الجمهـــوريـــــة الجزائـــريـــــة الديمقـــراطيـــــة الشـعبيـــة
République Algérienne Démocratique et Populaire
وزارة التـــعـلـيـــــم العـــــالـــــي والبـــحـــــث العـلمـــــي
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

جامعة وهران 2 محمد بن أ حمد
Université d'Oran 2 Mohamed Ben Ahmed

معهد الصيانة و الأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle
Département Maintenance en instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie industriel
Spécialité : Ingénierie de la Maintenance en instrumentation

Réalisation d'une interface PC « Programmateur AVR »

Présenté et soutenu publiquement par :

Nom : BELBEKHOUCHE Prénom : Karima

Nom : ZIANE Prénom : Sarra

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
AOUIMER Yamina	MAA	IMSI-Univ. D'Oran2	**Présidente**
HASSINI Abdelatif	PR	IMSI-Univ. D'Oran2	**Encadreur**
MEKKI Ibrahim el khalil	MCA	IMSI-Univ. D'Oran2	**Examinateur**

Année 2019/2020

Remerciement

Remerciement Nous tenons tout d'abord à remercier Allah le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

En second lieu, nous tenons remercier très chaleureusement à Mr. HASSINI ABDELATIF, qui nous a permis de bénéficier de son encadrement. Les conseils qu'il nous a prodigué, la patience, la confiance qu'il nous a témoignés ont été déterminants dans la réalisation de notre travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail, et de l'enrichir par leurs propositions.

Nous souhaitant adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire.

Un clin d'œil à nos familles et nos amis qui par leurs prières et leurs encouragements, on a pu surmonter tous les obstacles.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicaces

Toutes les lettres ne sauraient trouver les mots qu'il faut...

Tous les mots ne sauraient exprimer la gratitude,

L'amour, le respect, la reconnaissance...

Aussi, c'est tout simplement que Je dédie ce modeste travail...

A ma très chère mère

Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon bien être.

Je vous remercie pour tout le soutient et l'amour que vous me portez

Que ces modeste travail soit l'exaucement de vos vœux, le fruit de vos innombrables sacrifices. Puisse mon dieux le très haut vous accorder une bonne santé, bonheur et langue vie.

A mon très cher père

Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour toi. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Puisse Dieu, le tout puissant, te préserver et t'accorder santé, longue vie et bonheur.

A mon petit prince Haithem

Mon cher petit frère présent dans tous mes moments d'examens par son soutien moral et ses belles surprises sucrées. Je te souhaite un avenir plein de joie, de bonheur, de réussite et de sérénité. Je t'exprime à travers ce travail mes sentiments de fraternité et d'amour.

A ma chère sœur Zora

A ma deuxième ma maman Kheira, et mon oncle Mohammed

A ma chère tante et sœur Djamila, et mon oncle Abdelrahmane

A mes anges Hiba, RAwnek, Narimene, Nessrine et Chourouk

A toutes la famille Belbekhouche et Bouzianne el rahmani

A mes chers collègues

Je dédie ce travail a tous mes amis Ayoub, Nessrine, Romaissa, Sara, Asmaa, Nacera, Bouchra, Houda et Sara ceux avec qui j'ai passe mes 5 ans a l'IMSI. Mes collègues, C'était la joie pour moi de se regrouper ensemble, de discuter, de réviser ensemble et de partager tous les moments de joie et de tristesse.

En témoignage de l'amitié qui nous unie, des souvenirs et de moments que nous avons passés ensemble je vous dédie ce travail

Je vous aime tous et je vous souhaite une carrière plein de réussite et d'excellence.

A tous ceux qui m'ont aidé, de près ou de loin, même qu'il soit

Un mot d'encouragement et de gentillesse.

Belbekhouche Karima

Dédicaces

Je dédie cet évènement marquant de ma vie à la mémoire de ***mon père*** *disparu trop tôt, j'espère que, du monde qui est sien maintenant, il apprécie cet humble geste comme preuve de reconnaissance de la part d'une fille qui a toujours prié pour le salut de son âme .puisse dieu, le tout puissant, l'avoir en sainte miséricorde.*

A ***ma mère*** *le symbole de la bonté par excellence, la source de tendresse l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.*

A frères mes ***MOHAMMED*** *et* ***ABDELLATIF*** *qui m'avez toujours soutenu et encouragé durant ces années d'étude.*

A la femme de mon frère ***FATOUM*** *et mes princes* ***AYOUB*** *et* ***WAEL.***

Une pensée très spéciale envers ma sœur ***SARRAH*** *sa présence à mes côtés m'a toujours donné l'impression d'être proche de toute ma famille.*

A mon binôme ***KARIMA*** *qui m'accompagné durant cette période de travail, grâce à toi, j'ai beaucoup apprécié ton implication ton soucis et ton partage.*

ZIANE SARRA

Tables des matières

Chapitre I : Généralités sur l'interface PC /AVR

Chapitre II : Etude théorique sur l'AVR Atmega 238

Chapitre III : Description des parties logicielle et matérielle

Chapitre IV : Réalisation et tests

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Liste des acronymes

ADC : convertisseurs analogiques-numériques

ALU : L'unité arithmétique et logique

AREF : est l'entrée de référence analogue pour le Convertisseur A/D

ASCII : American Standard Code for Information Interchange

Atmel AVR : une famille de microcontrôleurs

AVCC : est une broche de tension d'alimentation pour le Convertisseur A/D

Clk : L'horloge

CTC : effacer le compteur de minuterie (Clear Timer Counter)

CPU : Un processeur (ou unité centrale de traitement)

CS : source d'horloge (clock source)

DPRAM : la mémoire vive dynamique

DDRx : registre de direction

ICR : registre d'entrée de capture (Input Capture Register)

GND : Masse de l'alimentation

MIPS : Million d'instructions par second

OCR : registre de comparaison en sortie (Output Compare Register)

PORTx : registre de données

PIND : Registre de lecture des données

PWM : Un signal à modulation de largeur d'impulsion

RAM : la mémoire vive

RISC : Architecture d'un processeur à jeu d'instructions réduit

ROM : la mémoire morte

SPI : Interface périphérique série

SRAM : la mémoire vive statique

SSI : Small squale Integration

MSI : Medium squale Integration

VLSI : Very Large Squale Integration

LSI : Large squale Integration

EPP : Enchanced Parralel Port

ECP : Enchanced Capabilities Port

ISP : In system programming

IPD : Programme et Interface de Débogage

PWM : Pulse Width Modulation

TWI : Two Wire Interface

MISO: Master In Slave Out

MOSI: Master Out Slave In

SCK: serial Clock

FTDI : Future Technnology Devices International

USB : Universal Serial Bus

PCB :Printed Circuit Board

Résumé

Jusqu'à présent, l'apprentissage de la logique se faisait à travers la découverte des fonctions logiques élémentaires contenues dans les circuits intégrés des familles 74xxx. Les expérimentations se limitaient aux fonctions proposées par les fabricants de ces circuits. La conception de fonctions logiques regroupant plusieurs de ces circuits nécessitait un câblage conséquent, et la réalisation d'un circuit imprimé de grande surface. L'apparition des circuits logiques programmables a permis de s'affranchir de cette limitation. En effet, l'utilisateur peut créer, dans ces circuits, toutes les fonctions logiques qu'il souhaite avec comme seules limitations, la place disponible dans le circuit choisi et/ou la vitesse de fonctionnement de celui-ci. La taille actuelle de ces circuits permet l'intégration d'un système à processeur complet. Ainsi, le but de ce rapport, est dans un premier temps de présenter les circuits logiques programmables. L'objectif principal est de maîtriser les méthodes de programmation d'un microcontrôleur. Simultanément les différents types de mémoires seront exposés. Toute les méthodes d'écriture dans une mémoire morte (EEPROM et Flash) d'un microcontrôleur de la famille AVR seront expliquées puis décrites en assembleur.

Dans un deuxième temps, l'accent sera mis sur les outils d'interfaçage entre l'AVR et un ordinateur. Cette section nous permettra ainsi de concevoir le schéma du programmateur. L'édition du schéma électrique sous Proteus, du circuit imprimé et de l'image en 3D de la carte sous ARES ainsi que la simulation, puis l'expérimentation sur banc d'essais et la réalisation sur bakélite viendront couronner le travail.

Abstract

Until now, the learning of logic was done through the discovery of the elementary logic functions contained in the integrated circuits of the 74xxx families. The experiments were limited to the functions offered by the manufacturers of these circuits. The design of logic functions grouping together several of these circuits required a consequent wiring, and the realization of a printed circuit of large surface. The appearance of programmable logic circuits has made it possible to overcome this limitation. Indeed, the user can create, in these circuits, all the logic functions which he wishes with as only limitations, the space available in the chosen circuit and / or the operating speed of the latter. The current size of these circuits allows the integration of a complete processor system. Thus, the aim of this report is first of all to present programmable logic circuits. The main objective is to master the methods of programming a microcontroller. At the same time, the different types of memories will be exposed. All the methods of writing in a read only memory (EEPROM and Flash) of a microcontroller of the AVR family will be explained and then described in assembler. Secondly, the focus will be on tools for interfacing between the AVR and a computer. This section will allow us to design the diagram of the programmer.

The editing of the electrical diagram under Proteus, the printed circuit and the 3D image of the card under ARES as well as the simulation, then the experiment on a test bench and the realization on Bakelite will come to crown the work.

Introduction général

1. Généralité

Aujourd'hui, l'électronique est de plus en plus remplacée par l'électronique programmée numérique qui présente de nombreux avantages sur l'analogique : grande insensibilité aux parasites et aux dérives diverses, modularité et reconfigurabilité, facilité de stockage de l'information. On parle aussi de système embarquée ou d'informatique embarquée. Son but est de simplifier les schémas électroniques et par conséquent réduire l'utilisation de composants électroniques, réduisant ainsi le coût de fabrication d'un produit. Il en résulte des systèmes plus complexes et performants pour un espace réduit.

L'évolution des systèmes électroniques amène de plus en plus souvent les concepteurs à remplacer l'électronique câblée à base de nombreux circuits intégrés par un circuit programmable qui remplit à lui seul toutes les fonctions. Les microcontrôleurs appartiennent à cette famille de circuits.

Les systèmes embarqués utilisent un ou plusieurs processeurs ou microcontrôleurs pour exécuter des opérations spécialisées dans un système plus complexe. Ces contrôleurs embarqués doivent communiquer avec d'autres composants, capteurs et même contrôleurs du système. Bien qu'ils soient courants, les protocoles et interfaces série complexes peuvent être très difficiles à programmer et à dépanner, surtout si le nombre de dispositifs avec lesquels ils communiquent est faible.

Les concepteurs ont besoin de microcontrôleurs, de périphériques et de capteurs dotés d'une interface numérique dispositif-à-dispositif simple, pouvant gérer des données de longueur arbitraire à haute vitesse et éliminer les tâches de programmation orientée protocole complexes.

Position du problème

Dans la vie moderne, on utilise beaucoup d'outils et d'accessoires de commande à distance afin de simplifier notre contrôle, donc nous chercherons toujours à se concentrer sur la souplesse de la commande et de contrôler sur une zone bien définie (notre environnement) le plus grand nombre possible d'accessoires. L'ordinateur occupe la première place d'objets qui ne nous quittent pas donc notre travail se concentre sur l'utilisation de ce dernier avec un système ou une carte de commande à partir de sa liaison qui s'appelle carte d'interfaçage.

2. Objectif du projet

Dans ce projet trois objectifs ont été visés :

Le premier objectif est de regrouper suffisamment d'informations sur catégorie de cartes d'interfaçage: son langage de programmation, sa construction, son principe de fonctionnement. Etudier brièvement le fonctionnement de la carte.

Le deuxième objectif est de trouver le montage le plus convenable et performant pour la programmation des microcontrôleurs AVR.

Le troisième objectif consiste à simuler et réaliser une carte électrique capable d'exécuter une action entre un ordinateur et une carte électronique en expliquant les différents bloquent de sa construction. Expérimenter la carte avec quelques exemples d'applications.

4-Présentation du mémoire :

Le premier chapitre sera consacré à l'historique des interfaces PC, puis, on mettra la lumière sur un modèle de base qui est « une interface PC programmateur AVR ». On parlera de microcontrôleurs AVR, ses déférentes familles, ses spécifiques applications et ses types.

Le deuxième chapitre sera consacré à l'étude d'AVR, ses caractéristiques, sa construction et leur brochage. Cette étude nous permettra de déterminer le montage à réaliser.

Dans le troisième chapitre, on présentera le hardware et le software du programmateur AVR.

Et dans le quatrième chapitre, on réalisera le circuit dans la réalité plus un exemple d'utilisation.

Enfin, on terminera avec une conclusion générale qui résumera l'intérêt de notre étude.

Chapitre I : Généralités sur l'interface PC /AVR

Introduction

Les fabricants de circuits intégrés numériques s'attachent-ils à fournir des circuits présentant des densités d'intégration toujours plus élevée, pour des vitesses de fonctionnement de plus en plus grandes.

D'abord réalisées avec des circuits SSI (Small Scale Intégration), les fonctions logiques intégrées se sont développées avec la mise au point du transistor MOS dont la facilité d'intégration a permis la réalisation de circuits MSI (Medium Scale Integration) puis LSI (Large Scale Integration) puis VLSI (Very Large Scale Integration).

L'intégration des principales fonctions numériques d'une carte au sein d'un même boîtier permet de répondre à la fois aux critères de densité et de rapidité (les capacités parasites étant plus faibles, la vitesse de fonctionnement peut augmenter). La plupart de ces circuits sont maintenant programmés à partir d'un simple ordinateur type PC directement sur la carte où ils vont être utilisés. En cas d'erreur, ils sont reprogrammables électriquement sans avoir à extraire le composant de son environnement.

Pour programmer nous avons besoin d'un matériel : un PC qui contient un logiciel de programmation puis une connexion avec un programmateur qui assure la relation entre l'ordinateur et le dispositif à programmer.

Maintenant, on va voir qu'est-ce qu'une interface ?

I.1. L'interface PC :

Une interface définit la frontière de communication entre deux entités, comme des éléments de logiciel, des composants de matériel informatique, ou un utilisateur. Elle se réfère généralement à une image abstraite qu'une entité fournit d'elle-même à l'extérieur. Cela permet de distinguer les méthodes de communication avec l'extérieur et les opérations internes, et autorise à modifier les opérations internes sans affecter la façon dont les entités externes interagissent avec elle, en même temps qu'elle en fournit des abstractions multiples. On appelle aussi interfaces des dispositifs fournissant un moyen de traduction entre des entités qui n'utilisent pas le même langage, comme entre l'Homme et un ordinateur. Étant donné que

ces interfaces réalisent des traductions et des adaptations, elles entraînent des coûts de développement supplémentaires par rapport à des communications directes. [1]

La figure I.1 représente la liaison entre l'interface de programmation et le circuit à programmer

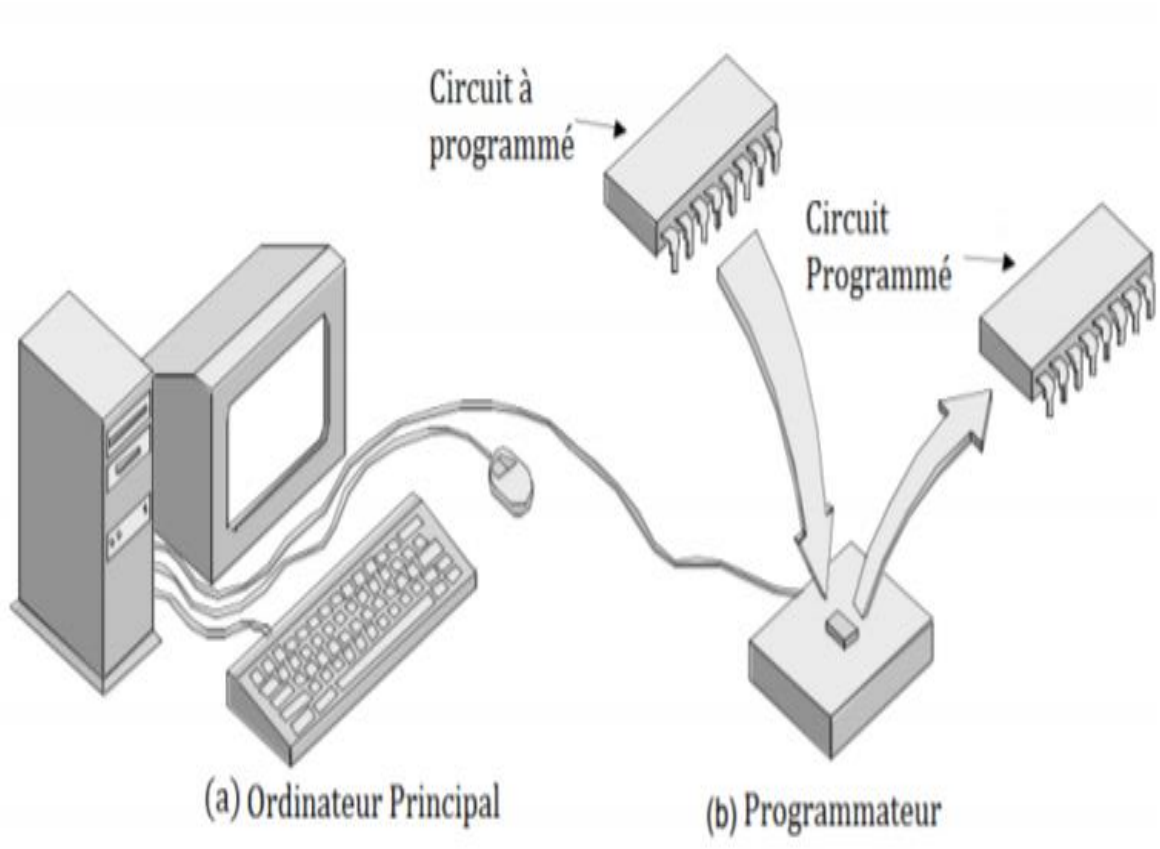


Figure I.1. Interface connectée à un PC et une carte à programmer

I.1.1. Les types des interfaces :

En informatique et en électronique, une interface est un dispositif qui permet des échanges et interactions entre différents acteurs :

Une interface Homme-machine : permet des échanges entre un humain et une machine.

Une interface de programmation : permet des échanges entre plusieurs logiciels.

Une interface, dans certains langages : objet (Java, C++…), est la déclaration de signature(s) d'une fonction que toutes les classes héritantes devront dûment implémenter**.** [1]

I.1.2. Port série et parallèle :

Port (dérivé du mot latin "porta" est une interface physique qui connecte un ordinateur à d'autres ordinateurs ou périphériques d'entrée / sortie matériels. Basé sur le transfert de signal, les ports sont divisés en trois groupes en tant que ports série, parallèle et USB.

a. La communication en série :

Les ports série (également appelés **RS-232**, nom de la norme à laquelle ils font référence) représentent les premières interfaces ayant permis aux ordinateurs d'échanger des informations avec le "monde extérieur". Le terme série désigne un envoi de données via un fil unique : les bits sont envoyés les uns à la suite des autres comme indiqué dans la figure I.3

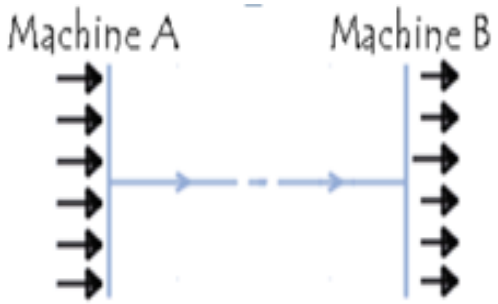


Figure I.2.Liaison série

A l'origine les ports série permettaient uniquement d'envoyer des données, mais pas d'en recevoir (voir la figure I.3), c'est pourquoi des ports bidirectionnels ont été mis au point (ceux qui équipent les ordinateurs actuels le sont); les ports séries bidirectionnels ont donc besoin de deux fils pour effectuer la communication.

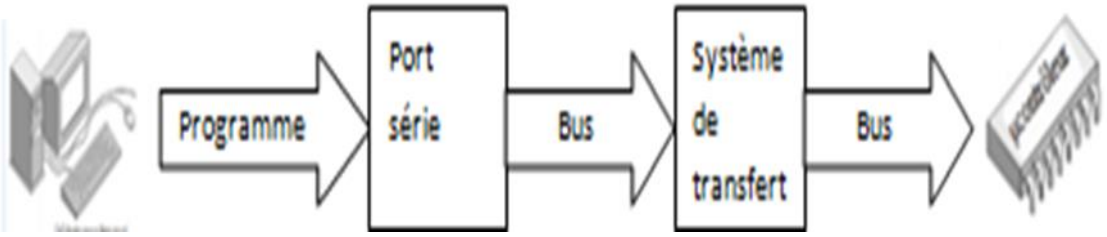


Figure I.3. Étapes de communication série

La communication série se fait de façon asynchrone, cela signifie qu'aucun signal de synchronisation (appelé horloge) n'est nécessaire: les données peuvent être envoyées à intervalle de temps arbitraire. En Contrepartie, le périphérique doit être capable de distinguer les caractères (un caractère a une longueur de 8 bits) parmi la suite de bits qui lui est envoyée. C'est la raison pour laquelle dans ce type de transmission, chaque caractère est précédé d'un bit de début (appelé bit *START*) et d'un bit de fin (bit *STOP*). Ces bits de contrôle, nécessaires pour une transmission série, occupant 20% de la bande passante (pour 10 bits envoyés, 8 servent à coder le caractère, 2 servent à assurer la réception).

Les ports série sont généralement intégrés à la carte mère, c'est pourquoi des connecteurs présents à l'arrière du boîtier, et reliés à la carte mère par une nappe de fils, permettent de connecter un élément extérieur. Les connecteurs séries possèdent généralement 9 ou 25 broches et se présentent sous la forme suivante (respectivement connecteurs DB9 et DB25) :

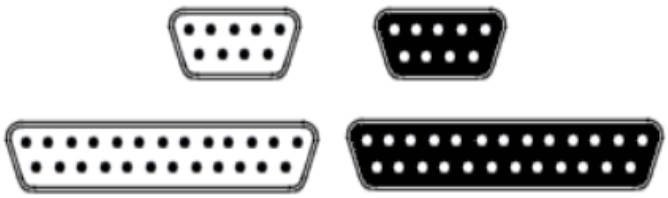


Figure I.4. Connecteurs mâle et femelle sub-DB9 et DB25

b. La communication en parallèle :

La transmission de données en parallèle consiste à envoyer des données simultanément sur plusieurs canaux (fils). Les ports parallèles présents sur les ordinateurs personnels permettent d'envoyer simultanément 8 bits (un octet) par l'intermédiaire de 8 fils Comme la figure I.5.

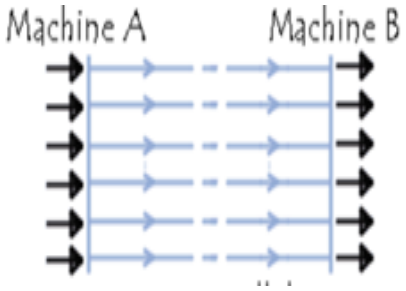


Figure I.5.Liaison parallèle

Les premiers ports parallèles bidirectionnels permettaient d'atteindre des débits de l'ordre de 2.4Mb/s. Toutefois des ports parallèles améliorés ont été mis au point afin d'obtenir des débits plus élevés :

- **Le port EPP** (EnhancedParralel *Port*, port parallèle amélioré) a permis d'atteindre des débits de l'ordre de 8 à 16 Mbps
- **Le port ECP** (EnhancedCapabilities Port, port à capacités améliorées), mis au point par Hewlett Packard et Microsoft. Il reprend les caractéristiques du port EPP en lui ajoutant un support Plug and Play, c'est-à-dire la possibilité pour l'ordinateur de reconnaître les périphériques branchés ce qui montre dans la figure I.6.

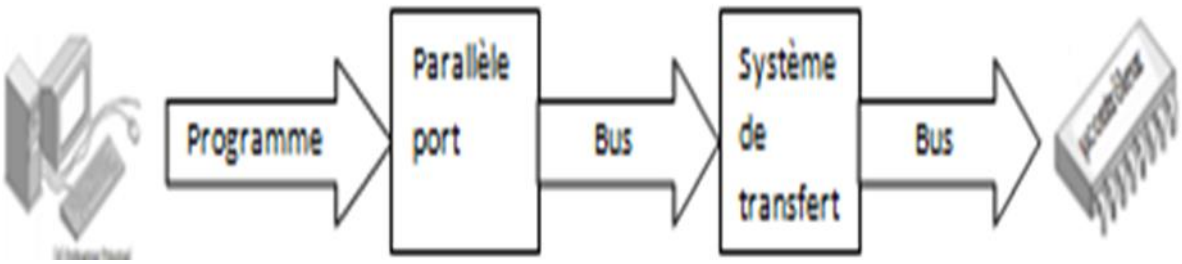


Figure I.6. Étapes de communication parallèle

Les ports parallèles sont, comme les ports série, intégrés à la carte mère. Les connecteurs DB25 permettent de connecter un élément extérieur.

Figure I.7. Connecteurs mâle et femelle sub- DB25

Les ports parallèles sont plus rapides que les ports série. Il est plus facile d'écrire des programmes pour les ports parallèles par rapport aux ports série. Mais les ports parallèles ont besoin de plus de lignes pour transférer les données. Par conséquent, les ports parallèles ne sont pas adaptés à la communication longue distance en raison du coût élevé et de la perte de données.

c. **Bus USB :**

Le bus USB (Universal Serial Bus, en français Bus série universel) est, comme son nom l'indique, basé sur une architecture de type série. Il s'agit toutefois d'une interface entrée-sortie beaucoup plus rapide que les ports séries standards. L'architecture qui a été retenue pour ce type de port est en série pour deux raisons principales :

- L'architecture série permet d'utiliser une cadence d'horloge beaucoup plus élevée qu'une interface parallèle, car celle-ci ne supporte pas des fréquences trop élevées (dans une architecture à haut débit, les bits circulant sur chaque fil arrivent avec des décalages, provoquant des erreurs).
- Les câbles série coûtent beaucoup moins cher que les câbles parallèles.

Port USB

Il existe deux types de connecteurs USB :

Les connecteurs dits de type A, dont la forme est rectangulaire et servant généralement pour des périphériques peu gourmands en bande passante (clavier, souris, webcam, etc.)
Les connecteurs dits de type B, dont la forme est carrée et utilisés principalement pour des périphériques à haut débit (disques durs externes, etc.) représenté dans la figure I.8.

Figure I.8. Différents types de port USB .

1. Alimentation +5V (VBUS) 100mA maximum
2. Données (D-)
3. Données (D+)
4. Masse (GND)

Fonctionnement du bus USB

L'architecture USB a pour caractéristique de fournir l'alimentation électrique aux périphériques qu'elle relie, dans la limite de 15 W maximum par périphérique. Elle utilise pour cela un câble composé de quatre fils (la masse GND, l'alimentation VBUS et deux fils de données appelés D- et D+) indiqué dans la figure suivante.

FigureI.9. Description des broches du port USB

I.1.3. Exemples des interfaces PC (Programmateur AVR) :

Il existe de nombreux moyens pour charger le code de programme dans une puce AVR. Les méthodes pour programmer les puces AVR varie de la famille AVR.

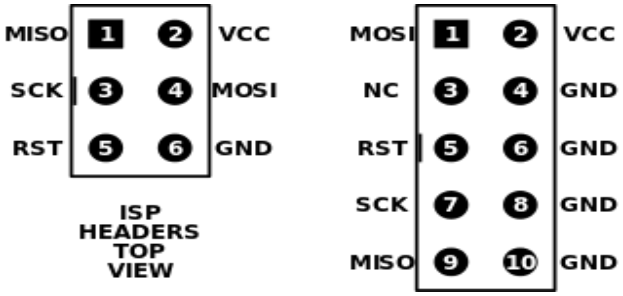
1. programmateur-isp-pour-avr-usb-isp-6-broches-et-10-broches-arduino :

Ce programmateur ISP pour Arduino et clones Arduino permet de programmer directement l'AVR de l'Arduino ou de charger un bootloader sur le circuit intégré au cœur de votre Arduino.

Ce programmateur pourra programmer les circuits intégrés de 64K max, et conviendra donc parfaitement à l'ATMEGA328P (mais pas à l'ATMEGA 2560). L'utilisation d'un programmateur permet aussi éventuellement de se passer du bootloader Arduino et de libérer de l'espace pour votre programme. [4]

Figure I.10.programmateur-ISP-pour-AVR-USB

ISP : La programmation dans le système méthode de programmation (ISP) est fonctionnellement effectuée par SPI, ainsi que quelques bidouilles de la ligne de réinitialisation. Tant que les broches SPI de l'AVR ne sont pas connectées à tout perturbateur, la puce AVR peut rester soudé sur un PCB en reprogrammation. Tout ce qui est nécessaire est un connecteur à 6 broches et un adaptateur de programmation. C'est la façon la plus courante de se développer



2. Carte programmateur AVR ISP interface parallèle :

Cette carte permet de programmer un microcontrôleur ATMEL de type AVR (ATmega8, ATMega32, ...) depuis le port parallèle imprimante LPT1 en utilisant la fonction ISP In System Programming.

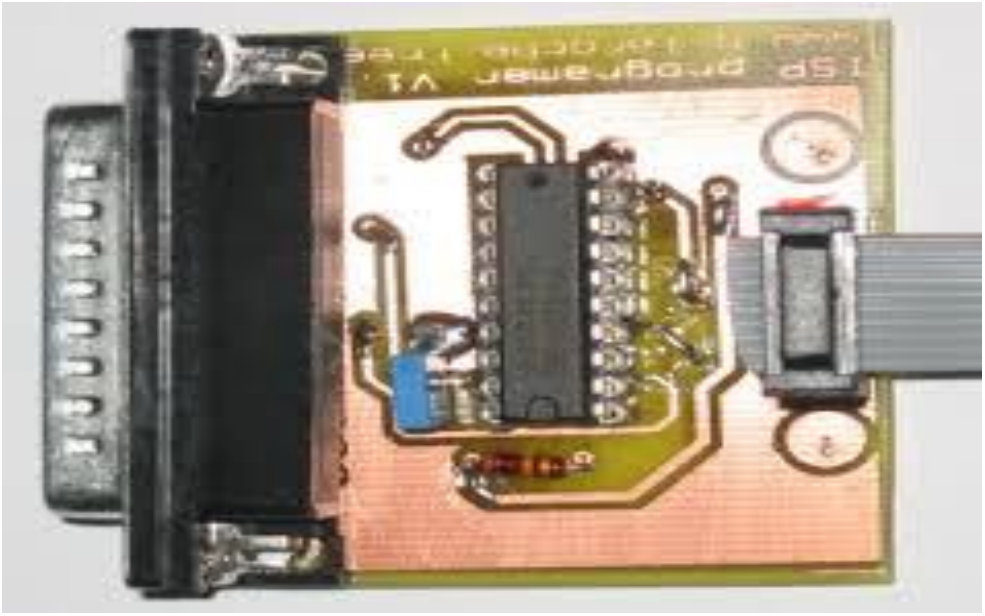


Figure I.11.programmateur-ISP-pour-AVR-Communication parallèle

Ce mode de programmation permet de programmer et/ou configurer le microcontrôleur directement sur la carte cible sans avoir besoin de le retirer et le positionner sur un programmateur séparé. [5]

Dans notre projet on va réaliser une interface PC communication série pour objectif de programmer les microcontrôleurs AVR

Dans le paragraphe qui suit, nous allons voir l'historique de l'AVR :

I.2. Définition AVR :

L 'AVR est une famille de microcontrôleurs RISC à l'architecture de Harvard développé par la société Américaine ATmel, il a été développé dans les laboratoires de l'entreprise situes en Norvège en 1990. L'AVR a été l'une des premières familles de microcontrôleur à utiliser un mémoire flash interne pour stocker le contenu du programme: cela permet d'annuler la mémoire de programme et de le réécrire avec une nouvelle version en quelques secondes et sans retirer le microcontrôleur de la carte sur laquelle il est monté, ce qui accélère considérablement le processus de correction et le réglage du code. Au cours des années nonante la plupart des micr²ocontrôleurs utilisés Un temps programmable ROM, EPROM, ou EEPROM. Pour cela Il est considéré comme l'un des microcontrôleurs les plus répandus au monde, en particulier en Chine.

