

## Mode d'Utilisation : Système Intelligent de Pilotage de Véhicule

Groupe projet M2



Réalisé par :

- **Fabien GUNST**
- **Eliott GILET**
- **Melissa BRAHIMI**

Tuteur en Université et Client :  
**BOEGLER Hervé**

## Table des matières

1.	Introduction .....	4
2.	Microphone intelligent STM32 (STM32 F746G DISCOVERY) .....	5
a)	Configuration sur cubeMX : .....	5
System Core : .....	5	
Connectivity .....	5	
Multimedia .....	8	
Computing .....	9	
Retour sur System Core .....	9	
Middleware and Softwares Packs .....	10	
b)	Configuration sur CubeIDE : .....	15
3.	Configuration voiture (Seeed Xiao nRF52840) .....	20
a)	Côté matériels .....	20
b)	Côté logiciel .....	25
ANNEXE 1 :	Caractéristiques Techniques des composants .....	27
Microcontrôleur : Seeed Studio XIAO nRF52840 .....	27	
SyRen10 : Contrôleur de moteur .....	27	
DAC MCP4725 : Convertisseur numérique-analogique .....	28	
NRF24L01 : Communication Radio .....	28	
ML2596s : Régulateur de tension à découpage Step-Down (Buck Converter) .....	29	
TECHNIPLUS T2M AS-12 : Servomoteur analogique .....	29	
Moteur : Max Power 450 Electric Brushed Motor .....	30	
ANNEXE 2 :	Modèle IA utilisé .....	31

## Table des figures

Figure 1 : Configuration SYS.....	5
Figure 2 : Configuration RCC.....	5
Figure 3 : Configuration SDMMC1 .....	5
Figure 4 : Configuration DMA de SDMMC1 .....	6
Figure 5: Configuration USART1 .....	6
Figure 6 : Configuration SPI2.....	7
Figure 7 : Configuration FMC.....	7
Figure 8 : Configuration SAI1.....	8
Figure 9 : Configuration DMA SAI1 .....	8
Figure 10 : Configuration CRC.....	9
Figure 11 : Vérification DMA.....	9
Figure 12 : Configuration GPIO .....	9
Figure 13 : Configuration NVIC .....	10
Figure 14 : Configuration FATFS .....	10
Figure 15 : Configuration DMA FATFS .....	10
Figure 16 : Activation CMSIS .....	11
Figure 17 : Activation CORE et DSP CMSIS .....	11
Figure 18 : Importation du pack ARM CMSIS .....	11
Figure 19 : Activation X-CUBE-AI .....	13
Figure 20 : Configuration X-CUBE-AI .....	13
Figure 21 : Résultats "Analyse" du modèle.....	14
Figure 22 : Configuration "project manager" .....	14
Figure 23 : Arborescence driver BSP .....	15
Figure 24 : Arborescence driver wave_audio .....	15
Figure 25 : Arborescence driver CMSIS .....	16
Figure 26 : Arborescence driver nRF24L01 .....	16
Figure 27 : Intégration des chemins des drivers au modèle.....	16
Figure 28 : Intégration du chemin de la librairie math CMSIS .....	17
Figure 29 : Intégration de la librairie math CMSIS .....	17
Figure 30 : Intégration de la compilation de la librairie ARM MATH.....	18
Figure 31 : Exclusion et mise en commentaire des dossiers/fichiers non nécessaires .....	18
Figure 32 : Arborescence projet .....	19
Figure 33 : Véhicule utilisé .....	20
Figure 34 : Schéma d'intégration des composants dans la voiture .....	20
Figure 35 : Composants du BCP.....	22
Figure 36 : Câblage BCP .....	22
Figure 37 : Schéma câblage voiture.....	23
Figure 38 : Schéma électrique voiture .....	23
Figure 39 : Schéma câblage sans PCB.....	24
Figure 40 : installation de la board seeed nrf52 .....	25
Figure 41 : Sélection de la carte d'utilisation .....	25
Figure 42 : Exemple installation librairie .....	26
Figure : Capture écran introduction code exemple .....	31

# 1. Introduction

Ce document a pour vocation de servir de guide d'utilisation pour les futures promotions d'étudiants de l'Université de Poitiers souhaitant comprendre et reprendre notre projet, réalisé dans le cadre de notre Master 2 en Objets Connectés et intitulé « Pilotage intelligent d'un véhicule ».

L'objectif principal de ce projet était d'explorer et de mettre en œuvre une application d'intelligence artificielle embarquée en testant différents microcontrôleurs et technologies d'IA. Partant de cet objectif et après une phase de recherche documentaire et d'expérimentation sur les technologies disponibles, nous avons choisi de développer un véhicule intelligent contrôlé par la voix.

L'idée du projet est née d'une recherche et d'analyse de projets existants, notamment sur des plateformes de partage comme Hackaday.io, où nous avons trouvé des concepts de pilotage de véhicules via radio-contrôle. S'inspirant de ces approches, nous avons opté pour une architecture basée sur deux microcontrôleurs communiquant via des antennes radio. L'un d'eux joue le rôle d'un système intelligent de reconnaissance vocale, c'est-à-dire qu'il fait office de microphone avec une couche IA capable d'enregistrer, analyser et interpréter une commande vocale avant de la transmettre au second microcontrôleur. Ce dernier est chargé de contrôler le déplacement du véhicule en fonction des instructions reçues. Ce choix avait pour ambition non seulement d'apprendre à utiliser les systèmes d'IA embarquée, mais aussi proposer une approche ludique et interactive pour l'appliquer.

Sur ce document, nous allons voir comment configurer le projet du microphone intelligent basé en STM32, et son grafcet, et aussi comment configurer la voiture basée en Seeed Xiao nRF52840, ses composants électriques et son grafcet.

## 2. Microphone intelligent STM32 (STM32 F746G DISCOVERY)

### a) Configuration sur cubeMX :

Créer un nouveau projet sur CubeMX en choisissant la STM32F746G-DISCO comme STBoard, on clear tous les pins actifs par défaut et on commence à faire les configurations suivantes :

#### System Core :

1. Sur **SYS** on choisit **Serial Wire** Comme **Debug** et **SysTick** comme **timebase Source** (Figure 1)

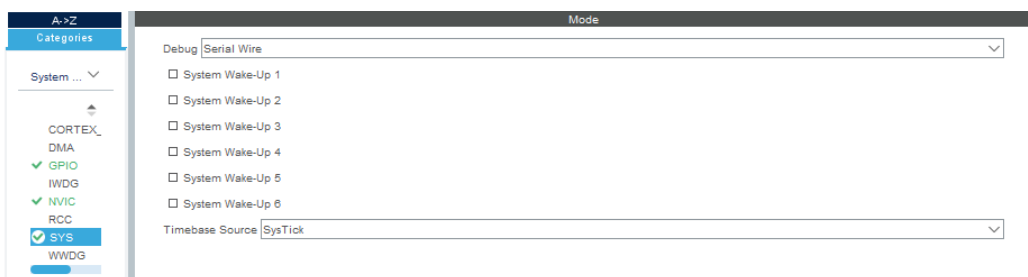


Figure 1 : Configuration SYS

2. Sur **RCC** on choisit **Crystal/Ceramic Resonator** comme High Speed Clock (HSE) (Figure 2)

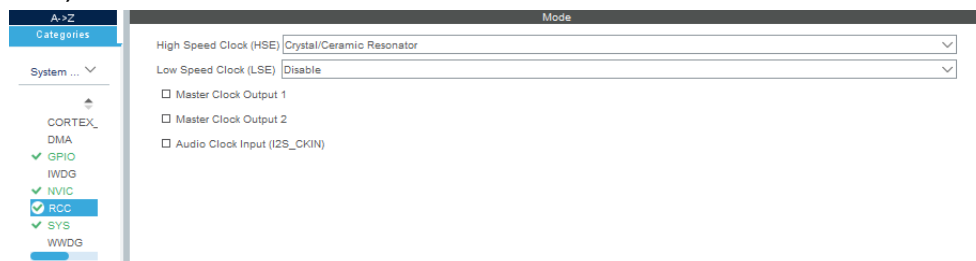


Figure 2 : Configuration RCC

#### Connectivity

1. Sur **SDMMC1** mettre le mode en **SD1 bit** -> nécessaire pour l'utilisation de la carte SD

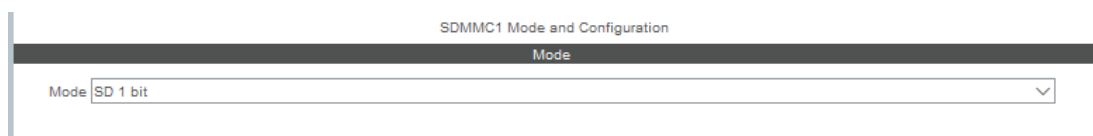


Figure 3 : Configuration SDMMC1

Sur le tab « DMA Settings », cliquez sur « add » et configurez RX et TX de la façon suivante :  
**RX et TX :**

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

DMA Settings

GPIO Settings

DMA Request	Stream	Direction	Priority
SDMMC1_RX	DMA2 Stream 3	Peripheral To Memory	Low
SDMMC1_TX	DMA2 Stream 6	Memory To Peripheral	Low

Add

Delete

DMA Request Settings

Mode

Peripheral Flow Con...

