TABLES DES MATIERES

NOTATIONS

.dll	dynamic link library
.lib	library
CI	Circuit Intégré
CMOS	Complementary Metal-Oxide Semi conductor
dd_	date début
df_	date fin
dr_	date réel
E/S	Entrée – Sortie
EDI	Environnement de Développement Intégré
hd_	heure début
hf_	heure fin
hr_	heure réel
LAN	Local Area Network
PC	Personnal Computer
PPE	Programmation Pilotée par Evénement
RNIS	Réseau Numérique à Intégration de Service
USB	Universal Serial Bus
VPN	Virtual Private Network
WAN	Wide Area Network

INTRODUCTION GENERALE

La domotique est un nom formé à partir du mot latin « domus » qui signifie domicile, et du suffixe « tique » qui désigne l'électronique et l'informatique.

La domotique regroupe l'ensemble des techniques et des technologies permettant de superviser, d'automatiser, de programmer et de coordonner les tâches de confort, de sécurité, de maintenance, donc de services dans un habitat disposant de l'électricité. Elle s'applique tant dans l'habitat résidentiel que dans les domaines industriels et commerciaux.

Le domaine de la domotique encadre toutes les applications touchant spécialement l'électricité, l'électronique et l'informatique, mais aussi les automatismes et le multimédia... Cette technologie permet de rendre une maison communicante et plus conviviale.

Par conséquent, elle englobe l'ensemble des technologies, télécommandes, systèmes d'alarme, portails automatiques, lumières, détecteurs, sécurité, contrôle d'accès, régulation de chauffage ou climatisation, gestion et supervision de l'énergie, arrosage automatique, musique, ambiances (gestion du son), interrupteurs, reconnaissance vocale, etc.

Les divers appareils de l'habitat échangent entre eux des informations signalétiques et assurent à l'habitant un meilleur usage et du confort. D'une part, sur le plan temporel, la domotique présente un avantage remarquable car nous ne sommes plus obligés de faire la tâche manuellement. D'autre part, l'erreur humaine comme l'oubli est évitée car tout est programmé. Néanmoins, la domotique ne traite pas les tâches non prédéfinies. Il y a des décisions que seul l'homme est capable de prendre.

La télécommunication nous fournit des moyens pour mettre en œuvre la domotique à distance. Les équipements de télécommunication offrent et établissent un système de transmission fiable, aussi bien pour la sécurité et la rapidité, que pour la confidentialité.

Compte tenu de l'opportunité offerte par cette nouvelle technologie, par rapport à la situation sous développée de Madagascar, la domotique avec l'ordinateur est-elle un système efficient ? Qu'en est-il de sa mise en pratique ? Et de ses capacités à faciliter le confort habituel ?

Pour répondre à cette problématique, nous allons voir successivement tout au long de ce travail :

- les généralités sur la domotique et le réseau informatique ;
- les conditions de sa mise en œuvre ;
- la réalisation.

Chapitre 1 GENERALITES ET PRESENTATIONS

1.1 Domotique [1]

1.1.1 Principe de base de la domotique

La réalisation de la domotique est composée de divers éléments : les paramétrages, l'unité de commande et les équipements contrôlés. L'unité de commande reçoit les paramètres par l'utilisateur. Les équipements sont centralisés sur cette unité. Cette dernière peut être purement électronique ou informatique. Elle enregistre et gère les données. En effet, l'unité prend la place de l'utilisateur et commande les appareils domestiques au moment opportun. Nous précisons que les tâches à faire, dites automatisées, sont déjà prédéfinies. Il existe aussi des unités préprogrammées par les constructeurs. Dans ce cas, elles ne sont plus pilotées selon la volonté de l'utilisateur, mais répondent aux différents phénomènes extérieurs, et sont prêtes à l'emploi comme les capteurs ou les détecteurs.

Le schéma simplifié suivant récapitule le principe de base de la domotique :

Figure 1.01 : Schéma de principe de la domotique

1.1.2 Mode de fonctionnement de la domotique

Il est possible de réguler l'ensemble de l'habitat par des commandes locales, dans l'habitat même, ou bien à partir de l'extérieur via un micro-ordinateur ou d'un téléphone mobile. La domotique peut être effectuée à partir d'une ou plusieurs commandes centralisées : télécommande, écran tactile, micro-ordinateur etc. En partant par des systèmes les plus simples, en passant par les plus

complexes, la domotique est toujours réalisable. A cet égard, on peut citer quelques exemples : la gestion du chauffage ou de la sécurité, de la lumière, de l'utilisation d'audio-vidéo, etc.

1.1.3 Appareils de commande de la domotique

Plusieurs techniques sont utilisées. La domotique peut se faire grâce à :

- une interface informatique ;
- des modules programmables sur lesquels les appareils sont connectés selon les produits domotiques existants ;
- des moyens de "prise en main" et de pilotage, généralement un téléphone portable ou une télécommande.

Il convient de noter que dans ce travail, notre étude se base sur l'utilisation de l'informatique, de la programmation et des périphériques connectés.

1.1.4 Différents modes de transmission des commandes

De même, il y a plusieurs variantes pour transmettre les commandes vers les dispositifs concernés. La transmission diffère selon l'application voulue. Autrement dit, les appareils en question varient en fonction des caractéristiques mises en jeu, en prenant par exemple la distance requises, la confidentialité, le coût d'infrastructure…

En général, nous retrouvons les transmissions suivantes :

- par ondes radio Hertzienne ;
- par satellites ;
- par infrarouge ;
- par réseau câblé ou filaire.

1.1.5 Domotique et informatique

La domotique fonctionnant à l'aide d'un ordinateur exploite pleinement tous les avantages de l'informatique. Toutefois, il faut noter qu'elle subit également des dérangements causés par des problèmes matériels et logiciels. Il s'agit notamment des virus informatiques, de la faible rapidité de traitement et de la mauvaise communication entre deux machines éloignées l'un de l'autre. Ainsi, la réalisation est strictement liée à l'ordinateur. La machine doit être en état de fonctionnement pour une bonne application de la domotique.

1.1.6 Domotique à Madagascar

Madagascar est une île dont la technologie commence à prendre place rien qu'en parlant de la téléphonie mobile, des différents micro-ordinateurs et de l'arrivée de la fibre optique. Par ailleurs il ne faut pas oublier que l'exploitation de l'existant est plus rentable par rapport aux influences tendancielles des pays développés. De ce fait, la domotique joue un grand rôle pour mettre en jeu cette exploitation. Or, l'implantation n'est pas une chose nouvelle, mais seulement une optimisation. Compte tenu de la contrainte imposée par la routine et le manque de temps d'aujourd'hui, il est temps maintenant pour les connaisseurs informatiques de s'orienter vers la domotique.

1.2 Réseau informatique [2] [3]

1.2.1 Présentation

Un réseau informatique est réalisé par un groupe d'ordinateurs et autres périphériques, tels que les imprimantes et les scanneurs, connectés entre eux par une liaison permettant à tous les périphériques d'interagir entre eux. Les réseaux peuvent être de grande ou de petite taille, connectés durablement ou temporairement. Le plus grand réseau est l'Internet, qui est un groupement de réseaux du monde entier.

1.2.2 Objectif du réseau informatique

Il a pour but de donner à l'utilisateur, à une distance éloignée, l'impression de travailler localement. Il offre un accès à un ordinateur distant. Pour cela, la liberté d'accès dépend du paramétrage de partage, par exemple l'accès au disque dur ou encore aux programmes d'application. Le transfert de données est donc faisable.

1.2.3 Quelques différents types de réseaux informatiques

Plusieurs types de réseaux informatiques existent. Ils se diffèrent selon l'équipement utilisé, les techniques de communication ou encore la distance mise en jeu. Nous pouvons citer ci-après quelques exemples de réseaux informatiques les plus connus :

- réseau étendu (WAN) caractérisé par sa distance géographique étendue ;
- réseau local (LAN) caractérisé par sa zone de travail limité (par exemple, dans un immeuble) ;

- réseau numérique à intégration de services (RNIS) établi par une ligne téléphonique numérique utilisée pour obtenir une bande passante plus importante ;
- réseau privé virtuel (VPN) caractérisé par des liaisons encapsulées, cryptées et authentifiées sur des réseaux partagés ou publics.

Par conséquent, nous pouvons rencontrer différents protocoles selon le type et le paramétrage du réseau. Un protocole est un ensemble de règles et de conventions relatives à l'envoi d'informations sur un réseau. Ces règles régissent le contenu, le format, la synchronisation, la mise en séquence et le contrôle des erreurs des messages échangés entre les périphériques en relation. Il convient de noter que les paramétrages sont requis pour un bon fonctionnement d'une connexion à distance.

1.2.4 Interconnexion

Il existe plusieurs façons de relier les ordinateurs et les équipements. Nous avons pu constater précédemment que l'interconnexion varie selon les techniques et les supports utilisés. Les interconnexions fréquemment utilisées sont les suivantes :

- par câble comme l'interconnexion avec du câble à paire torsadée d'une longueur bien définie ;
- par ligne téléphonique en utilisant un modem ;
- par onde radio à l'aide des émetteurs-récepteurs et des antennes ;
- par lumière infrarouge à travers des ports infrarouges.

1.2.5 Importance du réseau informatique à la domotique

Nous pouvons donc exploiter le principal avantage offert par le réseau informatique. Nous précisons que ce principal avantage consiste en la suppression des incommodités causées par la distance. Ainsi, il est maintenant possible sans se déplacer de contrôler l'élément qui gère les appareils domestiques. Cette méthode est similaire au procédé local, c'est-à-dire que les caractéristiques restent inchangées telles que la sécurité, la confidentialité ou même la vitesse de travail. Seul le coût de la communication distingue les deux méthodes, en supposant que la communication soit parfaite.

1.3 Programmation informatique

Un ordinateur travaille en mode binaire. Un développeur doit faire en sorte de lui donner les instructions nécessaires pour réaliser les tâches souhaitées. Il convient toutefois de préciser qu'écrire tout en langage binaire, serait fastidieux. C'est pour cela qu'il existe des langages de programmation beaucoup plus simples à utiliser.

