

**$\mu$ Robot :**

**Le robot  
suiveur  
de ligne**

**...**

**Projet réalisé par :  
El Jayidi Nawfal  
Benkirane Malik**

**Encadré par :  
M. ROBERT**

## Introduction

En première, déjà, nous étions fascinés par les domaines informatiques et électroniques. Cette motivation qui s'est avérée être une véritable passion fut le moteur de nos TPE. Notre projet était de réaliser un robot, ou plutôt un automate capable de suivre une ligne. L'idée par sa simplicité nous a fait rire, à tel point que notre enthousiasme nous amenait à des projets délirants. Mais nous négligions la richesse du problème que nous avions mis en place lors de nos TPE : Comment créer un "robot" capable de suivre une ligne?

$\mu$ Robot était né, mais dès les premiers essais, les espoirs formulés durant les quatre mois de développement se sont effondrés. Un suivi de ligne idéal s'était transformé en un suivi approximatif et incontrôlé. De cette expérience insatisfaisante, nous avons compris qu'un suivi de ligne aussi simple qu'il pouvait paraître, avait des applications bien plus complexes. En nous intéressant à ces applications, nous avons décidés, à l'occasion de cette XVIIIe édition des Olympiades de la Physique, de réétudier notre problématique et cela dans le souci de respecter une démarche scientifique rigoureuse, en s'intéressant plus particulièrement au problème :

« Comment suivre une ligne ? »

<b>I - La détection d'une ligne .....</b>	<b>0</b>
A - Le principe de la détection de la ligne :.....	3
B - La disposition des capteurs .....	5
1 - Inefficacité d'un système de capteurs alignés .....	5
2 - Une première alternative : "système d'endiguement".....	6
3 - Modification de la disposition et du rôle des capteurs .....	7
<b>II - Le "cerveau" du robot :.....</b>	<b>9</b>
A - Le rôle du contrôleur intelligent .....	9
B - Algorithme d'initialisation .....	10
1. Le message d'erreur.....	10
2. Le programme d'initialisation .....	10
C. Algorithme de traitement de ligne .....	11
1. Le suivi de ligne.....	11
2. L'optimisation du suivi de ligne .....	11
A - Le photo-transistor .....	12
B - Le contrôleur programmable intelligent 16F84 :.....	13
C - Le L298 : commande des moteurs .....	14
<b>IV - Réalisation pratique .....</b>	<b>16</b>
A - Structure générale.....	16
B - PWM ou Pulse Width Modulation .....	17
C - Spécifications techniques sur les capteurs.....	19
Conclusion.....	20

# I - La détection d'une ligne

Notre problématique nous amène à chercher à créer un robot capable de suivre une ligne. Mais qu'est-ce qu'une ligne ?

On définit souvent une ligne comme un trait ou même une courbe, ce qui est peut-être plus exact car cette définition se résume à dire qu'une ligne est une suite de points. Certes, mais en réalité, on observe une ligne lorsque celle-ci a une épaisseur et est suffisamment grande pour que notre œil puisse la voir. De plus il faut impérativement qu'elle ait une couleur assez différente du fond sur lequel elle est tracée.

Pour des raisons pratiques (impression...), on choisira pour nos prototypes une ligne de couleur noire qui sera dessinée, imprimée ou tracée sur un fond blanc. L'épaisseur de la ligne devra être suffisante pour que les "yeux" du robot la détectent mais modérément pour qu'elle ne soit pas considérée comme une figure pleine.

## A - Le principe de la détection de la ligne :

Nous savons qu'une radiation de longueur d'onde  $\lambda$  sera absorbée par un matériau, si celui-ci diffuse toutes les radiations sauf celle ayant cette longueur d'onde, et elle sera diffusée par un autre matériau diffusant des radiations dont une a pour longueur d'onde  $\lambda$ .

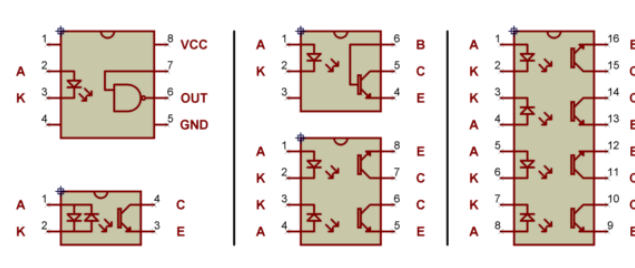
Pour notre cas, nous avons utilisé deux "couleurs" : le noir qui absorbe toutes les longueurs d'ondes dont les infrarouges, et le blanc qui, lui, diffuse toutes les longueurs d'ondes dont les infrarouges.

Quant aux yeux de notre robot, ils ne devront qu'être capables de détecter l'absence ou la présence de la ligne. Par conséquent des capteurs isolés devraient suffire. On assimilera par la suite ces capteurs à des points, mais en réalité ils ont une épaisseur donnée qu'il ne faudra pas négliger (raison pour laquelle la ligne doit avoir une épaisseur minimum).

Nos "yeux" seront des capteurs optiques détectant la présence ou l'absence d'un faisceau lumineux de longueur d'onde supérieure à 800 nm, donc dans les infrarouges. Il faut prendre en considération l'environnement de notre robot qui n'est pas forcément dans une obscurité totale; les seules sources de lumière non négligeables dans une salle proviennent de néons ou de lampes à incandescence dont les infrarouges émis ne perturbent pas notre prototype.

Les capteurs infrarouges que nous allons utiliser seront associés à des émetteurs qui émettent dans la même zone de longueurs d'ondes ( $> 800$  nm) que les capteurs. On obtient alors des couples émetteurs/capteurs que l'on notera couples ou complexe E/C.

Plusieurs composants en électronique permettent ces applications "optiques" : ce sont les composants opto-électroniques. Il existe des composants qui d'ailleurs suivent parfaitement le principe décrit ci-dessus : ce sont les opto-coupleurs (Figure 1). Mais ces derniers n'étant pas disponibles en magasin nous les avons "fabriqué maison".



*Figure 1 : Différents opto-coupleurs*

Nous avons donc dû coupler un émetteur infra-rouge à un récepteur infrarouge. L'émetteur de base est une simple DEL (diode électro luminecente : composant s'allumant lorsqu'elle est traversée par un courant dans le sens passant) infra-rouge. Pour les récepteurs, il en existe de deux types : les diodes IR réceptrices et les photo-transistors IR.

Lors de nos TPE nous avons déjà testé les diodes et elles s'avéraient très peu sensibles donc inefficaces et cela même pour de très faibles distances; par contre, pour des photo-transistors, les résultats de la Figure 2 s'avèrent être ceux attendus.

Manipulations réalisées :

1. Variation de la distance entre diode émettrice et photo-transistor (1)
2. Variation de la distance entre le complexe E/C et une feuille de papier blanche (2)
3. Variation de la distance entre le complexe E/C et une feuille de papier noire (3)

Manipulation	Distance entre les deux objets pris pour la manipulation (cm)						Tension entre le GND (0V) et la borne E du phototransistor (V)
	0	1	2	3	4	5	
(1)	4.8	4.1	2.2	1.8	0.8	0.3	
(2)	3.5	2.3	1.6	0.7	0.4	0.1	
(3)	0.1	0.0	0.0	0.0	0.0	0.0	

*Figure 2 :Résultats des manipulations (les cases vertes correspondes au HIGH logique et rouges au LOW logique)*

Pour résumer, nous utilisons donc un couple E/C constitué d'une DEL IR et d'un photo-transistor IR. A noter que l'environnement d'un photo-transistor sera précisé dans le paragraphe (III-A).

Mais comment disposer ces capteurs de manière à ce que le robot puisse suivre et de manière efficace la ligne ?

## B - La disposition des capteurs

**Attention**, dans cette sous partie, nous allons parler de détection ou de non détection de la ligne, contrairement à la sous partie précédente dans laquelle on parlait de noir et de blanc, détectés ou non par les capteurs.