I.2.1. Les familles des AVR

1- **AT90Sxxxx :** la famille classique, qui a été le premier lancement des microcontrôleurs AVR en 1990, a cessée de fabriquer.

2- **ATtinyxxx :** la petite famille est apparu en 2000, elle a petit nombre de pôles (6~32) et une taille de mémoire relativement petite (0.5~16 Kb)
1 ÷ 8 KB de (mémoire flash) pour le programme

Série de dispositifs matériels périphériques est limitée

3- **ATmegaxxxx :** la grande famille est apparu en 2003, elle a grand nombre de pôles (28~100)
4 à 256 kb (mémoire flash) pour le programme
Longue série d'instructions
Plus grand nombre de dispositifs disponibles (tels que la présence d'une ou plusieurs lignes série, la présence de plus PWM, etc.)

4- ATxmegaxxxx : la famille en évolution est apparue en 2008,

64-384 Ko de mémoire flash pour le programme

Longue série d'instructions

Bus de données interne 16 bits

Dispositifs disponibles encore augmenté par rapport à Méga

Gestion DMA et événement

5- AVR spéciaux pour des applications spécifiques :

On utilise les AVR comme :

- Contrôleur USB : comme les programmateurs SPI pour le transfert des données
- Contrôleur CAN : utilisé pour contrôler le protocole CAN et soutenir CAN open, Device Net, OSEK.
- Contrôleur Z-Link : utilisé dans les protocoles de transmission radio sans fil Zig Bee
- Contrôleur batterie : utilisé pour contrôler la charge des économies et il fonctionne à haute tension.
- Contrôleur Lighting : utilisé dans les applications de contrôle de puissance de la vitesse des moteurs et de l'intensité lumineuse.
- Contrôleur LCD : utilisé comme processeur de base pour les écrans LCD.
- Contrôleur auto motive : utilisé dans les systèmes de contrôle moteur et les systèmes de contrôle automobile.

6- FPSLIC (AVR avec FPGA) :

FPSLIC (Field Programmable System Level Integrated Circuit) ce circuit est un FPGA avec un cœur AVR, le cœur peut fonctionner jusqu'à 50 MHz en exécutant son programme en mémoire RAM à la différence des autres familles qui exécutent le programme en mémoire FLASH.

FPGA 5k à 40k portes

SRAM pour le code de programme AVR, contrairement à tous les autres AVRs

I.2.2. Principales caractéristiques :

• Haute performance
• Batterie faible
• Grand espace d'adressage
• Efficace
• Faible cher
• Architecture RISC comme ARM Microcontrôleur

I.2.3. Types de microcontrôleur AVR :

Microcontrôleur AVR Atmega8

Il se compose de 28 broches, d'une mémoire SRAM interne de 1 Ko, de 8 Ko de mémoire flash et de deux interruptions extérieures. Il a une interface à deux fils, une broche externe pour connecter deux tensions à deux entrées du comparateur.

Utilisations : Principalement utilisé pour construire des projets électriques et électroniques.

Microcontrôleur AVR Atmega16

Il se compose de 40 broches. Il a un type de mémoire flash, 16 MIPS Speed, 1KByte RAM, six modes d'économie d'énergie.

Utilisations : Il fonctionne sur Mobile Embedded System, satisfait Embedded System.

Microcontrôleur AVR Atmega32

Il se compose de 44 broches avec une taille de mémoire de 32 bits. Il a un type de mémoire flash, une vitesse de 16 MIPS, 2048 SRAM, une tension de fonctionnement allant de 2,7 à 5,5.

Microcontrôleur AVR Atmega328

L'architecture RISC contient une mémoire flash de 32 Ko, 2 Ko de SRAM, des plages de tension de fonctionnement de 1,8 à 5,5, 1 Ko d'EEROM (mémoire morte effaçable électriquement).

Utilisations : Largement utilisé dans les systèmes Arduino, robotique, de surveillance et de gestion de l'alimentation.

I.2.4. Les types de mémoires :

Avant de passer à la programmation des AVR, il faut d'abord connaitre les différents types de mémoires que l'on trouvé dans les microcontrôleurs AVR.

Alors Tout les microcontrôleurs AVR disposent de trois types de mémoire :

Mémoire Flash, prévue principalement pour recevoir le programme, qui est permanente

La mémoire vive (RAM = Random Access Memory), qui perd son contenu lorsque le circuit n'est plus alimenté. Pour commander des enseignes et afficheurs à LED, on placera souvent en mémoire non volatile non seulement le programme proprement dit, mais aussi les informations sur le contenu qui sera visualisé. En effet, les séquences d'une enseigne ou les textes d'un afficheur doivent être mémorisés de manière non-volatile, pour leur assurer un fonctionnement correct après une coupure de courant.

Une particularité de la mémoire Flash est son organisation en blocs. S'il est possible d'écrire une valeur dans une position mémoire précise, il n'est pas possible d'effacer une position mémoire seule. Les effacements se font toujours par bloc. La taille d'un bloc varie considérablement d'un microcontrôleur à un autre, parfois seulement 64 octets, jusqu'à plusieurs dizaines de kb.

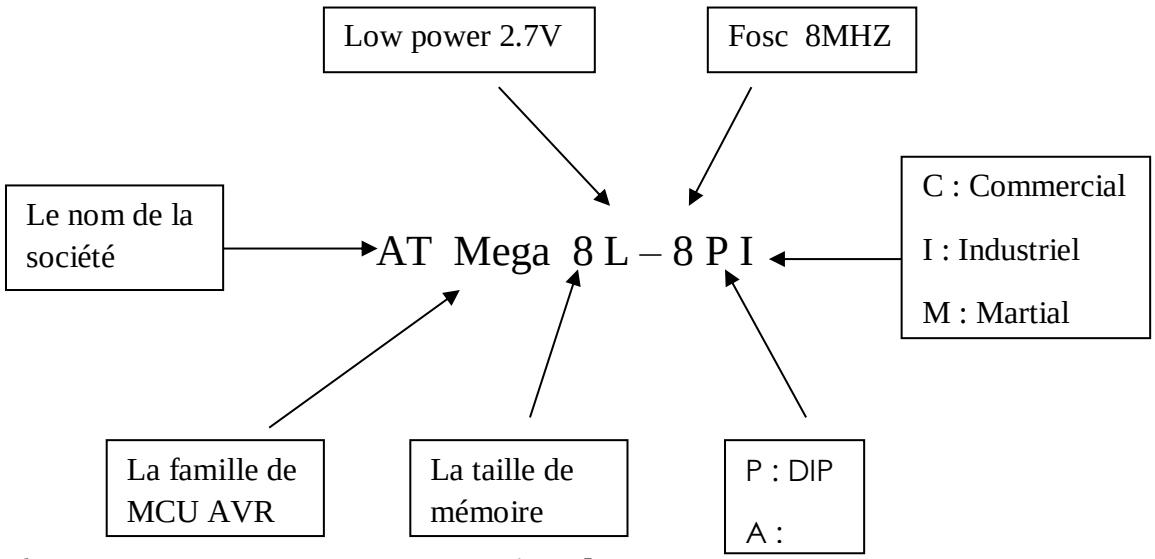
Les mémoires EEPROM

Certains microcontrôleurs disposent d'un troisième type de mémoire, en plus de la Flash et de la RAM. Il s'agit de mémoires EEPROM. Ce sont aussi des mémoires non-volatiles, mais chaque position mémoire peut être écrite et effacée indépendamment des autres. On trouve ce type de mémoire dans les microcontrôleurs AVR, dont l'ATmega328, bien connu pour être utilisé dans les Arduino.

L'accès en Lecture/Ecriture dans l'EEPROM

L'accès à l'EEPROM interne d'un microcontrôleur est généralement très simple. Des registres sont prévus pour l'adresse (EEAR) et pour la donnée (EEDR), en lecture ou en écriture. Sur les AVR, deux registres donnent l'accès aux adresses et aux données

I.2.5. Lecture de code de microcontrôleur AVR



I.2.6. Les langages de programmation des AVR :

Un langage de programmation est, d'un point de vue mathématique, un langage formel, c'est-à-dire un ensemble de suites (appelés mots) de symboles choisis dans un ensemble donné (appelé alphabet) qui vérifient certaines contraintes spécifiques au langage (syntaxe).

Dans le cas des langages de programmation:

- Soit on utilise un langage de bas niveau : on écrit un programme directement dans le langage machine (ou langage natif) compréhensible par le processeur, mais rarement intelligible puisque rares sont les personnes qui parlent le binaire couramment.

- Soit on utilise un langage de haut niveau : on écrit un programme dans un langage intelligible pour le processeur, qui sera ensuite "traduit" en langage machine afin que le processeur l'exécute.

Ces langages permettent de décrire des taches sans se soucier des détails sur la manière dont la machine l'exécute. Deux stratégies sont utilisées pour les langages de haut niveau. La différence réside dans la manière dont on traduit le programme, qui est à l'origine dans un ou plusieurs fichiers texte appelés fichiers source ou code source.

- L'Assembleur (ASM)

Inventé en 1945 par John Von Neumann, l'assembleur fut le premier vrai langage de

Programmation mais aussi le plus proche du langage machine (il s'adresse directement au processeur) et de ce fait, l'un des plus compliqués. Les instructions sont de type MOV, ADD,

PUSH, POP, INT, bref rarement plus de 5 lettres, ce qui les rend difficile à mémoriser.

- Le BASIC

Le BASIC (Beginners All purpose Simple instructions Code) est apparu en 1964. Sans doute le langage de programmation le plus simple au monde. Il permet de créer des programmes très facile et constitue ainsi une initiation à la programmation, surtout que c'est bien le seul usage qu'on pourrait encore lui trouver. En effet, le BASIC est loin d'être un langage "puissant"

- Le JAVA

Tirant ses origines du début des années 90, né officiellement en 1995, ce langage présente l'avantage d'être portable ; on peut exécuter un programme JAVA sous Windows, Mac, Linux, Cette portabilité est dû à une particularité de l'implémentation la plus répandue du langage JAVA : celui-ci n'est pas compilé en code machine comme les autreslangages mais dans un langage intermédiaire dit "ByteCode"

- Langage C

Le C est un langage très populaire. Il est l'un des langages les plus connus et les plus utilisés qui existent, utilisé pour programmer une grande partie des logiciels que nous
Connaissons. Il est aussi à la base des plus grands systèmes d'exploitation tels Unix (Linux et Mac OS) ou Windows. [6]

De nos jours, les microcontrôleurs sont le composant majeur des systèmes embarqués . Il est utilisé dans un système sans interférence humaine et fonctionne sur le système embarqué mobile

Conclusion

Ce chapitre permet d'avoir une vue d'ensemble sur les différents interfaces PC, son parcours ainsi quelques-uns de ces modèles. Ensuite on a défini et détaillé les microcontrôleurs AVR Par ailleurs, nous pouvons conclure que le programmateur AVR c'est un dispositif important dans le domaine d'automatique pour la communication entre la partie de commande et la partie opérative.

Chapitre II : Etude théorique sur l'AVR ATMega238

Introduction

Tout d'abord, une présentation des microcontrôleurs AVR quand peut les programmés à partir de notre programmateur réalisera. Ensuite on se base sur Atmega328P qui est plus utilisé et simple pour comprendre le fonctionnement du MCU et de l'action des instructions sur les éléments tel que la mémoire, l'unité arithmétique et logique, l'EEPROM et les interfaces diverses.

Les microcontrôleurs de la famille AVR d'Atmel possèdent de nombreuses caractéristiques. Bien qu'ayant un cœur similaire, les différents modèles se distinguent en termes de vitesse, mémoire, nombres d'entrées/sorties mais aussi au niveau de fonctions particulières.
Ceci donne l'option de choisir le type de microcontrôleur selon de l'utilisation qui est requise. Alors on peut programmer des différents types des microcontrôleurs selon leurs brochages. Comme cette figure indique.

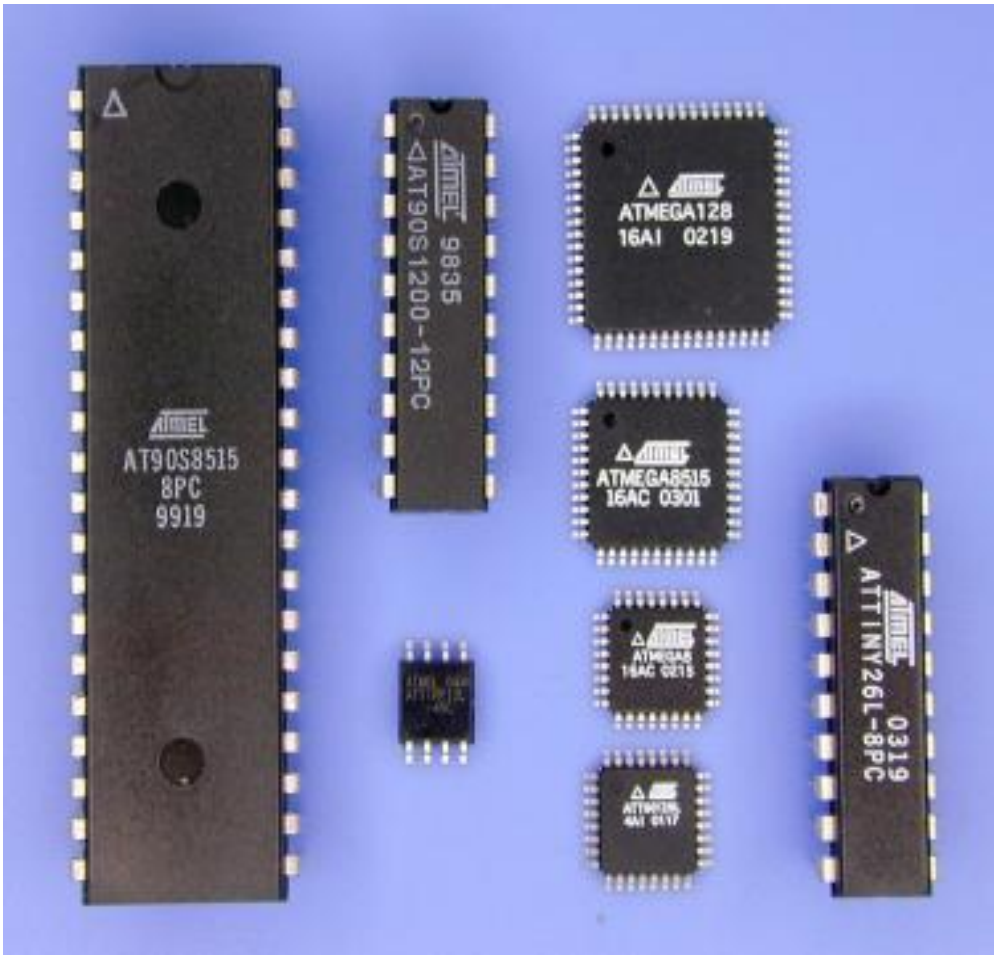


Figure II.1.les différents microcontrôleurs

II.2. Atmega328P :

L'Atmega328P est un simple puce microcontrôleur créé par Atmel dans la MegaAVR famille (plus tard micro chip technologie a acquis Atmel en 2016).
Il a une architecture de Havard modifiée 8-bits RISC cœur de processeur.
L'ATMega328 est un microcontrôleur populaire car il est un composant majeur des produits de carte Arduino Uno et Nano.

- Ses excellentes caractéristiques comprennent la rentabilité, la faible dissipation de puissance, le verrouillage de programmation pour des raisons de sécurité, un véritable compteur de minuterie avec oscillateur séparé.
- Il est normalement utilisé dans les applications de systèmes embarqués , nous pouvons tous les concevoir à l'aide de ce microcontrôleur.
- les fonctionnalités complètes d'ATMega328P sont :
    - ✓ nombre de broches : 28
    - ✓ CPU : AVC RISC 8 bits
    - ✓ Mémoire flash : 32 Ko
    - ✓ Mémoire données EEPROM : 1 Ko
    - ✓ Mémoire SRAM : 2 Ko
    - ✓ 32 Registres de travail d'accès rapide pour l'ALU
    - ✓ Ponts parallèles : 3 avec 23 broches E/S
    - ✓ Fréquence d'horloge : 16 Mhz (Maxi Tolérée 20 Mhz)
      Donc 16 cycles d'horloge par micro second
  - Périphériques internes :
    - ✓ 6 Convertisseurs Analogique/Numérique 10-bit
    - ✓ Comparateur analogique
    - ✓ 1 Timer 16-bit (T1), 2 Timer 8-bit (T0, T2)
    - ✓ 6 Canaux PWM
    - ✓ 1 Chien de garde watch dog
    - ✓ Communication en série : SPI, USART, TWI (I2C)
    - ✓ 26 Interruptions
    - ✓ 5 Modes d'économie d'énergie [8]
  - Brochage :
    - ✓ VCC : Alimentation 5V
    - ✓ GND : La masse

- ✓ AVCC : Alimentation CAN
- ✓ AREF : Entrée comparateur analogique
- ✓ E/S (GPIO) : PORTB(8), PORTC(7), PORTD(8)

Toutes les fonctionnalités précédentes sont logées dans un boîtier 28 DIP qui permet aux concepteurs de prototyper facilement leurs conceptions avant de s'engager dans la technologie de montage en surface. Avec une plage de température de -40 ° C à 105 ° C et une plage de tension de 1,8 V à 5,5 V, l'ATMEGA328 est vraiment un microcontrôleur polyvalent.

II.2.1. Applications générales :

- Automatisation de la maison
- Équipements d'interface et panneaux
- Systèmes basés sur Arduino
- Éducation

II.2.2. Descriptions des broches :

Nous allons passer en revue le brochage de la puce Atmega328.

L'ATMega328 possède 28 broches. Il dispose de 14 broches d'E / S numériques, dont 6 peuvent être utilisées comme sorties PWM et 6 broches d'entrée analogique.

Ces broches d'E / S représentent 20 des broches. Le brochage de l'Atmega328P est illustré ci-dessous.

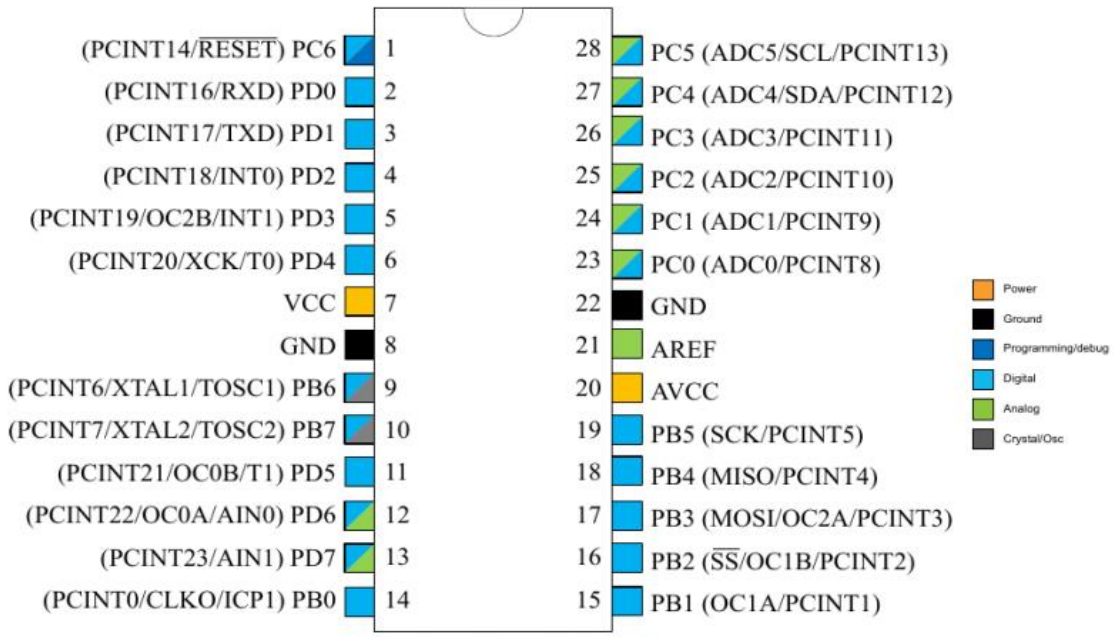


Figure II.2.le brochage de L'Amega328P

Broche 1 : PC6 Réinitialiser : La broche par défaut est utilisée comme broche RESET. PC6 peut uniquement être utilisé comme broche d'E / S lorsque le fusible RSTDISBL est programmé.

Broche 2 : PD0 (RXD) : RXD (broche d'entrée de données pour USART) Interface de communication série USART [Peut être utilisé pour la programmation]

Broche 3 : PD1 (TXD) : TXD (broche de sortie de données pour USART) Interface de communication série USART [Peut être utilisé pour la programmation]

INT2 (entrée d'interruption externe 2)

Broche 4 :PD2 (INT0) : Source d'interruption externe 0

Broche 5 :PD3 (INT1/ OC2B) : Source d'interruption externe1

OC2B (PWM - Sortie Timer / Counter2 Comparer la sortie Match B)

Broche 6 :PD3 (INT1/ OC2B) : Source d'interruption externe1

OC2B (PWM - Sortie Timer / Counter2 Comparer la sortie Match B)

Broche 7 :PD4 (XCK / T0) : T0 (entrée de compteur externe Timer0)

XCK (E / S d'horloge externe USART)

Broche 8 : VCC : Connecté à une tension positive

Broche 9 : GND : Connecté à la terre

Broche 10 :PB6 (XTAL1/ TOSC1) :XTAL1 (broche 1 de l'oscillateur d'horloge à puce ou entrée d'horloge externe)

TOSC1 (broche d'oscillateur de minuterie 1)

Broche 11 :PB7 (XTAL2/TOSC2) :XTAL2 (broche 2 de l'oscillateur d'horloge à puce)

TOSC2 (broche d'oscillateur de minuterie 2)

Broche 12 :PD5 (T1/OC0B) : T1 (entrée compteur externe Timer1)

OC0B (PWM - Sortie minuterie / compteur0 Comparer la sortie Match B)

Broche 13 : PD6 (AIN0/ OC0A) :AIN0 (comparateur analogique I / P positif)

OC0A (PWM - Sortie minuterie / compteur0Comparer la correspondance avec la sortie A)

Broche 14 :PD7 (AIN1) :AIN1 (comparateur analogique I / P négatif)

Broche 15:PB0 (ICP1 / CLKO):OC1A (sortie Timer / Counter1 Compare Match Output)

Broche 16 :PB2 (SS / OC1B) : SS (Entrée SPI Slave Select). Cette broche est basse lorsque le contrôleur agit comme esclave.

[Interface périphérique série (SPI) pour la programmation]

Broche 17 :PB3 (MOSI / OC2A) : MOSI (entrée esclave de sortie principale). Lorsque le contrôleur agit comme esclave, les données sont reçues par cette broche. [Interface périphérique série (SPI) pour la programmation]

OC2 (Timer / Counter2 Output Compare Match Output)

Broche 18:PB4 (MISO): MISO (Master Input Slave Output). Lorsque le contrôleur agit comme esclave, les données sont envoyées au maître par ce contrôleur via cette broche.

[Interface périphérique série (SPI) pour la programmation]

Broche 19 :PB5 (SCK) : SCK (horloge série du bus SPI). Il s'agit de l'horloge partagée entre ce contrôleur et un autre système pour un transfert de données précis.

[Interface périphérique série (SPI) pour la programmation]

Broche 20 : AVCC : Alimentation pour convertisseur ADC interne

Broche 21 : AREF : Broche de référence analogique pour ADC

Broche 22 : GND : SOL

Broche 23 :PC0 (ADC0) :ADC0 (ADC Input Channel 0)

Broche 24 :PC1(ADC1) :ADC1 (ADC Input Channel 1)

Broche 25 :PC2 (ADC2) :ADC2 (ADC Input Channel 2)

Broche 26 : PC3 (ADC3) : ADC3 (ADC Input Channel 3)

Broche 27 :PC4 (ADC4/ SDA) : ADC4 (ADC Input Channel 4)

SDA (ligne d'entrée / sortie de données de bus série à deux fils)

Broche 28 : PC5 (ADC5/ SCL) : ADC5 (ADC Input Channel 5)

SCL (ligne d'horloge de bus série à deux fils). [9]

II.1. 3. Les Mémoires d'Atmega328P :

La mémoire est l'un des principaux composants d'un ordinateur ou d'un microcontrôleur comme l'Atmega328P

Cette section décrit les différents types de mémoire de l'appareil. L'architecture AVR a deux principaux espaces mémoire, la mémoire de données et l'espace mémoire de programme. De plus, l'appareil dispose d'un Mémoire EEPROM pour le stockage des données. Tous les espaces mémoire sont linéaires et réguliers.

II.1.3.1. Mémoire de programme flash reprogrammable dans le système :

L'ATmega328 / P contient une mémoire Flash reprogrammable intégrée au système de 32 Ko pour le programme espace de rangement. Étant donné que toutes les instructions AVR ont une largeur de 16 ou 32 bits, le Flash est organisé en 16K x 16. Pour sécurité du logiciel, l'espace mémoire du programme Flash est divisé en deux sections - Section Boot Loader et la section du programme d'application dans l'appareil.

II.1.3.2. Mémoire de données SRAM :

L'appareil est un microcontrôleur complexe avec plus d'unités périphériques que ce qui peut être pris en charge dans le 64emplacements réservés dans l'Opcode pour les instructions IN et OUT. Pour l'espace d'E / S étendu de 0x60

- 0xFF dans SRAM, seules les instructions ST / STS / STD et LD / LDS / LDD peuvent être utilisées.

Les 2303 emplacements de mémoire de données inférieurs adressent à la fois le fichier de registre, la mémoire d'E / S, les E / S étendues mémoire et les données internes SRAM. Les 32 premiers emplacements concernent le fichier de registre, les 64 suivants emplacements de la mémoire d'E / S standard, puis 160 emplacements de mémoire d'E / S étendues et les 2K emplacements suivant s'adresser les données internes SRAM.

II.1.3.3. Mémoire de données EEPROM :

L'ATmega328 / P contient 1K octets de mémoire EEPROM de données. Il est organisé comme une donnée distincte espace, dans lequel des octets uniques peuvent être lus et écrits.

L'EEPROM a une endurance d'au moins100 000 cycles d'écriture / effacement. L'accès entre l'EEPROM et la CPU est décrit ci-après, spécification des registres d'adresses EEPROM, du registre de données EEPROM et de la commande EEPROM.

En lecture / écriture EEPROM :

Les registres d'accès EEPROM sont accessibles dans l'espace d'E / S.

Le temps d'accès en écriture pour l'EEPROM. Une fonction d'auto-chronométrage permet cependant au le logiciel utilisateur détecte quand l'octet suivant peut être écrit. Si le code utilisateur contient des instructions qui écrivent l'EEPROM, certaines précautions doivent être prises. Dans les alimentations fortement filtrées, le VCC est susceptible d'augmenter ou tomber lentement à la mise sous / hors tension. Cela fait fonctionner l'appareil pendant un certain temps à une tension inférieure que spécifié comme minimum pour la fréquence d'horloge utilisée. Veuillez-vous référer à Prévention de la corruption EEPROM pour savoir comment éviter les problèmes dans ces situations. [10]

Afin d'éviter les écritures EEPROM involontaires, une procédure d'écriture spécifique doit être suivie. Faire référence à la description du registre de contrôle EEPROM pour plus de détails à ce sujet.

II.2.4. Les Timers :

II.2.4.1. Présentation du Timer :

Les Timers/compteurs sont des périphériques de gestion de temps. Ils permettent des réaliser Les fonctions suivantes :

- comptage des évènements
- synchronisation des signaux
- fixer le débit d'une liaison série synchrone ou asynchrone
- génération des événements périodiques (échantillonnage des signaux analogiques, rafraichissement des afficheurs multiplexés …)
- génération des signaux périodiques (carré, MLI …)
- mesure de temps…

II.2.4.2. Fonctionnement du Timer :

Les Timers sont des compteurs formés généralement d'un pré-diviseur suivi d'un registre compteur de 8 ou 16 bits. L'entrée d'horloge peut être interne (mode timer) ou externe (mode

compteur d'événements). Lorsque le registre compteur atteint sa valeur maximale et repasse à 0, un bit indicateur (flag) sera positionné et une interruption pourra être générée, informant ainsi la CPU du débordement du timer. Il faut bien noter que le programmeur devra remettre à zéro cet indicateur après chaque débordement.

II.2.4.3. Les Timers de l'ATmega328P :

Le microcontrôleur AVR ATmega328P d'Atmel qui possède 3 timers :

• Le timer0, sur 8 bits, utilisé par les fonctions delay (), millis () et micros (). Il commande

Également des PWM sur les broches 5 et 6.

• Le timer1, sur 16 bits, qui compte de 0 à 65535 (0 à FFFF en hexadécimal) et qui est utilisé

Par la bibliothèque Servo ou bien pour de la PWM sur les broches 9 et 10.

• Le timer2, sur 8 bits, qui est utilisé par la fonction Tone () ou bien pour de la PWM sur les

Broches 3 et 11. [11]

II.2.5. Les interruptions de l'ATmega328P :

II.2.5.1. Définition :

Une interruption est un déroutage automatique et « quasi-instantané » du programme en cours d'exécution, en réponse à un signal émis par un périphérique interne ou par une entrée externe (appelés sources d'interruption), vers un sous-programme permettant de gérer l'évènement.

- Le mécanisme d'interruption s'oppose à la « scrutation », où le µP interroge régulièrement l'état de la source, par logiciel
  - dans la scrutation, le processeur obéit à des instructions du programme pour vérifier l'état des sources
  - alors que l'interruption est déclenchée automatiquement sur demande d'une source d'interruption, indépendamment du programme en cours d'exécution
  - Exemple du reset

Liste des interruptions de l'Atmega328P :

- 26 sources d'interruptions
- Leurs vecteurs sont constitués de 2 mots d'instructions
- Ils sont ordonnés par priorité décroissante. [12]

VectorNo.	Program Address(2)	Source	Interrupt Definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

TableauII.1.Liste des interruptions

II.2.6. Les ports d'entrées/sorties(PORTx) :

Les microcontrôleurs ATMEL sont pourvus des 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, … Mais pour le moment nous allons voir la fonction Numérique des ports.

