الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة البليدة 1
Université de BLIDA 1

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

LOCALISATION DE SOURCES ACOUSTIQUES PAR LES MÉTHODES D'INTER-CORRÉLATION GENERALISÉES

Mémoire de Master 2

Filière Électronique

Spécialité Traitement de l'Information et Systèmes

présenté par

SAHKI Faiza
&
DERRADJI Yasmine

Proposé par : Ykhlef Farid & Ykhlef Fayçal

Année Universitaire 2016-2017

Dédicaces

Je dédie ce travail

A mes enseignants du département d'électronique

A mon promoteur Monsieur Ykhlef Farid

A mon encadreur Mr Ykhlef Fayçal

A ma grand-mère Cherifa

A mes chers parents, en témoignage de monamour et de ma reconnaissance

A mon oncle Karim

A mon frère Anis, et ma sœur camélia

A tous mes collègues, plus particulièrement :

Chouaib, Islem, Mohamed

A tous mes amis, plus particulièrement :

Nassim, Imène, Ayoub, Faiza

A tous ceux qui m'ont soutenu, encouragéet aidé à réaliser ce travail.

YASMINE

Je dédie ce modeste travail

A vous très cher père et très chère mère, en témoignage de mon amour et de ma reconnaissance

A mes chères sœurs Fatima Zahra, Chahrazed, et Fatiha

A toute ma famille

A mon promoteur Mr Farid Ykhlef

A mon encadreur Mr Fayçal Ykhlef

A tous mes collègues, plus particulièrement : Fella, fadila, Zineb fouzia,Nouria,Asma,Naziha en témoignage de mon amitié sincère.

A tous mes amis, plus particulièrement : Hadjer, Bouchra,Yasser, Houceine, Abdelmalek, Omar, Amine, Sofiane, Chahinaz en témoignage de mon amitié sincère ;

A tous ceux qui de près ou de loin, m'ont apporté le soutien depuis le début de mon projet et dont j'ai oublié de mentionner les noms

A ma chère amie et binôme Yasmine et sa famille.

FAIZA

Remerciements

Nous remercions notre promoteur, monsieur Farid Ykhlef pour avoir assuré l'encadrement de ce travail ; Pour son soutien, et ses conseils. Son expérience et sa connaissance, ont contribué à notre formation scientifique.

Nous remercions également notre encadreur monsieur Fayçal Ykhlef, pour le privilège qu'il nous à fait en acceptant de diriger ce travail, sa disponibilité, sa riche expérience et l'accueil cordial nous a inspiré une grande admiration à son égard.

Nous exprimons notre profonde reconnaissance aux membres du jury pour avoir accepté de juger notre travail.

Nous souhaitons remercier également les enseignants du département d'électronique de l'université de SAAD DAHLEB-BLIDA et nos camarades, amis, ainsi que toutes les personnes qui nous ont aidées et soutenues.

Résumé

ملخص:

يركز هذا المشروع على تحديد موقع مصدر الصوت. تم تقييم خوارزمية من نوع - GCC-PHAT الذي يعمل في الوقت الحقيقي في سيناريو حقيق. تم تنفيذ هذا النظام لتحديد مصدر الصوت الصوتية باستخدام لوحة اردوينو واثنين من الميكروفونات.

كلمات المفاتيح: تحديد موقع مصدر الصوت ; كلمة المفاتيح ;خوارزمية من نوع - GCC-PHAT ; اردوينو.

Résumé :

Ce projet porte sur la localisation de sources sonores afin de déterminer l'emplacement spatial d'une source de son acoustique. Un algorithme de type GCC-PHAT opérant en temps réel a été évalué dans un scénario pratique. Ce système de localisation de source sonore acoustique a été réalisé à l'aide d'une carte Arduino et deux microphones.

Mots clés : localisation de sources sonores ; algorithme GCC-PHAT; Arduino.

Abstract :

This project focuses on the sound source location to determine the spatial location of an acoustic sound source. A real-time GCC-PHAT algorithm was evaluated in a practical scenario. This acoustic sound source location system was realized using an Arduino board and two microphones.

Keywords : sound source location; GCC-PHAT algorithm; Arduino.

Listes des acronymes et abréviations

SSL : Sound Source Localization.

GCC: Generalized Cross-Correlation.

PHAT: PHAse Transform.

TDOA: Time Difference Of Arrival.

ITD : Interaural Time Difference.

ILD : Interaural Level Difference.

SNR : Signal to-Noise Radio.

USB : Universel Serial Bus.

AVR : Automatic Voltage Regulator.

PWM : Pulse Width Modulation.

CAN: Convertisseur Analogique Numérique.

CNA : Convertisseur Numérique Analogique.

TDE: Time Delay Estimation.

GCC-PHAT : Generalized Cross-Correlation-PHAse Transform.

SRP-PHAT: Streered Reponse Power-PHAse Transform.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Les mécanismes de localisation de la source sonore (SSL) ont été largement étudiés. Beaucoup d'applications comme les systèmes de téléconférences ou d'amélioration de la parole nécessitent la localisation d'une ou plusieurs sources acoustiques. En outre, il est essentiel de localiser les sources dans des environnements bruyants et réverbérant. Les stratégies de localisation des sources sonores remontent aux systèmes de localisation radar et sonar.

La localisation d'une source sonore (SSL) est le processus de détermination de l'emplacement spatial d'une source de son.

Il existe deux classes principales pour la SSL : **les approches directes et indirectes.**

Les approches directes choisissent parmi l'ensemble des positions de source, le candidat le plus probable pour l'estimation de l'emplacement de la source sonore. Tandis que **les approches indirectes** se basent sur la localisation.

Pour les deux approches, des techniques telles que la méthode d'inter-corrélation généralisée (GCC) proposée par Knapp et Carter en 1976 [1], sont largement utilisées.

Il existe une autre classe d'algorithmes SSL importante basée sur un formateur de faisceau dirigé. Néanmoins les performances de localisation des techniques classiques de faisceaux de direction appliquent des filtres aux signaux de réseaux afin d'améliorer leurs performances.

À l'heure actuelle, l'algorithme SRP-PHAT (Steered Response Power- PHAT « puissance de réponse dirigée ») est devenu la méthode de localisation la plus populaire pour ses bonnes performances en environnement réel [2]. Toutefois, la complexité des calculs de cette méthode rend difficile sa mise en œuvre en. Pour cela on a opté pour la méthode d'inter-corrélation généralisée (GCC-PHAT) dont sa mise en œuvre est moins complexe pour de meilleurs résultats en temps réel.

Ce mémoire se compose de cinq chapitres comme suit :

Dans le premier chapitre nous donnons des définitions générales sur la localisation de la source sonore et leurs principales approches.

Le deuxième chapitre est consacré à la simulation et la validation expérimentale.

Dans le troisième chapitre, on décrira l'acquisition et le traitement des données et nous présentons les différents composants de notre système de localisation.

Le quatrième chapitre exposera l'implémentation de notre système, les différents logiciels et langages manipulés pour la mise en œuvre du projet. Aussi, ce chapitre présentera les tests, les résultats et ainsi que les difficultés rencontrée durant la réalisation de ce travail.

Chapitre 1 Généralités sur la localisation des sources sonores

1.1 Introduction

Ce chapitre a pour objectif de répertorier les informations fondamentales permettant de se familiariser avec les différentes notions et concepts liés à la localisation d'une source sonore (LSS). Dans une première partie, nous donnons quelques définitions relatives à la localisation d'une source sonore. Une deuxième partie aura pour but de décrire les différentes approches de la localisation d'une source sonore. Dans la troisième partie, nous exposerons les méthodes généralisées de la localisation de sources sonores.

1.2 Notions sur l'acoustique

1.2.1 Qu'est- ce qu'un son ?

Le son est le résultat de variations de pression, ou d'oscillations, dans un milieu élastique (air, eau, solide), généré par une surface vibrante ou un écoulement de fluide turbulent [3]. C'est aussi la sensation auditive à laquelle cette vibration est susceptible de donner naissance.

Dans un milieu compressible comme l'air, le son se propage sous forme d'une variation de pression (enchainement compression/dépression) engendré par la source sonore : le mouvement d'un diapason ou un haut-parleur, par exemples utilisent ce mécanisme.

Le son se propage aussi dans les solides, sous forme de vibrations des atomes (seule la vibration se propage, et non les atomes qui ne font que vibrer très faiblement autour de leur position d'équilibre). Le son se propage plus mal à l'horizontale que sous des angles montants, à cause du changement de densité de l'air qui varie avec l'altitude. Cette propriété a été prise en compte depuis l'antiquité dans la conception des théâtres en plein air par les Grecs ou les Romains par exemple.

Les ondes sonores se déplacent à environ 344 m/s (mètres par seconde) dans de l'air à une température de 20°C (Celsius) , vitesse qu'on peut arrondir à environ un kilomètre toutes les trois secondes, ce qui est utile pour mesurer grossièrement la distance d'un éclair lors

d'un orage, la vitesse de la lumière rendant sa perception quasi instantanée. En milieu non gazeux, le son peut se propager encore plus rapidement, dans l'eau, sa vitesse est de 1482 m/s ; dans l'acier de 5050 m/s. Il ne se propage pas dans le vide car il n'y a pas de moyen pour supporter les ondes produites. Le vide est donc un excellent isolant phonique [3].

1.2.2 Principe de la production d'un son

La production d'un son par l'appareil phonatoire humain repose sur le principe suivant :

Une soufflerie (appareil respiratoire) crée une pression d'air engendrant au niveau d'un vibreur (cordes vocales) la création d'un son qui est ensuite modulé par la partie résonatrice du système (les cavités buco-nasales).

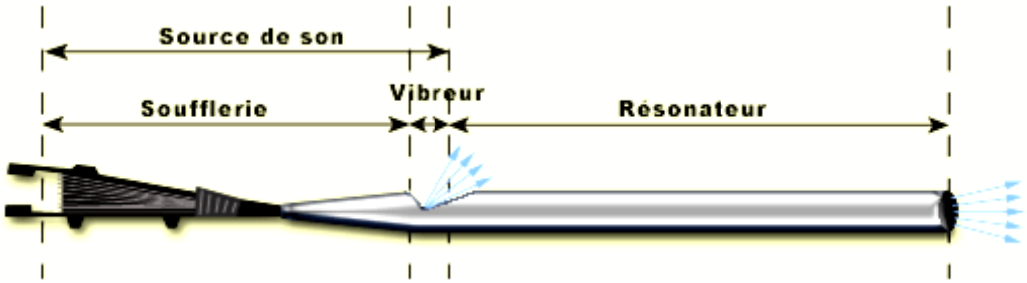


Figure 1.1 Principe de la production d'un son de l'appareil phonatoire humaine.

1.2.3 Caractéristique d'un son

1.2.3.1 Célérité et vitesse de propagation

La vitesse du son (sa célérité) va dépendre de la densité du milieu de propagation et de la température. Plus le milieu sera dense et plus le son se propagera vite. De même que la température sera élevée et plus le son se propagera vite.

A une température de 0° C, la vitesse du son dans l'air est de 330m/s. A 20° C, la vitesse du son dans l'air est de 340m/s. c'est cette valeur qui est la plus utilisée pour les calculs.

1.2.3.2 Fréquence

La fréquence correspond au nombre d'oscillations périodiques par unité de temps. La période est le temps que met une onde sonore pour accomplir un cycle complet. On la note T et est mesurée en seconde.

1.2.3.3 Longueur d'onde

La longueur d'onde (notée λ) est la distance parcourue par une onde pendant une période. Cette distance est calculée en mètre.

$$\lambda = c/f$$

c : célérité (340m/s à 20°C)

f : fréquence en Hertz.

1.2.4 Localisation d'une source sonore chez l'être humain

La localisation sonore désigne la capacité du système auditif à déterminer la position spatiale d'une source sonore au moyen de différents indices physiques. Ces indices peuvent être classés en deux catégories : les indices binauraux (c'est-à-dire issus de l'analyse combinée des sons parvenant aux deux oreilles) et des indices monauraux (c'est-à-dire ceux que l'on peut déterminer avec une seule oreille).

Chez l'être humain, le principal indice utilisé pour la localisation sonore est la comparaison Interaural : la différence de niveau et de délai. L'ingénierie acoustique exploite ce phénomène pour reproduire une impression d'espace sonore, par exemple via la stéréophonie ou les systèmes de enceintes home cinéma. Un autre indice est le HRTF, désignant la transformation du son par la tête, le cerveau étant capable d'interpréter la provenance d'un son par sa coloration.

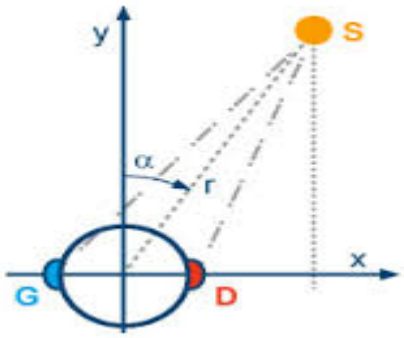


Figure 111.2 Localisation de la source sonore [4].

1.3 Les principales approches de la localisation d'une source sonore

La localisation spatiale d'une source sonore est nécessaire dans des nombreuses applications. Plusieurs procédés et systèmes de localisation spatiale d'une ou plusieurs sources sonores sont connus [2]. Ces procédés et systèmes se basent généralement sur une pluralité de microphones peu ou pas directifs et sur un traitement numérique des signaux captés par les microphones.

La première approche : utilise des techniques d'estimation spectrale, basées sur la matrice de corrélation des signaux captée par les microphones. Les méthodes basées sur cette approche tendent à être sensibles aux erreurs de modèles et très demandeuses de puissance de calcul. Elles conviennent principalement pour des signaux à bande étroite.

La deuxième approche : se base sur l'estimation des décalages temporels entre les signaux sonores reçus par des paires de microphones (TDOA, pour « Time Différence Of Arrival », c'est-à-dire « différence de temps d'arrivée ». Ces estimations sont utilisées, avec la connaissance des positions des microphones, pour calculer des courbes hyperboliques, dont l'intersection donne la position de la source. Les décalages temporels peuvent notamment être estimés par la méthode dite PHAT-GCC (« PHAseTransform-Generalized Cross-Correlation», ou « transformation de phase- inter-corrélation généralisée») qui exploite le calcul d'une inter-corrélation- ou corrélation croisée- entre signaux préalablement « blanchi» par filtrage. Ces méthodes sont légères computationnelles, mais elles ne sont pas robustes au bruit corrélé provenant de sources multiples et sons sujettes aux «faux positifs». En outre, elles sont peu robustes à la réverbération, à l'exception de la méthode PHAT-GCC.

La troisième approche : consiste à synthétiser un faisceau acoustique orientable en additionnant les signaux captés par les différents microphones auxquels a été appliqué un décalage temporel variable, et à identifier l'orientation du faisceau qui maximise la puissance du signal composite ainsi reçu. Les méthodes basées sur cette approche tendant à être peu robustes à la réverbération et au bruit, sauf certaines variantes qui sont cependant très demandeuses de puissance de calcul.

L'invention vise à procurer un procédé de localisation d'une source sonore présentant des meilleures propriétés d'immunité à la fois au bruit et à la réverbération par rapport aux procédés connus, tout en étant suffisamment léger du point de vue computationnel pour être mise en œuvre dans un système embarqué tel qu'un robot humanoïde, en temps réel et en parallèle d'autres tâches.

Un objet de l'invention est donc un procédé de localisation d'une source sonore comportant les étapes suivantes [2]:

a. Capter des signaux sonores issus d'une source sonore à localiser au moyen d'un ensemble d'au moins deux microphones.

b. Sélectionner au moins deux paires de microphones de l'ensemble et pour chaque paire, calculer une inter-corrélation généralisée des signaux sonores captés, le calcul étant

effectué pour une pluralité de valeurs d'un retard- différence de temps interarticulaire- entre les signaux sonores.

c. à partir des inter-corrélations généralisées, calculer une puissance de réponse dirigée exprimée en fonction d'un vecteur des différences de temps inter-auriculaires pour chaque paire de microphones.

d. Déterminer le vecteur des différences de temps inter-auriculaires qui maximise la puissance de réponse dirigée.

e. Estimer une direction de localisation de la source sonore en fonction du vecteur des différences de temps inter-auriculaires déterminé lors de l'étape d.

1.4 Méthodes de localisation d'une source sonore

Dans la littérature, il existe plusieurs méthodes pour la détermination de la direction d'arrivée d'une onde sonore. Les deux les plus utilisées sont l'ILD (Interaural Level Difference) et le TDOA (Time Difference Of Arrival) connue aussi sous le nom d'ITD (Interaural Time Difference).

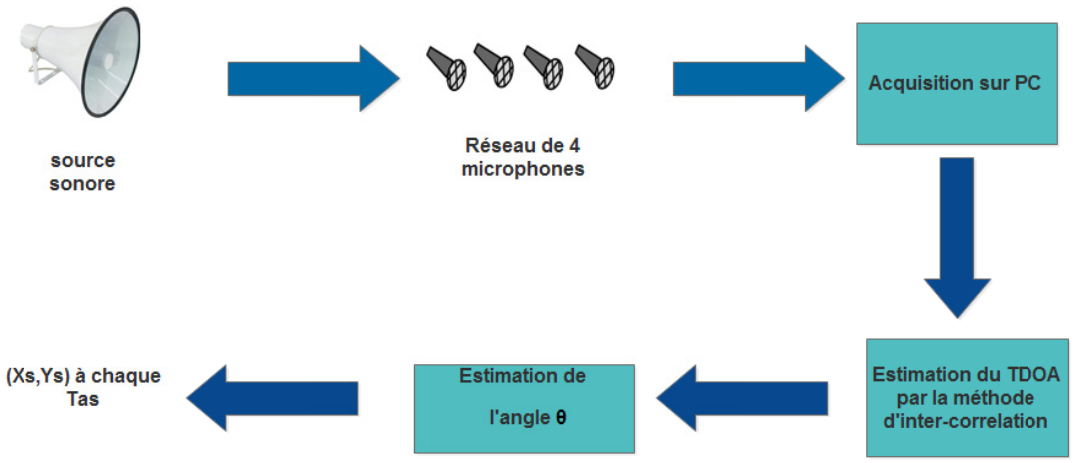


Figure 1.3 Localisation d'une source sonore par un réseau de 4 microphones.

1.4.1 Méthode d'ILD

L'ILD (Interaural Level Differences) mesure la différence d'intensité sonore entre les signaux reçus par deux récepteurs (microphones ou oreilles) qui caractérisent la différence "d'intensité sonore" entre deux récepteurs. Nous savons que l'intensité sonore varie inversement proportionnelle au carré de la distance (I=W/4 pi) tel que I est l'intensité sonore en watts/m², et d est la distance de la source en mètres. Donc connaitre l'intensité de

chaque microphone permet d'en calculer la différence et donc de déterminer la direction du son. L'avantage de cette méthode est qu'elle est simple et que la différence permet immédiatement, si elle est positive ou négative, de se rendre compte si l'interlocuteur se trouve à gauche ou à droite [5].

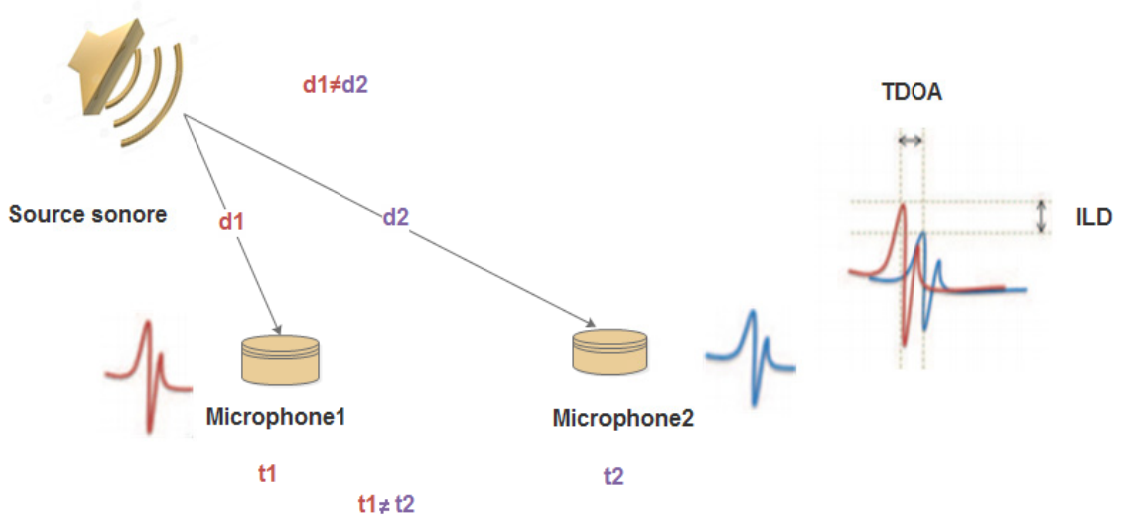


Figure 1.4 Méthode de calcul de la direction d'une source sonore.

