E-Theremin

Rapport de projet de Quatrième année

Louis CHAUCHARD
Romain IMBERT
Informatique, Microélectronique et Automatique
2014 - 2015

M. Alexandre BOÉ
M. Thomas VANTROYS

Sommaire

Introduction

Ce rapport a pour but d'exposer le travail réalisé dans le cadre du projet de quatrième année d'IMA (semestre 8). Les projets du semestre 8 permettent de travailler pendant plusieurs heures sur un sujet et donc de mettre en pratique les connaissances acquises durant notre cursus ainsi que de développer de nouvelles compétences.

Pour ce projet, nous avons choisi de nous pencher sur le sujet du "E-Theremin". Ce sujet initialement proposé par les enseignants consistait à faire, dans un premier temps, l'étude, mais aussi l'élaboration de notre propre Thérémine puis dans un second temps, utiliser celui-ci à l'aide d'un système embarqué tel un Arduino, pour simuler un dispositif de contrôle (remplacer une souris par exemple).

Afin d'exposer au mieux l'élaboration de ce projet, nous ferons tout d'abord sa présentation. Dans une seconde partie, nous développerons le plus précisément possible son déroulement et le travail accompli. Enfin, nous terminerons par notre ressenti face à ce projet, que ce soit concernant les difficultés rencontrées ou les améliorations possibles.

Présentation du projet

1. Introduction et historique

Le Thérémine (Theremin en anglais) est l'un des premiers instruments de musique électronique, inventé en 1919 par l'ingénieur russe Léon Theremin. C'est un instrument qui possède la particularité de fonctionner sans aucun contact physique de la part du musicien.

En effet, il est composé généralement de deux antennes.

Une antenne verticale (contrôlée par la main droite) qui va commander la hauteur (tonalité) de la note

Une antenne horizontale, utilisée pour faire varier le volume de la note

Le simple fait d'effectuer un mouvement plus ou moins proche des antennes va permettre de générer un son de fréquence et de volume variable.

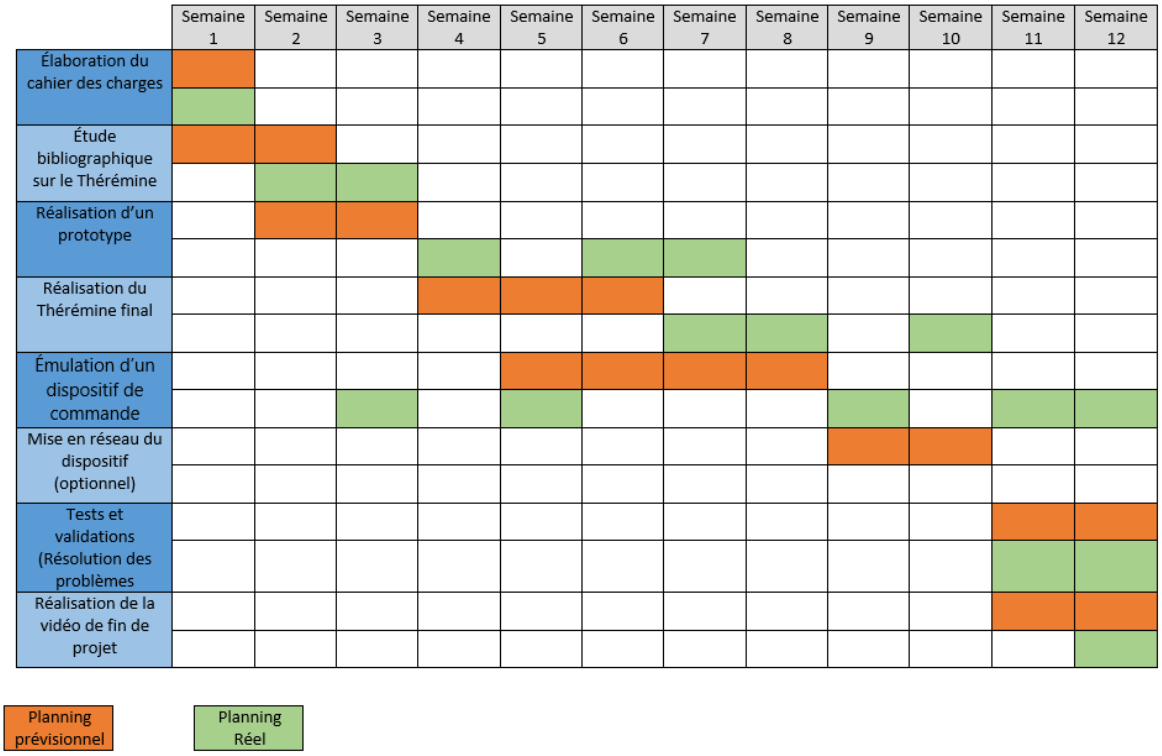
Figure 1 : Léon Theremin

2. Cahier des charges

Le projet était donc composé de deux grandes parties :
Une partie électronique où nous devions réaliser un Thérémine analogique.
Et une partie informatique permettant, à l'aide d'un système embarqué, de traiter les données analogiques (signal sinusoïdal) fournies par le Thérémine final.

Ce projet doit donc permettre de concevoir un instrument très connu et ancien et de le mettre au goût du jour en utilisant son fonctionnement comme une nouvelle possibilité de commande.

3. Déroulement du projet



Au départ, pour réaliser notre projet dans les temps, nous avions élaboré un planning prévisionnel simple et linéaire, structuré en diverses étapes.