II.2.6.1. Registres de contrôle des ports

- DDRx : le registre de direction (sens de transfert) du port
- PORTx : le registre de donnée du port
- PINx : le registre d'entrée du port

x représente le nom de port (B, C ou D)

Chaque registre et configurable bit à bit, c'est à dire que sur l'on peut utiliser sur le même port des fonctions en entrée et/ou en sortie simultanément. Par exemple, on peut avoir les quatre premiers bits en entrée et les quatre derniers bits en sortie sur un même port Notez que chacun des registres d'E / S a une largeur de 8 bits, et que chaque port a un maximum de 8 broches, là pour chaque bit des registres d'E / S affecte l'une des broches, le contenu du bit 0 de DDRB représente la direction de la broche PB0, et ainsi de suite).

- Chaque port de 8 bits est limité à un courant total de 200 mA.

- Le microcontrôleur lui-même peut supporter au maximum un courant de 400 mA

II.2.6.2. Fonction des registres :

DDRx : **enregistrer le rôle dans la sortie des données :**

Chacun des ports B-D de l'ATmega328P peut être utilisé pour l'entrée ou la sortie. Le registre d'E / S DDRx est utilisé uniquement dans le but de faire d'un port donné un port d'entrée ou de sortie. Par exemple, pour écrire un port, nous écrivons 1 dans le registre DDRx. En d'autres termes, pour envoyer des données à toutes les broches du port B, nous avons d'abord mis 0bi1111111 dans le registre DDRB pour que toutes les broches soient sorties.

Pinx : inscrire le rôle dans la saisie des données :

Pour lire les données présentes sur les broches, nous devrions lire le registre PINx. Il faut noter que pour lire des données dans la CPU à partir de broches, nous lisons le contenu du registre PINx, alors que pour envoyer des données aux broches, nous utilisons le registre PORTx.

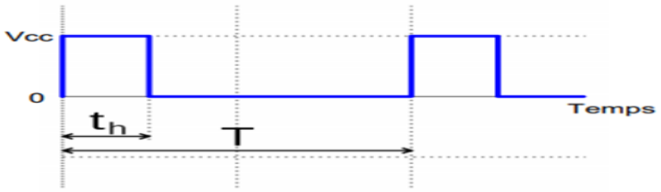
PORTx: enregistrer le rôle dans l'écriture des données :

Il y a une résistance de pull-up pour chacune des broches de l'AVR. Si nous mettons 1 dans des bits du registre PORTx, les résistances pull-up sont activées. Dans les cas où rien n'est connecté à la broche ou si les appareils connectés ont une haute impédance, la résistance tire la broche. [13]

II.2.7. PWM ATMega328P :

PWM signifier Pulse Width Modulation. Cette fonction fournit des microcontrôleurs au moyen, apparemment, de fournir des valeurs analogiques de tension entre (0-5) v. Au lieu de fournir des valeurs numériques faibles (0v) ou élevées (5v). [14]

Le signal est carré. Le niveau bas correspond généralement à 0 Volt. La période est notée **T** la durée de l'impulsion (pour laquelle la tension est celle de l'état haut) est appelée **th.**tel que la figureII.3 nous montre.



FigureII.3.Pulse Width Modulation

1. Applications :

- Variateurs de vitesse des moteurs
- Convertisseurs: AC/DC, DC/AC, DC/DC, AC/AC
- Générateur des signaux
- Modulateurs
- La conversion numérique-analogique
- Les amplificateurs de classe D
- Contrôle de puissance

II.2.8. Les Types de communication série :

1-SPI (interface périphérique série) :

Il s'agit d'un protocole de communication série de type synchrone qui se compose de deux lignes de données (MOSI et MISO), une ligne d'horloge (SCK) et une ligne de sélection d'esclave (SS) (voir la figure II.4).

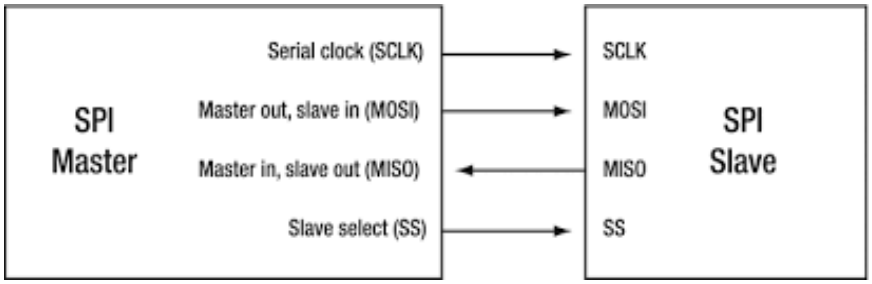


Figure II.4.Interface périphérique série

Maître – Appareil qui fournit une horloge pour la communication
Esclave – Appareil autre que le maître qui utilise l'horloge du maître pour communiquer
MOSI – Master Out Salve In (ligne par laquelle le maître envoie des données à ses esclaves)
MISO – Master In Slave Out (ligne à travers laquelle la salve répond au maître)
SCK – Serial Clock (horloge fournie par le dispositif maître)
SS – Slave Select (ligne utilisée pour sélectionner l'esclave avec lequel le maître veut communiquer)

Dans un SPI, à un moment donné, il ne peut y avoir qu'un seul appareil maître et plusieurs autres esclaves en dessous qui ne répondent qu'à l'appel du maître. L'ensemble de la communication est géré par le maître lui-même ; aucun esclave ne peut envoyer des données de sa propre volonté. Le maître envoie des données via MOSI tandis que les esclaves répondent via la ligne MISO. Dans l'ensemble du processus, SCK (horloge série) joue un rôle très important, chaque appareil esclave dépend de cette horloge pour lire les données de MOSI et répondre via MISO. SS (sélection d'esclave) est utilisé pour éveiller un esclave particulier avec lequel le maître veut communiquer. Voici une illustration de SPI :

Maintenant, il y a peu de registres qui sont utilisés pour implémenter la communication SPI. Nous les avons ci-dessous et comme vous pouvez le voir, nous avons SPDR, SPSR et SPCR, alors voyons chacun d'eux.

SPDR (SPI Data Register)

➢ Ceci est utilisé pour stocker un octet de données à transférer ou à recevoir.

SPSR (SPI Status Register)

➢ Ce registre contient les bits d'état impliqués dans la communication SPI

SPCR (SPI Control Register)

➢ Ce registre contient les bits de contrôle impliqués dans la communication SPI.

Tous les registres ci-dessus ont une longueur de 8 bits.

Avantages :

1. Fournit une communication série synchrone qui est beaucoup plus fiable que asynchrone
2. Plusieurs appareils (esclaves) peuvent être connectés à un seul maître
3. Forme plus rapide de communication série

Inconvénients :

1. Nécessite plusieurs fils de sélection d'esclaves pour connecter plusieurs esclaves
2. Seul le maître a le contrôle sur l'ensemble du processus de communication ; deux esclaves ne peuvent pas communiquer directement entre eux. [15]

2- I2C (circuit inter-intégré) ou interface à deux fils

Un autre protocole de communication série synchrone très utile est le protocole I2C ou Inter-Integrated Circuit. Contrairement à SPI, I2C n'utilise que deux fils pour l'ensemble du processus, c'est peut-être pourquoi il est également connu sous le nom de protocole TWI (Two Wire Interface). Ces deux fils sont SDA (Serial Data) et SCL (Serial Clock). Le protocole I2C peut prendre en charge plusieurs appareils esclaves, mais contrairement à SPI, qui ne prend en charge qu'un seul appareil maître, I2C peut également prendre en charge plusieurs appareils maîtres. Chaque appareil envoie / reçoit des données en utilisant un seul fil qui est SDA. SCL maintient la synchronisation entre les appareils via une horloge commune fournie par le maître actif.

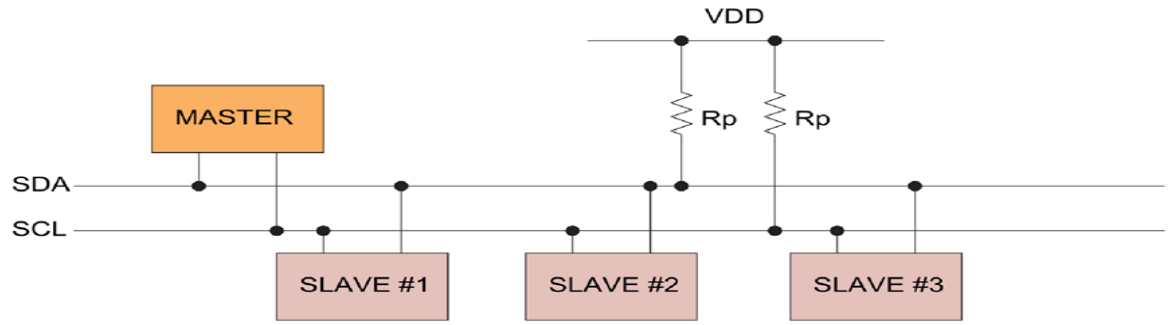


Figure II.5. I2C Interface à deux fils

Chaque esclave a sa propre adresse unique de 7 à 10 bits que le maître utilise pour les identifier. Chaque fois que le maître veut envoyer des données, il génère d'abord une requête qui a l'adresse particulière de cet esclave. Chaque esclave fait correspondre cette adresse avec la sienne et celui dont l'adresse est mise en correspondance répond au maître. Chaque message commence par une condition de démarrage et se termine par une condition d'arrêt. Un seul message peut contenir plusieurs octets de données, chacun ayant un bit d'acquittement (ACK) ou d'acquittement négatif (NACK) entre eux. Des résistances de rappel avec SDA et SCL sont nécessaires pour exécuter ce protocole.

Avantages :

1. Plusieurs maîtres et plusieurs esclaves peuvent être interfacés ensemble
2. Seuls deux fils sont nécessaires pour cette communication

Inconvénients :

1. Il est plus lent que SPI car beaucoup de travail de cadrage est effectué dans ce protocole.

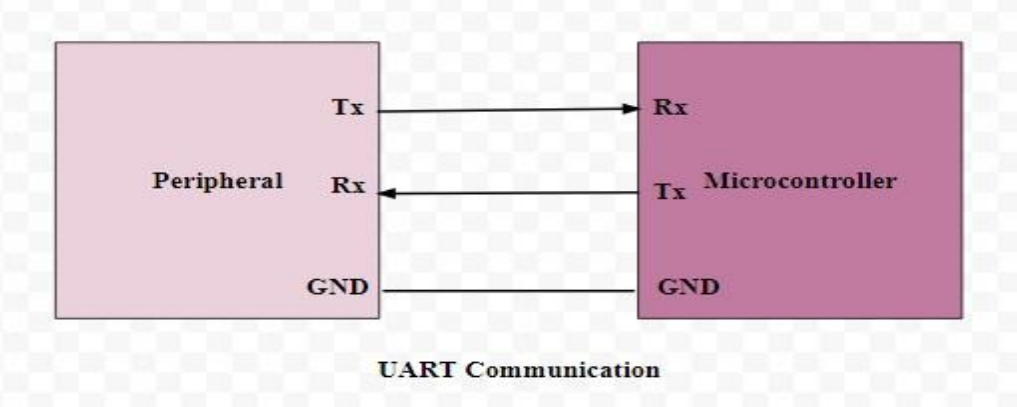
[16]

3 - UART / USART :

UART signifie récepteur et émetteur asynchrone universel tandis que USART signifie récepteur et émetteur universel synchrone et asynchrone. La différence entre eux est que UART effectue uniquement une communication série asynchrone, tandis que USART peut effectuer des processus de communication série synchrones et asynchrones.

Pour le mode asynchrone, ce protocole utilise seulement deux fils, à savoir Rx et Tx. Comme aucune horloge n'est nécessaire ici, les deux appareils doivent utiliser leurs horloges internes indépendantes pour fonctionner. Pourtant, il existe un terme appelé débit en bauds qui aide ces appareils à rester synchronisés en fixant la vitesse d'échange de données. Le débit en bauds fait référence au nombre de bits de données transmis par seconde, de sorte que les deux appareils doivent fonctionner sur le même débit en bauds afin de maintenir son bon fonctionnement. UART / USART a une grande limitation que seuls deux appareils peuvent communiquer en utilisant ce protocole à la fois. La broche TX d'un appareil transmet des données à la broche RX d'un autre appareil et de même TX de ce dernier transmet des données au RX de l'ancien appareil. C'est ainsi que se déroule l'échange de données.

Remarque : les deux appareils communicants doivent avoir une masse commune (GND). [17]



Avantages :

1. Fournit à la fois une communication série synchrone et asynchrone
2. Disponibilité de divers débits en bauds le rendant approprié pour des applications et des périphériques étendus
3. Une des formes les plus simples de communication série

Inconvénients :

1. Peut connecter seulement deux périphériques à la fois

Utilisez SPI lorsque vous n'avez qu'un seul maître et plusieurs appareils esclaves. SPI s'avère être un protocole plus rapide pour cela. Lorsque vous avez également plusieurs périphériques maîtres, mis à part plusieurs périphériques esclaves, il est préférable d'utiliser I2C ou TWI sur SPI. Cela réduira également le nombre de fils à utiliser. Maintenant, si vous recherchez une communication série de périphérique à périphérique, USART / UART se révèle être le meilleur car il est facile à gérer et largement utilisé dans de nombreux périphériques.

Conclusion :

Dans ce chapitre on a vu les différents types des microcontrôleurs AVR disponible à programmer, mais on a mis l'accent sur la description de l'Atmega328 et leurs caractéristiques qui nous aident de choisir le montage qui convient pour notre programmeur.

On a vu aussi les types de communication séries qui sont trouvés dans les microcontrôleurs AVR et d'après cette étude on a trouvé le montage.

Dans Le chapitre qui suit en va étudier plusieurs montages avec leur IDE.

Chapitre III :
Description des Parties Logicielle et Materielle

Introduction

Le programmateur des microcontrôleurs AVR 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.

III.1. Architecture de programmateur AVR (Hardware) :

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

III.1.1. Définition du circuit imprimé :

Le circuit imprimé est un support plan, flexible ou rigide, généralement composé d'époxy ou de fibre de verre, il possède des pistes électriques à base de cuivre qui permettent la mise en relation électrique des composants électroniques. Pour créer un système électronique qui fonctionne et qui réalise les opérations demandées, ce système porte le nom de carte électronique.

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, qu'on a déjà étudié dans le chapitre précédent. Le schéma synoptique dans la figure III.3 nous donne une idée générale sur l'architecture du programmateur AVR. [18].

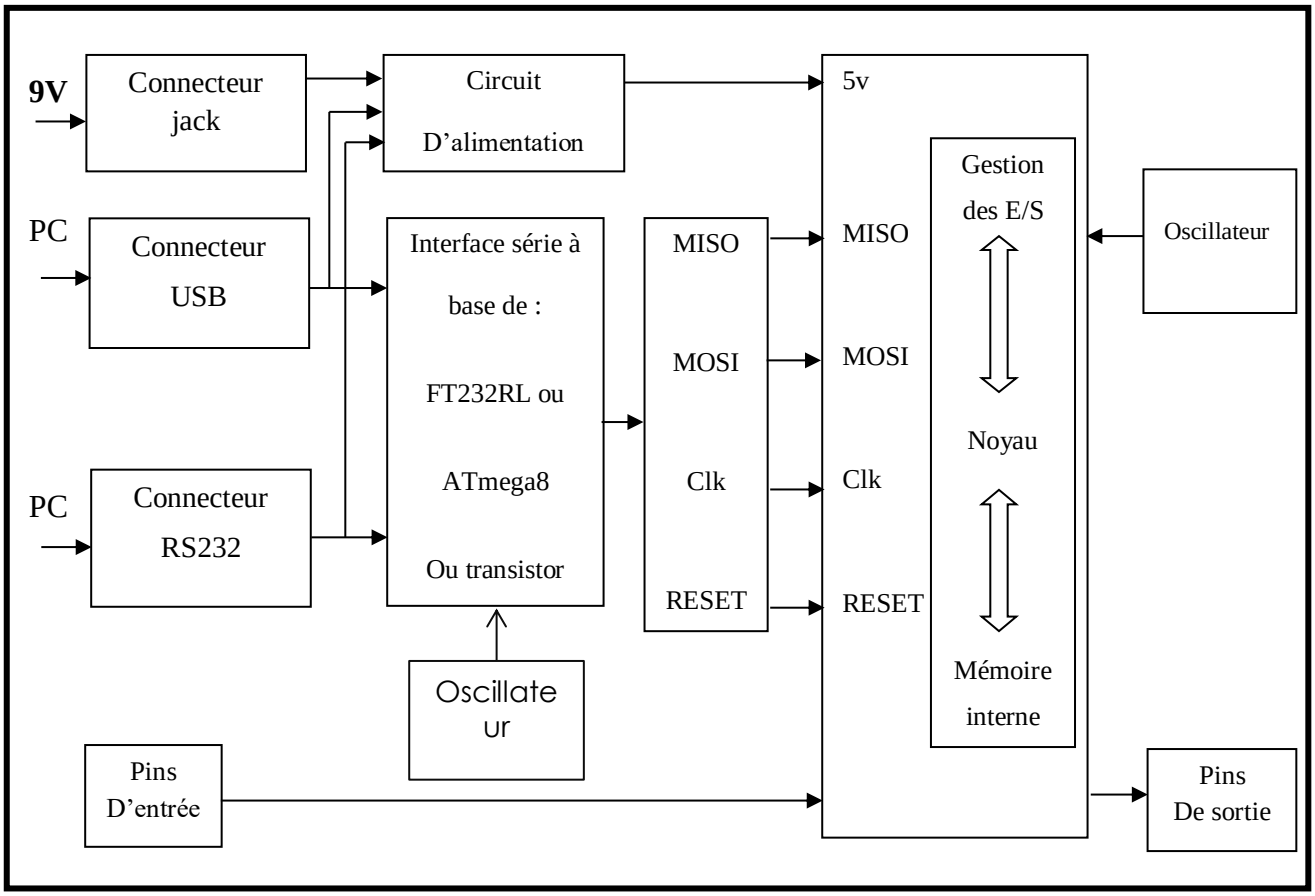


Figure III.1. Schéma synoptique du programmateur AVR

Les microcontrôleurs AVR sont faciles à programmer avec n'importe quel PC doté d'un port série gratuit. Cependant, le port série ne peut pas être connecté directement au microcontrôleur AVR

Pour cela on verra trois types de montage d'interfaçage de communication :

III.1.2. Schéma électrique avec un adaptateur FTDI :

Le premier montage du programmateur se base sur la communication série à travers un adaptateur convertisseur FT232RL USB à TTL série. [19]

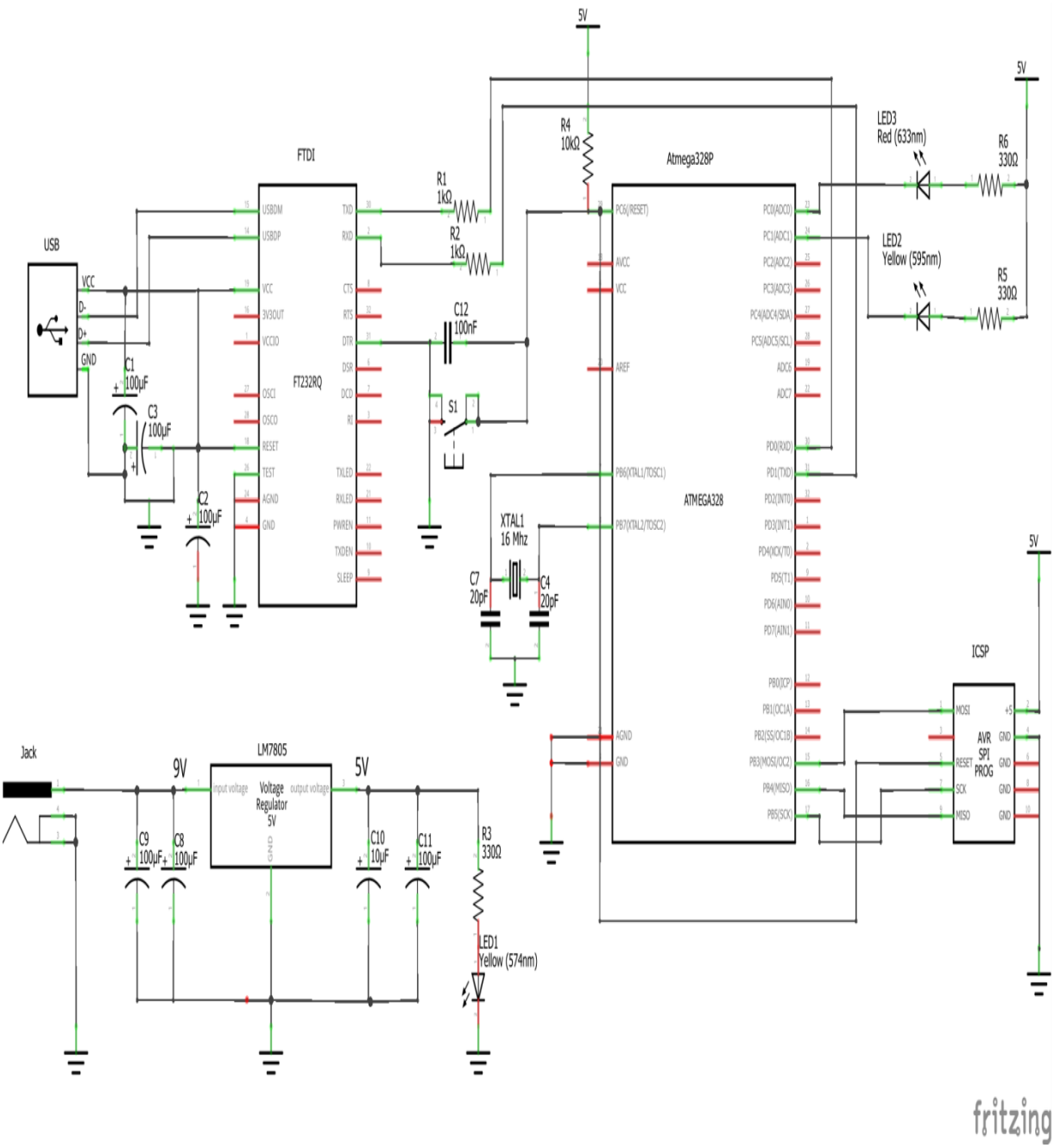


Figure III.2. Schéma électrique de programmateur AVR à base de FT232RL

Interface USB-UART :

La figure III.3 représente le schéma électrique de l'interface série USB.

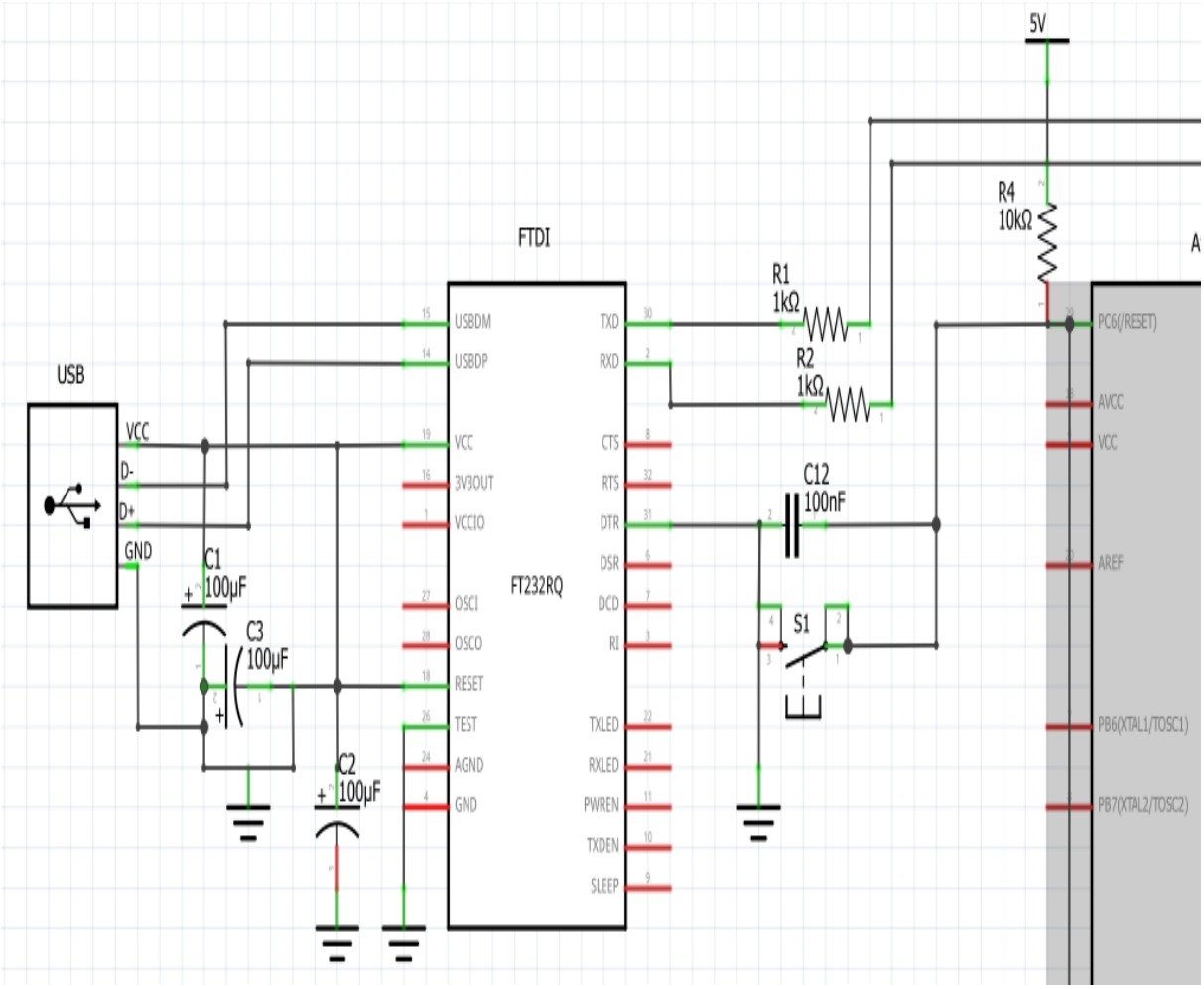


Figure III.3. Schéma électrique de l'interface FTDI

Une puce FT232RL :

C'est une puce FTDI, avec une interface série UART, munie du protocole USB 2.0, elle n'as pas besoin de circuit oscillatoire vu qu'elle est munie d'une horloge interne de 12MHz, ses caractéristiques sont détaillées dans l'annexe.[20]

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

III.1.3. Schéma électrique avec un ATmega8 :

Le deuxième montage repose sur la programmation à partir la communication série SPI à travers un microcontrôleur Atmega8

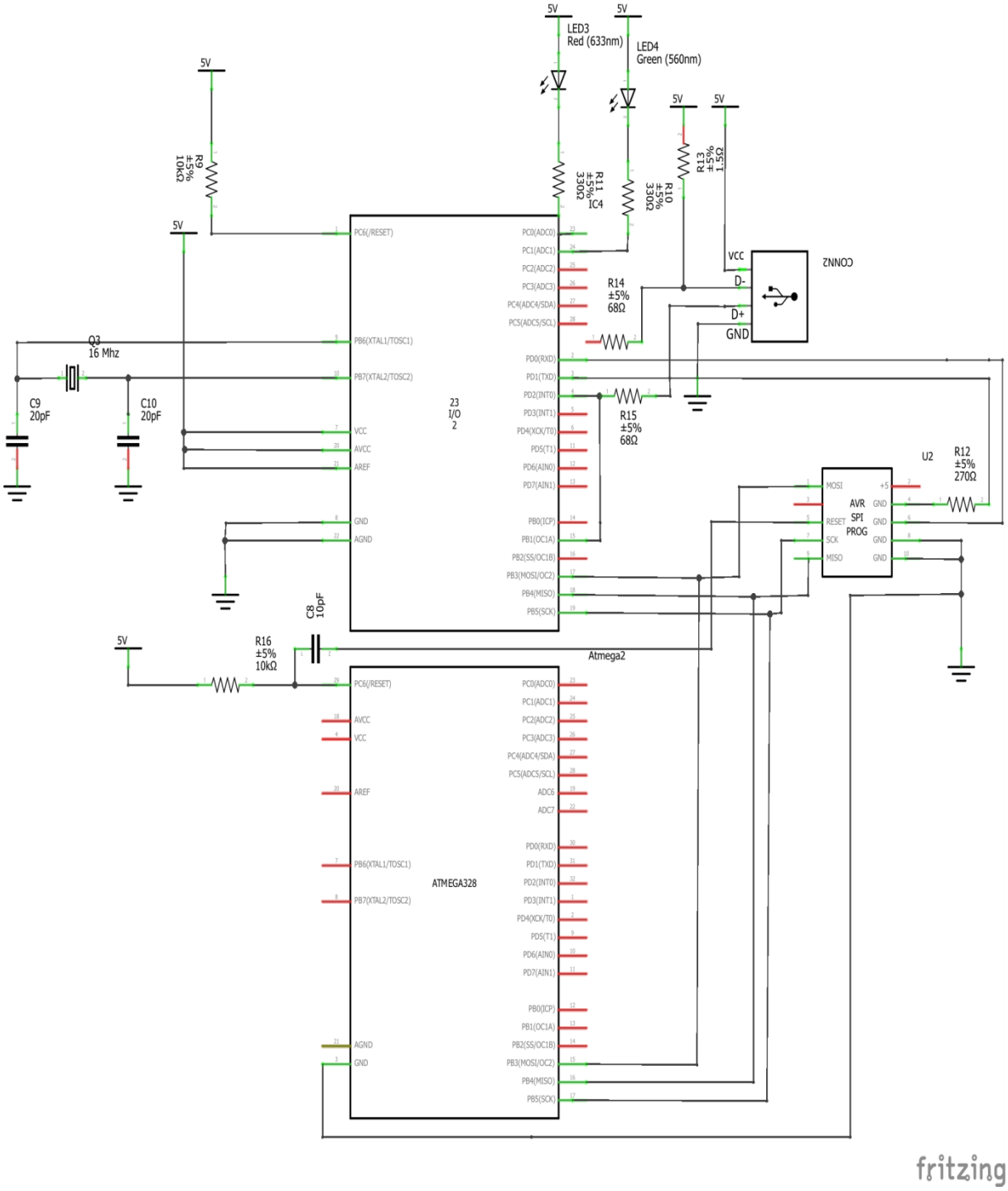


Figure III.4. Schéma électrique du programmateur AVR à base d'Atmega8

Interface USB –SPI :

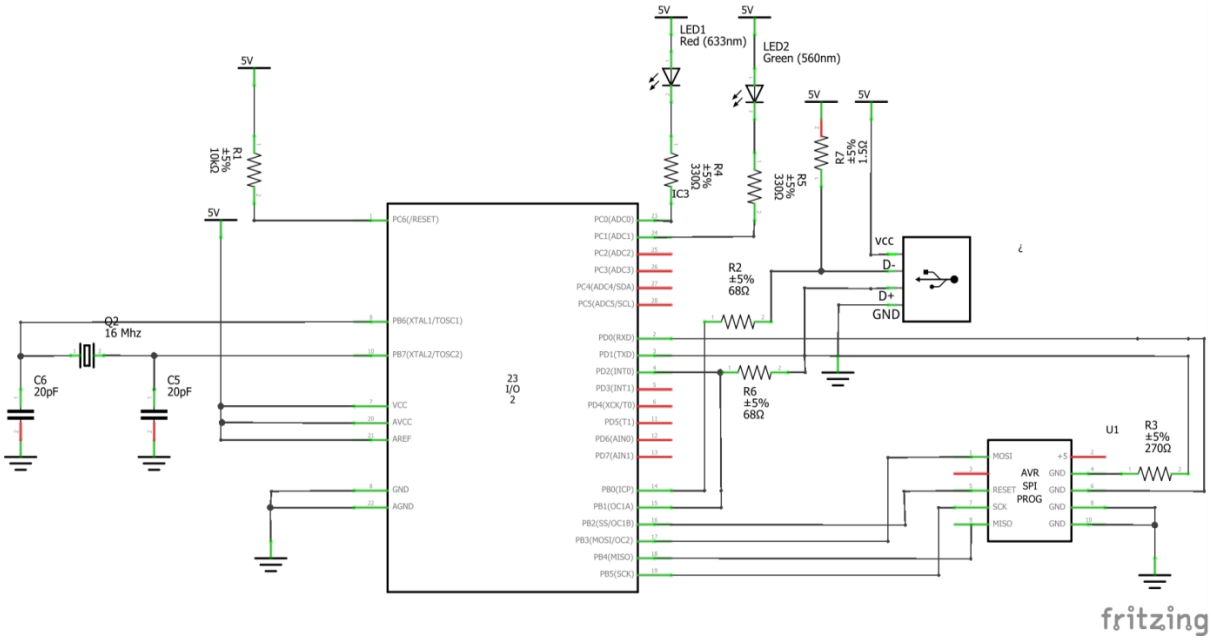


Figure III.5. Schéma électrique d'interface USB-SPI

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 sont reliées aux broches D+ et D- pour le filtrage.

