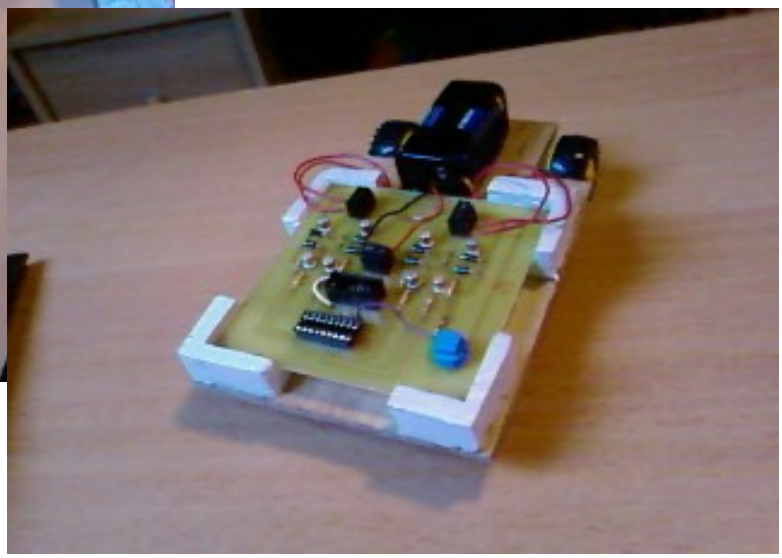
CHAUCHARD Léo
GAFFARD Pauline
FABRE Maxime

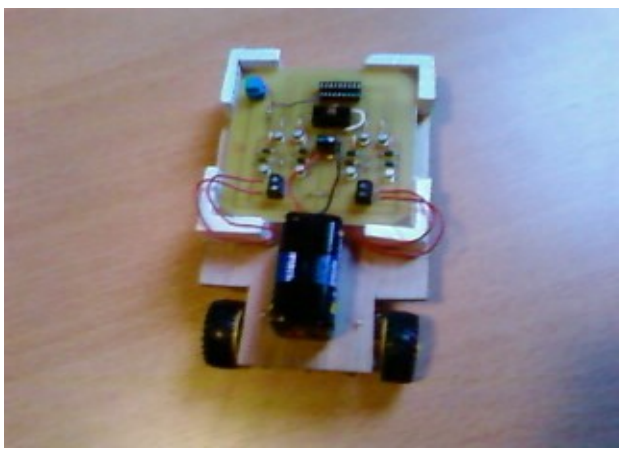
603
Groupe n°15
2008-2009

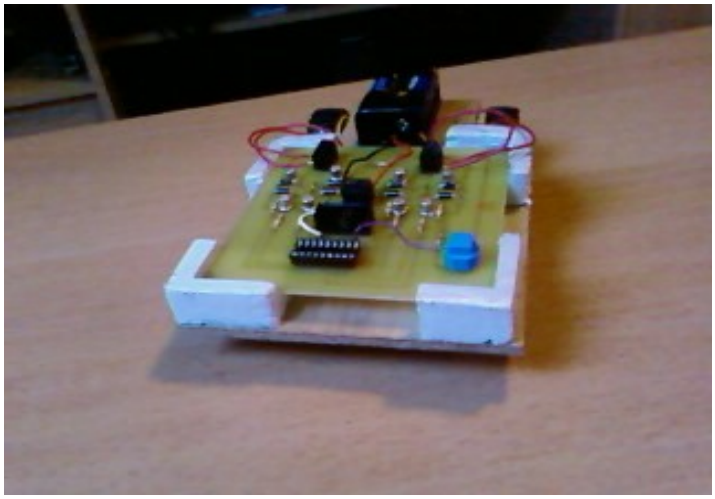


T.P.E

ROBOTIQUE

Création du robot *Léopomax*.





INTRODUCTION

Nous avons choisit le sujet de la robotique. En effet c'est un sujet totalement inconnu pour nous mais qui nous attire. Durant notre TPE nous allons essayer de construire un robot pour répondre à la problématique suivante :

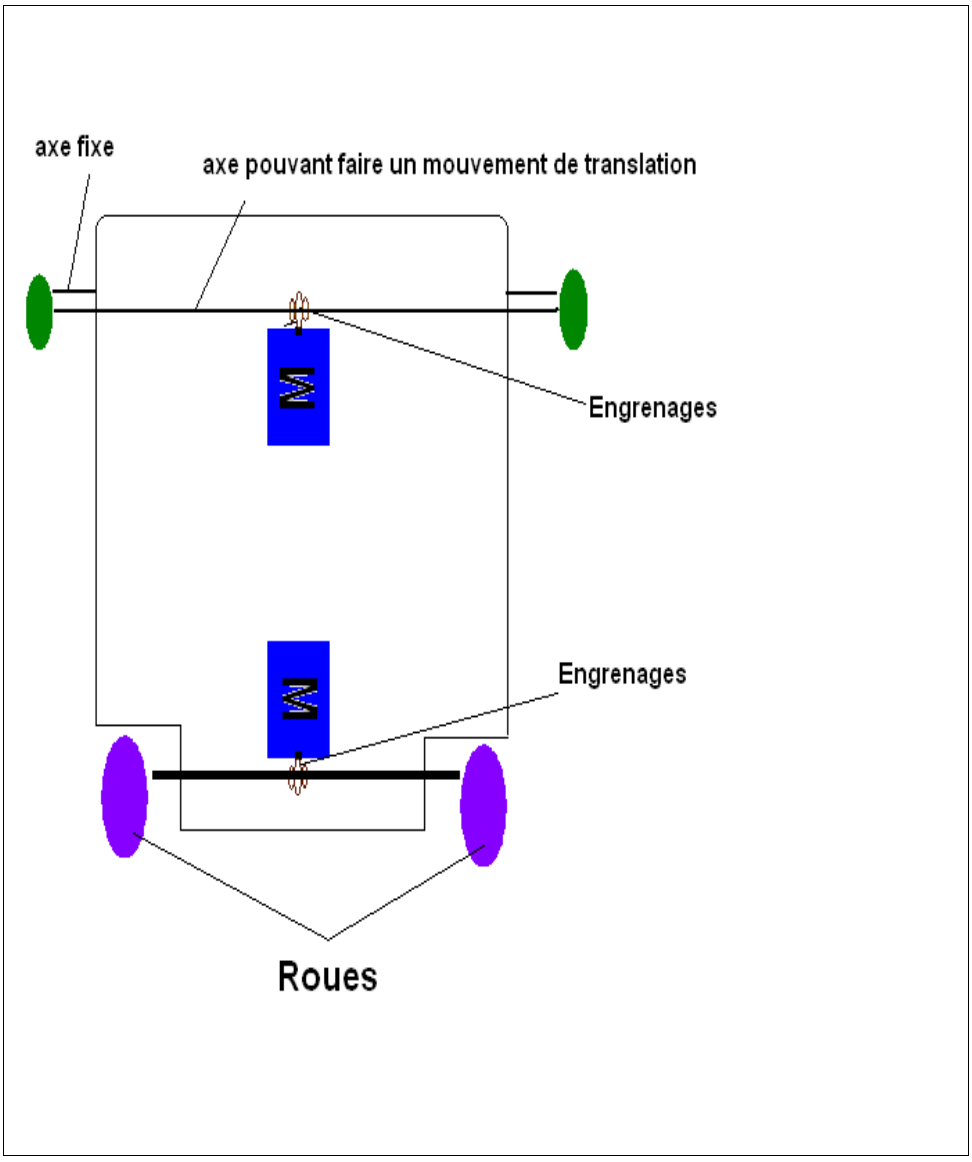
Comment intégrer l'automatisme dans les systèmes électronique ?

Notre réalisation se fera en trois parties à savoir une première partie mécanique, puis une seconde sur l'électronique pour enfin aborder la programmation.

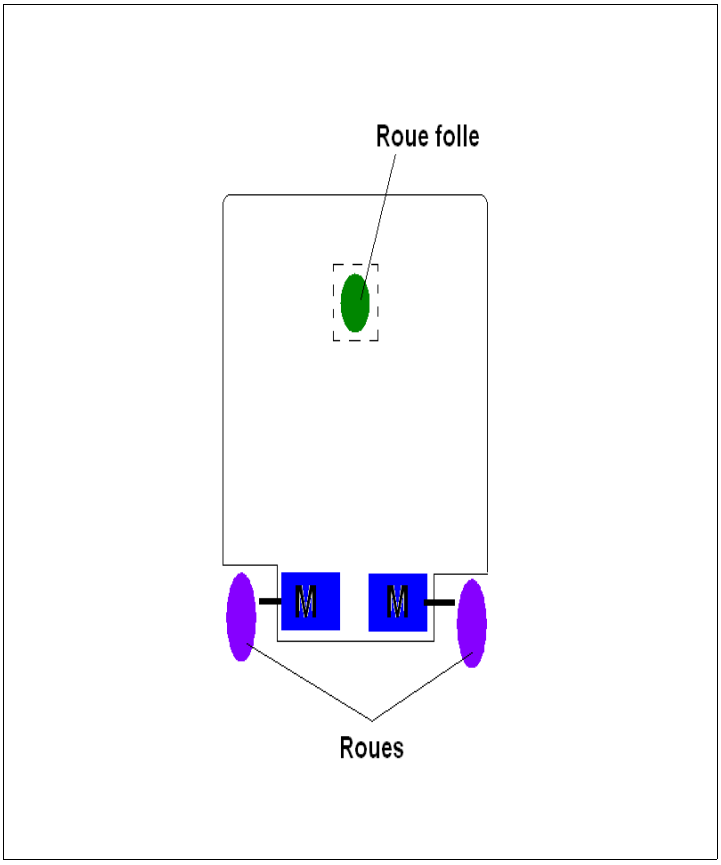
I. Le déplacement du robot ou comment la mécanique intervient dans l'automatisme.

A. Choix de nos moteurs et de l'alimentation.

Tout d'abord il nous faut déterminer le nombre de moteurs dont nous avons besoin et quelle sera leurs utilités. Nos contraintes sont un déplacement en avant et en arrière et un déplacement circulaire.
Deux choix se présentent à nous. Soit nous utilisons un moteur pour la propulsion et un autre pour la direction comme suit :



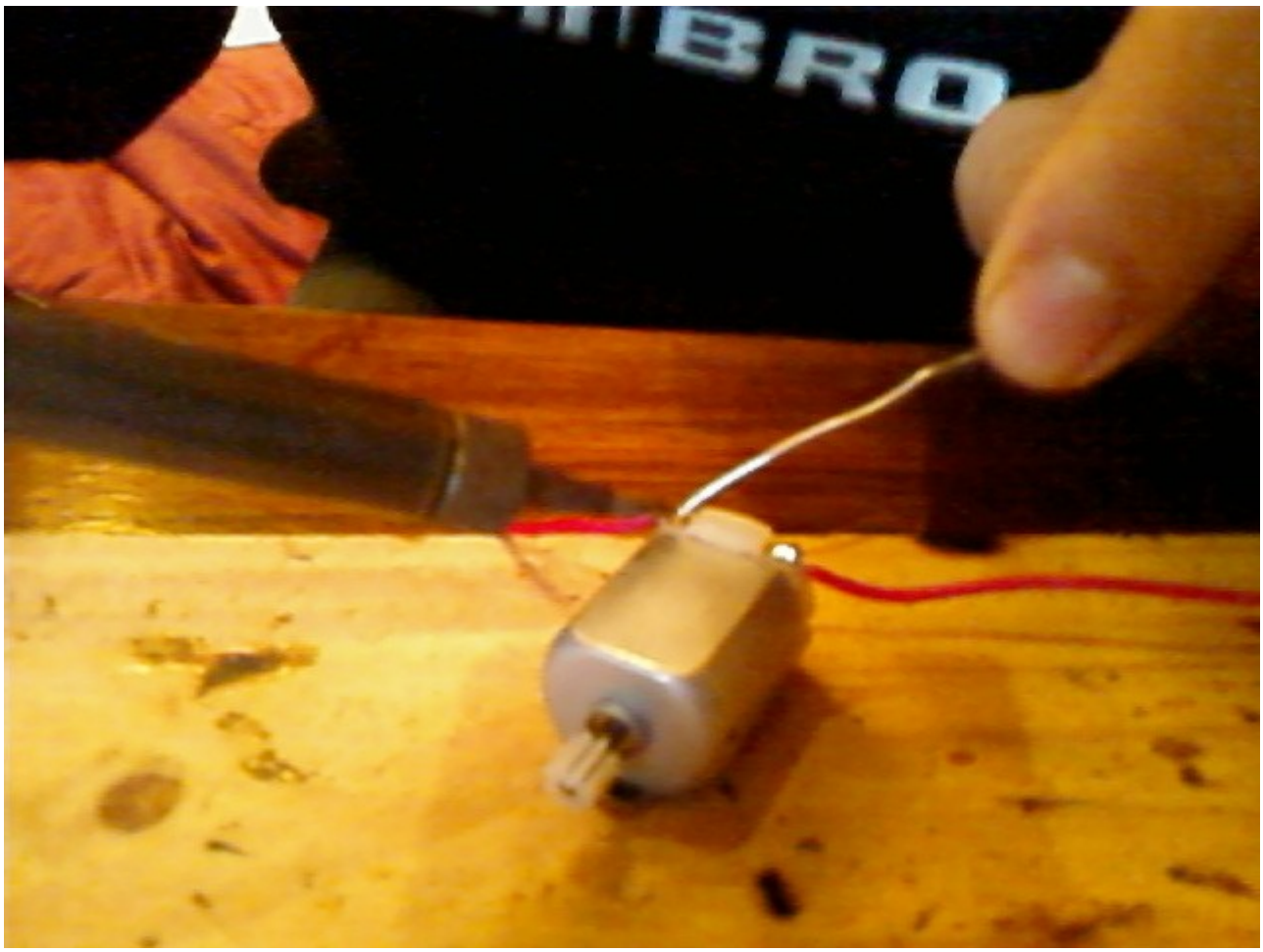
Soit nous utilisons les deux moteurs pour la propulsion, un pour chaque roue, et une roue folle qui permet de rester en équilibre.
En effet si on a un moteur pour chaque roue, si les vitesses de rotation de ces deux moteurs sont différentes alors le mouvement sera circulaire.



C'est sur cette alternative que va se porter notre choix.

Maintenant ceux sont les caractéristiques de ces moteurs qui vont nous intéresser. Ici se pose le problème du choix de l'alimentation. Il faut qu'elle soit simple, facile à trouver dans le commerce, légère, et il faut que notre robot soit libre de tout mouvement, c'est à dire qu'il ne doit pas être raccordé à une prise. Nous avons donc tout simplement choisit trois piles (LR6) de 1,5 Volts chacune raccordées par un coupleur. Ceci nous donne une alimentation de 4,5 Volts.

Ainsi nous avons trouvé un bloc double moteur fonctionnant entre 3 et 4,5 Volts et consommant environ 0,66 A avec un couple de 4,6 gcm, ce qui suffit amplement pour notre projet. De plus ce moteur a une vitesse de rotation comprise entre 7000 et 9000 tr/min, dans notre cas ce sera 9000 tr/min car nous avons pris la tension maximale. Bien évidemment cette vitesse est trop élevée c'est pourquoi il est fourni avec ceci un réducteur d'un rapport de 1/58. On a donc 9000*1/58 tr/min soit 155 tr/min à vide, ce qui est beaucoup plus raisonnable. Nous devons donc choisir nos roues.
A retenir qu'il nous a quand même fallu monter les moteurs avec le réducteur



B. Le choix des roues

Ce choix est important car la taille des roues influera beaucoup sur la vitesse et le couple. En effet plus les roues sont grandes, plus la vitesse du mouvement sera élevée mais moins le couple sera important. Notre premier souci étant le couple, en effet ce qui nous inquiète le plus est qu'il faut que le robot arrive à démarrer. Avec ce moteur nous pouvons trouver dans le commerce deux types de roues adaptable facilement. Il s'agira de roues de 36mm de diamètre et 16mm de largeur. Ainsi nous pouvons faire une première approximation de la vitesse à laquelle se déplacera notre robot. Nous avons vu précédemment que notre moteur a une vitesse de rotation à vide de 155 tr/min. Nous allons donc prendre pour nos calcul 130 tr/min à pleine charge. Ce qui donne :

V= Vm*P avec : -V : la vitesse moyenne du déplacement du robot en cm/s.
-Vm : la vitesse de rotation du moteur en tr/s.
-P : le périmètre des roues en cm.

