

# Programmation Python de cartes Arduino

## 1. Présentation

Ce document présente les possibilités de programmation d'une carte Arduino sous Python.

La librairie « py2duino » définit des fonctions de commande de la carte Arduino. Cette bibliothèque est utilisable sous Windows mais également sous Mac et Linux. Cette librairie est conçue pour Python 3.x.


Le programme Python échange avec l'arduino via le port série et dialogue avec un programme tournant dans l'Arduino et chargé d'exécuter les consignes.

## 2. Préparatifs

### ▪ Charger le programme dans l'Arduino

La console Python communique avec un programme fonctionnant dans l'Arduino. Ce programme doit être chargé une seule fois au préalable.

Brancher la carte Arduino sur un port USB.

Ouvrir le logiciel de programmation Arduino  puis ouvrir le fichier « toolbox\_arduino\_v3.ino ». Sélectionner dans le menu « outil » le type de carte (UNO) et le port série utilisé. Téléverser le programme. Vous pouvez ensuite fermer le logiciel Arduino.

### ▪ Se connecter à la carte Arduino

Si nécessaire, il faut installer le module Pyserial pour python (<https://pypi.python.org/pypi/pyserial>) qui lui permet de gérer la communication via un port série type USB.

Dans la console, spécifier l'importation de la bibliothèque « py2duino » :

```
from py2duino import *
```

Pour amorcer la communication avec la carte Arduino, une première commande permet de définir le port série dédié. En supposant que le port série est le COM4 :

```
ar1=Arduino(4)
```

Le message "Connexion ok avec l'arduino" doit s'afficher. À noter que le programme Python peut communiquer avec plusieurs cartes connectées sur différents ports USB/série.

Dans la console, taper « ar1. » puis la touche tabulation et vous aurez un premier aperçu des commandes disponibles.

### 3. Gestion des sorties

#### 3.1 Clignotement d'une LED

Pour allumer et éteindre la LED 13 (présente sur la carte Arduino, ne nécessite pas de câblage), il faut configurer le port 13 en sortie puis allumer et éteindre la LED depuis python :

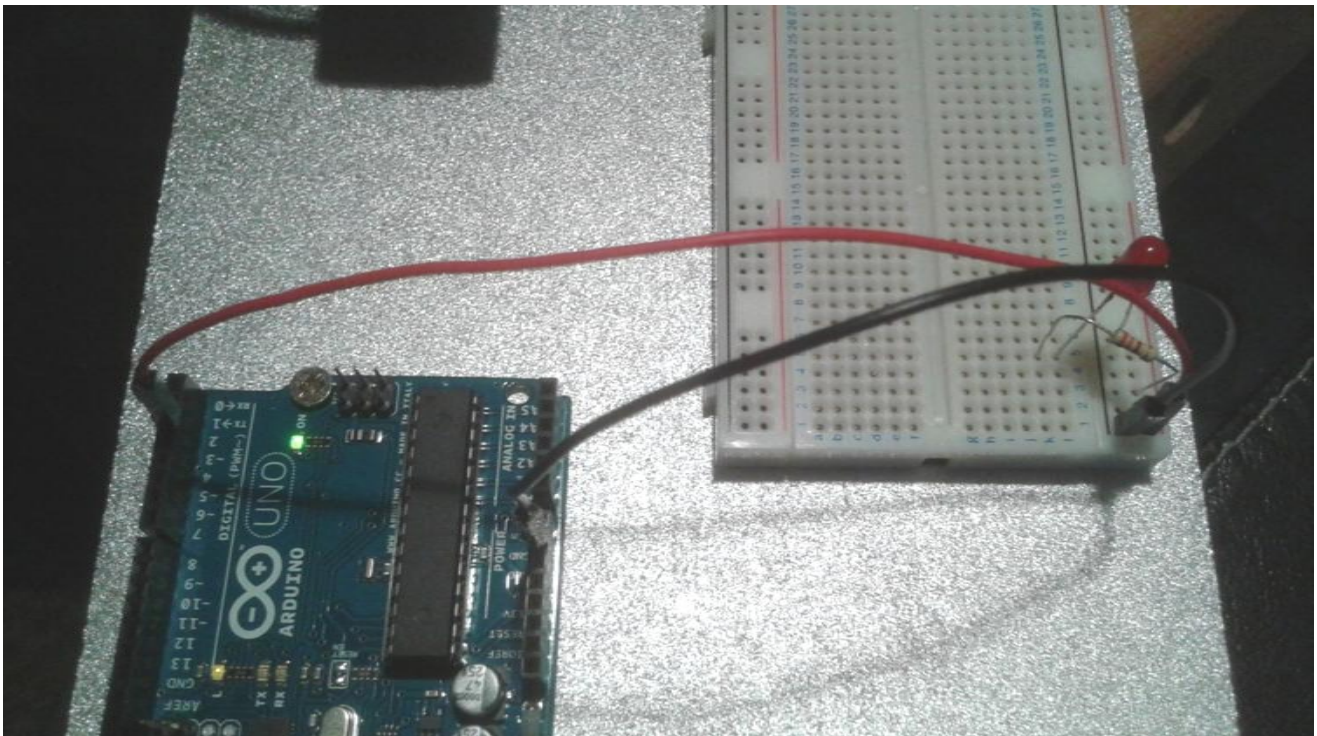
```
ar1.pinMode(13,"OUTPUT")
ar1.digitalWrite(13,"HIGH")
ar1.digitalWrite(13,"LOW")
```

