Antonin Segault
Cédric Lanoir

Projet de fin d'études

Documentation
version 2.42

Master 2 PSM – 2012/2013
UFR STGI – Université de Franche Comté

Sommaire

Introduction

Cette documentation, plus qu'un simple recueil technique, traite dans son ensemble de la conception et de la réalisation du Sy~, une installation sonore interactive créée dans un cadre étudiant.

Cette installation a été conçue et réalisée par Cédric Lanoir et Antonin Segault, tous deux étudiants en deuxième année de Master PSM (Produits et Services Multimédia) à l'Université de Franche-Comté à Montbéliard (France). C'est après une Licence SLIC, parcours Communication Multimédia (Sciences du Langage, de l'Information et de la Communication) qu'ils rejoignent la première année de Master où ils font leurs premiers pas dans le parcours Musique et Son. C'est dans ce même parcours qu'ils entament leur deuxième année qui a pour objet principal le PFE (Projet de Fin d'Etudes).

Souvent comparé aux chef-œuvres des apprentis artisans, le PFE clôt une formation entièrement consacrée aux métiers du Multimédia et a pour challenge la conduite d'un projet, de sa phase de conception à sa réalisation, tout en mettant l'accent sur l'autonomie et la gestion de projet. Les étudiants impliqués sont libres de choisir leur sujet et de former des groupes selon leur convenance mais en gardant à l'esprit les compétences nécessaires à la réalisation du projet.

Ce dossier présente tout d'abord le contexte de développement du projet : les aspects historiques, le sujet initial et les personnes impliquées. Il décrit ensuite les étapes de conception avant de détailler les spécifications techniques finales, tant matérielles que logicielles. Enfin, après un guide détaillé du déploiement de l'installation, il conclue sur les apports de ce projet et les nombreuses opportunités qu'il reste à exploiter.

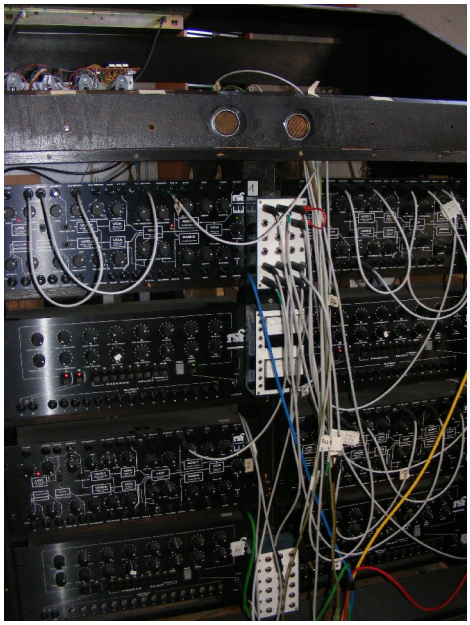
Le projet

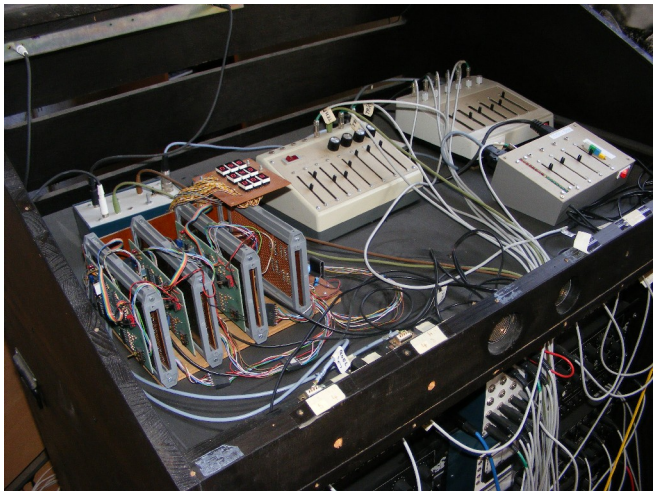
SYHAMO

De la même façon que l'on ne plonge jamais deux fois dans le même fleuve (Héraclite), l'écoute d'une musique sera différente d'une fois sur l'autre puisque l'environnement d'écoute sera différent. En 1989, le musée d'Olten (Suisse) a fait appel à plusieurs artistes autour du thème "Comment l'environnement influence-t-il à la fois l'objet et la perception du spectateur ou de l'auditeur ?". En réponse, Oscar Wiggli, sculpteur suisse reconnu internationalement et compositeur de musique contemporaine, a conçu le SYHAMO, ensuite construit par Janine Wiggli, sa femme.

"La vie des sons est influencée par l'entourage et par nous."
-- OscarWiggli

Le SYHAMO, SYstème Hybride Audio contrôlé par Ordinateur (le M facilite la prononciation), est une installation sonore (lutherie électronique) qui interprète une composition en temps réel en fonction de son environnement.





Le sujet

« Le projet vise à développer un nouveau SYHAMO utilisant la plate-forme Arduino pour la captation de l'environnement et Max pour la synthèse sonore. Tout en modélisant le fonctionnement structurel de votre instrument sur le SYHAMO, vous développerez l'aspect sonore avec la plus grande indépendance et originalité. »

Sy~

Nous avons donc recréé ce SYHAMO à l'aide des technologies actuelles en nous axant sur des outils numériques. Bien que certains aspects de la base matérielle soit restés sensiblement les mêmes (capteurs), une grande simplification a pu être obtenue grâce à de nouvelles technologies comme la carte Arduino, qui nous sert d'interface entre l'environnement et l'ordinateur. De même, Pure Data (logiciel de programmation graphique pour la création musicale et multimédia en temps réel) a permis de remplacer efficacement une bonne partie du SYHAMO tels que les synthétiseurs et les programmeurs. Ainsi, les fonctionnalités de ce système complexe et massif de jadis peuvent, à présent, être contenues dans une boîte à chaussures (matériel) et un ordinateur (logiciel).

Pour documenter l'avancement du projet, nous avons créé le site Web http://sy.retrodev.net. Nous avons choisi d'y diffuser l'intégralité de nos productions (dossiers, schémas, codes) selon les termes de la licence « Creative Common Paternité - Partage dans les Mêmes Conditions 3.0 France ». Ainsi, chacun est libre de les réutiliser, modifier, redistribuer, à condition de le faire sous la même licence. C'est une manière de remercier les communautés à l'origine des outils libres sur lesquels repose notre travail (Arduino, PureData, Framapad...), mais également d'aider à la diffusion de notre création.

L'équipe

Cédric Lanoir

Grand passionné de musique, notamment expérimentale et progressive, et lui-même musicien, c'est avec enthousiasme qu'il rejoint ce Projet de Fin d'Etudes qui lui permettra d'aborder de nouveaux aspects de cet art sonore et d'acquérir de nouvelles compétences tant dans des domaines théoriques (théorie des installations sonores...) que pratiques (électronique, programmation musicale...).