La méthode est basée sur la puissance du signal. Seule l'indication sommaire d'une direction est prise en compte. Pour la déterminer, l'idée est partie de pouvoir évaluer la "puissance" contenue dans un son. Pour un signal quelconque, la mesure de puissance (P) peut être assimilée à sa valeur efficace, soit pour un signal périodique s(t) de période T :

$$P = \frac{1}{T}\int_0^T s^2(t)dt \qquad (1.2)$$

Pour un signal quelconque s(t) :

$$P = K\int_0^{t_1} s^2(t)dt \qquad (1.3)$$

avec K est une constante et t_1 est le temps d'intégration.

En mesurant la puissance des sons reçus par différents microphones et en comparant leur valeur les unes aux autres, il semble possible de pouvoir localiser la direction de la source sonore. En effet, plus un microphone est près d'une source sonore, plus la "puissance" qu'il reçoit est importante. Cependant ces méthodes restent limitées car elles ne permettent qu'une localisation approximative. En effet, dans le cas où l'on utilise deux microphones, la localisation se fait à 180° et ne donne donc qu'une précision relative du type gauche ou droite. Pour quatre micros, la localisation donne une précision à 90 °, etc.

1.4.2 Méthode TDOA

L'une des méthodes les plus utilisées est la méthode de la différence du temps d'arrivée (TDOA) qui permet d'obtenir de bons résultats de localisation.

Si l'on considère le cas d'un réseau de deux capteurs (microphones), le principe de l'estimation de la direction d'arrivée d'une onde sonore repose sur le calcul de ses temps de capture par chaque microphone puis en déduire le retard (τ) ou la différence des temps d'arrivée (TDOA) enregistrée entre ces deux microphones. Ainsi, à l'instar du système auditif humain, l'exploitation du retard que met un son pour arriver à deux capteurs (microphones) pourrait nous conduire à déterminer la direction d'arrivée de ce son.

La méthode TDOA consiste à mesurer l'instant d'arrivée d'un signal en plusieurs points de l'espace et à comparer la différence entre les instants mesurés à chaque récepteur. La méthode classique utilisée pour estimer la différence TDOA consiste à calculer la corrélation croisée d'un signal arrivant dans deux récepteurs.

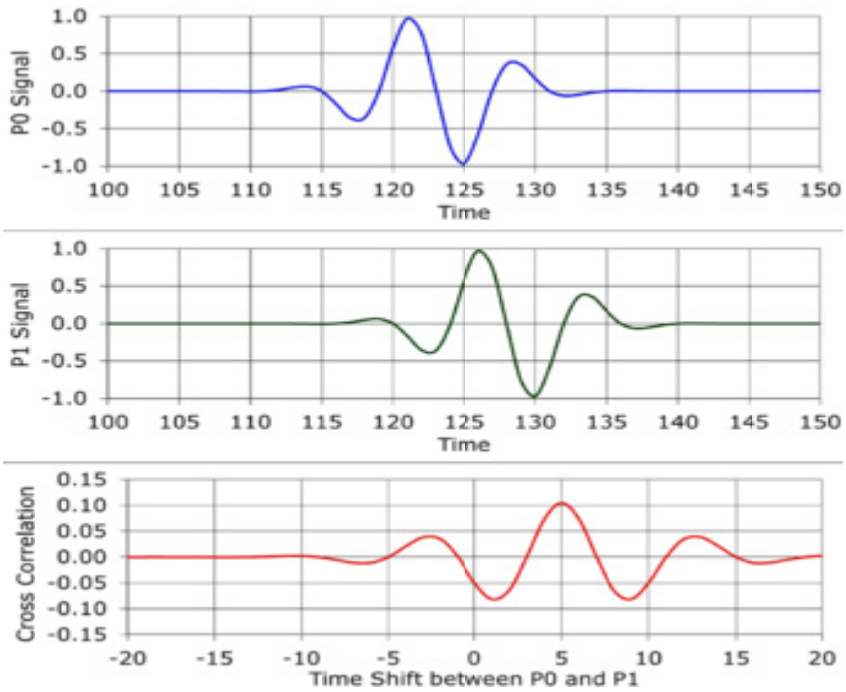


Figure 1.5 Application de méthode TDOA.

Cette estimation correspond au délai qui maximalise la fonction de corrélation croisée. Si on connaît l'emplacement de chaque récepteur, on peut alors en déduire une estimation de l'emplacement de la source des émissions sous réserve que tous les récepteurs soient synchronisés temporellement.

1.4.3 Choix de la méthode de localisation

La méthode TDOA est beaucoup plus précise et robuste aux perturbations que la première. En revanche, elle est beaucoup plus difficile à mettre en œuvre étant donné que les traitements et calculs sont lourds. En effet, il est nécessaire d'effectuer une corrélation entre les deux signaux enregistrés afin de déterminer le décalage.

Malgré tout nous avons besoin d'une méthode robuste pour effectuer le travail. En effet, la mesure approximative de l'ILD ne correspond pas aux objectifs fixés, c'est pourquoi, la méthode TDOA a été retenue [5].

Remarque : Pour localiser une source sonore dans un environnement 3D, il faut au moins 4 microphones (Dans notre cas, en 2D, 2 microphones suffisent).

1.4.4 Illustration pour une localisation sonore utilisant deux microphones

Nous considérons un système de deux capteurs et une source sonore placée en dehors du plan perpendiculaire à la droite passant par ces deux capteurs et passant par le milieu de leur entraxe. Le front d'onde de cette source sonore atteindra alors les deux capteurs avec une légère différence de temps ou délai, dépendant de l'angle d'arrivée de la source. La valeur maximale que peut prendre ce délai renvoie à une source sonore se trouvant sur la même droite passant par les deux capteurs (à 0° ou 180°). Le réseau que nous utilisons est composé de deux microphones pouvant être disposé comme illustré par la Figure (1.6).

a) Principe de la méthode d'inter-corrélation simple

Le but est de déterminer l'angle θ, pour cela on doit :

- fixer la distance entre les deux microphones, D=10 cm
- déterminer la période d'échantillonnage Te, tel que Te=1/Fe
- calculer le TDOA (a) : a=N*Te (grâce a l'inter-corrélation)
- utiliser la fonction d'inter-corrélation pour estimer le retard entre les deux signaux $x(t)$ et $y(t)$ acquis par les microphones:

$$C_{xy}(t) = \int_{-\infty}^{+\infty} x(t).y(t-\tau).d\tau \qquad (1.4)$$

Pour déterminer la différence de temps, on recherche la position temporelle par le maximum de la fonction d'inter-corrélation.

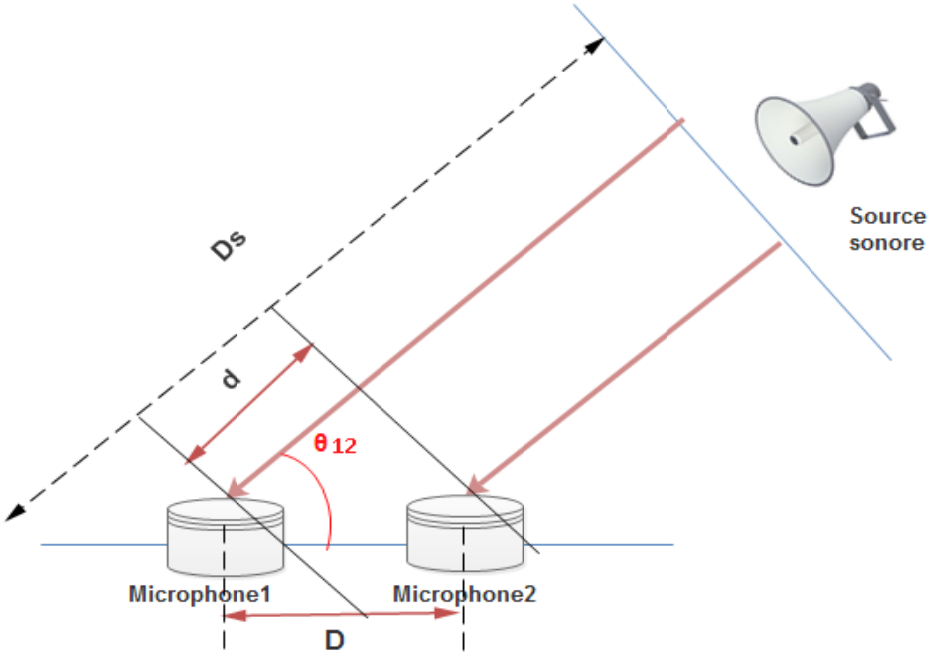


Figure 1.6 Réseau composé de deux microphones.

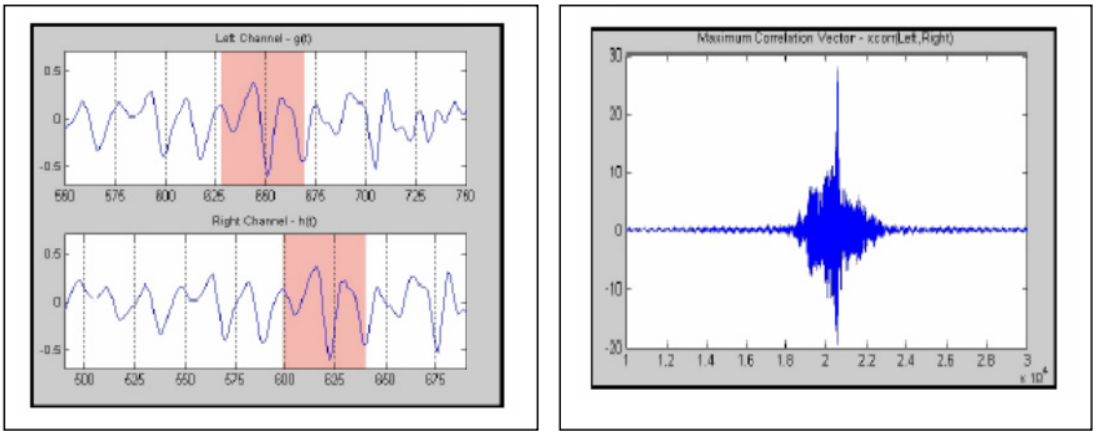


Figure 1.7 Illustration de l'introduction pour un son.

On determine d:

$$d = a.c \tag{1.5}$$

c : la vitesse de propagation du son dans l'air.

Selon la formule de Laplace :

$$c = (331 + 0.6.T)\ m/s \tag{1.6}$$

où T représente la température en degré Celsius.

Donc pour une température de 25°C on aura $c = 346\ m/s$.

où « D » et « a » sont déterminés par les formules trigonométriques

$$\cos\theta = (a.c/D) = x \quad (1.7)$$

d' où

$$\theta = \cos^{-1} x \quad (1.8)$$

b) Difficultés et intérêt de la méthode d'inter-corrélation simple

La méthode est assez simple certes, mais elle cache des problèmes liés a la cohérence entre les signaux issus des microphones, des problèmes dus aux bruits parasites corrèles, car en effet la méthode fait l'hypothèse d' inter-corrélation du bruit de fond entre les deux microphones, cette méthode n'est pas vérifiée, mais cela se traduit par l'apparition de pics parasites que l'on peut négliger a fort rapport signal bruit, il nous semble que cette méthode d'inter-corrélation ne soit pas la bonne pour la localisation de source dans notre cas, nous avons toutefois décidé de la mettre en œuvre en espérant que prés de la source, la cohérence sera suffisante a la détermination du temps de retard.

1.5 Méthodes d'inter-corrélations généralisées

Les méthodes d'inter-corrélation généralisées consistent à pré-filtrer chacun des signaux $x_1(t)$ et $x_2(t)$ à l'aide d'une fonction de pondération, avant d'effectuer l'inter-corrélation la fonction de pondération qui intervient dans l'inter-corrélation résultant de chacune de ces deux fonctions de pondération.

1.5.1 L'inter-corrélation généralisée (GCC)

L'inter-corrélation généralisée est une méthode populaire pour déterminer la différence de temps d'arrivée (TDOA) entre deux microphones dans une paire. Ensuite, à partir de multiples valeurs TDOA, on peut estimer l'emplacement source. On prend un groupe de microphones à 4 éléments comme l'exemple illustré à la figure (1.8).

Si la distance entre le microphone et la source est r m tel que (m = 1,2,3,4), le délai (en déplacement temps) du signal de la source à ce microphone est :

$$\tau_m = \frac{r_m}{c} \quad (1.9)$$

Ensuite, la différence de temps d'arrivée, TDOA entre deux microphones m et n peut être défini comme :

$$\tau_{mn} = \tau_m - \tau_n = \frac{\tau_m - \tau_n}{c} \quad (1.10)$$

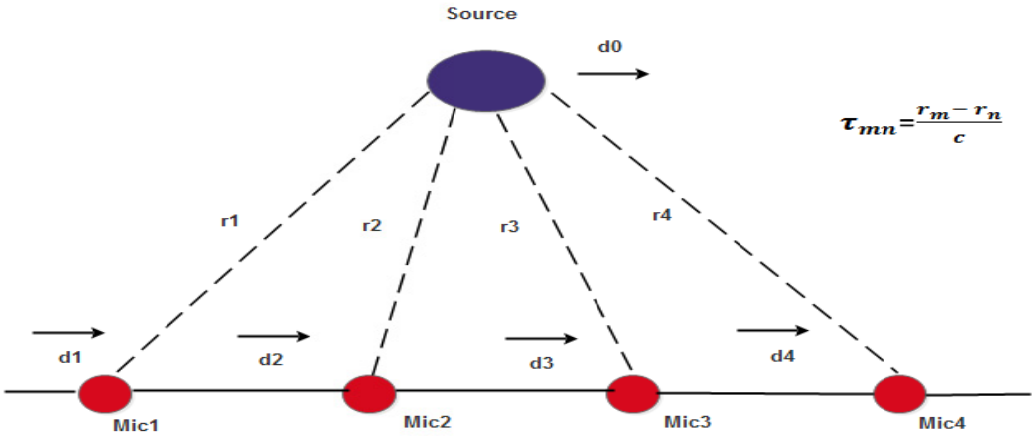


Figure 1.8 TDOA entre deux microphones.

A partir de cette relation entre le TDOA et les distances entre la source et le microphone, τ_m, on peut estimer l'emplacement source à partir de TDOA multiples.

1.5.2 Dérivation du CCG

Considérons un signal de microphone au microphone k :

$$x_k(t) = s(t) * h(d_s, t) + n_k(t) \quad (1.11)$$

Soit un signal sur un autre microphone l :

$$x_l(t) = s(t - \tau_{kl}) * h(d_s, t) + n_l(t) \quad (1.12)$$

Pour être précis, nous devrions inclure le délai τk Dans la source du signal $s(t)$, c'est-à-dire $s\,(t - \tau k)$ Dans l'équation (1.4) ci-dessus pour montrer le signal reçu au microphone k est une version retardée du signal source. Cependant, pour simplifier, nous nous sommes normalisées pour que le délai de la source au microphone k, τ_k est 0. En d'autres termes, nous ne sommes concernés que par La différence de temps relative d'arrivée, Ckl entre ces deux microphones k et l.

L'inter-corrélation de ces deux signaux de microphone montre un pic au décalage horaire où ces deux signaux décalés sont alignés, correspondant au TDOA, C_{kl}.

L'inter-corrélation de $x_k(t)$ et $x_l(t)$ est défini comme :

$$C_{kl}(\tau) = \int_{-\infty}^{+\infty} x_k(k) x_l(t + \tau) dt \quad (1.13)$$

$$C_{kl}(\omega) = \int_{-\infty}^{+\infty} C_{kl}(t) e^{j\omega\tau}\, d\tau \qquad (1.14)$$

On applique des propriétés de convolution de la transformée de Fourier pour (1.13) lors de la substitution, En (1.14), on a :

$$C_{kl}(\omega) = X_k(\omega) X_l^*(\omega) \qquad (1.15)$$

où $X_l(\omega)$ est la transformée de Fourier du signal $x_l(t)$, et '*' désigne le conjuguer.

La transformée inverse de Fourier de (1.15) nous donne la fonction inter-corrélation en termes de transformée de Fourier des signaux du microphone :

$$C_{kl}(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_k(\omega) X_l^*(\omega) e^{j\omega\tau} d\omega \qquad (1.16)$$

L'inter-corrélation généralisée (GCC) de $x_k(t)$ et $x_l(t)$ est l'intercorrélation de leurs deux versions filtrées. On note les transformées de Fourier de ces deux filtres en tant que $W_k(\omega)$ et $W_l(\omega)$, nous avons le CCG, $R_{kl}(t)$ est défini comme :

$$R_{kl}(\tau) \equiv \frac{1}{2\pi} \int_{-\infty}^{\infty} \big(W_k(\omega) X_k(\omega)\big)\big(W_l(\omega) X_l(\omega)\big)^* e^{j\omega\tau} d\omega \qquad (1.17)$$

Une fonction de pondération combinée, $\psi_{kl}(\omega)$ est définie comme :

$$\psi_{kl}(\omega) \equiv W_k(\omega) W_l^*(\omega) \qquad (1.18)$$

En remplaçant (1.18) par (1.17), l'inter-corrélation généralisée devient :

$$R_{kl}(\tau) \equiv \int_{-\infty}^{\infty} \psi_{kl}(\omega)\, X_k(\omega) X_l^*(\omega) e^{j\omega\tau} d\omega \qquad (1.19)$$

Le TDOA entre deux microphone k et l est le décalage temporel τ qui maximise l'intercorrélation généralisée $R_{kl}(\tau)$ dans la plage réelle limitée par la distance entre les microphones :

$$\hat{\tau}_{kl} = \underset{\tau}{\operatorname{argmax}} R_{kl}(\tau) \qquad (1.20)$$

1.5.3 Transformation de phase (PHAT)

Il a été démontré que la fonction de pondération de la phase (PHAT) est robuste dans un environnement réaliste. PHAT est défini comme suite :

$$\psi_{kl}(\omega) \equiv \frac{1}{\left| X_k(\omega) X_{l^*}(\omega) \right|} \qquad (1.21)$$

1.5.4 Transformation de phase-inter-corrélation généralisée (GCC-PHAT)

On applique de la fonction de pondération PHAT de l'équation (1.21) ci-dessus dans l'expression de GCC de l'équation (1.17), l'inter-corrélation généralisée en utilisant la transformation de phase (GCC-PHAT) pour les deux microphones k et l est défini comme suite :

$$R_{k,l}(\tau) \equiv \frac{1}{2\pi}\int_{-\infty}^{+\infty} \frac{1}{|X_k(\omega)X_{l^*}(\omega)|} X(\omega)X^*(\omega)e^{j\omega\tau}d\omega \qquad (1.22)$$

1.6 Conclusion

L'objectif de ce chapitre est de se familiariser avec les concepts liés à la localisation de la source sonore. Il donne des notions sur l'acoustique et un aperçu général sur la localisation d'une source sonore. Aussi dans ce chapitre, les principales approches de la localisation de la source sonore par les méthode TDOA et ILD sont détaillées avec la méthode généralisée de la localisation d'une source sonore (GCC-PHAT).

