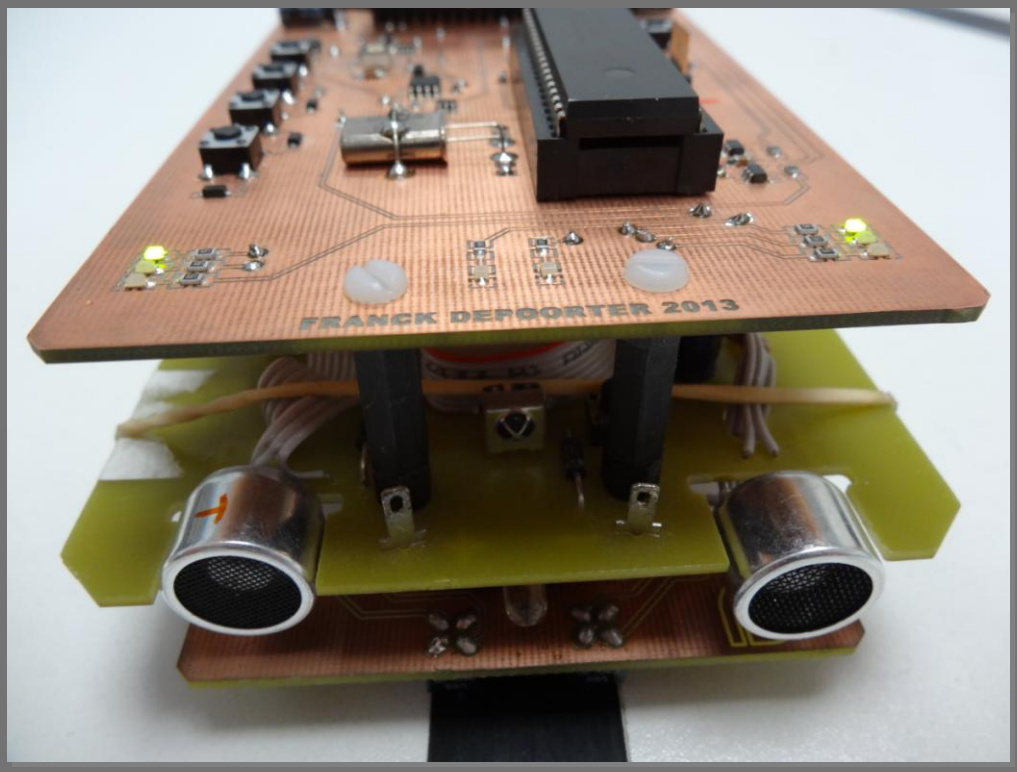
ETD
2013
2014

Robot suiveur de ligne

Franck Depoorter

Électronicien de Tests et Développement



Franck Depoorter
Electronicien de Tests et Développement

Promo ETD 2013-2014

FONDATION SANTÉ DES ÉTUDIANTS DE FRANCE

Sommaire

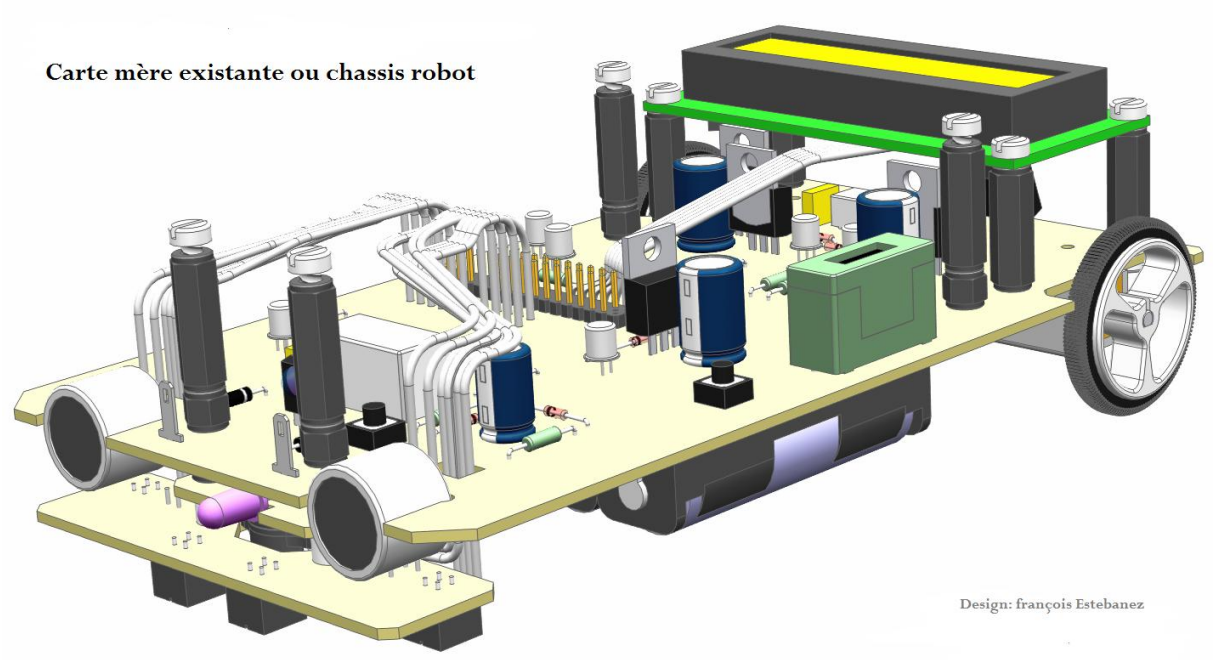
Introduction

Ce dossier se propose de documenter les différentes phases d'élaboration d'une carte électronique contrôlant l'évolution d'un robot suiveur de ligne.

Situation

Cette carte doit prendre place au dessus d'une carte ''mère'' existante, le châssis du robot, où l'on trouve les éléments significatifs suivants :

- 1 Emetteur + 1 récepteur ultrasonore (capteur de proximité).
- 4 émetteurs - récepteurs infrarouge de type CNY70 (capteurs de ligne).
- 1 LED émettrice IR + 1 photorécepteur infrarouge PNA4601M (non utilisé ici).
- 2 (Drivers Pushpull 2N2222- 2N2907 +MOSFETS BUZ11) de commande des moteurs D et G.
- 2 motoréducteurs Pololu avec leurs roues, bille ' folle ' avant.
- Relais miniatures HFD4/005 d'inversion de sens des moteurs.
- Une électronique de commande 'on/off' du robot par activation d'un relai bistable par 2 boutons poussoir. Possibilité d'alimenter le robot par commutation sur alimentation externe sur 'off'.
- Une électronique d'extinction du robot contrôlable de l'extérieur via broche A4 du connecteur 2* 17 (non utilisé ici).
- Bloc de batterie d'accumulateurs et régulateurs d'alimentation (+5v régulés moteurs séparés du +5v régulé carte fille et afficheur (low dropout regulators).
- Afficheur LCD 2 * 16 caractères à entrée série.
- Un connecteur 2 * 17 broches et sa nappe d'interconnexion entre carte mère et carte fille.



Cahier des charges de la carte de gestion

Elle se doit de mettre en œuvre :

Les fonctions nécessaires au fonctionnement des capteurs de ligne infrarouge existants :

- Alimentation et imposition du courant traversant les LEDS d'émission.
- Alimentation et imposition du courant traversant les phototransistors de réception.
- Conditionnement et bufferisation au niveau logique des signaux issus des 4 capteurs de ligne.
- Bufferisation analogique des signaux issus des 2 capteurs de ligne intérieurs.

Les fonctions nécessaires à la mise en œuvre du capteur de proximité :

- Génération hardware d'un signal carré constant à la fréquence de 40.6 kHz.
- Mise en œuvre d'un système de commande ayant pour objectif, l'émission 'on/off' sur la nappe de salves 40.6 kHz commandées par le microcontrôleur.
- Amplifier le signal issu du récepteur à ultrason et le conditionner en un signal tout ou rien qui puisse être reconnu par une broche du microcontrôleur configurée en entrée numérique.

Les fonctions nécessaires à l'interprétation des signaux issus des capteurs et à la commande par programmation des organes du robot:

- Implantation d'un microcontrôleur de type PIC16F877 fonctionnant à 20MHz.
- Mise en œuvre d'une fonction de Reset pour le microcontrôleur (manuelle par bouton poussoir et soft par interface extérieure sur base Maxx232).
- Mise en place d'une connectique de programmation (+5v et 0v pour l'alimentation de l'interface Maxx232, liaison de programmation série, Reset en soft).

Les fonctions nécessaires à la génération de commandes par clavier :

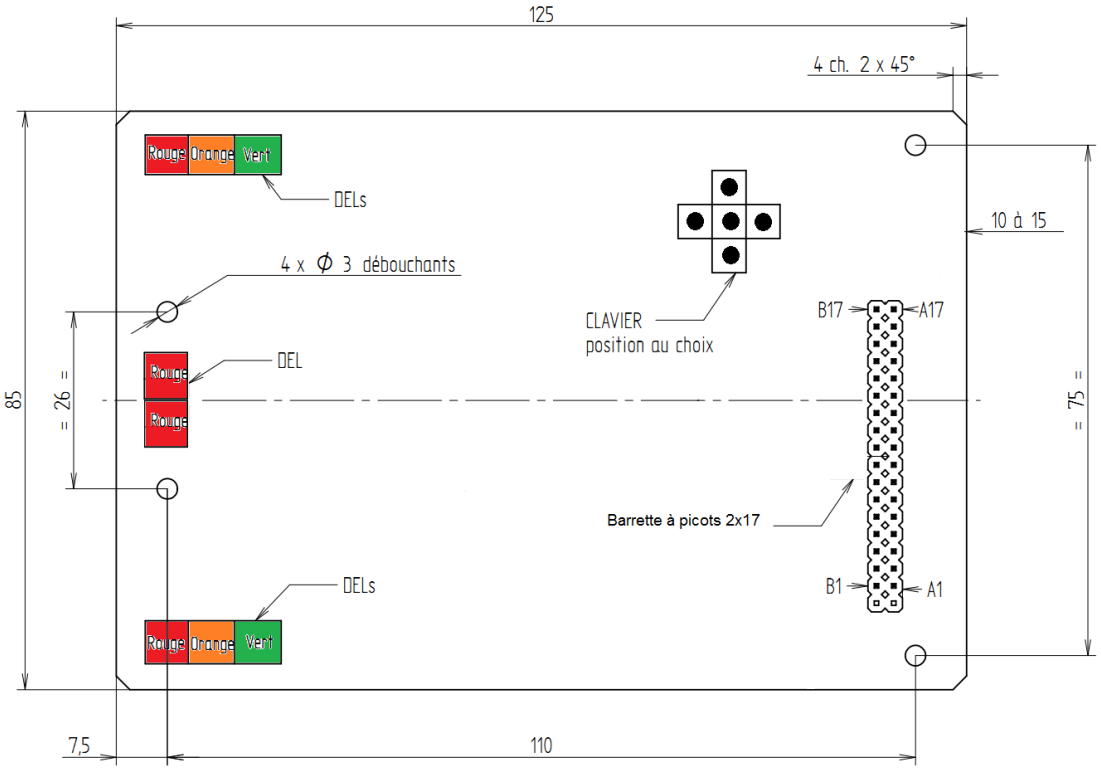
- Système de génération d'ordres par boutons poussoirs, à partir de la tension présente sur les accumulateurs.
- Bufferisation du niveau analogique d'ordre avant son entrée sur le microcontrôleur.

Les fonctions de signalisation lumineuse commandée par le microcontrôleur :

- Deux LEDS centrales rouge indépendantes (vers l'avant du robot).
- Une rampe de LEDS droite (rouge, jaune, vert) à commande indépendante pour chaque LED. (vers l'avant du robot).
- Une rampe de LEDS gauche (rouge, jaune, vert) à commande indépendante pour chaque LED.(vers l'avant du robot).

Implantation physique

- Dimensions de la carte (125mm x 85mm) et positions des 4 trous de fixations imposés



- Barrette à picots 2 x 17 broches en position arrière imposée (vers l'afficheur LCD).
- Quartz HC49 implanté à plat avec bride.

Technologies des composants :

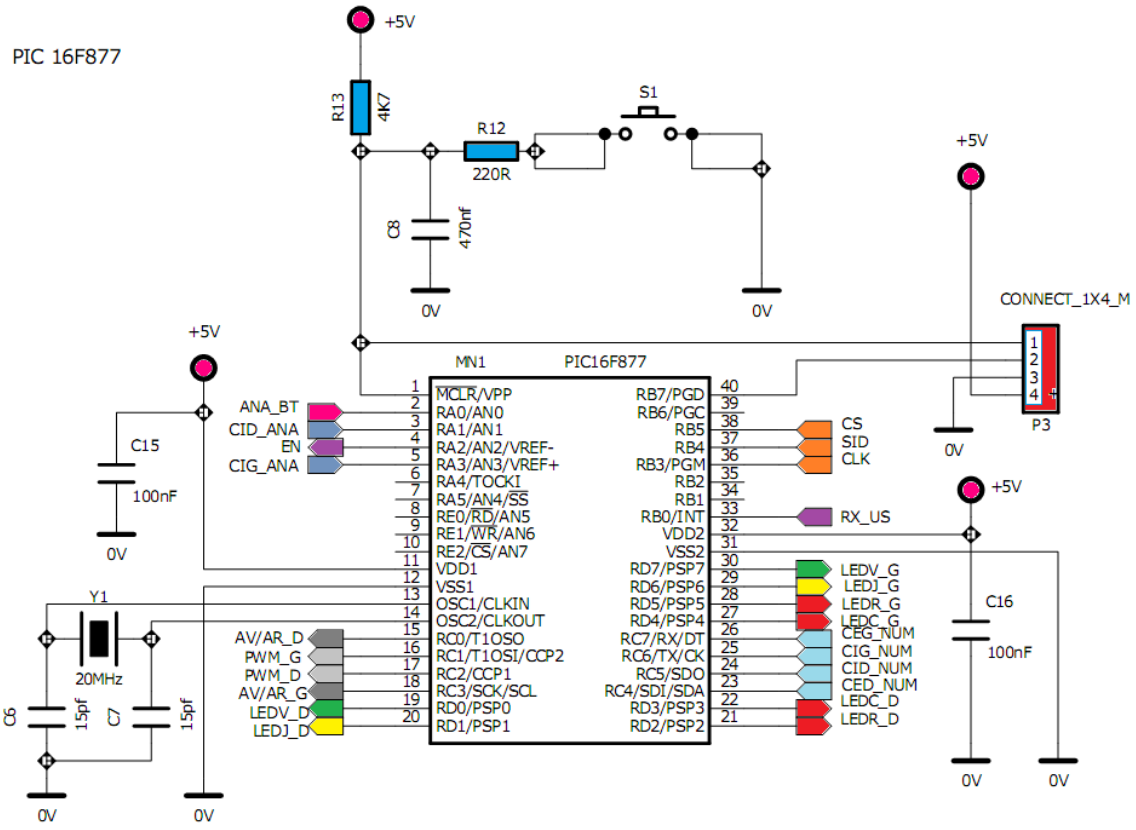
Au possible, les composants utilisés devront être de technologie CMS (si stocks ou technologie existante). Le support DIL40 du microcontrôleur, le microcontrôleur, les boutons poussoirs, les différentes connectiques, le quartz et les condensateurs de filtrage électrochimiques seront quant à eux en technologie traversante.

- **Le microcontrôleur : PIC16F877**

Le microcontrôleur choisi (en stock) est un composant de la société Microchip en boitier DIL40. Ses caractéristiques sont notamment, une architecture interne de son processeur de type RISC avec un jeu de 35 Instructions. Il accepte une horloge d'entrée d'une fréquence allant jusqu'à 20Mhz, lui permettant de traiter un maximum de 5 millions d'instructions par seconde. Il intègre une mémoire FLASH de programme de 8kilo-mots de 14 bits (Le compilateur CC5X utilisé en version gratuite bride à 1kilo-mots la mémoire utilisable), une mémoire RAM de 368 mots de données de 8 octets répartie en 4 banques et une EEPROM de 256 mots de 8octets. Un total de 14 interruptions sont possibles, internes et externes (broche RB0 par exemple). Trois timers différents sont disponibles, 2 en 8 bits et 1 en 16bits, tous avec prescaler.

