Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté de Génie Electrique et d'Informatique Département d'Electronique

MEMOIRE DE PROJET DE FIN D'ETUDES

pour l'obtention du diplôme de

MASTER EN ELECTRONIQUE

SPECIALITE : ELECTRONIQUE BIOMEDICALE

Présenté par : Mouloud Chibane

Commande d'un moteur Brushless à l'aide un Microcontrôleur Atmega et Interface à un Afficheur LCD

Soutenu le : septembre 2014 devant le Jury

Mr : ACHOUR		Président
Mr : Farid *Ait Ouamer*	*Doctorant*	Encadreur
Mr : F. OUALLOUCHE		Examinateur

Remerciement

Louanges à ALLAH pour les bénédictions et l'aide qu'il nous a accordé. Puisse-il faire en sorte que mon présent travail soit sincère et bénéfique.

Je tiens à remercier notre cher pays, qui nous a fourni les bonnes conditions de l'enseignement dans toutes ses phases : du primaire jusqu'au supérieur.

Mes remerciements vont aussi à tous le corps pédagogique : enseignants, administrateurs, employés du département de Génie électronique ainsi que toutes les personnes de notre faculté.

Je remercie également monsieur H. Bouzar, directeur responsable du laboratoire de physique (LPCQ) et son équipe pour avoir accepté de mettre le laboratoire à notre disposition.

Je tiens à remercier, particulièrement, mon encadreur le docteur : Farid Ait Ouamer pour le temps et l'aide qu'il a voulu me consacrer tout au long de la réalisation de ce mémoire, ainsi que pour ses remarques, ses critiques, ses conseils judicieux qui m'ont été très profitables.

Enfin Je tiens à témoigner ma reconnaissance et ma gratitude à mes parents, mes frères, ma sœur, mon oncle et mon grand père, ainsi qu'à toute ma famille, sans oublier aussi mes amis pour leur soutien et leur aide.

Liste des figures

Liste des tableaux

Introduction générale

L'industrie des semi conducteurs se classe parmi les plus avancées entreprises mondiales. Elle génèrent des milliards de dollars, fait travailler des milliers de chercheurs et d'ingénieurs. Elle leurs permet d'être à la pointe de la technologie et ce grâce aussi à leurs chercheurs et leurs étudiants. L'intégration des transistors est passé de quelques milliers à plusieurs milliards en quelques dizaines d'années à tel point qu'on arrive à avoir une puce ayant les fonctions d'un ordinateur. Ces puces mettent à nos dispositions une suite complète d'outils de développement pour gérer tous les aspects du processus de la conception à la réalisation où les étudiants peuvent transformer leurs idées en réalité. Les applications obtenues ont des retombées dans les déférents domaines d'activité scientifique et technologie comme : le médical, l'automobile et l'industrie.

Dans cette perspective, on a été emmené à choisir un projet de mémoire de fin d'étude intitulé : 'Programmation d'un microcontrôleur et interface à un afficheur LCD'. Ce projet combine un microcontrôleur de type Atmel programmable en langage C via un programmateur et d'une interface à un moteur brushless. Le programme est écrit de façon à déclencher le moteur pour tourner dans un sens ou l'autre tout en affichant les valeurs sur un afficheur. L'afficheur utilisé est basé sur le LCD Nokia 1100.

Notre tache consiste à :

- Faire une recherche bibliographique (majoritairement sur Internet).
- Comprendre et programmer ce type d'afficheur.
- Interfacer les microcontrôleurs (led, afficheur, moteur, interrupteur, etc…)
- Vérifier et tester plusieurs programmes
- Flasher le microcontrôleur

- Vérifier les valeurs de l'afficheur.

On doit, en préalable, avoir choisi une plateforme de développement simple, efficace, peu coûteuse et que nous pouvant facilement acquérir et installer.

Notre choix s'est porté sur les microcontrôleurs AVR ATmega8 qui sont produit par Atmel, une société américaine. Ces microcontrôleurs AVR sont programmables en C en utilisant un logiciel gratuit Avr Studio de chez Atmel. Ils ont plusieurs caractéristiques qui nous paraissent les plus adaptés à notre type de projet et qui selon plusieurs experts surpassent globalement d'autres plateformes de développement.

L'environnement d'Avr Studio utilise le de-facto stardard avr-gcc qui est disponible en tant que logiciel libre. Le programme écrit en langage C est conçu pour tourner le moteur en deux sens de rotation et de mesurer sa vitesse de rotation. Celui-ci est alimenté en courant continu, comme pour notre maquetter, utilise les tensions standards de la boite d'alimentation qu'on retrouve dans les PC.

Le moteur est attaché à un codeur. Celui-ci tourne autour de la fente d'un boîtier concernant un photo interrupteur. Le moteur, en tournant, génère un train d'impulsions dont le nombre permet de déduire la valeur du déplacement ainsi que la vitesse angulaire du moteur. Le changement de la direction de rotation est obtenu grâce à deux relais montés en pont H.

L'interface électronique de ce moteur à un microcontrôleur et la programmation des tâches désirées donnent des mesures précises et répétées. Les mesures permettent d'avoir quelques caractéristiques pertinentes des moteurs étudiés. On peut aussi trouver l'angle de rotation du moteur, dû à l'inertie, avant son arrêt complet et après que la commande d'arrêt est déclenchée. Cette valeur est à prendre en considération pour modifier le sens de rotation ou pour le positionner en un angle donné. Autrement dit, on calcule l'équation horaire du mouvement circulaire du moteur.

Pour pouvoir faire ces calcul, on besoin d'analyser les données. Plusieurs options s'offrent. On s'est concentré sur un afficheur en temps réel des valeurs désirées. L'affichage des valeurs est basé sur le LCD Nokia 1100 qui offre une solution adéquate pour ce type de projet.

En plus, on avait besoin d'un moyen d'input commode pour pouvoir entrer nos commandes. On a préféré un control à distance utilisant un détecteur de type TSOP17XX qui représente un récepteur infra rouge. On dispose de trois broches (GND, VCC, Out) et d'une télécommande

MP3 banale représentant un émetteur IR. Là, on a pris avantage d'une bibliothèque déjà existante au laboratoire où on a programmé quelques lignes de codes ; simplifiant ainsi le travail.

Ce mémoire est divisé en plusieurs chapitres. Ils sont décrits d'une manière logique, progressive et concise. On décrira le microcontrôleur Atmga8, puis on parlera de la plateforme de développement en C d'Avrstudio suivi d'une discussion des moteurs et codeurs ainsi que des composants électroniques utilisés. Avant de finir avec une conclusion, on citera les tests et résultats obtenus pour ce mémoire.

Il faut rappeler qu'en science ou technologie qu'aucun projet ne sort du néant et qu'aucun travail n'est complètement et entièrement l'acte d'un individu, mais repose sur les ressources déjà existantes que l'on adapte, réutiliser ou améliore. Dans cette vue, on a pleinement pris avantages des moyes mis à notre disposition, surtout la technologie de l'information qu'est l'Internet. Ceci est fondé sur le respecté la propriété intellectuelle des auteurs utilisée. Bien qu'on ait référencé les sources, tout oubli ou omission dans ce sens est non-intentionnel et on tient à leurs donner l'accréditation mérité.

Chapitre I :

Description des microcontrôleurs ATMEGA

1- Microprocesseurs et Microcontrôleur

L'histoire des microprocesseurs est intimement liée à celle de la technologie des semi-conducteurs dont l'invention du transistor an 1947 et l'apparition des nouvelles technologies comme la technologie bipolaire TTL et ECL, technologie MOS et CMOS.

La société américaine Intel réussit à placer tous les composants qui constituent un processeur sur un seul circuit intégré donnant ainsi naissance au microprocesseur. C'était un 4 bits baptisé 4004 destiné à équiper des calculatrices de bureau. En 1972, INTEL produit le premier microprocesseur 8 bits baptisé 8008 par référence au précédent [6].

Le microcontrôleur est un dérivé du microprocesseur. Sa structure est celle des systèmes à base de microprocesseur. Il est donc composé d'une unité centrale de traitement et de commande (équivalent au microprocesseur), de mémoire et de ports d'entrées /sorties.

2- La famille AVR d'ATMEL

Les microcontrôleurs de la famille AVR d'ATMEL possèdent de nombreuses caractéristiques différentes, aussi bien en termes de vitesse, mémoire, nombre d'entrées/sorties mais aussi au niveau des fonctions particulières qui sont disponibles. Il conviendra donc de choisir le microcontrôleur en fonction de l'utilisation qui est à faire.

Les microcontrôleurs de la famille ATmega en technologie CMOS sont des modèles à 8 bits AVR basés sur l'architecture RISC. En exécutant des instructions dans un cycle d'horloge

simple, l'ATmega réalise des opérations s'approchant de 1 MIPS par MHZ permettant de réaliser des systèmes à faible consommation électrique et simple au niveau électronique.

3- Deux architectures concurrentes : Von Neumann et Harvard

Architecture Von Neumann :

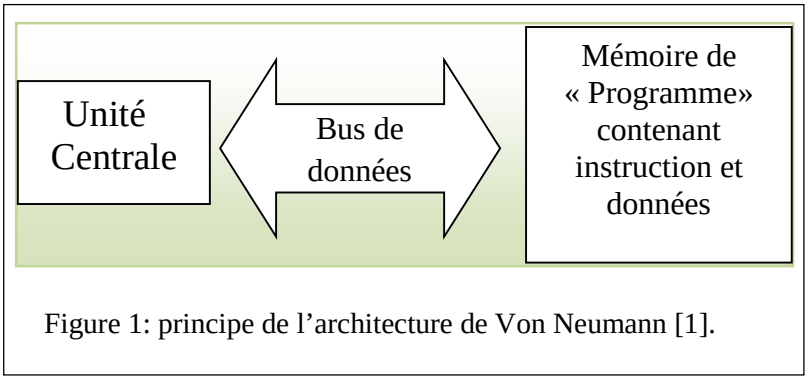


Figure 1: principe de l'architecture de Von Neumann [1].

C'est une architecture commune à celle que l'on rencontre habituellement dans les micro-ordinateurs. La mémoire, appelée improprement de programme, contient en fait des instructions et des données placées à la suite les unes des autres et on ne dispose que d'un bus, appelé bus de donnés qui leurs sont associes comme le montre la figure 1, cette architecture donne toute satisfaction mais elle pose quelques problèmes dès que l'on veut faire fonctionner l'ensemble rapidement.

En effet, l'exécution d'une seule instruction nécessite plusieurs échanges de donnés sur le seul et unique bus dévolu à cet usage puisqu'il faut tout d'abord aller chercher le code de l'instruction, puis la ou les données qu'elle doit manipuler [1].

Architecture Harvard :

Dans l'architecture Harvard les instructions et les données sont clairement différenciées. Ces dernières sont contenues dans des mémoires différentes et sont véhiculées sur des bus indépendants comme le montre la figure 2.

En effet, l'exécution d'une instruction ne fait plus appel qu'à un seul cycle machine puisque l'on peut simultanément, grâce aux deux bus et aux deux mémoires indépendants, rechercher le code de l'instruction et la ou les données qu'elle manipule.

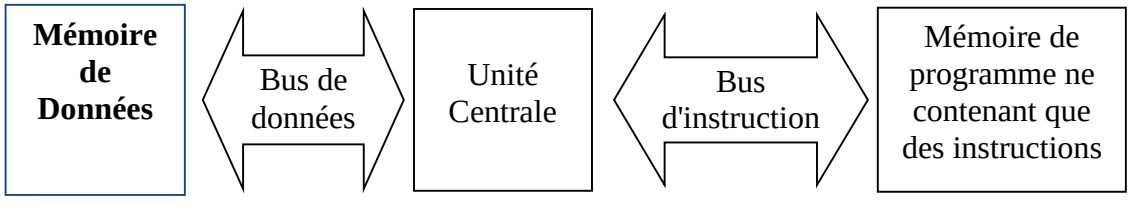


Figure 2 : principe de l'architecture de Harvard [1]

Rompant avec une tradition bien établie, les microcontrôleurs AVR d'Atmel utilisent une architecture Harvard mais aussi, ils font également appel à une architecture de type RISC.

4- Les points forts de l'architecture RISC

RISC signifie Reduced Instruction Set Computer ce qui veut effectivement dire circuit à jeu d'instructions réduit mais ce n'est pas tout. Un vrai circuit de type RISC doit en effet présenter un certain nombre de particularités propre à accroître sa vitesse de fonctionnement.

Les microcontrôleurs à architecture RISC utilisent des instructions codées sur un seul mot. Cela présente deux avantages : le premier est que tous les emplacements de la mémoire de programme contiennent une instruction, le second et qu'un seul cycle machine suffit pour lire le code complet d'une instruction, d'où un gain en vitesse d'exécution.

Le circuit RISC utilise ensuite une structure de type pipe-line que leur permet d'exécuter une instruction tout en cherchant la suivante en mémoire d'où, là encore, accroissement de la vitesse d'exécution.

Les circuits RISC exécutent tout les instructions en un seul cycle machine ce qui est du en grande partie au codage de l'instruction sur un seul mot. L'unité arithmétique ou logique, appelée encore ALU, dispose en effet en une seule fois de toutes les informations nécessaires à l'exécution de l'instruction [1].

5- Les différentes familles AVR

La famille AVR 8bits (ATMEL) regroupe une centaine de composants, chacun disponible dans plusieurs types de boîtiers physique (package). La famille AVR peut être découpée en 3 sous familles principales :

– Tiny AVR (8 à 20 broches)

– Méga AVR (28 à 100 broches)

– XMega AVR (44 à 100 broches)

En effet, même s'il est bien évident que l'on ne trouve pas les mêmes ressources internes dans un ATtiny10, qui est le plus petit modèle de la gamme, que dans un ATmega103 qui est, l'heure actuelle, des circuits AVR ; tous ces circuits font appel aux même sous-ensemble de base . Complétés par des familles développées des marchés spécifiques :

Automotive AVR : microcontrôleurs AVR dédiés industrie automobile

Lightning AVR : microcontrôleurs AVR dédiés éclairage intelligent

LCD AVR : microcontrôleurs AVR dédiés contrôle d'affichage LCD

USB AVR : microcontrôleurs AVR dédiés bus USB

Zlink AVR : microcontrôleurs AVR avec bus Zig Bee (Radio 2. 4Ghz)

Ces composants nombreux, se différencient par la quantité de mémoires (Données/Programmes) et le nombre et les types de périphériques qu'ils intègrent :

– Mémoire FLASH (programmes) de 64 (Tiny11) a 256koctets (ATMEGA256)

– Mémoire RAM (données) : de 0 (Tiny11) à 16kO (XMEGA)

– jusqu'à 8 ports de 8bits en E/Sortie Génériques

– Mémoire EEPROM de taille variable

– le nombre de PWM permettant de commander des moteurs

– Le nombre de timers/compteurs

– la variété des liaisons Séries (UART, SPI, I2C, . . .)

– la présence de convertisseurs analogique/numérique [7]

6- Caractéristique communes qui partage différents microcontrôleurs de la famille AVR :

Les instructions (assembleurs) ont une taille fixe de 16bits (ou 32 bits quelque fois)

– La plupart des instructions (assembleurs) sont exécutées en 1 cycle d'horloge (sauf ADDW et MULS et les instructions de sauts)

– La CPU contient 32 registres 8bits de travail. C'est sur ces registres que peuvent porter les instructions arithmétiques et logiques.

– 64 registres 8bits permettent de contrôler/configurer les périphériques et le cœur du microcontrôleur [1]

7- Présentation de l'ATMega 8

L'ATMega8 est un composant intermédiaire de la famille AVR 8 bits de taille raisonnable (28 broches), qui existe en boîtier DIP et qui est très bien fourni en périphériques.

– Disponible en boîtier DIP28

– 1K octets de SRAM

– 512 octets EEPROM

– 23 lignes d'entrée/sorties

– 8Ko de mémoire Flash

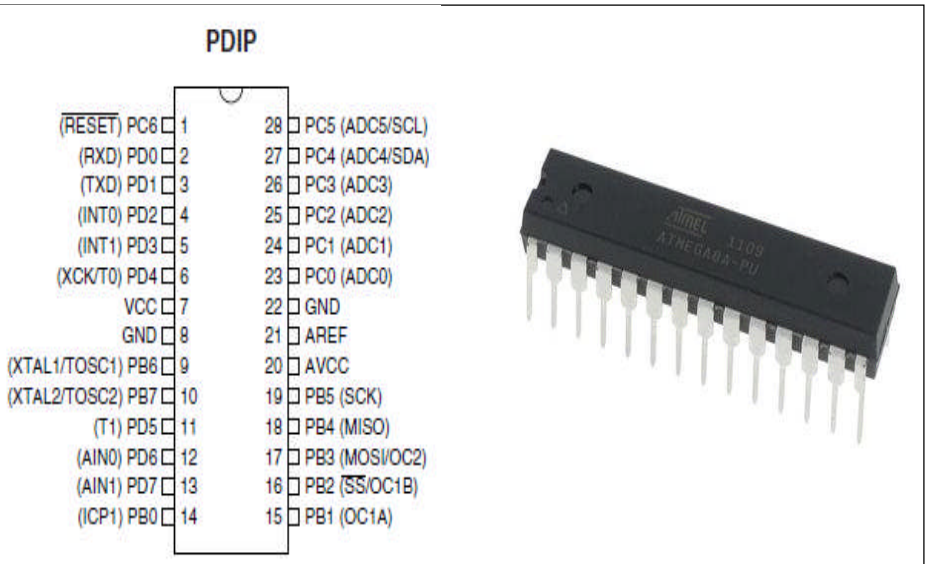


Figure 3 : brochage de l'ATmega8 avec image [8][9].

Le brochage (DIP) est donné sur la figure 3. Les noms entre parenthèses correspondent à une 2ème/3ème fonctionnalité de la broche. Ces fonctionnalités sont configurables (activables) de manière logicielle. Les broches indispensables pour câbler un ATmega

– VCC : +5V (fonctionne jusqu'à 2. 7V)
– GND : masse
– RESET : Réinitialisation (actif à l'état BAS, à connecter a +5V pour fonctionnement)
– XTAL1/XTAL2 Quartz ou entrée d'horloge (fré < 16MHz sous 5V)
Pour citer quelques unes des fonctionnalités :
-Port C (PC0, PC5. .) C est un port bidirectionnel port I / O à 7 bits
-Port D (PD7. . PD0) D est un port bidirectionnel port d'E / S à 8 bits
- PBx, PCx, PDx Port B,C,D : Entrées Sorties pour chaque port
-ADC0, ADC5 Seconde fonctionnalité du port C = Entrée analogique
- /SS, MOSI, MISO, SCLK Entrée/sortie série synchrone SPI (PB2/PB5)
-INT0/INT1/INT2 Entrée d'interruption du processeur par un événement externe
-RXD/TXD Liaison série Asynchrone RS232 (communication PC)
AREF est l'axe de référence analogique pour le convertisseur A / D
AVCC est la broche de tension d'alimentation pour le convertisseur

L'architecture interne du microcontrôleur est donnée sur figure [4]. Elle est représentative des microcontrôleurs AVR. Suivant le microcontrôleur (ATmega, tiny) certains périphériques disparaîtront ou apparaîtront, mais sans changer la structure interne. Au centre à gauche : le

CPU, c'est le cœur du microcontrôleur, c'est le CPU qui exécute les instructions. On y reconnaît l'architecture Harvard avec ses mémoires données (SRAM) et programme (PROGRAM FLASH) séparées.

En haut et en bas : les 3 ports d'entrées sorties B C D : On peut y remarquer les doubles fonctionnalités des broches. Par exemple, les broches du port C sont également connectées au bloc de conversion analogique numérique (MUX&ADC).

