INTRODUCTION

La Robotique fait partie des sciences des objets et des systèmes artificiels. Elle peut être vue comme la science de la perception et du mouvement et de leur intégration en une machine physique, mécanique et informatique.

Un robot est donc un système matériel possédant des capacités de perception, d'action, de décision et de communication, parfois capable d'améliorer ses propres performances par apprentissage automatique ou supervisé par des humains, pour :

- agir dans un environnement ouvert ou confiné, dynamique et imparfaitement modélisé, voire inconnu, à des échelles allant du nanomonde au macromonde ;
- exécuter de façon autonome ou en relation avec un humain, des tâches d'observation, d'exploration, de modélisation, de manipulation et/ou d'intervention sur l'environnement ;
- interagir le cas échéant avec d'autres machines ou avec des humains, matériellement ou virtuellement.

Au plan scientifique, la Robotique est dotée de plusieurs grands journaux internationaux (*IEEE Trans. on Robotics* **3**, *Int. Journal of Robotics Research, Int. Journal of Robotics and Autonomous Systems)* (à moindre titre : *Int. Journal of Advanced Robotics Systems, Int. Journal of Robotics and Mecatronics,*) et de plusieurs conférences internationales dont les deux plus importantes: *IEEE Int. Conf. on Robotics and Automation, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

Si la Robotique non manufacturière a été longtemps considérée comme un champ scientifique très prospectif dont les applications, en dehors de la conquête spatiale, semblaient futuristes, les progrès scientifiques et technologiques importants intervenus dans les grandes disciplines invoquées par la Robotique (mécanique, électronique, automatique, informatique) ont considérablement ouvert le spectre de ses applications, de plus en plus tangibles et crédibles.

On en trouve ci-dessous un résumé d'analyses de quelques domaines particuliers ou applications de la robotique

- **Mini-robots de précision :** ils sont conçus spécialement pour les applications de micro montage et les installations très compactes difficiles à manipuler par l'homme. Ils sont rapides, précis et très fiables destinés à répondre à toutes les contraintes posées par les opérations de précision de positionnement et d'assemblage en micromécanique.

- **Robots autonomes :** ils sont des robots capables de se mouvoir seuls sans assistance externes. Ceci implique qu'ils doivent se repérer sur l'aire de travail et être capable de s'adapter aux conditions du terrain qui est souvent adapté et balisé. On les conçoit sous différentes formes : véhicules, androïdes, tortues, tripodes, araignées, ou Aibo le chien.
- **Robots anthropomorphes :** ils sont des robots a forme humaine pour remplacer l'homme, et très souvent autonomes. Ils sont équipés de capteurs perfectionnés; tels que caméras, télémètres laser, organes de perception de force, etc. Ils disposent de techniques avancées de reconnaissance vocale, reconnaissance de formes, traitement d'images et derniers développements en informatique (Intelligence Artificielle).
- **Bras manipulateurs :** ils sont les équipements les plus utilisés au milieu industriel, ils sont conçus pour saisir et manipuler des objets avec ou sans assistance humaine. Ils réalisent des tâches répétitives et souvent pénibles pour l'homme. Ils sont rapides, précis, fidèles et manipulent des objets au milieu inaccessible, ou des produits dangereux ou au milieu nocif pour l'être humain.
- **Robotique terrestre et aérienne :** Les robots considérés ici se déplacent dans des environnements peu ou pas structurés, souvent naturels, et peu ou pas connus/modélisés, et se meuvent seuls ou en groupe coopératif.
- **Véhicules intelligents :** Les véhicules considérés ici sont ceux dédiés au transport des personnes ou des marchandises. On peut distinguer les véhicules routiers (autoroutes et routes) et les véhicules urbains (avenues et rues).
- **Radiolocalisation pour la robotique mobile :** Les évolutions dans les toute prochaines années en matière de technologie GPS de positionnement, notamment par la mise en place de GALILEO, feront que le positionnement centimétrique en extérieur sera possible partout à un faible coût. En intérieur et dans des milieux très enfouis, des solutions sont actuellement à l'étude dont l'hybridation avec des systèmes de télécommunications (verrous : stratégie de fusion multicapteurs pour la localisation ; utilisation pour la commande de robots d'intérieur par couplage avec des radars ULB - Ultra Large Band).
- **La Biorobotique :** (appelée parfois robotique bio-inspirée) a pour but de transférer dans des systèmes matériels (robots) ce qu'on a appris du comportement du monde animal, voire végétal, et des traitements de l'information qui y sont associés.
- **La Robotique hybride** vise à incorporer des parties animales dans des robots.

- **La Neurobotique** s'intéresse au couplage entre le système nerveux d'un être vivant (animal ou homme) et une machine, pour commander des systèmes robotiques (lien avec la Robotique hybride).

Les applications majeures actuelles concernent la défense, la sécurité civile, l'exploration planétaire, et à moindre titre l'agriculture ou l'environnement, ce dernier secteur étant susceptible de prendre une importance accrue dans le futur.

L'objectif de ce projet, c'est la maîtrise de la haute technologie au service de l'industrie pour l'étude, la conception et la réalisation de tous types de robots.

PARTIE I

ETUDE THÉORIQUE

Chapitre 1 : Présentation du projet

1. Définition du projet

Le robot élaboré doit être capable de suivre une bande noire d'une largeur définie, en totale autonomie et le plus rapidement possible, tout en gérant sa vitesse, sa direction et un système anti-collision.

Nous développons ce projet pour un mémoire de projet de fin d'étude. Le but de ce projet est de mettre en application tous nos acquis. L'objectif principal est donc de concevoir, de réaliser, de régler et monter les différents constituants sur le robot.

Figure1 : Robot FSR 2008

Fonctionnement : Un interrupteur permet de mettre le robot sous tension et un jack le fait démarrer. Pour l'arrêter il suffit d'appuyer sur le rupteur ou sur le bouton d'arrêt d'urgence. Toutefois, il est nécessaire de le programmer pour qu'il puisse fonctionner, car il s'agit d'un robot autonome.

2. Solutions possibles et solutions retenues

2.1 Partie matérielle

Détection de ligne	
Possibilités	Caméras, capteurs
Choix	Capteurs au nombre de huit
Justification	La caméra était plus difficile à gérer dans la durée proposée du projet. La manipulation facile des capteurs avec le microcontrôleur pic16f84 nous a permet de les choisir.

Commande des moteurs	
Possibilités	Pont en H de type N ou H, transistors de puissance montés en « «push-pull »
Choix	Pont en H de type H
Justification	Le pont en H est plus facile à gérer. Ce type a été retenu pour raison de disponibilité immédiate.

2.2 Partie logicielle

Programmation suivi de ligne et commande des moteurs	
Possibilités	Programmation en assembleur ou en langage C
Choix	Assembleur MPASM
Justification	Ce langage est le plus puissant, plus simple et plus rapide pour programmer les microcontrôleurs.

3. Cahier des charges fonctionnel

- **Fonctions principales**
  - Démarrer
  - Suivre une ligne noire (scotch ou peinture) de largeur 10 mm sur un sol blanc.
  - S'arrêter
- **Fonctions liées aux règlements proposés**
  - Avoir des certaines dimensions par exemple : 30 cm (longueur) x 20 cm (largeur) x 40 cm (hauteur).
  - Etre autonome en énergie : utiliser la batterie fournie pour la partie motorisation
  - Démarrer grâce au jack fourni
  - Retrouver la ligne en cas de sortie de piste
  - Faire un tour à un croisement
  - Respecter les priorités à droite

    ➔ Détecter la priorité à droite

    ➔ Ralentir

    ➔ Détecter la présence d'un obstacle
- Si présence d'un obstacle :

  + 1ère fois faire un demi-tour

  + 2ème fois arrêter

  + Détecter le passage du robot

  + Redémarrer et reprendre le suivi de piste
- Si absence d'un obstacle, accélérer

4. Analyse fonctionnelle

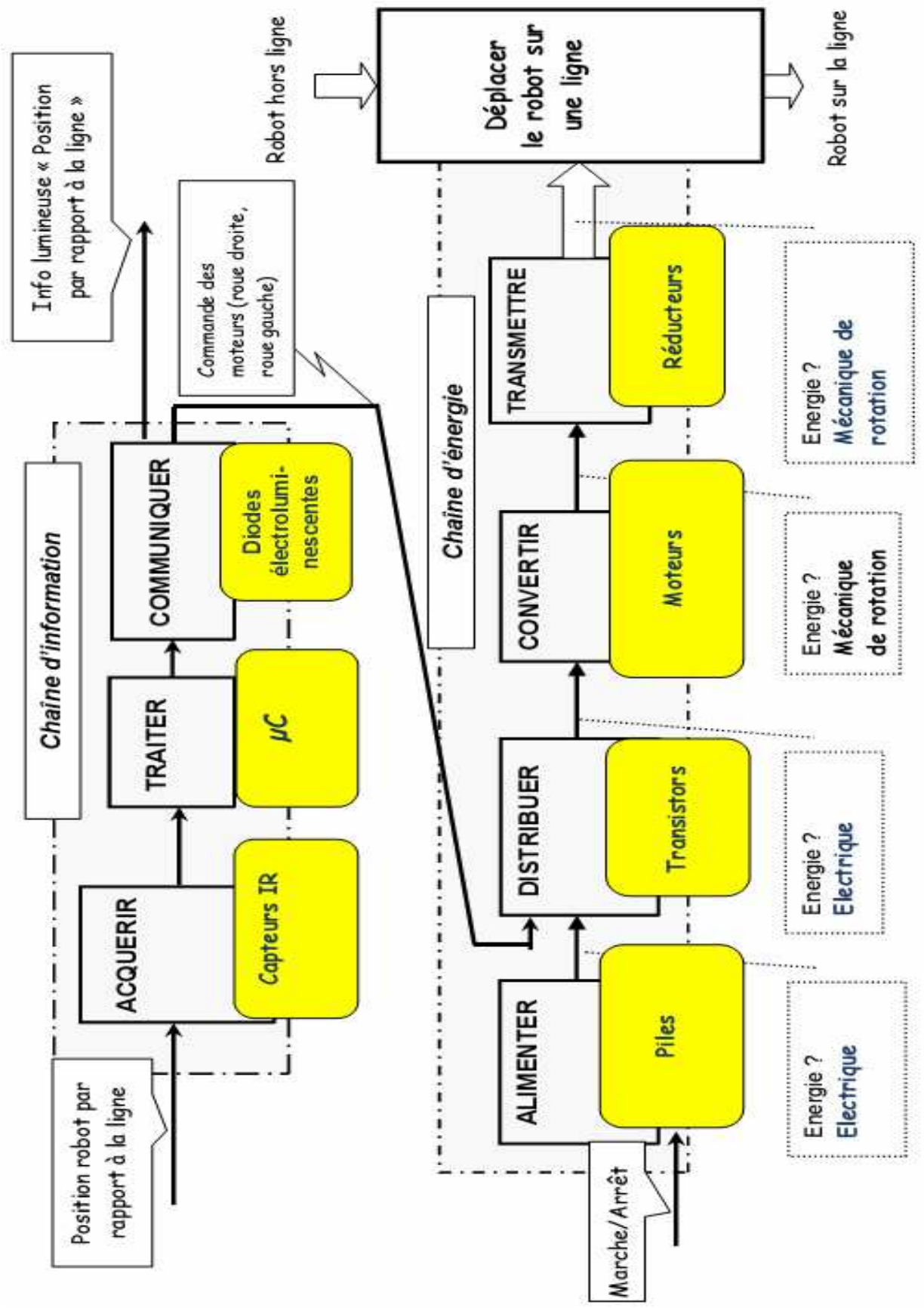


Figure 2 : Identification des solutions constructives

5. La chaîne d'information

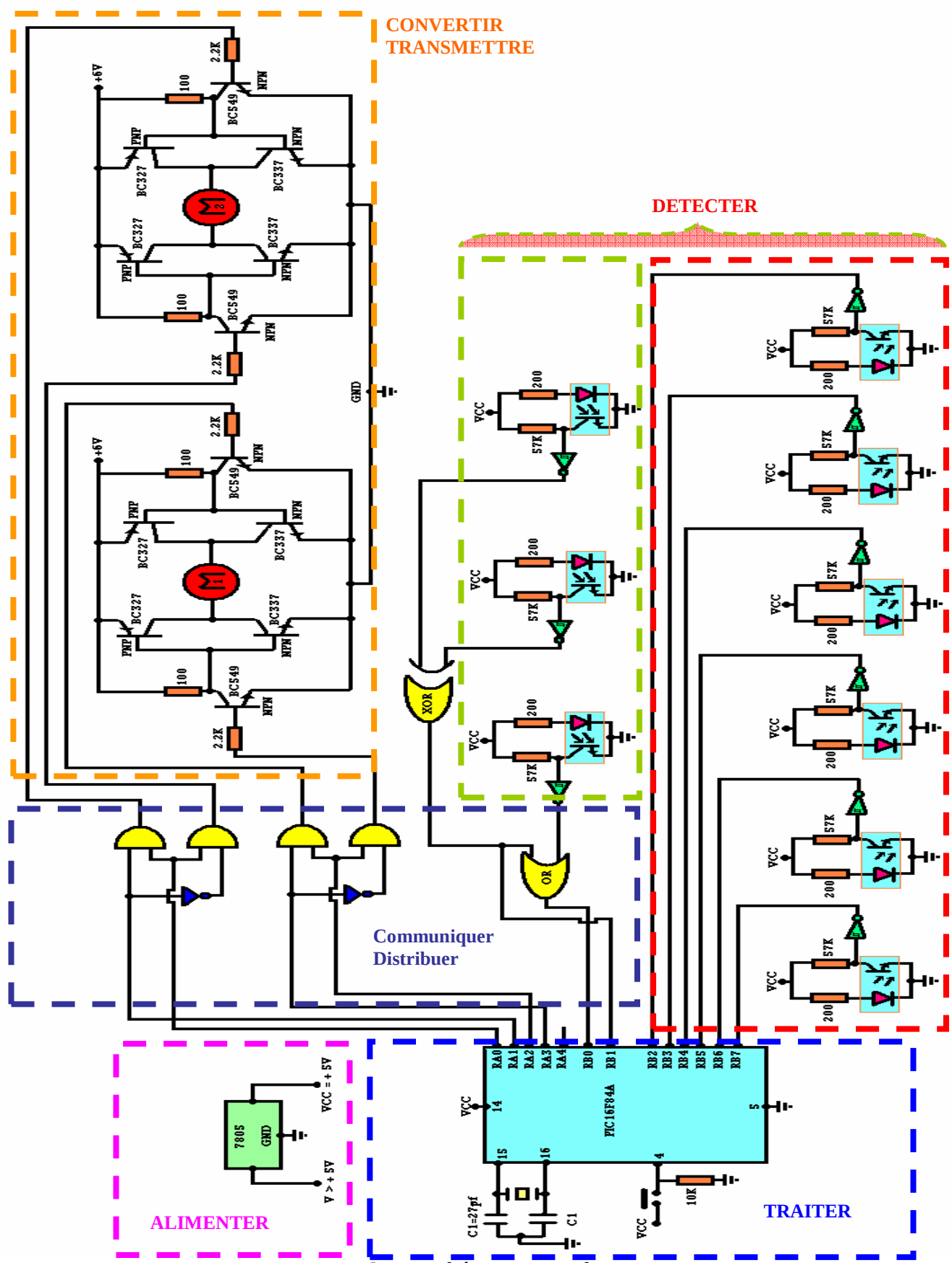
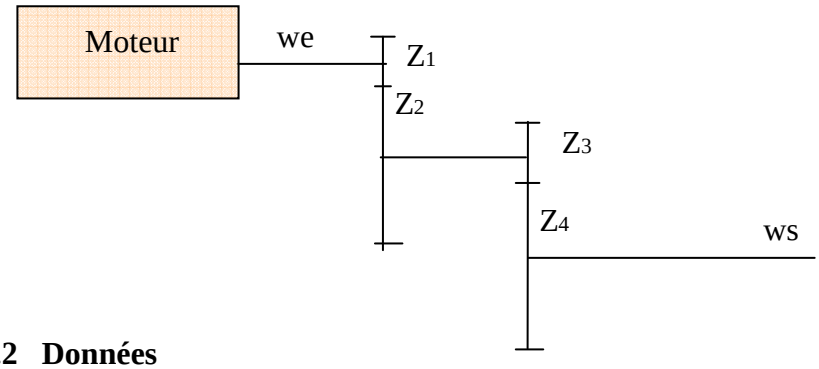


Figure 3 : Le schéma structurel

6. La chaîne d'énergie

6.1 Schéma



6.2 Données

Z_1 = 20 dents
Z_2 = 50 dents
Z_3 = 20 dents
Z_4 = 50 dents

we = 7000 tr/min pour une tension continu de 3V.

6.3 Formule

$$ws/we = (-1)^n * [(Z_1/Z_2)*(Z_3/Z_4)]$$

6.4 Application Numérique

$ws/we = (-1)^2 * [(2/5)x(2/5)]$
$=1(0.4x0.4)$
$=1(0.16)$
$ws = 0.16 * we$
$= 0.16 * 7000$
$ws = 1120$ tr/min

Le signe positif indique qu'il n'y a pas inversion du sens de rotation entre l'arbre d'entrée (moteur) et l'arbre de sortie.

6.5 Avantages

- Ces engrenages développent un couple suffisamment puissant pour faire tourner les roues d'un robot.
- A partir d'un petit moteur assez petit on peut s'en servir pour tracter ou porter un objet.

6.6 Inconvénient

- L'augmentation du couple réduit la vitesse de rotation.

6.7 Intérêt

- Le couple est très intéressant.

7. Conception du robot

Ce robot est équipé des yeux électroniques qui observent une ligne noire et détectent les obstacles. Chaque oeil électronique est constitué d'un capteur infrarouge de type OBP704.

Le robot doit suivre une ligne noire de 10 mm de large sur fond blanc.

C'est un tripode constitué d'une roue folle placée à l'avant et de deux moteurs, accouplés à deux roues, situés à l'arrière.

Les moteurs assurent la propulsion et la direction du robot. Des capteurs optiques situés à l'avant permettent la détection de la ligne.

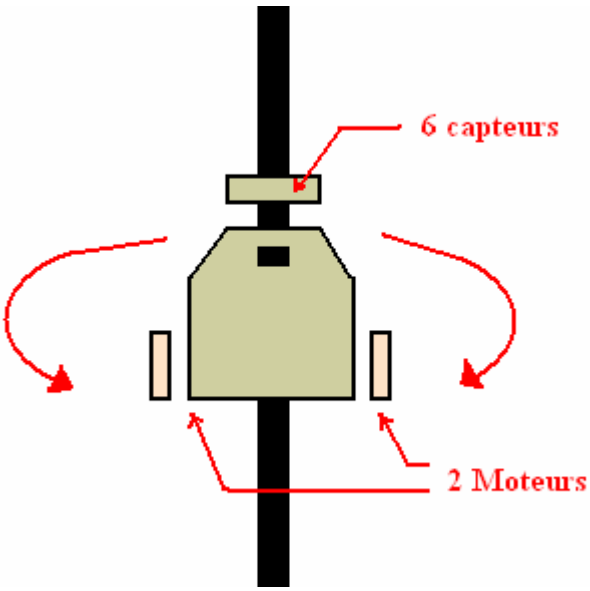
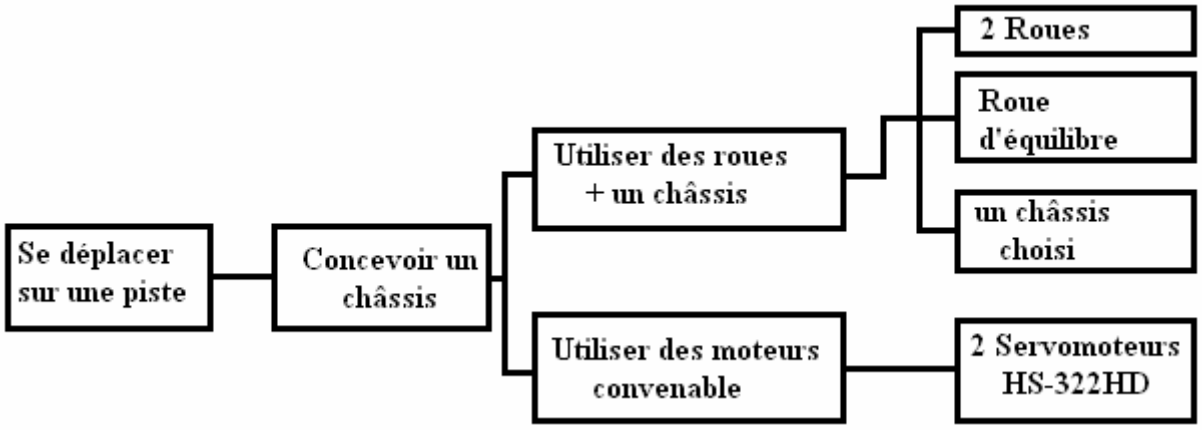
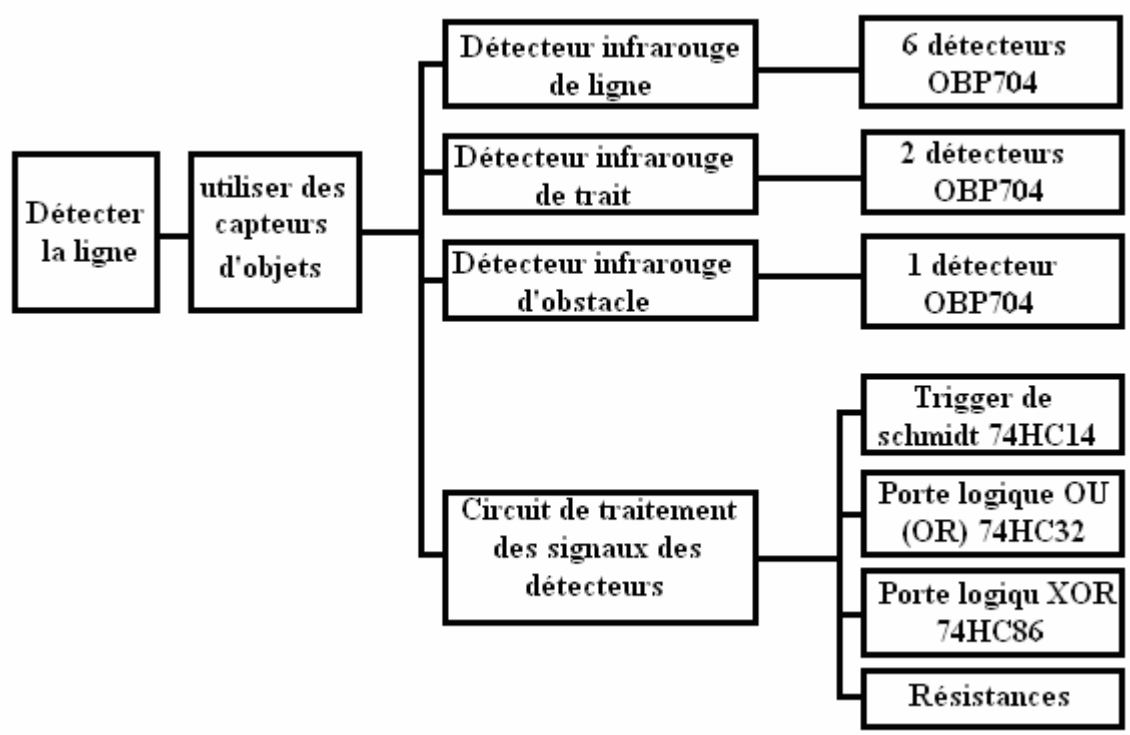


Figure 4 : Le robot sur la ligne

8. Diagramme





9. Schéma de principe de la réalisation

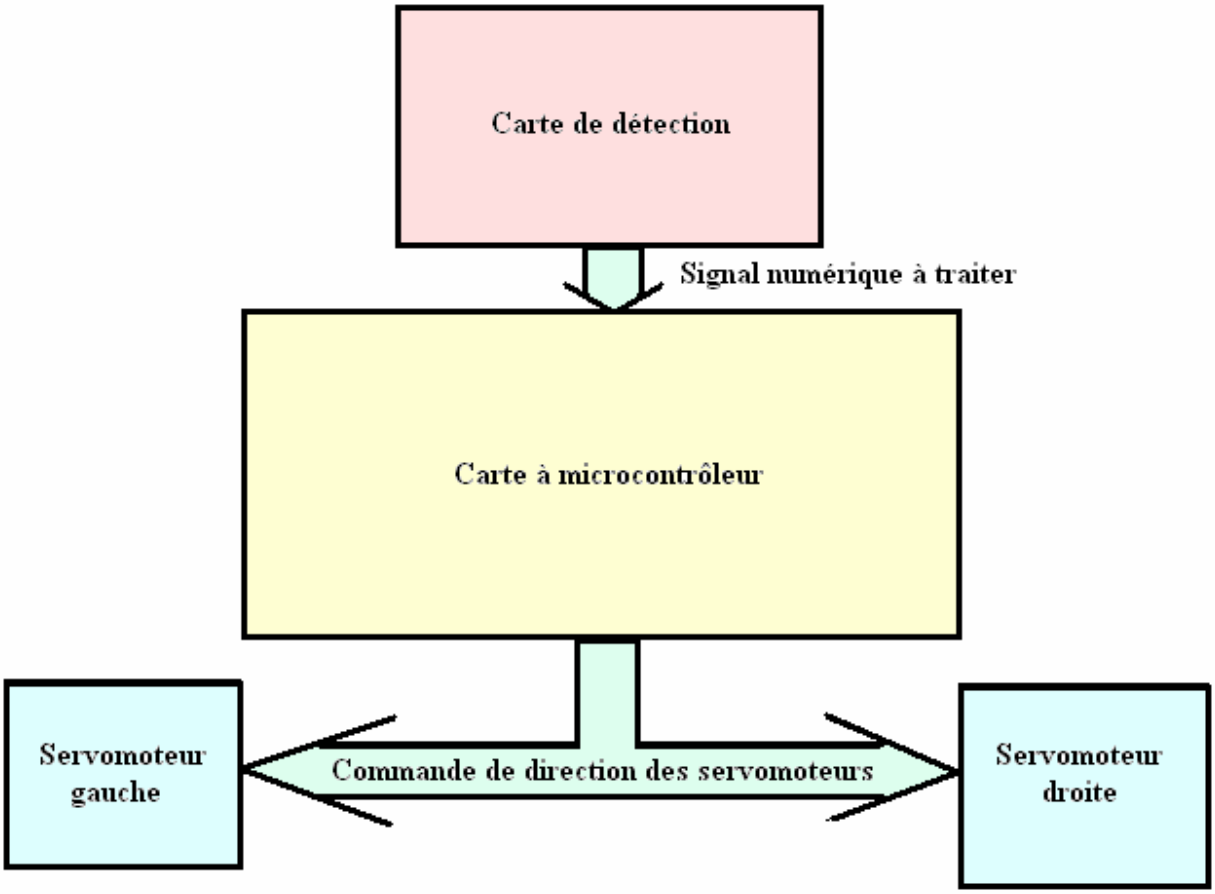


Figure 5 : Schéma synoptique

10. Schéma électronique

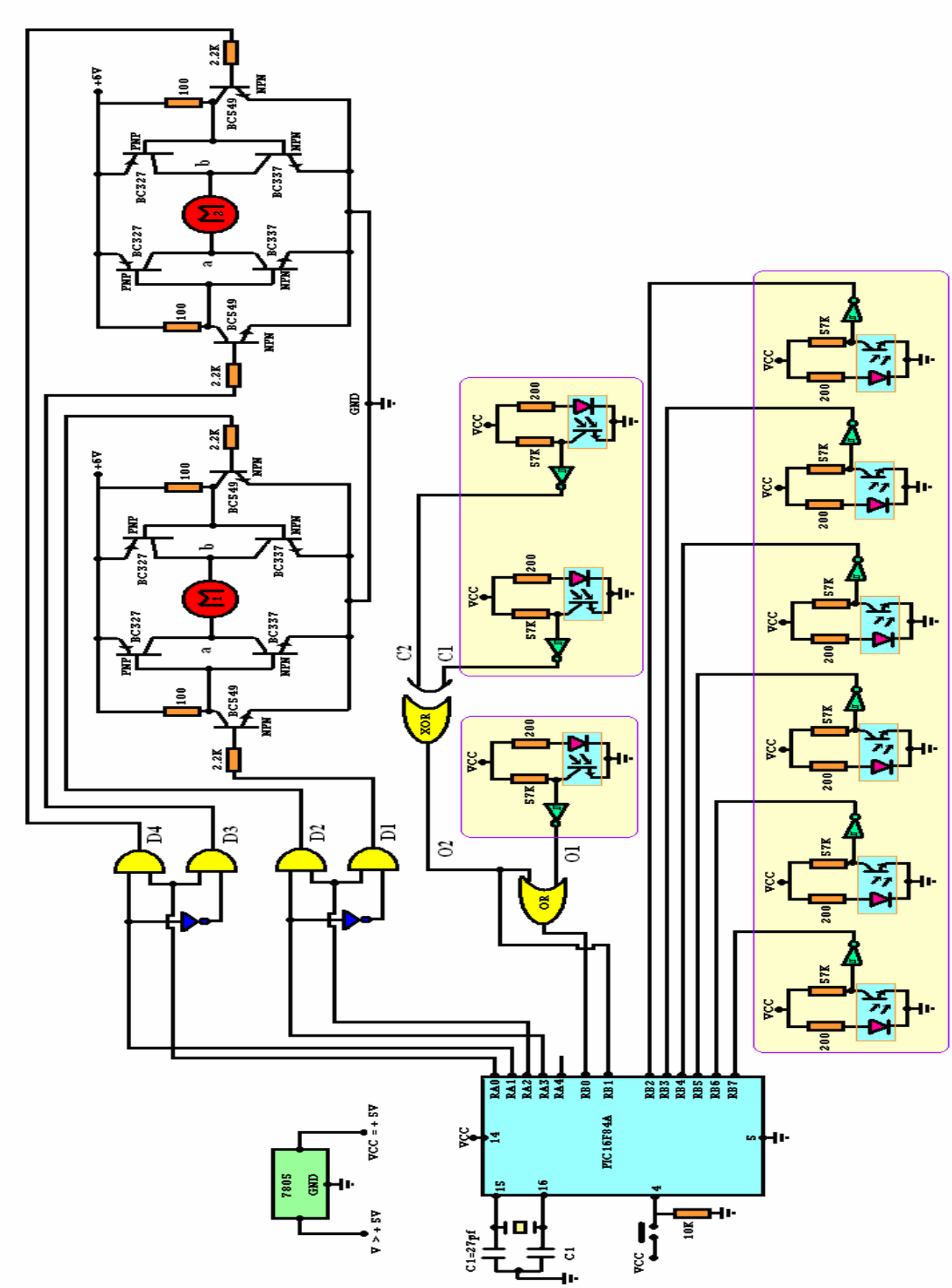


Figure 6 : Schéma électronique du robot EBESA

Le schéma de ce robot peut être décomposé en trois parties : les capteurs, l'interface de microcontrôleur et l'interface vers les moteurs. Chaque partie sera étudié individuel à la partie pratique.

Le fonctionnement de ces capteurs (OBP704) a été expliqué dans les parties qui suivent. Il est basé sur la quantité de photons reçus par le phototransistor. Un carénage est prévu pour éviter autant que possible les perturbations dues à la lumière extérieure.

La partie de microcontrôleur sert à traiter les données collectées par les capteurs.

La dernière partie est l'interface des deux moteurs. Le courant fournit par le pic est faible ce qui demande d'utiliser des transistors pour amplifier le courant. La ponte H permet de commander les moteurs dans un sens normal ou inverse, marche ou arrêt.

11. Le microcontrôleur

Le microcontrôleur choisi est le PIC 16F84 de Microchip. Ce microcontrôleur a été choisi en raison de sa popularité. Son prix reste correct au vu de ses capacités; Microchip propose un environnement de développement gratuit (MPLAB) et la réalisation d’un programmateur est simple et économique. De nombreux sites Internet proposent des logiciels de programmation gratuits et des exemples d’applications et parfois des cours d’initiation en français.

Le site Internet de Microchip est : www.microchip.com, on y trouvera la documentation du PIC16F84 et la dernière version de MPLAB.

Chapitre 2 : Étude détaillé sur les microcontrôleurs pic16f84

1. Introduction

Depuis l'arrivée du premier microprocesseur dans les années 70 (déjà plus de 38 ans), conçu par INTEL, les progrès en intégration de composants n 'ont cessé d 'évoluer, faisant de ce fait augmenter la puissance de calcul et la rapidité des microprocesseurs. Ce premier microprocesseur le "4004" deviendra peut être célèbre, en effet la sonde pionner 10 lancée dans les années 70 a quitté le système solaire en emportant avec elle le fameux circuit.

Un microprocesseur peut être assimilé à un circuit logique complexe exécutant un à un des ordres (instructions) enregistrés dans une mémoire de programme externe. Un microprocesseur ne fonctionne jamais seul on lui associe toujours des périphériques d'entrées- sorties afin de pouvoir exécuter une fonction particulière et dialoguer avec "l'extérieur" (prise en compte de l'état d'un capteur, commande d'un relais, mémoire programme de l'application, etc...).

Le grand avantage de cette logique programmable est que la modification d'une fonction ou d'une tâche ne nécessite pas de câblage supplémentaire, mais uniquement un nouveau programme à loger en mémoire.

L'arrivée du microcontrôleur qui est un microprocesseur auquel on a intégré les périphériques d'entrées sorties, va faire que les montages et applications deviennent encore plus simples à mettre en oeuvre, avec un gain de temps, de vitesse et un coût réduit (le pic 16f84 est à environ 50 DH).

Le PIC16F84 de la société Microchip est un micro contrôleur faisant partie de la famille RISC (Reduced Instruction set Computer) dont les caractéristiques sont vitesse d 'exécution et jeu d 'instruction réduit (le pic 16f84 ne comporte que 35 instructions).

