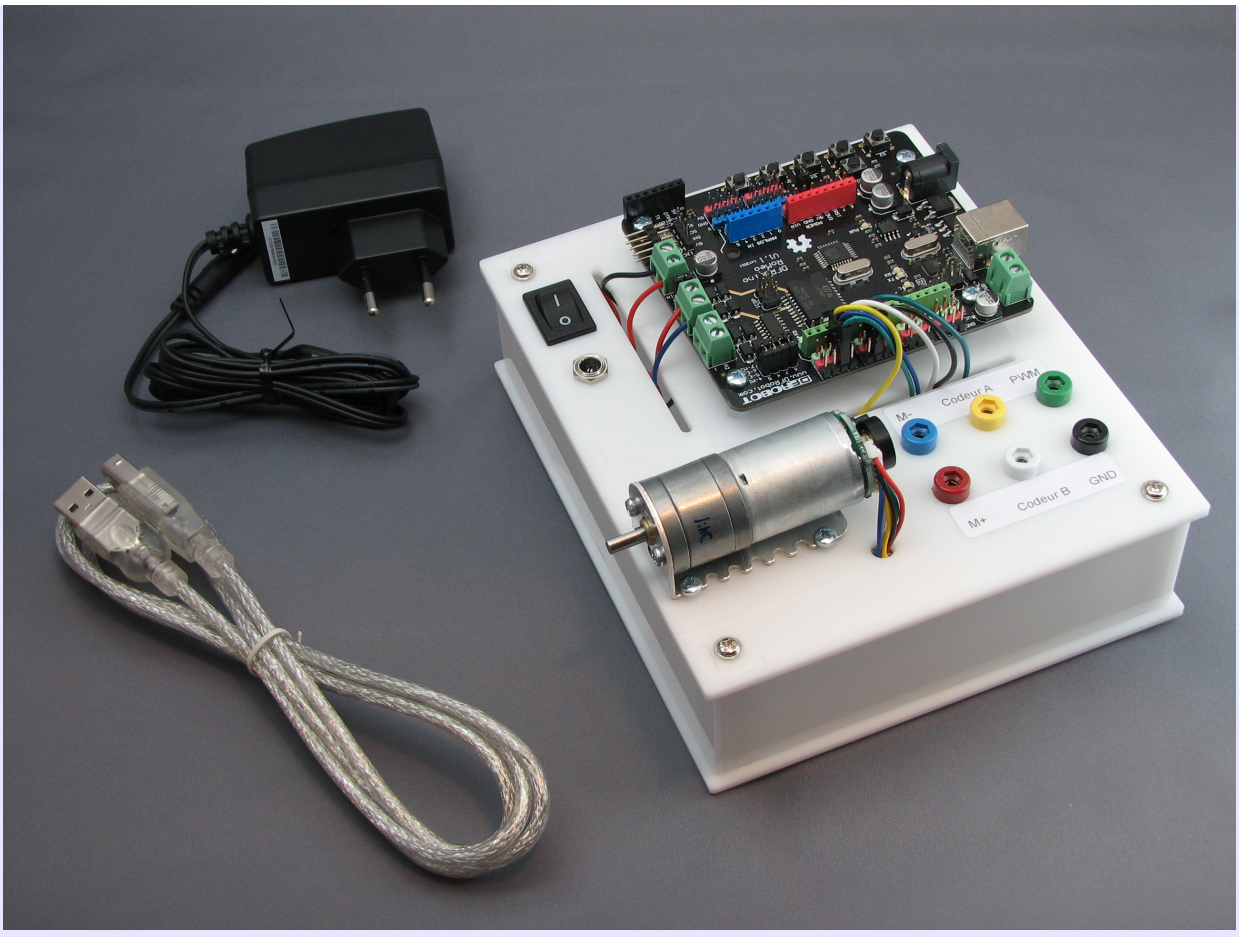
ENSEMBLE « COMMANDE DE MOTEUR À COURANT CONTINU »

DOCUMENTATION COMPLÈTE



Date de dernière mise à jour : 17/05/2017

Table des matières

1 - Introduction

Cet ensemble permet de réaliser différentes expériences sur la base d'un moteur à courant continu avec codeur incrémental associé, commandé par une carte Romeo, compatible Arduino Uno.

Une documentation détaillée de la carte Romeo ainsi que les programmes associés à cet ensemble sont téléchargeables sur notre site Web à l'adresse suivante:

http://www.3sigma.fr/telechargements

2 - Matériel inclus

Cet ensemble est livré monté et **fonctionnel (testé par nos soins avant la livraison)**. Il est composé des éléments suivants:

- un boîtier en plexiglas blanc, avec connecteur d'alimentation 5.5mm x 2.1mm, bouton marche-arrêt et douilles 2mm pour la mesure à l'oscilloscope de différents signaux.

  Repérage des douilles 2mm:

Rouge	Bleu	Jaune	Blanc	Vert	Noir
+ moteur	- moteur	Voie A codeur	Voie B codeur	Signal PWM	Masse

- une carte Romeo compatible Arduino Uno
- un moteur à courant continu 6V, rapport de réduction 34:1, avec codeur incrémental 48 CPR. Le moteur est assemblé sur le boîtier par l'intermédiaire d'un support de montage en équerre
- 1 câble USB A-B pour la programmation de la carte Romeo
- 1 alimentation 9V, 1A avec connecteur d'alimentation 5.5mm x 2.1mm

3 - Conformité

L'ensemble « Commande de moteur à courant continu », **dans sa configuration livrée aux clients**, est conforme à la directive 1999/EC.

4 - Installation de l'IDE Arduino

4.1 - Installation principale

L'IDE Arduino doit tout d'abord être téléchargé (http://arduino.cc/en/Main/Software) et installé (http://arduino.cc/en/Guide/HomePage). Il suffit de décompresser l'archive téléchargée dans le répertoire de votre choix. Notez bien ce répertoire car vous aurez besoin de le retrouver si pour installer des librairies additionnelles.

ATTENTION !
Avant de pouvoir compiler les programmes fournis avec cet ensemble, vous devez suivre les instructions suivantes pour installer les trois bibliothèques complémentaires **FlexiTimer2, digitalWriteFast**.et **EnableInterrupt**.

4.2 - Installation de la bibliothèque complémentaire FlexiTimer2

Cette bibliothèque permet d'exécuter à cadence fixe une partie du programme Arduino.

Vous pouvez la télécharger à l'adresse suivante: http://www.3sigma.fr/telechargements/FlexiTimer2.zip.

Une fois téléchargée, décompressez-là dans le répertoire des librairies de votre installation Arduino (typiquement, Documents\Arduino\libraries).

4.3 - Installation de la bibliothèque complémentaire digitalWriteFast

Cette bibliothèque permet de lire et d'écrire plus rapidement sur les entrées-sorties digitales de l'Arduino.

Vous pouvez la télécharger à l'adresse suivante:
http://www.3sigma.fr/telechargements/digitalWriteFast.zip.

Une fois téléchargée, décompressez-là dans le répertoire des librairies de votre installation Arduino (typiquement, Documents\Arduino\libraries).

4.4 - Installation de la bibliothèque complémentaire EnableInterrupt

Cette bibliothèque permet de gérer les interruptions d'une manière plus souple qu'avec les fonctions standard.

Vous pouvez la télécharger à l'adresse suivante:
http://www.3sigma.fr/telechargements/EnableInterrupt.zip.

Une fois téléchargée, décompressez-là dans le répertoire des librairies de votre installation Arduino (typiquement, Documents\Arduino\libraries).

ATTENTION !
L'environnement Arduino doit être redémarré après l'installation des bibliothèques complémentaires.

4.5 - Téléchargement des programmes

Dans la menu Outils → Type de carte de l'IDE Arduino, il faut sélectionner Arduino/Genuino Uno.

5 - Mise en œuvre de l'ensemble

5.1 - Utilisation standard

La mise en œuvre de l'ensemble « Commande de moteur à courant continu » est très simple:

- brancher l'alimentation 9V fournie sur le connecteur jack du plateau portant la carte Romeo et le moteur
- commuter l'interrupteur sur la position « I »
- connecter la carte Romeo à votre ordinateur à l'aide du câble USB fourni
- exécuter l'un des logiciels de pilotage (commande en tension, asservissement en vitesse ou en position, voir plus loin)

5.2 - Précautions d'emploi

Nous insistons sur le fait que cet ensemble est un matériel de développement qui nécessite un certain nombre de précautions d'emploi.

5.2.1 - Connexions d'alimentation sur la carte Romeo

Il est impératif de faire très attention aux connexions de l'alimentation de la carte Romeo car celle-ci n'est pas protégée contre les inversions de polarité. Une erreur de connexion sur les bornes d'alimentation risque d'entraîner la destruction du sous-ensemble de gestion d'alimentation de la carte et de rendre celle-ci inutilisable. L'ensemble « Commande de moteur à courant continu » étant livré connecté et fonctionnel, il est préférable de ne pas modifier les branchements sur les connecteurs d'alimentation.

5.2.2 - Tension d'alimentation

Cet ensemble est prévu pour fonctionner avec l'alimentation 9V, 1A fournie. Le moteur a une tension nominale de 6V et bien qu'il supporte sans problème des tensions jusqu'à 9V, le programme préchargé dans la carte (ainsi que tous les programmes téléchargeables sur notre site) empêche que la tension à ses bornes dépasse la valeur de 6V.