A droite les blocs fonctionnels de contrôles du microcontrôleur et de certains périphériques :

– TIMER,

-Clock source,

– Contrôleur d'interruption (INTERRUPT)

–Liaisons séries TWI, USART, SPI

Au milieu, un bus permettant le transfert de données entre les périphériques/mémoires et la CPU

8- Architecture interne

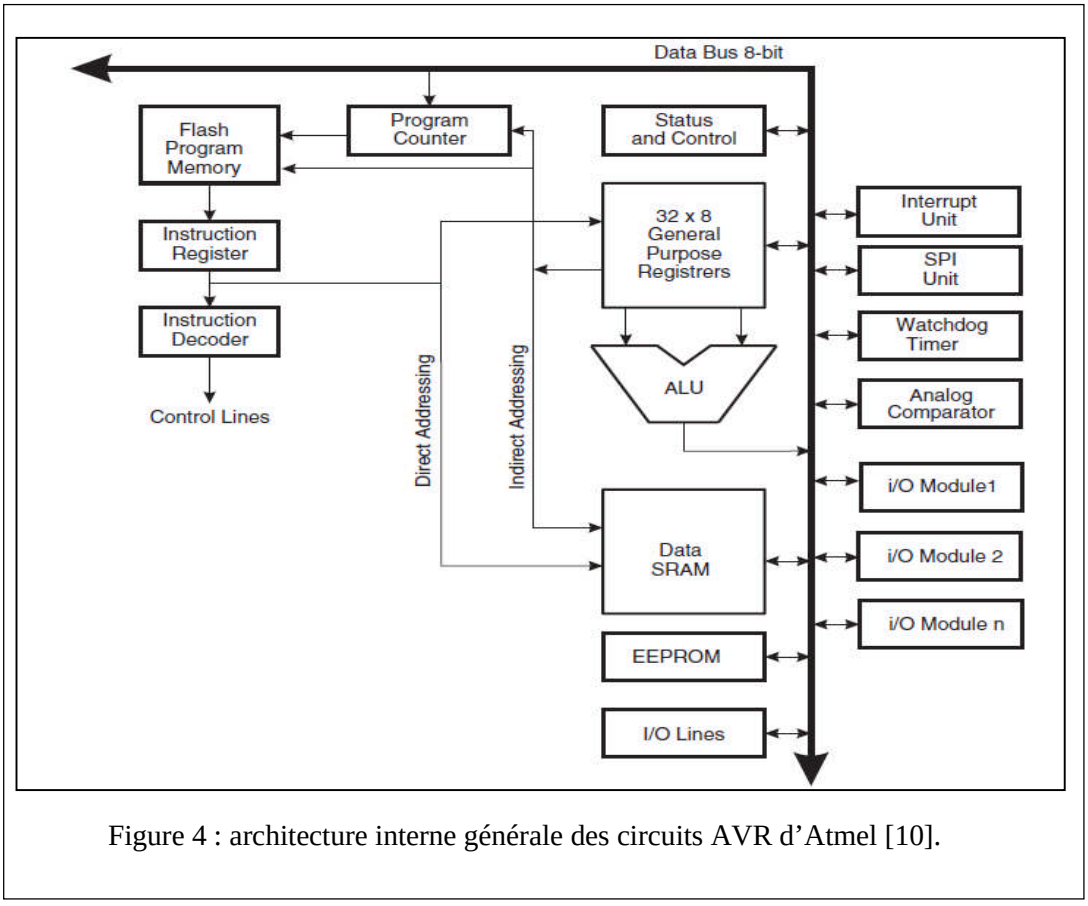


Figure 4 : architecture interne générale des circuits AVR d'Atmel [10].

Tous les circuits de la famille AVR adoptent la même architecture dont un exemple synoptique est présenté en figure 4.

9- L'Horloge Système et Option

Le choix du type d'horloge est déterminé lors de la programmation de la mémoire FLASH du Microcontrôleur et peux être choisi parmi les données du tableau suivant :

Dispositif d'Option d'horloge

Résonateur Externe Cristal/Céramique
Cristal Basse Fréquence Externe
Oscillateur Externe à RC
Oscillateur Calibré Interne à RC
Horloge Externe

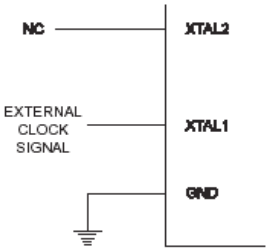


Figure 5 : schéma de l'horloge à quartz ou à résonateur céramique [10].

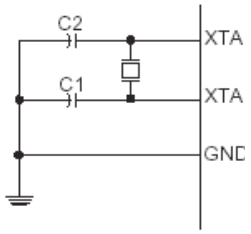


Figure 6 : signale d'horloge externe [10].

Résonateur Externe Cristal/Céramique

Tous les circuits de la famille AVR, sauf l'AT90S2343 sont prévus pour fonctionner en mode horloge à quartz, pilotée par un quartz ou résonateur céramique externe comme schématisé figure 5. La fréquence minimum du quartz n'est pas limitée par le microcontrôleur car ce dernier est de type statique et peut donc fonctionner avec des horloges aussi lentes que vous le souhaitez. Les valeurs exactes des condensateurs C1 et C2 dépendent des caractéristiques précises du quartz utilisé. Des condensateurs de 22pF environ donnent généralement satisfaction.

L'horloge Interne

Pour conduire un dispositif externe avec la source d'horloge interne, XTAL2 doit être laissée en l'air, tandis que l'on connecte XTAL1 au dispositif à piloter, comme indiqué dans la Figure 6 .

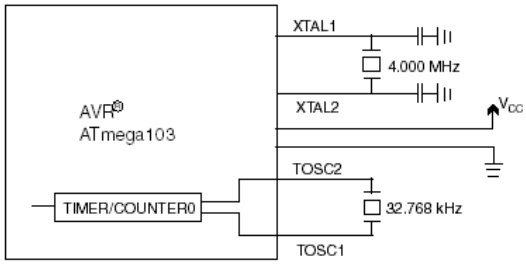


Figure 7 : exemple de connexion d'horloge temps réel [10].

L'oscillateur du Timer

Pour l'oscillateur du Timer temps réel, les broches TOSC1 et TOSC2 (PC6 & PC7) peuvent recevoir

un cristal de 32768 Hz directement connecté entre ces broches sans condensateur externe. L'application d'une source d'horloge externe sur TOSC1 n'est pas recommandée. Ce système permet de cadencer le Timer avec des valeurs de temps en sous-multiple de la seconde.

10 Le Plan Mémoire

Trois types de mémoire sont utilisés dans la série ATmega, la mémoire programme : FLASH, la mémoire de donnée SRAM et la mémoire morte de type EEPROM.

La mémoire programme

Touts les microcontrôleurs de la famille AVR d'Atmel dispose d'une mémoire EEPROM de donnés dont l'endurance minimum garantie, c'est-à-dire encore le nombre de cycles d'effacement /écriture, est de 100 000 cycle.

La taille de la mémoire EEPROM de données pour la famille AVR varie de 64 octets à 4 Koctets. La figure 8 donne un exemple de l'adressage de la mémoire FLASH du modèle ATMEGA32.

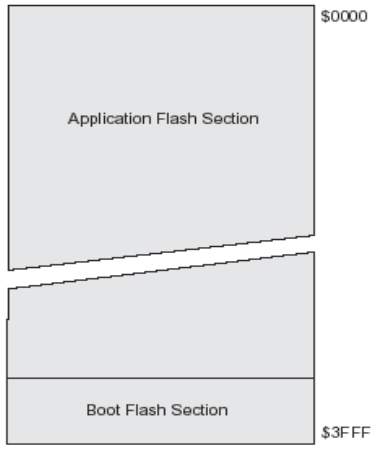


Figure 8 : adressage de la mémoire FLASH [10].

La mémoire de donnée

La mémoire de donnée contient les 32 registres de travail, les 64 registres de commande et la mémoire SRAM pour les variables du programme de 2048 octets pour le modèle ATMEGA32. La figure 9 présente les relations entre espace physique et registre.

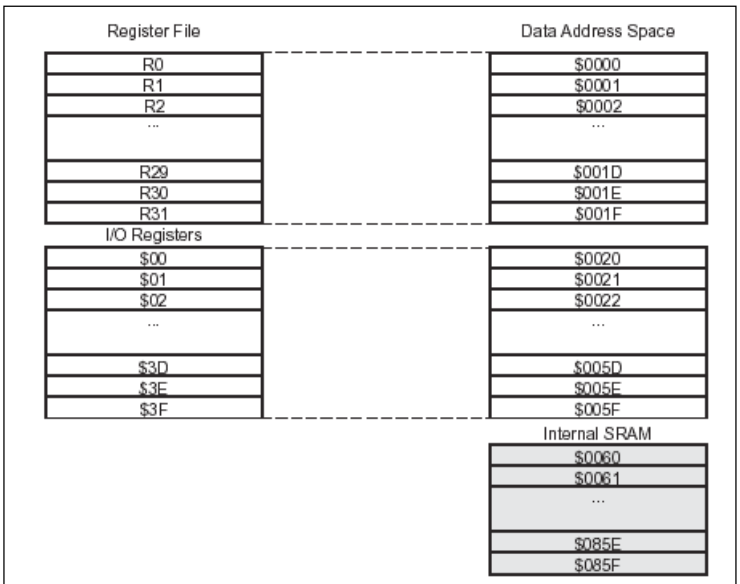


Figure 9 : la mémoire de donnée avec la SRAM [10].

La mémoire morte

La mémoire morte est de type EEPROM d'accès plus complexe contiendra la configuration du

programme et les données importantes qui seront sauvé pendant l'absence de courant électrique. On peut écrire jusqu'à 100.000 fois dans l'EEPROM. La taille de l'EEPROM est fonction du modèle de microcontrôleur ATMEGA (de 256 bits à 8 Ko).

12- Convertisseur Analogique / Numérique (ADC)

Le convertisseur analogique/numérique ADC intégré dans l'ATMEGA est doté de caractéristiques très intéressante avec une résolution de 10 bits, 8 entrées simultanées avec une non linéarité inférieur à 1/2 LSB, une erreur à 0 V inférieur à 1 LSB, le temps de conversion est réglable de 65 à 260 µS. Plus le temps est long, plus le résultat est précis. Le convertisseur étant chargé de convertir une tension analogique en résultat numérique (digital) codé sur 10 bits [10].

13- Les timers

a- Timer/Compteur0 à 8 Bits

Un Timer/Compteur est un ensemble logique qui permet d'effectuer du comptage : de temps, d'évènements, de base de temps pour la génération de signaux. Le Timer/Compteur0 est un compteur à 8 bits avec les particularités principales :

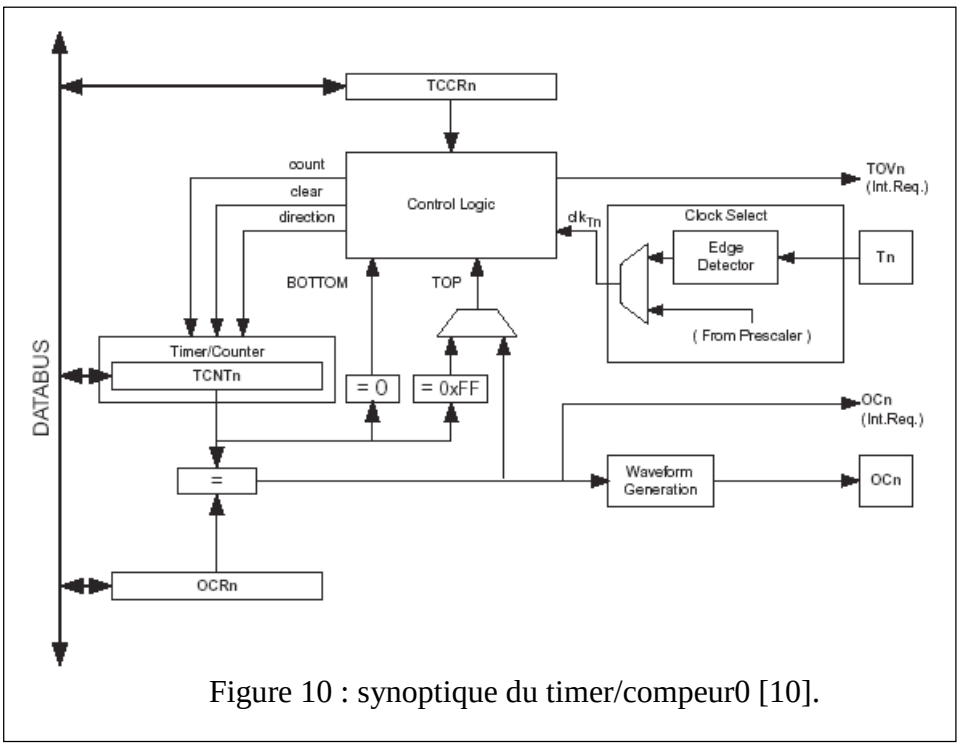


Figure 10 : synoptique du timer/compeur0 [10].

Remise à 0 du compteur sur Comparaison (Rechargement Automatique)

Générateur de Modulation de Phase en Largeur d'Impulsion (PWM)

Générateur de Fréquence

Compteur d'Événement Externe

Horloge à 9 bits avec pré diviseur

Interruption sur Débordent et Comparaison (TOV0 et OCF0).

b- Timer/Compteur1 à 16 Bits

L'unité de Timer/Compteur 1 à 16 bits permet le cadencement précis des programmes, la gestion d'événement, la génération d'onde, etc. 15 modes sont disponibles avec le Timer et les particularités principales sont :

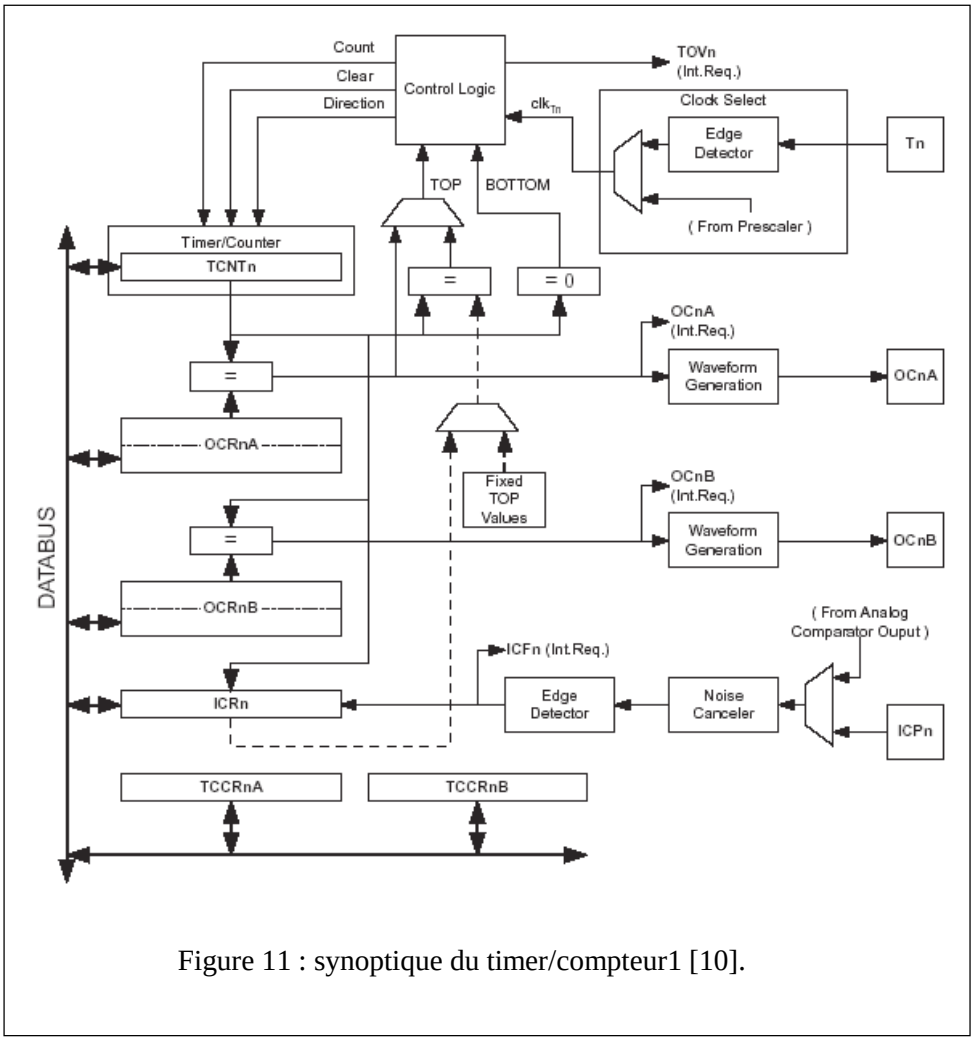


Figure 11 : synoptique du timer/compteur1 [10].

Une Conception 16 bits totale

Deux Unités de Comparaison Indépendante

Double Comparateur

Une Unité de Capture d'Entrée à faible Bruit

Comparateur avec remise à 0 Automatique

Générateur de Modulation de Phase en Largeur d'Impulsion Correct (PWM)

Générateur d’onde PWM Périodique

Générateur de Fréquence

Compteur d'Événement Externe

Quatre Sources Indépendant d’Interruption (TOV1, OCF1A, OCF1B et ICF1).

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

14- Les Interfaces Séries Synchrone et Asynchrone

a- L’Interface Série Synchrone SPI

L’interface SPI est l'abréviation en anglais de Serial Peripheral Interfacel. L’interface SPI permet le transfert de données rapide synchrone entre l'ATMEGA et d’autres périphériques soit en mode Full Duplex ou bien avec LSB d'abord ou MSB en premier.

L’interface SPI fonction au mode maître et esclave en fonction de l’état de la broche ‘/SS’. Donc c’est une interface série qui comporte quatre broches (4 fils)

-**MISO** : Master In Slave Out entrée des données série sur le Maître, sorties sur l'Esclave

-**MOSI** : Master Out Slave sortie des données série pour le Maître, entrée pour l'esclavage

-**SCK** : Serial Clock signaux d'horloge. Ces signaux d'horloges sont générés par le Maître.

-**SS** : Slave Select sélection du mode Maître de la liaison, le Maître la met à l’état haut.

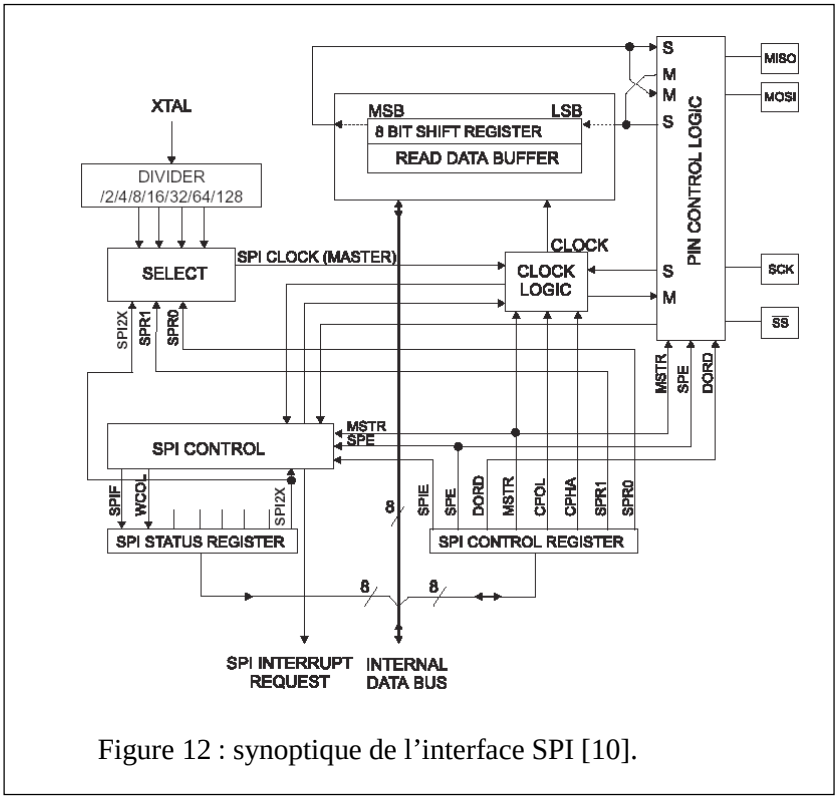


Figure 12 : synoptique de l’interface SPI [10].

b- L’interface Série USART

L'USART est l'abréviation de Universal Synchronous and Asynchronous Receiver and Transmitter. C’est une interface Série Synchrone et Asynchrone qui composée de trois partie principal : Générateur d'Horloge, Émetteur et Récepteur. L'application principale de ce périphérique est la communication entre le microcontrôleur et un ordinateur via le port série RS232.

c- L’interface I2C (TWI)

L'interface à deux conducteurs périodique TWI emplois seulement deux lignes : un pour l'horloge SCL et un pour les données SDA, et chaque ligne dispose d’une résistance de valeur (4,7 à 5 k ohm) connectées à la tension positive.