Antonin Segault

Musicien depuis son enfance, ce n'est que tardivement qu'il a découvert l'informatique. Le projet a été l'occasion réunir son goût pour les synthétiseurs analogiques et sa passion des logiciels libres. Après le Master, Antonin souhaite préparer une thèse pour devenir enseignant-chercheur dans le multimédia.

Jean-François Charles

Professeur au conservatoire et un des principaux intervenants du parcours Musique et Son pour le Multimédia du master PSM, il est à l'initiative du projet et, par conséquent, a été notre tuteur durant toute sa réalisation.

Alexandre Coutant

Maître de Conférence à l'Université, il est le responsable des Projets de Fin d'Etudes au département multimédia. Il a suivi la conception et la réalisation du projet, et nous a aiguillé pour l'organisation du travail.

Oscar et Janine Wiggli

Oscar Wiggli est un sculpteur Suisse célèbre pour son travail sur les formes forgées monumentales. Egalement compositeur/musicien, il a créé plusieurs albums de musiques électroacoustiques. En 1989, suite à une commande du musée d'Olten (Suisse) il conçoit le SYHAMO; c'est sa femme, Janine Wiggli qui le construira, allant jusqu'à suivre des formations d'électronique et d'informatique pour compléter ses compétences.

Lorenzo Bianchi

Professeur de musique électroacoustique au conservatoire de Montbéliard, c'est au cours de ses interventions au département multimédia qu'il nous a aidé à travailler efficacement et qu'il a apporté son expérience à la partie logicielle. Il est également à l'origine de la réflexion sur la spatialisation du son.

Giacomo Platini et Jacopo Baboni Schilingi

Professeurs au conservatoire de Montbéliard, ils ont animé les cours en lien avec les installation sonores au département multimédia tout au long de l'unité d'enseignement "Cognition et écritures musicales". Ils nous ont guidés dans la réflexion sur l'interactivité.

Philippe Canalda et Samir Chouali

Tous deux Maîtres de Conférence à l'université, ils ont orienté l'élaboration du dossier de conception, à l'occasion des cours de Méthodologie de Conception de Produits et Services Multimédia (MCPSM).

François Lachambre

Professeur de Physique Appliquée au lycée Jules Viette, à Montbéliard, et responsable du nouveau BTS d'électrotechnique, il nous a guidé pour la conception et la réalisation du circuit imprimé nécessaire à l'installation.

A toutes ces personnes, nous exprimons nos plus sincères remerciements pour leur aide et leur implication, de près ou de loin, dans ce projet.

Conception

Recherches

La première étape du projet fut de rassembler un maximum d'informations à propos du SYHAMO et de sa construction, afin de partir d'un bon pied dans sa réalisation. Ainsi, nous avons lu et étudié plusieurs ouvrages dédiés à cette installation sonore, empruntés au centre de ressources du conservatoire de Montbéliard. Ces premières informations, couplées à nos recherches sur le Web, n'ont pu être complètes qu'à la suite de notre visite à la fondation Wiggli. Après un trajet sur les routes vallonnées de la Suisse nous avons rencontré Janine et Oscar Wiggli qui nous en ont dit plus sur le SYHAMO, sa technologie et son histoire, ressortant pour l'occasion l'énorme machine noire de son coin poussiéreux.

Livrables

Plusieurs livrables furent rendus afin de définir l'organisation du projet au cours de l'année, les fonctionnalités du produit fini, sa gestion, la répartition des tâches...

Cahier des charges

Dans ce document, nous avons décrit le sujet de ce projet, le travail original sur lequel il est basé, et les contraintes qui nous ont été données pour sa réalisation. Nous avons également présenté notre appropriation du projet à travers nos premiers choix techniques et artistiques. Enfin, nous avons exposé l'organisation de notre collaboration, la planification du travail et la communication de notre production.

Cahier de spécification

Dans ce document, nous avons exposé l'avancée de nos recherches artistiques et techniques. Nous avons présenté et justifié les choix que nous avons fait en fonction du cahier des charges. Enfin, nous avons décrit l'organisation de la phase de production pour les trois mois à venir.

Matériel

Composants

La luminosité est mesurée à l'aide de deux photorésistances. La variation de sa résistance est captée à l'aide d'un pont diviseur de tension, utilisant une résistance de 10kΩ en série. Le même montage est utilisé avec les quatre thermistances permettant de capter les variations de température.

La mesure de la distance est réalisée à l'aide de deux capteurs à ultra-son (sonars). Le modèle que nous avons choisi (MaxSonar XL-EZ0) permet de récupérer directement une information numérique (codée en PWM).