Increment Address

☐

Peripheral

☐

Memory

☒

Use Fifo

☒

Threshold

Full

Data Width

Word

Burst Size

4 Increment

Word

4 Increment

Figure 4 : Configuration DMA de SDMMC1

- Mettre **USART1** en mode **Asynchronous** et s’assurer que les pins **PA9** et **PB7** sont en USART\_TX et USART\_RX respectivement. (Figure 5) -> nécessaire pour l’envoi des messages vers un serial terminal (ex : Putty)

USART1 Mode and Configuration

Mode

Mode

Asynchronous

Hardware Flow Control (RS232)

Disable

☐ Hardware Flow Control (RS485)

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

DMA Settings

GPIO Settings

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull...	Maximum output...	User Label	Modified
PA9	USART1_TX	n/a	Alternate Functio...	No pull-up and n...	Very High		<input type="checkbox"/>
PB7	USART1_RX	n/a	Alternate Functio...	No pull-up and n...	Very High		<input type="checkbox"/>

Figure 5: Configuration USART1

6

3. Mettre **SPI2** en mode **Full Duplex Master** -> nécessaire pour la communication à distance avec le module radio nRF24L01

**SPI2 Mode and Configuration**

**Mode**

Mode: Full-Duplex Master

Hardware NSS Signal: Disable

**Configuration**

Reset Configuration

Parameter Settings

Configure the below parameters :

Search (Ctrl+F)

**Basic Parameters**

Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

**Clock Parameters**

Prescaler (for Baud Rate)	16
Baud Rate	3.125 MBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge

**Advanced Parameters**

CRC Calculation	Disabled
NSSP Mode	Enabled
NSS Signal Type	Software

**Figure 6 : Configuration SPI2**

4. Sur **FMC** on suit les configurations de la figure 7 ci-après pour rajouter de l'espace a notre carte stm32, on aura besoin pour traiter le signal de la partie IA -> nécessaire pour l'utilisation de la SDRAM (plus d'espace de stockage)

**FMC Mode and Configuration**

**Mode**

NOR Flash/PSRAM/SRAM/ROM/LCD 1

NOR Flash/PSRAM/SRAM/ROM/LCD 2

NOR Flash/PSRAM/SRAM/ROM/LCD 3

NOR Flash/PSRAM/SRAM/ROM/LCD 4

NAND Flash 1

**SDRAM 1**

Clock and chip enable: SDCKE0+SDNE0

Internal bank number: 4 banks

Address: 13 bits

Data: 16 bits

Byte enable: Disable

**Configuration**

Reset Configuration

SDRAM 1

Configure the below parameters :

Search (Ctrl+F)

**SDRAM control**

Bank	SDRAM bank 1
Number of column address bits	9 bits
Number of row address bits	13 bits
CAS latency	3 memory clock cycles
Write protection	Disabled
SDRAM common clock	2 HCLK clock cycles
SDRAM common burst read	Disabled
SDRAM common read pipe delay	1 HCLK clock cycle

**SDRAM timing in memory clock cycles**

Load mode register to active delay	2
Exit self-refresh delay	6
Self-refresh time	4
SDRAM common row cycle delay	6
Write recovery time	2
SDRAM common row precharge delay	2
Row to column delay	2

**Figure 7 : Configuration FMC**

## Multimedia

1. Sur **SAI1**, mettre **SAI A** en mode **master** et reprendre les paramètres de la figure 8 dans **Parameter Settings** -> **nécessaire pour l'utilisation du microphone intégré du microcontrôleur**.

SAI1 Mode and Configuration

Mode

SAI A

Mode: Master

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

SAI A

Synchronization Inputs: Asynchronous

Basic Parameters

Protocol: Free

Audio Mode: Master Receive

Frame Length: 16 bits

Data Size: 16 Bits

Slot Size: DataSize

Output Mode: Stereo

Companding Mode: No companding mode

Frame Parameters

First Bit: MSB First

Frame Synchro Active Level Length: 1

Frame Synchro Definition: Start Frame

Frame Synchro Polarity: Active Low

Frame Synchro Offset: First Bit

Slot Parameters

First Bit Offset: 0

Number of Slots: 1

Slot Active Final Value: 0x00000000

Slot Active: Neither

Clock Parameters

Master Clock Divider: Enabled

Audio Frequency: 16 KHz

Real Audio Frequency: 15.789 KHz

Error between Selected: -1.31 %

Clock Strobing: Falling Edge

Advanced Parameters

Fifo Threshold: Empty

Output Drive: Disabled

**Figure 8 : Configuration SAI1**

Ensuite, cliquez sur la tab « DMA Settings » et configurez de la façon suivante (Figure 9) :

Mode

SAI A

SAI B

External Synchro Out

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

DMA Request	Stream	Direction	Priority
SAI1_A	DMA2 Stream 1	Peripheral To Memory	High

Add Delete

DMA Request Settings

Mode: Circular

Increment Address: ☒

Peripheral: ☐ Memory: ☒

Use Fifo: ☐ Threshold:

Data Width: Half Word

Burst Size:

**Figure 9 : Configuration DMA SAI1**



## Computing

### 1. Activer **CRC**

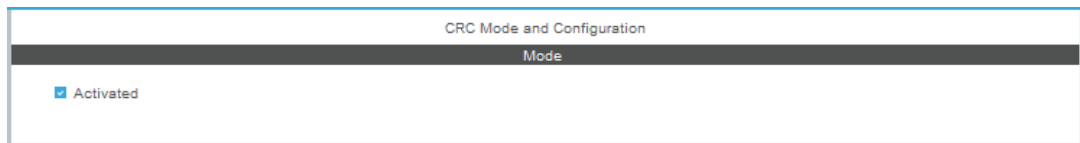


Figure 10 : Configuration CRC

## Retour sur System Core

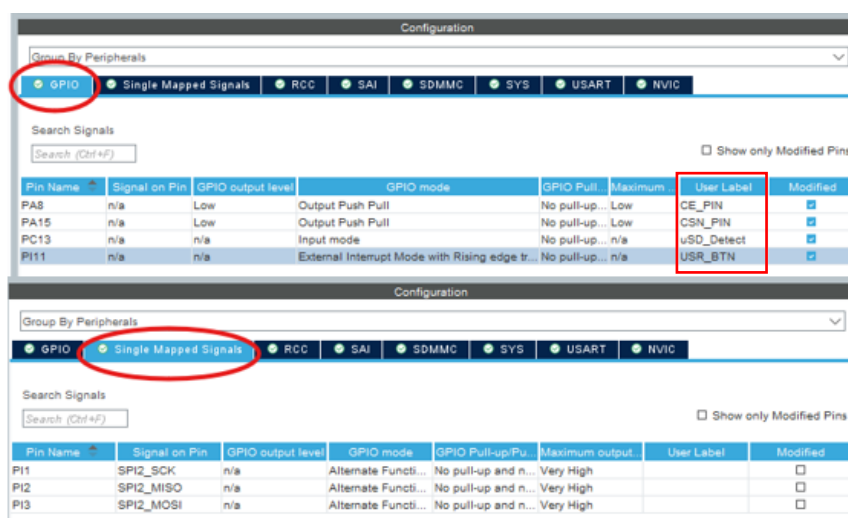
Sur la tab DMA2, nous devrions voir le résultat suivant (Figure 11) :

DMA Request	Stream	Direction	Priority
SAI1_A	DMA2 Stream 1	Peripheral To Memory	High
SDMMC1_RX	DMA2 Stream 3	Peripheral To Memory	Low
SDMMC1_TX	DMA2 Stream 6	Memory To Peripheral	Low

Figure 11 : Vérification DMA

### 1. Sur **GPIO** on cherche et configure les pins suivants :

- **PI1** → SCK de SPI (s'assurer que **SCK** de SPI soit bien sur la pin PI1 et non PD3 de la configuration par défaut)
- **PI2** → MISO de SPI
- **PI3** → MOSI de SPI
- **PA8** → CE\_PIN : CE de SPI (mettre la pin en mode GPIO-OUTPUT)
- **PA15** → CSN\_PIN : CSN de SPI (mettre la pin en mode GPIO-OUTPUT)
- **PC13** → uSD\_Detect : pour la carte SD (mettre la pin en mode GPIO-INPUT)
- **PI11** → USR\_BTN : pour l'utilisation du bouton bleu de la STM32 (pin en mode GPIO-EXTI11)



Attention :  
Les user Label doivent  
être correctement écrit  
pour être cohérents avec  
le code

Figure 12 : Configuration GPIO

2. Sur **NVIC** on fait les configurations des priorités d'interruptions suivant la figure 13 qui suit, et cocher les cases de la colonne « Enabled » où c'est le cas :

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input checked="" type="checkbox"/>	0	0
Flash global interrupt	<input checked="" type="checkbox"/>	0	0
RCC global interrupt	<input checked="" type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	0	0
FMC global interrupt	<input checked="" type="checkbox"/>	0	0
SDMMC1 global interrupt	<input checked="" type="checkbox"/>	1	0
DMA2 stream1 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA2 stream3 global interrupt	<input checked="" type="checkbox"/>	2	0
DMA2 stream0 global interrupt	<input checked="" type="checkbox"/>	2	0
FPU global interrupt	<input checked="" type="checkbox"/>	0	0
SAI1 global interrupt	<input checked="" type="checkbox"/>	0	0

Figure 13 : Configuration NVIC

## Middleware and Softwares Packs

1. Sur **FATFS** cocher **SD Card** et assigner la pin **PC13** qu'on a configuré auparavant :

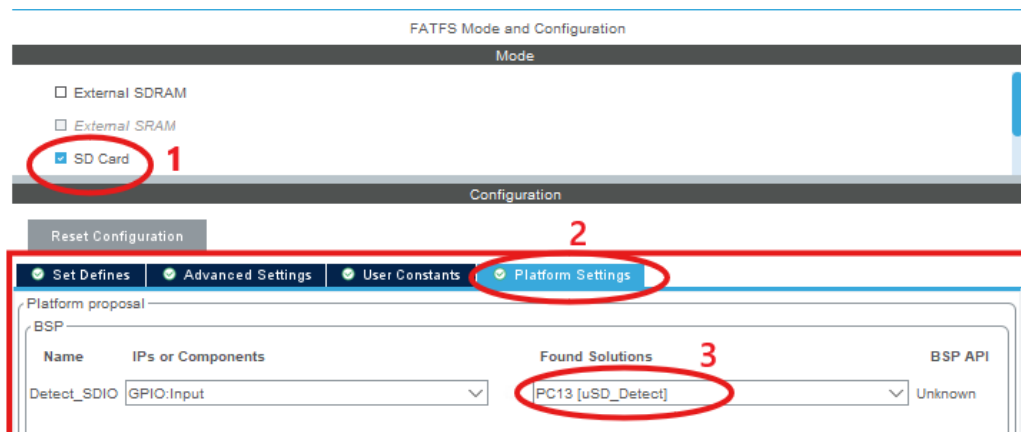


Figure 14 : Configuration FATFS

Ensuite sur **Advanced Settings** on active l'usage de DMA :