Tous les systèmes I2C Esclave sont en drain ouvert et exécutent une fonction ‘ET’. Un niveau bas sur une ligne du bus TWI est produit quand un ou plusieurs systèmes TWI sont commandés. Un haut niveau est produit quand tous les systèmes TWI sont en mode tri états, permettant aux résistances de tirer la ligne vers le haut [10].

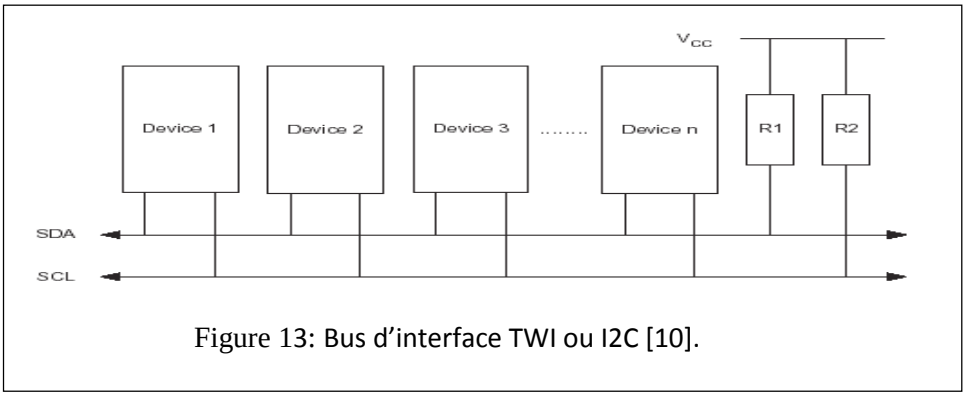


Figure 13: Bus d’interface TWI ou I2C [10].

15- Conclusion

Les Microcontrôleurs d'Atmel offrent une combinaison unique de performance, d'efficacité, de puissance et de flexibilité de conception. Avec leur facilité d'utilisation, faible consommation, et un haut niveau de l'intégration poussé, ils sont très indiqués pour des applications de différents types. Ils sont optimisés pour accélérer le temps de mise sur le marché des produits. Ils sont basés sur une architecture où le code est plus dense que la programmation se fasse en C ou en assembleur.

Chapitre II :

Généralités sur les moteurs et les codeurs

1- Moteur à courant continue

Beaucoup d'applications nécessitent un couple de démarrage élevé. Le Moteur à Courant Continu (MCC) possède une caractéristique couple/vitesse de pente importante, ce qui permet de vaincre un couple résistant élevé et d'absorber les à coups de charge : la vitesse du moteur s'adapte à sa charge. Le moteur à courant continu est utilisé quand on dispose d'une source d'alimentation continue (batterie par exemple). Il se caractérise par des lois de fonctionnement linéaires qui rendent l'exploitation de ses caractéristiques faciles d'emploi [11].

Figure 14 : moteur à courant continu [11]

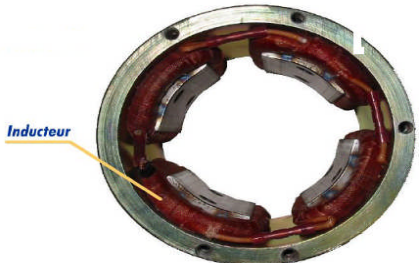


Figure15 : image de l'inducteur [12].

Figure16 : image de l'induit et de collecteur [12].

2- Constitution :

Le moteur à courant continu est composé des éléments suivants :

Un inducteur : appelé aussi stator.

Un induit : appelé aussi rotor.

Le rotor cylindrique est compose de tôles isolées entre elle et munies d'encoches dans les quelles sont réparties les conducteurs parcourus par un courant,

ceux –ci créent le champ magnétique dit rotorique.

Un collecteur fixé à l'induit, il est en contact avec les charbons. Des charbons appelés aussi balais, ils alimentent l'induit par le collecteur sur lequel ils frottent.

3- Excitation des moteurs à courant continu

Suivant l'application, les bobinages du l'inducteur et de l'induit peuvent être connectés de manière différente.

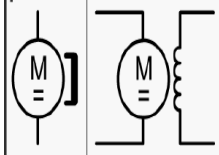
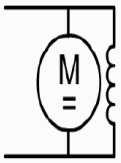
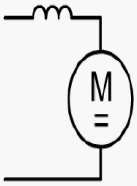
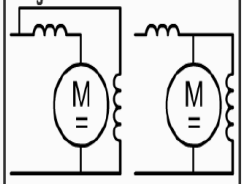
Excitation séparée	Excitation dérivation *(moteur shunt)*	Excitation série *(moteur universel)*	Excitation composée *(moteur compound)*
aimants permanents / inducteur bobiné 			Longue dérivation / Courte dérivation 
Dans le cas d'un induteur bobiné, nécessite deux alimentations. Petites puissances pour les moteurs à aimants permanents	Vitesse relativement constante, quelle que soit la charge (autorégulateur de vitesse). Absence d'emballement à vide. Couple de démarrage moyen.	Fort couple à basse vitesse. Autorégulateur de puissance : la vitesse décroit lorsque la charge augmente. Risque d'emballement à vide.	Couple de démarrage meilleur qu'en dérivation mais plus faible qu'en série. Vitesse relativement stable, quelle que soit la charge. Absence d'emballement à vide.

Tableau 1 : Quelques propriétés des moteurs [12].

4- Moteur à courant continu à aimants permanents

Cette technologie de moteur permet une réalisation économique de moteurs, en général de faible puissance pour des usages multiples : automobile, audiovisuel, robotique etc.

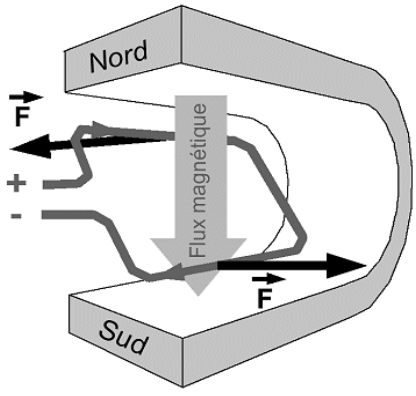


Figure 17 : schéma d'un aimant permanent [13].

5- Principe de fonctionnement

Si un conducteur forme de spire, parcouru par un courant I, est placé dans un champ magnétique, il est soumis à des forces de Laplace. Ces forces créent un couple de

rotation qui fait tourner la spire sur son axe. Quand la spire a fait un demi tour, il faut inverser la polarité pour inverser le sens des forces et continuer le mouvement. Ce sera le rôle du collecteur.

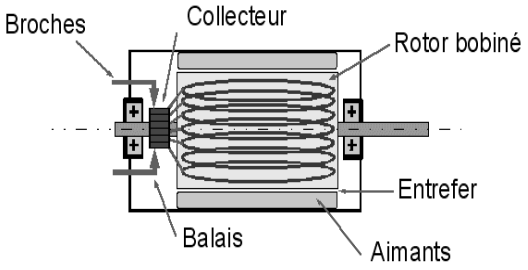


Figure 18 : schéma d'un moteur à aimant permanent [13].

Le rotor, (partie tournante), est constitué d'un noyau métallique avec un bobinage de cuivre, le stator comporte des aimants permanents qui engendrent un champ magnétique dont le flux traverse le rotor. L'espace étroit entre le rotor et le stator est nommé entrefer.

Le moteur se comporte comme une résistance en série avec un générateur de tension (fem : force électromotrice)

I : courant consommé par le moteur
U : Tension d'alimentation du moteur
E : force électromotrice
R : résistance interne du bobinage

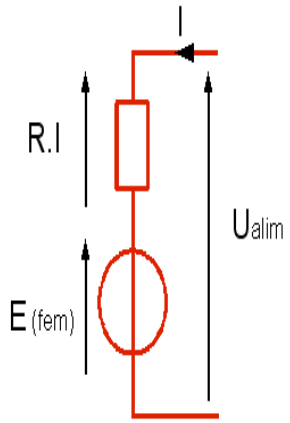


Figure 19 : Schéma equivalent d'un moteur [13].

6- Les équations caractéristiques du moteur sont les suivantes :

$U = E + RI$ qui découle directement du schéma de la figure 19.

$E = k\,\omega$ qui relie la fém à la vitesse angulaire ω (en *rad/s*) par une constante k .

$C = K\,I$ qui relie le couple de résistance (en m N) au courant I consommé par le moteur.

Les constantes k et K caractérisent le moteur.

On peut en déduire que :

- Pour faire varier la fréquence de rotation, il faut faire varier E et donc la tension d'alimentation U.

- Pour inverser le sens de rotation, il faut inverser E et donc la tension d'alimentation à ses bornes.

- Le courant varie avec le couple, on peut aussi limiter le courant pour limiter le couple [13].

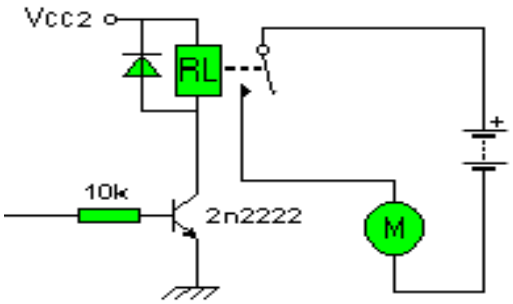


Figure 20 : commande de moteur avec un relais [13].

7- Alimentation du moteur

Le moteur peut être alimenté simplement via un relais électromécanique ou par un transistor associé à une diode de roue libre comme le montre la figure.

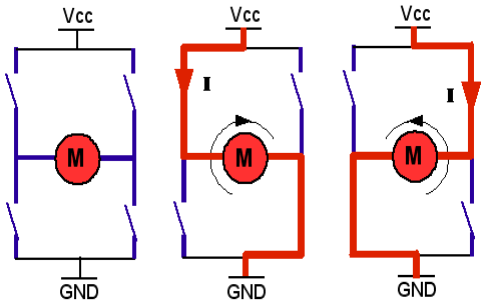


Figure21: schéma de pont en H[13]

8- Commande de moteur à deux sens de rotation

Le plus souvent, le moteur à besoin de tourner dans les deux sens de rotation on utilise alors un dispositif nommé pont en H, dans nous projet on a utilisé des relais. Il suffit de fermer deux des contacts pour faire tourner le moteur dans un sens ou dans l'autre. On utilise le plus souvent des transistors PNP et NPN, ou des MOS, ou bien des relais.

Dans notre manipulation, on a utilisé la sortie de broche PD6 du microcontrôleur ATMEGA pour commander le transistor.

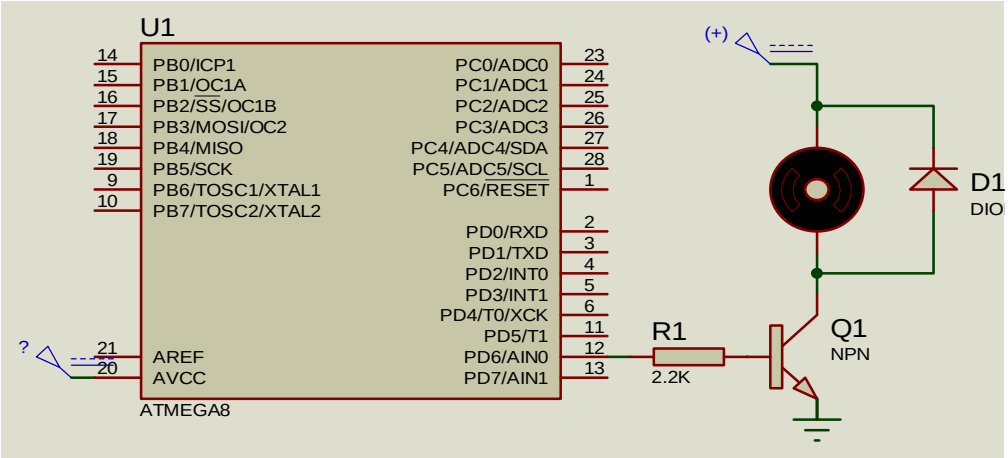


Figure 22 : schéma de commande de moteur réalisé sous ISIS.

9- Variation de vitesse du moteur

Pour faire varier la vitesse d'un moteur en faisant varier la tension d'alimentation à ses bornes. On a utilise une résistance variable. Pour chaque changement, on mesure la tension et la vitesse comme montre le tableau [4] dans le chapitre V.

10- Puissance et rendement

Les puissances intervenantes sur un système à moteur sont :

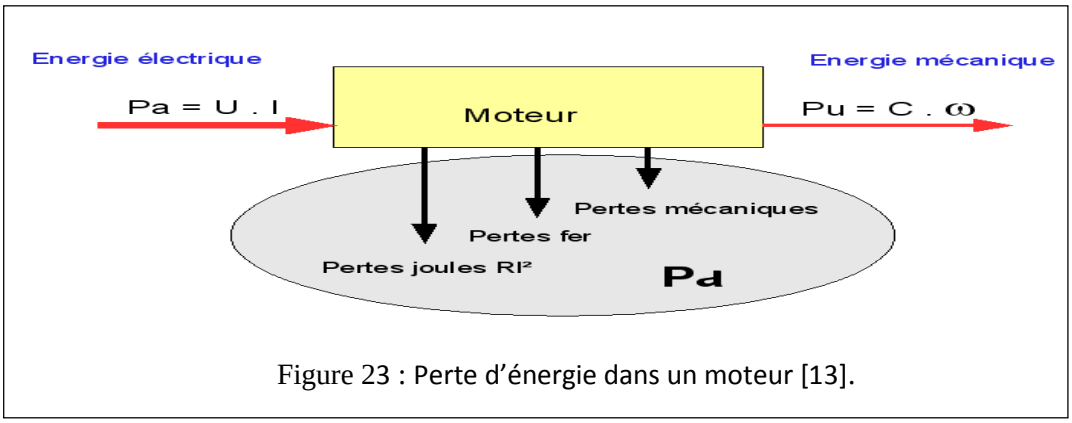


Figure 23 : Perte d'énergie dans un moteur [13].

Puissance absorbée (Watt) : c'est la puissance électrique prélevée sur l'alimentation Pa = U. I

Puissance utile (Watt) : c'est la puissance mécanique disponible sur l'arbre Pu = C.

Puissance dissipée (Watt) : cette puissance correspond aux pertes électriques par effet joules (R. I^2), aux pertes mécaniques et aux pertes magnétiques, La puissance utile est donc toujours plus faible que la puissance absorbée Pd = Pa – Pu

Le rendement (sans unité) est le rapport entre la puissance utile et la puissance absorbée. Il est toujours inférieur à 1 (μ = Pu / Pa) [13].

11- Moteur pas à pas

Les moteurs pas à pas sont différents par rapport aux moteurs à courant continu de leurs structures et du mode fonctionnements. Par exemple, ce moteur ne dispose pas de balais-collecteur comme le MCC.

Le moteur pas à pas apparaît comme l'interface numérique-mécanique idéale car il est commandé par des impulsions ou des états logiques. Contrairement aux autres moteurs, où la rotation est continue, la rotation est discrète et saccadée.

12- Différents types de moteur pas à pas

On rencontre 3 types de moteur pas à pas : à aimant permanent, à réluctance variable et hybride qui se différencient par le type de rotor. On distingue des moteurs pas à pas bipolaire et unipolaire.

Les bipolaires: ils ont 4 fils, avec deux bobines indépendantes

Les unipolaires : 5 fils, 1 commun aux 4 bobines, plus un par bobine.

– 6 fils, 2 bobines avec point milieu (parfois appelé « 2 phases »).

– 8 fils, 4 bobines indépendantes, également appelé universel [14].

a- Moteur pas-à-pas à réluctance variable MRV

Le moteur pas-à-pas réluctant (Variable reluctance motor VR) comporte n bobines distinctes au stator et un rotor ferromagnétique sans aimant, sans courant dans les bobinages statoriques.

Les avantages de ces moteurs résident dans l'absence d'aimants qui les rendent très économiques et aptes aux environnements très sévères [15].

b- Moteur pas à pas à aimants permanents MP

Dans ce type de moteur, on introduit les aiment.

c- Moteur Hybrides

Moteur pas à pas hybride est une combinaison de deux structures précédentes du moteur à réluctance variable et du moteur à aimant permanent. Cette combinaison permet de réunissent les avantages des deux moteurs précédents [15].

13- Avantages, Inconvénients du moteur pas à pas

Les Avantages

-dispose pas de balais-collecteur, pas d'entretien donc une grande durée de vie.
-grande stabilité en vitesse.

-Bien adapté au numérique.

-Inertie élevée du moteur.

-Asservissement de position ou de vitesse en boucle ouverte

Les inconvénients

-Faible puissance < 5 KW

-Faible rendement

-Positionnement discret

-Faible vitesse maximale

-Grand consommateur d'énergie

14- Moteur sans balais (brushless)

Les moteurs sans balais (brushless) permettent le cumul les avantages du moteur à courant continu. Leur conception est optimisé pour tourner à des vitesses élevées.

15- Composition du moteur brushless

Un moteur brushless comporte les mêmes éléments qu'un moteur à courant continu, sauf le collecteur. Donc, il est composé d'un stator et d'un rotor.

Stator : constitué d'un bobinage disposé en étoile ou en triangle.

Rotor : composé d'un ou plusieurs aimants permanents

Figure 24 : photo réelle de rotor et de stator d'un moteur sans balais outrunner [16].

16- Commande des moteurs sans balais (brushless)

Dans le moteur DC à charbons ou balais, le système de commutation réalisé mécaniquement par l'intermédiaire des lamelles du collecteur qui sont en contact avec le charbon qui permet de faire passer le courant dans les bobinages. Dans un moteur sans balais brushless, la commande est réalisée électroniquement. En effet, il faut connaître à chaque instant la position du rotor et envoyer le courant dans les groupes de bobines. Celle-ci transforme le courant continu en courant triphasé à fréquence variable et va alimenter successivement les bobines (enroulements) pour créer un champ tournant et donc la rotation.

17- Les différents types de moteurs brushless

Il existe une grande variété de moteurs brushless avec des couples, vitesses et inerties différents en fonction de leurs constitutions.

a- Moteurs sans balais ourtrunner

On appelle « outrunner » les moteurs brushless dont le rotor est autour du stator. Comme pour un moteur pas à pas, les moteurs brushless outrunners comprenant plus de 3 bobines et 2 pôles et ne font qu'une fraction du tour lorsque le champ a tourné de 180°. Leur fréquence de rotation est donc plus faible mais le couple très élevé. Ces moteurs brushless outrunners sont souvent utilisés dans des applications qui nécessitent un fort couple, car ils peuvent être reliés à la charge sans nécessiter de dispositif de réduction.

b- Moteurs sans balais inrunner

Les moteurs sans balais inrunners ont le rotor à l'intérieur du stator. Ils n'ont généralement qu'une seule paire de pôles sur le rotor et 3 bobines au stator. L'inertie du rotor est beaucoup plus faible que pour un moteur outrunner, et les vitesses atteintes par ce type de moteur sont beaucoup plus élevées (jusqu'à 7700 tr/min/V).

c- Moteurs sans balais disques

Le rotor et le stator peuvent également être constitués de deux disques faces à face, avec les rayons et les bobines répartis le long les rayons de ces deux disques [16].

18- Les avantages du moteur sans balais (brushless)

- Durée de vie et fiabilité : absence de frottement des charbons sur le collecteur, plus de parasites, plus d'échauffement du collecteur et de pertes dues à l'étincelage.
- Encombrement et poids : plus léger qu'un moteur DC (avec balai).
- Consommation électrique : pas de chute de tension au niveau du collecteur, Le rendement est bien supérieur.
- Niveau sonore : le bruit est assez réduit par rapport aux autres moteurs.
- Fort couple au démarrage [17].

19- Conclusion

Dans ce chapitre nous avons présenté quelques différents des moteurs électriques et un résumé sur leurs modes de fonctionnement. On a introduit :

- Les moteurs à courant continue
- Les moteurs pas à pas
- Les moteurs sans balais (brushless)

Le choix de moteur repose sur leurs caractéristiques telles que le couple, la vitesse et la puissance.

Les performances des moteurs pas à pas restent faibles : consommation élevée, couple et vitesse faibles (souvent inférieure à 3 000 tr / mn) [18].

