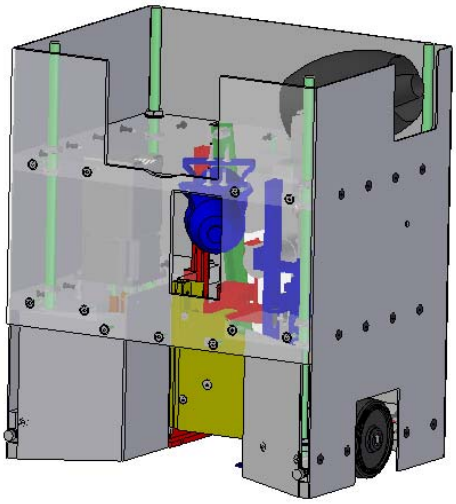
FACULTE POLYTECHNIQUE DE MONS

# Projet de Construction Mécanique

## Conception et fabrication d'un robot autonome en vue de la participation au concours Eurobot 2003



**Andrieu Cédric**
**De Arizon Jacques**
**Deplus Steve**
**Philipront Nicolas**
**Rivière – Lorphèvre Edouard**
**Van Wijnsberghe Frédéric**
5e Mécanique

Année Académique 2002-2003

# Sommaire

# Remerciements

Nous tenons à remercier

M. le Recteur Serge Boucher et M. le Doyen Jean Hanton pour leur financement ;
M. Le professeur Pierre Dehombreux, pour le suivi du projet;
M. Le professeur Enrico Filippi, pour le suivi du projet;
M. Le professeur Marc Delhaye, pour l'aide apportée dans le domaine électrique, dans le choix des moteurs et dans le design de la carte de puissance;
M. Le professeur Olivier Verlinden, pour ses conseils lors des réunions;
M. Pascal Repjuk, de la société Capflow, pour son implication dans le projet ;
Messieurs les assistants du service de Génie Mécanique, Raffaele Papantonio, Vincent Stalon et Olivier Basile, pour leur aide de tous les jours dans la réalisation de ce projet ;
M$^{elle}$ Antonella Contino, pour ses conseils et sa participation aux réunions ;
M. Pierre Lecomte, pour l'aide apportée au choix des capteurs et au développement de la CMUCam ;
M. Johan Dechristophoris, pour l'aide apportée à la réalisation des cartes électroniques et son aide lors des concours belge et européen ;
M. Vergari Laurent, pour l'aide à la fabrication de la structure du robot et son soutient moral tout au long de l'année;
M. Vincent Gaudissart, pour son aide dans le développement de la caméra Capflow ;
M. Laurent Pinchart, pour son savoir-faire informatique et sa motivation ;
Nos supporters et pom-pom girls, pour leurs encouragements ;
Messieurs Rossor et Zonta du service UAVM pour les enregistrements vidéos.

Toutes les personnes qui ont participé de près ou de loin à la réalisation du robot.

# 1. Introduction

## 1.1 Présentation du concours

EUROBOT, challenge technique, scientifique et ludique organisé en France, s'adresse aux étudiants d'écoles d'ingénieurs et d'universités, ainsi qu'aux clubs scientifiques indépendants. Ce concours oppose des robots totalement autonomes.
Le règlement de la compétition change chaque année et cela afin de donner toutes leurs chances aux équipes nouvelles, mais également dans le but de stimuler celles qui y ont déjà participé.
Néanmoins quelques constantes demeurent. Ainsi, il s'agit de matchs durant environ 1min30 entre robots totalement autonomes. Les robots, d'une taille d'environ 30-40cm de côté, se déplacent dans une aire de jeu d'à peu près 3m x 2m.
Seuls 3 robots par pays peuvent participer à Eurobot. Si un pays a plus de 3 équipes, des qualifications nationales sont alors organisées auparavant, comme c'est le cas déjà pour la France, la Suisse, l'Espagne, la Yougoslavie et depuis l'année 2002, la Belgique. Les 3 équipes qualifiées pourront alors participer à Eurobot avec les autres pays concurrents.

**Règlement 2003**
Les robots ont une minute et trente secondes pour faire sauter la banque. L'objectif est de placer un maximum de palets sur une couleur. Tous les moyens sont bons : lancer des projectiles, attraper les palets, les pousser, les empiler… L'aire de jeu comporte des palets bicolores (rouges et verts) disposés sur le plateau ainsi que des palets unicolores verts ou rouges placés en hauteur. Le robot qui part du côté noir (placé sur une case noire) doit faire apparaître le maximum de faces vertes des palets. Le robot partant du côté blanc (placé sur une case blanche) doit quant à lui faire apparaître le maximum de faces rouges.

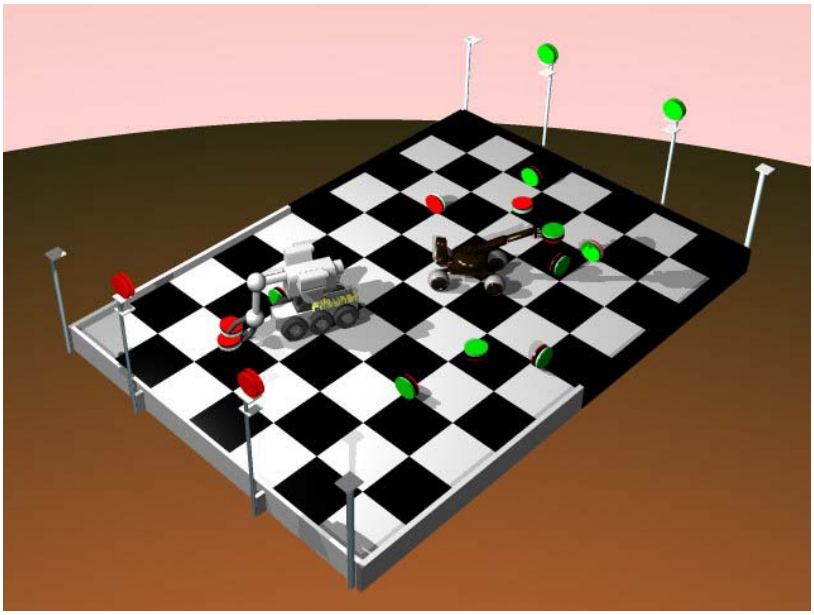


**Figure 1: Règlement 2003**

Le règlement complet est présenté en annexe, toutefois nous le présenterons ici dans les grandes lignes.

**Le Robot**

Un robot est une machine totalement autonome, emportant sa propre source d'énergie, ses actionneurs et ses systèmes de commande.
Les restrictions de sécurités découlent du bon sens, à savoir qu'il ne faut pas que le robot abîme intentionnellement le robot adverse ou l'aire de jeux. Il ne doit pas être dangereux pour les personnes environnantes. Ainsi, il ne peut comporter de parties saillantes ou pointues. De plus, il ne doit pas leurrer l'adversaire.
Un robot ne doit pas avoir été conçu pour transporter simultanément plus de 4 palets.

**Contraintes dimensionnelles**

Les contraintes quant aux dimensions sont plus strictes que l'année précédente, elles sont passées de 130cm de périmètre à 120 cm cette année[1] pour 40cm de hauteur (identique à l'année précédente). Un effort supplémentaire de miniaturisation aura été nécessaire cette année.

**Les cibles**

De part et d'autre de l'aire de jeu sont disposées des cibles[2] constituées de palets unicolores posés verticalement sur des supports. 2 palets verts sont placés du côté noir de l'aire de jeu et 2 palets rouges du côté blanc. Chaque cible peut être renversée par un projectile[3] afin d'être mise en jeu. On peut également utiliser une soufflerie pour faire tomber les cibles.

**Décompte des points**

Un palet couché sur la bonne couleur vaut un point. 2 palets empilés valent trois points. 3 palets empilés valent quant à eux cinq points. Seul l'état final des palets importe dans le décompte des points. Sont considérés comme valides à la fin du match les palets présents sur la table, strictement à l'extérieur du périmètre défini pour les robots et dont les faces sont strictement horizontales.

**Homologations**

Pour participer aux phases qualificatives, un robot doit être soumis au contrôle d'un arbitre qui vérifie si tout est en ordre. Un match a ensuite lieu. Le but est de vérifier si un robot peut gagner un match seul. Trois tentatives sont permises.

**Déroulement d'une partie**

Une couleur (noir ou blanc) est attribuée juste avant la partie. Les équipes disposent de trois minutes pour placer leur robot à son emplacement de départ. Les robots disposent d'1 minute et 30 secondes pour retourner le plus de palets sur la couleur défendue, et ceci de manière strictement autonome. En aucun cas, il n'est permis aux participants de toucher aux robots, aux palets, aux balises et à l'aire de jeu durant le match. Les palets qui sortent de l'aire de jeu n'y sont pas remis. Un robot n'ayant pas entièrement traversé au moins 2 carreaux du damier de l'aire de jeu à la fin d'une partie est déclaré forfait.

[1] Il est question du périmètre de l'enveloppe convexe englobant la projection verticale du robot au sol, c'est-à-dire que le chemin le plus court pour faire le tour de l'ombre du robot sous lumière verticale ne doit pas excéder 120 cm.
[2] Voir figure 1
[3] Les seuls projectiles valables sont des balles de tennis de table agréées par l'ITTF (International Table Tennis Federation)

## 1.2 Sponsoring

Cette année, un effort a été consacré à la recherche de sponsors. Malheureusement à cause de la mauvaise situation économique actuelle et de la faible connaissance du concours par les industriels, seuls deux sponsors ont décidé de participer à notre aventure.
Ces deux sponsors sont les entreprises Capflow et Farnell.
Le partenariat avec l'entreprise Capflow a permis le développement d'une caméra permettant au robot de repérer les palets sur la table. Par manque de temps de test avant le concours belge, Capflow s'est proposée, de continuer le développement de la caméra et de la commande du robot en vue du concours européen à la Ferté Bernard.

E-mail:www.capflow.com
Contact: Pascal Repjuk
Tel :0476 27 18 12
Cette collaboration s'est avérée fructueuse et prometteuse pour les années à venir.

L'entreprise Farnell quant à elle nous a offert une réduction de 15 % sur le prix d'achat de ses produits. (sous réserve d'un montant total de 2500 €)


E-mail : eric.goerres@farnellinone.be
Contact : Mr Goerres

Nul doute qu'avec la croissance du concours belge et la réussite de l'équipe, un plus grand nombre de sponsors décidera de collaborer avec nous.

## 1.3 Site internet

En vue de présenter nos activités à l'extérieur et d'offrir une visibilité supplémentaire pour nos sponsors, nous avons décidé de réaliser notre propre site internet. Celui-ci est accessible à l'adresse suivante : www.robot.fpms.ac.be .
Il reprend principalement :

- Une présentation de la coupe de robotique
- La présentation de l'équipe de cette année. Cette section inclut également des images des prototypes réalisés en cours d'année, une présentation du robot et un résumé de la coupe de Belgique de robotique
- Un espace présentant nos sponsors avec des liens vers leur site internet. Un dossier de presse destiné aux éventuels nouveaux sponsors intéressés est également disponible en ligne.
- Des archives présentant le déroulement du concours de l'année passée.

Ce site internet a été crée au cours de cette année et son développement est en perpétuelle évolution. Il s'agit en fait d'une base pour les années suivantes. Il permettra de garder une trace des différentes équipes et des divers robots réalisés.

# 2. Réunions

## 2.1 Organisation des réunions

Les réunions ont une importance capitale pour l'évolution de projets ayant l'ampleur de la construction d'un robot. De telles réunions permettent de montrer à tous l'évolution du projet et de soulever les problèmes urgents à résoudre. Les réunions doivent permettre à tous de s'exprimer tout en limitant la durée. Celles-ci se sont déroulées hebdomadairement depuis le début de l'année. L'heure, la date et le lieu de réunion étaient rappelés à tous par courriel. Les réunions étaient divisées comme suit. Premièrement l'ensemble des problèmes n'ayant rien avoir avec la construction du robot étaient envisagés ce fut le cas de la construction de la table de jeu, du sponsoring,… . Chaque étudiant prenait ensuite la parole un à un pour exposer l'évolution de chacune des tâches qui leur était attribués précédemment. A la fin de chaque élocution ou présentation Powerpoint, les questions étaient soulevées et les décisions prises en fonction de l'avis de chacun. Les tâches étaient enfin réparties à l'ensemble de l'équipe. Le retour des réunions sous forme de procès verbaux clairs et concis constituait l'un des aspects les plus important, chaque réunion doit en effet laisser une trace des décisions qui ont été prises et permettre aux gens qui n'ont pas pu assister aux réunions d'être informé des points soulevés.

## 2.2 Répartition des tâches

La construction d'un robot constitue un travail énorme et demande une bonne coordination de l'ensemble de l'équipe. Chaque étudiant est responsable d'une section technique répartie en début d'année :
Cédric Andrieu : Vision
Jacques De Arizon : Electronique de puissance
Steve Deplus : Programmation
Nicolas Philipront : Electronique de commande
Edouard Rivière-Lorphèvre : Mécanique générale
Fredéric Van Wijnsberg : Production Mécanique

Certes les tâches sont clairement réparties mais il est impossible de mettre la charrue avant les bœufs. Chaque tâche constitue un travail énorme mais certaines demandent pour pouvoir être lancées que d'autres soient réalisées. Chaque tâche a donc été classée et réalisée selon son urgence par la personne concernée aidée par d'autres membres de l'équipe. Ainsi les idées relatives par exemple au concept du robot ont été réalisées par l'ensemble de l'équipe, et on peut dire que la réalisation mécanique proprement dite ainsi que la programmation sont l'œuvre de l'équipe au complet. Soulignons encore dans ce point l'apport déterminant des assistants et techniciens dont la disponibilité n'a d'égal que la bonne ambiance qui régnait au sein de ce projet.

Le robot constitue donc un réel travail d'équipe. On peut dire que l'équipe s'est investit corps et âme pour représenter dignement et avec succès la Faculté Polytechnique de Mons. L'apport d'un tel projet dans la formation d'un Ingénieur Civil Mécanicien est donc énorme tant au point de vue technique que relationnelle.

La réalisation technique d'un projet engendre des tâches non techniques. Ces tâches ont également été traitées par les étudiants.
Ces tâches non techniques ont été réparties de la manière suivante :
Responsable de projet : Nicolas Philipront
Secrétaire : Edouard Rivière-Lorphèvre
Sponsoring : Jacques de Arizon
Contact extérieur : Cédric Andrieu
Analyse du règlement : Steve Deplus
Gestion de production : Frédérique Van Wijnsberghe

C'est la flexibilité et l'ardeur au travail de chaque membre de l'équipe (étudiants et assistants) qui nous a permis de réaliser le parcours qui fut le nôtre lors de la coupe de Belgique et le concours Eurobot.

# 3. Mécanique

Au vu du règlement de cette année, il est apparu clairement que les divers robots pouvant participer à cette coupe de robotique devraient pouvoir réaliser l'une ou plusieurs des opérations suivantes :

Coucher les palets mis sur la tranche
Emmagasiner un palet dans le robot
Retourner les palets posés sur la mauvaise couleur
Empiler les palets
Casser les piles de l'adversaire
Faire tomber les palets unicolores

La première partie du travail consistait à imaginer l'ensemble des systèmes pouvant remplir ces diverses fonctions. Ce travail de réflexion (et de réalisation de divers prototypes) s'est déroulé durant le premier semestre de l'année. Toute l'équipe a réfléchit aux diverses façons de réaliser les fonctions citées précédemment. Le but était d'avoir le choix le plus large possible et le mieux documenté pour choisir le ou les systèmes qui seront définitivement implantés sur le robot.
Partant du principe que ce projet est à la base un projet de mécanique, nous avons tous insisté sur le fait qu'il était souhaitable de réaliser un robot pouvant remplir l'ensemble des tâches précitées. Nous avons donc privilégié l'aspect constructif (système mécanique performant et polyvalent) à l'aspect compétition pure (un petit robot rapide mais très fiable aurait été alors plus intéressant).
Nous allons donc présenter, poste par poste les diverses idées envisagées, avec leurs avantages et inconvénients. Ensuite, nous présenterons plus en détail la solution définitive retenue ainsi que les diverses modalités de réalisation du robot

## 3.1 Diverses solutions envisagées – Prototypes

### 3.1.1 Systèmes permettant de coucher les palets posés sur la tranche

C'est ce système qui a été étudié en premier. En effet, il s'agit de l'opération de base que devra effectuer le robot. Une petite étude sur le palet, nous a permis de déterminer l'angle minimal pour pouvoir faire tomber un palet vertical de façon statique. (De façon à ce que la projection du centre de gravité sur la table se trouve en dehors de la surface de sustentation du palet). Cet angle est de 76 degrés entre le plan d'une des flasques et le plan de la table. De même, nous avons déterminé les enveloppes du contour du palet dans leur mouvement dans divers cas comme par exemple contre le bord de la table. Ceci a été fait de façon à déterminer la place que prend le palet dans son mouvement. Plusieurs principes ont été envisagés :

Le pare-chocs simple : une simple barre fixe située à l'avant du robot pousse le palet sur sa partie supérieure et le déséquilibre.
Le pare-chocs variable : 2 pare-chocs, un situé sur la partie supérieure et un situé sur la partie inférieure du robot, ces pare-chocs sont mobiles et peuvent être actionnés indépendamment l'un de l'autre.

Le mouvement des pare-chocs est lié un capteur de couleur qui reconnaissant la couleur de la face du palet permet d'actionner soit le pare-chocs supérieur soit le pare-chocs inférieur de façon à faire tomber le palet directement du bon côté.
Le vérin frappeur : il s'agit d'un système à peu près équivalent au pare-chocs variable, avec la différence que c'est le mouvement du vérin et non le mouvement du robot qui fait tomber le palet.

Un des problèmes qui pouvaient se passer avec les systèmes ci-dessus, était le fait que si le palet arrivait perpendiculairement au pare-chocs où étaient frappés perpendiculairement par le vérin, le palet roulait et ne tombait pas sur la table.
C'est pour éviter ce problème que nous avons envisagé les systèmes mécaniques suivant :
La moissonneuse : c'est un système identique au système que l'on retrouve à l'avant de moissonneuse batteuse afin de redresser les pailles avant de les engloutir dans la machine. Il est constitué de diverses tiges excentrées de par rapport à un axe. Cet axe tourne de façon à ce que la partie inférieure soit dirigée vers le robot. Dans ce cas, les tiges prennent les palets entre les flasques et les font tomber. Pour éviter les chocs sur le système, nous avons remplacé les tiges rigides par des élastiques.
Un prototype a été construit et il est apparu que dans certains cas, le palet pouvait rentrer dans le robot sans avoir été renversé. Ce système ne convenait donc pas.

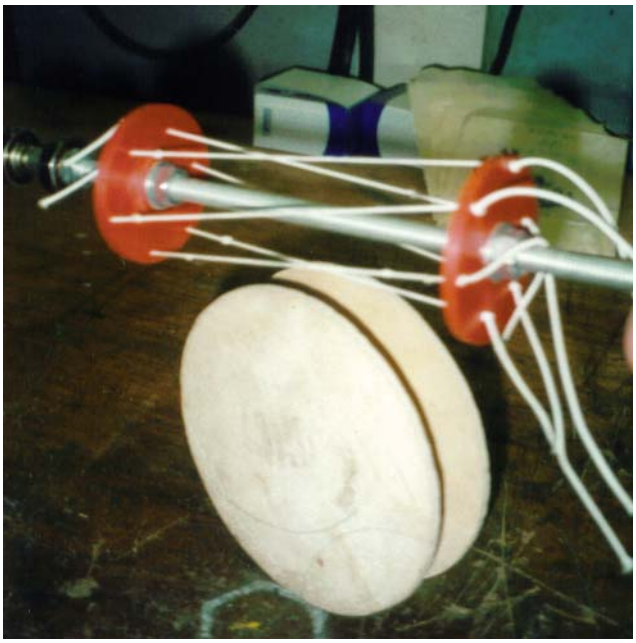


**Figure 2 : prototype du fléau**

Le fléau : ce système ressemble à une brosse circulaire que l'on ferait tourner autour de son axe. Grâce au frottement des poils de la brosse, il était envisagé de faire bouger les palets perpendiculaires afin de les mettre dans une position plus propice à les faire tomber. Il est apparu lors d'essais où l'on a fait tourner la brosse fixée sur une visseuse que les palets pouvaient encore entrer dans le robot sans avoir été renversé.
Le tapis roulant : ce système mécanique est composé d'une bande transporteuse inclinée dont le but est de prendre le palet par ses flasques supérieurs et le faire tomber sur la table. Afin d'améliorer le système nous avons ajouté à celle-ci de petits ergots en caoutchouc afin d'augmenter la préhension.

Ici aussi un petit prototype a été construit et il est apparu que ce système ne permettait pas de régler le problème des palets perpendiculaires qui roulait au contact de la bande.

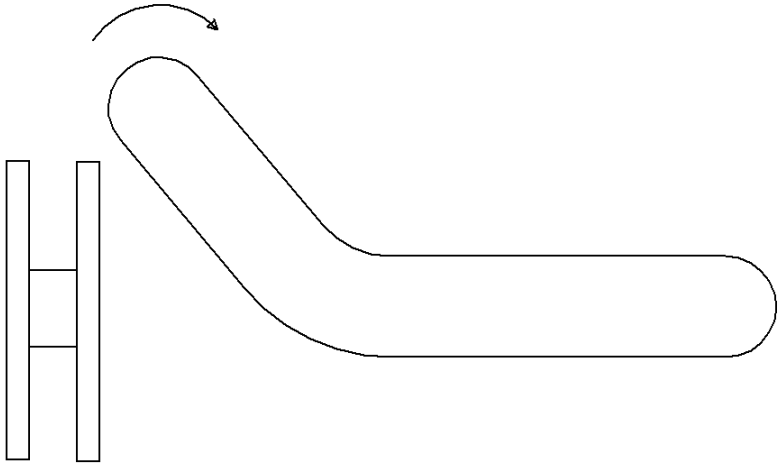


**Figure 3 : principe de fonctionnement du tapis roulant**

Le batteur à oeufs : ce système est constitué de deux barres excentrées de vis-à-vis de leur axe vertical. Ce système donnait des coups répétés sur le palet pour le faire tomber. Ce système avait aussi l'avantage de ramener les palets extérieurs vers le centre du robot. Le prototype construit nous a donné beaucoup de satisfactions et était très efficace. Celui-ci avait néanmoins un inconvénient majeur : il prenait une place très importante dans le robot car on ne pouvait jouer avec la taille des barreaux excentrés. Nous avons aussi envisagé le cas de remplacer les barreaux excentrés par une brosse de nylon mais aucune expérimentation n'a été effectuée.


**Figure 4 : prototype du "batteur à oeufs"**

Hachoir : il s'agissait d'une vis sans fin de pas supérieur à la largeur d'un palet et qui a été coupé longitudinalement sur sa moitié. Ce système permettait de faire tomber les palets grâce à la partie coupée qui venait taper entre les deux flasques et permettait de faire tomber les palets perpendiculaires en les entraînant dans la vis sans fin.

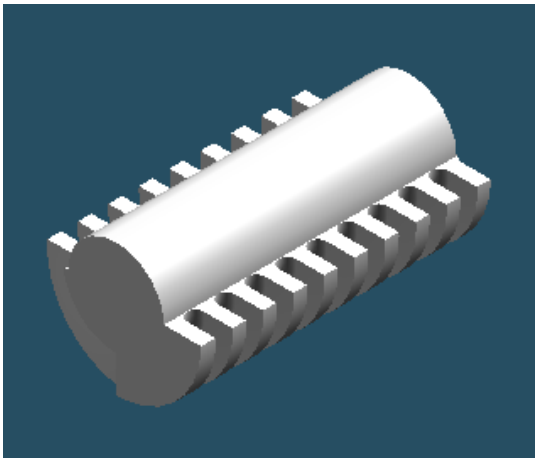


**Figure 5 : "hachoir"**

Nous nous sommes rapidement rendus compte que l'ensemble de ces solutions présentait le grand désavantage de prendre beaucoup de place sur le périmètre très limité du robot. Nous avons donc voulu nous rendre compte de la nécessité d'employer un tel système.
Une expérimentation a été réalisée avec le robot de l'année passée pour visualiser l'effet des collisions sur les palets. Nous avons lancé le robot en mode aléatoire sur la table après avoir disposé les palets comme lors du concours. Nous avons constaté que les chocs à faible vitesse sont capables de faire tomber les palets. Il a fallu que 45 secondes en mode aléatoire pour les faire tomber tous.
Cette constatation a rendu inutile les recherches futures sur un tel système.
Durant la compétition en Belgique, nous avons néanmoins remarqué que certains robots utilisaient des systèmes de ce type, comme par exemple le batteur à oeufs ou la moissonneuse.

D'autres systèmes ont été aussi envisagés sans pour autant être expérimentés ou construits. Parmi ceux-ci nous trouvons :

- Un système pneumatique constitué de deux électrovannes et de deux tuyères qui permettait de lancer un jet de dioxyde de carbone gazeux vers le haut ou vers le bas du palet à fin de le faire tomber directement du bon côté.
- Un système à jet d'air constitué d'une turbine qui souffle les palets devant le robot. Des petits essais effectués avec la turbine du retournement ont prouvé que l'on pouvait faire tomber dès palets à trois mètres et même de faire les piles.
- Un système mécanique constitué d'une bande transporteuse dont le moment est perpendiculaire au mouvement du robot. Cette bande transporteuse étant elle-même fixée sur un système excentrique lui permettant d'osciller. Dans ce cas, les palets perpendiculaires sont entraînés par la bande tandis que les autres sont déséquilibrés par le système d'excentrement.
- Un système de rouleaux tournant à l'avant du robot, les rouleaux étant constitués de mousse empêchait aux palets perpendiculaires de rouler.

### 3.1.2 Systèmes permettant d'emmagasiner les palets dans le robot

Lors des premières réflexions menées sur le robot, il est apparu qu'il serait réellement intéressant de disposer d'un système en continu permettant de traiter les palets à la file. En effet, s'il n'y a pas de délai entre le traitement de deux palets successifs, nous pouvons traiter un plus grand nombre de palets. Pour réaliser ce traitement en continu, nous nous sommes inspirés de l'exemple du robot de l'année passée qui emmagasinait les balles pour ensuite les traiter. Nous avons réfléchi aux diverses façons de prendre un palet à l'intérieur du robot.
Nous voulions créer un système permettant d'augmenter la hauteur du palet : en effet si nous pratiquons la formation de piles par le haut il faudra nécessairement montrer le palet à une hauteur de 60 mm (hauteur d'une pile de deux palets). Dans ce cas le retournement du palet se pratiquerait dans la partie supérieure du robot avant le système d'empilement.
Les diverses solutions envisagées sont :

Le système a bande transporteuse : ce système à été notre premier prototype construit : il s'agissait de deux bandes transporteuse qui prenaient le palet entre les deux flasques et l'amenait à une hauteur suffisante pour pouvoir les empiler.
Ce prototype (dont les plans sont fournis en annexe) s'est relevé peu pratique car l'angle de la bande était déterminé par l'enveloppe du palet dans son mouvement. Il y avait donc risque de coincement du palet sur la bande si la tension des courroies était trop importante. Par contre, une tension insuffisante ne permettait plus d'entraîner les courroies car le frottement sur les poulies motrices n'était plus suffisant.

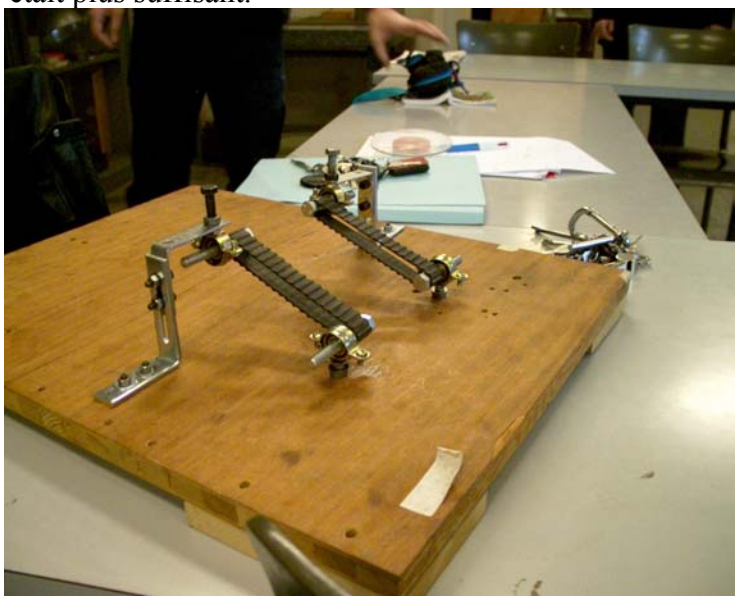


**Figure 6 : Premier prototype de système à bandes transporteuses**

De plus, lors de la réalisation pratique de ce système, il est apparu que le guidage prenait beaucoup de place et que seul un palet correctement placé devant le robot pouvait être pris. Un autre problème est apparu : le frottement des palets avec la table et avec la bande déterminait la bonne préhension de celui-ci. La préhension du palet était donc liée à la grande incertitude entourant les coefficients de frottement. Ce système s'est donc révélé très compliqué à régler et l'entraînement nécessitait une puissance importante. Ce système a donc été abandonné.

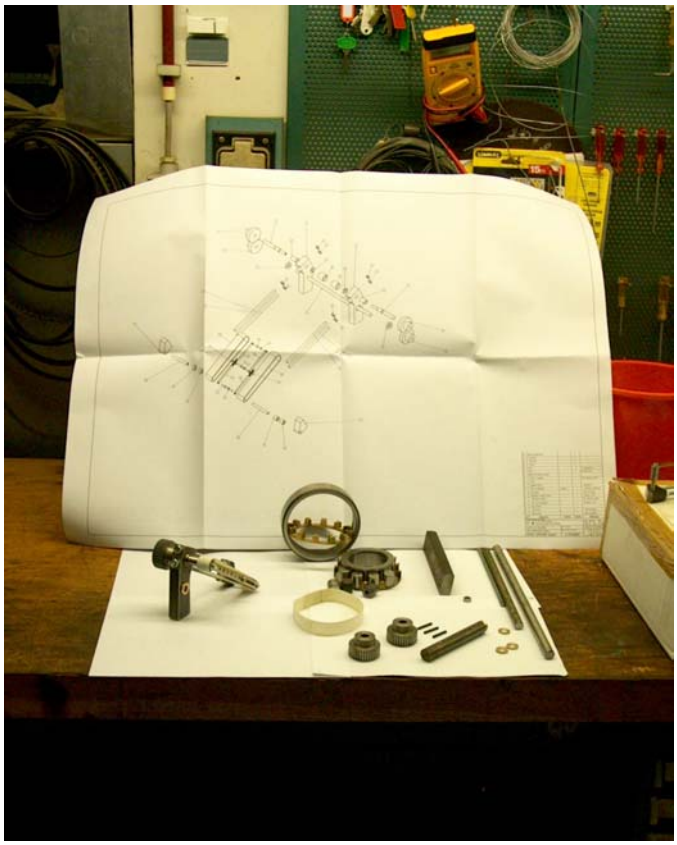


**Figure 7 : Ensemble des éléments ayant servi à la réalisation du prototype**

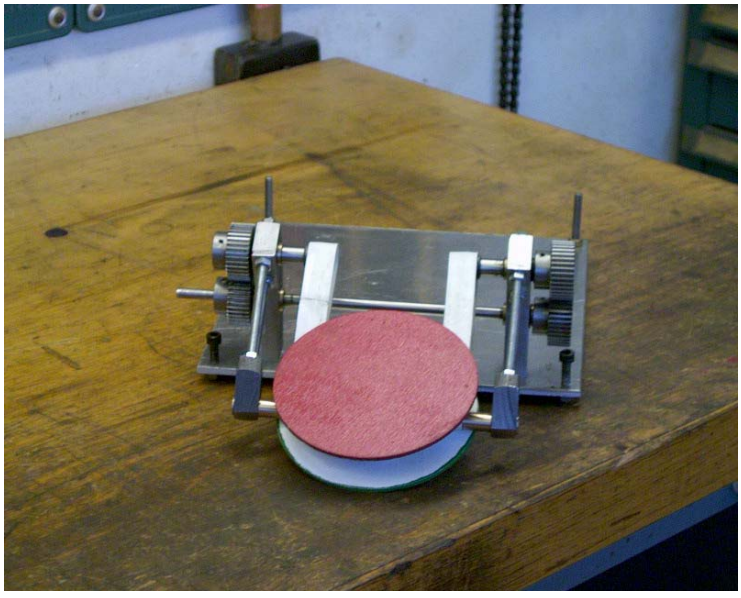


**Figure 8 : Prototype de bandes transporteuses**

D'autres systèmes de préhension du palet par bande transporteuse ont été envisagés. Par exemple, en prenant le palet par le flasque supérieur pour l'accélérer et le lancer sur un toboggan et lui donnant suffisamment de vitesse pour que son inertie puisse le retourner ou le faire monter à une hauteur suffisante.

Vis sans fin : Un système séduisant était l'emploi de vis sans fin contre rotatives. La valeur du pas était étudié pour que le palet se déplace verticalement. Ce système permettait de prendre des palets seuls ou des piles et de les emmener dans la partie supérieure du robot. Ceux-ci auraient été évacués grâce à une came qui guiderait leur mouvement dans la partie supérieure.

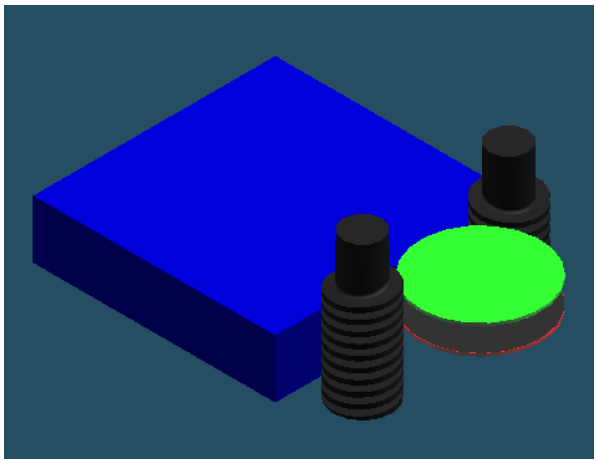


**Figure 9 : Principe de l'emploi de vis sans fin**

Bien que ce système présente de nombreux avantages, sa construction n'a pas été réalisée. En effet, la fabrication des vis sans fin se révélait quasiment impossible sans outillages spécialisés du type tour numérique. De plus, ce système prenait quasiment le tiers du volume du robot.

Tiges filetées

Un système à peu près équivalent, avait été trouvé il s'agissait de deux tiges filetées au pas opposé et contre rotatives. Sur ces tiges filetées se trouvent des boulons soudés à d'autres tiges perpendiculaires à l'axe de la tige filetée. Lorsqu'un palet est détecté, les tiges de filetées se mettent en rotation, les boulons se mettent en rotation jusqu'à ce que les tiges soudées touche le moyeu du palet. Celle-ci pousse le palet contre la butée et lorsque le palet est coincé, les boulons sont bloqués en rotation ce qui leur donne un mouvement de translation verticale. Ce système permet de centrer le palet dans une position précise et le décoincement du palet entraîne son évacuation. Ce système se remet en place en position initiale lorsque les tiges filetées tournent dans le sens opposé. Un des problèmes principaux à la construction de ce modèle et la fabrication de tiges filetées et de boulons au pas gauche. Ceci ne pouvait être construit qu'au tour à métaux. Or l'excès de charges de l'atelier mécanique ne permettait pas de construire un prototype. Ce système a été aussi envisagé comme système d'empilement par le bas. Celui-ci montait le palet supérieur de la hauteur donnée est ensuite par changement de sens de rotation dès tiges lâchaient celui-ci sur le palet inférieur.

