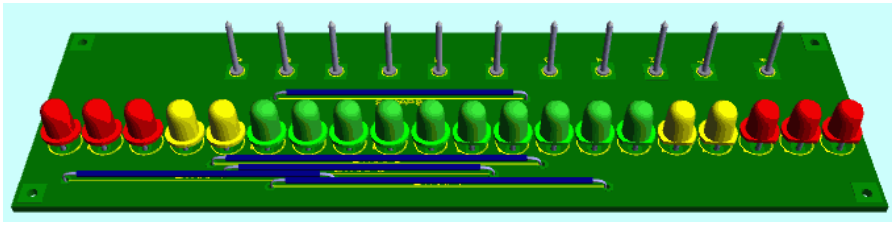
Université François Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle



Sonomètre avec affichage en bargraphe

Projet d'étude et réalisation

Théo Tournadre
Benoît Bento
Promotion 2012-2014

Enseignants
Thierry Lequeu
Philippe Auger

Université François Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

Comment capter un son pour ensuite afficher son volume sur des rangées de LED ?

Théo Tournadre
Benoît Bento
2013-2014

Enseignants
Thierry Lequeu
Philippe Auger

Introduction

Dans le cadre de l'étude et réalisation, nous avons pour objectif de réaliser un projet sur un sujet de notre choix concernant notre formation, nous avons choisi un projet se situant dans le domaine de l'électronique et de la programmation informatique. Notre choix s'est porté sur la réalisation d'un système permettant l'affichage d'un niveau sonore sur des LED.

Ainsi nous allons devoir mettre en place,dans un premier temps, une carte électronique qui va permettre de capter le son et ensuite de l'amplifier à différents niveaux de tensions. Sur une seconde carte électronique, il s'agira de programmer un composant programmable pour permettre ensuite l'affichage sur des LED. Ce projet permet la validation du cours d'étude et réalisation durant le semestre 4.

Index

Index des illustrations

I – Présentation

1 - Cahier des charges

Le but est de réaliser un sonomètre qui mesure l'intensité du son en stéréo puis l'affiche via deux rangées de 12 diodes électroluminescentes. Nous avons eu une période allant du 12 septembre 2013 jusqu'au 25 octobre 2013 pour réaliser notre projet, à raison de deux séances de TP par semaine.

Ce projet s'inscrit dans le cadre du cours d'étude et réalisation. Comme au semestre 3, nous avions la liberté du choix du projet à réaliser.

La fonction principale 1 du système sera d'amplifier les niveaux de tensions images des sons captés par le micro. Nous utiliserons ensuite un détecteur de crête qui permettra de mesurer les valeurs crête de notre tension (**Première carte).**

La fonction principale 2 sera d'afficher les variations de tension sur les rangées de LED (**Deuxième carte).**

Pour réaliser ce projet nous avons eu une limite de budget qui devait atteindre une somme inférieure à 100€, pour l'ensemble du projet.

Le tableau ci-dessous regroupe l'ensemble des composants achetés qui ont été nécessaires au projet, ainsi que les composants qui ont pu être trouvé au magasin de l'IUT :

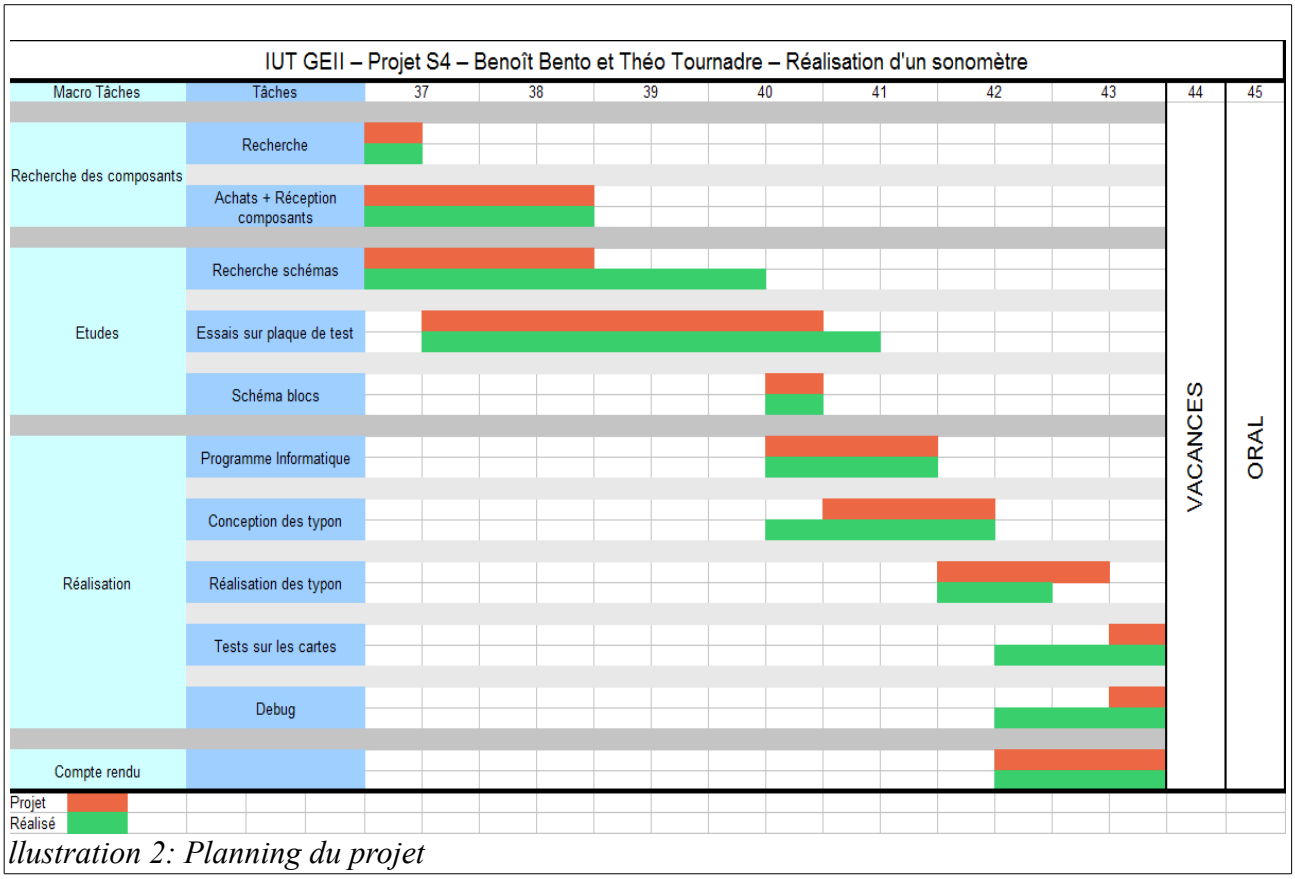
	Référence	Désignation	Quantité	Prix unitaire (en €)	Prix total (en €)
Cartes amplification		Internet			
	724-3134	Micro	2	0,67	1,34
		Magasin			
	LM324N	Amplificateur opérationnel	1	0,4	0,4
		Bornier à vis (2 voies)	2	0,72	1,44
	C	Condensateur (1µF, 470µF)	6	0,5	3
	POT R	Potentiomètre 10kohm	2	0,92	1,84
	R	Résistance (100, 820, 1k, 10k, 100)ohm	32	0,01	0,32
				Prix total cartes (en €)	7
Carte ATMEGA		Magasin			
	8535-16 – ATMEGA	Composant programmable (40 pattes)	1	3	3
		Bornier à vis (2 voies)	3	0,72	2,7
		Inductance (10µH, 47µH)	2	0,32	0,64
	LED 3mm – 2mA	Diode électroluminescente	24	0,08	1,92
	C	Condensateur (22pF, 100nF, 10µF, 100µF, 470µF)	8	0,5	4
	R	Résistance	48	0,01	0,48
				Prix total carte (en €)	12,74
				Prix total des cartes (en €)	19,74

Illustration 1: Liste des différents composants

Nous avons donc acheté les microphones sur un site spécialisé sur internet. Les composants « classiques » tels que des résistances ou des condensateurs ont pu facilement être trouvé au magasin de l'IUT.

2 - Déroulement du projet

Pour commencer ce projet, nous avons tout d'abord élaboré un planning prévisionnel des tâches à faire. Pour cela nous avons penser à toutes les choses que nous avions à réaliser et le temps que cela pouvait nous prendre. Nous en avons donc tiré le planning suivant.



llustration 2: Planning du projet

Le projet étant fini, nous avons ajouté à ce planning prévisionnel, le temps réel que toutes les tâches nous ont pris.

Nous pouvons remarquer que le planning a été dans un premier temps très bien respecté. Par la suite, nous avons pris du retard sur les phases de tests, voulant obtenir un sytème fonctionnant le mieux possible. De plus, lors de la réalisation du typon sur ordinateur, nous avons recontré des problèmes avec les différentes librairies de composants à obtenir pour effectuer notre carte.

Nous avons ensuite réalisé un deuxième planning, permettant de nous répartir les différentes tâches à réaliser au sein du groupe.