Il intègre également 2 modules PWM permettant de générer des signaux PWM jusqu'à 10bits de résolution. Enfin, il dispose d'un convertisseur analogique numérique d'une résolution de 10bits utilisable sur un maximum de 8 canaux analogiques. Les autres caractéristiques ne seront pas listées ici, car non utilisées.

Brochage du microcontrôleur et choix d'assignation



- **Reset et Programmation**

La broche 1 permet le reset du composant en manuel via le bouton poussoir S1 ou par soft via la carte d'interface de programmation RS232 externe se connectant sur P3. La broche 40 reprend l'implantation physique de l'une des deux entrées de programmation série utilisée normalement en ICSP (in circuit sérial programming). Mais cette broche est surveillée par un programme 'bootloader' chargé en mémoire et permettant de reprogrammer le PIC sans le retirer de son support. Au démarrage, l'interface impose un reset soutenu configurant de fait les différents ports en entrée. Le bootloader est programmé pour surveiller l'état de cette broche 40 afin de rentrer en communication avec le PC et son logiciel de transfert de programme si elle est à 1. Dans le cas contraire (broche à 0), la communication n'est pas établie et le bootloader renvoie à l'adresse de saut de début de programme.

- **Entrées/ sorties analogiques et numériques**

Le brochage tient compte de différentes contraintes, à la foi soft et hard. Les entrées analogiques sont regroupées sur le même port, en l'occurrence ici le port A. Comme seules 3 entrées de ce type sont nécessaires, le registre dédié ADCON1 du PIC est configuré pour disposer de 3 entrées ou sorties analogiques, respectivement RA0, RA1 et RA3. Le registre TRISA défini le mode de fonctionnement, entrée ou sortie. Le port analogique étant lu en switchant d'une entrée à l'autre pour échantillonner de

façon séquentielle sur l'unique convertisseur disponible, rien n'empêchait d'un point de vue pratique d'utiliser une des broches restante de port A pour une entrée/sortie numérique. La broche RA2 est donc utilisée pour l'Enable de l'oscillateur 40,6 khz.

Les sorties PWM de contrôle de vitesse de moteur droit et gauche sont imposées par la configuration physique du PIC lui-même. Ces deux sorties sont en RC1 et RC2. Les signaux numériques de commutation des relais gérant le sens de rotation des moteurs sont câblés sur le même port C, en RC0 et RC3. L'ensemble représente un quartet facilement masquable sur l'ensemble de l'octet du registre PORTC. (La permutation G/D résulte de nécessités hardware lors du routage de la carte).

CEG_NUM	CIG_NUM	CID_NUM	CED_NUM	AV/AR_G	PWM_D	PWM_G	AV/AR_D
RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0

Dans le même esprit, l'ensemble des signaux d'états numériques des capteurs intérieurs et extérieurs sont regroupés sur le même port, ici la seconde moitié du registre PORTC.

LEDV_G	LEDJ_G	LEDR_G	LEDC_G	LEDC_D	LEDR_D	LEDJ_D	LEDV_D
RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0

De la même façon, toutes les LEDS sont rassemblées sur le même PORTD et disposées selon une organisation logique (gauche-centre-droite). L'ensemble est aisément masquable ou configurable d'un seul tenant par programmation, sur le port tout entier

Les 3 signaux de communication avec l'afficheur seront câblés sur une partie du port B, en RB3, RB4 et RB5.

Le signal de détection d'obstacle RX_US, tout ou rien sera câblé sur l'entrée RB0 afin d'être utilisé en source d'interruption externe, comme le permet justement cette broche.

Les broches non utilisées sur l'ensemble des ports seront laissées en l'air et configurées par soft en sorties numériques.

Les broches propres à l'alimentation et à l'oscillateur externe n'appellent pas à commentaire particulier.

Détail des différentes fonctions électroniques et mise en œuvre

- **Système de détection d'obstacle par capteurs piezzo électriques**

Le cœur du procédé est constitué de 2 capsules (une émettrice, une réceptrice), construites autour d'un crystal résonnant piézoélectrique, couplé à un transducteur acoustique accordé. Ce crystal a la particularité de se déformer mécaniquement lorsqu'il est traversé par un signal électrique et de résonner selon une fréquence propre. Il se comporte mécaniquement comme sont équivalent électrique, le filtre RLC série. A la résonance, c'est un filtre sélectif à bande étroite. IL est donné dans cet état, pour une fréquence 40Khz.

Des tests effectués sur les capsules à notre disposition ont mis en évidence une fréquence de résonnance s'approchant plus des 40,6Khz pour l'amplitude maximum. Nous avons conservé cette valeur mesurée comme fréquence de fonctionnement de notre émetteur à ultrason embarqué.

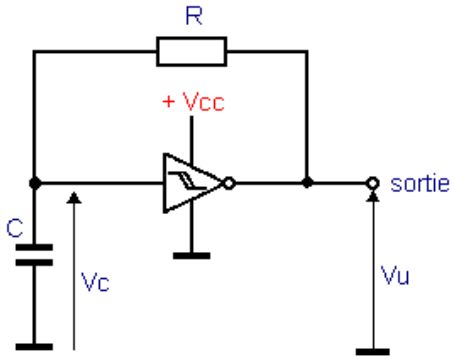
Soumise à un signal carré des ultrasons sont émis à 40,6 kHz par couplage avec le transducteur acoustique. La forme d'onde s'approche de la sinusoïde par rejection des harmoniques basses et hautes. L'amplitude du signal et la puissance fournie au système détermine la portée de l'onde émise, ainsi que la fatigue mécanique supportée. On notera qu'une composante continue à l'état de repos aura tendance à déformer et à laisser sous contrainte le crystal.

Le système mis en œuvre devra donc s'affranchir de cette composante continue et limiter la puissance d'émission dans un objectif de préservation du composant et de maitrise de la portée du système en fonctionnement constant (hors salves).

Le système devra pouvoir être piloté en ON/OFF par le microcontrôleur afin de générer des salves d'émission pour offrir une bonne rejection des échos parasites se propageant dans l'environnement de fonctionnement du robot.

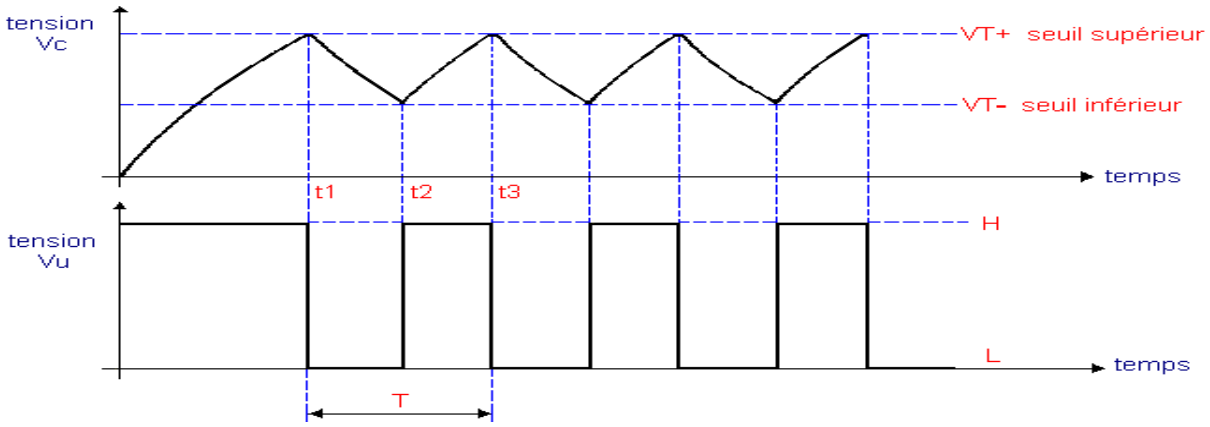
Le fonctionnement de la capsule est réversible. L'onde reçue sur le transducteur mécanique fait rentrer en résonnance le crystal, qui génère in fine un signal électrique de faible amplitude, qu'il conviendra d'amplifier, de filtrer et de mettre en forme afin qu'il constitue une information pertinente sur la présence d'obstacle pour le microcontrôleur.

Choix du circuit oscillateur



Oscillateur à trigger de Schmitt

L'oscillateur est réalisé autour d'un circuit inverseur à entrée trigger de Schmitt. Le circuit peut osciller grâce à la conformation intrinsèque du trigger, offrant 2 seuils distincts de basculement, haut et bas. Le condensateur C se charge et se décharge coincé entre ces 2 seuils. La constante de temps RC et le niveau respectif des seuils en question détermine la fréquence de fonctionnement de l'ensemble.



La période du signal se détermine selon les calculs suivants

$$T = \frac{1}{f\ oscillation} = R.C.\ \ln\left(\frac{Vcc - VT-}{Vcc - VT+} \times \frac{VT+}{VT-}\right)$$

Connaissant la fréquence d'oscillation, il faut déjà avoir une approximation des valeurs des seuils du composant qui est utilisé pour remplir la fonction de porte inverseuse.

Le choix s'est porté sur une porte logique à multiples fonctions configurable de type SN74LVC1G57.

Pour ce composant en moyennant les valeurs de seuils on obtient le tableau suivant :

Symbol	Parameter	Test Conditions	V_{CC}	Min	Typ.	Max	Unit
V_{T+}	Positive-going input threshold voltage		1.65V	0.70		1.20	
			2.3V	1.11		1.60	
			3V	1.50		2.00	
			4.5V	2.16	2,71	2.74	
			5.5V	2.61		3.33	
V_{T-}	Negative-going input threshold voltage		1.65V	0.30		0.75	
			2.3V	0.58		1.03	
			3V	0.80		1.33	
			4.5V	1.21	1.74	1.95	
			5.5V	1.45		2.35	

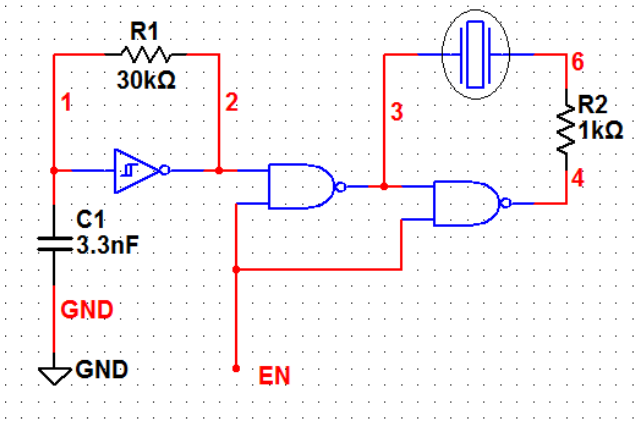
Pour ajuster le réglage de l'oscillateur, un potentiomètre CMS de 20 kΩ sera mis en série avec une résistance tampon de 10 kΩ. Pour trouver la valeur théorique de la capacité C on utilise les valeurs du tableau, potentiomètre à mi course à 10 kΩ.

C = (1 /40600) / (20k * (ln (((5-1,74)/(5-2,71)) * (2,71/1,74)))
C = 2 nF

Potentiomètre à 0 en butée d'un coté
C = (1 /40600) / (10k * (ln (((5-1,74)/(5-2,71)) * (2,71/1,74)))
C = 3 nF

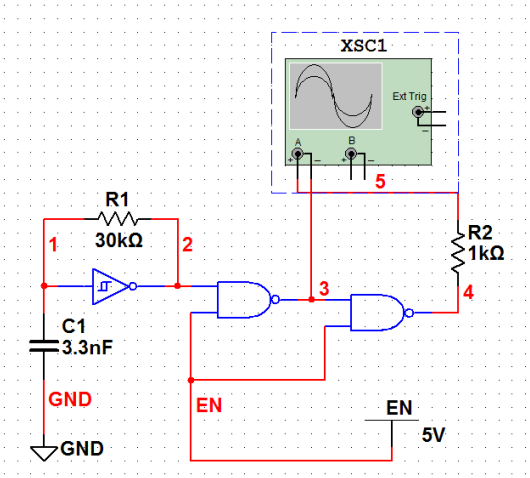
En pratique, une valeur de 3.3 nf conviendra pour obtenir le 40.6kHz ajustable sur la carte.
La première valeur qui avait été retenue, de 470 nF était complètement erronée. La fréquence du montage approchait difficilement les 500 Hz !!

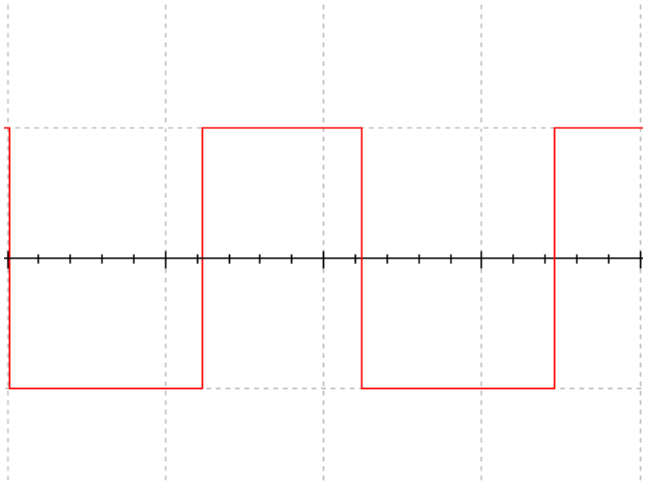
Le schéma complet retenu est le suivant



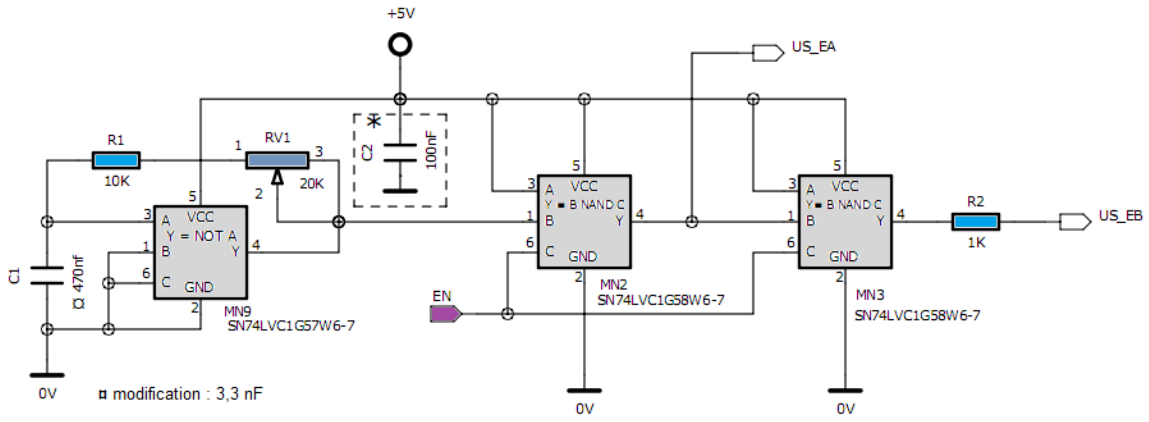
La sortie 2 de l'oscillateur est connectée sur l'une des entrées d'une porte Nand. La cellule piézoélectrique est intercalée entre la sortie 3 de cette porte est la sortie 4 de la seconde Nand via une résistance de 1k limitant le gain du montage. L'entrée EN à 0 positionne les sorties des 2 Nand à 1. La cellule n'est parcourue au repos par aucune composante continue. EN à 1 valide l'ouverture des 2 portes et la cellule voit sur ses entrée le 40,6 kHz en opposition de phase. L'excursion évolue entre +5V et -5V autour du 0V. Ce montage se comporte comme un amplificateur numérique en pont.

