Introduction générale

Depuis longtemps, l'être humain rêve de créer des machines intelligentes capables d'effectuer des tâches à sa place. Ainsi, les humains auraient plus de temps à consacrer pour leurs loisirs, ou prendraient moins de risques pour effectuer des taches dangereuses. Or, créer une machine pouvant réaliser des tâches que seuls les humains sont normalement capable de faire n'est pas aussi simple qu'en le pense. En effet, sans toujours y penser, les tâches les plus élémentaires de la vie quotidienne peuvent devenir extrêmement complexes lorsqu'on les analyse de plus près. Bien que ça semble être des tâches simples, pour nous les humains, cela est loin d'être aussi évident pour une station « self-service ».

Or, dans notre vie quotidienne, l'énergie fossile est un élément essentiel pour tous les moyens de transports, soit terrestres, maritimes ou aériens, ou les usagers de ces moyens sont obligés de s'approvisionner en énergie dans des endroits baptisés « stations–services ».

Une station-service est une infrastructure positionnée à bord d'une route, d'une ville ou d'une autoroute destinée principalement à l'provisionnement du carburant pour les automobilistes. En effet, les stations services sont multiples par rapport aux prestations qu'elles fournies. Le client cherche toujours une meilleurs présence charge ainsi les qualités de services fournies, tell que les boutiques d'accessoires automobiles, station de gonflage de pneumatiques, petite mécanique et dépannage. Elles proposent également des services à destination des automobilistes : toilettes, épicerie, restauration, téléphone public ….etc.

Avec l'avancée technologique et la miniaturisation des cartes électroniques, l'être humain cherche toujours à trouver les meilleurs moyens d'optimisations pour rendre le système autonome, or le basculement d'une station service à une station « Self-service » où le rôle du pompiste demeure inutile, nécessite une modification sur le système de commande et l'introduction d'une interface de gestion qui sera manipulé par un seul chef pompiste. En effet, les stations « Self-service » sont composés de deux parties essentielles, mécanique et électronique. L'invention qui a contribué à un gigantesque saut en avant de l'électronique, est sans doute le microprocesseur.

Toutefois, la« carte mère » sur laquelle sont montés les microprocesseurs, les contrôleurs de périphériques (RAM, Disque dure, EEPROM..etc). Avec les progrès de la miniaturisation, et la généralisation de l'utilisation des ordinateurs, les besoins les plus courants sont standardisés et on a pu disposer toute une carte mère au sein d'une seule et même puce, appelée « **Microcontrôleur** ». L'usage de microcontrôleurs est actuellement en plein développement dans toute l'informatique industrielle, et à tous les degrés de complexité (de 8 pattes à près de 200 pattes). On trouve une multitude de modèles de microcontrôleurs et leur choix dépend de l'application voulue (à réaliser).

C'est dans cette diversité de service et d'exploitation dans le domaine du contrôle et du traitement numérique qu'il nous donne, que s'insère notre travail intitulé « Etude et réalisation d'une pompe self-service à base de microcontrôleur ». Il consiste à utiliser un microcontrôleur pour gérer une pompe carburant avec trois afficheurs cristaux liquides. La carte Arduino « UNO » qui est basé sur le microcontrôleur ATmega328 aura comme tâche de compter le nombre de litre distribués et calculer par suite le prix total équivalant en se basant sur le prix unitaire. Le comptage du nombre de litre sera fait grâce à un convertisseur qui fait correspondre la rotation d'un disque à des impulsions numériques, le disque est en relation direct avec une ellipse, cette dernière tourne à la vitesse de l'écoulement du carburant, ainsi, l'arrêt de la rotation du disque correspond à l'absence des imputions. Et aussi une interface graphique pour contrôler la pompe par pc.

La suite du mémoire est organisée en trois chapitres structurés de la manière suivante : Le premier (1^{er}) est consacré pour des définitions de la station service, en se basant surtout sur la pompe et le dispositif qui permet de calculer le nombre de litres du carburant propulsés par la pompe. Le deuxième ($2^{ème}$) est consacré pour des rappels sur la carte « Arduino », le microcontrôleur ATMega328, codeur rotatif, et les afficheurs à cristaux liquides utilisés pour l'affichage des trois paramètres Le troisième ($3^{ème}$) présente les deux parties, logiciel et réalisation pratique. La partie logicielle donne les différents organigrammes pour permettre ensuite la réalisation du programme et la maquette avec les LCD. Ce chapitre est le résultat des deux premiers chapitres.

Chapitre 1 Description d'une station self service

1.Introduction :

Dans ce chapitre nous allons faire une étude d'une station Self-service d'une manière générale, où on va détailler le fonctionnement d'une seule pompe et la manière dont le comptage du carburant se fait.

2. Les stations services :

Les stations service du carburant (**figure 1.1**) sont constituées généralement de plusieurs postes de distribution, selon les produits et les capacités de la station en elle-même. Les stations peuvent fourni une multitude de produits à savoir : Gas-oil, Essence sans plomb, Essence Super et Normal, ces postes de distributions sont connectés avec des réservoirs (**figure 1.2**) qui sont généralement enterrés pour protéger la station d'une éventuelle explosion soit prémédité ou accidentelle ou contre les agressions climatiques telles que la haute température, le froid... etc.

Figure 1.1 Station service

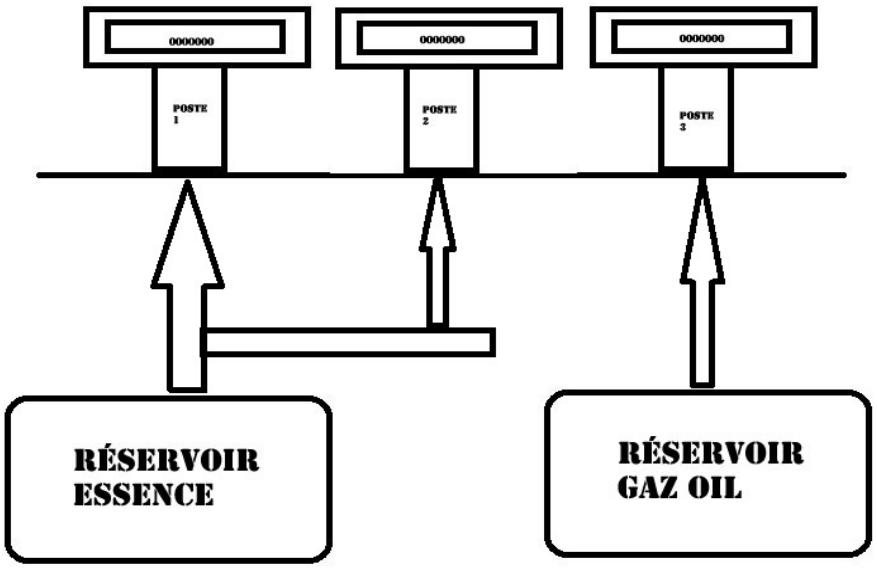

***Figure 1.* 2** Disposition des réservoirs avec les postes de distributions

2.1 Structure physiques :

La figure ci-dessous montre que la station « **Self- Service** » commande tous les blocs à distance par un chef pompistes qui peut contrôler tous les postes disponibles.

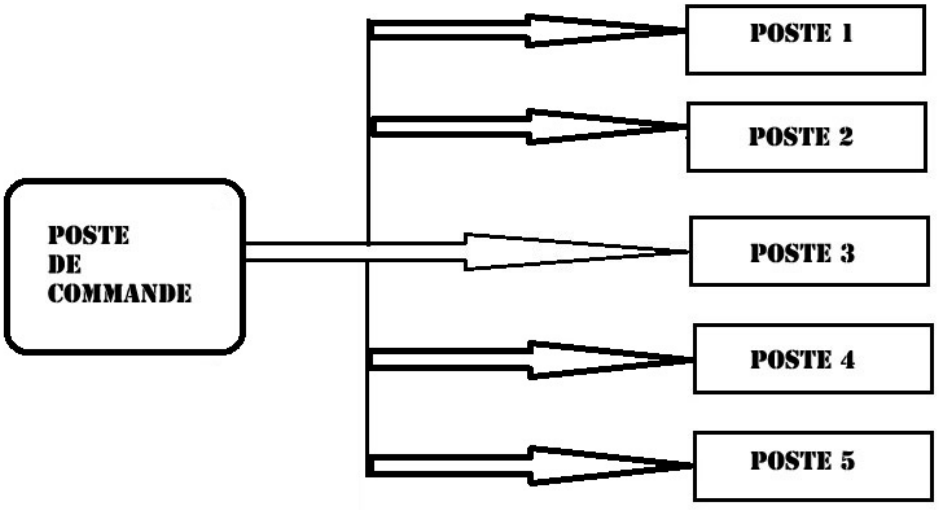

***Figure 1.* 3** Schéma bloc d'une station

2.2 Diagramme de service :

La figure ci-dessous **(figure 1.4**) montre le diagramme de service qui facilite la compréhension du fonctionnement et les différentes étapes que l'opérateur et les clients doivent suivre.

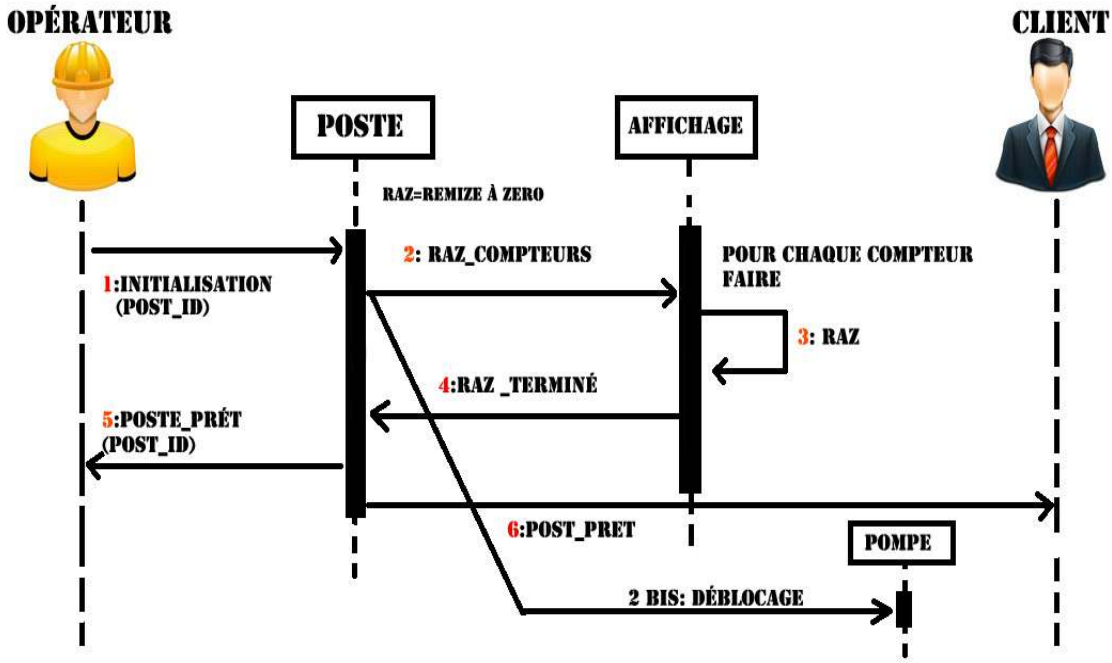

Figure 1.4 Diagramme de service

L'opérateur situé dans la cabine caisse initialise la transaction et met le compteur de volume à zéro pour démarrer la phase service. En effet, une fois le client tire le pistolet correspondant au carburant choisi, le moteur se met en marche et le compteur prix du carburant s'affiche avec le prix du carburant choisi. Le moteur propulse le carburant une fois que le client déclenche la gâchette du pistolet et le compteur de volume avec propulsion du liquide.

Le compteur de prix à payer évolue en conséquence et lorsque le client relâche la gâchette la propulsion du liquide est stoppée. Cette propulsion peut se faire plusieurs fois et lorsque le client met le pistolet à son emplacement la phase de service est terminée.

Remarque : Il est à signaler que notre travail consiste à détailler uniquement un seul poste de distribution (comptage, affichage ainsi la commande), en précisant que la tâche de l'opérateur est négligeable.

3. Développement d'un seul poste (Pompe) :

Chaque poste dispose une seule pompe, pour un seul type de carburant par poste, et le pistolet qui permet la possibilité de faire la distribution du carburant (**figure 1.5**)

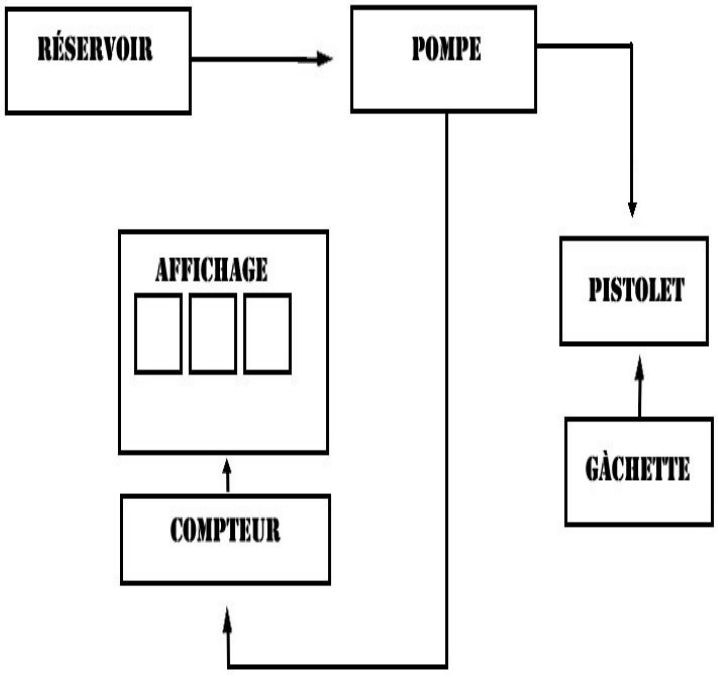

Figure 1. **5** Diagramme des déférentes parties de la pompe

Chaque pompe dispose les éléments essentiels suivants :

* **Un pistolet** avec une gâchette.

* **Un moteurs** (qui fait pompé le carburant du recevoir situer au sous sol).

* **Un compteur** qui peut mesurer la quantité du carburant propulsé et par la suite la conversion en dinar (que la tâche de l'opérateur est négligeable).

* **Trois afficheurs** (prix unitaire, nombre de litres et le prix total à payé).

Quand un client se présente à la station, il doit tout d'abord faire sa commande au niveau de la caisse, le chef pompiste lui indique la pompe libre pour avoir sa commande du carburant.

1- Le client tire le pistolet.
2- Le moteur se met en marche avec une vitesse constante.
3- Le compteur prix du carburant est affiché tant que le client le souhaite faire.
4- Le client déclenche la gâchette.
5- Le moteur propulse le carburant.
6- Le compteur du volume évolue en parallèle.
7- la vitesse du moteur diminue jusqu'à 50% quand le carburant propulsé par la pompe arrive à 80% de la consigne.
8- Le client relâche la gâchette.
9- Le moteur arrête la propulsion.
10- Le client remet en place le pistolet.
11- Le moteur est mis hors tension.

4. Fonctionnement de la pompe :

4.1. Vue générale :

L'appui sur la gâchette qui se trouve au niveau du pistolet, met automatiquement en marche le moteur (pompe), qui permet par la suite au carburant de circuler du réservoir source vers le réservoir client (Automobile, camion..). Ce carburant ainsi propulsé par la pompe doit faire tourné une ellipse, cette dernière est liée directement par un autre disque en fente, dans ce cas peut servir à définir la quantité du carburant libéré (**figure 1.6**)

Comment ?

Le disque est lié mécaniquement à un compteur à roue pour les anciens systèmes de pompe à essence.

Dans notre étude (on utilise un codeur rotatif de type incrémental avec un moteur à courant continué), le disque est placé sur un capteur à fourche (émission et réception infrarouge) le disque ayant des trous lorsqu'il tourne il coupe l'émission infrarouge.