Le programme ci-dessous (clignotement LED13.py) permet d'allumer 10 fois la LED avec une pause de 1s.

```
from py2duino import *
import time
ar1=Arduino(4)
ar1.pinMode(13,"OUTPUT")
i=1
while (i<11):
    ar1.digitalWrite(13,"HIGH")
    time.sleep(1) #pause 1s
    ar1.digitalWrite(13,"LOW")
    time.sleep(1) #pause 1s
    i+=1
```

Il est possible de faire la même chose avec une LED rouge montée sur le breadboard.

Il faut d'abord câbler la LED : un fil noir relié à la masse (GND), la LED, une résistance de 200Ω et un fil rouge relié au pin digital 2 de l'Arduino.



Le programme ci-dessous (clignotement LED Rouge.py) permet d'allumer 10 fois la LED avec une pause de 1s.

```
from py2duino import *
import time
ar1=Arduino(4)           # déclaration de l'Arduino connecté au COM4 en ar1
R=DigitalOutput(ar1,2)   # LED rouge (R) piloté par le pin digital 2
R.high()                 # Mise à l'état haut de la sortie => allumage LED
i=1
while (i<11):
    R.high()
    time.sleep(1) #pause 1s
    R.low()
    time.sleep(1) #pause 1s
    i+=1
```

