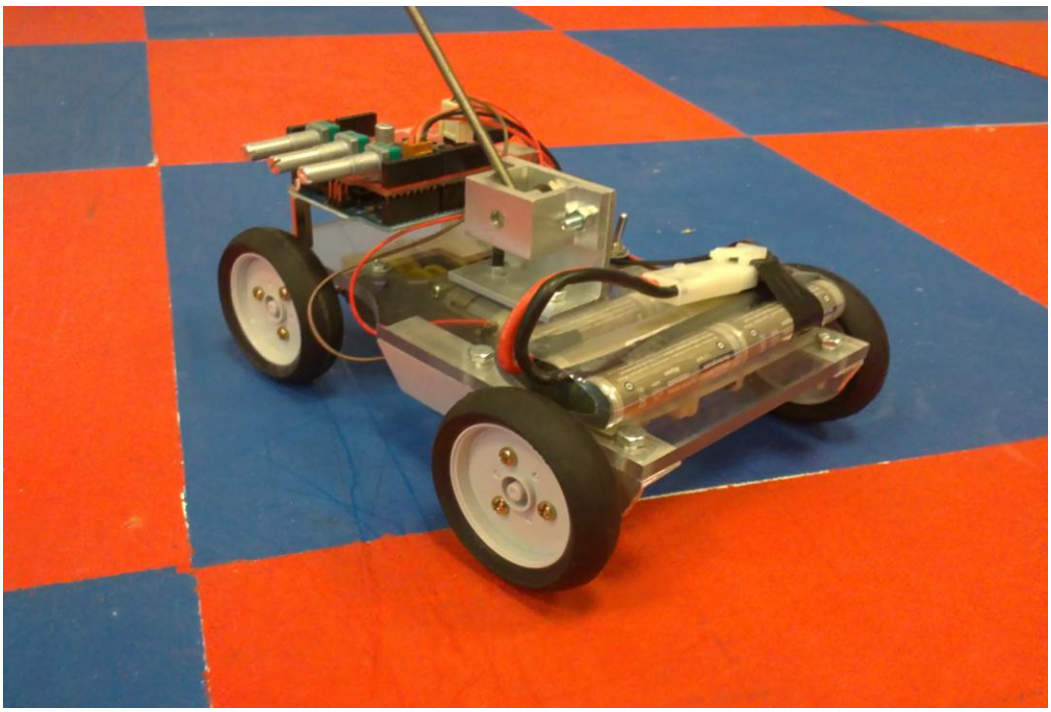
INSA ROUEN
À taille humaine,
à l'échelle du monde

Projet de P6-3
STPI/P6-3/2011-20

Robot à pendule inversé



<u>Etudiants</u> :

ANDRIAMIHAJA Stéphane
DOUASBIN Quentin
EDART Adrien
ENEE Claude
LEONE Mathilde
MAO Zhongyu
RESIDANT Arnaud

<u>Enseignant-Responsable du projet</u> :

DELAMARE Fabrice

Date de remise du rapport : ***18/06/11***

Référence du projet : ***STPI/P6-3/2011 – 20***

Intitulé du projet : ***Robot à pendule inversé***

Type de projet : ***expérimental***

Objectifs du projet :

Construire et programmer un robot capable de maintenir en équilibre un pendule inversé. Organiser ce projet en autonomie, en répartissant les tâches, en définissant un planning et en tenant un carnet de suivi.

Mots-clefs du projet :
robot
équipe
pendule inversé
PID
Arduino

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

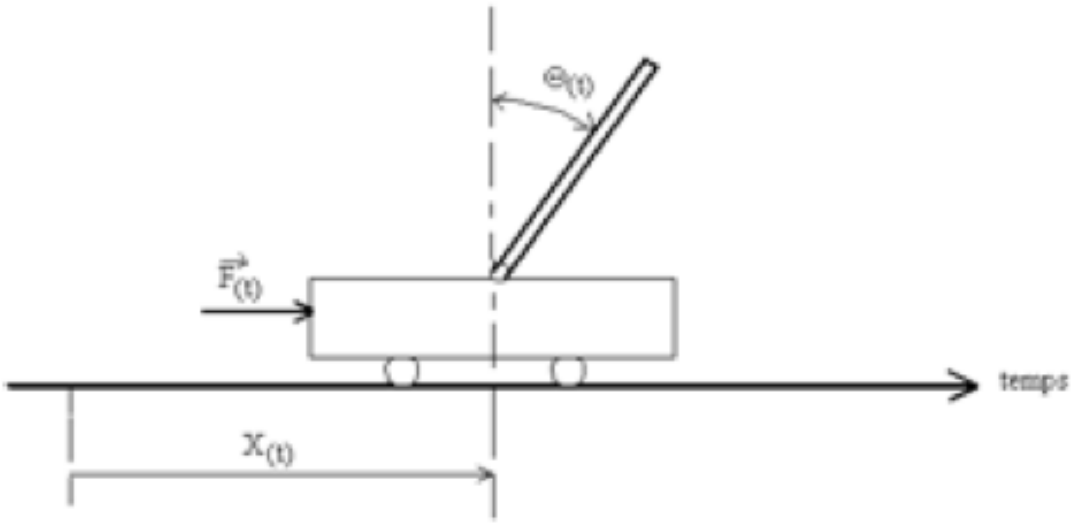
TABLE DES MATIERES

Notations, Acronymes

PID : Proportional Integral Derivative

I) **Introduction** :

Le pendule inversé est un pendule qui a une masse au-dessus de son point de pivot. Le pendule inversé peut se présenter comme suit :



Il est composé d'un chariot sur 3 ou 4 roues, pouvant se déplacer horizontalement, sur lequel se trouve un point de pivot. Alors qu'un pendule normal est stable avec une masse accroché en bas, le pendule inversé est instable puisqu'une masse est accrochée en haut de la tige. Il est donc important de trouver le bon équilibre pour que la tige reste droite tout en faisant avancer ou reculer le chariot. Ce bon équilibre ne peut se faire qu'en déplaçant le point de pivot et en trouvant la bonne masse à placer en haut de la tige. Il existe néanmoins une autre façon de stabiliser le pendule inversé. Elle consiste à faire osciller la tige du pendule de haut en bas. Il faut bien entendu que les oscillations du pendule soient suffisamment fortes pour récupérer les perturbations.

