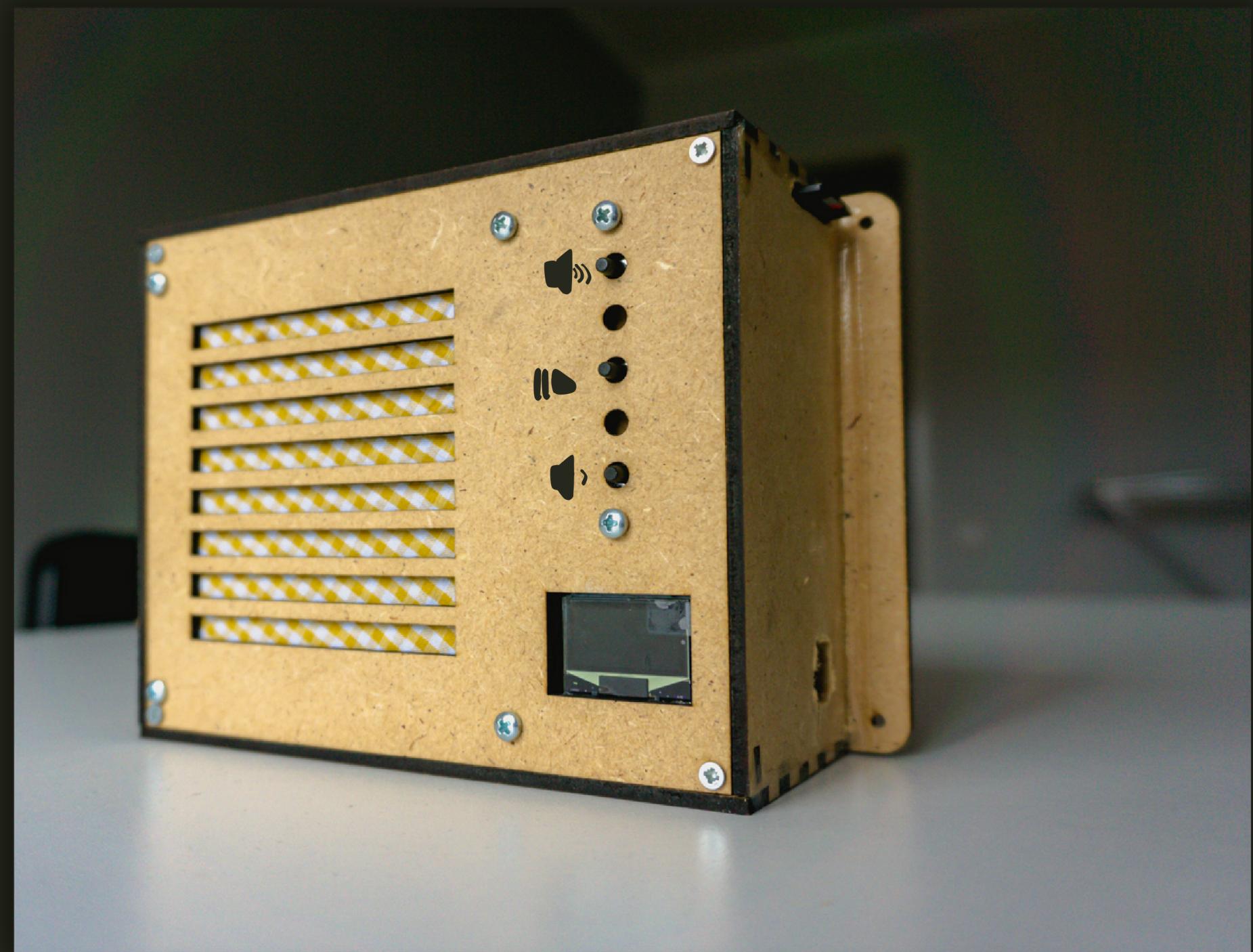


# Reproductor MP3 & WebRadio

*Yago Carballo Barroso  
Ramon Llobet Duch*

Proyecto Final de  
Procesadores Digitales

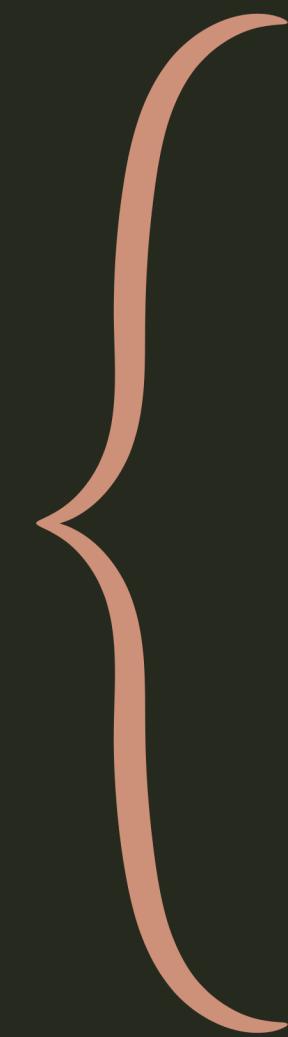


