

# UGV bon marché

Sven Borden

Eric Brunner

Travail de maturité

Gymnase de Morges

14 juin 2013

# Première partie

## 1 Avant-propos

Ce dossier est le résultat de onze mois de recherches effectuées dans le cadre du travail de maturité du gymnase de Morges. Ayant déjà quelques notions en informatique, nous nous sommes redirigés vers un domaine parallèle, la robotique. Le choix de ce sujet est issu de...

## 2 Remerciements

Ce projet n'aurait pu aboutir sans l'aide de nombreuses personnes. Voici l'occasion de les remercier : Mr. Denis Rochat et Mr. Phillipe Rochat pour leur disponibilité, leurs renseignements ainsi que les prêts matériels. Mr. Frederic Genevey ainsi que son site edurobot.ch pour avoir promu notre projet sur son site internet. Mme Pauline Pidoux pour nous avoir aidé lors de la rédaction de ce travail et nous tenions aussi à remercier Stefano Varricchio, du Laboratoire LIS pour ses informations très utiles.

## Résumé

Chaque chapitre de ce dossier traite d'une partie du drone, le premier expliquera la mécanique et l'électronique du véhicule, le deuxième chapitre traitera le *Hardware* nécessaire au bon fonctionnement de l'UGV ainsi que son fonctionnement. Le troisième chapitre parlera du *Software* utilisé dans le *Hardware* et le dernier chapitre concernera [à venir]

## Table des matières

<b>I</b>	<b>2</b>
<b>1 Avant-propos</b>	<b>2</b>
<b>2 Remerciements</b>	<b>3</b>
<b>3 Introduction</b>	<b>6</b>
<b>II La mécanique et l'électronique</b>	<b>6</b>
<b>4 Description du véhicule</b>	<b>6</b>
4.1 System directionel initial . . . . .	7
4.2 Remplacement du System de guidage . . . . .	8
4.3 contrôle du servo . . . . .	9
4.4 Programmation pour contrôler un servo . . . . .	10
<b>5 Moteur de propulsion</b>	<b>12</b>
5.1 Descriptif . . . . .	12
5.2 H-bridge . . . . .	12

5.3	Table de Vérité . . . . .	13
5.4	L293d . . . . .	14
5.5	Code . . . . .	15

### **III Hardware 18**

#### **6 Choix du hardware 18**

6.1	Arduino . . . . .	18
6.1.1	Choix du type d'Arduino . . . . .	19
6.2	Raspberry Pi . . . . .	19
6.2.1	Choix de l'OS . . . . .	20

### **IV Software 20**

## Table des figures

1	Schéma de la modulation du signal électrique [2] . . .	9
2	Schéma d'un H-bridge . . . . .	13
3	Schéma du l293d. <sup>1</sup> . . . . .	14
4	Informations tirées des fiches techniques publiées par TI [3] . . . . .	15

## Liste des tableaux

1	Table de vérité accompagnant le schéma du H-bridge (fig.(2)) . . . . .	14
---	---	----

### 3 Introduction

Le but de ce projet était de construire un véhicule roulant que l'on peut commander à distance. Plus qu'une simple voiture télécommandée, ce drone est capable d'être contrôlé sans avoir une vue directe sur celui-ci, car il possède des capteurs ainsi qu'une caméra. Ce types d'engins se nomment *UGV (Unmanned Ground Vehicle)* soit : véhicule roulant commandé à distance. Surtout utilisés dans l'armée, les modèles qu'on peut trouver sur le marché sont très coûteux, ils varient entre trois cents et mille trois cents francs. Notre but est donc de pouvoir construire un appareil semblable pour moins de cent septante-cinq francs.

## Deuxième partie

# La mécanique et l'électronique

### 4 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	longueur	35 cm
	longueur (centre de roue à centre de roue)	17 cm
	largeur	22 cm
	largeur (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

#### 4.1 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 l'électronique de la voiture, il nous restait deux possibilités : utiliser le système de guidage rudimentaire, mais déjà en place ou tout remplacer avec un servo conventionnel.

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.

## 4.2 Remplacement du System de guidage

Une installation 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 adhésive (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'intermédiaire 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](http://en.wikipedia.org/wiki/Ackermann_steering_geometry) sur “Ackerman steering”).

