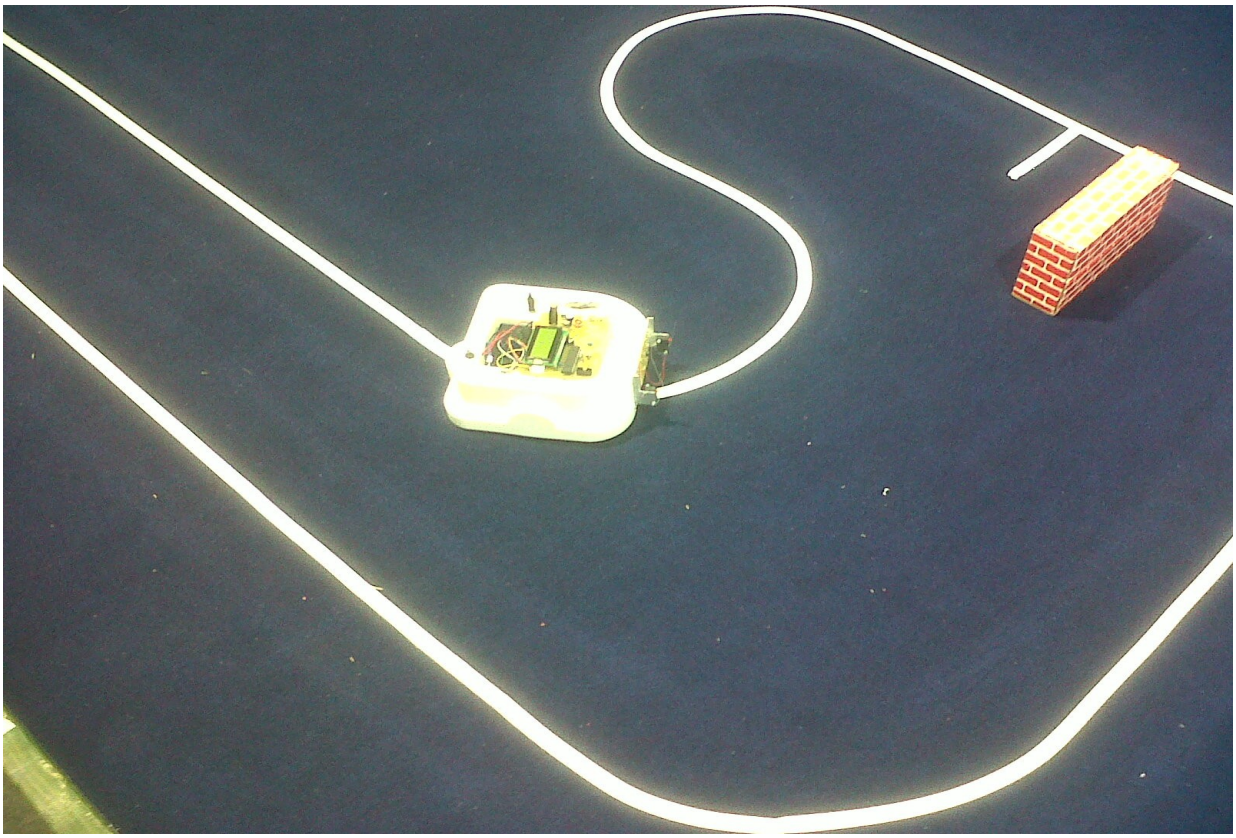
Projet d'étude et réalisation

Robot suiveur de ligne



LALLEMAND Jérémy
VIDAL Mathieu
2nd année
Groupe K3b

Professeur responsable :T LEQUEU

Projet d'étude et réalisation

Robot suiveur de ligne

Sommaire

Introduction.....5
Cahier des charges.....6
1. Présentation générale du projet.....7
1.1. Le robot.....7
1.2. Le terrain.....7
1.3. Planning de répartition des tâches effectuées.....8
2. La mécanique.....9
2.1. Le châssis.....9
2.2. Le bloc moteur.....9
1. L'électronique.....10
1.1. Présentation des composants utilisés.....10
1.2. Carte mère.....17
1.3. Cartes capteurs.....21
2. La programmation.....21
. Définitions des variables.....21
2.1. Les paramètres de configuration.....22
2.2. La reconnaissance de ligne.....23
2.3. La gestion des cas possibles.....25
3. Tests.....27
3.1. Carte mère.....27
3.2. Transfert et essais programme.....28
3.3. Carte capteurs.....29
4. Le challenge robotique.....30
4.1. Les premiers tests.....30
4.2. Le concours.....31
Conclusion.....32

Bibliographie.....33
Table des illustrations.....34
Annexes.....35

Introduction

Durant la seconde année de notre formation nous sommes amenés à réaliser un projet d'études et réalisation.

Les sujets proposés par l'enseignant sont variés et complets. Dans notre cas nous avons préféré choisir notre propre sujet .C'est à dire un robot suiveur de ligne de petite taille, comme son nom l'indique il doit pouvoir suivre un parcours en tout autonomie .

Nous avons comme contrainte un délai de réalisation de 9 semaines et un challenge robotique à préparer ,nous devons également faire attention à notre budget et respecter les consignes du cahier des charges.

Ce projet sera entièrement réalisé par nos soins de la partie électronique à la programmation en allant jusqu'à la conception.

Pour la partie mécanique nous sommes contrains par le cahier des charges et devons respecter les dimensions ainsi qu'utiliser le châssis réglementé par le challenge.

Cahier des charges

Robot Suiveur de lignes: Challenge Robotique de Vierzon

Contraintes liées au robot:

✗ dimensions:
- ➢ Largeur: 30cm
- ➢ Longueur: 40cm
- ➢ Hauteur: 50cm

✗ Autonomie durant tout le parcours
✗ Kit imposé (Support métallique en U, moteurs, roues, engrenages, batterie)
✗ Arrêt d'urgence obligatoire
✗ Prise jack pour le démarrage

Consigne à respecter durant le parcours:

✗ Suivre la ligne de couleur blanche et de 19mm de largeur sur un fond bleu (moquette)
✗ Respect des différentes consignes tout au long du parcours :

- ➢ Priorités à droite (à l'aide de barre blanche placées 45cm à droite avant la priorité)
- ➢ Raccourci (à l'aide de barres blanches placées 30cm à gauche avant le raccourci)
- ➢ Stop figure (à l'aide de deux barres blanches placées à droite et espacées de 20mm l'une de l'autre)
- ➢ Perçage des différents ballons placés tout au long du parcourt

✗ A l'arrivée il faut faire tomber la première poutrelle en bois et que le robot s'arrête avant la deuxième, elle deux situées a 8cm du sol

1 Présentation générale du projet

Dans ce chapitre nous expliquerons d'une manière simple le fonctionnement du robot . (voir en annexe les schémas fonctionnels et fast de notre projet)

1.1 Le robot

Pour vous donner une idée de la forme de notre robot nous avons pris plusieurs photos (visible notamment en première page).

Ce robot doit respecter les dimensions données dans le cahier des charges soit une hauteur maximale de 50 cm une largeur maximale de 30 cm et une longueur maximal de 40 cm. Il doit également être autonome durant tous le parcours, pour cela il sera alimenté par des batteries rechargeables.

1.2 Le terrain

Le robot se déplacera sur une moquette bleu et devra suivre une ligne blanche d'une largeur de 19 mm. Il pourra ensuite effectuer plusieurs figures et respecter les consignes indiquées dans le cahier des charges pour marquer des points et ainsi se qualifier pour les tours suivants. Cette piste sera étendue sur environ 10m.

1.3 Planning de répartition des tâches effectuées

VIDAL Mathieu

LALLEMAND Jérémy

	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
Essais théorique		X		X					X								
		O	O	O						O							
Orcad			X		X												
					O						O						
Typon					X						O		O				
					O	O											
Test						X							X		X	X	
						O							O				
Maintenance						X							X	X	X	X	
									O	O							
Soudure					X												
							O	O					O				
Essais pratiques						X								X	X	X	
									O				O				
Programmation										X	X						
										O	O		O				
Oral																	X
																	O

Planning prévisionnel	X
Planning Réel	O
Coupe robotique	

2 La mécanique

La partie mécanique est une des parties qui nous a pris le moins de temps puisque le châssis ainsi que la mécanique(moteur,roue,batterie),nous ont été fournis par les organisateurs du challenge.

2.1 Le châssis

Le châssis est en plastique de couleur blanche, les roues sont également en plastique et positionnées au centre du robot afin d'équilibrer celui-ci, on nous a fourni des pions en plastiques positionnés à l'avant ainsi qu'à l'arrière.

2.1.1 Le haut du robot

C'est sur ce support que viendra la carte mère et l'interrupteur électrique général qui ont été ajouté plus tard. Pour pouvoir commander l'alimentation du robot on y ajoutera également l'alimentation (batterie).

2.1.2 Le dessous du robot

Cette partie est celle qui supportera la plupart du poids du robot, on y fixera le bloc moteur, ainsi que la carte capteurs.

Pour fixer la carte capteurs nous avons utilisé des petites équerres que nous avons dû recouper pour être le plus prêt possible du sol.

2.2 Le bloc moteur

Les moteurs fonctionnent en 5 volts au maximum de leur puissance, après quelques tests nous avons remarqué que les moteurs avaient besoin de 1 A en charge pour fonctionner. Donc pour avoir une vitesse max nous appliquons une tension de 5V et pour arrêter les moteurs une tension de 0V.

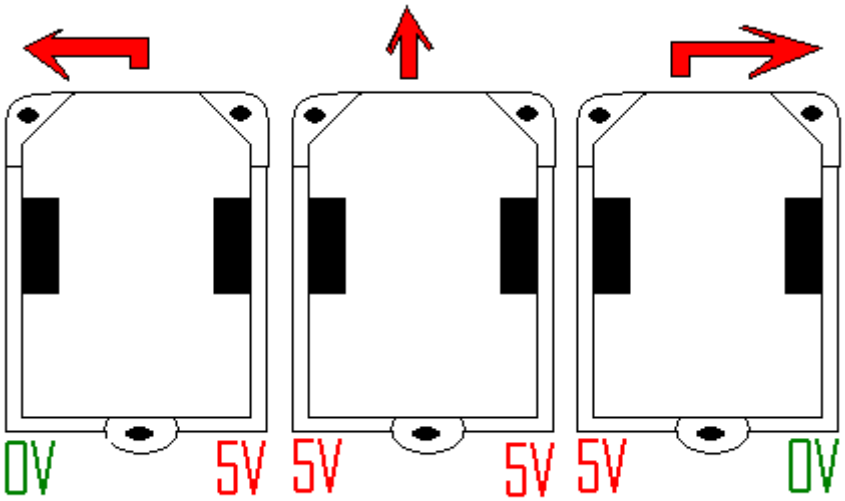
Principales caractéristiques :

- Tension d'alimentation nominale: 5 V
- Courant d'alimentation maximal : 1 A

5 Volts → 100 %

0 Volts → 0 %

Illustration 1: Directions possibles en fonctions des tensions moteurs



Nous aurons c'est 3 cas possibles aller tout droit tourner à gauche ou tourner à droite Nous ferons en sorte que le robot aille le plus vite possible en ligne droite, il faudra donc alimenter les 2 moteurs au maximum (5 Volts), et lorsque l'on tournera, l'un des 2 moteurs restera constamment au maximum et nous réglerons la vitesse de l'autre moteur afin de faire varier la vitesse en fonction du degrés du virage.

