

UNIVERSITE KASDI MERBAH OUARGLA

Faculté des Sciences Appliquées
Département de Génie Electrique



Mémoire

MASTER ACADEMIQUE

Domaine : Sciences et technologies

Filière : Génie électrique

Spécialité : Electrotechnique Industrielle

Présenté par :

KRAMA Abdelbasset

GOUGUI Abdelmoumen

Thème:

Etude et réalisation d'une carte de contrôle par Arduino via le système Androïde

Soutenu publiquement

Le :08/06/2015

Devant le jury :

M^{lle} **ZIDANI Ghania**

MA (A) Président

UKM Ouargla

M^r **REZOUG Mohamed Redha**

MA (A) Encadreur/rapporteur

UKM Ouargla

M^r **BOUAKAZ Ouahid**

MA (A) Examineur

UKM Ouargla

Année universitaire 2014/2015

Dédicace

Ma Mère, Mon Père

*Affable, honorable, aimable : vous représentez pour moi
Le symbole de la bonté par excellence, la source de tendresse
et l'exemple du
dévouement qui n'a pas cessé de m'encourager et de prier
pour moi.*

Soyez sûrs que je continuerai mon chemin.

*Je vous dédie ce travail en témoignage de mon profond amour.
Puisse Dieu, le tout puissant, vous préserver et vous accorder santé,
longue vie et bonheur.*

A mes sœurs

A mes frères

*En témoignage de l'attachement, de l'amour et de
L'affection que je porte pour vous.*

A tous les membres de ma famille, petits et grands

A tous mes amis de proches :

*Veillez trouver dans ce modeste travail l'expression de mon
Affection*

ABDELBASSET

ABDELMOUMEN

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Remerciements

En préambule à ce mémoire

*Nous remerciant ALLAH qui nous aide et nous donne la patience et le courage
durant
ces longues années d'étude.*

*Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui
nous
ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire
ainsi qu'à la
réussite de cette formidable année universitaire.*

*Nous tenant à remercier sincèrement Mr **REZOUG MOHAMED REDHA**,
en tant que Encadreur, qui à
toujours montré à l'écoute et très disponible tout au long de la réalisation de ce
mémoire,
ainsi pour l'inspiration, ainsi à Mlle **ZIDANI GHANIA** et Mr **Bouakaze
OUAHID***

Aux membres de jury

*Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis,
qui nous ont toujours soutenue et encouragée
au cours de la réalisation de ce mémoire.*

Merci à tous

Liste des figures

Figure	Nom de figure	Numéro de page
Figure (I.1)	La carte Arduino UNO	05
Figure (I.2)	Microcontrôleur ATmega328	7
Figure (I.3)	Constitution de la carte Arduino UNO	10
Figure (I.4)	Interface IDE Arduino	11
Figure (I.5)	Paramétrage de la carte	12
Figure (I.6)	Les étapes de téléchargement du code	13
Figure (I.7)	Type de modules Bluetooth	15
Figure (I.8)	Module shield wifi	16
Figure (I.9)	Module XBee	16
Figure (I.10)	Capteur Arduino	16
Figure (I.11)	Moteurs électriques	17
Figure (I.12)	Afficheurs LCD	17
Figure (I.13)	Relais	18
Figure(II.1)	Schéma synoptique du dispositif	20
Figure (II.2)	Schéma synoptique de l'alimentation	21
Figure (II.3)	Schéma électrique de l'alimentation	22
Figure (II.4)	Schéma simplificateur de la configuration BT	23
Figure (II.5)	L298HN (H=Modèle Horizontal)	26
Figure (II.6)	Schéma de brochage L298	28
Figure (II.7)	Schéma de principe	28
Figure (II.8)	La carte réalisée sous ISIS-PORTEUS	29
Figure (II.9)	Le chemin du fichier de code HEX de notre programme	30
Figure (II.10)	Typon et circuit imprimé	31
Figure (II.11)	Schéma final du Epoxy	32
Figure (III.1)	Organigramme Screene 1	36
Figure (III.2)	Organigramme Screene 2	36
Figure (III.3)	Présentation graphique du programme	37
Figure (III.4)	Création de nouveau projet sur App Inventor	43
Figure (III.5)	Première interface de la création App Inventor	44

Figure (III.6)	Les composants graphiques	44
Figure (III.7)	Les propriétés et les classes sous App Inventor	45
Figure (III. 8)	Ajustement d'un bouton sous App Inventor	46
Figure (III. 9)	Schéma global du Screene 1	46
Figure (III. 10)	Schéma global du Screene 2	47
Figure (III. 11)	En-tête d'éditeur de blocks App Inventor	48
Figure (III. 12)	Editeurs de blocks App Inventor	49
Figure (III. 13)	Echantillon d'un composant sous App Inventor	50
Figure (III. 14)	Schéma global du Scratch 1	51
Figure (III.15)	Schéma global du Scratch 2	51
Figure (III. 16)	Photo réelle de notre application BT contrôle	52

Liste des figures	I
Sommaire	III
Introduction générale	
1 Généralités	01
2 Position du problème	01
3 Objectif du projet	01
4 Présentation du mémoire	02
CHAPITRE I	
Le dispositif programmable Arduino	
I.1 Introduction	03
I.2 Définition du module Arduino	03
I.3 Les gammes de la carte Arduino	03
I.4 Pourquoi Arduino UNO	05
I. 5 La constitution de la carte Arduino UNO	06
I.5.1 Partie matérielle	07
I.5.1.1 Le MicrocontrôleurATMega328	07
I.5.1.2 Les sources de l'Alimentation de la carte	08
I.5.1.3 Les entrées & sorties	08
I.5.1.4 Les ports de communications	10
I.5.2 Partie programme	11
I.5.2.1 l'environnement de la programmation	11
I.5.2.2 Structure générale du programme (IDE Arduino)	11
I.5.2.3 Injection du programme	12
I.5.2.4 Description du programme	12
I.5.2.5 Les étapes de téléchargement du programme	14
I.6 Les Accessoires de la carte Arduino	15
I.6.1 Communication	15
I.6.1.1 Le module Arduino Bluetooth	15
I.6.1.2 Le module shield Arduino Wifi	15
I.6.1.3 Le Module XBee	16
I.6.2 Les capteurs	16
I.6.3 Les drivers	17

I.7 Conclusion	18
----------------	----

CHAITRE II

Réalisation d'un dispositif expérimental

II.1 Introduction	19
II.2 Les différentes étapes de la réalisation	19
II.3 Schéma synoptique général	20
II.3.1 Bloc d'alimentation	20
II.3.2 Bloc de traitement et contrôle	22
II.3.3 Bloc de commande	22
II.3.4 Bloc de puissance	25
II.3.4.1 Etage 1	26
II.3.4.2 Etage 2	26
II.4 La réalisation virtuelle « PORTEUS »	28
II.5 La réalisation PCB	30
II.6 Composants utilisés	32
II.7 Conclusion	33

CHAPITRE III

Développement d'une application Androïde

III.1 Introduction	34
III.2 Les types de programmation	35
III.2.1 Présentation de l'organigramme IDE	35
III.2.1.1 Organigramme Arduino UNO	35
III.2.1.2 Présentation du programme IDE	38
III.2.1.3 Explication du programme	40
III.2.2 le système Androïde	40
III.2.2.1 L'outil App Inventor	41
III.2.2.2 Historique de logiciel App Inventor	41
III.2.2.3 Langage JAVA	41
III.2.2.4 Un commencement avec App Inventor	41
III.2.2.5 Structure d'une application App Inventor	42
III.3 Explication et démarche	51
III.4 Conclusion	51

Conclusion générale

1 Généralités	52
2 Problèmes rencontrés	52
3 Perspectives du projet	52

Annexes

Annexe A	53
Annexe B	55
Annexe C	56
Annexe D	60
Annexe E	61

Bibliographie

Introduction Générale

1 Généralités

La forte augmentation des ventes de smart phone et de tablettes électronique se fait en même temps qu'une adoption rapide par le grand public des technologies de la domotique ainsi que l'autopilotage. Au fond, le smart phone, avec sa connectivité Bluetooth intégrée, devient une télécommande universelle pour toute la maison et les équipements électriques. Les utilisateurs pourront à terme contrôler à distance un très grand nombre de fonctions sans avoir à tenir compte de la marque ou de l'origine du produit qu'ils pilotent. Pour répondre à cette évolution majeure, nous avons créé une carte qui permet de contrôler n'importe quel appareil, véhicule, machine à travers un smart phone ou à une tablette. Cette carte électronique réalisée rend la commande et le contrôle facile et souple lors du pilotage à distance.[06]

2 Position du problème

Dans la vie moderne, on utilise pas mal d'outils et d'accessoires de commande à distance afin de simplifier notre contrôle, donc nous chercherons toujours à se concentrer sur la souplesse de la commande et de contrôler sur une zone bien définie (notre contour) le plus grand nombre possible d'accessoires.

Le smart phone occupe la première place d'objets qui ne nous quittent pas donc notre travail se concentre sur l'utilisation de ce dernier avec bien sûr sa liaison avec un système ou une carte de commande (carte d'interface) telle que l'Arduino.

3 Objectif du projet

Dans ce projet trois objectifs ont été visés :

- Le premier est de regrouper suffisamment d'informations sur une grande catégorie de cartes d'interfaçage (Arduino) : son langage de programmation, sa construction, son principe de fonctionnement.
- Le deuxième consiste à réaliser une carte électrique capable d'exécuter une action entre un smart phone et une carte d'interfaçage (Arduino) en expliquant les différents blocs de sa construction.
- Le troisième est de réaliser une application sous smart phone et de la programmer sous l'environnement JAVA afin de simplifier la commande et de montrer l'intérêt de la programmation orientée objet.

4 Présentation du mémoire

Le premier chapitre sera consacré à une étude approfondie sur les carte d'interface tel que l'Arduino, puis, on mettra la lumière sur un modèle de base qui est (Arduino UNO) sa construction son environnement de programmation et son principe de fonctionnement afin de simplifier son utilisation.

Le deuxième chapitre sera consacré à l'étude et la réalisation du dispositif expérimental ainsi que la description de chaque bloc du circuit.

Dans le troisième chapitre, on présentera un algorithme; on expliquera le fonctionnement de notre programme, son déclenchement à l'action à distance et cela sur deux modes de commande (domotique et l'autoguidage) après le chargement du code simulé dans la carte Arduino et de réaliser une application capable de gérer une telle commande sous smarte phone.

Enfin, on terminera avec une conclusion générale qui résumera l'intérêt de notre étude : les différents résultats obtenus expérimentalement seront donnés sur annexe (A) et (B) sous forme de photos, donnant ainsi un aperçu sur les performances des carte d'interfaçage, et une idée sur les problèmes à résoudre ultérieurement.

Chapitre I

Le dispositif programmable

Arduino

I.1 Introduction

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

Depuis que l'électronique existe, sa croissance est fulgurante et continue encore aujourd'hui. L'électronique est devenue accessible à toutes personnes en ayant l'envie : ce que nous allons apprendre dans ce travail est un mélange d'électronique et de programmation. On va en effet parler d'électronique embarquée qui est un sous-domaine de l'électronique et qui a l'habileté d'unir la puissance de la programmation à la puissance de l'électronique.

I.2 Définition du module Arduino

Le module Arduino est un circuit imprimé en matériel libre (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 comme l'électrotechnique industrielle et embarquée ; le modélisme, la domotique mais aussi dans des domaines différents comme l'art contemporain et le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur, commander des appareils mobiles (modélisme). Chaque module d'Arduino possède un régulateur de tension +5 V et un oscillateur à quartz 16 MHz (ou un résonateur céramique dans certains modèles). Pour programmer cette carte, on utilise l'logiciel IDE Arduino. [3]

I.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 d'USB pour programmer et usage d'un ATmega8.
- L'extrémité d'Arduino, avec une interface d'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 programme à l'aide porte USB cette version utilisant un microcontrôleur ATmega168 (ATmega328 pour une plus nouvelle version).
- Le LilyPad Arduino, une conception de minimaliste pour l'application wearable en utilisant un microcontrôleur ATmega168.
- Le NG d'Arduino plus, avec une interface d' 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 Diecimila, avec une interface d'USB et utilise un microcontrôleur ATmega168.
- L'Arduino Duemilanove ("2009"), en utilisant un microcontrôleur l'ATmega168 (ATmega328 pour une plus nouvelle version) et actionné par l'intermédiaire de la puissance d'USB/DC.
- L'Arduino Mega, en utilisant un microcontrôleur ATmega1280 pour I/O additionnel et mémoire.
- L'Arduino UNO, utilisations microcontrôleur ATmega328.
- L'Arduino Mega2560, utilisations un microcontrôleur ATmega2560, et possède toute la mémoire à 256 KBS. Elle incorpore également le nouvel ATmega8U2 (ATmega16U2 dans le jeu de puces d'USB de révision 3).
- L'Arduino Leonardo, avec un morceau ATmega32U4 qui élimine le besoin de raccordement d'USB et peut être employé comme clavier.
- L'Arduino Esplora : ressemblant à un contrôleur visuel de jeu, avec un manche et des sondes intégrées pour le bruit, la lumière, la température, et l'accélération. [4]

Parmi ces types, nous avons choisi une carte Arduino UNO (carte Basique). L'intérêt principal de cette carte est de faciliter la mise en œuvre d'une telle commande qui sera détaillée par la suite.

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 de 1 kilo. Elle est dotée de 14 entrées/sorties digitales (dont 6 peuvent être utilisées en tant que sortie PWM), 6 entrées analogiques et un cristal à 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 ci-dessous. [1]



Figure I.1 La carte Arduino UNO

I.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 à 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 complété par des programmeurs 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 Commons, 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.[20]

I.5 La 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 programmeur dédié ne soit pas nécessaire.

I.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.

I.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. la figure I.2 montre un microcontrôleur ATmega 328, qu'on trouve sur la carte Arduino.[1]



Le composant CMS



Le composant classique

Figure I.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 sur :

- Ñ **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).
- Ñ **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.
- Ñ **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]

I.5.1.2 Les sources de l'alimentation de la carte

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

- Ñ **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 le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.
- Ñ **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.
- Ñ **3V3**. Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de votre ordinateur et le port série de l'ATmega) de la carte est disponible : 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]

I.5.1.3 Les entrées & sorties

Cette carte possède 14 broches numériques (numérotée de 0 à 13) 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):

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` (ou TWI - Two-Wire interface - interface "2 fils").
- **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 6 entrées analogiques (numérotées de 0 à 5), chacune pouvant fournir une mesure d'une résolution de 10 bits (càd 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]

I.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) transmettre (les données séries de niveau TTL). Ces broches sont connectées aux broches correspondantes du circuit intégré ATmega328

programmé en convertisseur USB – vers – 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:

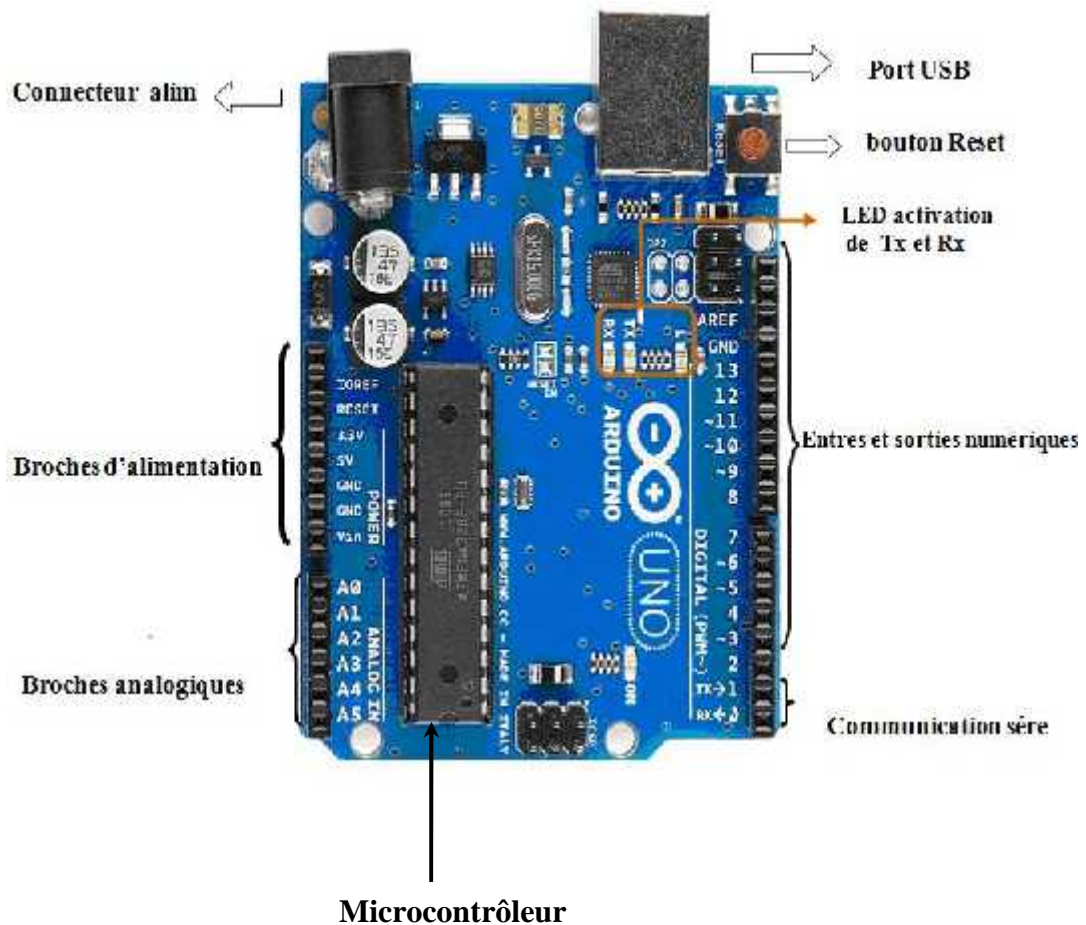


Figure I.3 Constitution de la carte Arduino UNO

I.5.2 Partie programme

Une telle carte d'acquisition qui se base sur sa construction sur un microcontrôleur doit être dotée d'une interface de programmation comme 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).

