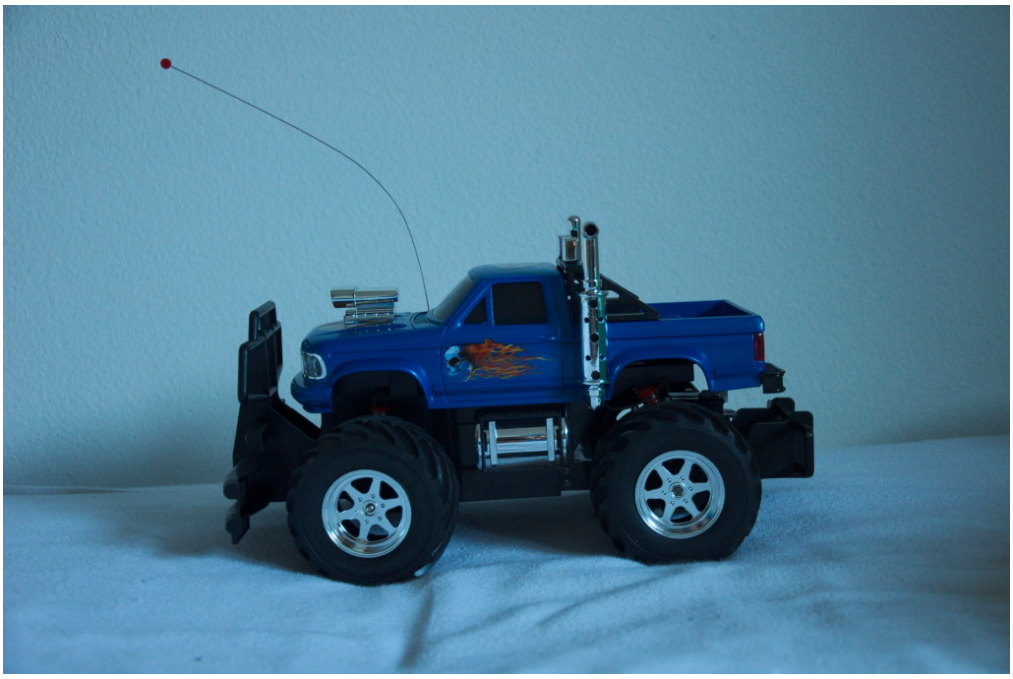
UGV (Unmanned Ground Vehicle)

Sven Borden Eric Brunner

Travail de maturité Gymnase de Morges

20 juin 2013



Avant-propos

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

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 promouvu 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'éléctronique 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 denier chapitre concernera [à venir]

Table des matières

Table des figures

Liste des tableaux

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'amée, 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.

Première partie

La mécanique et l'électronique

1 Description du véhicule

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

1.1 Système 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 (résistance variable) 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, tout 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 plus conventionel.
Après beaucoup de temps perdu à tenter de contrôler le guidage de base avec notre électronique importée, nous avons décidé de passer à un servo. Nous avons achèté un puissant servo de haute qualité chez

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
	Modèle	Metal gear DS558HV
	Voltage de marche	˜6V - ˜7.4V
	Courant	300mA - 400mA
	Charge maximale	12kg - 14kg

TABLE 1 – Caractéristiques du véhicule

une connaissance qui en avait commandé un gros lot pour la modique somme de 10.- CHF.

1.2 Remplacement du Système de guidage

Une instalation fiable d'un objet étranger dans un ensemble usiné tel la voiture n'est pas une tâche facile. Il fallait pourtant que le résultat final soit solide, si l'on voulait pouvoir 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 vis 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 métallique. Celle-ci est fixée à la corne du servo et possède une boucle soudée à l'ancien axe de transmission. Nous utilisons justement l'ancien axe de transmission pour une raison dévelopée plus tard. [1]

1.3 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 $180°$ de liberté en lui envoyant un signal électrique dit modulé. Ce signal est modulé d'une manière compréhénsible pour le servo. L'illustration suivante pourrait d'avantage éclairer le lecteur :

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, le "haut" est à 5V (standard pour le modélisme et l'électronique en

1. voir : http ://en.wikipedia.org/wiki/Ackermann_steering_geometry sur "Ackerman steering"

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

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.