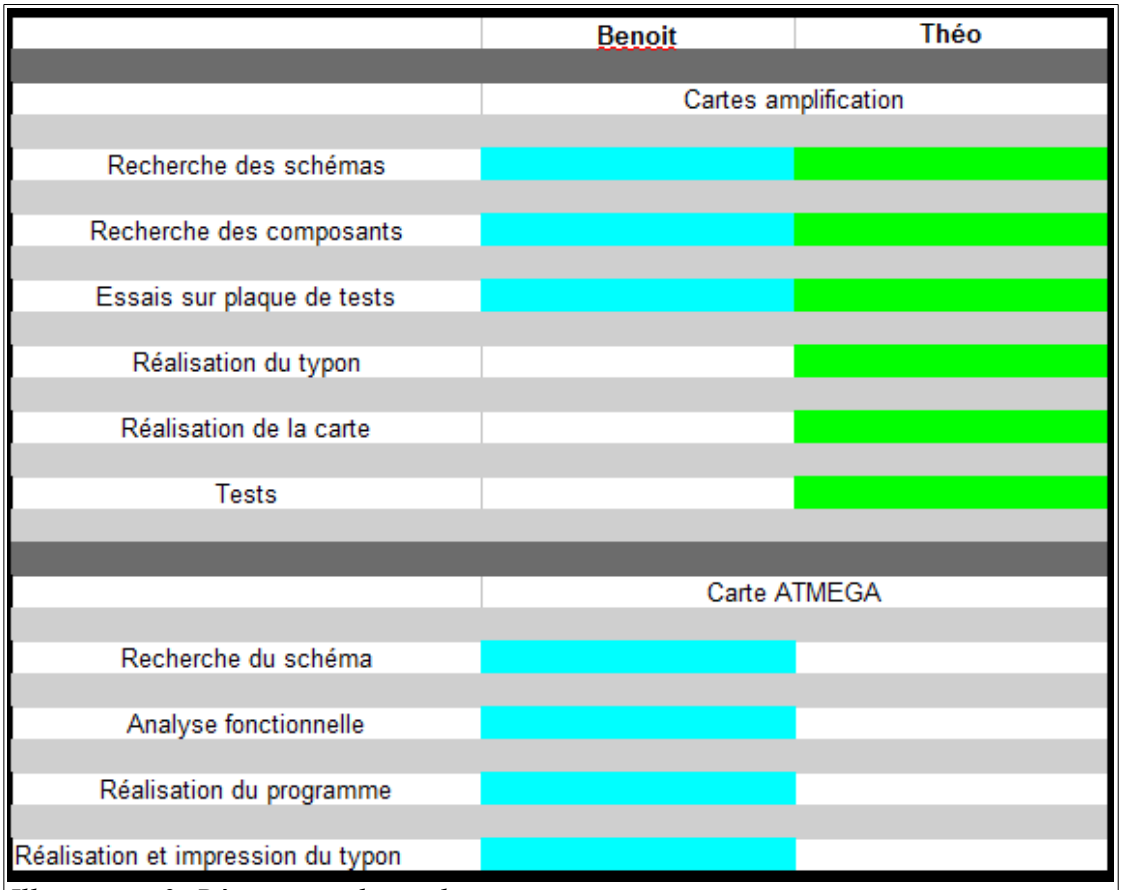


Illustration 3: Répartition des tâches

Cette répartition des tâches a permis une bonne organisation au sein du groupe et ainsi de mener plus facilement à bien notre projet.

II - Première carte

La première carte de notre montage se décompose en quatre parties. La première qui est la réception du son, la deuxième qui est l'amplification de la tension aux bornes du micro, la troisième qui est la détection de crêtes. Et enfin, le montage soustracteur permettant d'obtenir la tension voulu.

1 - Réception du son

Nous avons utilisé un micro condensateur pour capter le son, aux bornes du quel nous récuperons une tension très faible .

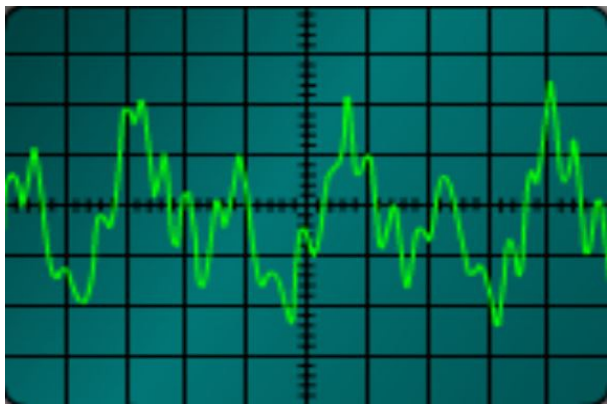


Illustration 4: Visualisation de la tension aux bornes d'un microphone

Cette tension variant de +/- 200mV, Il s'agira donc ensuite de l'amplifier pour permettre son utilisation.

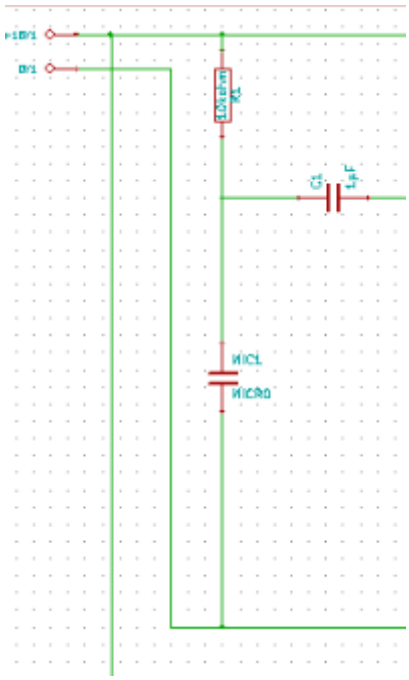
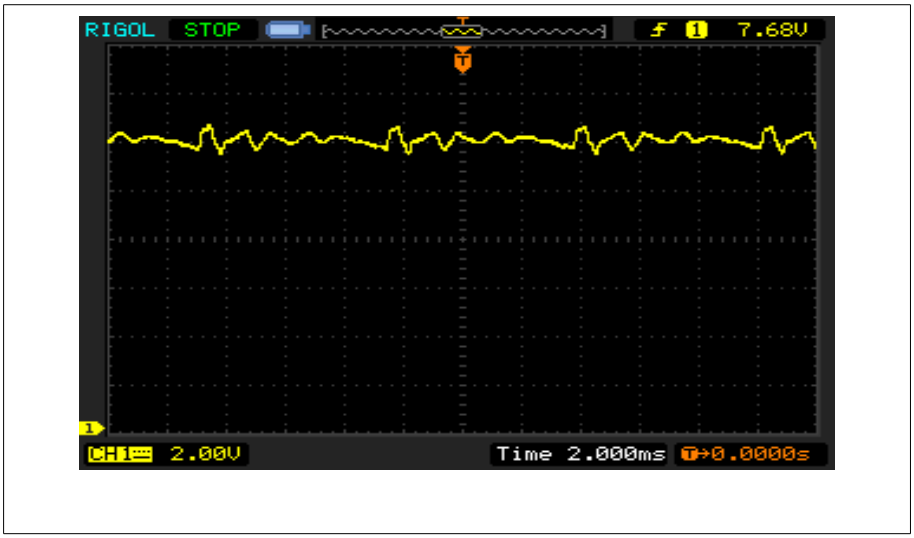


Illustration 5: Schéma Kicad : Microphone

Ceci est le montage permettant de faire fonctionner le microphone. Il est composé du microphone, d'une résistance et d'un condensateur.



Voici la tension que l'on peut visualiser lorsqu'on relève au niveau du nœud reliant les trois composants.

Le condensateur C1 va permettre par la suite de supprimer la composante continue présente sur notre signal. On aura donc par la suite uniquement la tension du micro, variant autour de 0.

2 - Choix de l'AOP

Afin de réaliser les montages comparateur, amplificateur et soustracteur de tension, il a été nécessaire d'utiliser comme composant un AOP. Notre choix s'est orienté vers le LM324N, en effet il est alimenté en asymétriques, c'est à dire en +15V/0V. Ce type d'alimentation est plus facile à réaliser que les alimentations symétriques (exemple : +15V/-15V).

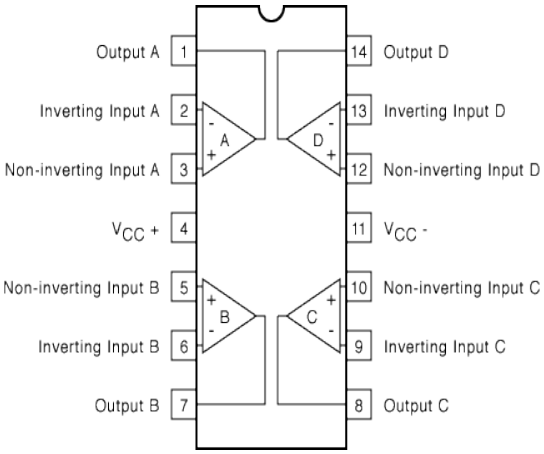


Illustration 6: AOP LM324N : assignation des différentes pattes

De plus, étant que l'on avait besoin de trois montages à AOP dans notre système, le LM324N c'est donc avéré le plus pratique avec ses quatre AOP intégrés.

3 - Premier amplificateur différentiel de tension

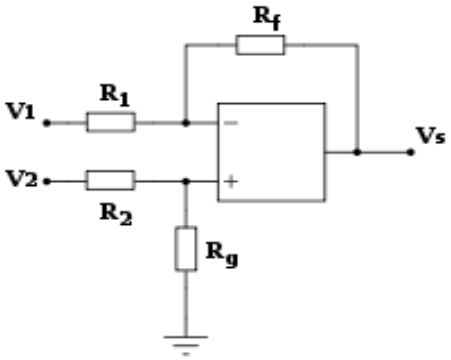


Illustration 7: Schéma AOP : amplificateur différentiel

Afin d'amplifier le signal du micro, nous avons pris un AOP, utilisé en tant qu'amplificateur différentiel.Ce montage répond aux équations suivantes :

$$R_1 = R_f \text{ et } R_2 = R_g, \qquad V_s = V_2 - V_1$$

Dans notre cas, les résistances présentes dans le montage sont toutes égales. On obtiendra donc comme tension de sortie, la différence entre la tension envoyée sur la borne + et la tension envoyée sur la borne - .