En théorie, il est très simple de comprendre le principe du pendule inversé. En effet quand le pendule penche vers la droite, le chariot doit le rattraper en effectuant un mouvement vers la droite, et vice versa. Mais en pratique, la difficulté est de régler l'intensité et la forme de la réaction du chariot en fonction de l'angle fait avec la verticale.

Le pendule inversé est un problème classique dans la dynamique et la théorie des commandes. Sa principale utilisation est de servir de repère pour les algorithmes d'essai de commande.

Voici quelques exemples de robot à pendule inversé :

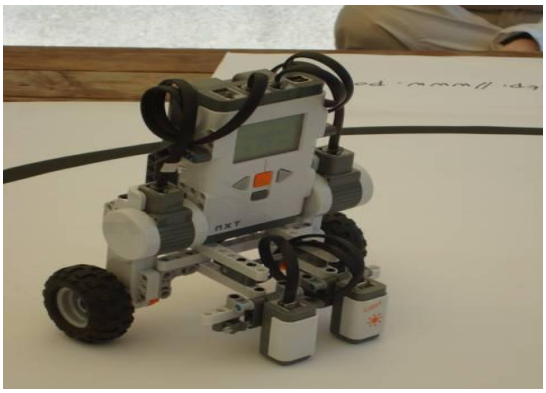


Image 1 : Un exemple de robot pendule

On peut retrouver ces robots dans la vie quotidienne dans l'utilisation des appareils électro-ménagers ou dans les appareils tels que les segways : Le Transporteur Personnel Segway est un gyropode (véhicule monoplace à deux roues où le conducteur se trouve debout). L'utilisateur est maintenu en équilibre grâce au modèle du pendule inversé.

Image 2 : robot wheelie

Image 3 : robot segway

Le système de stabilisation d'une fusée ou encore d'un missile (lors du décollage ou du début de la propulsion) est également une application directe du pendule inversé.
Enfin, il existe d'autres éléments que nous retrouvons dans la vie de tous les jours, tel que le système tonique postural (c'est à dire le système qui contrôle l'équilibre de l'être humain). Plus généralement nous pouvons généraliser ce modèle aux corps qui subissent un état d'équilibre instable.

II) Méthodologie - Organisation du travail :

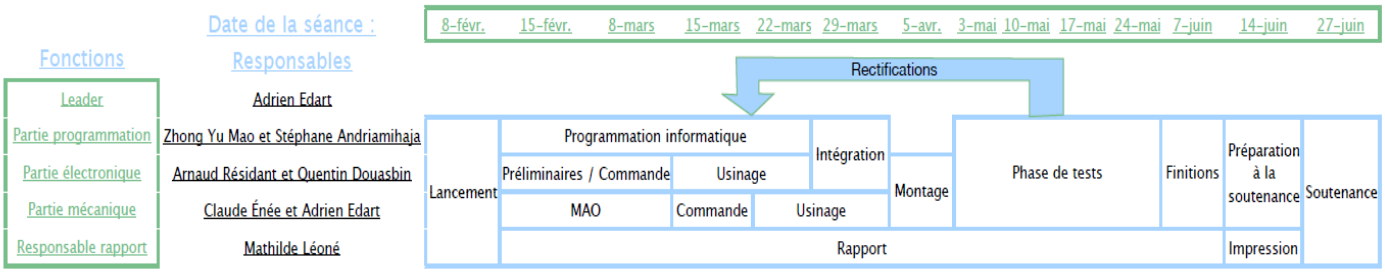
Pour réussir un tel projet, nous nous sommes fixés des objectifs dès le début.

Objectif au niveau du travail : Notre projet consiste en la fabrication d'un robot à pendule inversé. Pour se faire, le robot devra être construit et programmé. Notre objectif principal est que notre robot fonctionne, c'est-à-dire qu'il n'y ait pas d'erreurs de fabrication ni de programmation.

Objectif au niveau du groupe : Bien entendu, qui dit projet de groupe, dit aussi travail de groupe.

Pour nous organiser, nous avons choisi un «leader du projet» : Adrien Edart, et nous avons réparti le travail en 4 domaines de responsabilité :
- mécanique : ENEE Claude et EDART Adrien
- électronique : DOUASBIN Quentin et RESIDANT Arnaud
- programmation : ANDRIAMIHAJA Stéphane et MAO Zhongyu
- organisation du rapport : LEONE Mathilde

Nous avons également mis en place un planning afin de conserver des objectifs de temps.



Précision : Ce planning est purement indicatif, il est suceptible d'être modifié, et permet éventuellement de fixer des objectifs.

III) Constitution d'un robot à pendule inversé :

Dans cette partie, nous essayerons d'expliciter de quoi est constitué un robot à pendule inversé ou toute autre appareil utilisant un pendule inversé.

Tout d'abord, un robot mobile à pendule inversé possède un châssis qui peut être composé de différents matériaux (bois, métal, plastique…) de façon à être le plus léger et le plus solide possible. On peut prendre comme exemple le châssis du Segway qui est composé d'aluminium moulé. Celui-ci doit être en équilibre autour de l'axe des deux roues. Le moteur est un élément important du robot car il actionne son mouvement. Les moteurs utilisés lors de notre projet de P6 – 3 n'auront pas besoin d'une grande puissance comparée à ceux utilisés par exemple pour le segway. Dans ce cas, il y a même un moteur pour chaque roue d'une puissance de quatre chevaux chacun (soit 1880W).

Image 1 : châssis en aluminum

Le châssis supportera 4 roues, un 2 double moteur une carte Arduino (qui devra être programmée), le pendule et son potentiomètre (= point de pivot) et des batteries.