Il est possible d'utiliser un autre bloc d'alimentation à condition de bien respecter les points suivants:

- la tension ne doit pas dépasser 9V. Il est cependant recommandé de mettre éventuellement les programmes de la carte Arduino à jour si ceux-ci sont basés sur une valeur de tension différente
- la polarité doit être « positif au centre du connecteur »
- l'alimentation doit pouvoir fournir un courant suffisant, de préférence supérieur ou égal à 1A

5.2.3 - Utilisation

Il est fortement déconseillé de faire des expériences de fonctionnement « rotor bloqué » avec une tension d'alimentation du moteur trop élevée. Ce type d'expérience peut générer des courants trop forts qui réduisent la durée des vies des éléments.

6 - Expériences de commande de moteur à courant continu

Cet ensemble permet de réaliser différentes expériences de commande de moteur à courant continu.

6.1 - Caractéristiques du moteur à courant continu avec codeur incrémental associé

L'expérience de commande de moteur électrique embarque un moteur à courant continu 6V, de rapport de réduction 34:1, avec codeur incrémental 48 CPR (Counts Per Revolution).

Ce moteur est le même que celui utilisé sur le robot Geeros (http://boutique.3sigma.fr/12-robots).

Ses équations sont les suivantes:

$$\frac{\mathrm{d}}{\mathrm{d}t}\,\omega_m(t) = \frac{ratio\,K\,i_m(t) - d\,\omega_m(t)}{J \cdot ratio^2}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\,i_m(t) = \frac{V(t) - R\,i_m(t) - K \cdot ratio\,\omega_m(t)}{L}$$

Avec :

- R : résistance électrique interne: 3.0 Ohms
- L : inductance des enroulements: 3.0 mH
- J : moment d'inertie du rotor: 3.10^{-6} kg.m^2
- K : constante de couple = constante de fem: 0.01 N.m/A
- d : coefficient de frottement visqueux: 0.0025 N.m.s/rad
- ω_m : vitesse de rotation de l'arbre de sortie du réducteur (rad/s)
- i_m : courant dans le moteur (A)
- V : tension d'alimentation (V)

Ces paramètres ont été identifiés à partir d'un essai de réponse du moteur à un échelon de tension.

La fonction de transfert entre l'entrée *V(t)* et la sortie $\omega_m(t)$ est la suivante :

$$\frac{K\,ratio}{JL\,ratio^2\,s^2 + \left(JR\,ratio^2 + L\,d\right)s + K^2\,ratio^2 + R\,d}$$

La fonction de transfert entre l'entrée *V(t)* et la sortie $i_m(t)$ est la suivante :

$$\frac{J\,ratio^2\,s + d}{JL\,ratio^2\,s^2 + \left(JR\,ratio^2 + L\,d\right)s + K^2\,ratio^2 + R\,d}$$

Noter que les valeurs numériques données ci-dessus concernent l'identification du moteur avec son driver, l'influence de ce dernier étant importante et devant être pris en compte dans l'asservissement global du système.

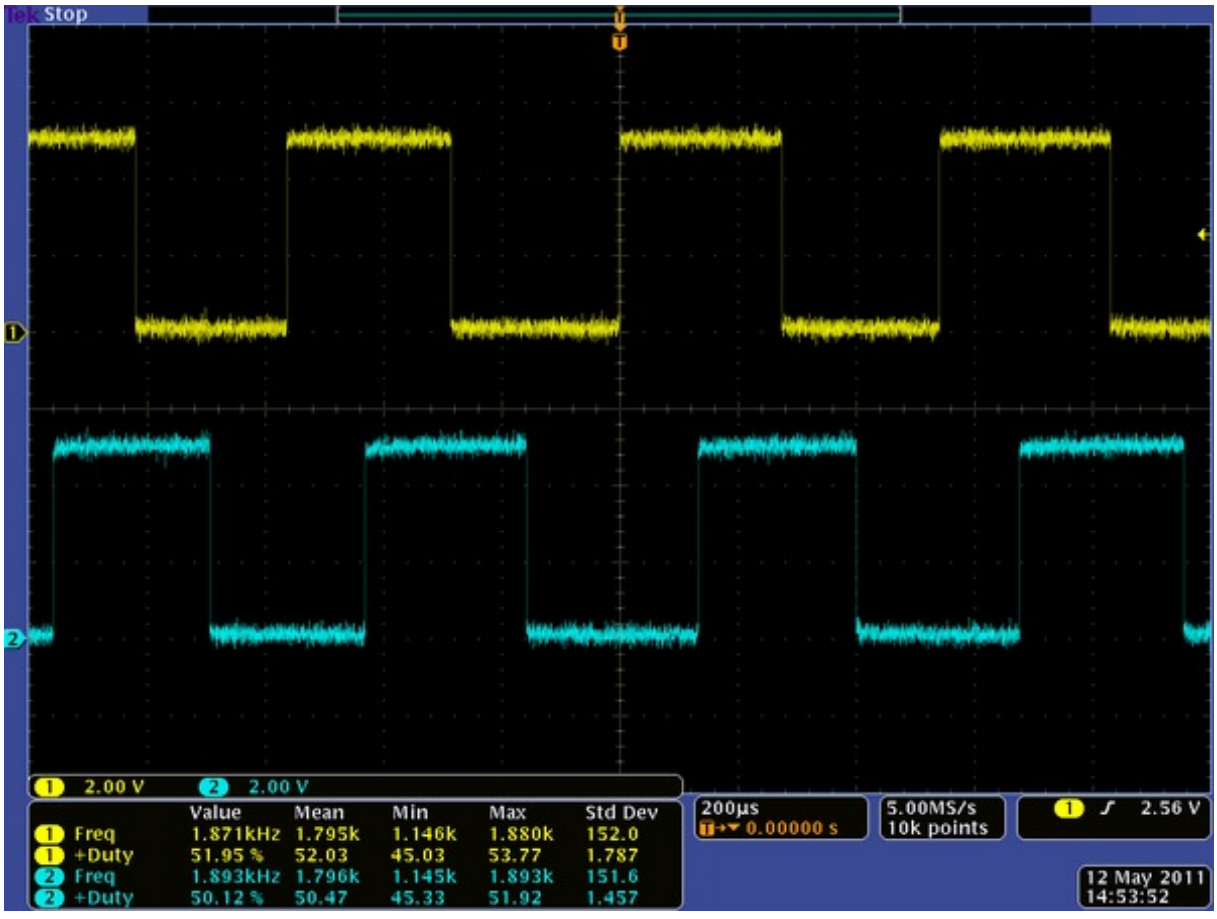
Le brochage du moteur (couleur des fils) est donné dans le tableau ci-dessous:

Rouge	Noir	Bleu	Vert	Jaune	Blanc
+ moteur	- moteur	5V codeur	Masse codeur	Voie A codeur	Voie B codeur

ATTENTION !
Ne pas confondre la couleur des fils du moteur et la couleur des douilles 2mm.

6.1.1 - Calcul de la vitesse de rotation du moteur

Le codeur incrémental fournit deux signaux carrés en quadrature, comme sur la capture ci-dessous:



Ces deux signaux permettent de mesurer à la fois la vitesse et le sens de rotation. La mesure de la vitesse se fait simplement en comptant le nombre d'impulsions pendant un temps fixe. Les données du problème sont les suivantes :

- Le codeur est fixé à l'arbre moteur et non pas à l'arbre de sortie du réducteur (celui utilisé pour l'entraînement). Le rapport de réduction étant 34:1, l'arbre moteur fait 34 tours lorsque l'arbre « principal » en fait 1
- Le codeur génère 48 impulsions à chaque fois qu'il fait un tour
- La cadence d'échantillonnage utilisée pour l'asservissement sera de 0.01 s

Par conséquent, lorsque l'arbre principal fait un tour, le codeur génère :
34 * 48 = 1632 impulsions.

Si N est le nombre d'impulsions comptées en 0.01 s, la vitesse est (en rad/s, l'unité standard, sachant qu'un tour fait 2*π radians) :
2*π*N/(0.01*1632)

ATTENTION !
Bien que le codeur soit placé sur l'arbre moteur, le calcul ci-dessus donne la vitesse en sortie du réducteur.

Un point très important concerne la résolution de la mesure, c'est-à-dire la plus petite valeur qu'il est possible de calculer. La formule est la suivante (en rad/s) :
2*π/(Ts*CPR*ratio)
avec :

- Ts : cadence d'échantillonnage
- CPR : nombre d'impulsions par tour du codeur
- ratio : rapport de réduction du moteur

Dans notre cas de figure, la résolution est la suivante
2*π/(0.01*1632) = 0.4 rad/s

6.1.2 - Comptage du nombre d'impulsions

Compter le nombre d'impulsions du codeur revient à compter le nombre de fronts montants et descendants des signaux jaune et bleu représentés sur l'image ci-dessus. Pour ce faire, la seule méthode viable consiste à brancher les deux signaux (les fils jaune et blanc sur le codeur utilisé) sur deux entrées « interruption » de la carte Arduino. Les deux autres fils (bleu et vert) seront respectivement branchés sur le 5 V et sur la masse de l'Arduino.

Sur une carte Romeo (comme sur un Arduino Uno d'ailleurs), il y a deux lignes d'interruption (numérotées 0 et 1), qui correspondent aux broches digitales 2 et 3. L'intérêt d'une ligne d'interruption est qu'elle permet, comme son nom l'indique, d'interrompre le déroulement des calculs sur le micro-contrôleur pour effectuer un traitement spécifique, en l'occurrence la mise à jour du compteur d'impulsions, avant de rendre la main à la boucle principale.

