ملخص:هذا العمل يتمثل في تصميم وإنجازأداة تتيح قياس التشابه بين إشارتينعن طريق حساب معامل الارتباط. ويستند هذا الإنجاز على بطاقةأردوينو ميقا 2560 التي تتميز بأنها صغيرةو بالتاليقابلة للنقل بسهولة. حيث يتم تحميل البرنامج في البطاقة و يتم عرض نتيجة معامل الارتباط على شاشة LCDالمرتبطة بالبطاقة.

كلمات المفاتيح: معامل الارتباط ، إشارة، بطاقة الأردوينو ميقا 2560،تطبيق أردوينو.

Résumé : Le présent travail consiste à concevoir et à réaliser un boitier de corrélation qui permet de mesurer la similarité entre deux signaux en calculant le coefficient de corrélation.

La réalisation se base sur une carte ARDUINO Méga 2560, qui présente l'avantage d'être petite et donc facilement transportable. Le programme en langage Arduinoest chargé dans la carte et le résultat du calcul du coefficient de corrélation s'affiche sur un écran LCD mis en relation avec la carte.

Mots clés :Coefficient de Corrélation, Signal, Arduino Méga 2560, IDE Arduino.

Abstract: The present work is to conceiving and realizing a correlation box that measures the similarity between two signals by calculating the correlation coefficient.

The realization is based on an Arduino Mega 2560 board that has the advantage of being small and therefore easily transportable. The Arduino language program is loaded into the board and the calculation result of the correlation coefficient is displayed on an LCD set in relation with the board.

Keywords: Correlation Coefficient, Signal, Arduino Mega 2560 ,Arduino IDE.

Les cartes d'interfaces (shields)

Pour la plupart des projets, il est souvent nécessaire d'ajouter des **fonctionnalités** aux cartes Arduino. Plutôt que d'ajouter soit même des composants extérieurs (sur une platine d'essai, circuit imprimé, etc.), il est possible d'ajouter des ***shields***. Un shield est une carte (voir figure 2.5) que l'on connecte directement sur la carte Arduino qui a pour but d'ajouter des composants sur la carte. Ces *shield* viennent généralement avec une **librairie** permettant de les contrôler. On retrouve par exemple, des *shields* Ethernet, de contrôle de moteur, lecteur de carte SD, etc [13].

Le principal avantage de ces *shields* est leurs simplicités d'utilisation. Il suffit des les emboiter sur la carte Arduino pour les connecter, les circuits électronique et les logiciels sont déjà faits et on peut en empiler plusieurs. C'est un atout majeur pour ces cartes pour pouvoir tester facilement de nouvelles fonctionnalités. Cependant il faut bien garder à l'esprit que les *shields* ont un prix. Suivant les composants qu'ils apportent, leurs prix peuvent aller de 5 à 100€ ! [13].

Figure 2.5. *Les cartes d'interfaces : Bluetooth, Wifi et Ethernet*

Bibliographie

[1] Marie TAHON : ‘Traitement du signal ‘ Laboratoire d’Acoustique, Conservatoire National des Arts et Métiers.

[2] Christian JUTTEN :’ théorie du signal ‘ Université Joseph Fourier - Polytech Grenoble.

[3] O. Laligant, F. Truchetet2010 :’Cours MP Techniques de Traitement du Signal Une approche phénoménologique ‘.

[4] James L. Crowley :’Signaux physiques et modèles théoriques ‘.

[5] Atman GUERCHAOUI :’ Traitement du signal ‘ office des publications Universitaire.

[6] Michel Armatte :‘ convolution et corrélation ‘.

[7] Mme Radjai Yamina Melle Oussaid Samia :’ Etude et Implémentation de la Méthode du Gradient Stochastique et Application au Signal de transmition ‘ Mémoire d’ingénieur d’état, Blida, 2002.

[8]RiccoRakotomalala :’ Analyse de corrélation Étude desdépendances Variables quantitatives’

[9] Zarrouk Fayçal :’cours de statistique ‘.

[10] Pierre MAGAIN :’ introduction aux méthodes quantitatives et élement de statistique ‘université de liége.

[11] Thierry JAMES :’ Techniques d’analyses en psychologie Cours 12. Corrélation et régression ‘.

[12] KRAMA Abdelbasset et GOUGUI Abdelmoumen :'Mémoire MASTER ACADEMIQUE : ‘Etude et réalisation d’une carte de contrôle par Arduino via le système Androïde ‘, Université Kasdi Merbah Ouargla, 2015.

[13] ASTUPS CampusFab et LECHALUPÉ Julien :'Cours d’initiation à Arduino', Université Paul Sabatier, 2014.

[14] Simon Landrault (Eskimon) et Hippolyte Weisslinger (olyte): ' Arduino Premiers pas en informatique embarquée', 2014.

[15] Hocine TAKHI : ' Conception et réalisation d'un robot mobile à base d'arduino ' Université Amar Telidji, 2014.

[16] YACINE A. AMKASSOU, Omar ELBARKANI, Sliman ELNAIRY :' Mini projet : Thermomètre a base du Arduino' , Ecole Nationale Des Sciences Appliquées Khouribga, 2012-2013

[17] PIERRE-YVES ROCHAT: 'Introduction Au Microcontrôleur', EPFL, 2016.

[18] Jean-Noël : ' Microcontrôleur ATMEGA ', Janvier 2005.

[19] Michael McRoberts: 'Beginning Arduino', Technology In Action, Second Edition.

[20] Christian Tavernier : ' Arduino Applications avancées ', DUNOD, 2012.

[21] Référence Arduino français http://www.mon-club-elec.fr

[22] Christian Tavernier :'Arduino : Maitrisez sa programmation et ses cartes d'interface (shields) ', DUNOD, 2014.

[23] Jean-Noël: 'Atelier Arduino : Initiation à la mise en œuvre matérielle et logicielle de l'Arduino, Centre de Ressources Art Sensitif, novembre 2006.

[24] S.V.D.Reyvanth, G.Shirish: ' PID controller using Arduino.

Bibliographie

Dédicace

Je dédie ce modeste travail

A mes très chers parents

A mes sœurs

A mes frères

A mes vrais amis

Hayat

Chapitre 1

Généralités sur les signaux

et la corrélation

Résumé

Liste des figures

Liste des tableaux

Listes des acronymes et abréviations

Remerciments

Dédicace

Annexe

Introduction Générale

Conclusion Générale

Table des matières

Sommaire

Introduction générale ……1

Chapitre1 :

Généralités sur les signaux et la corrélation

1.1 Introduction……..….3

1.2 Définition du signal. ……..….3

1.3 Classification des signaux……..….4

1.3.1 Classification statistique…….....4

a. Signaux déterministes où certains ……..…4

b. Signaux aléatoires……..…….5

1.3.2 Classifications énergétiques……..5

a. Signaux à énergie finie…….....5

b. Signaux à puissance moyenne finie……..5

1.3.3 Classifications à variation temporelle……...…6

a. Signaux à variation temporelle continue-discrète……...…6

b. Les signaux à variation temporelle discrète…….....…7

1.3.4 Signaux périodiques et signaux apériodiques……..…7

1.3.5 Classification spectrale……...……..…7

1.3.6 Causalité……..…8

1.3.7 Signaux élémentaires……..…8

1.4 Numérisation des Signaux……...9

1.4.1 Echantillonnage……9

1.4.2 Quantification……...…9

1.4.3 Théorème de Nyquist-Shannon et limites ……..…10

1.5 Corrélation des signaux……...……11

1.5.1 Produit de convolution……11

1.5.2 Fonction de corrélation……..….11

1.6 Coefficient de corrélation……...…..13

1.6.1 Analyse graphique…….……14

a. Relation linéaire……14

b. Relation non linéaire……..14

1.6.2 Notions statistiques……....15

a. Densité de probabilité d'une fonction aléatoire en temps continu……..…15

b. Indépendance……..…..16
c. Moyenne…….…….16
d. Espérance mathématique……..…16
e. La variance……...……16
f. L'écart type……...……16
g. Auto corrélation statistique……17
1.6.3 Covariance……..…18
a. Définition……..…18
b. Domaine de définition……19
c. Estimation……19
1.6.4 Coefficient de corrélation de Pearson……19
a. Définition……19
b. Propriétés de coefficient de corrélation……..……20
1.7 Conclusion…….……20

Chapitre2 :

Généralités sur la carte Arduino Méga 2560

2.1 Introduction……21
2.2 Définition du module Arduino……..21
2.3 Bref historique de l'Arduino……23
2.4 Les gammes de la carte Arduino……...24
2.5 Les avantages de l'Arduino……25
2.6 Généralité sur les microcontrôleurs……..……26
2.7 La carte Arduino Méga 2560……28
2.7.1 Description générale……..……28
2.7.2 Caractéristiques principales……..……30
2.8 Présentation du logiciel ……..…34
2.8.1 IDE Arduino……..……34
2.8.2 Langage Arduino……..……38
2.9 Conclusion……..…43

Chapitre 3

Conception et réalisation du boitier de corrélation

3.1 Introduction……..…...44
3.2 Programmation de la carte Arduino Méga 2560……..……44
3.3 La Réalisation pratique……..……54
3.1 Les composants utilisés……...…54
a La carte Arduino Méga 2560……..…..54
b Ecran LCD 20 x 4……..….54
c Un potentiomètre……..…..56
d 2 LED et 2 résistances……56
e Un bouton interrupteur…….56
f Alimentation……..……..57
3.2 La Réalisation sur la plaque d'essai……57
3.3 Tests expérimentaux…….……59
3.4 La réalisation du circuit imprimé……62
3.5 Conclusion……..……69

Conclusion générale ……..…….70

Références bibliographiques

Remerciements

Nous remercions tout d'abord ALLAH le tout puissant qui nous a donné la force de mener à bon terme ce modeste travail.

Nous tenons à remercier sincèrement toutes les personnes qui ont apporté leur contribution à l'aboutissement de ce projet.

Nous remercions notre promotrice M^elle CHENTIR pour son soutient et ses précieux conseils.

Nous remercions aussi tous les enseignants de département d'électronique pour les efforts consacré pour nous transmettre le savoir.

Nous remercions également les membres de jury M^r GUESSOUM et M^r BOUDISSA pour avoir accepté d'évaluer ce travail.

Notre reconnaissance s'adresse à toutes les personnes qui sont chères et ceux qui nous ont aidé de prés ou de loin.

Dédicace

Je dédie ce modeste travail a la femme qui a sacrifié sa vie à notre éducation

ma très chère mère

A l'homme qui signifie le mot qui m'encourage et me donne de la force de croire

mon très cher père

*A mon cher frère '**Amine**' et mes deux chères sœurs '**Nesrine** et '**Sara** ' que j'aime énormément.*

Sans oublier mes amies

et toute personne qui m'aime.

ILHEM

Liste des tableaux

Tableau 2.1. Caractéristiques Techniques d'Arduino Méga 2560……..…..29

Tableau 2.2. Opérateurs de comparaison et opérateurs logiques…….41

Tableau 2.3. Structures de contrôle……

Liste des figures

Figure 1.1. classification des signaux……4

Figure 1.2. Signal à variation temporelle……...6

Figure 1.3. Signal à variation temporelle discrète……..…7

Figure 1.4. Exemples de spectres de signaux à bande limitée……...…8

Figure 1.5. Echantillonnage etquantification……..….10

Figure 1.6. Densité de probabilité d'une fonction aléatoire en temps continu……...…..15

Figure 2.1. La carte Arduino Méga 2560……20

Figure 2.2. Schéma de l'Arduino Mega 2560 révision 3……21

Figure 2.3. Exemple de typon – carte Arduino……....22

Figure 2.4. Différent types de cartes Arduino…… .24

Figure 2.5. Architecture d'un Microcontrôleur…….. .26

Figure 2.6 .Microcontrôleurs ATMega……....27

Figure 2.7 .Les microcontrôleurs ATMEL ATMEGA modèle AVR……..28

Figure 2.8. Constitution de la carte Arduino méga 2560……30

Figure 2.9. Alimenter l'Arduino secteur ……....31

Figure 2.10. Fonction d'un Compilateur……..35

Figure 2.11. L'interface d'IDE Arduino…….....35

Figure 2.12. La barre d'outils rapide…… .36

Figure 2.13. Fenêtre de Moniteur série…….. .37

Figure 2.14. Téléverser le programme dans la carte Arduino……..……38

Figure 2.15. Exe Tableau 2.2. Opérateurs de comparaison et opérateurs logiques…….……39

Figure 3.1. Carte connectée et alimentée……..……...44

Figure 3.2. Sélection de type de la carte…….....45

Figure 3.3. Sélection de port série……...46

Figure 3.4. Ouvrir un exemple existant dans IDE Arduino……47
Figure 3.5. Montage de l'exemple……47
Figure 3.6. Téléverser l'exemple dans la carte Arduino……48
Figure 3.7. LED orange sur la carte qui clignote……49
Figure 3.8. Organigramme……51
Figure 3.9. Programme……53
Figure 3.10. Afficheur LCD 20x 4……55
Figure 3.11. Les broches de l'écran LCD……56
Figure 3.12. Support *piles avec connecteur jack*……57
Figure 3.13. Schéma de circuit sur logiciel Fritzing……58
Figure 3.14. La réalisation sur la platine d'essai……58
Figure 3.15. Déclaration du tableau……59
Figure 3.16. Exemple de signaux injectés aux entrées d'Arduino……60
Figure 3.17. Exemple de signal sinusoïdal injecté obtenue sur Excel……60
Figure 3.18. Résultat du programme en cas d'injection de 2 signaux différents……61
Figure 3.19. Résultat du programme en cas d'injection de même signal……61
Figure 3.20. La carte Arduino introuvable dans Proteus……63
Figure 3.21. Intégration des fichiers de la bibliothèque Arduino dans Proteus……63
Figure 3.22. La carte arduino méga dans la bibliothèque du proteus……64
Figure 3.23. Schéma électronique de notre montage sur Proteus ISIS……65
Figure 3.24. Typon avec composants développé sur Proteus ARES……66
Figure 3.25. Visualisation 3D de deux faces cuivré……67
Figure 3.26. *Typon positif de la première face cuivrée* ……68
Figure 3.27. Typon positif de la deuxième face cuivrée……6

1.1 Introduction

Le signal correspond à l'information relative à une grandeur physique qui évolue dans le temps. La plupart des grandeurs physiques peuvent provenir de sources très diverses, mais la plupart sont des signaux électriques ou devenus électriques à l'aide de capteurs et transducteurs (microphones, rétines, senseurs thermiques, optiques, de pression, de position, de vitesse, d'accélération et en général de toutes les grandeurs physiques et chimiques). Alors pour développer et étudie les techniques de traitement, d'analyse et d'interprétation des signaux on fait appel au traitement du signal.

Traiter un signal, c'est essentiellement en extraire l'information que l'on juge utile. Bien que cette discipline trouve son origine dans les sciences de l'ingénieur (particulièrement l'électronique et l'automatique), elle fait aujourd'hui largement appel à de nombreux domaines mathématiques, tel que la probabilité et la statistique.

En fait, le signal matérialise le voyage d'une information, les propriétés statistiques du signal x(t) sont différentes de celles du signal y(t), mais peuvent s'en déduire, on peut alors distinguer un certain nombre d'opérations tel que la prédiction, l'identification, la classification, et la fonction de corrélation. Dans de nombreuses applications du traitement du signal il est nécessaire de comparer des signaux entre eux, la méthode permettant de résoudre ce problème est la corrélation, qui donne une quantité liée à la similitude entre les signaux permettant de mesurer la "ressemblance" entre deux signaux.

1.2 Définition du signal

On désigne par signal l'information relative à une grandeur physique On distingue essentiellement les signaux analogiques d'une part, qui sont produits par divers capteurs, amplificateurs, convertisseurs numérique-analogique, des signaux numériques d'autre part, issus d'ordinateurs, de terminaux, de la lecture d'un support numérique ou d'une numérisation par un convertisseur analogique-numérique [1].

Le traitement peut être fait, sans numériser les signaux, par des circuits électroniques analogiques ou aussi des systèmes optiques (traitement du signal optique). Il est de plus en plus souvent réalisé par traitement numérique du signal, à l'aide d'ordinateurs, de microprocesseurs embarqués, de microprocesseurs spécialisés nommés DSP, de circuits reconfigurables (FPGA) ou de composants numériques dédiés (ASIC) [2] .

On distingue différentes propriétés des signaux. Ces propriétés seront très importantes pour la classification de signaux et, en conséquence, leur modélisation, les opérations possibles, la définition de mesures adaptées, etc… [3].

1.3 Classification des signaux

Lors de l'enregistrement d'un signal provoqué ou naturel, le résultat peut dépendre des conditions dans lesquelles cet enregistrement est fait et toute prédiction préalable pour avoir une version proche serait vaine. Ce sont des conditions qu'on appelle « le hasard » qui conduit à séparer les signaux selon différentes approches Fig I-1 [4].

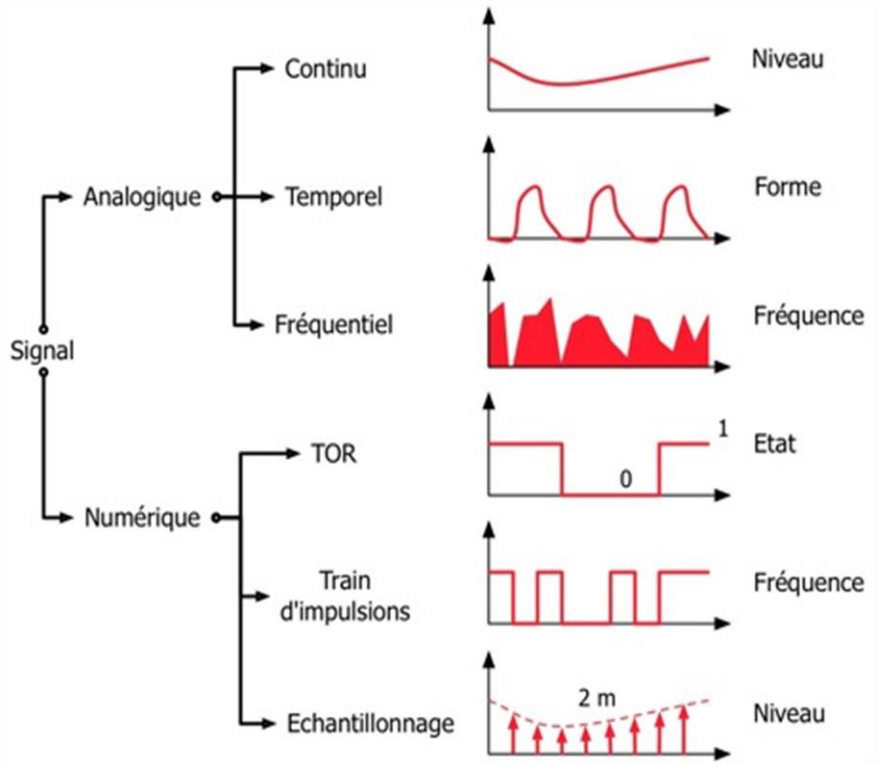


Figure 1.1 classification des signaux[3].

1.3.1 Classification statistique

Chaque signal de cet ensemble pouvant avoir une forme spécifique. De plus, chaque signal a une certaine probabilité de se produire ou non [5].

1.3.1.1 Signaux déterministes ou certains

Un signal x(t) est un signal déterministe pour lequel il n'y a aucune incertitude sur sa valeur: il peut donc être modélisé par une fonction du temps.

Un signal x(t) est certain (ou déterministe) s'il peut être décrit par un modèle mathématique [5].

1.3.1.2 Signaux aléatoires

Un signal x(t) est un signal aléatoire pour lequel avant qu'il se produise, il y a une certaine incertitude. En fait, c'est un signal qui appartient à un ensemble de signaux (appelé processus aléatoire) [5].

1.3.2 Classifications énergétiques

La classification des signaux en énergie finie ou en puissance finie est mutuellement exclusive : un signal à énergie finie est un signal à puissance nulle tandis qu'un signal à puissance finie a une énergie infinie [5].

1.3.2.1 Signaux à énergie finie

Caractériser les signaux par leur énergie totale, restreint l'étude à l'espace L_2 : en effet, tout signal admet une énergie totale non nulle, finie ou bien infinie. L'espace des « intéressants », étant plus large la puissance moyenne totale nulle s'avère être un outil lus adéquat pour leur classification [5].

Un signal x(t) est à énergie finie si l'intégrale suivante existe,

$$W_x = \int_{-\infty}^{+\infty} |x(t)^2|\, dt \qquad \text{I-1}$$

$$\text{Et} \qquad \int_{-\infty}^{+\infty} |x(t)^2|\, dt \; < \; +\infty \qquad \text{I-2}$$

1.3.2.2 Signaux à puissance moyenne finie

La puissance moyenne d'un signal périodique $x(n)$ de période fondamentale N est définie par l'équation suivante

$$P = \frac{1}{N} \sum_{n=0}^{N-1} x^2\,[n] \qquad \text{I-3}$$

On dira qu'un signal est à énergie finie, si et seulement si, son énergie totale est bornée, soit :

$$0 < E_{\infty} < \infty \qquad \text{I-4}$$

On dira qu'un signal est à puissance finie, si et seulement si, sa puissance moyenne est bornée :

$$0 < P_{\infty} < \infty \qquad \text{I-5}$$

1.3.3 Classifications à variation temporelle

1.3.3.1 Signaux à variation temporelle continue-discrète

Les signaux à variation temporelle continue sont des fonctions d'une ou plusieurs variables continues (définies dans un espace continu, par exemple l'ensemble des nombres réels entre$-\infty \; et \; +\infty$, ou encore entre -1 et +1), comme représenté sur la figure 1.2. La variable .t. et les parenthèses ( ) sont utilisées pour décrire une telle variation. Cette variable représente typiquement le temps, mais elle peut aussi représenter une température, une altitude……etc. [2].