### 3.2 Rotation Moteur

La rotation d'un moteur se commande comme une LED sauf qu'il faut piloter en sortie de la carte Arduino la commande de la carte d'alimentation du moteur et spécifier le sens de rotation (voir annexe câblage moteur). Le programme ci-dessous (Pilotage moteur.py) permet de faire tourner le moteur pendant 5s dans un sens puis dans l'autre sens après une pause de 2s.

```
from py2duino import *
import time

ar1=Arduino(4)           # déclaration de l'Arduino connecté au COM4 en ar1

A=DigitalOutput(ar1,7)   # Alimentation de la carte de puissance connecté au pin digital 7
A.high()                 # Mise à l'état haut du pin 7 = carte de puissance alimentée

R=DigitalOutput(ar1,5)    # Rotation du moteur piloté par le pin digital 5

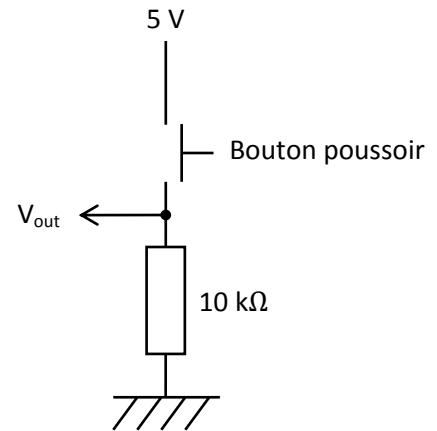
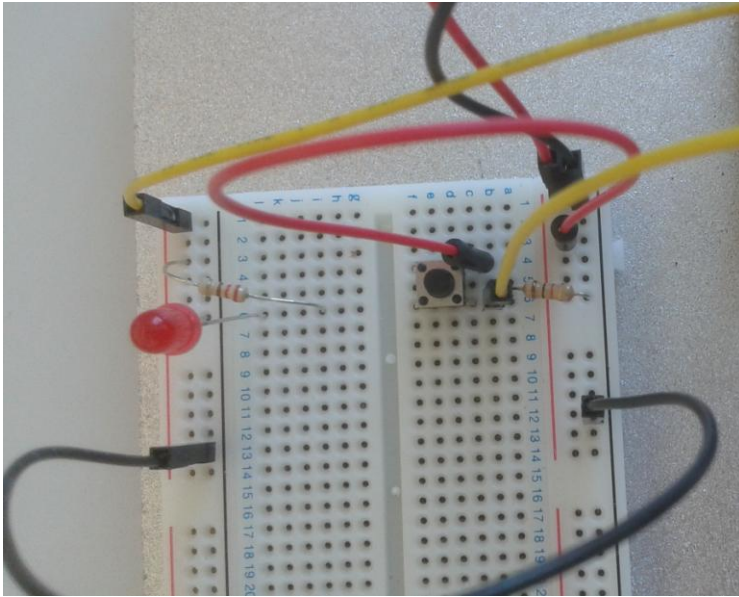
S=DigitalOutput(ar1,4)    # Sens de rotation du moteur piloté par le pin digital 4

S.high()
R.high()
time.sleep(5)            # Rotation pendant 5s dans un sens
R.low()
time.sleep(2)            # Pause 2s
S.low()
R.high()
time.sleep(5)            # Rotation pendant 5s dans l'autre sens
R.low()
```

## 4. Gestion des entrées

### 4.1 Bouton poussoir

Le câblage électrique d'un bouton poussoir est présenté ci-dessous. Quand le bouton poussoir est ouvert, la tension  $V_{out}$  est nulle et quand il est fermé  $V_{out}$  est égal à 5V. La tension  $V_{out}$  est à connecter à une entrée digitale de la carte Arduino qui sera à 0 sans appui sur le bouton et à 1 lors de l'appui.



Le programme ci-dessous (Allumage LED par bouton.py) permet d'allumer la LED quand lors d'un premier appui sur le bouton et de l'éteindre lors d'un deuxième appui.

```
from py2duino import *
import time

ar1=Arduino(4)
R=DigitalOutput(ar1,7)
R.low()
B=DigitalInput(ar1,2)

while (B.read()!=1) :
    time.sleep(1)
    print(B.read())

R.high()

time.sleep(2)

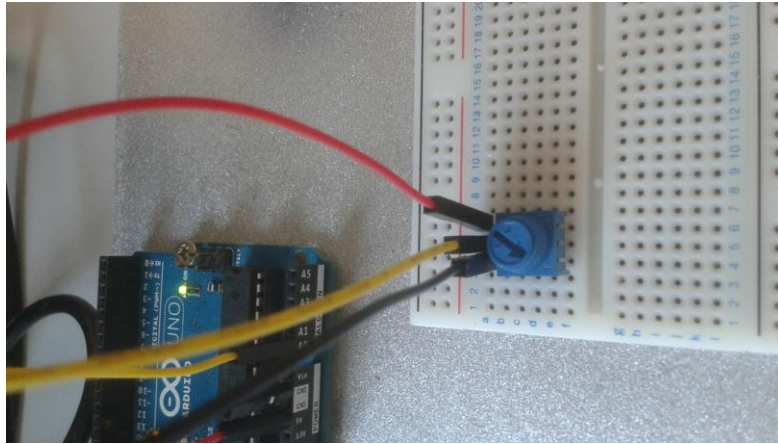
while (B.read()!=1) :
    time.sleep(1)
    print(B.read())

R.low()
```