Chapitre 2 Simulation et validation expérimentale

2.1 Introduction

L'objectif de ce chapitre est de réaliser un système de localisation de source sonore par la méthode d'inter-corrélation généralisée.

Au cours de ce chapitre, on va tout d'abord faire une acquisition d'un signal sonore par un logiciel nommé GOLDWAV (voir annexe) à l'aide de deux microphones. Après cela on fait tous les prétraitements nécessaires sur le logiciel MATLAB (FFT, DSP, Filtre..), puis on implémente l'algorithme nécessaire qui détermine l'emplacement de la source sonore.

2.2 Description du système

Le schéma dans la figure (2.1), décrit d'une manière globale les principales fonctions du système de localisation de deux microphones.

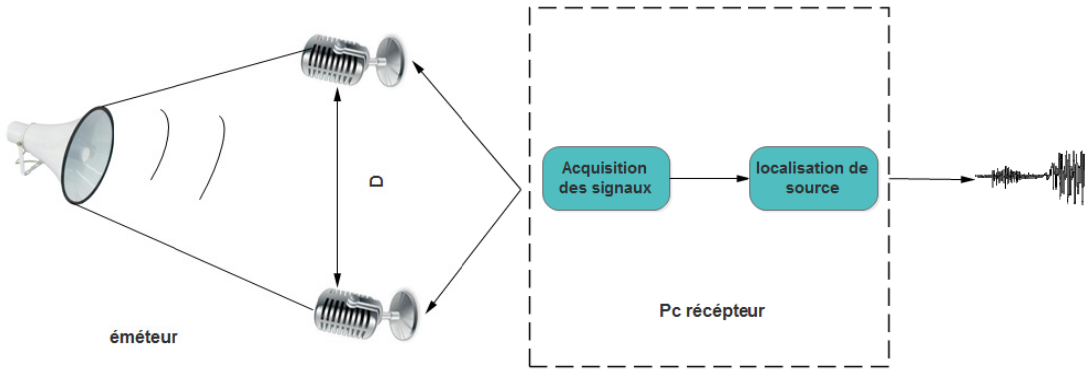


Figure 2.1 Description globale du système.

2.3 Acquisition d'un signal sonore

Tout son est en réalité composé d'un son fondamental et d'une série d'harmoniques.

Bien que nous ne percevions qu'un ensemble, chaque son possède une série d'harmoniques propres qui permet de le distinguer d'un autre. Afin de mettre en pratique le

principe du traitement de signal, nous avons connecté deux microphones et traité le signal acquis à l'aide du logiciel MATLAB.

Les étapes de traitement d'un signal sonore sont :

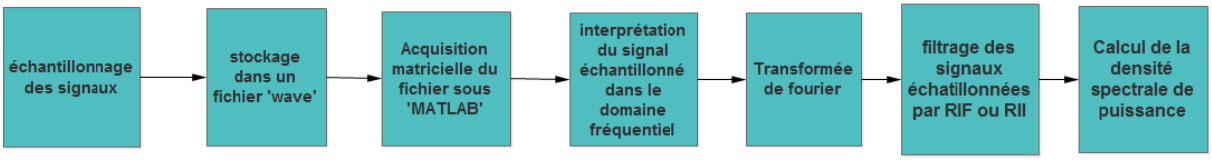


Figure 2.2 Les étapes de traitement d'un signal sonore.

2.3.1 Schéma de principe

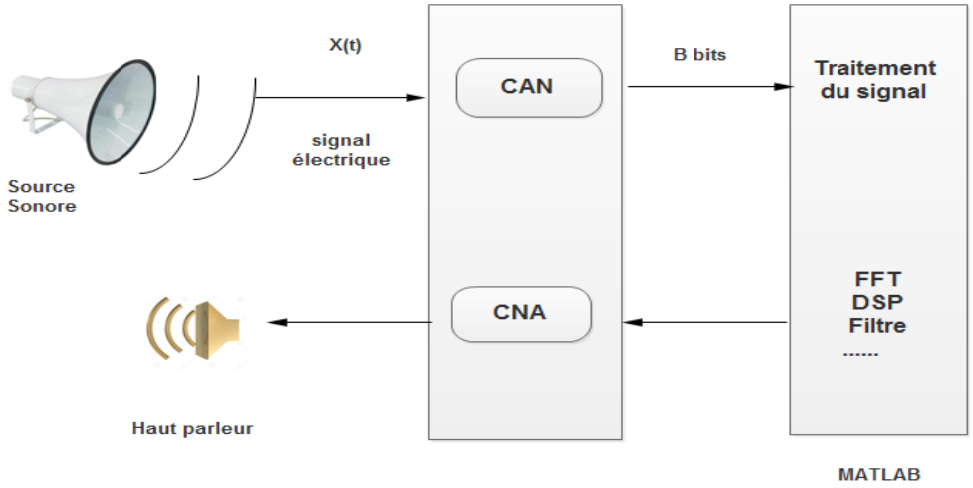


Figure 2.3 Schéma de principe de l'acquisition sonore.

2.3.2 Traitements des signaux

Les figures suivantes montrent les étapes citées ci-dessus du traitement de signal enregistré.

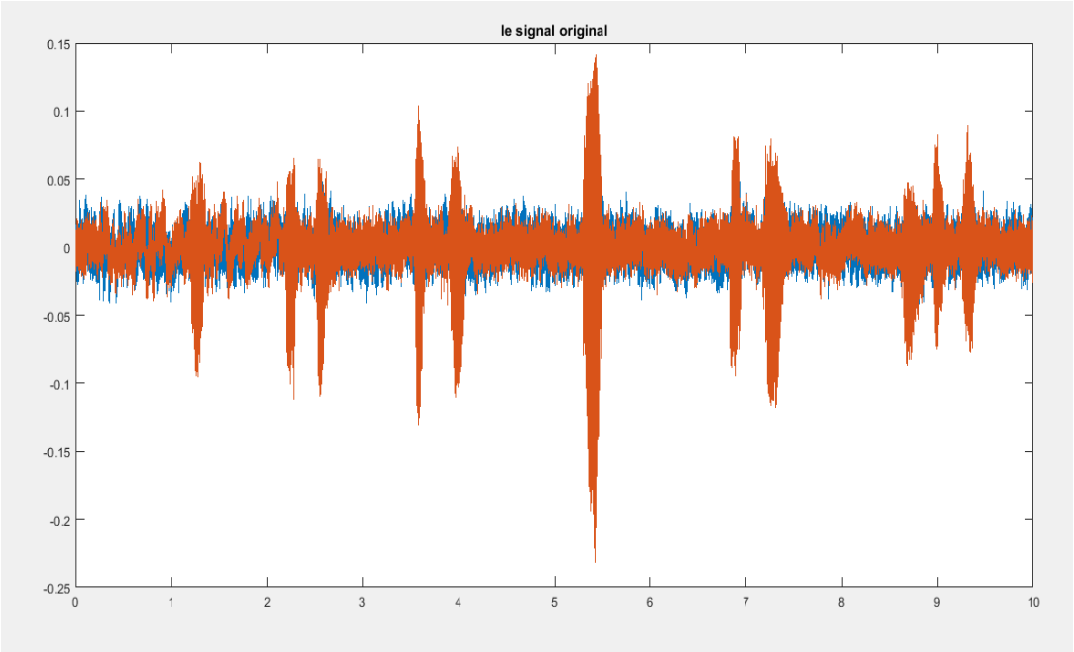


Figure 2.4 Le signal original enregistré.

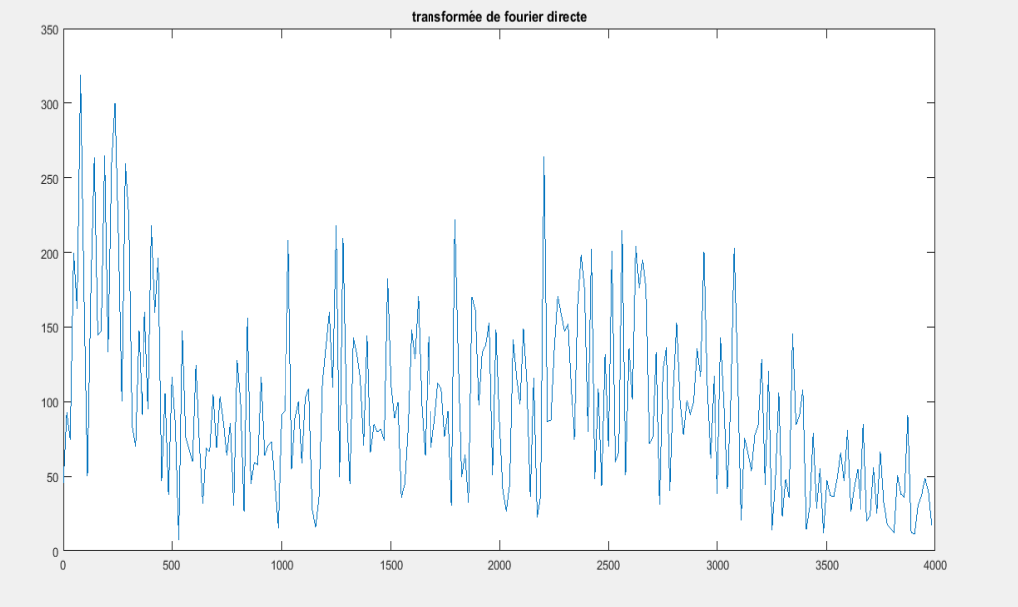


Figure 2.5 Transformée de Fourier du signal.

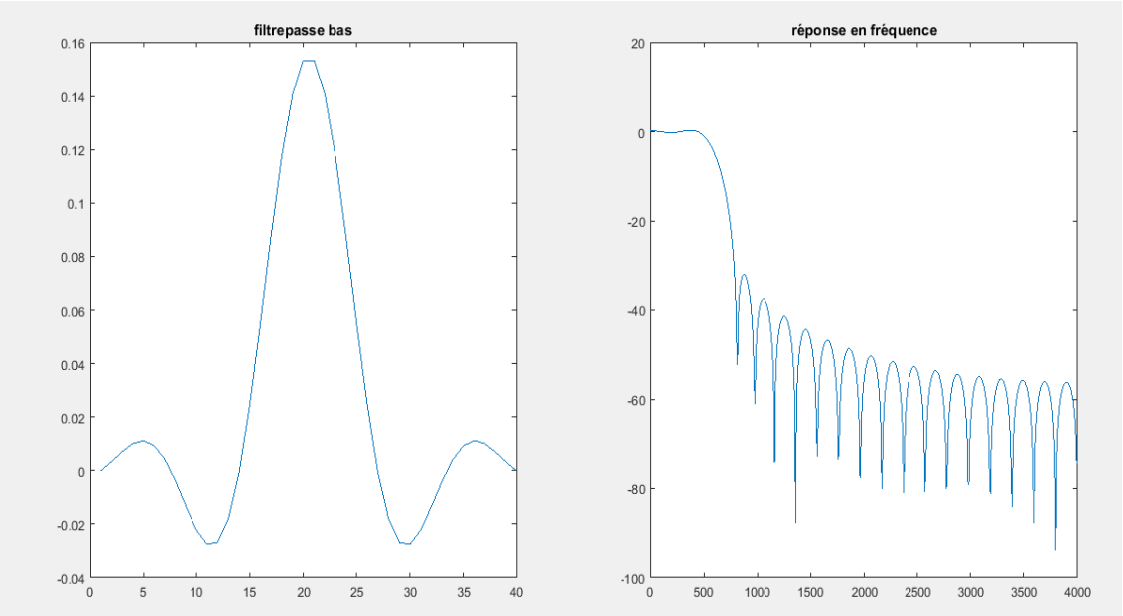


Figure 2.6 Application du filtre passe bas et tracé de la réponse de fréquence.

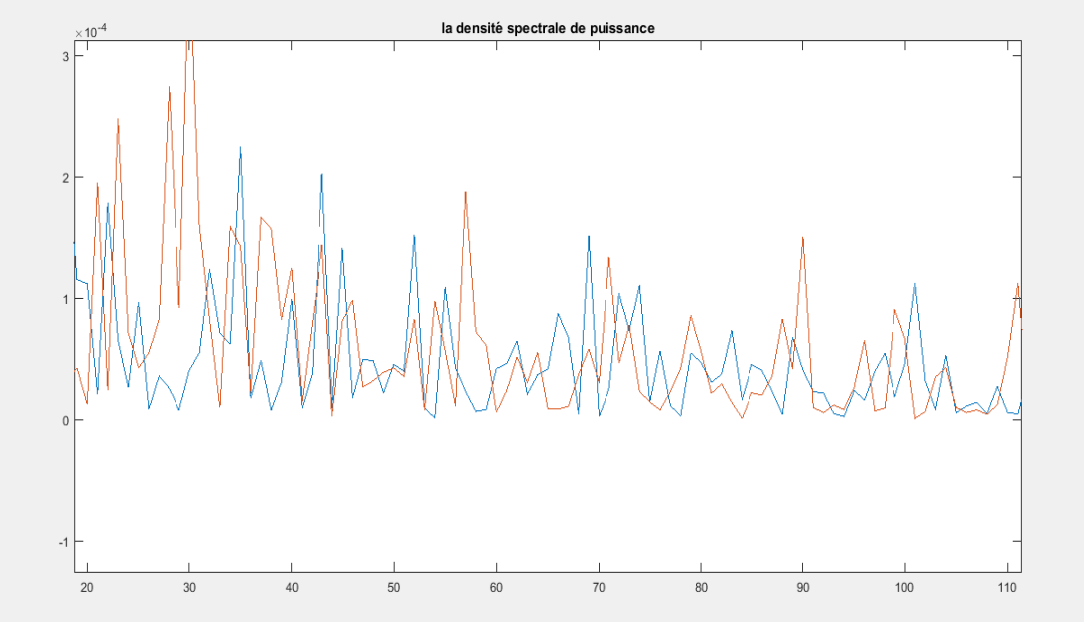


Figure 2.7 Densité spectrale de puissance.

2.4 Expérimentation

2.4.1 Partie 1

Les deux microphones sont liés à l'ordinateur par un câble jack stéréo ; D'abord on fait l'acquisition de notre signal sonore à l'aide du logiciel GOLD-WAVE (voir Annexe), ensuite les traitements avec MATLAB.

Figure 2.8 Exemple d'expérimentation de la méthode développée.

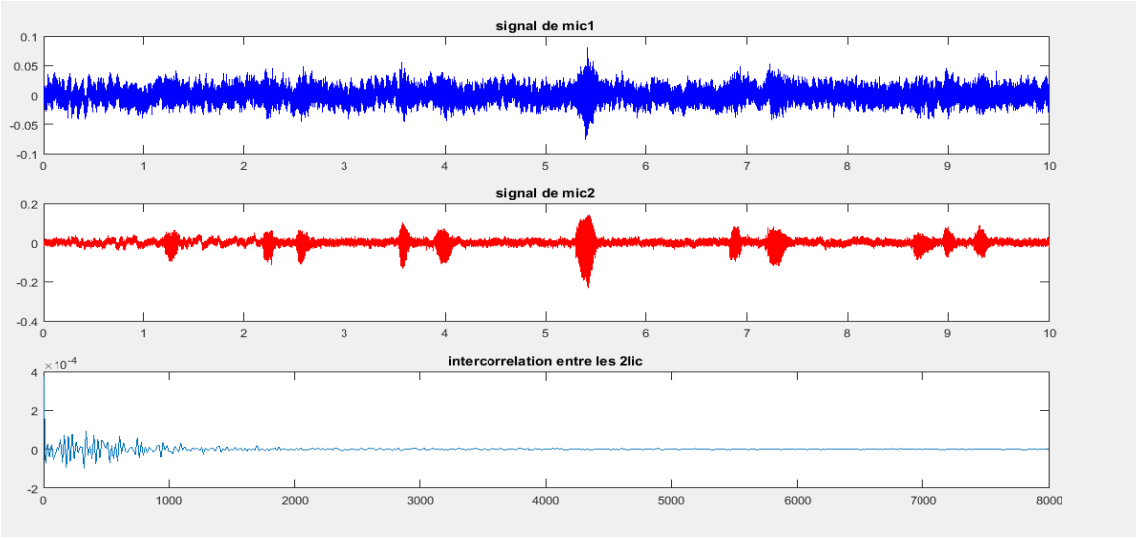


Figure 2.9 Inter-corrélation entre les deux signaux.

On varie la distance des les deux microphones pour évaluer l'influence de la distance (pour cinq positions théoriques de l'angle θ). Les résultats sont obtenus dans le tableau suivant (pour Fe=44100Hz).

Tableau 2.1 Résultats expérimentaux.

D(m)	Angle réel	Angle estimé	Erreur (degré)
0.5	0	0	0
	30	29.86	0.13
	45	46.19	1.19
	90	86.29	3.71
	130	127.44	2.56
1	0	0	0
	30	28.86	1.13
	45	41.12	3.88
	90	85.61	4.39
	130	131.27	1.27
2	0	0	0
	30	32.69	2.69
	45	44.42	0.58
	90	88.54	1.46
	130	133.22	3.88

Pour les cinq positions théoriques de l'angle θ (0°, 30°, 45°,90°,130°), on constate que la distance entre les microphones influence sur les résultats car plus la distance est proche, plus les résultats sont meilleurs et plus la source est proche des microphones, plus les résultats sont bon aussi.

Nous avons obtenu une erreur moyenne d'estimation de la direction d'arrivée de 5° ce qu'est considérer comme résultat acceptable.

2.4.2 Partie 2

Après l'acquisition du signal, on applique l'algorithme GCC-PHAT pour estimer la direction d'arrivée (DOA) d'une source sonore en utilisant deux microphones.

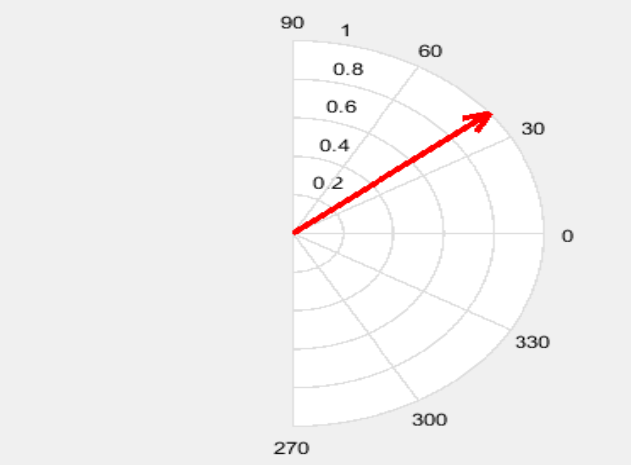


Figure 2.10 Application de l'algorithme GCC-PHAT.

2.5 Conclusion

Ce chapitre consiste à faire tous les traitements nécessaires pour nos sources sonores (acquisition, prétraitement..). Dans cette partie, on a étudié l'influence de la distance et de l'angle pour la localisation de la source sonore afin d'évaluer les performances de l'algorithme GCC-PHAT.

Chapitre 3 Acquisition et de traitement pratique des données – Partie matérielle (Hardware)

3.1 Introduction

Après avoir collecté les données nécessaires dans le chapitre précédant pour effectuer notre application, nous allons d'abord présenter dans ce chapitre la conception et la modélisation de notre système.

Pour commencer, nous allons présenter les différents composants du système de localisation de source sonore, et décrire l'implémentation de notre solution.

3.2 Arduino

3.2.1 Présentation générale de la carte Arduino

L'Arduino est une carte électronique basée autour d'un microcontrôleur et de composants minimum pour réaliser des fonctions plus ou moins évoluées à bas coût. Elle possède une interface USB pour la programmer.

