Banc de puissance à inertie

Institut Gériatrique des Solex
Bernique

Abstract

Tracer les courbes de couple et de puissance d'un cyclomoteur en fonction de son régime moteur.

Keywords: Solex, banc dynamique, Arduino, TCRT5000, trigger Schmitt, 74HC14, puissance, couple

---

Ce document, tant dans son contenu technique que sa rédaction, est le fruit d'un travail collaboratif entre passionnés d'électronique et de cyclomoteurs à Galet.
Merci à Jean-Noël S. et Nico_32 pour leur temps, leur savoir et leur plume à l'écriture de cette synthèse qui, je l'espère, vous donnera à vous aussi des envies de réalisations.

Table des matières

Banc de puissance à inertie ... 1
Abstract ... 1
I. Objectif ... 3
II. Principe du banc et équations générales Puissance et Couple ... 3
III. Calcul pratique vitesse et accélération angulaires par discrétisation ... 5
IV. Quelques éléments de dimensionnement ... 5
1 Hardware d'acquisition ... 5
2 Caractéristiques mécaniques du banc ... 6
V. Design du code de traitement de l'Arduino ... 7
1 Architecture et bases du code ... 7
2 Fonctions de la routine d'interruption ... 7
3 Fonctions de la boucle principale ... 7
4 Configuration de la macro PLX-DAQ ... 8
5 Initialisation de l'acquisition ... 8
6 Test de la programmation par simulation du capteur ... 8
VI. Conditionnement du signal en amont de l'Arduino ... 9
1 Choix du capteur ... 9
1.1. Option 1 : capteur inductif à effet Hall RCEXL ... 9
1.2. Option 2 : capteur optique OMRON 1041 ... 9
1.2.1. Mise en œuvre ... 9
1.2.2. Test d'acquisition ... 10
1.3. Option 3 : capteur optique TCRT5000 ... 10
1.3.1. Mise en œuvre ... 10
1.3.2. Test d'acquisition ... 12
2 S'affranchir des parasites sur l'entrée de l'Arduino ... 12
VII. De la théorie à la pratique : inertie, frottements du banc & glissement ... 14
1 Prise en compte de l'inertie réelle et des frottements ... 14
2 Prise en compte du glissement Galet|pneu|rouleau ... 16
VIII. Mises en œuvre pratiques ... 17
1 Exemple d'intégration sur banc maison (Nico_32) ... 17
2 Exemple d'intégration sur banc Marolo (Bernique) ... 17
3 Exemples d'exploitation des courbes obtenues ... 19
3.1. Impacts de trois réglages sur un bloc moteur (Nico-32) ... 19
3.2. Comparaison des performances entre moteurs (Bernique) ... 19
IX. Répétabilité des mesures ... 20
Bibliographie ... 21
Annexe : le code ... 22

I. Objectif

Tracer les courbes de couple et de puissance d'un cyclomoteur en fonction de son régime moteur, exprimé en RPM (Revolutions Per Minute = tours par minute).

Le principe retenu est la construction d'un banc à inertie sur lequel est installée la machine à tester. Le banc est instrumenté afin d'effectuer les acquisitions et les calculs en temps réels sur un PC connecté en USB.
Le présent document traite le cas particulier d'un Solex, mais le principe reste adaptable à d'autres types de motorisations (sous réserve d'adaptations dans les formules décrivant les comportements physiques de la liaison moteur/banc).

II. Principe du banc et équations générales Puissance et Couple

Le moteur, via la roue, entraîne en rotation un rouleau de moment d'inertie J[1] et de diamètre $D_{rouleau}$. Un capteur fixé sur le bâti du banc permet de relever avec précision la durée de rotation complète ou partielle du rouleau, donnant accès à l'information de vitesse angulaire $\omega_{rouleau}$.

Le couple d'entraînement du rouleau est déterminé par :

$$C_{rouleau} = J \frac{d\,\omega_{rouleau}}{dt} \quad \text{en m.N} \quad (2) \qquad \text{(R0)}$$

La puissance au niveau du rouleau est déterminée par :

$$P_{rouleau} = C_{rouleau} . \omega_{rouleau} \quad \text{en watt} \qquad \text{(R1)}$$

Ces deux équations n'incluent pas les frottements et autres inerties parasites qui seront traités plus tard.

1 si la réalisation pratique fait appel à deux rouleaux cylindriques pour des raisons de stabilité par exemple, il faudra considérer que le moment d'inertie J de l'ensemble est la somme des deux moments d'inertie de chacun des rouleaux. Plus généralement, il faudra considérer l'inertie des masses tournantes du banc, dont celle de la roue !.

2 cette formule théorique ne fait pas intervenir les frottements du banc. Elle est d'autant plus réaliste que l'inertie du banc sera importante. Le cas réel de la prise en compte des frottements est traité au §VII.

Creative Commons 3.0 by Bernique

Puisque l'on veut tracer la puissance et le couple du moteur en fonction de sa vitesse angulaire exprimée en RPM, il faut ramener ces grandeurs au niveau du moteur.
Il y a conservation de la puissance transmise dans la chaîne moteur/galet/roue/rouleau (aux glissements et échauffements du pneu près), ce qui s'écrit :

$$P_{moteur} = P_{rouleau} \quad \text{(R2)}$$

soit $$C_{moteur} \cdot \omega_{moteur} = C_{rouleau} \cdot \omega_{rouleau} \quad \text{(R3)}$$

Dans le cas d'un Solex et en l'absence de glissement, on a aussi conservation de la vitesse linéaire V au point de contact entre Galet et roue[3], mais aussi entre roue et rouleau, ce qui se traduit par :

$$V_{galet} = V_{roue} = V_{rouleau} \text{ , d'où } \omega_{galet} \cdot D_{galet} = \omega_{rouleau} \cdot D_{rouleau} \quad \text{(R4)}$$

De plus, moteur en fonctionnement (embrayage non glissant), le vilebrequin est en prise directe sur le Galet, ce qui se traduit par : $\omega_{galet} = \omega_{moteur}$.

De (R4) on peut déduire: $$\frac{\omega_{rouleau}}{\omega_{moteur}} = \frac{D_{galet}}{D_{rouleau}} \quad \text{(R5)}$$

d'où : $$\omega_{moteur} = \omega_{rouleau} \cdot \frac{D_{rouleau}}{D_{galet}} \quad \text{(R6)}$$

Par substitution de (R5) et (R0) dans (R3), on a : $$C_{moteur} = \frac{D_{galet}}{D_{rouleau}} \cdot J \frac{d\,\omega_{rouleau}}{dt} \text{ en m.N} \quad \text{(R7)}$$

Par substitution de (R0) et (R1) dans (R2), on a : $$P_{moteur} = J \frac{d\,\omega_{rouleau}}{dt} \cdot \omega_{rouleau} \text{ en watt} \quad \text{(R8)}$$

Ces trois équations (R6, R7 et R8) seront à la base de notre programmation pour tracer nos courbes de puissance et de couple en fonction du régime moteur.

En terme d'unité de restitution, on se servira de la notion de RPM pour la vitesse angulaire, avec pour conversion la relation $\omega_{RPM} = \frac{60}{2\pi} \omega_{rad/s}$.

De même, la puissance sera restituée en cv, avec la conversion $P_{cv} = \frac{P_{watt}}{736}$.

3 à adapter ici en fonction du type de machine, cette équation n'étant valable que pour un moteur à Galet

Creative Commons 3.0 by Bernique

III. Calcul pratique vitesse et accélération angulaires par discrétisation

Les équations de base déterminées au paragraphe précédent sont fonction de la vitesse et de accélération angulaires du rouleau (respectivement $\omega_{rouleau}$ et $\frac{d\omega_{rouleau}}{dt}$). Ces deux valeurs sont calculées à partir des temps mesurés entre deux impulsions du capteur du système d'acquisition situé sur le rouleau du banc. Un Arduino effectue la mesure à l'aide de la fonction micros().

Soit t_n la valeur de temps d'un point de mesure obtenu par la fonction micros() et p le nombre d'impulsions par tour. La vitesse angulaire du rouleau est déterminée par :

$$\omega_n = \frac{1}{p} \cdot \frac{2\pi}{(t_n - t_{n-1})} \quad \text{en rad.s}^{-1} \qquad \text{(R9)}$$

L'accélération angulaire est défini par :

$$\frac{d\omega_n}{dt} = \frac{\omega_n - \omega_{n-1}}{t_n - t_{n-1}} \quad \text{en rad.s}^{-2} \qquad \text{(R10)}$$

On a désormais un puzzle complet avec le quinté gagnant formé par les équations R6, R7, R8, R9 et R10 qu'il faut maintenant passer en code pour tracer nos courbes.

