Etude de l'arrêt d'un moteur à courant continu

Nous allons étudier le régime transitoire qui accompagne l'arrêt d'un moteur à courant continu à l'aide d'une carte Arduino_uno et de programmes générés sous Python (distribution Spyder).

I- Le matériel et les configurations

Il vous faudra installer :

- Python3.5 (environnement spyder) : https://www.python.org/downloads/
- Arduino : https://www.arduino.cc/en/Main/Software

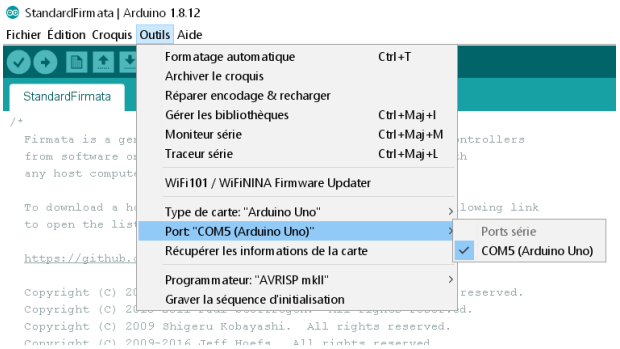
Ensuite, il sera nécessaire de configurer Arduino pour que la communication avec Python soit possible à l'aide du module Firmata :

Fichier → Exemple → Firmata → StandardFirmata

Téléversé ce programme.

Attention : la communication avec python pourra se rompre lors des différentes tentatives des programmes testés sous python. Il faudra rebrancher le câble USB puis téléverser une nouvelle fois Firmata

Il sera nécessaire de connaître le COM affecté à la carte Arduino, pour cela :*outils → Port*



Il reste une manipulation à faire sur python : installer le module Firmata avec les lignes de codes suivantes :

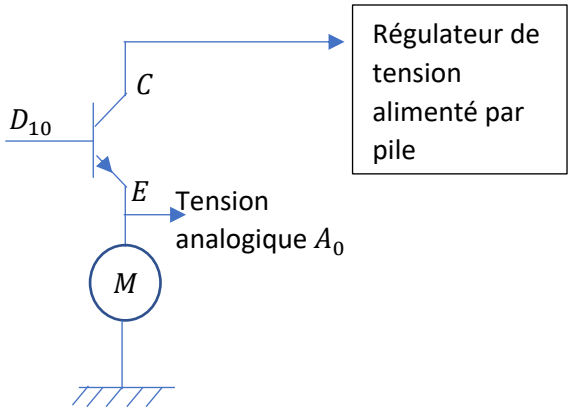
```
import pip
pip.main(['install','pyfirmata'])
```

On donne ci-dessous quelques lignes de code permettant de débuter une communication avec la carte.

```
from pyfirmata import Arduino, util
import time

#on déclare le bon port associé à la carte#
carte = Arduino('COM5')

#lancement de la communication avec la carte#
acquisition = util.Iterator(carte)
acquisition.start()

#Les entrées et sorties sur la carte se définissent grâce à la fonction get_pin()
#Avec pour un choix de paramètres dans la parenthèse :
#a analogique
#d digitale
#Le numéro de branchement sur la carte (de 0 à 13)
#i input , entrée
#o output, sortie
#p pwm
Exemple :
Pour déclarer la pin digitale 2 en entrée, on écrit :
digital_2 = carte.get_pin('d:2:i')
Pour déclarer la pin digitale 9 en sortie, on écrit :
digital_9 = carte.get_pin('d:9:o')
Pour lire la valeur de la pin 2, on écrit : digital_2.read()
Pour imposer une valeur à la pin, on écrit : digital_9.write(1)
```

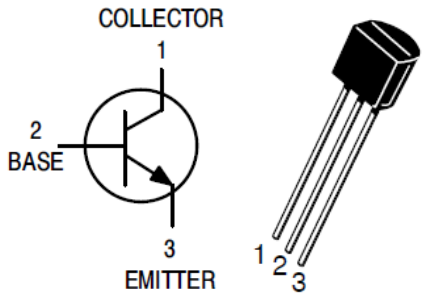
II- Alimentation

La carte Arduino va délivrer une tension digitale D_{10} afin de commander la marche ou l'arrêt du moteur à courant continu. Si D_{10} est à l'état haut alors le moteur tourne car le transistor T est passant (EC équivalent à un fil) et D_1 est à l'état bas alors le moteur n'est plus alimenté car le transistor T est bloqué (EC équivalent à un interrupteur ouvert).



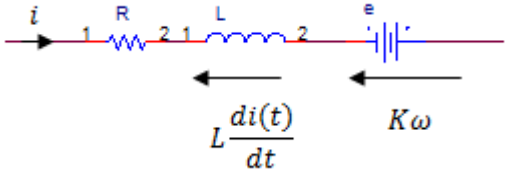
On mesure la tension analogique A_0 aux bornes du moteur.

On donne ci-dessous, le brochage du transistor $T(2N2222)$:



III- Modélisation du moteur lors de son arrêt

Nous avons vu en TD d'électromagnétisme qu'un « petit » moteur à courant continu se modélise par une tension induite e en série avec une inductance L et une résistance R :



Lorsque le transistor T est bloqué alors aucun courant ne traverse le moteur et la tension à ses bornes est $e = K\omega$ où K est la constante du moteur et ω sa vitesse de rotation. Parallèlement, d'un point de vu mécanique, il n'y a pas de moment des forces de Laplace (car pas de courant) mais il reste un moment M_0 du aux forces de frottement solide sur l'axe de rotation (à noter que le moteur tourne sans charge à entraîner). Nous ferons l'hypothèse d'un moment M_0 constant.

IV- Mesures

On donne ci-dessous le programme python permettant de commander un démarrage du moteur pendant 5s puis son arrêt. Pendant sa phase d'imobilisation la tension A0 est échantillonnée.

```
from pyfirmata import Arduino, util
import time

#on déclare le bon port associé à la
carte#
carte = Arduino('COM5')

#lancement de la communication avec la
carte"
acquisition = util.Iterator(carte)
acquisition.start()

consigne = carte.get_pin('d:10:o')
tension_moteur=carte.get_pin('a:0:i')

consigne.write(1)
time.sleep(5)#mise en rotation pendant
5s
t0=time.time()
t=time.time()-t0
tension=[]
temps=[]
consigne.write(0)#afin de relâcher la
liaison
while t<0.3: #0.3s correspond à un temps
typique d'immobilisation

tension.append(tension_moteur.read()*5)
    t =time.time()-t0
    temps.append(t)
    time.sleep(0.01)#fréquence
d'échantillonnage de 100Hz
carte.exit()
plt.plot(temps,tension)
plt.grid()
plt.show()
```

1) D'après le modèle proposé, comment varie $\omega(t)$ pendant la phase d'immobilisation ?
2) Faire une acquisition ainsi qu'un traitement informatisé qui permettra d'apprécier l'équation de cette courbe.

Pour cette dernière question on pourra s'aider de programme python déjà rencontrés en TP et aussi présent sur cahier de laboratoire.