Les résistances R2 et R6 sont des résistances de limitation de courant, qui protègent le port USB de l'ordinateur. La résistance R7 aide l'ordinateur à reconnaître le périphérique comme LS (basse vitesse). Les diodes D1 et D2 indiquent le transfert de données.

Atmega8 : est un microcontrôleur CMOS 8 bits basse consommation basé sur l'architecture AVR RISC, en exécutant des instructions puissantes en un seul cycle d'horloge, l'Atmega8 archive des débits approchant 1MIPS par MHZ, permettant au concepteur du système d'optimiser la consommation d'énergie par rapport à la vitesse de traitement.

ICSP (in-circuit-serial-programming) ou ISP est le nom donne a l'interface de programmation des microcontrôleurs pouvant être reprogrammé sans l'enlever de son circuit.

Les microcontrôleurs AVR peuvent être programmes via leurs ports SPI en jonction avec la broche RESET

Par ailleurs, les broches de programmation ISP dans Amega328P sont les suivantes :

GND = Ground = Masse; MISO = Master In Slave Out (ce signal va de l'Atmega au cible); MOSI = Master Out Slave In (ce signal va du cible à l'Atmega); SCK = Slave Clock (signal d'horloge pour l'Atmega). (Voir photo ci-dessous). Le Master est donc le Programmateur et le Slave est l'Atmega à programmer [20]

III.1.3. Schéma électrique à base d'un transistor :

Le programmateur AVR est connecté à un PC via l'interface série RS232

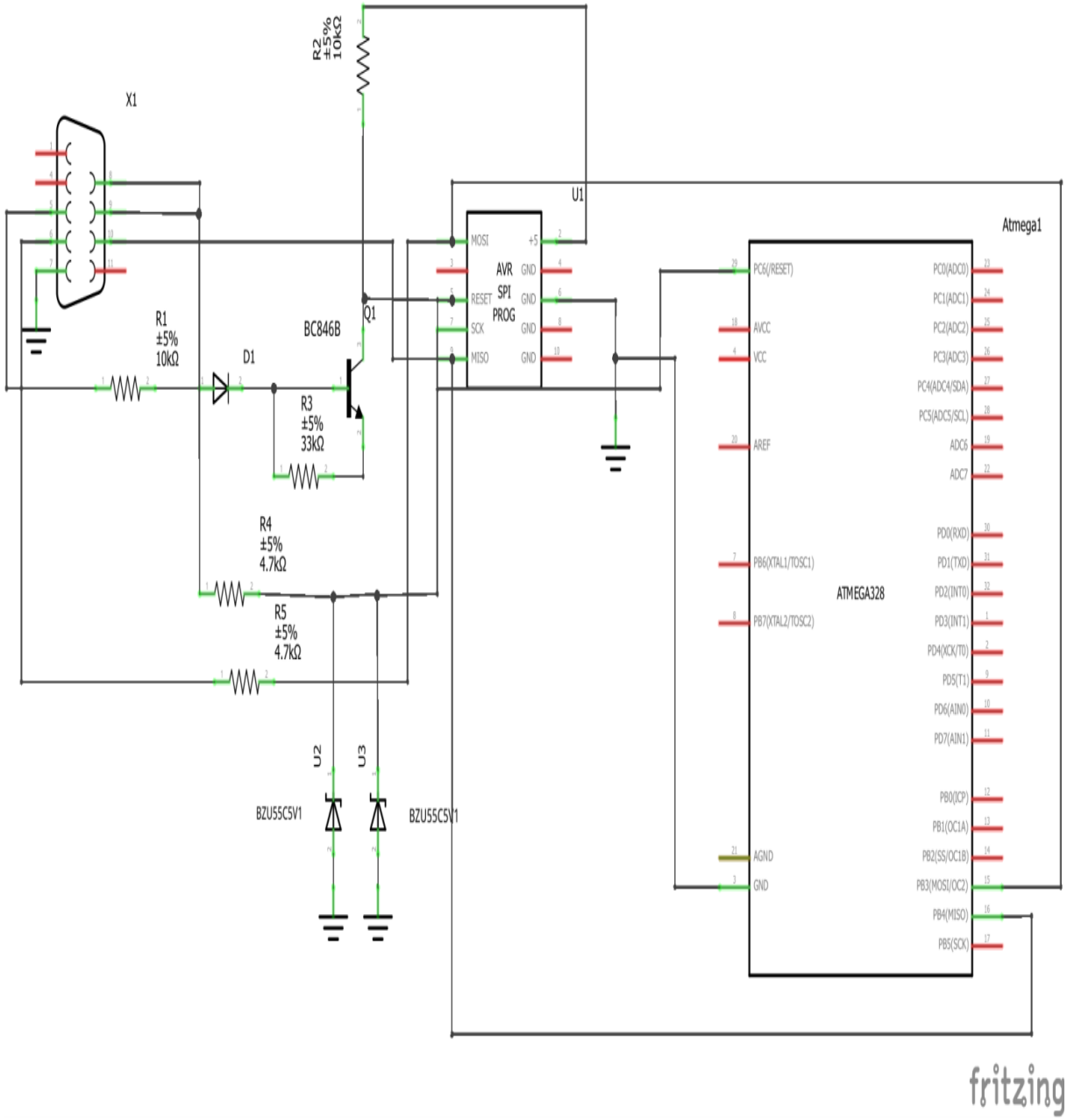


Figure III.6. Schéma électrique du programmateur AVR à base d'un transistor

L'interface série RS232 :

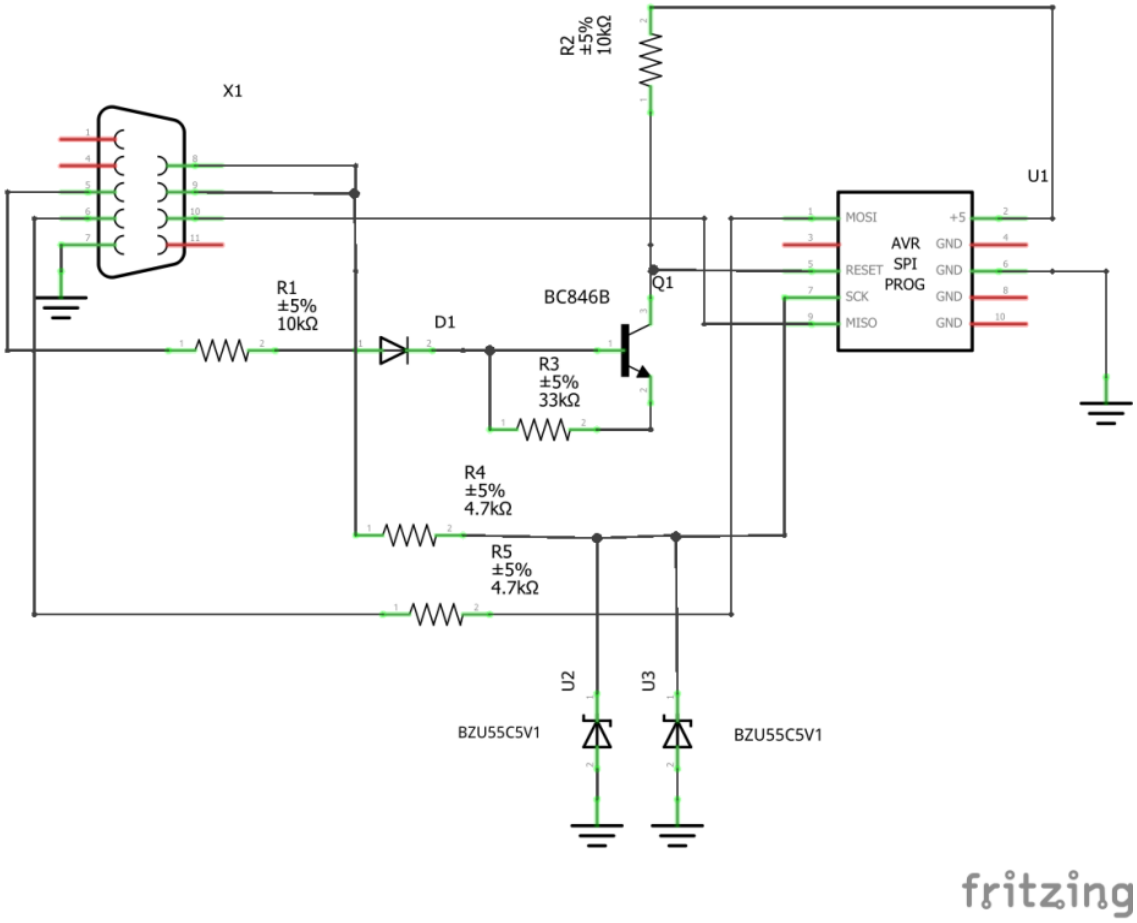


Figure III.7. schéma électrique d'interfaçage série RS232

RS232 :

RS-232 (parfois appelée EIA RS-232, EIA 232 ou TIA 232) est une norme standardisant une voie de communication de type série. Il est communément appelé le « port série ». Sur les systèmes d'exploitation MS-DOS et Windows, les ports RS-232 sont désignés par les noms COM1, COM2, etc. Cela leur a valu le surnom de « ports COM », encore utilisé de nos jours. Il est graduellement remplacé par le port USB depuis l'apparition de ce dernier, et le port RS-232 n'est désormais plus employé que dans des applications professionnelles particulières.

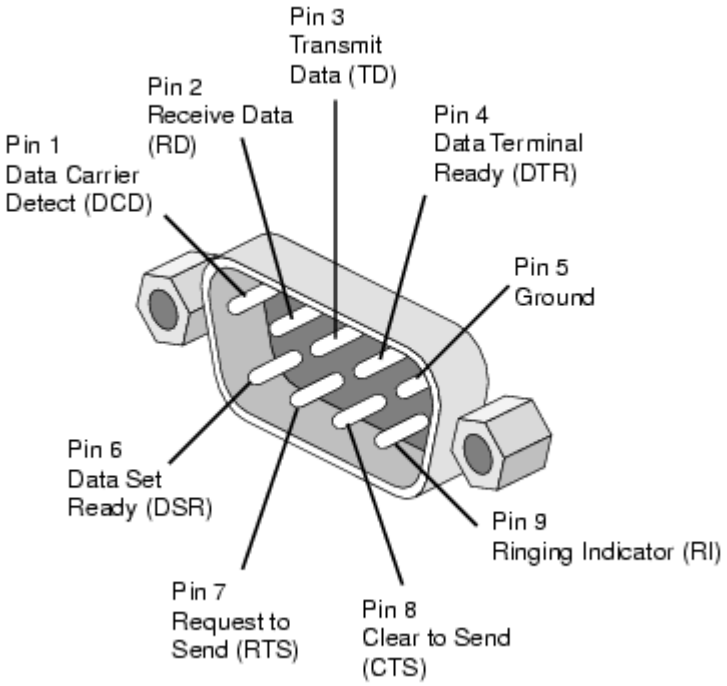


Figure III.8. Brochage du port série RS232

Les diodes Zener D2, D3 avec les résistances R2, R3 réduisent la tension des broches de sortie DTR, RTS sur le port série à environ 5V, ce qui convient au microcontrôleur (MOSI, SCK). Le signal MISO est connecté directement à la broche d'entrée CTS. La diode Zener D1 avec la résistance R1 pilote le transistor NPN T1, qui contrôle le signal RESET. Les microcontrôleurs AVR sont réinitialisés lorsque le signal à un niveau bas. La résistance R5 fonctionne comme un pull-up pour le signal de réinitialisation. La résistance R4 aide à fermer le transistor T1. Le programmeur a un en-tête standard à 10 broches. [21]

III-1-4-3/Circuit de noyau :

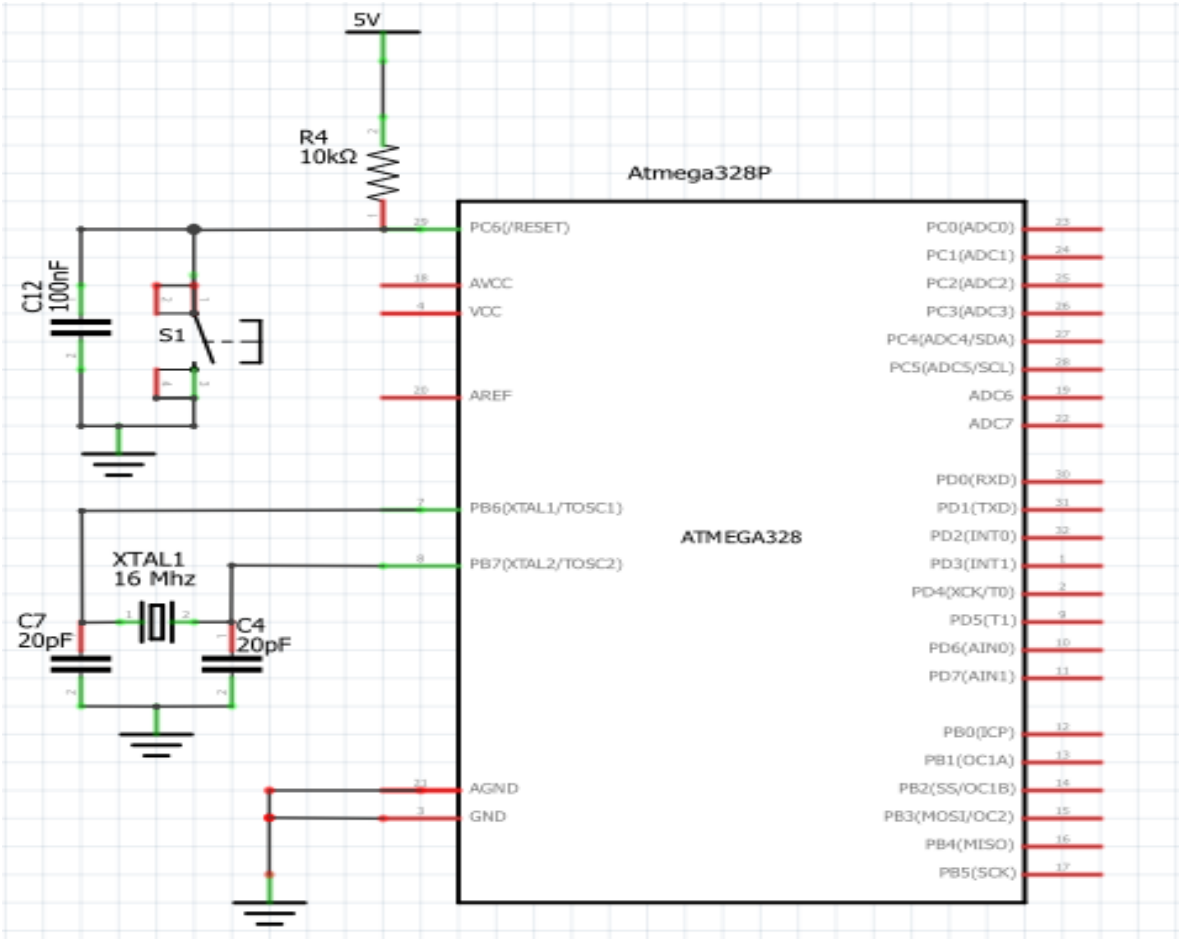


Figure III.9. Schéma électrique de circuit du noyau

Le noyau du programmateur AVR est constitué de :

- **Un microcontrôleur ATMega328P :** nous avons déjà parlé à ce dernier dans le chapitre précédent.

  Maintenant, nous allons expliquer tout ce qui y est lie avec le microcontrôleur ATMega328P :

- **Un oscillateur circuit :** 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 l'ATMega328P, tout microcontrôleur a besoin d'une source d'horloge.

- **Deux condensateurs de 20 pF :** céramique non polarise pour le filtrage et la protection électronique, sont nécessaires pour démarrer l'oscillation et pour maintenir le cristal sur la bonne fréquence.

Les condensateurs ont la propriété de stocker de charge électrique qui sert en cas de coupure brutale d'alimentation.

- **RESET :** suite à une opération de remise à zéro, le microcontrôleur effectue une phase de démarrage :
    1) **RESET :** il peut être déclenché par la mise sous tension du microcontrôleur, la réception d'un signal sur la broche RESET du microcontrôleur, une instruction de RESET
    2) **Initialisation du microcontrôleur :** le microcontrôleur effectue une temporisation afin de garantir la stabilité des signaux d'horloge.
    3) **Effacement des registres :** le microcontrôleur efface le contenu des registres (variable en fonction du « mode de RESET » que vous effectuez).
    4) **Lecture du vecteur RESET** Le microcontrôleur lit l'adresse du programme principal dans la mémoire programme
    5) Début de l'exécution du programme principal.

Maintenant, nous allons lier un circuit avec la broche qui RESET qui assure le fonctionnement de remise à zéro du microcontrôleur ATMega328P.

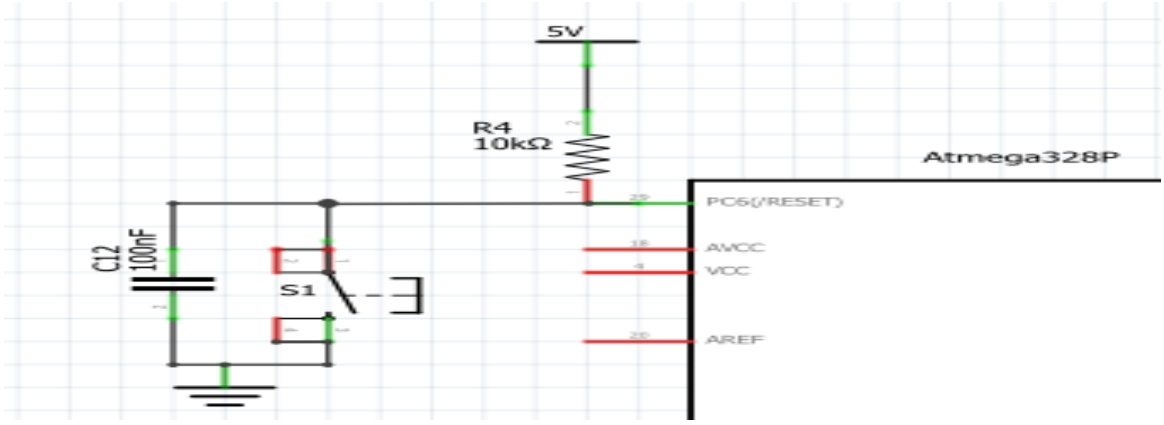


Figure III.10. Brochage du RESET

Une résistance et un condensateur forment un circuit RESET simple et abordable.

Le schéma montre les broches 1 et 2 du commutateur de réinitialisation connectées ensemble (connectées à la terre) et les broches 3/4 connectées ensemble (connectées à RESET).

En pratique, vous avez juste besoin du commutateur pour fonctionner.

La résistance 10K «tire» la broche de réinitialisation vers le haut pendant une activité normale. En tirant la broche de réinitialisation vers le haut, l'ATmega328 fonctionne normalement. Lorsque vous appuyez sur l'interrupteur de réinitialisation (S1), la broche de réinitialisation voit une connexion continue à la terre. Étant donné que la résistance à travers l'interrupteur enfoncé est presque nulle, elle gagne (par rapport à la résistance de la résistance 10K!) Et la broche de réinitialisation est tirée vers le bas, RESET est activé et l'ATmega328 se réinitialise. Relâchez le bouton et la broche de réinitialisation est à nouveau tirée vers le haut et l'ATmega328 sort de la réinitialisation.

La broche RESET de l'ATMega328P doit être connecte à vcc. [22]

III-1-4-1/ Circuit d'alimentation :

VCC et GND ce sont deux connections d'alimentation sur l'ATMega328P

VCC : c'est l'étiquette de la tension positive.

GND : est l'abréviation de Ground. Tout courant électrique besoin d'un moyen de retourner à la terre. Cela peut être appelé <commun> mais est souvent simplement étiquète GND.

ATMega328P alimente par un VCC de valeur 5v et GND. Où puis-je trouve ?

Notre programmateur AVR possède trois possibilités pour l'alimenter :

a. **Alimentation par le port USB :**

L'alimentation de circuit par le port USB est effectuée à partir d'une liaison entre les broches VCC et GND du port USB et les broches VCC et GND du microcontrôleur pour obtenir 5V.

b. Alimentation par le port jack :

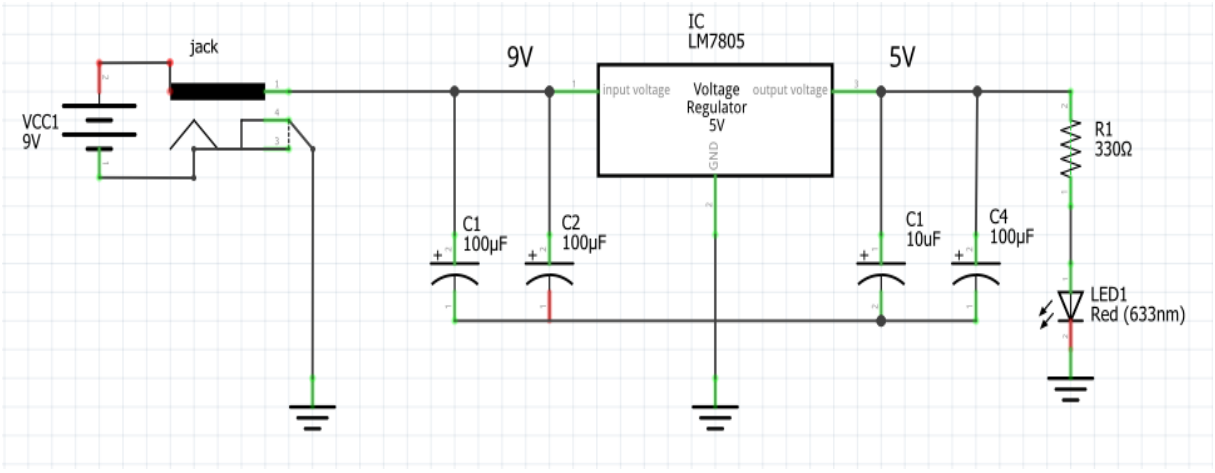


Figure III.11. Schéma de circuit d'alimentation

Port Jack : Prise / connecteur d'alimentation courant continue. Il est commuté pour fonctionner avec des piles.

Un Régulateur 7805 IC fournit une alimentation régulée de +5 volts avec des dispositions pour ajouter un dissipateur thermique.

- Plage de tension d'entrée 7V-35V
- Courant nominal I_c = 1A
- Plage de tension de sortie V_{Max}= 5.2V, V_{Min} = 4.8V

Ce régulateur est muni de deux condensateurs de filtrage :

- **Deux condensateurs céramiques non polarisée 100Nf :** pour éliminer les perturbations haute fréquence.
- **Deux condensateurs électrolytiques 10µF et 100µ :** pour absorber les plus grosses fluctuations. [23]

III-2-Interface de programmation IDE (Software) :

III.2.1. Présentation :

IDE = Integrated Developement Environement, qui est un programme pour écrire le code et il contient un environnement de travail dans lequel les outils de programmation dont vous avez besoin sont tous et chaque IDE est différent de l'autre et c'est ce qui le distingue de tout éditeur de texte régulier qui doit ajouter des extensions supplémentaires pour obtenir ces fonctionnalités et fonctionnalités qui diffèrent d'un IDE à l'autre.

Les outils d'un IDE peuvent être :

- Un éditeur de code intelligent (Coloration, auto complétions, mise en forme) ;
- Un simulateur (logiciel permettant de tester l'exécution de son logiciel) ;
- Un compilateur (qui va transformer le code source rédigé par le développeur en code binaire) ;
- Un débogueur (fonctionnalité d'aide à la correction debug).

Il existe de nombreux IDE.

Certains permettent de développer pour un système d'exploitation spécifique, d'autres sont polyvalents [24].

Qu'est-ce que Bootloader ?

Le Bootloader est un petit morceau de code (code exécutable au format .hex) qui réside dans la mémoire du microcontrôleur. Bootloader dans programmateur AVR SPI nous permet de programmer ce programmeur sur le port série, c'est-à-dire en utilisant un câble sub-d9

Le travail de Bootloader dans le programmateur AVR SPI est d'accepter le code de l'ordinateur et de le placer dans la mémoire du microcontrôleur.

Pourquoi avons-nous besoin d'un bootloader ?

Traditionnellement, les microcontrôleurs comme Atmega328 d'Atmel sont programmés à l'aide de programmeurs dédiés, ce qui implique des connexions sophistiquées. Les

bootloaders éliminent cette complexité et nous offrent un moyen simple de programmer le microcontrôleur, c'est-à-dire simplement en utilisant un connecteur sub-d9.

Les chargeurs de démarrage résident dans un emplacement sécurisé spécial de la mémoire flash programmable du microcontrôleur et occupent généralement moins de 1 Ko de mémoire.

Quel est le besoin de Burning Bootloader sur Atmega328 ?

Comme je l'ai mentionné plus tôt, si vous souhaitez télécharger des programmes sur un tout nouveau microcontrôleur Atmega328, vous devez utiliser un programmeur spécial (et également définir les bits de fusible). Mais si vous gravez Bootloader sur Atmega328, vous pouvez simplement programmer le microcontrôleur sur le port série.
Une fois que le microcontrôleur Atmega328 est prêt avec le chargeur de démarrage, vous pouvez simplement l'utiliser dans votre programmateur (en remplacement) ou l'utiliser comme microcontrôleur si vous envisagez de créer votre propre carte Arduino.

Comment graver Bootloader sur Atmega328P ?

Il existe deux façons de graver le chargeur de démarrage sur Atmega328 IC. La première consiste à utiliser un matériel de programmation AVR dédié. La deuxième façon consiste à utiliser une carte Arduino fonctionnelle comme programmeur et à graver le chargeur de démarrage sur le microcontrôleur Atmega328 cible.
Dans ce projet, nous utiliserons la première méthode, c'est-à-dire utiliser un programmateur avr Spi. [25]
Maintenant, nous avons besoin d'un logiciel pour programmer les microcontrôleurs :

III.2.2. Le logiciel de programmation Pony Prog :

PonyProg est un *logiciel de* programmation de périphériques série avec un cadre GUI convivial disponible pour Windows95 / 98 / ME / NT / 2000 / XP et Intel Linux. Son but est de lire et d'écrire chaque périphérique série. Pour le moment, il prend en charge I²C Bus, Microwire, SPI EEPROM, Atmel AVR et Microchip PIC micro.
SI-Prog est l'interface *matérielle* du programmeur pour PonyProg.
Avec PonyProg et SI-Prog, vous pouvez programmer Wafercard pour SAT, EEPROM dans

GSM, TV ou CAR-RADIO. En outre, il peut être utilisé comme kit de démarrage à faible coût pour PIC et AVR.

PonyProg fonctionne également avec d'autres interfaces matérielles simples comme AVR ISP (STK200 / 300), JDM / Ludipipo , EasyI2C et DT-006 AVR (par Dontronics).

Notre but dans ce projet est la programmation des AVRs ATMega328P. Alors la pony prog est le logiciel convenable pour faire cette opération à partir de ses caractéristiques :

- Prend en charge le microcontrôleur AVR Atmega103, Atmega161, Atmega163, Atmega 323, Atmega128, Atmega8, Atmega16, Atmega64, Atmega32, Atmega162, Atmega169, Atmega8515, Atmega8535
- Écriture de bits de verrouillage pour protéger le micro AVR de la lecture
- Écrivez à la fois la mémoire Flash et EEPROM du micro AVR
- Edition des bits de sécurité pour AVR.
- Bouton Recharger le fichier
- Commande de remplissage du tampon
- Programmation du numéro de série
- Fichiers de script pour la programmation par lots
- Vitesse améliorée avec WinNT / 2000 / XP avec un pilote pour les E / S directes [26].

Le programmeur AVR ISP est maintenant prêt, donc on va lancer le logiciel ponyprog comme la figure montre :

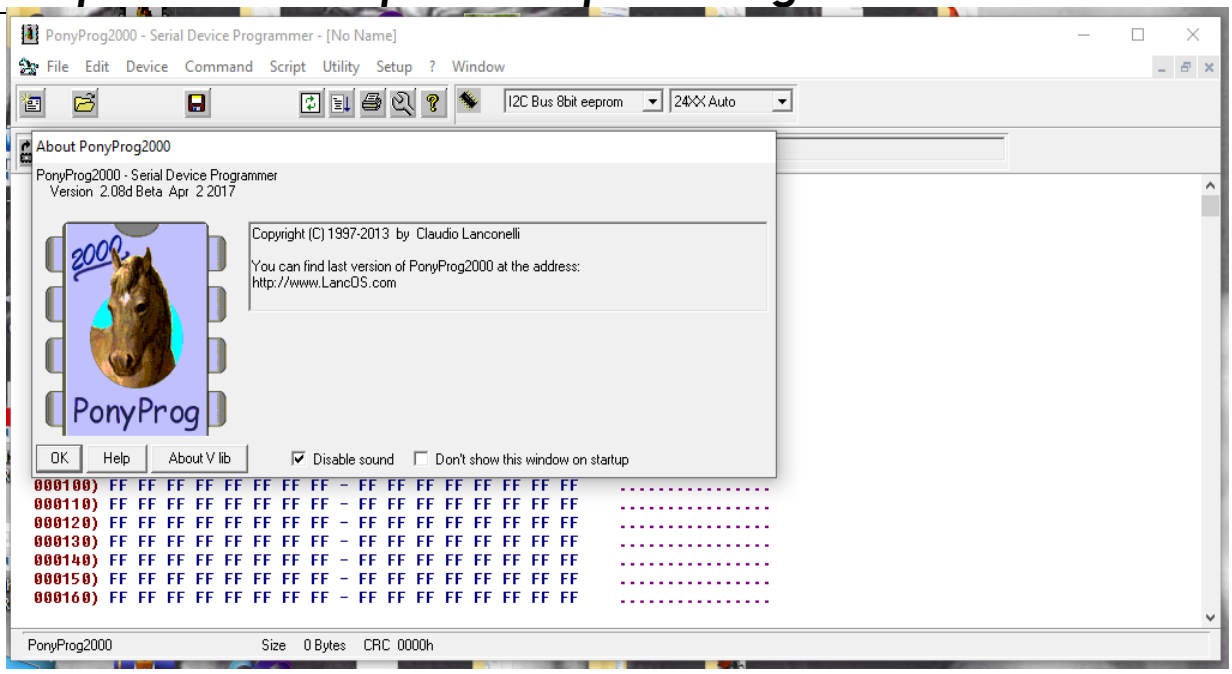


Figure III.12. Fenêtre d'accueil de Ponyprog

« Ponyprog » est lancé.

