REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 8 Mai 1945 – Guelma
Faculté des Sciences et de la Technologie
Département de Génie Electrotechnique et Automatique

964

Réf:...964.../2018

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER Académique**

Domaine: Sciences et Technologie

Filière: Automatique

Spécialité: Automatique et Informatique Industrielle

Par: Laoubi Mounir et Bouchareb Tawfik

Thème

Les systèmes embarqués en temps réel : Application à l'environnement de développement intégré ARDUINO

Soutenu publiquement, le 24 / 06 / 2018 devant le jury composé de:

M. MOUSSAOUI Abdelkrim	Professeur	Univ. Guelma	Président
M. MOUSSAOUI Abdelkrim	Professeur	Univ. Guelma	Encadreur
M. BOUDJEHEM Badreddine	Docteur MCA	Univ. Guelma	Examinateur
Mme. LOUCIF Fatiha	MAA	Univ. Guelma	Examinateur
Mme. BOUCERREJ Leila	Docteur MCB	Univ. Guelma	Examinateur

Année Universitaire: 2017/2018

REMERCIEMENT

Je remercie DIEU tout puissant de m'avoir donné le courage et la patience pour réaliser ce travail.

Ce n'est pas la rédaction d'un tel rapport qui exige un remerciement, mais si on remercie des gens c'est par ce qu'ils méritent.

Au terme de ce travail, on tient à remercier ***Mr. Pr.Moussaoui*** *de nous avoir encadrés et suivis durant notre projet de fin d'étude. Ainsi qu'à tous les professeurs de l'université de Guelma, qui nous ont enseignés durant notre formation universitaire.*

On remercie également nos parents pour leur soutien moral et financier durant nos études.

A tous nos collègues, amis, et tous ceux qui nous ont aidé et soutenu de près ou de loin.

Un spécial remerciement à mon chère ami BELAHCENE Houcem eddine

DEDICACE

A mes chers parents

Pour leur soutien, leur patience, et leur sacrifice, vous méritez tout éloge,

J'espère être l'image que vous êtes fait de moi,

que dieu vous garde et vous bénisse.

Je dédie aussi ce travail à toute ma famille

A tous mes amis à l'Université de Guelma et ailleurs.

A tous ceux qui m'ont aidé.

Sommaire

Liste des Figures :

Introduction générale

Introduction générale :

Les systèmes embarqués en temps réel sont des systèmes indispensables dans notre vie quotidienne (smartphone, machine à laver, centrale de sécurité, etc.).

Il existe beaucoup d'environnement de développement pour ce type système, parmi lesquels le fameux petit module Arduino.

En effet Arduino est un outil, une communauté et une façon de penser qui transforme notre regard sur la technologie et l'usage que nous en faisons. Il a ravivé l'intérêt pour l'électronique embarquée chez de nombreuses personnes et leur a permis de mieux la comprendre.

Arduino est un minuscule circuit imprimé au potentiel gigantesque. Selon la manière dont vous l'approchez, il peut être utilisé pour émettre un signal en code Morse à l'aide d'une LED ou pour contrôler toute les lumières d'un bâtiment. Ses possibilités dépassent de loin tout ce que vous pourriez imaginer. Arduino, c'est aussi une nouvelle approche pratique de l'éducation à la technique, qui réduit le coût d'entrée pour ceux qui souhaitent utiliser l'électronique embarquée pourréaliser de petits projets.

Arduino est plus qu'un « kit d'initiation» ; c'est un outil. Un concentré de technologie qui permet de comprendre et d'utiliser plus facilement les outils technologiques d'aujourd'hui.

Notre mémoire de Master s'inscrit dans le cadre de la perspective de découvrir puis de maîtriser les possibilités immense de la technologie qu'offre ce fameux environnement de développement intégré.

Ce mémoire est structuré ainsi en trois chapitres et une conclusion générale.

Le premier chapitre est consacré aux généralités sur les systèmes embarqués en temps réel, tout en donnant leur architecture, leurs propriétés et leurs domaines d'utilisations.

Dans le deuxième chapitre, on donne une brève présentation sur la carte Arduino, tout en donnant les différents types, les parties essentielles de la carte, le principe de fonctionnement et ses caractéristiques essentielles.

Le troisième chapitre est consacré à la conception et la réalisation de deux applications illustratives en l'occurrence une application aux Feux de signalisation et une application à la mesure de grandeurs physiques analogiques (température et humidité dans le cas de notre projet). Avant l'implémentation sur le module Arduino, nous avons effectué les simulations nécessaires sur un simulateur dédié d'Arduino.

Enfin, nous avons donné une conclusion générale.

Chapitre 1

Aperçu sur les Systèmes Embarqués en Temps Réel

I.1 Généralités :

Un système embarqué est un système électronique et informatique spécialisé dans une tâche bien précise.

Il est dit embarqué dans le sens où il fait parti d'un système complet et y intègre un rôle. L'exemple d'un système embarqué typique est celui du véhicule autonome donné dans la Figure I.1.

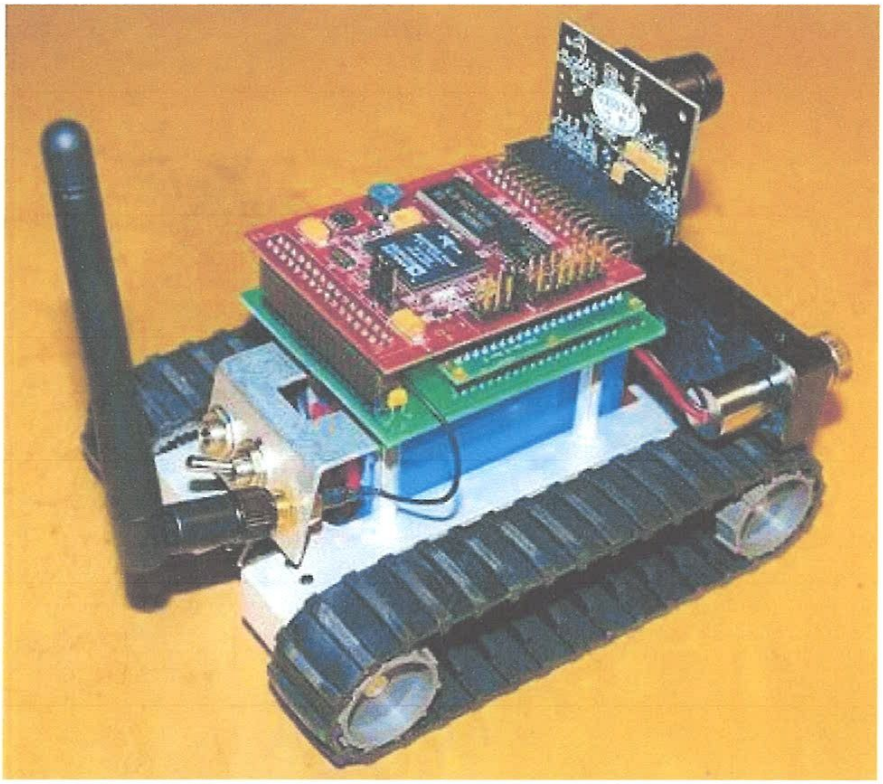


Figure I.1 : Un véhicule autonome.

I.2 Caractéristiques principales d'un SE :

- C'est un système principalement numérique.
- Dispose de ressources limitées. Cette limitation est généralement d'ordre spatial (encombrement réduit) et énergétique (consommation restreinte).
- Il ne possède pas d'entrées/sorties standards tels qu'un port série RS232, usb, … etc et de périphériques classiques comme un clavier d'ordinateur (possède périphériques

limités : des boutons poussoir, clavier matriciel …) ou un écran d'ordinateur (affichage limité ou aucun affichage : LED, écran LCD, ….etc).

- Exécute un logiciel dédié à des fonctionnalités bien précises, qui est en partie ou entièrement programmé dans le matériel (*firmware*).
- Possède généralement un fonctionnement en temps réel.
- Il est généralement ouvert au monde extérieur pour des mises à jour ou pour communiquer mais ce n'est pas indispensable.
- Dans certaines applications, il a un fonctionnement sûr pour ne pas mettre en péril des vies humaines ou en danger des investissements importants.

I.3 Classification des systèmes embarqués :

Une classification possible en fonction de ce qu'à quoi les SE sont destinés : On distingue 4 types de systèmes embarqués :

- Les SE dédiés aux calculs et aux jeux vidéo,
- Les SE dédiés au contrôle de systèmes,
- Les SE dédiés aux traitement du signal,
- Les SE dédiés aux communications et réseaux.

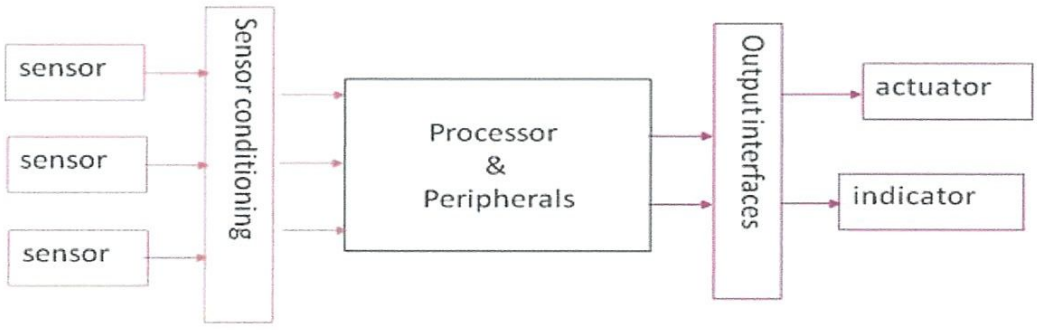


Figure I.2 : Schéma d'un système embarqué typique

I.4 Architecture d'un système embarqué :

Un SE est constitué de deux types de composantes : des composantes matérielles et des composantes logicielles.

Cette composition dépend fortement du type de SE. La figure suivante résume les composantes de base que l'on rencontre par exemple dans tous les SE dédiés au contrôle.

