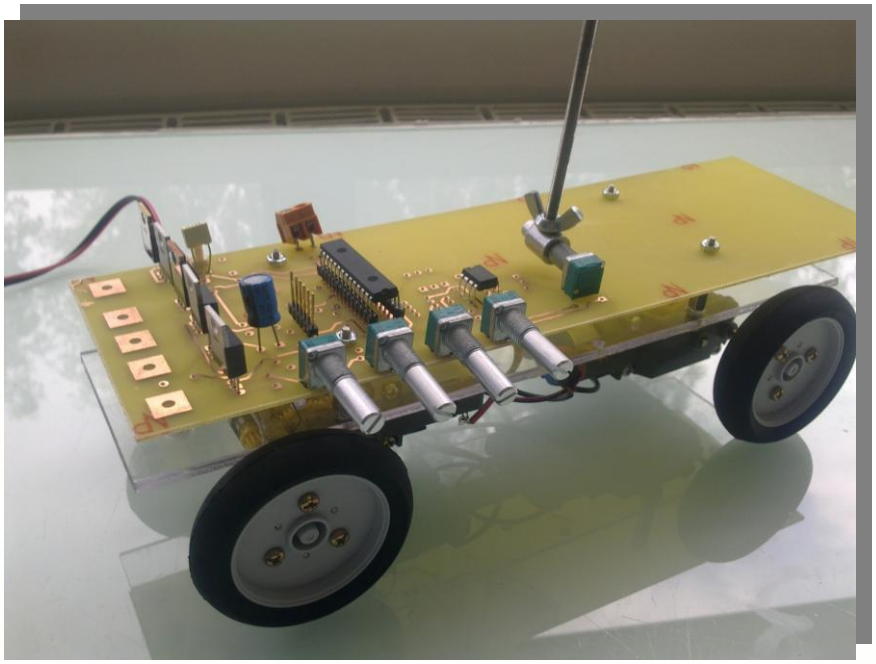
Projet de Physique P6-3

STPI/P6-3/2010 – 42

ROBOT MOBILE SUPPORTANT UN PENDULE INVERSE



Etudiants :

Julien BURGADE

Coralie FARGES

Matthew PERROTTE

Enseignant-responsable du projet :

Fabrice DELAMARE

Cette page est laissée intentionnellement vierge.

Date de remise du rapport : ***18/06/10***

Référence du projet : ***STPI/P6-3/2009 – 42***

Intitulé du projet : ***Robot mobile supportant un pendule inversé***

Type de projet : ***Robotique et électronique***

Objectifs du projet *(10 lignes maxi)* :

L'objectif de notre projet de créer un robot supportant un pendule inversé. C'est-à-dire que notre robot devra maintenir en équilibre un pendule en se déplaçant de manière rectiligne. Notre projet consiste en :

– créer un circuit imprimé : faire le schéma, faire la liste des empreintes des composants, placer les composants sur la plaque et agrandir les pistes et les pastilles.

– monter tous les composants et les moteurs : choisir les moteurs adaptés à notre robot et souder tous les composants sur la carte.

– créer un programme : établir la modélisation de notre robot supportant un pendule inversé et coder le programme.

Si existant, n° cahier de laboratoire associé : ***xxx***

INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE ROUEN
Département Sciences et Techniques Pour l'Ingénieur
BP 8 – place Emile Blondel - 76131 Mont-Saint-Aignan - tél : 33 2 35 52 83 00 - fax : 33 2 35 52 83 69

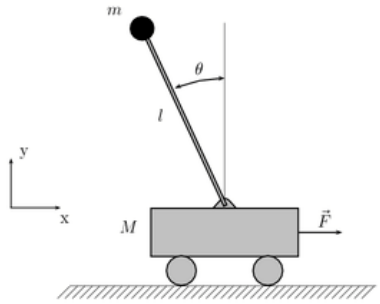
TABLE DES MATIERES

1. Introduction 6
2. Méthodologie / Organisation du travail 6
3. Travail réalisé et résultats 7
3.1. Aspect technique de notre robot 7
3.1.1. Contrôle du moteur 7
3.1.2. Régulateur de la tension 10
3.1.3. L'asservissement 13
3.1.4. Microcontrôleur 16
3.2. Programmation du PIC 18
3.3. Création du circuit imprimé 20
4. Conclusions et perspectives 23
5. Bibliographie 24
6. Annexes (non obligatoire) 25
6.1. Documentation technique 25
6.2. Listings des programmes réalisés 25
6.3. Schémas de montages, plans de conception… 37
6.4. Propositions de sujets de projets (en lien ou pas avec le projet réalisé) 38

NOTATIONS, ACRONYMES

- PID : Termes Proportionnel, Dérivatif et Intégratif qui vont nous permettre de résoudre les équations différentielles liées aux mouvements du pendule.
- PIC : Microcontrôleur de la société Microchip. C'est une unité de traitement de l'information.
- ALIEN : Asservissement Linéaire Echantillonné Numérique

1. INTRODUCTION



Notre projet a pour but de créer un robot mobile supportant un pendule inversé. Son principe est d'empêcher le pendule de tomber en se déplaçant de manière rectiligne, de façon à compenser les mouvements du pendule et le maintenir vertical après chaque perturbation.

Le pendule inversé est semblable au pendule simple, c'est-à-dire qu'il est constitué d'une masse m située à l'extrémité d'une tige rigide de masse négligeable par rapport à m. La tige est fixée en un point sur un support. A la différence du pendule simple, le système est inversé : le pendule inversée présente un équilibre instable à la verticale, c'est-à-dire la moindre perturbation va tendre à le faire tomber.

Notre projet passe aussi bien par de la physique, que par de l'informatique et de l'électronique. Au niveau physique, nous avons calculé l'asservissement linéaire échantillonné analogique le plus approprié. Au niveau informatique, nous avons mis en place un programme permettant de résoudre les équations différentielles dont nous avions besoin, de régler le sens et la vitesse des moteurs afin de maintenir en équilibre le pendule et de tenir compte de l'asservissement. Au niveau électronique, nous avons créé tout le circuit imprimé de notre robot, ainsi que monter les moteurs, le pendule, et tout ce qui le constitue.

Dans une première partie, nous allons expliquer la méthodologie et l'organisation du travail. Puis dans une seconde partie, nous allons exposer les aspects techniques de notre robot, c'est-à-dire le contrôle des moteurs, le rôle des régulateurs, l'asservissement, le microcontrôleur. Ensuite, nous allons parler de la programmation et enfin de la conception du circuit imprimé.

2. METHODOLOGIE / ORGANISATION DU TRAVAIL

Le robot supportant un pendule inversé étant un projet déjà existant et ayant été réalisé par Microchip sous le nom de AN964, nous avons tenté de prendre modèle sur l'AN964 pour réaliser notre propre robot. Cependant, il nous manque les bases essentiels pour maîtriser la création du circuit imprimé, le comprendre etc. M. Delamare nous a fourni les informations nécessaires à la compréhension du fonctionnement du circuit imprimé mais nous sommes quand même restreints aux contraintes imposées par le projet de Microchip.

Pour réaliser ce projet, nous ne sommes que 3 alors que la charge de travail exigée est assez importante. Nous avons divisé la conception de ce projet en 3 branches principales : la création du circuit imprimé, la programmation du PIC, le montage du robot.

Face à l'évolution du projet et aux nombreuses difficultés rencontrées, chacun a touché à toutes les parties afin de se focaliser sur les points qui faisaient stagner l'état d'avancement de notre projet, notamment la réalisation du circuit imprimé qui s'est avérée être une tache beaucoup plus difficile que nous l'avions prévu. Ceci pour plusieurs raisons : difficultés à utiliser le logiciel Kicad, changements réguliers de composants (PIC) ou encore la suppression des drivers de Mosfet.

Finalement, nous pouvons dire que tant que le plan de la répartition des taches que du planning, nous avons été surpris par le retard pris sur la création du circuit imprimé.

3. TRAVAIL REALISE ET RESULTATS

3.1. Aspect technique de notre robot

3.1.1. Contrôle du moteur

Contrôle du moteur avec un transistor

Avec un transistor on peut contrôler le moteur grâce au schéma suivant. La diode de roue libre permet la continuité du courant en dehors des phases d'alimentation. Le transistor a un rôle d'interrupteur.

Si IN =0, le moteur n'est pas alimenté, le robot n'avance pas.

Si IN=1, le moteur est alimenté, le robot avance.

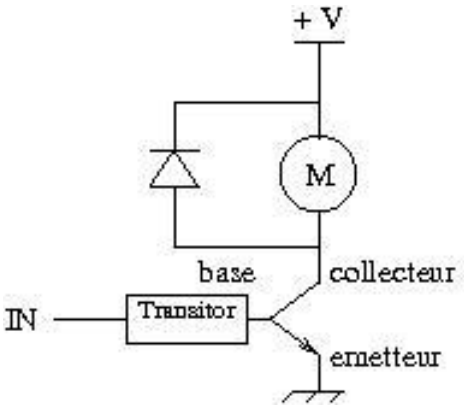


figure 1 : Schéma du moteur avec un transistor.

Contrôle du moteur avec un mosfet.

Le mosfet permet de moduler la vitesse. Il permettra au moteur de tourner les roues plus vite ou moins vite.