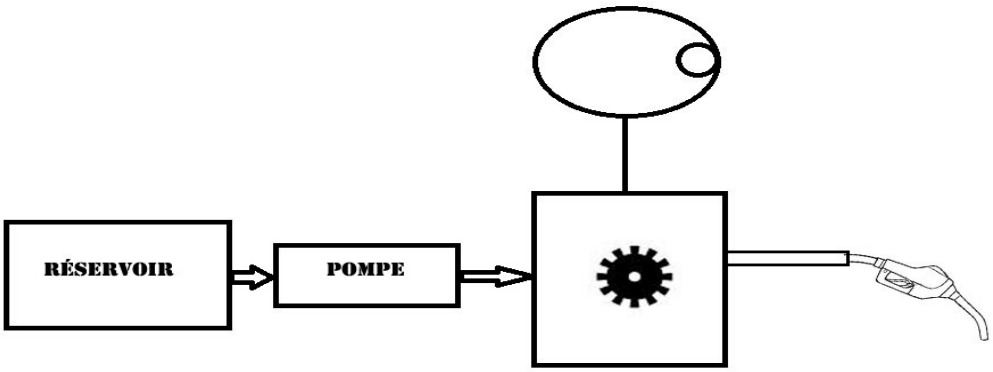

***Figure 1.* 6** Disque et le capteur.

Ce phénomène va se traduire par des impulsions électroniques au niveau du capteur qui peuvent alors servir à compter le nombre de litres propulsées.

4.2. Types de moteurs :

Ce qui nous intéresse de plus c'est les pompes (moteurs) puisque la puissance d'écoulement et le débit du carburant influent directement sur la rotation du disque. Donc sur la mesure du nombre de litre. La société **NAFTAL** utilise deux types de moteurs pour le pompage :

- **Premier type de moteur :**

  Ce type de moteur dispose les caractéristiques suivantes :

  - 5 m^3 /h
  - 90 L / 1min
  - 1 L / 0.6s

- **Deuxième moteur :**

  Ce type de moteur est celui utilisé dans notre cas, puisque chaque litre du carburant est l'équivalent de 75 impulsions, donc il facilite notre travail du point de vue conversion litre/impulsion.

  - 3m^3 /h
  - 50 L/1min.
  - 1 L/1.2s.

5. Exemple d'un distributeur série :

5.1 Notice descriptive

***Figure 1.* 7** Exemple d'un distributeur série GPL Auto

Une pompe **GPL** refoule le produit jusqu'à l'ensemble de mesurage.

Après passage du liquide dans les déférents dispositifs : électrovanne, filtre, dégazeur et un clapet anti-retour empêchant tout retour du liquide vers le dégazeur.

Le liquide s'écoule alors dans le mesureur lequel en traine l'indicateur.

Après avoir traversé le mesureur, le **GPL** passe dans la vanne de maintien de pression différentielle, cette vanne assure une sécurité en cas d'apparition de phase gazeuse dans le mesureur. Elle est raccordée à la phase gazeuse et cette dernière crée une pression qui s'additionne à celle du ressort taré.

La vanne de maintien de pression différentielle est réglée à une pression supérieure de 1 bar à la tension de vapeur saturante du liquide mesuré.

Un limiteur de débit dont l'obturation automatique se situe à environ 70 L/mn à pour rôle d'éviter tout écoulement après rupture du dispositif anti-arrachement ou écoulement du flexible, il est également pour vue d'une soupape de sécurité tarée à 22 **bar** protégeant l'ensemble de mesurage de toute augmentation de pression accidentelle, le limiteur de débit, le flexible, le dispositif anti-arrachement et le liquide arrive au robinet et ne peut être livré qu'après avoir raccordé ce dernier au réservoir du véhicule (**figure 1.8**)

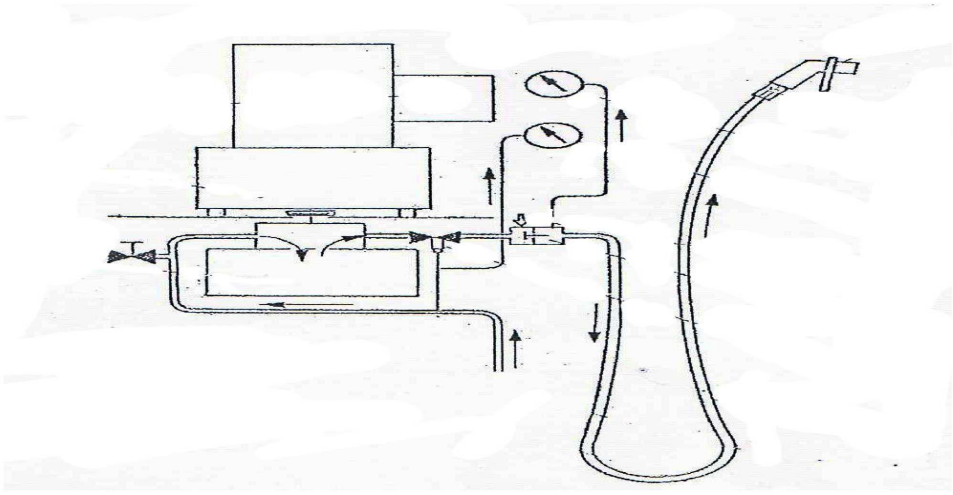

Figure 1.8 schéma hydrologue d'une pompe GPL

6. Conclusion :

A la fin de ce chapitre, ont peut dire qu'on est arrivé à éclaircir le fonctionnement d'une station de pompe carburant dans son cadre général. le travail demandé est de rendre le poste plus fiable, plus conviviale et pratique en lui ajoutant une carte ARDUINO afin de remplacer le comptage mécanique des anciens postes en comptages et affichages électronique avec la possibilité de rendre le poste capable de fournir le service de près payement déjà cité au début de ce chapitre.

Chapitre 2 Généralités sur la carte Arduino, LCD et codeur rotatif.

1. Présentation générale de l'ARDUINO :

L'Arduino est une «plate-forme de prototype électronique open-source» permettant d'utiliser simplement un microcontrôleur ATMEL AVR.

Le système Arduino est une carte électronique **(figure 2.1)** basée autour d'un microcontrôleur et de composants minimum pour réaliser des fonctions plus ou moins évoluées à bas coût. Elle possède une interface USB pour la programmer. C'est une plateforme open-source qui est basée sur une simple carte à microcontrôleur (de la famille AVR), et un logiciel, véritable environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte à microcontrôleur. Arduino peut être utilisée pour développer des applications matérielles industrielles légères ou des objets interactifs (création artistiques par exemple), et peut recevoir en entrées une très grande variété de capteurs. Arduino peut aussi contrôler une grande variété d'actionneurs (lumières, moteurs ou toutes autres sorties matériels). Les projets Arduino peuvent être autonomes, ou communiquer avec des logiciels sur un ordinateur (Flash, Processing ou MaxMSP). Les cartes électroniques peuvent être fabriquées manuellement ou bien être achetées pré assemblées le logiciel de développement open-source est téléchargeable gratuitement.

Par sa simplicité d'utilisation, Arduino est utilisée dans beaucoup d'applications comme l'électronique industrielle et l'informatique embarquée, la modélisation, la robotique et la domotique, et bien plus encore.

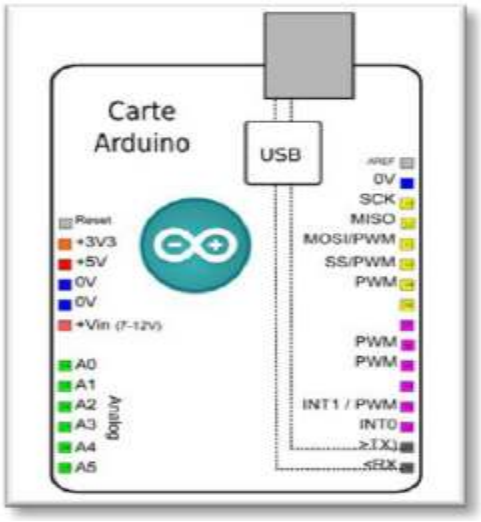

Figure 2.1 carte Arduino

2. Critère de choix :

Malgré la disponibilité de nombreux microcontrôleurs et de nombreuses plateformes de développement des fonctions comparables à Arduino, la majorité d'entre-elles présente soit des limitations d'exploitation, ou des licences limitées, et bien plus encore.

Il suffit d'utiliser n'importe quelle autre plateforme propriétaire pour se rendre compte des avantages indéniables et confirmés du système Arduino : l'alliance entre un hardware et un software garantissent la pérennité de ce système de développement ce qui lui donne toute sa puissance.

Donc, Arduino se distingue par rapport aux autres dans les points suivants :

- ✓ **Pas cher :** les cartes Arduino sont relativement peu coûteuses comparativement aux autres plateformes. La moins chère des versions du

module Arduino peut être assemblée à la main, et même les cartes Arduino pré-assemblées coûtent relativement moins de 25 Euros (microcontrôleur inclus...).

- ✓ **Multiplateformes :** Le logiciel Arduino, écrit en Java, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux. La plupart des systèmes à microcontrôleurs sont limités à Windows.
- ✓ **Un environnement de programmation clair et simple:** L'environnement de programmation Arduino est facile à utiliser, tout en étant assez flexible pour que les utilisateurs avancés puissent en tirer profit également.
- ✓ **Pour l'enseignement :** Arduino est compatible avec l'environnement **visuel basic** MATLAB, SIMULINK, C #, C++, JAVA... Ainsi d'autres logiciels.
- ✓ **Logiciel Open-Source et extensible :** Le logiciel Arduino et son langage sont publiés sous licence open source, disponible pour être complété par des programmateurs expérimentés.
- ✓ **Existence de « Shields » :** Ce sont des modules supplémentaires **(figure2.2)** qui se connectent parfaitement sur la carte Arduino pour augmenter les possibilités comme par exemple : afficheur graphique couleur, interface Wifi, Interface de puissance, etc.

Figure 2.2 Quelque Shields pour Arduino

3. Carte Arduino UNO :

3.1 Synthèse des caractéristiques :

Microcontrôleur	Atmega328
Tension de fonctionnement	5V
Tension d'alimentation (recommandée)	7-12V
Tension d'alimentation (limites)	20V
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)
Broches d'entrées analogiques	6 (utilisables en broches E/S numériques)
Intensité maxi disponible par broche E/S(5V)	40 mA
Intensité maxi disponible pour la sortie 3.3V	50 mA
Intensité maxi disponible pour la sortie 5V	Fonction de l'alimentation utilisée – 500 mA max si port USB utilisé seul
Mémoire Programme Flash	32 KB (Atmega328) dont 0.5 KB sont utilisés par le Bootloader
Mémoire SRAM (mémoire volatile)	2 KB (Atmega328)
Mémoire EEPROM (mémoire non volatile)	1 KB (Atmega328)
Vitesse d'horloge	16 MHz

Tableau 2.1 Synthèse des caractéristiques de la carte Arduino UNO

3.2 Les éléments du la carte UNO :

L'Arduino de base possède un microcontrôleur, qui rassemble la mémoire (morte et vive), le processeur et les périphériques d'entrées et sorties , des entrées et sorties USB et des prise d'alimentation pour transmettre les informations et alimenter l'Arduino avec une tension entre 7 et 12V, permettant d'utiliser différents types d'alimentations, des LED qui servent surtout à indiquer l'état d'activité de l'Arduino, un bouton reset manuel, un entête ICSP qui permet de programmer le microcontrôleur sans passer par la «couche» Arduino; une puce FTDI USB qui convertit l'information USB et enfin des pins dans tous les sens.

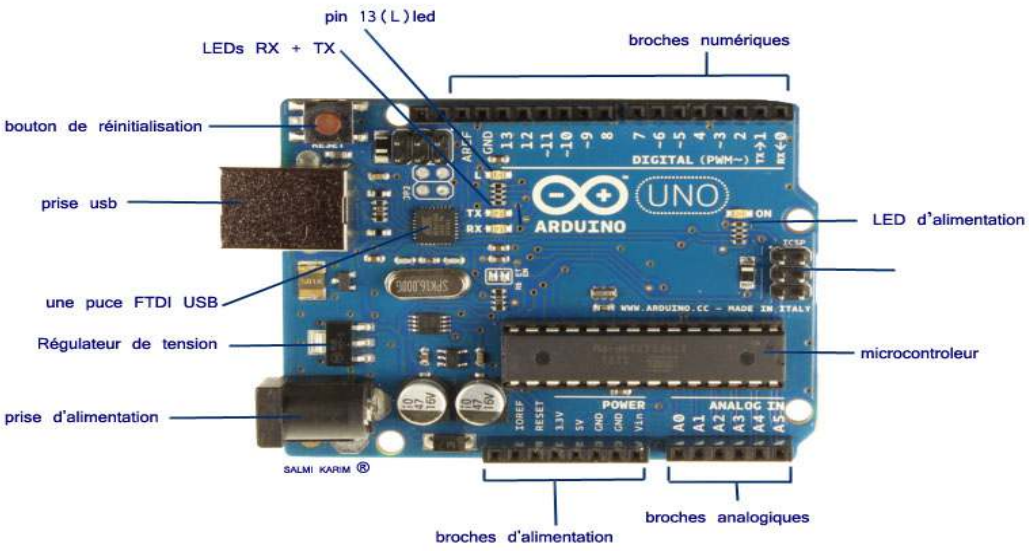

Figure 2.3 Les éléments du la carte Arduino UNO

✓ **Alimentation :**

La carte Arduino UNO peut être alimentée par l'USB ou par une alimentation externe. La source est sélectionnée automatiquement. La tension d'alimentation extérieure (hors USB) peut venir soit d'un adaptateur AC-DC ou de piles. L'adaptateur peut être connecté grâce à un **jack** (prise d'alimentation) de 2.1mm positif au centre. Le

raccordement vers un bloc de piles peut utiliser les bornes Gnd et Vin du connecteur d'alimentation (POWER).

Ce circuit inclut un régulateur de tension à 5 V mais il doit être alimenté entre 6 et 20 V. On conseille en général de l'alimenter plutôt entre 7 et 1 2 V pour garder une marge en basse tension et éviter que le circuit ne chauffe trop (car le régulateur de tension disperse toute surtension en chaleur)**.**

- ✓ **Mémoire :**

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

- ✓ **Entrées et sorties :**

Chacune des 14 broches numériques de la **UNO** peut être utilisée en entrée (input) ou en sortie (output) Elles fonctionnent en logique TTL (0V-5V) , chacune pouvant fournir (source) ou recevoir un courant maximal de 40 mA . Certaines broches ont des fonctions spécialisées :

• Serial : broche 0 (RX) et broche1 (TX). Permet de recevoir (RX) et de transmettre (TX) des données séries TTL.

• Interruptions externes 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur un front montant ou descendant, ou encore sur le changement de valeur.

• PWM : 3, 5, 6, 9, 10, et 11. Output 8-bit de PWM avec la fonction analogWrite().

• LED : 13. Il y a une LED connectée à la broche numérique 13.

La carte **UNO** a 6 broches d'entrée analogiques, A0 à A5, chacune avec 10 bits de résolution (1024 valeurs différentes).

• I2C : Permettent le support du bus I2C .

