

Rapport du Projet PSAR :
Dispositif Autonome de Synthèse Sonore

Encadrant : Hugues Genevois
Cahier de Charges

Pierre Mahé

2 mai 2015

Table des matières

1	Introduction	3
1.1	Presentation du projet	3
1.2	La Carte Udoo	3
1.3	Pure Data	3
1.3.1	Externals	3
1.4	Structure du projet	3
2	Traitement audio	4
2.1	Aquisition audio	4
2.2	Traitement bas niveau	4
2.2.1	Filtrage	4
2.2.2	Détecteur de notes	4
2.2.3	Bandes de fréquences	4
2.3	Extraction des méta-données	4
2.3.1	Mélodie et rythme	4
2.3.2	Pattern minimal	4
3	Récupération de l'environnement	5
3.1	Dispositif et capteurs	5
3.2	Communication inter plate-forme	5
3.3	External pour la communication	6
4	Synthèse musical	7
4.1	Modele physique au longterme	7
4.2	Synthèse implémenté	7
5	Interface Utilisateur	8
5.1	Premiere idée d'implementation	8
5.2	Envoie des données capteur	8
5.3	Envoie d'objet Pure data	8
6	Tutoriel et Documentation	9
6.1	Écriture documentation Pure data	9
6.2	Écriture du Tutoriel d'installation	9
7	Pour aller plus loin	10
7.1	Tests en environnement reel	10
7.2	Tests énergétiques	10
7.3	Serveur distant	10

1 Introduction

1.1 Contexte Global

Maîtrise d'œuvre : Hugues GENEVOIS

Maîtrise d'ouvrage : Pierre MAHÉ

Le projet DASS, Dispositif Autonome de Synthèse Sonore, est une collaboration de Michel RISSE, compositeur de "Décor Sonore" ayant à son actif de nombreuses créations et dispositifs sonores, et de Hugues GENEVOIS, chercheur au LAM de L'Institut Jean le Rond d'Alembert.

Le projet consiste à mettre en place un dispositif générant du son en prenant en compte son environnement dans le but de souligner les sons présents dans cet environnement. Le dispositif recueille les informations à l'aide de capteurs divers permettant de récupérer les bruits ambiants, la luminosité, la température.

Dans un premier temps devrait être la base pour une installation artistique, se déroulant en juillet prochain, dans le cadre d'un festival de musique à Noirlac. Dans un second temps Hugues GENEVOIS et Michel RISSE voudraient réaliser une installation plus importante avec de nombreux dispositifs avec pour chacun d'eux des comportements et des caractéristiques propres, pour voir l'interaction qu'ils pourraient avoir ensemble et les comportements qui en émergeraient.

1.2 Présentation de la demande

Dans le cadre de mon projet PSAR, mon but était de créer un premier prototype du projet pour s'en servir comme base par la suite.

J'avais donc pour mission de créer un dispositif portable interagissant avec son environnement. Ce dispositif doit capter l'environnement à l'aide de capteurs divers. Le but du projet est de créer un dispositif portable interagissant avec son environnement. Ce dispositif doit capter l'environnement à l'aide de capteurs : micro, luminosité, température, et humidité. Il devra traiter ces données pour en extraire des informations pertinentes sur les changements se produisant autour de lui (chants d'oiseaux, passage d'une personne au alentours...). Grâce à ces informations, il devra synthétiser du son.

La synthèse devra être paramétrable, pour permettre au compositeur de modifier l'influence des différents capteurs sur la synthèse sonore.

De plus une interface utilisateur devra être présente permettant au compositeur de simuler des données captées. Cela lui permettra de faire des expérimentations de sa composition.

Dans l'optique de la reproduction de ce dispositif pour des installations plus important. Une procédure devra permettre à un utilisateur de pouvoir reproduire l'ensemble du système, aussi bien niveau matériel que logiciel.

Le dispositif devant être portable, le programme doit fonctionner sur une nano-ordinateur de type `Udoo Quad`. De plus le langage du programme principal doit être le `PureData`, pour assurer une maintenabilité et une extensibilité plus facile.

1.3 La Carte Udoo

La carte Udoo est un nano ordinateur embarquant un processeur ARM et la connectique d'un ordinateur classique. Avec une entrée et sortie micro. La qualité étant suffisante pour notre projet.

La carte utilise une carte micro sd pour stocker ses données, ce qui est également un avantage pour le projet car cela permet de dupliquer la partie software du dispositif avec une simple copie de carte mémoire.

La caractéristique de cette carte est d'être compatible arduino et d'offrir les mêmes connectique qu'une Arduino Mega ce qui permet de simplifier la réception des capteurs ainsi que de l'utiliser les bibliothèques déjà implémenter par la vaste communauté Arduino.

1.4 Pure Data

1.4.1 Externals

1.5 Structure du projet

2 Traitement audio

2.1 Aquisition audio

2.2 Traitement bas niveau

2.2.1 Filtrage

2.2.2 Détecteur de notes

2.2.3 Bandes de fréquences

2.3 Extraction des méta-données

2.3.1 Mélodie et rythme

2.3.2 Pattern minimal

3 Récupération de l'environnement

L'un des avantages de la carte Udoo est d'être compatible Arduino est de proposer les même connectique qu'une Arduino Mega. Cela permet d'utiliser le même langage, d'utiliser les bibliothèques déjà écrites ainsi que les mêmes capteurs. De plus la partie micro contrôleur étant connecté directement à la partie nano ordinateur via un bus spécial, la vitesse de communication est plus rapide et avec une plus courte latence par rapport à une connexion usb classique.