Suite à l'élaboration du cahier des charges (en première séance), le but était de concevoir rapidement un prototype (semaine 2 et 3) puis un Thérémine final pour pouvoir ensuite avancer sur la seconde partie (partie informatique) qui nous semblait plus compliquée à réaliser. (Nous avions réservé 4 semaines pour sa réalisation).

Les deux dernières semaines devaient permettre de réaliser les tests finaux, de résoudre les problèmes possibles ainsi que la vidéo de présentation.

Cependant, si nous faisons un bilan de fin de projet, nous pouvons constater que notre planning n'a pas été aussi linéaire que celui initialement prévu.

En effet, nous n'avions pas pris en compte des délais assez importants entre la commande et la réception des composants nécessaires.

De plus, nous avons perdu du temps sur la réalisation du Thérémine final et plus précisément sur la conception du PCB.

Réalisation du projet

1. Partie Électronique

La partie électronique réside dans la réalisation de la première partie du cahier des charges, qui est la conception du Thérémine analogique.

Le montage est constitué de deux oscillateurs. Le premier fournit un signal de fréquence variable. L'antenne permet d'effectuer ces variations. Le second fournit une fréquence fixe.

Ces oscillateurs sont réalisés à l'aide de portes logiques inverseuses (de référence CD4069UBE), résistances et condensateurs.

On peut tout de même ajuster sa fréquence à l'aide d'un potentiomètre.

Ces deux signaux sont donc injectés dans un mélangeur.

En sortie de cette fonction, on obtient deux résultats de fréquences différentes.

Le premier résultat possède une fréquence qui est la somme des deux fréquences des signaux d'entrée et le second la différence.

Ces deux résultats sont placés à l'entrée d'un filtre passe-bas dans le but de supprimer le résultat de la somme des deux fréquences des oscillateurs, car ce signal est inaudible. Le signal gardé (différence des deux signaux) est ensuite amplifié pour être connecté à un haut-parleur de 8 Ohms. L'amplificateur audio utilisé est un LM 386.

La tension nécessaire pour alimenter les différents composants est de 5 Volts. Nous avons donc utilisé une pile 9 Volts que l'on a connectée à un régulateur LM 7805 pour obtenir les 5 Volts. Cet étage permet de rendre le Thérémine "portable".

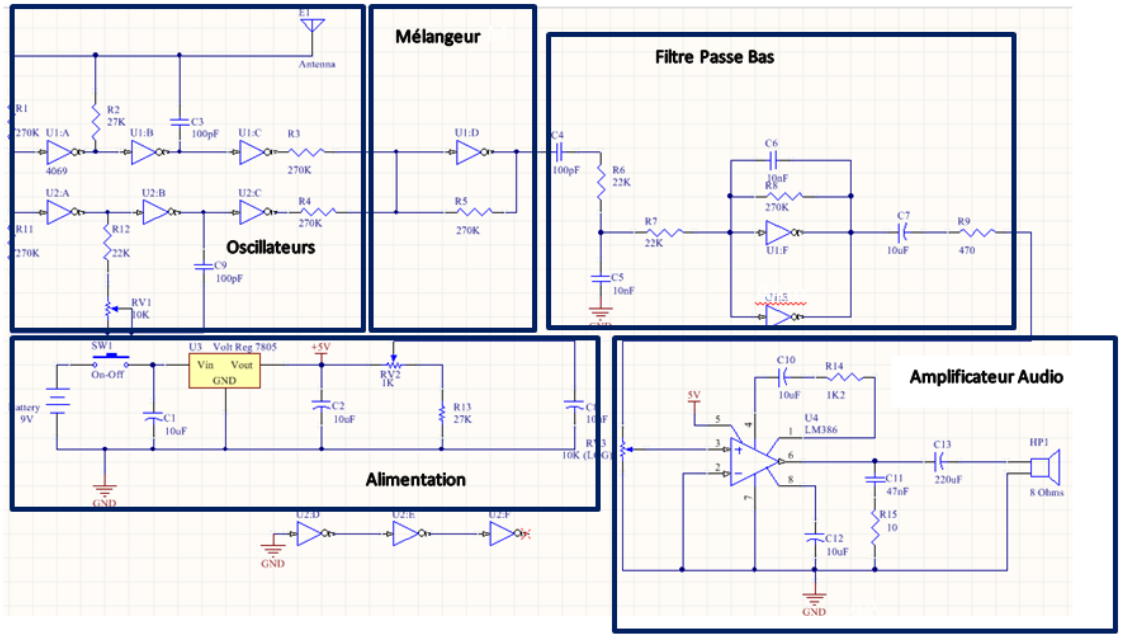


Figure 2 : Schéma final

Le but était d'avoir rapidement un prototype afin de valider le bon fonctionnement du montage, nous avons réalisé un câblage des composants sur plaque d'essai. Ce premier essai nous a permis de visualiser les différents signaux de ce montage.

Le signal obtenu en sortie était très bruité du fait de l'utilisation de nombreux fils et de sa sensibilité aux perturbations extérieures.