### 1 - Inefficacité d'un système de capteurs alignés

C'est le système que nous avons présenté pour nos TPE : il consiste à disposer 3 capteurs alignés sur une ligne perpendiculaire (et décalée par rapport au centre du robot) au sens de déplacement du robot (Figure 3a)

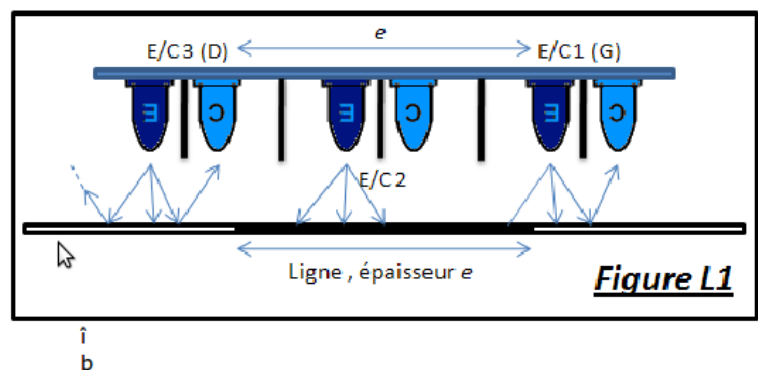
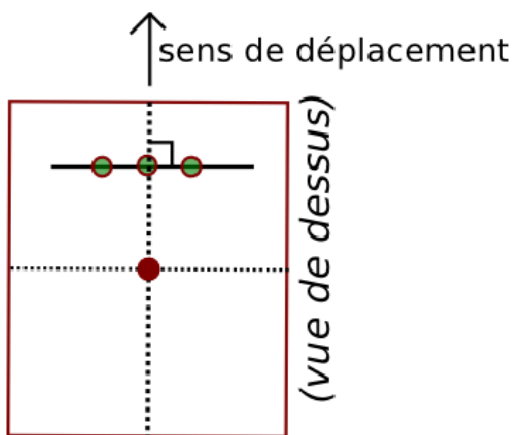


Figure 3 : disposition des capteurs pour un système aligné

<- a

Ce système consiste à détecter les deux sorties de lignes possibles, en supposant qu'initialement le robot est correctement placé sur la ligne. Si on détecte que le robot sort de celle-ci, il tournera à l'instant même dans le sens favorisant le retour sur la ligne. Ici, nous utilisons le capteur central pour certifier si le robot est bien sur la ligne (sachant que son épaisseur est suffisamment grande pour ne pas être "cachée" entre deux capteurs).

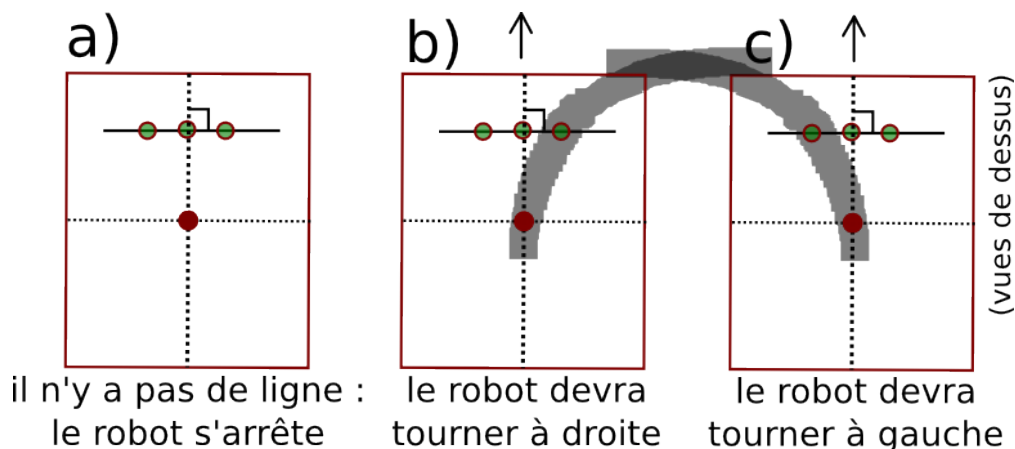


Figure 4 : Système simplifié de suivi de ligne mis en place

Pour résumer le système de suivi qui découle de ce système à trois capteurs alignés, on distingue trois cas (conditions de l'algorithme) :

- Si le robot détecte la ligne à droite, il tourne à droite (Figure 4b)
- Si le robot détecte la ligne à gauche, il tourne à gauche (Figure 4c)
- Si le robot ne détecte rien, il s'arrête (Figure 4a)

Ce système, en se limitant à cette conception théorique, semble totalement fonctionner mais on a constaté expérimentalement, lors des premiers tests du robot, qu'il tourne soit trop à gauche, soit trop à droite : ce qui peut gêner un suivi de ligne correct et même quelques fois sera la cause d'une sortie de la ligne.

On pourrait expliquer cette défaillance technique par le fait que l'analyse de la ligne se fait en continue, c'est-à-dire lorsque le robot avance :

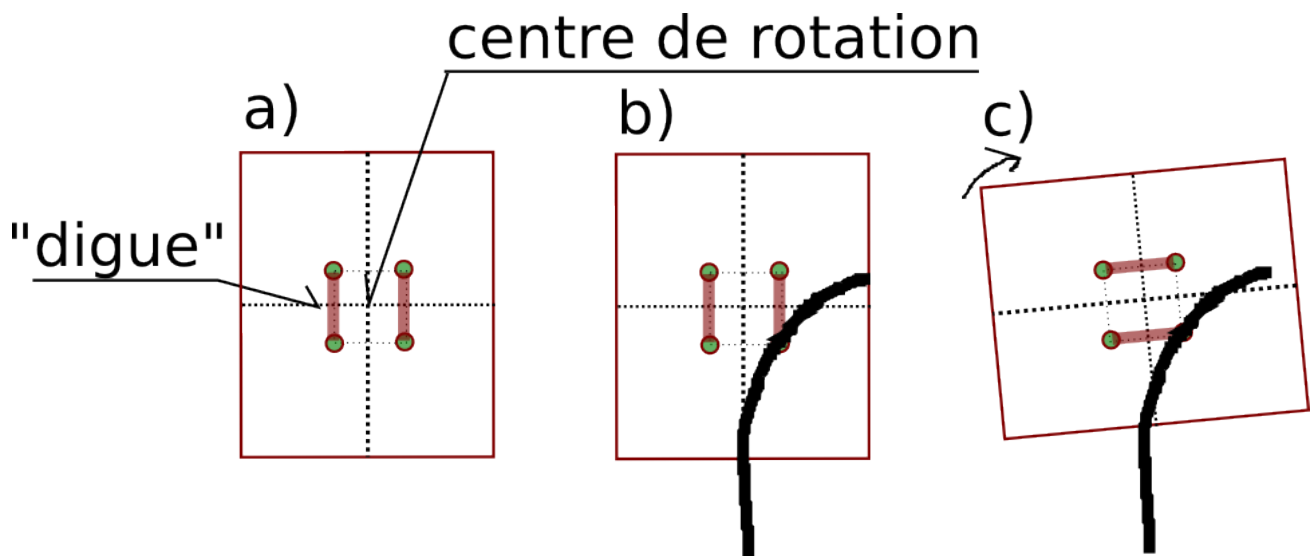
- il y a alors un décalage entre l'analyse des données et l'action mécanique des moteurs
- mais surtout, tant que l'un des capteurs latéraux est sur la ligne, le robot ne cesse de tourner : à la sortie du virage le robot trop rapide est dévié de la ligne, ou a du mal à effectuer un virage dans l'autre sens.

C'est sur ce second problème, que nous allons nous focaliser.

## 2 - Une première alternative : "système d'endiguement"

Cette notion d'endiguement vient directement du système précédent, non par une interprétation unidimensionnelle mais bidimensionnelle du problème : "comment suivre une ligne". En effet ce premier système fait en quelque sorte une coupe unidimensionnel d'une zone par la suite analysée simplement en une étape.

Le but de ce nouveau système est donc de créer un système qui délimiterait la ligne non pas par deux points mais par deux lignes (Figure 5a).