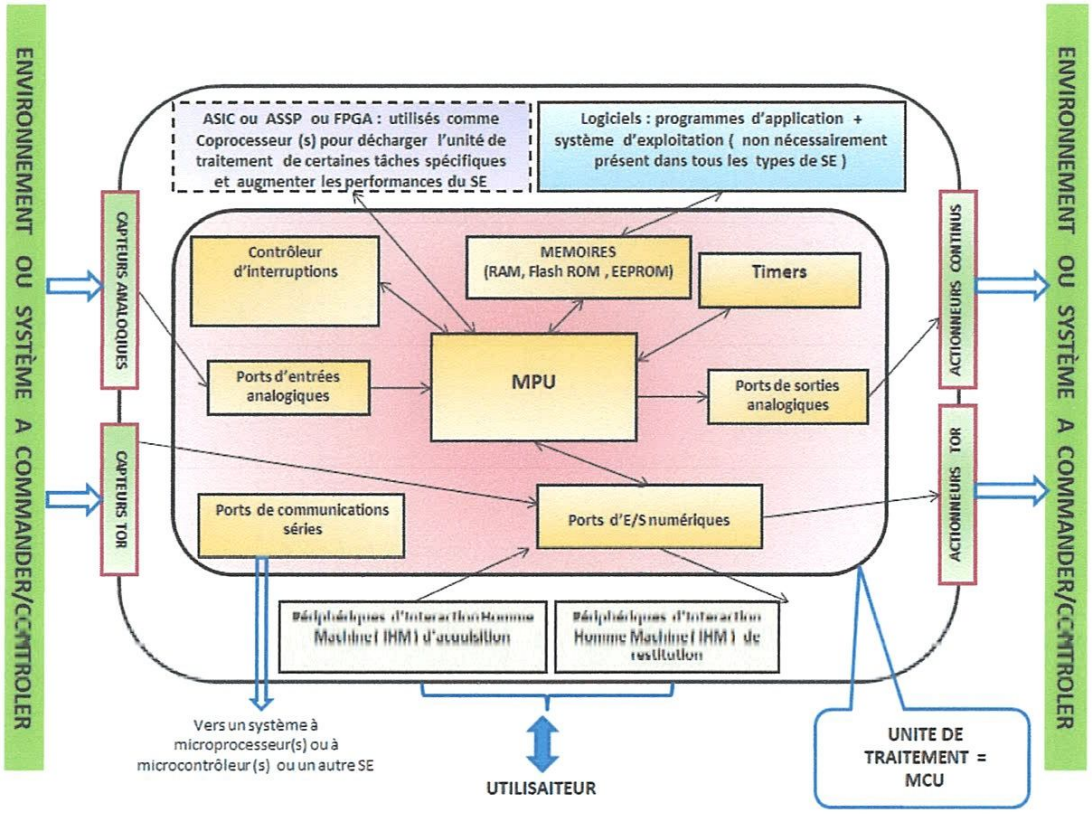


Figure I.3 : Architecture de base d'un SE

I.5 Descriptions de quelques blocs fonctionnels:

I.5.1 L'unité de traitement ou la CPU:

Peut être :

- Un GPP : General Purpose Processor = MPU standards avec les différents périphériques classiques , MCU standards, un processeur spécial comme l'ARM 7, L'INTEL i960, L' AMD 29050, etc…), ou un DSP .

Ou :

- un MPS utilisant plusieurs GPP : Multi Processor System utilisant plusieurs GPP .

I.5.2 L'ASIC/ ASSP :

Composants matériels qui ne se vendent pas et qui contiennent des programmes spécifiques ou dédiés à une tâche bien précise dès leur sortie de l'usine. Ces programmes sont appelés programmes maison ou Firmware.

Ces circuits intégrés ne sont pas reprogrammables après leur sortie de l'usine et donc leur Firmware ne peut pas être modifié par l'utilisateur.

I.5.3 FPGAs :

Contrairement aux ASIC et ASSP, ces circuits intégrés se vendent et peuvent ou non contenir un Firmware à leur sortie de l'usine.

Ces circuits sont reprogrammable après leur sortie de l'usine et donc leur firmware peut être modifié ou mis à jour par l'utilisateur.

I.6 Quelques exemples d'utilisation des SEs :

Freins ABS	Modems
Système de péage automatique	Cartes réseau
Transmission automatique	Commutateurs
Systèmes avioniques	Systèmes de navigation (GPS)
Chargeurs de piles	Photocopieurs
Caméras vidéo numériques	Jeux vidéos
Téléphones cellulaires	Imprimantes
Régulateur de vitesse	Numérisateurs (scanners)
Disques durs	Fours micro-ondes
Lecteurs de cartes bancaires	Lave-vaisselle
Instrumentation électronique	Reconnaissance de voix
Jouets électroniques	Téléconférence
Télécopieurs	Téléviseurs

Identification d'empreintes digitales	Systèmes de chauffage
Systèmes d'alarmes domestiques	Magnétoscopes et lecteurs DVD
Dispositifs médicaux	Électroménagers

I.7 Démarche à suivre pour concevoir et réaliser un système embarqué :

Partant d'un cahier de charge dans lequel est défini la structure globale du SE , il faut :

1 - Déterminer les composants matériels notamment la cible (le type de processeur) sur laquelle va être exécuté le programme d'application.

2 - Déterminer le système d'exploitation (pour le système embarqué) le plus approprié à la réalisation du SE, s'il y a lieu d'utiliser un système d'exploitation.

3 - Choisir une plate forme de développement : plate-forme sur la quelle vont être mis au point les différentes parties logicielles de la cible.

Sur cette plate forme doivent être installés au préalable un système d'exploitation et les différents outils nécessaires au développement du SE, soient :

Un éditeur, un compilateur, un éditeur de liens, un débogueur , ou un EDI (Environnement de Développement Intégré) qui intègre tous ces outils, un simulateur , … etc.

4 - Choisir une méthode de développement ou d'accès entre la plateforme de développement et la cible. Une méthode parmi les méthodes les plus utilisées est la méthode de développement dite **méthode connectée.** Dans cette méthode, la cible et la plate forme de développement sont reliées de manière permanente par un lien série et parfois aussi par un lien Ethernet utilisés pour le débogage et le chargement du code dans la cible. Cette configuration est illustrée dans la figure I.4.

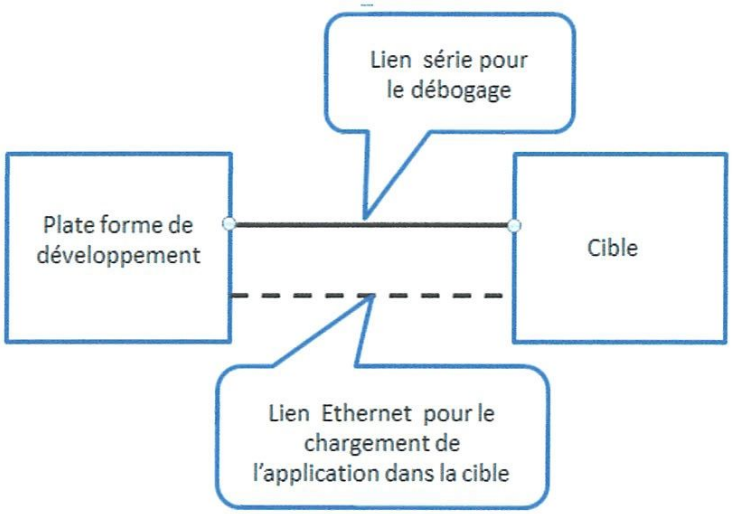


Figure I.4 : Illustration de la méthode connectée

I.8 Les systèmes embarqués en temps réel :

En tenant compte des notions mentionnées dans les sections précédentes, il est établi que les systèmes embarqués en temps réel sont utilisés pour des travaux spécialisés en fonction d'une application spécifique qui présente ***les contraintes de calcul en temps réel.*** Cette situation est commune dans de nombreux domaines d'applications, tels que les usines chimiques et l'industrie de l'automobile, les applications spatiales et de transport (voir Figure I.5)

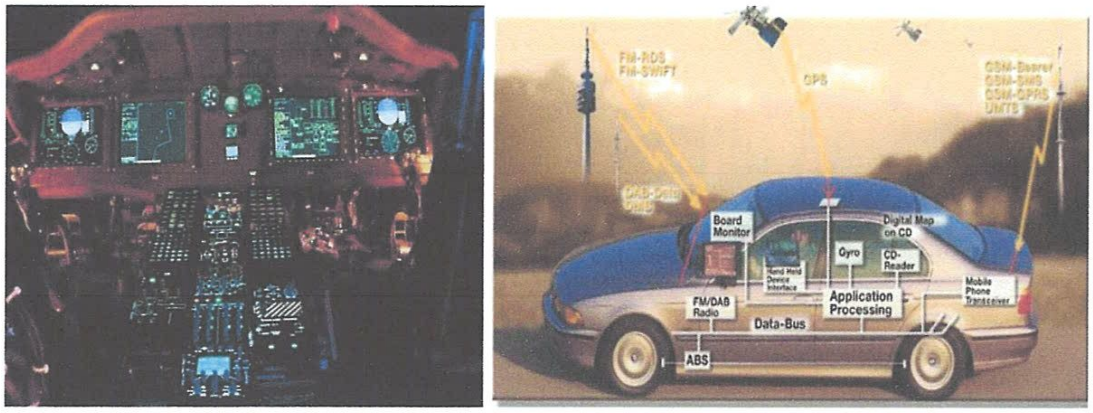


Figure I.5 : Exemples de systèmes embarqués en temps réel

I.8.1 Définition d'un système en temps réel :

Les systèmes en temps réel produisent un résultat d'un processus en exécution. Ce résultat, ne doit pas seulement être logiquement correct, mais doit être disponible dans un certain laps de temps. Dans les systèmes en temps réel, on considère, en général, deux types de tâches: les tâches périodiques et apériodiques. Dans les tâches périodiques, une période est définie comme la quantité de temps entre les itérations de la tâche répétée. Contrairement aux tâches périodiques, les tâches apériodiques répondre à des événements arrivant au hasard.

Compte tenu de l'importance du temps dans un système en temps réel, on peut dire qu'un système en temps réel n'a pas besoin d'être très rapide, mais il doit retourner des résultats corrects dans un temps acceptable, soit toujours avant une un temps limite. Par conséquent, cette contrainte de temps est une limite après laquelle le résultat du traitement peut être sans valeur.

Si l'on considère l'importance du respect des délais, les systèmes en temps réel (En Ligne) comparativement aux systèmes en temps différé (Hors Ligne) sont illustrés sur le spectre du temps de la Figure I.6.

