REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET

DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE A.MIRA DE BEJAIA

FACULTE DE LA TECHNOLOGIE

Département Automatique, Télécommunication et Electronique

Projet de fin d'études en vue de l'obtention d'un Master en Electronique

Option Automatique

Thème

Etude et Réalisation d'une Carte Arduino

Présentation:

Mr. DJAFRI Menad

Mr. CHELOUCHE Djalal

Encadreur :

Mr. HADJI Slimane

Membres des Jury de Soutenance :

Président : Mme BELLAHCENE.N.

Examinateur : Mrs GUENOUNOU. O.

Date de Soutenance le 20/06/2016

Remerciements

Nous sommes reconnaissants envers notre encadreur M.Hadji

Nous souhaitons adresser nos remerciements aux personnes qui nous ont apporté leurs aides, spécialement notre camarade et amis Hicham.

Et nous souhaitons remercier spécialements nos parents à qui nous avons dédié ce travail, ainsi que toute notre famille.

Je tiens à adresser mes remerciement à mes parents pour m'avoir supporté, mes sœurs Soraya, Zina et Kouta, ainsi que mes frères Myj et Khelil, et un très grand merci à mon grand frère Faouzi, qui nous a aidé et sans qui rien n'aurait été possible.

De la part de Menad

Je souhaite remercier mes chères parents qui se sont montrés très présents, mes frères et sœurs Imane, Hamza, Yeser, Wasim et Lyna, et toute la famille CHELOUCHE

De la part de Djalal

Table des matières

Liste des figures I

Liste des acronymes et abréviations III

Introduction Générale 01

Chapitre I : L'univers Arduino

I-1) Historique de du projet Arduino 02

I-2) Matériels et dérivées 03

I-3) Domaines et exemples d'utilisation 05

I-3-1) Produits artisanales numériques et machines-outils 05

a) Frida V 05

b) Fraiseuse numérique 06

c) RipRap 07

d) Solar Sinter 07

I-3-2) Projet Pédagogique 08

a) Valise pédagogique 08

Conclusion 09

Chapitre II : Hardware et Softwar10

Introduction 10

II-1) Architecture de la carte (Hardware) 10

II-1-1) Définition du circuit imprimé 10

II-1-2) Schéma synoptique 10

II-1-3) Schéma électrique 11

II-1-4) Composition 12

II-1-4-1) Le noyau 12

II-1-4-2) Le circuit d'alimentation 13

II-1-4-3) L'interface série USB 14

II-1-4-4) Les périphériques 16

II-1-4-5) Circuit Additionnel 17

II-2) Interface de programmation (Software) 18

II-2-1) Installation 18

II-2-2) Description de l'IDE 20

II-2-3) Structure du programme 22

II-2-4) Langage de programmation 23

II-2-5) Compilation et téléversement 23

II-3) Utilisation de la carte 25
II-3-1) Raccordement 25
II-3-2) Circuit de commande 25
II-3-3) Circuit de puissance 25
II-4) Simulation virtuelle de la carte Arduino (Proteus) 26
II-4-1) Présentation du logiciel Proteus 26
II-4-2) Schéma de circuit pour l'exemple Blink 26
II-4-3) Compilation et Téléversement du programme le logiciel Arduino 27
II-4-2) Simulation 27
Conclusion 27

Chapitre III : Réalisation et Test

Introduction 28
III-1) Réalisation sur BreadBord 28
III-1-1) Réalisation du circuit d'alimentation 28
III-1-1-1) Circuit du régulateur de tension L7805CV 28
III-1-1-2) Circuit du régulateur de tension AIC1084-33CM 29
III-1-1-3) Circuit de l'amplificateur Opérationnel LM358 29
III-1-1-4) Circuit du transistor à effet de champ IRF9640 30
III-1-2) Réalisation du circuit de l'interface série (USB) 31
III-1-2-1) Diagramme de l a puce FT232RL 33
III-1-2-2) Communication de données 33
III-1-2-3) Principe de la communication de données 34
III-1-3) Circuit du Cœur de la carte 34
III-1-4) Maquette de la carte Arduino Uno sur BreadBoord 35
III-2) Test de la maquette 35
III-3) Réalisation de la carte Arduino 38
III-3-1) Schéma de la carte Sur Proteus ISIS 38
III-3-2) Sur Proteus ARES 38
III-3-3) Vue 3D de la carte Arduino 39
III-3-4) Imprimé sur du papier calques 39
III-3-5) Circuit Imprimé 39
III-3-6) Soudure 40

III-4) Exemples d'application 40
III-4-1) Blink 40
III-4-2) Afficheur LCD 41
III-4-3) Maquette polyvalente 43
Conclusion 49
Conclusion Générale 54
Annexes : Structure et Architecture des PICs et Datasheet des Semi-conducteurs
Annexe 1 :L'ATMega328P 51
Annexe 2 : L'ATMega16U2 60
Annexe 3 : La puce FT232RL 61
Annexe 4 : Régulateur de tension L7805CV 64
Annexe5 : Régulateur de tension AIC1084-33CM 65
Annexe6 : Amplificateur Opérationnel LM358III-3-6) Soudure 66
Annexe7 : Transistor à effet de champ IRF9640.....67
Annexe8 : Pont H L293D 68
Références.

Liste des figures

Figure(I- 1) : Frida……6

Figure (I-2) : Fraiseuse Numérique……6

Figure (I-3) : Imprimante 3D (RepRap) ……7

Figure (I-4) : Solar Sinter……8

Figure(I- 5) : Valise Pédagogique……8

Figure (II-1) : Schéma Synoptique de la carte Arduino……10

Figure (II- 2): Schéma Electrique de la carte Arduino Uno ……11

Figure (II- 3) : Schéma du Noyau……12

Figure(II- 4): Schéma du Circuit d'Alimentation……13

Figure(II- 5): Schéma de l'interface série USB avec le microcontrôleur ATMega16U2……14

Figure(II- 6) : Schéma de l'interface USB avec la puce FT232RL……14

Figure(II- 7): Les Broches d'entrées/sorties……16

Figure (II- 8): Les Broches d'entrées Analogiques……16

Figure (II- 9): Les Broches d'Alimentation……17

Figure (II- 10): L'ICSP……17

Figure (II- 11): Icone d'Installation……18

Figure (II- 12): Mettre à jour le pilote……18

Figure (II- 13): Sélectionner le pilote……19

Figure (II- 14): Pilote mis à jour……19

Figure (II- 15): IDE……20

Figure (II- 16): Barre d'Action……21

Figure (II- 17): Message d'Erreur……21

Figure (II- 18) : Définition des constantes et des variables……22

Figure (II- 19): Configuration des Entrées/Sorties……22

Figure (II- 20): Programmation des interactions et comportements……23

Figure (II- 21): Compilation……23

Figure (II- 22): Sélection de la carte et du port……24

Figure (II- 23): Schéma du Circuit……26

Figure (II- 24): Compilation sur la carte……27

Figure (II- 25): Simulation. ……27

Figure(III-1) : Circuit du Régulateur L7805CV……28

Figure (III-2): Circuit et Schéma du Régulateur AIC1084-33CM (3.3V) ……29

Figure (III-3): Circuit et Schéma de l'Amplificateur Opérationnel LM358……29

Figure (III-4): Circuit et Schéma du Transistor à effet de champ IRF9640……30

Figure (III-5): Adaptateur USB FTDI. ……31

Figure (III-6): Connexion USB/FT23……32

Figure (III-7): Connexion FT232RL/ATmeg3……32

Figure (III-8): Diagramme de la puce FT232RL……33

Figure (III-9): Maquette de la carte Arduino Sur BreadBord……35

Figure (III-10): Programme Blink sur l'IDE Arduino……36

Figure (III-11): Sélection de la carte Arduino……36

Figure (III-12): Sélection du Port……37

Figure (III-13): Test de maquette avec l'exemple Blink……37

Figure (III-14): Schéma Electrique de la Carte Arduino Sur ISIS……38

Figure (III-15): Circuit de la Carte Arduino sur ARES……38

Figure (III-16): Vue 3D de la Carte Arduino……39

Figure (III-17): Imprimé du circuit ARES sur Du papier Calque……39

Figure (III-18): Circuit Imprimé de la Carte Arduino ……40

Figure (III-19): Le résultat final de la carte Arduino……40

Figure (III-20): Test de la carte avec le programme Blink ……41

Figure (III-21): Programme MizBang injecté dans notre carte Arduino……41

Figure (III-22): Afficheur LCD GDM1602A……42

Figure (III-23): Circuit de l'exemple de Afficheur LCD……42

Figure (III-24): Test de l'exemple Afficheur LCD……43

Figure (III-25): Organigramme de la Maquette Polyvalente ……44

Figure (III-26): Schéma de la Maquette Polyvalente……45

Figure (III-27): Vue de l'application……45

Figure (III-28): Programme "car" de la maquette polyvalente……48

Figure (III-29): Test de la Maquette polyvalente……48

Liste des acronymes et abréviations:

3D: Trois Dimensions.

3V3: 3.3 Volts.

ADK: Android developement Kit.

ADN: Acide désoxyribonucleique.

ALU: Arithmetic Logic Unit

Ampli-op: Amplificateur Opérationnel.

ARef: Alimentation de Reference.

BT: Bluetooth

CAN: Concerter Analogic Numeric

CMOS: complimentary Metal Oxyde Semiconductor.

CMS: Composant Monté en Surface.

D: Drain.

DB9: c'est juste une numérotation débile.

DIL: Dual In Line.

DTR: Data Terminal Ready.

EEPROM: Electrically Erasible Programmable Read-Only Memory.

E/S : Entrée/Sortie.

FM: Fréquence Moyenne.

FTDI: Future Technology Device International.

G: Gaine.

GND: Ground(mass).

GPS: Global Position System.

ICSP: In-Circuit Serial Programming.

IDE: Integrated Development Environment.

IDII: Interaction Design Institute Ivrea.

ISP: In System Programmer.

Ko: Kilo Octet.

Labs: Laboratoires.

LCD : Liquid Crystal Display

LED: Lighting Electricaly Diode.

MIPS: Million d'Instruction Par Seconde.

MIT: Massachusetts Institue of Tehnology .

PWM: Pulse Width Modulation.

RISC: Reduced Instruction Set Computer.

RX: Receiving X.

S: Source.

SIE: Serial Interface Engine..

SPI: Serial Peripheral Interface.

SRAM: Static Random acces Memory.

V: Volt.

Vin: Volt In.

VJin: Video Joking.

Vou : Volt Out.

TTL: Transistor Transistor Logic.

TX: Transfert x.

UART/USART: Universal asynchronous reiciving transmiting.

UMLS: United Medical Language System.

USB: Universal Serial Bus.

USBDM/DP: USB Data Minus/Plus.

VCC: Voltage.

WRT: Wireless RouTer.

Introduction Générale

Introduction Générale

La carte électronique est un assemblage de composants sur une plateforme dite circuit imprimée, elle réalise les fonctions de la plupart des appareils électroniques qu'on trouve dans notre vie quotidienne. Elle a fait son apparition au début des années 50, la période où on a appris à miniaturiser les composantes qui la constituent.
La carte électronique est dédiée à des fonctions précises, ainsi dans un PC on retrouve des cartes modem ou des cartes réseau, des cartes processeur, des cartes mémoire, des cartes son,...etc.

Un autre genre de carte a vu le jour au cours des années 2000, des plateformes de prototypage d'objet interactif à usage créatif appelée Arduino, elle couvre quasiment tous besoins des applications courantes. La carte Arduino est basée sur un microcontrôleur *Atmel* de la famille *AVR*.

Elle se caractérise par sa simplicité et sa variété d'utilisation tel que l'électronique industrielle et embarquée, le modélisme, la domotique mais aussi dans des domaines non technique comme l'art contemporain ou le spectacle, et autre.

Les questions posées ici sont : c'est quoi exactement cette carte ?, de quoi est elle constituée ?, comment fonctionne t-elle?, dans quel domaine on l'utilise?, et comment la réaliser ?

1. Objectif de projet

Dans ce projet trois objectifs sont visés :

- Le premier est de regrouper suffisamment d'information sur une grande catégorie de carte d'interfaçage (Arduino) : son langage de programmation, sa construction.
- Etudier brièvement le fonctionnement de la carte.
- Réaliser une carte Arduino UNO.
- Expérimenter la carte avec quelques exemples d'applications.

2. Présentation du mémoire

Le premier chapitre donne une idée générale sur l'environnement Arduino, sans trop d'introspection, il englobe le parcours, les domaines d'utilisations, ainsi que la famille Arduino, avec une petite touche nostalgique sur ses débuts.

Dans le deuxième chapitre, on mettra la lumière le hardware et le software, on parlera sur les différents composants qui constituent la carte ainsi que leurs rôles, puis on parlera sur l'IDE, l'installation, méthode de programmation, compilation, …etc. Ensuite on donnera une idée générale sur le mode de fonctionnement, et comment se fait le raccordement. Pour finir, on illustrera un petit exemple virtuel.

Quant au dernier chapitre, il se consacre à la réalisation de notre carte, d'abord sur BreadBord, puis on passera vers la conception sur ordinateur avec Proteus pour réaliser le circuit imprimé, et enfin la soudure des composantes. Après la réalisation de la carte, on la testera avec des petits exemples d'applications.

L'architecture interne des microcontrôleurs, puces, et le Datasheet des semi-conducteurs utilisés lors de la réalisation de la carte, seront décrits dans une annexe.

Chapitre I :
L’univers Arduino

I-1) Historique du projet Arduino

Au début des années 2000, L'initiateur et cofondateur du projet, **Massimo Banzi**, enseignait la physique informatique dans l'école *Interaction Design Institute Ivrea* (**IDII**) [1], dans le nord-ouest de l'Italie, il apprenait aux étudiants comment utiliser l'électronique pour prototyper des objets de robotique en utilisant des cartes électroniques assez compliquées, conçues à l'origine pour des ingénieurs, et compatible uniquement avec Windows. [5]

Comme beaucoup de ses collègues, **Banzi** se reposait sur le ***BASIC Stamp*** [2], un microcontrôleur créé et utilisé par l'entreprise californienne ***Parallax*** [3], depuis près de 10 ans. Codé avec le langage *BASIC,* le *Stamp* était un petit circuit, embarquant essentiellement : une alimentation, un microcontrôleur, de la mémoire et des ports d'entrée/sortie pour y connecter du matériel. Mais il n'avait pas assez de puissance de calcul, et il était relativement cher, de plus, **Banzi** avait besoin de quelque chose qui puisse tourner sur Macintosh, qui se trouvait être omniprésent à l'école. [5]

En **2001, Casey Reas** et **Ben Fry**, deux anciens étudiants du ***MIT***[2], avaient développé un langage de programmation intuitif du nom de ***Processing*** [4]. *Processing* gagna rapidement en popularité, parce qu'il permettait aux programmateurs sans expérience de créer des infographies[3] complexes dans un environnement de développement extrêmement facile à utiliser, **Banzi** se demanda s'il pourrait créer un logiciel similaire pour programmer un microcontrôleur, plutôt que des images sur l'écran. [5]

Un peu plus tard, en 2003, **Hernando Barragan**, pour sa thèse de fin d'étude, développa un prototype de plateforme, le ***Wiring*** [6], qui comprenait un environnement de développement facile à appréhender et une carte électronique prête à l'emploi. C'était un projet prometteur encore en activité à ce jour. Mais **Banzi** pensait déjà plus grand, il voulait faire une plateforme encore plus simple, moins chère et plus facile à utiliser.
Il convia donc **David Cuartielles**, un confrère qui enseignait en Suède, ensembles, ils réalisèrent la première carte à usage pédagogique. Deux autres étudiants de **Banzi** se joignirent au projet, **Nicholas Zambetti** et **David Mellis**, ce dernier devint cofondateur et développeur en chef de la partie logicielle d'Arduino, et ensemble ils ont commencé à développer le logiciel en s'inspirant du langage *Processing* et de la carte *Wiring*. [5]

L'objectif étant de mettre au point une plateforme rapide et facile d'accès, l'équipe Arduino croyaient fermement en l'open source, puisque elle a longtemps été utilisée pour aider à l'innovation logicielle. Ils ont aussi eu recourt à une licence ***Creative Commons*** [7], une organisation à but non-lucratif dont les contrats sont habituellement utilisés pour les travaux artistiques.

Grâce à ces deux perspectives, ile entamèrent en construisant et en remettant 300 circuits imprimés vierges (sans composants) aux étudiants de l'*IDII* avec une consigne simple : regarder

[1] *IDII* : L'ecole ' Interaction Design Institute Ivrea ' (IDII) est aujourd'hui située à Copenhague sous le nom de « Copenhagen Institute of Interaction Design
[2] *MIT* 'Massachussetts Instutut of Technology' université de technologie du Massachussetts, aux états unis d'Amérique.
[3] *Infographie* : Domaine de la création d'images numériques liée aux arts graphiques, assistée par ordinateur.

les instructions de montage mise en ligne, construire sa propre carte et l'utiliser pour faire quelque chose. Un des premiers projets était un réveil fait maison suspendu au plafond par un câble. Chaque fois qu'on pousse le bouton snooze[1], le réveil montait plus haut d'un ton railleur jusqu'à ce qu'on ne puisse plus se rendormir. [8]

Rapidement, l'histoire d'Arduino se répondait sur la toile. Elle attira l'attention de **Tim Igoe**, connu pour être le premier à avoir mis l'ensemble de ses outils pédagogiques en ligne pour accès libre au public. **Gianluca Martino** fut la dernière pièce du puzzle, un ingénieur d'*Ivrea* qui a aidé à l'industrialisation de la production d'Arduino. Le premier prototype commercialisé était au cours de l'année 2005, un modèle basé sur l'ATMega8, un microcontrôleur *Atmel* [9] de la famille *AVR*. [5]

Pour accélérer l'adoption d'Arduino, l'équipe cherchait à l'ancrer plus profondément dans le monde de l'éducation. Plusieurs universités, dont ***Carnegie Mellon*** et ***Stanford***, utilisent déjà Arduino. **Mellis** a observé comment les étudiants et les profanes abordaient l'électronique lors d'une série d'ateliers au ***MIT Media Lab***. **Mellis** a ainsi invité des groupes de 8 à 10 personnes à l'atelier où le projet à réaliser devait tenir dans une seule journée. Parmi les réalisations, on peut noter des enceintes pour ***iPod***, des radios FM, et une souris d'ordinateur utilisant certains composants similaires à ceux d'Arduino. En 2010, un de ces projets fut dévoilé au congrès de New York, leur première carte à processeur 32 bits, une puce *ARM* [2] à la place du processeur 8 bits. Cela a permis de répondre à la demande de puissance des périphériques plus évolués comme l'imprimante 3D à monter soi-même, basée sur l'Arduino. [10]

D'autres coups accélérant ont eu lieu cette année, comme le *kit de développement d'accessoires d'Android (ADK)*[3], initié par *Google* qui permet à un téléphone sous *Android* d'interagir avec des moteurs, des capteurs et autres dispositifs. On trouve aussi *des alcootests*, *des cubes à LED*, *des systèmes de domotique* [4], *des afficheurs Twitter et même des kits d'analyse ADN basés sur Arduino*. [11]

La petite carte est désormais devenue le couteau suisse de nombreux artistes, passionnés, étudiants, et tous ceux qui rêvaient d'un tel gadget. Plus de 250 000 cartes Arduino ont été vendues à travers le monde — sans compter celles construites à la maison.

I-2) Matériel et Dérivées [12]

Diffuser la bonne parole d'Arduino n'était qu'une partie du travail. L'équipe a due aussi répondre aux requêtes pour les cartes. En fait, la plateforme Arduino ne se résume plus aux deux types de carte déjà citées — il y a maintenant toute une famille de cartes. En plus du design originel :

[1] *Bouton snooze :* est un bouton qui déclenche un arrêt momentané.
[2] *ARM* : Architectures Matérielles RISC, Processeur 32/64 bits utilisés dans les téléphones portable et les Tablettes
[3] *ADK* :Anroid Kit of Developpment
[4] *Système* ***domotique*** est l'ensemble des techniques de l'électronique, de physique du bâtiment, d'automatisme, de l'informatique et des télécommunications utilisées dans les bâtiments.

Une vinteine de cartes Arduino ont été produites et vendues dans le commerce à ce jour, les plus connues sont citées ci-dessous :

Arduino	**Microcontrôleur**	**Flash ko**	**E/S numériques**	**PWM**	**Entrée analogique**	**Type d'interface USB**
Diecimila	ATmega168	16	14	6	6	FTDI
Due	Atmel SAM3X8E	512	54	12	12	ATmega16u2
Duemilanove	ATmega168/328P	16/32	14	6	6	FTDI
Esplora	ATmega32U4	32	N/A	N/A	N/A	ATmega32U4
Fio	ATmega328P	32	14	6	8	Aucune
Leonardo	ATmega32U4	32	20	7	12	ATmega32U4
LilyPad	ATmega168V or ATmega328V	16	14	6	6	Aucune
Mega	ATmega1280	128	54	15	16	FTDI
Mega2560	ATmega2560	256	54	15	16	ATmega8U2
Nano	ATmega168	16	14	6	8	FTDI
Uno	ATmega328P	32	14	6	6	ATmega16U2
Zero	ATSAMD21	256	14	12	6	ATmega32U2

Notre projet d'étude est principalement basé sur l'Arduino Uno et la Décimila.

I-3) Domaines et exemples d'utilisation

Véritable plate-forme logicielle et matérielle de création d'objets numériques, Arduino permet de programmer des circuits électroniques qui interagissent avec le milieu qui les entoure.

Connectée notamment à des capteurs sonores, thermiques, de mouvement, ces circuits électroniques peu coûteux, dénommés microcontrôleurs, elle peut en retour générer des images, actionner un bras articulé, envoyer des messages sur Internet,… etc. Des dizaines de milliers d'artistes, de designers, d'ingénieurs, de chercheurs, d'enseignants et même d'entreprises l'utilisent pour réaliser des projets dans de multiples domaines :

- **prototypage rapide de projets innovants utilisant l'électronique**, Arduino facilitant l'expérimentation en amont de la phase d'industrialisation ;
- **production artisanale d'objets numériques et de machines-outils à faible coût** dans la perspective d'une culture d'appropriation technologique favorisant le bricolage et la débrouille ;
- **captation et analyse de données scientifiques** (environnement, énergie, etc.) à des fins éducatives, de recherche ou d'appropriation citoyenne ;
- **spectacle vivant**, grâce aux nombreuses fonctions d'interaction offertes par Arduino, il est possible de créer des performances de VJing[1], utiliser le mouvement des danseurs pour générer en temps réel des effets sonores et visuels dans un spectacle ;
- **installations d'arts numériques**, Arduino permettant de réaliser des œuvres d'art interagissant de manière autonome avec le public ;
- **Mode et design textile**, plusieurs stylistes et designers investissant ce domaine créatif en exploitant les possibilités offertes par l'intégration de l'électronique notamment dans des vêtements (e-textile) ;
- **Projets pédagogiques** à destination d'étudiants, de professionnels ou du grand public selon les porteurs de ces initiatives : écoles supérieures, centres de formation spécialisée ou des **Media Labs**[2]. [13]

Pour bien illustrer l'utilité de la carte Arduino, voici quelques productions :

I-3-1) Produits artisanales numériques et machines-outils :

a) Frida V

Fixé sur un vélo, Frida V prend la forme d'un boîtier contenant un ensemble de capteurs qui analysent et compilent les informations récoltées durant le parcours d'un cycliste : pollution atmosphérique (capteurs de gaz), wifi ouvert dans la ville (routeur wifi), position géographique (GPS), documentation audiovisuelle des parcours (caméra et microphone). L'ensemble est assemblé dans une coque résistante et portable adaptable sur tout type de vélos.

Les données recueillies sont archivées sur un serveur et représentées graphiquement sur une carte, la **MeTaMap,** un programme configurable qui sert à cartographier un texte biomédical,

[1] *VJing :* Video Jockey (ing) Le VJing est un terme large qui désigne la performance visuelle en temps réel.
[2] *Media labs :* Terme qui signifie un laboratoire dédié à la recherche dans le domaine technique et à l'apprentissage.

basé sur le langage UMLS[1]. Ces informations peuvent être annotées par les participants en direct grâce aux contrôleurs de la Frida. La Frida est constituée notamment d'un routeur Linux embarqué et d'un système de micro-Arduinos. Les logiciels sont développés en libre et adaptables sur de nombreuses autres plates-formes matérielles de type Smartphone [14]

Figure (I-1) : Frida

b) Fraiseuse numérique

Arduino peut agir comme un automate de commande pour piloter une machine à commande numérique fait-maison, qui coûte bien moins cher qu'un modèle acheté. Cette fraiseuse est inspirée du modèle canadien « Oomloot », adapté aux mesures métriques et réalisée durant le bootcamp « Make ta machine » à Nantes au printemps 2011.

Cette machine peut réaliser des gravures, des découpes et des usinages dans divers matériaux (bois, aluminium, circuits électroniques, etc.). [14]

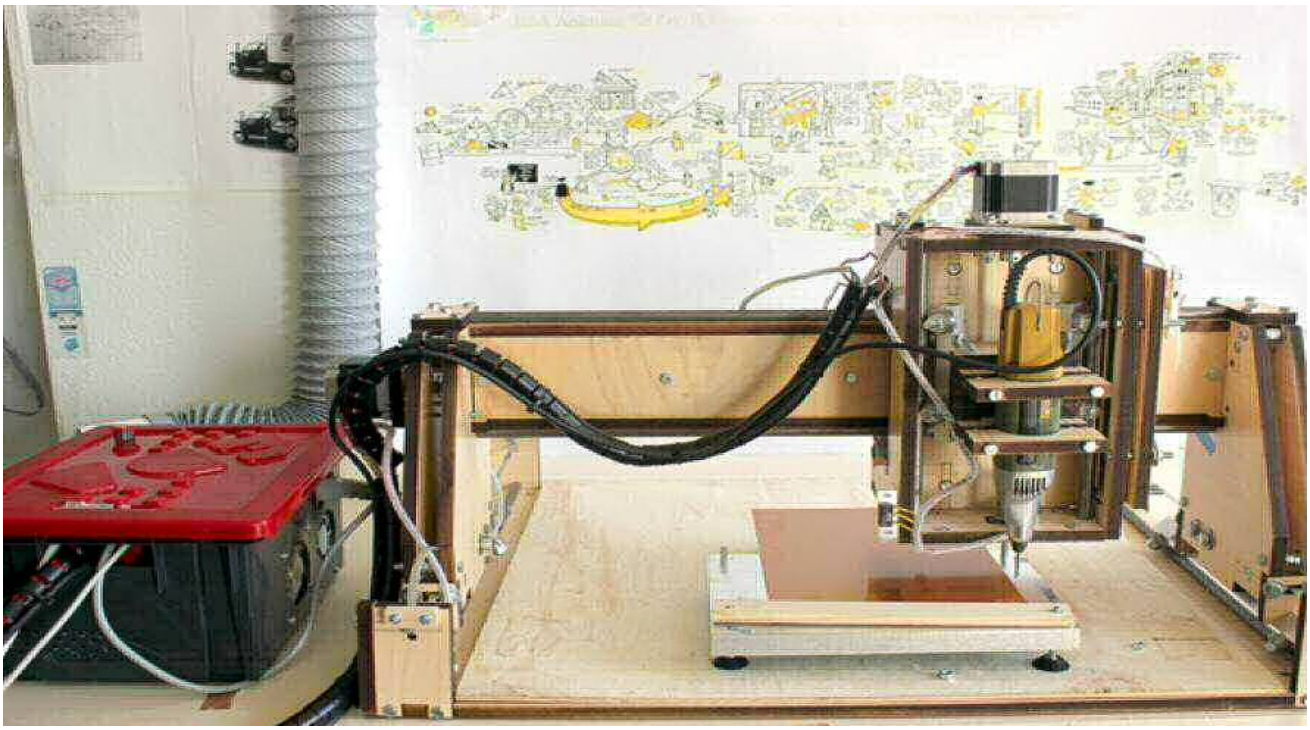


Figure (I-2) : Fraiseuse Numérique

[1] *UMLS:* United Medical Language System.

c) RepRap

RepRap est une imprimante de bureau 3D libre capable de fabriquer des objets en plastique. Comme la plupart des pièces de la RepRap sont faites de cette matière et que la RepRap peut les créer, cette machine est considérée comme « auto-réplicable ». RepRap s'inscrit dans le mouvement du matériel libre et de la fabrication de machines-outils à faible coût. L'absence de brevet et la mise à disposition gratuite des plans offrent la possibilité à quiconque de la construire avec du temps et un minimum de matériel. Cela signifie également que si on en possède une, on peut réaliser beaucoup de choses utiles, voire créer une autre RepRap.

Projet en constante amélioration, la RepRap existe en plusieurs modèles. L'électronique de pilotage de la RepRap est basée sur Arduino. [14]

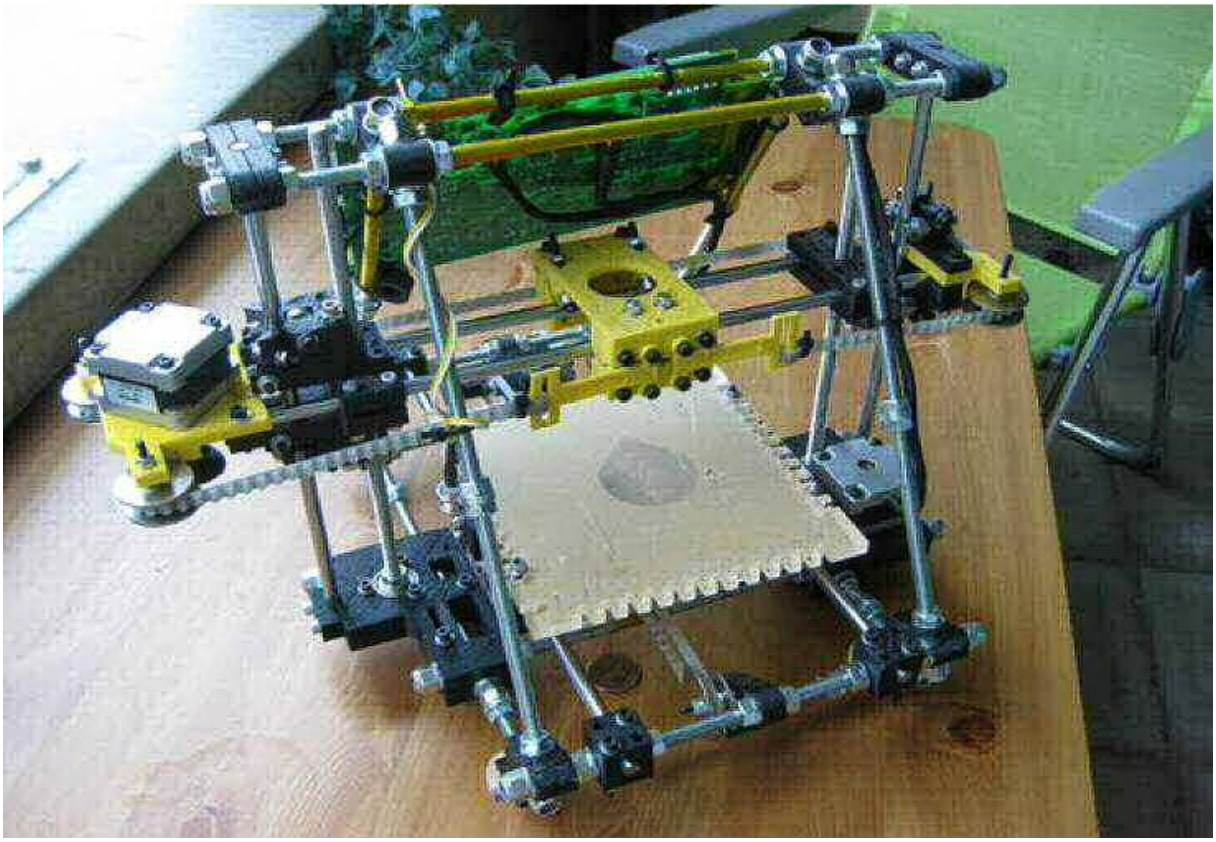


Figure (I-19) : Imprimante 3D (RepRap)

d) Solar Sinter

Solar Sinter Adoptant une approche originale où les rayons du soleil remplaçant la technologie laser et le sable divers produits chimiques utilisés par certaines imprimantes 3D, Solar Sinter est une machine à énergie solaire permettant de fabriquer des objets en verre.

Dans les déserts du monde, deux éléments dominent : le soleil et le sable. Le premier offre une source d'énergie potentielle immense, le second un approvisionnement presque illimité en silice sous forme de sable. Le sable de silice, lorsqu'il est chauffé à son point de fusion puis refroidi, se solidifie en formant du verre opaque.

L'Arduino sert à suivre le soleil dans son déplacement et à positionner les rayons concentrés par des lentilles optiques sur une surface de sable en fonction de la forme de l'objet désiré. [14]

Figure (I-20) : Solar Sinter

I-3-2) Projet Pédagogique

Plusieurs projets pédagogiques à destination d'étudiants, de professionnels ou du grand public reposent sur l'utilisation d'Arduino.

a) Valise pédagogique création interactive :

Présentation de la valise pédagogique à Kër Thiossane, Villa des arts et du multimédia.
Réalisé dans le cadre d'un projet d'essaimage de pratiques artistiques numériques en Afrique de l'Ouest et dans les Caraïbes (projet Rose des vents numériques) , la Valise pédagogique création interactive est un ensemble matériel, logiciel et documentaire pour l'apprentissage des technologies d'Interaction Temps Réel dans la création contemporaine, tous champs artistiques confondus (arts plastiques, danse, théâtre, musique, architecture, design, etc).
Équipée de deux cartes Arduino, la valise peut servir aussi bien de plate-forme d'apprentissage dans le cadre d'un atelier de découverte de l'interaction en art que d'outil de création pour artiste en permettant d'inventer, de simuler puis de réaliser des milliers de dispositifs interactifs différents.[14]

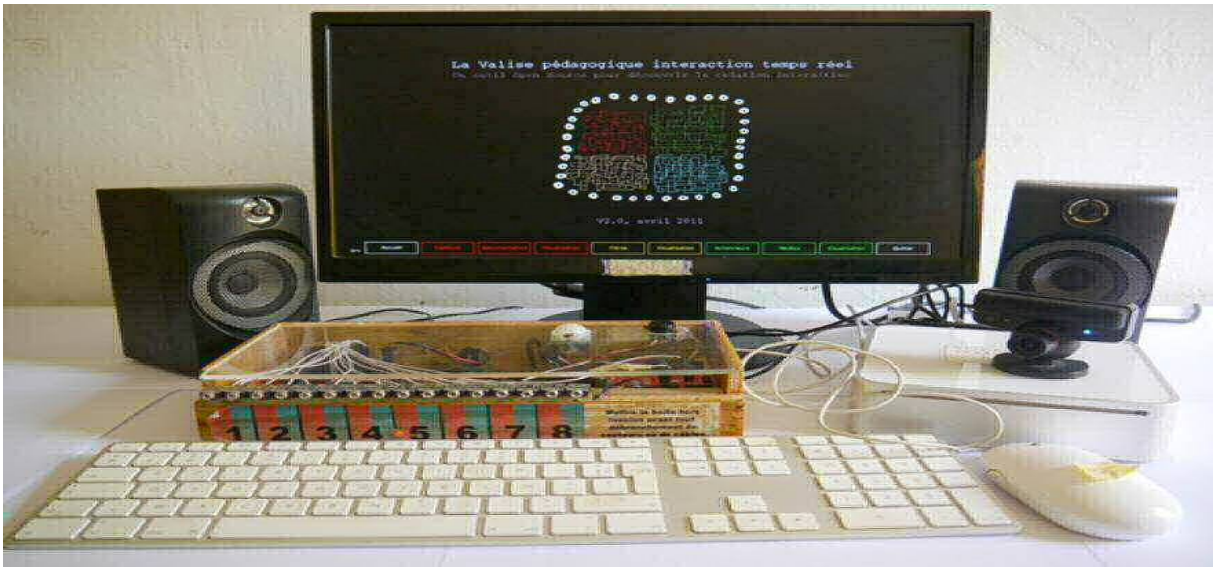


Figure (I-21) : Valise Pédagogique

Conclusion :

Ce chapitre permet d'avoir une vue d'ensemble sur la plateforme Arduino, son parcours ainsi quelques uns de ces modèles et exemples d'utilisation.