La seule « difficulté » est de savoir s'il faut incrémenter ou décrémenter le compteur dans le traitement de l'interruption. Il suffit pour cela d'observer les courbes ci-dessus, obtenues alors que le moteur tourne dans le sens positif. On constate que:

- Lorsque la voie A (en jaune) passe au niveau haut, la voie B (en bleu) est au niveau bas
- Lorsque la voie A passe au niveau bas, la voie B est au niveau haut

Quand le moteur tourne dans le sens positif, lors d'une interruption sur la voie A, les niveaux de A et B sont donc inversés.

En ce qui concerne l'interruption liée à la voie B, c'est l'inverse :

- Lorsque la voie B passe au niveau haut, la voie A est au niveau haut
- Lorsque la voie B passe au niveau bas, la voie A est au niveau bas

Le code permettant de compter les impulsions est implémenté à la fin des programmes Arduino associés à ce système.

6.2 - Commande en tension

Cette expérience (pré-chargée dans la carte Romeo à la livraison) permet de changer la vitesse de rotation du moteur en appliquant une tension variable par l'intermédiaire d'une application qui s'exécute sur votre ordinateur.

Le programme Arduino peut être téléchargé à l'adresse suivante:

http://www.3sigma.fr/Telechargements-Ensemble_Commande_de_moteur_a_courant_continu.html

Son nom est de la forme CommandeEnTension_x.y.zip (x.y correspond au numéro de version du programme).

Il comporte de nombreux commentaires permettant de comprendre facilement son fonctionnement.

Le principe de pilotage du moteur consiste à envoyer des signaux PWM sur le pont en H intégré à la carte Romeo afin de faire varier la tension d'alimentation du moteur. Cette tension peut être modifiée interactivement via une application qui s'exécute sur votre ordinateur. Elle peut se télécharger à l'adresse suivante:

http://www.3sigma.fr/Telechargements-Ensemble_Commande_de_moteur_a_courant_continu.html

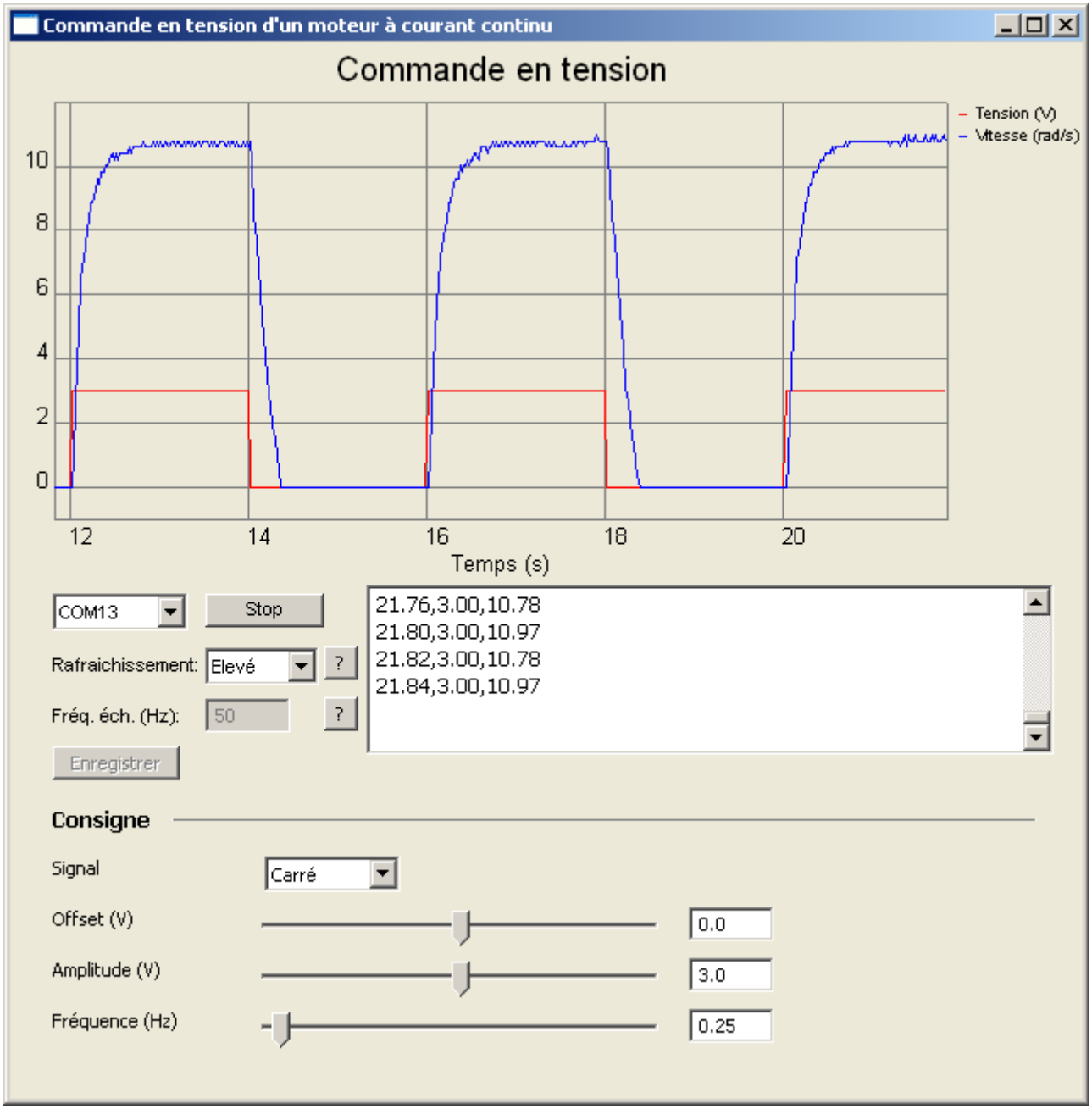
Son nom est de la forme CommandeMoteurEnTension_x.y.zip (x.y correspond au numéro de version du programme).

Pour l'installer, il suffit de décompresser l'archive dans le répertoire de votre choix. Pour l'exécuter, double-cliquer sur CommandeMoteurEnTension_x.y.exe.

Pour piloter la tension d'alimentation (et donc la vitesse) du moteur depuis votre ordinateur, il vous suffit de suivre les étapes suivantes (remarque: n'exécutez pas de nouveau celles que vous avez déjà effectuées):

- télécharger l'application de pilotage « CommandeMoteurEnTension » à l'adresse ci-dessus et l'installer
- télécharger et installer l'IDE Arduino (voir paragraphe 4)
- connecter l'alimentation 9V sur le connecteur jack du plateau supportant la carte Romeo et le moteur (voir paragraphe 5.1)
- connecter la carte Romeo à l'ordinateur avec le câble USB fourni (voir paragraphe 5.1)
- mettre le système sous tension en positionnant le bouton marche-arrêt sur « I » (voir paragraphe 5.1)
- lancer l'application « CommandeMoteurEnTension »

Voici une capture d'écran de l'interface de l'application de pilotage:



Les différents éléments sont les suivants (de haut en bas)

- courbes: l'interface permet de visualiser
  - la tension de commande (V, en rouge)
  - la vitesse mesurée (rad/s, en bleu)
- zone de sélection du port série: choisir le port série sur lequel est connectée votre carte Romeo et cliquer sur le bouton « Connexion ». Vous verrez alors les courbes se mettre à jour automatiquement et défiler des valeurs numériques dans la zone d'affichage à droite
- « Rafraîchissement »: ceci pilote la fréquence de rafraîchissement des courbes. En fonction de la vitesse de votre ordinateur, vous pouvez choisir parmi les 4 valeurs « Minimum », « Lent », « Moyen » et « Rapide ». Plus la fréquence de rafraîchissement est élevée, moins le tracé est saccadé. Mais si votre ordinateur n'est pas très rapide, vous risquez d'observer un retard entre les consignes et l'affichage. Dans ce cas, il faut choisir un rafraîchissement plus lent
- « Fréq. éch. (Hz) »: ce paramètre correspond à la fréquence d'échantillonnage des mesures dans le programme Arduino. Attention: ce paramètre doit être spécifié en Hz, alors que les valeurs correspondantes dans le programme Arduino (CADENCE_MS et TSDATA) sont spécifiées en ms. La relation entre les deux est la suivante:
  freq (Hz) = 1000 / cadence (ms)
- Bouton « Enregistrer »: il permet d'enregistrer les valeurs numériques affichées dans la zone de « log » vers un fichier texte pour une utilisation ultérieure avec n'importe quel logiciel permettant de lire des données séparées par des virgules dans une fichier texte.
  Attention : ce bouton n'est actif qu'après une séquence de mesure. Il est grisé le reste du temps (au démarrage du programme et pendant une séquence de mesure)
- Zone « Consigne »: elle permet de modifier la consigne de tension avec les curseurs. La vitesse du moteur varie alors. Vous pouvez la visualiser, ainsi que la consigne et la tension de commande, sur le graphique qui s'affiche en temps-réel

IMPORTANT !
Si l'affichage ne suit pas les consignes même avec une fréquence de rafraichissement très lente, veuillez diminuer la cadence d'échantillonnage dans le programme Arduino (augmenter les valeurs TSDATA et CADENCE_MS)

IMPORTANT !
Si vous comparez la vitesse de rotation obtenue sur votre ensemble avec celle affichée sur la capture d'écran au début de ce paragraphe, vous obtiendrez probablement une valeur différente, même si la tension de commande est la même : c'est normal, ceci est dû à la disparité des caractéristiques des moteurs à courant continus.
D'un point de vue pédagogique, ce point permet de souligner la nécessité de réaliser un asservissement de vitesse : si l'on souhaite une vitesse de rotation précise, on ne peut pas se contenter d'une commande en boucle ouverte.