I.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 appelé IDE Arduino. [7]

I.5.2.2 Structure générale du programme (IDE Arduino)

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

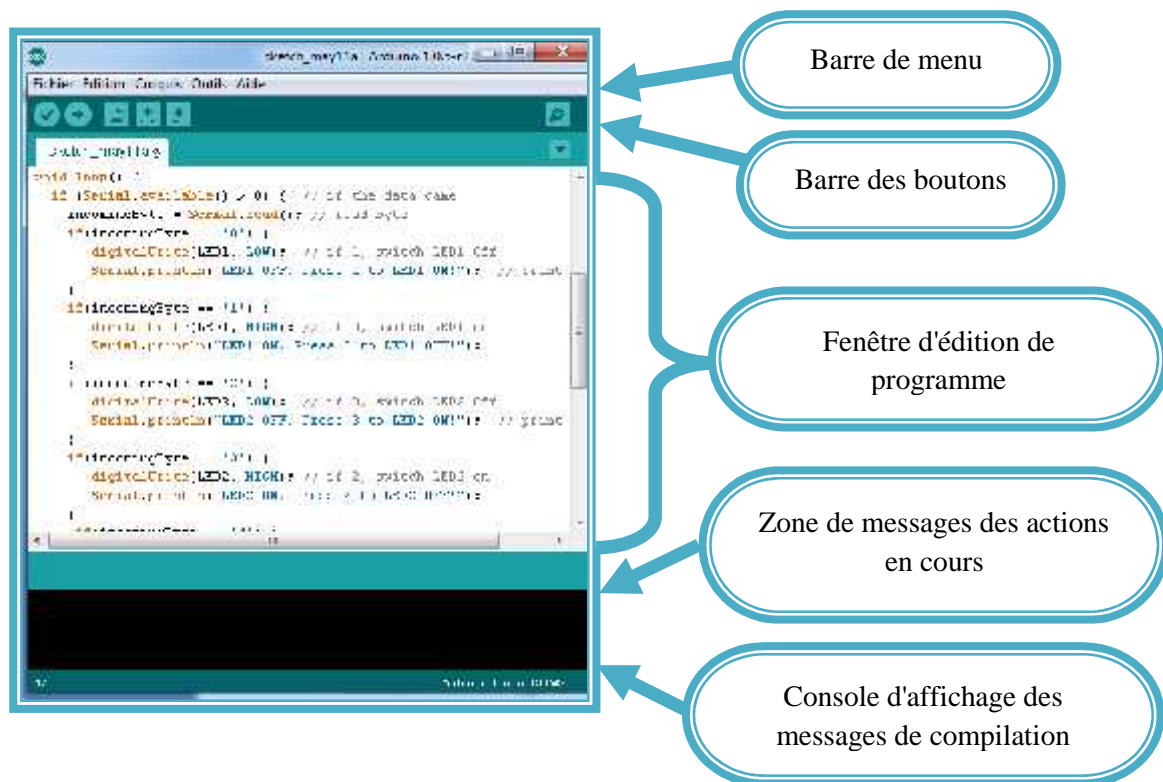


Figure I.4 Interface IDE Arduino

I.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) comme à titre d'exemple cette figure suivante.

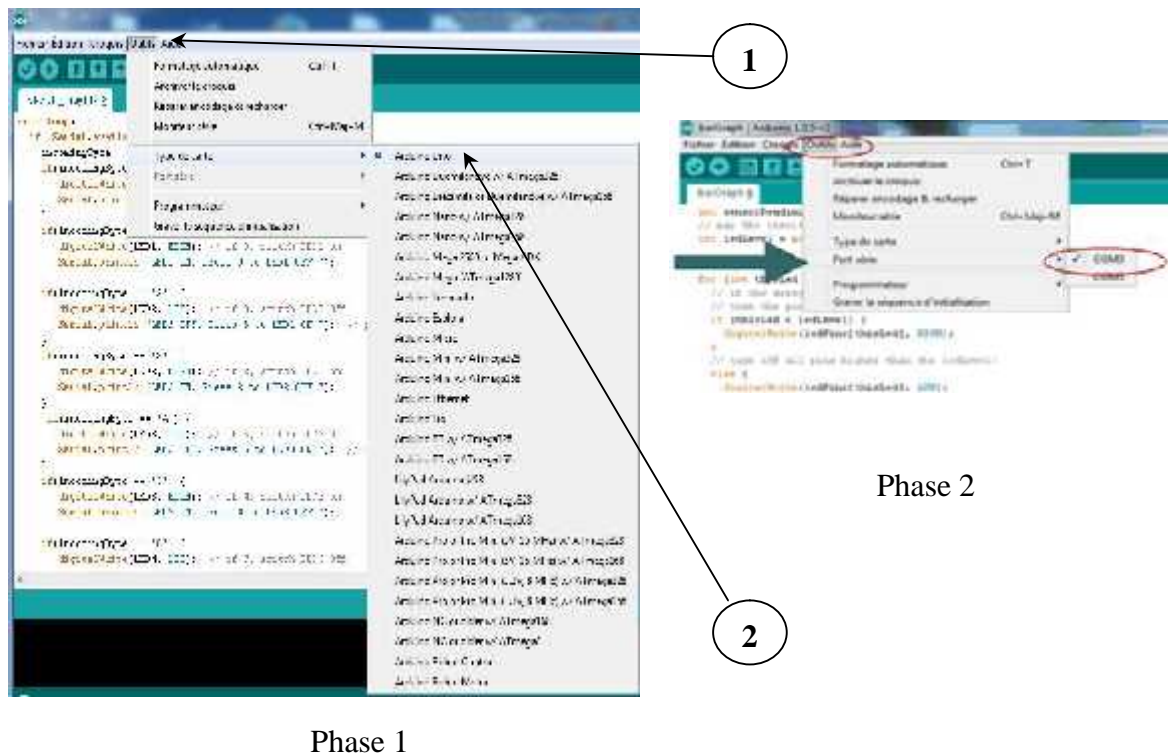


Figure I.5 Paramétrage de la carte

I.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 après 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 command DC moteur avec Smartphone via Bluetooth-----
- 2 *et fait également clignoter la diode de test de la carte-----
- 3 */-----

Définition des variables

Pour notre montage, on va utiliser une sortie numerique de la carte qui est par exemple la 3 éme sortie numerique ; cette variable doit être définie et nommée ici moteur pin 3 ; la syntaxe est pour désigner un nombre entier est **int**.

```
4 int moteur 1 = 3; // mettre le moteur au pin 3-----
```

Configuration des entres et des sorties void setup ()

les broches numeriques de l'arduino peuvent aussi bien être configurées en entrées numeriques ou en sorties numeriques; ici on va configurer moteur pin en sortie ; pin mode (nom,état) est une des quatre fonctions relatives aux entrées – sorties numériques.

```
5 void setup() {-----
```

```
6 // mettre le moteur 1 comme sortie:-----
```

```
7 pinMode(motor 1, 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 **digital write** (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.
- ne pas oublier les accolades qu' encadrent la boucle.