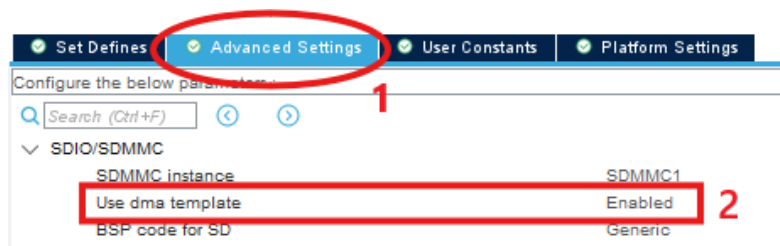


Figure 15 : Configuration DMA FATFS

## 2. Activer dans **CMSIS : CMSIS Core** et **CMSIS DSP**

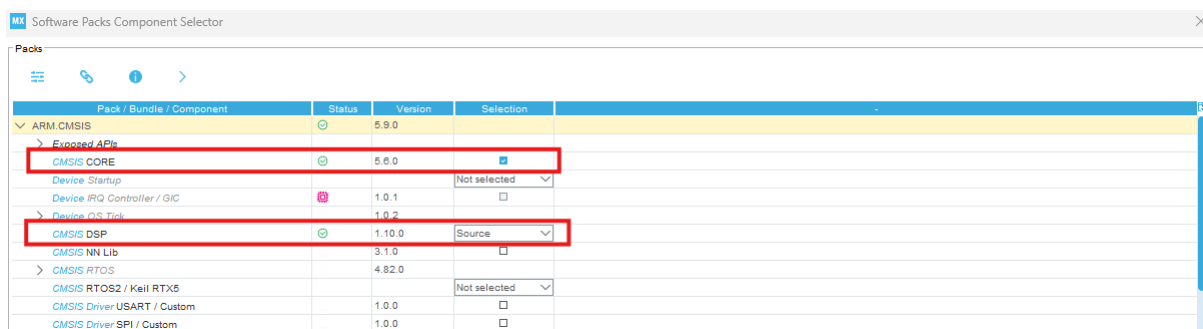


Figure 16 : Activation CMSIS

Après avoir cliqué sur **OK** faut les cocher pour les activer, sur **CMSIS** toujours :

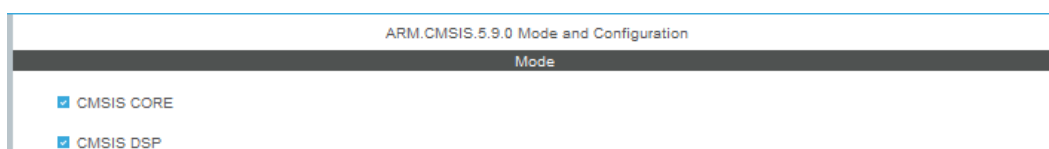


Figure 17 : Activation CORE et DSP CMSIS

NB : Si **CMSIS** n'est pas dans **Middleware and Softwares Packs** par défaut il faut télécharger le pack « **ARM.CMSIS.5.9.0.pack** » sur internet puis suivre les étapes suivantes pour l'installer sur CubeMX :

On va sur le **Help** de CubeMx >> **Manage Embedded Software Package**>> cliquer sur « **from local** » et choisir le pack installer (Figure18) :

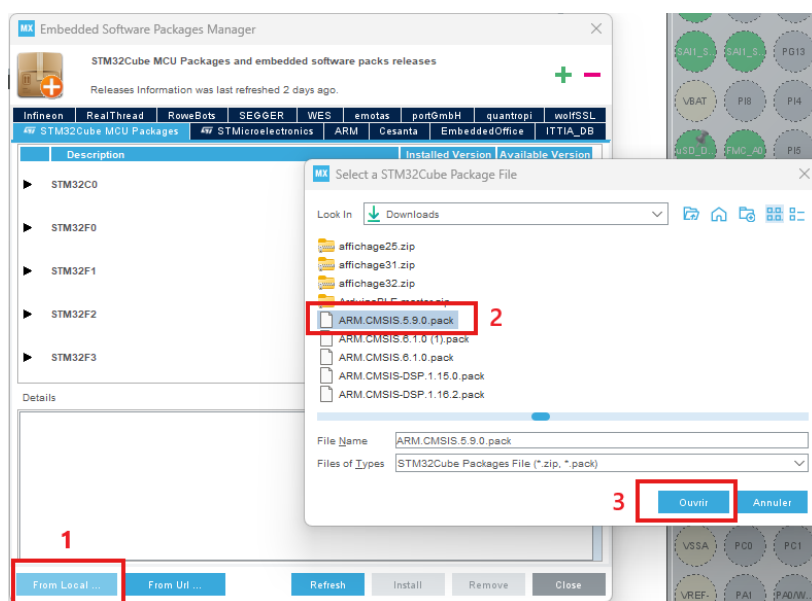


Figure 18 : Importation du pack ARM CMSIS

S'assurer que le heap et stack sont à 2000 :

Linker Settings	
Minimum Heap Size	0x2000
Minimum Stack Size	0x2000

Figure 19 : config de stack et heap

Copier cette config de clock :

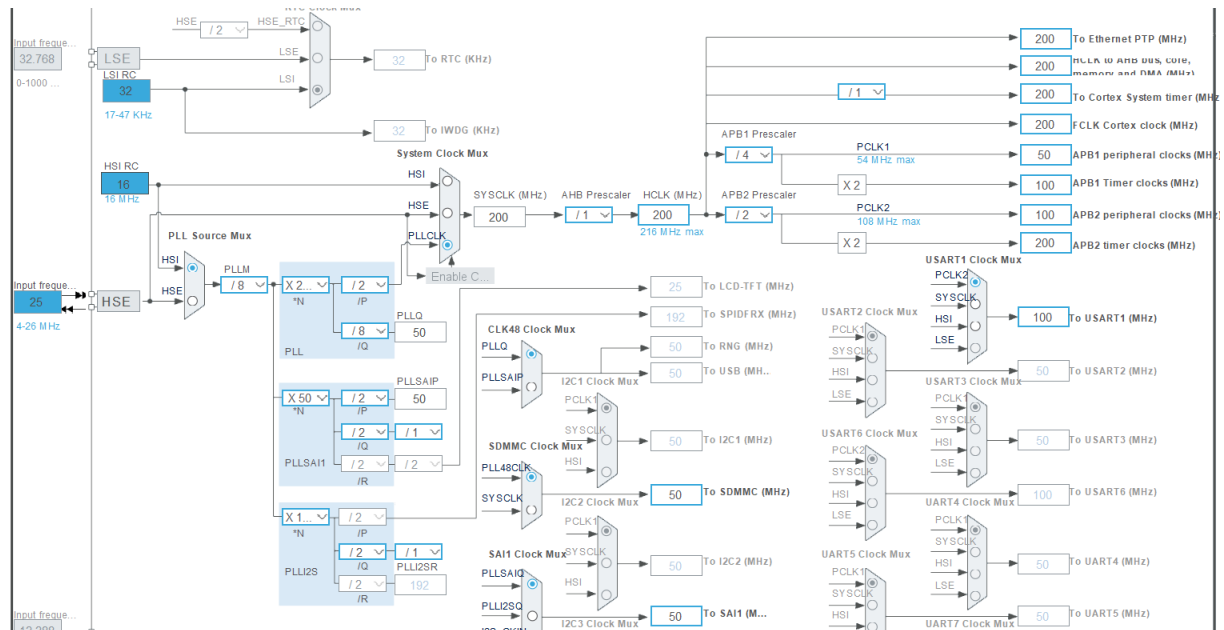


Figure 20 : Config horloge STM32

Et finalement vérifier la bonne activation d'interruptions dans NVIC :

EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	0
FMC global interrupt	<input type="checkbox"/>	0
SDMMC1 global interrupt	<input checked="" type="checkbox"/>	1
DMA2 stream1 global interrupt	<input checked="" type="checkbox"/>	0
DMA2 stream3 global interrupt	<input checked="" type="checkbox"/>	2
DMA2 stream6 global interrupt	<input checked="" type="checkbox"/>	2
FPU global interrupt	<input type="checkbox"/>	0
SAI1 global interrupt	<input type="checkbox"/>	0

Figure 21 : Activation d'interruptions

Ensuite sur **Software Packs >> Select composants >>** Sélectionner **CMSIS Core** et **CMSIS DSP**, les activer suivant les étapes de la figure 16 et 17 faites auparavant.

### 3. Sur **X-CUBE-AI** activer **Artificial Intelligence**.

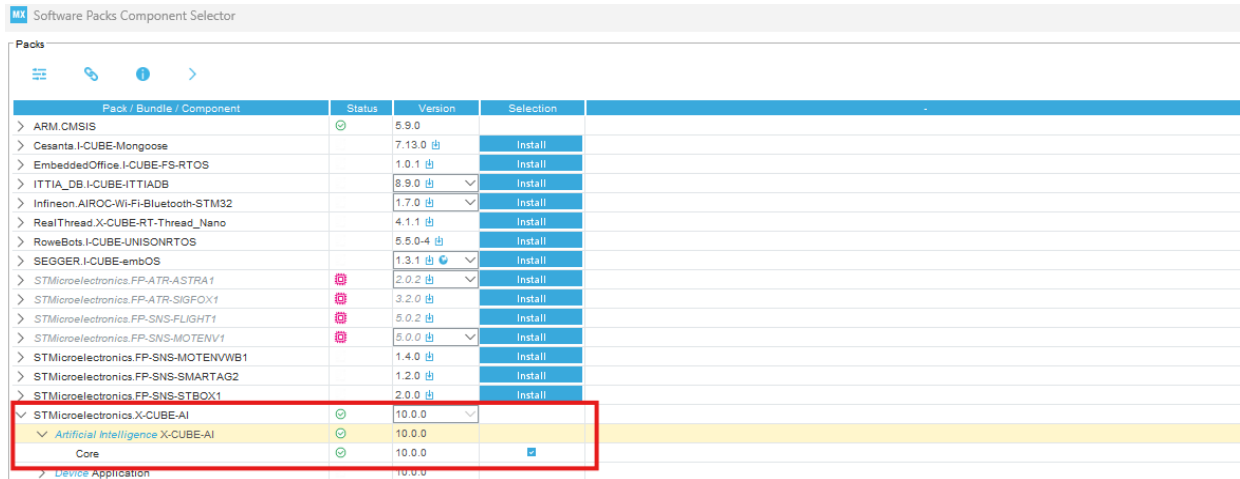


Figure 22 : Activation X-CUBE-AI

Ensuite toujours sur **X-CUBE-AI >> « + » >>** choisir **TFLite** comme modèle et ajouter notre modèle IA déjà entraîné ( se référer à l'annexe 2 pour plus d'information sur le modèle IA):