*Figure 5*

Or ce qui nous intéresse c'est la détection à l'avant du robot car ce dernier avance : on pourrait faire une analogie avec des «yeux». On devra donc donner la priorité aux capteurs arrières (Figure 5b), pour empêcher les sorties de ligne. Puis l'ajustement alors **quantifié**, se fera à l'avant (Figure 5c).

On peut remarquer que pour cet ajustement, le robot est sorti de la ligne et est donc hors de la limite imposée par ses "digues". Et cela est dû uniquement au fait que le robot tourne par rapport à un point qui ne devrait pas être confondu avec le centre du carré formé par les capteurs.

Pour remédier à ce problème, on doit donc déplacer les capteurs par une translation de vecteur  $2\vec{u}$  (Figure 6a). En appliquant le même système que le précédent, on peut constater au niveau des Figures 6b et 6c que l'ajustement s'est bien fait en fonction de la forme de la courbe.

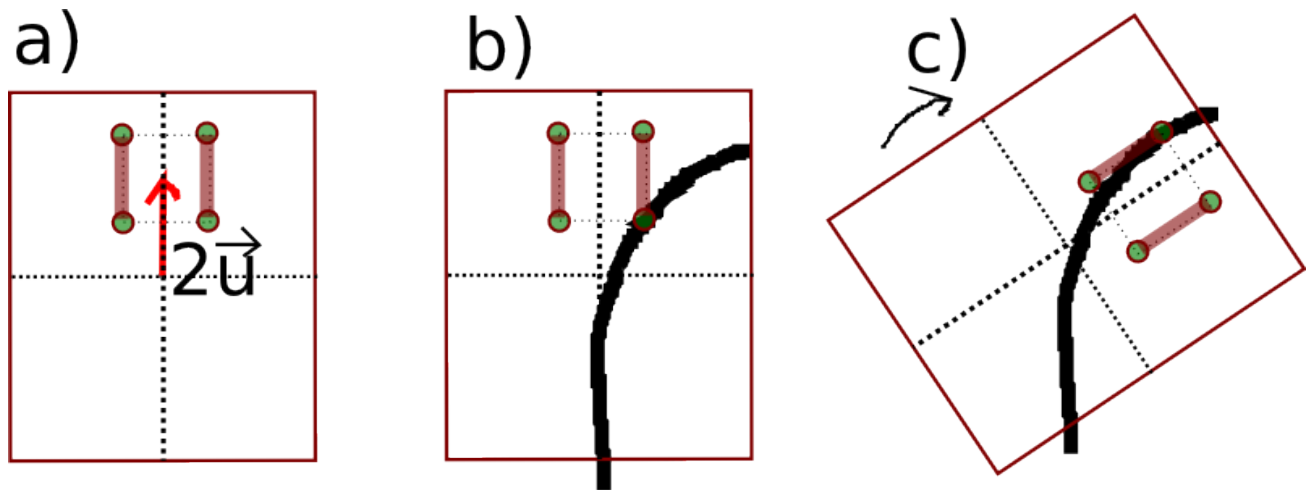


Figure 6

### 3 - Modification de la disposition et du rôle des capteurs

On peut remarquer, en simulant à de nombreuses reprises le système précédent, qu'il est possible de ne pas utiliser deux capteurs avants mais un seul (Figure 7a). En testant le nouveau système on remarque qu'il est optimal pour l'ajustement de toute courbe modérément inclinée. (Figure 7b)

Par contre, pour des courbes beaucoup plus inclinées, on ne peut plus effectuer de réajustements avec le capteur central car il est déjà placé sur la ligne. Par conséquent le robot resterait sur place avec un « court-circuit » de l'éventuel algorithme. (Figure 7c)

Pour y remédier, nous sommes dans l'obligation d'ajouter un capteur central que l'on notera dC : le capteur de précision. Celui-ci étant sur le même axe que le capteur C (central), il a la même fonction mais à une échelle plus réduite. On pourra donc suivre plus précisément, mais plus lentement. La lenteur du suivi de ligne que l'on peut visualiser à la figure 7d nous pousse à utiliser le capteur dC de manière événementielle : lorsque le capteur C (central) devient inutilisable.

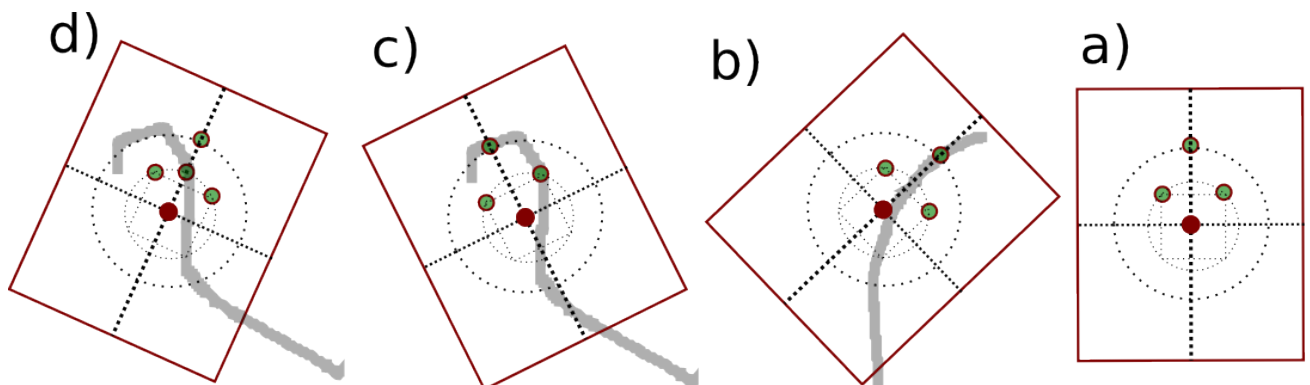
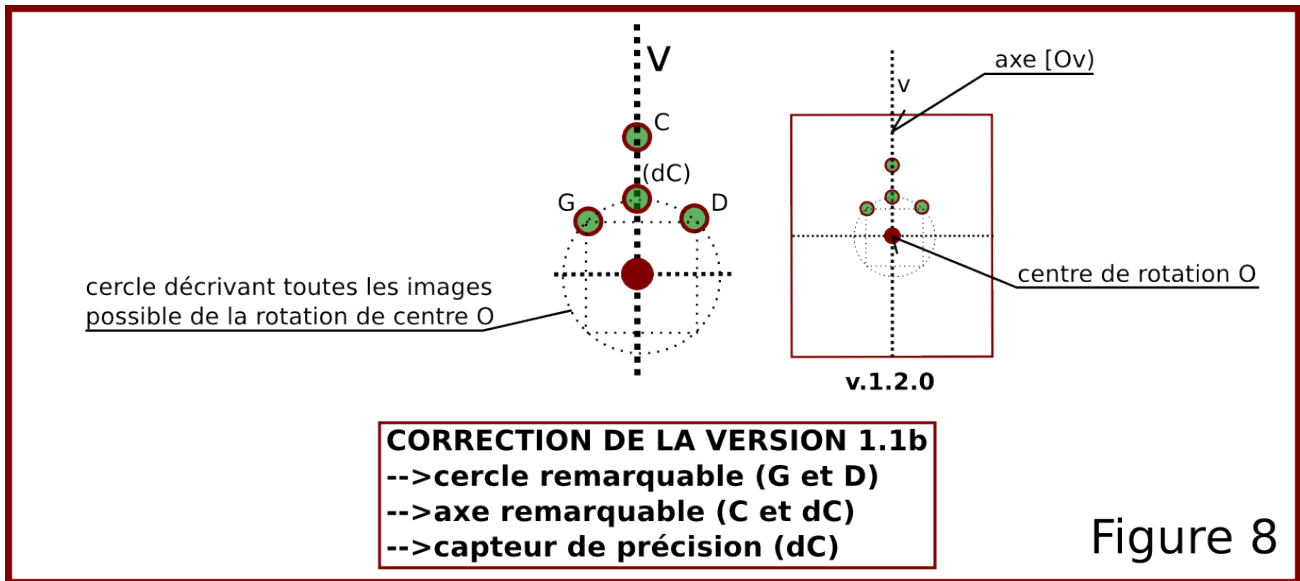


