



PROJET A3 TC : << *AUDISEN* >>

Partie informatique

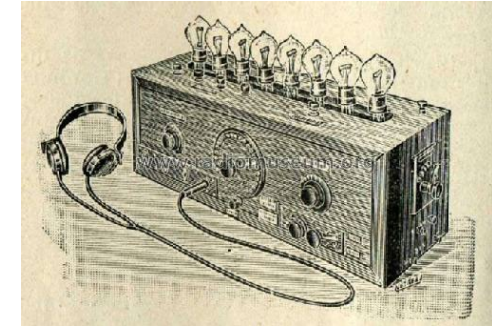




Contexte du projet



AUDISEN

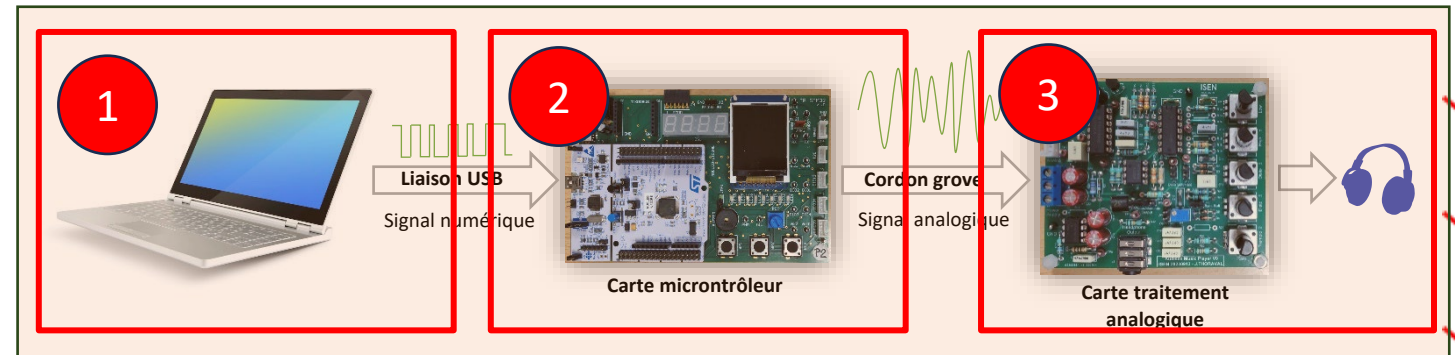


- Audisen Music Player :
 - Conception pédagogique de qualité rudimentaire
 - Qui réconcilie le numérique et l'analogique
- Fonctionnalités principales :
 - Mode Playlist :
 - Composer ses propres mélodies
 - Ecouter / Réécouter une playlist
 - Mode Snaper :
 - Jouer du piano
- Découpe en 3 parties
 - 1 – Informatique (Langage C)
 - 2 – Microcontrôleur (STM32)
 - 3 – Elec. Analogique

ISEN

ALL IS DIGITAL! → Mais pas que !
QUEST

Audisen Music Player



Projet A3 Tronc commun Informatique

- Encadrement

Site	Brest	Nantes	Caen
Enseignants	P.-J. Bouvet	N. Beaussé	S. Le Gloannec
	F. Legras	C. Buron	N. Abdallah-Saab
	T. Paviet-Salomon		
	K. Gharssalli		

- Pédagogie

- Le projet dure 30h sur 5 jours
- Encadrement par un enseignant les 4 premiers jours
- Encadrement par 2 enseignants le dernier jour

Critères d'évaluations

Critères d'évaluation	Sous-critères
Recette	L'étudiant sait planifier son projet
	L'étudiant sait coordonner un travail d'équipe.
	L'étudiant sait utiliser des méthodes de gestion de projet
	L'étudiant sait appliquer des méthodes techniques
	L'étudiant sait respecter un cahier des charges
Connaissances théoriques	L'étudiant maîtrise son sujet sur les aspects techniques.
	L'étudiant est capable de prendre du recul sur le sujet du projet
	L'étudiant sait appliquer des méthodes techniques
	L'étudiant maîtrise le langage de programmation C
Audit	L'étudiant se soucie de la suite de son projet (documentation...)
	L'étudiant est capable de rendre compte de son projet
	L'étudiant sait respecter des consignes de codage





Présentation générale

Principe d'encodage

- On segmente la partition en ticks
- A chaque tick on a une ou plusieurs notes (4 max)
- La hauteur de la note dépend de la position relative à la clé

♩ = 72

Encodage Audisen

001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016

G3 →

F2 →

A#3
D3
A#2
F2
A#2
D3
G4
F4

A#3
D3
A#2
F2
A#2
D3
G4
F4

A#1
A#1
A#1
A#1
A#1
A#1
A#1
A#1






A#1
A#1
A#1
A#1
A#1
A#1
A#1
A#1

Principe d'encodage

- De la partition à l'arrangement :



- On utilise un jeu limité de symboles

Symbole musical	Nom	Durée
	Ronde	4 temps
	Blanche	2 temps
	Noire	1 temps
	Croche	0,5 temps
	Demi-soupir	0,5 temps

- Le tempo => lien entre Durée note (temps) ↔ Temps réel d'exécution (seconde) = 1 **tick** (plus petite unité de temps exécutable par Audisen)
 - Ici 72 noires/min devient pour Audisen : 144 ticks/min (tpm)

Principe d'encodage

- Chaque note relevée correspond à une fréquence (tableau correspondance)
- A chaque saut d'interligne on se déplace de 2 cases dans le tableau
- On se limite à 5 gammes (5 x 12 notes, # inclus)

➤ Dorénavant on peut s'abstenir de la partition ...

Notation		Numéro note	Fréquence note (Hz)
Française	Internationale		
Si5	B5	60	1975,534
La#5	A#5	59	1864,656
La5	A5	58	1760
Sol#5	G#5	57	1661,219
Sol5	G5	56	1567,982
Fa#5	F#5	55	1479,978
Fa5	F5	54	1396,913
Mi5	E5	53	1318,511
Ré#5	D#5	52	1244,508
Ré5	D5	51	1174,66
Do#5	C#5	50	1108,731
Do5	C5	49	1046,503
Si4	B4	48	987,767
La#4	A#4	47	932,328
La4	A4	46	880
Sol#4	G#4	45	830,61
Sol4	G4	44	783,991
Fa#4	F#4	43	739,989
Fa4	F4	42	698,457
Mi4	E4	41	659,256
Ré#4	D#4	40	622,254
Ré4	D4	39	587,33
Do#4	C#4	38	554,366
Do4	C4	37	523,252
Si3	B3	36	493,884
La#3	A#3	35	466,164
La3	A3	34	440

Principe d'encodage

- On construit ligne par ligne les commandes pour l'automate
- Et on ajoute un effet d'accentuation (plus dynamique)

Diagram illustrating the encoding of musical notes into a sequence of commands for an automaton. The diagram shows two staves (treble and bass) with 16 measures. The treble staff has notes G3, A#3, D3, A#2, F2, A#2, D3, G4, F2, F4, F3, F2, A#3, D3, A#2, F2, A#2, D3, G4, F2, F4, F3, F2. The bass staff has notes A#1, A#1. The diagram illustrates the encoding of musical notes into a sequence of commands for an automaton.

Ligne 1 : A#1 ; A#2 ; D3 ; A#3
Ligne 2 : A#1 ; F2
Ligne 3 : A#1 ; A#2
...
Ligne 16 : A#1 ; F2

+ accentuation (^)

Ligne 1 : A#1 ^ ; A#2 ^ ; D3 ^ ; A#3 ^
Ligne 2 : A#1 x ; F2 ^
Ligne 3 : A#1 x ; A#2 ^
...
Ligne 16 : A#1 x ; F2 ^

Fichier AMS

- Les lignes de commande sont rangées dans un fichier .ams où figurent :
 - Titre du morceau
 - Tempo (bpm)
 - Commandes

■ Tempo (bpm)

■ Commandes

Titre

Tempo (bpm)

Numéro de note

9^{ème} tick

11 (A#1), 23 (A#2), 27 (D3) et 34 (A3)

Numéro de tick

Trame de communication

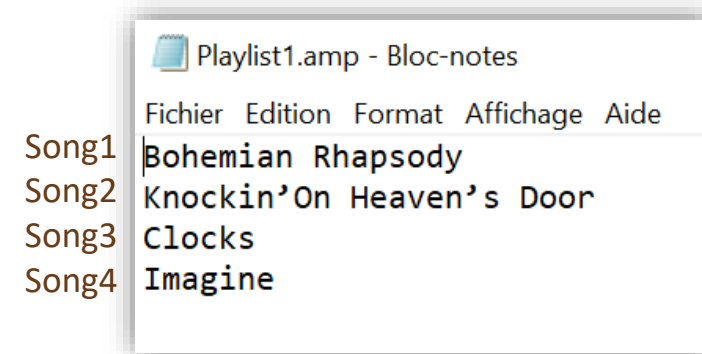
Format trame général :

`#mode,accent,note1,note2,note3,note4*checksum<CR><LF>`

- Trame avec caractères ascii, vitesse 9600 bauds
- Contenu :
 - Mode : '0' (playlist) ou '1' (snaper)
 - Accent : '0' (pas d'accent) ou '1' (accent). Accent si au moins une des 4 notes est accentuée
 - Note1 : '00' (demi-soupir) ou '01'...'60' (note)
 - Note2, 3, 4 : '00' (pas de note) ou '01'...'60' (note)
 - Checksum : XOR entre chaque octet compris entre # et *, écrit en hexadécimal
- En mode snaper : une trame par tick (ne pas oublier d'indiquer le tempo au soft)
- En mode playlist : une trame pour initialiser suivie des n ticks (= trames) du morceau

Lecture de la playlist

- On fournit aux étudiants plusieurs morceaux (fichier .ams)
- Une playlist est composée de 4 morceaux maxi (fichier .amp)
- Les playlists sont à créer puis on génère l'envoi d'une playlist
- Chaque envoi de morceau de la playlist est séparé d'1 seconde



Exemple



Exemple d'envoi de Song1 :