Image 2 : double moteur

Ensuite, il est nécessaire d'avoir un microcontrôleur (ou une carte-mère) car il est programmé pour gérer la façon dont le robot va se comporter suivant la situation à laquelle il est confronté. Les microcontrôleurs les plus performants se présentent de nos jours sous la forme de cartes mère de PC miniaturisées comme c'est le cas pour les robots pendules les plus évolués (microcontrôleur STM32 utilisé pour un robot pendulaire inversé conçu par STMicroelectronics et un Institut de Robotique Humanoïde japonais)

Image 3 : microcontrôleur

Le microcontrôleur peut être programmé en différents langages. Mais il faut savoir que plus le langage utilisé est complexe, plus la carte va mettre du temps à exécuter le code, et donc le robot sera plus lent face à certaines situations qu'il rencontrera. Dans ce cas (et dans le cadre de notre projet physique), un bon compromis est l'utilisation du langage C même s'il en existe beaucoup d'autres (BASIC, G, assembleur…). Il est compilé la plupart du temps grâce au compilateur GCC. Il faut souligner qu'avant d'être utilisé sur le robot, le microcontrôleur doit être testé (ce n'est cependant pas possible pour tous les microcontrôleurs). Il existe pour ce faire des simulateurs pour tester le comportement du microcontrôleur et du programme (MPLAB par exemple)

Pour que le robot maintienne le pendule inversé tout en se déplaçant latéralement, il est nécessaire de créer un algorithme de régulation pour gérer ce déplacement. Le problème du robot supportant un pendule inverse est le fait que la valeur de la consigne donnée par le robot doit en permanence être rectifiée pour s'adapter aux conditions réelles. Pour se faire, on rajoute un correcteur afin qu'il puisse affiner la consigne de départ en évaluant ce qui s'est passé et ce qui va se produire. Dans le cas de notre projet, nous avons utilisé un correcteur PID. Le correcteur PID va amplifier le signal (Proportional), va prendre en compte ce qui s'est passé (Integral) et va prévoir ce qui va se passer (Derivative). Le système de régulation gagne alors en rapidité et précision.

La complexité du correcteur PID vient du fait que les trois parties de son correcteur ont des avantages et des inconvénients qui parfois s'opposent. Il faut donc trouver un compromis entre ces trois correcteurs pour fabriquer le correcteur PID idéal.

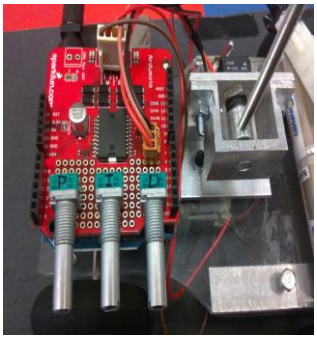


Image 4 : Correcteur PID

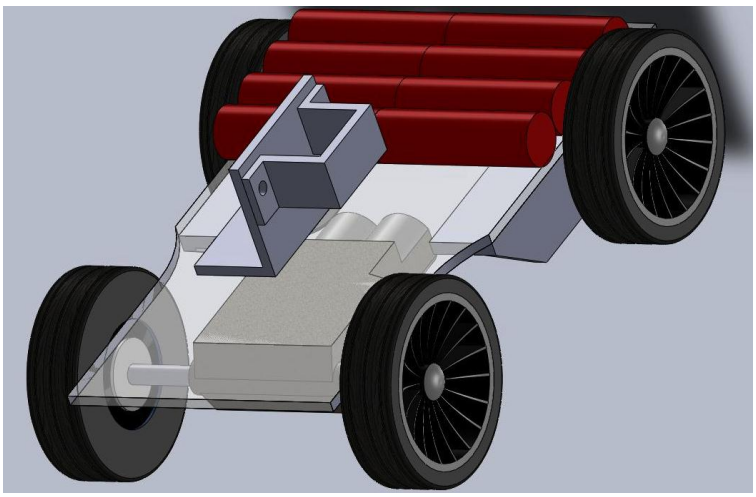
IV) Réalisation du projet :

A) Partie mécanique :

a) Les critères

Pour réaliser le robot supportant le pendule inversé, il nous a été mis à disposition un double moteur TAMIYA équipé de deux réducteurs ainsi que de quatre roues.
Nous avons donc opté pour un système châssis-roues équipé d'un accumulateur, des cartes électroniques ainsi que d'un pendule inverse.

Le robot ne devait se déplacer que dans une seule direction et le châssis devait être suffisamment rigide pour résister aux changements de sens brusques.

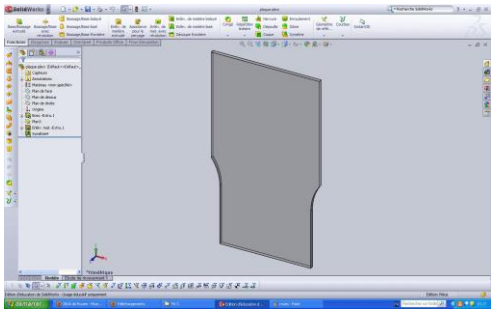


b) Conception

- **Equilibrage des masses**

Pour avoir une stabilité maximale, les charges les plus lourdes devaient être répartie de manière avoir le centre des masses centré. Nous avons donc choisi de mettre l'accumulateur d'un côté et le moteur et les circuits électriques de l'autre. Le pendule a été placé au centre du châssis pour éviter que le poids du pendule fasse basculer le châssis.

- **Le châssis**

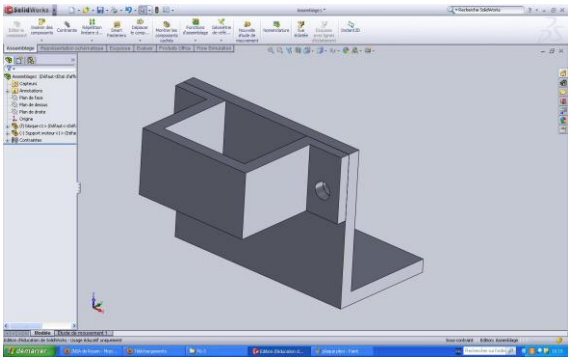


Le châssis devait être assez rigide pour qu'il reste parfaitement horizontal malgré les contraintes dues aux charges et aux changements de sens brusques. De plus il devait être facile à usiner et le plus léger possible. Nous avons choisi d'utiliser un châssis en Plexiglas, un thermoplastique transparent, très facile à couper et à percer, assez rigide et léger. De plus le Plexiglas est un matériau esthétique.

Pour alléger le robot au maximum, le châssis devait être le plus petit possible. De plus la distance entre les deux essieux devait être assez rapprochée pour éviter la flexion du châssis mais en même temps assez éloignée pour éviter le basculement du robot par le poids du pendule.
Nous avons ainsi opté pour un châssis rectangulaire de (10x17). La largeur de l'accumulateur étant plus grande que celle de l'essieu des roues motrices, nous avons pallié ce problème en découpant le châssis pour permettre le passage des roues motrices.