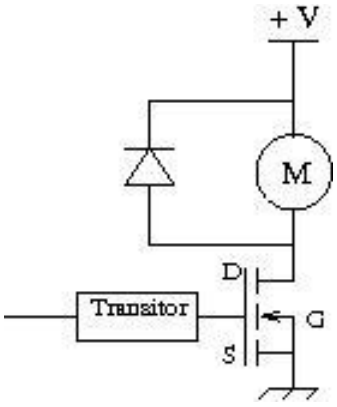


figure 2 : *Schéma du moteur avec un mosfet.*

Contrôle du moteur avec un pont en H.

Le pont en H permet de changer la polarisation du moteur ce qui fera avancer ou reculer le robot. De plus, nous pouvons aussi faire freiner le robot, en court-circuitant les bornes du moteur.

Le schéma rouge et le schéma bleu font rouler les roues dans un sens et dans l'ordre sens. Le schéma vert fait freiner le robot.

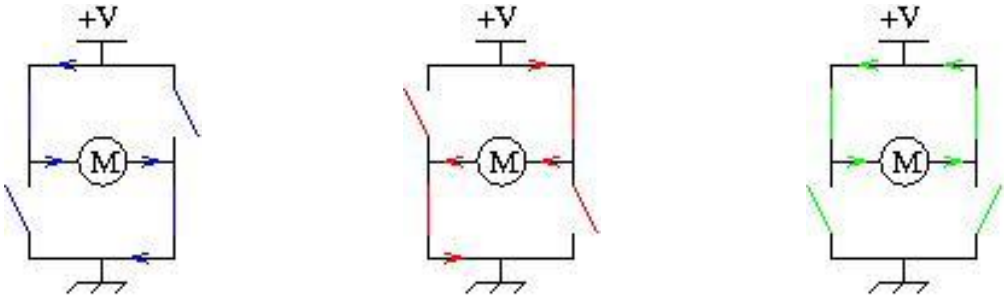


figure 3 : *Schémas de moteur avec un pont en H où seuls les interrupteurs varient.*

Notre projet.

Dans notre projet nous utilisons des diodes. Il y aussi le système de pont en H cependant à la place des interrupteur il y a des mosfets.

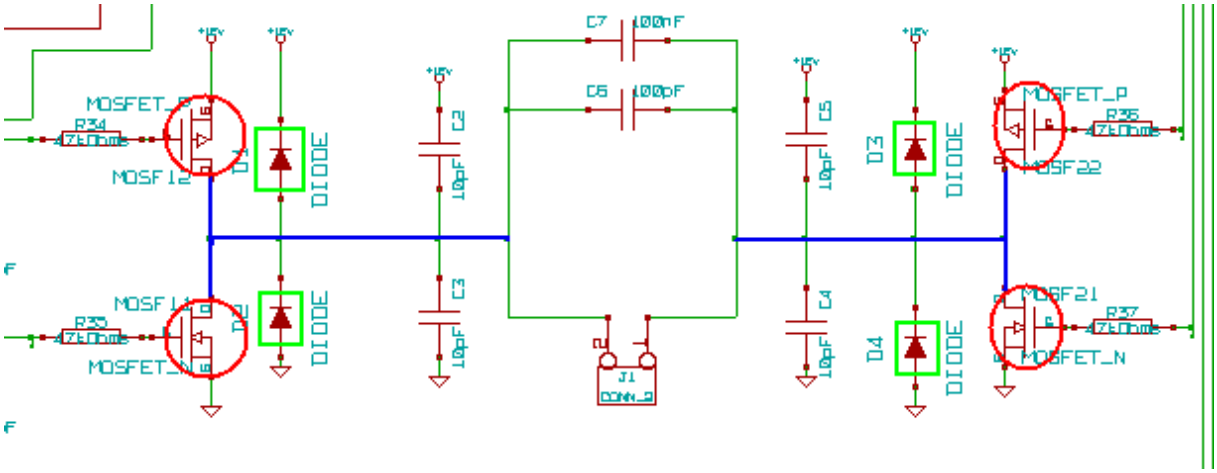


figure 4 : Schéma du moteur de notre robot sous kicad.

Comparaison de notre robot et l'AN964

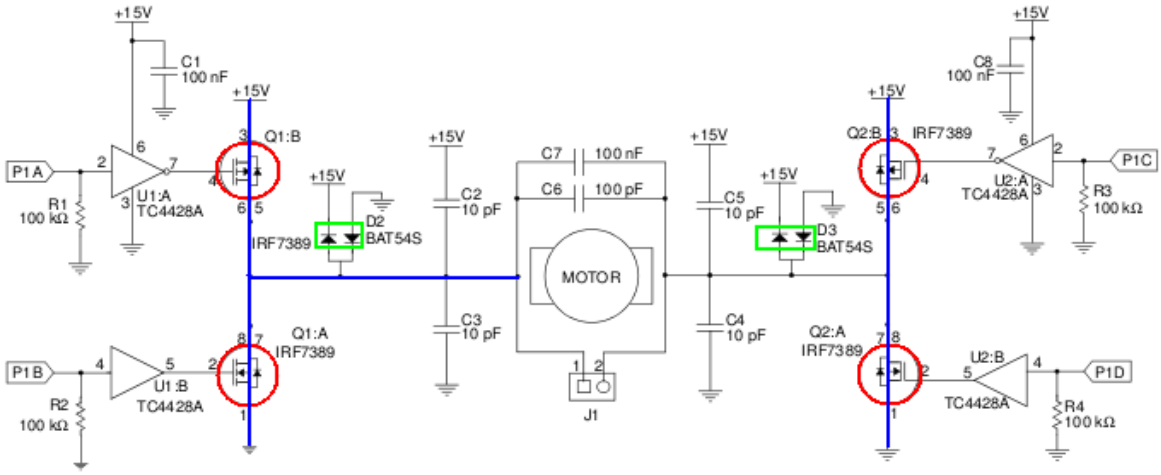


figure 5 : *Schéma du moteur de l'AN964.*

On voit bien avec les schémas que nous avons enlevé les drivers de mosfets car nous n'en avions pas à disposition. Nous les avons remplacés par des résistances.

Nous avons positionné les diodes droites, ceci n'a aucune influence, c'est juste esthétique.

Dans l'AN964, le robot tourne autour d'un axe fixe, pour facilité la construction, nous avons décidé de le mettre sur un charriot.

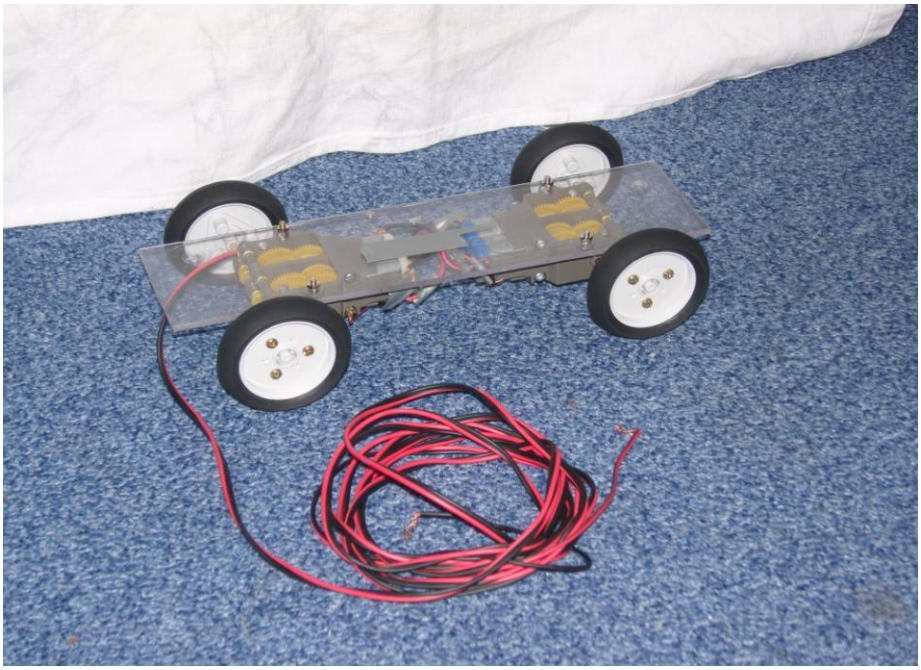


figure 6 : Charriot du robot avec les 4 moteurs.

3.1.2. Régulateur de la tension

Les régulateurs de tension permettent de stabiliser la tension à une valeur fixe. La tension ainsi ne fluctue pas. Il peut être composé d'un ensemble de composants classiques (diode, résistance, conducteur, ...) ou intégrés. Les régulateurs peuvent être fixes ou ajustables. Dans notre cas, il est intégré et fixe, nous nous sommes donc limité au régulateur fixe.

Régulateur fixe.

Les régulateurs fixes sont conçus pour délivrer une tension continue d'une valeur donnée qui reste toujours la même pour un régulateur. Prenons un exemple simple :

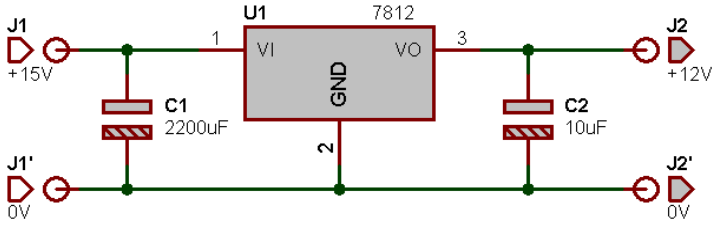


figure 7 : *Schéma exemple d'un régulateur fixe.*

Dans notre exemple le régulateur est de type 7812, c'est un régulateur positif de sortie 12V. On peut remarquer que deux condensateurs (C1 et C2) sont placés de chaque côté du régulateur. C1 assure le filtrage de la tension d'entrée, il permet d'obtenir une tension qui ressemble plus à du continu qu'à de l'alternatif. C2, lui n'est pas obligatoire, cependant il est conseillé de le mettre pour éviter tout risque d'oscillation parasite du régulateur.