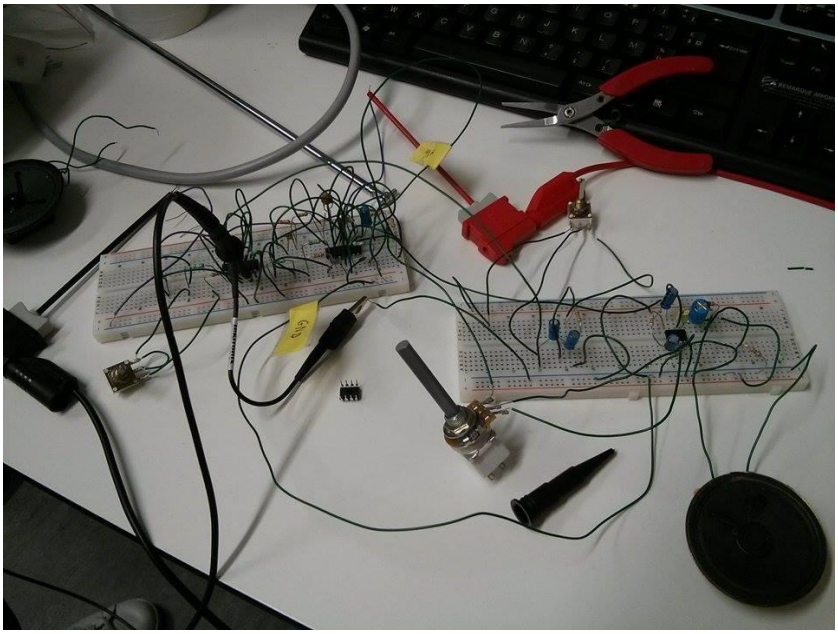


Figure 3 : Prototype sur plaque d'essai

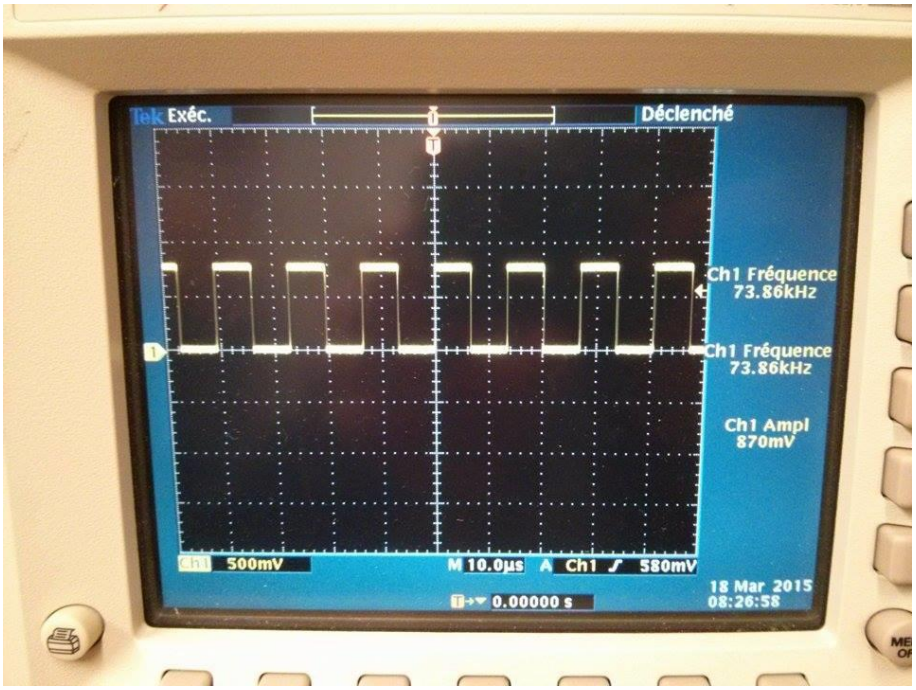


Figure 4 : Signal fourni par l'oscillateur de référence

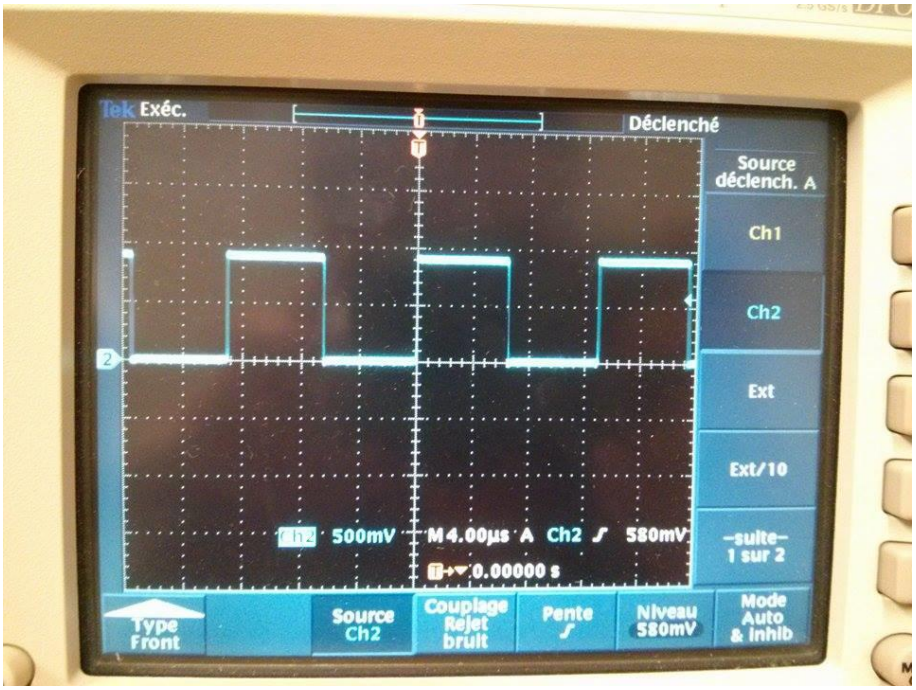


Figure 5 : Signal fourni par le second oscillateur (oscillateur variable)

Le prototype donnant un fonctionnement satisfaisant, nous nous sommes intéressés à la conception du PCB à l'aide du logiciel Altium Designer.

