Bien que la gestion d'afficheurs 7 segments s'apparente à du pilotage banal de LED, il s'en différentie par deux facettes propres à ce type d'applications. D'une part il faut disposer d'une table de transcodage entre les caractères à symboliser et les diodes électroluminescentes à illuminer sur les afficheurs. D'autres part le nombre important de LED à piloter dépassant celui des sorties disponibles sur le microcontrôleur et il faut avoir recours à du multiplexage. L'afficheur utilisé dans cette application est un SH5461AS facilement disponible dans le commerce actuellement. Tout modèle à cathodes communes conviendra. La Fig.2 précise la désignation standard des divers éléments d'un afficheur 7 segments banal. Le brochage du SH5461AS est présenté sur la Fig.3 sur laquelle **K1** à **K4** désigne les cathodes communes des divers chiffres pris dans l'ordre de la gauche vers la droite. Les douze broches sont toutes utilisées, c'est le nombre d'E/S qui seront mobilisées sur Arduino pour piloter entièrement cet afficheur. Sur la Fig.3 le composant est représenté vu de dessus comme pour un circuit intégré de type DIL. Électroniquement deux approches sont possibles pour interfacer le SH5461AS. La première consiste, comme montré sur la Fig.4 à porter au potentiel de la masse les cathodes communes avec les sorties de pilotage d'Arduino, et à alimenter en +5v les segments à travers une résistance de limitation d'1kΩ par exemple. Cette approche correspond à celle d'une application réelle. Elle présente l'avantage d'avoir une bonne luminosité de l'afficheur, et surtout chaque segment présente une clarté égale et constante quel que soit le symbole représenté. Si les huit éléments sont allumés, l'intensité que doit fournir la sortie d'Arduino qui pilote **Ki** atteind 40 mA. Ce courant correspond au maximum toléré par une sortie binaire. Cette solution reste donc compatible avec les limitations de l'ATmega328 et n'impose pas de transistor de commutation par exemple. Par contre, dans le cas d'une simple évaluation, il est tout à fait envisageable de simplifier le montage expérimental. La Fig.5 décrit le schéma optimisé dans lequel on n'utilise que quatre résistances au lieu de huit. Chaque résistance est intercalée dans le pilotage des cathodes communes. De ce fait le courant maximal pour l'ensemble du "DIGIT" ne dépassera pas 5mA. Ce courant va se répartir dans les divers segments allumés. De ce fait, la luminosité de l'afficheur va varier de façon sensible en fonction du nombre de LED actives. Quand le chiffre huit est représenté et que le point décimal est allumé, nous obtenons la luminosité minimale. Elle est maximale quand un seul élément s'éclaire. Pour ne pas surcharger ce segment ou le Dp, on limite le courant total aux 5mA annoncés. Compte tenu du très bon rendement des LED de l'afficheur, cette solution simplifiée se justifie pleinement et s'avère très largement suffisante pour développer le programme. Que l'on adopte la solution de la Fig.4 ou celle de la Fig.5 dans les deux cas le programme reste strictement le même. Seuls les courants mis en jeu seront différents en fonction du choix effectué. Le plus économe est naturellement le schéma de la Fig.5 qui en moyenne consomme un courant quatre fois plus faible. Par contre, dans une application pérenne il semble évident que le montage de la Fig.4 s'imposerait.

Fig.1

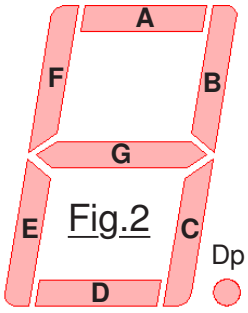


Fig.2

Brochage du SH5461AS vu de dessus.

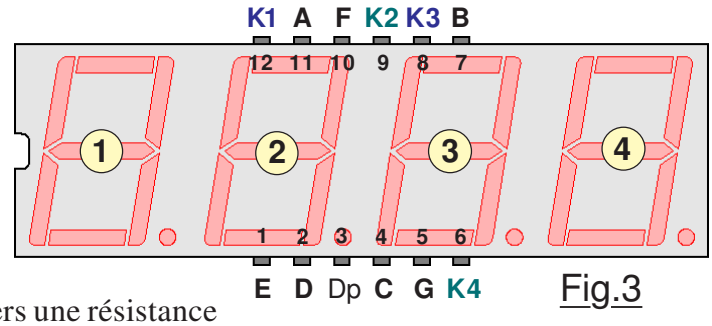


Fig.3

Fig.4

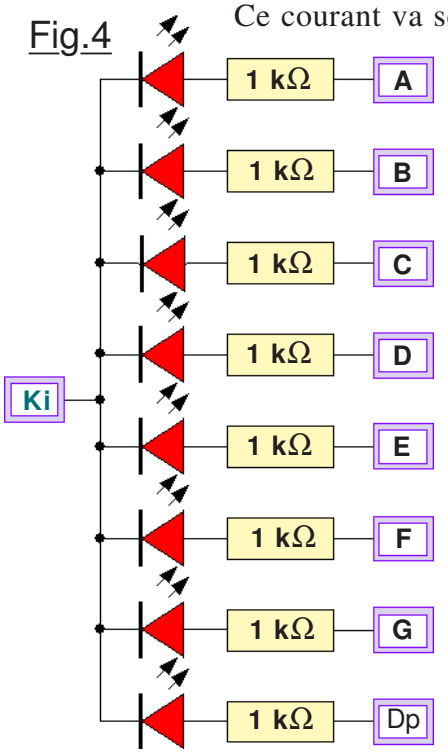
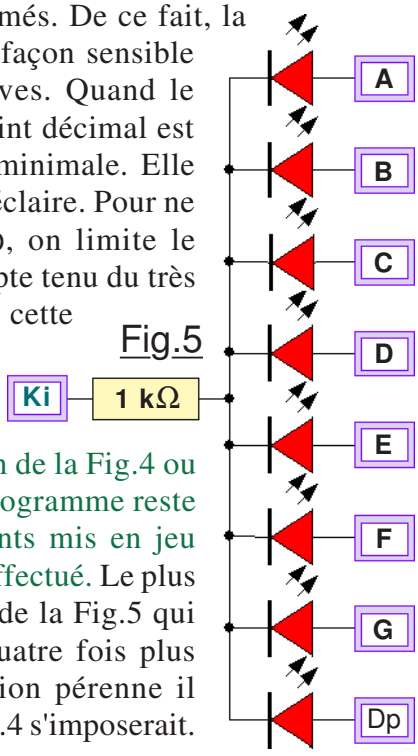


Fig.5



Analysons les détails d'agencement du programme Journal_defilant_7_segments.ino qui émule une petite application "ludique". Il s'agit d'un journal défilant. Le texte proposé est frappé dans la fenêtre du terminal série virtuel d'Arduino. Ce texte peut faire jusqu'à 63 caractères ce qui correspond au maximum possible dans la "fenêtre de saisie de la ligne USB". Lorsque l'on valide la ligne de texte saisie sur le terminal virtuel, celle-ci s'affiche par décalage de droite à gauche sur le SH5461AS. La vitesse de décalage est gérée par le programme. Le défilement latéral s'arrête sur les quatre derniers caractères des divers textes proposés. La nouvelle phrase "pousse la dernière". Compte tenu de la faible définition d'un afficheur 7 segments pour symboliser les caractères, il faut avoir recours à un mixage de lettres majuscules et de lettres minuscules pour pouvoir représenter tout l'alphabet. Le "W" , le "X" et "Z" ne sont pas très beaux, mais avec un peu d'habitude et dans le contexte ils se lisent facilement.

Le principe du multiplexage : La technique consiste à placer sur les 8 éléments *(Segments et Dp)* le niveau électrique haut d'environ +5v. Puis on allume l'afficheur concerné en portant sa cathode **Ki** à la masse. On laisse l'afficheur allumé durant 5mS puis on ramène sa cathode **Ki** au +5v pour l'éteindre. On passe immédiatement à l'afficheur voisin et on recommence. La rapidité de rafraichissement et l'inertie visuelle donnent l'illusion d'un éclairement permanent. Les mesures au périodemètre numérique donnent les valeurs suivantes : Allumage durant 5 mS, extinction pendant 15,2mS *(Balayage des trois autres "digits")* avec une période de 20,2mS qui se traduit par une fréquence de multiplexage légèrement inférieure à 50Hz. Concrètement c'est la fréquence de balayage de la boucle principale de base void **loop**. Cette boucle infinie représentée sur la Fig.6 passe la majorité du temps dans le multiplexage. Le traitement colorié par coloriage en rose sur l'organigramme n'exige que quelques milli secondes et passe totalement inaperçu sur le visuel. Le TAMPON comporte huit emplacements comme représenté en jaune sur la Fig.7 avec le FIFO colorié en bleu pastel. Le FIFO dispose de 63 octets pour pouvoir loger la ligne saisie sur le terminal série dont la mémoire de stockage correspond à cette taille. Tout caractère suivi d'un point décimal sera considéré comme un seul "DIGIT" puisqu'ils sont associés sur le SH5461AS. C'est la raison pour laquelle sur la Fig.7 les deux octets associés au même "DIGIT" sont mis en évidence par couleur orange. Comme rien n'interdit de frapper des caractères tous séparés par des points décimaux, il faut donc prévoir huit octets dans le TAMPON. Le balayage de multiplexage prend en compte tous les octets non nuls *($00 : Sentinelle)* dans le TAMPON, sachant qu'au maximum il ne peut y contenir que quatre caractères avec éventuellement un point décimal associé. Le transfert du FIFO vers le TAMPON en (2) est représenté sur la Fig.7 avec des marqueurs d'emplacements vides constitués d'octets nuls. Tous les caractères ne sont pas visualisables sur un afficheur 7 segments. Par exemple des caractères comme "#" et "%" seront ignorés. Lors du filtrage en (1) ces caractères ignorés sont remplacés par des "*" qui sont également non affichables. Ces caractères seront alors représentés en allumant uniquement le segment **D** pour les distinguer de l'espace qui lui éteint entièrement le "DIGIT" correspondant. Le transcodage caractère vers "DIGIT" utilise un octet avec les segments affectés comme montré ci-contre, les éléments étant ordonnés de la gauche vers la droite.

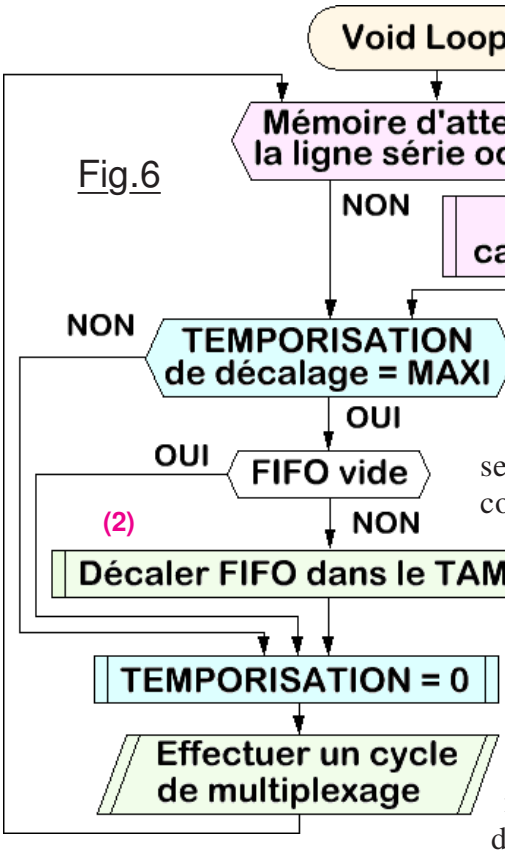


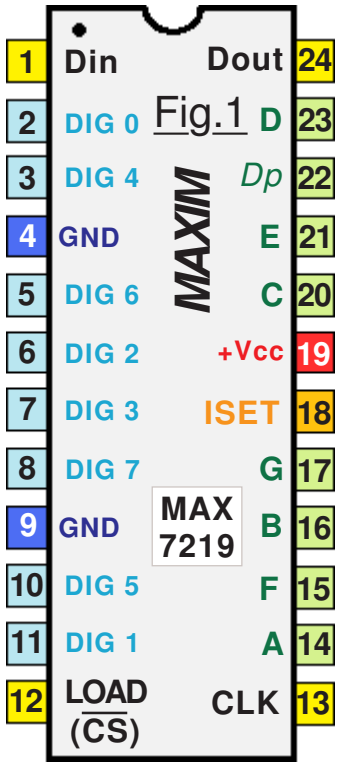
Fig.6

Fig.7 (2)

A	B	C	D	E	F	G	Dp

T	A	M	P	.	00	00	00	O	N	F	I	F	O	00	00	00

Toute application électronique exigeant un grand nombre de composants discrets pour sa mise en œuvre finit par imposer un ou des circuits intégrés spécialisés. Le multiplexage d'afficheurs 7 segments ou 16 segments n'échappe pas à cette règle imposée par les exigences commerciales. Le circuit MAX 7219 dont le brochage est donné en Fig.1 permet de multiplexer 8 afficheurs de type 7 segments à cathodes communes. Il peut également servir pour le multiplexage de matrices de LED très courantes sur le marché. Ce circuit intégré complexe se pilote par une ligne de type SPI, plusieurs de ces circuits pouvant être mis en "cascade" pour augmenter le nombre de "DIGIT" alignés sur le dispositif d'affichage. Une seule résistance **R** de 47 kΩ pour limiter le courant est suffisante. *(On peut descendre à 10kΩ mais ce n'est pas recommandé)* Montré sur la Fig.2 elle se place entre le **+Vcc** et **ISET**.



Caractéristiques techniques du circuit MAX7219 :

- Alimentation : 4,0 à 5,5 Vcc.
- Consommation en veille : 150 µA.
- Consommation tout éteint : 8 mA.
- Consommation tout allumé : 330 mA.
- Fréquence de balayage : 700 Hz à 1300 Hz. (1)

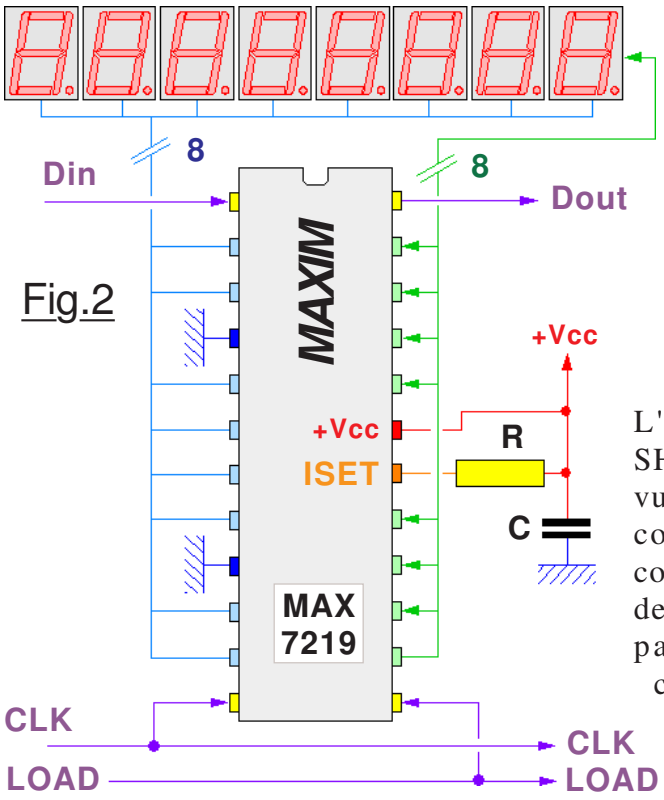


Fig.2

Brochage du SH5461AS vu de dessus.