2. LES MICROCONTROLEURS

2.1. Qu'est ce qu'un microcontrôleur

C'est un ordinateur monté dans un circuit intégré. Les avancées technologiques en matière d'intégration, ont permis d'implanter sur une puce de silicium de quelques millimètres carrés la totalité des composants qui forment la structure de base d'un ordinateur. Leur prix varie de quelques dizaines Dirhams à un millier de Dirhams pour les plus complexes. Comme tout ordinateur, on peut décomposer la structure interne d'un microprocesseur en trois parties :

-Les mémoires
-Le processeur
-Les périphériques

C'est ce qu'on peut voir sur la figure 6 :

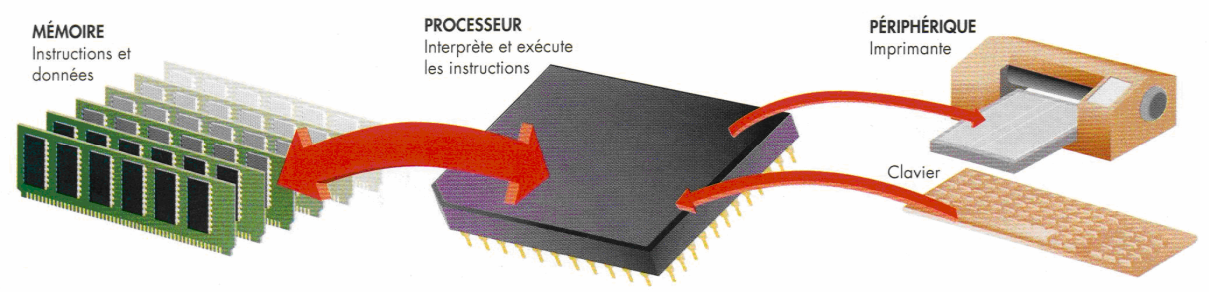


Figure 7

- les mémoires sont chargées de stocker le programme qui sera exécuté ainsi que les données nécessaires et les résultats obtenus
- le processeur est le cœur du système puisqu'il est chargé d'interpréter les instructions du programme en cours d'exécution et de réaliser les opérations qu'elles contiennent .Au sein du processeur, l'unité arithmétique et logique interprète, traduit et exécute les instructions de calcul.
- les périphériques ont pour tâche de connecter le processeur avec le monde extérieur dans les deux sens . Soit le processeur fournit des informations vers l'extérieur (périphérique de sortie), soit il en reçoit (périphérique d'entrée).

2.2. Intérêt des microcontrôleurs

Les microcontrôleurs sont de taille tellement réduite qu'ils peuvent être sans difficulté implantés sur l'application même qu'ils sont censés piloter. Leur prix et leurs performances simplifient énormément la conception de système électronique et informatique. L'utilisation des microcontrôleurs ne connaît de limite que l'ingéniosité des concepteurs, on les trouve dans nos cafetières, les magnétoscopes, les radios …..Une étude menée en l'an 2004 montre qu'en moyenne, un foyer américain héberge environ 240 microcontrôleurs.

3. PRESENTATION GENERALE DU PIC 16F84

3.1 Classification du PIC 16F84

Le PIC 16F84 est un microcontrôleur 8 bits. Il dispose donc d'un bus de données de huit bits. Puisqu'il traite des données de huit bits, il dispose d'une mémoire de donnée dans laquelle chaque emplacement (défini par une adresse) possède huit cases pouvant contenir chacune un bit.

3.2 Architecture interne

La structure générale du PIC 16F84 comporte 4 blocs comme le montre la figure 8 :

-Mémoire de programme
-Mémoire de données
-Processeur
-Ressources auxiliaires (périphériques)

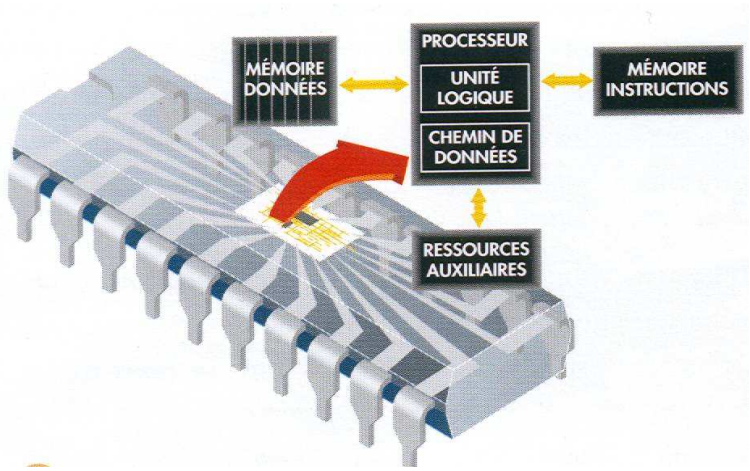


Figure 8

- La mémoire de programme contient les instructions pilotant l'application à laquelle le microcontrôleur est dédié. Il s'agit d'une mémoire non volatile (elle garde son contenu, même en l'absence de tension), elle est de type FLASH c'est à dire qu'elle peut être programmée et effacée par l'utilisateur via un programmateur et un PC. La technologie utilisée permet plus de 1000 cycles d'effacement et de programmation. Pour le PIC 16F84 cette mémoire est d'une taille de 1024*14 bits, c'est à dire qu'elle dispose de 1024 emplacements (de 000h à 3FFh) contenant chacun 14 cases car dans le cas du PIC, les instructions sont codées sur 14 bits. On peut donc stocker 1024 instructions.
- La mémoire de donnée est séparée en deux parties :
  -une mémoire RAM de 68 octets puisque le bus de donnée est de huit bits. Cette RAM est volatile (les données sont perdues à chaque coupure de courant). On peut y lire et écrire des données.
  -une mémoire EEPROM de 64 octets dans laquelle on peut lire et écrire des données (de huit bits soit un octet) et qui possède l'avantage d'être non volatile (les données sont conservées même en l'absence de tension). La lecture et l'écriture dans cette mémoire de données sont beaucoup plus lentes que dans la mémoire de données RAM.

- Le processeur est formé de deux parties :
  -une unité arithmétique et logique (UAL) chargée de faire des calculs.
  -un registre de travail noté W sur lequel travail l'UAL.

- Les ressources auxiliaires qui sont dans le cas du PIC16F84
  -ports d'entrées et de sorties.
  -temporisateur.
  -interruptions
  -chien de garde
  -mode sommeil

Ces ressources seront analysées dans la suite.

4. STRUCTURE INTERNE DU PIC 16F84

4.1 Brochage et caractéristiques principales

Le PIC16F84 est un circuit intégré de 18 broches (figure 9) :

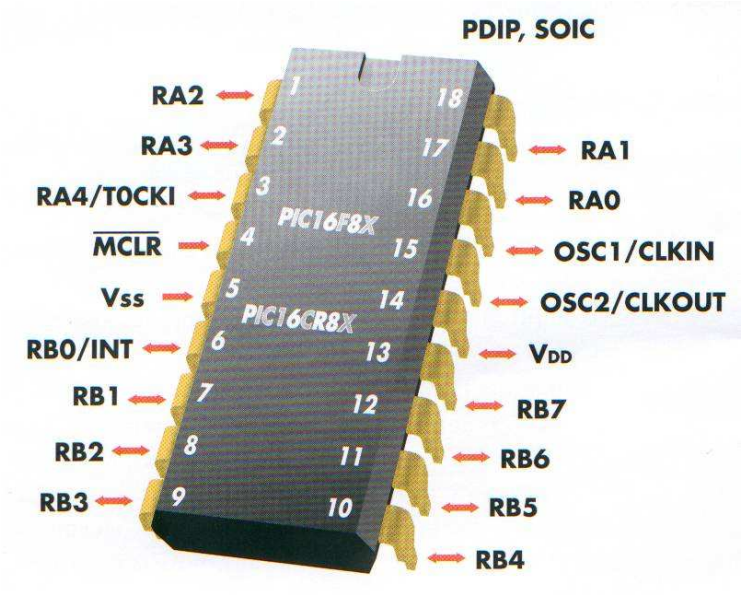


Figure 9

- L'alimentation du circuit est assurée par les pattes VDD et VSS. Elles permettent à l'ensemble des composants électroniques du PIC de fonctionner. Pour cela on relie VSS (patte 5) à la masse (0 Volt) et VDD (patte 14) à la borne positive de l'alimentation qui doit délivrer une tension continue comprise entre 3 et 6 Volts.

- Le microcontrôleur est un système qui exécute des instructions les unes après les autres à une vitesse (fréquence) qui est fixée par une horloge interne au circuit. Cette horloge doit être stabilisée de manière externe au moyen d'un cristal de quartz connecté aux pattes OSC1/CLKIN (patte 16) et OSC2/CLKOUT (patte 15). Nous reviendrons en détail sur l'horloge au paragraphe 3.

- La patte 4 est appelée $\overline{\text{MCLR}}$. Elle permet lorsque la tension appliquée est égale à 0V de réinitialiser le microcontrôleur. C'est à dire que si un niveau bas (0 Volt) est appliqué sur $\overline{\text{MCLR}}$ le microcontrôleur s'arrête, place tout ses registres dans un état connu et se redirige vers le début de la mémoire de programme pour recommencer le programme au début (adresse dans la mémoire de programme :0000).

A la mise sous tension, la patte $\overline{\text{MCLR}}$ étant à zéro, le programme démarre donc à l'adresse 0000, (MCLR=Master Clear Reset).

- Les broches RB0 à RB7 et RA0 à RA4 sont les lignes d'entrées/sorties numériques. Elles sont au nombre de 13 et peuvent être configurées en entrée ou en sortie. Ce sont elles qui permettent au microcontrôleur de dialoguer avec le monde extérieur (périphériques). L'ensemble des lignes RB0 à RB7 forme le port B et les lignes RA0 à

RA4 forment le port A. Certaines de ces broches ont aussi d'autres fonctions (interruption, timer).

4.2. Structure interne

La structure interne du PIC16F84 est donnée figure 10 : (structure HARVARD : la mémoire de programme et la mémoire de données sont séparées contrairement à l'architecture Von Neumann qui caractérise d'autres fabricants de microcontrôleurs)

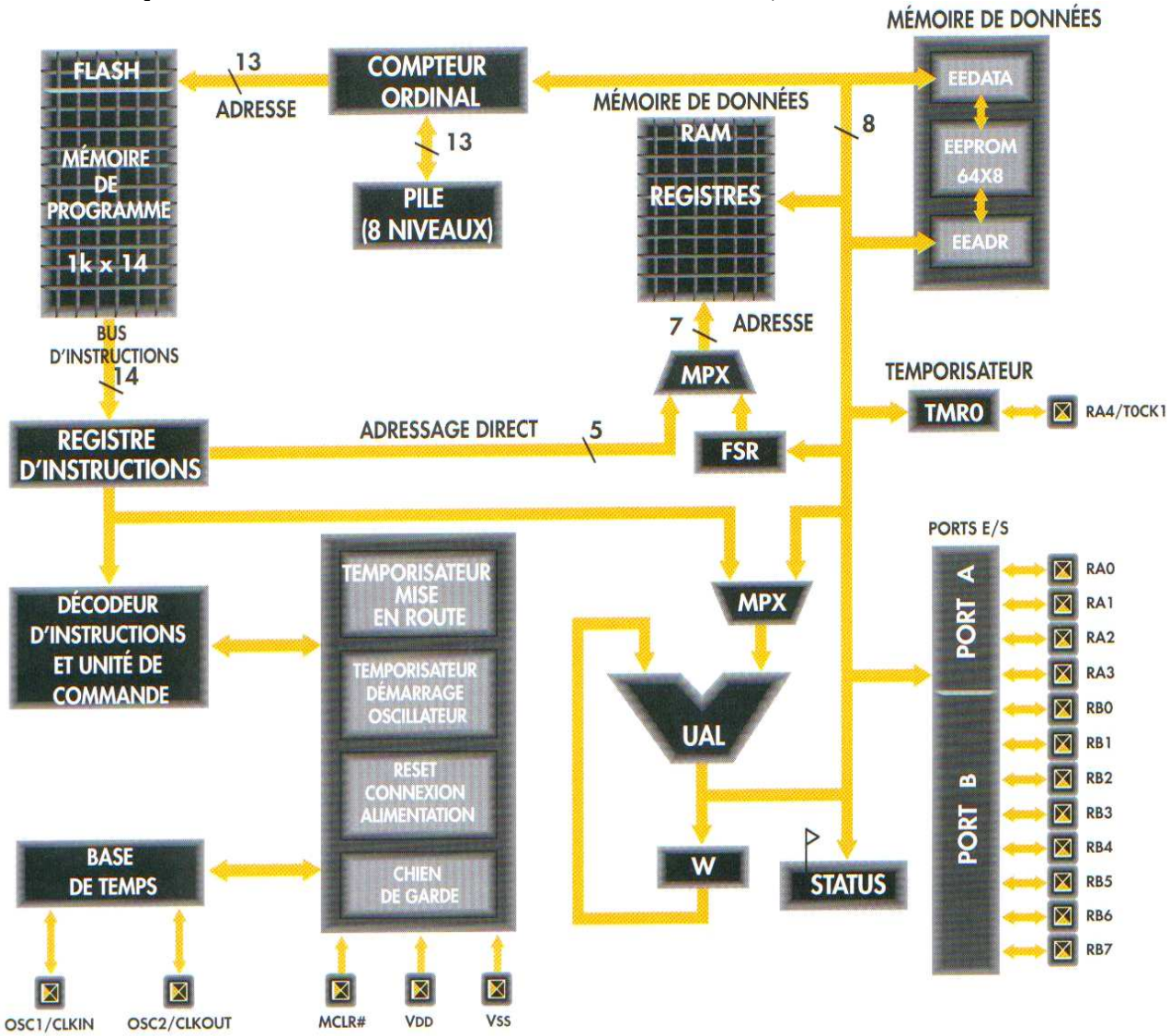


Figure 10

On retrouve sur ce schéma la mémoire de programme, la mémoire RAM de données, la mémoire EEPROM, les ports A et B, ainsi que la partie processeur avec l'UAL et le registre de travail W (work). Nous allons étudier à présent plus en détail le fonctionnement du PIC.

4.3. Principe de fonctionnement du PIC

Un microcontrôleur exécute des instructions. On définit « le cycle instruction » comme le temps nécessaire à l'exécution d'une instruction. Attention de ne pas confondre cette notion avec le cycle d'horloge qui correspond au temps nécessaire à l'exécution d'une opération élémentaire (soit un coup d'horloge).

Une instruction est exécutée en deux phases :

- La phase de recherche du code binaire de l'instruction stocké dans la mémoire de programme
- La phase d'exécution ou le code de l'instruction est interprété par le processeur et exécuté.

Chaque phase dure 4 cycles d'horloge comme le montre la figure 11 :

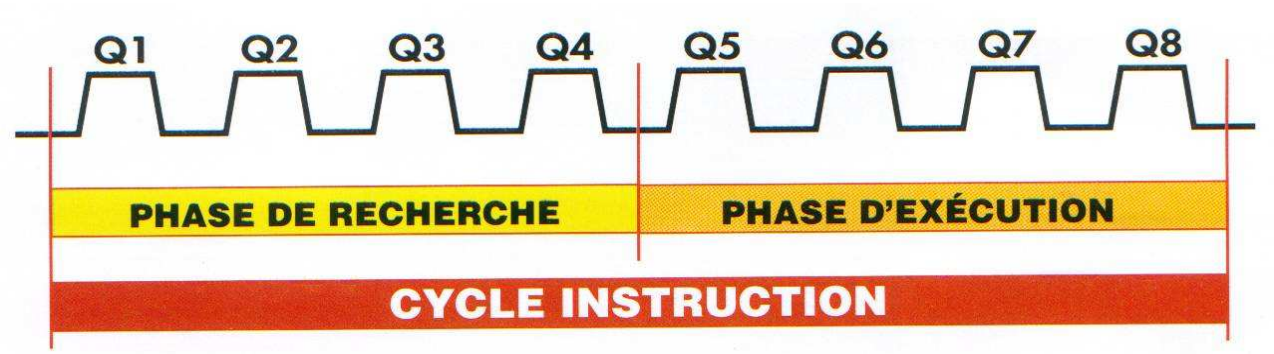


Figure 11

On pourrait donc croire qu'un cycle instruction dure 8 cycles d'horloge mais l'architecture particulière du PIC lui permet de réduire ce temps par deux. En effet, comme les instructions issues de la mémoire de programme circulent sur un bus différent de celui sur lequel circulent les données, ainsi le processeur peut effectuer la phase de recherche d'une instruction pendant qu'il exécute l'instruction précédente (Voir figure 12 et 13).

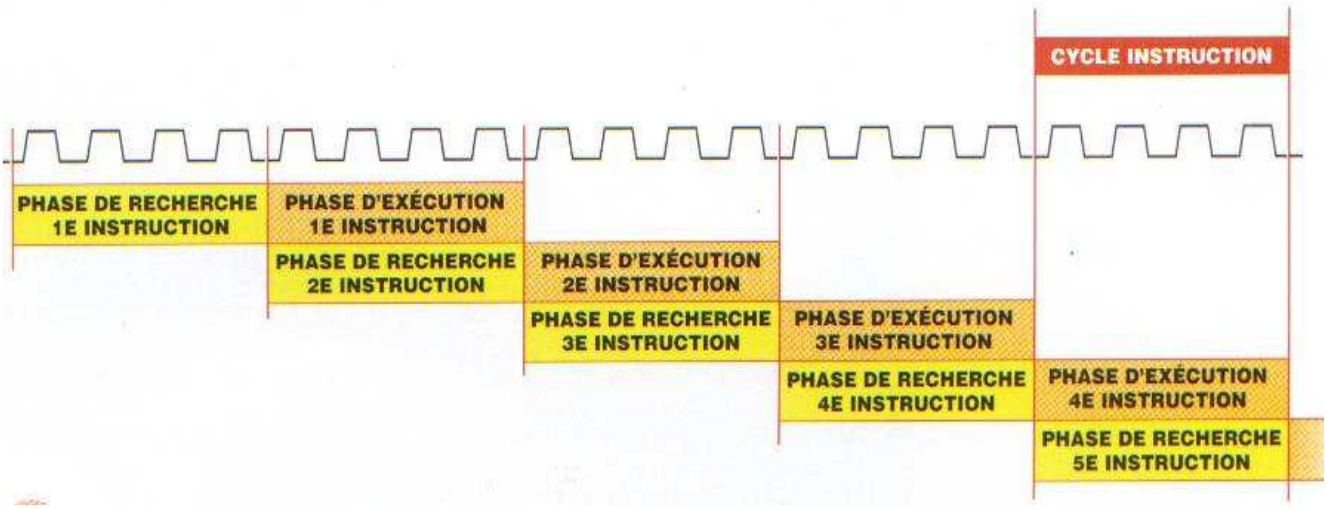


Figure 12

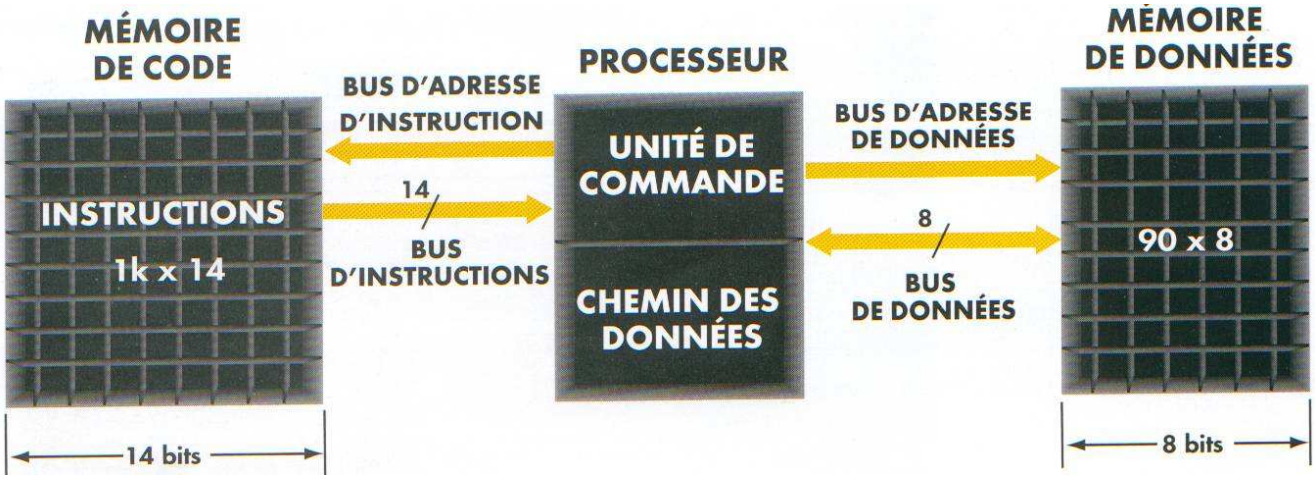


Figure 13

4.4. Déroulement d'un programme

Le déroulement d'un programme s'effectue de façon très simple. A la mise sous tension, le processeur va chercher la première instruction qui se trouve à l'adresse 0000 de la mémoire de programme, l'exécute puis va chercher la deuxième instruction à l'adresse 0001 et ainsi de suite (sauf cas de saut ou d'appel de sous programme que nous allons voir plus loin). On parle de fonctionnement séquentiel. La figure 14 va nous permettre de mieux comprendre le fonctionnement :

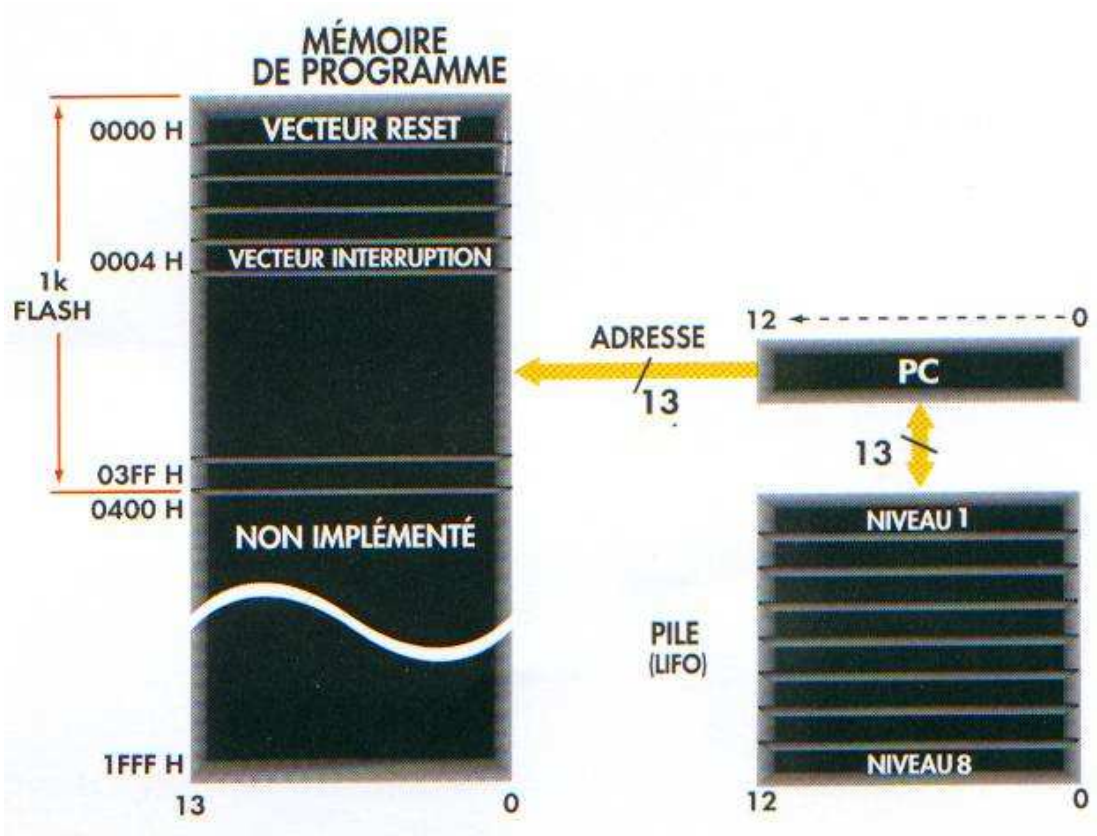


Figure 14

- On constate sur cette figure que la mémoire de programme contient 1024 emplacements (3FF en hexadécimale) contenant 14 bits (de 0 à 13). Une instruction occupe un emplacement qui est défini par une adresse. Le processeur peut alors sélectionner l'emplacement souhaité grâce au bus d'adresse et il peut lire son contenu (ici l'instruction) grâce à son bus d'instruction (voir figure 7). Cet adressage s'effectue à l'aide d'un compteur ordinal appelé PC qui lors de la mise sous tension démarre à zéro puis s'incrémente de 1 tous les quatre coups d'horloge, on exécute bien ainsi les instructions les unes à la suite des autres.
- Mais il arrive que dans un programme on fasse appel à un sous programme dont l'adresse de l'instruction ne se trouve pas juste après celle qui est en train d'être exécutée. C'est le rôle de la pile qui sert à emmagasiner de manière temporaire l'adresse d'une instruction. Elle est automatiquement utilisée chaque fois que l'on appelle un sous programme et elle permet une fois que l'exécution du sous programme est terminée de retourner dans le programme principal juste après l'endroit où l'on a appelé le sous programme. On constate que cette pile possède huit niveaux, cela signifie qu'il n'est pas possible d'imbriquer plus de huit sous programmes, car au-delà de huit, le processeur ne sera plus capable de retourner à l'adresse de base du programme principal.
- L'adresse 0000 est réservée au vecteur RESET, cela signifie que c'est à cette position que l'on accède chaque fois qu'il se produit une réinitialisation (0 volts sur la patte MCLR). C'est pour cette raison que le programme de fonctionnement du microcontrôleur doit toujours démarrer à cette adresse.
- L'adresse 0004 est assignée au vecteur d'interruption et fonctionne de manière similaire à celle du vecteur de Reset. Quand une interruption est produite et validée, le compteur ordinal PC se charge avec 0004 et l'instruction stockée à cet emplacement est exécutée.

4.5. La mémoire de données RAM

Si l'on regarde la mémoire de donnée RAM, on s'aperçoit que celle-ci est un peu particulière comme le montre la figure 15 :

On constate en effet que cette mémoire est séparée en deux pages (page 0 et page 1). De plus, on remarque que tant pour la page 0 que pour la page 1, les premiers octets sont réservés (SFR pour Special File Register). Ces emplacements sont en effet utilisés par le microcontrôleur pour configurer l'ensemble de son fonctionnement. On les appelle registres spécifiques et nous verrons au chapitre suivant leurs rôles.

Le bus d'adresse qui permet d'adresser la RAM est composé de 7 fils ce qui veut dire qu'il est capable d'adresser 128 emplacements différents. Or, chaque page de la RAM est composée de 128 octets, le bus d'adresse ne peut donc pas accéder aux deux pages, c'est pourquoi on utilise une astuce de programmation qui permet de diriger le bus d'adresse soit sur la page 0, soit sur la page 1. Cela est réalisé grâce à un bit d'un registre spécifique (le bit RP0 du registre STATUS) dont nous verrons le fonctionnement plus loin.

La RAM de données proprement dite se réduit donc à la zone notée GPR (Registre à usage générale) qui s'étend de l'adresse 0Ch (12 en décimale) jusqu'à 4Fh (79 en décimale), soit au total 68 registres en page 0 et autant en page 1, mais on constate que les données écrites en page 1 sont redirigées en page 0 cela signifie qu'au final l'utilisateur dispose uniquement de 68 registres (donc 68 octets de mémoire vive) dans lesquels il peut écrire et lire à volonté en sachant qu'à la mise hors tension, ces données seront perdues.

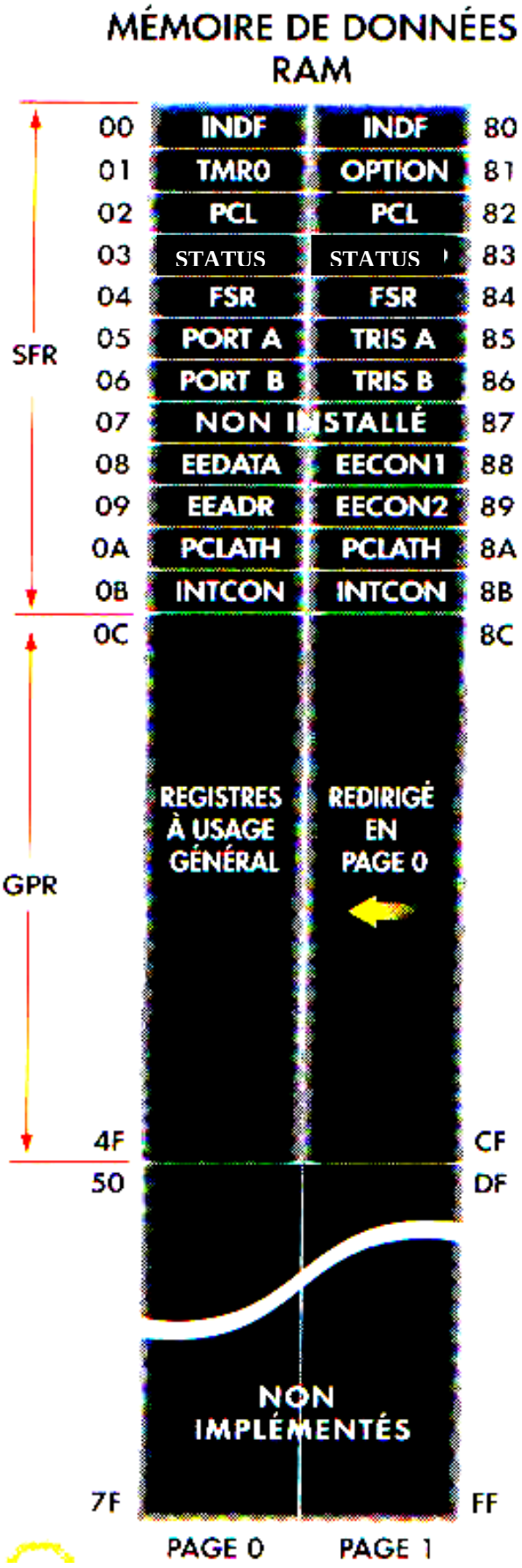

MÉMOIRE DE DONNÉES
RAM
SFR
GPR
00 INDF INDF 80
01 TMR0 OPTION 81
02 PCL PCL 82
03 STATUS STATUS 83
04 FSR FSR 84
05 PORT A TRIS A 85
06 PORT B TRIS B 86
07 NON INSTALLÉ 87
08 EEDATA EECON1 88
09 EEADR EECON2 89
0A PCLATH PCLATH 8A
0B INTCON INTCON 8B
0C 8C
REGISTRES À USAGE GÉNÉRAL
REDIRIGÉ EN PAGE 0
4F CF
50 DF
NON IMPLÉMENTÉS
7F FF
PAGE 0 PAGE 1

Figure 15

5. Les registres

Nous avons vu au chapitre précédent que la mémoire de données RAM contenait des registres spécifiques qui permettent de configurer le PIC, nous allons les détailler un à un et voir comment on peut accéder à la page 0 ou la page 1. Afin de faciliter la compréhension, les registres les plus utilisés sont encadrés.

5.1 adresse 00 et 80, INDF : Cette adresse ne contient pas de registre physique, elle sert pour l'adressage indirect. (non utilisée dans le projet de cette année)

5.2 adresse 01, TMR0 : Contenu du Timer (8 bits). Il peut être incrémenté par l'horloge (fosc/4) c'est à dire tous les 4 coups d'horloge ou par la broche RA4.

5.3 adresse 02 et 82, PCL : 8 bits de poids faibles du compteur ordinal PC. Les 5 (13-8) bits de poids forts sont dans PCLATH.

5.4 adresse 03 et 83, STATUS Registre d'état. :

Les cinq bits de poids faible de ce registre sont en lecture seule, ce sont des témoins (drapeaux ou flag en anglais) caractérisant le résultat de l'opération réalisée par l'UAL. Le bit RP0 est lui en lecture /écriture et c'est lui qui permet de sélectionner la page dans la mémoire RAM.

Si RP0=0 on accède à la page 0 et si RP0=1 on accède à la page 1.

RP0	TO/	PD/	Z	DC	C

Au reset, seul le bit RP0 de sélection de page est fixé (RP0=0 : page 0)
TO/ (Time Out) : débordement du timer WDT
PD/ : (Power Down) caractérise l'activité du chien de garde WDT
Z (zéro) résultat nul pour une opération arithmétique et logique.
DC (digit carry) retenue sur un quartet (4 bits)
C (carry) retenue sur un octet (8 bits).

5.5 adresse 04 et 84, FSR : Registre de sélection de registre : contient l'adresse d'un autre registre (adressage indirect, non utilisé dans le projet)

5.6 adresse 05, PORTA : Ce registre contient l'état des lignes du port À (voir paragraphe sur les ports).

5.7 adresse 06, PORTB : Ce registre contient l'état des lignes du port B (voir paragraphe sur les ports).

5.8 adresse 08, EEDATA : Contient un octet lu ou à écrire dans l'EEPROM de données.

5.9 adresse 09, EEADR : Contient l'adresse de la donnée lue ou écrite dans l'EEPROM de données.

adresse 0A et 8A, PCLATH : Voir l'adresse 02 PCL.

5.10 adresse 0B et 8B, INTCON : Contrôle des 4 interruptions

GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Masques :
GIE : (Global Interrupt Enable) : masque global d'inter.
EEIE : (EEProm Interrupt Enable) autorise l'interruption venant de l'EEPROM.
T0IE : (Timer 0 Interrupt Enable) autorise l'interruption provoquée par le débordement du TIMER0
INTE : (Interrupt Enable) autorise l'interruption provoquée par un changement d'état sur broche RB0/INT
RBIE : (RB Interrupt Enable) autorise les interruptions provoquées par un changement d'états sur l'une des broches RB4 à RB7.
Si ces bits sont mis à 1, ils autorisent les interruptions pour lesquels ils sont dédiés.
Drapeaux :
T0IF : (Timer 0 Interrupt Flag) débordement du timer
INTF (Interrupt Flag) interruption provoquée par la broche RB0/INT
RBIF (RB Interrupt Flag) interruption provoquée par les broches RB4-RB7.

5.11 adresse 81, OPTION
8 bits (tous à 1 au RESET) affectant le comportement des E/S et des timers.

