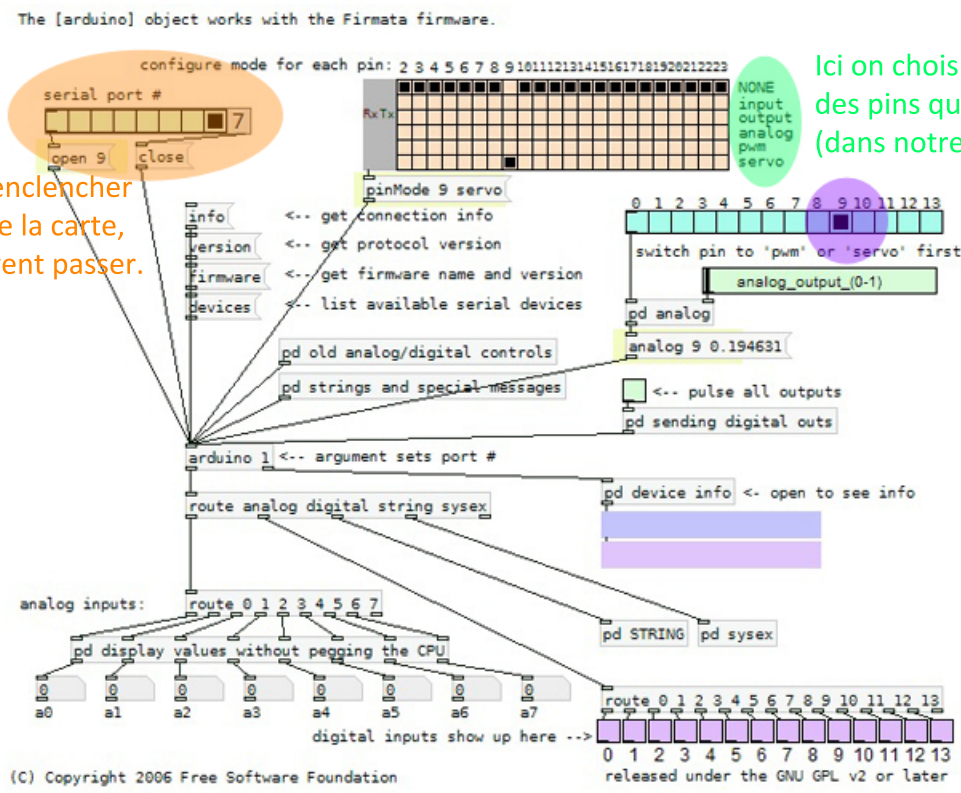
Tutoriel du programme de contrôle des moteurs

• ThêtaFantômes •

On a besoin d'une carte Arduino, de servomoteurs, d'un casque EPOC.
Pour les logiciels de programmation on utilisera PureData.

Tout d'abord nous devons faire «communiquer» PureData et la carte arduino. Pour cela nous allons utiliser le patch Firmata.

Ici on choisit le port où est branché la carte Arduino.
open (n°) permet d'enclencher le fonctionnement de la carte, le informations peuvent passer.



Ici on choisit la fonction des pins qui sont branchées. (dans notre cas : mode Servo)

On choisit ici la pin du moteur que l'on souhaite contrôler

Programme Firmata de base.

Pour le moment les valeurs reçues par le moteur sont données manuellement via la glissière. Pour notre projet le but est que ce soit les valeurs reçues par les capteurs du casque qui fassent bouger les moteurs.
Il faut maintenant faire communiquer le patch Firmata et le casque EPOC.
Pour cela nous allons utiliser le pilote du casque : EMOKIT.
Nous allons utiliser le programme Emokit_OSC pour envoyer les valeurs des capteurs au patch Firmata.

Dans ce «kit» nous avons un patch PureData qui reçoit les valeurs de chaque capteur du casque en temps réel.

Programme OSC

emokit_osc.cpp

```
/*
Simple example of sending an OSC message using oscpack.
*/

#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <csignal>
#include <iostream>
#include "oscpack/osc/OscOutboundPacketStream.h"
#include "oscpack/ip/UdpSocket.h"
#include "emokit/emokit.h"

#define ADDRESS "127.0.0.1"
fine PORT 9997

#define OUTPUT_BUFFER_SIZE 4096
```

Adresse avec laquelle le programme OSC va communiquer. Ici on mets l'adresse de loopback de l'ordinateur. On pourrait mettre l'adresse IP d'un autre ordinateur si on en utilisait un différentpour le patch Firmata et le patch du casque.