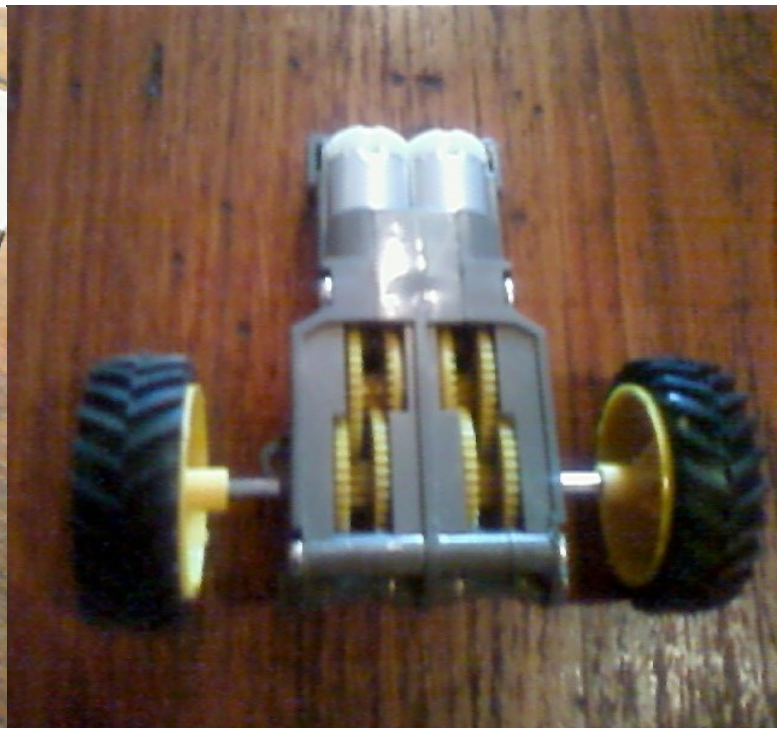
Vm = 130 tr/min
Vm = 130/60 tr/s
Vm = 2 tr/s

P = 2 * **π** *R

R = 36 mm
R = 0,36 cm

P = 2 * **π** * 0,36
P = 2,3 cm

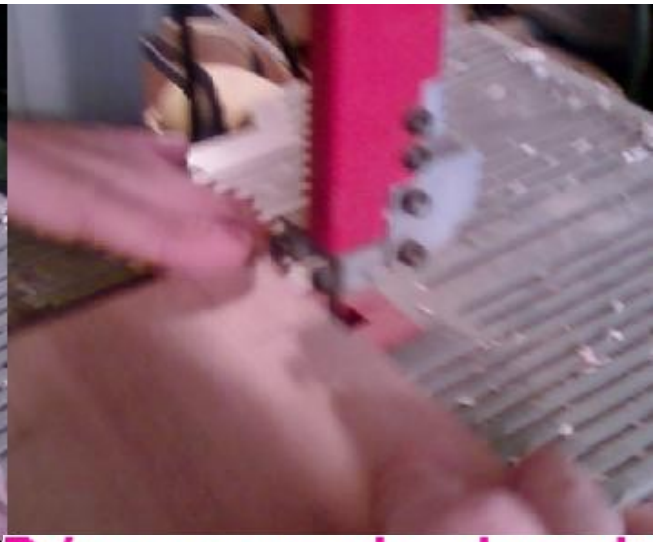
V = 2 * 2,3 cm/s
= 4,6 cm/s
Notre robot se déplacera donc à environ 4,6 cm/s.

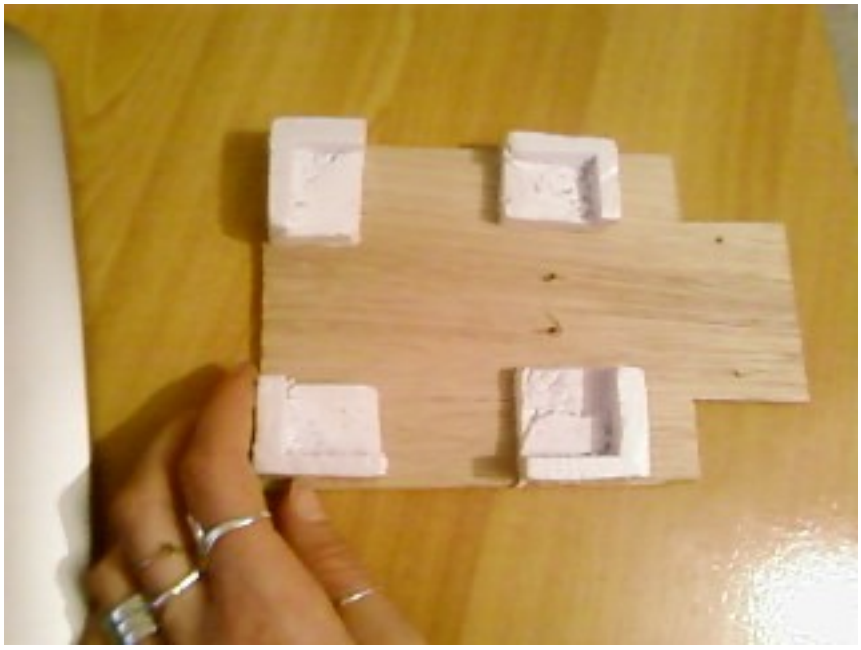


Ces roues conviennent donc bien pour notre robot. Nous pouvons alors choisir notre châssis.

C. Choix du châssis.

Ici les contraintes sont la masse et les dimensions. Nous devons choisir la matière. Pour des raisons de commodité nous avons opté pour du contre-plaqué, qu'on peut facilement usiné et qui est très léger. Nous avons donc pris en compte la place pour les roues ainsi que la place pour l'alimentation et le « futur » circuit imprimé que nous avons en réalité fait avant notre châssis.
Voici quelques traces de notre expérimentation expérimentalement expérimenté :





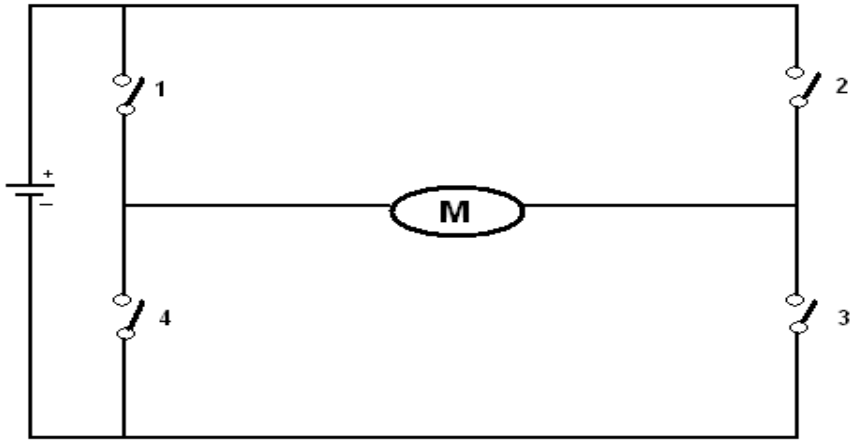
II.La conception de notre circuit électrique ou comment l'électronique sert l'automatisme.

A. Le choix des composants

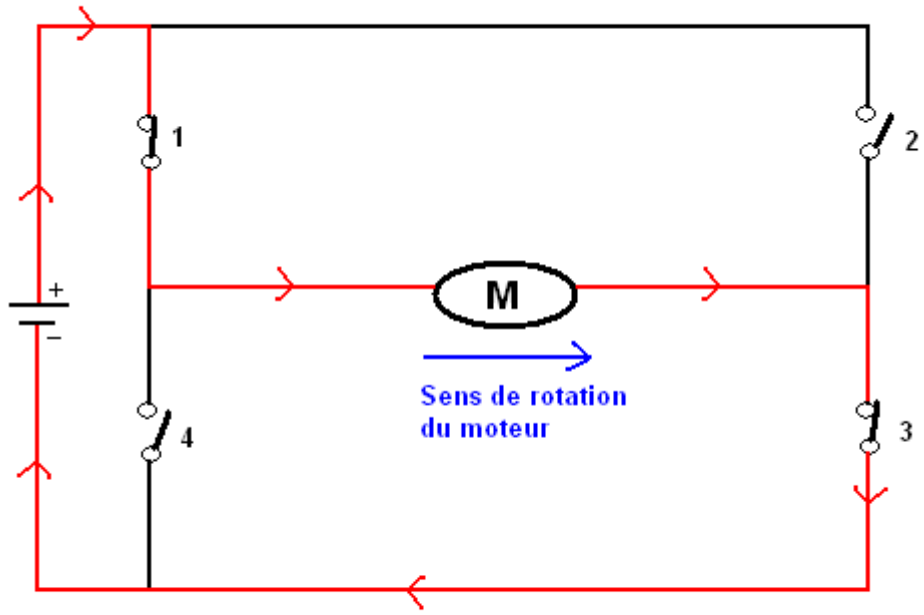
C'est probablement cette partie qui s'est révélée la plus difficile. En effet l'électronique présente un grand nombre de composants et impose de nombreuses contraintes. C'est pourquoi le choix d'un composant est assez compliqué.
Tout d'abord il nous a fallu trouver un système pour commander les moteurs, à savoir un pont en « H », permettant tous les mouvements voulus.

1. Les ponts en « H » pour commander les moteurs.

Premièrement un pont en « H » est un système permettant de contrôler la polarité de notre moteur. Il est vrai que si on change la polarité d'un moteur, celui-ci va tourner dans le sens inverse.
Voici un schéma montrant ceci :

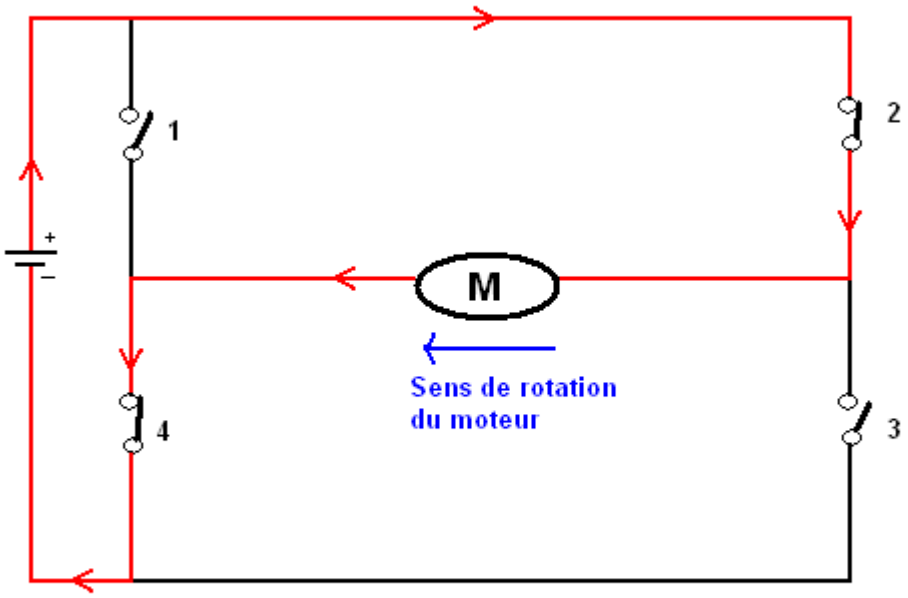


Ici on voit quatre interrupteurs qui vont fonctionner en couple. Effectivement si on ferme le 1 et le 3, on obtient le schéma suivant :



Dans ce cas le moteur tourne dans le sens des aiguilles d'une montre, le robot avance.

A l'inverse, si on ferme que les interrupteurs 2 et 4, on obtient le nouveau schéma que voici :



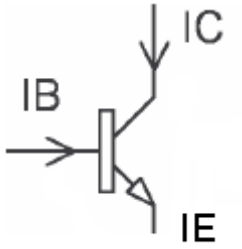
Ici le moteur tourne dans le sens inverse des aiguilles d'une montre, le robot recule.

2. Le choix des transistors et du PIC

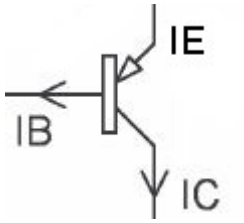
a)Les transistors

Cependant nous n'utiliserons pas des interrupteurs car ils ne sont pas automatisables, ils nécessitent l'intervention de l'homme. Pour pallier à cela nous allons utiliser des **transistors.**
Un transistor est un composant électronique utilisé la plupart du temps (dans notre cas) comme un interrupteur **commandé.** On distingue deux types de transistors, les NPN et les PNP.

Voici le schéma d'un transistor NPN :



Et celui d'un PNP :



Dans le cas des NPN, si on envoi du courant en IB (Intensité à la Base) le courant va circuler entre IE (Intensité à l'Emetteur) et IC (Intensité au Connecteur). On dit alors que le transistor est **saturé**. En revanche si le courant en IB est nul, le courant ne circulera pas entre IE et IC, on dit alors que la transistor est **bloqué**.

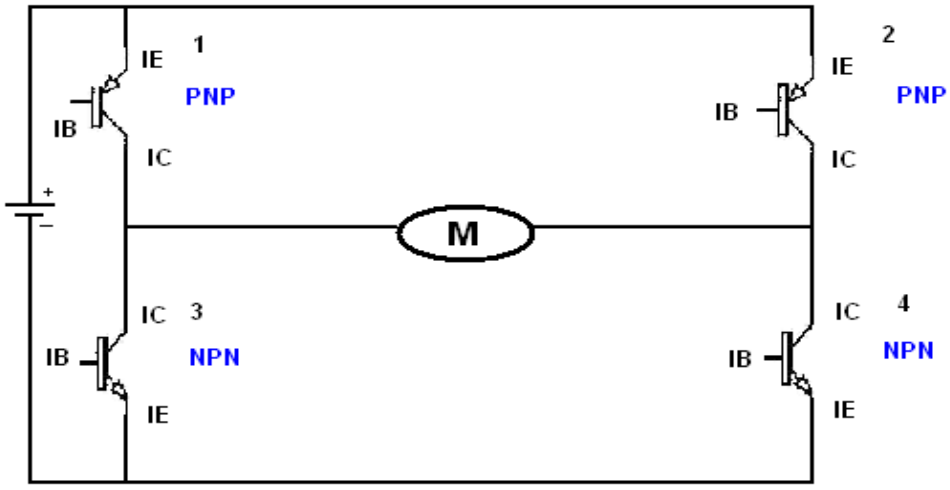
Dans le cas des PNP, pour **saturer** le transistor il faut que IB soit nul, et pour le **bloquer,** il faut envoyer du courant en IB.

Cependant il y a d'autres conditions. En effet un transistor est également un amplificateur de courant. C'est à dire qu'il existe un **coefficient multiplicateur** appelé HFE ou ß tel que :

ß > IC/IB

Ce coefficient ß s'appelle le **gain du transistor**, il est très variable selon les modèles et peut être compris entre 10 et 900.

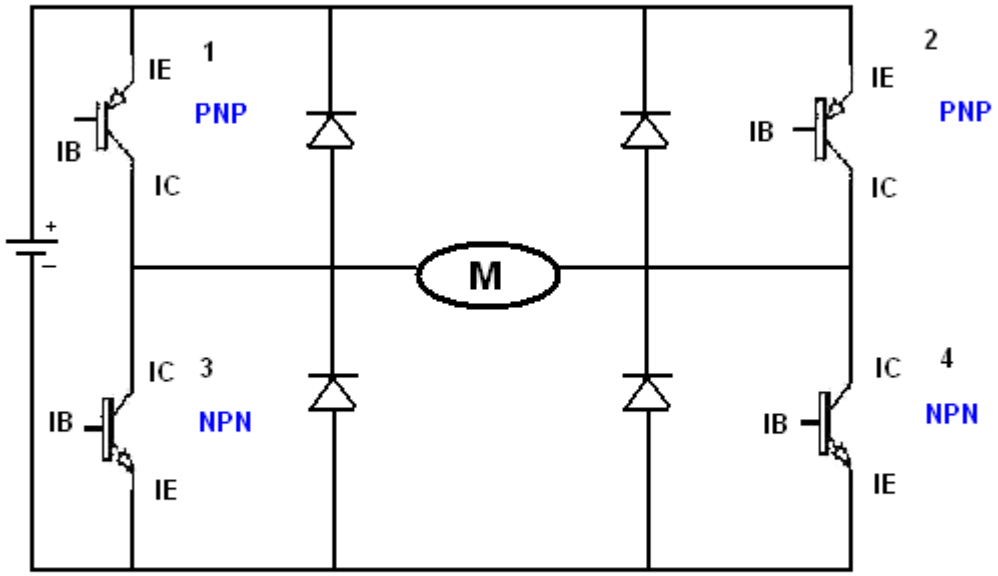
Voici maintenant ce que cela donne :



Ici on voit la disposition des transistors suivant leur type (NPN ou PNP). En effet celle-ci dépend de la borne ou patte de l'émetteur (sur le schéma : IE). Elle doit être directement reliée à l'alimentation.

