AUDIOSCOPE

Sistema Embebido de Visualización Espectral en Tiempo Real

Mario Gutierrez · Asier Andrés · Alexis Martinez | Proyecto Final — 2025

AUDIOSCOPE — VISIÓN GENERAL

Analizador espectral embebido con visualización RGB 64×64 y control I²C

🎵 Audio Input

RCA / XLR / Jack
(Fun Generation UA-202)

🖥️ Display

Matriz RGB 64×64 LEDs
(Adafruit HAT)

🎛️ Control

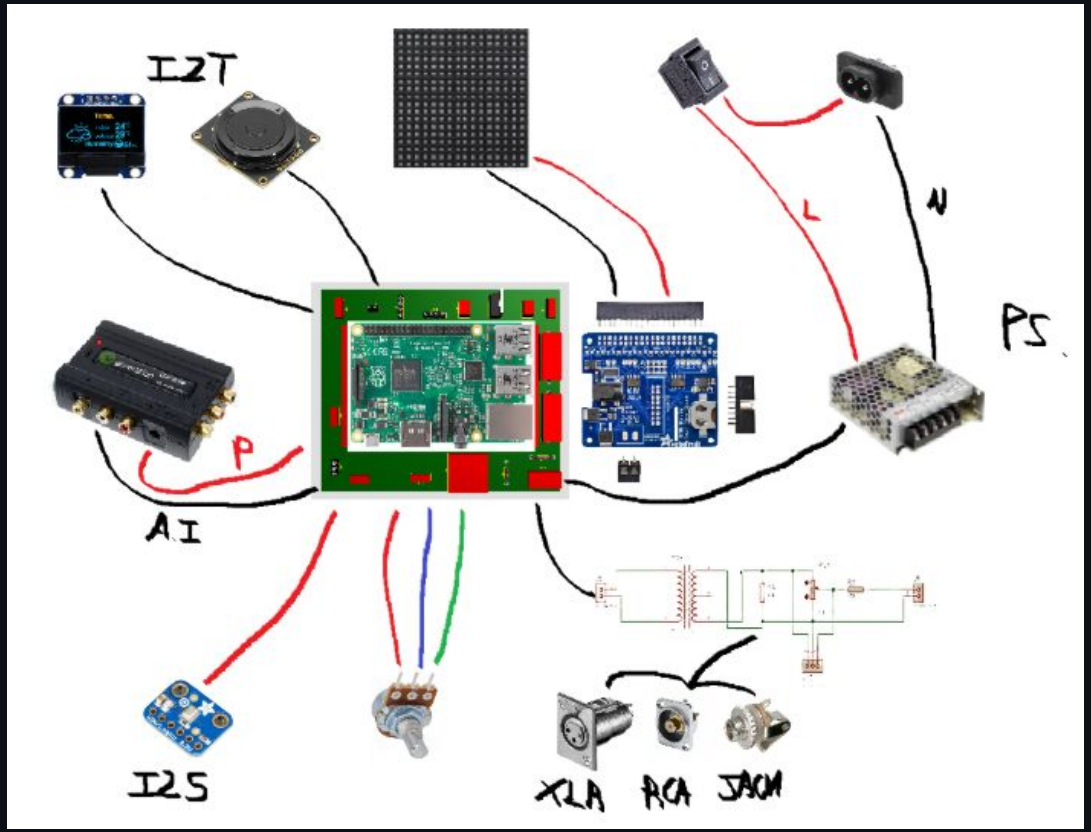
Encoder rotatorio ANO
+ OLED 128×64 I²C

⚡ Procesador

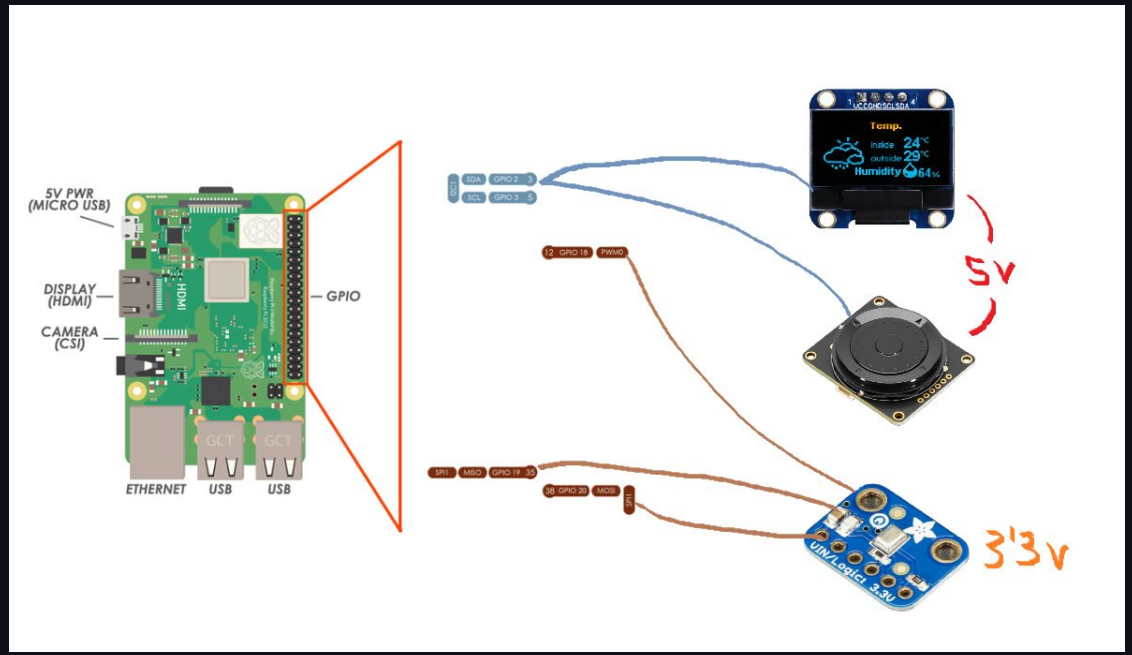
Raspberry Pi 4B
Python 3 / NumPy FFT

01 — ARQUITECTURA HARDWARE

Diagrama de conexiones y flujo de señal completo



02 — PINOUT RASPBERRY PI & BUS I²C



I²C Bus (GPIO 2/3)

- SDA ⟶ GPIO 2 (pin 3)
- SCL ⟶ GPIO 3 (pin 5)
- OLED SSD1306 ⟶ 0x3C
- Seesaw Encoder ⟶ 0x49

PWM / GPIO

- GPIO 18 (pin 12) ⟶ PWM0
- GPIO 19/20 ⟶ SPI (reservado)
- HAT ⟶ conector 40 pines

USB

- USB-C ⟶ Alimentación
- USB ⟶ Fun Gen UA-202
- Audio: 44100Hz mono

03 — ARQUITECTURA SOFTWARE

3 módulos Python · threading.Thread · Callback de audio asíncrono

Python 3 | NumPy / FFT | sounddevice | rgbmatrix | seesaw I²C | Proteus PCB

AudioS.py

Módulo Principal

- Main loop (polling encoder 50ms)
- Toggle START/STOP threading
- Estado global audio_running
- Lanza show_brillo/palette/mic

menu_RGB.py

UI & Hardware I²C