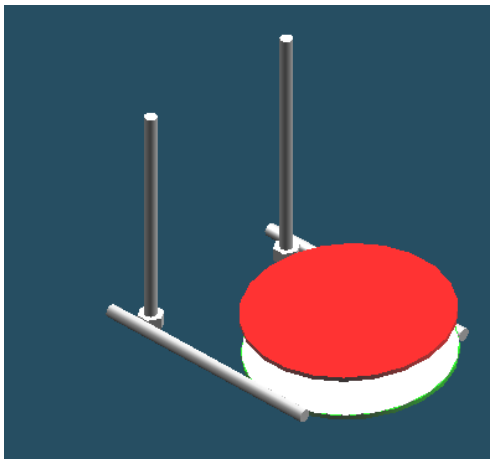


**Figure 10 : tiges filetées, position de départ**

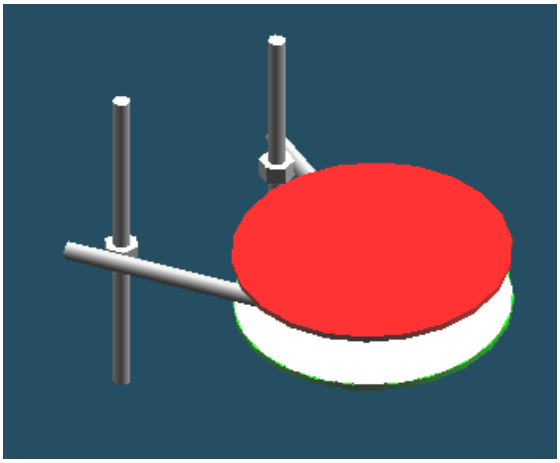


**Figure 11 : tiges filetées, levée du palet**

Système a bande transporteuse verticale : un système composé de deux bandes transporteuse verticale parallèle amenait le palet qui se plaçait entre celle-ci en hauteur. Afin d'améliorer la préhension du palet, la bande transporteuse était équipée d'ergot en caoutchouc.

Toboggan :
Un dernier système envisagé, a été un toboggan qui prenait le palet entre les deux flasques, celui-ci étant poussé par un ergot excentré sur un axe tournant, et monte le palet et lui donner une position verticale. Un petit prototype a été construit mais cela s'est avéré extrêmement encombrant.

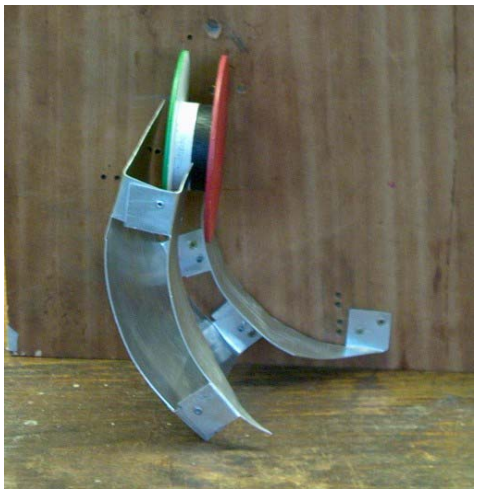


**Figure 12 : toboggan**

Système à pinces : un des premiers systèmes envisagés était le système à pinces. Une simple pince dont les côtés peuvent être ou non mobile prend le palet et éventuellement le retourne. Ensuite le palet est posé au niveau supérieur soit par un système de crémaillère (mouvement vertical et prismatique de la pince) soit par un système de rotation (la pince tourne autour d'un axe est le mouvement de celle-ci l'amène plus haut). Ce système pouvait être soit interne ou externe. Les systèmes externes ont été immédiatement exclus pour des raisons de solidité. Il s'est avéré que le système interne prenait énormément de place. De plus, ce système n'est pas continu et peut donc poser des problèmes lors du traitement de palets successifs.

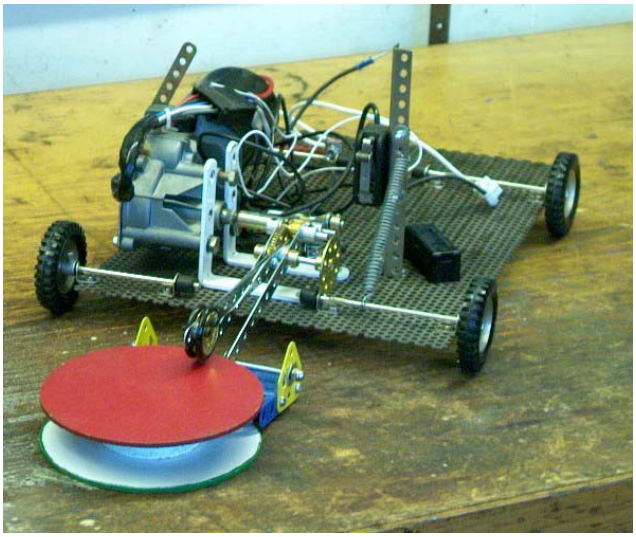


**Figure 13 : Prototype de pince externe**

Par la suite, dans la description des systèmes retournement, nous décrirons la nécessité ou pas d'avoir un système de préhension du palet.

### 3.1.3 Systèmes de retournement de palets

Il est apparu dès le départ que le retournement de palets était un système critique. En effet, il est apparu que très rapidement, un grand nombre de palets tombaient sur la table, et qu'il serait nécessaire de les retourner pour emmagasiner des points.
En étudiant le mouvement de retournement des palets, il est apparu que l'axe de rotation doit se situer obligatoirement dans un plan parallèle aux flasques passant par le centre du moyeu. Il est apparu aussi dès le départ que la que l'enveloppe décrit par le palet lors des mouvements est importante et donc prend une grande place dans le volume du robot. En effet, nous avons décidé, pour des raisons de solidité, de mettre le système de retournement à l'intérieur du robot.
Les systèmes étudiés peuvent se diviser en deux grandes catégories : ceux qui traitent les palets posés sur la table et ceux qui traitent des palets ayant au préalable étés emmagasinés dans le robot.

#### 3.1.3.1 Système traitant le palet emmagasiné dans le robot

Systèmes par gravité
Des systèmes se servant de la gravité ont été envisagés, ceux ci pouvaient marcher de diverses façons : le palet est soulevé avec certaine hauteur dans le robot, celui-ci peut rencontrer un ergot qui, se mettant entre les flasques, va donner une position particulière au palet lorsque celui-ci tombe choisissant ainsi la couleur. Une autre solution consiste à mettre cet ergot dans la trajectoire de chute et on se sert alors les forces d'inertie pour le retourner.

Le système du four de cimenterie : ce système consiste à amener le palet dans un tube dont le diamètre est égal à celui du palet. Ce tube est équipé de deux guides longitudinaux. Le palet rentre dans le tube et les flasques se mettent entre les deux guides. Ensuite, si le palet doit être retourné le tube sera une rotation autour de son axe de 180 degrés. Ce système relativement simple nécessite néanmoins un système de relevage du palet et un système pour pouvoir le mettre dans le tube et l'évacuer après un retournement éventuel.

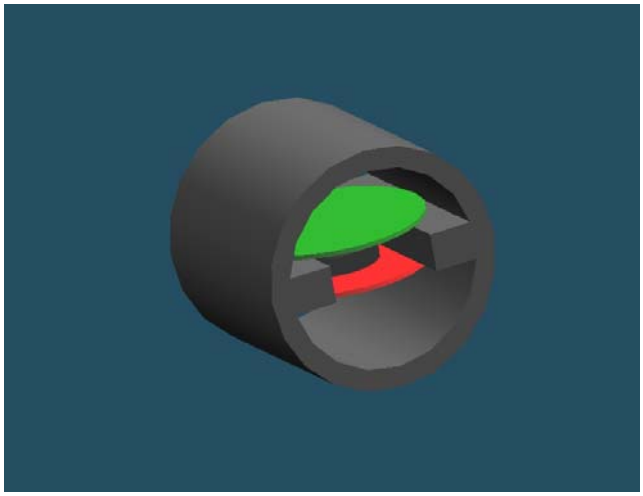


**Figure 14 : Solution de type 'four de cimenterie'**

#### 3.1.3.2 Systèmes traitant le palet sur la table

Un des systèmes envisagés a été de redresser le palet verticalement et ensuite de le surélever pour le mettre au-dessus de l'aire de stockage des piles le choix de la couleur étant beaucoup plus facile : il suffit de faire tomber le palet d'un côté ou de l'autre à l'aide d'un petit ergot mobile. Ce système a été arrêté au niveau de l'étude car nous n'avons pas pu trouver des systèmes simples pour redresser le palet.

Le système de pinces : ce système déjà cité auparavant consiste à apprendre le palet entre les flasques à l'aide d'une pince de soulever celui-ci du sol par un mouvement vertical est ensuite effectué une rotation de 180 degrés autour de la centrale de la pinces.
Un système à peu près équivalente à celui de la pince, et celui du levier : deux bandes métalliques se trouvant à auteur des flasques se trouvent dans la trajectoire du palet. Si celui-ci est de la bonne couleur, le palet passe à travers des bandes sans qu'aucun mouvement de son fait. Si par contre le palet et de la mauvaise couleur, la bande effectue une rotation de 180 degrés vis-à-vis d'un axe commun horizontal et retourna donc ainsi le palet. Ce système a été trouvé chez une équipe étrangère lors de la coupe de Belgique.

Systèmes employant l'inertie du palet :
Nous avons aussi effectué diverses études sur le système utilisant l'inertie du palet pour le retourner. Ces systèmes consistent à augmenter la vitesse du palet pour ensuite le bloquer sur une des flasques et celui-ci se retourne sous les forces inertie. La vitesse peut être donnée de différentes façons comme par exemple en donnant un choc en accélérant grâce à un tapis roulant ou simplement en lui donnant une position élevée, les forces d'inertie faisant le reste.
Le cas du tapis roulant avait été envisagé. Ce tapis roulant possède différents ergots qui attrapent le palet et l'accélèrent ensuite une pièce mobile attrape celui-ci sur les flasques et celui-ci sous ses forces inertie se retourne. Le principal problème était que si les ergots tombaient sur le palet, celui-ci se retrouvait coincé contre la table. Ce système a été retrouvé à la coupe de robotique, utilisé par l'équipe française de Minitech, (qui nous a battu à la finale de la coupe de Belgique). Cette équipe employait une bande transporteuse perpendiculaire à l'axe de déplacement du robot. Lorsque le palet devait être retourné, celui-ci butait sur une tige. La vitesse du palet suffisait à le retourner

Système de la casserole : Il s'agit ici d'un tube fermé sur l'un de ses côtés par une plaque munie d'un ergot mobile autour de l'axe du tube. Ce tube est aussi coupé dans sa partie circulaire à fin de pouvoir laisser passer un palet. Le fonctionnement le suivant : le palet rentre de façon longitudinale dans le tube, celui-ci est analysé de façon à savoir sa couleur avant retournement. Si cette couleur est bonne, le palet est évacué grâce à un ergot qui tourne dans un sens déterminé par la découpe du tube sans être retourné. Dans le cas contraire, l'ergo tournera dans le sens contraire et retournera le palet avant de l'évacuer par la même découpe.

Système par pivotement : Ces systèmes consistent à avoir une lame de forme déterminée se situant à la hauteur des flasques des palets. Lorsqu'un palet est en contact avec lame, celle-ci effectue une rotation autour de son axe central retournant le palet. La forme de cette lame déterminera la prise des palets par coincement. C'est un système à peu près équivalent qui a été trouvé sur le robot de l'école royale militaire lors de la compétition en Belgique.

Système de retournement par choc : Le système le plus simple envisagé, et finalement le système retenu. Une patte mobile est située entre les flasques des palets. Lorsque la couleur a été analysée et qu'il faut retourner le palet, cette patte aura un mouvement vertical qui donnera une 'claque' au palet pour le retourner. Les diverses options de cette méthode ainsi que les réglages seront expliqués par la suite.

### 3.1.4 Systèmes permettant de casser les piles adverses

Les systèmes pour casser les piles adversaire ont été difficilement envisagés car dans tous les cas de figure, nous nous sommes retrouvés confrontés au cas adjacent et critique d'une pile contre un bord. Tous systèmes mécaniques servant à soulever d'une pile posaient le problème que les palets tombaient hors de la table lorsque ceux-ci étaient contre le bord tandis que les systèmes de type pare-chocs (qui poussaient les palets supérieurs par exemple) bloquaient le robot si cette pile se trouvait contre le bord. En effet le microswitch détectant le contact ne pouvait pas se déclencher et le robot risquait de patiner.
Le seul cas viable et fiable est le système de pare-chocs mobile qui puisse se déformer lorsque que le robot rencontre les piles contre un bord et donc qui permettent au microswitch de se mettre en fonction. La prise des palets contre le bord sera elle résolue grâce à la programmation en faisant longer le robot le long de bord.
Ce système a finalement été créé grâce à un élastique large à usage textile sur lequel nous pouvons régler la tension. Cet élastique sert de pare-chocs pour faire tomber les palets verticaux, pousse les palets supérieurs des piles afin de les casser et est suffisamment déformable pour éviter les problèmes des piles contre bord (si nous rencontrons une pile contre un bord, la déformation de l'élastique permet quand même aux microswitchs de toucher le bord).

### 3.1.5 Systèmes permettant d'abattre les palets en hauteur

Dès le début de la construction mécanique, nous avons mis de côté le système de tir parce que celui-ci paraissait extrêmement compliqué à construire et parce que les divers essais que nous avons faits sur la table nous ont montré que l'endroit d'impact sur le palet était important pour le tomber. Cette partie du projet a été confiée à un élève de quatrième année dans le cadre d'un projet.
Lors de leurs recherches afin de trouver un système qui empêchait aux balles de ping-pong se trouvant sur la table de rentrer dans le robot, nous avons pensé un système de soufflerie qui pousserait celles-ci hors de notre trajectoire. Les divers essais faits avec différente soufflerie (décapeur thermique, moteurs d'avions télécommandés,...) nous ont permis de voir qu'un moteur d'avions télécommandés était non seulement capable de repousser les balles de ping-pong, mais aussi de faire tomber un palet.
Les hélices se trouvant sur ce moteur étant de grande taille, il était impossible de l'introduire dans le volume du robot. Néanmoins une petite recherche nous a permis de trouver des turbines de mêmes puissances et de plus petite taille.
Des essais ont été entamés à fin de diminuer le volume la turbine : diverses tuyères ont été fabriquées en fibre de verre polyester à fin de diminuer la taille du tube. Ces essais ont été infructueux car nous ne disposions pas des pales fixes qui se trouvent en aval de la turbine et qui ont pour effet d'augmenter le rendement et de redresser le flux. Nous ne disposions alors pas avec ses nouvelles tuyères de vitesse de flux suffisance pour pouvoir faire tomber les palets cibles.

D'autres idées ont été aussi évoquées pour faire tomber les palets en hauteur : un verra frappeur, un système de jet de gaz, une projection de gaz liquide, un canon à balle de ping-pong utilisant énergie à ressorts, l'énergie pneumatique (un vérin frappeur frappe la balle, des gaz comprimés sont injectés dans l'âme du canon propulsant la balle, des gaz comprimés sont injectés dans une membrane souple à l'intérieur de l'âme du canon pour économiser le gaz). Mais la solution du jet d'air est clairement apparue comme la plus simple et la moins encombrante

## 3.2 Réalisation pratique du robot

Suite aux diverses pistes évoquées l'architecture générale du robot a du être choisie. Cette partie du travail a été réalisée au début du deuxième semestre pour pouvoir débuter la construction du robot. La forme définitive du robot va maintenant être présentée.
Le robot est de forme parallélépipédique. Les dimensions de la base sont 250 mm x 330 mm (nous avons pris 40 mm de jeu vis-à-vis du règlement pour tenir compte des divers éléments pouvant être ajoutés en dernière minute (micro-switchs,…). A l'avant, se situe un entonnoir qui permet de collecter les palets et de las recentrer. Le grand côté est placé perpendiculairement à la direction d'avance pour avoir l'ouverture la plus large pour cet entonnoir.
Comme l'année passée, la traction est obtenue par deux roues placées au centre de deux des faces du robot pour permettre à celui-ci de tourner sur place. Le soutien est obtenu par quatre billes folles.

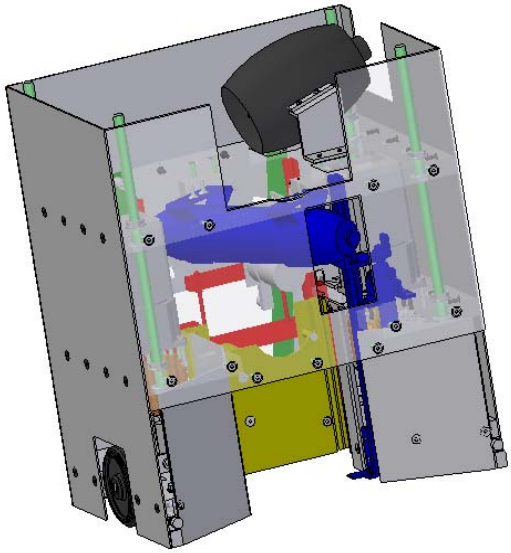


**Figure 15 : Forme générale du robot modélisé**

Les palets vont donc entrer dans le robot, aidés par l'entonnoir. Ensuite, ceux-ci sont arrêtés par une plaque intermédiaire (en jaune). La couleur est analysée. Si le palet doit être retourné, le système pneumatique (en bleu) est utilisé. Ensuite, les palets passent dans la zone d'empilement (en rouge).

### 3.2.1 Architecture générale – châssis

Le châssis a été réalisé grâce à des plaques de plexiglas de 16 mm d'épaisseur. Ce matériau présente de nombreux avantages, notamment au niveau de l'usinabilité. La structure de base du robot est constituée de trois étages, reliées entre eux par des tiges filetées M10. L'étage du bas est situé à hauteur des bords pour permettre de reprendre les chocs. L'étage intermédiaire permet de soutenir les moteurs de traction. L'étage supérieur comporte une grande partie des cartes électroniques et le microcontrôleur.

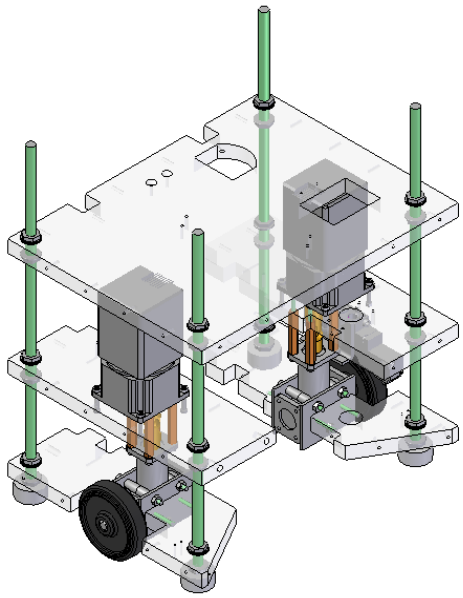


**Figure 16 : Châssis du robot**

En amont du robot nous avons placé un entonnoir formé de tôle en aluminium avec un angle de 30 degrés vis-à-vis de la face avant du robot de façon à ramener les palets légèrement excentrés vis-à-vis du robot dans le système de retournement. Ce système s'est révélé très efficace et est un bon compromis entre la taille du cône et l'angle nécessaire pour faire rouler les palets le long de celui-ci. De plus, sa profondeur a été étudiée de telle sorte que les palets situés sur le bord puissent être retournés, même si le robot arrive perpendiculairement à celui-ci.

### 3.2.2 Transmission mécanique – soutien du robot

#### 3.2.2.1 Installation de la transmission.

Fort de l'expérience des années précédentes, nous avons plus à appréhender les avantages inconvénients des moteurs de traction. Un des principaux inconvénients provient des problèmes de régulation aux basses vitesses. Il est apparu que le système de régulation n'était pas en fonction dans le domaine des vitesses très basses (inférieur à 10 % de la vitesse maximale). Grâce à des essais nous avons déterminé la courbe de régulation et donc la limite inférieure de vitesse à laquelle la régulation était efficace. À partir de cette cour de régulation, de la taille des roues disponibles et de la vitesse estimée du robot nous avons déterminé un rapport de réduction de la transmission.

Pour une vitesse nominale du robot de 60 cm/s et des roues de 80 mm de diamètre, un rapport de réduction 1 :2 était souhaitable pour se situer dans la zone de régulation efficace des moteurs.
Étant donné le volume pris dans le système de retournement, nous ne pouvions pas mettre l'arbre du moteur dans la même direction que l'arbre de la roue d'où la nécessité de trouver un renvoi d'angle à 90°. Étant donné la grande complexité tant au point de vue réglage et fabrication du renvoi d'angle, et étant donné le manque de moyens humains pour l'usinage de pièces spécifiques, nous avons décidé d'acheter cette pièce toute faite. Parmi les différentes pièces industrielles disponibles, nous en avons trouvé une si convenait au point de vue rapport de transmission et au point de vue taille, mais qu'il ne convenait pas au point de vue de la limite supérieure de puissance transmise. Ce renvoi d'angle de pourrait donc pas convenir dans un usage industriel est intensif mais satisfait à nos exigences étant donné le peu de temps d'utilisation de ce système.
Lors de l'installation de ce renvoi d'angle, il fallait l'accoupler aux moteurs. Nous avons choisi un accouplement par cardan. Ce système pour avantage de limiter les efforts sur les roulements du moteur en cas de défaut d'alignement des axes lors de la construction ou du montage. Cet accouplement était disponible parmi les pièces de l'atelier et n'a donc pas dû être commandé.
L'usage de cet accouplement nous a contraint à l'utilisation d'entretoise entre le renvoi d'angle et le moteur. Le système de transmission est hyperstatique car le moteur et le renvoi d'angle sont liés à la structure du robot.

#### 3.2.2.2 Les roues.

Les roues aient été achetées sur le même modèle que l'année passée. Elle présentait une adhérence et de solidité importante favorable à un robot massif. Néanmoins, les essais faits avec le robot d'année passée ont montre les limites de la fixation par collage. Il fallait cette année trouvait un système permettant aux roues de transmettre un couple important. Nous avons trouvé un système équivalent à un système d'arbres cannelé.
Étant donné le périmètre prédéterminé, et la taille des renvois d'angle, il a fallu usiner la jante et le moyeu de la roue à fin qu'elle puisse rentrer dans le périmètre. Une pièce intermédiaire devait adapter le diamètre intérieur de la roue et le diamètre extérieur de l'arbre du renvoi d'angle. C'est cette pièce qui devait disposer de cannelures.
Pour fabriquer cette pièce nous avons utilisé un barreau hexagonal qui sert normalement à fabriquer des entretoises. Celui-ci a été percé au niveau axial pour pouvoir accueillir l'arbre du renvoi d'angle. De plus, nous y avons fait un trou perpendiculaire à fin de pouvoir accueillir une goupille qui passera à travers l'arbre du renvoi d'angle.
Le barreau a été fixé sur l'arbre et ensuite forcé à travers le moyeu de la roue à la presse (effort nominal : 10 tonnes). Nous nous sommes servis de la grande plasticité du polymère du moyeu de la roue à fin de lui donner la forme de cannelures.
Ce système s'est révélé très efficace pour transmettre un couple important néanmoins, lors des essais où il est eu blocage du robot, nous avons remarqué une usure importante de la roue et de la table.

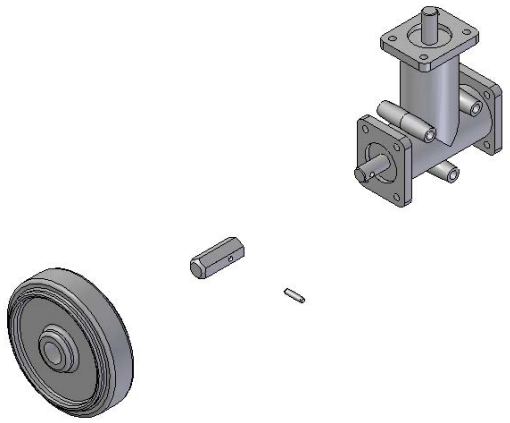


**Figure 17 : Assemblage des roues**

#### 3.2.2.3 Les billes folles.

Les deux points d'appui des roues n'étant pas suffisants pour assurer la stabilité, il était nécessaire d'employer des billes folles pour la construction du robot. Cette année, étant donné la diminution du périmètre du robot, était indispensable de trouver des billes folles de petite taille.
Les recherches de ce matériel au niveau industriel qui s'est révélé très difficile. En effet les délais de livraison étaient supérieurs à trois semaines. Nous nous sommes donc rabattus sur des billes de petite taille qui pouvait nous être fournie rapidement. Ces billes sont au départ conçu pour être intégré dans un trou de leur taille. Elle ne permettait donc pas de réglage. Nous avons résolu le problème par la construction d'une pièce nylon pouvant soutenir la bille et qui était fileté. Cette pièce est fixée sur les tiges filetées de la structure.
Il sera nécessaire pour les années futures de commander directement des billes folles ayant déjà un axe fileté que l'on fixerait dans le châssis. Ceci permettant d'avoir un réglage définitif qui ne nécessitera pas de fastidieux réglage à chaque montage démontage de la structure.

### 3.2.3 Système permettant de faire tomber les palets et de casser les piles adverses

Ce système a été adapté à la fin de la construction du robot. Il s'agit simplement d'un élastique assez large qui est fixé sur l'entonnoir, à une hauteur de 40 mm. Il permet non seulement de faire tomber les palets verticaux (le point d'appui est au-dessus du centre de gravité du palet, ce qui est favorable), mais il permet également de casser les piles adverses. En effet, vu le faible coefficient de frottement entre les faces des palets, lorsque nous roulions vers une pile de trois palets, les palets supérieurs étaient éjectés, sans faire bouger le palet du bas.

### 3.2.4 Système de retournement

Plusieurs essais ont été menés pour réaliser pratiquement le concept choisi (choc sur le palet). Nous avons notamment essayé un vérin pneumatique et un électroaimant. Tous les systèmes servant d'un moteur en rotation ont été exclus et ceci pour deux raisons : il nous fallait un système puissant et rapide or il y avait des problèmes d'inertie des masses rotation car la course dans le cas d'un levier est seulement de 60 degrés. De plus, ces systèmes nécessitaient un volume important.

#### 3.2.4.1 Electroaimant

Les essais par un électroaimant se sont révélés négatifs. Il apparaissait un manque de puissance et de course au niveau de l'électroaimant. De plus ce système a un volume important. La recherche d'un électroaimant mieux adapté parmi le matériel industriel nous a donné un résultat mais il apparaissait que celui-ci était beaucoup plus volumineux que l'électroaimant avec lequel nous avons réalisé les essais. De plus, les délais de livraison étaient de l'ordre de 3 à 6 semaines avec un prix de l'ordre de 500 €.

#### 3.2.4.2 Vérin pneumatique, système à levier

Les systèmes automatiques se sont révélés beaucoup plus performants. Ces essais ont été effectués avec un piston de 50 mm de course et de 12 mm de diamètre. Dans un premier temps, nous avons essayé le système par levier. Il apparaît que ces systèmes marchaient bien et était viable et fiable néanmoins, il apparaît que ce système prenait un volume important, que la forme de la patte de levage influait sur la fiabilité et que ce système devait être associé à une coulisse de permettant de retirer le capteur de couleur lors de retournement.

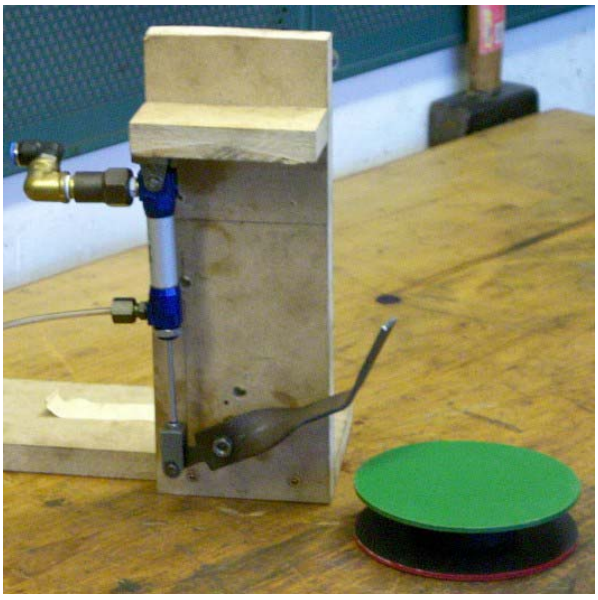


**Figure 18 : prototype de système de retournement avec vérin**

L'étude de ce système nous a donné un système à câble. Ce système permettait de choisir quel élément (la coulisse où le levier) aller bouger le premier en fonction de la force dès ressort de rappel. Le mouvement de l'autre élément se faisait lorsque le premier arrivait à butée. Ce système avait un inconvénient important : en cas de blocage du capteur de couleur, le levier pouvait quand même suivre sa course entraînant une destruction du capteur.

D'autres systèmes utilisant des lumières ont été essayés mais ceci était d'une grande complexité et donc peu fiable.

#### 3.2.4.3 Vérin pneumatique, système à glissière

Des essais utilisant une coulisse à la place le levier ont été effectués même si cette option avait été éliminée lors des réunions d'avancement du projet.
Les essais directs n'ont pas donné de bons résultats car il est apparu que la course de la glissière n'était pas suffisante : le palet en rotation frappait la patte de retournement et celui-ci était arrêté dans son mouvement. Grâce à un système de câbles et de poulies, nous avons augmenté de la course de la glissière tout en ayant la même course au niveau du piston. Ces essais se sont révélés très satisfaisant. C'est pourquoi nous avons choisi ce système de retournement.
Nous y avons ajouté quelques modifications par rapport au prototype initial et ceci pour deux raisons : lors de nos essais la pression était à la limite supérieure de la norme donnée pour le concours et le système à câble est très compliqué donc peu fiable. Nous avons choisi un piston de 18 mm de diamètre à fin de diminuer la pression d'usage. Sa course est de 100 mm avant de pouvoir utiliser le système directement.
Il est difficile de choisir le matériel pneumatique car il y a des éléments dont nous ne pouvons pas avoir connaissance : la masse du piston, le frottement dans le piston, ainsi que l'influence de l'électrovanne et du pressiostat qui règle le débit sur la pression que l'on peut retrouver à l'intérieur du piston. Ce choix a donc été un compromis entre la pression maximale autorisée et la consommation d'air comprimé.
Au niveau de la construction, le piston et la glissière ont été intégrés directement dans la structure du robot. Une pièce rivetée à la glissière sera boulonnée à la tige du piston afin d'assurait la liaison entre ces deux éléments. Sur la partie mobile de la glissière sont fixées par boulonnage deux éléments : la patte de retournement et le capteur de couleur. Sur la patte de retournement, nous retrouvons une charnière qui permettra au piston de retrouver sa position d'attente sans abîmer un éventuel palet qui pourrait se trouver dans sa course. Le capteur de couleur est quant à lui fixées sur un support en nylon et est maintenu grâce à une vis de pression.

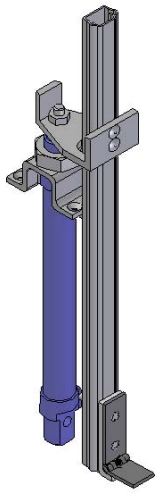


**Figure 19 : Système de retournement**

Le système pneumatique complet se compose des éléments suivants :
Un réservoir d'air comprimé qui sert de réserve d'énergie (pression maximale : 15 bars) (référence FESTO : DSN-18-100)
Un pressiostat qui maintient la pression de sortie à 4 bars (pression de service maximale autorisée (référence FESTO)
Une électrovanne qui permet de commander le vérin à double effet (type 5/2 : 5 orifices, 2 voies) (référence FESTO MEH-5/2-1/8)
Le vérin à double effet, course 100 mm, diamètre de tige 16 mm (référence FESTO CRVZS-0,4)

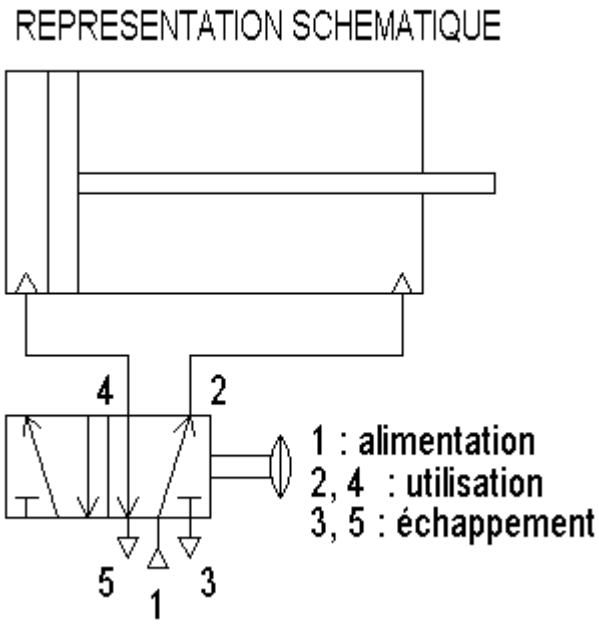



**Figure 20 : Commande d'un vérin double effet**

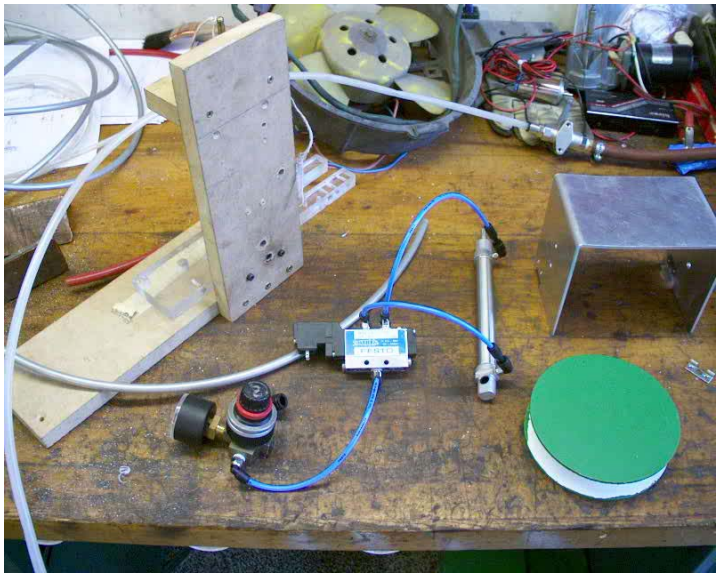


**Figure 21 : Système pneumatique**

L'électrovanne sera placée juste à côté du piston et sera fixée au renvoi d'angle grâce à une équerre. Cette électrovanne a été choisie en 24 V affin d'avoir la même alimentation que les moteurs de déplacement et que l'ascenseur.

Le principe de la commande est très simple :

- En position de repos, l'électrovanne est en position 1. La pression agit sur la partie supérieure du vérin, la chambre inférieure est mise à l'air libre. Le piston est donc en position basse
- Lorsque le palet doit être retourné, une impulsion de commande est donnée à l'électrovanne, il y a inversion de la pression dans les deux chambres du vérin (chambre inférieure mise sous pression, chambre supérieure à pression atmosphérique). Le vérin va donc se lever.