Il nous apprend aussi que l'univers Arduino a innové dans le domaine de prototypage d'objet, commençant par la pédagogie puis vers l'utilisation libre en se basant sur son imagination. En plus d'avoir une plateforme riche en carte électronique, il offre plusieurs possibilités d'utilisation.

Chapitre II :
Hardware et Software

Introduction

L'univers Arduino repose sur deux piliers, le premier s'agit de la carte électronique programmable (Hardware), composée de plusieurs composants semi-conducteurs, de circuits intégrés et des périphériques, le deuxième s'agit de l'interface de programmation (Software), qui possède un langage de programmation très spécifique, basé sur les langages C et C++, adapté aux possibilités de la carte.

II-1) Architecture de la carte (Hardware) :

Différents composants électroniques sont nécessaires pour la réalisation de la Carte Arduino, ces derniers sont soudés sur un circuit imprimé, mais en premier lieu, définissant le circuit imprimé :

II-1-1) Définition du circuit imprimé:

Le circuit imprimé est un support plan, flexible ou rigide, généralement composé d'époxy[1] ou de fibre de verre, il possède des pistes électriques disposées sur une ou plusieurs couches (en surface et/ou en interne) qui permettent la mise en relation électrique des composants électroniques. Chaque piste relie tel composant à tel autre, de façon à créer un système électronique qui fonctionne et qui réalise les opérations demandées, ce système porte le nom de carte électronique. [13]

Evidemment, tous les composants d'une carte électronique ne sont pas forcément relies entre eux. Le câblage des composants suit un plan spécifique à chaque carte électronique, qui se nomme le schéma électronique.

En ce qui nous concerne, les composants utilisés forment trois circuits reliés entre eux, un circuit pour l'alimentation, un autre pour la communication avec la partie Software, et le dernier s'agit du noyau, le cœur de la carte. Le schéma synoptique ci-dessous nous donne une idée générale sur l'architecture de la carte Arduino.

II-1-2) Schéma synoptique

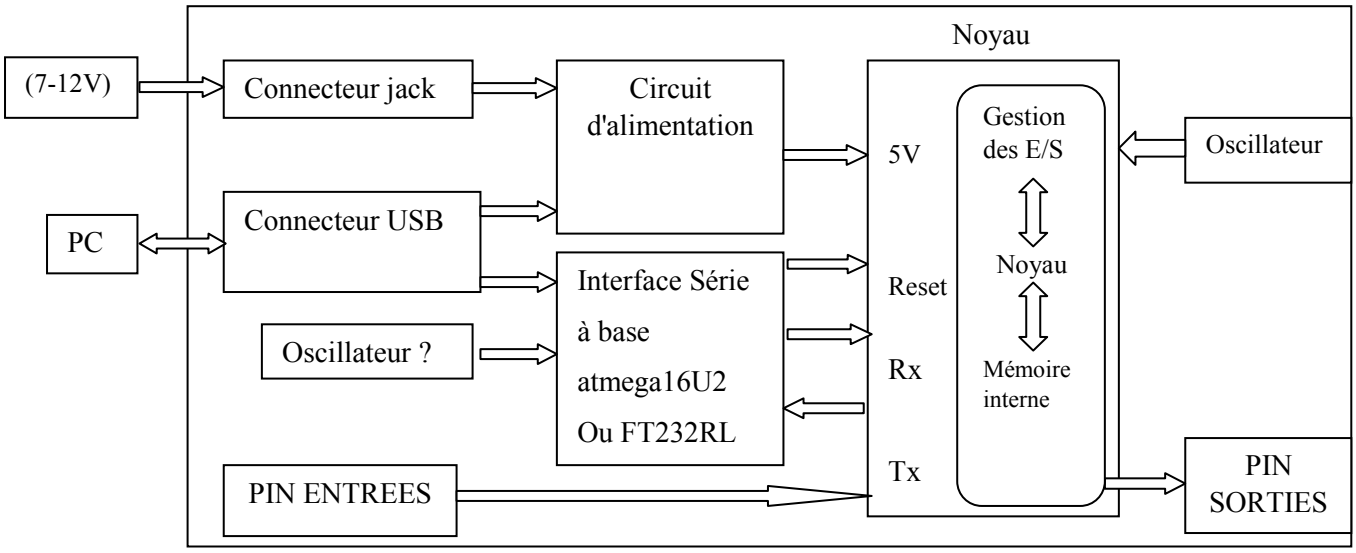


Figure (II-1) : Schéma Synoptique de la carte Arduino

[1] *Epoxy :* abréviation du mot polyépoxyde, un polymère à base d'époxyde (plastique).

II-1-3) Schéma électrique : [16]

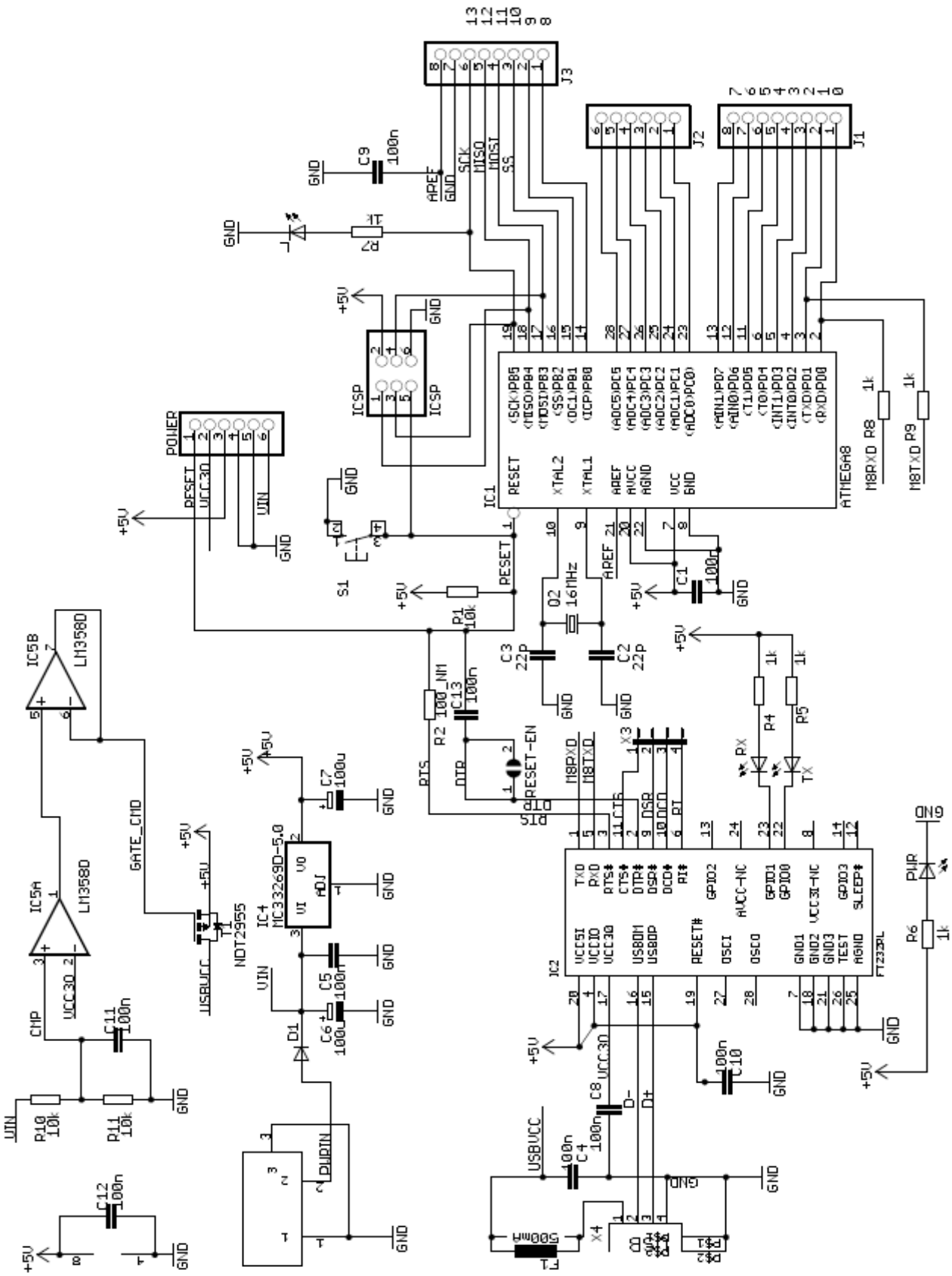


Figure (II-2) : Schéma Electrique de la carte Arduino Uno [14]

Comme nous l'avons brièvement cité dans le *Chapitre I*, notre projet de fin d'études est basé sur la carte Arduino Uno, nous allons alors énoncer des composants qui la constituent :

II-1-4) Composition : [16]

II-1-4-1) Le Noyau :

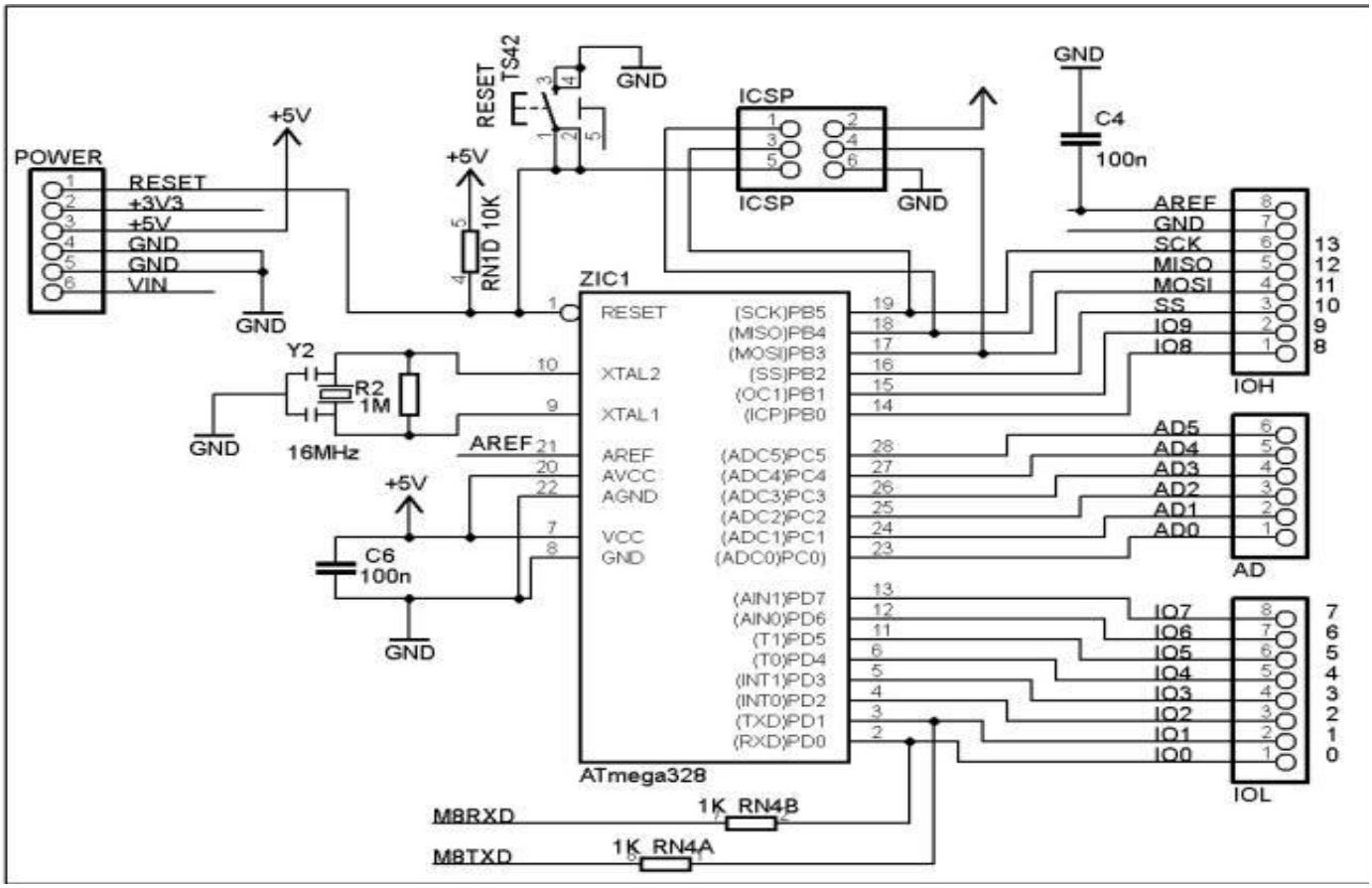


Figure (II-3) : Schéma du Noyau

Le noyau de la carte Arduino Uno est constitué de :

- **Un Microcontrôleur ATMega328** : avec bootloader[1] ;

  Les microcontrôleurs sont des unités de traitement de l'information, il renferme dans le même socle l'équivalent d'un microprocesseur, mais aussi les mémoires de programmes et de données, les périphériques d'E/S[2], les CAN[3], les Timers, le WatchDog,…etc. Pour fonctionner, un programme nommé bootloader doit être chargé et exécuté dans sa mémoire FLASH.

  Plusieurs variétés de microcontrôleurs existent, comme MOTOROLA, INTEL, ou encore MICROCHIP. Le notre s'agit de l'ATMEga328, un microcontrôleur ATMEL de la famille AVR, programmable avec le langage C. Le cœur AVR combine un jeu de 132 instructions avec 32 registres spécieux directement connectés à l'unité d'arithmétique logique (ALU), ces derniers permettent à deux registres indépendants d'y avoir accès avec une seule instruction exécutée en un cycle d'horloge. [16]

  Les registres et les mémoires qui interviennent lors de l'exécution des instructions du programme chargé dans le microcontrôleur sont expliqués plus en détail dans l'Annexe.

[1] *Bootloader :* programme qui permet de lancer le système d'exploitation du µc (comme le bios d'un PC).
[2] *E/S :* Entrées/Sorties.
[3] *CAN :* Convertisseur Analogique Numérique.

- **Un circuit Oscillatoire :** qui est composé de :

→ **Un Quartz 16MHz :** le quartz est un composant qui oscille à une fréquence stable lorsqu'il est stimulé électriquement, il a pour but de fournir une base de temps pour le microcontrôleur.

→ **Deux condensateurs de 22pF** pour le filtrage et la protection électronique, les condensateurs ont la propriété de stocker des charges électriques qui sert en cas de coupure brutale d'alimentation.

II-1-4-2) Le circuit d'alimentation

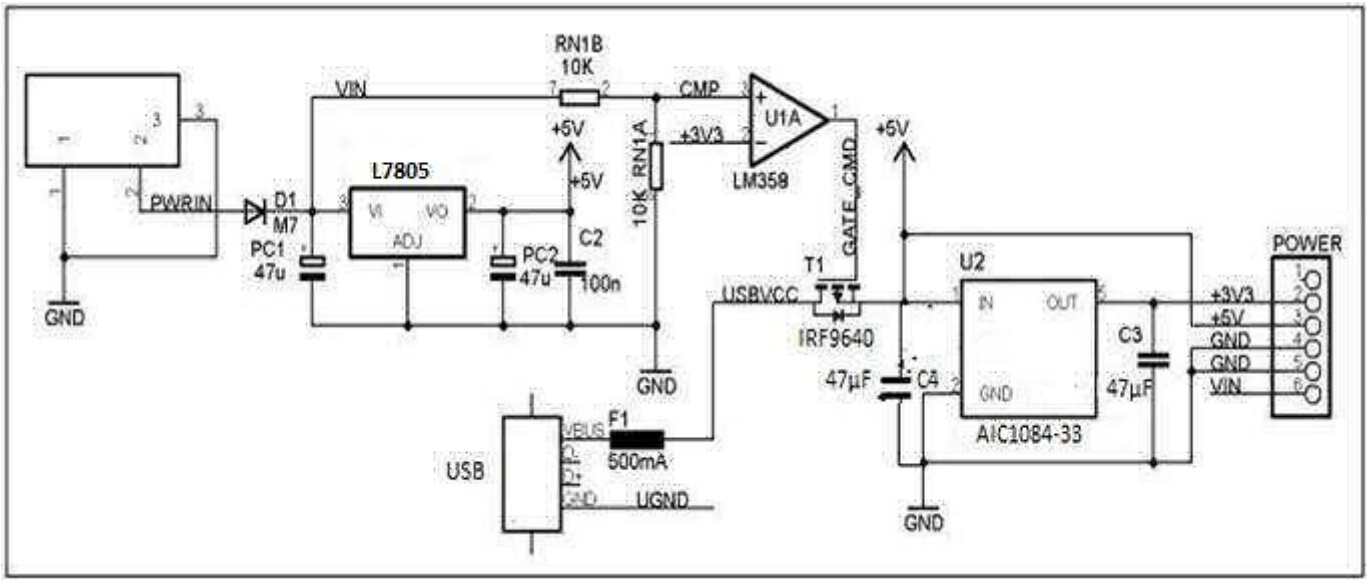


Figure (II-4) : Schéma du Circuit d'Alimentation

Comme son nom l'indique, le circuit d'alimentation sert à fournir de l'électricité pour alimenter la carte, et accessoirement les composants et les circuits externes. Le circuit d'alimentation est constitué de :

- **Un connecteur Jack :** qui achemine une tension Vin depuis la source vers, en premier lieu, à :
- **Une diode :** qui est un dipôle semi-conducteur, le courant passe que d'une seule direction, de l'anode vers la cathode, elle sert à protéger le circuit alimenté en courant continu des erreurs de branchement. Puis :
- **Un Régulateur L7805** qui régule cette tension d'entrée à 5V, Ce régulateur est muni de **deux condensateurs de 47µF** branche sur ses pins d'entrée/sortie, est un autre de **100nF** entre la masse et sa sortie, pour la protection, Sa sortie de valeur de 5V alimente la carte et elle aussi envoyée vers la broche 5V des connecteurs d'alimentation, puis vers :
- **Un Régulateur AIC1084-33** qui régule cette tension à 3.3V pour la broche 3V3 de la carte Arduino, et pour :
- **Un Amplificateur Opérationnelle LM358 :** qui, s'il détecte une différence de potentielle positive entre ces deux broches d'entrées alimentées par 3.3V et Vin/2 (dévisée grâce à circuit diviseur de tension formé par **Deux Résistances de 10KΩ)**, délivre une tension envoyée vers :

- **Un Transistor IRF9640 Canal P, à effet de champ :** qui rempli la fonction d'interrupteur électronique qui sélectionne la source d'énergie. En recevant une tension émise par l'Ampli-op, il sélectionne:
  - → Le connecteur lorsqu'on branche une source d'énergie à la carte (batterie, transformateur ou pile), le système peut fonctionner de manière autonome, ou bien :
  - → Le circuit USB, dans l'absence ou l'insuffisance du premier.

II-1-4-3) L'interface série USB : pour cette partie, nous pouvons la réaliser avec deux puces :

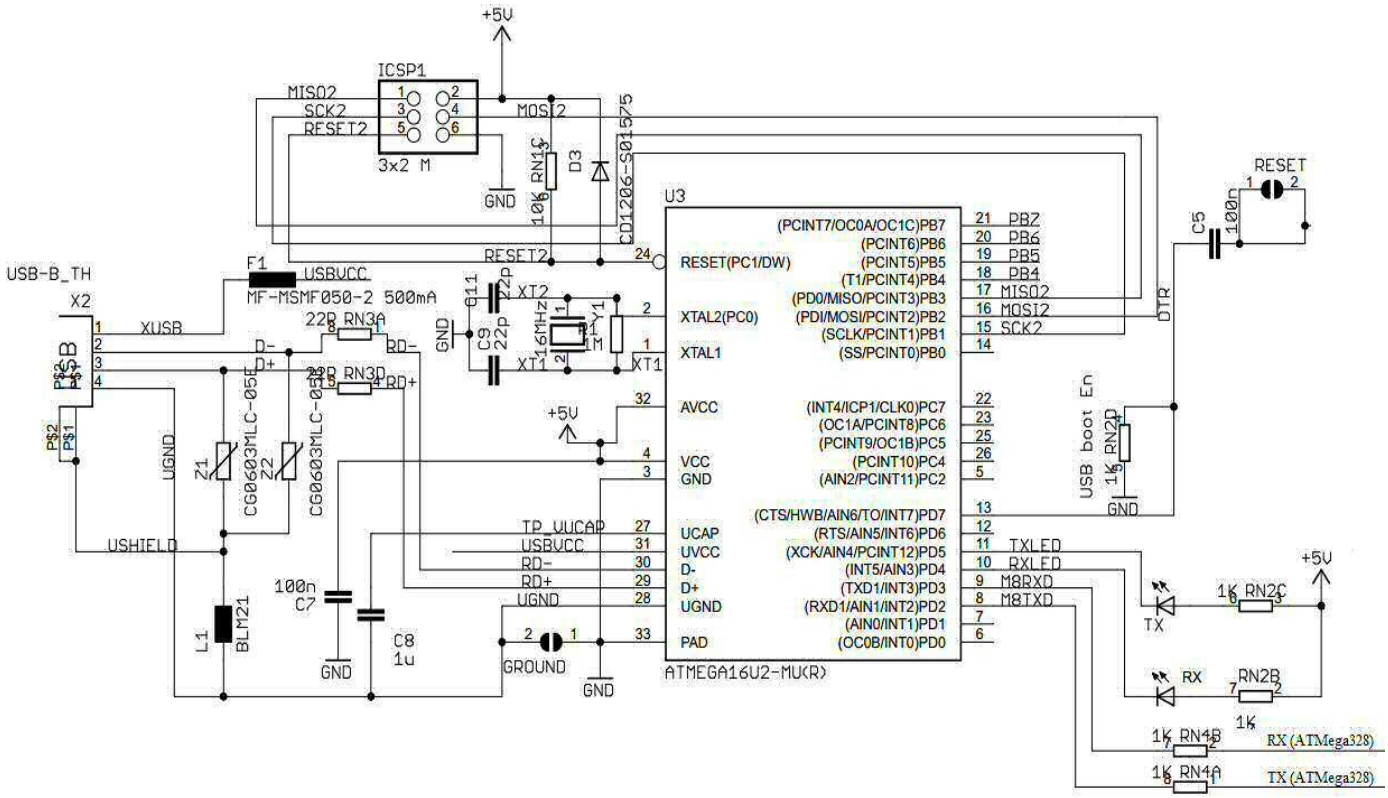


Figure (II-5) : Schéma de l'interface série USB avec un microcontrôleur ATMega16U2

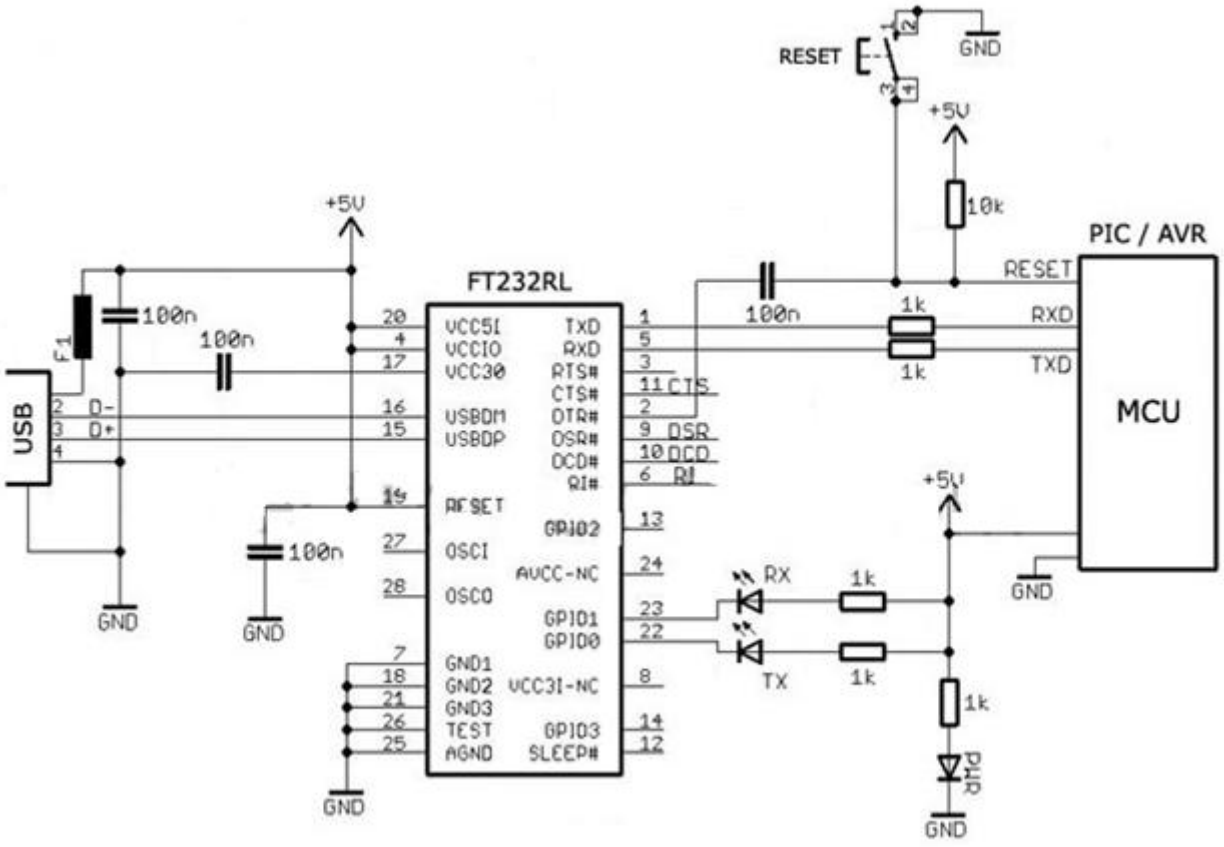


Figure (II-6) : Schéma de l'interface USB avec une puce FT232RL

Les deux circuits ci-dessus font la connexion entre le noyau de la carte et l'interface de programmation, ils assurent aussi l'alimentation de la carte dans le cas de l'insuffisance ou de l'absence du circuit d'alimentation, ils sont constitués de :

- **Un Noyau :** Comme sur les figures précédentes, l'interface série USB peut être composée de deux puces :
  - **Un Atmega16U2 :** il s'agit d'un microcontrôleur 8 bits CMOS[1] à faible puissance, basé sur l'architecture AVR RISC[2] amélioré. En exécutant des instructions puissantes dans un seul cycle d'horloge, l'ATmega16U2 atteint des débits approchant le 1 MIPS[3] par MHz permettant à un concepteur de système d'optimiser la consommation d'énergie par rapport à la vitesse de traitement. Comme pour l'ATMega328P, l'ATmega16U2 a besoin d'un **circuit oscillatoire.**

  - **FT232RL :** c'est une puce FTDI[4], avec une interface série UART, munie du protocole USB 2.0, à l'inverse de l'ATMega16U2, elle n'as pas besoin de circuit oscillatoire vu qu'elle est munie d'une horloge interne de 12MHz, ces caractérist sont détaillées dans l'annexe.

  - ❖ **Remarque :** Dans la partie réalisation, nous avons utilisé la FTDI, nous avons tenté de réaliser le circuit USB avec un microcontrôleur ATMega16U2, mais nous nous sommes confronté à un problème, nous n'avons pas pu injecter le bootloader, car le charger requière un programmateur qui nous faisait défaut. Dans la partie soudure, nous avons réalisé le circuit USB en reprenant les composants de la FTDI.

- **Un connecteur USB :** il est muni de Quatre Broches :
  - → VCC pour la source.
  - → GND pour la masse.
  - → D+ et D- pour la communication de données
  - ❖ Deux résistances et deux varistors[5] sont reliées aux broches D+ et D- pour le filtrage.

- **Un pont Vers le µC ATmega328 :** caractérisé par :
  - → Une double connexion inversée entre les pins RX et TX des deux puces, munie de **Deux résistance de 1KΩ**.
  - → Un relais entre les pins **DTR-Reset** ou **Reset-Reset** des deux puces **(**ça dépend de la puce utilisée**),** pour la réinitialisation, ce relais est doté d'un **Bouton Reset**, **une Résistance de 10KΩ**, **un Condensateur de 100nF**.

[1] *CMOS* : est un sigle anglais qui veut dire semi-conducteur en Oxyde métallique complémentaire
[2] *RISC*: Reduced Instruction Set Computer.
[3] *MIPS*: Million d'Instruction Par Seconde.
[4] *FTDI* : Future Technology Devices International.
[5] *Varistors* :composants électroniques à résistances non linéaire.

II-1-4-4) Les Périphériques :

- **14 broches numériques d'entrées/sorties :**

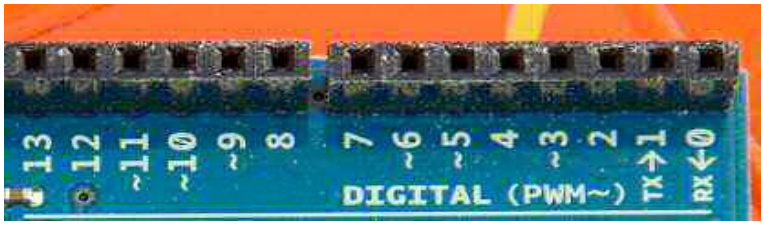


Figure (II-7) : Les Broches d'entrées/sorties

→ **Pins: 0 (RX) et 1 (TX)** sont utilisés pour la réception (RX) et d'émission (TX) des données en série TTL. Ceux - ci sont connectés aux broches correspondantes du port USB-TTL du processeur ATmega16U2 ou la puces FT232RL.

→ **Pins 2 et 3:** peuvent être configurés comme déclencheurs pour des événements externes, tels que la détection d'un front montant ou descendant d'un signal d'entrée.

→ **Pins 4, 5, 6, 9, 10 et 11:** peuvent être configurés via le logiciel avec la fonction **analogWrite ()** pour générer des signaux PWM avec une résolution de 8 bits en l'utilisant comme CAN. On peut grâce à un simple filtre RC obtenir des tensions continues de valeur variable.

→ **Pins 10, 11, 12, 13 :** peuvent être programmés pour réaliser une communication SPI[1], SPI utilise une bibliothèque spéciale.

→ **Pin 13** est relié à une diode interne à la carte, utile pour les messages de diagnostic. Lorsque le niveau de la broche est HAUT, le voyant est allumé, quand le niveau de la broche est faible, il est éteint.

- **6 entrées analogiques** :

Figure (II-8) : Les Broches d'entrées Analogiques

L'UNO dispose de 6 entrées analogiques, étiquetées de 0A à A5, dont chacun fournit 10 bits de résolution (en pratique 1024 valeurs différentes). Par défaut, on peut mesurer une tension de 5V à la masse, mais il est possible de changer l'extrémité supérieure de sa gamme en utilisant la broche AREF[2]. En outre, certaines broches ont des caractéristiques spéciales comme pour les broches numériques largeur d'impulsion. Il s'agit d'un artifice permettant de produire une tension variable à partir d'une tension fixe. La technique s'apparente approximativement à du morse : le signal de sortie est modulé sous forme d'un signal carré dont la largeur des créneaux varie pour faire varier la tension moyenne.

[1] *SPI* : Serial Peripheral Interface : interface série maitre/esclave.
[2] *AREF* : Alimentation de Référence.

- **6 Broches pour l'alimentation :**

Figure (II-9) : Les Broches d'Alimentation

Elle comporte :

→ Une connexion Reset pour la réinitialisation.
→ Une connexion 3V3 qui permet à des circuits de puissance compatible de se connecter à la carte Arduino.
→ Une connexion 5V fournit par le régulateur L7805CV, cette tension est utile pour d'autres circuits électriques compatibles avec 5 volts.
→ Une connexion GND pour la masse.
→ Une connexion Vin renvoie la tension appliquée à partir de la prise d'alimentation et peut être utilisée pour alimenter d'autres circuits qui ont déjà un régulateur de tension (par exemple le bouclier appliqué au module);

- **Un connecteur ICSP**[1] (programmation "in-circuit" est optionnel)

Figure (II-10) : L'ICSP

Dans le cas où le microcontrôleur de la carte Arduino viendrait à ne plus fonctionner, il est possible de remplacer la puce défectueuse. Ainsi le port ICSP permet de configurer une nouvelle puce pour la rendre compatible avec l'IDE Arduino. En effet le microcontrôleur de la carte Arduino possède un bootloader (ou bios) qui lui permet de dialoguer avec l'IDE. Il faut pour cela acheter un programmateur ISP[2] ou il est aussi possible d'utiliser une autre carte Arduino comme programmateur ISP.

II-1-4-5) Circuit Additionnel

Il est possible de spécialiser la carte Arduino en l'associant avec des circuits additionnels ou modules que l'on peut fabriquer soi-même ou acheter déjà montés. Lorsqu'ils se branchent directement sur la carte, ces circuits s'appellent des « Shields » ou cartes d'extension. Ces circuits spécialisés apportent au système des fonctionnalités diverses et étendues dont voici quelques exemples :

- Ethernet : communication réseau ;
- Bluetooth ou Zigbee : communication sans fil ;
- pilotage de moteurs (pas à pas ou à courant continu) ;
- pilotage de matrices de LEDs : pour piloter de nombreuses LEDs avec peu de sorties ;...etc

[1] *ICSP* : In-circuit Serial Programming
[2] *ISP* : In System Programmer : un programmateur AVR qui sert à injecter des programmes directement dans les puces AVR.

II-2) Interface de programmation IDE[1](Software).