#Bohemian Rhapsody,144,16*17<CR><LF>

Initialiser l'envoi d'un morceau, de tempo 144 tpm, composé de 16 ticks

#0,1,11,15,23,00*18<CR><LF>

Envoi du premier tick accentué de 3 notes

#0,0,09,11,00,00*18<CR><LF>

Envoi du 2ème tick non accentué de 2 notes

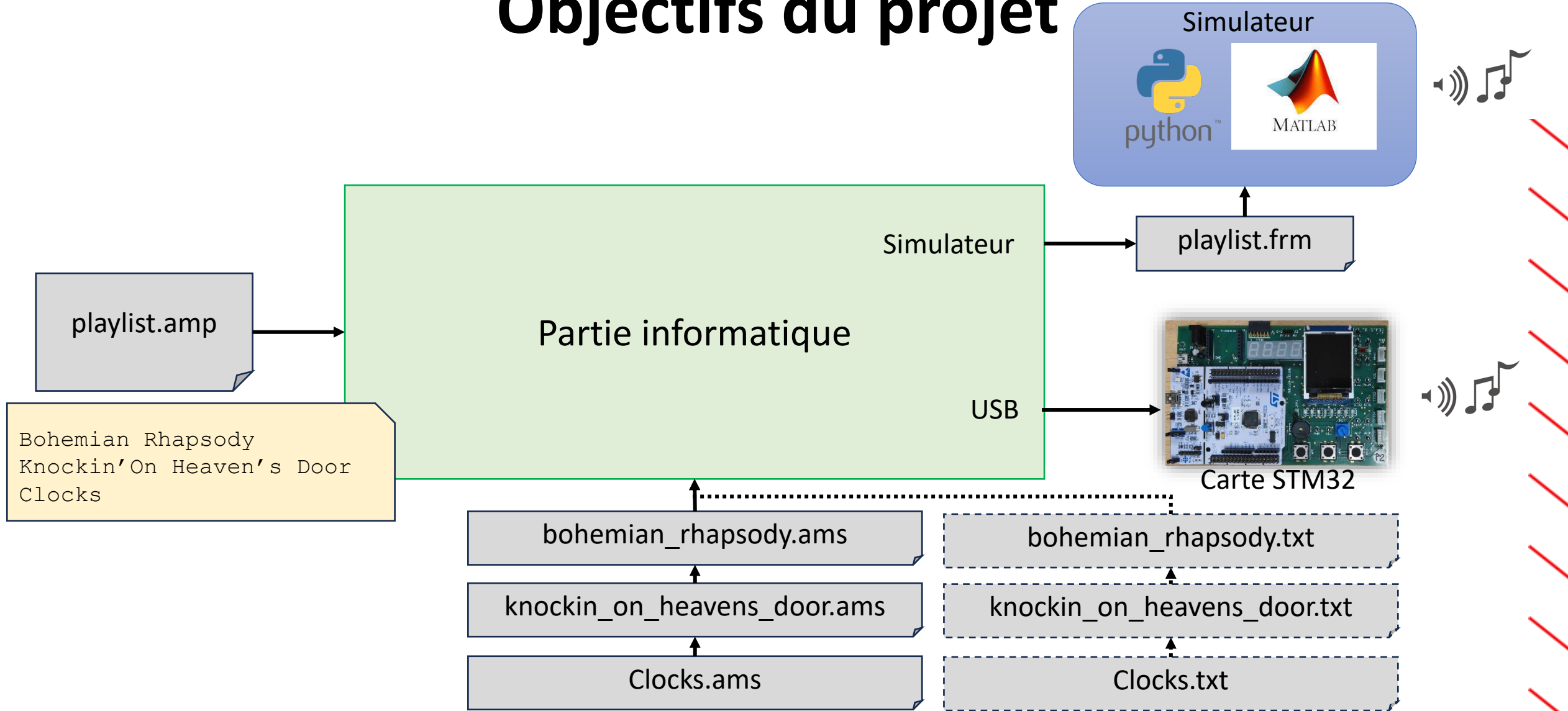
Puis envoyer les 14 autres ticks... et attendre 1 seconde puis envoyer Song2...