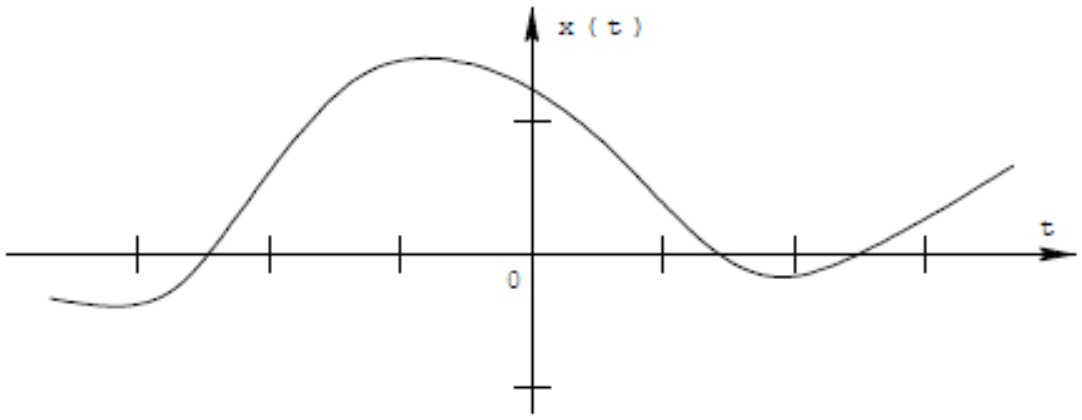


Figure 1.2 Signal à variation temporelle [2].

1.3.3.2 Les signaux à variation temporelle discrète

Sont des fonctions d'une ou plusieurs variables discrètes (définies pour certaines valeurs seulement et non pour un continuum de valeurs : l'ensemble des nombres entiers par exemple entre$-\infty\ et\ +\infty$, ou encore entre 0 et 40 par exemple), comme représenté sur la figure 1.3.

La variable *'n'* et les crochets [].sont utilisés pour décrire une telle variation. Cette variable représente en général le temps, mais elle peut aussi représenter d'autres quantités [2].

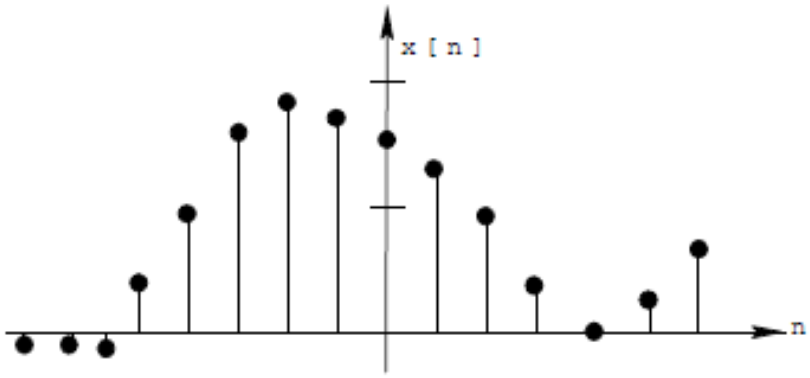


Figure 1.3 Signal à variation temporelle discrète [2].

1.3.4 Signaux périodiques et signaux apériodiques

Un signal périodique x(t) est une fonction du temps qui satisfait la condition

$$x(t) = x(t+T) \quad \text{pour tout } t \qquad \text{I-6}$$

Où T est la période qui définit donc la durée d'un cycle complet de $x(t)$ [3].

Un signal $x(t)$ pour lequel, il n'existe pas de valeur de T vérifiant $x(t) = x(t+T)$ pour tout t est appelé un signal apériodique ou signal non périodique [3].

1.3.5 Classification spectrale

L'analyse spectrale, par transformée de Fourier (TF), conduit à considérer le spectre des signaux, c'est-à-dire leur représentation dans le domaine fréquentiel, comme une représentation duale, équivalente d'un point de vue de l'information contenue [4].

On appelle largeur de bande,$B = f_2 - f_1$ le domaine des fréquences où le spectre a des valeurs non nulles.

Un signal dont le spectre est nul en dehors d'une bande de fréquence donnée est appelé signal à bande limitée, ou signal à spectre à support borné. La figure 3.1 donne quelques exemples de signaux à bande limitée [4] Fig I-4.

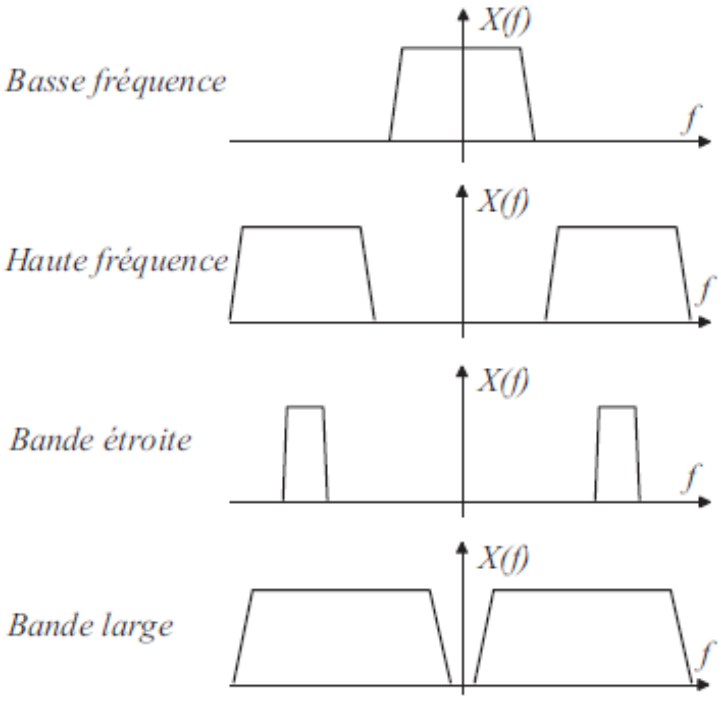


Figure 1.4 Exemples de spectres de signaux à bande limitée [3].

1.3.6 Causalité

Un signal est causal s'il est nul pour toute valeur négative de t [1].

On peut montrer qu'un signal réel causal est donc tel que

$$x_i(t) = x_p(t)sgn(t) \qquad \text{I-7}$$

1.3.7 Signaux élémentaires

Plusieurs signaux jouent un rôle prépondérant dans l'étude des signaux et des systèmes ; les signaux exponentiels et sinusoïdaux, la fonction échelon, l'impulsion de Dirac et la fonction rampe servent de briques élémentaires pour construire des signaux plus complexes. Il

est aussi important de les étudier pour eux-mêmes, car ils servent de modèles à des signaux physiques prenant naissance dans la nature. C'est pourquoi nous allons décrire ces signaux élémentaires, les uns après les autres [1].

1.4 Numérisation des Signaux

La numérisation est la conversion des informations d'un support (texte, image, audio, vidéo) ou d'un signal électrique en données numériques que des dispositifs informatiques ou d'électronique numérique pourront traiter [3].

1.4.1 Echantillonnage

Si l'on veut traiter un signal par voie numérique à l'aide d'un calculateur, il faut le représenter au préalable par une suite de valeurs numériques ponctuelles prélevées régulièrement ou irrégulièrement. Un tel prélèvement est appelé échantillonnage. Un échantillonnage représente un signal par une suite de valeurs ponctuelles voir figure I-5.

La représentation numérique des échantillons requiert une opération complémentaire de quantification et de codage, L'ensemble réalise une fonction de conversion analogique-numérique A/N, (Dite Analog to Digital ou A/D en Anglais) [3].

1.4.2 Quantification

En traitement des signaux, la quantification est le procédé qui permet d'approcher un signal continu par les valeurs d'un ensemble discret d'assez petite taille. On parle aussi de quantification pour approcher un signal à valeurs dans un ensemble discret de grande taille par un ensemble plus restreint [3].

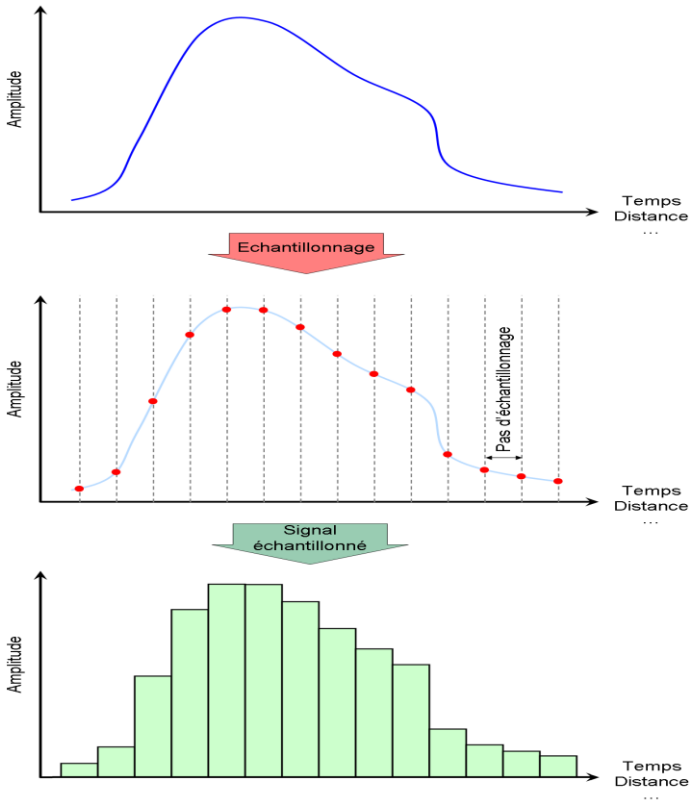


Figure 1.5 Echantillonnage et quantification [2].

1.4.3 Théorème de Nyquist-Shannon et limites

Le théorème de Nyquist-Shannon stipule que pour échantillonner correctement un signal, la fréquence d'échantillonnage doit être supérieure au double de la fréquence maximale du signal:

$$\frac{1}{T} > 2f_M \qquad \text{Où} \quad \begin{cases} \frac{1}{T} est\ la\ fréquence\ d'échantillonnage \\ f_M est\ la\ fréquence\ maximal\ contenu\ dans\ le\ signal \end{cases}$$

Cependant, ce théorème ne peut s'appliquer que dans le cas des signaux à bande passante majorée par f_M . Dans la réalité, ce n'est jamais le cas. Les fréquences supérieures à f_M introduisent un phénomène appelé recouvrement [3].

1.5 Corrélation des signaux

Dans de nombreuses applications du traitement du signal il est nécessaire de comparer des signaux entre eux, la méthode permettant de résoudre ce problème est la corrélation statistique, permettant de mesurer la "ressemblance" entre deux signaux.

1.5.1 Produit de convolution

- **Pour les signaux analogiques**

Soit deux signaux $h(t)$ et $g(t)$, on appel produit de convolution entre $h(t)$ et $g(t)$ le signal $s(t)$ définie par :

$$s(t) = h(t) * g(t) = \int_{-\infty}^{+\infty} h(\tau)\, g(t-\tau) d\sigma \qquad \text{I-8}$$

La sommation sous l'intégrale s'effectue sur la variable σ ce qui montre que le signal ainsi obtenu est une fonction de t et non pas un nombre comme le c'est le cas lorsque l'on effectue un produit scalaire [6].

Pour calculer un produit de convolution, il faut conserver le premier signal, trouver le symétrique du seconde par rapport à l'axe des ordonnées puis décaler ce signal du temps t, multiplier les deux signaux obtenus et finalement intégrer le résultat[6].

- **Pour les signaux numériques**

Soit deux signaux h et g, son numérique et définies par les suites $\{h(n)\}\, et\, \{g(n)\}$, le produit de convolution s'écrit alors :

$$s(t) = h(t) * g(t) = \sum_{k=0}^{N-1} h(k)\, g(n-k) \qquad \text{I-9}$$

Les signaux *h* et g tous les deux sont définis sur l'intervalle [0, N-1] [6].

1.5.2 Fonction de corrélation

La fonction de corrélation donne une quantité liée à la similitude entre les signaux permettant de mesurer la "ressemblance" entre deux signaux [6].

- **Pour les Signaux analogique**

Soient deux signaux analogiques $s(t)$ et $r(t)$ d'énergie finie ; on appel fonction de corrélation entre ces deux signaux, la fonction de τ définie par :

$$C_{sr}(\tau) = s(t) \otimes r(t) = \int_{-\infty}^{+\infty} s(t) r^*(t-\tau) dt = \int_{-\infty}^{+\infty} s(t+\tau) r^*(t) dt \qquad \text{I-10}$$

Cette fonction s'appelle aussi fonction d'intercorrélation entre les signaux $s(t)$ et $r(t)$, physiquement la fonction de corrélation est obtenue en décalant l'un des signaux, en multipliant le signal décalé par l'autre signal et puis en intégrant le produit obtenu[6].

- Si le signal $s(t) = r(t)$ quel que soit t alors on obtient la fonction d'autocorrélation du signal soit :

$$C_{ss}(\tau) = \int_{-\infty}^{+\infty} s(t) s^*(t-\tau) dt = s(t) \otimes s(t) \qquad \text{I-11}$$

Dans le cas particulier ou τ est nul, la fonction d'autocorrélation du signal donne :

$$C_{ss}(0) = \int_{-\infty}^{+\infty} s(t) s^*(t-\tau) dt \qquad \text{I-12}$$

Qui n'est rien d'autres que l'énergie contenue dans le signal. On peut démontrer que la fonction d'autocorrélation vérifie :

$$C_{ss}(\tau) < C_{ss}(0) \qquad \text{I-13}$$

- Si les signaux ne sont pas à énergie finie, on définit la fonction de corrélation sur l'intervalle T par :

$$C_{sr}(\tau) = \lim_{T \to \infty} \frac{1}{T} \int_{t}^{t+T} s(t) r^*(t-\tau) dt \qquad \text{I-14}$$

De même pour les signaux périodiques on définit la fonction de corrélation par :

$$C_{sr}(\tau) = \int_{t}^{t+T} s(t) r^*(t-\tau) dt \qquad \text{I-15}$$

- **Pour les Signaux numérique**

Pour les signaux numériques (que l'on supposera réels) définis par $\{s(n)\}$ et $\{r(n)\}$, on définit la fonction de corrélation par :

$$C_{sr}(n) = \sum_{m=0}^{N-1} s(m)r(m-n) \quad avec\ n \in [0, M+N-1] \qquad \text{I-16}$$

$$C_{sr}(n) = \sum_{m=0}^{N-1} s(m+n)r(m) \qquad \text{I-17}$$

La corrélation apparait comme la convolution du premier signal avec le conjugué du seconde signal retourné à l' instant τ donné

Si les deux signaux sont identiques, on voit que la fonction d'autocorrélation est donné par :

$$C_{ss}(\tau) = s(\sigma)s^*(-\tau) \qquad \text{I-18}$$

Non commutativité de la corrélation

La fonction de corrélation est une fonction non-commutative qui vérifie :

$$C_{sr}(\tau) = C_{sr}^*(-\tau) \qquad \text{I-19}$$

En effet

$$C_{rs}^*(-\tau) = [r(-\tau) * s^*(-\tau)]^* = r^*(-\tau) * s(\tau) = s(\tau) * r^*(-\tau) = C_{sr}(\tau) \qquad \text{I-20}$$

1.6 Coefficient de corrélation

Le coefficient de corrélation est un indice statistique qui exprime l'intensité et le sens (positif ou négatif) de la relation linéaire entre deux variables quantitatives. C'est une mesure de la liaison linéaire, c'est à dire de la capacité de prédire une variable x par une autre y à l'aide d'un modèle linéaire.

Il permet de mesurer l'intensité de la liaison entre deux caractères quantitatifs. En revanche, ce coefficient est nul (r = 0) lorsqu'il n'y a pas de relation linéaire entre les variables (ce qui n'exclut pas l'existence d'une relation autre que linéaire). Par ailleurs, le coefficient est de signe positif si la relation est positive (directe, croissante) et de signe négatif si la relation est négative (inverse, décroissante).

Ce coefficient varie entre -1 et +1 ; l'intensité de la relation linéaire sera donc d'autant plus forte que la valeur du coefficient est proche de +1 ou de - 1, et d'autant plus faible qu'elle est proche de 0.

- une valeur proche de +1 montre une forte liaison entre les deux caractères. La relation linéaire est ici croissante (c'est-à-dire que les variables varient dans le même sens);
- une valeur proche de -1 montre également une forte liaison mais la relation linéaire entre les deux caractères est décroissante (les variables varient dans le sens contraire);
- une valeur proche de 0 montre une absence de relation linéaire entre les deux caractères. [8]

1.6.1 Analyse graphique

1.6.1.1 Relation linéaire

Une relation est linéaire lorsque le nuage de points paraît étiré le long d'une droite.

Relation positive : lorsque les coordonnées verticales Y tendent à augmenter en même temps que les coordonnées horizontales X.

Relation négative : si les coordonnées verticales Y tendent à diminuer quand les coordonnées horizontales X augmentent [8].

1.6.1.2 Relation non linéaire

Une relation est non linéaire lorsque le nuage de points paraît étiré aléatoire.

- Liaison monotone positive non-linéaire : *X* et *Y* évoluent dans le même sens, mais la pente est différente selon le niveau de *X*.
- Liaison non-linéaire non-monotone : Il y a une relation fonctionnelle (de type sinusoïdale ici) entre *X* et *Y*. Mais la relation n'est pas monotone, *Y* peut augmenter ou diminuer selon la valeur de *X*.
- Absence de liaison. La valeur de X ne donne indication sur la valeur de Y, et inversement. L'autre situation caractéristique est que X (ou Y) est constant quelle que soit la valeur de la seconde variable dont le schéma est rappelé en Fig.I.6 [8].

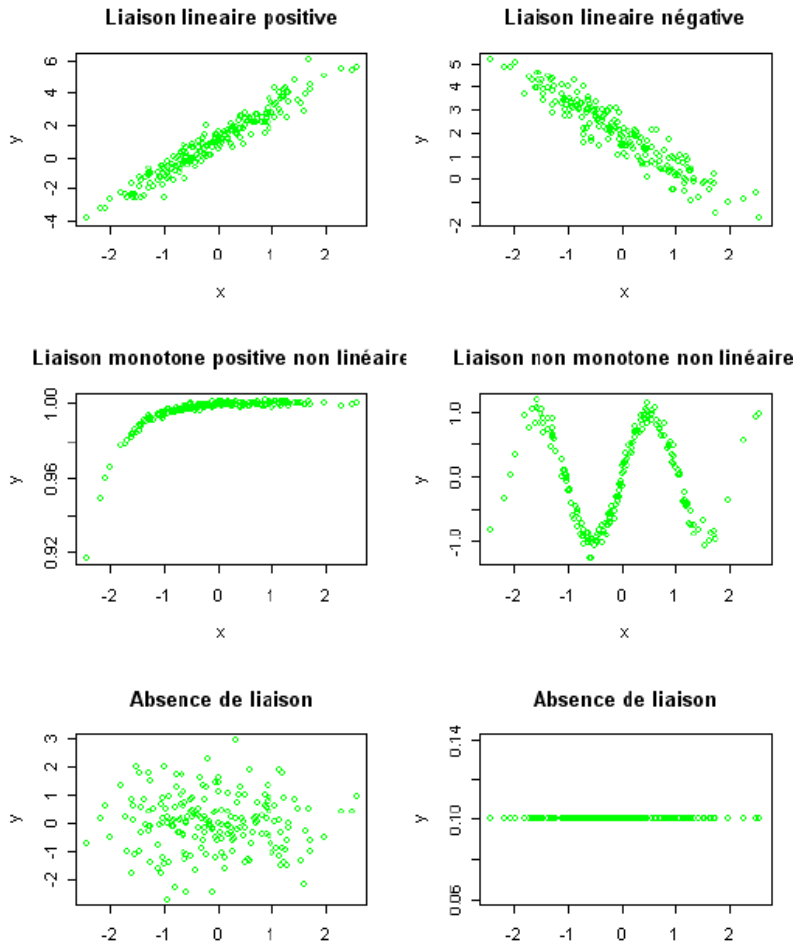


Figure 1.6 Densité de probabilité d'une fonction aléatoire en temps continu [8].

1.6.2 Notions statistiques

En probabilités et en statistiques, étudier la corrélation entre deux ou plusieurs variables aléatoires ou statistiques numériques, c'est étudier l'intensité de la liaison qui peut exister entre ces variables.

1.6.2.1 Densité de probabilité d'une fonction aléatoire en temps continu

Soit une fonction aléatoire $X(t)$. En considérant l'instant particulier $t = t_1$, on obtient une valeur notée $X(t_1)$. On note la densité de probabilité de cette variable aléatoire par $P_X[x(t_1)]$ pour rappeler la dépendance à l'égard de l'instant considéré. En considérant des instants $t_1, \ldots, t_n$, on obtient une variable aléatoire à n dimensions. Sa densité de probabilité est notée $P_X[x(t_1), \ldots, x(t_n)]$. Cette densité est appelée densité de probabilité du nième ordre de la fonction aléatoire. A partir d'une densité de probabilité d'un certain ordre, on en déduit

celles d'ordres inférieurs par passage aux densités marginales. Les notions de la fonction aléatoire en temps discret sont tout à fait semblables à celles exposées pour des fonctions aléatoires en temps continu. On utilise la notation $P_X[x(n)]$[7].

1.6.2.2 Indépendance

On dit qu'une fonction est à valeurs indépendantes si

$$P_X[x(t_1),\ldots,x(t_n)] = P_X[x(t_1)] \times \ldots \times P_X[x(t_n)] \quad \text{I-21}$$

A tout instant, le futur est indépendant du présent et du passé puisque

$$P_X[x(t_n)|x(t_1),\ldots,x(t_{n-1})] = P_X[x(t_n)] \quad \text{I-22}$$

1.6.2.3 Moyenne

On appelle moyenne de la fonction aléatoire en temps continu $X(t)$, la fonction de t qui pour toute valeur de t donne la moyenne de la variable aléatoire obtenue à cet instant, soit

$$m_X(t) = E[X(t)] = \int_{-\infty}^{+\infty} x\, P_X[x(t)]dx \quad \text{I-23}$$

1.6.2.4 Espérance mathématique

De façon générale, l'espérance mathématique correspond à la moyenne des valeurs numériques pondérées avec la probabilité que ce réalise chacune de ces valeurs.

$$E(X) = \sum_{i=1}^{n} p_i x_i = p_1x_1 + p_2x_2 + p_3x_3 \ldots + p_nx_n \quad \text{I-24}$$

Où $p_1, p_2 \ldots, p_n$ correspond à la probabilité que les événements 1, 2,3,…, n se réalisent

Et $m_1, m_2 \ldots, m_n$ correspond à la valeur numérique des événements 1, 2,3,…, n [7].