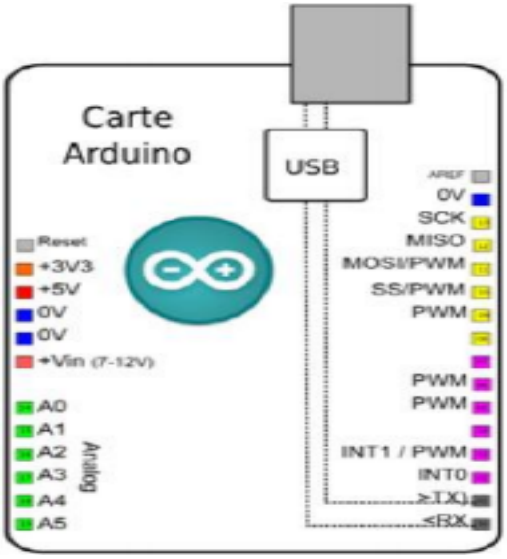


Figure 3.1 Présentation de la carte Arduino [6].

C'est une plateforme open-source qui est basée sur une simple carte à microcontrôleur (de la famille AVR), et un logiciel véritable , environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte à microcontrôleur.

L'Arduino peut être utilisé pour développer des applications matérielles industrielles légères ou des objets interactifs (création artistiques par exemple), et peut recevoir en entrées une très grande variété de capteurs. Il peut aussi contrôler une grande variété d'actionneurs (lumières, moteurs ou toutes autres sorties matériels). Les projets Arduino peuvent être autonomes, ou communiquer avec des logiciels sur un ordinateur (Flash, Processing ou MaxMSP). Les cartes électroniques peuvent être fabriquées manuellement ou bien être achetées pré-assemblées, le logiciel de développement open-source est téléchargeable gratuitement [6].

3.2.2 Synthèse des caractéristiques

Tableau 3.1 Caractéristiques de la carte Arduino [7].

Microcontrôleur	Atmega328
Tension de fonctionnement	5 v
Tension d'alimentation (recommandée)	7-12 v
Tension d'alimentation (Limites)	6-12 v
Broches E/S numériques	14(dont 6 disposent d'une sortie PWM)
Broches d'entrées analogiques	6 (utilisables en branches E/S numériques)
Intensité maxi disponible pour la sortie 3.3V	50 mA
Intensité maxi disponible pour la sortie 5V	Fonction de l'alimentation utilisée -500 mA maxi si port USB utilisé seul.
Mémoire Programme Flash	32 KB (Atmega 328) dont 0.5 MB sont utilisés par le bootloader
Mémoire SRAM (mémoire volatile)	2KB (Atmega 328)
Mémoire EEPROM (mémoire non volatile)	1KB (Atmega 328)
Vitesse d'horloge	16 MHz

3.2.3 Utilisations

On peut utiliser l'Arduino pour récupérer les information des capteurs, pour contrôler des moteurs, pour communiquer avec un ordinateur, avec téléphone portable, pour envoyer et lire des messages sur internet ,pour éteindre des appareils électriques ,pour piloter un robot mobile, pour servir de cerveau a un humanoïde ou tout simplement pour apprendre l'électronique, et encor bien d'autres usages.

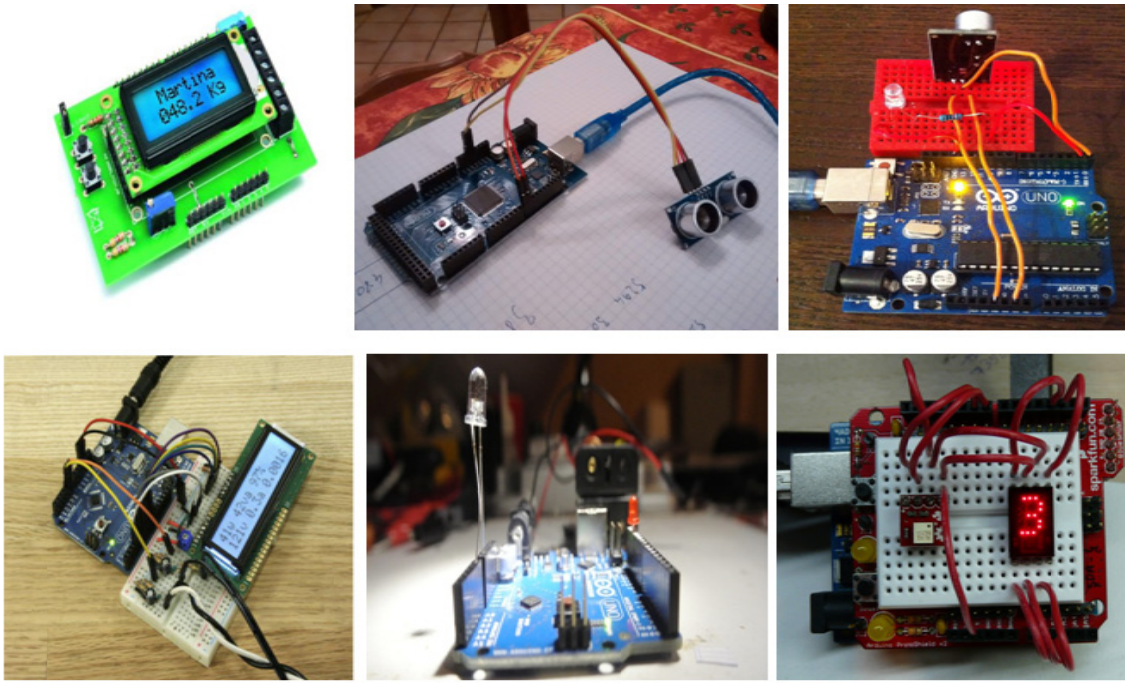


Figure 3.2 Différentes utilisations de carte Arduino.

3.2.4 Présentation de la carte Arduino Uno

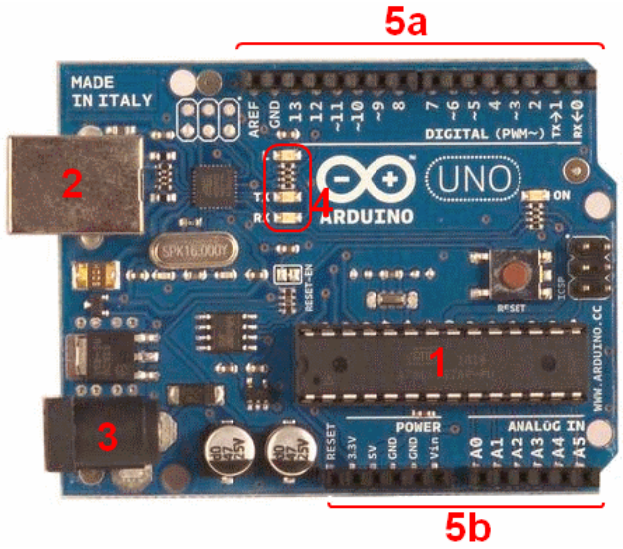


Figure 3.3 Présentation de la carte Arduino Uno [7].

3.2.4.1 La partie logicielle

Le logiciel de programmation des modules Arduino est une application Java, libre et multiplateformes, servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (RS232, Bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino, et de compiler les programmes en ligne de commande.

Le langage de programmation utilisé est le C++, compilé avec g++, et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino, à toute personne maitrisant le C ou le C++.

3.2.4.2 La partie matérielle

Un module Arduino est généralement construit autour d'un microcontrôleur ATMEL AVR(Atmega328 ou Atmega2560 pour les versions récentes, Atmega168 ou Atmega8 pour les plus anciennes), et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits. Chaque module possède au moins un régulateur linéaire 5V et un oscillateur à quartz 16 MHz (ou un résonateur céramique dans certains modèles).

Le microcontrôleur est préprogrammé avec un boot loader de façon à ce qu'un programmateur dédié ne soit pas nécessaire.

Les modules sont programmés au travers une connexion série RS-232, mais les connexions permettant cette programmation diffèrent selon les modèles. Les premiers Arduino possédaient un port série, puis l'USB est apparu sur les modèles Diecimila, tandis que certains modules destinés à une utilisation portable se sont affranchis de l'interface de programmation, relocalisée sur un module USB série dédié (sous forme de carte ou de câble).

L'Arduino utilise la plupart des entrées/sorties du microcontrôleur pour l'interfaçage avec les autres circuits. Le modèle Diecimila par exemple, possède 14 entrées/sorties numériques, dont 6 peuvent produire des signaux PWM, et 6 entrées analogiques. Les connexions sont établies au travers de connecteurs femelle HE14 situés sur le dessus de la carte, les modules d'extension venant s'empiler sur l'Arduino. Plusieurs sortes d'extensions sont disponibles dans le commerce [6],[7].

3.2.4.3 Constitution des éléments important de la carte

a) Le Micro- Contrôleur

Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte pour le programme, mémoire vive pour les données), unités périphériques et interfaces d'entrées-sorties.

Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique (quelques milliwatts en fonctionnement, quelques nano watts en veille), une vitesse de fonctionnement plus faible (quelques mégahertz à quelques centaines de mégahertz) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

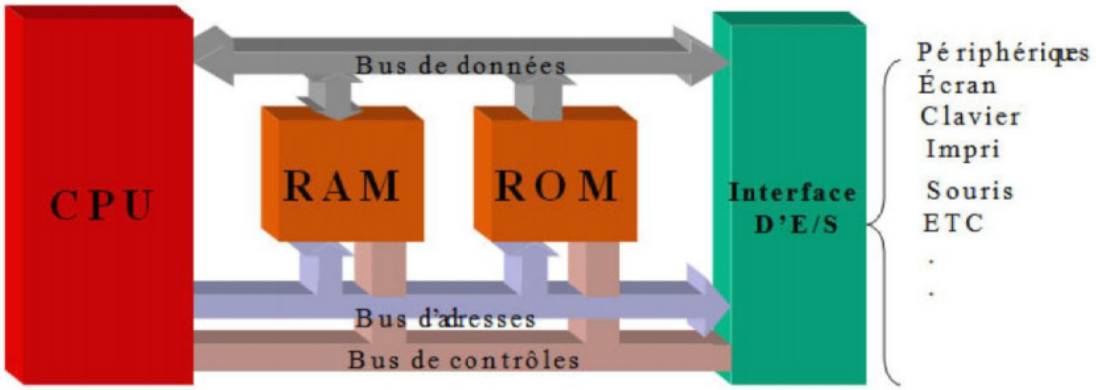


Figure 3.4 Schéma simplifié du contenu type d'un microcontrôleur [6].

b) Alimentation

Pour fonctionner, la carte Arduino Uno a besoin d'une alimentation, elle peut être alimentée par l'USB ou par une alimentation externe, La source est sélectionnée automatiquement. La tension d'alimentation extérieure (hors USB) peut venir soit d'un adaptateur AC-DC ou de piles.

L'adaptateur peut être connecté grâce à un 'jack' de 2.1mm positif au centre. Le raccordement vers un bloc de piles peut utiliser les bornes Gnd et Vin du connecteur d'alimentation (POWER).

La carte peut fonctionner à l'aide d'une tension extérieure de 7 à 12 volts. Les broches (pins) d'alimentation sont les suivantes :

VIN : La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V

régulée). Vous pouvez alimenter la carte à l'aide de cette broche, ou, si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.

5V : La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres

composants de la carte (pour info : les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite (tension régulée) obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino. Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.

3.3V : Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de votre ordinateur et le port série de l'Atmega) de la carte est disponible : ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V). L'intensité maximale disponible sur cette broche est de 50mAGND. Broche de masse (ou 0V).

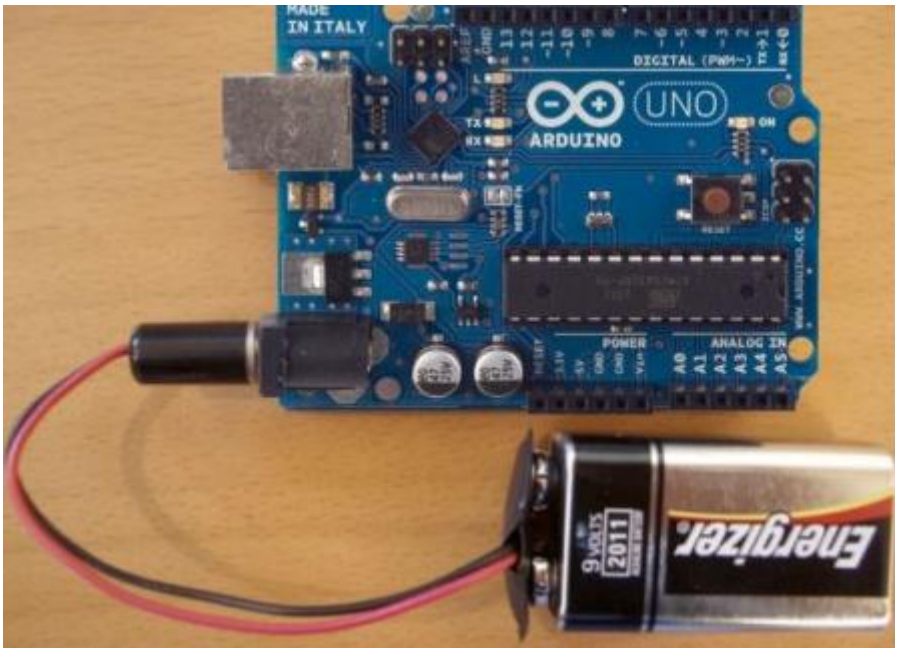


Figure 3.5 Câble d'alimentation Arduino pour pile.

c) Mémoire

L' Atmega328 a 32 KB de mémoire (dont 0.5 KB pour le bootloader). Il a également 2 KB de SRAM et 1 KB de mémoire non volatile EPROM (qui peut être écrite et lue grâce à la librairie 'EEPROM').

d) Visualisation

Les trois « points blancs » entourés en rouge (**4**) sont en fait des LED dont la taille est de l'ordre du millimètre. Ces LED servent à deux choses :

- Celle tout en haut du cadre : elle est connectée à une broche du microcontrôleur et va servir pour tester le matériel. *Nota* : Quand on branche la carte au PC, elle clignote quelques secondes.
- Les deux LED du bas du cadre : servent à visualiser l'activité sur la voie série (une pour l'émission et l'autre pour la réception). Le téléchargement du programme dans le micro-contrôleur se faisant par cette voie, on peut les voir clignoter lors du chargement.

e) La connectique

La carte Arduino ne possédant pas de composants qui peuvent être utilisés pour un programme, mis a par la LED connectée à la broche 13 du microcontrôleur, il est nécessaire de les rajouter. Mais pour ce faire, il faut les connecter à la carte. C'est là qu'intervient la connectique de la carte (en **5a** et **5b**). Par exemple, on veut connecter une LED sur une sortie du microcontrôleur. Il suffit juste le la connecter, avec une résistance en série, à la carte, sur les fiches de connections de la carte. Cette connectique est importante et a un brochage qu'il faudra respecter. Nous le verrons quand nous apprendrons à faire notre premier programme. C'est avec cette connectique que la carte est « extensible », car l'on peut y brancher tous types de montages et modules ! Par exemple, la carte Arduino Uno peut être étendue avec des shields, comme le « **Shield Ethernet** » qui permet de connecter cette dernière à internet.

3.3 Servomoteur

3.3.1 Introduction

Les servomoteurs sont très utilisés en robotique car ils sont à la base de nombreux mécanisme d'actionneurs, comme des pinces par exemple. Par contre, ils consomment souvent beaucoup de courant et demandent une tension souvent très bien régulée ce qui nécessite des dispositions particulières.

3.3.2 Qu'est-ce qu'un servomoteur ?

Un servomoteur est un dispositif typiquement utilisé en modélisme, par exemple pour diriger une voiture télécommandée ou contrôler à distance une caméra vidéo.

Figure 3.6 Le servomoteur.

Il comprend :

• Un moteur électrique de petite taille.

• Un réducteur en sortie de ce moteur pour avoir moins de vitesse et plus de couple.

• Un capteur : un potentiomètre qui produit une tension variable en fonction de l'angle de l'axe de sortie.

• Un asservissement électronique pour contrôler la position de cet axe de sortie.

3.3.3 Fonctionnement d'un servomoteur

L'utilisateur envoie à chaque instant un signal modulé représentatif d'un angle. Le servomoteur tourne vers cet angle et s'y maintient. La duré de l'impulsion se varie entre 0.9ms qui permet à l'axe de tourner jusqu'à la position extrême gauche et 2.1ms qui lui permet d'aller jusqu'à l'extrême droite. Et avec 1.5ms pour revenir au centre. Pour varier la vitesse d'un servomoteur en jouant sur la durée de l'impulsion. Plus la durée de l'impulsion est proche de 1.5ms plus la vitesse de rotation est faible [8].

3.3.4 Le potentiomètre

Le potentiomètre sert à l'asservissement de position. L'axe du potentiomètre est solidaire de l'axe de sortie qui lui permet de tourner en même temps que l'axe de sortie. La résistance à la borne du potentiomètre varie en fonction de la position de l'axe. La résistance du potentiomètre est en rapport avec la durée du signal. Par exemple le servomoteur reçoit un ordre pour se déplacer à la position médiane (1.5 ms). Le moteur va tourner jusqu'à ce que

la résistance entre le curseur du potentiomètre et les deux extrémités soient identiques et s'il reçoit une impulsion de 2.1ms il va tourner jusqu'à ce que la résistance entre le curseur et l'extrémité du potentiomètre soit maximale. L'électronique faire la relation entre la durée de l'impulsion et la résistance aux bornes du potentiomètre.

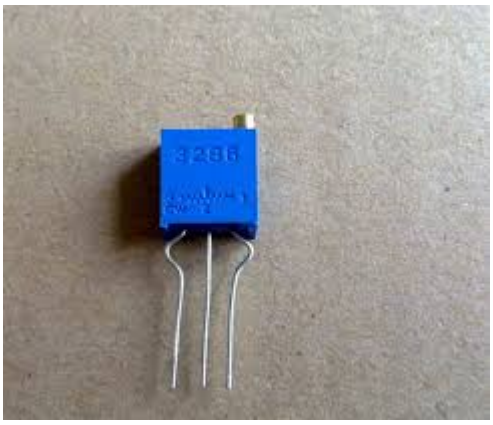


Figure 3.7 Le potentiomètre.

3.3.5 La commande du servomoteur

Un connecteur à trois fils permet de commander un servomoteur. Bien sûr, il y a la masse et le +5V (à peu près). Ces deux fils sont en général noirs et rouges, mais ils sont marrons et rouge pour les modèles Graupner. Le 3ème fil est le fil de commande. Il est normalement blanc, mais jaune dans le cas des Graupner. On pourrait dire sommairement qu'il faut fournir un signal PWM en tant que signal de commande, mais ce n'est pas tout à fait exact :

• Le PWM (Pulse Width Modulation : Modulation en largeur d'impulsion) est un signal dont le taux de modulation varie de 0 à 100% Tm=Ton/ T période

• Pour un servomoteur on parle de Pulse-Code Modulated Signal (Signal modulé en code d'impulsion) où seule l'impulsion compte, pas la fréquence (en théorie). En clair, il faut fournir au servomoteur une impulsion à 1 (suivie d'un retour à 0). Le servomoteur va prendre en compte la largeur temporelle de cette impulsion qu'il va convertir de façon linéaire en un angle. La durée de retour à 0 de l'impulsion n'est pas critique, en pratique il faut éviter de dépasser 20ms entre deux fronts montants. A noter aussi que ce système présente un avantage par rapport au PWM : une absence de signal (toujours à 1, ou à 0) laisse le servomoteur en "roue libre" comme si il n'était pas alimenté. Le microcontrôleur programmé est capable de fournir directement ce signal de commande sur une sortie [8].

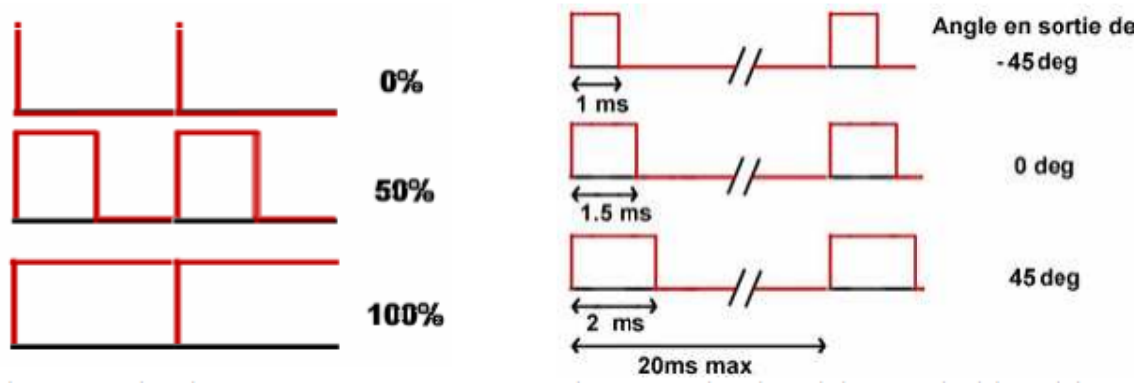


Figure 3.8 PWM (gauche) et Signal modulé en code d'impulsion (droite).