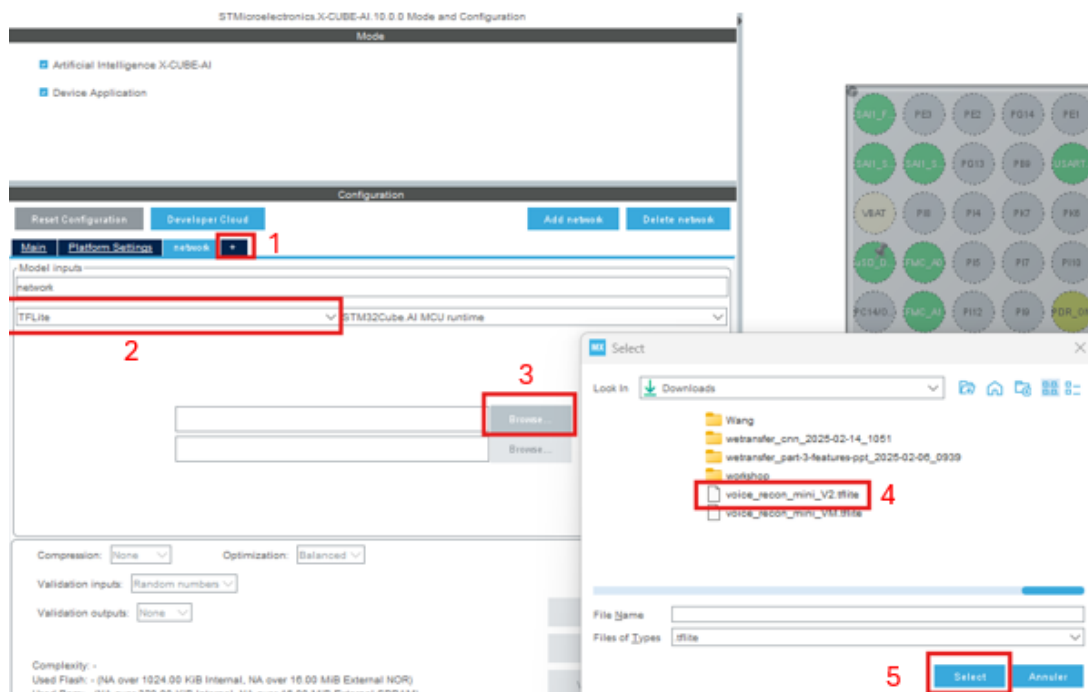
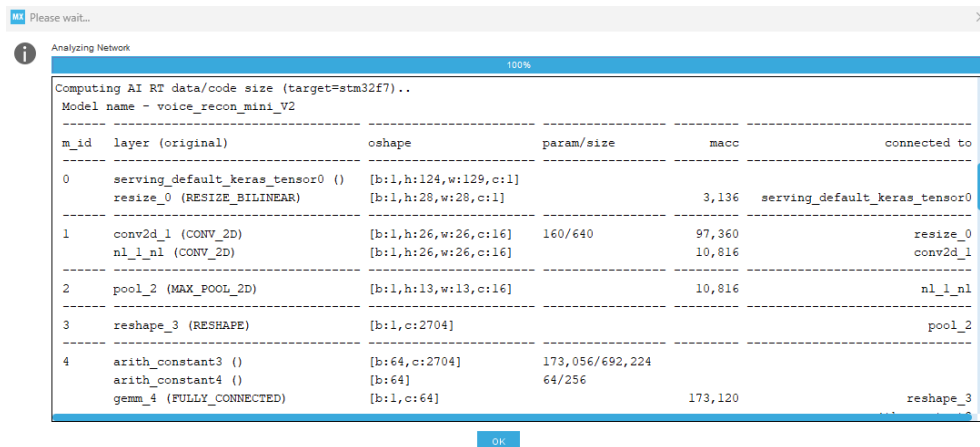


Figure 23 : Configuration X-CUBE-AI

Puis on clique sur **Analyze** pour confirmer que le modèle IA est compatible avec notre microcontrôleur, et fonctionnel.

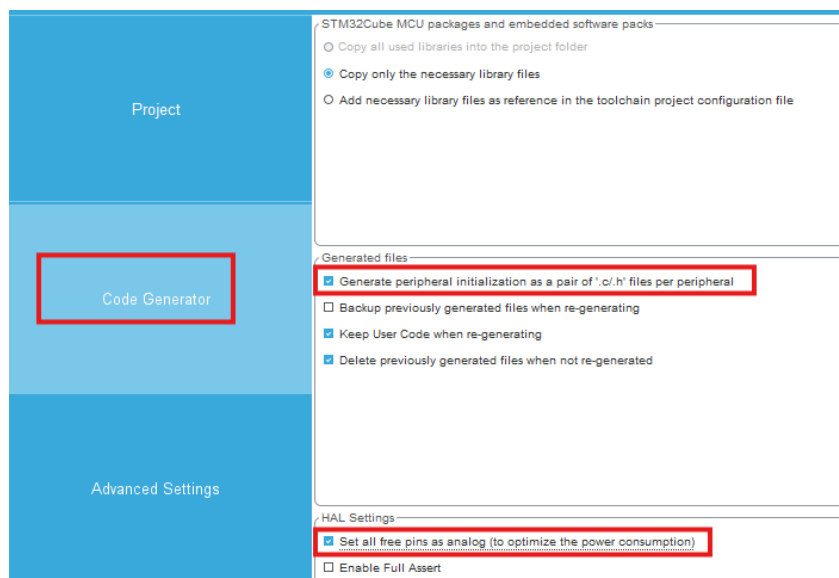


m_id	layer (original)	oshape	param/size	macc	connected to
0	serving_default_keras_tensor0 ()	[b:1,h:124,w:129,c:1]			
	resize_0 (RESIZE_BILINEAR)	[b:1,h:28,w:28,c:1]		3,136	serving_default_keras_tensor0
1	conv2d_1 (CONV_2D)	[b:1,h:26,w:26,c:16]	160/640	97,360	resize_0
	nl_l_nl (CONV_2D)	[b:1,h:26,w:26,c:16]		10,816	conv2d_1
2	pool_2 (MAX_POOL_2D)	[b:1,h:13,w:13,c:16]		10,816	nl_l_nl
3	reshape_3 (RESHAPE)	[b:1,c:2704]			pool_2
4	arith_constant3 ()	[b:64,c:2704]	173,056/692,224		
	arith_constant4 ()	[b:64]	64/256		
	gemm_4 (FULLY_CONNECTED)	[b:1,c:64]		173,120	reshape_3

**Figure 24 : Résultats "Analyze" du modèle**

4. **NB :** si **X-CUBE-AI** n'est pas installé par défaut faut aller dans **Software Packs >> Select components >> installer STMicroelectronics X-CUBE-AI**, activer **Artificial Intelligence**, ensuite suivre les mêmes étapes citées précédemment.

On arrive à la fin des configurations sur CubeMX, il reste plus qu'à donné un nom à notre projet choisir **STM32CubeIDE** comme **IDE** on va sur **Code Generator** on coche les paramètres suivant la figure 22 ci-dessous



**Figure 25 : Configuration "project manager"**

En dernier on clique sur **Generate code** et choisir **Open Folder** comme option pour ouvrir le projet sur **CubeIDE**

**ATTENTION :** L'activation de CubeIA a tendance d'enlever du projet le fichier « syscalls.c » qui se trouve dans le même dossier que le main.c quand le code est généré à partir de CubeMX. Si c'est le cas, il est important de remettre le fichier dans le dossier, sinon les fonctionnalités de printf sur le terminal ne fonctionneront pas. Vous pouvez trouver ce fichier dans le même dossier que d'autres projets STM32, faut le copier et le coller dans le votre.

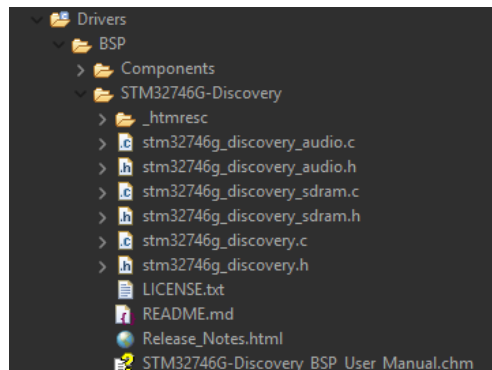
### ***b) Configuration sur CubeIDE :***

Ici on va passer à l'ajout des librairies et fichiers nécessaires pour le fonctionnement de notre projet.

Les librairies nécessaires peuvent être directement copiées depuis notre projet sur le Git suivant : [https://github.com/Projet-M2-GGB/Microphone\\_wav\\_SD](https://github.com/Projet-M2-GGB/Microphone_wav_SD)

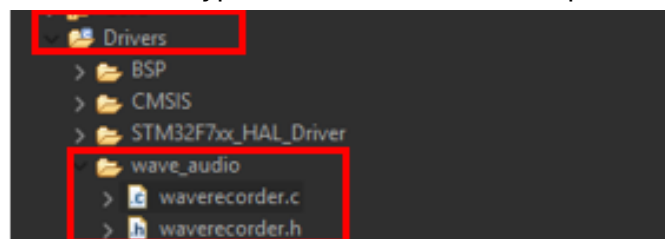
1. Ajouter le dossier **BSP** dans **Drivers** contenant deux autres dossiers **Components** et **STM32746G-Discovery**, ils sont nécessaires pour l'enregistrement audio avec les microphones, et l'utilisation de la SDRAM.

NB : Faut s'assurer que le dossier **STM32746G-Discovery** contient les fichiers suivant (Figure 23) :



**Figure 26 : Arborescence driver BSP**

2. Ajouter dans **Drivers** le dossier **wave\_audio** contenant les fichiers nécessaires pour l'enregistrement de fichiers de type « .wav » avec les microphones intégrés.