```
9 void loop() {-----
```

```
10 digital write ( moteur 1,HIGH); -----
```

```
11 delay (3000)-----
```

```
12 digital Write(moteur 1, LOW);-----
```

```
13 delay (1000)-----
```

```
14 }-----
```

I.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 modifie 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.

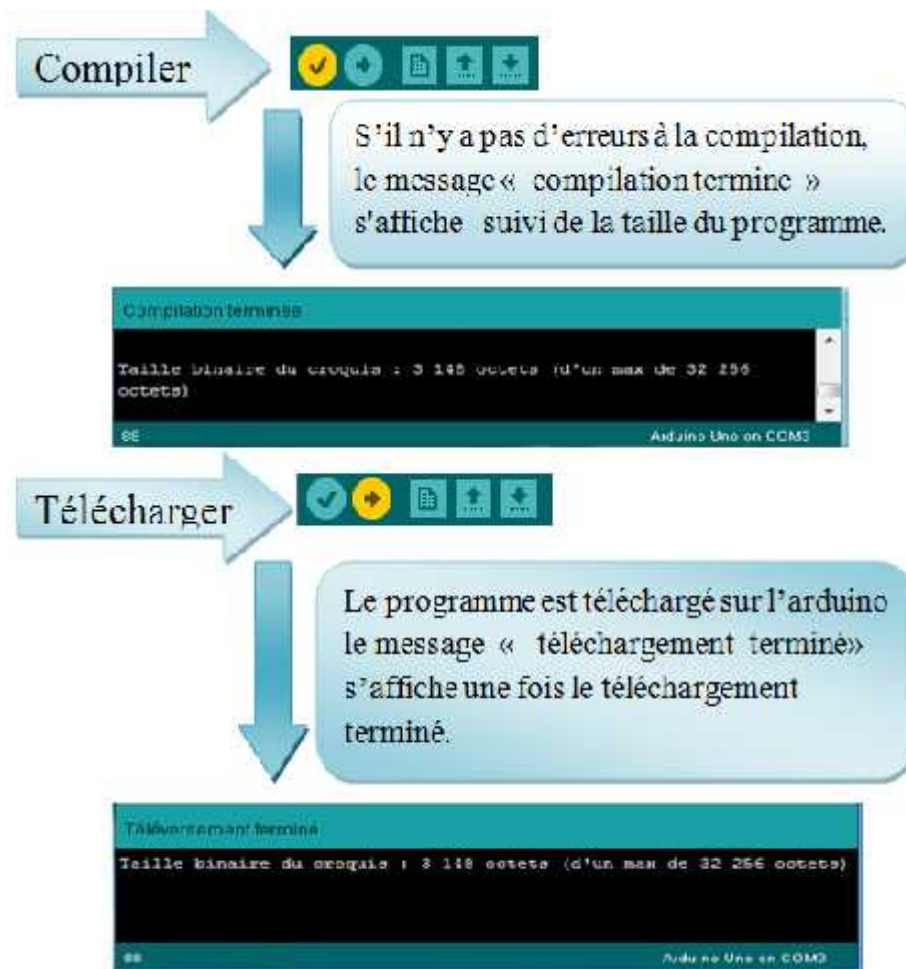


Figure I.6 Les étapes de téléchargement du code

I.6 Les Accessoires de la carte Arduino

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

I.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.

I.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.

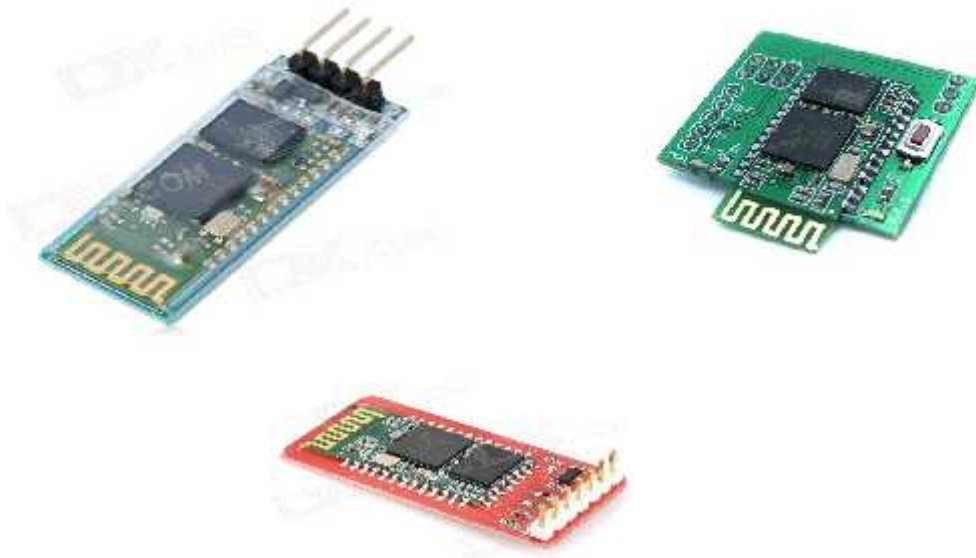


Figure I.7 Type de modules Bluetooth

I.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 I.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]

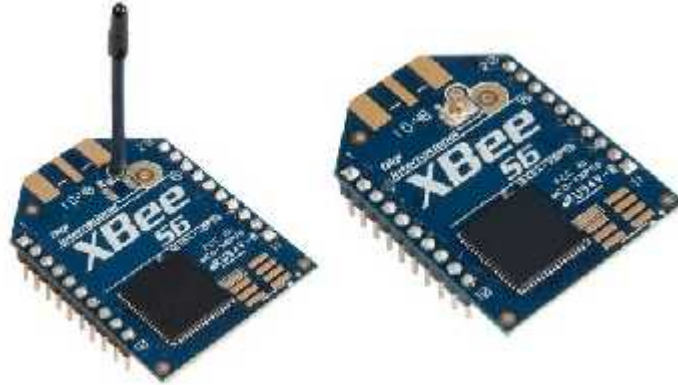


Figure I.9 Module XBee

I.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 I.10 Capteur Arduino

I.6.3 Les drivers

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

1 Des moteurs électriques



Figure I.11 Moteurs électriques

2 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]

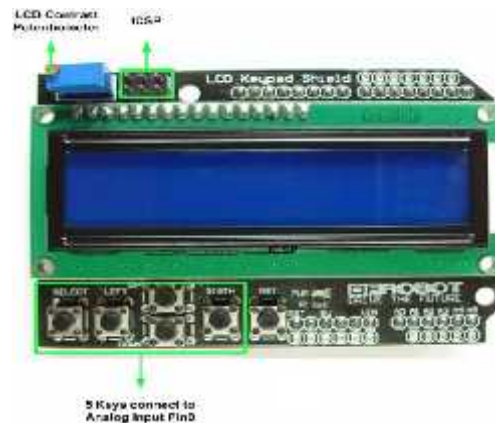


Figure I.12 Afficheurs LCD

3 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 oeuvre un mouvement mécanique. [10]



Figure I.13 Relais

I.7 Conclusion

Dans ce chapitre, nous avons projeté la lumière sur une carte d'acquisition qui est l'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 l'étude et la réalisation d'un dispositif de commande domotique comme une première phase et une deuxième phase de commande à distance des véhicules gérés par un smart phone.

Chapitre II

Réalisation d'un dispositif

Expérimental

II.1 Introduction

Dans ce chapitre, on présentera de manière sommaire une vue d'ensemble du dispositif expérimental « réalisation d'une carte électronique commandée par Arduino et contrôlée par un Smart phone.

Ce travail d'initiation à base d'une carte Arduino UNO permet de commander une maison intelligente (la domotique) et autoguidage d'un véhicule à distance (contrôle deux moteurs dans les deux sens de rotation) via système Bluetooth.

Après avoir donné dans le chapitre précédent une description théorique sur le module Arduino et son environnement de développement, on va procéder à l'application expérimentale ; pour cette raison, plusieurs blocs ont été nécessaires afin de réaliser une telle combinaison.

II.2 Les différentes étapes de la réalisation

Notre réalisation pratique a été faite en trois parties:

- La première partie est la conception de tout le système électronique.
- La deuxième partie est la réalisation pratique de la carte.
- La troisième partie est la création d'une application androïde. (voir chapitre III)

La première partie de notre projet est très importante, on est passé par plusieurs étapes:

1. Chercher les différentes structures des blocs constituant notre maquette et qui vont avec les objectifs fixés et les moyens disponibles.
2. Présenter les différents éléments ou composants constituant chacun des blocs en choisissant des composants aux caractéristiques voulus, à défaut, on choisira ceux disponibles sur le marché.

Dans la deuxième partie « réalisation pratique », on passe par les deux étapes suivantes :

1. Présenter les différentes étapes de la réalisation pratique de la carte.
2. On assemble ensuite les composants suivant notre montage sur le circuit imprimé, commençant par l'alimentation générale de notre dispositif.

Dans la troisième partie « la création d'une application androïde », on passe par les deux étapes suivantes :

1. Création de l'interface de notre application androïde.
2. Programmation orientée objet de notre application (programmation par bloc).

II.3 Schéma synoptique général

Le schéma synoptique général de notre dispositif est indiqué par la figure (II.1). Pour faciliter cette étude, on a décomposé le schéma synoptique en quatre blocs :

- Bloc d'alimentation;
- Bloc de traitement et contrôle;
- Bloc de commande ;
- Bloc de puissance;

En ce qui concerne l'élément principal de ce dispositif, notre choix était le dispositif Arduino UNO.

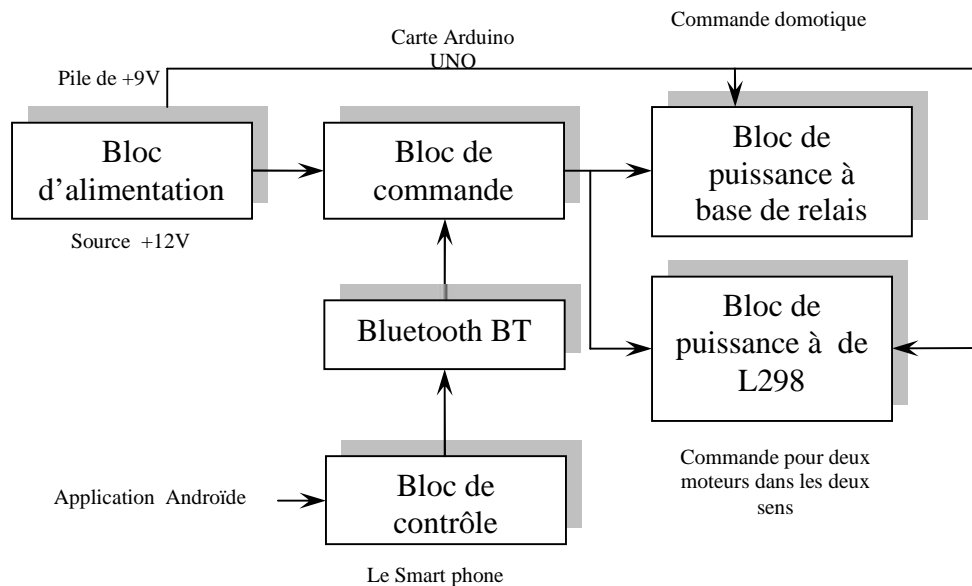


Figure II.1 Schéma synoptique du dispositif

La réalisation de notre système est représentée à la fin de ce chapitre et son circuit imprimé est illustré dans l'annexe (A).

II.3.1 Bloc d'alimentation

L'ensemble des dispositifs « pile de +9 volts et source externe, résistances, condensateurs, relais, Arduino UNO, transistors, photo-coupleurs » exige dans la plupart des cas, des alimentations stabilisées de (+5V) , alors dans notre travail, nous avons réalisé une alimentation

simple à partir d'une alimentation externe de ($<+12V$) avec une régulation et filtrage comme il est indiqué sur la figure (II.2).

Nous avons réalisé une alimentation continue de +5 volts Pour l'alimentation de l'Arduino, et une autre alimentation de +12 volts pour l'alimentation de l'étage de puissance.

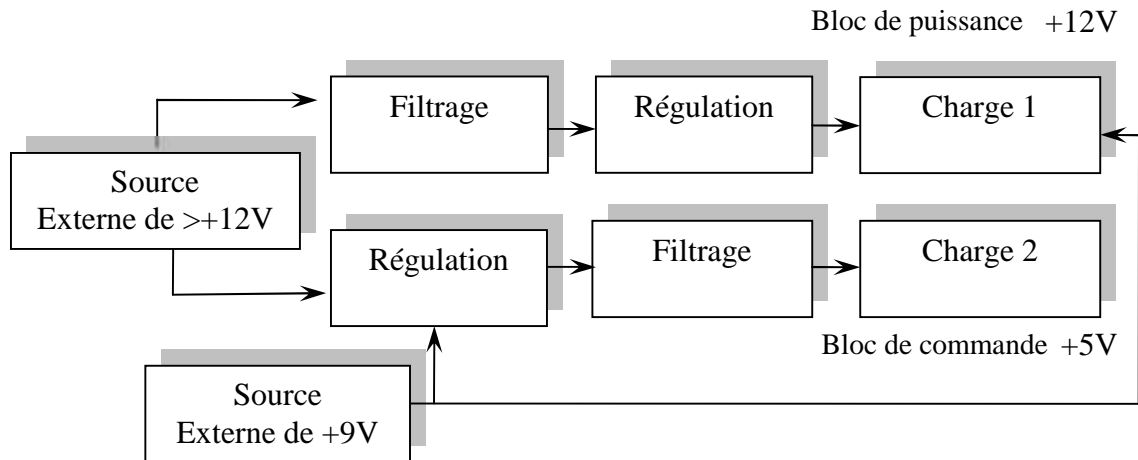


Figure II.2 Schéma synoptique de l'alimentation

Pour la réalisation électrique de ce schéma synoptique «l'alimentation du bloc de commande», figure (II.3), on a utilisé:

- Une résistance 10 k ohms « pour protection » ;
- Des condensateurs « pour filtrage ». condensateur de $15\mu F$, tous deux placés avant le régulateur;
- Un régulateur « baisse la source d'entrée de $>+12V$ à 12 nous avons utilisé le LM7812.
- Un régulateur « baisse la source d'entrée de +12V à +5V ; nous avons utilisé le LM7805.

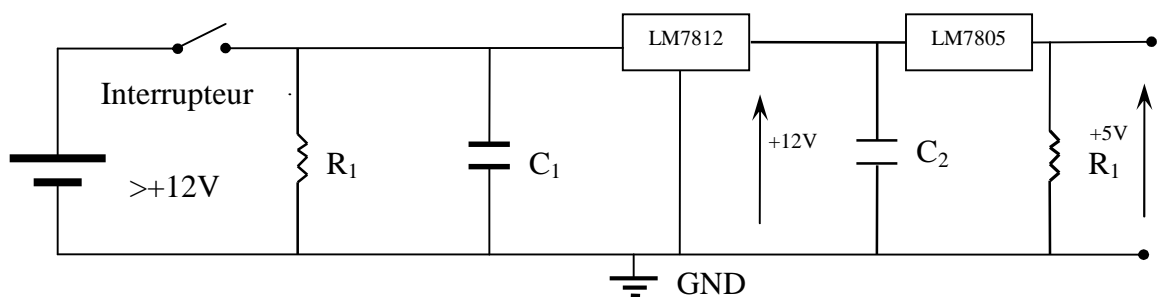


Figure II.3 Schéma électrique de l'alimentation

Le principe de fonctionnement d'une alimentation est très simple. Il consiste à générer ou réguler +5 volts à partir d'une source continue (+12V) ; cette dernière est filtrée via un filtre passif et régulée par un régulateur type (LM7805) à la sortie duquel, on aura une tension continue d'une valeur de +5 volts.

II.3.2 Bloc de traitement et contrôle

Le bloc concerné est le Smart phone, nous allons réaliser dans le chapitre qui suit une application à deux Screens dont : la première est capable de gérer une commande directe vers six relais (la commande domotique) et la deuxième Screen gère la commande de deux moteurs (autoguidage) et ce à travers le bluetooth.

II.3.3 bloc de commande

Notre bloque de commande, on le résume tout simplement par l'utilisation du module Arduino UNO qui est détaillé précédemment au chapitre I.

Dans cette partie, on peut classer l'accessoire Bluetooth de l'Arduino comme une suite de bloque de commande puisque il prend la relève de la validation des signaux émis par le smart phone vers l'arduino ; donc, il a besoin d'une configuration lors de la programmation. Nous avons utilisé un modèle nommé **HC-05**.

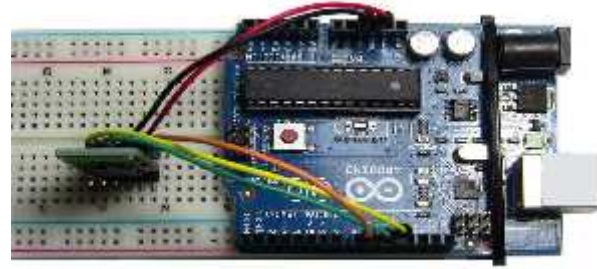
Avant de passer à la configuration du bluetooth, il faut réaliser quelques étapes et ce comme suit :

- Enlever toutes les communications ;
- Placer la position de fonctionnement **DAT** de commutateur ;
- Relier le port d'USB à l'ordinateur ;
- Télécharger le programme ou ouvrir une terminale série ;
- Employer ce mode pour vérifier ou changer la configuration de bluetooth : le mot de passe PIN, la vitesse baud, le mode de maître / esclave ;

La configuration d'un Bluetooth module de HC-05, peut être réalisé par l'intermédiaire de la terminale série aux commandes. (On peut utiliser IDE Arduino moniteur de série, sous menu " Tool / Serial Monitor", ou juste, cliquer sur CTRL+M du clavier du PC).



Partie IDE



Partie matérielle

Figure II.4 Schéma simplificateur de la configuration BT

Dans le tableau qui suit, nous allons citer les plus importantes commandes que nous avons utilisé pour configurer le module de Bluetooth HC-05.

Commande	Fonction
AT	Ne fait rien mais la réponse "OK" signifie bon essai pour connaître le mode de CMD.
AT+VERSION?	donner la version Bluetooth HC-05.
AT+ORGL	Restaurer les paramètres par défaut: 1.type de module : 0 2.code nécessaire : 0x009e8b33 3. Mode de travail: mode esclave 4.mode de connexion: connecter le module de Bluetooth spécifier 5.paramètres série: Baud rate: 38400 bits/s; Stop bit:1bit; bit de Parité: Non. 6.mot de passe : "1234" 7.le nom de périphérique "HC-2010-06-01".
AT+NAME?	Pour donner le nom du dispositif bluetooth.
AT+NAME=MyName	Placer le nom avec 'MyName'
AT+ROLE?	Donner le rôle.

AT+ROLE=rôle- numéro	Placer le nombre de rôle. 0---- le rôle esclave 1---- le rôle mètre 2---- le rôle esclave-boucle Défaut: 0
AT+PSWD?	Le mot de passe du Bluetooth à 4 chiffres
AT+PSWD= passcode	Placer le mot de passe du Bluetooth de 4 chiffres
AT+UART?	Pour donner la vitesse de baud, le bit de parité et le bit d'arrêt.
AT+UART= Bit de parité, bit d'arrêt.	Placer la vitesse de baud, avec bit de parité et bit d'arrêt.. les valeurs (Décimal) 4800 9600 19200 38400 57600 115200 23400 460800 921600 1382400 Paramètre de stop bit: 0----1 bit 1----2 bits Param3: bit de parité AT+UART? +UART=vitesse , stop-bit , parité-bit 0---- Aucun 1---- Imparité 2---- parité égale Défaut: 9600, 0, 0

Tableau II.1 Tableau de commande BT

NB : Réglez la vitesse à 38400 de transfert des données c'est le paramètre le plus essentiel dans la configuration du BT.

II.3.4 Bloc de puissance

Notre bloc de puissance se compose de deux sous blocs :

- Etage des relais (Screene 1) la commande domotique ;
- Etage de driver L298 (Screene 2) autoguidage.

II.3.4.1 Etage 1

Ce dernier se compose de six relais électromécaniques donc c'est un organe électrique permettant de dissocier la partie puissance de la partie commande. Chaque relais est excité dans sa bobine par un signal généré par l'arduino à travers un composant d'isolation (un optocoupleur) afin d'assurer une protection de la carte Arduino.

Ce bloc contient six LED qui s'allume et s'éteint à savoir le signal émis par le smart phone.

II.3.4.2 Etage 2

Dans cet étage nous avons utilisé un modèle de pont-H vraiment robuste, il existe le L298N (ou L298HN horizontal) qui peut tout à fait convenable, leur fiche technique est représentée dans l'annexe (C).



Figure II.5 L298HN (H=Modèle Horizontal)

Les principes de montages du L298N sont rigoureusement identiques à ceux du L293E, ce qui facilite grandement son utilisation seul le brochage diffère, parmi les avantages qui nous ont laissés choisir ce composant :

- L298 supporte une tension de puissance jusqu'à 46 volts maximum ;
- Un courant de service de 2A. supporte les pics occasionnels à 3A et pics répétitifs a 2.5A ;
- Compatible TTL (peut donc être commandé directement avec Arduino) ;