1.6.2.5 La variance

De façon similaire, la variance d'une fonction aléatoire scalaire est une fonction de t définie par [9]:

$$V_{XY} = E\{[X - m_x][X(t) - m_x]\} \quad \text{I-25}$$

1.6.2.6 L'écart type

L'écart type d'une fonction aléatoire est définie par

$$\sigma_{XY} = \sqrt{V_{XY}} \quad \text{I-26}$$

1.6.2.7 Auto corrélation statistique

On note de façon générale $X_i = x(t_i)$ la VA associer au processus aléatoire $x(t)$ pour $t = t_i$

Cas générale

$$R_{XX}(t_1, t_2) = E[X_1 X_2] = \iint x_1 x_2 p(x_1, x_2) dx_1 dx_2 \quad \text{I-27}$$

Cas d'un signal stationnaire, en notant $\tau = t_1 - t_2$

$$R_{XX}(\tau) = E[X(t)X(t-\tau)] = \iint x_1 x_2 p(x_1, x_2) dx_1 dx_2 \quad \text{I-28}$$

Les VA sont orthogonales ssi $R_{XX}(\tau) = 0$

Auto-covariance statistique

C'est l'auto-corrélation des processus aléatoires centrés. On la note $C_{XX}(\tau)$

Cas générale

$$C_{XX}(t_1, t_2) = E[(X_1 - E(X_1))(X_2 - E(X_2)) \quad \text{I-29}$$

$$= \iint \big(x_1 - E(X_1)\big)\big(x_2 - E(X_2)\big) p(x_1, x_2) dx_1 x_2 \quad \text{I-30}$$

$$= R_{XX}(t_1, t_2) - E(X_1)E(X_2) \quad \text{I-31}$$

Cas d'un signal stationnaire, on note $\tau = t_1 - t_2$

$$C_{XX}(\tau) = E[(X(t) - E(X))(X(t-\tau) - E(X))] \quad \text{I-32}$$

$$= \iint x_1 x_2 p(x_1, x_2) dx_1 dx_2 \quad \text{I-33}$$

$$= R_{XX}(\tau) - E(X)^2 \quad \text{I-34}$$

Les variables sont non corrélé ssi $C_{XX}(\tau) = 0$

1.6.3 Covariance

L'objectif de la covariance est de quantifier la liaison entre deux variables $X\ et\ Y$ de manière à mettre en évidence le sens de la liaison et son intensité [9].

1.6.3.1 Définition

La covariance est égale à l'espérance du produit des variables centrées.

$$cov(X,Y) = E\{[X - E(X)][Y - E(Y)]\} \quad \text{I-35}$$

On peut aussi l'écrire comme l'espérance du produit des variables, moins le produit des espérances [10].

$$cov(X,Y) = E(XY)E(X)E(Y) \quad \text{I-36}$$

$$cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y}) \quad \text{I-37}$$

Signification

La covariance mesure la tendance des deux variables à être simultanément au-dessus ou en dessous de leurs espérances respectives. Elle modélise une liaison monotone [8].

$COV(X,Y) > 0$: La relation est positive c.-à-d. lorsque X est plus grand que son espérance, Y a tendance à l'être également ;

$COV(X,Y) = 0$: Absence de relation monotone ;

$COV(X,Y) < 0$: La liaison est négative c.-à-d. lorsque X est plus grande que son espérance, Y a tendance à être plus petit que sa propre espérance.

La covariance d'une variable avec elle-même est la variance, la relation est toujours positive. En effet,

$$COV(X,Y) = E\{[X - E(X)][X - E(X)]\}$$

$$= E\left\{\left[[X - E(X)]\right]^2\right\}$$

$$= V(X) \quad \text{I-38}$$

> 0

1.6.3.2 Domaine de définition

La covariance est définie dans l'ensemble des réels Il permet de se rendre compte du sens de la liaison. Plus sa valeur est élevée (en valeur absolue), plus la liaison est forte. Mais nous ne savons pas quelle est la limite. Nous ne pouvons pas non plus comparer la covariance d'une variable X avec deux autres variables Y et. Dans la pratique, nous préférerons donc une mesure normalisée : le coefficient de corrélation répond à ces spécifications [11].

1.6.3.3 Estimation

Sur un échantillon de taille *n*, la covariance empirique est définie de la manière suivante :

$$\hat{s}_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y}) \qquad \text{I-39}$$

On montre que c'est un estimateur biaisé de la covariance, en effet

$$E\left[\hat{s}_{xy}\right] = \frac{n-1}{n}COV(X,Y) \qquad \text{I-40}$$

L'estimateur sans biais de la covariance s'écrit par conséquent :

$$\widehat{COV}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y}) \qquad \text{I-41}$$

1.6.4 Coefficient de corrélation de Pearson

1.6.4.1 Définition

Le coefficient de corrélation linéaire simple, dit de Bravais-Pearson (ou de Pearson), est une normalisation de la covariance par le produit des écarts-type des variables [8].

$$r = \frac{cov(X,Y)}{\sqrt{var(X).var(Y)}} \qquad \text{I-42}$$

$$r = \frac{COV(X,Y)}{\sigma_x \sigma_y} \qquad \text{I-43}$$

$$r_{xy} = \frac{\sum(x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \times \sqrt{\sum_i (y_i - \bar{y})^2}}$$

1.6.4.2 Propriétés de coefficient de corrélation

1. Il est de même signe que la covariance, avec les mêmes interprétations.
2. X et Y sont indépendants, alors $r = 0$. La réciproque est fausse, sauf cas particulier que nous précisons maintenant.
3. Lorsque le couple de variables $(X; Y)$ suit une loi normale bi-variée, et uniquement dans ce cas, nous avons l'équivalence $r = 0$, X et Y sont indépendants. Dans ce cas, le coefficient de corrélation caractérise parfaitement la liaison entre X et Y. Dans les autres cas, le coefficient de corrélation constitue une mesure parmi les autres de l'intensité de la corrélation
4. Le coefficient de corrélation constitue une mesure de l'intensité de liaison linéaire entre 2 variables. Il peut être égal à zéro alors qu'il existe une liaison fonctionnelle entre les variables. C'est le cas lorsque la liaison est non monotone.
5. La corrélation d'une variable avec elle-même est $r_{xx} = 1$ [11].

1.7 Conclusion

Dans ce chapitre nous avons décrit les principaux outils mathématiques de traitement de signal et de la corrélation des signaux en revanche on a étudié aussi quelques notions de la statistique pour le calcul des coefficients de corrélation.

Chapitre 2

Généralités sur la carte

Arduino Méga 2560

2.1 Introduction

Aujourd'hui, l'électronique est de plus en plus remplacée par de l'électronique programmée. On parle aussi de système embarquée ou d'informatique embarquée. Son but est de simplifier les schémas électroniques et par conséquent réduire l'utilisation de composants électroniques, réduisant ainsi le coût de fabrication d'un produit. Il en résulte des systèmes plus complexes et performants pour un espace réduit [12].

Les cartes Arduino permettent un accès simple et peu couteux à l'informatique embarquée. De plus, elles sont entièrement libres de droit, autant sur l'aspect du code source (Open Source) que sur l'aspect matériel (Open Hardware). Ainsi, il est possible de refaire sa propre carte Arduino dans le but de l'améliorer ou d'enlever des fonctionnalités inutiles au projet [13].

2.2 Définition du module Arduino

Arduino est un circuit imprimé en matériel libre (voir figure 2.1), dont les plans sont publiés en licence libre(voir figure 2.2 et 2.3), sur lequel se trouve un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses comme la domotique : le contrôle des appareils domestiques, éclairage, chauffage, le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur, commander des appareils mobiles (modélisme).etc [16].

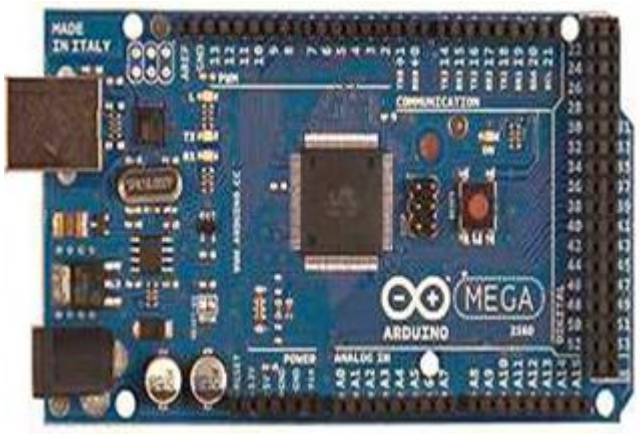


Figure 2.1. *La carte Arduino Méga 2560* [14].

C'est une plateforme basée sur une interface entrée/sortie simple. Il était destiné à l'origine principalement mais pas exclusivement à la programmation multimédia interactive en vue de spectacle ou d'animations artistiques. Arduino peut être utilisé pour construire des objets interactifs indépendants (prototypage rapide), ou bien peut être connecté à un ordinateur pour communiquer avec ses logiciels [16].

Le projet Arduino a reçu un titre honorifique à l'Ars Electronica 2006, dans la catégorie Digital Communities [16].

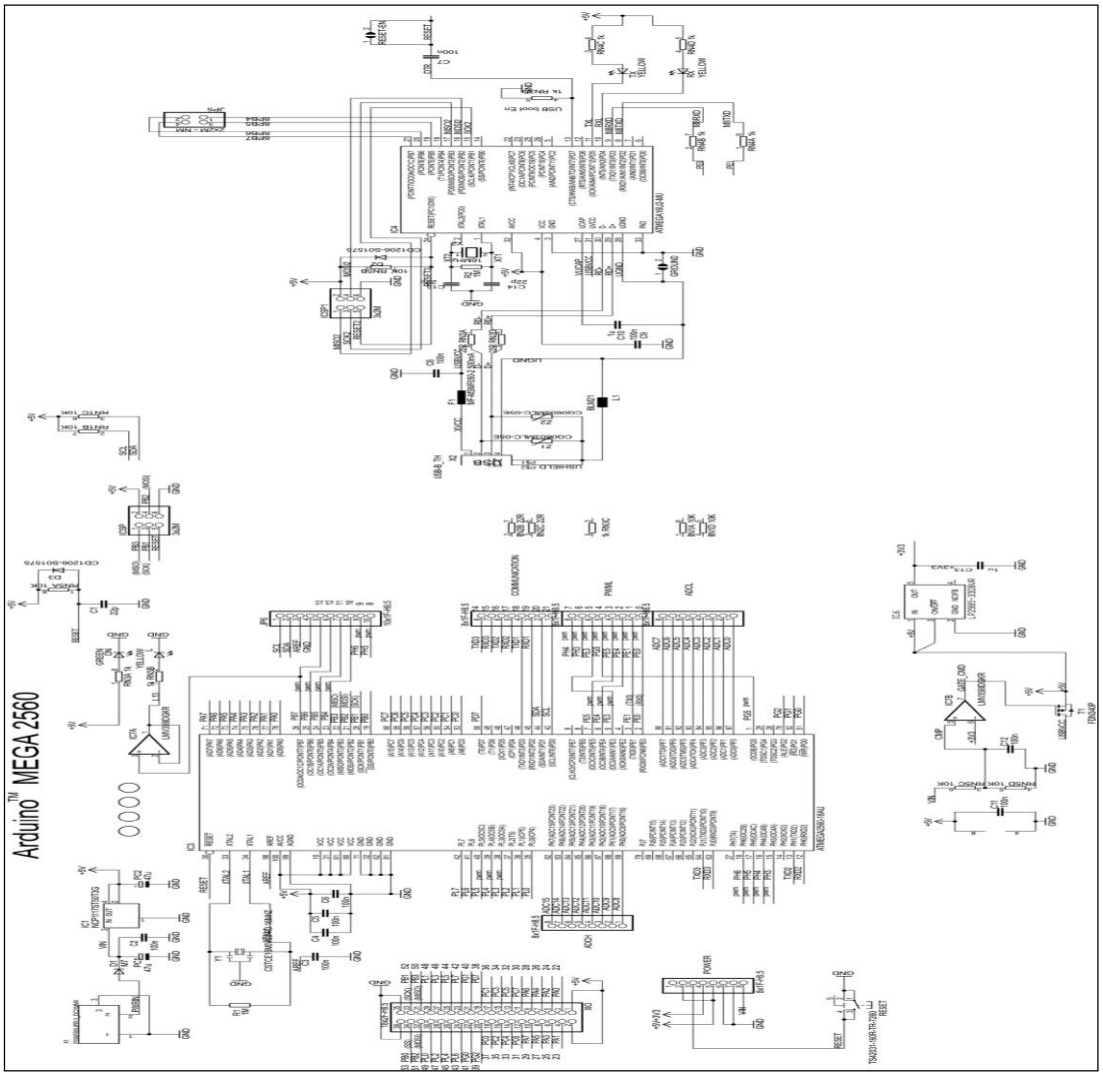


Figure 2.2. *Schéma de l'Arduino Mega 2560 révision 3* [20].

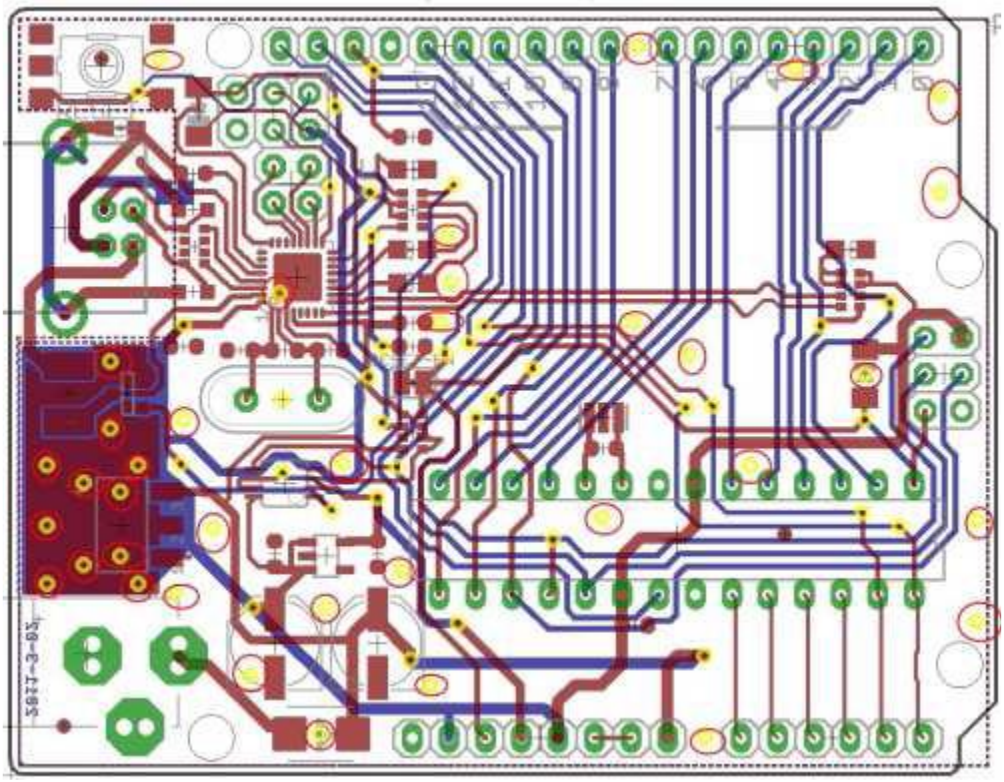


Figure 2.3. *Exemple de typon – carte Arduino* [14].

2.3 Bref historique de l'Arduino

En 2003, Hernando Barragan, pour sa thèse de fin d'études, avait entrepris le développement d'une carte électronique dénommée Wiring, accompagnée d'un environnement de programmation libre et ouvert. Pour ce travail, Hernando Barragan réutilisait les sources du projet Processing. Basée sur un langage de programmation facile d'accès et adaptée aux développements de projets de designers, la carte Wiring a donc inspiré le projet Arduino (2005).

Comme pour Wiring, l'objectif était d'arriver à un dispositif simple à utiliser, dont les coûts seraient peu élevés, les codes et les plans « libres » (c'est-à-dire dont les sources sont ouvertes et peuvent être modifiées, améliorées, distribuées par les utilisateurs eux-mêmes) et, enfin, « multi-plates-formes » (indépendant du système d'exploitation utilisé).

Arduino a été crée en 2005 dans l'institut de Design Interaction d'Ivrea en Italie, par une équipe de professeurs et d'étudiants. L'environnement Arduino est particulièrement adapté à la production artistique ainsi qu'au développement de conceptions qui peuvent trouver leurs réalisations dans la production industrielle [15].

Arduino est le nom d'un roi italien, personnage historique de la ville « Arduin d'Ivrée », ou encore un prénom italien masculin qui signifie « l'ami fort ».

2.4 Les gammes de la carte Arduino

Actuellement, il existe plus de 20 versions de module Arduino, nous citons quelques un afin d'éclaircir l'évaluation de ce produit scientifique et académique (voir figure 2.4) [24]:

- Le NG d'Arduino, avec une interface d'USB pour programmer et usage d'un ATmega8.
- L'Arduino Mini, une version miniature de l'Arduino en utilisant un microcontrôleur ATmega168.
- L'Arduino Nano, une petite carte programmé à l'aide du port USB, cette version utilise un microcontrôleur ATmega168 (ATmega328 pour une plus nouvelle version).
- Le NG d'Arduino plus, avec une interface d'USB pour programmer et usage d'un ATmega168.
- L'Arduino Bluetooth, avec une interface de Bluetooth pour programmer en utilisant un microcontrôleur ATmega168.
- L'Arduino Diecimila, avec une interface d'USB et utilise un microcontrôleur ATmega168.
- L'Arduino Duemilanove ("2009"), en utilisant un microcontrôleur l'ATmega168 (ATmega328 pour une plus nouvelle version).
- L'Arduino Mega, en utilisant un microcontrôleur ATmega1280.
- L'Arduino UNO, utilisations microcontrôleur ATmega328.

- L'Arduino Mega2560, utilisations un microcontrôleur ATmega2560, elle incorpore également le nouvel ATmega8U2 (ATmega16U2 dans le jeu de puces d'USB de révision 3).
- L'Arduino Leonardo, avec un microcontrôleur ATmega32U4.

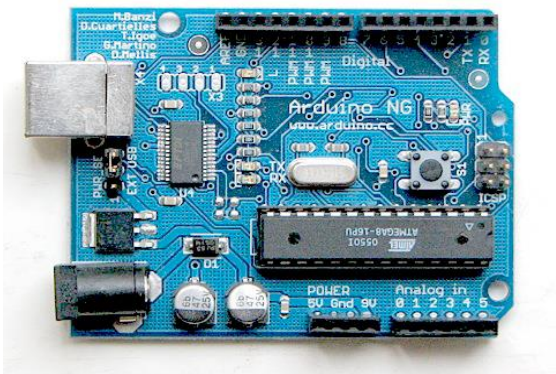
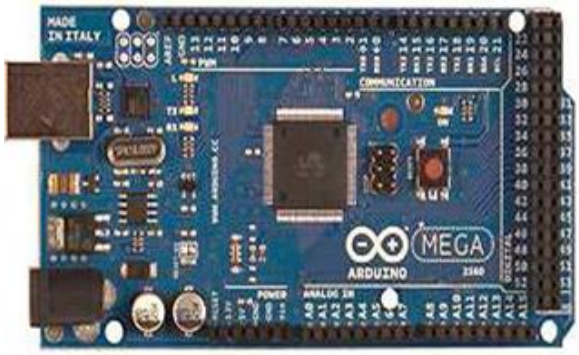
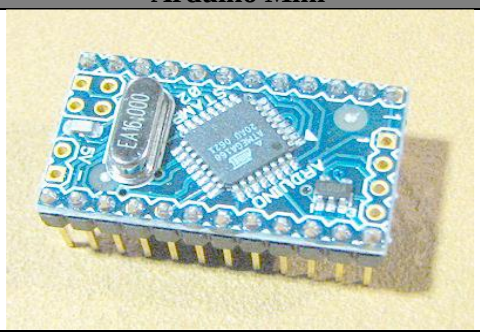
Arduino NG	Arduino Méga 2560
	
Arduino Duemilanove	**Arduino Uno**
	
Arduino Mini	**Arduino Nano**
	

Figure 2.4. *Différent types de cartes Arduino [13].*

2.5 Les avantages de l'Arduino

Il y a de nombreuses cartes électroniques qui possèdent des plateformes basées sur des microcontrôleurs disponibles pour l'électronique programmée. Tous ces outils prennent en charge les détails compliqués de la programmation et les intègrent dans une présentation facile à utiliser. De la même façon, le système Arduino simplifie la façon de travailler avec les microcontrôleurs tout en offrant à personnes intéressées plusieurs avantages cités comme suit [12] :

- **Le prix (réduits)** : les cartes Arduino sont relativement peu coûteuses comparativement aux autres plates-formes.

- **Multi plateforme** : le logiciel Arduino tourne sous les systèmes Windows, Macintosh et Linux. La plupart des systèmes à microcontrôleurs sont limités à Windows.

- **Un environnement de programmation clair et simple** : l'environnement de programmation Arduino (le logiciel Arduino IDE) est facile à utiliser pour les débutants, tout en étant assez flexible pour que les utilisateurs avancés puissent en tirer profit également.

- **Logiciel Open Source et extensible :** le logiciel Arduino est publié sous licence open source, disponible pour être complété par des programmateurs expérimentés.

- **Matériel Open source et extensible** : les cartes Arduino sont basées sur les Microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA 328, les schémas des modules sont publiés sous une licence créative Commons, et les concepteurs des circuits expérimentés peuvent réaliser leur propre version des cartes Arduino, en les complétant et en les améliorant. Même les utilisateurs relativement inexpérimentés peuvent fabriquer la version sur plaque d'essai de la carte Arduino, dont le but est de comprendre comment elle fonctionne pour économiser le coût.

2.6 Généralité sur les microcontrôleurs

Un microcontrôleur est une petite unité de calcul accompagné de mémoire, de ports d'entrée/sortie et de périphériques permettant d'interagir avec son environnement (voir figure 2.5). On peut comparer un micro contrôleurs à un ordinateur classique, mais avec une puissance de calcul considérablement plus faible [13].

