

IUT de St-Malo, département Réseau & Télécoms

# Rapport final de projet

R&T Enterprise – Drone terrestre Wifi



Réalisation : **Clément Courtel**

Client : **L. Parize**

Juin 2016

## Table des matières

Introduction.....	3
Les cartes et leur programme .....	5
La Raspberry Pi : le mini-ordinateur .....	5
Du côté de l'Arduino mini 0.....	6
Concernant l'Arduino mini 1 .....	7
Les différents modules .....	8
Les moteurs .....	8
Le retour vidéo .....	10
Les servomoteurs .....	10
Le capteur ultrason.....	10
Le Wifi.....	11
L'électronique.....	13
Les composants utilisés .....	13
Le circuit imprimé.....	16
Les protocoles de communications internes .....	17
Conclusion .....	19
Annexes 1 : Les codes sources (version avril 2016) .....	20
Le code complet de l'Arduino 0.....	20
Le code complet de l'Arduino 1.....	21
Le code complet du Raspberry Pi .....	22
Annexes 2 : Divers .....	24
Les fiches composants.....	24

## Table des illustrations

Illustration 1: Diagramme de Gannt initial .....	3
Illustration 2 : Diagramme de Gannt final.....	3
Illustration 3 : Touches clavier de pilotage.....	5
Illustration 4 : Principe du pont-H .....	8
Illustration 6 : Composant L293d .....	9
Illustration 7 : Ultrason HC-SR04.....	11
Illustration 8 : Champ d'action du capteur.....	11
Illustration 9 : résultat de la commande lsusb .....	11
Illustration 10 : Point d'accès physique.....	12
Illustration 11: Démarrage du réseau "Drone" .....	13
Illustration 12 : Schéma d'un optocoupleur.....	14
Illustration 13 : régulateur de tension LM2596 .....	14
Illustration 14 : Schéma du relais .....	14
Illustration 15 : Création de la carte sous Proteus .....	16
Illustration 16 : Lecture d'un bit par I <sup>2</sup> C .....	17
Illustration 17 : Schéma du protocole SPI .....	17

## Introduction

Dans le cadre des projets tuteurés de première année à l'IUT St-Malo, département R&T, j'ai décidé de réaliser un Drone Terrestre Wifi, avec pour base un jouet télécommandé basique, nommé R&T Enterprise.

Ce projet s'inscrit dans de nombreux enjeux comme celui du recyclage de matériels, à mettre à jour vers un produit intelligent, connecté et dans celui de rendre le projet souple et évolutif. L'objectif de ce projet est de réaliser un robot terrestre pilotable par ordinateur, en local ou à distance, avec du matériel recyclé, ou bon marché, et de rendre tous les travaux Open Source (libre de droits). Basé sur des cartes programmables grands publics, Raspberry Pi et Arduino, et facilement programmable, le lecteur n'aura pas besoin de connaissances particulières pour lire ce document.

Un projet comme celui-ci nécessite beaucoup de travail, aussi bien sur la recherche théorique de solutions que la mise en place technique. Il faut passer par une organisation du temps de travail. Les principales tâches sont la programmation des cartes Raspberry Pi et Arduino puis mettre en place les composants associés. J'ai réalisé un diagramme de temps au commencement du projet en indiquant les grands axes à suivre puis l'ai comparé aux objectifs réalisés en fin d'année.

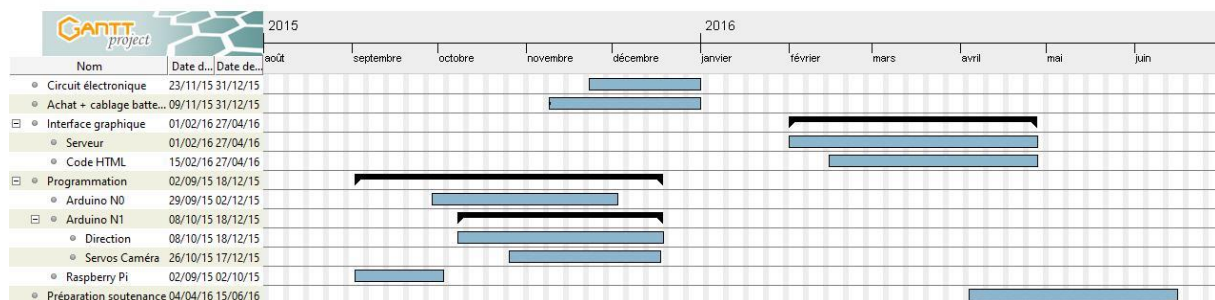


Illustration 1: Diagramme de Gantt initial

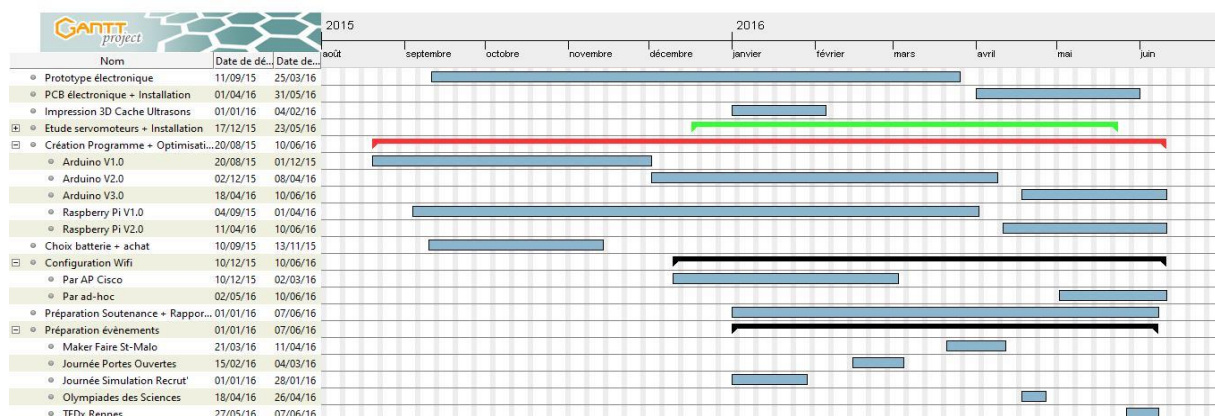


Illustration 2 : Diagramme de Gantt final

La comparaison des 2 diagrammes de Gantt montre quelques différences :

- La programmation initiale des cartes correspond aux programmes version 1.0 des cartes programmables. Cette première version fonctionnait et répondait aux exigences de bases, cependant je souhaitais ajouter des modules et des composants au drone.

- La partie « interface graphique » a été recherchée mais pas mise en place. Il existe de nombreux moyens de coder une interface graphique. Le langage pourrait être C++ avec l'utilitaire Qt.
- La tâche de réalisation d'un PCB de circuit électronique a été décalé car il fallait maîtriser le logiciel Proteus.

La réalisation de ce projet a nécessité la réunion de plusieurs acteurs sans qui ce projet n'aurait pas évolué. Je tiens à remercier toutes les personnes et sponsors qui ont aidé dans ce projet d'année, et plus particulièrement, M. L Parize : tuteur du projet, l'IUT de Saint-Malo pour la mise à disposition de matériels, l'association Digital Saint-Malo pour leur aide financière et l'IUT de Rennes pour l'impression du circuit électronique.

## Les cartes et leur programme

Pour réaliser des actions avec les cartes programmables, il faut leur envoyer un code qu'elles vont exécuter. Deux langages de programmations sont expliqués dans cette partie : le langage C++ pour les cartes Arduino, et le langage python pour la Raspberry Pi. Les différents programmes ont été écrits avec l'IDE (Environnement de développement intégré) Arduino, et l'application Notepad++ pour le python. Cette partie est dédiée à l'explication du matériel, des moyens mis à disposition et, ensuite, des programmes écrits pour utiliser ce matériel.

### La Raspberry Pi : le mini-ordinateur

La carte Raspberry Pi est une carte grand public largement répandue dans le monde depuis 2012. Son principe est de démocratiser la programmation et l'interaction homme-machine. Débutant ce projet sans compétences dans le domaine de l'électronique et de la programmation, je me suis orienté vers ce produit.

#### Ses caractéristiques matérielles

Depuis son lancement, la Raspberry a vu ses caractéristiques évoluer. À ce jour, la version 3 possède une grande puissance de calcul et une carte wifi intégrée. La version utilisée dans le drone est la 2 dont la puissance suffit à son utilisation. Les caractéristiques sont :

- Processeur 900MHz quad-core ARM Cortex-A7 CPU,
- 1GB RAM,
- 4 USB ports,
- 40 GPIO pins,
- Ethernet port,
- Micro SD card slot, ...

#### Son programme v2.0

Le programme de la Raspberry Pi permet de faire l'interface Utilisateur <-> drone (via Arduino). Les premières versions du programme permettaient au pilote d'entrer des valeurs pour effectuer les actions. À ce jour, la dernière version du programme permet de convertir un caractère entré par l'utilisateur, en valeur à envoyer à l'Arduino. Cette version inclut le pilotage par incrémentation de valeurs. C'est-à-dire qu'un caractère du clavier ajoute un indice à une action. Par exemple, plus la touche 'z' est entrée, plus la voiture roulera rapidement. Il faut savoir que les différentes actions sont réalisables via des caractères placés judicieusement sur un clavier Azerty.

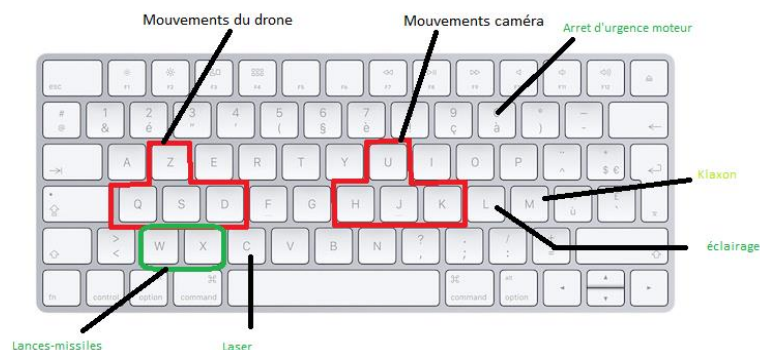


Illustration 3 : Touches clavier de pilotage

Le programme du Raspberry se compose de caractères, envoyés par l'utilisateur, de plage de valeurs, qu'il envoie aux Arduino, et d'acquittements, renvoyé par les Arduino après leurs actions.