3.3 Schéma simplifié de la carte Arduino UNO :

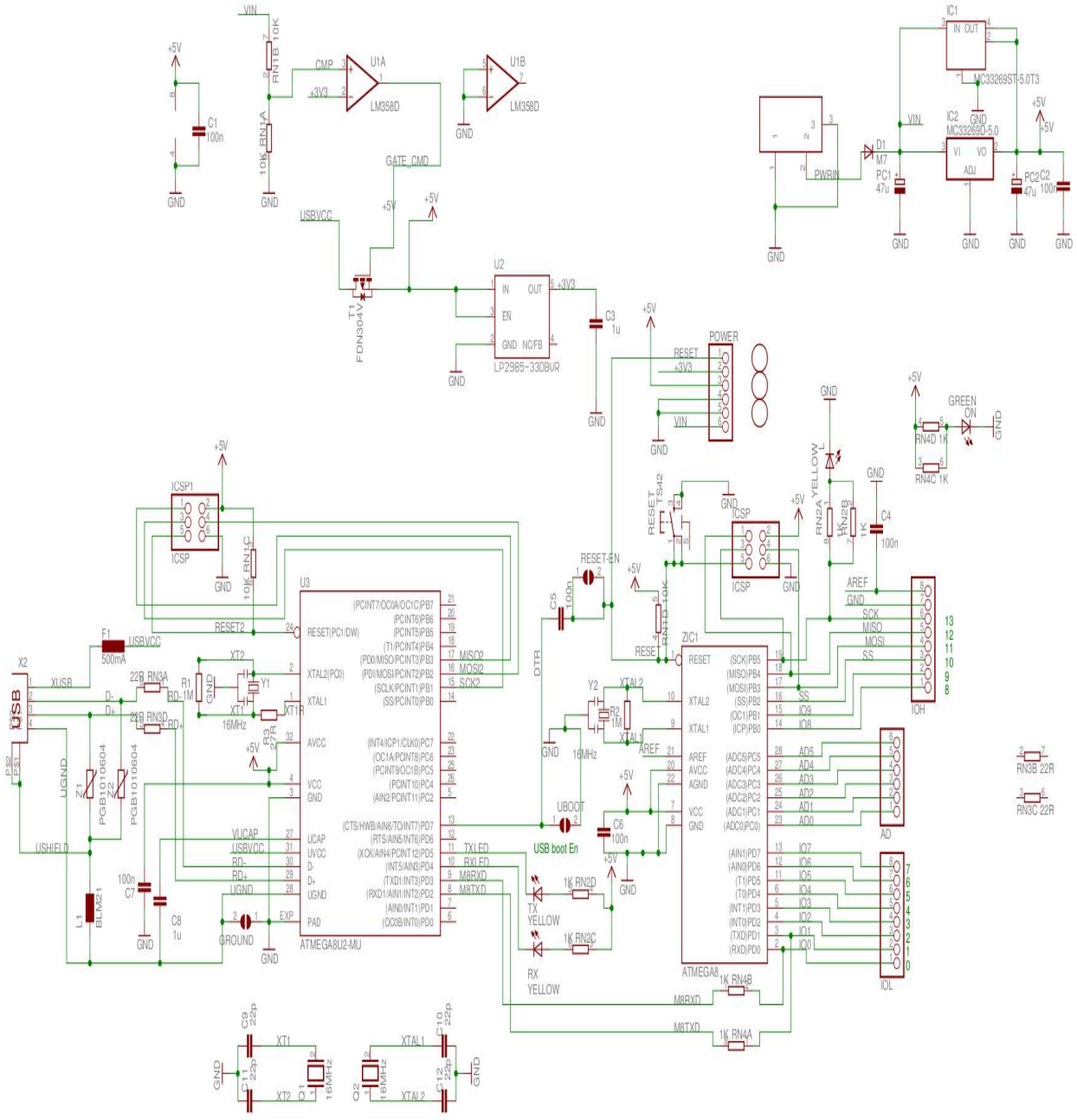

Figure 2.4 Schéma de carte Arduino UNO

4. Microcontrôleur ATMega328 :

Le modèle UNO de la société ARDUINO est une carte électronique dont le cœur est un microcontrôleur ATMEL de référence ATMega328. Ce dernier est un microcontrôleur 8bits de la famille AVR dont la programmation peut être réalisée en langage C.

4.1 Qu'est-ce qu'un microcontrôleur ?

Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte pour le programme, mémoire vive pour les données), unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique (quelques milliwatts en fonctionnement, quelques nano watts en veille), une vitesse de fonctionnement plus faible (quelques mégahertz à quelques centaines de mégahertz) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

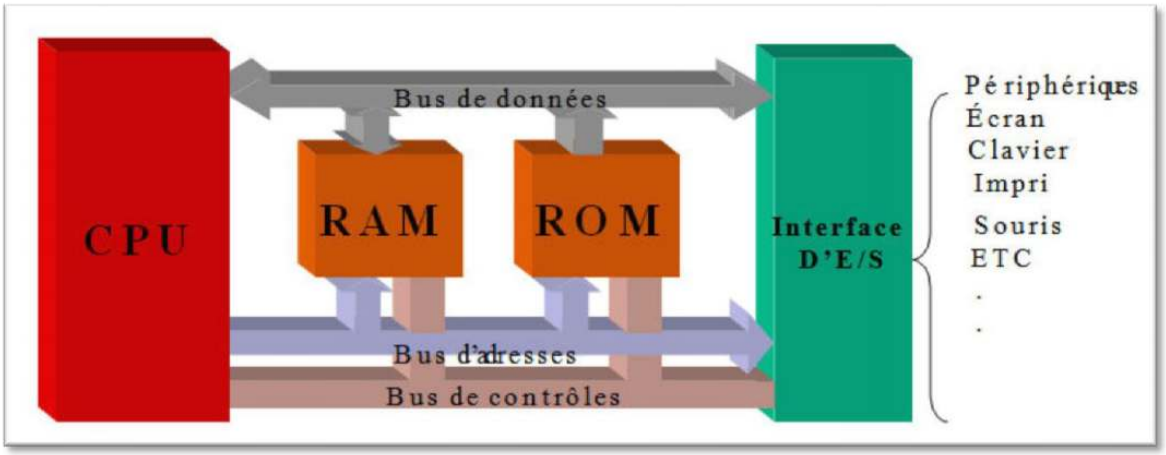

Figure 2.5 Schéma synoptique de microcontrôleur

4.2 Les principales caractéristiques du ATMEGA328 :

FLASH= mémoire programme de 32Ko

RAM= données (volatiles) 2Ko

EEPROM= données (non volatiles) 1Ko

Digital I/O (entrées-sorties Tout Ou Rien)= 3 ports PortB, PortC, PortD (soit 23 broches en tout I/O)

Temporisateurs / compteurs: Temporisateur 0 , Temporisateur 2 ,Timer0 (comptage 8 bits) et Timer1 (comptage 16bits)

Chaque temporisateur peut être utilisé pour générer deux signaux PWM. (6 broches OCxA/OCxB)

Plusieurs broches multifonctions : certaines broches peuvent avoir plusieurs fonctions différentes choisies par programmation

PWM= 6 broches OC0A (PD6), OC0B (PD5), 0C1A (PB1), OC1B (PB3), OC2A (PB3), OC2B (PD3)

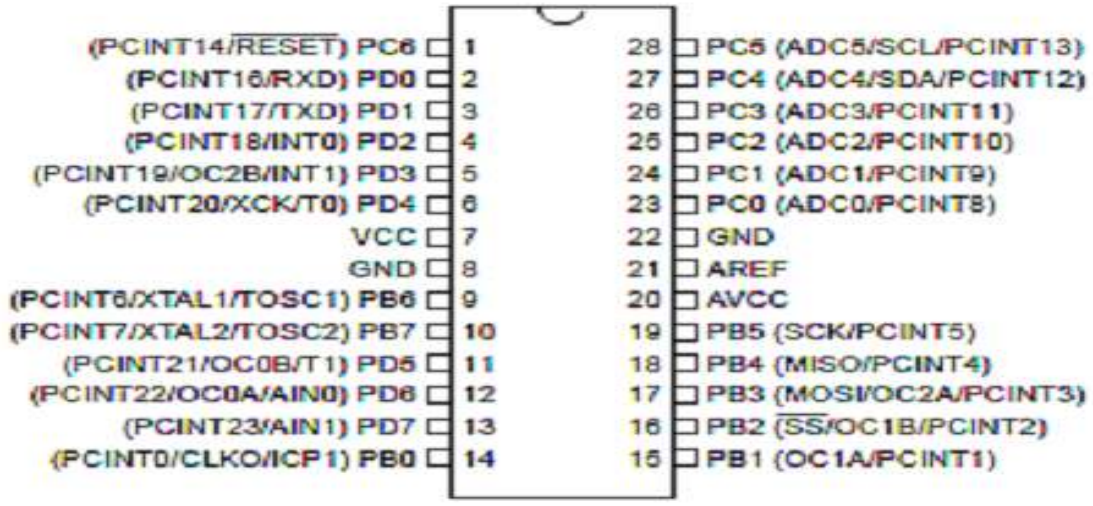

Figure 2.6 schéma de microcontrôleur ATMega328

Convertisseur analogique-numérique (résolution 10bits) = 6 entrées multiplexées ADC0(PC0) à ADC5(PC5)

Gestion bus I2C = le bus est exploité via les broches SDA(PC5)/SCL(PC4).

Port série = émission/réception série via les broches TXD(PD1)/RXD(PD0)

Comparateur Analogique= broches AIN0(PD6) et AIN1 (PD7) peut déclencher interruption

Gestion d'interruptions (24 sources possibles : en résumé

– Interruptions liées aux entrées INT0 (PD2) et INT1 (PD3)

– Interruptions sur changement d'état des broches PCINT0à PCINT23

– Interruptions liées aux Timers 0, 1 et 2

– Interruption liée au comparateur analogique

– Interruption de fin de conversion ADC

– Interruptions du port série USART

– Interruption du bus (I2C)

4.3 Structure interne de l'ATMega328 :

L'utilisation des périphériques intégrés (Entrées Sorties , Timers, ...) repose sur l'exploitation (lecture/écriture) de registres internes. Ces registres (8 bits), sont décrits par un nom, y compris dans la programmation en C. Cette section fournit quelques détails importants sur les registres internes du microcontrôleur ATMega328

Notation : par la suite, pour un registre nommé **R**, la notation **R.n** désigne le bit de rang n du registre R. Le bit R.0 est le bit de poids faible de **R**.

4.3.1 Entrées / Sorties numériques :

Souvent, les microcontrôleurs disposent de broches d'entrée/sortie TOR, comme sur un automate programmable industriel .Pour placer l'état d'une sortie à 0 ou 1, ou lire l'état d'une entrée, il faut exploiter des registres internes, décrits ci-dessous. Les entrées-sorties sont réparties dans 3 groupes de broches appelés ports. Le port B regroupe les broches notées **PBx**, le port C les broches **PCx** et le port D les broches **PDx**. Chaque port est configuré/exploité grâce à 3 registres.

PORTx= pour l'écriture de valeurs en sortie.

DDRx = détermine la direction de chaque broche du port (1-sortie 0-entrée).

PINx= permet la lecture de la valeur en entrée.

Direction des ports : si le bit DDRB.2 vaut 1 alors la broche PB2est une sortie TOR.

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

Ex: DDRB.5=1(doncPB5 en sortie) écrire 0 ou 1 dans le bit PORTB.5 permet de définir l'état de la sortie PB5.

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

Ex: DDRB.4=0(donc PB4en entrée), lire PINB.4 permet de connaître l'état de l'entrée PB4.

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Figure 2.7 les 3 registres pour la configurations du port B

4.3.2 Interruptions Externes

Un événement d'interruption dirige le flux d'exécution du programme à un morceau totalement indépendant du code, connu sous le nom "d'interruption sous-programme". Il existe de nombreuses sources d'interruptions qui sont disponibles pour un microcontrôleur. La plupart d'entre eux sont générés par les modules internes et sont connus comme les interruptions internes. Et il ya deux broches qui acceptent les signaux externes directement et permet aux sources externes pour générer des interruptions. Les broches impliquées doivent être configurées en entrée .Broches INT0 (PD2)/INT1(PD3) : configurables pour déclencher les interruptions (n° 2 et 3) sur niveau 0, front négatif ou positif. La cause d'interruption est choisie par le registre

L'interruption externe 1 est activé par le INT1 broche externe. Le niveau et les arêtes sur l'axe de INT1 externe qui activent l'alarme sont définis dans le tableau ci-dessous.

figure 2.8 registre ELCRA

EICRA – External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

pour générer une interruption.

ISC01	ISC00	
0	**0**	Le faible niveau de **INTx** génère une demande d'interruption
0	**1**	Tout changement logique sur **INTx** génère une demande d'interruption
1	**0**	Le front descendant du **INTx** génère une demande d'interruption
1	**1**	Le front montant du **INTx** produit dans une demande d'interruption

Tableau 2.2 Le contrôle de sens d'interruptions.

4.3.3 Temporisation / comptage de ATMega328 *:*

Les microcontrôleurs **ATMega328** disposent de modules de temporisation/comptage internes, fonctionnant pour certains avec des registres de comptage sur 8 bits, et pour d'autres sur 16 bits. Dans tous les cas, chaque événement de comptage conduit à une modification du registre de comptage (+1 ou -1).L'événement de comptage peut être un "tick" de l'horloge du microcontrôleur, ce qui revient à mesurer l'écoulement du temps. L'événement de comptage peut aussi être un front sur une broche d'entrée du microcontrôleur (les broches T0 et T1 peuvent servir d'entrée de comptage).

*__Fonction Temporisateur__ : lorsqu'on compte des "ticks" de l'horloge qui cadence le microcontrôleur, on mesure du temps. Les modules temporisation/comptage permettent de compter les ticks du signal d'horloge ou d'un signal de fréquence plus faible.

*__Fonction Compteur__ : lorsque l'on compte des fronts sur une entrée de comptage (broches T0 ouT1), on utilise alors la fonction "compteur" du module.

Le choix entre fonction de temporisation et fonction de comptage se fait par paramétrage de registres dédiés à la gestion des modules temporisation/comptage.
***Génération de signaux périodiques** : les modules temporisation/comptage sont assez complexes et chacun de ces modules peut générer des signaux **PWM** dont le rapport cyclique est facilement modifiable.

Remarque : ces périphériques intégrés sont assez complexes (environ 70 pages du datasheet de L' ATMega). Seule une vision simplifiée est fournie ici.

4.3.4 PWM (Pulse Width Modulation):

Un signal PWM est défini par son rapport cyclique (le temps au niveau haut par rapport à la période totale) mais aussi par sa fréquence. Grâce à l'environnement de programmation simplifié, une PWM s'obtient sur une Arduino avec la fonction **analogWrite** utilisé sur une patte numérique digitale 3, 5, 6, 9, 10 ou 11.

- ✓ **Modulation de largeur d'impulsion PWM :**

Est une technique pour fournir une puissance électrique à une charge qui a une réponse relativement lente. Le signal d'alimentation est constitué d'un train d'impulsions de tensions de telle sorte que la largeur des impulsions individuelles de contrôler le niveau de tension efficace à la charge. Les deux signaux AC et DC peuvent être simulés avec PWM. Dans ces notes, nous allons décrire l'utilisation de PWM sur un Arduino pour les LED de contrôle et moteurs à courant continu.
Le train d'impulsions PWM agit comme un signal de courant continu lorsque les périphériques qui reçoivent le signal ont un temps de réponse électromécanique qui est plus lente que la fréquence des impulsions. Pour un moteur à courant continu, le stockage d'énergie dans les enroulements du moteur atténue efficacement les salves d'énergie fournie par les impulsions d'entrée de sorte que le moteur subit une entrée de puissance électrique plus ou moins grande en fonction de la largeur des impulsions. Pour une LED, en utilisant PWM fait que la lumière est allumée et éteinte à la fréquence que nos yeux peuvent détecter. Nous percevons tout simplement la lumière plus ou moins brillants en fonction de la largeur des impulsions à la sortie PWM.

La figure2.9 montre un signal de tension comprennent des impulsions de durée pour que la répétition de toutes les unités de temps Tc. La sortie d'un canal de PWM est soit Vs volts durant l'impulsion de zéro volt ou autrement. Si ce signal est fourni en entrée d'un dispositif qui a un temps beaucoup plus grande que Tc de réponse, le dispositif connaîtra le signal en entrée d'environ DC avec une tension efficace de $\mathbf{Veff} = \boldsymbol{Vs}\ \frac{\tau o}{\tau c}$

Le rapport $\frac{\tau o}{\tau c}$ est appelé **le rapport cyclique** des impulsions à ondes carrées. La tension continue effective fournie à la charge est contrôlée en ajustant le rapport cyclique.

Figure 2.9 le rapport cyclique de pwm.

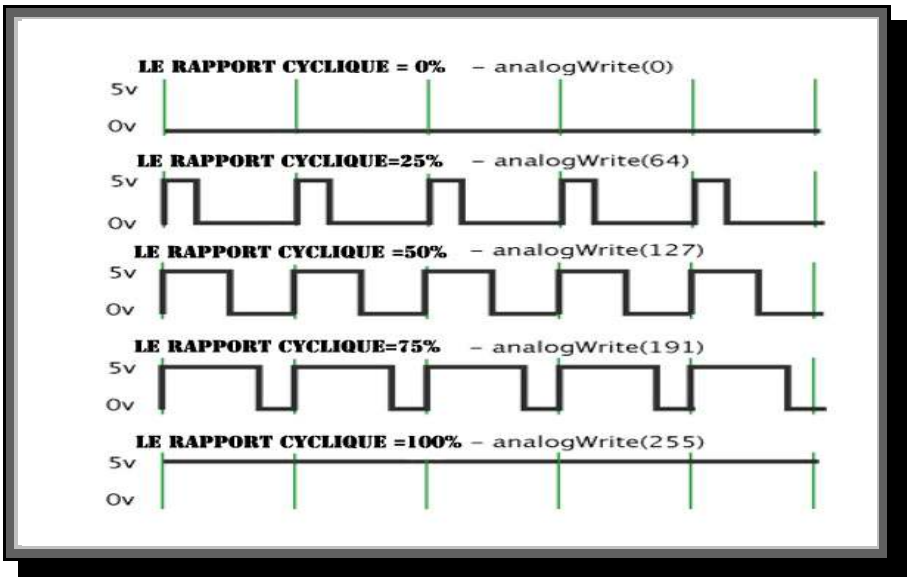

Figure 2.10 exemple du rapport cyclique

✓ **La fréquence de la PWM**

Cette fréquence est fixe, elle ne varie pas au cours du temps. Pour la carte UNO elle est d'environ 490Hz.

