Table des matières

[Introduction 2](#_Toc446256476)

[Périphérique USB 3](#_Toc446256477)

[Raspberry pi zero 3](#_Toc446256478)

[Arduino uno 3](#_Toc446256479)

[Test de la méthode unojoy 4](#_Toc446256480)

[LUFA 6](#_Toc446256481)

[Création de mon firmware 6](#_Toc446256482)

[Maquette cockpit hélicoptère 7](#_Toc446256483)

[Fonctionnement de la maquette 7](#_Toc446256484)

[Programme pour les quatre potentiomètres 8](#_Toc446256485)

[Reglage de Xplane 8](#_Toc446256486)

[Boitier arduino 9](#_Toc446256487)

[Prise capteur 9](#_Toc446256488)

[DB9 Potentiomètre 9](#_Toc446256489)

[DB9 Memsic2125 9](#_Toc446256490)

[DB9 ADXL345 9](#_Toc446256491)

[Boite 9](#_Toc446256492)

[Capteurs 10](#_Toc446256493)

[Memsic2125 10](#_Toc446256494)

[ADXL345 10](#_Toc446256495)

# Introduction

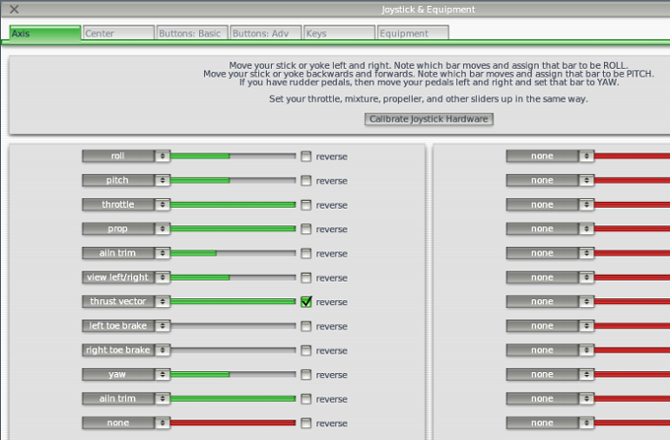
Comme il est précisé dans le cahier des charges, il faut que je produise un moyen de récupérer les mouvements des commandes de vol sur la cellule de l’alouette III fournie par le client.

Il y a en tout 3 commandes de vol :

* Le pas collectif, il permet de varier l’inclinaison des pales pour faire monter ou descendre l'hélicoptère ;
* Le pas cyclique, pour faire varier l’inclinaison du rotor principal ;
* Le palonnier, qui lui permet de varier l’inclinaison des pales du rotor arrière.

En tout le déplacement de ces trois commandes produit un déplacement sur 4 axes.

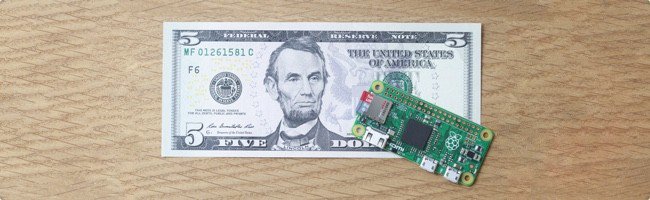
Il me reste à trouver un moyen de simuler ces quatre axes sur le logiciel de simulation x-plane.



# Périphérique USB

Après quelques recherches sur internet, j’ai trouvé deux solutions viables pour la récupération des informations de commandes de vol et l’envoie sur le simulateur en USB.

## Raspberry pi zero



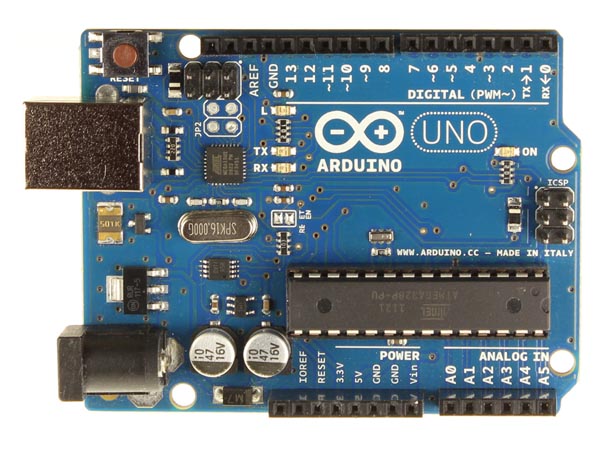
## Arduino uno

L’arduino est une simple carte électronique de petit format au prix de 20€ supportant un microcontrôleur entouré du minimum de composants nécessaires à son fonctionnement de base.