- Dispose d'un dispositif de mesure du courant (sensor/ pins). A raccorder à la masse si on ne l'utilise pas ;
- Chute de tension **V_{ce_sat}** totale de 1.8 volts (typique), 3.2v à IL=1A, 4.9v à IL=2A. Si cela semble beaucoup, c'est aussi un avantage car cela permet d'utiliser un accu de 7.2v directement avec un moteur +5V. La chute de tension **V_{ce_Sat}** du Pont-H sera suffisante pour adapter la tension au moteur (généralement +/- 5V).

Ce composant dispose de 15 broches ; on peut éclaircir son brochage sous forme de tirets comme suit :

- **VSS** - Alimentation de la logique de commande (+5V). A raccorder à la borne +5V d'Arduino (donc sur le régulateur d'Arduino) ;
- **VS** - Alimentation de puissance des moteurs ;
- **GND** - Doit être raccorder à la masse (GND) de la source d'alimentation de puissance VS (donc le négatif de l'accumulateur) et à la masse de la source d'alimentation de VSS (donc GND Arduino) ;
- **OUTPUT1, OUTPUT2** - Broches à raccorder à la charge (le moteur). Via ces broches que le L298 commande le sens de rotation du moteur.
- **INPUT1, INPUT2** - Broches de commande du Pont-H. Se raccorde a Arduino.
- **ENABLE A** - (Chip Activer) permet d'envoyer (ou pas) la tension sur les sorties du moteur via OUTPUT1 & OUTPUT2. **ENABLE. A** commande l'activation du premier Pont-H. Si **ENABLE A** = GND, le pont-H est déconnecté et le moteur ne fonctionne pas. Si **ENABLE A** = VSS, le pont-H est connecté aux sorties et le moteur fonctionne dans un sens ou l'autre ou pas en fonction des tensions appliquées sur INPUT1 & INPUT2.

- CURRENT SENSING A & CURRENT SENSING B** - permet de faire une mesure du courant dans le circuit de puissance. A placer impérativement sur GND si cette fonctionnalité n'est pas utilisée.

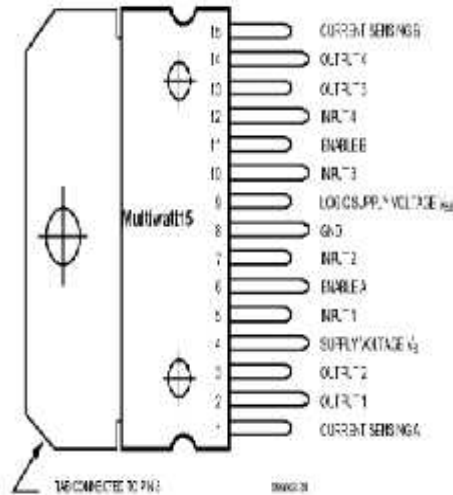


Figure II.6 Schéma de brochage L298

Le principe de montage de ce driver est similaire à celui du L293E ; on peut le présenter comme le montre cette figure.

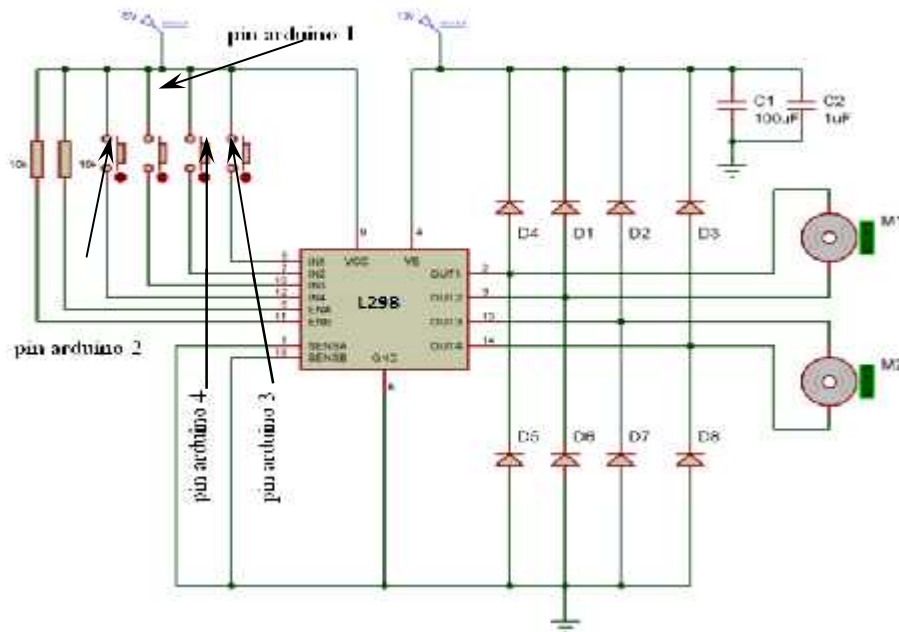


Figure II.7 Schéma de principe

II.4 La réalisation virtuelle « PORTEUS »

Avant de passer à la réalisation pratique, nous avons utilisé un CAO: il s'agit de ISIS-PORTEUS, c'est un CAO électronique perfectionné conçu par Labcenter Electronics qui permet de dessiner des schémas électroniques, de les simuler et de réaliser le circuit imprimé correspondant. Le CAO électronique « PROTEUS » » disponible et téléchargeable sur ce lien [21], se compose de nombreux outils regroupés en modules au sein d'une interface unique.

Ce dernier nous permet de schématiser notre carte électrique et la simuler virtuellement comme le montre la figure suivante :

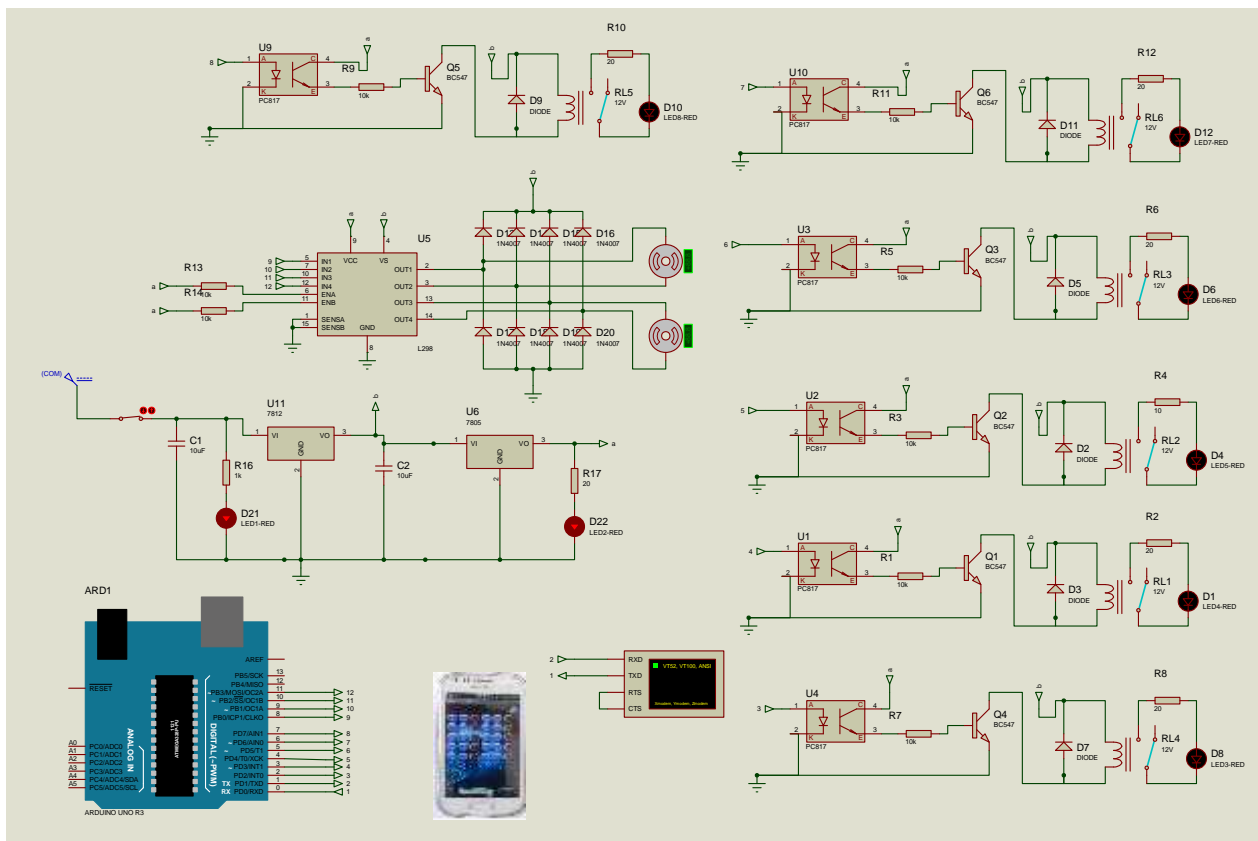


Figure II.8 La carte réalisée sous ISIS-PORTEUS

Ce CAO a la possibilité d'emporter même des codes hexadécimaux pour les réalisations qui contiennent des composants programmables ou des cartes programmables « Arduino » comme dans notre réalisation.

Le code hexadécimal de notre programme est représentée dans l'annexe (E).

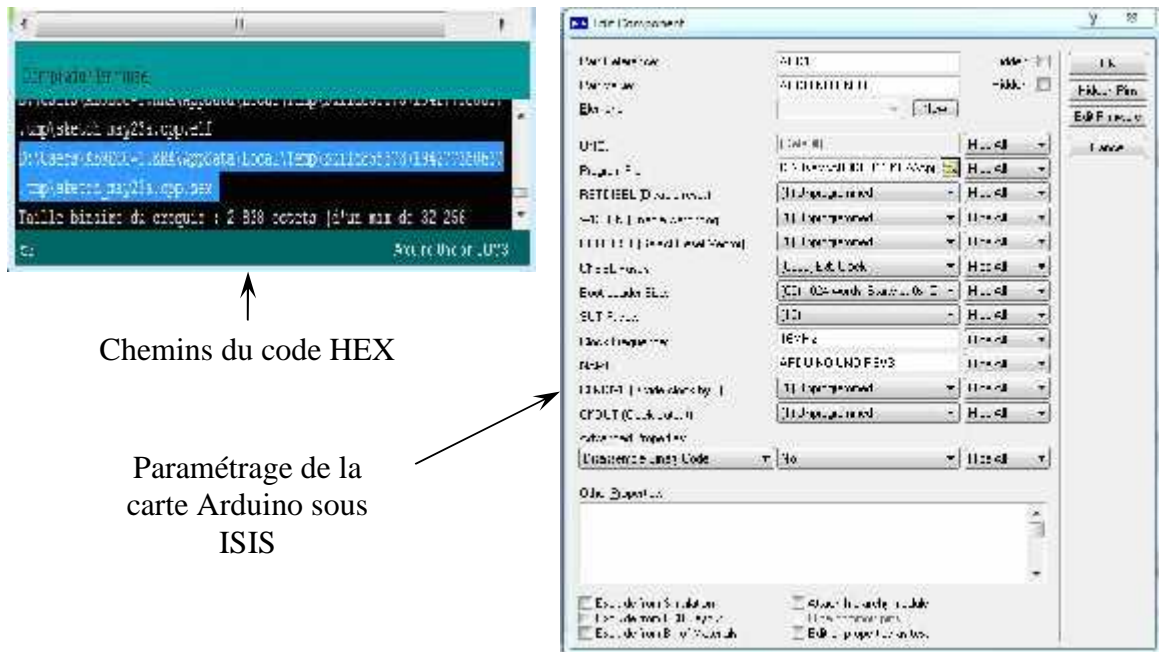


Figure II.9 Le chemin du fichier de code HEX de notre programme

La simulation de notre dispositif sous ISIS prouve la réussite, ce CAO possède même un virtuel USB qu'on peut utiliser comme un bluetooth (un terminal USB).

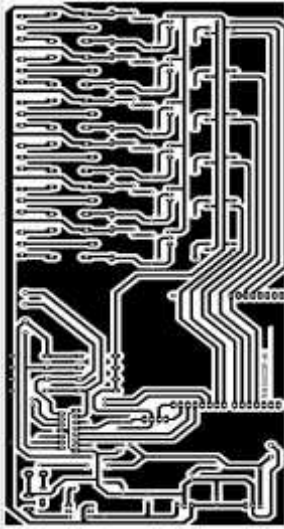
II.5 La réalisation PCB

Le typon est en effet le négatif de la partie cuivrée du circuit imprimé. Il s'agit donc de créer les liaisons entre les différents composants utilisés « résistances, condensateurs, relais, Arduino, transistors, photo-coupleurs etc », avec des pistes en cuivre sur le circuit imprimé. De ce fait, on utilise un CAO simple à manipuler : il s'agit d'un logiciel nommé « EAGEL » disponible et téléchargeable sur ce lien [22] pour tracer un tel typon voir annexe (A).

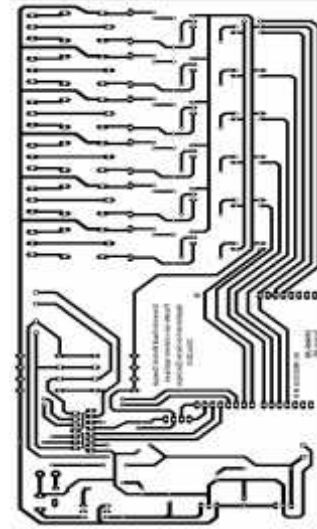
Pour réaliser notre carte électronique, nous allons passer par plusieurs étapes.

EAGLE nous permet de dessiner des schémas de haute qualité avec la possibilité de contrôler parfaitement l'apparence du dessin tout au long du processus d'édition: largeurs de lignes, styles de remplissage, couleurs et polices, etc.

Cette photo nous montre le circuit imprimé et le schéma électrique imprimés sur une feuille glacée.



Typon négatif

Typon avec
composant

Typon positif

Figure II.10 Typon et circuit imprimé

Pour réaliser une telle carte nous avons passé par plusieurs étapes

- Découper la plaque Epoxy /cuivre aux dimensions du futur circuit imprimé 10*23.4 cm ;
- Mettre le typon dans l'insoleuse en vérifiant bien son sens ;
- Pour que l'insolation puisse s'effectuer, il faut retirer l'adhésif opaque qui protège la couche photosensible des rayons ultra-violets de la lumière ambiante ;
- placer la plaque pré sensibilisée dans l'insoleuse, avec le typon côté cuivre une fois l'adhésif retiré ;
- L'étape qui suit présente « Une gravure chimique » dans laquelle est utilisé le produit attaquant le cuivre « le perchlorure de fer » ;

Cette figure regroupe ces étapes :

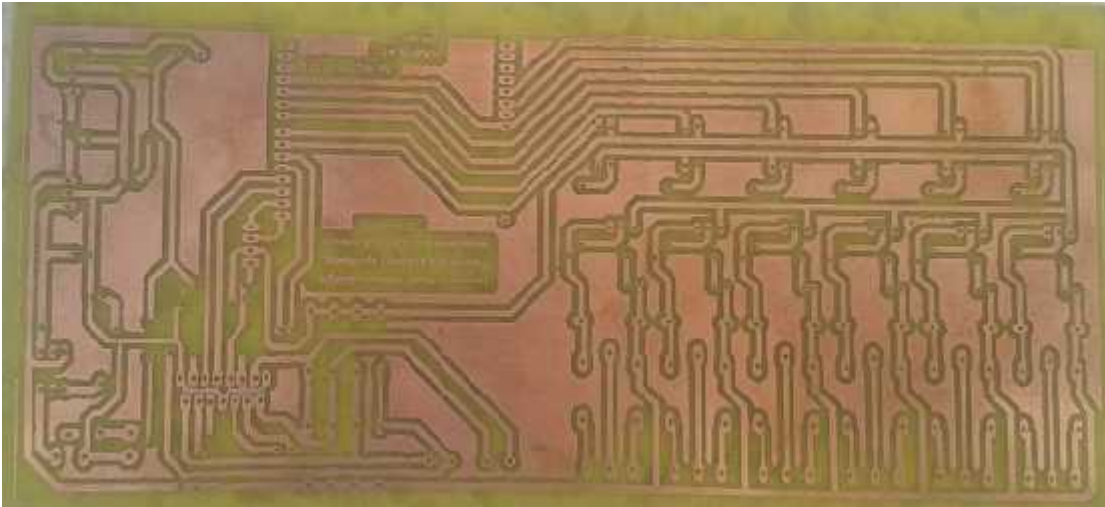


Figure II.11 Schéma final du Epoxy

- Enfin, on finalise notre réalisation par une opération « la soudure » est cela après le perforage des vias où on implante les différents composants sur leurs emplacements.

II.6 Composants utilisés

Pour une telle réalisation, nous avons assemblé les différents composants:

- Une Carte Arduino UNO ;
- Un Bluetooth module de HC-05 ;
- 8 LEDs ;
- 6 photo-coupleurs ;
- 16 résistances (470 ohms, 1Kohms, 10 Kohms) ;
- 2 capacités 15uf pour filtrage ;
- 1 L298 ;
- 9 diodes de protection 1N4007 ;
- Régulateur LM 7805 ;
- Régulateur LM 7812 ;
- Une pile 9V ;
- Fiche jack.
- Une plaque d'époxy de 10 * 23.4 cm ;

La photo globale de notre réalisation est mentionnée sur annexe (B)

II.7 Conclusion

Cette partie a donné lieu à la réalisation pratique d'une carte électronique pour commander une maison intelligente et une voiture qui possède deux moteurs. Ces commandes sont gérées à partir d'un smart phone (application Androïde) et une carte Arduino UNO. L'ensemble des travaux décrits dans ce chapitre est :

- La conception des composants électroniques qui compose la carte réalisée à l'aide logiciel (ISIS- PROTUSE), puis dessine le typon de notre carte à l'aide logiciel EAGALE.
- Le choix des composants convenables pour chaque bloc.
- L'identification des paramètres des circuits d'alimentation, circuit de commande et circuit de puissance.

Cette carte électronique est destinée directement aux fabricants de produits électriques et/ou électroniques afin qu'ils rendent de manière très simple leur produit pilotage à distance.

L'organigramme, le déroulement des différentes étapes du programme et la réalisation de l'application androïde seront présentés dans le chapitre suivant.

Chapitre III

Développement d'une

Application androïde

III.1 Introduction

L'Androïde est parmi les derniers systèmes d'exploitation qui développent les exigences des téléphones intelligents. La plateforme androïde de smart phone devient de plus en plus importante pour les réalisateurs de logiciel, en raison de ses puissantes possibilités et open source.

Lors des années précédentes, le traitement des données informatiques se fait par des ordinateurs ; en revanche le smart phone a des avantages qui ont les mêmes fonctions que l'outil informatique ; ce dernier porte l'intérêt de l'ordinateur grâce à l'androïde.

La téléphonie mobile a connu une explosion dans les années 2000 mais aucune révolution n'a semblé arriver depuis que les appareils se ressemblent. Les innovations n'avaient plus vraiment de saveur ; les applications étaient difficiles d'accès de par leur mode de distribution et souvent peu performantes à cause des faibles capacités des appareils.

Depuis quelques mois, les smart phones sont dotés d'une puissance plus importante et d'espaces de stockages conséquents. Les téléphones tendent à devenir des objets artistiques, presque de reconnaissance sociale, et possèdent des fonctionnalités qu'aucun téléphone ne pouvait espérer auparavant: connexion haut débit, localisation GPS, boussole, accéléromètre, écran tactile souvent multipoint, marché d'applications en ligne. Autant de qualités permettant de créer des applications innovantes et de les distribuer en toute simplicité.

La plate-forme Androïde apporte tout cela au consommateur, mais surtout, elle affranchit le développeur de nombreuses contraintes. Par son ouverture; elle permet à n'importe quel développeur de créer ses applications avec un ticket d'entrée quasi nul. Le framework et le système d'exploitation et outils associés ont un code source ouvert, leur accès est gratuit et illimité. Plus besoin de négocier avec le constructeur du téléphone pour qu'il vous laisse développer sur sa plate-forme. Tous les développeurs sont ainsi sur un même pied d'égalité, tous peuvent ajouter de la mobilité à des applications existantes.

Cette partie de notre étude ne nous donnera pas de bons résultats si on néglige certains paramètres ; donc le bon fonctionnement de notre système se base essentiellement sur une bonne démarche et une bonne réflexion de notre programme.

III.2 Les types de programmation

Du premier chapitre et du second, on peut combiner que notre réalisation software a besoin de deux étapes : la première consiste à un programme qui va s'injecter aux microcontrôleurs de la carte Arduino après avoir été convertie par le IDE en code HEX et la deuxième a un programme qui va se manipulé sous App Inventor et s'installé sous smart phone.

III.2.1 Présentation de l'organigramme IDE

L'organigramme est une représentation schématique des liens fonctionnels, organisationnels et hiérarchiques d'un organisme, d'un programme, etc. Il se doit de posséder une référence documentaire.

III.2.1.1 Organigramme Arduino UNO

Avant de passer à la programmation, nous devons réaliser un organigramme qui explique le déroulement des différentes séquences, tant intérieures qu'extérieures : il comportera plusieurs boucles dont la fin d'exécution succède toujours à son commencement.

Le programme principal

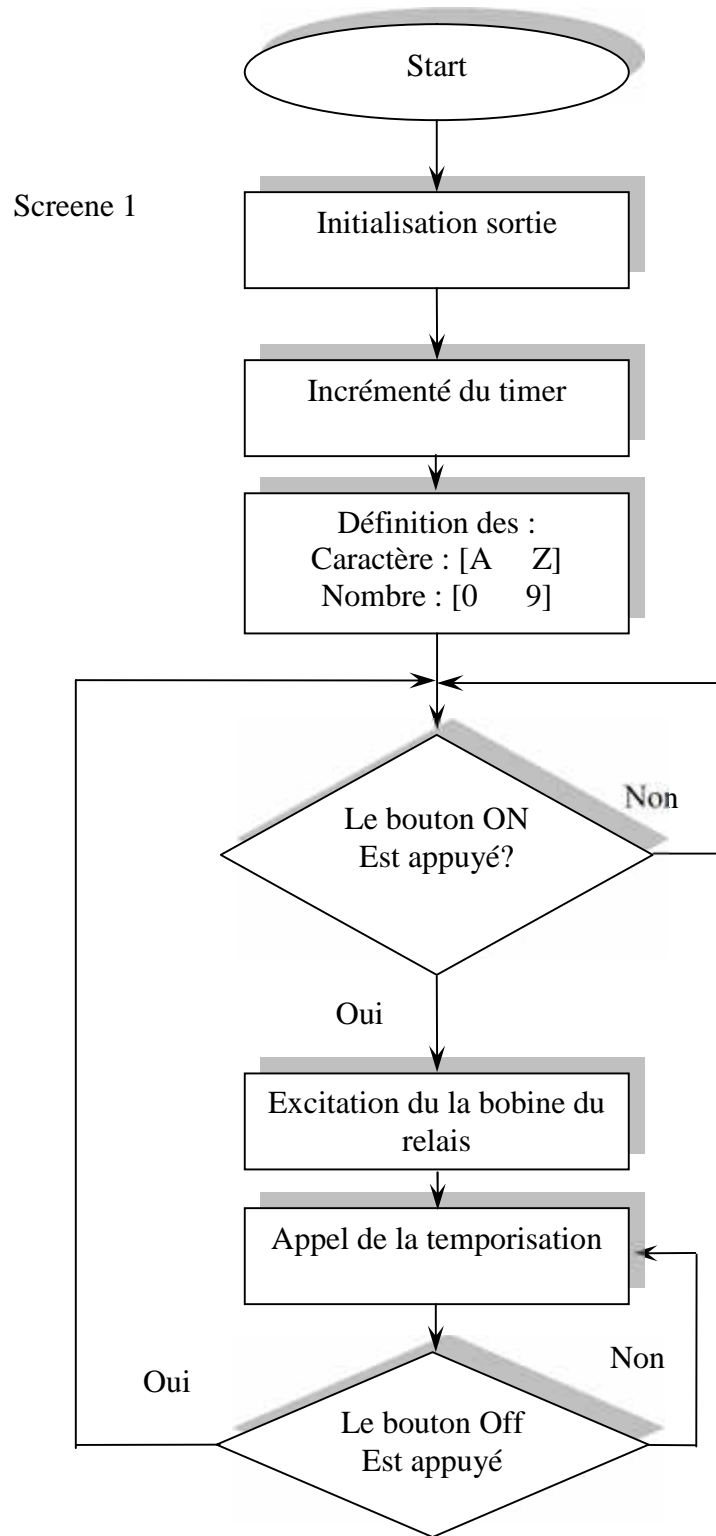


Figure III.1 Organigramme Screene 1

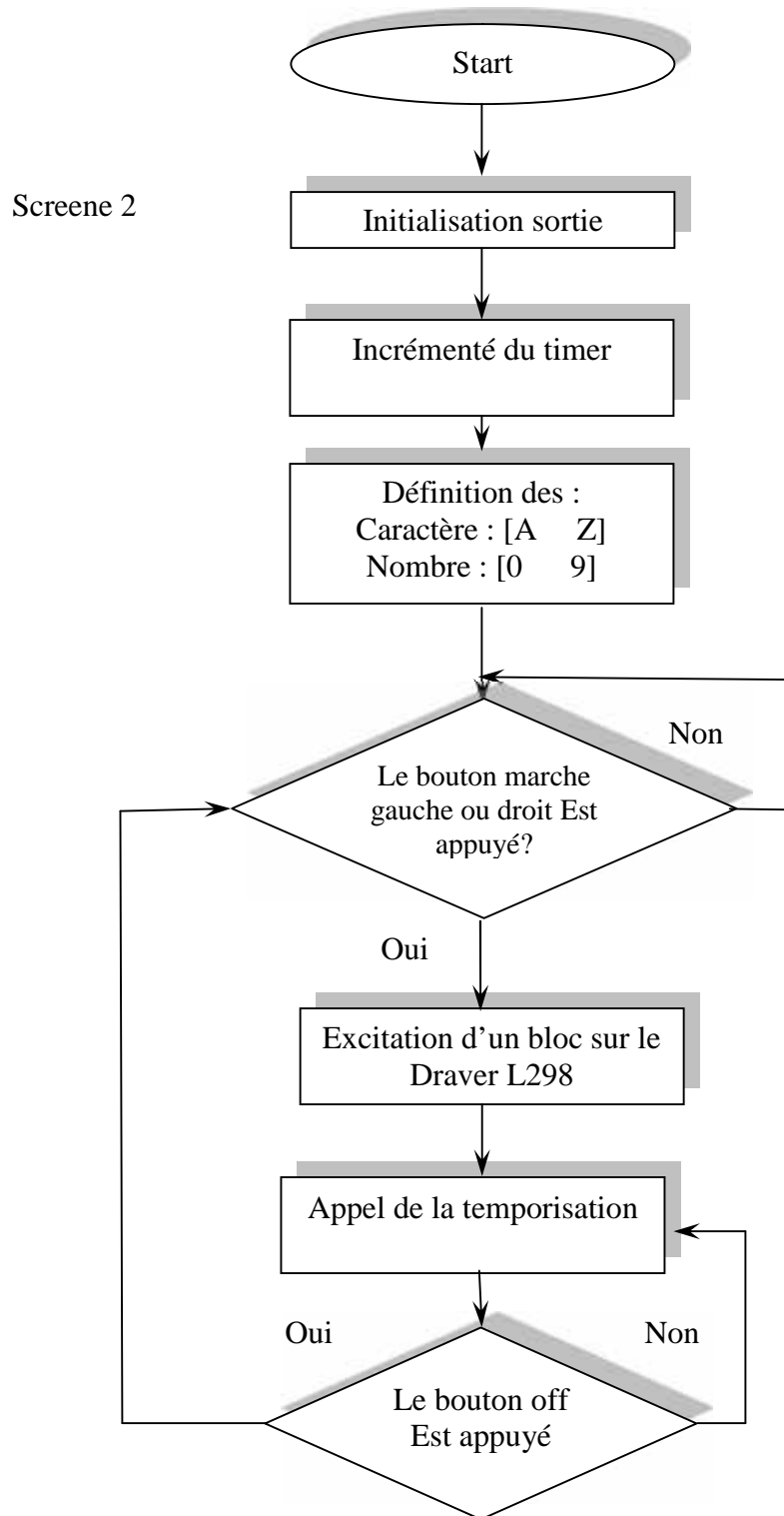
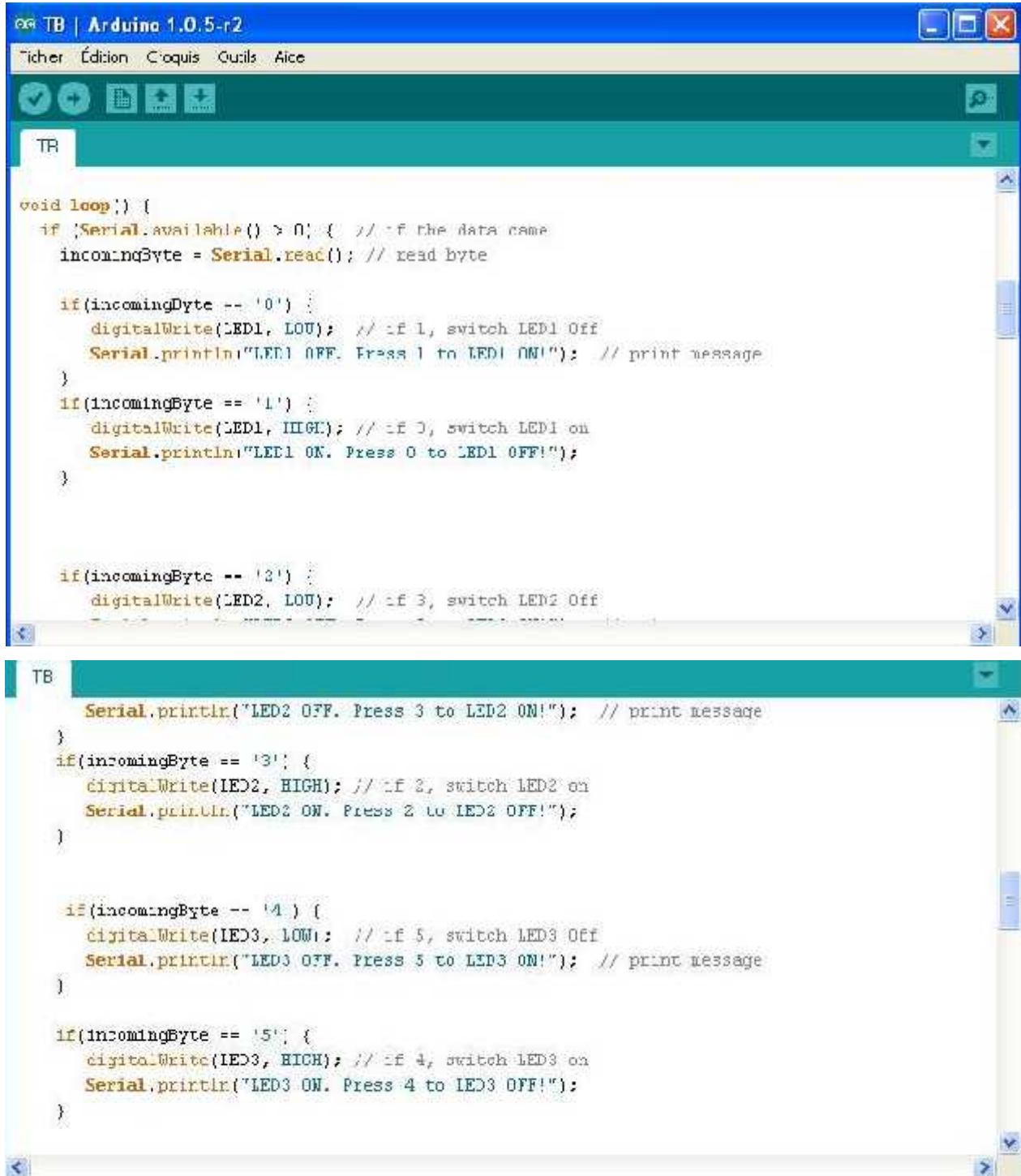


Figure III.2 Organigramme Screen 2

Chaque action est symbolisée par un rectangle et chaque test est symbolisé par un losange.

III.2.1.2 Présentation du programme IDE

Nous avons préféré de photographier l'interface du IDE de l'environnement Arduino dans lequel nous avons simulé notre programme.