6.3 - Asservissement de vitesse

Cette expérience permet d'asservir la vitesse de rotation du moteur en appliquant une consigne de vitesse par l'intermédiaire d'une application qui s'exécute sur votre ordinateur.

Le programme Arduino peut être téléchargé à l'adresse suivante:

http://www.3sigma.fr/Telechargements-Ensemble_Commande_de_moteur_a_courant_continu.html

Son nom est de la forme AsservissementVitesse_x.y.zip (x.y correspond au numéro de version du programme).

Il comporte de nombreux commentaires permettant de comprendre facilement son fonctionnement.

Le principe de pilotage du moteur consiste à calculer, grâce à un régulateur de type PID, la tension de commande à appliquer au moteur (via commande PWM du pont en H intégré sur la carte Romeo) pour qu'il suive la consigne de vitesse spécifiée.

Notez que dans ce programme, la vitesse mesurée est en fait la moyenne glissante des 10 derniers échantillons de mesure de vitesse instantanée, ce qui permet d'avoir une mesure plus lisse. En effet, la résolution de la mesure instantanée est la suivante:

$2*\pi/(Ts*CPR*ratio)$
avec :

- Ts : cadence d'échantillonnage (0.01 s)
- CPR : nombre d'impulsions par tour du codeur (48)
- ratio : rapport de réduction du moteur (34)

Ceci donne une résolution de $2*\pi/(0.01*1632) = 0.4$ rad/s. Le moyennage permet d'améliorer cette valeur.

La consigne de vitesse peut se fixer interactivement via une application qui s'exécute sur votre ordinateur. Elle peut se télécharger à l'adresse suivante:

http://www.3sigma.fr/Telechargements-Ensemble_Commande_de_moteur_a_courant_continu.html

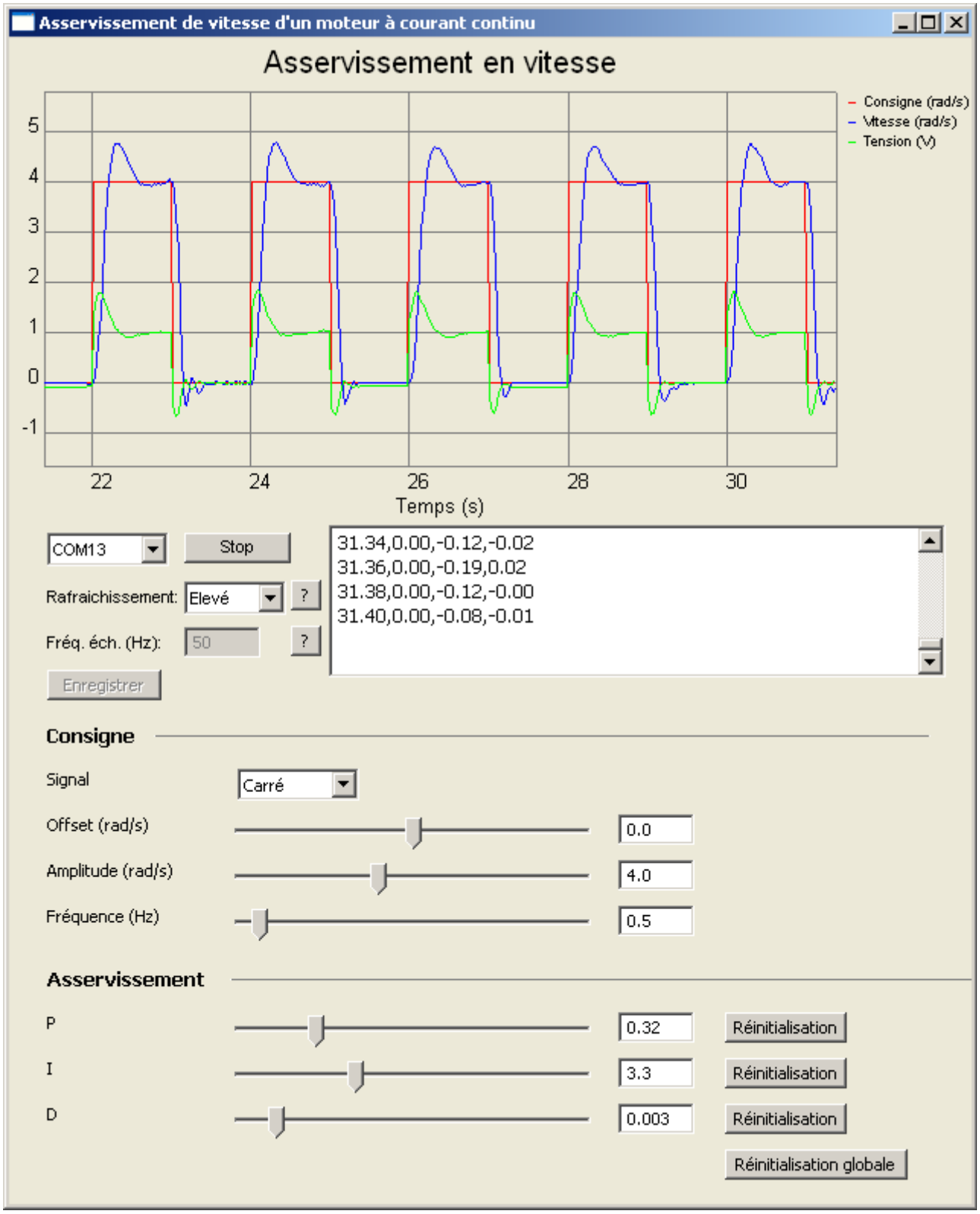
Son nom est de la forme AsservissementMoteurEnVitesse_x.y.zip (x.y correspond au numéro de version du programme).

Pour l'installer, il suffit de décompresser l'archive dans le répertoire de votre choix. Pour l'exécuter, double-cliquer sur AsservissementMoteurEnVitesse_x.y.exe.

Pour changer la consigne de vitesse du moteur depuis votre ordinateur, il vous suffit de suivre les étapes suivantes (remarque: n'exécutez pas de nouveau celles que vous avez déjà effectuées):

- télécharger l'application de pilotage «AsservissementMoteurEnVitesse» à l'adresse ci-dessus et l'installer
- télécharger et installer l'IDE Arduino (voir paragraphe 4)
- connecter l'alimentation 9V sur le connecteur jack du plateau supportant la carte Romeo et le moteur (voir paragraphe 5.1)
- connecter la carte Romeo à l'ordinateur avec le câble USB fourni (voir paragraphe 5.1)
- mettre le système sous tension en positionnant le bouton marche-arrêt sur « I » (voir paragraphe 5.1)
- lancer l'application «AsservissementMoteurEnVitesse»

Voici une capture d'écran de l'interface de l'application de pilotage:



Les différents éléments sont les suivants (de haut en bas)

- courbes: l'interface permet de visualiser
  - la consigne de vitesse (rad/s, en rouge)
  - la vitesse mesurée (rad/s, en bleu)
  - la tension de commande (V, en vert)
- zone de sélection du port série: choisir le port série sur lequel est connectée votre carte Romeo et cliquer sur le bouton « Connexion ». Vous verrez alors les courbes se mettre à jour automatiquement et défiler des valeurs numériques dans la zone d'affichage à droite
- « Rafraîchissement »: ceci pilote la fréquence de rafraîchissement des courbes. En fonction de la vitesse de votre ordinateur, vous pouvez choisir parmi les 4 valeurs « Minimum », « Lent », « Moyen » et « Rapide ». Plus la fréquence de rafraîchissement est élevée, moins le tracé est saccadé. Mais si votre ordinateur n'est pas très rapide, vous risquez d'observer un retard entre les consignes et l'affichage. Dans ce cas, il faut choisir un rafraîchissement plus lent
- « Fréq. éch. (Hz) »: ce paramètre correspond à la fréquence d'échantillonnage des mesures dans le programme Arduino. Attention: ce paramètre doit être spécifié en Hz, alors que la valeur correspondante dans le programme Arduino (TSDATA) est spécifiée en ms. La relation entre les deux est la suivante:
  freq (Hz) = 1000 / TSDATA (ms)
- Bouton « Enregistrer »: il permet d'enregistrer les valeurs numériques affichées dans la zone de « log » vers un fichier texte pour une utilisation ultérieure avec n'importe quel logiciel permettant de lire des données séparées par des virgules dans une fichier texte.
  Attention : ce bouton n'est actif qu'après une séquence de mesure. Il est grisé le reste du temps (au démarrage du programme et pendant une séquence de mesure)
- Zone « Consigne »: elle permet de modifier la consigne de vitesse avec les curseurs. La vitesse du moteur varie alors. Vous pouvez la visualiser, ainsi que la consigne et la tension de commande, sur le graphique qui s'affiche en temps-réel
- Zone « Asservissement »: elle permet de modifier les gains du régulateur PID pendant le fonctionnement du système. Cela permet de voir l'influence de ces gains sur les performances de l'asservissement et de régler précisément ce dernier.

IMPORTANT !
Si l'affichage ne suit pas les consignes même avec une fréquence de rafraichissement très lente, veuillez diminuer la cadence d'échantillonnage dans le programme Arduino (augmenter la valeur TSDATA). Ne pas modifier la valeur CADENCE_MS, sous peine de déstabiliser l'asservissement.

6.4 - Asservissement de position

Cette expérience permet d'asservir la position angulaire du moteur en appliquant une consigne d'angle par l'intermédiaire d'une application qui s'exécute sur votre ordinateur.

Le programme Arduino peut être téléchargé à l'adresse suivante:

http://www.3sigma.fr/Telechargements-Ensemble_Commande_de_moteur_a_courant_continu.html

Son nom est de la forme AsservissementPosition_x.y.zip (x.y correspond au numéro de version du programme).