Pour les moteurs a courant continue leur mise en œuvre est plus simple, car facilite de régler ou de faire varier leur vitesse, leur couple et leur sens de rotation.

Mais leur inconvénient réside dans les collecteurs qui imposent des ruptures de contact et provoque des arcs électriques à chaque commutation, qui peuvent créer des parasites dans le circuit d'alimentation.

Pour remédier à ces problèmes, on peut s'orienter ver les moteurs sans balais (brushless).

20- Les codeurs

La croissance de la puissance des systèmes de traitement ainsi que les impératifs de productivité dans tous les domaines de production industrielle poussent à l'acquisition et commandes continue, comme: le déplacement, la position, la vitesse des outils ou des produits.

Selon les besoins, les codeurs montés sur un système d'entraînement ont des tâches déférentes. Une des tâches principales d'un codeur que nous avons utilisé dans notre montage est de permettre de mesurer la vitesse et l'angle de rotation du moteur électrique alimenté en courant continu.

21- Principe de fonctionnement

Tous les codeurs optiques exploitent des principes de fonctionnement similaires. Ils sont constitués d'un disque comportant des zones opaques et des zones translucides. Le nombre de ces zones et leur disposition dépend de la nature du codeur et du type d'information que l'on souhaite obtenir.

Des diodes électroluminescentes (LED) émettent une lumière qui peut traverser les zones translucides. Des photo-transistors, situés de l'autre côté du disque en regard des LED, captent

cette lumière lorsqu'ils sont face à une ouverture et délivrent un signal électrique, image de la présence de cette ouverture [19].

22- Les types de codeurs optiques

Il existe deux types de codeurs optiques:

Les codeurs incrémentaux qui délivrent une information de déplacement angulaire du disque sous forme d'un train d'impulsions.

Les codeurs numériques de position (codeurs absolus), pour lesquels chaque position du disque correspond à une valeur numérique différente identifiable par la partie commande.

23- Le codeur incrémental

Le codeur incrémental est destiné à des applications où l'information de position est obtenue par la mesure du déplacement de l'objet. Le codeur délivre un train d'impulsions dont le nombre permet de déduire la valeur du déplacement ainsi que la vitesse car ce dernier est proportionnel à la fréquence des impulsions.

Il est constitué d'un disque comportant deux à trois pistes : A et Z.

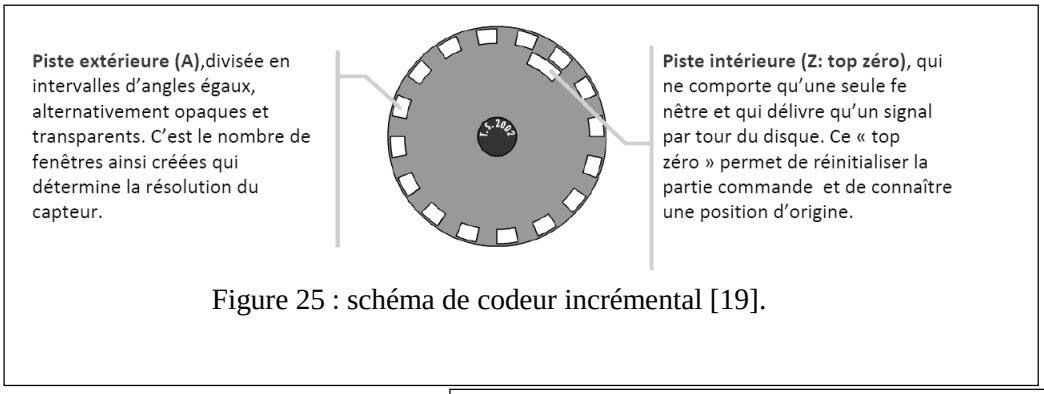


Figure 25 : schéma de codeur incrémental [19].

Pour un tour complet de l'axe du codeur, la partie commande reçoit autant d'impulsions électriques qu'il y a de fenêtres, dont la durée dépend de la vitesse de rotation du disque.

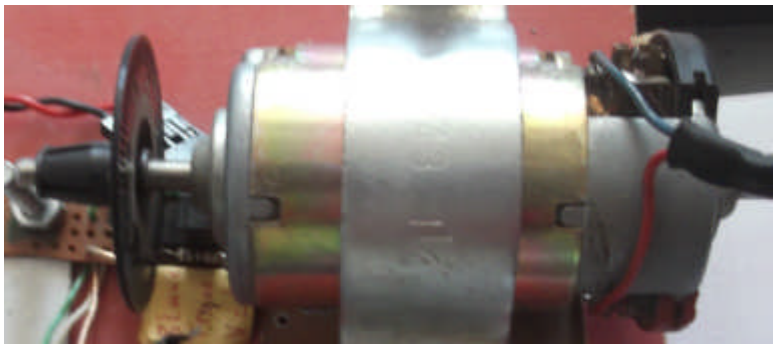


Figure : 26 image du codeur optique utilisé dans notre projet.

La figure 26 représente une image de l'un des codeurs optique utilisé dans notre projet

24- Particularités de fonctionnement

Un codeur incrémental possède trois têtes de lecture :

Une tête de lecture est affectée à la piste intérieure et délivre une impulsion par tour, permettant à la commande de compter le nombre d'impulsions reçues.

Deux têtes de lecture sont placées sur la piste extérieure. Chaque tête, prise isolement, permet à la partie commande de déterminer l'angle de rotation du disque en comptant le nombre d'impulsions reçues.

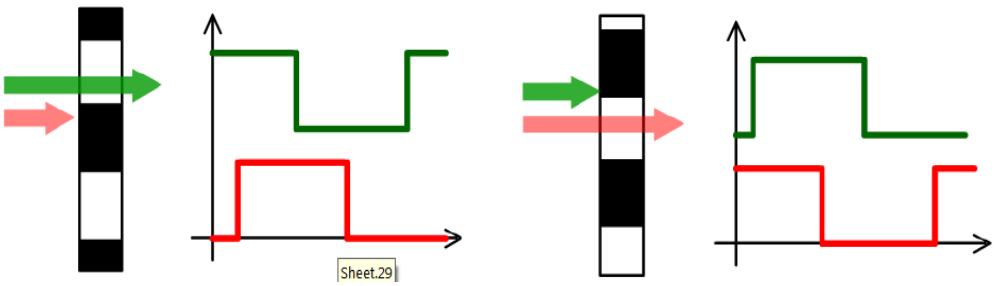


Figure 27 : schéma fonctionnelle d'un codeur incrémental [19]

Les deux têtes sont décalées l'une par rapport à l'autre d'un quart de largeur de fente. Ainsi, les signaux émis sont décalés dans le temps. La partie commande, en détectant quelle voie change d'état en premier peut déterminer le sens de rotation du disque.

Ce décalage permet de déterminer le sens de rotation :

- dans le sens de rotation 1, B = 0 au front montant de A.

- dans le sens de rotation 2, B = 1 au front montant de A.

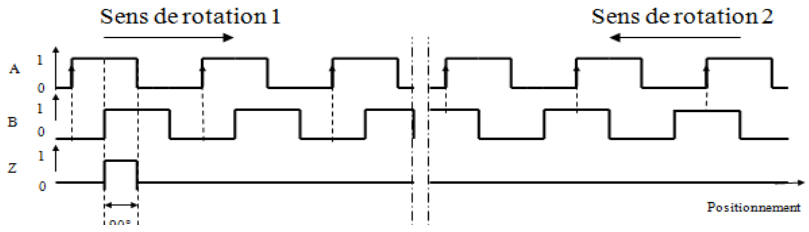


Figure 28 : schéma pour indique le sens de rotation [20].

25- Le codeur numérique (codeur absolu)

Il est constitué d'un disque comportant plusieurs pistes concentriques et d'une tête de lecture par piste. Le nombre de piste détermine le nombre de positions différentes qui peuvent être définies à l'intérieur d'un tour de disque. Les codeurs industriels comportent jusqu'à 24 pistes.

La partie commande reçoit directement un code numérique sur n bits (n étant le nombre de pistes), image de la position du disque à un instant donné. A l'intérieur d'un tour de disque, cette information est donc une information de position absolue (à la différence d'un codeur incrémental qui ne délivre qu'une information de déplacement par rapport à une origine qu'il a fallu définir au préalable) [21].

26- Avantages, Inconvénients des codeurs incrémentaux et absolus

a- Codeur incrémental

Avantages

Le codeur incrémental est de conception simple (son disque ne comporte que deux pistes) donc plus fiable et moins onéreux qu'un codeur absolu.

Inconvénients

Il est sensible aux coupures du réseau : chaque coupure du courant peut faire perdre la position réelle du mobile à l'unité de traitement. Il faudra alors procéder à la réinitialisation du système automatisé.

Il est sensible aux parasites en ligne, un parasite peut être comptabilisé par le système de traitement comme une impulsion délivrée par le codeur

b- Codeur absolu

Avantages

II est insensible aux coupures du réseau : la position du mobile est détenue dans une onde qui

est envoyé en parallèle au système de traitement.

L'information de position est donc disponible dès la mise sous tension.

Si le système de traitement «saute» une information de position délivrée par le codeur, la position réelle du mobile ne sera pas perdue car elle restera valide à la lecture suivante.

Inconvénients

II est de conception électrique et mécanique plus complexe aussi son coût sera plus élevé qu'un codeur incrémental.

Les informations de position sont délivrées « en parallèle » ; son utilisation mobilisera donc un nombre important d'entrées du système de traitement.

27- Choix de codeur

Un critère majeur pour le choix du codeur est la robustesse du système. En effet, certains codeurs sont montés directement sur le moteur ; ils doivent donc être insensible à la variation de température et aux vibrations afin de ne pas être les endommagés. Le niveau de sensibilité du codeur joue aussi un rôle primordial.

Chapitre III :

Installation et configuration du logiciel de développement

1- Ordre d'installation du logiciel de développement

L'installation doit se faire de préférence dans l'ordre donné ci-dessous afin que les librairies et outils soient automatiquement reconnus par AVR Studio.

– Pack d'outils WinAVR
– Pack de librairies AvrLib
– SP12 AVR Programmer
– Atmel AVR Studio 4

Installation

Lors de l'installation, on regroupe tous les outils dans un même répertoire (ex : c :\AVR). Ainsi, lors de la configuration, il est plus aisé de retrouver les différents composants. L'installation des outils se fait de façon automatique et ne requiert que le réglage du répertoire d'installation pour chacun des outils. SP12 requiert le redémarrage impératif de l'ordinateur pour permettre l'installation d'un pilote d'accès au port parallèle sous Windows 2000/7/NT/XP [22].

2- Création d'un projet

Lorsque l'installation des logiciels se termine, on peut directement utiliser AVR Studio pour commencer un nouveau projet.

Au lancement de AVR Studio, vous obtenez la fenêtre de la figure .

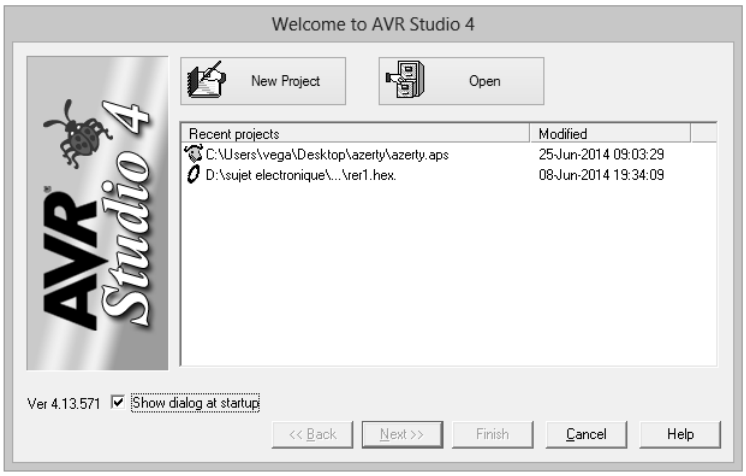


Figure 29 : fenêtre d'accueil de AVR Studio

Lorsque vous aurez crée un projet, celui-ci apparaîtra dans cette fenêtre au lancement et pourra ainsi facilement être rechargé.

3- Création d'un nouveau projet

Cliquez sur 'New Project' pour démarrer un nouveau projet. Vous obtiendrez alors la fenêtre suivante :

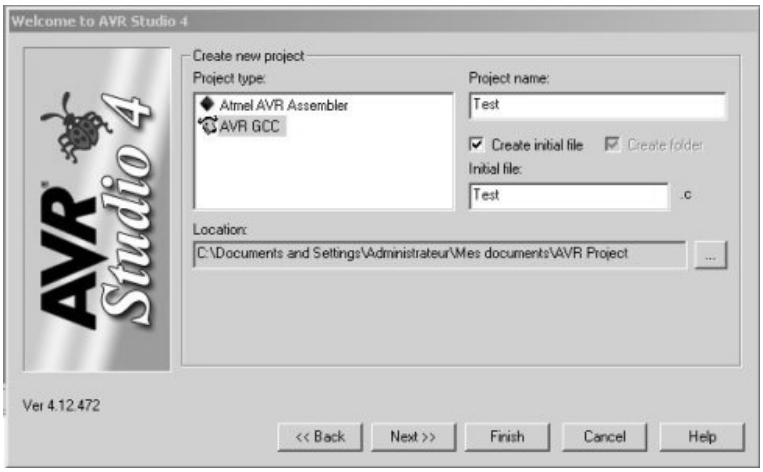


Figure 30 : sélection du type de projet .

Sélectionnez ’AVR GCC’ dans la liste ’Project Type’. Remplissez le champ ’Project name’ et choisissez le répertoire du projet dans le champ ’Location’. Une fois ces renseignements complétés, cliquez sur ’Next’.

Vous obtiendrez la fenêtre suivante :

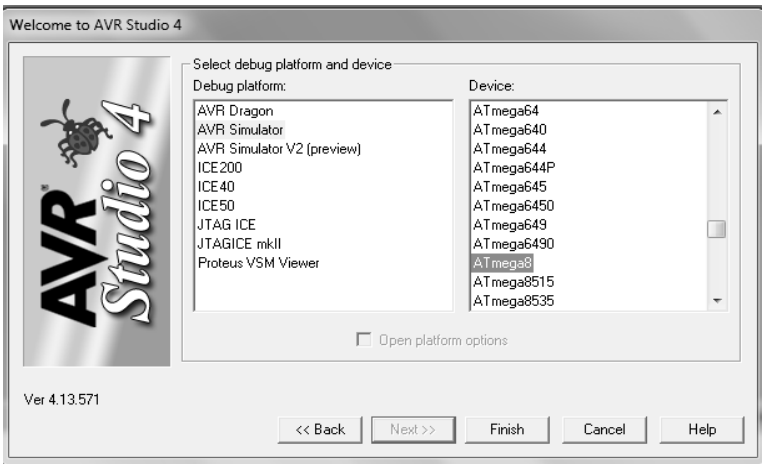


Figure 31 : sélection de la plateforme de déboguage.

Dans cette fenêtre, sélectionnez ’AVR Simulator’ dans la liste ’Debug Platform’. Dans la liste de droite, sélectionnez ’ATmega 8’ et cliquez sur le bouton ’Finish’. Vous obtiendrez la fenêtre suivante :

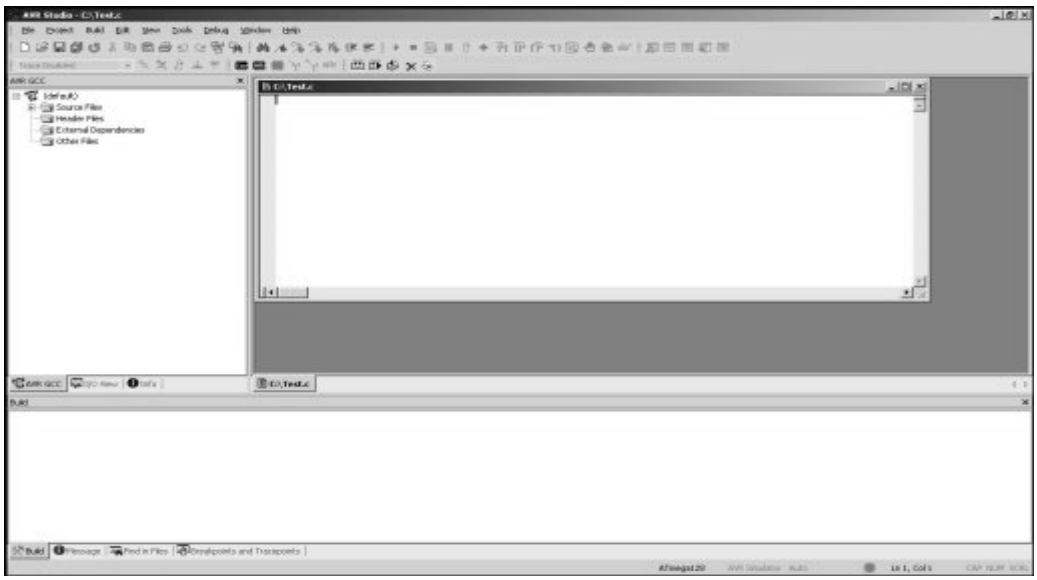


Figure 32: Fenêtre principale d’AVR Studio.

La création du projet est maintenant terminée. Pour tester les outils de compilations, entrez les lignes suivantes dans le fichier c :

```
int main(void)
{
        return 0 ;
}
```

Pour compiler ce programme, appuyez sur menu 'Build all'. Si les outils sont biens installés, la fenêtre 'Build all' doit s'afficher comme sur la figure.

Ce log de compilation résume les principales informations du programme (taille, pourcentage d'utilisation de la mémoire, erreurs, avertissements, etc). Les erreurs de compilation y seront affichées de même que les avertissements.

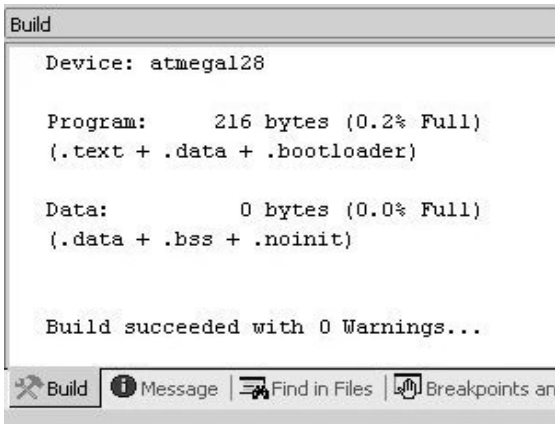


Figure 33 : Log de la compilation

4- Flashage

Lorsque le programme est compilé et n'affiche pas d'erreurs, on peut l'envoyer vers la mémoire flash de l'ATmega 8. On utilise avrdude. Au lieu d'utiliser le terminal et entrer plusieurs commandes, on préfère utiliser un script qui automatise la procédure. Le script suivant permet l'envoie du code vers le microcontrôleur à condition d'avoir un programmateur fonctionnel:

```
cd . .
cd . .
cd WinAVR
cd bin
cd C:\WinAVR\bin\
del C:\WinAVR\bin\Rer1.hex
copy C:\Users\moumou\Desktop\hhhh\default\ Rer1.hex C:\WinAVR\bin\Rer1.hex
avrdude -p atmega8 -P COM1 -c dasa -e -V -U flash:w:Rer1.hex
```

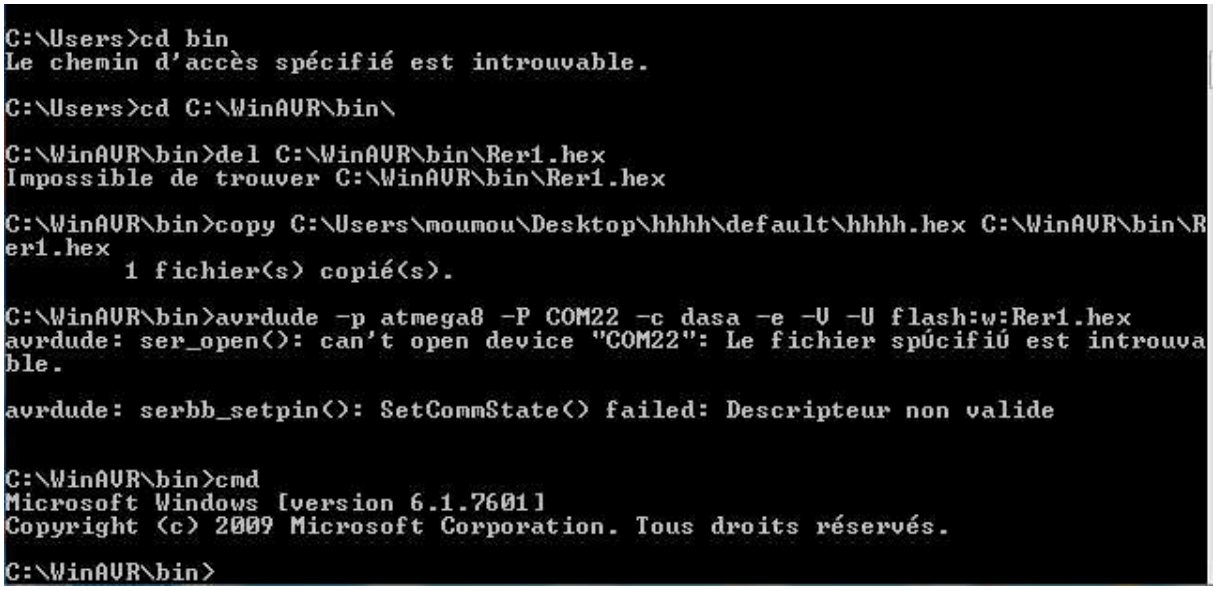


Figure 34 : Flashage de programme.