Les langages de programmation possèdent des syntaxes prédéfinies et normalisées. Cela est réalisé à l'aide de mots-clés, instructions et symboles spécifiques. Le fait de connaître ces derniers ainsi que les mécanismes sous-jacents constitue l'apprentissage d'un langage de programmation.

Ecrire un programme sera en fait l'édition d'un simple fichier texte facilement lisible par un programmeur appelé fichier source. Il ne reste ensuite qu'à compiler ce fichier source pour avoir un fichier exécutable.

1.3.1 Assembleur [4] [5] [6]

Etant un langage de bas niveau, l'assembleur est le langage qui se rapproche le plus du processeur, sans parler du langage machine. La connaissance de ce langage permettra également de mieux comprendre le fonctionnent des autres langages de haut niveau. De ce fait, ces derniers seront plus logiques, et bien souvent plus performants.

Note : Au début, le travail était fait en assembleur, mais compte tenu de la difficulté de concevoir une interface à jour et conviviale pour l'utilisateur, nous avons eu recours à d'autre langage de programmation.

1.3.1.1 Interruptions

Une interruption est comme un petit programme stocké en mémoire. Elle est conçue dans le but d'exécuter une tâche spécifique. Plusieurs interruptions existent mais il serait impossible de les retenir sans se baser sur des documents y afférents.

1.3.1.2 Registres

Un registre est un lieu de stockage à très haute vitesse implanté directement dans le circuit du processeur. La figure ci-dessous représente les registres d'exécution de base. Nous disposons de huit registres à usage général, de six registres de segment, d'un registre de drapeaux (EFLAGS) et

d’un registre pour le pointeur d’instruction (EIP). Les registres servent généralement à réaliser des calculs arithmétiques et des déplacements de données. Certains registres peuvent satisfaire aussi à des besoins spécifiques, dont notamment :

- un accumulateur pour les instructions de multiplication et de division ;
- un compteur de boucle ;
- un registre de gestion de données de la pile.

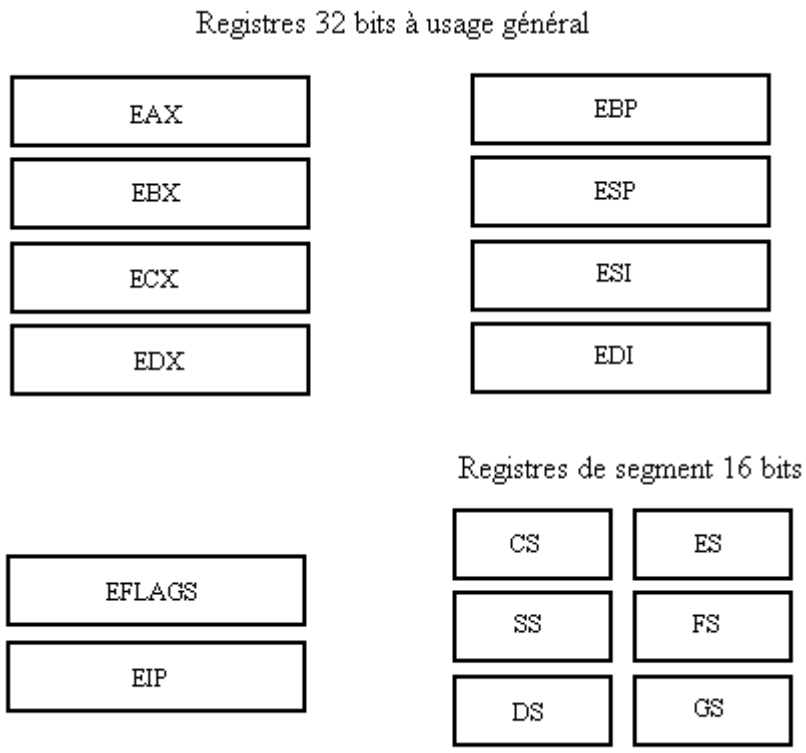

Figure 1.02 : Les registres d’exécution de base

1.3.1.3 Instructions

Une instruction est un ordre, exécutée par le processeur, une fois que le programme a été chargé en mémoire et lancé. Une instruction est constituée de quatre parties principales :

- un label (facultatif) est un identificateur qui permet de désigner un endroit dans le code source, soit une instruction, soit une donnée ;
- un mnémonique d’instruction (obligatoire) est un mot court d’origine anglaise qui identifie une instruction ;

- un ou des opérandes normalement obligatoires, entre zéro et trois en assembleur. Chaque opérande peut être le nom d'un registre, un opérande mémoire, une expression constante ou le nom d'un port d'entrée/sortie ;
- des commentaires constituent toujours un élément essentiel dans la conception de programmes informatiques. Il permet en effet d'ajouter des informations explicatives au sujet du fonctionnement du programme.

1.3.1.4 Points forts de l'assembleur

L'assembleur permet de bénéficier, non seulement d'outils de développement comme l'accès direct aux matériels, mais aussi de mis au point comme les drapeaux. En plus, la taille du programme en assembleur est petite et sa vitesse est plus rapide lors de l'exécution.

En assembleur, nous risquons d'être en difficulté face à l'hexadécimal. Cela est classé non pas comme un défaut mais plutôt comme un désavantage de l'utilisation de l'assembleur. Actuellement, en programmation, il existe des logiciels conçus pour être adaptés aux utilisateurs regroupant les différents outils nécessaires à la réalisation d'un projet, et qui sont très facile à utiliser ou du moins à comprendre. Par suite, il est préférable d'utiliser l'assembleur, non pas à l'écriture d'un programme tout entier, mais seulement à servir comme un outil d'optimisation des programmes faits en langage de haut niveau.

1.3.2 Borland C++ Builder [7] [8]

C'est un langage de haut niveau. Il est conçu pour être plus proche du développeur. Pour un aperçu général du langage, quelques définitions sont présentées ci-après :

1.3.2.1 L'environnement de développement intégré (EDI)

Au démarrage de C++ Builder, nous sommes placés immédiatement dans l'environnement de développement intégré, appelé également EDI. Cet environnement propose tous les outils nécessaires à la conception, au test, au débogage et au déploiement d'applications.

L'EDI contient un concepteur visuel des fiches, un inspecteur d'objets, une palette des composants, un gestionnaire de projet, un éditeur de code source, un débogueur et un outil d'installation. La modification de propriétés liées au code dans l'inspecteur d'objets transforme automatiquement le code source correspondant.

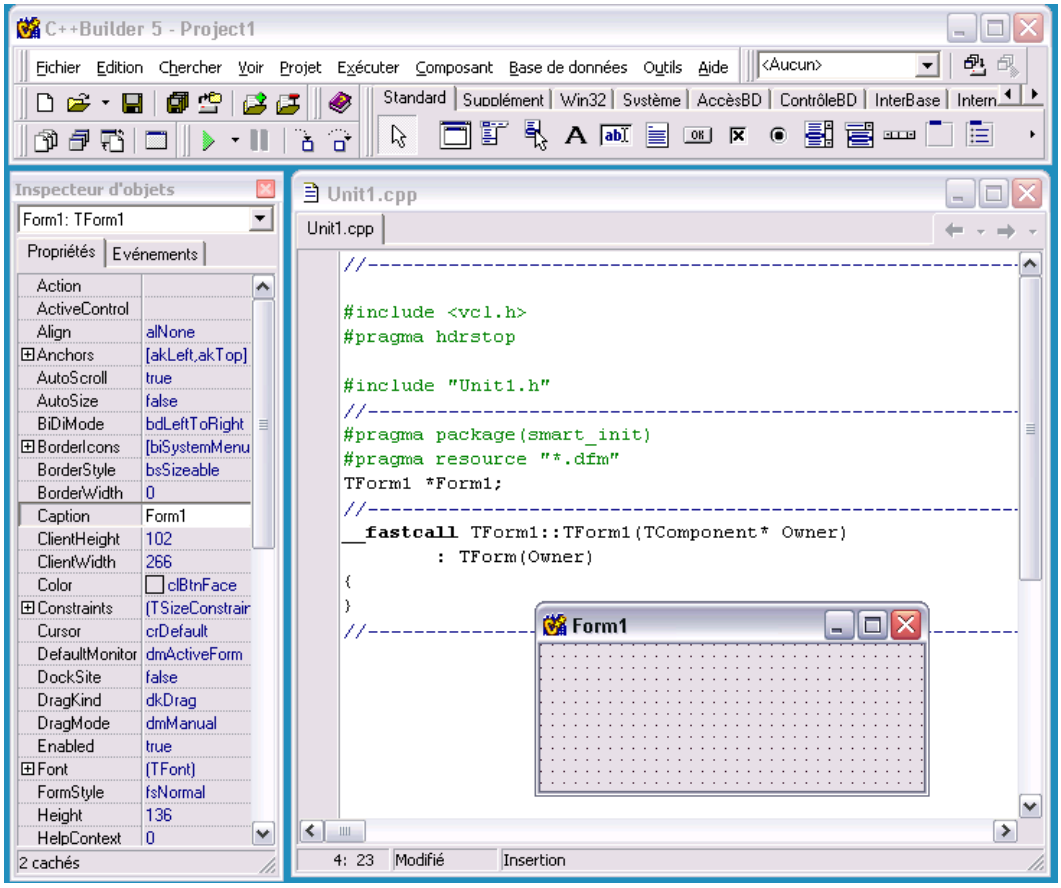

Figure 1.03 : Aperçu de l'EDI

1.3.2.2 Inspecteur d'objets

C'est une fenêtre à deux volets respectivement spécialisés pour l'édition des valeurs des propriétés de composants et pour les fonctions de leurs gestionnaires d'événements.

1.3.2.2.1 Propriétés

Les propriétés sont les caractéristiques des composants. Nous pouvons voir et modifier les propriétés à la conception et obtenir une réponse immédiate des composants dans l'EDI.

1.3.2.2.2 Evénements

La Programmation Pilotée par Evénement (PPE) signifie qu'elle dépend des événements voulus. Donc, plusieurs événements sont disponibles selon différents cas. Avec la programmation

événementielle, nous écrivons un code pour gérer les événements pouvant se produire facultativement, au lieu d'écrire un code s'exécutant toujours selon le même ordre.