IV. Quelques éléments de dimensionnement

1 Hardware d'acquisition

Il est indispensable que le matériel d'acquisition utilisé soit capable de recevoir et de traiter en temps réel l'ensemble des impulsions envoyées par le capteur du rouleau du banc.

Prenons le cas extrême d'un régime moteur de Solex de type VSX3800 calé à 12000 RPM (diamètre de Galet de 42mm). Avec un rouleau de diamètre 170mm, on obtient une vitesse de rotation maximum du rouleau de 3000 RPM, soit 50 tr/s. Dans l'hypothèse où le capteur génère 2 impulsions par tour, cela nous donne une fréquence maximum d'impulsions de 100 Hz, soit un écart entre impulsions de 10 ms.
Ce temps est tout à fait compatible avec une acquisition par Arduino, même s'il n'autorise pas beaucoup de calculs, surtout en flottant. Une autre raison pour bannir les calculs en flottant est la possible variation du temps de calcul.
Un Arduino Uno se trouve sur le Net pour 15 à 20€ en version genuine, 3 à 5€ en copie chinoise.

Présentation d'un Arduino UNO (à droite, enfin je crois !)

2 Caractéristiques mécaniques du banc

Si l'on veut réaliser des acquisitions représentatives de la vie du moteur en conditions réelles, il faut que le banc ait une dynamique d'accélération proche de celle rencontrée par la machine lorsqu'elle évolue sur route ou sur piste.

Pour comparer les deux alternatives (vraie vie versus banc), on regarde la variation d'énergie nécessaire pour accélérer le Solex initialement à l'arrêt dans les deux cas d'usage :

- passer de 0 à "V" km/h avec un pilote + Solex pesant 100kg tous mouillés,
- passer de 0 à "V" km/h sur le banc.

… ce qui revient à écrire, l'énergie initiale étant nulle dans les deux cas :

$$E_{Translation} = E_{Rotation}$$

avec : $E_{Translation} = \frac{1}{2} m v^2$, où m est la masse (machine + pilote) et v la vitesse.[4]

$E_{Rotation} = \frac{1}{2} J \omega^2$, où J est l'inertie du rouleau du banc et ω sa vitesse de rotation.

L'égalité des énergies se réécrit : $J_{rouleau} \omega_{rouleau}^2 = m_{solex+pilote} v_{solex}^2$

En utilisant (R6) dans le 1er terme, i.e. : $\omega_{rouleau} = \omega_{moteur} . \frac{D_{galet}}{D_{rouleau}}$

et en utilisant (R4) dans le 2nd terme, i.e. : $v_{solex} = \omega_{moteur} . \frac{D_{galet}}{2}$

l'égalité des énergies devient après simplification : $4. J_{rouleau} = m_{solex+pilote} . D_{rouleau}^2$ (R11)

Or, pour un cylindre plein, on a : $J_{rouleau} = \frac{1}{2} . \rho . \pi . h . \left(\frac{D_{rouleau}}{2} \right)^4$ (ρ = masse volumique ; h=hauteur du cylindre)

Injecté dans (R11), on obtient : $4. \frac{1}{2} . \rho . \pi . h . \left(\frac{D_{rouleau}}{2} \right)^4 = m_{solex+pilote} . D_{rouleau}^2$

soit : $h = \frac{2^3 . m_{solex+pilote}}{\rho . \pi . D_{rouleau}^2}$ (R12)

La suite de l'histoire consiste a choisir une valeur de $D_{rouleau}$, et d'en déduire h afin que le couple soit aisé à mettre physiquement en œuvre.[(5)]

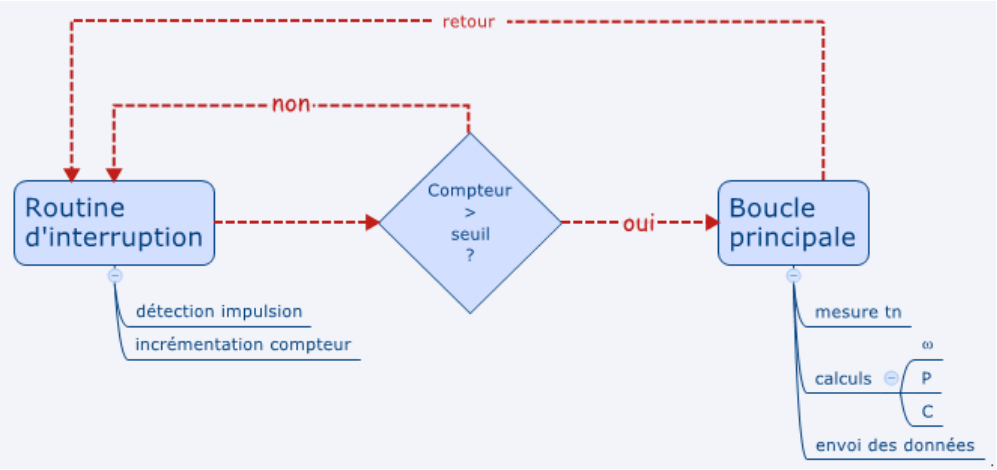
4 dans cette représentation, on simplifie le représentation de l'énergie du Solex qui est une combinaison de solides en translation et rotation (les roues par exemple!).

5 Prenons un équipage pilote-machine de 100kg. En choisissant un rouleau cylindrique en acier (7800kg/m³) de diamètre 170mm, il faut une hauteur de 1,13 mètres pour assurer la fonction d'inertie, ou plus pratiquement deux rouleaux de 56cm sur lesquelles reposera la roue du Solex !

Creative Commons 3.0 by Bernique

V. Design du code de traitement de l'Arduino

1 Architecture et bases du code



La base du code repose sur :
https://www.arduino.cc/en/Reference/AttachInterrupt
et
http://playground.arduino.cc/Code/Interrupts
Avec un Arduino Uno, il faut brancher les impulsions capteur sur l'entrée numérique D2, et avec un Leonardo sur l'entrée D3. Dans ce cas, c'est l'interrupt 0 qui sera utilisée.

2 Fonctions de la routine d'interruption

Elle effectue une simple incrémentation d'un compteur pour comptabiliser le nombre d'impulsions reçues. La boucle principale Arduino déclenche ensuite le calcul et le traitement d'envoi.
En séparant ainsi le code, on raccourcit au maximum la routine d'interruption, réduite à sa plus simple expression, et son temps d'exécution est de quelques microsecondes. L'erreur générée par ce temps de gestion non comptabilisé par micros() s'annule dans le calcul de ω_n (fonction de la différence entre t_n et t_{n+1}).

3 Fonctions de la boucle principale

C'est dans cette boucle que se font les traitements de mise en forme de la donnée. L'essentiel du temps est passé dans la transmission de données, qu'il est possible de raccourcir en réduisant le nombre de colonnes (quatre minimum, si calculs inclus), et en réduisant la précision des nombres flottants.
A noter, l'usage de la macro induit une certaine dé-optimisation du temps de transfert par rapport à un transfert binaire des données, mais c'est incontournable, car la macro PLX-DAQ demande des données textes.
Le passage en calcul en virgule fixe induit une gestion des débordements pénible. Comme le temps de calcul est faible (0,3 ms) devant le temps de transmission (6 ms), pas la peine de s'embêter, on fait les calculs en flottant, et le tout est bien plus lisible.

Creative Commons 3.0 by Bernique

4 Configuration de la macro PLX-DAQ

Cette macro (https://www.parallax.com/downloads/plx-daq) est utilisée pour rapatrier dans Excel les résultats de mesure. Elle ne fonctionne que sous Windows[6].

Si des traitements sur les données sont requis, il faut les faire dans des colonnes situées après les quatre colonnes servant à rapatrier la donnée de mesure. Des graphiques peuvent être créés à partir de ces colonnes calculées.

N° de port série de l'Arduino :

Le N° de port série est attribué automatiquement par le PC, en fonction des périphériques USB/série qui ont déjà auparavant été branchés sur le PC. La macro PLX-DAQ ne sait traiter que les ports COM1 … COM15.