Il faut faire attention, le brochage des régulateurs fixes positifs et négatifs n'est pas le même.

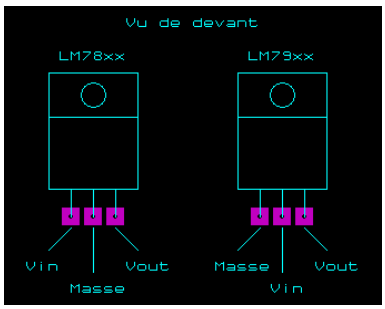


figure 8 : *Brochage du LM78xx (positif) et du LM79xx (négatif).*

Dans la pratique.

Les fabricants conseillent souvent de mettre quatre condensateurs comme le montre le schéma suivant.

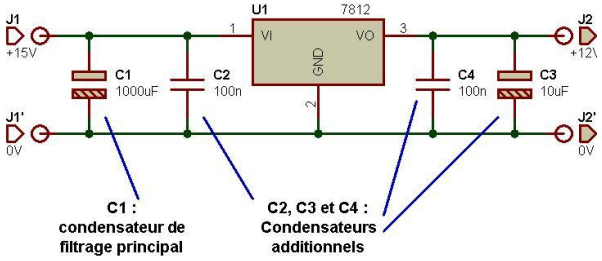


figure 9 : Schéma d'un régulateur dans la pratique.

Le condensateur C1 est le condensateur de filtrage principal, comme nous l'avons dit précédemment, permet de "lisser" les arches de sinusoïdes pour en obtenir une tension à peu près "droite".

Le condensateur C2 est placé en parallèle à C1, il doit être au plus proche du régulateur. Il permet d'améliorer la stabilité du régulateur et permet une meilleure réponse aux transitoires (brefs et importants appels de courant).

Le condensateur C3 a un rôle de réservoir d'énergie pour le circuit électronique qui tire profit de l'alimentation régulée. En cas de manque d'énergie, ce condensateur se décharge pour compenser. Il n'est pas rare de voir un condensateur en parallèle de C3 (différent de C4), il permet de limiter les bruits de haute fréquence.

Le condensateur C4 est facultatif. Il joue un rôle similaire à C2 en sortie : il améliore la stabilité du régulateur et permet une meilleure réponse aux transitoires.

Les 3 derniers condensateurs sont appelés des condensateurs additionnels.

Amélioration

Afin d'améliorer notre régulateur, il est conseillé d'ajouter des résistances de faibles valeurs en sortie aux condensateurs (cf. Dans la pratique). Cet assemblage permet de créer un circuit d'amortissement. Ce système s'oppose aux variations et permet d'avoir un courant plus ''propre''.

Notre projet

Dans notre projet, nous utilisons les condensateurs et les résistances pour former un circuit d'amortissement. On peut voir sur le schéma le régulateur, le condensateur de filtrage principal, les condensateurs additionnels de stabilité, le condensateur réservoir, le condensateur contre le bruit de hautes fréquences et les résistances d'amélioration.

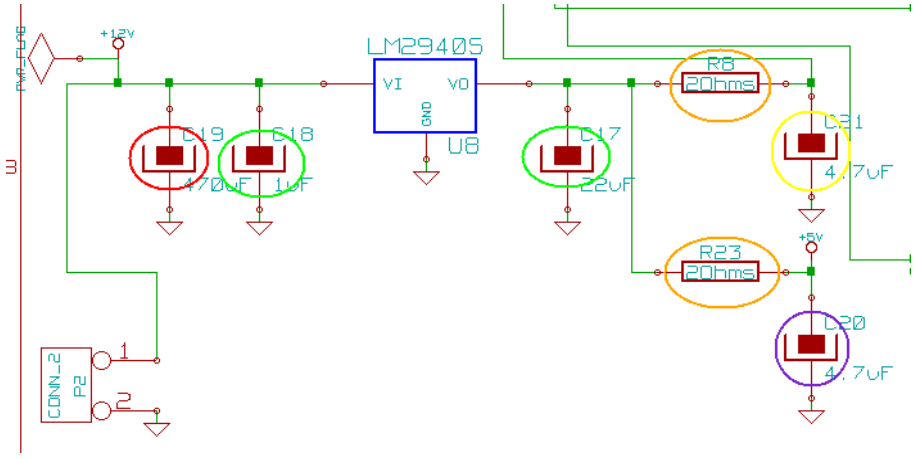


figure 10 : Schéma du régulateur de notre robot sous Kicad.

3.1.3. L'asservissement.

Les ALIENs (Asservissements Linéaires Échantillonnés Numériques)

Un asservissement linéaire échantillonné numérique est la mise en œuvre d'un dispositif informatisé ou micro-informatisé permettant de contrôler automatiquement les paramètres de fonctionnement d'un système physique, de manière à optimiser son comportement et maintenir ses performances quand les conditions de son environnement évoluent. Par exemple, un thermostat est un ALIEN : c'est un dispositif qui permet de contrôler la température dans une pièce.

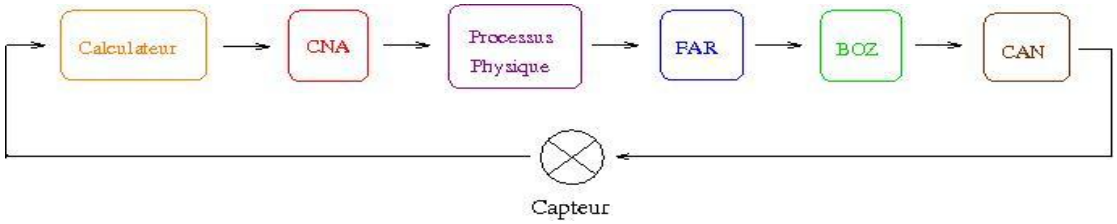


figure 11 : Schéma d'un ALIENs.

- Le Calculateur peut-être un ordinateur ou un régulateur numérique. Le programme exécuté permettra d'assurer la meilleure stabilité, précision et rapidité.
- Le Convertisseur Numérique Analogique permet de transformer une série de valeurs numériques exprimées en binaire en une tension proportionnelle.
- Le Processus analogique est le système physique que l'on veut automatiser.
- Le Filtre Anti-Repliement permet d'éliminer le bruit c'est-à-dire les signaux trop élevés qui perturberaient le fonctionnement du système.
- Le Bloqueur de d'Ordre Zéro maintient constantes les valeurs pendant tout une période d'horloge : on a un signal continu.
- Le Convertisseur Analogique Numérique permet de transformer un signal analogique continu en une suite discrétisée de valeurs binaires représentatives des valeurs analogiques

Reprenons notre exemple du thermostat :

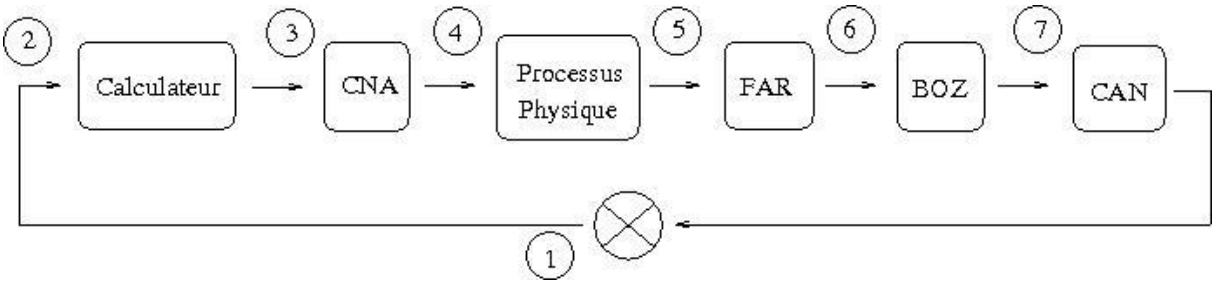


figure 12 : Schéma d'un ALIENs.

1. Le Capteur va détecter la température de la pièce et envoyer un signal au Calculateur.

2. Le Calculateur calcule la différence entre la température voulue et celle de la pièce et transforme cette information en instructions pour le chauffage.