# déclaration de l'Arduino connecté au COM4 en ar1  
# LED rouge (R) piloté par le pin digital 7  
# Mise à l'état bas de la sortie => LED éteinte  
# Bouton sur le pin digital 2 déclaré en entrée  
# boucle d'attente d'appui sur le bouton poussoir  
# contrôle de la valeur renvoyée par le bouton poussoir  
# Led allumée  
# boucle d'attente d'appui sur le bouton poussoir  
# contrôle de la valeur renvoyée par le bouton poussoir  
# Led éteinte

## 4.2 Potentiomètre

Un potentiomètre permet de faire varier une résistance proportionnellement à la rotation du bouton. Les deux broches extrêmes sont reliées pour l'une à la masse et pour l'autre à l'alimentation 5 V. La broche du milieu permet de recueillir une tension proportionnelle à la rotation du bouton (pont diviseur de tension). Elle doit être connectée à une entrée analogique de la carte Arduino.



Le programme ci-dessous (Allumage LED par potentiomètre.py) permet d'allumer la LED quand la valeur de la tension de sortie du potentiomètre dépasse 500 (tension max de 5V codée sur 1024).

```
from py2duino import *  
import time
```

```
ar1=Arduino(4)           # déclaration de l'Arduino connecté au COM4 en ar1  
R=DigitalOutput(ar1,2)   # LED rouge (R) piloté par le pin digital 2  
R.low()                  # Mise à l'état bas de la sortie => éteindre LED  
P=AnalogInput(ar1,0)     # Sortie du potentiomètre sur le pin analogique A0 déclaré en entrée
```

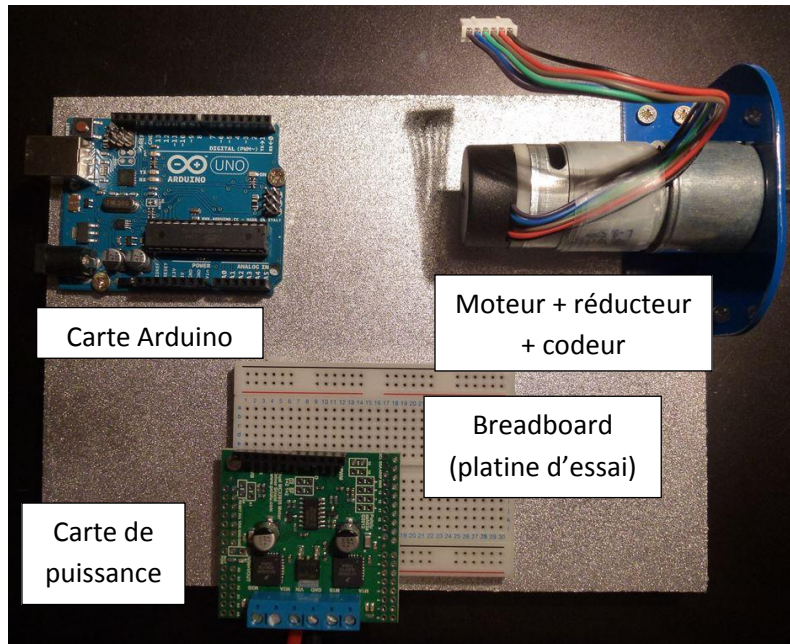
```
while P.read()<500 :     #boucle d'attente du dépassement de la valeur 500 en sortie du potentiomètre  
    time.sleep(1)  
    print(P.read())
```

```
R.high()
```