Si le N° affecté est dans cette gamme 0 … 15, pas de problème, c'est le N° qu'on utilise dans la configuration de la macro PLX-DAQ. Sinon, il faut changer ce N° de port. Pour ce faire :

- Aller dans Panneau de configuration / Système / Gestionnaire de périphériques / Ports (COM et LPT)
- Double cliquer sur la ligne correspondant à l'Arduino
- Cliquer sur l'onglet "Paramètres du port", puis sur "Avancé"
- Cliquer sur "Numéro de port COM", et choisir un numéro, même s'il est non disponible. Ne pas prendre les numéros de 1 et 2 qui peuvent correspondre à des ports série physiques

Vitesse du port série :

Le code étant configuré pour une vitesse série de 128000 bauds, choisir cette valeur dans l'ActiveX de contrôle de la macro, sinon, pas de communication possible[7].

5 Initialisation de l'acquisition

Elle se fait sur une boucle d'attente pour pouvoir disposer de données "propres" dans la feuille de calcul. La séquence de démarrage est la suivante :

- Mise sous tension de l'Arduino par branchement sur port USB du PC ; l'Arduino se met dans une boucle d'attente (pas d'acquisition à ce stade).
- Démarrage d'Excel sur la feuille contenant la macro PLX-DAQ ; le N° de port doit correspondre à celui de l'Arduino, et la vitesse doit être à 128000 bauds (115200 sous Libre Office). Connecter la macro à l'Arduino via la bouton "Connect".
- Démarrer les acquisitions sur l'Arduino (relier temporairement masse et Digital N°4 de l'Arduino, ou presser le bouton S1 du ProtoShield si présent) : les données commencent à arriver dans la feuille Excel, et le graphe s'alimente en direct.

6 Test de la programmation par simulation du capteur

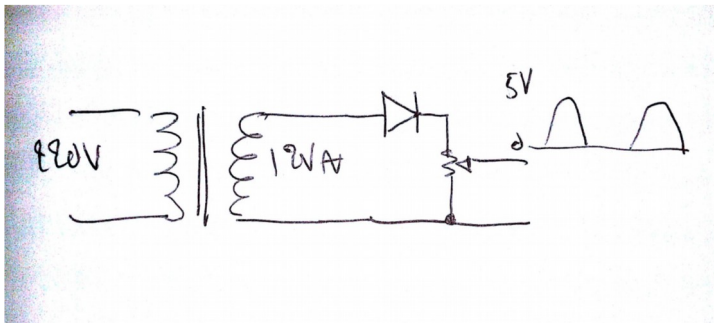
Pour faire le test sans le capteur, il est intéressant de simuler le signal. La solution la plus simple consiste à redresser en simple alternance un signal 50 Hz secteur, qui simule les impulsions obtenues à 1500 tr/mn (deux impulsions par tour). Le potentiomètre sert à ajuster le signal dans la gamme 0-5V.

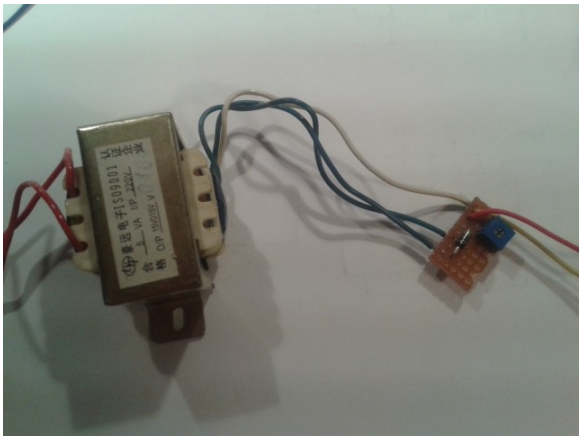
6 il existe également une version clone développées sous Libre Office et qui fonctionne indifféremment sous Windows ou Linux. Non testée ici, elle est disponible sur http://p.loussouarn.free.fr/arduino/arduino.html - OpenDaqCalc.

7 sous Libre Office, la vitesse est limitée à 115200 bauds ; il faudra ajuster le code de l'Arduino en conséquence

Creative Commons 3.0 by Bernique

Cette simulation est importante pour vérifier le fonctionnement de la routine d'interruption. Le reste du code est simple. Ce montage simule une vitesse de rotation fixe, le reste des tests devra donc se dérouler avec le capteur, sur le banc de test réel.





VI. Conditionnement du signal en amont de l'Arduino

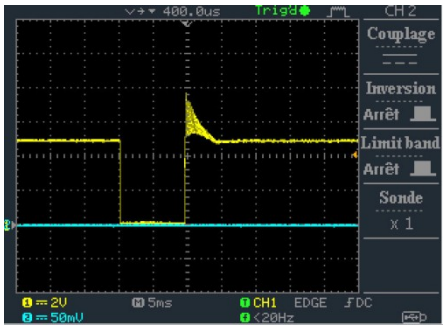
Pour ce qui est du conditionnement des signaux, l'Arduino est alimenté en 5V. Il nous faut donc des impulsions capteur conditionnées en 0-5V.

1 Choix du capteur

1.1. Option 1 : capteur inductif à effet Hall RCEXL

Il s'agit d'un capteur inductif alimenté en 5V et délivrant un signal 0-5V (pulses à 0 partant d'une base de signal à 5V).

Solution abandonnée pour cause d'interférences à priori électromagnétiques sur le capteur.

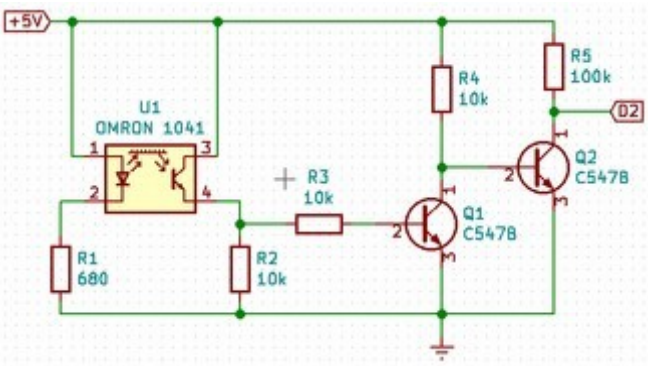


1.2. Option 2 : capteur optique OMRON 1041

1.2.1. Mise en œuvre

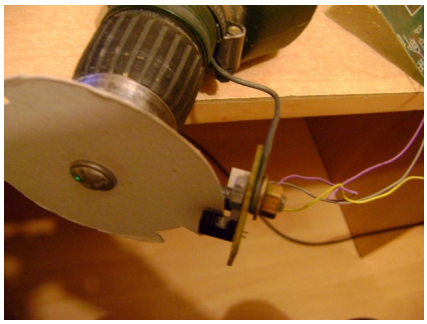
C'est un capteur en U composé d'une diode d'un côté et d'un phototransistor de l'autre. Il est alimenté avec le 5V de la carte Arduino. Son signal est de type carré.

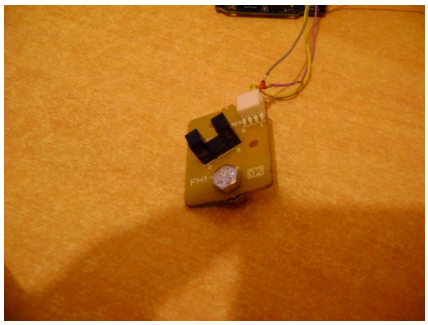
Le signal est remis "propre" par un montage à 2 transistors agissant comme un trigger de Schmitt 0 – 5V.
Consommation globale du circuit : 13mA.



1.2.2. Test d'acquisition

Un disque à 2 fentes est entraîné par le mandrin d'une perceuse pour les essais (accélération, stabilisation puis décélération).