II-2-1) Installation : [4]

L'installation de l'interface de programmation Arduino est relativement simple et possible sur les plates-formes Windows, Mac OS X et Linux. L'environnement de programmation Arduino (IDE en anglais) est une application écrite en Java et inspirée du langage *Processing*.

Pour télécharger le fichier d'installation, il suffit de se rendre sur le site officiel Arduino. Pour sélectionner une version, on doit cliquer sur le nom qui correspond à notre système d'exploitation et sauvegarder sur l'ordinateur le fichier correspondant. Il est à noter que la version anglaise contient habituellement des mises à jour plus récentes que la version française.

L'environnement Arduino est en open source, donc facilement téléchargeable, une fois téléchargé avec le lien On le décompresse, puis on le copie sur l'ordinateur, ensuite on lancer l'application Arduino Double -cliquer sur :

Figure (II-11) : Icone d'Installation

L'installation de l'IDE sur les différents systèmes d'exploitations se ressemble beaucoup sauf pour les pilotes (ou driver) requis pour la partie raccordement, en effet :

a) Sous Windows, durant la connexion de la carte interface, il est question de démarrage du processus d'installation du driver, mais pour l'installer, il faut se rendre à **la Gestion de l'ordinateur,** puis sélectionner **Gestion des périphériques,** le pilote non installé sera dans la rubrique **autres périphérique** avec un point d'exclamation en jaune**,** là on devra appuyer sur **mettre à jour le pilote**, une autre fenêtre va s'ouvrir :

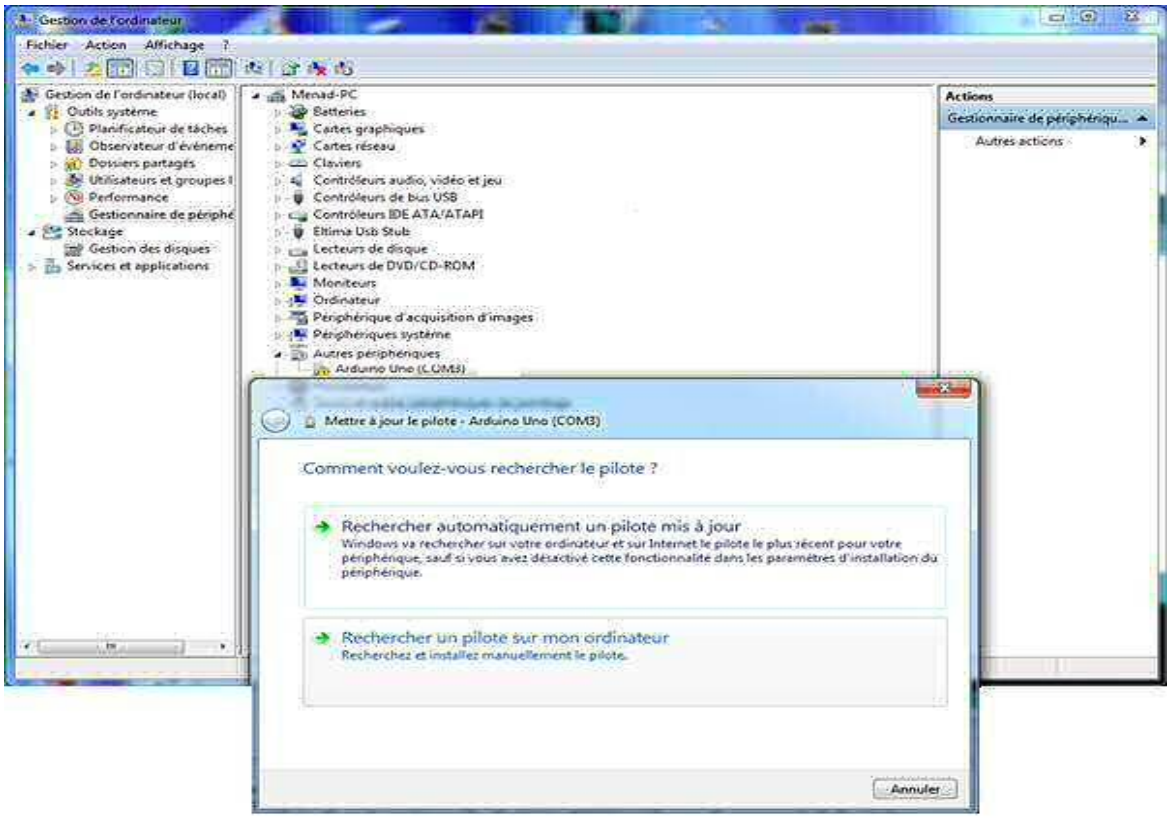


Figure (II-12): Mettre à jour le pilote

[1] *IDE :* Integrated Development Environment.

Là, on clique sur **Rechercher pilote sur mon ordinateur,** puis une autre fenêtre va apparaitre :

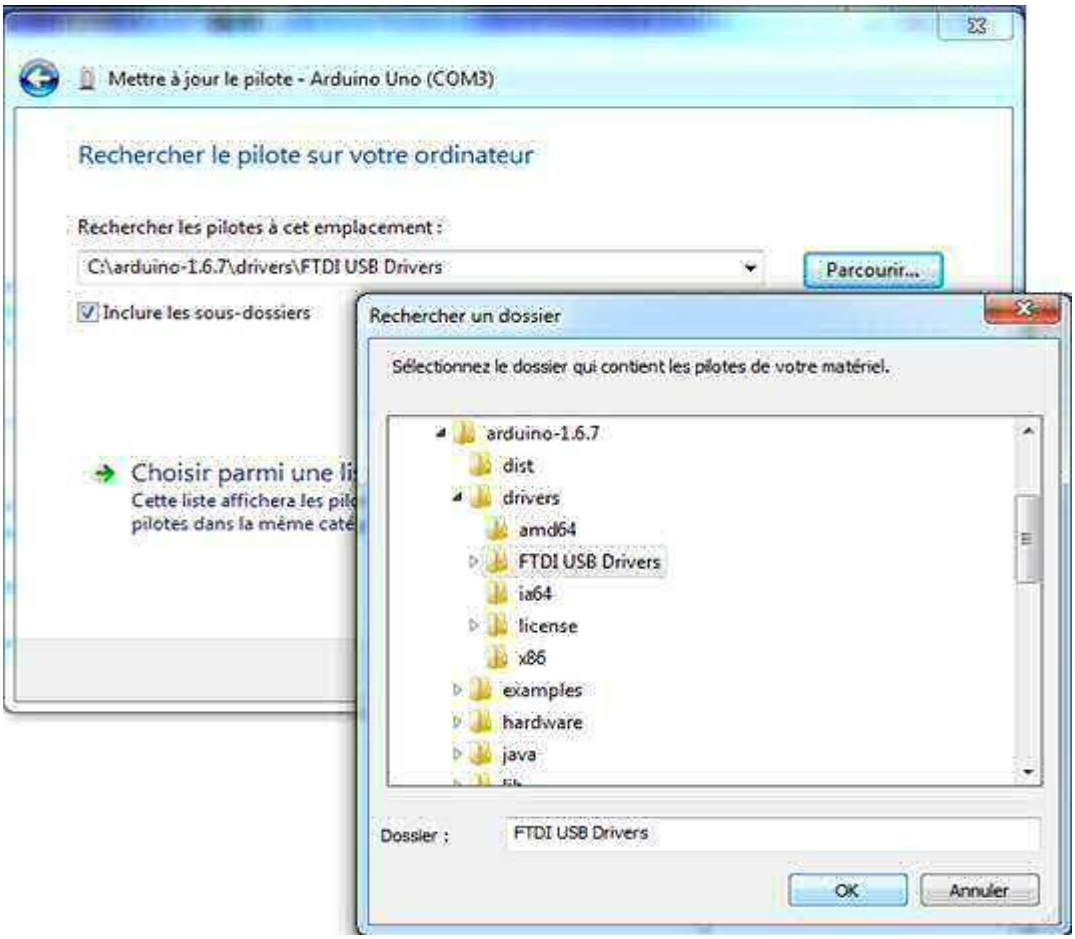


Figure (II-13) : Sélectionner le pilote

On appuis sur **Parcourir…,** puis on ouvre le **dossier Arduino-1.6.7** ou la version de l'IDE installée, puis **Drivers** et en fin, on sélectionne **FTDI USB Drivers**.
Cette fenêtre apparaitra si le pilote est installé correctement

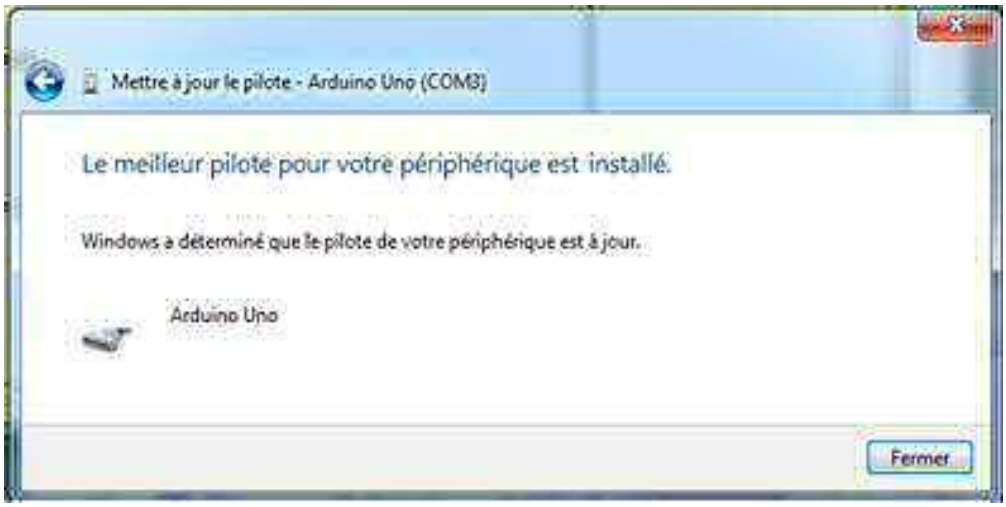


Figure (II-14) : Pilote mis à jour

Si le pilote n'est pas présent dans le dossier de l'IDE, il faut le télécharger.

b) Sous MAC OS X, Les pilotes sont déjà préexistants, mais pour les versions autres que l'UNO et la Mega2560, une nouvelle fenêtre apparaît indiquant qu'une nouvelle interface réseau a été détectée, on doit alors sélectionner « **Préférence Réseau** », à l'ouverture, on clique sur suivant, en fin on peut lancer l'application Arduino.

c) Sous GNU/Linux, La procédure d'installation sous GNU/Linux dépend de la distribution utilisée et l'exécution du programme dépend également de l'installation d'autres pilotes ou logiciels. Pour les utilisateurs de Linux.

II-2-2) Description de l'IDE :

L'IDE est un logiciel de programmation qui permet d'écrire, de modifier un programme et de le convertir en une série d'instructions compréhensibles pour la carte. Il programme par code, contenant une cinquantaine de commandes différentes. A l'ouverture, l'interface visuelle du logiciel contient le menu, des boutons de commande en haut, une page blanche vierge, une bande noire en bas, comme ceci :

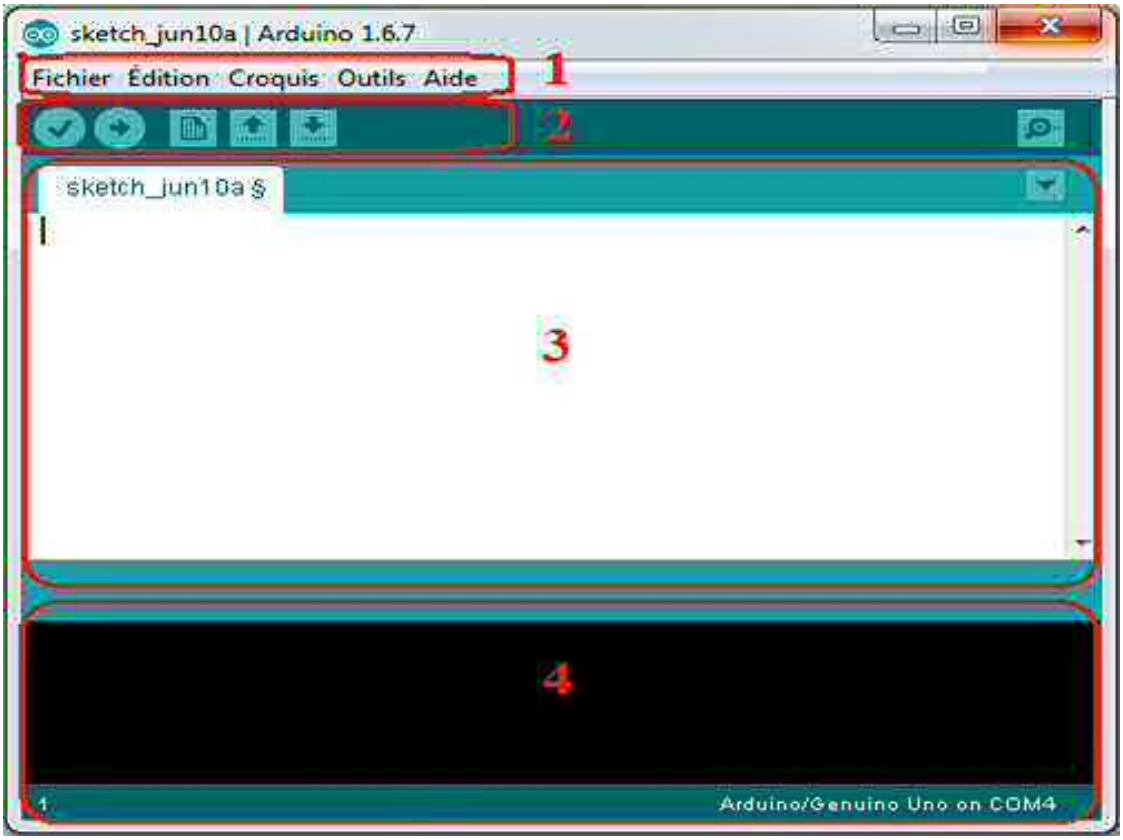


Figure (II-15) : IDE.

1. **Menu :** Les différents éléments du menu permettent de créer de nouveaux sketchs (programmes), de les sauvegarder, et de gérer les préférences du logiciel et les paramètres de communication avec votre carte Arduino. Le menu comprend : **Fichier** : pour créer, sauvegarder en spécifiant la destination, et d'appeler un programme ; **Edition** : Pour couper, copier, coller, supprimer, sélectionner,…etc. ; **Croquis** : regroupe les fichiers réalisées ; **Outils** : pour spécifier le type de la carte, le port série, formater, recharger et réparer l'encodage, graver la séquence d'initialisation, de la carte branchée sur l'ordinateur.

Barre d'action :

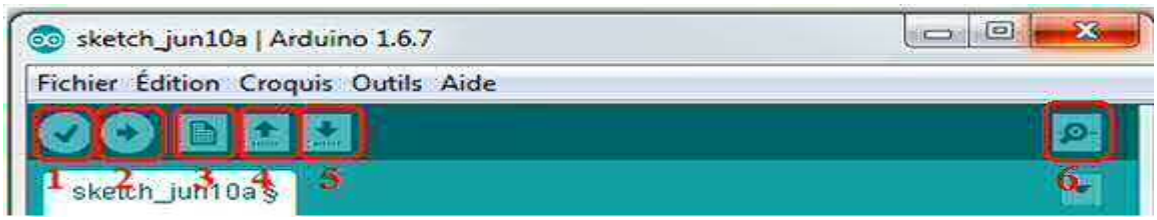


Figure (II-16) : Barre d'Action

1)  Bouton **Verify** (**Compiler**) : il permet de vérifier le programme pour trouver d'éventuelles erreurs. Cette procédure prend un certain temps d'exécution et lorsque elle est terminée, elle affiche un message de type « Binary sketch size : ... » Indiquant la taille du programme téléversé.

2)  Bouton **Upload** (**Téléverser**) : ce bouton permet de compiler et téléverser le programme sur la carte Arduino.

3)  Bouton **New** (**Nouveau**) : ce bouton permet d'ouvrir une nouvelle fenêtre de programmation.

4)  Bouton **Open** (**Ouvrir**) : il fait apparaître un menu qui permet d'ouvrir un programme qui figure dans le dossier de travail ou des exemples de programmes intégrés au logiciel.

5)  Bouton **Save** (**Sauvegarder**) : il permet de sauvegarder le programme.

6)  Bouton **Serial Monitor** (**Moniteur sériel**) : ce bouton fait apparaître le moniteur série.

2. **Fenêtre de Programmation :** est l'éditeur ou s'écrit le programme, chaque logiciel obéit à quelques notions pour pouvoir bien structurer le programme à fin de le compiler et éviter les erreurs de syntaxe et autres.
3. **Barre des erreurs :**
   La barre des erreurs affiche les erreurs faites au cours du programme, comme l'oubli d'un point-virgule, le manque d'une accolade ou toute autre erreur dans les instructions. Un exemple de ce genre s'affiche en cas d'erreurs :

Figure (II-17) : Message d'Erreur

II-2-3) Structure du programme :

Le programme comporte trois phases consécutives :

a) La Définition des constantes et des variables :

Cette partie est optionnelle, chaque entrée et sortie est définie et déclarée, en lui donnant un nom arbitraire et en lui affectant le numéro de l'entrée ou celui de la sortie voulue, sans oublier de préciser le type de la variable.

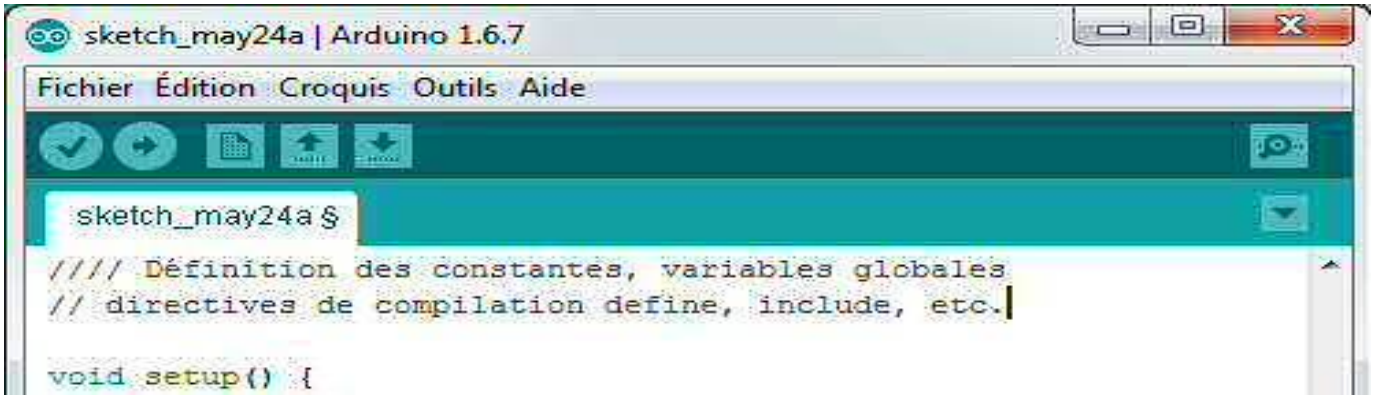


Figure (II-18) : Définition des constantes et des variables.

b) Configuration des entrées/sorties :

Les instructions viennent après le **void setup (),** après avoir ouvert une accolade, on peut manipuler les broches de la carte en les configurant comme étant des entrées ou des sorties, selon les besoins. Les entrées analogiques pour les capteurs par exemple, ne sont soumises à aucune configuration, car la carte possède 6 entrées analogiques qui ne font que cela.

```
void setup() {
  // initialisation des ressources de la carte,
  // configuration des entrées/sorties,
  // définition de la vitesse de fonctionnement du port série, etc.
  // setup() n'est exécuté qu'une seule fois.
}
```

Figure (II-19) : Configuration des Entrées/Sorties

c) Programmation des interactions et comportements :

Celles-ci viennent après le **void loop (),** c'est la partie principalement, ou on rédige les instructions et les opérations comme la lecture des données, les boucles, les affectations,...etc. Chacune d'elle doit obligatoirement finir par un point-virgule.

```
void loop() {
  // les instructions contenues ici sont exécutées indéfiniment en boucle
  // Seule une coupure de l'alimentation de la carte ou un appui sur le bouton Reset
  // permet de quitter le programme.
}
```

Figure (II-20) : Programmation des interactions et comportements

d) Les commentaires :

Comme chaque IDE, des commentaires peuvent être ajoutés au programme. Dans la configuration des entrées/sorties, les commentaires doivent être écrits après un slash ou une étoile ou les deux, tandis que sur une ligne de code, on les écrit après deux slash.

II-2-4) Le langage de programmation :

Comme nous l'avons mentionné, le langage de l'IDE Arduino est un mélange entre le C et le C++, il possède un jeu d'instruction très riche. Ces instructions décrivent :

- Les données, qui peuvent être numériques (*byte, int*, *word*,...), logiques (*booleen*, …), sous forme de tableaux (*array*), caractères ou chaines de caractères (*char, string*), ou constantes particulières (True/False, HIGH/LOW) …etc.
- Les fonctions arithmétiques et mathématiques : comme les fameuses quatre opérations arithmétiques (+, -, * et /) simples ou composées, les fonctions mathématiques (*abs, min, max,..*) et trigonométriques (*cos, sin, ...*).
- Les opérateurs logiques (*&&, ||, !,...*) et les opérateurs de comparaisons (=, <, >, ..).
- Les structures de contrôle comme les boucles (*for, while*,..), les prises de décision (*if-else,...*), les sauts (*breack, goto, continue,...*).
- Gestions du temps (*delay*) et des entrées /sorties numérique (*pinMode, digitalWrite/Read*) ou analogique (*analogRead/Write*).
- Fonctions diverses pour générer des nombres aléatoires (*random, randomSeed*), et pour manipuler des bits (*low/highByte, bitRead/Write/Set/Clear*,…), ainsi que pour gérer les interruptions (*attach/detach/noInterrupt*).
- Gestion du port série (*Serial.begin/.end/.available/.read/.print,...etc*).

II-2-5) Compilation et Téléversement :

Une fois le programme terminé et vérifié, on passe au test, en appuyant sur le bouton **Compiler,** une barre de progression s'affiche au-dessus de la barre des erreurs, s'il n y a pas d'erreur, on verra s'afficher le message « **Compilation terminée »,** suivi de la taille du programme. Dans le cas contraire, un message d'erreur s'affiche.

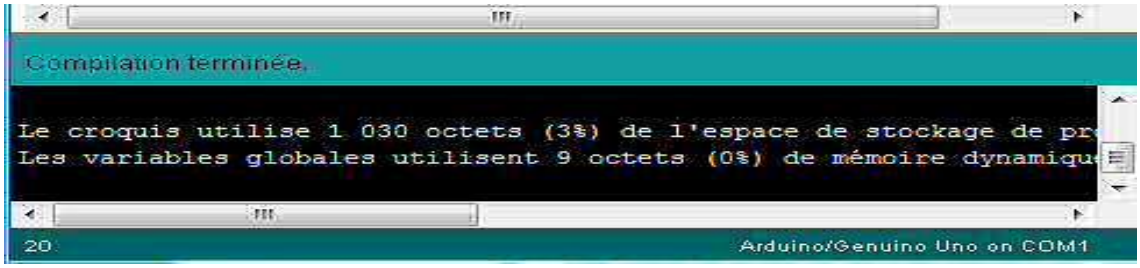


Figure (II-21) : Compilation

Une fois la compilation terminée, le programme sera sauvegardé. Mais avant de téléverser, il faut :

- → Réinitialiser la carte en appuyant sur le bouton de réinitialisation ;
- → La branchée sur l'ordinateur. Une fois branchée, on clique sur le bouton « **Outils »** ;
- → On sélectionne « **Type de carte »,** puis on coche la carte correspondante;
- → Toujours dans le menu « **Outils »,** on clique sur « **Serial port »,** pour sélectionner le type de la connexion; [4]

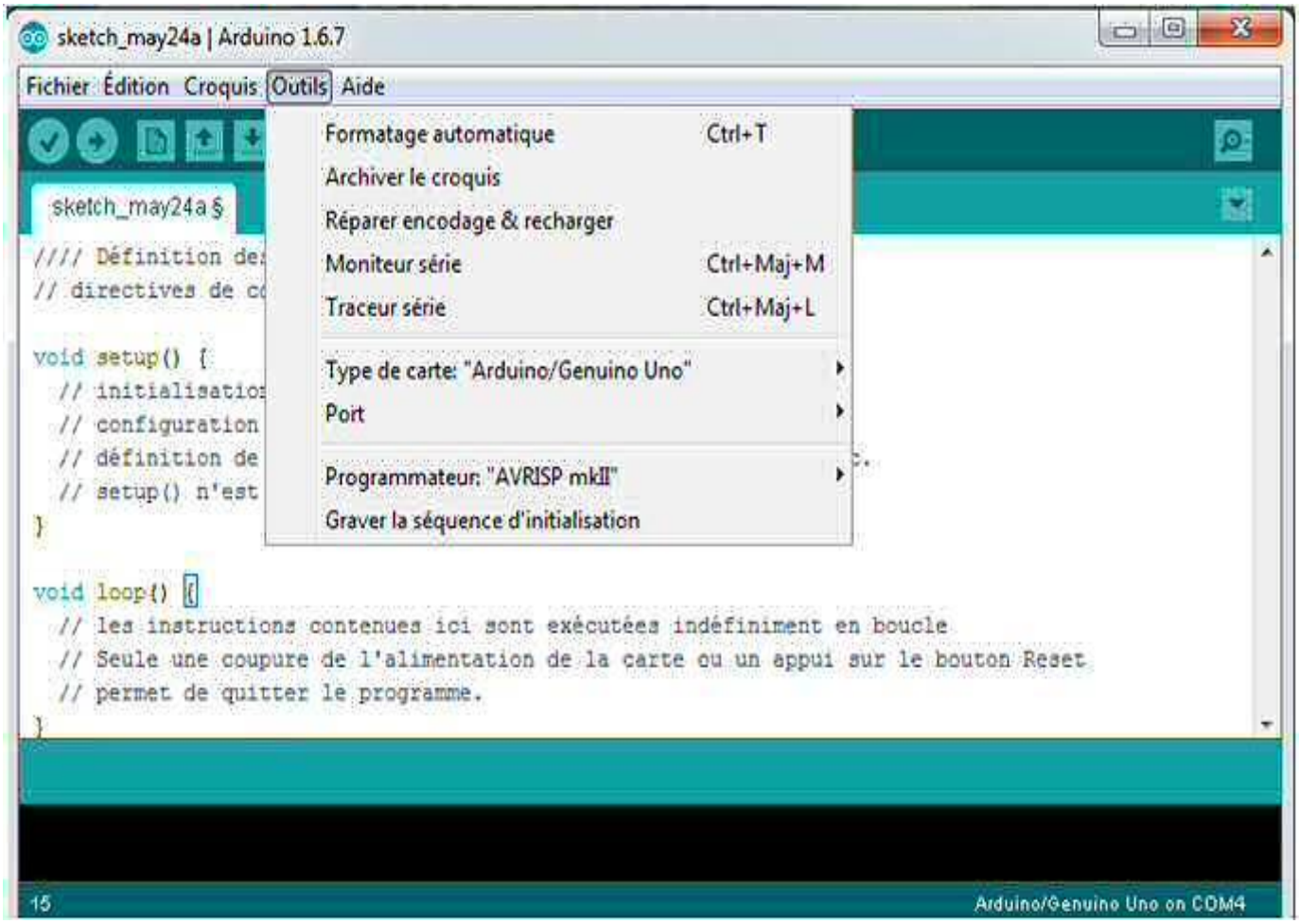


Figure (II-22) : Sélection de la carte et du port

- → On télécharge le programme en cliquant sur « **Téléverser »** plus haut sur la barre d'action, le message « **Compilation du croquis »** s'affiche au-dessus de la barre des erreurs, ce message signifie que le programme est en train d'être traduit en langage machine. Puis, on verra « **Téléversement… »,** qui veut dire que le programme se charge dans la carte.

La LED LX clignote dans le cas où la carte n'est pas défectueuse. Un exemple détaille la procédure dans le chapitre III.

❖ **<u>Remarque :</u>** pour connaitre le type de port, on doit se rendre dans le « **Panneau de configuration »,** on sélectionne « **Gestionnaire des périphérique »**, puis on regarde la ligne « **Port (COM et LPT),** et là on peut voir le nom de l'Arduino suivi du type de la connexion entre parenthèses, par exemple « Arduino Uno (COM3).

II-3) Utilisation de la carte [5]

II-3-1) Raccordement

Une fois la carte branchée et le programme téléversé, la carte est prête à être utilisé, maintenant il nous faut établir l'alimentation, comme nous l'avons cité ultérieurement, la carte Arduino peut être alimenté de deux façon, via une alimentation externe grâce au circuit d'alimentation, et via la connexion USB.

Le raccordement de la carte est comme la réalisation d'un circuit électrique, à l'aide de fils de connexion qu'on branche dans les périphériques de la carte, en prenant note que les microcontrôleurs n'aime pas délivrer du courant, ils préfèrent l'absorber, et en tenant compte aussi des spécifications des broches d'entrées/sorties citées plus haut, c'est-à-dire :

a) les entrées analogiques sont plutôt réservées aux capteurs, et
b) les broches numériques comme sortie pour les actionneurs, et les broches de l'alimentation pour les sources.

En principe, chacune des entrées/sorties de la carte ne peut pas délivrer plus de 20 mA. Cette quantité de courant est relativement faible mais permet, par exemple, de contrôler une LED ainsi que des actionneurs de faible puissance tel qu'un piézoélectrique ou encore un petit servomoteur.
En cas d'utilisation d'un élément qui requière plus d'énergie (>5V), il est nécessaire de confectionner une batterie qui relie cette élément à la carte, pour cela il faut réaliser deux circuits différents, un circuit de commande ou figure l'Arduino et un circuit de puissance qui active cet élément.

II-3-2) Circuit de commande

Comme son nom l'indique, c'est dans ce circuit que sont rassemblés tous les éléments de contrôle comme les boutons, les interfaces et le microcontrôleur. Il est alimenté en basse tension : moins de 50 V, souvent 12 V, ou avec la carte Arduino 5 V, on pourrait l'assimiler au système nerveux d'un organisme : c'est ici que se prennent les décisions mais peu d'énergie y circule.

La manière la plus simple de relayer les commandes émergeant de ce circuit pour les transmettre au circuit de puissance est d'utiliser des transistors ou encore des relais.

Lorsque les tensions d'alimentation des deux circuits sont plus importantes ou si l'on veut protéger la commande de retours accidentels de courant provenant de la puissance, des optocoupleurs[1] (plutôt que des transistors) assurent une isolation galvanique : l'information est transmise sous forme de lumière. Ainsi, les deux circuits sont complètement isolés électriquement.

II-3-3) Circuit de puissance

Ce circuit alimente les composants nécessitant beaucoup d'énergie (habituellement les moteurs et autres actionneurs). Sa tension d'alimentation dépend des composants en question.

En fonction de cette tension et des conditions d'utilisation, les précautions à prendre sont variables : dans tous les cas, une protection contre les courts-circuits est conseillée : fusible ou disjoncteur pour éviter de détruire les composants ; au-dessus de **50 V,** la tension peut menacer directement les humains : protéger les pièces nues sous tension **;** Les **230 V** nécessitent un interrupteur différentiel qui protège les humains des contacts accidentels (en général tout local est doté de ce dispositif).

Il est important de noter que ces deux circuits sont montés distinctement et sont isolés l'un de l'autre. Toutefois, il est primordial de lier leur masse (« GND ») afin qu'ils partagent le même potentiel de référence.

[1] *Optocoupleur :* ou photocouleur, est un semi-conducteur capable de transmettre un signal d'un circuit électrique à un autre, sans qu'il y ait de contact galvanique entre eux.

II-4) Simulation virtuelle de la carte Arduino (Proteus)

Pour bien assimilé l'idée de l'utilisation de la carte, nous allons faire une petite simulation sur Proteus, mais avant, petit laïus sur Proteus :

II-4-1) Présentation du logiciel Proteus (ISIS et ARES)

Proteus est une suite logicielle permettant la pour Conception électronique assistée par ordinateur éditée par la société **Labcenter Electronics** [15]. Proteus est composé de deux logiciels principaux : **ISIS**, permettant entre autres la création de schémas et la simulation électrique, et **ARES**, dédié à la création de circuits imprimés.

- **ISIS**

Le logiciel ISIS de Proteus est principalement connu pour éditer des schémas électriques. Par ailleurs, le logiciel permet également de simuler ces schémas, ce qui permet de déceler certaines erreurs dès l'étape de conception. Indirectement, les circuits électriques conçus grâce à ce logiciel peuvent être utilisé dans des documentations car le logiciel permet de contrôler la majorité de l'aspect graphique des circuits.

- **ARES**

Le logiciel ARES est un outil d'édition et de routage qui complète parfaitement ISIS. Un schéma électrique réalisé sur ISIS peut alors être importé facilement sur ARES pour réaliser le PCB de la carte électronique. Bien que l'édition d'un circuit imprimé soit plus efficiente lorsqu'elle est réalisée manuellement, ce logiciel permet de placer automatiquement les composants et de réaliser le routage automatiquement. [15]

Sur Proteus, nous avons réaliser un petit circuit qui illustre l'utilisation de la carte, la manière de faire le branchement, le téléversement, et enfin la simulation. Le circuit est composé de la carte Arduino, une LED munie d'une résistance de 1KΩ branchée entre les pins 13 (entrée 19 de l'ATMega328) et GND (broches 8 et22 de l'ATMega328) de la carte.