3.3.6 Caractéristique du servomoteur utilisé dans notre projet

le SM-S4303R est un servo de taille standard qui a été construit par SpringRC spécifiquement pour une rotation continue a 6 V, il a une vitesse de rotation maximale de 54 tr / min (sans charge) et peut produire un couple de couple de plus de 71 kg (5,1 kg). Il comporte deux roulements à billes sur l'arbre de sortie pour un frottement réduit, et il offre un accès facile au potentiomètre de réglage du point de repos. Le servo peut être commandé en utilisant une connexion directe à une seule ligne d'E / S microcontrôleur sans électronique supplémentaire, ce qui en fait un excellent actionneur pour les projets de robotique débutant.

Le servo de rotation continue SM-S4303R convertit les impulsions de position servo standard en vitesse de rotation continue. Le point de repos par défaut est de 1,5 ms, mais cela peut être réglé à l'aide d'un petit tournevis à fente pour tourner le positionneur du point central.

Figure 3.9 Le servomoteur SM-S4303R.

Les largeurs d'impulsion au-dessus du point de repos entraînent une rotation dans le sens inverse des aiguilles d'une montre, avec une augmentation de la vitesse à mesure que la largeur d'impulsion augmente. Les largeurs d'impulsion en dessous du point de repos entraînent une rotation dans le sens horaire, avec une augmentation de la vitesse à mesure que la largeur de l'impulsion diminue.

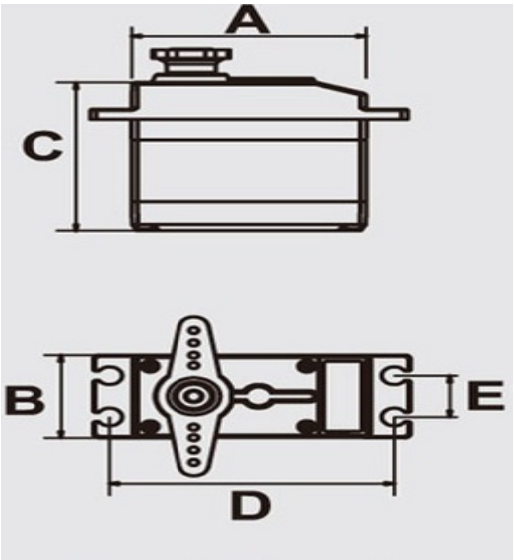


Figure 3.10 Dimension du servomoteur.

où :

- **A : 41.3**
- **B : 20.7**
- **C : 40.2**
- **D : 50.3**
- **E : 10.0**

3.4 Les microphones

3.4.1 Introduction

Un des principaux défis en robotique est de développer à l'instar de l'homme des capteur qui recevraient sans discrimination les informations en provenance de son Environnement. En effet, on considère souvent la perception comme un phénomène Purement passif. Dans le monde de la robotique nous serions des robots mobiles munis de microphones (les oreilles), de caméras (les yeux), de nombreux capteurs de pression et de température (la peau), de capteurs chimiques (le goût et l'odorat) et de capteurs cinétiques (les labyrinthes ou vestibules : organes de l'oreille renseignant l'individu sur sa position

dans l'espace et les mouvements qu'il y effectue). Dans notre cas, nous désirons développer la perception auditive. Donc, avant de commencer l'étude du capteur pour la localisation 2D, il m'a paru important de rappeler quelques définitions générales.

3.4.2 Généralités sur les signaux sonores

La perception naturelle de l'espace sonore passe par la localisation des sources qui nous Entourent. Localiser une source, c'est déterminer son azimut (sa direction), sa profondeur (sa distance) et son élévation (sa hauteur).

Un signal est un phénomène physique transportant une information. Nous nous intéressons ici aux signaux sonores qui peuvent être, par exemple, un 'clap', un sifflement, ou tout simplement un bruit. On sait qu'un signal sonore est une vibration de l'air se transmettant par des variations de pressions plus ou moins rapides et plus ou moins importantes, la rapidité étant caractérisée par la fréquence tandis que l'importance est caractérisée par l'amplitude de la vibration, la puissance du signal est proportionnelle au carré de son amplitude.

Les sons peuvent se propager dans tous les milieux matériels, mais, pour des raisons techniques, on ne peut les modifier qu'avec des appareils électroniques de sorte que, pour un son déterminé, il faut le transformer en signal électromagnétique pour lui faire subir le traitement que l'on désire. Tout comme la lumière, le son est constitué par un spectre. L'oreille est équipée pour l'analyse de ce spectre: elle est constituée de milliers de cellules spécialisées dans une gamme très sélective de fréquences. La perception est un procédé d'analyse du spectre sonore par décomposition en sons élémentaires. Notre oreille est sensible aux fréquences entre 20 Hz et 20kHz, avec un maximum de sensibilité aux alentours de 3 kHz. C'est aux alentours de cette fréquence que sont situés les sons produits par la voix.

La fréquence d'un signal sonore caractérise la perception de hauteur de ce son: plus la fréquence est grande, plus le son perçu est aigu et inversement, plus la fréquence est basse, plus le son perçu est grave.

L'amplitude sonore correspond à l'intensité, elle est mesurée en décibels acoustiques (dBa). 0 db correspond à notre seuil d'audition. Le carré de l'amplitude du signal est proportionnel à la sensation d'intensité sonore.

3.4.3 Etat de l'art des microphones

Le premier axe de notre recherche consistait à étudier les différentes technologies pour la mesure d'un son. Il existe trois technologies majeures:

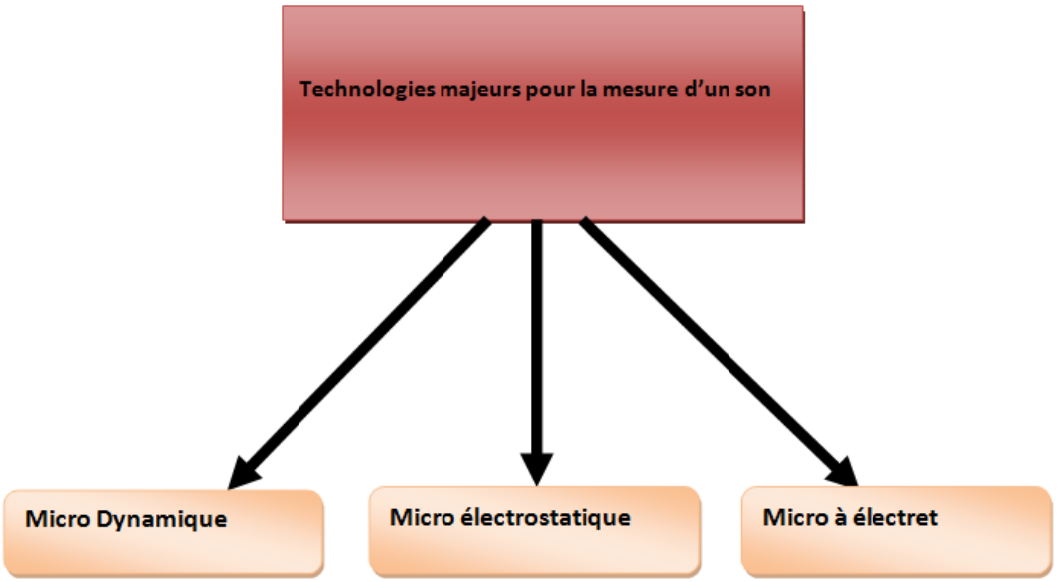


Figure 3.11 Représentation de technologies majeures pour la mesure d'un son.

3.4.4 Microphone dynamique

Une bobine mobile est solidaire de la membrane. Cette dernière, sous l'effet des variations de pression acoustique fait osciller la bobine dans un champ magnétique annulaire produit par un aimant permanent.

La bobine coupe les lignes de force du champ magnétique. En raison de ces oscillations périodiques de la membrane, et donc de la bobine dans le champ, ily a induction d'un courant électrique dans la bobine mobile - courant qui peut être rendu utilisable par une amplification appropriée.

Ne nécessite pas de source d'alimentation (bobine mobile).

- Très robuste en général.
- Pratiquement insaturable quel que soit le niveau du son à enregistrer.
- Ne craint pas l'humidité.
- D'un prix abordable dans les entrées de gamme.
- Présence de qualités et de défauts selon la directivité du capteur.
- Sensibilité généralement faible (1 à 4 mV/Pa).
- Peu ou pas de choix en microphones canon.

- Particulièrement adaptés aux prises de son de batterie
- Sa faible sensibilité le condamne à une utilisation de proximité ou aux niveaux élevés.

Figure 3.12 Exemples de microphones dynamiques.

Dans tout autre cas, elle le condamne à l'utilisation d'un préamplificateur spécialisé et à faible bruit sans quoi apparaissent très rapidement des problèmes de souffle inhérents aux préamplificateurs de microphones intégrés aux matériels d'enregistrement, souvent peu performant à ce niveau [5].

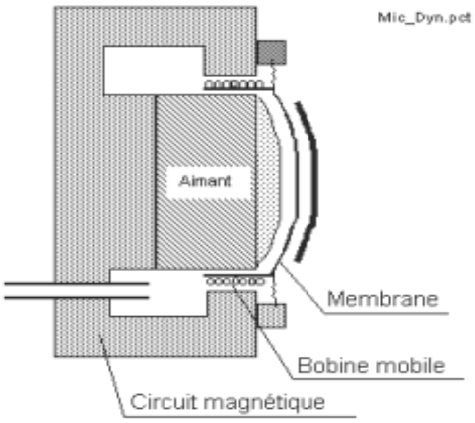
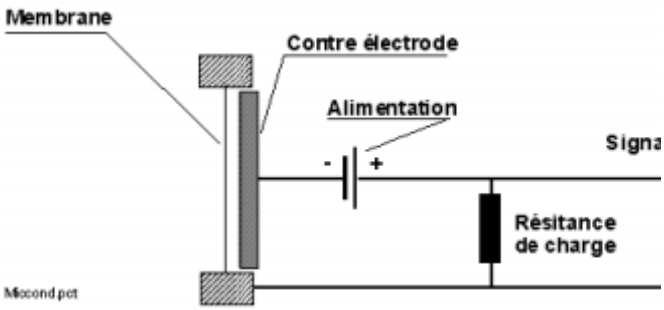


Figure 3.13 Fonctionnement d'un microphone dynamique [5].

3.4.5 Micro électrostatique

C'est en fait un condensateur dont une des armatures fait office de membrane Une tension de polarisation élevée est appliquée aux deux armatures. Toute différence ou variation de pression sur la membrane fait osciller celle-ci et fait donc varier la distance séparant la membrane de l'armature, et fait donc varier la capacité du condensateur.



3.14 Fonctionnement d'un micro électrostatique [5].

On obtient ainsi des variations de charge, donc production d'un courant alternatif permettant de recueillir une différence de potentiel variable aux bornes d'une résistance.

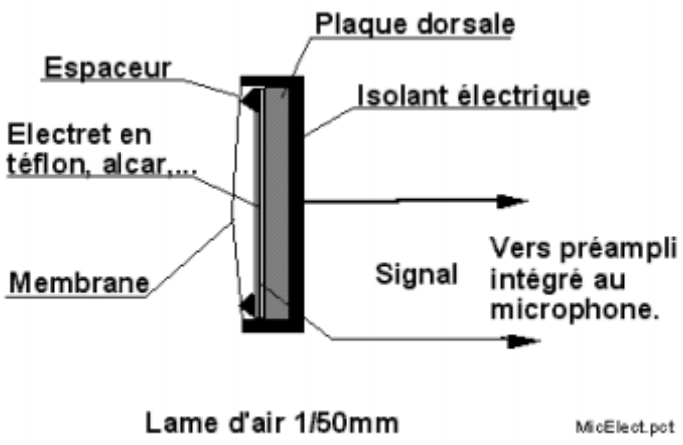
- Grande fidélité de reproduction.
- Sensibilité souvent importante permettant l'enregistrement de sons ténus
- Dynamique importante.
- Courbe de réponse généralement étendue.
- Très bon rapport Signal/Bruit.
- Relativement peu sensible aux bruits de contacts.
- Réservé bien souvent (à cause de son prix) à un usage professionnel.
- Craint l'humidité.
- Exige une source d'alimentation extérieure (12 à 48 V en général).
- Très utilisé en studio de prise de son ou les exigences acoustiques sont maximales.
- Disponible dans toutes les marques, dans toutes les directivités [5].

3.4.6 Micro à électret

Même principe que pour le microphone électrostatique, mais au lieu d'une tension de polarisation extérieure, cette polarisation est permanente. De même que l'on peut stocker du magnétisme dans certains corps magnétiques, on peut stocker une tension électrique dans certains corps électriques par un procédé spécial de polarisation. Ainsi, la pile présente dans ces microphones ne sert pas à la polarisation de la membrane, mais à l'alimentation d'un amplificateur/adaptateur d'impédance intégré au corps du micro [5].

- Souvent abordable quant au prix, surtout en entrée de gamme.
- Miniaturisation poussée (micros cravate de la taille d'une allumette).
- Peu sensible aux bruits de contact.
- Disponible dans toutes les directivités.
- Nécessite une source d'alimentation interne ou externe, entre 1,5 et 9V.

- Bruit de fond souvent important dans les entrées de gamme.
- Craint l'humidité et la chaleur en général.
- Sensibilité honorable (5 à 50 mV/Pa en moyenne).



3.15 Fonctionnement de micro à électret [5].

3.4.7 Directivité des microphones

La directivité d'un microphone définit sa sensibilité en fonction de la provenance d'un son. On présente ici les différentes directivités quelque soit le type de microphones :

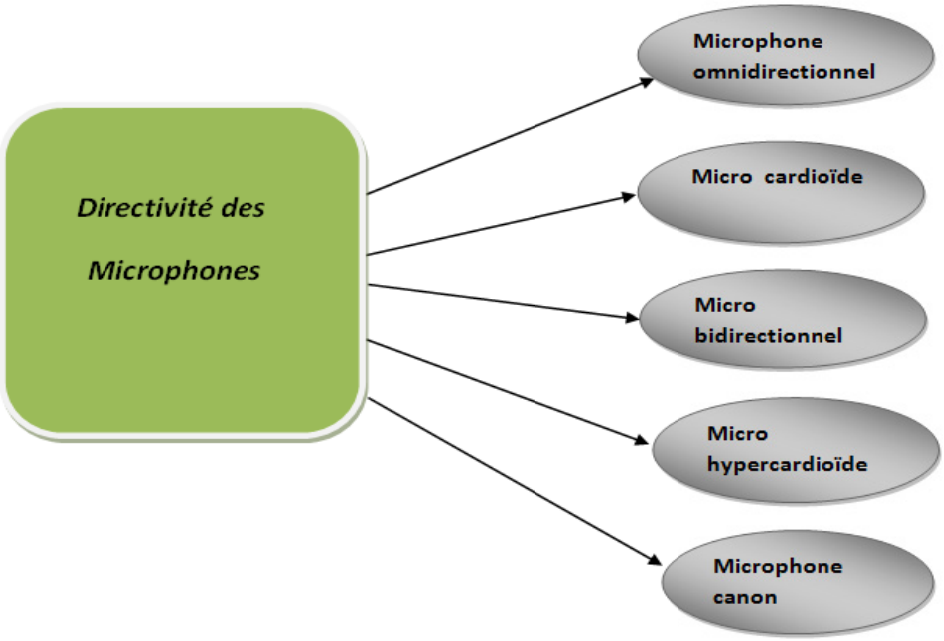
- **Le microphone omnidirectionnel** ne privilégie aucune provenance. Il est donc particulièrement adapté pour enregistrer des sons d'ambiance.

- **Le micro cardioïde** est définit par sa directivité vers l'avant, mais surtout par le fait qu'il capte peu les sons provenant de l'arrière. C'est pourquoi il est le microphone privilégié des chanteurs sur scène, car le risque de feedback avec un retour de scène est limité.

- **Le micro bidirectionnel** est sensible aux sons provenant de l'avant et de l'arrière. Il est donc indiqué pour les duos, ou pour être placé entre deux éléments.

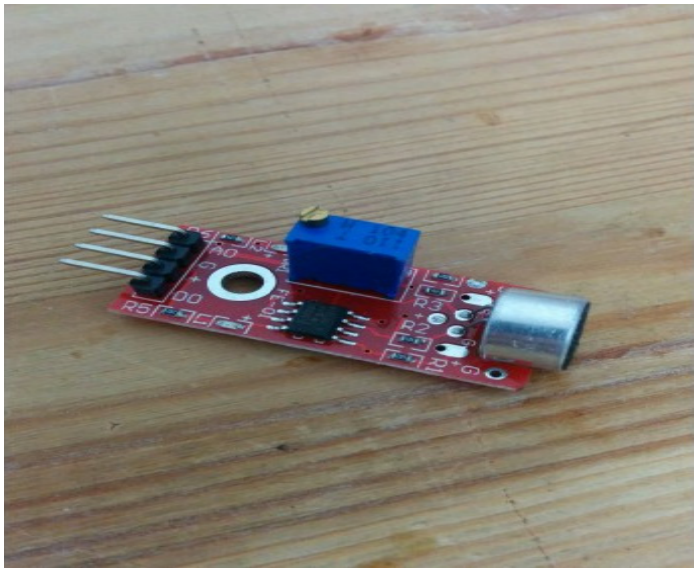
- **Le micro hypercardioïde** ressemble à un microphone cardioïde mais dont les angles morts diffèrent. De ce fait, il est adapté aux chanteurs utilisant deux retours de scène.

- **Le microphone canon** possède une forte directivité vers l'avant. Il est donc indiqué pour capter les sons d'une source éloignée en limitant les bruits parasites. Idéal pour capter une source éloignée (documentaires animaliers...).



3.16 Schéma de Directivité des microphones.

3.4.8 Choix du capteur



3.17 Module microphone haute sensibilité KY-037.

Pour le capteur, notre choix s'est porté sur une cellule microphonique, à électret, de directivité omnidirectionnelle. En effet, ce type de microphone capture un son proche de l'écoute humaine, sur 360°, ce qui correspond bien au cahier des charges fixé par la robotique mobile.

3.4.8.1 Description

Ceci est un module de détection son à microphone électret ultra sensible. Il est très pratique car il peut être fixé (trou 3mm). On peut l'utilisé via les sorties A0 ou D0.

A0: sortie analogique simple

D0 : sortie n'émettant un signal que lorsque le son atteint un certain seuil (threshold)

Il est équipé d'un chip amplificateur LM398 et d'une résistance réglable pour le seuil de sensibilité.

3.4.8.2 Caractéristique technique

Tableau 3.2 Caractéristiques techniques de microphone KY-037.

Signal de sortie	**Analogique**
Tension de fonctionnement	**5VDC**
Fixation	**Trou 3mm pour fixation**
Led	**Led d'indication de fonctionnement**
Dimension	**38.5 x 15.5 x 12.5mm**

3.4.8.3 Fonctionnement du capteur

Le capteur comporte 3 composants principaux sur sa carte de circuit imprimé. Tout d'abord, l'unité de détection à l'avant du module qui mesure la zone physiquement et envoie un signal analogique à la deuxième unité, l'amplificateur. L'amplificateur amplifie le signal, en fonction de la valeur résistante du potentiomètre, et envoie le signal à la sortie analogique du module.

Le troisième composant est un comparateur qui éteint la sortie numérique et la LED si le signal tombe sous une valeur spécifique.