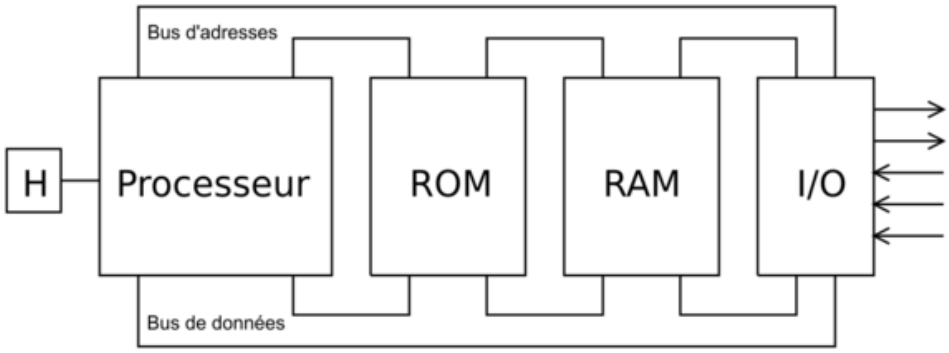


Figure 2.5. *Architecture d'un Microcontrôleur* [17].

L'intérêt des microcontrôleurs est leur coût très faible (parfois moins d'un Euro), leur consommation de courant très limitée (quelques dizaines de mA) et leur taille très réduite (un seul circuit intégré) (voir figure 2.6). Ils sont donc utilisés dans de très nombreuses applications [17].

Bien que des microcontrôleurs existent depuis les années 1970, ils se sont développés de manière spectaculaire depuis quelques années. Alors qu'il était encore complexe et coûteux de mettre en œuvre un microcontrôleur au début des années 2000, cette tâche est maintenant beaucoup plus simple et ne nécessite que du matériel très peu coûteux [17].

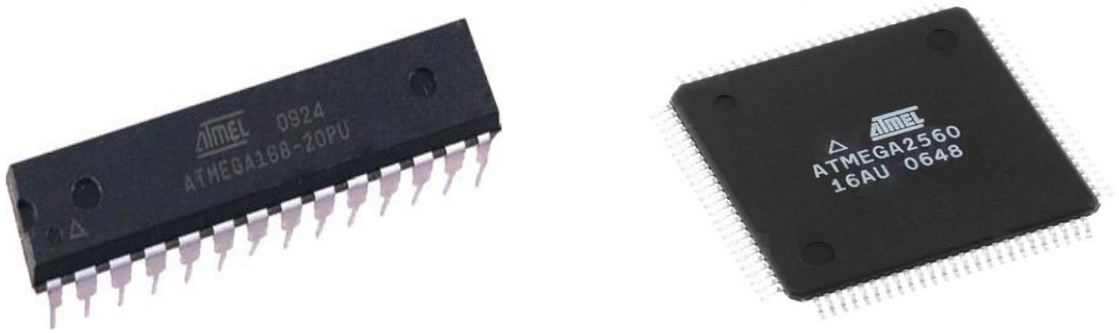


Figure 2.6. *Microcontrôleurs ATMega* [19].

Les microcontrôleurs sont donc devenus des composants électroniques incontournables dans les domaines de l'informatique embarquée, de l'automatique et de l'informatique industrielle. Ils permettent de réduire le nombre de composant et de simplifier la création de cartes électroniques [13].

Plusieurs fabricants proposent des microcontrôleurs (Microchip, Atmel, Texas-Instrument, NXP-FreeScale, ST-micro, Cypress, etc). Chaque fabricant propose souvent plusieurs familles de microcontrôleurs (PIC et dsPIC chez Microchip; AVR, AVR32 et ARM chez Atmel; MSP430, MSP432 chez Texas Intrument, etc) (voir figure 2.7). Chaque famille comporte souvent des dizaines de modèles, qui diffèrent principalement par les tailles mémoires (RAM et ROM) et le nombre de broches d'entrée-sortie [17].

Figure 2.7. *Les microcontrôleurs ATMEL ATMEGA modèle AVR* [18].

2.7 La carte Arduino Méga 2560

2.7.1 Description générale

La carte Arduino Mega 2560 est une carte à microcontrôleur basée sur un ATmega2560. Cette carte dispose (voir le tableau 2.1) [21] :

- de 54 broches numériques d'entrées/sorties (dont 14 peuvent être utilisées en sorties PWM (largeur d'impulsion modulée)),
- de 16 entrées analogiques,
- de 4 ports série,
- d'un quartz 16Mhz,
- d'une connexion USB,
- d'un connecteur d'alimentation jack,
- d'un connecteur ICSP,

- et d'un bouton de réinitialisation (reset).

Microcontrôleur	ATmega2560
Tension de fonctionnement	5V
Broches E/S numériques	54 (dont 14 disposent d'une sortie PWM)
Broches d'entrées analogiques	16
Intensité maxi disponible par broche E/S	40 mA (ATTENTION : 200mA cumulé pour l'ensemble des broches E/S)
Mémoire Programme Flash	256 KB dont **8 KB** sont utilisés par le bootloader
Mémoire SRAM (mémoire volatile)	8 KB
Mémoire EEPROM (mémoire non volatile)	4 KB
Vitesse d'horloge	16 MHz

Tableau 2.1. *Caractéristiques Techniques d'Arduino Méga 2560* [21].

Elle contient tout ce qui est nécessaire pour le fonctionnement du microcontrôleur (voir figure 2.8), Pour pouvoir l'utiliser et se lancer, il suffit simplement de la connecter à un ordinateur à l'aide d'un câble USB.

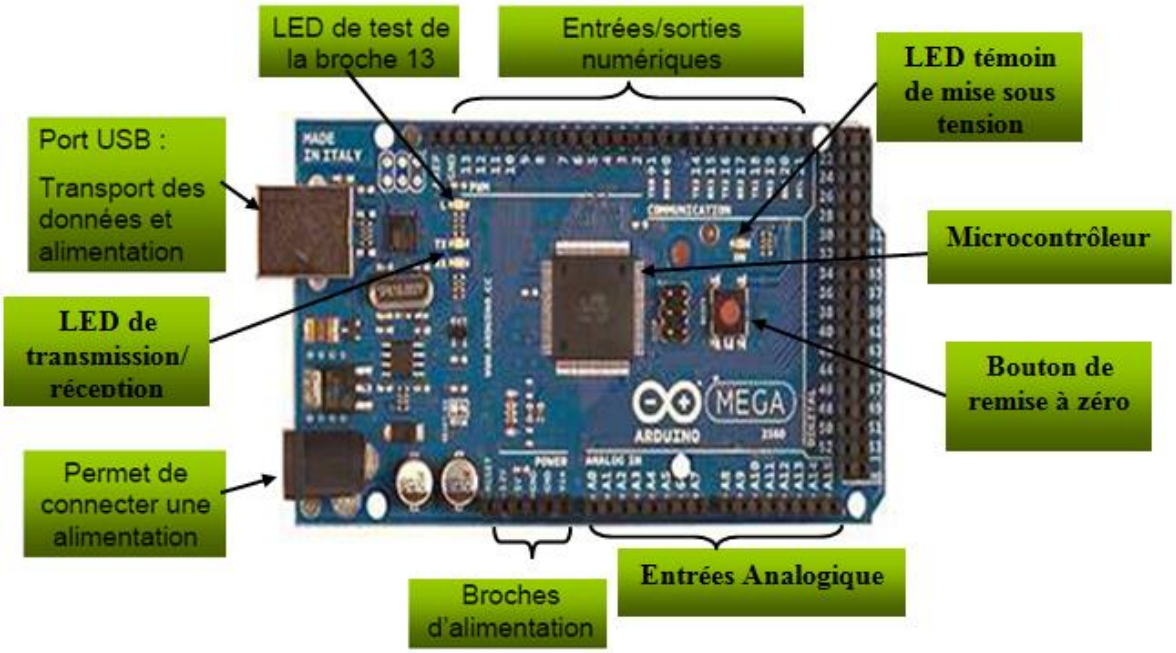


Figure 2.8. *Constitution de la carte Arduino méga 2560.*

2.7.2 Caractéristiques principales

a Alimentation

L'alimentation du microcontrôleur qui équipe les cartes a lieu sous une tension de 5 v qui peut provenir soit de la prise USB dont elles sont munies, ce qui est le cas lorsque la carte est reliée à un ordinateur, soit d'un bloc secteur externe (voir figure 2.9) via le jack standard également présent sur la carte [22].

(a) (b)

(c)

Figure 2.9. *Alimenter l'Arduino : (a) Avec une pile et un connecteur ;*
(b) Avec un adaptateur secteur ;
(c) Avec un cable USB.

Par ailleurs, et par mesure de sécurité pour l'ordinateur auquel sera relié l'Arduino, un fusible réarmable ou Polyfuse est présent sur la connexion d'alimentation 5 v de la prise USB. Toute consommation excessive sur cette dernière, c'est-à-dire en pratique supérieur à 500 mA, provoque le déclenchement de ce fusible, protégeant ainsi le port USB de l'ordinateur auquel la carte est reliée. Le fusible étant de type réarmable , il n'a pas besoin d'être remplacé lorsqu'il a été activé et il suffit de patienter quelques secondes pour qu'il retrouve son état normal [22] .

Les broches d'alimentations présentées sur la carte sont les suivantes [10] :

- **VIN :** La tension d'entrée lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB). 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. 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, intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V. L'intensité maximale disponible sur cette broche est de 50mA.

- **GND :** Broche de masse (ou 0V).

b Horloge

L'horloge fonctionne à la fréquence 16Mhz. En pratique, vous n'aurez quasiment jamais à vous soucier de sa fréquence car les instructions du langage de programmation qui permet de programmer l'Arduino la prennent en compte automatiquement [22] .

c Reset

Toutes les cartes Arduino actuelles sont équipées d'un poussoir de reset manuel. Un appui sur celui-ci permet donc de relancer l'exécution d'un programme si nécessaire, soit parce qu'il s'est « planté » soit tout simplement parce que l'on souhaite le faire repartir de son début [22].

d Les mémoires

L'ATmega 2560 qui équipe l'Arduino Méga 2560 dispose de 256 KO de mémoire de programme. Cette mémoire est de type Flash, analogue à celle que l'on trouve par exemple dans les clés USB, elle se programme électriquement. Une fois programmée, cette mémoire Flash conserve son contenu indéfiniment, même lorsque la carte n'est pas alimentée.

L'ATmega 2560 a également 8 ko mémoire vive ou RAM, cette mémoire est généralement utilisée pour les variables employées dans les programmes, pour stoker des résultats temporaires lors de calculs, etc.

L'ATmega 2560 dispose aussi de mémoire EEPROM, acronyme qui signifie mémoire programmable et effaçable électriquement, la taille de cette mémoire est de 4 KO. Elle est utilisée pour conserver des données ou des paramètres que l'on doit pouvoir retrouver d'une utilisation à l'autre de l'Arduino [22].

e Entrées et sorties numériques

Chacune des 54 broches numériques de la carte Mega peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions pinMode(), digitalWrite() et digitalRead() du langage Arduino. Ces broches fonctionnent en 5V.

De plus, certaines broches ont des fonctions spécialisées [21] :

- **Communication Série**: Port Série **0** : 0 (RX) et 1 (TX), Port Série Serial **1**: 19 (RX) et 18 (TX), Port Série **2**: 17 (RX) et 16 (TX), Port Série **3**: 15 (RX) et 14 (TX). Utilisées pour recevoir (RX) et transmettre (TX) les données.

- **Impulsion PWM (largeur d'impulsion modulée)**: Broches 0 à 13. Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analogWrite().

- **SPI (Interface Série Périphérique)**: Broches 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Ces broches supportent la communication SPI (Interface Série Périphérique) disponible avec la librairie pour communication SPI. Les broches SPI sont également connectées sur le connecteur ICSP.

- **I2C**: Broches 20 (SDA) et 21 (SCL). Supportent les communications de protocole I2C (ou interface TWI (Two Wire Interface - Interface "2 fils"), disponible en utilisant la librairie Wire/I2C (ou TWI).

f Broches analogiques

La carte Mega2560 dispose de 16 entrées analogiques, chacune pouvant fournir une mesure d'une résolution de 10 bits (c.à.d. sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead() du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction analogReference() du langage Arduino [21].

2.8 Présentation du logiciel

2.8.1 IDE Arduino

Un **IDE** (environnement de développement) libre et gratuit est distribué sur le site d'Arduino (compatible Windows, Linux et Mac) à l'adresse http://arduino.cc/en/main/software [2]. Il comporte un éditeur de texte permettant de traduire ce programme source en langage évolué, un compilateur permettant de traduire ce programme source en code binaire prêt à être placé en mémoire de l'Arduino (voir figure 2.10), et un programmateur assurant la programmation de la mémoire de l'Arduino au travers du port USB dont il est équipé [22].

Figure 2.10. *Fonction d'un Compilateur* [14].

L'interface de l'IDE Arduino est plutôt simple (voir figure 2.11), il offre une interface minimale pour développer un programme sur les cartes Arduino.

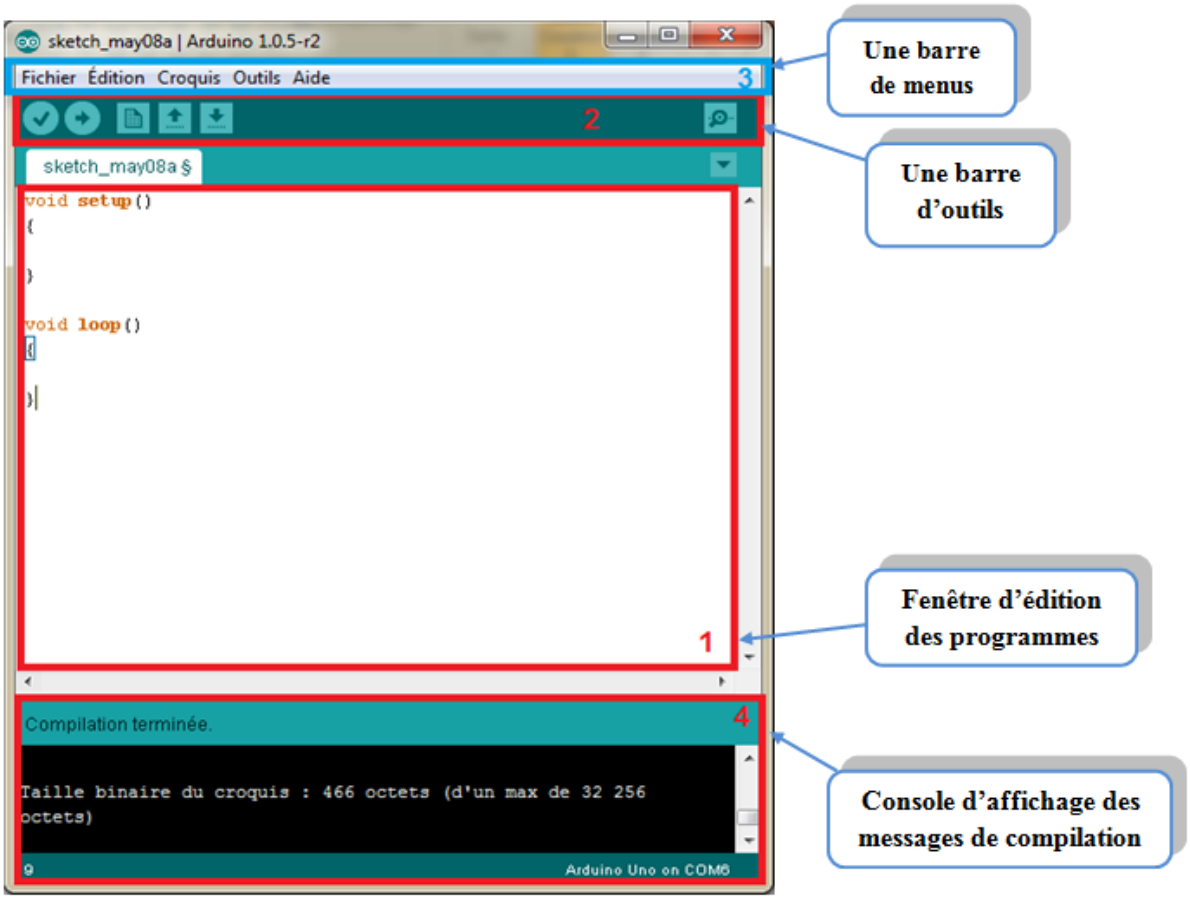


Figure 2.11. *L'interface de IDE Arduino.*

L'IDE est doté d'un éditeur de code avec coloration syntaxique et d'une **barre d'outils rapide** (voir figure 2.12). Ce sont les deux éléments les plus importants de l'interface, c'est ceux que l'on utilise le plus souvent. On retrouve aussi une **barre de menus** plus classique qui est utilisé pour accéder aux fonctions avancées de l'IDE. Enfin, une **console** affichant les résultats de la compilation du code source, des opérations sur la carte, etc **[13]**.

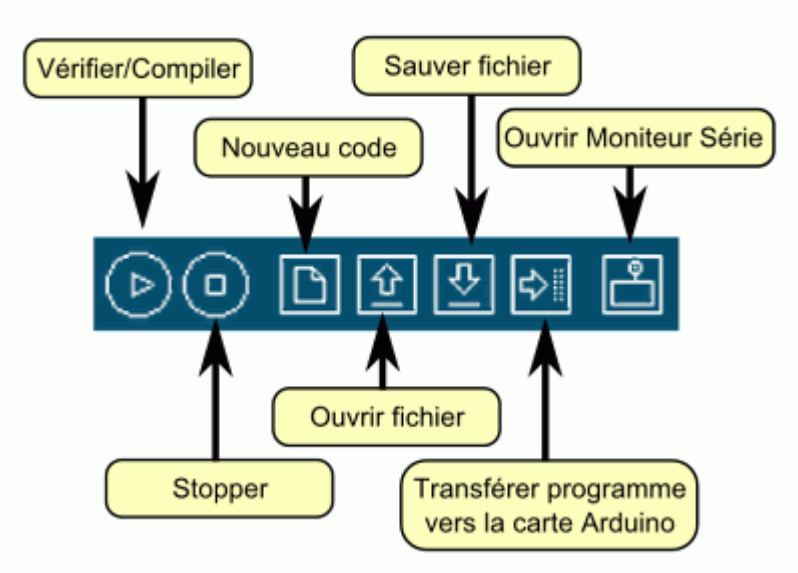


Figure 2.12. *La barre d'outils rapide.*

Le logiciel Arduino intègre également un **TERMINAL SERIE** (fenêtre séparée) (voir figure 2.13) qui permet d'afficher des messages textes reçus de la carte Arduino et d'envoyer des caractères vers la carte Arduino. Cette fonctionnalité permet une mise au point facilitée des programmes, permettant d'afficher sur l'ordinateur l'état de variables, de résultats de calculs ou de conversions analogique-numérique : un élément essentiel pour améliorer, tester et corriger ses programmes. Pour ouvrir la fenêtre terminal de l'IDE Arduino, un simple clic sur le bouton « Sérial Monitor »  . [21]

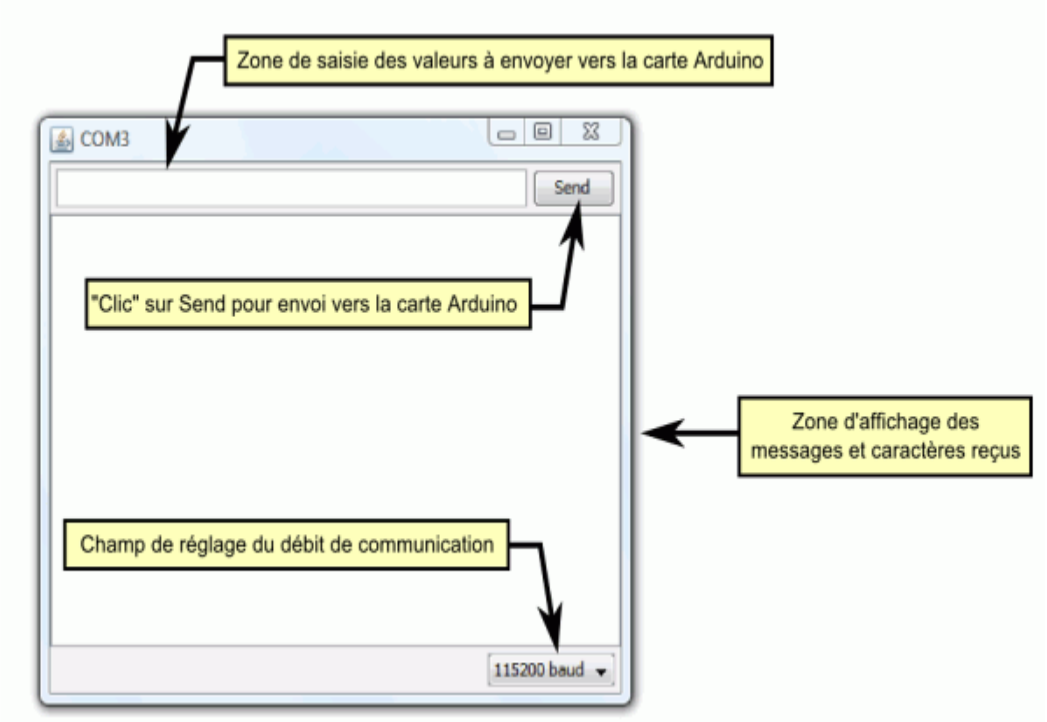


Figure 2.13. *Fenêtre de Moniteur série.*

En effet, après avoir écrit le programme au moyen de l'éditeur intégré à l'environnement de développement, il suffit d'un clic sur l'icône appropriée de sa barre d'outils pour qu'il retrouve programmé dans la mémoire de l'Arduino (voir figure 2.14), et donc que ce dernier l'exécute aussitôt [22].