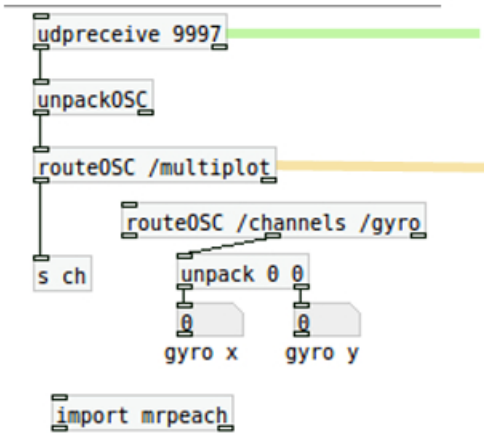
C'est le numéro du port dans le PC à travers lequel le programme OSC va faire passer les messages.

```
        int r;
        if((r=emokit_read_data_timeout(d, 1000)) > 0)
        {
                frame = emokit_get_next_frame(d);
                osc::OutboundPacketStream p( buffer, OUTPUT_BUFFER_SIZE );
                p << osc::BeginMessage( "/multiplot" )
                  << conv(frame.F3) << conv(frame.FC6) << conv(frame.P7)
                  << conv(frame.T8) << conv(frame.F7)  << conv(frame.F8)
                  << conv(frame.T7) << conv(frame.P8)  << conv(frame.AF4)
                  << conv(frame.F4) << conv(frame.AF3) << conv(frame.O2)
                  << conv(frame.O1) << conv(frame.FC5) << osc::EndMessage;

                transmitSocket.Send( p.Data(), p.Size() );
        } else if(r == 0)
                fprintf(stderr, "Headset Timeout\n");
        else {
                fprintf(stderr, "Headset Error\n");
                break;
        }
}
```

C'est l'adresse OSC des données. On l''a retrouve pour l'envoi des données et leur réception.

Une fois que l'on a configurer le programme OSC, on peut aller configurer le patch PureData : Emokit_osc.



On vérifie que le N° du port correspond à celui du programme OSC

On vérifie que l'on a bien la même adresse OSC des données en réception

Maintenant nous allons pouvoir recevoir les données des capteurs du casque.

Mise en marche du casque

Nous allons pouvoir mettre en marche le casque.
En premier nous vérifions qu'il est chargé. Sinon nous le mettons en charge via le câble USB fourni.
Il faut mettre la clé USB de réception sur l'ordinateur. On allume le casque.
Nous fonctionnons sous linux. Pour pouvoir activer la connexion nous allons utiliser des commandes dans le terminal.

- On se rends dans le dossier où se trouve le programme emokit_osc

pour cela on tape :

cd (le chemin du dossier)

- une fois dans le dossier nous pouvons activer la connexion de la manière suivante :

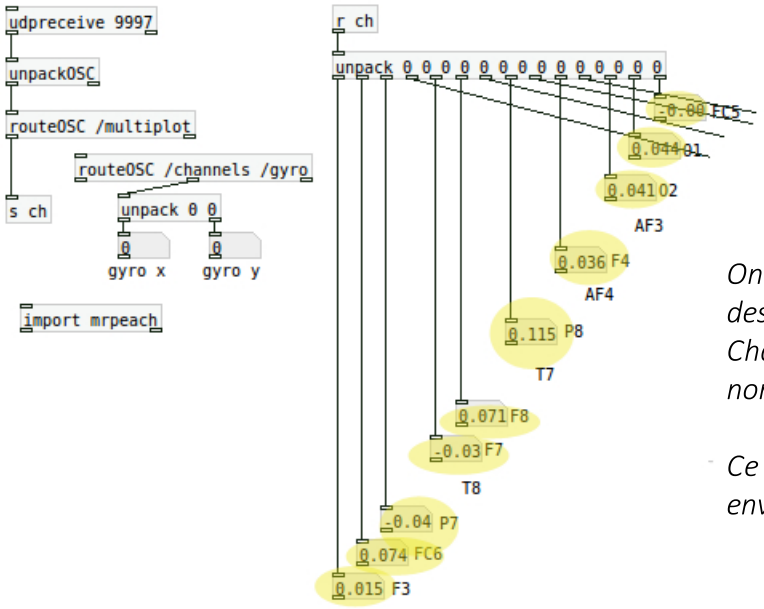
./emokit_osc

Maintenant on peut voir que le terminal nous écrit :

current epoc devices connected : 2

→ Le casque est connecté.

On va maintenant ouvrir le patch PureData : emokit_osc .