Simulation du montage sous Multisim :



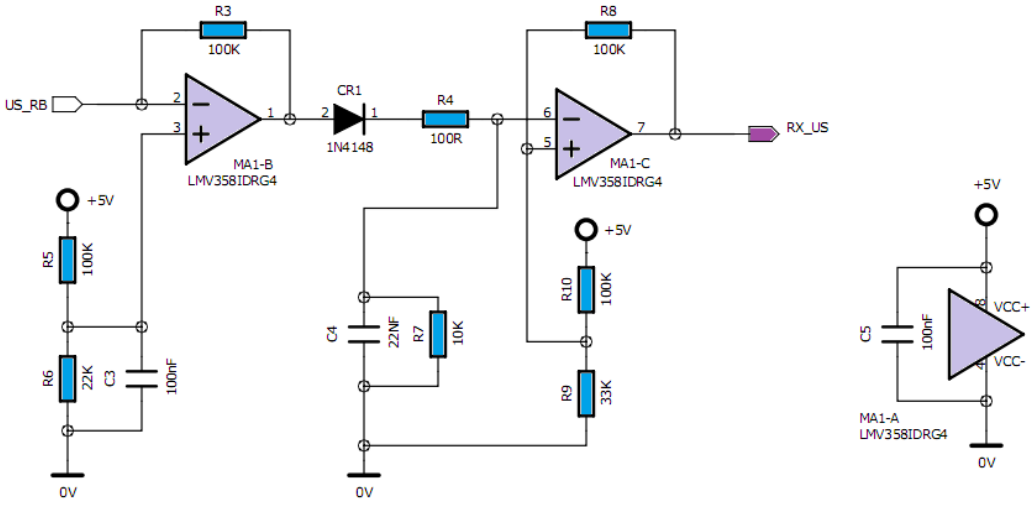


Le circuit final reprend le même schéma mais les portes Nand sont remplacées par des portes logiques à multiples fonctions configurables de type SN74LVC1G58. La résistance de 30kΩ est subdivisée en une résistance tampon de 10kΩ suivie d'un potentiomètre ajustable cms de 20 kΩ. US_EA et US_EB sont connectés avec les broches de l'émetteur piézoélectrique.



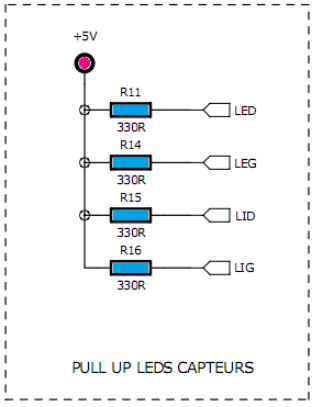
Circuit de réception et de remise en forme du signal

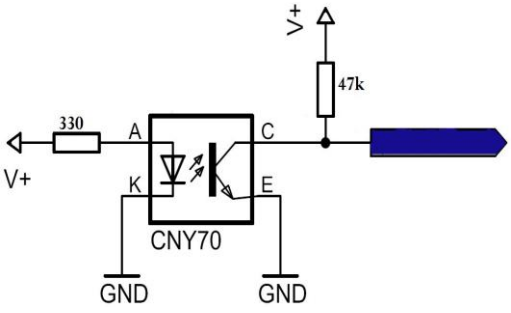
Dans les conditions optimales de réception, l'amplitude du signal mesuré aux bornes de la capsule de réception est de l'ordre de 150mv. Ce signal doit être amplifié et redressé afin de fournir une tension évoluant autour de deux seuils de basculement. Une diode 1N4148 suivie d'un circuit intégrateur redresse et filtre le signal. La résistance R7 permet de vider C4 instantanément lorsque que le signal vient à baisser en sortie de R4, sinon comme le trigger qui suit ne consomme pratiquement rien en entrée, C4 ne pourrait se décharger suffisamment vite. R4 a pour fonction de limiter le courant d'appel. Le montage trigger construit autour de MA1-C bascule autour de 2 seuils. A 2V il bascule et en dessous de 1V il retourne à sa position initiale. L'amplificateur retenu est un double AOP LMV358 en boitier cms. Ses caractéristiques Rail to Rail permettent d'approcher le VCC en sortie.



Capteurs de ligne et mise forme des signaux :

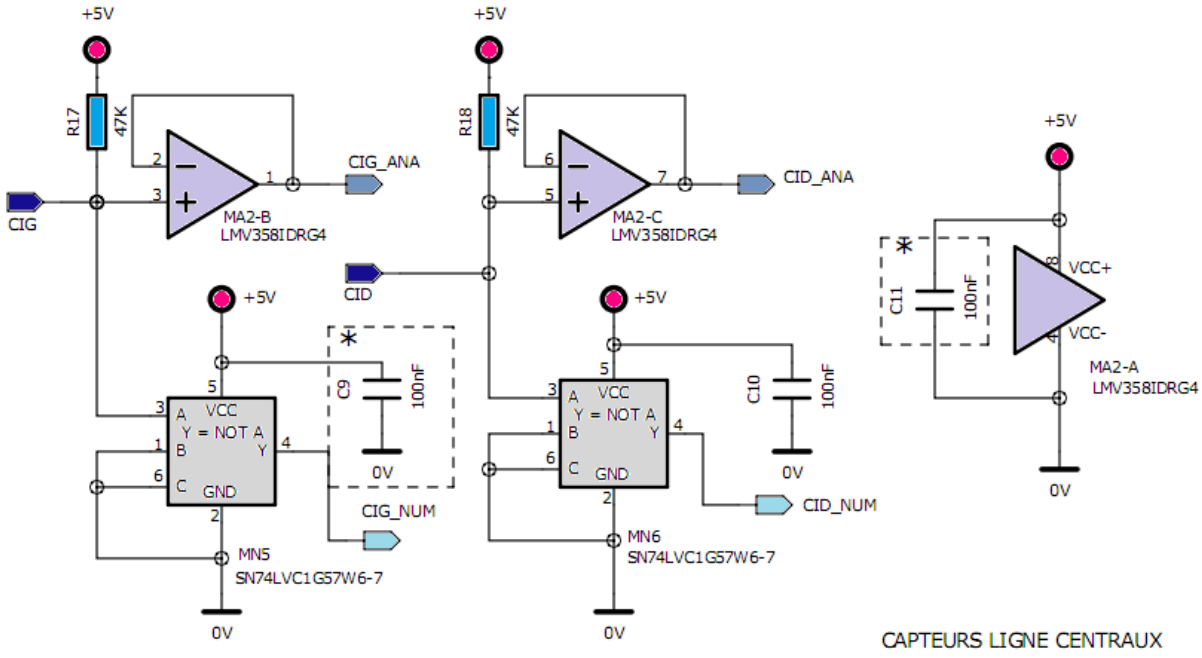
Les capteurs de ligne CNY70 sont constitués d'une LED émettrice et d'un phototransistor nécessitant des composants extérieur de polarisation. La carte mère du robot a été conçue pour permettre de choisir la méthode pour y parvenir de façon déportée, sur la carte fille de gestion, à l'autre bout de la nappe souple reliant les 2 cartes. Les LEDS peuvent supporter jusqu'à 50ma mais pour des raisons de consommation il a été choisi de limité expérimentalement le courant à environ 9mA.





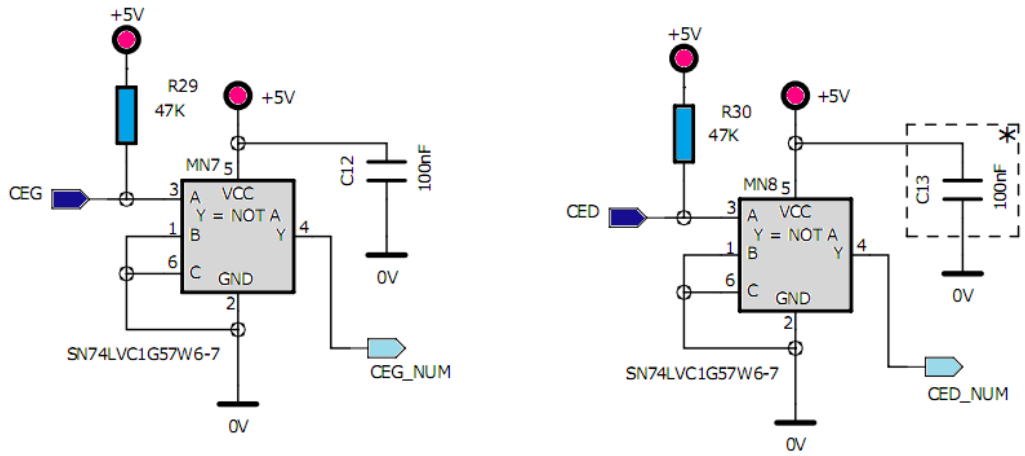
Les phototransistors de tous les capteurs de ligne sont polarisés par une résistance de 47 kΩ. Cette valeur à été choisi expérimentalement. Elle offre un compromis intéressant entre vitesse de transition noir/blanc en numérique et linéarité de variation de niveau en analogique.

Les capteurs centraux sont bufferisés par des AOP LMV358 montés en suiveurs de tension afin d'attaquer les entrées analogiques du PIC à basse impédance. Des porte logiques à multiples fonctions configurable de type SN74LVC1G57 montées en inverseurs permettent grâces à leurs entrées trigger de mettre en forme en tout ou rien les signaux issus des capteurs, tout en les bufferisant avant d'attaquer les entrées numériques du PIC.



CAPTEURS LIGNE CENTRAUX

La même chose pour les capteurs de ligne externes



Dispositif de commande par bouton poussoir :

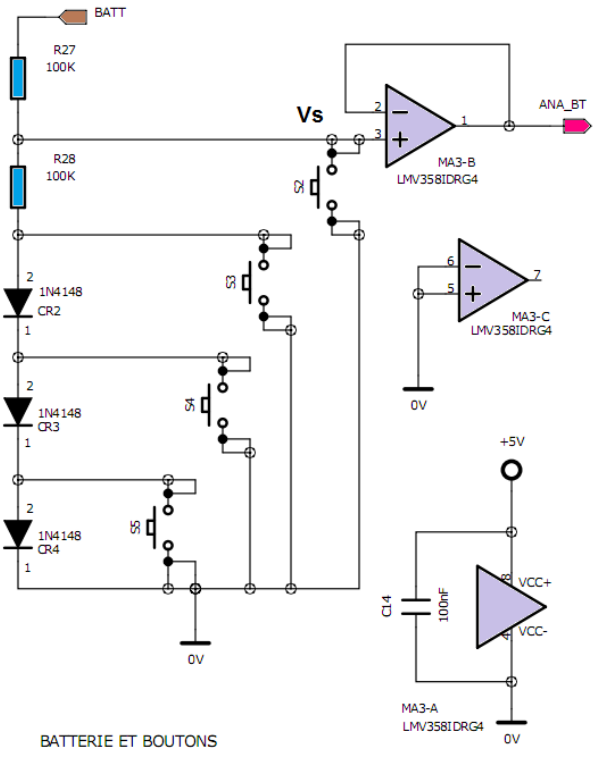
L'idée de départ qui à présidé à la définition du système était de pouvoir mesurer l'état de la tension de la batterie tout en pouvant également générer des ordres codés au PIC, par niveaux de tensions et ceci en utilisant qu'une seule broche d'entrée de celui-ci. L'état de charge de la batterie ne devant pas influer sur la discrimination du bouton enfoncé dans la plage de tension où le PIC fonctionne correctement, à savoir dans notre cas lorsque V BATT est compris entre 7,2 et 5,6 V(tension imposée par le dropout de 0,5 V du régulateur LM2940CT-5.0).

Le schéma suivant a été retenu malgré son gros défaut, un pouvoir de discrimination indigent. En effet la différence entre chaque niveau de tension est de seulement de 200mV, ce qui peut provoquer des erreurs d'interprétation de lecture de bouton par le microcontrôleur lors des fluctuations de l'alimentation du robot. Cette erreur de conception sera conservée pour imposer sa résolution par des astuces de programmation du microcontroleur.

la tension de sortie est la suivante Vs $\frac{Vbatt - n}{2} + n$

Le faible courant traversant les diodes influe sur les seuils (0,4V).Pour aucun bouton enfoncé, n = 1,2V. La valeur numérique une fois convertie en 8bits par le convertisseur du PIC devient :

$\frac{255}{5} Vs$



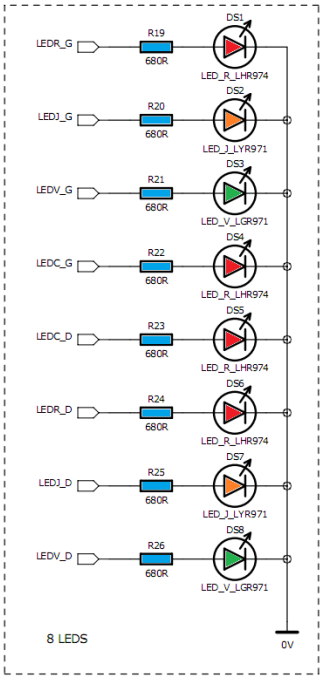
	n (V)	Vbatt	Vs	Valeur numérique convertie
aucun	1,2	7,2	4,2	215
S2	0	7,2	0	0
S3	0	7,2	3,6	183
S4	0,4	7,2	3,8	193
S5	0,8	7,2	4	204

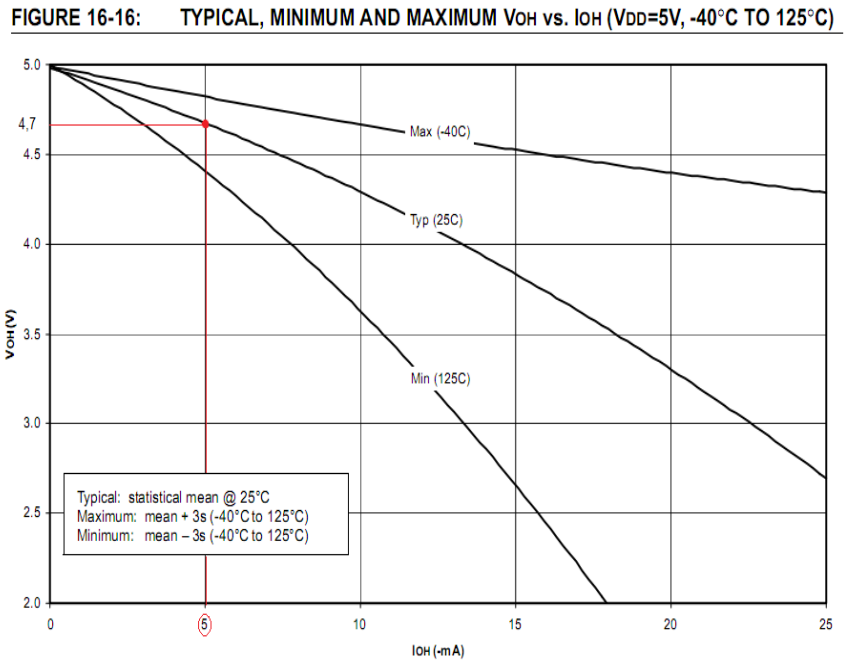
Tableau des valeurs théoriques attendues

L'AOP LMV358 monté en suiveur de tension pour attaquer la broche RA0 du PIC à basse impédance.

Les 8 LEDS :

Pour éviter de charger les sorties du PIC qui les alimente en direct, un courant de repos d'environ 5mA a été retenu. Des résistances normalisées de 680Ω sont intercalées entre les broches du PIC et les anodes. Les LEDS ont une tension de fonctionnement différente en fonction de la couleur, 1,7 V pour les rouges, 1,9 V pour les oranges et 1,95 V pour les vertes. Ces tensions sont confirmées par la mesure pour les LEDS cms utilisées. La tension mesurée en sortie du PIC est conforme à la valeur typique donnée sur le datasheet, à savoir 4,7 V.