Chaque module (moteur, direction, ...) est caractérisé dans le programme par un tramage de valeurs I<sup>2</sup>C pour pouvoir gérer les indices de modules indépendamment (vitesse moteur, angle servomoteurs,). On peut résumer ces informations dans un tableau comme ceci :

Caractère envoyé	Commande réalisée	Plage de valeurs	Valeur d'acquiescement
z	Avancer	301 - 550 (50 - 235)	11
	Arrêt par Ultrason avant		12
s	Ralentir / Reculer	50 - 299 (50 - 235)	13
q	Direction gauche	600 - 700 (40° à 140°)	14
d	Direction droit		
o	Arrêt d'urgence moteur	0 ou 300	10
u	Caméra bas	1100 - 1280 (0° à 180°)	16
j	Caméra haut		
h	Caméra droit	1300-1480 (0° à 180°)	18
k	Caméra Gauche		
w	Lancement missile 1	22 (0°)	20
x	Lancement missile 2	23 (180°)	21
c	Laser On	24	23
	Laser Off	24	24
m	Klaxon	21	22
l	Éclairage avant ON	26	25
	Éclairage avant OFF	26	26

Le programme python (en intégralité page 22) utilise des bibliothèques spécialisées. Le paquet SMbus permet d'utiliser les GPIO 2 et 3 pour la communication I<sup>2</sup>C avec les Arduino. SMbus contient plusieurs fonctions :

« bus.write\_word\_data(address1,0x00,m) » permet d'envoyer une donnée de plusieurs octets « m » vers une carte esclave « address1 » ;

« bus.read\_byte(address1) » permet de demander une donnée à la carte esclave, comme un acquiescement par exemple.

## Du côté de l'Arduino mini 0

Le drone possède deux Arduino dont un spécialisé dans la gestion du moteur principal arrière, sa signalisation et son capteur ultrason. Cette configuration permet de sécuriser le robot en réduisant au maximum le temps de traitement lors du pilotage du moteur. En effet, lors d'un obstacle trop important (par exemple un mur) c'est l'Arduino, avec son capteur ultrason, qui peut couper l'alimentation du moteur.

## Ses bibliothèques

Comme la Raspberry, plusieurs bibliothèques sont ajoutées au programme principal. La version 3 du programme inclut les nouveaux composants du circuit imprimé (capteurs ultrasons, Klaxon). La bibliothèque Wire.h permet la communication I<sup>2</sup>C entre les cartes (équivalent à SMbus pour Raspberry). Elle inclut les fonctions suivantes :

« Wire.available() » permet de tester si une donnée arrive depuis le maître ;

« Wire.read() » pour lire la valeur reçue ;

« Wire.write(ack) » pour envoyer une valeur acquiescement à la demande du maître.

La bibliothèque NewPing.h permet d'utiliser un capteur ultrason dans le programme. Il faut définir une variable « NewPing sonar(8, 9, 200) » avec comme paramètres, le pin Trigger, le pin Echo, et la distance max, ce qui permet d'optimiser les temps de calculs. La fonction « sonar.ping\_cm() » est une valeur entre 0 et la distance max en centimètre.

### Son programme version 3.0

Le programme Raspberry converti un caractère en valeur pour l'envoyer aux cartes Arduino. L'Arduino fait correspondre la valeur à une action. Les valeurs comprises par cet Arduino sont de 50 à 550 avec 0 et 21 :

- De 50 à 299 : règle, en PWM, la vitesse du moteur en sens arrière ;
- De 301 à 550 : règle, en PWM, la vitesse du moteur en sens avant ;
- La valeur 0 et 300 pour l'arrêt du moteur et l'allumage des leds rouges de signalisation
- La valeur 21 pour allumer le Klaxon

Le nombre d'entrée/sorties utilisés sur cet Arduino sont les suivants

Tableau 1 : Tableau des Pins utilisés

Éléments	Nombres d'E/S nécessaires (sans les alimentations)
Moteur	2 numériques PWM
Leds	2 numériques
Capteurs US + Buzzer	2 + 1 numériques
Connexion I2c	2 analogiques
Total	7 numériques + 2 analogiques

### Concernant l'Arduino mini 1

Le second Arduino du drone gère tous les autres modules, à savoir : les servomoteurs de la caméra, la direction du robot, l'éclairage leds avant, le laser et les lance-missiles. Toutes ces actions ne sont pas critiques pour le drone donc elles peuvent être gérées par un Arduino.

### Ses bibliothèques

Les bibliothèques utilisées sont, ici aussi, Wire.h pour la communication I<sup>2</sup>C entre les cartes. Mais aussi, la bibliothèque Servo.h, qui permet de gérer les servomoteurs utilisés pour l'inclinaison de la caméra et pour la direction. Cette bibliothèque permet de créer des variables uniques Servo (« Servo servo1 ») correspondant à un servomoteur. Pour piloter le servomoteur, l'Arduino envoie une valeur angulaire en degrés vers le servomoteur avec la fonction « servo1.write(90) ». Electroniquement, l'Arduino fait varier la tension sur un intervalle de temps à la manière d'un codage PWM, chaque niveau de tension appliqué tous les 20ms correspond à un angle.

### Son programme version 3.1

La version 3 du programme de cet Arduino inclut les nouveaux composants du circuit imprimé comme la nouvelle direction, le laser et le lance-missile, en plus des servomoteurs caméra et de l'éclairage.

Tableau 2 : Tableau des Pins utilisés

Eléments	Nombre d'E/S nécessaires (sans alimentation)
Servomoteurs (Direction, Caméra X, Caméra Y, Lance-missiles)	4 Numériques
Leds Avant	1 numérique
Connection I2c	2 Analogiques
Total	5 numériques + 2 analogiques

## Les différents modules

Cette partie explique quels sont les modules ajoutés aux cartes programmables et la raison de leur installation. Ici, il faut prendre en compte la taille du drone mais aussi son poids de 4.5kg.

### Les moteurs

Le drone est composé, à la base, de deux moteurs indispensables à son fonctionnement : un moteur à l'arrière pour propulser le drone dans deux sens différents, et un moteur pour diriger la voiture. Ces deux moteurs sont à courants continus et change de sens de rotation par inversion de polarité.

### La propulsion

Le moteur qui permet à la voiture d'avancer et de reculer, en variant sa vitesse, est imposant. En effet, il doit entraîner les deux roues arrières et tout le poids du véhicule. C'est un moteur à courant continu à 2 ampères en fonctionnement constant mais avec un pic d'intensité jusqu'à 3 ampères, au démarrage. Il est accompagné d'un circuit électronique qui permet son contrôle par simple changement de polarité, à l'aide de transistors de puissances de types NPN et PNP. Ces transistors forment un composant électronique appelé pont-H. Ce composant permet d'acheminer, ou non, l'électricité d'une source vers le moteur selon la tension appliqué à la base.

Ce circuit, présent à l'origine, est intéressant puisqu'il possède directement les transistors adaptés pour le moteur (adapté à sa tension et à son pic de démarrage). Cependant, il a été fondamental de retirer le microcontrôleur d'origine pour pouvoir le rendre pilotable via Arduino (In 1 et 2 sur le schéma).

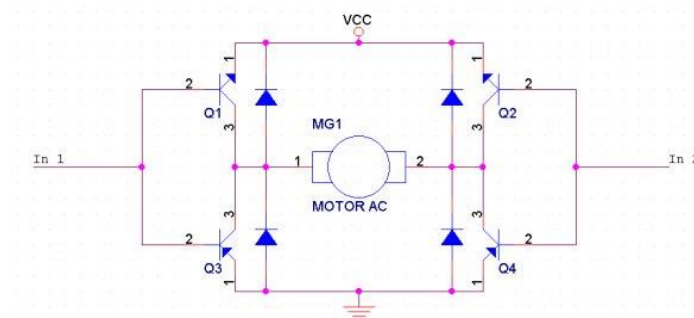


Illustration 4 : Principe du pont-H

Les 4 transistors du pont H ont comme référence SC2654 et SA1129 de la marque NEC (fiche constructeur visible en annexe). Ils peuvent gérer des courants théoriques maximum de 7A. Les tests de courant effectués ont permis d'attribuer au moteur un courant de 2.5A avec un pic, au démarrage, à 3A. L'alimentation de l'Arduino étant trop faible, aussi bien en courant (40mA / pins) qu'en tension (5V), la VCC est une source d'alimentation extérieure.

### La direction

La plupart des directions de jouets radiocommandés est assuré par des servomoteurs. C'est un composant simple d'utilisation, avec un angle de rotation de 0 à 180°, et qui nécessite 2 fils d'alimentations et 1 fil de contrôle. Cependant, ici, la direction est assurée par un moteur et d'un potentiomètre « décomposés ». Ce composant possède donc 6 fils configurés comme suit :

- Fils **noir** et **orange** : Polarité du Moteur
- Fil **Bleu** : Composante antiparasitaire
- Fils **Jaune** (GND) et **vert** (+5V) : alimentation du potentiomètre
- Fil **blanc** : Envoi de données du potentiomètre analogique

La partie motorisée doit avoir un courant supérieur à celui que peut fournir l'Arduino. Il est nécessaire d'utiliser un pont H, comme ci-dessus. Le pont-H choisit est une puce électronique L293D qui supporte un courant maximal de 0.8A environ.

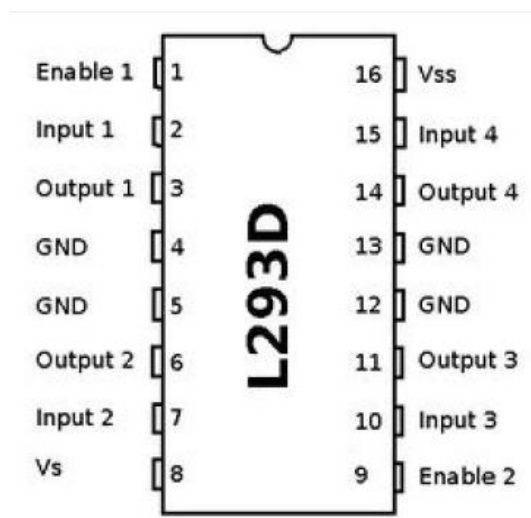


Illustration 5 : Composant L293d

Ce double pont H permet, théoriquement, de connecter 2 moteurs ayant chacun 2 sens de rotation différent. L'utilisation de ce composant pour la direction n'aura besoin de contrôler qu'un moteur donc toutes les broches ne seront pas utilisées.