Cette carte supporte également, sur sa périphérie, une rangée de connecteurs dans lesquels peuvent venir s’afficher une ribambelle de capteurs (tout juste ce qu’il nous faut).

Mais ce n’est pas tout, elle est également équipée d’un connecteur USB permettant de la connecter à un ordinateur pour écrire le programme destiné à piloter l’arduino, mais également pour programmer le microcontrôleur qui l’équipe sans aucune intervention matérielle en extérieur.

C’est sur ce dernier point que l’arduino nous intéresse fortement. Sur internet, on peut trouver ces petites plaques transformées en joystick USB unojoy.



## Test de la méthode unojoy

Ce projet permet de faire que l’arduino soit considéré comme un joystick USB par un ordinateur.

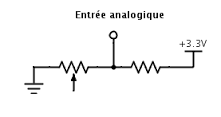
Pour cela, il faut télécharger le fichier unojoy.hex puis le placer dans le microcontrôleur.

La manipulation est décrite dans l’annexe.

J’ai à ma disposition une première maquette d’un axe de commande (photo ci-dessous), son fonctionnement est très simple. Lorsque l’on tire sur le manche une corde fait varier la valeur d’un potentiomètre.



Il faut donc que je récupère les valeurs du potentiomètre pour ensuite simuler le déplacement d’un d’axe de joystick.

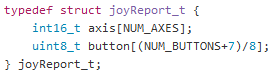
Dans un premier temps, j’ai fait un petit programme de lecture des valeurs du potentiomètre, rien de bien compliqué avec l’IDE de arduino. Une simple lecture du port analogique sur laquelle est branchée la sortie du potentiomètre suffit.

Maintenant que la lecture des valeurs analogiques est bien maîtrisée, il me faut réussir à faire un programme compatible avec l’envoi d’un rapport sur la sortie USB.

Pour cela, je m’inspire du code fourni avec le unojoy.hex.

Je vois que l’envoi du rapport des valeurs des boutons et du déplacement des axes est géré par la fonction sendJoyReport () (elle envoie le rapport HID sur l’interface USB).

La forme du rapport est définie par la structure joyReport\_t, dans le code déjà fourni, elle ressemble à ceci :



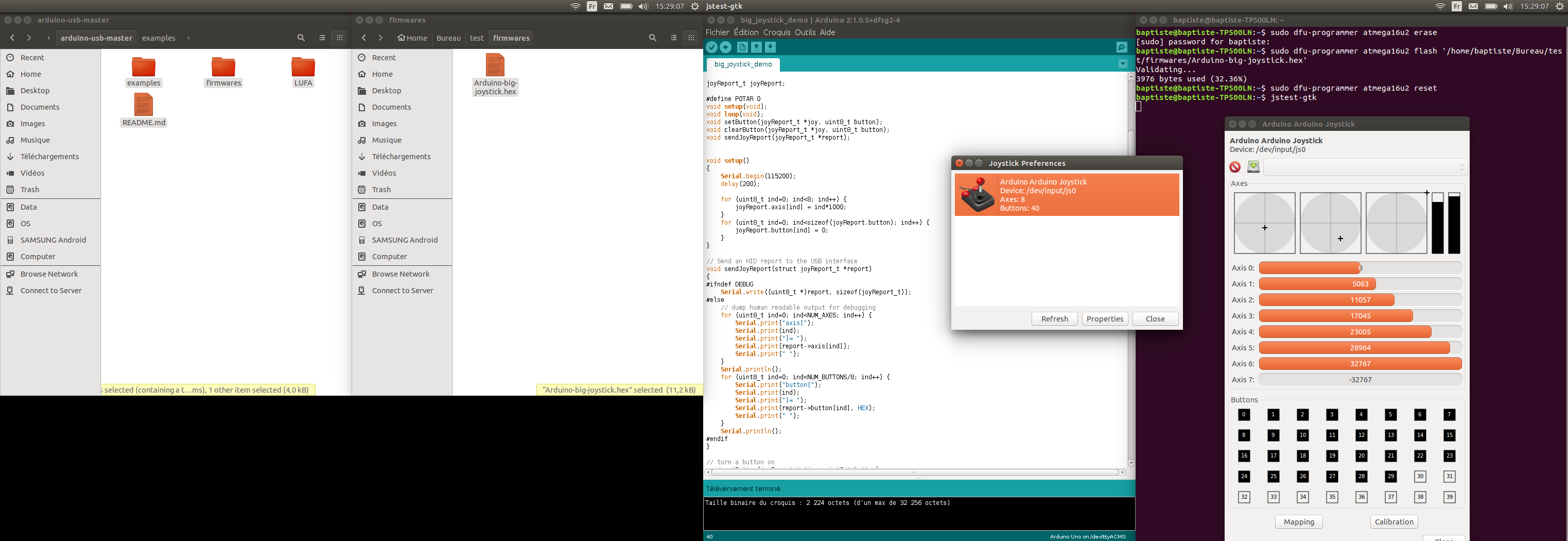
On voit qu’elle contient les valeurs des axes sur 16 bits et les boutons sur 8 bits.