3. Cette information est convertie en analogique pour que le chauffage la comprenne.

4. Le chauffage réagit aux instructions.

5. Le FAR élimine le bruit du signal.

6. Le BOZ évite les discontinuités.

7. Le signal analogique est converti en numérique et est reçu par le Capteur.

Corrections des ALIENs

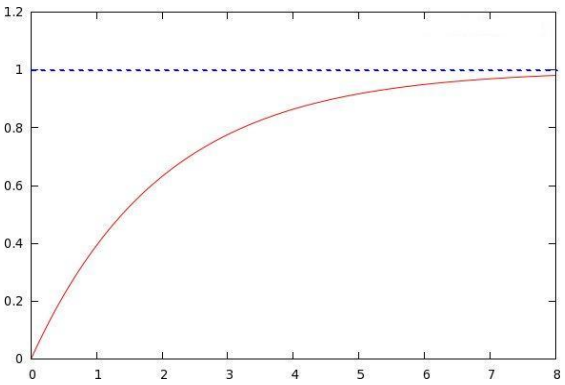


figure 13 : Réponse sans correcteurs

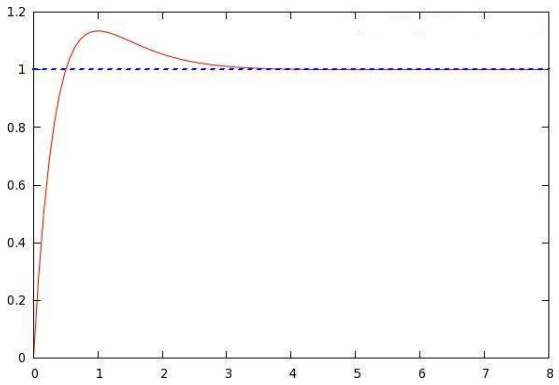


figure 14 : Réponse avec correcteurs

Comme on le voit sur les schémas la correction des ALIENs permet d'avoir une réponse plus rapide et précise. Il existe différentes corrections :

– *La correction proportionnelle et intégrale (PI):*

Le correcteur PI permet d'améliorer la précision de l'ALIEN. Il résulte de l'équation suivante :

Equ1 : $E_c(t)=K_P E(t)+K_I \int E(t)dt$

– *La correction proportionnelle et dérivée (PD):*

Le correcteur PD permet d'améliorer la précision de l'ALIEN. Il résulte de l'équation suivante:

Equ2 : $E_C(t)=K_P(t)+K_D\frac{dE(t)}{dt}$

– *La correction proportionnelle, intégrale et dérivée (PID):*

Le correcteur PID permet d'améliorer la précision de l'ALIEN. Il résulte de l'équation suivante:

Equ3 : $E_C(t)=K_P(t)+K_I\int E(t)dt+K_D\frac{dE(t)}{dt}$

Le correcteur P est le plus simple qui soit. Il permet d'appliquer une correction proportionnelle à l'erreur corrigeant de manière instantanée tout écart de la grandeur de sortie. Son rôle est d'amplifier l'erreur virtuellement pour que le système réagisse plus vivement, cependant le système perd en stabilité.

Une intégrale est une surface, le terme I va nous permettre de savoir ce qui s'est passé avant le moment présent. Le coefficient I permet de tenir compte du passé. Il compense l'erreur statistique et augmente la précision en régime permanent. L'idée est d'intégrer l'erreur depuis le début et d'ajouter cette erreur à la consigne : lorsqu'on se rapproche de la valeur demandée, l'erreur devient de plus en plus faible. Le terme proportionnel n'agit plus mais le terme de l'intégrale subsiste et reste stable. Cependant, un terme intégral trop important peut entraîner des dépassements de la consigne et une stabilisation lente voir des oscillations divergentes.

Le terme D provient de la dérivée de l'erreur. Il permet donc d'anticiper l'erreur : il représente le futur. Son action va dépendre du signe et de la vitesse de variation de l'erreur, et sera opposée à l'action proportionnelle. Ce correcteur devient prépondérant aux abords de la valeur demandée lorsque l'erreur devient faible, que l'action du terme proportionnel s'affaiblit et que l'intégrale varie peu : le terme D freine alors le système en limitant le dépassement de la consigne et en diminuant le temps de stabilisation.

En résumé :

Coefficient	Temps de montée	Temps de stabilisation	Dépassement	Erreur statistique
K_P	diminue	augmente	augmente	diminue
K_I	diminue	augmente	augmente	annule
K_D	-	diminue	diminue	-

Intérêt de notre robot.

Pour notre robot nous utilisons le PID afin d'équilibrer le pendule rapidement suivant la consigne (angle de 90° par rapport au sol). Il existe des potentiomètres qui permettent de régler le PID et de voir l'influence de chaque correcteur. On pourra ainsi ne mettre que le correcteur I, ou le D, ou le P et ajouter les autres par étapes pour voir les conséquences. On pourra aussi mettre des correcteurs forts et voir ainsi les états instables etc. ...

3.1.4. Microcontrôleur

L'intelligence de notre robot résidera dans son microcontrôleur qui est en fait le lieu de résidence du programme. Il va recevoir des données (par exemple : l'angle du pendule) et réagir en conséquence en envoyant des ordres (par exemple : faire fonctionner le moteur dans un sens ou dans l'autre.).

On retrouve le microcontrôleur dans beaucoup de cartes électroniques. Il est caractérisé par une petite taille et une consommation électrique faible quand il est en activité et extrêmement faible quand il est en sommeil. Il existe de nombreuses applications commerciales qui utilisent ce composant, tels que les téléphones portables, les cartes de crédit, les robots ménagers, les télés, radios, lecteurs DVD, MP3 et plein d'autres.

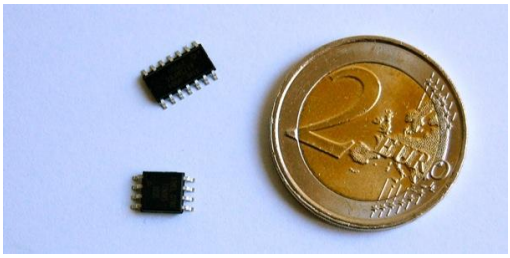


figure 15 : *Taille d'un microcontrôleur*

Ce schéma simplifié représente la structure d'un microcontrôleur. On y trouve :

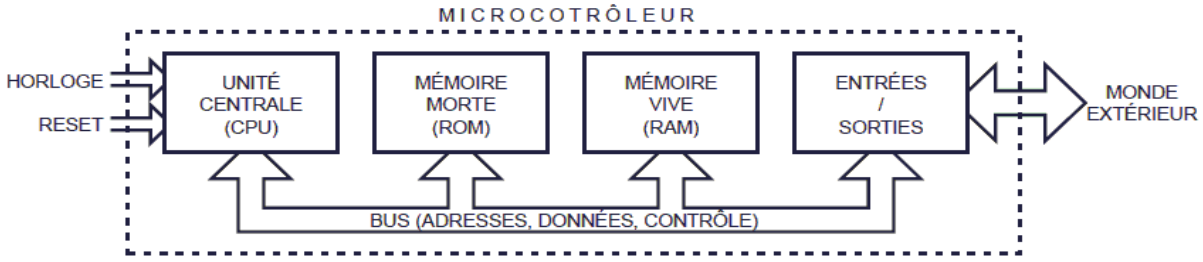


figure 16 : *Schéma explicatif du microcontrôleur*

– l'Unité Centrale de Traitement (Central Processing Unit) qui est en fait un microprocesseur, de la mémoire non volatile (ROM, flash,…) pour stocker le programme à exécuter,

– de la mémoire volatile (RAM) pour stocker les données modifiables courantes sur lesquelles le programme effectue des opérations,

– des circuits d'entrés/sorties (E/S) permettant au µC de recevoir par les entrées des informations de son environnement (capteur, clavier, souris, joystick,….) et de produire sur ses sorties des informations ou commande à destination de son environnement (écran, voyants, pré-actionneurs, actionneurs,…). Parmi les E/S, il existe de nombreux type de standard normalisé de communication par des bus, réseaux ou liaisons (RS232, I2C, SPI, USB, Ethernet, CAN,…)

Pour notre robot, nous devions utiliser un microcontrôleur PIC16F684 mais à défaut de disponibilité chez le fabricant, nous nous sommes rabattus sur un PIC16F676. Nous avons fait un comparatif des deux PICs et sommes arrivés à la conclusion qu'il était très similaire.

Caractéristiques	**PIC16F876**	**PIC16F684**
Nombre de pin	28	14
Operating speed	20MHz	20MHz
Program Memory (flash)	8K (14 bit words)	2048 (words)
Data Memory	368 Bytes	128 Bytes
EEPROM Data Memory	256 Bytes	256 Bytes
10 bit Analog to Digital Module	5 input channels	8 input channels

Caractéristiques	PIC16F876	PIC16F684
I/O Ports	22 (Port A,B,C)	12
Comparators	2	2
Timers	2/1 (8/16 bits)	2/1 (8/16 bits
Low Power Features (standby current)	< 1µA typical	50 nA @ 2,0V typica
Low Power Features (Operating current 4MHz)	< 2 mA typical @ 5V	220 µA @ 2V typical
Low Power Features (Operating current 32MHz)	20 µA typical @ 3V	11 µA @ 2V typical
ICSP	Via 2 pins	Via 2 pins
Wide Operating Voltage Range	2,0 V to 5,5 V	2,0 V to 5,5 V

3.2. Programmation du PIC

Pour la programmation du PIC, nous avons travaillé sur l'interface MPLAB. Le langage de programmation est le C.

Pour commencer, nous avons utilisé une carte électronique fournie par notre professeur, contenant entre autres, 4 leds, un pic16f877, un potentiomètre, un buzzer, un écran LCD.... Cette carte nous a permis de nous entrainer à programmer un pic, et de réaliser une simulation de notre futur robot.