# ÍNDICE

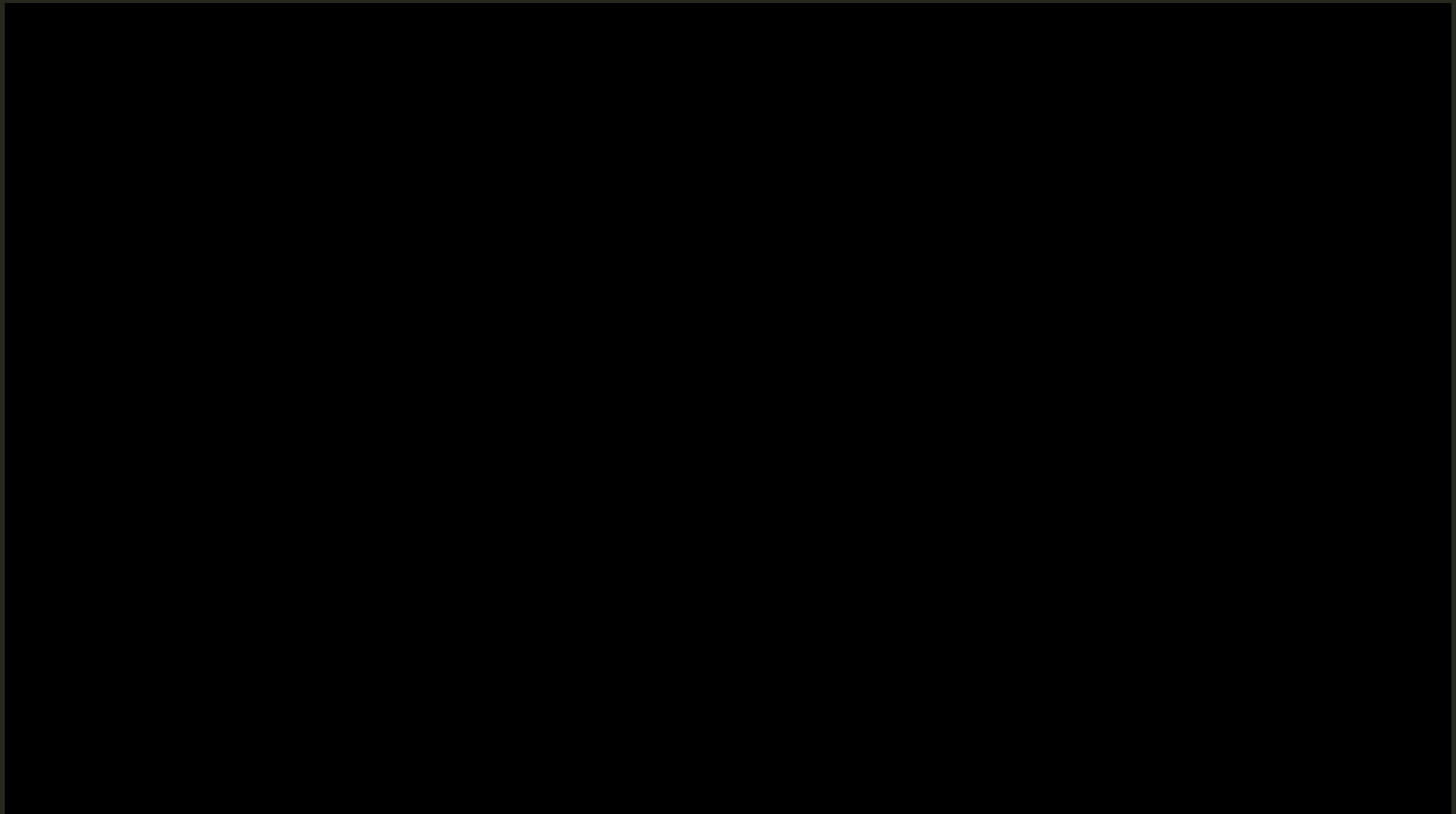
- Introducción
- Qué hemos hecho?
- Vídeo demostración
- Cómo lo hemos hecho?
- Diagrama de flujo
- Software
- Preguntas