Voici donc à quoi ressemble le circuit sur Proteus ISIS :

II-4-2) Schéma de circuit pour l'exemple Blink :

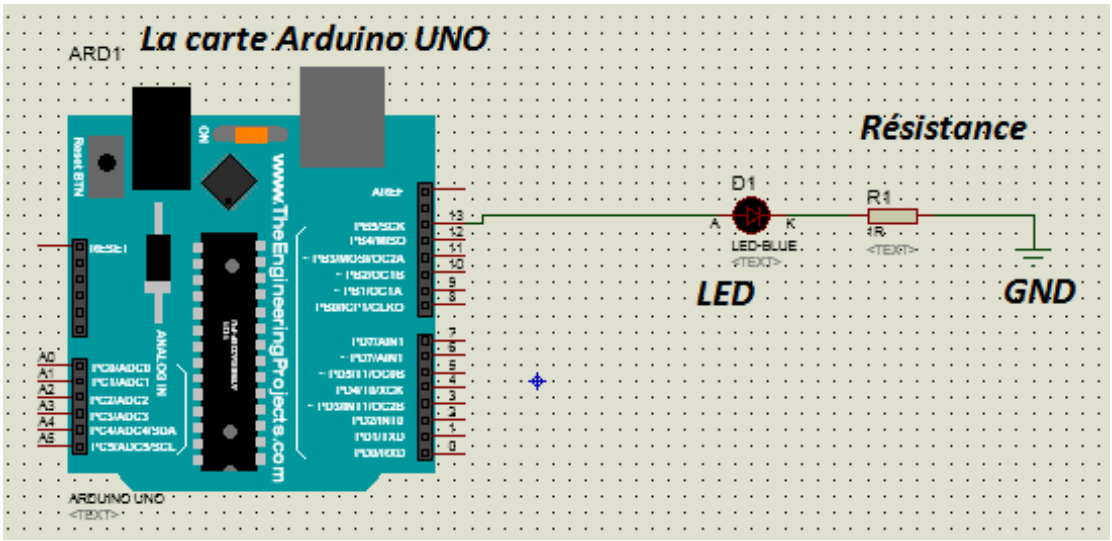


Figure (II-23) : Schéma du Circuit

II-4-3) compilation et téléversement du programme par le logiciel Arduino :

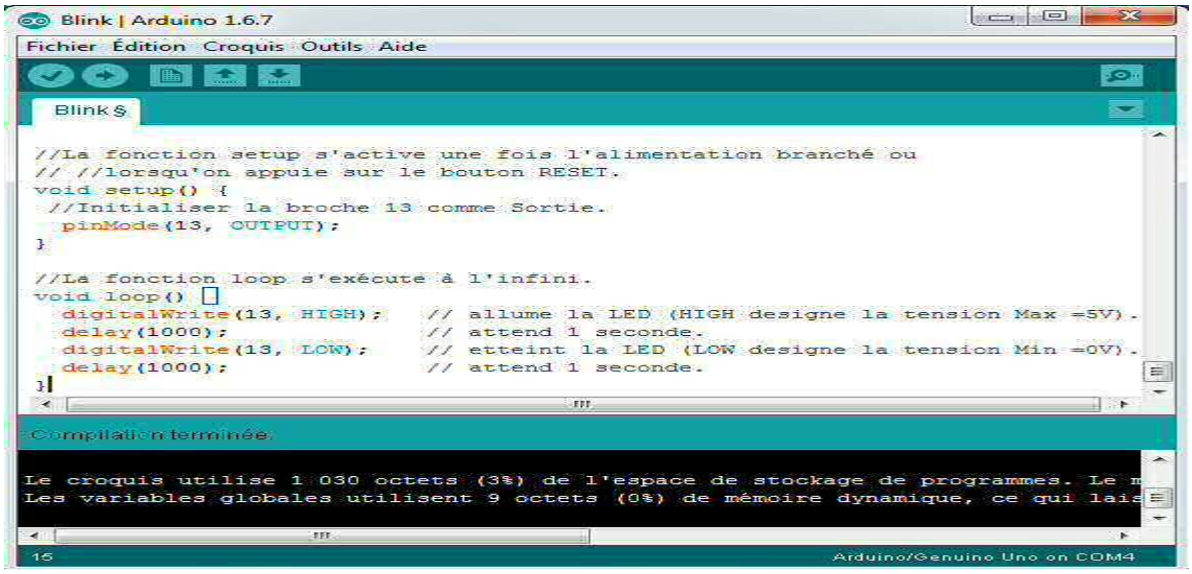


Figure (II-24) : Compilation sur la carte

Ce programme consiste à faire clignoter une LED avec un intervalle d'une seconde, ceci en :

- → Configurant le pin 13 comme sortie en utilisant l'instruction *pinMode,* dans le *void setup.*
- → Passer à l'état haut, puis à l'état bas avec l'instruction *digitalWrite,* dans le *void loop* pour allumer et étteindre la LED.
- → Attendre une seconde entre chaque état avec l'instruction *delay.*

II-4-4) Simulation :

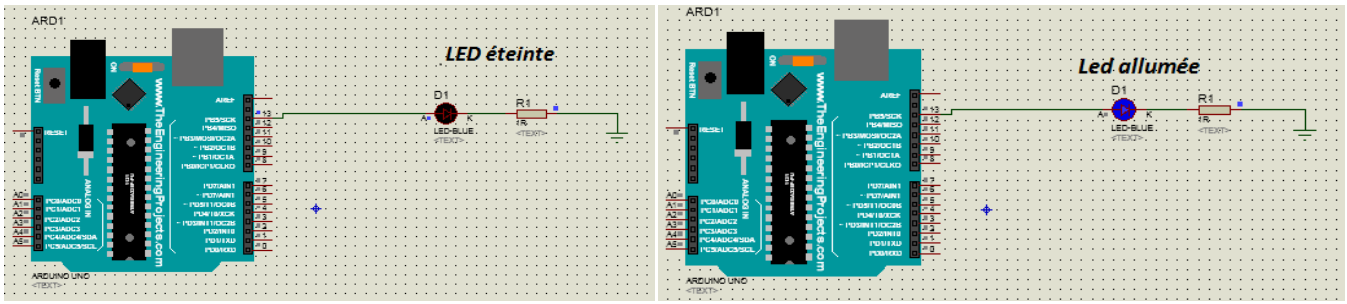


Figure (II-25) : Simulation

Cette simulation sur Proteus nous montre comment utiliser la carte Arduino, et comment se fait le raccordement avec les circuits externes.

Conclusion :

Ce chapitre nous montre de quoi sont composés les deux piliers de l'univers Arduino, Le Hardware et le Software. En gros, le Hardware est composé d'un ensemble de compartiments essentiels à son fonctionnement, un bloque d'alimentation, un noyau basé sur la technologie *Atmel*, des périphériques qui font le relais entre l'Arduino et les cartes interfaces, et une interface de communication pour la connexion avec la partie Software, cette dernière fonctionne sous Windows, Linux et Mac, dont le langage de programmation est un mélange entre les langages C et C++, comportant un menu, un éditeur, une barre de d'action et une barre d'erreurs, Ce langage est spécialement dédié aux cartes Arduino, et il possède des jeux d'instruction très faciles à maitriser, un exemple virtuel en fait état.

Chapitre III :
Réalisation et Test

Introduction

Trois compartiments de la carte Arduino assurent son fonctionnement, on distingue, un circuit d'alimentation qui est un assemblage de régulateur, de transistor et d'ampli-op, dont le rôle est de régulariser et sélectionner l'alimentation, puis une interface de communication série USB, qui assure la connexion avec le PC, et un noyau à base d'un microcontôleur ATMega328P, ou s'effectue le traitement des programmes.

III-1) Réalisation sur BreadBoard

III-1-1) Réalisation du circuit d'alimentation :

III-1-1-1) Circuit du régulateur de tension L7805CV :

Le L7805CV est un régulateur qui fournit une tension de 5V à partir d'une certaine tension d'alimentation.

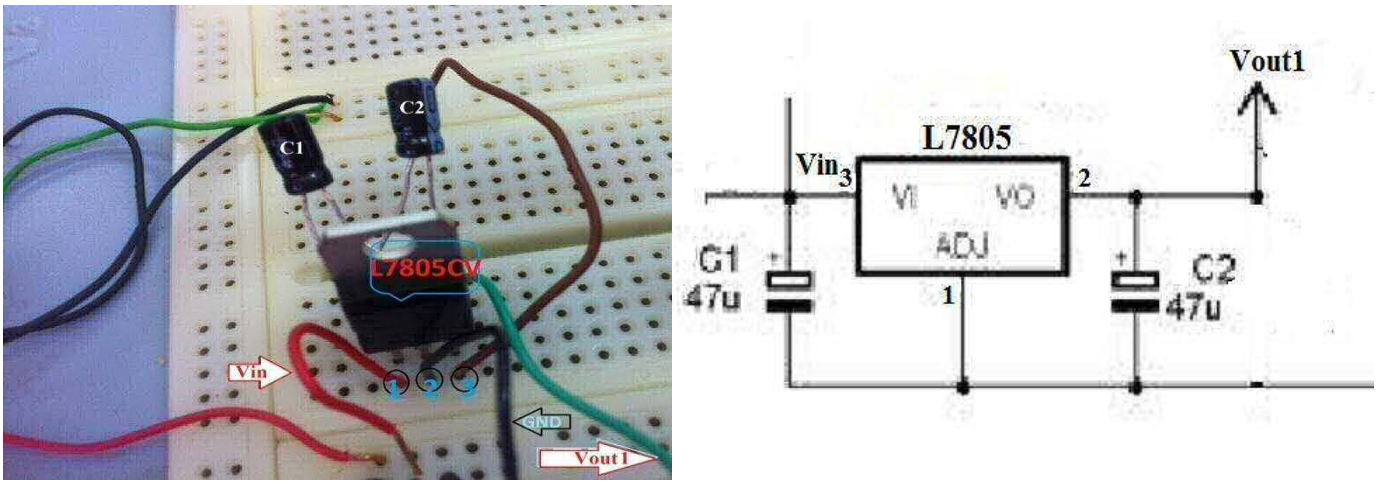


Figure (III-1) : Circuit du Régulateur L7805CV

Le régulateur L7805 possède trois électrodes : l'entrée Vin (3), la sortie Vout (2) ainsi que la masse GND (1). L'électrode Vin est branchée à la source, l'électrode GND est reliée à la masse et l'électrode Vout est la sortie.

On a alimenté ce circuit avec des tensions différentes, comprises entre 5 et 15 Volt, la tension maximum supportée par ce régulateur est de 32V. Les résultats sont relevés dans le tableau ci-dessous :

Vin (V)	5	6.5	7	10	15
Vout 1(V)	3.6	4.54	4.96	4.96	4.96

Les résultats indiquent que le régulateur délivre 5V avec une tension d'alimentation supérieure ou égale à 6.6V.

- Cette tension sert à :

→ alimenter la carte Arduino qui fonctionne sous 5V avec une tension externe comprise entre 7et 12V.

→ alimenter le régulateur AIC1084-33CM, utilisé dans le cas de l'emploi du microcontrôleur ATMega16U2.

→ alimenter la broche 5V de la carte Arduino pour brancher d'éventuels composants ou circuits externes.

III-1-1-2) Circuit du régulateur de tension AIC1084-33CM

Le régulateur AIC1084-33CM délivre une tension de 3.3V, à partir d'une certaine tension d'alimentation.

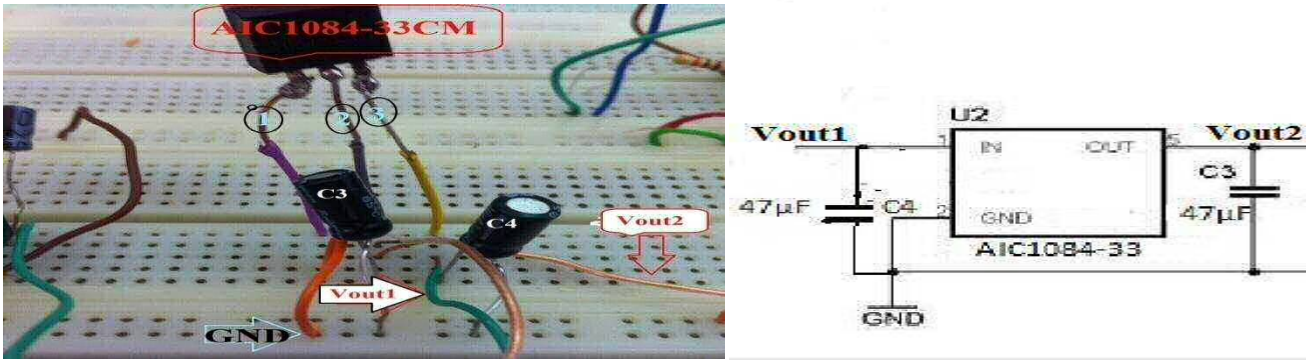


Figure (III-2) : Circuit et Schéma du Régulateur AIC1084-33CM (3.3V)

Comme le régulateur précédent, l'AIM1084-33MC possède aussi trois électrodes, Vin (3) alimenté par Vout1, Vout (2) et GND (1). Mais, nous avons modifié sa source en la rendant variable, c.-à-d. Vin est branché à un générateur de tension, en variant les valeurs des tensions d'entrées, nous avons mesuré les valeurs de sa sortie :

Vin(V)	3	4	4.5	6
Vout2 (V)	2.015	3.301	3.304	3.304

Les résultats nous indiquent que le régulateur délivre une tension de 3.3V après avoir reçu des tensions supérieures ou égale à 4.5 V, ce qui signifie qu'avec la tension Vout1, l'AIM1084-33MC délivre une tension de 3.3Vqui sert à :

- Alimenter la broche 2 de l'Amplificateur Opérationnel LM385.
- Alimenter la broche 3V3 de la carte.

❖ Dans le cas de l'utilisation de la puce FTDI, le régulateur 3.3V devient caduque, car la puce FT232RL fournie directement la tension de 3.3V, étant donner quelle integre un régulateur interne de 3.3V.

III-1-1-3) Circuit de l'amplificateur Opérationnel LM358

L'amplificateur Opérationnel LM358 agit comme un comparateur, il possède 2 broches d'alimentation et 4 broches d'entrées qui reçoivent des tensions, s'il detecte une différence de potentiels positive entres deux tensions comparées, il délivre une tension de sortie d'une valeur de 3.9V.

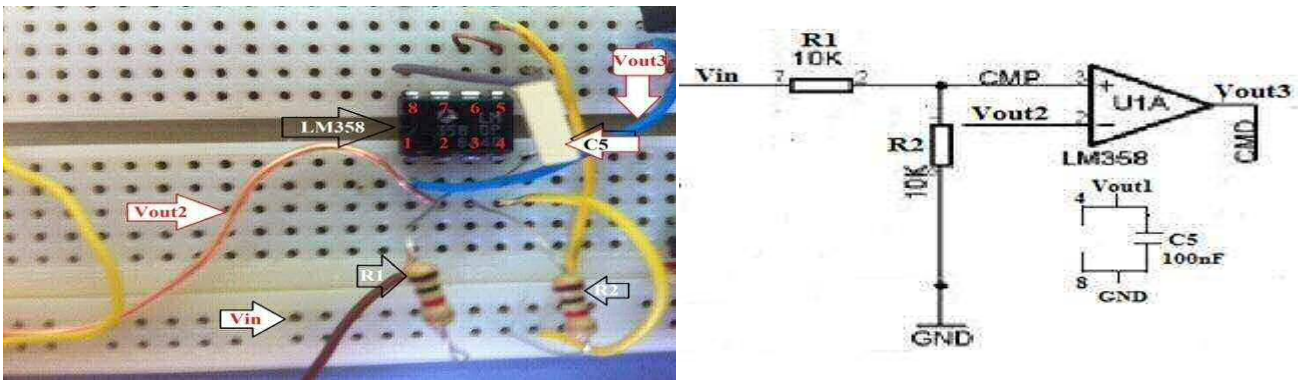


Figure (III-3) : Circuit et Schéma de l'Amplificateur Opérationnel LM358

- La tension Vin, passe par un circuit diviseur de tension formé par deux résistances de 10KΩ montées en série, puis alimente la broche 3 de l'ampli-op.
- La tension Vout2 de 3.3V, alimente la broche 2 de l'ampli-op.
- Sa propre alimentation se fait à travers des broches 4 et 8.
- La broche 1 est considérée comme sa sortie, cette dernière est branchée au transistor IRF9640.

On variant la tension Vin, on a relevé les tensions de sortie de l'ampli-op, les résultats sont portés dans le tableau suivant :

Vin	3	4	6	6.6	7	8
Vout3	0	0	0	3.9	3.9	3.9

Les résultats montrent que l'ampli-op délivre 3.9V vers le transistor à partir d'une tension d'alimentation de 6.6V, cette tension sera réduite à 3.3V par le circuit diviseur de tension. Dés que l'ampli-op detecte un dépassement, il envoie les 3.9V vers le transistor.

❖ L'utilisation del'interface FTDI fait que la broche 2 de l'ampli-op soit alimentée par la puce FT232RL.

III-1-1-4) Circuit du transistor à effet de champ IRF9640

Le transistor à effet de champ sert d'interrupteur, s'il reçoit une tension dans sa grille (G), il s'ouvre, dans le cas contrère il reste fermé.

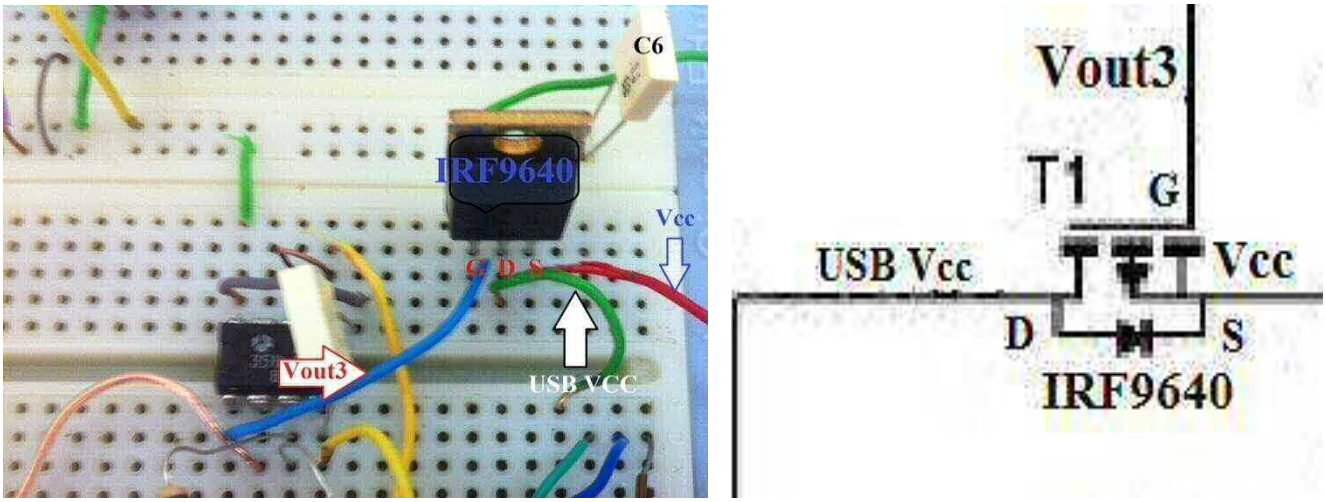


Figure (III-4) : Circuit et Schéma du Transistor à effet de champ IRF9640

Le transistor à effet de champ de type P, est un composant électronique doté de trois électrodes, la grille (G), la source (S) et le drain (D), son rôle consiste à faire passer un courant entre le drain D et la source S, le courant passe à travers un semi-conducteur de type P. Le contrôle de ce courant se fait au travers de la tension appliquée sur la grille G.

→ Lorsqu'on branche la carte à l'ordinateur, la tension passe du Drain vers la Source, à condition que la tension de la Grille soit nulle, dans ce cas l'alimentation se fait à travers le port USB.

→ Mais si la tension de la Grille est non-nulle, et qu'elle atteint une valeur de 3.9V, le transistor se bloque, ainsi que l'ampli-op, l'alimentation de la carte se fait donc à travers le régulateur 5V, le L7805.

❖ Le même raisonnement s'applique pour cas de l'utilisation de la puce FTDI. En d'autres termes, c'est le FTDI qui alimente la broche 2 de l'ampli-op à la place du régulateur AIC1084-33CM, et a sortie de l'ampli-op qui alimente la grille G du transistor.

III-1-2) Réalisation du circuit de l'interface série (USB)

La carte Arduino Uno Rev3 possède une interface série USB munie du microcontrôleur ATMega16U2, la version Rev2 utilisait l'ATMega8U2, mais à défaut de moyen, nous allons la réaliser avec un une puce FTDI, la FT232RL, on nous basant sur la Diecimila et la première version de la UNO. La puce FT232RL remplit le même rôle que l'ATMega16U2, en plus accessible, mais moins performante.

Le circuit de l'interface série USB est identique à celui de l'adaptateur série USB FTDI, nous l'avons réalisé directement sur le circuit imprimé de la carte, en utilisant les pièces de l'adaptateur, après l'avoir testé sur la maquette :

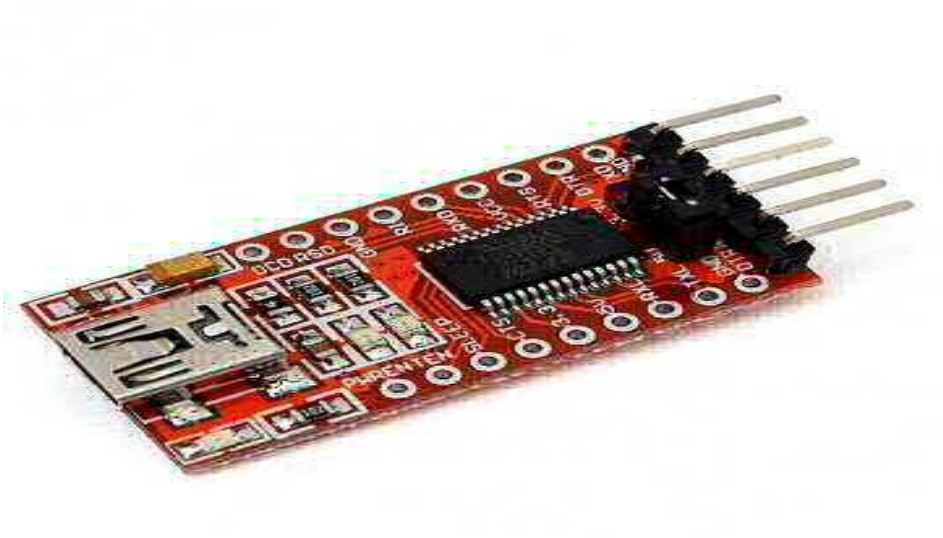


Figure (III-5) : Adaptateur USB FTDI.

L'adaptateur possède exactement les mêmes sorties que la puce FT232RL, présentes sur trois cotés, 18 en formes de trous sur les cotés latéraux, et 6 en forme de broches. Sur la maquette, nous avons utilisé les 6 broches de sortie suivantes :

→ VCC et GND, pour la source et la masse,
→ TX et RX pour la communication avec le microcontrôleur,
→ DTR et RTS connectés avec le pin RESET du µc.

❖ **Remarque** : l'utilisation de l'adaptateur, requière un pilote, pour l'installer, il faut suivre les mêmes instructions que celle du pilote de l'IDE Arduino (pages21, 22).

L'interface FTDI est simple à réaliser ; Du coté du connecteur USB,

→ deux condensateurs de 100Nf en série relient le pin 17(3.3V) de la puce à la broche 1 du connecteur
→ les pins 4, 19 et 20 sont reliés à la source de 5V.
→ les pins D+et D- sont reliés respectivement aux broches 2 et 3 du connecteur USB.

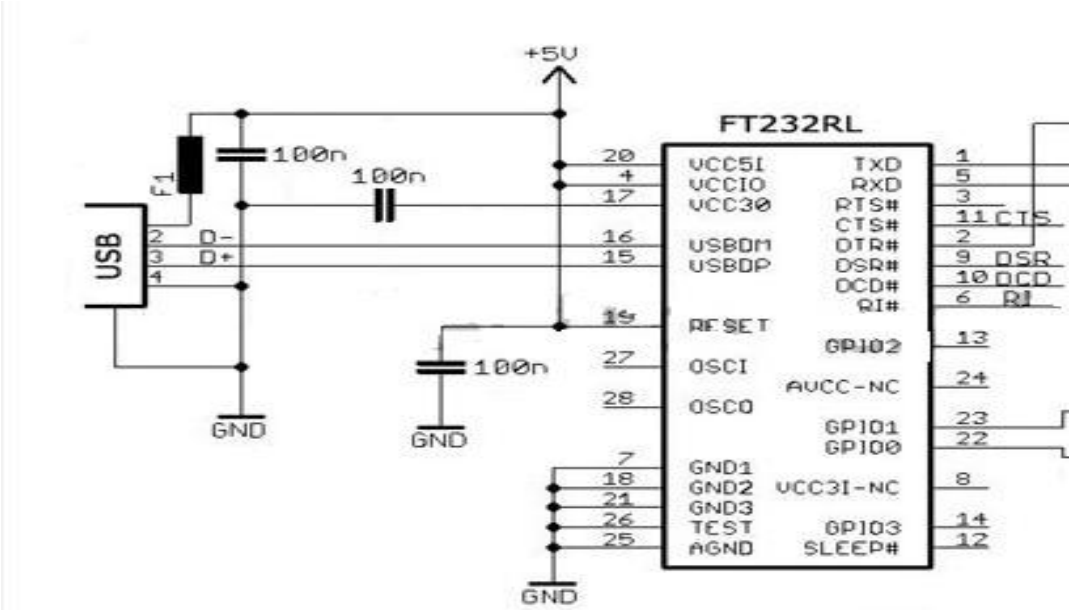


Figure (III-6) : Connexion USB/FT232RL

Du coté du pont entre les PICs :

- → le pin DTR est raccordé au pin Reset de l'ATMega328P avec un condensateur 100N.
- → les pins RX et TX des deux PICs sont reliés entre eux à travers deux résistances de 1KΩ.
- → les Deux LEDs TX et RX sont branchées à la source de 5V à travers deux résistances de 1KΩ, pour la signalisation du transfert des données.

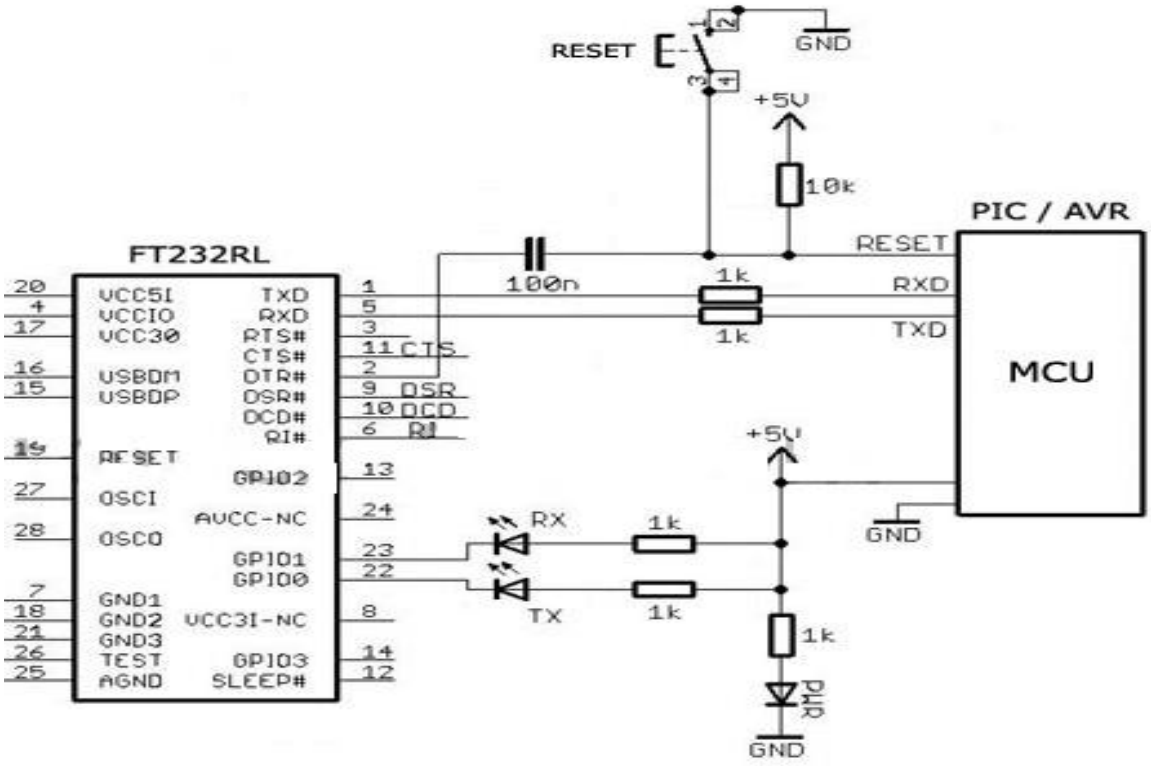


Figure (III-7) : Connexion FT232RL/ATmeg328

La puce FT232RL possède un régulateur de tension interne qui fournie une tension de 3.3V, qui est acheminée vers l'ampli-op et la broche 3V3 de la carte. D'où inutilisation du régulateur AIC1088-33CM.

III-1-2-1) Diagramme de la puce FT232RL

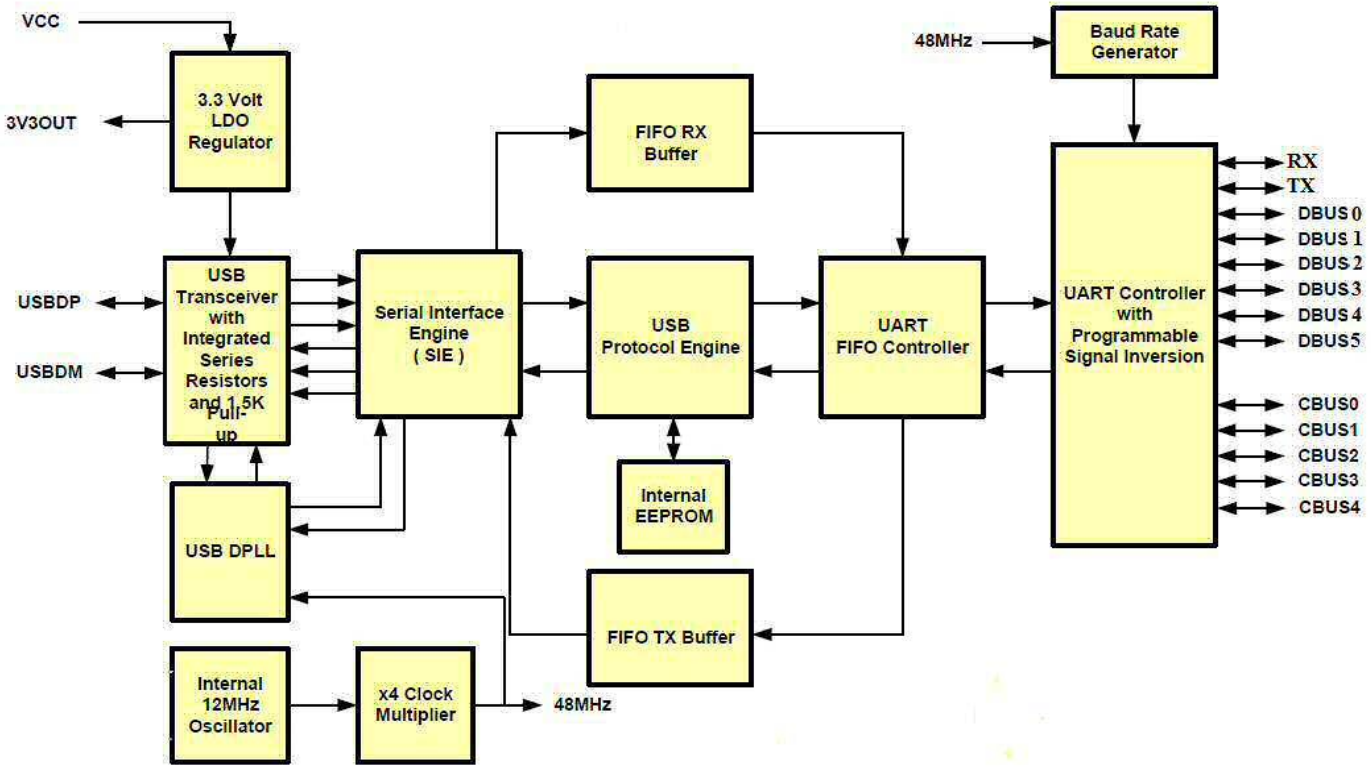


Figure (III-8) : Diagramme de la puce FT232RL

III-1-2-2) Communication des données :

Cette interface sert de communicateur entre la carte Arduino et l'ordinateur, il s'agit d'un adaptateur série USB, le principe est que :

- → les broches USBDP et USBDM du coté du connecteur, reçoivent et émettent les données en format binaire,
- → Ces données passent par le registre SIE, il s'agit d'un convertisseur série/parallèle.
- → Les données converties, passent par les FIFO[1] RX/TX Buffer, qui stock momentanément ces données, puis ;
- → Ces données sont acheminées vers le registre UART, muni de la technologie TTL[2], il convertit ces données en tension, puis,
- → Ces données sont envoyées vers le registre USART de l'ATMega328P à travers les broches TX/RX, le registre USART les reconvertit en format binaire, pour l'exécution.

[1] *FIFO :* First In First Out.
[2] *TTL* est acronyme pour Transistor Transistor Logic

III-1-2-2) Principe de la communication de données

La communication des données entre ces deux pics se fait à travers ces protagonistes :

a) **La technologie TTL** a été inventée en 1960. Cette famille est réalisée avec des transistors bipolaires. (De nos jours, la technologie TTL tend à être remplacée par la technologie CMOS). Les avantages de cette famille :
- Les entrées laissées en 'l'air' ont un état logique à 1 par défaut.
- Une bonne immunité au bruit.
- Un temps de propagation faible.

  Les inconvénients de cette famille :
- L'alimentation doit être précise à 5V +/- 5 % sinon on risque de détruire le circuit.
- Du fait qu'elle est réalisée avec des transistors bipolaires elle consomme pas mal de courant comparer à la famille CMOS. (Car les transistors bipolaires sont commandés en courant).

b) **USART** : est un acronyme pour Universal synchronous Asynchronous Receiver Transmitter, l'USART est un protocole utilisé pour la communication de données, comme son prédécesseur UART, Il utilise une trame pour transmettre et une autre pour recevoir des données. Le plus souvent, les données de 8 bits sont transférées comme suit: 1 bit de départ (bas niveau), 8 bits de données et 1 bit d'arrêt (de haut niveau). Le bit de démarrage de niveau bas et bit d'arrêt de haut niveau signifie qu'il ya toujours une transition haut-bas pour commencer la communication. Voilà ce que décrit UART. Pas de niveau de tension, de sorte qu'on peut l'avoir à 3,3 V ou 5 V, selon le microcontrôleur utilisé.

La dépendance temporelle est l'un des grands inconvénients de l'UART, c'est pour ça que l'USART est muni d'un protocole synchrone. En synchrone, il y a non seulement des données, mais aussi une horloge transmise. À chaque bit une impulsion d'horloge indique au récepteur qu'il doit verrouiller ce bit. Les protocoles synchrones ont besoin d'une bande passante plus élevée ou un fil supplémentaire pour l'horloge.

c) **Les broches TX, RX, D+ et D-,** ont les fonctionnent comme suit :

TXD1 : quand le registre USART1 est en mode émetteur, cette broche est configurée comme une sortie indépendamment du registre DDRD3.

RXD1 : quand le registre USART1 est en mode récepteur, cette broche est configurée comme entrée indépendamment du registre DDRD2, Lorsque le registre USART force cette broche à être une entrée, le pull-up peut encore être commandé par le bit PORTD2.

D+, D- sont des broches qui de la communication vers l'extérieur, les données véhiculées sont sons forme TTL.

III-1-3) Circuit du Cœur de la carte (Processeur) :

Le cœur de la carte Arduino est composé du microcontrôleur ATMega328P, muni d'un circuit oscillatoire formé par un Quartz de 16MHz et de deux condensateurs de 22pF, pour lui fournir la référence temporelle, il est raccordé directement au FTDI à travers les pins TX, RX, et Reset, déjà cité plus haut (page 36, 37).