La commande essentielle: avrdude -p atmega8 -P COM1 -c dasa -e -V -U flash:w:Rer1.hex prend les paramètres suivant :

-p atmega8 : type de microcontrôleur utilisé

-P COM1 : type de périphérique (port commun)

c dasa -e -V –U : type de programmateur

flash:w:Rer1. hex : écriture à la mémoire de microcontrôleur avec code hexadécimal

5- Ecriture du programme

La programmation des microcontrôleurs avec des langages de hauts niveaux tels que le BASIC, PASCAL et C sont devenues les langages les plus populaires, car leur style de programmation et leur contexte est plus facile à apprendre et à mettre en œuvre [5].

Sous l'environnement Windows, AVRstudio fourni par Atmel jumelé à WinAVR (version win32 d'avr-gcc) permet de développer en langage C gratuitement et sans aucune limitation.

Le langage C reste un des langages les plus utilisés actuellement. Cela est dû au fait que le langage C est un langage puissant, riche en fonctionnalités et en outils de développement, comportant des instructions et des structures de haut niveau tout en générant un code très rapide grâce à un compilateur très performant.

6- Envoie de programme

Dans la phase terminale, une fois le fichier source compilé et simulé, il va falloir transférer dans la mémoire du microcontrôleur le fichier code-

machine. Pour cela, il faut un programmateur. La plus simple à mettre en œuvre pour communiquer avec un ordinateur et un montage électronique est d'utilisation du port série.

N°	Nom	Entrée/sortie	Description
1	CD	E	Carrier detect
2	RXD	E	Received data
3	TXD	S	Transmited data
4	DTR	S	Data Terminal Ready
5	GND		Masse
6	DSR	E	Data Set Ready
7	RTS	S	Request To send
8	CTS	E	Clear To send
9	RI	E	Ring Indicator

Tableau [3] : description des broches RS-232[2].

Sur les ordinateurs plus récents qui ne disposent pas de port série, il est possible d'utiliser un adaptateur USB –RS232 afin de convertir un port USB en port série. Un driver permet à l'ordinateur de « voir » le circuit comme un véritable port série. Les niveaux de tension sont aussi conformes à la norme RS232 avec des valeurs de ±10 V [2].

Ce schéma ci-bas représente un programmateur destiné pour les microcontrôleurs Atmel de la famille AVR. Ce programmateur est connecté à un PC via l'interface série RS232 et peut être utilisé avec beaucoup de logiciels comme PonyProg ou avrdude.

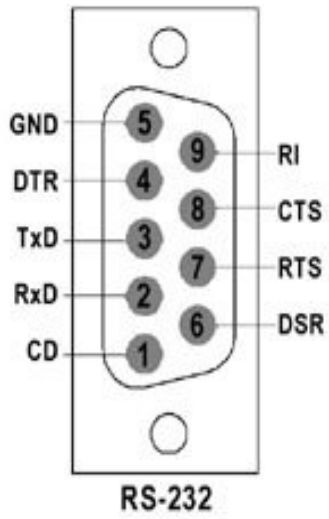


Figure 35 : Schémas de RS- 232 [27]. [[27]

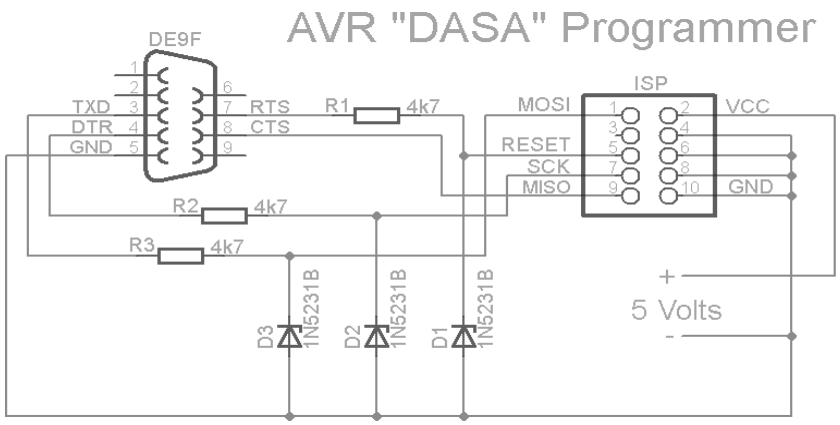


Figure 36 : Schéma électrique de programmateur [23].

7- Description du programmeur série

Le programmateur est assez simple à construire et ne nécessite que peu de composants. Les diodes Zener D1, D2, D3 avec les résistances R1, R2, R3 réduisent la tension des broches outputs DTR, RTS, TXD sur le port série à environ 5V, ce qui est approprié pour le microcontrôleur (MOSI, SCK, RESET). Le signal MISO est connecté directement à la broche d'entrée CTS.

Chapitre IV :

Les afficheurs LCD

1- Les afficheurs LCD

Lorsqu'on travaille avec un microcontrôleur, on est vite limité en broches. Avec les écrans LCD, nous allons pouvoir afficher du texte ou des graphiques sur un écran qui n'est pas très coûteux et ainsi avoir un terminal simple.

Le terme LCD signifie "Liquid Crystal Display" et se traduit, en français, par "Écran à Cristaux Liquides" sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement. Ils consomment relativement peu (de 1 à 5 mA), sont relativement bons marché et s'utilisent avec beaucoup de facilité [24].

2 -Principe des cristaux liquides

L'afficheur est constitué de deux lames de verre, distantes de 20 µm environ, sur lesquelles sont dessinées les mantisses formant les caractères. L'espace entre elles est rempli de cristal liquide normalement réfléchissant (pour les modèles réflectifs). L'application entre les deux faces d'une tension alternative basse fréquence de quelques volts (3 à 5 V) le rend absorbant. Les caractères apparaissent sombres sur fond clair. N'émettant pas de lumière, un afficheur à cristaux liquides réflectif ne peut être utilisé qu'avec un bon éclairage ambiant. Sa lisibilité augmente avec l'éclairage. Les modèles transmissifs fonctionnent différemment: normalement opaque au repos, le cristal liquide devient transparent lorsqu'il est excité; pour rendre un tel afficheur lisible, il est nécessaire de l'éclairer par l'arrière, comme c'est le cas pour les modèles rétro éclairés [25].

3- Les différents afficheurs LCD

Dans la grande famille afficheur LCD, on distingue plusieurs catégories :

Les afficheurs alphanumériques

Les afficheurs graphiques monochromes

Les afficheurs graphiques couleur

Figure 37 : Afficheur alphanumérique [24].

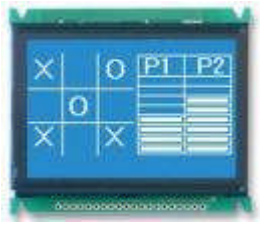


Figure 38 : Afficheur graphique (2 couleurs)[24].

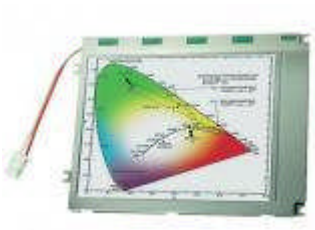


Figure 39 : Afficheur graphique en couleurs[24].

Les premiers sont les plus courants. Ils permettent d'afficher des lettres, des chiffres et quelques caractères spéciaux. Les seconds sont déjà plus avancés. On a accès à chacun des pixels et on peut donc produire des dessins beaucoup plus évolués. Ils sont cependant légèrement plus onéreux que les premiers. Les derniers sont l'évolution des précédents, la couleur en plus (soit 3 fois plus de pixels à gérer : un sous-pixel pour le rouge, un autre pour le bleu et un dernier pour le vert, le tout forme la couleur d'un seul pixel).

4- Choix d'afficheur LCD

Chaque afficheur LCD a sa propre caractéristique, pour l'afficheur alphanumérique sa taille de police est très limitée et nous ne pouvons pas tirer des graphiques. Donc en peut travailler avec des afficheurs graphiques (couleur) qui sont plus évolués par rapport à les afficheurs alphanumériques, mais sont coûteux.

Nokia 1100 LCD monochrome coûte moins cher, et nous pouvons afficher un grand texte de la police et des graphiques.

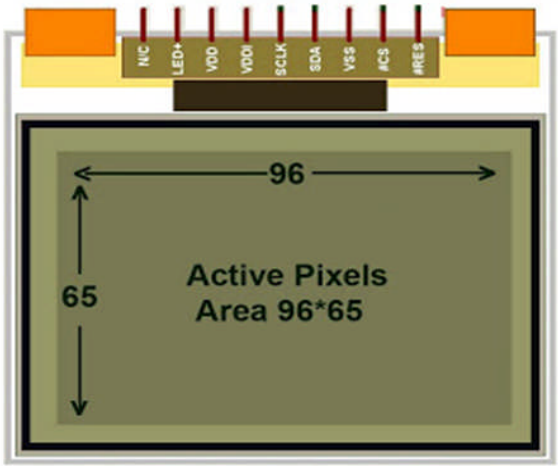


Figure 40 : afficheur LCD [25]

5- Description générale

Le LCD Nokia 1100 utilise la puce PCF8814 de Philips pour piloter un affichage graphique de 65 lignes et 96 colonnes. Il s'agit d'une puce sur verre (COG) avec 9 broches connectées à l'arrière de l'écran LCD.

6- Caractéristiques et architecture interne

- Affichage des données RAM 65x96 bits.
- Sélectionnables 3 lignes ou 4 lignes interfaces série, 6,5 MHz et haute vitesse I2C-bus.
- Oscillateur ne nécessitant aucun composant externe.
-Tension d'alimentation logique: 1. 7 à 3. 3 V.
- Tension haute tension d'alimentation du générateur: 2. 4 à 4. 5 V.
-Faible consommation d'énergie, adapté aux piles.

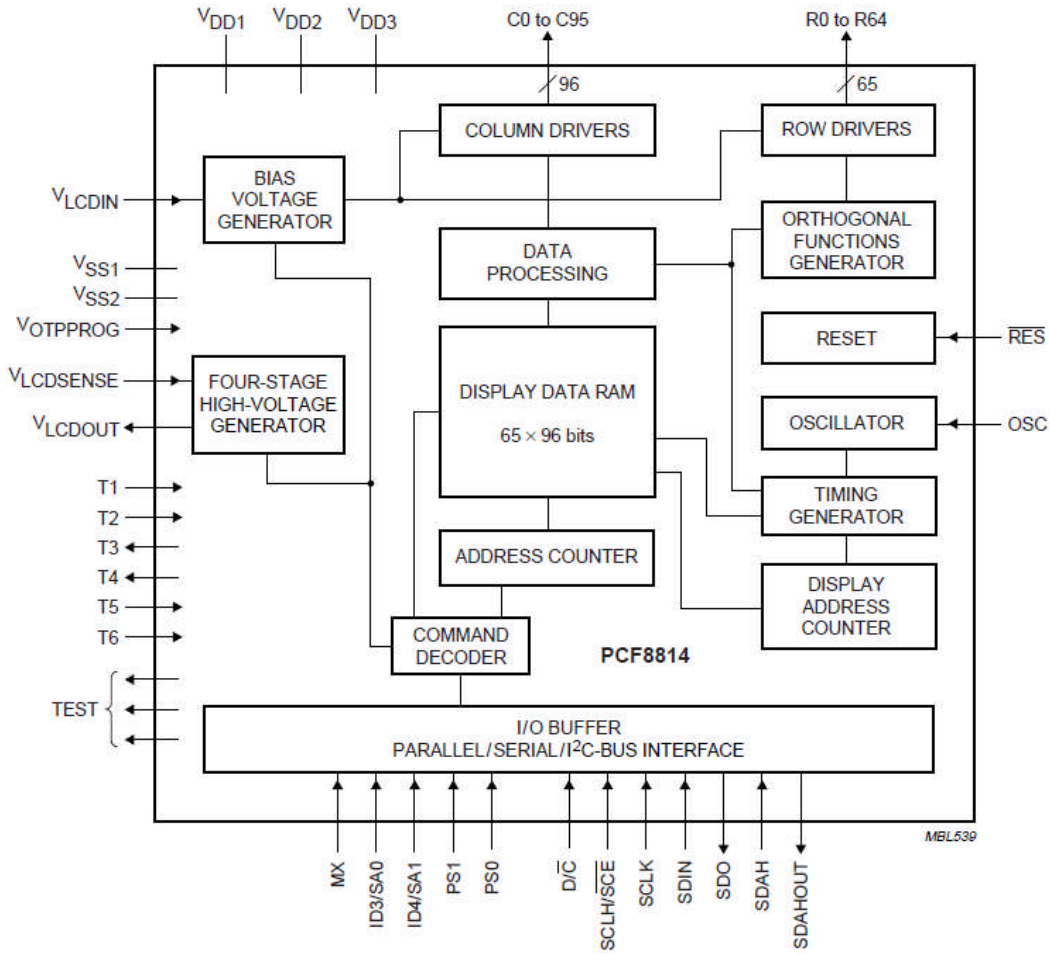


Figure 41 : architecture interne générale des circuits PCF8814 [26].

7- Description fonctionnelle

a- Oscillateur

L’oscillateur permet de fournir un signal d’horloge pour le système d’affichage ; qui ne nécessite pas l’introduction d’un composant externe. Lorsque l'oscillateur de la puce est utilisé, l’entré de OSC doit être relie a VDD1.

b- Compteur d'adresses

Le compteur d'adresse (AC) attribue des adresses à l’affichage de RAM de données pour l'écriture. X-adresse : X [06:00] et Y-adresse :Y [3:00] qui sont réglés séparément.

c- Affichage des données RAM

Le PCF8814 contient une RAM statique 96 x 65 bits qui stocke les données d’affichage. L'affichage RAM de données (DDRAM) est divisé en 9 banques (blocs) de 96 octets, même si, un seul bit de la 9e banque est utilisé. Lors de l'accès de RAM, les données sont transféré à la mémoire vive par l'intermédiaire de l'interface série.

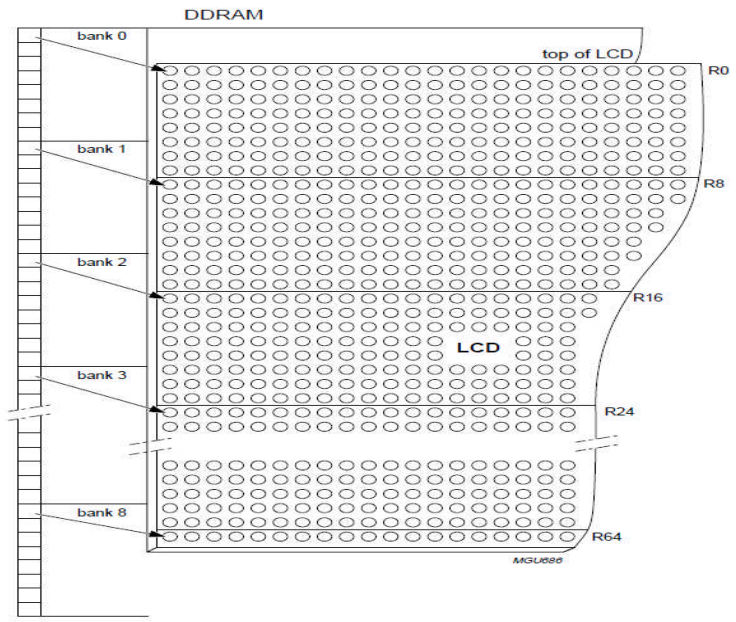


Figure 42 : DDRAM pour afficher la cartographie [26].

d- Adresse d'affichage compteur

L'affichage est généré par la lecture simultanément. L'état d'affichage (tous les points on / off et normal / inverse vidéo) est réglé par les bits DON, DAL et E dans la commande "Display Control".

8- Driver de ligne et de colonne du LCD

Le PCF8814 contient 65 lignes et 96 colonnes pilotes qui relient les tensions de polarisation à cristaux liquides. Le nombre de lignes sélectionnées simultanément est représenté par la valeur «p». Dans le PCF8814, «p» est réglé sur 4 ou 2 ou automatiquement, en fonction de l'affichage

partiel et le mode sélectionné. Les colonnes sont adressées par le pointeur d'adresse. X = 0 à 95 et Y = 0 à 8, Y représente numéro de banque.

9- Mode d'adressage

Il existe deux types de mode d'adressage : face verticale et face horizontal .

a- Face verticale

En mode d'adressage vertical chaque instruction d'écriture incrémente l'adresse dans la direction Y et puis retourner à l'adresse de départ

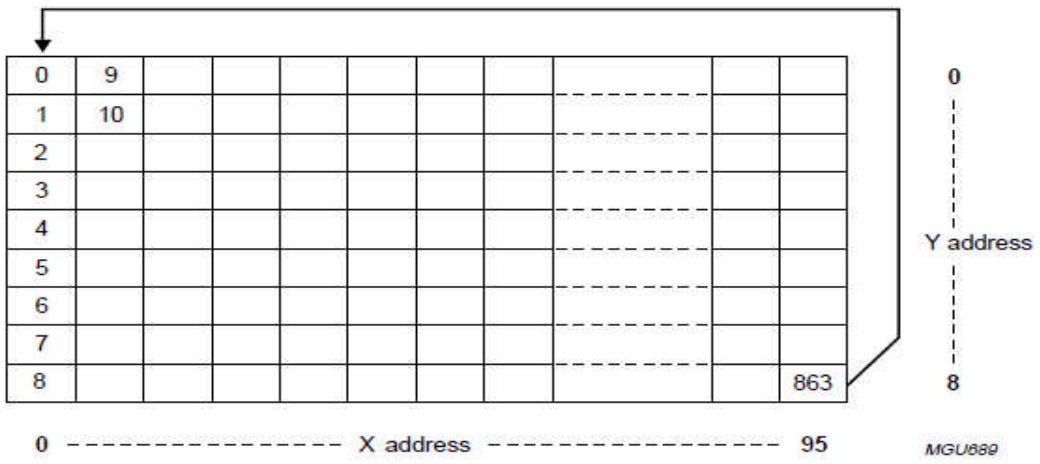


Figure 43 : adressage vertical.

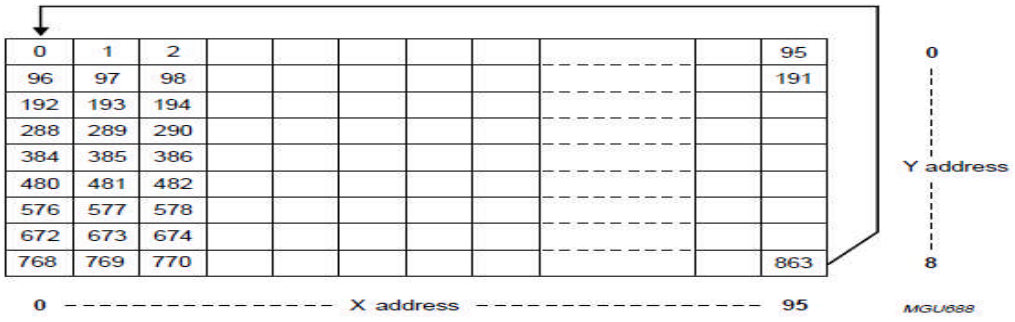


Figure 44 : adressage horizontal [26].

b- Face horizontal

Dans le mode d'adressage horizontal, l'adresse X incréments après chaque octet. Après la dernière adresse de X, X retourne a 0 et incréments Y pour traiter la ligne suivante

10- Interfaces série

La communication avec le microcontrôleur est réalisée par l'intermédiaire d'un

Synchronisé par horloge d'interface périphérique série.