# QUÉ HEMOS HECHO?

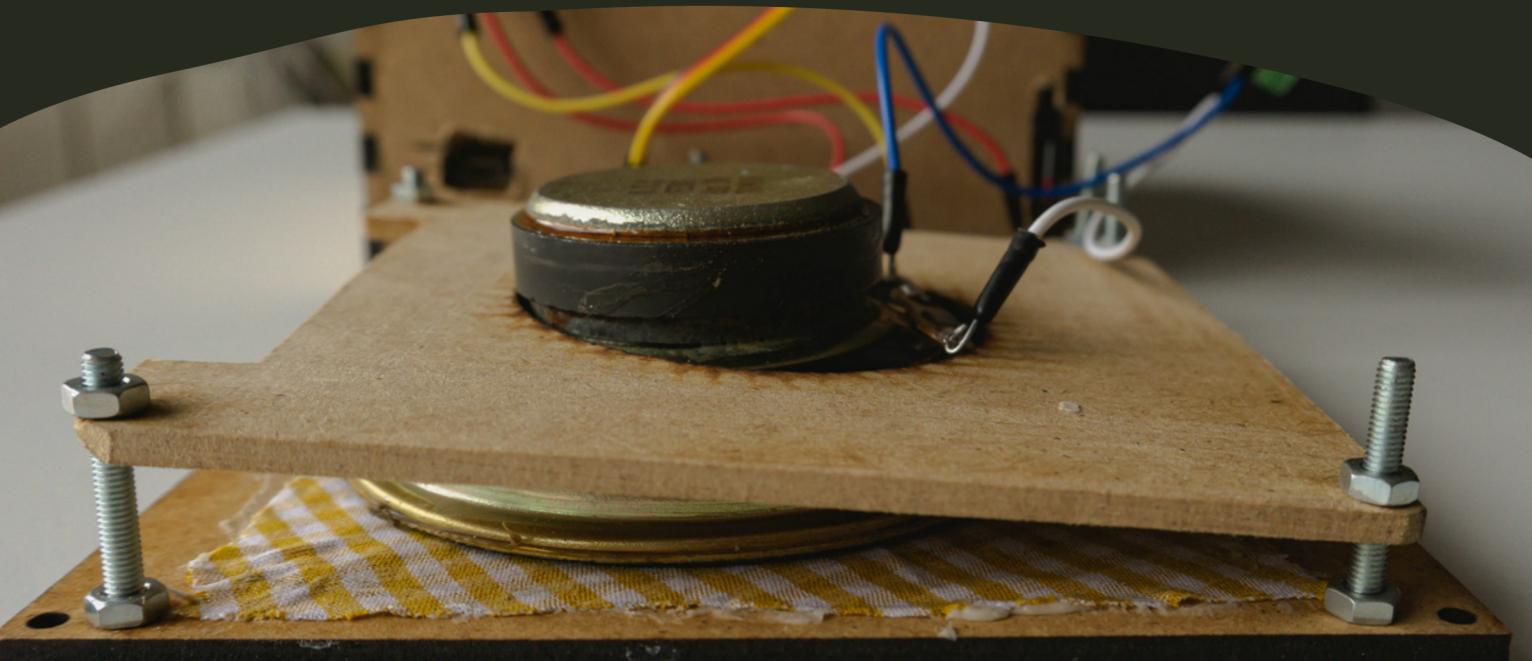
**Reproductor de audio  
MP3 y Webradios**