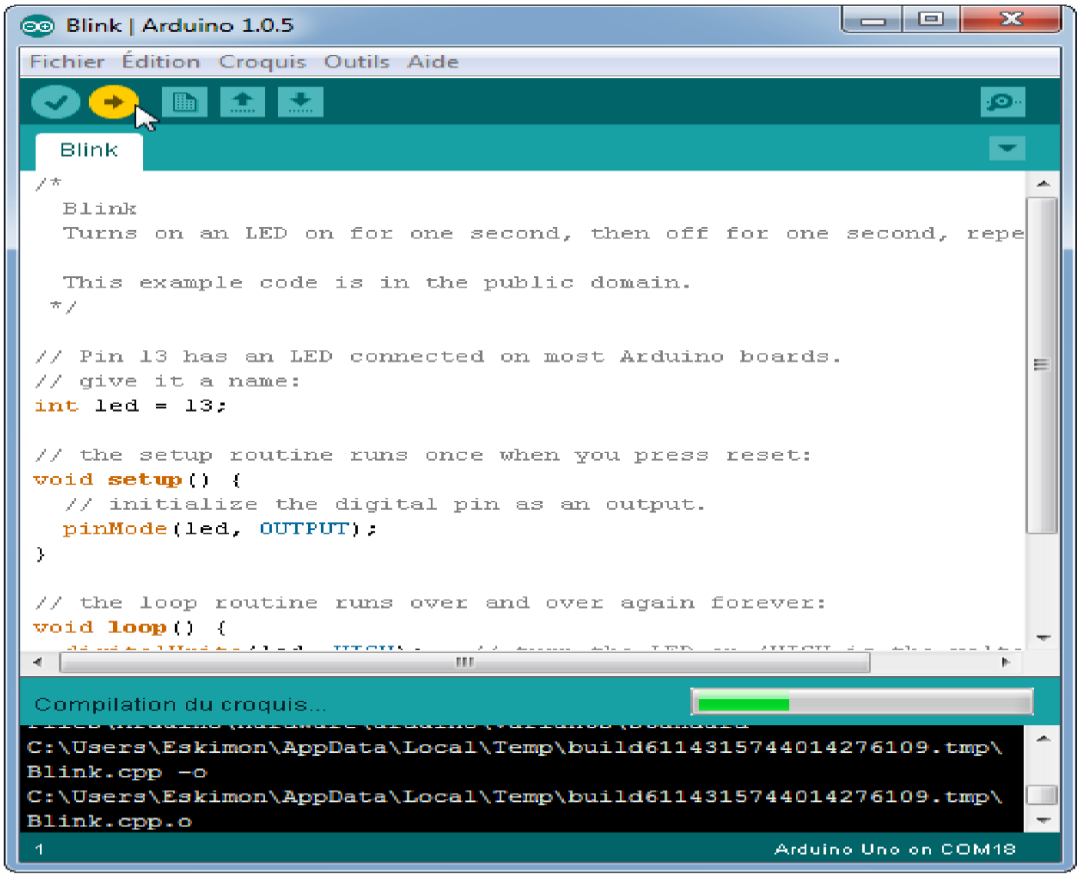


Figure 2.14. *Téléverser le programme dans la carte Arduino* [14].

2.8.2 Langage Arduino

a Structure de programme

Le langage Arduino est inspiré de plusieurs langages. On retrouve notamment des similarités avec le C, le C++, le Java et le Processing. Le langage impose une structure particulière typique de l'informatique embarquée (voir figure 2.15). La fonction ***setup*** contiendra toutes les opérations nécessaires à la configuration de la carte (directions des entrées sorties, débits de communications série, etc.) [13].

La fonction ***loop*** elle, est exécutée en boucle après l'exécution de la fonction *setup*. Elle continuera de boucler tant que la carte n'est pas mise hors tension ou redémarrée par le bouton ***reset***. Cette boucle est absolument nécessaire sur les microcontrôleurs étant donné

qu'il n'on pas de système d'exploitation. En effet, si l'on omettait cette boucle, à la fin du code produit, il sera impossible de reprendre la main sur la carte Arduino qui exécuterait alors du code aléatoire [13].

```
void setup() {
  // initialize both serial ports:
  Serial.begin(9600);
  Serial1.begin(9600);
}

void loop() {
  // read from port 1, send to port 0:
  if (Serial1.available()) {
    int inByte = Serial1.read();
    Serial.write(inByte);
  }

  // read from port 0, send to port 1:
  if (Serial.available()) {
    int inByte = Serial.read();
    Serial1.write(inByte);
  }
}
```

Figure 2.15. *Exemple d'un programme sur IDE Arduino.*

b Les variables

Une variable est un espace de stockage nommé qui permet de stocker une valeur utilisable par la suite dans la boucle d'un programme. Une variable peut aussi bien représenter des données lues ou envoyées sur un des ports analogiques ou numériques, une étape de calcul pour associer ou traiter des données. Une "variable" n'est donc pas exclusivement un paramètre variant dans le programme [23].

Il existe différents types de variables identifiés par un mot-clé dont les principaux sont [23] :

- char (variable 'caractère')
- int (variable 'nombre entier')
- long (variable 'nombre entier de très grande taille')

• string (variable ‘chaine de caractères’)
• array (tableau de variables)

c Fonctions

Une fonction 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, nous citons quelques unes [23] :

Entrées-sorties numériques

• pinMode(broche, état) (configuration des broches)
• digitalWrite(broche, état) (écrire un état sur une broche num.)
• digitalRead(broche) (lire un état sur une broche num.)
• unsigned long pulseIn(broche, état) (lire une impulsion sur une broche num.)

Entrées analogiques

• int analogRead(broche) :lire la valeur d’une broche analogique.
• analogWrite(broche, valeur): écrire une valeur analogique sur une broches.

Gestion du temps

• unsigned long millis() : temps de fonctionnement du programme.
• delay(ms) : attente, en millisecondes.
• delayMicroseconds(us) :attente, en microsecondes.

d Opérateurs de comparaison et opérateurs logiques

Le compilateur de l’Arduino dispose d’un certain nombre d’opérateurs de comparaison et d’opérateurs logiques hérités classiquement du langage C. Le tableau 2.2 résume ces opérateurs [23].

Notation	Fonction
= =	Equivalent à
!=	Différent de
<	Inférieur à
>	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à
&&	Et
||	Ou
!	et pas

Tableau 2.2. *Opérateurs de comparaison et opérateurs logiques* [23].

e Les structures de contrôle

Ce sont les éléments les plus importants du langage, puisque ce sont ces structures qui vont permettre des prises de décision ou bien encore des réalisations de boucles en fonction de l'état de diverses variables. Le tableau 2.3 montre quelques structures de contrôle [22].

Notation	Syntaxe
if (si...)	**If** (condition) **{** Instruction exécutées si la condition est vraie ; **}**
if...else (si...alors...)	**If** (condition) **{** Instruction exécutées si la condition est vraie ; **}** **Else** **{** Instruction exécutées si la condition est fausse ; **}**
for (pour...)	**for** (initialisation ; condition ; incrément) **{** Instruction exécutées dans la boucle ; **}**
switch case (dans le cas où...)	Switch (variable) **{** **Case** Valeur 1 : Instruction exécutées si Variable = Valeur 1 ; **break** ; **default** : Instruction exécutées si variable n'est égale à aucune des variables précédentes ; **}**
while (pendant que ...)	While (expression) **{** Instruction exécutées dans la boucle ; **}**

Tableau 2.3. *Structures de contrôle* [22].

f Librairies

En plus de la simplicité du langage et des nombreuses fonctionnalités qu'offre. L'IDE vient avec un nombre important de **librairies** évitant ainsi d'implémenter des fonctions courantes dans l'informatique embarquée [13].

Voici Quelques librairies d'Arduino [21] :

- La librairie Serial : pour les communications séries entre la carte Arduino et l'ordinateur ou d'autres composants.

- La librairie LCD : pour l'utilisation et le contrôle d'un afficheur LCD alpha-numérique standard.
- La librairie Stepper - pour contrôler les moteurs pas à pas.
- La librairie Keypad - pour l'utilisation des claviers matriciels. (hors référence).
- La librairie Ethernet - pour se connecter à Internet en utilisant le module Arduino Ethernet
- La librairie SD : pour utiliser une carte mémoire SD (utiliser des fichiers, stocker des données, ...)
- La librairie SPI (Serial Peripheral Interface) : pour communication série avec des modules externes supportant le protocole SPI
- Firmata - pour communiquer avec des applications sur l'ordinateur utilisant un protocole série standard.

Conclusion

On peut conclure sur le fait que les cartes Arduino sont un puissant outil de prototypage pour les cartes électroniques. Mais aussi, elles permettent un accès facile et intuitif à l'informatique embarqué. On pourra ainsi enrichir tout ces projets d'un microcontrôleur pour leurs donner une plus value importante [13].

Chapitre 3

Conception et réalisation

du boîtier de corrélation

3.1 Introduction

Le but essentiel de notre travail est d'utiliser la carte Arduino pour réaliser un boitier de corrélation qui permet la mesure de la similarité entre deux signaux, en calculant le coefficient de corrélation. Ce boitier sera utilisé comme outil didactique pour les étudiants dans les TP de traitement du signal. Ils verront en pratique la corrélation de deux signaux.

Ce chapitre est consacré à la conception du boitier de corrélation et à sa réalisation pratique avec des tests et mise en marche.

3.2 Programmation de la carte Arduino Méga 2560

Pour programmer la carte Arduino 2560 avec le logiciel d'Arduino, nous devons suivre les étapes suivantes :

Etape 1: Installation et Configuration du logiciel Arduino

1 - Télécharger le logiciel Arduino depuis le site officiel: http://arduino.cc/en/Main/Software. Puis l'installer.

2 - Brancher ensuite la carte Arduino à l'ordinateur via le port USB, une petite LED verte témoigne de la bonne alimentation de la carte (voir figure 3.1).

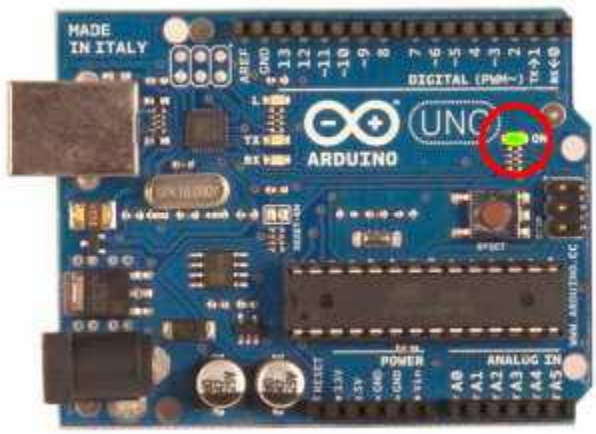


Figure 3.1. *Carte connectée et alimentée*

3 - Ouvrir le logiciel Arduino en cliquant sur 

4 - Aller dans le menu ‘outils’ puis dans ‘carte’ et sélectionner le type de la carte Arduino que nous allons programmer (voir figure 3.2), pour nous il s’agit de la carte “Méga 2560”.

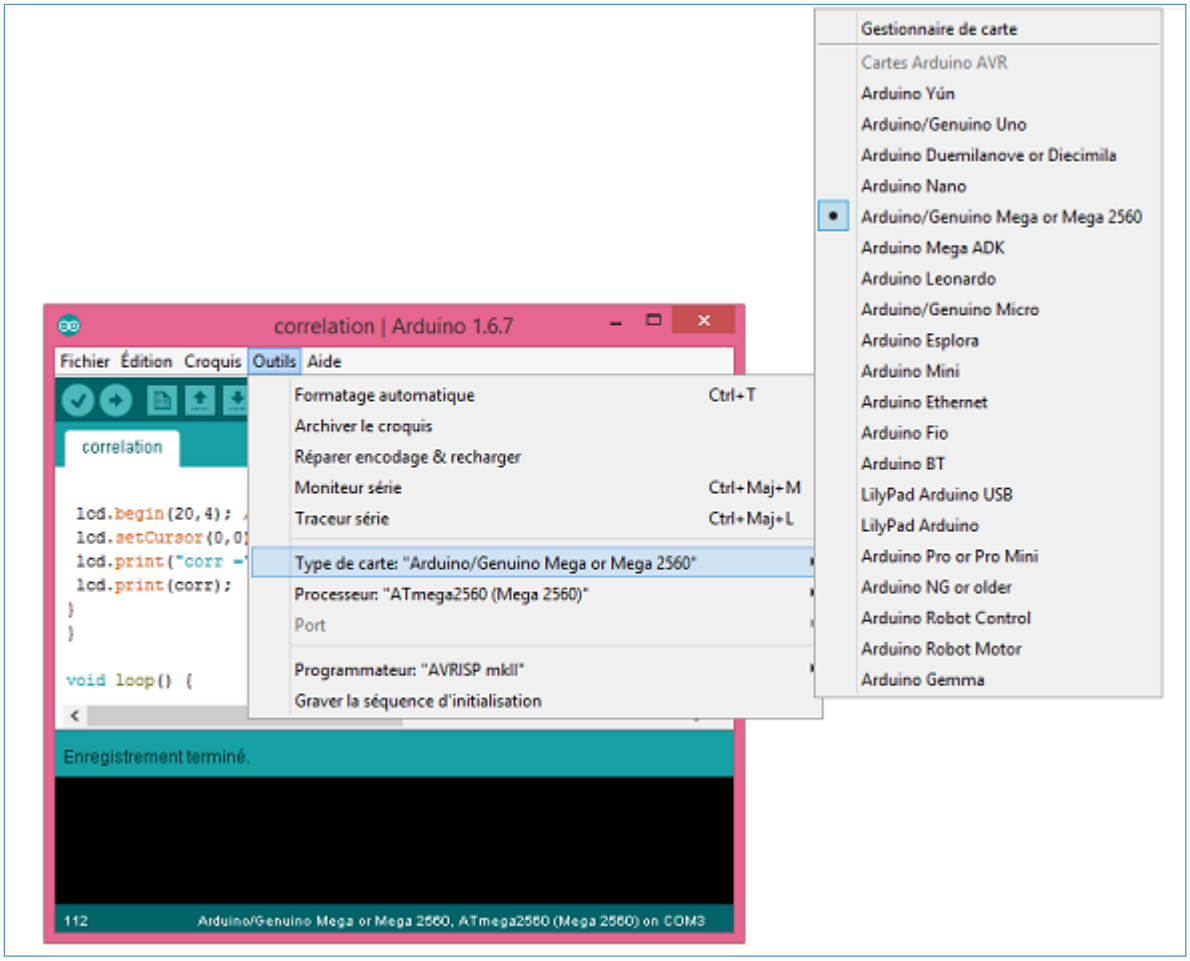


Figure 3.2. *Sélection de type de la carte*

5- Choisir le port série qui est utilisé par la carte arduino : ‘outils’⟶ ‘port série’ (voir la *Figure* 3.3*)*

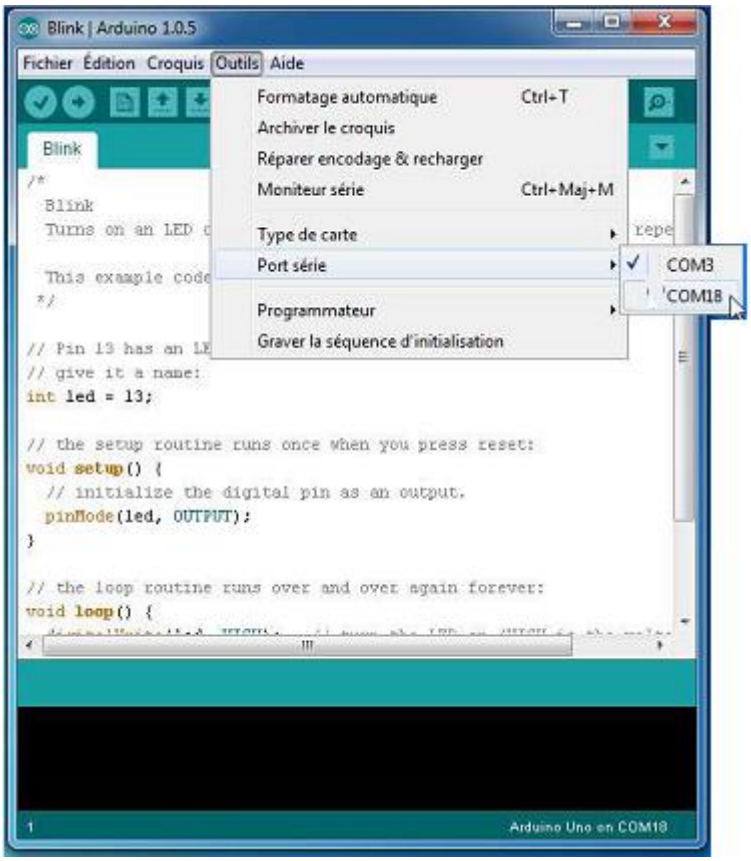


***Figure* 3.3.** *Sélection du port série.*

Etape 2: Tester le bon fonctionnement de la carte

1 - Tester le bon fonctionnement de la carte en chargeant un programme à partir des exemples de programmes existants dans le logiciel Arduino (voir la figure 3.4). Nous allons choisir un exemple tout simple qui consiste à faire clignoter une LED.

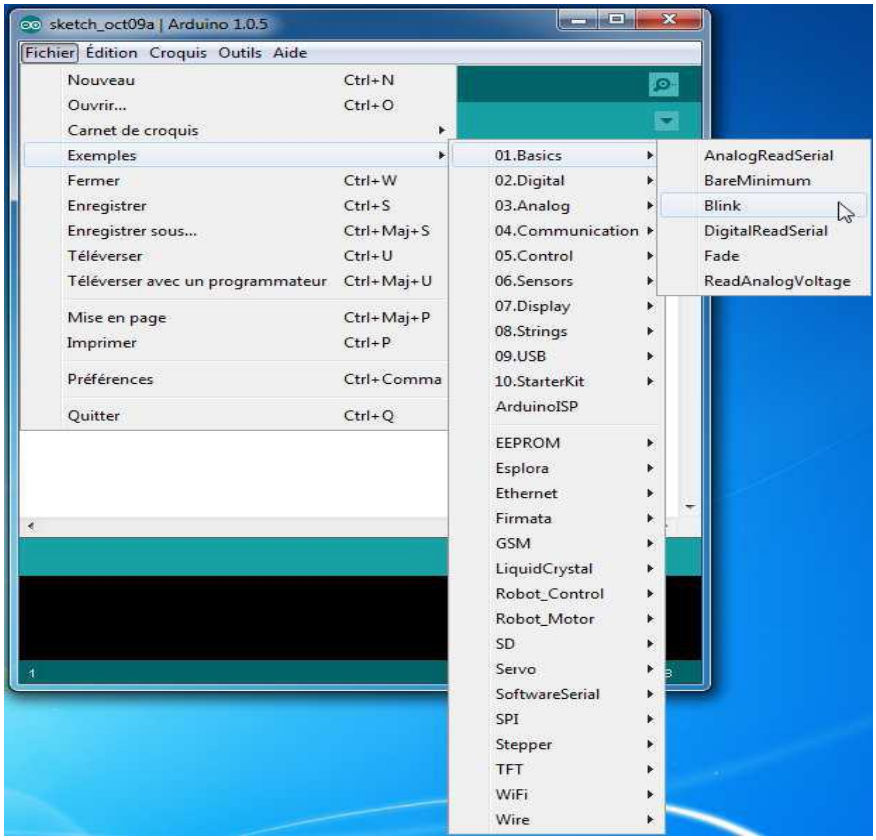


***Figure* 3.4.** *Ouvrir un exemple existant dans IDE Arduino*

2 - Réaliser l'exemple sur la plaque d'essai (voir la figure 3.5).

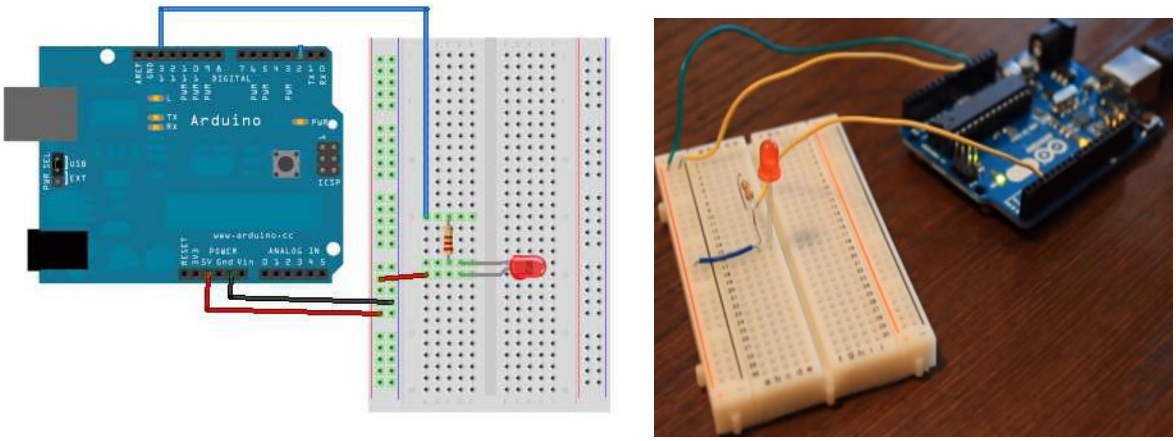


***Figure* 3.5.** *Montage de l'exemple.*

3 - Envoyer le programme dans la carte en cliquant sur le bouton Téléverser (voir la figure 3.6). Une fois que le téléversement est terminé, un message est affiché : " *Téléversement terminé*" qui signifiera que le programme a bien été chargé dans la carte. Si la carte fonctionne, on doit avoir une LED orange sur la carte qui clignote (voir la figure 3.7).