### 4.3 contrôle du servo

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

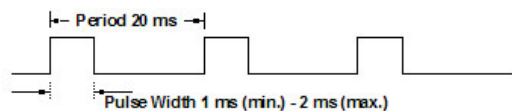


FIGURE 1 – Schéma de la modulation du signal électrique [2]

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 à une valeur 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 temps 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.

#### 4.4 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
```

```
}
```

```
void loop()
```

```
{
```

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

```
    for(pos = 180; pos >= 1; pos -= 1)    // goes from 180 degrees to 0
    {
        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 itérée plusieurs fois au moyen d'une boucle `for`, on pourra voir le servo décrire un mouvement de balayage.

D'ailleurs, ce programme est très pratique pour tester le fonctionnement d'un servo.

## 5 Moteur de propulsion

### 5.1 Descriptif

Le moteur de propulsion, au contraire du servo, est l'original de la voiture. Ses caractéristiques électriques (données que nous avons mesuré à l'aide d'un multimètre du gymnase) se trouvent à la section (4). Le moteur est muni d'une boîte 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 existante, à l'exception de l'électronique, subvient tout à fait à nos besoins.

### 5.2 H-bridge

Faire tourner l'axe d'un moteur électrique pose peu de problèmes. Il suffit de connecter l'un des pôles à la tension 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 courant passant dans le moteur.

Le schéma suivant pourra d'avantage éclairer le lecteur :

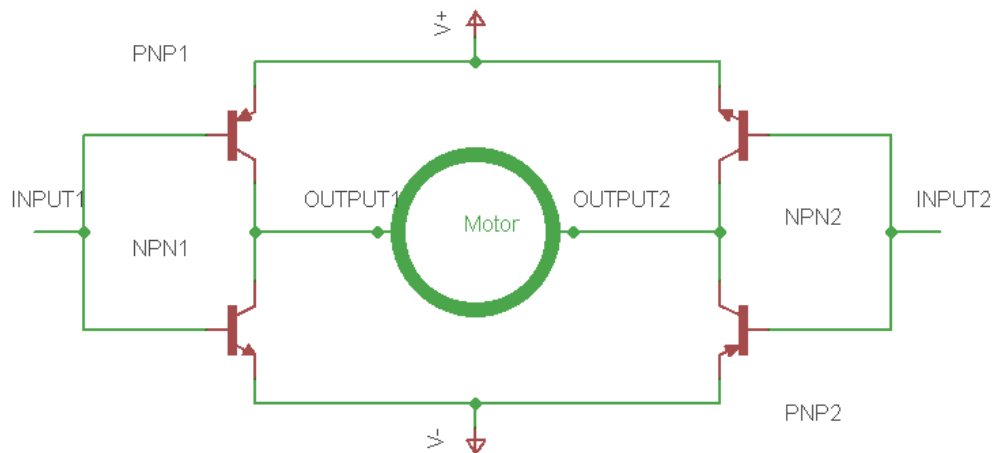


FIGURE 2 – Schéma d'un H-bridge

### 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 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[4].

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

### 5.3 Table de Vérité

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

<i>INPUT1</i>	<i>INPUT2</i>	Tension
H	L	$OUTPUT1 = OUTPUT2$
L	H	$OUTPUT1 = OUTPUT2$
H	H	$OUTPUT2 > OUTPUT1$
L	L	$OUTPUT1 > OUTPUT2$

TABLE 1 – Table de vérité accompagnant le schéma du H-bridge (fig.(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.

## 5.4 L293d

Nous avons décidé de ne pas construire notre propre H-Bridge, mais plutôt exploiter un chip qui sert le même but. Le chip en question est dénommé L293D. Pour commencer, une synthèse de la feuille de spécifications :

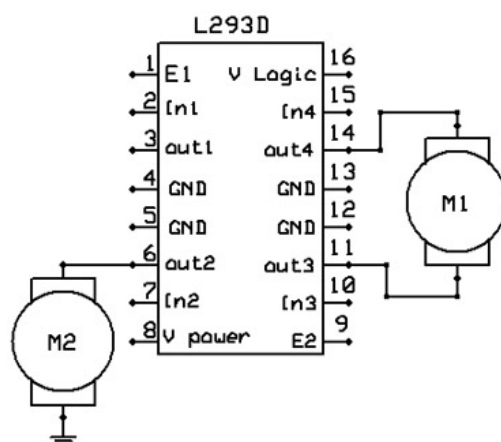


FIGURE 3 – Schéma du l293d.<sup>2</sup>

### Spécifications max :

2. <http://ikalogic.cluster006.ovh.net/wp-content/uploads/l293d.jpg>

$V_{logic}$	36V
$V_{power}$	36V
Peak output current	$\pm 2A$
Dissipation de chaleur à température ambiante (25°)	2075mW
Température max	70°