#### 3.2.4.4 Essais d'utilisation de dioxyde de carbone.

Au départ, nous voulions utiliser du dioxyde de carbone liquide comme réserve d'énergie. Un mélange biphasiques à température constante et donc à pression constante sert de réserve de gaz. Lorsque nous avons besoin de pression, nous soutirons du gaz de la réserve et ensuite une ébullition locale s'effectue. Cette ébullition est endothermique, elle se sert des calories du milieu ambiant pour revenir à l'équilibre.
Un des problèmes à régler dès le départ, est le fait qu'à l'équilibre la pression du dioxyde de carbone est de 58 bars. Il a fallu trouver des matériaux tels que des tuyaux et pressiostat qui résiste à cette pression. Une fois ces éléments trouvés, il est apparu que les petites capsules de dioxyde de carbone utilisées ne permettait pas de faire fonctionner le piston de façon correcte: la capsule n'avait pas une surface suffisante et donc ne recevait pas assez d'énergie pour permettre d'abord de pression suffisante pour le coup suivant. Dans un deuxième temps, nous avons détendu l'ensemble de la capsule dans un réservoir. Normalement, ce dioxyde de carbone totalement en phase gazeuse de réagir qu'un gaz comprimé. Les essais nous ont démontré que cela ne se passait pas comme la théorie. Aucune explication n'a pu nous être fournie par les professeurs de thermodynamique.
Nous avons donc choisi de travailler avec de l'air comprimé plutôt qu'avec du dioxyde de carbone. Cela à une conséquence importante de diminuer très fortement l'autonomie du robot au point de vue de retournement. Alors que l'on pouvait théoriquement retourner avec une cartouche de dioxyde de carbone une centaine de palet dans un volume de quelques centimètres cubes, dans le cas de l'air comprimé nous pouvons seulement retourner quatorze palets correctement avec un réservoir de 0,4 L à une pression de 12 bars. Le volume du réservoir et donc très important et de par sa fabrication et sa forme cylindrique, il était difficile à implanter dans le robot. Nous avons finalement pu placer celui-ci en dessous de la plaque supérieure.
En plus de ce réservoir, il a fallu ajouter un raccord en Y à l'entrée. Les deux orifices permettaient :

- de placer le pressiostat pour alimenter le circuit à une pression constante de 4 bars
- de prévoir les raccords nécessaires au rechargement du robot

### 3.2.5 Système d'empilement

Une fois le système mécanique de retournement choisi, il a été nécessaire de rechercher un moyen d'empiler les palets en partant par le bas. Le système imaginé est constitué d'un ascenseur pouvant effectuer un mouvement vertical en translation et de deux charnières mobiles. Le principe de fonctionnement est le suivant :

- Le premier palet se présente dans le système, les tiges se relèvent de la hauteur d'un palet
- Le second palet se présente
- Le dispositif effectue un aller-retour entre les positions basses et hautes. Les charnières passent du palet supérieur au palet inférieur.
- Le même cycle est recommencé pour le troisième palet

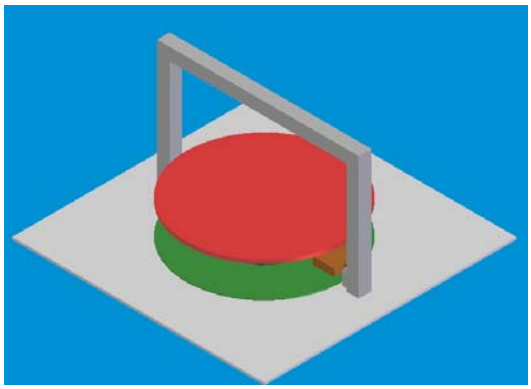
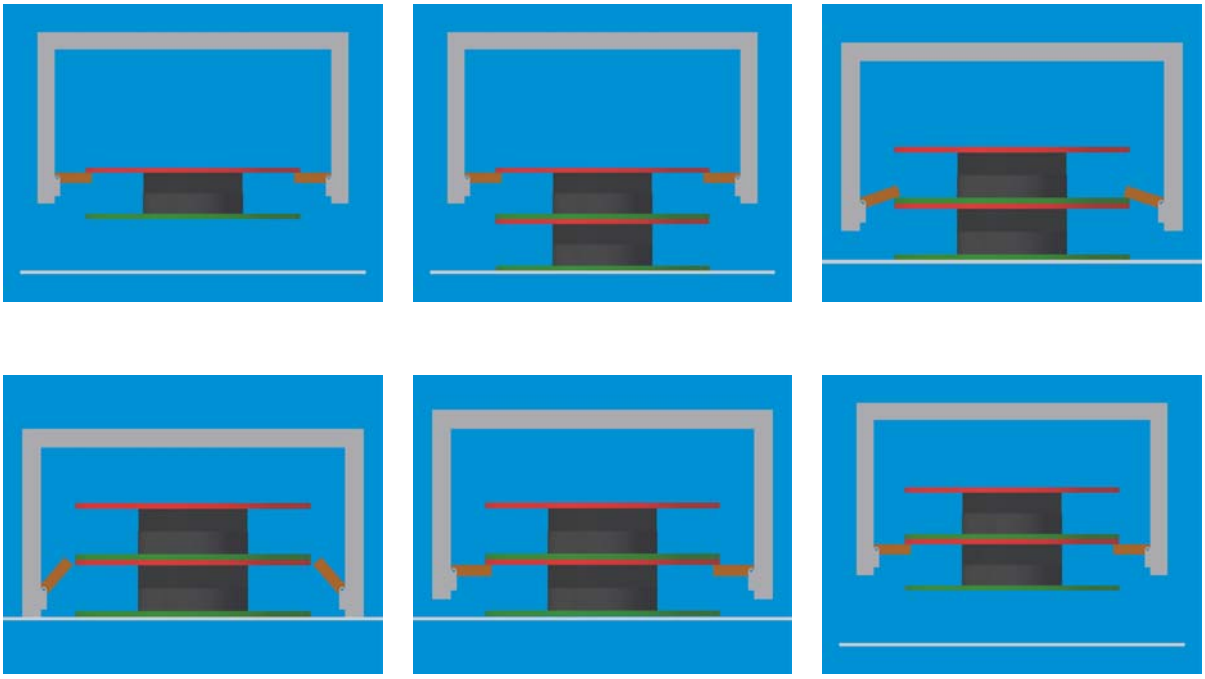


**Figure 22 : Principe de fonctionnement de l'ascenseur**



Pour éviter de perdre un palet en cours de match, des cales d'épaisseur ont été ajoutées pour assurer que les palets sont bien centrés.

Au niveau de l'entraînement de ce système, nous avions pensé à deux hypothèses :

- Une tige filetée solidaire de l'axe d'un moteur et un écrou relié à l'ascenseur. La rotation du moteur dans un sens ou dans l'autre permet de faire monter ou de faire descendre le système.
- Le mouvement alternatif est obtenu au moyen d'un système classique bielle-manivelle

Nous avons estimé que le système bielle-manivelle est beaucoup plus simple au niveau de la commande du moteur puisqu'il n'impose pas de changer le sens de rotation du moteur. C'est donc ce système qui a été choisi au final.
Des micro-switchs seront placés le long de la glissière pour déterminer la position de l'ascenseur.
Nous avons décidé d'employer trois positions :
Position basse : Les charnières se situent entre les deux flasques du palet. L'ascenseur doit passer par cette position pour prendre les palets. Lors du déchargement, l'ascenseur se place en position basse
Position intermédiaire : Les palets sont soulevés de quelques millimètres par rapport au sol. Il s'agit de la position par défaut. Le robot peut ainsi avancer sans que les palets ne frottent contre la table.
Position haute : cette position sert à lever la plaque intermédiaire (cf. chapitre suivant)

### 3.2.6 Plaque intermédiaire

Pour permettre de garder un comportement indépendant de la vitesse, nous avons décidé de séparer la zone de retournement de la zone d'empilage par une plaque mobile. Celle- ci joue plusieurs rôles :

- Elle permet de s'assurer qu le palet à retourner se présente toujours de la même manière quelle que soit la vitesse (il y a immanquablement un certain temps de réaction entre la détection du palet et l'action par le vérin).
- Elle limite l'accès à la zone d'empilage à un seul palet

De par ce fait, on s'assure qu le retournement pourra se faire même si la vitesse d'avance du robot varie.
Cette plaque est montée sur deux glissières pour permettre son mouvement vertical. Au repos, sa position est telle qu'elle ne permet pas le passage de palet. Sa levée est assurée par le mouvement de l'ascenseur. La liaison se fait par l'intermédiaire d'une came. Le mouvement de descente s'effectue par gravité. On évite ainsi d'avoir des problèmes si la plaque descend alors qu'un palet est en dessous de celle-ci

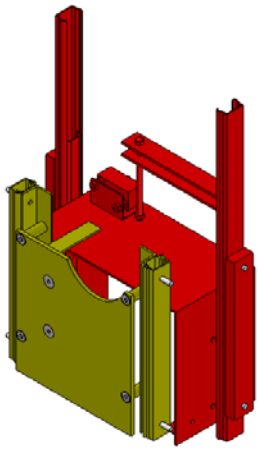


**Figure 23 : ascenseur (en rouge) et plaque (en jaune)**

La plaque intermédiaire est toujours baissée, sauf lorsque l'ascenseur se met en position haute (il y a donc un jeu entre la came et le haut de l'ascenseur).

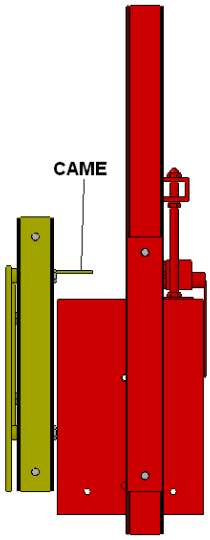



**Figure 24 : vue de côté**

### 3.2.7 Système de lâcher de piles

Lors de l'empilage, il est nécessaire de retenir les palets en place grâce à un système de portes. Le choix de ce système n'a pas posé le problème : il s'agit de deux barres qui se mettent en travers de l'ouverture arrière. Nous avons préféré employer deux barres plutôt qu'une plaque en raison du gain de place.

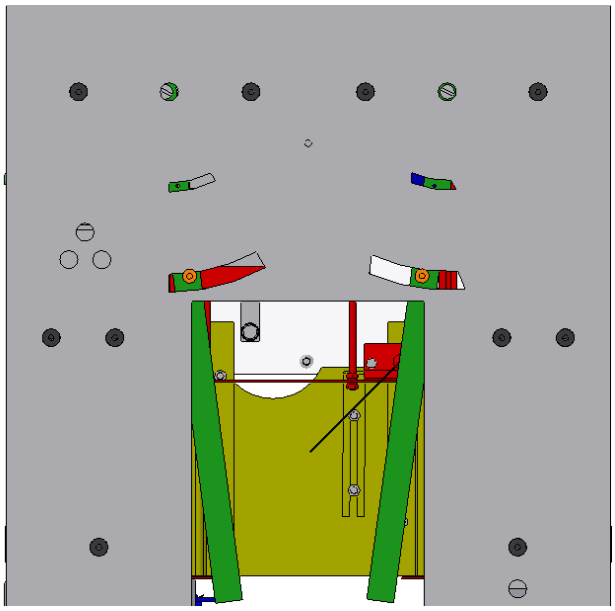


**Figure 25 : système de portes arrière**

Dans un premier temps, aux avons pensé utiliser une seule barre mobile est l'autre fixe. La pile devra dans ce cas contourner la barre fixe lors du déchargement. Mais lors de la construction, il est apparu que ce système nécessitait une place trop grande.
Étant donné la place laissée pour les guidages, nous avons décidé d'utiliser un système par rotation plutôt que par translation. L'axe des deux barres constituantes les portes seront fixées dans la plaque de plexiglas supérieur du châssis. Ces axes sont constitués de simple vis filetées M4 et les barres sont posées sur des entretoises en laiton afin d'améliorer le glissement et les guidages de celle-ci.
Toujours dans le but d'améliorer les guidages, une vis fixée dans les barres et pouvant se déplacer dans une lumière de la plaque d'habillage arrière a été placée (en orange sur la figure précédente).
Un des principaux problèmes que nous avons rencontré lors de la construction des portes arrière, a été le manque de préparation de la mécanique. En effet, l'électronique de commandes du moteur que nous avions envisagé pour ouvrir les portes arrière n'était pas prête. Nous avions du donc laisser de côté la cinématique préparée et avons dû élaborer avec les moyens trouve au laboratoire un système de remplacement.
Nous sommes retrouvés donc la semaine précédant la compétition de Belgique avec le système à construire sans compter la difficulté de trouver des matériels dans un délai très réduit. Nous avons donc élaboré le mécanisme en fonction de l'actuateur trouvé au laboratoire.
Dans un premier temps, nous avons travaillé avec un servomoteur servant à l'ouverture centralisée de porte de voitures. Ce servomoteur est constitué d'un petit moteur électrique entraînant une vis sans fin. Sur la vis en fin, se trouve un écrou guidé en translation sur lequel est fixée la partie mobile.
Ce système en translation a été utilisé pour faire tourner un axe par un système bielle manivelle. À partir de cette rotation, des leviers partant de la partie mobile sont fixée sur les barres de la porte pour amplifier le mouvement. Lors de la construction, de nombreux problèmes au niveau de guidages de l'axe sont apparus. Ceux-ci ont été résolus par M. Vergari qui a adapté des éléments de roulements récupérés sur un vélo.

Le système du servomoteur est tel que lorsque celui-ci arrive en bout de course, le moteur se bloque. Or suite à une erreur de programmation, il est apparu que le moteur bloqué est resté alimenté entraînant des surintensités provoquent la destruction du servomoteur par effet thermique.
Suite à ce problème nous avons dû trouver un autre actuateur pouvant convenir pour le système mécanique déjà assemblé. Nous utilisé un électroaimant qui était disponible au laboratoire.
Cet électroaimant ne marche que dans un sens et pourquoi il a fallu y ajouter des ressorts de rappel au niveau de l'électroaimant et des portes elle-même. Ce système s'est avéré très facile pour effectuer les différents réglages qui doivent être très précis et très fiable. Ici aussi, un savant compromis a été effectué entre la force de l'électroaimant et les systèmes de rappel.
En position de repos, les portes sont refermées, empêchant les palets de sortir du robot. Lorsque l'électroaimant est commandé, le palet (ou la pile) peut être relâché.

### 3.2.8 Système permettant d'abattre les palets unicolores

Le règlement du concours précisait que les cibles pouvaient être atteintes par des projectiles qui devaient être des balles de tennis de table. Par contre, rien n'interdisait a priori de souffler les palets. Dès que nous avons reçu confirmation de ce fait, nous avons fait divers essais afin d'optimiser la chute des palets.
Les conclusions de ces essais sont qu'il est impossible d'avoir une trajectoire de flux qui permettent de faire tomber les deux palets en même temps.
Il est impossible de séparer le flux de la turbine à fin de faire tomber les deux palets en même temps.
La construction d'un système qui permettrait de dévier le flux ou de changer le l'axe de la turbine est très compliqué à cause de la puissance de la turbine car celle-ci développe plus d'un kilo de poussée.
Travaillant avec un angle fixe, il apparaît que le seul moyen fiable de faire tomber les deux palets était une trajectoire préprogrammée.
L'intégration de ce système dans le robot n'a pas posé beaucoup de problèmes car un espace suffisant avait été laissé lors de la construction pour y intégrer le système de tir et les cartes de commandes. Néanmoins, il a fallu faire un trou dans la plaque supérieure de soutien.

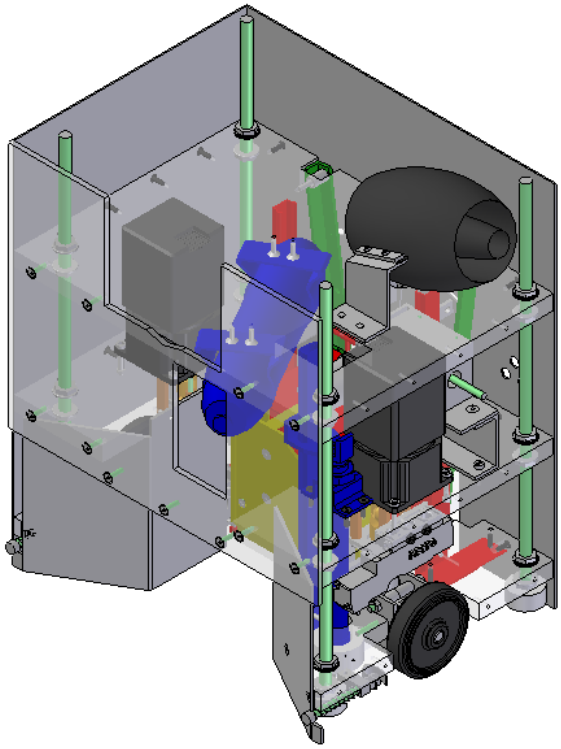


**Figure 26 : fixation de la turbine sur la plaque supérieure**

Pour la fixation de la turbine, nous avons construit une simple équerre qu'il a fallu fixer dans la plaque de plexiglas supérieure en la forant et la filetant. Cette équerre déterminera l'angle de flux de la turbine et sera fixée à celle-ci grâce à une patte spéciale.

### 3.2.9 Batteries.

Cette année étant donnée la plus grande compacité exigée au niveau du robot, nous avons dû utiliser les batteries modulables. À fin d'avoir une plus grande stabilité il était nécessaire de situer les batteries le plus près possible du sol. Nous avons utilisé les batteries modulables et de petite taille néanmoins celle-ci reste une puissance égale aux batteries au plomb utilisées l'année passée. Les batteries ont été séparées par groupe de façons de donner une attention de 12 vols. Les moteurs de propulsion exigeaient 24 V de tension, les batteries de 12 vols indépendantes étaient nécessaires pour l'alimentation du microcontrôleur et une batterie de 12 vols indépendante était aussi nécessaire pour l'alimentation de la turbine et de l'électroaimant Ceci nous a donc donné quatre éléments de 12 vols. À la fin de la construction du robot, nous avons vérifié avec des cubes identiques à la forme des batteries si celle-ci était bien implantable et qu'elles forment les éléments de 12 vols devraient avoir. Celles-ci ont été ensuite commandées aux formes désirées.
Au niveau de l'implantation dans le robot nous en effet particulièrement attention aux problèmes d'isolation au niveau des batteries elle-même et au niveau des plaques d'habillage extérieur qui est en contact avec celle-ci.

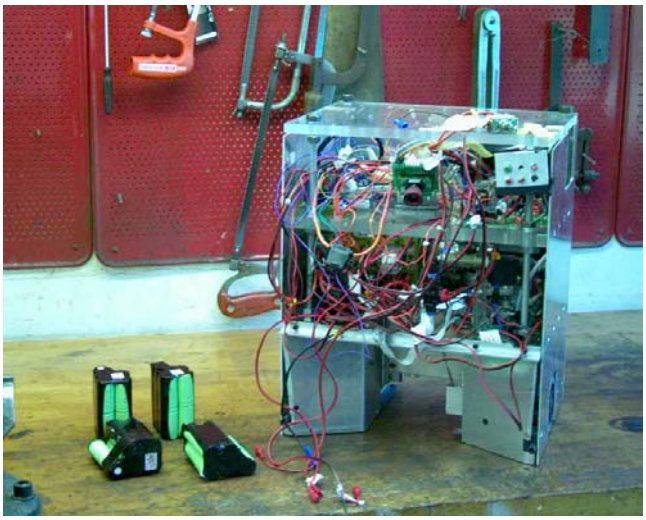


**Figure 27 : forme définitive du robot et ses batteries**

### 3.2.10 Microswitchs.

Les microswitch avant sont les capteurs qui servent à détecter la collision du robot contre les bords de la table ou contre l'adversaire. Ceux-ci sont soumis à de nombreux chocs et contraintes.
Fort de l'expérience de l'année passée, nous avons remarqué que ceux-ci sont fort sensibles aux coups et aux chocs qui peuvent survenir lors d'un match ou d'essais. Il a donc fallu éliminer tout contact direct entre les microswitch et le milieu extérieur. Ceci a augmenté la difficulté de l'aspect constructif. Il a fallu créer des systèmes mobiles qui évoluant dans leurs déplacements nécessaires ne forçaient jamais sur les ressorts et pattes des microswitchs. Ces systèmes devaient donc avoir des butées et des moyens de rappel indépendants des capteurs. Le système était construit de la façon suivante : une tige lisse en acier possédant un épaulement d'un côté et fileté de l'autre peut coulisser librement dans un bloc de laiton qui sera fixé sur la structure. Les efforts de rappel de cette coulisse sont donnés par des ressorts en compression. Une petite came permet de déclencher le microswitch et de le maintenir pendant toute la course de la coulisse.

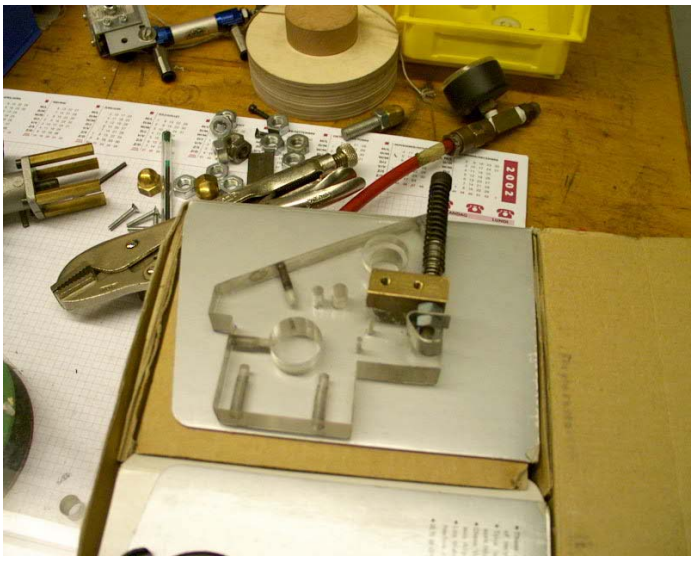



**Figure 28 : Système d'enclenchement des microswitchs avant**

### 3.2.11 Les plaques extérieures d'habillage.

Étant donné les jeux pris sur le périmètre et la taille importante des microswitch avant, il nous a fallu choisir des plaques d'habillage de 1,5 mm d'épaisseur. Dans un premier temps, nous avons essayé d'utiliser des plaques en inox mais la dureté de cet alliage entraîné une usure rapide des fraises coniques et rend l'usinage malaisé. Dans un deuxième temps nous sommes donc rabattues sur des plaques en aluminium. Ces plaques ont été percées pour le trou de fixation, pour les lumières des pièces mécaniques, ainsi que le nombreux trou à fin de pouvoir visualiser les LEDs des différents systèmes électroniques. Il est apparu lors d'un match que malgré une résistance plus faible, l'aluminium convenait pour les différentes contraintes et coups que recevait le robot. Seule de très faible déformation sont apparues. Dans un esprit peu fair-play, les plaques ont été polies à la pâte à polir à fin d'avoir un état de surface réfléchissant capable de dérouter la vision éventuelle de l'adversaire.
Au point des constructifs, nous avons dû veiller à isoler les plaques au niveau des batteries afin d'éviter tout court-circuit.

## 3.3 Problèmes liés à l'aspect constructif.

### 3.3.1 L'utilisation de plaque de plexiglas.

Il apparaît que l'utilisation de plaque de plexiglas pour la structure du robot nous a permis de gagner un temps énorme. En effet, ces plaques étant d'épaisseur relativement importante, elles ont permis d'économiser un nombre important d'équerre en permettant aux diverses pièces d'être fixé directement dans celle-ci.
De nombreuses pièces ont été fixées par boulonnage directement dans les plaques après avoir percé et fileté.
De plus, grâce à leur grande épaisseur, elles ont pu donner une grande rigidité à la structure du robot. Etant donné qu'elles se sont des isolants électriques, elles ont permis la fixation des différents éléments électroniques sans aucune autre protection. Elles permettent aussi d'alléger la structure du robot étant donnée leur faible densité.
Parmi les inconvénients de ce système, il apparaît que ces plaques occupent le volume le plus important du robot. Un autre inconvénient est l'intégration d'un nombre important de pièces sur cette structure : chaque modification de celle-ci entraînait de fastidieux montages et démontage. Point de vue des usinages, l'expérience nous a montré divers problèmes auxquels nous ne nous attendions pas.
Parmi ceux-ci nous retrouvons : l'aspect cassant des matériaux (certains usinages comme les fonds de trou entraînaient la rupture des matériaux.).
Les problèmes de fusion des matériaux (fusion du copeau sur la plaque, collages du copeau, absents d'évacuation du copeau, bourrage de l'outil, entraînait de mauvais usinages). Ce problème a été en partie résolue par l'utilisation d'un lubrifiant volatil.

### 3.3.2 L'utilisation de tiges filetées pour le châssis.

L'utilisation de tiges filetées pour le châssis paraissait séduisante au premier abord. Mais nous avons vite déchanté. Bien que ce système permette un réglage fin de la distance entre les différents de plaque de plexiglas qui constitue le châssis, il est apparu que le manque de soutien au niveau des écrous et la flexibilité des tiges filetées ne conféraient au châssis une grande rigidité. Dans les années futures, pour des montages identiques, il sera nécessaire d'utiliser les entretoises entre chaque plaque. Celles-ci auront un plus grand soutien, seront plus rigides et devront être placé en plus grand nombre. Si un réglage dans la dimension entre les plaques est nécessaire, il sera néanmoins obligatoire de refaire des entretoises.

### 3.3.3 Utilisation des charnières

Deux des systèmes mécaniques (ascenseur et patte de retournement) emploient des charnières. Dans chacun des cas, il est nécessaire de donner un angle de 90° à celle-ci au repos et d'avoir la possibilité de replier celle-ci jusqu'à l'amener verticalement lorsqu'elle rencontre un palet. Ensuite, il est nécessaire qu'elle reprenne sa position d'origine (grâce à un ressort de torsion)
Ceci est nécessaire pour :

- Réaliser la préhension des palets dans l'ascenseur (la charnière doit revenir pour soutenir le palet entre ses flasques)
- Permette le retour de la patte de retournement même si un palet est présent dans l'aire de retournement

Les charnières que nous avons trouvées s'ouvraient toutes à 270°. Il n'était pas possible de les employer telles qu'elles car beaucoup de place était perdue en dessous de leur axe, ce qui était exclu dans notre cas (il faut à tout prix descendre l'axe le plus bas possible).
La solution à ce problème a été d'employer une pièce supplémentaire fixée derrière la charnière qui empêche celle-ci de s'ouvrir plus bas que l'horizontale. Nous avons placé un ressort en torsion pour assurer le mouvement de rappel.

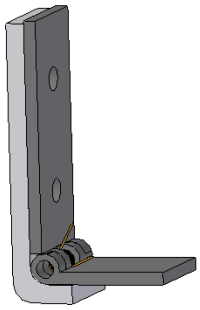


**Figure 29 : montage des charnières, position de repos**

## 3.4 Quelques systèmes offensifs.

Lors de leur réunion, nous avons pensé à quelque système offensif afin de dérouter l'adversaire. Aucun système n'a été installé sur le robot néanmoins ceux-ci peuvent constituer des idées intéressantes pour les années futures.
Fausses balises qui pulsent à des fréquences différentes des ultrasons et les infrarouges (ces systèmes doivent être de haute puissance) ayant pour but de perturber les radars et les éventuels télémètres de l'adversaire. Ces systèmes peuvent aussi être placés sur le robot. De même, les expériences faites avec une simple télécommande de télévision ont montré que l'on pouvait perturber le télémètre Sharp. Il est donc à déconseiller d'utiliser les télémètres pour éviter les chocs et des balises dont le signal n'a pas été crypté.
Un autre système consiste à charger l'habillage du robot par de la haute tension : les chocs avec un autre robot peuvent faire fluctuer les niveaux de tension des composants électroniques entraînant des perturbations qui peuvent être fatales au robot adverse.
Les plaques polies miroir : le fait de polir les plaques d habillage entraîne des reflets de la table et des objets à prendre entraînant des perturbations pour les robots adverses qui utiliseraient la vision.

## 3.5 Fabrication de la table

Une analyse rapide de la table de l'année passée nous a montré que celle-ci était mal supportée, qu'il n'était pas possible de régler le niveau, et que des rebords existaient entre les différentes pièces assemblées. En vue de perpétuer le concours sur longue période, nous avons décidé d'acheter une table d'une pièce qui pourrait servir pour plusieurs années. Nous avons trouvé un panneau dont la dimension était presque identique au plan de travail. Seule une découpe et le rajout d'une petite latte rabotée ont été nécessaires à fin que celui-ci et les dimensions requises. Nous sommes servis des restes de la découpe du panneau pour faire les bords qui seront vissé dans l'épaisseur du panneau. Une fois le panneau mis à la bonne dimension, celui-ci a été peint globalement en noir et ensuite à l'aide d'un masque, nous avons peint les carreaux blancs.
Pour soutenir cette table, nous avons entrepris la création d'un bâti. Ce bâti dispose de six pieds réglables en hauteur grâce à des tiges filetées.
Ce bâti est composé de trois paires de pieds reliaient ensemble part des profilés boulonnés. Ces sous-ensembles de pieds sont eux constitué de profilés soudés.
Le panneau constituant la table est simplement disposée sur le bâti.

## 3.6 Fabrication des palets

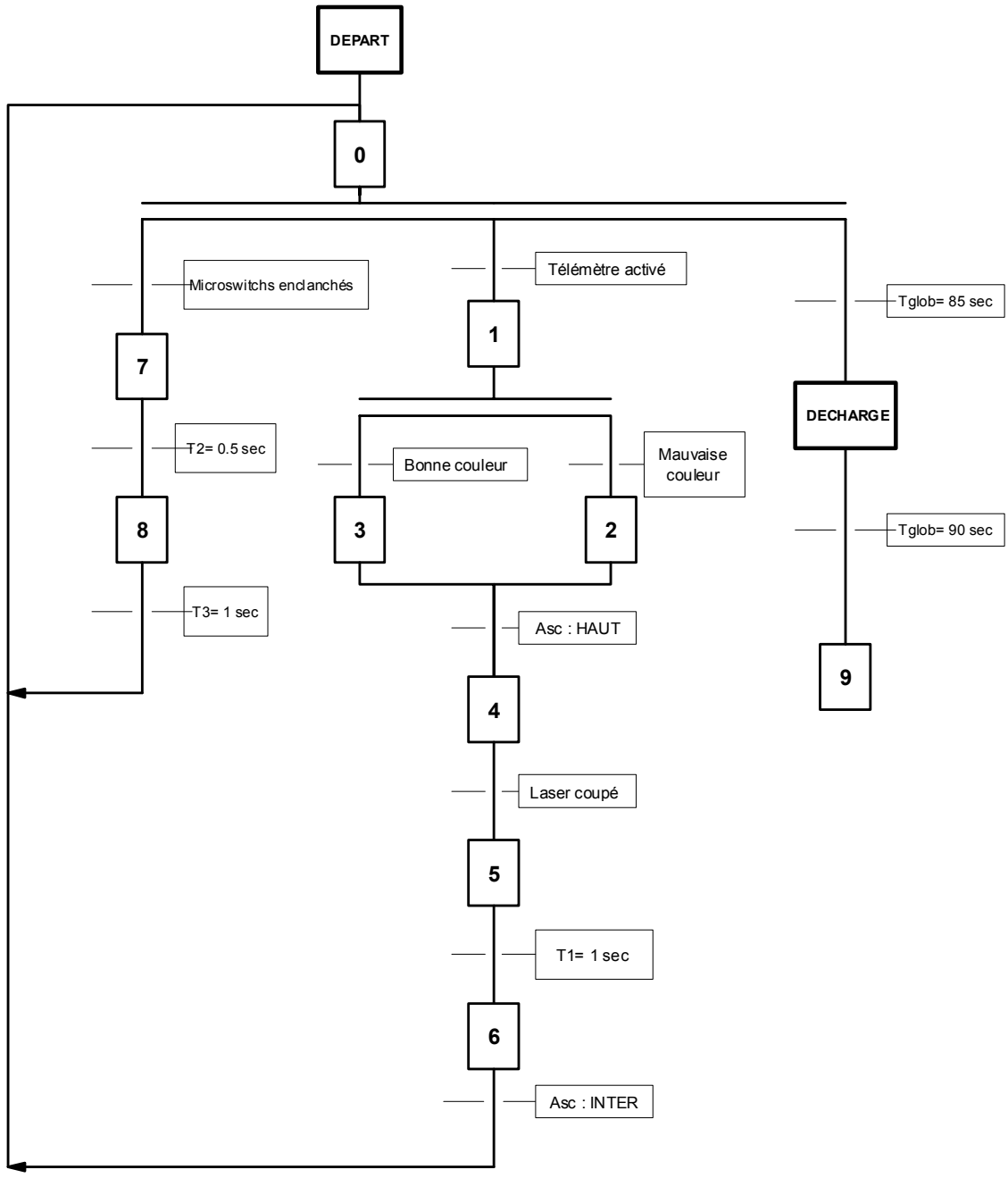
Dans un premier temps, suite aux normes données dans le règlement, nous sommes partis à la recherche de panneaux de bois en contreplaqué et de trois millimètres. Une fois ce bois trouvé, nous avons essayé dans un premier temps de fabriquer les flasques par découpage à la scie. Ce qui ne nous a pas donnés de très bons résultats. Dans un deuxième temps, M. Vergari, a réussi à tourner ses flasques au tour. Ces palets ont été ensuite peints et assemblés avec les moyens eux-mêmes déjà peint.
Durant l'année, il a été nécessaire de reconstruire des palets car les normes et les tolérances du règlement ont été changées. Il est apparu durant les matches que l'état surface et les moyens d'apposer la peinture ont eu une influence considérable sur la reconnaissance des palets.