Vous pouvez contrôler la sensibilité en ajustant le potentiomètre.

Ce capteur ne présente pas de valeurs absolues (comme la température exacte en ° C ou la force du champ magnétique dans mT).

Il s'agit d'une mesure relative: vous définissez une valeur extrême dans une situation d'environnement normale donnée et un signal sera envoyé si la mesure dépasse la valeur extrême.

Il est parfait pour le contrôle de la température (KY-028), le détecteur de proximité (KY-024, KY-025, KY-036), la détection des alarmes (KY-037, KY-038) ou le codeur rotatif (KY-026).

3.5 Conclusion

Dans ce chapitre on a élaboré d'une manière générale les différents composants et leurs caractéristiques qu'on a choisis pour la mise en œuvre de notre projet.

Tout d'abord on a décrit avec précision la carte Arduino, qui est indispensable pour la réalisation. On a aussi défini le servomoteur, son fonctionnement et ses caractéristiques.

Chapitre 4 Conception et mise en œuvre – Partie logicielle (Software)

4.1 Introduction

Dans ce chapitre nous allons voir d'une manière générale les différents logiciels et langages qu'on a utilisé pour la conception et la réalisation de notre projet.

Tout d'abord on va détailler le logiciel de la carte Arduino et décrire son interface, puis expliqué les approches et utilisations du logiciel et la structure d'un programme. Ensuite on va décrire le langage Java et les outils de programmations (NetBeans, MySQL) et aussi une mise en œuvre. A la fin, nous allons effectuer un schéma de principe de la réalisation et l'organigramme.

4.1.1 Programmation

La carte Arduino Uno peut être programmée directement avec « l'Arduino software » l'Atmega328 sur la carte Uno, qui est déjà pré programmée avec un 'bootloader' qui permet de charger le code d'une nouvelle application sans utiliser un programmateur hardware externe. La carte Arduino Uno communique avec un ordinateur en utilisant le protocole STK500 d'ATMEL. Mais, il est toujours possible de programmer le contrôleur de la carte en utilisant le port ICSP (In-Circuit Serial Programming) (le code source du firmware du contrôleur auxiliaire Atmega8U2 est disponible).

4.1.2 Reset automatique par software

Il est possible d'effectuer un reset via le logiciel Arduino. En effet, la ligne DTR sur l'Atmega8U2 est connectée à la ligne Reset de l'Atmega328 à travers une capacité. Lorsque cette ligne est amenée à l'état logique 0, un signal Reset est envoyé au contrôleur [9].

4.1.3 Protection de surintensité USB

La carte Arduino Uno possède une protection par fusible pour le port USB si un courant de plus de 500mA est demandé. La déconnexion durera tant que la source de consommation excessive n'aura pas cessé.

4.1.4 Dimensions

La longueur et largeur maximales du PCB sont de 6,9 et 5,3 cm respectivement [7].

4.1.5 Schéma structurel de l'Arduino

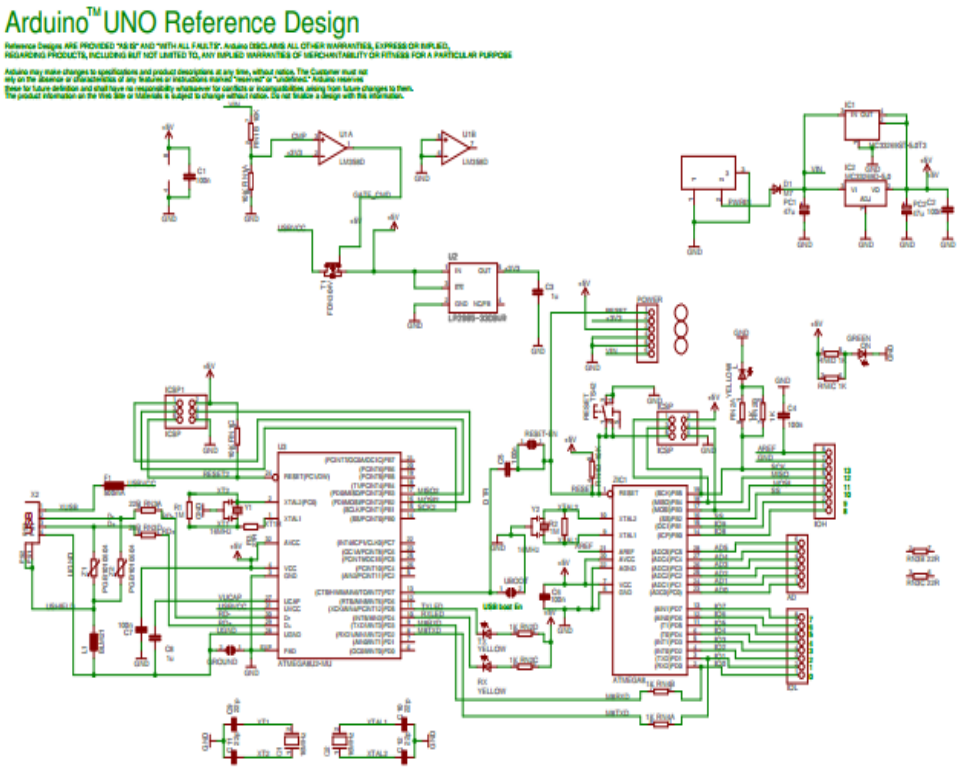


Figure 4.1 Schéma fonctionnel de l'Arduino [7].

4.2 Présentation de l'Espace de développement Intégré (EDI) Arduino

4.2.1 Le langage de programmation

Un langage de programmation est un langage permettant à un être humain d'écrire un ensemble d'instructions (code source) qui seront directement converties en langage machine grâce à un compilateur (c'est la compilation). L'exécution d'un programme Arduino s'effectue de manière séquentielle, c'est-à-dire que les instructions sont exécutées les unes à la suite des autres. Voyons plus en détail la structure d'un programme écrit en Arduino.

4.2.2 Description de l'interface

Le logiciel Arduino a pour fonctions principales :

- de pouvoir écrire et compiler des programmes pour la carte Arduino.
- de se connecter avec la carte Arduino pour y transférer les programmes.
- de communiquer avec la carte Arduino.

a) Lancement du logiciel

Lançons le logiciel en double-cliquant sur l'icône avec le symbole « infinie » en vert. C'est l'exécutable du logiciel. Après un léger temps de réflexion, une image s'affiche :

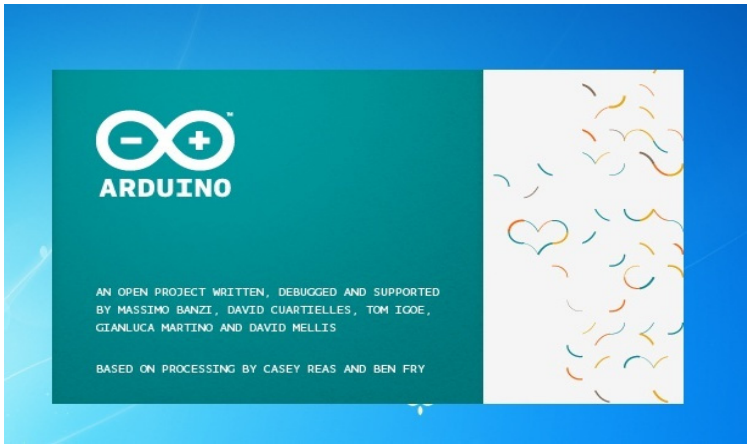


Figure 4.2 Page principale de logiciel Arduino [6].

Cette fois, après quelques secondes, le logiciel s'ouvre. Une fenêtre se présente à nous comme suite :

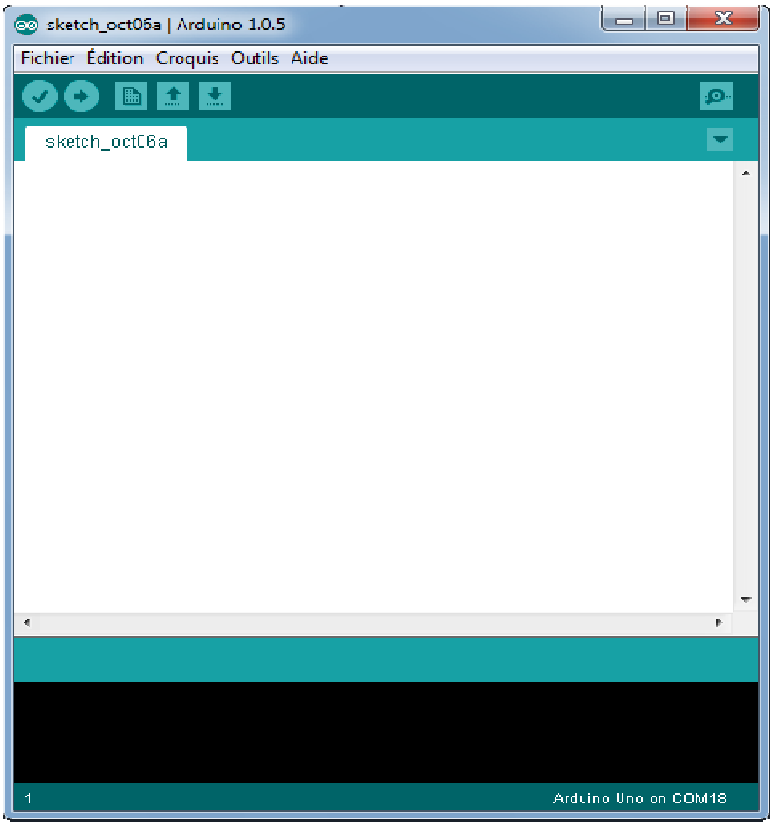


Figure 4.3 Fenêtre principale du logiciel Arduino [6].

L'interface est assez claire. Voyons maintenant ca composition.

On a découpé l'image précédente en plusieurs parties comme suite :

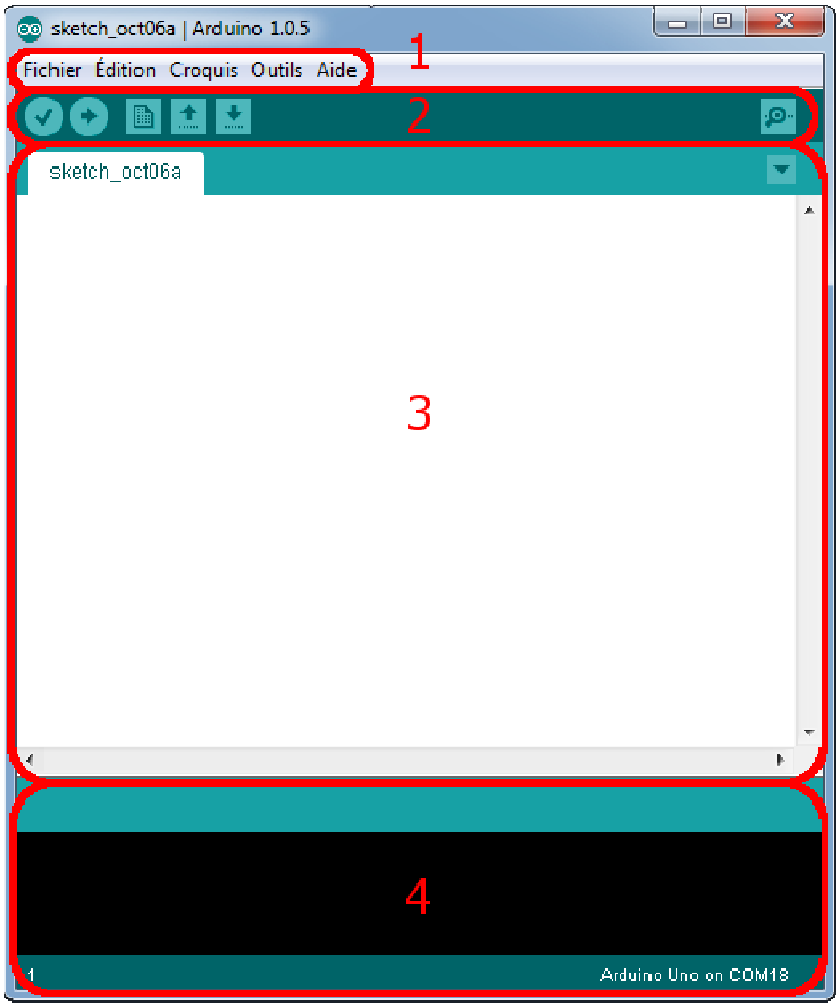


Figure 4.4 Décomposition de la fenêtre principale [6].

b) Correspondance

- Le cadre numéro 1 : ce sont les options de configuration du logiciel.
- Le cadre numéro 2 : il contient les boutons qui vont nous servir lorsque l'on va programmer nos cartes.
- Le cadre numéro 3 : ce bloc va contenir le programme que nous allons créer.
- Le cadre numéro 4 : celui-ci est important, car il va nous aider à corriger les fautes dans notre programme. C'est le **débogueur**.

4.2.3 Approche et utilisation du logiciel

Attaquons-nous maintenant à l'utilisation du logiciel. La barre des menus est entourée en rouge et numérotée par le chiffre 1.

4.2.3.1 Le menue file

C'est principalement ce menu que l'on va utiliser le plus. Il dispose d'un certain nombre de choses qui vont nous être très utiles. Il a été traduit en français progressivement, nous allons donc voir les quelques options qui sortent de l'ordinaire :

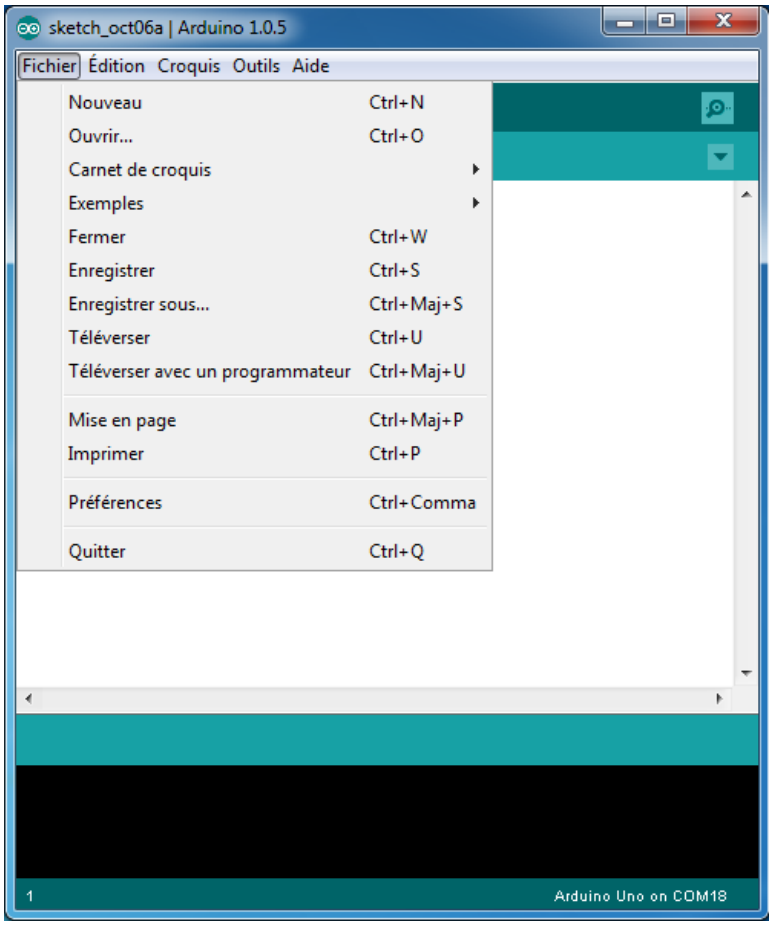


Figure 4.5 La fenêtre menu.

- ***Carnet de croquis*** : CE menu regroupe les fichiers que vous avez pu faire jusqu'à maintenant (et s'ils sont enregistré dans le dossier par défaut du logiciel).
- ***Exemples*** **(exemples)** : ceci est important, toute une liste se déroule pour afficher les noms d'exemples de programmes existants ; avec çà, vous pourrez vous aider/inspirer pour créer vos propres programmes ou tester de nouveaux composants
- ***Téléverser*** : Permet d'envoyer le programme sur la carte Arduino.

- ***Téléverser avec un programmateur*** : Idem que si dessus, mais avec l'utilisation d'un programmateur (vous n'en n'aurez que très rarement besoin).
- ***Préférences*** : Vous pourrez régler ici quelques paramètres du logiciel le reste des menus n'est pas intéressant pour l'instant, on y reviendra plus tard, avant de commencer à programmer [6].

4.2.3.2 Les Boutons

Voyons à présent à quoi servent les boutons, encadrés en rouge et numérotés :

- **Bouton 1** : Ce bouton permet de vérifier le programme, il actionne un module qui cherche les erreurs dans votre programme.
- **Bouton 2** : Charge (téléversé) le programme dans la carte Arduino.
- **Bouton 3** : Crée un nouveau fichier.
- **Bouton 4** : Ouvre un fichier.
- **Bouton 5** : Enregistre le fichier.
- **Bouton 6** : Ouvre le moniteur série [7].

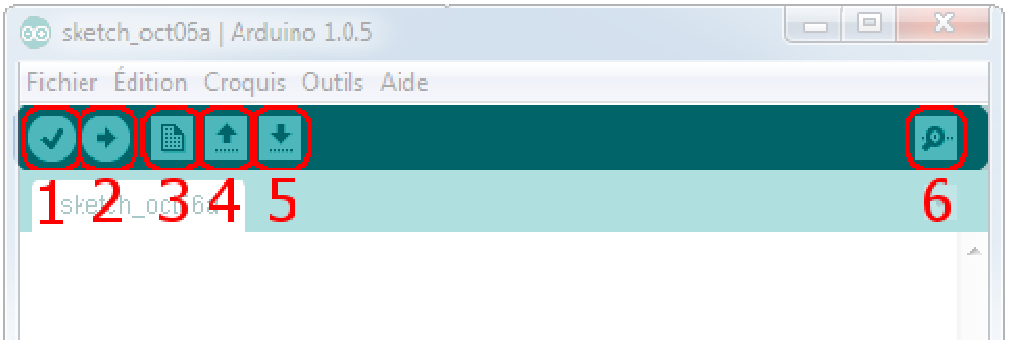


Figure 4.6 Les boutons nécessaires.

4.2.4 La structure d'un programme

Un programme Arduino comporte trois parties :

1. La partie déclaration des variables (optionnelle).
2. La partie initialisation et configuration des entrées/sorties : la fonction **setup ()**.
3. La partie principale qui s'exécute en boucle : la fonction **loop()**.

Dans chaque partie d'un programme sont utilisées différentes instructions issues de la syntaxe du langage Arduino.

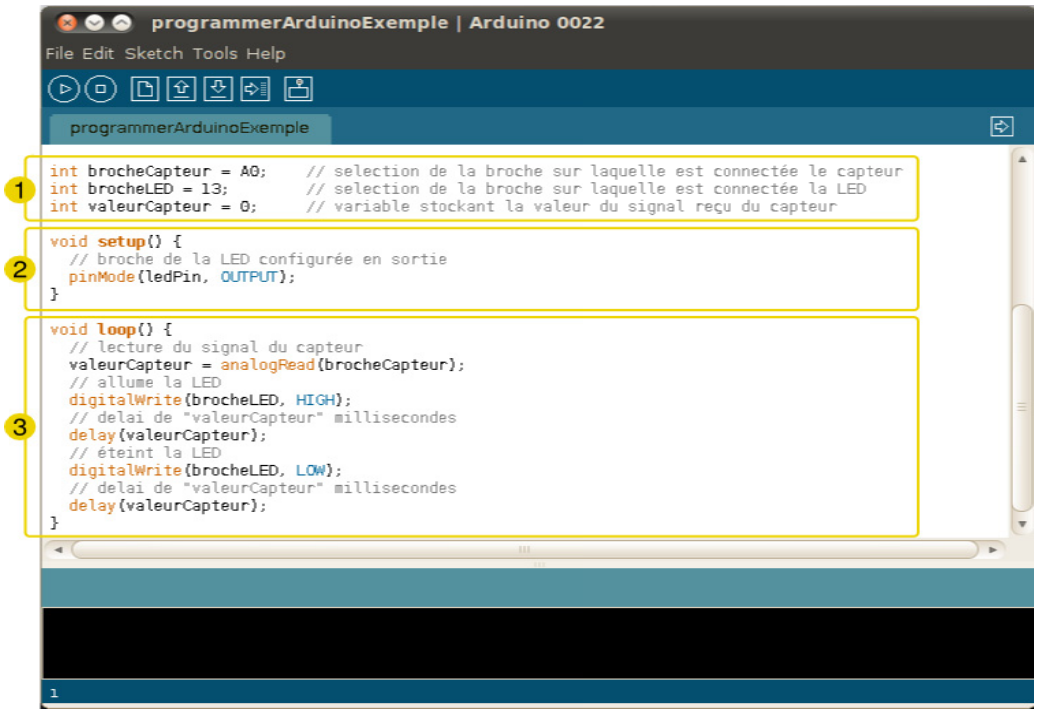


Figure 4.7 Exemple d'un programme Arduino.