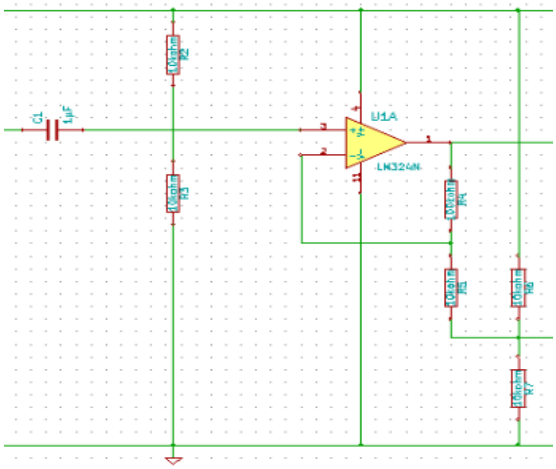


Illustration 8: Schéma Kicad : AOP utilisé en amplificateur différentiel

L'AOP est ici utilisé en tant qu'amplificateur différentiel, il va donc nous permettre comme son nom l'indique d'amplifier la différence entre les tensions qui arrivent à ses bornes + et -. La tension du micro étant de faible valeur, il est difficile pour l'AOP de l'amplifier. Nous avons donc mis en place sur les bornes + et – un diviseur de tension. Il va nous permettre d'obtenir d'un côté pour la borne – une tension de 7,5V continu et l'autre, une tension de 7,5V continu ajouté au signal du micro. L'AOP qui amplifie la différence entre les deux bornes va donc soustraire les tensions de 7,5V et n'amplifier que le signal du micro.

4 - Deuxième amplificateur différentiel de tension

La première amplification n'ayant pas été suffisante, il a donc fallu amplifier une deuxième fois le signal du micro.

Nous sommes cette fois ci dans le cas suivant :

$$R_1 = R_2 \text{ et } R_f = R_g, \qquad V_s = \frac{R_f}{R_1}(V_2 - V_1)$$

Il y a cette fois ci un rapport d'amplification en plus du fait que l'on fasse la différence entre les deux entrées. Il faut donc choisir convenablement les résistances pour obtenir le rapport d'amplification voulu.

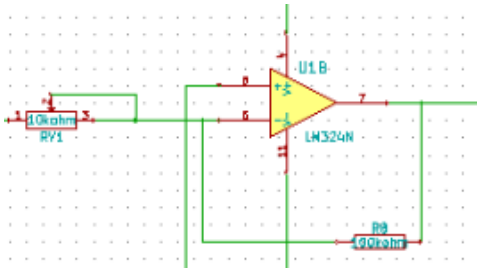


Illustration 9: Schéma Kicad : AOP utilisé en montage amplificateur différentiel

Pour permettre de régler facilement le rapport d'amplification, nous avons décidé de remplacer la résistance R1 par un potentiomètre. Nous pouvons ainsi modifier le rapport d'amplification à notre guise, ce changement facilite donc l'utilisation du montage.

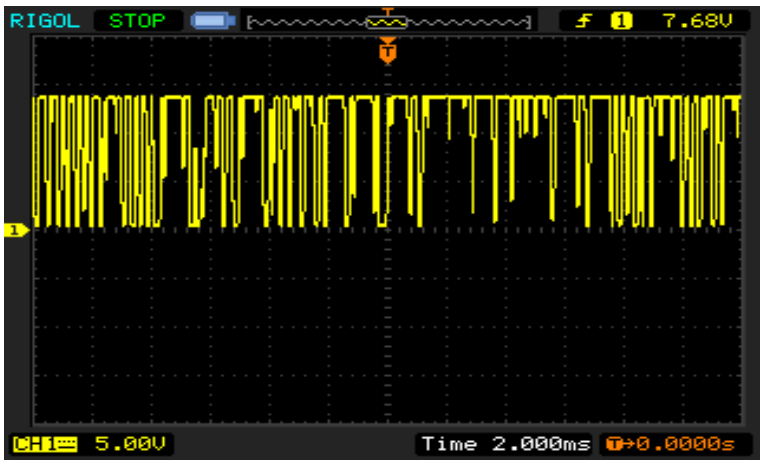


Illustration 10: Visualisation à la sortie de l'amplificateur : tension saturée

Cet oscillogramme présente la tension à la sortie de l'amplificateur. Nous sommes ici dans le cas où l'amplification est maximum et donc la tension de sortie se retrouve saturée, ne pouvant

dépasser les 15V d'alimentation de l'AOP.

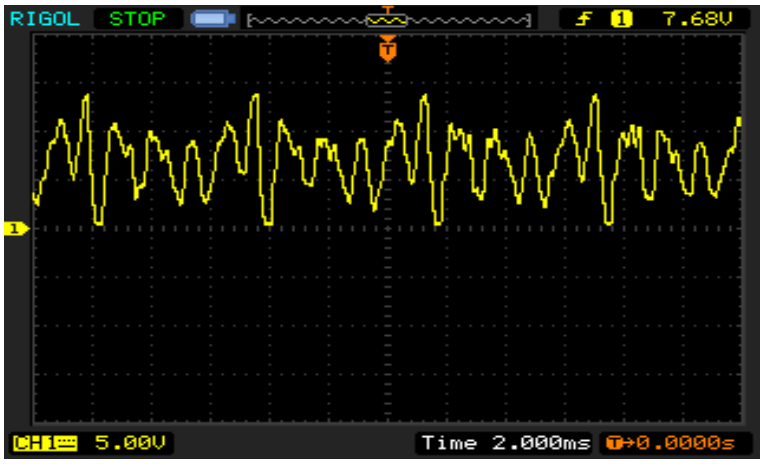


Illustration 11: Visualisation à la sortie de l'amplificateur : tension amplifiée

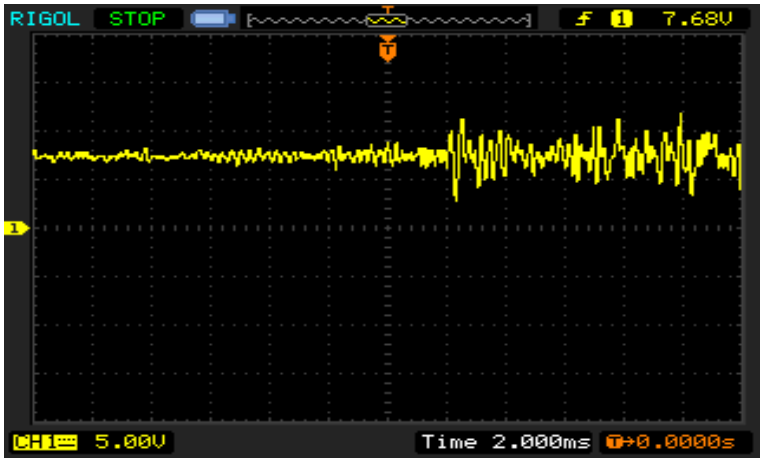


Illustration 12: Visualisation à la sortie de l'amplificateur : tension faiblement amplifiée

Les relevés d'oscillogramme correspondent à la tension amplifiée selon que la valeur du potentiomètre soit plus ou moins grande. Il est donc très facile d'obtenir l'amplification souhaitée.

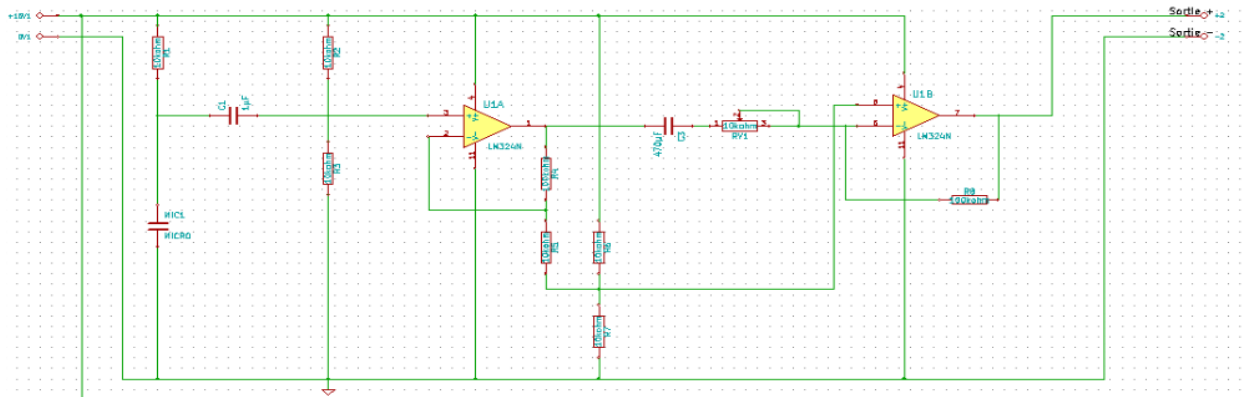


Illustration 13: Schéma Kicad : Micro suivi des deux amplificateurs différentiels

Voici le schéma du 'premièr étage' de notre montage, il est donc composé du microphone qui permet la réception du signal sonore ainsi que de deux amplificateurs différentiels.

5 - Détecteur de crêtes

Afin de répondre au cahier des charges, il était nécessaire pour notre montage d'afficher non pas l'ensemble des variations de la tension mais les valeurs maximum, dites valeurs crêtes, quelle peut atteindre. Nous avons donc pour cela utilisé un montage appelé détecteur de crête.

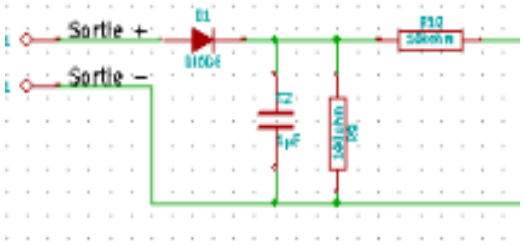


Illustration 14: Schéma Kicad : détecteur de crête

Un détecteur de crête est un montage permettant de mémoriser la tension la plus élevée parmi celles mesurées en entrée (c'est à dire la tension Sortie +). La partie basse du signal va donc être supprimée.