```
Arduino 1.0.5-r2
Fichier Édition Croquis Outils Aide

void loop() {
  if (Serial.available() > 0) { // if the data came
    incomingByte = Serial.read(); // read byte

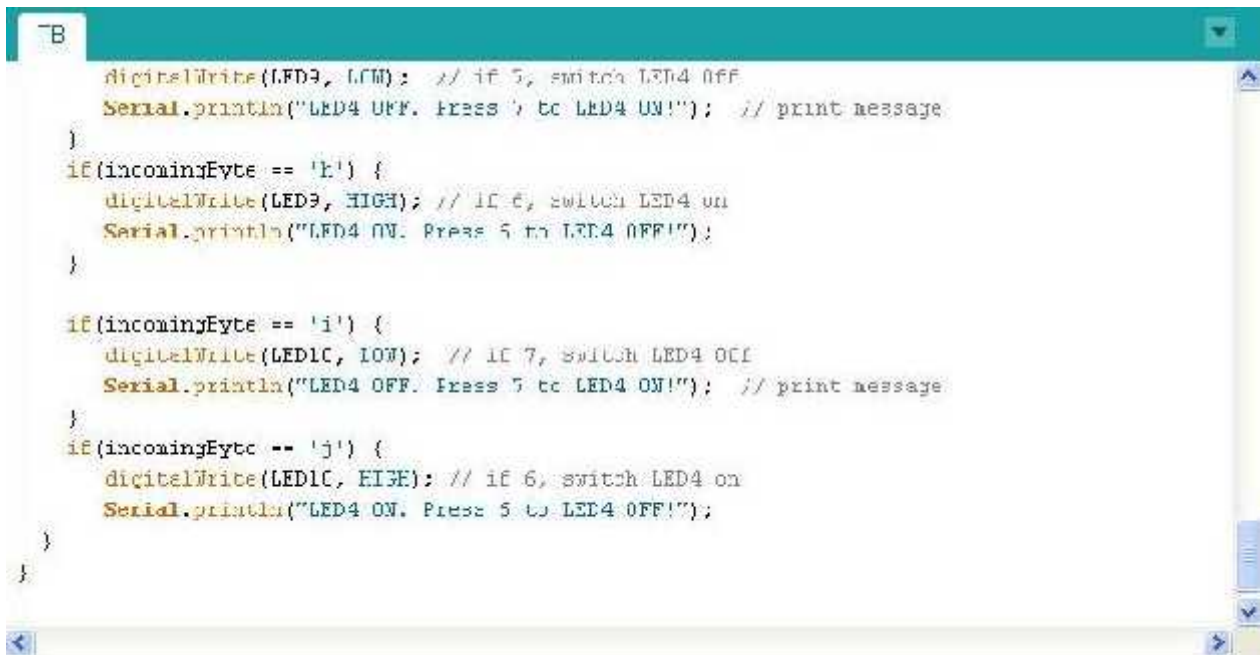
    if(incomingByte == '0') {
      digitalWrite(LED1, LOW); // if 1, switch LED1 Off
      Serial.println("LED1 OFF. Press 1 to LED1 ON!"); // print message
    }
    if(incomingByte == '1') {
      digitalWrite(LED1, HIGH); // if 0, switch LED1 on
      Serial.println("LED1 ON. Press 0 to LED1 OFF!");
    }

    if(incomingByte == '2') {
      digitalWrite(LED2, LOW); // if 3, switch LED2 Off
      Serial.println("LED2 OFF. Press 3 to LED2 ON!"); // print message
    }
    if(incomingByte == '3') {
      digitalWrite(LED2, HIGH); // if 2, switch LED2 on
      Serial.println("LED2 ON. Press 2 to LED2 OFF!");
    }

    if(incomingByte == '4') {
      digitalWrite(LED3, LOW); // if 5, switch LED3 Off
      Serial.println("LED3 OFF. Press 5 to LED3 ON!"); // print message
    }
    if(incomingByte == '5') {
      digitalWrite(LED3, HIGH); // if 4, switch LED3 on
      Serial.println("LED3 ON. Press 4 to LED3 OFF!");
    }
  }
}
```