On clique sur l'icône « device » puis sélectionner micro AVR

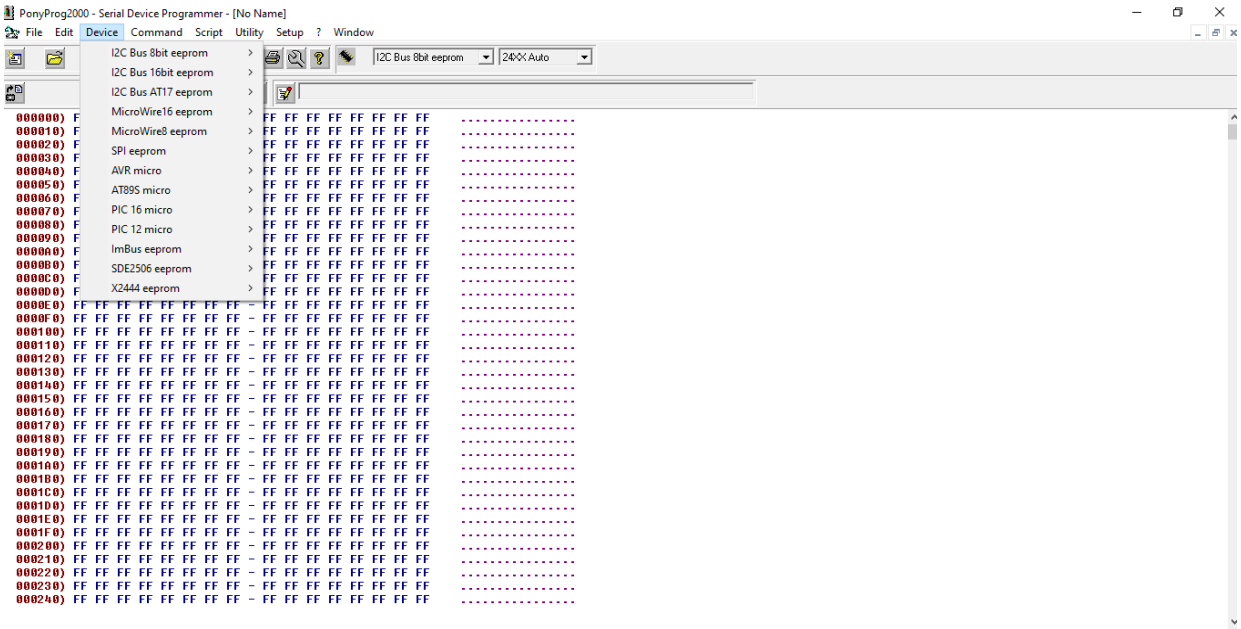
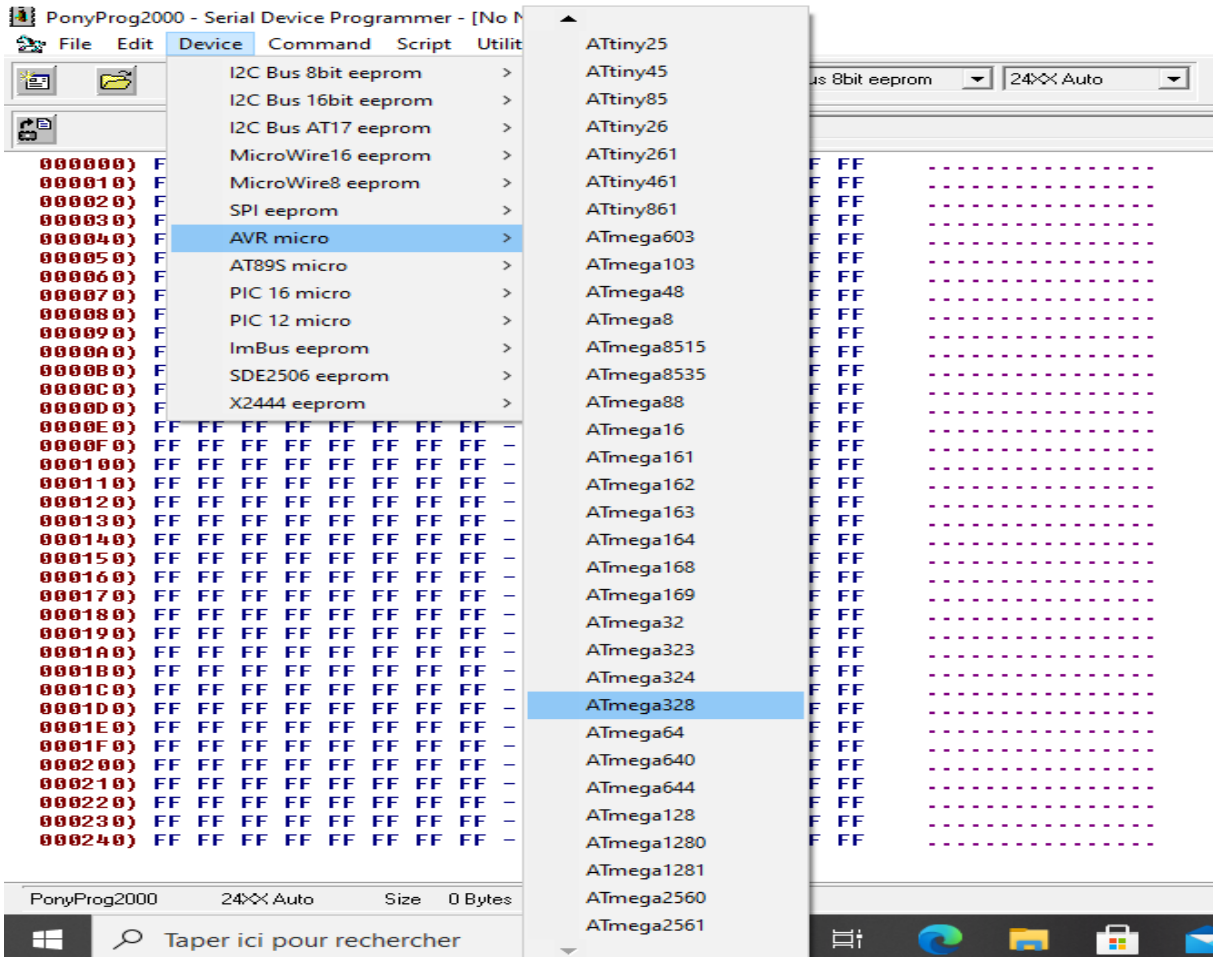
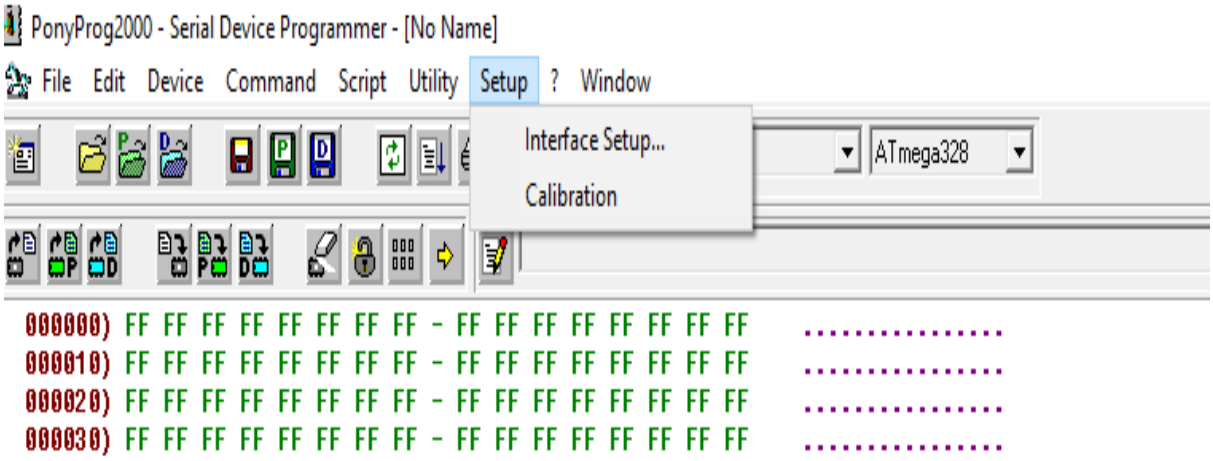


Figure III.13. Le choix de MCU

Maintenant, nous choisissons le type de microcontrôleurs que nous voulons programmées



Tout d'abord, les ports de connexion et le type du programmeur doivent être définis.



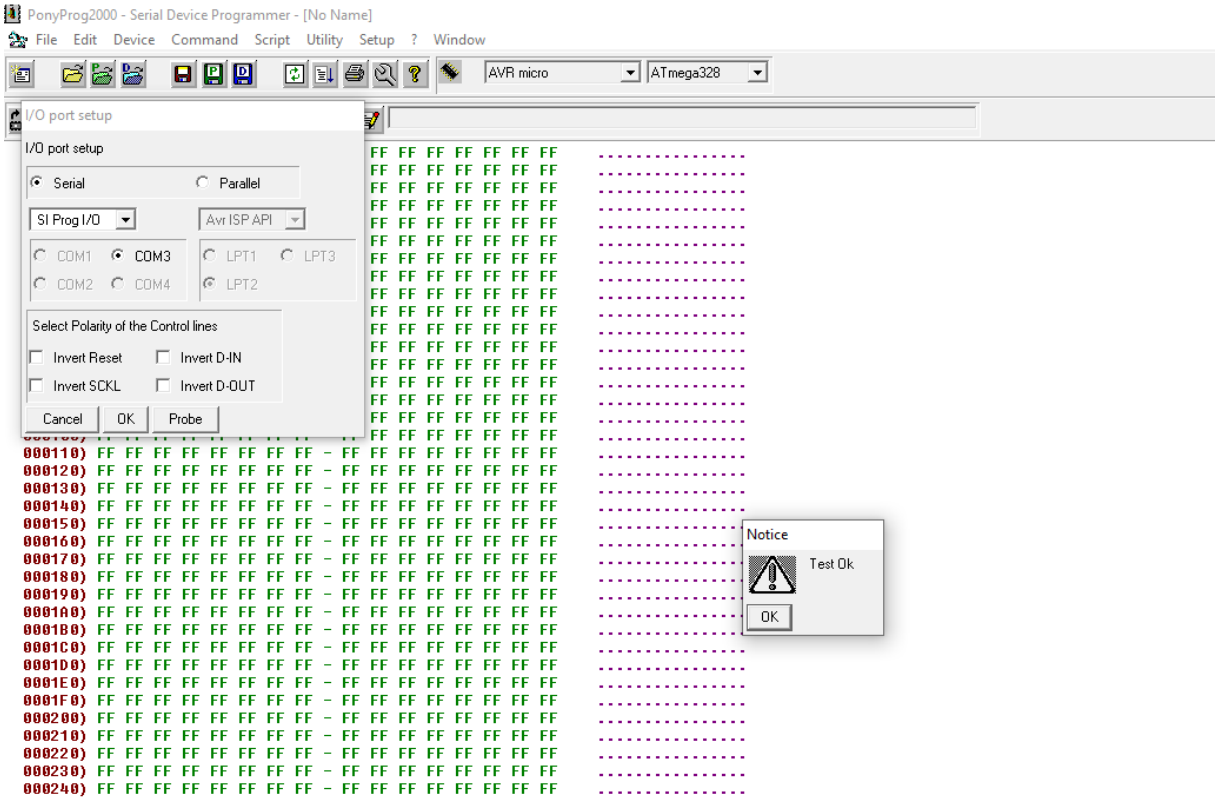
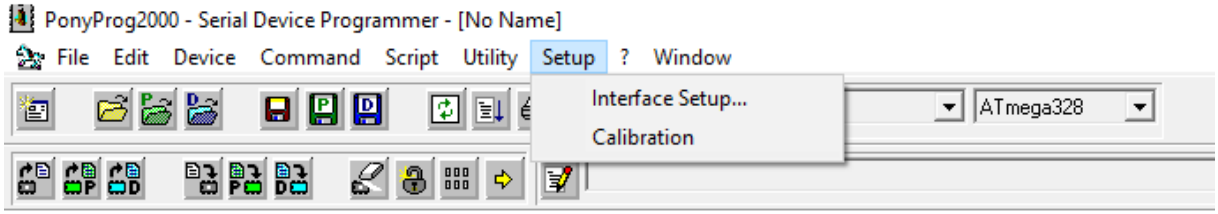
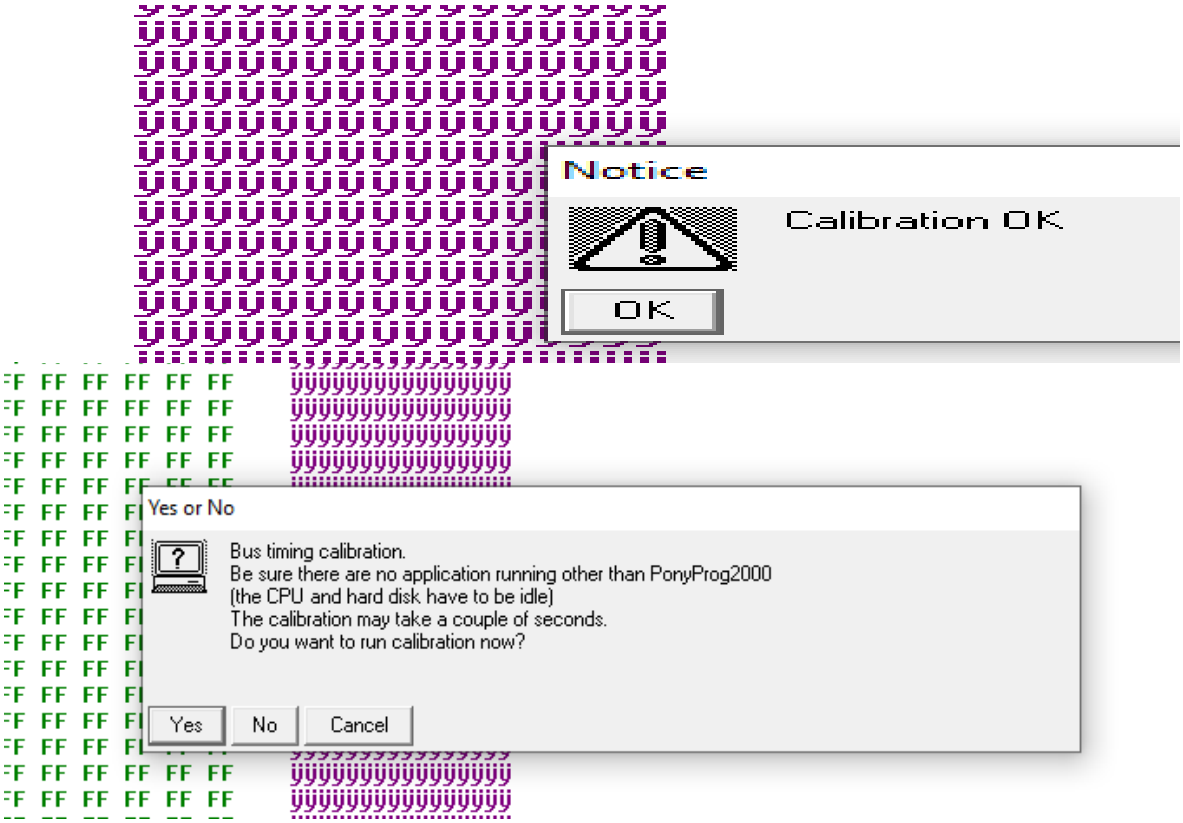


Figure III.14. Test de communication PC et Interface

Ainsi, l'étalonnage doit être effectué après une clique sur l'icône « calibration »

Selon la figure suivante





Après avoir assemblé le fil de pain et fourni la puce, il est temps de commencer le test.

Le programmateur est inséré dans le port RS232

"Ponyprog" est lancé.

Tout d'abord, les ports de connexion et le type du programmeur doivent être définis

III.2.3. Le logiciel mikroc for AVR :

Le mikroC PRO pour AVR est un outil de développement puissant et riche en fonctionnalités pour les microcontrôleurs AVR. Il est conçu pour fournir au programmeur la solution la plus simple possible pour développer des applications pour les systèmes embarqués, sans compromettre les performances ou le contrôle. [27]

mikroC PRO for AVR vous permet de développer et de déployer rapidement des applications complexes:

- Écrivez votre code source C à l'aide de l'éditeur de code intégré (assistants de code et de paramètres, pliage de code, mise en évidence de la syntaxe, correction automatique, modèles de code, etc.)
- Utilisez le mikroC PRO inclus pour les bibliothèques AVR pour accélérer considérablement le développement: acquisition de données, mémoire, affichages, conversions, communication, etc.
- Surveillez la structure, les variables et les fonctions de votre programme dans l'explorateur de code.
- Générez un assemblage commenté, lisible par l'homme et un format HEX standard compatible avec tous les programmeurs.
- Inspectez le déroulement du programme et déboguez la logique exécutable avec le simulateur de logiciel intégré.
- Générez un fichier COFF (Common Object File Format) pour le débogage logiciel et matériel sous AVR Studio®.
- Utilisez l'optimisation de l'assignation statique unique pour réduire votre code à une taille encore plus petite.
- Obtenez des rapports et des graphiques détaillés: carte de la RAM et de la ROM, statistiques du code, liste des assemblages, arborescence des appels, etc.
- Les commentaires actifs vous permettent de rendre vos commentaires vivants et interactifs.

- mikroC PRO for AVR fournit de nombreux exemples à étendre, à développer et à utiliser comme briques de construction dans vos projets. Copiez-les entièrement si vous le souhaitez - c'est pourquoi nous les avons inclus dans le compilateur

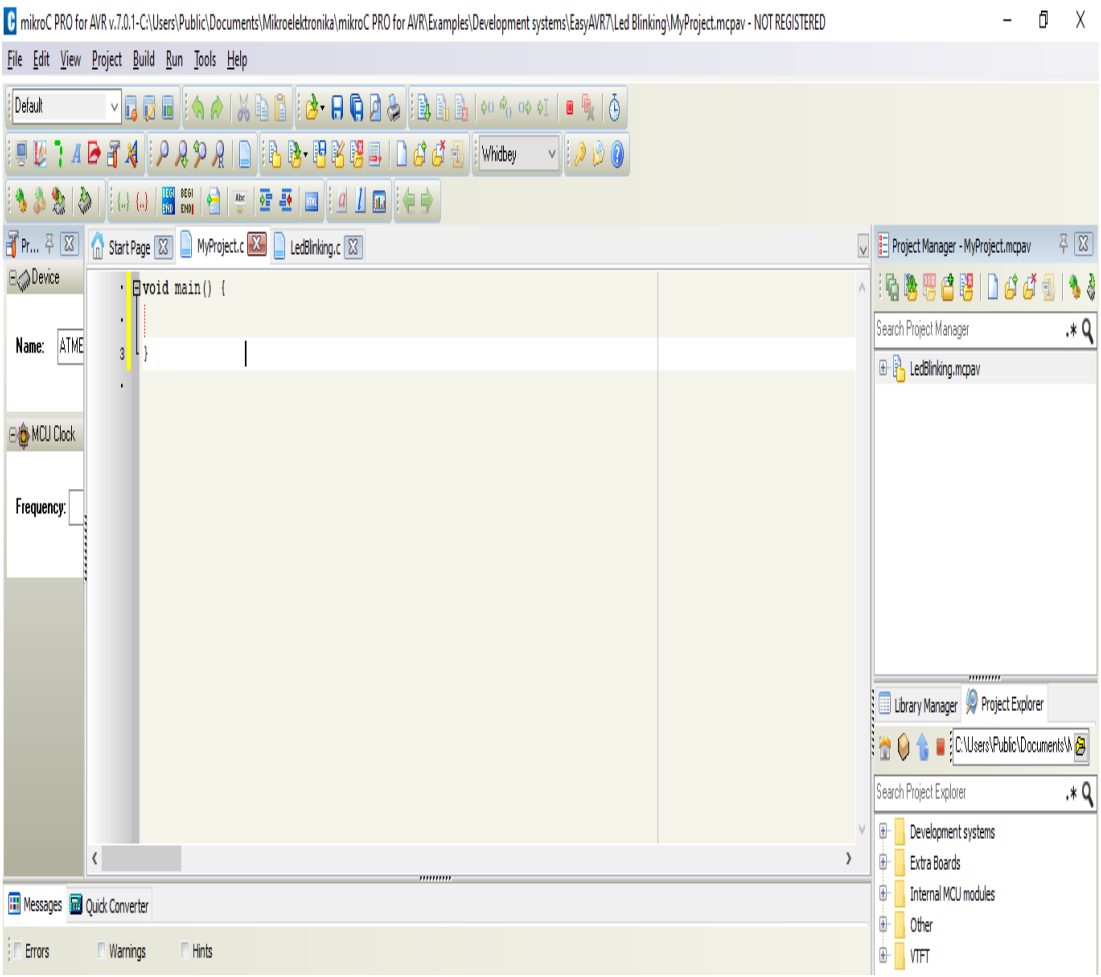
Figure III.15. La fenêtre d'accueil du logiciel mikroc for AVR

III.2.4. Proteus

Cette suite logicielle est très connue dans le domaine de l'électronique. De nombreuses entreprises et organismes de formation (incluant lycée et université) utilisent cette suite logicielle. Outre la popularité de l'outil, Proteus possède d'autres avantages

Pack contenant des logiciels facile et rapide à comprendre et utiliser

Le support technique est performant



L'outil de création de prototype virtuel permet de réduire les coûts matériel et logiciel lors de la conception d'un projet

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.

Sur Proteus, nous avons réalisé un petit circuit qui illustre l'utilisation l'Atmega328P programmé, la manière de faire le branchement, le téléversement, et enfin la simulation. Le circuit est composé de microcontrôleur Atmega328P, une LED munie d'une résistance de 1KΩ branchée entre broche 15 de l'ATMega328P et GND broches 8 et 22 de l'ATMega328P [28].

III.2.5. Schéma de simulation pour l'exemple Blink :

Tout d'abord, on dessine le circuit sur Proteus ISIS

Voici donc à quoi ressemble le circuit sur Proteus ISIS :

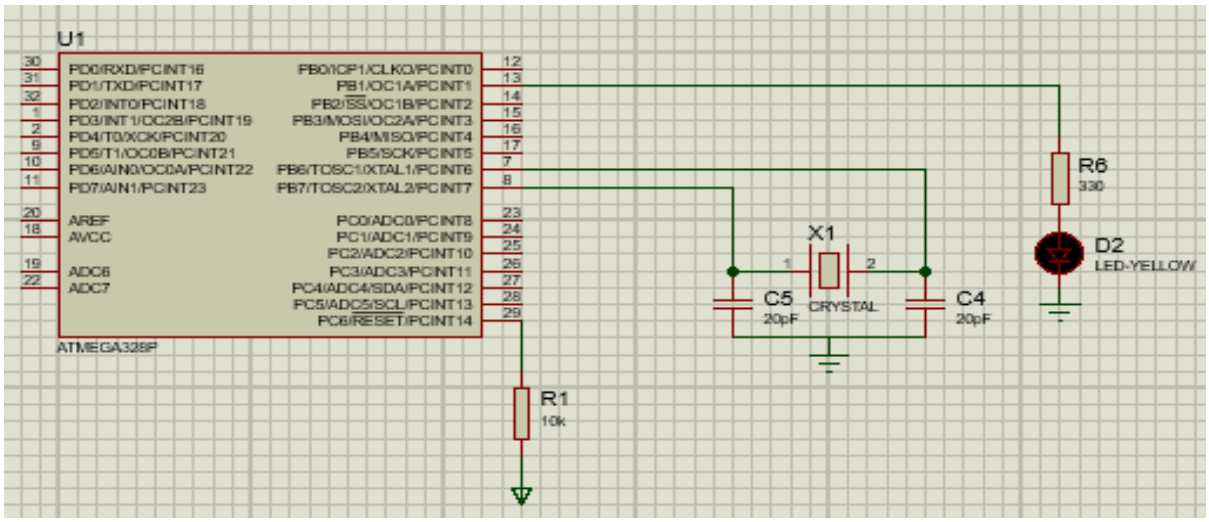


Figure III.16. Le circuit sur Proteus ISIS

Après, écrire le code source C à l'aide de l'éditeur de code intégré sur mikroc for AVR, et le compiler comme il est indiqué dans la figure qui suit

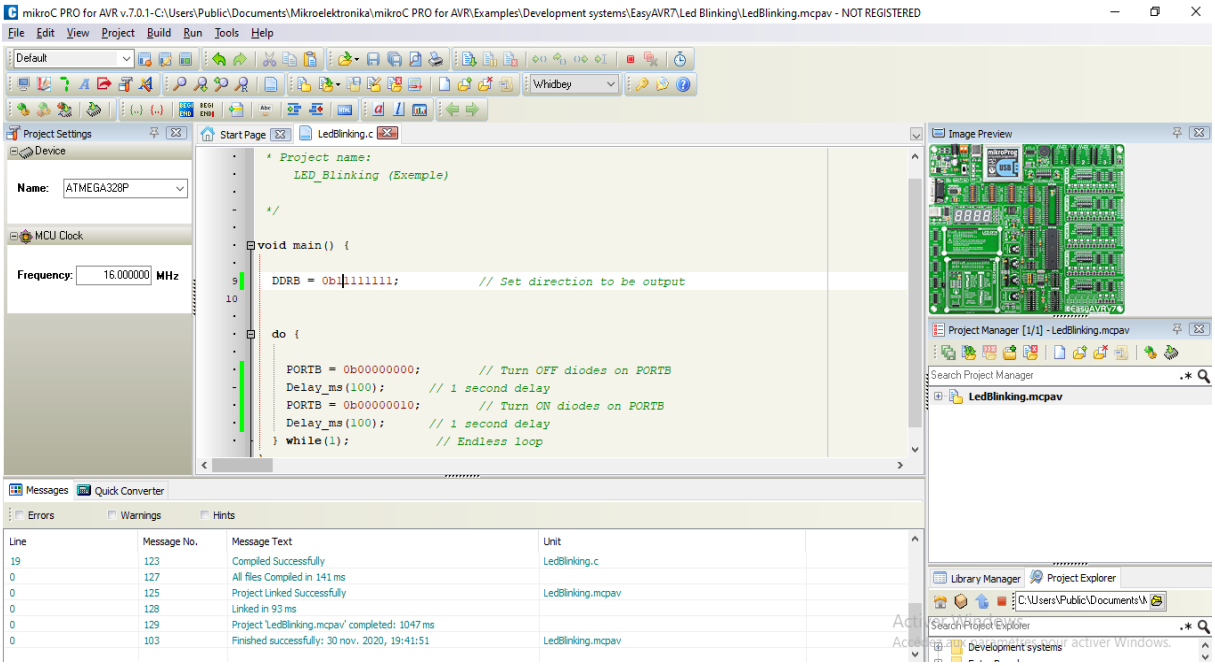


Figure III.17. Le circuit sur mikroc for AVR

Ensuite, ajouter le fichier .hex de l'exemple sur le microcontrôleur Atmega328P comme la figureIII.16 montre :

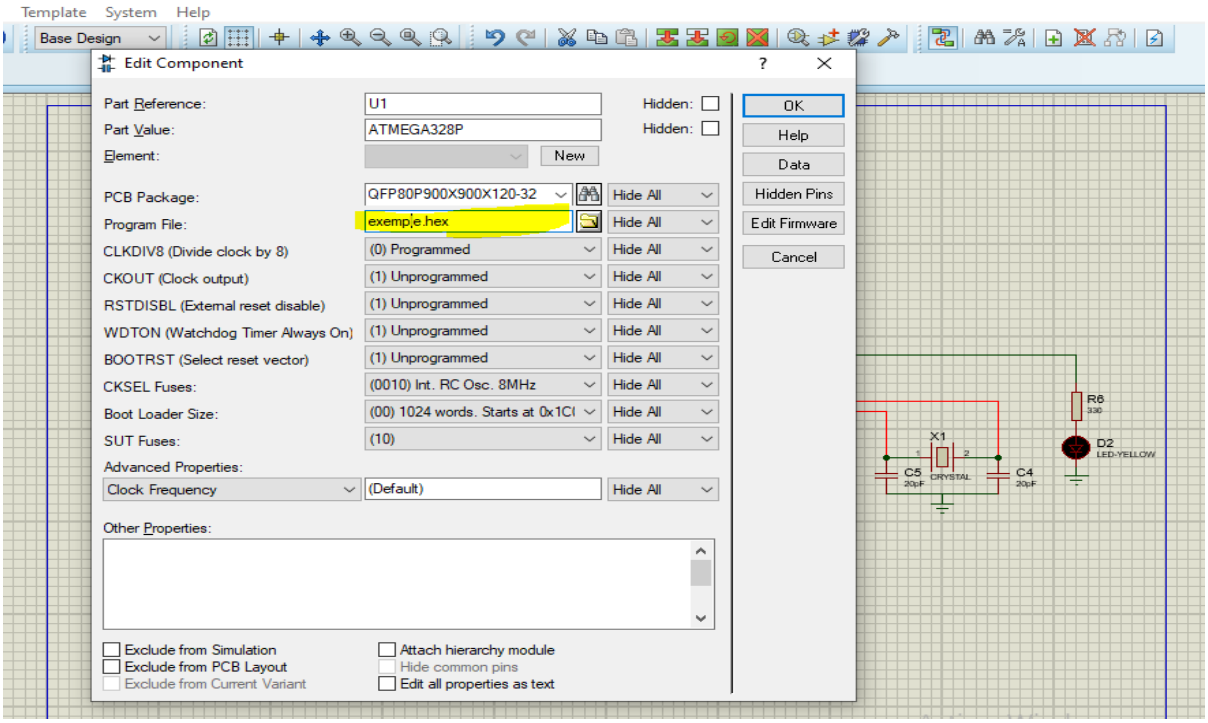


Figure III.18. Téléchargement de fichier hex

Après les étapes précédentes on lancera la simulation pour arriver au résultat de programmation

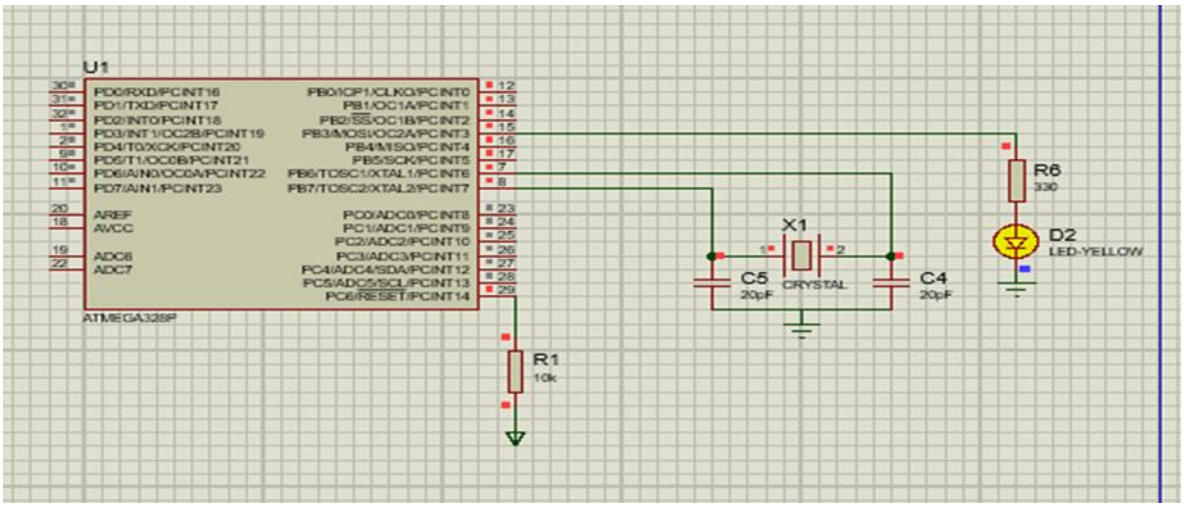


Figure III.19. Simulation réussite led clignote

Après la réussite de simulation on va passer maintenant à le téléverssement du code vers le microcontrôleur Atmega328P à partir de logiciel ponyprog afin de confirmer la communication entre le pc et le montage (software et hardware)

Voici la figure suivante indique la communication entre eux.