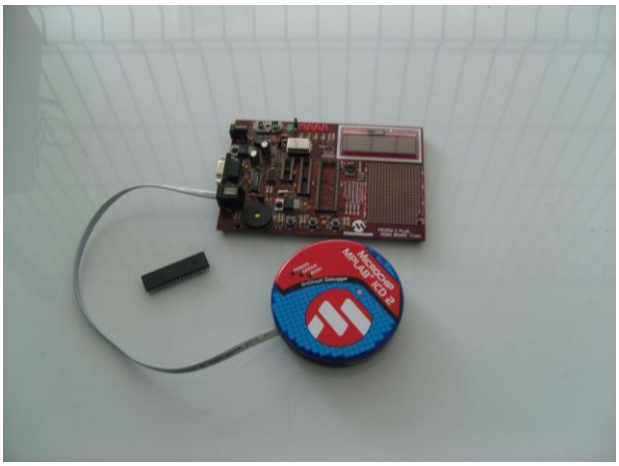


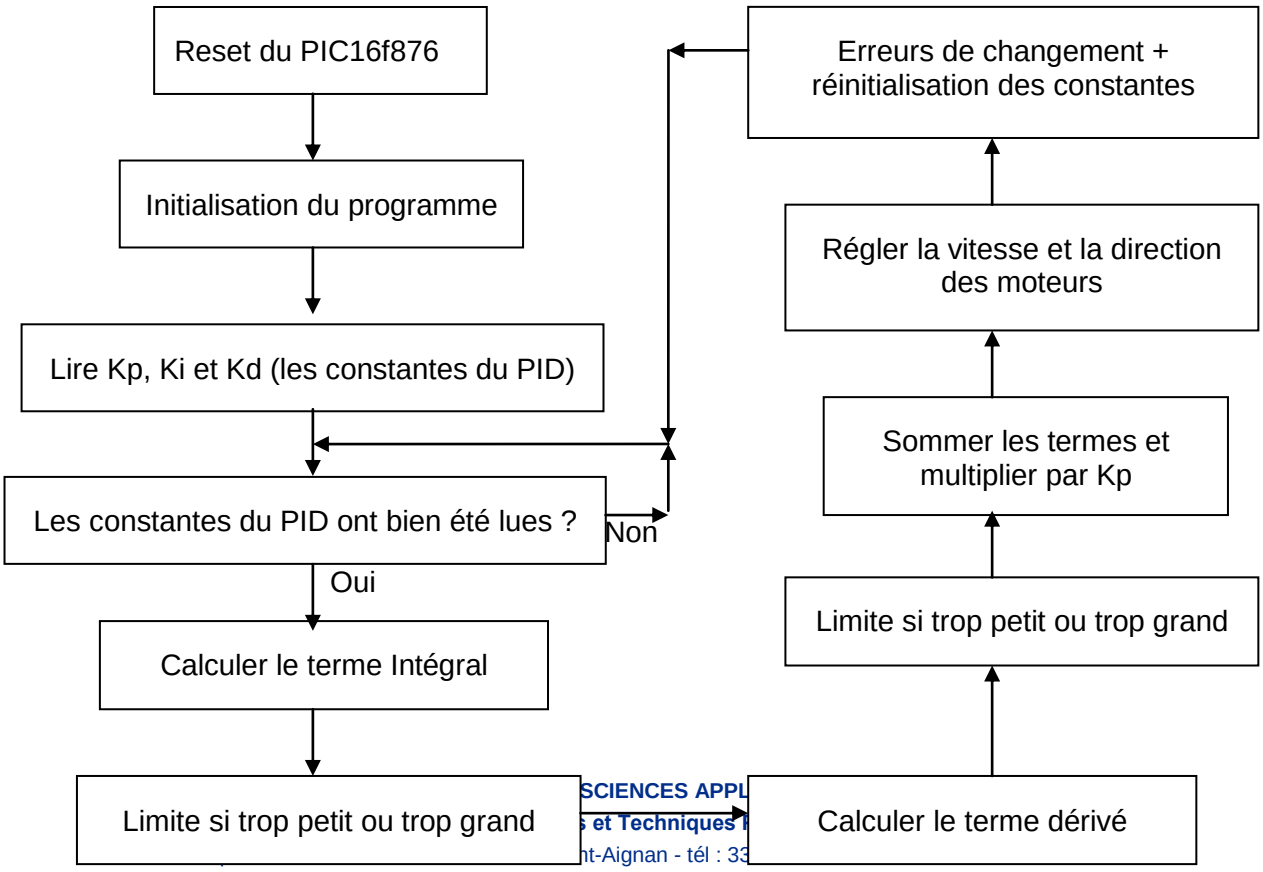
figure 17 : *Carte électronique fournie par notre professeur.*

Pour ce faire, nous avons simulé le moteur par les leds, et le capteur d'angle par le potentiomètre. Le principe était le suivant : nous avons créé un programme qui faisait apparaitre un chenillard avec les leds. Le sens du moteur était représenté par le sens du chenillard (de gauche à droite ou de droite à gauche) et la vitesse du moteur par la vitesse du chenillard. Lorsque nous tournions le potentiomètre de la carte, notre programme récupérait la valeur qu'il renvoyait. En fonction de celle-ci, il déterminait le sens du chenillard (c'est-à-dire le sens du moteur) ainsi que sa vitesse (équivalente pour notre robot à la vitesse du moteur).

Mais ceci ne constituait qu'une simulation. En effet, il nous fallait adapter notre programme d'une part à notre PIC (les entrées et sorties de notre PIC étaient différentes de celles du PIC de la carte fournie), et d'autre part à notre robot (en particulier au principe du PID). En ce qui concerne le PID (Proportionnel, Dérivatif, Intégratif), nous avons amélioré notre programme précédent (toujours pour la simulation) en permettant de rentrer au clavier de l'ordinateur les valeurs des termes proportionnel, dérivatif et intégratif. Pour notre robot, ces termes seront réglés grâce à des potentiomètres.

Le programme que nous avons créé fonctionne en théorie (la simulation donne des résultats très satisfaisant), mais nous n'avons pas pu le tester, car notre circuit imprimé n'a pas pu être réalisé à temps.

Voici toutes les étapes par lesquelles passe le programme afin de donner les ordres nécessaire au maintient en équilibre du pendule :



3.3. Création du circuit imprimé

La réalisation du circuit imprimé se déroule en 4 étapes principales :

- élaboration de la nomenclature des composants, du circuit imprimé sur papier...
- insertion des composants et création du schéma du circuit imprimé sur ordinateur et EESchema.
- Attribution de l'empreinte de chaque composant avec CVpcb.
- Routage de la carte avec PCBnew

Ces 4 logiciels sont regroupés en un seul outil : Kicad qui est un logiciel libre et gratuit. Détaillons maintenant ces différents logiciels et voyons à quoi ils ressemblent :

-EESchema :

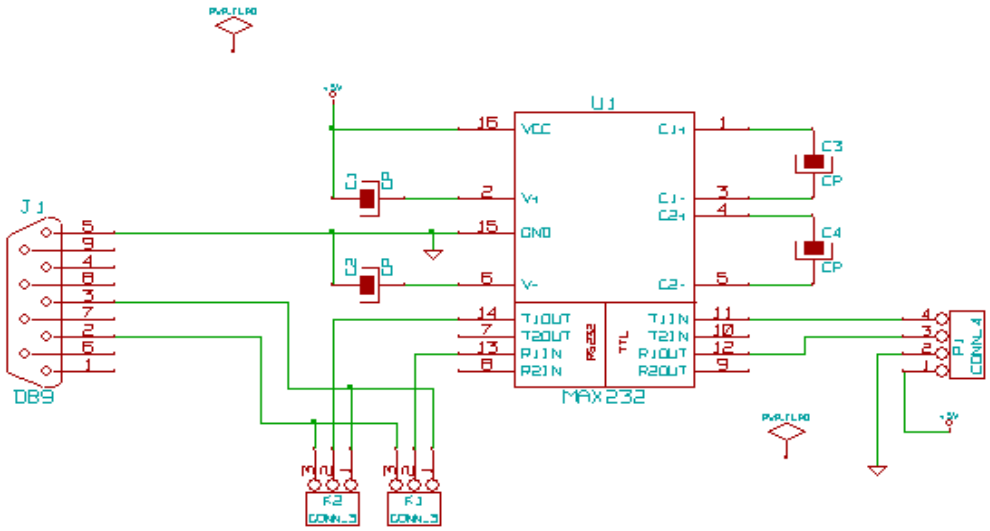


figure 18 : *Exemple de schéma avec EESchema.*

Tout d'abord, il faut ajouter tous les composants que l'on souhaite avoir sur le circuit imprimé sur la feuille EEschema. Il est conseillé de les disposer de telle sorte que les fils ne se croisent pas et que le schéma soit le plus lisible possible, mais cela n'a aucune incidence sur le bon fonctionnement du produit final. Ensuite on trace des fils entre les pattes des composants afin de définir les connections.

Une fois tous les composants placés et reliés, on lance le test ERC qui vérifie le respect des règles électriques. On peut ensuite passer sous Cvpcb.

- Cvpcb :

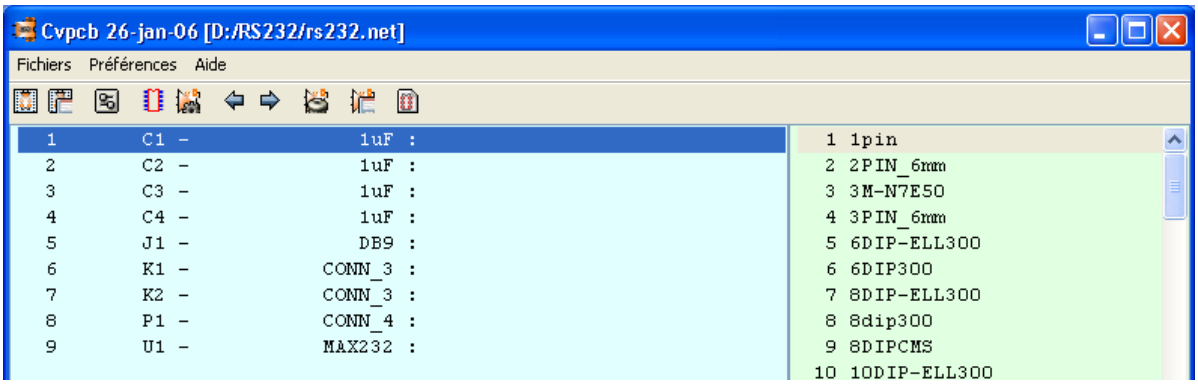


figure 19 : *Exemple d'utilisation de Cvpcb.*

On attribue alors à chaque composant son empreinte physique, ce qui correspond aux pastilles de cuivre qui seront tracées sur le circuit imprimé afin de souder ce composant. Il est donc fondamental de vérifier les dimensions entre les pattes des composants pour l'insertion de ceux-ci sur le circuit imprimé. Il est aussi indispensable de s'assurer de la concordance des numéros des pattes entre l'empreinte et le schéma pour que, lors du routage, les connections se fassent entre les bonnes pattes.