# 4. Stratégie

## 4.1 Grafcets

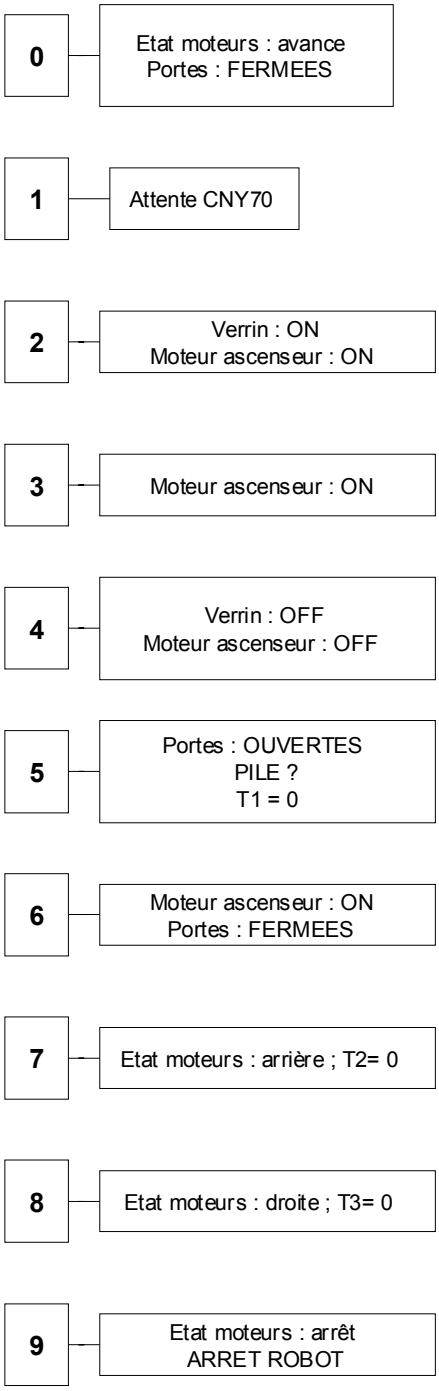
**Procédure de déplacement aléatoire + traitement palets**

**Grafcet de niveau 1**




PHILIPRONT N
DEPLUS S

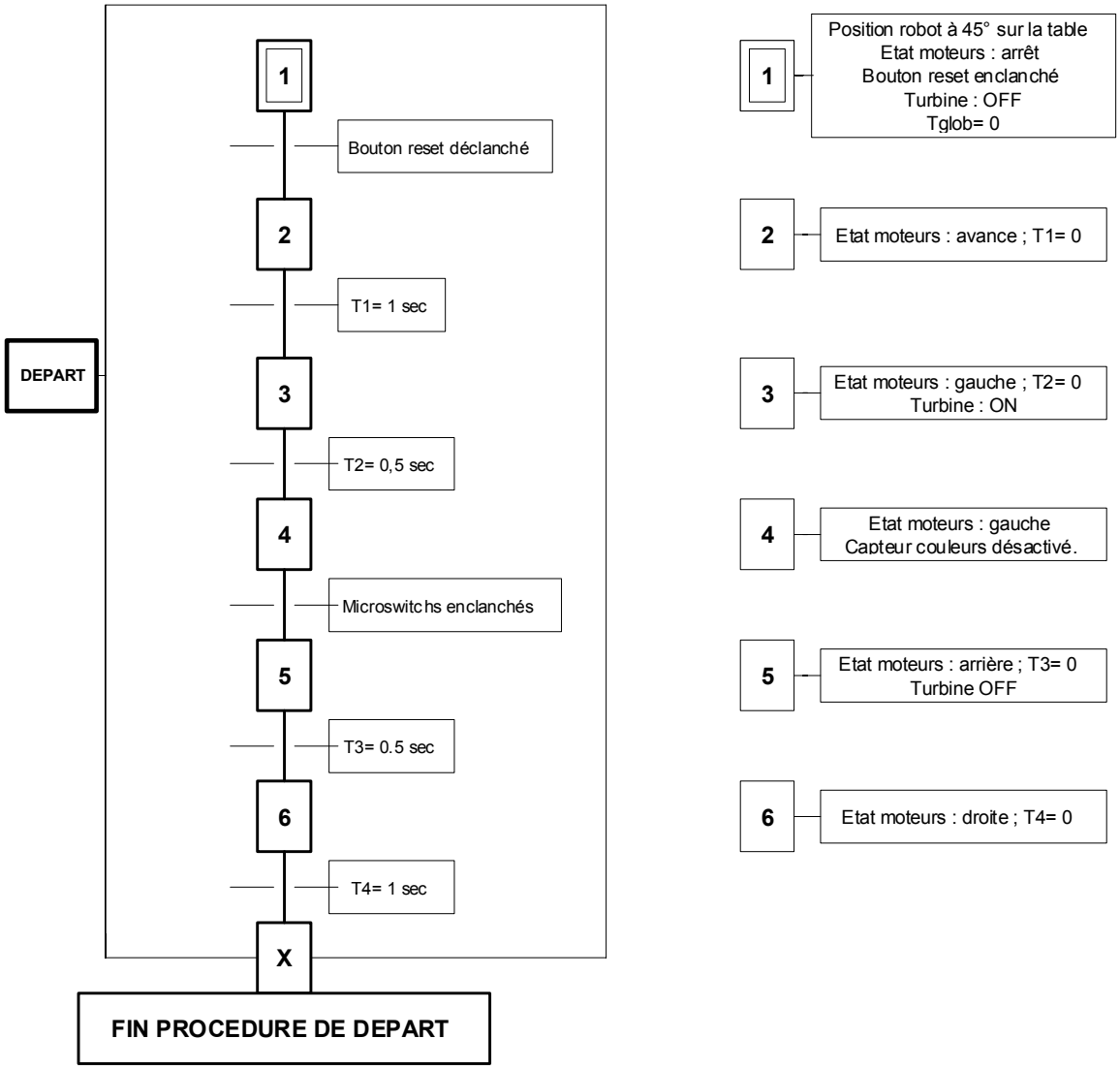
**Procédure de déplacement aléatoire + traitement palets**




PHILIPRONT N
DEPLUS S

## Procédure de départ

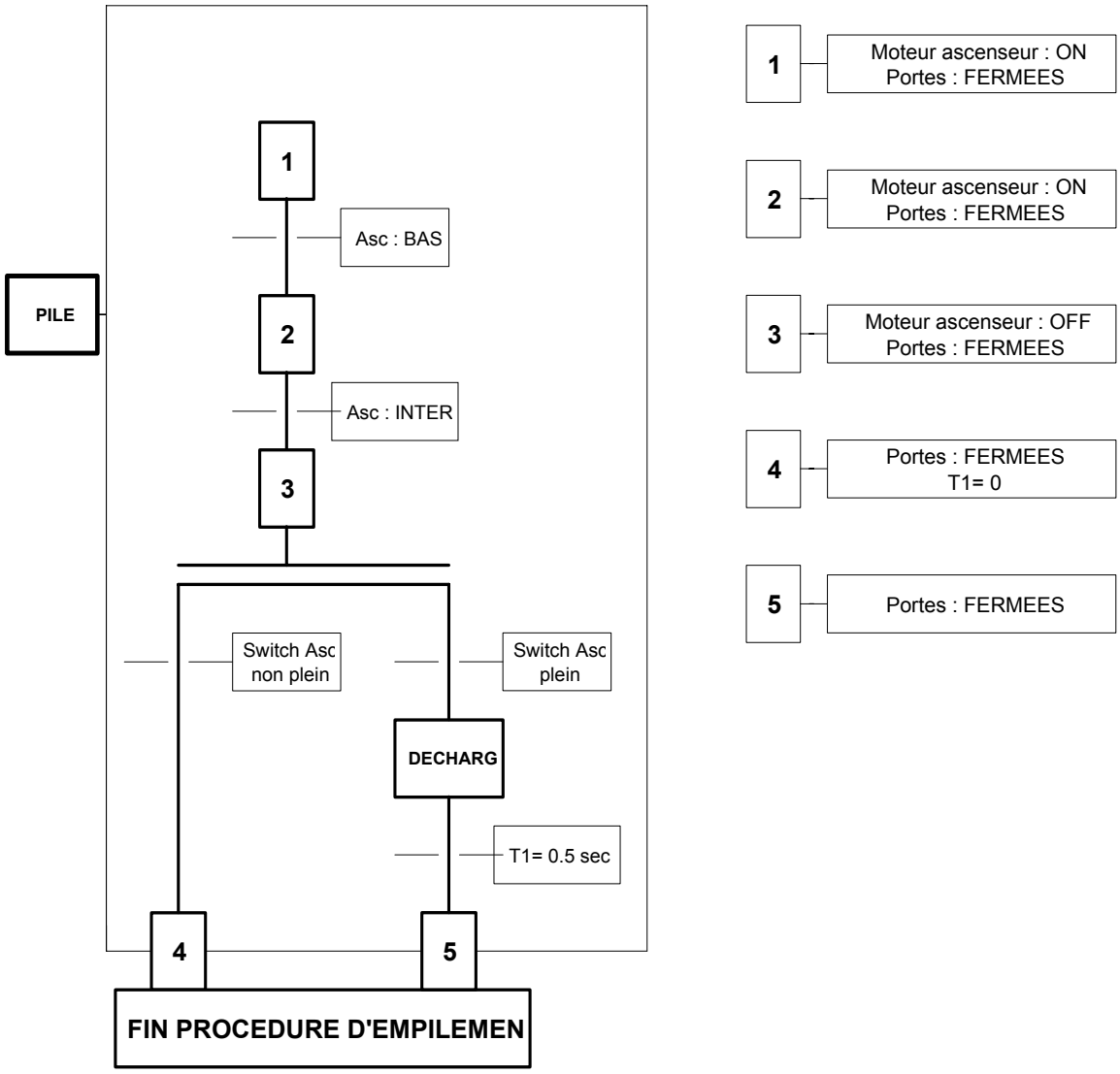
### Grafcet de niveau 1




PHILIPRONT N
DEPLUS S

## Procédure d'empilement

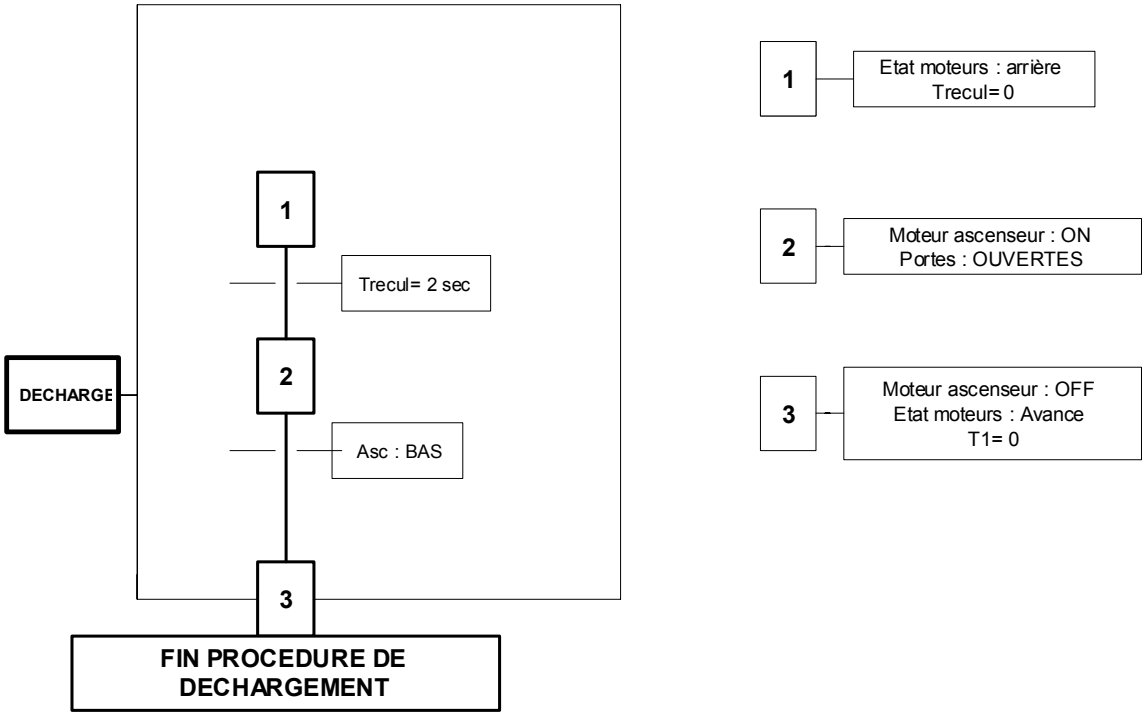
### Grafcet de niveau 1




PHILIPRONT N
DEPLUS S

**Procédure de déchargement**

**Grafcet de niveau 1**




PHILIPRONT N
DEPLUS S

## 4.2 Logique des programmes

Il y a deux grands points concernant la philosophie des programmes réalisés. Premièrement, il faut éviter que le robot ne se bloque sur la table de jeu. Pour cela, le robot donne la priorité au microswitchs de contact. Lorsque qu'un de ces microswitchs est enclenché, le robot recule, tourne sur lui-même et repart sur la table. Le programme (cf. Annexe) analyse également l'état des moteurs, si l'état des moteurs reste trop longtemps le même, on peut supposer qu'il y a un problème, le robot recule, tourne sur lui-même et repart.
Deuxièmement, le robot doit réaliser la tâche qui lui est demandée. Il doit pouvoir traiter les palets correctement. La détection du palet se fait dans la zone de retournement à l'aide d'un télémètre. L'analyse de la couleur se fait par un capteur IR (CNY70). Si la couleur d'un palet détecté est mauvaise, le vérin est actionné et le palet est retourné. La porte de l'ascenseur qui bloquait le palet s'ouvre, le palet entre alors dans la zone d'empilement.
Lorsque l'on ne veut pas faire de pile ; l'ascenseur reste en haut ; le faisceau laser est coupé ; l'ouverture des portes arrière est commandée, et le palet sort du périmètre du robot. Lorsqu'il y a empilement, le palet situé dans la zone d'empilement est bloqué par les portes situées à l'arrière du robot. L'ascenseur descend pour prendre le palet (empilement par le bas) puis remonte en position intermédiaire pour éviter le frottement du palet sur la table. Lorsqu'une pile est constituée, on peut selon la stratégie soit la déposer, soit la conserver pour la déposer à la fin du match. Lorsqu'un 4$^{ème}$ palet se présente, l'ascenseur est en haut lorsque ce palet entre dans la zone d'empilement. Il n'y a pas de contact entre la pile réalisée et ce 4$^{ème}$ palet qui coupe alors le faisceau laser. Les portes situées à l'arrière du robot s'ouvrent et libèrent ce 4ème palet. Les portes se referment alors pour préserver la pile et l'ascenseur se remet en position intermédiaire pour pouvoir bloquer un nouveau palet dans la zone de traitement tout en gardant la pile à un niveau supérieur à celui de la table.
Le traitement du palet est complété par une procédure de départ. Le robot suit au démarrage une trajectoire préprogrammée, les palets cible sont abattus à l'aide de la turbine.
Enfin, si la fin du match approche, les palets encore présents dans le robot doivent être libérés par la procédure de décharge classique. Lors de cette procédure, le robot recule ; l'ascenseur descend ; la pile est déposée sur la table ; les portes situées à l'arrière du robot s'ouvrent. Le robot avance et laisse ainsi la pile sur la table. Si c'est une pile de 3, elle augmente notre score de 5 points.

## 4.3 Les programmes

Sur base des différentes procédures définies par la stratégie, nous avons élaboré plusieurs programmes de complexité croissante.

Nous avons commencé par un programme de qualification ne prenant en compte que la procédure de départ. Il fait tomber les deux palets cibles et s'arrête après avoir touché un bord. Nous assurions 3 points : 2 points des palets cibles plus 1 point pour le palet se trouvant à chaque match devant le robot au départ.

Le programme « sans pile » est celui utilisé durant la coupe de Belgique car la procédure d'empilement n'était pas complètement terminée, tant au niveau mécanique que logiciel.

Le programme « avec pile » fut terminé et utilisé en coupe d'Europe avec succès. Le mode « multi-piles » est le mode qui permettait au robot de lâcher une pile dés qu'elle était faite. En mode « 1 pile », la pile est faite au début du match et nous attendons les dernières secondes pour la lâcher et ainsi s'assurer 5 points.

Nous avons commencé à développer une procédure qui permettait de longer les bords après 60 secondes mais nous n'avons pas pu la terminer. Nous n'avons donc pas utilisé ce programme durant la coupe d'Europe.

Un récapitulatif des différents programmes avec leurs procédures associées.

| Procédures | Départ | Empilement | | longe bords | retourne palet |
|---|---|---|---|---|---|
| | | 1 pile | Multi-piles | | |
| Qualification | X | | | | |
| Match sans pile | X | | | | X |
| Avec pile | X | X | | | X |
| Multi-piles | X | | X | | X |
| 1 pile + longe bords | X | X | | X | X |
| Multi-piles + longe bords | X | | X | X | X |
| Si piston ou capteur couleur ne fonctionne pas | X | X | | | |

A l'instar de l'année passée, les différents programmes informatiques présentés ci-dessous sont écrits en langage C. Le compilateur utilisé est l'AVR GCC (compilateur fourni avec l'Atmega). A la compilation, les programmes exécutables sont convertis - entre autres- en fichiers « rom » et ce sont ces mêmes fichiers que nous introduisons dans l'Atmega 103 à l'aide du logiciel PonyProg. Il est important de signaler que, pour assurer la compilation d'un fichier « .c », il faut y associer un fichier appelé « makefile ».
La structure générale des programmes est identique quel que soit le programme utilisé à savoir :
Définition des librairies
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>

Remarque : l'exécution des programmes présentés dans ce rapport nécessite la présence des fichiers io.h, interrupt.h, sig-avr.h dans le dossier « include » du répertoire « avrgcc ».

Définition des variables

Définition des différents comportements du robot
Définition des différents états des moteurs
Définition de la stratégie
Définition des interruptions
Initialisation des ports entrée/sortie et des timers
Programme principal
Tous les programmes utilisés possèdent le programme principal suivant :

```
int main(void)
{ ioinit();
for (;;);
}
```

Contrairement à ce qui a été dit l'année passée, à chaque seconde, l'ATMEGA (103) effectue non pas 7228 interruptions mais bien 14456. Une erreur de programmation conduisait à incrémenter le compteur d'interruptions tous les 2 cycles seulement.
A chaque interruption, les fonctions bilan(), capteurs() et MAJmoteurs() sont analysées. La fonction bilan() permet de définir la stratégie adoptée, la fonction capteurs() détermine tous les comportements du robot disponibles pour exécuter celle-ci et la fonction MAJmoteurs() reprend la définition des états des moteurs qui permettent de définir les comportements du robot.

Les stratégies des différents programmes sont expliquées en détail dans les GRAFCETS. Nous avons pris soin de détailler (en commentaire) chaque fonction dans les différents programmes.

# 5. Vision

Nous avons envisagé deux voies parallèles pour doter le robot d'un système de vision. Le but était bien évidemment de permettre au robot de se diriger directement vers les palets. Cela devait nous permettre de traiter un maximum de palets dans le temps imparti ce que ne permet pas forcément une trajectoire aléatoire du robot. Les deux voies totalement différentes furent les suivantes :

- Utiliser un système de faible coût acheté tel quel dans le commerce : La CMUCAM
- Développer complètement le système de reconnaissance des palets avec le concours d'une société extérieure : Capflow.

## 5.1 La transmission d'informations via le port RS-232

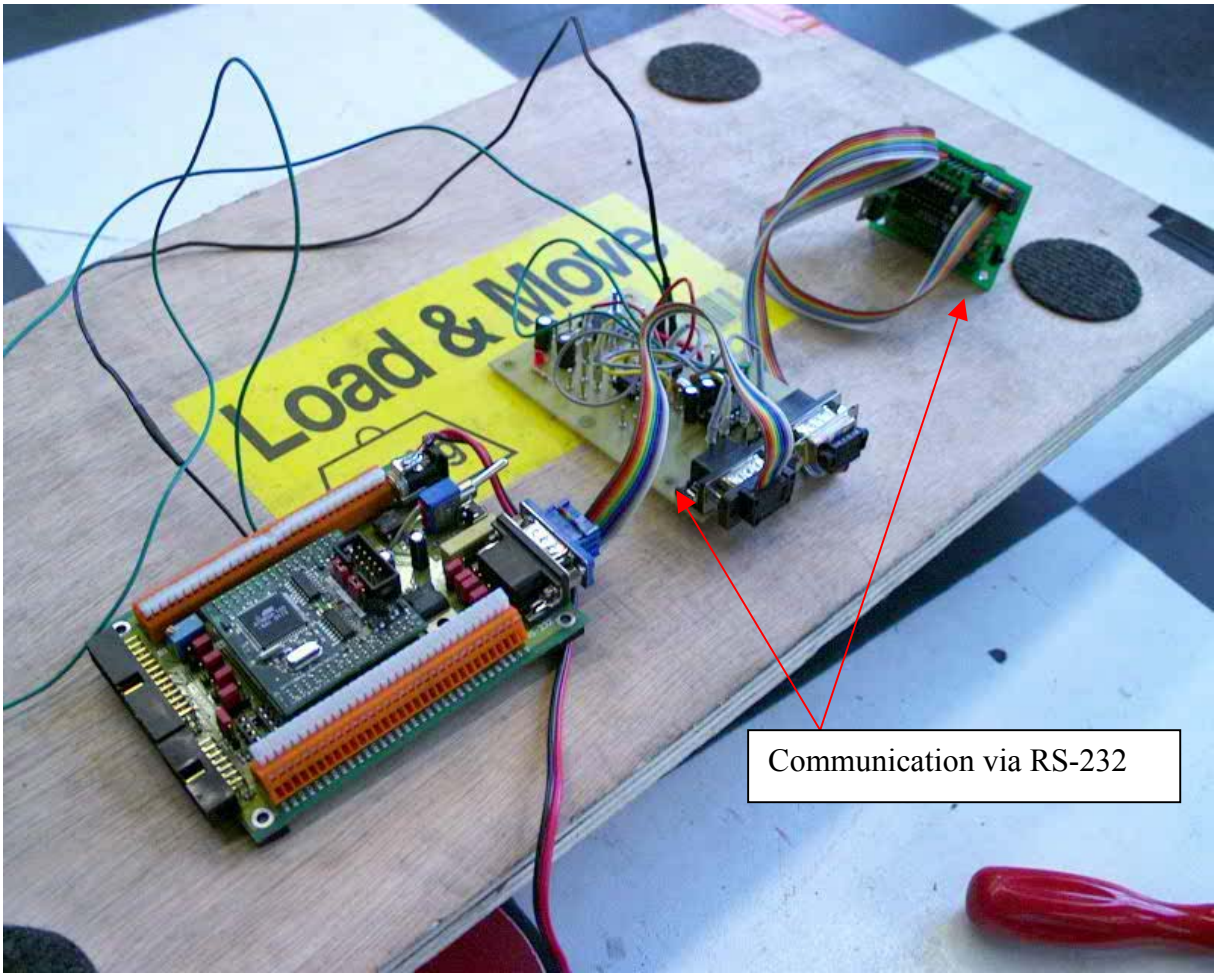
Le port RS-232 se présente sous l'aspect d'un connecteur DB9 (Port COM) à l'arrière du microcontrôleur. Ce port permet d'envoyer des informations par paquets. Le microcontrôleur permet l'envoi de Niveau de 5 V sur les ports situés latéralement sur la carte support. Le périphérique RS-232 permet lui d'envoyer des chaînes de caractères ou nombre soit sur un ordinateur soit sur un autre port COM1.
Le transfert d'informations via ce port est un moyen couramment utilisé lors de dialogue entre éléments électroniques. Une liaison de ce type était indispensable pour communiquer avec la CMUcam ainsi qu'avec des télémètres SHARP dont l'achat avait à un moment été envisagé.
L'intérêt de communication via ce port est réel. Outre la nécessité de communiquer via ce mode à certains éléments électroniques, il peut servir dans la vérification du comportement du robot en envoyant des informations sur l'hyperterminal d'un ordinateur. Ainsi par exemple, le programme informatique réglant le comportement du robot a pu être testé sans celui-ci sur une autre plate-forme isolée ou sur le robot sans la présence des moteurs.
Le comportement du robot peut être simulé par des microswitchs mis sur une plate-forme et sa réaction contrôlée via l'hyperterminal sur un PC. Pour exemple, deux microswitchs sont placés sur les ports PA4 et PA5 de la plate-forme mise à nue. Ils symbolisent les deux microswitchs de contact à l'avant du robot. Lorsque l'on appuie sur un de ces deux microswitchs, on voit apparaître sur l'écran de l'ordinateur un « R » signalant que le robot recule. Une demi- seconde plus tard, un « D » apparaît. Il symbolise une rotation à droite du robot. Enfin, deux secondes plus tard un « A » signale que le robot se met en marche. De la même manière, ce port à permis de vérifier que le moteur passait bien par un état arrêt durant 150 interruptions lorsqu'un moteur changeait de sens et donc d'état.
Le comportement du robot a ainsi pu être simulé sans sa présence et on peut ainsi vérifier que l'on entre bien dans l'ensemble des boucles voulues. De plus, ce port permet de détecter facilement à quel endroit à lieu une erreur de logique.

## 5.2 Aspect informatique de la transmission de données.

Informatiquement le port RS-232 transfère des informations de 8 BITS par caractère sous format ASCII via le connecteur DB9 situé à l'arrière avec une certaine vitesse de transmission. Cette vitesse de transmission et la mise en forme ASCII d'un caractère dépend de la vitesse du micro-processeur et de la vitesse d'envoi des informations. Pour comprendre les informations envoyées par ce port il faut que la vitesse de réception soit la même que celle d'émission et que le codage ASCII soit correctement effectué à la vitesse adéquate du micro-processeur.
La logique du programme informatique est la suivante. La phase d'initialisation précédant l'envoi d'interruptions est complétée par une procédure d'initialisation du port RS-232. Lors de l'envoi de caractères, des procédures transfèrent après codage l'ensemble des informations du caractère (8 bits) dans un Buffer. L'envoi étant ensuite réalisé en dirigeant ces informations à l'adresse du port RS-232. De la même manière, lors de la réception d'informations, d'autres procédures se chargent de réunir les 8 bits constitutifs d'un caractère pour les traduire sous forme classique. L'envoi et la réception d'informations demandent du temps. Le programme repris de l'année précédente parcourrait l'ensemble des lignes de commande lors de chaque interruption. Le stockage dans le buffer d'une chaîne de caractères demandait plus de temps que le temps disponible lors d'une instruction. C'est la raison pour laquelle il était impossible d'envoyer ou recevoir plus de deux caractères lors d'une interruption. Le programme réalisé par Laurent Pinchart de la société Capflow possède une philosophie différente. Les interruptions se font de manière indépendantes du programme. Ainsi le programme est parcouru, une interruption intervient, le temps est incrémenté et le programme continue naturellement son cours.
L'envoi de caractères se fait dans le programme par l'envoi de la commande suivante :
« uart_puts(« A ») ; » pour envoyer le caractère A.

## 5.3 Application du RS-232

Le port RS-232 est utilisé pour l'envoi d'informations de l'Atméga au PIC (cf. chapitre **VISION3**). Les paramètres envoyés permettront de calibrer la CMUcam en fonction des conditions de luminosité. Le PIC recevra ces informations et commandera la CMUcam via une autre liaison RS-232.




## 5.4 CMUCam

### 5.4.1 Introduction

La CMUCam est une caméra couleur de petites dimensions qui comprend outre le capteur numérique CMOS (de marque Omnivision) et l'optique, un micro-controleur dans lequel est implémenté un ensemble de traitement d'images préprogrammés. La résolution de la caméra est de 80 x 143 pixels.

La caméra est prévue pour communiquer avec un microcontrôleur ou un microprocesseur externe. Les informations disponibles en sortie en provenance du microcontrôleur de la caméra permettront de diriger le robot.

Nous avons dans un premier temps recensé toutes les fonctions possibles de la caméra et ensuite essayer de les utiliser au mieux pour corroborer nos besoins. La caméra est fournie avec un logiciel qui permet d'accéder à l'ensemble des fonctions de celle –ci via un port série (RS 232).

Parmi les fonctions offertes, les suivantes ont semblées utiles à notre application.

possibilité de rechercher une couleur définie par des paramètres RGB
Choix du mode de fonctionnement pour les couleurs (RGB ou YUV)
Réglage du gain
Réglage des dimensions de la zone de traitement
Nombre d'images par seconde

### 5.4.2 Recherche d'un palet

La recherche d'un palet va passer par la recherche d'une couleur. Cette couleur va être définie par les paramètres suivants $R_{min}$, $R_{max}$, $G_{min}$, $G_{max}$, $B_{min}$, $B_{max}$. Le logiciel nous permet de connaître les valeurs des composantes RGB pour une image prise par la caméra. Ces paramètres constituent des bornes minimales et maximales pour chaque composante R, G, B. Ils sont définis pour le rouge et pour le vert indépendamment. Cette détermination des paramètres consiste donc en une étape décisive pour un bon fonctionnement du système.

La détection de la couleur voulue apparaît sous la forme d'un rectangle définie par les coordonnées de ses extrémités supérieure gauche et inférieure droite ainsi qu'un intervalle de confiance. Ce rectangle englobe une zone où il trouve la couleur demandée. Ce sont les paramètres de ce rectangle qui sont disponibles en sortie de la caméra pour la commande de direction du robot. Cette sortie se fait sous la forme d'une ligne structurée comme suit :

$x_1\ y_1\ x_2\ y_2$ intervalle de confiance

Très tôt, nous avons remarqué que ces paramètres étaient fortement influencés par les conditions d'éclairage. Il est donc très difficile de déterminer des paramètres qui vont pouvoir être employés dans n'importe quelles conditions. Le problème s'est avéré moins flagrant pour la couleur rouge que pour la couleur verte. Dans le cas du rouge, nous avons pu déterminer des paramètres fonctionnant dans 90 % des cas. Tandis que pour le vert, le réglage s'est avéré bien plus délicat. Il devait en effet être effectué en début de chaque utilisation en fonction de l'éclairage implanté.

Nous avons aussi remarqué que le nombre d'images par seconde était un facteur permettant de mieux discerner les couleurs au détriment de la rapidité bien évidemment.

### 5.4.3 Contrôle d'une plate – forme de test

Très rapidement, nous avons pu tester le fonctionnement de la caméra grâce à une petite plate-forme de tests.

Cette plate-forme était constituée des éléments suivants
La CMUCAM
Un PIC
Deux moteurs DC

Nous avons développé un programme en BASIC qui a été ensuite implémenté dans le PIC. Ce programme permettait de diriger le robot vers une des deux couleurs prédéfinies par ses paramètres dans le PIC.

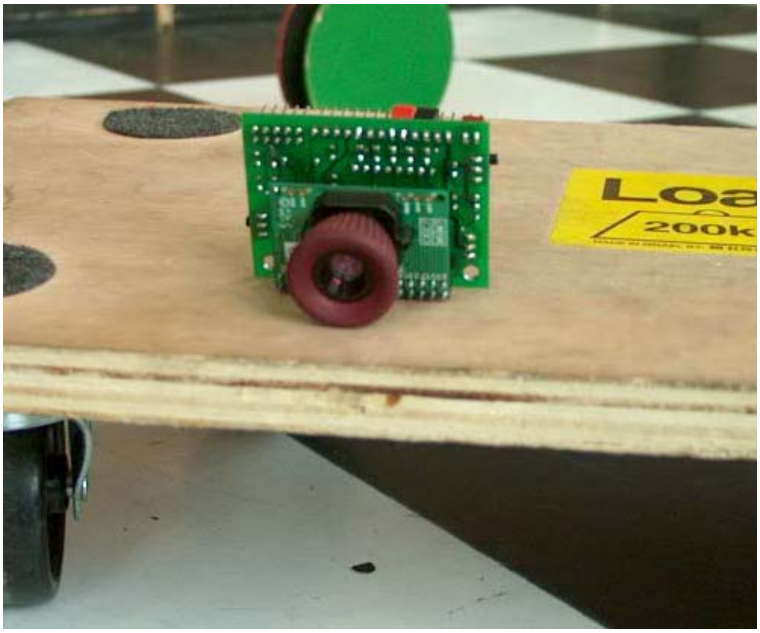
#### 5.4.3.1 Programme du PIC en basique

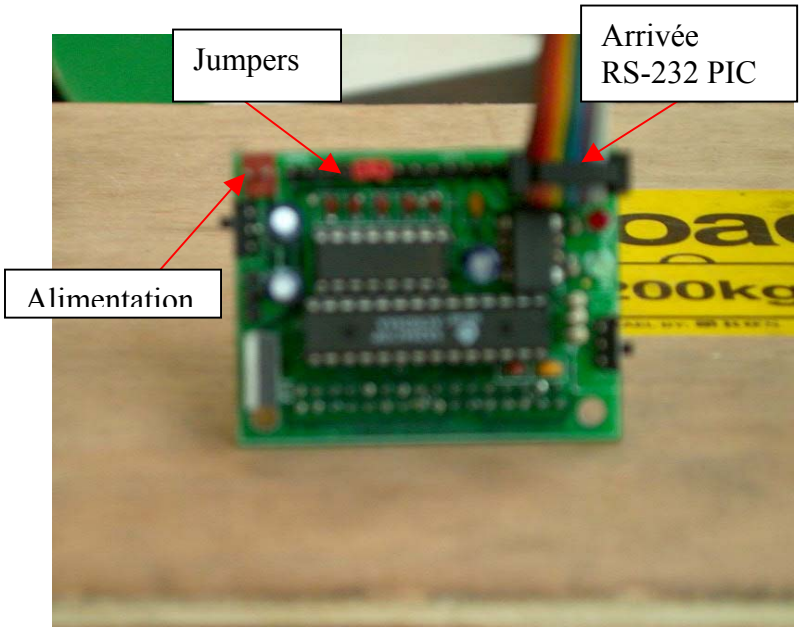
Le PIC fait l'interface entre l'Atmega (cf. Annexe) et la CMUcam. Le programme envoie des ordres à la caméra et traite les informations reçues. Lorsqu'un palet est trouvé, le PIC calcule sa position par rapport au centre de l'image. Un signale de 5 Volts est envoyé sur deux entrées de l'Atmega définissant un code logique.
Le code est le suivant :

| | |
|---|---|
| **1-0** | **Palet à gauche** |
| **0-1** | **Palet à droite** |
| **1-1** | **Palet droit devant** |
| **0-0** | **Pas de palet** |

C'est l'Atmega qui choisira si ces informations sont à prendre en compte ou non et qui dirigera le robot vers la cible trouvée.

#### 5.4.3.2 Description de CMUcam

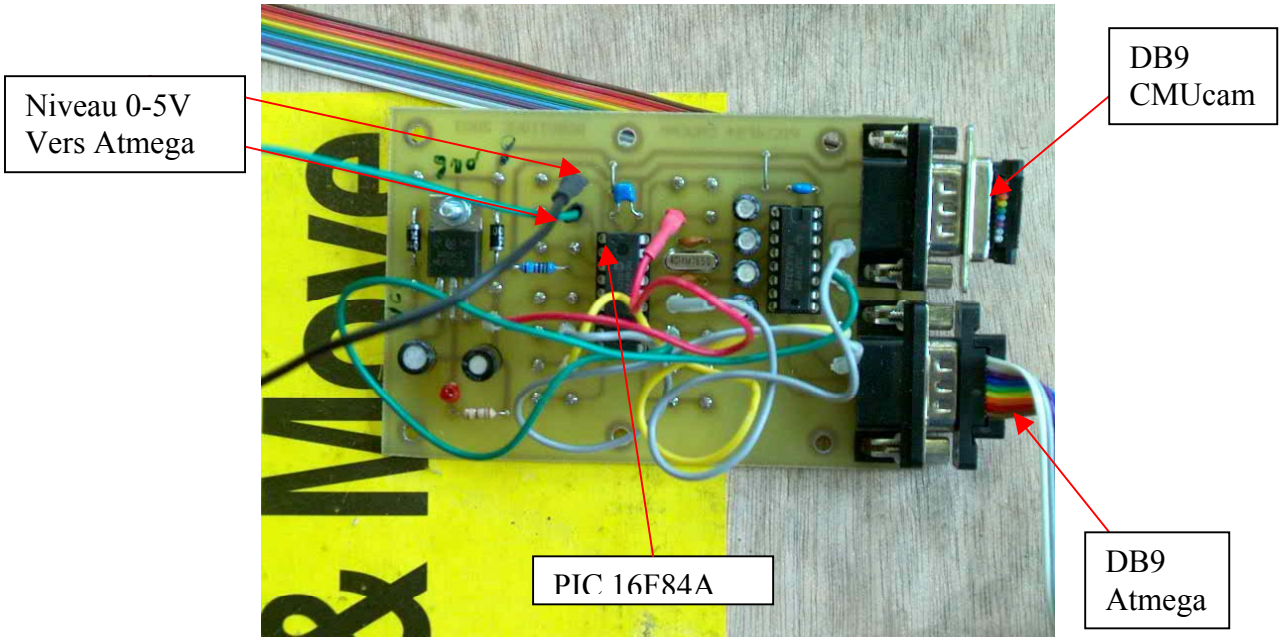





La CMUcam est commandée par le PIC. La première phase consiste en une initialisation de la caméra. Le PIC se charge ensuite de régler les paramètres de contraste et de luminosité de la caméra. Enfin, le PIC transmet les paramètres de la couleur recherchée et la détection peut commencer.
La caméra renvoie des informations sur la taille de l'objet trouvé. La couleur de cet objet est comparée à celle recherchée. Ce paramètre est quantifié par un intervalle de confiance. C'est ce paramètre qui sera analysé par le PIC. Si l'intervalle de confiance est supérieur à une valeur limite l'objet est pris en considération, la position du centre de masse est analysée. Le robot se dirigera alors vers cet objet.