5 La liaison série

La carte Arduino UNO a de nombreuses possibilités de communications avec l'extérieur. L'Atmega328 possède une communication série UART TTL (5V), grâce aux broches numériques 0 (RX) et 1 (TX).Un contrôleur Atmega8U2 sur la carte, gère cette communication série vers l'USB et apparaît comme un port de communication virtuel pour le logiciel sur l'ordinateur. Le firmware de l'8U2 utilise le protocole USB, et aucun driver externe n'est nécessaire. Windows a cependant besoin d'un fichier .inf., à l'installation. Le logiciel Arduino possède un logiciel série intégré permettant l'envoi et la réception de texte. Les LEDs RX et TX sur la carte clignoteront pour indiquer la transmission de données vers l'ordinateur. Une librairie 'Software Serial' permet la transmission de données série à partie de chacune des broches numériques du UNO.

La voie série permet de communiquer de manière directe et unique entre deux supports. Ici, elle se fera entre un ordinateur et la platine Arduino, mais elle pourrait aussi se faire par exemple entre deux cartes Arduino. Dans sa forme la plus simple, elle ne nécessite que 3 fils : 2 pour l'émission/réception et 1 pour la masse.

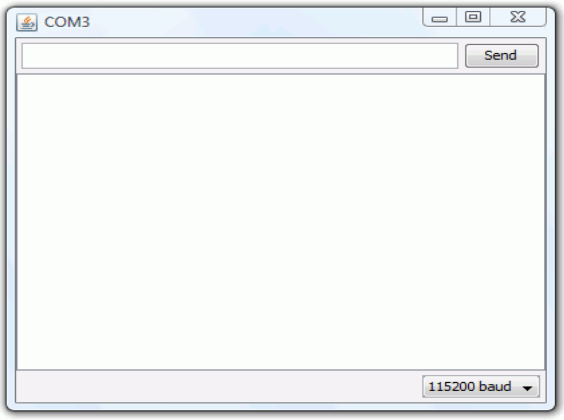

Figure 2.11 Ecran du moniteur série du L'Arduino

5.1 Principe de la voie série

Pour faire des communications entre différents supports, il existe différents moyens. Pour n'en citer que quelques-uns, on retrouve les bus CAN, le bus I²C, l'Ethernet, etc. et la liste est longue. Dans notre cas, nous allons étudier la **communication série**, aussi appelée **RS232**, puisqu'elle est intégrée par défaut dans la carte Arduino.

5.2 Les types de liaison série

Le premier type est la liaison **simplex**. Il n'y a qu'un émetteur et un seul récepteur. Par exemple, seul l'ordinateur peut envoyer des données à la carte Arduino. Ça nous n'est pas très utile si on veut faire le contraire. On n'utilisera donc pas ce type de liaison.

Le deuxième est la **liaison semi-duplex**. En fait, c'est un peu lorsque l'on communique à quelqu'un avec un talkie-walkie. L'un parle pendant que l'autre écoute. Nous n'utiliserons pas ce type de communication.

Le dernier est la liaison **full-duplex**. Là, c'est un peu comme le téléphone, chacun peut parler et écouter en même temps ce que l'autre dit. Avec Arduino, c'est de ce type de communication que nous disposons. Ce qui est bien pratique afin d'éviter d'attendre que l'on ait réceptionné ce que l'ordinateur envoie pour ensuite lui émettre des données.

5.3 Matériel série

Le matériel série envoie et reçoit des données sous la forme d'impulsions électriques qui représentent les bits séquentiels. Les zéros et les uns qui transportent les informations qui composent un octet peuvent être représentées de différentes manières. Le modèle utilisé par Arduino est 0 volt pour représenter une valeur de bit égale à 0, et 5 volts (ou 3,3 volts) pour représenter une valeur de bit égale à 1. La carte comme **UNO**, est un circuit pour convertir le port série matériel de la puce Arduino en une connexion **USB** pour la liaison au port série matériel. D'autres cartes,

comme le Mini, le Pro, le Pro Mini ne prennent pas en charge l'USB et nécessitent un adaptateur pour se connecter à un ordinateur et convertir les signaux TTL en USB. Certains périphériques série utilisent le standard RS-232 pour la connexion série. Ils ont en général un connecteur à neuf broches, et un adaptateur est nécessaire pour les utiliser avec l'Arduino. La norme RS-232 est un protocole de communication ancien qui utilise des niveaux de voltage incompatibles avec les broches numériques de l'Arduino.

Nom du port	Broche de transmission	Broche de réception
Serial	USB	USB
Serial1	0	1

Tableau 2.3 Ports série de l'Arduino UNO

5.4 La vitesse de communication :

Quand on va utiliser la voie série, on va définir la vitesse à laquelle sont transférées les données. En effet, comme les bits sont transmis un par un, la liaison série envoie les données en un temps prédéfini. Par exemple, on pourra envoyer une totalité de 9600 bits par secondes (9600 bps). Avec cette liaison, on peut envoyer entre 75 et 115200 bits par secondes ! Ce sera à nous de définir cette vitesse.

6 Afficheur à cristaux liquides(LCD) :

Les afficheurs à cristaux liquides, autrement appelés afficheurs LCD, sont des modules compactes nécessitent peu de composants externes pour un bon fonctionnement. Ils consomment relativement peu (de 1 à 5 mA), sont relativement bons marchés et s'utilisent avec beaucoup de facilité. Ils sont très utilisés dans les montages à microcontrôleur, et permettent une grande convivialité. Ils peuvent aussi être utilisés lors de la phase de développement d'un programme, car on peut facilement afficher les valeurs de différentes variables.

Figure 2.1 2 Faces d'un afficheur LCD 2*16

6.1 Principe de fonctionnement

L'afficheur comporte d'autres composants que l'afficheur à cristaux liquides(LCD) seul. Un circuit intégré de commande spéciale, le contrôleur-LCD est chargé de la gestion du module. **Le « contrôleur »**remplit une double fonction : d'une part, il commande l'affichage et d'autre se charge de la communication avec l'extérieur.

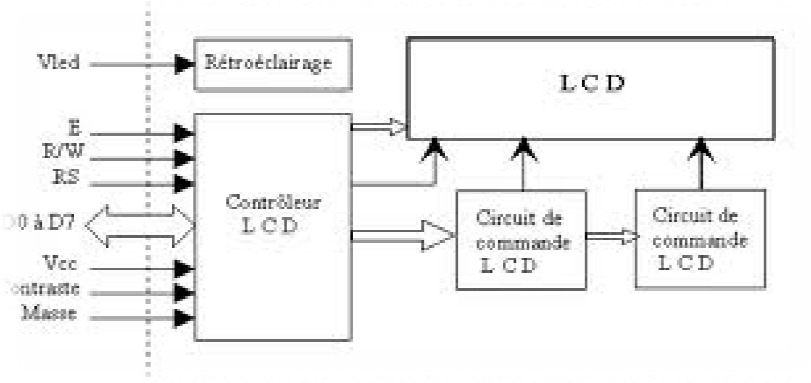

Figure 2.13 Schéma fonctionnel d'un LCD

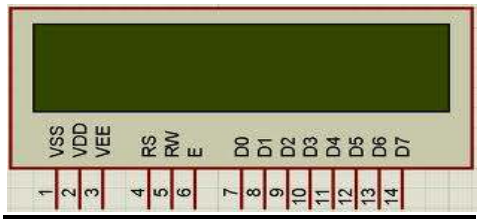

Figure 2.14 Le brochage de l'afficheur LCD* 16

6.2 Brochage :

Broche	Nom	Niveau	Fonction
1	Vss	-	Masse
2	Vdd	-	Alimentation positive +5V
3	VEE	0-5V	Cette tension permet, en la faisant varier entre 0 et +5V, le réglage du contraste de l'afficheur.
4	RS	TTL	Sélection du registre Grâce à cette broche, l'afficheur est capable de faire la différence entre une commande et une donnée.
5	R/W	TTL	Lecture ou écriture L : Écriture H : Lecture
6	E	TTL	Entrée de validation (Enable) active sur front descendant. Le niveau haut doit être maintenue pendant au moins 450 ns à l'état haut.
7	D0	TTL	Bus de données bidirectionnel 3 états (haute impédance lorsque E=0)
8	D1	TTL	
9	D2	TTL	
10	D3	TTL	
11	D4	TTL	
12	D5	TTL	
13	D6	TTL	
14	D7	TTL	
15	A	-	Anode rétro éclairage (+5V)
16	K	-	Cathode rétro éclairage (masse)

Tableau 2.4 Brochage du LCD

6.3 Le jeu de commandes standard :

Instructions	Code										Description	Durée
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
Effacer Afficher	0	0	0	0	0	0	0	0	0	1	Efface l'ensemble de la mémoire de donnée sans toucher au générateur de caractères. Ramène le curseur en position « home », à l'adresse 00.	1,64 ms
Retour à la maison	0	0	0	0	0	0	0	0	1	X	Ramène le curseur en position « home », à l'adresse 00. Si l'affichage était décalé, il est remis à sa position d'origine : l'adresse 00 se trouve à nouveau en haut à gauche.	1,64 ms
Mode Entrée ensemble	0	0	0	0	0	0	0	1	I/D	S	Définit le sens de déplacement du curseur après l'apparition d'un caractère (vers la gauche si I/D=1, vers la droite si I/D=0) et si l'affichage accompagne le curseur dans son déplacement ou non (S).	40 µs
Afficher commande marche / arrêt	0	0	0	0	0	0	1	D	C	B	Met l'affichage en ou hors fonction l'affichage (D), le curseur (C), le clignotement du curseur (B).	40 µs

Curseur et l'affichage décalage	0	0	0	0	0	1	S/C	R/L	X	X	Déplace le curseur (S/C=1) ou l'affichage (S/C=0) d'une position vers la gauche (R/L=1) ou la droite (R/L=0) sans changer la DD RAM.	40 µs
fonction set	0	0	0	0	1	DL	N	F	X	X	Définit la taille de l'interface (DL=0 pour mode 4 bits, DL=1 pour mode 8 bits), le nombre de lignes (NL=0 pour 1 ligne, N=1 pour 2 ou 4 lignes)	40 µs
Set CG RAM adresse	0	0	0	1	A5	A4	A3	A2	A1	A0	Définit l'adresse de la CG RAM. Les données de la CG RAM sont envoyées après cette commande.	40 µs
Set DD RAM adresse	0	0	1	A6	A5	A4	A3	A2	A1	A0	Définit l'adresse de la DD RAM. Les données de la DD RAM sont envoyées après cette commande.	40 µs
Lire busy flag & adresse	0	1	BF	A6	A5	A4	A3	A2	A1	A0	Lit le flag busy (BF), et l'adresse de la position du curseur. BF vaut 0	1 µs
Écrire des données à CGI ou DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Ecrit des données dans la DD RAM ou la CG RAM.	40 µs
lire les données	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Lit les données de la DD RAM ou de la CG RAM.	40 µs

Tableau 2.5 jeux de commande d'un afficheur LCD

DD RAM : mémoire d'affichage **,** **CG RAM** : mémoire du générateur de caractères

X : indifférent

I/D=0 : Déplacement vers la gauche, **I/D**=1 : déplacement vers la droite.

S=1 :L'afficheur accompagne le curseur dans son déplacement

S/C=0 : Le curseur se déplace , **S/C**=0 :l'afficheur se déplace

R/L=0 : Vers la gauche , **R/L**=1 : vers la droite

DL=0 Interface 4 bits , **DL**=1: interface 8 bits

N=0: 1 Ligne , **N**=1: 2(ou4)lignes

F=0: Caractères 5*7 , **F**=1:caractère 5*10

BF=0 : Accepte une instruction , **BF**=1 occupé

7. Le codeur rotatif (incrémental)

Le contrôle du déplacement et de la position d'un mobile est un problème couramment rencontré sur un grand nombre de systèmes automatisés. Le codeur rotatif est un capteur de position angulaire. Lié mécaniquement à un arbre qui l'entraîne, son axe fait tourner un disque qui lui est solidaire. Ce disque comporte une succession de parties opaques et transparentes. Une lumière émise par des diodes électro-lumineuses, (DEL ou LED) traverse les fentes de ce disque et crée sur les photodiodes réceptrices un signal analogique.

7.1 Principe de fonctionnement

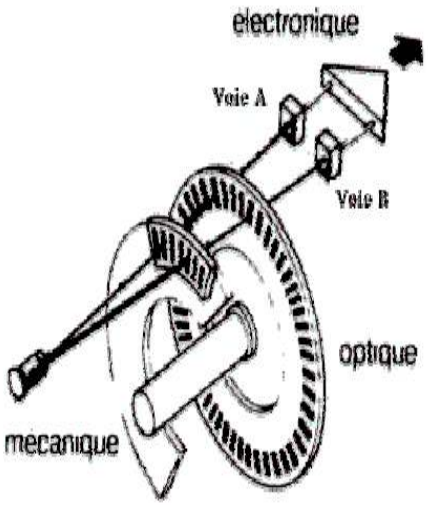

Figure 2.15 Principe de fonctionnement de codeur rotatif

Les codeurs incrémentaux sont destinés à des applications de positionnement et de contrôle de déplacement d'un mobile par comptage et décomptage des impulsions qu'ils délivrent. Le disque d'un codeur incrémental comporte deux types de pistes :

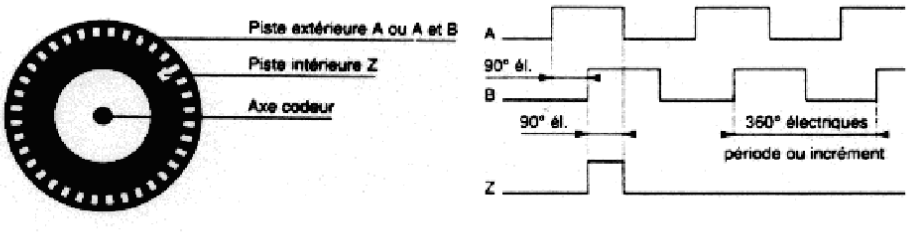

Figure 2.16 Les pistes du codeur

La piste extérieure : (voie A ou voie A et B) est divisée en « n » intervalles d'angles égaux alternativement opaques et transparents, « n » s'appelant la résolution ou nombre de périodes c'est en effet le nombre d'impulsions qui seront délivrées par le

codeur pour un tour complet de son disque.(notre codeur possède de 75 impulsions/tours)

Derrière la piste extérieure sont installées deux photodiodes décalées qui délivrent des signaux carrés A et B en quadrature.

2.3 Les avantages/Inconvénients du codeur rotatif

- ✓ **Les avantages:**

  Le codeur incrémental est de conception simple (son disque ne comporte que deux pistes) donc plus fiable et moins chère.

- ✓ **Les 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.

8. Conclusion

Il existe des instructions simples pour éviter de programmes les registres qui contrôlent les groupes du bouches(B,C et D) et les interruptions externes cette options nous a permis d'éviter de programmer le comptage en testant l'état du port cela nous a fait gagne de temps et bien sur beaucoup d'espace mémoire. Un autre avantage apporté par cette option est que le registre de comptage peut recevoir des impulsions et s'incrémente même si le microcontrôleur exécute d'autre tache

Tous ce qu'on a vu jusqu'à maintenant sur les microcontrôleurs, les LCD et le codeur rotatif va servir à concevoir notre maquette de comptage, d'affichage et de maitre en œuvre les organigrammes adéquates qui nous permettra par la suite de rédiger nos programmes

Chapitre 3 Partie logiciel et réalisation pratique

1. Introduction :

Après avoir présenté la station service et la carte Arduino UNO, nous passons à la partie réalisation, où nous allons aborder la maquette et l'interface graphique qui permet de contrôler et gérer la station Self-service.

2. Etude et réalisation :

Notre projet repose sur trois points essentiels à savoir :

1- La programmation de la carte Arduino UNO

2- La gestion du comptage

3-L'affichage des trois paramètres (nombres de litres, pris de payement et le prix unitaire) sur les afficheurs.