Il comporte de nombreux commentaires permettant de comprendre facilement son fonctionnement.

Le principe de pilotage du moteur consiste à calculer, grâce à un régulateur de type PID, la tension de commande à appliquer au moteur (via commande PWM du pont en H intégré sur la carte Romeo) pour qu'il suive la consigne de position spécifiée.

La consigne de position peut se fixer interactivement via une application qui s'exécute sur votre ordinateur. Elle peut se télécharger à l'adresse suivante:

http://www.3sigma.fr/Telechargements-Ensemble_Commande_de_moteur_a_courant_continu.html

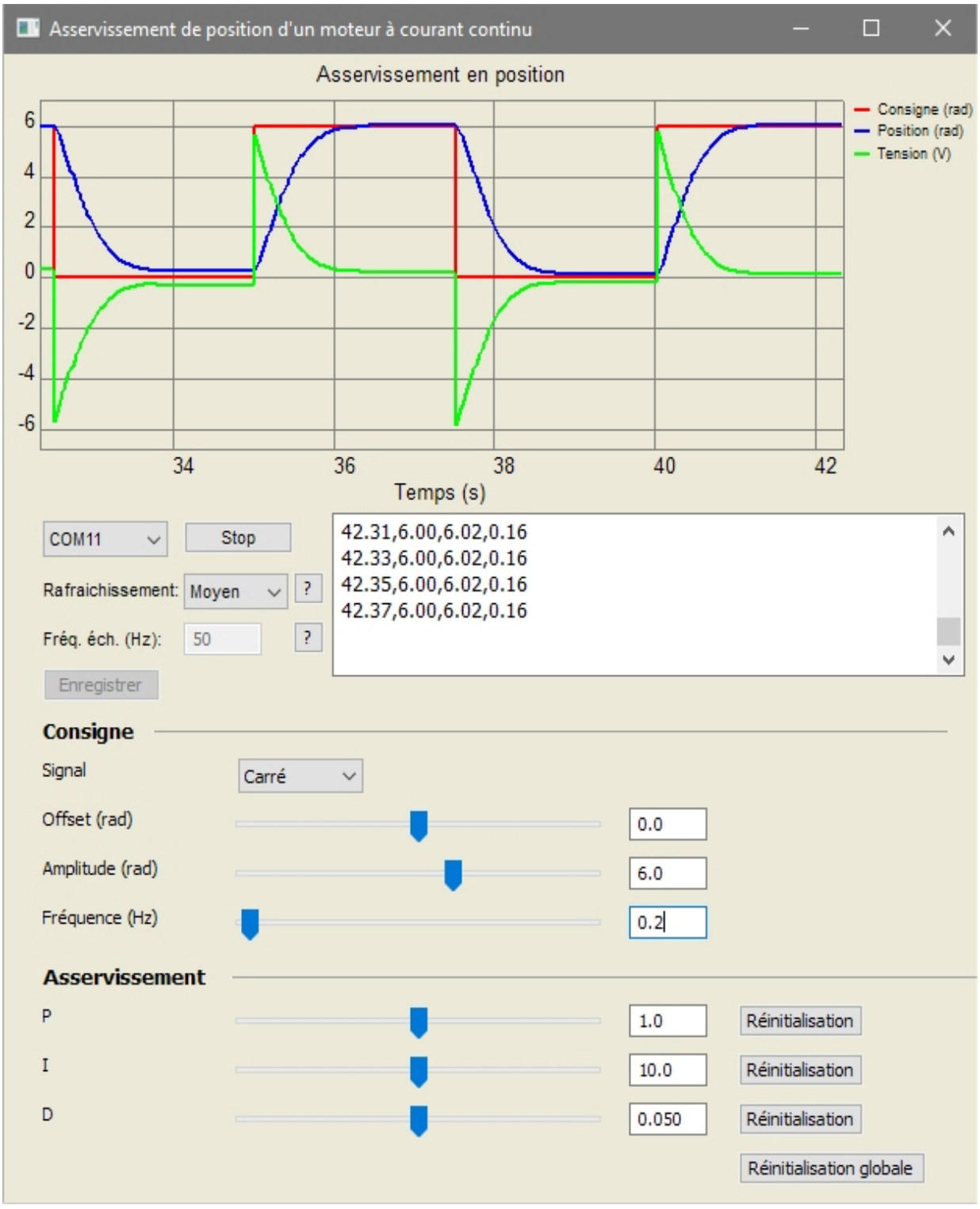
Son nom est de la forme AsservissementMoteurEnPosition_x.y.zip (x.y correspond au numéro de version du programme).

Pour l'installer, il suffit de décompresser l'archive dans le répertoire de votre choix. Pour l'exécuter, double-cliquer sur AsservissementMoteurEnPosition_x.y.exe.

Pour changer la consigne de position du moteur depuis votre ordinateur, il vous suffit de suivre les étapes suivantes (remarque: n'exécutez pas de nouveau celles que vous avez déjà effectuées):

- télécharger l'application de pilotage «AsservissementMoteurEnPosition» à l'adresse ci-dessus et l'installer
- télécharger et installer l'IDE Arduino (voir paragraphe 4)
- connecter l'alimentation 9V sur le connecteur jack du plateau supportant la carte Romeo et le moteur (voir paragraphe 5.1)
- connecter la carte Romeo à l'ordinateur avec le câble USB fourni (voir paragraphe 5.1)
- mettre le système sous tension en positionnant le bouton marche-arrêt sur « I » (voir paragraphe 5.1)
- lancer l'application «AsservissementMoteurEnPosition»

Voici une capture d'écran de l'interface de l'application de pilotage:



Les différents éléments sont les suivants (de haut en bas)

- courbes: l'interface permet de visualiser
  - la consigne de position (rad, en rouge)
  - la position mesurée (rad, en bleu)
  - la tension de commande (V, en vert)
- zone de sélection du port série: choisir le port série sur lequel est connectée votre carte Romeo et cliquer sur le bouton « Connexion ». Vous verrez alors les courbes se mettre à jour automatiquement et défiler des valeurs numériques dans la zone d'affichage à droite
- « Rafraîchissement »: ceci pilote la fréquence de rafraîchissement des courbes. En fonction de la vitesse de votre ordinateur, vous pouvez choisir parmi les 4 valeurs « Minimum », « Lent », « Moyen » et « Rapide ». Plus la fréquence de rafraîchissement est élevée, moins le tracé est saccadé. Mais si votre ordinateur n'est pas très rapide, vous risquez d'observer un retard entre les consignes et l'affichage. Dans ce cas, il faut choisir un rafraîchissement plus lent
- « Fréq. éch. (Hz) »: ce paramètre correspond à la fréquence d'échantillonnage des mesures dans le programme Arduino. Attention: ce paramètre doit être spécifié en Hz, alors que la valeur correspondante dans le programme Arduino (TSDATA) est spécifiée en ms. La relation entre les deux est la suivante:
  freq (Hz) = 1000 / TSDATA (ms)
- Bouton « Enregistrer »: il permet d'enregistrer les valeurs numériques affichées dans la zone de « log » vers un fichier texte pour une utilisation ultérieure avec n'importe quel logiciel permettant de lire des données séparées par des virgules dans une fichier texte.
  Attention : ce bouton n'est actif qu'après une séquence de mesure. Il est grisé le reste du temps (au démarrage du programme et pendant une séquence de mesure)
- Zone « Consigne »: elle permet de modifier la consigne de position avec les curseurs. La position du moteur varie alors. Vous pouvez la visualiser, ainsi que la consigne et la tension de commande, sur le graphique qui s'affiche en temps-réel
- Zone « Asservissement »: elle permet de modifier les gains du régulateur PID pendant le fonctionnement du système. Cela permet de voir l'influence de ces gains sur les performances de l'asservissement et de régler précisément ce dernier.

IMPORTANT !
Si l'affichage ne suit pas les consignes même avec une fréquence de rafraichissement très lente, veuillez diminuer la cadence d'échantillonnage dans le programme Arduino (augmenter la valeur TSDATA). Ne pas modifier la valeur CADENCE_MS, sous peine de déstabiliser l'asservissement.

7 - Identification des paramètres du moteur

Les systèmes livrés après le 1er décembre 2016 permettent une identification automatique des paramètres du moteur, grâce au capteur de mesure de courant intégré au système.

7.1 - Essais à réaliser sur le système

Rappelons ci-dessous les équations d'un moteur à courant continu :

$$J_m\, n^2 \frac{\mathrm{d}}{\mathrm{d}t}\omega_m\,(t) + d\omega_m\,(t) - nK i_m\,(t) = Tr\,(t) \tag{7.1}$$

$$L\frac{\mathrm{d}}{\mathrm{d}t} i_m\,(t) + R_m\, i_m\,(t) = V\,(t) - Kn\omega_m\,(t) \tag{7.2}$$

Avec :

- R_m : résistance électrique interne
- L : inductance des enroulements
- J_m : moment d'inertie du rotor
- K : constante de couple = constante de fem
- d : coefficient de frottement visqueux
- ω_m : vitesse de rotation de l'arbre de sortie du réducteur
- i_m : courant dans le moteur
- V : tension d'alimentation
- Tr : couple résistant
- n : rapport de réduction (100)

On est donc en présence de deux équations électro-mécaniques couplées.