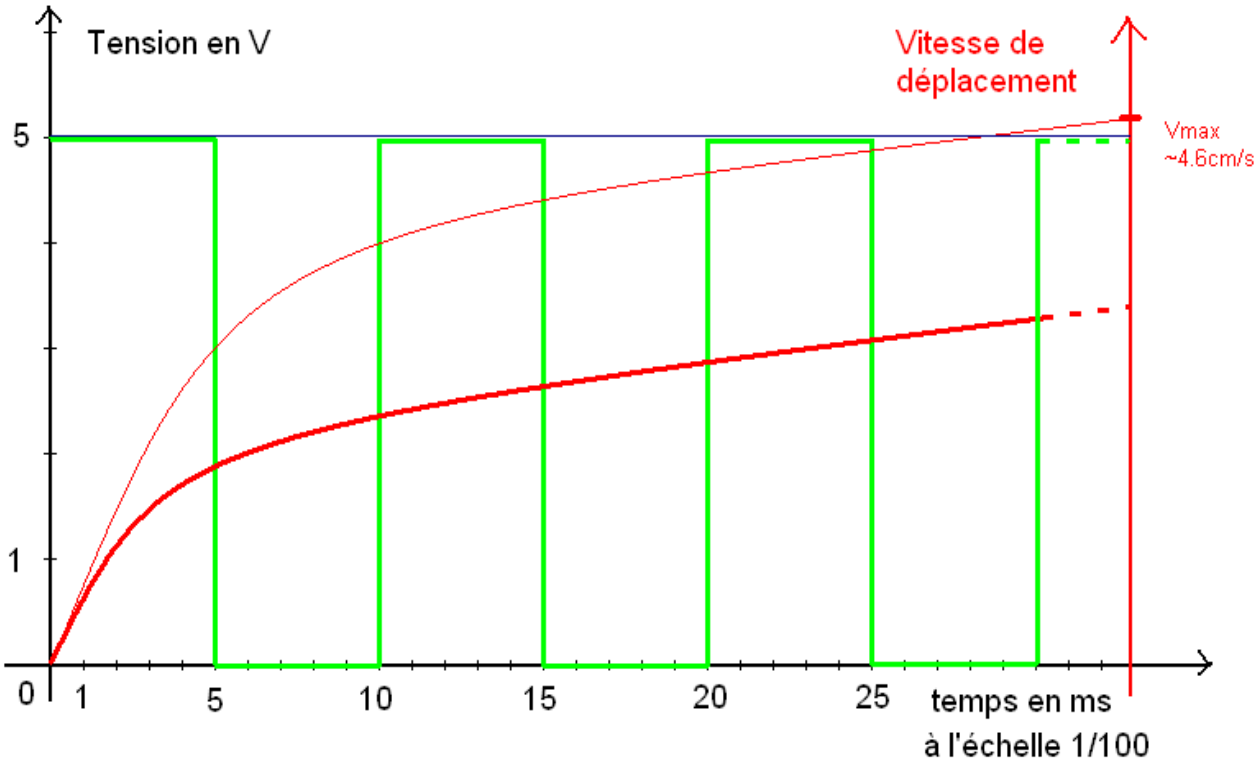
Encore persiste un problème. Si nous faisons fonctionner le moteur en saturant par exemple les transistors 1 et 4, et que nous les bloquons ensuite pour ouvrir le circuit et arrêter d'alimenter le moteur, le moteur lui ne va pas s'arrêter dessuite de tourner et par conséquent il va produire du courant. Si on ne fait rien ce courant risque de griller les transistors. C'est pour cela qu'on va devoir rajouter des **diodes**.

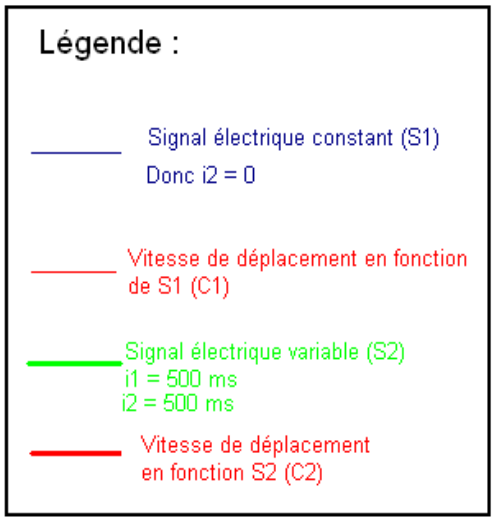
Une diode est un composant polarisé qui ne laisse passer le courant que dans un sens. Une seule contrainte pour son choix, il faut qu'elle supporte la tension. Dans notre cas celle-ci est très faible par conséquent on a pris un modèle courant supportant cette tension, le modèle 1N4001.
On a donc un pont en « H » comme
ceci :



En plus de cela, les transistors permettent de moduler les signaux électriques.
Par exemple si on sature les transistors 1 et 4 pendant i1 = 500ms puis qu'on les bloquent pendant i2 = 500ms et ainsi dessuite on obtient un signal presque sinusoïdal. En modifiant les intervalles de temps i1 et i2 on agit sur la vitesse de déplacement du robot.

Un exemple comparatif de deux signaux :





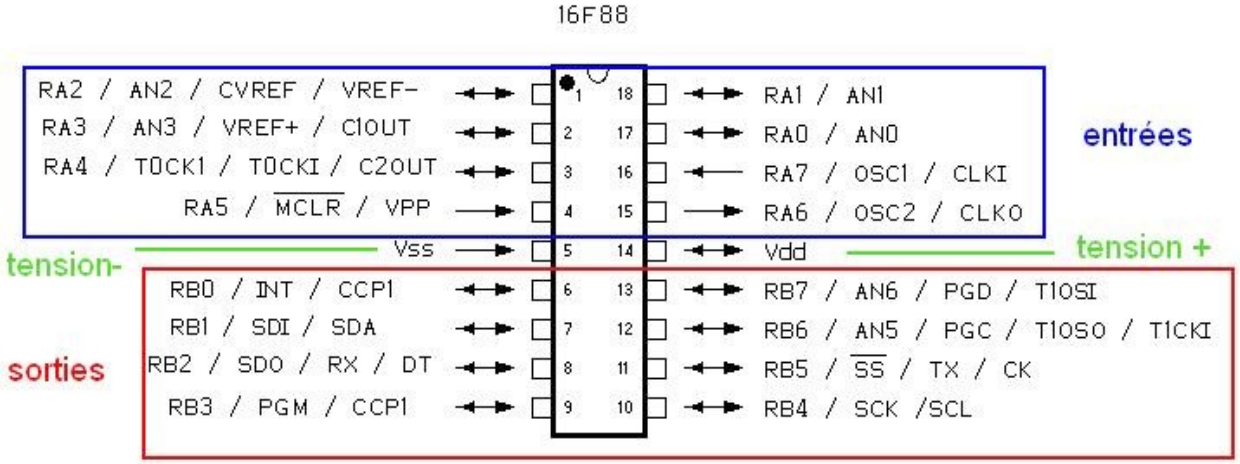
Sur ce schéma nous comparons deux signaux électriques et les vitesses résultantes. La courbe S1 représente un signal constant de 5V, avec i2 nul et i1 étant le temps écoulé. La courbe C1 représentative de la vitesse de déplacement du robot correspondante atteint la vitesse maximale et se stabilise à ce seuil.
La courbe S2 représente elle un signal variable, aux allures sinusoïdales, avec les intervalles de temps i1 et i2 égaux à 500ms.

La courbe C2 représente la vitesse de déplacement correspondante. On observe qu'elle n'atteint pas la vitesse maximale et qu'elle se stabilise à un certain seuil.
On voit donc que la vitesse dépend des signaux électriques modélisés avec les intervalles de temps i1 et i2. Voilà un des avantages des transistors. Il est vrai qu'il sera facile d'agir ainsi grâce au PIC qui est temporisé avec une horloge.

Pour récapituler, nous avons deux moteurs donc deux ponts en « H » et par conséquent huit transistors.
Pour commander ces transistors nous allons utiliser un PIC qui sera relié aux transistors au niveau de IB. Il nous faut donc le choisir avant de pouvoir déterminer les transistors qu'il nous faut.

b)Le PIC, un composant clé de l'automatisme.

Un PIC, aussi appelé **microcontrôleur** est un **circuit intégré** qui rassemble les éléments essentiels d'un ordinateur. A savoir un **processeur**, qui interprète les instructions d'un **programme**, **la mémoire morte**, qui contiendra le programme lui-même et **la mémoire vive**, dans laquelle sera stockée les données du programme.
De plus il contient des **entrées** et des **sorties** permettant le contrôle de la circulation des signaux électriques.
C'est donc grâce à ce composant que l'on va pouvoir réellement intégrer l'automatisme.
Nous avons dit plus haut qu'il nous fallait commander huit transistors. Nous aurons donc besoin de PIC avec huit sorties. Pour cela conviennent les modèles 16FXX. Il s'est avéré que le 16F88 facilite son intégration en comprenant la gestion de l'horloge, ce que nous ne développerons pas plus ici.
Voici à quoi ressemble le PIC :



Ce PIC délivre 25mA à chaque sortie. Pour nos transistors on a désormais IB = 25mA. De plus le PIC fonctionne à 5V ce qui convient pour notre alimentation de 4,5V.
De l'utilisation du PIC découle l'intégration d'un composant permettant sa programmation. Ce composant est le connecteur ICD 10 aussi appelé « PIC Flash connector ».
Nous reviendrons sur tout ceci dans la partie sur la programmation.

c)Détermination des transistors

Nous revenons donc aux choix de nos transistors. Nous connaissons depuis le choix du PIC l'intensité qui arrive en IB, à savoir 25mA.
Nous connaissons maintenant les intensités IC et IE depuis le choix du moteur. Celui-ci consomme 0,66A soit 660mA. Donc nous avons :

IB = 25mA
IC = 660mA
IE = 660mA

Nous savons également que :

ß > IC/IB
ß > 660/25
ß > 26,4

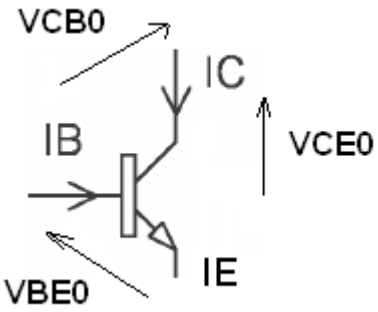
Avec toutes ces caractéristiques on peut trouver un transistor qui convient. Pour chercher nous avons utilisé le site www.farnell.fr qui comporte les informations techniques pour chaque transistor. Après plusieurs recherches nous avons trouvé ceux qui pouvaient convenir, le modèle 2N2222A pour les NPN. Et pour les PNP correspondant il s'agit du modèle 2N2907.
Voici comment nous avons procédé pour trouver les NPN, le modèle trouvé ayant des transistors PNP correspondant :

- Tout d'abord il nous faut regarder que le transistor peut supporter le courant qui le traverse. Il faut regarder les informations suivantes :

Absolute Maximum Ratings

Parameter	Symbol	Rating	Unit
Collector-Emitter Voltage	V_{CEO}	40	V
Collector-Base Voltage	V_{CBO}	75	
Emitter-Base Voltage	V_{EBO}	6.0	
Collector Current Continuous	I_C	800	mA
Power Dissipation at T_a = 25°C Derate above 25°C	P_D	500 2.28	mW mW/°C
Power Dissipation at T_C= 25°C Derate above 25°C	P_D	1.2 6.85	W mW/°C
Operating and Storage Junction Temperature Range	T_J, T_{stg}	-65 to +200	°C

Avant de continuer voici à quoi correspondent les tensions évoquées dans le tableau :



VCB0 est la tension qui traverse la base et le collecteur, VCE0 celle qui traverse l'émetteur et le collecteur et VBE0 la tension entre l'émetteur et la base.
Dans notre cas nous savons qu'aucune de ces tensions ne dépassera 4,5V(Tension de notre alimentation). Donc d'après le tableau on voit que le transistor supporte notre tension.
Maintenant se pose alors la même question pour l'intensité. Là il faut regarder l'intensité au collecteur (IC). Sur le tableau on voit que le transistor peut supporter une intensité continue de 800mA en IC. Il n'y a donc pas de problèmes pour nous, car notre IC vaut 660mA (consommation de notre moteur).

- Ensuite il nous faut regarder si le gain de courant (hFE ou ß) correspond à celui dont nous avons besoin, à savoir environ 26,4.

Voici les informations à regarder :

Electrical Characteristics (T_a = 25°C unless otherwise specified)

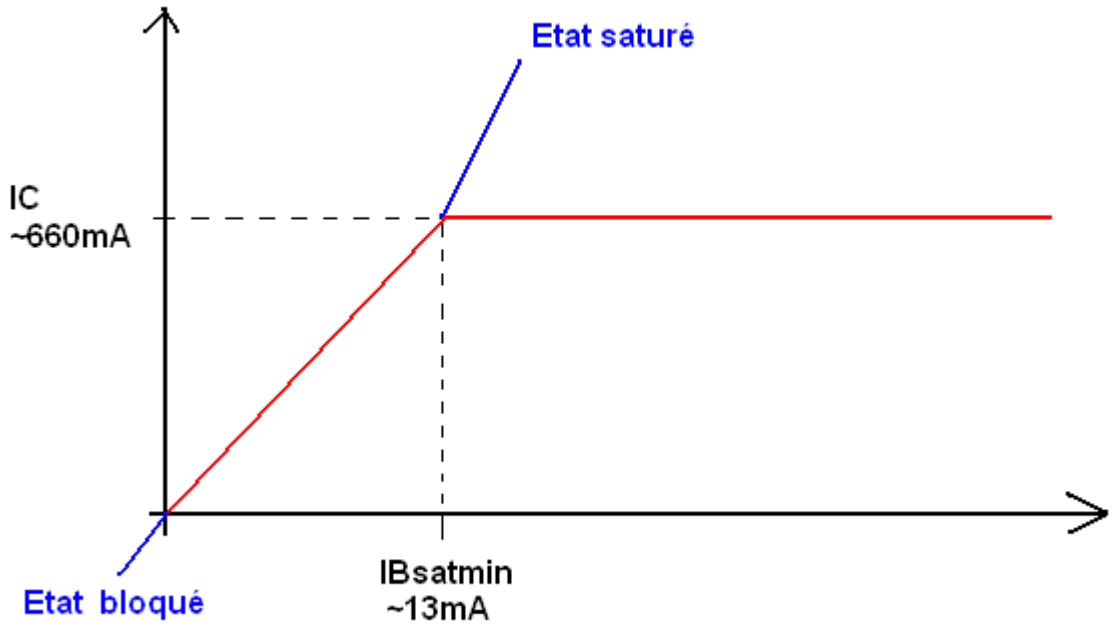
Parameter	Symbol	Test Condition	Rating	Unit
DC Current Gain	h_{FE} aussi appelé ß	I_C = 0.1mA, V_{CE} = 10V I_C = 1mA, V_{CE} = 10V I_C = 10mA, V_{CE} = 10V T_a = 55°C I_C = 10mA, V_{CE} = 10V I_C = 150mA, V_{CE} = 10V I_C = 150mA, V_{CE} = 1V I_C = 500mA, V_{CE} = 10V	>35 >50 >75 >35 100-300 >50 >40	-

On observe que hFE varie en fonction de IC. Pour IC égal à 500 mA on a hFE supérieur à 40. Donc ce sera convenable pour nous puisque nous avons IC égal à 660 mA, par conséquent hFE sera encore légèrement supérieur, donc on va prendre 50 qui est supérieur à notre 26,4.

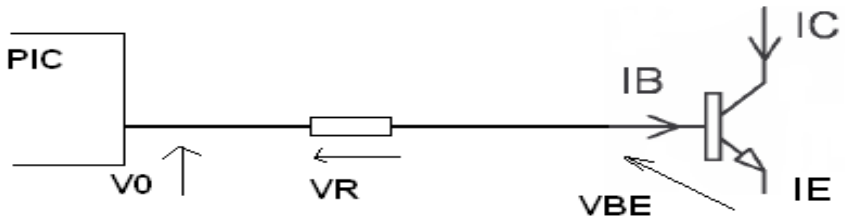
Pour saturer un transistor il faut que l'intensité à la base satisfasse la relation suivante :

IB > IC/ß

Donc dans notre cas :
Avec
IC = 660 mA
ß = 50
On a
IB > 660/50
IB > 13,2 mA
On appellera la valeur 13,2 IBsatmin. Voici un schéma montrant l'état de saturation du transistor en fonction de IB et IC. Dans notre cas IC sera constant à valeur 660mA.