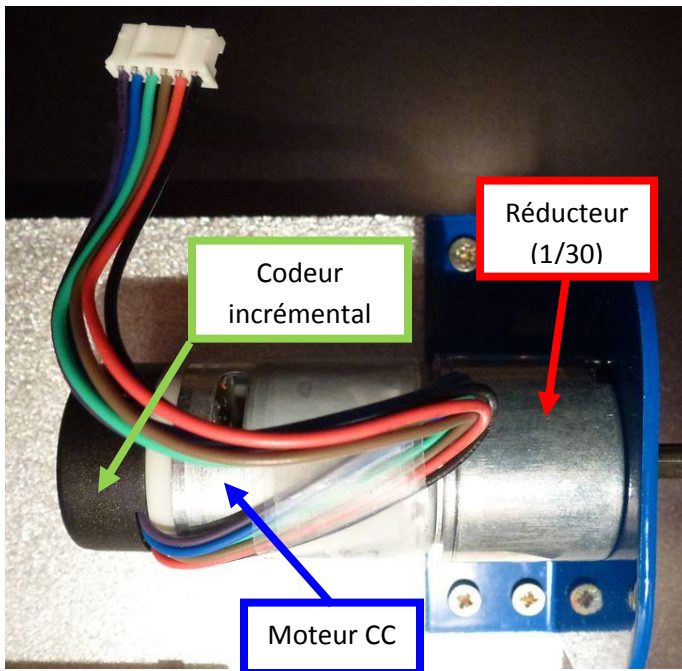


## Câblage Moteur

La platine d'essai à votre disposition est la suivante :



Les différents composants sont détaillés ci-dessous.



### Bloc moteur

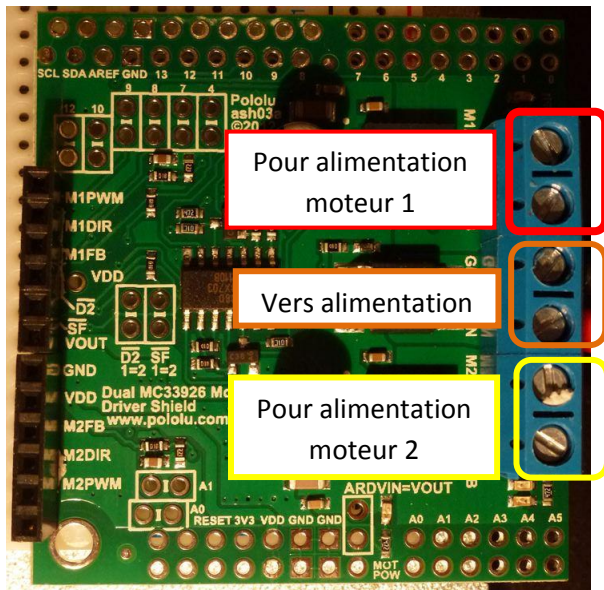
L'ensemble comprend un moteur à courant continu, un réducteur et un codeur incrémental.



Le fil rouge et le fil noir servent à alimenter le moteur à courant continu.

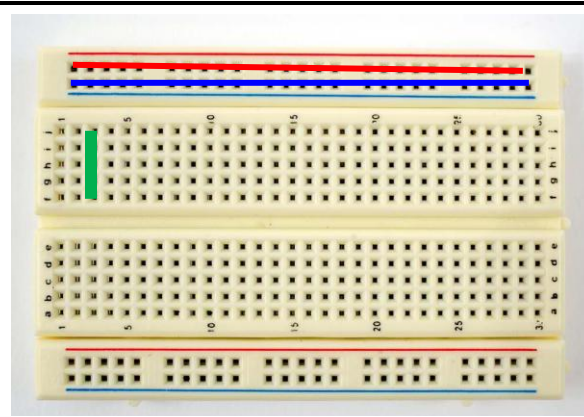
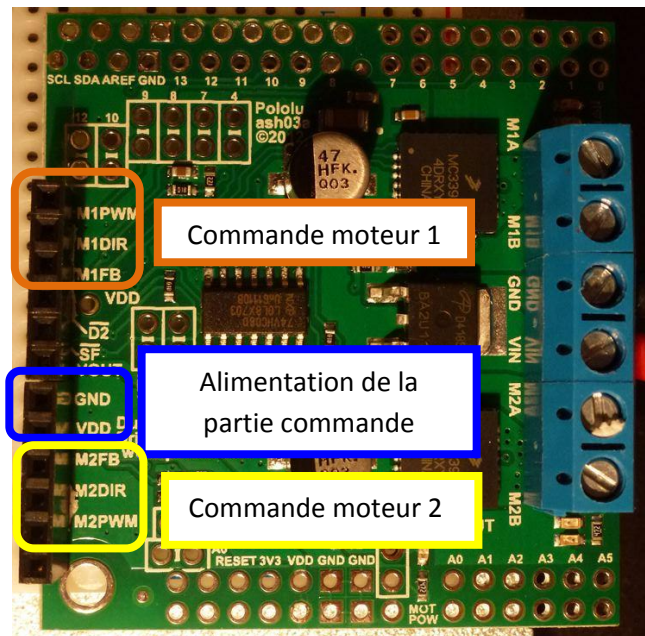
Le fil vert et le fil marron servent à alimenter le codeur.