Afin d'identifier les paramètres du moteur, nous effectuerons les deux essais suivants :

- Mesure du courant sur un échelon de tension à vitesse nulle
- Mesure de la vitesse de rotation sur un échelon de tension avec un couple résistant nul

7.1.1 - Mesure du courant sur un échelon de tension à vitesse nulle

Lorsque la vitesse est nulle, l'équation (7.2) s'écrit :

$$L\frac{\mathrm{d}}{\mathrm{d}t}i_m\left(t\right)+R_m\,i_m\left(t\right)=V\left(t\right) \tag{7.3}$$

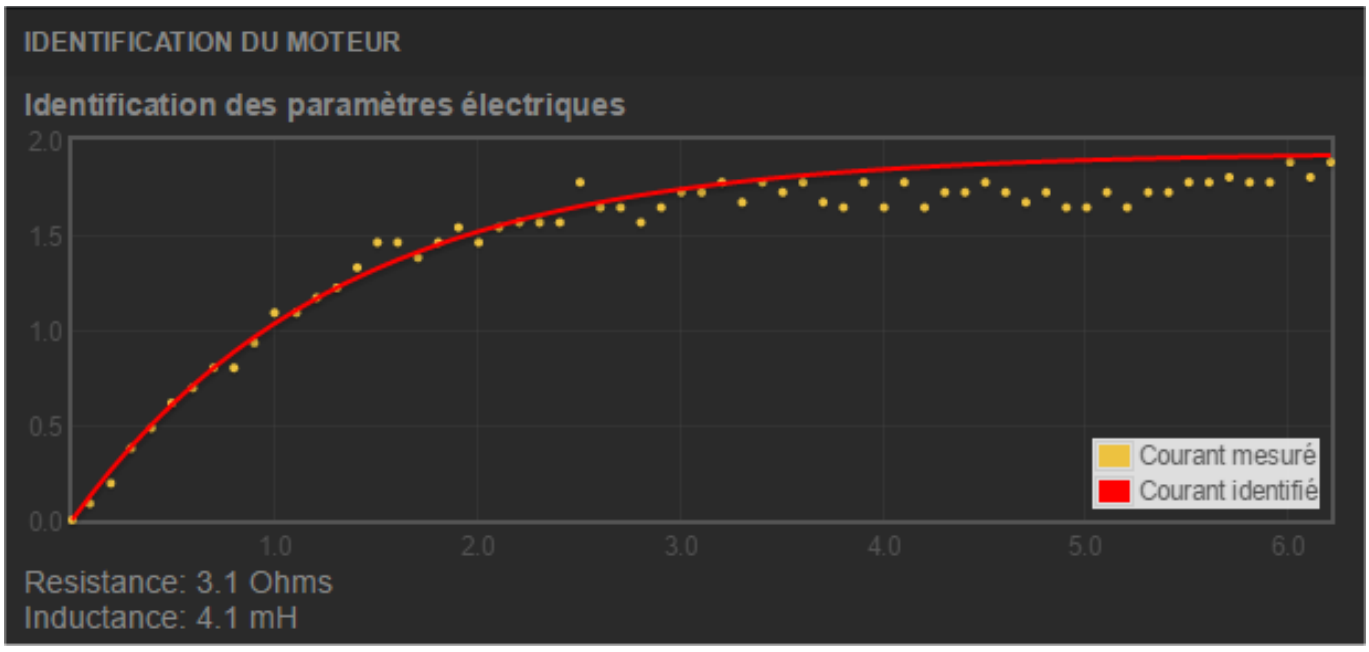
On peut donc facilement estimer l'inductance et la résistance à partir de la mesure du courant, puisqu'on est en présence d'un système du premier ordre.

Ce type d'essai se réalise traditionnellement en bloquant le rotor du moteur, ce qui présente deux inconvénients :

- ce n'est pas toujours très facile à réaliser d'un point de vue mécanique
- cela peut conduire à une surchauffe, voir à une destruction de certains composants si l'essai dure trop longtemps

La stratégie adoptée sur le système et mise en œuvre dans le programme Arduino d'identification des moteurs consiste à capturer le courant sur un échelon initial très court avec une tension de 6V, alors que le moteur n'a pas encore commencé à tourner. Afin de capturer cette très courte période avec la meilleure résolution possible, le temps de conversion des convertisseurs analogiques-numériques de l'Arduino est accélérée d'un facteur 16 dans ce programme, pour qu'on puisse faire une mesure toutes les 100 µs.

L'application d'identification qui s'exécute avec le logiciel MyViz (voir plus loin) permet ainsi d'obtenir ce type de capture :



Noter que l'unité de l'abscisse est la milliseconde.

7.1.2 - Mesure de la vitesse de rotation sur un échelon de tension avec un couple résistant nul

La mise en vitesse du moteur se fait avec une constante de temps beaucoup plus importante que la montée en courant que nous venons de voir. Pour faciliter les choses, nous allons par conséquent négliger l'inductance dans ce qui suit.

Si on annule en plus le couple résistant (le moteur tournant donc à vide), les équations (7.1) et (7.2) se combinent en :

$$J_m\, n^2 \left(\frac{\mathrm{d}}{\mathrm{d}t}\omega_m\left(t\right)\right) R_m + \left(K^2 n^2 + R_m\, d\right) \omega_m\left(t\right) = KV\left(t\right) n \qquad (7.4)$$

Nous retrouvons, comme précédemment, un premier ordre. Cependant, nous devons identifier ici 3 paramètres : J_m, K et d. Il faut donc ajouter une équation car la courbe de réponse d'un premier ordre ne permet d'identifier que 2 paramètres.

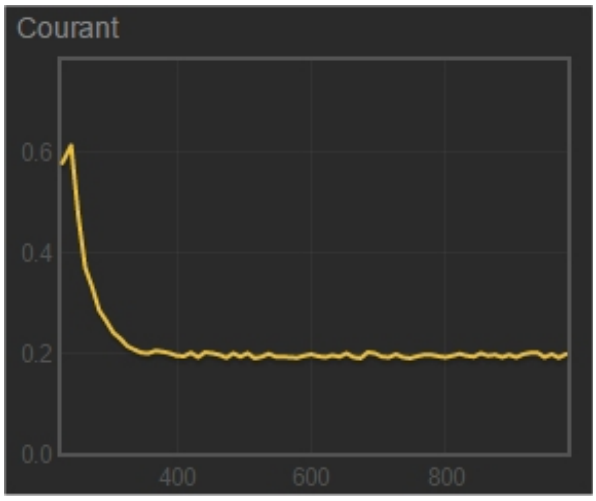
Cette équation est celle de la vitesse en régime permanent (lorsque l'accélération est nulle) issue de (7.1).

$$d\omega_m\left(t\right) - nKi_m\left(t\right) = 0 \qquad (7.5)$$

La mesure de la vitesse et du courant en régime permanent nous permettra donc de remonter à la relation entre la constante de couple et le coefficient de frottement.

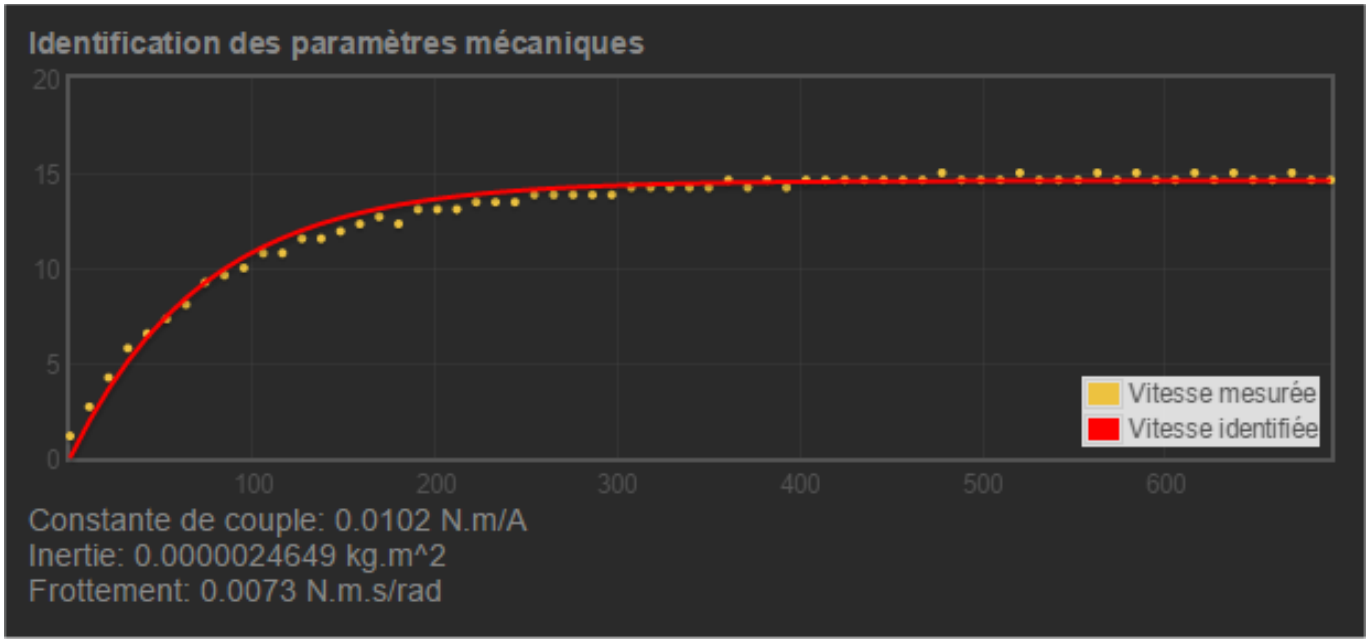
La stratégie adoptée sur le système et mise en œuvre dans le programme Arduino d'identification des moteurs consiste à calculer à la fin de chaque période de PWM une moyenne des courants mesurés pendant cette période écoulée. La fréquence des PWM étant de 500 Hz, ceci permet d'avoir une mesure toutes les 2 ms, réalisée sur la moyenne de 20 échantillons de courant (la cadence des mesures de courant étant de 100 µs).