Afin de limiter les perturbations, l'utilisation d'un plan de masse a été choisie.

Voici la disposition des composants sur le PCB[1] :

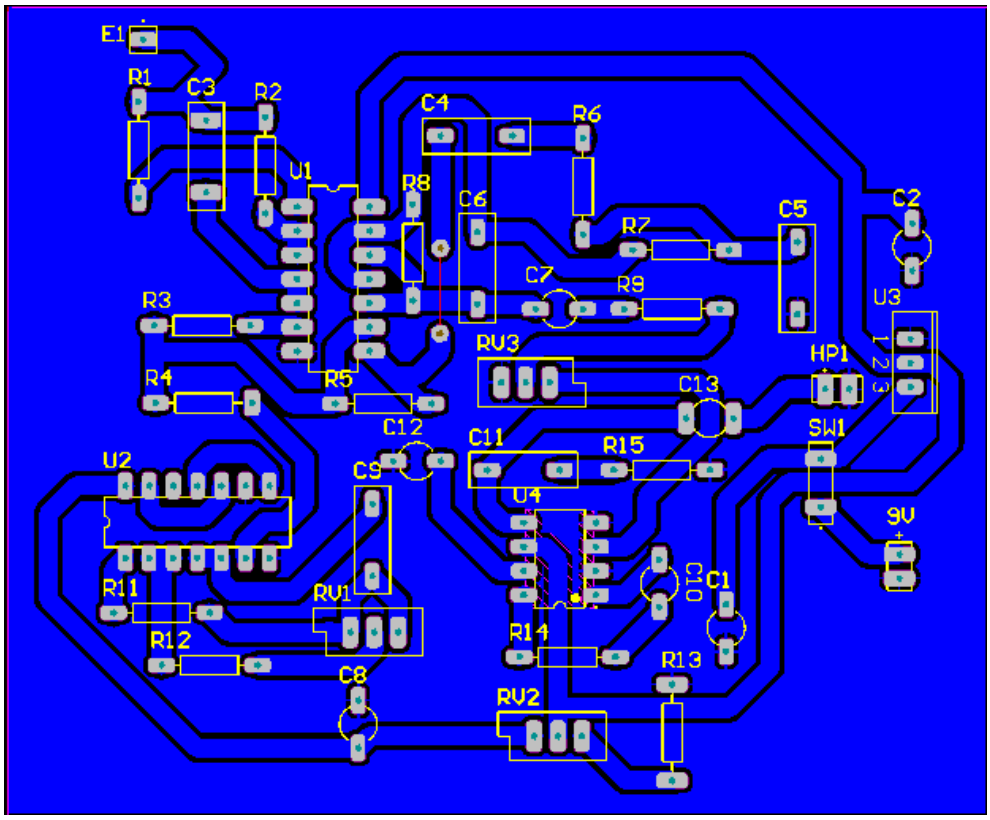


Figure 6 : PCB du Thérémine

Après la réception du PCB, nous avons implanté les différents composants sur celui-ci.

Les différents fils présents permettent la connexion aux différents potentiomètres, interrupteur et antenne.

[1] Voir Annexe 1 pour les brochages de certains composants (Amplificateur, portes inverseuses et régulateur)

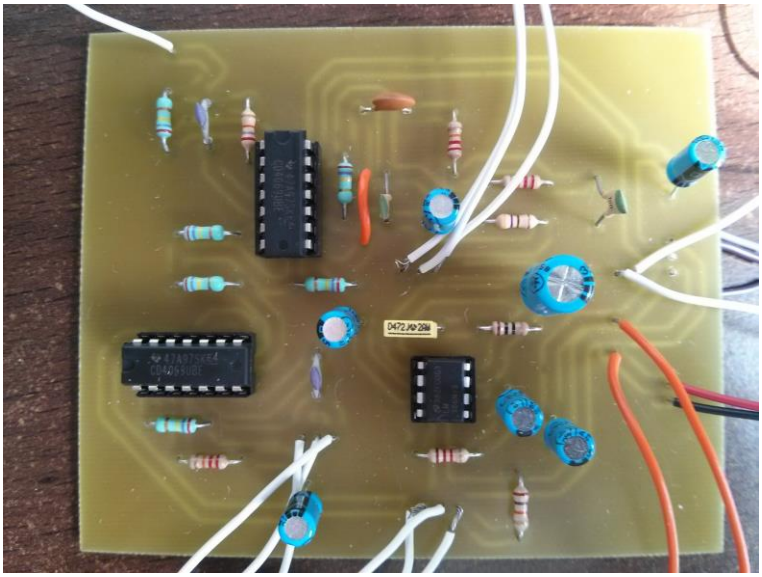


Figure 7 : Carte finale

Nous avons réalisé le boîtier à l'aide de la découpe laser. Elle nous a permis de découper et de graver du texte sur le bois. Pour assembler les différents morceaux, nous les avons collés puis cloués entre eux.

En revanche le couvercle n'a pas été collé afin de permettre le changement de la pile en cas de besoin. Nous avons réalisé des trous pour y accueillir les différents potentiomètres, interrupteur et l'antenne du Thérémine.

Ce boîtier contient aussi l'Arduino, alors deux trous ont été réalisés, un pour la liaison par USB et le second pour son alimentation.

Figure 8 : Thérémine final

2. Partie Informatique

Le signal produit par le Thérémine est une sinusoïde. Pour calculer sa fréquence, Il faut détecter le maximum puis le minimum de ce signal. Ces paramètres vont permettre de déterminer la demi-période de ce signal. Pour calculer le premier maximum, on va utiliser 3 variables : x, y, z. La fonction « ad_sample() » récupère la valeur de l'échantillon.