```
TB
if(incomingByte == '6' ) {
    digitalWrite(LED4, LOW); // if 7, switch LED4 OFF
    Serial.println("LED4 OFF. Press 7 to LED4 ON"); // print message
}
if(incomingByte == '7' ) {
    digitalWrite(LED4, HIGH); // if 6, switch LED4 on
    Serial.println("LED4 ON. Press 6 to LED4 OFF!");
}
:
if(incomingByte == '8' ) {
    digitalWrite(LED5, LOW); // if 7, switch LED4 OFF
}
:
if(incomingByte == '9' ) {
    digitalWrite(LED5, HIGH); // if 6, switch LED4 on
}
:
if(incomingByte == 'a' ) {
    digitalWrite(LED6, LOW); // if 7, switch LED4 OFF
}
```

```
IB
}
if(incomingByte == 'b' ) {
    digitalWrite(LED6, HIGH); // if 6, switch LED4 on
}
if(incomingByte == 'c' ) {
    digitalWrite(LED7, LOW); // if 7, switch LED4 Off
}
if(incomingByte == 'd' ) {
    digitalWrite(LED7, HIGH); // if 6, switch LED4 on
}
if(incomingByte == 'e' ) {
    digitalWrite(LED8, LOW); // if 7, switch LED4 Off
}
if(incomingByte == 'f' ) {
    digitalWrite(LED8, HIGH); // if 6, switch LED4 on
}
}
```



```

digitalWrite(LED9, LOW); // if 5, switch LED4 off
Serial.println("LED4 OFF. Press 5 to LED4 ON!"); // print message
}
if(incomingByte == 'k') {
  digitalWrite(LED9, HIGH); // if 6, switch LED4 on
  Serial.println("LED4 ON. Press 5 to LED4 OFF!");
}