1.4 Programmation pour contrôler un servo

Le programme se trouvant en annexe (A.2) est un exemple proposé dans la section “apprentissage”[4] 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 en l’émettant par un des pins de l’arduino capable de cette modulation.

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

2 Moteur de propulsion

2.1 Descriptif

Le moteur de propulsion, au contraire du servo, n'a pas été changé. Ses caractéristiques électriques (données que nous avon mesuré à l'aide d'un multimètre du gymnase) se trouvent à la section (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 des modifications. La configuration déjà existente, à l'exception de l'électronique, subvient tout à fait à nos besoins.

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 tension négative. Ceci fera tourner l'axe du moteur dans un sens. Si vous souhaitez le faire tourner dans le sens inverse, il vous suffira 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 de-

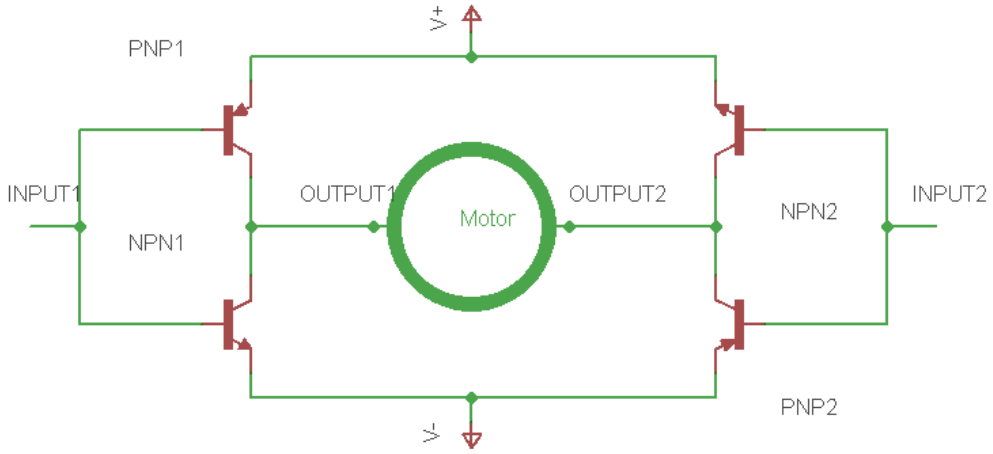


FIGURE 2 – Schéma d'un H-bridge

mande 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[3].

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

2.2.1 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 2 – 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.

2.2.2 L293d

Nous avons décidé d'utiliser un H-Bridge qui se nomme L293d.

Deuxième partie

Hardware

3 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 microcontrolleur qui permet de contrôler presque ce qu'on veut grâce à un language de programation 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.

3.1 Arduino

L'Arduino [5] est un microcontrolleur *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 microcontrolleur se programme avec un language proche du C.

3.1.1 Choix du type d'Arduino

Pour ce projet, nous avons choisi l'Arduino Uno, c'est l'Arduino de base. En effet, Arduino possède de nombreux produits afin de subvenir aux différents besoin. Nous avons hésité à prendre l'Arduino Mega, mais les avantages qu'il offre ne sont pas très utiles pour notre projet. L'Arduino Uno est un microcontrolleur qui ressemble à ceci :