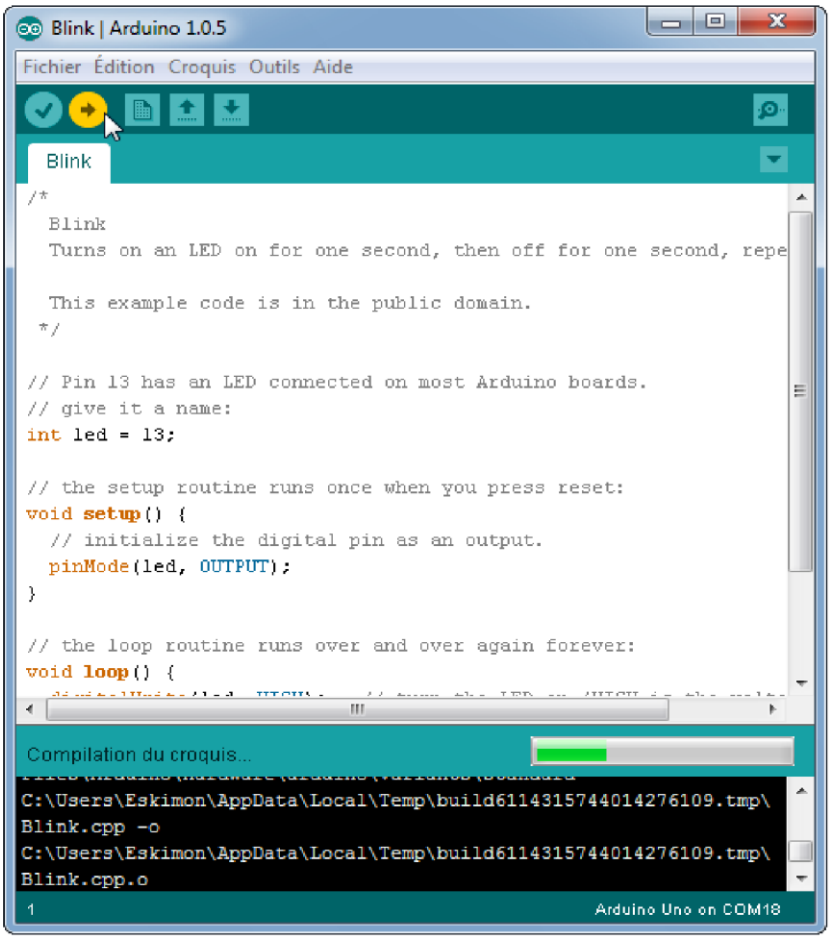


Figure 3.6. *Téléverser l'exemple dans la carte Arduino.*

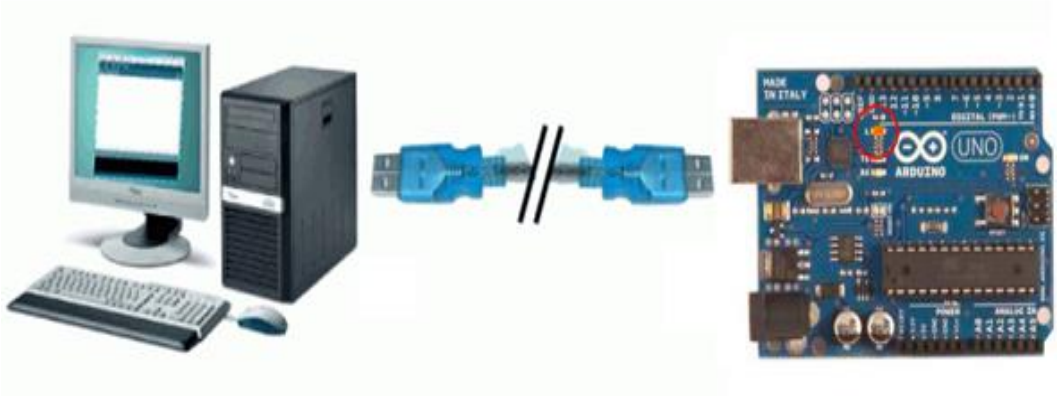
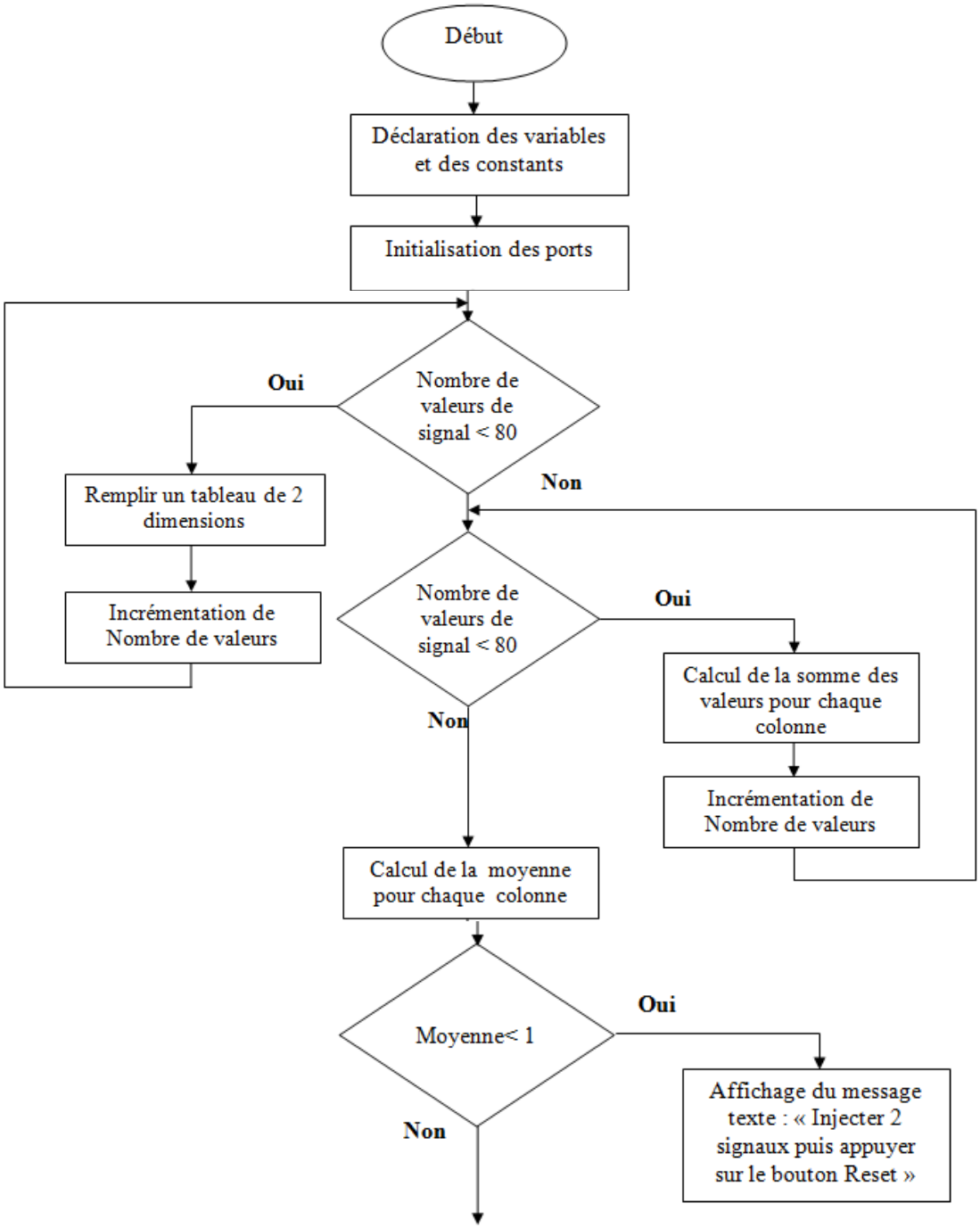


***Figure* 3.7.** *LED orange sur la carte qui clignote [13].*

Etape 3: Développement du programme à réaliser

1- Présentation de l'organigramme

Avant de passer à la programmation, nous devons réaliser un organigramme qui explique le déroulement des différentes séquences. L'organigramme de notre programme est représenté sur la figure 3.8 :


Début
Déclaration des variables et des constants
Initialisation des ports
Nombre de valeurs de signal < 80
Oui
Remplir un tableau de 2 dimensions
Incrémentation de Nombre de valeurs
Non
Nombre de valeurs de signal < 80
Oui
Calcul de la somme des valeurs pour chaque colonne
Incrémentation de Nombre de valeurs
Non
Calcul de la moyenne pour chaque colonne
Moyenne< 1
Oui
Affichage du message texte : « Injecter 2 signaux puis appuyer sur le bouton Reset »
Non

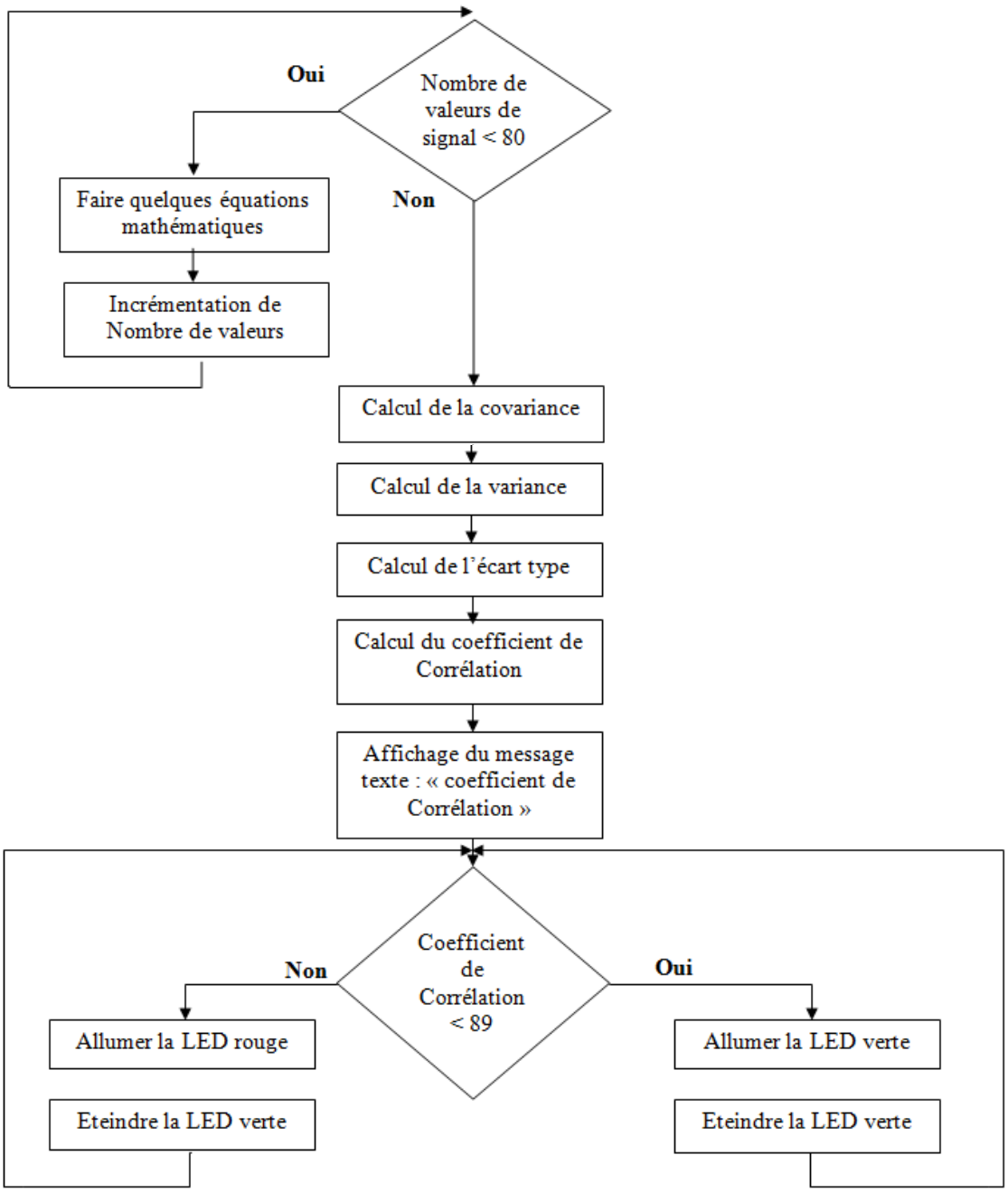


***Figure* 3.8.** *Organigramme.*

2- Présentation du programme

Le programme est représenté sur la figure 3.9 :

```
#include <LiquidCrystal.h> //ajout de la librairie
#include <math.h>
const int led_rouge=8; //LED rouge branché sur la branche 8
const int led_verte=9; //LED verte branché sur la branche 9
const int signalX = A0, signalY=A1;//Mettre lesignal x sur A0  et le
signal y sur A1
int valeurLueX, valeurLueY;
LiquidCrystal lcd(7,6,5,4,3,2); //liaison 4 bits de données
int tab[80][2], i=0; // Déclaration du tableau
float somX=0, somY=0, moyX, moyY, sousX, sousY, A, B, prod, som=0,
somA=0;
float  somB=0, cov, varX, varY, ecartY, ecartX, pro_ecart, corr;


void setup() {

  pinMode(8 , OUTPUT); //broche de sortie
  pinMode(9 , OUTPUT); //broche de sortie

  Serial.begin(9600); // initialisation de la communication série à
9600 bps
  for (i = 0 ; i < 80 ; i++)
{
  valeurLueX = analogRead(signalX),valeurLueY =
analogRead(signalY);
  tab[i][0] =valeurLueX*5/1023,tab[i][1] =valeurLueY*5/1023;
//Remplir le tabeau  1ière colonne avec des valeurs de signal X
   //Remplir le tabeau  2ème colonne avec des valeurs de signal Y
delay (20);
 Serial.println(tab[i][0]);
 Serial.println(tab[i][1]);

}

 for (i = 0 ; i < 80 ; i++)
{
  somX = somX + tab[i][0]; // somme de valeurs X
  somY = somY + tab[i][1]; // somme de valeurs Y
}
moyX = somX/80;


moyY = somY/80;

if(  moyY <1 ||moyX <1  )
{
  lcd.begin(20,4); //utilisation d'un écran 20 colonnes et 4 lignes
 lcd.setCursor(0,0);//Mettre le curs dans la positiont 0,0
 lcd.print("injecter 2 signaux");
 lcd.setCursor(0,1);
 lcd.print("puis appuer sur")
```

```
 lcd.setCursor(0,2);
 lcd.print("le bouton Reset");
else
{
for (i = 0 ; i < 80 ; i++)
{
  sousX =  tab[i][0] - moyX; //valeurs X moins valeur moy X
  A = sousX * sousX;
  sousY =  tab[i][1] - moyY; //valeurs Y moins valeur moy Y
  B = sousY * sousY;
  prod = sousX * sousY;
  som = som + prod;
  somA = somA + A;
  somB = somB + B;
 }
cov = som/80;// Calcul de covariance
varX = somA/80;//Calcul de la variance X
varY = somB/80;//Calcul de la variance Y
ecartX = sqrt(varX);//Calcul de l'ecart type X
ecartY = sqrt(varY);//Calcul de l'ecart type X
pro_ecart = ecartX * ecartY;
corr = cov/pro_ecart;//Calcul de coefficient de correlation
 pinMode (led_rouge,OUTPUT);
 pinMode (led_verte,OUTPUT);

Serial.print("corr =");
Serial.println(corr);
lcd.begin(20,4); //utilisation d'un écran 16 colonnes et 1 lignes
lcd.setCursor(0,0);//Mettre le curseur dans la positiont 0,0
lcd.print("corr =");
 lcd.print(corr);
}
}
void loop() {

   if (corr > 0.89)
{
  digitalWrite(led_verte,LOW); // LED verte allumée si coef
supérieur ou
  digitalWrite(led_rouge,HIGH);                              //égal à
0.89
}
else
{
digitalWrite(led_rouge,LOW); // LED rouge allumée si coef inferieur
à 0.89
digitalWrite(led_verte,HIGH);
}
}
```

***Figure* 3.9.** *Programme.*

a Description du programme

Le but de ce programme est de calculer le coefficient de corrélation de deux signaux présentés aux entrées analogiques A_0 et A_1 de la carte Arduino.

Au début, on déclare toutes les variables utilisées, après on initialise les ports du l'Arduino comme des sorties en utilisant la fonction « ***pinMode*** ». Ensuite, on commence à stocker les états des signaux dans un tableau à deux dimensions (deux colonnes), une pour les états du premier signal et l'autre pour ceux de deuxième signal, en spécifiant l'index i de position que l'on indiquera entre des crochets. Une fois le tableau rempli, on commence à faire les différents calculs nécessaires pour la détermination de coefficient de corrélation : la somme, la moyenne, la covariance..etc. Pour cela, on fait appel chaque fois à la boucle *For* qui nous permet de parcourir le tableau et de faire ces différents calculs.
L'écriture **i++** est un raccourci pour **i = i +1**. C'est-à-dire qu'à chaque tour de boucle, on augmente la valeur de i d'une unité.

Enfin, dans la fonction « ***loop*** », on utilise La structure conditionnelle « ***IF ... ELSE*** » pour commander deux LED une verte et l'autre rouge : La LED verte sera allumée si le coefficient de corrélation est supérieur à 0.89 et la LED rouge sera allumée dans le cas contraire.

3.3 La Réalisation pratique

3.1 Les composants utilisés

a La carte Arduino Méga 2560

Le traitement du signal nous a imposé à choisir une carte plus performante, plus puissante qui possède une vitesse de transmission plus rapide. Pour cela, nous avons choisi une carte Arduino Méga 2560. En plus de sa rapidité, elle possède d'une mémoire plus importantes et des interfaces entrées/sorties plus nombreuses qu'avec les versions précédentes (Arduino UNO).

b Ecran LCD 20 x 4

Afin de rendre le résultat de calcul de coefficient de corrélation accessible en lecture par l'utilisateur et pour pouvoir renseigner ce dernier, on a décidé de mettre en place un écran

LCD « Liquid Crystal Display » (voir figure 3.10**)** qui affiche directement la valeur du coefficient de corrélation.

L'afficheur LCD 20 x 4 qu'on a utilisé possède 4 lignes et 20 colonnes, il permet comme tous les afficheurs alphanumériques, d'afficher des lettres, des chiffres et quelques caractères spéciaux.

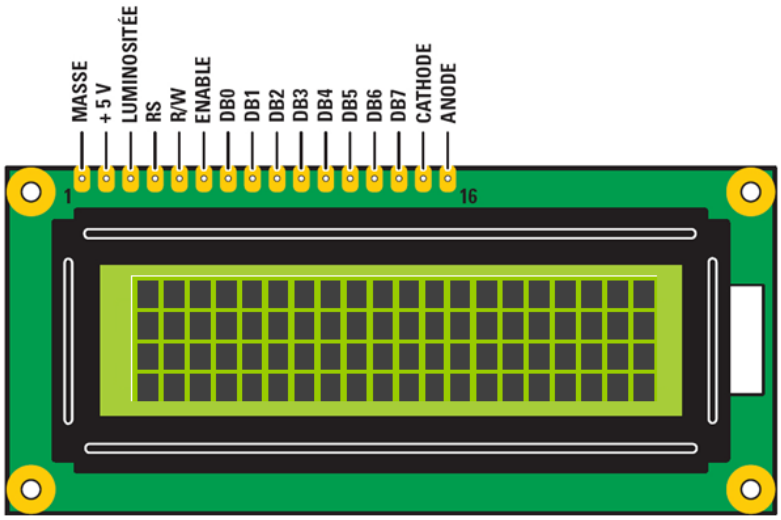


***Figure* 3.10.** Afficheur LCD 20x 4 [16].

➢ Détails de l'assemblage (voir figure 3.11**)**:

- La broche VSS est reliée à la masse (Gnd).
- La broche VDD est reliée à l'alimentation 5v.
- RS est reliée au port digital 12.
- RW est reliée à la masse, une façon de lui donner une valeur basse pour passer en mode écriture.
- E est reliée au port digital 11.
- V0 est reliée à la broche de données du potentiomètre, au centre.
- La broche Anode est reliée à 5v.
- La broche Cathode à la masse.
- Les broches D4 à D7 du LCD sont reliées aux ports digitaux 2 à 5 de l'Arduino.

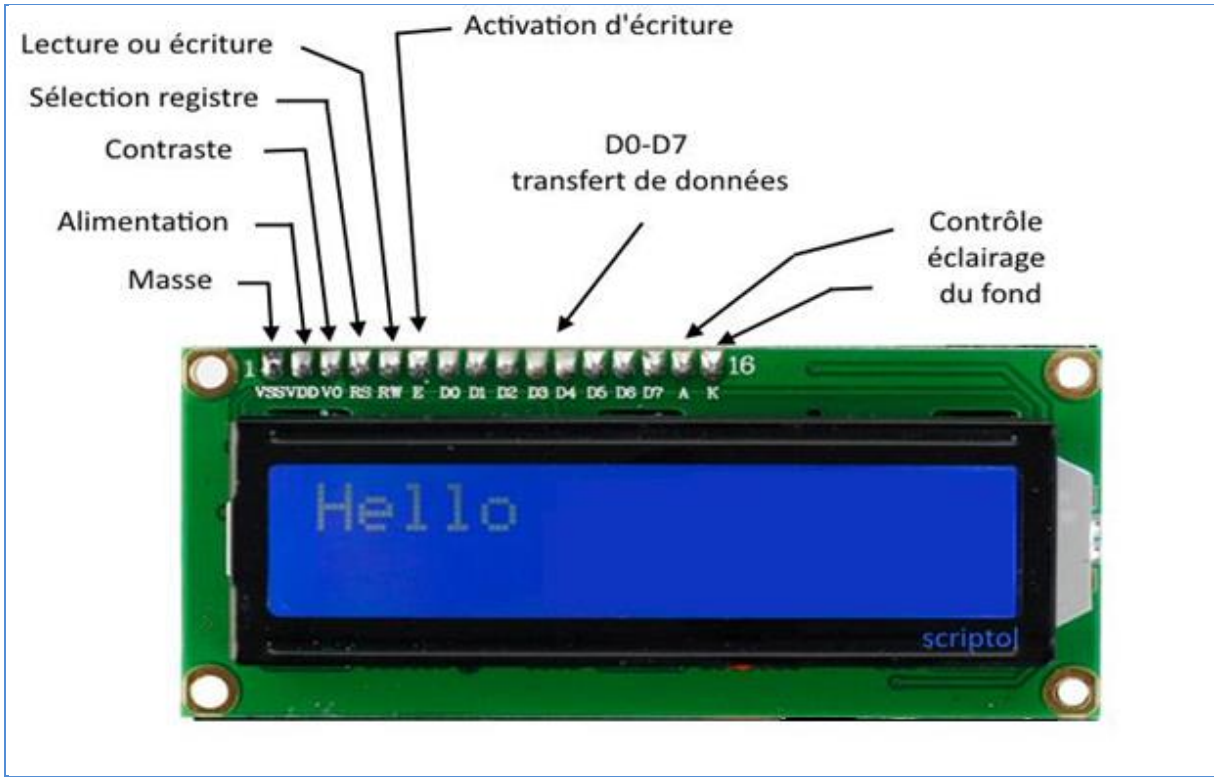


Figure 3.11. *Les broches de l'écran LCD.*

c Un potentiomètre

Pour régler le contraste de notre écran LCD, on branche un potentiomètre de 1 KΩ à la broche V_0 de l'écran.

d 2 LED et 2 résistances

On a mis en place deux LED une rouge et une verte qui jouent le rôle de voyant lumineux sur notre boitier et qui permettent de voir si les signaux sont corrélés ou non. La LED verte s'allume si le coefficient de corrélation est proche ou égale à 1, et la LED rouge s'allume dans le cas contraire.