if(incomingByte == 'i') {
  digitalWrite(LED10, LOW); // if 7, switch LED4 off
  Serial.println("LED4 OFF. Press 5 to LED4 ON!"); // print message
}
if(incomingByte == 'j') {
  digitalWrite(LED10, HIGH); // if 6, switch LED4 on
  Serial.println("LED4 ON. Press 5 to LED4 OFF!");
}
}
}

```

Figure III.3 Présentation graphique du programme

III.2.1.3 Explication du programme

Le but de ce programme est de réaliser des actions « caractères » codées par des lettres ou par des chiffres émis de l'Arduino vers le smart phone ; chaque caractère active ou désactive une bobine sur la carte réalisée (commande domotique) ou faire tourner un de deux moteurs gauche ou droite et cela en agissant sur le driver L298. Chaque caractère est codé d'une façon simple et courte ; il comprend des déclarations décimales sur les lignes d'entrée du microcontrôleur.

On commence par la programmation du microcontrôleur situé sur l'Arduino ; au début, on déclare toutes les variables utilisées après on initialise les ports de l'Arduino comme des sorties « OUTPUTS », ensuite ; on appelle le sous programme de temporisation déjà situé sous le compilateur IDE. Le programme incrémente le code du caractère automatiquement afin de faire exécuter l'activation ou la désactivation d'une commande ou plus : soit sur le Screen 1 ou le Screen 2.

III.2.2 le système Androïde

Androïde est un système d'exploitation développé initialement pour les Smart phones. Il utilise un noyau Linux qui est un système d'exploitation libre pour PC et intègre tous les

utilitaires et les périphériques nécessaires à un smart phone. Il est optimisé pour les outils Gmail. Aussi, l'androïde est libre et gratuit et a été ainsi rapidement adopté par des fabricants.

La société Androide a été rachetée en 2007 par Google. Mais aujourd'hui, l'Androide est utilisé dans de nombreux appareils mobiles (smart phones). Les applications sont exécutées par un processeur de type ARM à travers un interpréteur JAVA. En plus de cela, l'androïde concurrence l'opérateur système d'Apple qu'il tend à dépasser en nombre d'utilisateurs. Androide évolue pour mieux gérer l'hétérogénéité des appareils qu'il utilise. [13]

III.2.2.1 L'outil App Inventor

App Inventor est un outil de développement des applications en ligne pour les smart phones sous androïde et permet à chacun de créer son application personnelle pour le système d'exploitation Androide qui est développée par Google.

La plateforme de développement est offerte à tous les utilisateurs possédant un compte Gmail. Elle rappelle certains langages de programmation simplifiés basés sur une interface graphique similaire à Scratch. Les informations des applications sont stockées sur des serveurs distants. [13] Elles sont actuellement entretenues par le Massachusetts Institute of Technologie (MIT).

L'environnement de App Inventor contient trois fenêtres qui sont proposées pendant le développement :

- Une pour la création de l'interface homme machine : permet de créer l'allure de l'application (App Inventor Designer) ;
- Une pour la programmation par elle-même : elle permet, par l'assemblage des blocs de créer le comportement de l'application (**App Inventor Block Editor**) ;
- Une pour l'émulateur : qui permet de remplacer un terminal réel pour vérifier le bon fonctionnement du programme.

La connexion d'un terminal réel sous Androide permettra ensuite d'y télécharger le programme pour un test réel. Ce terminal pourra aussi bien être un téléphone qu'une tablette ; le comportement du programme sera identique. [13]

N.B : Scratch est un environnement graphique permettant aux débutants de s'initier 'à la programmation. On y programme le comportement de lutins animés 'a l'aide de briques visuelles qui s'emboîtent comme des Legos par Drag and Drop.[16]

III.2.2.2 Historique de logiciel App Inventor

2009 : Début du développement du logiciel App Inventor par Google à partir de recherches dans l'enseignement de l'informatique faites par le MIT, Boston près de New-York. L'Objectif de l'enseignement permet à des étudiants qui débutent en informatique d'apprendre à programmer sans se noyer sous le code Java.

2011: Google rend App Inventor open source. Le MIT poursuit le développement.

2012: Version bêta d' App Inventor diffusé par le MIT encore en version bêta aujourd'hui.

III.2.2.3 Langage JAVA

JAVA est un langage de programmation orienté objet, développé par Sun Microsystems, sorti en 1995. Sun Microsystems est racheté en 2009 par Oracle, une application écrite en JAVA et facilement portable sur plusieurs systèmes d'exploitation.

Une application exécutable sous Androïde (interprétable par une interface en JAVA) est un fichier avec l'extension « APK » [14]

III.2.2.4 Un commencement avec App Inventor

Google fournit gratuitement un kit de développement (SDK) prévu pour s'intégrer (sous la forme d'un Plugin) à l'environnement de développement Eclipse (libre). Il permet de développer des applications codées en langage Java pour les différentes versions d'Androïde.

1. Se connecter à Internet.
2. Ouvrir notre navigateur et se connecter au compte Google.
3. Se connecter au site Internet d'App Inventor du MIT : [15]. Cela peut-être assez long parfois car App Inventor est lancée depuis le Web en mode Cloud.
4. Créer un nouveau projet :

Cliquer sur My Project (en haut à gauche) / New / Project Name (sans espace) / OK.



Notre application

Figure III.4 Création de nouveau projet sur App Inventor

Donc le projet est sauvegardé automatiquement sur notre compte Google. (Dans le Cloud).

III.2.2.5 Structure d'une application App Inventor

Une application développée sous App Inventor est constituée de deux parties distinctes mais étroitement liées.

A. L'interface graphique :

Pour créer l'application sous App Inventor l'interface graphique contient nos propriétés (taille, couleurs, position, textes Ets).



Figure III.5 Première interface de la création App Inventor

Cette partie contient des composants (visibles ou invisibles) [13]. Cette interface graphique contient quatre parties :

Partie 1 :

Une palette sous App Inventor contenant tous les éléments qui peuvent être positionnés sur l'écran du téléphone.



Figure III.6 Les composants graphiques

Partie 2 :

C'est la surface du téléphone ajusté automatiquement par app inventor ou manuellement par nous même en utilisant le composant « Screen arrangement ».

Partie 3 :

La liste des éléments et des médias utilisés sur l'écran.

Partie 4 :

Les propriétés des différents éléments utilisés par exemple la couleur et la taille du bouton ou texture.

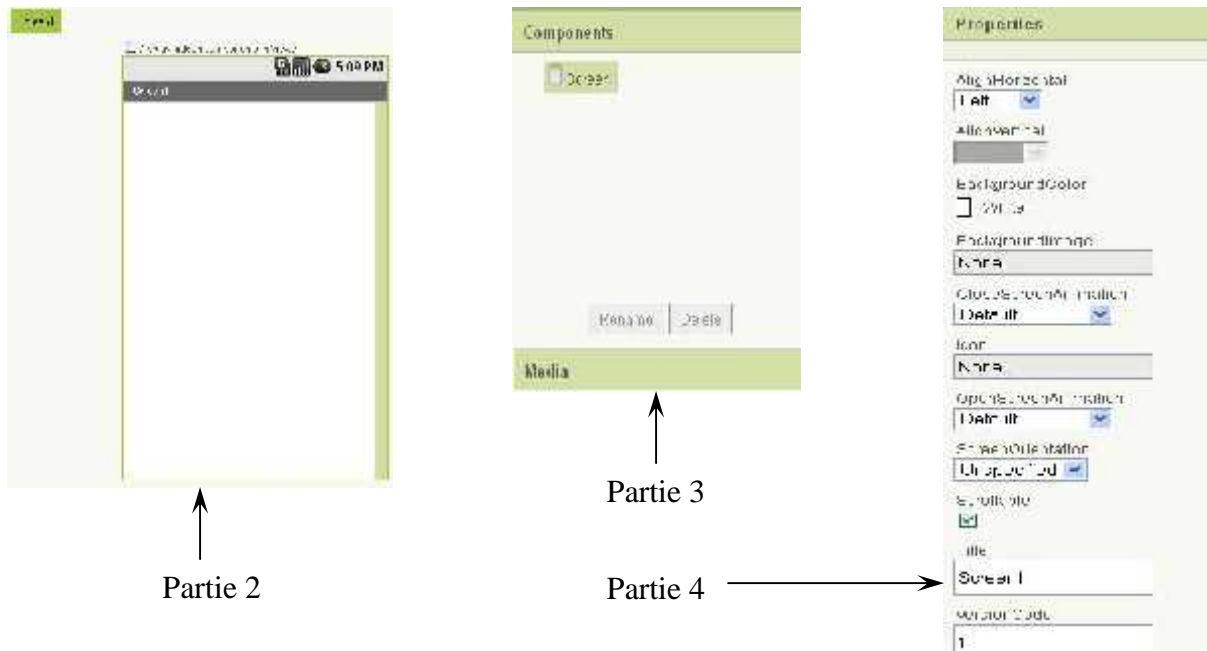


Figure III.7 Les propriétés et les classes sous App Inventor

Pour ajouter un bouton sur l'écran d'émulateur virtuel, on clique sans lâcher sur le mot « Bouton » dans la **palette** en déplaçant la souris sur **Viewer** et relâcher le bouton et là un nouveau bouton va apparaître sur le **Viewer**.

Pour ajuster la place du bouton, on utilise la propriété **Width** de notre exemple ; on utilise le paramètre **Fill parent**.

Pour utiliser le bouton sur la largeur de l'écran, on sélectionne les nombres de pixels de l'écran 50 pixels. De ces opérations, on obtient l'écran suivant :



Figure III.8 Ajustement d'un bouton sous App Inventor

Après l'assemblage des différents composants qui constituent notre application, on peut résumer cette phase « *Scratch* » par la figure qui suit et cela pour les deux Screen (1 et 2).



Figure III.9 Schéma global du Screen 1



Figure III.10 Schéma global du Screenshot 2

B. Éditeur de blocs (Fenêtre Scratch)

Une fois les composants de l'écran de téléphone mis en place et désigné, nous passons à la deuxième phase de développement d'une application via App Inventor : l'interface Scratch, pour cela, il faut cliquer sur «Open the Blocks Editor» en haut à droite de la page.

L'interface Scratch permet d'imbriquer des éléments graphiques entre eux pour effectuer la partie programmation de l'application à développer. De cette partie, on peut assembler les différents blocs de l'application et indiquer comment les composants doivent se comporter et qui s'affichent dans l'émulateur virtuel (par exemple ; ce qui se produit quand un utilisateur clique un bouton) déterminant le fonctionnement même de l'application, en réaction à des **événements** (internes ou externes) ou à des **réponses**. [13]

Pour ouvrir ce Block, on a cliqué en haut à gauche sur « Open the Blocks Editor ». App Inventor, lancé le Blocks Editor depuis le serveur du MIT et proposé l'exécution du fichier App Inventor For Androïde Code blocks, « d'un format *.Jnlp ». Après téléchargement, JAVA ouvre la fenêtre du Block Editor sur votre PC (cela peut être assez long).

N.B Jnlp:est un format de fichier associé à la technologie JAVA web start. Il s'agit de pouvoir déployer facilement des applications Java à partir d'un simple navigateur web, [17]

Comme indiqué sur la figure (III.11) App Inventor, éditeur de blocks, cette interface est très simple et épurée. En effet, en haut ; on retrouve des éléments classiques «Save», «Undo» et

«Redo» ainsi qu'un bouton de test pour lancer l'application sur le mobile ou sur l'émulateur. Cette interface contient des parties **Built-in** et **blocks**.



Figure III.11 En-tête d'éditeur de blocks App Inventor

La fenêtre à gauche contient les blocs qui assemblent la partie droite de la fenêtre pour décrire le comportement de notre application. Les blocs peuvent être standards (dans l'onglet « Built-in ») ou définis spécifiquement pour l'application (dans l'onglet « My Blocks »).

Aussi, la corbeille est utilisée pour jeter des morceaux de pseudo-code et la loupe sert à changer la taille l'écran de l'éditeur.

Chaque élément contient des blocks : (ces pages décrivent les blocs que nous pouvons utiliser lorsque nous construisons nos applications App Inventor.

Dans l'onglet « Built-in », nous retrouvons toujours les mêmes éléments :

- **Définition** : morceaux permettant de définir des procédures (avec/sans résultats/attributs).
- **Texte** : morceaux permettant de traiter le texte. Assimilables au type *char* et à la classe *String* en JAVA.
- **Lists** : morceaux permettant de traiter des listes. Assimilables aux sous-classes de *List* en JAVA.
- **Math** : morceaux permettant de traiter des nombres. Assimilables au type *int* et à la classe *Integer* en JAVA.
- **Logic** : morceaux permettant de traiter des booléens. Assimilables au type *boolean* et à la classe *Boolean* en JAVA.
- **control** : outils permettant d'effectuer de la programmation conditionnelle par exemple : dans l'élément logic.

Lorsqu' on clique sur «My Blocks» en haut et à droite de la page : nous obtenons la figure suivante :



Figure III. 12 Editeurs de blocks App Inventor

La palette des variables et fonctions est à gauche, l'onglet My blocks propose les fonctions associées aux éléments déposés sur notre écran au préalable

Dans l'onglet «My Blocks», on retrouvera les éléments et leurs accesseurs et fonctions :

- **My Definitions** : variables et procédures globales.
- **Button1** : variables et procédures spécifiques au bouton.
- **Label1** : variables et procédures spécifiques au label.
- **Orientation Sensor1** : variables et procédures spécifiques au capteur.
- **Screen1** : variables et procédures spécifiques à l'écran (assez restreint).

Lorsque on Clique sur l'élément « Button1 » on obtient la figure suivante est cela comme échantillon pour n'importe quel composant associé dans notre application.

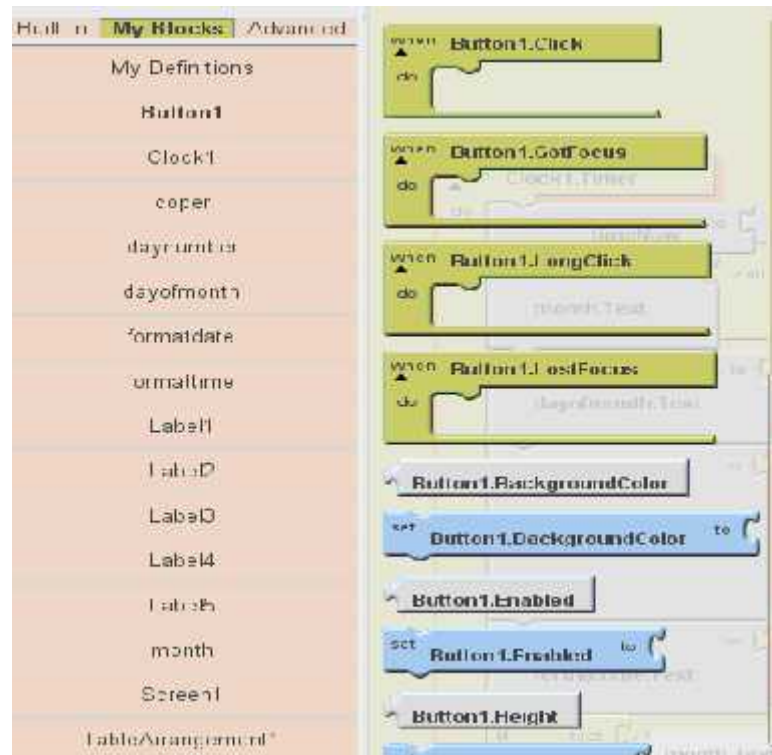


Figure III.13 Echantillon d'un composant sous App Inventor

A partir de l'onglet «**Package for phone**» ce test assure trois solutions accessibles :

1. L'émulateur : un écran s'affichera sur l'ordinateur.
2. la connexion se fera directement sur le smart phone en wifi.
3. USB : la connexion se fera sur le smart phone via un câble USB.

Pour installer notre application sur un appareil Androïde, on exécute une des trois solutions à partir du bouton "**Package for Phone**" (en haut à droite de l'interface web).