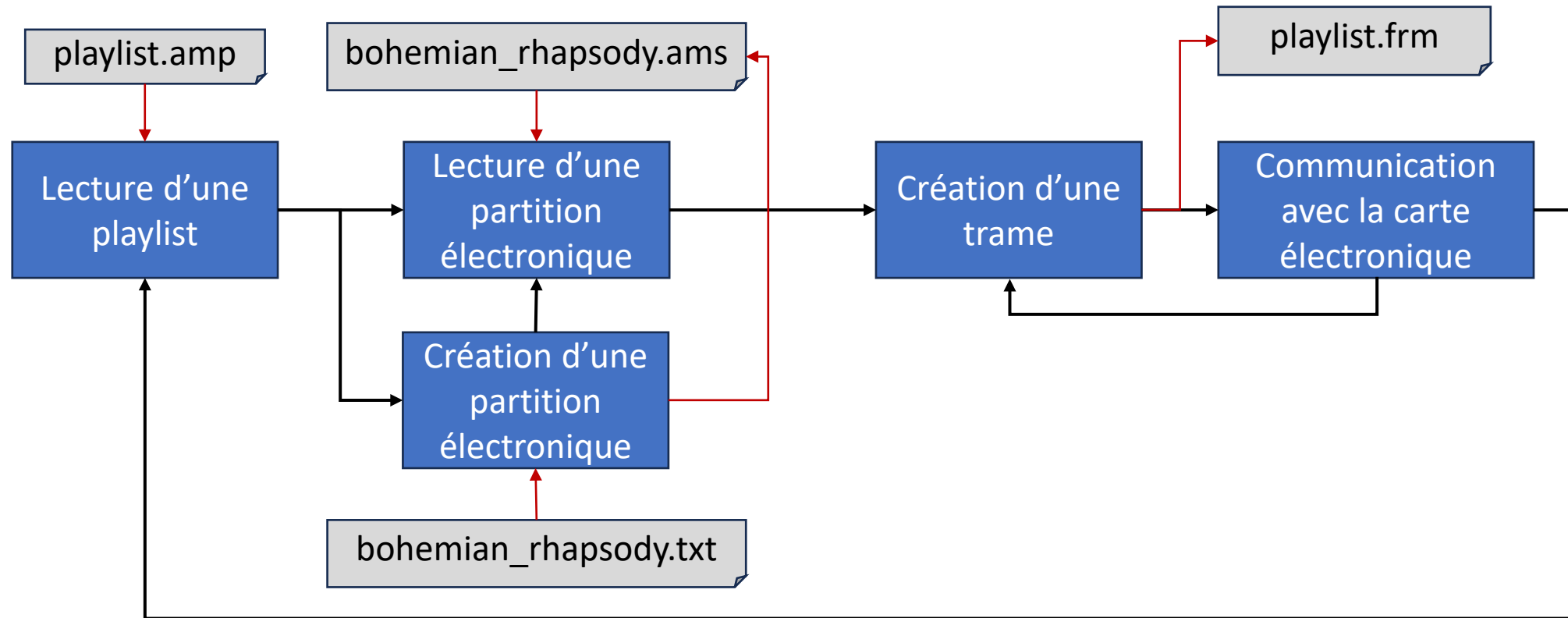
Architecture



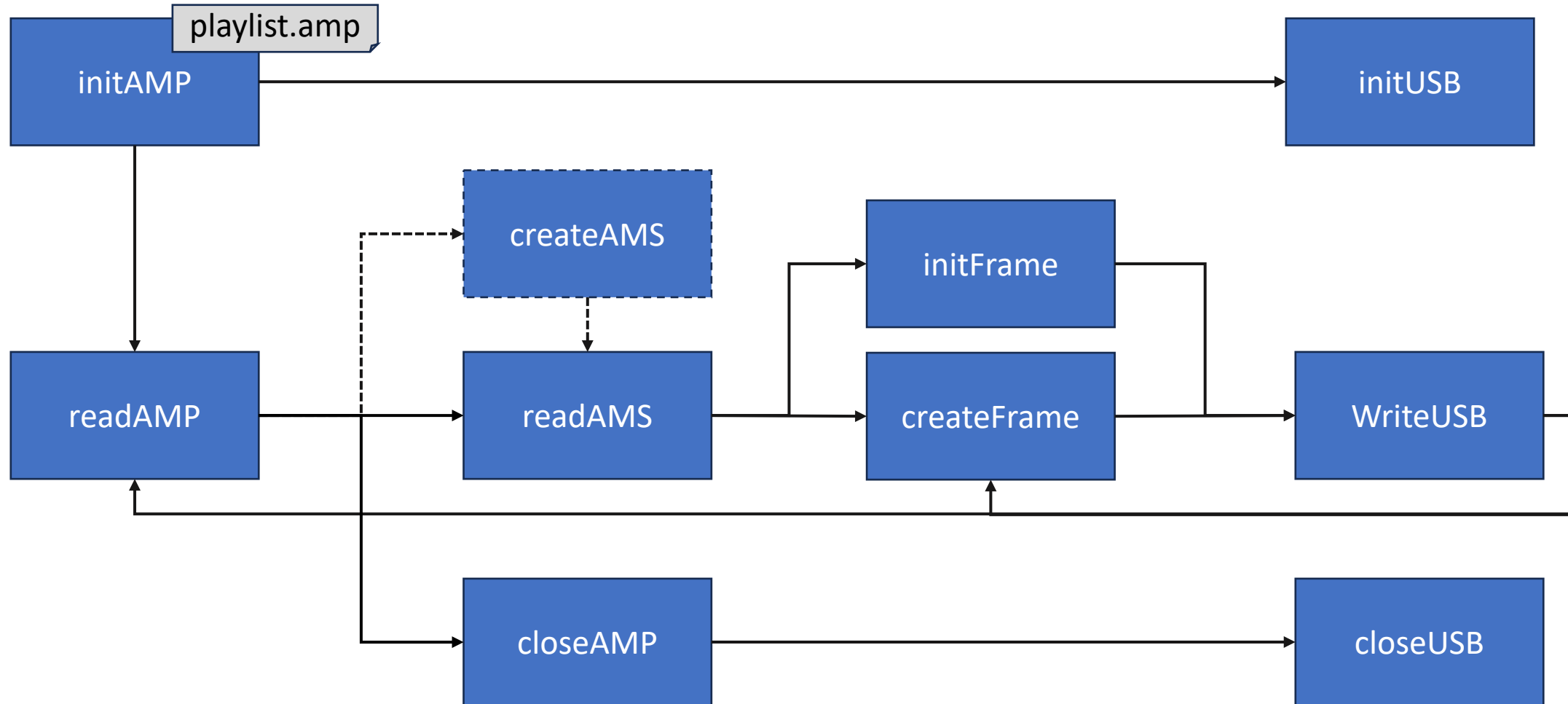
Objectifs du projet



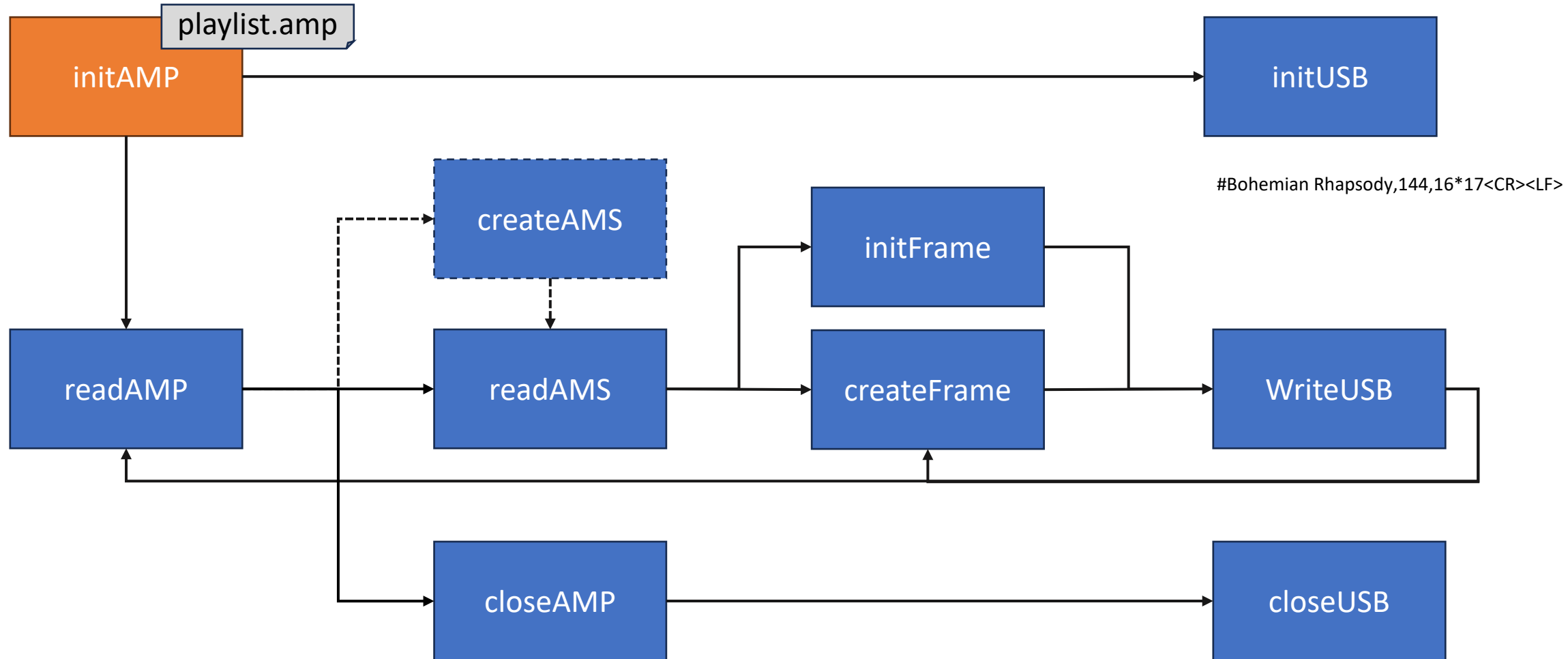
Synoptique



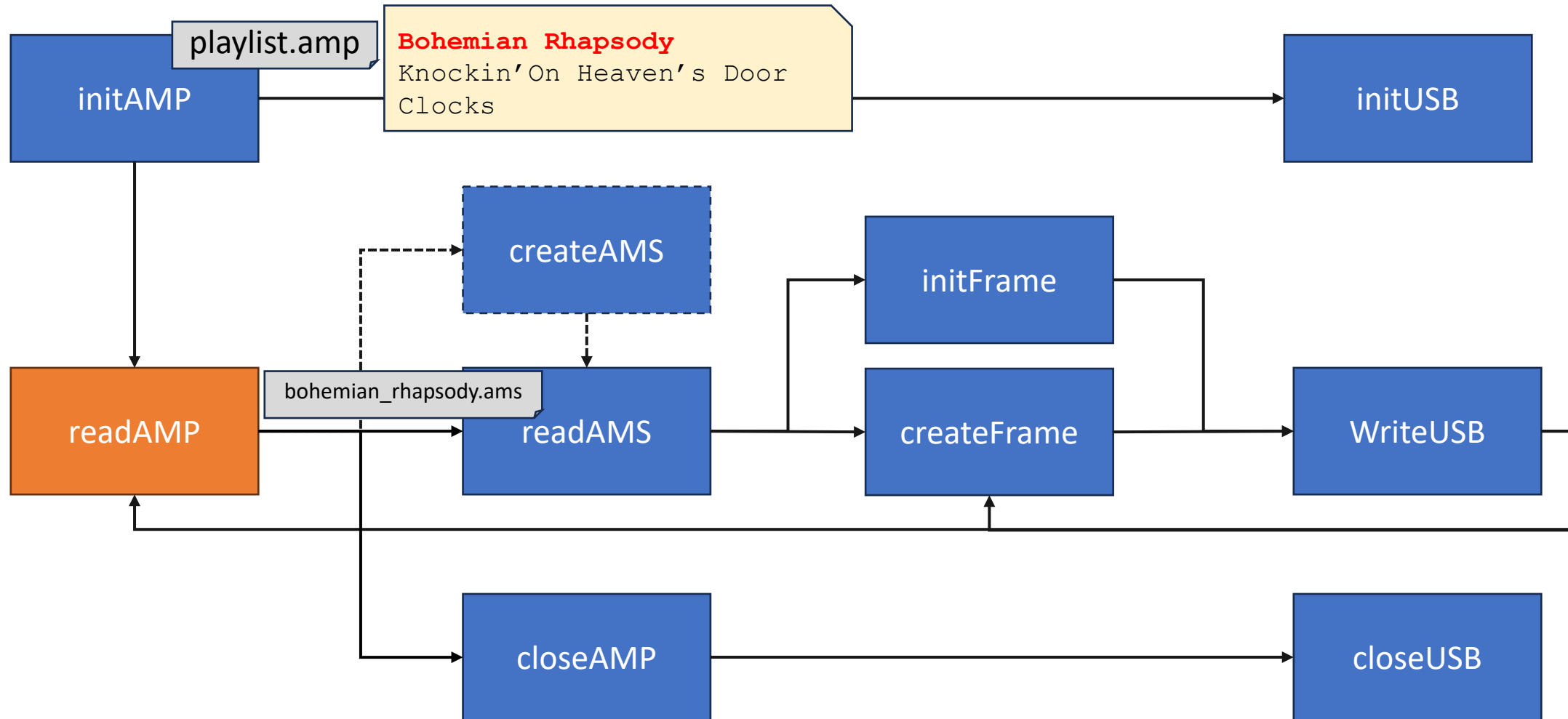
Exécution du programme



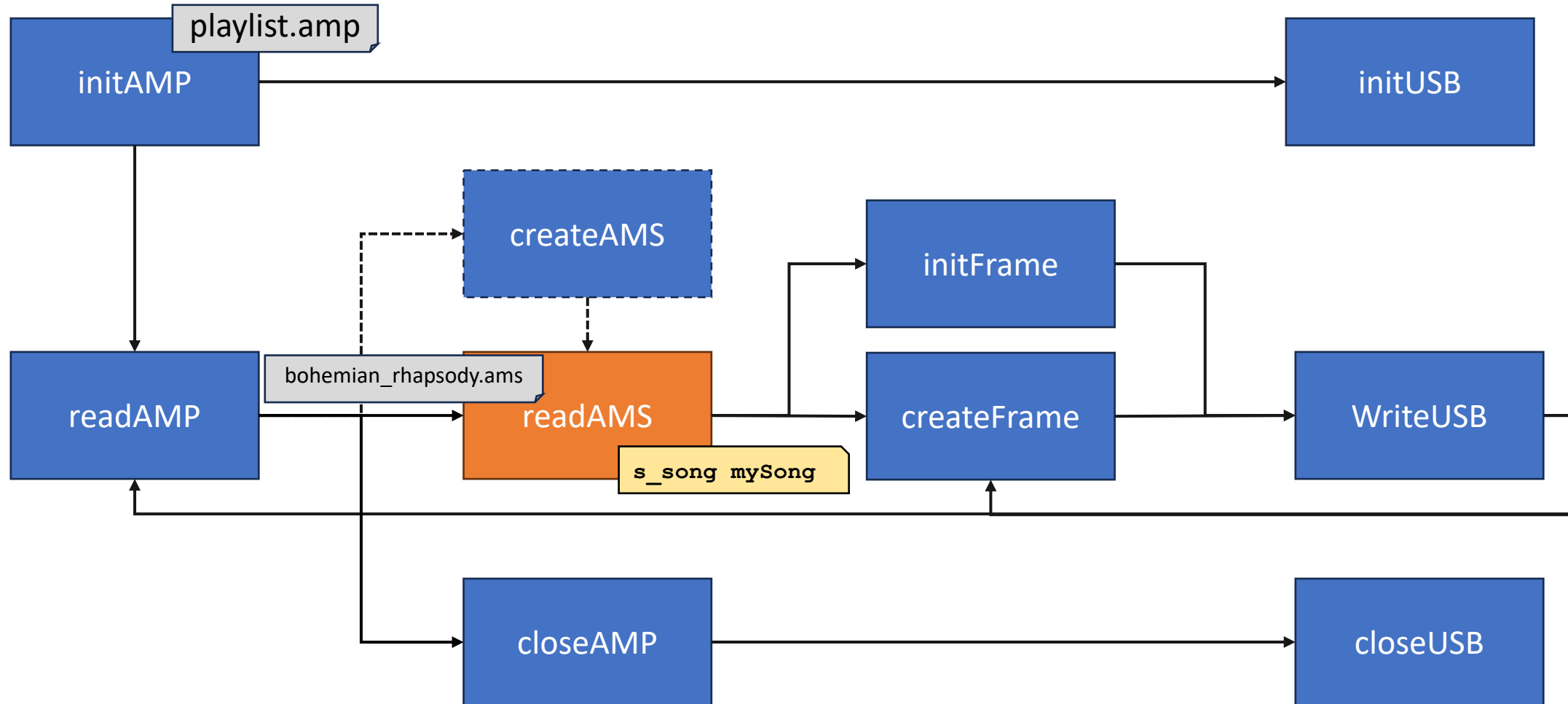
Execution du programme



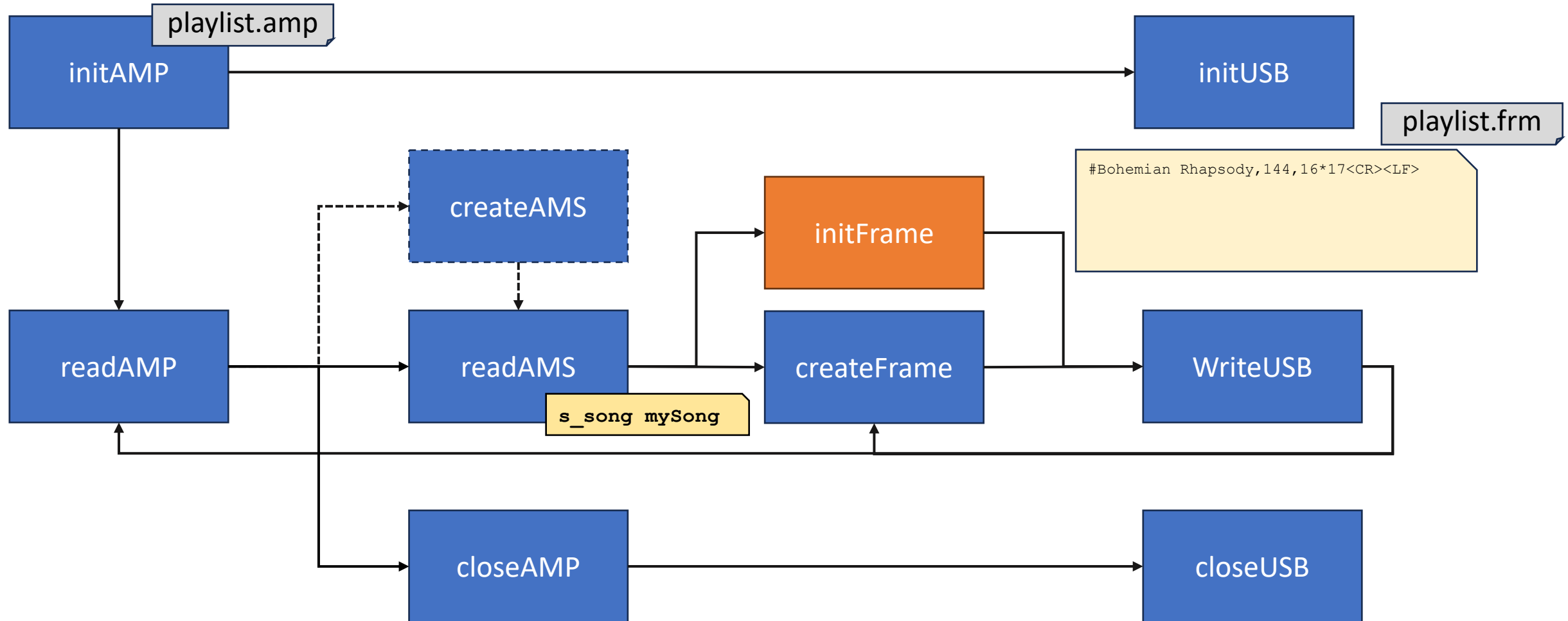
Mode player



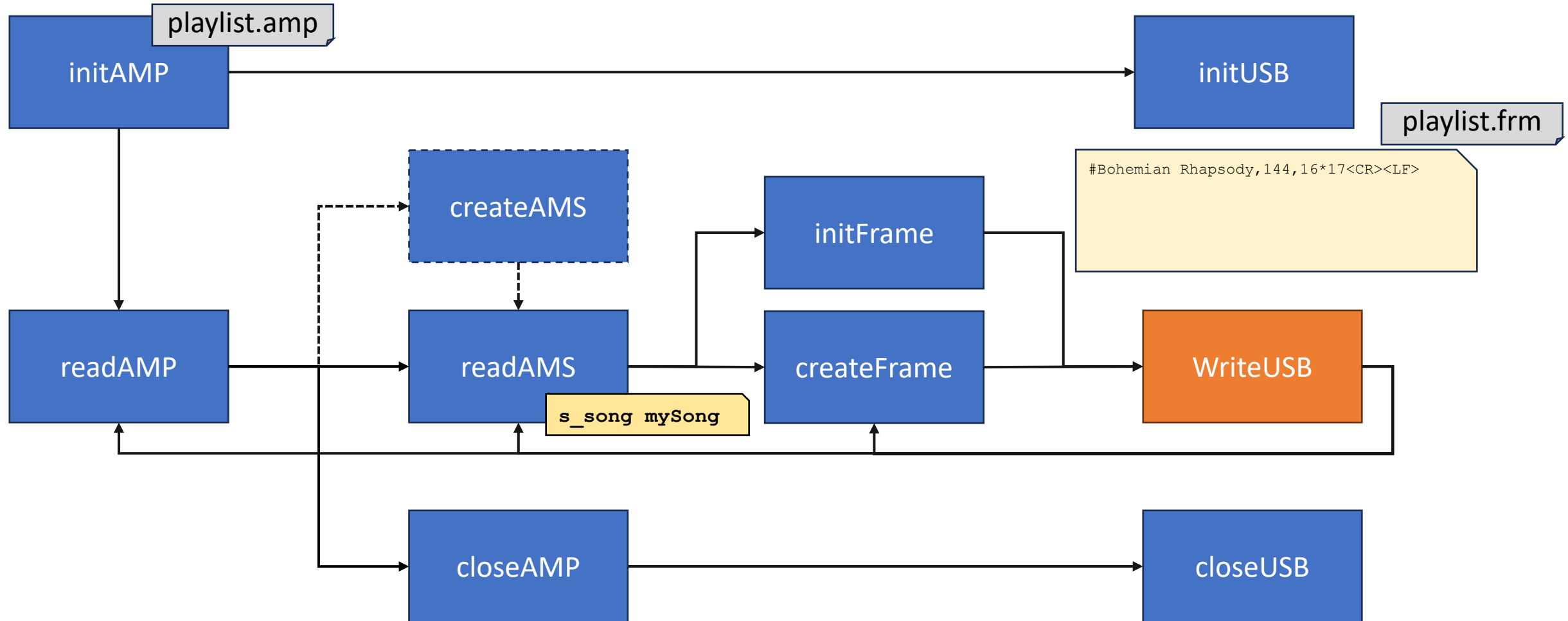
Execution du programme



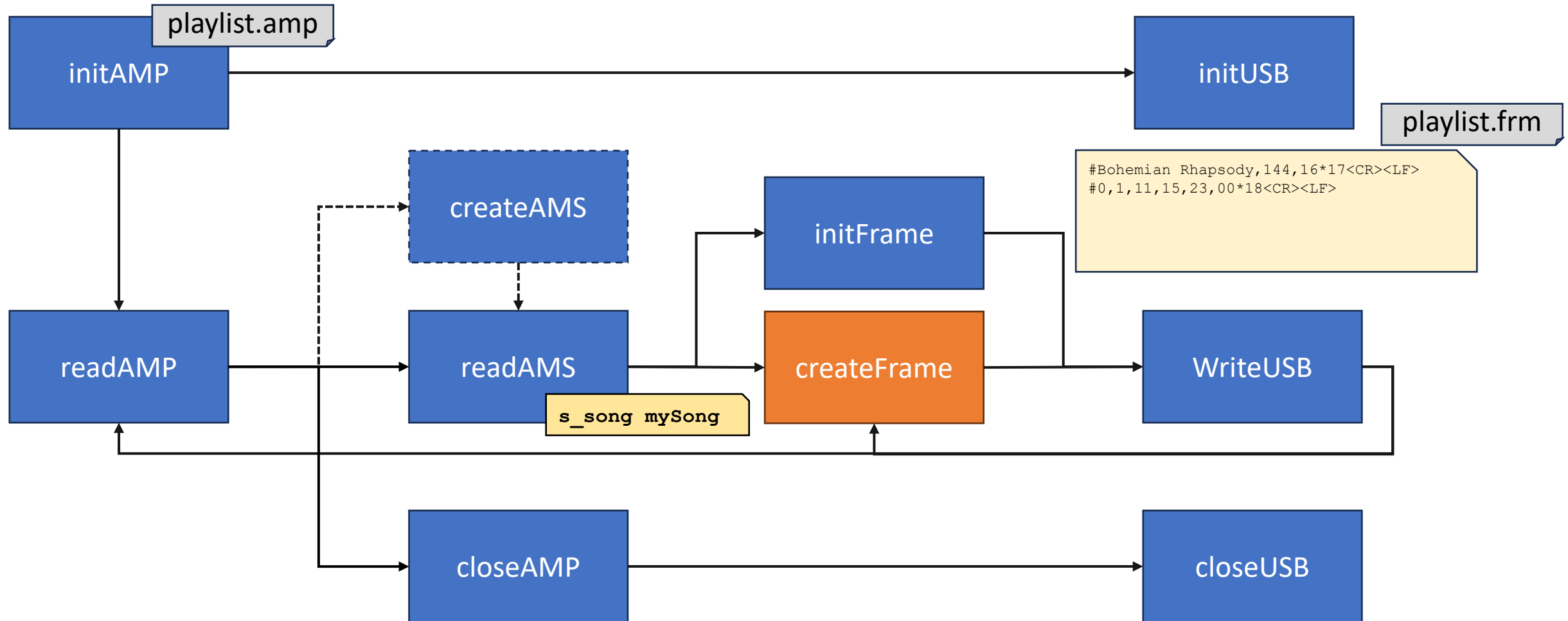
Execution du programme



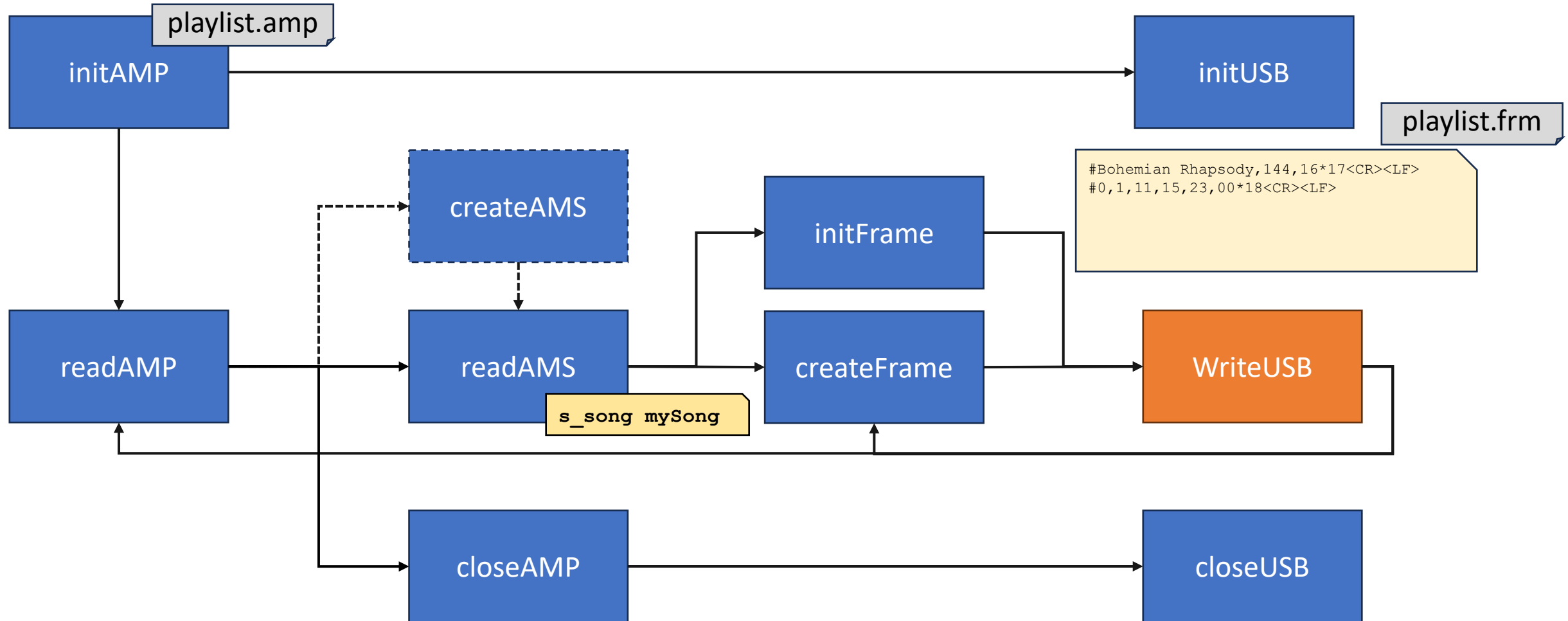
Execution du programme



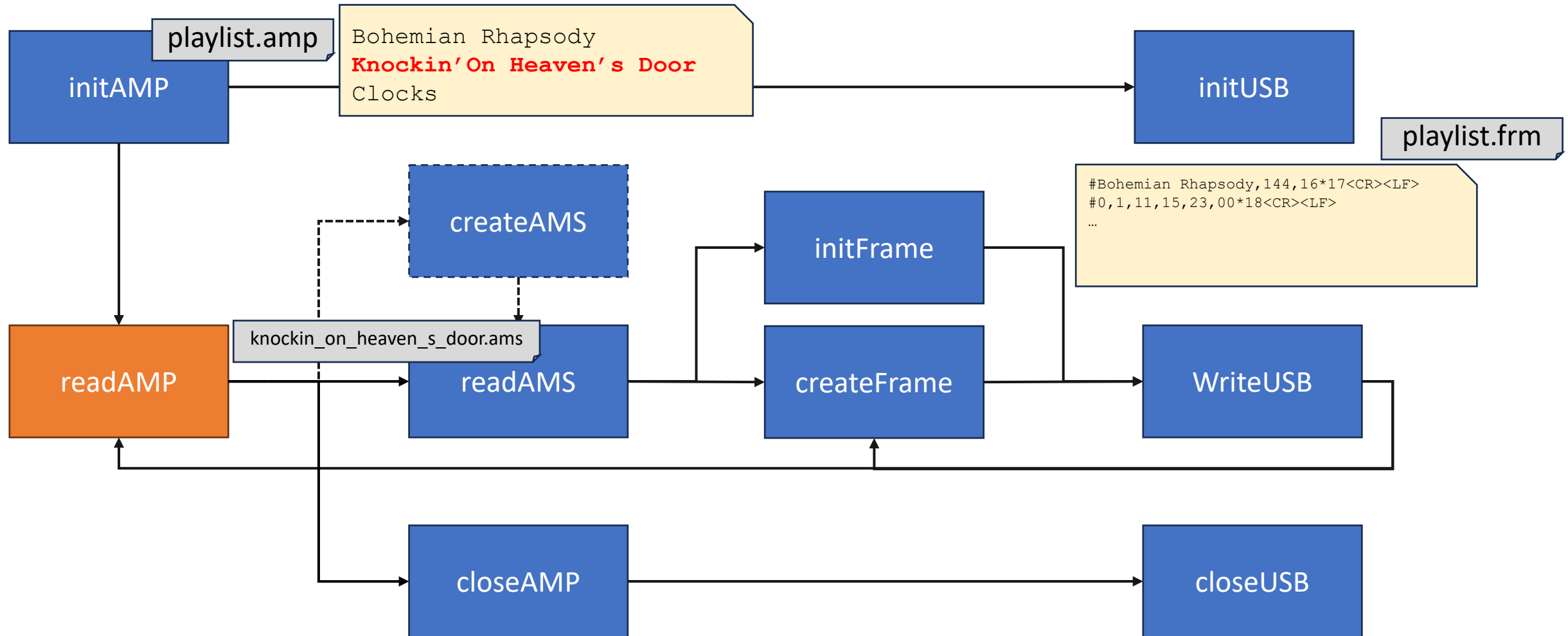
Execution du programme



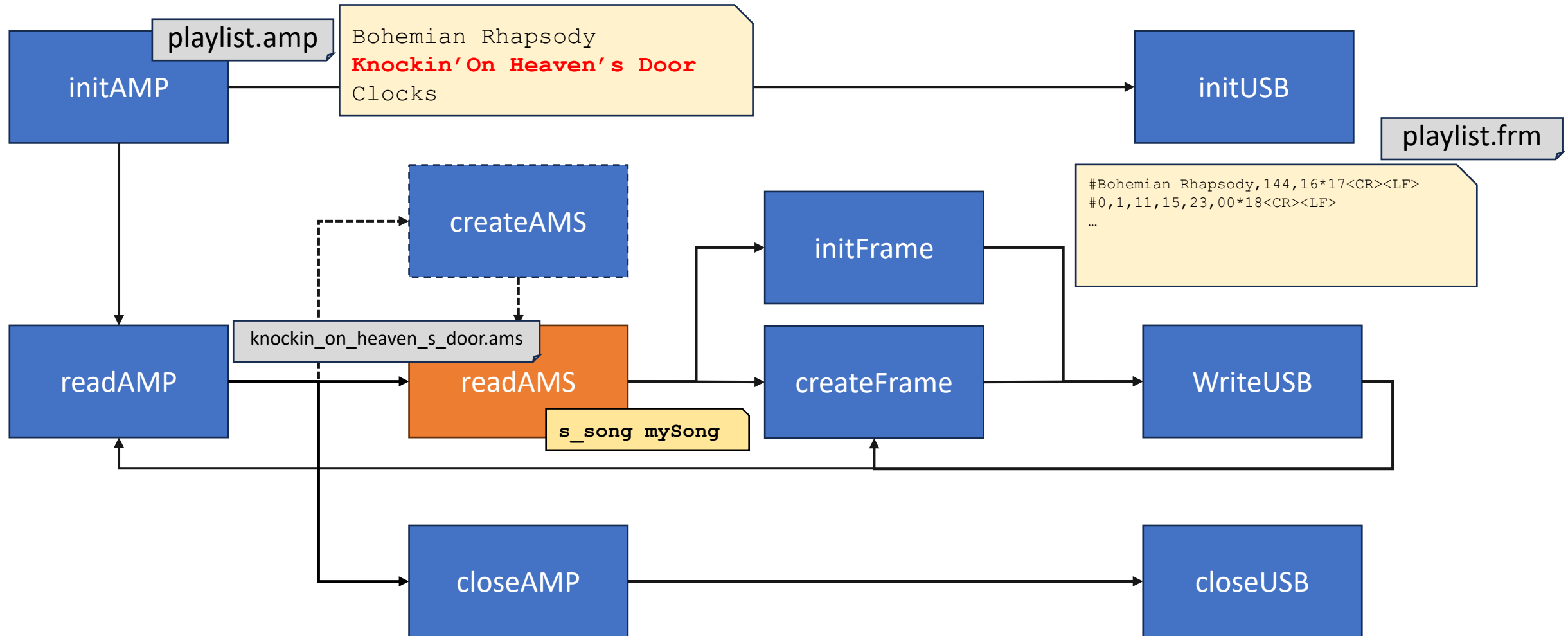
Execution du programme



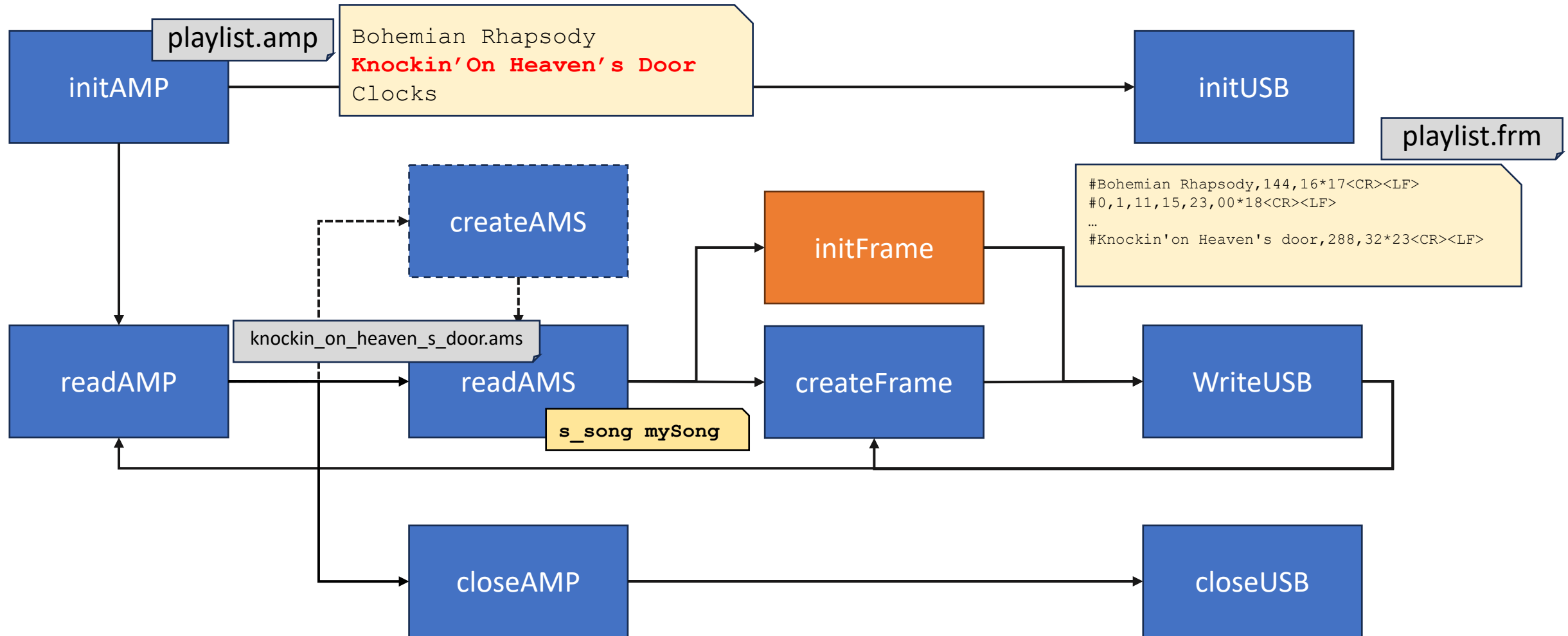
Execution du programme



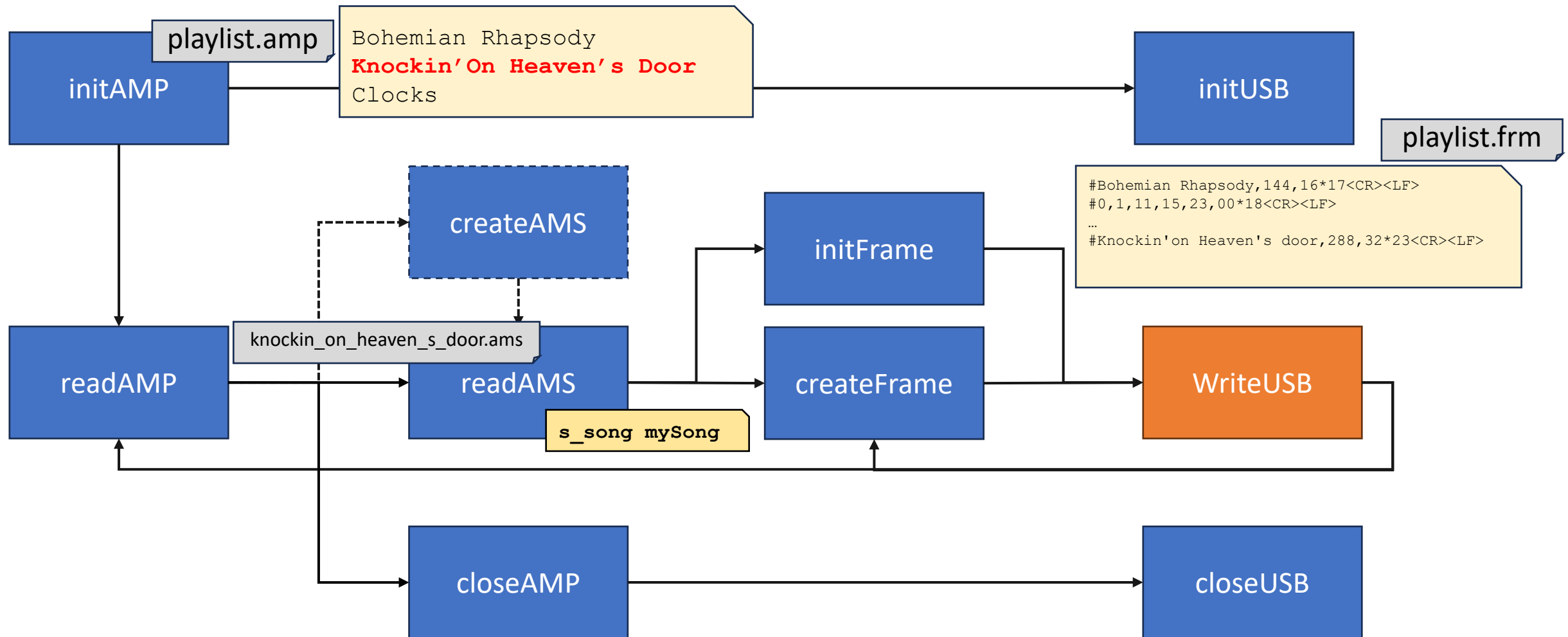
Execution du programme



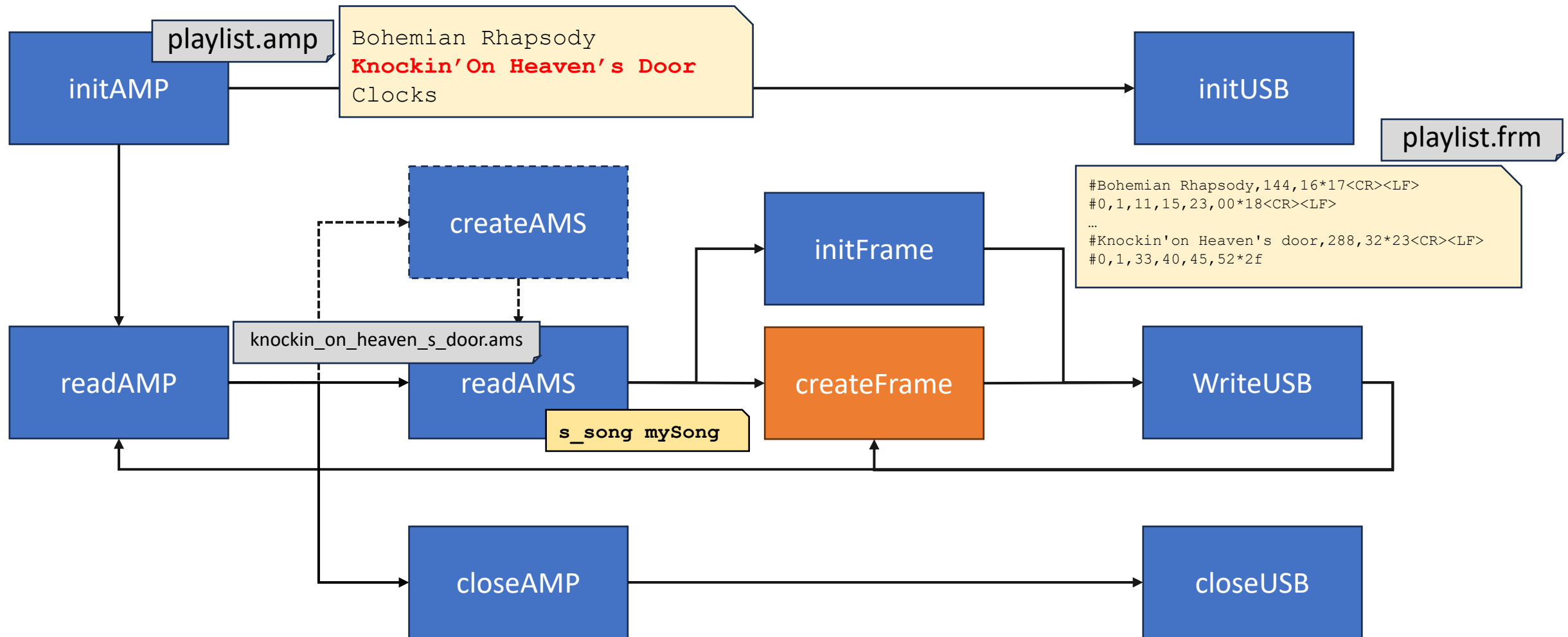
Execution du programme



