Chapitre 1

Véhicule

1.1 Description du véhicule

La voiture sur laquelle nous nous basons est un model réduit télécomandé de type 4x4. Le tableau suivant est un inventaire de ses caractéristiques en son état actuel :

Grandeurs	longueure	35 cm
	longueure (centre de roue à centre de roue)	17 cm
	largeure	22 cm
	largeure (centre de roue à centre de roue)	17 cm
	hauteur au sol	4 cm
Moteur de propulsion	Voltage de marche	˜5V - ˜10V
	Courant min (roue libre)	˜2A
	Courant max (roue bloquée)	˜3A
Servo de guidage	Fabriquant	Corona
	Model	Metal gear DS558HV
	Voltage de marche	˜6V - ˜7.4V
	Courant	300mA - 400mA
	Charge maximale	12kg - 14kg

1.1.1 System directionel initial

Dans son état initial, le système de guidage pouvait se comparer à un servo très rudimentaire. Il en comptait toutes les caractéristiques, mais en un état simplifié, en particulier le système de positionnement. Dans un servo classique, il s'agit d'un potentiomètre (variateur de résistance) qui permet de savoir en tout moment la position de la corne. Par contre, dans le cas de la voiture, un système de balais (voir image) assure cette tâche. Il en résulte une identification de position très basique : gauche, droit ou droite.

(insérer images)
Sachant que nous avons retiré lélectronique de la voiture, il nous restait deux possibilités : utiliser le système de guidage rudimentaire, mais déjà en place ou tout remplacer avec un servo conventionel.
Après beaucoup de temps perdu à tenter de contrôler le guidage de base avec notre électronique importée, on décide de passer à un servo. On achète un puissant servo de haute qualité chez une connaissance qui en avait commandé un gros lot pour la modique somme de 10.- CHF.

1.1.2 Remplacement du System de guidage

Une instalation fiable d'un objet étranger dans un ensemble usiné tel la voiture n'est pas tâche facile. Il fallait pourtant que le résultat final soit solide, si l'on voulait compter dessus. C'est pour cela que nous avons créé une base en contreplaqué pour y loger le Servo.

Ce montage permet de retirer le servo en cas de besoin, donc de pouvoir le remplacer. En effet, la plaque supérieure est fixée au moyen de vices à bois. Le servo est accompagné, dans son logement, d'un morceau de gomme adhérante (morceau de chambre à air). La structure épouse les formes de la voiture pour un maximum de rigidité. La transmission de la force aux roues se fait par l'intermédiare d'une tige de métal. Celle-ci, est fixée à la corne du servo et à une boucle soudée à l'ancien axe de transmission. Nous utilisons justement l'ancien axe de transmission pour une raison développée plus tard (voir : http ://en.wikipedia.org/wiki/Ackermann_steering_geometry sur "Ackerman steering").

1.1.3 contrôle du servo

Le contrôle d'un servo est un devoir qu'un microcontrôleur tel que l'arduino effectue avec aise. On peut indiquer à un servo de se rendre vers un de ses $180°$ de liberté en lui envoyant un signal électrique dit modulé. Ce signal est modulé d'une manière comprhénsible pour le servo. L'illustration suivante pourrait d'avantage éclairer le lecteur :

FIGURE 1.1 – Schéma de la modulation du signal électrique [1]

Comme l'on peut voir, le dit signal, est formé de hauts et de bas. Lorsqu'il est "bas", cela veut dire que la tension est basse ou égale à 0V. Lorsqu'il est "haut", cela veut dire que la tension est a une valeure définie auparavant, standard, différente de 0V. Dans notre cas, "haut" est à 5V (standard pour le modélisme et l'électronique en général). On appelle la période pendant laquelle le signal est "haut" une pulsation.

Chaque début de pulsation est séparé par un temp bien défini de 20ms. Ce qui peut varier d'une pulsation à l'autre, donc ce qui informe le servo en quel angle il doit se positionner, est la longueur de la pulsation. Comme indiqué, celle-ci peut varier de 1ms à 2ms.

1.1.4 Programmation pour contrôler un servo

Le programme suivant est un exemple proposé dans la section "apprentissage" du site officiel d'Arduino. Il utilise la librairie "Servo" installée avec l'IDE Arduino. Ce que font les méthodes de cette classe est de produire un signal comme celui discuté à la section précédente et l'émettre par le pin choisi.

Les commentaires sont en anglais, mais le code sera expliqué tout de suite :

```
// Sweep
// by BARRAGAN <http://barraganstudio.com>
// This example code is in the public domain.


#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

int pos = 0;    // variable to store the servo position

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}


void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position stored
                                     // in variable 'pos'
    delay(15);                       // waits 15ms for the servo
                                     // to reach the position
  }
```

1. Wiki sur les Servos : http ://en.wikipedia.org/wiki/File :ServoPwm.png

```
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);
    delay(15);
  }
}
```

Commentaires :

On commence par inclure la class Servo, puis on crée un objet *Servo*. Dans la partie emphsetup du programme, on lie l'objet *myservo* au pin 9 de l'arduino. Ensuite, dans la partie *loop*, on fait varier la position du servo grâce à la méthode *write*. Un délai(le programme s'arrête en ce point) de 15ms pour permettre au servo d'atteindre la position demandée. Etant donné que cette opération est ittérée plusieurs fois au moyen d'une boucle *for*, on pourra voir le servo décrire un mouvement de balayage.

D'ailleur, ce programme est très pratique pour tester le fonctionement d'un servo.