Celle-ci va être stockée dans la variable x. Avant d'obtenir un nouvel échantillon, la variable y récupère le contenu de la variable x et la variable z le contenu de la variable y. Cette méthode permet de garder les deux échantillons précédents. Ensuite on va comparer leurs 3 valeurs.

Si la variable y est supérieure à x et z cela signifie que l'on a un maximum, on affecte alors 1 à la variable max. À partir de cette détection, on va incrémenter un compteur. On continue de comparer ces 3 variables à chaque nouvel échantillon jusqu' à trouver le minimum (x> y et y<z)..

```
int main(void){
  int temp=0,cmpt=0,freq=0,max=0;
  int x=0,y=0,z=0,j=0;
  char s[30];
  init_printf();
  output_init();
  input_init();
  init_serial(SERIAL_SPEED);
  ad_init(0);
  sei();

while(1){
  z=y;
  y=x;
  x=ad_sample();
  printf("points: %d %d %d\n" ,x,y,z);

  if(max==1){
  cmpt++;
       }
  if((x<y)&&(y>z)){
  cmpt++;
  max=1;
       }

  else {if((x>y) && (y<z) && (max==1)){
  freq=(1/(2*cmpt*0.000001));
  printf("Frequence: %d \n",freq);
  sprintf(s, "%d", freq);

  for(j=0;j<strlen(s);j++)
```

```
  {
        send_serial(s[j]);
  }
  max= 0;
  cmpt=0;
  }
  }
 _delay_us(1);

  }

        return 0;
}
```

La fonction de calcul de fréquence n'étant pas fonctionnelle, nous avons testé l'application Xwindow avec un test basique de l'Arduino.

En effet, dès que l'Arduino détecte une valeur sur son port A[2], il doit émettre un caractère (dans notre cas le caractère "A") par le port série.

Cette action est réalisée par la fonction send_serial.

[2] Voir Annexe 2 : Mapping de l'Atmega2560

```c
#include <avr/io.h>     // for the input/output register
#include <util/delay.h>
#include <string.h>
#include <avr/interrupt.h>
#include "serial.h"

#define SERIAL_SPEED 9600   // For the serial port
#define CPU_FREQ        16000000L       // Assume a CPU frequency of 16Mhz
#define NB_ECH 40

void init_serial(int speed)
{
  /* Set baud rate */
  UBRR0 = CPU_FREQ/(((unsigned long int)speed)<<4)-1;

  /* Enable transmitter & receiver */
  UCSR0B = (1<<TXEN0 | 1<<RXEN0);

  /* Set 8 bits character and 1 stop bit */
  UCSR0C = (1<<UCSZ01 | 1<<UCSZ00);

  /* Set off UART baud doubler
     UCSR0A &= ~(1 << U2X0);*/
}

void send_serial(char c)
{
  loop_until_bit_is_set(UCSR0A, UDRE0);
  UDR0 = c;
}

char get_serial(void) {
  loop_until_bit_is_set(UCSR0A, RXC0);
  return UDR0;
}

```

```
37    // For the AD converter
38    void ad_init(unsigned char channel)
39    {
40      ADCSRA|=(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
41      ADMUX|=(1<<REFS0)|(1<<ADLAR);
42      ADMUX=(ADMUX&0xf0)|channel;
43      ADCSRA|=(1<<ADEN);
44    }
45    // For the I/O
46    void output_init(void){
47      DDRC |= 0xff;
48    }
49
50    void output_set(unsigned char value){
51      if(value==0) PORTB &= 0xfe; else PORTB |= 0x01;
52    }
53
54    void input_init(void){
55      DDRA|=0xFF, //portA en entrée
56      DDRD &= 0xfb;
57      PORTD |= 0x04; // Pull-up activated on PIN 2
58    }
59
60    int main(void){
61      output_init();
62      input_init();
63      init_serial(SERIAL_SPEED); //VITESSE A METTRE ENTRE PARENTHESES
64      ad_init(0);
65      sei();
66      while(1){
67        if(PINA & 0x01 ==0x01){
68          _delay_ms(500);
69          send_serial('A');
70        }
71      }
72      return 0;
73    }
74
```

Un second programme est destiné pour la machine gérant l'application Xwindow.

Le système Xwindow ou X11 est un environnement graphique permettant la gestion des interactions homme - machine (que ce soit par l'écran, le clavier ou la souris) en réseau.

La structure du programme est divisée en trois parties:

- Tout d'abord, l'ouverture d'une connexion à un serveur X
- Puis, les initialisations des différentes fenêtres et actions
- Enfin, dans une boucle infinie, l'attente d'évènements (tels que le clic d'une souris, la pression d'une touche sur le clavier ou autres..). Ces divers évènements vont déclencher les fenêtres et actions initialisées dans la seconde partie.