CAO : réalisation du schéma et routage de la carte

Le travail de CAO a été réalisé avec le logiciel Padslogic de Mentor Graphic. La carte comporte des éléments traversant, ainsi que des composants cms. Les contraintes de fabrication sont les suivantes :

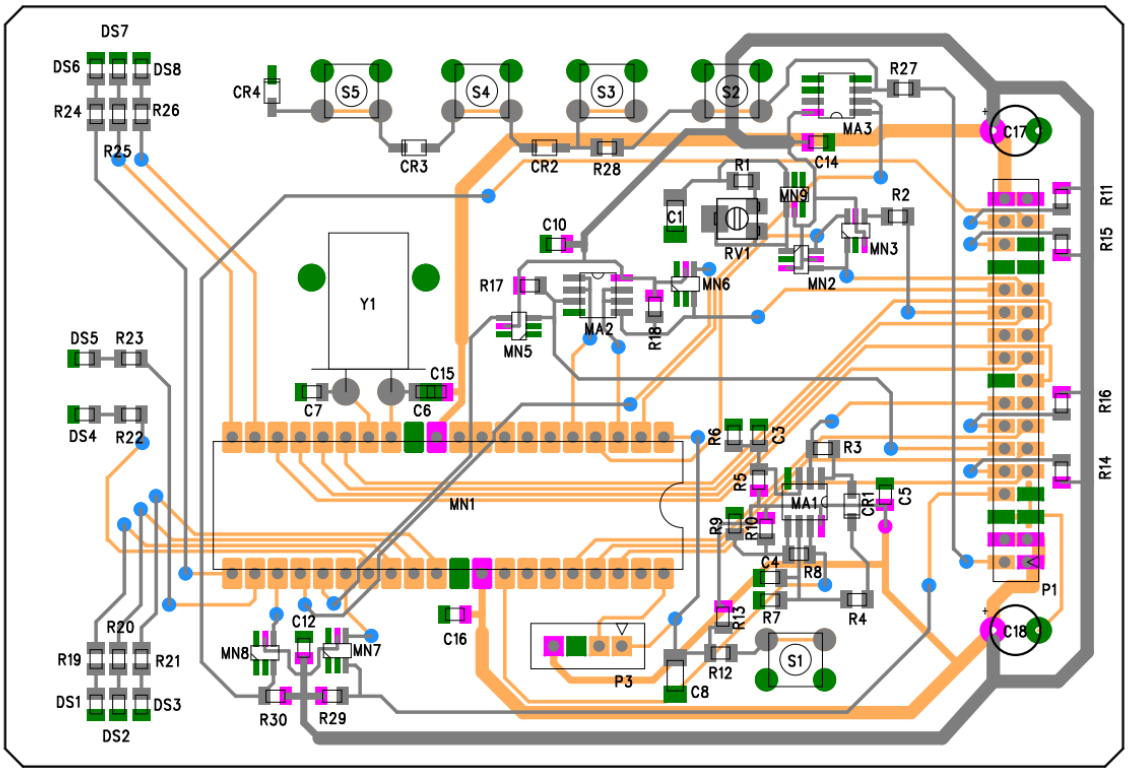
- Gravure directe du circuit sur machine LPKF (isolement de 0,2 mm minimum en 2 passes de fraise).
- Si possible rester dans cet isolement pour limiter le nombre de passes (usure de l'outil).
- Détourage de pastilles en 0,3 mm pour faciliter la soudure manuelle des composants traversant.
- Pas de routage en Top :
  - Pour le PIC (impossibilité de soudure sous le support DIL40), pastilles crées détourées en Top pour ne pas toucher le plan de masse.
  - Idem pour les connecteurs, pastilles crées détourées en Top, impossibilité de soudure en Top.
- Limitation du nombre de Via entre Top et Bottom (à souder manuellement, pas de trous métallisés, via réalisée avec des queues de composants traversant).
- Modification des freins thermiques sur les Via en plan de masse (limitation de conductivité thermique, facilité de soudage manuel).

- Agrandissement des plages de réception des composants cms pour augmenter la conductivité thermique de celles-ci et éviter le surf des cms sur la carte lors des phases de refusion de la pate. Les composants sont posés et tenus uniquement par contact et gravité et peuvent migrer avec la pate si celle-ci vient à s'écouler selon des gradients de température.

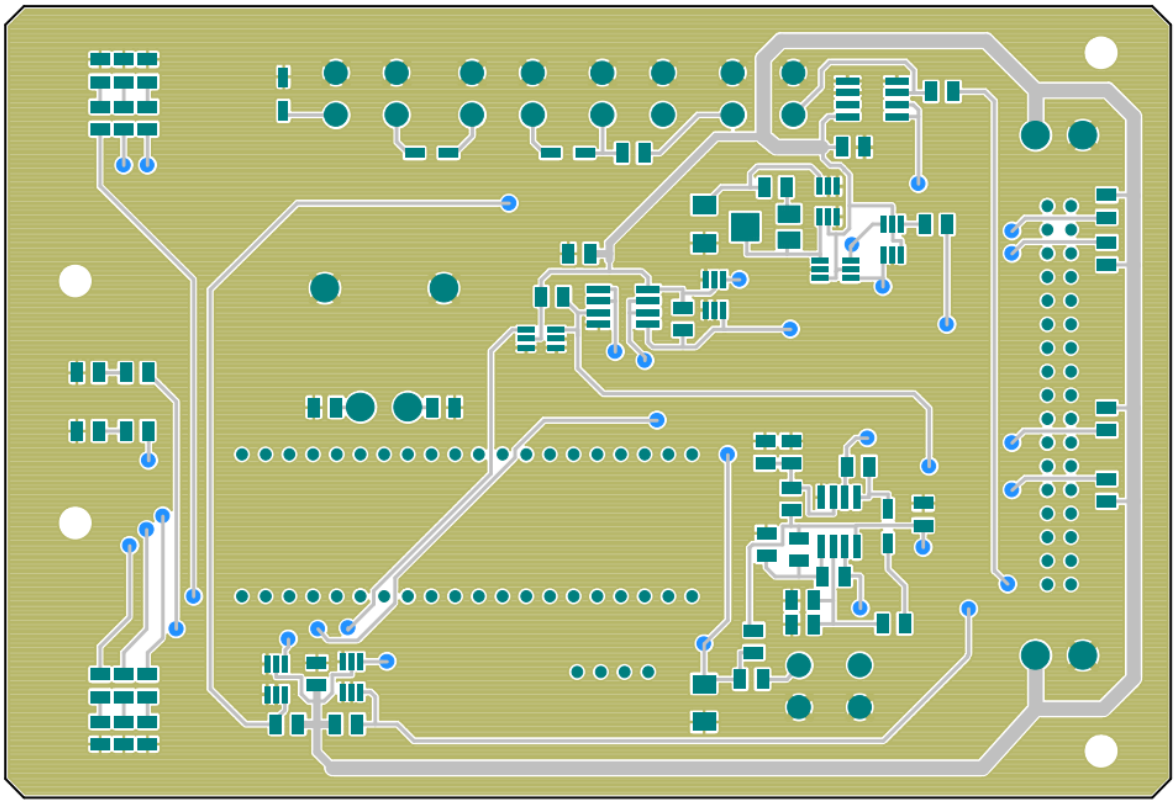
Mes choix :

Comme j'ai privilégié l'assignation des broches du PIC selon des considérations de programmation, le routage s'en trouve impacté. J'ai placé le PIC afin de pouvoir router le plus directement possible par-dessous en bottom les signaux de commandes moteurs, d'afficheur et d'allumage des LEDS. J'ai pris soin de regrouper les signaux par fonctions en aménageant entre ces regroupements des plans de masse ayant le plus de surface possible et le moins de fentes. Autant que faire ce peut, j'ai essayé d'accompagner chaque piste d'un plan de masse et d'éviter la diaphonie entre pistes en les éloignant, évitant qu'elles se suivent et les faisant se superposer d'un plan à l'autre (Top-Bottom), en un seul point. Le routage des LEDS par Via n'est pas évité puisque n'ayant pas d'incidence importante sur l'intégrité du signal. Le quartz et ses condensateurs sont montés au plus prêt du PIC, dans une zone de plan de masse dégagée. J'ai limité la longueur des pistes entre le tampon analogique MA2 et les entrées analogiques du PIC. J'ai placé l'oscillateur et l'amplificateur Ultrasons vers le connecteur 2x17, au plus prêt de la nappe. J'ai placé 2 condensateurs électrochimiques au plus prête des sorties 5v du connecteur 2x17 et j'ai routé 2 larges pistes de 5v en Top et Bottom. . Les condensateurs céramiques 100nF de découplage des alimentations sont placés au plus prêt des boitiers. Enfin, le clavier, le bouton de reset et le connecteur de programmation sont disposé en bord de carte, à droite et à gauche, dans une zone dégagée afin d'en facilité l'accès une foi la carte montée sur le robot.

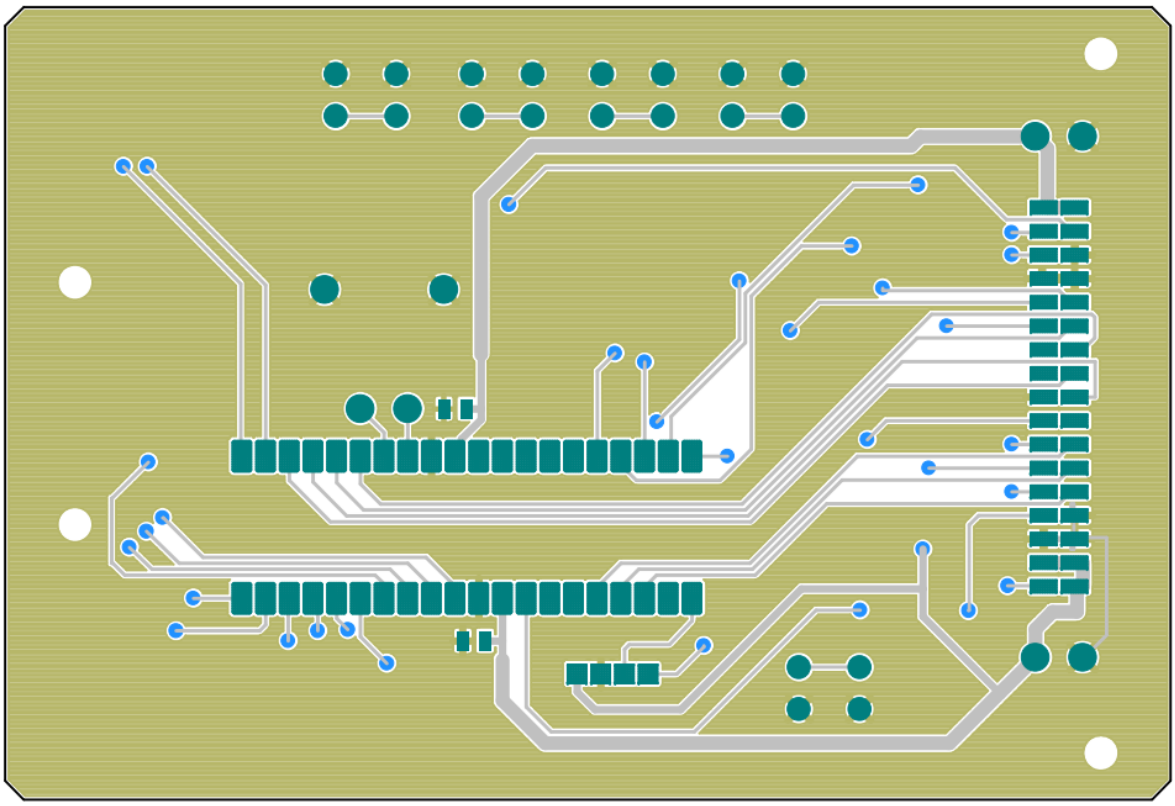
Plan d'équipement



Top & plan 0 volt

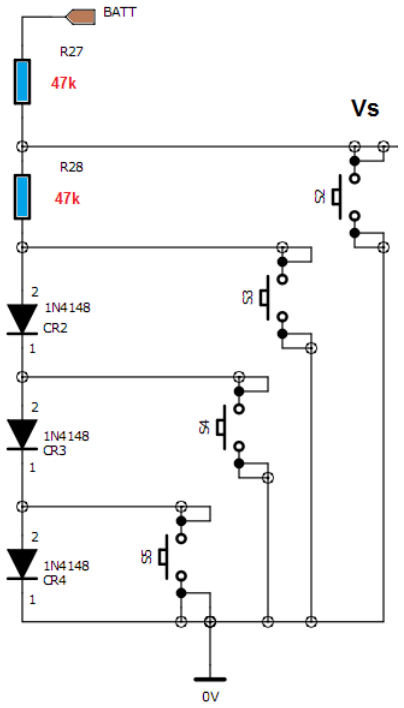


Bottom & plan 0 volt

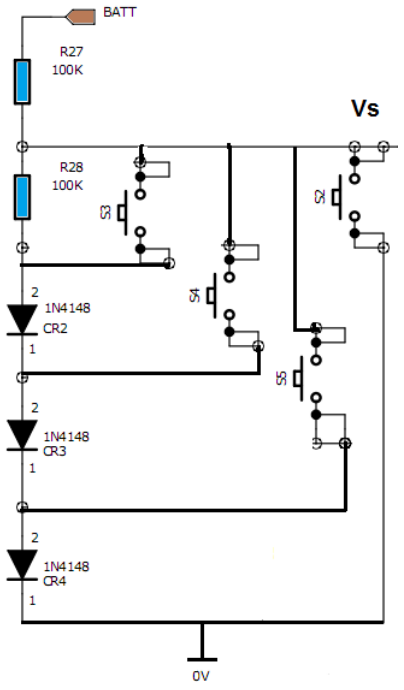


Correctifs

Propositions d'amélioration du dispositif de clavier

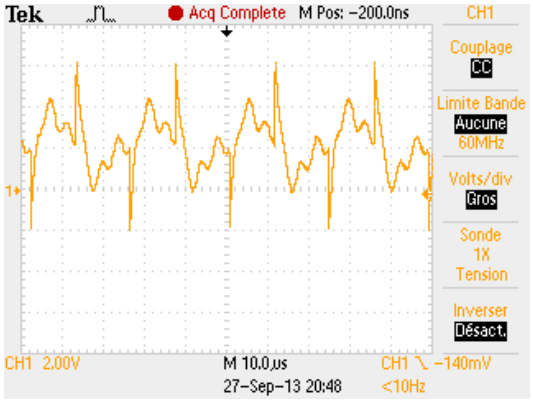


Remplacement des résistances R27 et R28 du pont diviseur par des valeurs plus faibles pour augmenter le courant dans les diodes et relever leur seuil. Pas de 300mv entre chaque bouton. En diminuant encore plus la valeur des résistances (4,7KΩ) on augmente le courant consommé au détriment de l'autonomie, mais ce courant disponible à la sortie de R27 peut permettre de s'affranchir du tampon analogique.

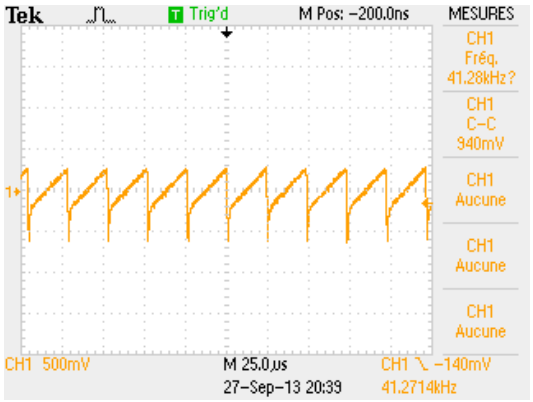


Les boutons poussoirs sont connectés entre le point milieu du pont diviseur et une addition successive de diodes. Le pas entre chaque bouton est équivalent au seuil des diodes utilisées en fonction du courant les traversant (ici 400mV).

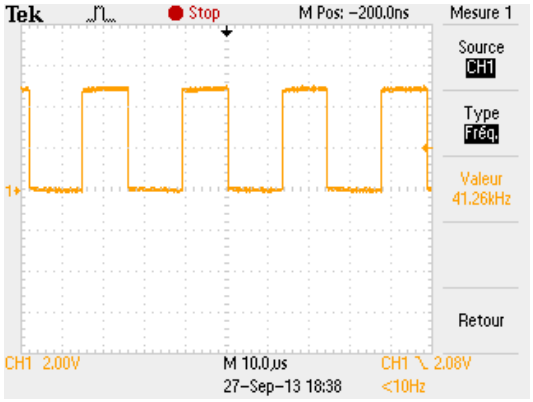
Mesures



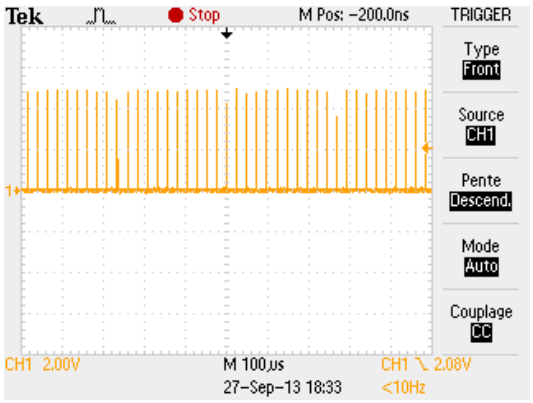
Forme du signal perturbé mesuré après la résistance de R2 de 1KΩ en sortie d'oscillateur. Retour par la nappe d'un signal généré par la capsule émettrice elle-même et se superposant au 40Khz.