Pour protéger ces LED, on branche chacune en série avec une résistance.

e Un bouton interrupteur

Afin de commander notre boitier, on utilise un bouton interrupteur.

f Alimentation

Pour alimenter notre boitier, on utilise 3 piles de 1.5v (voir figure 3.12**)** qui donnent une tension totale égale à 4.5v.

***Figure* 3.12.** *Support piles avec connecteur jack.*

3.2 La Réalisation sur la plaque d'essai

Avant de passer à la réalisation sur circuit imprimé, nous avons branché notre montage sur une plaque d'essai (voir les figures 3.13 et 3.14**)** qui nous a permet de tester notre montage sans besoin de le souder.

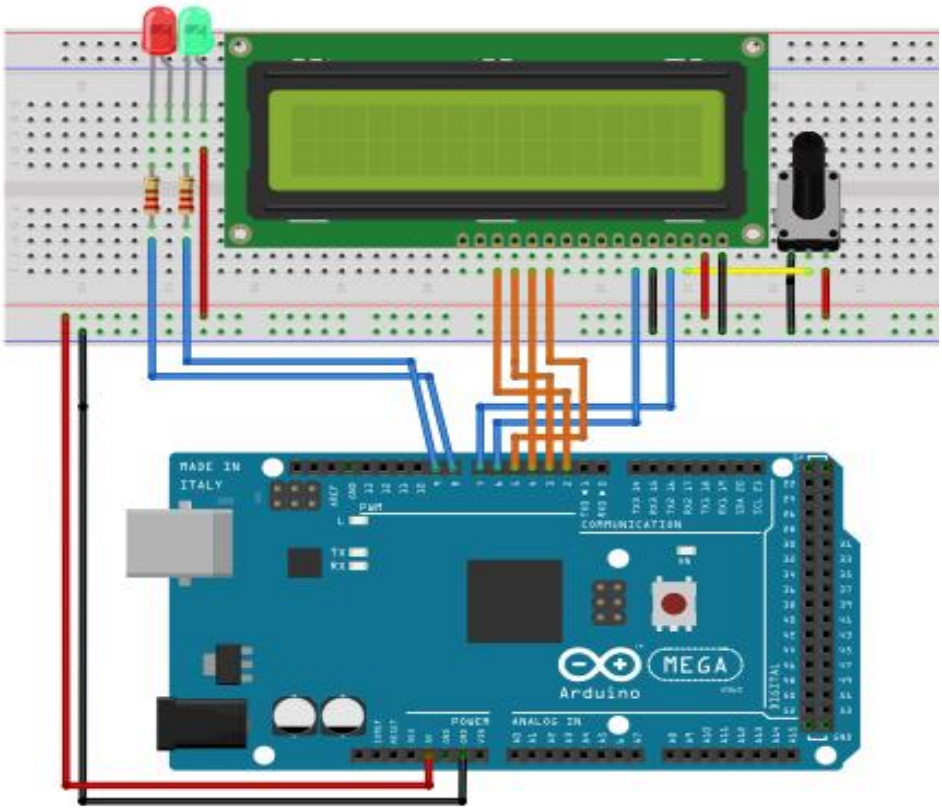


***Figure* 3.13.** *Schéma de circuit sur logiciel Fritzing.*

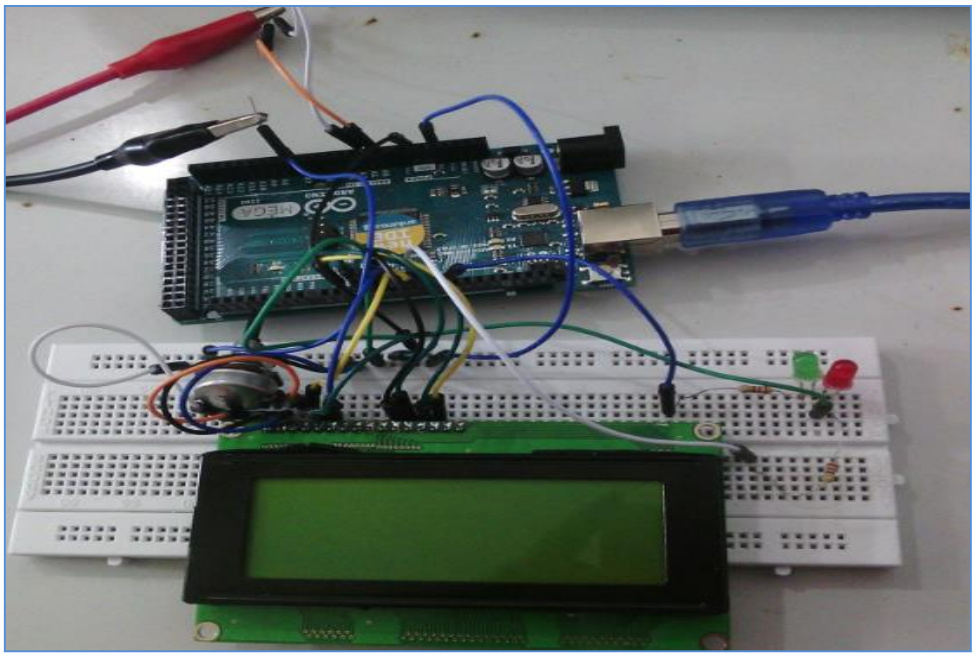


***Figure* 3.14.** *La réalisation sur la platine d'essai.*

3.3 Tests expérimentaux

Pour vérifier l'exactitude de notre programme ainsi que les formules de corrélation utilisées, on a fait plusieurs tests.

Dans un premier temps pour simplifier la tâche, au lieu d'utiliser les signaux on a utilisé un petit tableau de deux dimensions (voir figure 3.15**)**. Tout d'abord, pour voir si le coefficient de corrélation calculé par notre programme était juste, on le recalcule, pour la même série de valeurs, avec le logiciel Excel en utilisant la fonction COEFFICIENT.CORRELATION, puis on compare les coefficients.

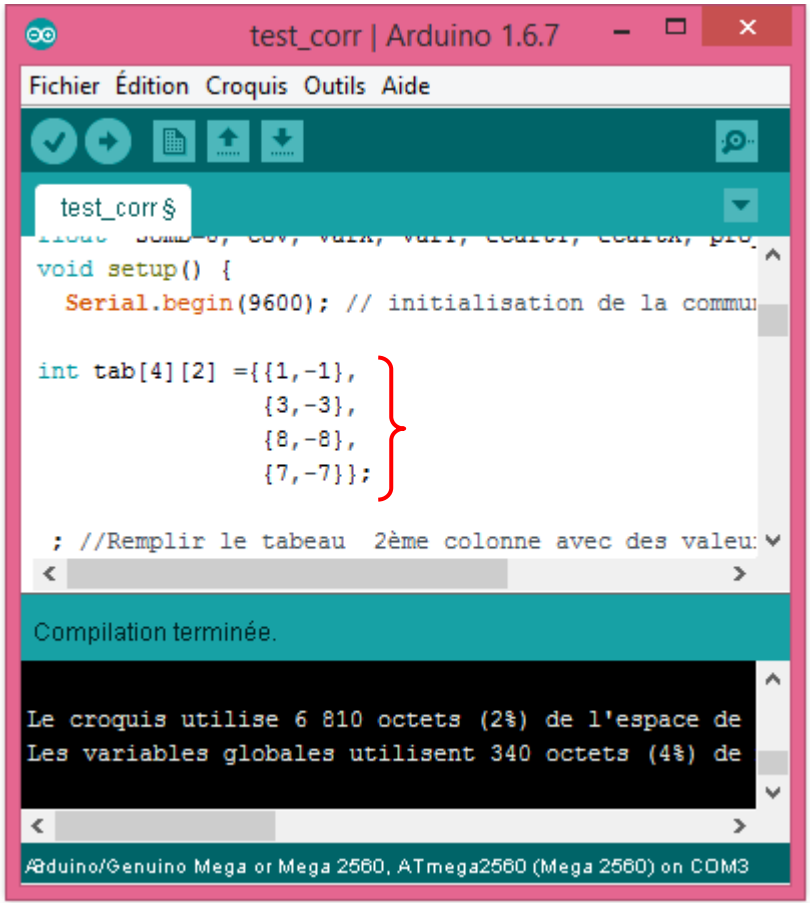


***Figure* 3.15.** *Déclaration du tableau.*

Afin de détecter les erreurs et les corriger, on affiche les résultats de calculs de chaque étape de notre programme sur le moniteur série intégré avec le logiciel Arduino.

Dans un second temps, on a fait les tests sur des signaux. Pour cela on a utilisé deux GBF pour générer deux signaux et un oscilloscope pour les visualiser (voir figure 3.16**)**. Avant d'injecter les signaux, il faut d'abord régler le type de signal (voir figure 3.17**)**, sa fréquence et son amplitude qui doit être positive et inférieure à 5 v.

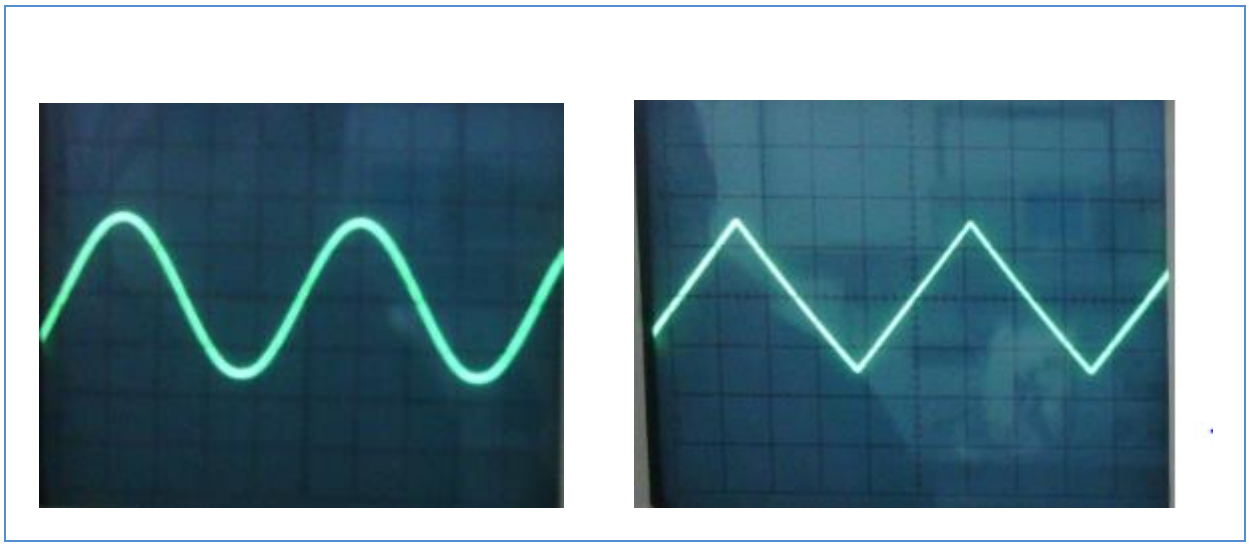


***Figure* 3.16.** *Exemple de signaux injectés aux entrées d'Arduino.*

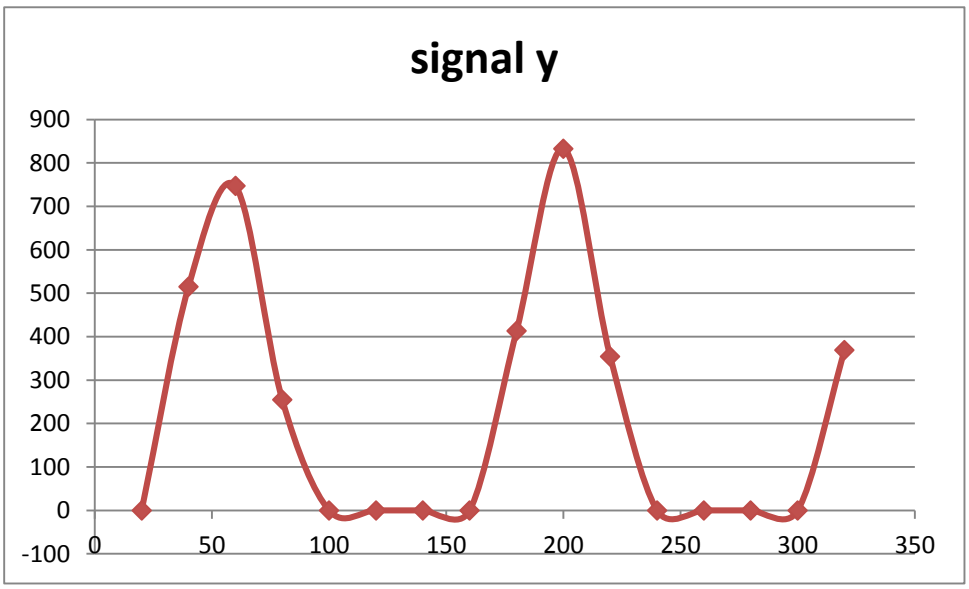


***Figure* 3.17.** *Exemple de signal sinusoïdal injecté obtenue sur Excel.*

Une fois les signaux réglés, on charge le programme dans la carte et on constate les résultats suivants (voir les figures 3.18 et 3.19**)**.

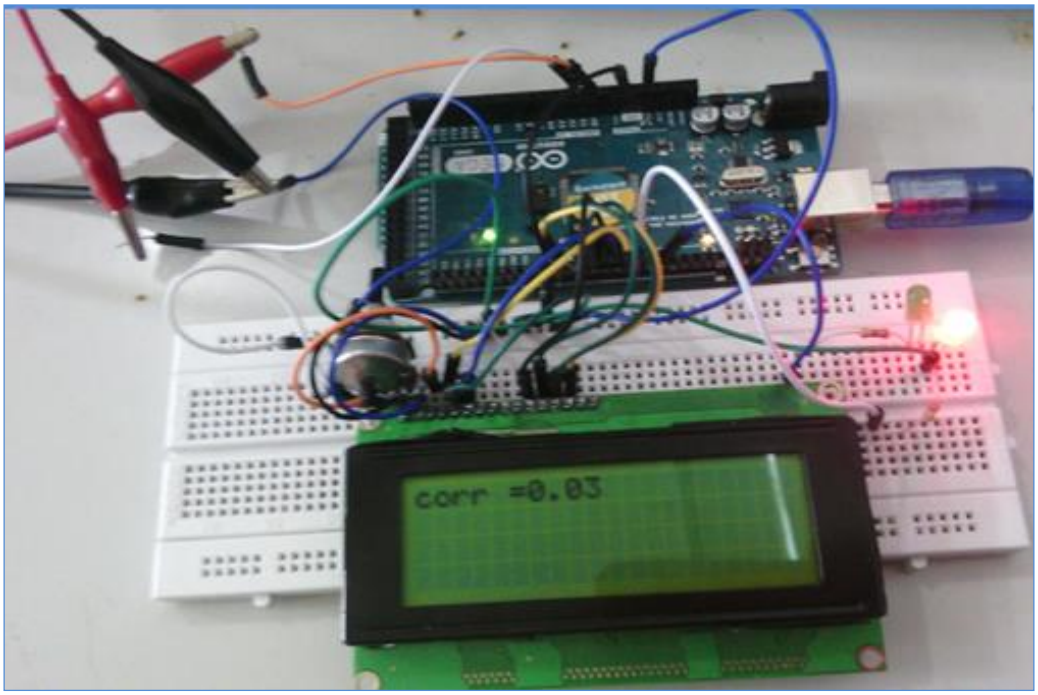


***Figure* 3.18.** *Résultat du programme en cas d'injection de 2 signaux différents.*

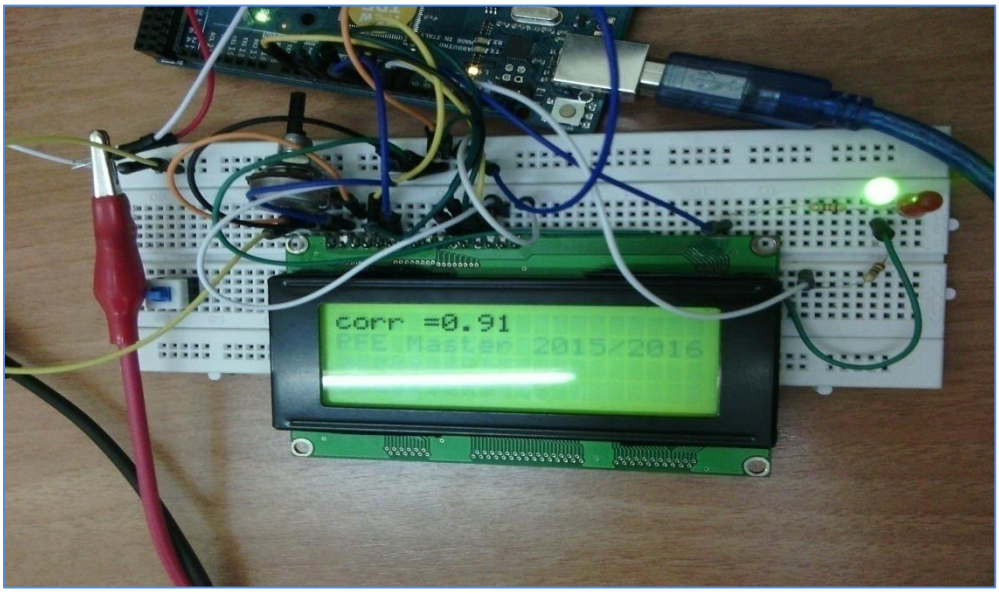


***Figure* 3.19.** *Résultat du programme en cas d'injection de même signal.*

3.4 La réalisation du circuit imprimé

Dans le but de réaliser matériellement un boîtier compact pour notre projet, il a fallu créer un circuit imprimé pour faire le lien entre la carte Arduino et les différents composants. Pour ce faire, nous avons modélisé sur ordinateur le circuit à l'aide du logiciel Proteus professionnel.

Ce logiciel permet la conception assistée par ordinateur. Il est composé de deux logiciels principaux : ISIS, permettant entre autres la création de schéma et la simulation électrique, et ARES, dédié à la création de circuits imprimés.

Pour réaliser le circuit imprimé, nous devons suivre les étapes suivantes :

Etape 1: Dessiner le Schéma électrique sur ISIS

Dans un premier temps pour dessiner le schéma électrique, il faut d'abord ajouter les bibliothèques de composants non incluses dans le Proteus. Dans notre cas, la bibliothèque d'Arduino n'est pas intégrée dans le Proteus (voir figure 3.20**)**, donc il est nécessaire de l'ajouter. Pour ce faire, on a téléchargé d'abord la bibliothèque d'Arduino qui contient deux fichiers *ARDUINO.LIB et ARDUINO.IDX*, puis on a copié ces fichiers dans la bibliothèque de Proteus (voir figure 3.21**)**.

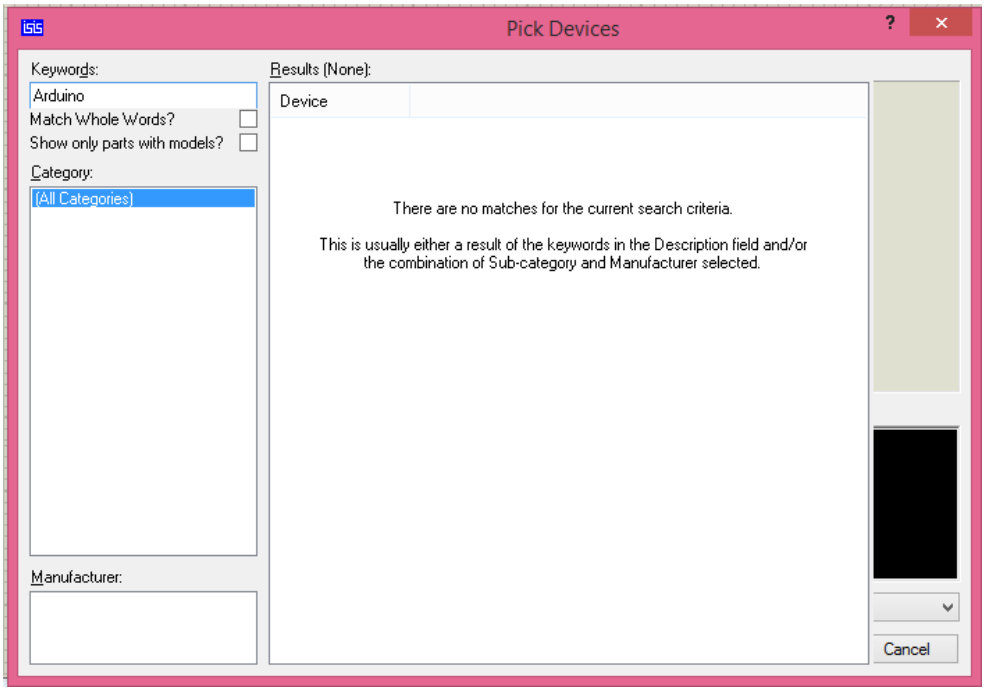


Figure 3.20. *La carte Arduino introuvable dans Proteus.*

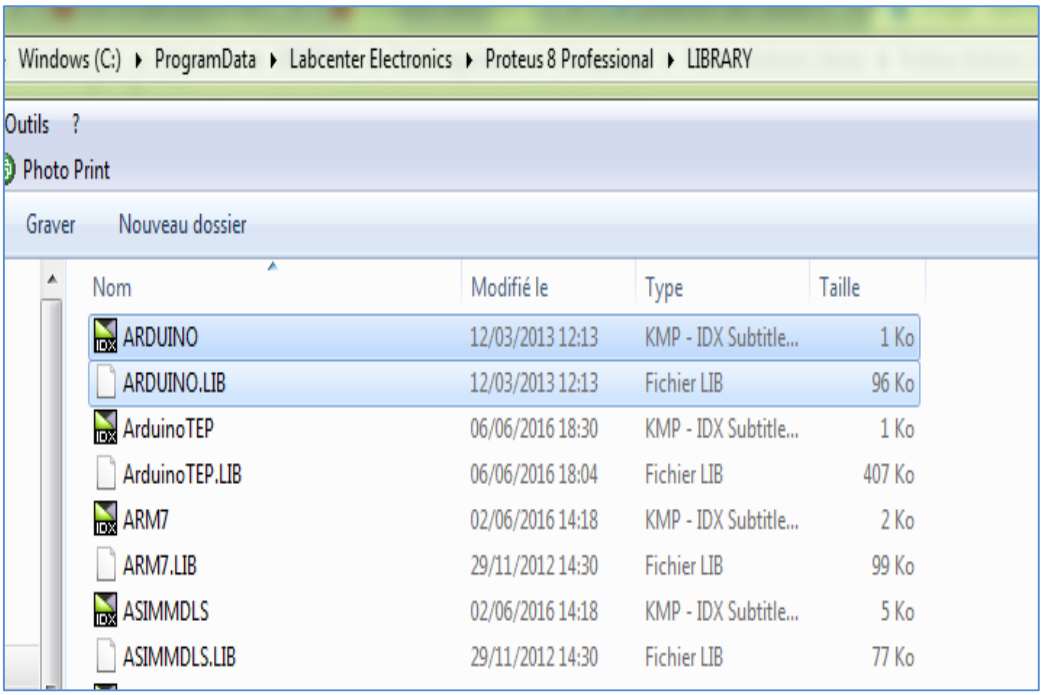


Figure 3.21*. Intégration des fichiers de la bibliothèque Arduino dans Proteus.*

Maintenant, notre carte Arduino est prête à être utilisée par le Proteus comme le montre la figure 3.22.