Dans les valeurs envoyés par un capteur deux choses sont intéressantes, la première la valeur et la variation de celle-ci (si elle augmente ou si elle diminue).

3.1 Dispositif et capteurs

Dans le cadre du projet, seuls trois capteurs étaient prévus mais le programme devrait permettre un ajout de capteurs aisé.

Les capteurs utilisés sont des capteurs les plus basiques : un capteur de luminosité, de température et d'humidité.

Pour capter la luminosité, nous avons utilisé une photo-résistance, son fonctionnement est assez simple, plus la luminosité va être forte plus la résistance interne du composant va être faible. Il suffit de récupérer la tension aux bornes du composant pour connaître l'indice de luminosité.

Malheureusement cette indice ne permet pas de connaître l'intensité lumineuse en Candela ou en Lux mais permet d'avoir une échelle d'avoir des valeurs relatives. Ce qui est suffisant dans le cadre de ce projet.

Pour la Température et l'humidité le fonctionnement est très similaire. Contrairement à la luminosité grâce à une formule (dépendant de chaque capteur) il sera possible d'avoir la température ou l'hygrométrie précise.

3.2 Communication inter plate-forme

Pour la partie micro-contrôleur l'envoi de données est assez intuitive car il suffit d'écrire sur le port Série (Serial port), avec des primitives d'écriture basiques.

Pour la partie ordinateur avec le logiciel Pure Data, il faut utiliser une boîte pour se connecter à un périphérique puis on peut lire et écrire sur celui-ci.

La connexion entre l'Arduino et Pure Data devant être unique, il a fallu trouver un moyen de pouvoir différencier les données des différents capteurs.

Pour communiquer entre les deux parties, j'ai donc défini un micro-protocole qui étiquette les données envoyés.

Tous les X secondes le programme arduino lit les valeurs des capteurs et envoie un message de la forme :

```
1 NOM_CAPTEURvl: VALEUR;    //vl pour la valeur
2 MON_CAPTEURvr: VALEUR;    //vr pour la variation
```

Pour ajouter un capteur, il suffit d'écrire sur le port série avec un nom de capteur pas encore utilisé et c'est dans la partie Pure data que nous traiterons la nouvelle étiquette.

Puis pour il nous est venu l'idée de permettre à Pure data de pouvoir changer le délai entre chaque envoi de valeur. Pour cela il a fallu permettre à l'arduino de pouvoir recevoir des données avec des messages ayant la même forme que l'envoi de donnée avec l'étiquette **delay**.

C'est à ce moment là que nous avons découvert un vice de Pure Data, étant un langage très faiblement typé. Les données reçues de l'arduino sont des octets que Pure Data caste en type Nombre et il n'existe pas d'objet permettant à partir de ce flux d'octets de récupérer des entiers (valeur des capteurs envoyé par la parité arduino).

Au premier abord nous avons voulu utiliser l'objet **PackOSC** qui permet à la base de convertir un message quelconque en commande OSC (**O**pen **S**ound **C**ontrol). Ce protocole étant un protocole avec un codage ascii nous pouvions l'utiliser pour encoder et decoder les communications avec l'arduino.

Mais cela aurait été une manipulation de l'objet de base de plus cela pourrait porter à confusion car nous ne gérons en rien le protocole.

Nous avons discuté avec Hugues GENEVOIS et nous avons utilisé de programmer nos propres externes.

3.3 External pour la communication

En s'inspirant du fonctionnement des objets **PackOSC** et **UnPackOSC**, nous avons programmé deux externes, un qui reçoit un message et qui le transforme en tableau d'octets (en code ascii plus particulièrement) en ajoutant le code ascii d'un point virgule à la fin.

Le second reçoit un flux d'octets et le stocke dans un tableau jusqu'à recevoir un point virgule, à ce moment là il envoie toute la chaîne reçue.

Le fait d'utiliser le caractère ';' n'est pas un problème car en PureData ce caractère est généralement le délimiteur de fin de chaîne, la fin de liste et de tableau ainsi qu'internes pour fin des messages tcp ainsi que la fin d'une lecture de fichier.

4 Synthèse musical

4.1 Modèle physique au longterme

4.2 Synthèse implémenté

5 Interface Utilisateur

5.1 Premiere idée d'implementation

5.2 Envoie des données capteur

5.3 Envoie d'objet Pure data

6 Tutoriel et Documentation

6.1 Écriture documentation Pure data

6.2 Écriture du Tutoriel d'installation

7 Pour aller plus loin

7.1 Tests en environnement reel

7.2 Tests énergétiques

7.3 Serveur distant

8 Bibliographie

Références

- [1] Andéa-Novel Brigitte, Fabre Benoit, and Jouvelot Pierre. *Acoustique-Informatique-Musique*. Presses des Mines, 2012.
- [2] Leipp Émile. *Acoustique et Musique*. Presses des Mines, 2010.
- [3] Thomas Grill. Pure data patch repository, 2008(accessed mars, 2015).
- [4] Hans. pd, 2006 (accessed avril, 2015).
- [5] Adam Hyde. Pure data, (accessed mars, 2015).
- [6] Laurent Millot. *Traitement du signal audiovisuel, Applications avec Pure Data*. Dunod, 2008.