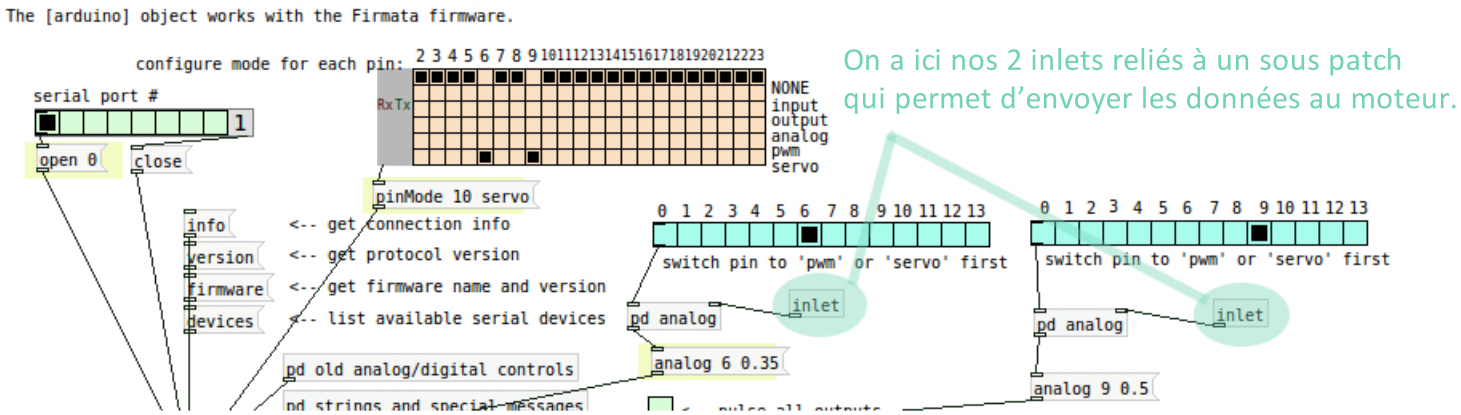
On a ici toutes les valeurs des capteurs du casque. Chaque capteur a son nom prédéfini. (AF3 / T8 / FC6 ...)

Ce sont ces valeurs que nous allons envoyer vers le patch Firmata.

Nous allons pouvoir maintenant passer à la transmission des données entre le patch emokit_osc et le patch Firmata.
Nous allons créer un sous patch dans le patch Emokit_osc.
Ce sous patch contiendra les éléments du patch Firmata.

On commence par créer un objet [pd nomdusouspatch] dans le patch emokit_osc. Une nouvelle fenêtre s'ouvre. On va y mettre le patch Firmata à l'intérieur.
On ajoute les objets [inlet] nécessaire. Pour commencer nous en avons mis 2. (pour réceptionner les valeurs de 2 capteurs).