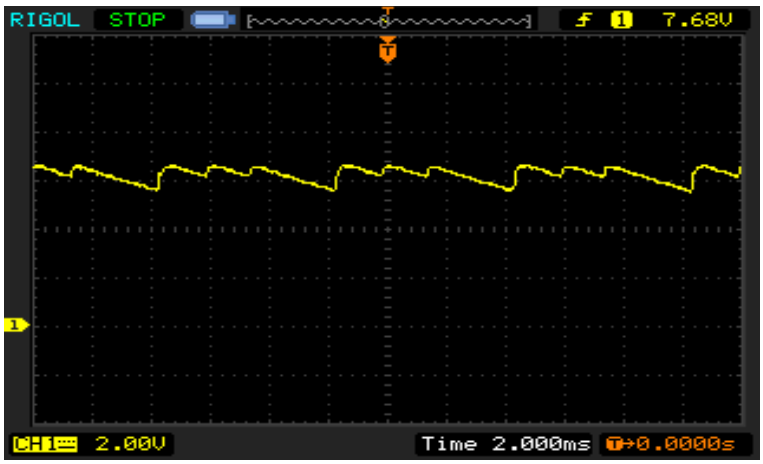


Illustration 15: Visualisation de la tension à la sortie du détecteur de crête

On obtient donc une tension ayant cette allure, avec toutefois une composante continue que l'on cherchera à supprimer par la suite.

6 - Soustracteur

Nous allons cette fois-ci utiliser un des AOP du LM324N en montage soustracteur. En effet, on remarque qu'à la sortie du détecteur de crête on obtient la forme du signal désiré. Toutefois pour envoyer cette tension sur la carte suivante qui va permettre l'affichage sur les LED, il est nécessaire que cette tension soit comprise entre 5V et 0V.

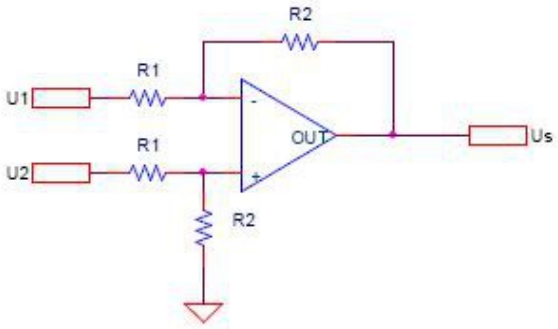


Illustration 16: Schéma électrique : AOP utilisé en montage soustracteur

$$Us = \frac{R_2}{R_1}(U_2 - U_1)$$

Le montage soustracteur va donc soustraire du signal sortant du détecteur de crête une tension continue. On aura donc en sortie de l'AOP uniquement la détection de crête.

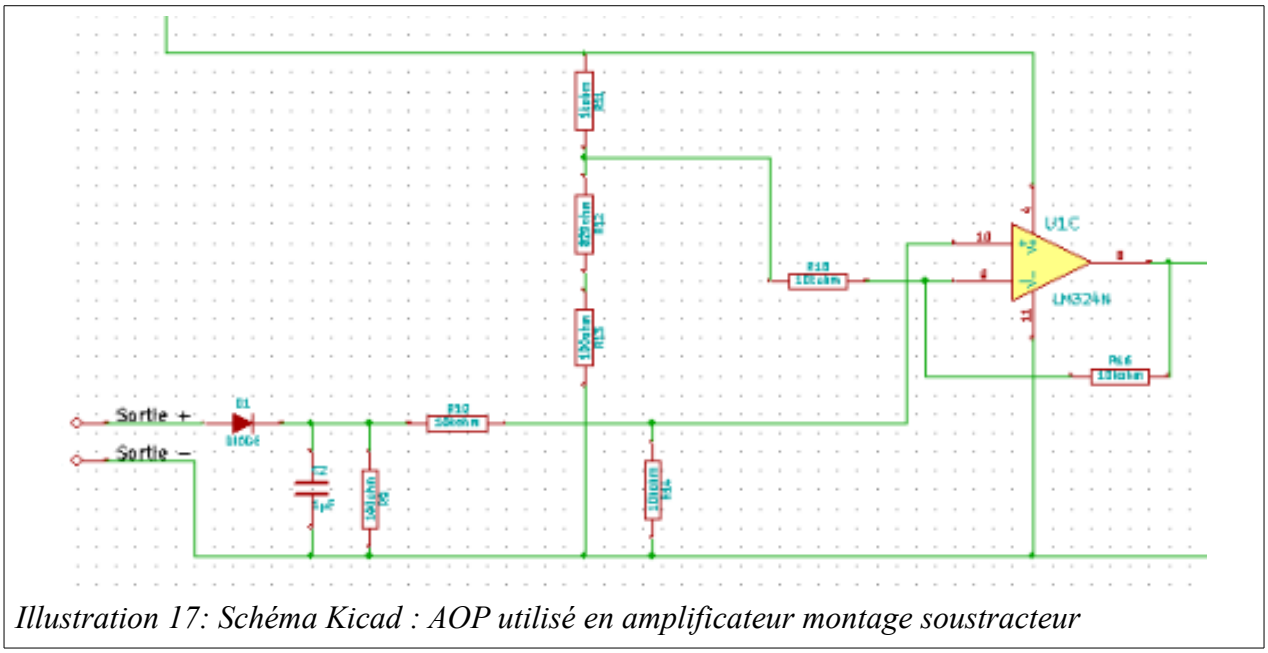


Illustration 17: Schéma Kicad : AOP utilisé en amplificateur montage soustracteur

7 - Montage complet

Nous assemblons ensuite les différents montages pour obtenir le système suivant.

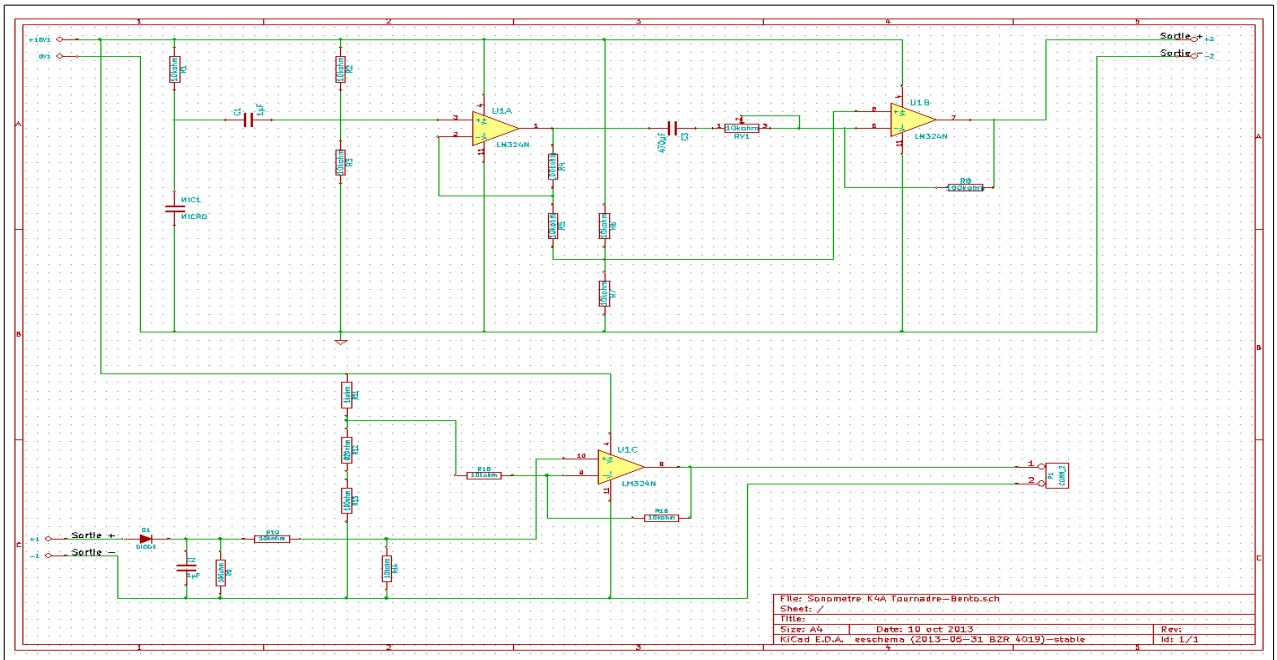


Illustration 18: Schéma Kicad : ensemble du montage comparateur, amplificateur, détecteur de crêtes et soustracteur

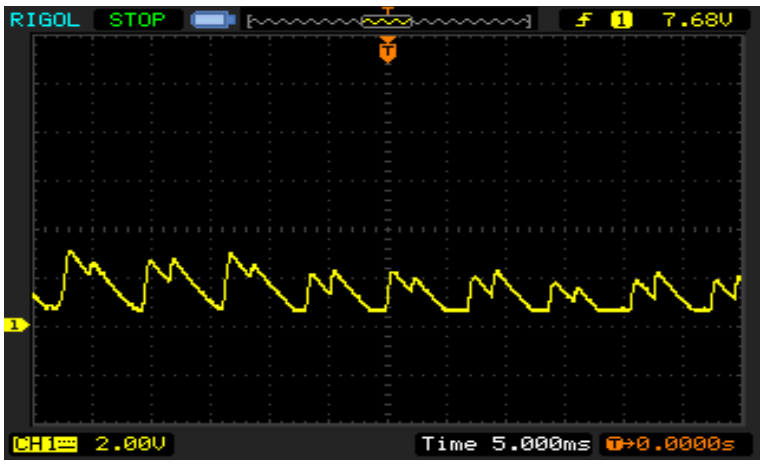


Illustration 19: Visualisation de la tension de sortie du montage

Nous obtenons donc en sortie du montage une tension comprise entre 5V et 0V. Elle pourra donc exploitable par la suite avec la carte où se trouve le composant ATMEGA.

I - programmation de l'atmega8535

1 - présentation de l'atmega8535

L'atmega8535 est le microcontrôleur que nous allons programmer afin qu'il puisse gérer la mesure de la tension récupéré et l'affichage de celle ci sur les LED.Cette carte est mise a notre disposition pour l'affiche sur un écran LCD.