**Figure 27 : Arborescence driver wave\_audio**

3. Ajouter dans **Drivers >> CMSIS** le dossier **Lib** contenant les librairies requises pour les calculs nécessaire au traitement de l'audio (calcul de la FFT)

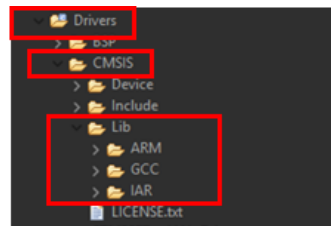


Figure 28 : Arborescence driver CMSIS

4. Ajouter dans **Drivers** le dossier **nrf24L01** contenant les fichiers nécessaires au fonctionnement de l'antenne radio qui transmet les commandes à la voiture

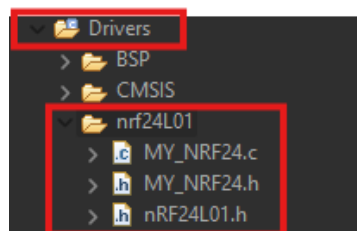


Figure 29 : Arborescence driver nRF24L01

5. Maintenant on doit inclure le chemin des dossiers qu'on vient d'ajouter afin qu'il soit pris en considération, pour ce faire, on fait un clic droit sur le projet >> **Properties >> C/C++ General >> Path and Symbols >> Add >> File system >> choisir le path de chaque dossier ajouter >> Apply and close**, finalement la liste des paths ressemblera à ceci (Figure 27) :

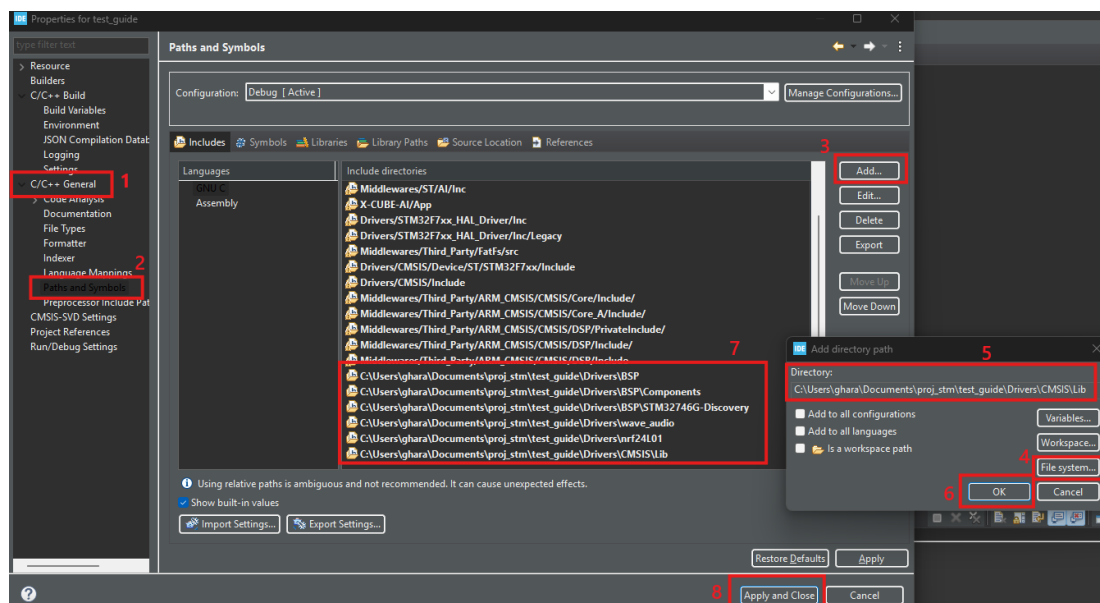


Figure 30 : Intégration des chemins des drivers au modèle



- Ensuite on ajoute le path de la librairie nécessaire pour les calculs mathématique, dans **Proprieties** du projet >> **C/C++ Build** >> **Settings** >> **MCU/MPU GCC Linker** >> **Librairies** >> **Library search path (-L)** >> **Add** >> **File system** >> choisir le path de notre librairie (Figure 28)

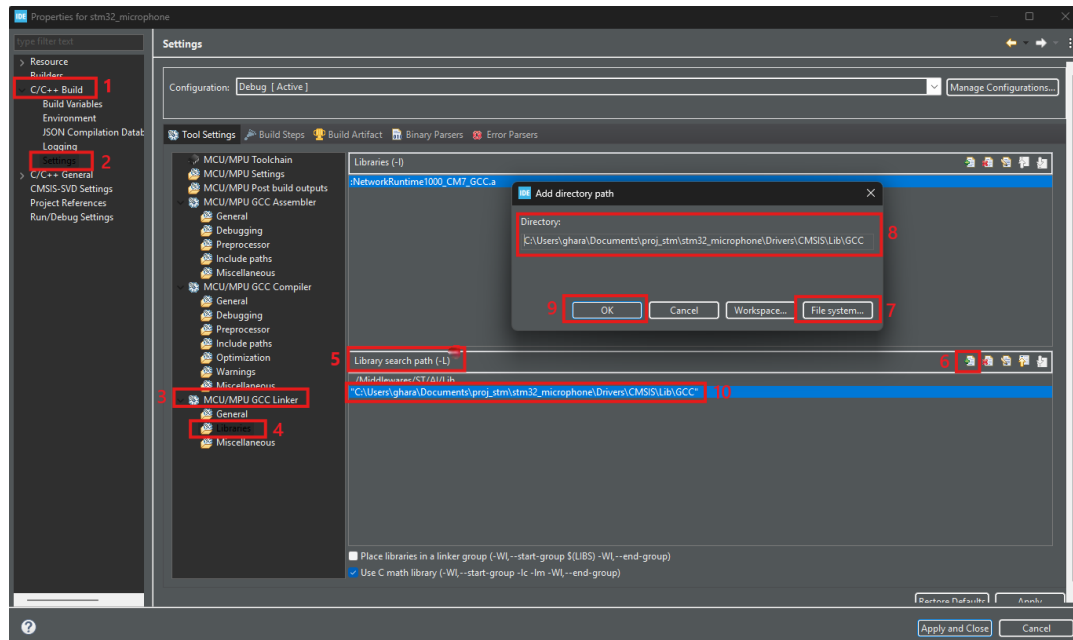


Figure 31 : Intégration du chemin de la librairie math CMSIS

Ensuite dans **Librairies (-l)** >> **Add** >> on ajoute cette librairie « **arm\_cortexM7ldp\_math** » >> **Apply** (Figure 29)

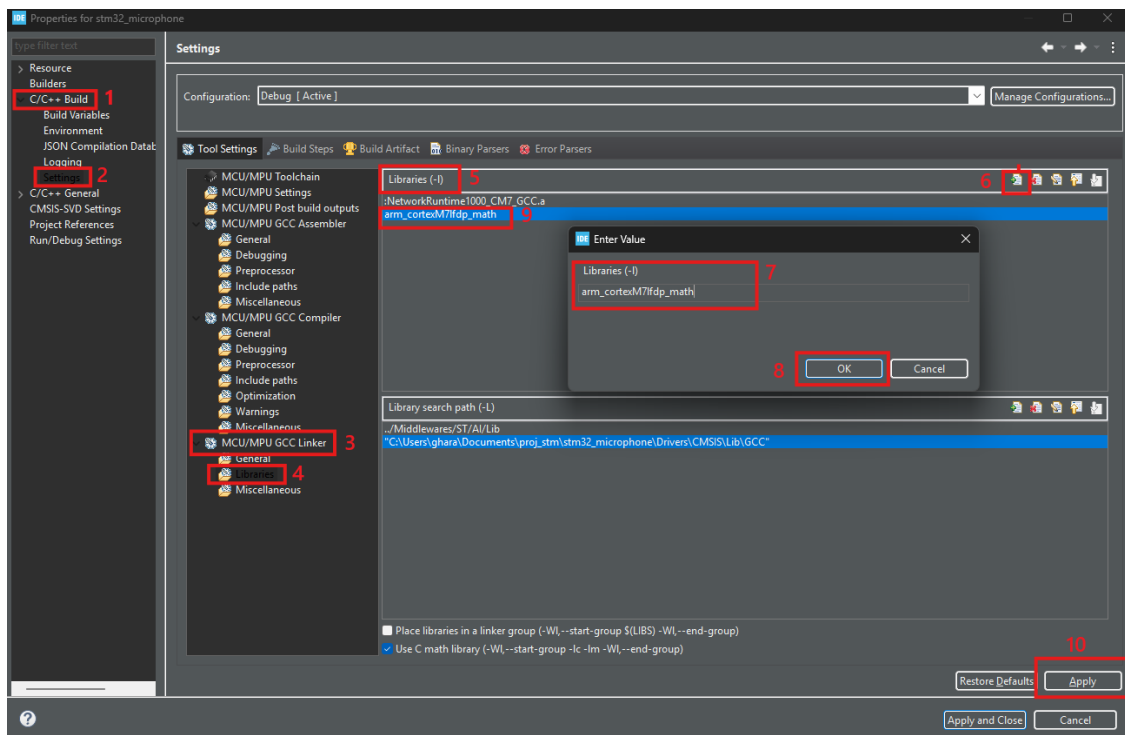


Figure 32 : Intégration de la librairie math CMSIS

Sur **MCU/MPU GCC Compiler >> Preprocessor >> Define Symbols (-D) >> Add >>** on ajoute « **ARM\_MATH\_CM7** » au projet (Figure 30)

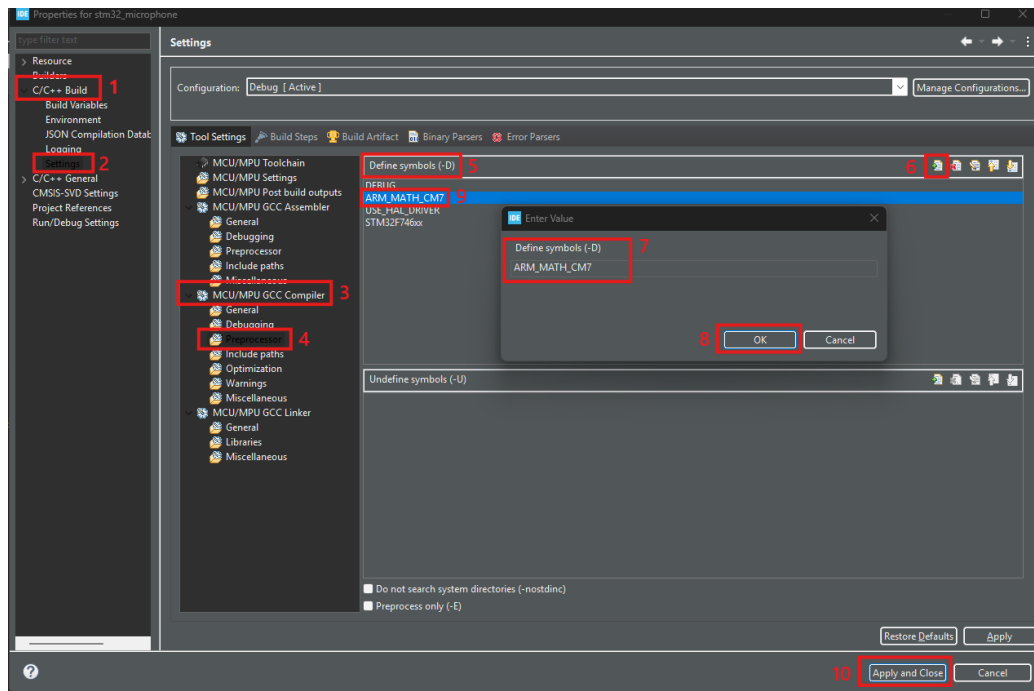


Figure 33 : Intégration de la compilation de la librairie ARM MATH

En dernier, il nous reste plus qu'à bien nettoyer notre librairie pour utiliser que le header du fichier « **arm\_math.h** », pour cela, il faut exclure de la compilation de notre projet quelques dossiers et laisser que « **SupportFunctions** » mais on commente tous les includes dans les fichiers .c de ce dossier (SupportFunctions.c et SupportFunctionsF16.c) suivre les étapes de la figure 31 ci-après