T(s)	RPM	C(m.daN)	P(CV)
20:15:05	**102,67**	**0,033118**	**0,004838**
13:12:22	849,32	1,99224	2,407499
05:47:14	868,46	0,052216	0,064521
22:07:41	881,23	0,03534	0,044311
14:12:28	895,53	0,040252	0,051288
04:20:03	1019,36	0,396543	0,575134
17:08:50	1123,86	0,368938	0,58995
04:31:49	1265,04	0,561107	1,009954
14:41:08	1417,99	0,681352	1,374662
23:35:49	1615,91	1,004753	2,310084
07:37:10	1794,95	1,009619	2,578466
15:16:44	1879,98	0,502158	1,34321
22:46:01	1923,10	0,260524	0,712854
06:10:33	1943,63	0,125372	0,346709
13:29:42	1967,41	0,14696	0,411382
20:46:47	1976,73	0,057888	0,162813
04:02:53	1981,24	0,028052	0,079076
11:17:53	1986,19	0,03091	0,087352
18:32:24	1988,40	0,013803	0,039049
01:46:18	1991,24	0,017795	0,050416
08:59:41	1993,62	0,014859	0,042149
16:12:21	1996,96	0,020961	0,059557
23:24:36	1998,84	0,011792	0,033538
06:39:57	1984,59	-0,08884	-0,25087
14:24:25	1860,19	-0,72697	-1,92409
22:53:20	1697,71	-0,86656	-2,09323
08:31:20	1494,81	-0,95287	-2,02661
19:40:13	1291,72	-0,82413	-1,51466
10:37:15	963,17	-0,99415	-1,36241

1.3. Option 3 : capteur optique TCRT5000

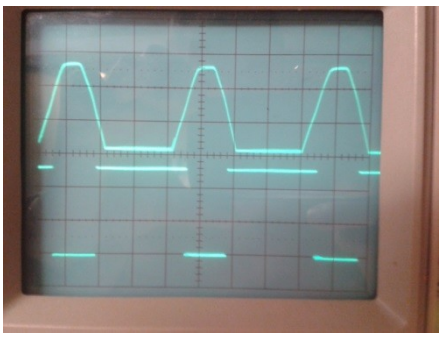
1.3.1. Mise en œuvre

Il s'agit d'un capteur de proximité composé d'une diode d'un côté et d'un phototransistor de l'autre, qui présente l'avantage d'une bonne portée de détection en distance (20mm en conditions optimales) et qui possède des filtres permettant de s'affranchir de la lumière du jour.

Il est alimenté en 5V et délivre un signal 0-5V.
Il nécessite une surface de réflexion adaptée (alternance clair/foncé).
Disponible sur eBay pour 0,20€

Le signal du capteur est relié par 2 diodes Schottky 1N6263 au +5V et à la masse, de manière à garantir qu'il reste dans une limite de 0,3V au-dessus du +5V ou en-dessous de la masse (protection de l'Arduino).
Il passe ensuite dans un 74HC14 (Trigger de Schmitt) qui nettoie les fronts montants et descendants et inverse le signal.

Ci-contre le résultat, à partir d'un signal de test constitué par des demi-alternances à 50 Hz.

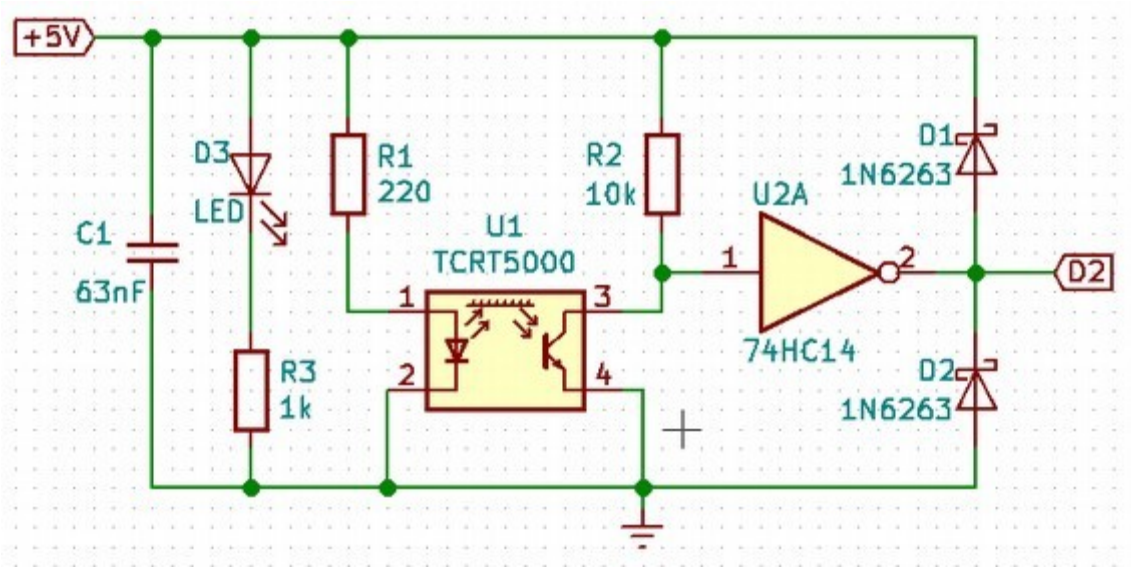


En vertical, 2V/division ;
En horizontal, 5 ms/division

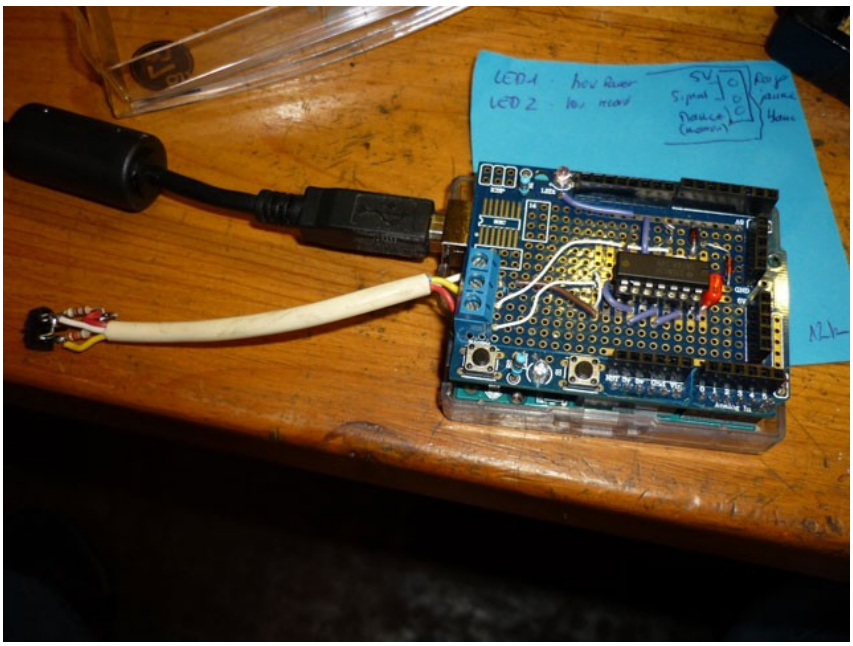
On utilise 2 inverseurs du 74HC14, l'un relié à l'IO D2, l'autre à D3, car selon les Arduino, l'IRQ0 n'est pas reliée à la même IO digitale.
Un condensateur de 63nF déparasite l'alimentation du capteur (situé entre 5V et masse).

L'électronique de conditionnement est câblée sur un Protoshield Arduino (façon Adafruit), que l'on peut acheter sur eBay pour moins de 5€. Un connecteur +5V|signal capteur|GND est fourni pour le raccordement du capteur.