1 L'électronique

Probablement la partie où nous avons passé le plus de temps, c'est également celle où les recherches ont été les plus longues ainsi que les tests préalables, et c'est aussi malheureusement la partie qui nous a causé le plus de soucis et d'erreurs.

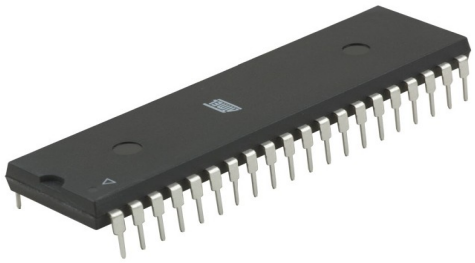
1.1 Présentation des composants utilisés

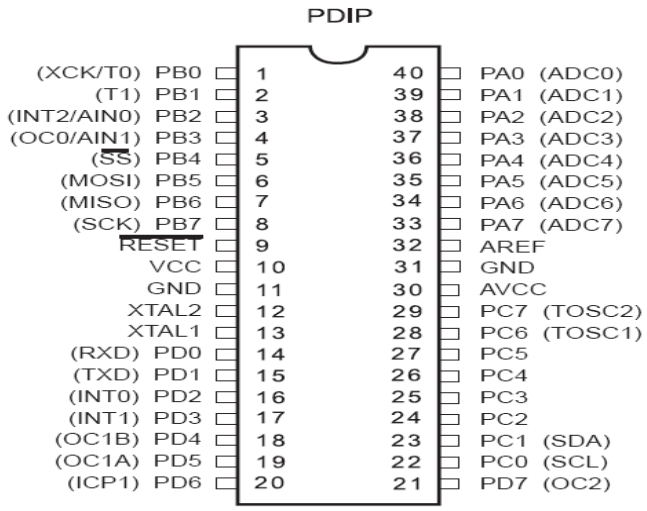
Pour la réalisation du projet, de nombreux composants ont été indispensables, tel que le LM2575 pour l'alimentation,ainsi que le LM2574 pour notre seconde alimentation nécessaire pour les moteurs. le L298 nous permet de gérer les moteurs, bien sur l'ATmega8535 qui gère les diverses informations du montage, les capteurs qui détectent au sol la direction du parcours, et enfin diverses résistances, capacités, inductances et autres diodes utiles au bon fonctionnement du montage.

1.1.1 ATmega 8535

L'ATmega8535 est le cœur de notre carte ainsi que ce qui permet au projet d'être viable et fonctionnel en effet il sert à recevoir, traiter et renvoyer les informations nécessaires au bon fonctionnement du robot.

Illustration 2: L'ATmega 8535





Principales caractéristiques :

- Vitesse de cadence jusqu'à 16 MHz
- Tension d'alimentation de 4.5 V à 5.5 V
- Courant d'alimentation : 10mA
- 40 PINs
- 32 entrées/sorties
- 8 canaux analogiques
- Convertisseur CAN 8 bits (0 → 1023 = 1024 possibilités)
- 2 sorties PWM (MLI)
- Courant maximal par entrées/sorties : 40 mA
- 8 Ko de mémoire programme
- EEPROM de 512 Octets
- SRAM de 512 Octets
- 3 timers (2 de 8bits, 1 de 16 bits)
- Programmation langage C

1.1.1 Capteur CNY70

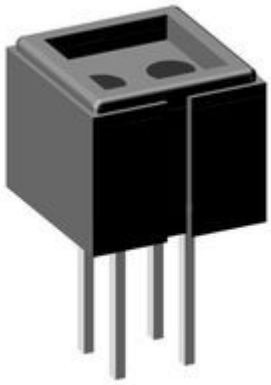


Illustration 3: Capteur CNY70

Le CNY70 est un capteur infrarouge sensible à la couleur, en effet il est au niveau logique haut(5V) lorsqu'il perçoit une ligne blanche et au niveau bas pour du noir. Le niveau de tension varie selon la surface et la couleur utilisée. En effet dans notre cas le fond bleu ne nous donnera pas une tension tout à fait égale a 0V, c'est pourquoi nous avons utilisé des tensions de seuil. Le capteur doit être, en outre, placé très proche du sol pour capter de manière optimal.

Le principe de fonctionnement d'un CNY70 est une émission et réception infrarouge. En voici le schéma:

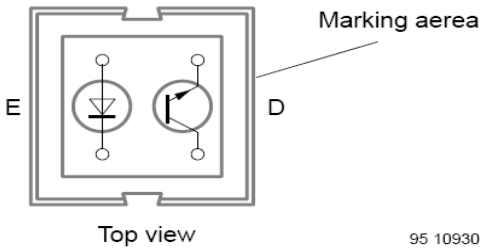


Illustration 4: Schéma de principe du CNY70

Principales caractéristiques :

Émetteur

- Tension d'alimentation nominale : 1.25 V

- Tension d'alimentation maximale : 1,6 V
- Courant d'alimentation : 50 mA

Détecteur

- Tension de sortie nominale : 5 V
- Tension de sortie maximale : 5 V
- Courant de sortie nominal : 100µA
- Courant de sortie maximum (théorique) : 1mA

1.1.1 Le L298

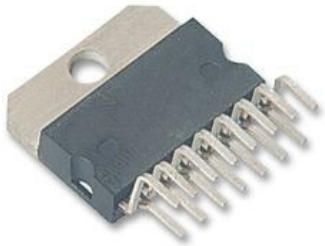


Illustration 5: Hacheur L298

L'ATmega8535 délivre une tension de 40mA par sortie (voir précédemment[1]), d'où le choix du l298 pour pouvoir alimenter les 2 moteurs cependant celui-ci consomme beaucoup de courant (1A en fonctionnement normal et 2A en charge maximal) Ce qui nous oblige à rajouter une alimentation qui fournit suffisamment de courant, mais permet d'empêcher les perturbations sur le micro-contrôleur, en effet ce dernier n'influera que sur la tension.

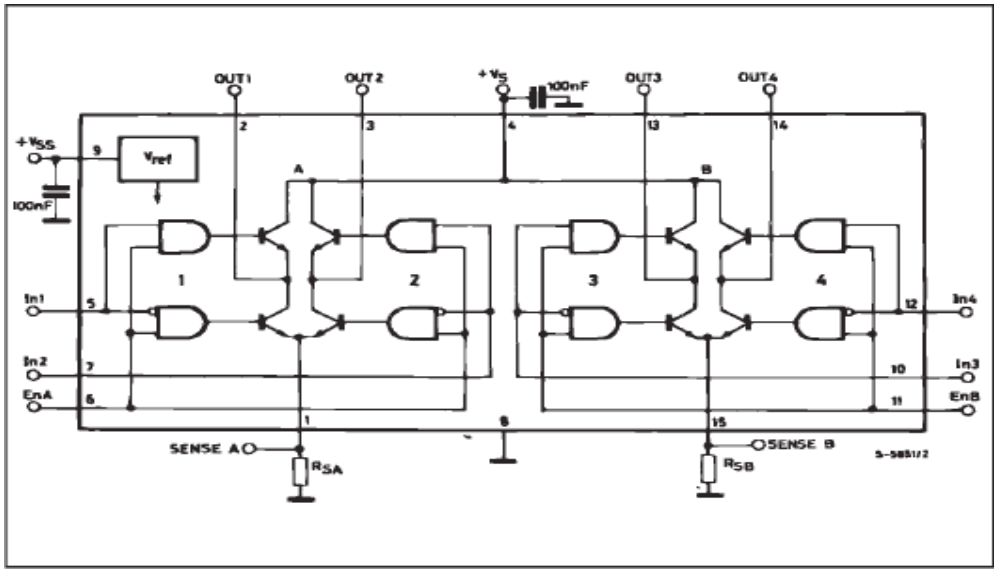


Illustration 6: Schéma interne du L298

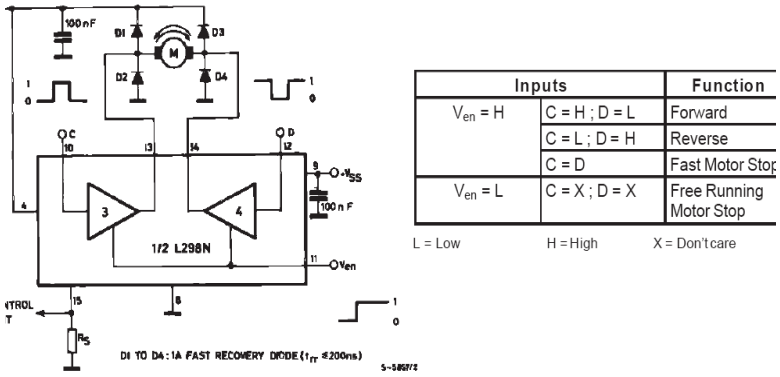
1 Voir chapitre 3.1.1 - ATmega 8535

Nous avons choisi de brancher le moteur droit du robot sur les pattes 13 et 14 du composant et le moteur gauche sur les pattes 2 et 3. Rsa et Rsb sont des résistances de limitation de courant dans les moteurs en effet pour éviter les sur-intensités il était nécessaire de les calculer:

On veut limiter le courant à 1A et on a 5V en sortie des pattes 1 et 15 donc

→ R=U/I=5/1=5 ohms

Nous avons ensuite choisi de mettre des diodes entre les moteurs et le composant en lui même pour éviter les court-circuits lors d'un changement de sens de rotation du moteur. En effet lors de cette séquence, les transistors se retrouvent, pendant un court temps, en commutation simultanément cela peut donc créer un court-circuit qui serait fatal pour le composant. Schéma du câblage des diodes sur un moteur:



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Illustration 7: schéma de câblage des diodes

Principales caractéristiques :

- Tension d'alimentation du circuit logique : 5 V
- Tension d'alimentation du circuit puissance : 0 V 48 V (nous serons à 5 V)
- Tension minimale de sensibilité MLI : 2.3 V
- Courant maximal du circuit puissance : 3A

1.1.1 Régulateur LM2574/5 Volts

L'utilité du régulateur choisi ici est de pouvoir transformer une tension d'entrée allant jusqu'à 45V et délivrant 5V en sortie, tout en délivrant un courant de 500mA. Dans notre cas, nous devons transformer la tension de la batterie qui est de 12V. Ce sera donc l'alimentation de la quasi-totalité de notre carte, à savoir, l'ATmega8535, nos capteurs ainsi que le L298. Grâce à lui nous obtenons donc une tension stabilisée de 5V quelque soit le courant demandé en sortie. L'avantage non négligeable du composant, est qu'il est de petite taille donc facilite le routage, et surtout, il ne chauffe pas.