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <X11/Xlib.h>

GC      gc;
Display *display;
int     temp, screen;
Window  win, root;
unsigned long white_pixel, black_pixel;
int i=0;
char byte;

int main() {
  if ((display = XOpenDisplay ("")) == NULL) {
    fprintf (stderr, "Can't open Display\n");
    exit (1);
  }
  gc = DefaultGC (display, screen);
  screen = DefaultScreen (display);
  root = RootWindow (display, screen);
  white_pixel = WhitePixel (display, screen);
  black_pixel = BlackPixel (display, screen);
  win = XCreateSimpleWindow (display, root,
                  0, 0, 800, 800, 2, black_pixel, white_pixel); //Création d'une fenetre
  XSelectInput (display, win, ExposureMask | ButtonPressMask | KeyPressMask);
  XStoreName (display, win, "E-Theremin");  //Titre de la fenêtre
  XMapWindow (display, win);   //Affiche la fenêtre
  XColor xcolour;
  xcolour.green = 32000;
  xcolour.flags = DoRed;
  XSetForeground(display, gc, xcolour.green);

```

Le “display” définit la connexion de l’application à un serveur X. À ce display est associé une ou plusieurs unités d’affichage “screen”.

Afin de stocker tous les paramètres d’affichages (tels que la couleur de fond, type d’affichage, couleur du dessin..) une structure nommée GC (graphic context) est réalisée.