Le profil de courant obtenu dans MyViz lors de l'essai en vitesse est le suivant :



Seule la valeur en régime permanent nous est utile, mais il est cependant intéressant de constater que cette méthode permet d'obtenir un tracé de très bonne qualité dans la phase transitoire

La vitesse est quant à elle mesurée d'une façon plus classique, à partir du comptage des interruptions générées par le codeur incrémental sur des intervalles de temps de 10 ms. L'identification se fait ensuite facilement à partir de cette mesure de vitesse :



Noter que la tension d'alimentation des moteurs est fixée ici à 6 V.

7.2 - Programme Arduino

Le programme Arduino mettant en œuvre la stratégie présentée plus haut peut être téléchargé à l'adresse suivante:

http://www.3sigma.fr/Telechargements-Ensemble_Commande_de_moteur_a_courant_continu.html

Son nom est de la forme Identification_x.y.zip (x.y correspond au numéro de version du programme).

Il est nécessaire de télécharger ce programme sur la carte Arduino pour réaliser cette procédure d'identification.

La procédure à suivre pour la programmation est la suivante:

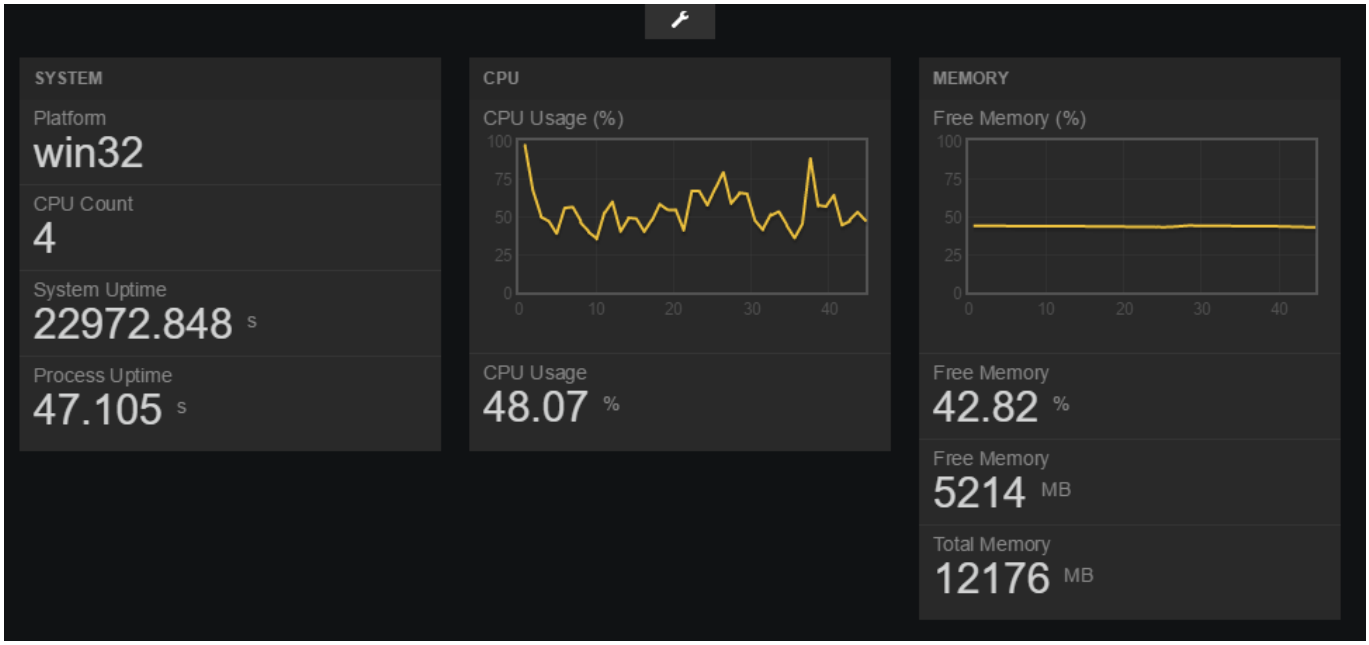
- Si ce n'est pas déjà fait, installez l'IDE Arduino et les bibliothèques additionnelles nécessaires (voir paragraphe 4)
- connecter l'alimentation 9V sur le connecteur jack du plateau supportant la carte Arduino et le moteur (voir paragraphe 5.1)
- mettre le système sous tension en positionnant le bouton marche-arrêt sur « I » (voir paragraphe 5.1)
- connecter la carte Arduino à l'ordinateur avec le câble USB fourni (voir paragraphe 5.1)
- Lancer le téléchargement

L'essai démarre tout de suite à la fin du téléchargement.

7.3 - Application MyViz d'identification automatique

Cette activité utilise le logiciel MyViz, très souple pour créer des tableaux de bord de pilotage et de visualisation de données.

Après l'avoir téléchargé (http://www.3sigma.fr/Telechargements-MyViz.html) et installé, lancez son exécution. Le tableau de bord initialement affiché sera similaire à la capture d'écran ci-dessous :



Ce tableau de bord n'est qu'un exemple de ce qui peut être réalisé avec MyViz. Nous verrons plus loin comment charger celui correspondant à l'expérience que nous souhaitons réaliser dans ce chapitre.

Pour réaliser la procédure d'identification automatique, les conditions suivantes doivent être remplies :

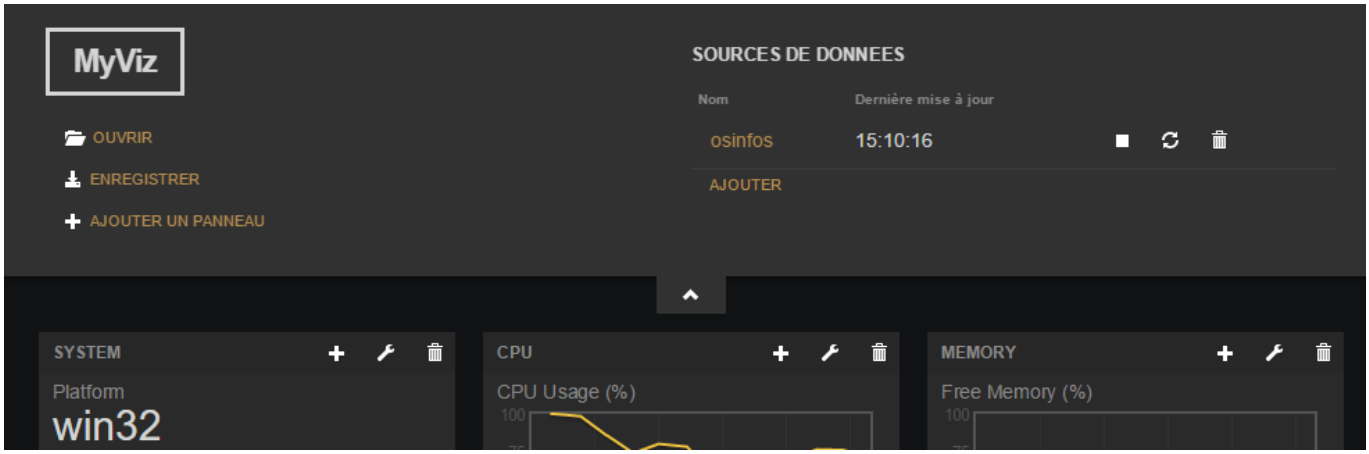
- le système doit être allumé
- le câble USB doit relier l'ordinateur et la carte Arduino
- le programme Arduino d'identification doit être chargé (voir chapitre 7.2)

Charger ensuite le tableau de bord d'identification automatique dans MyViz. Pour cela, il faut tout d'abord récupérer ce dernier sur votre ordinateur, à partir du lien suivant :

http://www.3sigma.fr/telechargements/Identification.json

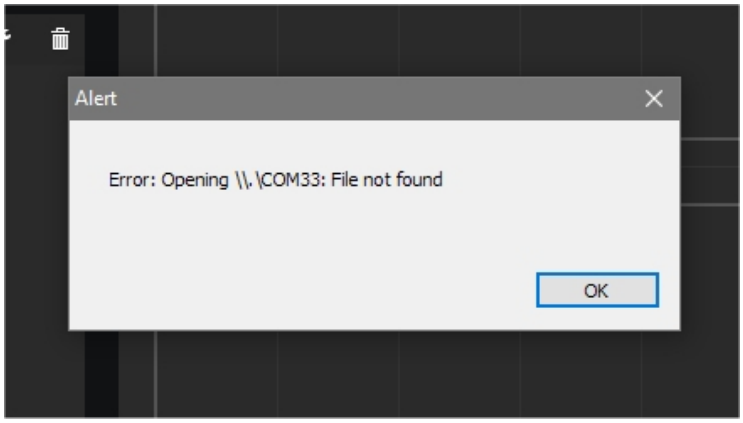
Pour l'ouvrir dans MyViz, il suffit ensuite de cliquer sur la clé en haut de la fenêtre :

Ceci permet de déplier le panneau supérieur :



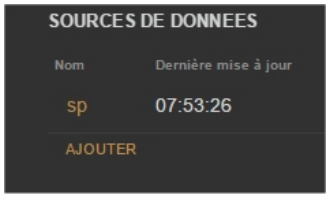
Cliquez sur « Ouvrir » et sélectionnez le fichier Identification.json que vous venez de télécharger.