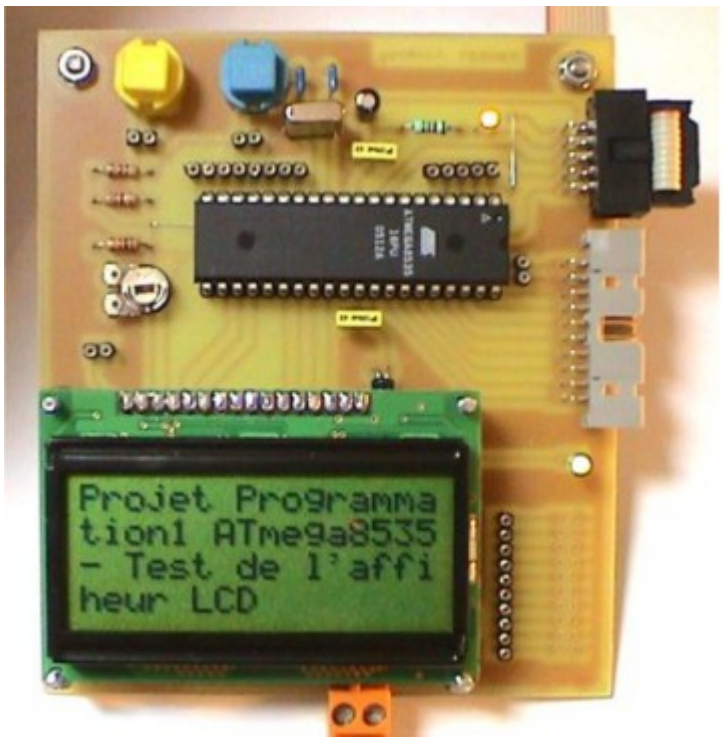


Illustration 20: Carte de programmation de l'ATmega et affichage

Dans ce projet l4ATmega8535 est déjà installé sur une carte mise à notre disposition. Pour implémenter les programme sur le microcontrôleur, il suffit de raccorder l'ATmega au bus du PC avec un connecteur de type J-Tag/HE10.

Illustration 21: Connecteur J-Tag

Il suffit ensuite d'exécuter le programme à l'aide d'un logiciel comme AVR Studio ou CodeVision AVR. Une fois la compilation effectuée le programme est chargé dans le microcontrôleur et est exécuté. l'ATmega est un composant CMOS[1] 8 bits qui est cadencé par une horloge de 16 mégahertz. Ce composant permet de programmer la partie intelligente de la carte. Les donnée échangées sont stockées dans des mémoires :

- 512 octets EEPROM[2]
- 512 octets SRAM[3]

2 - description des pattes de l'ATmega

l'ATmega comprend plusieurs ports bidirectionnels qui peuvent être des entrées comme des sorties.

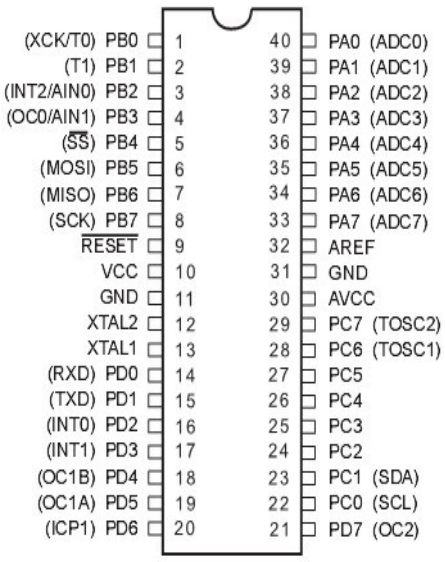


Illustration 22: Broches de l'ATmega8535

VCC est la broche d'alimentation principales de l'ATmega.

GND est la masse de l'ATmega.

Port A(PA0àPA7)

Port de 8 bits bidirectionnels, ils peuvent etre assignés comme entrées ou sorties. C'est un port analogique utilisé pour la conversion analogique-numérique. Cependant, ils sont toujours alimentés entre 0-5V.

Port B(PB0 àPB7)

porte de 8 bits bidirectionnels.

1 Complementary Metal Oxyde Semi-conductor.
2 Type de mémoire utilisé qui permet au programme de rester meme si l'ATmega n'est plus alimenté.
3 Tye de mémoire utilisé pour suprimé les fichiers temporaire une fois que l'ATmega n'est plus alimenté.

Port C(PC0 àPC7)

porte de 8 bits bidirectionnels.

Port D(PD0 àPD7)
porte de 8 bits bidirectionnels.

Reset peut générer une remise à zéro du système.

XTAL1 est une entrée d'horloge qui permet le fonctionnement de l'ATmega.

XTAL2 est une sortie de la patte inverseuse de l'amplificateur de l'oscillateur.

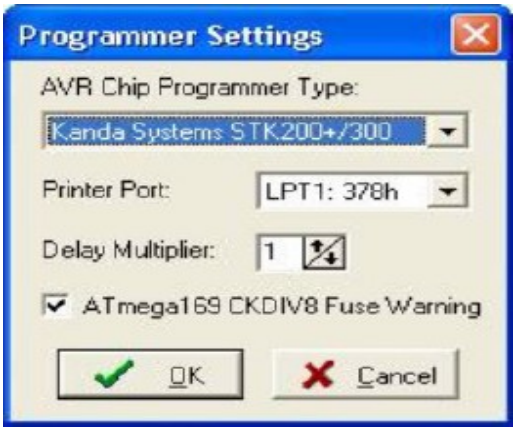
AVCC est la tension d'alimentation du port A et du convertisseur analogique-numérique.Cette patte doit être connectée à VCC même si le convertisseur n'est pas utilisé.

AREF est une patte analogique de référence pour le convertisseur.

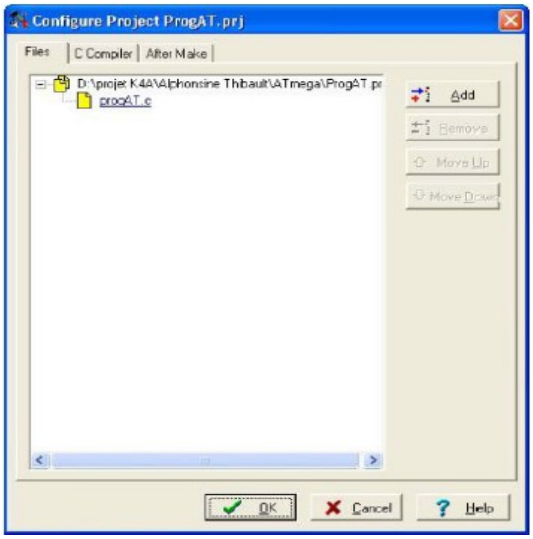
3 - Configuration du logiciel CodeVision AVR

Pour réaliser la programmation de l'ATmega, nous avons utilisé le logiciel CodeVision AVR qui permet d'écrire des programmes informatique en langage C et de programmer des microcontrôleur.

Dans la barre de Menu >>Setting>>Programmer, on choisit la puce Kanda Systems STK200+/300 pour la programmation de l'ATmega8535.

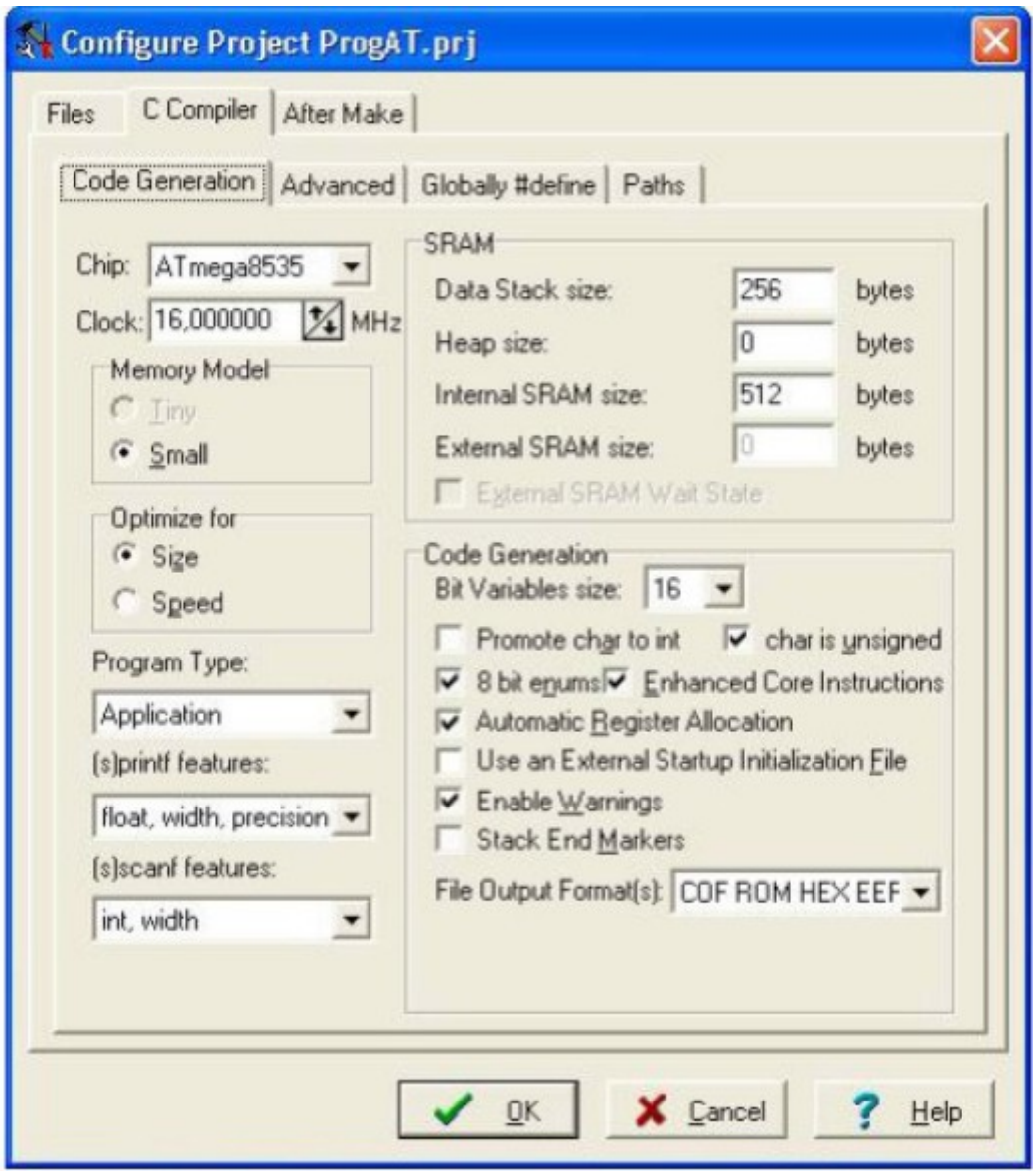


Ensuite, Tool>>Configure>>Configure project. Ceci permet de configurer l'emplacement où l'on veut enregistrer notre programme.