Circuit

Schéma électrique

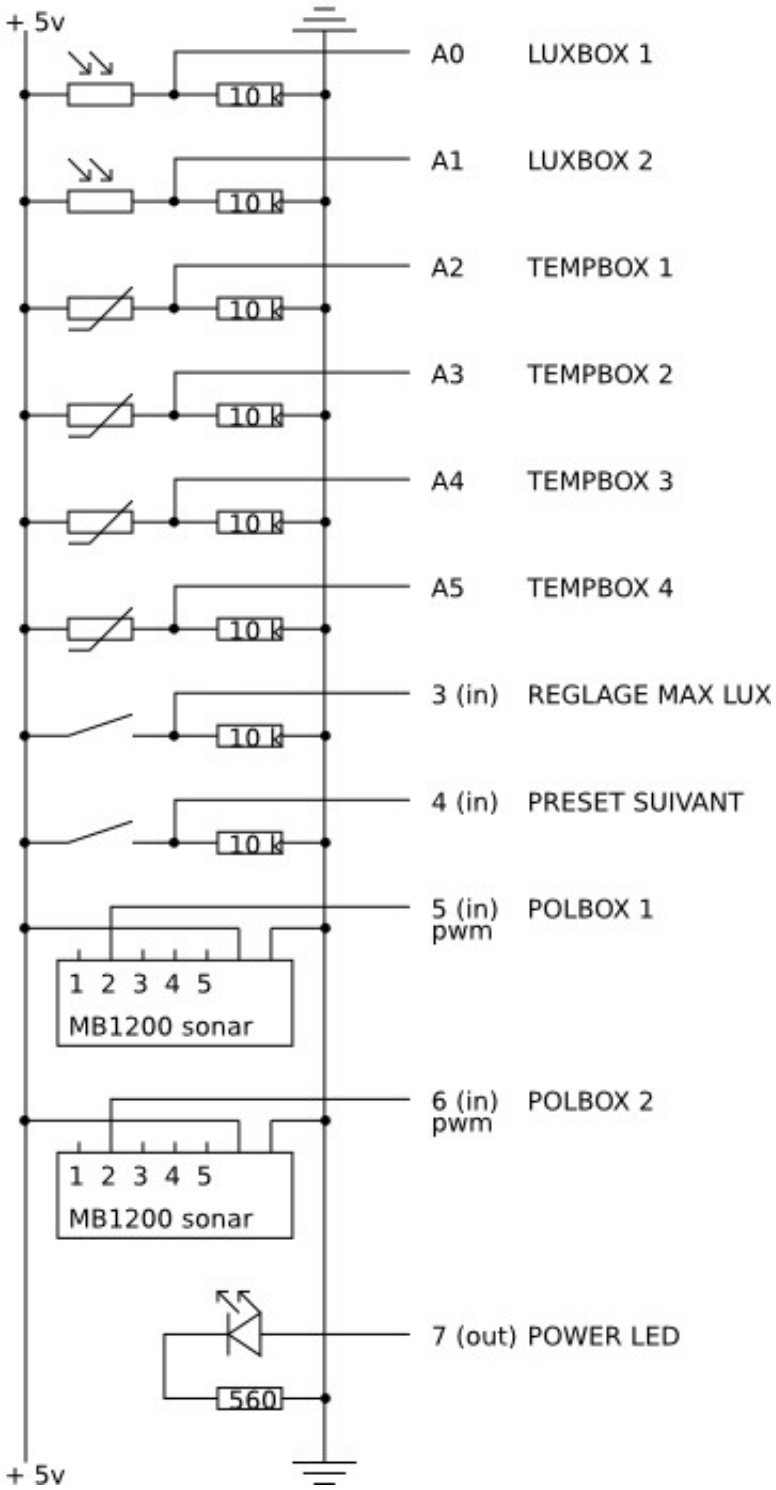
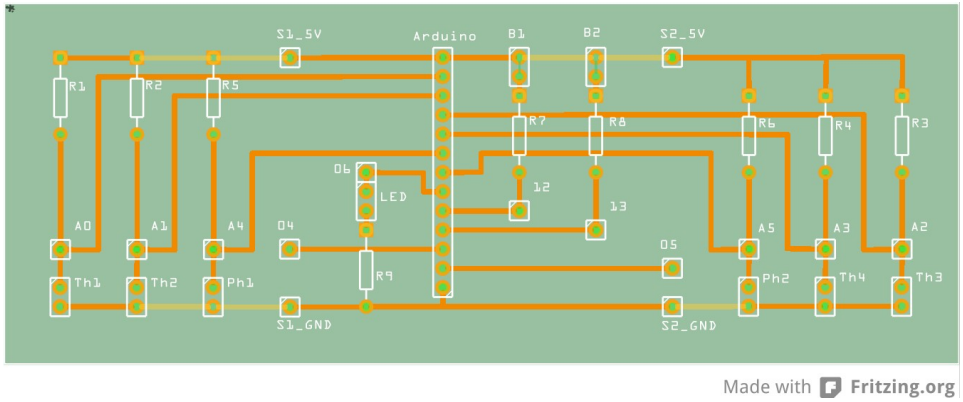


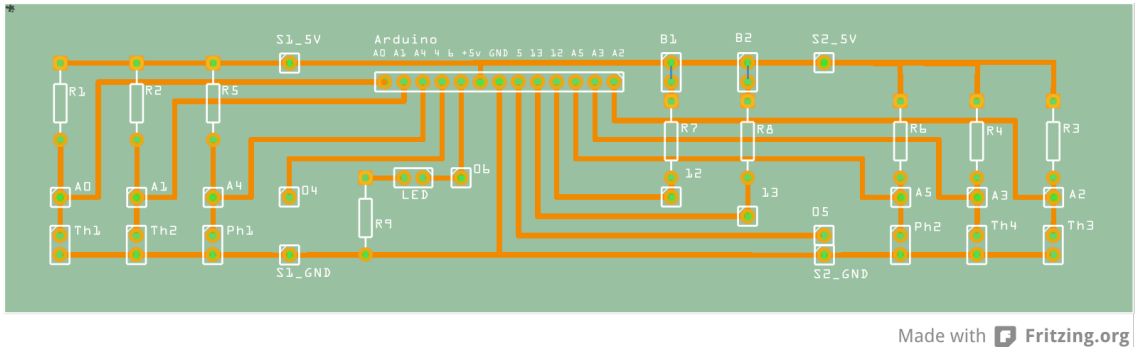
Schéma du circuit électronique

Typon

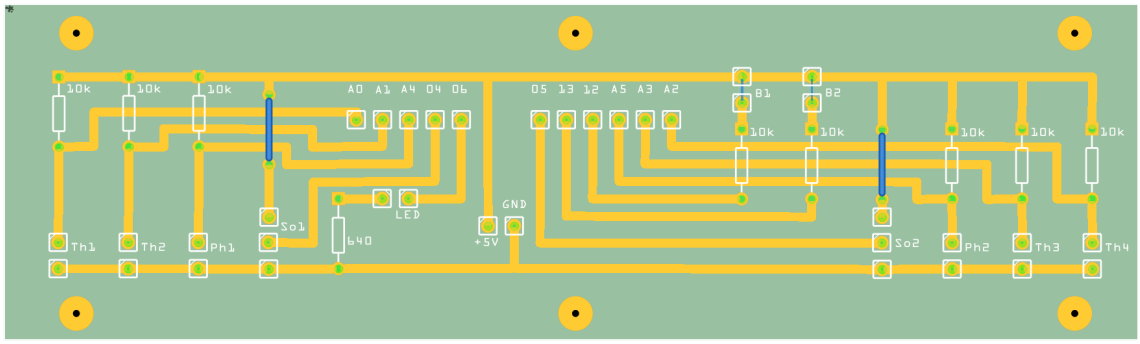
Pour réaliser le circuit imprimé, il a fallu concevoir un typon. Plusieurs versions successives ont été crées, et modifiées selon les conseils de François Lachambre.