- **Le pendule**

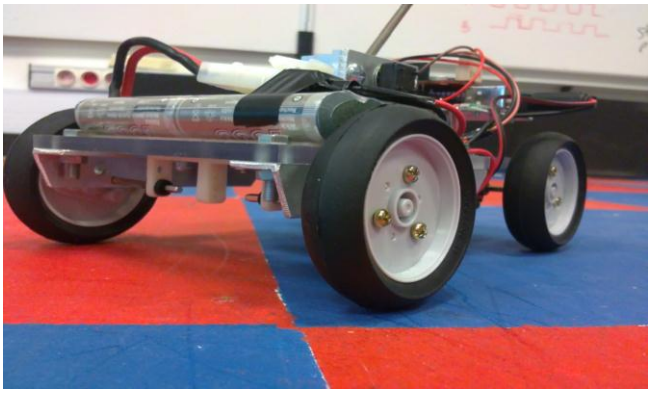
Nous avons réutilisé le pendule de l'an passé. Il était composé d'une tige filetée qui se visse dans un porte-tige fixé au potentiomètre.



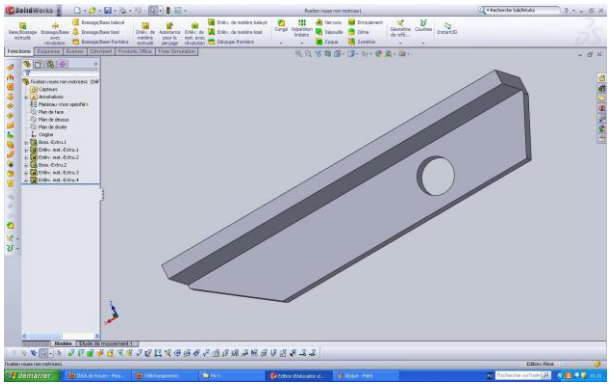
Le porte-tige a été usiné pour correspondre au nouveau potentiomètre.

Nous avons dû rajouter une butée, afin de limiter l'angle du pendule et de ne pas endommager les cartes électroniques.

- **Les roues**

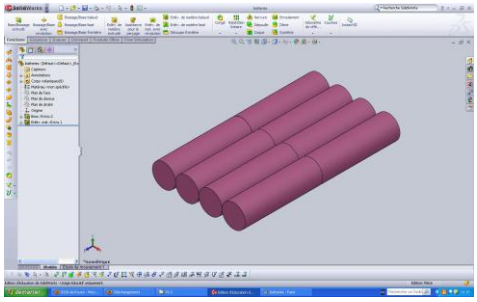


Les roues ont été récupérées sur un ancien robot. Leur diamètre nous convenait, en revanche leur adhérence ne nous suffisait pas.

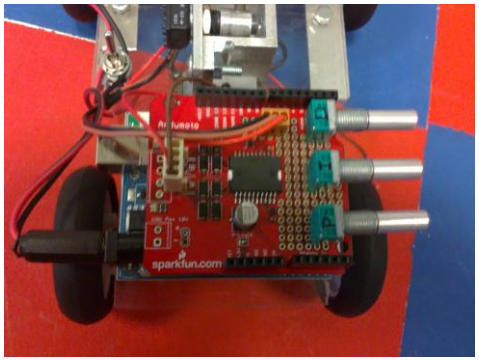


Le support des roues non-motrices permet en même temps de rigidifier la plaque horizontale.

- **L'électronique**

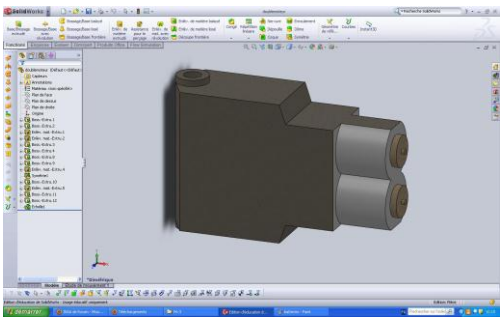


Nous avons installé les batteries entre les 2 roues non-motrices de façon à minimiser l'espace utilisé et à équilibrer l'ensemble.

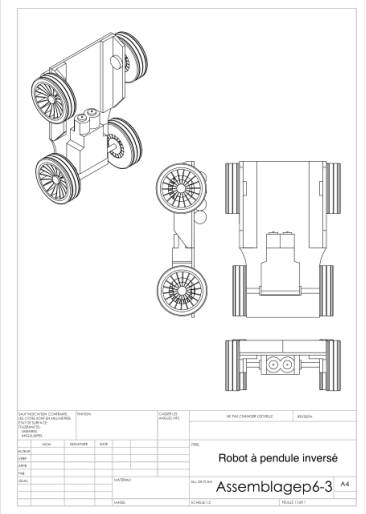


Afin de libérer de la place sous les cartes électroniques, les cartes électroniques ont été surélevées à l'aide d'entretoises fixées au châssis.

Le double-moteur Twin-motor Gearbox Tamiya permet de déplacer le robot. Nous avons choisi d'installer les réducteurs pour obtenir un rapport de 58:1 (ou bien 203:1).



Synthèse : notre objectif était de construire une structure pour le robot à la fois assez solide pour rester horizontale, léger pour faciliter le travail du moteur et esthétique.

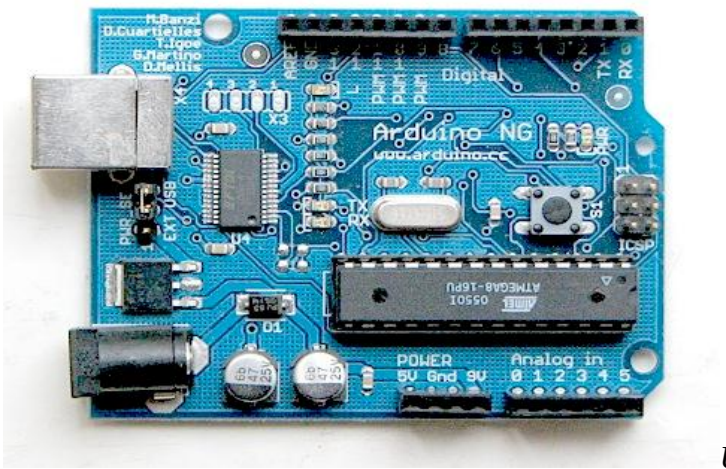


B) Partie programmation :

Notre robot pourra avancer avec l'aide d'un moteur à courant continu tout en supportant un pendule inversé sur son châssis. L'action du moteur est dirigée par une carte Arduino dont la programmation se fait en langage C.

a) La Carte Arduino

Cette carte est basée sur un microcontrôleur Atmel ARV (ATmega 328). Elle dispose de 8 à 128ko de mémoire flash pour stocker les programmes qu'on transfère dessus. Selon les versions, elle possède entre 13 et 14 entrées/sorties numériques et 6 entrées analogiques qui lui permettent de « communiquer » avec d'autres éléments (LED, interrupteurs, moteurs...).