- PCBnew :

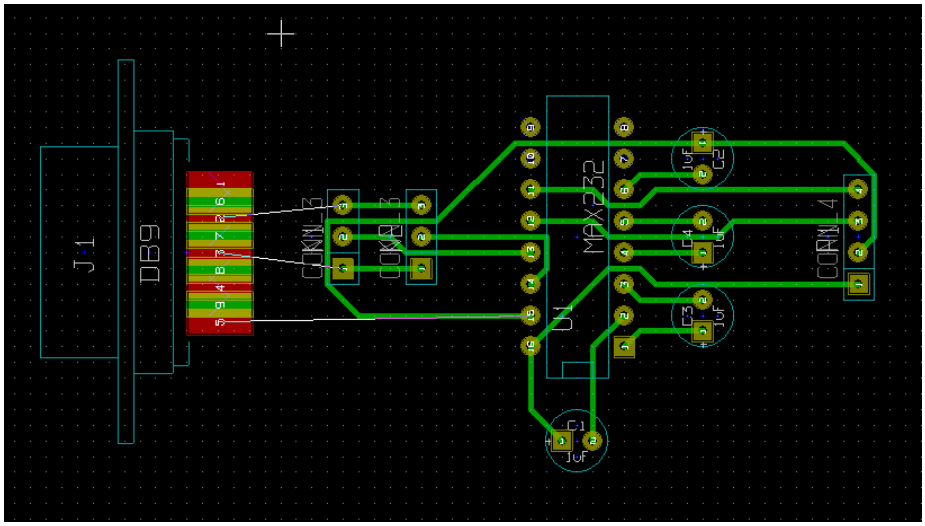


figure 20 : *Exemple de schéma avec PCBnew.*

Avec PCBnew, on choisit tout d'abord la largeur des pistes, ainsi que la forme et les dimensions des pastilles de cuivre. En effet, elles doivent être suffisamment grandes pour permettre le perçage et une bonne soudure du composant.

Ensuite on lance le routage automatique, mais le premier n'est jamais le bon. Il faut tester différentes dispositions jusqu'à obtenir une carte avec des pistes larges et une bonne connexion entre tous les composants suivants le dispositif initial fixé dans EESchéma.

Le circuit est alors fin prêt à être fabriquer. Nous avons obtenu cette carte :

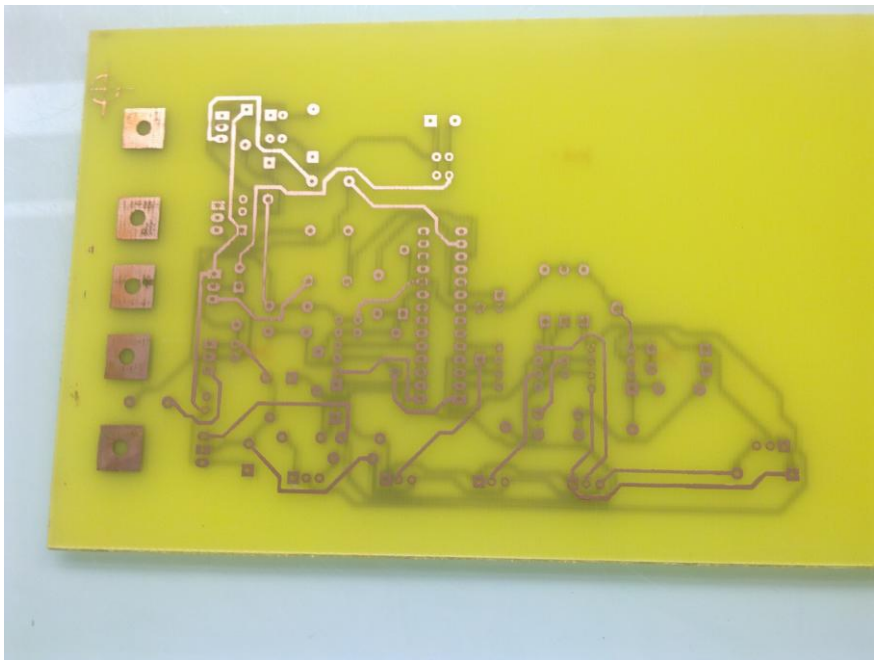


figure 21 : Notre circuit imprimé

4. CONCLUSIONS ET PERSPECTIVES

Pour conclure ce dossier, nous pouvons dire qu'au départ, les connaissances mécaniques et électroniques requises par ce projet dépasser de loin notre formation en département STPI. Grâce à des recherches et aux cours spécifiques fournis par M.Delamare nous avons été plus à même de comprendre ce projet et notamment le fonctionnement du robot ainsi que de ses nombreux points techniques comme par exemple : principe du pont en H, PID pour prévoir l'action à venir en s'appuyant sur l'état passé, programmation du PIC etc.

En réalisant ce projet, nous avons donc appris de nombreuses choses sur le fonctionnement général des robots gérés par des PICs et nous avons également été formés au logiciel de création de circuit imprimé Kicad, à la programmation en C adapté aux PICs de Microchip.

Ce projet de P6-3 nous aura été très instructif autant sur le plan pédagogique (électronique et mécanique) mais également sur le plan du management de projet. En effet, nous avons vite réalisé que de n'être que 3 pour réaliser un tel projet alors qu'aucun d'entre nous n'avez de connaissances particulières dans les domaines requis était un gros handicap pour mener à bien notre projet en respectant le planning défini au début du semestre. Le manque d'effectif ne nous a ni permis de diviser efficacement les tâches à accomplir ni d'affecter plus de monde sur une tâche particulière au bon moment. Le problème principal rencontré reste le circuit imprimé pour 2 raisons : tout d'abord nous étions contraints « à copier » le modèle du circuit imprimé de l'AN964 fourni par Microchip car nous n'avons pas les connaissances suffisantes pour le concevoir nous-mêmes. Cependant, nous avons du faire face à des changements réguliers (PIC 28 branches au lieu de 14, suppression des drivers de mosfet, trouver des équivalents et remplacer tous les composants CMS, etc.).

Finalement, nous avons réussi à créer le circuit imprimé et à monter notre robot à la dernière minute, mais des tests et des réglages restent à faire pour que notre robot accomplisse sa tâche et empêche le pendule de tomber.

En ce qui concerne les perspectives d'avenir pour ce projet, nous pensions que peut-être nous pourrions le finir en temps que projet personnel pour l'année prochaine.

5. BIBLIOGRAPHIE

[1] Fabrice DELAMARE, Cours sur les microcontrôleurs.

[2] Fabrice DELAMARE, Cours sur les ALIENs.

[3] lien internet : http://www.sonelec-musique.com/electronique_theorie_reg_tension.html (valide à la date du 19/04/2010).

[4] John Charais, Ruan Lourens et Microchip Technology Inc., « *Software PID Control of an Inverted Pendulum Using the PIC16F684 »,* http://ww1.microchip.com/downloads/en/AppNotes/00964A.pdf

[5] lien internet : http://fr.wikipedia.org/wiki/Pendule_invers%C3%A9

[6] lien internet : http://www.tavernier-c.com/programmateur_de_pic.htm

[7] lien internet : http://fr.wikipedia.org/wiki/Microcontr%C3%B4leur_PIC

[8] lien internet : http://www.abcelectronique.com/acquier/PIC.html

[9] lien internet : http://www.elektronique.fr/logiciels/kicad.php

[10] lien internet : http://www.kicadlib.org/Fichiers/Tutorial_Kicad_FR.pdf

6. ANNEXES (NON OBLIGATOIRE)

6.1. Documentation technique

Nous avons beaucoup utilisé les datasheet des différents composants de notre circuit imprimé, afin par exemple d'établir une comparaison entre les différents PIC, ou encore de vérifier le branchement de certains composants.