On résume notre application pour cette phase « *Scratch* » par la figure qui suit.



Figure III.14 Schéma global du Scratch 1

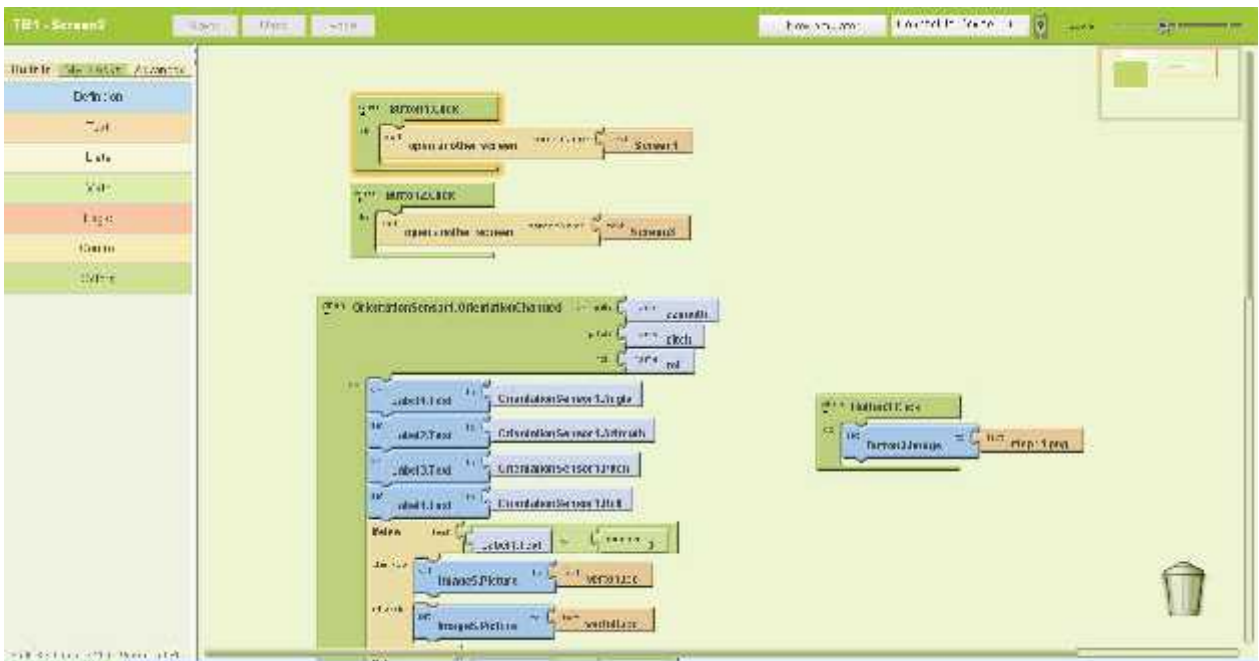


Figure III.15 Schéma global du Scratch 2

III.3 Explication et démarche

Notre interface contient une touche d'activation et désactivation du Bluetooth donc avant d'envoyer n'importe quel caractère à travers les boutons « ON », il faut d'abord s'assurer que le bluetooth est allumé ; un mot de passe pour raison de protection a été programmé afin d'exécuter un ordre quelconque. Cette figure montre que notre application nommée « BT contrôle » peut être installée sur smart phone, sa taille est de deux mégas comme première version.



Figure III.16 Photo réelle de notre application BT contrôle

III.4 Conclusion

Sur le plan pratique, une manipulation adéquate du logiciel « Arduino » nous permet alors d'utiliser un compilateur ; il s'agit de « IDE » ce dernier possède une capacité de créer un code HEX, qui peut être injecté sur son microcontrôleur d'une part ; d'une autre part une simple élaboration de l'environnement App Inventor qui nous pousse à réaliser une application « *.apk » sous smart phone capable de lier une carte Arduino et un smart phone afin d'exécuter des ordres bien définis. On peut conclure que les résultats obtenus sont satisfaisants compte tenu des limitations du matériel et des moyens dont nous disposons.

Conclusion Générale

1. Généralités

Le développement sous androïde se fait en JAVA. En effet, Google a porté un sous-ensemble de Java Standard (SE) ainsi qu'un certain nombre de projets open source dont il avaient besoin. Si vous êtes habitué à faire du JAVA, attendez-vous donc à ne pas retrouver la totalité des paquets que vous utilisez quotidiennement.

Afin de développer une application Androïde, vous devrez utiliser les composants mis à disposition par Google.

Le système Arduino est un outil pour fabriquer des dispositifs qui peuvent capter et contrôler davantage de choses du monde matériel que votre ordinateur. C'est une plateforme open-source d'électronique programmée 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.

la programmation du microcontrôleur AVR exige une connaissance en langage « C » comme soft et la maîtrise des outils « IDE », afin d'arriver à l'objectif souhaité.

Dans ce travail, un assemblage de plusieurs logiciels est fait afin de réaliser un tel résultat voir annexe (D).

2. Problèmes rencontrés

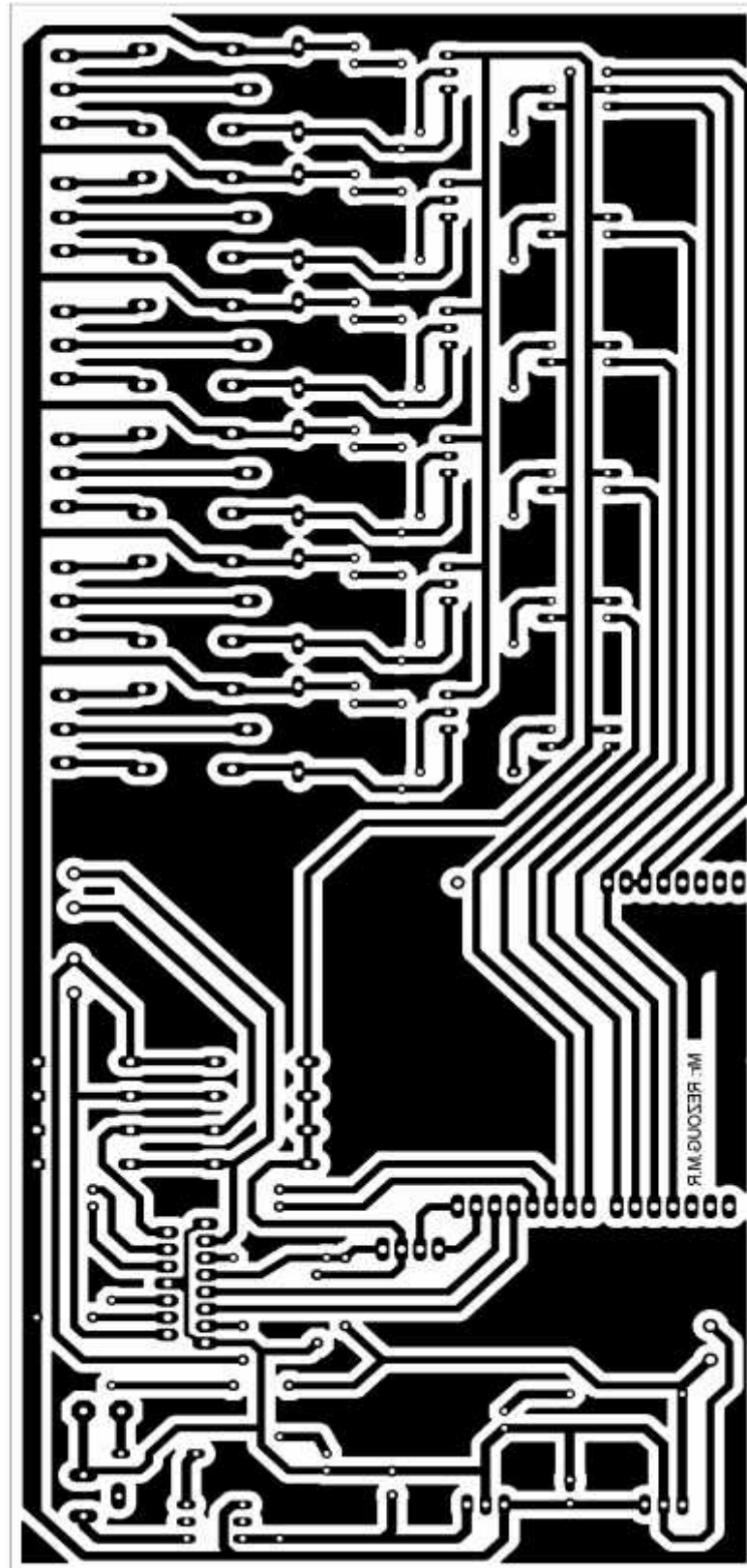
Une telle réalisation n'est pas dénuée de difficultés. Il est à noter que nous nous sommes confrontés à plusieurs problèmes surtout dans la partie réalisation de la carte.

Cependant, on peut dire que malgré ces difficultés, les résultats obtenus à travers cette étude qu'ils soient pratiques ou théoriques, permettent d'ouvrir la porte à d'autres études.

3. Perspectives du projet

Nous souhaitons vivement que ce projet puisse servir comme élément de base pour d'autres études plus approfondies pour le faire intégrer sous des systèmes plus complexes.

Annexes



Typon de la carte réalisée

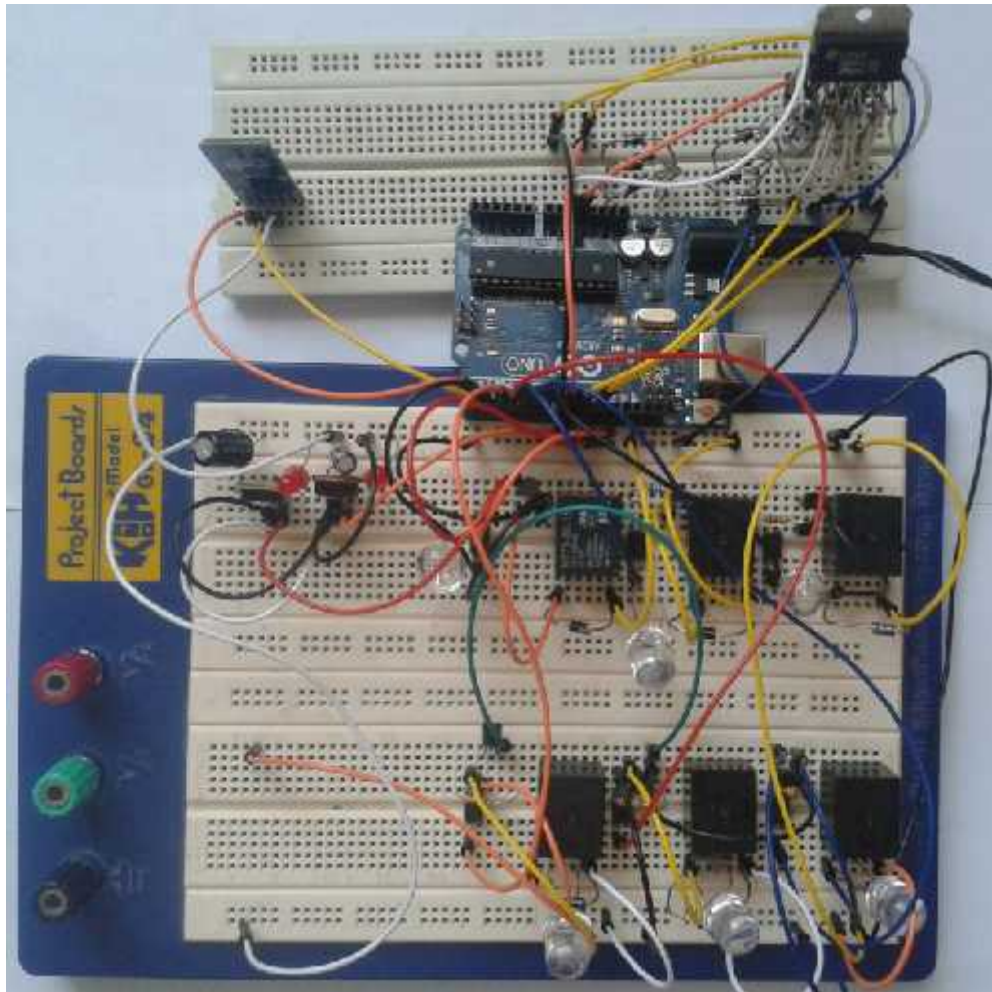


Photo de la réalisation pratique



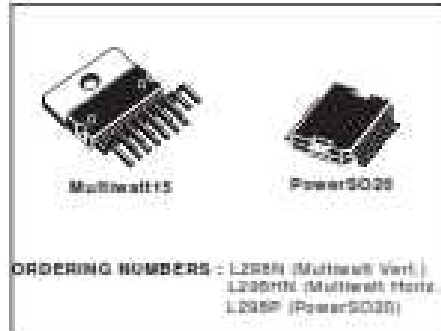
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL '0' INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

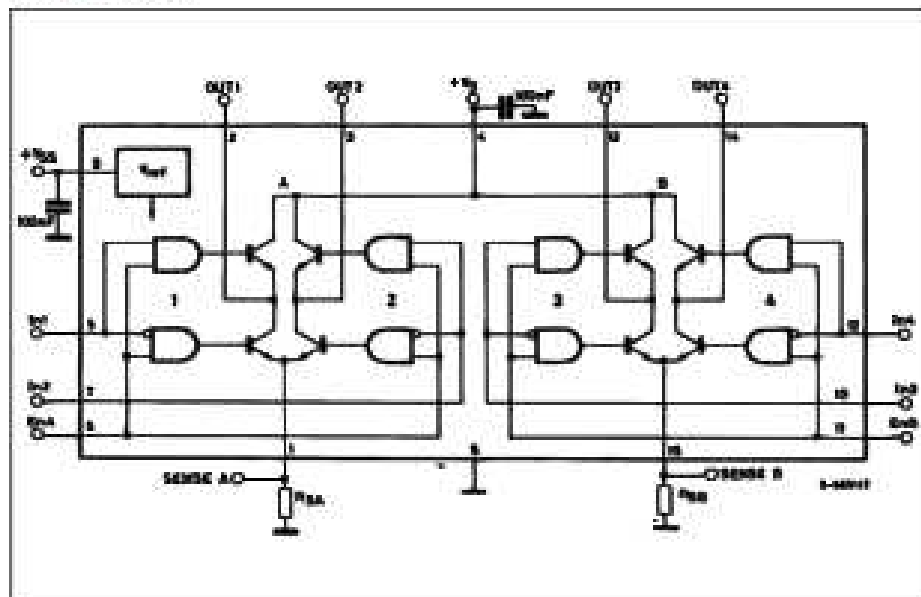
DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multilett and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the con-



nection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM



January 2001

1/13

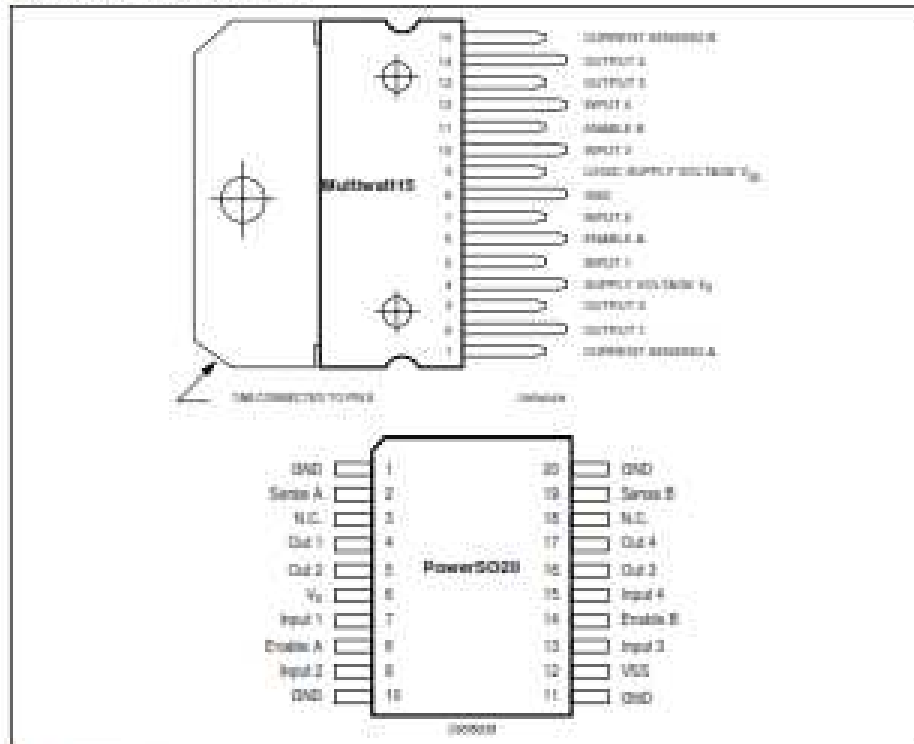
La fiche technique de L 298 (01)

L298

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{DD}	Power Supply	-50	V
V_{DD}	Logic Supply Voltage	7	V
$V_{I, V_{OH}}$	Input and Output Voltage	-0.5 to 7	V
I_O	Peak Output Current (each Channel)		
	- Non Repetitive ($t = 100\mu s$)	3	A
	- Repetitive (50% on -30% off $t_{on} = 10ms$)	2.5	A
	- DC Operation	2	A
V_{sense}	Sensing Voltage	-1 to 2.3	V
P_{tot}	Total Power Dissipation ($T_{amb} = 25^{\circ}C$)	26	W
T_{op}	Junction Operating Temperature	-25 to 100	$^{\circ}C$
T_{stg}, T_j	Storage and Junction Temperature	-40 to 150	$^{\circ}C$

PIN CONNECTIONS (top view)



THERMAL DATA

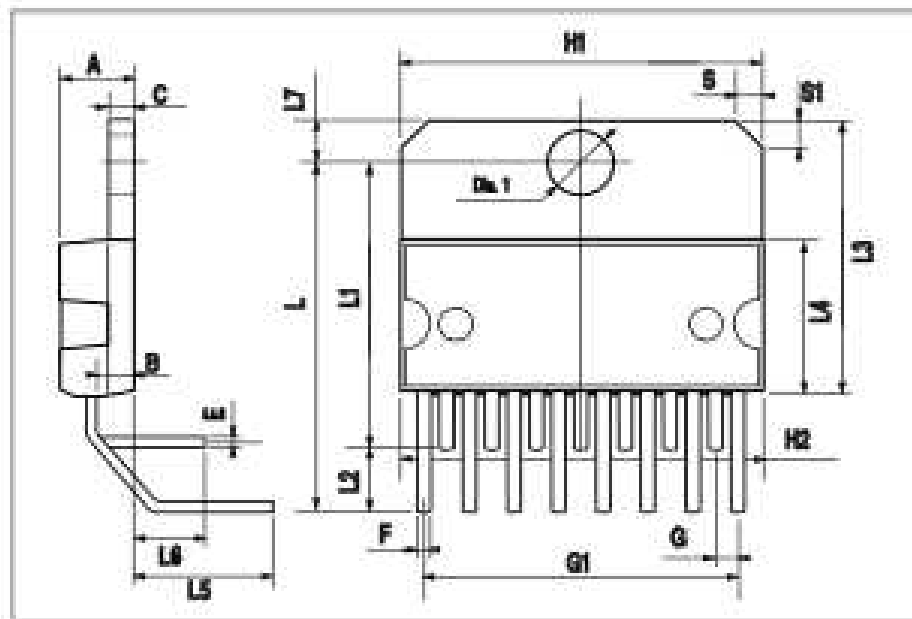
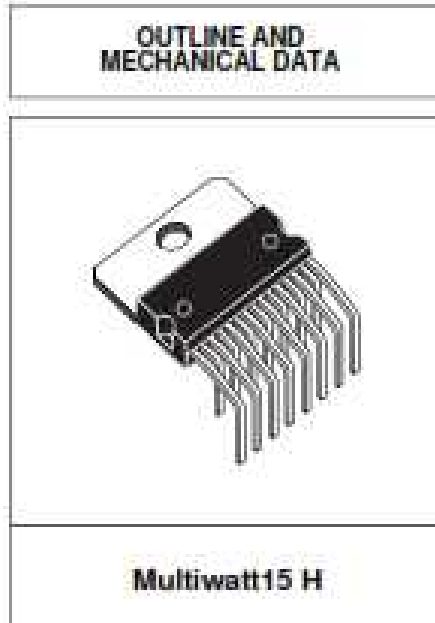
Symbol	Parameter	PowerSO28	Multitrait15	Unit
$R_{th(j-c)}$	Thermal Resistance Junction-case	Max.	3	$^{\circ}C/W$
$R_{th(j-a)}$	Thermal Resistance Junction-ambient	Max.	13 (*)	$^{\circ}C/W$

(*) Mounted on aluminum substrate



L298

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.2			0.083
E	0.49		0.55	0.019		0.022
F	0.65		0.75	0.026		0.030
D	1.14	1.27	1.4	0.045	0.050	0.055
D1	17.57	17.75	17.81	0.692	0.700	0.705
H1	10.6			0.417		
H2			20.2			0.795
L		20.57			0.810	
L1		15.03			0.592	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.688	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.25			0.208	
L6		2.35			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
DIM1	3.85		3.85	0.152		0.152



11/13

La fiche technique de L 298 (03)

L298

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.0			0.142
a1	0.1		0.3	0.004		0.012
a2			3.3			0.130
a3	0		0.1	0.000		0.004
b	0.4		0.53	0.016		0.021
c	0.20		0.30	0.008		0.012
D (1)	12.0		18	0.630		0.830
D1	0.4		0.8	0.016		0.315
E	12.0		14.5	0.547		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.0		11.1	0.430		0.437
E2			2.0			0.114
E3	3.0		6.2	0.228		0.244
D	0		0.1	0.000		0.004
F	12.5		12.9	0.610		0.620
h			1.1			0.043
L	0.5		1.1	0.020		0.043
H	10 (max.)					
S	8 (max.)					
T		10			0.394	

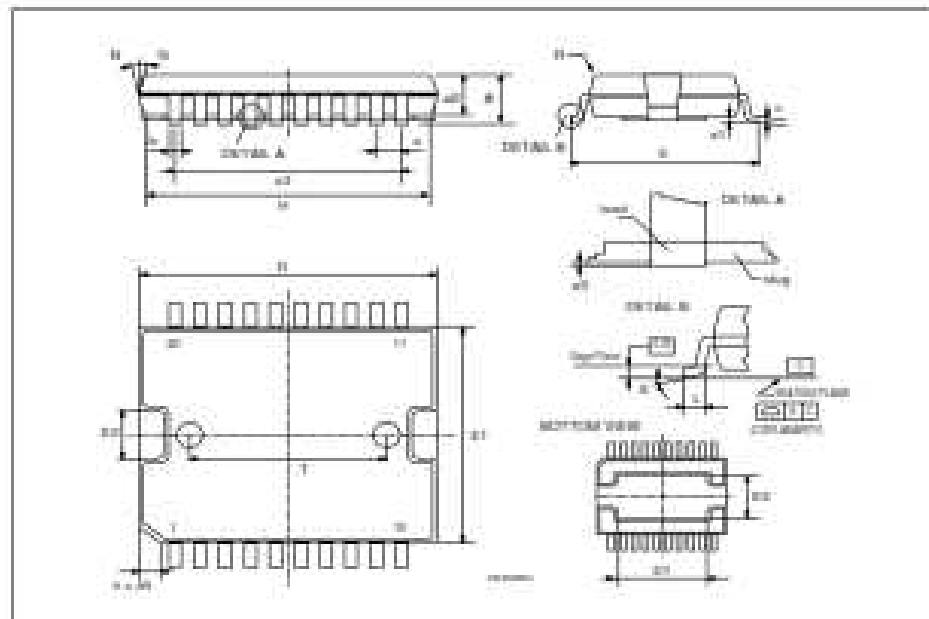
(1) D and F do not include mold flash or protrusion.
Mold flash or protrusion shall not exceed 0.15 mm (0.006").
Critical dimensions: "E", "D" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-156

PowerSO20



62/13



La fiche technique de L 298 (04)

Les Logiciels Utilisés



Pour programmer Arduino



Pour la simulation virtuelle de notre carte



Pour l'application APK



Pour le dessin de typon

:02000004EEE0FA
:10000000B2017728A70089100910091083168910FA
:10001011091009100F30880583122708B2182D28FF
:10001453A30A100FC305E203330A6004120103071
:10003000A1056353088000FF330A100FBCCC0102
:100050002C20B21427082D283214A600321C3B286D
:100060000910033C031C402840200730A100D03079
:100070005E20031408003214FE3C03197228091490
:10008000321C321089140F3088052608F039880494
:100090008910A60E3218412832305D200314080062
:1000A000A301A200FF30A207031CA307031C7228B0
:1000B0000330A100DF305E205228A101E83EA000FD
:1000C000A109FCC569EEC6728A00703186428A007B7
:1000D0006400A10F642820186D28A01C712800005E
:1000E0007128080083130313831264000800831629
:1000F00007309F0083120911831609118312013002
:10010000A3002C308A01512083168101F030860033
:100110008701831286018701FE308A0102208F3019
:100120008A01022034308A01022096308A01502050
:10013000FE308A0102208E308A01022034308A018A
:10014000022058308A01022096308A015020FE3069
:100150008A0102208D308A01022034308A01022077
:1001600058308A01022034308A01022096308A01F8
:100170005020FE308A0102208C308A010220343067
:100180008A01022058308A01022034308A0102207C
:1001900020308A01022096308A015020FE308A01E8
:1001A00002208B308A01022034308A01022058302C
:1001B0008A01022034308A01022020308A01022084
:100200005020FE308A01022089308A0102203430D9
:100210008A01022058308A01022034308A010220EB
:1002200020308A01022043308A0102204C308A01AA
:10023000022041308A01022096308A015020FE308F
:100240008A01022088308A01022034308A0102208B
:1002500058308A01022034308A01022020308A017D
:101C500033308A01022020308A01022033308A0189
:101C70008A01022044308A01022020308A01022099
:0A1C8000080063008A150A12412EC5
:02400E007D3FF4
:00000001FF

Le code hexadécimal de notre programme

- [01] C. Tavernier, « Arduino applications avancées ». Version Dunod.
- [02] <http://www.acm.uiuc.edu/sigbot/tutorials/2009-11-17-arduino-basics>.
consulter le: mars 2015.
- [03] <http://www.generationrobots.com/fr/152-arduino>. consulter le: mars 2015.
- [04] S.V.D.Reyvanth, G.Shirish, « PID controller using Arduino ».
- [05] A.SAI Kishore, J.Sandeep, A.Dadapeer, P.Sai Srinivas.« Three-phase inverter using Arduino» .
- [06] Eskimon, Olyte « Arduino pour bien commencer en électronique et en programmation ».
- [07] Jean- Noël, « livret Arduino en français », centre de ressources art sensitif .
- [08] <http://arduino.cc/en/Main/ArduinoBoardUno>. . consulter le: mars 2015.
- [09] <http://arduino.cc/fr/Main/Librairies>. consulter le: mars 2015.
- [10] A. Grimault, J. Querard « Articl Procédé et dispositif de commutation d'un relais électromagnétique » . EP2312598 A1.
- [11] ERIK Bartmann . « Le grand liver d'Arduino » . 2^{eme} Edition , date de parution 02/01/2014.
- [12] [http://www.Datasheetcatalog.com](http://www.datasheetcatalog.com) .+ (datasheets for electronics components) . consulter le: mai 2015.
- [13] sig.fgranotier.info/IMG/pdf/debuter_app_inventor.
- [14] <http://blogpeda.ac-poitiers.fr> . consulter le: mars 2015.
- [15] <http://www.lifl.fr/tarby>. consulter le: mars 2015.
- [16] <http://scratch.mit.edu/>. consulter le: mars 2015.
- [17] <http://coursstimartinique.fr/>. consulter le: mars 2015.
- [18] <http://fr.wikipedia.org/wiki/JNLP>. consulter le: mars 2015.
- [19] J. Oxe, H. Blemings « Practical Arduino Cool Projects for Open Source Hardware ».
- [20] X.HINAULT. www.mon-club-elec.fr.
- [21] <http://www.labcenter.com/>
- [22] <http://www.cadsoftusa.com/>

يسمح لنا هذا المشروع بالغوص في عالم التواصل من أجل الوصول إلي إنشاء بطاقة إلكترونية تربط بين الهاتف الذكي وبطاقة الأردوينو والمنفذات.

الهدف الأولي من هذا المشروع هو إتقان لغة برمجة من أجل الوصول إلى إنشاء تطبيق أندرويد قادر على إرسال أوامر من طرف مستخدم هذا التطبيق المثبت على الهاتف الذكي إلى محركات التيار المستمر و أجهزة كهربائية أخرى .
فهم مختلف بروتوكولات الاتصال واستعمال المنفذ التسلسلي كمنفذ لإرسال البرنامج إلى بطاقة الأردوينو.
في هذا المشروع قمنا بتحقيق جهاز إتصال عن بعد عن طريق تقنية البلوتوث من أجل تحقيق نوعين من الأوامر، الأولى التحكم في معدات المنازل الذكية والثانية القيادة الذاتية لمركبة عن طريق تطبيق أندرويد مثبت على الهاتف الذكي .
الكلمات المفتاحية: تطبيق أندرويد ، بطاقة الأردوينو، المنفذات، لغة البرمجة، محركات التيار المستمر ، بروتوكولات الإتصال، المنفذ التسلسلي .

Résumé

Ce projet nous permet de plonger dans le monde d'interfaçage afin d'arriver à réaliser une carte électronique qui communique entre (le système Androïde et la carte Arduino ainsi que les actionneurs).

L'objectif préliminaire est de manipuler un langage de programmation afin d'arriver à réaliser une application capable de transmettre des ordres émis par l'utilisateur vers des moteurs à courants continus et autre accessoires électriques via un smart phone.

Faire comprendre les différents protocoles de communication et extraire le port série comme étant un port de transfert du programme dans l'Arduino.

Mots Clés : système Androïde, carte Arduino, les actionneurs, langage de programmation, moteurs à courants continus, protocoles de communication, port série.

Abstract

This project allows us to delve into the world of interfacing in order to arrive to create an electronic card that communicates between (Android system and the Adriano board and the actuators).

The preliminary goal is to manipulate a programming language in order to reach build an application able to transmit commands emitted by the user toward direct current motors and electrical equipment via a smart phone.

Understanding of the different communication protocols and extracts the serial port as a port for transferring program to Arduino board.

We performed a remote control device through a Bluetooth accessory to provide two types of control; the first is the home automation control and the second is self-guiding vehicles by intervention of an application in smart phone.

Key words: Android system, Adriano board, actuators, programming language, direct current motor, communication protocols, serial port.