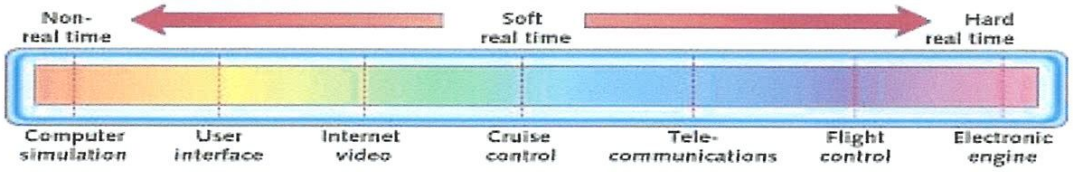


Figure I.6 : Spectre du temps des systèmes embarqués en temps réel

Comme on peut le voir dans le spectre de temps, à une extrémité, où les systèmes en temps non réel (hors ligne) se trouvent, il n'y a pas de délais ou les délais ne sont pas importants. A l'autre bout, soit le cas des systèmes en temps réel, les délais de temps doivent être respectés, sinon cet événement entraîne une défaillance critique du système.

Le non-respect du délai sur un système en temps réel entraine une défaillance du système. Un exemple d'un système en temps réel dur est le contrôleur de vol. Si le contrôleur

ne répond pas dans les délais appropriés, il est possible que l'avion subisse un accident grave. Donc, dans ce type de systèmes, les erreurs ne sont pas tolérées.

En ce qui concerne les systèmes en temps réel doux, les limites de temps doivent aussi être respectées, sauf que leur violation occasionnelle ne doit pas donner lieu à une défaillance critique du système. Mais en aucun cas, cela signifie qu'il est acceptable d'avoir une violation continue. Par exemple, dans le cas de l'application de régulateur de vitesse, si le système ne parvient pas à mesurer la vitesse dans le temps approprié, il peut utiliser la valeur précédente de celle-ci, sans qu'il en résulte une erreur, comme la différence entre l'ancienne et la valeur actuelle de la vitesse est très petite. Mais si le défaut de mesure persiste, il pourrait conduire à un problème, parce que le régulateur de vitesse cesserait probablement de répondre aux exigences de l'application.

I.8.2. Prévisibilité d'un système en temps réel :

Prévisibilité est un terme utilisé pour décrire les systèmes en temps réel. Quand nous disons qu'un système en temps réel est prévisible, cela signifie que son comportement de synchronisation est dans un intervalle de temps acceptable. En d'autres termes, un système en temps réel doit se comporter d'une manière qui peut être mathématiquement prédite. Ainsi, afin de concevoir un système en temps réel prévisible, nous avons besoin de connaître la période, le délai et le temps d'exécution « pire cas » de chaque application individuelle. En tenant compte de ces paramètres, un système peut être conçu avec l'algorithme d'ordonnancement le plus approprié, afin d'assurer qu'il répond à toutes les contraintes de temps pour la prévisibilité.

Une propriété particulière d'un système en temps réel prévisible est *le déterminisme*. Celui-ci représente la capacité d'assurer l'exécution d'une application sans craindre que des facteurs extérieurs perturbent l'exécution d'une manière imprévisible. Cela signifie que son comportement temporel peut être prédéterminé. L'exécution des tâches dans ce cas ne se produit que pendant des intervalles de temps prédéfinis. Par conséquent, des limites supérieures sur le temps d'attente pour chaque tâche peuvent être calculées avec précision et permettent d'éviter des anomalies dans la prévisibilité du système [5].

I.9. Conclusion

Dans ce chapitre, nous avons donné un bref aperçu sur les systèmes embarqués en temps réel, tout en donnant leur architecture, leurs propriétés et leurs domaines d'utilisations.

Dans le prochain chapitre, nous allons nous intéresser à un environnement de développement en temps réel à caractère pédagogique et scientifique, en l'occurrence le module ARDUINO.

Chapitre 2

Présentation de l'environnement de développement Arduino

II.1 Introduction :

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

II.2 Définition du module Arduino :

Le module Arduino est une plateforme de contrôle dont les plans de la carte elle-même sont publiés en licence libre dont certains composants de la carte : comme le microcontrôleur et les composants complémentaires qui ne sont pas en licence libre. Un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diverses. Arduino est utilisé dans beaucoup d'applications en temps réel comme l'automatique, l'électrotechnique industrielle; la domotique ; la robotique, la commande des moteurs la communication avec l'ordinateur, la commande des appareils mobiles.

Chaque module d'Arduino possède un régulateur de tension +5 V et un oscillateur à quartez 16 MHz (ou un résonateur céramique dans certains modèles). Pour programmer cette carte, on utilise le *logiciel IDE Arduino*. [3]

II.3 Les gammes de la carte Arduino :

Actuellement, il existe plus de 20 versions de module Arduino, nous citons quelques un afin d'éclaircir l'évaluation de ce produit scientifique et académique:

- Le NG d'Arduino, avec une interface USB pour programmer et usage d'un Microcontrôleur ATmega8.
- L'Arduino Mini, une version miniature de l'Arduino en utilisant un microcontrôleur ATmega168.
- L'Arduino Nano, une petite carte programmable à l'aide du port USB. Cette version utilisant un microcontrôleur ATmega168 (ATmega328 pour une plus nouvelle version).
- Le NG d'Arduino plus, avec une interface USB pour programmer et usage d'un ATmega168.

- L'Arduino Bluetooth, avec une interface de Bluetooth pour programmer en utilisant un microcontrôleur ATmega168.
- L'Arduino Mega, en utilisant un microcontrôleur ATmega1280 pour I/O additionnelles et mémoire.
- L'Arduino UNO, utilisant le microcontrôleur ATmega328.
- L'Arduino Mega2560, utilisant le microcontrôleur ATmega2560, et possédant une mémoire de 256 KBs. Elle incorpore également le nouvel ATmega8U2 (ATmega16U2 compatible avec la révision 3 du port USB).

Parmi ces types, la carte Arduino UNO est la plus utilisée (carte Basique). L'intérêt principal de cette carte est de faciliter la mise en œuvre des applications en temps réel.
L'Arduino fournit un environnement de développement s'appuyant sur des outils open source comme interface de programmation. L'injection du programme déjà converti par l'environnement sous forme d'un code « HEX » dans la mémoire du microcontrôleur se fait d'une façon très simple par la liaison USB. En outre, des bibliothèques de fonctions "clé en main" sont également fournies pour l'exploitation d'entrées-sorties. Cette carte est basée sur un microcontrôleur ATmega 328 et des composants complémentaires.
La carte Arduino contient :

- une mémoire morte (ROM) de 1 kilo.
- 14 entrées/sorties digitales (dont 6 peuvent être utilisées en tant que sortie PWM « modulation en largeur d'impulsion »),
- 6 entrées analogiques (précision de 10 bits),
- un oscillateur à quartz de 16 MHz,
- une connexion USB
- et Possède un bouton de remise à zéro et une prise jack d'alimentation.

La carte est illustrée dans la figure II.1.

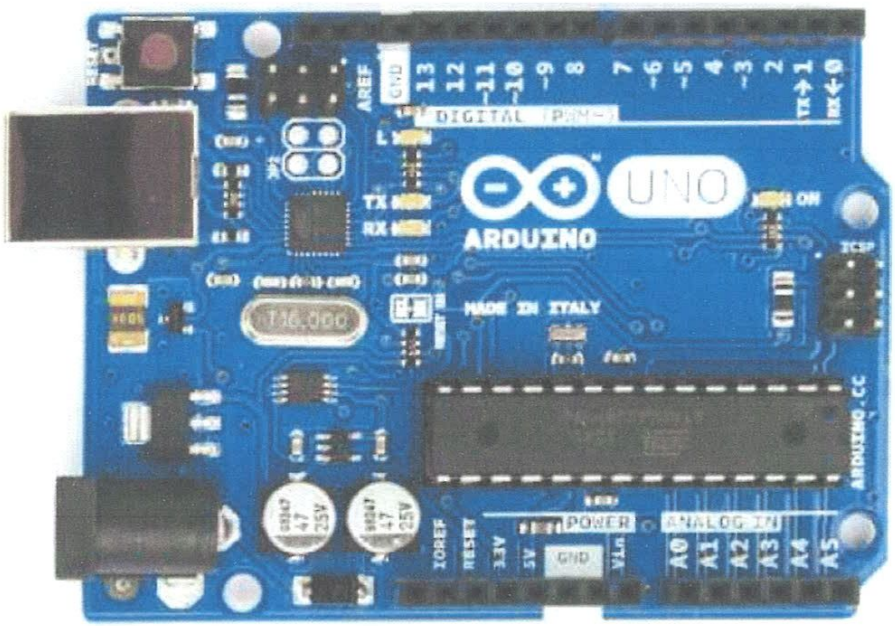


Figure II.1 : La carte Arduino UNO

II.4 Pourquoi Arduino UNO :

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

- **Le prix (réduits)** : les cartes Arduino sont relativement peu coûteuses comparativement aux autres plates-formes. La moins chère des versions du module Arduino peut être assemblée à la main, (les cartes Arduino pré-assemblées coûtent moins de 2500 Dinars).
- **Multi plateforme** : 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 (le logiciel *Arduino IDE*) est facile à utiliser pour les débutants, tout en étant assez flexible pour que les utilisateurs avancés puissent en tirer profit également.
- **Logiciel Open Source et extensible :** le logiciel Arduino et le langage Arduino sont publiés sous licence open source, disponible pour être complétés par des programmateurs expérimentés. Le logiciel de programmation des modules Arduino est une application JAVA multi plateformes (fonctionnant sur tout système d'exploitation), servant d'éditeur de code et de compilateur, et qui peut transférer le programme au travers de la liaison série (RS232, Bluetooth ou USB selon le module).
- **Matériel Open source et extensible** : les cartes Arduino sont basées sur les Microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA 328. Les schémas des modules sont publiés sous une licence créative Commune et les concepteurs des circuits expérimentés peuvent réaliser leur propre version des cartes Arduino en les complétant et en les améliorant. Même les utilisateurs relativement inexpérimentés peuvent fabriquer la version sur plaque d'essai de la carte Arduino, dont le but est de comprendre comment elle fonctionne pour économiser le coût.