6.2. Listings des programmes réalisés

Notre programme (simulation) :

```
//Importation des bibliothèques
#include <pic.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

unsigned char temp; //variable qui va nous permettre de temporiser pour récupérer l'angle
unsigned short int i; //variable qui va nous permettre de voir le chenillard des leds défiler
unsigned int x; //variable dans laquelle va être stocké l'angle

void main(){
    TRISA = 0b100000; //initialisation des ports (en entrées avec des 1 ou sorties avec 0)
    TRISB = 0b00000000;
    TRISC = 0b0000;

    ADFM = 1;     // justification bit à droite registre de réception ADRESH ADRESL
    ADON = 1;     //Activer la conversion A/D

    RA0 = 1;
    RB0 = 0;
    RB1 = 0;
    RB2 = 0;
    RB3 = 0;

    while(1){        //Boucle infinie
```

```
x = 0;   //initialisation de x
temp = 200;    // On donne du temps pour récupérer la valeur de x
while(temp){
        temp--;
}
GODONE = 1;
while(GODONE);{
        temp = 0;                       // Tant que GODONE=1, on ne fait rien
}
x = ADRESH << 8;            // On stock le résultat de la conversion A/D dans
x
x = x + ADRESL;
x = (x - 893)*100;          //ceci permettra de donner une vitesse au
chenillard proportionelle à x

if(x<6500){                             //Si x < valeur centrale, le chenillard va de
droite à gauche
        PORTB = 0b0001;       //Choix de la led à allumer
for (i=0 ; i<x ; i++)       //Vitesse du chenillard proportionelle à x
continue;
PORTB = 0b0010;
for (i=0 ; i<x ; i++)
continue;
        PORTB = 0b0100;
for (i=0 ; i<x ; i++)
continue;
        PORTB = 0b1000;
for (i=0 ; i<x ; i++)
continue;
        }

if(x>6500){      //Si x > valeur centrale, le chenillard va de gauche à droite
        PORTB = 0b1000;
for (i=13000 ; i>x ; i--)
continue;
PORTB = 0b0100;
for (i=13000 ; i>x ; i--)
continue;
```

```
                PORTB = 0b0010;
            for (i=13000 ; i>x ; i--)
            continue;
                PORTB = 0b0001;
            for (i=13000 ; i>x ; i--)
            continue;
                }

            /*if(x=6500){          //Petit plus: si la valeur centrale est atteinte, toute les leds s'allument, et le buzzer sonne
                PORTB = 0b1111;
                RC2 = 1;
                temp = 200;
                while(temp){
                    temp--;
                    }
                RC2 = 0;*/
                }
            }
        }
```

Programme de l'AN964 :

```
//Importation des bibliothèques
#include <pic.h>
#include <pic16f684.h>
#include <math.h>
#include <stdlib.h>

//Déclaration des procédures
void Init();
void PID();
void Set_Constants();

//Déclaration des variables
bit flag1,do_PID,int_flag;
signed char en0, en1, en2, en3, term1_char, term2_char, off_set;
unsigned char temp;
short int temp_int;
unsigned short int ki, kd, kp;
signed int SumE_Min, SumE_Max, SumE, integral_term, derivative_term, un;
signed long Cn;

//        __CONFIG    _CP_OFF & _CPD_OFF & _BOD_OFF & _MCLRE_ON & _WDT_OFF
& _INTRC_OSC_NOCLKOUT & _FCMEN_ON

//*************************************************************************
// Positional PID 256 Hz
//*************************************************************************

//*************************************************************************
//Main() - Main Routine
//*************************************************************************

//Fonction principale
void main()
{
        Init();                 //Initialiser le PIC
        Set_Constants();        //Obtenir les coefficients du PID ki, kp et kd
        while(1)            //Boucle infinie
```

```
        {
                if(do_PID){
                        PID();
                }
        }
}

//*************************************************************************
//Init - Initialization Routine
//*************************************************************************

//Fonction Init
void Init()
{
        PORTA = 0;
        TRISA = 0b00101101;             // Mettre RA0, RA2, RA3 et RA5 en entrées, le reste en sortie
        PORTC = 0;
        TRISC = 0b00000011;             // Mettre RC0 et RC1 en entrées, le reste en sortie
        CMCON0 = 0x07;                  // Désactiver le comparateur

        IRCF0 = 1;
        IRCF1 = 1;
        IRCF2 = 1;

        CCP1CON = 0b01001100;           // Full bridge PWM forward
        ECCPAS = 0;
        PR2 = 0x3F;                     // régler la période PWM à 31.2 kHz
        T2CON = 0;                      // TMR2 Off
        CCPR1L = 0;
        TMR2ON = 1;                     // Démarrer Timer2

        ANSEL = 0b00110101;                     // Configurer AN0,AN2,AN4 et AN5 en analogue
        VCFG = 0;                       // Utiliser Vdd comme Ref
        ADFM = 1;                       // Justification bit à droite registre de réception ADRESH ADRESL
        ADCS0 = 1;
```

```
        ADCS1 = 0;
        ADCS2 = 1;
        CHS0 = 0;                       // La chaine selectionnée est AN0
        CHS1 = 0;
        CHS2 = 0;
        ADON = 1;                       // Activer la conversion A/D


        //Initialisation des variables
        en0 = en1 = en2 = en3 = term1_char = term2_char =0;
        ki = kd = 0;
        kp = off_set = 0;
        temp_int = integral_term = derivative_term = un =0;
        SumE_Max = 30000;
        SumE_Min = 1 - SumE_Max;
        do_PID = 1;                     // Autoriser à faire la fonction PID
        T0CS = 0;                       // Timer0 comme timer et pas comme compteur
        TMR0 = 10;                      // Valeur préchargée
        PSA = 0;
        PS0 = 0;
        PS1 = 0;
        PS2 = 1;
        INTCON = 0;
        PIE1 = 0;
        T0IE = 1;
        GIE = 1;
        return;
}

// Fonction PID
void PID()                              // La formule: C(n) = K(E(n) + (Ts/Ti)SumE +
(Td/Ts)[E(n) - E(n-1)])
{
        integral_term = derivative_term = 0;

// Calculer le terme intégral
        SumE = SumE + en0;                                      // SumE est la
somme des erreurs
```

```
        if(SumE > SumE_Max){                                    // Tester si la somme est tros grande
                SumE = SumE_Max;
        }
        if(SumE < SumE_Min){                                    // Tester si la somme est trop petite
                SumE = SumE_Min;
        }                                                       // Le terme intégral est (Ts/Ti)*SumE où Ti est Kp/Ki
                                                                // et Ts est la période
                                                                //
L'équation utilisée pour calculer le terme intégral est
                                                                //
Ki*SumE/(Kp*Fs*X) où X  est un facteur inconnu
                                                                // et Fs est la fréquence
        integral_term = SumE / 256;                     // Diviser par la fréquence
        integral_term = integral_term * ki;             // Mutiplier par Ki
        integral_term = integral_term / 16;             // Combinaison entre X et Kp

// Calculer le terme dérivatif
        derivative_term = en0 - en3;
        if(derivative_term > 120){                      // Tester si trop grand
                derivative_term = 120;
        }
        if(derivative_term < -120){                     // Tester si trop petit
                derivative_term = -120;
        }                                                       //
Calculer le terme dérivatif en utilisant (Td/Ts)[E(n) - E(n-1)]
                                                                // Où Td est Kd/Kp
                                                                //
L'équation utilisée est Kd(en0-en3)/(Kp*X*3*Ts)
        derivative_term = derivative_term * kd;         // Où X un facteur inconnu
        derivative_term = derivative_term >> 5;     // Diviser par 32 précalculé par Kp*X*3*Ts

        if(derivative_term > 120){
                derivative_term = 120;
        }
        if(derivative_term < -120){
```

```
                derivative_term = -120;
        }
                                                                                //
C(n) = K(E(n) + (Ts/Ti)SumE + (Td/Ts)[E(n) - E(n-1)])
        Cn = en0 + integral_term + derivative_term;          // Sommer les termes
        Cn = Cn * kp / 1024;                                        // Multiplier par Kp


        if(Cn >= 1000)                                                  // Utilisé pour limiter
le cicle
        {
                Cn = 1000;
        }
   if(Cn <= -1000)
        {
                Cn = -1000;
        }
        if(Cn == 0){                            // Régler la vitesse des moteurs
                DC1B1 = DC1B1 = 0;
                CCPR1L = 0;
        }
        if(Cn > 0){                                     // Faire fonctionner les moteurs en
marche avant
                P1M1 = 0;                               // Les moteurs fonctionnent en marche
avant
                temp = Cn;
                if(temp^0b00000001){
                        DC1B0 = 1;
                }
                else{
                        DC1B0 = 0;
                }
                if(temp^0b00000010){
                        DC1B1 = 1;
                }
                else{
                        DC1B1 = 0;
                }
                CCPR1L = Cn >> 2;               // Utilisé pour éviter que le robot avance
continuellement si le pendule est tombé
```

```
            off_set = off_set +1;          // L'offset est utilisé pour ajuster légèrement l'angle du pendule
            if(off_set > 55){
                off_set = 55;
            }
        }

        else {                             // Faire fonctionner les moteurs en marche arrière
            P1M1 = 1;                      // Les moteurs fonctionnent en marche arrière
            temp_int = abs(Cn);            // Retourner la valeur absolue de Cn
            temp = temp_int;
            if(temp^0b00000001){
                DC1B0 = 1;
            }
            else{
                DC1B0 = 0;
            }
            if(temp^0b00000010){
                DC1B1 = 1;
            }
            else{
                DC1B1 = 0;
            }
            CCPR1L = temp_int >> 2;        // Utilisé pour éviter que le robot recule continuellement si le pendule est tombé
            off_set = off_set -1;
            if(off_set < -55){
                off_set = -55;
            }
        }
        en3 = en2;              // Echanger l'erreur de signal
        en2 = en1;
        en1 = en0;
        en0 = 0;
        do_PID = 0;                        // C'est fait
        RA4 = 0;                           // Test pour mesurer la vitesse de la boucle
        return;
```