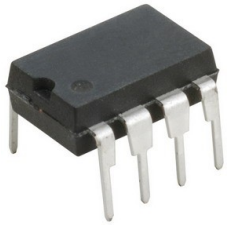


Illustration 8: LM2574

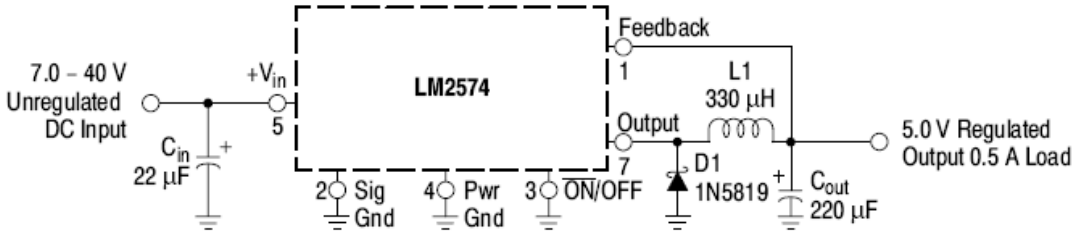


Illustration 9: Schéma de branchement du LM2574

On observe toutefois un inconvénient de taille, c'est qu'il doit être entouré de pas mal de composants, donc le routage s'en trouve plus difficile étant donné les contraintes de taille de la carte.

Principales caractéristiques :

- Tension maximale d'entrée : 45 V
- Tension de sortie : 5 V
- Courant maximal de sortie : 500 mA

1.1.1 Régulateur LM2575/5 Volts

Le choix de ce régulateur s'est fait après les tests tardifs effectués concernant la consommation des moteurs en charge. En effet ce test nous a permis de nous rendre compte que notre première alimentation ne suffirait pas aux moteurs. Ceux-ci consommes donc 1 A en charge (c'est à dire le robot roulant normalement avec le poids de la batterie et des divers cartes et moteurs). Il a donc fallu rajouter le LM2575 capable de fournir un courant supérieur au LM2574, ce premier nous permet d'avoir le nécessaire, à savoir, 1A.

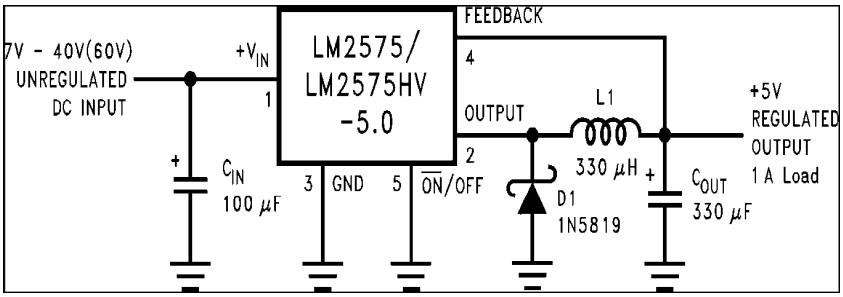


Illustration 10: Schéma de branchement du LM2575

Principales caractéristiques :

- Tension maximale d'entrée : 18 V
- Tension de sortie : 5 V
- Courant maximal de sortie : 1 A

1.1.1 L'écran LCD

Nous avons décidé d'ajouter un écran LCD au projet car, bien qu'inutile au bon fonctionnement du robot, il nous permettra de faciliter le débeugage de la carte en cas de problèmes. En effet nous pourrons y afficher différentes valeurs de tensions sur les endroits souhaités.

Illustration 11: L'écran LCD

1.1.2 Batterie 9,6 Volts – 1,2 Ah

Pour rendre le robot autonome il faut bien entendu une batterie, celle de notre projet nous a été imposée par les organisateurs du concours de Vierzon.

Illustration 12: Batterie 12 Volts – 1,2 Ah

Cette batterie a l'avantage de fournir la puissance nécessaire aux besoins de notre carte et d'avoir l'autonomie adéquate pour la durée du parcours, cependant elle est encombrante et pesante. Nous disposons par ailleurs d'un logement pour l'y mettre et le poids est gênant car cela fait consommer plus, mais la batterie suffit à fournir ce qu'il faut.

1.1.3 Charger la batterie

Pour charger la batterie le principe est simple, nous ne disposons pas de chargeur pour le concours il va donc falloir ce contenter d'une alimentation réglée sur 13,5V et limiter le courant à 1A.

1.2 Carte mère

Ce chapitre est dédié à notre carte principal, celle qui fait fonctionner la totalité du robot puisqu'elle comprend l'Atmega8535, cœur de notre projet, mais nous ne traiterons que de la partie électronique, la programmation sera abordée plus tard.

Nous avons dû reprendre la conception de cette carte à cause de différents problèmes rencontrés au court de notre première réalisation. En effet le test tardif de la consommation des moteurs est à l'origine de cette nouvelle carte. Nous avons en outre ajouté une entrée supplémentaire sur l'ATmega8535 pour un capteur supplémentaire destiné à faciliter la réalisation d'un obstacle au niveau du parcours. Enfin nous avions négligé les résistances de limitation de courant sur le L298, cette deuxième carte nous a donc permis de régler ces différents problèmes. La réalisation des 2 cartes s'est faite sur les logiciels Capture puis Orcad spécialement destinés à la réalisation de schéma de carte(Capture) et au routage de celle-ci (Orcad).

En annexe n°1 et 2 , vous pourrez trouver le routage ainsi que le schéma complet de

la carte dont nous allons aborder les différentes parties ci-après.

1.2.1 L'ATmega8535

Schéma Orcad – partie ATmega8535

Le schéma de base de l'ATmega8535 est composé d'un connecteur de programmation avec des prises normalisées pour éviter toutes inversions, c'est une prise de typeHE10 - 10 broches. Nous avons placé une LED en série avec une résistance qui crée une chute de tension aux bornes de la LED, c'est elle qui fera office de témoin de programmation lors des transferts de programmes depuis l'ordinateur. Le calcul de la résistance est le suivant:

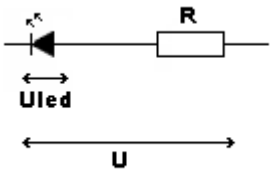


Illustration 13: Calcul de résistance

R = (U - Uled) / I avec I = 10 mA, courant absorbé par la LED

Uled = 2 Volts, chute de tension aux bornes de le LED

U = 5 Volts

Application Numérique =

R= (5-2) / 10.10-3

R = 300 Ω

Ce calcul reste valable pour la LED placée en sortie de notre LM2575.

Placé entre les pattes 12 et 13, le quartz de 16 Mhz[2] complété de 2 condensateurs de 22 pF de valeurs imposées par le quartz lui même, sert à cadencer l'ATmega8535.

Pour récupérer les signaux envoyés depuis les capteurs il est indispensable de les relier au micro-contrôleur, elles seront fixées de la patte ADC0 à la patte ADC5 (6 capteurs) via une prise Molex 8 broches (JP3) les signaux obtenus devront être convertis en valeurs numériques à l'aide des CAN (convertisseur analogique-numérique) présents dans le CPLD. Cette conversion sera faite ultérieurement.

Pour effectuer la conversion précédemment évoquée il sera donc nécessaire de fixer AVCC ainsi que AREF à 5V.

2 Fréquence maximale supportée par l'ATmega

Les commandes des deux moteurs du robot seront de nature MLI[3], l'ATmega ne comportant que 2 broches pour ses sorties MLI, nous seront obligé d'utiliser ces fonctions sans les changer de broche, ce qui ne devrait pas nous poser de problèmes.

Nous avons placé un condensateur de découplage sur l'alimentation de notre Atmega8535, celui-ci sert a protéger ce dernier contre les pics d'intensités.

1.2.2 Le hacheur L298

Schéma Orcad – partie Hacheur

Ce hacheur étant commandé en signaux MLI, nous avons raccordé directement les sorties MLI du micro-contrôleur sur les bornes 5 et 12 du hacheur.

La réalisation du câblage s'est effectuée à partir de la documentation constructeur prise sur internet. Ainsi nous pouvons voir 2 condensateurs de découplage C8 et C9 placés entre le +5V et la masse, nous disposons de 2 condensateurs car un sert à l'alimentation de composant et l'autre pour les moteurs. Visible également, les 2 résistances R5 et R6 servent à limiter le courant débité par le L298. En voici le calcul:

Nous avons 5V sur les pattes 1 et 15 et nous avons décidé de limiter le courant à 1A d'où → R=U/I → R=5/1=5ohms

Nous avons décidé de mettre des diodes de protection au niveau des moteurs, celle-ci seront routées sur la carte directement pour empêcher les court-circuits en cas d'inversion de sens des moteurs. Nous avons fait le choix de les souder sur des pastilles avec leur pattes reliées à la masse et au +5V en l'air pour faciliter le routage étant donné que 4 sont reliées ensemble à la masse et 4 autres au +5V donc nous avons les 4 pattes en l'air reliées à une 5eme reliée directement à la masse ou au +5V.

Comme pour les capteurs les moteurs sont reliés à une prise Molex située sur la carte de manière à pouvoir les débrancher plus facilement en cas de besoin, cette prise est constituée de 4 broches (JP4 moteurs)

Nous avons également choisi de torsader les fils reliant les moteurs à la carte afin d'éviter au maximum toutes perturbations.

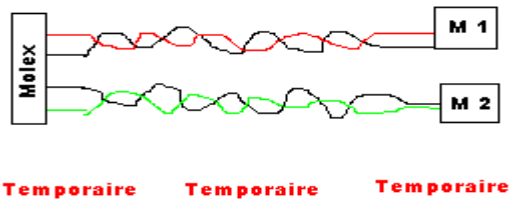


Illustration 14: Torsader les fils

3 Modulation à Largeur d'Impultions (PWM en anglais pour Pulses Width Modulation)

1.2.3 Le régulateur LM2574

Schéma Orcad – LM2574

C'est la partie de notre montage qui alimente toute la carte en 5V. Nous avons donc apporté un grand soin quant à la réalisation du routage des divers composants l'entourant pour assurer son bon fonctionnement. Le schéma nous est donné par la documentation constructeur (voir en annexe). Grâce à celui-ci nous pouvons réaliser sans souci notre alimentation 5V stabilisée.

Directement en entrée de la carte, en sortie, donc, du connecteur reliant la batterie à la carte on observe une diode en série avec celle-ci qui permet de protéger l'ensemble de la carte en cas d'un mauvais branchement (inversion de sens au niveau du branchement du connecteur de la batterie).