I. 5 Constitution de la carte Arduino UNO :

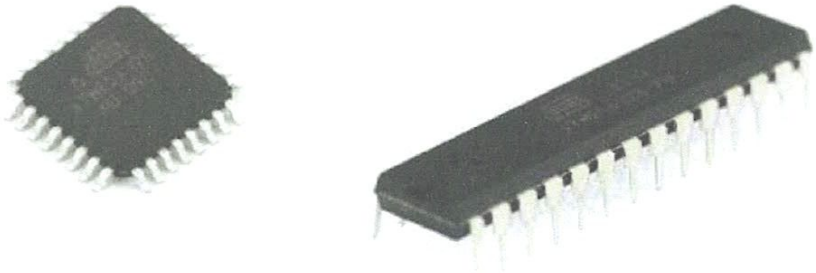
Un module Arduino est généralement construit autour d'un microcontrôleur ATMEL AVR, et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits. Chaque module possède au moins un régulateur linéaire 5V et un oscillateur à quartz 16 MHz (ou un résonateur céramique dans certains modèles). Le microcontrôleur est préprogrammé avec un « *bootloader* » de façon à ce qu'un programmateur dédié ne soit pas nécessaire.

II.5.1 Partie matérielle :

Généralement tout module électronique qui possède une interface de programmation est basé toujours dans sa construction sur un circuit programmable ou plus.

II.5.1.1 Le Microcontrôleur ATMega328 :

Un microcontrôleur ATMega328 est un circuit intégré qui rassemble sur une puce plusieurs éléments complexes dans un espace réduit au temps des pionniers de l'électronique. Aujourd'hui, en soudant un grand nombre de composants encombrants ; tels que les transistors; les résistances et les condensateurs tout peut être logé dans un petit boîtier en plastique noir muni d'un certain nombre de broches dont la programmation peut être réalisée en langage C, ou JAVA. La figure II.2 montre un microcontrôleur ATmega 328, qu'on trouve sur la carte Arduino. [1]



Le composant CMS

Le composant classique

Figure II.2 : Microcontrôleur ATMega328

Le microcontrôleur ATMega328 est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur. Globalement, l'architecture interne de ce circuit programmable se compose essentiellement de :

- ***La mémoire Flash***: C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible mémoire programme de 32Ko (dont bootloader de 0.5 ko).
- ***RAM*** **:** c'est la mémoire dite "vive", elle va contenir les variables du programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur. Sa capacité est 2 ko.

- ***EEPROM*** : C'est le « disque dur » du microcontrôleur. On y enregistre des infos qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme. [2]

II.5.1.2 Les sources d'alimentation de la carte :

On peut distinguer deux genres de sources d'alimentation (Entrée/Sortie) et cela comme suit :

- *VIN*. La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). On peut alimenter la carte à l'aide de cette broche, ou, si l'alimentation est fournie par la fiche jack d'alimentation, accéder à la tension d'alimentation sur cette broche.
- **5V**. La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (pour info : les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.
- ***3.3V*** C'est une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de l'ordinateur et le port série de l'ATmega) de la carte : ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V. L'intensité maximale disponible sur cette broche est de 50mA. [4]

II.5.1.3 Les Entrées-Sorties :

Cette carte possède 14 broches numériques (numérotée de 0 à 13). Chaque broche peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions *pinMode( ), digitalWrite( ) et digitalRead( )* du langage Arduino. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité

et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de 20-50 KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digital *Write(broche, HIGH).*

En plus, certaines broches ont des fonctions spécialisées :

- **Interruptions Externes**: Broches 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur.
- **Impulsion PWM (largeur d'impulsion modulée):** Les broches 3, 5, 6, 9, 10, et 11 fournissent une impulsion PWM 8-bits à l'aide de l'instruction *analogWrite ( ).*
- **SPI** (Interface Série Périphérique): Broches 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches supportent la communication SPI (Interface Série Périphérique) disponible avec la librairie pour communication SPI. Les broches SPI sont également connectées sur le connecteur ICSP qui est mécaniquement compatible avec les cartes Mega.
- **I2C**: Broches 4 (SDA) et 5 (SCL). Supportent les communications de protocole I2C (ou interface TWI (Two Wire Interface - Interface "2 fils"), disponible en utilisant la librairie Wire/I2C.
- **LED**: Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

La carte UNO dispose de 6 entrées analogiques (numérotées de 0 à 5), chacune pouvant fournir une mesure d'une résolution de 10 bits (en d'autres termes sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead( ) du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction *analogReference( )* du langage Arduino.

La carte Arduino UNO intègre un fusible qui protège le port USB de l'ordinateur contre les surcharges en intensité (le port USB est généralement limité à 500mA en intensité). Bien que la plupart des ordinateurs aient leur propre protection interne, le fusible de la carte fournit

une couche supplémentaire de protection. Si plus de 500mA sont appliqués au port USB, le fusible de la carte coupera automatiquement la connexion jusqu'à ce que le court-circuit ou la surcharge soit stoppé. [3]

II.5.1.4 Les ports de communications :

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).

On utilise (RX) pour recevoir et (TX) pour transmettre (les données séries de niveau TTL). Ces broches sont connectées aux broches correspondantes du circuit intégré ATmega328 programmé en convertisseur **USB-série** de la carte, composant qui assure l'interface entre les niveaux TTL et le port USB de l'ordinateur.

Comme un port de communication virtuel pour le logiciel sur l'ordinateur, la connexion série de l'Arduino est très pratique pour communiquer avec un PC, mais son inconvénient est le câble USB. Pour éviter cela, il existe différentes méthodes pour utiliser ce dernier sans fil:

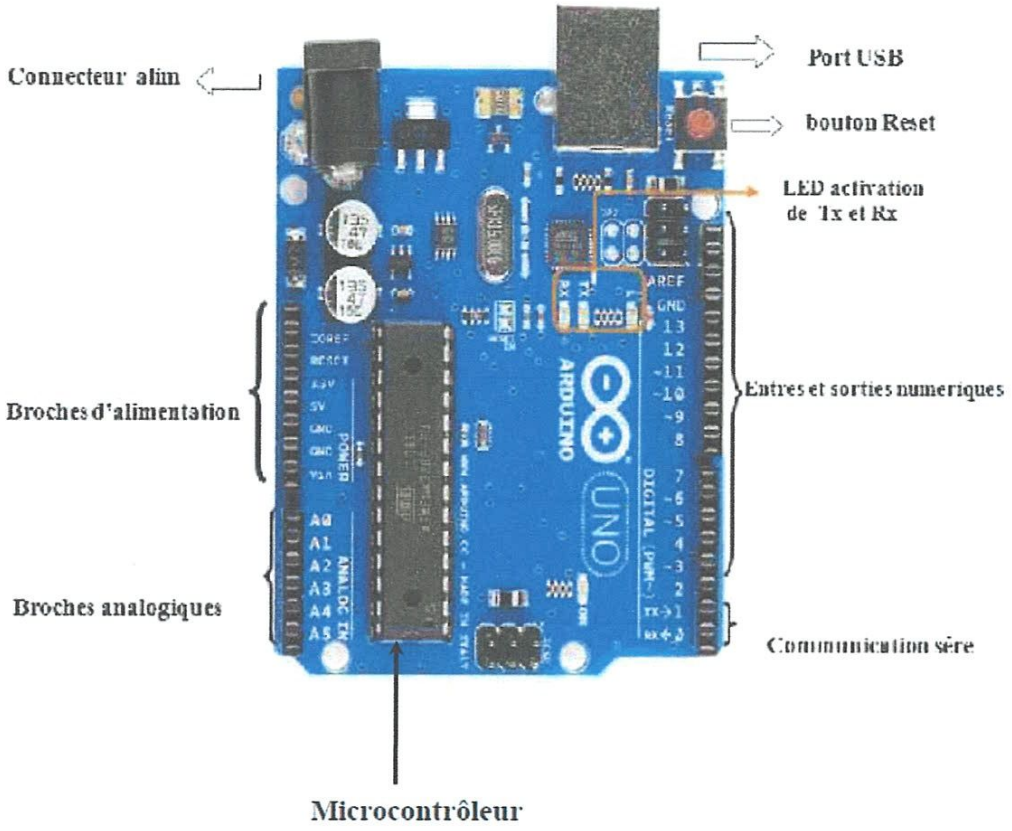


Figure II.3 : Constitution de la carte Arduino UNO

II.5.2 Partie programme :

Une telle carte d'acquisition qui se base dans sa construction sur un microcontrôleur doit être dotée d'une interface de programmation comme c'est le cas de notre carte. L'environnement de programmation open-source pour Arduino peut être téléchargé gratuitement (pour Mac OS X, Windows, et Linux).

II.5.2.1 l'environnement de la programmation :

Le logiciel de programmation de la carte Arduino sert d'éditeur de code (langage proche du C). Une fois, le programme tapé ou modifié au clavier, il sera transféré et mémorisé dans la carte à travers de la liaison USB. Le câble USB alimente à la fois en énergie la carte et transporte aussi l'information. Ce programme est appelé **IDE Arduino.** [7]

II.5.2.2 Structure générale du programme (IDE Arduino) :

Comme n'importe quel langage de programmation, c'une interface souple et simple exécutable sur n'importe quel système d'exploitation et basé sur la programmation en C.

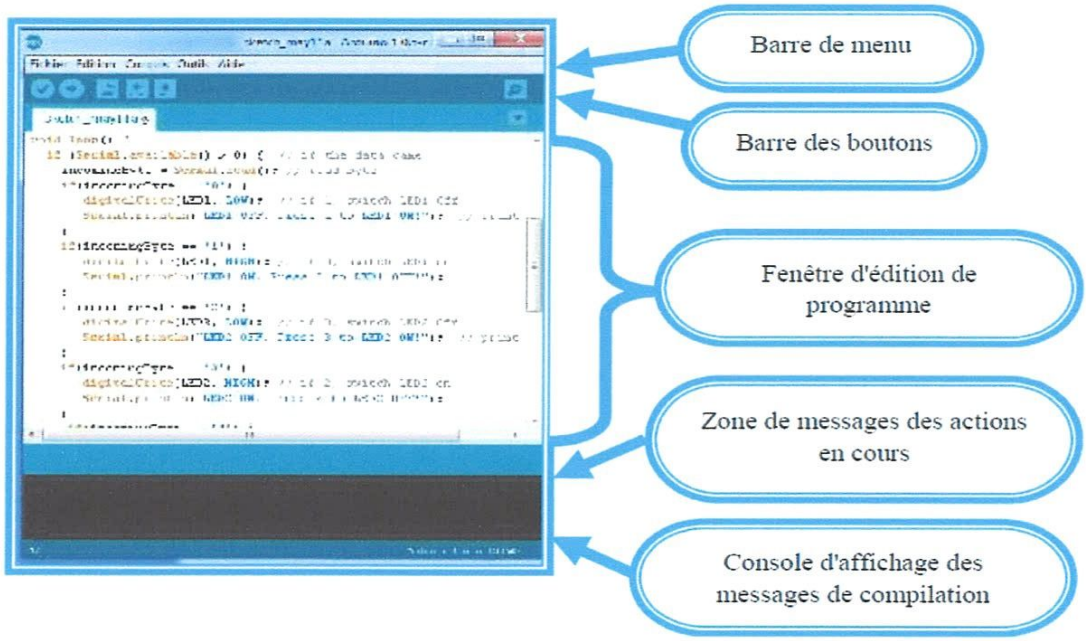


Figure II.4 : Interface IDE Arduino

II.5.2.3 Injection du programme :

Avant d'envoyer un programme dans la carte, il est nécessaire de sélectionner le type de la carte (Arduino UNO) et le numéro de port USB (COM 3) à titre d'exemple comme il montré sur la figure II.5.