Tout d'abord, la broche Enable 1 (1) permet d'activer (si relié à VSS) ou désactiver (si relié à GND) le 1<sup>er</sup> pont H, en relation avec OUTPUT 1 (3) et OUTPUT 2 (6). Par ailleurs, on peut faire varier la vitesse d'un moteur en PWM (Pulse With Modulation), cependant je ne compte pas utiliser cette fonction car elle est inutile dans mon cas.

La broche VSS (16) est l'alimentation de la logique de commandes. Elle est connectée à une sortie +5V de l'Arduino. La broche VS (8) correspond à l'alimentation externe qui alimentera le moteur. Les broches INPUT 1 (2) et INPUT 2 (7) sont les broches de commandes du Pont-H des sorties OUTPUT 1/2. Elles seront branchées à deux pins de l'Arduino qui n'auront jamais la même tension, sinon cela endommagerait le circuit. Les broches OUTPUT 1 (3) et OUTPUT 2 (6) sont à reliés au moteur, la tension de sortie maximale est d'environ 35v pour 600mA (1200mA en pointe). Les broches GND correspondent à la masse commune entre VSS et VS.

Cependant, mes compétences en programmation Arduino imposait au moteur de direction d'effectuer une oscillation infinie. Une étude d'un pilotage par asservissement PID (proportionnel intégrale dérivé) puis par proportionnalité simple ont montré une certaine complexité. La solution que j'ai retenue est l'intégration d'un autre modèle de servomoteurs spécialisé pour Arduino.

## Le retour vidéo

Afin de connaître le milieu où se situe le drone, j'ai décidé d'y ajouter une caméra de type Webcam USB C200 de Logitech, d'une résolution de 640 x 480 pixels. Elle permet d'avoir un rendu en temps réel grâce à la carte Raspberry Pi. Cette solution est relativement simple car, la carte possède des ports USB femelles. L'application qui va gérer ce stream vidéo sera « Motion ». Tout d'abord, on installe l'application avec la commande « `sudo apt-get install motion` ». Ensuite, les paramétrages du logiciel se font via la commande « `sudo nano /etc/motion/motion.conf` ». Les paramètres à changer pour mon cas sont les suivants :

- Daemon on (permet un démarrage automatique et en arrière-plan)
- Framerate 40 (définie le nombre d'image/seconde)
- Auto\_brightness on
- Stream\_localhost off (permet de partager le flux sur le réseau)

Après cela, il suffit de se connecter depuis un PC, via un navigateur Web à l'adresse IP du Raspberry et du port 8081 (par défaut). Pour mon cas, on tape la ligne « 192.168.137.2 :8081 » dans un navigateur.

## Les servomoteurs ...

Les servomoteurs sont des modules pour Arduino qui permettent d'effectuer un mouvement de rotation, du plus souvent, de 0 à 180°. Ceux que j'ai choisis dans mon projet sont les servomoteurs MG995 et les micro servomoteurs.

### ... de direction

Comme expliqué précédemment, la direction est, maintenant géré par un servomoteurs Arduino. Son modèle est MG995. Il est caractérisé par un couple d'environ 12kg/cm en 5V à une vitesse de 0.15 secondes /60°. Les pignons sont tous en métal ce qui permet d'appliquer au Servo une certaine force de résistance. Par rapport aux précédents modules de direction, ce servomoteur inclut un moteur, un potentiomètre et ses systèmes de corrections.

### ... de rotation caméra

La caméra utilisée pour le retour vidéo est caractérisé par un champs de vision restreint. Pour augmenter ce dernier, la caméra sera placée sur deux servomoteurs pour couvrir les axes X et Y. Ces servomoteurs, comme ceux utilisés pour la direction, sont de modèle MG995 avec une rotation de 0° à 180°. Ces servomoteurs sont adaptés pour les Arduino, il n'y a donc pas de composants supplémentaires à ajouter.

### ... de lance-missile et son laser

Afin de justifier une application pseudo-militaire et pour ajouter de la fantaisie, des lance-missiles lego NXT ont été ajoutés. Ces modules sont secondaires et n'ont pas d'utilité propre. Un laser rouge fera office de viseur. Ces lance-missiles, à ressort, sont liés à des projectiles en mousse. Ils s'activent à l'aide d'une butée, sur le dessus, relié à un ressort. L'activation à distance passe par la rotation d'un servomoteur qui appuie contre la butée pour lancer le missile. Le servomoteur utilisé ici est un micro-servo 9g.

## Le capteur ultrason

Si un incident survient sur la liaison Wifi ou sur le Raspberry Pi, le drone doit être en mesure d'effectuer un « arrêt d'urgence » face à un obstacle. Prenons l'exemple de la marche avant, si le pilote se retrouve déconnecté du drone, celui-ci doit pouvoir éviter de rentrer dans un mur ou une personne.

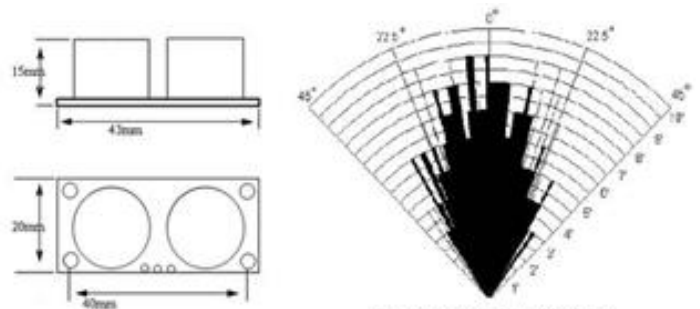
J'ai donc décidé d'ajouter un capteur de distance par ultrasons. Ce type de capteur calcule le temps entre l'envoi et la réception d'ondes ultrasons. En effet, plus l'obstacle est proche, plus les ondes ultrasons reviennent vite vers le capteur.

Les types de capteurs utilisés pour Arduino sont des HC-SR04, le meilleur compromis entre prix et performances. Pour récupérer une distance en centimètre, il faut installer la bibliothèque *NewPing* pour Arduino. Les connecteurs présents étant sensibles, la manipulation des câbles doit se faire avec précaution.

Son champ d'action est restreint entre +/- 15° par rapport à sa perpendiculaire. Même si la détection n'est pas optimale, cela reste une première sécurité.



Illustration 6 : Ultrason HC-SR04



Practical test of performance,  
Best in 30 degree angle

Illustration 7 : Champ d'action du capteur

Un cache pour ce capteur a été fabriqué avec l'aide du département GIM. Pour cela, j'ai modélisé le cache par ordinateur avec le logiciel SolidWorks en prenant une base sur Internet partagée en Open Source. Ensuite, le modèle 3D a été envoyé vers une imprimante 3D. Une pièce comme celle-ci prend environ 30 minutes à être imprimée.

## Le Wifi

Un des objectifs du projet est de pouvoir contrôler un drone aussi bien en local que depuis l'Internet entier, avec la contrainte d'un retour vidéo et en quasi-temps réel. Pour cela, on distingue deux types de maquette.

### Un Raspberry Pi en Wifi

Le mini-ordinateur utilisé dans le drone est un Raspberry Pi en version 2. Il ne possède pas le Wifi par défaut (contrairement à la nouvelle version 3). Un module Wifi doit être ajouté sur les ports USB. Le modèle de cette clé est une Sagem XG-760N. Les caractéristiques principales sont le type de Wifi en 802.11 b/g 2.4GHz. Pour configurer cette clé, il faut d'abord taper la commande « *lsusb* » qui permet de vérifier que le système repère la clé wifi. Ici elle est connectée au device 005.

```
pi@rpiclement2:~ $ lsusb
Bus 001 Device 005: ID 079b:0062 Sagem XG-76NA 802.11bg
Bus 001 Device 006: ID 046d:0802 Logitech, Inc. Webcam C200
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp. LAN9500 Ethernet 10/100 Adapter / SMSC9512/9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Illustration 8 : résultat de la commande *lsusb*

Pour pouvoir configurer et utiliser cette clé, il faut installer les drivers spécifiques à la clé avec la commande « *sudo apt-get install zd1211-firmware* ». Pour la suite, il y a 2 moyens de paramétrer les connexions :

- Manuellement via le fichier */etc/network/interfaces* ou
- Par l'utilitaire « **wicd-curses** »

La deuxième solution a été étudiée au 1<sup>er</sup> semestre mais les résultats n'étaient pas suffisants au niveau de la latence. C'est donc la première solution qui a été retenue. Pour ajouter une configuration Wifi à la carte, on se rend sur le fichier */etc/network/interfaces* en mode root. Avec l'utilitaire nano, on peut ajouter ces lignes :

```
allow-hotplug wlan0
iface wlan0 inet static

address 192.168.137.2
netmask 255.255.255.0
gateway 192.168.137.1
wpa-ssid Drone
wpa-psk *password*
```

### 1<sup>ère</sup> maquette : Le pilotage via l'AP (Point d'accès)

Pour que le projet devienne pilotable depuis l'internet entier donc depuis des endroits géographiques éloignés, il faut suivre quelques étapes. Tout d'abord, il faut, soit posséder une adresse IP publique, soit inscrire son routeur de tête dans une liste de DNS dynamique (ddns). C'est cette deuxième option qui a été utilisée pour un réseau local personnel. J'ai inscrit le routeur sur le site *noip.com* qui permet d'associer mon IP publique dynamique à un nom de domaine statique du type *clementcourtel.ddns.net*. Après cela, des ports privés doivent être ouverts sur le routeur pour pouvoir accéder à l'interface de pilotage.



Le point d'accès choisi ici a été prêté par l'IUT, c'est un modèle Cisco Aironet 1200 series. Pour accéder au mode administrateur il faut renseigner les champs suivants : Login = admin ; Password = Cisco. Il faut commencer par lui attribuer une adresse IP via le port console par la commande

