TUTORIEL — CONSTRUCTION D'UNE ENCEINTE BLUETOOTH PORTABLE À BASE D'ESP32

---

1. Présentation du projet

L'objectif est de construire une **enceinte Bluetooth portable** capable de lire de la musique depuis un téléphone via Bluetooth, avec une bonne qualité sonore, le tout alimenté par une batterie LiPo rechargeable.

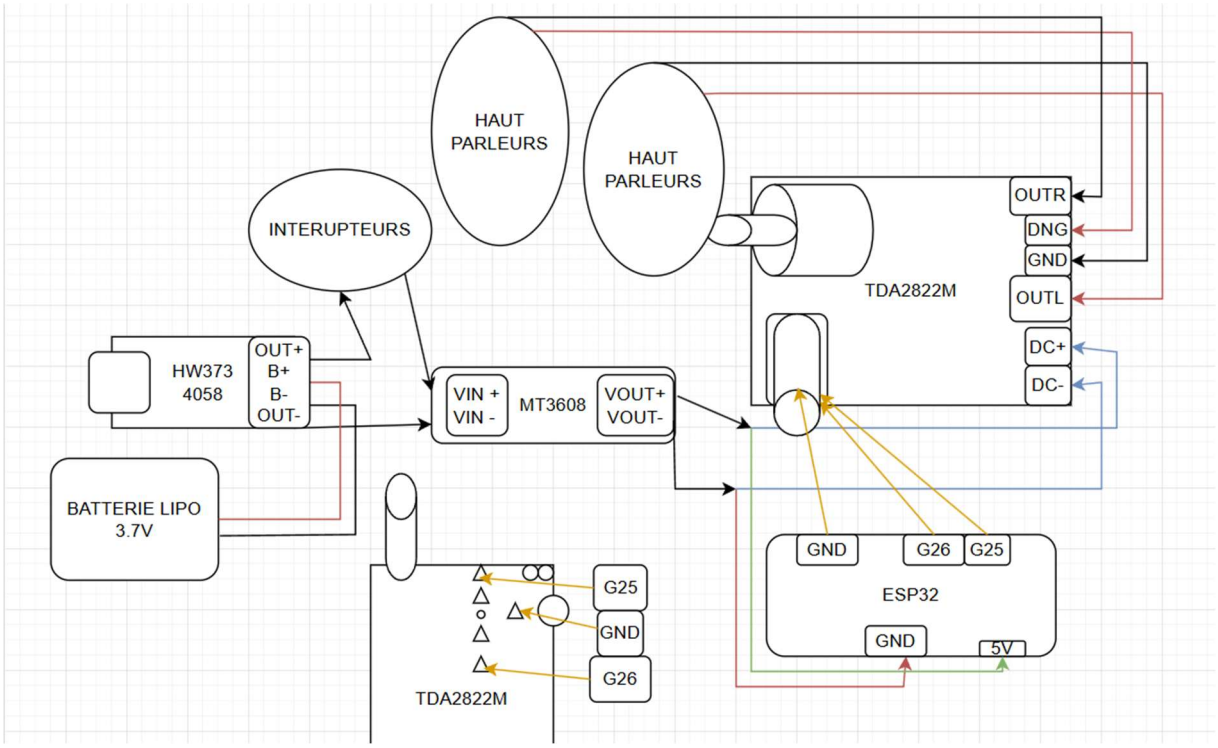
Le cœur du système est une **carte ESP32** qui reçoit le son via Bluetooth et le transmet à deux amplificateurs **TDA2822M** pour alimenter les haut-parleurs gauches et droit.

---

2. Matériel et rôle de chaque composant

Composant	Rôle	Pourquoi ce choix
Batterie LiPo 3,7 V	Source d'énergie du système	Compacte, légère, grande capacité pour un petit volume, idéale pour appareils portables
Module HW-373 4056	Charge et protection de la batterie	Protège la batterie contre les surcharges et décharges profondes + permet la recharge via micro-USB
Interrupteur	Coupe ou établit l'alimentation générale	Permet d'allumer/éteindre l'enceinte facilement sans débrancher la batterie
MT3608 réglé à 5 V	Convertit 3,7 V → 5 V	Alimente l'ESP32 avec une tension stable
ESP32	Réception Bluetooth et sortie audio	Carte compacte avec Bluetooth intégré, capable de convertir et envoyer un signal audio
TDA2822M	Amplificateurs audio	Peu chers, efficaces pour alimenter des haut-parleurs de taille moyenne
Haut-parleurs (G/D)	Restitution du son	Permettent de profiter d'un son stéréo (son à Gauche et à Droite)

Schéma électrique et logique du câblage



Le code Arduino

Maintenant nous allons passer au code qui ira sur la carte ESP32.
Il est tout simple et très concis.
Le voici :

sketch_nov15a.ino

```
1   #include "AudioTools.h"
2   #include "BluetoothA2DPSink.h"
3
4   AnalogAudioStream out;
5   BluetoothA2DPSink a2dp_sink(out);
6
7   void setup() {
8     Serial.begin(115200);
9     a2dp_sink.start("MonEnceinteESP32");
10  }
11
12  void loop() {
13  }
14
```

Comment ça marche ?