Le fil bleu et le fil violet servent à récupérer les informations émises par le codeur.



### Carte de puissance

La carte de puissance utilisée permet de piloter deux moteurs (M1 et M2). On ne pilote ici qu'un moteur (M1).



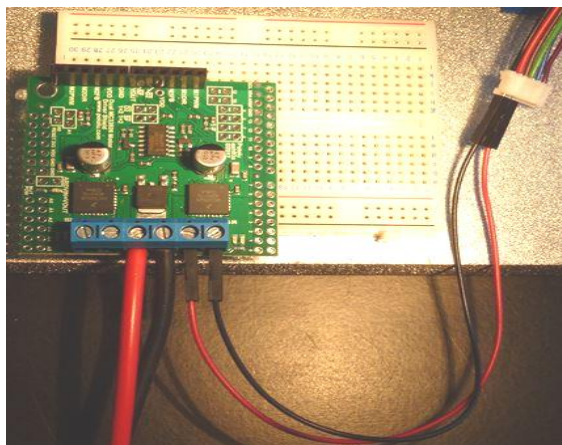
### Platine d'essai

Ces platines permettent de faciliter la mise en place de circuits électriques.

Les bornes situées de part et d'autre et accompagnées des traits bleu et rouge sont reliées entre elles suivant les lignes indiquées sur la platine (voir photo ci-contre).

Dans la partie centrale de la plaque, les bornes sont elles aussi reliées, suivant le trait vert indiqué ci-contre.

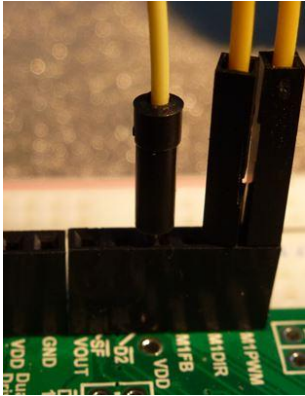
La carte de puissance est alimentée en 12 V par une alimentation extérieure. Le moteur est alimenté par la carte de puissance comme ci-dessous.



Pour permettre la transmission des ordres de la carte de commande Arduino vers la carte de puissance, trois fils reliant les bornes de la carte Arduino à la carte de puissance sont indispensables :

- le premier, relié à la borne M1PWM de la carte de puissance, permet de moduler la tension d'alimentation du moteur et donc de faire varier sa vitesse de rotation ;
- le deuxième, relié à la borne M1DIR de la carte de puissance, permet de définir le sens de rotation du moteur ;
- le troisième, relié à  $\overline{D2}$ , permet d'activer la carte de puissance.