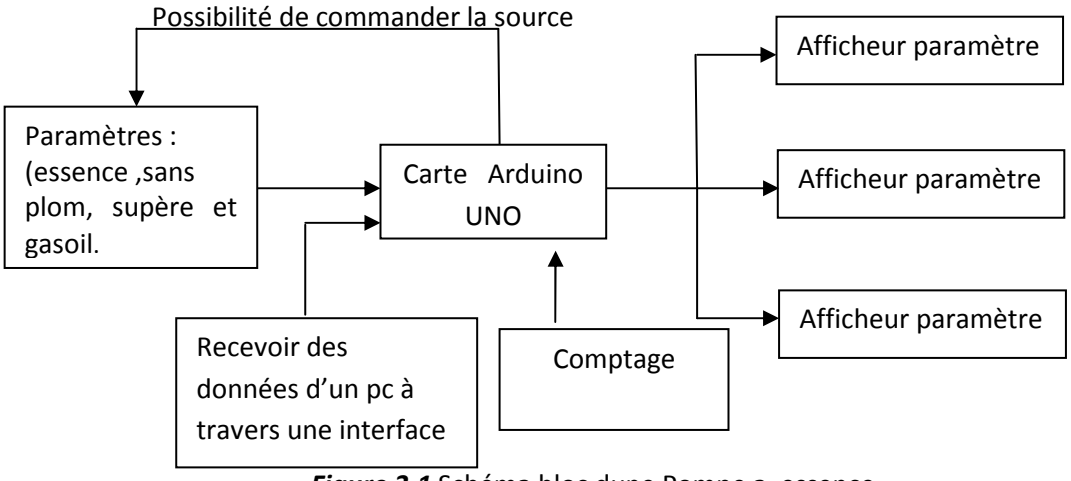

Figure 3.1 Schéma bloc dune Pompe a essence

2.1 Application :

Il est à signaler que notre étude peut avoir plusieurs applications tell que :

- ✓ Le calcule de la vitesse du vent et l'affichage de ce paramètre.
- ✓ L'extraction des paramètres météorologique.
- ✓ La gestion d'écoulement d'un liquide quelconque.

Pour notre étude on a opté pour une station self-service, il est demander donc :

- ➢ Compter le nombre de litres.
- ➢ Donner l'équivalence en dinars
- ➢ Faire varier le prix unitaire à notre choix, et affiché les trois paramètres sur les afficheurs
- ➢ Sauvegarder ces paramètres dans une base de données

La figure (figure 3.2) montre le schéma général de l'application :

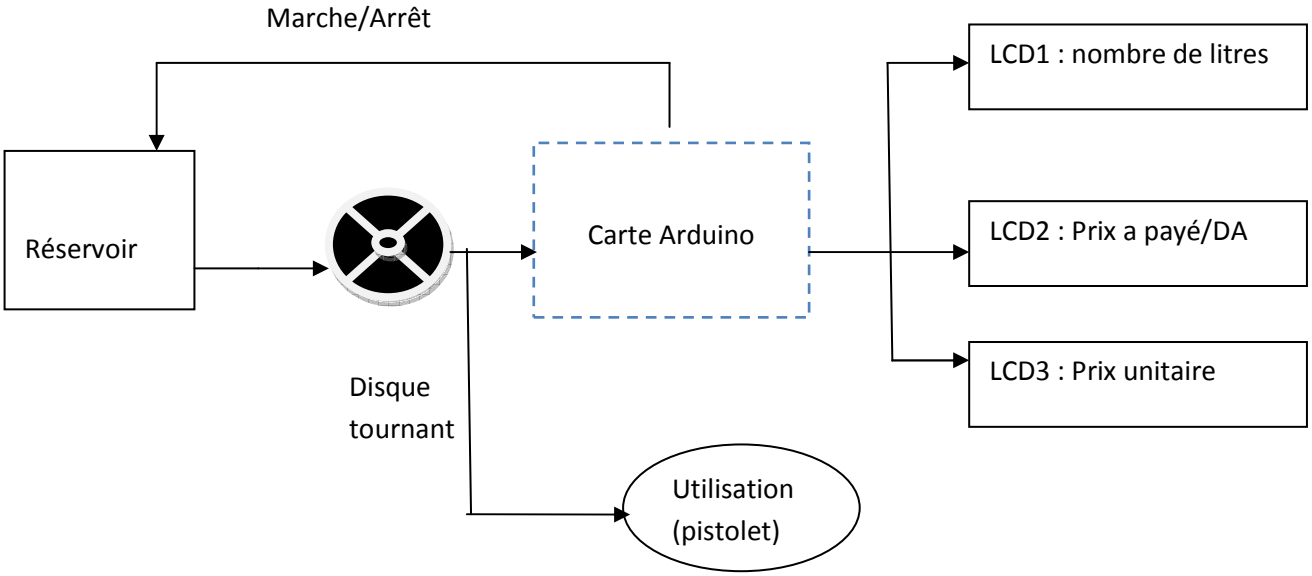

Figure 3.2 Schéma bloc de l'application.

2.2 Schéma synoptique :

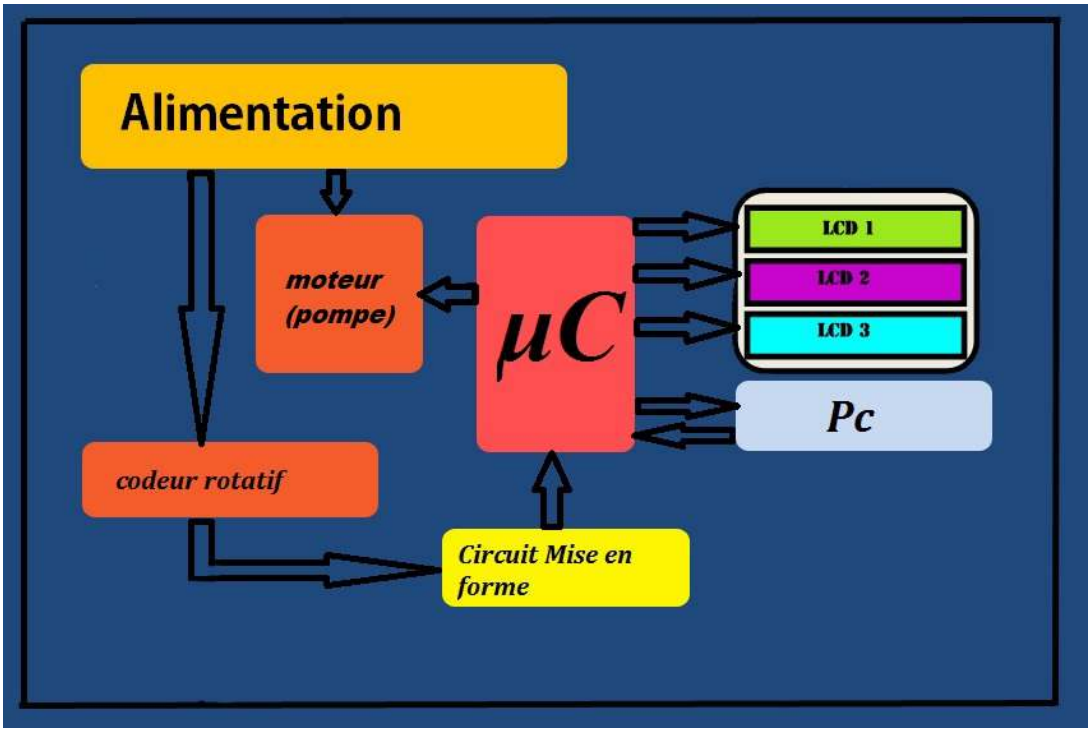

Figure 3.3 *schéma synoptique.*

3. Développement de chaque bloc :

L'interface graphique contient quatre distributeurs et chaque fenêtre correspondante à un produit fourni (SP-Sup ou Gz), pour faire fonctionner l'ensemble de distributeurs il faut relier douze cartes Arduino avec leurs accessoires, ce qui rend notre projet non seulement encombré mais couteux, pour cela, on a opté pour la réalisation d'une seule carte et faire fonctionner un seul distributeur.

3.1 Alimentation :

Le fonctionnement du moteur nécessite une alimentation stabilisée de 12V-0.18A, (figure 3.4).

- **Présentation**

C'est une alimentation construite autour d'un régulateur de tension intégré de type 7812, et l'intensité maximale délivrée est de 1.5 A.

- **Schéma**

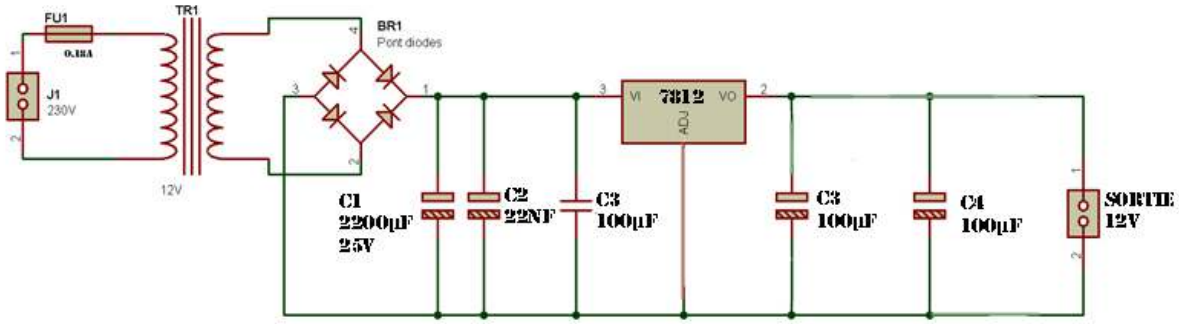

Figure 3.4 *Schéma détaillé de l'alimentation.*

- **Puissance**

Le régulateur fourni un courant maximal de 1.5A. Pour une bonne dissipation de la chaleur dégagée par le régulateur, il est souhaitable de coller un dissipateur qui doit être en contact intime (thermique) avec le régulateur pour jouer efficacement son rôle.

- **Transformation de tension**

La transformation de la tension est assurée par un transformateur 220/12V qui permet de convertir la tension secteur 220VAC à une tension de 12 VAC.

- **Redressement**

Le redressement de la tension alternative, délivrée par le secondaire du transformateur, est assurée par le pont de diodes BR1, qui est capable de supporter un courant direct et permanent de 5 A au minimum.

Filtrage principal

Le rôle du condensateur de filtrage C1 et C2, est de réduire l'ondulation d'une tension préalablement redressée. Il permet, en quelque sorte, de "lisser" la tension

ondulée. Les condensateurs de filtrage sont montés en parallèle avec la sortie du pont de diodes

- **Régulation**

Pour fonctionner correctement l'entrée du régulateur doit être alimentée par une tension non seulement redressée mais aussi filtrée grâce au montage avec condensateurs que nous venons de présenter. Donc le régulateur intégré 7812 utilisé sert pour fournir une tension stabilisée de 12 V à sa sortie pour alimenter le moteur à courant continu.

- **Circuit imprimé**

La figure ci-dessous montre le circuit imprimé du circuit d'alimentation :

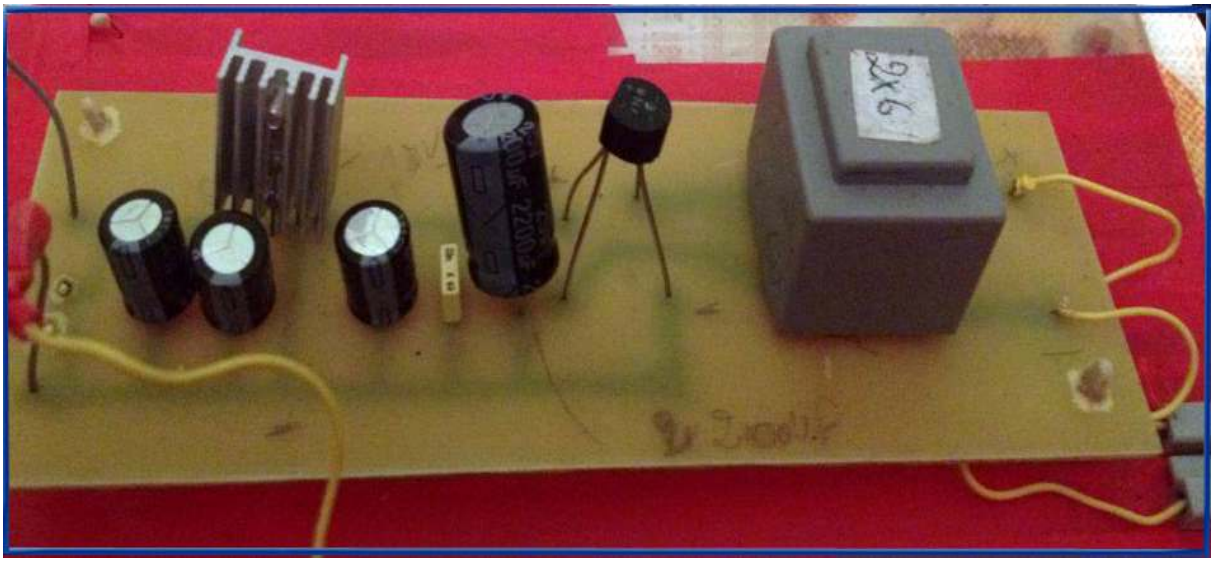

Figure 3.5 Alimentation 12V/0.18A.

3.2 Circuit d'interface (isolation) :

3.2.1 Principe d'isolation

Pour isoler électriquement le circuit de commande de celui de puissance, on a utilisé deux sources d'alimentation ainsi que deux masses différentes.

Un avantage évident de l'utilisation de deux masses différentes est l'isolation contre le bruit électrique causé par le retour de masses, en effet une différence de potentiel entre deux points de masse cause d'énormes problèmes dans le milieu industriel.

3.2.2 Isolation par opto-coupleur

Un opto-coupleur est considéré comme un élément de transfert de signal dont l'entrée et la sortie sont électriquement isolées l'une par rapport à l'autre par un couplage optique. **La figure** ci-dessous illustre le principe de fonctionnement d'un opto-coupleur.

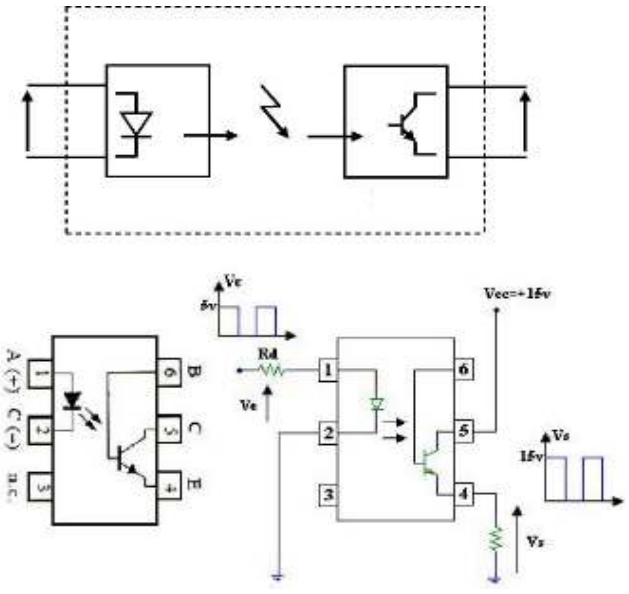

Figure 3.6 Schéma de l'opto-coupleur 4N25.

3.2.3 Contrôle du moteur par opto-coupleur :

On peut contrôler la vitesse d'un moteur par une interface (**figure 3.7**) composé par un transistor et Opto-coupleur, cette carte est utiliser pour protéger le notre système de commande. Dans notre cas, la vitesse du moteur est contrôlé par la PWM générée par l'ARDUINO.