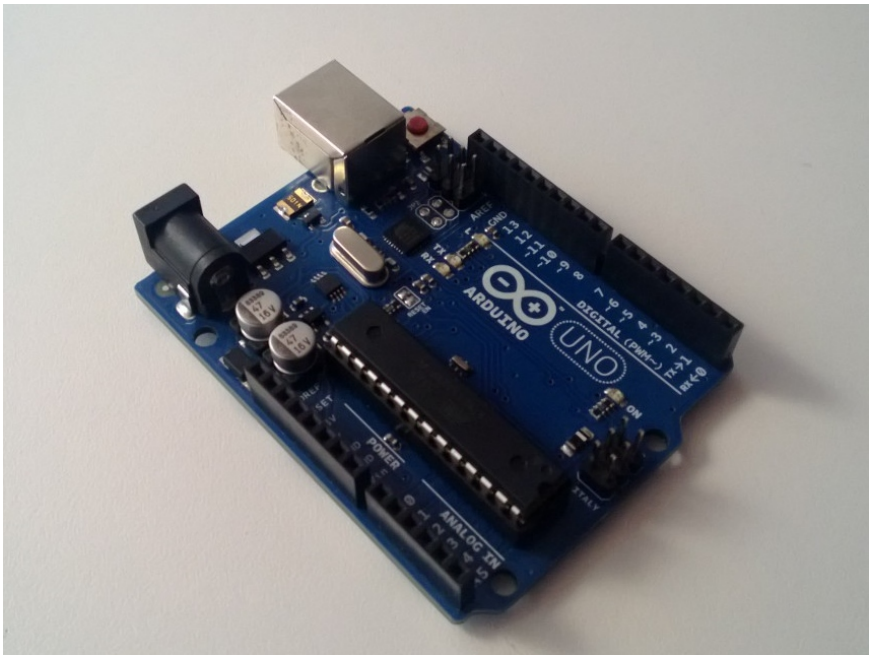


FIGURE 3 – Arduino Uno R3

L'Arduino Uno possède les caractéristiques suivantes :

1. 14 pin digitaux (signal 1 ou 0) où on définit INPUT ou OUPUT dont 6 d'entres eux peuvent être analogique en OUTPUT.
2. 6 analogiques INPUT
3. Connection USB
4. Oscillateur à quartz cadencé à 16MHz
5. Courant continu sur les pin digitaux (40mA)
6. Une sortie 3.3V et une sortie 5V en courant continu
7. 32KB de mémoire flash
8. 2KB de RAM
9. Microcontrôleur ATmega328

Bien qu'à première vue on pourrait se dire qu'il n'est pas très performant. Il faut se rappeler qu'il ne dois pas faire fonctionner un système d'exploitation, mais qu'il dois simplement traiter des mesures et envoyer des signaux digitaux ou analogiques. L'avantahe de l'Arduino Uno est qu'il n'y a pas besoin de faire de soudure, les pin sont directements accessibles grâce à des trous (les pin) où on peut y glisser un fil métallique. Dans le cas où nous souhaiterions commercialiser notre produit, nous choisirons probablement l'Arduino mini car il est encore plus petit et nous pouvons directement souder les fils sur le microcontrôleur.

3.2 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 interressant 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.

3.2.1 Choix de l'OS

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

- Androïd
- Firefox OS
- RISC OS
- Fedora
- Debian
- ArchLinux
- Gentoo
- Raspbian

Notre choix à été porté sur Raspbian, qui est un dérivé de Debian, pour plusieurs raisons. Tout d'abord, cet OS à été dévellopé spécialement pour le Raspberry Pi et il est donc continuellement dévellopé 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.

Troisième partie

Software

A Sketchbook

A.1 Sketch exemple pour le moteur

```
#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 a

  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);
}
```

A.2 Sketch exemple pour le servo

[4]

```
// 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 positi
                                     // 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 classe Servo, puis on crée un objet *Servo*. Dans la fonction *setup* du programme, on lie l'objet *myservo* au pin 9 de l'arduino. Ensuite, dans la fonction *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.

Références

[1] the free encyclopedia From Wikipedia. Raspberry pi, ??

[2] the free encyclopedia From Wikipedia. Servo control, Octobre 2012.

[3] Robot Room. H-bridge motor driver using bipolar transistors, ??

[4] Barragan Studio. Servo sweep, Septembre 2010.

[5] Arduino Team. Arduino spec., ??