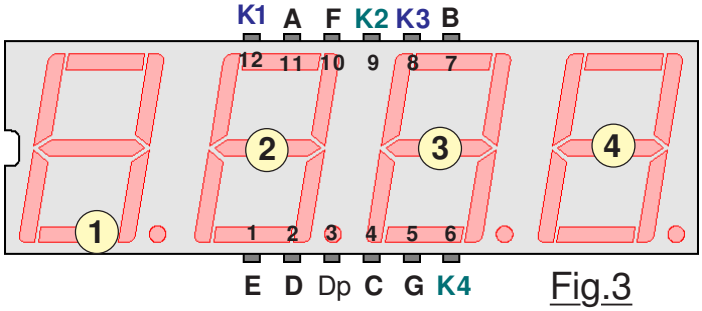


Fig.3

L'application présentée ici utilise un afficheur SH5461AS dont le brochage est donné sur la Fig.3 en vue de dessus. Il regroupe quatre "DIGIT" à cathodes communes. Il est impératif de mettre en place le condensateur **C** d'environ 0.01µF ou le circuit présente des aléas de fonctionnement. Dès que le courant exigé par les afficheurs dépasse les possibilités du composant, il ne fonctionne pas correctement et les "DIGIT" restent ± noirs. Il est préférable d'insérer une résistance de 47 kΩ quitte à limiter le balayage à quatre "DIGIT" pour rétablir une forte luminosité. Le programme Essai_AFF_7_SEGmt.ino donne un exemple très simple de mise en œuvre d'une petite application d'affichage sur 7 segments par utilisation de la bibliothèque **LedControl.h** qui fournit les fonctions de base pour gérer la ligne de commande SPI. *(Et le codage Binaire vers 7 SEGMENTS pour symboliser facilement les chiffres)* Dans cet exemple le balayage est limité aux quatre "DIGIT" et l'on se trouve aux limites de courant acceptable par le circuit intégré. La fréquence de balayage mesurée se situe aux environs de 1300 Hz. Si on valide le balayage sur tous les afficheurs elle chute à 700 Hz. Le courant moyen mesuré est de 33 mA. Le programme permet, au moyen d'un potentiomètre dont la tension variable entre 0 et +5Vcc branchée sur l'entrée analogique A0, de faire varier progressivement la luminosité de l'affichage qui sur RESET est minimal. On constate que lorsque la lumière est maximale, de petits et courts aléas apparaissent furtivement. *(Courant limite de courts instants)*

(1) : La fréquence de balayage est directement fonction du nombre de DIGIT balayés.

Gérer un grand nombre d'afficheurs 7 segments ou un nombre important de LED ne se conçoit plus sans le secours d'un circuit intégré de multiplexage tel que le MAX 7219 déjà mis en œuvre dans l'application précédente. Des matrices de LED existent à profusion dans le commerce, dont les dimensions sont variables. Parmi les plus courantes actuellement on trouve le module 1088AS par exemple dont la popularité à généré des kits "clef en main" tel que celui de la Fig.1 facilitant grandement la concrétisation de petites applications. Le 1088AS comporte 64 LED arrangée en huit lignes et huit colonnes. La Fig.2 en présente le brochage lorsque ce composant est vu par le dessus. Les broches sont numérotées comme celles d'un circuit intégré de type DIL, toujours observé par dessus. Le petit montage électronique de la Fig.1 tient compte des caractéristiques de cet afficheur matriciel. Comme montré sur la Fig.3 les connecteurs sont conformes à son brochage interne. Notons au passage que si l'espacement entre deux broches sur le 1088AS fait bien les 2,54mm standard, la distance entre les deux lignes de broches n'est pas normalisée puisqu'elle fait 24,5mm ce qui ne correspond pas à exactement un pouce.

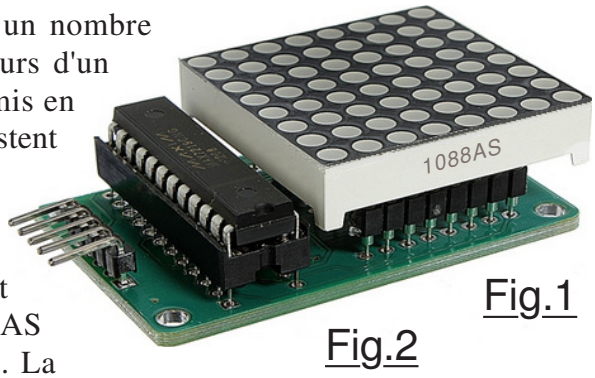
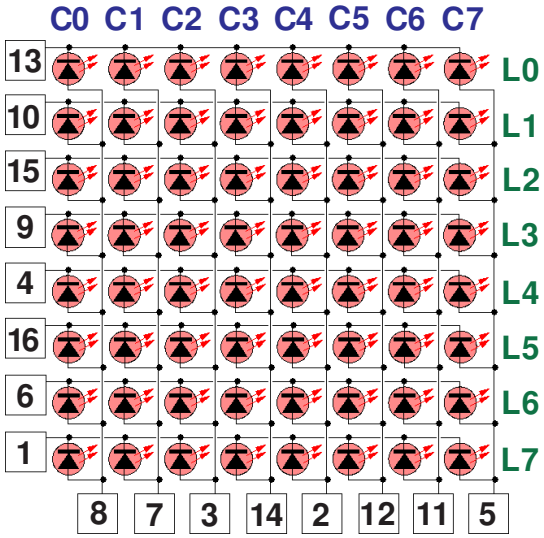


Fig.1

Fig.2



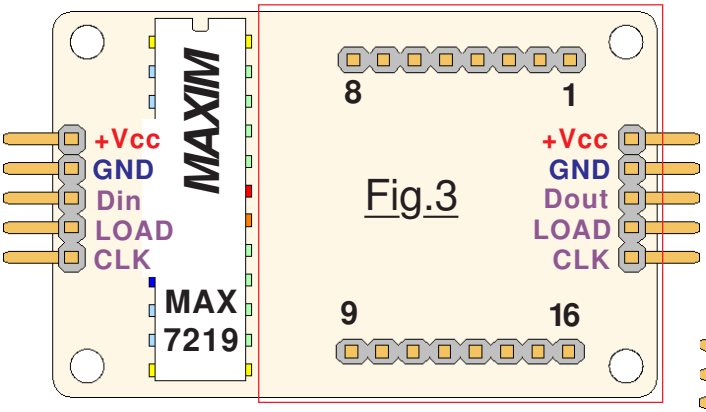
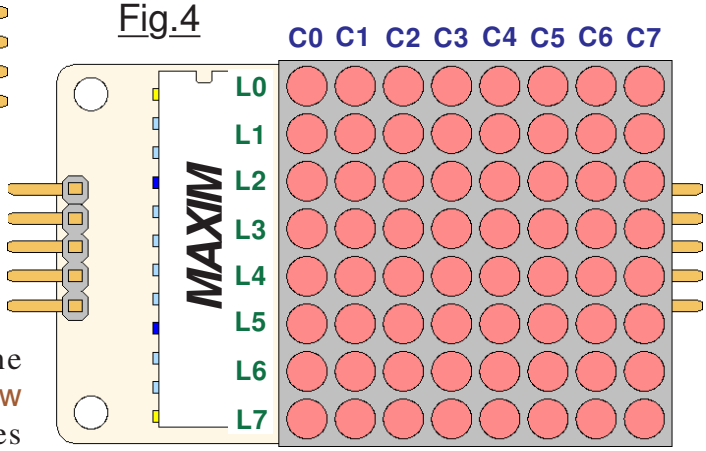


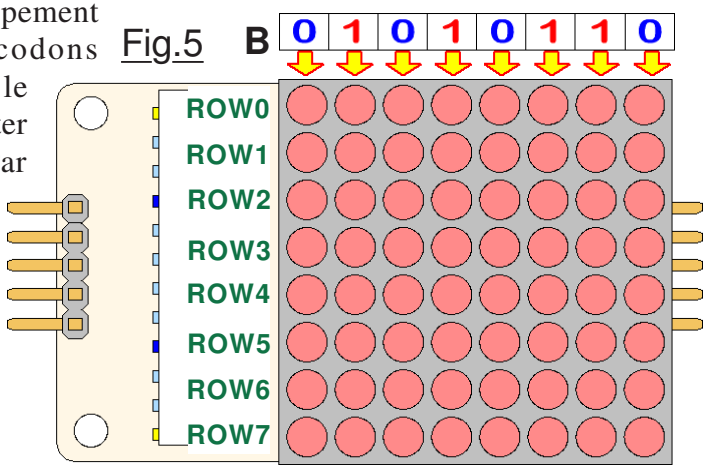
Fig.3

Fig.4



La bibliothèque **LedControl.h** permet de gérer l'allumage des LED soit individuellement soit par "Rangées". Pour programmer facilement une application en usant des procédures setLed et setRow il importe de faire le lien entre les coordonnées indiquées et la position de l'afficheur sur le module électronique. La Fig.4 précise l'ordre des lignes et des colonnes quand on programme LED par LED avec setLed en précisant leurs coordonnées. La Fig.5 pour son compte précise l'ordre des "rangées" pour setRow avec le symbole de l'OCTET dans lequel on précise par des "1" et des "0" l'état des LED dans le groupement traité. Mais comme nous codons librement les coordonnées ou le contenu des octets, on peut orienter l'afficheur comme on le désire. Par exemple, pour le petit programme l'application **Essai_Matrice_LED_8x8.ino** le circuit imprimé est orienté comme montré sur la Fig.6 de façon à ne pas être masqué par les fils de liaison pour l'ISP.

Fig.5



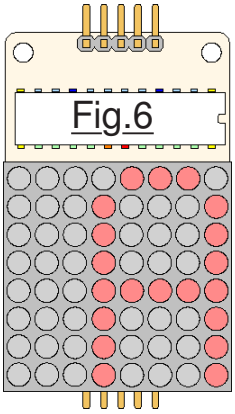


Fig.6

Piloter un moteur pas à pas relève d'une logique élémentaire et ne présente pas de difficulté particulière. Il importe toutefois d'interfacer le microcontrôleur avec une électronique de commutation apte à gérer la tension d'alimentation des bobines, le courant consommé et les surtensions de commutation. Compte tenu de l'importance de ce type de moteurs dans les automatismes, les produits commerciaux couvrent tous les besoins, que ce soit avec des moteurs de toutes puissances, ou avec des circuits intégrés spécialisés pour les piloter. L'un des plus populaires étant l'ULN2003AN qui simplifie considérablement l'interface de commutation de puissance. Le petit moteur utilisé dans cette application montré sur la Fig.1 est un 28BYJ-48 avec en Fig.2 le brochage de son connecteur.

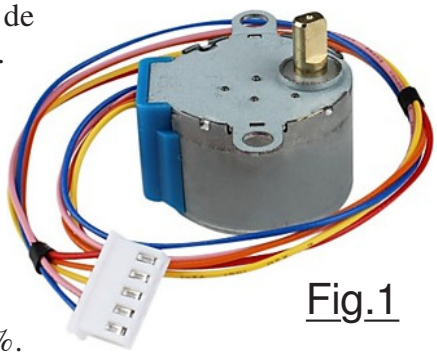
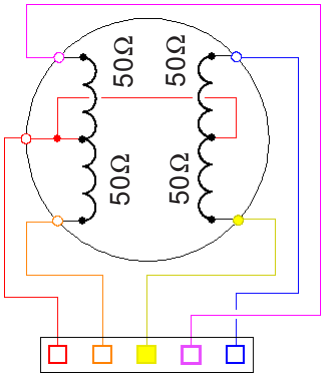


Fig.1

Caractéristiques techniques du moteur 28BYJ-48 :

Fig.2



- Tension d'alimentation : 5Vcc.
- Résistance en courant continu : 50Ω ± 7%.
- Résistance d'isolement >10MΩ. *(500V)*
- Rapport de réduction: 1/64.
- Angle de rotation par pas: 5,625° / 64. *(Soit 0,08789°/pas)*
- Fréquence d'utilisation : 100Hz.
- Diamètre 28 mm, épaisseur 20mm et implantation 35 mm.

Les moteurs pas à pas bipolaires comportent généralement quatre fils sortant de ces derniers plus éventuellement un commun "central", contrairement aux moteurs unipolaires qui n'ont aucun lien de centre commun. Le modèle utilisé 28BYJ-48 est bipolaire avec fonctionnement à huit phases.

Le pilotage d'un moteur pas à pas :

Le schéma fondamental d'une électronique de pilotage est développé en Fig3 sur lequel on retrouve un transistor par bobine à commuter. Pour gérer un moteur bipolaire avec commun central il faut donc deux sections analogues à celle proposée ci-contre. On retrouve en **Rb** la résistance de limitation du courant de base, et surtout les diodes **D** qui éliminent les surtensions inductives. On peut éventuellement augmenter le gain en courant en complétant **T1** *(Et T2)* par un deuxième transistor agencé en Darlington. La chute de tension à l'état saturé avoisine 1Vcc.

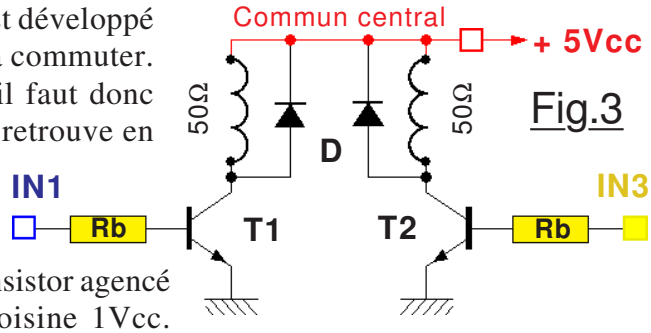


Fig.3

Alimenté sous 5V, chaque enroulement de 50Ω va consommer environ 100mA. Dans le montage d'expérimentation les transistors sont des composants ordinaires pour faible puissance. Leur gain en courant étant d'environ 100, pas besoin de les assembler en Darlington. Avec une résistance de base d'1kΩ ils sont en saturation. Ils ne chauffent absolument pas, donc inutile de les placer sur radiateurs. Les diodes **D** sont des composants pour petits signaux banales. Chaque collecteur est relié à une LED qui va au +5Vcc avec une résistance de limitation de courant également d'1kΩ pour visualiser l'état passant des transistors sur chaque bobine et également pouvoir contrôler le mode veille. Quand le moteur est en rotation et les quatre LED allumées, le courant total fait environ 120mA ce qui est largement supportable par une alimentation USB de la carte Arduino. Le petit programme Moteur_PAS_A_PAS.ino permet de tester l'interface de composants discrets représentée en Fig.3 qui utilise deux fois ce petit schéma.

