, “EV\_MQTT\_TO\_EDIT” o “EV\_MQTT\_PLAY\_ROW”, que también son consumidos en el loop.

### *Conexiones del circuito*



El circuito se monta sobre un ESP32 DevKit v1, una pantalla TFT ILI9341 conectada por SPI, dos encoders KY-040, un módulo MAX98357A para la salida de audio y, opcionalmente, un pequeño parlante de baja potencia. La alimentación se realiza típicamente desde el puerto USB del ESP32. Todas las masas (GND) se comparten para asegurar una referencia común.

### *Descripción Fisico-Electronica*

#### **1. Display TFT ILI9341**

- Función: Visualiza la grilla de pasos (16×4, con 6 filas visibles), los íconos de canal en la parte superior, el indicador de estado y el tempo en BPM. El encabezado muestra el título “FAMICO” y se utiliza una barra lateral tipo scrollbar para indicar la porción visible de la matriz.
- Alimentación: Se conecta a 3V de del propio sistema, con GND común al ESP32. Esto garantiza suficiente corriente para el refresco de pantalla y evita reinicios inesperados del microcontrolador.
- Conexión
 

	lógica		(SPI):
• CS	→	GPIO14	del
• RST	→	GPIO12	del
• DC	→	GPIO13	del
• MOSI→	→	GPIO23	del
• SCK	→	GPIO18	del
• MISO→	→	GPIO19	del

 ESP32

- Señales: Recibe comandos SPI del ESP32 a través de la librería TFT\_eSPI, la cual se configura con tft.setRotation(1) para trabajar en orientación apaisada. El firmware dibuja la grilla, los valores numéricos de las celdas, el resaltado de selección y el panel de estado (modo y BPM).

## 2. Encoder rotativo KY-040 (Encoder 1 y Encoder 2)

- Función: Permiten la navegación en la grilla y la edición de valores, así como el cambio de estado según la pulsación de sus botones.
- Conexiones del Encoder 1:
 

• CLK	→	GPIO21	1:
• DT	→	GPIO22	(ENC1_DT)
• SW	→	GPIO27	(ENC1_SW)
• VCC	→	3.3	V
• GND	→		GND
- Conexiones del Encoder 2:
 

• CLK	→	GPIO32	2:
• DT	→	GPIO35	(ENC2_DT)
• SW	→	GPIO5	(ENC2_SW)
• VCC	→	3.3	V
• GND	→		GND

## 3. Módulo MAX98357A (I<sup>2</sup>S)

- Función: Convierte la señal de audio digital I<sup>2</sup>S generada por el ESP32 en audio analógico amplificado para un parlante pequeño. Implementa un amplificador clase D con entrada I<sup>2</sup>S.
- Conexiones I<sup>2</sup>S desde el ESP32:
 

• BCLK	→	GPIO33	ESP32:
• LRCLK	→	GPIO25	(PIN_BCLK) / WS
• DIN	→	GPIO26	(PIN_LRCK)
• GND	→		(PIN_DOUT)
• VIN	→	3.3	GND V
- El firmware configura el periférico I<sup>2</sup>S del ESP32 para trabajar como master transmisor a 44,1 kHz, 16 bits estéreo. En cada iteración del loop, la máquina de estados genera un búfer de muestras PCM a partir de las voces NES (dos osciladores cuadrados, uno triangular y ruido con envolvente) y lo envía al MAX98357A mediante DMA.

## 5. ESP32 DevKit V1

- **Función:** Controla todo el sistema embebido: lectura de encoders y potenciómetro, actualización del display, ejecución de la lógica de reproducción.
- **Detalles electrónicos:**
  - Alimentación: 3.3 V interno.
  - GPIO digitales: lectura de encoders y manejo de la interfaz SPI hacia la TFT.
  - Periférico I<sup>2</sup>S: generación de audio digital hacia el MAX98357A.
  - WiFi 2.4 GHz: conexión a la red inalámbrica y al broker MQTT (por ejemplo broker.emqx.io).
- El firmware utiliza FreeRTOS (incluido en el SDK del ESP32) para separar responsabilidades en distintas tareas:
  - vInputTask: lectura de encoders y envío de eventos a la cola.
  - vMqttTask: único “dueño” del cliente MQTT; se encarga de reconectar, procesar mqttClient.loop() y publicar mensajes pendientes.
  - Loop principal: contiene la máquina de estados, redibuja la interfaz cuando es necesario y genera continuamente las muestras de audio para el periférico I<sup>2</sup>S.