Figure 7

Le nouveau système :  $\mu$ Robot v.1.2, à 4 capteurs, sera donc celui utilisé. (Figure 8)





## II - Le "cerveau" du robot :

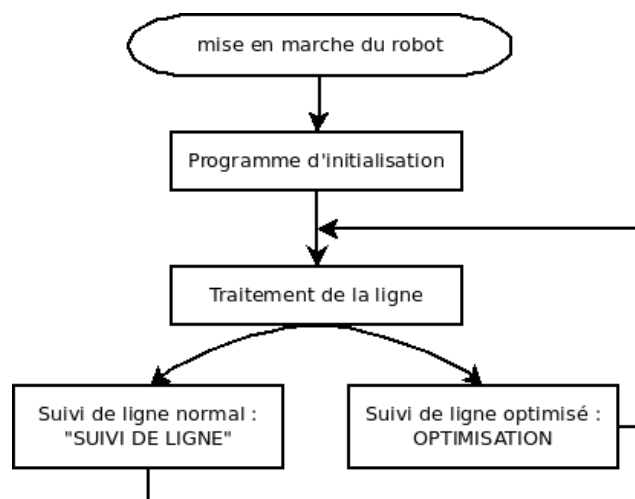
### A - Le rôle du contrôleur intelligent

Dans cette partie (II), nous allons devoir décrire des algorithmes c'est à dire " un processus systémique de résolution [...] d'un problème permettant de présenter les étapes vers le résultat. En d'autres termes, un algorithme est un énoncé d'une suite finie et non-ambiguë d'opérations permettant de donner la réponse à un problème. " (Wikipédia)

Mais notre prototype est bien trop petit pour qu'on puisse y poser un ordinateur. On utilise alors des micro-contrôleurs programmables à partir d'un ordinateur. Comme le processeur de tout ordinateur, ce dernier a un "jeu" d'instruction et doit normalement être codé en assembleur. Les développeurs ont créé des langages de programmation de plus haut niveau comme le C ou encore le C++ (orienté objet) pour faciliter la programmation, les processeurs ayant de plus en plus d'instructions.

Pour ce qui est de notre processeur, le PIC16F84, il n'a qu'un jeu de 37 instructions simples que l'on peut retrouver sur son "datasheet" (fiche technique). Le but d'un programme est de mettre en relation ces instructions pour qu'elles soient structurées de façon à ce que l'on obtienne l'algorithme souhaité, d'où la nécessité de réaliser ces algorithmes.

Mais la réalisation de ces derniers ne suffit pas car il faut aussi déterminer les relations entre eux, sachant qu'au final on ne devra avoir qu'un seul programme contenant tous ces algorithmes. (Figure A1)



*Figure A1 : relation entre les différents programmes*

## B - Algorithme d'initialisation

### 1. Le message d'erreur

Ce message est réutilisé à plusieurs reprises dans le programme donc on le définira maintenant et y fera allusion par la suite.

Il affiche sur l'écran « Replacez-moi » et les DEL clignotent toutes à intervalles d'une seconde.

Ce message s'efface une fois que le robot est remplacé et que le bouton 2 est enfoncé pour confirmer l'action citée précédemment.

On définit aussi les notations suivantes : une DEL éteinte correspond à "+" et une DEL allumée à "0".

- Bp1 correspond au bouton gauche
- Bp2 correspond au bouton central
- Bp3 correspond au bouton droit

### 2. Le programme d'initialisation

Dès que le robot est mis en marche, il entame une rotation à gauche pour vérifier la présence d'une ligne et cela par le biais de ses capteurs : si le capteur gauche détecte la ligne, alors le robot est considéré comme en dehors de celle-ci (figures A2) et nous envoie le message d'erreur défini ci-dessus.

Par contre, si le capteur droit détecte la ligne, cela signifie que la ligne se situe bien entre les capteurs latéraux (figures A3). Mais pour confirmer cela, il faut que le robot parvienne à se replacer sur la ligne à l'aide de son capteur central : le robot va effectuer une rotation à droite jusqu'à ce que le capteur central détecte la ligne (figures A4).

Mais si la ligne est alors détectée par un capteur latéral au lieu du capteur central, cela voudra dire que le robot n'est pas situé au dessus d'une ligne mais est légèrement décalé (figures A5). Dans ce cas le robot nous demande de le remettre sur la ligne en affichant le message d'erreur (cf III.B.1).

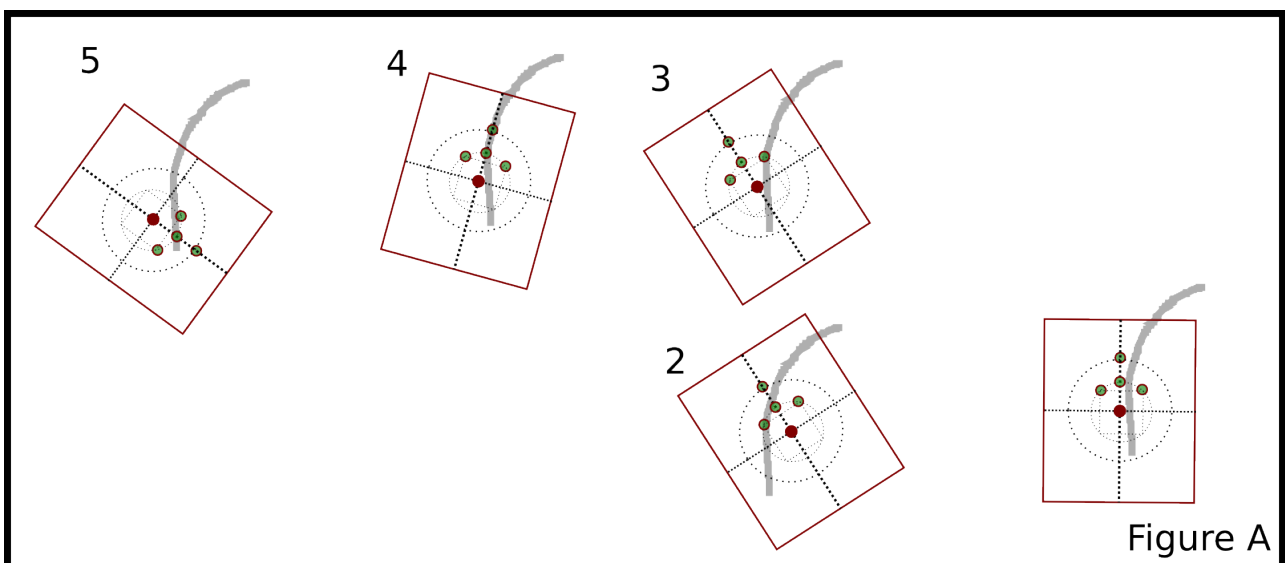


Figure A

Dans le cas contraire c'est à dire lorsque le capteur centrale a détecté la ligne, le robot est bien positionné sur la ligne. L'initialisation est donc terminée et le traitement de la ligne peut commencer.

## **C. Algorithme de traitement de ligne**

Comme nous l'avons vu sur la figure A1 et dans le paragraphe précédent, le traitement de la ligne ne commence que si le robot est sur la ligne c'est à dire que l'initialisation a permis au robot de traiter la ligne.

Le robot se met alors à avancer et dès qu'il commence à avancer, un traitement de la ligne se met en place et se répète à la fréquence d'exécution du processeur pendant que le robot avance. Ce programme va permettre au robot de différencier le cas où il doit optimiser le suivi de ligne et le cas où le suivi doit se faire de manière normale.

La différence entre ces deux modes de suivi réside dans leur définition ; en effet comme nous l'avons précisé dans le I.C.3 :

- + le mode optimisé intervient lorsque l'un des capteurs latéraux et le capteur central détectent en même temps la ligne : cela signifie que la ligne est bien trop inclinée pour que le système par défaut agisse, cette vérification est donc prioritaire au cas du suivi de ligne ordinaire.