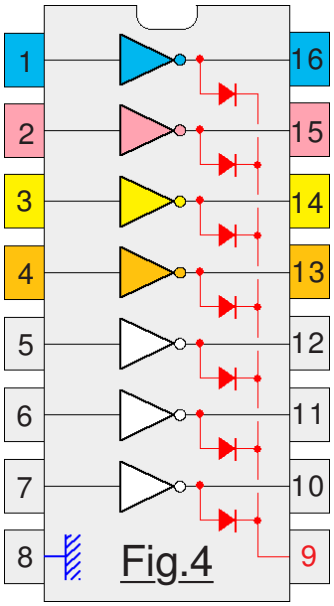


Fig.4

Le circuit intégré de pilotage ULN2003 :

Bien que le nombre de composants pour gérer la commutation du moteur pas à pas bipolaire ne soit pas très élevé, il est bien plus avantageux d'utiliser un circuit spécialisé dont le coût n'est pas plus élevé que celui de la solution intégrant des composants discrets et qui tient bien moins de place sur

le circuit imprimé. Un ULN2003 dont le brochage est dévoilé sur la Fig.4 intègre sept circuits de commutation indépendants schématisés sur la Fig.5 et pourvus des diodes de protection sur les entrées, ainsi que les diodes **D** toutes réunies à un Commun. Toujours avec le programme Moteur_PAS_A_PAS.ino il suffit de remplacer les composants discrets par un tel circuit intégré pour obtenir immédiatement le même comportement du moteur. Mis à part éventuellement les DEL et leurs résistances de limitation de courant, il n'y a rien à ajouter au circuit intégré pour piloter le 28BYJ-48. On remarque sur la Fig.5 que la résistance de base **Rb** est intégrée, ainsi que ***D1*** pour protéger l'entrée d'une tension négative et ***D2*** pour éviter un courant inverse sur la sortie. Le montage adopté est celui d'un amplificateur de courant en montage Darlington. La tension résiduelle en état de saturation est inférieure à 1Vcc, du coup le courant total passe à 170mA. De plus le moteur est plus "nerveux" et il devient possible de passer delay(2) à delay(1) pour obtenir une rotation plus rapide sans pour autant perdre des pas. Sachant que le collecteur peut commuter 500mA et que le courant de base peut atteindre 25mA en permanence, dans notre application l'ULN2003 n'est vraiment pas très sollicité.

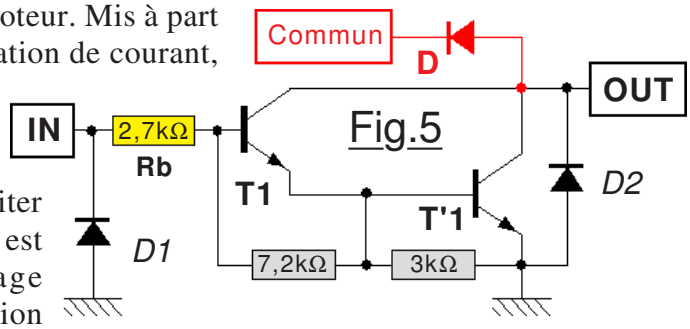


Fig.5

Carte électronique du commerce :

Compte tenu de la forte demande de tel circuits pour des petites applications de robotique, on se doute qu'il existe dans le commerce une foule de cartes toutes câblées nous épargnant l'étude et la réalisation d'un circuit imprimé. Leur coût restant très modéré, les utiliser relève pratiquement de l'évidence. La Fig.6 montre l'un de ces produits dont le connecteur est directement compatible avec le branchement de la fiche d'un 28BYJ-48. De plus, ce tout petit circuit intègre quatre LED de visualisation avec leurs résistances de limitation en courant. Le "strap" visible sur la Fig.6 permet d'isoler le moteur du Commun

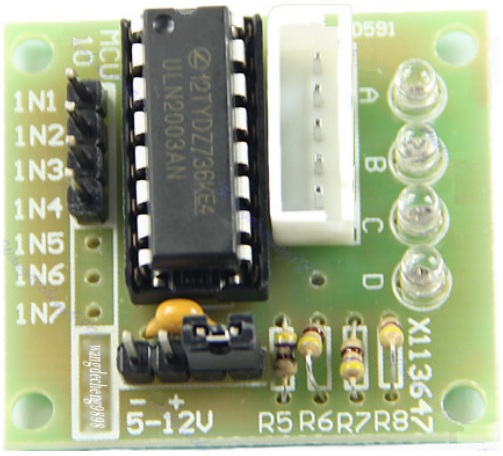


Fig.6

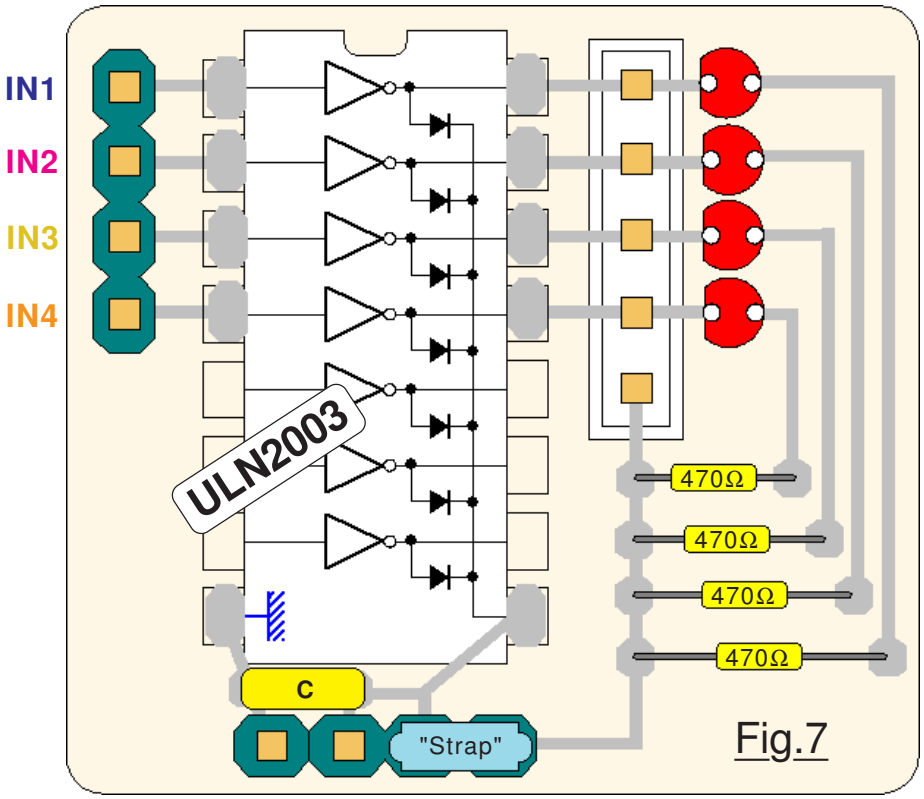


Fig.7

ce qui permet à la demande de commuter aussi des moteurs unipolaires. Le dessin de la Fig.7 ci-contre présente le câblage de la petite carte électronique du commerce. On notera la présence d'un condensateur de découplage **C** entre le +Vcc et la masse. Les trois opérateurs non utilisés ne sont pas reliés aux connecteurs et de ce fait sont indisponibles. Le circuit intégré est placé sur un support pour en faciliter l'éventuel changement bien que ce composant soit très fiable. La tension d'alimentation préconisée par la sérigraphie précise que l'on peur aller jusqu'à +12Vcc si le moteur utilisé est conçu pour cette tension.

L'utilisation des moteurs pas à pas n'est pas très compliquée. On peut définir ses propres procédures comme dans le programme Moteur_PAS_A_PAS.ino cité en exemple en page 60. Il est naturellement possible d'utiliser la bibliothèque Stepper.h décrite en page 7 du livret consacré aux librairies les plus courantes. Le programme est bien plus simple à écrire, et plus facile à lire. Mais c'est au détriment de la taille occupée en mémoire. L'utilisation de cette bibliothèque consomme environ 570 octets de plus que le même programme Sans_Librairie.ino écrit intégralement avec des procédures personnelles. Pour montrer la facilité d'utilisation de la librairie spécialisée, le programme Librairie_PAS_A_PAS.ino listé ci-dessous permet de piloter un 28BYJ-48 commandé par un potentiomètre dont la sortie va sur l'entrée analogique A0. Noter qu'il n'est pas nécessaire d'aller télécharger Stepper.h sur Internet car elle fait partie des librairies déjà disponibles par défaut à l'installation du compilateur.

```
/* Test des moteurs pas à pas en utilisant la bibliothèque Stepper.h décrite sur :
   http://www.arduino.cc/en/Reference/Stepper
   Un potentiomètre est branché sur l'entrée analogique A0. Le moteur pas à pas
   recopie linéairement les variations de tension présentes sur A0. */

#include <Stepper.h>

#define Nb_tours_par_minute 240 // Vitesse de rotation imposée au moteur.
#define Nb_pas_par_tour 64 // Définition du moteur réducteur non pris en compte.

// Déclaration des brachements.
#define Broche1 9  // Bleu   (broche 1)
#define Broche2 11  // Rose   (broche 2)
#define Broche3 8  // Jaune  (broche 3)
#define Broche4 10  // Orange (broche 4)

Stepper MOTEUR(Nb_pas_par_tour, Broche3, Broche1, Broche4, Broche2);

int Mesure_sur_A0, A0_Precedente = 0;

void setup() {
  MOTEUR.setSpeed(Nb_tours_par_minute); // Définir la vitesse de rotation.
  Mesurer(); A0_Precedente = Mesure_sur_A0; } // Immobile au RESET.

void loop() {
  Mesurer();
  MOTEUR.step((Mesure_sur_A0 - A0_Precedente) * 2.05); @
  // Valeur multipliées par 2.05 pour que la portée du potentiomètre corresponde à
  // un tour effectué en sortie de réducteur sur le moteur 28BYJ-48.
  A0_Precedente = Mesure_sur_A0; }

void Mesurer() {
  Mesure_sur_A0 = analogRead(0); // 50 mesures pour stabiliser le résultat.
  for (byte I = 0; I < 50; I++) {Mesure_sur_A0 = (Mesure_sur_A0 + analogRead(0)) / 2; } }
```

Ce programme "recopie" la position de l'axe du potentiomètre. Toute modification de son ajustement se traduit par une rotation du moteur dans le sens direct ou dans le sens rétrograde pour respecter la consigne. L'instruction @ ajuste le nombre de pas de telle façon que la portée du potentiomètre *(De 0 à +5Vcc sur l'entrée A0)* corresponde à pratiquement un tour d'arbre en sortie du réducteur.

ATTENTION : Le moteur ne fonctionnera correctement que si l'on respecte les branchements électriques définis sur la Fig.1 ci-contre.

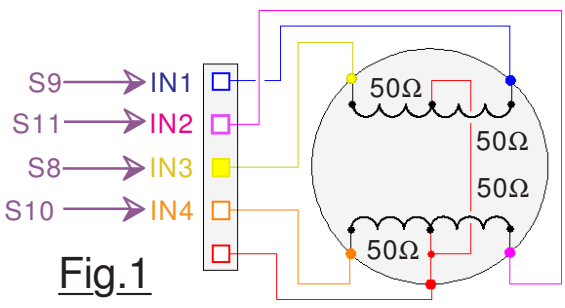
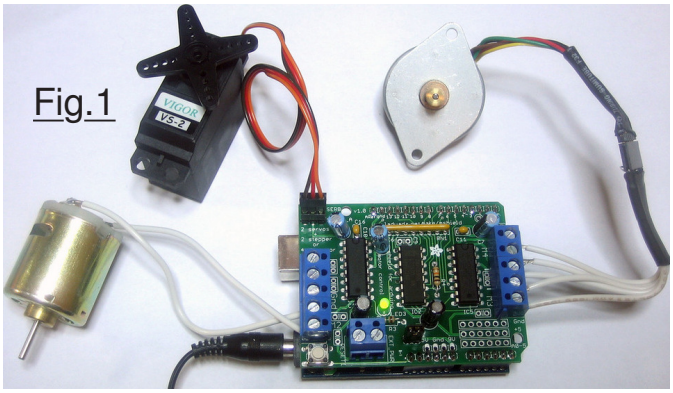


Fig.1

Shield ADAFRUIT pour piloter des moteurs à courant continu :

L'entreprise ADAFRUIT implantée à New York s'est spécialisée notamment dans la conception et la fourniture de petits modules pour les cartes Arduino et propose divers produits **dont un shield spécifique pour piloter des petits moteurs**. Comme montré sur la Fig.1, cette carte est très polyvalente, et l'on peut aussi-bien y brancher des moteurs à courant continu, des moteurs pas à pas ou des servomoteurs. Le nombre de moteurs pilotables simultanément dépend des types de motorisations utilisées.



Caractéristiques de la carte d'interface :

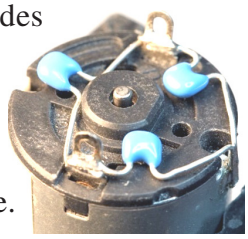
- Deux connexions pour des servomoteurs 5V. *(Liaisons directes sur les E/S d'Arduino.)*
- Quatre ponts en H gérés par des circuits intégrés L293D capables de fournir 0,6 A par pont *(1.2A crête)* pourvus d'une protection d'arrêt thermique et de diodes internes "de roue libre". Ces ponts peuvent fonctionner avec des tensions allant de 4.5Vcc à 25Vcc.
- Gère dans les deux sens de rotation possibles jusqu'à quatre moteurs à courant continu avec sélection individuelle de la vitesse sur 8 bits.
- Pilote jusqu'à deux moteurs pas à pas *(Unipolaires ou bipolaires)* avec une seule bobine, double bobinage ou enroulements entrelacés.
- Les résistances de charge des moteurs sont désactivées pendant la mise sous tension.
- Un bornier sérieux permet d'établir aisément les liaisons électriques avec les moteurs. *(18-26AWG)*
- Le bouton de réinitialisation d'Arduino déporté sur le circuit imprimé.
- Deux broches et un cavalier permettent d'isoler l'alimentation externe des moteurs de la logique.
- Possibilité de brancher simultanément deux servomoteurs sous 5Vcc et :
  4 moteurs à courant continu ou,
  2 moteurs pas à pas ou,
  2 moteurs à courant continu et un moteur pas à pas.

Entrées/Sorties d'Arduino utilisées :

- Les six broches des entrées analogiques sont totalement disponibles.
- Les deux broches des E/S numériques 2 et 13 sont totalement disponibles.