RBPU/	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0

RBPU/ (RB Pull Up) Résistances de tirage à Vdd des entrées du port B (voir le détail du fonctionnement au chapitre port).Si RBPU/=0 les résistances de pull-up sont connectées en interne sur l'ensemble du port B.
INTEDG (Interrupt Edge) sélection du front actif de l'interruption sur RB0/INT (1 pour front montant et 0 pour front descendant).
RTS (Real Timer Source) sélection du signal alimentant le timer 0 : 0 pour horloge interne, 1 pour RA4/T0CLK
RTE (Real Timer Edge) sélection du front actif du signal timer (0 pour front montant).
PSA (Prescaler assignment) 0 pour Timer 0 et 1 pour chien de garde WDT.
PS2.0 (Prescaler 210) sélection de la valeur du diviseur de fréquence pour les timers.

5.12 adresse 85, TRISA : Direction des données pour le port A : 0 pour sortir et 1 pour entrer (voir paragraphe sur les ports).

5.13 adesse 86 , TRISB : Direction des données pour le port B : 0 pour sortir et 1 pour entrer (voir paragraphe sur les ports).

5.14 dresse 88, EECON1 Contrôle le comportement de l'EEPROM de données.

EEIF	WRERR	WREN	WR	RD

EEIF (EEProm Interrupt Flag) passe à 1 quand l'écriture est terminée.
WRERR (Write Error) 1 si erreur d'écriture.
WREN (Write Enable) : 0 pour interdire l'écriture en EEPROM de données.
WR (Write) 1 pour écrire une donnée. Bit remis automatiquement à 0
RD (Read) : 1 pour lire une donnée. Bit remis automatiquement à 0

5.15 adresse 89, EECON2. Registre de sécurité d'écriture en EEPROM de données.
Une donnée ne peut être écrite qu'après avoir écrit successivement 0x55 et 0xAA dans ce registre.

6. Les ports d'entrées/sorties

Le PIC16F84 est équipé de 13 lignes d'entrées/sorties réparties en deux ports :
-le port A : RA0 à RA4
-le port B : RB0 à RB7

Chaque ligne peut être configurée soit en entrée, soit en sortie, et ceci indépendamment l'une de l'autre. Pour cela on utilise les registres TRISA et TRISB. Le bit de poids faible (b0) du registre TRISA correspond à la ligne RA0, le bit b1 de TRISA correspond à RA1 et ainsi de suite. Il en est de même pour le port B et le registre TRISB (b0 de TRISB correspond à RB0 → b7 correspond à RB7). Si l'on veut placer une ligne en sortie il suffit de mettre le bit correspond dans TRISA ou TRISB à 0 (retenez 0 comme Output=sortie). Si l'on veut placer une ligne en entrée, il suffit de placer le bit correspondant dans TRISA ou TRISB à 1 (retenez 1 comme Input=entrée).

Les bits des deux registres PORTA et PORTB permettent soit de lire l'état d'une ligne si celle-ci est en entrée, soit de définir le niveau logique d'une ligne si celle-ci est en sortie.

Lors d'un RESET, toutes les lignes sont configurées en entrées.

- Particularité du portA : les bits b7 à b5 des registres TRISA et PORTA ne correspondent à rien car il n'y a que 5 lignes (b0 à b4). **RA4 est une ligne à collecteur ouvert**, cela veut dire que configurée en sortie cette broche assure 0Volt à l'état bas, mais qu'à l'état haut, il est nécessaire de fixer la valeur de la tension grâce à une résistance de tirage (pull up en anglais)
- Particularité du portB : il est possible de connecter de façon interne sur chaque ligne une résistance de tirage (pull up) dont le rôle consiste à fixer la tension de la patte (configuré en entrée) à un niveau haut lorsque qu'aucun signal n'est appliqué sur la patte en question. Pour connecter ces résistances, il suffit de placer le bit RBPU/ du registre OPTION à 0.

7. Le Timer

Dans la majeure partie des applications, il est nécessaire de contrôler le temps; afin de ne pas occuper le microcontrôleur qu'à cette tâche (boucle de comptage qui monopolise le micro), on le décharge en utilisant un timer. Le pic 16F84 dispose de deux timers, un à usage général (le TMR0) et un autre utilisé pour le chien de garde (watch dog WDG).

Le TMR0 est un compteur ascendant (qui compte) de 8 bits qui peur être chargé avec une valeur initiale quelconque. Il est ensuite incrémenté à chaque coup d'horloge jusqu'à ce que le débordement ait lieu (passage de FF à 00); Le principe est représenté figure 16 :

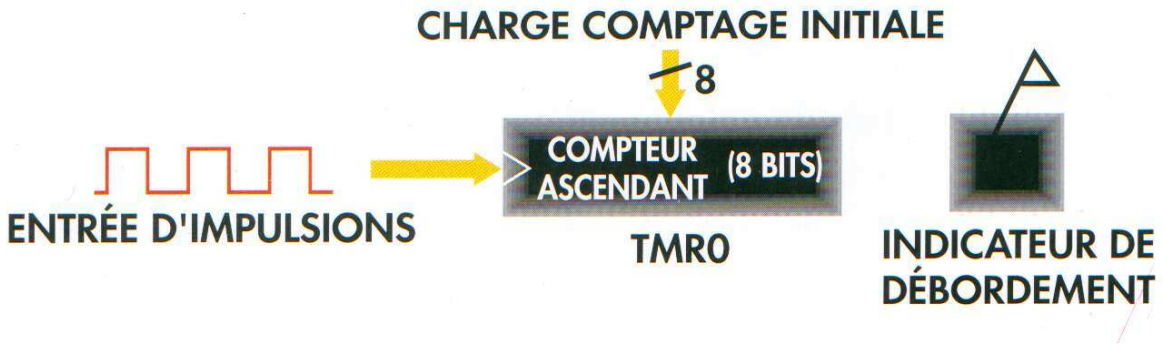


Figure 16

Le TMR0 peut remplir deux fonctions :

- Temporisateur ou Contrôle du temps son entrée d'incrémentation est alors l'horloge qui correspond au cycle instruction (Fosc/4). Il est possible d'utiliser un pré-diviseur de fréquence que nous verrons plus loin.
- Compteur d'événements. Dans ce cas les d'impulsions d'entrées du timer sont fournies par la patte RA4/TOCK1

Le choix s'effectue grâce au bit RTS du registre OPTION.

Le pic 16F84 dispose d'un diviseur de fréquence qui peut être assigné soit au chien de garde, soit au TMR0 (uniquement un à la fois). L'assignation du pré diviseur se fait grâce au bit PSA du registre OPTION. La structure interne du TMR0 est donc la suivante (figure 17) :

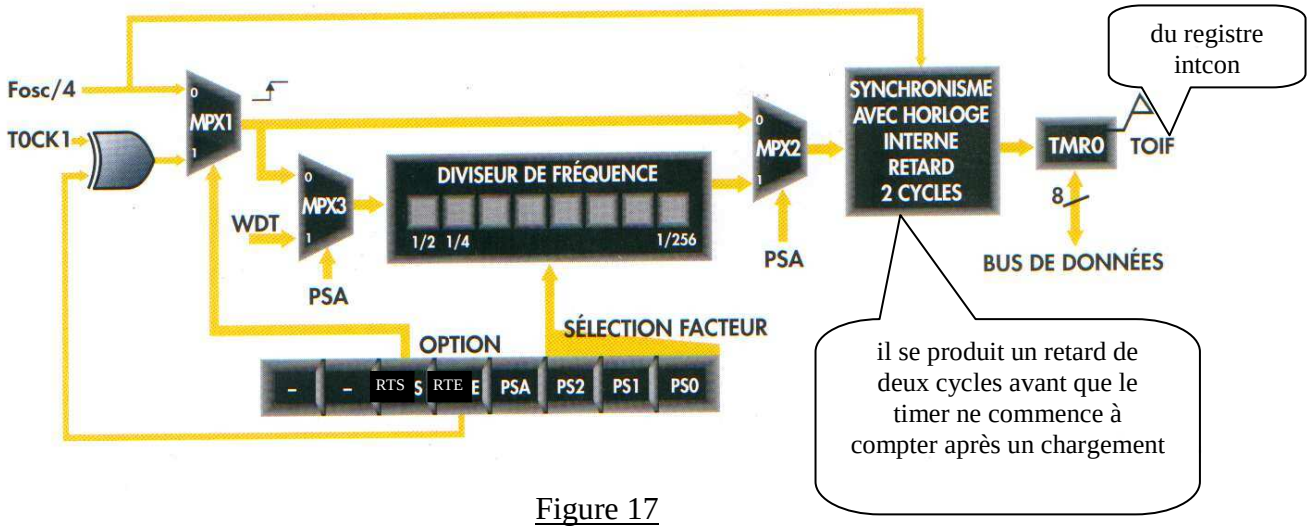


Figure 17

Suivant que le pré-diviseur est assigné au chien de garde ou au TMR0, la valeur de la pré-division n'est pas la même, il faut donc être vigilant lors de la programmation comme le montre la figure 18 :

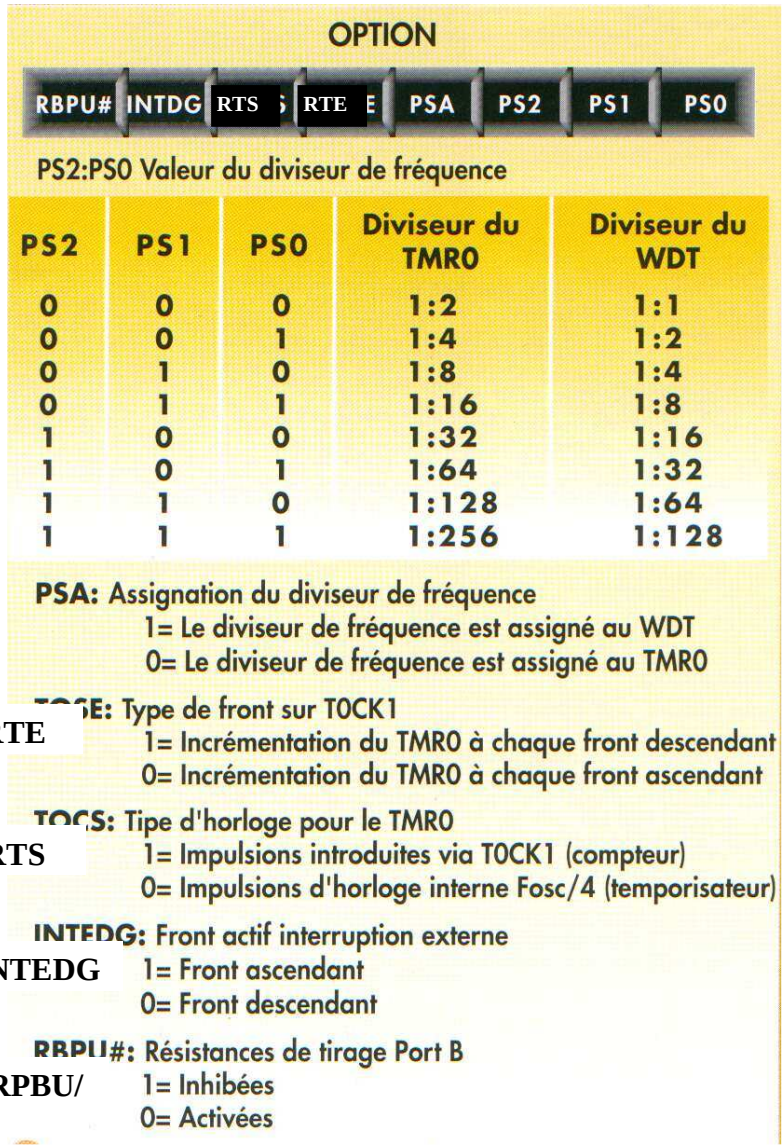


Figure 18

Enfin, vu que le timer ne peut que compter, cela oblige à une petite gymnastique lors de l'introduction de la valeur de pré chargement :

Exemple :

On veut que le timer nous indique par la mise à un du drapeau T0IF l'écoulement d'une durée de 20ms (la fréquence d'horloge étant de 4MHz) d'où **Fosc/4=1µs**

Si on choisit une pré division de 256, on aura donc **20000 µs / 256 =78**

Il ne faut pas charger le TMR0 avec 78 mais avec le complément à deux de cette valeur (car le timer compte et ne décompte pas) d'où **256-78=178** soit en hexadécimale la valeur **B2h** à charger dans le registre TMR0.

8. Mise en oeuvre

L'utilisation et la mise en œuvre très simple des PICs les ont rendus extrêmement populaire au point que la société qui les fabrique (MICROCHIP) est en passe de devenir le leader mondial dans le domaine des microcontrôleurs devant MOTOROLA et INTEL.

Il suffit d'alimenter le circuit par ses deux broches VDD et VSS, de fixer sa vitesse de fonctionnement à l'aide d'un quartz (figure19) et d'élaborer un petit système pour permettre de réinitialiser le microcontrôleur sans avoir à couper l'alimentation (figure 20).

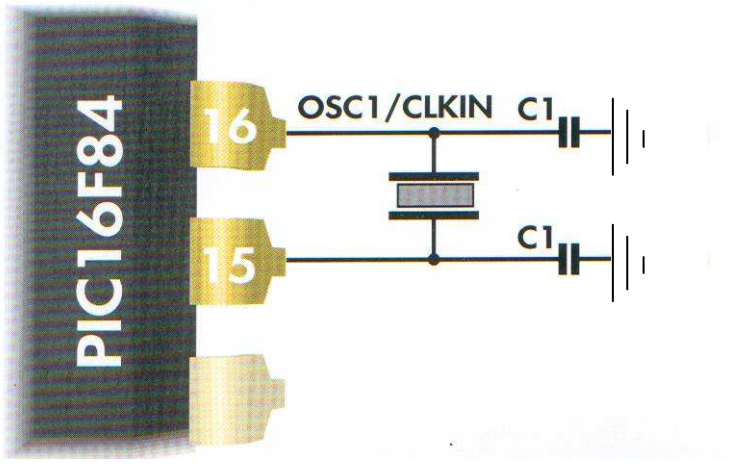


Figure 19

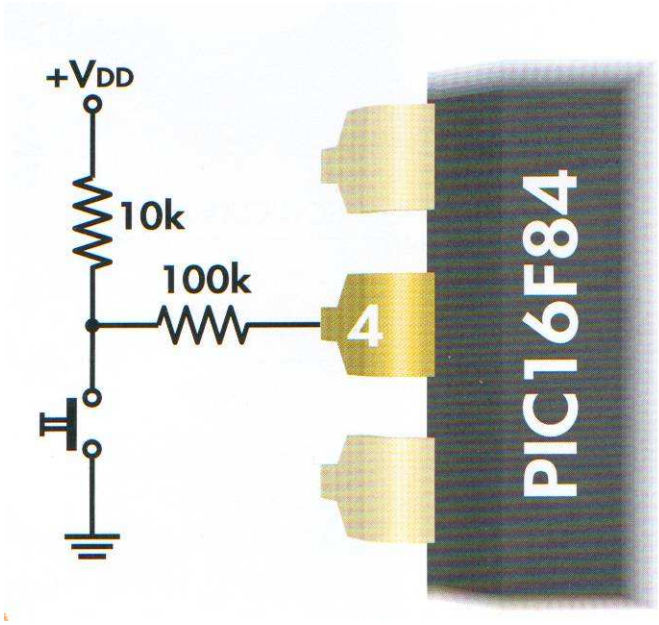


Figure 20

Il suffit ensuite d'écrire le programme en langage assembleur ou en C sur un ordinateur grâce au logiciel MPLAB de MICROCHIP (logiciel gratuit) puis de le compiler pour le transformer en langage machine et le transférer dans le PIC grâce à un programmateur. Lors de la mise sous tension, tous les registres spécifiques sont placés dans un état déterminé comme le montre la figure 21.

PAGE 0

ADR.	NOM	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	VALEUR APRÈS LE RESET	VALEUR AUTRES RESETS
00 h	INDF	Utilise le contenu de FSR pour adresser la mémoire (pas d'existence physique)								---- ----	---- ----
01 h	TMR0	Horloge/compteur en temps réel 8 bits								xxxx xxxx	uuuu uuuu
02 h	PLC	Octet de moindre poids du PC								0000 0000	0000 0000
03 h	STATUS	IRP	IRP1	RP0	TO#	PD#	Z	DC	C	0001 1xxx	000q quuu
04 h	FSR	Adressage indirect avec INDF								xxxx xxxx	uuuu uuuu
05 h	PORT A	-	-	-	RA4/T0CK	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06 h	PORT B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07 h		Non implémenté, lu comme étant à 0								---- ----	---- ----
08 h	EEDATA	Registre de données EEPROM								xxxx xxxx	uuuu uuuu
09 h	EEADR	Registre d'adresses EEPROM								xxxx xxxx	uuuu uuuu
0A h	PCLATH	-	-	-	S'écrit sur les 5 bits de PCH					---0 0000	---0 0000
0B h	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 0000

PAGE 1

ADR.	NOM	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	VALEUR APRÈS LE RESET	VALEUR AUTRES RESETS
80 h	INDF	Utilise le contenu de FSR pour adresser la mémoire (pas d'existence physique)								---- ----	---- ----
81 h	OPTION	RBPU#	INTDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82 h	PLC	Octet de moindre poids du PC								0000 0000	0000 0000
83 h	STATUS	IRP	IRP1	RP0	TO#	PD#	Z	DC	C	0001 1xxx	000q quuu
84 h	FSR	Adressage indirect avec INDF								xxxx xxxx	uuuu uuuu
85 h	TRISA	-	-	-	Configuration port A					---1 1111	---1 1111
86 h	TRISB	Configuration port B								1111 1111	1111 1111
87 h		Non implémenté, lu comme étant à 0								---- ----	---- ----
88 h	EECON1	-	-	-	EEIF	WRERR	WREN	WR	RD	---0 x000	---- q000
89 h	EECON2	Registre de commande EEPROM (pas d'existence physique)								---- ----	---- ----
0A h	PCLATH	-	-	-	Configuration port A					---0 0000	---0 0000
	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

u=unchanged ; x=unknown ; -=unimplemented (read as 0) ; q=value depends on condition

Figure 21

Il ne nous reste plus qu'à voir le jeu d'instruction de programmation en assembleur du PIC et c'est là que réside tout l'intérêt puisqu'il ne dispose que de 35 instructions qui lui permettent de réaliser toutes les tâches.

9. Jeu d'Instructions

Afin de comprendre la fonction de chaque instruction, la notation adoptée pour les données et adresses manipulées par les instructions est fort simple et est la suivante :

-**f** représente un registre
-**b** représente un numéro de bit en sachant que 0 correspond toujours au bit de poids faible (le plus à droite dans le registre)
-**k** représente une donnée aussi appelée littérale

Un certain nombre d'instructions (ADDWF, ANDWF, etc...) utilise une notation spéciale présentée sous la forme :

ADDWF f,d — Où f indique le registre et où d peut prendre deux valeurs (0 ou 1), ce qui change le comportement de l'instruction . Si d est à 0, le résultat est placé dans le registre de travail W, la valeur dans le registre f est alors inchangée, alors que si d est à 1, le résultat est placé dans le registre f.

Un autre type d'instruction mérite quelques éclaircissements, ce sont les instructions de branchement conditionnel. Prenons comme exemple :

BTFSC f,b — Qui va vouloir dire (Bit Test File Skip if Clear) qui signifie que l'on va tester le bit b du registre f (b peut prendre une valeur de 0 à 7 pour un registre 8 bits) .Il peut alors y avoir deux solutions :

-Soit le bit testé est à 1, donc la condition testée n'est pas réalisée, le programme continue alors son déroulement normalement en séquence avec l'instruction juste en dessous.

-Soit le bit testé vaut 0, donc la condition testée est réalisée et le programme saute l'instruction qui suit le BTFSC dans le programme.

Cette façon de programmer peut paraître étrange, mais avec de l'habitude, elle s'avère très pratique et permet de réaliser des programmes compacts et performants.

Les 35 instructions sont donc les suivantes :

9.1 Instructions arithmétiques:

9.1.1 ADDLW (Add Literal to W)

syntaxe : ADDLW k

Opération : W+k→W

Bits d'état du registre STATUS affectés : C,DC,Z

on ajoute au registre de travail la valeur k et on place le résultat dans le registre de travail W

durée : 1 cycle instruction (4 cycles d'horloge)

9.1.2 ADDWF (Add W to F)

syntaxe : ADDWF f,d

Opération : W+f→f si d=1 ou W+f→W si d=0

Bits d'état du registre STATUS affectés : C,DC,Z

on ajoute le contenu de W et le contenu de f et on place le résultat dans f si d=1 ou dans W si d=0

durée : 1 cycle instruction (4 cycles d'horloge)

9.1.3 SUBLW (Substract W from Literal)

syntaxe : SUBLW k

opération : k-W→W

Bits d'état du registre STATUS affectés :C,DC,Z

On soustrait le contenu du registre W du littéral k et on place le résultat dans W (soustraction par la méthode du complément à 2).

durée : 1 cycle instruction (4 cycles d'horloge)

9.1.4 SUBWF (Substract W from F)

syntaxe : SUBWF f,d

opération : f-W→W si d=0 ou f-W→f si d=1

Bits d'état du registre STATUS affectés :C,DC,Z

On soustrait le contenu du registre W du contenu du registre f et on place le résultat dans W si d=0, ou dans f si d=1 (soustraction par la méthode du complément à 2).

durée : 1 cycle instruction (4 cycles d'horloge)

9.2 Instructions logiques:

9.2.1 ANDLW (And Literal and W)

syntaxe : ANDLW k

Opération : W ET k→W

Bit d'état du registre STATUS affecté : Z

on effectue un ET logique entre le contenu de W et le littéral k , on place le résultat dans W

durée : 1 cycle instruction (4 cycles d'horloge)

9.2.2 ANDWF (And W with F)

syntaxe : ANDWF f,d

Opération : W ET f→f si d=1 ou W ET f→W si d=0

Bit d'état du registre STATUS affecté :Z

on effectue un ET logique entre le contenu de W et le contenu de f , on place le résultat dans W si d=0 ou dans f si d=1

durée : 1 cycle instruction (4 cycles d'horloge)

9.2.3 COMF (Complement F)

syntaxe : COMF f,d

opération : /f→f si d=1 ou /f→W si d=0

Bit d'état du registre STATUS affecté :Z

On complémente le contenu du registre f bit à bit, le résultat est placé dans f si d=1 , dans W si d=0 .

durée : 1 cycle instruction (4 cycles d'horloge)

9.2.4 IORLW (Inclusive Or Literal with W)

syntaxe: IORLW k

opération: W OU k → W

Bit d'état du registre STATUS affecté :Z

On effectue un OU logique entre le contenu de W et le littéral k, le résultat est placé dans W.

durée : 1 cycle instruction (4 cycles d'horloge)

9.2.5 IORWF (Inclusive Or W with F)

syntaxe : IORWF f,d

opération : W OU f→f si d=1 ou W OU f→W si d=0

Bit d'état du registre STATUS affecté :Z

On effectue un OU entre le contenu de W et le contenu de f , on place le résultat dans f si d=1, dans W si d=0

durée : 1 cycle instruction (4 cycles d'horloge)

9.2.6 XORLW (Exclusive Or Literal with W)

syntaxe : XORLW k

opération : W OU EXCLUSIF k→W

Bit d'état du registre STATUS affecté :Z

On effectue un OU Exclusif entre W et le littéral k, le résultat est placé dans W

durée : 1 cycle instruction (4 cycles d'horloge)

9.2.7 XORWF (Exclusive Or W with F)

syntaxe : XORWF f,d

opération : W OU EXCLUSIF f→W si d=0 ou W OU EXCLUSIF f→f si d=1

Bit d'état du registre STATUS affecté :Z

On effectue un OU Exclusif entre W et le contenu de f, le résultat est placé dans W si d=0, sinon il est placé dans f.

durée : 1 cycle instruction (4 cycles d'horloge)

9.3 Instructions d'incrémentation :

9.3.1 DECF (Decrement F)

syntaxe : DECF f,d

opération : f-1→f si d=1 ou f-1→W si d=0

Bit d'état du registre STATUS affecté :Z

On diminue le contenu du registre f d'une unité, le résultat est placé dans f si d=1, dans W si d=0 (dans ce cas f reste inchangé).

durée : 1 cycle instruction (4 cycles d'horloge)

9.3.2 DECFSZ (Decrement F ,Skip if Zero)

syntaxe : DECFSZ f,d

opération : f-1→f si d=1 ou f-1→W si d=0 et saut si f-1=0

Bit d'état du registre STATUS affecté : aucun

On diminue le contenu du registre f d'une unité, le résultat est placé dans f si d=1, dans W si d=0 (dans ce cas f reste inchangé).Si le résultat est nul, l'instruction suivante est ignorée et dans ce cas, cette instruction dure deux cycles.

durée : 1 cycle instruction (4 cycles d'horloge) ou 2 cycles

9.3.3 INCF (Increment F)

syntaxe : INCF f,d

opération : f+1→f si d=1 ou f+1→W si d=0

Bit d'état du registre STATUS affecté :Z

On augment le contenu du registre f d'une unité, le résultat est placé dans f si d=1 , dans W si d=0 (dans ce cas f reste inchangé).

durée : 1 cycle instruction (4 cycles d'horloge)

9.3.4 INCFSZ (Increment F , Skip if Zero)

syntaxe : INCFSZ f,d

opération : f+1→f si d=1 ou f+1→W si d=0 et saut si f-1=0

Bit d'état du registre STATUS affecté : aucun

On augmente le contenu du registre f d'une unité, le résultat est placé dans f si d=1, dans W si d=0 (dans ce cas f reste inchangé).Si le résultat est nul, l'instruction suivante est ignorée et dans ce cas, cette instruction dure deux cycles.

durée : 1 cycle instruction (4 cycles d'horloge) ou 2 cycles

9.4 Instructions d'effacement :

9.4.1 CLRF (Clear F)

syntaxe : CLRF f

Opération : 0→F

Bit d'état du registre STATUS affecté : Z

On met le contenu du registre f à 0 et on positionne Z

durée : 1 cycle instruction (4 cycles d'horloge)

9.4.2 CLRW (Clear W)

syntaxe : CLRW

Opération : 0→W

Bit d'état du registre STATUS affecté : Z

On met le contenu du registre W à 0 et on positionne Z

durée : 1 cycle instruction (4 cycles d'horloge)

9.4.3 CLRWDT (Clear WatchDog Timer)

syntaxe : CLRWDT

Opération : 0→WDT et 0→pré diviseur du Timer

On met le contenu du registre du timer chien de garde à 0 ainsi que le pré diviseur

durée : 1 cycle instruction (4 cycles d'horloge)

9.5 Instructions de mouvement :

9.5.1 MOVF (Move F)

syntaxe : MOVF f,d

opération : f→f si d=1 ou f→W si d=0

Bit d'état du registre STATUS affecté :Z

On déplace le contenu de f dans f si d=1 ou de f dans W si d=0. Attention, le déplacement de f dans f semble à priori inutile, mais il permet en fait de tester le contenu de f par rapport à 0 et de positionner le bit Z

durée : 1 cycle instruction (4 cycles d'horloge)

9.5.2 MOVLW (Move Literal to W)

syntaxe : MOVLW k

opération : k→W

Bit d'état du registre STATUS affecté :aucun

On charge le contenu de W avec le littéral k

durée : 1 cycle instruction (4 cycles d'horloge)

9.5.3 MOVWF (Move W to F)

syntaxe : MOVWF f

opération : W→f

Bit d'état du registre STATUS affecté :aucun

On charge le contenu de f avec le contenu de W

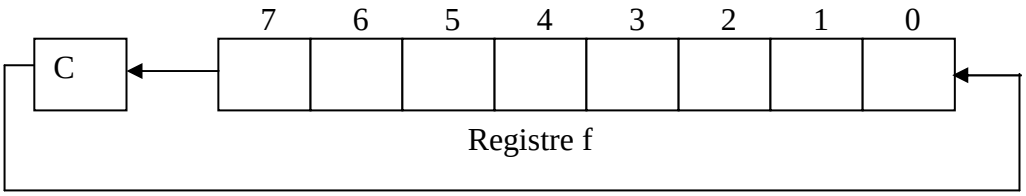
durée : 1 cycle instruction (4 cycles d'horloge)

9.6 Instructions de rotation:

9.6.1 RLF (Rotate Left F through carry)

syntaxe : RLF f,d

opération :



Bit d'état du registre STATUS affecté :C

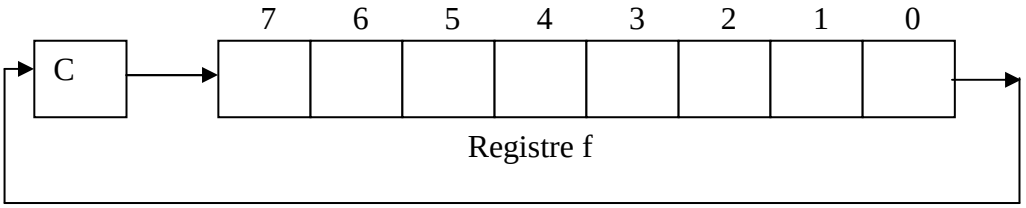
On effectue une rotation à gauche de un bit du contenu du registre f en passant par le bit de retenu C . Si d=1 le résultat est placé dans f , si d=0 , le résultat est placé dans W

durée : 1 cycle instruction (4 cycles d'horloge)

9.6.2 RRF (Rotate Right F through carry)

syntaxe : RRF f,d

opération :



Bit d'état du registre STATUS affecté : C

On effectue une rotation à droite de un bit du contenu du registre f en passant par le bit de retenu C. Si d=1 le résultat est placé dans f , si d=0, le résultat est placé dans W

durée : 1 cycle instruction (4 cycles d'horloge)

9.7 Instructions de saut et branchement :

9.7.1 CALL (subroutine Call)

syntaxe : CALL label

Bits d'état du registre STATUS affectés : aucuns

On sauvegarde l'adresse de retour dans la pile puis on appelle le sous programme définit avec l'étiquette label

durée : 2 cycles instruction (8 cycles d'horloge)

9.7.2 GOTO (branchement inconditionnel)

syntaxe : GOTO label

Bit d'état du registre STATUS affecté : aucun

On effectue un saut dans le programme pour aller à l'adresse pointé par le label précisé dans GOTO

Durée : 2 cycles (8 cycles d'horloge)

9.7.3 RETFIE (Return From Interrupt)

syntaxe : RETFIE

opération : Pile→ PC

Bit d'état du registre STATUS affecté : aucun

On charge le compteur ordinal avec la valeur qui se trouve au sommet de la pile pour revenir au programme principal lorsque l'exécution de sous programme est terminée.

durée : 2 cycles instruction (8 cycles d'horloge)

9.7.4 RETLW (Return Literal to W)

syntaxe : RETLW k

opération : k→W , Pile→PC

Bit d'état du registre STATUS affecté : aucun

On charge le contenu de W avec le littéral k puis on charge le compteur ordinal PC avec la valeur qui se trouve au sommet de la pile effectuent ainsi un retour de sous programme.

durée : 2 cycles instruction (8 cycles d'horloge)

9.7.5 RETURN (Return from subroutine)

syntaxe : RETURN

opération : Pile→PC

Bit d'état du registre STATUS affecté : aucun

On charge le compteur ordinal PC avec la valeur qui se trouve au sommet de la pile effectuent ainsi un retour de sous programme. C'est un RETLW simplifié.

durée : 2 cycles instruction (8 cycles d'horloge)

9.8 Instructions agissant sur les bits :

9.8.1 BCF (Bit Clear F)

syntaxe : BCF f,b

Opération : 0→b(f)

Bits d'état du registre STATUS affectés : aucuns

on met à 0 le bit b du registre f

durée : 1 cycle instruction (4 cycles d'horloge)

9.8.2 BSF (Bit Set F)

syntaxe : BSF f,b

Opération : 1→b(f)

Bits d'état du registre STATUS affectés : aucuns

on met à 1 le bit b du registre f

durée : 1 cycle instruction (4 cycles d'horloge)

9.8.3 BTFSC (Bit Test , Skip if Clear)

syntaxe : BTFSC f,b

Opération : saut de l'instruction qui suit si b(f)=0

Bits d'état du registre STATUS affectés : aucuns

Si le bit b de f est nul, l'instruction qui suit celle-ci est ignorée et traitée comme un NOP. Dans ce cas et dans ce cas seulement, l'instruction BTFSC demande deux cycles pour s'exécuter.

durée : 1 cycle instruction (4 cycles d'horloge) ou 2 cycles

9.8.4 BTFSS (Bit Test , Skip if Set)

syntaxe : BTFSS f,b

Opération : saut de l'instruction qui suit si b(f)=1

Bits d'état du registre STATUS affectés : aucuns

Si le bit b de f est à 1, l'instruction qui suit celle-ci est ignorée et traitée comme un NOP. Dans ce cas et dans ce cas seulement, l'instruction BTFSS demande deux cycles pour s'exécuter.

durée : 1 cycle instruction (4 cycles d'horloge) ou 2 cycles

9.9 Instructions diverses :

9.9.1 NOP (No Operation)

syntaxe: NOP

opération: néant

Bit d'état du registre STATUS affecté : aucun

On ne fait que consommer du temps machine (un cycle dans ce cas)

durée : 1 cycle instruction (4 cycles d'horloge)

9.9.2 SLEEP (Sleep)

syntaxe : SLEEP

opération : 0→PD, 1→T0, 0→WDT , 0→ pré diviseur

On place le circuit en mode sommeil avec arrêt de l'oscillateur. Cette commande est à utiliser avec précaution, elle nécessite la connaissance du mode sommeil.

9.9.3 SWAPF (Swap F)

syntaxe : SWAPF f,d

opération : f(0-3)→f(4-7) et f(4-7)→f(0-3) résultat dans W ou f selon d

Bit d'état du registre STATUS affecté : aucun

On échange les quatre bits de poids forts avec les quatre bits de poids faibles et on place le résultat dans W si d=0, ou dans f si d=1

durée : 1 cycle instruction (4 cycles d'horloge)

Chapitre 3 : Les Servomoteurs

1. Introduction :

Les servomoteurs sont très utilisés en robotique car ils sont à la base de nombreux mécanisme d'actionneurs, comme des pinces par exemple. Par contre, ils consomment souvent beaucoup de courant et demandent une tension souvent très bien régulée ce qui nécessite des dispositions particulières.