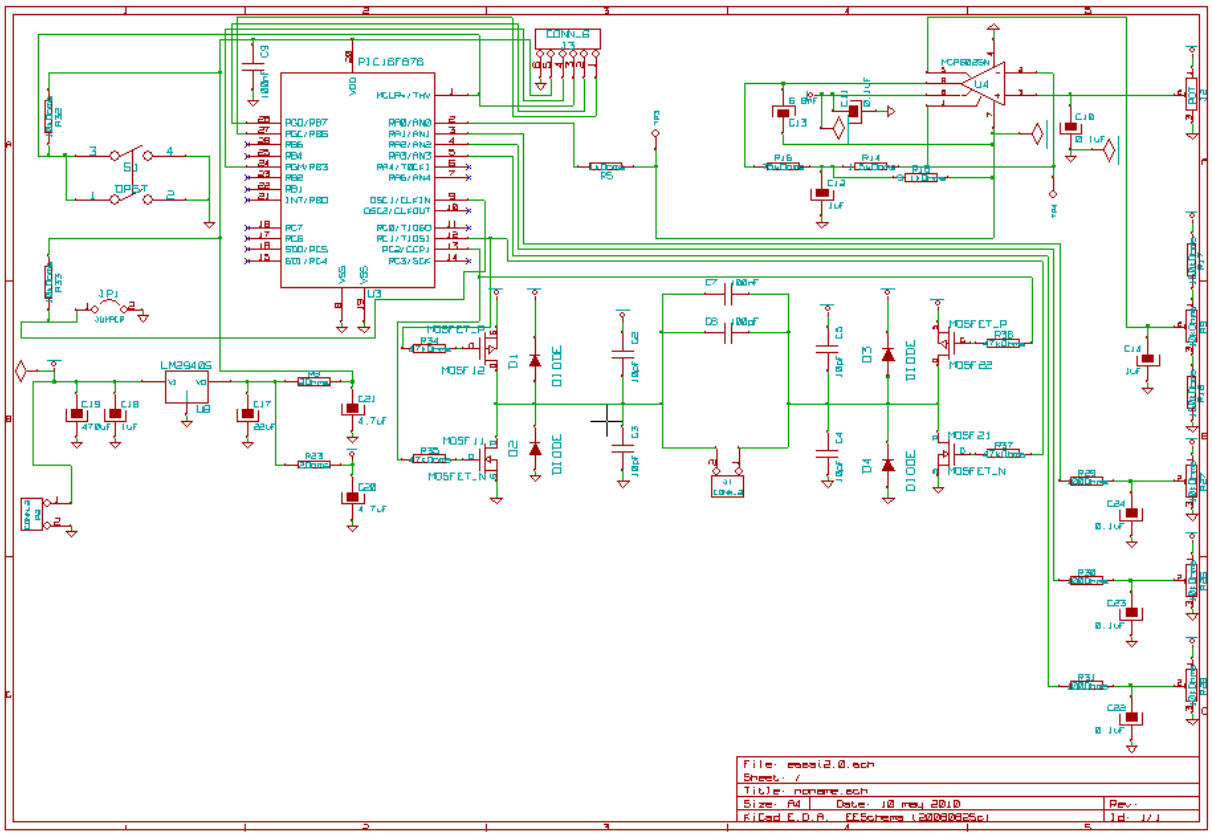
```
}

//Fonction Set_Constants
void Set_Constants()
{
	ANS2 = 1;			// Configurer AN2 en analogue
	ANS4 = 1;			// Configurer AN4 en analogue
	ANS5 = 1;			// Configurer AN5 en analogue

	ADFM = 1;			/ Justification bit à droite registre de réception ADRESH ADRESL
	CHS0 = 0;			// Chaine sélectionnée est AN4
	CHS1 = 0;
	CHS2 = 1;
	temp = 200;			// Donner du temps
	while(temp){
		temp--;
	}
	GODONE = 1;
	while(GODONE);{
		temp = 0;		// Ne rien faire...
	}
	ki = ADRESH << 8;		// Stocker le résultat de la conversion A/D dans la constante Intégrale
	ki = ki + ADRESL;

	CHS0 = 1;			// Chaine sélectionnée est AN5
	CHS1 = 0;
	CHS2 = 1;
	temp = 200;			// Donner du temps
	while(temp){
		temp--;
	}
	GODONE = 1;
	while(GODONE);{
		temp = 0;		// Ne rien faire...
	}
```

```
	kd = ADRESH << 8;		// Stocker le résultat de la conversion A/D dans la constante différentielle
	kd = kd + ADRESL;

	CHS0 = 0;				// Chaine sélectionnée est AN2
	CHS1 = 1;
	CHS2 = 0;
	temp = 200;				// Donner du temps
	while(temp){
		temp--;
	}
	GODONE = 1;
	while(GODONE);{
		temp = 0;			// Ne rien faire...
	}
	kp = ADRESH << 8;		// Stocker le résultat de la conversion A/D dans la constante proportionnelle
	kp = kp + ADRESL;
	CHS0 = 0;				// Chaine sélectionnée est AN0
	CHS1 = 0;
	CHS2 = 0;
}

void interrupt Isr()
{

	if(T0IF&&T0IE){
		TMR0 = 10;				// Valeur préchargée
		T0IF = 0;

//		flag1 = (!flag1);
		RA4 = 1;

		temp_int = 0;
		temp_int = ADRESH << 8;	// Stocker le résultat de la conversion A/D avec l'offset
		temp_int = temp_int + ADRESL - 512;
		en0 = temp_int + off_set/8;
```

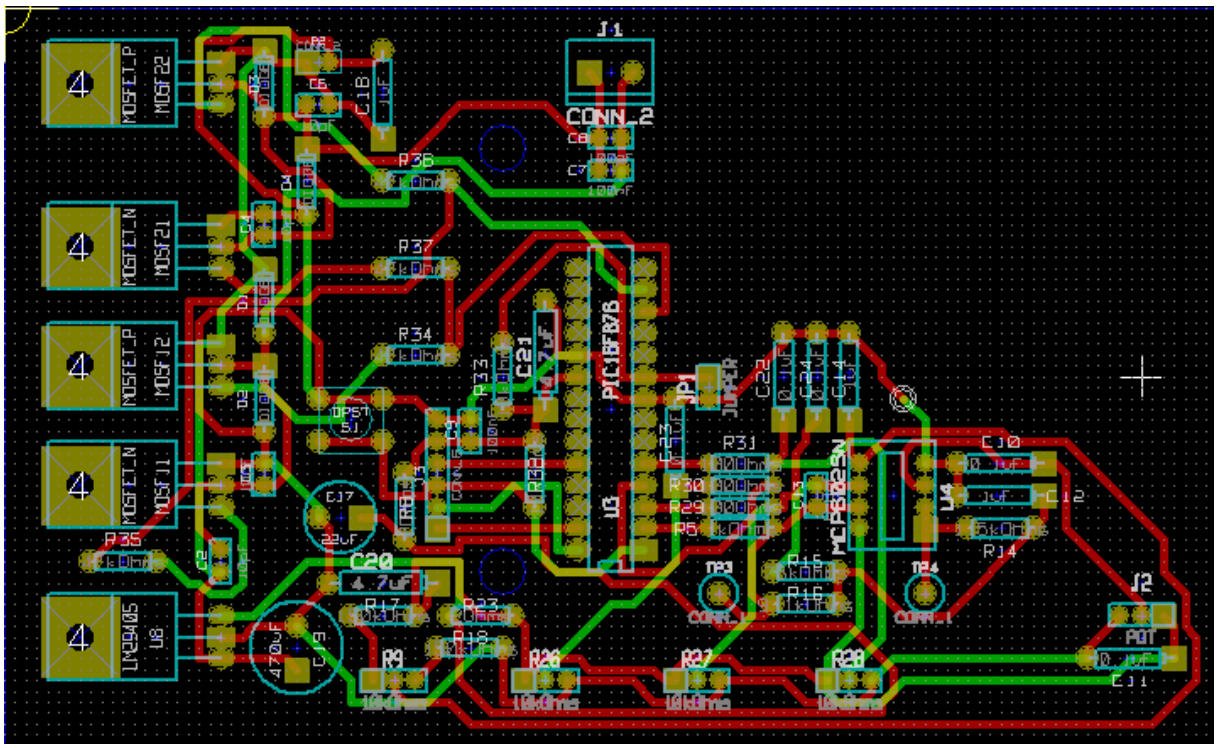
```
            do_PID = 1;                         // Autoriser à faire la fonction PID
            GODONE = 1;                         // Commencer le cycle A/D suivant
        }
        else
        {
            PIR1 = 0;
            RAIF = 0;
            INTF = 0;
        }
        if(temp_int > 180){             // Vérifier si l'erreur est trop grande (positive)

            DC1B0 = DC1B1 = 0;          // Stopper les moteur
            CCPR1L = 0;
            en0 = en1 = en2 = en3 = term1_char = term2_char = off_set = 0;          //
Réinitialiser toutes les constantes PID
            Cn = integral_term = derivative_term = SumE = RA4 = 0;
            do_PID = 0;                         // Stopper PID
        }
        if(temp_int < -180){            //Vérifier si l'erreur est trop grande (négative)

            DC1B0 = DC1B1 = 0;          // Stopper les moteurs
            CCPR1L = 0;
            en0 = en1 = en2 = en3 = term1_char = term2_char = off_set = 0;          //
Réinitialiser toutes les constantes PID
            Cn = integral_term = derivative_term = SumE = RA4 = 0;
            do_PID = 0;                         // Stopper PID
        }

}
```

6.3. Schémas de montages, plans de conception…

Schéma Kicad :



Circuit imprimé final :



6.4. Propositions de sujets de projets (en lien ou pas avec le projet réalisé)

Nous pensons qu'il serait intéressant que l'année prochaine, un projet porte sur la continuation du notre. Le projet consisterait à régler le PID, comprendre les différentes influences de chaque correcteur et ainsi que faire différents test pour que le robot puisse éviter la chute du pendule.