Une carte arduino

Il existe bien sûr des microcontrôleurs autant voire plus performants que ceux de la carte Arduino, mais celle-ci est plus intéressante pour plusieurs raisons.

Tout d'abord, elle n'est pas chère comparée à ses concurrentes. En plus, le logiciel nous permettant de d'écrire, compiler puis transférer le programme de la carte est compatible avec

à la fois Linux, Windows et MacOS. Enfin, il utilise le langage C qui est très utilisé que ce soit chez les utilisateurs les plus avancés ou les débutants.

b) Programmer avec Arduino

Bien que nous n'ayons jamais fait d'informatique embarquée durant notre cursus INSA, nos bases en C/C++ nous permettaient d'aborder la partie programmation sans être totalement perdu. Ce projet était donc l'occasion de véritablement mettre en pratique certaines connaissances que nous avions acquises en cours d'informatique durant les semestres précédents.

Tout d'abord, il nous a fallu partir en quête d'informations pour apprendre à programmer une carte Arduino. Grâce à des recherches faites sur internet et à des indications données par notre professeur, nous nous sommes naturellement dirigés vers le site et le forum d'Arduino : http://arduino.cc/fr.
Sur ce site, nous avons pu télécharger l'application permettant de rédiger un programme, le compiler puis le transférer sur la carte via un câble USB :

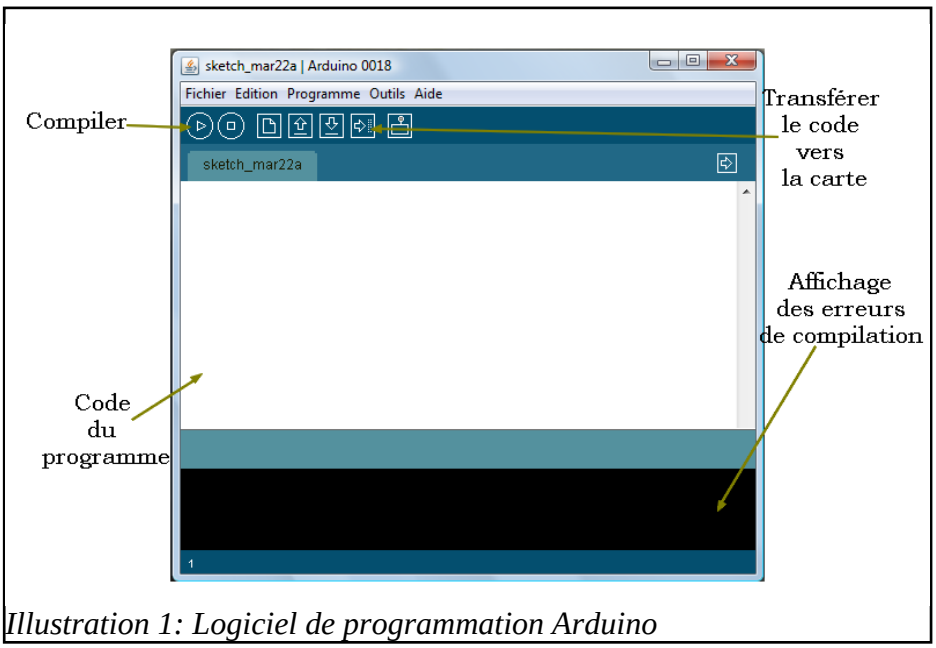
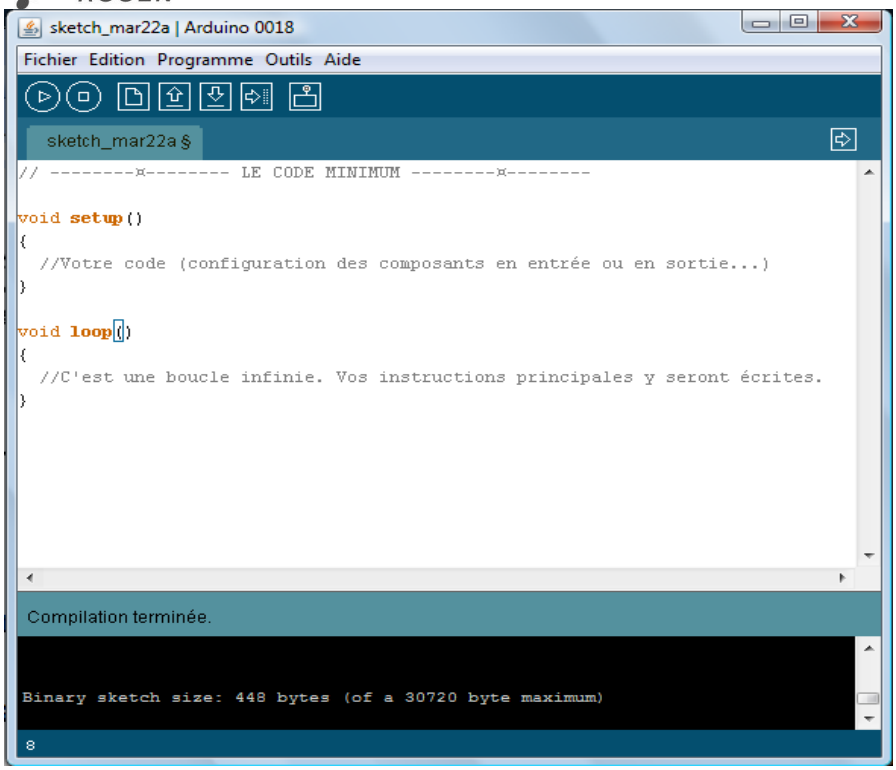


Illustration 1: Logiciel de programmation Arduino

c) Structure du code « minimum »

Lors de l'apprentissage de la programmation de la carte Arduino, un code « minimum » et obligatoire est à connaître pour pouvoir aller de l'avant. Il compose tous les programmes envoyés dans les cartes Arduino, du plus simple au plus élaboré.