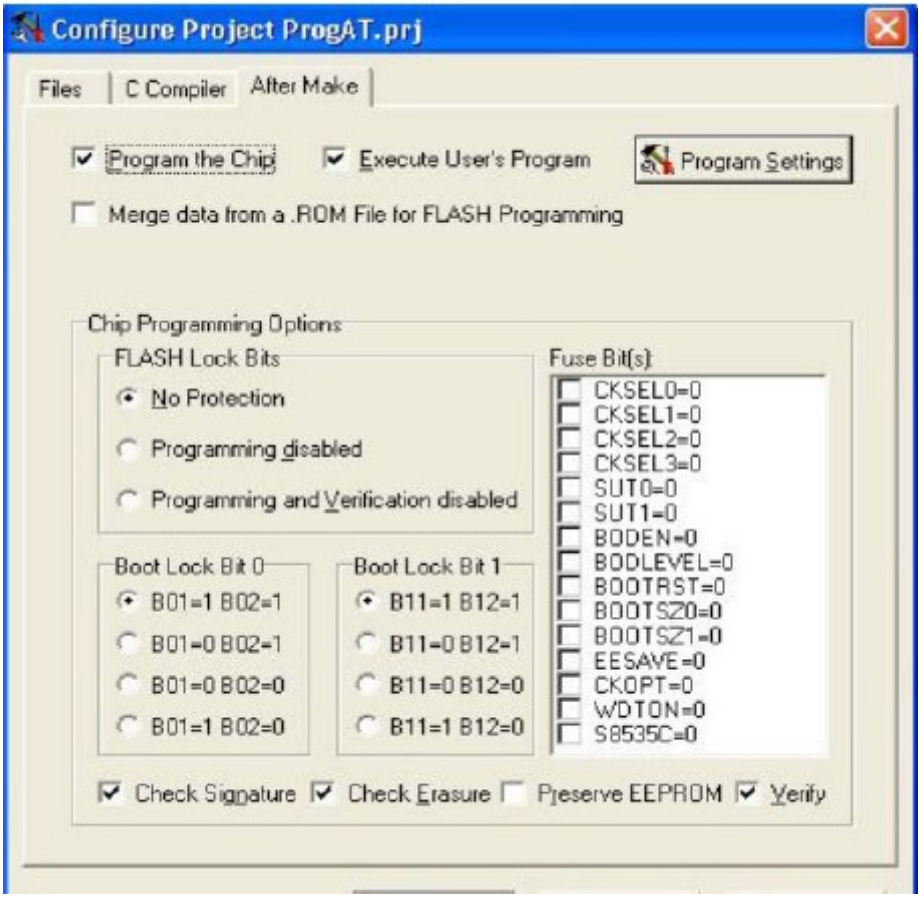


Une fois l'emplacement défini, on paramètre le composant ATmega8535 à une fréquence de fonctionnement 16Mhz. Dans la mémoire SRAM, les données doivent être contenues dans 256 octets. La mémoire interne SRAM de l'ATmega est de 512 octets.

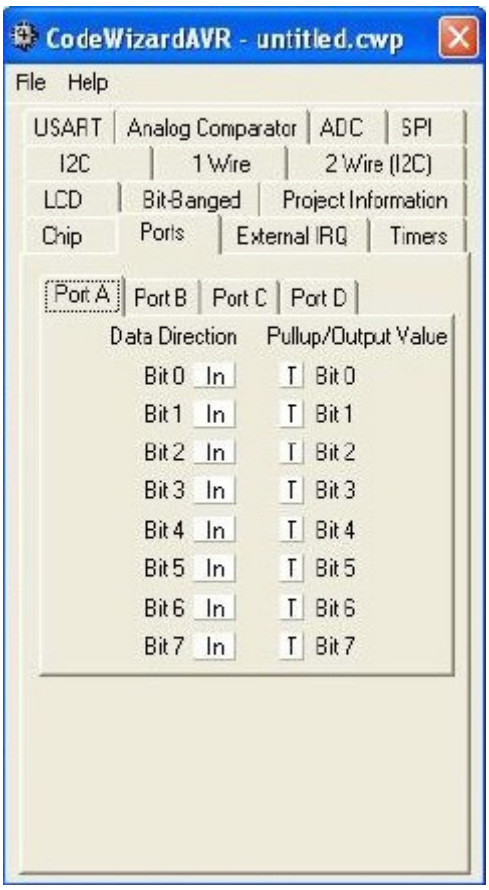
Pour le codage, on peut utiliser des variables de taille maximum 16 bits, de type caractère non signé(unsigned char en langage C qui permet de gérer des nombres positifs uniquement).


Configure Project ProgAT.prj
Files
C Compiler
After Make
Code Generation
Advanced
Globally #define
Paths
Chip:
ATmega8535
Clock:
16,000000
MHz
Memory Model
Tiny
Small
Optimize for
Size
Speed
Program Type:
Application
(s)printf features:
float, width, precision
(s)scanf features:
int, width
SRAM
Data Stack size:
256
bytes
Heap size:
0
bytes
Internal SRAM size:
512
bytes
External SRAM size:
0
bytes
External SRAM Wait State
Code Generation
Bit Variables size:
16
Promote char to int
char is unsigned
8 bit enums
Enhanced Core Instructions
Automatic Register Allocation
Use an External Startup Initialization File
Enable Warnings
Stack End Markers
File Output Format(s):
COF ROM HEX EEF
OK
Cancel
Help

Dans la configuration de l'ATmega, on transfert le programme dans la puce et on autorise l'exécution du programme.



On configure les ports de l'ATMEGA comme des entrées et des sorties selon les besoins du programme :


CodeWizardAVR - untitled.cwp
File Help
USART Analog Comparator ADC SPI
I2C 1 Wire 2 Wire (I2C)
LCD Bit-Banged Project Information
Chip Ports External IRQ Timers
Port A Port B Port C Port D
Data Direction Pullup/Output Value
Bit 0 In T Bit 0
Bit 1 In T Bit 1
Bit 2 In T Bit 2
Bit 3 In T Bit 3
Bit 4 In T Bit 4
Bit 5 In T Bit 5
Bit 6 In T Bit 6
Bit 7 In T Bit 7

4 - La commande des LED

Le contrôle des LED pour signaler le niveau de tension s'effectue selon l'ordinogramme suivant :

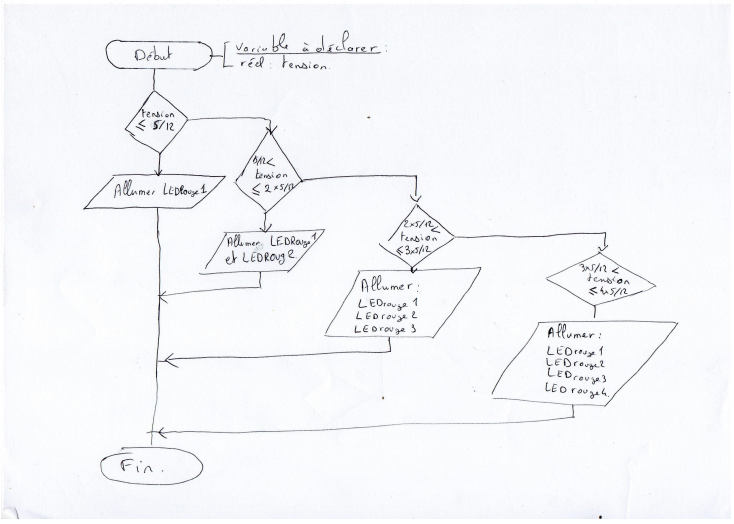


Illustration 23: Analyse fonctionnelle de l'allumage des LED

Voici l'analyse fonctionnelle de l'allumage de 4 LED sur les 24 pilotés par l'ATmega8535.On a comme tension de référence 5V que l'on divise par 12 . Car nous captons un son stéréo, à droite et à gauche, donc des signaux reçu sur ADC1 et ADC2, soit 12 pour la droite et 12 pour la gauche .

Les tensions délivrées étant des tensions analogiques, il nous a fallu les convertir en valeurs numérique pour pouvoir les traiter dans notre programme. Pour cela, nous avons utilisé le convertisseur analogique-numérique présent dans l'ATmega8535. Le programme complet est disponible en annexe, à la fin du dossier.

Conclusion

Nous avons tous deux trouvé le projet intéressant, de plus, il nous a permis de consolider nos bases et connaissances à la fois en électronique et en informatique. Nous les avons mises en pratique dans l'autonomie la plus complète dans un projet totalement personnel, ce qui a été pour nous une source de motivation.

Nous avons notamment pu mettre en œuvre nos connaissances sur les AOP ainsi que sur l'ATMEGA, ces composants sont énormément utilisés en électronique et nous familiariser avec la lecture de datasheets.