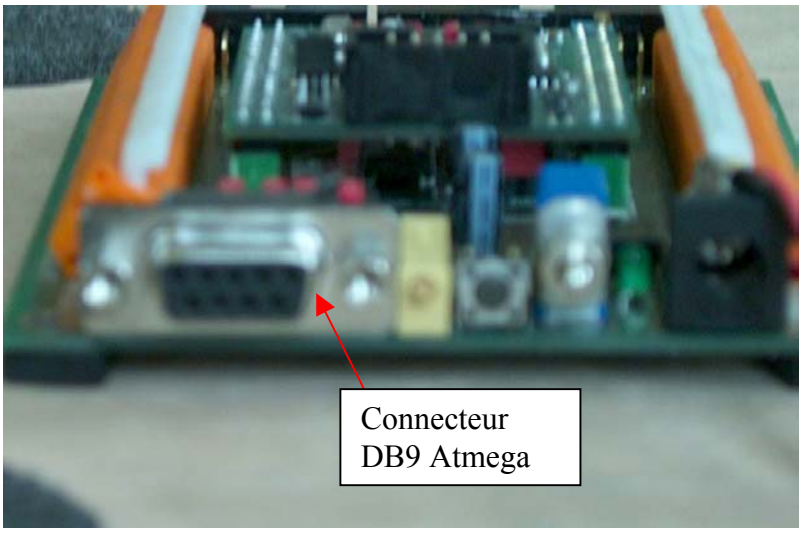
Les jumpers définissent la vitesse d'informations entre le PIC et la Caméra. Lorsque les 2 jumpers sont mis, la vitesse de transmission est de 9600 Baud/sec. La vitesse peut augmenter jusqu'à 115200 Baud/sec.
Ces informations sont transférées via le Flat Cable.

#### 5.4.3.3 Description du PIC




Le PIC reçoit des informations de l'Atmega sur les paramètres de couleurs à rechercher. Il envoie ensuite des ordres structurés à la CMUcam. Les informations reçues de la caméra sont traitées. Il y a comparaison entre l'intervalle de confiance de l'objet trouvé et la valeur seuil reçue de l'Atméga. Si un objet est trouvé, le PIC suit alors le mouvement du centre de masse. Le PIC envoie une commande à l'Atmega sous la forme de niveau pour commander les moteurs. Lorsque le centre de masse se trouve au centre de l'image vue par la caméra, le robot se dirige à grande vitesse vers le palet trouvé.

#### 5.4.3.4 Rôle de l'ATMEGA

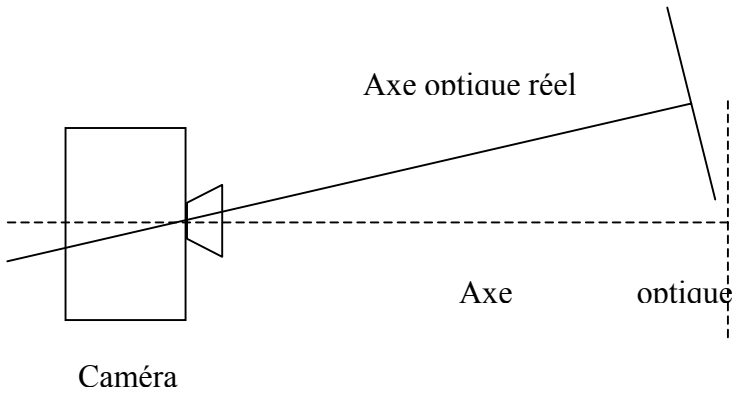



L'Atméga indique au PIC les paramètres de couleurs recherchées. Il n'y a pas de retour d'informations.

### 5.4.4 Constatations

Deux problèmes conséquents apparaissent lors des premiers essais.

Le premier est une déviation de l'axe optique. La caméra ne voit pas parfaitement devant elle comme on aurait pu le supposer (en pointillés sur la figure ci-dessous), la zone dans laquelle elle voit est légèrement décalée vers la gauche d'un angle d'environ 7° (en trait continu sur la figure).



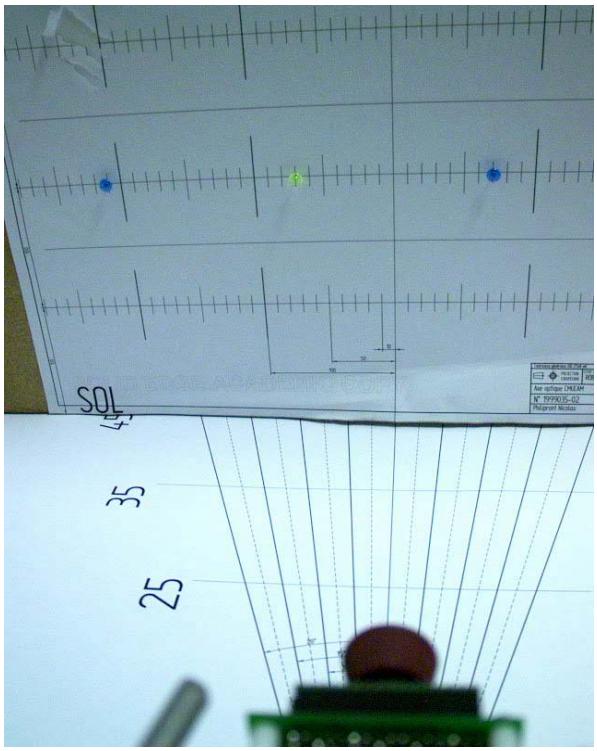



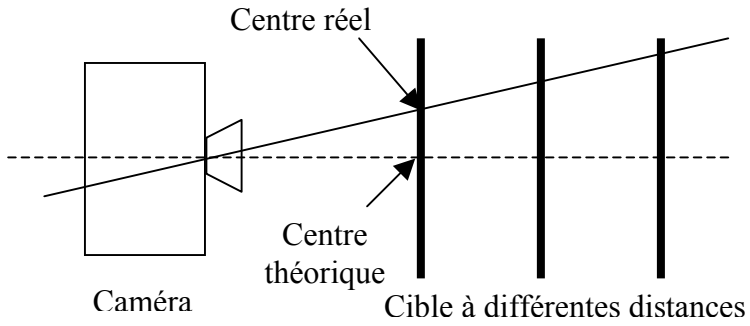

**Figure 30:détermination de la déviation de l'axe optique de la CMUCam**

L'angle a été déterminé par des essais en utilisant une cible sur laquelle des repères étaient dessinés. En plaçant cette cible à différentes distances de manière parfaitement perpendiculaire à l'axe optique théorique, nous avons pu recalculer pour chaque distance, le centre réel de l'image que voyait la caméra. Sur les photos ci-dessus la zone de vision est

celle définie par les punaises bleues. Le centre de l'image de la caméra (punaise jaune) est fortement décalé du centre de la cible.

Connaissant, la position du centre théorique et du centre réel ainsi que la distance entre la cible et la caméra, nous pouvons par la trigonométrie déterminer l'angle de décalage de l'axe optique. En répétant cette procédure pour plusieurs distances entre la caméra et la cible de manière à confirmer les résultats, nous trouvons donc un angle de déviation de l'axe optique qui est donc une moyenne.

Les essais ont été réalisés avec le plus de précision possible, cependant l'angle trouvé n'est qu'une estimation plus ou moins bonne de l'angle réel de déviation. La déviation a été corrigée par un support mécanique adapté. Le fait que l'estimation de l'angle de déviation n'est pas d'une extrême précision ne pose pas de problème, vu que nous n'aurions de toute façon pas pu le reproduire au centième de degré lors de la fabrication du support correctif de la caméra.




Le second est le fait que la caméra ne peut pas distinguer deux palets de couleur identique cote à cote. En effet, la manière dont travaille la caméra nous donne un rectangle englobant les deux palets. Nous avons essayé de corriger ce problème en réduisant la zone dans laquelle la caméra voyait. Cette réduction permettait bien de ne plus détecter qu'un seul palet mais réduisait considérablement les éventualités de détection. La lenteur de la caméra fait qu'avec cette réduction de zone de détection, la caméra éprouve d'énormes difficultés à détecter un palet se présentant à elle. Des tests ont été réalisés, la CMUcam détectait bien le rouge, mais le robot se perdait souvent sur la table lorsqu'il détectait des perturbations. Il y aurait eu moyen d'optimiser le comportement de cette caméra sur le robot, mais cela ne constituait pas la tâche la plus urgente. La CMUcam n'a donc pas été utilisée sur le robot.

## 5.5 Capflow

### 5.5.1 Introduction

La société Capflow S.A. par l'intermédiaire de Pascal REPJUK et Laurent PINCHART s'est investie dans le développement d'un système de vision adapté à notre application.

Le matériel est une caméra CAMELEON développée par la société. Cette caméra comprend outre l'optique et un capteur CMOS, toute l'électronique nécessaire à un traitement hardware des images. Ses dimensions sont de 3 pouces sur 3. Sa résolution maximale est de 1280 x 1024. Elle dispose d'une interface Cameralink.

Le développement s'est effectué en deux étapes :

1. Recherche de l'algorithme de traitement d'images le plus adéquat à notre application. Pour cela, le développement se fait en software au moyen d'une carte d'acquisition EURESYS Grablink Value (interface Cameralink) et d'un traitement programmé en C.

2. Programmation hardware (VHDL) du traitement développé précédemment de manière software. Cette programmation a été réalisée par un étudiant électricien Vincent Gaudissart dans le cadre de son travail de fin d'études.

L'équipe des étudiants mécaniciens a participé à la recherche de l'algorithme et à sa programmation en C avec l'aide de Pascal REPJUK. La programmation hardware de la caméra a été quant à elle réalisée par Vincent Gaudissart avec l'aide de Laurent PINCHART.

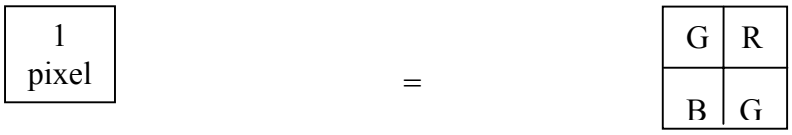
### 5.5.2 Algorithme de traitement d'images

Nous allons décrire dans ce paragraphe l'algorithme programmé en C. Celui-ci a subi quelques modifications lors de son implémentation en langage VHDL. Ces modifications consistent en des optimisations ou des modifications liées aux spécificités ou nécessités de la programmation hardware.

Le programme C tel qu'il a été développé et avant modifications en vue de son optimisation et de son implémentation en hardware est fournie en annexe.

Au départ, la résolution de la caméra était de 1280 x 1024, des problèmes de lenteur étant rapidement apparus, il a été décidé de la réduire à 320 x 256. Cette résolution est plus que suffisante pour l'application qui nous intéresse.

Une couleur (R, G-bleu, G-rouge, B) est associée à chacun des pixels et donc 4 pixels sont nécessaires pour recréer un pixel RGB. Cette description succincte suffira pour comprendre le traitement d'images.




Comme pour la CMUcam, on va travailler avec les couleurs pour la détection des palets et pas les formes. Le but du traitement d'image est de ne plus voir que les palets, rouges ou verts, en éliminant tous les éléments autour tel que le damier de la table de jeu.

Le programme comprend 2 parties :

1. Une partie qui permet la détermination des valeurs des paramètres RGB d'un palet.
2. Une partie qui réalise le traitement d'images a proprement parlé (détection des palets)

**1ère partie du programme**

Cette partie comprend un calcul des valeurs RGB. Pour effectuer ce calcul, il est nécessaire de placer un palet devant l'objectif de la caméra. Trois zones de l'image ont été définies dans le programme, chacune de ces zones correspond à un des paramètres R, G ou B. Le programme effectue dans chacune de ces zones une moyenne sur les couleurs des pixels. Cela permet d'avoir une estimation des valeurs de R, de G et de B pour chaque couleur de palet. Ici aussi, la luminosité a beaucoup d'influence sur les valeurs de ces paramètres. Dès maintenant, nous voyons la nécessité d'effectuer une calibration en début de chaque match.

**2e partie du programme**

Cette partie consiste en le traitement d'image à proprement parler. C'est avec la succession de traitement décrit ici que nous allons pouvoir isoler les palets.

Les traitements à effectuer sont les suivants, dans l'ordre :

- une inversion de l'image (robot_swapy)
- une quantification (robot_quantifie)
- une érosion (robot_erode)
- un remplissage horizontal (robot_hfill)
- une labellisation (robot_labelise)

**Figure 31 : cas se présentant sur l'aire de jeu**

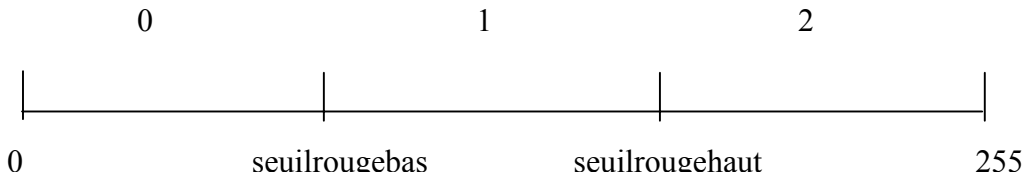
### Inversion d'image

Ce traitement consiste simplement à effectuer une symétrie orthogonale de l'image par rapport à un axe horizontal situé en son milieu. Par défaut l'origine de l'image se trouve dans le coin supérieur gauche, vu que le robot avance sur l'aire de jeu, il faut donc commencer par traiter les pixels se trouvant dans le coin inférieur gauche. Ce traitement permet en retournant l'image de résoudre ce problème.

### Quantification ou binarisation

Pour la quantification, nous avons divisé la gamme de chaque paramètre en trois. Cette gamme s'étend de 0 à 255. Nous avons défini pour chacun des paramètres R, G et B, deux seuils, un seuil bas et un seuil haut.

Par exemple pour le rouge




Nous allons balayer l'image selon x et selon y en évaluant chacun des pixels. Pour chaque pixel, nous allons comparer les valeurs de ses sous pixels aux seuils bas et haut prédéfinis de la couleur adéquate. Si la valeur du pixel est inférieure au seuil bas, nous imposons une variable Rb à 0, si la valeur du pixel est comprise entre le seuil bas et le seuil haut, on impose la valeur de Rb =1 et si elle est supérieure au seuil haut, la valeur de Rb est imposée à 2.
On fait cette opération pour les 3 composantes R, G, B. A noter qu'on ne considère qu'un deux des sous-pixels verts, celui donne un vert plus proche du bleu. Cela pour éviter au maximum les interactions entre le rouge et le vert qui pourrait causer des problèmes de détection.
Pour chaque pixel, nous avons donc une information Rb, Gb et Bb dont les valeurs respectives sont soit égales à 0, à 1 ou à 2.

A partir de ce moment là, nous allons utiliser une table de quantification. Cette table a pour but de faire apparaître les palets et de faire disparaître tous les éléments se trouvant autour qui ne sont pas des palets (damier,…).

Nous allons pondérer les valeurs des composantes Rb, Gb, Bb par la formule suivante :

$$index = Bb + Gb \times 4 + Rb \times 16$$

L'index nous repère dans le vecteur. Nous imposons les conditions sur Rb, Gb et Bb pour qu'on ait du rouge ou du vert. Le tableau suivant reprend des exemples de valeur de Rb, Gb et Bb qui associée correspondent à une couleur.

| **Rb** | **Gb** | **Bb** | **Index** | **Couleur** |
|---|---|---|---|---|
| 2 | 1 | 1 | 37 | Rouge |
| 2 | 1 | 0 | 36 | Rouge |
| 2 | 0 | 0 | 32 | Rouge |
| 1 | 1 | 1 | 21 | Vert |
| 1 | 1 | 0 | 20 | Vert |
| 0 | 1 | 0 | 4 | Vert |
| 0 | 1 | 1 | 5 | Vert |
| 1 | 1 | 1 | 25 | Vert |
| 1 | 1 | 0 | 24 | Vert |

Chaque fois que l'une des combinaisons prédéfinies des 3 paramètres se produira, l'index associé référencera vers une couleur. Dans ce cas, les quatre sous pixels seront imposés à cette couleur, rouge ou verte. Dans tous les autres cas, la couleur du pixel sera le fond de l'image, c'est-à-dire noir.

La valeur dans le programme pour la couleur rouge est 2 et pour la couleur verte 1, le fond est égal à 0. On initialise donc la table à 0, et on vient remplacer par 1 ou 2 les éléments correspondant à un index prédéfini.

Nous avons aussi besoin de modifier quelque peu les valeurs associées aux couleurs pour pouvoir visualiser les choses sur l'écran. En effet, les valeurs que nous avons imposées précédemment pour le fond, le rouge et le vert ne nous permettent pas de visualiser quelque chose. On va donc réaliser un affichage noir et blanc dans lequel le fond sera noir, le rouge sera blanc et le vert sera gris. On va donc simplement balayer l'image en remplaçant les valeurs 1 et 2 que l'on trouve par respectivement 128 (gris) et 255 (blanc). Les valeurs 0 ne sont pas modifiées, on aura donc un fond noir.

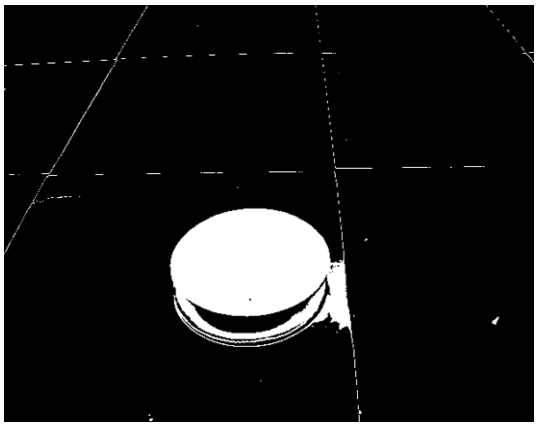


**Figure 32 : traitement après quantification**

**Erosion**

Pour l'érosion, on va balayer toute l'image verticalement et horizontalement. On va comparer un pixel courant à ses 8 voisins. Si les 8 voisins ont la même couleur que le pixel courant, on ne change pas la couleur de ce pixel. Si par contre, un des 8 voisins n'a pas la même couleur que le pixel courant, on impose le pixel courant à zéro, c'est-à-dire qu'on l'impose égal au fond de l'image. Par ce traitement, on élimine tous les parasites de l'image, on fait disparaître tous les pixels isolés qui pourraient induire en erreur.

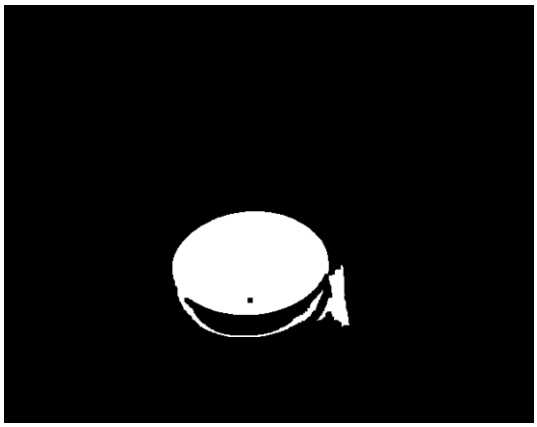


**Figure 33 : traitement après érosion**

**Remplissage horizontal**

Ce traitement est destiné à combler d'éventuels trous au milieu d'une masse de pixel représentant une couleur. Son usage ne s'est pas avéré indispensable en final. Le principe est le suivant : on balaye l'image horizontalement et verticalement, pour une ligne, on compare le pixel courant à un pixel situé n pixels plus loin sur la même ligne. Si ce pixel est égal au pixel courant, on impose la couleur de tous les pixels entre le pixel courant et le pixel se trouvant n pixels plus loin égale à la couleur du pixel courant.

**Labellisation**

Dans ce traitement, on cherche à distinguer les palets. On a donc sur l'image un ou plusieurs palets. A priori, quand le programme balaye l'image et qu'il trouve un segment d'une couleur, il ne sait pas à quel palet il appartient. Cette information est indispensable pour diriger le robot. On va donc effectuer ce qu'on appelle un collage, on va balayer l'image de bas et haut et attribuer chaque segment à un palet. Lorsqu'on rencontre pour la première fois un segment d'une couleur, on l'attribue à un palet. Ensuite pour s'assurer qu'un segment est collé à un autre, on s'assure que son centre est compris entre les extrémités du précédent.

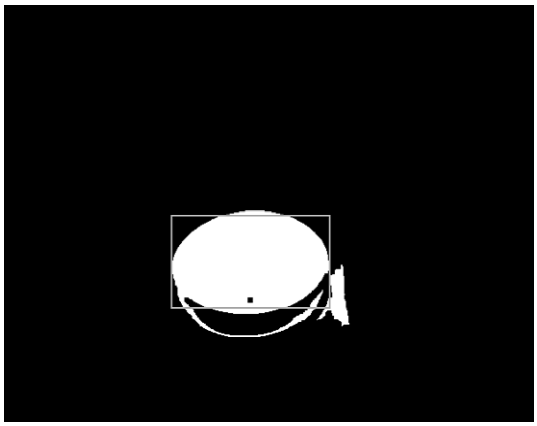


**Figure 34 : traitement après labellisation**

**Commande**

Maintenant que nous avons attribué tous les segments à leur palet respectif, nous pouvons diriger le robot. Le principe de commande est assez simple. Il a été choisi à l'identique pour les deux systèmes de vision de manière à pouvoir les intervertir aisément. La commande du robot repose sur une commande par 2 niveaux TTL, un correspond à la direction gauche, l'autre à la direction droite. La direction du palet se fait par rapport au centre du palet. Cette partie du programme n'a jamais été testée en software.

### 5.5.3 Programmation hardware

On observe quelques différences entre le traitement software et le traitement hardware.

Premièrement, le traitement hardware ne se limite pas à une résolution de 320 x 256 mais à 640 x 512.

Deuxièmement, le remplissage horizontal a été implémenté mais pas utilisé car nous nous sommes rendu compte qu'il n'était pas nécessaire.

Troisièmement, l'inversion d'image, traitement impossible à réaliser en hardware car il nécessite de stocker une image complète en mémoire, a été réalisée mécaniquement en fixant la caméra à l'envers.

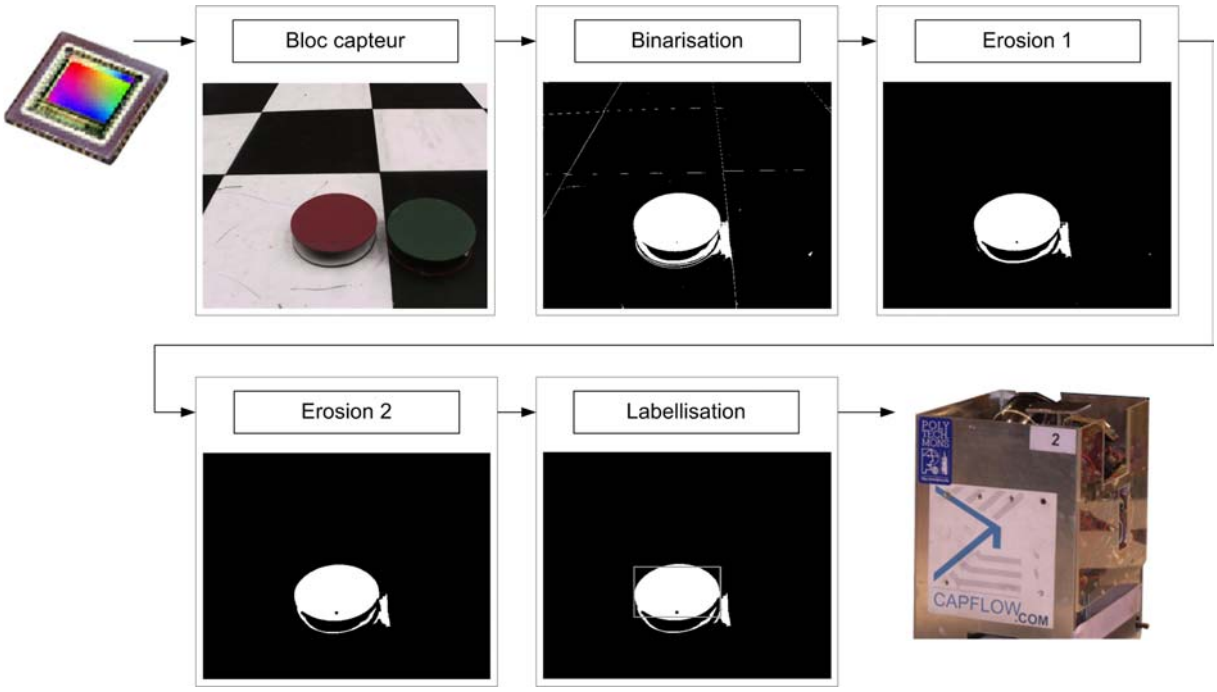



**Figure 35 : chaîne de traitement**

Le traitement hardware effectue les opérations suivantes (Figure 35) :

- Compression 10 bits -> 8 bits par pixel (le capteur sort une valeur sur 10 bits pour chaque pixel, et il est plus simple de traiter une valeur 8 bits). Cette compression s'effectue soit de manière linéaire (on garde uniquement les 8 bits de poids fort), soit de manière logarithmique, pour obtenir une meilleure précision au milieu de la plage de valeurs.

- Quantification, identique à celle en C
- Erosion (deux fois)
- Labellisation

La programmation hardware a été achevée juste avant la coupe de Belgique, n'ayant pas fait suffisamment d'essais nous avons décidé de ne pas l'utiliser pour le concours. Nous avons donc opté pour une trajectoire aléatoire sur l'aire de jeu

### 5.5.4 Calibration

Ce système de vision CAMELEON s'est avéré beaucoup plus performant que la CMUCam. Cependant certains problèmes subsistent encore. Notamment, l'influence de la luminosité sur les paramètres RGB. Pour résoudre ce problème, étant donné l'impossible de réaliser une calibration matérielle de la caméra avant chaque match (nécessité de disposer d'un PC, temps,…), nous nous sommes penchés sur une procédure de calibration automatique. Pour ce faire, le calcul d'un histogramme pour chaque composante de couleur (R, G et B) a été implémenté en hardware. Sur base de ces histogrammes, il est possible de déterminer la valeur du pixel le plus exposé dans l'image (valeur maximale de l'image), et d'allumer une LED si cette valeur est comprise dans un intervalle donné (240-250, juste avant saturation).

Ceci permet de régler l'ouverture de l'iris de la caméra en tournant la bague jusqu'à ce que la LED s'allume. Nous obtenons alors une luminosité relativement constante, sans nécessiter la connexion à un PC pour calibrer

La seconde partie de l'algorithme de calibration calcule, toujours sur base des histogrammes, les valeurs moyennes pour chaque composante couleur. Lors de ce calcul, la caméra doit voir la table de jeu (noir et blanche) sans aucune interférence (palet, robot adverse). Ces valeurs doivent être identiques, puisque la caméra ne voit que du noir et du blanc, pour lesquels les 3 composantes couleurs sont identiques. En pratique, cette identité n'est pas vérifiée, car les pixels R, G et B ont des sensibilités différentes. Le microcontrôleur présent sur la caméra modifie alors les gains associés à ces 3 composantes (le capteur permet de modifier le gain par composante couleur) en fonction des moyennes calculées, pour que les moyennes pondérées deviennent identiques. Cette opération s'appelle "balance des blancs", et est faite une fois de manière automatique après le réglage de l'iris, au début du match.

Une troisième opération de calibration consiste en le calcul des seuils pour la quantification. Cette opération n'a pas été implémentée, en partant de l'hypothèse que les couleurs des palets, après réglage de l'iris et balance des blancs, seraient similaires quelles que soient les conditions d'éclairage.

### 5.5.5 Conclusion

La calibration automatique a donné de bons résultats, mais la caméra n'a pas pu être testée en conditions réelles faute de temps. Il restait encore à valider l'hypothèse ci-dessus, à déterminer les seuils de quantification et à tester quelques modifications dans la table de lookup de l'algorithme de quantification.

A côté des problèmes purement visioniques, une certaine inertie a été constatée. Lorsque le robot tournait sur lui-même pour trouver un palet, la caméra ne lui donnait les informations qu'avec un certain retard (traitement à 8.5 images par seconde), ce qui équivalait à un angle de rotation de 16°. Ce problème a pu être contourné assez simplement, en tournant plus lentement lorsqu'un palet avait été détecté pour se caler dessus avec précision.

En conclusion, si la caméra n'a pas été utilisée, c'est principalement à cause d'un manque de temps.

# 6. Capteurs

En automatisme, on distingue dans un robot la *partie commande* qui traite toutes les informations, les *actionneurs* qui effectuent une action et les *capteurs* qui informent le robot. Les capteurs ont une place prépondérante dans le système de traitement d'un robot. Ils peuvent à la fois informer le robot sur le milieu extérieur et le renseigner sur ses propres actions en vérifiant l'état de ses actionneurs. Ils sont donc l'élément indispensable à un robot autonome pour savoir ce qu'il fait, ce qui se passe et prendre les bonnes décisions en conséquence. Toutefois, les capteurs n'étant pas fiables à 100%, augmenter leur nombre signifie diminuer la fiabilité globale du robot. Un nombre optimal de ces capteurs doit donc être trouvé pour avoir une bonne fiabilité tout en ayant un maximum d'informations.

## 6.1 Microswitchs

Les microswitchs sont des boutons poussoir. Ils sont très simples d'utilisation, ils donnent l'information 1 lorsqu'ils sont déclenchés et 0 en position de repos. Cette année, nous en avons utilisé à trois endroits sur le robot :

- A l'avant du robot, pour détecter les collisions contre les bords ou contre un adversaire. Deux microswitchs sont suffisants, un à l'avant gauche, l'autre à l'avant droit.
- Dans l'ascenseur, pour savoir à quel endroit il se situe. Trois microswitchs sont nécessaire car l'ascenseur se déplace sur trois positions. La position basse pour prendre un palet ou pour déposer une pile. La position intermédiaire pour surélever la pile tout en empêchant un palet non traité de rentrer dans la zone d'empilement. La position haute pour laisser passer le palet traité dans la zone d'empilement.
- Enfin un microswitch signale lorsque l'ascenseur est plein auquel cas le robot ne doit plus essayer d'empiler mais doit laisser passer les autres palets.

En ce qui concerne les microswitchs avant, il s'est avéré l'année passée que les mettre directement au contact de chocs diminuait fortement leur durée de vie. Un astucieux système mécanique a alors été mis en œuvre pour éviter de détériorer les microswitchs.

Le système proposé actionne en permanence ce microswitch (figure 2) et ce n'est que lors du contact qu'il est relâché (figure 3). Un système de ressort amorti fortement le choc, et après celui-ci replace la tige à sa position de départ. 2 ressorts sont mis en série : le premier (de faible constante de raideur) permet au switch de très vite s'actionner. Le second de forte constante de raideur permet une bonne absorption des chocs.
La tige qui s'enfonce (lors du choc) est guidée dans un bloc en laiton.

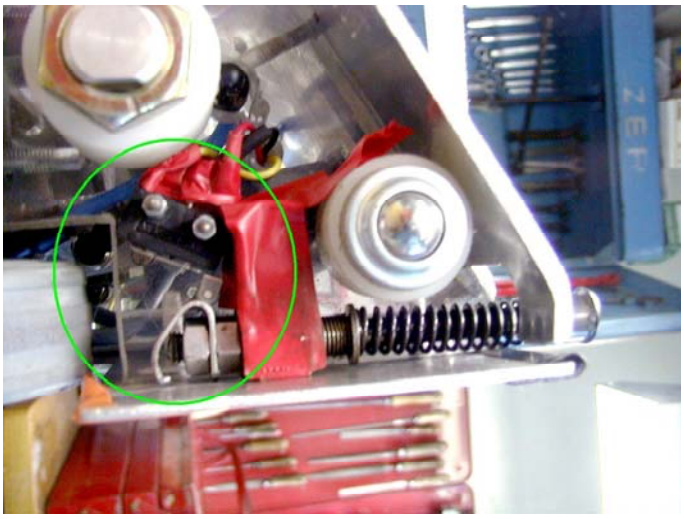


**Figure 36 : microswitch au repos**

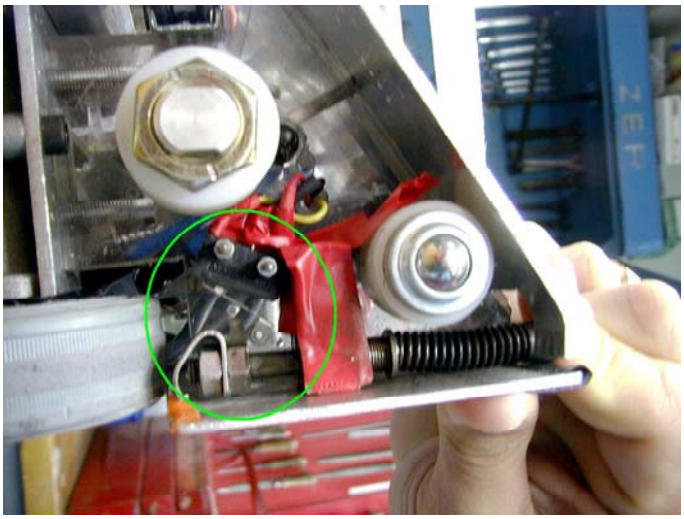


**Figure 37 : microswitch enclenché**

## 6.2 Télémètres

En robotique, on cherche toujours à obtenir le maximum d'informations sur l'environnement afin de pouvoir adapter le comportement des robots. Une information particulièrement intéressante à utiliser est celle concernant la distance. (distance du robot par rapport à un mur, à un palet, au sol, etc.)
Les télémètres peuvent nous renseigner sur la distance réelle en cm ou simplement nous donner l'information suivante : "le mur est à moins de X cm". Le système le plus communément utilisé en robotique actuellement était le système de télémètre à ultrasons. Mais on trouve des systèmes totalement infrarouges, des capteurs SHARP (comme le GP2D12) qui présentent de nombreux avantages comme notamment le fait d'être moins sensible aux parasites et brouillages. Or pendant les matchs de la coupe, il s'est avéré que les systèmes à ultrasons étaient trop souvent perturbés. Cela nous a conduit à opter pour le choix des télémètres infrarouges SHARP.

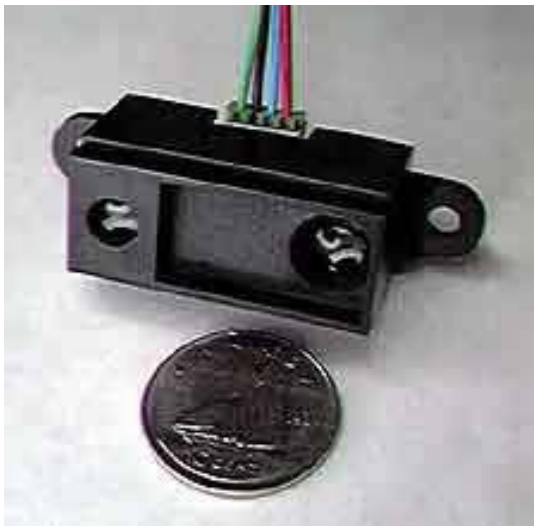


**Figure 38 : télémètre infrarouge SHARP**

Comme nous le voyons sur la figure précédente, ces télémètres ont l'avantage d'avoir des dimensions fort réduites, la contrainte de place n'est donc plus un problème. Nous en avons placé un à l'intérieur du robot afin de détecter la présence d'un palet lorsqu'un de ceux-ci pénètre dans le robot et vient se placer à l'endroit précis du retournement éventuel.
Il faut cependant s'assurer qu'aucun objet ne coupera le faisceau infrarouge en dessous de 10 cm, limite minimum de bon fonctionnement du télémètre.

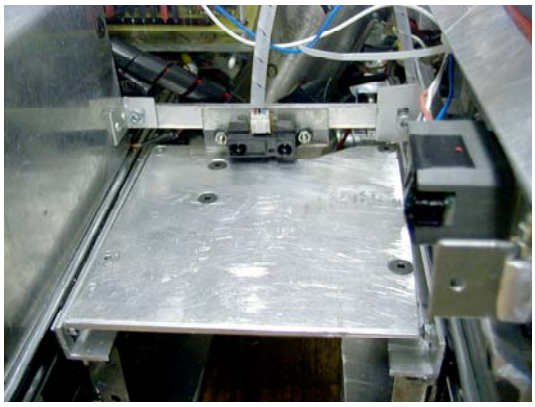


**Figure 39 : Le télémètre détecte la présence du palet à l'intérieur du robot**

Remarque :

Un autre télémètre aurait pu être placé à l'avant du robot pour détecter la présence éventuelle d'un robot adverse et ainsi éviter la collision. Afin d'éviter de devoir ralentir chaque fois que le robot est à proximité d'un palet, il suffit de le placer à une hauteur telle qu'il ne verra pas les palets devant lui mais uniquement le robot adverse. Nous n'avons pas eu le temps de développer ce système cette année mais les robots devenant de plus en plus rapide et lourd, un système d'évitement des autres robots deviendra obligatoire dans les années à venir. Le système proposé ici est une des solutions les plus simples.