Les broches suivantes ne sont utilisées que si un moteur s'y trouve en cours d'utilisation :
Broche numérique 11:Moteur DC n ° 1 / Pas à pas n ° 1*(Activation / contrôle de vitesse)*
Broche numérique 3: Moteur DC n ° 2 / Pas à pas n ° 1*(Activation / contrôle de vitesse)*
Broche numérique 5: Moteur DC n ° 3 / Pas à pas n ° 2*(Activation / contrôle de vitesse)*
Broche numérique 6: Moteur DC n ° 4 / Pas à pas n ° 2*(Activation / contrôle de vitesse)*
Les broche numérique 4, 7, 8 et 12 sont utilisés pour commander les moteurs DC ou pas à pas via la logique de pilotage du circuit intégré 74HC595.
Les broches suivantes ne sont utilisées que si un servomoteur y est en cours d'utilisation :
Broche PWM numérique 9 : Commande du connecteur **SERVO_2**.
Broche PWM numérique 10 : Commande du connecteur **SER1**.

NOTE : Une LED s'allume *(Voir Fig.2)* si l'alimentation de puissance est effective. Si elle n'est pas allumée, les moteurs DC ou les moteurs pas à pas ne fonctionneront pas. Les ports des servomoteurs sont en 5Vcc et n'utilisent pas l'alimentation extérieure. Ce module ne peut pas piloter des moteurs à faible tension prévus pour 3Vcc. Il est prévu pour une tension de 6Vcc ou plus. Les moteurs DC génèrent beaucoup de parasites tant magnétiques qu'électriques pouvant perturber Arduino. Il est donc fortement conseillé de les alimenter par la ligne extérieure et de munir chaque moteur de trois condensateurs d'antiparasitage : Un condensateur en parallèle sur le moteur, deux entre les broches d'alimentation et la masse.



Alimentation des moteurs :

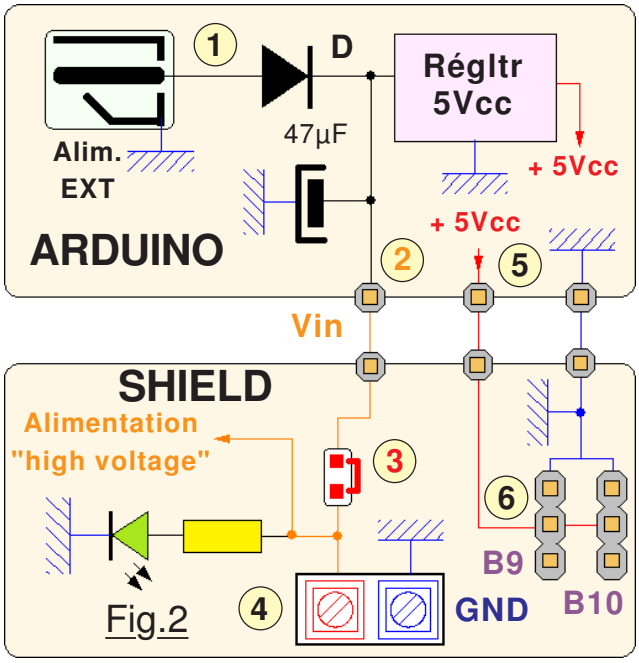


Fig.2

Si on désire n'utiliser qu'une seule source pour fournir l'énergie électrique à Arduino et au module de gestion des moteurs, il suffit de brancher une alimentation externe sur la prise prévue à cet effet **1**. Dans ce cas on n'utilise pas le connecteur **4** et le cavalier **3** doit être en place sur son connecteur. Cette méthode présente toutefois un certain nombre d'inconvénients. Le courant fourni par la broche **Vin** en **2** est limité par les caractéristiques de la diode **D**. (*Vin est également désignée par **UTN** sur certains schémas électroniques d'Arduino.*) De plus, les parasites générés par les moteurs électriques peuvent transiter vers Arduino via le régulateur intégré fournissant le +5Vcc au microcontrôleur. Cette solution n'est donc acceptable que si les moteurs utilisés sont de faible puissance et ne consomment qu'un courant modéré. Il est préférable d'alimenter Arduino soit par sa ligne USB soit par une alimentation externe, et utiliser pour l'interface de motorisation une alimentation à part branchée sur le connecteur **4**. Dans ce cas il faut retirer le cavalier **3** de son support. Noter que les servomoteurs sont reliés directement au +5Vcc d'Arduino par la broche **5**. On ne peut utiliser les connecteurs dédiés **6** que si les modèles utilisés restent dans des fourchettes de consommation faibles. Dès qu'ils deviennent plus exigeants en courant électriques, il faut comme pour les moteurs DC les alimenter par un "bloc" extérieur.

UTILISATION DU CIRCUIT SHIELD AVEC DEUX SERVOMOTEURS.

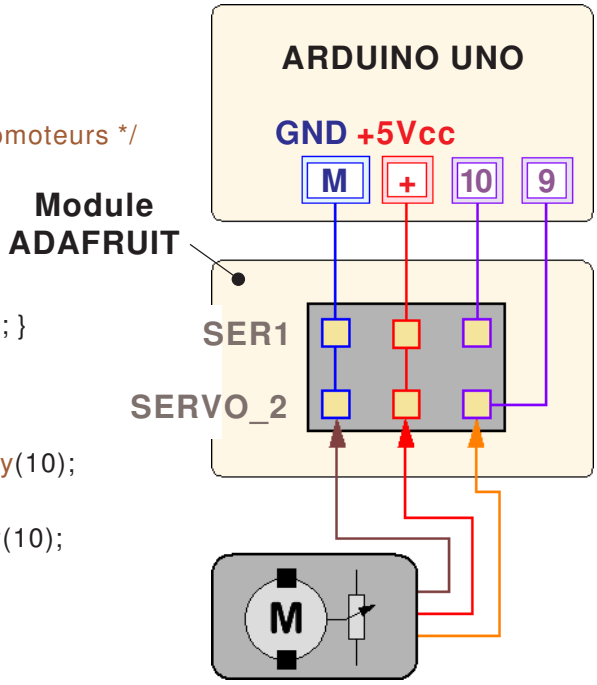
Comme montré sur les schémas Fig.2 et Fig.3, le petit circuit imprimé d'ADAFRUIT n'apporte strictement rien de plus à la carte Arduino en terme d'électronique. Il ne fait que relier les connecteurs **SER1** et **SERVO_2** à la masse, au +5Vcc d'Arduino et aux sorties binaires **B9** et **B10** en respectant l'ordre des fils sur les fiches à trois broches. Comme les servomoteurs puisent leur énergie directement sur le +5Vcc d'Arduino il importe d'en vérifier la consommation. Si elle n'est pas compatible avec les possibilités de la prise USB il faut alimenter les moteurs à part. La programmation est donc banale et utilise les informations données à ce sujet dans les pages 38 et 39.

Fig.3

Exemple de pilotage ANGULAIRE avec la librairie Servo.h :

Programme Test_4_avec_deux_Servomoteurs.ino

```
/* Test de la carte ADAFRUIT pour le pilotage de deux servomoteurs */
#include <Servo.h>
Servo Servo1; Servo Servo2;
void setup() {
  Serial.begin(115200); Servo1.attach(9); Servo2.attach(10); }
int I;
void loop() {
  for (I=0; I<180; I++)  {Servo1.write(I); Servo2.write(I); delay(10);
                          Serial.print(" I = ");Serial.println(I); }
  for (I=180; I!=0; I--) {Servo1.write(I); Servo2.write(I); delay(10);
                          Serial.print(" I = ");Serial.println(I); } }
```



Programmation et utilisation du Shield ADAFRUIT :

Comme c'est les cas presque chaque fois qu'un composant spécifique ou qu'une carte électronique spécialisée sont disponibles sur le marché, une ou des bibliothèques sont développées et mises en ligne pour le public. Le shield ADAFRUIT n'échappe pas à cet adage et plusieurs librairies le concernant sont disponibles sur internet. En particulier AFMotor.h est dédiée à ce module d'interfaçage et couvre déjà pas mal les besoins de la programmation des petites motorisations .

UTILISATION DES MOTEURS À COURANT CONTINU.

Bien que l'électronique autorise le branchement d'un moteur par opérateur de commutation, *(Cas où un seul sens de rotation est suffisant.)* seuls les liaisons en H sont prises en compte par la bibliothèque AFMotor.h, mais avec pour corollaire la possibilité d'imposer à convenance un sens de rotation. On peut dans ces conditions gérer simultanément jusqu'à quatre petits moteurs à courant continu.
Le programme Test_2_moteurs_DC_avec_rampe_de_vitesse.ino est un exemple de mise en œuvre de la carte électronique qui fait tourner quatre moteurs *(Le maximum possible)* dans un seul sens pour simplifier le programme, mais avec une rampe de variation de vitesse en accélération et ralentissement.
Le programme Test_3_moteur_DC_avec_Rampe_et_Repos.ino est une variante du précédent qui ne fait tourner qu'un seul moteur sur le port #4, mais dans les deux sens, toujours avec une rampe de variation de vitesse en accélération et ralentissement. Dans ce programme la déclaration le 1kHZ par défaut est imposée pour la fréquence PWM. Après un cycle dans les deux sens de rotation, le moteur est mis au repos durant une seconde. L'utilisation des procédures fournies par AFMotor.h est quasiment évidente, par contre il faut tenir compte des courants consommés comme vu dans les pages précédentes.

NOTE : Lors des divers essais effectués, aucun de ces moteurs n'était pourvu de condensateurs d'antiparasitage comme montré sur le dessin situé en bas de la page 63. Que ce soit en alimentation directe par Arduino ou par le truchement d'une alimentation extérieure, aucune perturbation n'a été constatée lors du déroulement des programmes. Quand les moteurs utilisés sont de faible puissance et alimentés en tensions faibles, les parasites restent donc suffisamment modérés pour ne pas imposer de condensateurs.

Caractéristiques des petits moteurs utilisés en local.

Plusieurs petits moteurs de récupération sont disponibles "dans les tiroirs".

Moteur DC miniature **RF-300C-09550** :
- Appel de courant au démarrage sous 7Vcc : Inférieur à 100 mA.
- Courant moyen en régime continu à vide sous 7Vcc : 30 mA.
- Courant sous 5Vcc **rotor bloqué** mécaniquement : 800 mA.

Moteur DC miniature **MDN3BL3DRB** :
- Appel de courant au démarrage sous 7Vcc : Environ 200 mA.
- Courant moyen en régime continu à vide sous 7Vcc : 35 mA.
- Courant sous 5Vcc **rotor bloqué** mécaniquement : 800 mA.

Moteur DC MITSUMI **P/NC8974-60010** :
- Pas d'appel de courant au démarrage sous 5Vcc.
- Courant moyen en régime continu à vide sous 5Vcc : 150 mA.
- Courant sous 5Vcc **rotor bloqué** mécaniquement : 500 mA.

Moteur DC "moyen" **HC385MG** :
- Pas d'appel de courant au démarrage sous 5Vcc.
- Courant moyen en régime continu à vide sous 5Vcc : 90 mA.
- Courant sous 5Vcc **rotor bloqué** mécaniquement : 700 mA.

Moteur DC "moyen" sans référence :
- Appel de courant au démarrage sous 5Vcc : Environ 80 mA.
- Courant moyen en régime continu à vide sous 5Vcc : 50 mA.
- Courant sous 5Vcc **rotor bloqué** mécaniquement : 700 mA à 800 mA.

UTILISATION DES MOTEURS PAS À PAS.

Contrairement aux caractéristiques des moteurs à courant continu qui n'influencent pas réellement la programmation, le codage pour les moteurs PAS À PAS est directement fonction de leur résolution. La procédure setSpeed(RPM) qui gère la vitesse de rotation utilise la résolution du moteur utilisé. Outre les problèmes liés au courant consommé, il faudra tenir compte du nombre de **pas par tour**, raison pour laquelle l'encadré donné en bas de cette page précise les caractéristiques des moteurs utilisés en local. On peut noter au passage qu'un moteur PAS À PAS tel que le 28BYJ-48 n'est pas utilisable car le point milieu des deux bobinages est commun et perturberait la commutation en H.

UTILISATION ÉLÉMENTAIRE DES MOTEURS PAS À PAS.

Par élémentaire il faut traduire : Pas de pilotage simultané de deux moteurs PAS À PAS et pas de rampes d'accélération ou de ralentissement. En effet, la bibliothèque AFMotor.h ne prévoit pas de procédure pour traiter les rampes d'accélération ou de ralentissement. Par ailleurs le pilotage simultané de deux moteurs pas à pas est laissé à l'initiative du programmeur par usage de la procédure onestep.

Exemple de pilotage avec les quatre modes possibles :

Le programme Test_5_avec_Moteur_pas_a_pas_et_4_modes.ino assure le pilotage d'un moteur PAS À PAS avec une boucle qui enchaîne les divers modes SINGLE, DOUBLE, INTERLEAVE et MICROSTEP avec une pause release d'une seconde entre chaque sens de rotation au cours desquels le moteur effectue un tour complet. Le programme est écrit pour des STP-42D144 à 200 pas par tour. On peut ainsi expérimenter les comportements qui en résultent. La vitesse programmée est de 15 tours par minute. Comme on peut l'observer dans le listage partiel du programme donné ci-dessous, le nombre de pas et le mode de commutation sont passés en paramètre dans une procédure gérant un cycle complet.

```
void Faire_un_aller_retour(int NB_PAS, byte MODE) {
  Mon_pas_a_pas.step(NB_PAS, FORWARD, MODE);
  Mon_pas_a_pas.release(); delay(1000);
  Mon_pas_a_pas.step(NB_PAS, BACKWARD, MODE);
  Mon_pas_a_pas.release(); delay(1000);}
```

Exemples d'appels :

```
Faire_un_aller_retour(200, SINGLE);
Faire_un_aller_retour(200, DOUBLE);
Faire_un_aller_retour(400, INTERLEAVE);
Faire_un_aller_retour(200, MICROSTEP);
```

Caractéristiques des moteurs PAS À PAS utilisés en local.

Trois petits moteurs pas à pas de récupération sont disponibles "dans les tiroirs".

Moteurs **STP-42D144** et **17PM-K212-P1T** :

Ces deux moteurs sont strictement équivalents.
Tension nominale : 7Vcc.
Courant : 0,7 A.
Impédance : Deux bobinages de 10Ω. *(Sorties sur 4 fils.)*
Couple : 3,2 cm*kg.
Résolution : 200 pas par tour.
Angle : 1,8° par pas.
Poids : 175 grammes.
Mesures effectuées lorsque des résistances de 10Ω sont ajoutées en séries avec chaque bobinage : 200mA par bobinage sous 5Vcc et 300mA sous 7Vcc.

Moteur **SANYO 103-490-0121** :

Deux bobinages à point milieu.
Résolution : 400 pas par tour.
Angle : 0.9° par pas.
Impédance : 34Ω par bobine élémentaire.
Courant par bobinage :
75mA sous 5Vcc, 100mA sous 7Vcc.
Courant par bobine élémentaire :
147mA sous 5Vcc, 200mA sous 7Vcc.