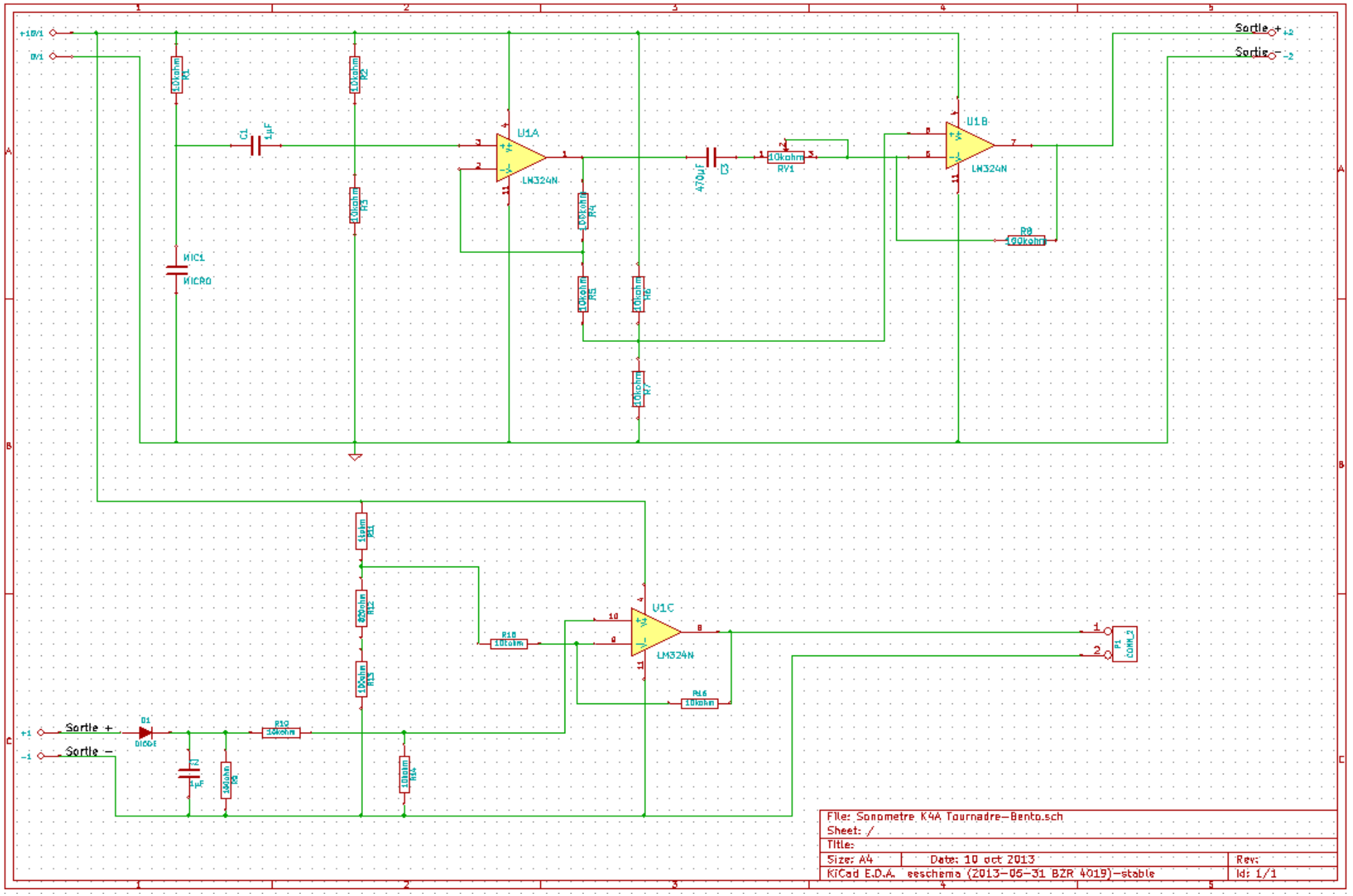
La partie réception et amplification du signal fonctionne. Sur la deuxième partie de notre montage qui est composée de l'ATMEGA, le programme comportant des erreurs le bon fonctionnement de la carte n'est donc pas possible.

C'est un projet qui nous aura également permis de nous habituer au travail en équipe et à mettre en place une bonne organisation afin que le projet se déroule le mieux possible. Ce fut très enrichissant pour nous deux.

Résumé

Dans le cadre de l'étude et réalisation au semestre 4, nous avons à mener un projet, puis à rédiger un rapport écrit et à le présenter lors d'un oral.Nous avons choisi comme projet de travailler sur la programmation d'un microcontrôleur, l'ATmega8535, qui récupère des niveaux de tensions transmis par des capteurs sonores et les visualiser sur des LED. Pour cela, nous avons utilisé le logiciel de programmation informatique CodeVision AVR. Nous avons utilisé le langage C pour écrire le programme. On a ensuite implémenté le programme dans l'ATmega, en le raccordant au bus du PC avec un connecteur J-Tag. Une fois exécuté, le programme traite la mesure de la tension analogique en la convertissant en valeur numérique et allume les LED en fonction de cette valeur. Lors des tests, nous avons rencontré des problèmes avec la compilation du programme, qui comporte des erreurs.

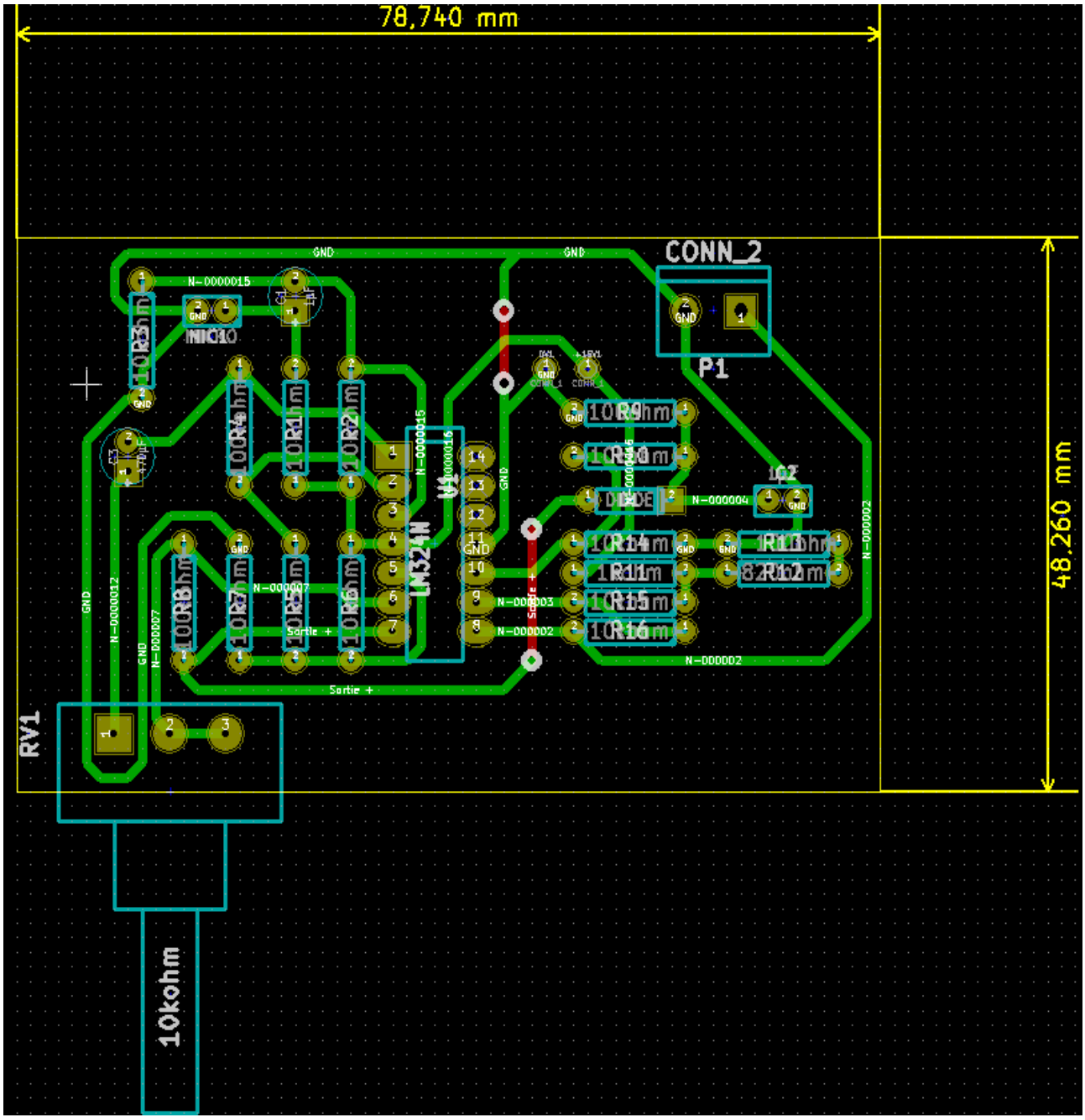
Annexes


+10V1
0V1
R1 10kohm
R2 10kohm
C1 1µF
R3 10kohm
MIC1
MICRO
U1A
LM324N
R4 100kohm
R5 10kohm
R6 10kohm
R7 10kohm
C3 470µF
RV1 10kohm
U1B
LM324N
R8 100kohm
Sortie +
Sortie −
P1 1ohm
R12 820ohm
R13 10kohm
U1C
LM324N
R10 10kohm
R16 10kohm
P1 CONN_2
+1
Sortie +
−1
Sortie −
D1 DIODE
C2 1µF
R9 10kohm
R11 10kohm
R14 10kohm
R15 10kohm
File: Sonometre K4A Tournadre-Bento.sch
Sheet: /
Title:
Size: A4
Date: 10 oct 2013
Rev:
KiCad E.D.A. eeschema (2013-05-31 BZR 4019)-stable
Id: 1/1

Netlist du capteur sonore

```
 1       +1 -           CONN_1 : PINTST
 2       +2 -           CONN_1 : PINTST
 3    +15V1 -           CONN_1 : PINTST
 4       -1 -           CONN_1 : PINTST
 5       -2 -           CONN_1 : PINTST
 6      0V1 -           CONN_1 : PINTST
 7       C1 -              1µF : C1V5
 8       C2 -              1µF : C1
 9       C3 -             470µF : C1V5
10       D1 -            DIODE : D3
11     MIC1 -            MICRO : C1
12       P1 -           CONN_2 : bornier2
13       R1 -           10kohm : R4
14       R2 -           10kohm : R4
15       R3 -           10kohm : R4
16       R4 -          100kohm : R4
17       R5 -           10kohm : R4
18       R6 -           10kohm : R4
19       R7 -           10kohm : R4
20       R8 -          100kohm : R4
21       R9 -           100ohm : R4
22      R10 -           10kohm : R4
23      R11 -            1kohm : R4
24      R12 -           820ohm : R4
25      R13 -           100ohm : R4
26      R14 -           10kohm : R4
27      R15 -           10kohm : R4
28      R16 -           10kohm : R4
29      RV1 -           10kohm : POTA_Fav
30       U1 -           LM324N : DIP-14__300_ELL
```