Ce programme transforme notre ESP32 en **récepteur Bluetooth audio**, un peu comme une petite enceinte connectée que vous pourriez acheter dans le commerce.
Quand vous connectez votre téléphone à l'ESP32 et que vous lancez une chanson, la musique est envoyée **sans fil** et l'ESP32 la transmet aux haut-parleurs.

Explication pas à pas

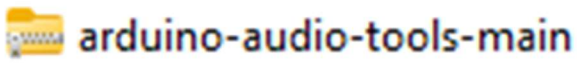
```
1   #include "AudioTools.h"
```

On dit à l'ESP32 :

"Je veux utiliser une boîte à outils pour travailler avec le son."

La bibliothèque **AudioTools** est comme un **kit de Lego spécial pour l'audio**. Elle contient plein de fonctions toutes prêtes pour gérer le son sans qu'on ait à tout programmer nous-mêmes.

INSTALLER : arduino-audio-tools-main FICHIER .ZIP = Sketch, Include Librery, Add.ZIP Librery.

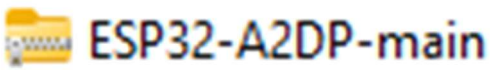



```
2   #include "BluetoothA2DPSink.h"
```

On ajoute une autre boîte à outils, mais cette fois pour le **Bluetooth**.

- **Bluetooth** : technologie qui permet de transférer des données (ici, de la musique) **sans fil** entre deux appareils.
- **A2DP** (*Advanced Audio Distribution Profile*) : c'est un "profil Bluetooth" qui définit **comment envoyer du son stéréo de bonne qualité** d'un appareil (comme un téléphone) vers un autre (comme notre ESP32).
- **Sink** (en anglais "lavabo" ou "récepteur") : ça veut dire que l'ESP32 **reçoit** le son, il ne l'envoie pas.

INSTALLER : ESP32-A2DP-main FICHIER .ZIP = Sketch, Include Librery, Add.ZIP Librery.



Exemple concret :

- A2DP = le langage commun que votre téléphone et votre enceinte utilisent pour se comprendre.
- Sans ce profil, le son serait haché, déformé ou inexistant.

```
4   AnalogAudioStream out;
```

Ici, on crée une “sortie audio analogique” pour l’ESP32.
Cela veut dire qu’on va convertir les données numériques reçues en Bluetooth en un signal électrique que les amplis et haut-parleurs peuvent comprendre.

- Cette conversion se fait grâce à un DAC (*Digital to Analog Converter* ou convertisseur numérique-analogique).
- L’ESP32 possède un DAC intégré relié aux broches GPIO 25 (canal gauche) et GPIO 26 (canal droit).

Image mentale :

- Données numériques : un fichier MP3 dans votre téléphone → juste une suite de nombres.
- Signal analogique : une onde électrique qui fait vibrer le haut-parleur.
- Le DAC est comme un traducteur qui transforme les “nombres” en “vibrations sonores”.

```
5   BluetoothA2DPSink a2dp_sink(out);
```

Ici, on dit :

“Le son que je reçois en Bluetooth, envoie-le vers la sortie audio que je viens de créer (out).”

C’est la passerelle entre le Bluetooth et les sorties GPIO 25/26.

```
7   void setup() {
```

C’est ce qui s’exécute au démarrage de l’ESP32.
Un peu comme la mise en route d’un appareil quand tu appuies sur ON.

```
8     Serial.begin(115200);
```

On démarre la **connexion série**.

- C'est un moyen pour l'ESP32 de "parler" à l'ordinateur par le câble USB.
- Ici, ça sert surtout au débogage (afficher des messages techniques pendant que ça fonctionne).

Exemple : quand vous branchez votre téléphone à un PC pour voir les messages d'erreur ou vérifier que tout marche.

```
9     a2dp_sink.start("MonEnceinteESP32");
```

- On **active le Bluetooth**.
- On donne un **nom** à notre enceinte : "MonEnceinteESP32".
- C'est ce nom que vous verras apparaître quand vous chercherez un appareil Bluetooth sur votre téléphone.

```
10  }
11
12  void loop() {
13  }
14
```

- La "boucle principale" du programme est vide.
- Ici, on n'a pas besoin d'écrire quoi que ce soit car **la bibliothèque s'occupe toute seule de recevoir et jouer la musique** en arrière-plan.