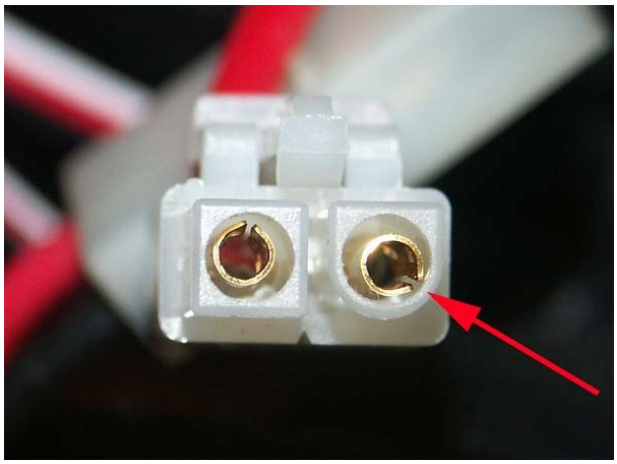


Illustration 15: Connecteur 2 broches

Comme on peut le voir ci-dessus le connecteur dispose d'un détrompeur qui rend impossible l'erreur d'inversion de sens, cependant lors du câblage nous pouvons par inattention faire l'erreur d'inverser la masse avec le +12V.

1.2.4 Le régulateur LM2575

Schéma Orcad – LM2575

Comme dit précédemment le régulateur LM2575 alimente les moteurs du fait qu'il fournit assez de courant. Le schéma de câblage n'est pas beaucoup plus simple que pour le LM2574 étant donné que nous avons ajouté un potentiomètre de réglage de la tension en sortie de celui-ci, cela va nous permettre ci besoin est, d'augmenter la tension en sortie, cela aura pour effet de faire avancer plus vite le robot.

1.3 Cartes capteurs

Notre carte de détection de la ligne blanche au sol (carte de capteur) est relativement simple, il s'agit en fait d'un seul circuit que nous avons recopier 6 fois puisque nous avons décidé de mettre 6 capteurs sur la carte. Nous avons également réalisé une autre petite carte ne comportant qu'un seul capteur pour la fin du parcours où il faudra détecter une barre de bois située à 8 cm du sol et la faire tomber sans atteindre la 2ème barre située 20cm plus loin. Pour cela il sera nécessaire d'avoir un seuil de détection inférieur car la barre n'est pas de couleur blanche (voir programmation).

Schéma d'un capteur

Nous envoyons sur notre carte une tension de 5V continu, or nous avons une tension de 1,25V en entrée des capteurs il sera donc utile de rajouter une résistance de 68ohms pour obtenir la tension utile.

Typon WinCircuit

L'utilisation de WinCircuit nous a grandement simplifié la tâche et nous a fait gagner un temps précieux, en effet le routage étant relativement simple et répétitif, il a été possible de l'effectuer rapidement.

2 La programmation

La programmation a été pour nous la partie la plus compliquée. Programmer sur un ATmega8535 est nouveau pour nous, il a donc fallu lire les documentations et solliciter nos camarades.

Pour la programmation de notre ATmega8535, nous utiliserons le logiciel CodeVision AVR, c'est un logiciel servant à créer des programmes en langage C, à les compiler, puis à envoyer le programme vers le micro-contrôleur rentré en paramètres.

Définitions des variables

Nous aurons besoin lors de l'élaboration de notre programme de plusieurs variables, de deux types différents, Entier et Tableau de caractères. Ces variables seront déclarées en variables globales de façon à êtres modifiables par n'importe quelle fonction du programme.

```
int buf, i, val2;

unsigned char tampon[20], val[8];
```

Langage C

Nous utiliserons :

-Un tableau de caractères à 8 cases 'Val' (une case représentera alors l'état d'un capteur)

-Un tableau de caractères à 20 cases 'tampon' (utile pour l'affichage sur le lcd)

-deux variables entières 'l', 'buf' (nous verrons leurs utilitées dans un prochain chapitre)

-Une variable entière 'Val2' (concaténation des 7 valeurs de capteurs)

2.1 Les paramètres de configuration

La configuration de l'ATmega8535 est simple il faut uniquement rentrer le type d'ATmega, sa vitesse puis Code vision génère automatiquement le code de configuration de celui-ci.

```
//  Définition du port A en entrée
PORTA=0x00;
DDRA=0x00;

//  Définition du port D en sortie
PORTD=0x00;
DDRD=0x30;

//  Définition du timer 1 pour les MLI
TCCR1A=0xA1;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Définition du comparateur
ACSR=0x80;
SFIOR=0x00;

// Initialisation du convertisseur Analogique Numérique
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
SFIOR&=0xEF;
```

Langage C

Dans ce chapitre nous verrons toutes les étapes de programmation avec la reconnaissance de la ligne, l'affichage sur le LCD puis enfin le suivi de la ligne par les moteurs voir programme en annexe.

L'emplacement des différents composants sur l'ATmega8535 sont résumés sur le Datasheet de celui-ci avec les différentes pattes du composant.

2.2 La reconnaissance de ligne

Pour reconnaître la ligne,nous utiliserons la tension délivrée par le capteur CNY70, soit 5V lorsque celui ci capte la ligne blanche puis « 0V » lorsque qu'il ne capte rien .

Pour cela nous allons utiliser la conversion Analogique Numérique de l'ATmega8535.

Cette configuration est directement inscrite dans le logiciel CodeVisionAVR .

```
//  lecture du résultat de la conversion Analogique Numérique
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
delay_us(10);
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
```

Langage C

Cette fonction sera exécutée autant de fois qu'il y a de capteurs, donc 7 fois au total. Le résultat de la conversion est envoyée dans une variable val2 pour ensuite être comparé et traité.

Lorsque cette fonction est appelée, elle doit savoir quelle entrée analogique est concernée, c'est pourquoi le programme principal devra le lui préciser.

```
for (i=0; i<8; i++)
{

buf = read_adc(i);

if(buf>250)
     val[i]=1;
else
 val[i]=0;
}
val2= val[0]+val[1]*2+val[2]*4+val[3]*8+val[4]*16+val[5]*32+val[6]*64;
```

Langage C

Dans la fonction MAIN[4], on appelle la fonction[5], celle-ci nous renvoie donc une valeur comprise entre 0 et 1023. On entre cette valeur dans une variable « buf » puis on la compare à un seuil, nous avons choisi 250 afin d'avoir une meilleure précision des capteurs. Il est vrai que le tapis bleu renvoie une tension au capteur négligeable mais présente.

4 Fonction principale
5 read_adc()

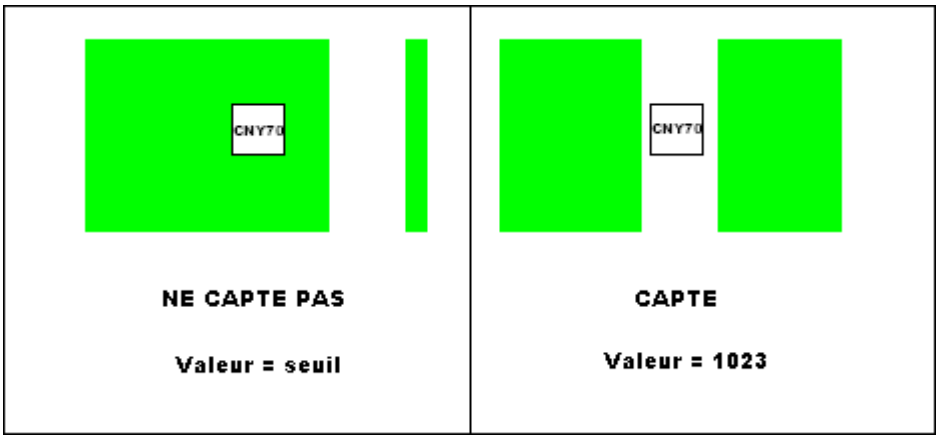


Illustration 16: Modélisation du seuil

Comme le montre ce schéma, le seuil sera la valeur lorsque le capteur voit le tapis.

Si la valeur retournée est supérieure au seuil, la valeur 1 est entrée dans le tableau de caractères 'Val' à la case correspondante au numéro de capteur.

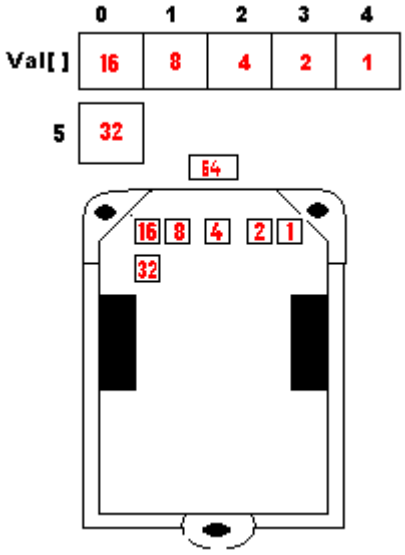


Illustration 17: Correspondance des capteurs avec le tableau

Le programme questionne ainsi les 7 capteurs par le biais de la fonction read_adc(), Lorsque le programme a obtenu les informations des 7 capteurs, le tableau 'Val' contient une suite de 1 et de 0, le programme principal exécute le calcul suivant.

Val2 = Val[0] + Val[1]*2 + Val[2]*4 + Val[3]*8 + Val[4]*16+Val[5]*32+Val[6]*64

On obtient donc une image décimale de l'état des capteurs.

2.3 La gestion des cas possibles

16 cas de positions de ligne s'offrent à nous, à chaque cas correspond une valeur décimale de l'état des capteurs. Cette partie du programme saura différencier ces cas, et effectuera les tâches nécessaires au bon déplacement du robot.

La conception du programme s'est réellement déroulée lors du concours robotique, nous avons pû faire des tests sur piste ainsi que des réglages.

Nous avons neuf cas possibles pour le suivi de ligne nous avons pris en compte le degré des virages, nous avons également huit cas pour les différentes actions que le robot devra effectuer lors de son parcours, voir cahier des charges, les priorités à droite, les intersections en font partie.

État des capteurs							**Valeur Hexadécimale**
Capteurs							
5	6	0	1	2	3	4	
0	**0**	**0**	**0**	**1**	**0**	**0**	0x04
0	**0**	**0**	**0**	**1**	**1**	**0**	0x06
0	**0**	**0**	**0**	**0**	**1**	**0**	0x02
0	**0**	**0**	**0**	**0**	**1**	**1**	0x03
0	**0**	**0**	**0**	**0**	**0**	**1**	0x01
0	**0**	**0**	**1**	**1**	**0**	**0**	0x0C
0	**0**	**0**	**1**	**0**	**1**	**0**	0x08
0	**0**	**1**	**0**	**0**	**0**	**0**	0x10
0	**0**	**1**	**1**	**0**	**0**	**0**	0x18
0	**0**	**0**	**0**	**0**	**0**	**0**	0x00
0	**0**	**0**	**0**	**1**	**1**	**1**	0x07
0	**1**	**0**	**0**	**0**	**0**	**1**	0x21
0	**1**	**1**	**1**	**1**	**1**	**1**	0x3F
1	**0**	**0**	**0**	**1**	**0**	**0**	0x44
1	**0**	**0**	**0**	**0**	**0**	**0**	0x40
0	**0**	**1**	**1**	**1**	**0**	**0**	0x1C

```
switch (val2)
        {
            case 0x04:          //ligne droite
                OCR1AL=190;
                OCR1BL=190;
                MDAR=0;
                MGAR=0;
            break;
            case 0x06:          // tres petit virage droite
                OCR1AL=90;
                OCR1BL=190;
                MDAR=0;
                MGAR=0;
            break;
            case 0x02:   //virage droite
                OCR1AL= 0;
                OCR1BL=190;
                MDAR=0;
                MGAR=0;
            break;
            case 0x03:   // gros virage droite
                OCR1AL=90;
                OCR1BL=190;
                MDAR=0;
                MGAR=0;
            break;
            case 0x01:   //tres gros virage droite
                OCR1AL=90;
                OCR1BL=190;
                MDAR=0;
                MGAR=0;
            break;
```

Les deux variables OCR1AL et OCR1BL correspondent à la valeur du alpha du MLI de chacun des deux moteurs. OCR1AL pour le moteur "droite", OCR1BL pour le moteur "gauche". Cette valeur varie de 0 à 255, 0 correspondant à 0 Volts, 255 correspondant à 5 Volts[6]. Les valeurs données sont susceptibles d'êtres modifiées, notamment lors des tests finals du robot.