En utilisant les entrées PS1 et PS0, pour choisir l'une

de l'interface de tableau 2.

PS1	PS0	INTERFACE
0	0	3-line SPI
0	1	4-line SPI
1	0	I2C-bus
1	1	3-line serial interface

Tableau [3] : interface de sélection [2]

Interface périphérique série

Interface périphérique série (SPI) est un 3-ligne ou 4 lignes Interface pour la communication entre le microcontrôleur et le circuit d'affichage à cristaux liquides.

Les trois lignes sont: SCE (validation de puce), SCLK (série horloge) et SDIN (données série). Lorsque l'interface SPI 3-ligne est utilisée l'affichage des données / commande est contrôlée par logiciel.

Pour l'interface série 4 en ligne de la ligne D/C est ajouté. La PCF8814 est reliée à la série de données d'E / S de microcontrôleur par deux axes: SDIN (entrée de données) et SDO (sortie de données) reliés entre eux.

11- Mode d'écriture

L'affichage des données / indication de commande peut être commandé soit via le logiciel ou en utilisant le D / C sélectionner l'entrée. Lorsque D / C entrée est utilisée, les données d'affichage sont transmises, lorsque D / C est HIGH (haut), et les données de commande est transmise D / C est LOW (bas) (figures 45 et 46).

Nombre d'octets de données d'affichage (1 à 255) sont d'être transmis (Fig. 47). L'octet suivant après l'affichage chaîne de données est traité comme une commande d'instruction. Si SCE est tirée vers le haut lors d'un flux de données d'affichage de série, l'octet est interrompue données non valides mais tout auparavant les données transmises sont valides. L'octet suivant reçu sera traité comme une commande d'instruction. (Fig.48).

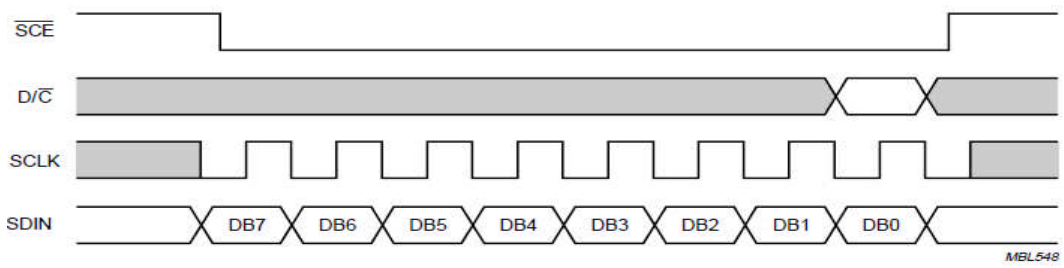


Figure 45: Protocole de transmission d'un octet.

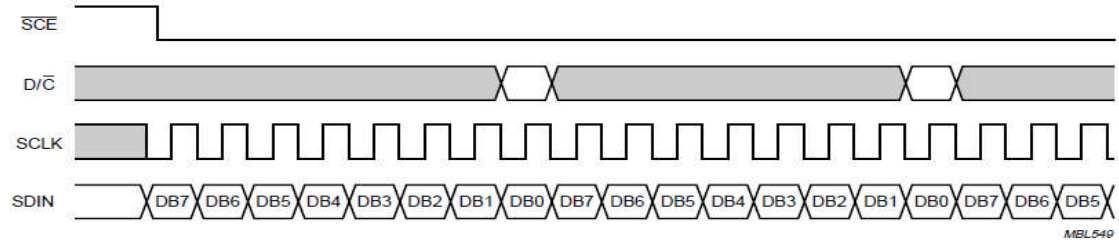


Figure 46: Protocole de transmission de plusieurs octets [26].

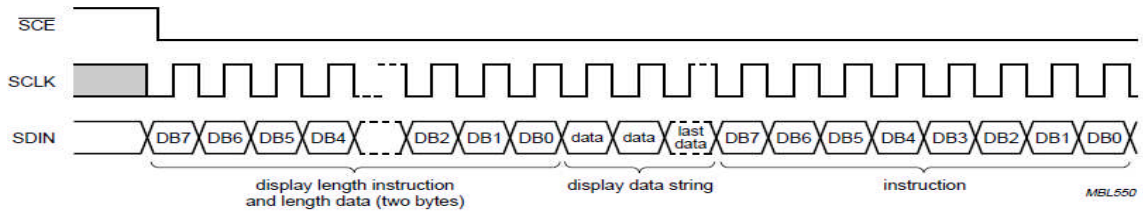


Figure 47 : transmissions de plusieurs octets.

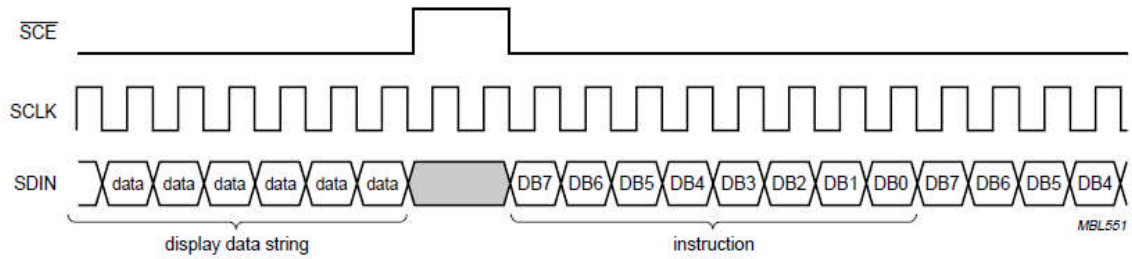


Figure 48 : transmission interrompue par SCE [26].

12- Mode de lecture (READ MODE)

Dans le mode de l'interface de lecture du microcontrôleur lit les données du PCF8814. Pour ce faire le microcontrôleur doit envoyer la commande d'état de lecture, le PCF8814 répondra par la transmission de données sur la ligne de SDO. SCE => haut (Fig. 49).

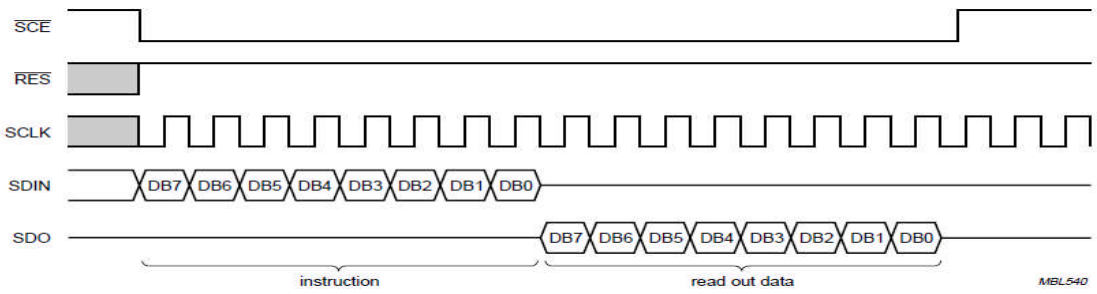


Figure 49: mode de lecture, SPI 3-line and 4-line [26]

13- Interface I2C-BUS

Le bus I2C HS est une interface qui utilise 2 lignes de communication avec des vitesses allant jusqu'à 3,4 MHz.

Les deux lignes doivent être connectées à un positif d'alimentation par l'intermédiaire d'une résistance de pull-up. Le transfert de données peut être déclenché que lorsque le bus n'est pas occupé.

14- Configuration du système

La configuration du système représenté sur la figure 50 comprend:

- L'émetteur: le dispositif qui émet les données vers le bus

- Récepteur: le dispositif qui reçoit les données à partir de bus

- Master: le dispositif qui déclenche un transfert, génère signaux d'horloge et termine le transfert.

- Esclave : le dispositif adressé par le maître

- Multi-maître: plus d'un maître peut tenter de contrôle du bus en même temps sans altérer le message

- Arbitrage: procédure pour s'assurer le contrôle de bus.

- Synchronisation: procédure pour synchroniser l'horloge des signaux de deux ou plusieurs dispositifs [26]

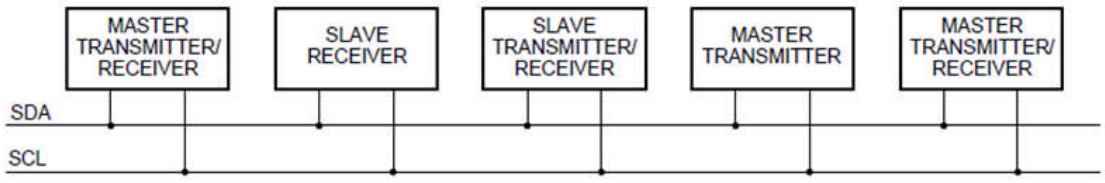


Figure 50 : configuration de system I2C *[26]*

Chapitre V :

Manipulation, tests et résultats

1- Introduction :

Ce chapitre est consacré aux manipulations, résultats et tests que nous avons effectués au laboratoire pour ce projet. Les manipulations et tests ont été performés d'une manière progressives; c.-à-d, on commence par le plus aisé pour en augmenter la difficulté. Cette technique était essentielle du moment que ce type de microcontrôleur ainsi que sa plateforme de développement étaient de nouveaux concepts. La figure 51 montre le diagramme de notre manipulation.

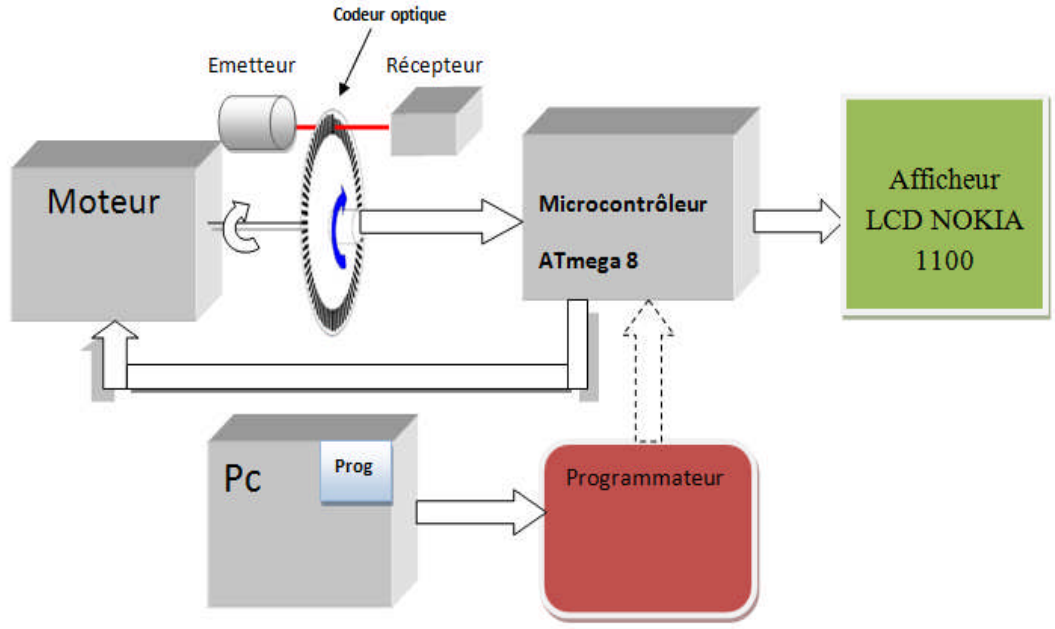


Figure 51 : Schéma global qui représente les blocs essentiels de la plaque d'essai.

Pour la suite de ce texte, on va présenter plusieurs tests en montrant le schéma descriptif, le programme utilisé et une explication concise. Au final, on donnera la mesure des valeurs et les courbes trouvées.

2- Réalisation des schémas et écriture de programme

Pour réalisation des schémas, on a utilisé le logiciel isis proteus. Il est efficace et facile à comprendre. Il permet de saisir le schéma électronique et de le simuler. ARES, contient un module qui permet de faire le routage et la conception du circuit imprimé basé sur la conception dans la suite logicielle Proteus.

Test 1 : Ce programme fait clignoter une LED connectée au port PD7 d'un microcontrôleur Atmega8 à des intervalles de temps donnés .

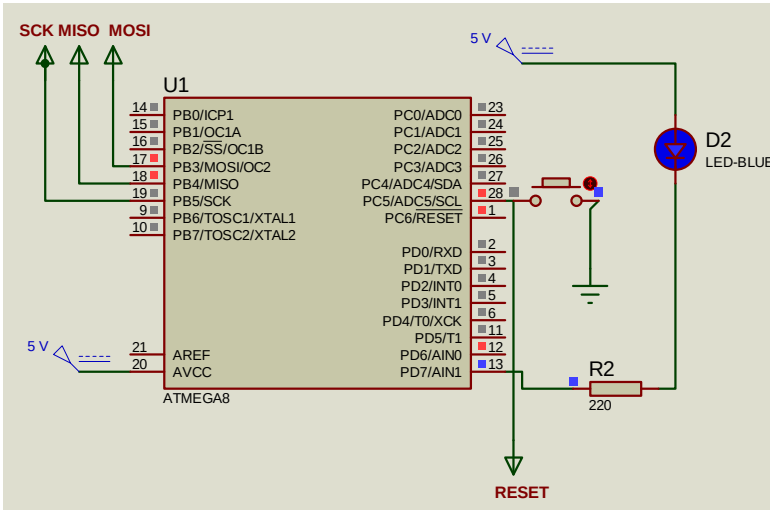


Figure 52 : fenêtre de simulation d'une LED par proteus.

```
/*****************************************************************************
******************************************************************************/

/* déclarations des variables  globale */
#define sbi(x,y) (x |= (1<<y));          /* set bit y in byte x */
#define cbi(x,y) (x &= (~(1<<y)));       /* clear bit y in byte x */
#include <avr/io.h>
#include <util/delay.h>

int test=1;

int main (void)
{
if (test == 1)                  //   test one pin
{
    sbi (DDRD, 7);              //configurer PORTD en sortie
    while(1)
```

```
        {                                   // Boucle sans fins
                sbi (PORTD, 7);             // active le port 7
              _delay_ms(1000);            // attendre 1 seconde
               cbi (PORTD, 7);             // désactivé  port 7
              _delay_ms (500);            // attendre 0.5 seconde
        }}
  return(0);
}
```

Test2 : Ce programme fait clignoter deux LED connectées au ports PD7 et PD6 d'un microcontrôleur Atmega8 à des intervalles de temps donnés .

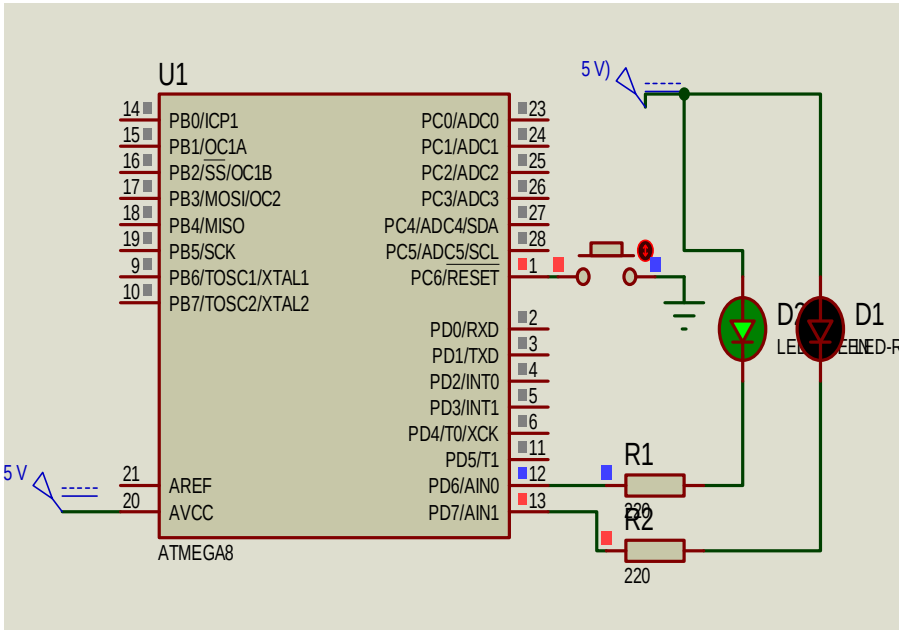


Figure 53 : fenêtre de simulation de deux LED par proteus.

```
/*********************************************************************

*********************************************************************/

if (test == 101)                    // test 2 pins
{
    sbi(DDRD,6);                    // configurer PORTD 6 en sortie
    sbi(DDRD,7);                   //    configurer PORTD 7 en sortie

    while(1)                        //  Boucle sans fins
             {
             sbi(PORTD,6);             //active port 6
             cbi(PORTD,7);            // désactivé  port 7
             _delay_ms(500);           // Attendre  0.5 seconde
             cbi(PORTD,6);             // désactivé  port 6
             sbi(PORTD,7);             //active port 7
            _delay_ms(500);            // Attendre  0.5 seconde
            }}
return(0);
}
```

```
   if (test == 102)                    // fonction test avec autre méthode
        sbi(DDRD,6);                   // configurer PORTD 6 en sortie
        sbi(DDRD,7);                   // configurer PORTD 7 en sortie
        while(1)                       // boucle sans fins
              {
                  pin_7_6();
 }            }                        // Fin de  test
return(0);
}
```

Test3 : Se basant sur le programme du test2, on ajoute au circuit deux relais permettant de faire tourner le moteur dans les deux sens de rotation. Les deux relais avec des transistors NPN sont connectés au port PD7 et PD6 du microcontrôleur Atmega 8. En une boucle infinie, le moteur tourne dans un sens, s'arrête un certain temps, puis on inverse son sens de rotation. Les paramètres sont indiqués dans le programme.

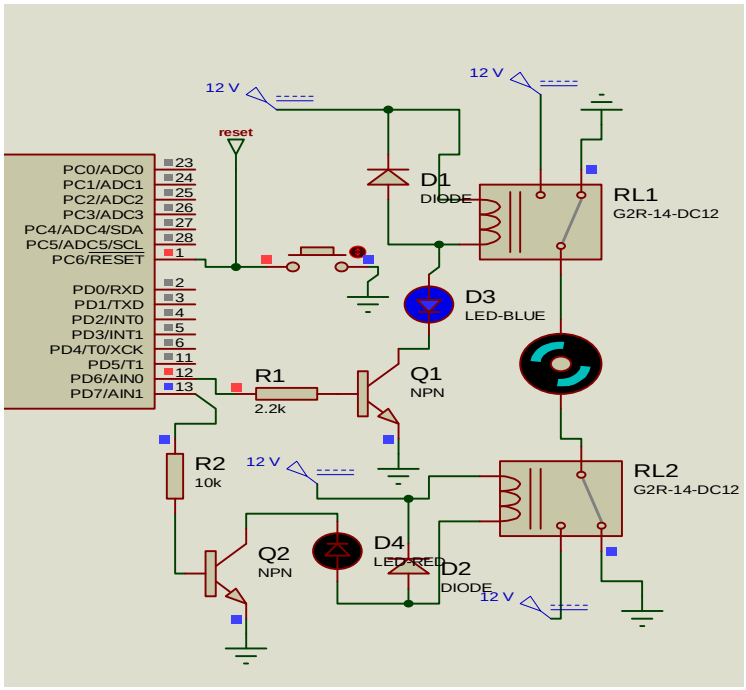


Figure 54 : simulation du circuit du moteur par proteus.

```
/*************************************************************************

*************************************************************************
/
#define sbi(x,y) (x |= (1<<y));            /* set bit y in byte x */
#define cbi(x,y) (x &= (~(1<<y)));        /* clear bit y in byte x */
#include <avr/io.h>
#include <util/delay.h>

int test=106 ;

    int main (void)
     {
     if (test == 106)                  // test sens  de rotation de moteur
     {
     sbi(DDRD,7);                      // configurer PORTD 7 en sortie
     sbi(DDRD,6);                      // configurer PORTD 6 en sortie
         _delay_ms(5000);              // Attendre 5 seconde
         _delay_ms(5000);              // Attendre 5 seconde
         _delay_ms(5000);              // Attendre 5 seconde
     while(1)
```

```
        {
        sbi(PORTD,6);              // active port 6
        _delay_ms(5000);           // attendre 5 seconde
        cbi(PORTD,6);              // éteindre  port 6
        _delay_ms(250);            // attendre 0.25 seconde
        sbi(PORTD,7);              // active port 6
        _delay_ms(5000);           // attendre 5 seconde
        cbi(PORTD,7);              // éteindre  port 7
        _delay_ms(250);            // attendre 0.25 seconde
    }}

return(0);
    }
```

Test 4 : Ce programme permet d'afficher des valeurs ou des caractères sur l'écran LCD Nokia 1100. Il permet d'introduire et d'utiliser la bibliothèque pour ce type de LCD.