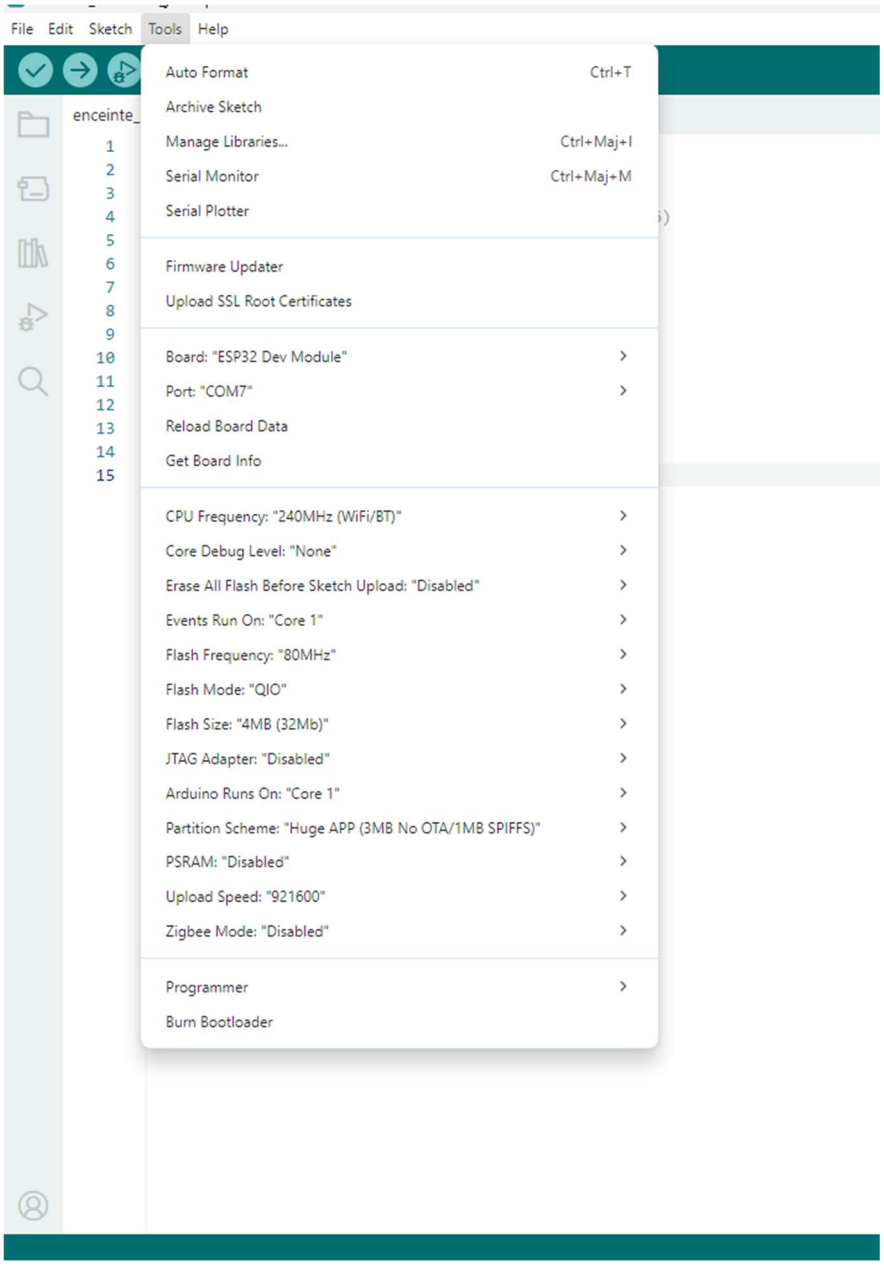
Résumé simple

1. L'ESP32 **s'allume**.
2. Il **active le Bluetooth** et se montre comme "MonEnceinteESP32".
3. Votre téléphone se connecte et **envoie la musique**.
4. L'ESP32 **transforme les données numériques** en un signal électrique grâce au **DAC**.
5. Ce signal part vers les amplis TDA2030A, qui le **grossissent** pour faire bouger les membranes des haut-parleurs.
6. **La musique sort**.