Sauf hasard extraordinaire, vous devriez voir apparaître un message d'erreur de ce type :

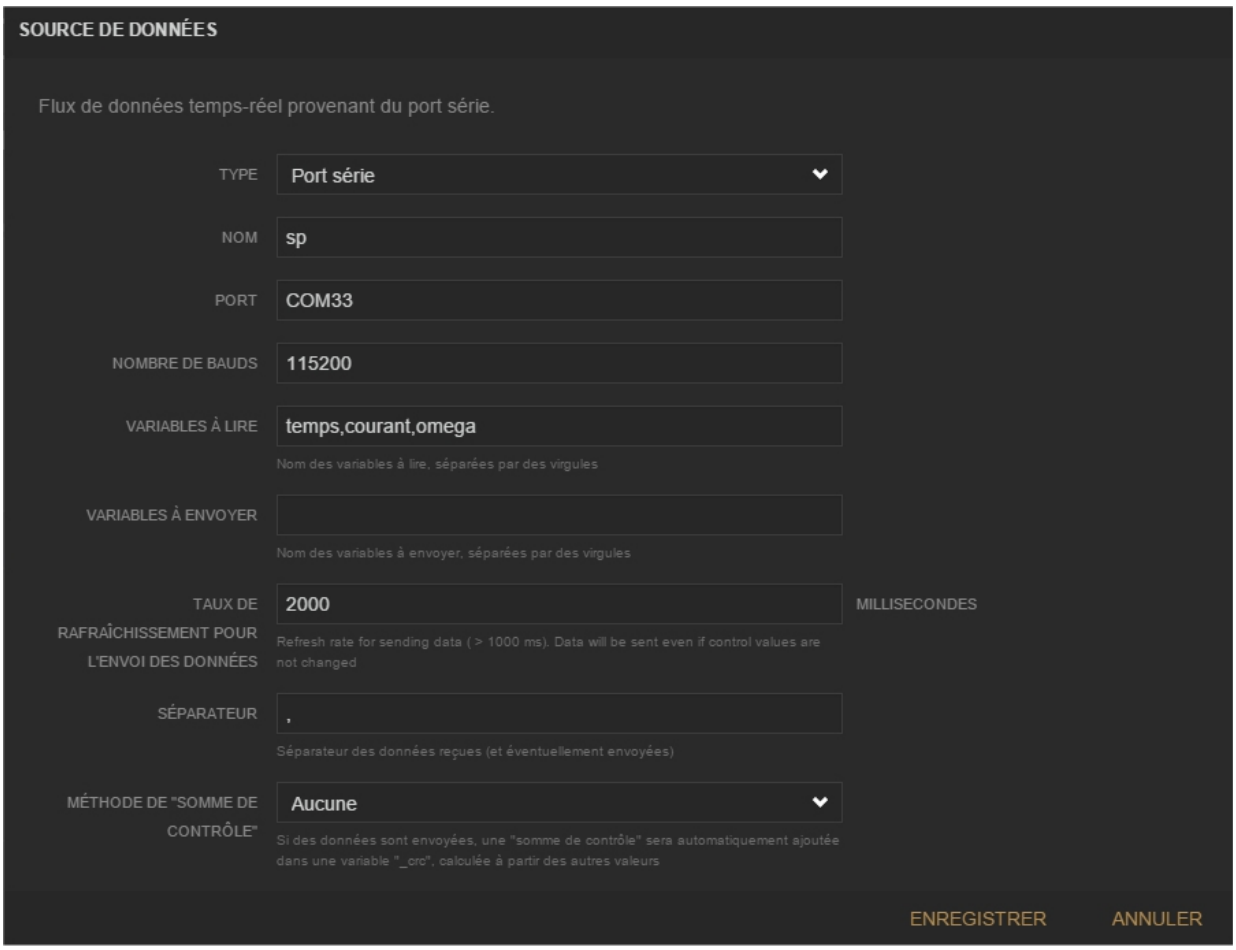


Ce message est normal, il indique simplement que le port série sur lequel votre système est branché n'est pas le même que celui enregistré dans le tableau de bord. Pour le changer, il suffit de suivre ces étapes :

- cliquer sur la clé pour déplier le panneau supérieur de MyViz
- dans la zone « Sources de données », cliquez sur « sp » :

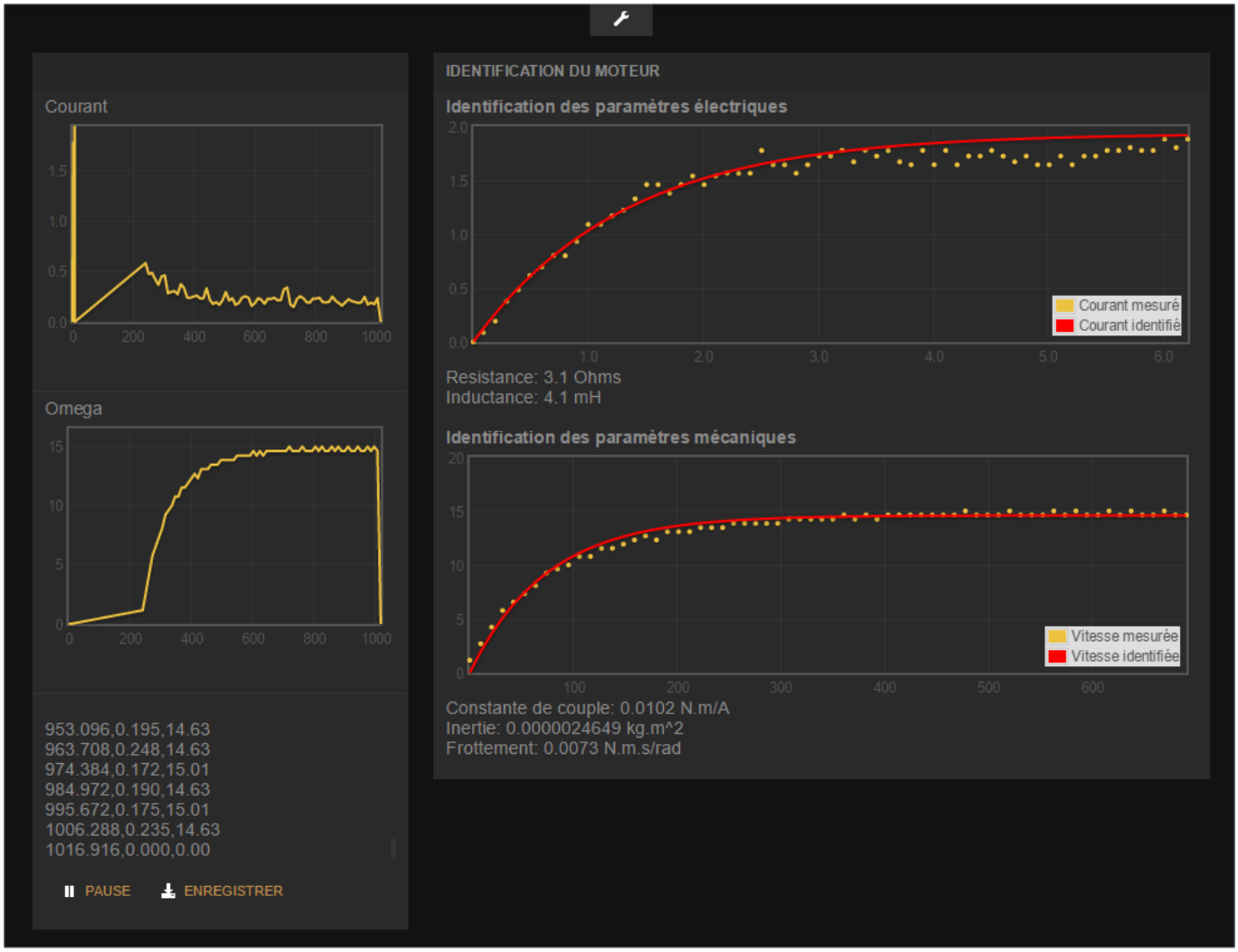


- il s'ouvre alors la fenêtre suivante, permettant le paramétrage de la communication série :



- dans la zone « Port », indiquez le port série sur lequel votre carte Arduino est connectée, puis cliquez sur le bouton « Enregistrer »
- vous allez entendre le moteur tourner brièvement et voir s'afficher dans MyViz le résultat de l'identification.

Si tout s'est passé normalement, vous devriez obtenir un résultat de ce type :



Si vous souhaitez faire une nouvelle identification (il arrive que les bruits de mesure rendent certains essais particulièrement mauvais), vous devez :

- couper la liaison série en cliquant sur le bouton « Stop » de la zone « Sources de données »

- puis cliquer sur le bouton de rechargement à côté :

Remarques :

- à condition d'avoir replié le panneau supérieur de MyViz, il est possible de zoomer avec la souris dans les tracés de la colonne de gauche. Pour dézoomer, double-cliquer dans la fenêtre
- vous pouvez recommencer la procédure d'identification autant que vous le souhaitez. Vous devez pour cela, à chaque fois, cliquer sur le bouton d'arrêt puis sur le bouton de rechargement de la liaison série (sp) dans la zone des sources de données
- si vous souhaitez charger un autre programme Arduino, il faut au préalable arrêter la communication série dans MyViz en cliquant sur le bouton d'arrêt

8 - Important

Cet ensemble est un produit « vivant » en constant développement pour améliorer ou lui ajouter de nouvelles fonctionnalités. Si vous avez des idées ou des besoins pour des développements spécifiques, n'hésitez pas à nous contacter (info@3sigma.fr).

Ne restez jamais bloqué sans nous contacter !
Pour tout problème ou toute requête, contactez-nous à l'adresse support@3sigma.fr.