Le code minimum

d) Contenu général du code

Cette méthode employée pour l'asservissement utilise trois correcteurs qui doivent permettre de réaliser le meilleur compromis possible entre précision, stabilité et rapidité du système. Elle se sert de l'erreur (la différence entre la consigne, valeur renvoyée par le potentiomètre à l'équilibre, et la mesure) pour faire ces trois correcteurs. Elle est essentielle à l'optimisation du fonctionnement de notre robot à pendule inversé. Dans un premier temps, il fallait définir les constantes et variables du programme qui représentent les éléments connectés aux broches numérotées de la carte Arduino (LED, interrupteur…). On leur assigne ou non une valeur qui correspond à la broche à laquelle l'élément est relié et on indique devant son nom son type (int, float, double…).
Il faut ensuite écrire les instructions qui seront dans les procédures void setup() et void loop().

I. void setup() :
   Dans cette procédure, on configure en entrée (INPUT) ou en sortie (OUTPUT) les éléments connectés aux broches de la carte. Il faut avoir au préalable défini les variables et constantes leur correspondant. On peut également y décider si l'on veut que les valeurs de certains paramètres apparaissent sur l'écran grâce à la commande : Serial.begin()

II. void loop() :

Dans cette boucle, ce sont les interactions et les comportements des éléments branchés à la carte Arduino qui sont définis. Les opérations à effectuer sont écrites dans l'ordre dans lequel elles doivent être réalisées.

e) Transfert du programme

Après sa rédaction puis sa compilation, le programme réalisé avec le logiciel Arduino est envoyé sur la carte via un câble USB. Lorsque deux LED de la carte clignotent deux fois, cela signifie que le programme a été chargé.

f) Le programme et le PID

Le PID est une méthode utilisée fréquemment dans l'industrie qui permet de contrôler un grand nombre de procédés.

Correcteur proportionnel

Ce correcteur a pour but d'améliorer la précision (donc diminuer l'erreur statique) du système utilisé avec son coefficient kp. Il suit la fonction de transfert suivante dans notre programme : Sortie_P = kp*erreur

Cette fonction est équivalente à la formule originale : $U(p)/\mathcal{E}(p) = kp$ avec $\mathcal{E}(p)$ l'erreur et $U(p)$ la sortie.

On applique ainsi une consigne proportionnelle à l'erreur. Cependant, ce correcteur augmente également l'instabilité du système, d'où la nécessité qu'il soit couplé avec les deux autres.

Correcteur intégral

Il s'ajoute au correcteur proportionnel et permet d'éviter l'accumulation de l'erreur au cours du temps dans notre programme grâce à la formule : Sortie_PI = Sortie_P + ki*SommeDesErreurs

Cela permet donc d'augmenter la précision avec laquelle notre robot réagit, et plus la constante ki est grande, plus l'erreur statique est corrigée.

Correcteur dérivateur

Ce correcteur limite le dépassement de la consigne : il permet une stabilisation plus rapide du système en amortissant le mouvement du pendule. On a la formule suivante : Sortie_PID = Sortie_PI + kd*VariationErreur = proportionnel + integrale + derivateur

Ici, VariationErreur est la différence entre la première erreur mesurée et la dernière erreur mesurée.

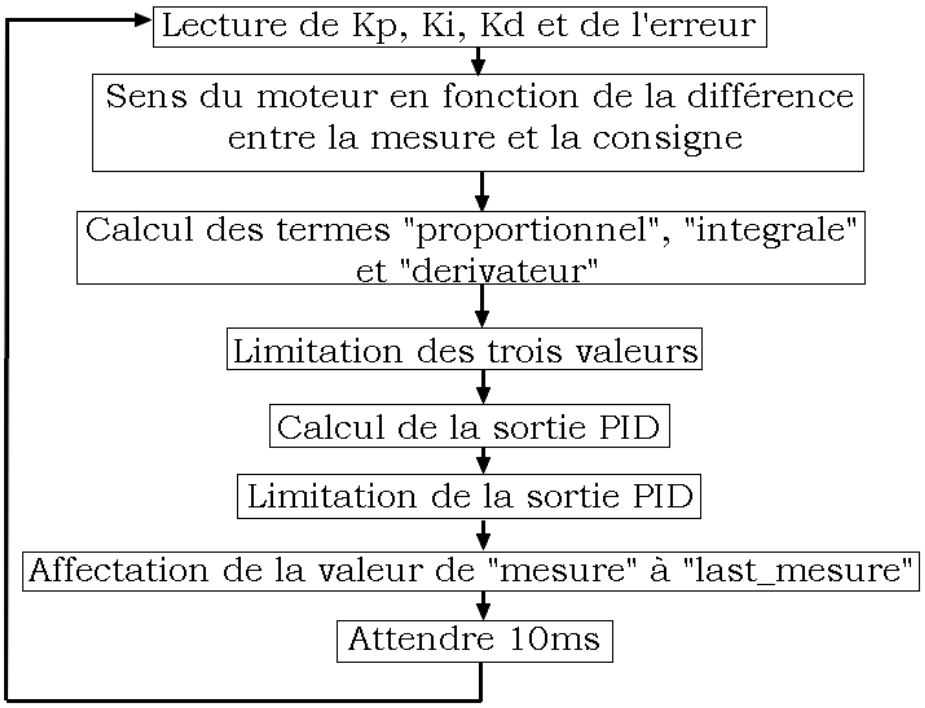
Les trois correcteurs kp, ki et kd peuvent être déterminés mathématiquement ou expérimentalement.

Avec l'aide de notre professeur qui nous a aiguillé quant aux valeurs possibles de ces correcteurs, nous avons donc essayé de déterminer quelles valeurs seraient les plus efficaces. Nous inspirant d'un projet similaire, nous avons donc tenté de trouver ces valeurs avec l'utilisation de trois potentiomètres : les valeurs qu'ils renvoyaient étaient attribuées à celles des constantes recherchées. Les tests effectués devaient ainsi nous permettre de déterminer à quel moment notre robot était le plus stable, rapide et précis...