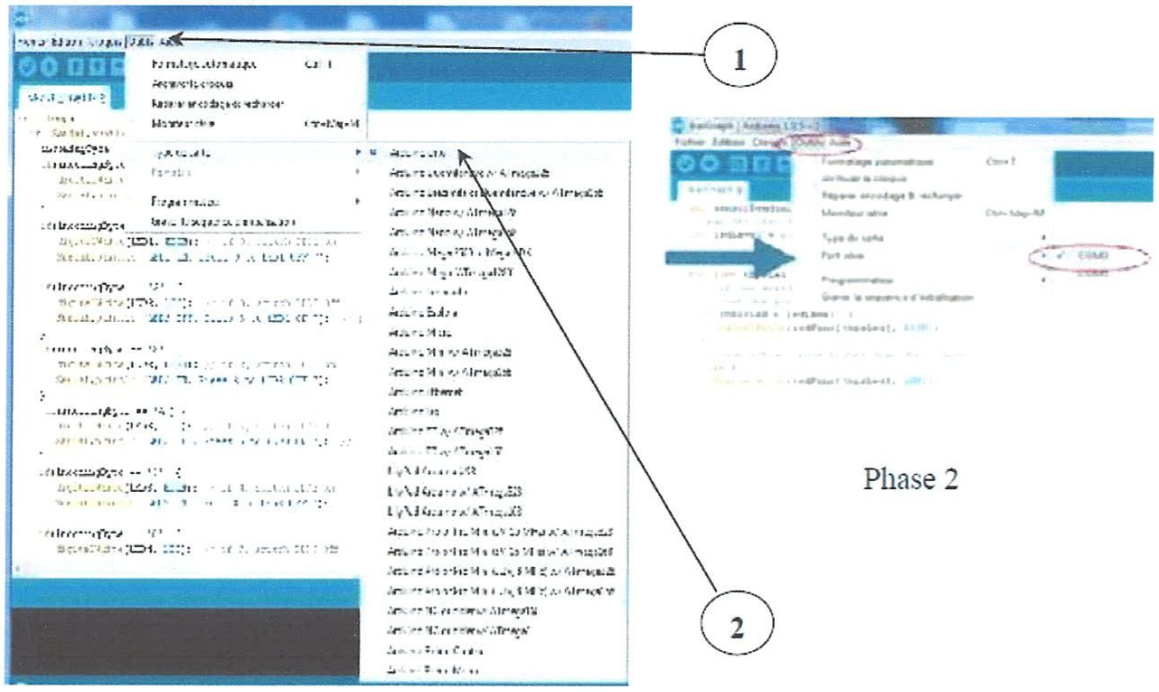


Figure II.5 : Paramétrage de la carte

II.5.2.4 Description du programme :

Un programme Arduino est une suite d'instructions élémentaires sous forme textuelle (ligne par ligne). La carte lit puis effectue les instructions les unes aprés les autres dans l'ordre défini par les lignes de codes.

Commentaires :

Les **commentaires** sont, en programmation informatique, des portions du code source ignorées par le compilateur ou l'interpréteur, car ils ne sont pas censés influencer l'exécution du programme.

```
1 /* programme de commande de Moteur DC
2 *avec clignotement de la diode de test de la carte
3 */
```

Définition des variables :

Pour une sortie numérique par exemple la 3ème sortie numérique ; cette variable doit être définie et nommée par exemple « moteur » associée au pin 3 ; la syntaxe pour désigner un nombre entier est **int.**

```
4 int moteur = 3; // mettre le moteur au pin 3
```

Configuration des entrées et des sorties void setup () :

Les broches numériques de l'Arduino peuvent aussi bien être configurées en entrées numériques ou en sorties numériques; à titre d'exemple, on va configurer la variable moteur pin en sortie ; pinMode(nom,état) est une des quatre fonctions relatives aux entrées – sorties numériques.

```
5 void setup() {
6 // mettre le moteur comme sortie:
7 pinMode(moteur , OUTPUT); // lorsque le pin 3 est activé le moteur tourne
8 }
```

Programmation des interactions void loop :

Dans cette boucle, on définit les opérations à effectuer dans l'ordre **digitalWrite**(nom, état) est une autre des quatre fonctions relatives aux entrées – sorties numériques.

- **delay** (temps en mili-seconde) est la commande d'attente entre deux instructions.
- chaque ligne d'instruction est terminée par un point virgule.
- Les accolades délimitent le début et la fin de la boucle.

```
9 void loop() {
10 digitalWrite(moteur ,HIGH);
11 delay (3000)
12 digitalWrite(moteur, LOW);
13 delay (1000)
14 }
```

II.5.2.5 Les étapes de téléchargement du programme :

Une simple manipulation enchaînée doit être suivie afin d'injecter un code vers la carte Arduino via le port USB.

1. On conçoit ou on ouvre un programme existant avec le logiciel IDE Arduino.
2. On vérifie ce programme avec le logiciel Arduino (compilation).
3. Si des erreurs sont signalées, on corrige le programme.
4. On charge le programme sur la carte.
5. On câble le montage électronique.
6. L'exécution du programme est automatique après quelques secondes.
7. On alimente la carte soit par le port USB, soit par une source d'alimentation autonome (pile 9 volts par exemple).
8. On vérifie que notre montage fonctionne.

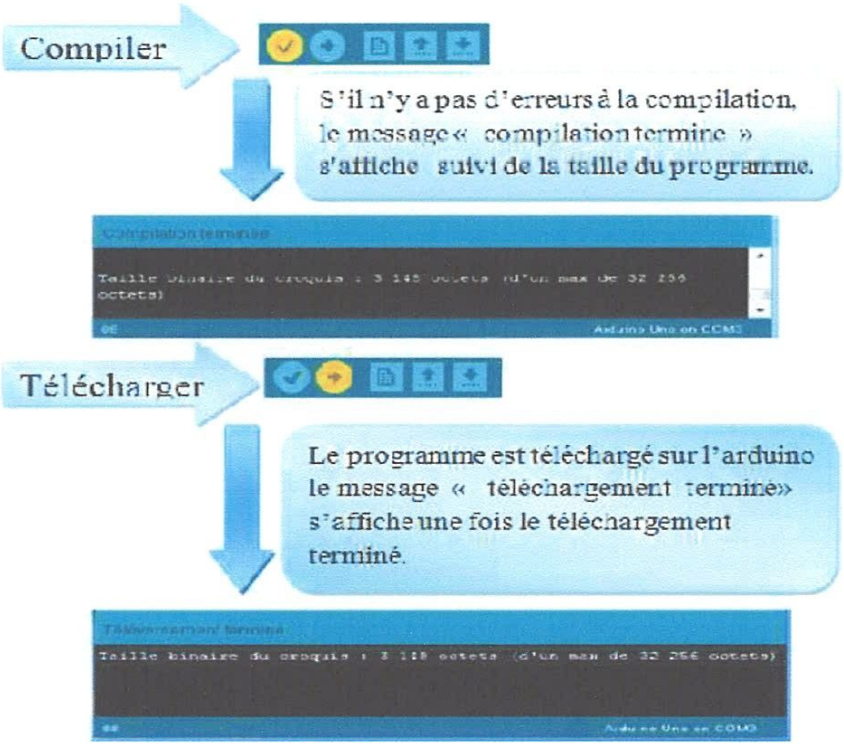


Figure II.6 : Les étapes de téléchargement du code

II.6 Les Accessoires de la carte Arduino :

La carte Arduino est généralement associée aux accessoires qui simplifient les réalisations.

II.6.1 Communication :

Le constructeur a suggéré qu'une telle carte doit être dotée de plusieurs ports de communications ; on peut éclaircir actuellement quelques types.

II.6.1.1 Le module Arduino Bluetooth :

Le Module Microcontrôleur Arduino Bluetooth est la plateforme populaire Arduino avec une connexion sérielle Bluetooth à la place d'une connexion USB, très faible consommation d'énergie, très faible portée (sur un rayon de l'ordre d'une dizaine de mètres), faible débit, très bon marché et peu encombrant.

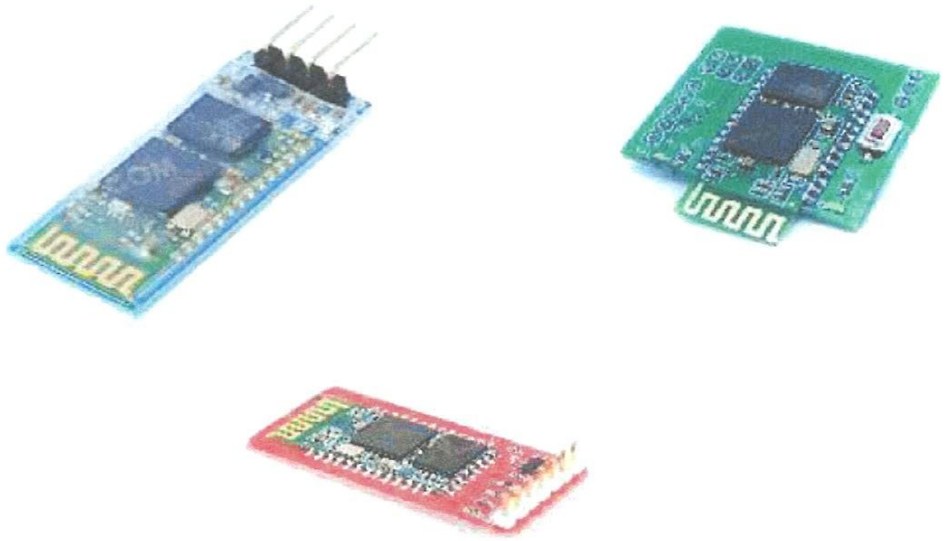


Figure II.7 : Types de modules Bluetooth

II.6.1.2 Le module shield Arduino Wifi :

Le module Shield Arduino Wifi permet de connecter une carte Arduino à un réseau internet sans fil Wifi.