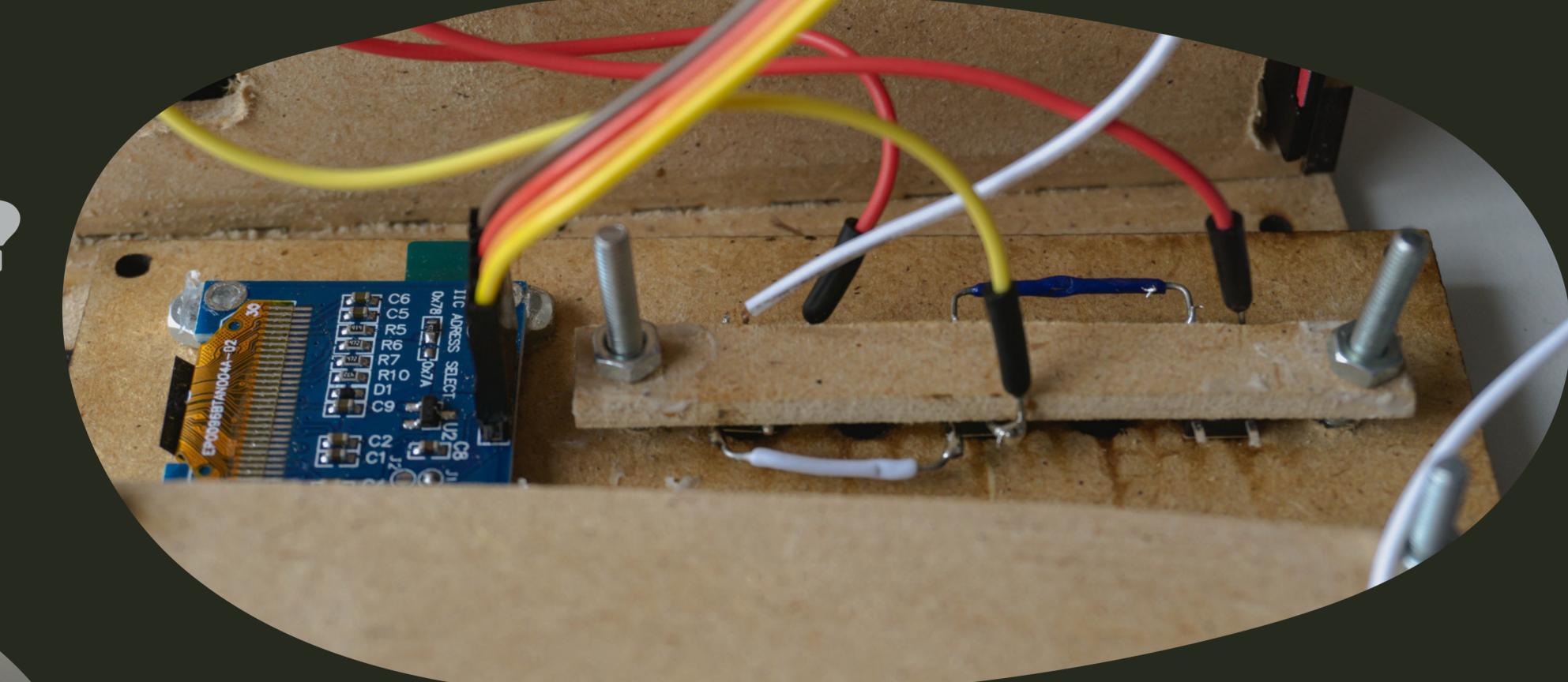
- Reproducir canciones desde SD
- Controlar la reproducción por web y botones.
- Retransmitir emisoras de radio
- Filtros
- Datos por pantalla OLED