- + Le suivi de ligne ordinaire s'effectue donc lorsqu'il n'y a pas d'optimisation, c'est à dire que la ligne n'est détectée que par l'un des capteurs latéraux.

Ce programme de traitement de ligne débouche alors sur deux programmes différents en fonctions des deux cas précédents.

### **1. Le suivi de ligne**

Dans ce premier cas, grâce à l'algorithme précédent, le robot sait qu'il est en présence d'une ligne courbée modérément. Il vérifie alors lequel des capteurs latéraux détecte la ligne.

- +Si c'est le capteur droit qui détecte la ligne, cela signifie qu'il est en train de dévier par la gauche il doit donc effectuer une rotation à droite.

- +Si c'est le capteur gauche qui détecte la ligne, cela signifie le cas contraire : il dévie par la droite et doit donc effectuer une rotation à gauche.

Cette rotation continue tant que le capteur central n'a pas détecté la ligne. Et s'il la détecte, la rotation s'arrête et l'ajustement par rapport à la ligne est terminé. Le programme se terminant alors, il repasse la main au programme du traitement de la ligne afin que le robot continue de suivre le reste de la ligne.

### **2. L'optimisation du suivi de ligne**

Dans ce second cas, grâce au traitement de ligne qui a précédé l'appel de ce programme, le robot sait qu'il est en présence d'une ligne très courbée. Il vérifie alors comme pour le suivi de ligne ordinaire, lequel des capteurs latéraux détecte la ligne.

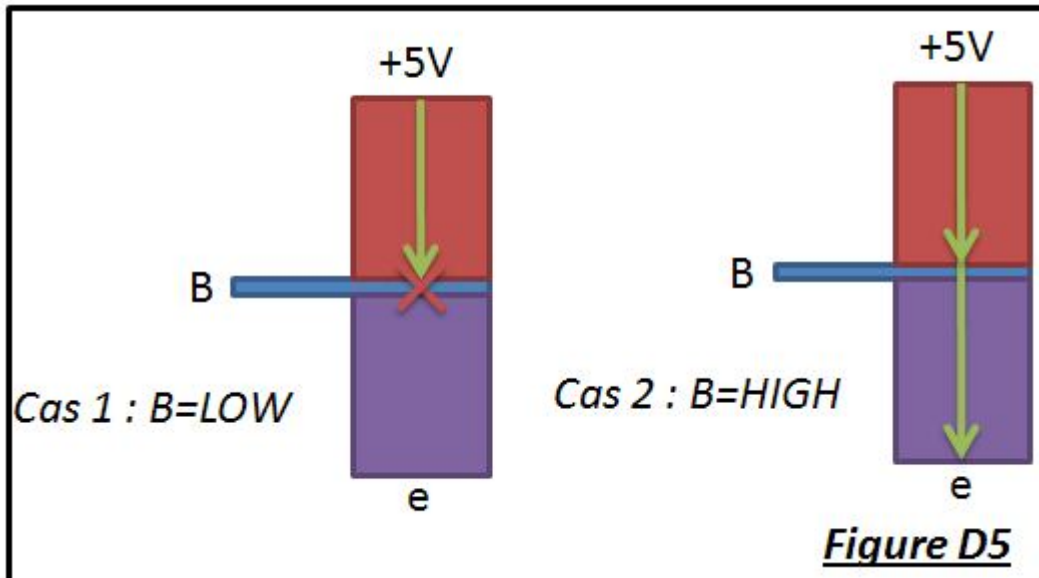
- +Si c'est le capteur droit qui détecte la ligne, cela signifie qu'il est en train de dévier par la gauche il doit donc effectuer une rotation à droite.

- +Si c'est le capteur gauche qui détecte la ligne, cela signifie le cas contraire : il dévie par la droite et doit donc effectuer une rotation à gauche.

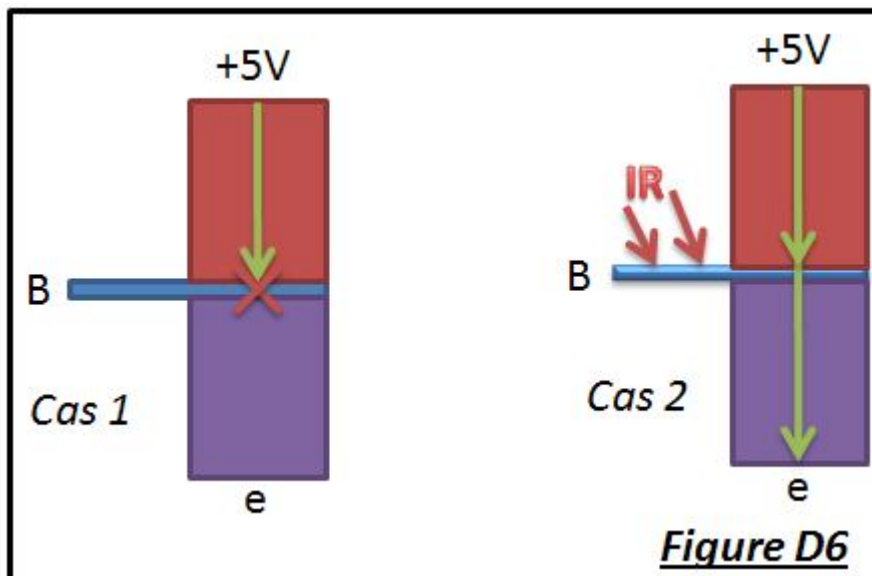
Mais contrairement au cas précédent la rotation n'est ici pas contrôlée par le capteur C, qui détecte déjà la ligne, mais par le capteur de précision dC. Le robot tourne donc tant que le capteur dC ne capte pas la ligne. Puis s'arrête une fois la ligne détectée. A présent l'ajustement par rapport à la ligne terminé, le programme cède la main au programme de traitement de ligne pour continuer à suivre le reste de la ligne.

### III - Les composants électroniques

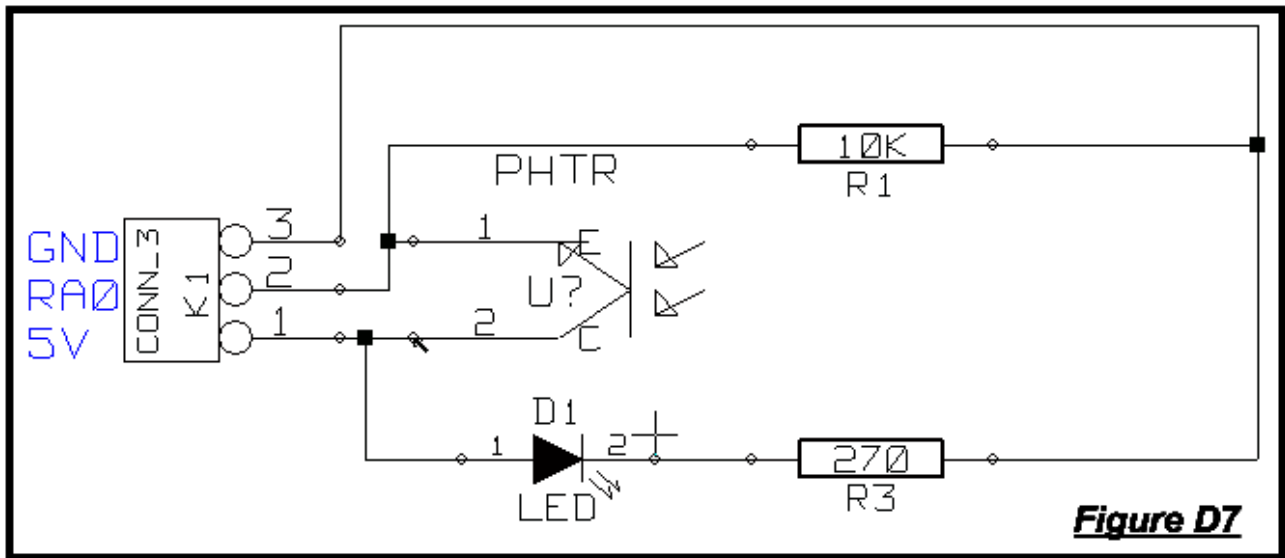
#### A - Le photo-transistor



Comme son nom l'indique, c'est un transistor (Figure D5) dont la base est pas stimulée non pas par un signal électrique, mais par une réception d'un faisceau lumineux. Contrairement aux photo-diodes ou diodes ordinaires, celui-ci est très sensible aux rayons IR. Leur fonctionnement est illustré sur la figure ci-dessous (Figure D6)