- Init OLED SSD1306 @ 0x3C
- Init Seesaw encoder @ 0x49
- draw_menu(), show_brillo()
- show_palette(), show_mic()

f_RGB.py

DSP + Visualización

- audio_callback() async
- FFT + 32 bandas log
- ATTACK/RELEASE + PEAK
- Paletas HSV + SetImage()

⚡ audio_thread (daemon=True): callback asíncrono en paralelo al main loop — stop por lambda: not audio_running

04 — PIPELINE DE AUDIO & DSP

Captura USB ⟶ Windowing ⟶ FFT ⟶ 32 bandas logarítmicas ⟶ Normalización

1
InputStream
sounddevice
FS=44100Hz
Block=1024

2
Hanning Window
Reducción fugas FFT

3
rfft()
NumPy FFT
N=1024
513 bins

4
dB Scale
$20 \cdot \log_{10}(|X|)$
Clip [0, 120]

5
32 Bandas Log
20Hz–20kHz
log-spaced

6
RMS Banda
irfft() + $\sqrt{(\mu^2)}$
norm ×20

PARÁMETROS CLAVE

Sample Rate **44,100 Hz**	Block Size **1,024 muestras**	Latencia **~23 ms/bloque**
Bandas **32 (log)**	Boost >16kHz **× 1.25**	Normalización **rms×20, clamp 0-1**

05 — MOTOR DE VISUALIZACIÓN RGB

Suavizado dinámico, barras de pico y 5 paletas HSV con degradado 3D

SUAVIZADO TEMPORAL

ATTACK = 0.8
Subida rápida: prev + (target−prev)×0.8

RELEASE = 4.25
Caída lenta: prev − 4.25 px/frame

PEAK_FALL = 1
Pico flotante cae 1 px/frame tras máximo

PALETAS DE COLOR (HSV)

RED	H 0.0→0.11
YELLOW	H 0.08→0.23
BLUE	H 0.40→0.57
RAINBOW	sin(π·level)
SPECIAL	Verde→Naranja→Rojo