Première version complète du typon



Version réduisant l'encombrement



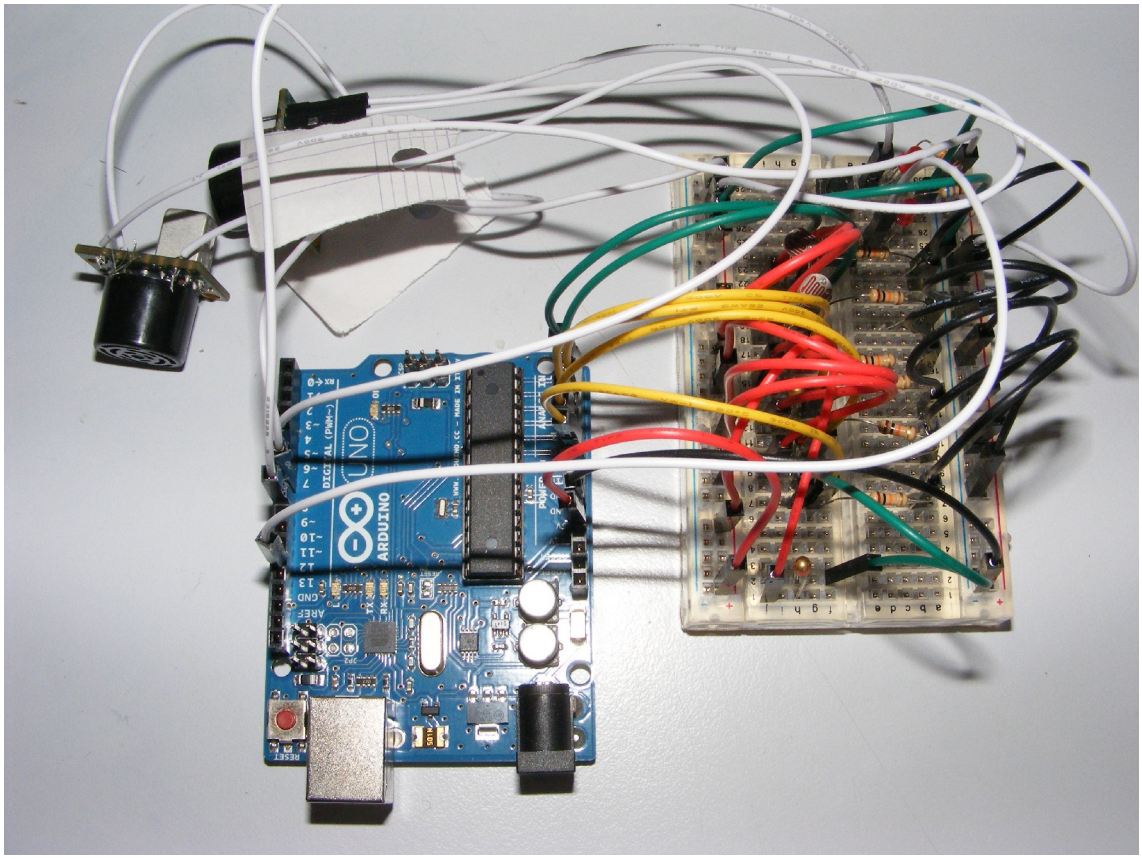
Version finale, avec les cotes exactes

Lors de l'assemblage, des problèmes liés à l'encombrement des connecteurs ont cependant nécessité des ajustements directement sur la carte. Une nouvelle version du typon pourrait y remédier.

Réalisation

Prototypage

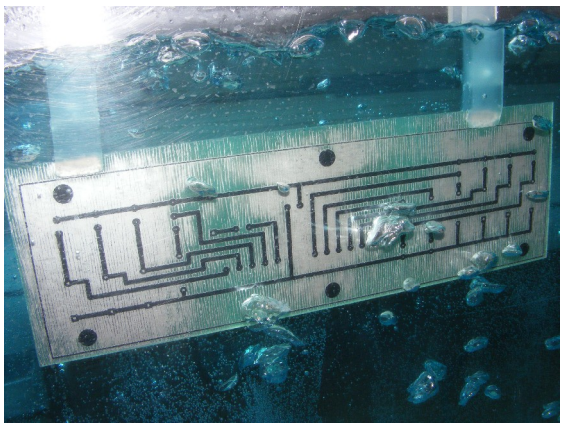
Dans les premiers temps, les tests des capteurs ont été effectués avec un système dit de "prototypage". Les composants n'étaient pas soudés mais simplement fichés dans une plaque dédiée (plaque de prototypage ou breadboard). Ce système, très pratique pour la conception (où le circuit est susceptible de changer), n'est cependant pas très fiable, surtout quand le circuit prend de l'importance (nombreux faux-contacts).



Le prototype du circuit complet

Usinage

La fabrication du circuit imprimé s'est faite au laboratoire d'électronique du Lycée Viette, avec, tout d'abord, la gravure de la carte, puis le perçage.

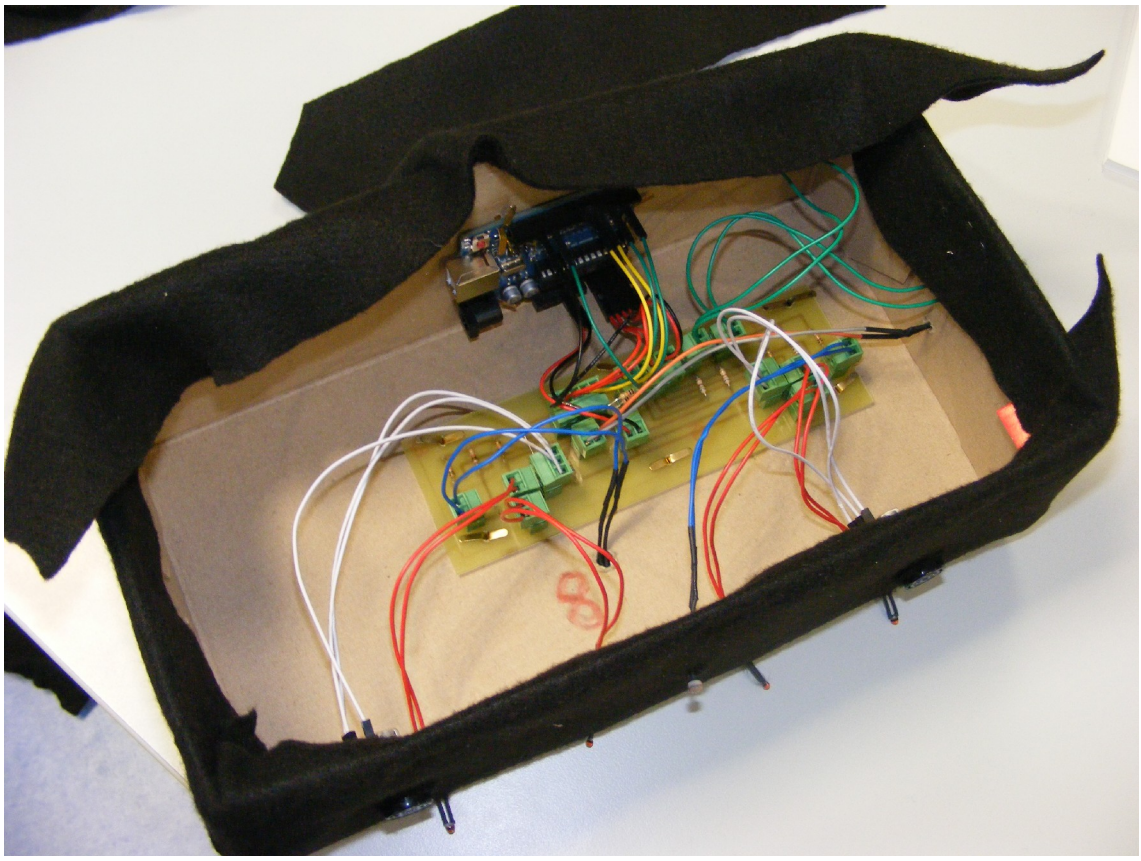


Circuit en cours de gravure dans son bain chimique

Cédric perce, aux côtés de François Lachambre

Assemblage

L'assemblage, commencé au lycée, a été terminé au département multimédia. Le circuit a été fixé dans une boîte à chaussures percée et recouverte de feutrine noire.