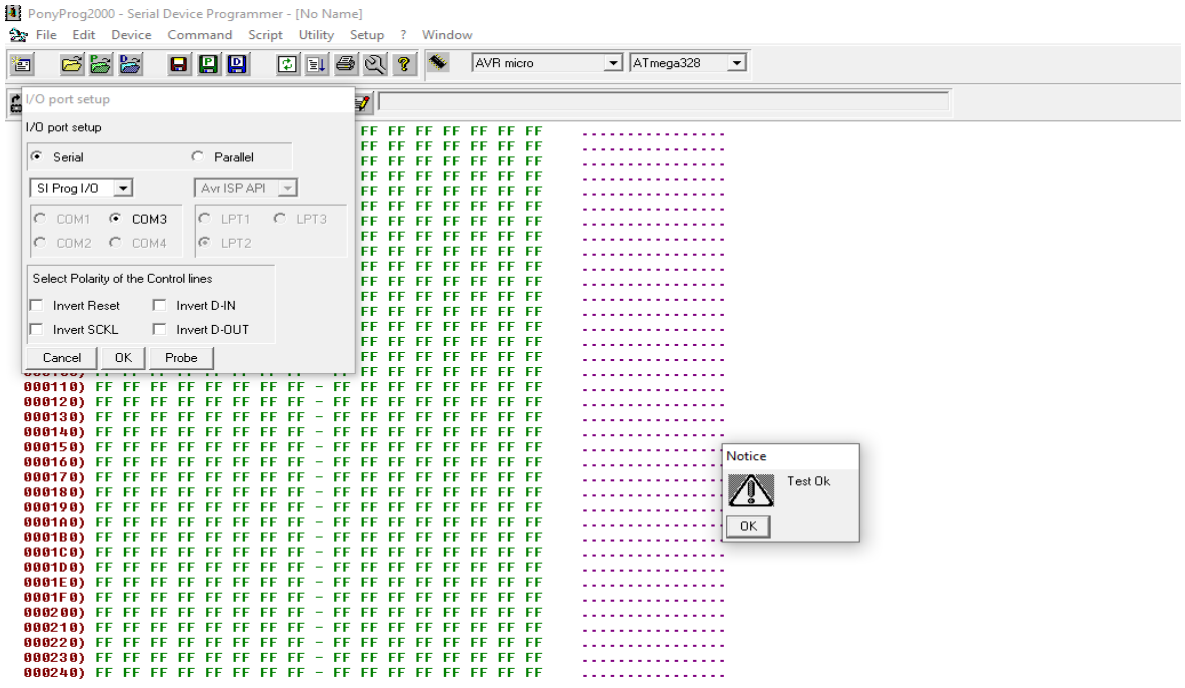
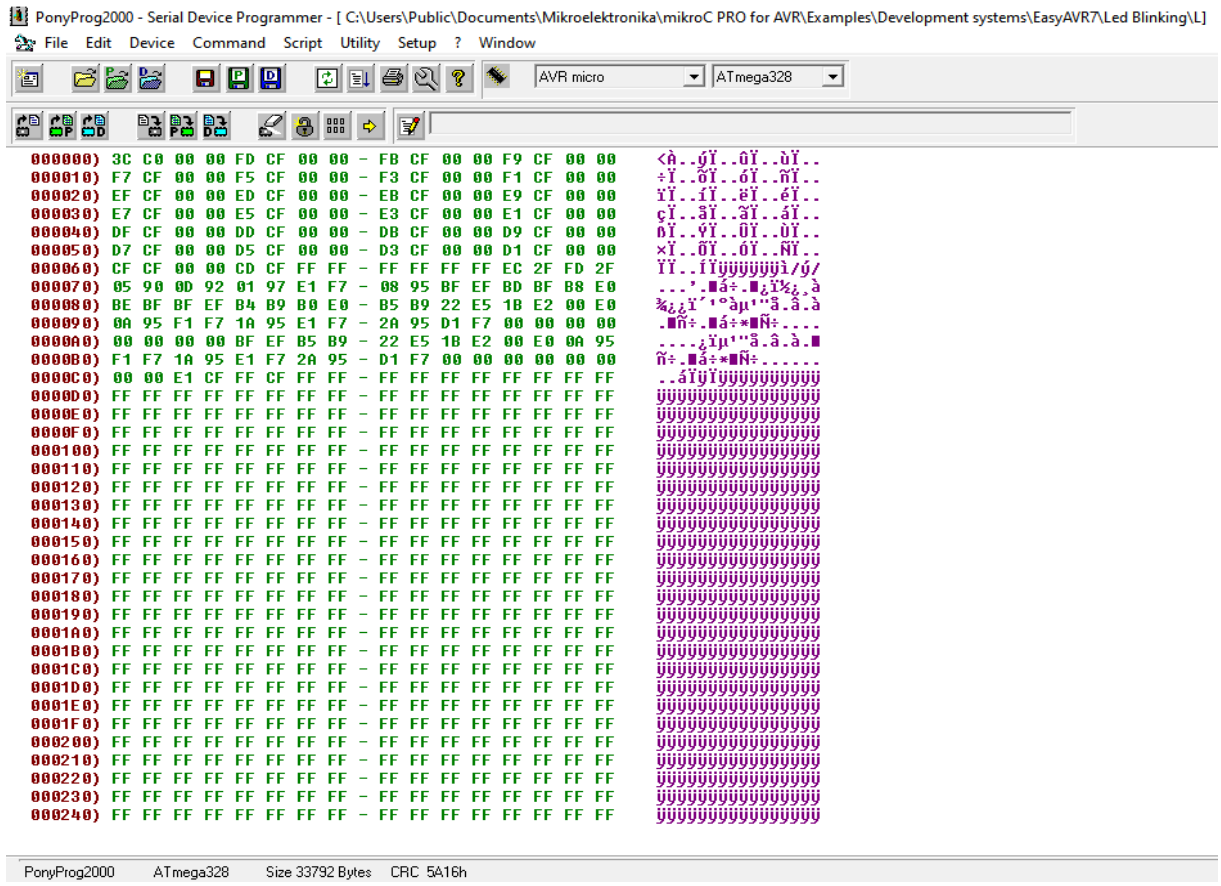
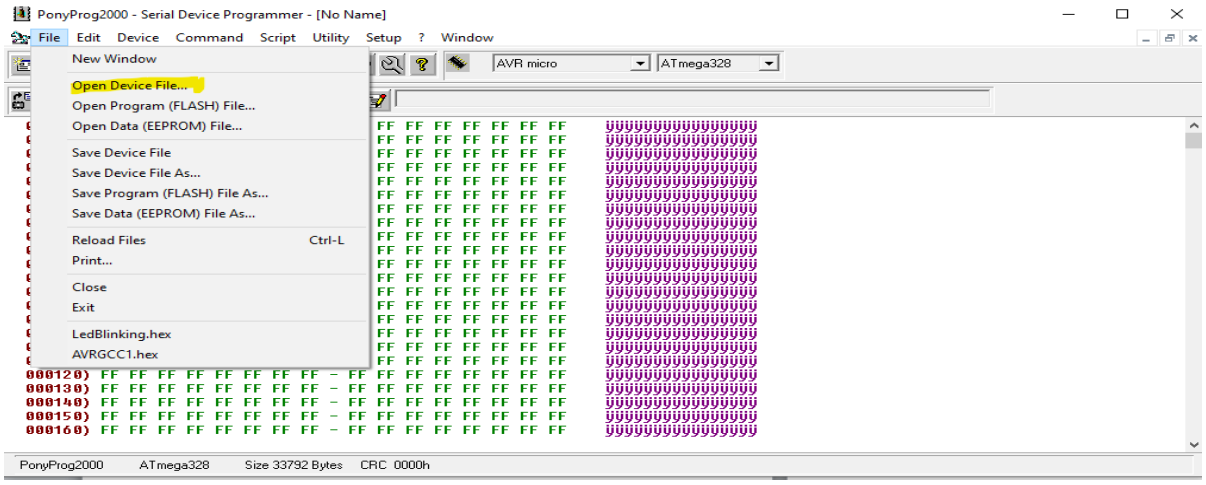


Figure III.20. Communication confirmée

Test OK c'est-à-dire que la communication est confirmé.
Maintenant, on ajoute le fichier .hex



Après l'ajout du fichier .hex, la fenêtre suivante s'affiche.



Ensuite, on passera au téléverssement vers le Atmega328P comme ci-dessus :

Figure III.21.le téléversseement du programme

Ce téléversseement va être appliqué en réelle dans le chapitre qui suit.

Conclusion

Ce chapitre nous montre de quoi sont composés les deux piliers de l'univers programmateur, Le Hardware et le Software. En gros, le Hardware est composé d'un ensemble de compartiments essentiels à son fonctionnement, un bloc d'alimentation, un noyau basé sur la technologie Atmel, des périphériques qui font le relais entre le programmateur et les cartes interfaces, et une interface de communication pour la connexion avec la partie Software, cette dernière fonctionne sous Windows, Linux et Mac.

Dans le chapitre suivant on va appliquer ce qu'on a vu dans la théorie.

Chapitre IV : Réalisation et Tests

Introduction

Après avoir détaillé dans le chapitre précédent la partie hardware et software AVR. On passe dans ce chapitre à la réalisation des différents montages du programmateur obtenu dans l'étude théorique. Puis nous allons faire des essais, après on a les tests pour interpréter les résultats (entre l'étude théorique et la pratique)

Donc, en premier lieu illustrer quelques photos des montages réalisées et testées du circuit de programmateur AVR.

Trois compartiments du programmateur AVR assurent son fonctionnement, on distingue, un circuit d'alimentation qui est un assemblage de régulateur et de condensateur, dont le rôle est de régulariser et sélectionner l'alimentation, puis une interface de communication série USB ou série RS232 qui assure la connexion avec le PC, et un noyau à base d'un microcontrôleur ATMega328P, ou s'effectue le traitement des programmes.

IV.1. Réalisation et test sur plaque d'essais :

IV.1.1. Réalisation du circuit d'alimentation :

IV.1.1.1. Circuit du régulateur de tension L7805 :

Le L7805 c'est un régulateur qui fournit une tension de 5V à partir d'une certaine tension d'alimentation.

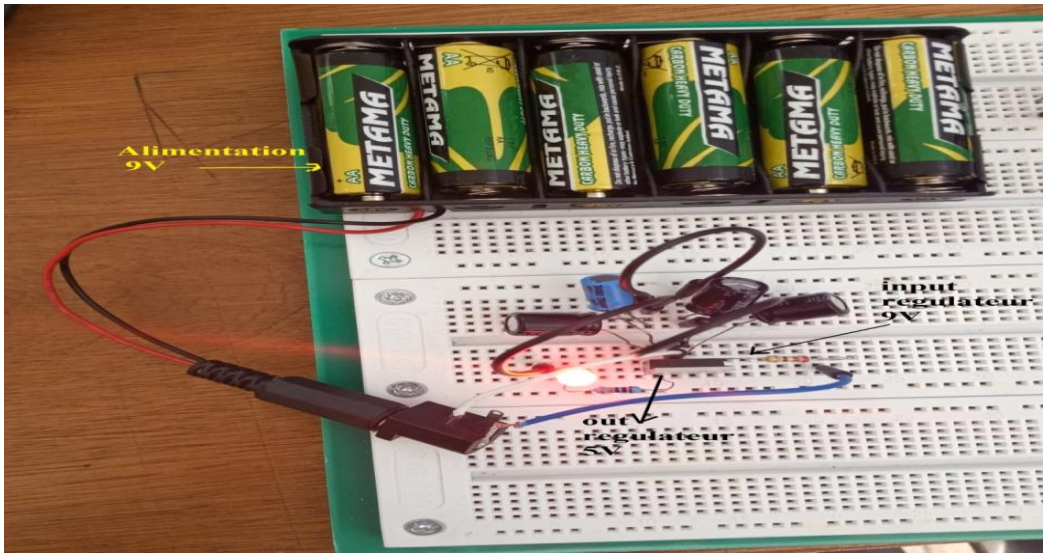


Figure IV.1. Le montage d'alimentation

Le régulateur L7805 possède trois électrodes : l'entrée Vin, la sortie V .out ainsi que la masse GND. L'électrode Vin est branchée à la source, l'électrode GND est reliée à la masse et l'électrode V out est la sortie.

Maintenant, on passe au test du circuit :

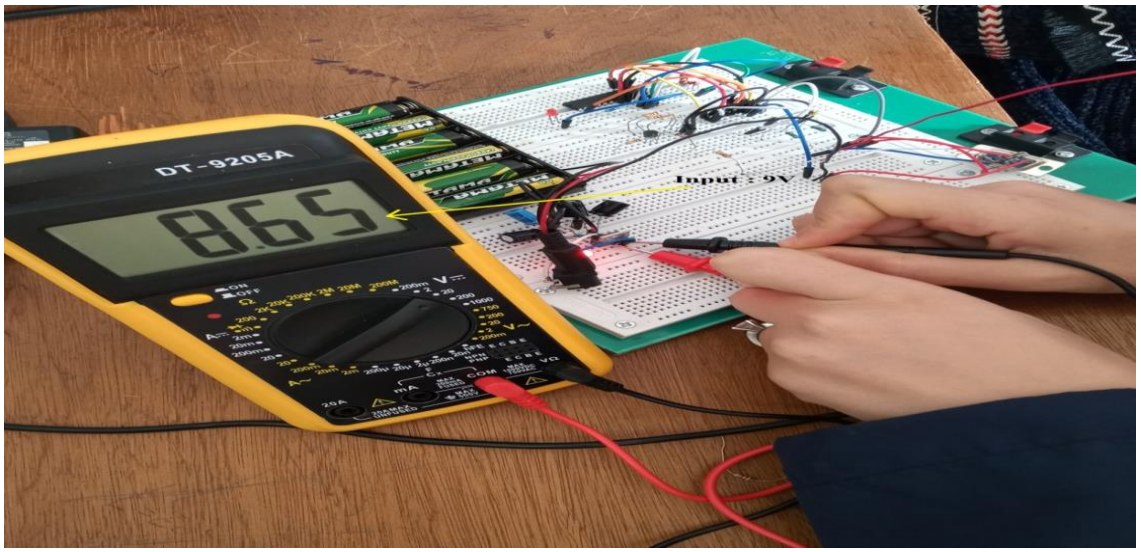


Figure IV.2. Test d'alimentation d'entrée

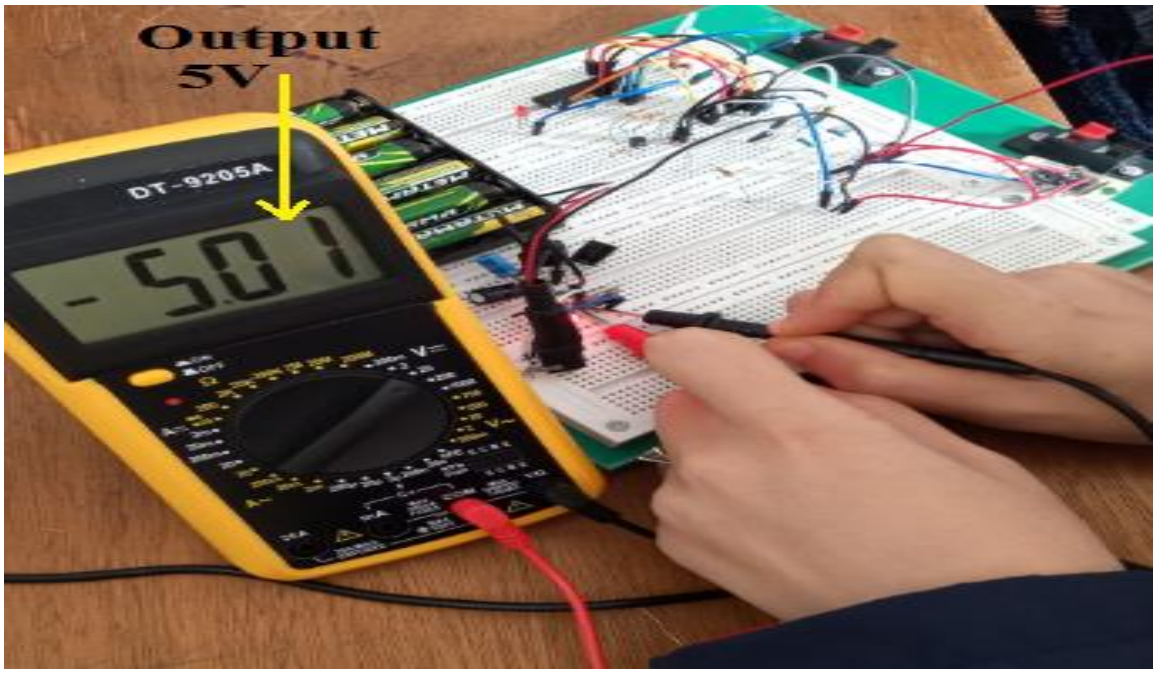


Figure IV.3. Test d'alimentation de sortie

IV.1.2.1. Programmateur à base d'un adaptateur FTDI :

Réalisation du circuit de l'interface série (USB)

Le programmateur AVR UART possède une interface série USB munie du microcontrôleur ATMega16U2, Le circuit de l'interface série USB est identique à celui de l'adaptateur série USB FTDI, nous l'avons réalisé sur la plaque d'essais et le tester avec l'IDE Arduino,

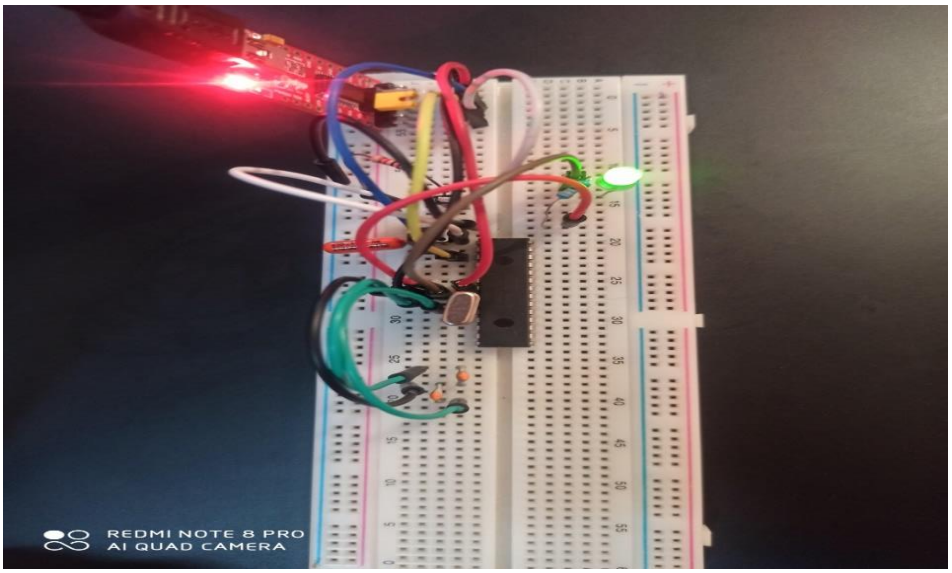


Figure IV.4. Le montage du programmateur AVR à base de FTDI

Ce montage nous a causé un problème dans la programmation comme la figure suivante indique :

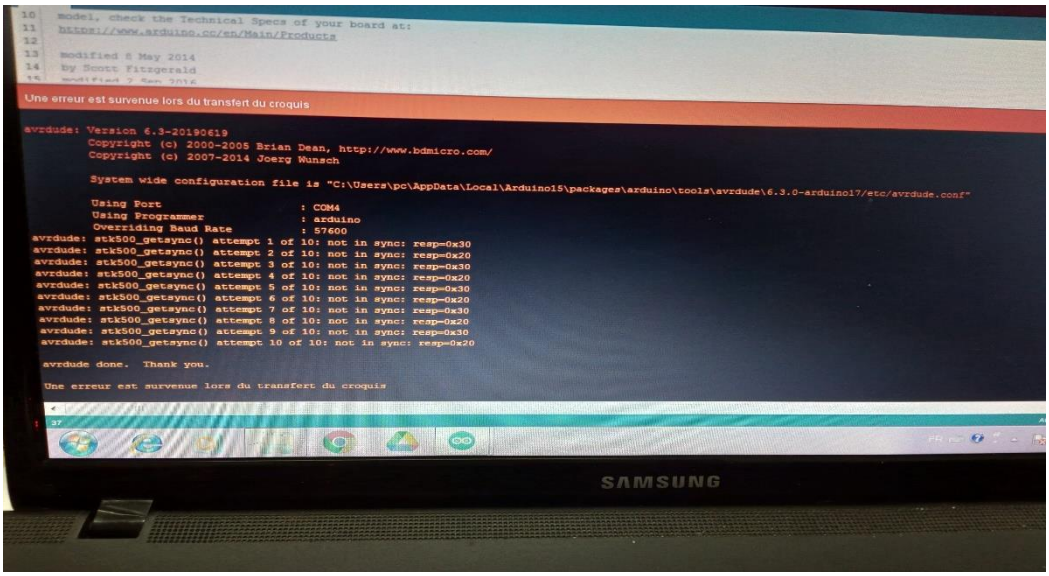


Figure IV.5. Test du montage échoué

Donc, comme vous voyez le test de ce montage est échoué à cause de l'absence d'un microcontrôleur programmeur Atmega16U.

IV.1.2.2. Programmateur à base d'un Atmega8 :

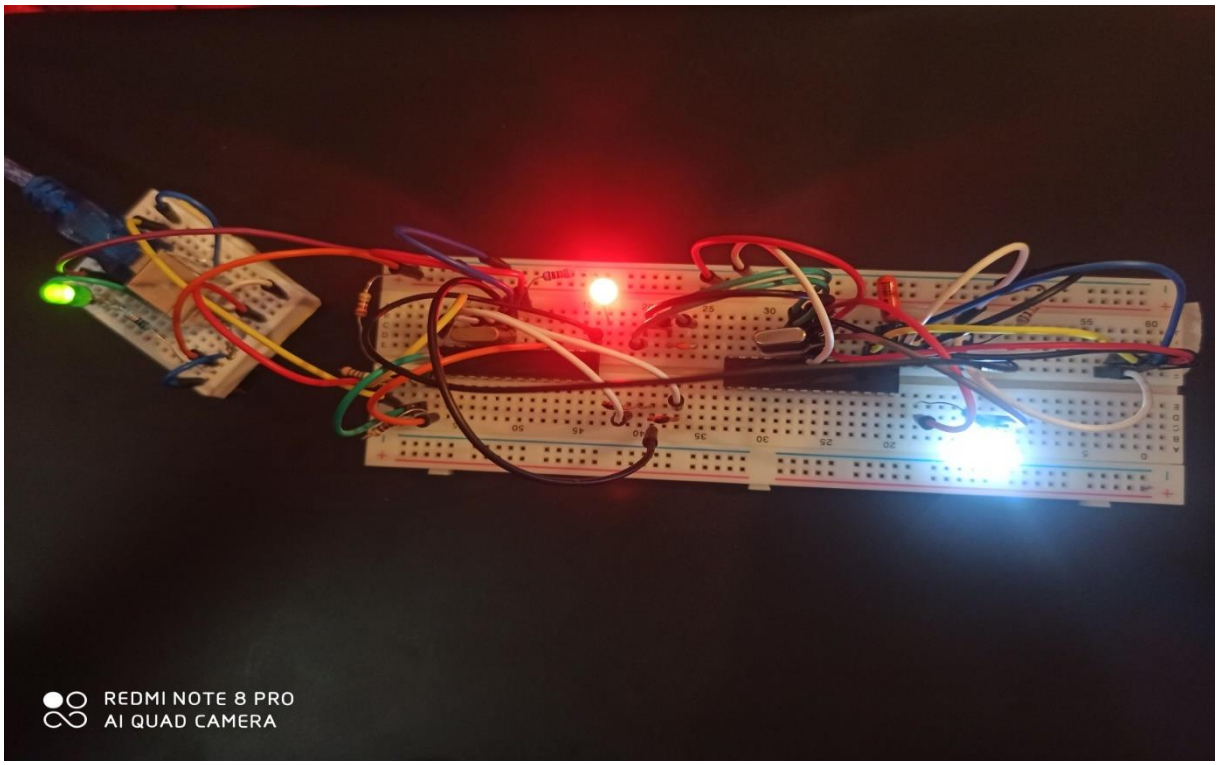
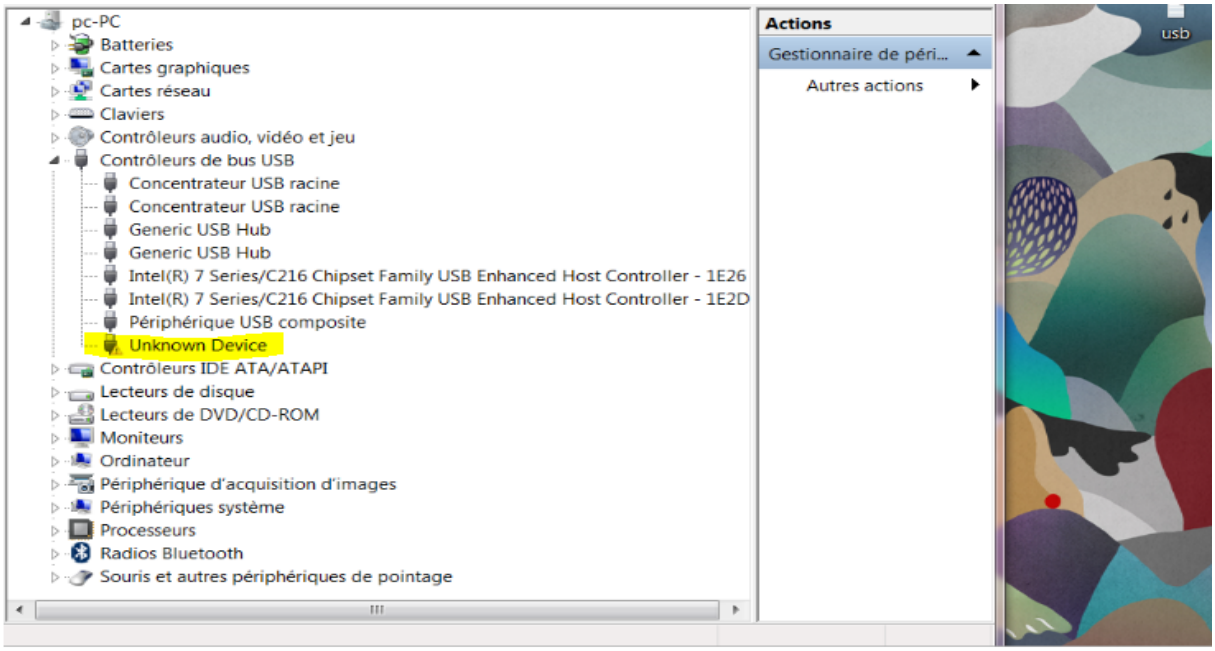
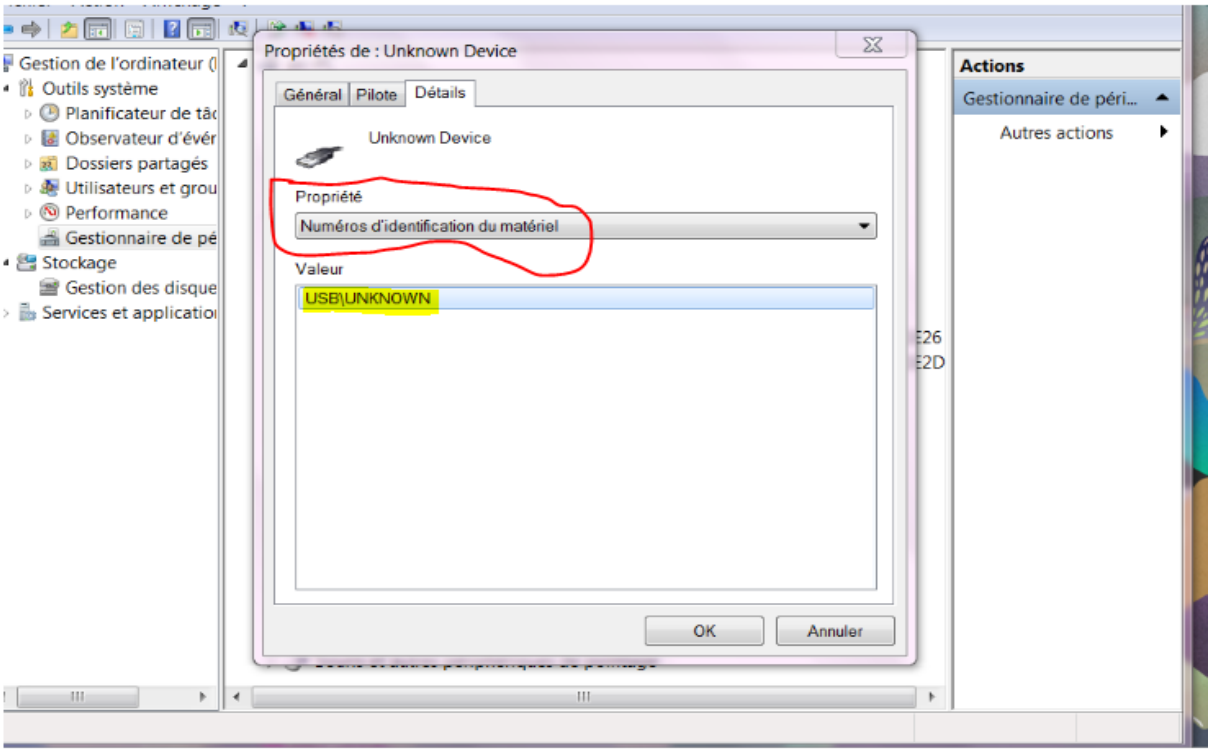


Figure IV.6. Le montage du programmateur AVR à base d'Atmega8

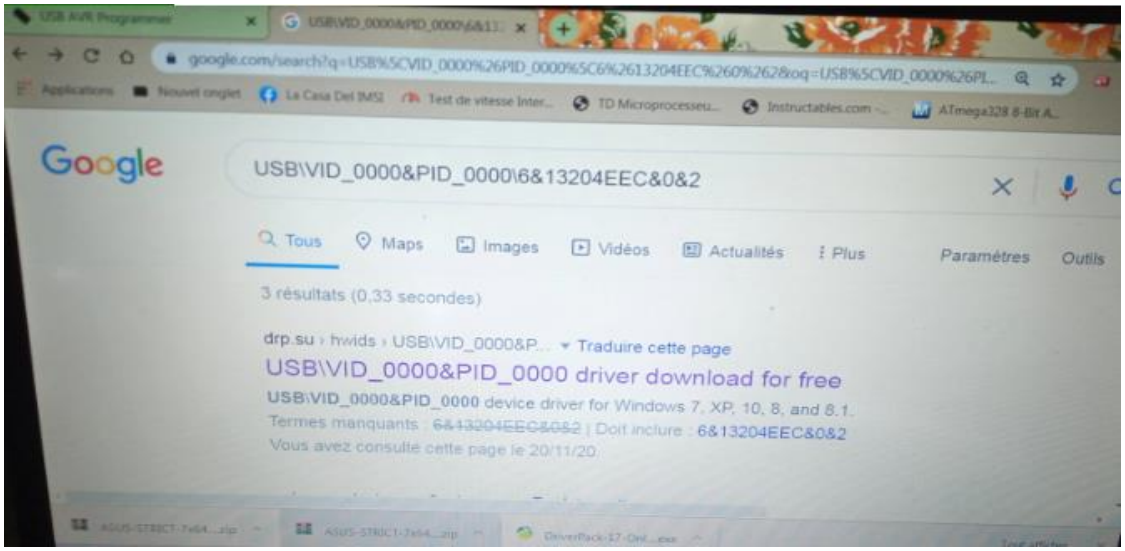
Concernant le montage du programmateur AVR dans la figure (IV.6) A trouvées un problème de communication lorsqu'on a lié l'USB avec le pc une fenêtre s'affiche comme suit :



L'USB est affiché inconnu, alors maintenant on va chercher le pilote correspond à ce circuit à partir du montage, la figure ci-dessous montre cette étape :



Toujours le même problème, le matériel et aussi inconnu donc quand à chercher sur net a ce pilote n'affiche rien. Tandis que la recherche sur net normalement doit être affiche le driver du programmateur USBasp.