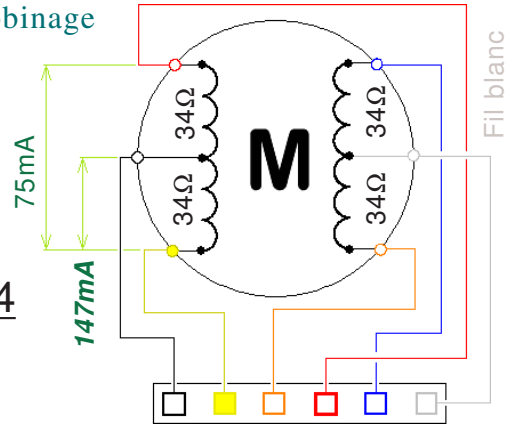


Fig.4

NOTE : Si le sens de rotation du moteur pas à pas à deux bobinages séparés n'est pas celui souhaité, il suffit de permuter les deux fils le branchement d'un bobinage pour l'inverser.

Exemple de pilotage simultané de deux moteurs PAS À PAS :

Le programme Test_6_Deux_pas_a_pas_simultanes réalise le pilotage simultané de deux moteurs pas à pas dans une boucle qui fait appel à la procédure dédiée onestep d'AFMotor.h.
Ce programme est simplifié car les deux moteurs sont de même résolution et effectuent une rotation d'amplitude égale. Mais il reste facile de transposer ce logiciel de base à des moteurs de résolutions différentes ou si la rotation angulaire désirée est différente sur chaque unité.

```
#include <AFMotor.h>

const int DELAI = 1; // Délai effectué entre chaque pas.

AF_Stepper Moteur_1(200, 1); // Moteur 200 pas/tour sur le port #1.
AF_Stepper Moteur_2(200, 2); // Moteur 200 pas/tour sur le port #2.

void setup() { }

void loop() {
  Faire_un_tour_sur_les_deux_moteurs(FORWARD, SINGLE);
  Faire_un_tour_sur_les_deux_moteurs(BACKWARD, SINGLE);
  Faire_un_tour_sur_les_deux_moteurs(FORWARD, DOUBLE);
  Faire_un_tour_sur_les_deux_moteurs(BACKWARD, DOUBLE);
  Faire_un_tour_sur_les_deux_moteurs(FORWARD, INTERLEAVE);
  Faire_un_tour_sur_les_deux_moteurs(BACKWARD, INTERLEAVE); }

void Faire_un_tour_sur_les_deux_moteurs(boolean SENS, byte MODE) {
   int MAX=401; if (MODE=INTERLEAVE) {MAX=801;}
   for (int NB_PAS = 1; NB_PAS < MAX; NB_PAS++)
    {Moteur_1.onestep(SENS, MODE); Moteur_2.onestep(SENS, MODE);
     delay(DELAI);}
   Moteur_1.release(); Moteur_2.release(); delay(1000);}
```

Ce listage est partiel, car le logiciel réel intègre des instructions d'affichage de l'état de déroulement du programme sur la ligne série USB.

Noter que l'instruction onestep() ne permet pas d'utiliser le mode ~~MICROSTEP~~ et **la vitesse n'est plus gérée** par la procédure setSpeed(RPM). Il faut donc imposer un petit délai entre chaque pas pour gérer la vitesse. Dans ce programme la constante DELAI permet de tester la rapidité maximale permise sur les moteurs PAS À PAS connectés à l'interface de puissance d'ADAFRUIT. On peut observer que le MODE est passé en paramètre dans la procédure qui gère la rotation des moteurs, mais également le sens de rotation par le truchement d'un booléen.

REMARQUE : Sur Internet on rencontre souvent des programmes qui remplacent la procédure onestep d'AFMotor.h par l'instructions step :

```
PAS_A_PAS.step(1, FORWARD, SINGLE);
PAS_A_PAS.onestep(FORWARD, SINGLE);
```

} *Instructions équivalentes.*

Ces deux instructions conduisent au même résultat, mais préférer onestep() qui est plus lisible et traduit plus directement sa finalité dans le code du logiciel.

MIXAGE DES TROIS TYPES DE MOTORISATION POSSIBLES.

Comme précisé sur la fiche dédiée au SHIELD d'ADAFRUIT on peut à convenance piloter plusieurs types de moteurs simultanément, la combinatoire étant fonction des modèles connectés. La bibliothèque AFMotor.h ne permet pas de brancher des charges "à sens de rotation unique" comme montré sur la Fig.4 de la fiche relative au Pont en H et circuit intégré L293D. Toutefois, sans faire appel à des programmes plus élaborés, il est déjà possible d'agencer des combinaisons très étoffées. Le programme Test_7_Trois_types_de_M_simultanes.ino en est un exemple dans lequel tous les opérateurs en H du module électronique sont mis à contribution. Un seul SERVOMOTEUR a été intégré à ce petit projet pour alléger "la lisibilité" du programme, mais naturellement en brancher un deuxième reste élémentaire.
ATTENTION : Une alimentation EXTérieure est fortement conseillée sur +VS vu le nombre de moteurs.

Shield ADAFRUIT pour piloter un rapporteur à LASER :

Avec cette application, on aborde la réalisation d'un véritable petit projet dont le but fondamental consiste à maitriser la commande d'un moteur PAS À PAS par utilisation de consignes angulaires exprimées en degrés. Bien que constituant une petite application complète, il sert en réalité d'étude préalable pour envisager la faisabilité d'applications futures plus importantes incluant l'utilisation de deux moteurs et devant impérativement prendre en compte des rotations exprimées en degrés.

Description de l'application.

Le dispositif se résume à un pointeur LASER immobilité sur un petit plateau **1** entrainé en rotation par le moteur PAS À PAS. La rotation du moteur est pilotée par la ligne série USB d'Arduino. On peut ainsi se servir de ce rapporteur électronique soit en sortie, soit en entrée. **En sortie**, il permet de tracer des angles désirés dans l'espace environnant. On dirige le faisceau lumineux vers la première direction de l'angle à repérer. Puis on impose une consigne de rotation pour la valeur angulaire désirée. Le LASER pointe alors la deuxième direction de l'angle recherché. Utilisé en **Entrée**, c'est l'application inverse : On désire mesurer un angle compris entre deux éléments de l'environnement local. On fait pointer le LASER vers le premier élément. On mémorise cette "origine". Puis on le dirige par les consignes USB vers la deuxième référence et l'on demande la position du moteur avec "**P**". La différance angulaire entre les deux positions nous donne la valeur recherchée. Pour dégrossir les pointages, un mini-joystick permet de piloter directement les rotations. Comme à la base cette application reste plus une expérience informatique qu'un projet vraiment désirée, le support du LASER est doublé par un disque gradué **2** et d'un réticule gradué **3** pour permettre un contrôle visuel strictement indépendant des informations délivrées par l'informatique. Comme l'on désire un pointage dont la précision sera d'un degré, un moteur à 400 PAS par tour est utilisé. Ainsi, par l'entremise des demi-PAS il sera possible de positionner au degré prés, voir mieux. Le cercle extérieur du rapporteur est gradué de 0 à ± 180°, mais un cercle intérieur imprimé en rouge exprime les orientations en nombre de PAS. La précision de pointage est alors meilleure que le demi grade. *(Définition : Un tour contient 400 grades.)*

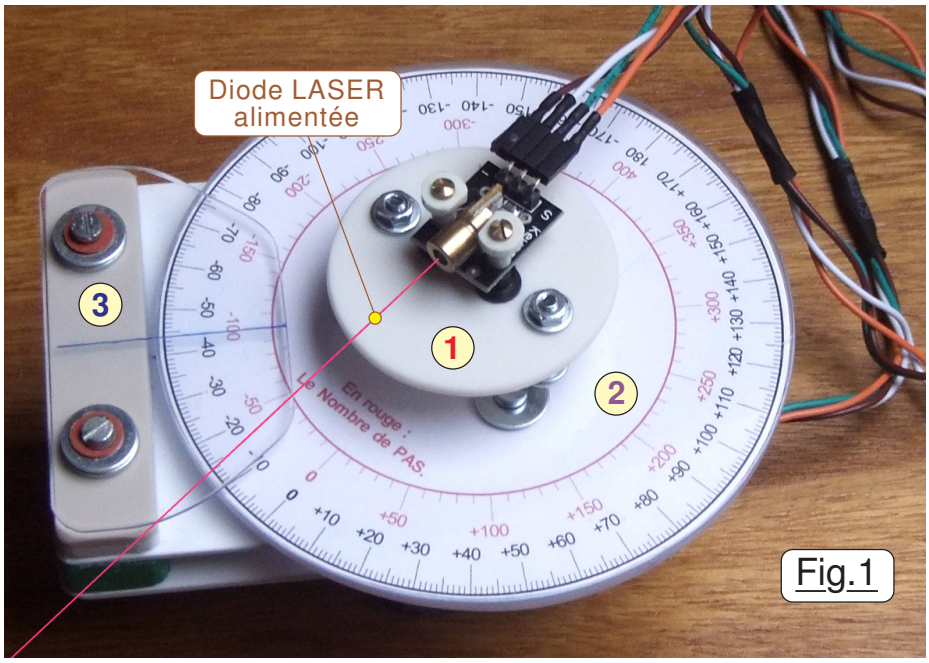


Fig.1

Un peu de bricolage.

La Fig.2 montre le dispositif vu de coté sur lequel le petit circuit imprimé **1** du LASER est débranché. On retrouve facilement le disque gradué **2** ainsi que le réticule **3** solidaire du support **4** immobilisé sur le moteur **7**. Quand le moteur est placé en position verticale pour un balayage LASER horizontal, il porte sur sa face **A**. Son poids est largement suffisant pour procéder aux essais expérimentaux. Mais pour effectuer des mesures angulaires fiables, un statif plus efficace sera impératif pour pallier les rotations parasites résultant des inerties au

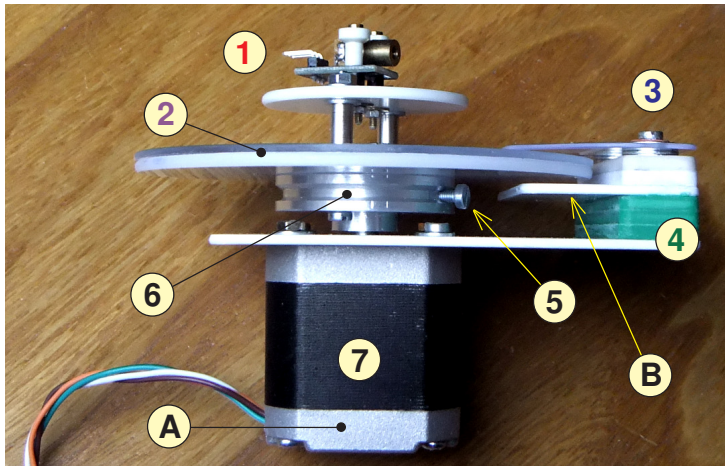


Fig.2

démarrage et à l'arrêt du moteur. Le disque gradué **2** est immobilisé sur le plateau d'accouplement **6**, lui même rendu solidaire de l'arbre moteur au moyen de la vis repérée **5**. On peut observer une plaque **B** solidaire de **4** qui se trouve sous le plateau **2**. Elle présente un trait de repérage comme celui pratiqué sur le réticule gradué **3** et ne sert que si le disque **2** supportant le rapporteur imprimé est remplacé par un disque gradué transparent, ce qui était le cas lors des essais préliminaires. Se construire un modèle de disque gradué avec un quelconque logiciel de dessin tel que celui visible sur la Fig.1 s'avère pour le moins très indigeste. Vous trouverez pour vous éviter cette galère, proposé sur la Fig.3 ci-contre, le dessin à l'échelle 1 du disque sur lequel sont également repérés avec précision les trous de passage des diverses vis de liaison, le trou central servant au centrage sur **6**. *(Le plateau 6 est centré sur l'arbre du moteur 7 par un petit manchon concentrique.)*

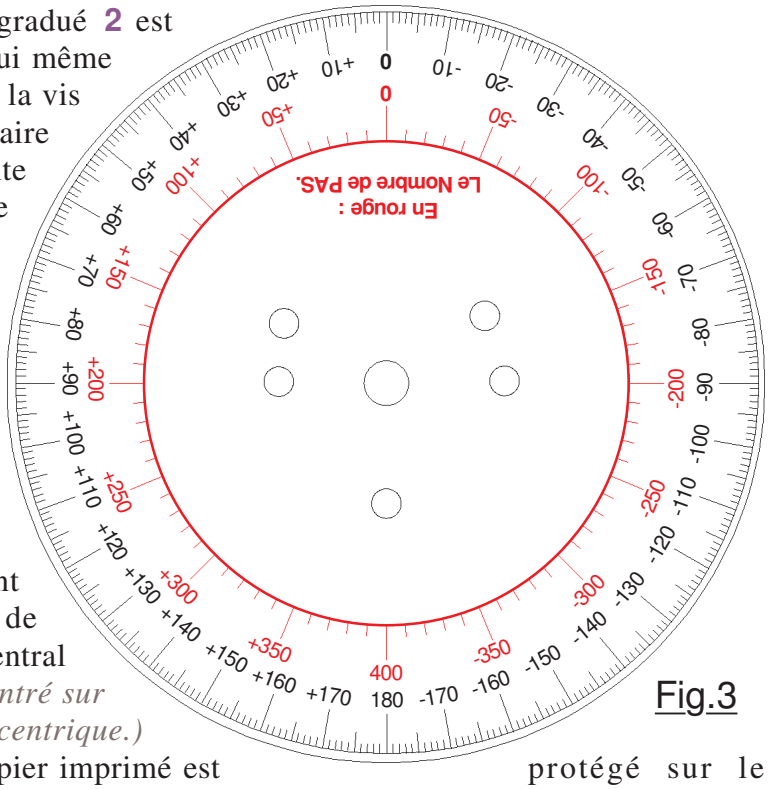


Fig.3

Sur le dispositif expérimental le disque en papier imprimé est protégé sur le dessus par un disque en matière plastique parfaitement transparent et invisible sur la photographie. Mais sa réalisation est laborieuse, donc si vous disposez d'un appareil à plastifier, c'est l'occasion où jamais de le mettre à contribution pour vous éviter ce travail. Par contre, le disque transparent a pour mérite de bien plaquer le disque en papier sur **2** car avec les variation d'hygrométrie de l'air il a tendance à se cintrer.

RAPPEL IMPORTANT : Toute utilisation d'un quelconque pointeur LASER est assujettie impérativement à des mesures de sécurité incontournables. En aucun cas, lors de son utilisation il faut diriger le faisceau lumineux vers les yeux d'un animal ou d'une personne, y compris s'ils sont situées loin de la diode LASER dont la portée est considérable.

Les commandes clavier de ce petit projet envoyés par la ligne USB .