La fin de l'assemblage

Logiciel

Arduino

Initialisation

```
/* digital pins */
const int recButton = 12;
const int presetButton = 2;
const int sonar1 = 6;
const int sonar2 = 7;
const int led = 4;
/* analog pins */
const int temp1 = 0;
const int temp2 = 1;
const int temp3 = 2;
const int temp4 = 3;
const int lux1 = 4;
const int lux2 = 5;
long pulse1 = 0;
long pulse2 = 0;
int rec = 0;
int preset = 0;
float maxLux1 = 1024;
float maxLux2 = 1024;
float minTemp1 = 1024;
float minTemp2 = 1024;
float minTemp3 = 1024;
float minTemp4 = 1024;
float maxSonar1 = 1024;
float maxSonar2 = 1024;
float maxSonar1Cm = 42;
float maxSonar2Cm = 42;
```

Cette première partie du code Arduino permet l'initialisation les différentes valeurs telles que les maxima des différents capteurs. Elle définit, de la même façon, sur quelles broches sont connectés quels capteurs.

Setup

```
void setup() {
  Serial.begin(9600);
  pinMode(sonar1, INPUT);
  pinMode(sonar2, INPUT);
  pinMode(recButton, INPUT);
  pinMode(presetButton, INPUT);
  pinMode(led, OUTPUT);
}
```

Cette fonction appelée au lancement du programme nous permet de démarrer la communication en série et de définir le sens de certaines broches (INPUT ou OUTPUT).

Loop

```
void loop() {

  /* ... */
}
```

Cette fonction, comme son nom l'indique, va s'exécuter en boucle sans fin. L'intégralité de son contenu sera donc exécuté à chaque passage de boucle, ce contenu est détaillé dans ce qui suit.

Gestion de la LED

```
/* gestion de la LED */
/* pendant calibration, clignotement */
if(rec > 0){
  if(rec % 3 == 1){
    digitalWrite(led, HIGH);
  }
  else{
    digitalWrite(led, LOW);
  }
}
else{
  /* si le bouton de preset est actif, extinction */
  if(digitalRead(presetButton) == HIGH){
    digitalWrite(led, LOW);
  }
  /* sinon, LED alumee */
  else{
    digitalWrite(led, HIGH);
  }
}
```

Cette première partie de la boucle permet la gestion de la LED en fonction du feedback que l'on veut renvoyer à l'utilisateur. Elle va donc clignoter pendant la calibration, s'éteindre lors d'un changement de preset et simplement rester allumée dans le cas d'un fonctionnement normal du système.

Changement de preset

```
/* si on appuie sur le bouton de preset */
if(digitalRead(presetButton) == HIGH){
    preset = 1;
}else{
    preset = 0;
}
```

Ce petit bout de code permet simplement de passer la variable « preset » à 1 lorsque l'on appuie sur le bouton correspondant et inversement. Cette valeur sera par la suite envoyée par communication en série.

```
Serial.print(preset);
Serial.print("*");
```

Calibration

```
/* si le bouton de calibrage est actif */
if(digitalRead(recButton) == HIGH){
  maxSonar1 = 0;
  maxSonar2 = 0;
  maxLux1 = 1024;
  maxLux2 = 1024;
  minTemp1 = 0;
  minTemp2 = 0;
  minTemp3 = 0;
  minTemp4 = 0;
  rec = 50; /* nombre de pas de calibration */
}
```

```
/* pendant la calibration */
if(rec > 0){
  if(pulse1 > maxSonar1){
    maxSonar1 = pulse1;
  }
  if(pulse2 > maxSonar2){
    maxSonar2 = pulse2;
  }
  if(analogRead(lux1) < maxLux1){
    maxLux1 = analogRead(lux1);
  }
  if(analogRead(lux2) < maxLux2){
    maxLux2= analogRead(lux2);
  }
  if(analogRead(temp1) > minTemp1){
    minTemp1 = analogRead(temp1);
  }
  if(analogRead(temp2) > minTemp2){
    minTemp2 = analogRead(temp2);
  }
  if(analogRead(temp3) > minTemp3){
    minTemp3 = analogRead(temp3);
  }
  if(analogRead(temp4) > minTemp4){
    minTemp4 = analogRead(temp4);
  }
  maxSonar1Cm = (maxSonar1 / 147) * 2.54;
  maxSonar2Cm = (maxSonar2 / 147) * 2.54;
  rec = rec - 1;
}
```

Cette partie permet, via une pression sur un bouton, la calibration des différents capteurs. Pendant 5 secondes (50 itérations de la boucle loop()), le minimum ou le maximum (selon le type de capteur) est enregistré pour chaque capteur, de manière à définir l'état neutre de l'installation. Les variations des capteurs sont ensuite calculées par rapport à cet état neutre. Ainsi, l'installation peut s'adapter à des lieux différents (taille de la pièce, luminosité ambiante, …).

Gestion des capteurs : Sonars

```
/* on lit les entrees des sonars */
pulse1 = pulseIn(sonar1, HIGH);
pulse2 = pulseIn(sonar2, HIGH);

/* envoi des donnees */
pulse1 = (pulse1 / 147) * 2.54;
int send_sonar1 = int((pulse1 / maxSonar1Cm) * 800);
if (send_sonar1 > 800){
  send_sonar1 = 800;
}
else if (send_sonar1 < 0){
  send_sonar1 = 0;
}
Serial.print(send_sonar1);
Serial.print("*");

pulse2 = (pulse2 / 147) * 2.54;
int send_sonar2 = int((pulse2 / maxSonar2Cm) * 800);
if (send_sonar2 > 800){
  send_sonar2 = 800;
}
else if (send_sonar2 < 0){
  send_sonar2 = 0;
}
Serial.print(send_sonar2);
Serial.print("*");
```

Ici, on récupère les valeurs données par les capteurs sonars, qui sont ensuite traitées (conversion, mise à l'échelle) avant d'être envoyées par communication en série. A noter, et ce pour toutes les données envoyées en série, que les valeurs sont séparées par le caractère « * » permettant de pouvoir les traiter dans Pure Data.