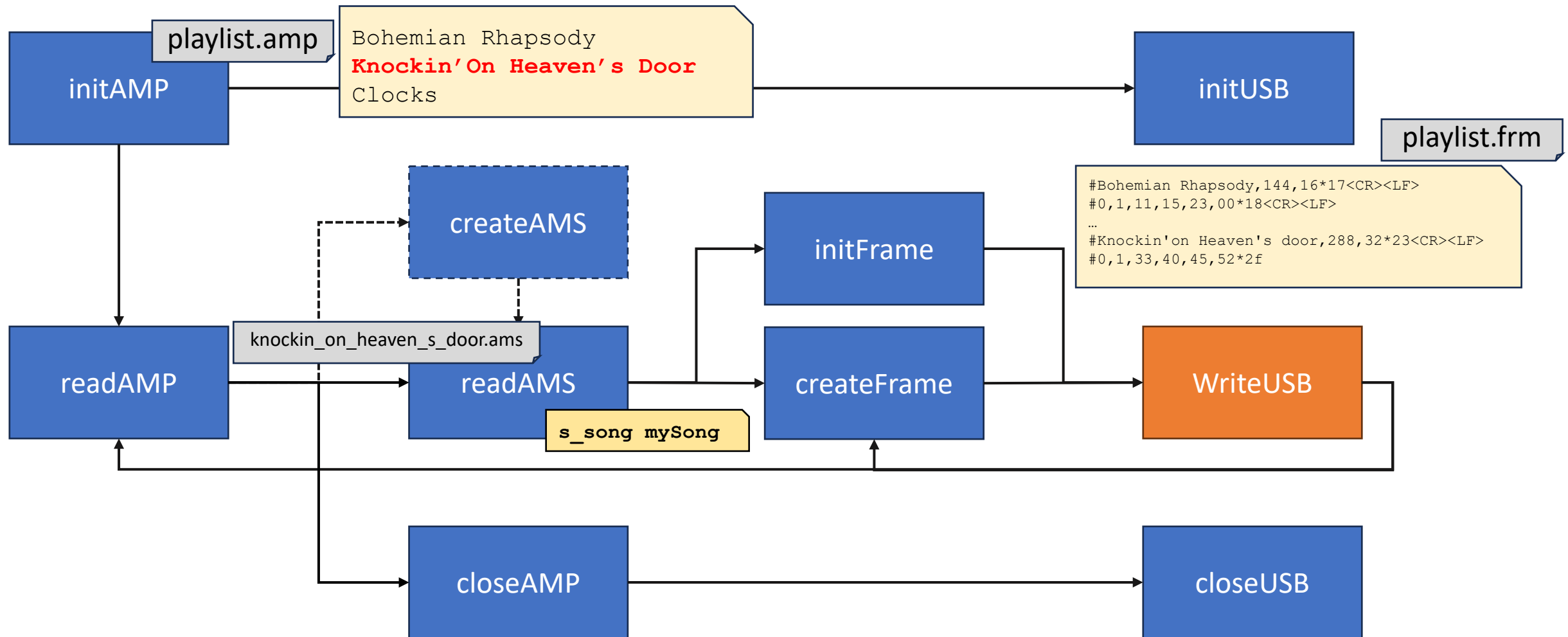
Execution du programme



Execution du programme



Execution du programme

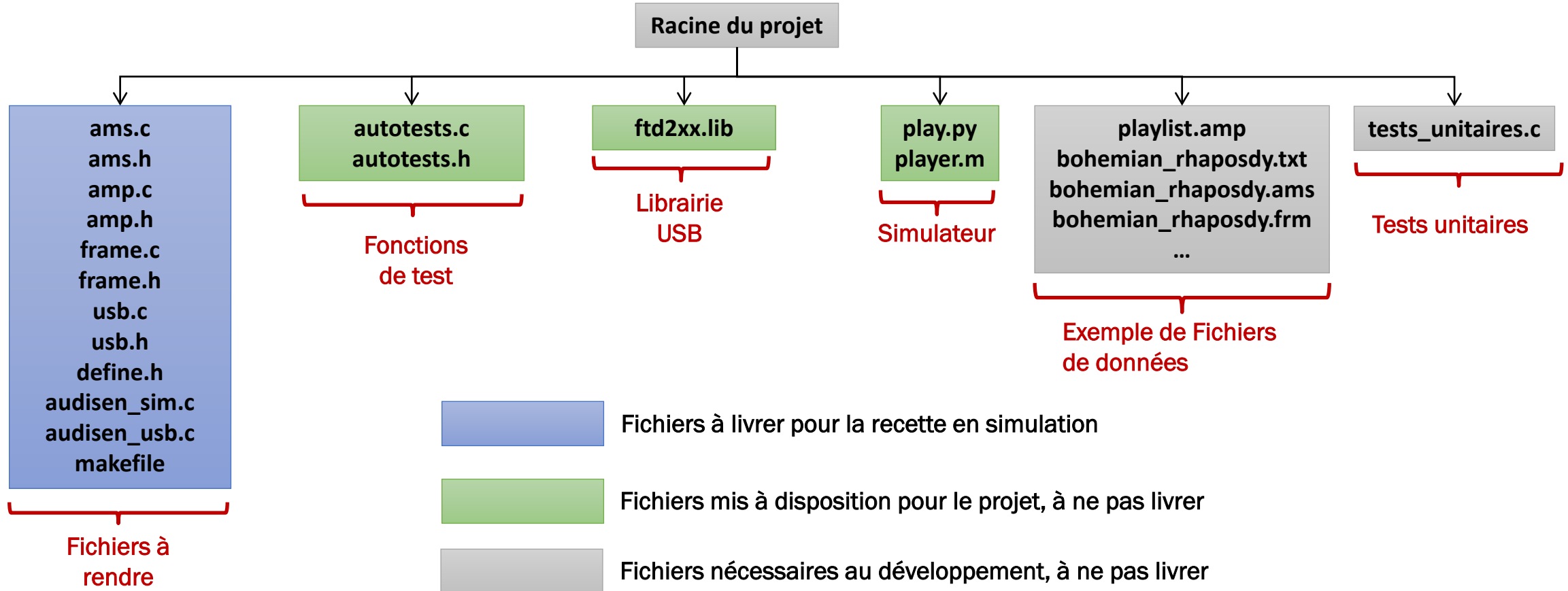




Développement



Arborescence



Définitions

```
#define MAX_SIZE_TITLE 40
#define MAX_SIZE_LINE 190
#define MAX_NUMBER_TICKS 999

typedef struct tick{
    int accent;//accentuation du tick (0=Non, 1=Oui)
    int note[4];// Tableau de 4 notes (0 à 60)
}s_tick;

typedef struct song{
    int tpm;// ticks par minute
    int nTicks; // Nombre de ticks dans le morceau
    char title[MAX_SIZE_TITLE];// Titre du morceau
    struct tick tickTab[MAX_NUMBER_TICKS]; // Tableau de ticks
}s_song;
```

Fonctions à développer

Nom fonctions	Fichiers	Param	Retour	Description
initAMP	amp.c et amp.h	char* fileName	FILE* pf	Ouverture fichier .amp
readAMP	amp.c et amp.h	FILE* pf, char* songFileName		Renvoie le titre d'une chanson
closeAMP	amp.c et amp.h	FILE* pf		Ferme le fichier .amp