3 Tests

Afin de régler les problèmes de manière plus efficace et rapide nous avons effectué une batterie de tests après chaque réalisation, cela nous permet de régler les problèmes petit à petit.

3.1 Carte mère

La phase de test à proprement parler s'est déroulée en plusieurs étapes, tout d'abord nous avons fait des tests préalables avant d'implanter tout composant programmable, nous avons donc testé si nous avions bien les bonnes tensions d'alimentation aux bons endroits (sur l'ATmega8535, le hacheur, et autres composants du circuit), ainsi que quelques tests de continuité aux endroits susceptibles de poser problèmes. Ensuite nous avons ajouté les composants préalablement mis de coté et effectué de nouveaux tests, bien sur étant donné que nous avons du réaliser 2 cartes, ces tests ont été effectué sur les 2 cartes.

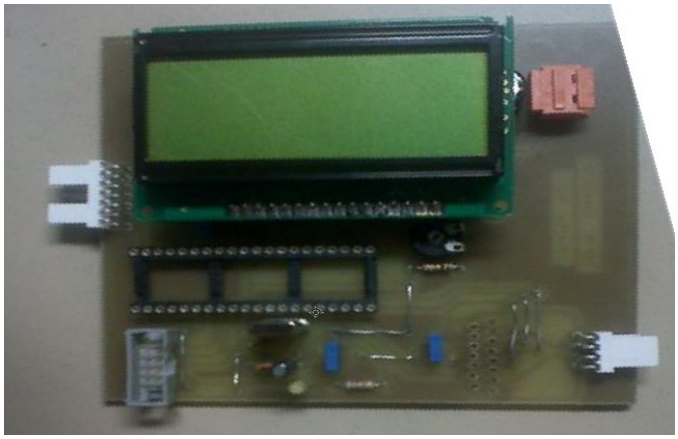


Illustration 18: test sans composant programmable

Lors de ces premiers tests nous n'avons détecté aucun problème de court-circuit ou de discontinuité de pistes, les LEDs censés s'allumer se sont allumées, nous avons donc pu passer à des tests plus poussés, avec l'ajout de l'ATmega8535 et du LM2574.

6 Voir chapitre 2.2 – Le bloc moteur

3.2 Transfert et essais programme

Pour les essais de programmation, nous avons transféré nos programmes à l'aide de la carte fourni par l'IUT qui permet de relier notre Atmega8535 au PC. Étant donné nos maigres connaissances en programmation sur le composant ici employé nous avons préféré programmer par petits morceaux avant le test total. Les premiers test étaient un simple affichage sur l'écran LCD:

Après quelques tests similaires nous avons pu passer à plus sérieux, notamment à l'envoi de tensions sur le L298, c'est là où nous avons rencontré nos premiers véritables problèmes, nous avons d'abord pensé à un manque de courant sur les moteurs d'où la réalisation de notre 2ème carte, cependant même après la réalisation de celle-ci le problème subsistait, les moteurs ne tournaient pas. Après multiples vérifications de tout ce qui nous semblais susceptible de ne pas fonctionner nous nous sommes rendu compte qu'une erreur sur Orcad était à l'origine du dysfonctionnement, en effet les pattes du L298 ont toutes été inversées dans la bibliothèque du logiciel, ainsi nous n'y avons pas pensé du tout et c'est par une vérification chanceuse que nous avons découvert cela, nous avons décidé de souder le composant de l'autre coté de la carte pour un gain de temps (nous ne pouvions pas recommencer une carte). Un second problème est toute fois survenu après cela lorsque nous avons tenté de reprogrammer l'ATmega8535, nous avions par inadvertance soudé le connecteur de programmation à l'envers, ce qui a probablement causé un court-circuit sur le CPLD et l'a ainsi « grillé ». Après remplacement de ce dernier le programme ainsi que le reste de la carte ont très bien fonctionné.

Illustration 19: premier test de programmation

3.3 Carte capteurs

Notre carte capteur a, elle, fonctionné du premier coup, les tests effectués sur celle-ci ont été concluant, cependant quand nous l'avons relié à la carte principale un problème est survenu, 2 de nos capteurs semblait ne plus fonctionner, nous avons effectué de nouveaux tests en la débranchant et tout fonctionnait normalement, nous en avons donc conclu que cela venait de l'ATmega8535 et en effet c'était le problème évoqué précédemment concernant le composant défectueux. Après remplacement la carte fonctionnait parfaitement bien mis à part quelques réglages de seuil depuis la programmation.

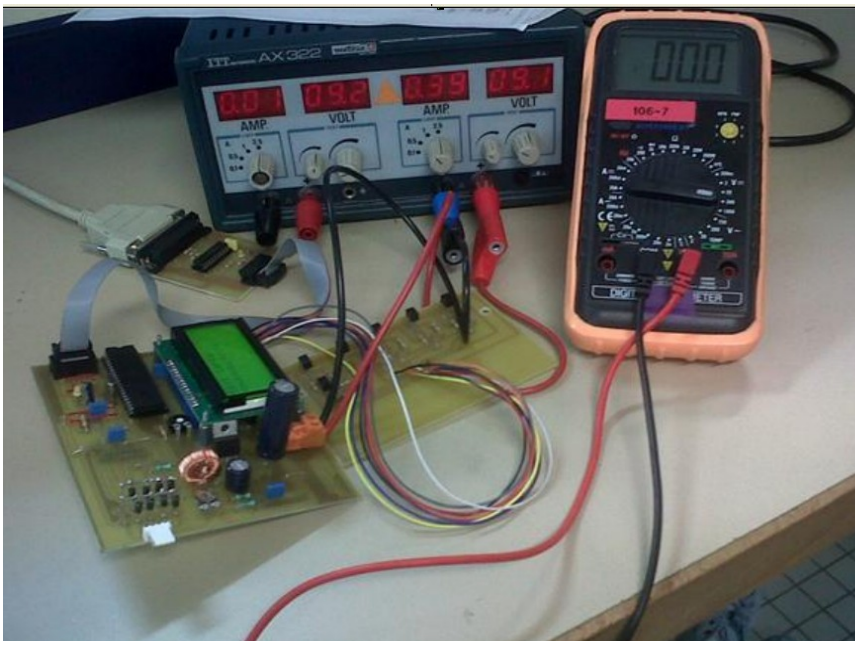


Illustration 20: test avec la carte capteur

4 Le challenge robotique

Nous sommes arrivés le jeudi 27 Mai vers 15h00, nous avons déchargé le matériel et nous nous sommes mis au travail avec notamment les réglages des vitesses des moteurs avec la programmation.

4.1 Les premiers tests.

Les premiers tests sur piste ont été catastrophiques, le robot ne répondait à rien et ne suivait pas la ligne blanche. Nous avons dû travaillé jusqu'à 22h00 pour trouver la panne et pouvoir se qualifier pour le premier tour du concours.

Après fonctionnement total du robot (c'est à dire que l'ATmega8535 commande les moteurs et qu'il récupère bien la valeur des capteurs), différents problèmes se sont posés à nous, en effet le robot fonctionnait et les moteurs réagissaient en fonction des capteurs ce qui était le plus important, cependant la difficulté s'est posée quand à suivre la ligne sans la perdre, notre robot avançant trop vite il lui était difficile de garder la ligne en captage. Nous avons donc dû baisser progressivement la vitesse du robot pour qu'il garde la ligne sous le champ de rayon des capteurs. Puis après quelques réglages et inversion des moteurs notre robot parvient enfin à prendre une trajectoire et a pu être homologué pour participer au concours! C'était le plus important pour nous car même si nous ne visons pas la gagne du concours, participer reste une belle réussite pour nous car ça n'était pas gagné d'avance. Voici une photo du robot en phase de test:

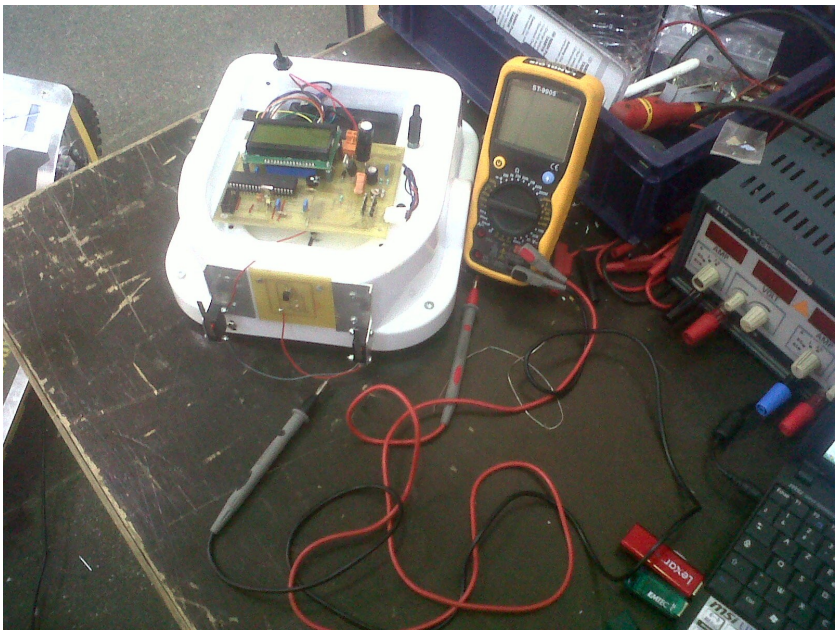


Illustration 21: Robot en phase de test

4.2 Le concours

Le concours fût intéressant et nous a confirmé que l'électronique est capricieuse. Effectivement le robot fonctionnait sur piste d 'essai mais pourtant nous sommes pas parvenu à terminer le parcours. Nous nous sommes fait éliminer dès les premiers tours.

Ce concours a été intéressant et enrichissant avec de nombreuses rencontres et une solidarité entre concurrents et entre IUT impressionnante.

Ce concours nous a notamment appris à travailler seuls sans aide des enseignants et aussi travailler en équipe avec son binôme. Nous avons également dû gérer notre emploi du temps, le concours était bien orchestré avec des horaires précises à suivre.