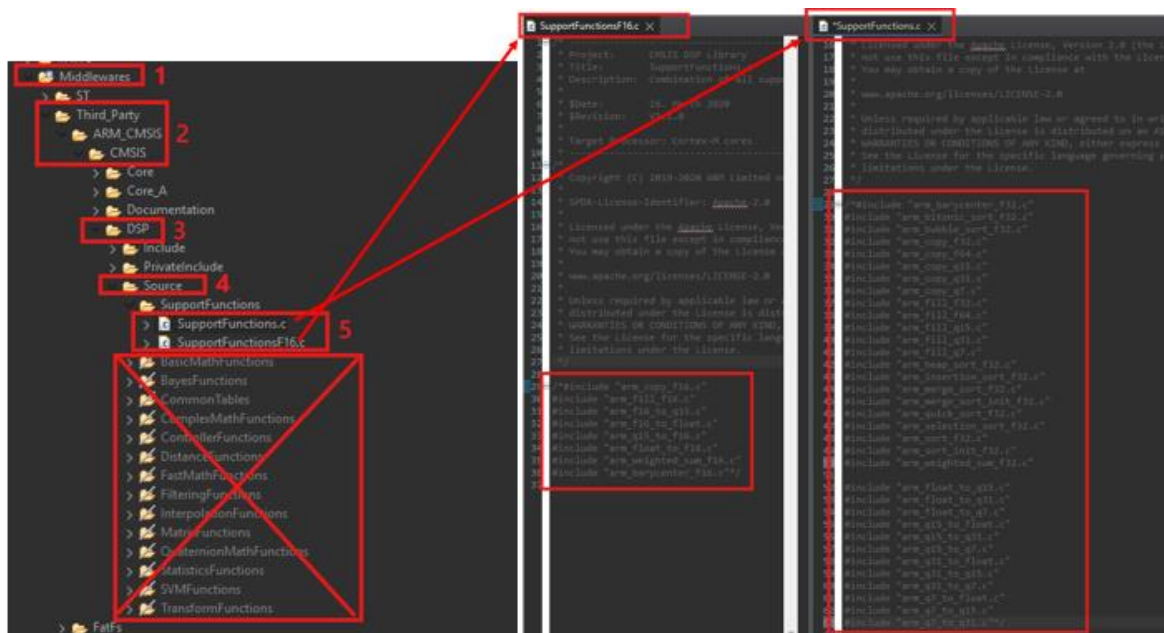
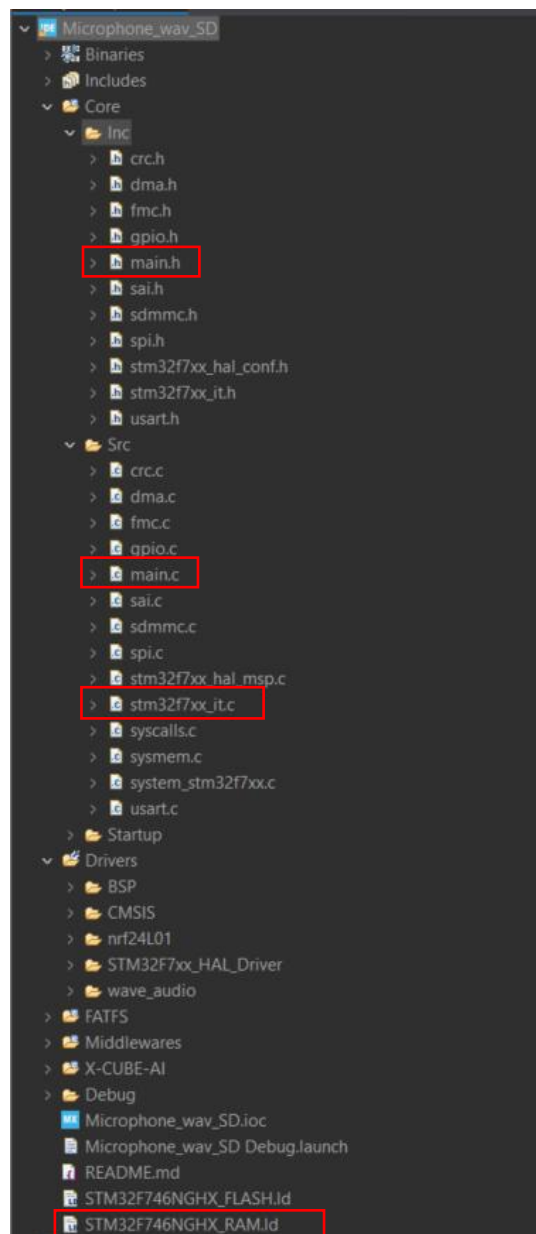


Figure 34 : Exclusion et mise en commentaire des dossiers/fichiers non nécessaires

À présent, toutes les configurations nécessaires pour le projet ont été complétées. Il ne reste plus qu'à copier le code des fichiers suivants de notre projet Git, et les coller sur les mêmes fichiers de votre projet récemment généré :

- « main.c » pour le code principal
- « main.h » pour avoir les ressources utiles qui seront utilisés sur « main.c »
- « stm32f7xx\_it.c » qui a des fonctions nécessaires pour le fonctionnement de l'enregistrement audio
- STM32F746NGHX\_FLASH.ld pour la configuration de la SDRAM



**Figure 35 : Arborescence projet**

Pour consulter la logique du code, consultez le GRAFCET (document dans le dossier ressources utiles de notre git).

### 3. Configuration voiture (Seeed Xiao nRF52840)

#### a) Côté matériels

Cette partie présente le matériel qu'on a utilisé pour construire la voiture, les schémas électriques ainsi que le branchement des composants entre eux.

Comme début, on a repris une voiture déjà servie pour un autre projet de l'université qu'on a démonté et pris uniquement les composants dont a besoin tels que le moteur et le servomoteur, puis nous avons intégré nos propres composants (Figure 34). Le modèle de la voiture est une **Renault Clio, 1994** (Figure 33).



Figure 36 : Véhicule utilisé

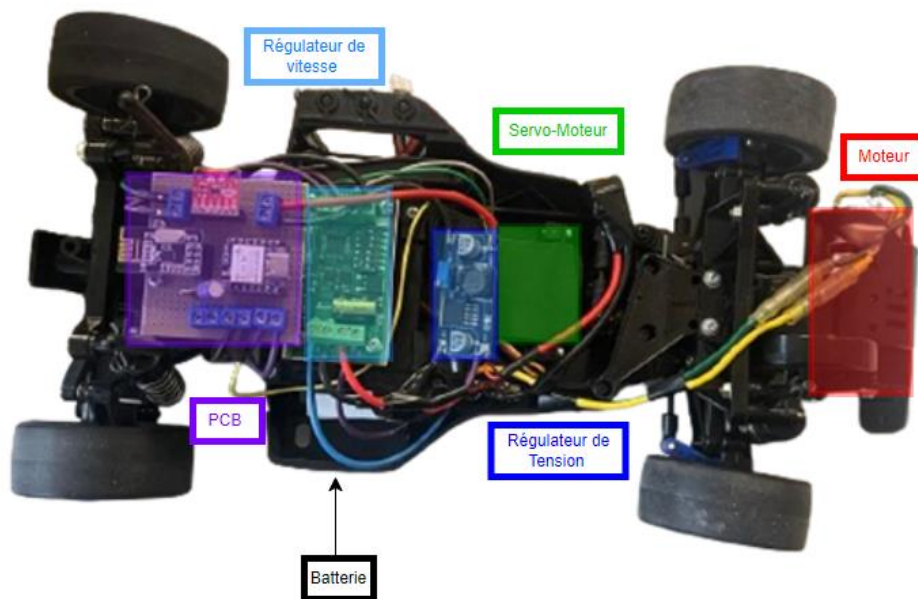


Figure 37 : Schéma d'intégration des composants dans la voiture

Le tableau ci-après reprend la liste de tous les composants intégrés (pour plus d'informations techniques sur un composant veuillez-vous référer à l'annexe 1) :

Composant	Image	Rôle	Justification	Prix
<b>Régulateur vitesse (repris de l'université)</b>		Permet de choisir le sens de rotation des roues, et la vitesse	Besoin de ce composant pour piloter le moteur correctement	54,50 €
<b>Seeed Xiao (acheter)</b>		Microcontrôleur de la voiture, permet de recevoir les commandes de mouvement et les exécuter	Besoin pour être capable de piloter la logique de la voiture	19,90 €
<b>Régulateur de tension (acheter)</b>		Réduire la tension de la batterie pour être compatible avec le microcontrôleur.	Pour alimenter le montage, nous avons besoin de réduire la tension de 7v de la batterie, vers 5v (maximum supporté)	2,70 €
<b>Convertisseur Analogique-Numérique (acheter)</b>		Lisser le signal envoyé par le microcontrôleur vers le régulateur de vitesse	Pour que le microcontrôleur puisse envoyer un signal stable au régulateur de vitesse, permettant une vitesse lente et contrôlée de mouvement.	7,10 €
<b>Connecteurs batterie (acheter)</b>		Adaptateur pour alimenter le montage	Besoin de cette pièce pour connecter la batterie au montage	0,95 €
<b>Antenne nRF24L01 (repris de l'université)</b>		Communiquer entre microphone et voiture	Cette antenne permettra d'établir la communication de commandes entre le microphone et le véhicule	5 € (paire)
<b>Batterie (repris de l'université)</b>		Alimenter les composant de la voiture	Fournit une autonomie complète a la voiture	19,95 €
<b>Servomoteur (repris de l'université)</b>		Permet de gérer la direction des roues de la voiture	Ce servomoteur sera le responsable des commandes de direction gauche, droite, tout droit de la voiture	19 €
<b>Moteur (repris de l'université)</b>		Gérer le déplacement de la voiture.	Gérer le déplacement de la voiture.	16,70 €

Afin de faciliter l'intégration de certains composants dans la voiture, tels que le microcontrôleur, l'antenne et le DAC, nous avons conçu un PCB permettant de les regrouper. Se référer à la figure 35 ci-dessous pour visualiser leur emplacement sur le PCB et à la figure 36 pour voir le câblage.