1.3.2.3 Utilisation des composants

Nous accédons à la plupart des composants visuels de l'EDI sur la palette des composants. Pour concevoir l'interface, utilisateur de l'application, il suffit de sélectionner les composants dans la palette et de les déposer dans la fiche concernée. Une fois qu'un composant visuel est placé dans la fiche, il est possible de modifier sa position, sa taille, ainsi que de nombreuses autres propriétés.

1.3.2.4 Les syntaxes

En plus de ces propres syntaxes, le logiciel de programmation intègre aussi des syntaxes similaires employées en C++. Afin d'élaborer un bon travail, il est préférable de consulter l' « aide » de ce logiciel. Celui-ci offre, par simple clic, toutes les indications complémentaires.

Chapitre 2 LES ELEMENTS MATERIELS UTILISES

2.1 Port parallèle [9] [10]

2.1.1 Préliminaires

Un port d'entrée/sortie (E/S) est un canal de transfert des données entre un périphérique et le microprocesseur. Le microprocesseur considère le port comme une ou plusieurs adresses mémoires qu'ils peuvent utiliser pour envoyer ou recevoir des données. Un port est un point de connexion de l'ordinateur auquel nous pouvons raccorder plusieurs périphériques. Voici quelques exemples :

- le port de jeu qui est un connecteur d'entrée/sortie auquel un joystick ou manette de jeu est connecté ;
- le port infrarouge qui est un port optique utilisant lumière infrarouge pour se communiquer ;
- le port parallèle qui est une interface parallèle, généralement conçu pour les imprimantes ;
- le port série qui est une interface qui permet la transmission asynchrone de caractères de données d'un bit à la fois, nommé aussi port COM ;
- le port USB qui est une interface permettant la connexion d'un périphérique USB.

Nous allons nous intéresser plus objectivement sur le port parallèle qui nous servira un port de communication pour une application de la domotique.

2.1.2 Présentation du port parallèle

Parmi tous les ports cités supra, c'est le port parallèle du PC qui est le plus intéressants grâce à ses possibilités et à sa simplicité de programmation. Contrairement au port série, il ne nécessite aucun protocole de transmission. Il existe différents types de liaison parallèle, mais nous allons travailler dans le mode unidirectionnel, qui est paramétré dans le BIOS pour avoir seulement 8 bits de données en sortie. Les états de sorties peuvent être à un niveau 0 ou 1(0 ou 5V).

En plus de ces 8 lignes de données, le port parallèle dispose également de 4 lignes de contrôle (sortie), 5 lignes d'état (entrée) et plusieurs masses.

2.1.2.1 Connecteurs

Le port parallèle des PC se présente sous la forme d'un connecteur DB-25 femelle. Logiquement, un connecteur mâle sera branché au PC.

Les schémas suivants décrivent les numérotations des broches.

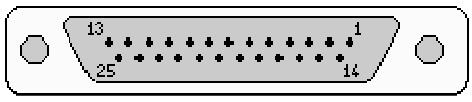

Figure 2.01 : Connecteur DB-25 femelle

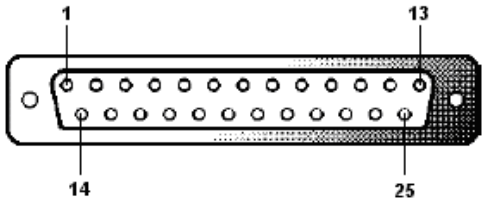

Figure 2.02 : Connecteur DB-25 mâle

2.1.2.2 Description des signaux sur chaque broche

Au début, le port parallèle nommé LPT est utilisé pour la transmission de données vers une imprimante. Mais l'évolution de la technologie tend actuellement à délaisser cette technique.

Pour bien comprendre les fonctions de chaque broche, supposons qu'une imprimante est branchée sur l'ordinateur :

- STROBE (1) : cette ligne active basse (donc a 0) indique à l'imprimante que des données sont présentes sur les lignes D0 à D7 et qu'il faut les prendre en compte.
- D0 à D7 (2-9) : c'est le bus de données sur lequel véhicule la valeur du caractère à imprimer.

Bit	7	6	5	4	3	2	1	0
Nom	D7	D6	D5	D4	D3	D2	D1	D0

Tableau 2.01 : Nomenclature des bits

- ACK (10) : l'imprimante met à 0 cette ligne pour indiquer à l'ordinateur qu'elle a bien reçu le caractère transmis et qu'il peut continuer la transmission.
- BUSY (11) : cette ligne est mise à 0 par l'imprimante lorsque son « Buffer » de réception est plein. L'ordinateur est ainsi averti que celle-ci ne peut plus recevoir de données. Il doit attendre que cette ligne revienne à 1 pour recommencer à émettre.
- PE (12) : signifie " Paper Error ". L'imprimante indique par cette ligne à l'ordinateur que l'alimentation en papier a été interrompue.
- SELECT (13) : cette ligne indique à l'ordinateur si l'imprimante est "on line" ou "off line".
- AUTOFEED (14) : lorsque ce signal est à 1, l'imprimante doit effectuer un saut de ligne à chaque caractère "return" reçu. En effet, certaines imprimantes se contentent d'effectuer un simple retour du chariot en présence de ce caractère.
- ERROR (15) : indique à l'ordinateur que l'imprimante a détecté une erreur.
- INIT (16) : l'ordinateur peut effectuer une initialisation de l'imprimante par l'intermédiaire de cette ligne.
- SELECT IN (17) : l'ordinateur peut mettre l'imprimante hors ligne par l'intermédiaire de ce signal.
- MASSE (18-25) : c'est la masse du PC.

2.1.2.3 Adresses des différents registres

Il est très facile de programmer cette interface en connaissant les adresses utilisées. Trois registres seulement sont nécessaires au contrôle total des signaux. Un registre est un endroit où sont stockées des valeurs. Chaque port parallèle possède ses propres registres:

Port parallèle	Port du registre de données	Port du registre d'état	Port du registre de commande
n°1	378h	379h	37Ah
n°2	278h	279h	27Ah
n°3	3BCh	3BDh	3BEh

Tableau 2.02 : Adresse des différents registres

Lorsque nous voulons écrire directement sur ce port, nous nous intéressons spécialement au registre de données. L'octet à envoyer vers le périphérique est alors placé dans ce registre. Il est ensuite immédiatement placé sur les lignes D0 à D7 du port parallèle.

Pour les systèmes actuels le port LPT1 se situe toujours à l'adresse 0x378 et le port LPT2 à l'adresse 0x278. Mais pour en être sûr nous pouvons vérifier dans Windows en suivant les deux méthodes suivantes:

1ère méthode : (voir Figure 2.03)

- Panneau de configuration
- Système
- Gestionnaire de périphérique
- Ports (COM et LPT)
- Port imprimante (LPTx)
- Ressources
- Plage d'entrée/sortie qui vous indiquera les ports utilisés.

2ème méthode : (voir Figure 2.04)

- Menu Démarrer
- Programmes
- Accessoires
- Outils système
- Informations système
- Ressources système
- Port E/S.