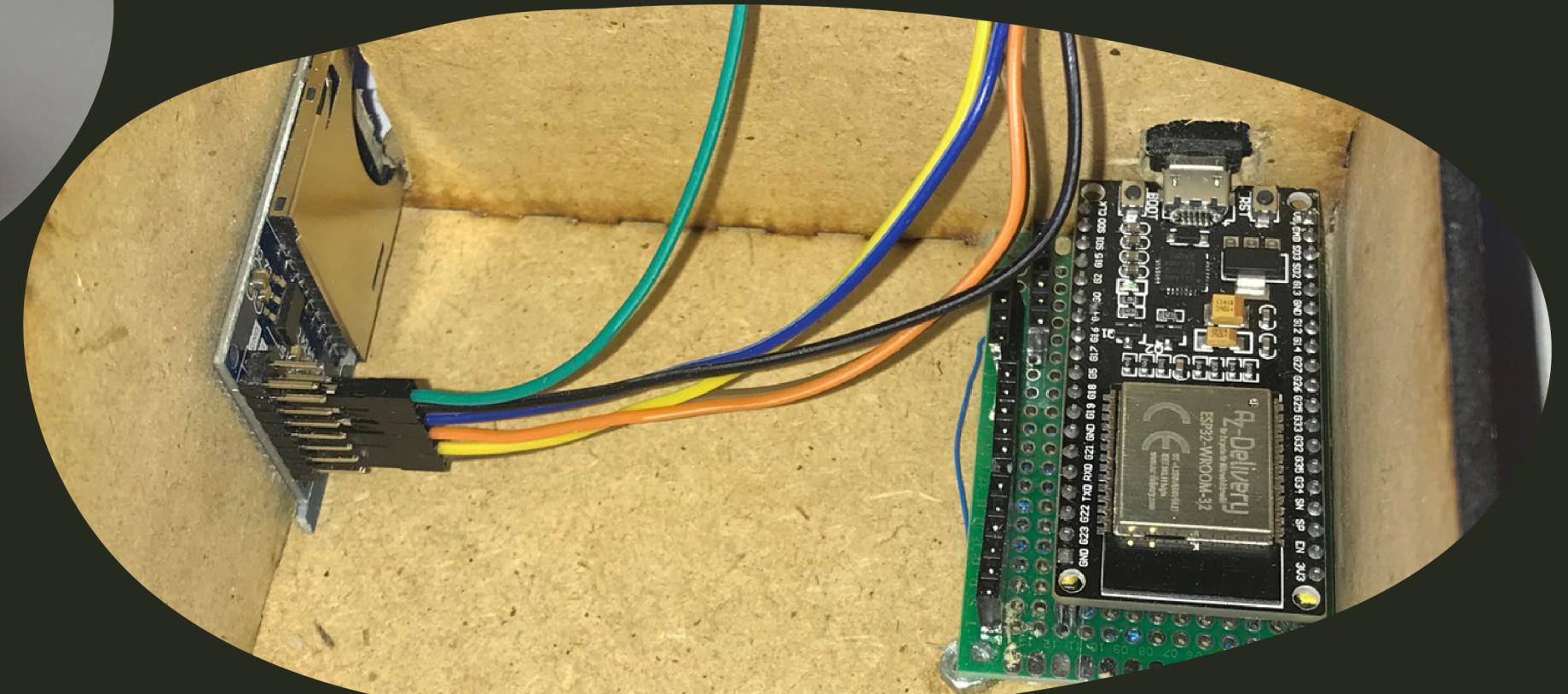
# CÓMO LO HEMOS HECHO?