Forme du signal perturbé mesuré en sortie du tampon analogique LMV358 du capteur intérieur droit (CID). Lorsque l'Enable d'ultrason est actif, le 40Khz perturbe par couplage dans la nappe. (1V c à c)



Transition sur le capteur, du noir au blanc, perturbation mesurée sur la sortie de la porte 1G57 (CID). Lorsque l'Enable de l'ultrason est actif, le 40 kHz perturbe par couplage dans la nappe.



Transition sur le capteur, du blanc au noir en sortie de la porte 1G57 (CID). Lorsque l'Enable d'ultrason est actif, le 40 kHz perturbe par couplage dans la nappe.

Conclusion

Bien qu'ayant déjà une expérience de la réalisation de cartes électroniques, celle-ci était restée à la mesure des compétences que j'avais jusqu'alors et des modestes moyens dont je disposais.

Ce projet m'a permis de mettre en application les connaissances que j'ai pu acquérir lors du second module de la formation ETD, à savoir 'Mise au point et prototypage d'un système électronique'.

J'ai pu appréhender les différentes étapes du développement d'une carte électronique sous un aspect professionnel, en utilisant des outils logiciels et matériels adaptés. Je me suis familiarisé avec la mise en œuvre des composants de technologie CMS, la gravure directe de carte prototype, l'outil de placement manuel des CMS et enfin avec la soudure par refusion.

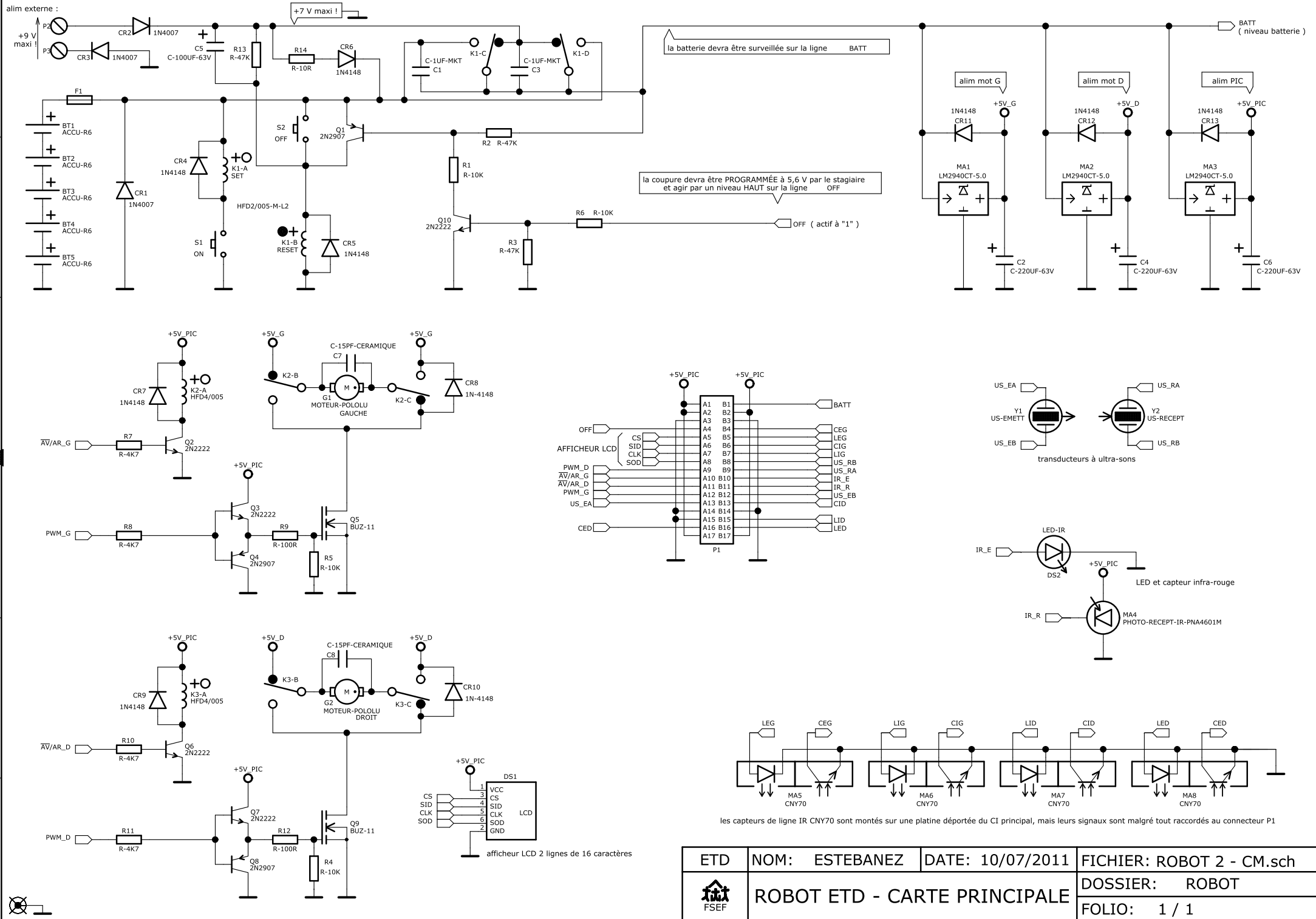
L'intégration d'un microcontrôleur dans le projet m'a donné l'opportunité de faire le lien entre l'univers Hardware, capteurs et circuits associés et celui de la programmation. J'ai mis en pratique ce que j'ai appris de la programmation en langage C pour faire évoluer avec succès le robot suiveur de ligne.

Annexes

1. Schéma carte mère robot
2. Schémas carte de gestion du robot
3. Schéma interface PIC RS232
4. Nomenclature
5. Photographies
6. Outils de prototypage
7. Programmation en langage C
   - Suivi de ligne et détection d'obstacle
   - Enregistrement et restitution de parcours

Schéma

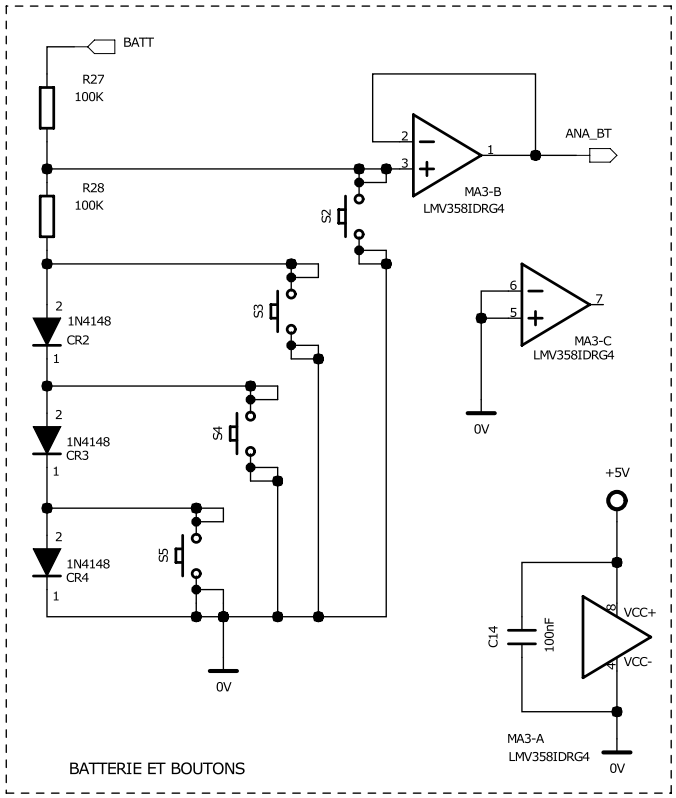
Carte mère robot

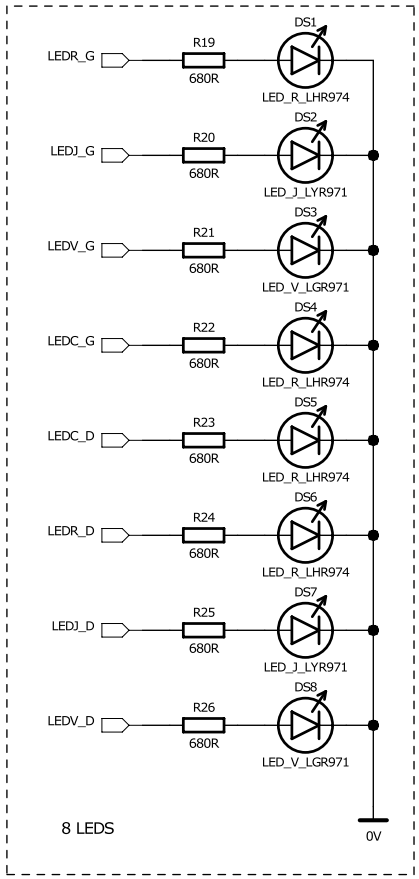


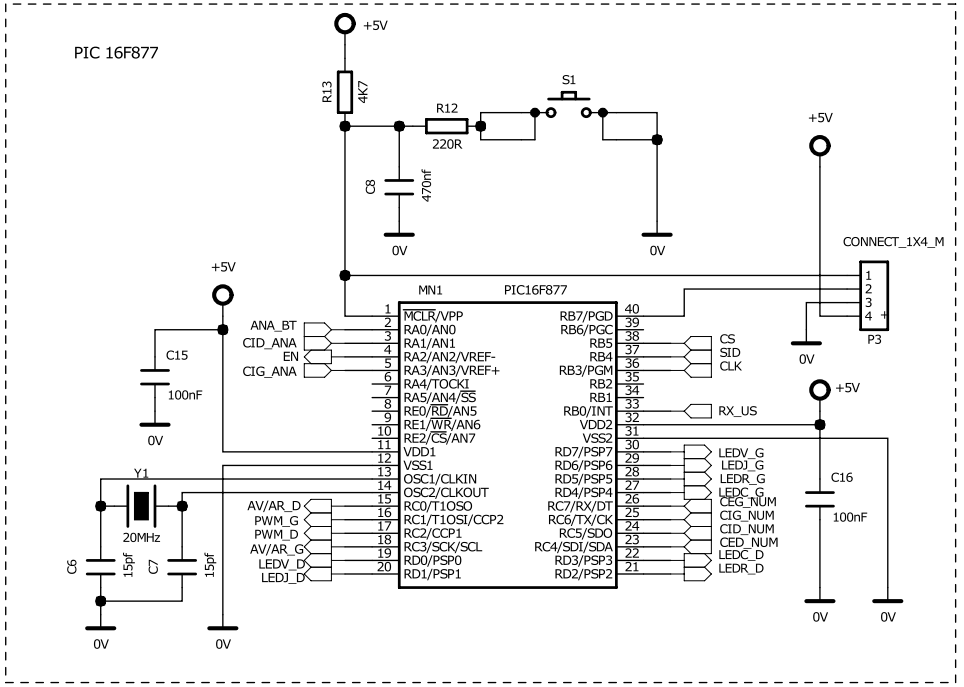
ETD	NOM: ESTEBANEZ	DATE: 10/07/2011	FICHIER: ROBOT 2 - CM.sch
FSEF	ROBOT ETD - CARTE PRINCIPALE		DOSSIER: ROBOT
			FOLIO: 1 / 1