RENDERIZADO POR BANDA (×32):

① Calcular new_height con ATTACK/RELEASE

② Actualizar peak_height (−PEAK_FALL px/frame)

③ Colorear: fastled_palette(row/63)

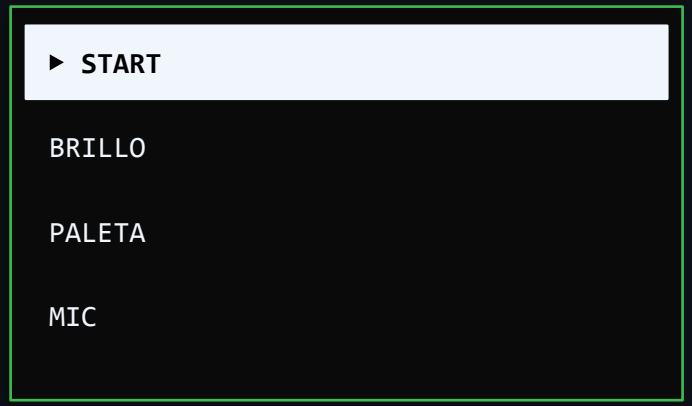
④ img[63−row, band×2] y band×2+1

⑤ Peak pixel en peak_y = 63−peak_height

⑥ matrix.SetImage(Image.fromarray(img))

06 — INTERFAZ DE USUARIO

OLED 128×64 (0x3C) + Encoder ANO Seesaw I²C (0x49) + 4 botones



OLED 128×64
(Midas MDOB128064WV)

ENCODER + BOTONES

- **GIRO CW/CCW**
  → Navegar / ajustar valor
- **BTN CENTER (pin 1)**
  → Confirmar selección
- **BTN LEFT (pin 5)**
  → Volver al menú

BRILLO

Encoder: ajusta 1–100%
Barra progreso en OLED
matrix.brightness = valor

PALETA

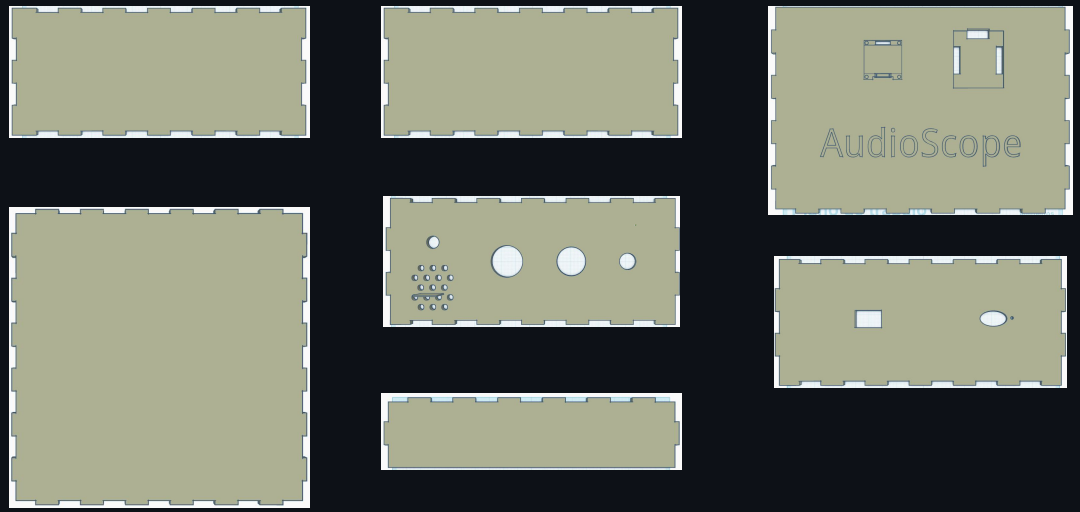
Encoder: cicla 5 paletas
red/yellow/blue/rainbow/special

MIC

Animación ícono micrófono
Reservado para futuras funciones

07 — DISEÑO MECÁNICO & CARCASA

Piezas diseñadas en Inkscape + MakerCase — Impresión 3D



Software CAD

Inkscape (diseño 2D)
MakerCase (generación de encastres)
Tinkercad (3D preview)