2. Qu'est-ce qu'un servomoteur ?

Un servomoteur est un dispositif typiquement utilisé en modélisme, par exemple pour diriger une voiture télécommandée ou contrôler à distance une caméra vidéo.

Figure22 : Servomoteur

Il comprend :

• Un moteur électrique de petite taille.

• Un réducteur en sortie de ce moteur pour avoir moins de vitesse et plus de couple.

• Un capteur : un potentiomètre qui produit une tension variable en fonction de l'angle de l'axe de sortie.

• Un asservissement électronique pour contrôler la position de cet axe de sortie.

3. Fonctionnement d'un servomoteur :

L'utilisateur envoie à chaque instant un signal modulé représentatif d'un angle. Le servomoteur tourne vers cet angle et s'y maintient.

La duré de l'impulsion se varie entre 0.9ms qui permet à l'axe de tourner jusqu'à la position extrême gauche et 2.1ms qui lui permet d'aller jusqu'à l'extrême droite. Et avec 1.5ms pour revenir au centre.

Pour varier la vitesse d'un servomoteur en jouant sur la durée de l'impulsion. Plus la durée de l'impulsion est proche de 1.5ms plus la vitesse de rotation est faible.

4. Le potentiomètre :

Le potentiomètre sert à l'asservissement de position. L'axe du potentiomètre est solidaire de l'axe de sortie qui lui permet de tourner en même temps que l'axe de sortie. La résistance à la borne du potentiomètre varie en fonction de la position de l'axe. La résistance du potentiomètre est en rapport avec la durée du signal. Par exemple le servomoteur reçoit un ordre pour se déplacer à la position médiane (1.5ms). Le moteur va tourner jusqu'à ce que la résistance entre le curseur du potentiomètre et les deux extrémités soient identiques et s'il reçoit une impulsion de 2.1ms il va tourner jusqu'à ce que la résistance entre le curseur et l'extrémité du potentiomètre soit maximale.

L'électronique faire la relation entre la durée de l'impulsion et la résistance aux bornes du potentiomètre.

5. La commande du servomoteur

Un connecteur à trois fils permet de commander un servo-moteur. Bien sûr, il y a la masse et le +5V (à peu près). Ces deux fils sont en général noir et rouge, mais ils sont marrons et rouge pour les modèles Graupner. Le 3ème fil est le fil de commande. Il est normalement blanc, mais jaune dans le cas des Graupner.

On pourrait dire sommairement qu'il faut fournir un signal **PWM** en tant que signal de commande, mais ce n'est pas tout à fait exact :

• Le PWM (**Pulse Width Modulation** : Modulation en largeur d'impulsion) est un signal dont le taux de modulation varie de 0 à 100% $T_M = \frac{T_{ON}}{T_{PERIODE}}$.

• Pour un servomoteur on parle de **Pulse-Code Modulated Signal** (Signal modulé en code d'impulsion) où seule l'impulsion compte, pas la fréquence (en théorie).

En clair, il faut fournir au servomoteur une impulsion à 1 (suivie d'un retour à 0). Le servomoteur va prendre en compte la largeur temporelle de cette impulsion qu'il va convertir de façon linéaire en un angle.

La durée de retour à 0 de l'impulsion n'est pas critique, en pratique il faut éviter de dépasser 20ms entre deux fronts montants.

A noter aussi que ce système présente un avantage par rapport au PWM : une absence de signal (toujours à 1, ou à 0) laisse le servomoteur en "roue libre" comme si il n'était pas alimenté. Le microcontrôleur programmé est capable de fournir directement ce signal de commande sur une sortie.

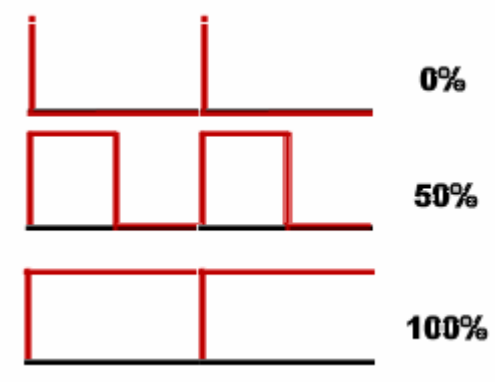


Figure23 : Signal PWM

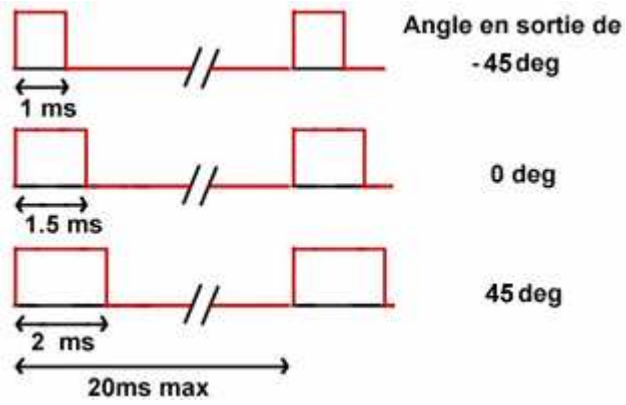


Figure24 : Signal modulé en code d'impulsion

6. Caractéristique du servomoteur utilisé dans ce projet :

- Référence : le HS322 de HITEC
- Tension d'alimentation = 5 à 6V
- Dimension : 41×20×37 (mm)
- Poids : 43gr
- Couple : 2.8 à 3.5 Kg/cm
- Vitesse : 0.19s/60°
- Pignons : nylon

Figure25 : Dimension du servomoteur

Le servomoteur nécessite une modification pour compléter la rotation.

Chapitre 4 : Les détecteurs

1. L'électronique des Capteurs :

Il s'agit d'optocoupleur de référence OBP704 (Constructeur OPTEK) qui émet un rayon focalisé et récupère la réflexion de ce rayon par une photo transistor que nous utiliserons pour repérer la position de la bande de guidage.

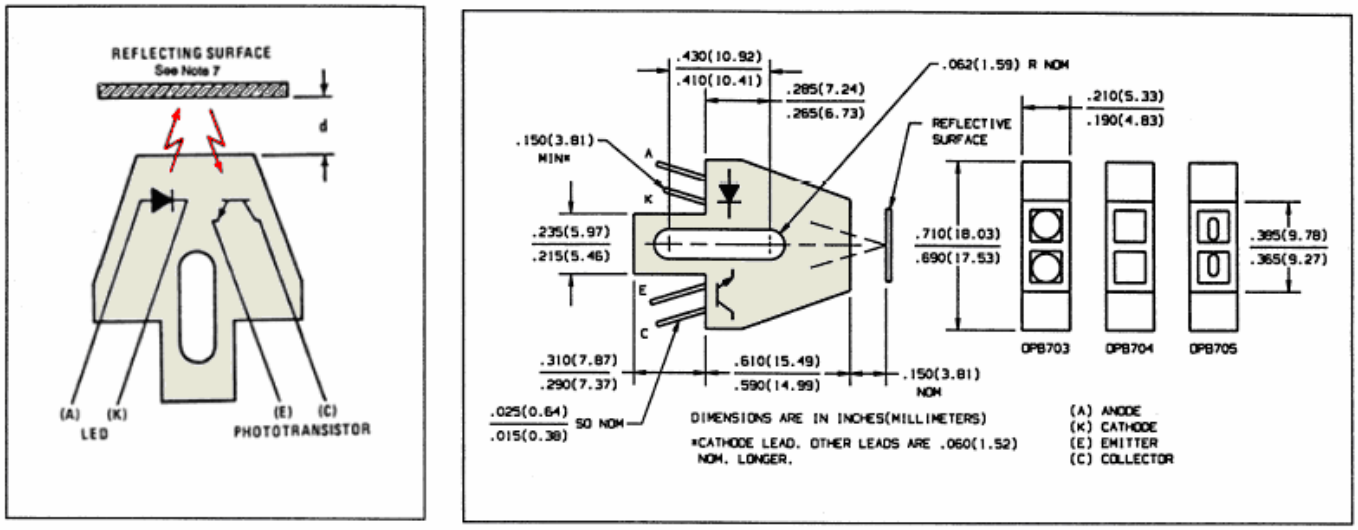


Figure26 : Description du capteur de suivi de la ligne de guidage

La mise en forme est confiée à un circuit TTL 74HC14 (Une porte Nand trigger de Schmitt) et permet de disposer de 6 signaux TTL actifs au niveau 1, nommés capteur1... capteur6 présents sur le bus de connexion du robot. Le capteur 1 désigne le capteur extrême avant droit et le capteur 6 désigne le capteur avant extrême gauche. Référence capteurs : OPB704

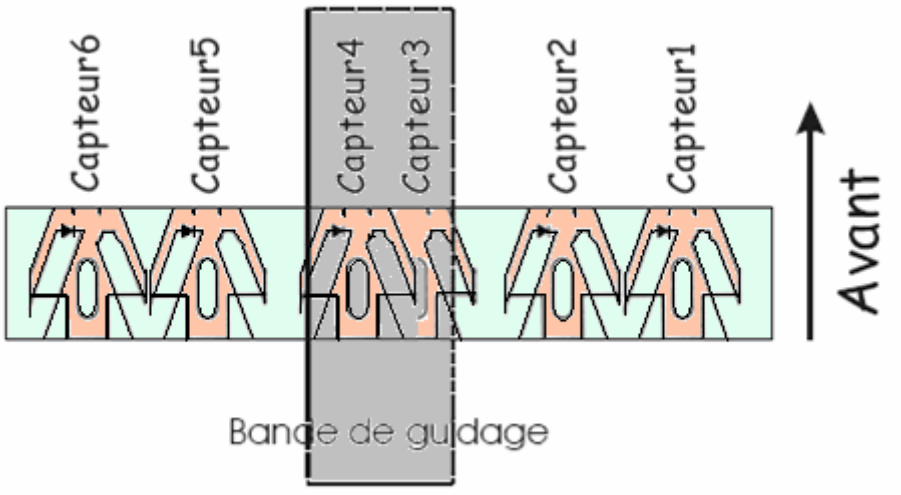


Figure 27 : Montage des capteurs de position à l'avant du robot

Le montage des capteurs de position permet de détecter la bande de guidage de manière exclusive pour les capteurs 1, 2, 5 et 6 (un seul capteur actif à la fois) et non exclusive pour les capteurs centraux (3 et 4). Cette disposition permettra d'optimiser la vitesse du robot lorsque celui-ci sera parfaitement centré par rapport à la bande de guidage.

2. Description et fonctionnement :

Le principe est de l'émission d'une onde et de sa réception. Donc deux cas peuvent se produire : soit le capteur détecte la piste, soit il ne la détecte pas.

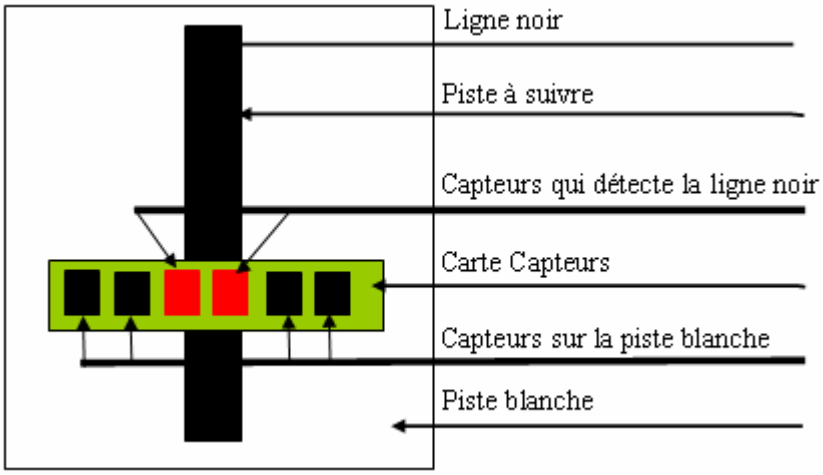


Figure 28

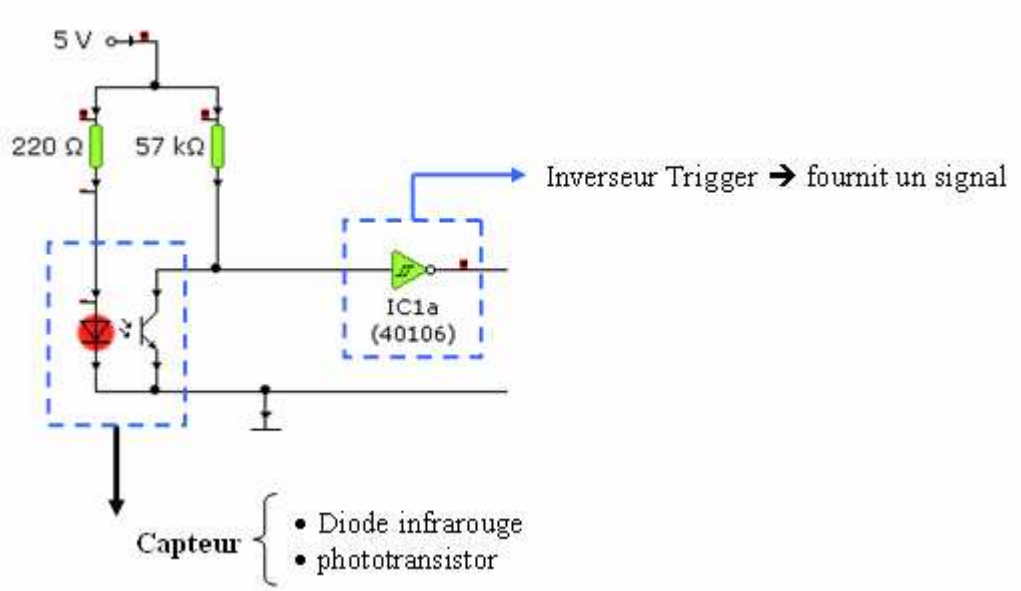
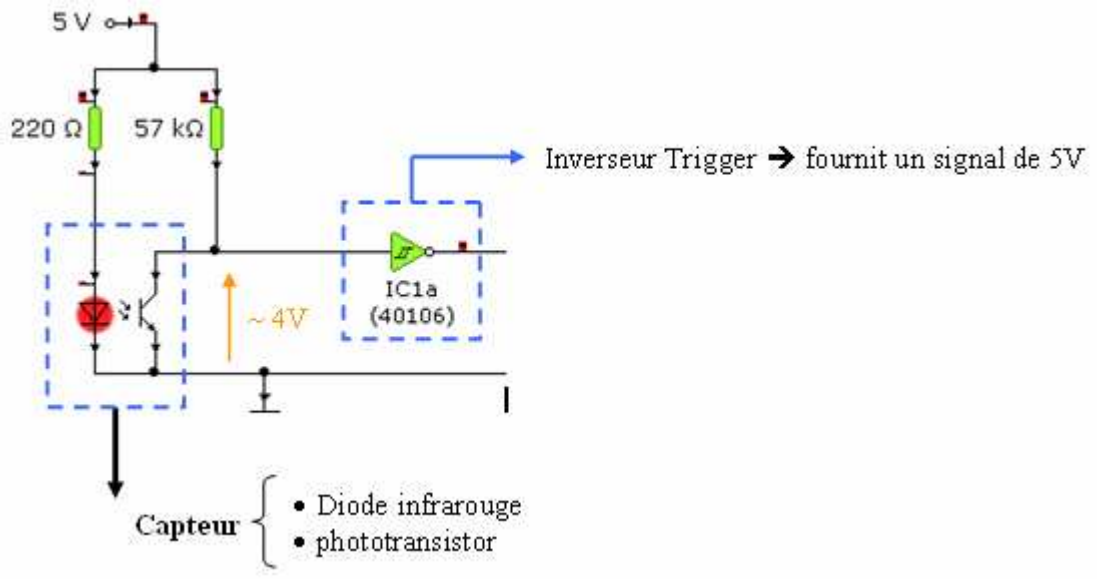
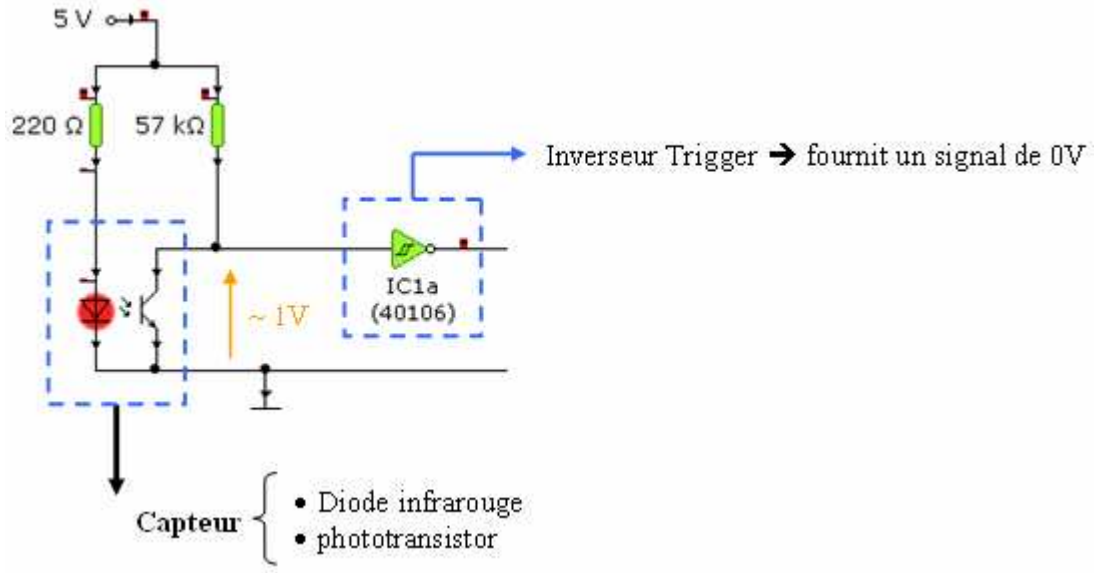


Figure 28 : Montage de capteur avec inverseur **trigger de Schmitt**

Cas de non détection de la ligne :



Cas de détection de la ligne :



PARTIE II

ETUDE PRATIQUE

Chapitre 1 : Modification des servomoteurs

Le servomoteur, pour des applications en modélisme, n'effectue qu'une rotation de ±150°, ce qui nécessite une modification de telle sorte que son axe de sortie effectue une rotation complète de 360°.

1. Principe de la modification

Le principe c'est qu'on va remplacer le potentiomètre par 2 résistances fixes de même valeur. Ceci aura pour effet de faire croire à l'électronique du servomoteur que l'axe de sortie se trouve toujours en positon médiane. Si on lui envoie une impulsion de durée supérieure à 1.5ms, le servomoteur va tourner vers la droite en essayant d'atteindre la valeur de résistance correspondante. Comme les résistances sont fixes il va tourner indéfiniment à droite. Idem pour la gauche. Par contre si on lui envoie une impulsion de 1.5ms le moteur va s'arrêter puisque la valeur des résistances correspondantes à la valeur médiane.

2. En résumé pour

- tourner en avant : Impulsion > 1.5ms
- en arrière : < 1.5ms
- arrêt : 1.5ms

Le servomoteur que nous allons modifier est le HS-300BB de Hitec.

Figure 29 : servomoteur

3. Il y a 3 modifications à apporter au servomoteur d'origine

- modifier le potentiomètre pour une rotation complète
- rajouter 2 résistances sur le circuit imprimé
- couper l'ergot sur le pignon de sortie

4. Démonter le servomoteur

A l'aide d'un petit tournevis cruciforme dévissez les 4 vis du boîtier.

Vous obtenez ceci

Figure 30

Extraire le circuit imprimé

Figure 31

Déposez les roues dentées et le roulement à bille en repérant bien la position de montage.

Figure 32

5. Modification du potentiomètre

On va garder juste la fonction mécanique du potentiomètre comme l'axe de ce dernier est le support du pignon de sortie. Il faut que le potentiomètre puisse effectuer un tour complet. En ce qui concerne la fonction électrique on va remplacer le potentiomètre par deux résistances fixes. Dévissez l'écrou de fixation du potentiomètre et l'extraire du boîtier.

Figure 33

Ouvrir le potentiomètre en dépliant les 3 languettes à l'aide d'un petit tournevis.

Figure34

Modifier le potentiomètre afin que son axe puisse faire une rotation complète. A l'aide d'une petite pince coupante couper la butée du capot et la plier vers l'extérieur. Couper le contact à pression au ras du disque afin que l'axe du potentiomètre puisse effectuer un tour complet sans frottement.

Coupez également les 2 contacts à pression du couvercle, ceci diminue les frottements.

Figure 35

Refermez le potentiomètre en resserrant les languettes avec une pince. Dessoudez les fils rouge, jaune et bleu du potentiomètre et du circuit imprimé. Utilisez un petit fer à souder surtout pour le circuit imprimé. Pour nettoyer les trous pastillées, utilisez de la tresse à dessouder.

Remettre le potentiomètre en place.

A effectuer uniquement sur un des 2 servomoteurs. Dessoudez les 2 fils du moteur CC et les croiser. Ceci afin que les 2 servomoteurs tournent dans le même sens pour une même impulsion.

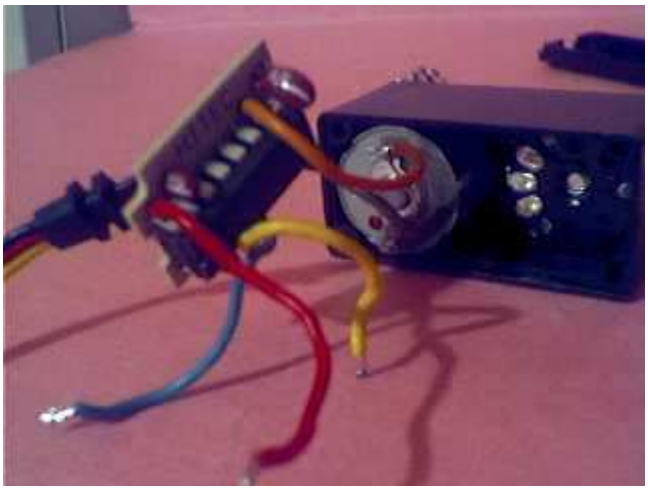


Figure 36

6. Mise en place des résistances

Prenez 4 résistances de 1,5 KΩ (2 pour chaque servomoteur). Attention elles doivent être rigoureusement égales. Si vous avez un lot de résistances 5%, mesurez-les à l'ohmmètre pour en trouver 2 identiques. Sinon, prenez des résistances 1%.
Soudez les comme indiqué sur la photo au plus près du circuit imprimé. Veillez à ne pas faire de court-circuit entre les pistes en soudant.

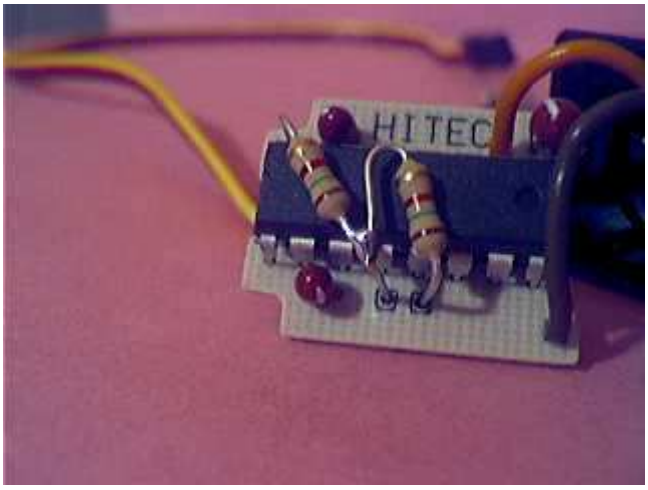


Figure 37

7. Modification du pignon de sortie

Couper l'ergot du pignon de sortie.

AVANT

Figure 38

APRES

Figure 39

Remettre les pignons en place. Vérifiez que le pignon de sortie puisse effectuer un tour complet.

Remettre le circuit imprimé en place.

Refermez le boîtier.

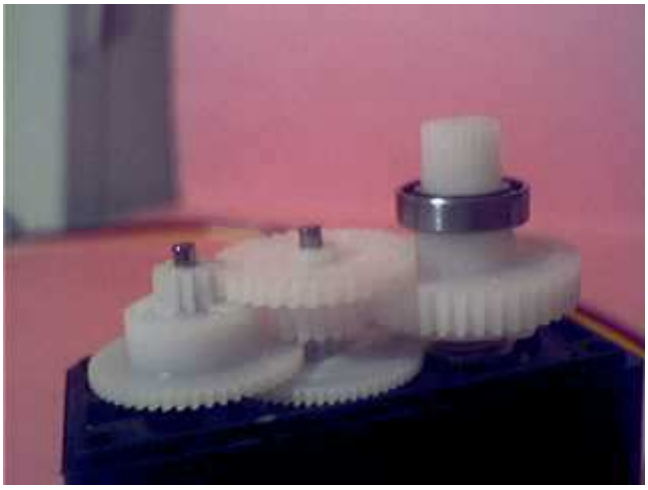


Figure 40 : La modification est terminée

Chapitre 2 : Réalisation des Circuits

1. Réalisation d'un circuit imprimé

1.1 Matériel nécessaire

- Plaque d'époxy pré-sensibilisé simple face
- La scie à métaux pour découper l'époxy
- Un sachet de révélateur
- Du perchlorure de fer liquide ou en granulés à diluer (lire la doc fournie avec pour la préparation)
- Une insoleuse (par défaut une simple ampoule de 100 Watts avec un plaque de verre peu convenir)
- Une mini-perseuse avec des forets 0.8, 1, 1.5 et 2mm.

1.2 Le typon

Un typon est une image du circuit imprimé. Les pistes sont en noir et le reste doit être transparent à la lumière.

Notre méthode est d'imprimer le typon sur une feuille blanche et verser l'huile de cuisine sur les deux faces pour la rendre transparente à la lumière.

1.3 Insoleuse / insolation

Une insoleuse est un appareil fort simple, doté de très peu de composants, que l'on peut parfaitement fabriquer soi-même. Ce type d'appareil est doté de 1 à 4 tubes rayonnant des UV, et est destiné à insoler des plaques de circuit imprimé présensibilisées (sous-entendus sensibles aux UV). Il est aussi possible d'utiliser une ampoule spéciale, mais les temps d'insolation sont bien plus longs et la régularité de l'insolation n'est pas excellente, surtout pour les grands circuits.

Figure 41

Figure42 : plaque à Epoxy

Cette plaque est recouverte d'une résine sensible aux Ultraviolets. La résine est protégée de la lumière par une pellicule protectrice. L'opération consiste à mettre l'image du typon sur la plaque époxy en l'exposant aux UV.
Le temps d'insolation est très important qui prend un peu près 4min.

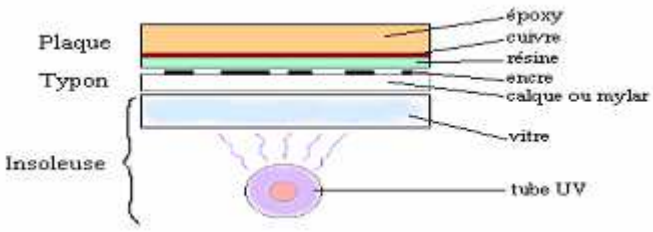


Figure 43 : Insolation

Utilisation d'une insoleuse :

- Découper la plaque d'époxy à la taille du typon en laissant une marge de 0.5 à 1cm
- Bien placer le typon sur la vitre de l'insoleuse (Vue cotée composantes)
- Retirer la pellicule protectrice de la plaque
- Placer le coté vert de la plaque sur le typon dans l'insoleuse et fermez le capot.
- Régler la minuterie sur la valeur 4.

Utilisation d'une ampoule :

- Découper la plaque d'époxy à la taille du typon en laissant une marge de 0.5 à 1cm
- Placer la plaque sur un endroit plat, le coté vert vers le haut
- Mettre le typon au dessus de la plaque
- Mettre une vitre en verre au dessus pour bien aplatir le typon sur la plaque d'époxy
- Fixer une ampoule de 100 W à 250 W environ 30 cm au dessus
- Insoler 15 à 30 minute (Il est préférable de faire des tests sur des petites plaque avant de lancer l'insolation des vraies cartes)

Après l'insolation vient la révélation.

1.4 Révélation

Le révélateur doit être mis dans une bassine en plastique (PVC) ou en verre, le temps de révéler le circuit imprimé, puis devra être remis dans son récipient de stockage juste après usage. La température optimale d'utilisation de ce produit est généralement de 20 à 24°C. Le temps de révélation peut varier de quelques secondes à quelques dizaines de seconde, mais cela reste rapide dans tous les cas, et est parfaitement visible. Pendant qu'on remue (délicatement) la bassine, On voie apparaître petit à petit le cuivre nu (couleur rose) aux endroits exposés aux UV, c'est à dire aux endroits qui ne devront pas subsister à l'étape de gravure finale.

1.5 La gravure

La gravure va permettre au cuivre non protégé par la couche de résine (issu de la révélation ou des méthodes 1 et 2) de se dissoudre. Plonger le circuit dans un bain de perchlorure de fer à 50 ° pendant le temps nécessaire à la dissolution en agitant fortement. Attention, le perchlorure de fer est un acide toxique et corrosif pour la peau mais aussi pour la tuyauterie. Ne pas laisser

trop longtemps le circuit dans le bain sinon tout sera dissous. Bien rincer après gravure. Le nettoyage du circuit se fait avec de l'acétone ou du dissolvant à ongle.

1.6 Perçage

Une simple perceuse alimentée en 12 V.

Figure 44

Paramètre des perçages :

- Résistances, Diodes, Circuits intégrés, Transistors, Capacités... ➔ 0.8mm
- Fils ➔ 1.5mm
- Autres composantes ➔ 1.2 ou 1.5mm

1.7 Soudage

La technique du soudage nécessite une technique suivante : préchauffer la pastille et la patte avec le fer pendant un bref délai, et seulement ensuite appliquer le fil d'étain. On doit avoir l'étain fondre et se positionner tout seul autour de la patte. Retirer le fil puis le fer.

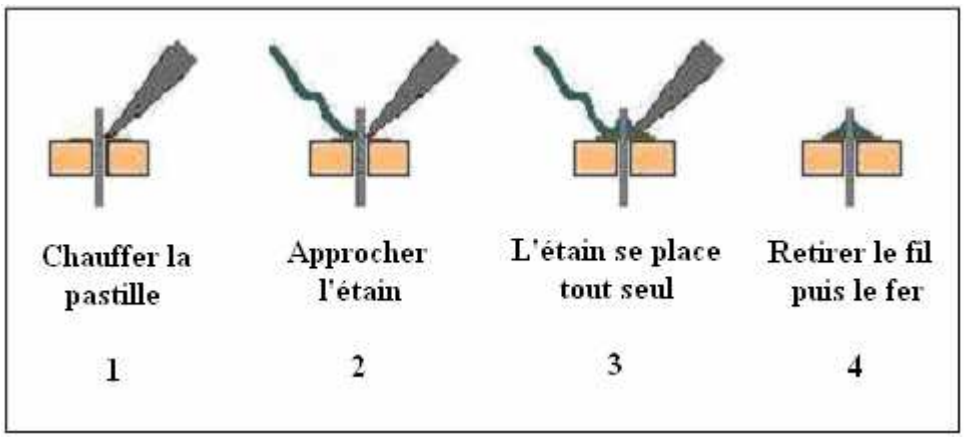


Figure 45 : Soudage

2. La carte capteurs de détection

2.1Mode de fonctionnement de capteur

La diode est sous tension émet toujours un signal.

- Cas de présence de ligne noire on n'aura pas de réflexion, alors le phototransistor sera bloqué ➔ la sortie **S = 0**.

- Cas d'absence de ligne noire on aura de réflexion, alors le phototransistor sera saturé ➔ la sortie **S = 1**.

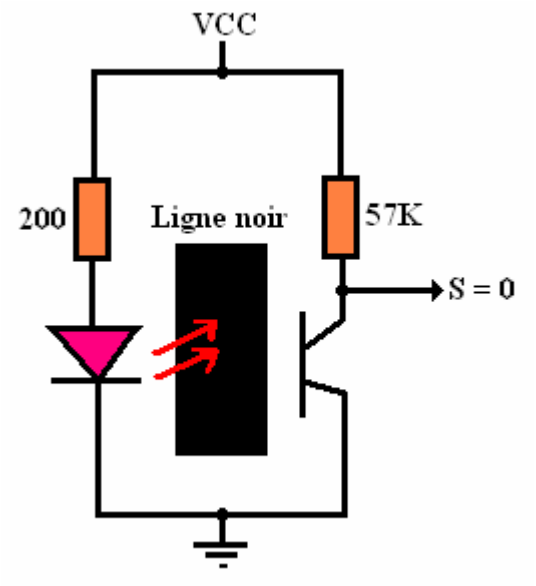


Figure 46 : Cas de présence de ligne

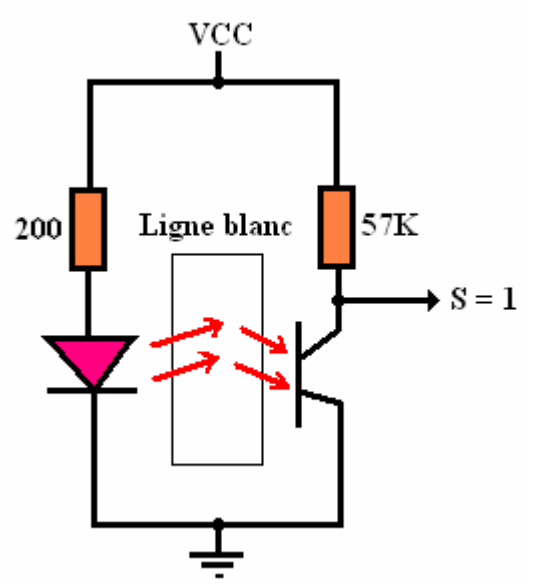


Figure 47 : Cas d'absence de ligne

2.2 capteurs de détection de ligne

Le schéma ci-dessous est pour les six détecteurs de ligne noire qui disposent une sortie pour chaqu'un vers l'es entrées de pic de RB2 à RB7.