readAMS	ams.c et ams.h	char* fileName	s_song mySong	Lit le fichier .ams d'une chanson
createAMS	ams.c et ams.h	char* txtFileName char* amsFileName		Crée un fichier .ams d'une chanson à partir d'une partition simplifiée décrite dans un fichier .txt
createInitFrame	frame.c et frame.h	s_song mySong, char* frame		Crée la trame d'initialisation
createTickFrame	frame.c et frame.h	s_tick myTick, char* frame		Crée une trame pour un tick
initUSB	usb.c et usb.h		FT_HANDLE myHandle	Ouvre l'USB
writeUSB	usb.c et usb.h	char* frame FT_HANDLE myHandle		Ecrit une trame sur l'USB
closeUSB	usb.c et usb.h	FT_HANDLE myHandle		Ferme l'USB

Fonction readAMP

- Fichier amp

- Un titre par ligne, écrit "normalement" :

Bohemian Rhapsody
Knockin'on Heaven's door

```
void readAMP(FILE * fp, char* song_filename);
```

On garde un pointeur sur le fichier AMP

- Lecture d'une ligne :

- Suppression des majuscules
 - Un espace, une apostrophe ... devient un '_' (à vous de réfléchir aux exceptions)
 - Suppression des ____ multiples pour devenir un seul _
 - 40 caractères max.

Séparer chaque petit traitement dans une fonction

- Retour de la fonction :

- le titre bien formaté en char*.
 - Le titre doit correspondre au nom du fichier AMS

- Créer une partition électronique (<chanson>.ams) à partir d'une partition simplifiée (<chanson>.txt)

...

[illegible]

Fonction createInitFrame

- Signature à respecter :

```
void createInitFrame (s_song mySong, char* frame);
```

- Paramètres:

- Une chanson (struct s_song)
- L'adresse d'un tableau suffisamment grand pour recevoir la trame + un caractère '\0' de fin de chaîne de caractère (utile pour les tests, à ne pas envoyer sur USB)

- Effet : écrit la suite d'octets correspondant à la chanson + '\0' dans le tableau passé en paramètre Ex:

```
#Bohemian Rhapsody,144,16*<checksum><CR><LF>\0
```

Fonction createTickFrame

- Signature à respecter :

```
void createTickFrame(s_tick myTick, char* frame);
```

- Paramètres:

- Un tick (struct s_tick)
- L'adresse d'un tableau suffisamment grand pour recevoir la trame + un caractère '\0' de fin de chaîne de caractère (utile pour les tests, à ne pas envoyer sur USB)
- Effet : écrit la suite d'octets correspondant à la trame + '\0' dans le tableau passé en paramètre

```
#<mode>,<acc>,<n1>,<n2>,<n3>,<n4>*<checksum>*<CR><LF>\0
```


Fonction writeUSB

- Utilisation de la bibliothèque ftd2xx
- Compilation sous Windows (utiliser MinGW)
- Paramètres de la connexion
 - Fonctionne à 9600 Baud,
 - 8 bits de données,
 - Un bit de stop d'une longueur de 1,
 - Pas de contrôle de parité,
 - Pas d'utilisation de « flow control »,
- Pour plus de détails : lire la description détaillée du projet et le tuto d'installation MinGW.