Les diverses commandes possible par l'intermédiaire de la ligne série USB sont résumées dans le tableau de la page 70. Elles ne sont pas listées par ordre alphabétique comme pour la commande "**F**" du logiciel, mais coloriées et classées par types de fonctions. Certaines sont accessoires, mais beaucoup sont issues de l'expérimentation pour aboutir à un rapporteur de mesures angulaires parfaitement opérationnel. Celles dont on pourrait se passer anticipent des projets plus élaborés pouvant exiger jusqu'à trois références de pointage mémorisées. C'est la raison pour laquelle le retour automatique à zéro et la mémorisation d'une **O**rigine RELATIVE sont complétées par la définition possible d'un **J**alon. Dans certains cas d'éclairement, la tâche lumineuse générée par le pointeur LASER sur l'environnement peut s'avérer délicate à repérer. C'est la raison pour laquelle faire clignoter le faisceau lumineux est prévu dans les fonctions par la commande "**C**". On peut oublier la fonction "**G**" qui n'est utile que durant la mise au point du programme pour déterminer la vitesse maximale qu'il ne faut pas dépasser pour être certain que le moteur réalisera tous les PAS et demi-PAS commandés. Comme il y a un risque pour les liaisons filaires souples d'alimentation de la diode LASER, la commande exige exactement trois caractères "G" pour être validée. De plus, le premier peut être à loisir en majuscule ou en minuscule, mais les deux derniers doivent impérativement être frappés en lettres minuscules. Ainsi frapper par erreur un "**G**" ne déclenchera pas la fonction et un message d'erreur est délivré. *(Les trois "G" et la longueur de trois du message sont vérifiés.)*

INITIALISATION DU SYSTÈME.

Les commandes coloriées en bleu sont relatives à l'initialisation du système pour lequel, sur RESET, le moteur est mis en veille. Avant de pouvoir utiliser le rapporteur d'angle lumineux, il faut initialiser l'origine. En premier, on oriente approximativement le secteur gradué sur zéro. À la mise sous tension du moteur

L	Joystick utilisé en vitesse LENTE.
M	Joystick utilisé en vitesse MAXIMALE.
V	Moteur mis en VEILLE. *(On peut facilement orienter le disque gradué à la main.)*
A	Alimente le moteur. *(Le disque gradué se positionne aléatoirement.)*
+	Tourner d'un PAS dans le sens direct. *(Affiner l'origine initiale en manuel.)*
-	Tourner d'un PAS dans le sens rétrograde. *(Affiner l'origine initiale en manuel.)*
Z	Remise à zéro des coordonnées mémorisées sans déplacer de disque gradué.
N	La position actuelle devient la NOUVELLE ORIGINE RELATIVE.
D	Définit les coordonnées du JALON pour la position actuelle.
P	Affiche les coordonnées de la POSITION actuelle.
*****	Retour automatique sur l'origine ABSOLUE. *(Zéro enregistré avec la commande **Z**.)*
O	Déplacement jusqu'à une position angulaire par rapport à l'ORIGINE RELATIVE.
J	Déplacement jusqu'à la position angulaire du JALON mémorisé par "**D**".
B	Déplacer en BUTÉE de 180°. *(Opposé de l'origine absolue définie avec **Z**.)*
/	Allume ou éteint la diode LASER.
C	Mode CLIGNOTEMENT ou allumage permanent. *(Quand la diode LASER est activée)*
U	Utilisation : Affiche toutes les données en PAS ou en DEGRÉS.
E	ÉTAT actuel du système. Précise l'état actuel de diverses options, ainsi que les coordonnées mémorisées pour l'ORIGINE RELATIVE et pour le JALON.
T	**Tourne** de l'angle indiqué en positif ou en négatif. La valeur indiquée sera considérée en DEGRÉS ou en nombre de PAS en fonction de l'option mémorisée "**U**".
R	**Déplace** à la position indiquée en positif ou en négatif par rapport à l'ORIGINE RELATIVE. La position sera considérée en DEGRÉS ou en nombre de PAS fonction de l'option "**U**".
X	**Déplace** à la position indiquée en positif ou en négatif par rapport à l'ORIGINE ABSOLUE. La position sera considérée en DEGRÉS ou en nombre de PAS fonction de l'option "**U**".
F	Rappel de toutes les FONCTIONS pilotées par la ligne USB.
G	Fonction accessoire qui fait effectuer au moteur dix tours à vitesse maximale dans les deux sens. Il faut débrancher la diode LASER. Ne sert qu'en cours de mise au point du programme.

avec la commande "**A**", le disque se verrouille généralement à quelques demi-PAS du réticule zéro. Si l'on a oublié d'effectuer l'approximation à la main, il suffit de faire tourner le rapporteur avec le mini joystick. Sur une amplitude faible de déviation du mini-manche, le disque tourne lentement demi-PAS par demi-PAS. Dès que l'on dépasse notablement la position neutre, le disque tourne à une vitesse fonction de l'option mémorisée avec "**L**" ou avec "**M**". Quand le zéro du disque gradué est proche du réticule, affiner avec précision par l'utilisation des commandes "**+**" et "**-**". *(Cette commande est réalisée par demi-PAS quelle que soit l'option mémorisée à l'aide de "**U**".)* À ce stade, envoyer la commande "**Z**". Le moteur ne change pas de position, mais toutes les origines sont forcées à zéro.

D'une façon générale, dans le tableau :
Les commandes en BLEU sont relatives à l'initialisation du zéro ABSOLU.
Les fonctions en MARRON imposent des déplacements angulaires.
Les directives en VERT imposent ORIGINE RELATIVE et JALON.
Les instructions en ROUGE imposent des retours sur des positions angulaires précises.
Les consignes en VIOLET sont dédiées aux demandes d'informations concernant le système.

Outre les informations que l'on peut obtenir par la ligne série USB et celles disponibles sur le disque gradué, une LED est dédiée au mode VEILLE et s'illumine quand le moteur n'est plus alimenté.
Toute commande envoyée sur la ligne série dont le premier caractère ne correspond pas à un ordre valide engendrera le message d'erreur "Consigne non valide.".
Par ailleurs, l'expérience montre que parfois une commande envoyée sur la ligne série reste sans effet sur le système. Dans le but d'avoir une information supplémentaire concernant la ligne série, toute réception correctement détectée provoque le clignotement rapide de la LED d'Arduino. *(Trois éclairements très courts mais parfaitement visibles.)*

Quelques particularités de ce petit programme.

Fondamentalement ce logiciel **Pointeur_LASER_ANGULAIRE_PAS_a_PAS.ino** a initialement été développé uniquement pour apprendre à piloter un moteurs PAS À PAS avec des commandes angulaires exprimées en DEGRÉS. Ce savoir faire sera impératif pour un projet futur. Tant qu'à créer le petit banc d'essai montré sur la Fig.1 et pousser en détail l'analyse, autant développer une petite application complète. Examinons quelques détails ou procédures spécifiques :

Procédure qui transforme la valeur de consigne exprimée en degrés en nombre de PAS :

Dans le programme, ARGUMENT est un int qui représente la valeur numérique de l'angle dans la consigne et peut être précédé d'un signe. Mais toute valeur extraite s'arrêtera dès le premier caractère non valide rencontré, et en particulier un point décimal. *(Exemple -123.4B retournera -123)* Les angles pris en compte pour la transposition seront forcément des entiers dans la variable globale ARGUMENT.

```
void Transpose_en_Nb_PAS() {
 float VALEUR; int SAV; SAV = ARGUMENT; VALEUR = ARGUMENT / 0.45;
 // ========== Arrondir la valeur calculée. ==========
 ARGUMENT = int(VALEUR);
 if (abs(VALEUR - int(VALEUR)) >= 0.5)
   {if (ARGUMENT >0) {ARGUMENT++;} else {ARGUMENT--;} } }
```

L'instruction VALEUR = ARGUMENT / 0.45; place dans le réel VALEUR le nombre de pas à effectuer, car la constante 0.45 représente l'angle balayé lors d'un demi-PAS pour le moteur utilisé. *(Angle balayé exprimé en DEGRÉS)* Exemple : ARGUMENT = -123 donc VALEUR = -123/0.45 = -273,3333333...
Dans un premier temps, avec ARGUMENT = int(VALEUR); on transforme ce réel en un entier, donc dans notre exemple ARGUMENT vaut maintenant -273.
Puis, l'évaluation VALEUR - int(VALEUR) calcule la valeur décimale, ici -0.3333333...
En prendre la valeur absolue avec abs permet de s'affranchir du signe. Ici l'entité devient +0.3333333...
Enfin, si cette valeur décimale dépasse 0.5 il faut effectuer un pas de plus dans le bon sens. C'est la raison pour laquelle le dernier test if (ARGUMENT >0) impose d'incrémenter ou de décrémenter ARGUMENT.

Procédure qui transforme un nombre de PAS en DEGRÉS :

C'est la réciproque. Mais ici on désire un résultat avec deux décimales, car les positions angulaires exprimées en degrés pour un fonctionnement à 800 PAS par tour n'engendrent pas que des valeurs entières. Du coup la procédure devient élémentaire :

```
void Affiche_un_angle(int ANGLE) {
  if (Mode_DEGRES) { Serial.print((ANGLE * 0.45),2);
                     Serial.println(char(176));}
    else {Serial.println(ANGLE); Serial.println(" PAS.");} }
```

Conformément aux informations données en page 9 la diode LASER est alimentée via une résistance de 100Ω pour en augmenter la durée de vie. L'éclairement qui en résulte est largement suffisant pour développer le programme.

0.45 pour avoir l'angle exprimé en degrés car un PAS corresponde à 360 / 800 = 0,45°.
2 pour imposer l'affichage du réel ANGLE avec deux décimales. C'est suffisant puisque les "débordements" décimaux ne dépasseront jamais deux chiffres significatifs.

Clignotement de la diode LASER :

Il est obtenu par une "bascule" chronométrée dans la boucle de base. La valeur 500 du délai de clignotement définie par #define est fonction de la rapidité de la boucle de base et en dépend directement.

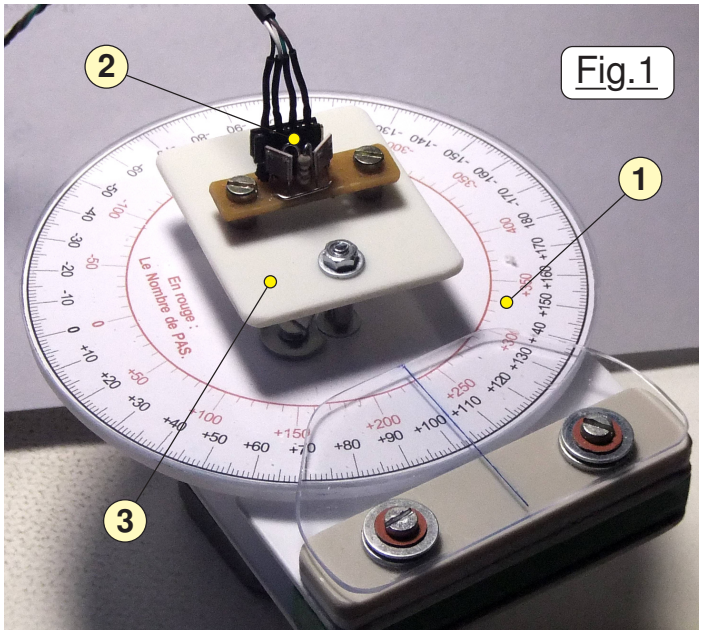
```
#define Delai_clignotement_LASER 500 // Compteur de clignotement.
int Compteur_du_delai_de_clignotement_LASER;

void Fait_clignoter_la_diode_LASER() {
 if (Compteur_du_delai_de_clignotement_LASER < Delai_clignotement_LASER)
   {Compteur_du_delai_de_clignotement_LASER++;}
    else {Compteur_du_delai_de_clignotement_LASER = 0; Inverser_alimentation_LASER();} }

void inverser_alimentation_LASER() {LUMIERE = !LUMIERE;
    if (LUMIERE) {digitalWrite(Pilotage_LASER, HIGH);}
    else {digitalWrite(Pilotage_LASER, LOW); } }
```

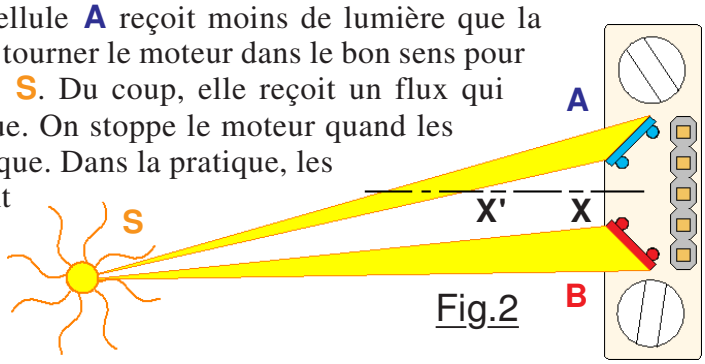
Asservissement pour piloter la poursuite d'une source lumineuse :

Analogue à la petite application de la page 68, on reprend une bonne partie de son code, tant pour les dialogues sur la ligne série USB que pour la gestion du moteur PAS À PAS. Par contre, le module électronique de la diode LASER qui fonctionnait "en sortie d'information" est remplacé par une électronique élémentaire qui fournit à Arduino des données concernant l'environnement. Un petit circuit imprimé permet de mettre en œuvre deux cellules photorésistantes pour prendre en compte l'environnement lumineux du dispositif. L'asservissement consiste à engendrer un suivi automatique de la source lumineuse prépondérante en orientant le capteur électronique dans sa direction. La Fig.1 présente l'aspect matériel de cette petite application, pour laquelle on retrouve en **1** le disque graduéentrainé en rotation par un moteur PAS À PAS, mais en **2** est fixé nouveau petit module électronique spécifique à cette expérimentation, avec en **3** son petit support adapté pour sa liaison avec **1**.



Principe utilisé pour capter la lumière.

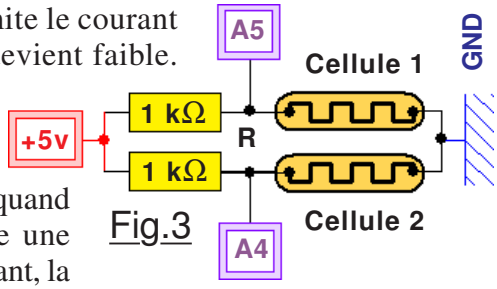
Déterminer la direction d'où arrive une source lumineuse prépondérante s'avère en fin de compte plus simple que ce qui avait été envisagé lors des études préliminaires. Il suffit de disposer sur un petit circuit imprimé, comme montré sur la Fig.2, deux cellules photoélectriques identiques et orientées à 90° degrés l'une de l'autre. Considérons la source lumineuse prépondérante **S**. Comme le système n'est pas dirigé strictement vers elle, la cellule **A** reçoit moins de lumière que la cellule **B**. Pour asservir l'ensemble, il suffit de faire tourner le moteur dans le bon sens pour amener la cellule la moins éclairée vers la source **S**. Du coup, elle reçoit un flux qui augmente, alors que simultanément sur **B** il diminue. On stoppe le moteur quand les deux cellules sont soumises à un éclairement identique. Dans la pratique, les deux cellules n'ont pas des caractéristiques strictement identiques, la direction adoptée vers le cible ne sera pas strictement celle de l'axe de symétrie **X'X** mais c'est sans importance. Par contre, si l'on ne veut pas que le disque n'oscille en permanence autour d'une zone dirigée vers la source lumineuse **S**, il faut introduire dans le programme une tolérance, c'est à dire accepter une différence de valeur sur les cellules qui n'engendrera pas une correction de suivi.