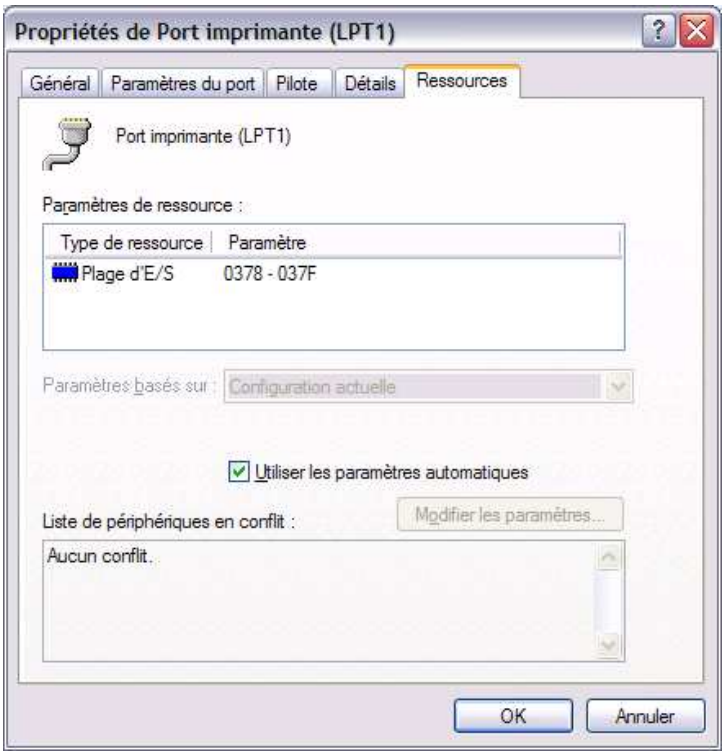

Figure 2.03 : 1ère méthode de vérification d'adresse de registre

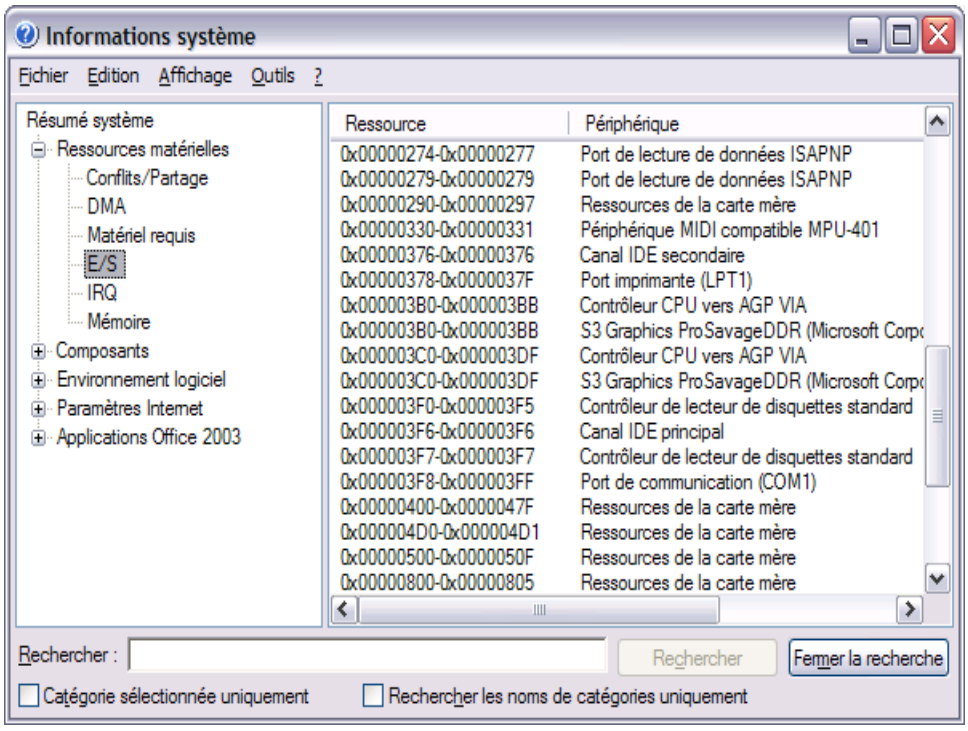

Figure 2.04 : 2ère méthode de vérification d'adresse de registre

2.1.2.4 Programmation

Le langage utilisé doit permettre de manipuler des octets et d'accéder aux adresses d'entrée/sortie. Par exemple en assembleur nous utilisons les instructions IN et OUT. Le système d'exploitation doit autoriser cet accès direct, ou bien il faut se procurer le composant logiciel capable de faire le travail: pilote, DLL pour Windows 9x, etc. Ensuite, il ne reste plus qu'à écrire dans ces registres à ces adresses pour commander directement les lignes du port parallèle.

2.2 Composants électroniques [11] [12] [13]

La domotique nous oblige, non seulement à concevoir des programmes de gestion qui traitent les informations venant de l'utilisateur, mais aussi à construire un dispositif électronique qui joue le rôle d'une passerelle pour communique vers les appareils externes. Ce dispositif traduit alors les données informatiques en une action bien définie. Nous parlerons donc ci après quelques composants considérés comme obligatoires à la réalisation.

2.2.1 Transistor en commutation [1]

Le transistor en commutation est utilisé afin d'ouvrir ou de fermer un circuit. Ainsi il peut commander une LED, un relais, un moteur... Nous assimilons généralement le circuit de sortie du transistor à un interrupteur qui est commandé soit par une tension, soit par un courant suivant le type de transistor choisi. De ce fait, il est nécessaire de dimensionner les composants utilisés pour un bon fonctionnement d'un transistor en commutation.

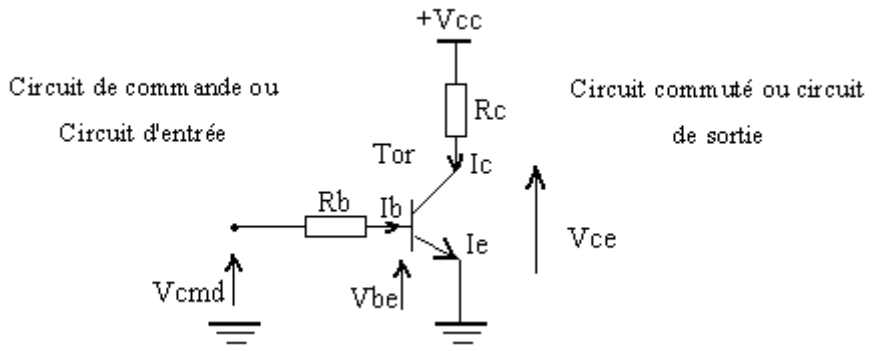

Figure 2.05 : Schéma du montage d'un transistor en commutation

Nous remarquons qu'un transistor est convenablement polarisé dans l'état passant (ou saturé) si la jonction Base-Emetteur est polarisée en sens direct et la jonction Base-Collecteur est polarisée en sens inverse.

2.2.2 Relais [11] [12]

2.2.2.1 Présentation

Un relais est un appareil électrique dans lequel un phénomène électrique (courant ou tension) contrôle la commutation On / Off d'un élément mécanique (relais électromagnétique) ou d'un élément électronique (relais statique). C'est en quelque sorte un interrupteur que l'on peut actionner à distance. La tension et le courant de commande (partie "Commande"), ainsi que le pouvoir de commutation (partie "Puissance") dépendent du relais, il faut choisir ces paramètres en fonction de l'application désirée. En effet, nous pouvons définir différentes caractéristiques pour un relais:

- caractéristique concernant le circuit d'alimentation ;
- caractéristique concernant le circuit d'utilisation ;
- caractéristique concernant les conditions d'emploi.

Grâce à l'évolution de la technologie, nous pouvons distinguer plusieurs types de relais tels que le relais à induction, le relais thermique, le relais statique… Toutefois le relais électromagnétique reste le plus utilisé et le plus facile à trouver sur le marché.

2.2.2.2 Relais électromagnétique

Ci-contre un exemple de représentation d'un relais électromagnétique :

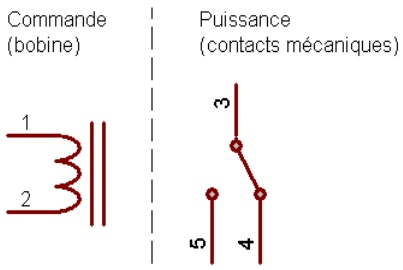

Figure 2.06 : Schéma d'un relais électromagnétique

Un relais électromagnétique est doté d'un bobinage en guise d'organe de commande. La tension appliquée à ce bobinage va créer un courant. Ce courant produit un champ électromagnétique à l'extrémité de la bobine. Ce champ magnétique va être capable de faire déplacer un élément mécanique métallique, monté sur un axe mobile, et qui déplacera alors des contacts mécaniques.

2.2.3 Portes logiques **[14]**

Nous savons qu'un ordinateur présente un nombre fini de ports d'entrée-sortie. Par conséquent, les lignes des données sont limitées. Cela nous oblige à maîtriser les différentes portes logiques. En effet, nous nous basons sur des fonctions logiques afin de palier à cette inégalité entre les lignes. Autrement dit, nous devons jongler aves ces circuits élémentaires logiques pour gérer un nombre supérieur de périphérique à partir des lignes de données limitées. Nous trouvons trois opérateurs élémentaires :

- l'addition logique qui est réalisée par la porte OU ;
- la multiplication logique qui est réalisée par la porte ET ;
- l'inversion logique nommée aussi la complémentation est réalisée par la porte NON.

Il importe de noter que d'autres variantes dérivées de ces derniers existent tels que la porte XOR, la porte EXNOR…

En ce qui concerne les circuits logiques, ils ne peuvent prendre que deux valeurs. Le système de numérotation binaire joue donc un rôle particulier. Nous représentons alors respectivement les plages de tension 0V – 0.8V et 2V - 5V par les valeurs 0 et 1.

Chapitre 3 REALISATION

Pour concrétiser l'étude antérieure, il convient maintenant de procéder à une application de la domotique. L'application se fait sur un système d'exploitation Windows 98. La raison du choix de ce système d'exploitation est que, ce dernier autorise l'accès direct au port parallèle sans intervention d'un quelconque programme. Il est à noter que cette application est aussi faisable avec la version Windows XP en apportant quelque modification au programme.

Cette application consiste à piloter un dispositif comportant deux sources électriques programmables et une source électrique indépendante (actionné manuellement), de 220V – 5A chacun. Des programmes avec des interfaces graphiques sont donc associés à ce dispositif, afin de le manier facilement.

3.1 Description du coté dispositif

3.1.1 Présentation du dispositif

Le dispositif se caractérise par les dimensions suivantes : 26cm x 5cm x 4cm. Il dispose de trois sources électriques. Un connecteur DB25 femelle sert de port de communication pour accéder à une carte électronique. Autrement dit, toutes les lignes entrantes, y compris les alimentations de cette carte sont reliées sur ce connecteur. Par conséquent, les « périphérique 1 » et « périphérique 2 » sont commandés par cette même carte. Deux voyants « LED 1 » et « LED 2 » servent d'indications d'états (ouvert ou fermé) des deux sources placées au milieu. La source nommée « périphérique 3 » se trouvant tout au bout du dispositif est accessible manuellement par l'interrupteur « k ».

La photo suivante montre l'aspect du dispositif :

Photo 3.01 : Schéma du dispositif

La figure ci-après montre l'implantation des composants sur la carte électronique avec les câblages existants, plus les autres composants complémentaires.

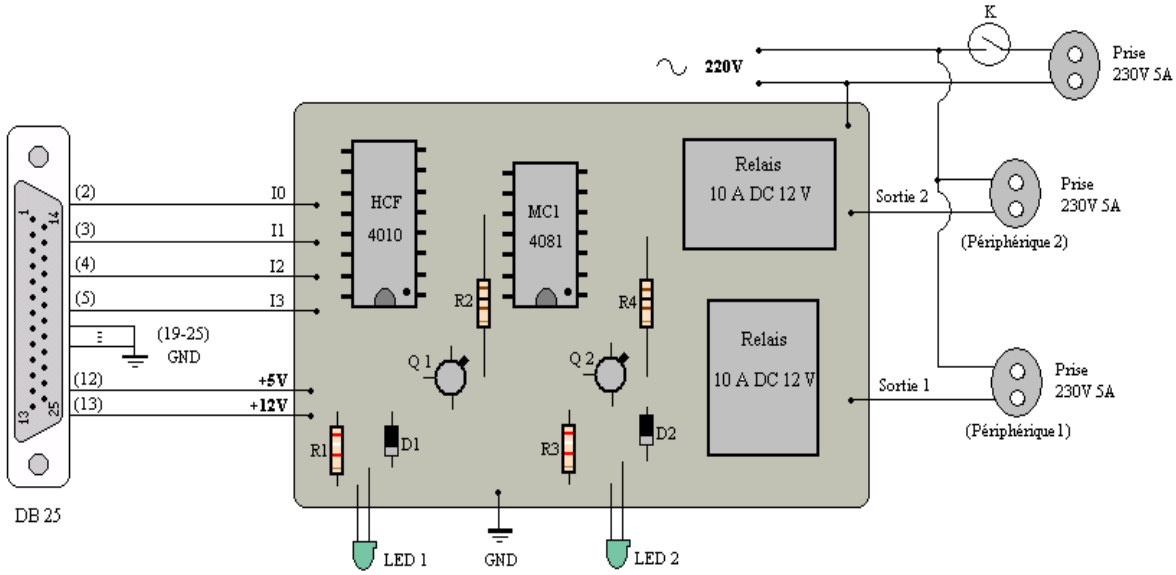

Figure 3.01 : Schéma interne du dispositif

3.1.2 Fonctionnement

La figure ci-dessous montre le schéma de base de la carte électronique ou précisément du circuit de commande. Le schéma peut être subdivisé en deux blocs identiques avec le même fonctionnement. Notons qu'I0, I1, I2 et I3 sont les entrées de commande.