### 6.2.1 Fonctionnement

Le capteur SHARP fonctionne en mesurant l'angle de réflexion d'une émission modulée[4] d'IR grâce à une rangée de récepteurs. La portée officielle est de 10 à 80cm. Si on détecte le signal émis, cela veut dire qu'un objet se trouvant devant le capteur a renvoyé le signal.

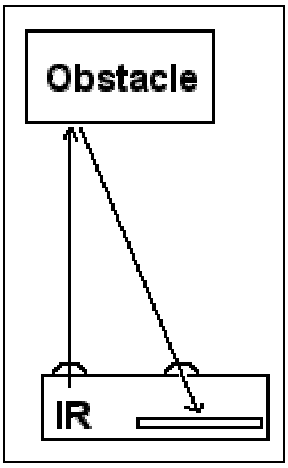



**Figure 40 : principe de fonctionnement**

La précision du capteur dépend de la distance (simple conséquence trigonométrique). Excellente à 10 cm, elle régresse de plus en plus jusqu'à 80cm. Mais la précision reste bonne et est d'environs 1 cm à 80cm. La directivité est bien meilleure que pour les ultrasons, le cône d'émission est de 5°.
Le temps de réaction est très rapide et est de quelques ms seulement.

### 6.2.2 Carte télémètre :

Les télémètres sont alimentés en 9v. Cette tension est ensuite abaissée à 5v par un régulateur de tension intégré sur le circuit électronique. (Possibilité de repiquer 5V sur la carte télémètre).
La carte des télémètres est composée de 4 circuits identiques permettant au télémètre de différencier plusieurs seuils.
Chacun de ces circuits est composé :

- D'un amplificateur opérationnel dont le rôle sera de comparer la tension donnée par le télémètre à une tension de référence (rhéostats variables R7, R15 R8 et R16) ;
- De résistances variables (R1, R2 ; R4, R5 ; R9, R10 ; R12, R13 ) dont le rôle est de former une bascule de Schmidt en vue d'éviter l'instabilité du signal de sortie lorsque la tension de sortie du télémètre approche de la tension de référence ;
- led témoin en série avec une résistance qui annonce la détection ou non.

[4] La modulation permet de s'affranchir de l'éclairage ambiant. Avec la modulation, le capteur travaille essentiellement sur les flancs montants et descendants du signal infrarouge. Or une émission de lumière continue ne modifie que l'amplitude du signal mais pas sa modulation. L'inconvénient est qu'alors il devient sensible aux émissions modulées qui lui parviendraient, comme par exemple une émission de télécommande TV.

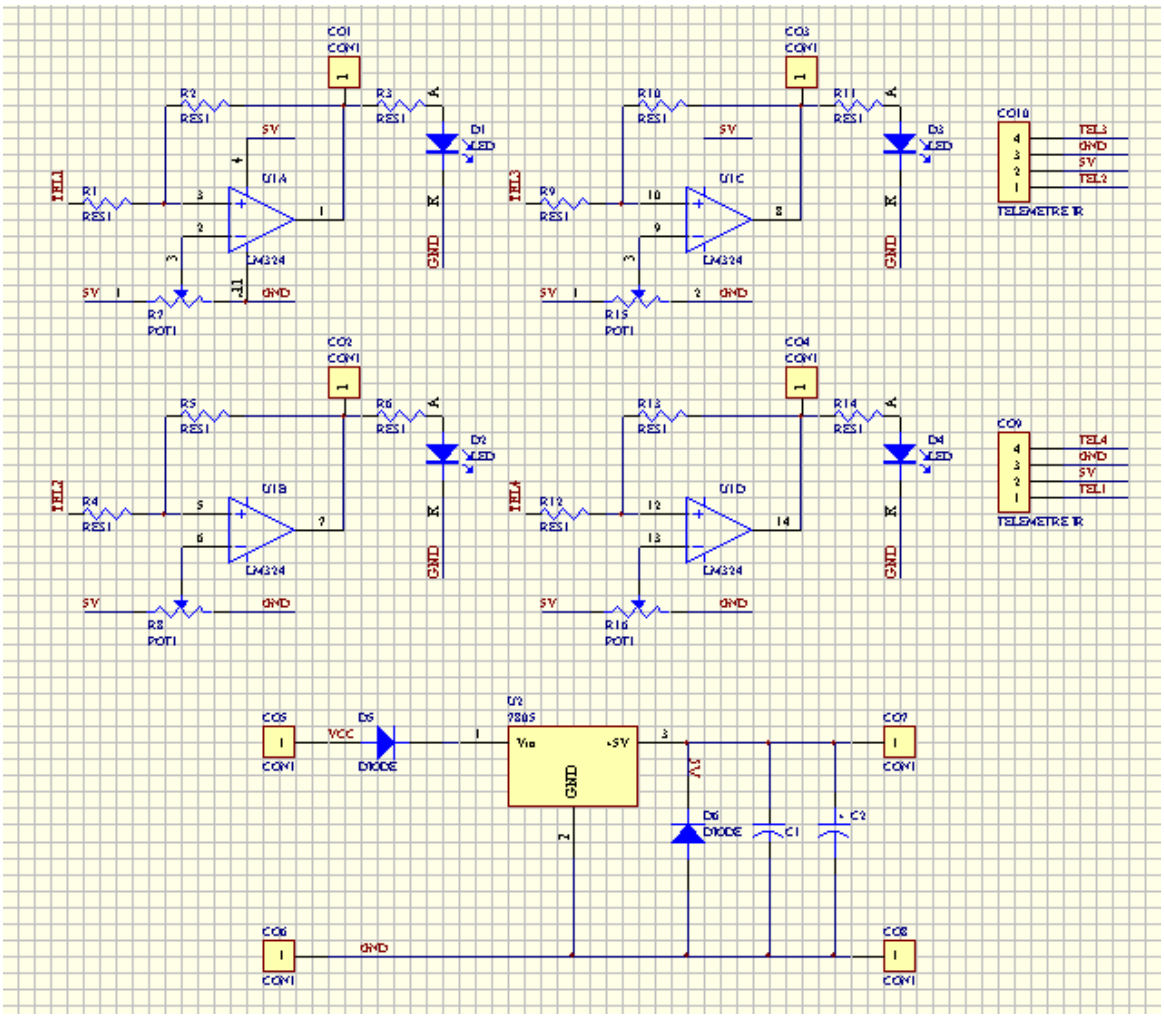



**Figure 41 : Schéma de principe carte télémètre**

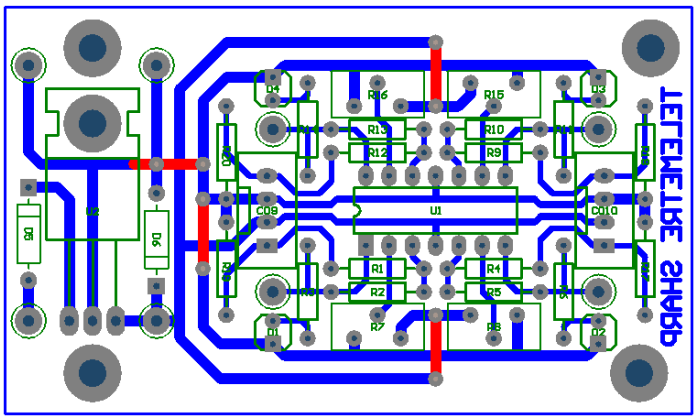



**Figure 42 : carte télémètre**

## 6.3 Capteurs de couleurs

### 6.3.1 Fonctionnement

Les capteurs de couleurs sont généralement réalisés à partir d'un couple émetteur / récepteur infrarouge. La diode émet une lumière infrarouge en direction de l'objet et le phototransistor reçoit les photons par réflexion sur celui-ci. On utilise les infrarouges, car les capteurs sont plus petits. Mais aussi pour s'affranchir des problèmes dus à la lumière ambiante. Si on utilise un capteur dont la longueur d'onde se situe dans le visible, l'éclairage ambiant pourrait venir perturber le capteur. La lumière (naturelle ou artificielle) possède aussi une composante infrarouge qui pourra venir dégrader la fiabilité du capteur, nous en parlerons plus loin. Pour avoir une bonne précision, la distance entre le capteur et l'objet doit être toujours la même et doit se situer entre 1 et 4 mm au maximum.
Voici le schéma de principe de la détection infrarouge :

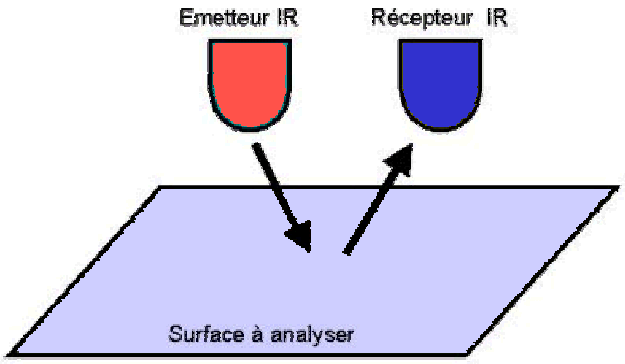



**Figure 43 : émetteur-récepteur infrarouge**

L'émetteur et le récepteur doivent être face à la surface à analyser. Si la surface est claire, les infrarouges seront réfléchis. Si la surface est mate et/ou foncée, les infrarouges seront absorbés. Dans le premier cas le récepteur détectera la présence d'infrarouge, et dans le second, le récepteur sera bloqué.

### 6.3.2 CNY70

**Figure 44 : capteur couleur CNY70**

Le CNY70 est un composant spécialement créé pour ce genre de capteurs. Il comporte un émetteur et un récepteur infrarouge. L'émetteur et le récepteur sont accordés sur la même longueur d'onde. L'émetteur est une diode infrarouge centrée sur la longueur d'onde 950 nm. Le récepteur est un phototransistor. Voici l'implantation d'un CNY70 :

L'émetteur est constamment alimenté. Selon la luminosité infrarouge reçue sur la base du transistor, la tension sur la borne du CNY70 va varier entre 0 et 5 Volts. Ce qui nous intéresse ici est de discerner deux états : réfléchi ou non réfléchi. Nous allons utiliser un comparateur de tensions pour comparer la tension reçue du phototransistor avec une tension de seuil réglable grâce à un potentiomètre.
Pour régler le capteur, il faut procéder comme suit :

- Placer le palet de couleur claire (rouge) en dessous du capteur et relever la tension moyenne Vc sur la borne du CNY70. Faire tourner le palet pour relever la tension à différents endroits car une petite différence dans la couleur peut faire varier assez fortement la tension.

- Faire de même pour la couleur foncée (verte) et relever la tension Vf.

- Réglez le seuil de réaction à l'aide du potentiomètre afin d'obtenir une tension de (Vc+Vf) / 2 sur la borne de ce dernier.

Le schéma ci-dessus renvoie un 1 au microcontrôleur pour une surface rouge et un 0 pour une surface verte.

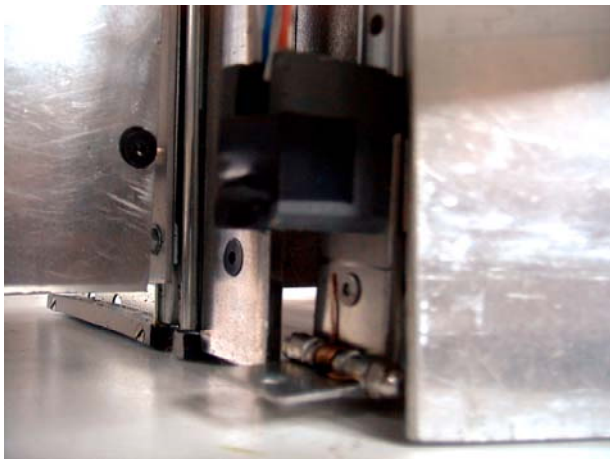


**Figure 45 : capteur couleur à l'entrée du robot**

### 6.3.3 Problèmes rencontrés

Ce capteur fonctionne très bien lorsque les conditions ambiantes restent invariables. La lumière artificielle ou naturelle comporte toujours une composante infrarouge. Il arrive que cette composante vienne perturber le bon fonctionnement du capteur. Nous avons rencontré cet inconvénient lors de la coupe de Belgique lorsque le robot c'est déplacé sur la table officielle pendant les premiers matchs car les seuils n'avaient pas été bien réglés (pour ces conditions). La forte luminosité a perturbé le capteur ce qui nous a valu de retourner tous les palets rencontrés.
Pour éviter ce genre de problème, il faudrait moduler (ou coder) les signaux afin de les différentier de la composante continue de la lumière.
Un autre problème est survenu à la suite d'un choc qui a déplacé le capteur de sa position initiale. Les seuils étant réglés à une position déterminée, ce changement de distance capteur palet (dans notre cas la distance a augmenté) faisait qu'on ne détectait plus la couleur rouge (l'intensité réfléchie de l'infrarouge n'était plus suffisante pour que l'état passe à 1)

La seule solution est de mieux fixer le capteur pour qu'il ne tourne plus sur son axe de fixation lors de chocs.

### 6.3.4 Carte des capteurs de couleur

Les capteurs de couleur Cny70 sont composés d'un émetteur infrarouge et d'un phototransistor qui capte la réflexion du signal IR. La carte est composée des éléments suivants :

- résistance R1 dont le but est d'abaisser la tension à une valeur acceptable pour l'émetteur ;
- résistance R2 dont le but est de polariser le transistor ;
- un amplificateur opérationnel dont le rôle est de comparer la tension à la sortie du phototransistor à une tension de référence (la tension de référence est elle-même réglable par l'intermédiaire de R3) ;
- diode témoin en série avec une résistance.

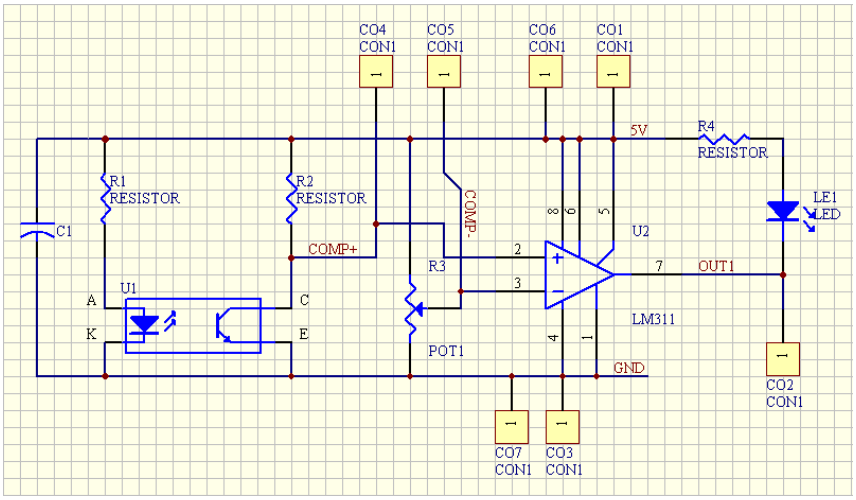



**Figure 46 : schéma de la carte des capteurs de couleur**

## 6.4 Capteur laser

Le récepteur employé est un récepteur infrarouge mais il réagit également lorsqu'on l'éclaire avec un laser[5]. Le rôle de ce capteur est de détecter le palet lorsqu'il est entré dans la zone d'empilement sans venir au contact avec celui-ci. Le palet qui rentre dans cette zone coupe le faisceau laser et la photodiode ne reçoit plus de lumière.

[5] Un laser est normalement censé émettre une lumière monochromatique dans le visible autour de 640 nm de longueur d'onde.

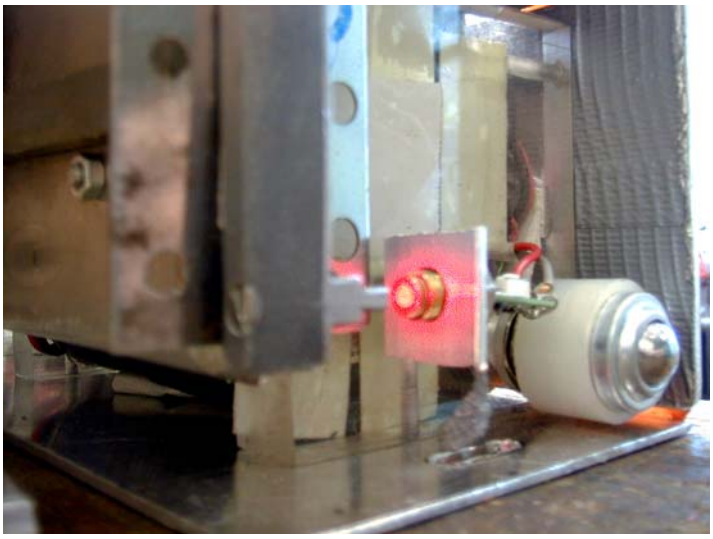


**Figure 47 : Laser**

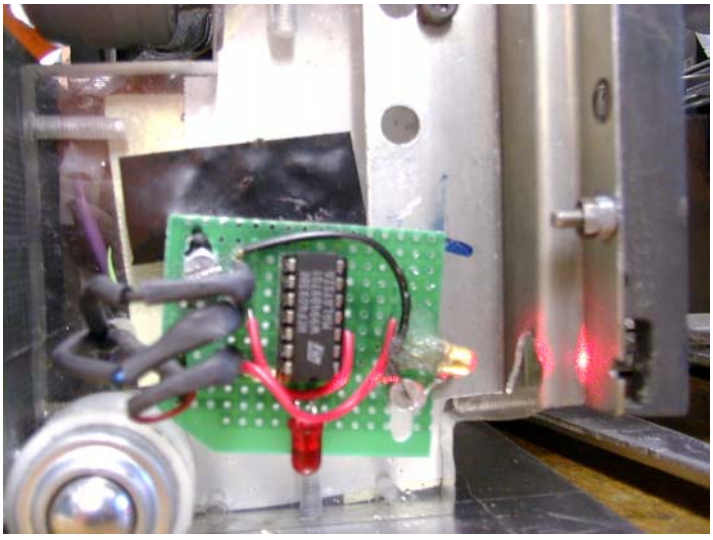


**Figure 48 : Photodiode infrarouge (récepteur)**

# 7. Motorisation

## 7.1 Moteurs de propulsion

Très tôt s'est posé un certain nombre de problèmes concernant la motorisation :
- Coût des moteurs Crouzet brushless ;
- Encombrement important et donc impossibilité de les mettre dans l'axe l'un de l'autre.

Afin de résoudre ces problèmes, différentes solutions ont été envisagées :
- Trouver des moteurs brushless plus petits ;
- Utiliser des moteurs à courant continus classiques ;
- Placer les moteurs verticalement et attaquer les roues par un renvoi d'angle.

### 7.1.1 Moteurs à courant continu classiques

Différentes offres de moteurs à courant continu avec brush ont été considérées. Malgré un prix plus faible, ces moteurs présentent comme inconvénient que la commutation entre balais et collecteur peut créer des interférences et arcs électriques qui pourraient venir parasiter notre électronique.
Remarquons cependant que bon nombre des équipes qui ont participé à la coupe de robotique cette année semble avoir opté pour cette option.

#### 7.1.1.1 Offre de Crouzet

Le moteur Crouzet 80803 005 pourrait convenir à notre application ses caractéristiques sont les suivantes.

| | | |
|---|---|---|
| Tension d'alimentation | Volt | 24 |
| Puissance utile | W | 16 |
| Vitesse | T/min | 1à 161 |
| Alimentation par variation de la tension entre 50 et 150 % de la tension nominale | | |
| Prix | € | 106 |

Ce moteur présente l'avantage de présenter un encombrement et un prix réduit par rapport au moteur Crouzet Brushless.

Inconvénients des moteurs à courant continu : puissance moins grande, pas de régulation de vitesse, parasitage, gain de place peu important.
Afin de palier le risque que représentent les parasites pour notre électronique, nous avons porté notre choix sur des moteurs brushless.

### 7.1.2 Moteurs Brushless

Les moteurs brushless représentent une avancée considérable au niveau des performances, de la fiabilité, de la durée de vie par rapport aux moteurs à courant continus classiques. Leur seul inconvénient est à l'heure actuelle leur prix, sensiblement plus élevé que le prix d'un moteur classique à balais.

Remarque générale sur les moteurs Brushless :
Lors du choix d'un moteur de ce type, il faut garder à l'esprit que l'électronique doit être intégrée directement sur le moteur. En effet, la plupart des constructeurs proposent des moteurs brushless sans l'électronique ce qui implique l'obligation de placer des cartes supplémentaires à l'extérieur des robots. Ces cartes trop encombrantes et chères ne sont pas implantables dans le périmètre restreint du robot.
Les différentes offres que nous avons considérées sont détailles ci-après :

#### 7.1.2.1 Moteurs MPD

**Contact :Mdp motorisation**
21, porte du grand Lyon –Neyron-01707 Miribel Cedex/ France
Tel :0033/472019019

Moteur BG40X25E5

| | | |
|---|---|---|
| Tension d'alimentation | V | 24 |
| Vitesse nominale | Tr/min | 2885 |
| Couple au courant nominal | MN/m | 57 |
| Puissance utile à 24 V | W | 18.9 |
| Courant de démarrage à 24 V | MA | 3900 |
| Courant à vide | MA | 400 |
| Electronique intégrée | | |
| Pas d'alimentation Pwm<br>Alimentation par variation de tension de commande entre 7 et 28 V qui implique la présence de convertisseurs | | |

Ce moteur peut-être couplé à un réducteur planétaire. Afin d'atteindre une vitesse de sortie semblable à celle que l'on avait avec les moteurs du concours 2001-2002, une réduction à deux étages sera nécessaire rendant l'encombrement de ce moteur identique à celle des Crouzet.
Prix unitaire : 350 €

#### 7.1.2.2 Moteurs Mc Leannan

**Contact : Syservo Sprl.**
Spoorwegbaan 47,
1730 Asse, Belgium.
Telephone: **32 2 452 41 45**
Fax No: **32 2 453 23 31**

BLDC48

Le moteur BLDC48 de Mc Leannan est un moteur DC brushless avec électronique intégrée développant une puissance utile de 12W.
Deux types de commande de ce moteur sont possibles :
- commande de la vitesse du moteur à partir de la tension d'alimentation ( 0-12V) ;
- commande de la vitesse du moteur à partir de l'électronique (0-5 V).

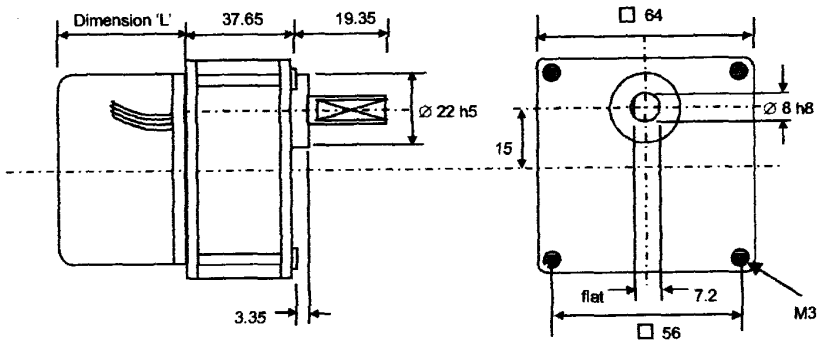
Les caractéristiques de ce moteur sont les suivantes :

**Specification**

| Model | | BLDC48-8L | | BLDC48-12L | |
|---|---|---|---|---|---|
| Direction of rotation | | cw | ccw | cw | ccw |
| 12 Volt 2 wire version order code | BLDC48- | 8L-005 | 8L-001 | 12L-025 | 12L-021 |
| 24 Volt 2 wire version order code | BLDC48- | 8L-015 | 8L-011 | 12L-035 | 12L-031 |
| 12 Volt 4 wire version order code | BLDC48- | 8L-007 | 8L-003 | 12L-027 | 12L-023 |
| 24 Volt 4 wire version order code | BLDC48- | 8L-017 | 8L-013 | 12L-037 | 12L-033 |
| Continuous output power | Watts | 8 | | 12 | |
| Maximum speed | rpm | 4300 | | 4550 | |
| Maximum speed @ rated torque | rpm | ≥ 2900 | | 3200 | |
| Rated Torque | Nm | 0.022 | | 0.03 | |
| Stall Torque | Nm | ≥ 0.032 | | 0.052 | |
| Rotor inertia | $Kgcm^2$ | 0.22 | | 0.3 | |
| Motor Supply voltage | Vdc | 12 | 24 | 12 | 24 |
| Motor supply current @ rated torque | Amps | 1.01 | 0.51 | 1.33 | 0.69 |
| Peak current @ stall ( internally limited ) | Amps | 1.4 | 0.7 | 1.95 | 0.95 |
| Analogue speed control signal | V/1000 rpm | 0.83 : only available on 4 wire versions | | | |
| Digital output speed monitor | ppr | 6: only available on 4 wire versions | | | |
| Internal Over-temperature protection | | standard | | | |
| Bearing type | | Ball | | | |
| Maximum radial load | N | 40 @ 15 mm from mounting face | | | |

**Figure 49 : caractéristiques du moteur BLDC 48**

Dans le souci d'avoir une vitesse de sortie adaptée à notre application, un réducteur doit être monté en bout d'arbre. Les caractéristiques de ce réducteur sont fournies à la figure suivante :




**Performance using BLDC48**

| Gearhead | Ratio n:1 | Using BLDC48-8L | | | Using BLDC48-12L | | |
|---|---|---|---|---|---|---|---|
| | | Rated Speed rpm | Rated Torque Nm | Peak Torque Nm | Rated Speed rpm | Rated Torque Nm | Peak Torque Nm |
| S64A0005AA | 5 | 600 | 0.09 | 0.13 | 640 | 0.12 | 0.2 |
| S64A0012AA | 12.5 | 230 | 0.2 | 0.3 | 250 | 0.26 | 0.45 |
| S64A0025AA | 25 | 120 | 0.36 | 0.5 | 125 | 0.5 | 0.85 |
| S64A0050AA | 50 | 60 | 0.72 | 1.0 | 60 | 1.0 | 1.7 |
| S64A0100AA | 100 | 30 | 1.3 | 1.9 | 30 | 1.75 | 3.0 |
| S64A0125AA | 125 | 24 | 1.6 | 2.3 | 25 | 2.2 | 3.8 |
| S64A0250AA | 250 | 12 | 3.0 | 4.5 | Use 8 watt motor | | |

**Figure 50 : caractéristiques du réducteur du moteur BLDC48**

Ce moteur est de dimension nettement inférieure à celle des moteurs Crouzet. Ceux-ci nous auraient peut-être permis de placer les moteurs dans l'axe l'un de l'autre. Cependant la perte de puissance (un de nos atouts en match) nous semblait trop importante pour pouvoir implanter ces moteurs.
Aucune offre de prix n'a été faite par le distributeur belge pour ce moteur.

Dans les années futures il pourrait être intéressant d'utiliser ce type de moteur (pour des applications autres que la motorisation) pour s'affranchir totalement du problème de parasites provoqués par les moteurs à courant continu classiques (ex : moteur ascenseur).

Remarque : McLeannan propose dans la même gamme, des moteurs de 38 W de même encombrement et de même prix que les moteurs Crouzet. Nous leur avons donc préféré les moteurs Crouzet.

#### 7.1.2.3 Moteurs Crouzet

Les moteurs Crouzet de type 80 035 509 sont les moteurs déjà utilisés lors de l'édition 2001-2002 du concours. Ce choix a été réédité cette année et s'est avéré particulièrement judicieux malgré les ennuis que nous avons eus avec la motorisation.

Leurs caractéristiques sont les suivantes :

| | | |
|---|---|---|
| Tension d'alimentation | Volt | 24 |
| Vitesse de rotation | Tr/min | 5400 |
| Puissance utile | W | 30 |
| Puissance absorbée | W | 45 |
| Courant absorbé | A | 0.22 |
| Courant de démarrage | A | 2.2 |
| Electronique intégrée<br>Régulation de vitesse<br>Commande par Pwm ou par régulation de tension | | |
| Réduction de 12.5 | | |

Ces moteurs sont directement commandés à partir de l'Atmega en utilisant une commande PWM (Pulse Width Modulation). Cette commande se base sur le principe suivant :

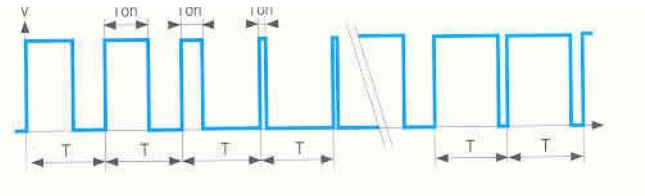



**Figure 51 : Pwm**

T est constant mais ton varie ; soit le rapport Ton/T le rapport cyclique ;
-Si le rapport cyclique est de 0%, la vitesse est de 0tr/min ;
-Si le rapport cyclique est de 100%, la vitesse est égale à la vitesse à vide ;
-Si le rapport cyclique est de 50%, la vitesse est égale à la (vitesse à vide)/2.

L'utilisation des moteurs Crouzet présente divers avantages :
-Couple important ;
-Commande informatique déjà réalisée ;
-Encombrement semblable à celui des autres moteurs de même classe ;
-Puissance légèrement supérieure ;
-Expérience acquise lors du concours précédent.

Malgré ces avantages, nous avons rencontré certains problèmes, avec les moteurs de propulsion. En effet, par deux fois l'électronique de ceux-ci a brûlé. La cause de la première défaillance est une erreur de manipulation alors que la deuxième défaillance requiert une attention plus particulière.

En effet, lors d'essai à vide de la Cmucam, le moteur a été amené à changer un certain nombre de fois son sens de rotation. Ce changement brutal de sens a provoqué un fonctionnement en génératrice du moteur provoquant le bris de l'électronique. Cette hypothèse a été confirmée par Crouzet, nous affirmant que bon nombre de leurs moteurs leur étaient revenus en service après vente pour des raisons semblables. Pour pallier ces défauts une nouvelle électronique a été conçue. Actuellement 4 des 5 moteurs présentent cette nouvelle électronique.
En outre, une boucle supplémentaire dans la programmation a été prévue afin d'être certain du passage à l'arrêt du moteur à chaque changement de sens de rotation.
Commande
Les moteurs seront commandés par un signal PWM.
Chaque moteur nécessite l'utilisation de 5 sorties. Le code couleur est le suivant :
-Fil orange pour la sortie du signal (PB5/PB6) ;
-Fil rouge pour l'alimentation 24 V ;
-Fil noir pour la masse de l'alimentation ;
-Fil bleu sur la masse de l'Atmega (GND) ;
-Fil vert sur une sortie pour l'information Marche/Arrêt (PB0/PB2) ;
-Fil jaune sur une sortie pour l'information Avant /Arrière (PB1/PB3).

### 7.1.3 Etalonnage des moteurs

Dans le but de comparer les vitesses de sortie des moteurs, un étalonnage a été réalisé.
Afin de réaliser cette opération, nous utilisons le codeur interne à l'électronique (fil blanc) qui envoie 12 impulsions par tour du moteur.
Remarque : les indications données par le codeur donnent une indication de la vitesse de rotation avant le réducteur.
Les résultats obtenus sur les deux moteurs du robot 2001-2002 sont présentés à la figure suivante :

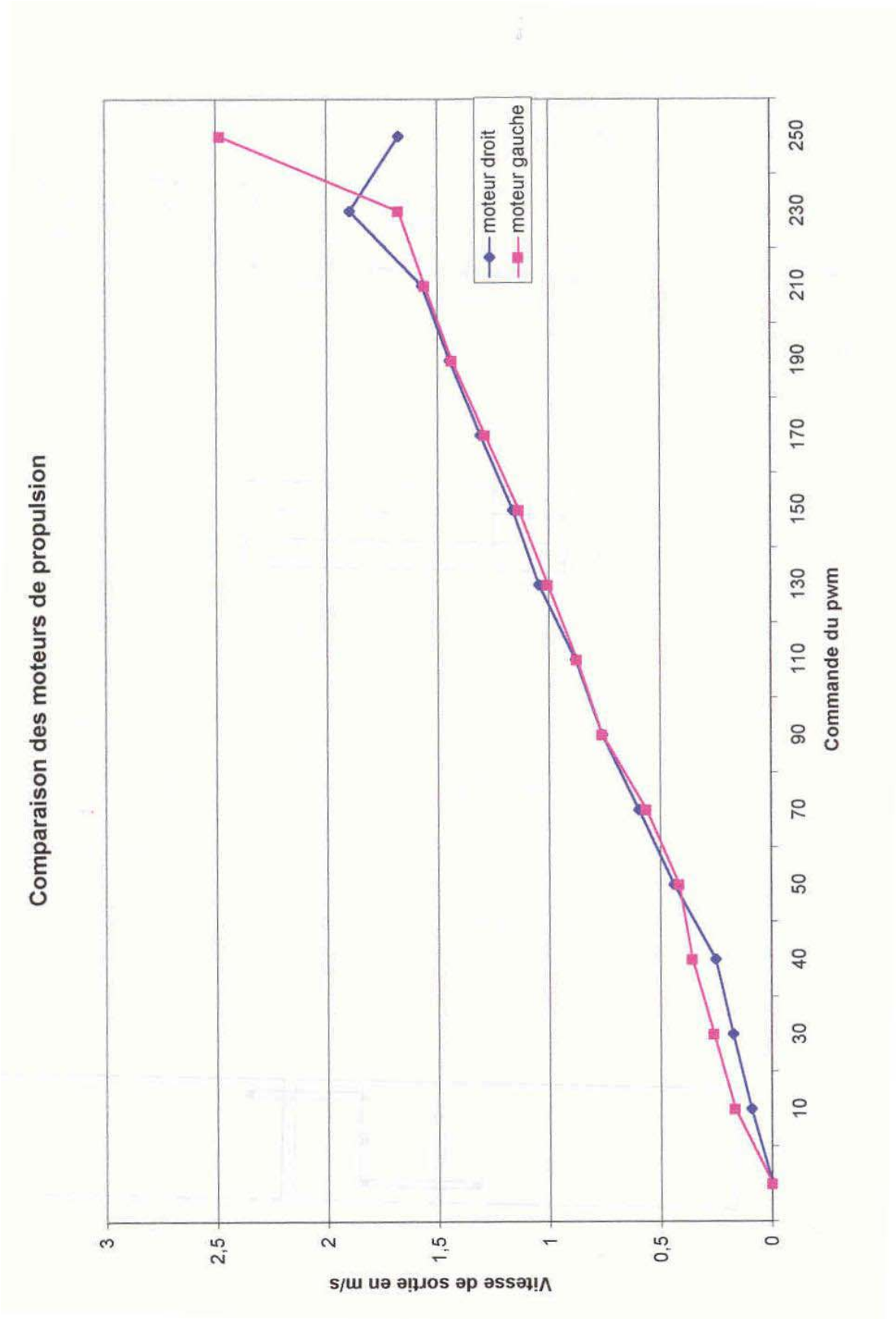



**Figure 52 : Etalonnage des moteurs Crouzet**

Il apparaît après examen de ces courbes que les deux moteurs donnent une vitesse semblable qui évolue linéairement en fonction de la commande. Cette linéarité se manifeste dans toutes les zones exceptées les zones extrêmes de commande. Ceci est confirmé par la data-sheet des moteurs.

Remarque :
Cette année, un comportement anormal du robot s'est manifesté. En effet, bien que la consigne soit la même pour les deux moteurs, la vitesse de rotation des roues n'était pas la même des deux cotés du robot ; ceci ayant comme résultat de provoquer une légère rotation du robot. Il serait peut-être utile de refaire un étalonnage afin de vérifier si l'origine de ce phénomène est l'un des moteurs ou un renvoi-d'angle.