FIGURE 4 – Informations tirées des fiches techniques publiées par TI [3]

## 5.5 Code

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

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 contrôler 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 a*

```
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*

```
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);
}
```

## Troisième partie

# Hardware

## 6 Choix du hardware

Pour réaliser ce projet, nous avons dû faire des choix au niveau du hardware. Notre choix s'est porté sur deux système. Le premier, l'Arduino, est un microcontrôleur qui permet de contrôler presque ce qu'on veut grâce à un langage de programmation proche du C. Le second est le Raspberry Pi, qui est un ordinateur bon marché (trente-cinq francs) qui est récemment sorti sur les marchés.

### 6.1 Arduino

L'Arduino [5] est un microcontrôleur *Open Source*, ce qui veut dire que tout le monde peut non seulement avoir accès aux plans et aux codes, mais peut aussi les modifier. Ce microcontrôleur se programme avec un langage proche du C.

### 6.1.1 Choix du type d'Arduino

## 6.2 Raspberry Pi

Le Raspberry Pi[1] est un ordinateur de la taille d'une carte de crédit sur lequel on peut installer différents systèmes d'exploitations dérivés de UNIX/Linux. Le Raspberry Pi est acheté nu, c'est-à-dire que cet ordinateur ne possède pas d'écran, ni de clavier ou de souris, néanmoins le Raspberry Pi possède plusieurs ports où on peut brancher écran (via l'interface HDMI ou Composite), un câble Ethernet et presque ce qu'on veut grâce aux deux ports USB. Le Raspberry Pi est très intéressant non pas du point de vue de sa puissance calculatoire, mais du point de vue rapport qualité-prix. En effet, pour trente-cinq francs, il a les caractéristiques suivantes :

1. poid de 45g environ
2. Processeur ARM1176JZF-S (ARMv6) 700MHz Broadcom 2835
3. 512Mo de RAM (sur la version B, soit celle que nous avons choisie)
4. 2 sorties vidéo (HDMI et Composite)
5. Sortie audio stéréo Jack (3.5mm) (le son passe aussi par le HDMI en sortie 5.1)
6. Ecriture et lecture possible sur une carte mémoire sous forme de carte SD (supporte les formats : SDHC, MMC et SDIO)
7. 2 ports USB 2.0 et 1 port Ethernet
8. Alimentation par câble micro USB
9. Faible consommation (5W, 5V, 1A)
10. Communication possible via les Pin GPIO

11. Décodeur permettant de lire le FullHD 1080p

12. API logiciel vidéo (OpenGL)

Bien qu'à première vue la Framboise ne semble pas très performante, il faut prendre en compte son prix qui est bas, sa taille ainsi que les possibilités qui sont presque infinies.

### 6.2.1 Choix de l'OS

Plusieurs types de systèmes d'exploitations fonctionnant sur le Raspberry Pi existent. Pour n'en citer que quelques-uns :

- |              |             |
|--------------|-------------|
| – Android    | – Debian    |
| – Firefox OS | – ArchLinux |
| – RISC OS    | – Gentoo    |
| – Fedora     | – Raspbian  |

Notre choix à été porté sur Raspbian, qui est un dérivé de Debian, pour plusieurs raisons. Tout d'abord, cet OS à été développé spécialement pour le Raspberry Pi et il est donc continuellement développé par la communauté du Raspberry Pi. Cet OS étant basé sur un environnement Linux, cela offre un grand nombre de liberté afin de travailler dessus. Raspbian est aussi gratuit, se qui rentre en compte puisque nous essayons de réduire les coûts.

## Quatrième partie

# Software

## Références

- [1] the free encyclopedia From Wikipedia. Raspberry pi, ??
- [2] the free encyclopedia From Wikipedia. Servo control, Octobre 2012.
- [3] Texas Instrument. L293, l293d quadruple half-h drivers, Juin 2002.
- [4] Robot Room. H-bridge motor driver using bipolar transistors, ??
- [5] Arduino Team. Arduino spec., ??