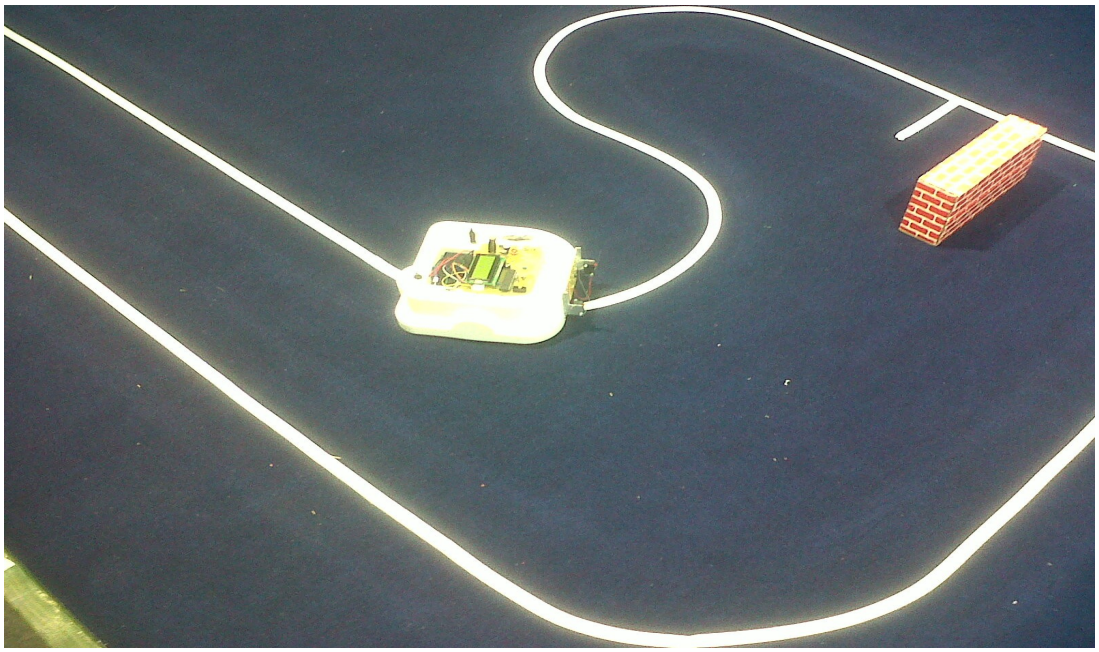


Illustration 22: Robot sur piste concours Robotique

Conclusion

A travers ce projet nous avons pu réinvestir la majorité des notions apprises ces 2 années d'IUT au sein du GEII. En effet ce projet demandait des notions à la fois d'électronique, d'informatique, et de mécanique, nous avons donc dû faire preuve d'une certaine agilité en terme de compétences.

Le projet nous a également permis de consolider des bases dans certains domaines ainsi que de découvrir de nouvelles façons de faire ou l'utilisation de nouveaux outils tel que Orcad que nous n'avions pas encore eu à exploiter.

Nous avons été confronté à un planning très serré en regard de la charge de travail, mais cela nous a permis de faire face à la pression par moment et à des situations proches de celles que nous pourrions rencontrer dans le milieu professionnel.

Nous avons quand même quelques regrets concernant le temps justement car nous avions divers obstacles tout au long du parcours et certains ne pourront pas être effectués faute de temps.

Malgré cela le fait de mener un projet de bout en bout en partant de zéro a été une expérience enrichissante et motivante pour nous.

Bibliographie

Datasheets:

-cny70

-l298

-ATmega 8335

-LM2574

-LM2575

Documentation sur le site internet de thierry lequeu (http://www.thierry-lequeu.fr/)

http://benoit.stef.free.fr/robotika_fichiers/Rapport%20de%20projet%20robot.pdf
http://www.yopdf.com/shema-de-cny70-pdf20.html#a6

Table des illustrations

Index des illustrations

Illustration 1: Directions possibles en fonctions des tensions moteurs....10
Illustration 2: L'ATmega 8535....11
Illustration 3: Capteur CNY70....12
Illustration 4: Schéma de principe du CNY70....12
Illustration 5: Hacheur L298....13
Illustration 6: Schéma interne du L298....13
Illustration 7: schéma de câblage des diodes....14
Illustration 8: LM2574....15
Illustration 9: Schéma de branchement du LM2574....15
Illustration 10: Schéma de branchement du LM2575....16
Illustration 11: L'écran LCD....16
Illustration 12: Batterie 12 Volts – 1,2 Ah....17
Illustration 13: Calcul de résistance....18
Illustration 14: Torsader les fils....19
Illustration 15: Connecteur 2 broches....20
Illustration 16: Modélisation du seuil....24
Illustration 17: Correspondance des capteurs avec le tableau....24
Illustration 18: test sans composant programmable....27
Illustration 19: premier test de programmation....28
Illustration 20: test avec la carte capteur....29
Illustration 21: Robot en phase de test 30
Illustration 22: Robot sur piste concours Robotique....31

Annexes

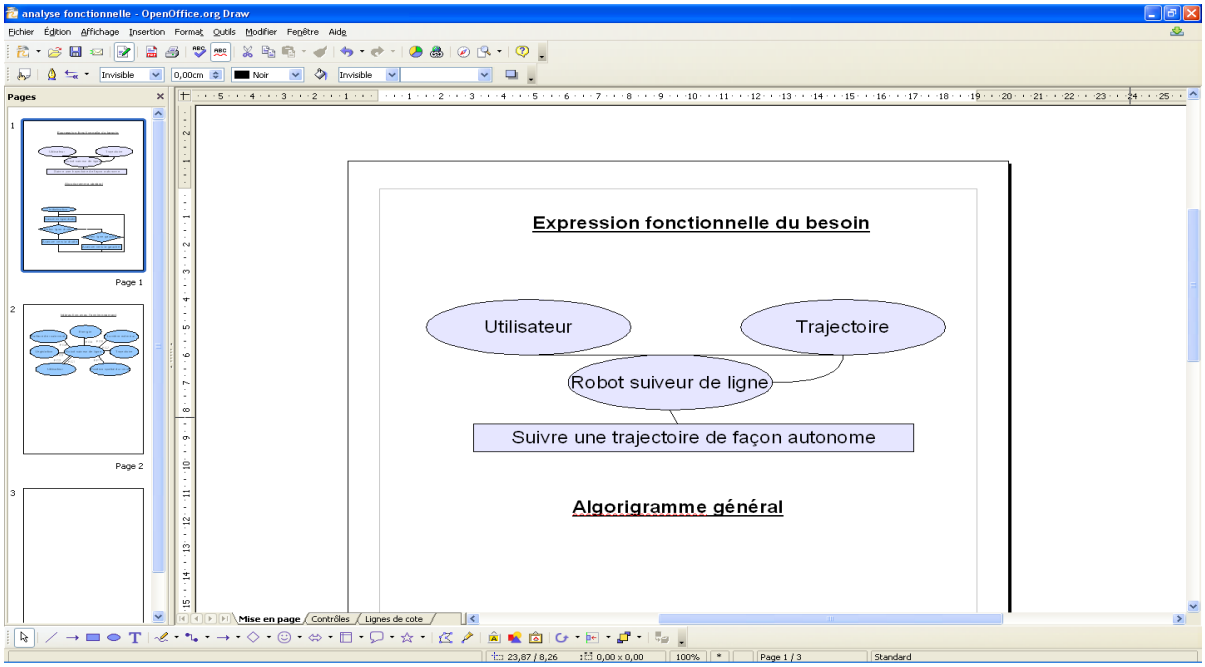
-Schémas fonctionnels et fast

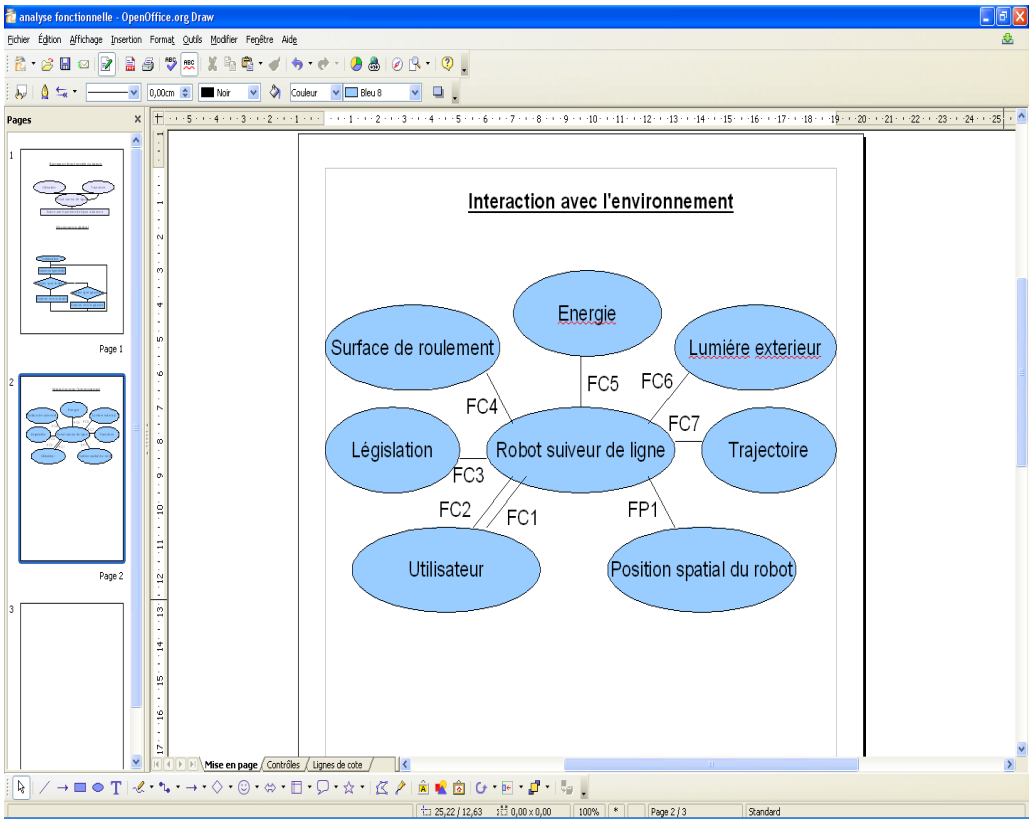
-Schéma sous capture des différentes cartes du robot