Schéma du câblage à mettre en place sur le ProtoShield :



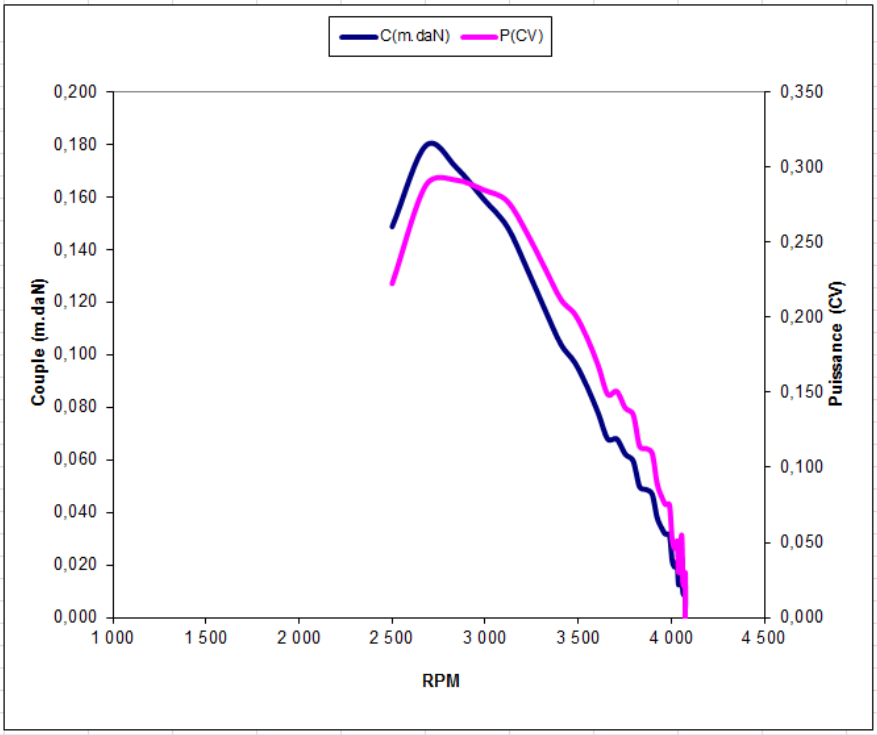
Le protoshield assemblé :



1.3.2. Test d'acquisition

Effectué sur un moteur de type 3800 refait à neuf, posé sur un banc Marolo instrumenté dont le diamètre de rouleau de roulage est de 99mm environ[8].

T(s)	RPM	C(m.daN)	P(CV)
29,089	2 501	0,149	0,222
29,355	2 682	0,180	0,288
29,606	2 845	0,171	0,291
29,845	2 990	0,159	0,285
30,074	3 119	0,149	0,277
30,295	3 229	0,132	0,255
30,510	3 325	0,117	0,232
30,720	3 407	0,104	0,212
30,925	3 483	0,097	0,202
31,126	3 549	0,088	0,186
31,324	3 608	0,078	0,168
31,519	3 658	0,068	0,149
31,711	3 708	0,068	0,151
31,902	3 753	0,062	0,140
32,090	3 795	0,060	0,135
32,276	3 830	0,050	0,114
32,461	3 865	0,049	0,113
32,644	3 897	0,047	0,110
32,826	3 924	0,038	0,090
33,007	3 947	0,034	0,081
33,187	3 969	0,032	0,075
33,366	3 990	0,032	0,075
33,544	4 005	0,022	0,052
33,722	4 018	0,019	0,046
33,899	4 032	0,021	0,051
34,076	4 040	0,012	0,030
34,252	4 056	0,023	0,055
34,427	4 062	0,010	0,024
34,603	4 068	0,009	0,021
34,778	4 076	0,012	0,029
34,953	4 075	-0,001	-0,003



2 S'affranchir des parasites sur l'entrée de l'Arduino

Entrées flottantes CMOS

Dans le cas de l'utilisation d'un composant de type CMOS comme le 74HC14, les entrées non utilisées flottantes présentent un fort risque d'interférences. Il sera préférable de les relier systématiquement au 5V.

La même logique pourra être appliquée aux portes logiques IO inutilisées en sorties de l'Arduino afin de les charger et éviter les potentielles informations erratiques et polluantes. Pour ce faire, le recours à la fonction pinMode (pin,INPUT_PULLUP) sera d'un précieux secours.

Exemple vécu sur le développement de ce module : les résistances R1 et R2 avaient été placées en aval du capteur TCRT5000 afin de pouvoir être intégrées sur la platine Protoshield et gagner ainsi de la place côté capteur… ce qui s'est révélé être une fausse bonne idée car la ligne de signal se comportait, dans le cas d'un capteur non passant, comme une entrée flottante générant des parasites à tout va ! L'intégration des résistances en amont et proches du capteur a résolue le problème !

Torsade

Si le câble (entre capteur et Arduino) traverse la zone perturbée par les champs électromagnétiques produits par un téléviseur, un GSM, un aspirateur, votre belle-mère, etc... Chacun de ses conducteurs verra naître entre ses extrémités une tension induite (surtout la belle-mère !). Les champs n'étant jamais homogènes, deux conducteurs séparés par une faible distance vont produire une tension différente. A l'extrémité du câble, à l'entrée de l'appareil utilisateur, on recueille la différence de ces tensions, mélangée

8 paramètre d'inertie J chargé dans l'Arduino non réaliste → valeurs de sortie fantaisistes !

au signal utile. En torsadant le câble, on présente alternativement chaque conducteur de façon symétrique au champ perturbateur et les tensions produites sur chaque conducteur sont égales. A l'entrée côté utilisation, les deux tensions perturbatrices égales se neutralisent et on reçoit le signal utile pur. L'effet sera d'autant meilleur que la torsade sera serrée : pour info, le câble réseau est torsadé bien plus vigoureusement que les paires téléphoniques.

Blindage

Le blindage doit impérativement être relié à la terre dans des conditions précises et contraignantes, faute de quoi le remède est pire que le mal (dans notre cas, à la masse de l'Arduino exclusivement).
Dans les cas de signaux faibles (lignes longues), il y a mieux à faire : utiliser des câbles torsadés de qualité, limiter le nombre des contacts, raccourcir les longueurs de câbles, centraliser les masses en étoile.

Implantation physique de l'électronique d'acquisition

Enfin et pour finir sur le sujet, le positionnement de l'électronique d'acquisition (Arduino, Protoshield et capteur) peut s'avérer décisif. En effet, les parasites émis par l'allumage de la machine (liés à la génération d'impulsions de plusieurs milliers de volts au secondaire) rayonnent de façon difficilement quantifiable, tant en valeur qu'en positionnement dans l'espace.

Creative Commons 3.0 by Bernique

VII. De la théorie à la pratique : inertie, frottements du banc & glissement

Les équations présentées dans les paragraphes précédents décrivent des systèmes théoriques parfaits et pour lesquels l'inertie J est connue avec précision. En y réfléchissant bien, ce n'est pas aussi évident dans la pratique avec toutes ces pièces en rotation qui sont autant de frottements, sans parler de leurs inerties complexes purement shuntées jusqu'à présent (comme la roue avant du Solex qui tourne par exemple) !

A partir de là, vous avez le choix :

- vouloir obtenir un système qui fait des mesures absolues… pas d'autre choix que de plancher comme une bête pour décrire et introduire les notions de frottements et autres perturbateurs afin d'aboutir à un système théorique qui colle à la réalité… mais je vous laisse griffonner les tonnes de papier qui vont avec !
- Se contenter d'un système approchant qui permet de faire des mesures comparatives… et là le passage en mode expérimental va nous sauvez la mise ;-)

Vous l'aurez compris, je ne développerai que le second point dans la suite du document.

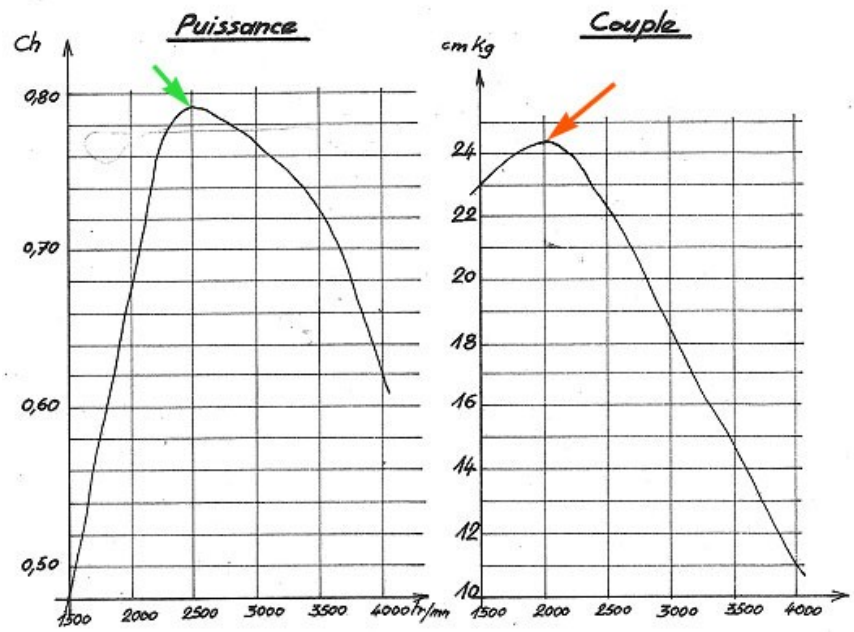
1 Prise en compte de l'inertie réelle et des frottements

On procède expérimentalement en s'appuyant sur une caractéristique constructeur connue d'un moteur de référence (comme la puissance maximum à un régime donné, voir ci-dessous [9]).