III-1-4) Maquette de la carte Arduino Uno sur BreadBoord

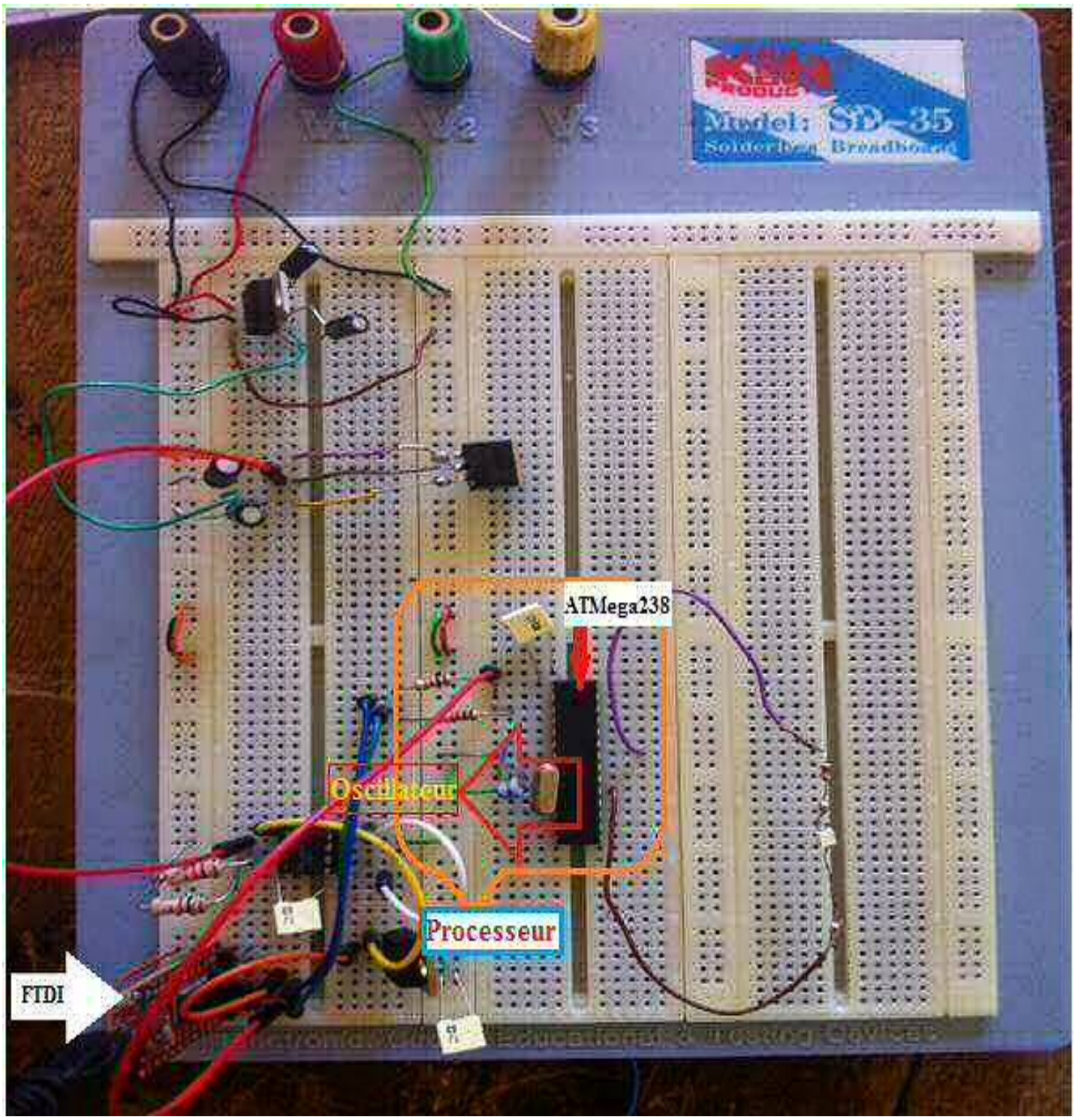


Figure (III-9) : Maquette de la carte Arduino Sur BreadBord

III-2) Test de la maquette :

Après avoir ouvert l'IDE Arduino, on a cliqué sur **Fichier**, puis sur **Exemples**, puis **Basics**, et on a choisi le programme **Blink**, un des exemples simple de la bibliothèque de l'IDE Arduino :

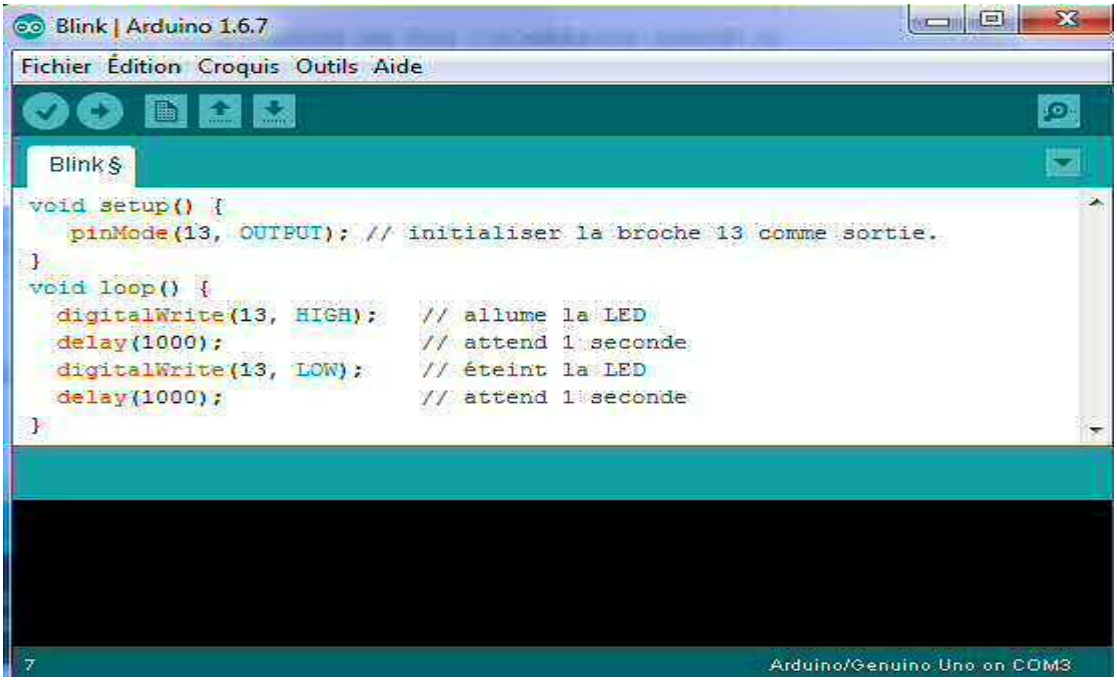


Figure (III-10) : Programme Blink sur l'IDE Arduino

Le programme est plus tôt simple, comme c'est mentionné dans l'éditeur, on utilise le pin 13 comme sortie, puis on l'allume la LED puis on l'éteint avec un intervalle d'une seconde à l'infini.

On le compile pour vérifier qu'il n'y a pas d'erreurs, puis on le téléverse et pour ça, on doit d'abord :

→ Branche la carte dans le PC via l'adaptateur série USB FTDI, puis
→ On clique sur l'icône **outils**, puis on choisi le **type de carte** ainsi que le **port**, comme ceci :

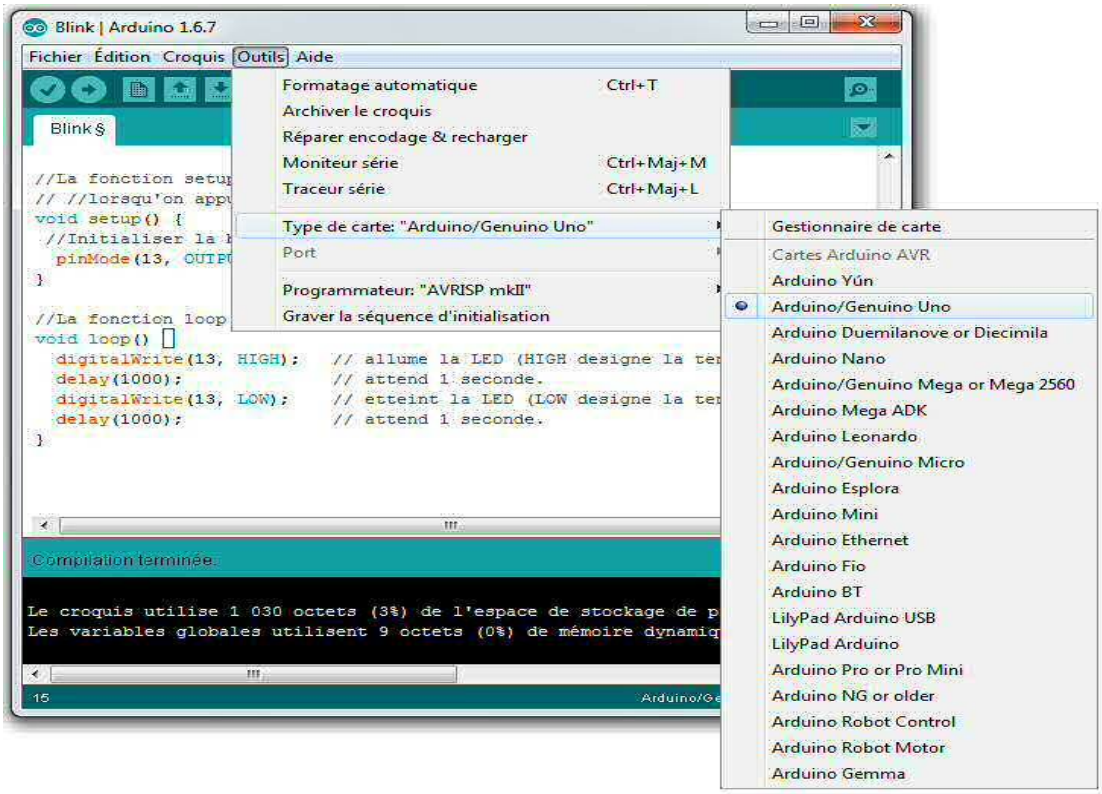


Figure (III-11) : Sélection de la carte Arduino

Et :

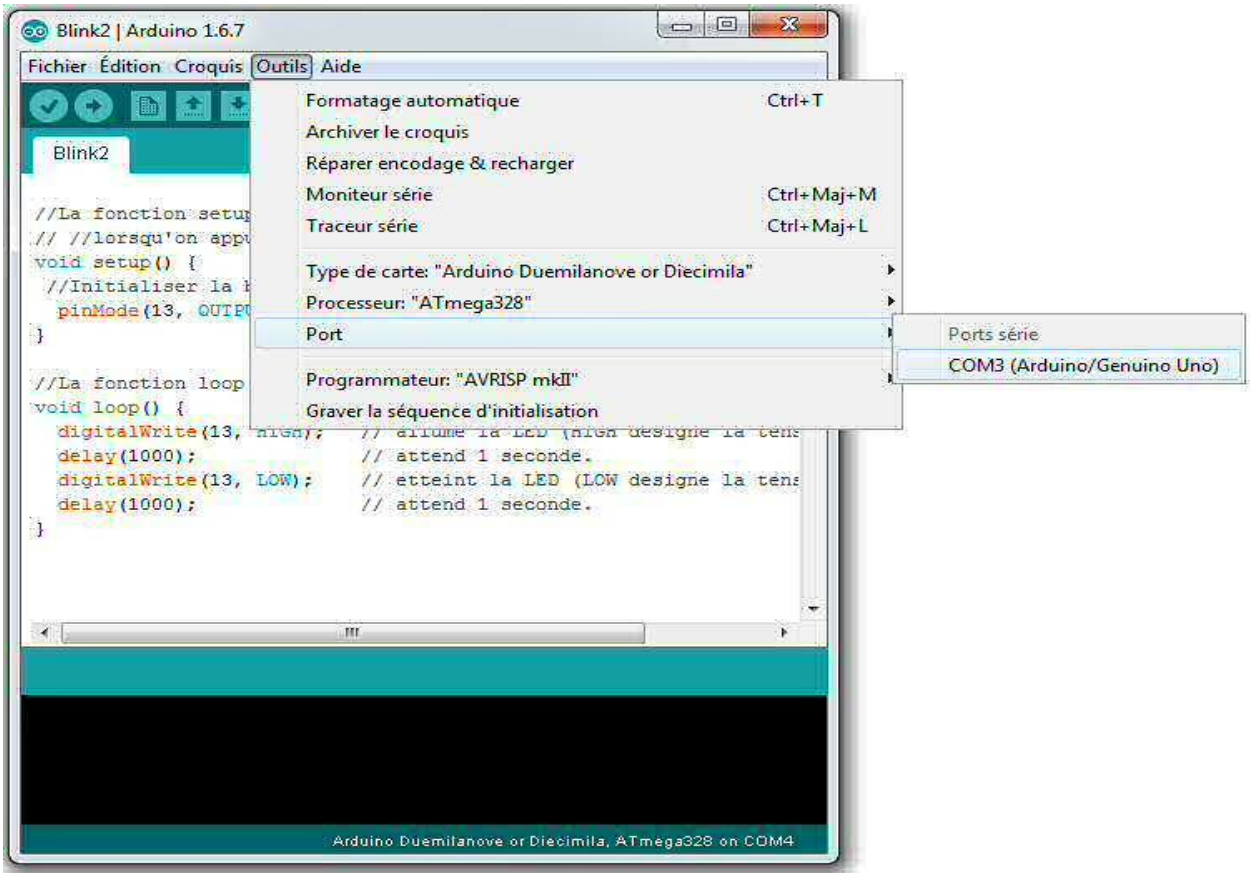


Figure (III-12) : Sélection du Port

Quelques secondes plus tard, les LEDs TX et RX du FTDI s'allument brièvement, puis la LED s'est mise à clignoter avec une période d'une seconde, comme c'était décrit dans le programme Blink.

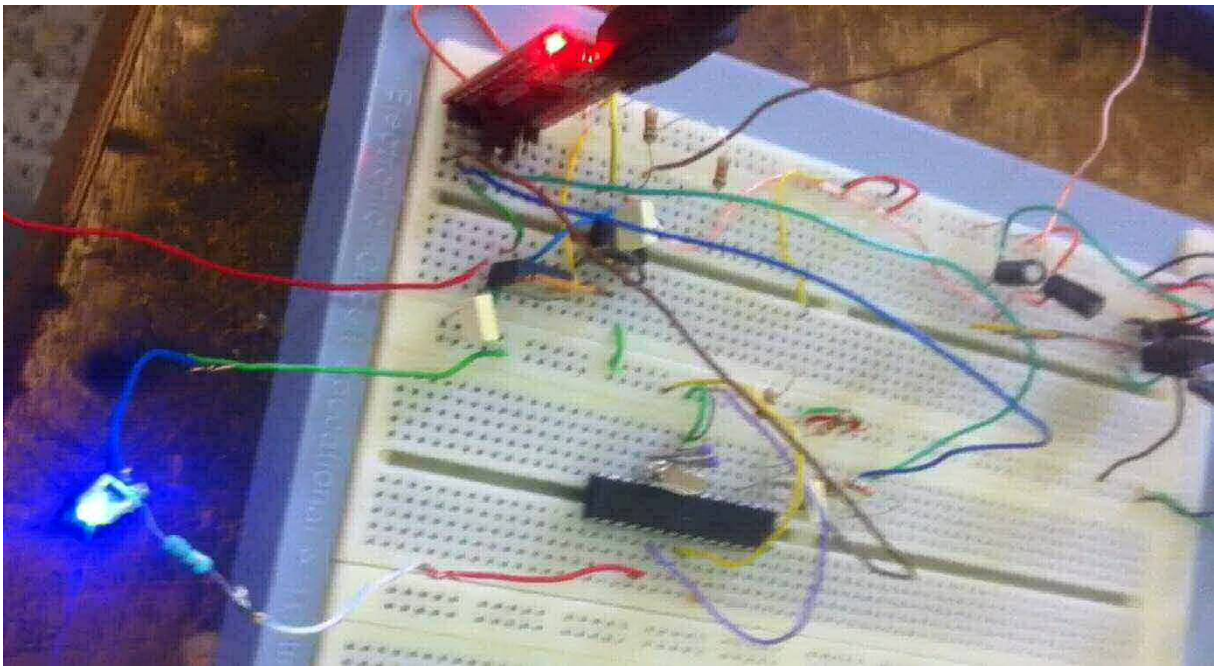


Figure (III-13) : Test de maquette avec l'exemple Blink

III-3) Réalisation de la carte Arduino

III-3-1) Schéma de la carte Sur Proteus ISIS :

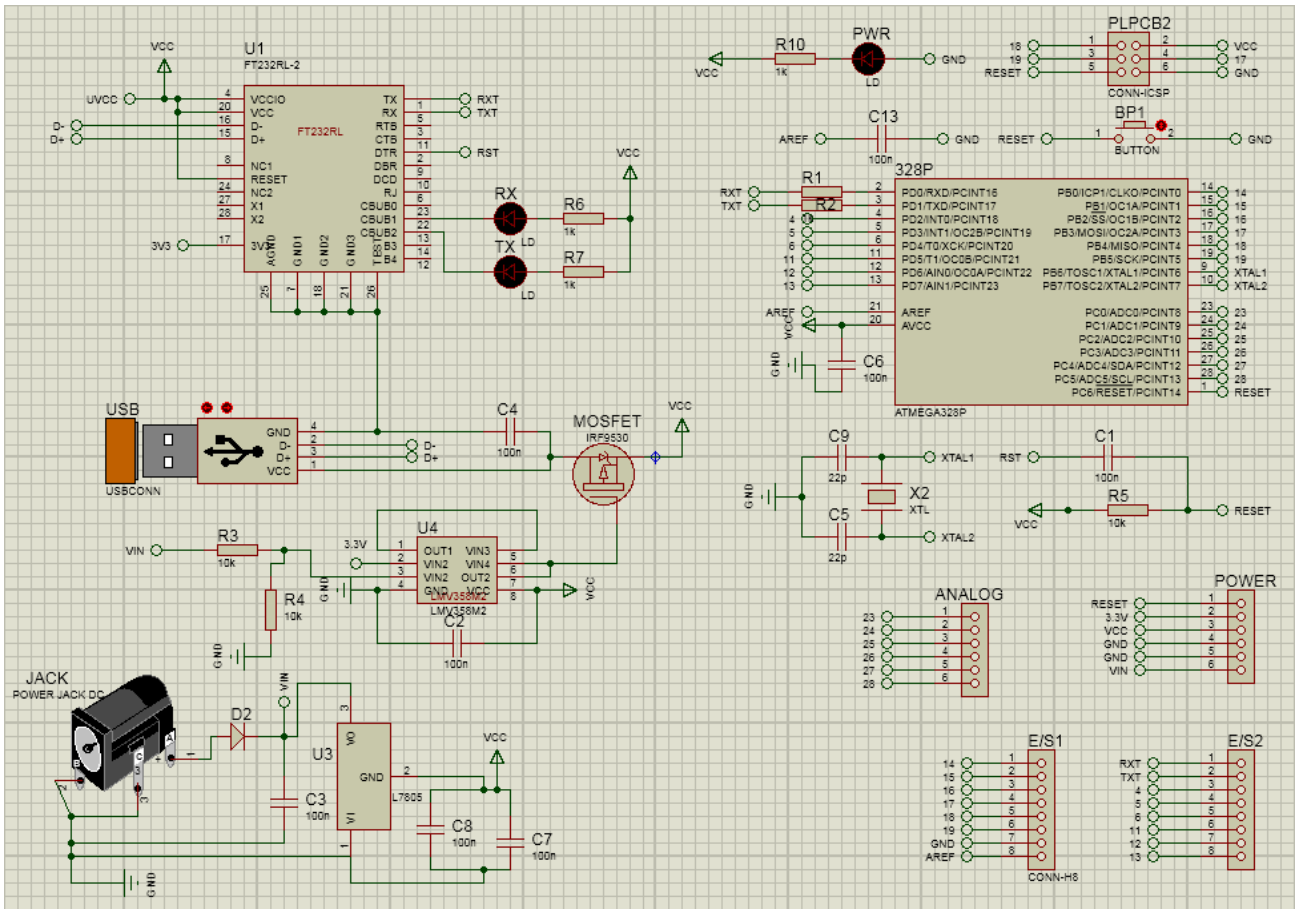


Figure (III-14) : Schéma Electrique de la Carte Arduino Sur ISIS

III-3-2) Sur Proteus ARES :

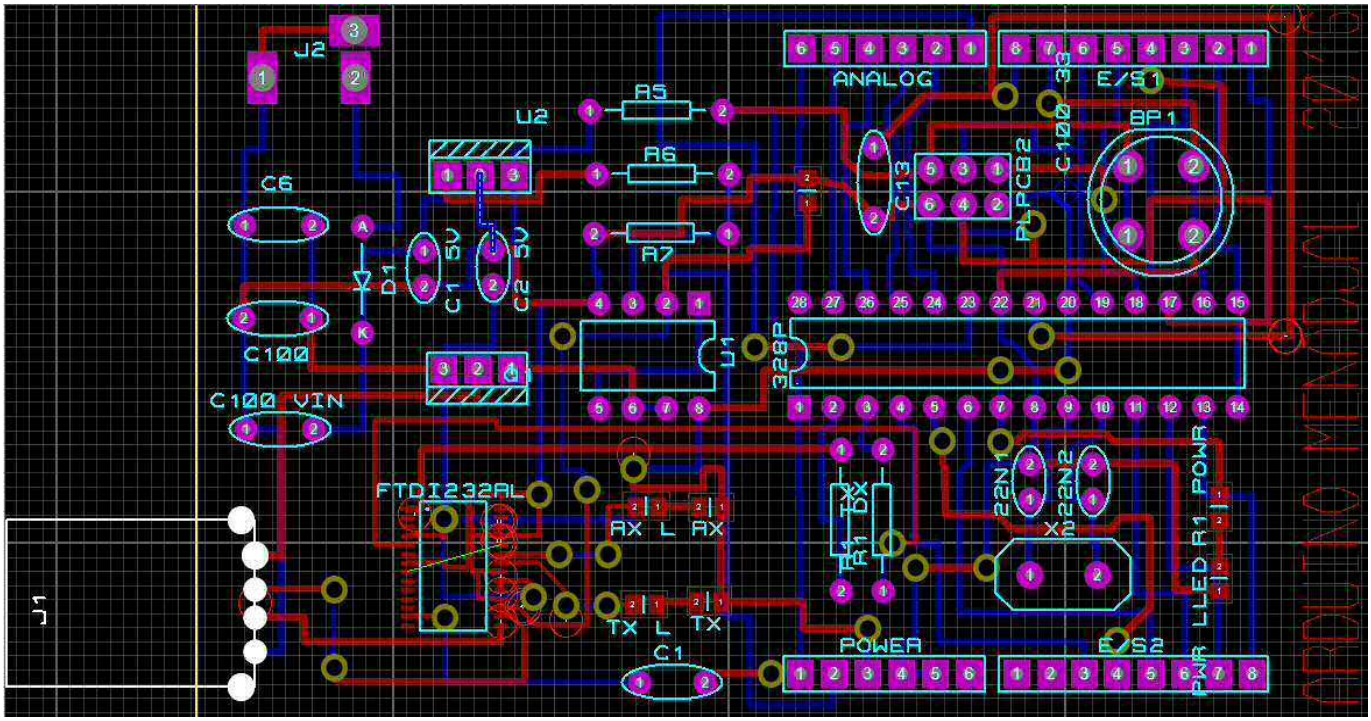


Figure (III-15) : Circuit de la Carte Arduino sur ARES

III-3-3) Vue 3D de la carte Arduino

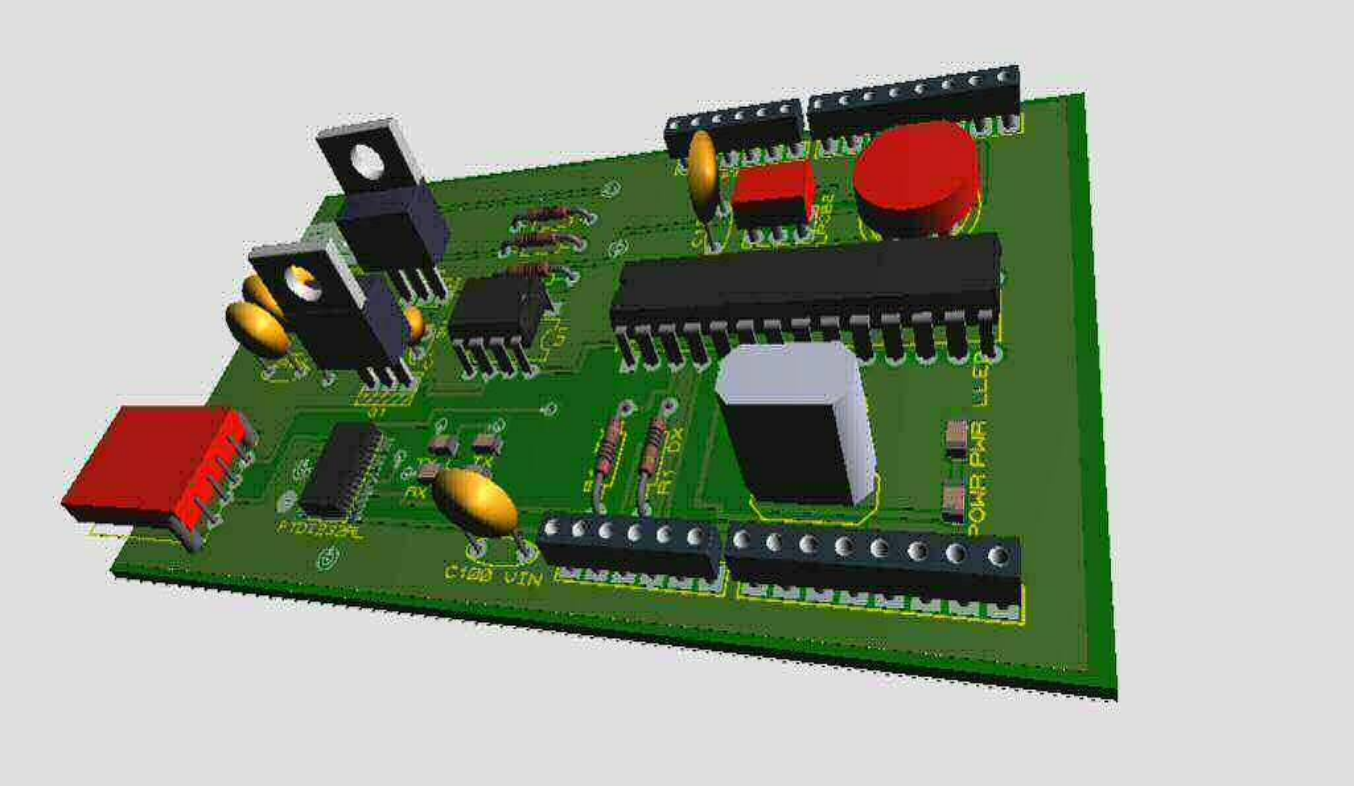


Figure (III-16) : Vue 3D de la Carte Arduino

III-3-4) Imprimé sur du papier calques (typon) :

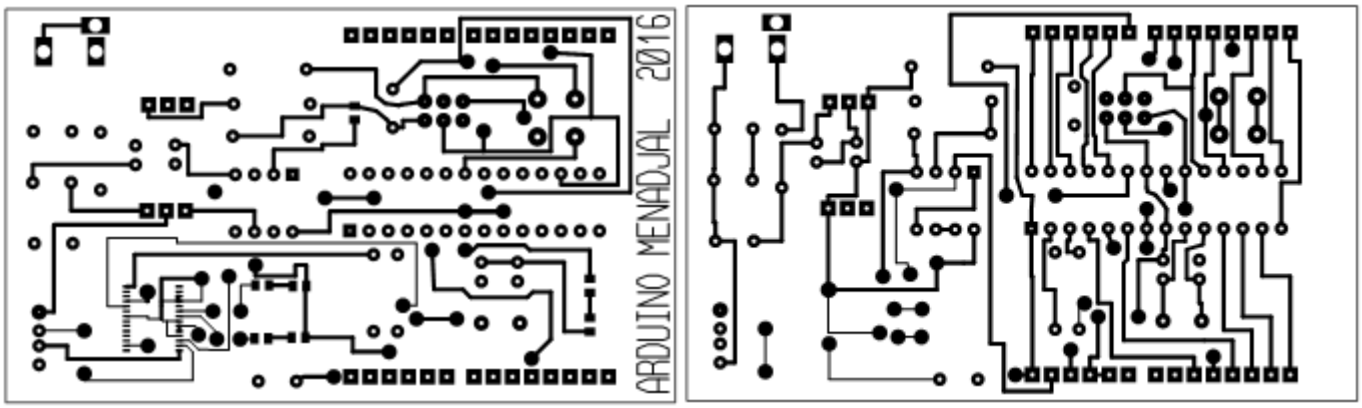


Figure (III-17) : Imprimé du circuit ARES sur Du papier Calque (typon)

III-3-5) Circuit Imprimé :

Nous avons réalisé le circuit imprimé en utilisant la méthode de l'insoleuse (une boite munie de néons), cette méthode consiste à prendre l'époxy et lui coller le typon, et le mettre dans l'insoleuse pendant trois minutes. Le principe est que les Ultraviolets émis par les néons vont détruire la couche photosensible de l'époxy aux endroits non protégés par le tracé du typon.

Une fois le temps écoulé, on sort la plaque pré sensibilisée, on lui retire le typon, puis on la trompe dans une solution révélatrice, puis on la plonge dans un bain de perchlorure de fer en la frottant pour supprimer le cuivre là ou il n'y a pas de liaisons. Le résultat est :

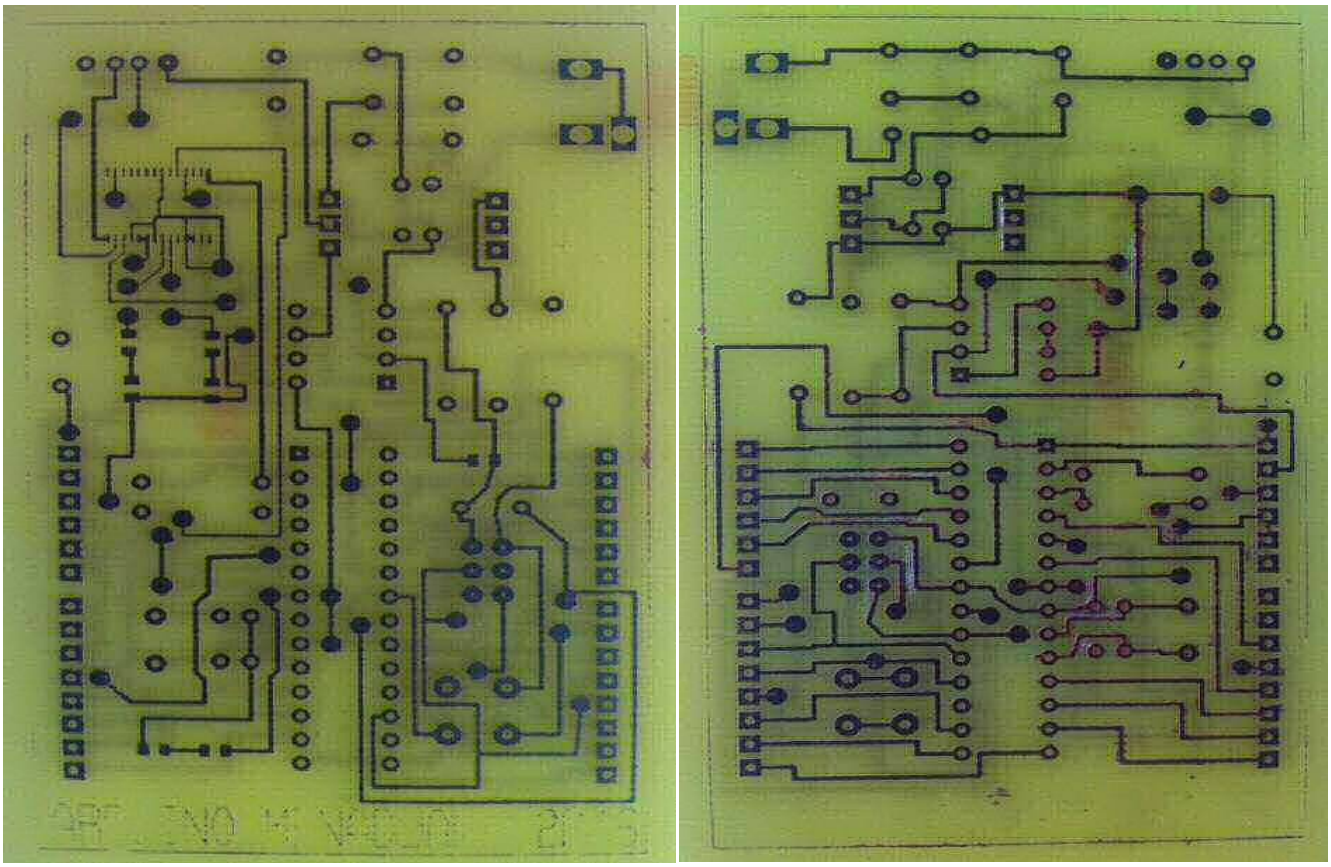


Figure (III-18) : Circuit Imprimé de la carte Arduino (Face de dessus à gauche et face du dessous à droite)

III-3-6) Soudure :

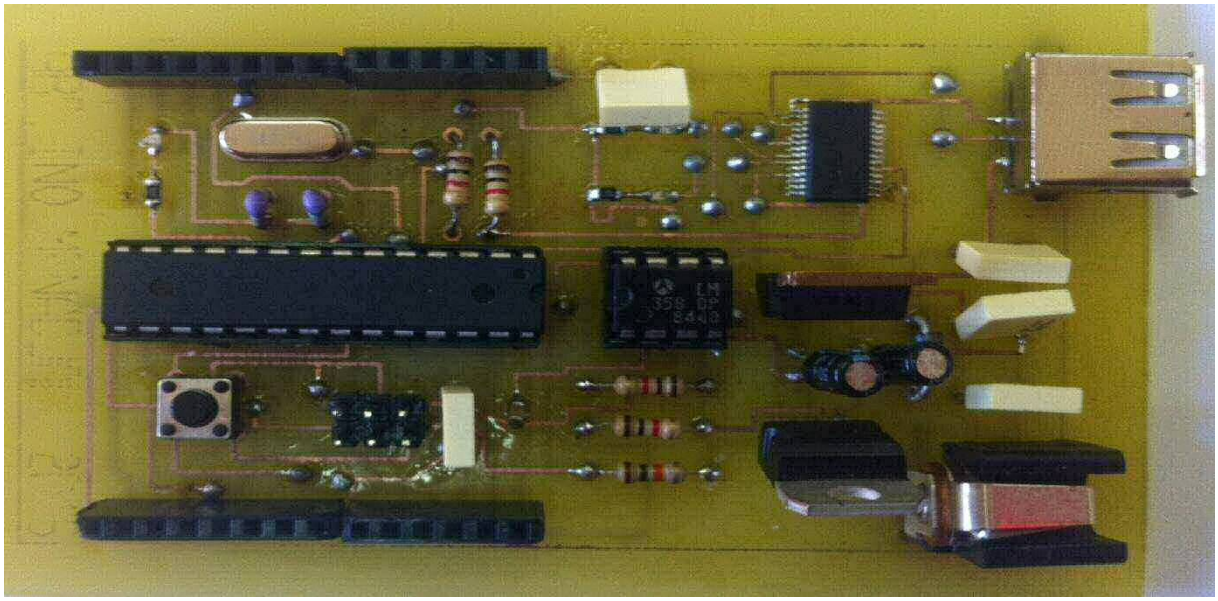


Figure (III-19) : Le résultat final de la Carte Arduino

III-4) Exemples d'application :

III-4-1) Blink

L'exemple Blink est le premier test fait avec la carte, son schéma ainsi que son programme sont cités dans les pages (26, 27).