ALTAZOZ

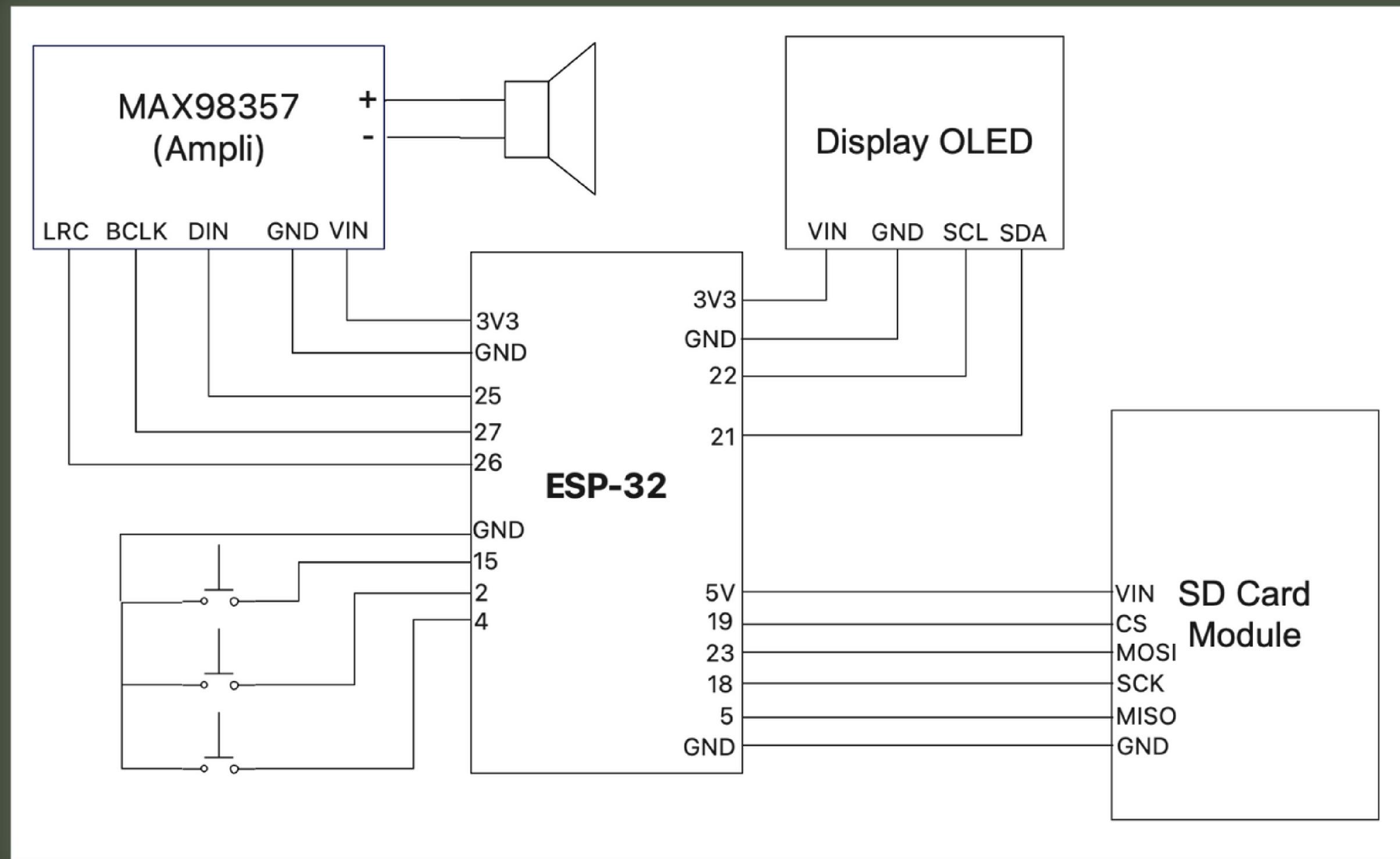


BOTONES Y PANTALLA

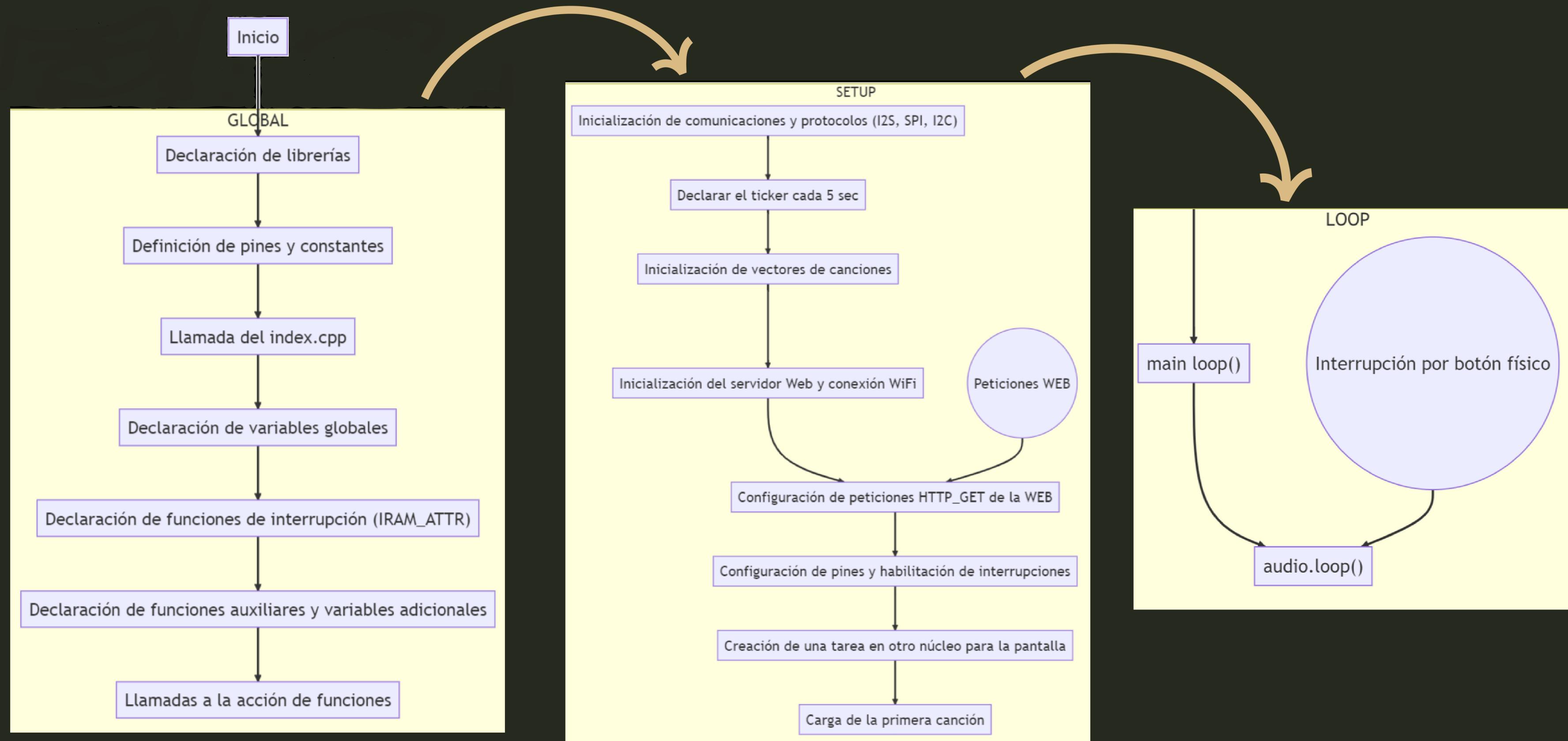


LECTOR SD Y ESP32

# CONEXIONES



# DIAGRAMA DE FLUJO



# SOFTWARE I

## Librerías

```
#include <Arduino.h>
#include <Audio.h>
#include "SD.h"
#include "FS.h"
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <SPI.h>
#include <vector>
#include <Ticker.h>
#include <Adafruit_SSD1306.h>
```