L'objectif est de commuter le relais pilotant la source électrique d'un état à l'autre (ouvert ou fermé) suivant les données présentes sur les entrées. Deux entrées logiques suivis d'un amplificateur chacun sont alors disponibles pour piloter deux entrées du porte ET. Par la suite, la sortie de cette porte définit l'état du montage transistor en commutation. D'un niveau de tension de sortie un peu plus grand par rapport à l'entrée, ce montage actionne le relais énoncé précédemment. Ce relais a donc pour objectif de faire passer ou non une tension de 220V.

Au cas où il y a retour de signal causé par un court-circuit ou une défaillance matérielle, l'amplificateur servira de protection pour le port de l'ordinateur concerné. En effet, c'est cet amplificateur qui risque d'être endommagé mais non pas le port de l'ordinateur.

En ce qui concerne de la source électrique, deux états se présentent :

- *état ouvert* : si l'un a au moins un niveau logique 0
- *état fermé* : si et seulement si les deux entrées ont le même niveau logique 1

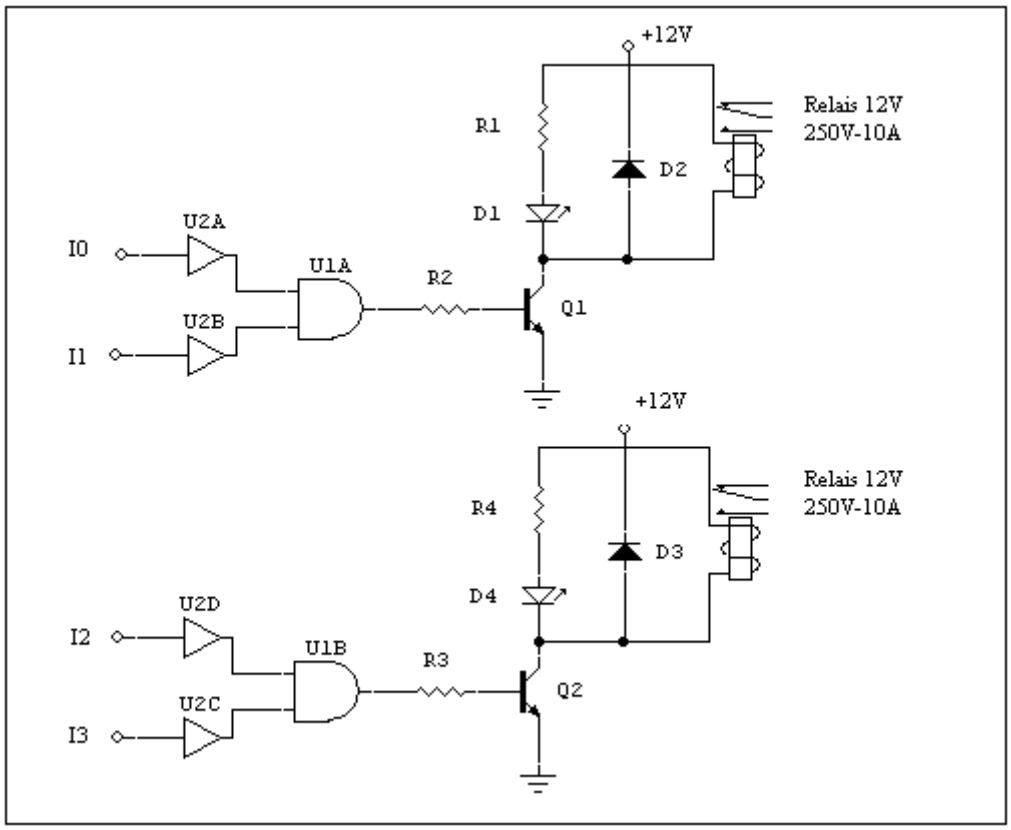

Figure 3.02 : Schéma de base du circuit de commande

Une résistance et une diode LED sont montées en parallèle avec le relais pour indiquer le passage du courant ou la fermeture du relais. Pour la protection de la bobine du relais, il est primordial de placer en parallèle une diode dont la cathode est reliée au potentiel supérieur.

3.1.3 Raccordement

Il est nécessaire de raccorder le dispositif à l'ordinateur. Nous concevons alors un cordon muni de deux connecteurs DB25 à chaque extrémité, avec un connecteur à quatre éléments pour alimenter la carte électronique. Sa longueur est environ de 80cm. La photo ci-après montre ce cordon :

Photo 3.02 : Schéma du cordon de raccordement

La connexion entre les différents connecteurs est détaillée dans le schéma suivant :

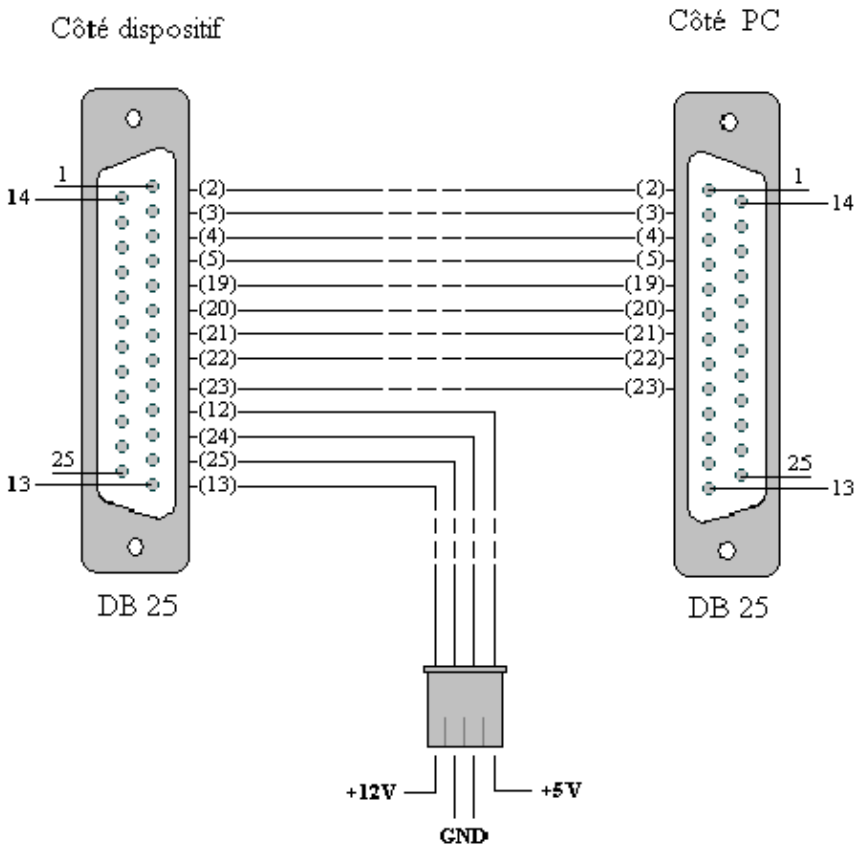

Figure 3.03 : Câblage du cordon de raccordement

3.1.4 Alimentations

Les alimentations 5V et 12V sont indispensables à la mise en marche de la carte électronique. Une alimentation pouvant débiter cette requête est donc nécessaire comme celle se trouvant dans les ordinateurs de bureau. Cette alimentation doit permettre aussi la connexion. Par conséquent, les ordinateurs portables sont exclus du domaine d'application sans un apport d'une alimentation externe.

Dans notre cas, les alimentations sont tirées à partir de l'unité centrale. En plus, cette méthode favorise une économie de place et de composant.

3.1.5 Brochages des circuits intégrés utilisés

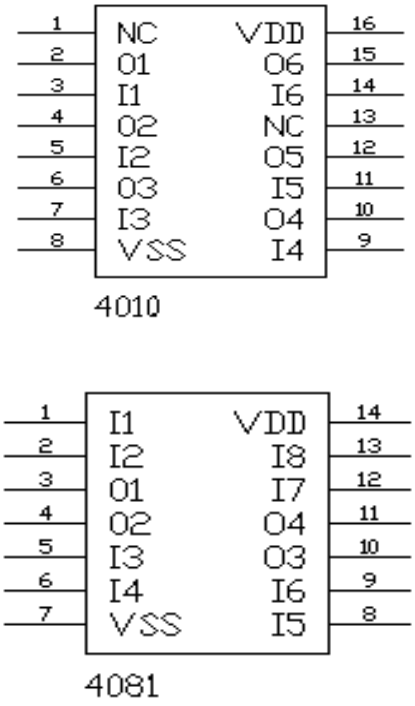

Figure 3.04 : Brochages des circuits intégrés

Ces deux circuits sont de la famille CMOS. Les descriptions sont les suivantes :

- *VDD* : tension d'alimentation ;
- *VSS* : tension de référence ;
- *I1 - I8* : tensions d'entrée ;
- *O1- O6* : tensions de sortie.