Pour conclure cette méthode, récapitulons les effets importants de chaque correcteur :

- kp et ki augmentent le temps de stabilisation
- ki diminue considérablement l'erreur statique
- kd diminue le temps de stabilisation et diminue les risques que la mesure dépasse la consigne

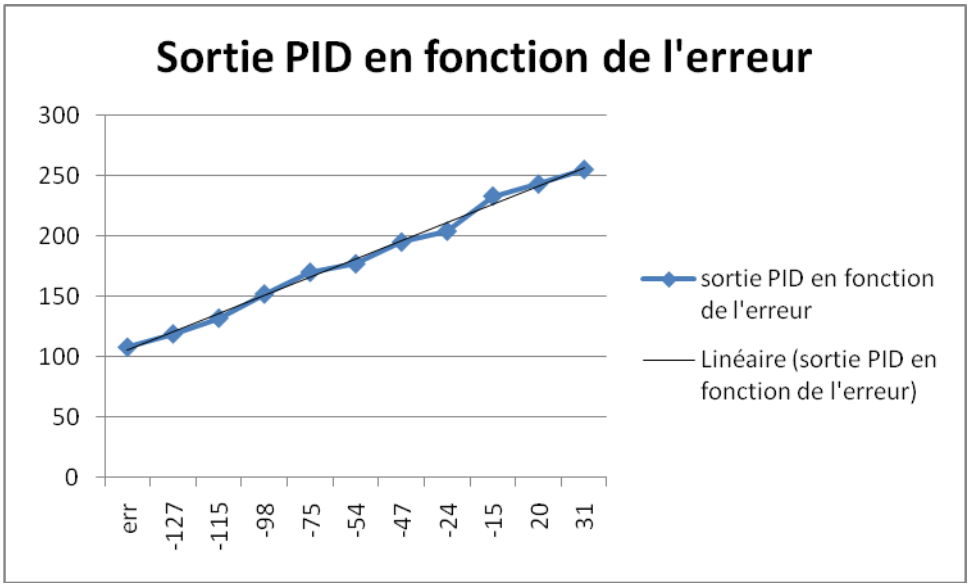
g) Organisation du programme



En résumé, le déplacement du robot dépend de deux choses : l'erreur et la sortie PID.
L'erreur détermine le sens de déplacement de notre robot selon qu'elle soit supérieure à la consigne ou inférieure à celle-ci.
La vitesse de rotation des roues augmente proportionnellement avec la valeur de la sortie PID.

Optimisation de la programmation

Une fois le programme codé, nous avons effectué des phases de test. Nous avons donc fait des prises de mesures pour vérifier que le programme réagissait comme voulu. Les prises de mesures ont permis de réaliser une courbe de la sortie PID en fonction de l'erreur :



Ce graphique nous a permis de comprendre qu'il y avait un problème dans le programme, en effet la sortie_PID est uniquement positive dans la plage d'erreur ou ont été effectuée les mesures.

C) Partie électronique :

a) Choix de l'alimentation électrique et problèmes rencontrés

Tout d'abord, pour plus de mobilité, nous avons choisi d'alimenter le moteur et les cartes à l'aide d'une batterie. Le choix de cette batterie fut assez difficile car il ne fallait pas choisir un bloc d'alimentation trop lourd au risque d'alourdir le robot, mais il fallait en choisir un assez gros pour ne pas avoir non plus à changer à chaque séance. D'un côté purement électronique, il fallait surtout choisir une batterie assez puissante pour que le robot réagisse bien sans toutefois griller la carte. Nous avons donc choisit un bloc d'alimentation de 7,2 V et de 1800 mAh (180mAh pendant 16h).

Lors des essais techniques nous avons vite compris que l'utilisation de cette batterie serait pénible car celle-ci se déchargée très vite et qu'elle alourdissait le robot de manière considérable. Nous avons donc réalisé des câbles d'alimentation afin de relier le GBF au robot. Ces câbles sont à clipser directement à la place de la batterie. Comme ils sont très utiles nous les avons laissé dans la salle pour que tous les groupes puissent les utiliser.

b) Installation d'un interrupteur entre le bloc d'alimentation et la carte shield

Comme le robot est mobile et peut parfois faire des mouvements assez brusques il nous a paru souhaitable d'installer un interrupteur entre le bloc d'alimentation et la carte shield. Cet interrupteur a été récupéré d'un projet précédent et est fixé au châssis. Il est situé sur le fil positif comme tout interrupteur bien monté.

c) Câblage intelligent des éléments fixes

Nous avons décidé de prendre ou de récupérer uniquement des composants qui pouvaient se fixer et s'enlever facilement sur la carte shield. Ainsi, les éléments fixes sur le robot peuvent y rester pendant que l'on retire la carte shield pour revoir la programmation par exemple. C'est pour cela que le potentiomètre qui donne la valeur de l'angle et le bloc moteur peuvent se clipser sur la carte shield et ne sont pas soudés directement dessus. De plus, la carte shield est alimentée directement sur le côté par une connectique d'alimentation de type ordinateur portable afin d'enlever plus facilement la carte.

d) Soudure des différentes parties

Évidemment, à chaque étape de l'électronique il y avait de la soudure à faire, que ce soit pour récupérer d'anciens composants ou pour en ajouter de nouveaux. Il a donc fallu dessouder puis ressouder l'interrupteur, les potentiomètres mal soudés, les moteurs...
De plus, certains objets de récupérations étaient en mauvais état, il a donc fallu changer tous les câbles sur le moteur ou encore dessouder une partie de la shield qui avait grillée et qui risquait de provoquer un court-circuit.
Il a bien sûr fallu vérifier le bon fonctionnement de chaque composant (de récupération ou non) afin d'éviter les mauvaises surprises à la fin.

Conclusions et perspectives

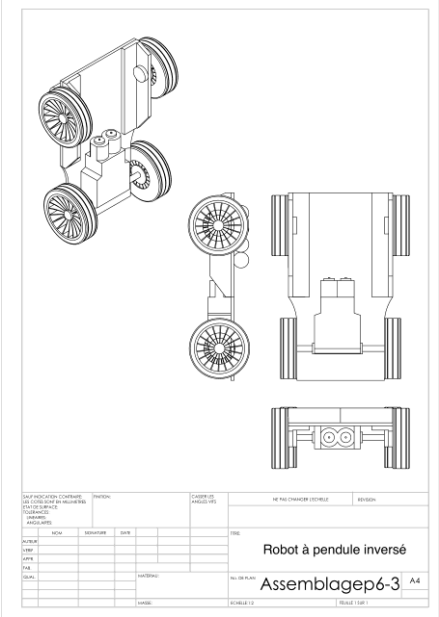
Nous sommes très satisfaits de la réalisation mécanique et électronique de notre robot. Le contrat est rempli : toutes les exigences du cahier des charges ont été respecté. Notre construction est fiable et solide, les différents éléments sont agencés de manière minimaliste et intelligente, et cela avec des moyens de récupération souvent.