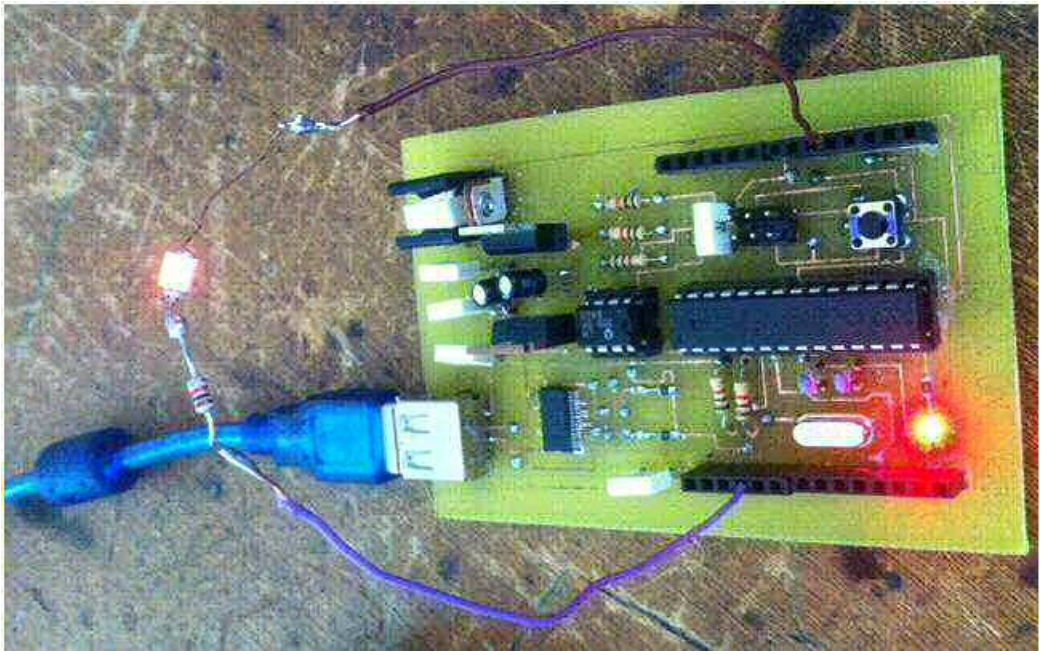


Figure (III-20) : Teste de la carte avec le programme Blink

III-4-2) Afficheur LCD

Pour ce test, nous avons eu recourt à un afficheur LCD 16 pins ainsi que la carte Arduino, le programme qui va suivre nous donne les entrées /sorties de la carte utilisées :

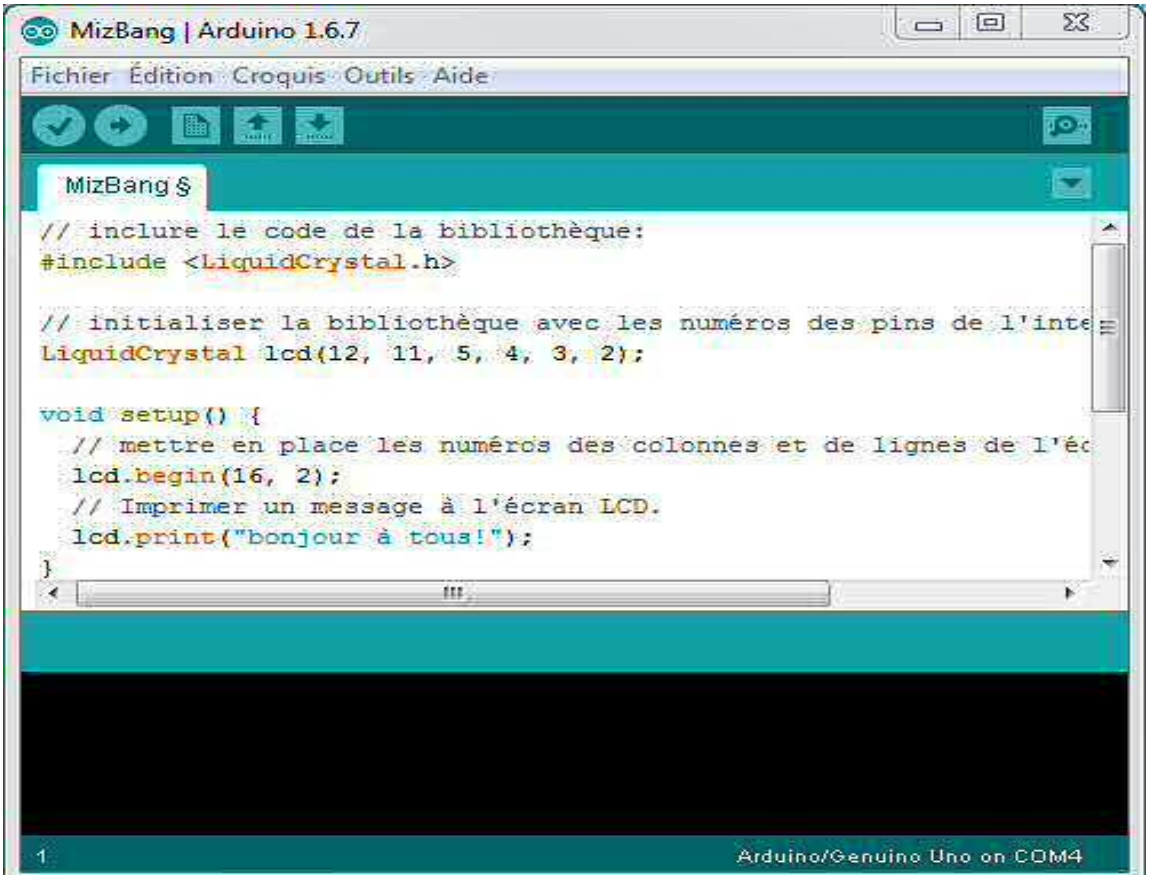


Figure (III-21) : Programme MizBang injecté dans notre carte Arduino

- **Description des pins de l'afficheur LCD**
  - → Les pins 1, 2 et 3 de l'afficheur sont des broches d'alimentation, on peut les utiliser pour le réglage du contraste en y associant un potentiomètre,
  - → le pin 4 pour sélectionner le registre,
  - → le 5 pour lire/écrire sur l'afficheur,
  - → le 6 pour la validation,
  - → les pins 7 à 14 sont des entrées de données,
  - → les deux dernières sont pour illuminer l'afficheur.

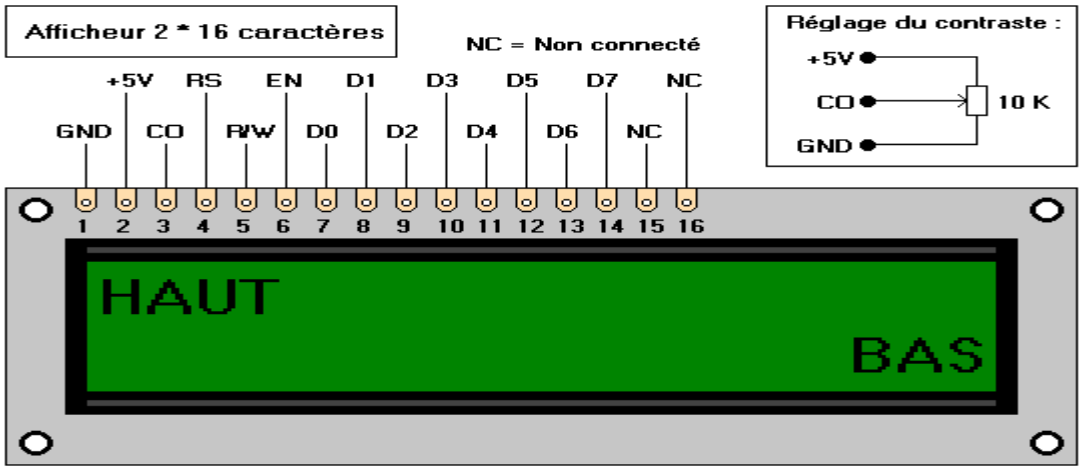


Figure (III-22) : Afficheur LCD GDM1602A

Comme l'indique le programme, nous avons connecté les pins 2, 3, 4, 5,11 et 12 de la carte au pins 11, 12, 13, 14, 3, 4 et 5 de l'afficheur respectivement, le schéma suivant montre le raccordement :

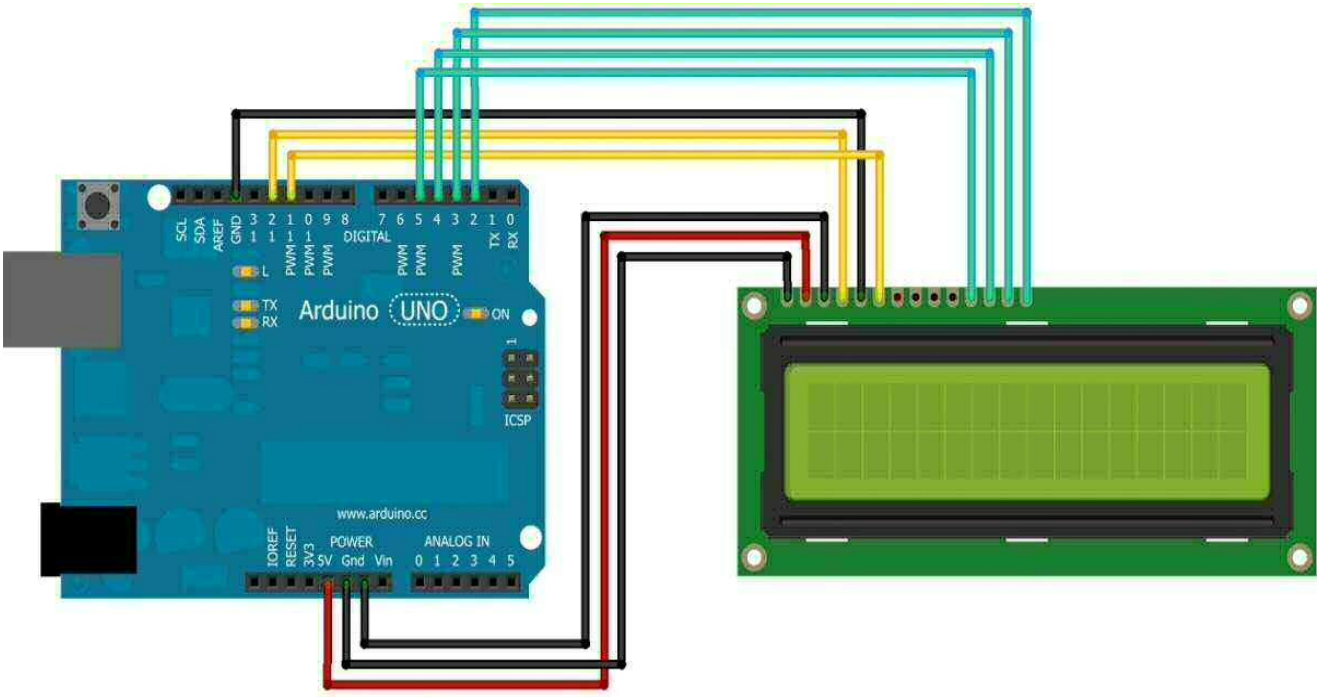


Figure (III-23) : Circuit de l'exemple Afficheur LCD

Le résultat est :

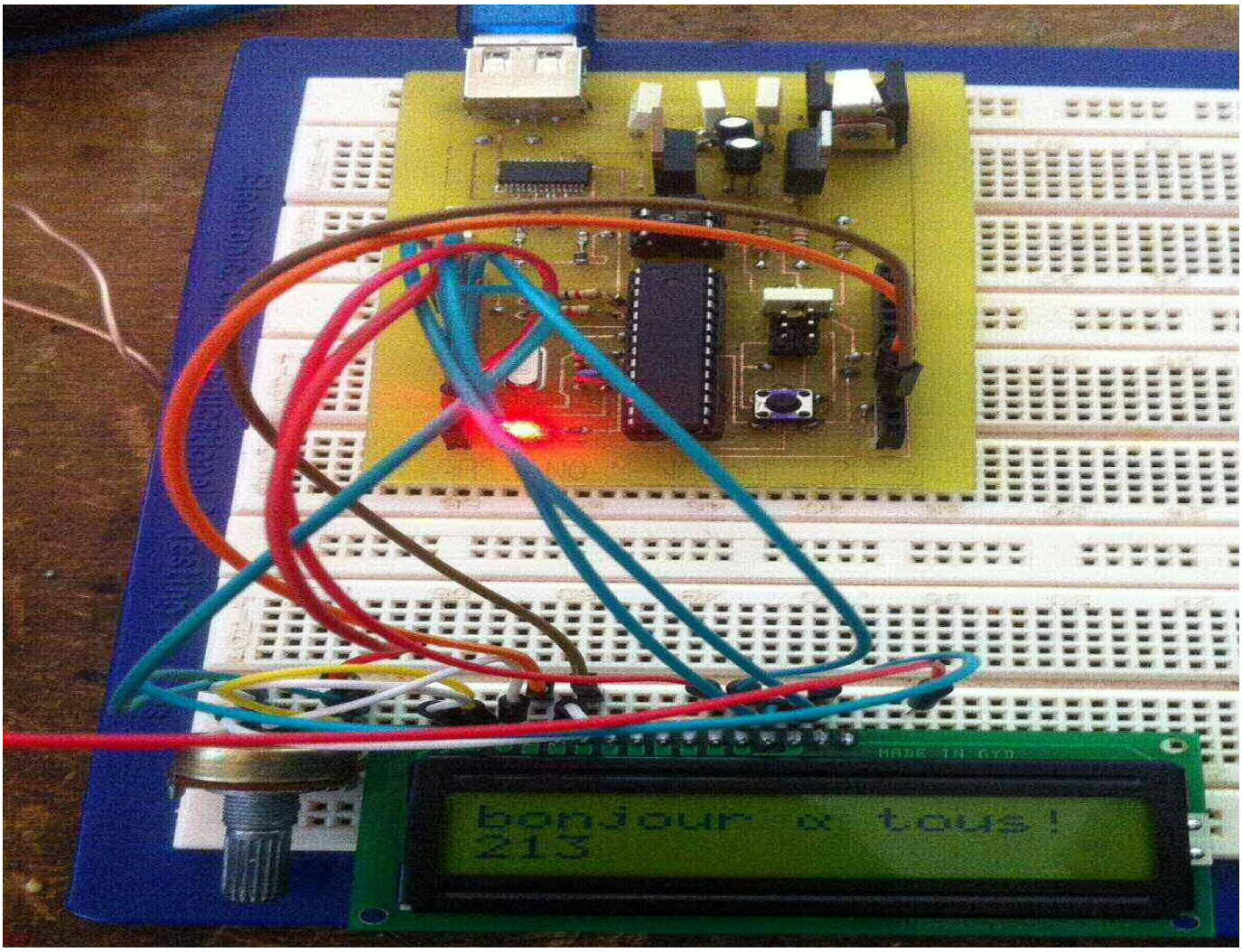


Figure (III-24) : Teste de l'exemple Afficheur LCD

III-4-3) maquette polyvalente :

La maquette polyvalente est un assemblage de plusieurs modules (un circuit de puissance et un circuit de commande) connectés à la carte. Notre but étant de voir la diversité de notre carte, on la testant avec un maximum de périphériques et circuits externes.

Pour cette exemple, nous aurons besoin de :

- Notre carte Arduino.
- Un module BT (Bluetooth) : pour émettre et recevoir les données avec la carte Arduino.
- Un Smartphone sous Androïd : pour télécharger l'application *Arduino Control Car* à fin de choisir la direction.
- Deux moteurs.
- Le pont-H L293D pour gérer la puissance fournie aux deux moteurs.
- Un écran LCD : pour afficher la direction suivie et la durée.
- Un potentiomètre : pour régler le contraste de l'afficheur LCD.
- Deux Roues.

- Deux piles de 12V : pour l'alimentation.
- Un BreadBord (plaque d'essai).
- Deux LEDs pour signaler la direction.
- Deux Résistances 1KΩ.
- Des fils de connexion.

La maquette est composée de 4 parties, l'organigramme qui va suivre nous donne une idée sur le déroulement :

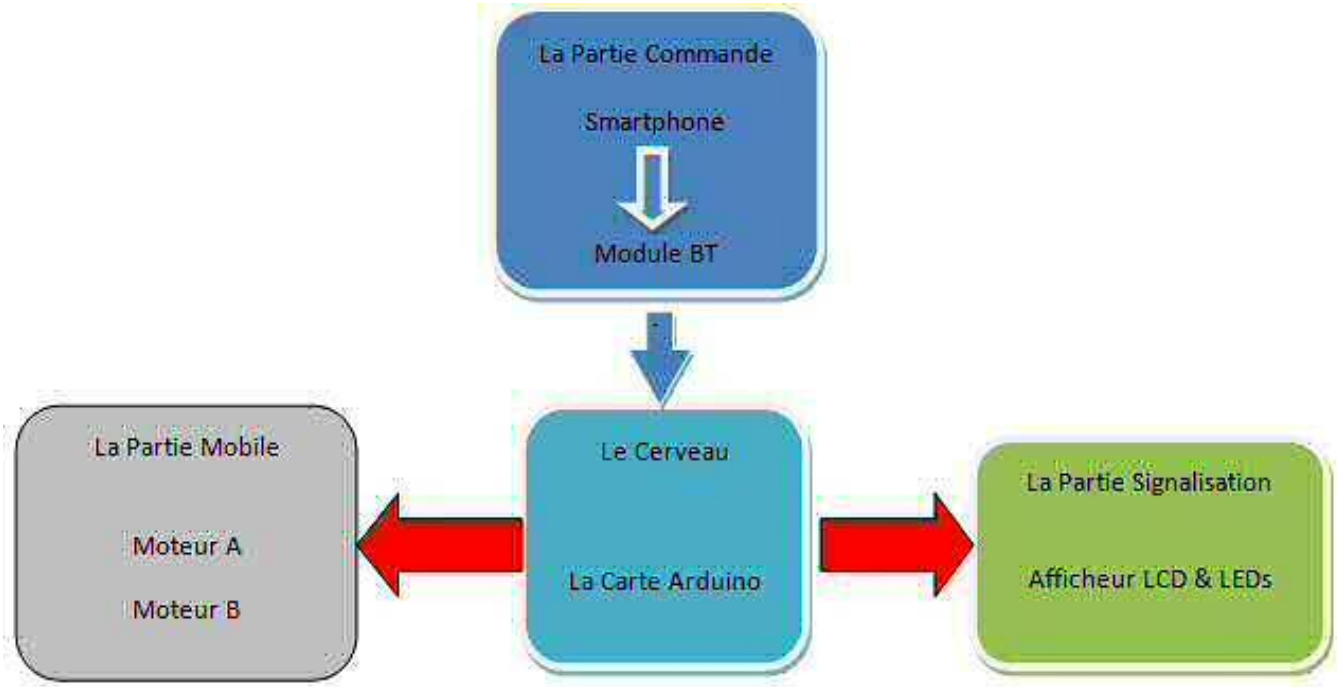


Figure (III-25) : Organigramme de la Maquette Polyvalente.

- **Branchement :**

1) ARDUINO	Afficheur LCD
pin2	pin14
pin3	pin13
pin4	pin12
pin5	pin11
pin6	pin6
pin7	pin4
5V	pin3
GND	pin1, 5
pins12, 13	2 LEDs

2) ARDUINO	L239D (pont-H)
pin8	pin2
pin 9	pin7
pin10	pin10
pin11	pin15
5V	pin1, 9, 16
GND	pin4, 5, 12, 13

3) ARDUINO	Module BT
pin0 (RX)	TX
pin1 (TX)	RX
5V	VCC
GND	GND

4) L239D Pont H	Moteurs
pins 3 et 6	Moteur A.
pins 11, 14	Moteur B.

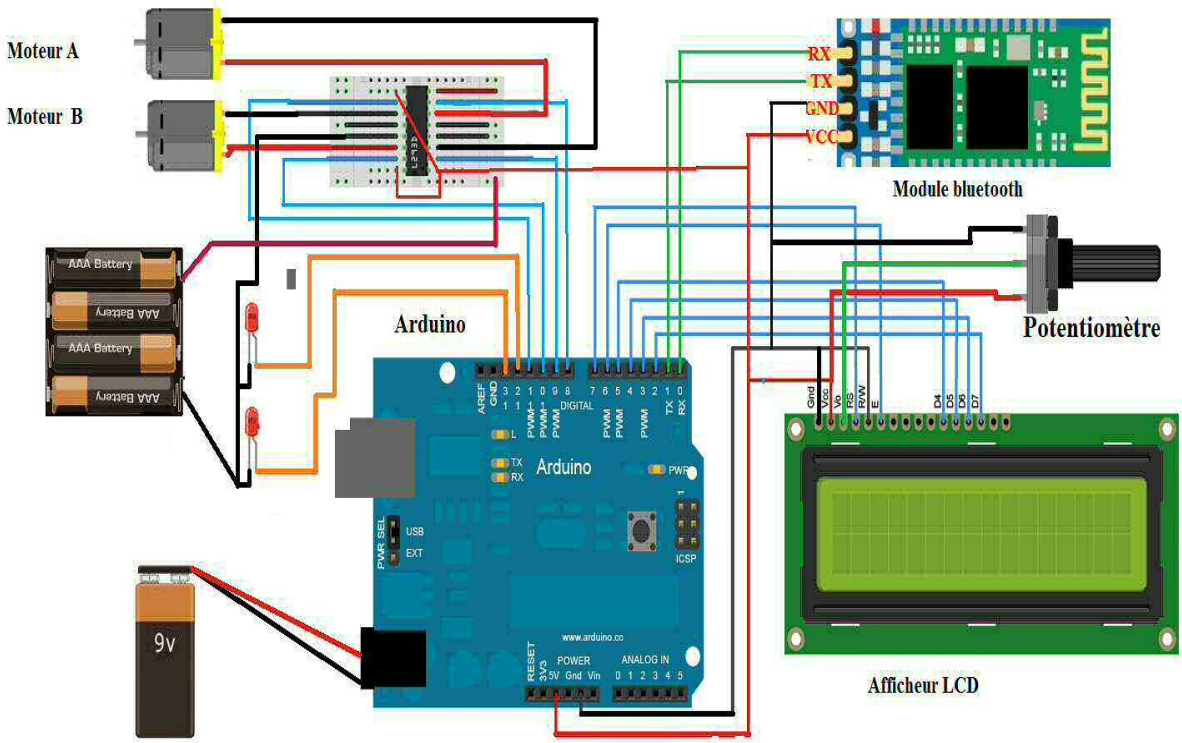


Figure (III-26) : Schéma de la Maquette Polyvalente

- **Description de l'application**

L'application *Arduino Control Car* est un programme autonome conçu pour s'exécuter sur un terminal mobile, comme un Smartphone ou une tablette tactile, cette application envoie le caractère sélectionné au module Bluetooth

Flèche haut = caractère 'a'
Flèche gauche = caractère 'b'
Bouton carré = Caractère 'c'
Flèche droite = caractère 'd'
Flèche bas = caractère 'e'
Bouton ON = Caractère 'f'
Bouton OFF = Caractère 'g'
Bouton Bluetooth = consiste à sélectionner le module BT avec lequel nous nous connectons.(Le module BT doit d'abord être synchronisé avec le Smartphone).

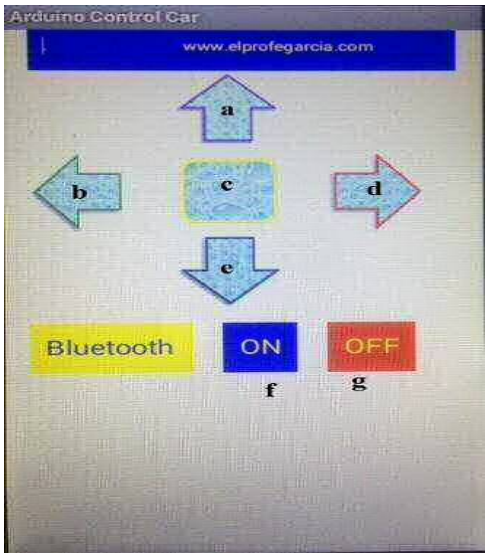


Figure (III-27) : Vue de l'application

- **Déroulement et fonctionnement**

→ Dans la partie de commande :

Une fois le Smartphone connecté au module BT, on ouvre l'application *Arduino Control Car*, un menu comme sur la figure 73 apparait, puis on clique sur « Bluetooth » puis on sélection le module HC-06.

Il nous suffit maintenant d'appuyer sur l'une des icones de direction, et un signale s'envoi du Smartphone vers le module BT, qui lui même le transmet à partir de pin RX et TX vers le Cerveau (la carte Arduino).

→ Dans le Cerveau

Une fois le signal arrivé, la carte le décode pour identifier le message (Lire le caractère a, b,....g), ensuite elle envoie des signaux de commande à la partie signalisation et à la partie mobile.

→ Dans la partie signalisation et la partie mobile

Type de signal	Type de commande	Moteurs	Afficheur LCD	LEDs
Caractère 'a'	Marche avant	Moteurs A, B tournent +	Marche avant	LED 1et 2 clignotes
Caractère 'b'	Tourne à droite	Moteur B tourne +	Tourne à droite	LED 1 s'allume
Caractère 'c'	Arrêt	Les deux s'arrêtent	Arrêt	Les deux s'éteignent
Caractère 'd'	Tourne à gauche	Moteur A tourne +	Tourne à gauche	LED 2 s'allume
Caractère 'e'	Marche arrière	Moteurs A, B tournent -	Marche arrière	LED1et2 s'allume

Programme :

car | Arduino 1.6.7

Fichier Édition Croquis Outils Aide

car §

```
#include <LiquidCrystal.h>
//initialiser la bibliothèque avec les chiffre des pins de l'interface
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);//affecter les 13-8 pour l'afficheur LCD
int led1 = 13; //affecter la pin 13 pour la led1
int led2 = 12; //affecter la pin 12pour la led2
int avA = 8;   //affecter la pin 8 pour la pin 10 du L293D, pour la marche avant du moteur A
int avB = 9;   //affecter la pin 9 pour la pin 2 du L293D, pour la marche avant du moteur B
int arA = 10;  //affecter la pin 10 pour la pin 7 du L293D, pour la marche arrière du moteur A
int arB = 11;  //affecter la pin 11 pour la pin 15 du L293D, pour la marche arrière du moteur B
int val = 255;     // Valeur max moteurs (0-255) pour controler la vitesse de rotation
int ist = 'g';     // initialiser detenido

void setup()  {
    Serial.begin(9600);    //  initailiser le port série pour la communication bluetooth
  pinMode(arA, OUTPUT);  // initailiser arA comme sortie
  pinMode(arB, OUTPUT);  // initailiser arB comme sortie
  pinMode(avA, OUTPUT);  // initailiser avA comme sortie
  pinMode(avB, OUTPUT);  // initailiser avB comme sortie
  pinMode(led1, OUTPUT);  // initailiser Led1 comme sortieo
  pinMode(led2, OUTPUT);  // initailiser Led2 comme sortieo
  lcd.begin(16, 2);       // Peciser le genre d'afficheur, c-à-d un afficher 16 colones, 2 lignes
  lcd.setCursor(1, 0);    //Mettre le curseur dans la case (1, 0) au debut de la ligne 1
  lcd.print("<< EN MARCHE >>  "); // imprimer le message "EN MARCHE"
  lcd.setCursor(2, 1);    //Mettre le curseur dans la case (2, 1) au debut de la ligne 2
```

```
  lcd.print("SELECT DIRECT"); // imprimer le message "SELECT DIRECT" pour selectionner la direction

   }

void loop()  {

  if(Serial.available()>0){        // Si le bluetooth est allumé
      ist = Serial.read();         // Lire l'instruction du module Bluetouth
      lcd.clear();                 // effacer le message
  }
  if(ist=='a'){              // déplacement vers l'avant
      lcd.setCursor(1, 0);
      lcd.print("MARCHE AVANT");//imprimmer le message "MARCHE AVANT"
      lcd.setCursor(2, 1);
      lcd.print(millis() / 1000);// enclanché le chronomètre
      analogWrite(arB, 0);      //valeur de la sortie arB =0
      analogWrite(avB, 0);      //valeur de la sortie avB =0
      analogWrite(arA, val);    //valeur de la sortie arA =val
      analogWrite(avA, val);    //valeur de la sortie avB =val
      // ces instructions font que les deux moteurs tourne à la valeur maximale dans le sens avant
      digitalWrite(led1, HIGH);//allumer la Led1
      digitalWrite(led2, HIGH);// allumer la Led2
      delay(100);               // delay de 100ms
      digitalWrite(led1, LOW); //etteinre la Led1
      digitalWrite(led2, LOW); //etteinre la Led2
      delay(100);               // delay de 100ms
      lcd.setCursor(2, 1);
      lcd.print(millis() / 1000); //imprimer le chronomètre
  }

  if(ist=='b'){          // tourner adroit

      lcd.setCursor(1, 0);
      lcd.print("TOURNE A GAUCHE");//imprimmer le message "TOURNE A GAUCHE"
      lcd.setCursor(2, 1);
      lcd.print(millis() / 1000);
      analogWrite(arB, 0);
      analogWrite(avB, 0);
      analogWrite(arA, 0);
      analogWrite(avA, val);
      digitalWrite(led1, LOW);  //etteindre la Led1
      digitalWrite(led2, HIGH); // allumer la Led2
      }
      //Ces instruction font que le moteur A s'arrete tandis que le moteur B marche,
      ////pour generer une rotation vers la gauche

  if(ist=='c'){         // stop
      lcd.setCursor(1, 0);
      lcd.print("ARRET");//imprimer le message "ARRET"
      lcd.setCursor(2, 4);//mettre le curseur dans la case(2, 4)
      lcd.print(millis() / 1000);
      analogWrite(arB, 0);
      analogWrite(avB, 0);
      analogWrite(arA, 0);
```

```
    analogWrite(avA, 0);
    digitalWrite(led1, LOW);//etteindre le Led1
    digitalWrite(led2, LOW); //etteindre le Led2

}
//Ces instruction font que les deux moteurs s'arrêtent
if(ist=='d'){          // tourne à droite
    lcd.setCursor(1, 0);
    lcd.print("TOURNE A DRIOTE");imprimer le message "tourne à droite"
    lcd.setCursor(2, 1);
    lcd.print(millis() / 1000);
    analogWrite(arB, 0);
    analogWrite(avB, 0);
    analogWrite(avA, 0);
    analogWrite(arA, val);
    digitalWrite(led1, HIGH);//allumer le Led1
    digitalWrite(led2, LOW); //etteindre le Led2
}
  //Ces instruction font que le moteur A s'enclanche tandis que le moteur B s'arrête,
    ////pour generer une rotation vers la droite
if(ist=='e'){          // deplacement arier

    lcd.setCursor(1, 0);
    lcd.print("MARCHE ARRIERE ");//imprimer le message "MARCHE ARRIERE"
    lcd.setCursor(2, 1);
    lcd.print(millis() / 1000);
    analogWrite(arA, 0);
    analogWrite(avA, 0);
    analogWrite(arB, val);
    analogWrite(avB, val);
    digitalWrite(led1, HIGH);// allumer les deux LEDs
    digitalWrite(led2, HIGH);//
}


}
```

Figure (III-28) : Programme "car" de la maquette polyvalente.

Le résultat :

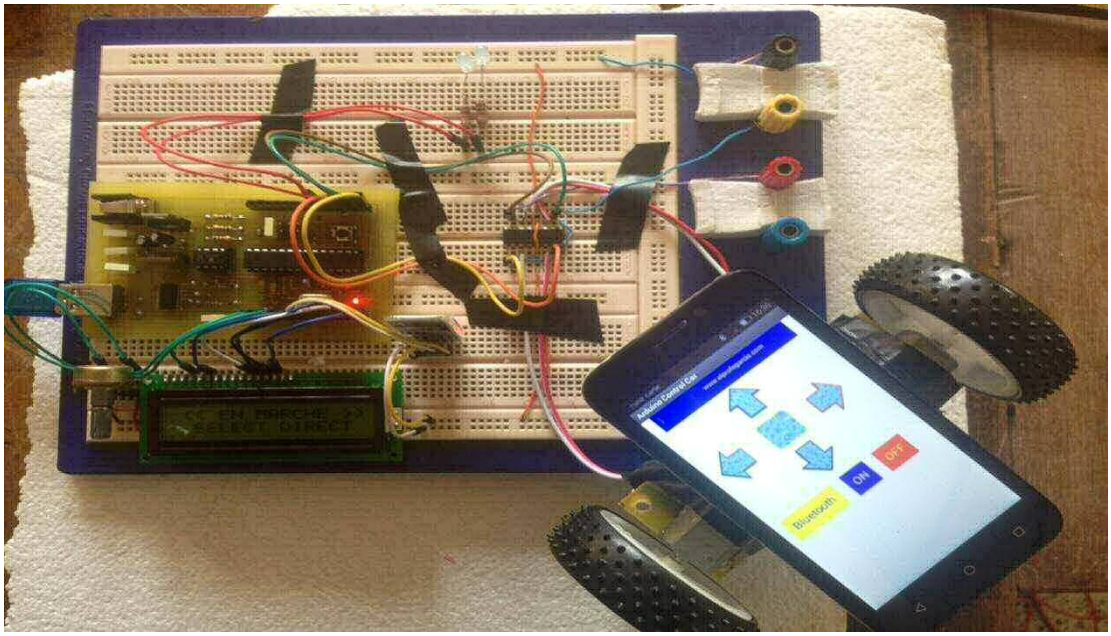


Figure (III-29) : Teste de la Maquette polyvalente

Conclusion

Ce chapitre montre comment réaliser la carte sur un BreadBord, puis sur un circuit imprimé, en manipulant les composants, en soudant les circuits intégrés. Il montre aussi la programmation avec les deux maquettes, en réalisant de petits exemples d'application comme l'exemple Blink et l'afficheur LCD. L'exemple de maquette polyvalente associe plusieurs circuits additionnels en plus de la carte, tel qu'un circuit de puissance ou encore un circuit de commande, tout en ayant le but de tester les limites de la carte.