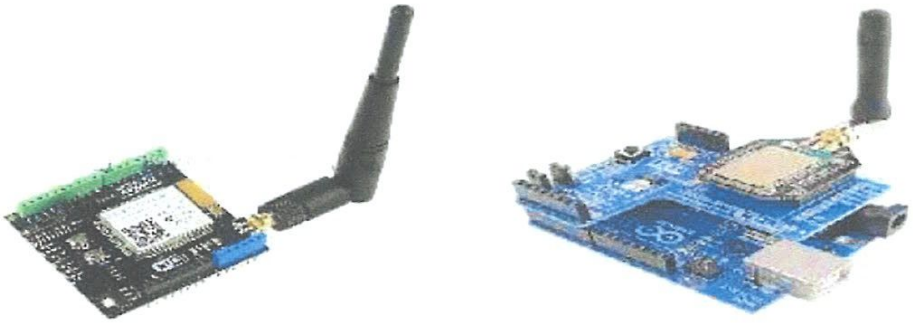


Figure II.8 : Module shield wifi

I.6.1.3 Le Module XBee :

Ce module permet de faire de la transmission sans fil, faible distance/consommation /débit/prix. [6]

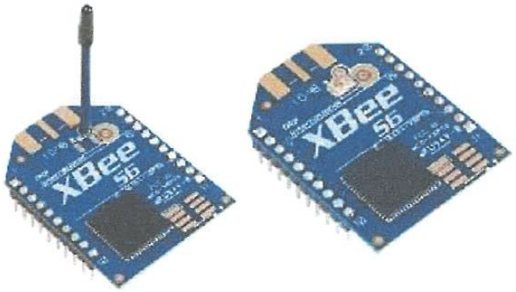


Figure II.9 : Module XBee

II.6.2 Les capteurs :

Un capteur est une interface entre un processus physique et une information manipulable. Il ne mesure rien, mais fournit une information en fonction de la sollicitation à laquelle il est soumis. Il fournit cette information grâce à une électronique à laquelle il est associé. [6]

Figure II.10 : Exemple de Capteur adaptable à ARDUINO

II.6.3 Les Drivers :

Il existe plusieurs Drivers comme des cartes auxiliaires qui peuvent être attachées avec la carte Arduino afin de faciliter la commande ; on peut citer quelques types.

- Les moteurs électriques :

Figure II.11 : Moteurs électriques

- Les afficheurs LCD :

Les afficheurs LCD sont devenus indispensables dans les systèmes techniques qui nécessitent l'affichage des paramètres de fonctionnement. Ces Afficheurs permettent d'afficher des lettres, des chiffres et quelques caractères spéciaux. Les caractères sont prédéfinis. [6]

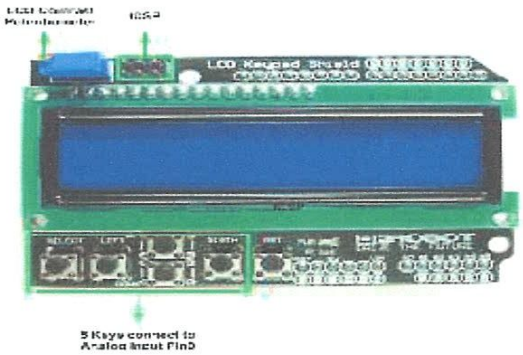


Figure II.12 : Afficheur LCD

- Le relais :

C'est un composant qui possède une bobine (électro-aimant) qui est parcourue par un courant électrique agissant sur un ou plusieurs contacts. Le relais est une solution à la commande en puissance. Il assure en outre une isolation galvanique en mettant en œuvre un mouvement mécanique. [10]

Figure II.13 : Les Relais

II.7 Conclusion :

Dans ce chapitre, nous avons projeté la lumière sur la carte Arduino donnant ainsi les raisons pour lesquelles on l'a choisie, puis nous avons cité des différents types de cette dernière. Ensuite, nous avons expliqué les deux parties essentielles de l'Arduino; (la partie

matérielle et la partie de programmation) plus précisément. Nous avons également expliqué le principe de fonctionnement de la carte Arduino sans oublier ses caractéristiques.

Le chapitre suivant sera consacré à présentation d'un simulateur de la carte Arduino, en l'occurrence « Virtronics Simulator for Arduino », tout en donnant des exemples d'applications illustratifs.

Chapitre 3

Applications d'Arduino

III.1 Introduction :

Dans ce chapitre, nous nous intéressons à quelques applications types d'Arduino qui mettent en évidence les grandeurs numériques et analogiques. Pour ce faire, nous allons tester nos applications sur un simulateur dédié d'Arduino, en l'occurrence Virtronics Arduino Simulator. Et après avoir effectué tous les tests nécessaires, le programme est implémenté (embarqué) en temps réel sur le module Arduino. Les applications types sont dans notre cas les feux de circulation (Traffic Ligths) pour mettre en évidence les entrées-sorties numériques (digitales) et un exemple de mesure de température et d'humidité à l'aide d'un capteur dédié (DHT11) pour mettre en évidence les entrées-sorties analogiques.

III.2 Description du Simulateur d'Arduino :

Le Simulateur d'Arduino est un environnement informatique copiant le fonctionnement du module Arduino. Il présente les avantages et les caractéristiques suivants :

- Utilisation d'Arduino sans avoir besoin du matériel.
- Déboguage de l'esquisse (le code), étape par étape ou exécution du code avec un point d'arrêt ce qui permet de voir le changement des variables .
- Faire une démonstration d'un projet à un client potentiel à distance.

III.3 Utilisation de l'IDE :

Le simulateur pour Arduino IDE a plusieurs sections. Ceux-ci sont:

- Le système de menu qui contient des options et paramètres.
- La barre d'outils de raccourci permet une action simple clic pour les opérations couramment utilisées.
- Le programme ou la fenêtre d'esquisse.
- La zone Variables.
- La zone Arduino.

L'ordre de la fenêtre du programme, les variables et l'image Arduino peuvent être permutés en sélectionnant Affichage | Arduino gauche. La méthode préférée est d'avoir la fenêtre de programme sur la gauche (voir Figure III.1).

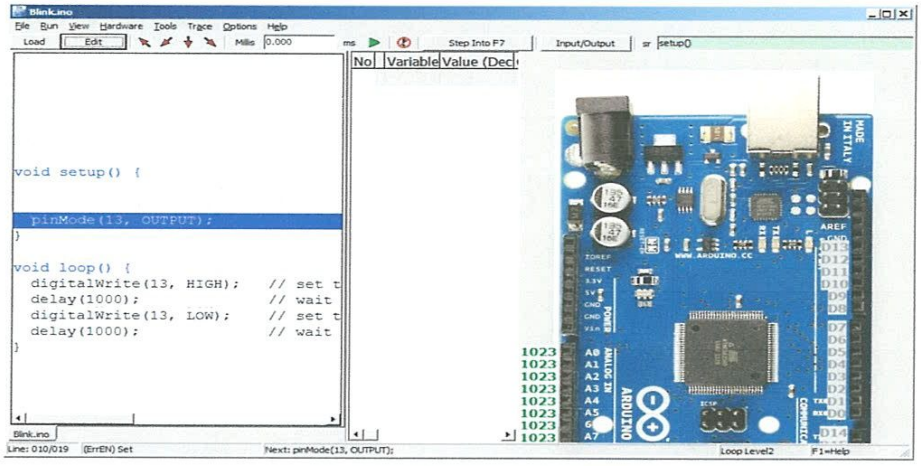


Figure III.1 : Aperçu du simulateur d'Arduino de Virtronics

III.3.1 Fenêtre programme :

La zone de la fenêtre du programme affiche la liste des programmes. Elle permet l'accès au programme. Les touches suivantes peuvent être utilisées pour exécuter le programme :

- F2 peut être utilisée pour forcer une réinitialisation.
- F3 peut être utilisée pour ouvrir la fenêtre Rechercher pour la recherche d'un texte.
- F6 peut être utilisée pour modifier l'esquisse.
- F7 peut être utilisée pour parcourir le programme.
- F8 peut être utilisée pour l'exécution des sous-routines .
- Maj F8 peut être utilisée pour sortir d'une routine.
- F9 peut être utilisée pour exécuter le programme.

Une nouvelle ligne peut être sélectionnée par un clic droit partout dans le programme ou la fenêtre d'esquisse et en sélectionnant « Définir l'instruction suivante ».

La barre d'état indique le numéro de ligne, le dernier numéro de ligne et le code suivant à exécuter. Si le temps autostep est réglé à moins de 2 ms, la barre d'état ne sera pas mise à jour jusqu'à ce que l'exécution ou autostep soient arrêtés.

La légende peut être activée en sélectionnant voir | Légende ou en cliquant sur l'image à droite et en sélectionnant Légende. Elle montre les couleurs et les états de la sortie numérique. La valeur analogWrite d'une broche numérique sera affichée à gauche de l'axe numérique lorsque la routine analogWrite() est utilisée.

III.3.2 Chargement d'une esquisse :

Pour charger une esquisse, sélectionnez Fichier | chargez esquisse, appuyez sur le bouton chargez esquisse en haut à gauche ou appuyez sur F4.

Accédez au dossier correct et sélectionnez le fichier * .ino (ou * .pde pour les versions précédentes Arduino). Les fichiers en C peuvent aussi être simulés car la syntaxe Arduino est compatible.

III.3.3 Exécution d'une esquisse :

Après une esquisse a été chargée, sera mis en évidence la première ligne de code exécutable. Maintenant, appuyez sur le bouton Exécution ou la flèche vers le bas. Alternativement, l'étape simple F7 ou sur la touche F8 peut être pressée. Le bouton F9 permet au programme d'être en mode « autorun » avec un intervalle de 5 ms (réglable autostep). L'intervalle peut être réglé à 500ms pour démontrer au ralenti le fonctionnement d'un croquis ou 1ms à courir aussi vite que possible avec un minimum de rafraîchissement.

III.4 Quelques applications d'Arduino :

III.4.1 Application aux Feux de circulation :

Les feux de circulation permettent de résoudre les conflits inhérents aux intersections. Cependant, les modes de gestion des feux d'un carrefour isolé reposent parfois sur des modèles pensés d'une manière qui les rend inadéquats au traitement des différentes situations rencontrées au niveau local.