```
> enable
> conf t
> interface bvi1
> ip address 192.168.1.100 255.255.255.0
```

Cela permet de configurer l'équipement par un port Ethernet avec une interface graphique.

### 2<sup>ème</sup> maquette : Un PC comme AP

Pour un fonctionnement en local et/ou pour des tests rapides, il est plus facile de ne pas avoir de machines intermédiaires entre le drone et l'interface de pilotage. Une connexion de type Ad-Hoc a donc été étudiée. Ce type de lien repose sur une interface Wifi et permet d'utiliser un ordinateur portable en tant que point d'accès Wifi où la voiture pourra se connecter. Le site web de

redmondpie.com explique, en anglais, comment configurer une machine Windows 10 en point d'accès Wifi.

Le paramétrage est simple, il suffit de renseigner le réseau IP dans lequel le PC et la voiture devront être, puis, de lancer le point d'accès avec la commande : `netsh wlan start hostednetwork`. L'arrêt du point d'accès passe par la ligne : `netsh wlan stop hostednetwork`.

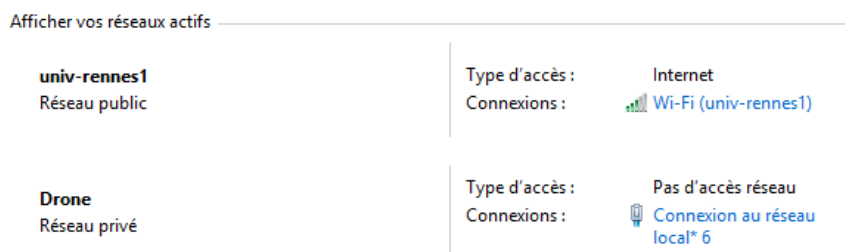


Illustration 10: Démarrage du réseau "Drone"

## L'électronique

Ce projet a aussi demandé des compétences en électronique pour pouvoir réaliser un circuit qui soit le plus optimisé possible, fiable et sécurisé.

### Les composants utilisés

#### La source d'alimentation

Pour être mobile et pouvoir rouler, le drone a besoin d'une batterie importante. Le marché des batteries propose de nombreuses technologies différentes selon les besoins. Il faut comparer les différents modèles.

Technologie de batteries :	Les +	Les -	Retenus ?
Plomb	Prix faible, Rapport puissance-prix important	Contient acides et autres toxines, Poids très important	NON
Lithium ION	Plus légère, puissante, recharge facile	Prix élevé, durée de vie plus faible, encombrant	NON
Li-Po (Lithium Polymère)	Taille réduite, puissante, bonne capacité	Chargeur spécifique, tension de sortie non précise (14,8v ; 11,1v; ...), Prix élevé.	NON
NiMH (accumulateur nickel-hydrure métallique)	Taille réduite, légère, puissante, prix (+chargeur) faible	Détection de fin de charge	OUI

La batterie et son chargeur, pris en charge par l'IUT, sont de type NiMh. C'est une technologie très largement répandue dans le monde du modélisme, fiable et économiquement accessible. Les caractéristiques choisies sont une tension de 12v pour un courant de 4Ah pour un poids de 1 kg. La batterie devra donc alimenter :

- Le moteur de propulsion ;
- Les cartes programmables ;

- La clé Wifi et la Caméra vidéo ;
- Les 4 servomoteurs ;
- Les différentes leds.

### Les optocoupleurs

Tout d'abord, il faut savoir qu'un optocoupleur est un composant capable de transmettre un signal d'un circuit électrique à un autre, sans qu'il y ait de contacts électriques mais photoniques, cela sert à sécuriser le circuit. Le circuit possède 2 optocoupleurs. Ils servent au contrôle des 2 sens du moteur principal par l'Arduino.

Sur ce composant, d'un côté, nous avons une diode électroluminescente qui émet de la lumière infrarouge lorsqu'elle est polarisée. De l'autre côté, nous avons le phototransistor qui est une variante d'un transistor NPN : le courant passe du collecteur vers l'émetteur. Ce transistor s'actionne lorsque la base reçoit de la lumière infrarouge (contrairement à un courant électrique).

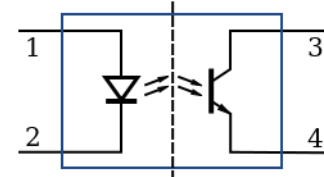


Illustration 11 : Schéma d'un optocoupleur

### Le régulateur de tension

Les différentes cartes et certains modules ne peuvent être alimentés qu'en 5V pour fonctionner correctement. Or la batterie installée possède une différence de potentiel de 12v. Des régulateurs de tensions doivent être mis en place pour abaisser cette tension. Les régulateurs choisis sont des LM2596. Les caractéristiques sont une tension d'entrée à 12V et une tension de sortie réglable à 5v avec un courant de 2A.

A cause d'un courant limité, le circuit électrique comporte 2 régulateurs LM2596. Un qui se charge d'alimenter les Arduino et les modules (Servomoteurs, ultrason, ...) et un qui se charge d'alimenter le Raspberry Pi avec sa clé Wifi et sa caméra.



Illustration 12 : régulateur de tension LM2596

### Le relais

Un relais permet l'ouverture/fermeture d'un circuit électrique par un second circuit complètement isolé. Il faut imposer une tension sur une bobine qui se charge d'ouvrir ou de fermer un autre circuit. Dans ce projet, un relais est utilisé pour piloter l'allumage ou l'extinction de l'éclairage avant. Les 4 Leds hautes-luminosités qui constituent l'éclairage ont une tension globale de 12v, il est donc impossible de le contrôler directement depuis l'Arduino. C'est pour cela qu'un relais permet l'ouverture ou la fermeture du circuit d'éclairage en 12v selon la tension de la bobine en 5v.

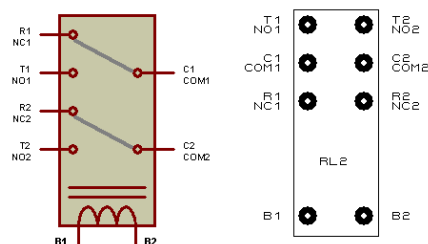


Illustration 13 : Schéma du relais

## Le buzzer

Un buzzer est un composant électronique qui crée un son selon la période de tension appliqué à ses bornes. Pour contrôler la fréquence  $f$  du son, l'Arduino manipule sa période  $T = 1/f$  avec la fonction `delayMicrosecondes`. Pour produire un son, l'Arduino applique au buzzer une tension positive tous les  $T/2$  avant de revenir à 0V. Pour simplifier, la tension appliquée au buzzer peut être représentée par un signal carré d'amplitude  $v = 5$ .

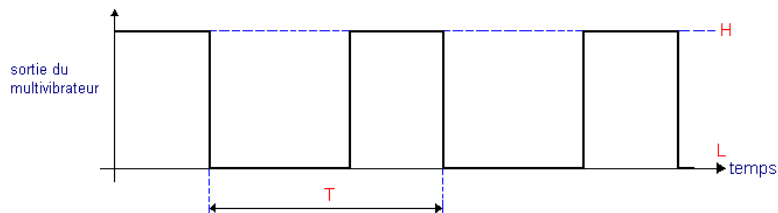


Fig. 43. - Signal en sortie d'un montage astable.

## Les diodes et fusibles

Les diodes ont pour caractéristique d'être polarisées, elles ne sont utiles que pour un sens de courant spécifique. On distingue les diodes non lumineuses (ou de sécurité) et les diodes lumineuses (ou DEL). Le drone utilise ces 2 types de diodes. D'une part pour servir d'éclairage à l'avant ou de signal lumineux à l'arrière. Ces DEL sont polarisées et sont caractérisées par une tension minimal de 1.5V. D'autre part, les diodes de sécurités (ou anti-retour) sont mises en place sur les pins où arrive les tensions. Grâce à cela, lors d'une inversion accidentelle de polarité, ou de parasites, ce sont les fusibles qui vont casser à la place des cartes.

Les fusibles sont des composants qui servent à limiter le courant dans un circuit. Les fusibles sont nécessaires pour protéger des composants, ou cartes, sensible au pic de courant. Il en existe de différentes formes mais ceux du projet sont à usage unique. Cela veut dire que lorsqu'une hausse de courant apparaît dans le circuit (cela peut être le cas avec le démarrage du moteur principal) le fusible se rompt pour couper l'alimentation d'une partie du circuit.

Par mesure de sécurité, j'ai placé trois diodes de protections couplées à trois fusibles. Un couple pour la protection globale du circuit sur le 12V avec un fusible  $T=4A$  (l'indice T veut dire que le fusible peut accepter un faible pourcentage au-dessus de sa limite). Un couple sur chaque régulateur de tension tel que  $F = 1.8A$  pour les Arduino et ses modules et  $F = 1A$  pour le Raspberry Pi (l'indice F veut dire que le fusible est « fast » et qu'il se coupe dès qu'une surintensité intervient).

## Les résistances et condensateurs

La principale caractéristique d'une résistance est d'opposer une résistance à la circulation du courant électrique. Il y en a plusieurs types :

- Les résistances de faibles puissances, qui sont utilisés avec l'alimentation des leds avec une valeur de 300R (ohms) environs et d'une puissance inférieure à 1W ;
- Les résistances de puissance, caractérisés par une puissance de plusieurs dizaines de watts. Une résistance de ce type a été ajoutée sur l'alimentation du moteur pour limiter le courant imposé au moteur. Sa valeur est de 2R pour 50 Watts.

Un condensateur a pour propriété principale de pouvoir stocker des charges électriques opposées sur ses armatures. Le condensateur est caractérisé par le coefficient de proportionnalité entre charge et tension appelé capacité électrique et exprimée en farads (F). La principale utilisation

des condensateurs pour le drone est la stabilisation des alimentations des modules. En effet, il se décharge lors des chutes de tension et se charge lors des pics de tension.

## Le circuit imprimé

Les premiers composants nécessaires aux fonctions élémentaires du drone étaient soudés, sur une carte de prototypage, avec ses fils. Suite à l'évolution du projet et à la multitude de composants, un circuit plus compact et plus abouti devenait obligatoire.