# SOFTWARE II

```
vector<const char *> songsRock;
vector<const char *> songsPop;
vector<const char *> songsMetal;
vector<const char *> songsOtros;
vector<const char *> *songFiles[4] = {&songsRock, &songsPop, &songsMetal, &songsOtros};
```

## Vectores de canciones

```
linea = archivo.readStringUntil('.');
linea.trim();
NsongsRock.push_back(linea);
prov = URLconverter(linea, GenreNames[GenreNames.size() - 1]);
char *temp = new char[prov.length() + 1];
strcpy(temp, prov.c_str());
songsRock.push_back(temp);
```

**Leer txt y meterlo en `vector<const char*>`**

# SOFTWARE III

```
attachInterrupt(digitalPinToInterrupt(BUTTON_pin), isr, RISING); // Asignar la interrup
attachInterrupt(digitalPinToInterrupt(BUTTON_volumeUP_pin), ISR_volumeUP, RISING); // As
attachInterrupt(digitalPinToInterrupt(BUTTON_volumeDOWN_pin), ISR_volumeDOWN, RISING); /

//Se crea una tarea en segundo plano para controlar la pantalla OLED:
xTaskCreatePinnedToCore(
    pantalla, // Función que implementa la tarea.
    "pantalla", // Nombre de la tarea.
    2000, // Tamaño de la pila de la tarea
    NULL, // Parámetros de la tarea
    1, // Prioridad de la tarea
    NULL, // Referencia a la tarea
    0); // Núcleo donde se ejecutará la tarea
```

**Interrupciones para botones físicos y Tarea en núcleo '0'  
para actualizar la pantalla cada segundo**

# SOFTWARE IV - WEB I

index.cpp

main.cpp



main.cpp

index.cpp

src > index.cpp > [e] html

```
1 #include<Arduino.h>
2
3 String html =<html><head><title>Reproductor MP3</title>
```

42 extern String html;



# SOFTWARE IV - WEB II

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    html += "<div class=\"album-list\">";
    html += "<div class=\"album\">";
    html += "<img src=\"https://m.media-amazon.com/images/I/919JyJJiTtL.\_SL1500\_.jpg\" alt=\"rock\">";
    html += "<h2>Rock</h2><ul>";

    for (int i = 0; i < NsongsRock.size(); i++)
    {
        html += "<li>" + NsongsRock[i] + "</li>";
    }
});
```

Función para mostrar álbumes

# SOFTWARE IV - WEB II

```
"<button class=\"btn\" onclick=\"toggleMusic()\">
```

▼ **Atributo onclick()** ▼

```
"<script>function toggleMusic() {var xhr = new XMLHttpRequest(); xhr.open('GET', '/toggle', true); xhr.send();};"
```

▼ **Función de JavaScript** ▼

```
server.on("/toggle", HTTP_GET, [](AsyncWebRequest *request) {
    audio.pauseResume();
    request->send(200);
});
```

**Función C++**

# SOFTWARE V

```
auto tick = []()
{
    if (audio.getAudioCurrentTime() >= audio.getAudioFileDuration() && audio.getAudioFileDuration() != 0)
    { // Cuando pones radios la duración es 0
        SongIndex++;
        if (SongIndex >= (*songFiles[GenreIndex]).size())
            SongIndex = 0;
        audio.connecttoFS(SD, (*songFiles[GenreIndex])[SongIndex]);
    };
};
```

**Ticker para cambio de canción automático**

# Preguntas