# ***Manual de usuario de Famico***

## **¿Qué es?**

Famico es un tracker de música por pasos sobre ESP32 que se inspira en el flujo de trabajo y el sonido de la NES. El usuario edita una grilla de 16 pasos por 4 canales, donde cada fila corresponde a un step temporal y cada columna a un canal de sonido: percusión/ruido, Square 1, Square 2 y Triangle. El dispositivo permite reproducir la secuencia completa (PLAY\_ALL), preescuchar una fila concreta (PLAY\_LINE) y modificar el tempo en BPM desde la propia interfaz o a través de MQTT.

## **Hardware Utilizado**

- ESP32 DevKit v1
- Pantalla TFT ILI9341 (SPI)
- 2 encoders rotativos con pulsador (KY-040)
- Módulo MAX98357A (amplificador clase D con entrada I<sup>2</sup>S)
- Parlante pequeño
- Protoboard y cables de conexión

Nota: la versión actual del firmware ya implementa la síntesis de audio en tiempo real y la salida por I<sup>2</sup>S. El prototipo puede utilizarse como instrumento autónomo o en combinación con una aplicación Android que actúa como editor remoto de patrones.

## **Interfaz**

- Grilla: 16 filas × 4 columnas (6 filas visibles a la vez). Cada celda muestra un valor entero en decimal.
  - Columna 0: valor 0–15 que selecciona un preset de percusión/ruido estilo NES.
  - Columnas 1–3: valor 0–127 interpretado como nota MIDI para las voces cuadradas y triangular.
- Selección: la fila seleccionada se resalta en color distinto y se indica con un cursor de edición; la columna seleccionada se usa para elegir qué canal editar.
- Panel de estado: ubicado en la esquina superior derecha, muestra el modo actual (IDLE, EDIT, PLAY\_ALL, PLAY\_LINE) y el tempo en BPM.
- Scrollbar lateral: indica qué parte de la matriz (qué filas) está siendo visible.

## **Estados:**

- ◆ IDLE: estado de reposo. Permite navegar por las filas sin reproducir ni editar.
- ◆ EDIT: estado de edición. Se modifican los valores de la celda actual y el tempo en BPM.
- ◆ PLAY\_ALL: reproduce de forma continua todos los steps, avanzando por la matriz.
- ◆ PLAY\_LINE: previsualiza (preview) la fila actual durante un breve intervalo y luego vuelve a EDIT.

## Controles

### Encoder 1

- En IDLE:
  - Giro: mueve la fila seleccionada (cursor vertical).
  - Botón: inicia la reproducción completa (PLAY\_ALL).
- En EDIT:
  - Giro: si está seleccionada una columna de la grilla (0–3), incrementa/decrementa el valor de la celda actual.  
Si está seleccionada la “columna virtual” BPM, aumenta/disminuye el tempo en bloques de 10 BPM.
  - Botón: inicia PLAY\_LINE (previvualización de la fila actual).
- En PLAY\_ALL:
  - Giro: mueve la fila de referencia mientras se reproduce (para inspeccionar otra parte de la grilla).
  - Botón: detiene la reproducción y vuelve a IDLE.
- En PLAY\_LINE:
  - Giro: permite seguir editando el valor de la celda o el BPM mientras suena el preview.
  - Botón: cancela el preview y vuelve a EDIT.

### Encoder 2

- En IDLE:
  - Botón: entra en EDIT en la fila seleccionada.
- En EDIT:
  - Giro: cambia la columna seleccionada (0–3) y permite seleccionar la posición de edición de BPM.
  - Botón: vuelve a IDLE.
- En PLAY\_ALL:
  - Botón: pasa directamente a EDIT.
- En PLAY\_LINE:
  - Giro: cambia la columna seleccionada durante el preview.
  - Botón: vuelve a IDLE.