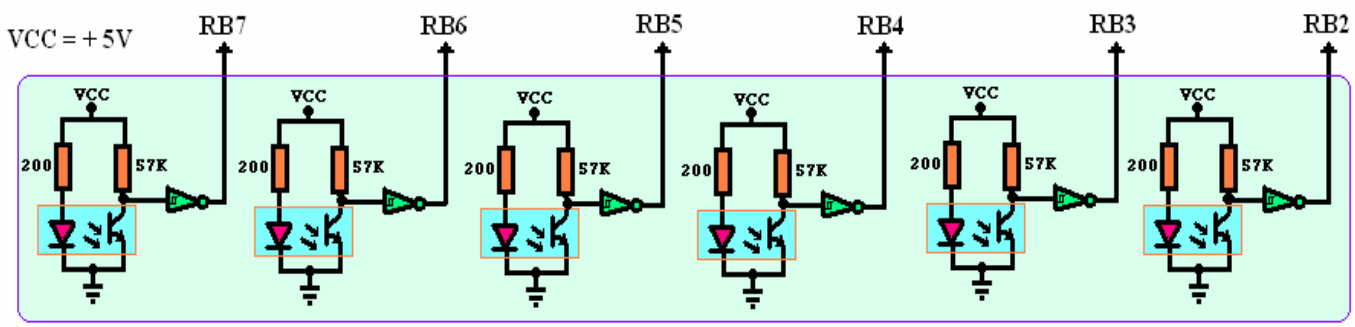


Figure 48 : Capteurs de la piste

Ces capteurs résultent différentes combinaisons possibles (Voir les figures 49 et 50).

2.2.1 Cas de ligne d'épaisseur = 10mm

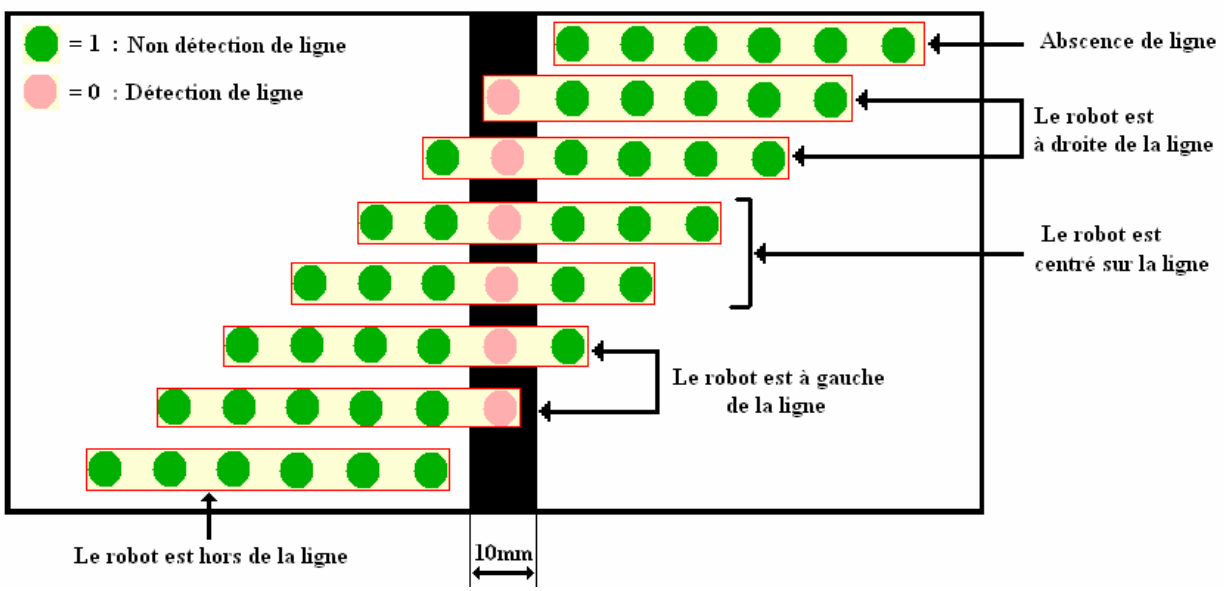


Figure 49 : cas de ligne d'épaisseur = 10mm

Combinaison binaire	L'état du robot % à la ligne	Commande des moteurs
1 1 1 1 1 1	Absence de ligne	Arrêter les deux moteurs
0 1 1 1 1 1	Robot à droite de la ligne	Freinage forte de moteur gauche pour revenir au centre
1 0 1 1 1 1	Robot à droite de la ligne	Freinage moyen de moteur gauche pour revenir au centre
1 1 0 1 1 1	Robot à droite de la ligne	Freinage très faible de moteur gauche pour revenir au centre
1 1 1 0 1 1	Robot à gauche de la ligne	Freinage très faible de moteur droit pour revenir au centre
1 1 1 1 0 1	Robot à gauche de la ligne	Freinage moyen de moteur droit pour revenir au centre
1 1 1 1 1 0	Robot à gauche de la ligne	Freinage forte de moteur droit pour revenir au centre
1 1 1 1 1 1	Robot hors de la ligne	Chercher la ligne

Ce tableau montre les valeurs possibles des détecteurs de ligne qui vont être transmis vers les ports d'entrés du pic de RB2 à RB7. Selon ces valeurs le pic va commander le robot.

2.2.2 Cas de ligne d'épaisseur > 10mm

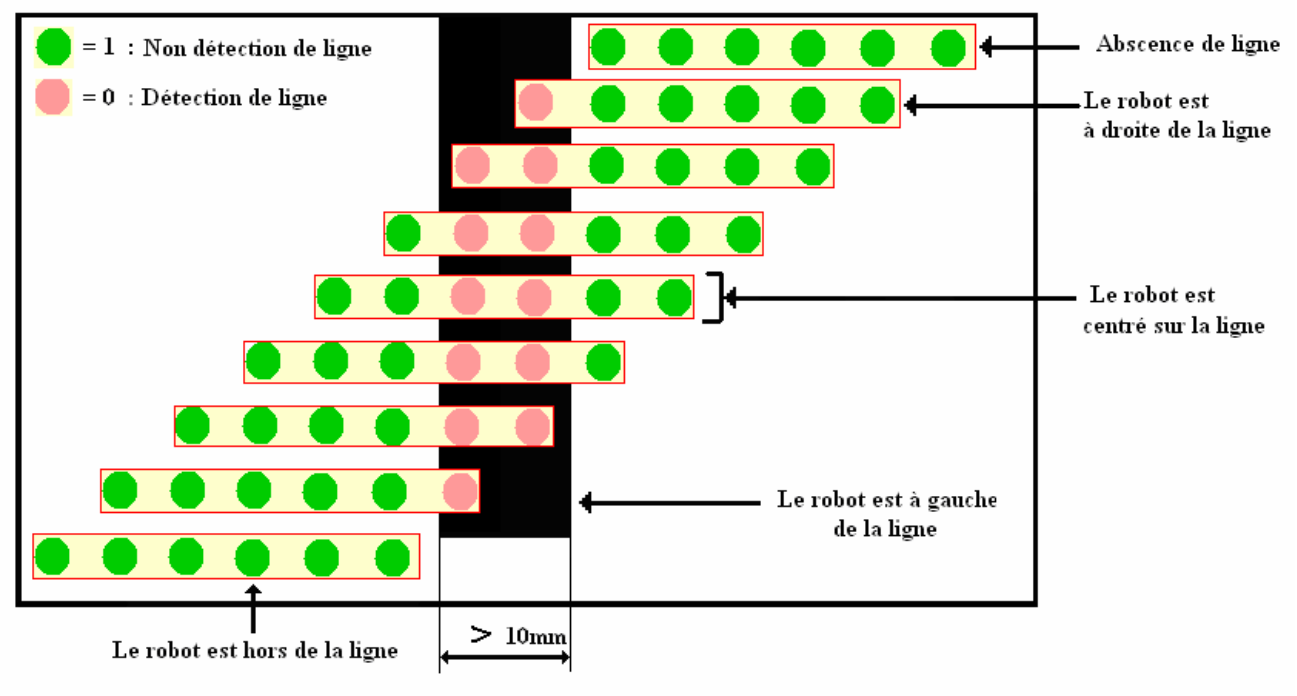


Figure 50 : cas de ligne d'épaisseur > 10mm

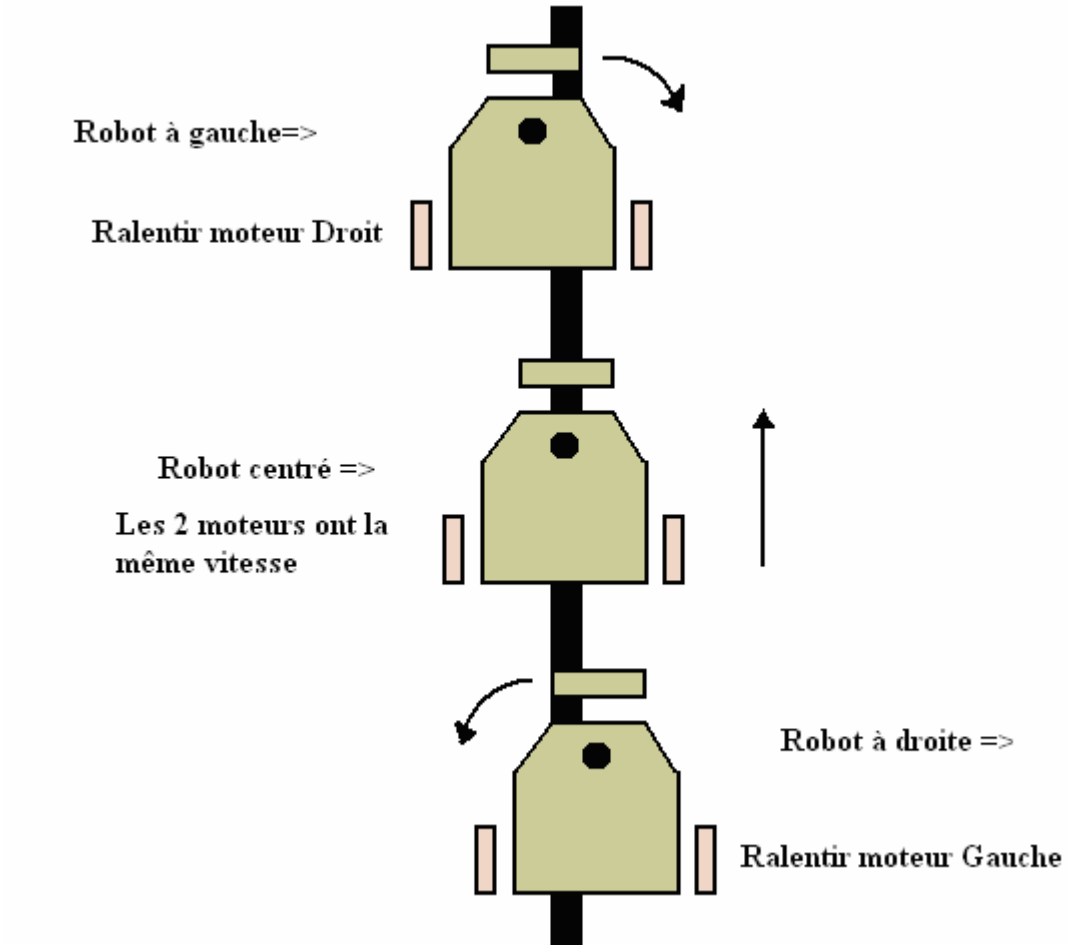


Figure 50 : La disposition du robot sur la ligne

2.3 Capteurs de détection de traits

Pour une détection de trait tracée à gauche ou à droite de la ligne, on utilise deux capteurs, un à droite et l'autre à gauche.

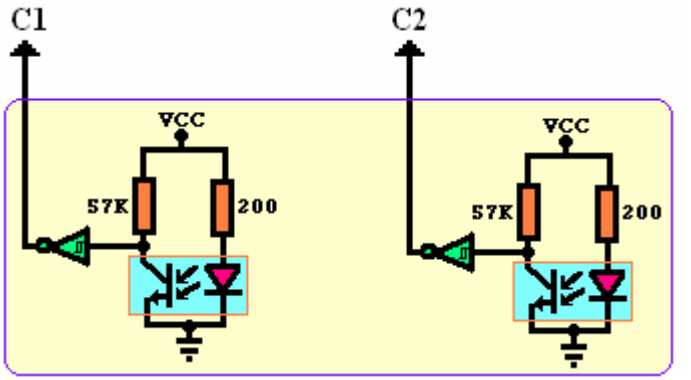
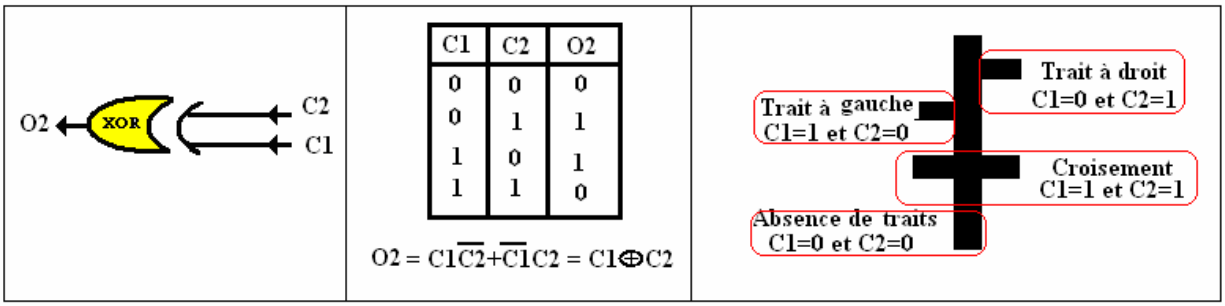


Figure 51 : capteurs de traits



2.4 Capteur de détection d'obstacle

Pour la détection d'obstacle, on a utilisé un capteur de même type OBP704. La sortie de ce capteur transporte la valeur 1 dans le cas de la détection et 0 dans le cas contraire vers l'entré de la porte logique OU (OR).

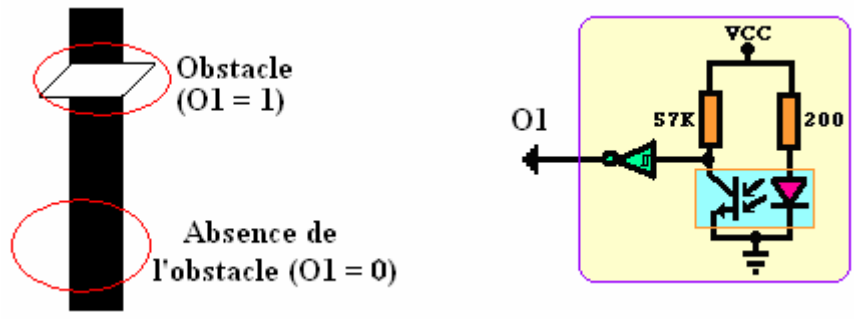
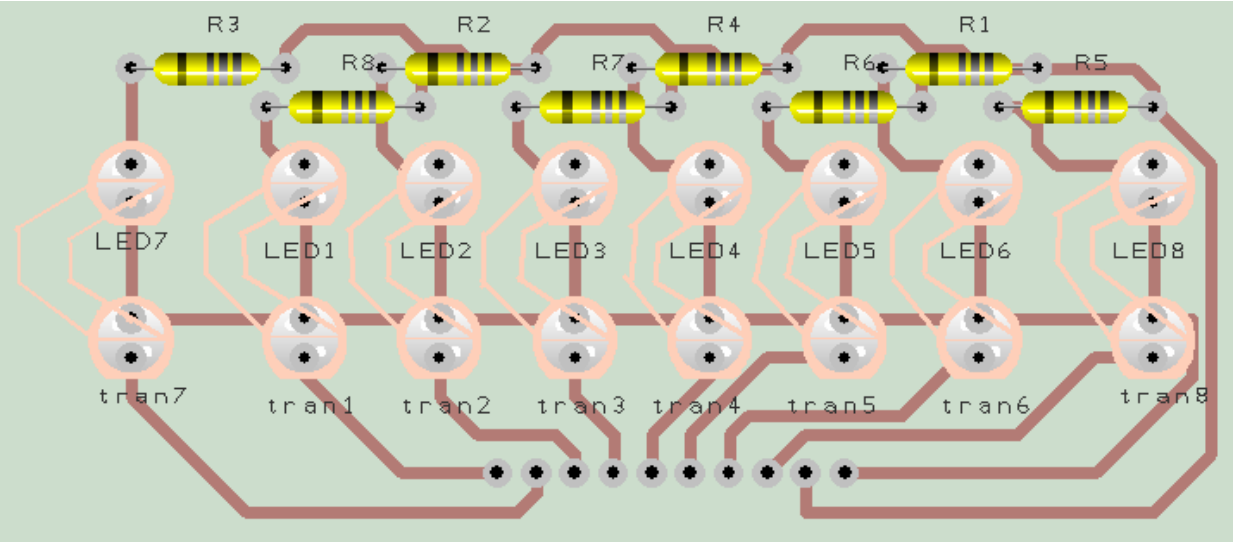


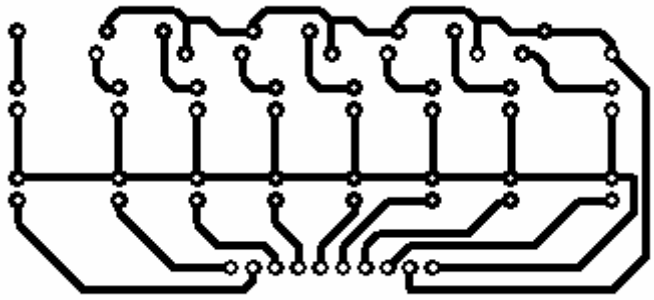
Figure 52 : Capteur de détection d'obstacle

2.5 Réalisation du Typon de la carte capteurs

COTE COMOPSANTS



COTE CUIVRE



2.6 Nomenclature de la carte des détecteurs

Désignation	Référence	Quantité	Symbole/Réel	Prix unitaire (DH)
Résistances	200 Ω	9	 	1
	57 kΩ	9		1
Détecteurs	OBP704	9		70 (7€)

3. La carte du Microcontrôleur

La carte de PIC possède :

- Des entrées RB2 à RB7 sur lesquelles arrivent les informations provenant des opto-détecteurs (photo-transistor) après une mise en forme de signal
- Des entrées RB0 et RB1 sur lesquelles arrivent les informations provenant des deux détecteurs de traits et du détecteur d'obstacle pour former une interruption
- Des sorties RA0, RA1, RA2 et RA3 qui commandent le moteur gauche et le moteur droit
- Des portes logiques 74HC86, 74HC32, 74HC14, 74HC04 et 74HC08

3.1 Le schéma de la carte

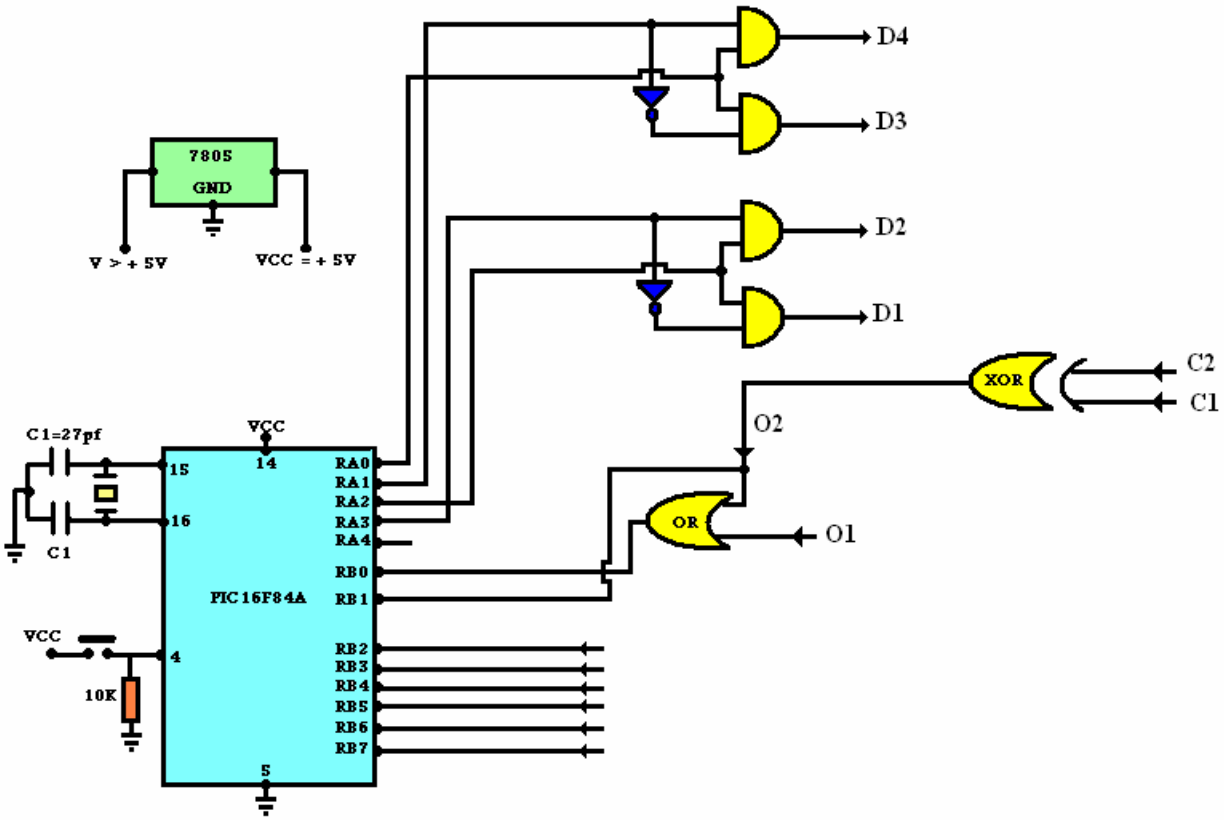


Figure 53 : Schéma de la carte de microcontrôleur

La porte logique OU (74HC32) a deux entrées O1 et O2 :

- L'entrée O1 est la sortie de détecteur d'obstacle qui vaut 1 au cas de la détection d'obstacle et vaut 0 au cas de non détection
- L'entrée O2 est la sortie de la porte logique XOR (74HC86)

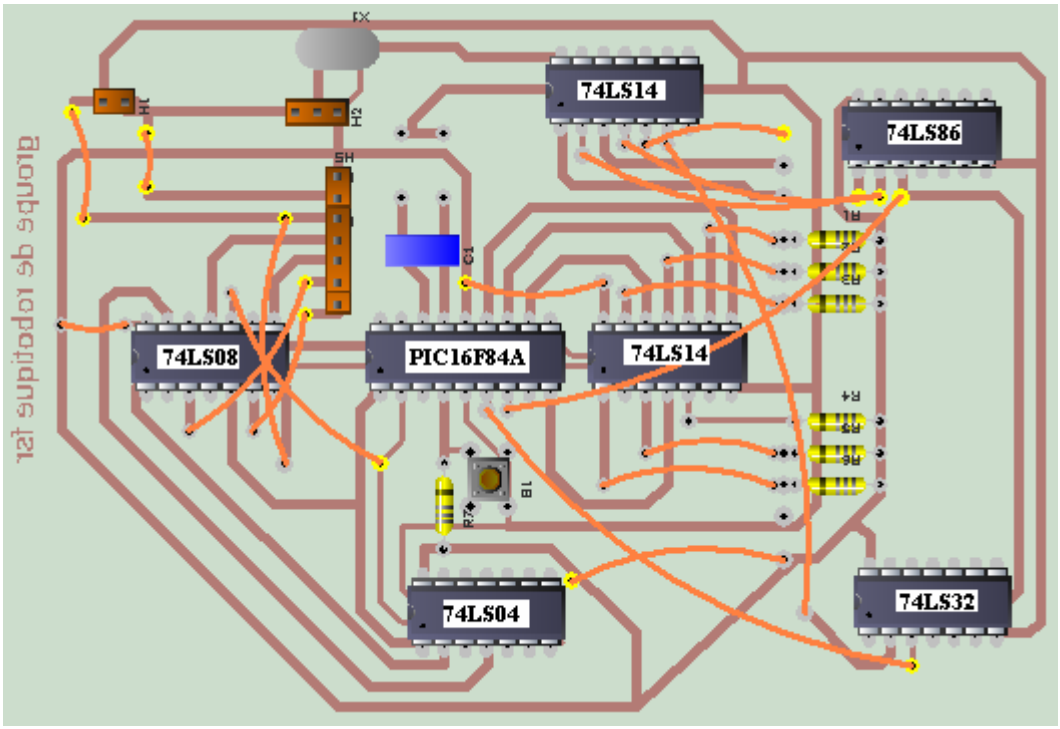
O1	O2	Y=RB0
0	0	0
0	1	1
1	0	1
1	1	1

Y= a + b

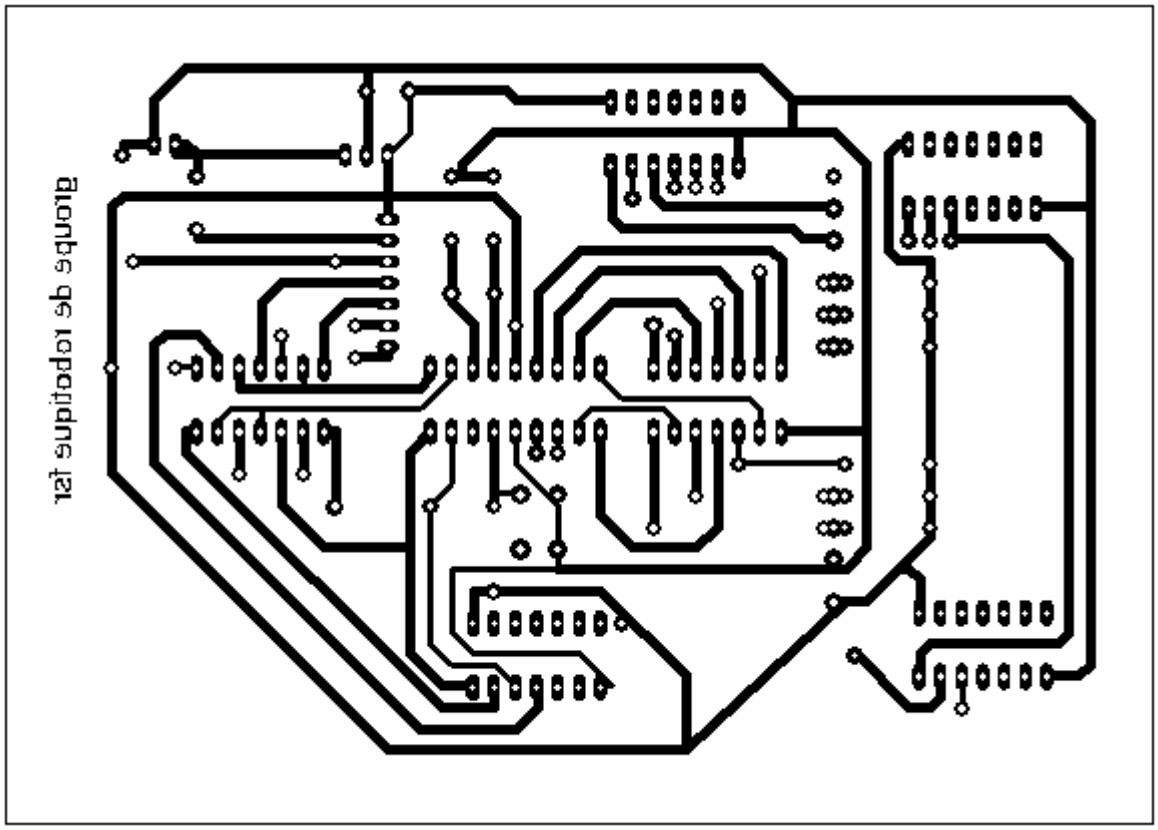
Les portes logiques AND (74HC08) agir sur les sens des moteurs.

3.2 Typon de la carte

COTE COMOPSANTS



COTE CUIVRE



3.3 Nomenclature de la carte du microcontrôleur :

Désignation	Référence	Quantité	Symbole/Réel	Prix unitaire (DH)
Microcontrôleur	PIC16F84A	1		50
Trigger of schmidth	74LS14	2		12
Porte XOR	74LS86	1		12
Porte OR	74LS32	1		12
Inverseur	74LS04	1		8
Régulateur de Tension	7805	1		7
Support de pic	9 broches	1		5

Quartz	4Mhz	1		10
Capacité	27pf	2		3
Résistance	10 kΩ	7		1

4. La carte de commande de puissance des moteurs

Les signaux de commande sont générés par un programme du microcontrôleur puis amplifiés grâce à un pont en H, capable de délivrer le courant nécessaire.

4.1 Amplification des signaux émis par le microcontrôleur

Le signal reçu en sortie de microcontrôleur est très faible d'où la nécessité de l'amplifier. Le montage d'amplification possède un double pont en H capable de délivrer un courant suffisant à chaque moteur et de les commander séparément.

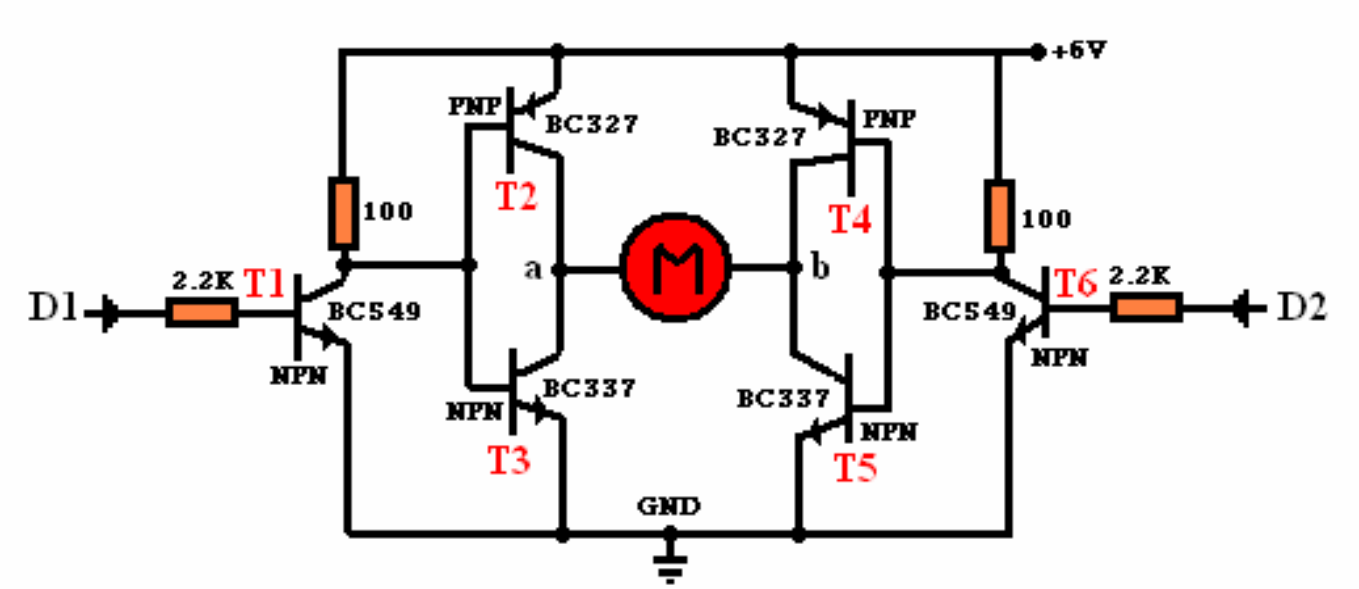


Figure 54 : Montage de commande de moteur M

4.2 Le Moteur M1

Le tableau qui suit le schéma montre l'état du moteur **M1** selon les valeurs des signaux **D1** et **D2**.

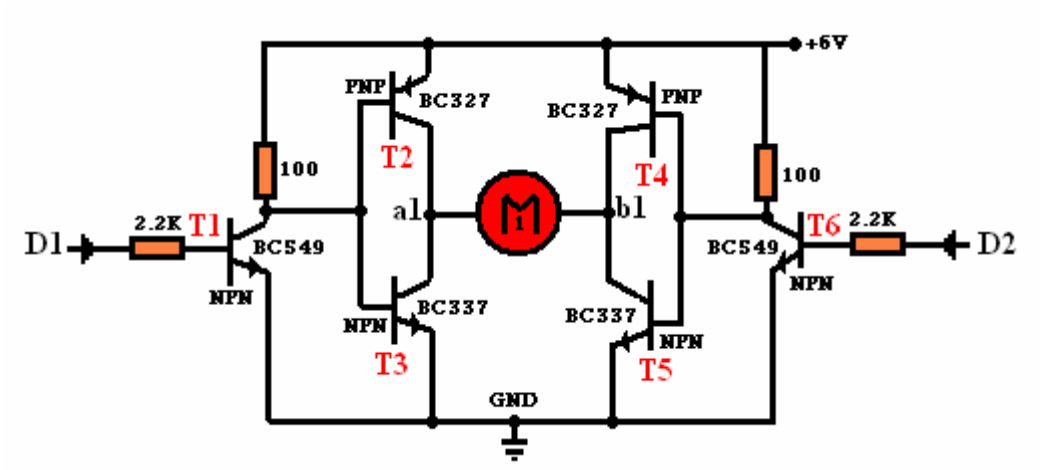


Figure 55 : Montage de commande de moteur **M1**

D1	D2	T1	T2	T3	T4	T5	T6	L'état du moteur M1
0	**0**	Bloqué	Bloqué	Saturé	Bloqué	Saturé	Bloqué	Le moteur au repos (les deux broches **a1** et **b1** du moteur sont alimentées par la masse (**GND**))
0	**1**	Bloqué	Bloqué	Saturé	Saturé	Bloqué	Saturé	Le moteur marche au sens normal : - **GND** → broche **a1** - **+ 6V** → broche **b1**
1	**0**	Saturé	Saturé	Bloqué	Bloqué	Saturé	Bloqué	Le moteur marche au sens inverse : - **+ 6V** → broche **a1** - **GND** → broche **b1**
1	**1**	Saturé	Saturé	Bloqué	Saturé	Bloqué	Saturé	Le moteur au repos (les deux broches **a1** et **b1** du moteur sont alimentées par **+6V**)

4.3 Le Moteur M2

Le tableau qui suit le schéma montre l'état du moteur **M2** selon les valeurs des signaux **D3** et **D4**.