Gestion des capteurs : Photorésistances

```
int send_lux1 = int(((1024 - analogRead(lux1)) / (1024 - maxLux1)) * 1024);
if(send_lux1 > 1024){
  send_lux1 = 1024;
}
else if(send_lux1 < 0){
  send_lux1 = 0;
}
Serial.print(send_lux1);
Serial.print("*");

int send_lux2 = int(((1024 - analogRead(lux2)) / (1024 - maxLux2)) * 1024);
if(send_lux2 > 1024){
  send_lux2 = 1024;
}
else if(send_lux2 < 0){
  send_lux2 = 0;
}
Serial.print(send_lux2);
Serial.print("*");
```

De la même façon que pour les capteurs sonars, les valeurs des photorésistances sont « lues », traitées et envoyées par communication série.

Gestion des capteurs : Thermistances

```
int send_temp1 = int(((analogRead(temp1) - (minTemp1 - 100)) / 100) * 1024);
if(send_temp1 > 1024){
  send_temp1 = 1024;
}
else if(send_temp1 < 0){
  send_temp1 = 0;
}
Serial.print(send_temp1);
Serial.print("*");

int send_temp2 = int(((analogRead(temp2) - (minTemp2 - 100)) / 100) * 1024);
if(send_temp2 > 1024){
  send_temp2 = 1024;
}
else if(send_temp2 < 0){
  send_temp2 = 0;
}
Serial.print(send_temp2);
Serial.print("*");

int send_temp3 = int(((analogRead(temp3) - (minTemp3 - 100)) / 100) * 1024);
if(send_temp3 > 1024){
  send_temp3 = 1024;
}
else if(send_temp3 < 0){
  send_temp3 = 0;
}
Serial.print(send_temp3);
Serial.print("*");

int send_temp4 = int(((analogRead(temp4) - (minTemp4 - 100)) / 100) * 1024);
if(send_temp4 > 1024){
  send_temp4 = 1024;
}
else if(send_temp4 < 0){
  send_temp4 = 0;
}
Serial.print(send_temp4);
Serial.print("*");
```

A nouveau la récupération, le traitement et l'envoi en série des valeurs de thermistances sont assurés par cette partie du code.

Précisions sur l'envoi des valeurs

Les différentes valeurs des capteurs, après avoir été lues et traitées sont envoyées en série selon ce modèle :

S1 * S2 * T1 * T2 * T3 * T4 * L1 * L2 * P /

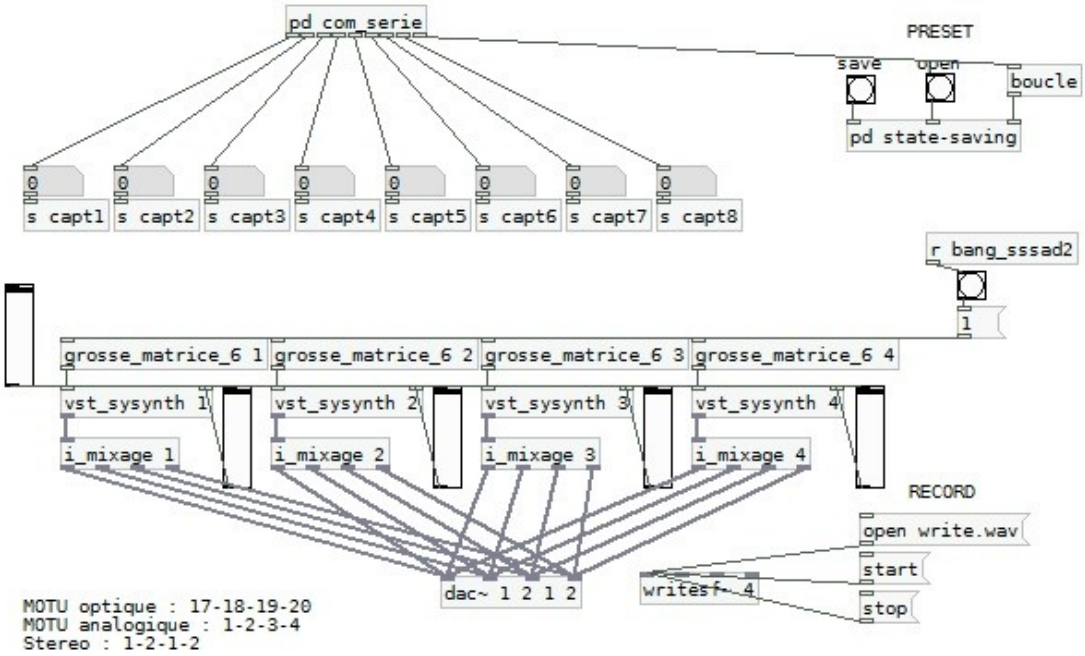
Avec « S » les capteurs sonars, « T » les thermistances, « L » les photorésistances et « P » le booléen de preset.

```
Serial.print("/");
delay(100);
```

Afin de grouper les différentes valeurs et qu'elles puissent être traitées dans Pure Data, un « / » est envoyé avant de laisser un délai de 100ms pour recommencer la boucle.