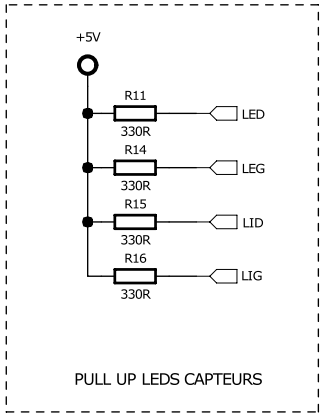
Schémas

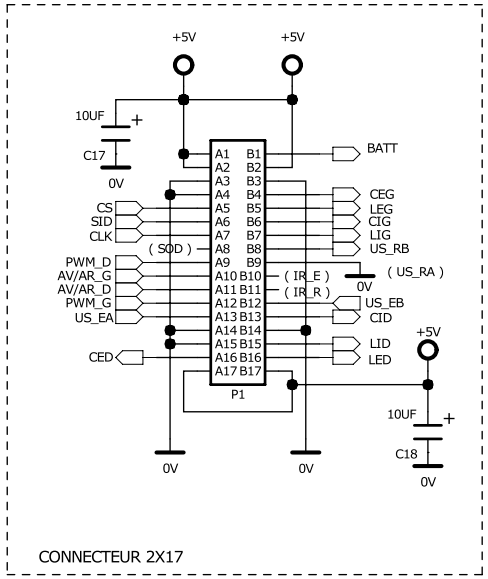
Carte de gestion du robot

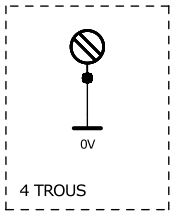




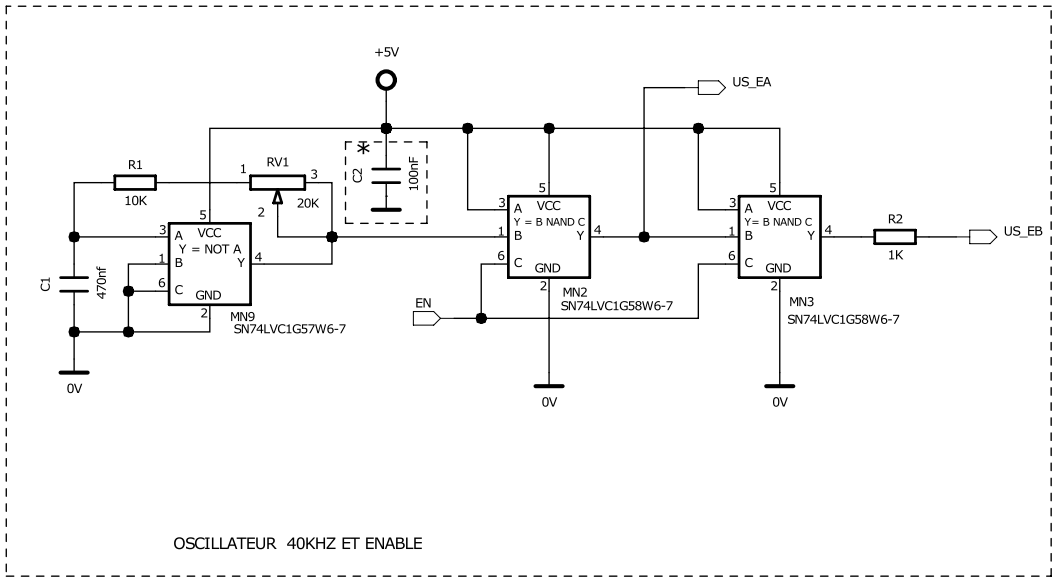


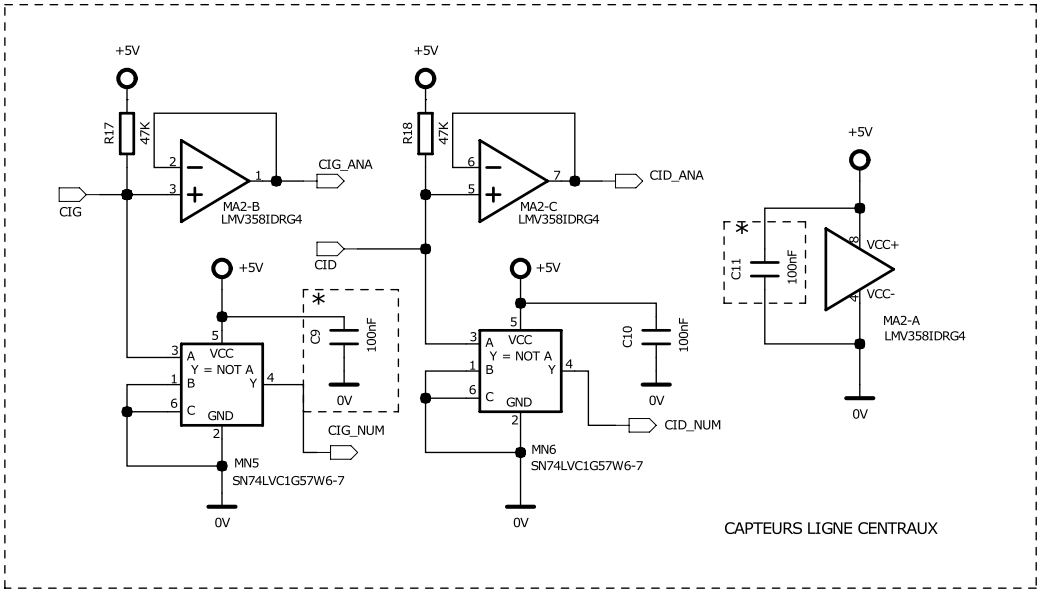




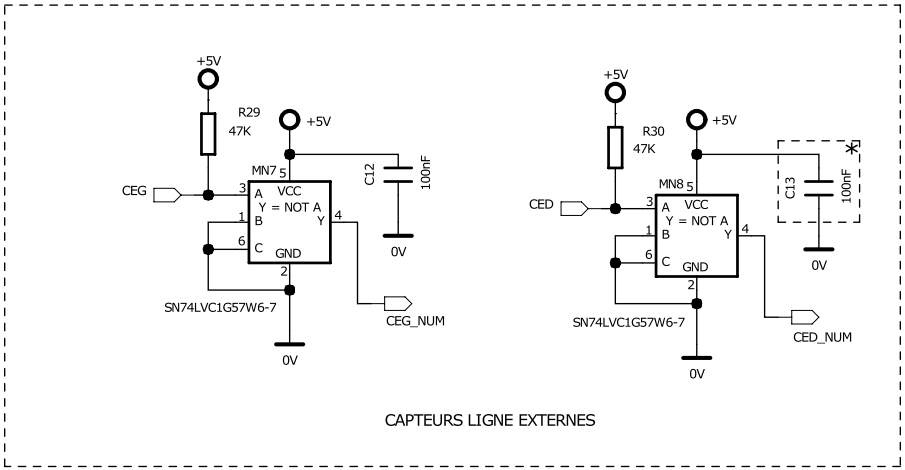


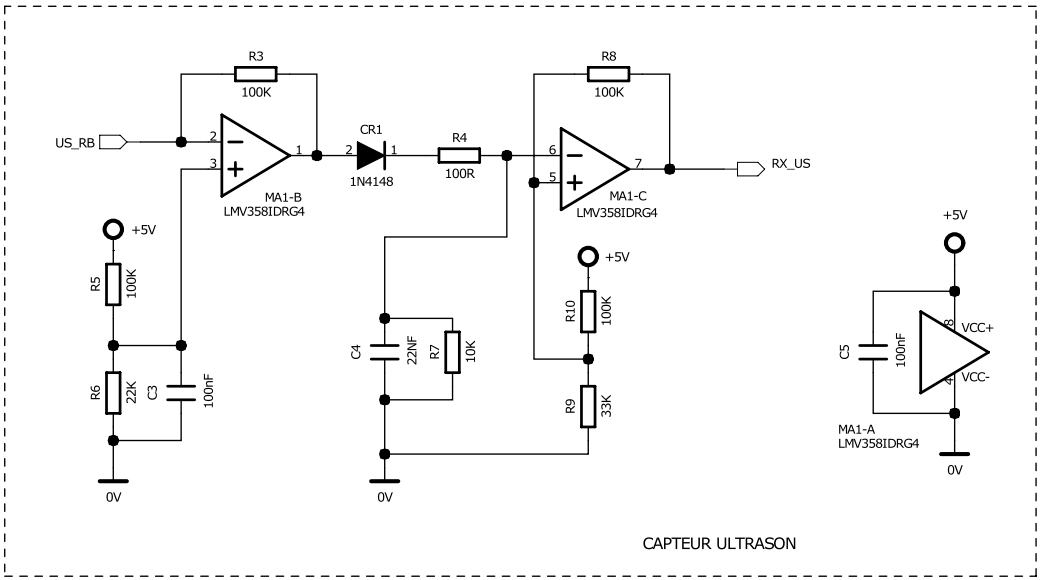
ETD12	NOM: Depoorter	DATE: 04/09/2013	FICHIER:
FSEF	Robot suiveur de ligne		DOSSIER: module 2
			FOLIO: 1/2





* Condensateurs de découplage non implantés, boitiers proches, 100 nf suffisant.

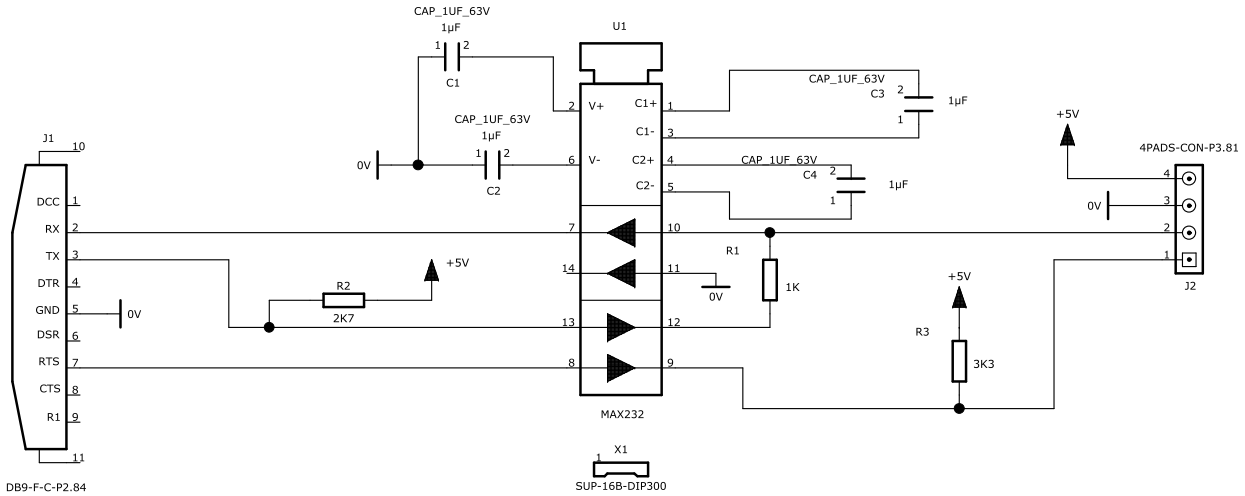


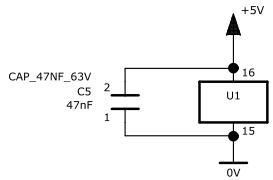


ETD12	NOM: Depoorter	DATE: 04/09/2013	FICHIER:
FSEF	Robot suiveur de ligne		DOSSIER: module 2
			FOLIO: 2/2

Schéma

Interface PIC RS232





10ADME	NOM: MALGHEM C.	DATE: 28/06/2011	FICHIER: Interface PIC 1.0.sch
FSEF	INTERFACE PIC		DOSSIER:
			FOLIO: 1 / 1

Nomenclature

Nomenclature

Reference	Description	VALEUR
C1	Condensateur cms boitier 1206	470nf
C2	Condensateur cms boitier 0805	100nF
C3	Condensateur cms boitier 0805	100nF
C4	Condensateur cms boitier 0805	22NF
C5	Condensateur cms boitier 0805	100nF
C6	Condensateur cms boitier 0805	15pf
C7	Condensateur cms boitier 0805	15pf
C8	Condensateur cms boitier 0805	470nf
C9	Condensateur cms boitier 0805	100nF
C10	Condensateur cms boitier 0805	100nF
C11	Condensateur cms boitier 0805	100nF
C12	Condensateur cms boitier 0805	100nF
C13	Condensateur cms boitier 0805	100nF
C14	Condensateur cms boitier 0805	100nF
C15	Condensateur cms boitier 0805	100nF
C16	Condensateur cms boitier 0805	100nF
C17	Condensateur cms boitier 0805	10UF
C18	Condensateur cms boitier 0805	10UF
CR1	MULTICOMP - 1N4148W - DIODE,SMALL SIGNAL, 75V, SOD-123F	1N4148
CR2	MULTICOMP - 1N4148W - DIODE,SMALL SIGNAL, 75V, SOD-123F	1N4148
CR3	MULTICOMP - 1N4148W - DIODE,SMALL SIGNAL, 75V, SOD-123F	1N4148
CR4	MULTICOMP - 1N4148W - DIODE,SMALL SIGNAL, 75V, SOD-123F	1N4148
DS1	led cms boitier 0805	ROUGE
DS2	led cms boitier 0805	JAUNE
DS3	led cms boitier 0805	VERT
DS4	led cms boitier 0805	ROUGE
DS5	led cms boitier 0805	ROUGE
DS6	led cms boitier 0805	ROUGE
DS7	led cms boitier 0805	JAUNE
DS8	led cms boitier 0805	VERT
MA1	double Amplificateur opérationnel rail to rail SOIC D 8	LMV358
MA2	double Amplificateur opérationnel rail to rail SOIC D 8	LMV358

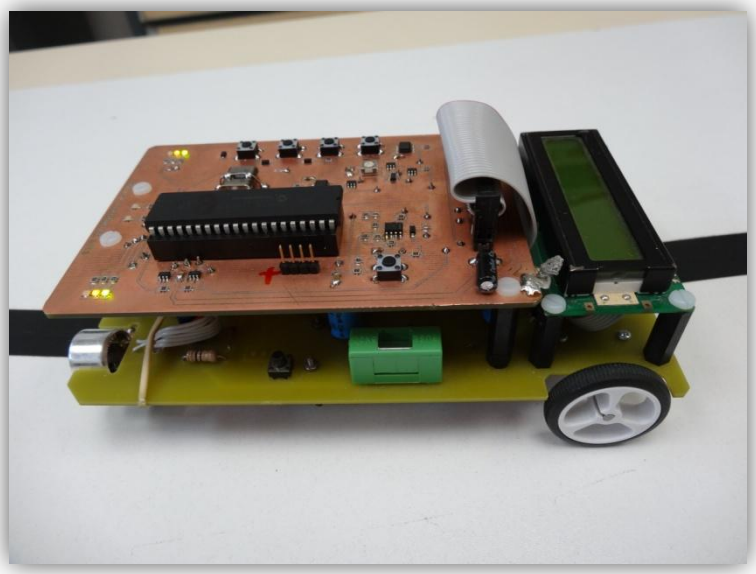
Nomenclature

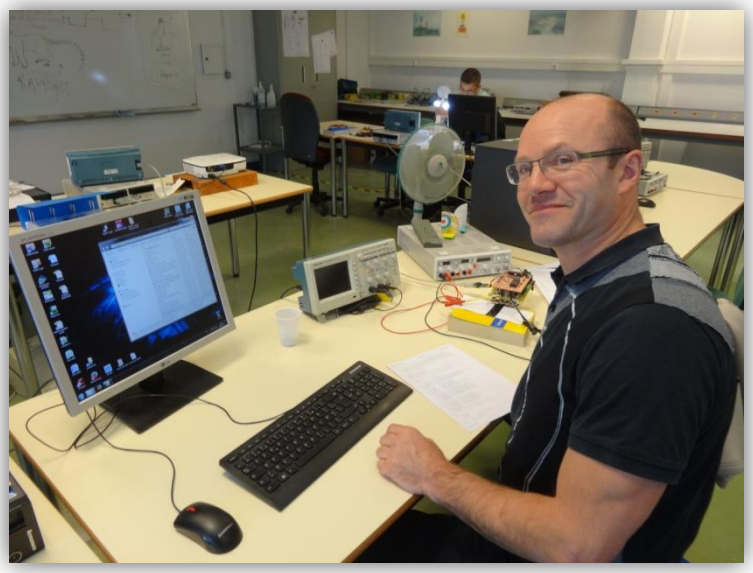
Reference	Description	VALEUR
MA3	double Amplificateur opérationnel rail to rail SOIC D 8	LMV358
MN1	Microcontroleur microchip DIL 40	PIC16F877
MN2	porte programmable SN74LVC1G58W6-7 boitier SOT 26	Y = B NAND C
MN3	porte programmable SN74LVC1G58W6-7 boitier SOT 26	Y= B NAND C
MN5	porte programmable SN74LVC1G57W6-7 boitier SOT 26	Y = NOT A
MN6	porte programmable SN74LVC1G57W6-7 boitier SOT 26	Y = NOT A
MN7	porte programmable SN74LVC1G57W6-7 boitier SOT 26	Y = NOT A
MN8	porte programmable SN74LVC1G57W6-7 boitier SOT 26	Y = NOT A
MN9	porte programmable SN74LVC1G57W6-7 boitier SOT 26	Y = NOT A
P1	connecteur barrette picots mâle	2X17M
P3	connecteur barrette picots femelle	1X4F
R1	résistance cms boitier 0805	10K
R2	résistance cms boitier 0805	1K
R3	résistance cms boitier 0805	100K
R4	résistance cms boitier 0805	100R
R5	résistance cms boitier 0805	100K
R6	résistance cms boitier 0805	22K
R7	résistance cms boitier 0805	10K
R8	résistance cms boitier 0805	100K
R9	résistance cms boitier 0805	33K
R10	résistance cms boitier 0805	100K
R11	résistance cms boitier 0805	330R
R12	résistance cms boitier 0805	220R
R13	résistance cms boitier 0805	4K7
R14	résistance cms boitier 0805	330R
R15	résistance cms boitier 0805	330R
R16	résistance cms boitier 0805	330R
R17	résistance cms boitier 0805	47K
R18	résistance cms boitier 0805	47K
R19	résistance cms boitier 0805	680R
R20	résistance cms boitier 0805	680R
R21	résistance cms boitier 0805	680R

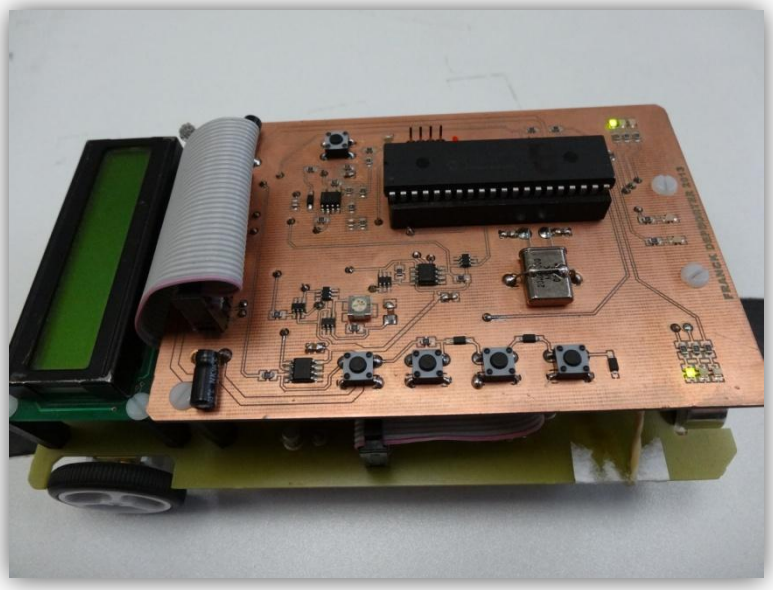
Nomenclature

Reference	Description	VALEUR
R22	résistance cms boitier 0805	680R
R23	résistance cms boitier 0805	680R
R24	résistance cms boitier 0805	680R
R25	résistance cms boitier 0805	680R
R26	résistance cms boitier 0805	680R
R27	résistance cms boitier 0805	100K
R28	résistance cms boitier 0805	100K
R29	résistance cms boitier 0805	47K
R30	résistance cms boitier 0805	47K
RV1	TRIMMER CERMET 4MM 20K	20K
S1	bouton poussoir	1NO
S2	bouton poussoir	1NO
S3	bouton poussoir	1NO
S4	bouton poussoir	1NO
S5	bouton poussoir	1NO
Y1	CRYSTAL, HC-49/S, 20.0MHZ	20MHz