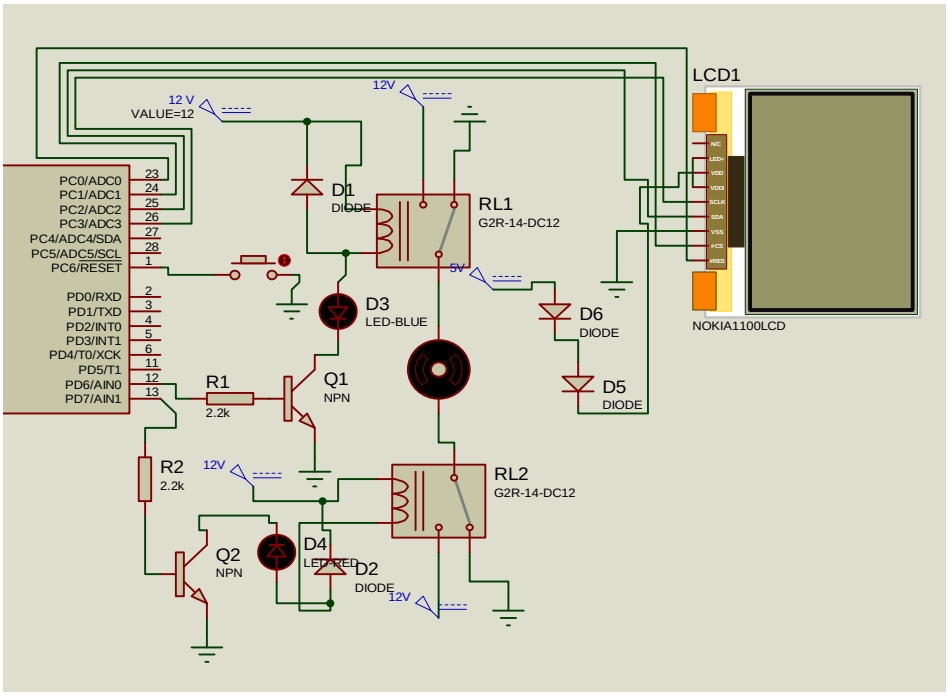


Figure 55 : simulation de l'afficheur LCD NOKIA 1100 avec proteus

```
/*****************************************************************************

*****************************************************************************/
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include "nokia1110_lcd_lib.h"
char buffer[36];
int test=112;
int main (void)
{
if (test == 110)              // test 1 lcd
      {
      nlcd_Init();           // appel  de la fonction nlcd_Init()
            while(1){
            }}               // end of test
      if (test == 112)       // test 2 lcd
      {
      nlcd_Init();
           while(1)         // Boucle sans fins
          {
```

```
        nlcd_GotoXY(20,2)       // définir la position x ligne et y colonne
        nlcd_Putc('V',1);       // afficher  un caractère
                 }}             // end of test
             if (test == 113)   // test  LCD
             {
             nlcd_Init();
               while(1)
               {
               sprintf(buffer,"HELLO"); //afficher  un texte
               nlcd_GotoXY(0,2);     //définir la position x ligne et y colonne
               lcd_str(&buffer[0],1);
               }}
      return(0);
      }
/*************************************************************************
```

Pour la configuration de la bibliothèque, on doit spécifier les ports qui relient le microcontrôleur et l'interface LCD.

```
*************************************************************************/
      // Port connecté au contrôleur LCD NOKIA 1100
      #define PORT_LCD PORTC
      #define PIN_LCD  PINC
      #define DDR_LCD  DDRC
      //les broches du port qui sont relies au contrôleur LCD
      #define SCLK_LCD_PIN    3
      #define SDA_LCD_PIN     2
      #define CS_LCD_PIN      1
      #define RST_LCD_PIN     0
```

Test 5 : La bibliothèque "nokia1100_lcd_font.h" contient des polices qui nous permettent d'afficher des caractères ASCII tel que caractère spéciaux, des chiffres et l'alphabet latin. L'écran Nokia 1100 a une matrice de 96x65 pixels. Dans ce test, on affiche le caractère Ж, par exemple, en changeant les données du caractère dans la police.

```
      /*************************************************************
      ***********
      0b01010100,
      0b01010100,
      0b01111100,
      0b00010000,
      0b00010000,
      0b01111100,
      0b01010100,
      0b01010100,
      *************************************************************
      **********/
      #include <avr/io.h>
      #include <string.h>
      #include <stdio.h>
      #include <stdint.h>
      #include "nokia1110_lcd_lib.h"
```

```
char buffer[36];

int test=112;

            int main (void)
{

if (test == 140)       // test
     {
  nlcd_Init();          // appel  de la fonction nlcd_Init()
_delay_ms(5000);    // Attendre 5 seconde
 while(1){
     buffer[0]=0;
     sprintf(buffer,"!#");     // affichage caractère amazigh
     nlcd_GotoXY(0,1);         //définir la position x ligne et y
colonne
     lcd_str(&buffer[0],0);
     delay_ms(5000);           // Attendre 5 seconde
     nlcd_Clear();             // affacer l'ecran
       }}                    // end of test
 Return (0);
}
```

Test6 : Dans ce programme on introduit le timer /compteur pour l'afficher du nombre interruptions et le calcul de la vitesse du moteur.

```
/*****************************************************************

*****************************************************************/
//déclarer les compteurs
void timer1_init(void);
volatile long unsigned int counter1=0,counter2=0;
volatile long unsigned int counter3=1,counter4=0;
volatile unsigned int ico=1,ico1=0, read1=0;
volatile unsigned int dpulse;

if (test == 114)
     {
sbi(DDRD,7);           // configurer PORTD 7 en sortie
cbi(DDRB,0);           // configurer PORTD 0 en entre icp

 timer1_init()        // appel  de la fonction timer1_init()
 nlcd_Init();         // appel  de la fonction nlcd_Init()
     while(1){
       if(read1 == 1) {
       read1 = 0;
          if((counter3%3) == 0);
{
     buffer[0]=0;
     sprintf(buffer,"T%06d",(64*counter3)/100);
     sprintf(buffer,"T%06d",counter3);
     nlcd_GotoXY(0,0);      // la position x ligne et y colonne
          lcd_str(&buffer[0],1);
```

```
            sprintf(buffer,"%07d ",counter1/10);
            nlcd_GotoXY(0,2);  // la position x ligne et y colonne
            lcd_str(&buffer[0],1);
            sprintf(buffer,"%07d ",counter2/counter4);
            nlcd_GotoXY(0,4);  // la position x ligne et y colonne
            lcd_str(&buffer[0],1);
                 counter2 = 0;
                 counter4 = 1;
}}}}                              //end of test

return(0);
}
// configuration des timers
void timer1_init()
{
 // configure le timer avec prescaler = 1024
TCCR1B |= (1 << WGM12)| (1<<ICNC1);
TCCR1B |= (1 << CS10) | (1 << CS11);
// initialisation de compteur
TCNT1 = 0;
// permettre à débordement interruption
//TIMSK |= (1 << TOIE1);
// permettre à débordement interruption et icp
TIMSK |= (1 << TOIE1)| (1<<TICIE1);
// permettre les interruption globale
    sei();
}

ISR(TIMER1_OVF_vect)
{
     counter3 +=1;        // Incrémenté le compteur 3
     counter4 +=1;        // Incrémenté le compteur 4
     read1 = 1;
return;
}

ISR(TIMER1_CAPT_vect)
{
     counter1 +=1;          // Incrémenté le compteur 1
     counter2 +=1;          // Incrémenté le compteur 2
return;
}
```

3- Calcul de la fréquence du microcontrôleur ATMega 8

A l'aide d'un chronomètre, on compte le nombre de clignotements d'une LED attachée au microcontrôleur.

Pour 100 clignotements (C) en obtient 50,87 seconds. Ce qui donne un clignotement en 0,51 second. Avec un prescaler de 64 et un compteur de 16 bits allant de 0 à $2^{16}-1$, le clignotement, se fait en 64×2^{16} de cycle du microcontrôleur.

La fréquence est donc $f = 64 \times 2^{16} / 0{,}5 = 8{,}2\,MHz$.

4- Réalisation de la plaque d'essai

La figure 56 ci-dessous représente une photo réelle de la carte de commande réalisée sur une plaque d'essai, après avoir testé et vérifier le programme au laboratoire.

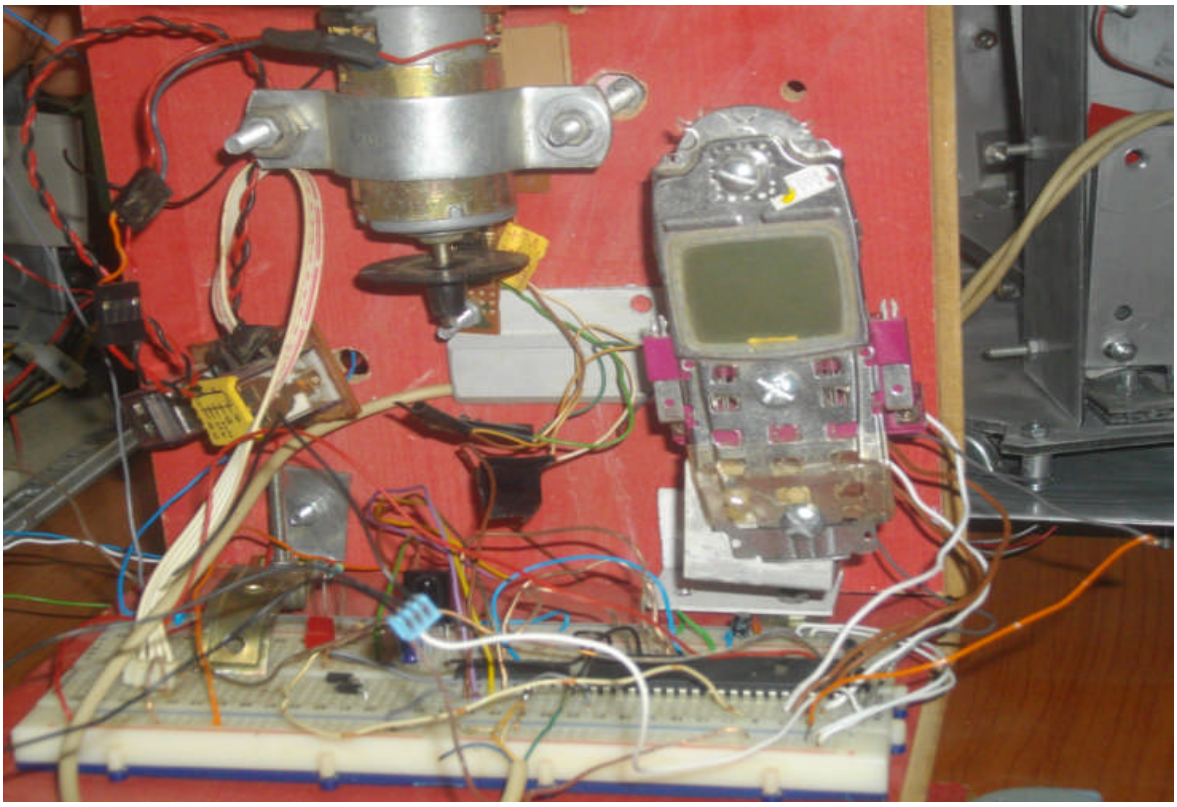


Figure 56 : image de la plaque d'essai.

5- Détermination de la vitesse de rotation (en tr/s)

Les pistes extérieures du codeur sont divisées en N intervalles égaux. Chaque intervalle correspond à une fente transparente.

En un tour complet, le faisceau IR est interrompu N fois. Le signal électrique est délivré au microcontrôleur via un phototransistor placé avec le codeur. Pour ce codeur, on compte 72 fentes. Sous une tension de $13{,}8V$, le programme affiche 1306 interruptions par seconde. La vitesse de rotation du moteur est donc $\omega = 1306/72 = 18\,tours/s$.

6- Variation de vitesse du moteur

Comme on a indiqué dans le chapitre II, le tableau suivant représente les mesures de la variation de la vitesse de rotation du moteur en fonction de la tension à ses bornes.

N	Tension (V)	Résistance (Ω)	Vitesse (tr/min)
1	12 ,36	2,7	930
2	9 ,33	38 ,4	687
3	7,50	46,5	524
4	6 ,80	60	472
5	5,78	71,1	393 ,5
6	4,70	83	300
7	3,98	90	236

Tableau [4]

On utilise une résistance variable en série avec le moteur. A chaque valeur de la résistance, on mesure la tension et on reprend la valeur de la vitesse affichée sur l'afficheur LCD. On a une tension limite de 3,98 V du déclenchement de la rotation.

Ses valeurs nous permettent de tracer la courbe correspondante. La courbe obtenue est une droite. Résultat attendu car théoriquement, l'énergie électrique consommé par le moteur $W_{elec} = V^2 t / R$ est transformé en énergie cinétique de rotation $E_c = I\omega^2 / 2$. Avec un rendement de $\mu = E_c / W_{elec}$, on obtient $I\omega^2 / 2 = \mu W_{elec} = \mu V^2 t / R$. On conclut que ω est proportionnel à V.

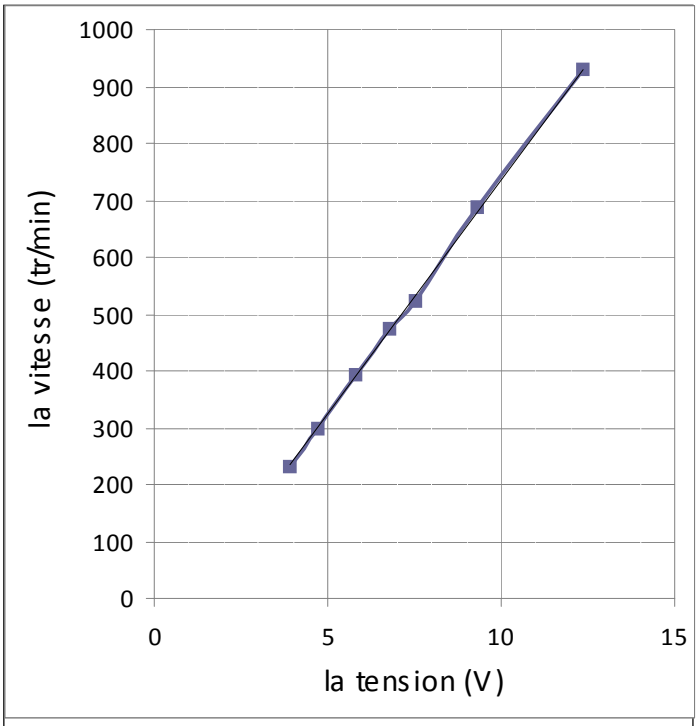


Figure 57 : variation de la vitesse en fonction de la tension du moteur étudié.

7- Visualisation des signaux de commande :

Pour visualiser les signaux générés par des broches de Port C du microcontrôleur où sont attachés les pins de l'afficheur LCD, on a utilisé le logiciel scope. Celui-ci utilise la carte son pour acquérir et afficher le signal.

Les quatre signaux visualisés (voir figures ci-bas) sont :

SCLK - ligne horloge

SDA - ligne des données et des commandes

CS - validation de puce

RST - réinitialisation de ligne

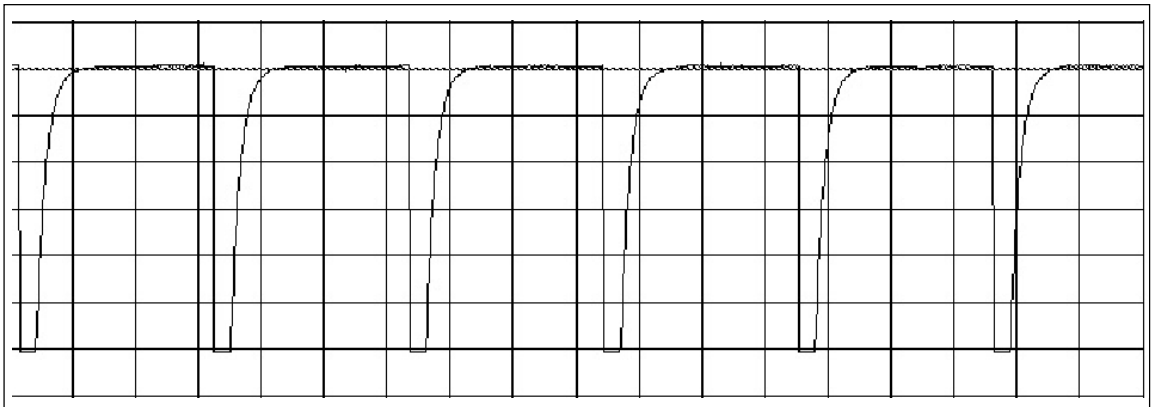


Figure 58 : La courbe obtenue sur l'oscilloscope du signal SCLK.

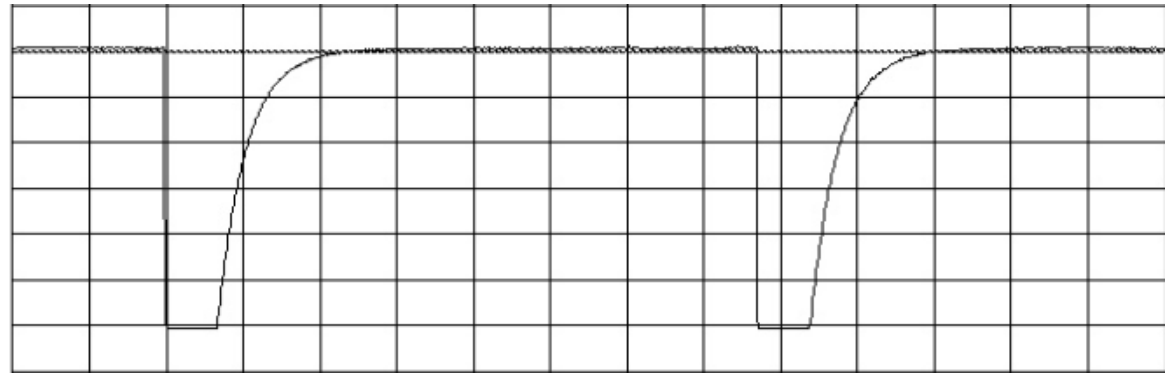


Figure 59 - La courbe obtenue sur l'oscilloscope du signal SDA.

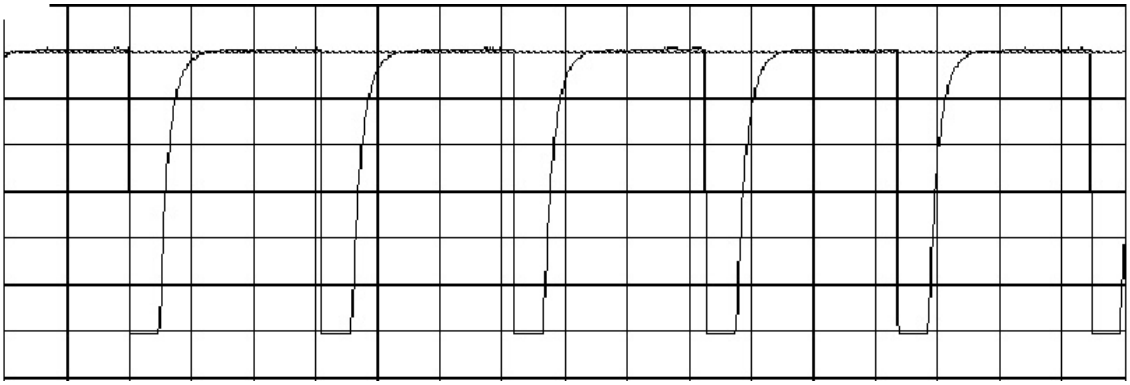


Figure 60 - La courbe obtenue sur l’oscilloscope du signal CS.