```c
  while(1){
    int fd = open("/dev/ttyACM0", 'r');
    ssize_t size = read(fd, &byte, 1);
    if(byte=='A'){
      XEvent ev;
      XNextEvent (display, &ev);
      switch (ev.type) {
        case Expose :
            XFillRectangle (display, win, gc, 60,60, 400, 400);
        break;
        case KeyPress :
            XClearWindow (display, win);
            XDrawString (display, win, gc, 30, 30, "TEST2",54);
            XFlush (display);
            sleep(5);
            XDestroyWindow(display, win);   //Destruction de la fenêtre
            XCloseDisplay(display); //Ferme le canal d'affichage
            exit(0);
        break;
      default :
        break;
      }
    }
  }
  return 0;
}

```

La librairie Xlib comporte un nombre important de fonctions permettant diverses actions graphiques. Pour notre application test, nous avons utilisé des fonctions simples telles que :

XFillRectangle qui permet de dessiner un rectangle plein dans la fenêtre selon diverses spécifications (coin gauche (x,y), largeur, hauteur)

XDrawString qui dessine un nombre de caractères d'une chaîne prédéfinie. Ces différentes actions sont réalisées en fonction d'un évènement (XEvent).

Ces évènements sont identifiés grâce à leurs types. Ils peuvent être en lien avec la souris (ButtonPress, ButtonRelease), le clavier (KeyPress, KeyRelease) ou bien avec le premier affichage d'une fenêtre (Expose).

Enfin, des fonctions comme XCloseDisplay, XDestroyWindow permettent de fermer les différents dispositifs utilisés.

Ces deux parties de programme (Envoi Arduino et application X11) nous ont permis de réaliser une première application très basique permettant d'afficher une fenêtre avec un rectangle vert de taille prédéfinie dès que l'Arduino détectait une donnée sur l'une des entrées de son port A.

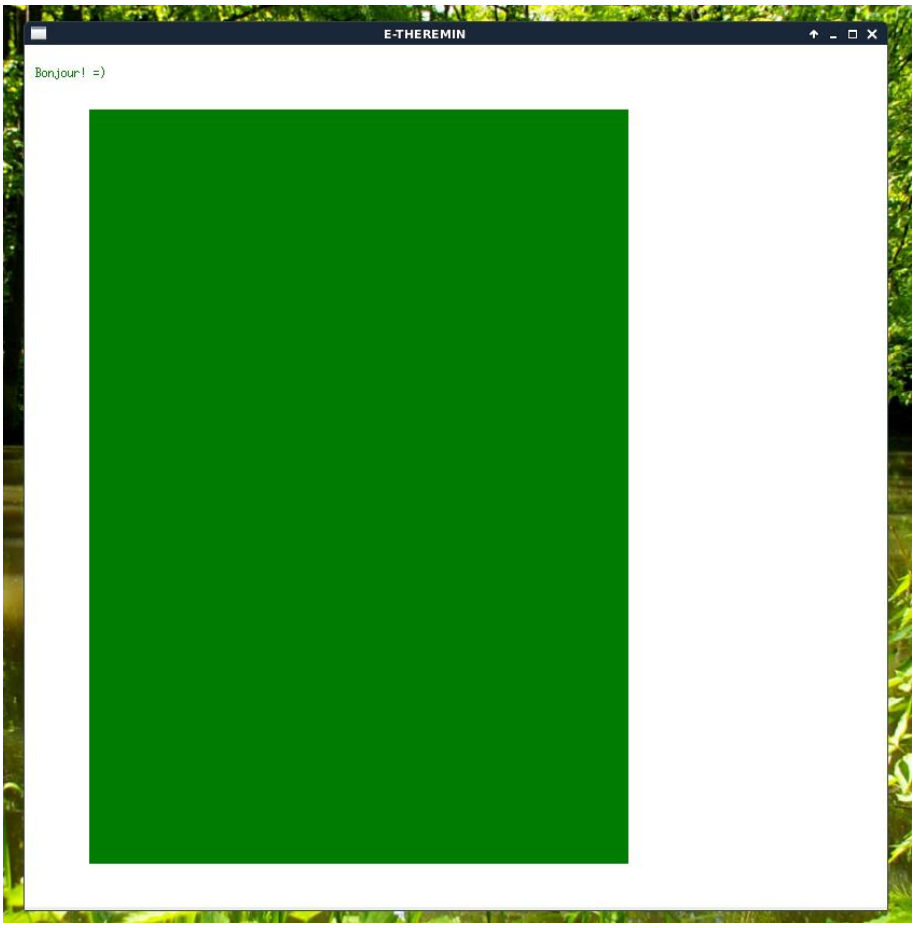


Figure 9 : Application Xwindow

Ressentis vis-à-vis du projet

1. Tests et difficultés rencontrées

Tout d'abord, nous n'avons pas trouvé d'empreinte pour le régulateur LM 7805. Il a donc était nécessaire d'utiliser l'empreinte d'un régulateur similaire. En dépannant le circuit, il est apparu que l'empreinte choisie n'avait pas la même attribution des pins que celle du LM 7805. Des modifications ont donc dû être réalisées directement au niveau des soudures.

Après avoir réalisé ce dépannage, nous obtenons le signal de sortie suivant:

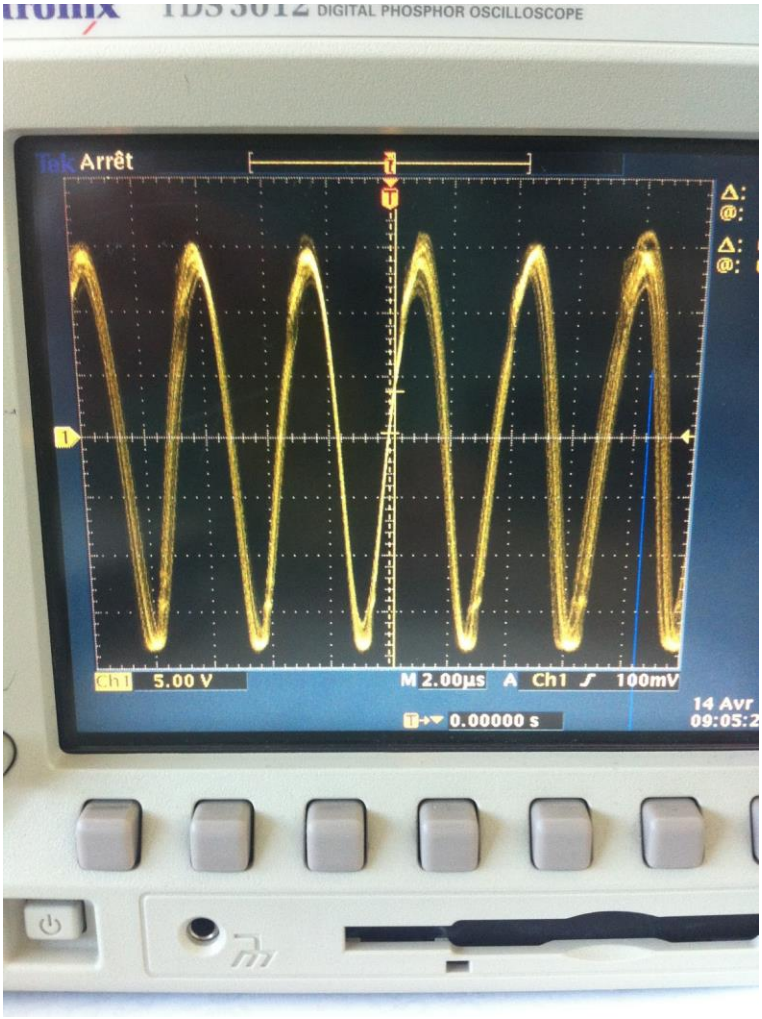


Figure 10 : Signal de sortie avant modifications

Le signal de sortie obtenu étant assez bruité, nous avons utilisé des câbles blindés pour connecter l'antenne à l'oscillateur. Le blindage est connecté au plan de masse.

Toujours pour éviter le maximum de bruit, nous avons également utilisé ce type de câble pour lier le potentiomètre logarithmique à l'amplificateur LM 386.

Suite à ces modifications, nous obtenons un signal de sortie beaucoup moins bruité.

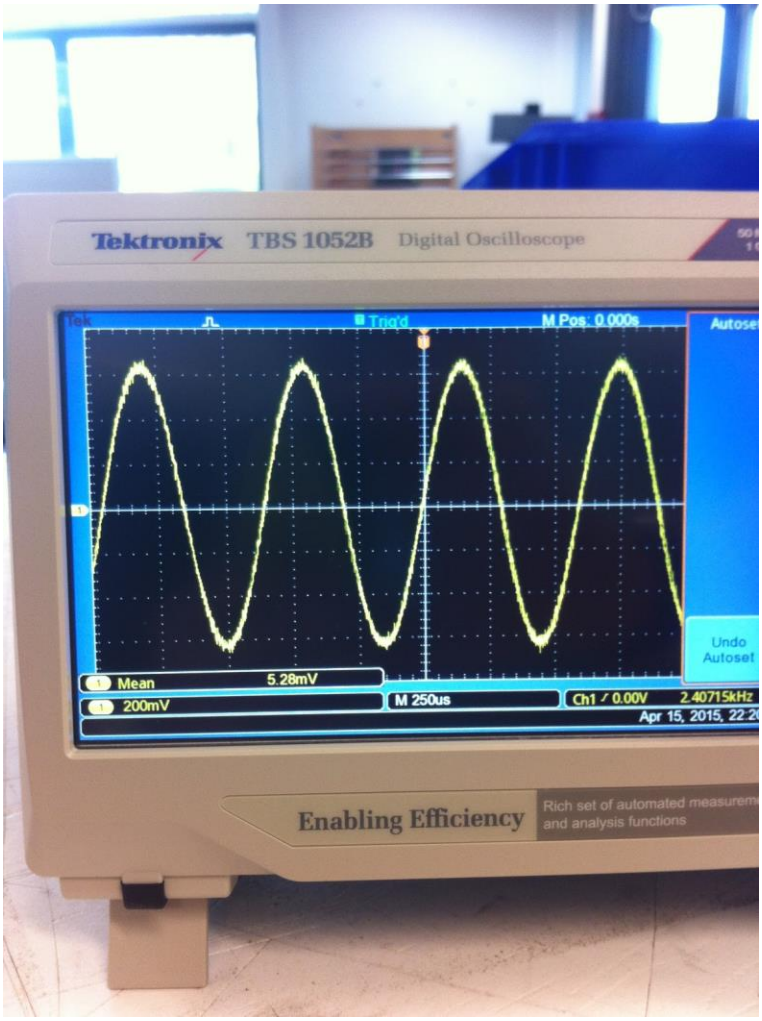


Figure 11 : Signal de sortie après modifications

2. Amélioration et perspectives possibles

Une des améliorations possibles de ce projet une fois abouti serait de mettre en réseau ce dispositif afin de pouvoir commander plusieurs ordinateurs d'une même salle en même temps.