Un trois fois rien d'électronique.

(Voir l'encadré bleu donné en page 81)

L'électronique du capteur de lumière destiné à cette petite application et montrée sur le schéma de la Fig.3 se résume à peu de chose. Visible sur la Fig.4 ce circuit imprimé n'utilise que deux résistances de limitation de courant et deux cellules photorésistantes. Un grand nombre de modèles étaient disponibles, tous plus ou moins équivalents. J'ai choisi deux cellules pour leurs petites dimensions. La référence n'est pas mentionnée sur ces composants, mais tout modèle de type LDR conviendra, les caractéristiques ne sont pas critiques à partir du moment où vous utilisez deux éléments identiques. Pour chaque branche, la résistance **R** porte à **+5Vcc** la tension sur l'entrée analogique, et limite le courant si la résistance introduite par la **Cellule** entre l'entrée et la masse devient faible. Avec ce type de composant, la résistance interne diminue quand le flux lumineux augmente, la résistance étant maximale dans le noir complet. Plus la **Cellule** est éclairée, plus la tension sur l'entrée analogique diminue. Avec les composants utilisés sur cette maquette, quand il fait très sombre la tension monte à environ +3,5Vcc qui donne une conversion numérique qui s'approche de 550. Sous un éclairage important, la tension chute vers +0,2Vcc avec une conversion numérique voisine de 90. Comme c'est la différence



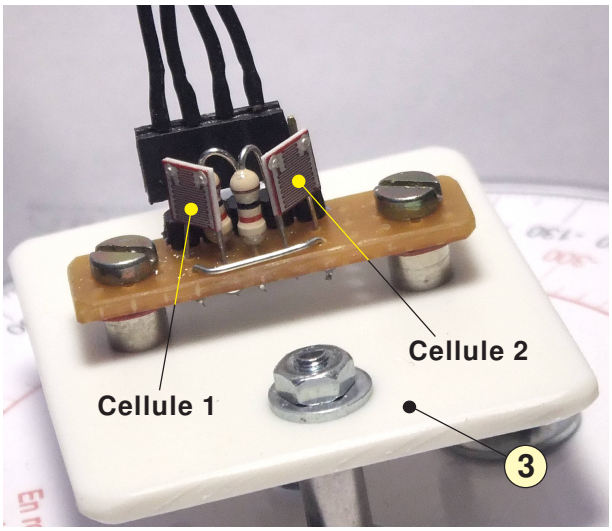
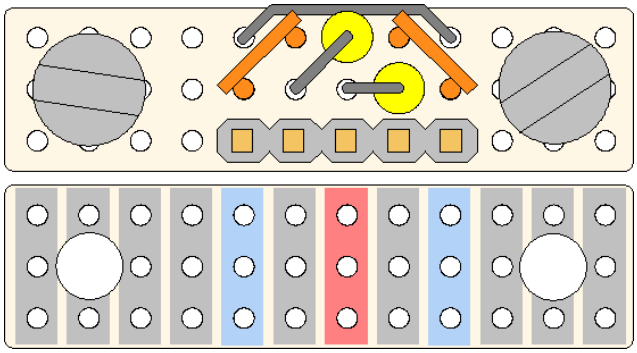


Fig.4

Dessin du tout petit circuit imprimé.



entre les valeurs fournies par les deux cellules qui sera prise en compte, il n'est pas important du tout d'employer des composants de même modèle. Toute cellule photo résistante de caractéristiques ordinaires conviendra. C'est juste à titre d'information que les tensions mesurées sont indiquées ici, pour mieux comprendre l'agencement du capteur et son principe de fonctionnement.

L'aspect logiciel.

Dérivé du projet précédent, **Asservissement_en_lumirere.ino** émule les fonctions décrites dans le tableau donné ci-dessous, dont la majorité est directement issue du code mis au point pour gérer le pointeur LASER. Seules les procédures qui permettent d'initialiser les orientations de base sont conservées. La seule nouvelle fonction est invoquée par "**S**". Elle valide ou non le mode de suivi automatique orientant le capteur vers la source de lumière prépondérante dans l'environnement immédiat du module expérimental.

Comportement du programme.

Sur un RESET le moteur est en mode VEILLE, la vitesse LENTE et le mode de Suivi lumineux NON activé. Les informations affichées seront systématiquement en degrés pour les angles. Contrairement au programme élaboré pour le pointeur LASER, les vitesses lentes imposée par "**L**" ou rapides consignée avec "**M**" sont prises en compte aussi bien pour les mouvements provoqués à l'aide du joystick que ceux résultant du suivi lumineux. Par contre, les rotations automatiques déclenchées par "*****" et "**J**" se font à vitesse maximale quelle que soit l'option de vitesse mémorisée. Si le système est en capture optique lorsque l'on transmet les consignes "*****" et "**J**", ces dernières sont bien prises en compte, mais dès que l'angle cible est atteint, on sort de la procédure et le disque tourne immédiatement pour se réorienter vers la clarté. Durant une rotation de suivi lumineux, une consigne "**L**" ou surtout "**M**" est immédiatement prise en compte même si la "position d'équilibre optique" n'est pas encore atteinte. Il y a contradiction entre les consignes de mouvement envoyées à l'aide du Joystick et le suivi lumineux automatique. Donc, si l'on désire orienter en manuel, il faut désactiver le suivi optique avec "**S**" avant d'utiliser le mini-manche. Pour ne pas que le moteur ne soit engagé dans de permanentes corrections autour de l'orientation idoine, une "zone neutre" est instaurée, c'est à dire qu'il faut que la différence de numérisation des signaux lumineux dépasse un seuil avant que le système ne réagisse pour corriger l'orientation. *(Valeur du seuil trouvée expérimentalement.)*

L	Joystick et Poursuite lumineuse en vitesse LENTE.
M	Joystick et Poursuite lumineuse en vitesse MAXIMALE.
V	Moteur mis en VEILLE. *(On peut facilement orienter le disque gradué à la main.)*
A	Alimente le moteur. *(Le disque gradué se positionne aléatoirement.)*
+	Tourner d'un PAS dans le sens direct. *(Affiner l'origine initiale en manuel.)*
-	Tourner d'un PAS dans le sens rétrograde. *(Affiner l'origine initiale en manuel.)*
Z	Remise à zéro des coordonnées mémorisées sans déplacer de disque gradué.
N	La position actuelle devient le nouveau JALON.
P	Affiche les coordonnées de la POSITION actuelle.
*****	Retour automatique sur l'origine ABSOLUE. *(Zéro enregistré avec la commande **Z**.)*
J	Déplacement jusqu'à la position angulaire du JALON mémorisé par "**N**".
E	ÉTAT actuel du système.
F	Rappel de toutes les FONCTIONS pilotées par la ligne USB. *(Ce Menu.)*
S	Suivi de la source lumineuse prépondérante OUI / NON.

Shield ADAFRUIT pour piloter une structure "en joints de Cardan" :

Anticipant le développement d'un projet plus ambitieux, *(Plateforme gyrostabilisée)* ce programme constitue la première étape du pilotage de la motorisation de deux articulations croisées de type "joints de Cardan" équipées de moteurs PAS À PAS. Des servomoteurs ont été initialement envisagés, mais les tentatives pour valider le concept ont rapidement démontré certaines difficultés rencontrées dans l'asservissement en boucle fermée avec cette technologie. *(Oscillations autour du point de stabilisation)* De plus la rotation angulaire possible est limitée à 180° pour les servomoteurs les plus courants.

Réalisation mécanique de la structure articulée.

Comme il s'agit au final d'une modeste expérimentation de loisir, et non de développer une centrale inertielle totalement opérationnelle, les frais initiaux ont été minimisés par l'utilisation de moteurs déjà disponibles. Comme montré sur la Fig.1 la plateforme se résume à une simple petite équerre métallique **2** sur laquelle seront fixés les modules électroniques **1** affecté aux projets futurs. Cette équerre est entrainée autour de l'axe de **TANGAGE X'X** par le moteur PAS À PAS **3**. Ce moteur est lui-même immobilisé sur un étrier qui va l'orienter en rotation autour de l'axe **Y'Y**. C'est le moteur PAS À PAS **4** qui entraine en **ROULIS** le support de **3**. Les deux axes **X'X** et **Y'Y** sont disposés à angle droit pour aboutir à deux articulations agencées en "joints de Cardan". Les axes de rotation **X'X** et **Y'Y** sont désignés pour correspondre également à ceux du système de repérage d'un capteur d'accélération électronique qui sera ultérieurement placé en **1**. Le moteur affecté au **TANGAGE** est le modèle **SANYO 103-490-0121** à 400 pas par tour. Celui assurant les mouvements de **ROULIS** et montré sur la Fig.2 était initialement le **17PM-K212-P1T** à 200 pas par tour. Les mouvements en **ROULIS** étaient exagérément vibrés et moins fins que ceux relatifs au mouvement de **TANGAGE**. Pour aboutir à une solution bien plus homogène et générant des mouvements de **ROULIS** plus fluides, l'achat d'un deuxième moteur présentant 400 pas par tour a été effectué. Ce moteur est un modèle 42BYGH44958A. Il est plus long que l'ancien montré sur la Fig.2 mais ne compromet en rien la stabilité du bâtit. Comme le couple moteur présenté par ce modèle est plus important, les rondelles servant d'entretoise ne sont plus utiles et le moteur **3** peut être directement placé sur l'étrier. *(Elles servaient à amener sur X'X le centre de gravité de l'ensemble mobile en roulis)* L'image de la Fig.2 montre la concrétisation matérielle en vue de dessus, les axes de rotation étant en orientation "neutre". Noter que les liaisons filaires sont réalisées avec des conducteurs très fins torsadés pour obtenir des faisceaux très souples. Un câblage réalisé

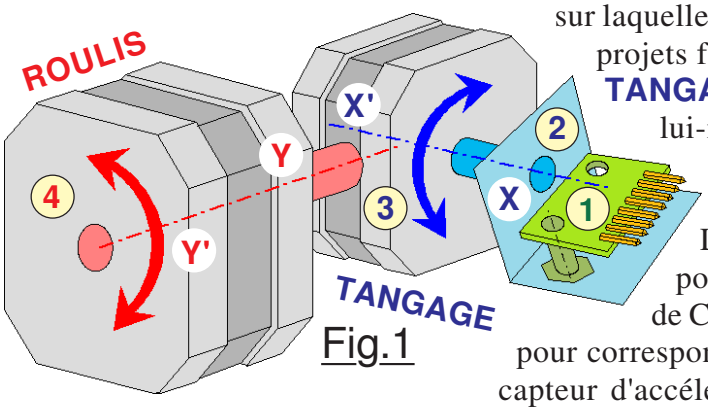


Fig.1

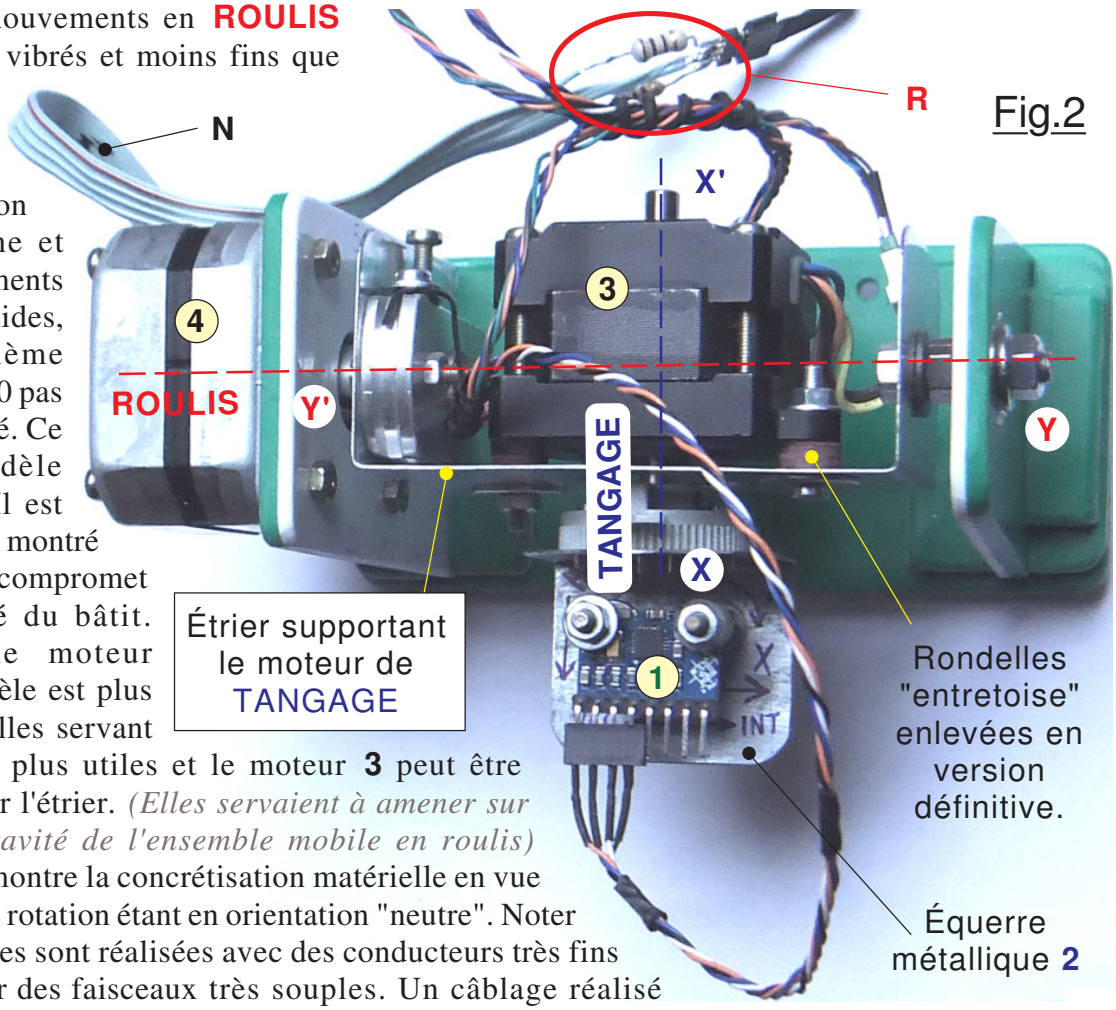


Fig.2

avec des fils électriques ordinaires serait trop rigide par rapport au couple magnétique relativement faible fourni par les deux moteurs PAS À PAS. *(Très sous-alimentés pour en minimiser l'échauffement et surtout la consommation)* La nappe électrique qui assure le raccordement au moteur **4** est d'origine et moins souple, mais ce moteur est immobilisé sur le support fixe, la rigidité des fils n'est donc pas un inconvénient. Les résistances **R** de 22Ω sont mises en série avec les enroulements du moteur **4** pour en baisser la consommation, le couple qui en résulte restant suffisant pour l'application envisagée. Sur la Fig.2 le disque gradué visible sur la Fig.3 et permettant de vérifier le comportement de l'axe de **ROULIS** n'est pas en place. La zone rouge est la plage d'interdiction pour laquelle l'équerre supportant le module **1** peut venir en butée avec le support vert et blanc. La position zéro repérée en bleu correspond à l'orientation initiale, l'étrier est alors en position verticale, l'axe de **TANGAGE** étant horizontal.

