Hoja de ruta detallada (junio → octubre)

> Meta: **llegar a la primera semana de octubre con un prototipo que capture audio, lo clasifique en la Raspberry Pi Zero 2 W y, según esa clasificación, ejecute filtrado/atenuación en el ESP32 para que el usuario escuche el sonido "limpio" en tiempo real (latencia ≤ 40 ms).**

---

0 · Preparativos (hoy–15 jun)

Acción	Resultado	Recursos rápidos
Crear **repositorio Git** y carpeta compartida.	Control de versiones y documentación centralizada.	GitHub Classroom / GitLab.
Instalar **tool-chains**	- PlatformIO + ESP-IDF (C/C++) - Python 3.10 + pip + TensorFlow Lite	Tutoriales oficiales.
Armar **estación de pruebas**	ESP32 + 1 INMP441 + parlante o auriculares simples + Raspberry Pi conectada por USB-UART	Fuente 5 V ≥ 2 A, cables jumper.

---

1 · Sprint 1 — Captura & reproducción local (16 jun–30 jun)

Sub-objetivo	Tareas	Entregable
ESP32 graba y reproduce	• Configurar I2S a 16 kHz/16 bits. • Implementar *buffer* circular de 512 muestras. • Reproducir audio en loop por DAC interno o PCM5102.	Video + código que demuestre "grabar-y-reproducir" sin cortes.
Medir latencia base	• Inyectar clap/chasquido y medir tiempo hasta la salida.	Documento con ms medidos y ajustes sugeridos.

Qué estudiar ► I2S, DMA en ESP32, buffers circulares, latencia en audio embebido.

---

2 · Sprint 2 — Preprocesamiento DSP (1 jul–15 jul)

Sub-objetivo	Tareas	Entregable
Filtros básicos	• Implementar paso-alto 120 Hz y paso-bajo 8 kHz (ESP-DSP).	Audio “antes/después” guardado en SD o PC.
Notch dinámico	• Diseñar función que reciba frecuencia & Q-factor y actualice coeficientes IIR en caliente.	Demo CLI “notch 3 kHz” que atenúe un tono de prueba.

Qué estudiar ► teoría FIR/IIR, librería **esp-dsp**, arduinoFFT para validación.

---

3 · Sprint 3 — IA de clasificación (16 jul–31 jul)

Sub-objetivo	Tareas	Entregable
Dataset mínimo	• Extraer 3 clases: voz, bocina, grito (UrbanSound8K). • 100 clips x clase.	Carpeta `/data` + README.
Modelo TFLite	• Generar espectrogramas 128×64. • CNN simple (Keras). • Exportar a TFLite < 1 MB.	`model.tflite` + script de inferencia que corre < 50 ms en la RPi.
Interfaz serie	• Enviar string JSON `{"cls":"bocina","conf":0.88}` por UART a 115 200 bps.	Log en la Raspberry mostrando tráfico en vivo.

Qué estudiar ► MFCC, espectrogramas, Keras → TFLite, `pyserial`.

---

4 · Sprint 4 — Integración IA ↔ DSP (1 ago–15 ago)

Sub-objetivo	Tareas	Entregable
Protocolo comando-respuesta	• En ESP32 parsear UART. • Asociar cada clase a preset de filtro/notch.	Documento `.proto` + demo “grito → atenuar 3–5 kHz”.
Timing end-to-end	• Medir: sonido → detección → acción → salida. • Meta < 40 ms.	Tabla con tiempos y cuellos de botella.

---

5 · Sprint 5 — Hardware & energía (16 ago–31 ago)

Sub-objetivo	Tareas	Entregable
Power-pack	• BMS LiPo 2000 mAh + TPS63020 boost. • Medir consumo (µCurrent / multímetro).	Autonomía real ≥ 4 h con ciclo de prueba.
PCB prototipo	• Diseñar carrier ESP32 + MEMS + conectores. • Ordenar PCB 1 capa (JLCPCB).	PCB soldados y funcionando en mesa.

Qué estudiar ► gestión de batería, buck-boost, KiCad básico.

---

6 · Sprint 6 — Carcasa & ergonomía (1 sep–15 sep)

Sub-objetivo	Tareas	Entregable
Diseño 3D	• Fusion 360: alojamiento placas + pads auriculares.	STL listo para imprimir (PLA).
Iteración rápida	• Imprimir, ensamblar, testear ajuste. • Registrar feedback de usuario.	Fotos + reporte de confort / peso.

---

7 · Sprint 7 — Validaciones finales (16 sep–30 sep)

Sub-objetivo	Tareas	Entregable
Pruebas de campo	• Aula real y avenida ruidosa. • Recoger opiniones y métricas (SPL antes/después).	Matriz de resultados + vídeo demo.
Bug-fix & polish	• Afinar filtros, limpiar código, documentar.	Release `v0.9-oct` en Git + manual corto.

---

8 · Entrega de octubre

- **Demo en vivo**: bocina grabada + conversación → TEARIS atenúa bocina, deja pasar voz.
- **Pitch de 5 min** con diagrama de bloques, cifras de latencia y autonomía.
- **Dossier PDF** (20 páginas) con diseño, código, lecciones y próximos pasos (ANC avanzado).

Roles sugeridos (4 personas)

Persona	Responsabilidades desde hoy
A – Audio DSP	Sprints 1–2–4 (I2S, filtros, timing).
B – ML & Back-end	Sprints 3–4 (dataset, modelo, protocolo).
C – Hardware & Energía	Sprint 5 (PCB, batería) + soporte integración.
D – Diseño & QA	Sprint 6 (carcasa), Sprint 7 (pruebas), documentación y pitch.

Rotación : cuando uno termina su bloque, apoya a otro (ej. ML ayuda a ajustes de filtros, diseño ayuda a mediciones de latencia).

Recursos de estudio rápidos

- **I2S & DMA en ESP32** → *Espressif ESP-IDF I2S Driver Guide*.
- **DSP en microcontroladores** → *esp-dsp/examples/audio*.
- **Audio ML básico** → TensorFlow tutorial *“Simple audio recognition”*.
- **FxLMS cancelation** → Blog *Harris Kim “ANC on microcontrollers”* (código C).
- **Gestión LiPo & TPS63020** → TI App-Note *SLVA517*.

Próximo paso inmediato

1. Montar el banco de pruebas (ESP32 + mic + auricular).
2. Crear repositorio y subir *Hello-I2S* esta misma semana.

Con este plan y disciplina quincenal de revisión de hitos, llegarán a octubre con un **prototipo real, demostrable y con métricas medibles**. ¡A trabajar!