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

1.1 Presentation du projet

1.2 La Carte Udoo

1.3 Pure Data

1.3.1 Externals

1.4 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'etre compatible Arduino est de proposer les même connectique qu'une Arduino Mega. Cela permet d'utiliser le même langage, d'utiliser les bibliotheques deja écrites ainsi que les mêmes capteurs.
De plus la partie micro controleur etant connecté directement a la partie nano ordinateur via un bus special, la vite de comunication est plus rapide et avec une plus courte latence par rapport a une connexion usb classique.

Dans les valeurs envoyés par un capteur deux choses sont interessante, la premiere la valeur et la variation de celci (si elle augmente ou si elle diminue).

3.1 Dispositif et capteurs

Dans le cadre du projet, seul trois capteurs étaient prevu mais le programme devrait permettre un ajout de capteurs aisé.
Les capteurs utilisés sont des capteurs les plus basique : un capteur de luminosité, de temperature et d'humidité.
Pour capteur la luminosité, nous avons utilisé une photo-resistance, son fonctionnement est asses simple, plus la luminosité va etre forte plus la resitance interne de composant va etre faible. Il suffit de recupere la tension au borne de composant pour connaitre l'indice de luminosité.
Malheureusement cette indice ne permet pas de connaitre l'intancité lumineuse en Candela ou en Lux mais permet d'avoir une echelle d'avoir des valeur relavite. Ce qui est suffisant dans le cadre de ce projet.

Pour la Temperature et l'humidité le fonctionnement est tres similaire. Contrairement a la luminosité grace a une formule (dépendent de chaque capteur) il sera possible d'avoir la température ou l'hygrométrie précise.

3.2 Communication inter plate-forme

Pour la partie micro-controleur l'envoi de donnnées est asses intruitive car il suffit d'ecrire sur le port Série (Serial port), avec des primite d'ecriture basique.
Pour la partie ordinateur avec le logiciel Pure Data, il faut utiliser une boite pour se connecter a un périphérique puis on peut lire et ecrire sur celui.
La connexion entre l'arduino et Pure data devant etre unique, il a fallu trouver un moyen de pouvoir differensier les données des différant capteur.

Pour communiquer entre les deux parties, j'ai donc define un micro-protocole qui étiquette les données envoyés.
Tous les X secondes le programme arduino li les valeurs des capteurs et envoi un message de la forme :

```
NOM_CAPTEURvl: VALEUR;      //vl pour la valeur
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 utiliser et c'est dans la partie Pure data que nous traiterons la nouvelle étiquette.

Puis pour il nous ai venu l'idée de permettre à Pure data de pouvoir changer le délai entre chaque envoi de valeur. Pour cela il a fallu permettre a 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 a ce moment la que nous avons découvert un vice de Pure Data, étant un langage tres faiblement typé. Les donnée recu de l'arduino sont des octets que Pure Data caste en type Nombre et il n'existe pas d'objet permettant a partir de ce flux d'octets de recuperer des integers (valeur des capteurs envoyé par la parite arduino).

Au premiere abord nous avons voulu utiliser l'objet `PakcOSC` qui permet a la base de convertir un message quelconque en commande OSC (`Open Sound Control`). Ce protocole etant un protocole avec un codage ascii nous pouvions l'utiliser pour encoder et decoder les communications avec l'arduino.
Mais cela aurai ete une manipulation de l'objet de base de plus cela pourrai porter a confusion car nous ne gerions en rien le protocole.
Nous avons discuté avec Hugues GENEVOIS et nous avons utilisé de programmer nos prope externals.

3.3 External pour la communication

En s'inspirant du fonctionnement des objets `PackOSC` et `UnPackOSC`, nous avons programmé deux externals, un qui recoi un message et qui le transforme en tableau d'octects (en code ascii plus particulierement) en ajoutant le code ascii d'un point virgule a la fin.
Le second recoi un flux d'octets et le stock dans un tableau jusqu'a recevoir un point virgule, a ce momment là il envoi toute la chaine récu.
Le fait d'utiliser le caractere ';' n'est pas un probleme car en PureData ce caratere est generalement le delimiter de fin de chaine, la fin de liste et de tableau ainsi quand interne pour fin des messages tcp ainsi que la fin d'une lecture de fichier.

4 Synthèse musical

4.1 Modele 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.