Comme pour les transistors normaux ces photo-transistors ont aussi physiquement une base qui n'est pas reliée à la masse mais qui doit rester à l'air. Cette patte du composant pourtant n'est pas représentée sur son symbole normalisé. On obtient alors avec la DEL IR un environnement spécifique à ce photo transistor (Figure D7)

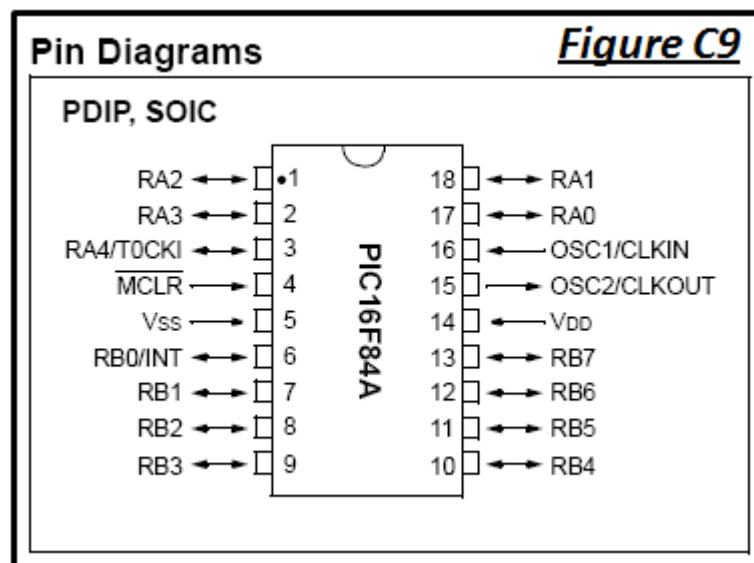


**Figure D7**

Dans la figure D7, la sortie "RA0" correspond à la donnée qui devra être traitée par le PIC, "GND" à la masse, et "5V" à l'entrée de 5V permettant d'alimenter en dérivation la DEL IR et le photo-transistor au niveau du collecteur.

## B - Le contrôleur programmable intelligent 16F84 :

Comme nous l'avons précisé dans le paragraphe II-A, nous avons besoin d'un processeur afin qu'il puisse exécuter les algorithmes que nous avons conçus. Comme nous n'avons pas un programme trop complexe, nous avons choisi le PIC 16F84, ou en anglais 'Programmable intelligent controller 14-bit flash-program-memory (84:ref)'. On trouve d'autre part de nombreux tutoriaux sur Internet expliquant simplement et de manière détaillée son fonctionnement (cours de Bigonoff) ou la datasheet du composant de Microchip (R) pourrait suffire.

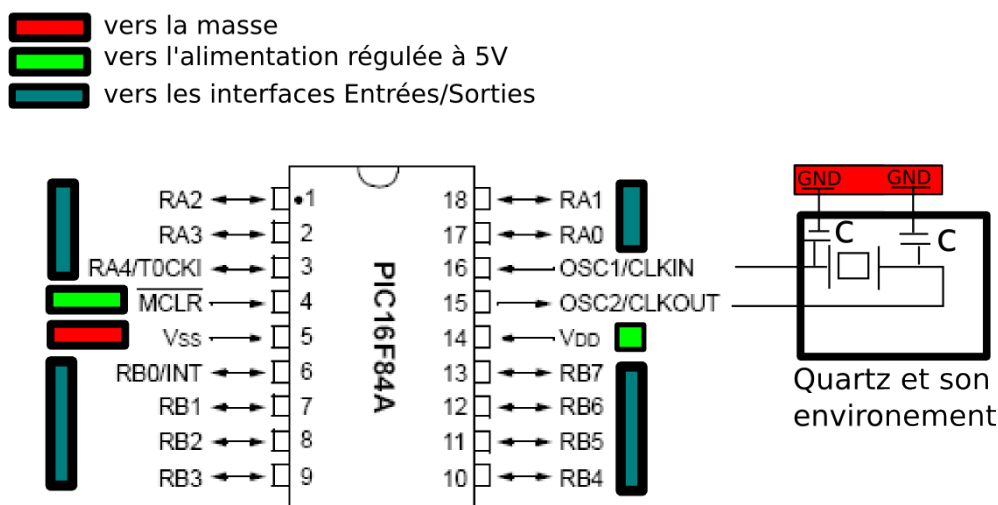


**Figure C9**

*Figure représentant la disposition des pattes sur le CI*

### Description de la fonction de chacune des pattes du PIC (voir Annexes)

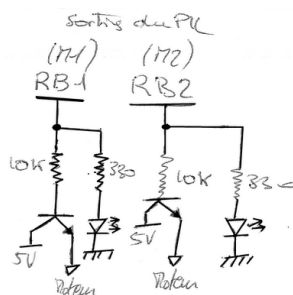
Comme on peut le voir sur les deux figures ci-dessus, le PIC doit être installé dans un certain environnement. En effet il dispose de broches qui lui permettent de recevoir ou d'émettre des signaux numériques (Pins RA0->RA4 et RB0->RB) ; des broches qui lui fournissent une alimentation régulée à 5V pour qu'il puisse fonctionner ; une broche MCLR qui doit toujours être alimentée pour que le programme et les données enregistrées dans sa RAM (mémoire vive interne du processeur) ne puissent pas être remis à zéro ; et deux broches notées OSC1/OSC2 ou CLKIN/CLKOUT qui doivent être cadencées par un quartz dont la fréquence maximale peut-être 20MHz mais par habitude on choisit 4MHz. Ce montage traditionnel, est schématisé sur la figure 9.



*Figure 9 : montage de base du PIC 16F84A*

## **C - Le L298 : commande des moteurs**

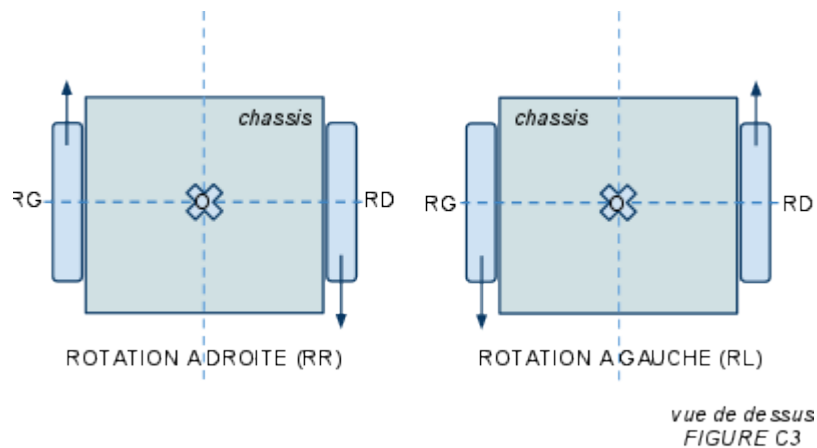
Lors de notre TPE, nous avons utilisé un simple système de relais et d'amplification grâce à un transistor typique (2N222). Ce système nous permettait dans un premier temps d'amplifier le signal émis par le PIC après son traitement. Puis en n'utilisant que ce système de transistor (Figure 10), nous avons rencontré de nombreux problèmes de puissance et les moteurs présentaient même une faible tension à leurs bornes lorsque le PIC n'émettait aucune tension.



*Figure 10 : jeu des transistors permettant d'amplifier le signal émis par le PIC*

Nous avons donc, sous les conseils d'un technicien en électronique (Youssef), utilisé un système de relais (composants dont un interrupteur physique influencé lorsque l'on alimente une bobine électromagnétique). Ce système s'est avéré efficace. Mais le problème c'est qu'il prend bien trop de place entre la carte des relais et celle de l'amplification, sachant que la taille des relais n'est pas à négliger.