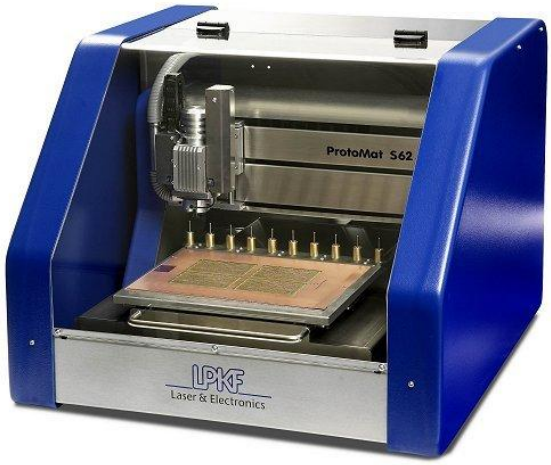
Photographies



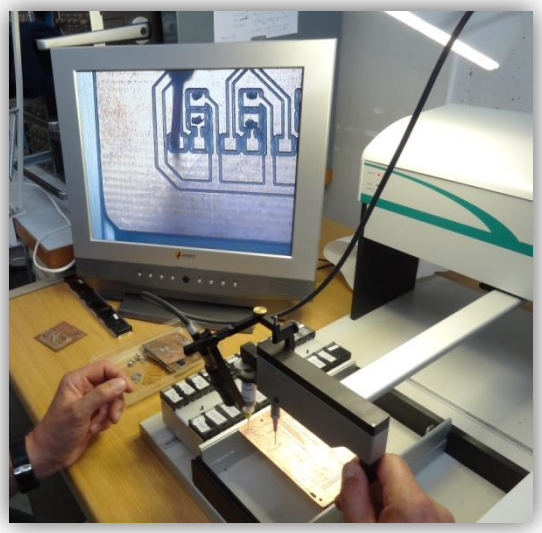




Outils de prototypage



Graveuse LPKF



Outil de placement manuel de cms FRITSCH LM901

Four à refusion

Programmation
En
Langage C

Suivi de ligne et détection d'obstacle

```
1
2   // ----- gestion mixte " analogique et numérique " de suivi de ligne + ultrasons ---------------
3
4   // la gestion de suivi de ligne est gérée en analogique. Chaque moteur a sa vitesse modulée
5   // par l'évolution analogique de son capteur opposé. En parallèle un switch est effectué sur
6   // un octet "vitesse", lequel est alimenté par l'état numérique des capteurs de ligne centraux.
7   // Chaque "case" gère des flags gauche et droit de perte de ligne enregistrés dans l'octet vitesse
8   // également. Cette gestion de l'évolution de l'etat des capteurs et des flags permet de controler
9   // le retour ligne extreme uniquement en numérique en sachant d'où vient le robot et ceci même si
10  // il s'arrete à cause d'un obstacle.
11
12  // identification de la cible HARD
13
14  #define capteur_droit vitesse.1
15  #define capteur_gauche vitesse.2
16  #define flag_gauche vitesse.3
17  #define flag_droit vitesse.0
18  #pragma chip PIC16F877 // attention, PIC16F877.h dans dossier programmation robot
19
20  #include "int16cxx.h"  // ajoute la gestion du vecteur d' interruption
21
22  //#pragma origin 4          // origin 5 mini pour prog sans interruption ; origin 4 avec int
23  #pragma origin 4 // gestion interruption
24
25  // personnalisation de l'appellation des ports
26
27  bit ANA_BT @PORTA.0;  // entrée analogique des boutons poussoirs// RA0
28  bit CID_ANA @PORTA.1; // entrée capteur intérieur droit en analogique // RA1
29  bit EN @PORTA.2;      // sortie enable des portes logiques & de l'oscillateur 40,6 khz // RA2
30  bit CIG_ANA @PORTA.3; // entrée capteur intérieur gauche en analogique// RA3
31  //TRISA=0b001011      // broches RA4 RA5 non cablées ---> en out ob00----
32
33  bit RX_US @PORTB.0;   // récepteur ultrason RB0
34  bit CLK @PORTB.3;     // sortie CLK afficheur RB3
35  bit SID @PORTB.4;     // sortie SID afficheur RB4
36  bit CS @PORTB.5;      // sortie CS afficheur RB5
37  bit PGD @PORTB.7;     // entrée programmation ( pas d'autre usage ) RB7
38  //TRISB=0b10000001    // broches RB1 RB2 RB6 non cablées ---> en out
39
40  bit AV_AR_D @PORTC.0; // sortie sens moteur droit RC0
41  bit PWM_G @PORTC.1;   // sortie commande vitesse moteur gauche RC1
42  bit PWM_D @PORTC.2;   // sortie commande vitesse moteur droit  RC2
43  bit AV_AR_G @PORTC.3; // sortie sens moteur gauche RC3
44  bit CED_NUM @PORTC.4; // entrée numérique capteur externe droit RC4
45  bit CID_NUM @PORTC.5; // entrée numérique capteur interne droit RC5
46  bit CIG_NUM @PORTC.6; // entrée numérique capteur interne gauche RC6
47  bit CEG_NUM @PORTC.7; // entrée numérique capteur externe gauche RC7
48  // TRISC=0b11110000
49
50  bit LEDV_D @PORTD.0;  // sortie led verte droite RD0
51  bit LEDJ_D @PORTD.1;  // sortie led jaune droite RD1
52  bit LEDR_D @PORTD.2;  // sortie led rouge droite RD2
53  bit LEDC_D @PORTD.3;  // sortie led centrale rouge droite DR3
54  bit LEDC_G @PORTD.4;  // sortie led centrale rouge gauche RD4
55  bit LEDR_G @PORTD.5;  // sortie led rouge gauche RD5
56  bit LEDJ_G @PORTD.6;  // sortie led jaune gauche RD6
57  bit LEDV_G @PORTD.7;  // sortie led verte gauche RD7
58  //TRISD=0b00000000
59
60  //TRISE=0b000 // broches RE0 RE1 RE2 non cablées ---> en out
61
62  // ----------------- déclaration des variables et constantes------------
63
64  bit obstacle;
65  uns8 vitesse;
66  uns8 valD;
67  uns8 valG;
68
69  // ------------------déclaration des ss-programmes----------------------
70
71  void conversion_AN (void);
72
```

```
73
74
75  // ----------vecteur d'interruption ( doit etre la première fonction déclarée ) --------------
76  //
77  // -------------- emission salves d'ultrasons et detection d'obstacles -----------------------
78  //
79  // Lorsque le timer 0 déborde le flag T0IF de débordement à 1 --> remise à 0 de T0IF, puis toggle
80  // sur "EN" l'enable des portes logique qui activent la sortie du 40,6 khz. Si EN est actif ( à 1 ),
81  // le timer est rechargé à 220 pour générer une courte salve 40,6 khz pendant EN à 1. Si EN est à 0
82  // le timer est rechargé normalement et on récupere le flag INTF qui marque la detection du signal
83  // de retour US sur RB0. INTF a peu de risque d'etre pris en compte par écho hors de la distance de
84  // sensibilité grace à la durée controlée de la salve ). La copie de INTF sur le bit obstacle arrete
85  // le robot avec clignotement orange quand INTF à 1. Remise à zero du flag INTF. Mais si un obstacle
86  // encore présent le robot reste bloqué par repassage rapide à 1 d'INTF.
87
88  interrupt ServerX (void)
89  {
90      int_save_registers ;
91       GIE = 0;                                  //interdit toutes les interrutions
92       if ( T0IF )
93       {  T0IF = 0;
94          EN = !EN;
95          if ( EN) TMR0 = 220 ;
96          else {obstacle = INTF;INTF = 0;}
97       }
98       GIE = 1;                                 // autorisation d'interruption avant de sortir
99      int_restore_registers;
100 }
101
102 // --------utilisation du sous programme LCD ( avec interruption doit être apres le vecteur int )
103
104 #include "LCD.c"
105
106 // ---------------- programme principal-------------------------------------------------
107
108 void main (void)
109 {
110     ADCON0= 0b10000001; // initialisation du convertisseur et selecteur de mesure initial sur boutons
        poussoirs
111     ADCON1=4;           //
112     T1CON = 49 ;        // timer 1 prescaler max à 8
113     T2CON= 6;           // TIMER2 sur ON , prescaler à 4
114     PR2= 255;           // avec prescaler à 4 et PR2 à 255 ==> 1.22 KHZ FRéquence PWM
115     CCPR1L = 0;         // Temps ON de la sortie à 0, sortie à 0   ( initialisation forçée )
116     CCPR2L= 0 ;         // Temps ON  de la sortie à 0, sortie à 0 ( initialisation forçée )
117     CCP1CON = 12;
118     CCP2CON = 12;
119     OPTION=7;           // RBPU option_reg à 0 pour activer les pull up et prescaler TIMER0 = 256
120     TRISA=0b001011;         //portA
121     TRISB=0b10000001;           //portB
122     TRISC=0b11110000;           //portC
123     TRISD=0b00000000;           //portD
124     TRISE=0b000;            //portE
125     PORTA = 0;
126     PORTB = 0;
127     PORTC = 0;
128     PORTD = 0;
129     PORTE = 0;
130     INTCON = 160; // 0b10100000 , GIE= 1, T0IE = 1 // autorisation interruption générale + débordement TIMER0
131
132
133         //---------temporisation de démarrage du robot ------------------
134         //             2 pauses de 1,2 secondes
135         //                fonction de LCD.C
136
137                      pausemillisecondes(1200);
138                      pausemillisecondes(1200);
139
140         //-------------------------------------------------------------
141
142         vitesse = 0; // remise à 0 de vitesse avant utilisation dans la boucle sans fin
143
```

```
144
145
146         for ( ; ; )
147         {
148             if (obstacle)
149             {  CCPR1L = 0; CCPR2L = 0 ;     // si obstacle vrai le robot s'arrete
150
151                LEDJ_D = LEDJ_G = TMR1H.7;   // --tempo non blocante avec TIMER1---
152                                             // timer 1 compte en boucle , TMR1L (8bits) alimente TMR1H (8bits).
153             }                               // quand le bit 7 de TMR1H passe à 1,Led jaune gauche et led jaune
154                                             // droite allumées. TMR1H à 0 leds eteintes ->> clignotement sur
155                                             // obstacle environ 10 hertz
156
157             else
158             {
159                   conversion_AN ( );
160                   valD = ADRESH + 30;       // +30 à la valeur de conversion pour approcher le max 255
161                   if ( valD < 30 )          // ( capteurs autour de 210-220)
162                   { valD = 255;}            //si valD <30 c'est qu'on a fait le tour donc on est à 255 en fait
163                    if ( valD < 90 )         // en dessous de 30 moteur démarre pas, avec + 30 le mini est à 60
164                   { valD = 0;}              // par securité on estime qu'on alimente les moteurs qu'à 90,
165                                             // coupés à 0 sinon.
166                   ADCON0 = 0b10011001;      // commutation sur canal d'acquisition capteur gauche analogique
167                   pause20microsecondes();
168                   conversion_AN ( );
169                   ADCON0 = 0b10001001;// conversion terminée sur capteur gauche, on commute sur le droit
170                   valG = ADRESH + 40 ;// en prépa tour suivant. pas de pause, le temps de boucle suffit.
171                   if ( valG < 40 )          //+40 sur gauche au lieu +30 pour compenser différence
172                   { valG = 255;}            //de niveaux analogique capteurs
173                   if ( valG < 90 )
174                   { valG = 0;}
175
176       //
          ------------------------------------------------------------------------------------
177                 // alimentation du moteur opposé au capteur par copie
178                 // du registre valG ou D divisé par 2 pour pas aller trop
179                 // vite. Un seuil de 20 décale les vitesses vers le haut.
180
181               CCPR1L = ( valG>>1 ) + 20   ; CCPR2L = ( valD>>1 ) + 20 ;
182
183       //      ------------------------------------------------------------------------------------
184
185           capteur_droit = CID_NUM;      //copie de l'état du capteur interieur droit sur bit 1 de vitesse
186           capteur_gauche = CIG_NUM;     // copie de l'état du capteur interieur gauche sur bit 2 de vitesse
187
188           /*
189           //-------- correction de trajectoire du robot ------------------------------------------------
190           //
191           // l'évolution à chaque boucle de l'état des 4 bits de poids faible sur vitesse
192           // |0|0|0|0|flag_gauche|capteur_gauche|capteur_droit|flag_droit|
193           // fait évoluer les actions au sein du switch.
194           // ( flags actifs à 1 - capteurs actifs à 0 )
195           // Dans cette application, le case ne gère les moteur en numérique qu'en sortie extrême de ligne
196           // ( case 7 et 14 ). Dans les autres cas,  les moteurs sont gérés en analogique et le case ne sert
197           // qu'à gérer les flags pour permettre de retrouver la ligne,mémorisant d'où vient le robot
198           // et ceci même si il s'arrête hors ligne par detection d'obstacle.
199           // Dans les cas 7 et 14, le moteur opposé au moteur gérant le retour ligne est mis à zéro
200           // Son relai est ensuite inversé pour vider la bobine moteur et mieux le freiner pour tourner
201           // Le moteur opposé est enfin reglé à la vitesse retour ligne.
202           // Dans les autres cas que 7 et 14, les relais de sens sont mis à 0 ( marche avant ) pour éviter
203           // des inversions non souhaitées.
204
205
206
207
208
209
210
211
212
213
214
```

```
215
216
217           // ---------------------------------------------------------------------------------------------
218
219           //                                    [0|0|0|0] case 0  ( droit devant )
220           //          gauche  ------------------------|------------------------- droite
221           //                  |                  [0|1|1|0] case 6              |
222           //       case2  [0|0|1|0]                  stop                  [0|1|0|0] case 4
223           //  retour leger à droite (non actif )                                    retour leger à gauche
224           //                  |                   case 10                        |        ( non actif )
225           //       case3  [0|0|1|1] <-------------[1|0|1|0] <----------------[1|1|0|0] case 12
226           //  retour leger à droite     | le robot recupere la ligne |        retour leger à gauche
227           //  (non actif)     |         | sur capteur gauche mais le |           |   ( non actif )
228           //                  |         | flag gauche resté levé.    |           |
229           //                  |         |                            |           |
230           //                  |          -->------[1|1|0|1] >--------            |
231           //                  |           le robot récupère la ligne             |
232           //                  |           sur capteur droit mais le              |
233           //                  |           flag droit resté levé.                 |
234           //  case7  _________|_________                                _________|_________
235           //         |                |                                |                |
236           //     [0|1|1|1]         [0|0|0|1] case1             case8 [1|0|0|0]        [1|1|1|0] case14
237           //    flag droit levé    flag droit levé             flag gauche levé       flag gauche levé
238           //    retour fort        on baisse le flag           on baisse le flag      retour fort
239           //    à droite           droit devant                droit devant           à gauche
240           //                                    |             |
241           //                                    [0|0|0|0]    case0
242           //                                  droit devant
243           */
244
245                   switch (vitesse)
246                   {
247
248                     case 0:AV_AR_D = AV_AR_G = 0;flag_gauche = 0;flag_droit = 0; PORTD = 0b10000001;
249                     break;
250                     case 1:
251                     case 8: AV_AR_D = AV_AR_G = 0;flag_gauche = 0;flag_droit = 0; PORTD = 0b10000001;
252                     break;
253
254                     case 2:
255                     case 3:
256                     case 10: AV_AR_D = AV_AR_G = 0; flag_droit = 1;flag_gauche = 0; PORTD = 0b10000101;
257                     break;
258
259                     case 4:
260                     case 12:
261                     case 13: AV_AR_D = AV_AR_G = 0;flag_gauche = 1;flag_droit = 0; PORTD = 0b10100001;
262                     break;
263
264                     case 6: AV_AR_D = AV_AR_G = 0;CCPR1L = 0; CCPR2L = 0 ; flag_gauche = 0; flag_droit = 0;
265                             PORTD = 0b00111100;
266                     break;
267
268                     case 7: CCPR2L = 0; AV_AR_G = 1; AV_AR_D = 0;CCPR1L = 150; PORTD = 0b00100001;
269                     break;
270
271                     case 14:CCPR1L = 0; AV_AR_D = 1;  AV_AR_G = 0;CCPR2L = 150; PORTD = 0b10000100;
272                     break;
273
274                     default:
275                       }
276                 }
277               }
278     }
279
280      //---------sous programme conversion_AN ------------------------------
281
282             void conversion_AN (void)
283             { GO = 1; while ( GO );}
284
285
286
```