III.4.1.1 Paramètres nécessaires au calcul de la durée des feux :

Ces paramètres sont très utiles pour la mise en œuvre des stratégies de commande. Leur évolution est considérée comme un facteur déterminant pour l'efficacité de telles stratégies. Cependant, quelle que soit la méthode de commande appliquée à l'intersection, les indications des feux (vert, orange et rouge) doivent se succéder à l'intérieur d'un cycle défini comme étant la durée fixe ou variable, séparant deux passages successifs de l'ensemble des signaux par le même état. Un cycle type d'un feu de signalement est présenté dans l'organigramme de la Figure III.2.

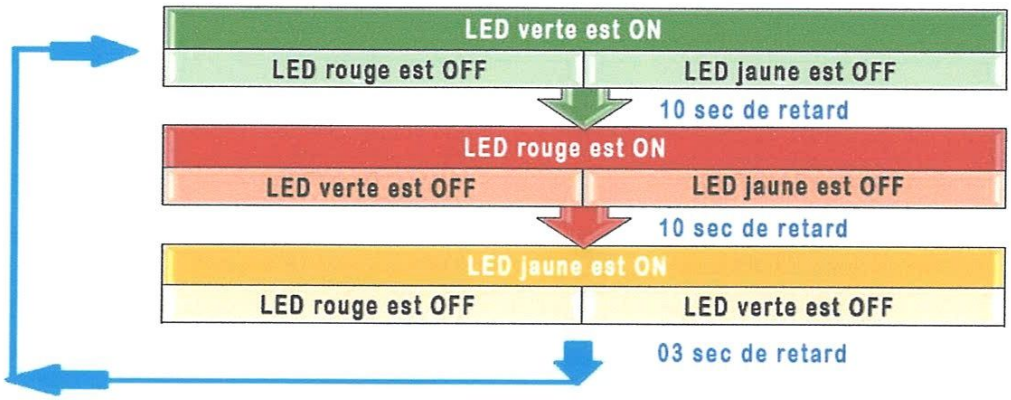


Figure III.2 : Organigramme de Séquences d'un feu de signalisation

Un montage type de simulation d'un feu de signalisation peut être réalisé en utilisant le module Arduino et 3 LED verte, jaune et rouge respectivement. Ce montage est représenté sur la Figure III.3.

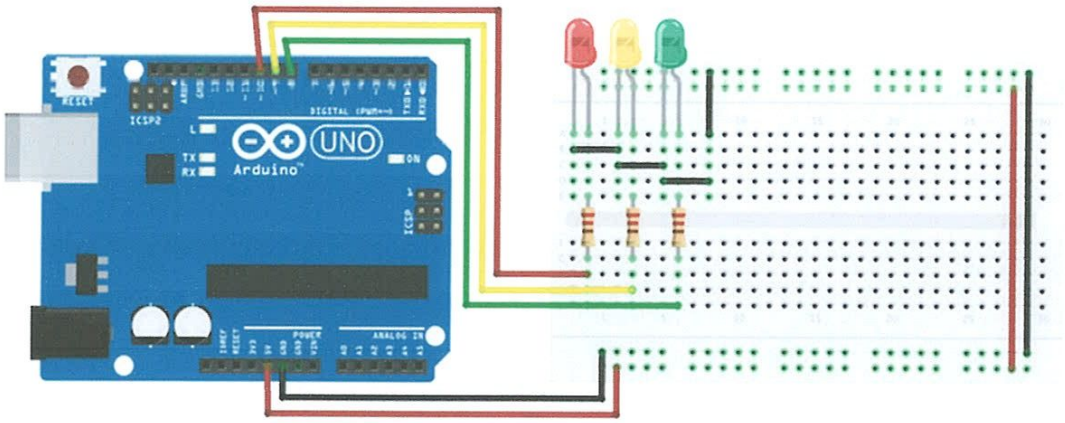


Figure III.3 : Montage de simulation d'un feu de signalisation

Dans un carrefour à feux, la signalisation tricolore peut être régie de trois manières différentes :

- **La gestion 1 :** un feu est vert et les autres sont rouges. Le changement d'état des feux est régulé par ***un temporisateur*** qui applique une durée de vert identique pour chaque voie. Ainsi, pour chaque voie, le feu est vert pendant un quart de cycle .

- **La gestion 2** : deux feux opposés (Nord-Sud ou Est-Ouest) sont verts et les deux autres sont rouges. Le changement d'état des feux est régulé par un temporisateur qui applique une durée de vert identique pour chaque binôme de voies. Ainsi, pour chaque voie, le feu est vert pendant la moitié du cycle .
- **La gestion 3** : deux feux opposés (Nord-Sud ou Est-Ouest) sont verts et les deux autres sont rouges. Le changement d'état des feux est régulé selon la circulation.

Dans le cadre de notre mémoire de Master, nous nous sommes contenté seulement au cas d'un seul feu de signalisation (voir Figure III.3) et ce pour illustrer l'utilisation des entrées-sorties d'Arduino. Le programme de contrôle du feu de circulation est testé tout d'abord sur le simulateur d'Arduino. Puis celui-ci est embarqué en temps réel sur le module Arduino. Le programme est donné comme suit :

```
//---------- Exemple d'un Feu de Circulation ----------
// Déclaration des variables
Int  LED_V = 8;  // mettre la LED Verte au pin 8
Int  LED_J = 9;  // mettre la LED Jaune au pin 9
Int  LED_R = 10;  // mettre la LED Rouge au pin 10

void setup() {
// mettre les LED comme sortie:
pinMode(LED_V , OUTPUT); // lorsque le pin 8 est activé la LED Verte est allumée
pinMode(LED_J , OUTPUT); // lorsque le pin 9 est activé la LED Jaune est allumée
pinMode(LED_R , OUTPUT); // lorsque le pin 10 est activé la LED Rouge est allumée
}

void loop()
{
digitalWrite(LED_V, HIGH);
 digitalWrite(LED_J, LOW);
digitalWrite(LED_R, LOW);
delay(5000); delay(5000);
digitalWrite(LED_V, LOW);
digitalWrite(LED_J, LOW);
digitalWrite(LED_R, HIGH);
delay(5000);delay(5000);
digitalWrite(LED_V, LOW);
digitalWrite(LED_J, HIGH);
```

```
digitalWrite(LED_R, LOW);
delay(3000);
}
```

III.4.2. Application à la mesure de la température et l'humidité :

Pour la mesure de la température et de l'humidité avec le module Arduino, on utilise le capteur **DHT11.** Celui-ci possède les caractéristiques suivantes :

- Mesure de la température de 0 à 50°C.
- Mesure du taux d'humidité de 20 à 96%.
- Alimentation en courant continu comprise entre 3V et 5.5V.
- Consommation : Comprise entre 0.5 mA et 2.5 mA.
- Dimension (Longueur, largeur, Hauteur): 15.5mm, 12mm, 5.5mm.
- Précision pour la mesure de température: ± 2°.
- Précision pour le taux d'humidité: ± 5%.
- Nombre de broches: 4.

D'après la fiche technique de ce capteur, il a :

- Une excellente stabilité à long terme et de temps de réponse très rapidc.
- Un Poids très léger et une petite taille.

Les broches de ce sont, de gauche à droite: VCC | DATA | NC | GDD, comme il montré sur la Figure III.4.

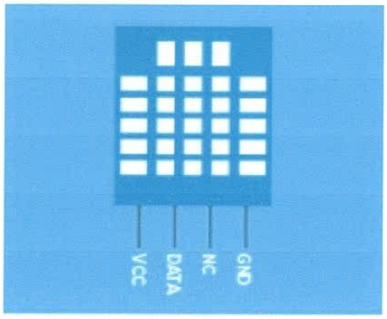


Figure III.4 : Schéma du capteur DHT11

Pour le branchement de ce capteur d'humidité et de température avec la carte Arduino, on ajoute une résistance de 10K entre VCC et DATA, comme il montré dans la figure III.5.

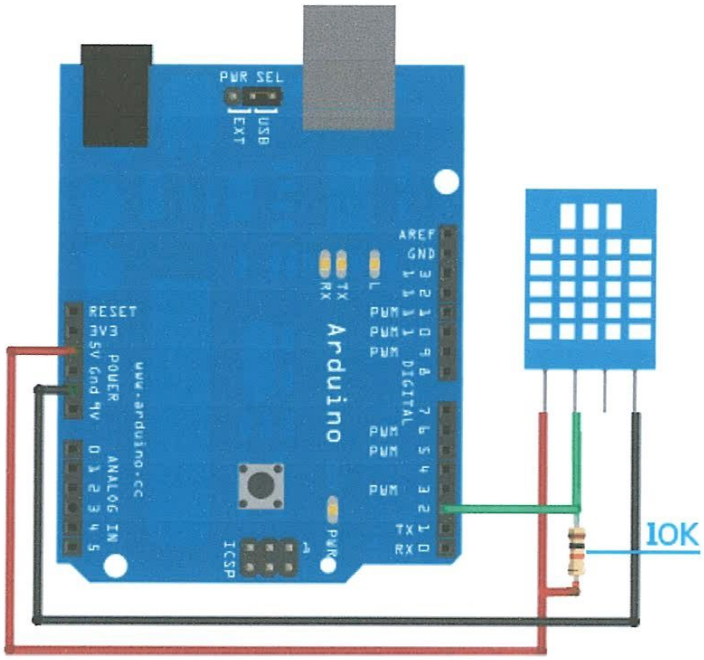


Figure III.5 : Schéma de branchement du capteur DHT11 avec le module Arduino

L'implémentation du programme de mesure de température et d'humidité sur Arduino nécessite l'utilisation d'une bibliothèque spécifique au capteur DHT11 (librairie DHT D'Adafruit). Pour commencer, nous allons l'installer dans l'IDE Arduino. La procédure d'installation est la suivante :

- Allez dans l'onglet « Croquis » -> « Importer bibliothèque » -> « Add library ».
- Une fois la librairie ajoutée fermez l'IDE Arduino.

La librairie d'adafruit est des plus simples. Elle offre l'exploitation de toutes les gammes DHT, dont la DHT11.