Cette idée donnée par Mr Vantroys était initialement prévue dans notre planning prévisionnel au début du projet (comme partie optionnelle), mais n'a pas été réalisée du fait de notre manque de temps.

La carte de notre Thérémine étant fonctionnelle, il pourrait être intéressant de réaliser cette fois-ci, un nouveau boîtier (en plexiglass par exemple) dans le but d'améliorer l'esthétique de notre Thérémine.

Enfin, le passage vers un Thérémine possédant deux antennes pourrait être un avantage non négligeable concernant l'émulation de la souris.

En effet la possibilité d'avoir deux signaux (hauteur et volume de la note) variables pourrait servir à représenter le déplacement de la souris sur des axes X et Y.

Conclusion

En conclusion, nous avons répondu en partie à la problématique qui nous a été posée en réalisant un Thérémine avec un fonctionnement plutôt satisfaisant. Nous sommes cependant déçus, de ne pas avoir réussi à fournir une solution concernant la seconde partie du projet qui était l'émulation de la souris.

En effet, cette partie n'a malheureusement pas été aboutie. Seules certaines fonctions sont opérationnelles, mais le fonctionnement global du dispositif et plus précisément la communication entre le Thérémine et l'application Xwindow n'est pas achevée.

Néanmoins, la façon dont nous avons réalisé le projet doit pouvoir permettre de le continuer et de l'améliorer assez aisément.

D'un point de vue "humain", ce projet nous a permis de travailler en binôme, avec les difficultés qui en découlent (être en accord avec l'autre, recherche d'efficacité...).

Il nous a aussi mis en situation d'adaptation face aux aléas des temps d'attente de livraisons ou autre.

Enfin, ce projet nous a amenés à réaliser à la fois un cahier des charges, mais également, un planning prévisionnel. Ce planning prévisionnel doit être le plus détaillé possible dès le début du projet. En effet, nous aurions dû prévoir des jours supplémentaires pour certaines réalisations (Boîtier, PCB, partie informatique) et détailler au maximum les différentes étapes de réalisation du Thérémine.

Bibliographie

- Documentations sur le Thérémine

http://www.etheremin.com/technique/

http://www.sonelec-musique.com/electronique_realisations_theremin_presentation.html

- Datasheet des composants

http://pdf.datasheetcatalog.com/datasheet/texasinstruments2/lm386.pdf

http://pdf.datasheetcatalog.com/datasheets/150/206783_DS.pdf

https://www.sparkfun.com/datasheets/Components/LM7805.pdf

- Documentation de l'Arduino Mega2560

http://www.atmel.com/images/doc2549.pdf

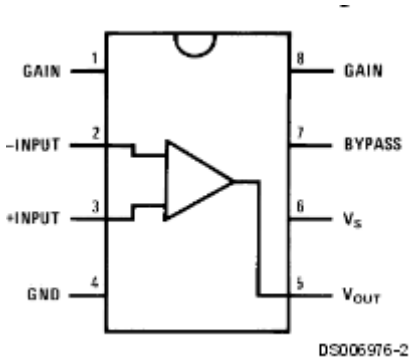
- Documentation sur la librairie « Xlib » utile à la réalisation d'application X11

http://www.x.org/releases/X11R7.7/doc/libX11/libX11/libX11.html#Display_Functions

Annexes

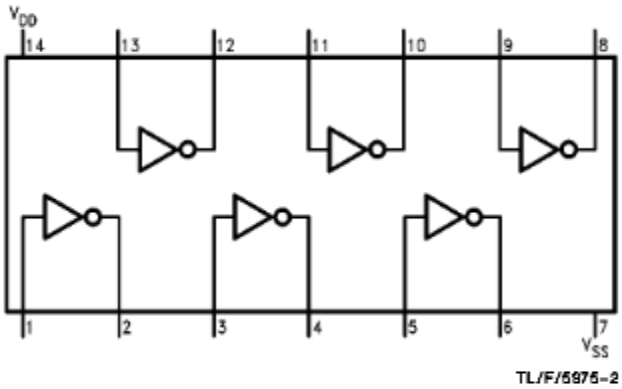
Annexe 1 : Brochage des composants

Brochage de l'amplificateur audio LM386:



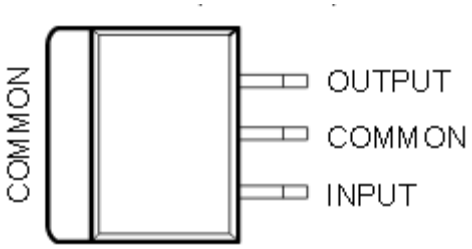
Brochage du boîtier contenant les portes inverseuses CD4069UB:

Dual-In-Line Package



Order Number CD4069UB

Brochage du régulateur LM 7805:



Annexe 2 : Mapping de l'ATMEGA 2560