Pour le moment, je n’ai qu’un axe à transmettre mais je ne peux pas modifier ce rapport car il est aussi défini dans le .hex et je ne sais pas encore le modifier, je me pencherai sur ce problème plus tard.

*J’ai maintenant le contrôle de cette axe grâce au potentiomètre*

Je place donc simplement la valeur de mon axe dans le rapport ainsi :

JoyReport.axis[0] = analogRead(0) ;



Je peux voir que mon travail est correct par le lien d’un logiciel de simulateur de joystick et je peux aussi voir le déplacement de l’axe attaché à mon potentiomètre. Je remarque que le contrôle a bien lieu mais un problème auquel je n’avais pas pensé apparait.

Je ne me déplace pas sur toute la plage de valeur de l’axe, ce problème et lié au fait que mon potentiomètre ne se déplace que sur une plage de valeur entre 0 et 1023 (seulement dans un cas idéal en vérité c’est plutôt entre 200 et 1000). Il me faut donc créer un système de calibrage des butées maximum et minimum.

Pour cela j’utilise une fonction qui est dans la librairie de base de l’arduino map().

Cette fonction effectue simplement un produit en croix entre mes valeurs minimum et maximum de mon potentiomètre et celles de mon axe. J’obtiens donc cela :

joyReport.axis[0] =map(analogRead(0), valeur\_Min, valeur\_Max, -32768,32767 );

Le problème est résolu.

Je peux donc confirmer le choix de cette solution, qui consiste à utiliser un arduino comme périphérique USB.

# LUFA

Maintenant que j’ai la confirmation que la méthode unojoy fonctionne, je me penche plus en détail sur son fonctionnement.

Après quelques recherches sur internet je trouve comment Le fichier. hex que je charge dans l’arduino, est créé à l’aide de LUFA.



LUFA est une bibliothèque USB open-source l’USB-enabled AVR des microcontrôleurs.

Sur leur site il y a une liste de projets déjà réalisés, on peut y trouver justement unojoy. Mais cette fois le code permettant de réaliser le fichier .hex s’y trouve et est disponible sur [https://github.com/harlequin-tech/arduino-usb](https://github.com/harlequin-tech/arduino-usb%20) .