4.2.4.1 Coloration syntaxique

Lorsque du code est écrit dans l'interface de programmation, certains mots apparaissent en différentes couleurs qui clarifient le statut des différents éléments :

En **orange**, apparaissent les mots-clés reconnus par le langage Arduino comme des **fonctions** existantes. Lorsqu'on sélectionne un mot coloré en orange et qu'on effectue un clic avec le bouton droit de la souris, l'on a la possibilité de choisir « Find in reference » : cette commande ouvre directement la documentation de la fonction sélectionnée.

En **bleu**, apparaissent les mots-clés reconnus par le langage Arduino comme des **constantes**.

En **gris**, apparaissent les **commentaires** qui ne seront **pas exécutés dans le programme**. Il est utile de bien commenter son code pour s'y retrouver facilement ou pour le transmettre à d'autres personnes. L'on peut déclarer un commentaire de deux manières différentes :

- dans une ligne de code, tout ce qui se trouve après « *//* » sera un commentaire.
- l'on peut encadrer des commentaires sur plusieurs lignes entre « ***/**** » et « ****/*** ».

4.2.4.2 Syntaxe du langage

Le code est structuré par une ponctuation stricte :

- **toute ligne** de code se termine par un point-virgule « **;** »
- le contenu d'une **fonction** est délimité par des accolades « **{** » et « **}** »
- les **paramètres** d'une fonction sont contenus pas des parenthèses « **(** » et « **)** ».

Une erreur fréquente consiste à oublier un de ces éléments.

4.2.4.3 Les variables

Une variable est un espace réservé dans la mémoire de l'ordinateur. C'est comme un compartiment dont la taille n'est adéquate que pour un seul type d'information. Elle est caractérisée par un nom qui permet d'y accéder facilement.

Il existe différents types de variables identifiés par un mot-clé dont les principaux sont :

- nombres entiers (int)
- nombres à virgule flottante (float)
- texte (String)
- valeurs vrai/faux (boolean).

Un nombre à décimales, par exemple *3.14159*, peut se stocker dans une variable de type float. Notez que l'on utilise un point et non une virgule pour les nombres à décimales. Dans Arduino, il est nécessaire de déclarer les variables pour leurs réserver un espace mémoire adéquat. On déclare une variable en spécifiant son type, son nom puis en lui assignant une valeur initiale (optionnel).

Exemple :

```
int ma_variable = 45;

 // int est le type,  ma_variable le nom et = 45 assigne une valeur.
```

4.2.4.4 Les fonctions

Une fonction (également désignée sous le nom de procédure ou de sous-routine) est un bloc d'instructions que l'on peut appeler à tout endroit du programme.

Le langage Arduino est constitué d'un certain nombre de fonctions, par exemple : analogRead(), digitalWrite() ou delay().

Il est possible de déclarer ses propres fonctions par exemple :

```
void clignote()
{
```

```
  digitalWrite (brocheLED,  HIGH) ;
  delay (1000) ;
  digitalWrite (brocheLED,  LOW) ;
delay (1000) ;
}
```

Pour exécuter cette fonction, il suffit de taper la commande :

```
clignote();
```

On peut faire intervenir un ou des **paramètres** dans une fonction :

```
 void clignote(int broche,int vitesse)
{
  digitalWrite (broche, HIGH) ;
  delay (1000/vitesse) ;
  digitalWrite (broche, LOW) ;
  delay (1000/vitesse) ;
}
```

Dans ce cas, l'on peut moduler leurs valeurs depuis la commande qui l'appelle :

```
clignote(5,1000); //la sortie 5 clignotera vite
clignote(3,250); //la sortie 3 clignotera lentement.
```

4.2.4.5 Les structures de contrôle

Les structures de contrôle sont des blocs d'instructions qui s'exécutent en fonction du respect d'un certain nombre de conditions.

Il existe quatre types de structure :

if...else : exécute un code **si** certaines conditions sont remplies et éventuellement exécutera un autre code avec **sinon.**

exemple :

```
//si la valeur du capteur depasse le seuil
if(valeurCapteur>seuil){
   //appel de la fonction clignote
   clignote();
}
```

4.3 Langage Java

4.3.1 Introduction

Le langage JAVA est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (confondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portable sur plusieurs systèmes d'exploitation tels que UNIX , Windows, Mac OS, GNU/Linux, avec peu ou pas de modifications, pour cela, divers plateformes et Framework associés visent à guider, sinon garantir, cette portabilité des applications développées en Java[3].

Figure 4.8 Logo de Java.

Afin de réaliser notre interface permettant aux utilisateurs de contrôler le système, nous avons choisi le langage Java.

4.3.2 Motivations

Ce choix a été motivé par les raisons suivantes :

- Java assure une totale indépendance des applications vis-à-vis de l'environnement d'exécution, c'est-à-dire que toute machine supportant Java est en mesure d'exécuter un

programme sans aucune adaptation (ni recompilation, ni paramétrage de variables d'environnement).

- Une programmation orientée objet et une bibliothèque immense d'objets. Dès sa naissance, les programmeurs de Sun ont doté leur langage d'une des plus grandes bibliothèques d'objets prête à l'emploi.

4.4 Environnement de développement

4.4.1 Architecture physique de déploiement

La mise en œuvre de l'interface proposée est réalisée sur un ordinateur doté d'un système d'exploitation Windows7 sur lequel est installé le Moteur OpenJDK Java 8, le système de gestion de la base de données MySQL. La manière dont les composants physiques du système sont organisés est illustrée dans le diagramme de déploiement suivant (Figure 4.9) :

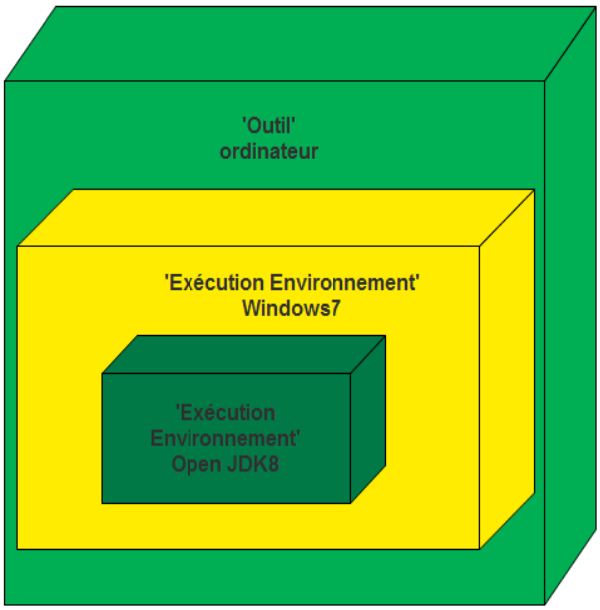


Figure 4.9 Diagramme de déploiement du système.

4.4.2 Outils et langage de programmation

4.4.2.1 NetBeans

NetBeans est un projet open source ayant un succès et une base d'utilisateurs très large. Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X. Aujourd'hui, deux projets existent :

- **EDI NetBeans :** c'est un environnement de développement, un outil pour les programmeurs pour écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java, mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'EDI NetBeans.

- **Plateforme NetBeans :** c'est une fondation modulable et extensible utilisée comme brique logicielle pour la création d'applications bureautiques. Les partenaires privilégiés fournissent des modules à valeurs ajoutées qui s'intègrent facilement à la plateforme et peuvent être utilisés pour développer ses propres outils et solutions.

Figure 4.10 NetBeans IDE 8.1 in Windows7.

4.4.2.2 MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il fait partie des logiciels de gestion de base de données les plus utilisés au monde. Il est un serveur de base de données SQL très rapide, multithread, multi-utilisateurs et robuste. MySQL est développé, distribué et supporté par MySQL AB.

Figure 4.11 Logo de MySQL.

4.5 Mise en œuvre de l'interface

En utilisant les outils cités précédemment, nous avons abouti à une implémentation composée des interfaces suivantes **:**

4.5.1 Algorithmes

Les algorithmes nécessaires qu'on a utilisés sont les suivants :

Algorithme1 : Allumer un Système

```
try {
com=SerialPort.getCommPort(jTextField1.getText());
        if(com.openPort()){
           com.setComPortParameters(9600, 8, 1, SerialPort.NO_PARITY);
JOptionPane.showMessageDialog(null, "La connexion est établie avec le système","Information",JOptionPane.INFORMATION_MESSAGE);
pw=new PrintWriter(com.getOutputStream());
MsgEnvoi m=new MsgEnvoi(pw, "c");
m.start();
            bf=new BufferedReader(new InputStreamReader(com.getInputStream()));
            MsgRec mr=new MsgRec(bf, jLabel5);
            mr.start();
            System.out.println("4");
}
      }catch(Exception e){
JOptionPane.showMessageDialog(null, "Un problème de communication avec le system","Erreur",JOptionPane.ERROR_MESSAGE);
       }
      }
```

***Algorithme2 :** Ajouter un Système

```
private void jLabel6MouseClicked(java.awt.event.MouseEvent evt) {
int y=(int)(t3.getValue());
if (t1.getText().length()==0){
         JOptionPane.showMessageDialog(null, "Remplire le champ Nom du robot SVP!!","Attention",JOptionPane.WARNING_MESSAGE);
      }
       else {
```

```
        if(t4.getText().length()==0){
JOptionPane.showMessageDialog(null, "Remplire le champ Nombre de capteur SVP!!","Attention",JOptionPane.WARNING_MESSAGE);
        }
        else {
          if (y==0){
              JOptionPane.showMessageDialog(null, "Remplire le champ Langeur du détection SVP!!","Attention",JOptionPane.WARNING_MESSAGE);
  }
          else {

            try {
  st.executeUpdate("insert into Systéme  (nom,type,nbr_cap,dist,date_cre) values ('"+t1.getText()+"','"+t2.getSelectedItem().toString()+"','"+y+"','"+t4.getText()+"','"+x+"') ");
  JOptionPane.showMessageDialog(null, "Le Système "+t1.getText()+" a été bien ajouté","Information",JOptionPane.INFORMATION_MESSAGE);
       } catch (SQLException ex) {
           JOptionPane.showMessageDialog(null, "Un problème d'ajout avec le système "+t1.getText(),"Erreur",JOptionPane.ERROR_MESSAGE);
      }
          }
   }}
       t1.setText(""); t2.setSelectedIndex(0); t4.setText("");t3.setValue(0);
     }
```

*__Algorithme3 :__ Liste des utilisateurs

```
this.setTitle("Liste des utilisateurs");
this.setSize(600,460);
       this.setLocationRelativeTo(null);
       this.setResizable(false);
       try {
         st=conBDD.obtenirconection().createStatement();
         rs=st.executeQuery("select                                    *,CONVERT (DATEDIFF('"+x+"',date_nais)/366,SIGNED INTEGER) as age from utilisateur");
```

```
            while(rs.next()){
              m1.addRow(new
Object[]{rs.getString(1),rs.getString(3),rs.getString("age"),rs.getString(5),rs.getString(6
)});
            }
            jTable1.setModel(m1);
             rs=st.executeQuery("select * from utilisateur where user='"+user+"'");
    if(rs.next()){
               jLabel1.setText("Bonjour "+rs.getString("nom")+" sur votre profil");
            }
         } catch (SQLException ex) {
           JOptionPane.showMessageDialog(null, "Erreur de connexion avec la base
de donnée","Erreur",JOptionPane.ERROR_MESSAGE);
    }  }
```

*__Algorithme 4 :__ Recevoir un message

```
    while(true){
    try{

    String m = bf.readLine();System.out.println("1");
                        if(m.equals("d"))
                       {System.out.println("2");
                         //aff5.setIcon(new
ImageIcon(getClass().getResource("droite.png")));
                         }if(m.equals("g")){
                           System.out.println("3");
                          //                                          aff5.setIcon(new
ImageIcon(getClass().getResource("gauche.png")));
    }
                      }
                    catch(Exception ex){
                       JOptionPane.showMessageDialog(null, "erreur de rec");
                    }
     }
```

***Algorithme5 :** Envoyer un message

```
public class MsgEnvoi  extends Thread{
public PrintWriter pw;
public String c;
    public MsgEnvoi(PrintWriter pw,String c) {this.pw = pw; this.c=c;}

    public void run() {
     pw.println(c);
pw.flush();
JOptionPane.showMessageDialog(null, "Le système a commencé son travail","Information",JOptionPane.INFORMATION_MESSAGE);
}
}
```

4.5.2 Mise en œuvre de l'interface Java

4.5.2.1 Interface d'Authentification

Dans cette interface on doit remplir les cases de **Nom d'utilisateur** et le **Mot de passe**, si le Nom D'utilisateur et le Mot de passe sont enregistrés dans notre base de données on passe à l'interface du Menu, sinon le système affiche un message :(Nom d'utilisateur et/ou mot de passe incorrect).

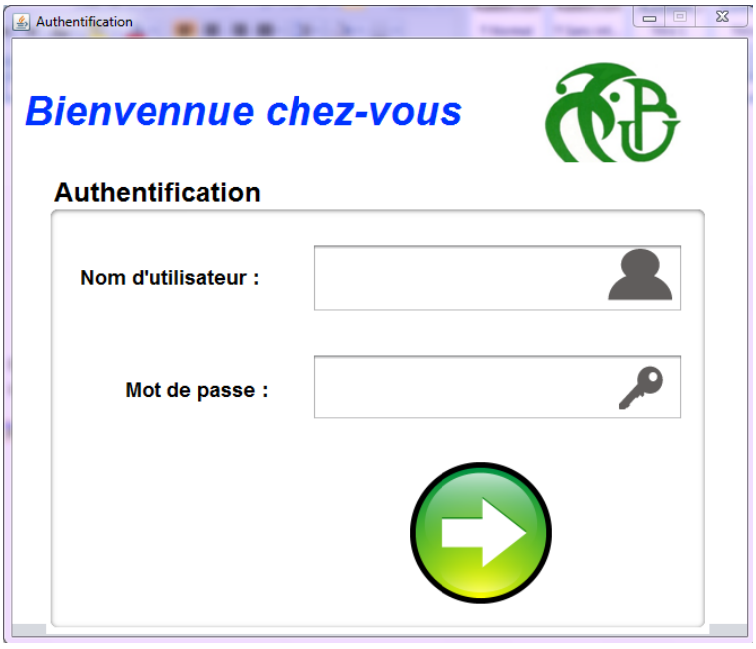


Figure 4.12 Interface d'Authentification.

4.5.2.2 Accueil

Dans cette interface on peut communiquer avec un ou plusieurs systèmes , on doit sélectionner un système, ensuite on ajoute le Port de notre système dans la case **Serial communication,** et on clique sur **Start** pour allumer notre système et **Stop** pour arrêter le système, et dans la case **Détection** on doit préciser la direction de notre système.

Figure 4.13 L'interface du menu principale.

4.5.2.3 Liste des utilisateurs

Pour trouver cette interface on clique sur le bouton fichier et on choisit **Liste des utilisateurs.**

Figure 4.14 Interface de liste des utilisateurs.

4.5.2.4 Liste des systèmes

Pour trouver cette interface on doit aussi cliquer sur le bouton Fichier et on choisit **Liste des systèmes**. Les données (SystemeID, Nomsysteme, Typesysteme ,Date création,NombreCapteur,langeur) sont enregistrées dans notre base de donnée.

Figure 4.15 Interface de la liste des systèmes.

4.6 La programmation

4.6.1 Schéma de principe de la réalisation

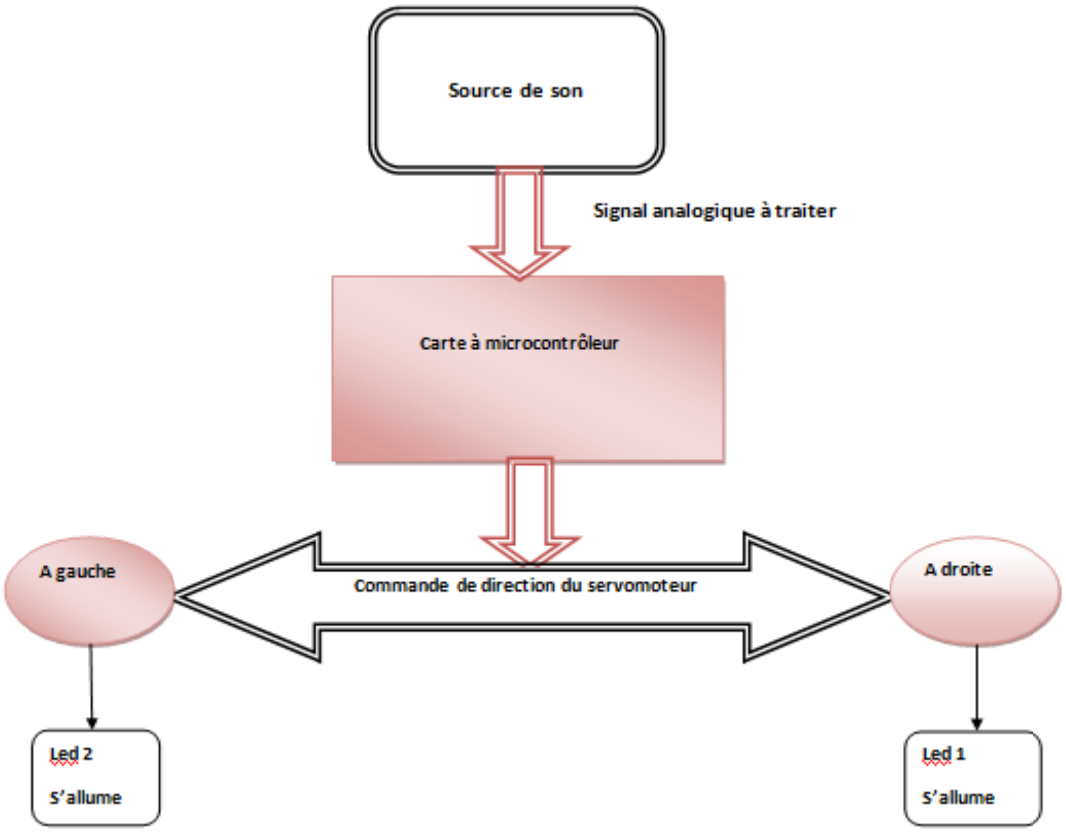


Figure 4.16 Schéma synoptique.

4.6.2 Fonctionnement

Tout d'abord en utilisant l'interface java, on fait démarrer notre système. Initialement il est en repos à 90°,à l'aide de capteurs de son, il doit tourner et suivre la position de la source, lorsqu'on siffle à droite le servomoteur doit tourner à 30° a droite et la led 1 s'allume, et lorsqu'on siffle à gauche, le servomoteur doit tourner à 140° à gauche et la led 2 s'allume. Pour arrêter le fonctionnement on clique sur stop.

4.6.3 Organigramme

La figure suivante représente l'Organigramme de la programmation de gestion de la commande du moteur.