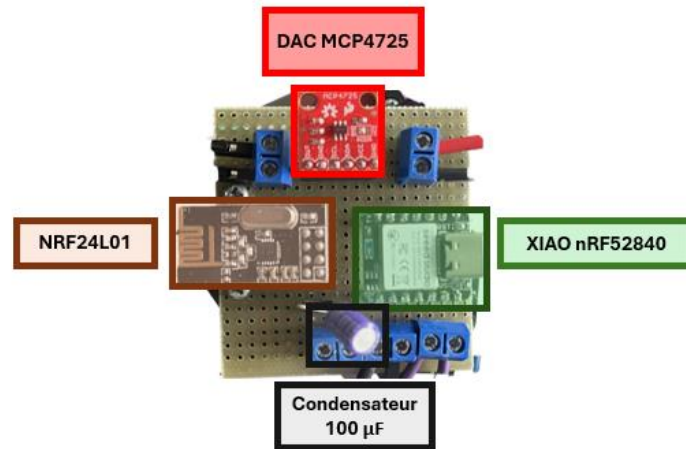


Figure 38 : Composants du BCP

**NB :** le condensateur 100  $\mu\text{F}$  mis en parallèle de l'alimentation est indispensable pour éviter tout dysfonctionnement de l'antenne et assurer une réception sans latence.

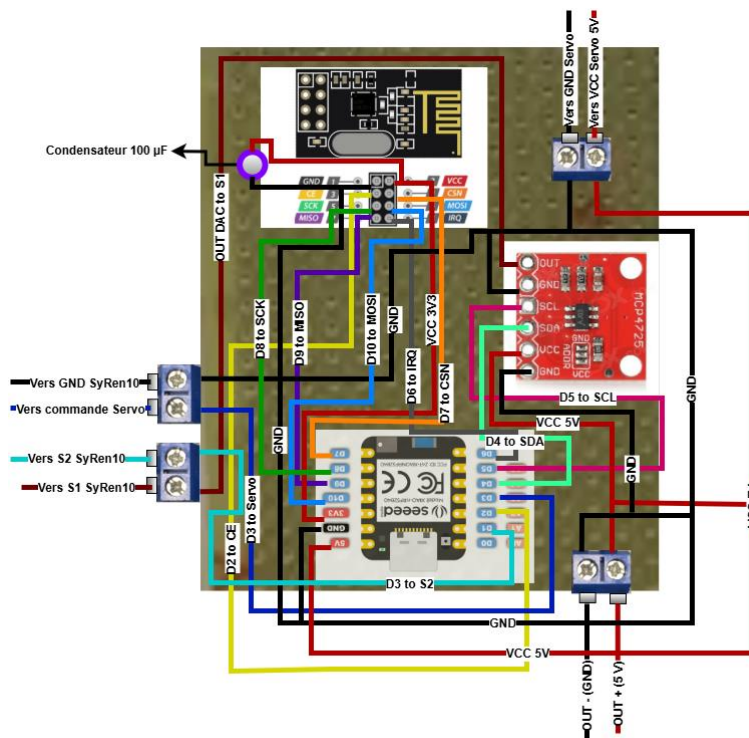


Figure 39 : Câblage BCP

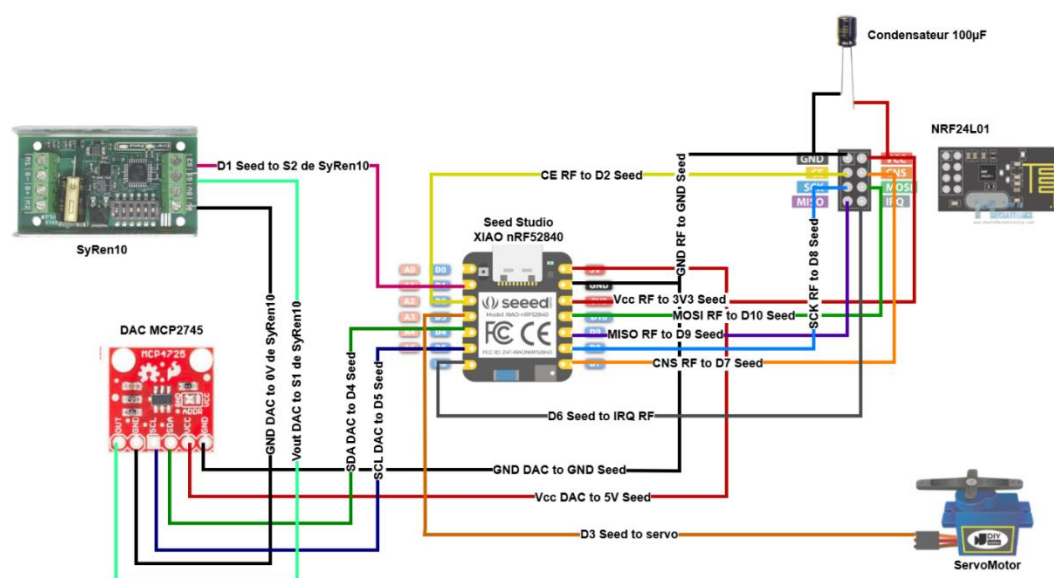
Finalement on aura le schéma d'interconnexions des composants de la voiture comme ceci (Figure 37)





toute sécurité. Quant au module radio NRF24L01 et le convertisseur numérique-analogique (DAC) ils sont directement alimentés par la sortie 3V3 et 5V respectivement de la Seeed XIAO.

**NB :** si on veut travailler sans le PCB le schéma d'interconnexion sera comme ceci.



**Figure 39 42: Schéma câblage sans PCB**

Le tableau ci-après résume les pins utilisés de la Seeed XIAO :

Composants	Pin Seeed XIAO	Description
Servo-moteur	D3	Contrôle de la direction
Moteur (S2)	D1	Contrôle du sens
nRF24L01	D2 (CE), D7 (CSN)	Communication sans fil
IRQ (nRF24L01)	D6	Interruption
DAC MCP4725	I2C (SDA -D4-/SCL-D5-)	Contrôle de la vitesse
LED Interne	D13	Indication de commande reçue



## b) Côté logiciel

Pour la programmation de la Seeed Xiao nRF52840 on a utilisé **Arduino IDE** comme éditeur de code, car il se programme avec le langage de code Arduino.

En premier, on a besoin d'installer la board seeed Studio sur Arduino IDE pour qu'il arrive à reconnecter notre carte quand on la branche au PC, pour ce faire, on va sur **BOARDS MANAGER >> chercher Seeed nrf52 boards >> cliquer sur INSTALLER**

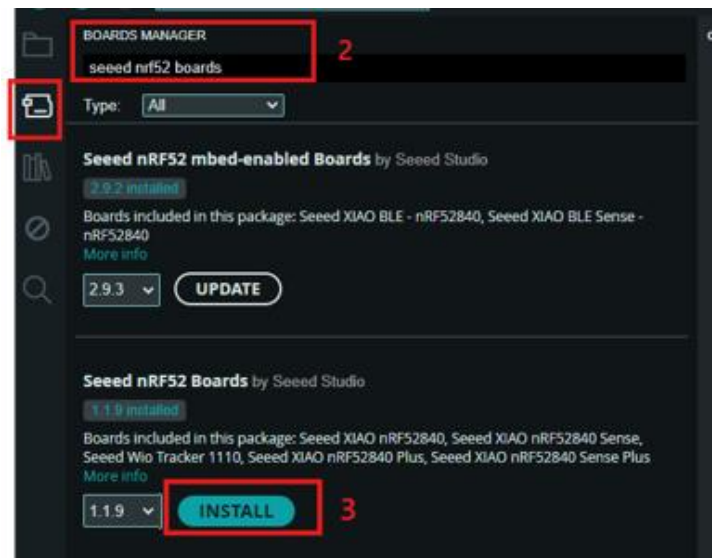


Figure 4043 : installation de la board seeed nrf52

Ensuite, sur **Tools >> Boards >> Seeed nrf52 boards >> on selection seeed XIAO NRF52840 Sens** (Figure 41)