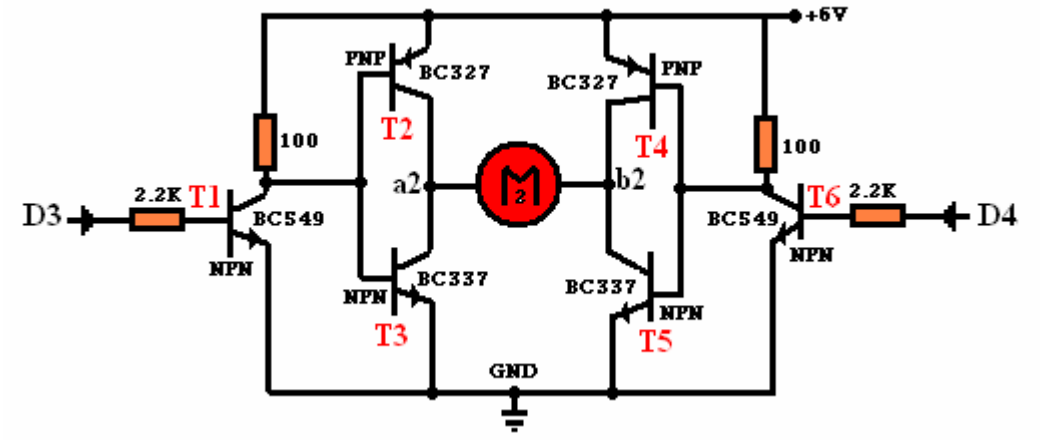


Figure 56 : Montage de commande de moteur **M2**

D3	D4	T1	T2	T3	T4	T5	T6	L'état du moteur M2
0	**0**	Bloqué	Bloqué	Saturé	Bloqué	Saturé	Bloqué	Le moteur au repos (les deux broches **a2** et **b2** du moteur sont alimentées par la masse (**GND**))
0	**1**	Bloqué	Bloqué	Saturé	Saturé	Bloqué	Saturé	Le moteur marche au sens normal : - **GND** → broche **a2** - **+ 6V** → broche **b2**
1	**0**	Saturé	Saturé	Bloqué	Bloqué	Saturé	Bloqué	Le moteur marche au sens inverse : - **+ 6V** → broche **a2** - **GND** → broche **b2**
1	**1**	Saturé	Saturé	Bloqué	Saturé	Bloqué	Saturé	Le moteur au repos (les deux broches **a2** et **b2** du moteur sont alimentées par **+6V**)

Remarque :

- Le Transistor **PNP** : - Si l'état de la base B=1 ➔ le transistor est bloqué
  - Si l'état de la base B=0 ➔ le transistor est Saturé

- Le Transistor **NPN** : - Si l'état de la base B=1 ➔ le transistor est Saturé
  - Si l'état de la base B=0 ➔ le transistor est bloqué

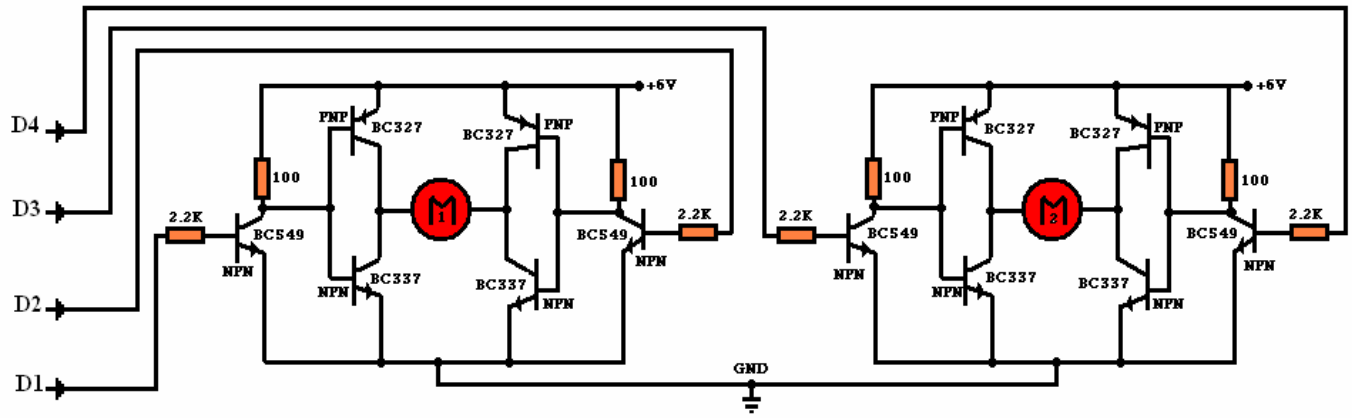
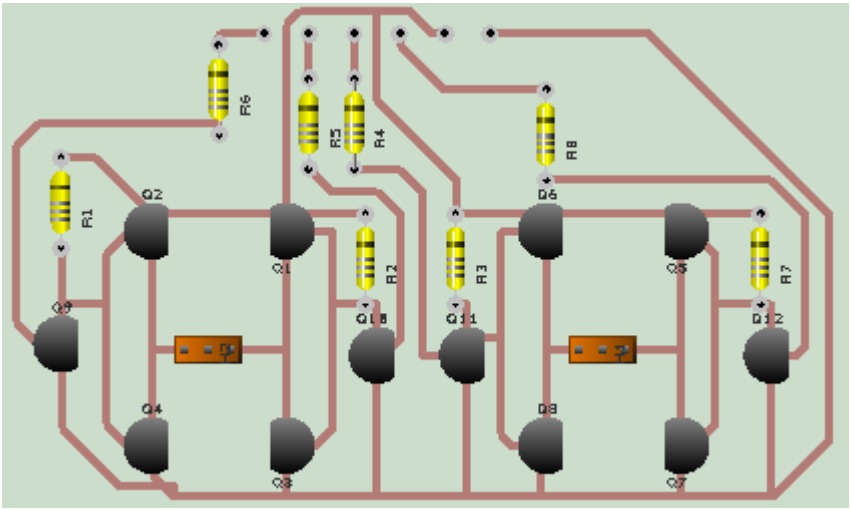


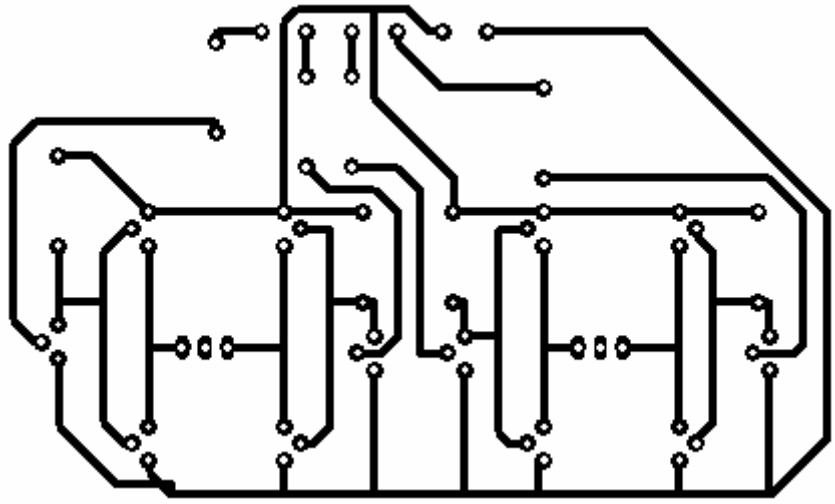
Figure 57 : Montage de commande des deux moteurs

4.4 Le circuit imprimé de la carte de commande de puissance

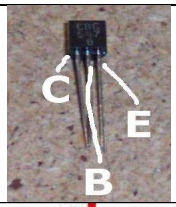
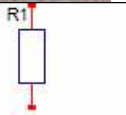
COTE COMOPSANTS



COTE CUIVRE



4.5 Nomenclature de la carte de commande de puissance

Désignation	Référence	Quantité	Symbole/Réel	Prix unitaire (DH)
Transistors	BC327 PNP	4		2
	BC337 NPN	4		2
	BC549 NPN	4		2
Résistances	100 Ω	4		1
	2.2 KΩ	4		

Servomoteur	hs322_s : Hitech	2		143 (13€)

5. La carte et logiciels de programmation

5.1 MPLAB

MPLAB est un outil de développement pour les PIC 16F84 et autres, il présente l'énorme avantage d'être gratuit. On le télécharge sur le site de Microchip.

Il contient :

- un éditeur,
- un assembleur,
- un simulateur.

Il permet :

- La rédaction du fichier source en langage assembleur (fichier.ASM).
- Sa transformation en fichier objet (fichier.HEX) prêt à être chargé dans la mémoire programme du microcontrôleur.
- L'ensemble des fichiers nécessaires à ces opérations est regroupé dans un espace " projet " (fichier.PJT).
- Tous les fichiers d'un même projet doivent porter exactement le même nom que le projet lui même, seules diffèrent les extensions.

Dans l' Annexe III les vues d'écran nous permettent de créer une application.

5.2 La carte de programmation

Très simple, ce petit programmateur est facile à réaliser. Il doit être connecté à la prise série (DB9) d'un ordinateur type PC au moyen d'un câble standard non croisé (Modem). Il nécessite une alimentation de 5V. Le logiciel "PP84.exe" (freeware) permet de le mettre en oeuvre facilement sous Microsoft Windows. La masse du PC est connectée à la masse (Vss) du PIC, ce qui permet la programmation in situ.

5.2.1 Principe d'interconnexion au PC

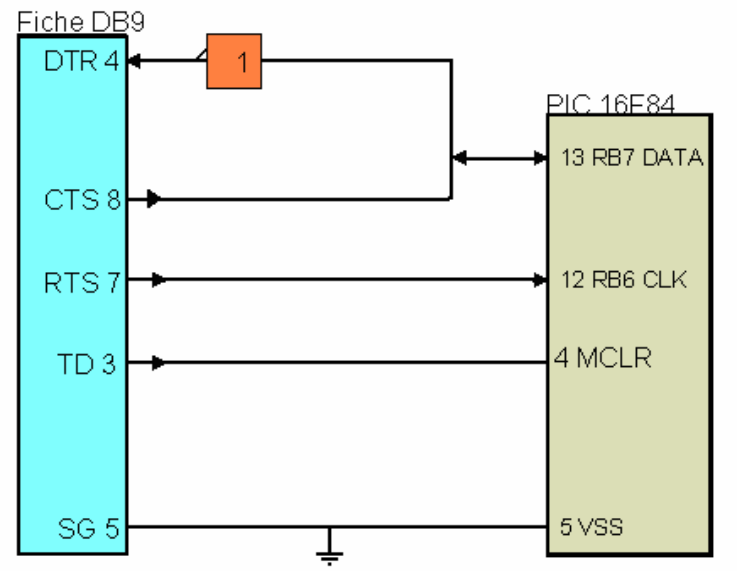


Figure 58

5.2.2 Schéma de structures

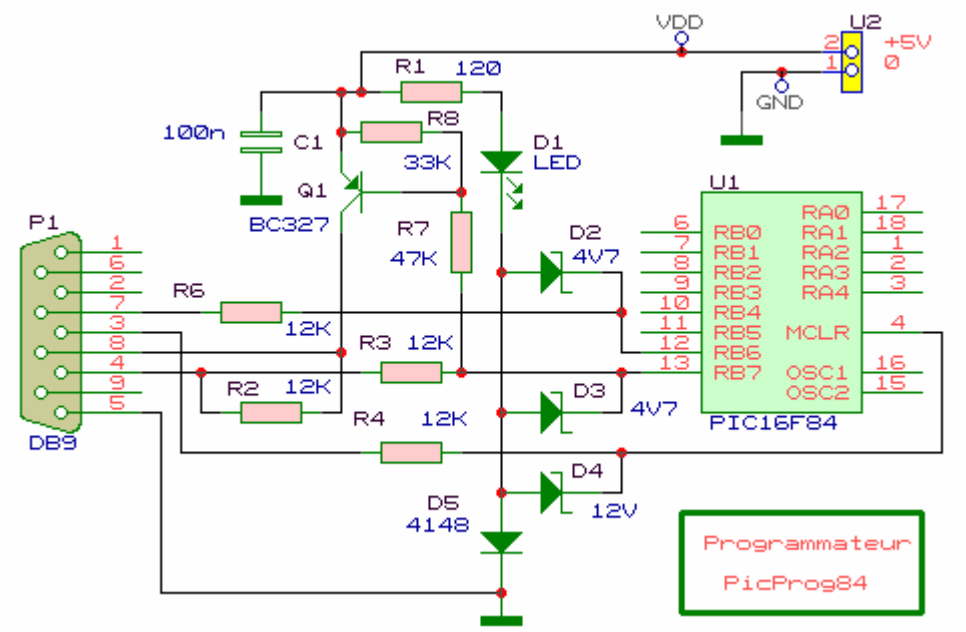
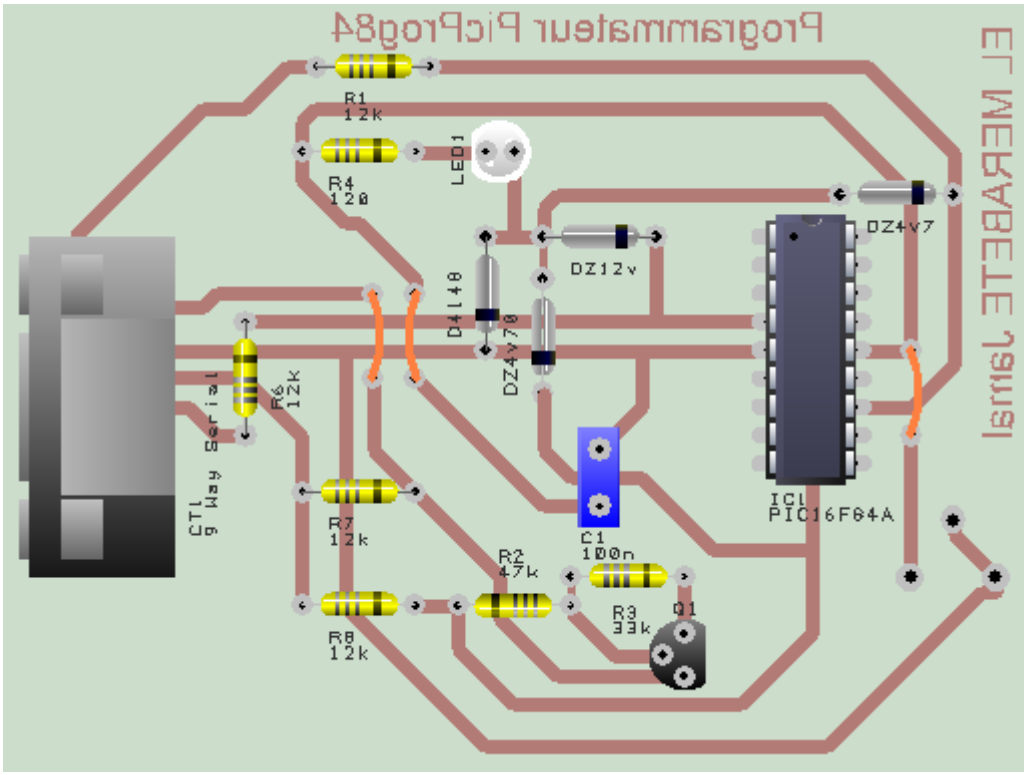


Figure 59 : Schéma de structures

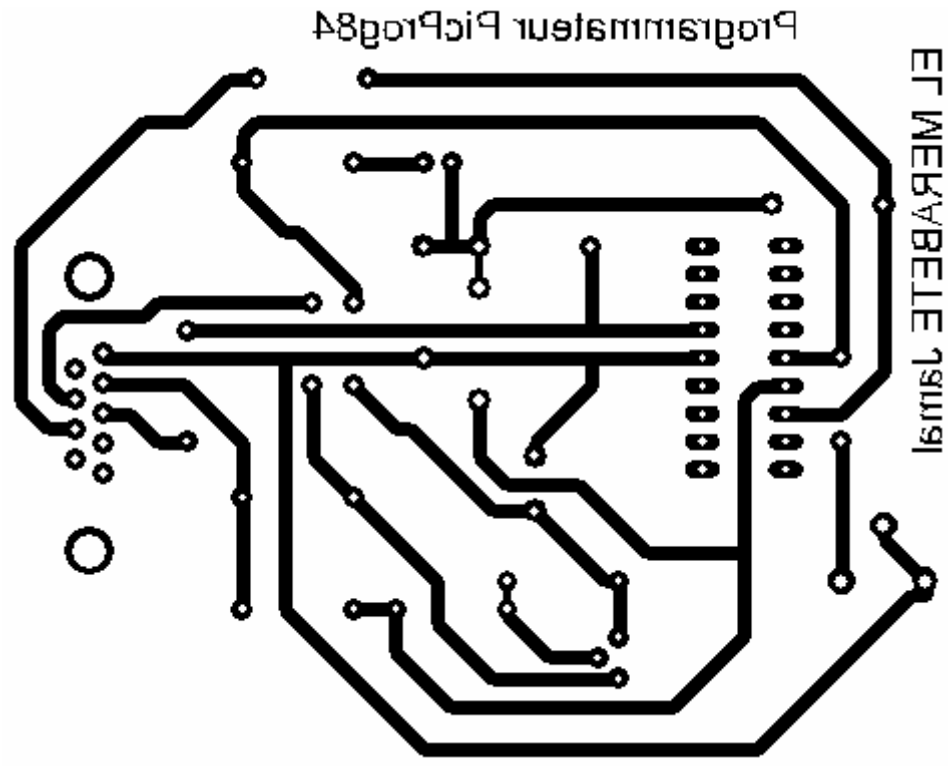
5.2.3 Réalisation du circuit imprimé

Le circuit imprimé et son implantation montre la simplicité de la réalisation. Il est important d'utiliser un bon support pour le PIC, un modèle *tulipe*. Car il faut pouvoir placer et enlever souvent le composant à programmer. On peux utiliser une rallonge de port série normal, c'est à dire non croisée.

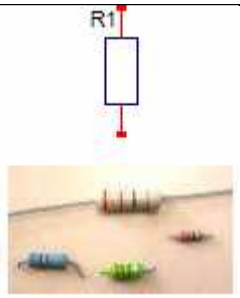
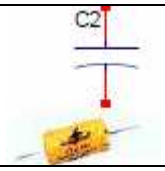
COTE COMOPSANTS



COTE CUIVRE



5.2.4 Nomenclature de la carte de programmation

Désignation	Référence	Quantité	Symbole/Réel	Prix unitaire (DH)
Résistances	R1:120 Ω (marron, rouge, marron)	1		1
	12kΩ	4		1
	47kΩ (jaune, move, orange)	1		1
	33kΩ (orange, orange, orange)	1		1
Capacité	100 nf	1		2
Transistor	BC327 PNP	1		2
LED	Rouge 3mm faible consommation	1		1
Diodes	Zener 4,7 V	2		2
	Zener 12V	1		2
	4148	1		1
Connecteur	DB9 femelle	1		15
Support	18 broches tulipe	1		5
câble	rallonge du port série	1		--

2 Transfert du programme sur le PIC

Pour transférer le programme sur le PIC, nous allons utiliser PP84. La procédure pour programmer le PIC16F84A décrire ci-après suppose que les options de configuration sont au départ celles par défaut. Il est toujours possible de réinitialiser la configuration par défaut dans l'onglet « Config ».

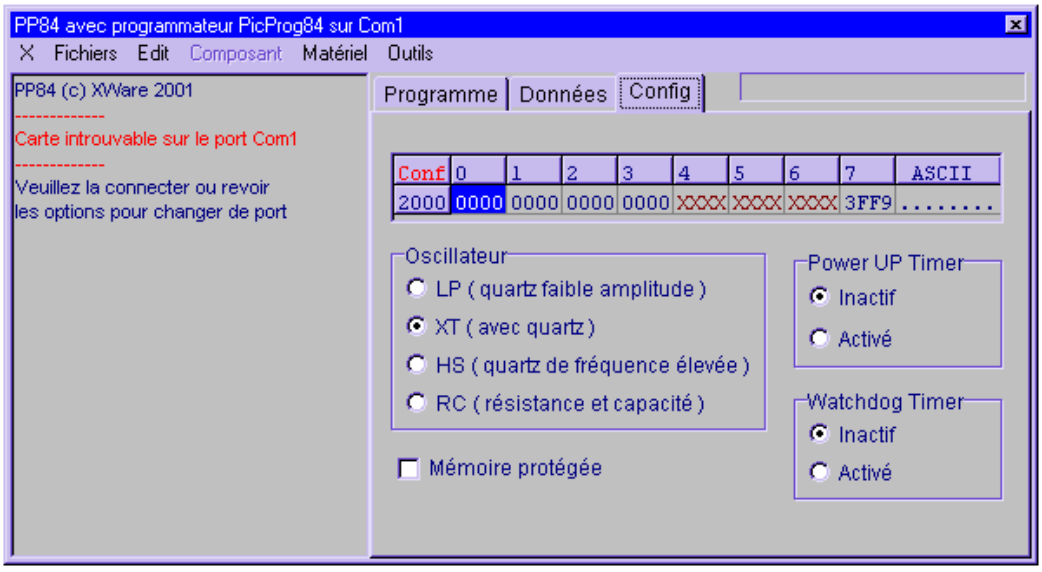


Figure 60 : Le PP84

Chapitre 3 : la programmation

1. Rappel du cahier de charges fonctionnel et technique

Le suivi de ligne est le cœur du robot, c'est de cette partie que va surgir les différentes stratégies pour gagner en vitesse, position et en temps…

Les différentes fonctions sont :

- Analyser la trace
- Anticiper un virage par ralentissement
- Détecter que le marquage au sol indique une priorité à respecter (cas de trait)
- Détecter un obstacle indique une priorité à respecter.

La priorité est lancée grâce à l'interruption que l'on va expliquer après.

Cette programmation se fera en langage assembleur MPASM.

2. Fonctionnement :

Tout d'abord tirer le jack de robot pour mettre à marche et poser le sur la ligne, le robot doit avancer tout droit, s'il rencontre un trait à gauche grâce à un détecteur de coté gauche, il doit faire un tour complète de 360° en inversant le sens de moteur droit. Après le robot marche en avant sur la ligne. S'il rencontre un trait à droit grâce à un détecteur de coté droit, il doit faire un tour complète de 360° en inversant le sens de moteur droit. Après la tour le robot avance tout droit sur la ligne et s'il rencontre un croisement, le robot doit marcher en avant.
Si le robot rencontre un obstacle pour la 1ère fois, il fait un demi-tour (180°) et marche en avant.
Si le robot encontre un obstacle pour la 2ème fois, il s'arrête.

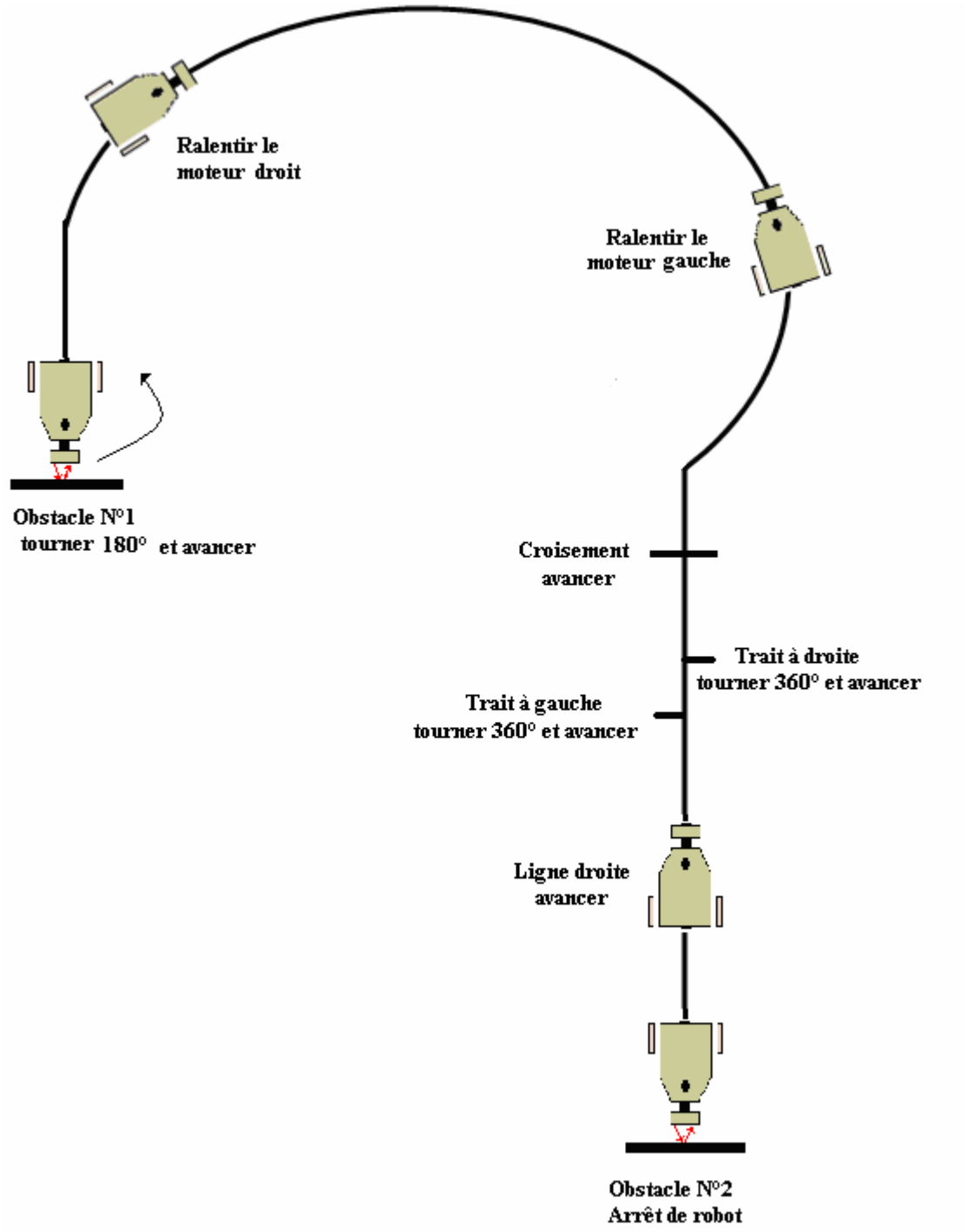


Figure 61 : différents situations du robot

3. Organigramme :

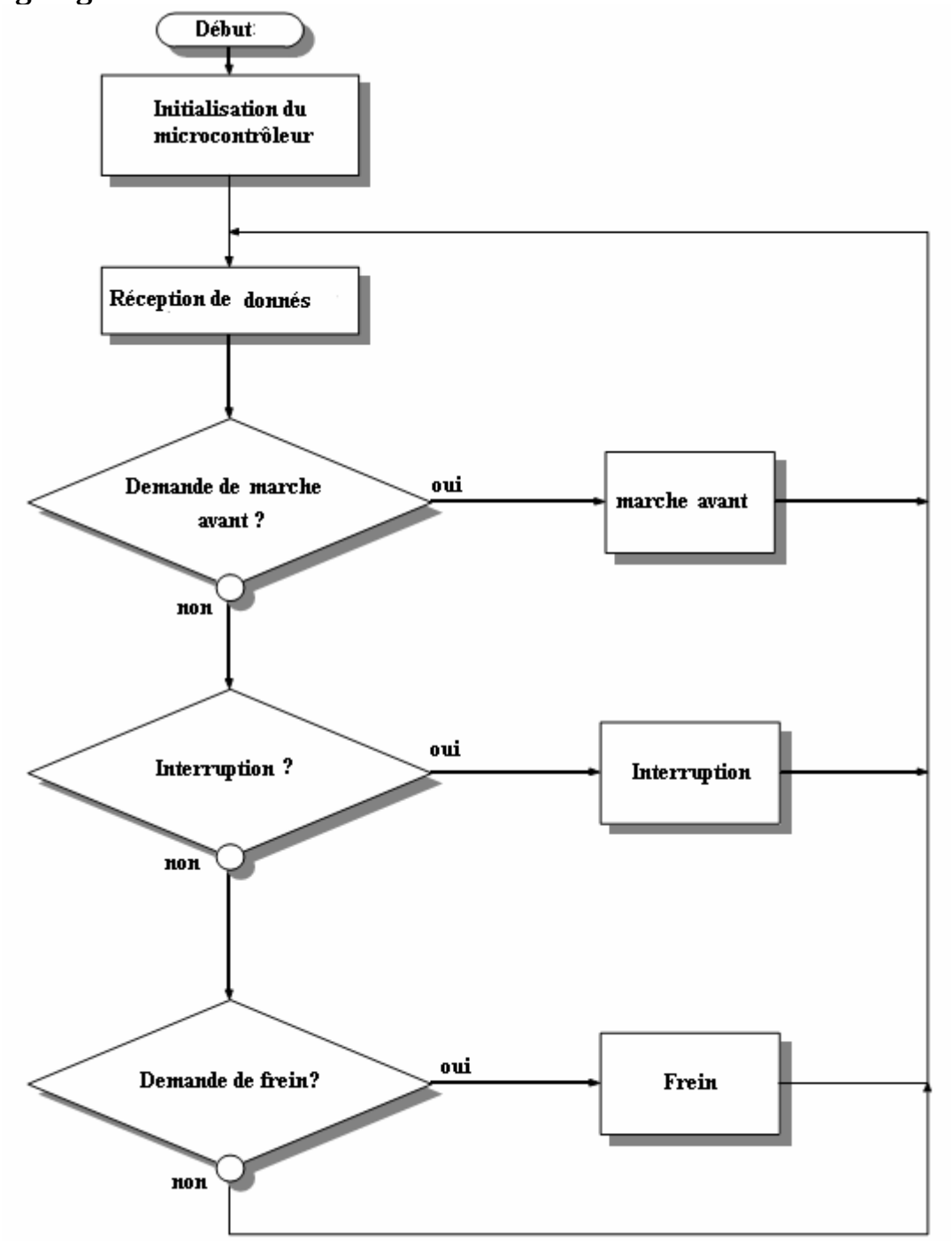


Figure 62 : organigramme de la programmation de gestion de la commande des moteurs

4. Explication de l'organigramme :

On doit tout d'abord initialiser le microcontrôleur pour permettre son fonctionnement. Ensuite on reçoit des données provenant d'un sous programme qui nous fournit le sens de rotation des moteurs. Enfin selon les données reçues le programme envoie des signaux qui permettent de gérer la direction des moteurs, c'est-à-dire sa marche avant, rotation ou frein.

5. Algorithme principal

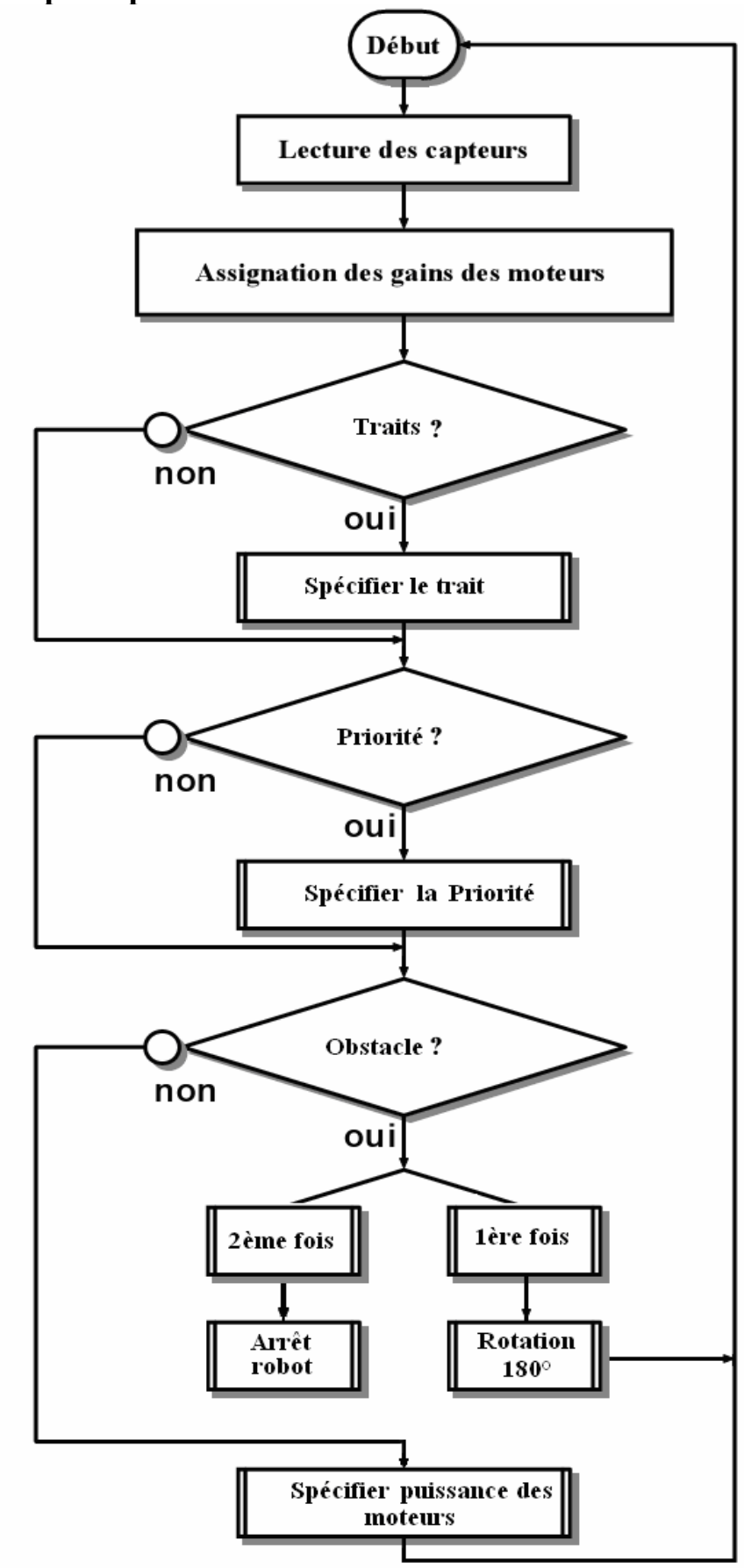


Figure 63 : Algorithme principal

6. Interruption

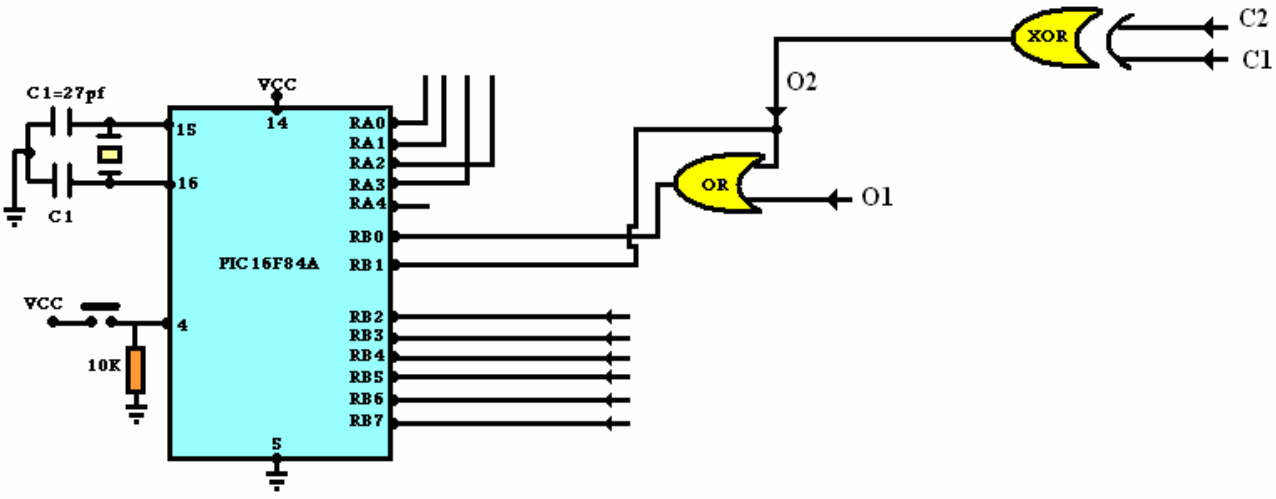


Figure : 64

La porte OU (OR) pour lancer l'interruption :

- Si la sortie de OU = 0 ➔ RB0 = 0 alors on n'aura pas d'interruption.
- Si la sortie de OU = 0 ➔ RB0 = 1 alors on aura l'interruption.