Nous avons donc cherché un composant qui se présenterait sous la forme d'un circuit intégré et qui pourrait jouer ces deux rôles. Il ne faut pas oublier qu'un jeu de deux relais ne nous permet de contrôler que l'arrêt et la marche de deux moteurs, et ne nous permet pas d'inverser le sens de rotation des moteurs. Or comme il a été vu précédemment, notre nouveau système de capteurs est basé sur des rotations par rapport à un centre de rotation spécifique (Figure 8). Il est donc indispensable d'utiliser pour une rotation, non pas un système simple dans lequel on bloque une roue, mais plutôt un système similaire dans lequel cette roue devrait changer de sens de rotation (FIGURE C3).



Nous avons finalement réussi à trouver un circuit répondant à tous nos besoins et disponible dans un point de vente de matériel électronique à proximité. Le fonctionnement de ce circuit intégré est expliqué sur la figure 11 (source : datasheet du L298).

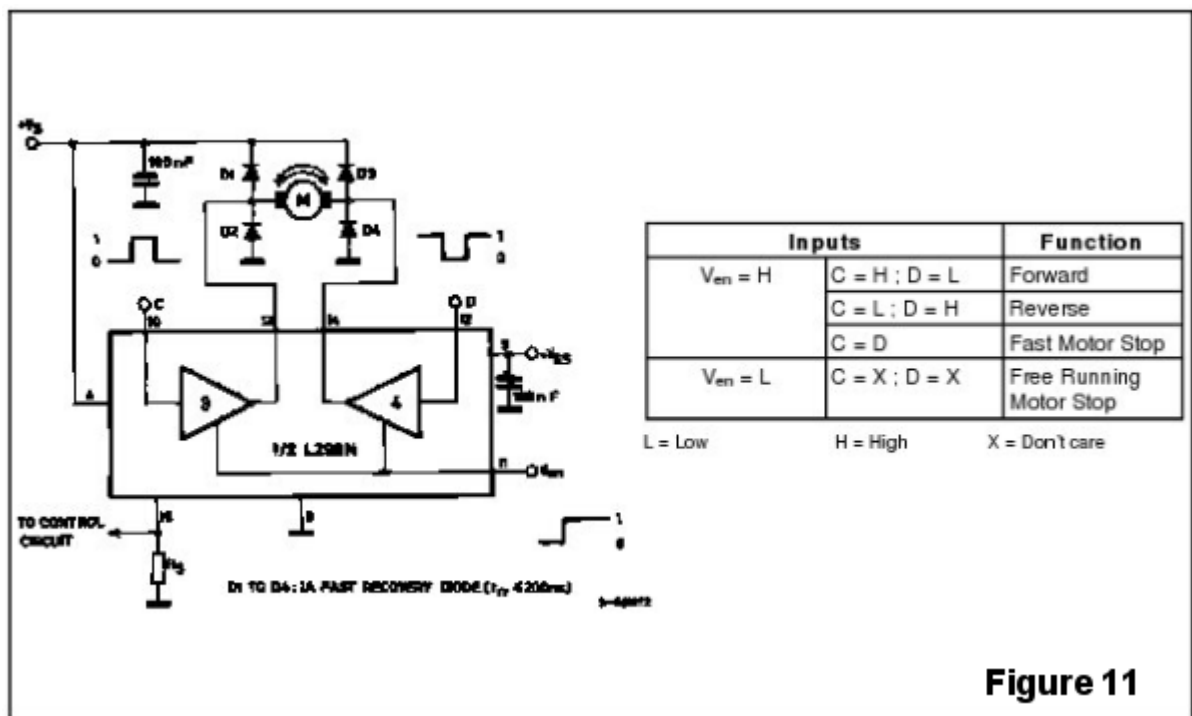
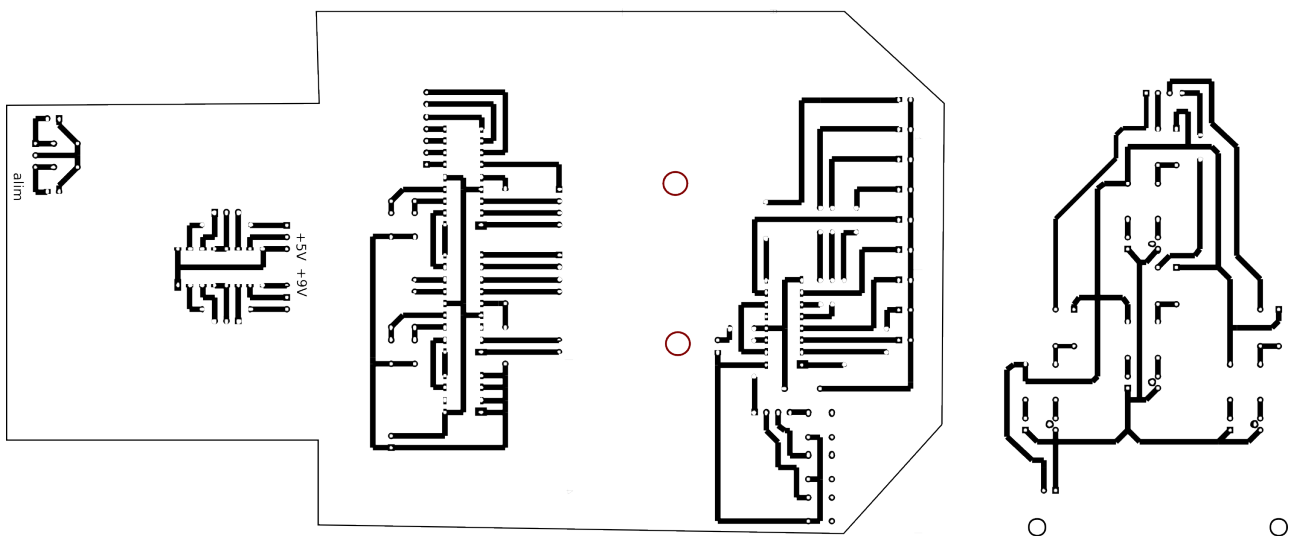
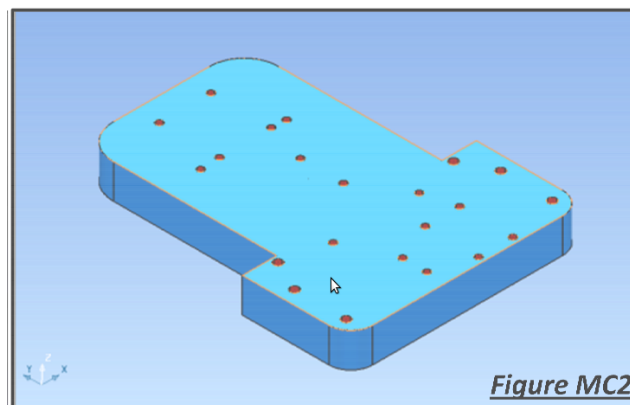


Figure 11

## IV - Réalisation pratique

### A - Structure générale

Nous avons lors de notre TPE, avant de réaliser notre robot, réalisé une maquette de ce dernier (Figure MC2). Pour notre projet d'amélioration nous avons décidé que le robot ne sera constitué que de deux plaques pour des raisons pratiques (Figure 12). L'une recouvrera toute la surface supérieure du robot : c'est la plaque principale ; et l'autre sera celle où l'on disposera les capteurs comme ils ont été schématisés sur la figure 8.



*Figure 12 : disposition des plaque*

*De gauche à droite : la plaque d'alimentation, la plaques du contrôle des moteurs, la plaque de commande (principale = centre de traitement des données), La plaque d'affichage, la plaque où sont disposés les capteurs et les émetteurs IR.*

Les schémas électroniques d'où résultent ces typons sont en annexes.