Pour configurer la DHT11 :

- on lance tout d'abord l'IDE Arduino.
- Allez dans fichiers -> Examples -> DHT-sensor-library-master -> DHTtester.
- on insère un symbole Commentaire dans la ligne: **#define DHTTYPE DHT22 .**
- ET on enlève le symbole commentaire de la ligne : **#define DHTTYPE DHT11 // DHT 11.**
- On charge ensuite le programme lancer le moniteur série (dans le cas de l'utilisation du simulateur) et on voit s'afficher le résultat (Température et Humidité).

Il faut signaler au passage que l'affichage du résultat est fonction de la résolution des entrées analogiques de l'Arduino qui est de 10 bits. Ce qui fait que pour la mesure, il tenir de la gamme de mesure qui est entre 0 et 5V, soit entre 0 et 1023. Le facteur d'échelle correspondant est le suivant :

Xm= 5 / 1023=0.00488

Exemple :

pour une valeur mesurée de 950, issue du CAN, nous avons une valeur en tension égale 950*Xm=4.64 V.

Le programme de prise en charge correspondant est donné comme suit :

```
// Programme de mesure d'humidité/température avec le capteur DHT11

#include "DHT.h"

#define DHTPIN 2     // Le pin connecté

// On enlève le symbole de commentaire pour le type de capteur
// utilisé

#define DHTTYPE DHT11   // DHT 11

//#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

// Connect pin 1 (sur la gauche) du capteur au +5V
// Initialize DHT sensor for normal 16mhz Arduino

DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin();
}

void loop() {

  // Wait a few seconds between measurements.

 delay(2000);

  // La lecture de la température ou de l'humidité prend 250 milliseconds!

  float h = dht.readHumidity();

// Lire température en degree Celsius
  float t = dht.readTemperature();

// Lire température en degree Fahrenheit
  float f = dht.readTemperature(true);

// Vérification de la lecture (On essaye encore s'il y a échec de
//lecture).

 if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Echec de lecture à partir de la DHT !");
 return;
  }

 // Calcul de l'indice de chaleur
 // La temperature doit être envoyée en Fahrenheit!

 float hi = dht.computeHeatIndex(f, h);

  Serial.print("Humidité: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Température: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.print(" *F\t");
  Serial.print("Indice de Chaleur: ");
  Serial.print(hi);
  Serial.println(" *F");
}
```

III.5. Conclusion :

Dans ce chapitre, est dans le but de comprendre le fonctionnement de l'environnement de développement Arduino, nous avons essayé tout d'abord d'expliquer comment on doit tester un programme sur un simulateur avant de le charger sur le module physique et ce pour enlever toutes les erreurs et les éventuels défauts qui peuvent surgir dans le cas de l'implémentation en temps réel. Et dans un but de montrer les possibilités du module Arduino en termes de flexibilité et d'adaptabilité, nous avons donné deux applications illustratives mettant en évidence ces entrées-sorties numériques et analogiques.

Conclusion générale

Conclusion générale :

Il est établi que Les systèmes embarqués en temps réel sont des systèmes indispensables dans notre vie quotidienne.

Dans notre mémoire, nous avons essayé de montrer l'intérêt de ce type d'outil de développement en l'occurrence le module Arduino, tout en donnant un aperçu sur ses caractéristiques essentielles et les avantages majeurs de son utilisation.

L'intérêt de cet IDE est mis en évidence à travers des exemples d'application illustratifs.

Bibliographie

[1] TAVERNIER C, "Arduino applications avancées", Editions: Dunod.

[2] "Arduino basics", http://www.acm.uiuc.edu/sigbot/tutorials/ 2009-11-17-arduino-basics

[3] "Arduino", http://www.generationrobots.com/fr/152-arduino

[4] NUSSEY John, " Arduino pour les Nuls ", Editions: First - 2ème édition, 2017.

[5] "Les systèmes embarqués. Généralités", http://slideplayer.fr/slide/11815689

[6] ESKIMON, OLYTE, "Arduino pour bien commencer en électronique et en programmation".

[7] NOEL Jean, "Livret Arduino en français", Centre de ressources art sensitif.

- "Simulator for Arduino", http://virtronics.com.au/Simulator-for-Arduino.html

Annexe

- **Indice de chaleur :**

L'indice de chaleur (nom original en anglais Heat Index (HI) ou humiture[1]) est un indice développé aux États-Unis. Il combine la température de l'air ambiant et l'humidité relative, dans des zones ombragées, pour tenter de déterminer la perception de la température que ressent le corps humain[2], c'est-à-dire de combien il ressentirait la chaleur si l'hygrométrie était à une autre valeur à l'ombre. Le résultat est également connu comme la « température ressentie à l'air » ou « la température apparente »

L'indice de chaleur est basé sur la capacité du corps humain à refroidir la peau par la production de sueur. Celle-ci s'évapore dans l'air ce qui nécessite de l'énergie qui est prise au milieu et baisse la température de la couche limite touchant à la peau, donnant une sensation de fraîcheur. Quand l'humidité relative de l'air augmente, l'évaporation se fait moins bien et donne une sensation subjective de chaleur accrue.

Sa formule est calculé à partir des degrés Fahrenheit (°F) selon :

$$\mathrm{HI} = c_1 + c_2T + c_3R + c_4TR + c_5T^2 + c_6R^2 + c_7T^2R + c_8TR^2 + c_9T^2R^2$$

Où :

HI = Indice de chaleur (°F)
T= température de l'air (°F)
R= humidité relative (0 à 100)

$$c_1 = -42.379,\ c_2 = 2.04901523,\ c_3 = 10.14333127,\ c_4 = -0.22475541,$$
$$c_5 = -6.83783 \times 10^{-3},\ c_6 = -5.481717 \times 10^{-2},\ c_7 = 1.22874 \times 10^{-3},$$
$$c_8 = 8.5282 \times 10^{-4},\ c_9 = -1.99 \times 10^{-6}.$$

Comme les coefficients sont reliés à leur unité de température, les valeurs équivalentes en degrés Celsius doivent être donc obtenues par conversion subséquente. Voici un tableau pour des températures et humidité typiques :

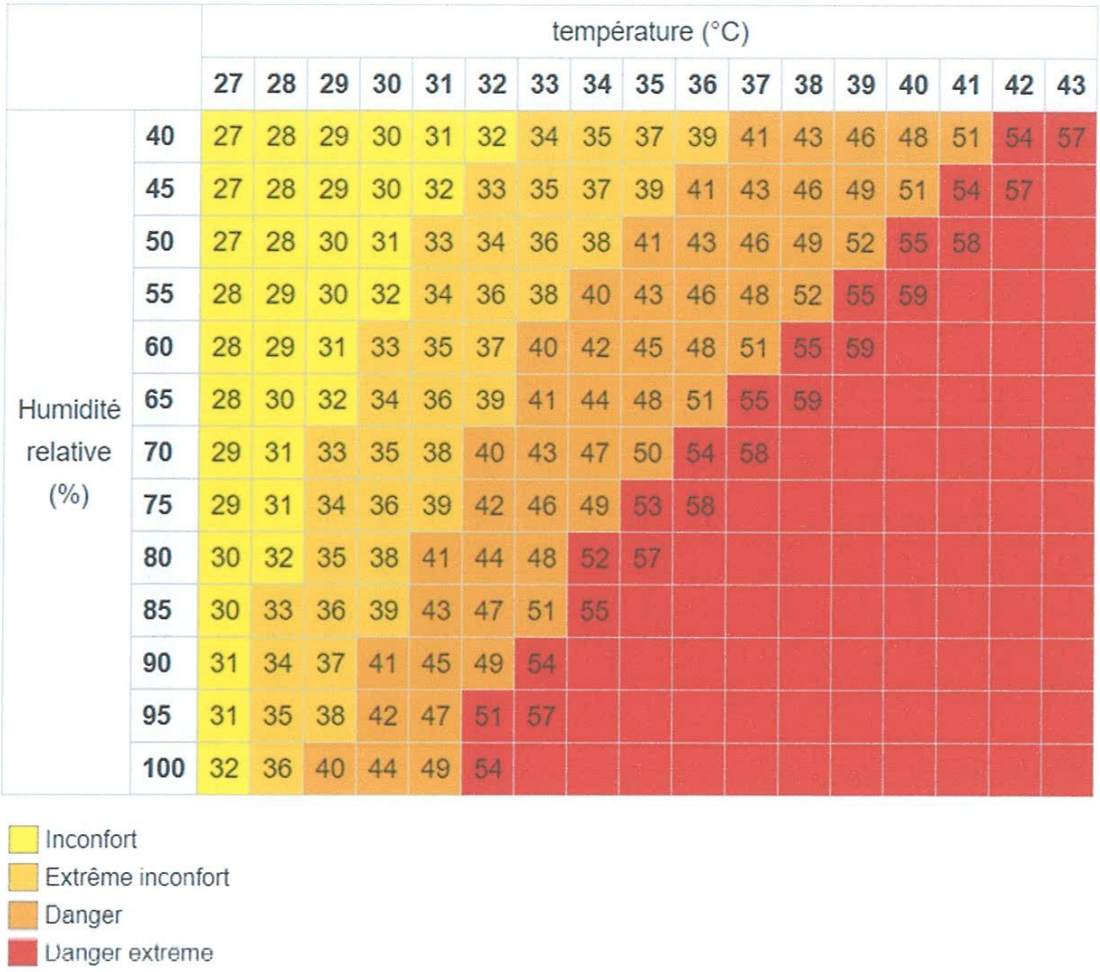


Figure 1 : Indice de chaleur en degrés Celsius

- **Effets de l'indice de chaleur sur le corps humain (valeurs à l'ombre) :**

en degrés Celsius	en degrés Fahrenheit	Notes
27–32 °C	80–90 °F	Attention : la fatigue est possible suite à une activité et une exposition prolongées. Une activité continue pourrait entraîner des crampes de chaleur
32–41 °C	90–105 °F	Attention extrême : des crampes de chaleur et un épuisement par la chaleur sont possibles. L'activité continue peut entraîner une hyperthermie (coup de chaleur/Insolation).
41–54 °C	105–130 °F	Danger : les crampes de chaleur et l'épuisement par la chaleur sont probables; L'hyperthermie (coup de chaleur/Insolation) est probable lors d'une activité continue.
au-delà de 54 °C	au-delà de 130 °F	Danger extrême : L'hyperthermie (coup de chaleur/Insolation) est imminente.

L'exposition directe et prolongée au soleil peut augmenter les valeurs de l'indice de chaleur jusqu'à 8 °C (14 °F). La formule est intégrée dans la bibliothèque de la dht.h.