La porte XOR spécifié l'interruption :

- Si la sortie de XOR =1 alors l'interruption vient des détecteurs de traits pour lancer un tour complète
- Si la sortie de XOR = 0 alors l'interruption vient du détecteur d'obstacle.

7. Programme de l'interruption

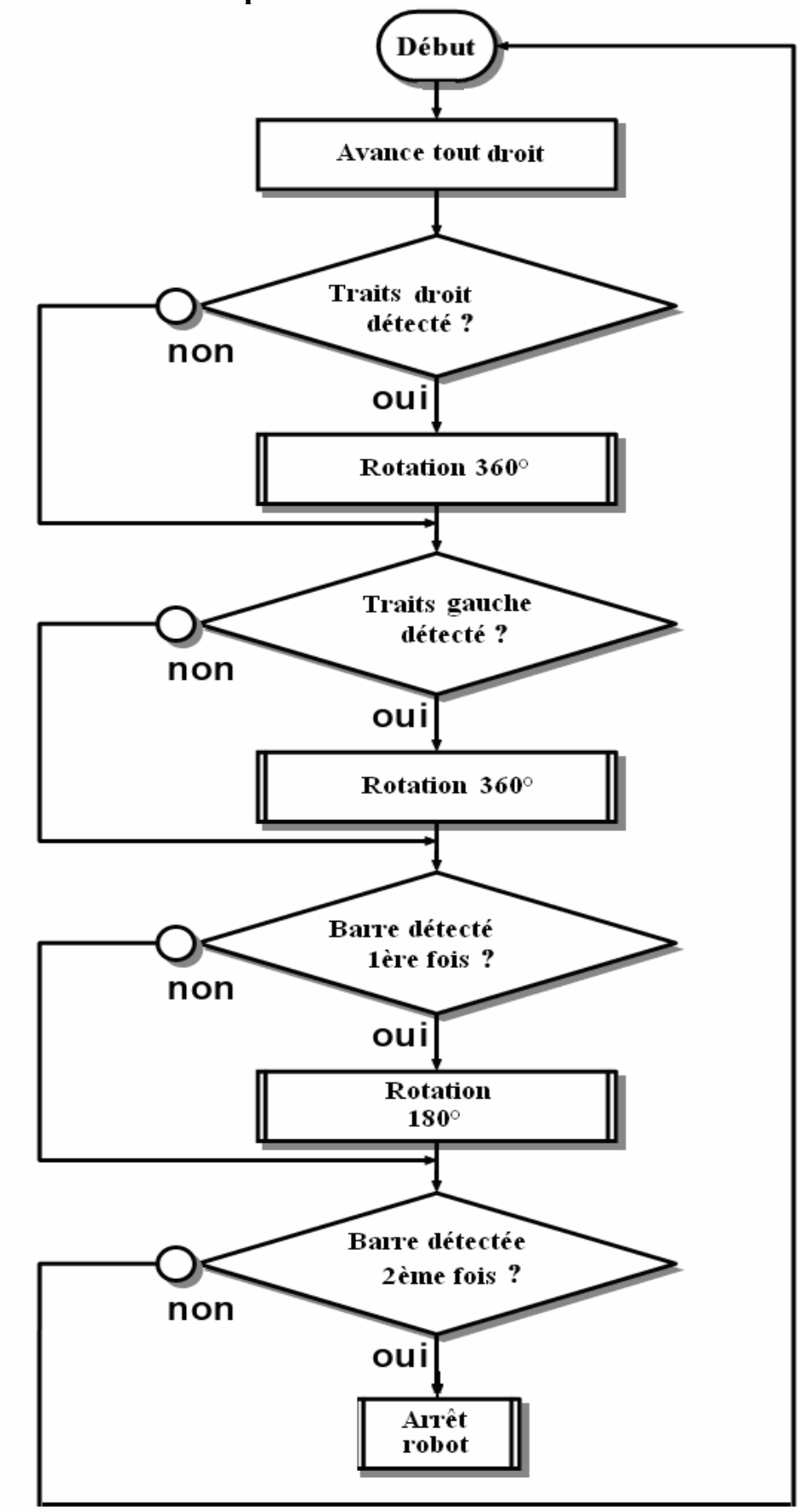


Figure 65 : programme de l'interruption

CONCLUSION GÉNÉRALE

Ce projet couvre un grand nombre de domaines de compétences :

- réalisation d'une carte électronique (CAO)
- étude de la commande de moteur en MLI
- programmation microcontrôleur
- rédaction d'un dossier de projet

A la fin de ce travail qui consiste a l'étude et la mise en œuvre d'un robot suiveur de ligne .le travail de ce type de robot a exige la maîtrise de plusieurs notion**s** techniques aussi bien le domaine micro électronique et celui de l'informatique, ce que fait que j'ai eu l'occasion à découvrir, la programmation bas niveau par **Mplab**, les techniques nécessaires pour l'utilisation des microcontrôleurs dans les montages électroniques.

D'après le planning que nous nous étions fixés, on peut dire que nous sommes arrivés à terme du projet dans le temps.

PARTIE III

ANNEXES

ANNEXE 1 : JEU D'INSTRUCTION

MNÉMONIQUES	PARAMÈTRES	EXPLICATION	ACTIVATION DRAPEAUX
Instructions qui manipulent des registres			
addwf	f, d	ADDITIONNE W et f	C, DC, Z
andwf	f, d	AND de W et f	Z
clrf	f	MISE à 0 de f	Z
clrw		MISE à 0 de W	Z
comf	f, d	COMPLÉMENT de f	Z
decf	f, d	DÉCRÉMENT de f	Z
incf	f, d	INCRÉMENT de f	Z
iorwf	f, d	OU entre W et f	Z
movf	f, d	DÉPLACEMENT de f	Z
movwf	f	DÉPLACEMENT de W dans f	
rlf	f, d	ROTATION à gauche avec retenue	C
rrf	f, d	ROTATION à droite avec retenue	C
subwf	f, d	SOUSTRAIT W de f	C, DC, Z
swapf	f, d	ÉCHANGE les quatre bits de poids fort et les 4 bits de poids faible	
xorwf	f, d	OU exclusif entre W et f	Z
Instructions qui manipulent des bits			
bcf	f, b	MISE à 0 bit b de f	
bsf	f, b	MISE à 1 bit b de f	
Instructions qui manipulent des opérandes immédiats			
addlw	k	ADDITIONNE literal à W	C, DC, Z
andlw	k	AND entre literal et W	Z
iorlw	k	OR entre literal et W	Z
movlw	k	DÉPLACEMENT de literal dans W	
sublw	k	SOUSTRAIT W de literal (K-W))	C, DC, Z
xorlw	k	OU exclusif entre literal et W	Z
Instructions de saut			
btfsc	f, b	TEST du bit b de f. SAUT si 0	
btfss	f, b	TEST du bit b de f. SAUT si 1	
decfsz	f, d	DÉCRÉMENT de f. ; SAUT si 0	
incfsz	f, d	INCRÉMENT de f. ; SAUT si 0	
Instructions de commande et spéciales			
call	k	APPEL à sous-programme	
clrwdt		MISE à 0 du WDT	TO#, PD#
goto	k	BRANCHEMENT sur une adresse	
nop		AUCUNE OPÉRATION	
retfie		RETOUR d'interruption	
retlw	k	RETOUR et charge de W avec literal	
return		RETOUR de sous-programme	
sleep		MISE du microprocesseur en stand-by ou sommeil	TO#, PD#

Figure 66

ANNEXE 2 : Exemple de programmes pour le pic16f84

LED Clignotante :

1) Faire Clignoter une LED

Nous considérons le montage suivant de la figure 64.

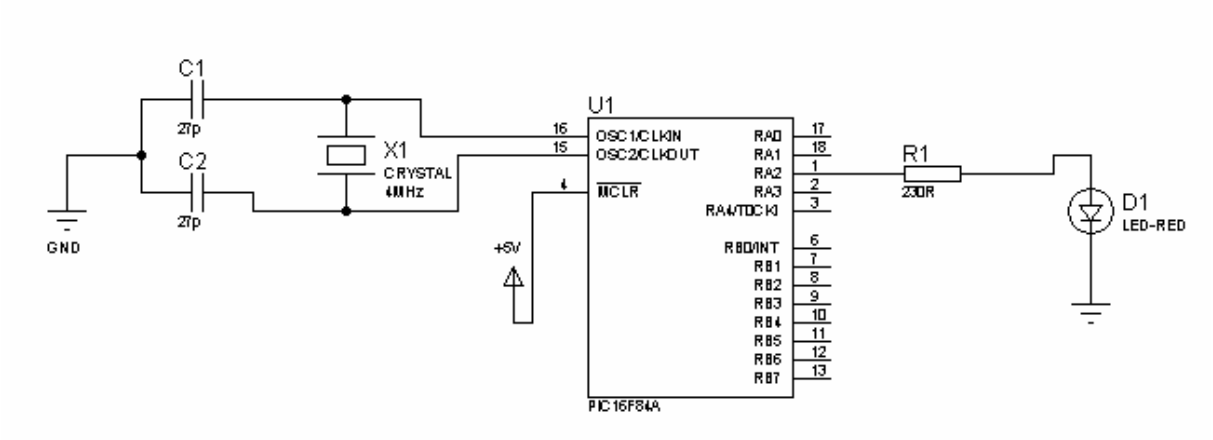


figure 67

Le programme suivant permet de faire clignoter la LED D1. C'est l'un des programmes les plus simples qui soit et qui est très pratique pour tester le bon fonctionnement des montages, qui devraient toujours avoir une LED témoin permettant d'afficher l'état du PIC.

Programme faisant clignoter une LED

```
LIST      p=16F84A            ; Définition de processeur

#include"P16F84A.INC"         ; fichier include
#define LED PORTA,2           ;Led rouge

cblock 0c                     ; début de la zone variables
c1:1
c2:1
endc                          ; fin de la zone



org 00                        ; Adresse de départ après reset
goto start                    ; Adresse 0 : initialiser

tempo:
movlw 00
movf c1,f
movf c2,f
ETC1:
ETC2:
incfsz c1,f
```

```
goto ETC2
incfsz c2,f
goto ETC1
return

start:
bsf STATUS,5                ; on passe en bank1
bcf TRISA,2                 ; RA2 en sortie
bcf STATUS,5                ; on repasse en bank0

first:
bsf LED                     ; Allumer LED
call tempo                  ; tempo de 0.5s
call tempo
call tempo
bcf LED                     ; Eteindre LED
call tempo                  ; tempo de 0.5s
call tempo
call tempo
goto first                  ; boucler

end
```

Algorithme : L'algorithme, très simple, est présenté sur la figure 65.

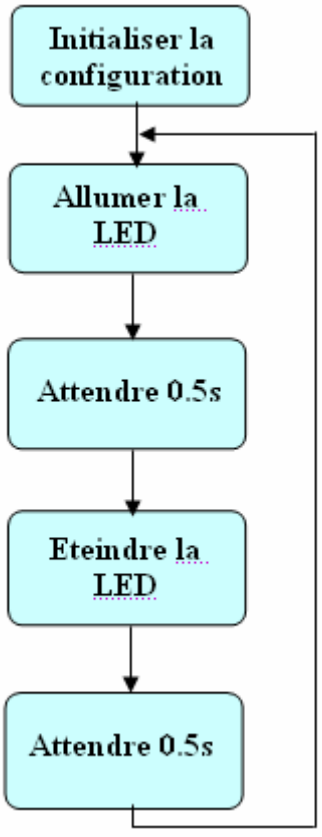


Figure 65 : Algorithme du programme faisant clignoter une LED

2) Led clignotante avec bouton poussoir

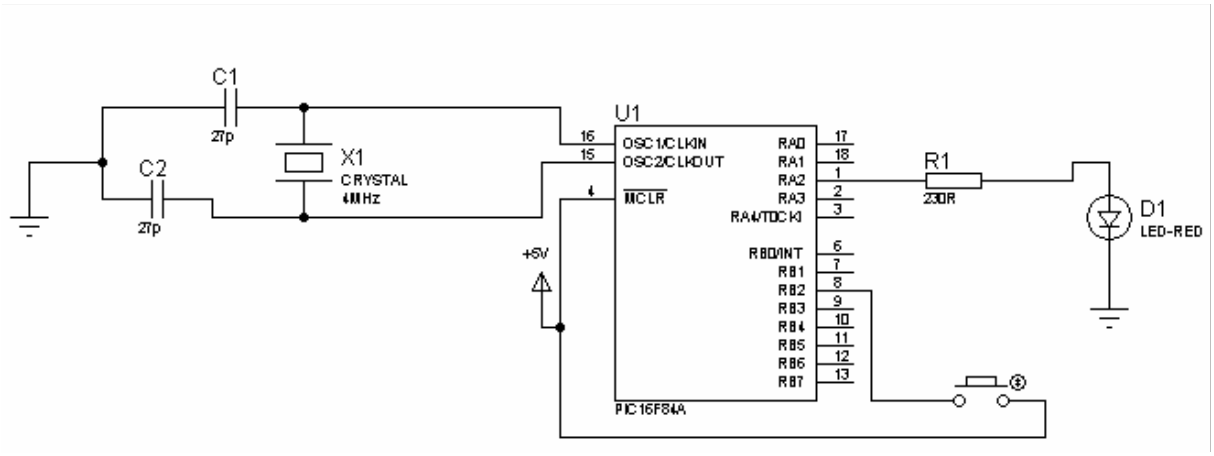


Figure 68 : Led clignotante avec bouton poussoir

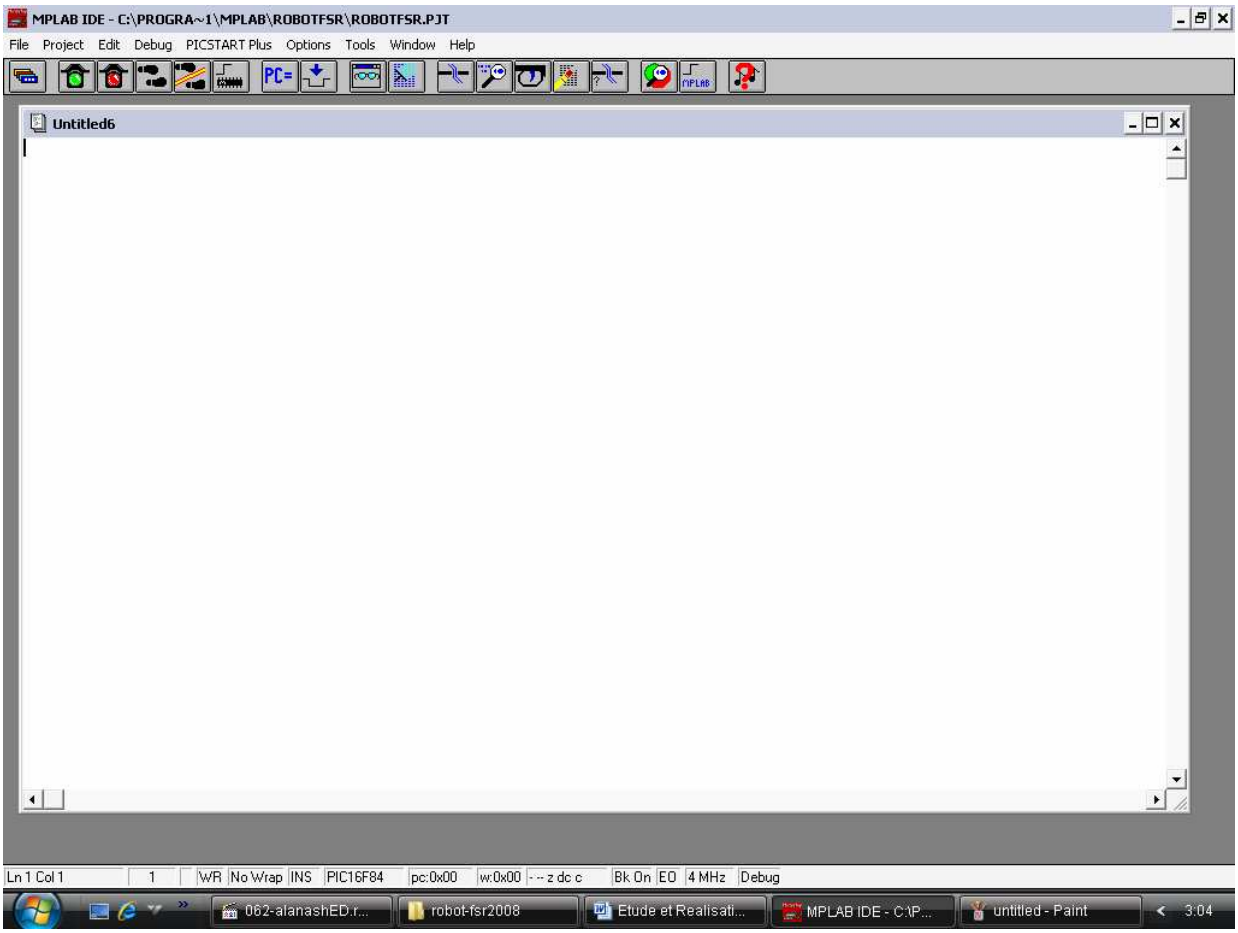
Programme faisant clignoter une LED avec bouton poussoir

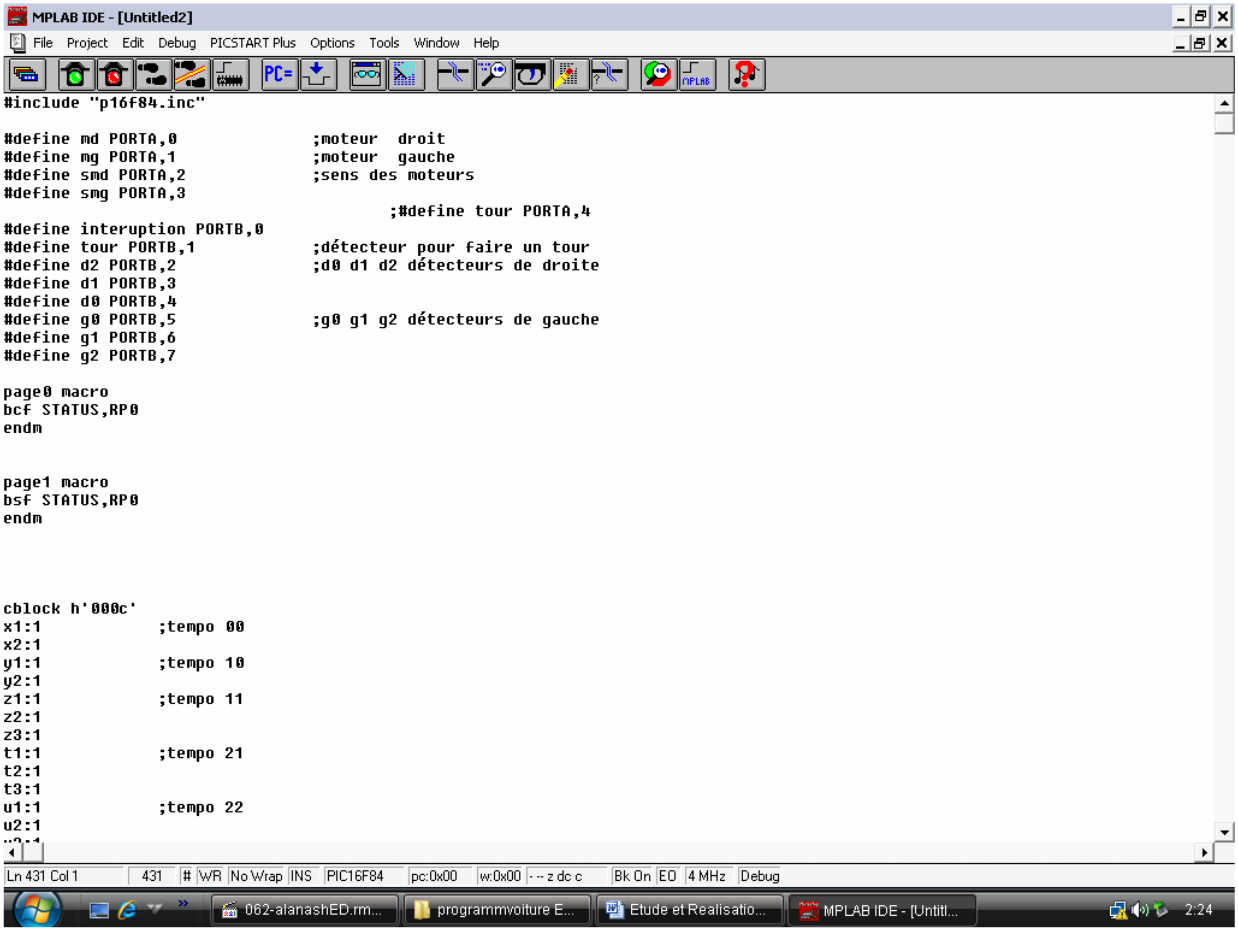
```
LIST      p=16F877A             ; Définition de processeur

#include"P16F84A.INC"          ; fichier include
#define LED PORTA,2            ;Led rouge
#define switch PORTB,2         ;bouton pousoir
cblock 0c
c1:1
c2:1
endc



org 00                ;Adresse de départ après reset
goto start            ; Adresse 0 : initialiser

tempo:
movlw 00
movf c1,f
movf c2,f
ETC1:
ETC2:
incfsz c1,f
goto ETC2
incfsz c2,f
goto ETC1
return

start:
bsf STATUS,5                    ; bank1
bcf TRISA,2                     ; RA2 en sortie
bcf STATUS,5                    ; mise à 0
```

```
BTFSC switch
goto first
goto off
off:
bcf LED

BTFSC switch
goto first
goto off
first:
bsf LED                         ; Allumer LED
call tempo                      ; tempo d 0.5s
call temp
call tempo
bcf LED                         ; Eteindre LED
BTFSS switch
goto off
call tempo                      ; tempo de 0.5s
call tempo
call tempo
BTFSS switch
goto off

goto first                      ; boucler

end
```

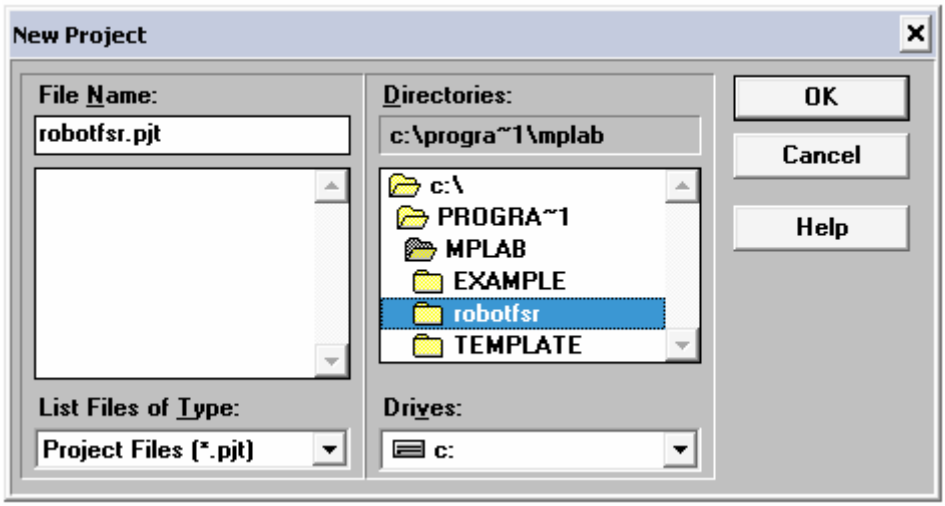
ANNEXE 3 : Création d'une application en MPLAB

Vous avez téléchargez MPLAB, celui est décompressé et installé sur votre disque dur. Lancer **MPLAB.exe** vous visualisez alors cet écran :

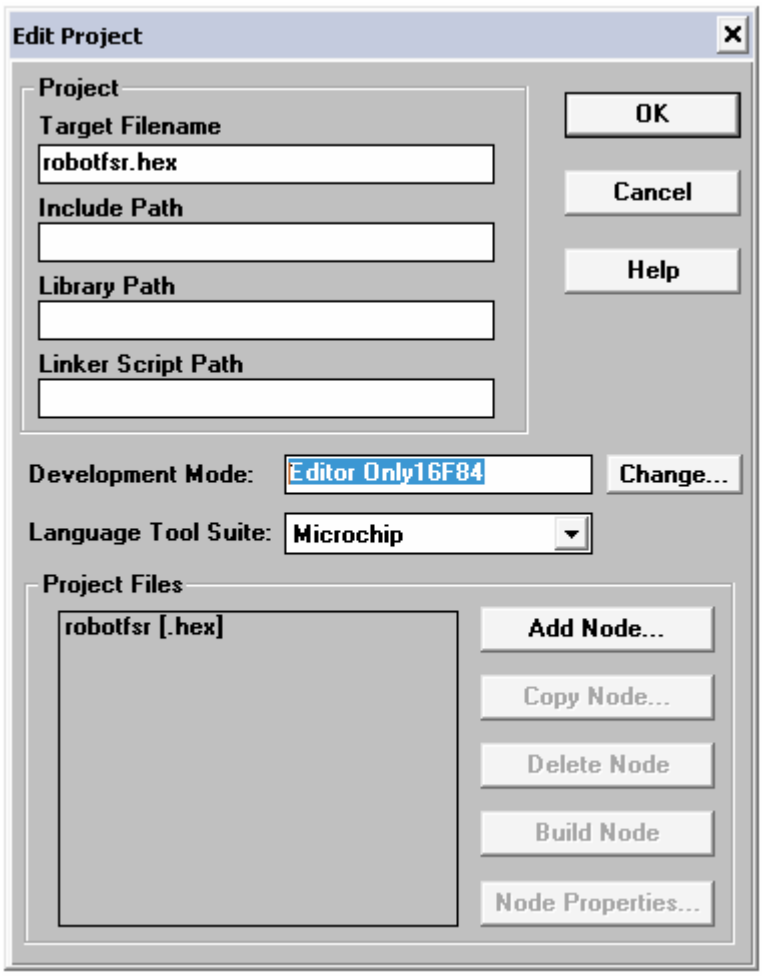




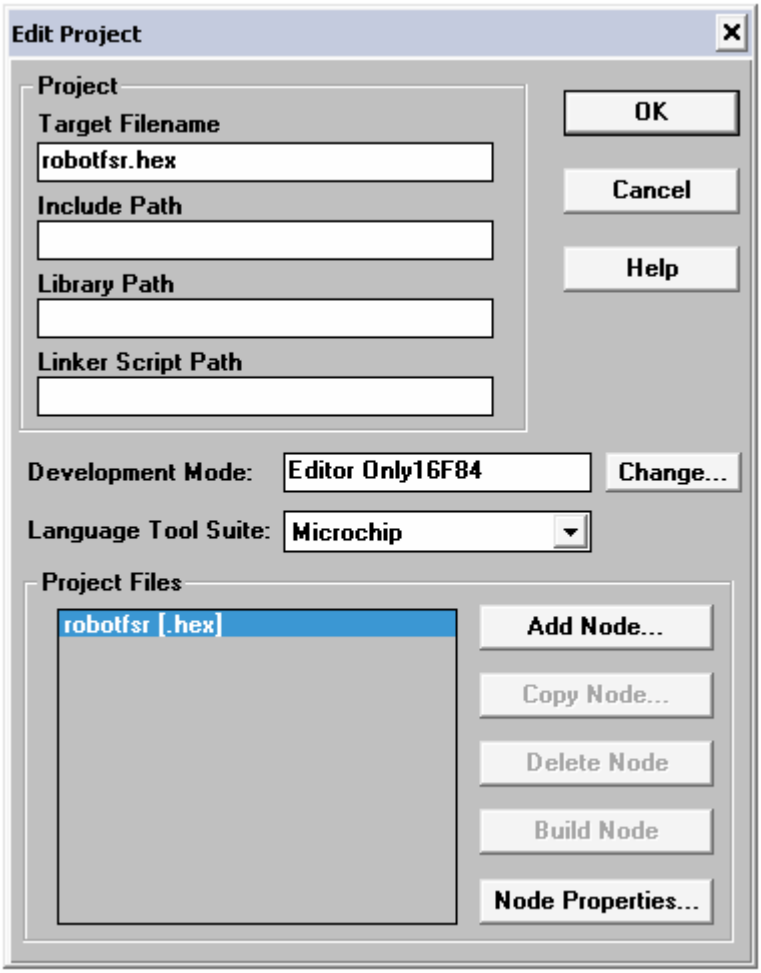
Dans le menu **project** cliquez sur **new project** et donnez un nom à votre future application (dans l'exemple essai.pjt) puis cliquez sur "OK"



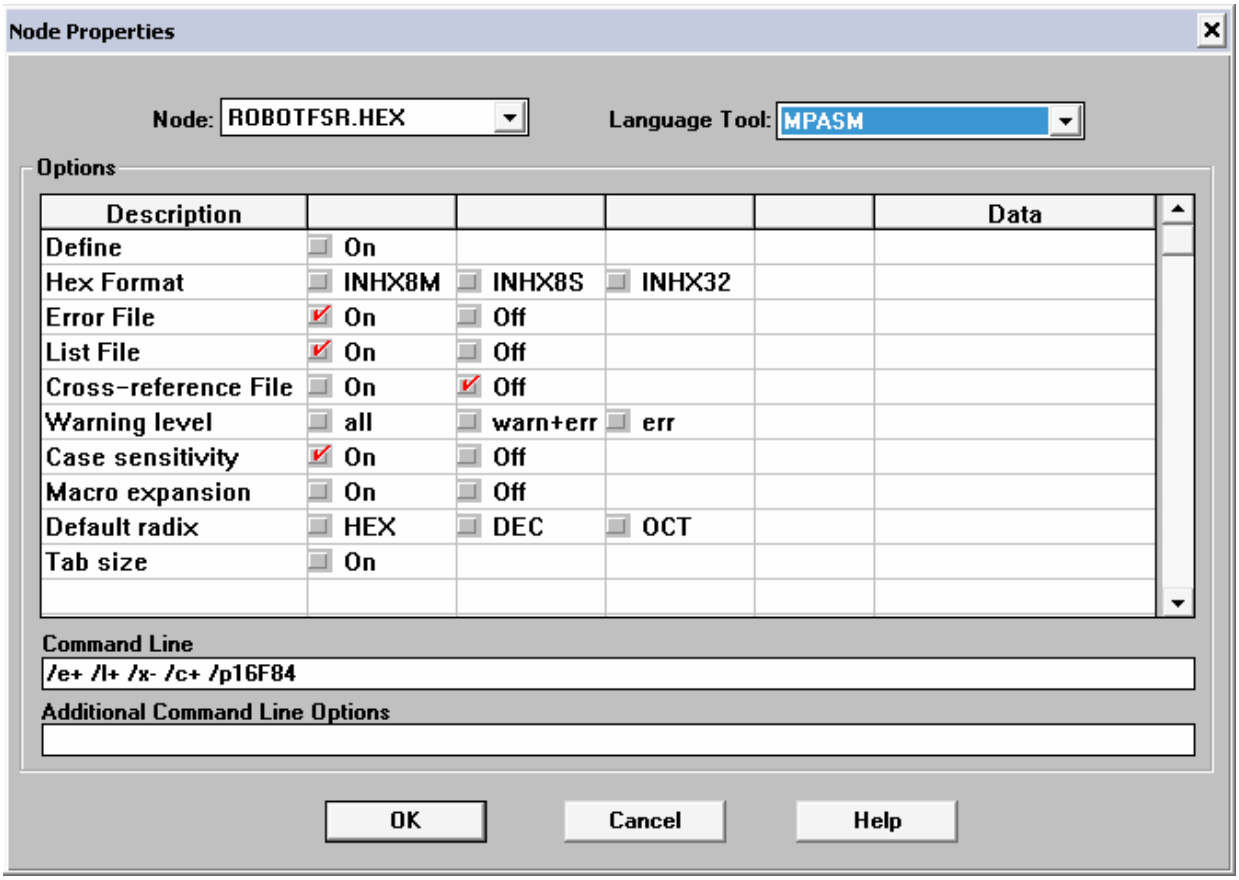
Une fois le nom donné cet écran apparaît, vous devez sélectionner 'éditeur' et le pic sur lequel vous travaillez dans l'exemple "Editor Only" et "PIC 16F84A" , pour changer de micro contrôleur cliquez sur le bouton Change... et choisissez votre option.