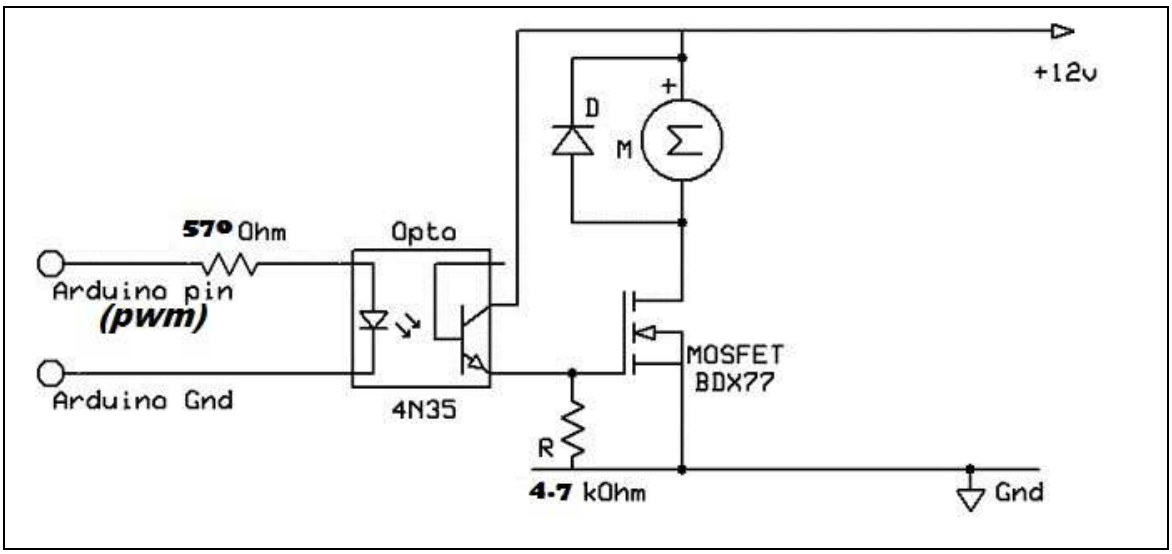

Figure 3.7 *Schéma de la carte d'isolation.*

L'opto-coupleur permet de redresser les fronts pour un signal de la PWM faible ou proche de 100%. L'opto-coupleur agisse comme un amplificateur afin de piloter le transistor avec un signal PWM qui varie entre 0 et 100%, pour le contrôle total de la vitesse

- **Circuit imprimé :**

Le radiateur est collé sur le transistor pour une meilleur dissipation de chaleur

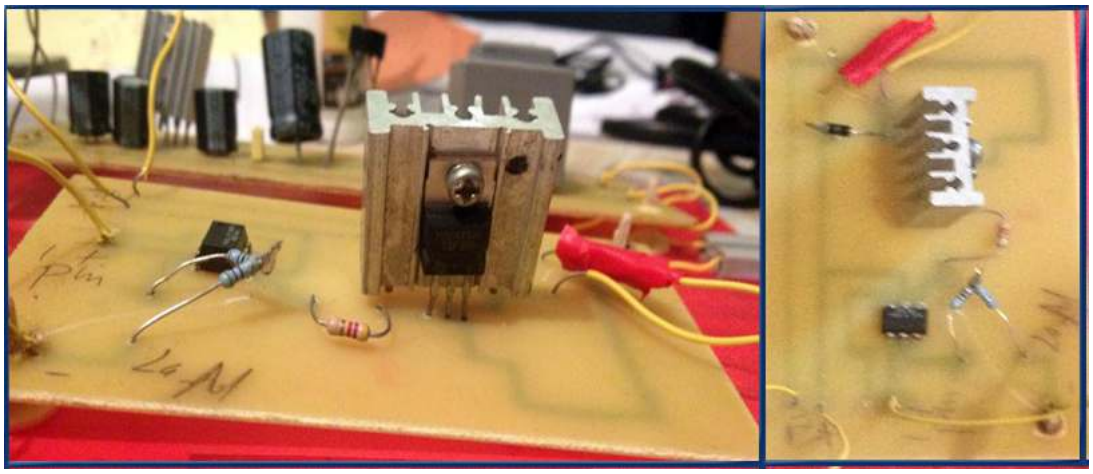

Figure3.8 La carte d'isolation.

3.3 Compteur d'impulsion :

3 .3.1 Le convertisseur (impulsions /litres) :

Une partie de ce travail consiste à compter le nombre de litres propulsés par le moteur (pompe), pour cela, le moteur choisi pour cette opération est doté d'un disque plain d'encoches, où un tour complet donne un litre.

Pour notre cas, le disque contient 75 orifices, or une rotation complète équivalente à un litre.

75 impulsions ⟶ 1 litres.

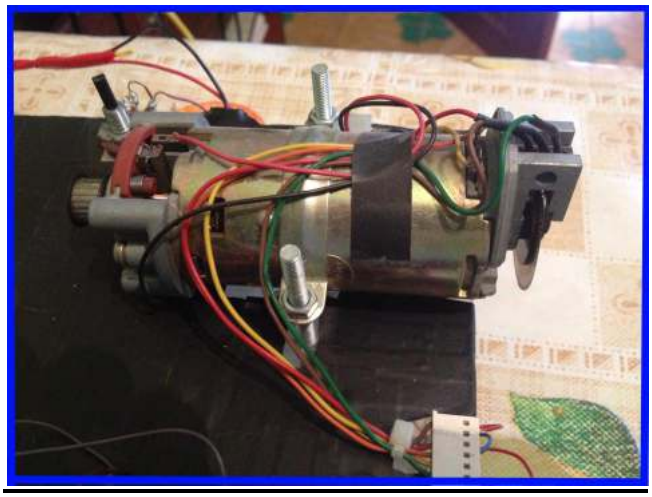

Figure 3.9 codeur rotatif**.**

3.3.2 Circuit mise en forme

Le signal arrive à la broche **2** de la carte ARDUINO doit avoir une forme carrée pour que le microcontrôleur puisse le traité, dans ce cas se signal doit varier entre 0 et 5v. Le codeur rotatif contient un circuit imprimé qui a le rôle d'un circuit de mise en forme. Le circuit est une interface électronique (qui est inclue dans le codeur) amplifie ce signal puis le convertit en signal carré qui est alors transmis à un système de traitement .

Figure 3.10 format du maquette finale

4. Partie logiciel :

Pour faire fonctionner notre maquette (Hardware), on a utilisé l'IDE 1.0.5(Arduino) pour contrôler nos cartes et visuel Basic qui sert a créé une interface graphique pour piloter la station self-service.

4.1 Organigramme général du programme :

L'organigramme général du programme est constitué de huit parties :

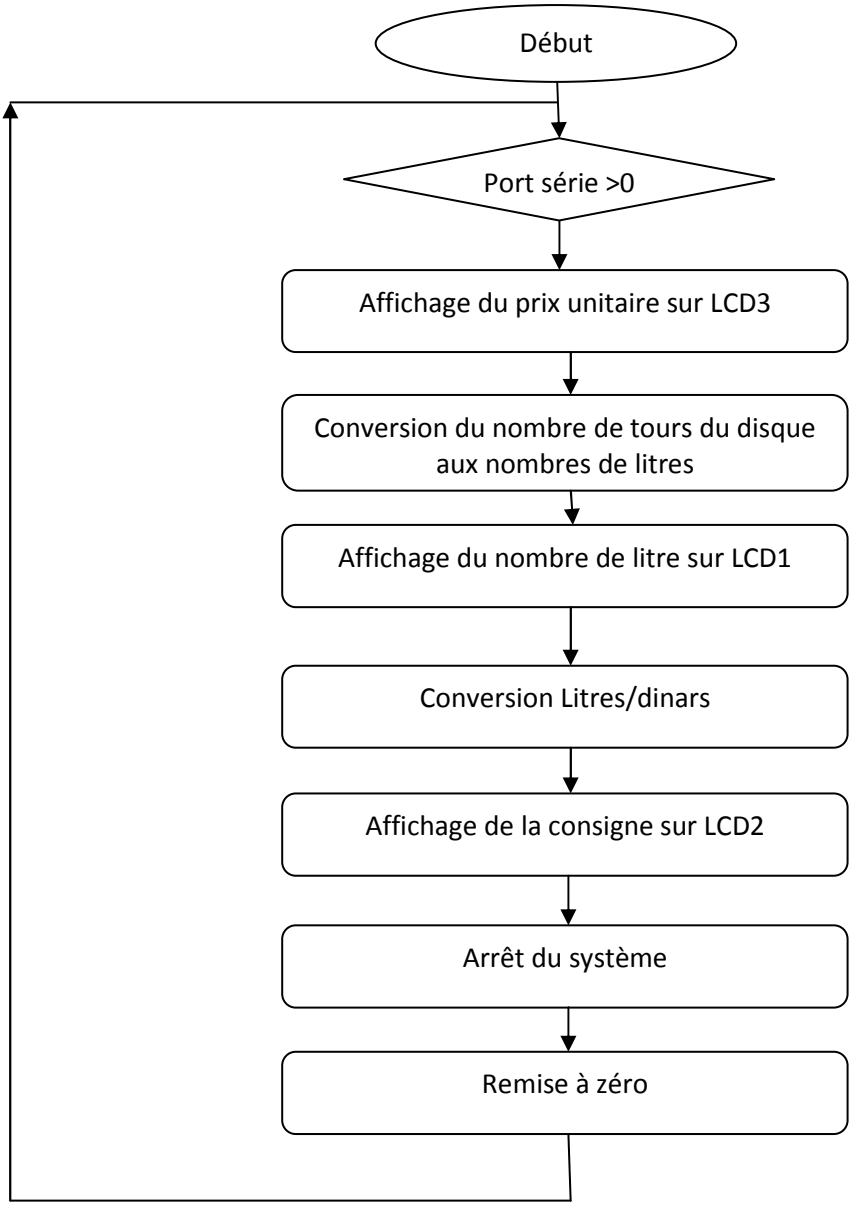

Figure 3.11 Organigramme général.

4.2 Interface graphique :

4.2.1 Visuel basic :

Visual Basic (VB) est un langage de programmation événementielle de troisième génération ainsi qu'un environnement de développement intégré, créé par Microsoft pour son modèle de programmation COM. Visual Basic est directement dérivé du BASIC et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données en utilisant les technologies DAO, ADO et RDO, ainsi que la création de contrôles ou objets Active X.

Visual Basic est un des langages les plus utilisés pour l'écriture d'applications commerciales. Il a également été très utilisé dans le monde de l'ingénierie et de la recherche appliquée en raison de sa capacité à permettre des développements très rapides et très efficients permettant ainsi aux scientifiques de se consacrer davantage à l'algorithmique et moins aux aspects formels du codage.

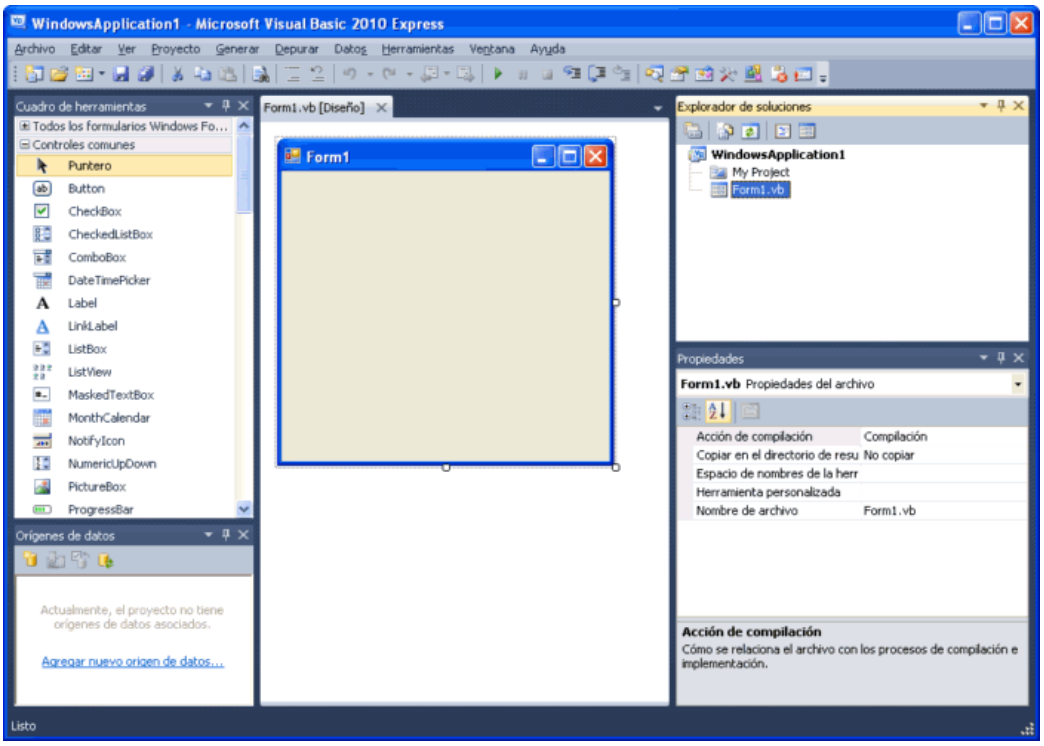

Figure 3.12 Visual basic 1

4.2.2 Description de l'interface :

Notre système est commandé par une interface développée par le visuel basic.net. C'est une interface simple et pratique qui permet de commander les distributeurs à distance. Elle est composée de :

* Trois fenêtres (essence sans plomb, super et gasoil), où chacune est dotée de quatre distributeurs. Le prix unitaire est affiché dès le démarrage du système.

* Un bouton de remise à **zéro**, qui sert à la réinitialisation des afficheurs dès le démarrage.

* Un bouton qui sert à l'envoi de la consigne vers l'afficheur via le port série. Cette consigne est sauvegardée au même temps dans une base de données.

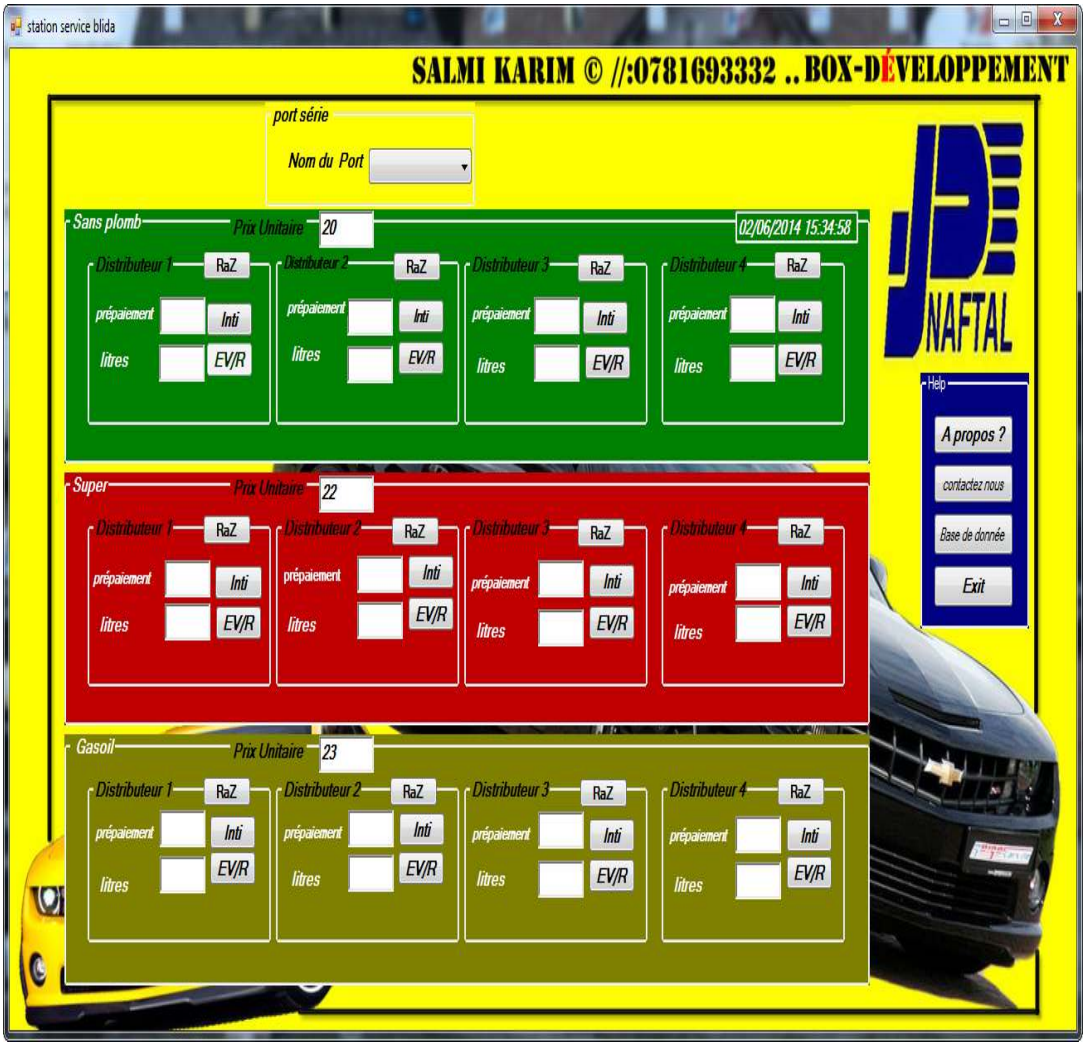

Figure 3.13 L'interface graphique

4.2.3 Organigramme de l'interface :

L'organigramme de l'interface est constitué de sept parties :