On s'est rendu compte que la carte était puissante, elle pouvait traiter une importante masse de données, mais plus le programme est lourd, plus le temps du chargement devient long, et on a constaté aussi qu'elle consomme plus d'énergie au fur et à mesure qu'on lui raccorde des circuits interfaces.

Conclusion Générale

Conclusion Générale

Ce projet donne une vue globale et approfondie sur la platefotme Arduino, commençant par un aperçu sur son parcours et en citant quelques uns de ces modèles et domaines d'utilisation, passant par sa composition matérielle et une discription de son logicielle, et enfin, terminant par la réalisation, d'abord sur une plaque d'essaie, ensuite sur un circuit imprimé. Cette dernière partie nous a appris les étapes à suivre pour fabriquer une carte Arduino.

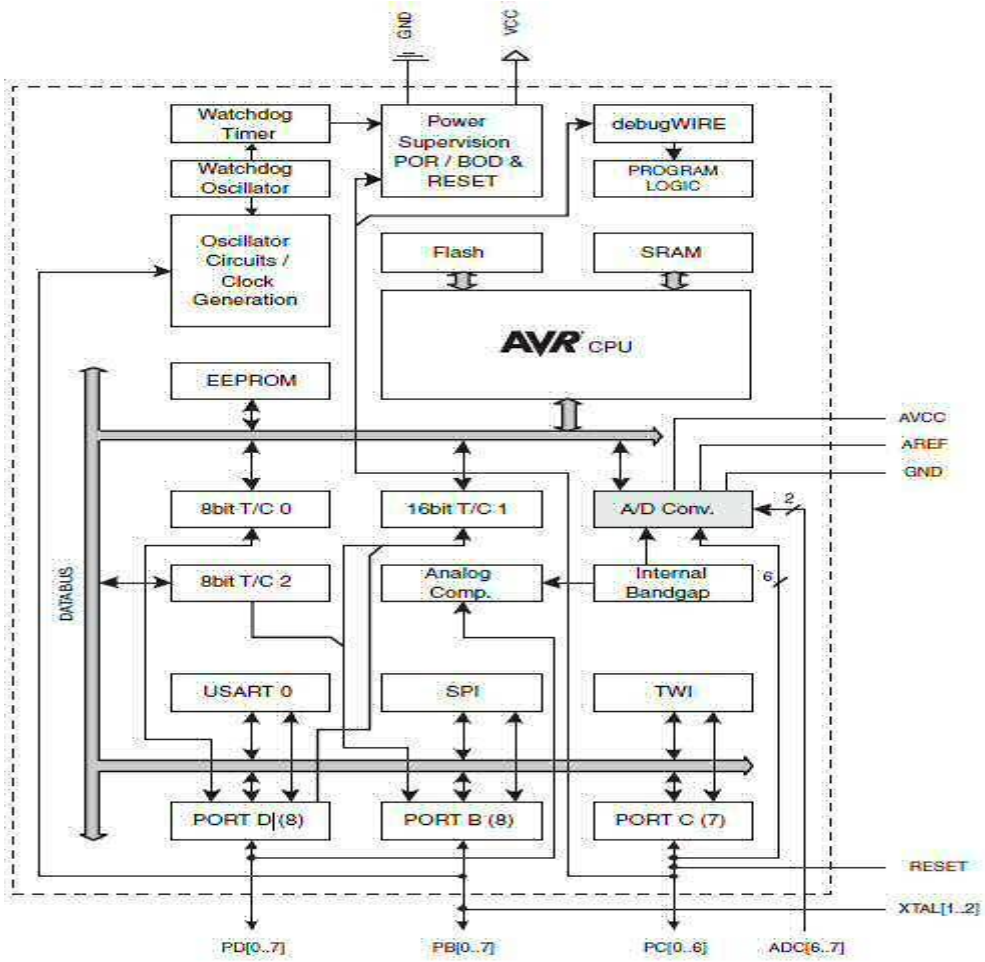
Ce projet nous a aussi permis de réaliser des projets et des applications en utilisant la carte fabriquée, et en ayant recourt à quelques modules et composant pour les interface externes, cela nous a permis de déduire que sans avoir de profondes connaissances sur les microcontrôleurs, seulement il fallait maitriser la partie Software avec quelques notions des composants nécessaire à la réalisation. Ces exemples ont pour but de s'assurer du bon fonctionnement de la carte réalisée.

Autres que les exemples cités et testés, la carte peut aussi servir à commander des bras articulés en robotique, ou encore contrôler des servomoteurs pour tous projets futurs en automatique.

Annexes

Annexe 1 : L'ATMega328P :

Le diagramme de l'ATMega328 ressemble à chose près à :



Architecture interne de l'ATMega328P

1) Architecture interne de l'ATMega328 :

Le microcontrôleur est devisé en deux parties, le noyau et les périphériques :

1-1) Le noyau : C'est un automate séquentiel dont le rôle est la lecture**,** le décodage puis l'exécution des instructions et qui constituent le programme, disposant d'un large chemin de données (bus) allant de 4 bits pour les modèles les plus basiques a 32 ou 64 bits pour les modelés les plus évolués, on y trouve :

1-1-1) L'unité arithmétique et logique ALU : elle représente le registre d'accumulateur, qui se charge des exécutions des opérations arithmétiques de base (addition, soustraction,…), ainsi que les opérations logique de base (ET, OU, ….).

1-1-2) La mémoire programme FLASH : Elle contient le code binaire correspondant aux instructions que doit exécuter le microcontrôleur, sa capacité est de 32KB dont 0.5KB est réservé pour le bootloader.

1-1-3) Le registre compteur programme : est un séquenceur qui est chargé de pointeur l'adresse mémoire courante contenant l'instruction à traiter par le microcontrôleur. Le contenu de registre évolue selon le pas de programme.

1-1-4) Le registre d'instruction : est un registre de 28 bits de taille, il contient le code binaire correspondant à l'instruction en cours d'exécution.

1-1-5) La RAM : La mémoire RAM (Random Access Memory) est une mémoire à accès aléatoire qui peut être **lue** ou **écrite** indéfiniment. Dans les microsystèmes, elle est souvent réservée pour stocker des variables, des résultats intermédiaires et sert également de « pile » pour le processeur. Comparativement à la ROM, le buffer de

sortie est **bidirectionnel** et c'est le signal de lecture/écriture issu des lignes de contrôle qui fixe le sens de circulation des données.

1-1-6) Les Registres d'Etat et de Pile : Deux registres sont indispensables au système pour fonctionner, le registre d'état **SREG** et le registre de gestion de la pile pour les interruptions et la gestion des sous-programmes.

a) Registre SREG (Status Register) :

Le registre **SREG** ou registre d'état sert principalement avec les fonctions arithmétiques et logiques pour les opérations de branchements. Il indique et autorise aussi le fonctionnement des interruptions. Il est modifié par le résultat des manipulations mathématiques et logiques. C'est le principal registre qui sert à effectuer les branchements conditionnels après une opération arithmétique ou logique. Il peut être lu et écrit à tout moment sans aucune restriction.

b) Registre de la Pile (STACK Register) :

Le registre de pointeur de pile est utilisé par les instructions **PUSH** et **POP** de gestion de pile. La gestion de pile utilise les deux registres 8 bits qui suivent pour former une adresse 16 bit. L'adresse ainsi crée doit être positionné en général en fin de la mémoire **SRAM**. La pile est décrémenté avec l'instruction **PUSH** et incrémenté avec **POP**, donc le positionnement en fin de **SRAM** ne pose aucun problème de taille en général, mais attention aux boucles trop grande avec des **PUSH** qui pourrait arriver dans l'espace de stockage des variables programmes ou système.

c) Registres Spéciaux :

Ils sont au nombre de 32 et travaille directement avec l'ALU, ils permettent à deux registres indépendant d'être en accès directs par l'intermédiaire d'une simple instruction et exécutée sur un seul cycle d'horloge. Cela signifie que pendant un cycle d'horloge simple l'Unité Arithmétique et Logique **ALU** exécute l'opération et le résultat est stocké en arrière dans le registre de sortie, le tout dans un cycle d'horloge. L'architecture résultante est plus efficace en réalisant des opérations jusqu'à dix fois plus rapidement qu'avec des microcontrôleurs conventionnels **CISC**.

Les registres spéciaux sont dit aussi registre d'accès rapide et 6 des 32 registres peuvent être employés comme trois registre d'adresse 16 bits pour l'adressage indirects d'espace de données (**X, Y & Z**). Le troisième **Z** est aussi employé comme indicateur d'adresse pour la fonction de consultation de table des constantes.

Les 32 registres sont détaillés dans le tableau qui suit avec l'adresse effective dans la mémoire Les informations sont diffusées par un bus de donnée à 8 bits dans l'ensemble du circuit.

Bits 7 à 0	Adresses	Registres Spéciaux
R0	00	
R1	01	
Rn	xx	
R26	1A	Registre **X** Partie Basse
R27	1B	Registre **X** Partie Haute
R28	1C	Registre **Y** Partie Basse
R29	1D	Registre **Y** Partie Haute
R30	1E	Registre **Z** Partie Basse
R31	1F	Registre **Z** Partie Haute

Le microcontrôleur possède aussi un mode sommeil qui arrêter l'unité centrale en permettant à la **SRAM**, les Timer/Compteurs, l'interface **SPI** d'interrompre la veille du système pour reprendre le fonctionnement. Lors de l'arrêt de l'énergie électrique, le mode économie sauve le contenue des registres et gèle l'oscillateur, mettant hors de service toutes autres fonctions du circuit avant qu'une éventuelle interruption logicielle ou matérielle soit émise.

Dans le mode économie, l'oscillateur du minuteur continue à courir, permettant à l'utilisateur d'entretenir le minuteur **RTC** tandis que le reste du dispositif dort. Le dispositif est fabriqué en employant la technologie de mémoire à haute densité non volatile d'**ATMEL**.

La mémoire **FLASH** est reprogrammable par le système avec l'interface **SPI** ou par un programmateur de mémoire conventionnel non volatile (voir le chapitre sur la programmation du **MPU**).

1-1-7) Les Registres Systèmes

Les registres systèmes permettent de programmer le microcontrôleur selon le choix d'utilisation que le programmeur veut en faire. Ils sont aux nombres de 4 :

- Registre MCUCR (MCU Control Register),
- Registre MCUSR (MCU Control ans Status),
- Registre OSCCAL (Oscilator Calibration Register),
- Registre SPMCR (Store Program Memory Control Register),

Mais seuls les deux premiers sont importants. Le fonctionnement du microcontrôleur est modifié lors de la programmation de la mémoire **FLASH**, le choix d'horloge.

1-1) Les périphériques :

Ce sont les parties qui font la différence entre le microcontrôleur et le microprocesseur. Ils facilitent l'interfaçage avec l'environnement extérieur et performent les tâches internes comme la génération des différentes bases de temps, On peut citer :

1-2-1) Les Entrées/Sorties Numériques (PORTx)

Les microcontrôleurs **ATMEL** sont pourvus de ports d'entrées/sorties numériques pour communiquer avec l'extérieur. Ces ports sont multidirectionnels et configurable broche à broche soit en entrée, soit en sortie.

D'autres modes sont aussi utilisables comme des entrées analogiques, des fonctions spéciales de comparaison, de communication synchrone.

L'ATMega328 possède 3 ports : les PortB, C, et D (soit 23 broches en tout I/O).

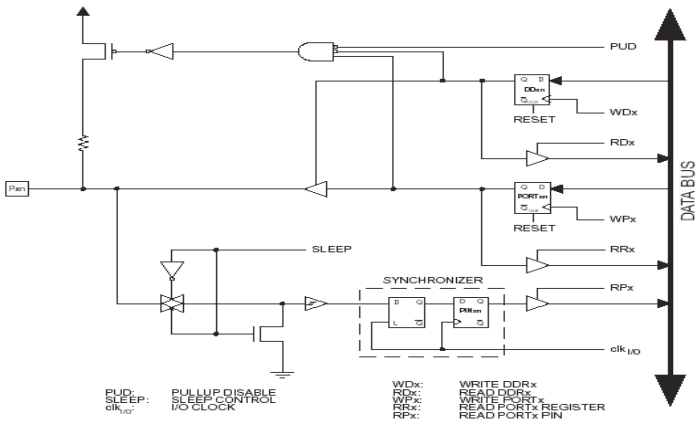


Schéma interne des E/S

Trois registres sont disponibles pour contrôler ces ports :

- **DDRx** : Registre de direction (Mode Entrée, Sortie)
- **PORTx** : Registre de donnés (Donnée à lire ou à écrire)
- **PINx** : Registre d'état (Donnée disponible)

x Représente le nom de port (A, B, C ou D).

Chaque broche de port E/S a une résistance de pull-up interne qui peut être désactivée. Le bit **PUD** du registre MCUCR permet la désactivation des résistances de pull-up.

→ **Direction des ports** : Si le bit **DDRB.2** vaut 1 alors la broche **PB2** est une sortie TOR.

→ **Ecriture des sorties TOR** : si une broche est configurée en sortie (DDRx.n=1) alors l'écriture du bit PORTx.n permet de définir l'état de la sortie (0 ou 1).

→ **Lectures des entrées TOR** : si une broche est configurée en entrée (DDRx.n=0) alors la lecture du bit PINx.n permet de connaitre l'état de l'entrée.

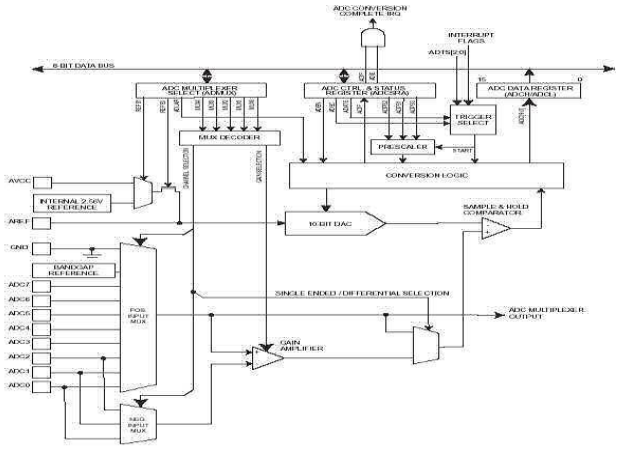
1-2-2) La mémoire EEPROM

L'EEPROM est une mémoire programmable est effaçable électriquement. La particularité de cette mémoire et de pouvoir garder les informations stockées longtemps même hors tension. L'ATMega328 contient 1024 octets de données dans la mémoire EEPROM. Elle est organisée comme un espace de donné séparé, dans lequel des octets simples peuvent être lus et écrits. L'EEPROM a une endurance d'au moins 100,000 cycles d'écriture/effacement. Les trois registres utilisés par l'EEPROM sont :

- Registre EEAR (The EEPROM Address Register).
- Registre EEDR (The EEPROM Data Register).
- Registre EECR (The EEPROM Controle Register).

1-2-3) Les convertisseurs analogiques-numériques CAN :

Le convertisseur analogique/numérique **ADC** intégré dans l'**ATMEGA** est doté de caractéristique très intéressante avec une résolution sur 10 bits, 8 entrées simultanées avec une non-linéarité inférieure à 1/2 **LSB**, une erreur à 0 V inférieur à 1 **LSB**, le temps de conversion est réglable de 65 à 260 **µS** plus le temps est long, plus le résultat est précis. Près de 15000 échantillons/secondes avec le maximum de résolution son possible. Le convertisseur possède 7 entrées différentielles normales et 2 entrées différentielles avec gain optionnel de 10x (et 200x sur les boîtiers carré **TQFP** et **MLF** uniquement). La tension de référence peur être externe (conversion de **0** à **AREF** analogique, le maximum étant **VCC**) ou peut être interne avec la tension de référence de **2,56 V**. L'**ADC** à une interruption sur conversion complète. La limitation du bruit en mode de sommeil est possible.



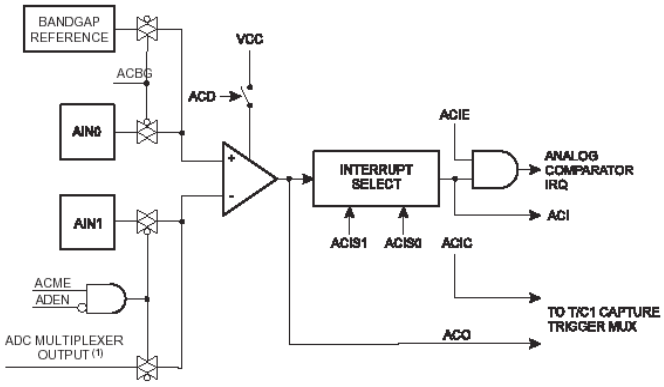
Architecture interne du CAN

Les registres utilisés par **l'ADC** son aux nombres de quatre :

- Registre ADMUX (ADC Multiplexer Selection Register)
- Registre ADCSR (ADC Control and Status Register)
- Registre ADCH et ADCL (ADC Data Register)
- Registre SFIOR (Special Function I/O Register)

1-2-4) Le Comparateur Analogique (AC)

Le comparateur analogique compare les valeurs d'entrée sur la broche positive **AIN0** (**PB2**) et la broche négative **AIN1 (PB3)**. Quand la tension sur la broche **AIN0** est plus haute que la tension sur la broche **AIN1**, le comparateur analogique **ACO** est mis à 1. Le comparateur analogique peut être utilisé pour déclencher la fonction de capture d'entrée du Timer/Compteur1. De plus, le comparateur analogique peut déclencher une interruption séparée exclusive. L'utilisateur peut choisir le front montant ou descendant pour déclencher la sortie **ACO**. L'entrée négative du comparateur peut être l'une des 8 entrées analogiques de **l'ADC**.

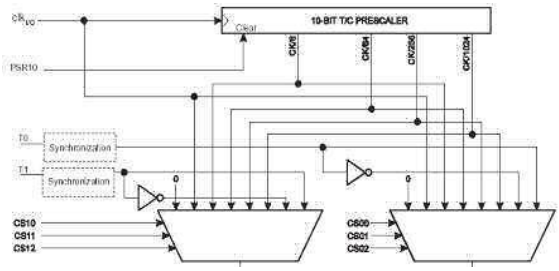


Architecture interne du AC

Le comparateur utilise un registre propre, Le **Registre ACSR (Analog Comparator Control and Status Register)**, et partage les deux registres **Registre SFIOR (Special Function I/O Register)**, **Registre ADMUX (ADC Multiplexer Selection Register)** avec d'autre module.

1-2-5) Le Pré Diviseur des Timer/Compteur0 & 1

Le Timer/Compteur0 et le Timer/Compteur1 partage le même module de prédivision, mais les
Timer/Compteurs peuvent avoir des options différentes de prédiviseur.
Le Timer/Compteur2 à son propre pré diviseur à 10 bits



Architecture interne du Prédiviseur

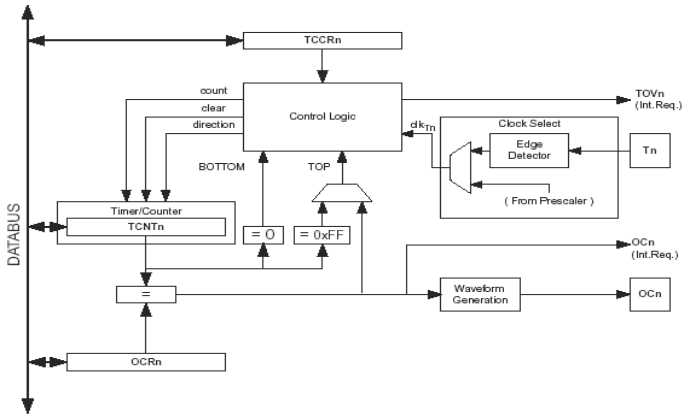
1-2-6) Les Timers/Compteur :

Un Timer/Compteur est un ensemble logique qui permet d'effectuer du comptage : de temps, d'évènements, de base de temps pour la génération de signaux...etc.
L'ATMega328 comme beaucoup des ATMEL, possède 3 Timers/counters :

a) Le Timer/Compteur0

Le timer/compteur est un compteur à 8 bits avec les particularités principales :

- Remise à 0 du compteur sur Comparaison (Rechargement Automatique)
- Générateur de Modulation de Phase en Largeur d'Impulsion (**PWM**)
- Générateur de Fréquence
- Compteur d'Événement Externe
- Horloge à 9 bits avec pré diviseur
- Interruption sur Débordent et Comparaison (**TOV0** et **OCF0**).



Architecture interne duTimer/Counter

Le Timer0 utilise 3 registres spécifiques et deux registres d'interruption.

- Registre TCCR0 (Timer/Counter Control Register)
- Registre TCNT0 (Timer/Counter Register)
- Registre TIMSK (Timer/Counter Interrupt Mask)

b) Le Timer/Compteur1 à 16 Bits (TIMER1)

L'unité de Timer/Compteur1 à 16 bits permet le cadencement précis des programmes, la gestion d'événement, la génération d'onde, etc. 15 modes sont disponibles avec le Timer et les particularités principales sont :

- Une Conception 16 bits totale
- Deux Unités de Comparaison Indépendante
- Double Comparateur
- Une Unité de Capture d'Entrée à faible Bruit
- Comparateur avec remise à 0 Automatique
- Générateur de Modulation de Phase en Largeur d'Impulsion Correct (**PWM**)
- Générateur d'onde **PWM** Périodique
- Générateur de Fréquence
- Compteur d'Événement Externe
- Quatre Sources Indépendant d'Interruption (**TOV1**, **OCF1A**, **OCF1B** et **ICF1**).

Le Timer1 utilise 5 registres spécifiques et deux registres d'interruption :

- Registre TCCR1 (Timer/Counter1 Controle Register1)
- Registre TCNT1H & TCNT1L (Timer/Counter1 Register)
- Registre OCR1H and OCR1L (Output Compare Register 1)
- Registre ICR1H and ICR1L (Input Compare Register 1)
- Registre TIMSK (Timer/Counter Interrupt Mask)

Le synoptique simplifié du Timer/Compteur1 à 16 bits avec les broches d'entrée-sortie.

c) Le Timer/Compteur2 à 8 Bits (Timer2)

Le Timer/Compteur2 à usage général à 8 bits avec les particularités principales :

- Simple Compteur à 8 bits
- Comparaison à Rechargement Automatique
- Générateur de Modulation de Phase en Largeur d'Impulsion Correct (**PWM**)
- Générateur de Fréquence
- Horloge à 10 bits de Pré diviseur
- Source d'Interruption sur Débordement et Comparaison (**TOV2 et OCF2**)
- Horloge Externe 32 768 **Hz** par Quartz Indépendant

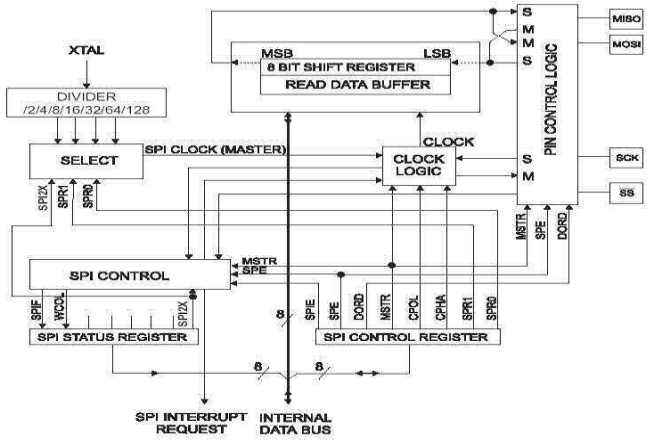
Le Timer2 utilise 4 registres spécifiques et deux registres d'interruption :

- Registre TCCR2 (Timer/Coounter2 Controle Register)
- Registre TCNT2 (Timer/Counter2 Register)
- Registre ASSR (Asynchronous Status Register)
- Registre TIMSK (Timer/Counter Interrupt Mask

1-2-7) L'interface Série Synchrone SPI

L'interface SPI est l'abréviation **Serial Peripheral Interfacel Synchronous**, soit Interface Série Synchrone. Contrairement à **l'UART** et comme son nom l'indique, ce type de périphérique génère les signaux d'horloge de synchronisation, arbitré par un Maître, c'est une interface série à 4 fils.

L'interface **SPI** permet le transfert de données ultrarapide synchrone entre l'**ATMega328** et des périphériques ou entre plusieurs dispositifs. L'interface **SPI** inclut les particularités suivantes :



Architecture interne de l'interface SPI

- Transfert de Données Full Duplex à Quatre Fils Synchrone
- Maître ou Esclave
- Transfert de Données avec **LSB** d'abord ou **MSB** en premier
- 7 Taux de Transfert Programmables
- Drapeau de Fin de Transmission pour Interruption
- Protection de Drapeau de Collision
- Mode Inoccupé
- **SPI** Mode Maître à Vitesse Double (**CK**/2)

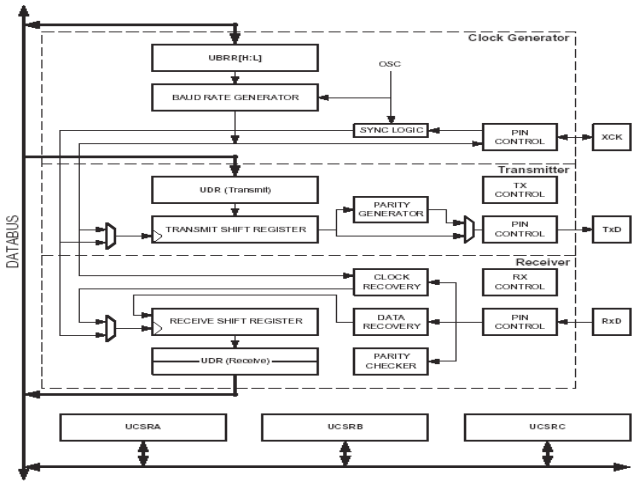
Ils sont au nombre de 3 et permettent de contrôler l'interface **SPI** :

- Registre SPCR (SPI Control Register)
- Registre SPSR (SPI Status Register)
- Registre SPDR (SPI Data Register)

1-2-8) L'interface Série USART

L'**USART** est l'abréviation de Universal Synchronous and Asynchronous Receiver and Transmitter, soit Interface Série Synchrone et Asynchrone avec les caractéristiques suivantes :

- Générateur interne de fréquence de cadencement
- Asynchrone et Synchrone opération
- Horloge Maître ou Esclave
- Echanges de données sur 5 à 9 bits et 1 ou 2 bits de stop
- Gestion des parités
- Communication en Full duplex (peut émettre et recevoir en même temps)
- Protection des débordements
- Détection de faux départs de transmission
- Filtrage de l'entrée
- Plusieurs interruptions programmables sur le mode émission et réception
- Mode double vitesse de communication



Architecture interne de l'interface USART

L'application principale de ce périphérique est la communication entre le microcontrôleur et un ordinateur via le port série **RS232**.

Les registres de l'**USART** sont aux nombres de trois avec une particularité, car deux registres occupent la même adresse physique, **UBRRH** partage le même emplacement d'entrée-sortie que le Registre **UCSRC.**

- Registre UCSR (USART Control and Status Register)
- Registre UBRR (USART Baud Rate Register)
- Registre UDR (USART I/O Data Register)

1-2-9) L'interface I2C (TWI)

L'interface à deux conducteurs périodique **TWI** est un dérivé de l'interface **I2C** de Philips, c'est une nouveauté du modèle **ATMEGA**, il n'existe pas dans les versions précédentes **AT89** et **AT90**. Le protocole **TWI** permet de connecter jusqu'à 111 systèmes différents employant seulement deux lignes de bus bidirectionnelles, un pour l'horloge **SCL** et un pour les données **SDA**. Le seul matériel externe nécessaire pour la mise en œuvre du bus est un simple jeu de résistance pour chacune des lignes (**R1**, **R2**).

Tous les systèmes connectés au bus ont des adresses individuelles et les mécanismes de control sont inhérents au protocole **I2C**. L'interface **TWI** utilise 5 registres pour contrôler et gérer totalement l'interface :

- Registre TWCR (TWI Control Register)
- Registre TWBR (TWI Bit Rate Register)
- Registre TWSR (TWI Status Register)
- Registre TWAR (TWI (Slave) Address Register)
- Registre TWDR (TWI Data Register)

1-2-10) Les Interruptions (INT)

La gestion des interruptions paraît souvent difficile et pourtant ce n'est pas plus compliquer que de gérer une interface **SPI** par exemple.

Afin d'éviter les boucles d'attente sans fin et les risques de blocage, il est assez simple de mettre en œuvre les interruptions du microcontrôleur qui dispose d'un grand nombre de possibilité d'activation.

En premier lever le drapeau d'autorisation d'interruption général dans **SREG** (instruction **SEI**) et il faut donner au périphérique concerné la possibilité d'interrompre provisoirement le programme principal en réglant les bits d'interruption dans les registres respectifs.

Chaque interruption va provoquer un saut de programme à une adresse bien précise correspondant à votre programme de gestion de l'interruption que vous avez écrit (voir le tableau décrit plus loin).
Utiliser l'instruction **RETI** pour effectuer la fin de l'interruption et provoqué le retour au programme principal.

Sauvegarder impérativement vos registres de travail, surtout **SREG**, dans la pile ou en mémoire, car dans le programme d'interruption, la valeur de ce registre à de très grande chance de changer et au retour de l'interruption vous pouvez avoir de drôle de surprise. Pour sauvegarder un registre, il faut le mettre sur la pile avec l'instruction **PUSH**, au début du programme d'interruption, puis le replacer à l'aide de l'instruction **POP** en fin de programme avant **RETI.**

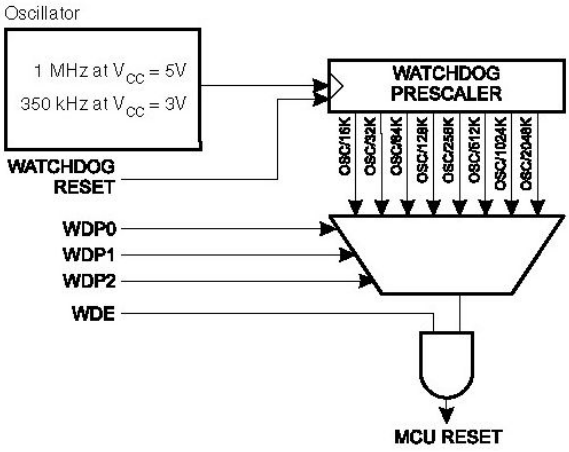
Si une seconde interruption intervient pendant le traitement de la première, le programme la traitera aussitôt fini la première (au retour sur **RETI**).

Quatre registres d'interruption existent dans l'ATMega328 :

- Registre GICR (General Interrupt Control Register):
- Registre GIFR (General Interrupt Flag Register)
- Registre TIFR (Timer/Counter Interrupt Flag)
- Registre TIMSK (Timer/Counter Interrupt Mask).

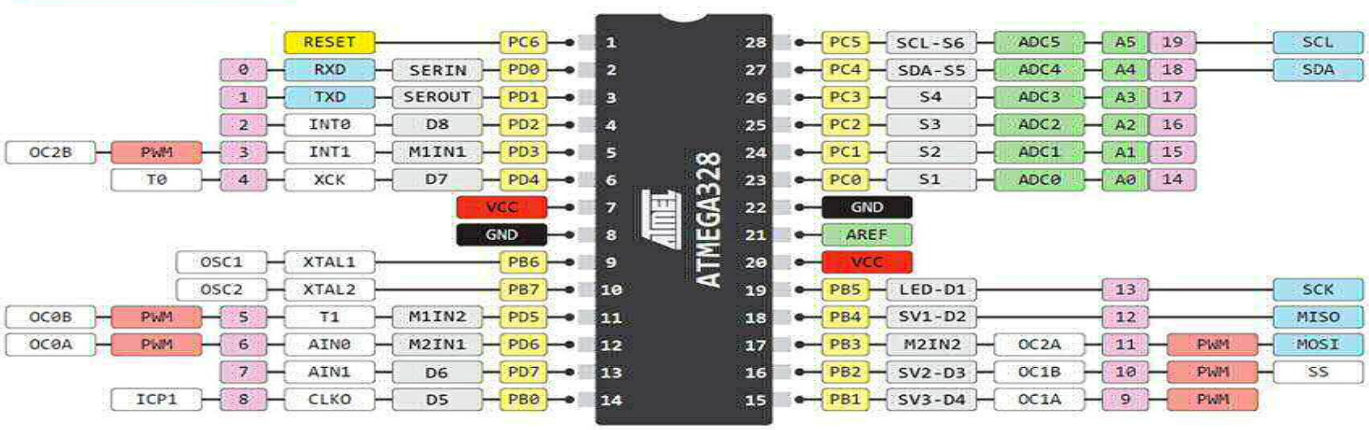
1-2-11) Le Chien de Garde ou Watchdog (WDT)

Le chien de garde ou Watchdog, est un compteur qui permet de palier au blocage du microcontrôleur. Ce blocage peut être d'ordre logiciel (retour impossible, mis en boucle infinie ou tout simplement erreur de structuration), soit matériel (parasites, chute de tensions) ; dans les deux cas, le blocage du programme peut avoir des conséquences très embêtantes.
Le Watchdog est un compteur dont la source d'horloge est indépendante du quartz du microcontrôleur, mais pas de l'alimentation. La fréquence de fonctionnement est de 1 **MHz** pour une tension d'alimentation de 5 V et 350 **KHz** pour 3 V.