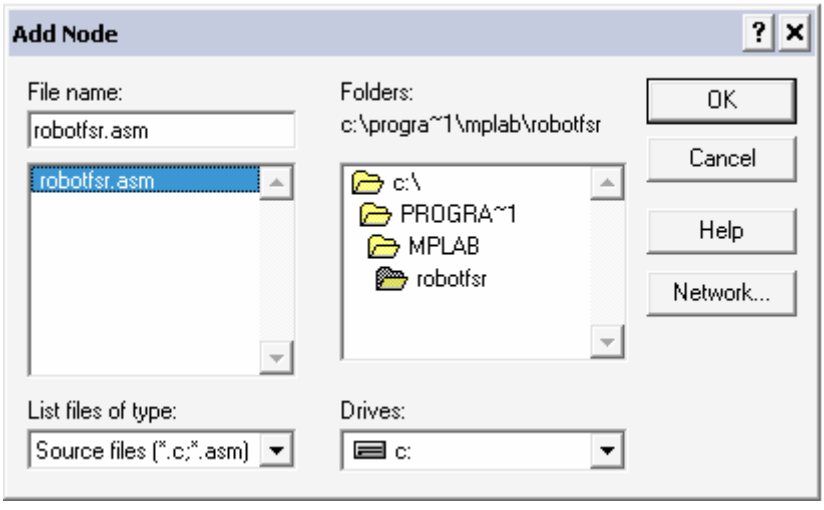
Cliquez ensuite sur le nom dans la fenêtre du bas (dans l'exemple essai.hex) puis cliquez sur "Node Properties"



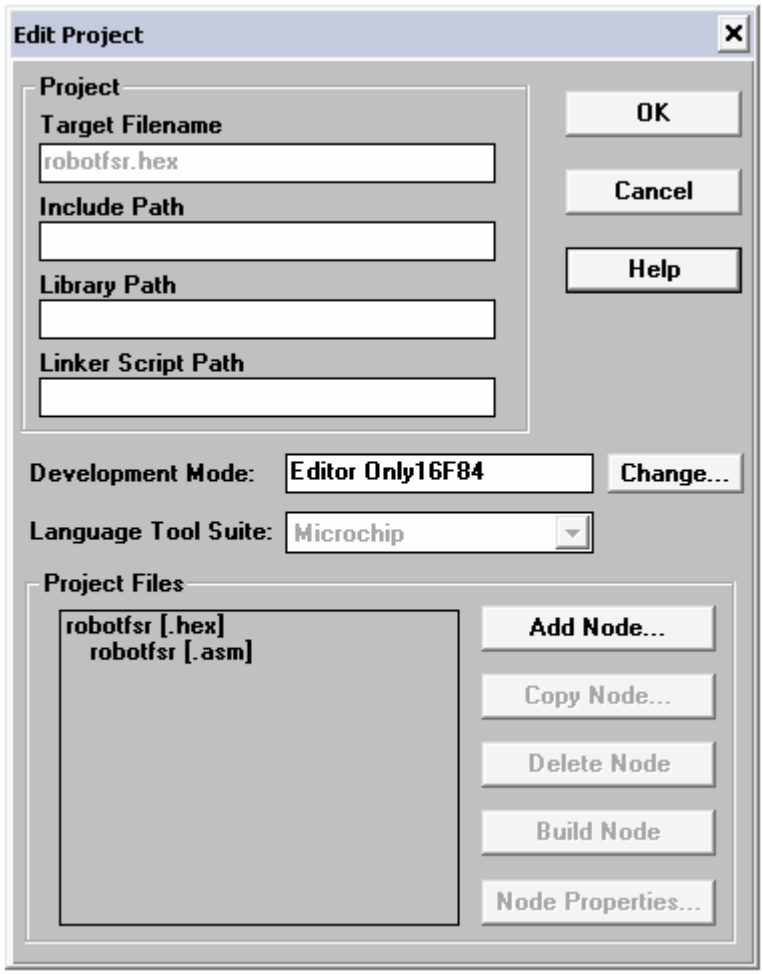
Cet écran apparaît sélectionnez les options indiquées ci-dessous et cliquez sur le bouton "OK"



En validant, vous revenez alors à l'écran précédent, cliquez alors sur le bouton "Add Node..." et indiquez le nom du fichier assembleur que vous allez créez (dans l'exemple 'essai.asm') puis validez avec "OK".

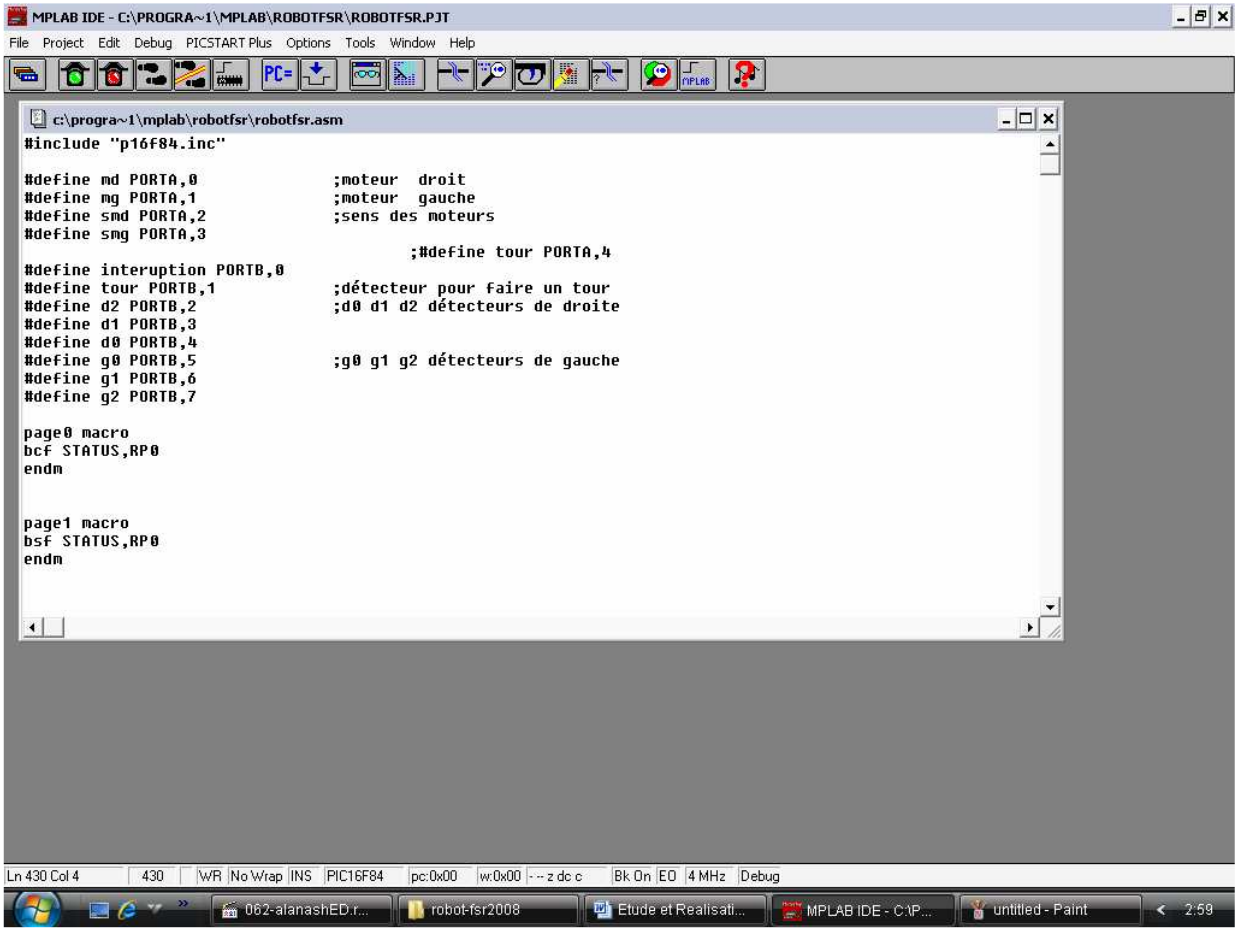


Vous devez alors avoir cet écran :

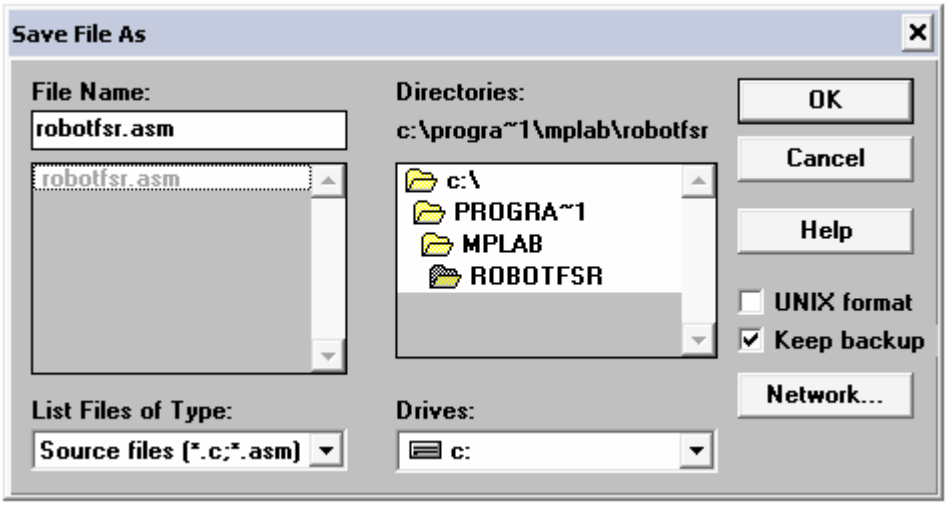


Cliquez sur le bouton "OK vous revenez alors au premier écran.

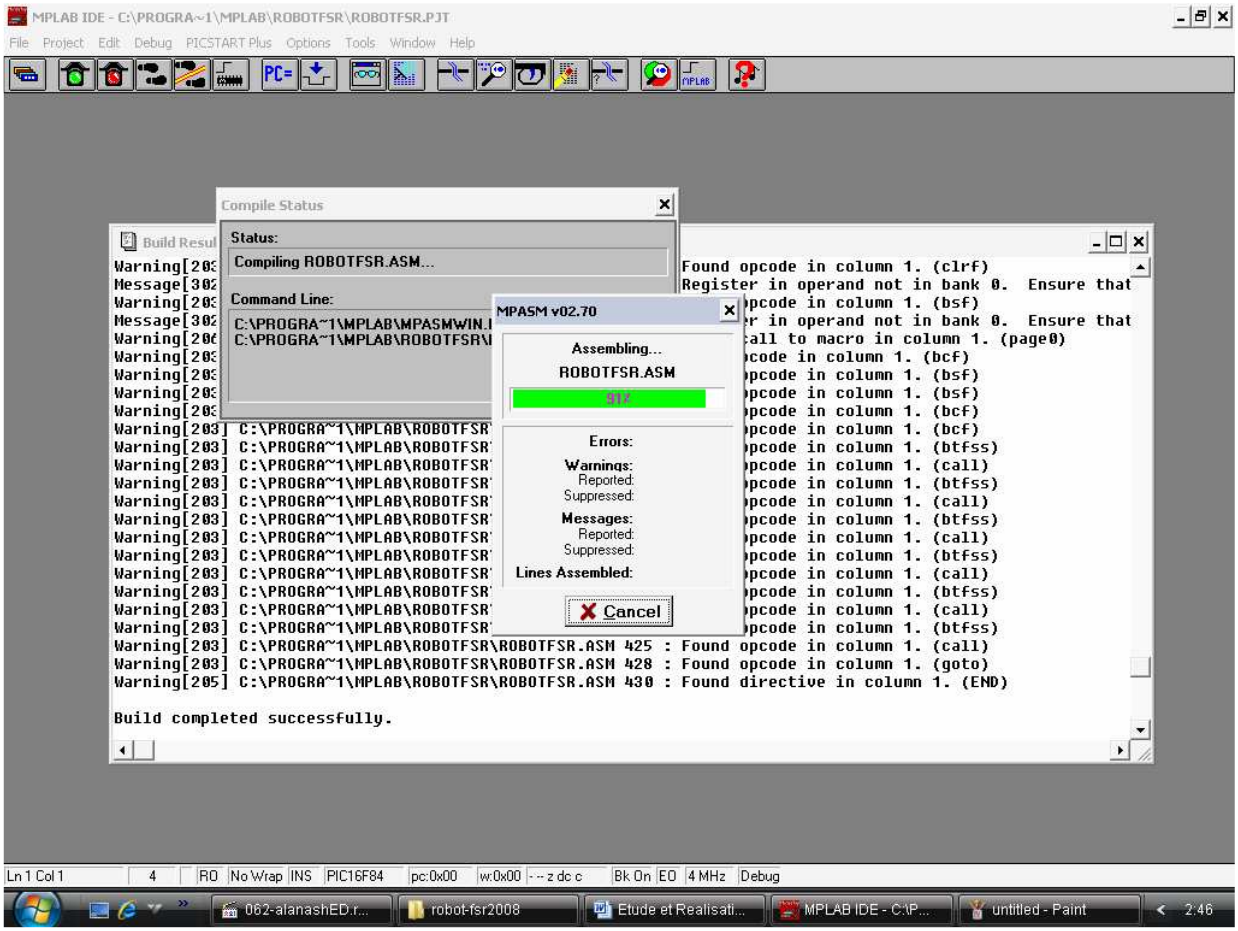
Cliquez sur "file" puis new, une fenêtre vous permet alors de commencer à tapez votre source en assembleur :



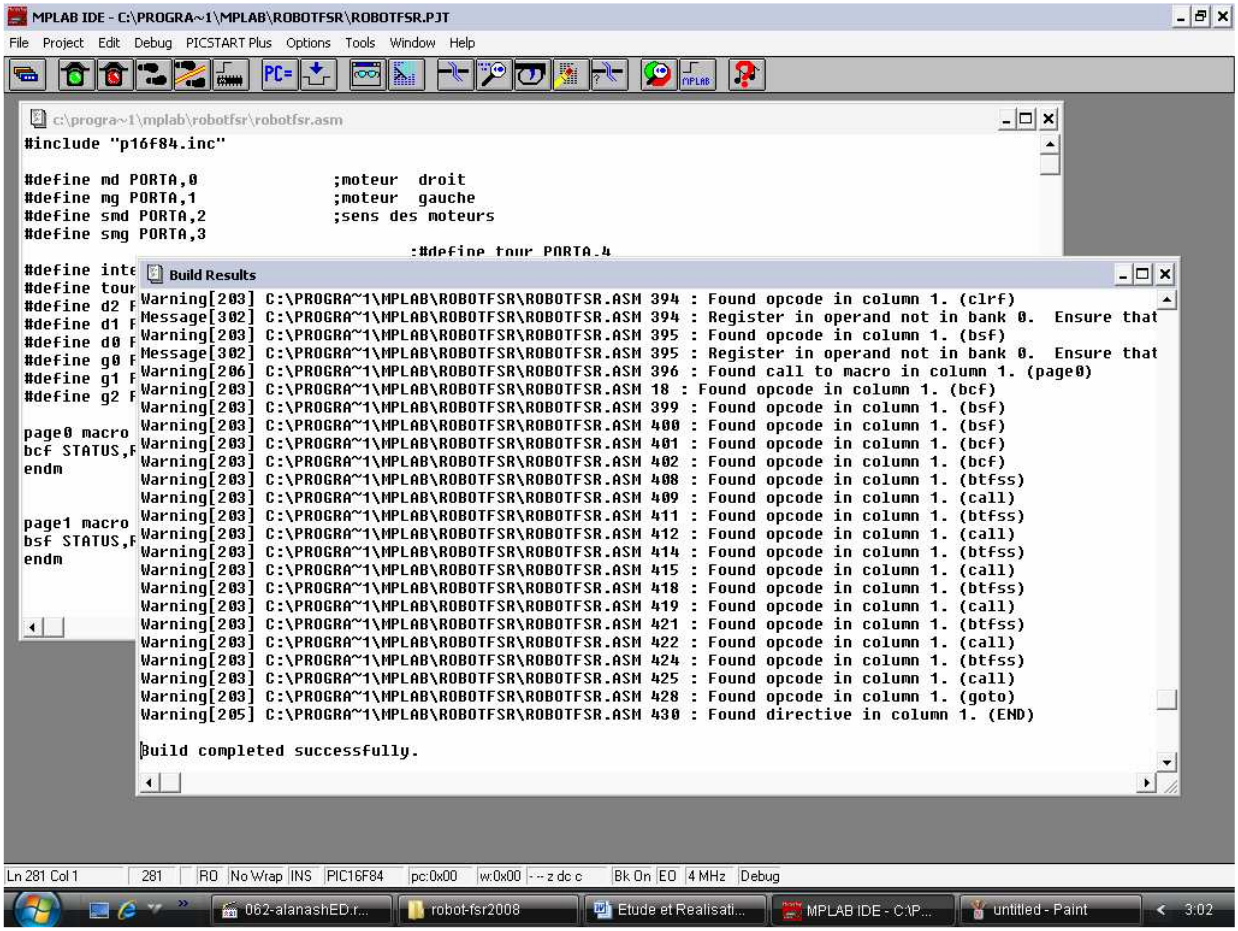
Une fois votre source tapez cliquez sur 'file' et 'save as' et entrez le nom de votre fichier source dans notre exemple se sera 'essai.asm'.



Il vous reste maintenant à compiler votre source afin de créer un fichier 'hex' que vous pourrez télécharger dans la mémoire du pic. Pour se faire cliquez dans le menu "Project' puis "Make project" vous obtenez alors l' cran suivant :



Si tout est bien c'est à dire qu'il n'y a aucune erreur de syntaxe vous obtenez l'écran suivant :



La compilation est terminée, le fichier "essai.hex" dans notre exemple est disponible dans le répertoire de MPLAB. Si vous possédez un logiciel de programmation tel que ICPROG.EXE par exemple cliquez sur "ouvrir" puis sélectionnez votre fichier "essai.hex" afin de le transférer vers la mémoire du PIC à programmer.

Dans le cas où il y a des erreurs le logiciel MPLAB indique la ligne à laquelle se trouve l'erreur de syntaxe.

ANNEXE 4 : Présentation du Logiciel ISIS

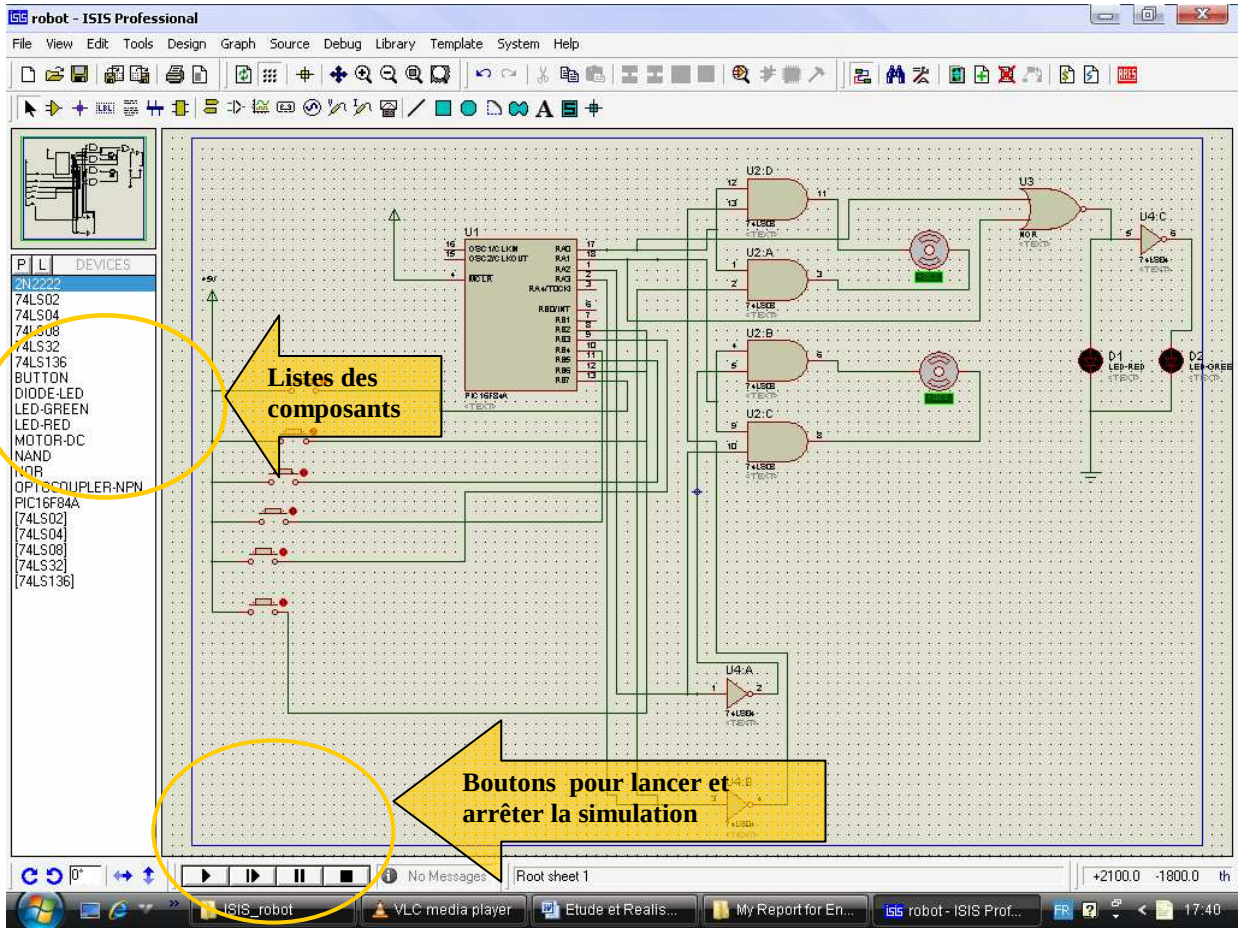
GÉNÉRALITÉS

Isis est un éditeur de schémas qui intègre un simulateur analogique, logique ou mixte. Toutes les opérations se passent dans cet environnement, aussi bien la configuration des différentes sources que le placement des sondes et le tracé des courbes.
Dans le cadre de ce projet, c'est la simulation numérique qui a sollicité notre intérêt.

La simulation analogique

Saisir un schéma

Pour saisir le schéma, il faut créer un nouveau projet puis placer les composants qui doivent être sélectionné à partir de la bibliothèque des composants :



Associer le programme au processeur

Dans le menu « source » sélectionnez la commande « Ajout/suppression fichiers source » puis sélectionnez le fichier ASM du programme à utiliser dans la simulation.
Puis choisissez « Outil de génération de code» pour générer un fichier HEX

Editez le PIC16F876 et ajoutez le fichier HEX dans les champs « programme file »
Appuyez sur le bouton jouer du « magnétoscope ». La barre des messages doit indiquer le temps écoulé depuis le lancement de l'animation

Afficher les fenêtres « Registers » « Source Code » « Data memory » par l'intermédiaire du menu « Debug » pour contrôler le fonctionnement du programme, détecter et corriger les erreurs

Le schéma à été réalisé en utilisant le logiciel ISIS ce dernier nous a permis de vérifier le fonctionnement du microcontrôleur à travers la simulation

ANNEXE 5 : Programme du robot (Code Source)

Code source.ASM

```
#include "p16f84.inc"

#define md PORTA,0          ;moteur  droit
#define mg PORTA,1          ;moteur  gauche
#define smd PORTA,2         ;sens des moteurs
#define smg PORTA,3
                            ;#define tour PORTA,4
#define interuption PORTB,0
#define tour PORTB,1        ;détecteur pour faire un tour
#define d2 PORTB,2          ;d0 d1 d2 détecteurs de droite
#define d1 PORTB,3
#define d0 PORTB,4
#define g0 PORTB,5          ;g0 g1 g2 détecteurs de gauche
#define g1 PORTB,6
#define g2 PORTB,7

page0 macro
bcf STATUS,RP0
endm


page1 macro
bsf STATUS,RP0
endm




cblock h'000c'
x1:1        ;tempo 00
x2:1
y1:1        ;tempo 10
y2:1
z1:1        ;tempo 11
z2:1
z3:1
t1:1        ;tempo 21
t2:1
t3:1
u1:1        ;tempo 22
```

```
u2:1
u3:1
v1:1       ;tempo 1 tour
v2:1
v3:1
w1:1       ;tempo 1/2 tour
w2:1
w3:1
endc

org h'0000'
goto debut

org h'0004'

routine:
                ;bcf INTCON,INTF  ; pour pouvoir quitter la routine

bcf md                ;arret des 2 moteurs
bcf mg

btfss tour
goto untour
bcf INTCON,INTF

arret:
btfss interuption     ;ne pas quitter la routine si RB0=1
goto quitter
goto arret

quitter:
bsf md                ;les 2 moteurs en marche
bsf mg
RETFIE

untour:
bcf smd                     ;changer le sens du md
nop
nop
bsf md                      ;les 2 moteurs en marche
bsf mg
call tempo1tour
bcf md                      ;arret des 2 moteurs
bcf mg
bsf smd                     ;changer le sens du md
nop
nop
bsf md                      ;les 2 moteurs en marche
bsf mg
bcf INTCON,INTF
RETFIE

g222:
btfsc g2
return
btfss g1
```

```
goto g21
goto g22

g2121:
btfss g1
goto g111
btfss g2
goto g222
return

g111:
btfsc g1
return
btfss g2
goto g21
btfss g0
goto g10
goto g11

g1010:
btfss g0
goto g000
btfss g1
goto g111
return

g000:
btfsc g0
return
btfss g1
goto g10
btfss d0
goto mil
goto g00

d000:
btfsc d0
return
btfss d1
goto d10
btfss g0
goto mil
goto d00

d1010:
btfss d0
goto d000
btfss d1
goto d111
return

d111:
btfsc d1
return
btfss d2
goto d21
```

```
btfss d0
goto d10
goto d11

d2121:
btfss d1
goto d111
btfss d2
goto d222
return

d222:
btfsc d2
return
btfss d1
goto d21
goto d22


mil:
bsf md                    ;les 2 moteurs en marche
bsf mg
etmil:
btfsc d0
RETURN
btfsc g0
RETURN
goto etmil


d00:
bcf md          ;md en arret
bsf mg
call tempo00
bsf md
bsf mg
call tempo00
goto d000

d10:
bcf md          ;md en arret
bsf mg
call tempo10
bsf md
bsf mg
call tempo10
goto d1010

d11:
bcf md
bsf mg
call tempo11
bsf md
bsf mg
call tempo10
goto d111
```

```
d21:
bcf md
bsf mg
call tempo21
bsf md
bsf mg
call tempo10
goto d2121

d22:
bcf md
bsf mg
call tempo22
bsf md
bsf mg
goto d222

g00:
bsf md
bcf mg
call tempo00
bsf md
bsf mg
call tempo00
goto g000

g10:
bsf md
bcf mg
call tempo10
bsf md
bsf mg
call tempo10
goto g1010

g11:
bsf md
bcf mg
call tempo11
bsf md
bsf mg
call tempo11
goto g111

g21:
bsf md
bcf mg
call tempo21
bsf md
bsf mg
call tempo21
goto g2121

g22:
bsf md
```

```
bcf mg
call tempo22
bsf md
bsf mg
goto g222


tempo1tour:
movlw 00
movwf v1
movlw 00
movwf v2
movlw 00
movwf v3
untr3:
untr2:
untr1:
decfsz v1,f
goto untr1
decfsz v2,f
goto untr2
decfsz v3,f
goto untr3
RETURN

tempomitour:
movlw 00
movwf w1
movlw 00
movwf w2
movlw 00
movwf w3
mitr3:
mitr2:
mitr1:
decfsz w1,f
goto mitr1
decfsz w2,f
goto mitr2
decfsz w3,f
goto mitr3
RETURN

tempo00:   ;66000
movlw 0b5
movwf x1
movlw 56
movwf x2
et002:
et001:
decfsz x1,f
goto et001
decfsz x2,f
goto et002
RETURN
```

```
tempo10:   ;100000
movlw 0dd
movwf y1
movlw 82
movwf y2
et102:
et101:
decfsz y1,f
goto et101
decfsz y2,f
goto et102
RETURN

tempo11:   ;133000
movlw 0b8
movwf z1
movlw 0ab
movwf z2
           ;movlw 00
           ;movwf z3
           ;et113:
et112:
et111:
decfsz z1,f
goto et111
decfsz z2,f
goto et112
           ;decfsz z3,f
           ;goto et113
RETURN

tempo21:   ;200000
movlw 0b9
movwf t1
movlw 04
movwf t2
movlw 02
movwf t3
et213:
et212:
et211:
decfsz t1,f
goto et211
decfsz t2,f
goto et212
decfsz t3,f
goto et213
RETURN

tempo22:   ;233000
movlw 95
movwf u1
movlw 2f
movwf u2
movlw 02
movwf u3
```

```
et223:
et222:
et221:
decfsz u1,f
goto et221
decfsz u2,f
goto et222
decfsz u3,f
goto et223
RETURN


debut:
bsf INTCON,GIE           ;autoriser les interuptions
bsf INTCON,INTE          ;autoriser les interuptions par la broche RB0
page1
bsf OPTION_REG,INTEDG     ;front montant
clrf TRISA               ; porta en sortie
bsf TRISA,4              ;portA4  EN ENTRE
page0


bsf smd
bsf smg
bcf md
bcf mg




etq:
btfss g2
call g222

btfss g1
call g111

btfss g0
call g000


btfss d2
call d222

btfss d1
call d111

btfss d0
call d000


goto etq

END
```

Code source.HEX

```
:02000000F228E4
:0800080005108510861C0F286D
:100010008B10061C0C2809280514851409000511ED
:1000200000000000005148514A32005108510051597
:1000300000000000051485148B100900861B0800C1
:10004000061F96289D28061F2828861F1E280800A0
:10005000061B0800861F9628861E88288F28861E65
:100060003428061F28280800861A0800061F88283A
:10007000061E58288128061A0800861D6628861E36
:1000800058285F28061E3B28861D47280800861929
:100090000800061D7428061E66286D28861D472840
:1000A000061D5328080006190800861D74287B28A1
:1000B00005148514061A0800861A08005A28051027
:1000C0008514BD2005148514BD203B28051085141A
:1000D000C62005148514C620422805108514CF209B
:1000E00005148514C62047280510851 4D82005144A
:1000F0008514C6204E2805108514E52005148514A6
:10010000532805148510BD2005148514BD203428FE
:10011000051485 10C62005148514C6202F28051443
:100120008510CF2005148514CF20282805148510AC
:10013000D82005148514D820232805148510E5201F
:1001400051485141E280030990000309A000030F4
:100150009B00990BA9289A0BA9289B0BA92808009A
:1001600000309C0000309D0000309E009C0BB628A3
:100170009D0BB6289E0BB6280800B5308C00563073
:100180008D008C0BC1288D0BC1280800DD308E003E
:1001900082308F008E0BCA288F0BCA280800B83017
:1001A0009000AB309100900BD328910BD32808001E
:1001B000B9309300043094000230950093 0BDE2890
:1001C000940BDE28950BDE280800953096002F3022
:1001D000970002309800960BEB28970BEB28980BB2
:1001E000EB2808008B170B168316011785010516DF
:1001F000831205158515051085 10861F1E20061F04
:10020002820861E3420061D5320861D4720061EEA
:040210003B20FD286A
:00000001FF
```

ANNEXE 6 : Projets proposés

1- Construction virtuelle de ce Projet en langage JAVA ou Flash;
2- Camera pour un robot suiveur de ligne blanche sur un fond bleu sombre avec une caméra noir et blanc en utilisant par exemple un PIC16F876A avec une camera CMOS;
3- **Robot intelligent :** manipulateur capable d'analyser les modifications de son environnement et de réagie en conséquence;
4- **Robot Youpi (Programmation par PIC) :** Le robot Youpi est un bras mécanique composé de 5 moteurs pas à pas de qualité industrielle de 200 pas/tour (précision de

1,8°) : base, épaule, coude, poignet, rotation main, pince. Il est raccordé au réseau électrique 220 V. Les transmissions entre les différents axes se font par le biais de pignons à denture droite ainsi que des courroies crantées. Sa vitesse maximale est de 40° par seconde pour une charge maximale de 400g. La pince permet de saisir des objets de moins de 80 mm de diamètre. Le serrage de la pince s'effectue par le biais d'un système vis écrou irréversible.

Le principe est de commander un bras robotisé (le robot YOUPI) à l'aide d'une carte à PIC.

Figure 69 : Le robot Youpi

5- **Le Robot Photovore :** nous allons prendre l'exemple du robot photovore (qui cherche la lumière). Donc le robot va se diriger vers la lumière en utilisant un capteur sensible à la lumière.

ANNEXE 7 : Budget

On a proposé pour ce projet un prix de 2000 DH, alors que la réalisation est faite avec 1146 DH. Ce qui donne une réduction de 50%. Donc de coté économique il est optimal.

RÉFÉRENCE

Ouvrages

- Cours d'électronique numérique de M. HAMRI 2007-2008
- Cours de microcontrôleurs "PIC16f84" de M. ZAHID 2007-2008
- Datasheet des composantes électroniques
- Datasheet de capteur OPB704 (Reflective Object Sensors: Type OPB703, OPB704, OPB705)
- PIC16F8X, document DS30430C, www.microchip.com
- PIC16F84a, document DS35007A, www.microchip.com
- Programmation des PIC, Première partie-PIC16F84-Révision 5, par BIGONOFF, http://www.abcelectronique.com/bigonoff/organisation.php?2654c

Sites web

- www.microchip.com
- www.roboticsplatform.com
- http://www.nsf.gov/eng/roboticsorg
- www.ieee-ras.org
- www.ifr.org
- www.euron.org
- http://tcremel.free.fr
- http://tcremel.free.fr/doc/bidouille_servo.pdf
- http://www.planete-sciences.org/robot/
- http://www.microchip.com/10/tools/picmicro/devenv/mplabi/index.htm
- http://www.admiroutes.asso.fr/larevue/2000/1/paradigm.htm
- http://louispayen.apinc.org/2002-2003/ressources-tpe.htm