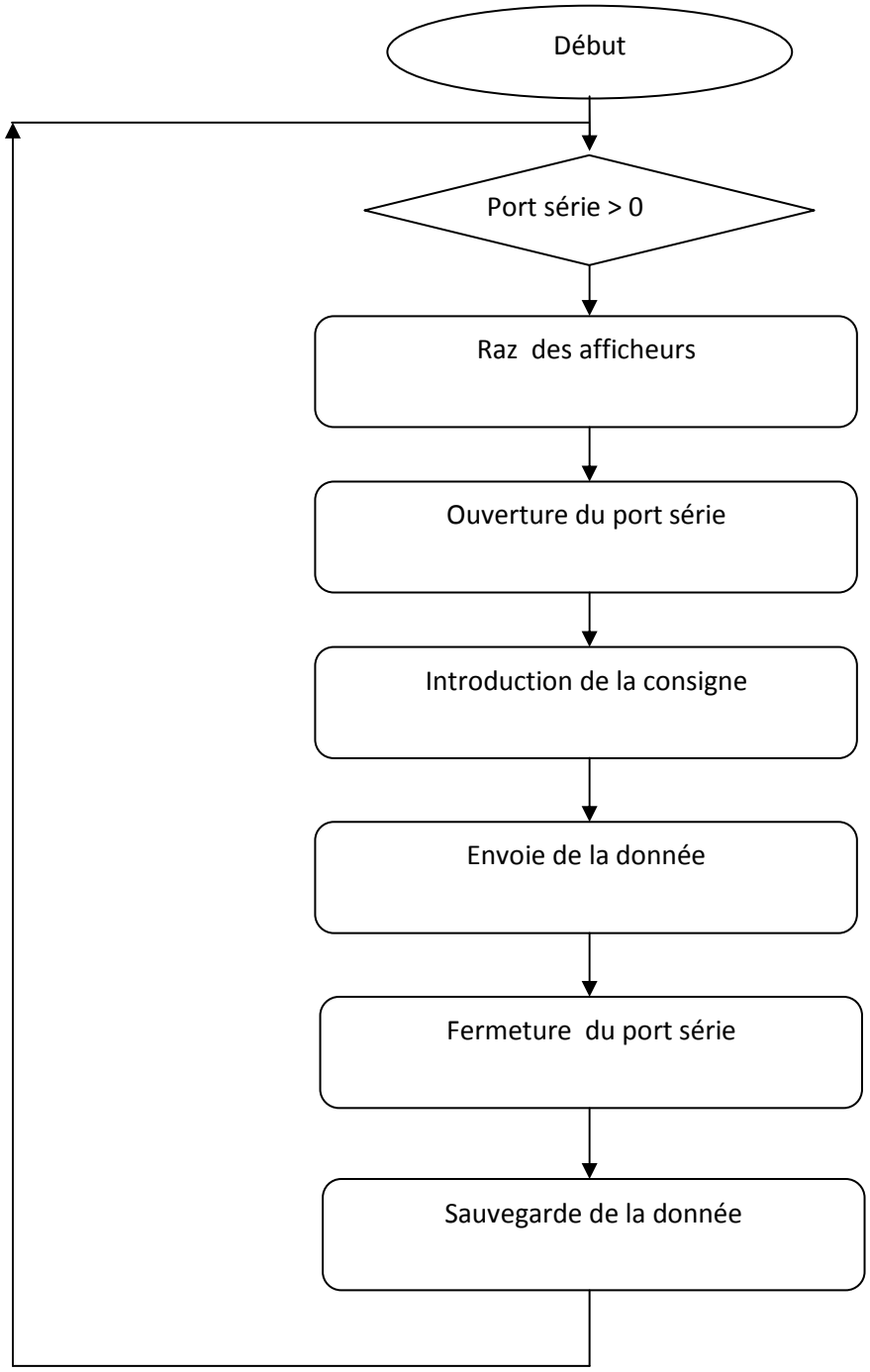

Figure 3.14 L'organigramme de l'interface graphique.

4.2.4 Fonctionnement de l'interface graphique :

Le schéma ci-dessous montre le fonctionnement général de l'interface graphique

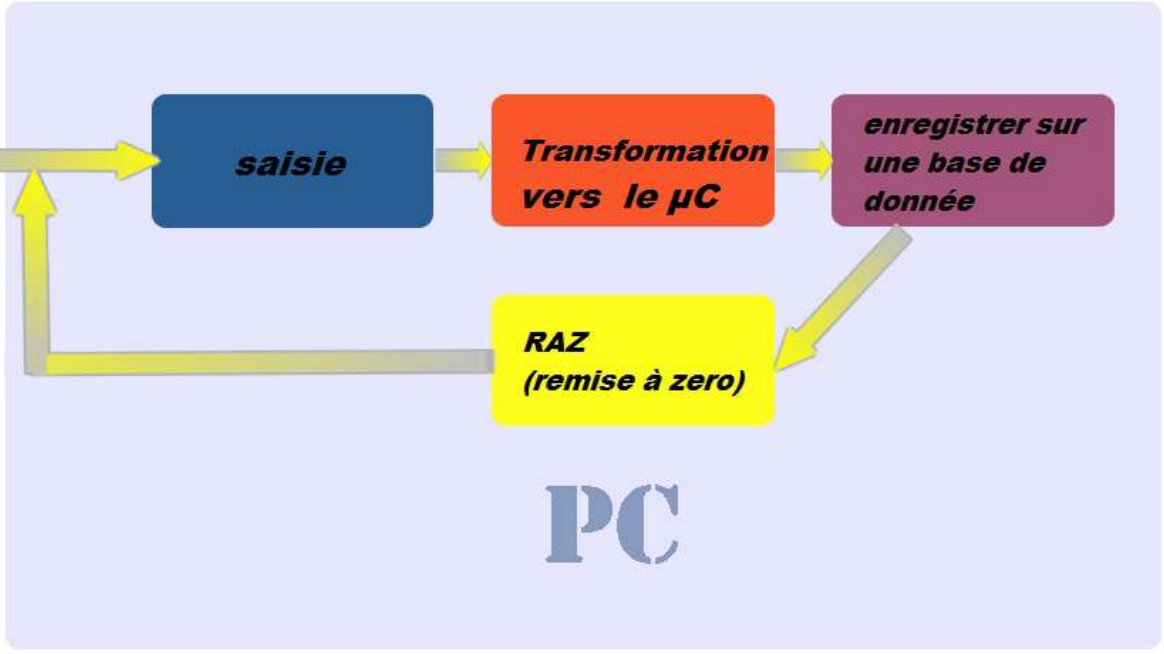

Figure 3.15 schéma de fonctionnement de l'interface.

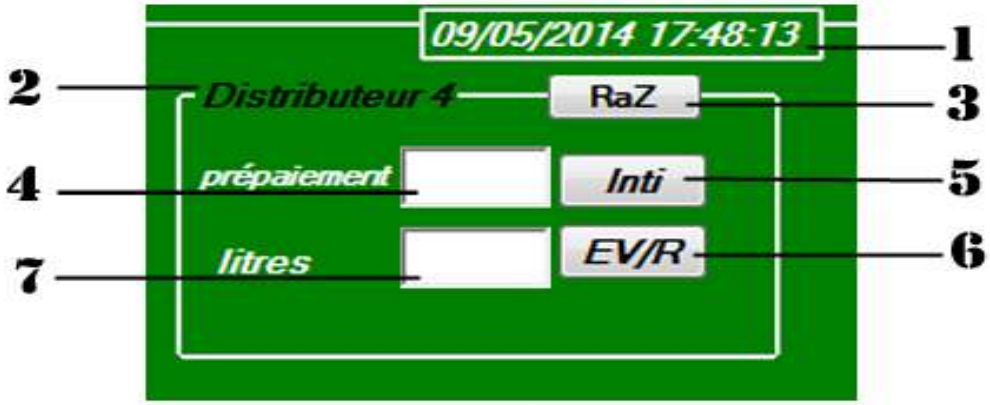

Figure 3.16 Fenêtre d'un seul distributeur.

1 :l'horloge

2 : nom du distributeur (la zone)

3 : bouton de remise à zéro pour les afficheurs de la fenêtre

4 : box du prépaiement

5 : bouton pour l'initialisation du port série et l'envoi du prix unitaire

6 : bouton pour l'envoi de la consigne et la sauvegarde des données.

7 : box pour l'affichage du nombre de litres

4.3 Le logiciel l'IDE Arduino

4.3.1 L'IDE

Le logiciel de programmation des modules Arduino est une application Java, libre et multi-plateforme, servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (RS-232, Bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino, et de compiler et uploader les programmes via l'interface en ligne de commande. Le langage de programmation utilisé est le C++, compilé avec avrg++ , et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties.

Figure 3.17 le logiciel de programmation

4.3.2 Organigramme de l'ARDUINO

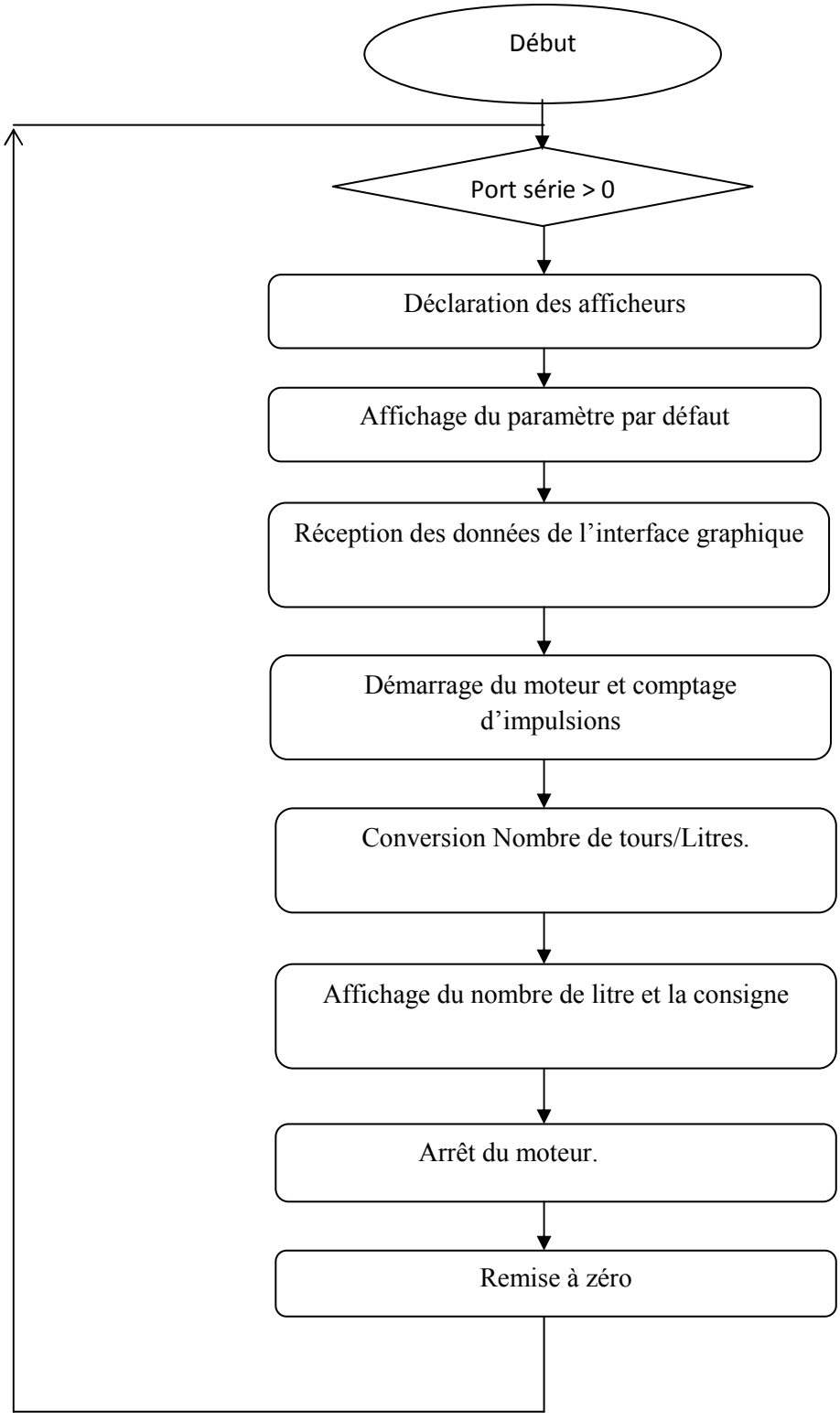

Figure 3.18 Organigramme du programme (Arduino)

4.3.3 Programme (Arduino):

```
// --- Déclaration des variables globales ---

volatile float comptageImpulsion=0; // variable accessible dans la routine interruption externe 0

int moteur = 4;

int led = 5;

int ledPin = 3;

float a; // a = prix unitaire

float b; // b= la consigne

char junk = ' ';

float total=0;

float prixUnitaire=20;

float nombreDelitres=0;

float arr=total/prixUnitaire;

#include <LiquidCrystal.h>

// initialize the library with the number

LiquidCrystal lcd1(13, 12, 9, 8, 7, 6);

LiquidCrystal lcd2(13, 11, 9, 8, 7, 6);

LiquidCrystal lcd3(13, 10, 9, 8, 7, 6);

void setup()   { // debut de la fonction setup()

pinMode(moteur, OUTPUT);

pinMode(led, OUTPUT);
```

```
// --- ici instructions à exécuter 1 seule fois au démarrage du programme ---

Serial.begin(115200);

lcd1.begin(16, 2);

lcd2.begin(16,2);

lcd3.begin(16,2);

 lcd3.print( prixUnitaire);

lcd1.clear();

lcd1.setCursor(0,1);

lcd1.print(nombreDelitres);

lcd2.clear();

lcd2.setCursor(0,1);

lcd2.print("0.00");

while (Serial.available() == 0) ;  // Attendez ici jusqu'à ce que le buffer d'entrée a un caractère

{

//Side 1

a = Serial.parseFloat();       // nouvelle commande

lcd3.clear();

lcd3.setCursor(0,1);

lcd3.print(a);//// prix unitaire

while (Serial.available() > 0)  // parseFloat () peut laisser des caractères non numériques
```

```
{ junk = Serial.read() ; }      // effacer la mémoire du buffer

}

while (Serial.available() == 0) ;

{

//Side 2

b = Serial.parseFloat();// la consigne

while (Serial.available() > 0)

{ junk = Serial.read() ; }

}

digitalWrite(moteur, HIGH);

digitalWrite(led, HIGH);

analogWrite(ledPin, 255);

} // fin de la fonction setup()

// FONCTION LOOP = Boucle sans fin = coeur du programme

// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino est sous tension

void loop(){

attachInterrupt(0, gestionINT0, RISING);

nombreDelitres=comptageImpulsion/75;// conversion tours/litres

lcd1.setCursor(0, 1);

lcd1.print(nombreDelitres);

lcd2.begin(16, 2);
```

```
lcd2.print(nombreDelitres*a);

if((nombreDelitres*a)>=0.8*b)// condition pour baisser la vitesse du moteur

{ analogWrite(ledPin, 175);// 50% de pwm

}

if (nombreDelitres*a >= b)//condition pour l'arrêt

{digitalWrite(moteur, LOW);

digitalWrite(led, LOW);

delay(10000);

software_Reset() ;// Redémarrer le programme du début, mais ne réinitialise pas les périphériques et les registres

}} // fin de la fonction loop() - le programme recommence au début de la fonction loop sans fin

// ---------- fonction de gestion l'interruption externe n°0 (broche 2) ----------

// cette fonction est appelée à chaque fois que l'interruption a lieu selon le mode configuré (LOW, CHANGE, RISING, FALLING)

void gestionINT0() {// la fonction appelée par l'interruption externe n°0

comptageImpulsion=comptageImpulsion+1;// Incrémente la variable de comptage

}

void software_Reset()//la fonction appelée par la commande  software_Reset()

{

asm volatile ("  jmp 0");}
```

4.4 Base de données :

Une base de données est un conteneur informatique permettant de stocker l'intégralité des informations en rapport avec une activité. Une base de données permet de stocker et de retrouver un ensemble d'informations de plusieurs natures ainsi que les liens qui existent entre les différentes informations.