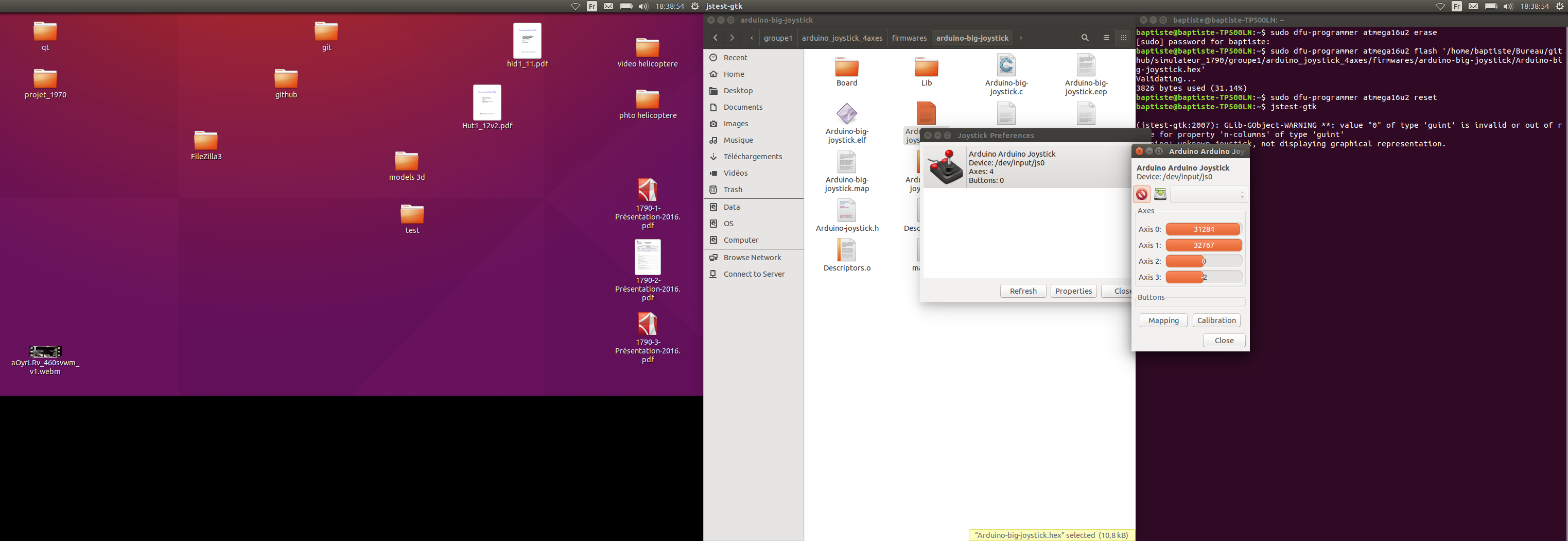
## Création de mon firmware

En partant donc des fichiers de projets déjà existants du projet unojoy je vais maintenant créer mon propre firmware(.hex).Il faut en tout 5 fichiers :

* Arduino-big-joystick.c et arduino-joytsick.h qui s’occupent de la configuration de l’arduino et de récupérer les informations (rapport USB) du programme dans l’arduino ;
* Descripeur.c et descripteur.h, transfèrent les informations récupérées par arduino-big-joytick.c pour les envoyer sur l’USB ;
* Et pour finir le fichier makefile qui permet de transformer le tout en un fichier. hex qui sera Chargé dans le microcontrôleur en mode DFU.

Tous utilisent bien-sûr les fonctions de la bibliothèque LUFA.

Il m’est maintenant très simple de créer mon propre firmware (après une lecture de la documentation officielle sur l’USB) qui permettra de créer un arduino-joystick de 4 axes (les fichiers se trouvent en annexe).



# Maquette cockpit hélicoptère

## Fonctionnement de la maquette

Le lycée a mis maintenant à disposition une maquette d’un cockpit d’hélicoptère contenant les quatre commandes que je dois simuler sur mon joystick-arduino.



Sur le dessous de la maquette se trouve un mécanisme basé sur le même fonctionnement que la première maquette, quand les commandes bougent, des potentiomètres changent de valeurs, il y a un potentiomètre par commande.

/\*photo du dessous \*/

/\*plus d’explication\*/

## Programme pour les quatre potentiomètres

Maintenant je peux créer entièrement mon propre programme puisque je maîtrise entière les informations que je dois recevoir dans le rapport.

Je dois donc récupérer les valeurs analogiques des quatre potentiomètres.