Material

Material de impresión 3D PLA

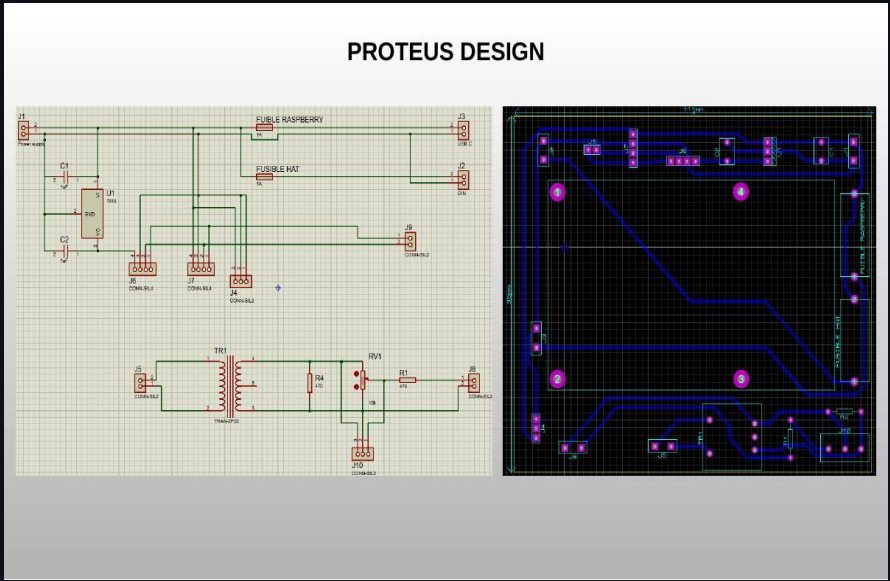
Componentes

Panel frontal: conect. RCA, XLR, Jack, Poten.
Panel superior: OLED + Encoder
Panel trasero: switch + alimentación

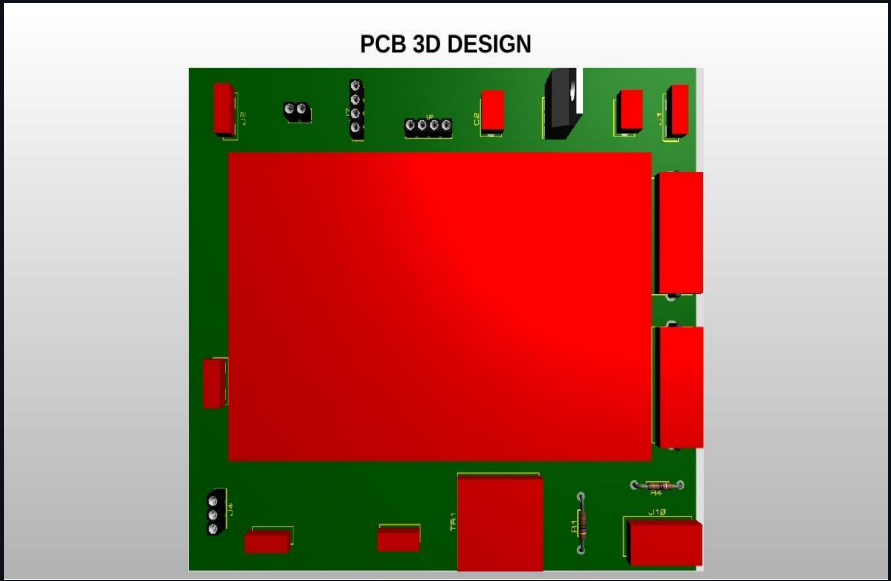
08 — DISEÑO ELECTRÓNICO (Proteus + PCB)

Esquemático en Proteus · PCB 3D con fuente de alimentación integrada

Esquemático Proteus



PCB 3D Design



Regulador

Caps filtro

Transformador

Reducción a 3v para alimentar microfono

Fusibles

1A Raspberry + 1A HAT

Conectores

SIL-2/3/4 modular

Problemas Encontrados

Tuvimos que investigar conexiones de componentes que inicialmente parecían complejas y difíciles de entender.

Aprender a diseñar las piezas 3D nos llevó mucho tiempo, asi como la impresiõn y el ensamblaje de la caja.

El código en Python fue uno de los aspectos más difíciles del proyecto.

Dependencia de mantener la Raspberry Pi conectada en todo momento impedía que un compañero pudiera trabajar de forma independiente.

🔧 FUTURAS MEJORAS

Más estilos visuales

Osciloscopio(sinusoidal), circular, etc…

Pantalla RGB mayor

Configurar una segunda matriz P3 64×64

Altavoces integrados

Escuchar el audio en el dispositivo a la vez que se pueda visualizar el display

A.M.A

AudioScope

Electronic Maintenance Specialists

▶ Video demo

⌥ GitHub — código fuente

🌐 Webpage del proyecto

📸 Instagram

🎵 TikTok

Escanea para
acceder al proyecto

Mario Gutierrez · Asier Andrés · Alexis Martinez