Première carte ; capteur sonore



```
Programme
/
*****************************************
*************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP
InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 22/10/2013
Author  :
Company :
Comments:


Chip type           : ATmega8535
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 128
*********************************
*****************/

#include <mega8535.h>
#include <stdio.h>
#include <delay.h>
#include <math.h>

//annonce des fonctions appelées
low1(float a,float x);
medium1(float b,float x);
high1(float c,float x);
/* deux fois le même type de fonction car
nous captons deux signaux (mode stéréo), à droite
et à gauche
   donc memes traitements mais pas les
memes commandes au niveau des ports à
activer.*/
low2(float a,float x);
medium2(float b,float x);
high2(float c,float x);



// déclaration des entrée

#define tension1 PINA.0
#define tension2 PINA.1

// déclaration desdifférentes sortie utilisé
pour le vumètre

#define ledV1 PORTB.0 //LED VERTES
#define ledV2 PORTB.1
#define ledV3 PORTA.2
#define ledV4 PORTA.3
#define ledJ1 PORTA.4 //LED JAUNES
#define ledJ2 PORTA.5
#define ledJ3 PORTA.6
#define ledJ4 PORTA.7
#define ledR1 PORTC.7 //LEDROUGES
#define ledR2 PORTC.6
#define ledR3 PORTC.5
#define ledR4 PORTC.4

#define led2V1 PORTB.0 //LED VERTES
#define led2V2 PORTB.1
#define led2V3 PORTA.2
```

```
#define led2V4 PORTA.3
#define led2J1 PORTA.4 //LED JAUNES
#define led2J2 PORTA.5
#define led2J3 PORTA.6
#define led2J4 PORTD.7
#define led2R1 PORTC.0 //LEDROUGES
#define led2R2 PORTC.1
#define led2R3 PORTC.2
#define led2R4 PORTC.3


unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
// Declare your global variables here


void main(void)
{
// Declare your local variables here


float tension_ADC1,tension_ADC2;
float X;


X=5/12;
```

```
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;


// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;


// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;


// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;


// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
```

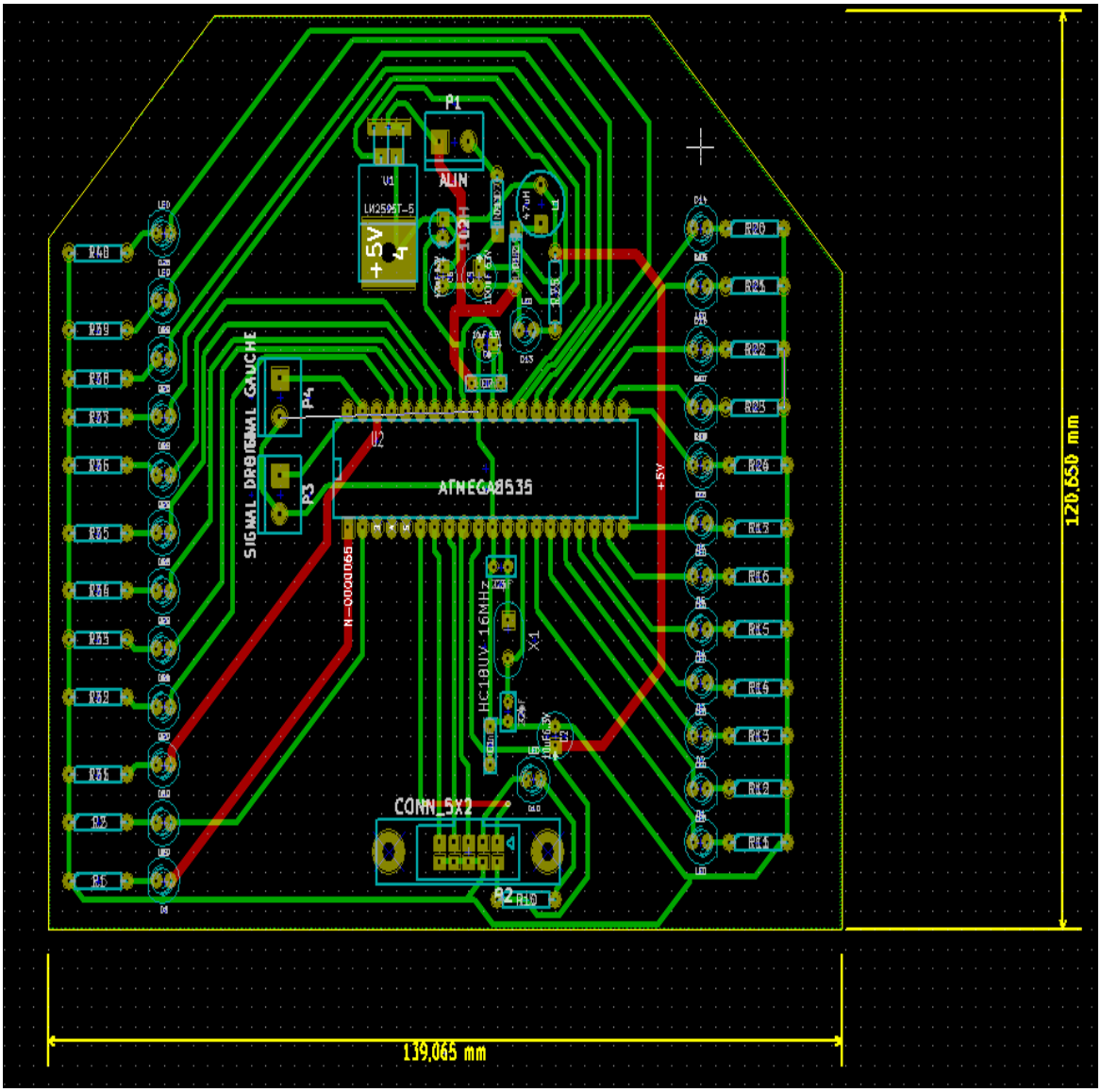
```
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

while (1)
    {

     tension_ADC1=read_adc(0);
     tension_ADC2=read_adc(1);

     low1(tension_ADC1);
     medium1(tension_ADC1);
     high1(tension_ADC1);

     low2(tension_ADC2);
     medium2(tension_ADC2);
     high2(tension_ADC2);
    };
}
```

```
low1(float a,float x)
{
      float A,B,C;

      A=X;
      B=2*A;
      C=3*A;

      if(a<A)
      ledV1=1;
      else
            if(a>A||a<B)
            {
              ledV1=1;
              ledV2=1;
            }
            else
                  if(a>B||a<C)
                  {
                    ledV1=1;
                    ledV2=1;
                    ledV3=1;
                  }
                  else
                        if(a>C)
                        {
                           ledV1=1;
                           ledV2=1;
                           ledV3=1;
                           ledV4=1;
                        }
}
medium1(float b,float x)
{
      float A,B,C;

      A=(5/12);
      B=2*A;
      C=3*A;

      if(a<A)
      ledJ1=1;
      else
            if(a>A||a<B)
            {
              ledJ1=1;
              ledJ2=1;
            }
            else
                  if(a>B||a<C)
                  {
                    ledJ1=1;
                    ledJ2=1;
                    ledJ3=1;
                  }
                  else
                        if(a>C)
                        {
                           ledJ1=1;
                           ledJ2=1;
                           ledJ3=1;
                           ledJ4=1;
                        }
}
high1(float c,float x);
{
      float A,B,C;

      A=(5/12);
```

```
        B=2*A;
        C=3*A;

        if(a<A)
        ledR1=1;
        else
            if(a>A||a<B)
            {
              ledR1=1;
              ledR2=1;
            }
            else
                if(a>B||a<C)
                {
                  ledR1=1;
                  ledR2=1;
                  ledR3=1;
                }
                else
                    if(a>C)
                    {
                      ledR1=1;
                      ledR2=1;
                      ledR3=1;
                      ledR4=1;
                    }
}

low2(float a,float x)
{
        float A,B,C;

        A=X;
        B=2*A;
        C=3*A;

        if(a<A)
        led2V1=1;
        else
            if(a>A||a<B)
            {
              led2V1=1;
              led2V2=1;
            }
            else
                if(a>B||a<C)
                {
                  led2V1=1;
                  led2V2=1;
                  led2V3=1;
                }
                else
                    if(a>C)
                    {
                      led2V1=1;
                      led2V2=1;
                      led2V3=1;
                      led2V4=1;
                    }
}
medium2(float b,float x)
{
        float A,B,C;

        A=(5/12);
        B=2*A;
        C=3*A;

        if(a<A)
        led2J1=1;
```

```
        else
            if(a>A||a<B)
            {
              led2J1=1;
              led2J2=1;
            }
            else
                if(a>B||a<C)
                {
                  led2J1=1;
                  led2J2=1;
                  led2J3=1;
                }
                else
                    if(a>C)
                    {
                       led2J1=1;
                       led2J2=1;
                       led2J3=1;
                       led2J4=1;
                    }
}
high2(float c,float x);
{
      float A,B,C;

      A=(5/12);
      B=2*A;
      C=3*A;

      if(a<A)
      led2R1=1;
      else
            if(a>A||a<B)
            {
              led2R1=1;
              led2R2=1;
            }
            else
                if(a>B||a<C)
                {
                  led2R1=1;
                  led2R2=1;
                  led2R3=1;
                }
                else
                    if(a>C)
                    {
                       led2R1=1;
                       led2R2=1;
                       led2R3=1;
                       led2R4=1;
                    }
}
```

carte n°2 ATmega8535



schémas n°2 ATmega8535