La structure joyReport\_t devient donc :

Et la lecture des quatre potentiomètres et l’écriture du rapport est cela :

## Reglage de Xplane

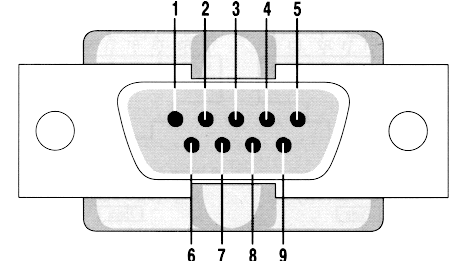
/\*a remplir \*/

# Boitier arduino

Pour le prototype du joystick-arduino, il m’ais parue intéressent de fabriquer un boitier l’y ranger.

Il serait ainsi plus transportable, et sur les façades de la boite des prises permettrais le branchement des capteurs.

## Prise capteur

La solution choisie est de prendre des prise standard DB9 pour brancher les capteurs sur le arduino-joystick.

Chaque capteur aura donc son propre connecteur avec un mapping diffèrent.

### DB9 Potentiomètre

|  |  |  |
| --- | --- | --- |
| PIN | Signal | Couleur fil |
| 1 | +3.3V | Rouge |
| 3 | Sortie potentiomètre | Bleu |
| 9 | masse | Noir |