## 7.2 Moteur de l'ascenseur

En raison du manque de temps, une seule offre a été considérée pour le moteur de l'ascenseur à palets.
Le calcul suivant a motivé notre choix :

| | |
|---|---|
| Poids estimé d'un palet | 0.12 Kg |
| Nombre de palet | 3 |
| Poids estimé de l'ascenseur | 0.4910 kg |
| Poids total estimé | 0.85 kg |
| Bras de levier = l | 0.02 m |
| Vitesse estimée de rotation | 6.2832 rad/sec |

L'accélération maximale que nous voulons donner au dispositif peut se déduire de la formule suivante :

$a_{max}=\frac{V^2}{R}$ =0.78 (m/s^2)

Le couple que va devoir fournir le moteur se déduit de :
C = $m \cdot g \cdot l + m \cdot a_{max} \cdot l$ = 0.18041 Nm
Et donc la puissance motrice nécessaire est égale à :
P = $C \cdot \omega$ *facteur de sécurité =2.26 W
Le moto réducteur choisi est de la marque Crouzet et de type 82 862 003. Il s'agit d'un moteur à courant continu ayant les caractéristiques suivantes :

| | | |
|---|---|---|
| Tension d'alimentation | Volt | 24 |
| Puissance utile | W | 3 |
| Vitesse de rotation | T/min | 45 |
| Réduction d'un facteur 95.4<br>Prix | <br>€ | <br>62 |

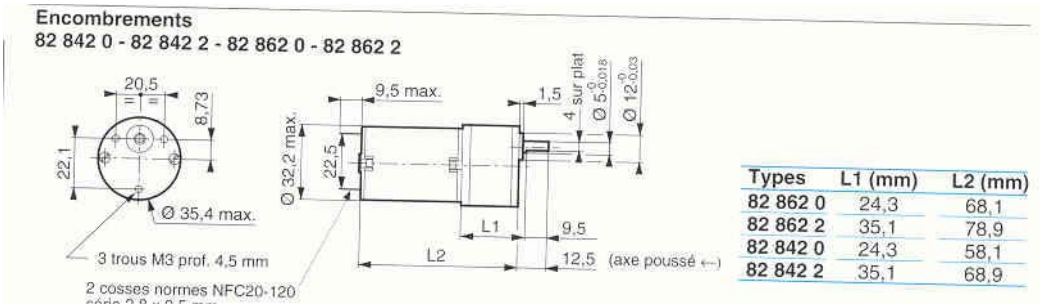



**Figure 53 : schéma du moteur de l'ascenseur**

# 8. Electronique

## 8.1 ATMEGA

Le microcontrôleur utilisé est un Atmega 103. C'est le même microcontrôleur que celui employé l'année passée. Pour éviter que ce rapport ne prenne une importance trop grande, nous limiterons les explications sur l'Atmega proprement dit. Rappelons qu'une documentation détaillée est fournie par le constructeur. Le micro contrôleur, travaille par niveau logique (0 ou 5Volts). Les ports de la carte support (A à F) travaillent soit purement en entrée, soit purement en sortie ou encore en entrée-sortie. L'horloge interne de l'Atmega travaille à un peu moins de 4Mhz (8Mhz pour l'atmega 128). Cela définit le nombre d'interruptions qu'effectue le microcontrôleur en une seconde. L'Atmega 128 réalise environs 7228 interruptions par seconde, c'est-à-dire qu'il peut parcourir les programme 7228 fois par seconde.

## 8.2 Electronique de puissance

La carte de puissance sert d'intermédiaire entre les alimentations (par les batteries) et les différents éléments électriques qui constituent le robot. La carte a été construite sur base de celle de l'année 2001-2002, avec cependant un certain nombre de modifications.

### 8.2.1 Batteries

Cette année, nous avons préféré aux batteries au plomb utilisées durant le concours 2001-2002, quatre batteries Ni-Mh.
Leurs caractéristiques sont les suivantes :

| | |
|---|---|
| Voltage | 12V |
| Ampérage | 3000 mAh |

Ce choix a été motivé pour différentes raisons :

- la possibilité de leur donner une forme quelconque. En effet, les batteries Ni-Mh se présentent comme un assemblage de petits éléments (qui ressemblent à des piles) que l'on peut disposer suivant différentes combinaisons. Si nous avions dû utiliser des batteries au plomb, nous nous serions heurté à de sérieux problèmes de place ;
- leur capacité de 3000 mAh, nous a donné une très grande autonomie.

### 8.2.2 Circuit du haut

Le circuit du haut est quasiment identique à celui de l'année 2001-2002. Il sert principalement à alimenter les moteurs de propulsion en 24V (deux batteries 12V Ni-Mh en série)
Ce circuit est composé :

- D'un interrupteur général qui permet de couper l'alimentation 24V ;
- D'un fusible de 5 A (capable de supporter le courant de démarrage des moteurs (2 * 2.2 A)) ;

- De diodes de contrôle avant et après fusible respectivement pour vérifier l'état de la tension et l'état du fusible ;
- D'un bouton poussoir arrêt (rouge) normalement fermé. L'ouverture de ce circuit provoque l'arrêt des moteurs du robot et des éléments commandés par le relais Km1 ;
- D'un relais Km1 (4 inverseurs) qui permet de fermer 4 contacts pour le passage d'un courant plus puissant par le passage d'un courant plus petit ;

    - Un premier contact de ce relais est mis en parallèle avec le bouton poussoir « Marche ». Cette opération est nécessaire car les boutons poussoirs n'ont pas de mémoire. Pour maintenir le contact, il faudrait maintenir le bouton appuyé pendant tout le match (ce qui bien sur est interdit par le règlement). Une fois le relais activé le circuit se fermera définitivement et ne pourra être coupé que par action sur le bouton arrêt.

    - Un deuxième contact du relais Km1 est utilisé pour fermer un autre relais (Km2). Ceci est nécessaire car le relais Km1 ne supporte pas de grands courants. En effet, on voit que le relais Km2 sert à fermer le circuit des moteurs. Le circuit Km2 verra donc passer l'entièreté du courant qui passe dans les moteurs.

    - Un troisième contact est utilisé pour la commande de la vanne ;

    - Le dernier contact est laissé libre ;

- D'un bouton poussoir marche (rouge) normalement ouvert et mis en parallèle avec un contact du relais. La fermeture de ce circuit provoque la mise sous tension des circuits 24V, 9V et 5V ;
- Deux fusibles de 2 A pour protéger les moteurs des surintensités.
- De Leds de contrôle (clignotantes) qui sont mises en parallèle sur les fusibles qui protègent les moteurs

Remarquons que malgré que le courant de démarrage des moteurs soit de 2.2 A, les fusibles de 2 A ne se détériorent pas en raison du temps nécessaire pour qu'une surintensité les fasse fondre. Le courant de démarrage n'étant que temporaire, nous n'avons pas rencontré de problèmes à ce niveau.

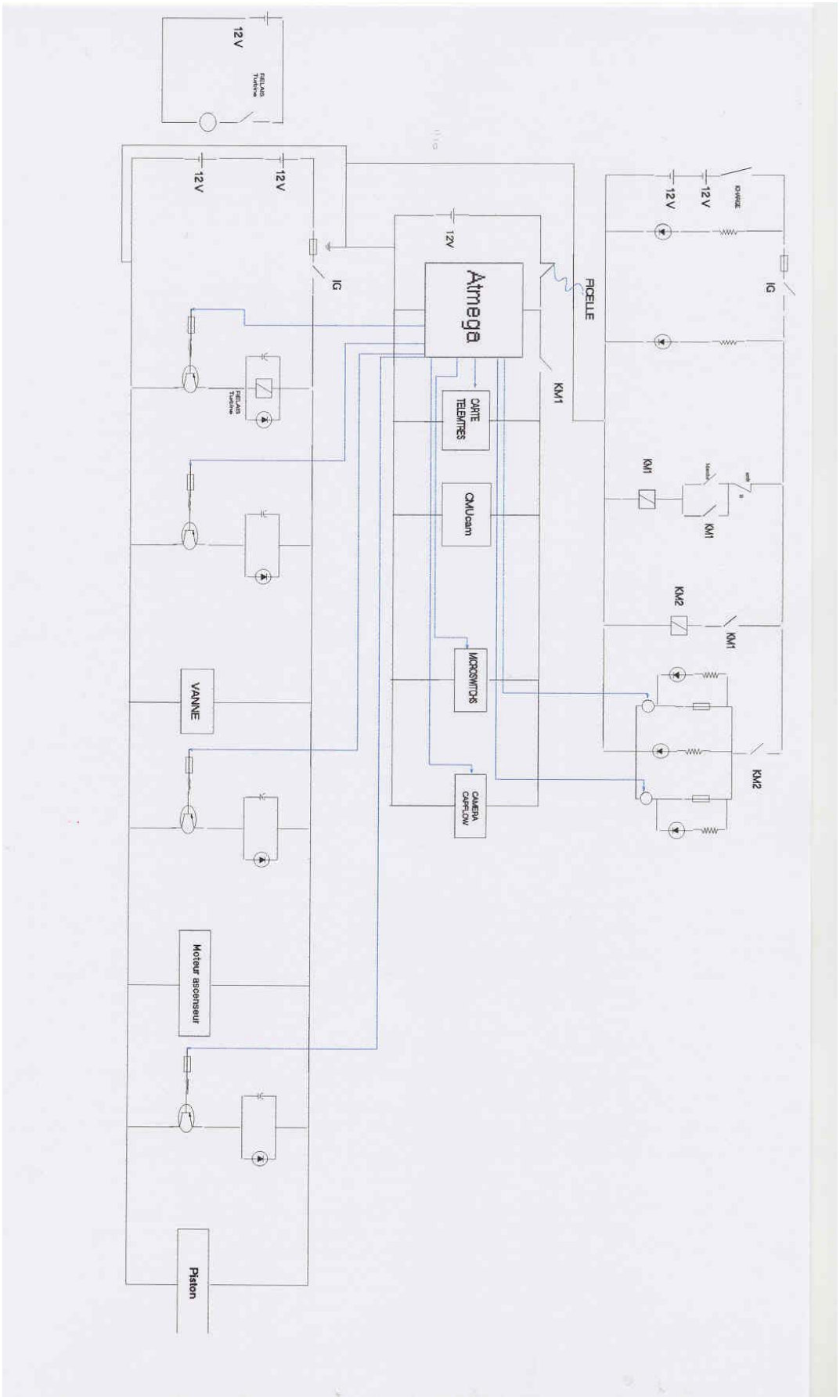



**Figure 54 : schéma de  principe de la carte de puissance**

### 8.2.3 Circuit du milieu

Ce circuit est composé :
- D'une batterie 12V ;
- D'un fusible de 1 A ;
- D'un interrupteur ;
- De diodes de contrôle avant et après fusible ;
- De régulateurs de tension à 9 et à 5V.

Le but du circuit du milieu est d'alimenter l'ensemble des capteurs (télémètres, microswitches, camera, récepteur laser) et l'Atmel en 5, 9 et 12 V. Ce circuit est alimenté par une batterie 12 Volt située à la même masse que les batteries 24V

#### 8.2.3.1 Remarques

- Le respect des masses communes est un aspect fondamental de la bonne marche du robot ;
- Contrairement à l'année 2001-2002 l'alimentation de l'Atmega s'est faite à partir d'une batterie 12 V et non 8 V. Ce choix a été fait en raison de problèmes rencontrés avec l'Atmel lorsque sa tension d'alimentation diminue (< 7 V). Le choix d'une batterie 12 v nous libère plus longtemps de ce problème.

### 8.2.4 Circuit du bas

Le circuit du bas est un circuit alimenté en 24 volts qui sert à alimenter les différents actuateurs tels que l'électrovanne, le moteur de l'ascenseur et la turbine.
Ces différents actuateurs doivent être commandés à partir de l'Atmega ; pour ce faire, le signal (5V) provenant de l'Atmega est envoyé vers un transistor Darlington qui joue le rôle d'interrupteur.

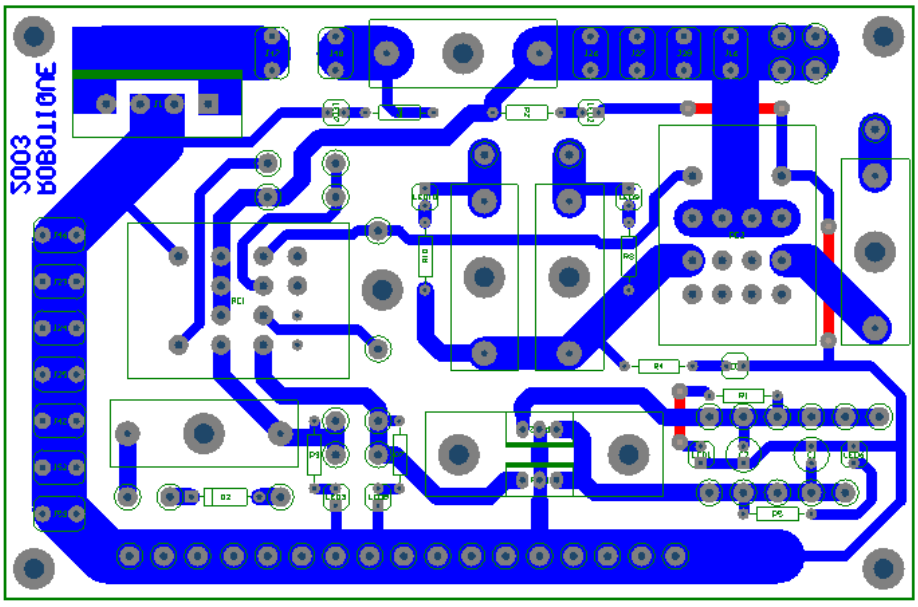


**Figure 55 : Carte de puissance**

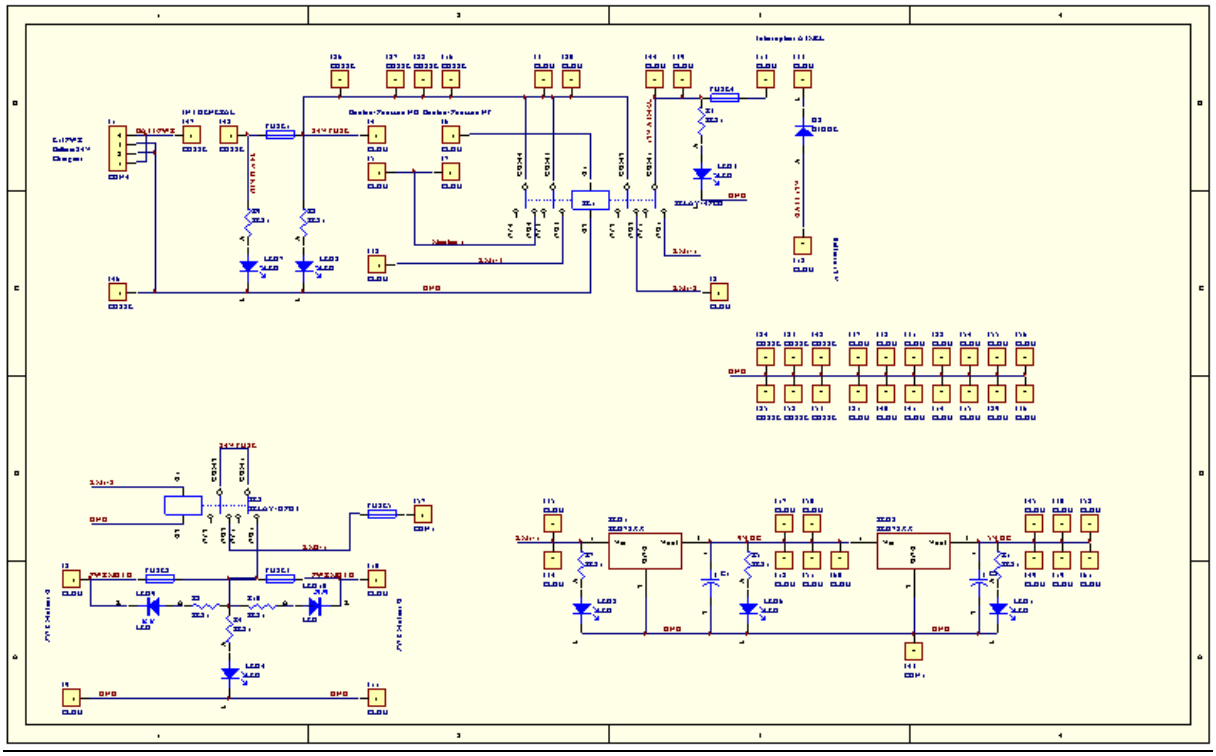


**Figure 56 : schéma de principe de la carte de puissance**

## 8.3 Carte de commande de la vanne, du moteur et des portes

Ces actuateurs ne demandent pas un courant trop important, ils peuvent donc être commandés directement par le transistor sans avoir recours à un relais intermédiaire.

Cette carte se compose :

- D'une diode D4 pour se protéger des inversions de tension ;
- D'un fusible pour se protéger des surintensités ;
- D'une diode mise en série avec une résistance pour vérifier l'état du fusible ;
- De résistances R1 et R2 qui forment un pont diviseur dont le but est de ramener la tension à 5 V au niveau du transistor Darlington ;
- D'un bouton poussoir ;
- D'une branche de commande à partir de l'Atmega ;
- D'une diode D1 dont le rôle est d'empêcher qu'un courant passe vers l'Atmega lorsque le bouton poussoir est activé ;
- D'une résistance R3 dont le rôle est d'encore abaisser la tension ;
- D'un transistor Darlington qui joue le rôle d'interrupteur ;
- Des clous J3 et J4 auxquels sont raccordés le + et le – de l'actuateur considéré ;
- D'une diode de roue libre D3 dont le rôle est d'éviter que le courant revienne vers le transistor lorsque l'on coupe le courant.

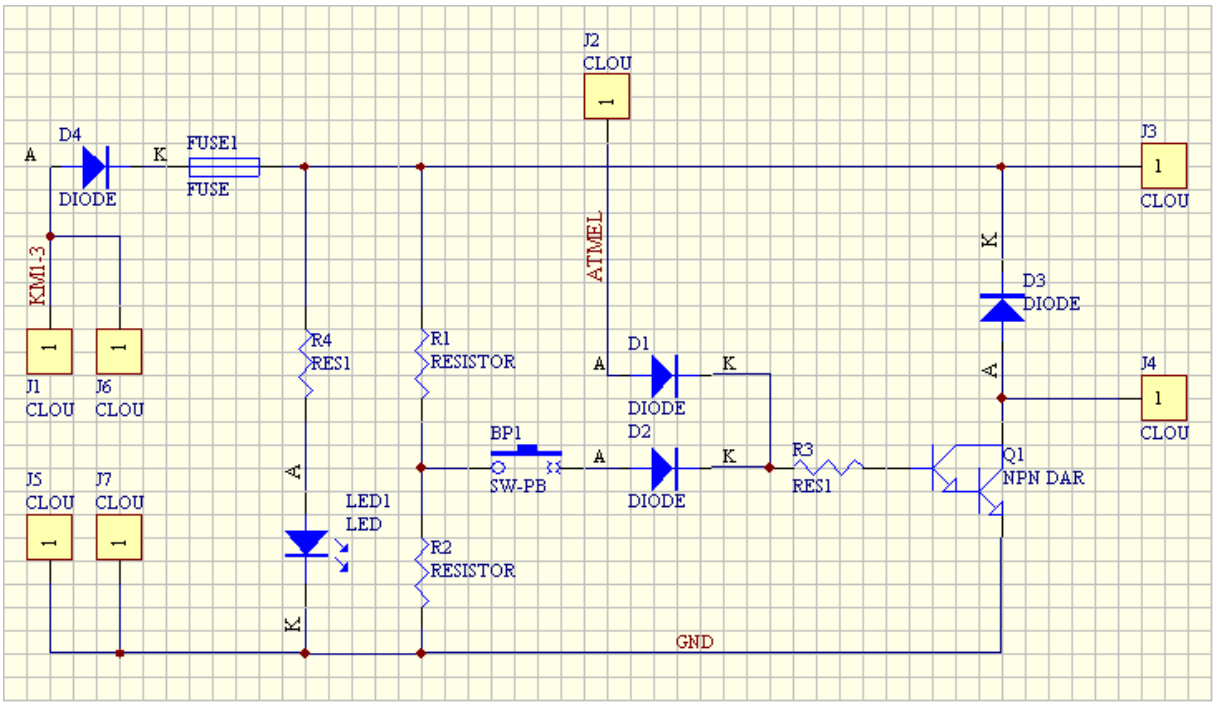



**Figure 57 : schéma de principe de la commande des actuateurs**

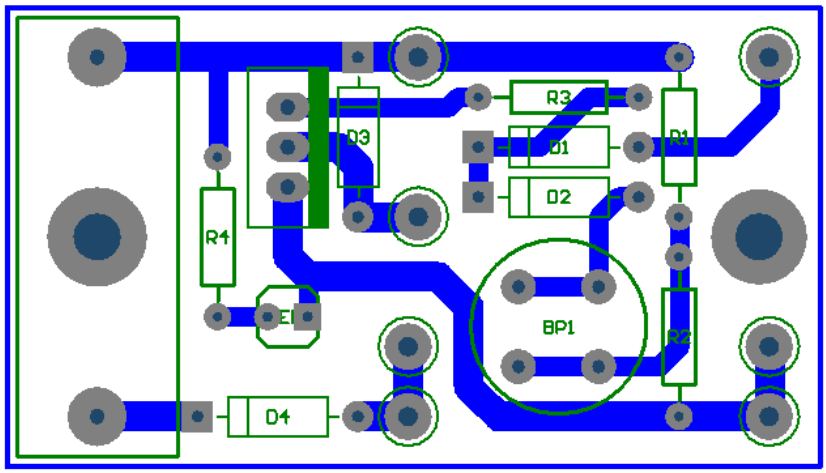



**Figure 58 : carte de commande des actuateurs**

Remarques : le moteur de la turbine est un moteur à courant continu qui tourne à une vitesse élevée (17000 t/min). Afin d'éviter que des parasites dus aux contacts balais – collecteur ne viennent perturber les circuits électroniques, il a été décidé de placer le circuit d'alimentation de la turbine sur un circuit séparé (pas de masse commune avec la carte de puissance). Un relais sera donc nécessaire pour commander la turbine.

Le circuit de cette carte est semblable aux circuits nécessaires à la commande de la vanne, du moteur et des portes de l'ascenseur à la seule différence qu'au lieu de commander l'actuateur directement, le transistor commande un relais. La commutation de ce relais fermera le circuit 12V de commande de la turbine.

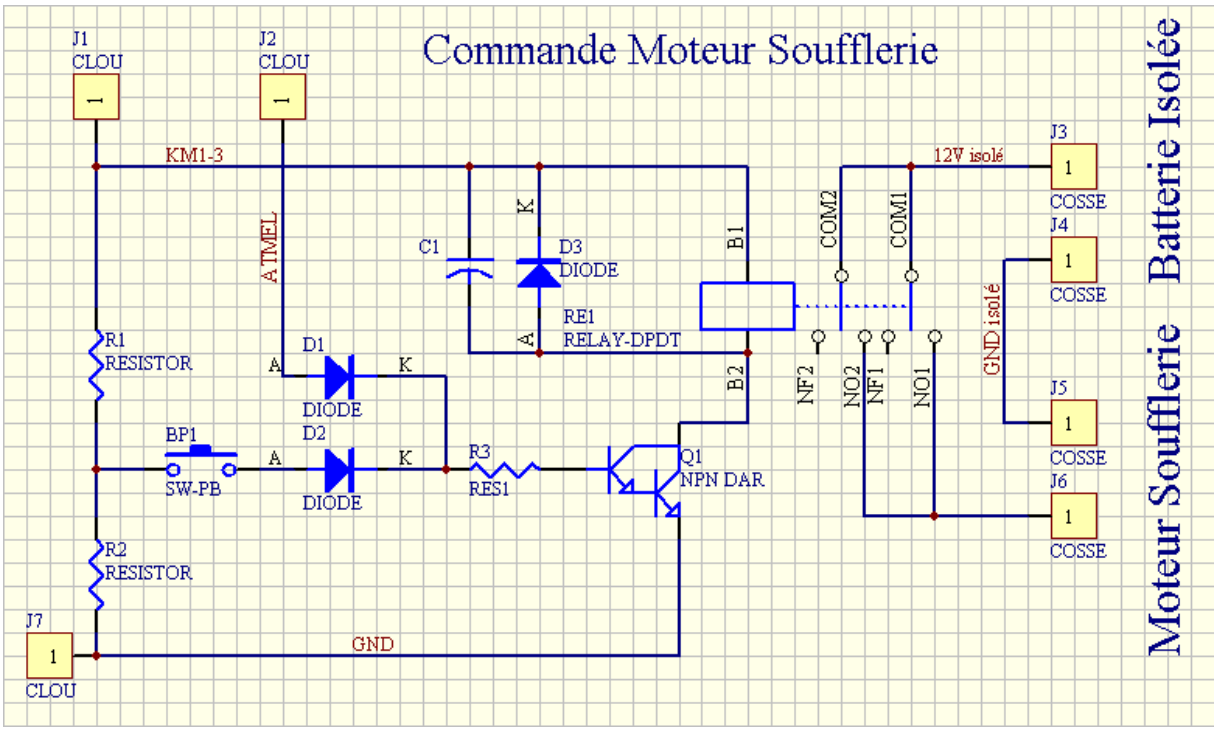



**Figure 59 : schéma de principe de la commande de la turbine**

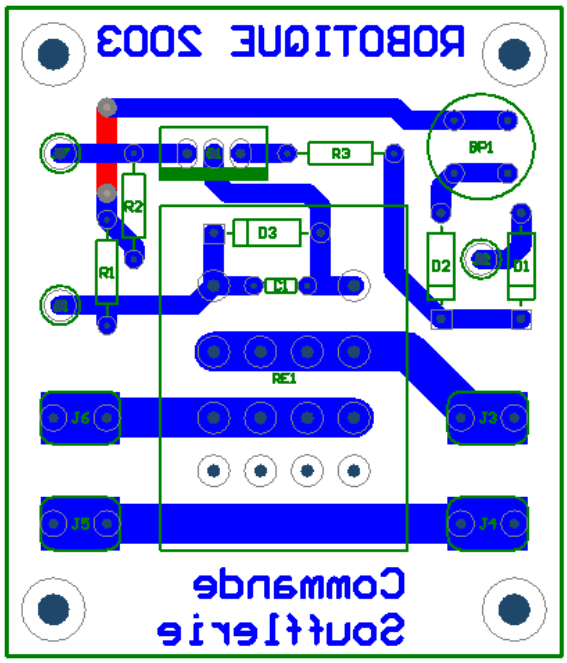



**Figure 60 : carte de commande de la turbine**

# 9. **Conclusions**

Ce projet de robotique nous a permis d'aborder les divers aspects associés à cette discipline. En effet, le robot ne se limite pas à une base mécanique, mais bien à un ensemble de systèmes fortement liés entre eux. On imagine difficilement a priori comment une opération simple en apparence (retourner un palet par exemple), peut se révéler complexe à faire effectuer par un robot. L'électronique et l'informatique jouent un rôle prépondérant dans le fonctionnement du robot. Sans avoir une grande formation dans ces domaines, nous avons pu, grâce aux conseils avisés de tous les membres de l'équipe, réaliser quelque chose de fonctionnel.

L'organisation en équipe projet prévoyait des secteurs d'activités bien spécifiques pour chacun d'entre nous, mais il est rapidement apparu qu'il était nécessaire de comprendre le travail de chacun pour pouvoir concevoir plus harmonieusement notre propre système.
Nous avons également eu la chance de développer un système complexe de la phase d'étude (recherche de concepts, réalisation de plans) à son aboutissement, c'est-à-dire à sa construction. Cette implication dans la construction nous a permis de réaliser quelques-unes unes des diverses réalités du monde industriel : délais de livraison, coût des divers éléments.

Il faut également signaler que ce projet nous a permis de dépasser les objectifs que nous nous étions fixés au départ, à savoir de réaliser un robot capable de s'homologuer pour la coupe de Belgique.

Au final, nous terminons Champion de Belgique et onzièmes au niveau européen, pour une première participation à ce niveau, il s'agit de résultats inespérés.

# 10. Encouragements aux suivants

La réalisation de ce type de projet permet de mettre en pratique un grand nombre de connaissances acquises au fil des études et engendre la grande satisfaction de voir son travail se concrétiser réellement. Nous souhaitons beaucoup de courage et de réussite à ceux qui nous suivrons dans cette aventure palpitante.

# 11. Table des figures

# Annexe 1 : exemple de programme pour la commande du robot

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>
#include <string.h>
#include <stdlib.h>

#include <progmem.h>

#include "uart.h"
#include "uart.c"

#include "config.h"

#ifdef ROBOT_MODE_STOP

#undef ROBOT_SPEED_STOP
#undef ROBOT_SPEED_FORWARD_START
#undef ROBOT_SPEED_FORWARD_BLOW
#undef ROBOT_SPEED_ROTATE_START
#undef ROBOT_SPEED_FORWARD_SLOW
#undef ROBOT_SPEED_FORWARD_FAST
#undef ROBOT_SPEED_FORWARD_EDGE
#undef ROBOT_SPEED_ROTATE_SEARCH
#undef ROBOT_SPEED_ROTATE_LOCATE
#undef ROBOT_SPEED_ROTATE_FAST
#undef ROBOT_SPEED_ROTATE_SLOW
#undef ROBOT_SPEED_BACKWARD

#define ROBOT_SPEED_STOP                0
#define ROBOT_SPEED_FORWARD_START       0
#define ROBOT_SPEED_FORWARD_BLOW        0
#define ROBOT_SPEED_ROTATE_START        0
#define ROBOT_SPEED_FORWARD_SLOW        0
#define ROBOT_SPEED_FORWARD_FAST        0
#define ROBOT_SPEED_FORWARD_EDGE        0
#define ROBOT_SPEED_ROTATE_SEARCH       0
#define ROBOT_SPEED_ROTATE_LOCATE       0
#define ROBOT_SPEED_ROTATE_FAST         0
#define ROBOT_SPEED_ROTATE_SLOW                 0
#define ROBOT_SPEED_BACKWARD            0

#endif

char bin2hex[16] = { '0', '1', '2', '3', '4', '5', '6', '7',
                     '8', '9', 'a', 'b', 'c', 'd', 'e', 'f' };
```

```
void uart_putint (int val)
{
	uart_putc(bin2hex[(val >> 12) & 0x0f]);
	uart_putc(bin2hex[(val >> 8) & 0x0f]);
	uart_putc(bin2hex[(val >> 4) & 0x0f]);
	uart_putc(bin2hex[val & 0x0f]);
}

/* Set an unsigned long value atomically */
inline void atomic_set32 (volatile unsigned long *addr, unsigned long val)
{
	cli ();
	*addr = val;
	sei ();
}

/* Read an unsigned long value atomically */
inline unsigned long atomic_read32 (volatile unsigned long *addr)
{
	unsigned long val;

	cli ();
	val = *addr;
	sei ();

	return val;
}

#define XTAL_CPU 3686400 /*
3.6864 Mhz internal clock */
#define UART_BAUD_RATE 9600 /*
9600 baud RS 232 */
#define TICKS_PER_SEC ((unsigned long)14400) /* 3686400 / 256
*/

enum robot_state
{
	ROBOT_STATE_IDLE = 0, /* Idle, wait for the start signal */
	ROBOT_STATE_START, /* Start procedure (blow discs)
	*/
	ROBOT_STATE_COLLIDE_STOP, /* React to a collision
	*/
	ROBOT_STATE_COLLIDE_GO_AWAY,/* React to a collision */
	ROBOT_STATE_SEARCH_DISC, /* Search for a disc */
	ROBOT_STATE_LOCATE_DISC, /* Locate a disc */
	ROBOT_STATE_FETCH_DISC, /* Fetch a located disc
	*/
	ROBOT_STATE_RANDOM_MOVE, /* Move randomly
	*/
```

```
        ROBOT_STATE_UNLOAD_DISCS,               /* Unload stored discs
                */
        ROBOT_STATE_FOLLOW_EDGE,
        ROBOT_STATE_FOLLOW_EDGE_COLLIDE,
        ROBOT_STATE_END
};

enum robot_direction
{
        ROBOT_DIR_FORWARD = 0,
        ROBOT_DIR_LEFT = 1,
        ROBOT_DIR_RIGHT = 2,
        ROBOT_DIR_BACKWARD = 3
};

enum lift_position
{
        LIFT_POSITION_DOWN = 0,
        LIFT_POSITION_INTERMEDIATE,
        LIFT_POSITION_UP
};

enum door_state
{
        DOOR_STATE_CLOSED = 0,
        DOOR_STATE_OPEN
};

enum turbine_state
{
        TURBINE_STATE_OFF = 0,
        TURBINE_STATE_ON
};

enum disc_color
{
        DISC_COLOR_GREEN = 0,
        DISC_COLOR_RED,
        DISC_COLOR_OTHER
};

typedef struct _sensors_s
{
        unsigned int _unused0       : 1,
                     color_switch   : 1,
                     cam_left       : 1,
                     cam_right      : 1,
                     collision_left : 1,
                     collision_right : 1,
                     color          : 1,
```

```
                telemeter_disc  : 1,
                _unused8        : 1,
                lift_low        : 1,
                lift_middle     : 1,
                lift_high       : 1,
                laser           : 1,
                lift_full       : 1,
                telemeter_left  : 1,
                telemeter_right : 1;
} sensors_t;

typedef struct _motor_s
{
        unsigned int            speed;
        enum robot_direction direction;
} motor_t;

typedef struct _robot_s
{
        enum turbine_state              turbine;

        struct
        {
                enum door_state                 state;
                unsigned long           time;
        } door;

        struct
        {
                motor_t                         target;
                motor_t                         command;
                unsigned long           time;
        } motors;

        struct
        {
                unsigned int            flip;
                unsigned long           time;
        } jack;

        struct
        {
                enum robot_state        state;
                enum robot_state        last_state;
                volatile unsigned long          ticks;
                volatile unsigned long          int_count;
        } state;

        struct
        {
```

```
                enum disc_color         target;
                enum disc_color         current;
        } color;

        struct
        {
                enum robot_direction collision;
        } position;

        struct
        {
                enum lift_position      last_position;
                enum lift_position      position;
                enum lift_position      command;
                unsigned long           time;
                unsigned int            full;
        } lift;

        sensors_t                       sensors;
        sensors_t                       last_sensors;
} robot_t;

robot_t robot;

/*
 *
 */
void robot_move (unsigned int speed, enum robot_direction dir)
{
        if ( robot.motors.target.speed == speed &&
            robot.motors.target.direction == dir )
                return;

        robot.motors.target.speed = speed;
        robot.motors.target.direction = dir;
        robot.motors.time = atomic_read32(&robot.state.int_count)
                        + ROBOT_TIME_MOTOR_DELAY;
}