3.1.6 Nomenclature des composants et des matériels utilisés

Le tableau suivant récapitule les composants et les matériels utilisés, avec quelques caractéristiques et notations utilisées :

Descriptions	Références/ Caractéristiques	Notations	Nb
Amplificateur	HCF 4010 BE	U2A, U2B, U2D, U2C	04
Porte ET	MC1 4081 BCP	U1A, U1B	02
Transistors	BC107	Q1, Q2	02
Resistances	0.1kΩ - 1/2W	R2, R4	02
Resistances	3.3kΩ - 1/2W	R1, R3	02
Diodes	1N4001	D1, D2	02
Diodes (LED)	5mm	LED1, LED2	02
Relais	10A DC 12V	-	02
Plaquette	9cm x 5cm	-	01
Nappe	80cm à17 lignes	-	01
Connecteur	DB25 femelle	-	01
Connecteurs	DB25 mâle	-	02
Connecteur	4 éléments (alimentation)	-	01
Prises (standard)	230V – 5A	-	03

Tableau 3.01 : Nomenclature des composants et des matériels utilisés

3.2 Description du coté programme

Le dispositif présenté précédemment nécessite donc un programme d'exploitation. Ce dernier a été écrit avec le logiciel de développement Borland C++ Builder. Nous l'avons baptisé sous le nom de « MAprise ». C'est une programmation événementielle. Ce programme se déroule selon l'initiative de l'utilisateur. Il inclut aussi des modules écrits en langage assembleur. Un installateur est conçu et disponible par un simple clic pour faciliter l'installation sur la machine. Par la suite, une icône

nommée « MAprise » est automatiquement placée sur le bureau et qui va servir de raccourci pour l'exécution du programme.

3.2.1 Programme d'initialisation

Pour initialiser la machine, une partie du programme est placé dans un répertoire où ceci est lancé automatiquement au démarrage par le système d'exploitation. Nous pouvons vérifier dans « Utilitaire de configuration système ». Ce petit programme a le rôle de mettre immédiatement à zéro les lignes de données du port parallèle au démarrage de la machine. Cette procédure est adoptée afin d'éviter l'activation du dispositif sans assistance de l'utilisateur, car des valeurs inattendues seront présentes sur les lignes de données à cause des tests faits par la machine.

3.2.1.1 Organigramme du programme d'initialisation

Il se base sur l'organigramme suivant :

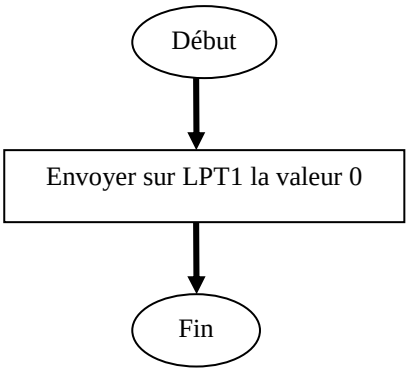

Figure 3.05 : Organigramme du programme d'initialisation

L'organigramme montre que l'écriture sur le port est exécutée sans aucune attente.

3.2.1.2 Aperçu du programme d'initialisation

Une boîte de dialogue qui est non redimensionnable s'affiche en bas droit de l'écran juste au moment de démarrage de la machine. C'est une simple boîte d'information. Il n'y a aucun paramètres ou boutons à manipuler. D'une durée de 3 secondes, elle s'affiche et initialise le port parallèle. Ce programme sera détaillé dans l'annexe 1. L'aperçu est présenté comme suit :

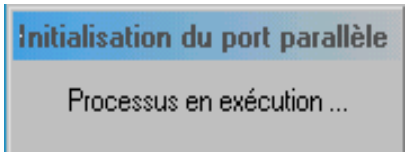

Figure 3.06 : Aperçu de la boîte indiquant l'initialisation au démarrage

3.2.2 Programme principal

Ce programme est exécuté en cliquant sur l'icône nommée « MAprise ». Contrairement au programme d'initialisation, il prend en compte les paramètres venant de l'utilisateur. Le fonctionnement du dispositif est donc strictement lié à ces paramètres. Par conséquent, le dispositif est sous contrôle de l'utilisateur. Ce programme sera détaillé dans l'annexe 2.

3.2.2.1 Organigramme de base de l'application

L'objectif est de pouvoir piloter le dispositif c'est-à-dire de le mettre en marche ou en arrêt à un temps bien défini, et selon le choix de l'utilisateur. De ce fait, ce programme se base sur les deux organigrammes suivants : l'organigramme de démarrage et l'organigramme d'arrêt.

3.2.2.1.1 Organigramme de démarrage

Un test du jour est fait au début. S'il est évalué vrai alors nous passons au second test, sinon c'est la fin de procédure. Comme dans le cas du premier test, la valeur vraie du second test qui teste l'heure impliquera à lire la valeur sur le port LPT1. Suite à cette lecture, nous faisons une opération **OU logique** avec une valeur prédéfinie selon le périphérique concerné. Cette opération nous servira à extraire et à mettre au niveau logique 1 les bits commandant ce périphérique. Il ne reste ensuite qu'à écrire cette nouvelle valeur sur le port LPT1.

Nous notons que les tests ne sont que des comparaisons de valeurs fixes avec la date et l'heure du système. Théoriquement, une boucle *while* est idéale pour réaliser ces tests. Mais pratiquement, quelque problème se pose lors de l'exécution. Borland C++ Builder a un composant appelé « Timer » capable de s'auto exécuter à un intervalle paramétrable. Le problème est résolu en intégrant un organigramme se basant sur une instruction *if* dans ce composant. Voici l'organigramme qui vient d'être décrit :

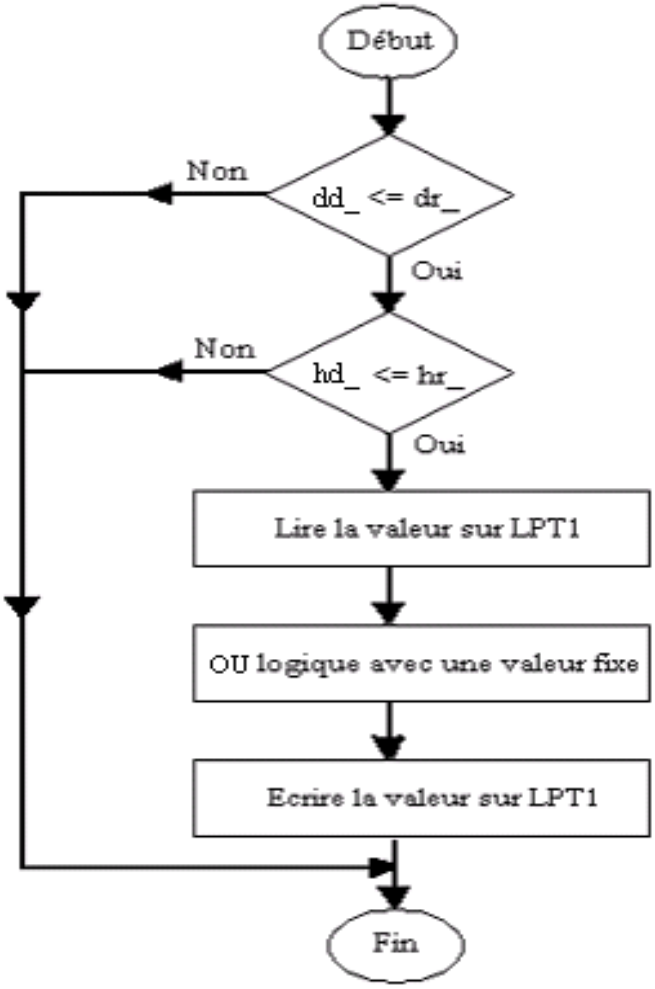

Figure 3.07 : Organigramme de démarrage

3.2.2.1.2 Organigramme de l'arrêt

L'organigramme est le même que celui du programme de démarrage. Il se diffère seulement au niveau de l'opération OU logique par l'opération ET logique. Cette dernière nous servira à extraire et à mettre au niveau logique 0 les bits concernés. Ensuite, nous passons à la réécriture de la nouvelle valeur sur le port LPT1.

Cet organigramme se présente comme suit :

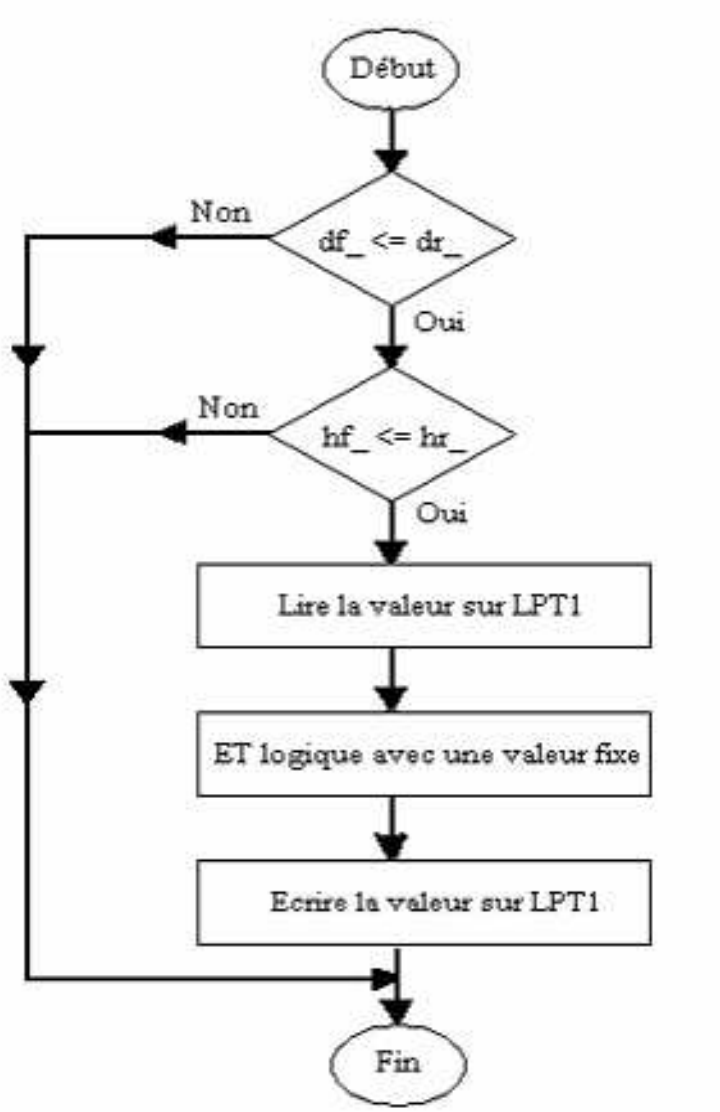

Figure 3.08 : Organigramme d'arrêt

3.2.2.2 Description de l'interface graphique

En cliquant sur l'icône nommée « MAprise » placée sur le bureau, une fenêtre s'ouvre au milieu de l'écran. Elle comprend les onglets suivants : « Général » et « Paramètres ». Les deux figures ci-après montrent les apparences de la fenêtre contenant les différents onglets.