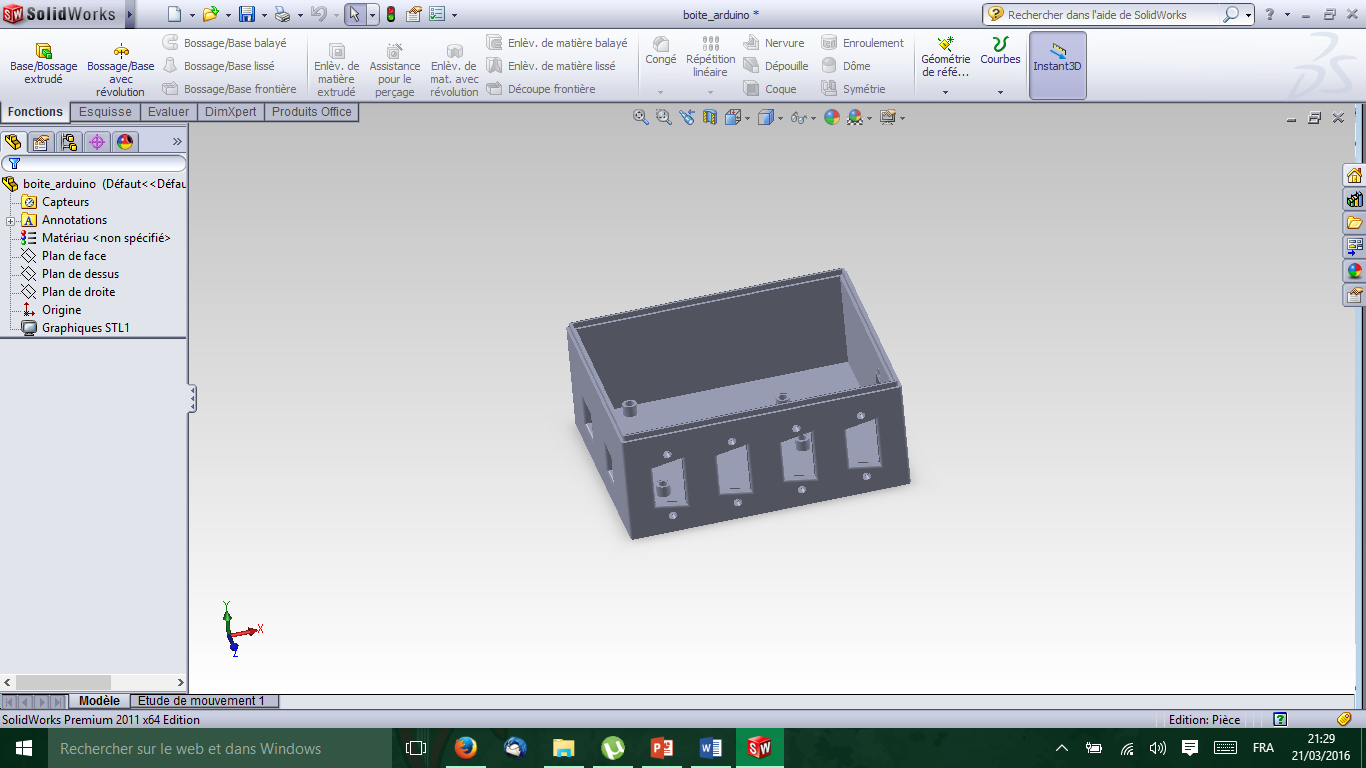
### DB9 Memsic2125

|  |  |  |
| --- | --- | --- |
| PIN | Signal | Couleur fil |
| 1 | +5V | Rouge |
| 3 | Xout | Bleu |
| 4 | Yout | Jaune |
| 9 | masse | Noir |

### DB9 ADXL345

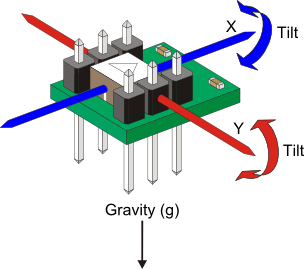
|  |  |  |
| --- | --- | --- |
| PIN | Signal | Couleur fil |
| 1 | VIO | Rouge |
| 2 | SCL | Bleu |
| 3 | SDO | Jaune |
| 4 | SDA | Rose |
| 5 | CS | Blanc |
| 7 | VS | Marron |
| 9 | GND | Vert |

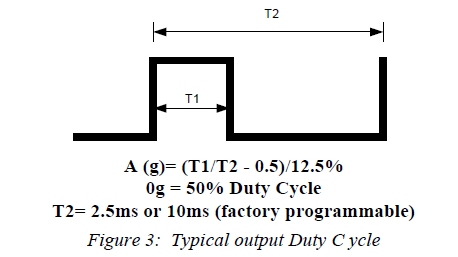
## Boite

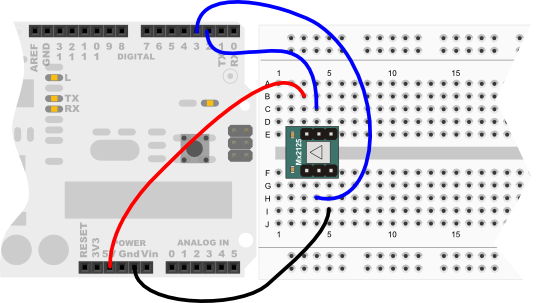


# Capteurs

## Memsic2125

Ce capteur est un accéléromètre qui retransmet les déplacements sur 2 axes (x, y), il est donc parfait pour la commande du pas cyclique.

Sont branchement et très simple, il suffit de brancher les sorties sur les ports digitaux pour y récupère le signal transmis par l’accéléromètre.



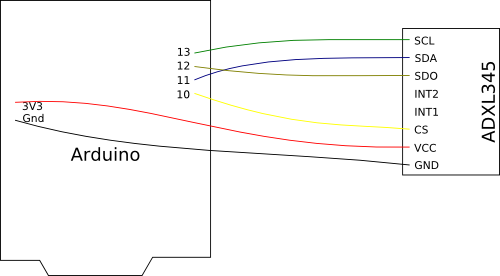
Le principe est simple en fonction de l’inclinaison du capteur sur l’axe, celui-ci transmet un bit a 1 de longueur T1 sur la période totale du signal T2. En fonction de ceci, on peut connaitre notre inclinaison sur l’axe.

La lecture de la durée de T1 peut être effectue par la fonction pulseIn() par default dans la librairie arduino.

## ADXL345

C’est lui aussi un accéléromètre mais bien plus précis et possède 3 axes. Mais la grande différence réside dans la manière de communiquer entre l’arduino et le capteur, puisque celui-ci peut transmettre les informations via une liaison SPI ou I2C.

### SPI

L’avantage du SPI est qu’il a un meilleur débit que le I2C et que le transfert d’information se fait sur deux fils diffèrent, un pour les communiquer en sortie et un en entrée. Il peut donc communiquer dans les deux sens en simultanée. Cette liaison utilise un fil supplémentaire par appareil pour sélectionnait avec lequel transférée les données.