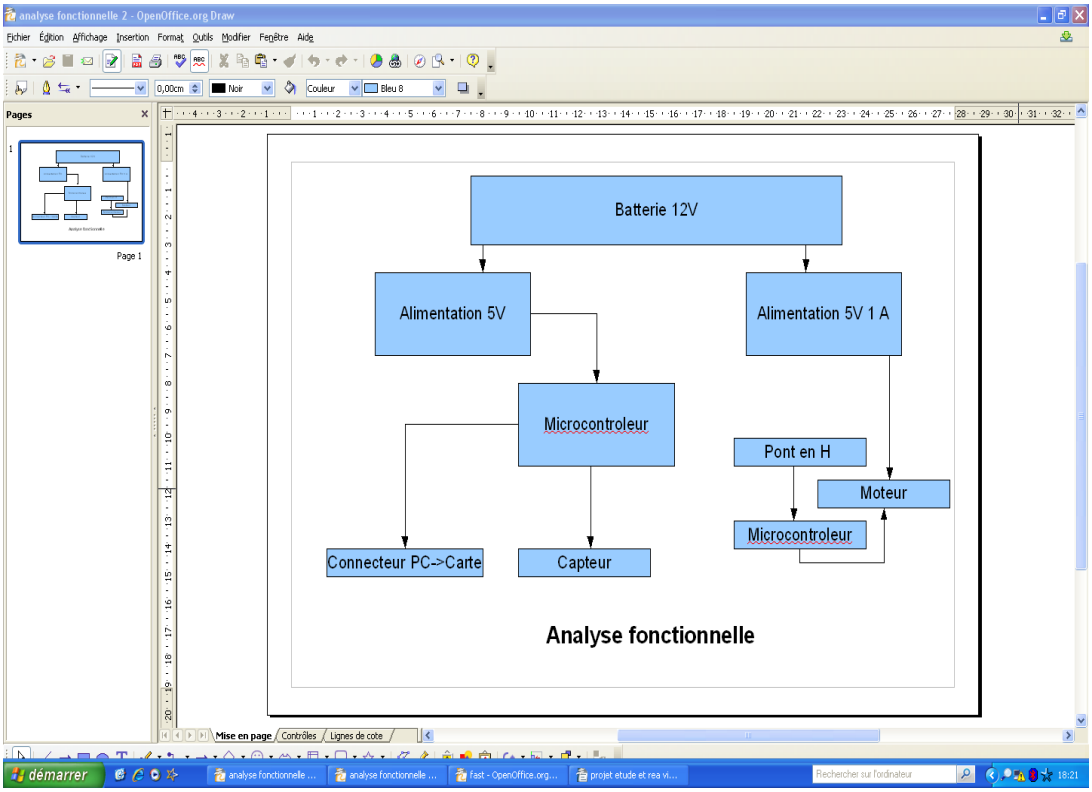
-Schéma Orcad des différentes cartes du robot

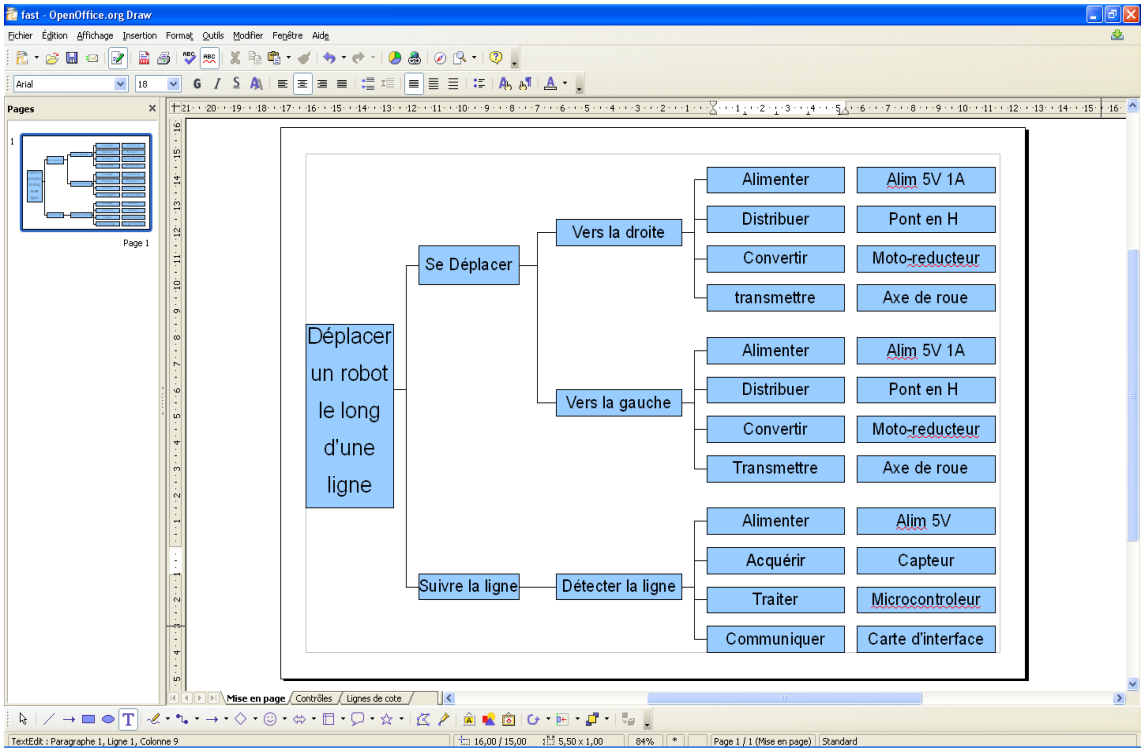
-Programme de l'ATmega8535 sur CodeVision

Annexe 1


analyse fonctionnelle - OpenOffice.org Draw
Expression fonctionnelle du besoin
Utilisateur
Trajectoire
Robot suiveur de ligne
Suivre une trajectoire de façon autonome
Algorigramme général


analyse fonctionnelle - OpenOffice.org Draw
Interaction avec l'environnement
Energie
Surface de roulement
Lumière exterieur
FC5
FC6
FC4
FC7
Législation
Robot suiveur de ligne
Trajectoire
FC3
FC2
FC1
FP1
Utilisateur
Position spatial du robot


Batterie 12V
Alimentation 5V
Alimentation 5V 1 A
Microcontroleur
Pont en H
Moteur
Microcontroleur
Connecteur PC->Carte
Capteur
Analyse fonctionnelle


Déplacer un robot le long d'une ligne
Se Déplacer
Vers la droite
Vers la gauche
Suivre la ligne
Détecter la ligne
Alimenter
Distribuer
Convertir
transmettre
Alim 5V 1A
Pont en H
Moto-reducteur
Axe de roue
Alimenter
Distribuer
Convertir
Transmettre
Alim 5V 1A
Pont en H
Moto-reducteur
Axe de roue
Alimenter
Acquérir
Traiter
Communiquer
Alim 5V
Capteur
Microcontroleur
Carte d'interface

Fast

ANNEXE 2

Schéma Capture de la carte principale

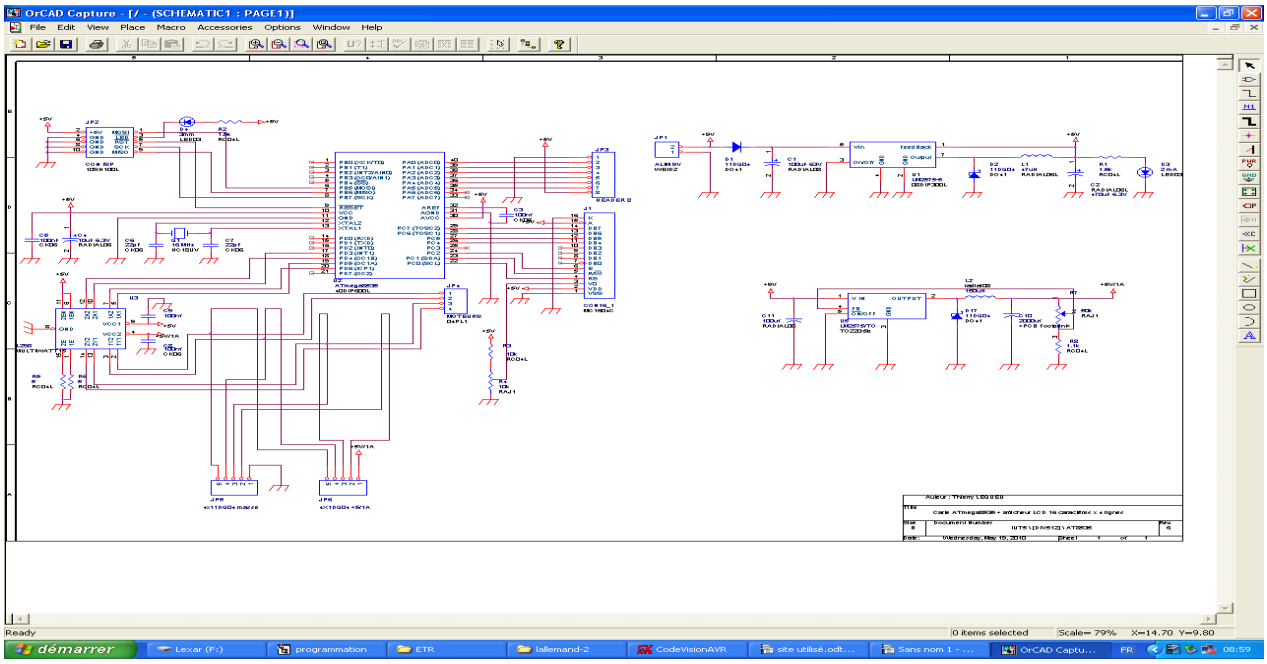
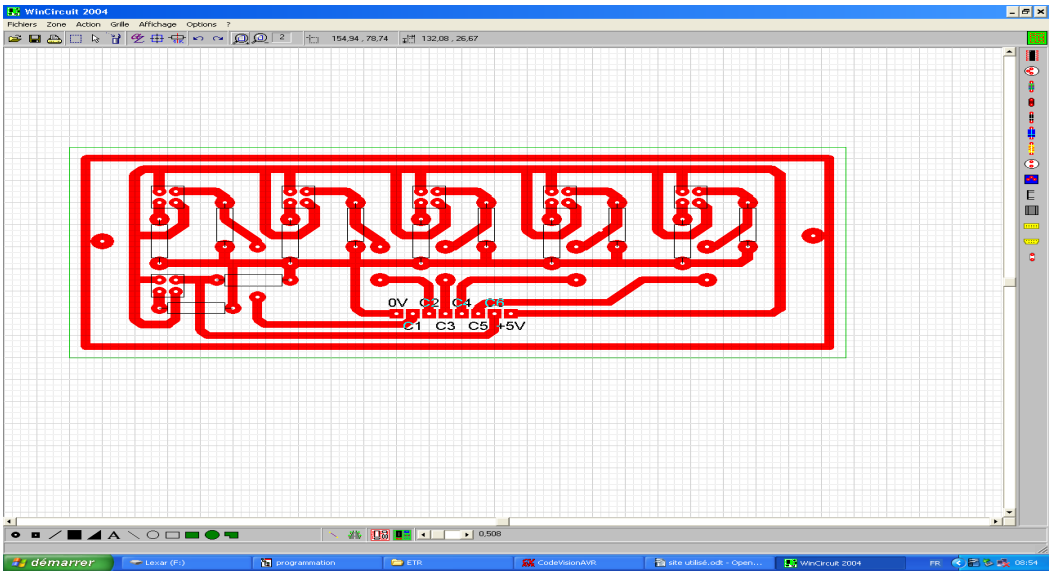
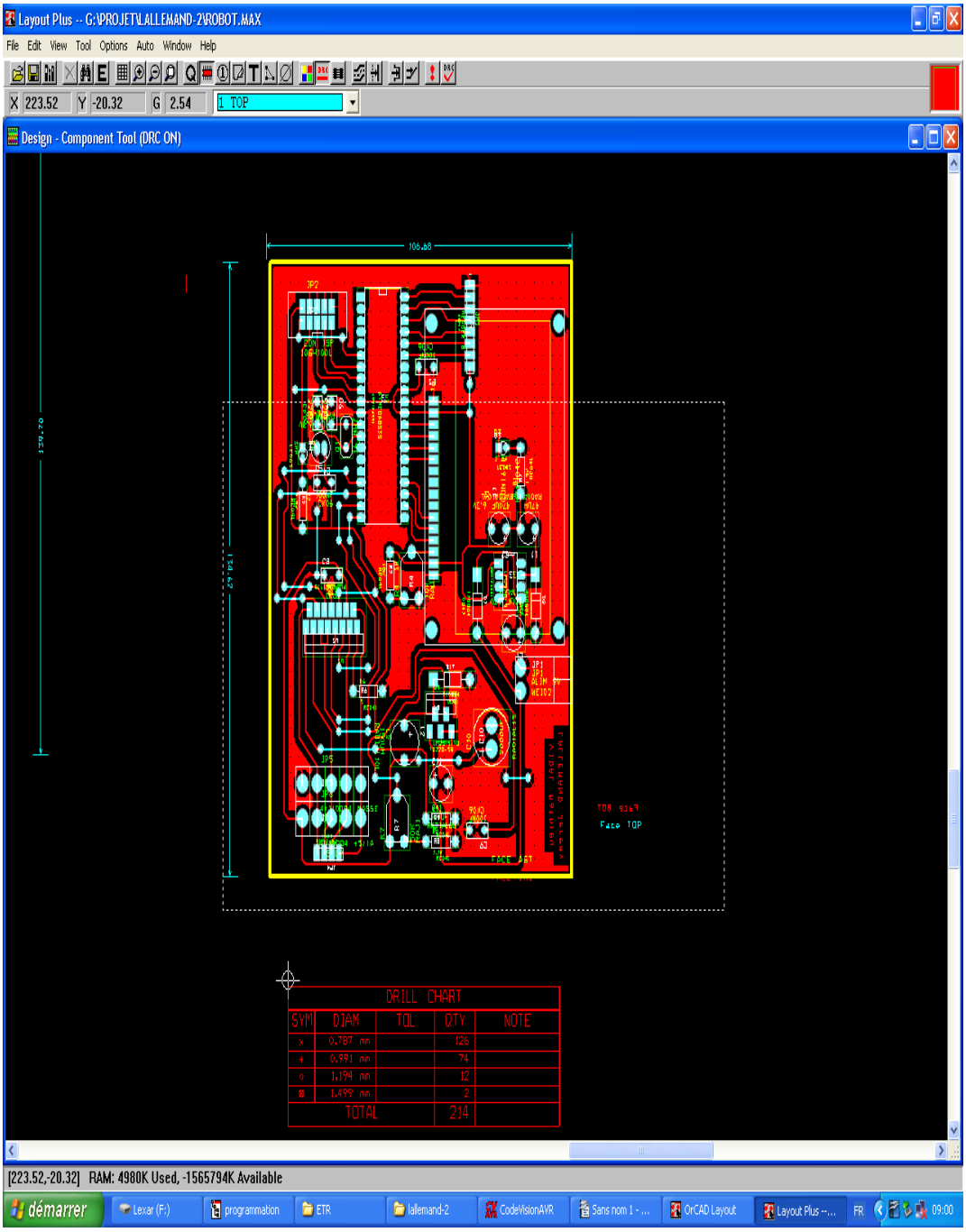