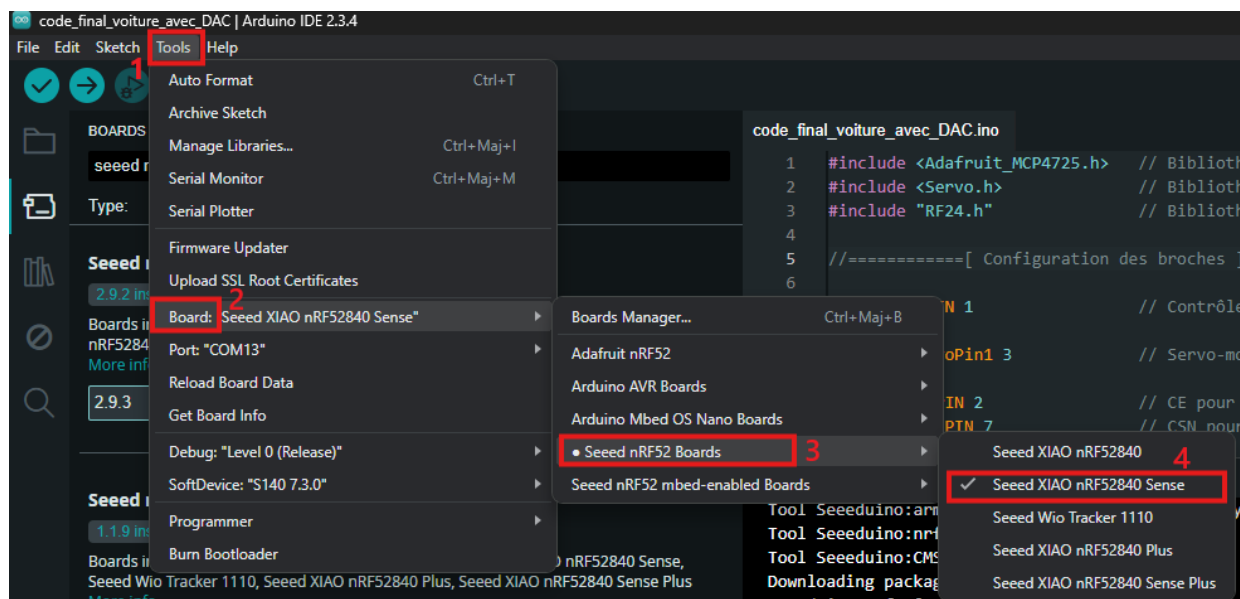


Figure 41 44 : Sélection de la carte d'utilisation

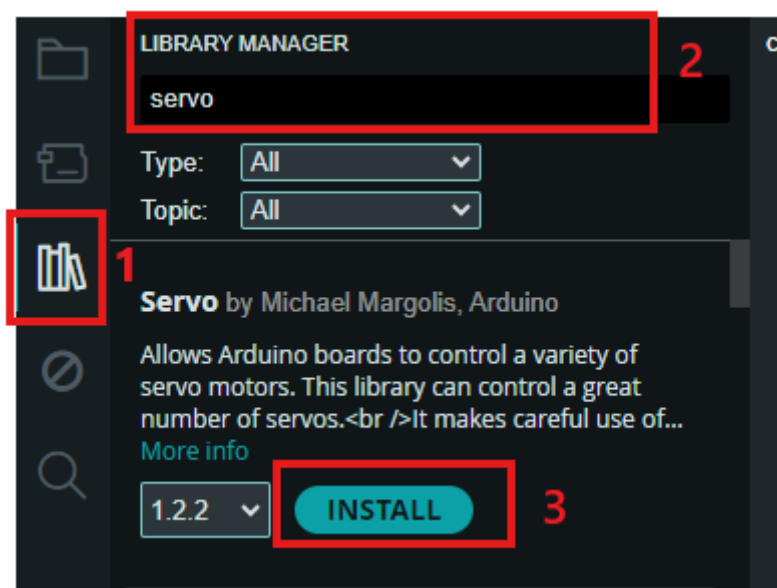
Maintenant notre IDE arrive à reconnaître la carte XIAO sans problème.

Deuxièmement, on a besoin d'installer ces 03 bibliothèques :

- **Adafruit\_MCP4725** >> pour l'utilisation du DAC
- **Servo** >> pour l'utilisation du servomoteur
- **RF24** >> pour l'utilisation de l'antenne NRF24L01

Pour les installer, il faut aller sur **LIBRARY MANAGER** de Arduino IDE, dans la barre de recherche on cherche la bibliothèque et on clique sur **installer**.

Un exemple d'installation est illustré sur la figure 42 ci-dessous



*Figure 4245 : Exemple installation librairie*

À présent, toutes les configurations nécessaires ont été complétées. Il ne reste plus qu'à copier le code du fonctionnement de la voiture de notre projet Git, et le coller sur le vôtre.

Pour consulter la logique du code, consultez le GRAFCET de la voiture (document dans le dossier ressources utiles de notre git).

Nous arrivons à la fin de ce guide, à ce stade vous avez le nécessaire coté matériels et logiciels pour faire fonctionner l'ensemble du projet, il reste plus qu'à tester.

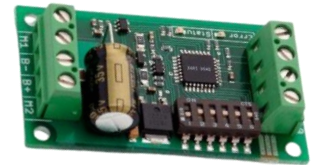
## ANNEXE 1 : Caractéristiques Techniques des composants

### Microcontrôleur : Seeed Studio XIAO nRF52840



- **Capacités sans fil puissantes** : Bluetooth 5.0 avec antenne intégrée/NFC
- **Processeur performant** : Nordic nRF52840, ARM Cortex-M4 32 bits avec FPU, 64 MHz
- **Ultra basse consommation** : Consommation en veille inférieure à 5µA
- **Puce de gestion de charge** : Prise en charge de la gestion de charge et de décharge des batteries lithium
- **Mémoire** : 256 KB de RAM, 1 MB de Flash, ainsi que 2 MB de Flash embarquée.
- **Format ultra compact** : 21 x 17.8 mm, adapté aux dispositifs portables
- **Interface** : 1xUART, 1xIIC, 1xSPI, 1xNFC, 1xSWD  
11x GPIO (PWM), 6xADC

### SyRen10 : Contrôleur de moteur



- **Tension d'entrée** : 6V à 24V nominal, 30V absolu max
- **Courant continu** : 10A (avec une tension d'entrée jusqu'à 18V en air libre sans dissipateur thermique), 15A en crête pendant quelques secondes
- **Protection** : Surchauffe, surtension, sous-tension.
- **Modes de contrôle** : Analog Input, R/C Input, Simplified serial, Packetized serial
- **Dimensions** : 1.4" x 2.25" x 0.55" (≈ 35.6mm x 57.1mm x 14mm)
- **Poids** : 26g
- **Connectique** :
  - **M1 & M2** : Connexion du moteur (permet l'inversion du sens de rotation en inversant les câbles).
  - **B+ & B-** : Connexion de la batterie ou de l'alimentation.
  - **S1 & S2** : Entrées de commande (S1 = principal, S2 = optionnel selon le mode sélectionné).
  - **0V & 5V** : Alimentation de périphériques externes (5V @ 100mA max si source ≤ 12.6V, sinon 10mA max).

## DAC MCP4725 : Convertisseur numérique-analogique



- **Résolution** : 12 bits, offrant 4096 niveaux de sortie distincts.
- **Tension d'alimentation** : 2.7V à 5.5V
- **Un courant de sortie** : pouvant monter jusqu'à 15 mA
- **Interface de communication** : I2C
- **Un adressage possible sur 8 adresses I2C différentes** : 0x60 ou 0x61, pour les plus classiques, suivant si on relie la broche A0 à GND ou +Vcc
- **Courant de sortie** : 25 mA max
- **Mémoire EEPROM intégrée** : Permet de stocker la valeur de sortie du DAC, assurant ainsi la restitution de cette valeur lors de la remise sous tension.
- **Faible consommation d'énergie** : Conçu pour des applications nécessitant une consommation réduite.
- **Dimensions** : Disponible en boîtier SOT-23 à 6 broches (Vcc, GND, SDA, SCL, OUT, GND), idéal pour les conceptions compactes.

## NRF24L01 : Communication Radio



- **Fréquence** : 2,4 GHz – 2,525 GHz
- **Débits** : 250 kbps, 1 Mbps, 2 Mbps
- **Alimentation** : 1,9V à 3,6V (fonctionne en 3,3V max)
- **Consommation** : ~11,3 mA en émission, ~13,5 mA en réception, ~26 µA en veille
- **Interface** : SPI (jusqu'à 10 Mbps)
- **Portée** : ~100 m en champ libre (avec antenne PCB)
- **Puissance d'émission** : Réglable (0, -6, -12, -18 dBm)
- **Canaux** : 125 disponibles, jusqu'à 6 connexions simultanées
- **Antenne** : Intégrée sur PCB
- **Brochage (8 broches)** : GND, VCC, CE, CSN, SCK, MOSI, MISO, IRQ

### ML2596s : Régulateur de tension à découpage Step-Down (Buck Converter)



- **Tension d'entrée** : 4V à 40V DC
- **Tension de sortie** : 1,25V à 35V DC (réglable via le potentiomètre)
- **Courant de sortie** : Jusqu'à 3A (recommandé 2A max en continu pour éviter la surchauffe)
- **Rendement** : Jusqu'à 92% (dépend de la charge et de la tension d'entrée)
- **Fréquence de commutation** : 150 kHz
- **Précision** :  $\pm 4\%$  sur la tension de sortie
- **Condensateurs** : 220 $\mu$ F 35V pour la stabilisation
- **Inductance** : Bobine d'environ 100 $\mu$ H pour la conversion d'énergie
- **Brochage** :
  - **IN+** : Entrée positive (alimentation)
  - **IN-** : Entrée négative (masse)
  - **OUT+** : Sortie positive (tension régulée)
  - **OUT-** : Sortie négative (masse)

### TECHNIPLUS T2M AS-12 : Servomoteur analogique



- **Tension d'alimentation** : 4.8V à 6V
- **Couple** :
  - Environ 3 à 4 kg·cm sous 4.8V
  - Peut monter légèrement sous 6V
- **Vitesse** : Environ 0.15s/60° sous 4.8V
- **Dimensions** : 40 x 20 x 38 mm (format standard)
- **Poids** : Environ 40-50g
- **Angle de rotation** : 0° à ~180° (dépend du signal PWM)
- **Connecteur** : 3 fils (Signal, VCC, GND)
  - **Noir** → GND (masse)
  - **Rouge** → VCC
  - **Jaune** → Signal PWM (de 1ms à 2ms pour contrôler l'angle)

**Moteur : Max Power 450 Electric Brushed Motor**

- **Vitesse nominale** : 11500 tr/min sous 12V
- **Tension nominale** : 12v
- **Plage de tension** : de 3V à 24V.
- **Courant à vide** : 0.4A
- **Courant au rendement max** : 2.4A
- **Courant au blocage** : 16,0A
- **Diamètre arbre** : Ø2,3mm
- **Diamètre du corps** : Ø30mm
- **Longueur du corps** : 46,5mm
- **Poids** : 100 g





## ANNEXE 2 : Modèle IA utilisé

Pour créer notre modèle IA, nous avons utilisé ce projet suivant :

[https://www.tensorflow.org/tutorials/audio/simple\\_audio](https://www.tensorflow.org/tutorials/audio/simple_audio)

TensorFlow > Learn > TensorFlow Core

Was this helpful?  

### Simple audio recognition: Recognizing keywords



Run in Google Colab



View source on GitHub



Download notebook

This tutorial demonstrates how to preprocess audio files in the WAV format and build and train a basic [automatic speech recognition](#) (ASR) model for recognizing ten different words. You will use a portion of the [Speech Commands dataset](#) ([Warden, 2018](#)), which contains short (one-second or less) audio clips of commands, such as "down", "go", "left", "no", "right", "stop", "up" and "yes".

Real-world speech and audio recognition [systems](#) are complex. But, like [image classification with the MNIST dataset](#), this tutorial should give you a basic understanding of the techniques involved.

**Figure 4346 : Capture écran introduction code exemple**

Nous avons réduit la quantité de commandes de 8 à 6 pour avoir juste les nécessaires, et réduit la taille du modèle entraîné le plus possible, pour qu'il puisse être compatible avec la taille mémoire flash déjà assez limitée du microcontrôleur.

Ce dataset modifié nommé « mini\_speech\_commands » sera présent dans le répertoire de ressources utiles du projet.

Voici le lien de notre projet modifié :

<https://colab.research.google.com/drive/1fl1zPlOgHq6ogIS0qICN0NdJLK5FVXia?usp=sharing>

Le code s'occupe de transformer les fichiers .wav du dataset dans le format nécessaire, de la même façon que notre code STM32.

Pour exporter un nouveau modèle de format « .tflite », suivre les pas indiqués sur le code, du début à la fin.