### Le circuit préparé via CAO (conception assistée par ordinateur)

Avant toute réalisation concrète, il est nécessaire de vérifier que les modules et composants fonctionnent sur le circuit de base pour, ensuite, le simuler sur ordinateur. C'est avec le logiciel Proteus qu'il est possible de créer un support PCB. J'ai ensuite eu la chance de rencontrer un enseignant de GEII (Génie électrique Informatique et Industriel) qui s'occupe d'une machine à imprimer des circuits électroniques à l'IUT de Rennes. Il m'a aussi indiqué les nombreuses contraintes de leur machine à prendre en compte :

- Une largeur de piste de, minimum, 25 th (millièmes de pouces) pour des signaux de commandes et de 40th pour 1A ;
- Des vias (ce qui permet de changer de couche) d'un diamètre de 70 th ;
- Les pistes des pins sur couche basse pour pouvoir les souder ultérieurement ;
- Les pattes des composants électroniques sur des supports spécifiques.

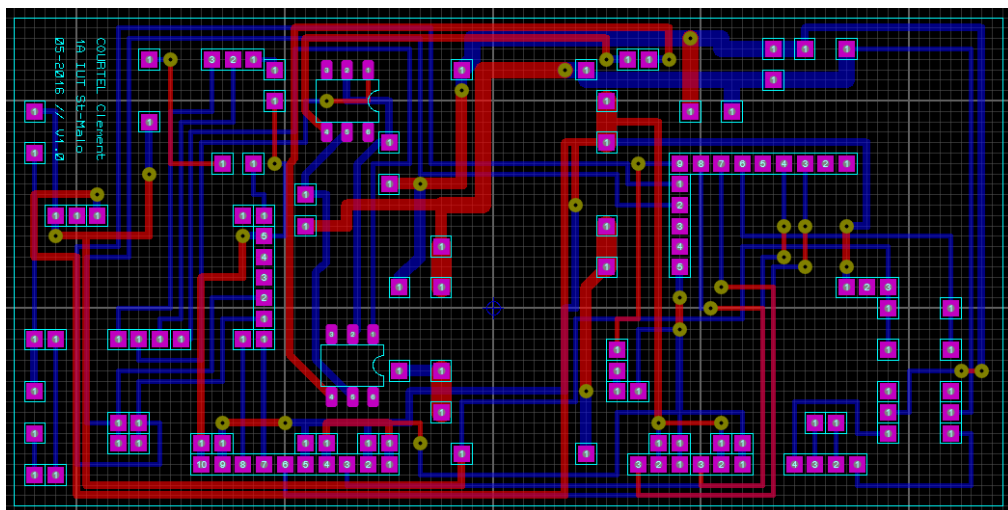


Illustration 14 : Création de la carte sous Proteus

## La méthode de réalisation

Après avoir réalisé la carte avec l'application ARES Proteus, il est possible de l'imprimer. Cela se passe sur un support PCB sensible aux ultraviolets. La première étape est de passer la plaque dans une insoleuse à UV. À Rennes, la carte est mise sous une plaque de verre et sous-vide pour accroître la qualité de traitement. Suite à cette étape d'environ 2 minutes, on passe la plaque dans un bain de liquide corrosif appelé « révélateur ». Le circuit est maintenant visible sur le PCB, cependant pour finir le nettoyage une autre machine sur rouleau est utilisée.

Tous les cercles de pins et de vias avec leurs pistes sont visibles. La carte doit être maintenant percer pour y souder des tiges métalliques dans les vias, et les composants à leur emplacement prévu.

## Les protocoles de communications internes

Les cartes Arduino et Raspberry Pi fonctionnent en coopération, elles doivent donc avoir un protocole de communication identique pour pouvoir s'échanger des données. Le protocole que j'ai choisi est l'I<sup>2</sup>C.

### L'I<sup>2</sup>C

Le protocole I<sup>2</sup>C signifie : Inter-Integrated Circuit en anglais (ou TWI pour Two Wire Interface). Il a été conçu, à la base, par Philips pour des applications domotiques. I<sup>2</sup>C est un bus série synchrone bidirectionnel half-duplex et repose sur un système en bus de type maître-esclave. Son débit, paramétré par défaut est d'environ 100kbits/s. Sa mise en place est simple puisqu'il n'y a que deux fils (et une masse commune) à connecter entre les cartes :

- SDA (Serial Data Line) : ligne de données bidirectionnelle,
- SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle.

Dans le cas du drone, le maître désigné est la Raspberry Pi avec sa bibliothèque Open Source SMBus. C'est elle qui gère les prises de parole et toutes les données passent par ce maître. Les esclaves sont les 2 cartes Arduino. Elles attendent de recevoir une donnée du maître envoyé vers une adresse unique pour chaque esclave. Le point négatif de ce protocole est que l'esclave ne peut pas prendre lui-même la parole, elle doit être initiée par le maître.

Le codage utilisé pour une communication est de type NRZ. Pour lire un bit, le niveau de la ligne SDA doit être maintenu stable pendant le niveau « HIGH » sur la ligne SCL.

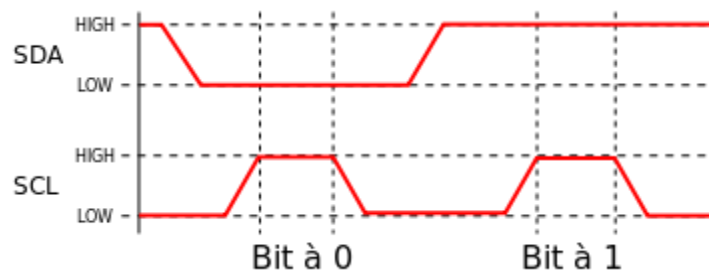


Illustration 15 : Lecture d'un bit par I<sup>2</sup>C

Il existe de nombreux autres protocoles de communication entre les cartes Arduino et Raspberry Pi, comme le protocole SPI. Ce système, contrairement à l'I<sup>2</sup>C, est full Duplex (les 2 cartes peuvent communiquer au même moment) et est caractérisé par des débits plus importants. Cependant, ces deux derniers paramètres ne sont que facultatifs pour une utilisation normale dans le drone et sa mise en place est plus compliquée (4 fils à mettre en commun).

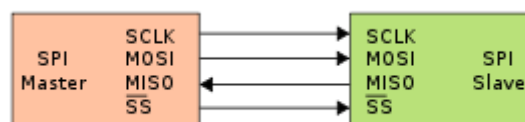


Illustration 16 : Schéma du protocole SPI

### Protocole Série

Les cartes Arduino mis en place au mois de mai sont si petites qu'elles ne possèdent pas de port USB femelle pour se connecter à un PC et pouvoir recevoir un programme. Les fabricants ont privilégié le téléversement de programme depuis une liaison série, il faut donc s'équiper d'un

convertisseur USB <-> Série spécialisé pour Arduino si possible. Mon choix s'est porté sur le FTD1232 générique compatible Arduino. Il possède toutes les broches nécessaires, à savoir :

- Le Rx : la broche pour recevoir des données de l'Arduino
- Le Tx : La broche pour envoyer le programme
- La masse commune et son alimentation
- Le DTR (Data Terminal Ready) : pour initier la communication entre les cartes.

De son côté, l'Arduino possède les broches Rx et Tx (pour communiquer avec le convertisseur) et son alimentation. La broche DTR du convertisseur est relié au pin Reset de l'Arduino. Cependant pour initier la communication, il ne faut pas appliquer un état stable au Reset, il faut ajouter un condensateur de 100nF.

Une liaison série peut aussi être ajoutée entre le Raspberry Pi et Arduino pour pouvoir téléverser un nouveau programme Arduino depuis l'interface du Raspberry. Cette solution n'a pas encore été entièrement étudiée mais peut être mise en place grâce au circuit électronique qui prévoit les pins et les résistances pour cette communication. Par la suite, il suffit de régler les différents débits et paramètres de communications entre les cartes, puis, installer l'IDE Arduino pour Raspberry Pi.

## Conclusion

L'étude de ce projet m'a permis de développer, au fil de l'année, des connaissances approfondies dans plusieurs domaines tel que : l'électronique, la programmation, la robotique, mais aussi dans le relationnel dû à une présence dans des forums et salons. Les événements sont la JSR (journée Simulation Recrutement) et la JPO (journée Portes Ouvertes) à l'IUT de St-Malo, la Maker Faire St-Malo, l'olympiade des sciences de l'ingénieur à Fougères et le salon TEDx à Rennes. Ces différents événements m'ont permis d'apprendre à discuter d'un projet technique en adaptant le discours au niveau du public, mais aussi, de confronter mes idées avec d'autres personnes de différents milieux.

Les applications concrètes d'un projet comme celui-ci peuvent être de différents ordres. Tout d'abord, d'ordre scientifique afin d'épargner à l'homme l'exploration d'environnements dangereux, pollués, toxiques. Il suffirait d'envoyer la machine avec l'ajout de différents capteurs, facilement programmable avec les cartes Arduino ou Raspberry Pi. Mais aussi, d'ordre civil en prenant la fonction de surveillance de maisons. Le wifi est presque omniprésent dans les foyers, il est donc possible d'intégrer un drone sans ajouter de matériel.

Le budget du projet final s'élève à, environ, 240€. Ce montant ne prend pas en compte le matériel rendu inutilisable au cours de l'année, comme certaines cartes de mauvaises qualités. Ce coût pourrait s'élever à 20% de plus. Le budget de 240€ prend en compte, tout d'abord, la batterie et son chargeur, pris en charge par l'IUT de Saint-Malo, d'une valeur de 110€ ; les cartes programmables, d'une valeur totale de 90€ et tous les modules (servomoteurs, composants, ...).

Le projet est, à ce stade, bien abouti. L'objectif de réaliser un Drone Terrestre Wifi est respecté. Il est possible de piloter le drone en Wifi, depuis internet et en local, avec un retour vidéo, Cependant de nombreux autres axes d'améliorations existent, comme celui de l'interface graphique. Il serait intéressant de remplacer l'interface actuelle (ligne de commandes) par une interface de pilotage graphique. Le langage de programmation peut s'orienter vers du Java, du python, du C++ avec Qt mais aussi, il est aussi possible d'utiliser nodejs sur Raspberry Pi.

Un autre axe d'amélioration peut être la mise en place de la liaison série entre la carte Arduino et Raspberry Pi. Chaque modification, ou optimisation de programmes Arduino pourrait se faire directement en Wifi depuis le Raspberry Pi.

D'autre part, un objectif est de rendre ces travaux prochainement publics ; il sera possible de retrouver les étapes de réalisations pour fabriquer son propre Drone à partir d'un jouet télécommandé de base.

## Annexes 1 : Les codes sources (version avril 2016)

### Le code complet de l'Arduino 0