La méthodologie est la suivante :

- introduire une valeur arbitraire de J dans le code de l'Arduino (par exemple J=0,25kg.m²) et lancer une acquisition avec le moteur de référence ;
- relever la puissance maximum sur le graphique obetnu, par exemple 1,2cv à 2500rpm ;
- relever la puissance maximum sur les courbes constructeur (voir ci-dessous), par exemple 0,75cv ;
- remplacer J dans le code Arduino par J_{new} défini par une simple règle de trois : $J_{new} = \frac{0,75}{1,2} . J$

Exemple : extrait des archives VéloSolex d'époque… les points singuliers permettent de caler une inertie J expérimentale.



Document disponible dans la Documathèque CycloGalet.

9 notez que l'on peut procéder avec une référence calée sur le couple plutôt que la puissance, fonction des informations disponibles et des priorités que l'on se fixe…

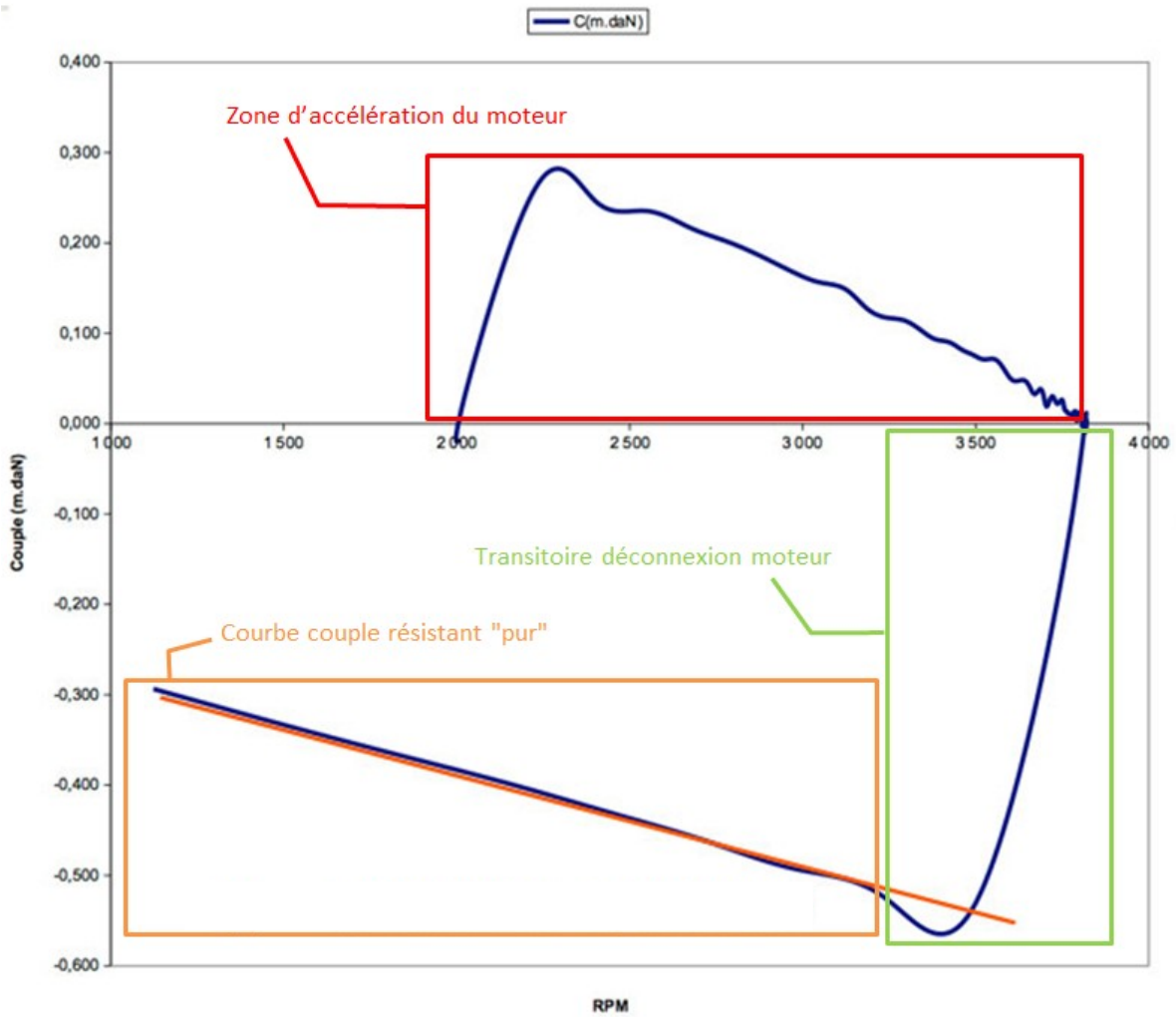
Cette méthodologie permet de caler le modèle sur un point de fonctionnement, de façon linéaire entre vitesse moteur et grandeur mesurée (utilisation d'une règle de trois!)… mais on est en droit d'avoir des doutes sur sa représentativité sur l'ensemble de la courbe obtenue. En effet, si l'influence des frottements n'est pas linéaire avec la vitesse, notre recalage est faux partout ailleurs sur la courbe !

Pour se rassurer, on va faire une petite expérience pour visualiser les frottements en fonction de la vitesse moteur :

- on initie une acquisition sur le banc à l'aide de notre Arduino connecté au PC,
- on démarre un moteur posé sur le banc et on le pousse au maximum de sa vitesse,
- puis on coupe brutalement la liaison moteur/banc (en relevant le moteur dans le cas d'un Solex !)
- et on enregistre la décélération du banc à l'aide de notre Arduino jusqu'à l'arrêt complet du rouleau.

On obtient une courbe dont la partie inférieure nous intéresse tout particulièrement puisqu'elle correspond au couple résistant du banc (phase de ralentissement, moteur relevé)… et comme un bon schéma vaut mieux que dix mauvaises explications, voici ce que l'on obtient:

ce qui nous intéresse ici est dans le rectangle orange… une belle décélération linéaire en fonction de la vitesse moteur, qui valide notre méthodologie de détermination de l'inertie par recalage par règle de trois ;-)



2 Prise en compte du glissement Galet|pneu|rouleau

Enfin, et pour les puristes, il faut tenir compte du glissement du Galet sur le pneu (cas du Solex par exemple), et de ce même pneu sur le rouleau. Cela peut paraître anodin, mais une erreur de 5% (valeur communément admise pour un Solex) à 4000rpm donne un décalage des courbes de 200rpm… et ça fait pas zoli sur les graphiques ;-) [10]

D'un point de vue théorique, il suffit d'inclure le glissement noté g% dans les formules de détermination de la vitesse angulaire moteur en fonction de la vitesse angulaire rouleau de la relation (R6) vue précédemment.

On a alors :

$$\omega_{moteur} = (1 + g\%) \frac{60}{2\pi} \omega_n \frac{D_{rouleau}}{D_{galet}}$$

Il suffit de corriger cette formule dans le code de l'Arduino, de déclarer la variable g% et le tour est joué !

Pour déterminer ce glissement (plus précisément cette combinaison de glissements !), le plus simple est d'avoir recours à un compte tour relié à l'allumage du moteur et de comparer la vitesse mesurée sur ce dernier avec celle affichée par le PC sur la base des acquisitions de l'Arduino effectuées au niveau du rouleau.

10 pour déterminer ce glissement (plus précisément cette combinaison de glissements !), le plus simple est d'avoir recours à un compte tour relié à l'allumage du moteur et de comparer la vitesse mesurée sur ce dernier avec celle affichée par le PC sur la base des acquisitions de l'Arduino effectuées au niveau du rouleau.