On remarque bien que le transistor est saturé qu'au moment où IB et supérieur ou égal à la valeur IBsatmin.
De ce fait, pour être sur de bien saturer notre transistor on va prendre IB égal à 15 mA.
Seulement nous avons un PIC qui délivre 25mA. Il nous faut donc choisir des résistances qui nous ferons passer d'une intensité de 25mA à 15mA.
Pour trouver la valeur de ces résistances il nous faut également connaître la tension qui la traverse. Nous avons la disposition suivante :



Ici on va évoquer le fait qu'un transistor engendre une légère perte de courant. Celle-ci est propre à chaque transistor bien que avoisinant 0,7Volts. Dans la documentation du composant on trouve cette information ici :

Electrical Characteristics (T_a = 25°C unless specified otherwise)

Description	Symbol	Test Condition	Value		Unit
			Minimum	Maximum	
Collector Emitter Breakdown Voltage	BV_{CEO}	I_C = 10mA, I_B = 0	30	-	V
Collector Base Breakdown Voltage	BV_{CBO}	I_C = 10µA, I_E = 0	60	-	
Emitter Base Breakdown Voltage	V_{EBOf}	I_E = 10µA, I_C = 0	5	-	
Collector Leakage Current	I_{CBO}	V_{CB} = 50V, I_E = 0 V_{CB} = 50V, I_E = 0 T_a = 150°C	-	10 10	nA µA
Collector Emitter Saturation Voltage	$*V_{CE\ (Sat)}$	I_C = 150mA, I_B = 15mA I_C = 500mA, I_B = 50mA	-	0.4 1.6	V
Base Emitter Saturation Voltage	$*V_{BE\ (Sat)}$	I_C = 150mA, I_B = 15mA I_C = 500mA, I_B = 50mA	0.6	1.3 2.6	

Pour ce modèle de transistor, VBE sera égal à 0,6V lorsqu'il sera en état de saturation.
Avec VR la tension qui traverse la résistance et V0 la tension à la sortie du PIC.
On va donc utiliser la loi des mailles pour calculer VR.
On a la relation suivante :

V0 – VR – VBE = 0
VR = V0 – VBE
VR = 4,5 – 0,6
VR ~= 4

Maintenant on peut utiliser la loi d'ohm :

R = U / I

Avec

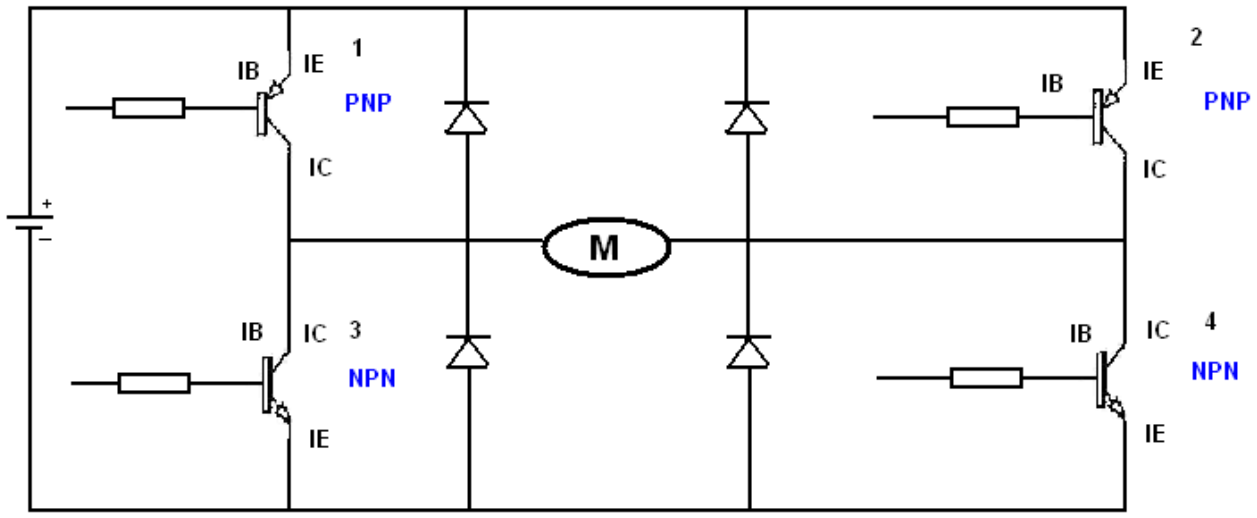
U = VR
U = 4

I = IB
I = 0,015 A

D'où

R = 4 / 0,015
R = 270 ohm

Nous avons maintenant les transistors et les résistances qu'il nous faut.
Nous pouvons donc faire une nouvelle esquisse du schéma électrique de nos pont H :



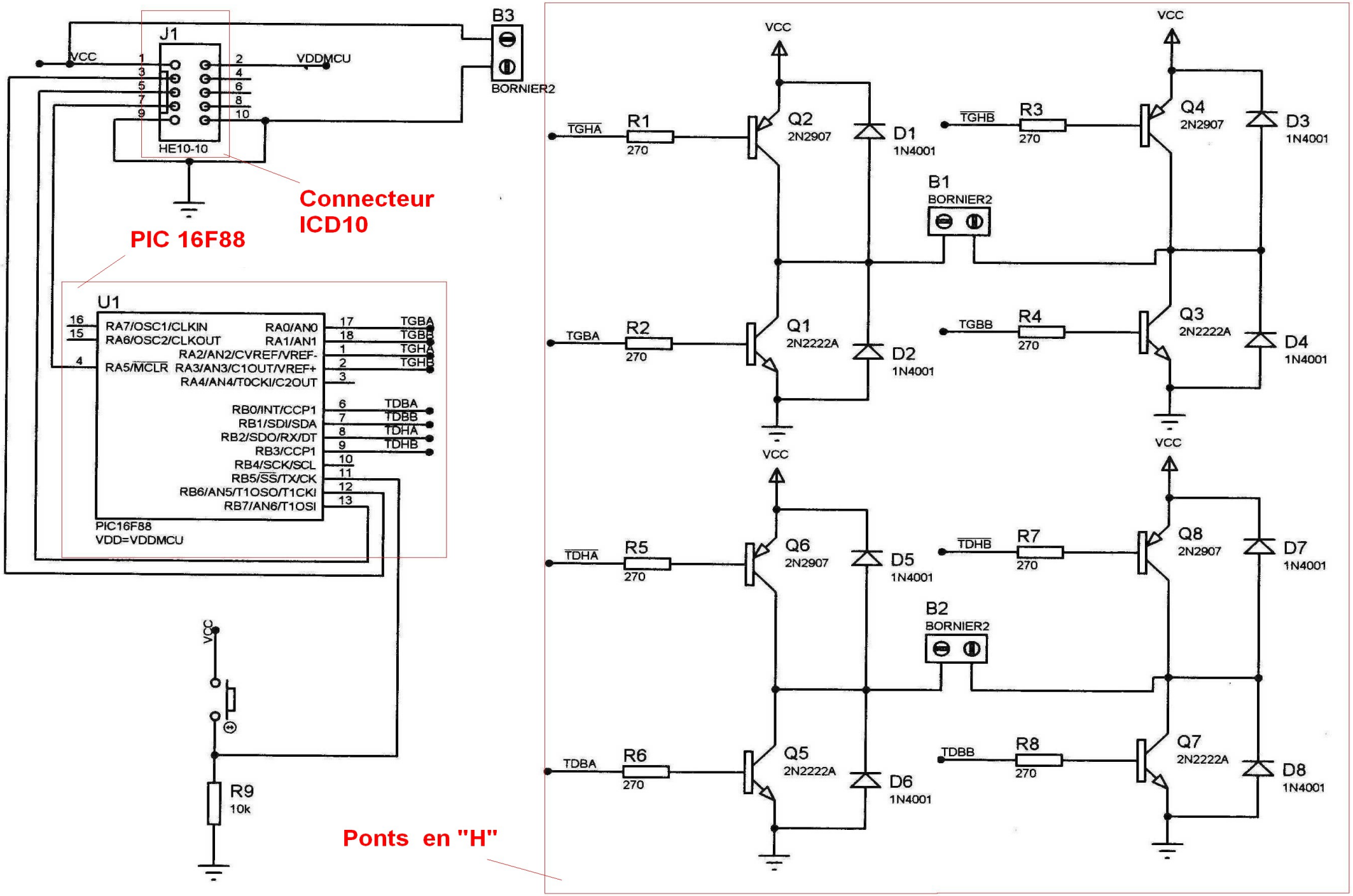
Ces deux vérifications suffisent pour choisir les transistors.

Il reste un petit détail à régler, en effet il nous faut mettre un bouton poussoir que l'on va brancher sur une entrée du pic ainsi on pourra vérifier avec le programme si un utilisateur active le bouton. De plus en parallèle avec le bouton poussoir il nous a fallu mettre une résistance permettant d'éviter tout parasite.

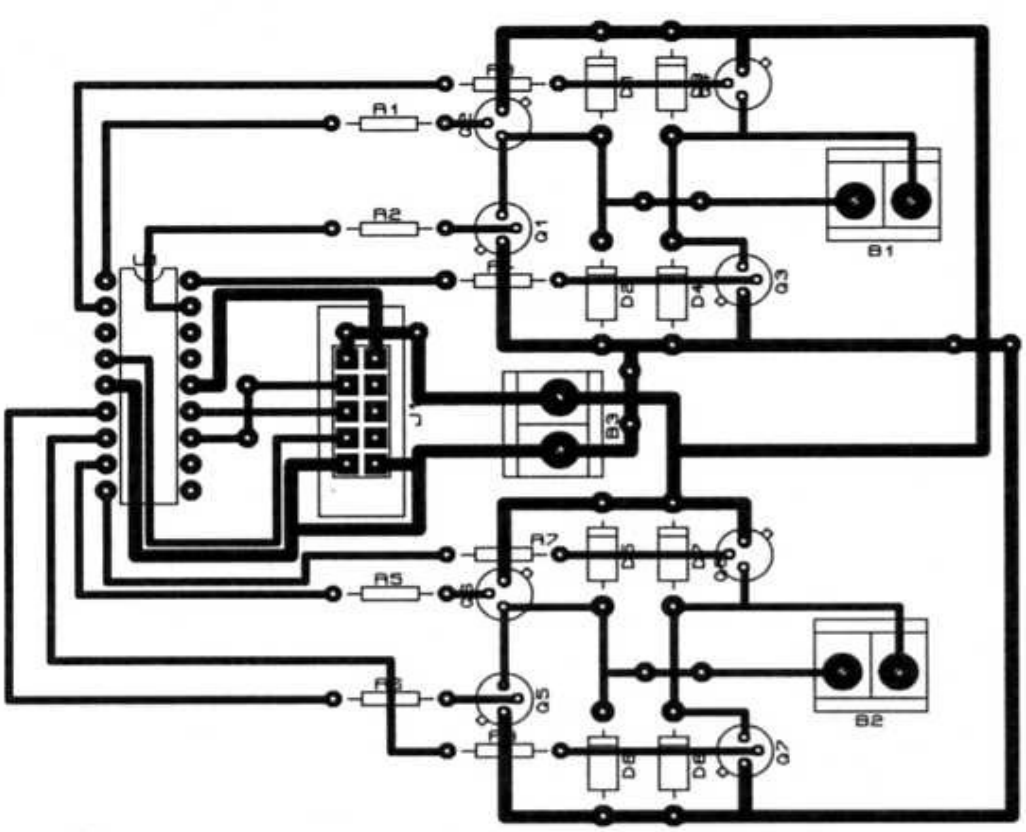
B. La conception du circuit

1. Conception du schéma électrique et routage.

Avant de fabriquer notre circuit imprimé il faut bien évidemment le concevoir. Pour cela on a utilisé le logiciel PROTEUS qui permet en premier lieu de faire le schéma électrique complet de notre circuit, et ensuite de faire le **routage**, c'est-à-dire dessiner les pistes et placer les composants au bon endroit sur le circuit. Une fois ceci fait on aura ce qu'on appelle le **typon** du circuit.
Nous avons réalisé le schéma électrique avec le module ISIS de PROTEUS :

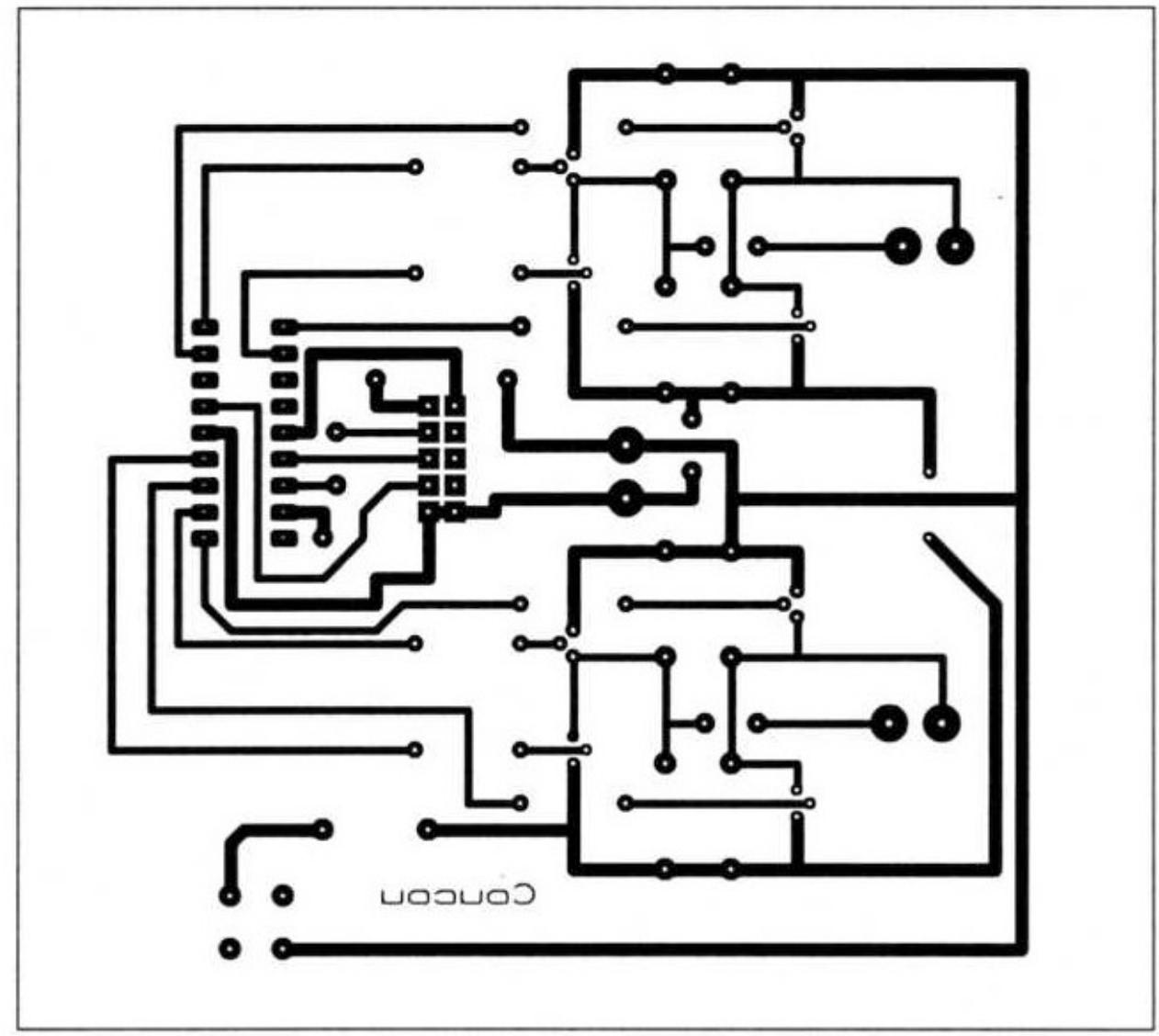

J1
VCC
VDDMCU
HE10-10
B3
BORNIER2
Connecteur
ICD10
PIC 16F88
U1
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
RA5/MCLR
RA0/AN0
RA1/AN1
RA2/AN2/CVREF/VREF-
RA3/AN3/C1OUT/VREF+
RA4/AN4/T0CKI/C2OUT
RB0/INT/CCP1
RB1/SDI/SDA
RB2/SDO/RX/DT
RB3/CCP1
RB4/SCK/SCL
RB5/SS/TX/CK
RB6/AN5/T1OSO/T1CKI
RB7/AN6/T1OSI
TGBA
TGBB
TGHA
TGHB
TDBA
TDBB
TDHA
TDHB
PIC16F88
VDD=VDDMCU
R9
10k
Ponts en "H"
R1
R2
R3
R4
R5
R6
R7
R8
270
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
2N2222A
2N2907
D1
D2
D3
D4
D5
D6
D7
D8
1N4001
B1
B2
BORNIER2

Et ensuite nous avons fait le routage avec le module ARES :



Le gros avantage de ces logiciels est qu'ils comportent des bibliothèques contenant les caractéristiques physiques de presque tous les composants, ce qui sert par exemple à connaître l'espacement entre les pastilles de chacun.
Le schéma du typon est le même que celui du routage sauf qu'il ne faut pas mettre les composants. On l'imprimera ensuite sur du papier calque pour réaliser le circuit.

Le typon :

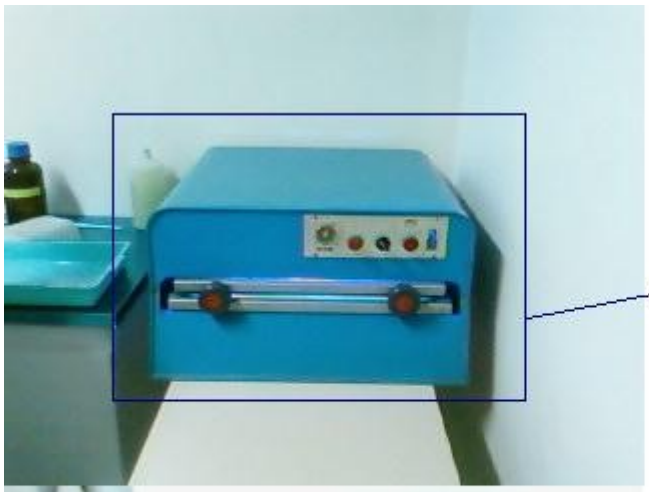


2.Réalisation du circuit.

Une fois que nous avons le typon nous pouvons passer réellement à la réalisation du circuit. Les circuits se font sur des plaques spéciales en fibres de verre recouvertes d'une fine couche de cuivre par dessus laquelle a été répandue une couche de résine recouvertes par un produit chimique sensible aux UV (Ultra Violet) et protégeant du prochain révélateur. Attention il ne faut pas exposer la plaque aux rayons du soleil voilà pourquoi elle est recouverte d'un film anti-UV. Cette réalisation s'effectue en trois étapes :

-l'insolation : cette partie ce fait avec une insoleuse. Nous recouvrons la plaque avec notre calque sur lequel est imprimé le typon après avoir retiré le film anti-UV. Nous insérons la plaque dans l'insoleuse qui va soumettre la partie de la plaque non recouverte par les pistes du typon aux UV et de ce fait va retirer le produit

chimique dont nous avons parlé précédemment sauf aux endroits cachés par le typon. Cette opération a duré 3min15sec.



Insoleuse

calque et plaque superposés

- **la révélation :** cette opération requière l'utilisation d'un produit, le révélateur (soude), nécessitant le port de gants. Le révélateur doit être disposé dans un bac en PVC ou en verre. On met le circuit à l'intérieur du bac, on remue avec attention pendant quelques dizaines de secondes. Le révélateur va attaquer la résine non protégée. On voit alors apparaître peu à peu le cuivre nu, de couleur rose, qui a été soumis aux UV. Le dessin du typon lui est toujours recouvert par la résine.

- **la gravure** : on introduit la plaque dans une machine à graver.