Une base de données est la pièce centrale des dispositifs informatiques qui servent à la collecte, le stockage, le travail et l'utilisation d'informations. Le dispositif comporte un système de gestion de base de données (abr. SGBD) : un logiciel moteur qui manipule la base de données et dirige l'accès à son contenu. De tels dispositifs comportent également des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations.

La manipulation de données est une des utilisations les plus courantes des ordinateurs. Les bases de données sont fréquentes dans les secteurs de la finance, des assurances, des écoles, de l'administration publique et les médias.

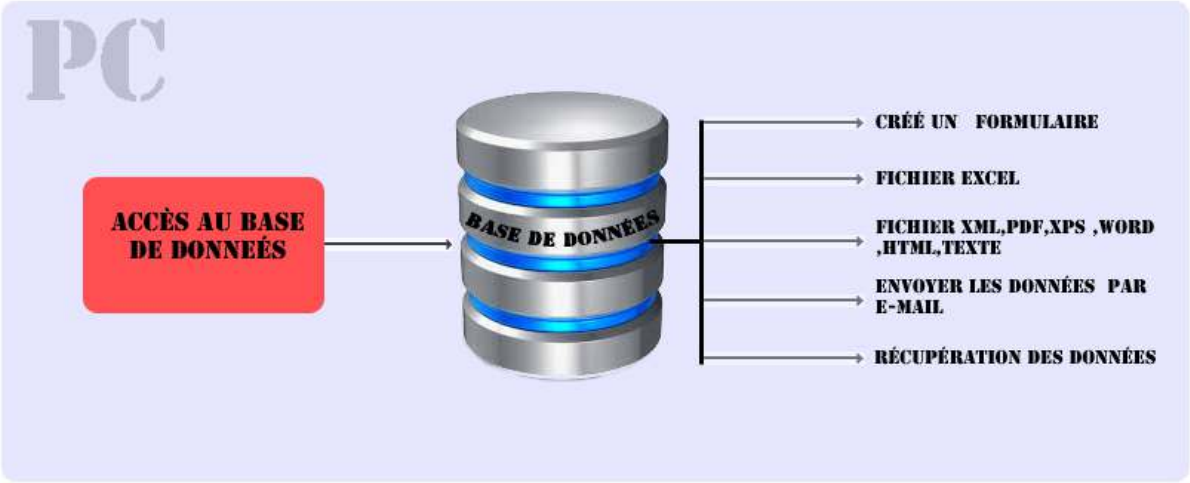

Figure 3.19 l'utilisation de la base de données.

Il y une multitude de logiciels pour la création des bases de données, on cite à titre d'exemple : MySQL, DB2, interbase, Access…..etc. dans notre cas, la base de donnée du système conçu par « ACCESS » est relié directement à l'interface graphique, pour enregistrer toutes les opérations faites par le chef pompiste.

4.4.1 Les Avantages de L'Access :

L'Access offre des fonctionnalités de calcul et de calcul étendues pour la visualisation et la manipulation des données tabulaires. Ils peuvent également être utiles pour le stockage de listes d'informations, telles que des listes de publipostage ou des inventaires. Toutefois, à mesure que vos exigences de listes croissent en complexité, il peut devenir difficile de conserver vos données dans un tableur. Un programme de base de données relationnelle, tel que Microsoft Office Access 2007, convient mieux à la création et à l'utilisation de listes complexes et offre des outils plus puissants pour la saisie, l'organisation, la maintenance et la récupération de vos données.

4.4.2 Description de la base de données

La base de données du système est protégée par un mot de passe de niveau 2, ou uniquement le gérant de la station qui peut accéder.

Cette base de données est composée de 03 tableaux (sans-plomb, super, gasoil), et chaque tableau contient 5 colonnes (numéro des lignes, prix unitaire, prépaiement, nombres de litres et l'heure/date) et le nombre de ligne dépend de la quantité vendu.

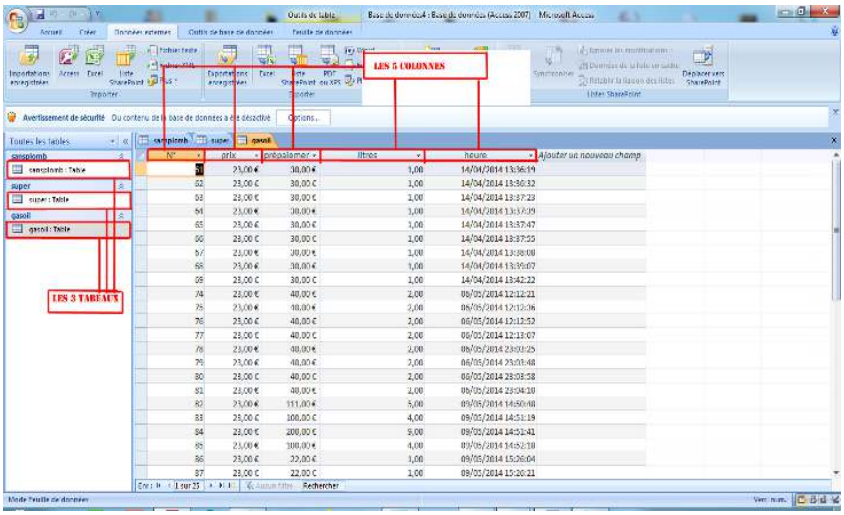

Figure 3.20 Base de données à base de l'Access.

5. Conclusion :

Le projet se trouve prêt à l'utilisation dans les stations Self- services, avec des adaptations adéquates. La gestion de ce projet est assurée par une interface conçu par le visuel basic qui nous a beaucoup aider malgré les difficultés rencontrer. En premier lieu, on a tenté de construire une interface avec le Matlab, chose qui n'était pas facile, où on a trouvé des difficultés sur la synchronisation de l'envoi des données. Pour cela on a opté pour le visuel basic.

Conclusion générale

Le travail présenté dans ce mémoire a été effectué avec succès, le but de ce projet consistait a automatisé les anciens systèmes des poste de stations service. Il peut être directement utilisé dans les stations self-services bien sur après quelques modifications et retouches selon les besoins.

Comme toute réalisation ne peut être à l'abri de difficulté, l'interface qui gère l'ensemble des fonctions de ce système avait pris la majorité du temps, et surtout la connexion entre PC et la carte Arduino. Les différents algorithmes développés dans ce travail ont été implémentés en langage IDE, ce qui le rend intéressant pour d'autres applications.

Au cours de la préparation de ce projet, nous avons pu acquérir beaucoup de connaissances, surtout dans le domaine de la programmation (Visuel Basic et IDE).

Et dans tout ça, nous n'avons pas manqués de renforcer les notions théoriques avec des implémentations concrètes dans des langages comme l'IDE et Visuel Basic.

Enfin, on peut dire que les résultats obtenus dans ce travail sont satisfaisantes, et le sujet reste ouvert pour toutes améliorations souhaitées.

En perspective a ce travail, on prévoit l'assemblage de plusieurs cartes Arduino pour rendre fonctionnel la station self-service.la liaison entre ces cartes sera assurée par le bus I2C. Ce qui nous proposons aux étudiants de l'année prochaine.

Annexes

1. L’afficheur LCD

1.1 La mémoire

L'afficheur possède deux types de mémoire, la DD RAM et la CG RAM. La DD RAM est la mémoire d'affichage et la CG RAM est la mémoire du générateur de caractères.

1.1.1 La mémoire d'affichage (DD RAM)

La DD RAM est la mémoire qui stocke les caractères actuellement affiché à l'écran. Pour un afficheur de 2 lignes de 16 caractères, les adresses sont définies de la façon suivante :

Ligne	Visible	Invisible
Haut	00H.....0FH	10H.....27H
Bas	40H.....4FH	50H.....67H

L'adresse 00H correspond à la ligne du haut à gauche, 0FH à droite. L'adresse 40H correspond à la ligne du bas à gauche, 4FH à droite. La zone invisible correspond à la mémoire de l'afficheur (48 caractères). Lorsqu'un caractère est inscrit à l'adresse 27H, le caractère suivant apparaît à la ligne suivante.

1.1.2 la mémoire du générateur de caractéres (CG RAM)

Le générateur de caractère est quelque chose de très utile. Il permet la création d'un maximum de 8 caractères ou symboles 5x7. Une fois les nouveaux caractères chargés en mémoire, il est possible d'y accéder comme s'il s'agissait de caractères classiques stockés en ROM. La CG RAM utilise des mots de 8 bits de large, mais seul les 5 bits de poids faible apparaissent sur le LCD. Ainsi D4 représente le point le plus à gauche et D0 le point le plus à droite. Par exemple, charger un octet de la CG RAM à 1Fh fait apparaître tous les points de cette rangée ; charger un octet à 00h éteint tous ces points. Les 8 lignes d'un caractère doivent être chargées dans la CG RAM.

1.2 Commande d'un afficheur LCD

Deux modes de fonctionnement de l'afficheur sont disponibles, le mode 4 bits et le mode 8 bits, modes que l'on choisira à l'initialisation de l'afficheur.

Mode 8 bits :

Dans ce mode 8 bits, les données sont envoyées à l'afficheur sur les broches **D0** à **D7**. On place la ligne **RS** à 0 ou à 1 selon que l'on désire transmettre une commande ou une donnée. Il faut aussi placer la ligne **R/W** à 0 pour indiquer à l'afficheur que l'on désire effectuer une écriture. Il reste à envoyer une impulsion d'au moins 450 ns sur l'entrée **E**, pour indiquer que des données valides sont présentes sur les broches **D0** à **D7**. L'afficheur lira la donnée sur le front descendant de cette entrée.
Si on désire au contraire effectuer une lecture, la procédure est identique, mais on place cette fois la ligne **R/W** à 1 pour demander une lecture. Les données seront valides sur les lignes D0 à D7 lors de l'état haut de la ligne **E**.

Mode 4 bits :

Il peut, dans certains cas, être nécessaire de diminuer le nombre de fils utilisés pour commander l'afficheur, comme, par exemple lorsqu'on dispose de très peu de broches d'entrées sorties disponibles sur un microcontrôleur. Dans ce cas, on peut utiliser le mode quatre bits de l'afficheur LCD. Dans ce mode, seuls les 4 bits de poids fort (**D4** à **D7**) de l'afficheur sont utilisées pour transmettre les données et les lire. Les 4 bits de poids faible (**D0** à **D3**) sont alors connectés à la masse. On a donc besoin, hors alimentation de sept fils pour commander l'afficheur. Les données sont alors écrites ou lues en envoyant séquentiellement les quatres bits de poids fort suivi des quatres bits de poids faible. Une impulsion positive d'au moins 450 ns doit être envoyée sur la ligne E pour valider chaque demi-octet.

2. programme Visual basic

```
Imports System
Imports System.Threading
Imports System.IO.Ports
Imports System.ComponentModel
Imports System.Data.OleDb
Public Class Form2
Public connstring As String = "Provider=microsoft.ace.oledb.12.0; data source=C:\Users\karim\Desktop\Base de données4.accdb"
Public conn As New OleDbConnection
'------------------------------------------------
Dim myPort As Array
Delegate Sub SetTextCallback(ByVal [text] As String) 'Added to prevent threading errors during receiveing of data
'------------------------------------------------
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
myPort = IO.Ports.SerialPort.GetPortNames()
ComboBox1.Items.AddRange(myPort)
Button3.Enabled = False
conn.ConnectionString = connstring
If conn.State = ConnectionState.Closed Then
conn.Open()
Else
MsgBox("close")
End If
End Sub
'------------------------------------------------
Private Sub ComboBox1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
'------------------------------------------------
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

End Sub
'------------------------------------------------
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

End Sub
Private Sub SerialPort1_DataReceived(ByVal sender As System.Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs)
End Sub
```

```
Private Sub RichTextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
Private Sub ComboBox2_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
Private Sub RichTextBox2_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
Private Sub GroupBox1_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs)
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
Private Sub GroupBox7_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
TextBox1.Text = Now
End Sub
Private Sub PictureBox1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
Private Sub Button37_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button37.Click
Form3.ShowDialog()
End Sub
Private Sub Button38_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button38.Click
Process.Start(“ e-mail")
End Sub
Private Sub Button39_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button39.Click
```

```
Form4.ShowDialog()
End Sub
Private Sub Button40_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button40.Click
Me.Close()
End Sub
Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged
End Sub
Private Sub Button2_Click_2(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
SerialPort1.PortName = ComboBox1.Text
SerialPort1.BaudRate = 115200
Button3.Enabled = True
SerialPort1.Open()
SerialPort1.Write(RichTextBox1.Text & vbCr)
End Sub
Private Sub Button3_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
SerialPort1.Write(RichTextBox2.Text & vbCr)
SerialPort1.Close()
Button3.Enabled = False
MsgBox("l'envoi sur port série est terminé avec succès", MsgBoxStyle.Information, "port série")
Dim num1 As String
Dim num2 As String
Dim num3 As String
num1 = RichTextBox1.Text
num2 = RichTextBox2.Text
num3 = num2 / num1
RichTextBox3.Text = num3
```

```
If RichTextBox2.Text = "" Then
MsgBox("SVP entrez une valeur de prépaiement !", MsgBoxStyle.Critical, "Error")
End If
Try
Dim SqlQuery As String = "INSERT INTO sansplomb (prix,prépaiement,litres,heure) VALUES ('" & RichTextBox1.Text & "','" & RichTextBox2.Text & "','" & RichTextBox3.Text & "','" & TextBox1.Text & "')"
Dim SqlCommand As New OleDbCommand
With SqlCommand
.CommandText = SqlQuery
.Connection = conn
.ExecuteNonQuery()
End With
MsgBox("le sauvegarde sur les bases de données est terminé avec succés ", MsgBoxStyle.Information, "Base de données")
Catch ex As Exception
MsgBox(ex.ToString)
End Try
End Sub
Private Sub Button4_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
Private Sub Button1_Click_2(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
SerialPort1.PortName = ComboBox1.Text
SerialPort1.BaudRate = 115200
Button6.Enabled = True
SerialPort1.Open()
SerialPort1.Write(RichTextBox1.Text & vbCr)
End Sub
```

```
Private Sub Button6_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
SerialPort1.Write(RichTextBox6.Text & vbCr)
SerialPort1.Close()
Button6.Enabled = False
MsgBox("l'envoi sur port série est terminé avec succès", MsgBoxStyle.Information, "port série")
Dim num4 As String
Dim num5 As String
Dim num6 As String
num4 = RichTextBox1.Text
num5 = RichTextBox6.Text
num6 = num5 / num4
RichTextBox5.Text = num6
If RichTextBox6.Text = "" Then
MsgBox("SVP entrez une valeur de prépaiement !", MsgBoxStyle.Critical, "Error")
End If
Try
Dim SqlQuery As String = "INSERT INTO sansplomb (prix,prépaiement,litres,heure)
Dim SqlCommand As New OleDbCommand
With SqlCommand
.CommandText = SqlQuery
.Connection = conn
.ExecuteNonQuery()
End With
MsgBox("le sauvegarde sur les bases de données est terminé avec succés ", MsgBoxStyle.Information, "Base de données")
Catch ex As Exception
MsgBox(ex.ToString)
End Try
End Sub
```

```
Private Sub Button5_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
End Sub
Priva
With SqlCommand
Private Sub GroupBox1_Enter_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles GroupBox1.Enter
= ""
End Sub
End Class
Public Class Form4
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
If TextBox1.Text = "karim" And TextBox2.Text = "*********" Then
TextBox1.Enabled = False
TextBox2.Enabled = False
Timer1.Start()
If TextBox1.Text = "" Then
MsgBox("Pas de nom d'utilisateur", MsgBoxStyle.Critical, "Error")
MsgBox("Nom d'utilisateur invalide et / ou mot de passe !", MsgBoxStyle.Critical, "Error")
End If
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
ProgressBar1.Increment(35)
If ProgressBar1.Value = ProgressBar1.Maximum Then
Timer1.Stop()
Try
Shell("C:\Users\karim\Desktop\Base de données4.accdb")
Catch ex As Exception
Process.Start("C:\Users\karim\Desktop\Base de données4.accdb")
```

Bibliographie

[1] Schlumberger Industries: 'système de stations-services Equipements et services', Schlumberger , 1991.

[2] http://arduino.cc/

[3] http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-vb-net