Le comportement du robot est cependant la faiblesse de notre réalisation. Le temps de réaction n'est pas suffisamment faible pour maintenir le pendule en équilibre. En revanche, sur une planète dont la gravité serait légèrement plus faible que sur Terre, notre robot serait tout à fait capable de maintenir son pendule en équilibre.

Nos premiers pas à tous dans le monde de la robotique se sont déroulés dans une très bonne ambiance de travail, à la fois sérieuse et détendue. La répartition des tâches a permis à chacun d'assumer ses responsabilités et le planning prévisionnel nous a donné le recul suffisant sur l'avancement du projet. Le bilan est très positif.

Annexes

Plan de conception :



Idées de projets pour les années suivantes :

- Pédales d'effets pour guitares électriques
- Enceintes avec égalisateur et filtre
- Système d'adaptation à la lumière à la luminosité ambiante

Code source du programme :

```
int pot_Kp =0;        //potentiomètre P à broche 0
int pot_Ki =1;        //potentiomètre I à broche 1
int pot_Kd =2;        //potentiomètre D à broche
int pot_Pendule=3;    //potentiomètre pendule inversé à broche 3
int sortie_moteur1=3;  //moteur à broche 3
int sortie_PID=0;     //initialiser la sortie du PID
int sortie_moteur;    //initialiser la sortie moteur
int sens_av=12;       //rotation avant à broche 12
int sens_arr=13;      //rotation arrière à broche 13
int consigne=128;     //valeur indiquée par le pendule à l'équilibre
float last_mesure=0;  //initialisation de la mesure précédente
float proportionnel = 0;
long integrale=0;
float derivateur=0;
float SommeErr = 0;
int pe=10;            //periode d'échantillonage
```

```
void setup()

{

// ------- Broches en entrée -------
  pinMode(pot_Kp,INPUT);
  pinMode(pot_Ki,INPUT);
  pinMode(pot_Kd,INPUT);
  pinMode(pot_Pendule,INPUT);

  // ------- Broches en sortie -------
  pinMode(sens_av,OUTPUT);
  pinMode(sens_arr,OUTPUT);
  pinMode(sortie_moteur1,OUTPUT);
  pinMode(sortie_moteur2,OUTPUT);
 Serial.begin(9600);
}

void loop()
{
        //Lecture des valeurs indiquée par les potentiomètres
        float Kp=analogRead(pot_Kp);
        float Ki=analogRead(pot_Ki);
        float Kd=analogRead(pot_Kd);
        float mesure=analogRead(pot_Pendule);

        Kp=(Kp*20)/1023;   //limiter Kp entre 0 et 20
        Ki=(Ki*100)/1023;  //limiter Kp entre 0 et 100
        Kd=(Kd*20)/1023;   //limiter Kd entre 0 et 20

        mesure=map(mesure,0,1023,0,255);  //limiter la mesure entre 0 et 255
        float Err=consigne-mesure;   //calcul de l'erreur
        float last_Err=consigne-last_mesure;   //calcul de l'erreur précédente
        SommeErr = SommeErr + Err;

        if (Err<-3)
        {
        digitalWrite(sens_av, LOW); //mettre la bronche sens_av au niveau bas
        digitalWrite(sens_arr, HIGH);   //mettre la bronche sens_rr au niveau haut
        }
        if (Err>3)
        {
        digitalWrite(sens_av, HIGH);   //mettre la bronche sens_av au niveau haut
        digitalWrite(sens_arr,LOW);    //mettre la bronche sens_av au niveau bas
        }

        if ((Err>=-3)&&(Err<=3))
        {
        Err=0;
```

```
        }

        proportionnel = Kp*Err;  //calcul du correcteur proportionnel
        integral = Ki*SommeErr; //calcul du correcteur integral



    derivateur=Kd*((Err-last_Err)/pe);   //calcul du terme derivateur

     sortie_PID=proportionnel + integral + derivateur;

        sortie_PID=map(sortie_PID,0,1023,0,255);   //limiter la sortie du  PID entre 0 et 255
        sortie_moteur=abs(sortie_PID);
        analogWrite(sortie_moteur1,sortie_moteur);   //affecter la valeur de la sortie du PID à  la sortie
moteur
// ------- Visualiser les variables sur le Serial Monitor -------

        Serial.print("Kp= ");
        Serial.println(Kp);
        Serial.print("        Kd= ");
        Serial.println(Kd);
        Serial.print("              Ki= ");
        Serial.println(Ki);
        Serial.print("                      mesure= ");
        Serial.println(mesure);
        Serial.print("                                Err= ");
        Serial.println(Err);
        Serial.print("                                        integrale= ");
        Serial.println(integrale);
        Serial.print("                                                  sortie_PID= ");
        Serial.println(sortie_PID);
        Serial.print("                                                              sortie_moteur= ");
        Serial.println(sortie_moteur);

        last_mesure=mesure;   //affecter la mesure à  la mesure précédente
        delay(pe);          //attendre pendant "pe=10ms" periode d'échantillonage
  }
```

Crédit d'illustration

http://www.zappedlab.com/wp-content/uploads/2009/05/scooty_v01_a_400x600.gif
http://www.planet-techno-science.com/wp-content/upLoads/wheelie.jpg
http://www.zonerobotique.com/images/dossiers_moyen/neorobot1223293475.jpg
http://grangeramp.com/shop/images/50w_chassis_steel.gif
http://www.hvwtech.com/products/70/22040_PV.jpg
http://www.vulgarisation-informatique.com/images/architecture/processeur/processeur.jpg

Photos prises par nous-mêmes

Bibliographie

Begining Arduino by Michael McROBERTS (2010)

Atelier Arduino – Initiation à la mise en œuvre matérielle et logicielle de l'Arduino par Jean-Noël MONTAGNE

Arduino Starter Kit Manual - A Complete Beginners Guide to the Arduino by M. R. McROBERTS (2009)

Site du Clubelek de l'INSA Lyon :
http://clubelek.insa-lyon.fr/joomla/fr/base_de_connaissances/informatique/asservissement_et_pilotage_de_robot_autonome_introduc_5.php

Wikipédia : http://fr.wikipedia.org/wiki/R%C3%A9gulateur_PID