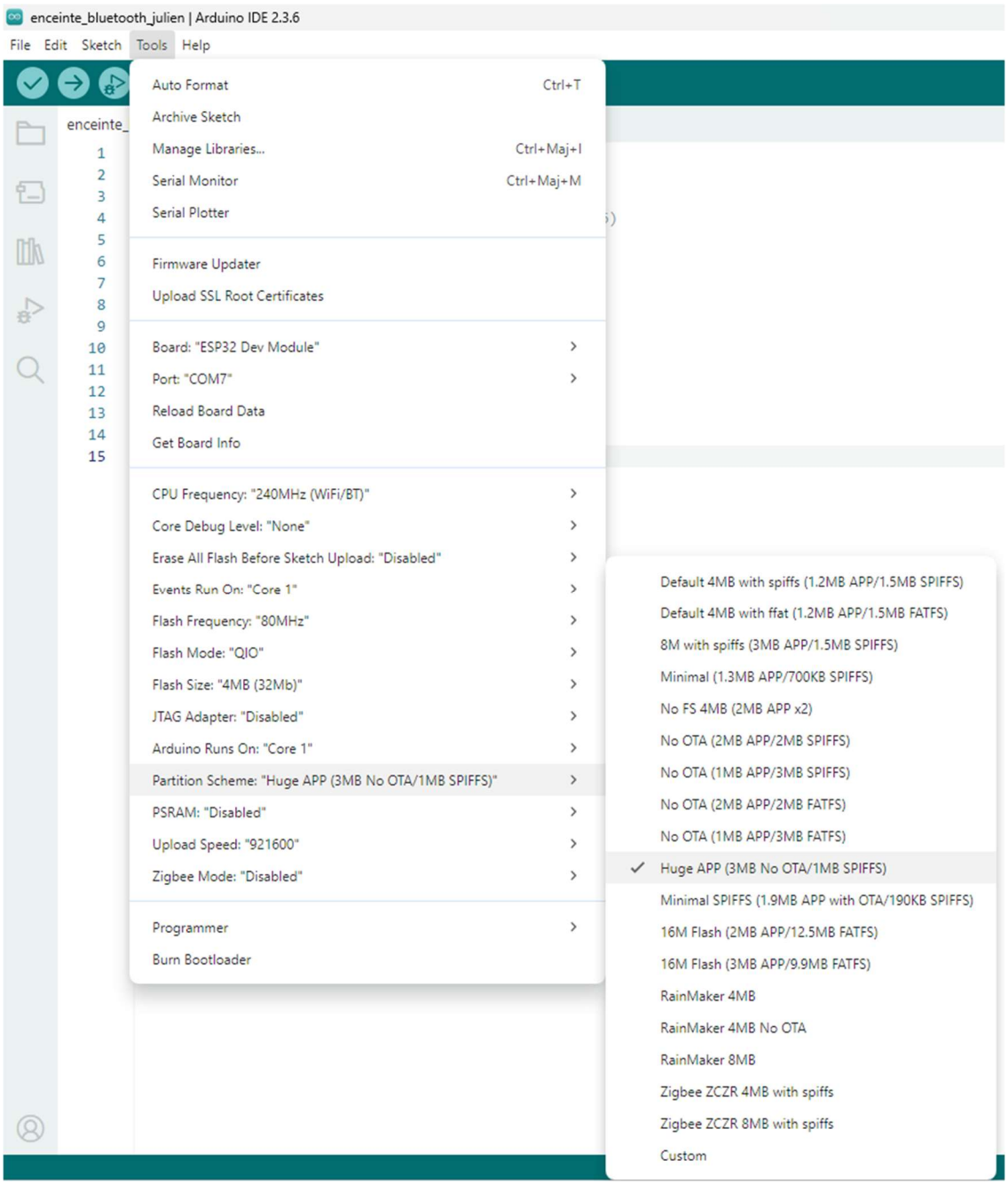
Maintenant que le code est bon, que nous avons tout vérifier : que ce soit les fautes dans le code s'il y en a, le câblage de notre enceinte et que notre ESP32 est branché au PC, nous allons pouvoir envoyer le code dans la carte.
Cependant, si vous le faites maintenant, Arduino IDE va vous faire un message d'erreur mentionnant que la carte ne possède pas assez de mémoire pour le code.

Pas de panique, pour remédier à ce problème nous allons simplement changer le schéma de partition du code. C'est très simple :

Premièrement vous allez cliquer sur le bouton « Tools » ou « outils » en anglais



Ensuite, vous allez dans le menu « Partition Scheme » en bas du menu déroulant et sélectionner « Huge App (3MB No OTA/1MB SPIFFS) » :



Et voilà, vous avez finit le projet, vous n'avez plus qu'à envoyer le code dans la carte ESP 32, ne pas oublier de maintenir le bouton « BOOT » sur la carte lors de l'envoie pour lui permettre de recevoir le code.
Vous pourrez ensuite vous connecter à l'ESP32 sur votre téléphone et profiter de votre enceinte bluetooth.