Pure Data

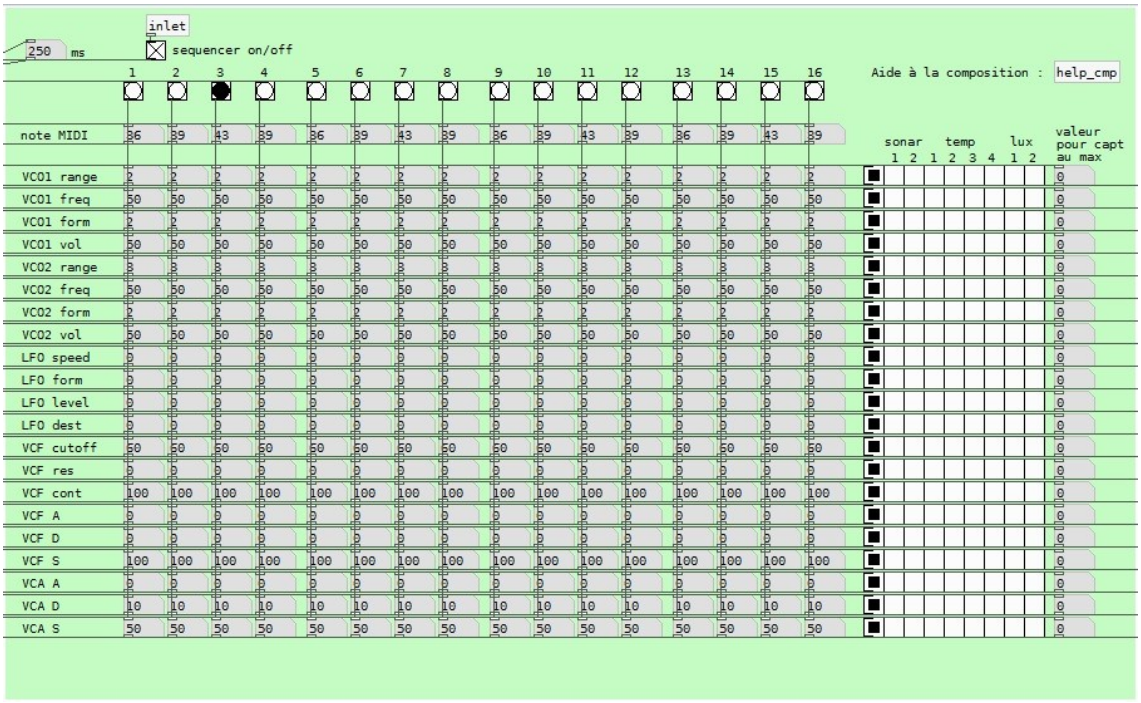
Sy~



Le patch principal du projet donne une vision assez claire de la structure du programme. Un sous-patch recevant les données des capteurs en provenance d'Arduino (com_serie), un autre gérant les presets (state-saving), puis quatre ensembles composition (gros_matrice) / synthèse (vst_synth) / mixage (i_mixage), reliés à la sortie audio quadriphonique (dac~).

Des curseurs permettent de régler le volume global et celui de chaque synthétiseur, mais uniquement à des fins de tests (le mixage artistique est détaillé plus bas). En bas à droite, un système permet également l'enregistrement audio de la performance (sous forme d'un fichier WAV avec quatre canaux).

Composition



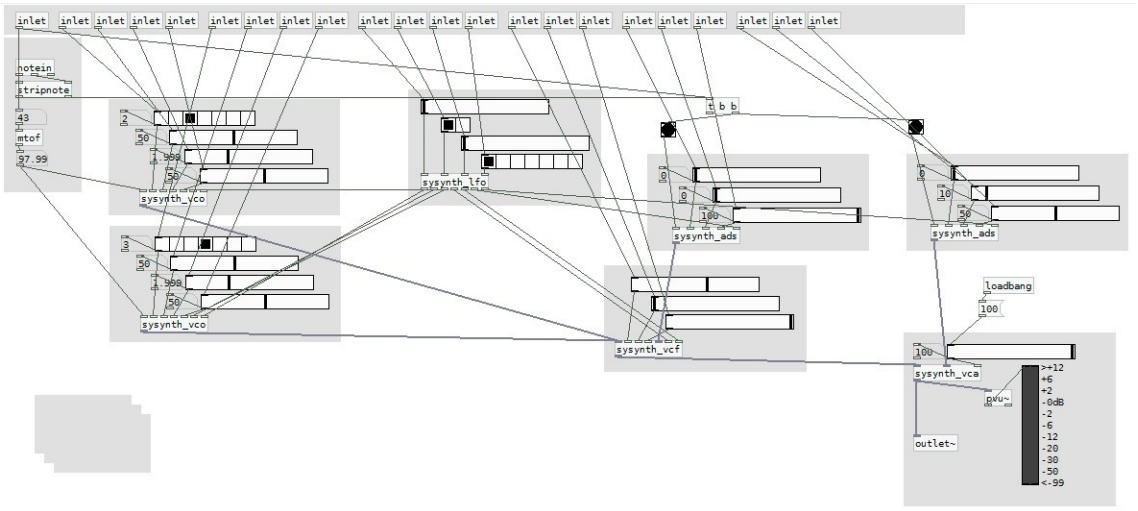
La composition pour Sy~ est abordée à la manière d'un séquenceur (ou d'une boîte à rythme) : de gauche à droite, 16 colonnes, correspondant aux 16 pas de la séquence qui sont déclenchés successivement. On définit la durée d'un pas en haut à gauche.

Pour chaque pas, 22 valeurs doivent être définies : une note (notation MIDI) et 21 paramètres de synthèses. Pour faciliter la saisie de ces nombres, une petite fenêtre d'aide à la composition regroupe des indications précieuses sur les éléments les plus obscurs.

A droite du séquenceur, se trouve la matrice permettant d'associer à chaque paramètre, un capteur qui modulera sa valeur.

La programmation d'une composition complète reste cependant longue et complexe. Des systèmes d'aide à la saisie pourront être mis en place, notamment pour automatiser certaines tâches répétitives (à la manière d'un « copier-coller », par exemple).

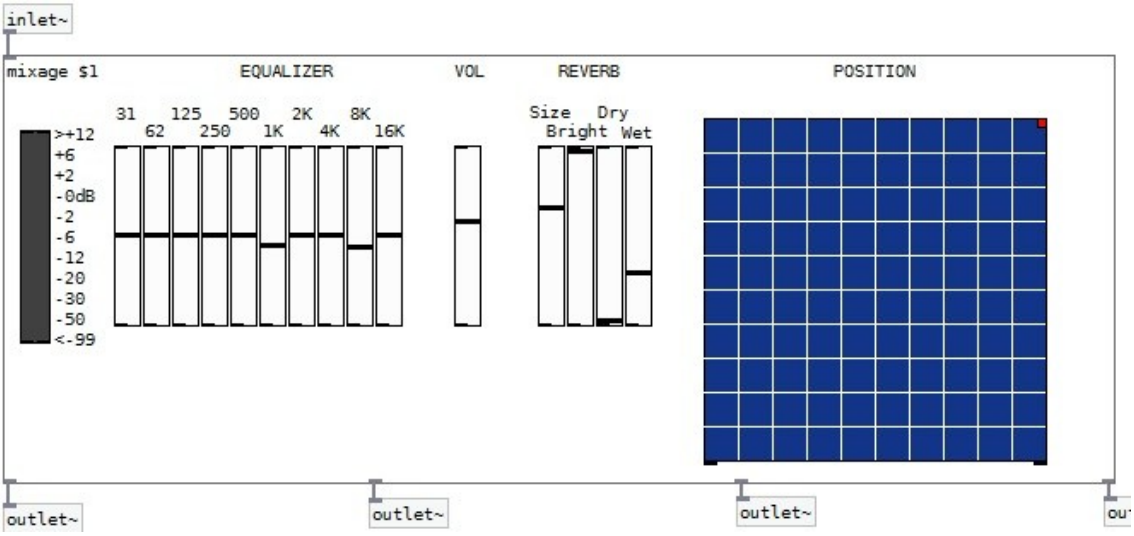
Synthèse



Sysynth est un synthétiseur à modélisation analogique réalisé en Pure Data par Antonin Segault, durant la seconde partie du projet. Ses fonctionnalités émulent celles du Minimoog (ainsi qu'une spécificité du Kobol : le passage continu entre les différentes formes d'ondes).