Après cet essai, on n'arrive pas à résoudre le problème.

IV.1.2. 3. Programmateur à base d'un port RS-232 :

On a réalisé le montage qui déjà étudié dans le chapitre III sur une plaque d'essai pour le tester, le montage est dans la figure (IV.7)

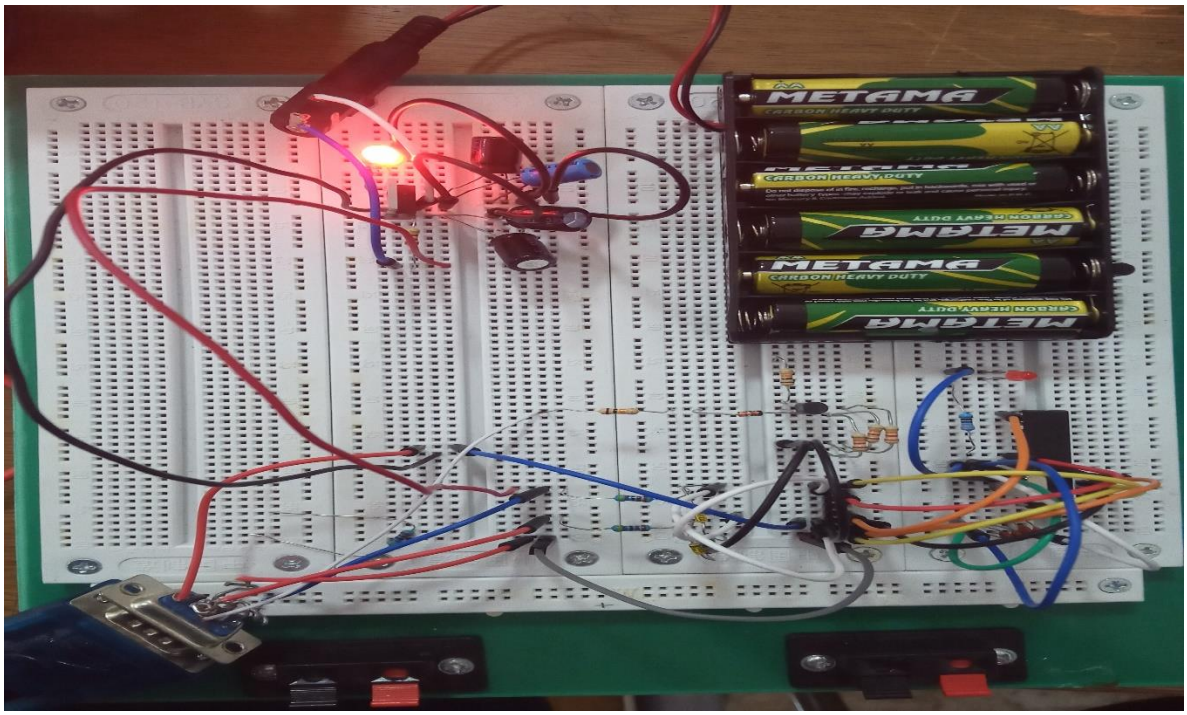


Figure IV.7. Montage du programmateur à base de RS-232

La communication de ce programmateur avec le PC et s'effectuée partir d'un adaptateur RS-232 USB

Cet adaptateur permet de transformer un port USB en port de communication série RS-232 avec un connecteur DB9 Male. Il est idéal pour le raccordement de modem, PDA, GPS, appareils de mesure, machines, outils industriel...

Figure IV.8. Adaptateur USB RS-232

Maintenant, on va lier le pc avec le programmateur avec l'adaptateur mais il faut télécharger le driver correspond comme la figure montre :

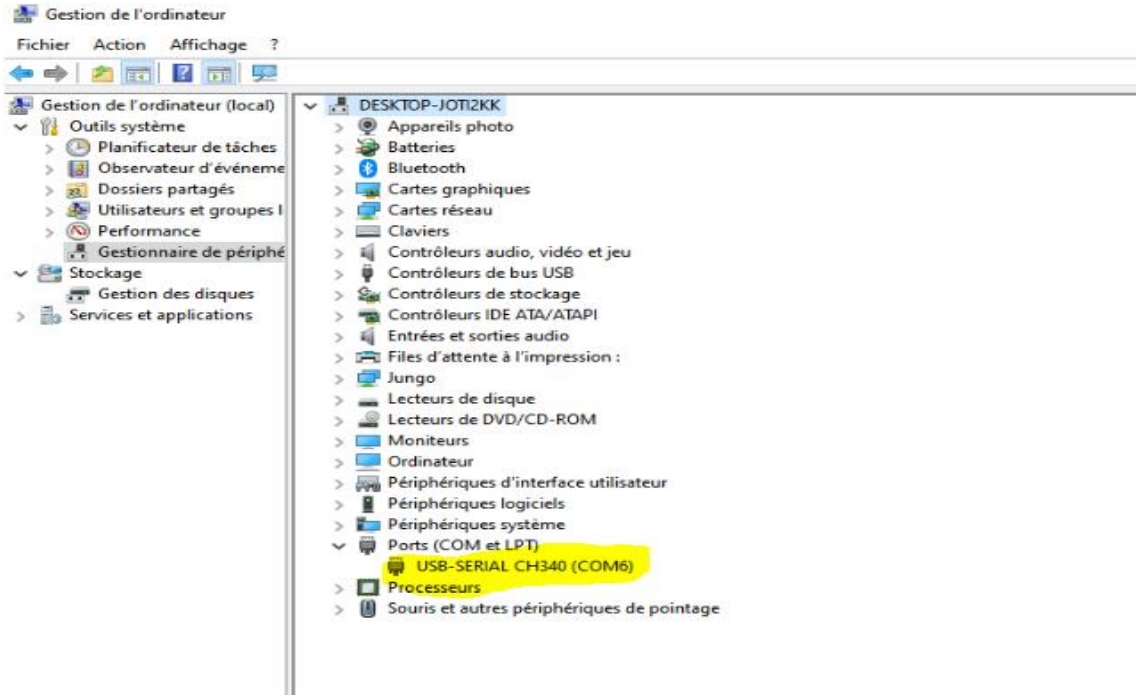


Figure IV.9. USB reconnu

Après la réussite de la communication, on passe maintenant au téléversement du programme simulé dans le chapitre III.

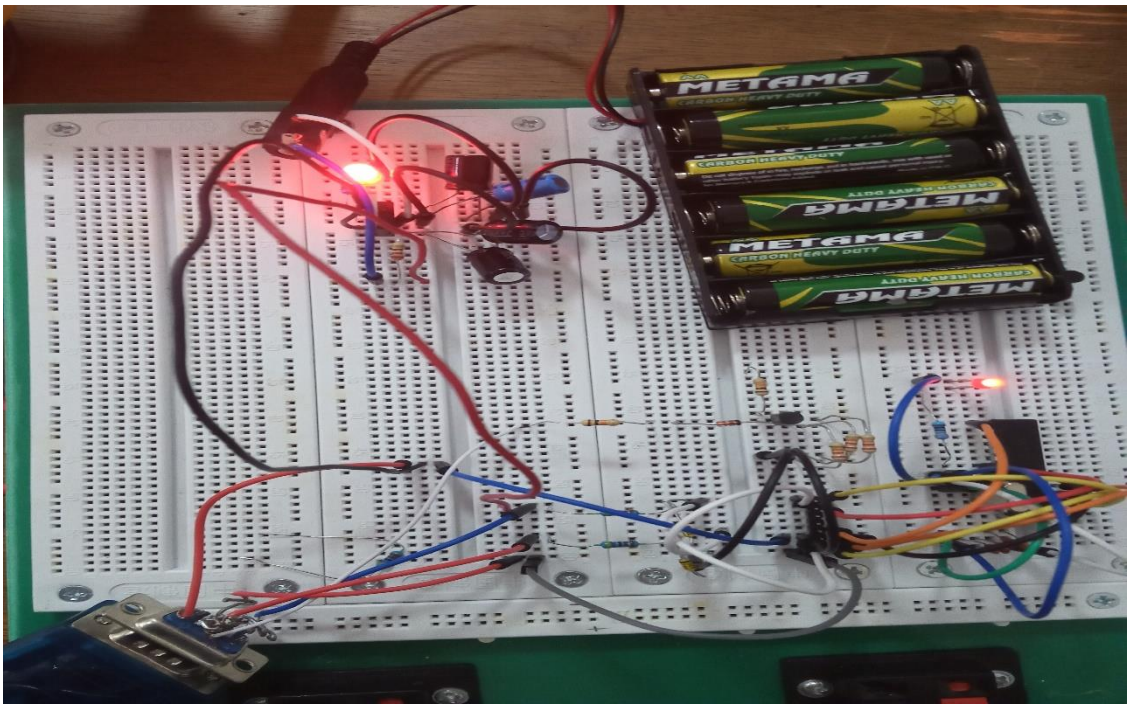


Figure IV.10. Test de montage

IV.2. Réalisation et test sur la plaque PCB :

IV.3.1. La carte d'interfaçage :

IV.3.1.1. Schéma de la carte Sur Proteus ISIS:

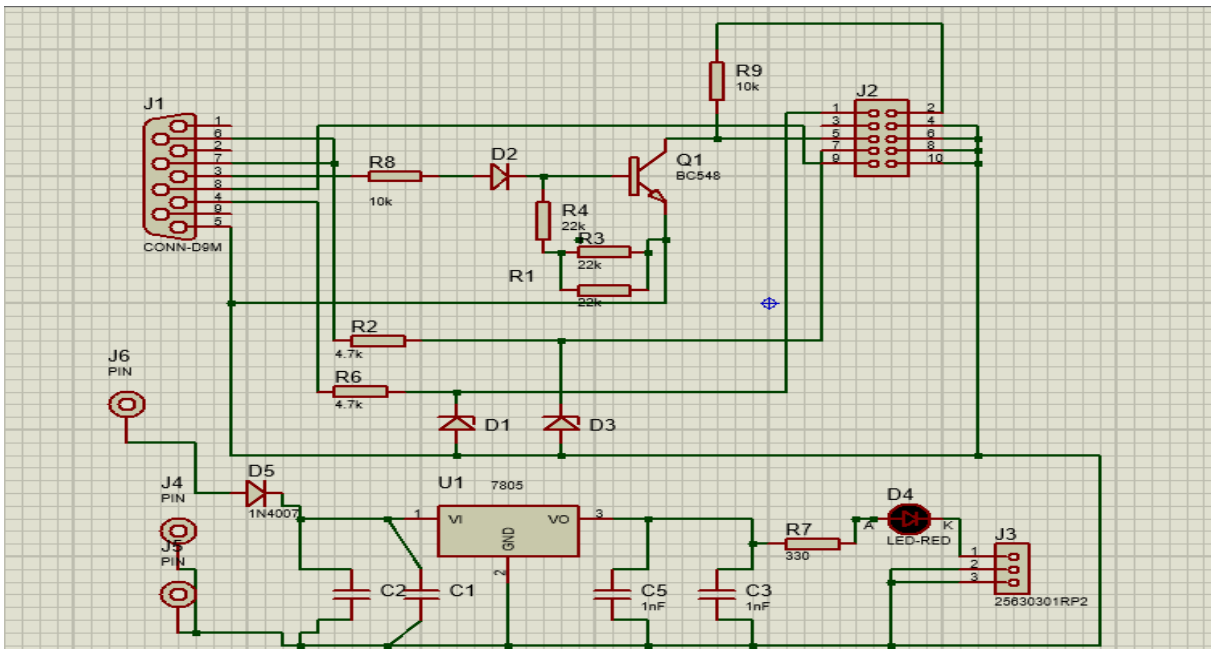


Figure IV.11. Schéma Electrique d'interfaçage Sur ISIS

IV.1.1 Sur Proteus ARES :

On ajuste ses dimensions à l'aide du logiciel ARES Proteus pour les faire correspondre à celles du papier et de la carte.

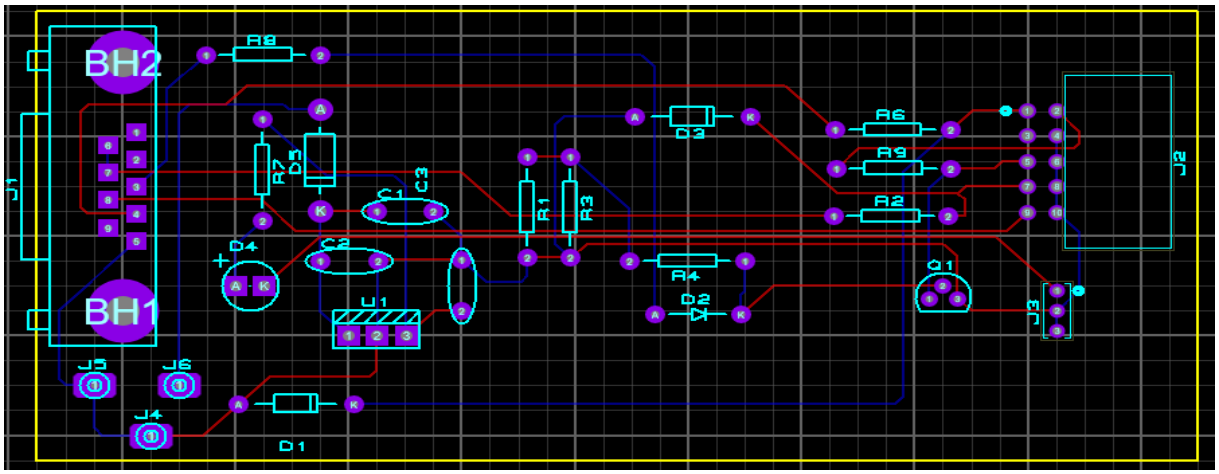


Figure IV.12. Circuit d'interfaçage sur ARES

III.3.3. Vue 3D de la carte :

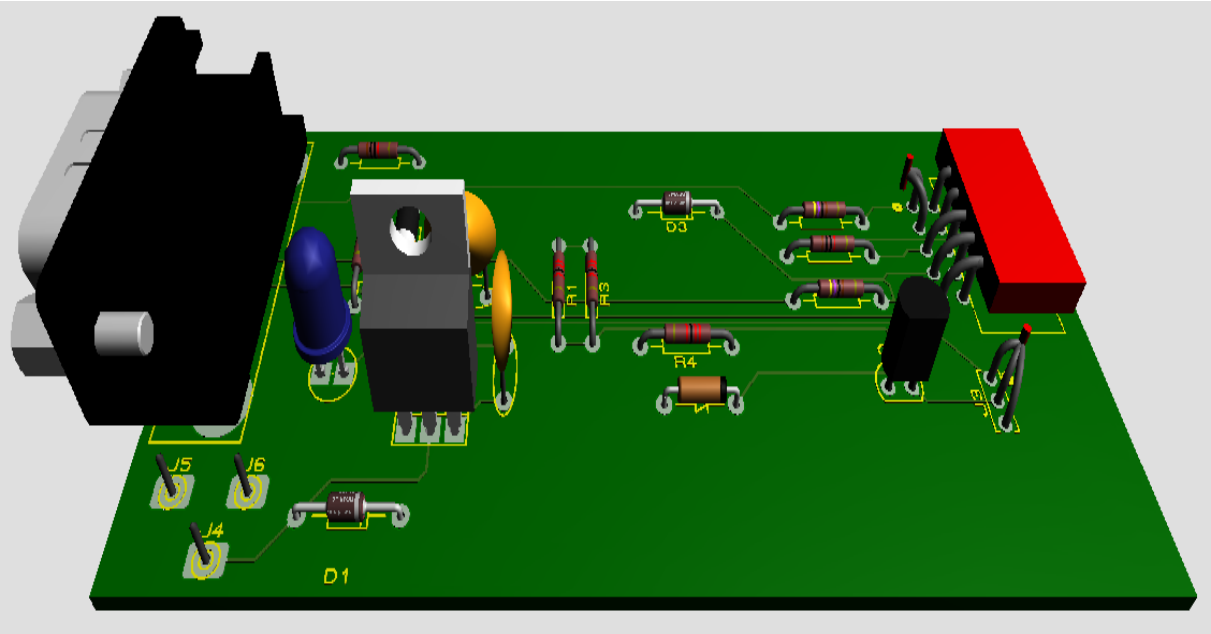
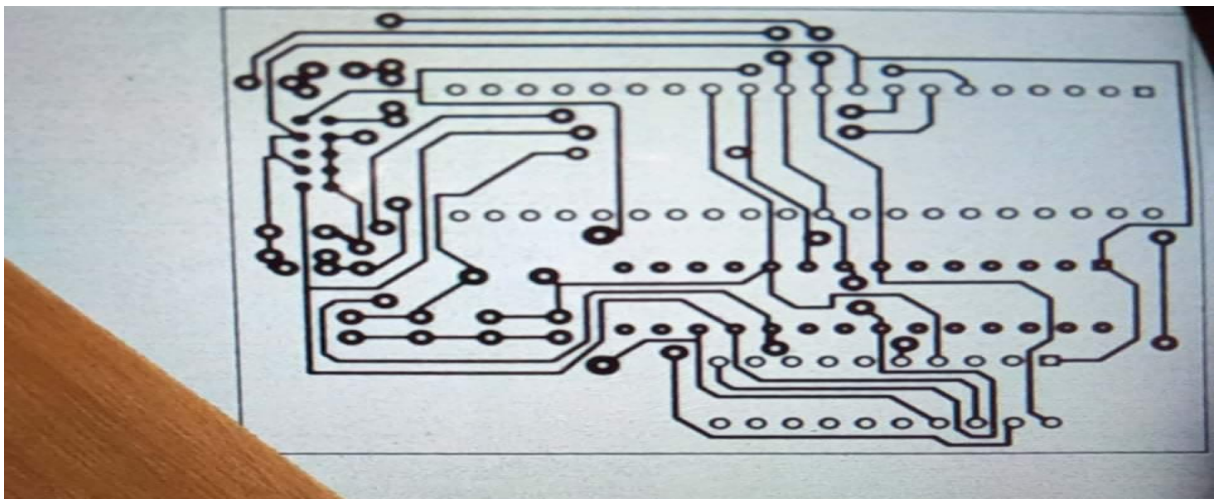


Figure IV.13. Vue 3D de la Carte d'interfaçage

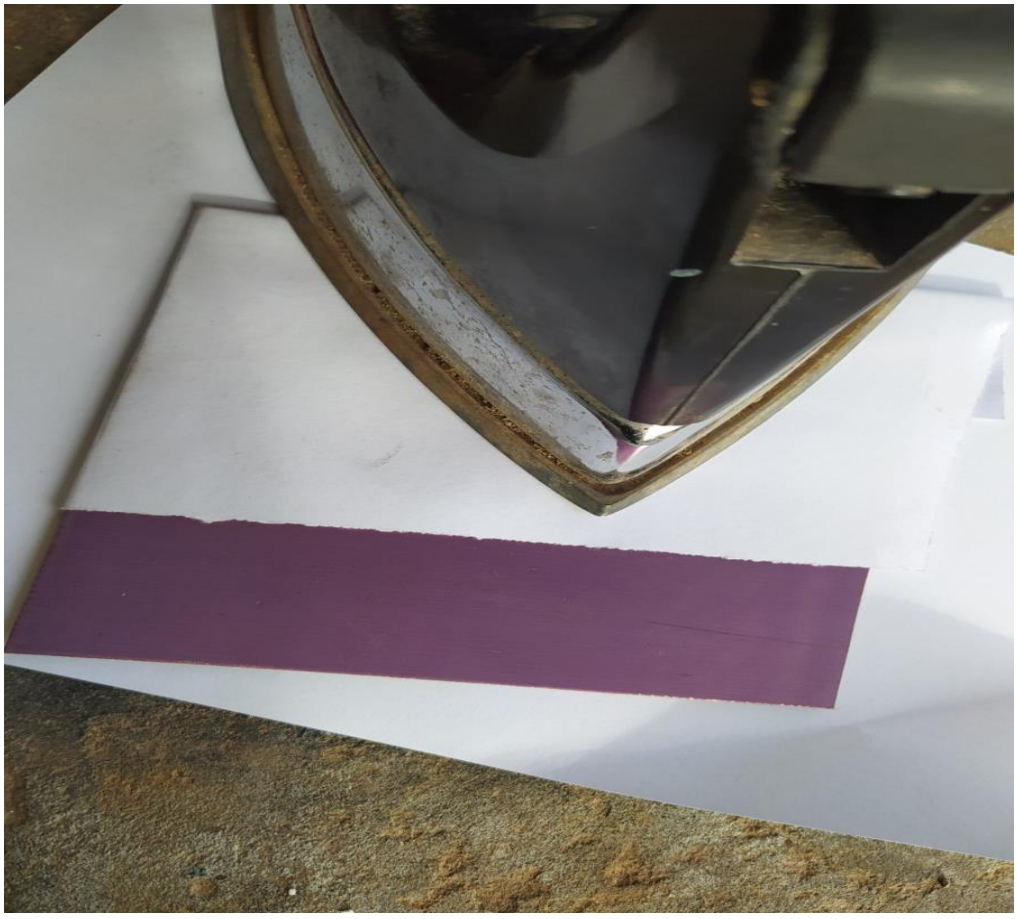
III.3.4. Imprimé sur le papier calques (typon) :

L'impression du schéma sera effectuée sur un papier



FigureIV.14. Imprimé du circuit ARES sur le papier

Passons à l'étape suivante. Posons le recto du schéma sur la face cuivrée de la carte en veillant à aligner les deux pièces. Mettons en route le fer électrique, ensuite on va placer le sur son support et attendez qu'il chauffe bien.



FigureIV.15.le fer électrique

On Laisse le fer en place pendant 30 à 45 secondes

Après, Refroidissement de la carte mettons la plaque et le papier dans un récipient rempli d'eau chaude pendant quelque temps, sans dépasser 10 minutes et enlèves le papier nous reste que le schéma sur la PCB.

Maintenant mettons la plaque sur le réactif d'attaque (perchlorure de fer), après 5 minutes l'immersion est terminer on sortant et lavons la carte quand tout le cuivre superflu aura disparu sous l'effet de l'acide.

FigureIV.16.la plaque sur le réactif d'attaque

Après la finition on va percer les trous de montage et enfin, on passe à la soudure des composants sur la plaque

Le résultat est :

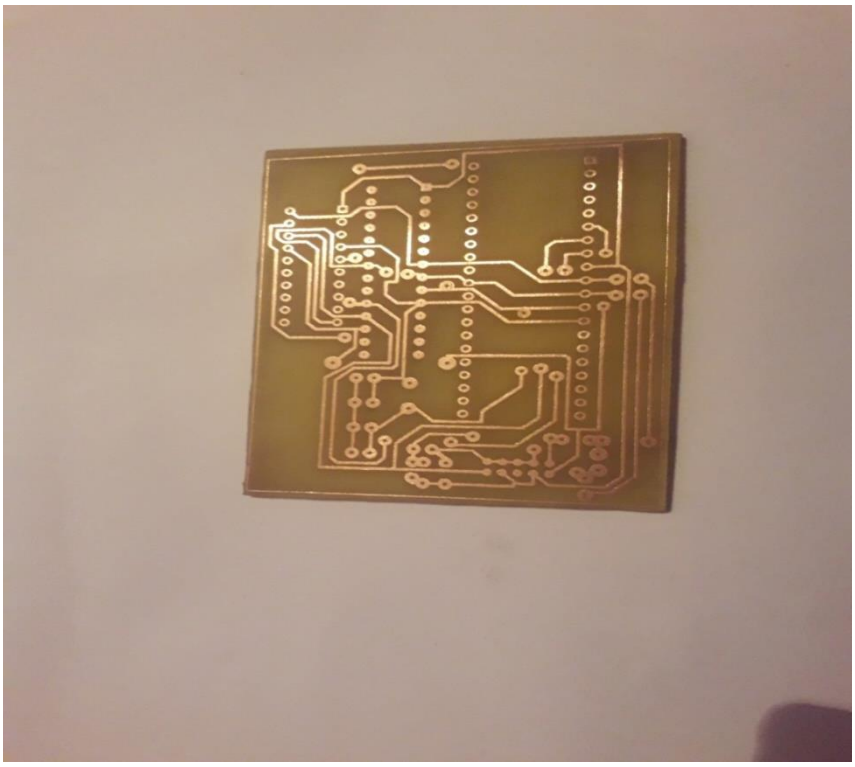


Figure IV.17. Circuit Imprimé de la carte d'interfaçage

La soudure :

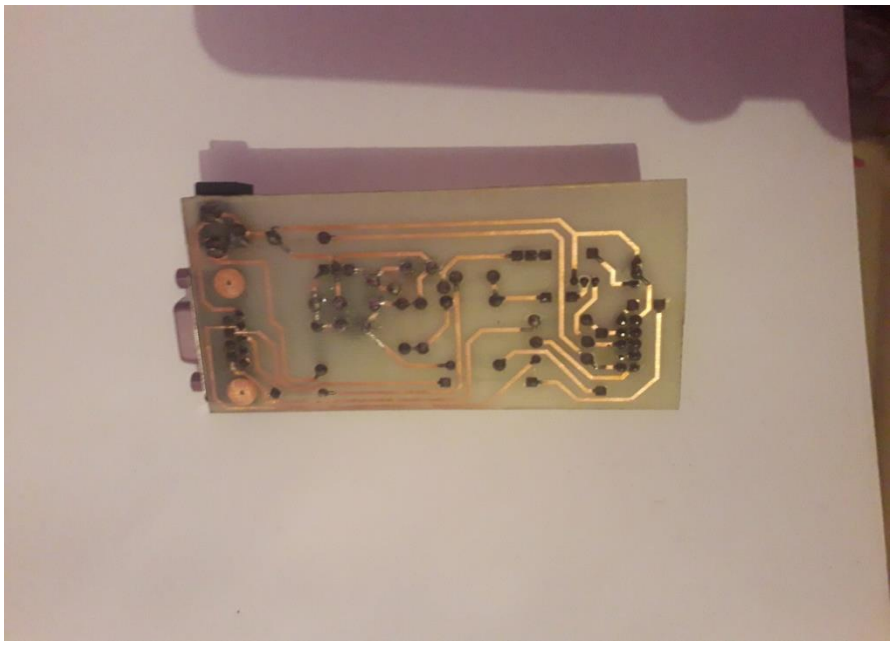
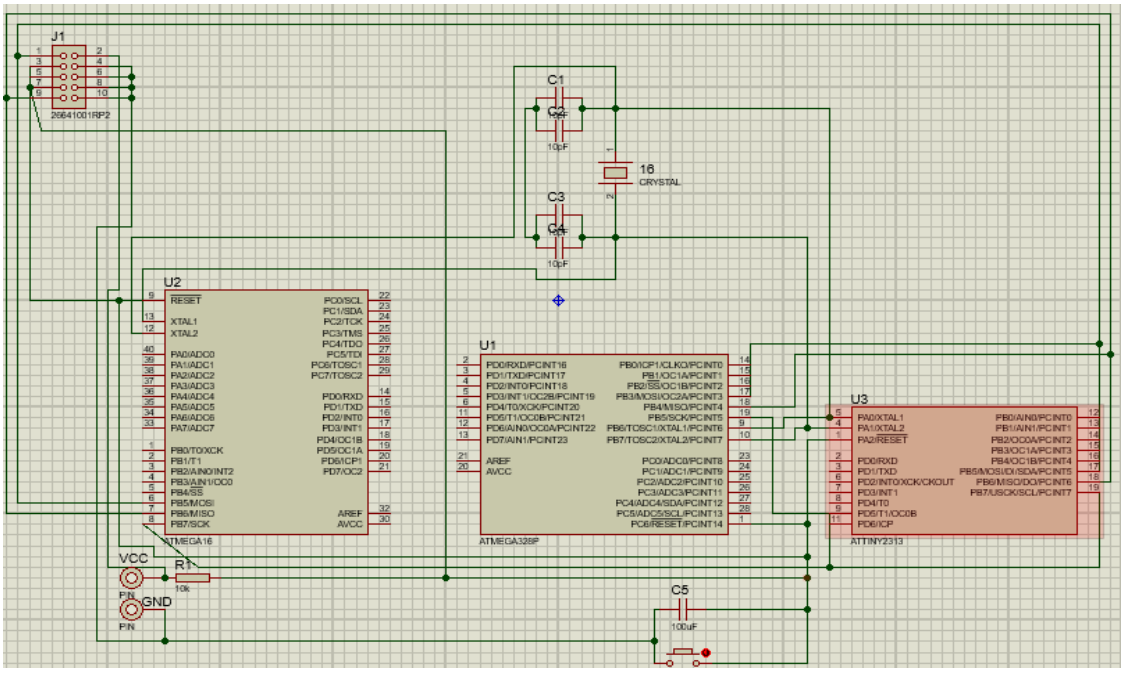