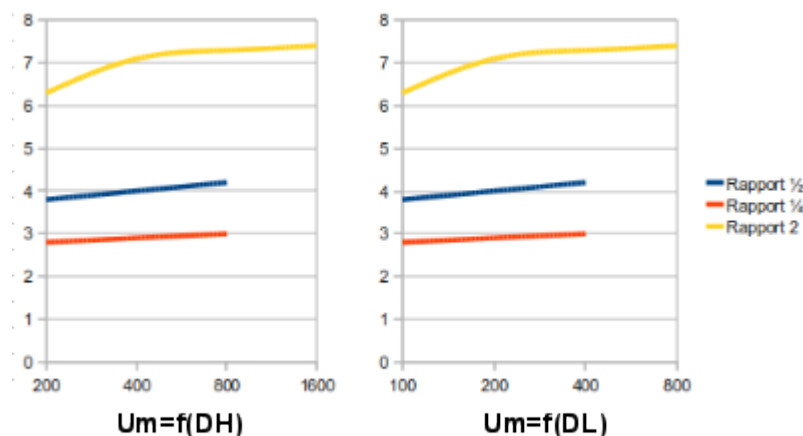


## B - PWM ou Pulse Width Modulation

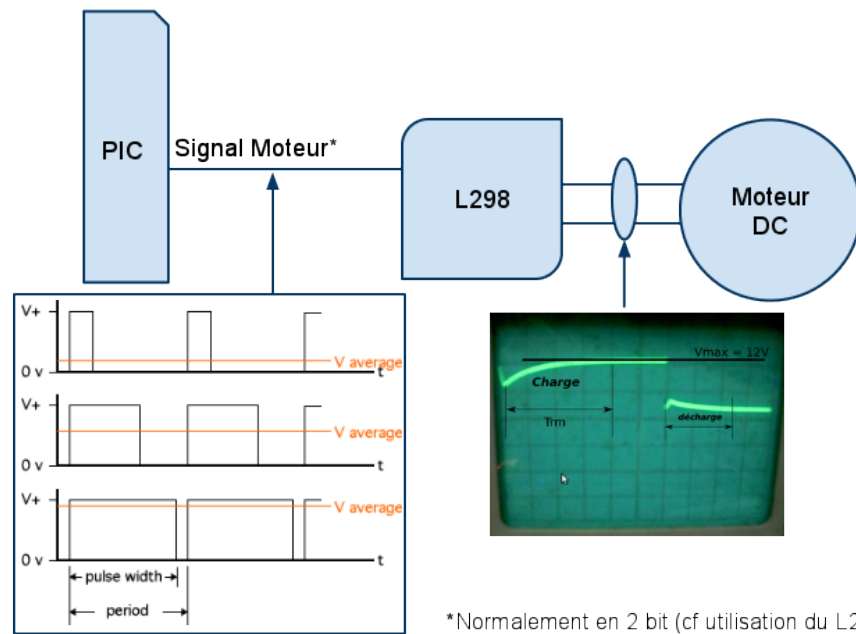
En simulation sur Ktechlab, le programme marche parfaitement : les quatre routines a\_INIT, a\_TDL, a\_SVL et a\_OPT (cf. Annexe ou Site) se relayent correctement et traitent de manière univoque les données testées des capteurs selon les derniers algorithmes (correctif n°2). Mais en pratique, les choses se sont avérées très différentes : le robot balaye en quelque sorte le sol pour détecter la ligne et la suivre. En effet, le robot suit la ligne mais les ajustements qu'il effectue sont tels que le temps entre la détection de la ligne et la réaction du moteur est suffisant pour que le robot sorte de la ligne : son énergie cinétique lors de ces rotations est telle que l'arrêt simple des moteurs n'est pas suffisant pour stopper cette rotation. Il est donc nécessaire de réduire la vitesse du robot.

Pour cela nous avons utilisé la méthode "Pulse Width Modulation" ou "Modulation de largeur d'impulsion". Cette méthode consiste, tout en gardant la même tension instantanée aux bornes des moteurs DC (à courant continu), à faire varier la durée des impulsions émises par le PIC dans le but de modifier la tension moyenne aux bornes du moteur. Cette dernière peut ainsi être réduite, donc par conséquent sa vitesse de rotation moyenne, tout en gardant un couple suffisant pour mettre en mouvement le robot.

Afin de tester l'efficacité de cette méthode, nous avons réalisé une série d'essais avec plusieurs rapports de modulation :



$U_m$  (V) Correspond à la tension moyenne mesurée aux bornes du moteur,  $DH$  ( $\mu s$ ) correspond à la durée d'une impulsion,  $DL$  ( $\mu s$ ) celle d'un état bas, et le rapport  $R$  correspond au rapport  $\frac{\Delta t_H}{\Delta t_L}$ . De manière générale  $U_m \approx U_{th}$  à  $10^{-1}$  près. On peut expliquer cet écart car le signal aux bornes du moteur dépend de son inductance. Ainsi, on observe sur un oscilloscope, un temps de "charge" moyen  $\Delta t_{rm} = 400 \mu s$  (Figure P1) qui modifie la moyenne  $U_{th}$  théorique déterminée par l'équation simplifiée  $\frac{12 \cdot dH}{dH + dL} = U_{th}$ .

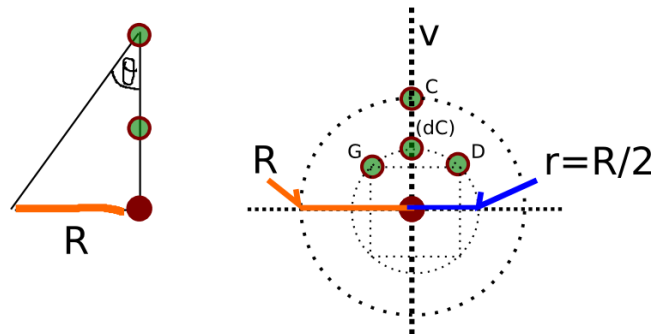


### Résumé de la méthode PWM (Figure P1)

Enfin nous avons déterminé pour chaque mouvement du robot (en 3 essais, cf. videos) un rapport tel que le suivi de ligne soit le plus précis.

	Rotation à gauche	Rotation à droite	Avance
$R$	2	1	5
$\Delta t_H$	200	100	500
$\Delta t_L$	100	100	100
$U_m$	6.3	$\approx 5$	$\approx 9$

## C - Spécifications techniques sur les capteurs



*Figure 13 : disposition des capteurs et distances clés*

Définition de  $r = 3cm$  et  $R = 2r = 6cm$ . Ces mesures ont été obtenues à partir du logiciel de dessin vectoriel utilisé et pour une épaisseur de ligne de 5mm (correspondant à la largeur d'un photo-transistor). Pour les obtenir, nous avons réalisé plusieurs modèles du même prototype que nous avons simulés puis comparés. Les résultats obtenus correspondent au modèle qui a donné les meilleurs résultats.

Calcul de l'angle minimal entre deux directions d'un virage (direction d'entrée et de sortie) :

$$\tan \theta = \frac{R}{2r} = 1 \text{ d'où } \theta = \frac{\pi}{4} = 45^\circ.$$

## **Conclusion**

Dans cette seconde phase d'optimisation de notre projet de TPE, nous avons donc mis en place et défini un nouveau système, plus performant et optimal pour la détection de tout type de ligne, même si ce dernier a nécessité une vitesse limitée. A présent nous pouvons nous demander quelles seraient les applications industrielles de ce système de suivi de ligne et quelles seraient les nouvelles optimisations pour le rendre autonome au sein de réseaux de transports routiers complexes, par exemple dans un port ou un grand hangar.

## **Remerciements :**

Nous remercions tous ceux qui nous ont, de près ou de loin, aidé à réaliser notre projet. Nous tenons plus particulièrement à remercier Florent Touchard, Jérôme Monceau et Youssef Messaoudi pour leur disponibilité et leurs précieux conseils.

Mais nous remercions aussi et surtout Kelian Aicardi et Hicham Rhouni, qui n'ont pas pu participer à ces olympiades de la physique, mais qui faisaient partie de notre groupe de TPE l'année dernière.