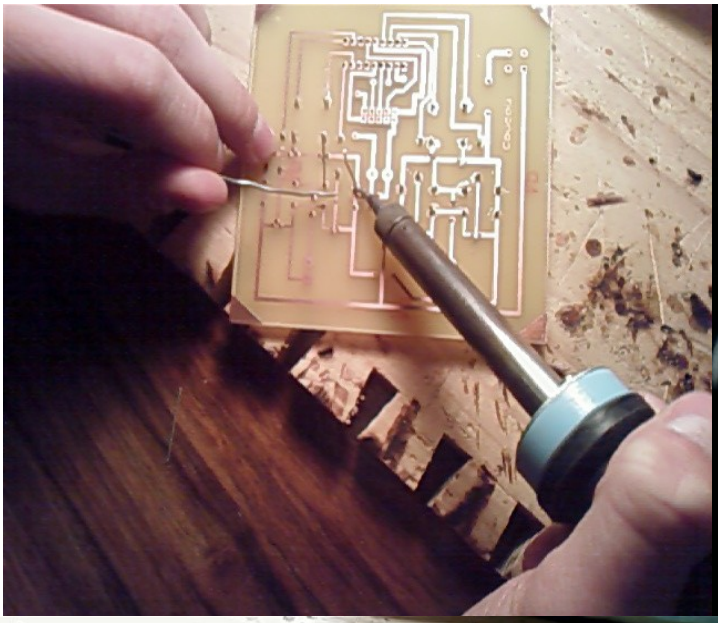
Celle-ci va soumettre la plaque à du perchlorure de fer qui va attaquer le cuivre non protégé par la résine.

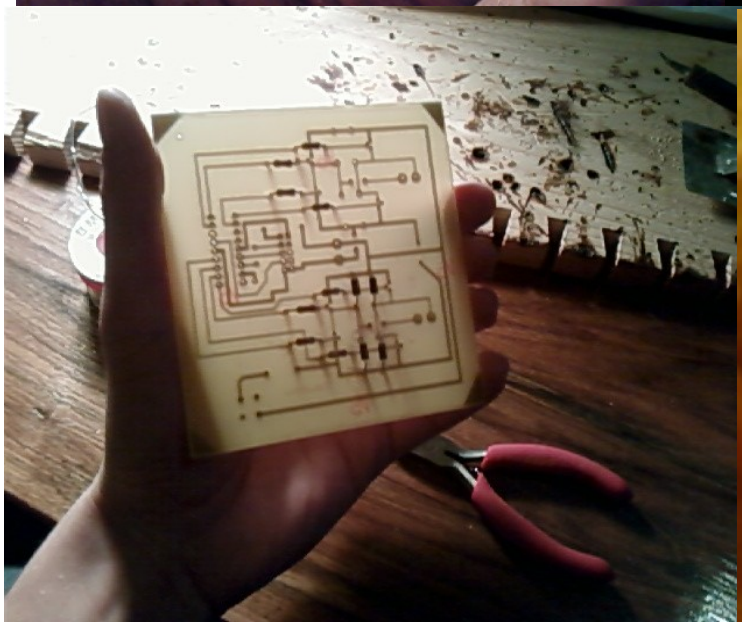
Ainsi nous avons notre circuit imprimé prêt au perçage et à la soudure.

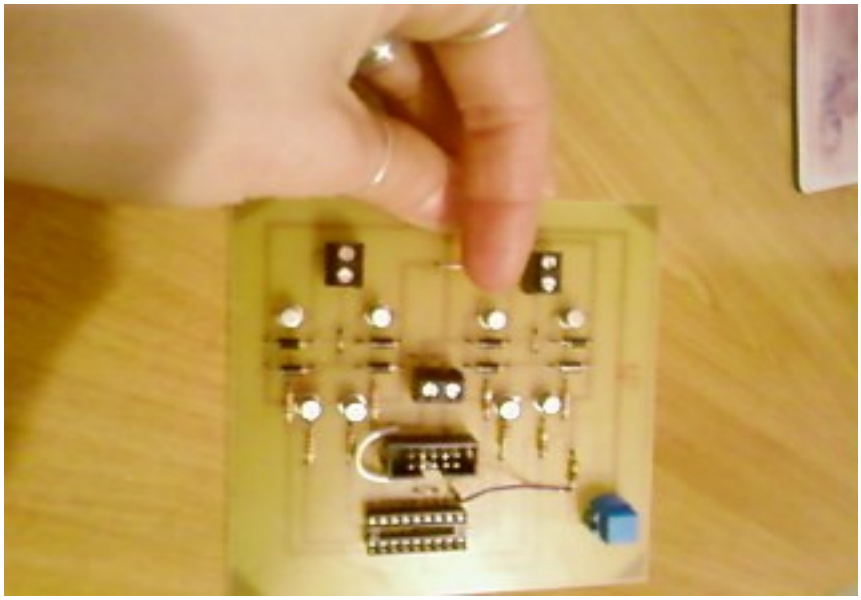
3.Perçage et soudage du circuit.

Tout d'abord il nous faut percer le circuit. Pour cela il nous faut du matériel très précis, ce qui va nous poser problème. En effet nous avons besoin d'une perceuse montée sur une colonne pour percer droit ainsi que de forets très fin de 0,6mm de diamètre pour les composants comme les transistors ou encore le PIC et 0,8mm pour la plupart des autres composants(résistances, diodes, ...). Des forets que nous n'avions évidemment pas puisque le plus petit que nous trouvons est de 1mm de diamètre. Mais nous avons quand même réussis à en « fabriquer » un. Aussi nous avons meulé un foret de 1mm pour y donner une forme conique pour avoir des diamètres allant de 0,5mm à 1mm.

Une fois le circuit percé il nous reste plus qu'à souder les composants du plus petit au plus gros. Pour cela pas de problème nous avons tout ce qu'il faut.





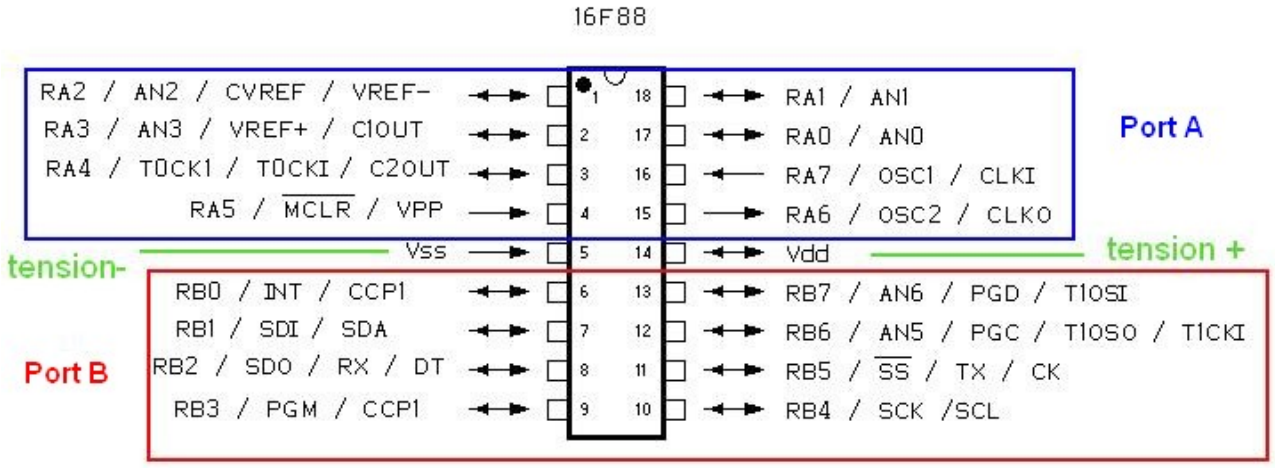


III. La programmation du robot ou comment l'informatique est un point majeur de l'automatisme ?

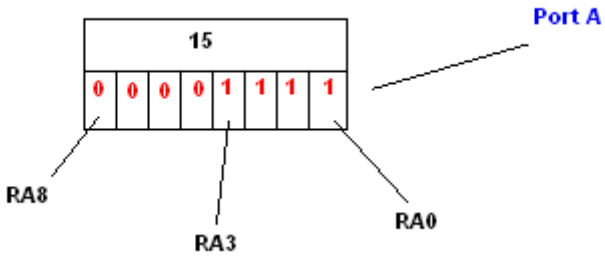
A. Concepts fondamentaux.

1. Précisions sur le PIC.

Comme nous l'avons dit plus haut, le PIC permet de gérer les signaux grâce à des entrées et sorties. Pour vous remémorer cela voici comment est organisé notre microcontrôleur :



En convention le Port A désigne les entrées et le Port B les sorties. Mais en fait le choix revient au programmeur.
Chacun de ces ports est constitué par huit liaisons. Dans notre programme nous allons pouvoir manipuler ces liaisons. Cette manipulation consiste à envoyer du courant ou non à ces liaisons. Ceci se traduira respectivement dans notre programme par un « 1 » ou un « 0 ». Ainsi si on donne la valeur « 1 » à la liaison RA1 le courant circulera. Sinon, si on envoi un « 0 » le courant ne passera pas. Seulement nous ne pouvons pas directement donner une valeur à ces liaisons. Il est vrai que nous ne pouvons que donner une valeur au port lui-même. Cette valeur sera elle convertie en binaire sur 8 bits, c'est à dire qu'elle sera décomposée en huit sous valeurs qui elles seront affectées à chaque liaison. Voyons le schéma suivant :



Ici on donne la valeur 15 au port A. Si on convertit 15 en binaire on obtient :

15 = 2^3+2^2+2^1+2^0

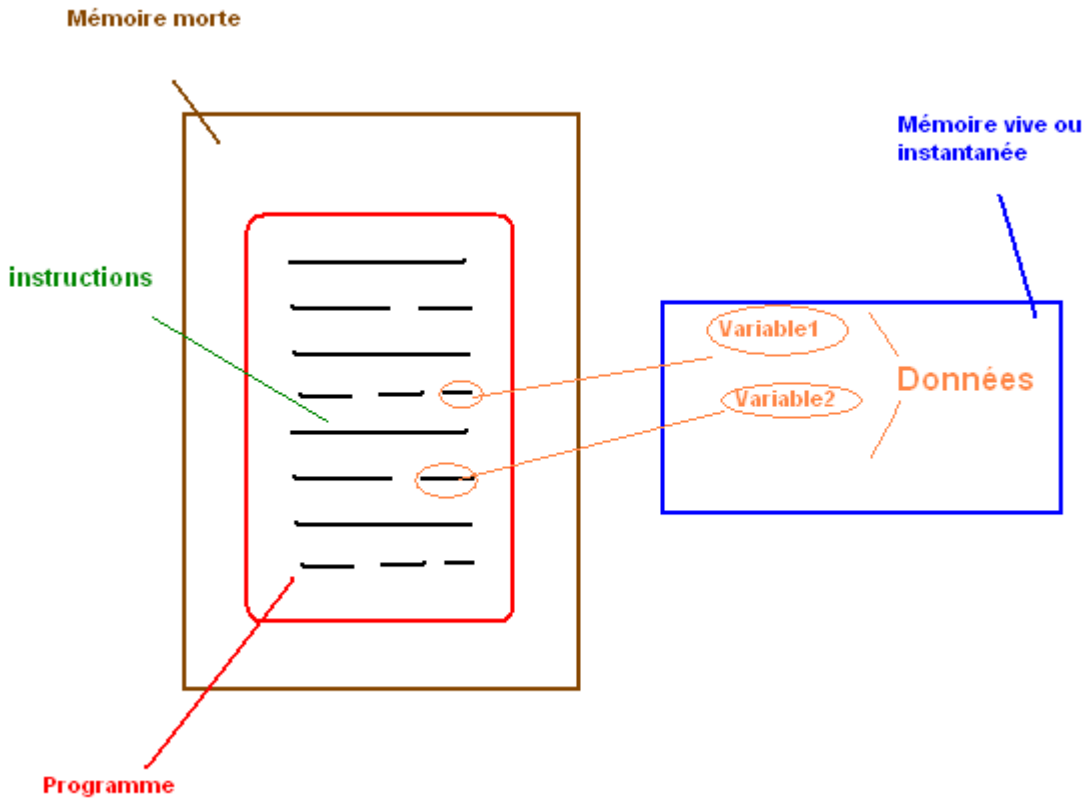
D'où 15 donne 00001111 en binaire.
Comme on le voit sur le schéma, chaque bit de cette valeur binaire est affecté à une liaison. Par exemple RA0 est affecté d'un 1, de même pour RA3 mais RA4 est affecté d'un 0 tout comme RA8. Noté que la lecture se fait de la droite vers la gauche.
Lorsqu'on donne une valeur au port A, on considère que les liaisons sont des sorties mais si on veut se servir de ces liaisons comme entrées il faudra récupérer la valeur qu'il contient. Par exemple si le port A contient la valeur 16, 00010000 en binaire, cela voudra dire que l'entrée RA5 reçoit du courant.
C'est sur ce concept d'entrées/sorties que se base la programmation d'un microcontrôleur.

Et maintenant voyons comment cela se passe dans un programme.

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

Comme nous l'avons expliqué précédemment, un pic possède deux types de mémoire. Une mémoire morte contenant le programme lui-même, c'est une mémoire à très long terme (plusieurs dizaines d'années) et une mémoire vive qui sera accessible par le programme où il pourra stocker ses données.

Voici un schéma montrant ceci :



Maintenant nous pouvons concrètement vous expliquer ce qu'est un programme : c'est une suite d'instructions servant à manipuler des variables et les entrées et sorties précédemment évoquées de la façon suivante :

INSTRUCTION variable
ou INSTRUCTION variable, valeur

Ceci n'est qu'un exemple. On pourrait continuer à expliquer le fonctionnement d'un programme mais ceci serait beaucoup trop long. Retenez seulement qu'une instruction permet d'effectuer des opérations sur des variables comme des affectations, des additions, des soustractions...Mais aussi certaines instructions permettent de faire des sauts dans le programme et aussi des vérifications. C'est grâce à ces instructions que l'on va pouvoir aborder les boucles.
Il faut également savoir qu'une instruction est située à une adresse dans le programme. Par exemple :

Adresse 1 : INSTRUCTION 1
Adresse 2 : INSTRUCTION 2
...
Adresse n : INSTRUCTION n

Sans plus attendre voilà la structure d'une boucle :

Adresse 1 : INSTRUCTION
...
Adresse x : VERIFICATION
Adresse y : SAUT_CONDITIONNEL en Adresse 1
Adresse n : INSTRUCTION n

Ici le programme va exécuter des instructions jusqu'à arriver à la vérification d'une condition quelconque et ensuite le saut se fera si la condition est remplie par exemple et le programme retournera à l'adresse 1.
En programmation on utilise des mots clef pour représenter ces instructions. Sans entrer dans la diversité des langages de programmation voici un pseudo-code représentant une boucle :

```
faire:
        opérations
tant que condition
```

Un exemple avec une variable x qu'on va initialiser à 10 puis la décrémenter jusqu'à ce qu'elle soit nulle :

```
x = 10
faire :
        x = x – 1
tant que x > 0
```

Le concept des boucles est vraiment la base de la programmation. En effet les boucles permettent d'éviter la répétition d'instructions inutile. Voyez plutôt à quoi ressemblerait le « pseudo-programme » précédent si on ne pouvait pas faire de boucles :

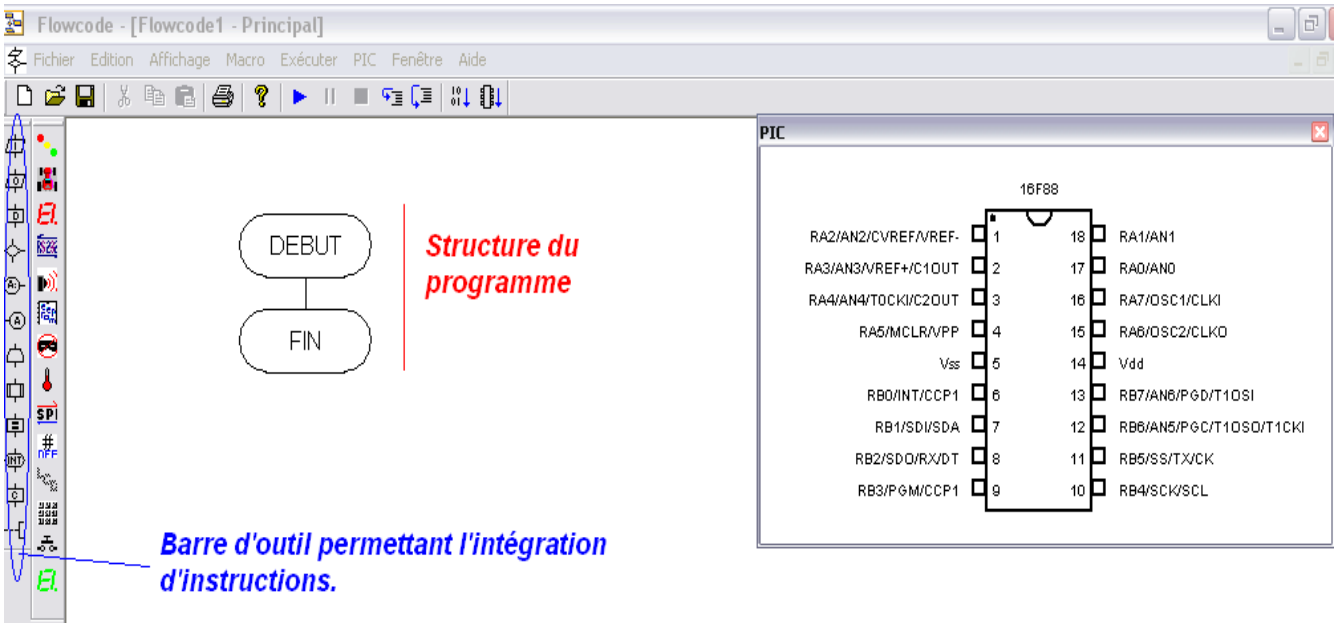
```
x = 10
x = x – 1          // x = 9
x = x – 1
x = x – 1
x = x – 1
x = x – 1
x = x – 1
x = x – 1
x = x – 1
x = x – 1
x = x – 1
x = x – 1          // x  = 0
```

Imaginez si x vaux 10000 au départ!