/*
 * Actuate the lift command according to the sensors and the robot global state.
 */
void robot_lift_actuate (void)
{
#ifdef ROBOT_MODE_HOMOLOGATION
        robot.lift.command = LIFT_POSITION_UP;
        return;
#endif

        switch ( robot.state.state )
```

```
	{
	case ROBOT_STATE_UNLOAD_DISCS:
	case ROBOT_STATE_END:
		/* When unloading the discs at the end of the game, we open the doors,
		 * go backwards for a second, move the lift down and go forward.
		 */
		if ( atomic_read32(&robot.state.ticks) >= ROBOT_TIME_LIFT_UNLOAD )
			robot.lift.command = LIFT_POSITION_DOWN;
		break;

	default:
		/* If the telemeter detects a disc, the lift goes up. */
		if ( robot.sensors.telemeter_disc )
			robot.lift.command = LIFT_POSITION_UP;
		/* If the lift is up and the laser detects a disc, the lift goes down
		 * to grab the disc if not full, or goes intermediate after a delay if full.
		 */
		else if ( robot.sensors.laser &&
			robot.lift.position == LIFT_POSITION_UP )
		{
			if ( robot.lift.full )
				robot.lift.time = atomic_read32(&robot.state.int_count)
					+ ROBOT_TIME_LIFT_UP;
			else
				robot.lift.command = LIFT_POSITION_DOWN;
		}
		/* If the lift is down, it goes to intermediate */
		else if ( robot.lift.position == LIFT_POSITION_DOWN )
			robot.lift.command = LIFT_POSITION_INTERMEDIATE;
		/* If the delay has elapsed, the lift goes to intermediate */
		else if ( robot.lift.time < atomic_read32(&robot.state.int_count) )
		{
			robot.lift.command = LIFT_POSITION_INTERMEDIATE;
			robot.lift.time = (unsigned long)-1;
		}
		break;
	}
}

/*
 * Actuate the pneumatic jack command according to the sensors and the robot global state.
 */
void robot_jack_actuate (void)
{
#ifdef ROBOT_MODE_HOMOLOGATION
	robot.jack.flip = 0;
	return;
#endif

	if ( robot.jack.time <= atomic_read32(&robot.state.int_count) )
```

```
                robot.jack.flip = 0;

        switch ( robot.state.state )
        {
        case ROBOT_STATE_UNLOAD_DISCS:
                /* No point in flipping discs when unloading, as the disc we could flip
                 * will stay under the robot anyway.
                 */
                break;

        default:
                /* If the telemeter detects a disc which is not of our color,
                 * flip it, unless the lift is up. In that case, flipping the disc
                 * could be harmful so it should be avoided.
                 */
                if ( robot.sensors.telemeter_disc &&
                    robot.color.target != robot.color.current &&
                    robot.lift.position != LIFT_POSITION_UP &&
                    robot.jack.flip == 0 )
                {
                        /* Move the pneumatic jack up for one second */
                        robot.jack.time = atomic_read32(&robot.state.int_count)
                                        + ROBOT_TIME_JACK_UP;
                        robot.jack.flip = 1;
                }
                break;
        }
}

/*
 * Actuate the turbine command according to the sensors and the robot global state.
 */
void robot_turbine_actuate (void)
{
        /* Turn on the turbine after turning left */
        if ( robot.state.state == ROBOT_STATE_START &&
            atomic_read32(&robot.state.ticks) >= ROBOT_TIME_START_FORWARD )
                robot.turbine = TURBINE_STATE_ON;
        else
                robot.turbine = TURBINE_STATE_OFF;
}

/*
 * Actuate the rear doors command according to the sensors and the robot global state.
 */
void robot_doors_actuate (void)
{
#if defined(ROBOT_MODE_HOMOLOGATION) || defined(ROBOT_MODE_NO_PILE)
        /* The doors should be opened, but this will be ensured
         * mechanically not to waste energy.
```

```
         */
        robot.door.state = DOOR_STATE_CLOSED;
        return;
#endif

        switch ( robot.state.state )
        {
        case ROBOT_STATE_UNLOAD_DISCS:
                robot.door.state = DOOR_STATE_OPEN;
                break;

        default:
                /* Don't test the lift position, and the robot will
                 * start making piles after piles.
                 */
                if ( robot.sensors.laser &&
                    robot.lift.full &&
                    robot.lift.position == LIFT_POSITION_UP )
                {
                        robot.door.state = DOOR_STATE_OPEN;
                        robot.door.time = atomic_read32(&robot.state.int_count)
                                        + ROBOT_TIME_DOOR_OPEN;
                }

                if ( robot.sensors.collision_left || robot.sensors.collision_right )
                        robot.door.state = DOOR_STATE_CLOSED;

                if ( robot.door.time < atomic_read32(&robot.state.int_count) )
                        robot.door.state = DOOR_STATE_CLOSED;

                break;
        }
}

/*
 *
 */
void robot_motors_actuate_command (void)
{
        if ( robot.motors.command.direction == robot.motors.target.direction )
        {
                /* XXX: Add a linear variation. */
                robot.motors.command.speed = robot.motors.target.speed;
        }
        else
        {
                if ( atomic_read32(&robot.state.int_count) <= robot.motors.time )
                        robot.motors.command.speed = 0;
                else
                {
```

```
                        robot.motors.command.direction = robot.motors.target.direction;
                        robot.motors.command.speed = robot.motors.target.speed;
                }
        }
}

/*
 *
 */
void robot_motors_actuate (void)
{
        switch ( robot.state.state )
        {
        case ROBOT_STATE_IDLE:
                /* In the IDLE state, stop the motors */
                robot_move (ROBOT_SPEED_STOP, ROBOT_DIR_FORWARD);
                break;

        case ROBOT_STATE_START:
                /* Go forward for 2 second, then rotate left for 0.4 ticks,
                 * and go forward.
                 */
                if ( atomic_read32(&robot.state.ticks) < ROBOT_TIME_START_FORWARD
)
                        robot_move(ROBOT_SPEED_FORWARD_START,
ROBOT_DIR_FORWARD);
                else if ( atomic_read32(&robot.state.ticks) <
ROBOT_TIME_START_FORWARD_ROTATE )
                        robot_move(ROBOT_SPEED_ROTATE_START,
ROBOT_DIR_LEFT);
                else
                        robot_move(ROBOT_SPEED_FORWARD_BLOW,
ROBOT_DIR_FORWARD);
                break;

        case ROBOT_STATE_COLLIDE_STOP:
                robot_move (ROBOT_SPEED_STOP, ROBOT_DIR_BACKWARD);
                break;

        case ROBOT_STATE_FOLLOW_EDGE:
                robot_move(ROBOT_SPEED_FORWARD_EDGE,
ROBOT_DIR_FORWARD);
                break;

        case ROBOT_STATE_FOLLOW_EDGE_COLLIDE:
                if ( atomic_read32(&robot.state.ticks) <= TICKS_PER_SEC * 0.3 )
                        robot_move (ROBOT_SPEED_BACKWARD,
ROBOT_DIR_BACKWARD);
                else
```

```
                robot_move (ROBOT_SPEED_ROTATE_SLOW,
ROBOT_DIR_RIGHT);
                break;

        case ROBOT_STATE_COLLIDE_GO_AWAY:
                /* Go backwards for 1 seconds, then rotate for 2 seconds */
                if ( atomic_read32(&robot.state.ticks) <=
ROBOT_TIME_COLLIDE_BACKWARD )
                        robot_move(ROBOT_SPEED_BACKWARD,
ROBOT_DIR_BACKWARD);
                else
                        robot_move(ROBOT_SPEED_ROTATE_FAST,
robot.position.collision);
                break;

        case ROBOT_STATE_SEARCH_DISC:
                if ( robot.sensors.cam_left )
                        robot_move(ROBOT_SPEED_ROTATE_SEARCH,
ROBOT_DIR_LEFT);
                else if ( robot.sensors.cam_right )
                        robot_move(ROBOT_SPEED_ROTATE_SEARCH,
ROBOT_DIR_RIGHT);
                else
                        robot_move(ROBOT_SPEED_ROTATE_SEARCH,
robot.position.collision);
                break;

        case ROBOT_STATE_LOCATE_DISC:
                if ( robot.sensors.cam_left )
                        robot_move(ROBOT_SPEED_ROTATE_LOCATE,
ROBOT_DIR_LEFT);
                else if ( robot.sensors.cam_right )
                        robot_move(ROBOT_SPEED_ROTATE_LOCATE,
ROBOT_DIR_RIGHT);
                else
                        robot_move(ROBOT_SPEED_ROTATE_LOCATE,
robot.position.collision);
                break;

        case ROBOT_STATE_FETCH_DISC:
                robot_move(ROBOT_SPEED_FORWARD_FAST,
ROBOT_DIR_FORWARD);
                break;

        case ROBOT_STATE_RANDOM_MOVE:
                robot_move(ROBOT_SPEED_ROTATE_FAST, robot.position.collision);
                break;

        case ROBOT_STATE_UNLOAD_DISCS:
                /* Go backwards for 2 seconds, then go forward at high speed */
```

```
                if ( atomic_read32(&robot.state.ticks) <
ROBOT_TIME_UNLOAD_BACKWARD )
                        robot_move(ROBOT_SPEED_BACKWARD,
ROBOT_DIR_BACKWARD);
                else
                        robot_move(ROBOT_SPEED_FORWARD_FAST,
ROBOT_DIR_FORWARD);
                break;

        case ROBOT_STATE_END:
                /* In the END state, stop the motors */
                robot_move (ROBOT_SPEED_STOP, ROBOT_DIR_FORWARD);
                break;
        }

        robot_motors_actuate_command();
}

/*
 *
 */
void robot_fsm_step (void)
{
        static int trigger_edge = 0;
#if !defined(ROBOT_MODE_NO_PILE)
        static int trigger_unload = 0;
#endif
        static int trigger_end = 0;

        enum robot_state new_state = robot.state.state;

        /* If the robot has spent 7 seconds in the same state, go to
         * ROBOT_STATE_COLLIDE_GO_AWAY.
         */
        if ( atomic_read32(&robot.state.ticks) >= ROBOT_TIME_MAX_STATE_TIME )
        {
                /* If the robot has been in the same state for more than
                 * MAX_STATE_TIME seconds, switch to a new state.
                 */
                atomic_set32(&robot.state.ticks, 0);
           new_state = ROBOT_STATE_COLLIDE_GO_AWAY;
        }


        if ( atomic_read32(&robot.state.int_count) >=
ROBOT_TIME_FOLLOW_EDGE_TIME &&
            trigger_edge == 0 )
        {
                new_state = ROBOT_STATE_FOLLOW_EDGE;
                trigger_edge = 1;
```

```
        }
#if !defined(ROBOT_MODE_NO_PILE)
        /* After 85 seconds, go to ROBOT_STATE_UNLOAD_DISCS. */
        else if ( atomic_read32(&robot.state.int_count) >= ROBOT_TIME_UNLOAD_TIME
&&
                  trigger_unload == 0 )
        {
                new_state = ROBOT_STATE_UNLOAD_DISCS;
                trigger_unload = 1;
        }
#endif
        /* After 90 seconds, go to ROBOT_STATE_END. */
        else if ( atomic_read32(&robot.state.int_count) >= ROBOT_TIME_END_TIME &&
                  trigger_end == 0 )
        {
                new_state = ROBOT_STATE_END;
                trigger_end = 1;
        }

        switch (robot.state.state)
        {
        case ROBOT_STATE_IDLE:
                /* Wait for the start signal */
                if ( atomic_read32(&robot.state.ticks) >= ROBOT_TIME_START_DELAY )
                        new_state = ROBOT_STATE_START;
                break;

        case ROBOT_STATE_START:
                /* Wait for a collision with the edge */
                if ( robot.sensors.collision_left || robot.sensors.collision_right )
#ifdef ROBOT_MODE_HOMOLOGATION
                        new_state = ROBOT_STATE_END;
#else
                        new_state = ROBOT_STATE_COLLIDE_STOP;
#endif
                break;

        case ROBOT_STATE_COLLIDE_STOP:
                /* Store positional information if a collision has just been detected */
                if ( robot.state.last_state != ROBOT_STATE_COLLIDE_STOP )
                {
                        if ( robot.last_sensors.collision_left )
                                robot.position.collision = ROBOT_DIR_RIGHT;
                        else
                                robot.position.collision = ROBOT_DIR_LEFT;
                }

                /* Go away if the laser doesn't detect a disc, or if the lift is full, or if
                   the lift is at its intermediate position. This ensures that, if the lift is
                   not full, it will have time to grab the disc before we go away. */
```

```
                if ( !robot.sensors.laser ||
                    robot.lift.full ||
                    robot.lift.position == LIFT_POSITION_INTERMEDIATE )
                        new_state = ROBOT_STATE_COLLIDE_GO_AWAY;
                break;

        case ROBOT_STATE_COLLIDE_GO_AWAY:
                /* Wait 3 second while we go backwards and rotate, and start to search discs */
                if ( atomic_read32(&robot.state.ticks) >=
ROBOT_TIME_COLLIDE_BACKWARD_ROTATE )
#ifdef ROBOT_MODE_CAMERA
                        new_state = ROBOT_STATE_SEARCH_DISC;
#else
//                      new_state = ROBOT_STATE_RANDOM_MOVE;
                        new_state = ROBOT_STATE_FETCH_DISC;
#endif
                break;

        case ROBOT_STATE_SEARCH_DISC:
                if ( robot.sensors.collision_left || robot.sensors.collision_right )
                {
                        new_state = ROBOT_STATE_COLLIDE_STOP;
                        break;
                }

                /* Wait until we found a disc */
                if ( (robot.sensors.cam_left && robot.sensors.cam_right) ||
                    (robot.sensors.cam_left && robot.last_sensors.cam_right) ||
                    (robot.sensors.cam_right && robot.last_sensors.cam_left) )
                      /* The camera indicates that a disc has been found either by telling the
                       * robot to go forward. If we 'jumped' over a disc while searching, the
                       * camera might tell us to change the rotation direction. Thus, if the
                       * camera gives us a different direction that at the last clock tick,
                       * we consider that we found a disc. */
                        new_state = ROBOT_STATE_LOCATE_DISC;
                else if ( atomic_read32(&robot.state.ticks) >=
ROBOT_TIME_SEARCH_DISC )
                        /* No disc has been found after rotating for 3 seconds. Fall back to
                         * random mode.
                         */
                        new_state = ROBOT_STATE_RANDOM_MOVE;
                break;

        case ROBOT_STATE_LOCATE_DISC:
                if ( robot.sensors.collision_left || robot.sensors.collision_right )
                {
                        new_state = ROBOT_STATE_COLLIDE_STOP;
                        break;
                }
```

```
                if ( (robot.sensors.cam_left && robot.sensors.cam_right) ||
                    (robot.sensors.cam_left && robot.last_sensors.cam_right) ||
                    (robot.sensors.cam_right && robot.last_sensors.cam_left) )
                        new_state = ROBOT_STATE_FETCH_DISC;
                else if ( atomic_read32(&robot.state.ticks) >=
ROBOT_TIME_LOCATE_DISC )
                        new_state = ROBOT_STATE_RANDOM_MOVE;
                break;

        case ROBOT_STATE_FETCH_DISC:
                /* Go forward until we hit the edge */
                if ( robot.sensors.collision_left || robot.sensors.collision_right )
                        new_state = ROBOT_STATE_COLLIDE_STOP;
                break;

        case ROBOT_STATE_RANDOM_MOVE:
                if ( robot.sensors.collision_left || robot.sensors.collision_right )
                {
                        new_state = ROBOT_STATE_COLLIDE_STOP;
                        break;
                }

                /* Rotate randomly and pretend we found a disc */
                if ( atomic_read32(&robot.state.ticks) >= ROBOT_TIME_RANDOM_MOVE
)
                        new_state = ROBOT_STATE_FETCH_DISC;
                break;

        case ROBOT_STATE_FOLLOW_EDGE:
                if ( robot.sensors.collision_left || robot.sensors.collision_right )
                        new_state = ROBOT_STATE_FOLLOW_EDGE_COLLIDE;
                break;

        case ROBOT_STATE_FOLLOW_EDGE_COLLIDE:
                if ( ( atomic_read32(&robot.state.ticks) >= TICKS_PER_SEC * 1 ) &&
                    robot.sensors.telemeter_left )
                        new_state = ROBOT_STATE_FOLLOW_EDGE;
                break;

        case ROBOT_STATE_UNLOAD_DISCS:
                if ( robot.sensors.collision_left || robot.sensors.collision_right )
                        new_state = ROBOT_STATE_END;
                break;

        case ROBOT_STATE_END:
                /* Don't trigger the 'stuck avoidance algorithm' */
                atomic_set32(&robot.state.ticks, 0);
                new_state = ROBOT_STATE_END;
                break;
        }
```

```
	/* Update the state and reset the counters */
	robot.state.last_state = robot.state.state;
	if ( robot.state.state != new_state )
	{
		atomic_set32(&robot.state.ticks, 0);
		robot.state.state = new_state;
	}

	/* Actuate the robot commands */
	robot_motors_actuate();
	robot_lift_actuate();
	robot_jack_actuate();
	robot_turbine_actuate();
	robot_doors_actuate();
}

/*
 * Read the state of all the sensors
 */
void robot_read_sensors (void)
{
	robot.last_sensors = robot.sensors;
	*(unsigned int*)&robot.sensors = ( inp(PIND) << 8 ) | inp(PINA);

	/* Check the lift position */
	if ( robot.sensors.lift_low && robot.sensors.lift_high )
	{
		robot.lift.position = LIFT_POSITION_DOWN;
		robot.lift.full = robot.sensors.lift_full;
	}
	else if ( ! robot.sensors.lift_low &&
		  ! robot.sensors.lift_middle &&
		robot.sensors.lift_high )
	{
		if ( robot.lift.last_position == LIFT_POSITION_DOWN )
			robot.lift.position = LIFT_POSITION_INTERMEDIATE;
	}
	else if ( ! robot.sensors.lift_low &&
		! robot.sensors.lift_high )
		robot.lift.position = LIFT_POSITION_UP;

#ifdef ROBOT_MODE_NO_PILE
	robot.lift.full = 1;
#endif
	robot.lift.last_position = robot.lift.position;

	/* Check the detected disc color */
	if ( robot.sensors.color )
		robot.color.current = DISC_COLOR_GREEN;
```

```
	else
		robot.color.current = DISC_COLOR_RED;

	/* Select the color target */
	if ( robot.state.state == ROBOT_STATE_IDLE )
		robot.color.target = robot.sensors.color_switch ?
				DISC_COLOR_RED : DISC_COLOR_GREEN;
}

/*
 *
 */
void robot_update_commands (void)
{
	/* Set the motors direction */
	if ( robot.motors.command.direction & 2 )
		cbi(PORTB, 1);
	else
		sbi(PORTB, 1);

	if ( robot.motors.command.direction & 1 )
		sbi(PORTB, 3);
	else
		cbi(PORTB, 3);

	/* Set the motors speed.
	 * Apply the speed correction to compensate for mechanical problems.
	 */
	__outw (robot.motors.command.speed * ROBOT_SPEED_CORRECTION,
OCR1AL);
	__outw (robot.motors.command.speed, OCR1BL);

	/* Turn on the lift motor if the lift position if different
	 * from the command.
	 */
	if ( robot.lift.command != robot.lift.position )
		sbi (PORTB, 7);
	else
		cbi (PORTB, 7);

	if ( robot.jack.flip )
		sbi (PORTB, 4);
	else
		cbi (PORTB, 4);

	if ( robot.turbine == TURBINE_STATE_ON )
		sbi (PORTC, 0);
	else
		cbi (PORTC, 0);
```

```
        if ( robot.door.state == DOOR_STATE_OPEN )
                sbi (PORTC, 4);
        else
                cbi (PORTC, 4);
}

void robot_init (void)
{
      /* Initialize the state variables */
        memset( &robot, 0, sizeof robot );
        robot.lift.command = LIFT_POSITION_INTERMEDIATE;
        robot.state.state = ROBOT_STATE_IDLE;
        robot.state.last_state = ROBOT_STATE_IDLE;
        robot.motors.command.direction = ROBOT_DIR_FORWARD;
        robot.motors.target.direction = ROBOT_DIR_FORWARD;

        robot.door.time   = (unsigned long)-1;
        robot.motors.time = (unsigned long)-1;
        robot.jack.time   = (unsigned long)-1;
        robot.lift.time   = (unsigned long)-1;

        /* Read the sensors */
        robot_read_sensors();
}

#ifdef DEBUG
void uart_print_state (enum robot_state state)
{
        switch ( state )
        {
        case ROBOT_STATE_IDLE:
                uart_puts ("ROBOT_STATE_IDLE\r\n");
                break;
        case ROBOT_STATE_START:
                uart_puts ("ROBOT_STATE_START\r\n");
                break;
        case ROBOT_STATE_COLLIDE_STOP:
                uart_puts ("ROBOT_STATE_COLLIDE_STOP\r\n");
                break;
        case ROBOT_STATE_COLLIDE_GO_AWAY:
                uart_puts ("ROBOT_STATE_COLLIDE_GO_AWAY\r\n");
                break;
        case ROBOT_STATE_SEARCH_DISC:
                uart_puts ("ROBOT_STATE_SEARCH_DISC\r\n");
                break;
        case ROBOT_STATE_LOCATE_DISC:
                uart_puts ("ROBOT_STATE_LOCATE_DISC\r\n");
                break;
        case ROBOT_STATE_FETCH_DISC:
                uart_puts ("ROBOT_STATE_FETCH_DISC\r\n");
```

```
                break;
        case ROBOT_STATE_RANDOM_MOVE:
                uart_puts ("ROBOT_STATE_RANDOM_MOVE\r\n");
                break;
        case ROBOT_STATE_UNLOAD_DISCS:
                uart_puts ("ROBOT_STATE_UNLOAD_DISCS\r\n");
                break;
        case ROBOT_STATE_END:
                uart_puts ("ROBOT_STATE_END\r\n");
                break;
        }
}
#endif

/*
 * Timer0 overflow interrupt handler
 */
SIGNAL (SIG_OVERFLOW0)
{
        /* Update the time counters */
        robot.state.int_count++;
        robot.state.ticks++;
}

/*
 * void ioinit (void)
 */
void ioinit (void)
{
        /* Initialize the IO ports */
        outp (0x00, DDRA);  /* PA7:PA0 : input high-Z    */
        outp (0x00, PORTA);
        outp (0xff, DDRB);    /* PB0:PB7 : output low                */
        outp (0x00, PORTB);
        outp (0x00, DDRD);  /* PD0:PD7 : input high-Z    */
        outp (0x00, PORTD);

        /* Setup Timer 1 (PWM mode) */
        outp (BV (PWM10) | BV (COM1B1) | BV (COM1A1), TCCR1A);        /* tmr1 8-bit
PWM */
        outp (BV (CS10), TCCR1B);/* tmr1  full MCU clock */
        sbi (PORTB, 0);                         /* Power on right motor */
        sbi (PORTB, 2);                         /* Power on left motor */

        /* Start the motors, speed 0 */
        __outw (0, OCR1AL);
        __outw (0, OCR1BL);

        /* Setup Timer 0 */
        outp ((1 << TOIE0), TIMSK);         /* enable TCNT0 overflow */
```

```
	outp (0, TCNT0);		/* reset TCNT0 */

	/* Enable interrupts globally */
	sei ();
}

/*
 * int main (void)
 */
int main (void)
{
	/* Initialize the IO ports and internal registers */
	ioinit ();

	/* Initialize the robot state */
	robot_init();

#ifdef DEBUG
	uart_init (UART_BAUD_SELECT (UART_BAUD_RATE, XTAL_CPU));
	uart_puts("Eurobot\r\n");
#endif

	/* Start timer 0 */
	outp (1, TCCR0);		/* tmr0 running on full MCU clock */

	while ( robot.state.state != ROBOT_STATE_END )
	{
#ifdef DEBUG
		enum robot_state last_state = robot.state.state;
#endif

		/* Advance the FSM by one step */
		robot_read_sensors();
		robot_fsm_step();
		robot_update_commands();

#ifdef DEBUG
		/* Print the state on the serial port */
		if ( robot.state.state != last_state )
			uart_print_state(robot.state.state);
		if ( robot.turbine == TURBINE_STATE_ON )
			uart_putc('T');
#endif
	}

	/* Do something fun here to impress the audience :-) */

	/* Power off the motors. */
	cbi (PORTB, 0);			/* Power off right motor */
	cbi (PORTB, 2);			/* Power off left motor */
```

```
        /* I don't know what happens after main() returns, so don't tempt the
         * devil.
         */
        while ( 1 ) {};
        return 0;
}
```

# Annexe 2 : Traitements d'image en C pour la caméra CAMELEON de Capflow

**Inversion de l'image**

```
void robot_swapy(unsigned int x_size, unsigned int y_size, unsigned char * img_data)
{
 unsigned int x,y;
 //unsigned char buf[1280];
 unsigned char pixbuf;

        for ( y = 0 ; y < (y_size>>1); y++ )
      {
                 //memcpy(buf, (img_data+((y_size-1-y)*x_size)), x_size);
                 //memcpy( (img_data+((y_size-1-y)*x_size)), (img_data+(y*x_size)), x_size);
                 //memcpy((img_data+(y*x_size)), buf, x_size);
                for (x=0; x < x_size ; x++)
                {
                pixbuf = img_data [x + ((y_size-y-1)*x_size)]; /* on mémorise le bas */
                img_data [x + ((y_size-y-1)*x_size)] = img_data [x+(y*x_size)]; /* on inverse
haut sur bas */
                img_data [x+(y*x_size)] = pixbuf; /* on stocke le bas sur le haut */
                }
      }

}
```

**Calibration**

```
void robot_white_balance(unsigned int x_size, unsigned int y_size, unsigned char *img_data,
unsigned int waitimage)
{
        unsigned int n,w,x,y; /* n ecart par rapport au bord et w largeur du cadre */
        unsigned long sumrouge, sumvert, sumbleu;

        sumrouge = 0;
        sumvert = 0;
        sumbleu = 0;

        n = 50;
        w = 50;
// pour le rouge
        for ( y = n ; y < n+w; y+=2)
      for ( x = n ; x < n+w; x+=2 )
                {
                 img_data[(y)*x_size+(x)] = img_data[(y+1)*x_size+(x+1)];
                 img_data[(y)*x_size+(x+1)] = img_data[(y+1)*x_size+(x+1)];
                 img_data[(y+1)*x_size+(x)] = img_data[(y+1)*x_size+(x+1)];
                 sumrouge += img_data[(y+1)*x_size+(x+1)];
                }
```

```
// pour le vert
        for ( y = n ; y < n+w; y+=2)
      for ( x = 2*n ; x < (2*n)+w; x+=2 )
                  {
                   img_data[(y)*x_size+(x)] = img_data[y*x_size+(x+1)];
                   img_data[(y+1)*x_size+(x)] = img_data[y*x_size+(x+1)];
                   img_data[(y+1)*x_size+(x+1)] = img_data[y*x_size+(x+1)];
                   sumvert += img_data[y*x_size+(x+1)];
                  }

// pour le bleu
        for ( y = n ; y < n+w; y+=2)
      for ( x = 3*n ; x < (3*n)+w; x+=2 )
                  {
                   img_data[(y)*x_size+(x+1)] = img_data[y*x_size+(x)];
                   img_data[(y+1)*x_size+(x)] = img_data[y*x_size+(x)];
                   img_data[(y+1)*x_size+(x+1)] = img_data[y*x_size+(x)];
                   sumbleu += img_data[y*x_size+(x)];
                  }

  sumrouge = sumrouge / 625; /*auparavant 10000*/
  sumvert = sumvert / 625;
  sumbleu = sumbleu / 625;

  if (waitimage == 200)
          img_data[0] = 0;

}
```

**Définition des seuils**

```
void robot_seuils(unsigned int x_size, unsigned int y_size, unsigned char * img_data,
                                        unsigned char *seuilrougehaut, unsigned char
*seuilverthaut, unsigned char *seuilbleuhaut,
                                        unsigned char *seuilrougebas, unsigned char
*seuilvertbas, unsigned char *seuilbleubas)
{


 *seuilrougehaut = 100;
 *seuilbleuhaut = 100;
 *seuilverthaut = 100;
 *seuilrougebas = 60;
 *seuilbleubas = 50;
 *seuilvertbas = 60;


}
```

**Quantification**

```
void robot_quantifie(unsigned int x_size, unsigned int y_size, unsigned char * img_data,
                     unsigned char seuilrougehaut, unsigned char seuilverthaut, unsigned char seuilbleuhaut,
                     unsigned char seuilrougebas, unsigned char seuilvertbas, unsigned char seuilbleubas)
{
unsigned int  x,y,i;
unsigned char R,G,B,Rb, Bb, Gb, index, couleur;
unsigned char quantif_LUT [64];

    /* initialisation du tableau quantif_LUT*/

    for ( i=0 ; i < 64 ; i++)
    {
        quantif_LUT [i] = 0;
    }
    quantif_LUT[36] = 2;
    quantif_LUT[37] = 2;
    quantif_LUT[32] = 2;
    quantif_LUT[20] = 1;
    quantif_LUT[21] = 1;
    quantif_LUT[4] = 1;
    quantif_LUT[5] = 1;
    quantif_LUT[24] = 1;
    quantif_LUT[25] = 1;

    for ( y = 0 ; y < y_size-2; y+=2 )
  for ( x = 0 ; x < x_size-2; x+=2 )
        {
            R = img_data[(y+1)*x_size +(x+1)];
            B = img_data[y*x_size + x];
            G = img_data[y*x_size+(x+1)];

            if (R <= seuilrougebas)
                Rb = 0;
            else
                {
                if (R >= seuilrougehaut)
                    Rb =2;
                else Rb=1;
                }

            if (B <= seuilbleubas)
                Bb = 0;
            else
                {
                if (B >= seuilbleuhaut)
```

```
                        Bb =2;
                    else Bb=1;
                    }

                if (G <= seuilvertbas)
                    Gb = 0;
                else
                    {
                    if (G >= seuilverthaut)
                        Gb =2;
                    else Gb=1;
                    }

                index = (Bb + Gb*4 + Rb *16);

                couleur = quantif_LUT [index];
                img_data[(y+1)*x_size +(x+1)] = couleur;
                img_data[y*x_size + x] = couleur;
                img_data[(y+1)*x_size+x] = couleur;
                img_data[y*x_size+(x+1)] = couleur;
            }

}
```

**Remplissage horizontal**

```
void robot_h_fill(unsigned int x_size, unsigned int y_size, unsigned char *img_data, unsigned
char n)
{
 unsigned char current_pix;
 unsigned int x,y,i;

      for ( y = 0 ; y < y_size; y++)
    for ( x = 0 ; x < x_size; x++ )
            {
                if (x+n <= x_size)
                    if (img_data[y*x_size +x] == img_data[y*x_size+(x+n)])
                    {
                        current_pix = img_data [y*x_size+x];
                        for (i = x ; i < x+n ; i++)
                        {
                        img_data[y*x_size+i] = current_pix;
                        }

                    }

            }
}
```

**Erosion**

```
void robot_erode(unsigned int x_size, unsigned int y_size, unsigned char * img_data, unsigned int n) /* travail sur rouge et vert séparement */
{

unsigned int x,y,i;
unsigned char *prev_line, *cur_line,*next_line, *tempechange;


prev_line = (unsigned char * ) malloc (x_size);
cur_line = (unsigned char * ) malloc (x_size);
next_line = (unsigned char * ) malloc (x_size);

for (i = 1 ; i < n ; i++)
{
/* initialisation */
for (x = 0 ; x < x_size; x++)
{
        prev_line[x] = img_data [x];
        cur_line[x] = img_data [x + x_size];
        next_line[x] = img_data [x+(2*x_size)];

}

/* comparaison 8 pixels voisins*/

        for ( y = 1 ; y < y_size-1; y++ )
        {
      for ( x = 1 ; x < x_size-1; x++ )
                {
                        if  ((img_data[(x-1)+y*x_size] == img_data[(x-1)+(y-1)*x_size])
                         && (img_data[(x-1)+y*x_size] == img_data[ x   +(y-1)*x_size])
                         && (img_data[(x-1)+y*x_size] == img_data[(x+1)+(y-1)*x_size])
                         &&  (img_data[(x-1)+y*x_size]  ==  img_data[(x+1)+    y    *x_size])
                         && (img_data[(x-1)+y*x_size] == img_data[(x+1)+(y+1)*x_size])
                         && (img_data[(x-1)+y*x_size] == img_data[(x  )+(y+1)*x_size])
                         && (img_data[(x-1)+y*x_size] == img_data[(x-1)+(y+1)*x_size]) )
                                        cur_line[x] = img_data[(x-1)+y*x_size];
                                else
                                        cur_line[x]=0;


                }


                /*passage à la ligne suivante*/
```

```
                for ( x = 0 ; x < x_size-1; x++ )
                        img_data[x+(y-1)*x_size]=prev_line[x];

        tempechange = prev_line;
                prev_line=cur_line;
                cur_line=next_line;
                next_line = tempechange;

                for ( x = 0 ; x < x_size-1; x++ )
                        next_line[x]=img_data[x+(y+1)*x_size];
        }
}
}
```

**Labellisation**

```
void robot_collage(unsigned int x_size, unsigned int y_size, unsigned char * img_data, Tsegment *segment, Tpalet palets[]) /* on trouve les 4 premiers palets */
{
unsigned int i,centre;
centre = (segment->xmin+segment->xmax) >> 1;

for (i = 0; i<MAX_PALETS ; i++)
{
                if (palets[i].xmax == 0)  // assigner les valeurs si c'est le premier segment du palet
                {
                        palets[i].xmax = segment->xmax;
                        palets[i].xmin = segment->xmin;
                        palets[i].ycurrent = segment->y;
                        palets[i].ymax = segment->y;
                        palets[i].ymin = segment->y;
                        palets[i].color = segment->color;
                        break;
                }
                if ((palets[i].xmin < centre) && (palets[i].xmax > centre)  && (segment->y == (palets[i].ycurrent)+1))
                // on s'assure que centre est entre x min t xmax

        {
                        if (palets[i].xmin > segment->xmin)
                                palets[i].xmin = segment->xmin;
                        if (palets[i].xmax < segment->xmax)
                                palets[i].xmax = segment->xmax;
                        palets[i].ymax = segment->y;
                        palets[i].ycurrent=segment->y;
                        break;
                }
}
```

```
}
```

**Commande du robot**

```
int setRobotIO(unsigned char direction, unsigned char state)
{

 if (state == 0)
 {
 regvalue = regvalue & (255-direction);
 }
 else
 {
  regvalue = regvalue | direction;
 }

 DO_WritePort(card_num,0,regvalue);

 return(1);
}


void robot_commande(Tpalet palets[], unsigned int x_size,unsigned int couleur_cherchee)
{
unsigned int i, palet_centre, sortie1, sortie2;
int status;

  for (i = 0; i<MAX_PALETS ; i++)
  {
        // 2.3 CONTROLE DE LA DIRECTION DU ROBOT

        if (palets[i].color == couleur_cherchee)
        {
                palet_centre=(palets[i].xmax+palets[i].xmin) >> 1;
                if (palet_centre < ((x_size >> 1) - TOLERANCE))
                        {
                         status = setRobotIO(GAUCHE,ON);
                         status = setRobotIO(DROITE,OFF);
                         break;
                        }

                if (palet_centre > ((x_size >> 1) + TOLERANCE))
                        {
                         status = setRobotIO(GAUCHE,OFF);
                         status = setRobotIO(DROITE,ON);
                         break;
                        }

                if ( (palet_centre > ((x_size >> 1) - TOLERANCE)) && (palet_centre <
((x_size >> 1) + TOLERANCE)))
                        {
```

```
                        status = setRobotIO(GAUCHE,ON);
                        status = setRobotIO(DROITE,ON);
                        break;
                        }
        }
  }
}
```

# Annexe 3 : entrées / sorties de l'Atmega

| Entrées | | Sorties | |
|---|---|---|---|
| PA4 | Switch avant gauche | PB4 | Vérin |
| PA5 | Switch avant droit | | |
| PA6 | capteur couleur | | |
| PD1 | Switch ascenseur bas | PB7 | Moteur ascensceur |
| PD2 | Switch ascenseur intermédiaire | | |
| PD3 | Switch ascenseur haut | | |
| PA7 | Télémètre palet | PC0 | Moteur turbine |
| PD4 | Laser | | |
| PD5 | Switch ascenseur plein | | |
| PA2 | Palet à gauche (caméra) | PC4 | Electro-aimant (porte arrière) |
| PA3 | Palet à droite (caméra) | | |
| PA2 + PA3 | Palet en face du robot | | |
| PA0 | Interupteur couleur | | |