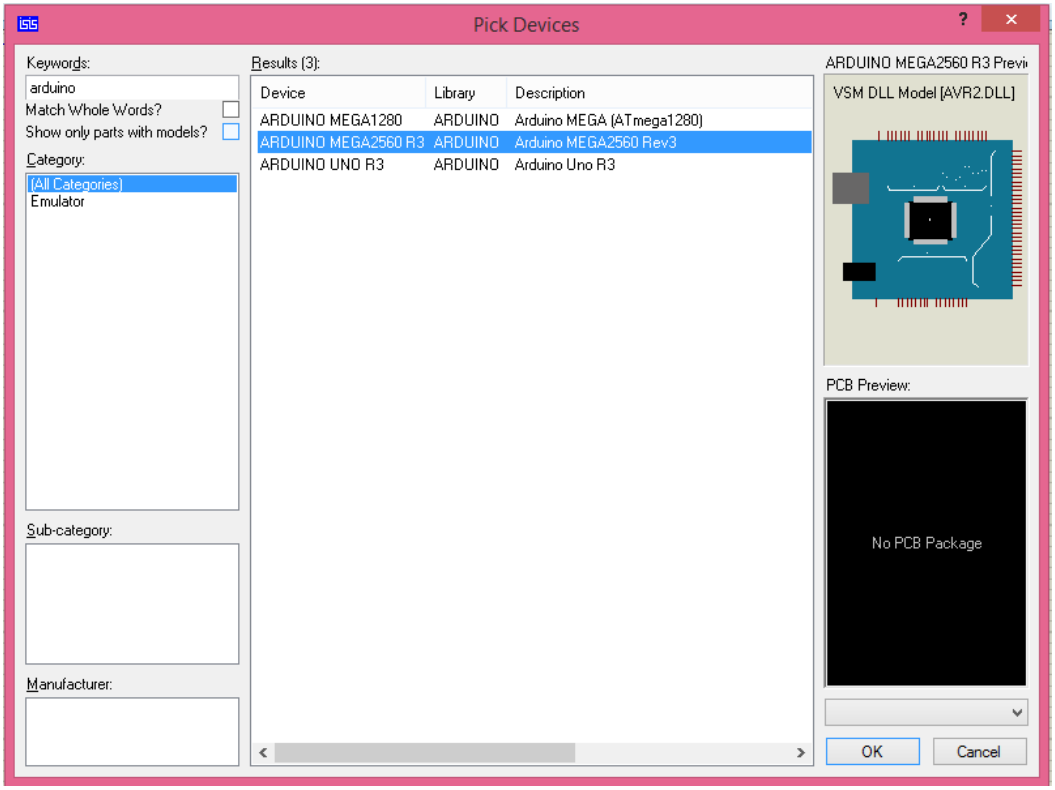


***Figure 3.22**. La carte arduino méga dans la bibliothèque du proteus.*

Dans un second temps, on a commencé à dessiner notre circuit sur ISIS, en insérant d'abord les différents composants. Puis on les a reliés pour former notre circuit (voir figure 3.23).

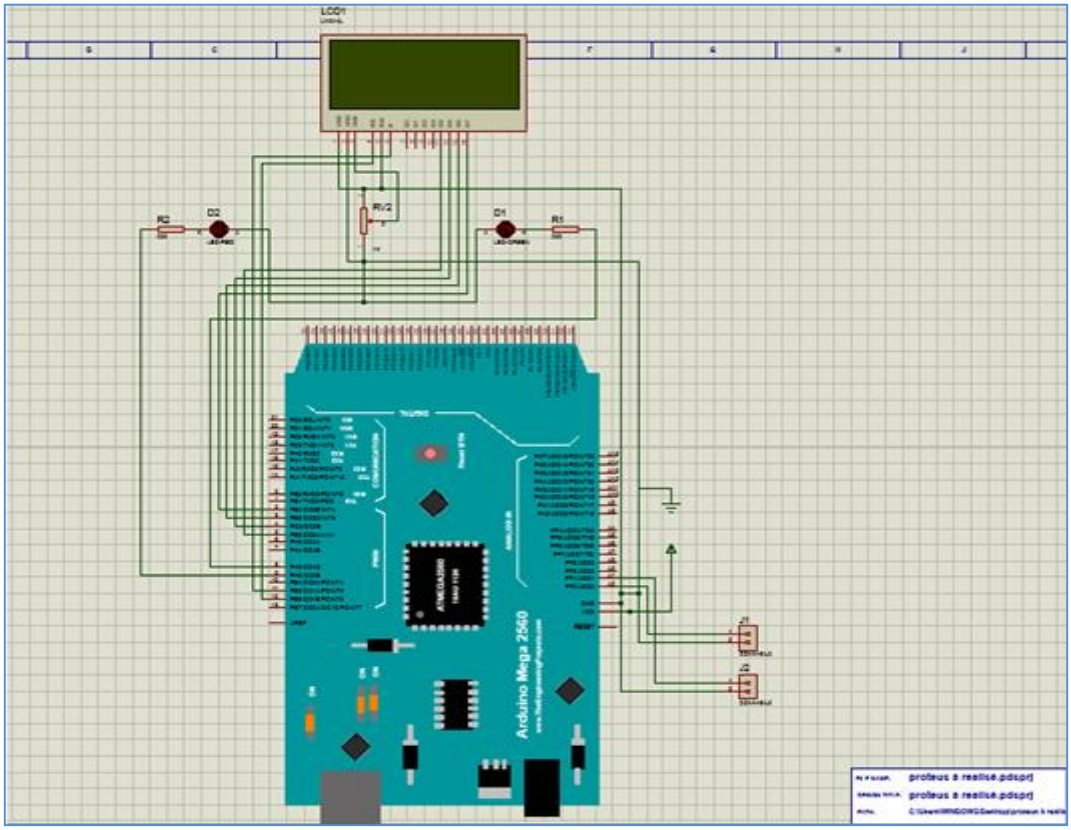


***Figure 3.23**. Schéma électronique de notre montage sur Proteus ISIS.*

Etape 2: Création de Typon

Maintenant que notre circuit est prêt, on va commencer à créer le typon de notre circuit. Le typon est en effet le négatif de la partie cuivrée du circuit imprime. Il s'agit donc de créer les liaisons entre les différents composants utilisés : Arduino, diodes, afficheur LCD, etc. avec des pistes en cuivre sur le circuit imprimé. De ce fait, on a utilisé le logiciel ARES inclue dans Proteus.

Pour créer le typon, on place d'abord les différents composants de notre circuit tout en définissant la taille de chaque composant et celle de la platine cible ($14\ cm \times 12.5\ cm$). Une fois disposée les composants sur la platine, il faut router toutes les pistes électriques du circuit. Cela s´effectue à la main (on remplace les fils volants par des pistes de largeur définie), ou grâce à la fonction auto-routage. Notons que la combinaison de deux méthodes manuelle-automatique donne les meilleurs résultats.

Après de nombreux essais infructueux, nous somme enfin parvenues à router toutes les pistes sans erreurs en créant deux faces, le schéma final sur ARES est représenté sur la figure 3.24.

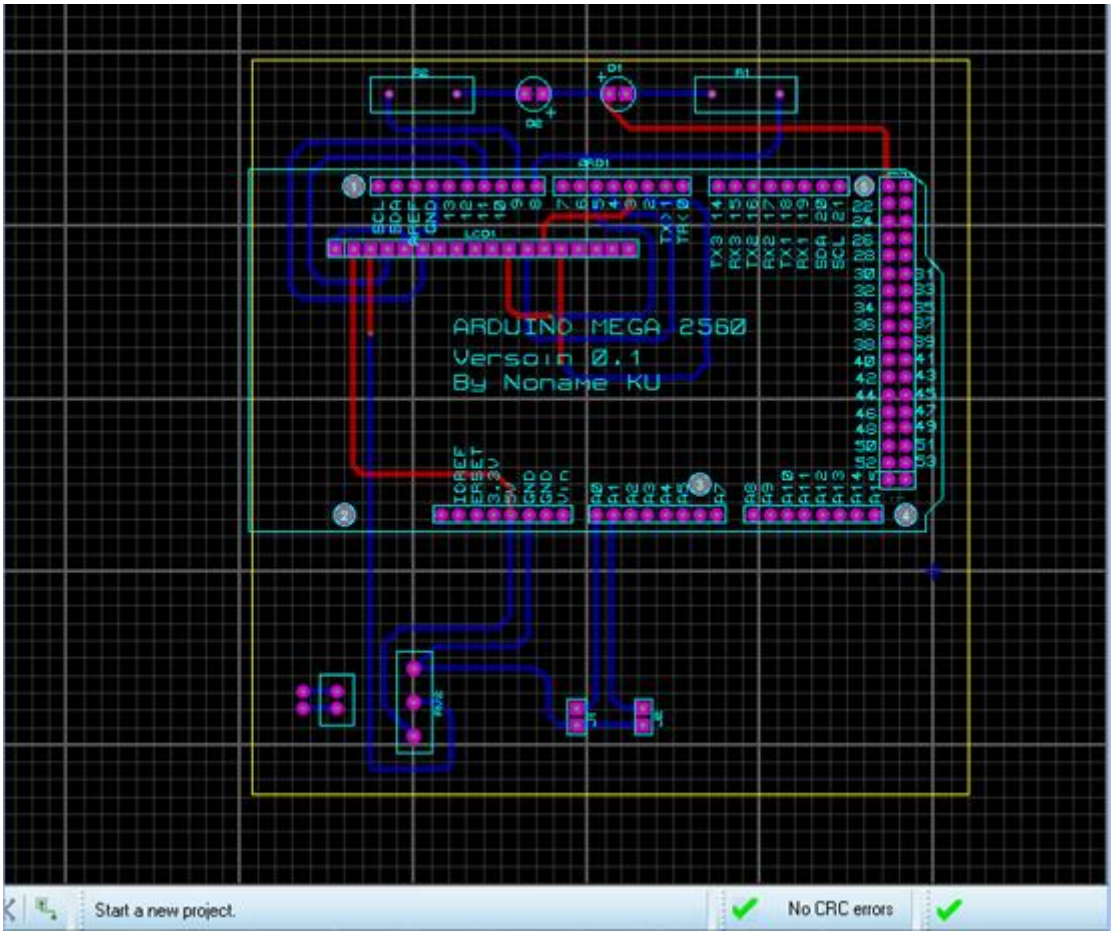


***Figure 3.24**. Typon avec composants développé sur Proteus ARES.*

Pour visualiser notre carte électronique en 3D, on clique sur « **Visualisation 3D** ». On peut alors zoomer et faire pivoter la carte dans l'espace pour l'observer sous différents angles (voir figure 3.25**)**.

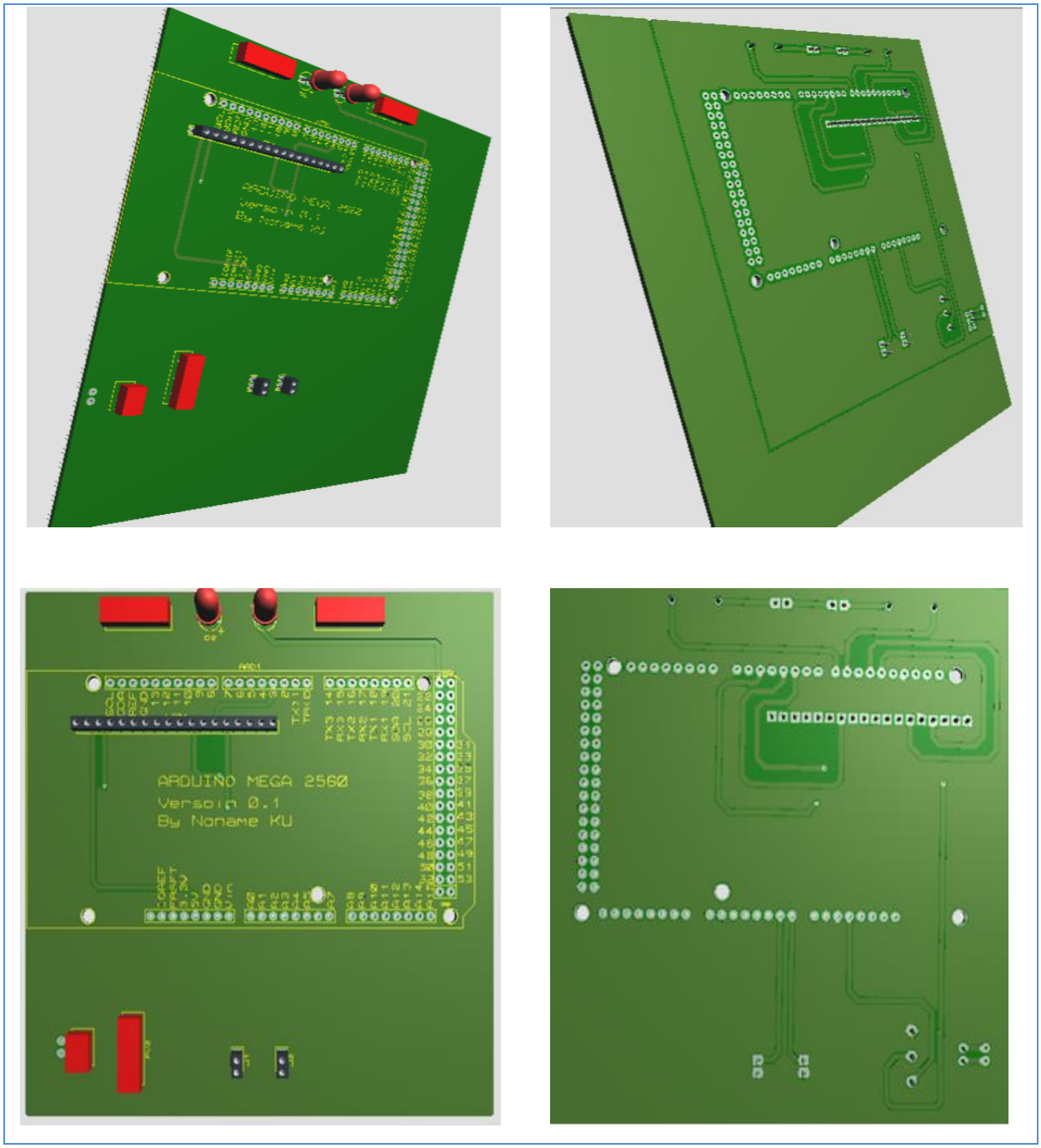


***Figure 3.25**. Visualisation 3D de deux faces cuivré.*

Enfin pour avoir le typon à imprimer, on clique sur '*output*' dans le menu ARES, puis sur '*print layout'*, un fichier se génère contient le typon final à imprimer sur un support transparent (voir les figures 3.26 et 3.27**)**.

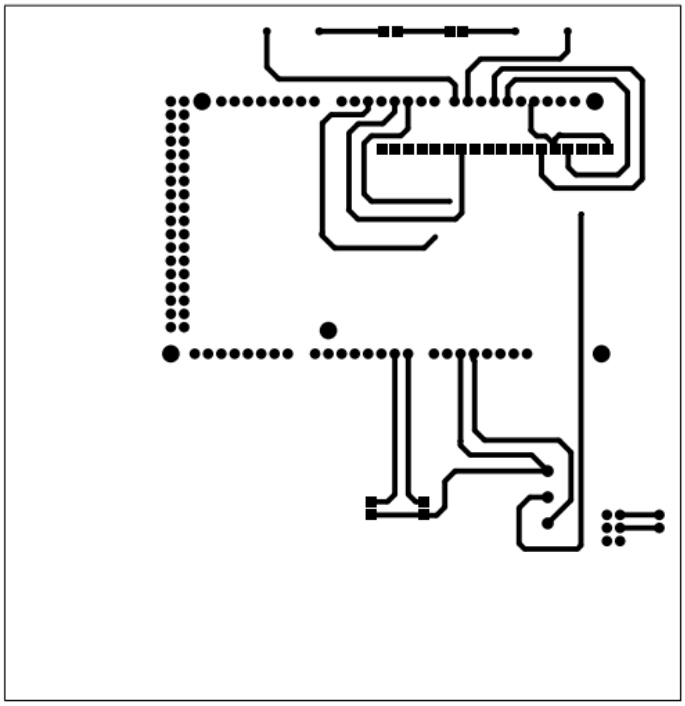


***Figure 3.26**. Typon positif de la première face cuivrée.*

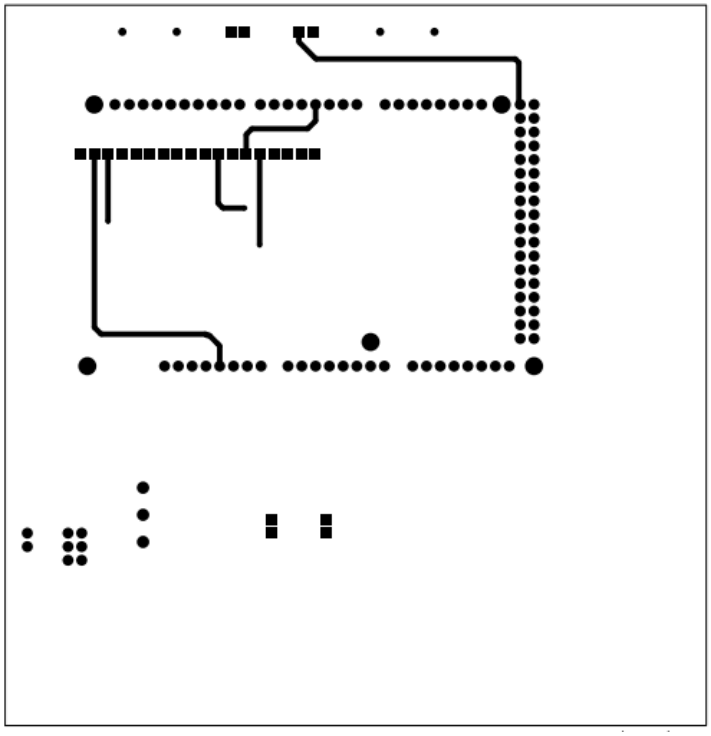


***Figure 3.27.** Typon positif de la deuxième face cuivrée.*

3.5 Conclusion:

Dans ce dernier chapitre, on a procédé à la conception et la réalisation du boitier de corrélation. Pour cela, on a présenté d'abord le programme développé sous l'environnement Arduino en expliquant ses différentes étapes. Ensuite, on a parlé des différents composants utilisés pour réaliser le boîtier. Enfin, on a expliqué la réalisation pratique du boîtier ainsi que les tests expérimentaux.

Conclusion générale

Les cartes arduino sont des cartes électronique programmable matériellement libres sur lesquelles se trouve un microcontrôleur(d'architecture Atmel AVR) , et un logiciel multiplateforme. Les schémas de ces cartes sont publiés en licence libre, qui puisse être accessible dans le but de créer facilement des systèmes électroniques. Les cartes arduino ils sont à l'avant-garde de la révolution numérique et permettent une plus grande rapidité et efficacité des produit numérique actuels permettent un accès simple et peu couteux à l'informatique embarquée.

Dans le but de comprendre le principe de fonctionnement de ces cartes l'objectif principale était de se familiariser avec le fonctionnement de l'environnement de développement arduino IDE après une description bien développé de la carte, nous nous sommes intéressé premièrement, au logiciel arduino qui n'est autre qu'un environnement de développement intégré qui sert à programmé notre application puis la chargé directement dans notre carte pour une application en temps réel et afin d'appliquer les connaissances acquises, nous avons commencé par l'implémentation de quelques tests préliminaires sur notre carte en temps réel en générant des signaux de notre carte elle-même.

Ensuite nous nous sommes intéressés à une opération très importante dans le traitement du signal qui est la corrélation des signaux afin de comparer deux signaux et de savoir si ces deux signaux proviennent d'une même origine. Nous avons alors procédé au développement du programme grâce à l'environnement Arduino, après nous sommes passé à une implémentation directe sur notre carte Arduino en temps réel après l'injection des deux signaux, et pour tester l'acquisition des signaux modélisés par notre Arduino, nous avons réalisé plusieurs tests de fréquence. Pour cela, nous avons utilisé un GBF qui émettait un signal sinusoïdal et respectait plusieurs critères.

Dans le but de réaliser matériellement un boîtier compact pour notre projet, il a fallu créer un circuit imprimé pour faire le lien entre la carte Arduino et les différents composants

Conclusion générale

en utilisant l'un des logiciels électroniques les plus puissants qui n'est d'autre que le proteus et à la fin on a arrivé à réaliser le circuit imprimé. L'avantage principal du boîtier réside dans le fait qu'une fois le code téléversé dans la carte Arduino, il n'a plus besoin d'être relié à un ordinateur pour calculer et afficher le coefficient de corrélation, il est donc autonome, il a seulement besoin d'être relié à une pile. De plus, le code peut être amélioré à tout moment, il suffit simplement de brancher la carte à un ordinateur équipé du logiciel pour effectuer la mise à jour en quelques secondes.