### Tempo (BPM)

- El tempo se maneja en BPM (rango típico 20–300). Puede modificarse:
  - Desde la interfaz, seleccionando la “columna BPM” y girando el Encoder 1.
  - Remotamente, enviando un mensaje al tópico /simulator/tempo con un valor en BPM o en milisegundos por paso (por ejemplo, “240” o “100ms”).

## Uso paso a paso

### 1. Encendido

Al energizar el sistema, la pantalla muestra la grilla y la barra de estado en modo IDLE. La fila inicial suele ser la 0.

### 2. Navegación básica (IDLE)

- Girar el Encoder 1 para desplazarse por las filas.
- Pulsar el Encoder 2 para entrar en EDIT sobre la fila seleccionada.

### 3. Edición de celdas (EDIT)

- Girar el Encoder 2 para seleccionar la columna (0–3) o la posición BPM.

- Girar el Encoder 1 para aumentar/dismuir el valor de la celda o el BPM según la selección.
- Pulsar el Encoder 1 para previsualizar la fila actual (PLAY\_LINE); al terminar el preview el sistema vuelve a EDIT.
- Pulsar el Encoder 2 para volver a IDLE.

#### 4. Reproducción completa (PLAY\_ALL)

- Desde IDLE, pulsar el Encoder 1 para iniciar PLAY\_ALL.
- El sistema recorre la matriz step a step, disparando los presets de percusión y notas de cada fila.
- Mientras suena, es posible mover la fila de referencia con el Encoder 1.
- Pulsar el Encoder 1 detiene la reproducción y devuelve a IDLE; pulsar el Encoder 2 pasa a EDIT.

## Uso paso a paso

1. **Encendido:** El Display muestra la grilla con filas enumeradas. Estado inicial IDLE.
2. **Navegación básica (IDLE):**
  - a. Rotar Encoder1: mover cursor columna (izq/der).
  - b. Rotar Encoder2: mover cursor fila (arriba/abajo).
  - c. Selección visible: celda resaltada en amarillo.
3. **Entrar en edición:**
  - a. Pulsar Encoder1 (ENC1\_SW). Estado cambia a EDIT (indicador verde).
  - b. En EDIT, Encoder1 rotación: cambia menuPos (selecciona qué parámetro del PARAMS editar).
  - c. En EDIT, Encoder2 rotación: incrementa/decrementa el valor del parámetro seleccionado en la celda actual. Valores mostrados en hex (sprintf "%02X").
  - d. Para salir de EDIT: pulsar Encoder1 otra vez → vuelve a IDLE.
4. **Reproducir una celda:**
  - a. En EDIT, pulsar Encoder2 (ENC2\_SW) → playCell(cursorRow, cursorCol) muestra la marca "PLAY r=c=" y hace un pulso corto.
5. **Reproducción completa:**
  - a. En IDLE, pulsar Encoder2 → cambia a PLAY\_ALL. La tarea FreeRTOS taskPlay ejecuta playAll(currentRow, stepCol) en cada paso y avanza columnas y filas según tempo.
  - b. Al finalizar todas las filas, la reproducción para y vuelve a IDLE.
  - c. Durante PLAY\_ALL, pulsar Encoder2 detiene y vuelve a IDLE.
6. **Ajustar tempo en tiempo real:** girar potenciómetro; la tarea de reproducción lee tempoMs con cada paso.

- URL al proyecto de Wokwi:  
<https://wokwi.com/projects/449174337447368705>

## Referencias

1. — SOA-UNLaM: “PUBLICO: Sistemas embebidos e Internet de las Cosas”, disponible en: [https://www.soa-unlam.com.ar/wiki/index.php/PUBLICO%3ASistemas\\_embebidos\\_e\\_Internet\\_de\\_las\\_Cosas](https://www.soa-unlam.com.ar/wiki/index.php/PUBLICO%3ASistemas_embebidos_e_Internet_de_las_Cosas)
2. "NES Audio (APU) - Emulation Online," Emulation Online, 2025, <https://www.emulationonline.com/systems/nes/apu-audio/>
3. [Llama Elitista], "¿Cómo funciona la Música de Videojuegos Retro?" YouTube, 11 de Abril del 2025. <https://www.youtube.com/watch?v=v8Z2ft8XTHk&t=860s>