```
/*
  Code arduino N0 pour le Drone Terrestre Wifi / R&T
  Enterprise
  Fonctions de propulsion PWM, signalement lumineux ,
  capteur ultrason, klaxon
  Communication en I2C avec le Rpi (programme 2.0.1)
  @author : Clément Courtel, étudiant IUT St-Malo dpt
  Réseaux et Télécommunications
  @version : 3.0.1b, MAJ du 27/04
*/

#include <Wire.h> // bibliothèque I2C
#include <NewPing.h> // bibliothèque pour capteur
ultrason
#define SLAVE_ADDRESS 0x10 // adresse esclave

NewPing sonar(8, 9, 200); // TRIGERS - ECHO --
Distancemax
char ack; // Variable qui contient la valeur
d'acquiescement, voir rapport
int dR[3], s; // Valeurs reçu de l'I2C
int a; // Valeur s mappée
char i;

void setup() {
  Wire.begin(SLAVE_ADDRESS);
  Wire.onReceive(receiveData);
  Wire.onRequest(sendData);

  pinMode(3, OUTPUT); // marche AR
  pinMode(5, OUTPUT); // Marche AV
  pinMode(2, OUTPUT); // Leds Rouges
  pinMode(4, OUTPUT); // Led Blanche
  pinMode(10, OUTPUT); //Buzzer
}

void loop() {
  // Remise à 0 de l'indice du tableau
  i = 0;
  buzz(10, 0,0);

  // Si la valeur reçu est 0 alors le moteur s'arrête, les leds
  Stop s'alluments
  if (s == 0 || s == 300) {
    digitalWrite(5, 0);
    digitalWrite(3, 0);
    digitalWrite(4, LOW);
    digitalWrite(2, HIGH);
    ack = 10;
  }

  // Si la valeur reçu est comprise entre 301 et 550, il y a
  vérification des distances puis une possible marche avant
  else if ((s >= 301) && (s <= 550)){
    a = map(s, 301, 550, 50, 235); // Limitation de vitesse
    logiciel
    analogWrite(5, a);
    analogWrite(3, 0);
    digitalWrite(4, LOW);
    digitalWrite(2, LOW);
    ack = 11;
  }
  // Si la valeur est comprise entre 50 et 299, il y a marche
  arrière
  else if ((s >= 50) && (s <= 299)) {
    a = map(s, 50, 299, 235, 50); // Limitation de vitesse
    logiciel
    analogWrite(5, 0);
    analogWrite(3, a);
    digitalWrite(4, HIGH);
    digitalWrite(2, LOW);
    ack = 13;
  }
  }

  else if (s==21){
    buzz(10, 2500, 1000); // buzz the buzzer on pin 4 at
    2500Hz for 1000 milliseconds
    s=0;
  }
}

void receiveData(int byteCount) {
  while (Wire.available()) {
    dR[i] = Wire.read();
    i++;
  }
  s = dR[1] + dR[2] * 256;
}

void sendData() {
  Wire.write(ack);
}

void buzz(int targetPin, long frequency, long length) {

  long delayValue = 1000000/frequency/2; // calculate the
  delay value between transitions
  //// 1 second's worth of microseconds, divided by the
  frequency, then split in half since
  //// there are two phases to each cycle

  long numCycles = frequency * length/ 1000; // calculate
  the number of cycles for proper timing

  //// multiply frequency, which is really cycles per second,
  by the number of seconds to
  //// get the total number of cycles to produce
```

```

for (long i=0; i < numCycles; i++){ // for the calculated
length of time...
    digitalWrite(targetPin,HIGH); // write the buzzer pin
high to push out the diaphragm
    delayMicroseconds(delayValue); // wait for the
calculated delay value

```

## Le code complet de l'Arduino 1

```

/*
Code arduino N1 pour le Drone Terrestre Wifi / R&T
Enterprise
Fonctions de direction, rotation caméra, éclairage
avant, lance-missile
Communication en I2C avec le Rpi
@author : Clément Courtel, étudiant IUT St-Malo dpt
Réseaux et Télécommunications
@version : 3.1.2,
MAJ du 30/04 : Direction par servoMoteur
*/

#include <Servo.h> // bibliothèque Servomoteur caméra
#include <Wire.h> // bibliothèque I2C
#define SLAVE_ADDRESS 0x20 // adresse esclave

Servo servo1; // servo1_cameraX
Servo servo2; // servo2_cameraY
Servo servo3; //servo3_missile
Servo servo4; //servo4_Direction

int dR[3], s; //Variables reçu de l'I2C
int a; //Valeur s mappé

boolean f = 0, l=0; // état des feux/laser avants
char i ; //indice tableau
char ack; //valeur s'acquittement

void setup() {
    Serial.begin(9600);
    Wire.begin(SLAVE_ADDRESS);
    Wire.onReceive(receiveData);
    Wire.onRequest(sendData);

    servo1.attach(4); // caméra X
    servo2.attach(3); // caméra Y
    servo4.attach(7); // Direction
    servo3.attach(8); // lance missile

    pinMode(6, OUTPUT); // Lumière avant
    pinMode(9, OUTPUT); // Laser

    servo1.write(90); // initialisation du servo X à 90°
    servo2.write(90); // initialisation du servo Y à 90°
    servo3.write(90); // initialisation lance-missile à 90°
    servo4.write(90);
}

```

```

    digitalWrite(targetPin,LOW); // write the buzzer pin
low to pull back the diaphragm
    delayMicroseconds(delayValue); // wait againf or the
calculated delay value
}

```

```

}

```

```

void loop() {
    // Remise à 0 de l'indice du tableau
    i = 0;

    // Si la valeur est comprise entre 640 et 1360, tranche
direction
    if ((s >= 600) && (s <= 700)){
        a = map(s,600,700,40,140);
        servo4.write(a);
        ack = 14;
        s=0;
    }

    else if ((s >= 1100) && (s <= 1280)) { // Caméra_X
        a = map(s,1100,1280,0,180);
        servo1.write(a); // positionnement = pin 4
        ack = 16;
        s=0;
    }

    else if ((s >= 1300) && (s <= 1480)) { // Caméra_Y
        a = map(s, 1300, 1480, 0, 180); //redéfinie dR entre 0 et
180
        servo2.write(a); // positionnement = pin 5
        ack = 18;
        s = 0;
    }

    else if (s == 20){ // éclairage
        f = !f;
        digitalWrite(6, f);
        if (f == 0){ack = 26;}
        else {ack = 25;}
        s = 0;
    }

    else if (s==22){ // Missile 1
        servo3.write(0);
        ack = 20;
    }

    else if (s ==23){ // Missile 2
        servo3.write(180);
        ack = 21;
    }

    else if (s==24){ //Laser
        l = !l;
        digitalWrite(9,l);
        if (l == 0){ack = 24;}
        else {ack = 23;}
        s=0;
    }
}

```

```

}

void receiveData(int byteCount) {
  while (Wire.available()) {
    dR[i] = Wire.read();
    i++;
  }
}

```

## Le code complet du Raspberry Pi

```
# -*- coding: utf-8 -*-
```

```

#Fichier stocké sur Raspberry Pi, Interface le Drone
Terrestre Wifi / R&T Enterprise
#Communication en I²C avec les esclaves arduinos, en
correspondance avec la version 3.0 et 3.1
#@author : Clément Courtel, étudiant IUT St-Malo dpt
Réseaux et Télécommunication
#@version 2.0.1, MAJ du 27/04

```

```

import smbus
import time
i=1

```

```

# Remplacer 0 par 1 si nouveau Raspberry
bus = smbus.SMBus(1)

```

```

#Plan d'adressage des cartes esclaves
address1 = 0x10 #arduino0Propulsion
address2 = 0x20 #arduino1Direction

```

```

#Initialisation des valeurs
m=300
d= 800
klaxon = 21
eclairage = 20
servo_X = 1190
servo_Y = 1390
precision_camera = 4
precision_direction = 5
precision_pwm = 5
delais = 0.1while i==1:
    b = raw_input("Entrez la commande (o : STOP)
:) # caractère

```

```
    #Affichage sur l'interface
```

```

    if (b == 'z') : #Avancer
        m = m + precision_pwm

```

```

        bus.write_word_data(address1,0x00,m)
        print "11 = Avancer : ", m
        time.sleep(delais)
        reponse = bus.read_byte(address1)
        print "La reponse de l'arduino : ",

```

```
reponse
```

```

    elif (b == 's') : #Reculer ralentir
        m = m - precision_pwm

```

```

        bus.write_word_data(address1,0x00,m)
        print "13 = Reculer : ", m
        time.sleep(delais)
        reponse = bus.read_byte(address1)

```

```

        s = dR[1] + dR[2] * 256;
    }

```

```

void sendData() {
  Wire.write(ack);
}

```

```

        print "La reponse de l'arduino : ",
reponse

```

```

    elif (b == 'q') : #Virage Gauche
        d = d + precision_direction

```

```

        bus.write_word_data(address2,0x00,d)
        print "Gauche : ", d
        time.sleep(delais)
        reponse = bus.read_byte(address2)
        print "La reponse de l'arduino : ",

```

```
reponse
```

```

    elif (b == 'd') : #Virage Droite
        d = d - precision_direction

```

```

        bus.write_word_data(address2,0x00,d)
        print "Droite : ", d
        time.sleep(delais)
        reponse = bus.read_byte(address2)
        print "La reponse de l'arduino : ",

```

```
reponse
```

```

    elif (b == 'o') : #Arret d'urgence Moteur
        m = 300

```

```

        bus.write_word_data(address1,0x00,m)
        print "10 = Arret moteur "
        time.sleep(delais)
        reponse = bus.read_byte(address1)
        print "La reponse de l'arduino : ",

```

```
reponse
```

```

    elif (b == 'h') : #Rotation Caméra X +
        servo_X = servo_X + precision_camera

```

```

        bus.write_word_data(address2,0x00,servo_X)
        print "16 = Rotation_X : ", servo_X
        time.sleep(delais)
        reponse = bus.read_byte(address2)
        print "La reponse de l'arduino : ",

```

```
reponse
```

```

    elif (b == 'k') : #Rotation Caméra X -
        servo_X = servo_X - precision_camera

```

```

        bus.write_word_data(address2,0x00,servo_X)
        print "16 = Rotation_X : ", servo_X
        time.sleep(delais)
        reponse = bus.read_byte(address2)
        print "La reponse de l'arduino : ",

```

```
reponse
```

```

    elif (b == 'u') : #Rotation Caméra Y +

```

```

servo_Y = servo_Y + precision_camera

bus.write_word_data(address2,0x00,servo_Y)
print "18 = Rotation_Y : ", servo_Y
time.sleep(delais)
reponse = bus.read_byte(address2)
print "La reponse de l'arduino : ",
reponse

elif (b == 'j') : #Rotation Caméra Y -
servo_Y = servo_Y - precision_camera

bus.write_word_data(address2,0x00,servo_Y)
print "18 = Rotation_Y : ", servo_Y
time.sleep(delais)
reponse = bus.read_byte(address2)
print "La reponse de l'arduino : ",
reponse

elif (b == 'w') : #Lance Missile 1

bus.write_word_data(address2,0x00,22)
print "20 = Lancement Missile 1 "
time.sleep(delais)
reponse = bus.read_byte(address2)
print "La reponse de l'arduino : ",
reponse

elif (b == 'x') : #Lance Missile 2

bus.write_word_data(address2,0x00,23)
print "21 = Lancement Missile 2 "
time.sleep(delais)

```