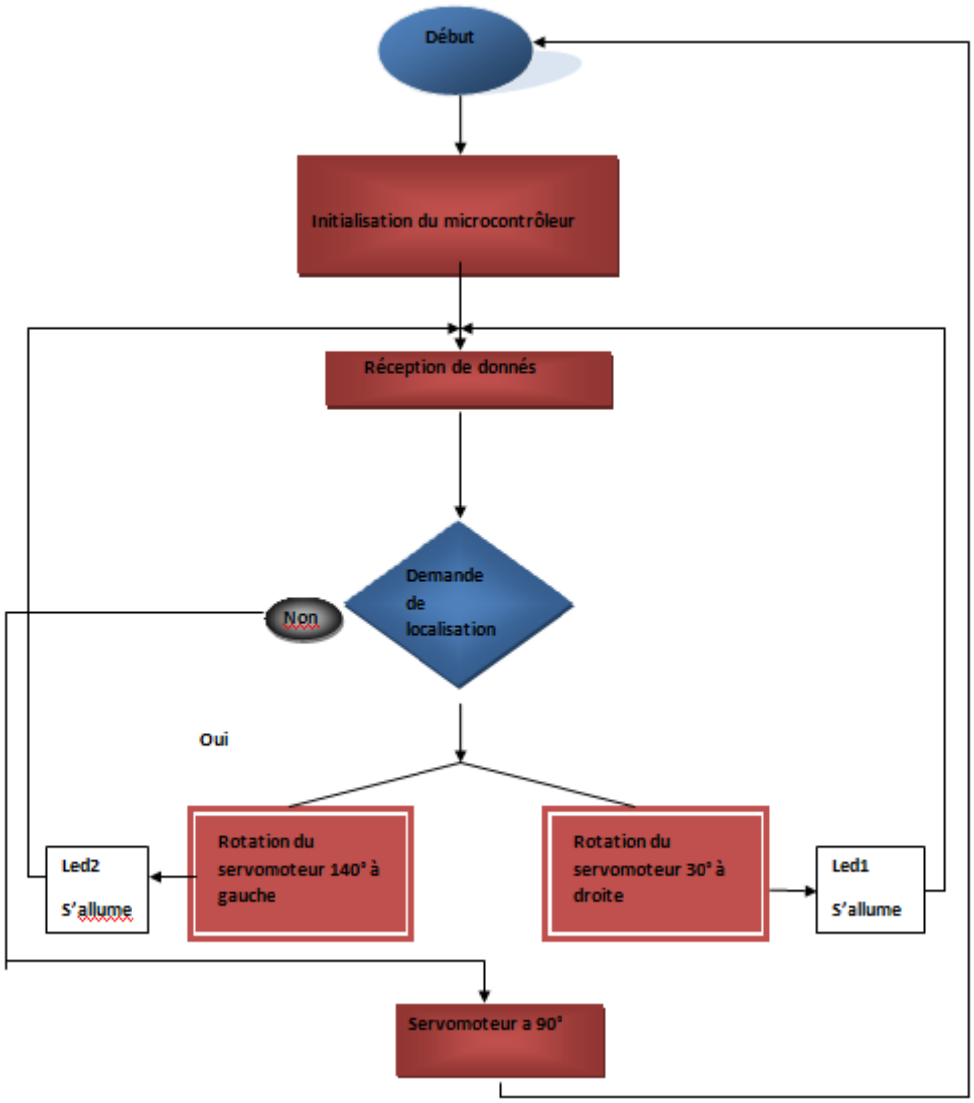


Figure 4.17 Organigramme de la programmation de gestion de la commande du moteur.

4.6.4 Explication de l'organigramme

On doit tout d'abord initialiser le microcontrôleur pour permettre son fonctionnement. Ensuite on reçoit des données provenant d'un programme qui nous fournit le sens de rotation du servomoteur. Enfin selon les données reçues le programme envoie des signaux qui permettent de gérer la direction du moteur, c'est-à-dire sa rotation à droite ou à gauche selon la source sonore.

4.7 Tests et essais

Dans cette partie, nous allons montrer les différents tests effectués.

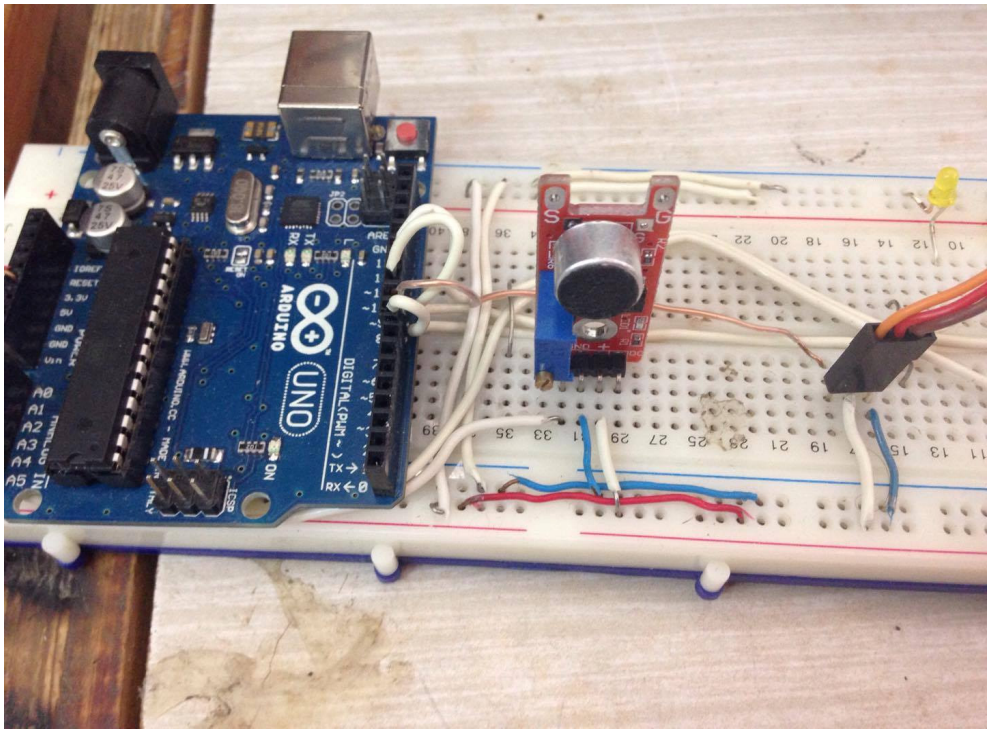


Figure 4.18 Connexion de la carte Arduino avec les différents composants.

Comme nous l'avons mentionné au départ, nous avons réalisé un système de localisation à temps réel à l'aide d'une carte Arduino et deux microphones.

Dans notre système de localisation, nous avons été contraintes de choisir le moteur qui lui convenait le mieux. Ce dernier est assez puissant pour suivre la source du son. Pour cela la consommation du courant doit être minime, donc nous avons opté pour le servomoteur SM-S4303R pour une rotation continue à 6v qui était idéal pour notre prototype. Aussi, on a utilisé aussi des microphones de haute sensibilité ky-037 pour une bonne acquisition du son,

car ils sont équipés d'un amplificateur LM 398 et d'une résistance réglable pour le seuil de sensibilité.

Après avoir implémenté le code pour faire fonctionner les capteurs de son et réaliser le code pour faire fonctionner le système et après plusieurs tests, un problème de synchronisation des microphones est survenu. On a dû donc régler à chaque fois les microphones pour avoir approximativement la même fréquence. Et on a remarqué que lorsqu'on augmente les fréquences au-delà de 605 Hz, le servomoteur tourne aléatoirement. Donc on a dû mettre des résistances aux microphones pour les stabiliser et réduire le bruit.

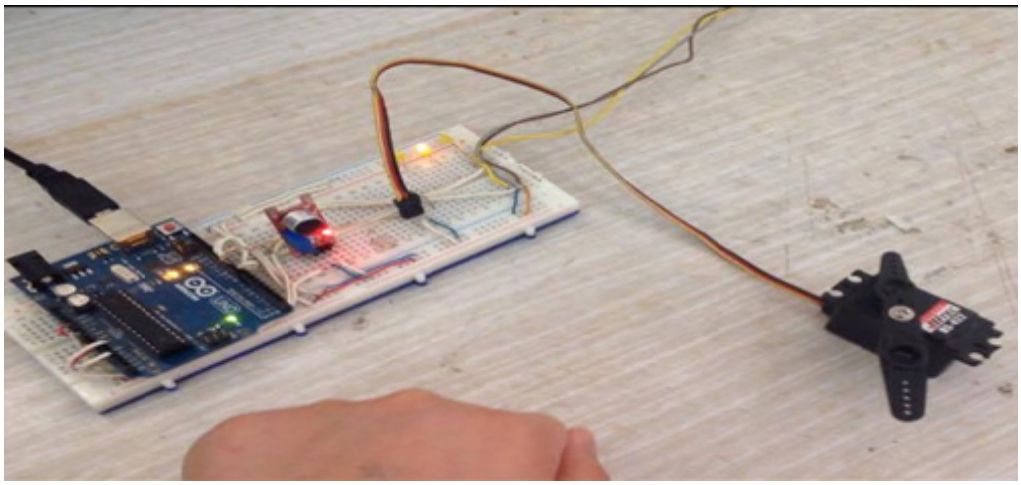


Figure 4.19 Localisation à gauche et le servomoteur tourne à gauche.

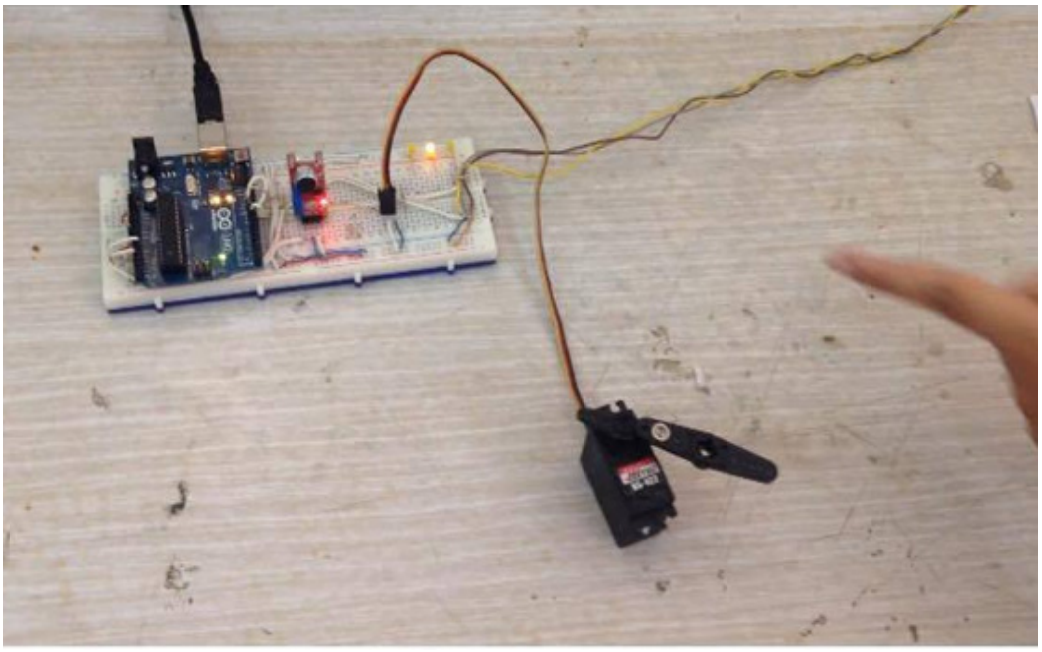


Figure 4.20 Localisation à droite et le servomoteur tourne à droite.

4.8 Conclusion

Dans ce chapitre on a décrit les différents logiciels et langages qu'on a manipulé pour la mise en œuvre de notre projet. Des logiciels informatiques tel que le Java qui nous a permis de réaliser une interface pour notre application, et électroniques tel que l'Arduino qui nous a permis de programmer notre carte Arduino et de transférer des programmes.

Malgré la simplicité apparente du système, nous avons vu qu'il ne manquait pas de difficultés à la réalisation, son mécanisme est tout aussi complexe que sa programmation.

Conclusion générale

L'objectif de ce travail est de réaliser un système de localisation de source sonore à l'aide de la carte Arduino et de deux microphones. Pour atteindre notre objectif final, on a été amené à utiliser une méthode appelée inter-corrélation généralisée (GCC) qui opère en temps réel. La méthode présentée a été intégrée dans une étape de détection des locuteurs pour localiser une source active. En résumé, un algorithme de type GCC-PHAT pour SSL en temps réel a été employé et évalué dans un scénario pratique.

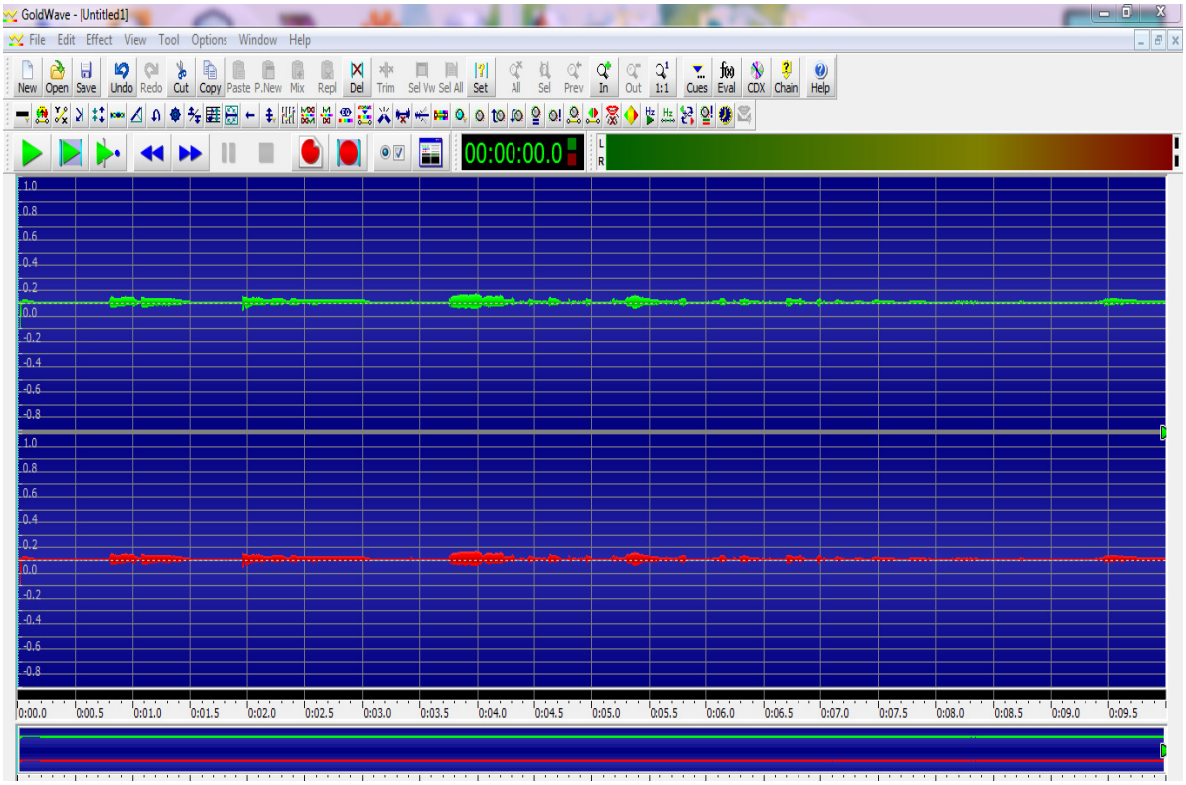
La motivation de ce choix est d'arriver à embarquer cette méthode de localisation sonore pour des applications de surveillances assistées par des caméras.

Ce projet exige des connaissances diverses et variées telles que l'électronique, l'acoustique, le traitement du signal et l'informatique. L'interaction de ces différentes disciplines constitue une expérience enrichissante, à la fois sur le plan humain et scientifique.

Annexe 1 - Logiciel GoldWave

GoldWave est un logiciel de montage numérique audio, un éditeur audio populaire et commercial pour la manipulation des sons numériques, audio multipistes/vidéo et logiciel de mixage, pour des systèmes informatiques personnels.

"GoldWave Inc" est nommée en 2001 après l'édition de "GoldWave Digital Audio" qui a été commercialisée en avril 1993. Le logiciel a continué de s'améliorer au cours des 20 dernières années et est considéré actuellement comme l'un des meilleurs de sa catégorie.GoldWave" est un logiciel réalisé à l'origine par Chris Craig (Canada) à partir d'une idée de Freeware (ScopeTrax 1992). "GoldWave Audio Digital" devient, à partir de 2001, "GoldWave Inc". "GoldWave" permet de créer, d'éditer, de traiter, de modifier, de scinder, d'assembler, de convertir des sons numériques.



Annexe 2 - Programme Arduino

```
#include <Servo.h>
Servo myservo;
 int pos=90,drop=90,load=140,e=0,down=40;

int DA = A3; // Pin for Analog Output - AO
int threshold = 41; //Set minimum threshold for LED lit
int sensorvalue = 0;
int DA1 = A2;
int threshold1 = 605; // Set minimum threshold for LED lit
int sensorvalue1 = 0;
void setup() {
  Serial.begin(9600);
  myservo.attach(9);
  myservo.write(90);
  pinMode(8, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(12, OUTPUT);

  pinMode(DA, INPUT);
  pinMode(DA1,INPUT);
}
void loop()
{
  while(1){
  sensorvalue = analogRead(DA);  //Read the analog value
   sensorvalue1 = analogRead(DA1);  //Read the analog value
int index=Serial.read(); // lire un premier caractère
if(index >= 'A' && index <= 'z'){
int valeur = Serial.parseInt();
  // delay(500);
// myservo.write(90);
  switch(index){
     case 'c':
  sensorvalue = analogRead(DA);  //Read the analog value
   sensorvalue1 = analogRead(DA1);  //Read the analog value
   if (sensorvalue > 95 )
   {
       Serial.print("d");
       delay(100);
       digitalWrite (8,HIGH);
```

```
      digitalWrite (10,LOW);
      digitalWrite (12,LOW);
      myservo.write(140);
      delay(1000);
   }
   else {
      myservo.write(90);
      delay(100);
   }
    if (sensorvalue1 > 120 )
   {
      Serial.print("g");
      delay(100);
      digitalWrite (8,LOW);
      digitalWrite (10,LOW);
      digitalWrite (12,HIGH);
      myservo.write(30);
      delay(1000);
   }
   else {
      myservo.write(90);
      delay(100);    }
  break;
    case 't':
      digitalWrite (8,LOW);
      digitalWrite (10,LOW);
      digitalWrite (12,LOW);
      myservo.write(90);
      delay(1000);
break;
  }  }   }
 }
```

Bibliographie

[1] Knapp C. H. and Carter G. C., "The Generalized Correlation Method for Estimation of Time Delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 4, pp. 320-327, August 1976.

[2] M. S. Brandstein, J. E. Adcock, and H. F. Silverman, "Method and Apparatus for Source Location Estimation from Microphone-Array Time-Delay Estimates," Patent 5737431, April 7, 1998.

[3] Claude Lesueur, "Éléments de base en acoustique physiologique et physique," in *Acoustique*., 1997, ch. 1, p. 15.

[4] Antoine Lorenzi and Anthony Gentil. (2016, Dec.) Voyage au centre de l'audition. [Online]. http://www.cochlea.eu/son/psychoacoustique/localisation

[5] Mariam Sall, "La localisation d'une source sonore dans un environnement 2D," Université Joseph Fourier, Grenoble, France, Rapport de stage de Master 1ére Année EEATS 2015.

[6] Eskimon. (2014) Le blog d'Eskimon. [Online]. http://eskimon.fr/73-arduino-101-presentation

[7] (2013) For Projects. [Online]. http://duino4projects.com/sound-localization-using-arduino/

[8] (2017) Projets Diy. [Online]. https://projetsdiy.fr/piloter-servomoteur-arduino/#.WQUEykWLTIV

[9] Grégory Rump, "Method for locating a sound source, and humanoid robot using such a method ," WO2015049199 A1, April 9, 2015.