Architecture interne du WatchDog

1) La structure externe de l'ATMega328 :



Structure Externe de l'ATMega328P

1-2-1) Description des pins :

- **PC, PB et PD** sont les PORT C, B, et D. Les PORTD et B sont configuré comme entrées digitales, et le PORTC est configuré comme entré analogique.
- **RESET-** Un niveau bas sur cette broche effectue un reset du microcontrôleur
- **RXD** et **TXD**: Entrée et Sortie des données de l'interface de communication USART
- **VCC, AVCC:** Alimentation positive du microcontrôleur et du convertisseur Analogique / Digital respectivement
- **GND** : Alimentation négative du microcontrôleur La masse.
- **T0, T1**: Source d'horloge externe du Compteur/Timer 0, 1
- **XCK** : Source d'horloge externe pour L'USART quand il est utilisé en mode synchrone
- **T1** : Source d'horloge externe du compteur/timer 1
- **AIN0, 1** : Entrée respectivement positive et négative du comparateur analogique.
- **INT0, 1, 2** : Entrée d'interruption externe N°0,1et 2.
- **XTAL1, 2** Entrée et Sortie d'horloge externe
- **T0SC1, 2** : broche de quartz pour horloge externe du Timer2.
- **INT0, 1**: Entrée d'interruption externe "0" et "1".
- **AIN0, 1** : Entrées positive et négative du comparateur analogique.
- **ICP1 :** Unité de capture pour le Timer/compteur1.
- **OC0, 1, 2** : Sortie lors d'une comparaison réussie pour le Compteur/Timer0, 1, 2, soit pour le mode PWM, il est suivi par A ou B qui désigne les registre OCR0A, B, les registre OCR1A, B et le registre OCR12, ou du mode PWM.
- **ADC5-0 :** Entrée du signal analogique du convertisseur Analogique / Digital, il possède six entrées disposée sur le PORTC.
- **AREF** Référence de tension externe pour le convertisseur Analogique / Digital.
- **SCK** : Sortie d'horloge de synchronisation en mode SPI maitre, entrée horloge en mode esclave.
- **MOSI** : Sortie des données en mode SPI maitre, entrée en mode esclave.
- **MISO** : Entrée des données en mode SPI maitre, sortie en mode esclave.
- **SS** : Entrée pour selection du mode esclave de la SPI.
- **SCL** : Signal d'horloge de l'interface de communication 2 fils (I2C)
- **SDA** : Signal de données de l'interface de communication 2 fils (I2C).

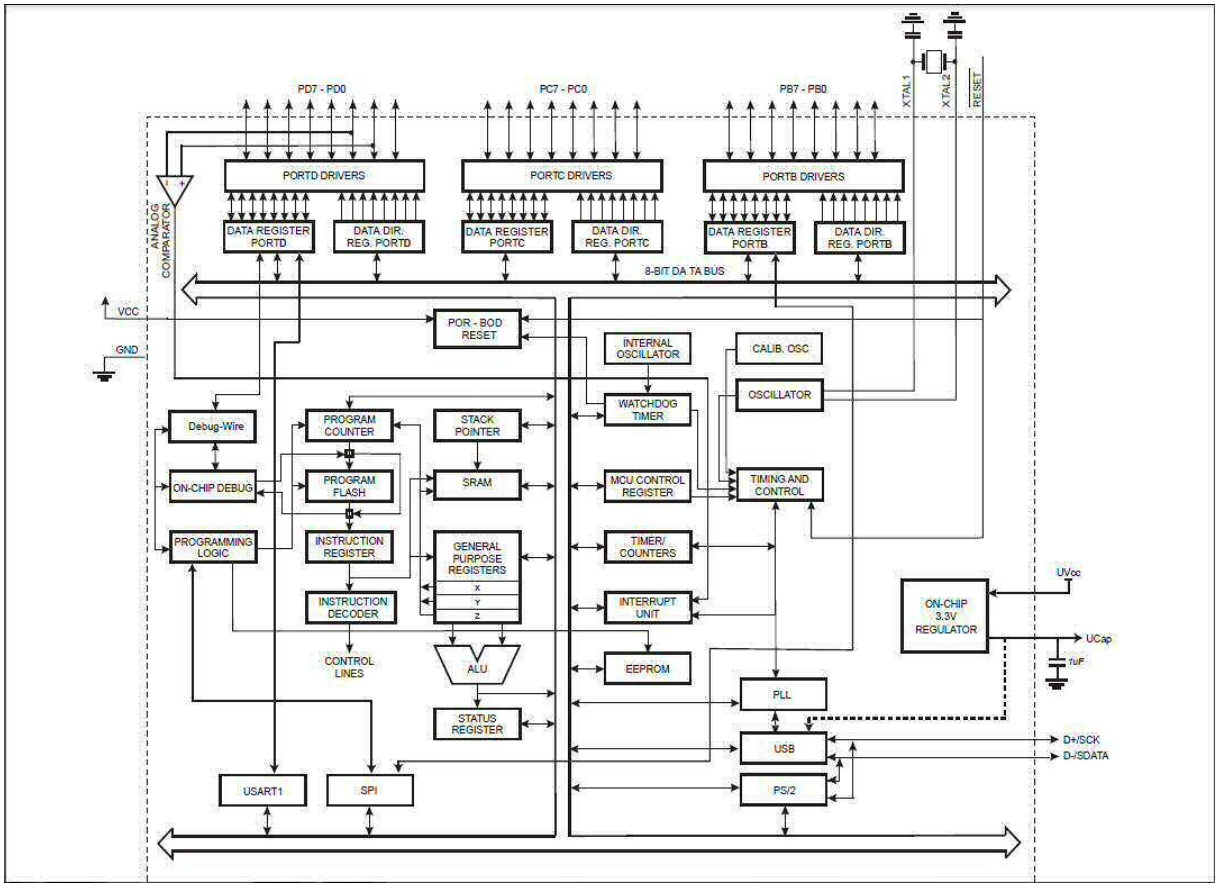
Annexe 2 : L'ATMega16U2

1) StructureATMega16U2 :

Le ATmega 16U2, comme les ATMega8U2 et 32U2, est un microcontrôleur 8 bits CMOS (un sigle anglais qui veut dire semi-conducteur en Oxyde métallique complémentaire) à faible puissance il est basée sur l'architecture AVR RISC amélioré.

En exécutant des instructions puissantes dans un seul cycle d'horloge, l'ATmega16U2 atteint des débits approchant le 1 MIPS par MHz permettant à un concepteur de système d'optimiser la consommation d'énergie par rapport à la vitesse de traitement.

1-1) Diagramme de l'ATMega16U2:

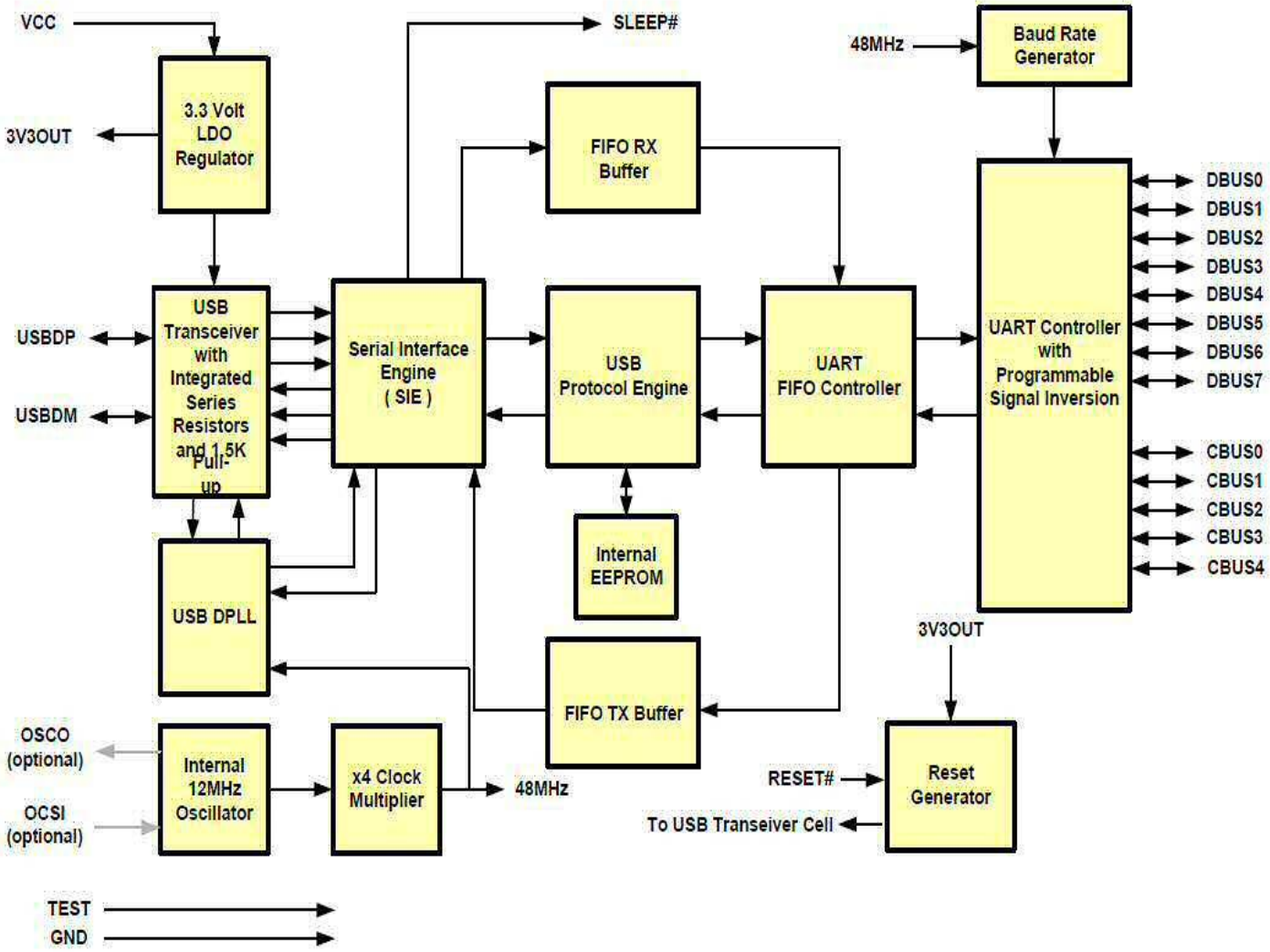


Architecture interne de l'ATMega16U2

Comme on peut le voir sur le diagramme ci-dessus, l'architecture de l'ATMega16U2 ressemble beaucoup à celle de l'ATMega328P. Effectivement leurs noyaux AVR combinent un riche ensemble d'instruction avec 32 registres de travail à usage général. Tous les registres sont directement connectés à l'unité arithmétique et logique (ALU), permettant à deux indépendants registres sont accessibles à une seule instruction exécutée en un seul cycle d'horloge,cela dit, il y a comme même des différences entre ces deux puces Atmel, premièrement l'ATMega328 possède des mémoires FLASH, EEPROM, SRAM plus importantes, des interfaces JTAG et ADC, un convertisseur A/N, et des PORTs munies de Buffers (amplificateur de courant de sortie), quant à l'ATMega16U2, il possède un régulateur de tension intégré qui délivre une tension de 3.3V à travers la pin UCAP, des pins D+ et D- pour la lecture et l'écriture des données en utilisant le registre UPOE, en gros ces pins permettent le transfert de données et un générateur de haute fréquence DLL.

Annexe 3 : La puce FT232RL

1) Architecture de la puce FT232RL



Architecture interne de la puce FT232RL

La puce FT232RL est une interface USB à UART Série de FTDI, elle permet une communication entre PIC ou AVR à un PC via le protocole USB 2.0, Muni d'une horloge interne de 12MHz et d'un régulateur de tension de 3.3V, elle facilite toute communication via l'USB de la carte Arduino.

2) Architecture Interne de la puce :

- **La mémoire EEPOM :** est utilisée pour stocker l'identité du fournisseur, aussi pour configurer les broches CBUS. La mémoire EEPROM est préprogrammé néanmoins, elle peut reprogrammée via l'USB grâce au logiciel MPROG, une partie de sa mémoire peut être programmable pour stocker des données **supplémentaires**.

- **Le régulateur LDO 3.3V :** qui fournit une tension de 3.3V pour commander les tampons de sortie de la cellule émetteur/récepteur USB. Il fournit également une tension de 3,3V à travers une résistance interne pull up de 1.5kΩ pour l'USBDP. La fonction principale de la LDO est d'alimenter le les cellules du générateur de réinitialisation plutôt que de la puissance logique externe Emission/réception USB. Cependant, il peut être utilisé pour alimenter un circuit externe nécessitant une alimentation nominale de 3,3 V à un courant maximal de 50 mA.

- **Cellule Emetteur/Récepteur USB :** fournit l'interface physique des USB1.1/USB2.0 full Speed, le pilote externe fournit également une tension de 3.3V pour le contrôle du débit de transmission.
- **USB DPLL :** Les cellules DPLL USB sont renfermées sur les données NRZI USB et régénère l'horloge et les signaux de données pour l'interface SIE .
- **Oscillateur interne12MHz** pour générer l'horloge de la puce
- **Prédiviseur :** pour manipuler la fréquence générée par l'oscillateur
- **Bloque SIE[1] :** se charge de la conversion parallèle/série et vice versa.
- **USB Protocol Engine :** dirige le flux de données reçus.
- **FIFORX Buffer** Les données envoyées depuis le contrôleur hôte USB vers le registre UART Via D- sont stockées dans le tampon FIFORX, puis redirigées du tampon vers le registre UART sous la supervision de tampon FIFO Controller.
- **FIFOTX Buffer :** Les données provenant du registre de réception UART sont stockées dans la mémoire tampon TX, le contrôleur hôte USB les supprime du tampon TX FIFO, en envoyant une demande de données depuis la commande de données IN.
- **UART FIFO Contrôleur** : se charge du transfert des données entre FIFORX et les tampons TX ainsi que le registre d'Emetteur/récepteur UART.

- **Controlleur UART avec Inversion des Signaux Programmables et High Drive** : Conjointement avec le contrôleur FIFO UART qui gère le transfert de données entre la mémoire FIFO RX et les tampons TX FIFO et le registre transmission/réception UART. Il effectue une conversion asynchrone de 7 ou 8 bits de parallèle en série et vice versa des données sur l'interface RS232 (ou RS422 ou RS485).

  Les signaux de commande pris en charge par le mode UART comprennent les signaux RTS, CTS, DSR, DTR, DCD et RI. Le contrôleur UART fournit également un émetteur permettant l'option de commande de signal de (TXDEN) pour aider à l'interfaçage avec les émetteurs-récepteurs RS485. Les options RTS / CTS, DSR / DTR et XON / XOFF sont également pris en charge. Handshaking (ou la poigné de main) est traitée dans le matériel pour assurer des temps de réponse rapides. L'interface UART prend également en charge le réglage et de détection des conditions RS232 BREAK.

  En outre, les signaux UART peuvent chacun être inversé individuellement et ont une capacité de force d'entraînement élevée configurable. Ces deux caractéristiques sont configurables dans l'EEPROM.
- **Bauds Générateur** - Le générateur de vitesse Baud fournit une entrée 16 fois l'horloge du contrôleur UART à partir de l'horloge de référence qui est de 48MHz. Il se compose d'un 14 bits prédiviseurs et 3 bits de registre qui fournissent un réglage fin de la vitesse de transmission (utilisé pour diviser par un nombre plus une fraction ou "sous-entier"). Ceci détermine le taux de l'UART, qui est programmable de 183 bauds à 3 Mbauds.Le FT232R supporte toutes les vitesses de transmission standard et les vitesses de transmission non-standard de 183 bauds jusqu'à 3 Mbauds. Les vitesses de transmission non standard suivent la règle **:**

  le Taux de Bauds = 3000000 / (n + x) Où **n** peut être un nombre entier quelconque compris entre 2 et 16384 (= 214) et **x** peut être un sous-entier de la valeur 0, 0,125, 0,25, 0,375, 0,5, 0,625, 0,75 ou 0,875. Lorsque n = 1, x = 0, i.e. les taux ayant des valeurs comprises entre 1 et 2 ne sont pas possibles.
- **RESET** : Cellule de réinitialisation intégrée fournit une tension au circuit interne de la puce au démarrage, la broche RESET permet à un périphérique externe de réinitialiser la puce FT232RL.

[1] *SIE* : Serial Interface Engine.

3) Description des broches

Pin	Nom	Type	Description
1	TX	Output	Transfert asynchrone des données
2	DTR	Output	Data Terminal Ready, contrôle des sorties de signaux de données
3	RTS	Output	Request To Send, demande d'envoi de signal.
4	VCCIO	PWR	alimentation de 1,8V à +5.25V aux broches de l'Interface UART et CBUS groupe (1 ... 3, 5, 6, 9 ... 14, 22 , 23)
5	RX	Input	Réception asynchrone des données
6	RI	Input	Ring Indicator, indicateur de cycle.
7, 18, 21	GND	PWR	La masse
8, 24	NC	NC	No internal Connection, Pas de connexion interne
9	DSR	Input	Data Set Ready, contrôle des configurations des données
10	DCD	Input	Data Carrier Detect, Détecteur de porteur de données
11	CTS	Input	Clear To Send, effacé pour envoyer
12, 13, 14, 22, 23	CBUS	I/O	Sortie/entrée configurable.
15, 16	USBD+/-	I/O	USB Data +/-, Bus de communication de données en format TTL
19	Reset	Input	Activation de la réinitialisation.
20	VCC	PWR	Source
25	AGND	PWR	Masse pour horloge interne.
26	Test	Input	Bascule la puce vers le mode Test.
27, 28	XTL1, 2	I, O	Broches d'E/S pour un oscillateur externe.

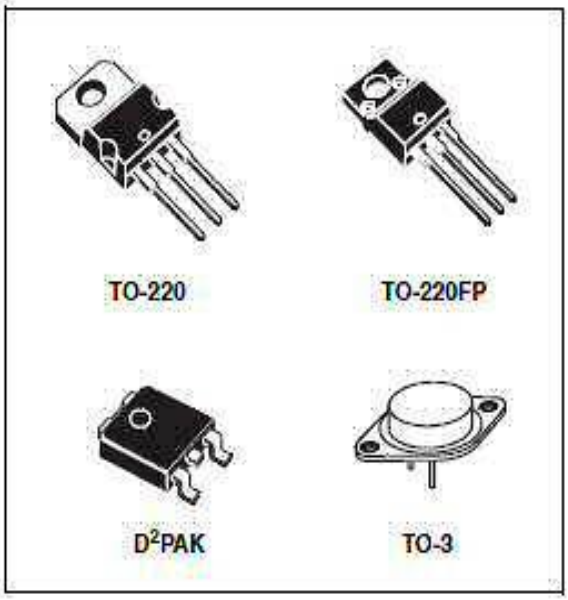
Annexe 4 : Régulateur de tension L7805

L7800 series

Positive voltage regulators

Feature summary

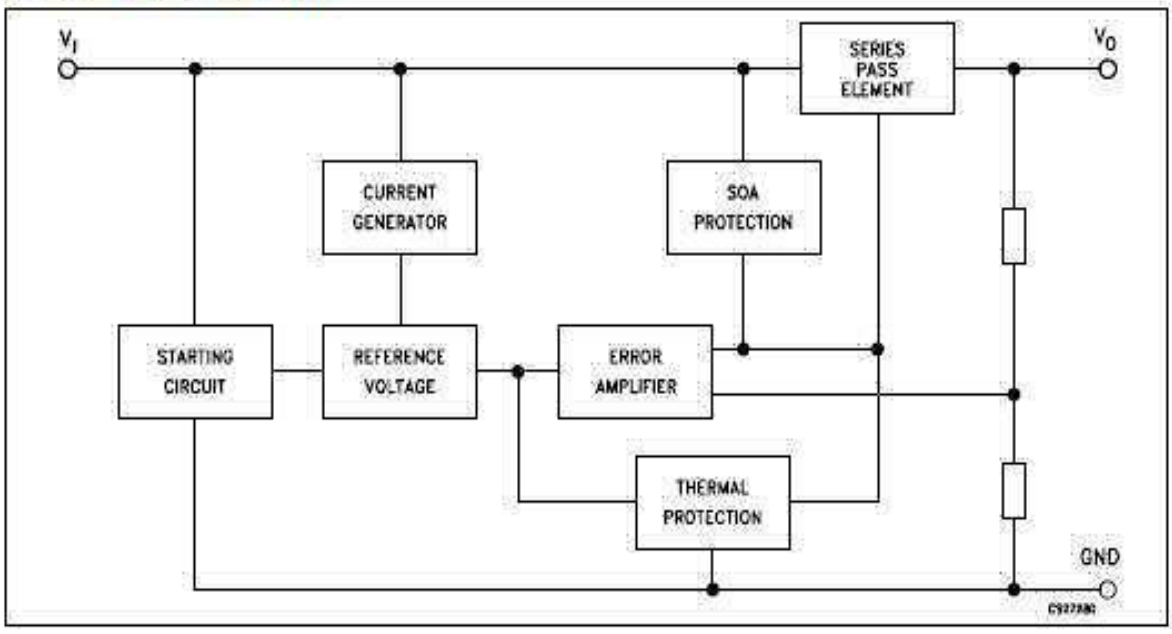
- Output current to 1.5A
- Output voltages of 5; 5.2; 6; 8; 8.5; 9; 10; 12; 15; 18; 24V
- Thermal overload protection
- Short circuit protection
- Output transition SOA protection



Description

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3 and D^2PAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

Schematic diagram



Datasheet du L7805CM

Annexe5 : Régulateur de tension AIC1084-33CM

aic

AIC1084

5A Low Dropout Positive Regulator

FEATURES

- Dropout Voltage 1.3V at 5A Output Current.
- Fast Transient Response.
- Extremely Tight Line and Load Regulation.
- Current Limiting and Thermal Protection.
- Adjustable Output Voltage or Fixed 1.5V, 1.8V, 2.5V, 3.3V.
- Standard 3-Pin Power Packages.

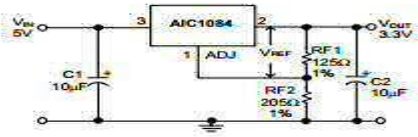
APPLICATIONS

- Mother Board I/O Power Supplies.
- Microprocessor Power Supplies.
- High Current Regulator.
- Post Regulator for Switching Supply.

DESCRIPTION

The AIC1084 is a low dropout three terminal regulator with 5A output current capability. The output voltage is adjustable with the use of a resistor divider or fixed 1.5V, 1.8V, 2.5V and 3.3V. Dropout voltage is guaranteed to be at maximum of 1.4V with the maximum output current. Its low dropout voltage and fast transient response make it ideal for low voltage microprocessor applications. Current limit and thermal protection provide protection against any overload condition that would create excessive junction temperatures.

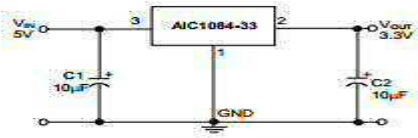
TYPICAL APPLICATION CIRCUIT



Adjustable Voltage Regulator

$V_{REF}=V_{OUT}-V_{ADJ}=1.25V$ (typ.)
$V_{OUT}=V_{REF} \times (1+RF2/RF1)+ I_{ADJ} \times RF2$
$I_{ADJ}=55\mu A$ (typ.)

(1) C1 needed if device is far away from filter capacitors.
(2) C2 required for stability.



Fixed Voltage Regulator

aic

AIC1084

ORDERING INFORMATION

AIC1084-XXXXXX

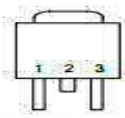
- PACKING TYPE
  - TB: TUBE
  - TR: TAPING & REEL
- PACKAGING TYPE
  - E: TO-252
  - M: TO-263
  - T: TO-220
- C: COMMERCIAL
  P: LEAD FREE COMMERCIAL
- OUTPUT VOLTAGE
  DEFAULT: ADJUSTABLE
  - 15: 1.5V
  - 18: 1.8V
  - 25: 2.5V
  - 33: 3.3V

Example: AIC1084-15CETR
→ 1.5V version in TO-252 Package & Taping & Reel Packing Type

Example: AIC1084-15PMTR
→ 1.5V version in TO-263 Lead Free Package & Taping & Reel Packing Type

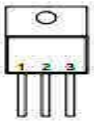
PIN CONFIGURATION

TO-252 (CE&PE)
TOP VIEW
1: ADJ (GND)
2: VOUT (TAB)
3: VIN



TO-263 (CM&PM)
TOP VIEW
1: ADJ (GND)
2: VOUT (TAB)
3: VIN

TO-220 (CT&PT)
FRONT VIEW
1: ADJ (GND)
2: VOUT (TAB)
3: VIN



ABSOLUTE MAXIMUM RATINGS

VIN pin to ADJ/GND pin		7V
Operating Temperature Range		–40°C to 85°C
Maximum Junction Temperature		125°C
Storage Temperature Range		- 65°C ~ 150°C
Lead Temperature (Soldering) 10 sec.		260°C
Thermal Resistance Junction to Case	TO-252	12.5°C/W
	TO-263, TO-220	3°C /W
Thermal Resistance Junction to Ambient	TO-252	100°C/W
(Assume no ambient airflow, no heatsink)	TO-263	60°C /W
	TO-220	50°C /W

Absolute Maximum Ratings are those values beyond which the life of a device may be impaired.

Datasheet du AIC1084-33CM

Annexe 6 : Amplificateur Opérationnel LM358

National Semiconductor

October 2005

LM158/LM258/LM358/LM2904
Low Power Dual Operational Amplifiers

General Description

The LM158 series consists of two independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

Application areas include transducer amplifiers, dc gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM158 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional ±15V power supplies.

The LM358 and LM2904 are available in a chip sized package (8-Bump micro SMD) using National's micro SMD package technology.

Unique Characteristics

- In the linear mode the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage.
- The unity gain cross frequency is temperature compensated.
- The input bias current is also temperature compensated.

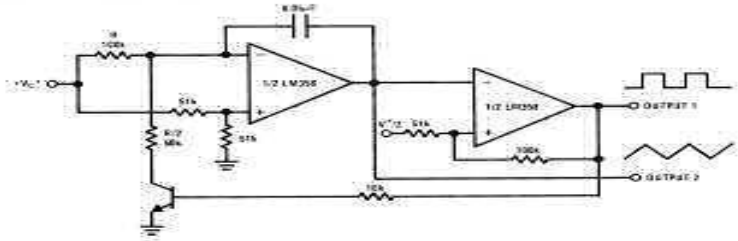
Advantages

- Two internally compensated op amps
- Eliminates need for dual supplies
- Allows direct sensing near GND and V_{OUT} also goes to GND
- Compatible with all forms of logic
- Power drain suitable for battery operation

Features

- Available in 8-Bump micro SMD chip sized package, (See AN-1112)
- Internally frequency compensated for unity gain
- Large dc voltage gain: 100 dB
- Wide bandwidth (unity gain): 1 MHz (temperature compensated)
- Wide power supply range:
  — Single supply: 3V to 32V
  — or dual supplies: ±1.5V to ±16V
- Very low supply current drain (500 µA)—essentially independent of supply voltage
- Low input offset voltage: 2 mV
- Input common-mode voltage range includes ground
- Differential input voltage range equal to the power supply voltage
- Large output voltage swing

Voltage Controlled Oscillator (VCO)



LM158/LM258/LM358/LM2904

Absolute Maximum Ratings (Note 9)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

	LM158/LM258/LM358 LM158A/LM258A/LM358A	LM2904
Supply Voltage, V+	32V	26V
Differential Input Voltage	32V	26V
Input Voltage	−0.3V to +32V	−0.3V to +26V
Power Dissipation (Note 1)		
Molded DIP	830 mW	830 mW
Metal Can	550 mW	
Small Outline Package (M)	530 mW	530 mW
micro SMD	435mW	
Output Short-Circuit to GND		
(One Amplifier) (Note 2)		
V+ ≤ 15V and T_A = 25°C	Continuous	Continuous
Input Current (V_{IN} < −0.3V) (Note 3)	50 mA	50 mA
Operating Temperature Range		
LM358	0°C to +70°C	−40°C to +85°C
LM258	−25°C to +85°C	
LM158	−55°C to +125°C	
Storage Temperature Range	−65°C to +150°C	−65°C to +150°C
Lead Temperature, DIP		
(Soldering, 10 seconds)	260°C	260°C
Lead Temperature, Metal Can		
(Soldering, 10 seconds)	300°C	300°C
Soldering Information		
Dual-In-Line Package		
Soldering (10 seconds)	260°C	260°C
Small Outline Package		
Vapor Phase (60 seconds)	215°C	215°C
Infrared (15 seconds)	220°C	220°C
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.		
ESD Tolerance (Note 10)	250V	250V

Datasheet du LM358

Annexe 7 : Transistor à effet de champ IRF9640

FAIRCHILD
SEMICONDUCTOR®

IRF9640, RF1S9640SM

Data Sheet **January 2002**

11A, 200V, 0.500 Ohm, P-Channel Power MOSFETs

These are P-Channel enhancement mode silicon-gate power field-effect transistors. They are advanced power MOSFETs designed, tested, and guaranteed to withstand a specified level of energy in the breakdown avalanche mode of operation. All of these power MOSFETs are designed for applications such as switching regulators, switching converters, motor drivers, relay drivers and as drivers for other high-power switching devices. The high input impedance allows these types to be operated directly from integrated circuits.

Formerly developmental type TA17522.

Features

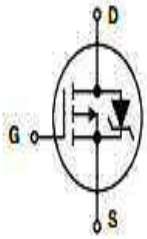
- 11A, 200V
- $r_{DS(ON)} = 0.500\Omega$
- Single Pulse Avalanche Energy Rated
- SOA is Power Dissipation Limited
- Nanosecond Switching Speeds
- Linear Transfer Characteristics
- High Input Impedance
- Related Literature
  - TB334, "Guidelines for Soldering Surface Mount Components to PC Boards"

Ordering Information

PART NUMBER	PACKAGE	BRAND
IRF9640	TO-220AB	IRF9640
RF1S9640SM	TO-263AB	RF1S9640

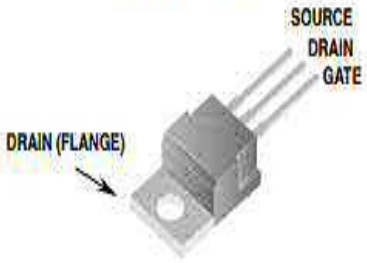
NOTE: When ordering, use the entire part number. Add the suffix 9A to obtain the TO-263AB variant in the tape and reel, i.e., RF1S9640SM9A.

Symbol

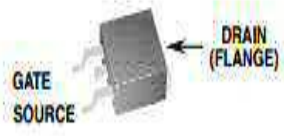


Packaging

JEDEC TO-220AB



JEDEC TO-263AB



Datasheet del'IRF9640

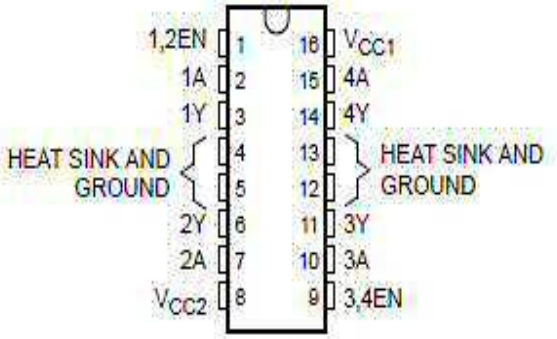
Annexe 8 : Pont H L293D

L293, L293D
QUADRUPLE HALF-H DRIVERS

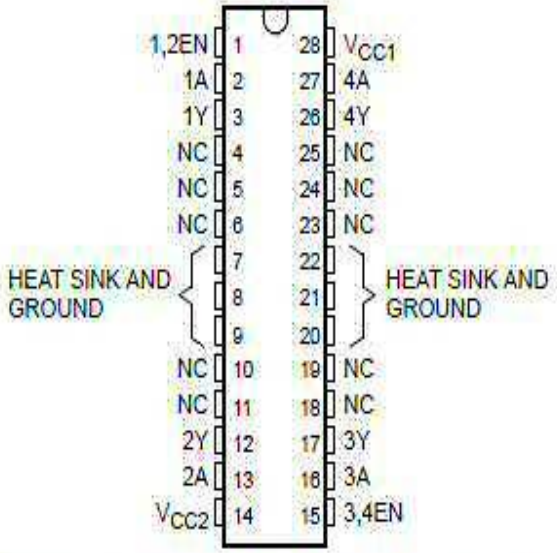
SLRS008B – SEPTEMBER 1986 – REVISED JUNE 2002

- Featuring Unitrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functional Replacements for SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

N, NE PACKAGE
(TOP VIEW)



DWP PACKAGE
(TOP VIEW)



description

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation.

The L293and L293D are characterized for operation from 0°C to 70°C.

Datasheet du L293D

Références

[1] http://fr.flossmanuals.net/arduino/historique-du-projet-arduino/, Mars 2016.

[2] https://www.parallax.com/catalog/microcontrollers/basic-stamp, Mars2016.

[3] https://www.parallax.com, Février 2016.

[4] Livre ARDUINO, publié en ligne en 2011, écrit par : téléchargé depuis le lien www.flossmanualsfr.net/_booki/arduino/arduino.pdf, Avril 2016.

[5] https://www.processing.org, Mai 2016.

[6] Wiring.org.com, Avril 2016.

[7] Creativecommons.org., Mars 2016.

[8] Magazine SHY, Entrevue avec Massimo Banzi fondateur d'Arduino_20150609, Avril 2016.

[9] www.atmel.com, Avril 2016.

[10] Le making of d'Arduino ou la fabuleuse Histoire d'une carte électronique, de David Kushner, magazine Spectrum, revue d'octobre 2011, Février 2016.

[11] www.arduino.cc, Mars 2016.

[12] www.gnu.org/gnu/linux-and-gnu.fr.html, Avril 2016.

[13] www.wikipédia.com, Février 2016.

[14]Forum de l'association WDA, Association de la préservation du patrimoine numérique, 5 Mars 2016.

[15] http://www.elektronique.fr/logiciels/proteus.php, Mars 2016.

[16] http://www.adrirobot.it/arduino/arduino_UNO/scheda_arduino_UNO.htm , Mars2016.

[17] http://datasheet.octopart.com/L7805CV-STMicroelectronics-datasheet-7264666.pdf, Avril 2016.

[18] http://pdf.datasheetcatalog.com/datasheet2/d/0jaogtcf1zujrfao71a5ce37qhky.pdf, Mars 2016.

[19] https://www.sparkfun.com/datasheets/Components/General/LM358.pdf, Avril 2016.

[20] http://pdf.datasheetcatalog.com/datasheet/fairchild/RF1S9640SM.pdf, Mars 2016.

[21] https://www.atheros-drivers.com/, Avril 2016.

Résumé

Ce projet nous permet de plonger dans l'univers Arduino dans le but de réaliser la célèbre carte électronique et de manipuler son IDE.

L'objectif préliminaire est de connaitre l'environnement de la carte : son parcours, ses modèles, ses utilisations, l'objectif principale est l'étude de la carte, et cela en assimilant son principe de fonctionnement, et savoir de quoi elle est constituée et comment la réaliser et l'utiliser pour prototyper des objets.

Abstract

This project immerses us in the Arduino's Universe, with one purpose, make the famous electronic card and manipulate its IDE.

The preliminary objective is to know the card's environment, its history, its models, its utilizations domains, the main objective is to assimilate the running basics, and know with what it's constituted, how to make and how to use it, for prototyping stuff.

.

ملخص

ذا الشروع مكننا من الخول الي ع الم ا ردوينو و ذالك من اجل إنشاء بطاقة الكترونية و التحكم في برمجتها ه

ملهدف الأولي لهذا المشروع هو: معرفة محيط الطاقة ا لكترونية ر لحل تقدمها وظور امرور اعلي أنواعها و مجلات , استخدامها .

الهدف الرئيسي هو إدراك مبدأ سيرها ,مكوناتها , كيفية إنشائها و أخيرا استعمالها في نماذج تطبيقية و في الأخير التحكم في المنتوج.