Figure IV.18.la soudure de la carte

Figure IV.19. Le résultat final de la Carte

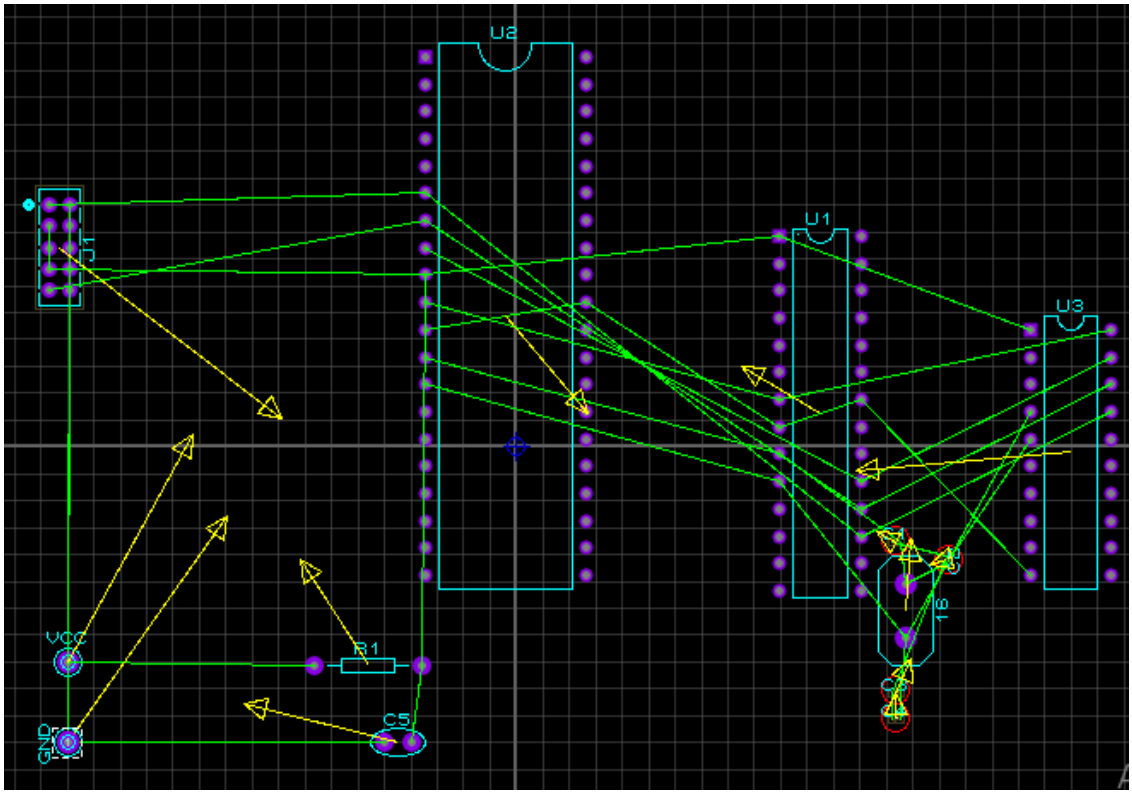
IV.3.1. La carte des microcontrôleurs AVR :

a. Sur Proteus



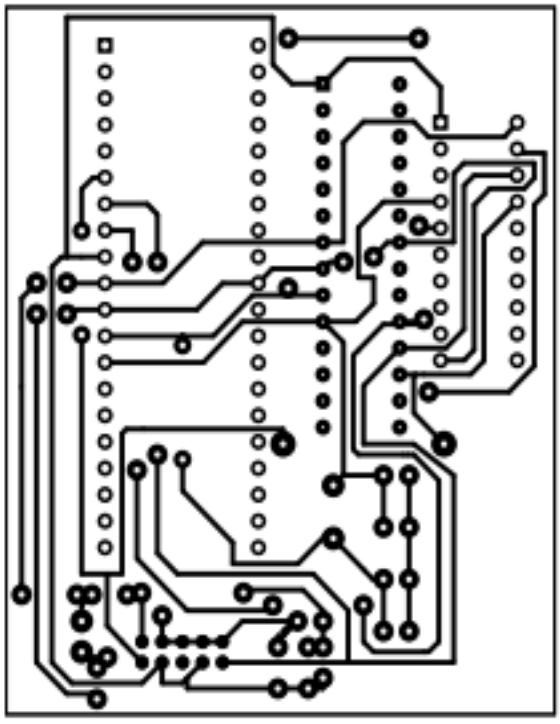
FigureIV.21.les MCR sur Proteus

b. sûr Ares



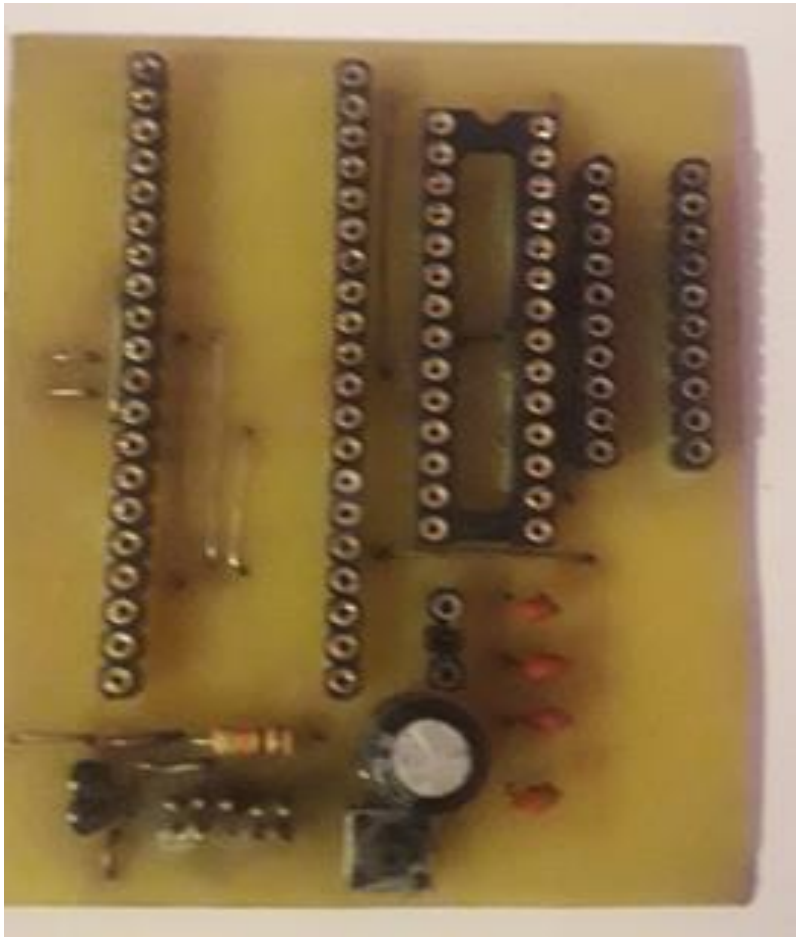
FigureIV.20. les MCR sur fenêtre Ares

c. impression de circuit Ares sur le papier



FigureIV.21.Les MCR sur le papier

Résultat finale de la carte



FigureIV.22.résultat final de la carte

IV.3. Exemple d'utilisation d'un microcontrôleur AVR :

Le robot suiveur de ligne est l'un des types de robots les plus importants dans le monde des microcontrôleurs car il a de nombreuses utilisations car il peut être utilise dans les usines et les entreprises.

Là où nous voyons qu'il est largement utilise dans les usines les compagnies maritimes et bien d'autres choses, il peut également être utilise dans les hôpitaux pour éviter le contact avec les blesses et leur livrer de la nourriture, des boissons et des médicaments

Maintenant, nous passons a comment cela fonctionne, nous le concevrons sur une voiture robotique.

IV .3.1. Le matériel utilisé :

- 3x détecteurs de ligne
- Atmega328P
- 2x moteurs à courant continu
- Pont en h-bridge
- Batterie 9v

La figure suivante représente le matériel utilisé pour réaliser un robot suivi la ligne :

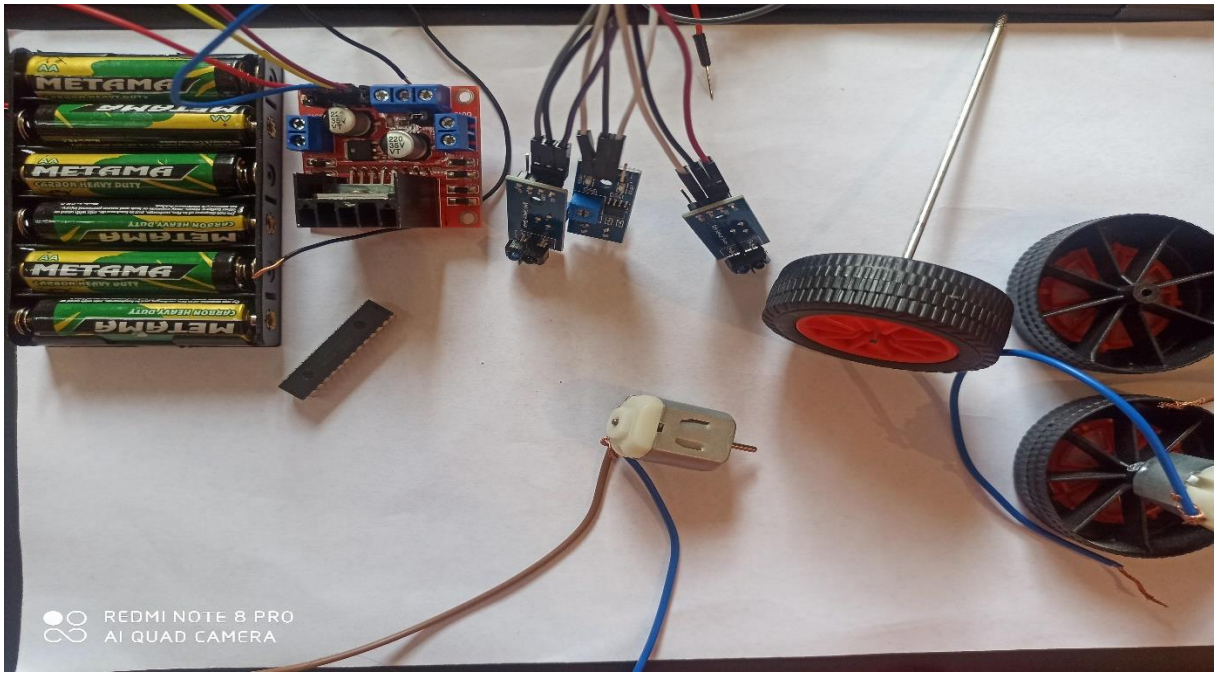


Figure IV.20 Matériel utilisé

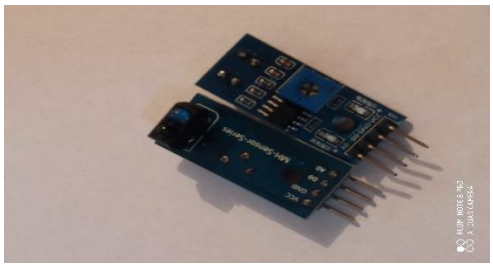
a. Capteur de ligne :

C'est un capteur important avec Atmega328P, car il est sensible aux couleurs noir et blanc, il en donne un

Nous en utiliserons trois pour indiquer au robot la direction de la ligne si la ligne va tout droit

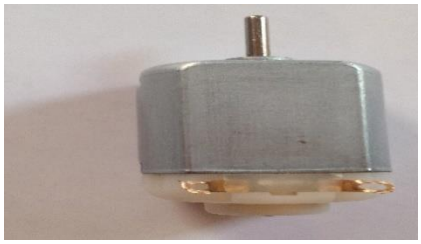
Le capteur du milieu donnera un zéro

Si la ligne va vers la gauche, le capteur gauche lit zéro [29].



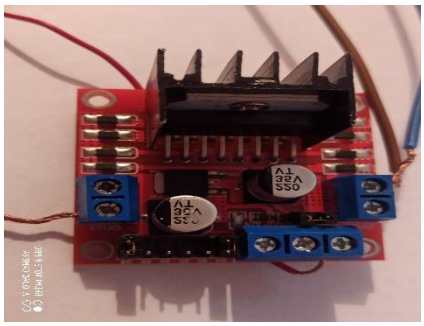
b. DC Moteur :

C'est un type de moteur utilisé dans les petits robots, Nous utiliserons ce type de moteur dans ce projet.



c. h-bridge :

C'est la puce qui contrôle le moteur, il est strictement interdit de connecter l'Atmega328P directement aux moteurs sinon, l'Atmega328P sera bruler cette puce prend une source d'alimentation externe et est souvent une batterie de 9v il prend le signal de l'Atmega328P pour démarrer n'importe quel moteur tout en déterminant son sens de rotation [2].



IV .3.2. Les connexions :

Les ports en doivent être connectes aux prises numériques, et ces ports sont responsables du fonctionnement des moteurs

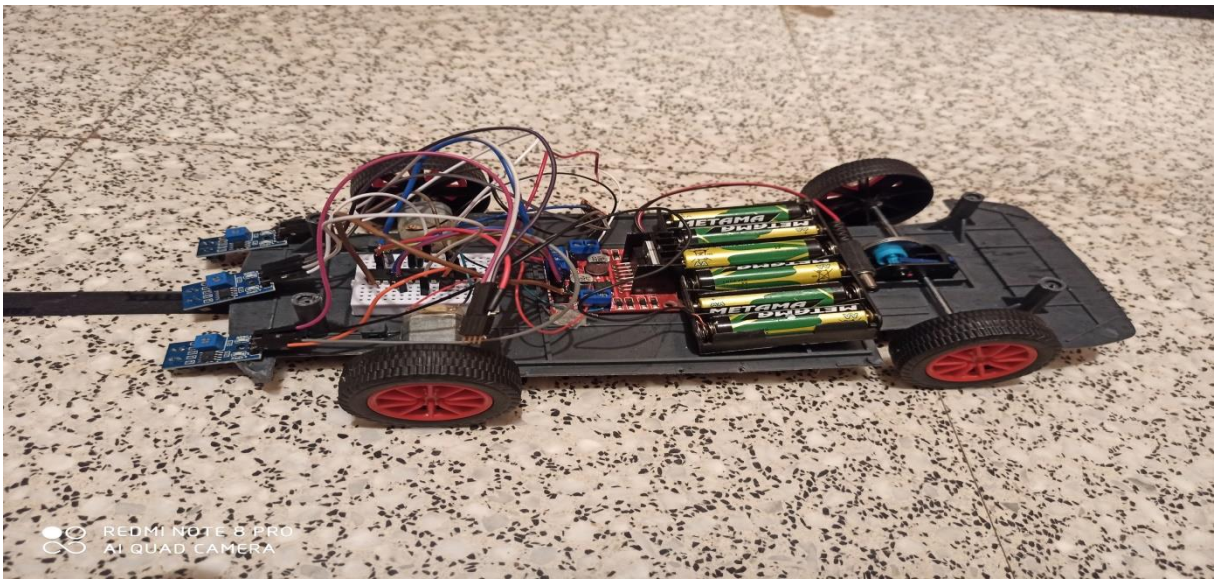
On colle les moteurs comme la figure montre .Après cela, on mettre le reste des composants.

- Le capteur de ligne doit être placé à l'avant du robot

- Un capteur doit être placé à l'extrême droite de l'avant du robot.

- Un autre a été placé à l'extrême gauche de l'avant du robot.

- Le dernier est placé au centre de l'avant du robot dans la partie face au sol.



Après la programmation de microntroleur Atmega328P le robot fonctionne.

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 . 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

Nos objectifs de départ sont réalisés, la carte d'interfaçage est fonctionnelle et le programme du microcontrôleur est stable. Le programme situé sur l'ordinateur reçoit correctement les informations provenant du télémètre et n'a plus qu'à les afficher. L'ajout de nouveaux matériels sur la carte actuelle devrait être simple et ne nécessiter que la reprogrammation du microcontrôleur. Néanmoins, nous aurions aimé trouver le matériel qui nous aurait permis de brancher et de débrancher facilement les périphériques de la carte. Ici, il faut souder ou dessouder les périphériques.

Concernant l'apport personnel du projet, nous pouvons dire que nous avons été très intéressés par le sujet et que nous serions heureux de poursuivre les recherches sur les microcontrôleurs. La gestion des membres et la répartition des tâches a aussi été un apprentissage qui nous a été bénéfique. Nous regrettons cependant d'avoir parfois été bloqués à certaines étapes de la fabrication de la carte pour des raisons telles que le manque de composants ou le délai de réalisation des circuits imprimés.

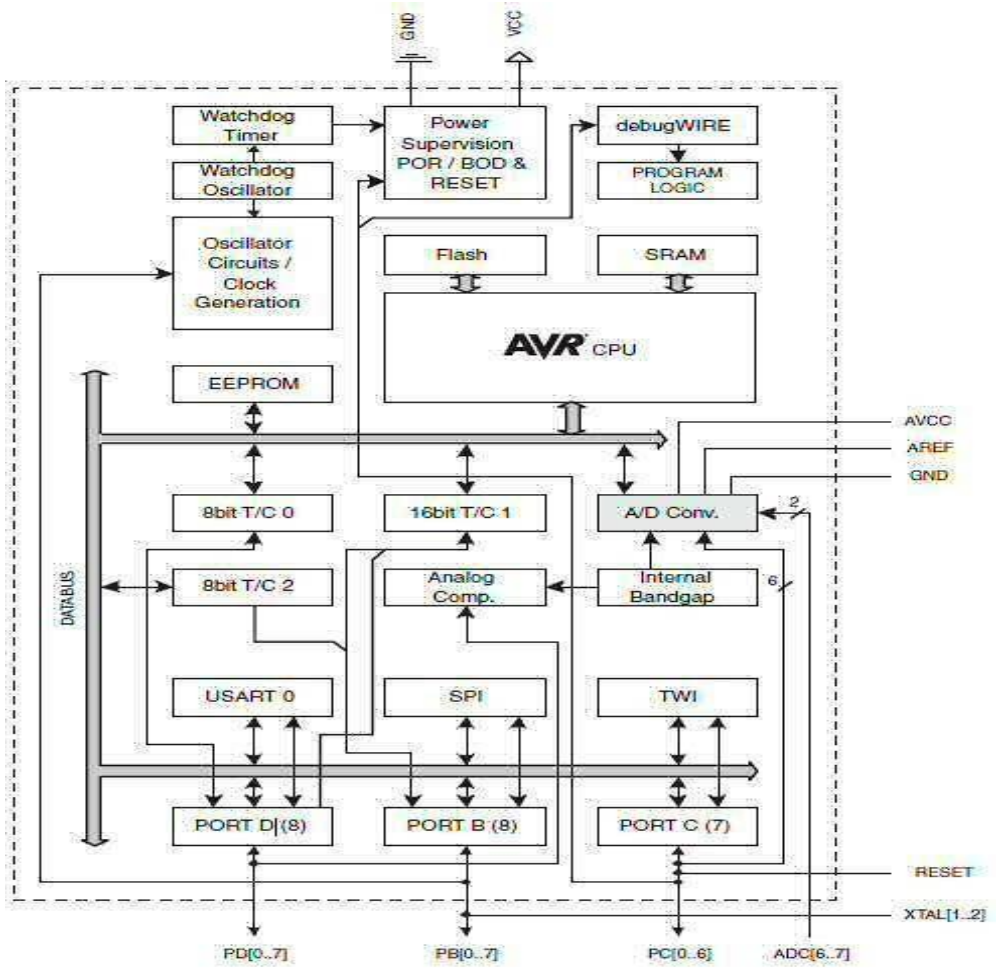
Les perspectives de poursuite du projet sont multiples. Nous utilisons actuellement notre carte d'interfaçage dans le cadre d'un projet de robotique, mais nous pourrions l'utiliser par exemple pour des mesures de température, de pression, de luminosité dans une installation domotique. Aussi, nous pourrions utiliser des capteurs infrarouges pour permettre un contrôle de la carte par télécommande.

Enfin, nous espérons que ce projet sera développé dans le futur surtout par nos amis de l'IMSI nous les encourageons à le faire.

Annexes

Annexe 1 :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

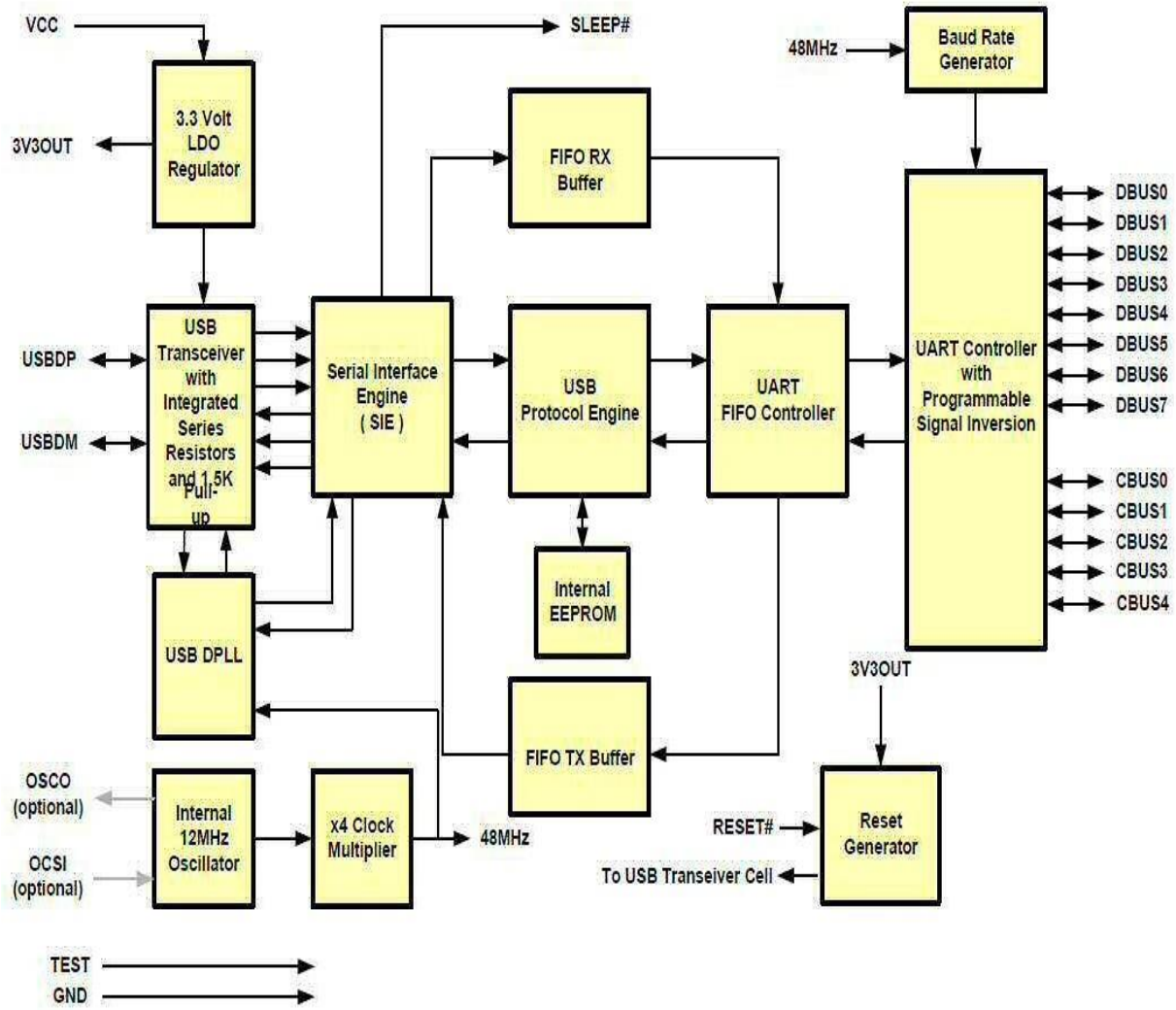
programmeur veut en faire. Ils sont aux nombres de 4 :

- 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.

Annexe2 : 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.

- 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.

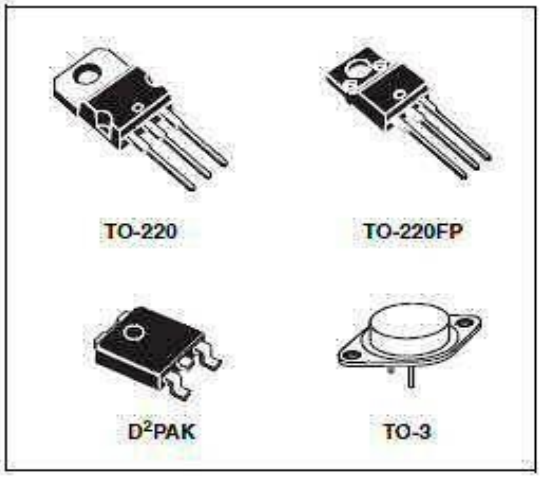
Annexe03 : Régulateur de tension L7805

L7800 series

Positive voltage regulators

Feature summary

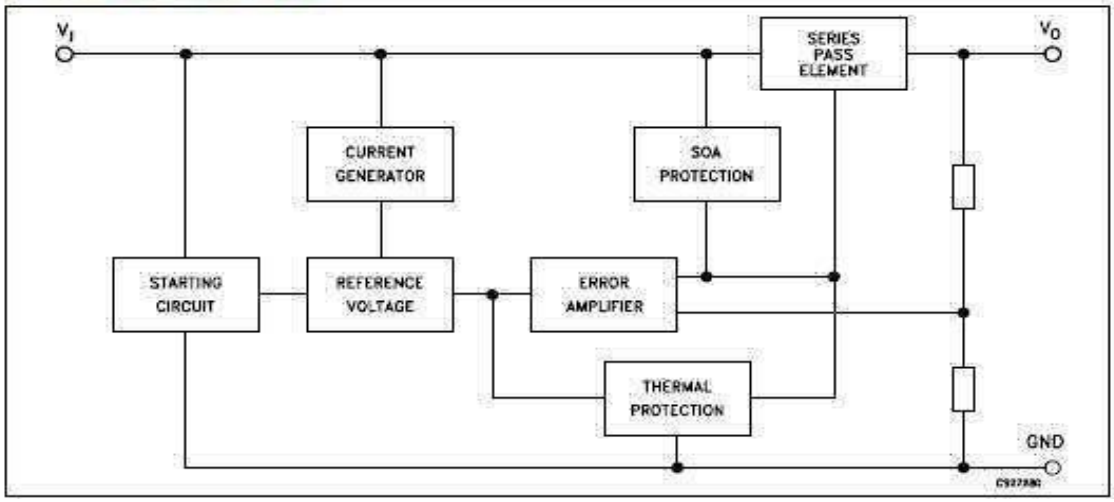
- Output current to 1.5A
- Output voltages of 5; 5.2; 6; 8; 8.5; 9; 10; 12; 15; 18; 24V
- Thermal overload protection
- Short circuit protection
- Output transition SOA protection



Description

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3 and D²PAK 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



Références Bibliographique

[1] https://fr.wikipedia.org/wiki/Interface_(informatique) (consulté le 28/03/2020 à 9 :30 :06)

[2] CCM (Comment ça Marche) par Jean-Francois Pillou tous droits réservés 2007 Quidea (consulté le 06/08/2020 à 11 :25:19)

[3] fr.betweenmates.com/difference between serial and parallel 2017 (consulté le 06/08/2020 à 12:36:05)

[4] Euro-makers/accessoires arduino (consulté le 25/08/2020 à 08:16:53)

[5] Site de G. Laroche/hardware/progAVR (consulté le 25/08/2020 à 08:20:32)

[6] http://www.youtube.com/watch?v=pWavnF7P0EQ&173s

AVR Programming Crush Course 2012 Session: 01 Par Walid Balid (consulté le 12/04/2020 15:36:2)

AVR Programming Crush Course 2012 Session: 01

[7] MCS ELECTRONICS (BASKOM AVR)

[8] Datasheet Atmega48PA-88PA-168PA-328P Consulter le 05/07/2020 à 17 :05 :30

[9] https://www.componentsinfo.com/atmega328p-pinout-configuration-datasheet/ Consulter le 20/05/2020 à 15 :40 :02

Par AMMI ADEL Et KHELIF MOHAMED. Consulté le10/07/2020 09:30 :08

[10] Atmega328p Datasheet complète depuis 11/2016 par Atmel Consulter le 20/07/2020 à 14 :15 :04

[11]https://www.technologuepro.com/microcontroleur-2/chapitre-6-les-timers.pdf
par RKHISSI KAMMOUN Consulté le10/07/2020 11:38 :05

[12] Microcontrôleur Atmel Atmega Consulté le 20/07/2020 16:15:25

[13] Les périphériques interne systèmes embarques et microcontrôleurs

Consulter Consulter le 23/07/2020 20 :40 :35

[14] file:///C:/Users/NET/Downloads/Documents/PWM-MLI.pdf Consulter le 16/08/2020 17 :45 :10

[15] WIKEPEDIA Serial Peripheral Interface

[16] Programming AVR I2C interface Par ADMIN Consulté le 29/08/2020 00 :15 :33[10]

[17]ATmega328P to PC Serial Communication using USART Tutorial Par RAHUL SREEDHARAN le 09/07/2019 09h :26 am Consulté le 28/08/2020 22 :35 :14

18]file:///C:/Users/pc/Desktop/memoire/pdf%20essenciel/Etude%20et%20R%C3%A9alisation%20d'une%20Carte%20Arduino.pdf par DJAFRI Menad et CHELOUCHE Djalal le20/06/2016 (consulté le 10/07/2020 12 :03 :51)

[19]Electronics-DIY.com FT232RL USB to Serial Adapter for AVR ATMEGA 04/01/2009 par MIROSLAV BATEK (consulté le 14/08/2020 à 12 :20 :01)

[20]Science prog (consulté le 18 /08/2020 à 13 :30 :05)

[21]SparkFun Electronics par par Miroslav Batek le 04/01/2009 (consulté le 14/08/2020 à 12 :36 :12)

[22] Électronique les microcontrôleurs AVR (consulté le 20/08/2020 à 19 :00 :01)

[23]https://arlontronique.wordpress.com/alimentation-de-latmega8/module-dalimentation-5v/ le 01/09/2011 (consulté le 14/08/2020 10 :15 :06)

[24] JDN IDE en informatique : définition pratique et détaillée par Osama E(consulté 19/08/2020 à 20 :45 :02)

[25]electricshub-bootloader atmega328 par Ravi(consulté le 23/08/2020 à 15 :55 :12)

[26] Lancos PonyProg serial device programmer (consulté le 24/08/2020 à 16 :50 :13)

[27]MicroC FPGA | | Cours | Électronique (consulté le 24/08/2020 à 16 :55 :15)

[28]EleKtronique COURS ET MONTAGE ELECTRONIQUE (consulté le 24/08/2020 à 17 :10 :12)

[29] ELECTRONIQUE en amateur (Robot suiveur de ligne) 19/04/2020 par Yves Pelletier consulté le 20/09/2020 à 21 :33 :05