Creative Commons 3.0 by Bernique

VIII. Mises en œuvre pratiques

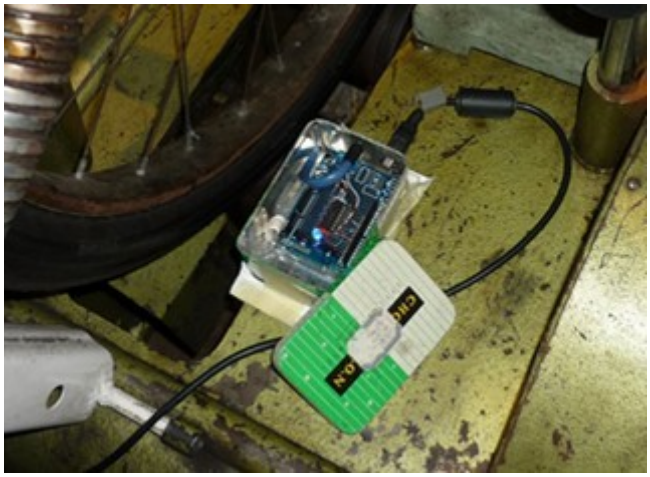
1 Exemple d'intégration sur banc maison (Nico_32)

Le banc peut être réalisé d'une façon simple : un cadre métallique permettant la fixation rapide du Solex, la roue AV de ce dernier roule sur un galet entraînant une roue à inertie de 16kg (rouleau de droite sur le banc). Une rondelle échancrée solidaire de l'axe de rotation de cette roue permet l'acquisition d'un signal par le capteur OMRON 1041.

2 Exemple d'intégration sur banc Marolo (Bernique)

Il s'agit d'un banc professionnel comme il s'en construisait autrefois à l'usage des concessionnaires.

Le dispositif Arduino + capteur est intégré sur le banc à l'aide de fixations adaptées (boite en fer servant de blindage, retenue sur le banc par des aimants). Le galet d'entraînement du banc est équipé d'un unique réflecteur qui viendra activer le codeur de proximité TCRT5000L.



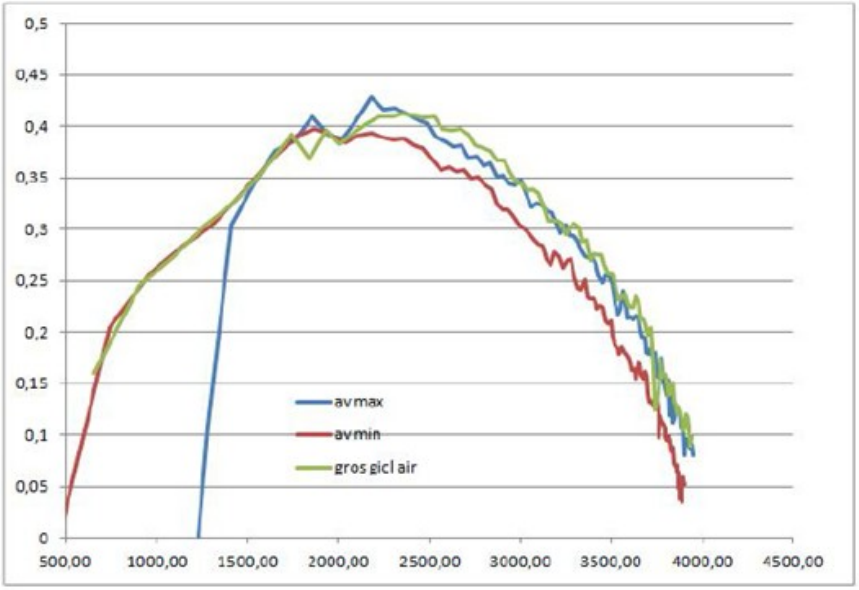
L'inertie des masses tournantes de ce banc (rouleaux + système de ventilation + motorisation modifiée + mesure des frottements, voir paragraphes précédents) déterminée par expérience est de 0,1735kg.m².

Creative Commons 3.0 by Bernique

3 Exemples d'exploitation des courbes obtenues

3.1.Impacts de trois réglages sur un bloc moteur (Nico-32)

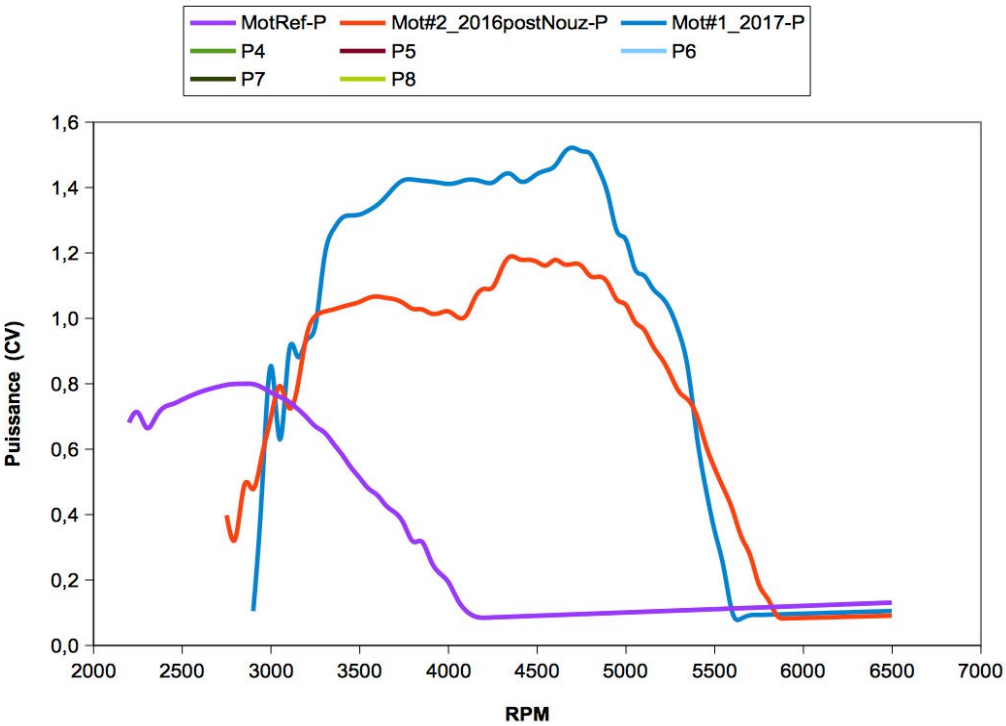
Étude de l'impact sur la puissance de trois modifications successives sur un même bloc moteur.



— Gicleur d'air normal, avance maximum - — Gicleur d'air normal, avance minimum - — Gros gicleur d'air, avance normale

3.2.Comparaison des performances entre moteurs (Bernique)

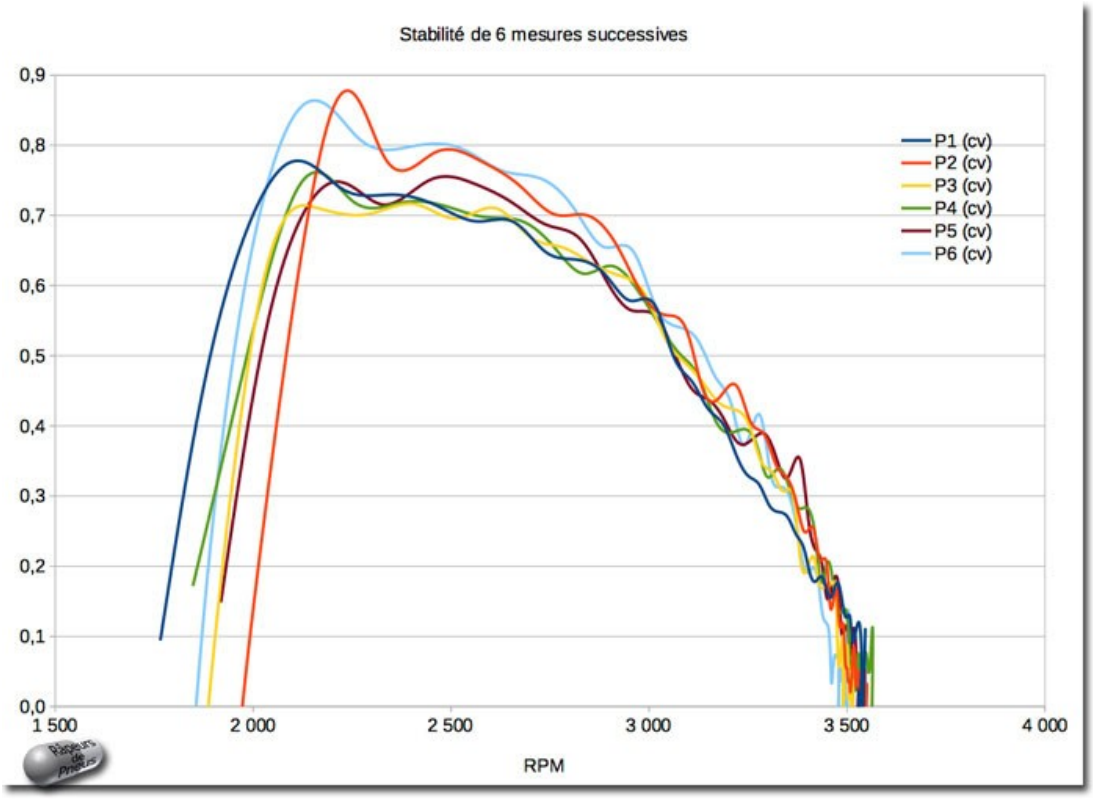
Comparaison d'un moteur de 3800 originel refait à neuf (courbe violette) et de deux moteurs préparés pour la course (catégorie Origine, team #90 Bernique Racing)



IX. Répétabilité des mesures

La courbe ci-dessous donne une idée de la répétabilité des mesures de puissance effectuée sur un bloc moteur de Solex 3800 refait à neuf et tester sur un banc Marolo instrumenté (six mesures successives sur un temps très court afin de limiter les paramètres exogènes).