On y trouve donc deux oscillateurs (avec sept formes d'ondes), un filtre résonnant (presque aussi versatile que son équivalent analogique, lorsqu'il est poussé au maximum), deux générateurs d'enveloppes (pour le filtre et pour l'amplificateur) ainsi qu'une LFO (pouvant moduler les fréquences des oscillateurs, les fréquences du filtre ou encore le relâchement des enveloppes).

Mixage



Pour la seconde partie du projet, Cédric Lanoir s'est, quant à lui, consacré à l'élaboration d'un système de mixage et, sur les conseils de Lorenzo Bianchi, d'une interface de spatialisation. En effet, notre première version, comme le SYHAMO original, associait simplement chaque synthétiseur à une enceinte, ne permettant pas de le positionner finement.

Le patch intègre donc un contrôle de volume avec VU-mètre, un égaliseur 10 bandes, une réverbération et une matrice de position. L'association de l'ensemble de ces paramètres permet de suggérer un espace virtuel complexe autour des quatre synthétiseurs.

Utilisation

Déploiement

- Mise en route de l'ordinateur
  - Allumer l'ordinateur et attendre le chargement complet de celui-ci
  - S'assurer que l'ordinateur est branché sur secteur ou qu'il a suffisamment de batterie
- Branchement de la carte son quadriphonique
  - Brancher le câble USB/Firewire de la carte son quadriphonique dans un port USB/Firewire de l'ordinateur
  - Allumer la carte son quadriphonique
- Branchement du Sy~
  - Brancher le câble USB du boîtier dans un port USB de l'ordinateur
- Lancement du patch Pure Data
  - Double-cliquer sur le fichier "sy_v2.42.pd"
- Vérification des sorties
  - Aller dans l'onglet "Audio settings..." du menu "Media"
  - Choisir la carte son quadriphonique dans la liste après avoir cliqué sur le bouton à droite de "Output device"
  - Entrer le nombre de canaux en fonction des entrées de la carte son quadriphonique (ce nombre doit être supérieur ou égal au plus grand numéro d'entrée de la carte son quadriphonique)
  - Vérifier l'exactitude des sorties dans l'objet "dac~" du patch principal (et les corriger le cas échéant)
- Ouverture de la communication série
  - Cliquer sur le message "device" du sous-patch "com_serie" afin d'avoir une liste des ports disponibles
  - Modifier le numéro du message "open" par le numéro du port voulu
  - Cliquer sur le message "open" afin de démarrer la communication série
- Mise à l'échelle
  - Quitter la zone d'interaction des capteurs
  - Appuyer sur le bouton situé à l'arrière du boîtier
  - Attendre que la LED située sur le côté gauche du boîtier cesse de clignoter

Changement de preset

- Appuyer sur le bouton situé sur le côté gauche du boîtier
- La LED s'éteint brièvement pour indiquer le succès de l'opération

Arrêt du système

- Arrêt de la communication série
  - Modifier le numéro du message "close" du sous-patch "com_serie" par le numéro du port utilisé
  - Cliquer sur le message "close" afin de stopper la communication série
- Débrancher Sy~
  - Débrancher le câble USB (celui sortant du boitier) du port USB de l'ordinateur
- Quitter Pure Data
- Eteindre la carte son quadriphonique et débrancher le câble USB/Firewire du port USB/Firewire de l'ordinateur
- Eteindre l'ordinateur

Conclusion

Ce projet a tout d'abord été l'occasion d'acquérir de nombreuses compétences, autant théoriques (interactivité, composition, acoustique, histoire de la musique, …) que pratiques (électronique, programmation et informatique musicales, ingénierie, …). Plus encore, il nous a donné l'opportunité de gagner en expérience et en maturité dans ces domaines aussi vastes que variés.

L'aboutissement de ce travail est pour nous une grande fierté. Nous nous y sommes investis sérieusement durant six mois, et la récompense est à la hauteur de l'effort fourni. Pour la première fois durant nos études, notre production est concrète, solide et matérielle, ce qui est la source d'une grande satisfaction.

Le projet ne s'arrête pas là, car il nous a ouvert de fantastiques opportunités. Tout d'abord, une exposition pourrait être organisée au conservatoire de Montbéliard, pour présenter notre travail au public, aux élèves et aux professeurs. Plus incroyable, encore, notre participation a été proposée dans le cadre de l'Electronic Music Week (Semaine de la Musique Electronique) à Shanghai (Chine), fin 2013.

De nombreuses améliorations peuvent encore être apportées à notre travail, pour en développer le potentiel artistique, et l'exploiter dans de nouvelles créations. Dans ce but, deux stages ont été proposés par Pays de Montbéliard Agglomération (PMA), à Cédric pour poursuivre le développement de la partie musicale, et à Camille Henriot (un collègue de notre promotion) pour ajouter une dimension visuelle à notre installation.

L'aboutissement de ce Projet de Fin d'Etudes n'est en fait que le début d'une nouvelle et palpitante aventure.

Sources

Bibliographie

SYHAMO, La vie des sons
Janine et Oscar Wiggli
Editions IROISE, 1999
ISBN 2-940091-11-0

Oscar Wiggli ou Le plaisir d'inventer des sons
Kjell Keller, Gilles Gobeil
Editions IROISE, septembre 2000
ISBN 2-940091-14-5

GUAREC - Les sons dans la caverne
Oscar Wiggli
Editions IROISE, 1998
ISBN 2-940091-05-6

Oscar Wiggli – Symboles musicaux
Kjell Keller
Editions IROISE, Octobre 2010
ISBN 9782940091225

Discographie

SYHAMO
Oscar Wiggli
ARGOL Records, Décembre 1999

CELIANDE
Oscar Wiggli
ARGOL Records, Mai 2009

Webographie

Arduino
Site officiel et documentation francophone
http://arduino.cc/fr/

PureData
Site officiel et documentation
http://puredata.info/

PureData sur FlossManuals
Tutoriel francophone
http://fr.flossmanuals.net/puredata/

Fondation Wiggli
http://www.fondationwiggli.ch/

Editions Iroise / ARGOL Records
http://www.iroise.ch/fondation-wiggli/

Sparkfun
Catalogue de composants électroniques
https://www.sparkfun.com/