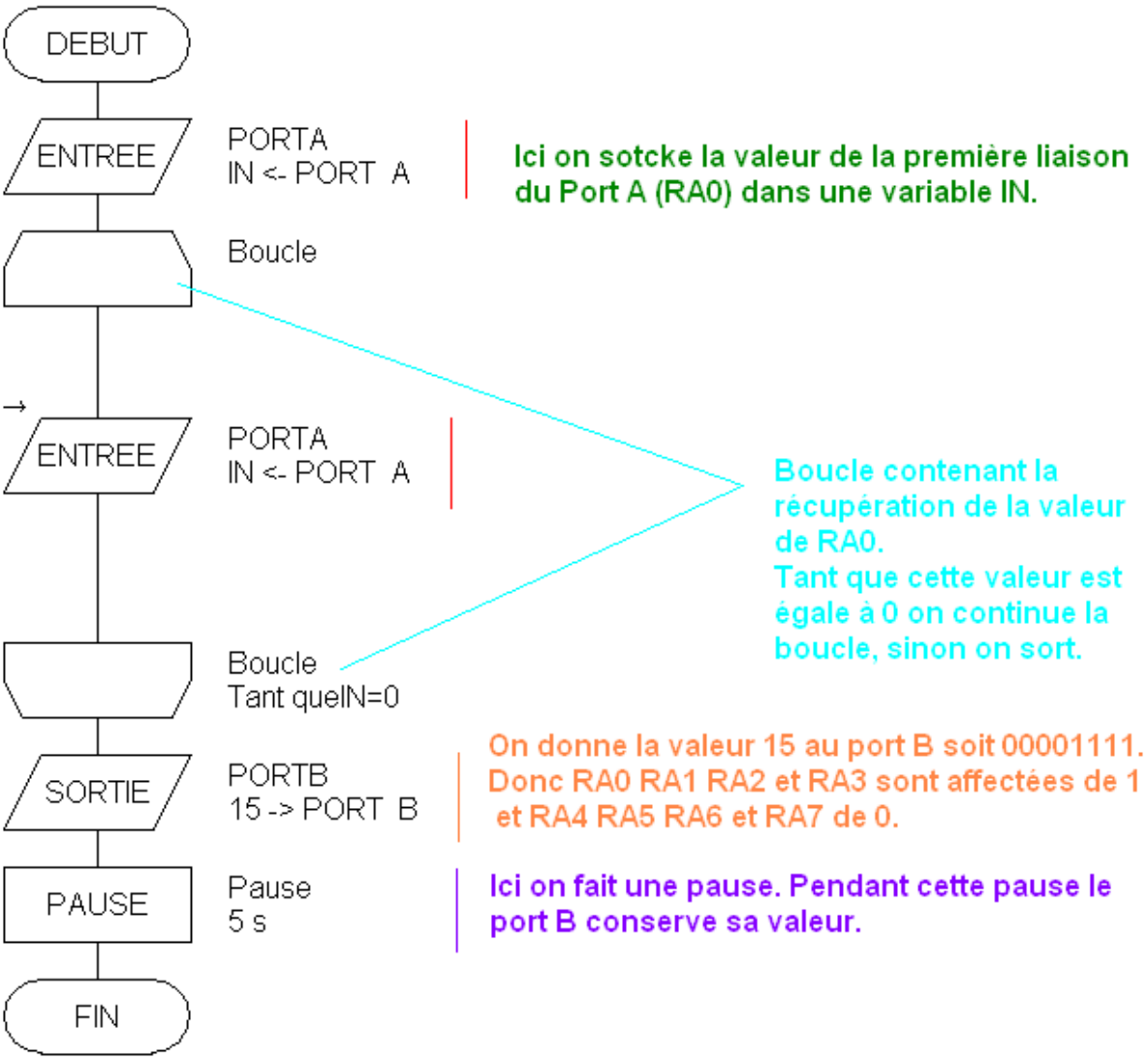
Maintenant que ceci est acquis on va pouvoir réaliser un vrai programme.

3.Vers un premier programme.

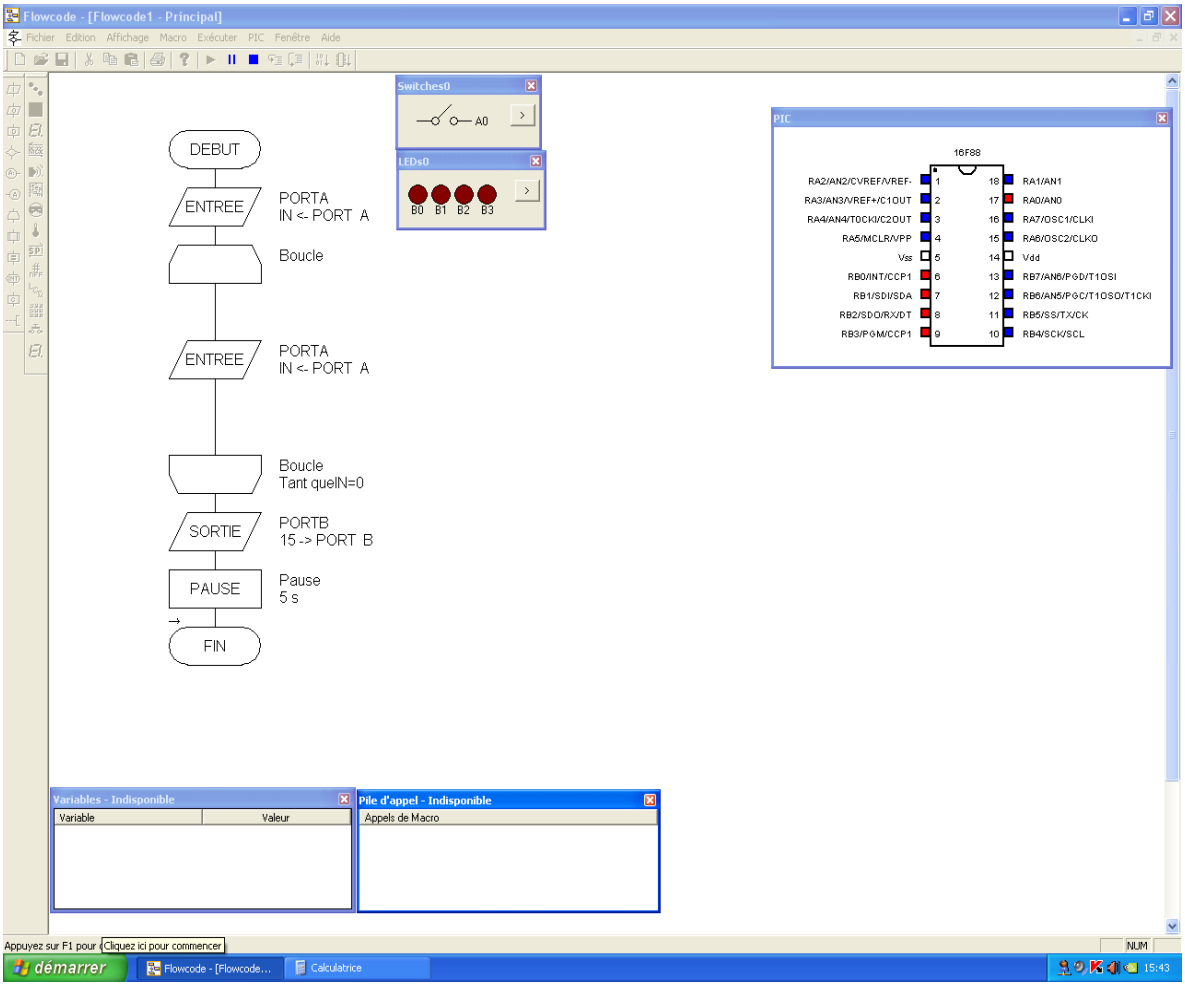
Pour programmer notre PIC nous avons évoqué dans la partie concernant l'électronique qu'il fallait intégrer un connecteur permettant de brancher un appareil appelé microchip. On peut avec cet appareil transférer un programme écrit ou conçut avec un ordinateur sur le pic. Il existe un logiciel permettant de concevoir un programme facilement, il s'agit de flowcode. Nous avons pu nous procurer ce logiciel et créer un premier projet. Il nous faut tout d'abord définir le pic que nous voulons utiliser. Maintenant le programme fournit une première structure de programme que voici :

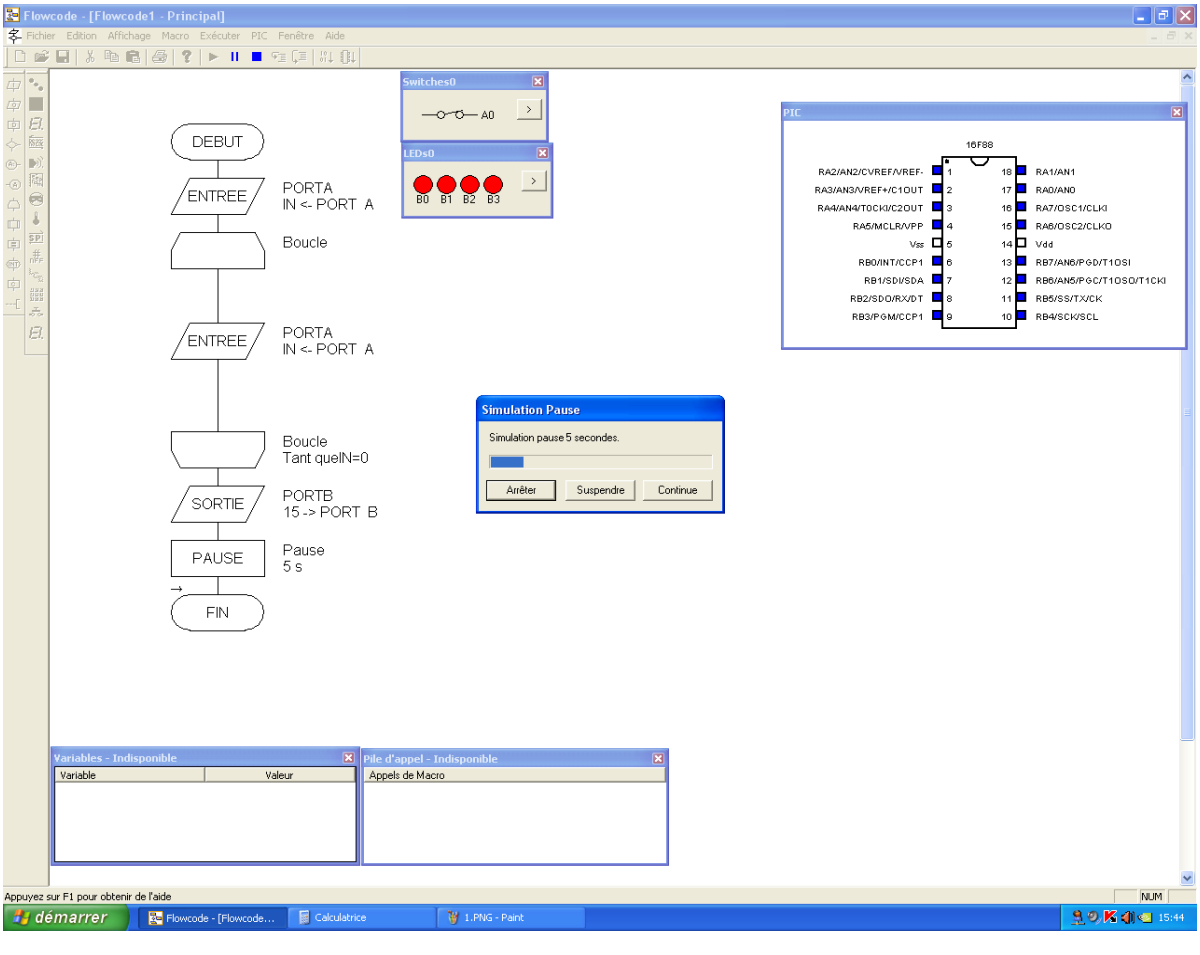


On voit bien que le logiciel permet de faire un programme très facilement à l'aide d'un organigramme. En réalité derrière cet organigramme se cache un programme écrit en C (langage informatique). Ce même programme va être converti en assembleur (un autre langage informatique très proche de la machine). Une nouvelle fois ce programme sera converti en binaire et pourra être interprété par le processeur du pic.
Grâce à Flowcode nous pouvons effectuer des opérations comme des calculs ou des affectations mais encore des boucles et des vérifications.
Un autre avantage de ce logiciel est qu'il permet de simuler le programme et de vérifier son fonctionnement. Nous avons donc pu faire un premier petit programme permettant d'allumer quatre LEDs. Voici le programme commenté:



Et voici la simulation du programme après avoir connecté (virtuellement) quatre LEDs à RB0, RB1, RB2 et RB3 et un interrupteur à l'entrée RA0 :


Flowcode - [Flowcode1 - Principal]
Fichier Edition Affichage Macro Exécuter PIC Fenêtre Aide
Switches0
A0
LEDs0
B0 B1 B2 B3
PIC
16F88
RA2/AN2/CVREF/VREF-
RA3/AN3/VREF+/C1OUT
RA4/AN4/T0CKI/C2OUT
RA5/MCLR/VPP
Vss
RB0/INT/CCP1
RB1/SDI/SDA
RB2/SDO/RX/DT
RB3/PGM/CCP1
RA1/AN1
RA0/AN0
RA7/OSC1/CLKI
RA6/OSC2/CLKO
Vdd
RB7/AN6/PGD/T1OSI
RB6/AN5/PGC/T1OSO/T1CKI
RB5/SS/TX/CK
RB4/SCK/SCL
DEBUT
ENTREE
PORTA
IN <- PORT A
Boucle
ENTREE
PORTA
IN <- PORT A
Boucle
Tant queIN=0
SORTIE
PORTB
15 -> PORT B
PAUSE
Pause
5 s
FIN
Variables - Indisponible
Variable
Valeur
Pile d'appel - Indisponible
Appels de Macro
Appuyez sur F1 pour
Cliquez ici pour commencer
NUM
démarrer
Flowcode - [Flowcode...
Calculatrice
15:43


Flowcode - [Flowcode1 - Principal]
Fichier Edition Affichage Macro Exécuter PIC Fenêtre Aide
Switches0
A0
LEDs0
B0 B1 B2 B3
PIC
16F88
RA2/AN2/CVREF/VREF-
RA3/AN3/VREF+/C1OUT
RA4/AN4/T0CKI/C2OUT
RA5/MCLR/VPP
Vss
RB0/INT/CCP1
RB1/SDI/SDA
RB2/SDO/RX/DT
RB3/PGM/CCP1
RA1/AN1
RA0/AN0
RA7/OSC1/CLKI
RA6/OSC2/CLKO
Vdd
RB7/AN6/PGD/T1OSI
RB6/AN5/PGC/T1OSO/T1CKI
RB5/SS/TX/CK
RB4/SCK/SCL
DEBUT
ENTREE
PORTA
IN <- PORT A
Boucle
ENTREE
PORTA
IN <- PORT A
Boucle
Tant queIN=0
SORTIE
PORTB
15 -> PORT B
PAUSE
Pause
5 s
FIN
Simulation Pause
Simulation pause 5 secondes.
Arrêter
Suspendre
Continue
Variables - Indisponible
Variable
Valeur
Pile d'appel - Indisponible
Appels de Macro
Appuyez sur F1 pour obtenir de l'aide
NUM
démarrer
Flowcode - [Flowcode...
Calculatrice
1.PNG - Paint
15:44

Encore persiste un atout de ce logiciel, c'est qu'il permet d'exécuter le programme en mode « pas à pas » c'est à dire instruction après instruction. Ceci permet de localiser et corriger les erreurs. On parle alors de « débogage » (debugging en anglais). Vu que notre programme est très petit nous n'avons pas besoin de procéder ainsi.
Maintenant que tous ces principes sont expliqués nous allons pouvoir établir un programme pour notre robot.