Programmation En Langage C

Enregistrement et restitution de parcours

```
1
2   // Programme d'enregistrement du suivi de ligne en numérique par capteurs de ligne centraux,
3   // avec flags gauche et droit , puis lecture-reproduction et
4   // recadencement par le robot des séquences roulées mémorisées.
5
6   // identification de la cible HARD
7
8   #define taille 35     // 35 cases mémoire max - limitation du compilateur CC5X
9   #define capteur_droit vitesse.1
10  #define capteur_gauche vitesse.2
11  #define flag_gauche vitesse.3
12  #define flag_droit vitesse.0
13  #pragma chip PIC16F877 // attention, PIC16F877.h dans dossier programmation robot
14
15  #include "int16cxx.h"  // ajoute la gestion du vecteur d' interruption
16
17  //#pragma origin 4   // origin 5 mini pour prog sans interruption ; origin 4 avec int
18  #pragma origin 4     // gestion interruption
19
20  // personnalisation de l'appellation des ports
21
22  bit ANA_BT @PORTA.0;  // entrée analogique des boutons poussoirs// RA0
23  bit CID_ANA @PORTA.1; // entrée capteur intérieur droit en analogique // RA1
24  bit EN @PORTA.2;      // sortie enable des portes logiques & de l'oscillateur 40,6 khz // RA2
25  bit CIG_ANA @PORTA.3; // entrée capteur intérieur gauche en analogique// RA3
26  //TRISA=0b001011      // broches RA4 RA5 non cablées ---> en out ob00----
27
28  bit RX_US @PORTB.0;   // récepteur ultrason RB0
29  bit CLK @PORTB.3;     // sortie CLK afficheur RB3
30  bit SID @PORTB.4;     // sortie SID afficheur RB4
31  bit CS @PORTB.5;      // sortie CS afficheur RB5
32  bit PGD @PORTB.7;     // entrée programmation ( pas d'autre usage ) RB7
33  //TRISB=0b10000001    // broches RB1 RB2 RB6 non cablées ---> en out
34
35  bit AV_AR_D @PORTC.0; // sortie sens moteur droit RC0
36  bit PWM_G @PORTC.1;   // sortie commande vitesse moteur gauche RC1
37  bit PWM_D @PORTC.2;   // sortie commande vitesse moteur droit  RC2
38  bit AV_AR_G @PORTC.3; // sortie sens moteur gauche RC3
39  bit CED_NUM @PORTC.4; // entrée numérique capteur externe droit RC4
40  bit CID_NUM @PORTC.5; // entrée numérique capteur interne droit RC5
41  bit CIG_NUM @PORTC.6; // entrée numérique capteur interne gauche RC6
42  bit CEG_NUM @PORTC.7; // entrée numérique capteur externe gauche RC7
43  // TRISC=0b11110000
44
45  bit LEDV_D @PORTD.0;  // sortie led verte droite RD0
46  bit LEDJ_D @PORTD.1;  // sortie led jaune droite RD1
47  bit LEDR_D @PORTD.2;  // sortie led rouge droite RD2
48  bit LEDC_D @PORTD.3;  // sortie led centrale rouge droite DR3
49  bit LEDC_G @PORTD.4;  // sortie led centrale rouge gauche RD4
50  bit LEDR_G @PORTD.5;  // sortie led rouge gauche RD5
51  bit LEDJ_G @PORTD.6;  // sortie led jaune gauche RD6
52  bit LEDV_G @PORTD.7;  // sortie led verte gauche RD7
53  //TRISD=0b00000000
54  //TRISE=0b000 // broches RE0 RE1 RE2 non cablées ---> en out
55
56  // déclaration des variables et constantes
57  uns8 vitesse;
58  uns8 tableau_capteurs [taille],indice;
59  uns8 tableau_temps    [taille], chrono;
60  uns8 batt,pression,bouton;
61  // vecteur d'interruption
62
63  interrupt ServerX (void)
64  {
65      int_save_registers ;          // incrémentation du compteur chrono quand le flag TMR1IF
66                                    // de debordement du TIMER1 passe à 1. Remise à 0 du flag.
67        chrono ++ ;
68        TMR1IF = 0;
69
70      int_restore_registers;
71  }
72
```

```
73
74  // déclaration des ss-programmes
75  void roulage ( uns8 );
76  void conversion_AN (void);
77  uns8 decodage_clavier (uns8);
78  // programme principal
79
80  void main (void)
81  {
82      ADCON0= 0b10000001; // initialisation du convertisseur pour AN0 boutons clavier
83      ADCON1=4;           //
84      T1CON = 49 ;        // timer 1 prescaler 8 max
85      T2CON= 6;           // TIMER2 sur ON , prescaler à 4
86      PR2= 255;            // avec prescaler à 4 et PR2 à 255 ==> 1.22 KHZ FRéquence PWM
87      CCPR1L = 0;          // Temps ON de la sortie à 0, sortie à 0   ( initialisation forçée )
88      CCPR2L= 0 ;          // Temps ON  de la sortie à 0, sortie à 0 ( initialisation forçée )
89      CCP1CON = 12;         //
90      CCP2CON = 12;          //
91      OPTION=7;        // RBPU option_reg à 0 pour activer les pull up et prescaler TIMER0 = 256
92      TRISA=0b001011;         //portA
93      TRISB=0b10000001;           //portB
94      TRISC=0b11110000;           //portC
95      TRISD=0b00000000;           //portD
96      TRISE=0b000;            //portE
97      PORTA = 0;
98      PORTB = 0;
99      PORTC = 0;
100     PORTD = 0;
101     PORTE = 0;
102
103     INTCON = 192; // GIE = PEIE = 1 // autorisation interruptions globale et périphériques
104     TMR1IE = 1 ; // Enable interruption timer 1
105
106     batt = 40 ; // initialisation de batt pour que la lecture bouton puisse etre reconnue ( pas de detection
        d'appuie sinon )
107
108         //---------temporisation de démarrage du robot ------------------
109         // boucle de 15 intégrant une tempo de 156ms (15 * 156ms = 2.34 sec )
110         // on utilise vitesse qui est libre à cet endroit du programme
111
112         uns16 tempo ;
113         for (vitesse = 15 ; vitesse ; -- vitesse )
114         {
115             for ( tempo = 65000; tempo ; -- tempo )
116             { };
117         }
118
119         //-------------------------------------------------------------------------------------
120         // indice des tableaux,compteur de durée de sequences de roulage, le flag de débordement
121         // du TIMER1 et les 2 registres L et H  du TIMER1 remis à 0.
122         // remise à 0 de vitesse avant utilisation
123
124         indice = chrono = TMR1IF = TMR1H = TMR1L = 0 ;
125         vitesse = 0;
126
127        //--------------  lecture capteurs +  flags ->    enregistrement -----------------------
128
129        // taille represente le nombre de cases max des tableaux capteurs et temps
130        // Tant que les indices incrémentés sont inférieurs à taille
131        // on reste dans la boucle While d'enregistrement de parcours ( sinon plantage si débordement).
132        // On lis les capteurs pour le sous programme de gestion de suivi de ligne.
133        // Un switch sur l'octet vitesse gère ce suivi. Des flags de perte de ligne sont gérés
134        // dans les "case". Ces flags sont mémorisés dans 2 bits de l'octet vitesse.
135        // Enregistrement des tableaux temps et vitesse au demarrage ( indice 0 )
136        // Ensuite, au tour suivant, si le nouvel octet vitesse est différent de l'octet
137        // du tableau à l'indice antérieur, on enregistre. Dans le cas d'une égalité ,
138        // on memorise rien ( exemple: roulage sur ligne, recup externe ). Quand taille atteinte,
139        // Mémoire pleine, on sort de la boucle. ARRET moteurs. toutes les leds du portD à 0
140
141
142
143
```

```
144
145             while (indice != taille )
146             {
147
148              capteur_droit = CID_NUM; //copie de l'état du capteur interieur droit sur bit 1 de vitesse
149              capteur_gauche = CIG_NUM; // copie de l'état du capteur interieur gauche sur bit 2 de vitesse
150
151              roulage ( vitesse );      // sous programme de suivi alimenté par l'octet "vitesse"
152
153                  if (( indice == 0) || ( vitesse != tableau_capteurs [indice-1] ) )
154                  {
155                       tableau_capteurs [indice] = vitesse;
156                       tableau_temps    [indice] = chrono;
157                       indice++;
158                  }
159             }
160
161              CCPR1L = 0; CCPR2L = 0 ;      // moteurs à 0
162
163              PORTD = 0b00000000;           // toutes les leds éteintes
164
165              // Une fois l'enregistrement terminé on rentre dans une nouvelle boucle, cette fois
166              // infinie.le bouton est remis à 0. Puis, tant que le bouton n'est pas égal à 4, on reste
167              // dans une sous boucle qui relance la conversion de mesure bouton et le sous programme de
168              // décodage de la valeur mesurée.
169              // Quand le bouton 4 est détecté, on sort de cette sous boucle test bouton.
170              // Une pause est marquée avant de lancer la boucle de lecture du parcours enregistré
171              // De nouveau, indice des tableaux,compteur de durée de sequences de roulage,
172              // le flag de débordement du TIMER1 et les 2 registres L et H  du TIMER1 remis à 0.
173              // Tant que l'indice n'est pas arrivé au max ( taille ) on reboucle la lecture.
174              // Pour recadencer le guidage du robot selon les temps mémorisés à chaque enregistrement
175              // successif de l'octet "vitesse", on autorise l'incrémentation de lecture de la mémoire
176              // tableau_capteurs que si il y a égalité entre l'état du compteur chrono incrémenté par TMR1IF
177              // et le temps mémorisé initialement lors de cet enregistrement dans tableau_temps.
178
179
180                     while(1){ bouton = 0;
181                               while ( bouton != 4)                        // sous boucle test bouton
182                                   {
183                                     conversion_AN ( );
184                                     bouton = decodage_clavier ( ADRESH ) ;
185                                   };
186
187                               uns16 tempo ;
188                               for (vitesse = 15 ; vitesse ; -- vitesse )  // tempo 2.34 sec avant lecture
189                               {
190                                 for ( tempo = 65000; tempo ; -- tempo )
191                                 { };
192                               }
193                               indice = chrono = TMR1IF = TMR1H = TMR1L = 0 ;
194                                  while (indice != taille )                        // boucle de lecture
195                                  {                                                // du parcours enregistré
196                                      if ( chrono  == tableau_temps [indice] )     // recadencement
197                                      {
198                                         vitesse = tableau_capteurs [indice];
199                                         roulage ( vitesse );   // la mémoire alimente le sous programme de
                                           roulage
200                                         indice++;
201                                      }
202                                  }
203                                          // quand l'indice max atteint ( taille ), sortie de boucle et arrêt
204                                          // moteurs , remise à 0 des flags. Retour à la mise à 0 de bouton.
205                                          // relecture d'état bouton.
206
207                                CCPR1L = 0; CCPR2L = 0 ; flag_gauche = 0; flag_droit = 0;
208
209                               };
210         }
211
212
213
214
```

```
215         // sous programme suivi de ligne robot ( enregistrement ) ou guidage robot ( lecture mémoire )
            -----------
216
217         void  roulage ( uns8 z )
218         {
219          /*
220         // l'évolution à chaque boucle de l'état des 4 bits de poids faible sur vitesse
221         // |0|0|0|0|flag_gauche|capteur_gauche|capteur_droit|flag_droit|
222         // fait évoluer les actions au sein du switch.
223         // flags actifs à 1 - capteurs actifs à 0
224         // -------------------------------------------------------
225
226         //                                  [0|0|0|0] case 0  ( droit devant )
227         //          gauche  ------------------------|-------------------------- droite
228         //                  |                  [0|1|1|0] case 6               |
229         //       case2  [0|0|1|0]                  stop                   [0|1|0|0] case 4
230         //  retour leger à droite                                          retour leger à gauche
231         //                  |                   case 10                          |
232         //       case3  [0|0|1|1] <-------------[1|0|1|0] <----------------[1|1|0|0] case 12
233         //  retour leger à droite     | le robot recupere la ligne |        retour leger à gauche
234         //                  |         | sur capteur gauche mais le |           |
235         //                  |         | flag gauche resté levé.    |           |
236         //                  |         |                            |           |
237         //                  |          -->------[1|1|0|1] >--------            |
238         //                  |           le robot récupère la ligne             |
239         //                  |           sur capteur droit mais le              |
240         //                  |           flag droit resté levé.                 |
241         //  case7  _________|_________                                ________|_________
242         //         |                |                                |                |
243         //     [0|1|1|1]         [0|0|0|1] case1             case8 [1|0|0|0]        [1|1|1|0] case14
244         //    flag droit levé    flag droit levé             flag gauche levé       flag gauche levé
245         //    retour fort        on baisse le flag           on baisse le flag      retour fort
246         //    à droite           droit devant                droit devant           à gauche
247         //                                    |             |
248         //                                  [0|0|0|0]    case0
249         //                                 droit devant
250         */
251
252         switch (z)
253         {
254             case 0:
255             case 1:
256             case 8:     AV_AR_D = AV_AR_G = 0; flag_gauche = 0;flag_droit = 0;
257                         CCPR1L = 80; CCPR2L = 80;PORTD = 0b01000010; break;
258
259             case 2:
260             case 3:
261             case 10:    AV_AR_D = AV_AR_G = 0;flag_gauche = 0;flag_droit = 1;
262                         CCPR1L = 150; CCPR2L = 60; PORTD = 0b10000101; break;
263
264             case 4:
265             case 12:
266             case 13:    AV_AR_D = AV_AR_G = 0;flag_gauche = 1;flag_droit = 0;
267                         CCPR1L = 60 ;CCPR2L = 150; PORTD = 0b10100001; break;
268
269             case 6:     AV_AR_D = AV_AR_G = 0; CCPR1L = 0; CCPR2L = 0 ;
270                         flag_gauche = 0; flag_droit = 0;PORTD = 0b00111100; break;
271
272             case 7:     CCPR2L = 0; AV_AR_G = 1; AV_AR_D = 0;CCPR1L = 170;
273                         PORTD = 0b00100001; break;
274
275             case 14:    CCPR1L = 0; AV_AR_D = 1;  AV_AR_G = 0;CCPR2L = 170;
276                         PORTD = 0b10000100; break;
277
278             default:
279         }
280        }
281
282
283
284
285
```

```
286
287         //---------sous programme conversion_AN -------------------------------
288
289          void conversion_AN (void)
290          {
291              GO = 1;
292              while ( GO );
293          }
294
295  // ----fonction de décodage clavier ----------------------------------------
296
297  // tests de la valeur de l'octet ADRESH rafraichi, à chaque tour de conversion, la valeur de référence
298  // de la batterie. (l'écart entre les valeurs numérique des bouton est légèrement glissant autour de 12,
299  // correspondant à un écart de seuil de 0.2 volts env ). Si le résultat du premier test est vrai on est sur
300  // que la tension batterie a peut etre bouger légèrement, mais qu'aucun bouton n'a été appuyé. Dans ce cas
301  // on rafraichi uniquement la valeur stockée de la réf batterie et on monte un flag pression de non appuie.
302  // On test ensuite ADRESH / à une succession d'écarts de 11 cumulés ( n * 0.2 volts ) pour déterminer quel
303  // bouton a été appuyé. Le résultat attribue le rang du bouton correspondant.
304  // ne pas oublier d' initialiser batt ( 40/exp ) pour que a ( ADRESH ) soit sup à batt - 6, sinon
305  // batt peut valoir /exp 249. Comme valeur max de ADRESH env 225 , bouton sera toujours à 0.
306
307
308          uns8 decodage_clavier (uns8 a)
309          {
310              if ( a > ( batt - 6 ) )    // valeur de seuil modifiée pour meilleur discrimination *
311              {
312                  batt = a;
313                  a = 0;
314                  pression = 1;
315              }
316
317              else
318              {
319                  if ( a > batt - 24 )                    // * (+4)
320                  { a = 1;}
321                  else
322                  {
323                      if ( a > batt - 32 )               //* (+4)
324                          { a = 2;}
325                      else
326                      {
327                          if ( a > batt - 43 )          //* (+4)
328                          { a = 3;}
329
330                          else
331                          { a = 4;}
332                      }
333                  }
334              }
335
336              return a ;
337          }
338
339
340
341
```

Lien Vidéo :

Video de suivi de ligne