1.2 Moteur de propulsion

1.2.1 Descriptif

Le moteur de propulsion, au contraire du servo, est l'original de la voiture. Ses caractéristiques électriques (données que nous avon mesuré à l'aide d'un multimètre du gymnase) se trouvent à la section (1.1). Le moteur est muni d'une boite à vitesse ainsi qu'un différentiel. Nous avons estimé qu'il aurait été inutilement compliqué d'y apporter quelques modifications qu'ils soient. La configuration déjà existente, à l'exception de l'électronique, subvient tout à fait à nos besoins.

1.2.2 H-bridge

Faire tourner l'axe d'un moteur électrique pose peu de problèmes. Il suffit de connecter l'un des pôles à la tention positive et l'autre à la négative. Ceci fera tourner l'axe du moteur dans un sens. Si vous souhaitez le faire tourner dans le sens inverse, il suffit d'échanger les fils électriques aux pôles du moteur.

Le problème suivant se pose alors : comment inverser le sens de marche du moteur sans intervention manuelle ?

La réponse est donnée par un astucieux circuit composé de transistors. Il permet, au moyen de deux signaux actionnant les transistors, de contrôler le sens du courrant passant dans le moteur.

Le schéma suivant pourra d'avantage éclairer le lecteur :

Explications :

Tout d'abord, que sont des transistors NPN et PNP ? La différence pratique entre ces deux types de transistors est que l'un demande un signal déclencheur haut pour être ouvert tandis que l'autre en demande un bas ou la terre. Dans ce cas, on utilise deux types de transistors différents (en grande partie pour clarifier le schéma), mais

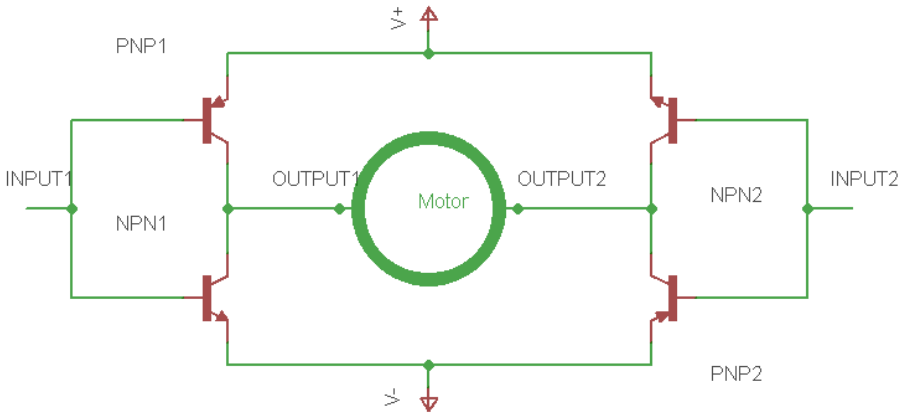

FIGURE 1.2 – Schéma d'un H-bridge

l'on pourrait très bien utiliser uniquement des transistors du même type. On obtiendrait le même résultat en connectant les *INPUT* à des transistors diagonalement opposés. [2]

On peut voir que selon le *INPUT*, on obtiendra des tensions au bornes du moteur ou *OUTPUT* variables.

1.2.3 Table de Vérité

Rédigeons un tableau de vérité pour mieux illustrer la situation, où H (high) signifie haut et L (low) signifie bas :

INPUT1	*INPUT2*	Tension
H	L	$OUTPUT1 = OUTPUT2$
L	H	$OUTPUT1 = OUTPUT2$
H	H	$OUTPUT2 > OUTPUT1$
L	L	$OUTPUT1 > OUTPUT2$

TABLE 1.1 – Table de vérité accompagnant le schéma du H-bridge (fig.(1.2))

Quand les signaux *INPUT* sont opposés, le moteur est à l'arrêt. Quand ils sont équivalents, le moteur est en marche, dans un sens ou dans l'autre.

1.2.4 Code

Maintenant que nous avons compris le fonctionement d'un H-bridge, en particulier les conséquences d'*INPUT* différents, on pourra créer un petit programme Arduino pour contrôler un moteur.

2. Explications inspirées par les informations trouvées sur le site suivant : http ://www.robotroom.com/BipolarHBridge.html

Le code suivant a été rédigé par Eric :

```
#define A 12 // define input pin "A"
#define B 11 // define input pin "B"
#define E 10 // define enable pin "E"

byte i;

void setup(){

// tous les pins sont des "OUTPUT", ils vont contoler le moteur

  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(E, OUTPUT);
}

void loop(){

// boucles faisant varier la vitesse lineairement en avant et en arriere

  for(i = 0; i < 200; i++){
    forward(A, B, E, i);
    delay(50);
  }
  for(i = 200; i > 0; i--){
    forward(A, B, E, i);
    delay(50);
  }
  for(i = 0; i < 200; i++){
    backward(A, B, E, i);
    delay(50);
  }
  for(i = 200; i > 0; i--){
    backward(A, B, E, i);
    delay(50);
  }
}

//methode globale pour controler le moteur, fait appel aux
//methodes halt, backward et forward.
void motor(int speedo){
  if (abs(speedo) > 255){
    halt(A,B);
    break;
  }
```

```
  else if (speedo > 0){
   forward(A, B, E, speedo);
  }
  else if (speedo < 0){
    backward(A, B, E, speedo);
  }
  else if (speedo == 0){
    halt (A, B);
  }
}

//fonction faisant tourner le moteur en avant ou l'inverse d'en arriere.
void forward(byte pin1, byte pin2, byte pinEnable, byte speedo){
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, LOW);
  analogWrite(pinEnable, speedo);
}

//fonction faisant tourner le moteur en arriere ou l'inverse d'en avant.
void backward(byte pin1, byte pin2, byte pinEnable, byte speedo){
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, HIGH);
  analogWrite(pinEnable, speedo);
}

//fonction servant a arreter le moteur.
void halt(byte pin1, byte pin2){
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
}
```