Le câblage à réaliser est le suivant :



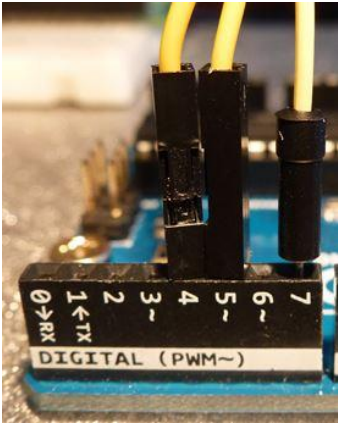
Carte de puissance

VD	GN	VO	SF	D2	M1	M1	M1
D	D	UT			FB	DIR	PWM

Prendre un fil jaune et le placer sur le pin M1PWM. Relier l'autre extrémité du fil au pin numérique 5 de la carte Arduino. Cette entrée est marquée d'un ~ afin d'indiquer qu'elle peut émettre une consigne variable MLI (Modulation de la Largeur d'Impulsion) vers une carte de puissance.

Prendre un fil jaune et le placer sur le pin M1DIR. Relier l'autre extrémité du fil au pin numérique 4 de la carte Arduino.

Prendre un fil jaune et le placer sur le pin  $\overline{D2}$  de la carte de puissance. Relier l'autre extrémité de ce fil au pin numérique 7 de la carte de puissance.

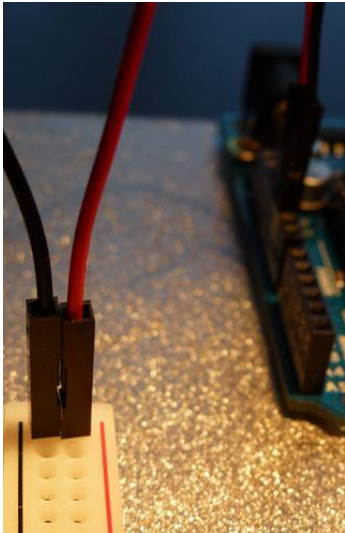


Carte Arduino

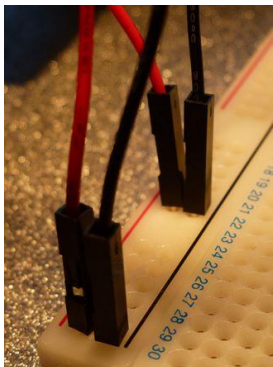
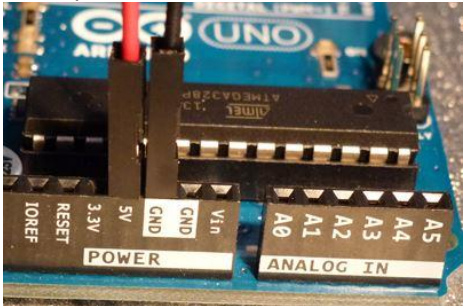
La carte Arduino peut désormais émettre des ordres à destination de la carte de puissance.



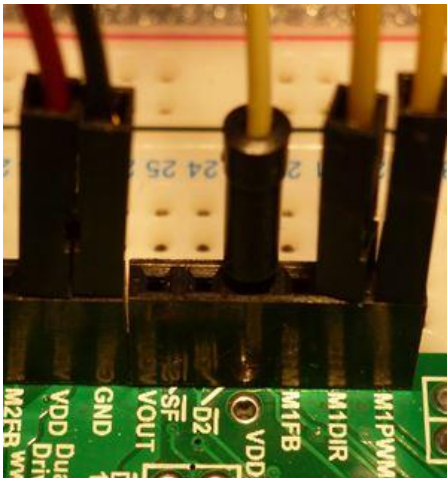
Cependant, la carte de puissance doit elle-même être alimentée. Si le moteur à courant continu nécessite une alimentation de 12 V, la partie commande de la carte peut être alimentée en 5 V. Cette alimentation se fait par l'intermédiaire de la carte Arduino et de la platine d'essai :



Avec un fil rouge, relier l'alimentation 5 V de la carte Arduino et la ligne positive (rouge) de la platine d'essai.  
Avec un fil noir, relier la masse (GND pour ground) de la carte Arduino et la ligne négative (noire) de platine d'essai.



A l'aide de deux autres fils rouge et noir, relier l'alimentation de la platine d'essai et la carte de puissance : le fil rouge doit se situer sur la borne VDD de la carte de puissance. Le fil noir sur la borne GND comme indiqué ci-dessous.



La carte de puissance est désormais alimentée.