B. Le programme de notre robot.

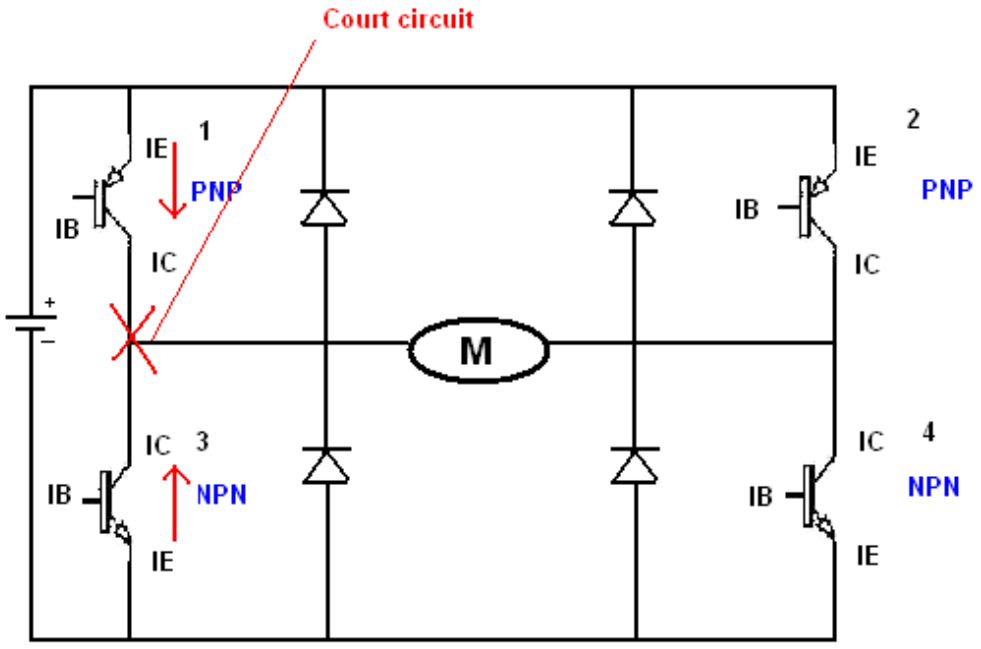
1. Définition des mouvements.

Les mouvements que nous aimerions faire effectuer à notre robot sont:

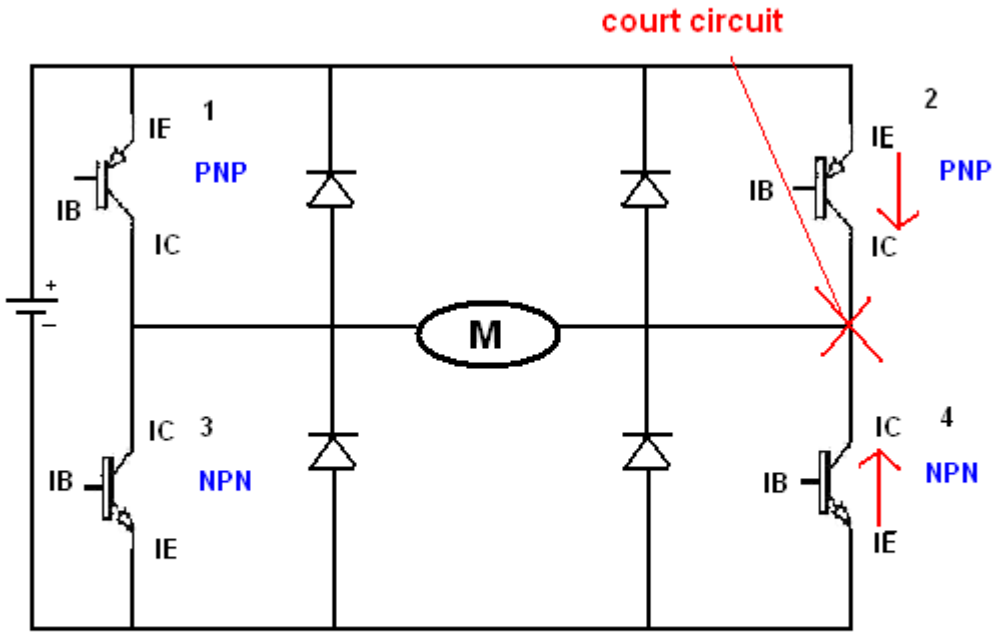
- la marche avant, les deux moteurs tournent dans un même premier sens.
- la marche arrière, les moteurs tournent dans le sens contraire.
- la rotation sur lui-même, les moteurs tournent dans des sens inverses (l'un dans un sens l'autre dans l'autre).
- le mouvement circulaire, un moteur tourne plus vite que l'autre mais tout de même dans le même sens.

2. Spécifications nécessaires.

Il nous faut faire attention à ne pas créer de cours circuit au niveau de notre pont en H. Voici les configurations à éviter:



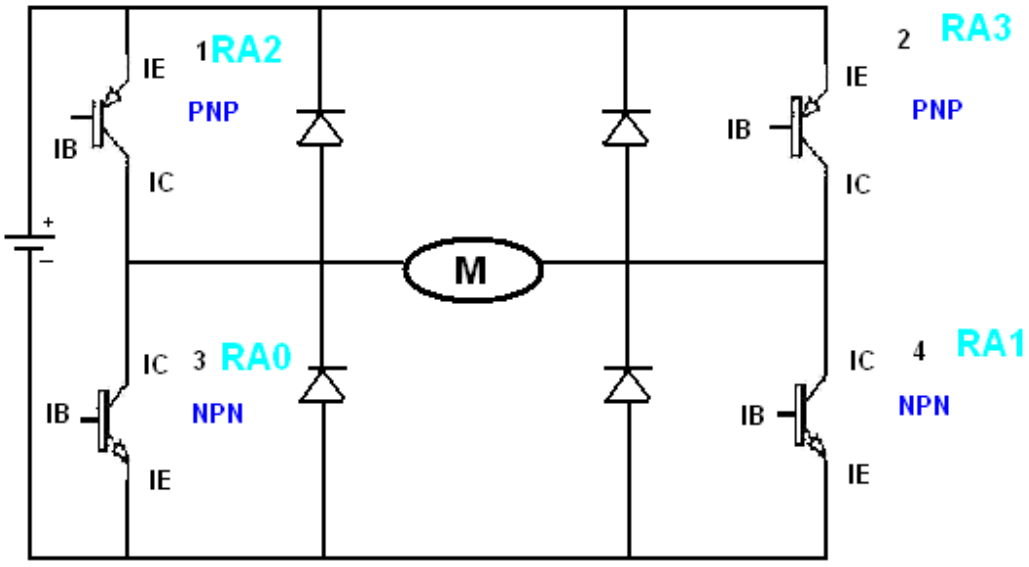
Les transistors 1 et 3 sont passants. Il y a donc un court circuit.



Les transistors 2 et 4 sont passants. Il y a de nouveau un court circuit.

Il nous faut maintenant regarder à quelles liaisons du pic sont reliés chaque transistors et établir la liste des valeurs interdites pour chaque ports.

D'après le schéma électrique on en déduit le schéma suivant:



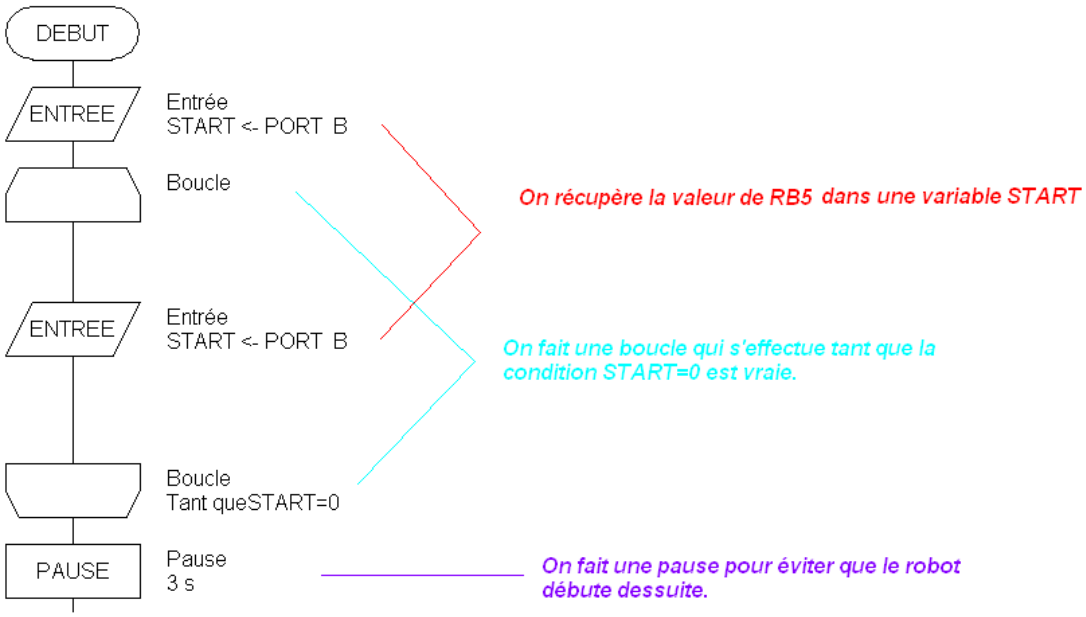
De ceci nous allons en déduire les valeurs interdites pour le port A qui seront les mêmes pour le port B.
Rappel: pour qu'un transistor PNP soit passant il ne faut pas envoyer de courant en IB. Et pour les NPN, il faut en envoyer. Pour que le transistor 1 soit saturé, il faut affecter RA2 de la valeur 0. Pour le transistor 4, il faut affecter RA1 de la valeur 1.
Tableau des valeurs interdites:

RA0	RA1	RA2	RA3	Correspondance en décimal
Situation 1				
1	0	0	0	8
1	0	0	1	9
1	1	0	0	12
1	1	0	1	13
Situation 2				
0	1	0	0	4
0	1	1	0	6
1	1	0	0	12
1	1	1	0	14

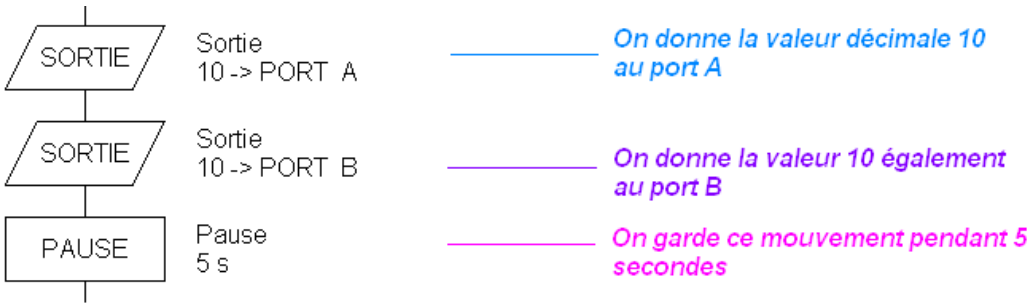
3. Conception du programme.

Toutes ces précautions prises nous pouvons élaborer notre programme.
Tout d'abord nous savons que l'ouverture ou la fermeture de notre circuit se fait avec un bouton poussoir que nous avons connecté à RB5 du port B.
On va donc faire une boucle qui va vérifier si du courant arrive en RB5. Si c'est le cas, on sort de la boucle et on continue le programme.

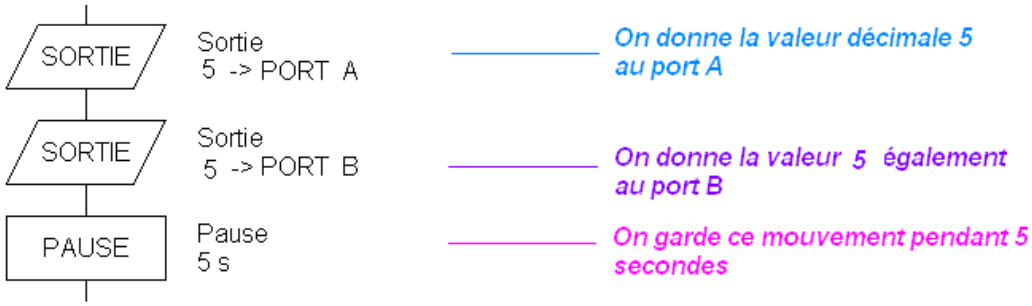
Voici ce que cela donne :



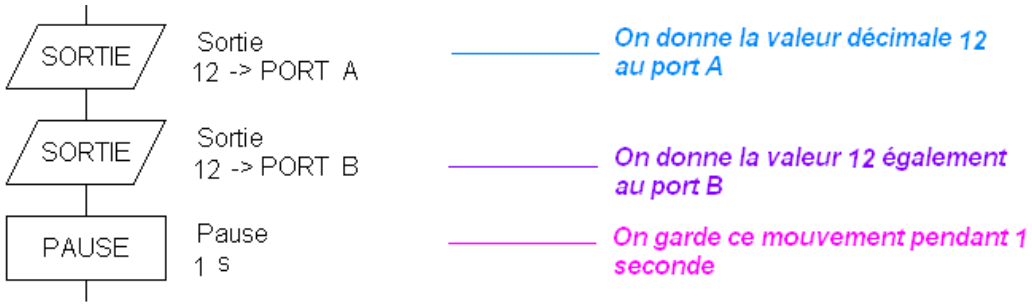
Ensuite on va faire avancer notre robot. Pour cela il faut deux transistors correspondant soit passant. Il faut donc par exemple pour le port A que RA0 reçoive la valeur 0, RA1 la valeur 1, RA2 0 et RA3 1.
On a donc la valeur 1010 soit 10 pour le port A. Pour le port B ce sera de même puisque les branchements sont effectués dans le même ordre.
Voici la séquence d'instructions :



Maintenant pour reculer c'est l'inverse, il nous faut la valeur binaire 0101 soit 5 en décimal. On a :



Seulement on ne peut pas avancer puis reculer dessuite sinon on risque d'avoir un court circuit. Il faut donc arrêter les moteurs. Pour que tous les transistors soient bloqués il faut donner la valeur 1100 au port soit 12 en décimal. On a :

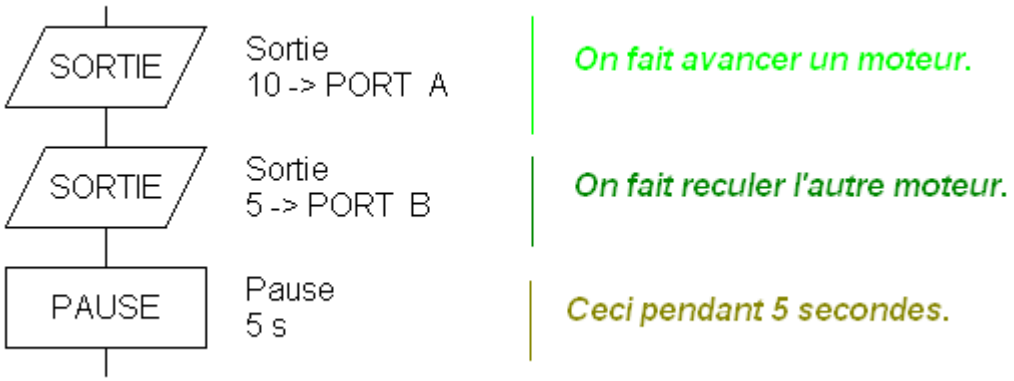


Nous avons donc trouver les séquences d'instructions permettant de faire avancer, reculer et arrêter le robot :

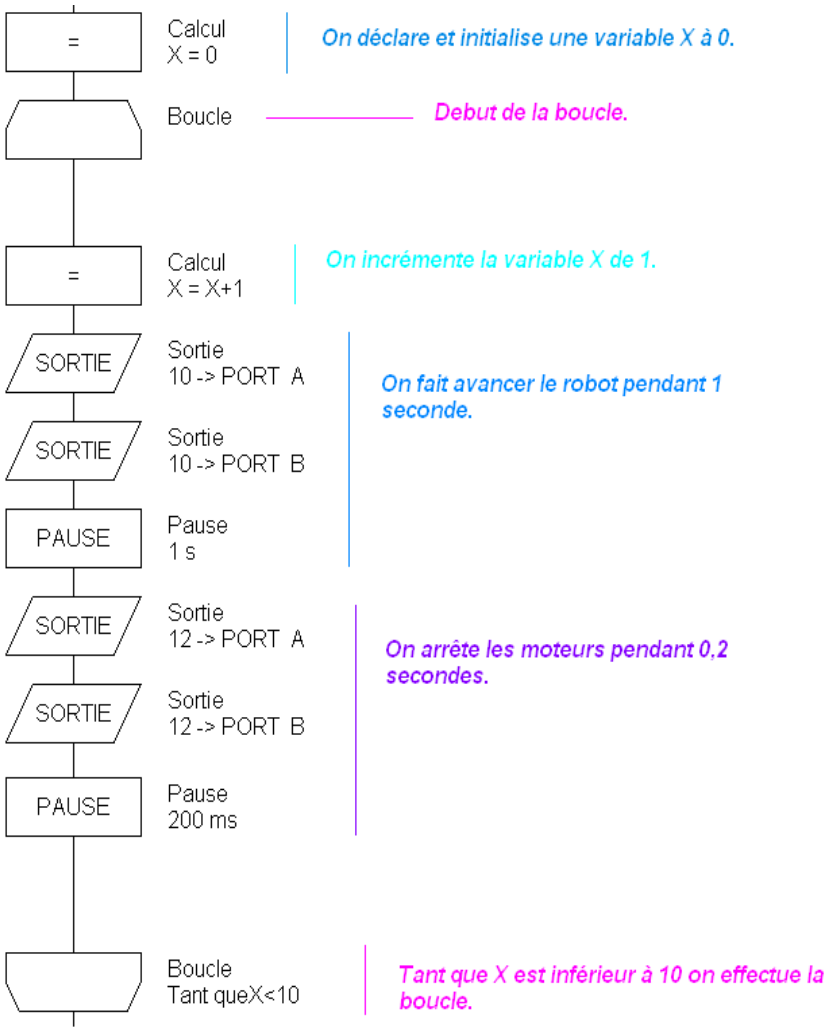
-la valeur 10 pour un port fait avancer.
-la valeur 5 fait reculer.
-la valeur 12 pour s'arrêter.

On va donc à chaque fois réutiliser ces valeurs.