Transmission des valeurs aux moteurs

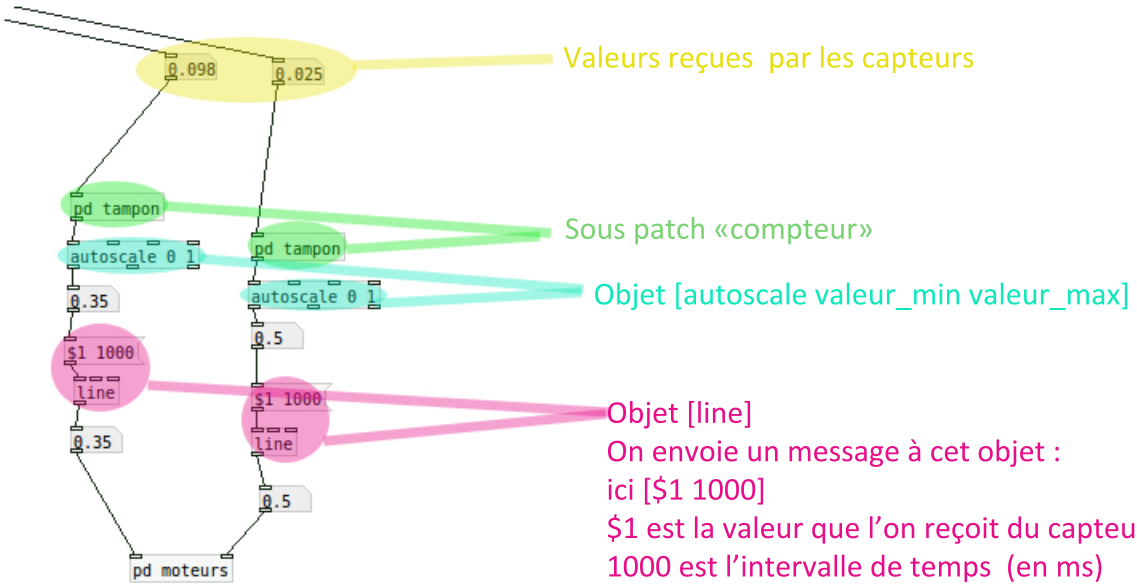


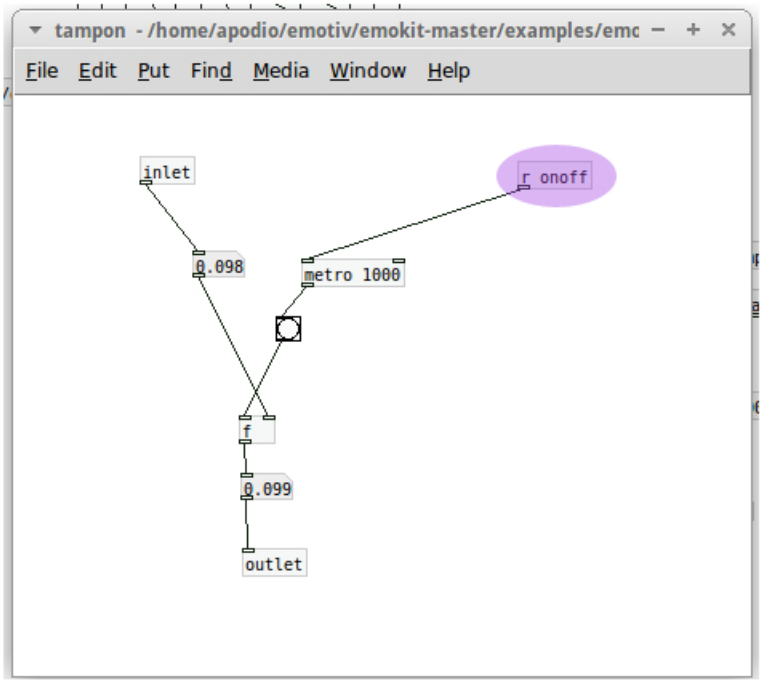
On a ici la carte Arduino sur le port 0 et les 2 moteurs sur les pins 6 et 9. Dans ce cas là les valeurs des capteurs sont brutes et n'ont pas été modifiées. Les moteurs n'avaient pas de mouvement car les valeurs des capteurs varient trop peu et trop rapidement. Il faut donc faire des modifications pour que la valeur varie plus lentement et que les écarts d'une valeur à l'autre soient plus importants.

Pour palier à ce problème nous allons rajouter différents objets dans le patch parent (emokit_osc)

En premier pour que la valeur change moins rapidement nous avons décidé d'utiliser un compteur (objet [metro]) afin que l'on réceptionne la valeur à intervalle régulier. Par exemple toutes les secondes. Pour cela nous avons créer un autre sous patch.
Ensuite pour que les écarts soient plus grand nous avons utiliser l'objet [autoscale] ce qui permet de maper les valeurs reçu dans un certain intervalle. Ici nous avons choisi entre 0 et 1. Ce sont la valeurs minimale et maximale de mouvement des moteurs que nous avons.

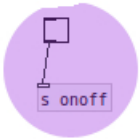
Nous avons ensuite réalisé de nouveaux essais. Un nouveau paramètre s'est présenté : pour passer d'une position à une autre le moteur avait une vitesse trop rapide.
Pour que le mouvement soit plus lent et plus fluide il faut utiliser l'objet [line]. Il permet de donner un temps précis pour passer d'une valeur à une autre.





Ici on a le sous patch «compteur»
Il comporte un objet [metro] qui permet d'envoyer un signal à intervalle régulier.
On a un objet [float] f il lit la valeur qu'il reçoit et il l'a renvoitsi il reçoit un bang du metro.
On mets le Bang du metro sur l'entrée chaude et la valeur du capteur sur l'entrée froide.
De cette manière la valeur du casque ne sera renvoyée dans le patch parent que toutes les secondes.

On a aussi un objet [r nomdonné] c'est un «receive»
Ici il va nous permettre de déclencher le compteur sans avoir besoin de rentrer dans le sous patch.
On place un objet [s nomdonné] dans le patch parent c'est un «send».
Il faut donner les mêmes noms aux deux objets et ils seront reliés.
Ici on a [r onoff] et [s onoff]



Objet [s onoff] dans le patch parent, on a ajouté un interrupteur pour enclenché le «send» qui va transmettre un «bang » au receive. Le compteur sera lancé..