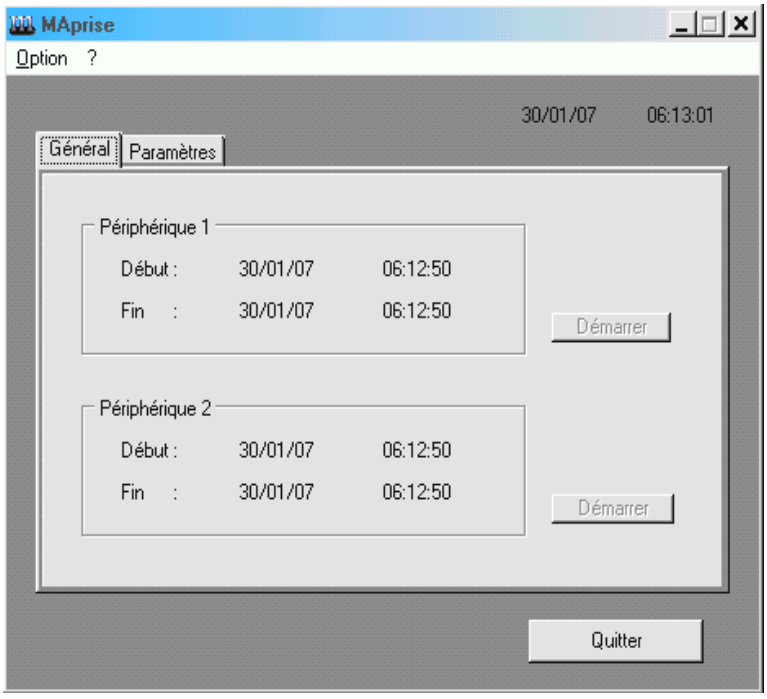

Figure 3.09 : Aperçu de la fenêtre avec l'onglet « Général »

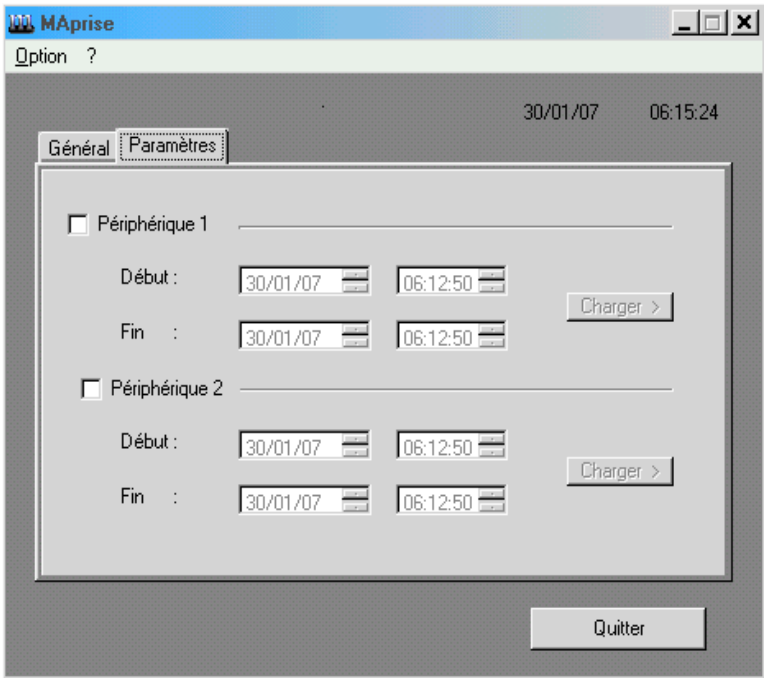

Figure 3.10 : Aperçu de la fenêtre avec l'onglet « Paramètres »

Entrons dans l'onglet « Général », deux zones d'informations concernant les paramètres prises en compte à l'exécution s'affichent.

Des boutons « Démarrer » ou « Arrêter » peuvent être actifs selon l'état de chaque périphérique.

En parlant de l'onglet « Paramètres », des cases à cocher sont disponibles pour activer les sous champs intégrés. Ces champs serviront pour la sélection des dates et des jours.

Les boutons « charger > » sont activés automatiquement pour chaque changement de données effectué.

La fenêtre principale dispose deux menus :

- *« Option »* comportant les sous menus suivants :

  - *« Actualiser » :* charge immédiatement la date et l'heure en cours dans tous les champs de données.
  - *« Arrêter » :* stoppe toute exécution en cours.
  - *« Fermer » :* ferme la fenêtre.

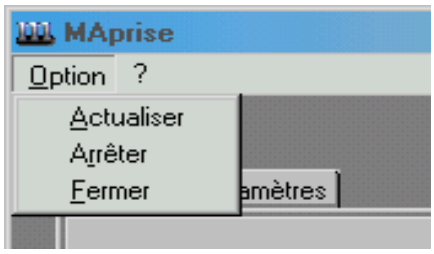

Figure 3.11 : Aperçu du menu « Option »

- « *?* » comportant à son tour :

  - *« A propos … » :* affiche les détails de la version de l'application.
  - *« Présentation » :* donne une information utile concernant l'application.

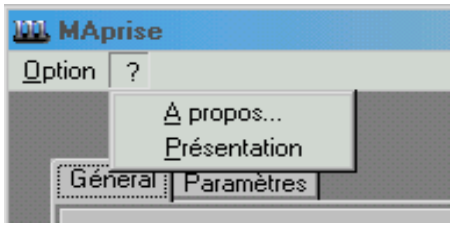

Figure 3.12 : Aperçu du menu « ? »

Pour fermer, il est possible de cliquer sur le bouton « Quitter ». Une exception est possible de se présenter en cas de mégarde. Si nous tentons de fermer la fenêtre en cours de processus, une boîte de dialogue d'information s'affiche (voir Figure 3.15). Il est alors impérativement nécessaire de laisser terminer ou d'arrêter tous processus en marche afin de quitter l'application.
Voici la boîte d'avertissement pour le cas de l'exception :

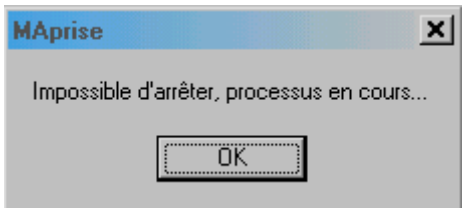

Figure 3.13 : Boîte d'avertissement

Pour terminer, cette application est fait en ayant la caractéristique de prendre en charge toutes mauvaises manipulations possibles. Il convient de noter qu'il est impossible d'agrandir ou de rétrécir la fenêtre principale.
Les paramètres introduits sont recueillis et traités par l'application. Cette dernière renvoie les commandes sur le port parallèle au moment opportun. Les commandes servent à ouvrir ou à fermer les deux sources électriques programmables.

3.3 Estimation du coût de la réalisation

Le tableau estimatif ci-contre récapitule le coût de la réalisation :

Designations des tâches	**Coûts estimatifs**
Composants électroniques	50 000Ar
Main d'œuvre	50 000Ar
Programme	800 000Ar
Divers	20 000Ar
Estimation totale du coût de la réalisation	**920 000Ar**

Tableau 3.02 : Estimation du coût de la réalisation

CONCLUSION GENERALE

Notre recherche intitulée « domotique » donne un aspect global tant sur la programmation informatique que sur la réalisation électronique. Ce mémoire présente un aperçu général des connaissances nécessaires à l'aboutissement de la domotique. Certains détails peuvent ne pas être inclus. Toutefois, ce mémoire nous inspire et nous oriente vers la domotique elle-même.

Le micro-ordinateur a été inventé pour faciliter les méthodes de travail dans le but d'ordonner, de traiter ou encore de mémoriser des données informatiques à une vitesse très élevée. Parallèlement, un support physique est capital pour extérioriser ces données. Cette étude nous a permis de conclure qu'il est impossible de tout faire d'un seul coup.

En ce qui concerne le programme, il y a toujours des améliorations à apporter. C'est ainsi qu'il existe de nombreuses versions renouvelées qui sont indispensables pour répondre aux demandes des utilisateurs. De même pour le périphérique qui sert de support physique, nous pouvons apporter quelques retouches aux composants utilisés, par exemple les mettre à jour, ou encore créer un nouveau design plus convivial. Autrement dit, les conceptions du point de vue technique et l'aspect physique doivent être améliorées pour arriver à un bon résultat, sécurisant et abordable aux utilisateurs.

En étant compatible avec d'autres langages de programmation, l'assembleur est un grand outil pour optimiser un programme élaboré en langage de haut niveau comme le C++. De ce fait, il rend les programmes plus rapides et moins encombrants.

Personnellement, ce travail nous a fait comprendre au mieux le micro-ordinateur et tout ce qui l'entoure. Il nous a également servi d'outil de simulation dans le monde du travail. Des améliorations s'avèrent évidemment nécessaires si nous voulons aboutir à une bonne application.

Finalement, la domotique est un sujet intéressant pour ceux qui sont à la recherche d'une vie aisée et confortable. C'est une technologie qui mérite d'être maîtrisée, et elle pourra même nous conduire à la création d'une nouvelle chose, voire une maison intelligente. L'arrivée proche de la fibre optique promet un avenir plus sophistiqué. Alors pensez déjà à la domotique.