Voici l'illustration de ces valeurs pour faire effectuer au robot un mouvement de rotation sur lui-même :

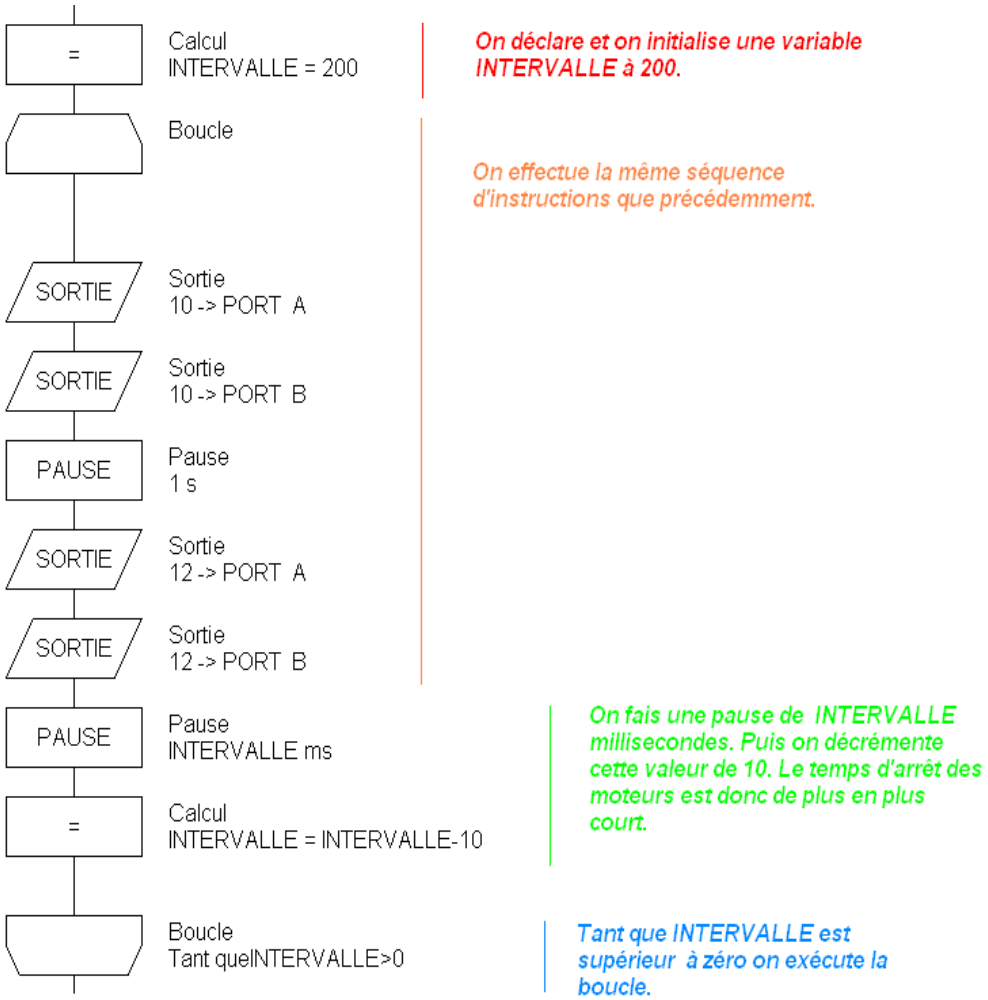


Nous l'avions dis précédemment, avec nos transistors on peut moduler le signal et ainsi faire avancer le moteur à une allure modérée.
Pour faire cela nous allons faire une boucle qui alimentera puis coupera les moteurs sur des petits intervalles de temps.
Voici le résultat :

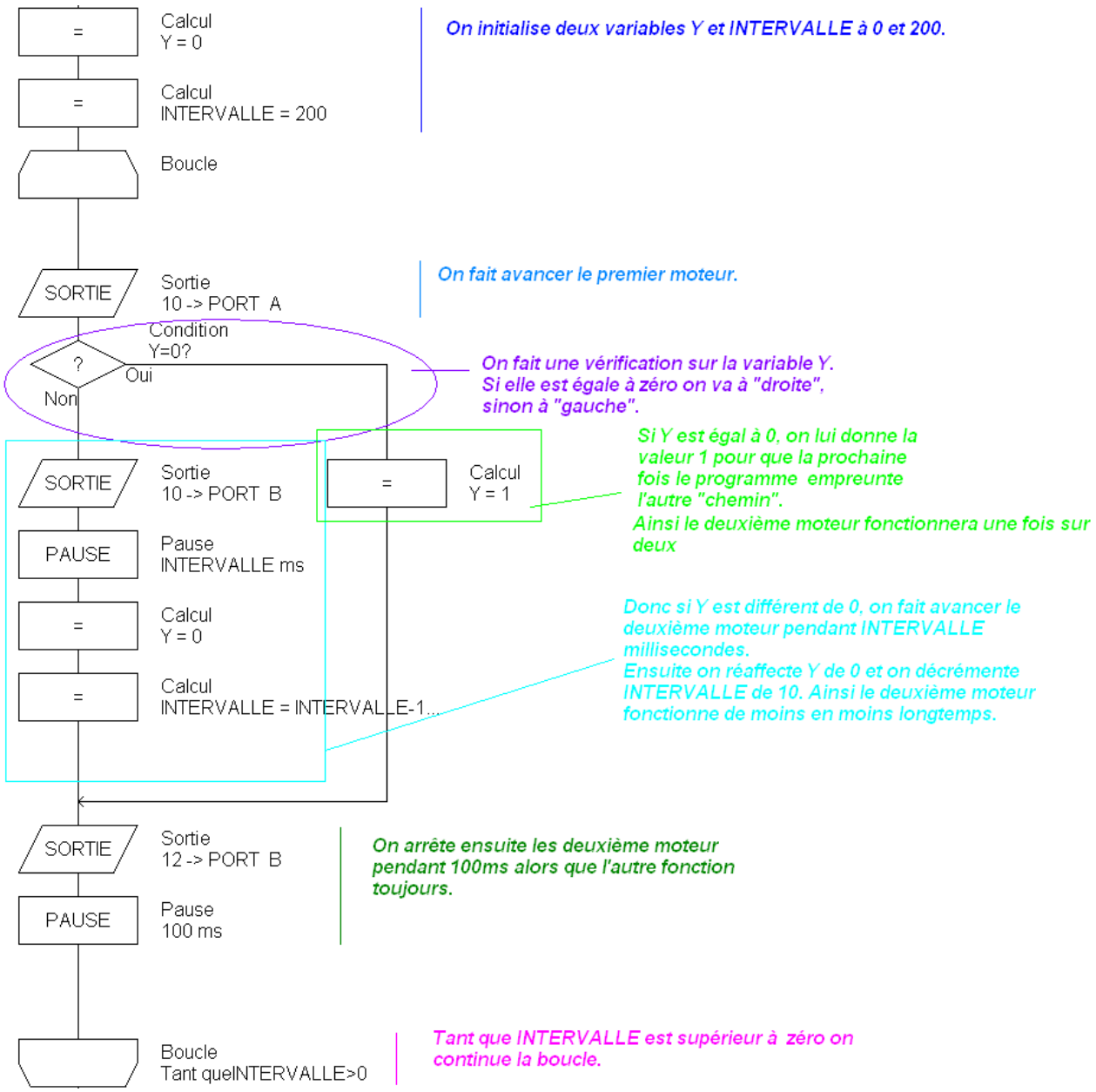


Voilà maintenant on va faire une accélération. Là il va falloir agir sur l'intervalle de temps d'arrêt des moteurs et le rendre de plus en plus petit.

Ceci donne :



Voilà pour l'accélération.
Nous nous sommes maintenant lancé le défis de faire effectuer au robot un mouvement en spirale. Le code commenté :



C'est à peu près tout pour le programme de notre robot. Il reste encore beaucoup de mouvements possible mais nous avons réellement manqué de temps, nous vous avons donc montré les principaux ou les plus pertinents.

4. Vers une programmation plus évoluée.

Il est vrai que Flowcode est très pratique d'utilisation cependant il s'avère très limité pour la programmation. Nous avions abordé plus haut les langages informatiques comme le C ou l'assembleur. En effet ces langages offrent beaucoup plus de possibilités, mais la complexité est également au rendez-vous. Voici le tout premier programme que nous avons réalisé avec Flowcode mais cette fois-ci en C :

```
//Definir pour microcontrôleur

//Fonctions PIC
#pragma CLOCK_FREQ 40000
#define P16F88
#include <system.h>

#define MX_EE
#define MX_EE_TYPE2
const char MX_EE_SIZE = 256;
#define MX_SPI
#define MX_SPI_B
#define MX_SPI_SDI 1
#define MX_SPI_SDO 2
#define MX_SPI_SCK 4
#define MX_UART
#define MX_UART_B
#define MX_UART_TX 5
#define MX_UART_RX 2


//Déclarations de fonction Macro


//Déclarations de Variable
char FCV_IN;

//Implémentations Macro



void main()
{
//Initialisation PIC>
ansel = 0;
cmcon = 0x07;

//Code d'initialisation d'Interruption
option_reg = 0xC0;


//On récupère la valeur du premier bit du port A dans la variable FCV_IN

        TRISA = TRISA | 0x01;
        FCV_IN = PORTA & 0x01;

        /* On fait une boucle infinie

En effet while(1) signifie "tant que la condition 1 est vrai" et 1 est  toujours vrai. */

        while(1)
        {
                TRISA = TRISA | 0x01;

//On récupère la valeur du premier bit du port A dans la variable FCV_IN

                FCV_IN = PORTA & 0x01;
// Si  (if) la variable FCV_IN est égale à 1 alors on sort de la boucle (break)

                if (( FCV_IN == 0 ) == 0) break;
        }
//On affecte les quatre premiers bits du port B de la valeur 1.
 En effet la valeur 0xF0 en hexadécimal coresspondant à 00001111 est affecté au port B.

        TRISB = TRISB & 0xf0;
        PORTB = PORTB & 0xf0 | 15;

//On attend 5 secondes

        delay_s(5);
//On termine le programme par une boucle infini qui ne fait rien.

mainendloop: goto mainendloop;
}
```

On voit bien la complexité de ce code. En effet il requière beaucoup de connaissance dans ce langage. Nous nous attarderons donc pas plus là-dessus.

Conclusion

En conclusion l'automatisme réside en trois points principaux, la mécanique, l'électronique et surtout la programmation. La partie la plus difficile à mener aura été l'électronique. Effectivement nous avons été très gênés par la complexité de ce domaine surtout que nous n'avions presque aucune connaissance de base.

A noter que tous les schémas ont été réalisés par nos soins. De plus aucune de nos explications n'est tirée d'internet ou de documentation toute faite.

SOMMAIRE

I. Le déplacement du robot ou comment la mécanique intervient dans l'automatisme.

A. Choix de nos moteurs et de l'alimentation.

B. Le choix des roues

C. Choix du châssis.

II. La conception de notre circuit électrique ou comment l'électronique sert l'automatisme.

A. Le choix des composants.

1. Les ponts en « H » pour commander les moteurs.

2. Le choix des transistors et du PIC

a) Les transistors.

b) Le PIC, un composant clé de l'automatisme.

c) Détermination des transistors.

B. La conception du circuit

1.Conception du schéma électrique et routage.

2.Réalisation du circuit.

3.Perçage et soudage du circuit.

III. La programmation du robot ou comment l'informatique est un point majeur de l'automatisme.

A. Concepts fondamentaux.

1. Précisions sur le PIC.
2. Vers un premier programme.

B. Le programme de notre robot.

1. Définition des mouvements.
2. Spécifications nécessaires.
3. Conception du programme.
4. Vers une programmation plus évoluée.

Remerciements

Nous remercions M. Loupias, M. Poiris ainsi que les autres professeurs d'électronique, Mme Bastide, M. Marret, Mme Laville, Mme Moulinier, Mme Gueyrard.

En espérant n'avoir oublié personne.

Documentations

Sites web :

-http://p.may-chez-alice.fr
-http://fribotte.free.fr
-http://www.robotix.fr
-http://geii.iut-nimes.fr
-http://www.farnell.fr
-http://lextronic.fr

Et bien d'autres...

Livres :

-*Formation pratique à l'électronique moderne*, Michel Archambault.

-Manuels d'électronique scolaires.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>fl1.c</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="generator" content="SynEdit HTML exporter" />
<style type="text/css">
<!--
body { color: #000000; background-color: #FFFFFF; }
.cpp1-assembler { color: #0000FF; }
.cpp1-character { color: #000000; }
.cpp1-comment { color: #3399FE; font-style: italic; }
.cpp1-float { color: #800080; }
.cpp1-hexadecimal { color: #800080; }
.cpp1-identifier { color: #000000; }
.cpp1-illegalchar { color: #000000; }
.cpp1-number { color: #800080; }
.cpp1-octal { color: #800080; }
.cpp1-preprocessor { color: #008000; }
.cpp1-reservedword { color: #000000; font-weight: bold; }
.cpp1-space { background-color: #FFFFFF; color: #000000; }
.cpp1-string { color: #FF0000; }
.cpp1-symbol { color: #000000; }
-->
</style>
</head>
<body>
<pre>
<code><span style="font: 10pt Courier New;"><span class="cpp1-comment">//Definir
pour microcontr&ocirc;leur

//Fonctions PIC
</span><span class="cpp1-preprocessor">#pragma CLOCK_FREQ 40000
#define P16F88
#include &lt;system.h&gt;

#define MX_EE
#define MX_EE_TYPE2
</span><span class="cpp1-reservedword">const</span><span class="cpp1-space">
</span><span class="cpp1-reservedword">char</span><span class="cpp1-space">
MX_EE_SIZE = </span><span class="cpp1-number">256</span><span class="cpp1-
symbol">;
</span><span class="cpp1-preprocessor">#define MX_SPI
#define MX_SPI_B
#define MX_SPI_SDI 1
#define MX_SPI_SDO 2
#define MX_SPI_SCK 4
#define MX_UART
#define MX_UART_B
#define MX_UART_TX 5
#define MX_UART_RX 2


</span><span class="cpp1-comment">//D&eacute;clarations de fonction Macro


//D&eacute;clarations de Variable
</span><span class="cpp1-reservedword">char</span><span class="cpp1-space">
FCV_IN;

</span><span class="cpp1-comment">//Impl&eacute;mentations Macro
```

```
</span><span class="cpp1-reservedword">void</span><span class="cpp1-space">
main()
{
</span><span class="cpp1-comment">//Initialisation PIC&gt;
</span><span class="cpp1-identifier">ansel = </span><span class="cpp1-
number">0</span><span class="cpp1-symbol">;
cmcon = </span><span class="cpp1-hexadecimal">0x07</span><span class="cpp1-
symbol">;

</span><span class="cpp1-comment">//Code d'initialisation d'Interruption
</span><span class="cpp1-identifier">option_reg = </span><span class="cpp1-
hexadecimal">0xC0</span><span class="cpp1-symbol">;


<span class="cpp1-comment">//On récupère la valeur du premier bit du port A dans
la variable FCV_IN
      </span>
      TRISA = TRISA | </span><span class="cpp1-hexadecimal">0x01</span><span
class="cpp1-symbol">;
      FCV_IN = PORTA &amp; </span><span class="cpp1-
hexadecimal">0x01</span><span class="cpp1-symbol">;

      <span class="cpp1-comment">/* On fait une boucle infinie
<br>En effet while(1) signifie "tant que la condition 1 est vrai" et 1 est
toujours vrai. */
      </span>
      </span><span class="cpp1-reservedword">while</span><span class="cpp1-
symbol">(</span><span class="cpp1-number">1</span><span class="cpp1-symbol">)
      {
            TRISA = TRISA | </span><span class="cpp1-
hexadecimal">0x01</span><span class="cpp1-symbol">;

<span class="cpp1-comment">//On récupère la valeur du premier bit du port A dans
la variable FCV_IN
      </span>
            FCV_IN = PORTA &amp; </span><span class="cpp1-
hexadecimal">0x01</span><span class="cpp1-symbol">;
<span class="cpp1-comment">// Si  (if) la variable FCV_IN est égale à 1 alors on
sort de la boucle (break)
      </span>
            </span><span class="cpp1-reservedword">if</span><span class="cpp1-
space"> (( FCV_IN == </span><span class="cpp1-number">0</span><span class="cpp1-
space"> ) == </span><span class="cpp1-number">0</span><span class="cpp1-
symbol">) </span><span class="cpp1-reservedword">break</span><span class="cpp1-
symbol">;
      }
<span class="cpp1-comment">//On affecte les quatre premiers bits du port B de la
valeur 1.<br> En effet la valeur 0xF0 en hexadécimal coresspondant à 00001111
est affecté au port B.
      </span>
      TRISB = TRISB &amp; </span><span class="cpp1-hexadecimal">0xf0</span><span
class="cpp1-symbol">;
      PORTB = PORTB &amp; </span><span class="cpp1-hexadecimal">0xf0</span><span
class="cpp1-space"> | </span><span class="cpp1-number">15</span><span
class="cpp1-symbol">;

<span class="cpp1-comment">//On attend 5 secondes
      </span>
      delay_s(</span><span class="cpp1-number">5</span><span class="cpp1-
symbol">);
<span class="cpp1-comment">//On termine le programme par une boucle infini qui
ne fait rien.
```

```
      </span>
mainendloop: </span><span class="cpp1-reservedword">goto</span><span
class="cpp1-space"> mainendloop;
}



</span></span>
</code></pre>
</body>
</html>
```