```

reponse = bus.read_byte(address2)
print "La reponse de l'arduino : ",
reponse

elif (b == 'm') : #Klaxon

bus.write_word_data(address1,0x00,21)
print "Klaxon"
time.sleep(delais)
reponse = bus.read_byte(address1)
print "La reponse de l'arduino : ",
reponse

elif (b == 'l') : #éclairage

bus.write_word_data(address2,0x00,20)
print "25 = Allumage éclairage"
print "26 = Extinction éclairage"
time.sleep(delais)
reponse = bus.read_byte(address2)
print "La reponse de l'arduino : ",
reponse

elif (b == 'c') : #laser

bus.write_word_data(address2,0x00,24)
print "24 = Allumage Laser"
time.sleep(delais)
reponse = bus.read_byte(address2)
print "Reponse de l'arduino : ",
reponse

```

## Annexes 2 : Divers

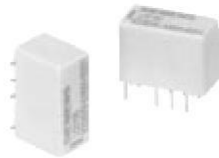
### Les fiches composants

Le relais :

#### Siemens Electromechanical Components

Siemens® Relays

Through Hole or Surface Mount



### V23079 series

**2 Amp, High Dielectric**  
**2 Pole Polarized**  
**FCC Part 68**  
**PC Board Relay**

File E48393

File LR45064

CECC 16 100/16 200/16 500

#### Features

- Surface and through hole mounting types.
- Breakdown voltage between contacts and coil: 1,800V.
- Surge withstand between contacts and coil: 2,500V (Bellcore).
- High capacity contact: 2A @30VDC.
- 2 Form C contact arrangement.
- Board space saving, vertical mount (14.6 x 7.2mm surface area).
- Immersion cleanable, plastic sealed case.
- Single and dual coil latching versions available.

#### Contact Data

**Arrangement:** 2 Form C (DPDT).

**Material: B201:Stationary Contacts:** Gold overlay on silver palladium.

**Movable Contacts:** Palladium silver.

**B301:Stationary and Movable Contacts:**  
 Gold overlay on silver nickel.

#### Rating:

**Max. Switching Voltage:** 250VAC, 220VDC.

**Max. Switching Current:** 2A.

**Max Carrying Current:** 2A.

**Max Switching Power:** 60W, DC, resistive.

62.5VA, AC, resistive.

**Min. Permissible Load:** 500μV.

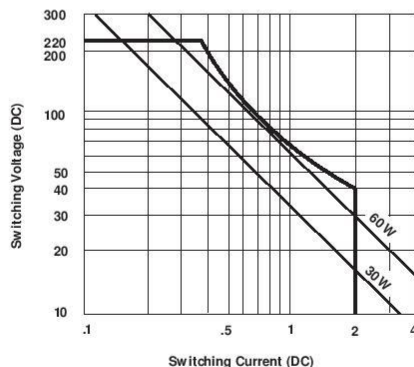
**Expected Mechanical Life:** Approx. 100 million ops.

**Expected Electrical Life:** 500,000 ops. @ 1A, 30VDC.

10 million ops. @ 100mA, 6VDC.

**Initial Contact Resistance:** 50 milliohms @ 10mA, 20mV.

Figure 1 - Limiting Curve for Contact Loads



#### Initial Dielectric Strength

**Between Open Contacts:** 1,000V rms for 1 min.

**Between Adjacent Contact Terminals:** 1,800V rms for 1 min.

**Between Contact and Coil:** 1,800V rms for 1 min.

**Surge Voltage:**

**Between Contact and Coil (10 x 160 μs):** 1,500V (FCC Part 68).

**Between Contact and Coil (2 x 10 μs):** 2,500V (Bellcore).

#### Initial Insulation Resistance

**Between Mutually Insulated Conductors:** 10<sup>9</sup> ohms @ 500VDC.

#### Coil Data @20°C

**Voltage:** 3-48V.

**Nominal Power:**

**Non-Latching:** 140mW.

**Single Coil Latching:** 70mW.

**Dual Coil Latching:** 140mW.

Nominal Voltage (VDC)	Operating Range @20°C		Coil Resistance @20°C
	Must Operate Voltage (VDC)	Max. Voltage (VDC)	
Non-Latching, 140mW Nominal Power			
3	2.25	6.5	64 ± 6
4.5	3.375	9.8	145 ± 15
5	3.75	10.9	178 ± 18
6	4.50	13.0	257 ± 26
9	6.75	19.6	578 ± 58
12	9.0	26.1	1,029 ± 103
24	18.0	52.3	4,114 ± 411
48	36.0	101.0	15,362 ± 1,536
Single Coil Latching, 70mW Nominal Power			
3	2.25	9.2	128 ± 13
4.5	3.375	13.8	289 ± 29
5	3.75	15.3	357 ± 36
6	4.5	18.5	514 ± 51
9	6.75	27.7	1,157 ± 116
12	9.0	37.0	2,057 ± 206
24	18.0	74.0	8,228 ± 823
Dual Coil Latching, 140mW Nominal Power			
3	2.25	6.5	64 ± 6
4.5	3.375	9.8	145 ± 15
5	3.75	10.9	178 ± 18
6	4.5	13.0	257 ± 26
9	6.75	19.6	578 ± 58
12	9.0	26.1	1,029 ± 103
24	18.0	52.3	4,114 ± 411

#### Operate Data @20°C

**Must Operate Voltage:** 75% of nominal or less.

**Must Release Voltage:** 10% of nominal or more.

**Operate Time (Excluding Bounce):** 3ms, typical.

**Release Time (Excluding Bounce):** 3ms, typical.

**Bounce Time:** 2ms, typical.

#### Environmental Data

**Temperature Range:** -40 to +85°C

**Vibration, Operational:** 35g, 10-1,000 Hz.

**Shock, Functional:** 50g, 11ms 1/2 sinusoidal impulse.

**Destructive:** 150g, 11ms 1/2 sinusoidal impulse.

#### Mechanical Data

**Termination:** Through hole or surface mount printed circuit terminals.

**Enclosure:** Immersion cleanable sealed plastic case.

**Weight:** 2.5g approximately.

Le pont H, L293d :



## L293, L293D

www.ti.com

SLRS008D – SEPTEMBER 1986 – REVISED JANUARY 2016

### 6.5 Electrical Characteristics

over operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
$V_{OH}$ High-level output voltage	L293: $I_{OH} = -1$ A L293D: $I_{OH} = -0.6$ A	$V_{CC2} - 1.8$	$V_{CC2} - 1.4$		V
$V_{OL}$ Low-level output voltage	L293: $I_{OL} = 1$ A L293D: $I_{OL} = 0.6$ A		1.2	1.8	V
$V_{OKH}$ High-level output clamp voltage	L293D: $I_{OK} = -0.6$ A		$V_{CC2} + 1.3$		V
$V_{OKL}$ Low-level output clamp voltage	L293D: $I_{OK} = 0.6$ A		1.3		V
$I_{IH}$ High-level input current	A EN	$V_I = 7$ V	0.2	100	$\mu$ A
$I_{IL}$ Low-level input current	A EN	$V_I = 0$	-3	-10	$\mu$ A
$I_{CC1}$ Logic supply current	$I_O = 0$	All outputs at high level All outputs at low level All outputs at high impedance	13 35 8	22 60 24	mA
$I_{CC2}$ Output supply current	$I_O = 0$	All outputs at high level All outputs at low level All outputs at high impedance	14 2 2	24 6 4	mA

### 6.6 Switching Characteristics

over operating free-air temperature range (unless otherwise noted)  $V_{CC1} = 5$  V,  $V_{CC2} = 24$  V,  $T_A = 25^\circ\text{C}$

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{PLH}$ Propagation delay time, low-to-high-level output from A input	L293NE, L293DNE L293DWP, L293N L293DN		800		ns
$t_{PHL}$ Propagation delay time, high-to-low-level output from A input	L293NE, L293DNE L293DWP, L293N L293DN		400		ns
$t_{TLH}$ Transition time, low-to-high-level output	L293NE, L293DNE L293DWP, L293N L293DN		200		ns
$t_{THL}$ Transition time, high-to-low-level output	L293NE, L293DNE L293DWP, L293N L293DN		300		ns

### 6.7 Typical Characteristics

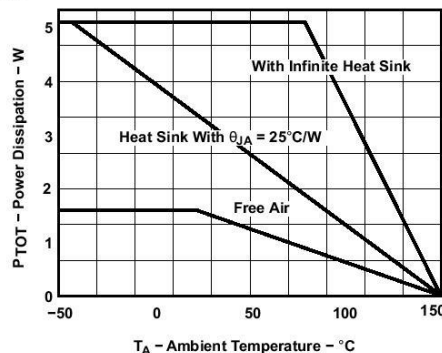


Figure 1. Maximum Power Dissipation vs Ambient Temperature

**L293, L293D**

SLRS008D – SEPTEMBER 1986 – REVISED JANUARY 2016

www.ti.com

## 6 Specifications

### 6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)<sup>(1)</sup>

	MIN	MAX	UNIT
Supply voltage, $V_{CC1}$ <sup>(2)</sup>		36	V
Output supply voltage, $V_{CC2}$		36	V
Input voltage, $V_I$		7	V
Output voltage, $V_O$	–3	$V_{CC2} + 3$	V
Peak output current, $I_O$ (nonrepetitive, $t \leq 5$ ms): L293	–2	2	A
Peak output current, $I_O$ (nonrepetitive, $t \leq 100$ $\mu$ s): L293D	–1.2	1.2	A
Continuous output current, $I_O$ : L293	–1	1	A
Continuous output current, $I_O$ : L293D	–600	600	mA
Maximum junction temperature, $T_J$		150	°C
Storage temperature, $T_{stg}$	–65	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

(2) All voltage values are with respect to the network ground terminal.

### 6.2 ESD Ratings

	VALUE	UNIT