A N N E X E S

Annexe 1 Programme d'initialisation

Le programme d'initialisation élaboré avec le langage de développement Borland C++ Builder, incluant un module en assembleur, est le suivant :

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
        //initialiser le port parallèle
        asm{
                mov dx,0x378
                mov al,0x0
                out dx,al
        }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
        //fermer le programme
        Close();
}
//---------------------------------------------------------------------------
```

Annexe 2 Programme principal

Le programme principal élaboré avec le langage de développement Borland C++ Builder, incluant des modules en assembleur, est le suivant :

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit01.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
 void Init()
        {
            asm {
                    mov dx,0x378
                    mov al,0x0
                    out dx,al
                    }
        }
void Dem1()
        {
            asm {
                     mov dx,0x378
                     in al,dx
                     or al,0xc
                     out dx,al
                   }
        }
void Arr1()
        {
                asm {
```

```
                        mov dx,0x378
                        in al,dx
                        and al,0x3
                        out dx,al
                    }
        }
void Dem2()
        {
                asm {
                        mov dx,0x378
                        in al,dx
                        or al,0x3
                        out dx,al
                    }
        }
void Arr2()
        {
                asm {
                        mov dx,0x378
                        in al,dx
                        and al,0xc
                        out dx,al
                    }
        }
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{ }
//---------------------------------------------------------------------------
void __fastcall TForm1::BClick(TObject *Sender)
{
        // Affichage de date&heure actuelles
          A->Caption=Date();
```

```
        B->Caption=Time();
        /* actualiser par le menu "actualiser",
        (actualiser Onclick sur fromcreate)*/
        if((Button4->Visible) || (Button6->Visible)==true)
                Actualiser1->Enabled=false;
        else
                Actualiser1->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
        if((Button4->Visible) || (Button6->Visible)==true)
                ShowMessage(AnsiString("Impossible d'arrêter, processus en cours..."));
        else
                Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{
        // Se commuter sur "arrêter"
        Button6->Visible=true;
        Button5->Visible=false;
        Button5->Enabled=false;
        dem2->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
        // Se commuter sur "démarrer" et le rendre active
        Button5->Visible=true;
        Button6->Visible=false;
        Arr2();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
        // Se commuter sur "arrêter"
        Button4->Visible=true;
        Button3->Visible=false;
        Button3->Enabled=false;
        dem1->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
        // Se commuter sur "démarrer" et le rendre active
        Button3->Visible=true;
        Button4->Visible=false;
        Arr1();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Arreter1Click(TObject *Sender)
{
         Init();        // Arrêter et initialiser les périphériques
        if(Button4->Visible==true)   // Afficher les deux "Démarrer"
        {
                Button3->Visible=true;
                Button4->Visible=false;
        }
        if(Button6->Visible==true)
        {
                Button5->Visible=true;
                Button6->Visible=false;
        }
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
        {
         // Paramétrages par défaut dans FormCreate
         Init();
        // Actualiser si "démarrer" est désactivé
          if(Button3->Enabled==false)
         {
                DateTimePicker1->Date=Date();
                Label5->Caption = DateToStr(DateTimePicker1->Date);

                DateTimePicker2->Time=Time();
                Label6->Caption = TimeToStr(DateTimePicker2->Time);

                DateTimePicker3->Date=Date();
                Label7->Caption = DateToStr(DateTimePicker3->Date);

                DateTimePicker4->Time=Time();
                Label8->Caption = TimeToStr(DateTimePicker4->Time);
         }
          if(Button5->Enabled==false)
         {
                DateTimePicker5->Date=Date();
                Label9->Caption = DateToStr(DateTimePicker5->Date);

                DateTimePicker6->Time=Time();
                Label10->Caption = TimeToStr(DateTimePicker6->Time);

                DateTimePicker7->Date=Date();
                Label11->Caption = DateToStr(DateTimePicker7->Date);

                DateTimePicker8->Time=Time();
                Label12->Caption = TimeToStr(DateTimePicker8->Time);
         }
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        // Charger les nouvelles valeurs dans périphérique1
        Label5->Caption = DateToStr(DateTimePicker1->Date);
        Label6->Caption = TimeToStr(DateTimePicker2->Time);
        Label7->Caption = DateToStr(DateTimePicker3->Date);
        Label8->Caption = TimeToStr(DateTimePicker4->Time);
        // activer "Démarrer" du périphérique1 dans général
        // et à condition que "périphérique1" est coché
        if(CheckBox2->Checked==true)
                Button3->Enabled=true;
        else
                Button3->Enabled=false;
        // Désactiver "charger >" après chargement des valeurs
        Button1->Enabled=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{
        // Chargement des nouvelles valeurs dans périphérique2
        Label9->Caption = DateToStr(DateTimePicker5->Date);
        Label10->Caption = TimeToStr(DateTimePicker6->Time);
        Label11->Caption = DateToStr(DateTimePicker7->Date);
        Label12->Caption = TimeToStr(DateTimePicker8->Time);

        // Activer "Démarrer" du périphérique1 dans général et à condition que "périphérique1"
        // est coché
        if(CheckBox3->Checked==true)
                Button5->Enabled=true;
        else
                Button5->Enabled=false;
```

```
        // Désactiver "charger >" après chargement des valeurs
        Button7->Enabled=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
        // Gérer les champs internes du péripherique1
        if(CheckBox2->Checked==true)
                {   DateTimePicker1->Enabled=true;
                    DateTimePicker2->Enabled=true;
                    DateTimePicker3->Enabled=true;
                    DateTimePicker4->Enabled=true;
                }
        else
                {   DateTimePicker1->Enabled=false;
                    DateTimePicker2->Enabled=false;
                    DateTimePicker3->Enabled=false;
                    DateTimePicker4->Enabled=false;
                }

        // Activer "Charger >"  à chaque mouvement
        Button1->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox3Click(TObject *Sender)
{
        // Gérer les champs internes du péripherique2
        if(CheckBox3->Checked==true)
                {   DateTimePicker5->Enabled=true;
                    DateTimePicker6->Enabled=true;
                    DateTimePicker7->Enabled=true;
                    DateTimePicker8->Enabled=true;
```

```
                }
        else
                {   DateTimePicker5->Enabled=false;
                    DateTimePicker6->Enabled=false;
                    DateTimePicker7->Enabled=false;
                    DateTimePicker8->Enabled=false;
                }
        // Activer "Charger >"  à chaque mouvement
        Button7->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::dem1Timer(TObject *Sender)
{
        if(DateToStr(DateTimePicker1->Date)<= DateToStr(Date()))
                if(TimeToStr(DateTimePicker2->Time)<= TimeToStr(Time()))
                        {
                            Dem1();
                             arr1->Enabled=true;
                             dem1->Enabled=false;
                        }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::arr1Timer(TObject *Sender)
{
        if(DateToStr(DateTimePicker3->Date)<= DateToStr(Date()))
                if(TimeToStr(DateTimePicker4->Time)<= TimeToStr(Time()))
                {
                        Arr1();
                        Button3->Visible=true;
                        Button4->Visible=false;
                        arr1->Enabled=false;
                }
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::dem2Timer(TObject *Sender)
{
        if(DateToStr(DateTimePicker5->Date)<= DateToStr(Date()))
                if(TimeToStr(DateTimePicker6->Time)<= TimeToStr(Time()))
                {
                        Dem2();
                        arr2->Enabled=true;
                        dem2->Enabled=false;
                }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::arr2Timer(TObject *Sender)
{
        if(DateToStr(DateTimePicker7->Date)<= DateToStr(Date()))
                if(TimeToStr(DateTimePicker8->Time)<= TimeToStr(Time()))
                {
                        Arr2();
                        Button5->Visible=true;
                        Button6->Visible=false;
                        arr2->Enabled=false;
                }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Apropos1Click(TObject *Sender)
{
                 A_propos->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Pagedaceuil1Click(TObject *Sender)
{
         system ("start Présentation.txt");
}
```

Annexe 3 Tableau de correspondance entre base 10, base 2 et base 16

Décimal Base 10	**Binaire Base 2**	**Hexadécimal Base 16**
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Tableau A : Correspondance entre base 10, base 2 et base 16

BIBLIOGRAPHIE

[1] http://www.wikipedia.org

[2] http://www.commentcamarche.net

[3] *Aide,* disponible sur Microsoft Windows XP professionnel service pack 2

[4] K. Irvine, *Assembleur x86* - édition 2003

[5] http:// www.iprezo.org

[6] http://www.develloper.com

[7] http://www.develloper.com

[8] *Aide,* disponible sur Borland C++ Builder 5

[9] http://www.progzone.free.fr

[10] http://crteknologies.free.fr

[11] Quillet, *Encyclopédie des sciences industrielles : électricité, électronique, application*

édition 1973

[12] http://www.sonelec-musique.com

[13] http://www.lelectronique.com

[14] Z. Andriamiasy, *Circuit logique 1*, cours 1ère année, Dép. TCO – ESPA, A.U. : 2003/2004

RENSEIGNEMENTS

Nom : **ANDRIANAIVOSOA**

Prénom : **Falinirina**

Titre du mémoire : ***DOMOTIQUE AVEC ORDINATEUR***

Nombre de pages : 55

Nombre de tableaux : 05

Nombre de figures : 24

Mots clés : domotique, réseau informatique, port parallèle, C++, asm, transistor en commutation, relais, porte logique

Directeur de mémoire : Monsieur RATSIMBAZAFY Andriamanga

Adresse de l'auteur: II N 182 D Soavinandriana Besarety - Antananarivo 101 – Madagascar

E-mail : falifalikoa@yahoo.fr

RESUME

La domotique est l'automatisation des appareils domestiques. Elle requiert l'utilisation des différentes techniques comme la Télécommunication. Cette recherche présente la domotique fonctionnant à l'aide d'un ordinateur, et éventuellement d'un réseau informatique.
Pour cela, les maîtrises des généralités sur la domotique, sur le réseau informatique ainsi que sur la programmation sont indispensables. De même, programmation et éléments matériels sont interdépendants pour l'usage de la domotique. De ce fait, une mise en pratique de cette recherche nous semble être le moyen le plus adéquat pour concrétiser notre étude.

ABSTRACT

The "Domotique" or the home automation is the system of domestically tools which requires the use of many techniques such as the Telecommunication. This book is about the realization of the home automation using a computer and a network.
For this, the masteries of the generalities on the home automation, on the computer network as well as on the programming are indispensable. In the same way, programming and material elements are interdependent for the use of the home automation. Of this fact, a realization seems for us to be most adequate means for its concretization.