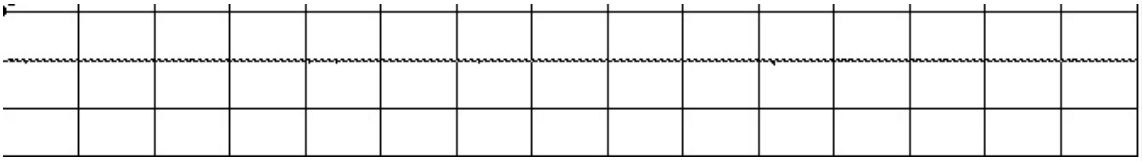


Figure 61 - La courbe obtenue sur l’oscilloscope du signal RST.

Ces signaux sont bien ceux nécessaires pour faire fonctionner le LCD.

Conclusion générale

Ce projet pratique nous a permis d'acquérir des connaissances importantes dans le monde des systèmes embarqués et qui touche à plusieurs thématiques connexes : électroniques, automatique, électrotechnique, programmation, etc.

Après l'introduction, nous avons exposé, au premier chapitre, la description du microcontrôleur de type Atmel et ses caractéristiques de fonctionnement. Ses périphériques tel que les mémoires, les timers /compteurs, nous ont permis la mise en œuvre d'une interface fonctionnel.

Au second chapitre, on a présenté quelques généralités sur les moteurs à courant continu. On a cité leurs avantages et inconvénients. Ce qui nous guide, entre autre, à un faire le choix de moteur pour la réalisation de ce projet.

Suivi d'un chapitre consacré à l'environnement de programmation, la réalisation et simulation des schémas. La programmation consiste en trois étapes essentielles : écrire le programme, le compiler et le charger dans la mémoire de microcontrôleur. Pour écriture de programme, on a utilisé le langage C qui est bien adapté à ce microcontrôleur. Pour transférer le fichier binaire, on a utilisé le programmateur que nous avons assemblé.

Le programmateur consiste à la fois d'un logiciel et d'un hardware. Le hardware se relie au PC avec le microcontrôleur. Le logiciel que nous avons utilisé est appelé avrdude qui est disponible pour Linux, Mac et Windows. Pour vérifier et simuler le fonctionnement du programme, on a utilisé logiciel isis proteus.

L'objectif visé est atteint en finalisant ce projet. On a pu crée des montages électroniques basics. Quelques tests ont été mentionnés dans ce mémoire. Une fois familier avec cette plateforme, on est passé à un projet conséquent qui combine plusieurs disciplines : électronique, mécanique, physique et électromécanique. Plusieurs techniques d'ingénierie ont été utilisées, y compris la collecte et l'analyse de données. Une table de valeurs mesurées avec son graphe à été présenté au chapitre V compatible avec ce que l'on prédit théoriquement.

Faisant parti de notre pédagogie et formation, on a aussi pu aussi s'apercevoir des déficiences techniques, linguistiques et pratiques que nous avons rencontrées. On se forcera à les rectifier et à s'améliorer.

Ces acquis sont certainement un atout pour entamer une vie active d'ingénierie. Additionnement on a appris à travailler en groupe dans un projet pluridisciplinaire. La coopération interdépartementale et les discussions ont facilité notre tache. La réutilisation de composants de recyclage et l'utilisation de plateforme performante mais gratuite doivent nous inciter à concevoir des projets avec des moyens derisoires. Aussi, on s'est attelé à faire une recherche biobibliographique pour préparer notre mémoire. Cette expérience me donne le courage, l'envie et l'espoir de développer et d'améliorer mes capacités pour confronter le mode professionnel.

Enfin et comme perspective touchant à notre domaine de spécialité, nous pouvons entrevoir plusieurs adaptations pratiques à ce système dans la spécialité 'électronique biomédicale'. Moyennant une amélioration et un rajout de composants, par exemple, un agitateur ou mélangeur rotatif, précis et automatique peut facilement être mis en œuvre.

Bibliographie

[1] Christian Tavernier, Les Microcontrôleurs 4et 8 bits, Initiation pratique. DUNOD, Paris, 1995

[2] David Rey, contrôle, commande et mesure via Internet, montage électronique et protocole http. Dunod, Paris, 2008

[3] Florian Schäffer, Au cœur de l'AVR d'ATMEL - Programmation en C des microcontrôleurs RISC AVR®,-© 2009 Elektor-Publitronic

[4] Microcontrôleurs AVR : des ATtiny aux ATmega - Description et mise en œuvre. Dunod, Paris

[5] mémoire : étude et réalisation d'un kit électronique du type système embarque a base de microcontrôleur, iddir hayet

[6] http://www/people.coins-lab.org/.../chapitre%203_le%20microprocesseur

[7] http:// iutva.free.fr/IMG/pdf/chap123.pdf

[8] http://www.atmel.com/.../atmel-2486-8-bit-avr-microcontrôleur

[9] http://www.protostack.com/microcontrollers/atmega8a-pu-atmel-8-bit-8k-avr-microcontroller

[10] http://www.reality.be/elo/labos2/files/AtMega32DocFr.pdf

[11] http://mlkssi.free.fr/Cours_fichiers/mcccor.pdf

[12] http://ww2.ac-poitiers.fr/electrotechnique/IMG/pdf/moteur_a_cc.pd

[13] http://sti.discip.ac-caen.fr/sites/sti.discip.../le_moteur_a_courant_continu.pdf

[14] http://osegouin.free.fr/cnc/paps.pdf

[15] http:// web.cluny.ensam.fr/sites/EEA/.../Moteurs-pas-a-pas_BM_2004.pdf

[16] http://www.moteurindustrie.com/brushless/technique.html

[17] *http://*www.lmdindustrie.com/.../KNF_techno36_1204.pdf

[18] http://5lair.free.fr/Teachings/Files/CM%20Mecanique.pdf

[19] http://www.courseleec.fr/IMG/pdf/capteurs_optiques.pdf

[20] http:// ww2.ac-poitiers.fr/électrotechnique/IMG/doc/Codeurs.doc

[21] http://ww2.ac-poitiers.fr/electrotechnique/img/pdf/**c**odeurs_optiques_prof.pdf

[22] http://sed-wiki.inrialpes.fr/pub/Platforms/Misc/.../AvrManual.pdf

[23] http://www.electro-tech-online.com/attachments/avr_dasa-png.41938/

[24] http://fr.openclassrooms.com/sciences/cours/arduino-pour-bien-commencer-en-electronique-et-en-programmation/les-ecrans-lcd-1

[25] http://www.aurel32.net/elec/lcd.php

[26] http://fr.scribd.com/doc/41122796/PCF8814

[27] http://www.distrimedia.fr/cablage.html

[28] http://www.zeitnitz.eu/scope_en

[29] http://html.alldatasheet.fr/html-pdf/26594/VISHAY/TSOP17XX/182/1/TSOP17XX.html

[30] http://tel.archives-ouvertes.fr/docs/00/04/67/90/PDF

[31] https://cours.etsmtl.ca/gpa668/aCours/GPA668_pas_a_pas_E2011.pd

[32] http://arlontronique.wordpress.com/le-programmateur/programmateur-port-serie/

[33] http://www.electrosome.com/ir-transmitter-receiver-led-tsop1738

[34] http://fr.wingwit.com/Hardware/computer-drives-storage/49120.html#.VAdYvfl_suk

[35] http://tel.archives-ouvertes.fr/docs/00/04/67/90/PDF

[36] http://www. linuxfocus. org/Francais/November2004/article352.shtml

[37] wiki.jelectronique.coma/doku.php?id=vr

http://www.youtube.com/watch?v=w5a3jP0-baI

Annexe :

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 8K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and MLF package
    Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    Eight Channels 10-bit Accuracy
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad MLF
- Operating Voltages
  - 2.7 - 5.5V (ATmega8L)
  - 4.5 - 5.5V (ATmega8)
- Speed Grades
  - 0 - 8 MHz (ATmega8L)
  - 0 - 16 MHz (ATmega8)
- Power Consumption at 4 Mhz, 3V, 25°C
  - Active: 3.6 mA
  - Idle Mode: 1.0 mA
  - Power-down Mode: 0.5 µA

ATMEL®

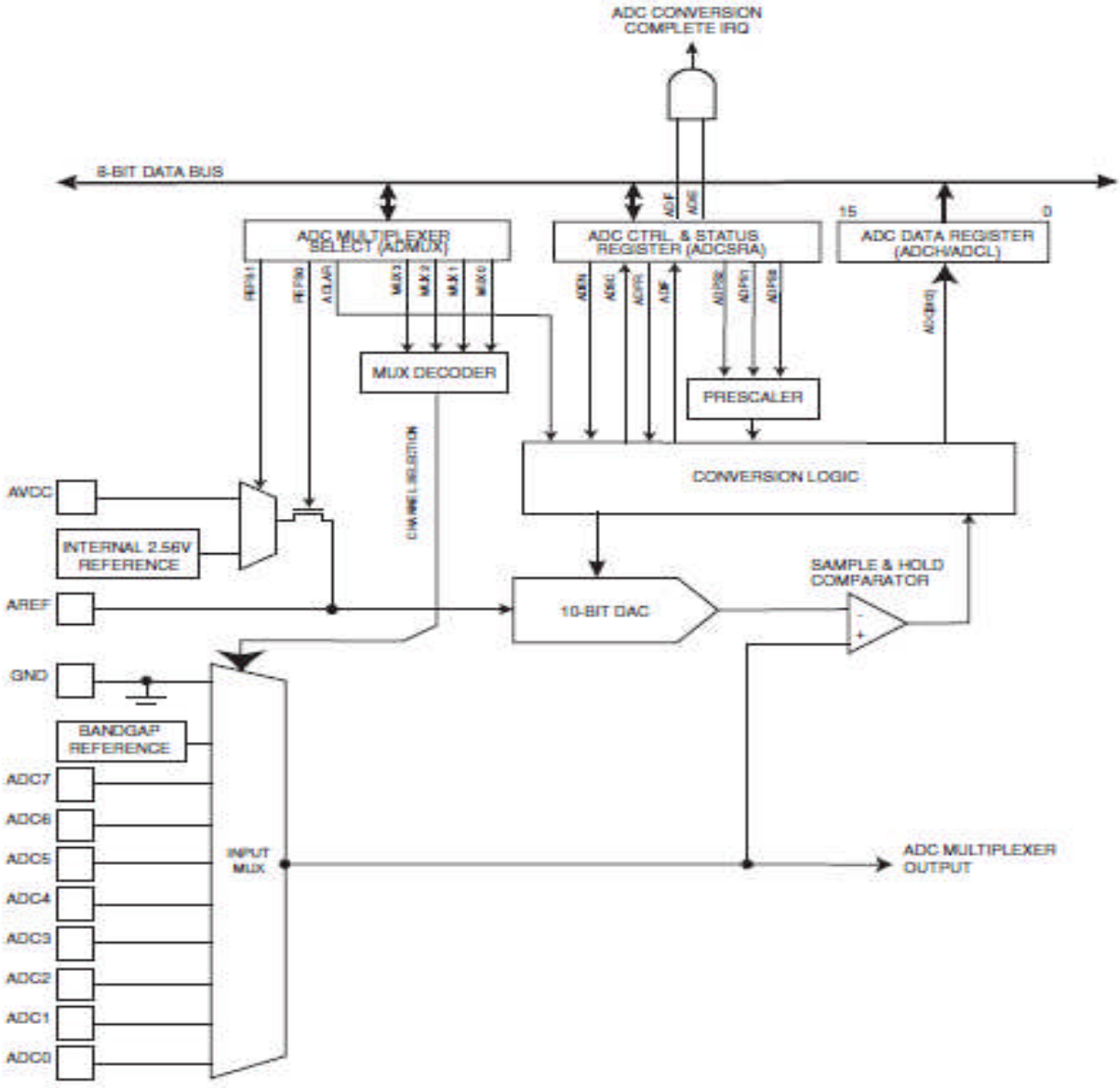
8-bit AVR® with 8K Bytes In-System Programmable Flash

ATmega8
ATmega8L

2486O–AVR–10/04

ATMEL®

Schémas Synoptique du convertisseur AD



Moteur courant continu
1.13.021.3xx

Les avantages :
Moteur d'entrainement industriel,
Idéal pour fonctionnement en start/stop
et inversion de sens de rotation

Les produits associés :

> Alimentation
DR-30-12 / 24
PS-24/2
S-60-24

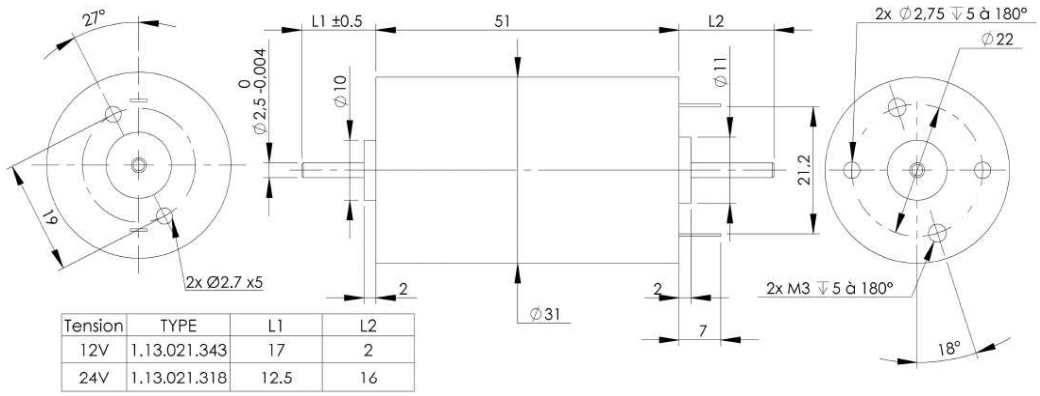
> Réducteur
GS38A
PM32
W02
W13

> Cartes électroniques
FIRST-DC-1Q 50/5
NANO DC 1Q 30/3

buehler **6 W - 12 W**

Tension d'alimentation (Ua)	V	12	24
Vitesse au courant In	tr/mn	2980	3969
Couple au courant In	mNm	20.00	21.00
Courant max permanent (In)	mA	900	540
Vitesse à vide à Ua à +/- 10%	tr/mn	4400	5250
Courant à vide à +/- 50%	mA	100	70
Couple de démarrage à Ua	mNm	61.00	85.00
Courant de démarrage à Ua	mA	2500	2000
Constante de couple	mNm/A	25.00	42.00
Constante de vitesse	tr/mn/V	382	227
Pente vitesse/couple	tr/mn/mN	71	61
Vitesse limite	tr/mn	4400	5250
Puissance utile max. à Ua	W	7.0	11.7
Rendement maximum	%	65	69
Constante de temps	ms	11	11
Inertie	gcm²	16	16
Résistance aux bornes	Ohm	4.80	12
Résistance thermique	K/W	11	11
Résistance thermique Rotor/Boîtier	K/W	10	10

Commutation	Graphite
Nombre de lames au collecteur	7
Paliers	Roulement à billes
Aimants	Ferrite
Charge axiale maximum (dynamique)	5 N
Jeu axial minimum	0.05 mm
Jeu axial maximum	0.70 mm
Charge radiale maximum	20 N
à une distance de la face de :	15 mm
Température ambiante mini de	-10 °C
Température ambiante maxi de	+70 °C
Température max. rotor	155 °C
Poids	135 g



Tension	TYPE	L1	L2
12V	1.13.021.343	17	2
24V	1.13.021.318	12.5	16

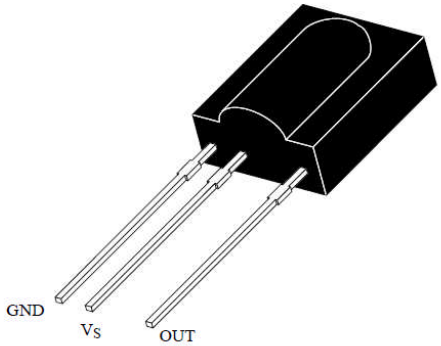
Photo Modules for PCM Remote Control Systems

Available types for different carrier frequencies

Type	fo	Type	fo
TSOP1730	30 kHz	TSOP1733	33 kHz
TSOP1736	36 kHz	TSOP1737	36.7 kHz
TSOP1738	38 kHz	TSOP1740	40 kHz
TSOP1756	56 kHz		

Description

The TSOP17.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.
The demodulated output signal can directly be decoded by a microprocessor. TSOP17.. is the standard IR remote control receiver series, supporting all major transmission codes.

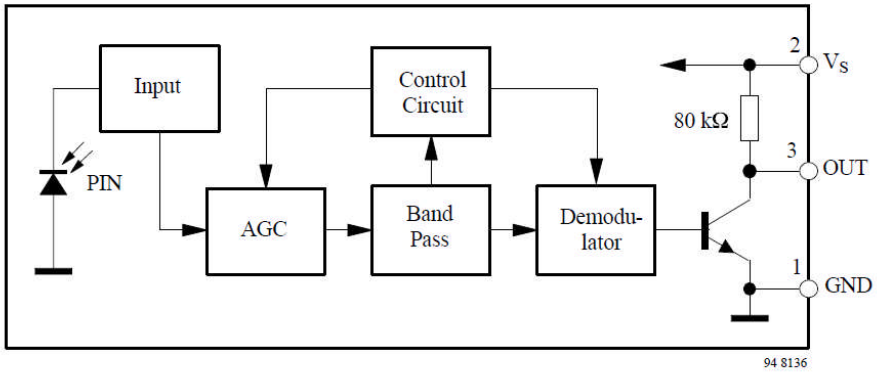


94 8691

Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption
- High immunity against ambient light
- Continuous data transmission possible (up to 2400 bps)
- Suitable burst length ≥10 cycles/burst

Block Diagram



94 8136

65 × 96 pixels matrix LCD driver PCF8814

1 FEATURES

- Single-chip LCD controller or driver
- 65 row, 96 column outputs
- Display data RAM 65 × 96 bits
- Selectable 3-line or 4-line serial interfaces, 6.5 MHz and High-speed I²C-bus
- On-chip:
  - Configurable voltage multiplier generating V_{LCD}; external V_{LCD} also possible
  - Four segment V_{LCD} temperature compensation
  - Generation of intermediate LCD bias voltage
  - Oscillator requires no external components; external clock input also possible
- Temperature read-back via interface
- External reset input pin
- CMOS compatible inputs
- Programmable MUX rate: 1 : 8 to 1 : 65
- Logic supply voltage: 1.7 to 3.3 V
- High voltage generator supply voltage: 2.4 to 4.5 V
- Display supply voltage: 3 to 9 V
- Low power consumption, suitable for battery operated systems
- Programmable row pad mirroring, for compatibility with both Tape Carrier Packages (TCP) and Chip-On-Glass (COG) applications
- Status read which allows for chip recognition
- Start address line, which allows for instance, the scrolling of the displayed image

- Slim chip layout, suited to COG and TCP applications
- Operating temperature: –40 to +85 °C.

2 APPLICATIONS

- Telecom equipment
- Portable instruments
- Point-of-sale terminals.

3 GENERAL DESCRIPTION

The PCF8814(1) is a low power CMOS LCD controller driver, designed to drive a graphic display of 65 rows and 96 columns. All necessary functions for the display are provided in a single-chip, including on-chip generation of PCF8814 LCD supply and bias voltages, resulting in a minimum of external components and low power consumption. The can be interfaced to microcontrollers via a serial bus.

(1) Type PCF8814MU/2DB/2 is sold under a Motif® license agreement. Motif® is a registered trademark of The Open Group in the US and other countries. Type PCF8814U/2DB/2 is not covered by a Philips/Motif License agreement. For further information on Motif licensed and unlicensed LCD drivers from Philips, please contact your local Philips office.

4 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PCF8814U/2DB/2	–	chip with bumps in tray	–
PCF8814MU/2DB/2	–	chip with bumps in tray	–