$V_{(ESD)}$ Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001 <sup>(1)</sup>	±2000
	Charged-device model (CDM), per JEDEC specification JESD22-C101 <sup>(2)</sup>	±1000

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

### 6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

	MIN	NOM	MAX	UNIT
Supply voltage	$V_{CC1}$	4.5	7	V
	$V_{CC2}$	$V_{CC1}$	36	V
$V_{IH}$ High-level input voltage	$V_{CC1} \leq 7$ V	2.3	$V_{CC1}$	V
	$V_{CC1} \geq 7$ V	2.3	7	V
$V_{IL}$ Low-level output voltage	–0.3 <sup>(1)</sup>		1.5	V
$T_A$ Operating free-air temperature	0		70	°C

(1) The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

### 6.4 Thermal Information

THERMAL METRIC <sup>(1)</sup>		L293, L293D	UNIT
		NE (PDIP)	
		16 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance <sup>(2)</sup>	36.4	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	22.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	16.5	°C/W
$\Psi_{JT}$	Junction-to-top characterization parameter	7.1	°C/W
$\Psi_{JB}$	Junction-to-board characterization parameter	16.3	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, SPRA953.

(2) The package thermal impedance is calculated in accordance with JESD 51-7.



## LM2596 SIMPLE SWITCHER® Power Converter 150 kHz 3A Step-Down Voltage Regulator

Check for Samples: [LM2596](#)

### FEATURES

- 3.3V, 5V, 12V, and Adjustable Output Versions
- Adjustable Version Output Voltage Range, 1.2V to 37V  $\pm 4\%$  Max Over Line and Load Conditions
- Available in TO-220 and TO-263 Packages
- Ensured 3A Output Load Current
- Input Voltage Range Up to 40V
- Requires Only 4 External Components
- Excellent Line and Load Regulation Specifications
- 150 kHz Fixed Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode,  $I_Q$  Typically 80  $\mu A$
- High Efficiency
- Uses Readily Available Standard Inductors
- Thermal Shutdown and Current Limit Protection

### APPLICATIONS

- Simple High-Efficiency Step-Down (Buck) Regulator
- On-Card Switching Regulators
- Positive to Negative Converter

### DESCRIPTION

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3V, 5V, 12V, and an adjustable output version.

Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation, and a fixed-frequency oscillator.

The LM2596 series operates at a switching frequency of 150 kHz thus allowing smaller sized filter components than what would be needed with lower frequency switching regulators. Available in a standard 5-lead TO-220 package with several different lead bend options, and a 5-lead TO-263 surface mount package.

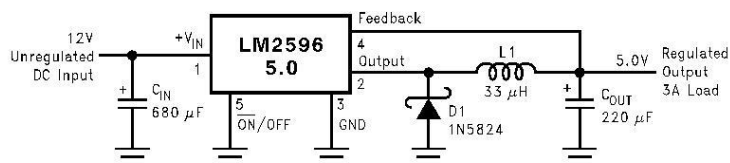
A standard series of inductors are available from several different manufacturers optimized for use with the LM2596 series. This feature greatly simplifies the design of switch-mode power supplies.

Other features include a ensured  $\pm 4\%$  tolerance on output voltage under specified input voltage and output load conditions, and  $\pm 15\%$  on the oscillator frequency. External shutdown is included, featuring typically 80  $\mu A$  standby current. Self protection features include a two stage frequency reducing current limit for the output switch and an over temperature shutdown for complete protection under fault conditions. <sup>(1)</sup>

(1) † Patent Number 5,382,918.

### Typical Application

(Fixed Output Voltage Versions)



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet. SIMPLE SWITCHER is a registered trademark of Texas Instruments. All other trademarks are the property of their respective owners.

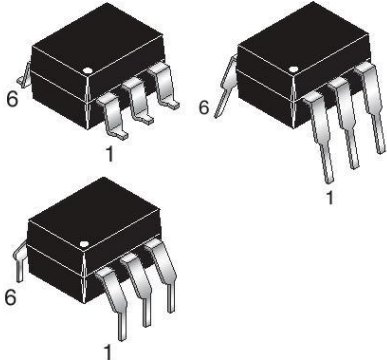
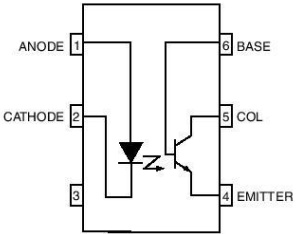
PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 1999–2013, Texas Instruments Incorporated

L'optocoupleur :



## 6-PIN PHOTOTRANSISTOR OPTOCOUPLEDERS

SL5500	SL5501	SL5504	SL5511
<b>PACKAGE</b>			
			
<b>SCHEMATIC</b>			
			

### DESCRIPTION

The SL5500, SL5501, SL5504 and SL5511 are optically coupled isolators each consisting of an infrared emitting GaAs diode and a silicon NPN phototransistor with accessible base. These devices are housed in 6-pin dual-in-line packages (DIP).

### FEATURES

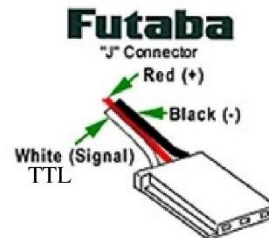
- High output/input DC current transfer ratio
- Low saturation voltage
- High isolation voltage of 5.3 kV RMS
- UL recognized (File # E90700)
- VDE recognized (File # 94766)
- Ordering option '300' (e.g. SL5500.300)

### APPLICATIONS

- Power supply regulators
- Digital logic inputs
- Microprocessor inputs
- Appliance sensor systems
- Industrial controls

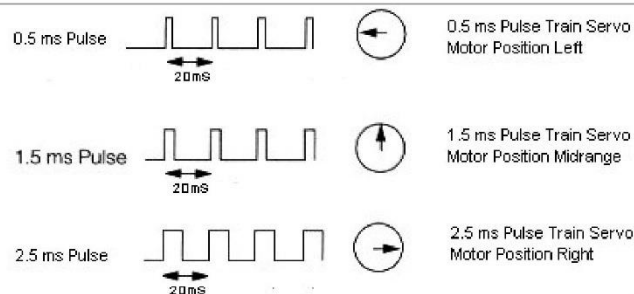
## SERVO-MOTEUR FUTABA S3003

Dimensions: 40,4 x 19,8 x 36,0 mm  
 Poids: 37 g  
 Couple: 4,1 Kg.cm  
 Vitesse: 0,19s / 60°  
 Alimentation: 4,8 - 6 V  
 Roulements: Néant  
 Engrenage: Plastique

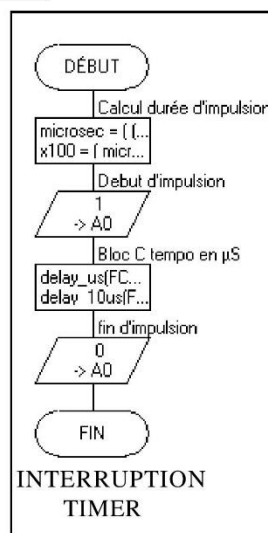
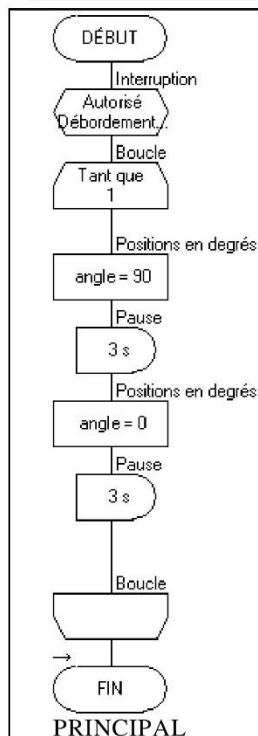


**Pour commander la position angulaire il faut envoyer sur le fil de commande des impulsions de largeur variable (entre 0.5 et 2.5 ms) toutes les 20 ms (50 HZ). En l'absence d'impulsions, le servo reste en position mais pour bénéficier du couple maxi il faut envoyer les impulsions en continu.**

**le signal de commande est de niveau TTL**



Pulse Width Duty / ms	Angle / degrees
0.5	0
1.0	45
1.5	90 (center)
2.0	135
2.5	180



Nom de variable	Type de variable
angle	OCTET
x10	OCTET
microsec	ENTIER
x100	OCTET
x1	OCTET

### Exemple avec FLOWCODE

PIC 16F84 à 3,2769 MHZ

On utilise l'interruption TIMER que l'on programme pour une exécution 50 fois par seconde.

Dans la macro Interruption TIMER on calcule la durée de l'impulsion en µS en fonction de l'angle (11,1µS/°)  

$$\text{microsec} = ((\text{angle} * 111) / 10) + 500$$
 La plus petite tempo possible avec FLOWCODE étant de 1ms on utilise un bloc C pour des tempo en µS la valeur attendue étant sur un octet un pavé calcul détermine les centaines, dizaines et unité de µS

$$\begin{aligned}
 x100 &= (\text{microsec} / 100) \\
 x10 &= (\text{microsec} - (x100 * 100)) / 10 \\
 x1 &= (\text{microsec} - (x100 * 100)) - (x10 * 10) / 10
 \end{aligned}$$

**Silicon NPN Power Transistors****2SC2654****DESCRIPTION**

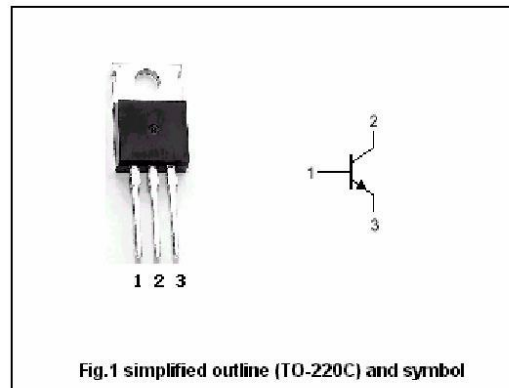
- With TO-220 package
- Complement to type 2SA1129
- Low collector saturation voltage

**APPLICATIONS**

- For low-frequency power amplifiers and mid-speed switching applications

**PINNING**

PIN	DESCRIPTION
1	Base
2	Collector; connected to mounting base
3	Emitter

**Absolute maximum ratings ( $T_a=25^\circ\text{C}$ )**

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$V_{CBO}$	Collector-base voltage	Open emitter	100	V
$V_{CEO}$	Collector-emitter voltage	Open base	40	V
$V_{EBO}$	Emitter-base voltage	Open collector	7	V
$I_C$	Collector current (DC)		7	A
$I_{CM}$	Collector current-peak		15	A
$I_B$	Base current (DC)		3.5	A
$P_T$	Total power dissipation	$T_C=25$	40	W
		$T_a=25$	1.5	
$T_j$	Junction temperature		150	
$T_{stg}$	Storage temperature		-55~150	