Programmes principaux

- Mode simulation
 - Nom du fichier : audisen_sim.c
 - Le nom de la playlist est paramétré dans le code
 - Intègre les blocs précédents sauf writeUSB()
 - Ecrit les différentes trames dans un fichier <playlist>.frm
- Mode USB
 - Nom du fichier : audisen_usb.c
 - Le nom de la playlist est paramétré dans le code
 - Intègre tous les blocs précédents
 - Communique avec la carte STM32 via le port USB



Déroulé du projet

4 phases



Lundi matin	Lundi après-midi	Mardi matin	Mardi après-midi	Mercredi matin	Mercredi après-midi	Vendredi matin	Vendredi après-midi
Analyse	Réalisation				Intégration		Recette

- ✓ Analyse de chaque bloc
- ✓ Architecture du programme
- ✓ Définition des Types de données

- ✓ Code source de chaque bloc
- ✓ Tests unitaires
- ✓ Makefile

- ✓ Code source programme principal
- ✓ Test intégration avec le simulateur ou avec la carte
- ✓ Validation avec la carte

- ✓ **Code source à rendre avant 15:00**
- ✓ QCM de 15:00 à 15:30
- ✓ Validation automatique et manuelle par l'enseignant sur PC enseignant à partir de 15:30
- ✓ Qualimétrie

Au jour le jour

- Travail en binôme
 - Les binômes sont constitués le premier jour
 - Les binômes à constituer au sein de chaque sous-groupe
 - Les redoublants doivent se mettre ensemble ou en monôme le cas échéant
 - Chaque étudiant connaît l'ensemble du projet
 - Attention à bien se répartir le travail
- Ressources externes
 - Tous les documents sont autorisés...
 - ... mais les informations nécessaires au projet sont fournies !!!
 - La transmission de code entre étudiants de binômes différentes est interdite
 - (un outil anti-plagiat sera utilisé en fin projet)
 - L'utilisation d'outils de génération de codes par intelligence artificielle de type ChatGPT est totalement proscrit et sera sévèrement sanctionné, vous serez également noté sur votre compréhension du code.
- Livraison de code ou de document
 - Ne pas attendre la dernière minute pour poster le livrable
 - Préparer des livrables intermédiaires
 - Sauvegarder régulièrement vos données





Notation



Ce qu'il faut rendre

- Fichiers sources à rendre (au minimum)

amp.c	amp.h	define.h
ams.c	ams.h	makefile
frame.c	frame.h	audisen_sim.c
usb.c	usb.h	audisen_usb.c



sources_xx_yy.zip

Pour vous aider, une coquille vide comprenant l'architecture imposée est disponible sur l'ENT

Barème final (indicatif)

	Coefficient
Recette programme	10
Audit étudiants	10
QCM	10
Total	30

Au Prorata
de la recette



Barème recette (indicatif)

#	Nom du test	Type de test	Fonction de test	A3 hors CIR3	CIR3
1	Lecture d'un fichier AMP	Automatique	testReadAMP()	4	2
2	Lecture d'un fichier AMS	Automatique	testReadAMS()	4	3
3	Création de trames	Automatique	testFrame()	4	2
4	Création d'un fichier AMS	Automatique	testCreateAMS()	4	5
6	Programme global en simulation	Manuel		4	4
7	Programme global en USB	Manuel		4	4
Total				24	20



Pour qu'un bloc soit validé :

- **Les fonctionnalités d'exécution de votre code doivent être correctes (aucun point positif au regard du code)**

Tests automatiques

- Nous vous mettons à disposition un fichier de fonctions (autotests.c/autotests.h) permettant de tester unitairement les différents blocs :
 - testReadAMP()
 - testReadAMS()
 - testFrame()
 - testCreateAMS ()

```
---- Autotest results of block ReadAMP ----
--> test 0 : 1/1
--> test 1 : 1/1
--> test 2 : 1/1
--> test 3 : 1/1
--> test 4 : 1/1
--> test 5 : 1/1
Finish autotest of block ReadAMP => total score : 100.0 %

---- Autotest results of block ReadAMS ----
--> test 0 : 1/1
--> test 1 : 1/1
Finish autotest of block ReadAMS => total score : 100.0 %

---- Autotest results of block Frame ----
--> test 0 : 1/1
--> test 1 : 1/1
--> test 2 : 1/1
--> test 3 : 1/1
Finish autotest of block Frame => total score : 100.0 %

---- Autotest results of block CreateAMS ----
--> test 0 : 1/1
--> test 1 : 9/9
Finish autotest of block CreateAMS => total score : 100.0 %
```



Les ressources

Le matériel

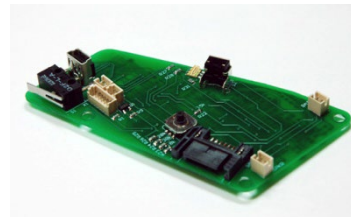
- Votre PC portable
 - Analyse
 - Réalisation
 - Intégration
- Carte électronique
 - Validation de la partie USB



Casque / écouteur
audio déconseillés (consignes non
appliqués = pénalité) !



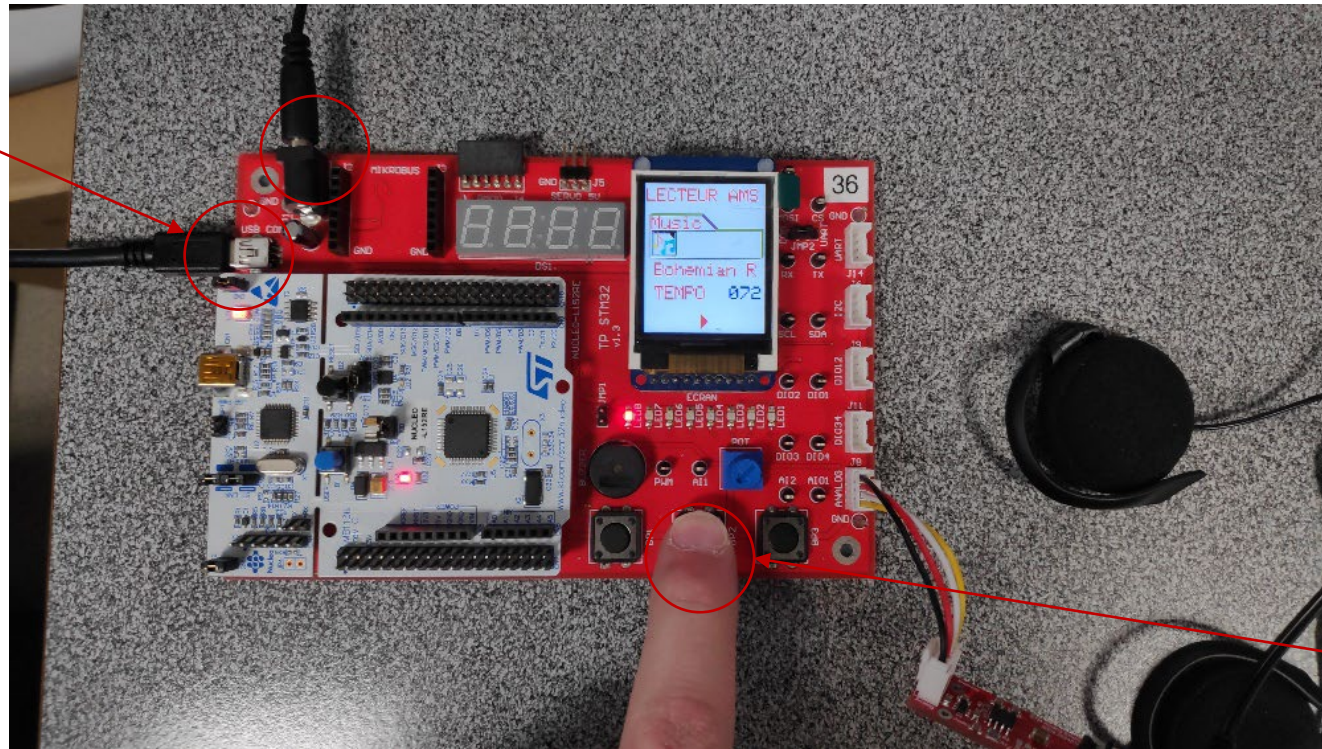
Boissons et nourriture
interdites !



Carte électronique

Alimentation
carte

Communication
USB

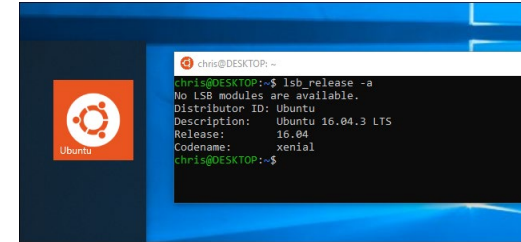


Bouton play

Outils de développement

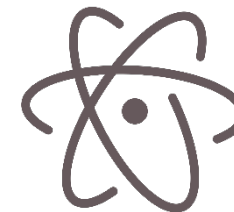
- Noyau de compilation

- Terminal Windows
- MinGW (indispensable pour USB)
- Jupyterhub
- Linux natif
- WSL ...



- Editeur

- Notepad++
- Sublime Text
- Atom
- Visual studio code
- CLION
- ...



Simulateur

- Reproduit le comportement de la carte électronique
- Deux versions disponibles

- Python

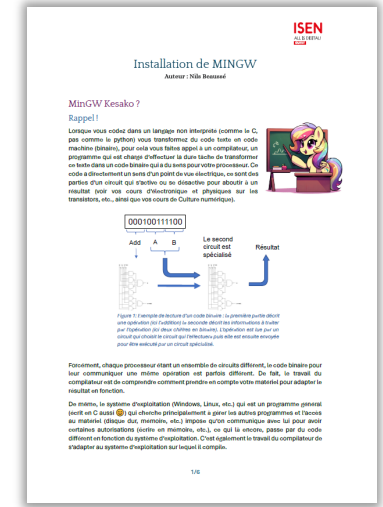
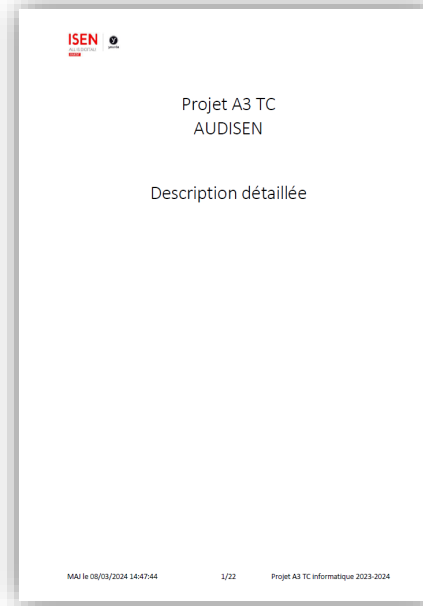


- Matlab



Documentation

- Support de présentation
- Description détaillée
- Guide de programmation FTDI
- Tutoriel d'installation MINGW



ISEN

ALL IS DIGITAL!



yncréa

MERCI
Des questions ?