On constate des écarts aux faibles régimes entre 2000 et 2500rpm. Ils sont essentiellement liés au mode opératoire. En effet, pour lancer l'acquisition, le moteur est freiné pour atteindre 20km/h par un dispositif à friction disponible sur le banc, puis lâché pour accélérer librement. La mécanique de freinage est difficile à régler et peu reproductible, d'où un moteur qui s'élance de façon différente à chaque essai avec notamment un embrayage qui peut, dans ces plages de vitesse, entrer en fonctionnement et fausser la mesure.
Ceci étant, hormis les courbes #2 et #6, on est à 10% de variation sur une plage courte de 500rpm, puis bien en deçà ensuite.



Bibliographie

- Site et forum Arduino
- Cours Arduino
- LOCODUINO
- ChiOSZ robots
- Forum Solex Compétition
- Parallax Inc.
- Millenium Modelisme
- Vishay
- Sheepdog Main Kennel
- ArduinoInfo
- Documathèque Cyclogalet

Annexe : le code

```
/*****************************************************************************************
 *  Engine power test based on inertia principle
 *
 *  Under GPL V3 license, http://www.gnu.org/licenses/gpl-3.0.en.html
 *
 *****************************************************************************************/

/*
 * Constants
 *   Values for the actual bench setup
 *   Update data every PUBLISH counts; with 20: 1s @ 3000 tr/mn, 10s @ 300 tr/mn
 *   Increase this for better RPM resolution, decrease for faster update
 */
#define PPR          1            // Pulses per revolution
#define J            0.3689       // par règle de 3
#define START_PB     4            // Pushbutton to start measurement on D4
int PUBLISH = 8;                  // Output data every PUBLISH pulses; must be a multiple of PPR, or 1

/*
 * Global variables
 */

/*diamètre galet en mm... à adapter selon machine (décocher la ligne à utiliser) */
float GALET=45.37;                // moteur de référence
//float GALET=42.00;              // moteurs course

int   pulse_count;                // Pulse count
float rpm, wn, wn1;               // Rotation speed respectively in rpm, rd/s
float wdn, wdn1;                  // Derivative of rotation speed in rd/s²
float tn, tn1, dt, t0;            // Sample times in s, compute time in ms, initial time
float Cn, C;                      // Couple in m.N and m.daN
float Pn, Pcv;                    // Power in W and CV
float V;                          // vitesse du Solex
float RATIO;                      // ratio galet banc sur galet moteur
boolean first_point;
const int LED=5;                  // declaration LED initialisation comptage

/*
 * Run once to setup
 *   Serial port transmission speed to the maximum possible (we use 128000, as this is what PLX-DAQ wants
 *   Spreadsheet header
 */
void setup() {
  // There is a catch here, as it takes time for the USB serial port to be recognized by the PC
  // we cannot initialize the spreadsheet header here
  Serial.begin(128000); // Transmission speed, i.e. 128000 bytes/s

  RATIO=99.1/GALET;

  pinMode(LED, OUTPUT);                                   //met la broche en sortie
  pinMode(START_PB, INPUT_PULLUP);
  pinMode(2,INPUT_PULLUP);
  pinMode(3,INPUT_PULLUP);
  pinMode(6,INPUT_PULLUP);
  pinMode(7,INPUT_PULLUP);
```

```
  pinMode(8,INPUT_PULLUP);
  pinMode(9,INPUT_PULLUP);
  pinMode(10,INPUT_PULLUP);
  pinMode(11,INPUT_PULLUP);
  pinMode(12,INPUT_PULLUP);
  pinMode(13,INPUT_PULLUP);
  pinMode(14,INPUT_PULLUP);
  pinMode(15,INPUT_PULLUP);
  pinMode(16,INPUT_PULLUP);
  pinMode(17,INPUT_PULLUP);
  pinMode(18,INPUT_PULLUP);
  pinMode(19,INPUT_PULLUP);

  //while(digitalRead(START_PB))   { };                    // Wait until button pressed
  digitalWrite(LED,HIGH);                                  // met la broche au niveau haut (5V) – allume la LED

  // Button pressed to start measurement
  // Clear spreadsheet and send header, then connect to interrupt routine
  Serial.println("MSG,Starting ...");
  Serial.println("CLEARDATA");                             // Clear data already in the spreadsheet (starting from row 2)
  Serial.print("LABEL,T(s),RPM,C(m.daN),P(CV),V(km/h),PUBLISH,");   // Column title, goes to row 1
  Serial.println(GALET,DEC);

  // Connect Interrupt 0 to interrupt routine
  attachInterrupt(0, pulse_int, RISING);

  pulse_count = 0;
  tn1 = 0; wn1 = 0; wdn1 = 0;
  t0 = micros()/1e6;

  first_point = 1;
}

/*
 * Main loop
 *   is used to compute measurements and output values to the serial port.
 *   We print statements which the PLX-DAQ macro is able to use
 *   Refer to :
 *     https://www.parallax.com/downloads/plx-daq
 *   for more details
 */
void loop()    {
    if (pulse_count >= PUBLISH) {
                if (first_point) {                         // First point is not a valid measurement, we discard
                            pulse_count = 0;
                            first_point=0;
                }
                else {
                            tn = micros()/1e6 - t0;
                            wn = (2*PI*pulse_count/PPR)/(tn-tn1);   // Angular speed rouleau en rad/s
                            wdn = (wn-wn1)/(tn-tn1);                // Angular speed derivative rouleau en rad/s²

                            // Save values for next time
                            tn1 = tn; wn1 = wn; wdn1 = wdn;
                            pulse_count = 0;

                            // Compute torque
                            Cn = J*wdn;                    // couple rouleau
```

```
                    // Compute power in W
                    Pn = wn*Cn;                                   // puissance rouleau

                    // Rotation speed in rpm
                    rpm = 60*wn*RATIO/(2*PI);                     // rpm moteur
                    // torque in m.daN
                    C = Cn/(10*RATIO);                            // si pas de prise en compte du couple résistant
                    // Power in CV
                    Pcv = Pn/736;

                    // vitesse en km/h
                    V=wn*(GALET/(2*1000))*RATIO*3600/1000;

                    Serial.print("DATA,");                        // envoi du temps et la mesure à Excel
                    Serial.print(tn,DEC);    Serial.print(',');
                    Serial.print(rpm,DEC);   Serial.print(',');
                    Serial.print(C,DEC);     Serial.print(',');
                    Serial.print(Pcv,DEC); Serial.print(',');
                    Serial.print(V,DEC); Serial.print(',');
                    Serial.println(PUBLISH,DEC);
                    dt = (micros()/1e6 - t0 - tn)*1000;
                    Serial.print("MSG,Compute time is ");         // envoi du temps et la mesure à Excel
                    Serial.print(dt,2); Serial.println(" ms");

                    // Publish variation
                    PUBLISH = 2*ceil((0.021*rpm-32)/2);
                    if (PUBLISH<4) PUBLISH=4;
                }
    }
}

/*
 * Interrupt routine
 *   All it does is to count pulses
 *   Each rotation, this interrupt function is run PPR times
 */
void pulse_int() {
  pulse_count++;
}
```