Schéma sous Wincircuit de la carte capteur



ANNEXE 3

Schéma Orcad de la carte principale



Annexe 4

Nomenclature et prix des composants

N°	Quantité	Référence	Désignation	Empreinte	Prix unitaire
1	1	C1	100uF 63V	RADIAL08	1,01 €
2	1	C2	470uF 6.3V	RADIAL06L	0,45 €
3	3	C3,C5,C8	100nf 6.3V	CK06	0,138 €
4	1	C4	10uF 6.3V	RADIAL06	0,45 €
5	2	C6,C7	22pF	CK06	0,000 €
6	1	C9	100nf 6.3V		0,14 €
7	1	C10	2000uf		7,29 €
8	1	C11	100uf	RADIAL08	1,01 €
9	3	D1,D2,D17	11DQ04	DO41	0,104 €
10	1	D3	2 mA	LED03	0,191 €
11	1	D4	3mm	LED03	1,21 €
12	1	JP1	ALIM 9V	WEID2	2,16€+4,59€
13	1	JP2	CON ISP	10SH100L	0,000 €
14	1	JP3	HEADER 8		0,000 €
15	1	JP4	MOTEURS	04PL1	0,000 €
16	1	JP5	4x11DQ04 masse		0,000 €
17	1	JP6	4X1DQ04 +5/1A		0,000 €
18	1	J1	CON16_1	MC1604C	0,000 €
19	1	L1	47uH	RADIAL06L	1,293 €
20	1	L2	150uH	radial08	0,98 €
21	1	Q1	16 MHz	HC18UV	0,000 €
22	2	R1,R2	1.5k	RC04L	0,01 €
23	1	R3	10k	RC04L	0,01 €
24	1	R4	10k	RAJ1	0,03 €
25	2	R5,R6	5k	RC04L	0,01 €
26	1	R7	50k	RAJ1	1,90 €
27	1	R8	1.1k	RC04L	0,01 €

Soit un cout total de: 49,91€ ainsi que 400 euros de frais d'inscriptions pour le concours de Vierzon.

ANNEXE 5

Programme de l'ATmega8535 sur CodeVision

```
#include <mega8535.h>
  #asm
   .equ __lcd_port=0x15 ;PORTC
  #endasm
  #include <lcd.h>
  #include <delay.h>
  #include <stdio.h>
  #define MGAV PORTD.4
  #define MGAR PORTD.3
  #define MDAV PORTD.5
  #define MDAR PORTD.6
  #define ADC_VREF_TYPE 0x40
  unsigned int read_adc(unsigned char adc_input)  // Read the AD conversion result
  {
          ADMUX=adc_input|ADC_VREF_TYPE;
          ADCSRA|=0x40;    // Start the AD conversion
          while ((ADCSRA & 0x10)==0);   // Wait for the AD conversion to complete
          ADCSRA|=0x10;
          return ADCW;
  }
  int buf,i,val2; //déclaration des variable en global
  unsigned char tampon[40], val[8];
void main(void)
{
  // Declare your local variables here
  // Input/Output Ports initialization
  // Port A initialization
  // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
  // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

```
PORTA=0x00;
DDRA=0x00;
  // Port B initialization
  // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
  // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;
  // Port C initialization
  // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
  // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;
  // Port D initialization
  // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
  Func1=Out Func0=Out
  // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0x30;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
  // Timer/Counter 1 initialization
  // Clock source: System Clock
  // Clock value: Timer 1 Stopped
  // Mode: Normal top=FFFFh
  // OC1A output: Discon.
  // OC1B output: Discon.
  // Noise Canceler: Off
```

```
  // Input Capture on Falling Edge
TCCR1A=0xA1;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
  // Timer/Counter 2 initialization
  // Clock source: System Clock
  // Clock value: Timer 2 Stopped
  // Mode: Normal top=FFh
  // OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
  // External Interrupt(s) initialization
  // INT0: Off
  // INT1: Off
  // INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
  // Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
  // Analog Comparator initialization
  // Analog Comparator: Off
  // Analog Comparator Input Capture by Timer/Counter 1: Off
  // Analog Comparator Output: Off
```

```
ACSR=0x80;
SFIOR=0x00;
  // ADC initialization
  // ADC Clock frequency: 125,000 kHz
  // ADC Voltage Reference: AVCC pin
  // ADC High Speed Mode: Off
  // ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;
 // LCD module initialization
lcd_init(16);
lcd_clear();
while (1)
    {
        for (i=0; i<8; i++)
        {
                buf = read_adc(i);
                if(buf>250)
                        val[i]=1;
                else
                        val[i]=0;
        }
    val2= val[0]+val[1]*2+val[2]*4+val[3]*8+val[4]*16+val[5]*32+val[6]*64;
     lcd_gotoxy(0,0); // emplacement de l'ecriture sur le LCD
     sprintf(tampon, "%d", buf); // entre le buf dans tampon
     lcd_puts(tampon); // restitue le tampon sur le LCD
     lcd_gotoxy(0,1);
     sprintf(tampon, "%2d", val2);
     lcd_puts(tampon);
     lcd_gotoxy(0,2);
     sprintf(tampon, "%3d", OCR1AL);
```

```
        lcd_puts(tampon);
        lcd_gotoxy(0,3);
        sprintf(tampon, "%3d", OCR1BL);
        lcd_puts(tampon);
        lcd_gotoxy(7,0);
        lcd_putsf("mathieu"); // ecriture du mots « mathieu » sur le LCD
        lcd_gotoxy(7,1);
        lcd_putsf("jeremy");

    switch (val2)
        {
            case 0x04:      //ligne droite
            OCR1AL=190;
            OCR1BL=190;
            break;
            case 0x06:  // tres petit virage droite
            OCR1AL=90;
            OCR1BL=190;
            break;
            case 0x02:  //virage droite
            OCR1AL=90;
            OCR1BL=190;
            break;
            case 0x03:  // gros virage droite
            OCR1AL=90;
            OCR1BL=190;
            break;
            case 0x01:  //tres gros virage droite
            OCR1AL=90;
            OCR1BL=190;
            break;
```

```
case 0x0C:  //tres petit virage gauche
OCR1AL=190;
OCR1BL=90;
break;
case 0x08:  //virage gauche
OCR1AL=190;
OCR1BL=90;
break;
case 0x10:  //tres gros virage gauche
OCR1AL=190;
OCR1BL=90;
 break;
case 0x18:  // gros virage gauche
OCR1AL=190;
OCR1BL=90;
break;
case 0x00:  // hors piste
OCR1AL=190;
OCR1BL=190;
break;
case 0x07: //priorité a droite
OCR1AL=190;
OCR1BL=190;
break;
case 0x21:  //figure
OCR1AL=190;
OCR1BL=190;
break;
case 0x3F:  // intersection
OCR1AL=190;
OCR1BL=190;
break;
```

```
                    case 0x1C:  // raccourcis à gauche
                    delay_ms(10);
                    OCR1AL=40;
                    OCR1BL=150;
                    break;
                    case 0x44:
                    OCR1AL=0;  //arret final
                    OCR1BL=0;
                    delay_ms(10000);
                    break;
                    case 0x40:
                    OCR1AL=0;
                    OCR1BL=0;
                    delay_ms(10000);
                    break;
                }
        }
}
```