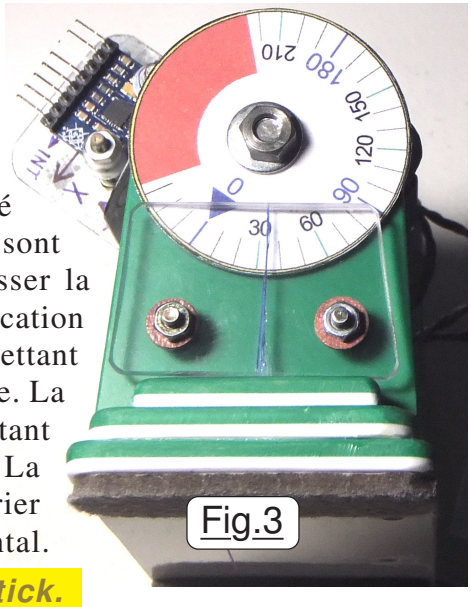


Fig.3

Première expérience : Pilotage avec un mini joystick.

L'expérimentation de base du dispositif réalisé et montré sur la Fig.2 se contente de piloter les deux moteurs avec une consigne manuelle issue du mini-joystick montré sur la Fig.4 et dont la mise en œuvre a été abordée en page 13 de ce document. Avec le code qui y est proposé, les valeurs numériques retournées pour int VarX et int VarY varient globalement entre 0 et 1023. Pour chaque axe trois vitesses de rotation sont implémentées pour expérimenter les déplacements lents et les mouvements rapides. Compte tenu du comportement du mini-joystick, les plages de valeurs montrées sur la Fig.5 ont été adoptées. La rapidité des mouvements est fonction de l'amplitude de rotation du mini-manche. Le programme développé

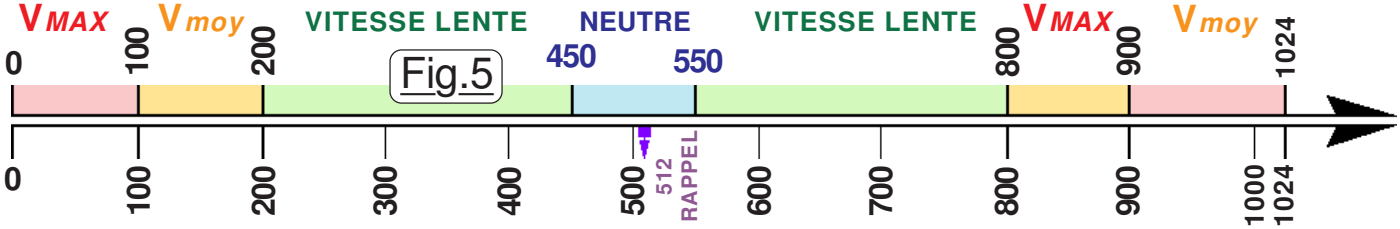


Fig.5

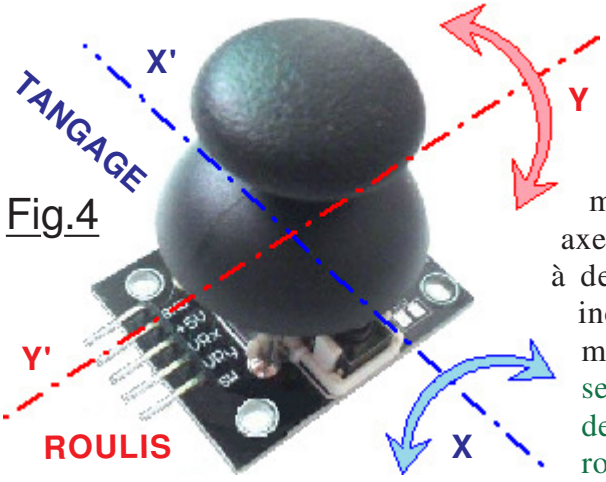


Fig.4

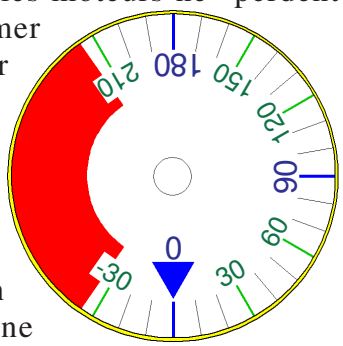
qui anime la platine Arduino pour cette expérimentation est Test_8_Pilotage_Cardan_XX_et_YY.ino et traite la vitesse par des instructions setSpeed(n) au lieu de la procédure delay() trop souvent utilisée à la place.

Compte tenu de l'orientation pratique et de la façon dont le mini-manche est tenu en main pour piloter le système, les axes sont programmés comme montrés sur la Fig.4 pour aboutir à des "mouvements image directs". C'est à dire que toute inclinaison du bouton du joystick engendre une rotation sur le même axe et du même coté. N'oublions pas que si l'un des sens observé est inversé il suffit de permuter deux fils sur l'un des bobinages du moteur pour rétablir la conformité de la rotation. Ce programme permet en particulier de tester les vitesses maximales permises compte tenu des moteurs utilisés et des résistances mécaniques rencontrées. On trouve respectivement setSpeed(30) pour le **ROULIS** et setSpeed(60) pour le **TANGAGE**.

CARACTÉRISTIQUES DU MOTEUR 42BYGH44958A.
Chaque enroulement présente une résistance de 40Ω. Comme il est plus volumineux que le **SANYO 103-490-0121** son couple nominal est supérieur. On peut ainsi diminuer la consommation électrique sans perte "d'accrochage" à des vitesses relativement élevées. Les résistances mises en série ont été choisies à une valeur de 22Ω. Les fils sont vert et orange pour un bobinage, et blanc et marron pour l'autre. Avec 400 PAS par tour, en mode demi PAS on dispose ainsi de 800 positions pour 360°.

Deuxième expérience : Pilotage des axes via la ligne série USB d'Arduino.

C'est la suite logique qui permet par des caractères de commande envoyés sur la ligne série USB de piloter les deux axes numériquement et ainsi déterminer avec précision les mouvements limites autorisés, vérifier qu'à vitesse de rotation maximale les moteurs ne "perdent pas des pas", mettre en place le mode veille, l'utilisation de DEL pour informer de la configuration du système. Globalement ce programme consiste à traiter sur les deux axes ce qui a été fait pour le petit logiciel décrit en page 68 de ce document. Pour faciliter la mise au point du code de **CARDANS_Pilotage_avec_la_ligne_USB_sur_XX_et_YY.ino**, rien n'interdit d'utiliser encore le petit joystick pour faire tourner les axes en mode manuel. Le disque visible sur la Fig.3 est représenté ci-contre à l'échelle 1 pour ceux qui le désireraient, bien que les limites de rotation en **ROULIS** repérées par la zone rouge seront probablement différentes sur une autre maquette. Naturellement on va retrouver globalement des fonctions tout à fait analogues à celles du programme développé pour le pointeur LASER, mais avec pilotage sur deux axes au lieu d'un seul. Le tableau donné ci-dessous résume par ordre alphabétique les diverses commandes utilisables. On retrouve pour les couleurs les mêmes conventions que celles données en Page 70 :



Les commandes en BLEU sont relatives à l'initialisation du zéro ABSOLU.
Les fonctions en MARRON imposent des déplacements angulaires.
Les directives en VERT imposent ORIGINE RELATIVE et JALON.
Les instructions en ROUGE imposent des retours sur des positions angulaires précises.
Les consignes en VIOLET sont dédiées aux requêtes d'informations concernant le système.

A	Alimente les moteurs. *(Le disque gradué se positionne aléatoirement.)*
C	Rappel de toutes les CONSIGNES pilotées par la ligne USB.
D	Définit sur les deux axes les coordonnées du JALON pour la position actuelle.
E	ÉTAT actuel du système. Précise l'état actuel des diverses options, ainsi que les coordonnées mémorisées pour l'ORIGINE RELATIVE et pour le JALON.
G	Gigoter : Effectue quatre alternances de butée en butée à vitesse maximale pour vérifier que les moteurs ne "perdent pas des PAS".
J	Déplacement jusqu'à la position angulaire du JALON mémorisé par "**D**".
L	Joystick utilisé en vitesse LENTE.
M	Joystick utilisé en vitesse MAXIMALE.
N	La position actuelle devient la NOUVELLE ORIGINE RELATIVE.
O	Déplacement jusqu'à l'ORIGINE RELATIVE mémorisé par "**N**".
P	Affiche les orientations de la POSITION actuelle.
R	Tourne en ROULIS d'une valeur positive ou en négative en DEGRÉS ou en nombre de PAS en fonction de l'option "**U**" adoptée. *(En degrés par défaut.)*
T	Tourne en TANGAGE d'une valeur positive ou en négative en DEGRÉS ou en nombre de PAS en fonction de l'option "**U**" adoptée. *(En degrés par défaut.)*
U	Utilisation : Affiche toutes les données en PAS ou en DEGRÉS.
V	Moteurs mis en VEILLE. *(On peut facilement orienter les deux axes à la main.)*
X	Positionne le ROULIS en positif ou en négatif par rapport à l'ORIGINE RELATIVE. La position sera considérée en DEGRÉS ou en nombre de PAS fonction de l'option "**U**".
Y	Positionne le TANGAGE en positif ou en négatif par rapport à l'ORIGINE RELATIVE. La position sera considérée en DEGRÉS ou en nombre de PAS fonction de l'option "**U**".
Z	Remise à zéro des coordonnées mémorisées sans faire tourner les moteurs.
+	Tourner d'un PAS en ROULIS dans le sens direct. *(Affiner l'origine initiale en manuel.)*
-	Tourner d'un PAS en ROULIS dans le sens rétrograde. *(Affiner l'origine initiale.)*
<	Tourner d'un PAS en TANGAGE dans le sens direct. *(Affiner l'origine initiale.)*
W	Tourner d'un PAS en TANGAGE dans le sens rétrograde. *(Affiner l'origine initiale.)*
*****	Retour automatique sur l'origine ABSOLUE. *(Zéro enregistré avec la commande **Z**.)*

QUELQUES PARTICULARITÉS DE CE PROGRAMME :

Concrètement, le système de cardans croisés motorisés n'est qu'un vecteur pour orienter un module électronique quelconque placé sur la "plateforme". Ne rien y placer rend stérile ce système. La conception du programme doit impérativement prendre en compte "la charge utile potentielle" qui transformera cette expérience en une vraie application. Il importe en conséquence de tenir compte des futurs développements envisagés pour répartir judicieusement les broches d'E/S dédiées à la gestion proprement dite des moteurs et des systèmes de visualisation. *(DEL, afficheurs etc.)*

- Comme on peut le constater sur le tableau donné en bas de page, l'affectation des E/S prévoit le dialogue avec des modules électroniques complémentaires par le truchement de la SPI. Dans cette optique les broches analogiques **A4** et **A5** ne sont pas utilisées par ce programme. Les deux broches binaires **D0** et **D1** sont naturellement ***dédiées au dialogue sur la voie série USB***. La gestion des moteurs PAS À PAS monopolise les broches repérées en bleu dans le tableau. **D9** et **D10** restent disponibles si aucun servomoteur n'est déclaré dans le futur programme.
- Compte tenu de la fragilité potentielle du module électronique qui sera dédié à une application spécifique, et en particulier le lien souple de raccordement et son connecteur, il importe impérativement de protéger le système contre toute rotation forcée "en butée mécanique". Dans ce but, des limites angulaires sont définies par les constantes RlimiteMAX, RlimiteMIN, *(Qui délimitent la zone ROUGE)* TlimiteMAX et TlimiteMIN. Une sécurité logicielle empêche de dépasser ces angles de rotation quel que soit le moyen utilisé pour imposer un mouvement.
- Une LED prévient quand on approche des butées et s'allume à une anticipation de la butée définie par la constante MargeLimite 15. Un message d'alerte spécifique est généré quand un mouvement est refusé car une butée est atteinte et précise laquelle.
- La commande "**G**" a montré que pour les vitesses maximales il ne faut pas dépasser 30 en **ROULIS** et 50 en **TANGAGE**. Le couple moteur étant relativement faible en **TANGAGE**, une liaison filaire pas très souple a été utilisée sur le module électronique "embarqué" pour déterminer une valeur fiable.
- La commande "**W**" peut étonner, car on s'attendrait à "**>**" pour raison d'homogénéité avec "**<**". Mais à l'usage, avoir à utiliser **[MAJ]** est peu commode et "**W**" est juste à coté de "**<**".

AFFECTATION DES ENTRÉES / SORTIES

Broche	Utilisation
D0	Réservée en standard pour la liaison série USB. (RX)
D1	Réservée en standard pour la liaison série USB. (TX)
D2	*Disponible F.E.*
D3	Pilotage V3bis & 4bis du module électronique Shield ADAFRUIT.
D4	Pilotage SHIFT du module électronique Shield ADAFRUIT.
D5	Pilotage V1 & 2 du module électronique Shield ADAFRUIT.
D6	Pilotage V3 & 4 du module électronique Shield ADAFRUIT.
D7	Pilotage OUTPUT du module électronique Shield ADAFRUIT.
D8	Pilotage INPUT du module électronique Shield ADAFRUIT.
D9	*Disponible F.E. si pas de servomoteur déclaré dans le programme.*
D10	*Disponible F.E. si pas de servomoteur déclaré dans le programme.*
D11	Pilotage V1bis & 2bis du module électronique Shield ADAFRUIT.
D12	Pilotage LATCH du module électronique Shield ADAFRUIT.
D13	LED Arduino utilisée pour accuser réception d'une consigne USB.
A0	Entrée analogique JOYSTICK pour le TANGAGE.
A1	Entrée analogique JOYSTICK pour le ROULIS.
A2	(16) Sortie binaire pour piloter la LED signalant le mode VEILLE.
A3	(17) Sortie binaire pour piloter la LED signalant l'approche des butées.
A4	*Réservée pour F.E. pour gérer la SPI. (Ligne I2C.)*
A5	*Réservée pour F.E. pour gérer la SPI. (Ligne I2C.)*

En bleu les broches utilisées par le module électronique Shield ADAFRUIT.

Première application : Théodolite à LASER.

Sous ce vocable un peu "savant" se cache une application élémentaire. Comme schématisé sur la Fig.1 un théodolite est un instrument astronomique ou de géomètre pour pointer des directions précises en **Azimut** et en **Hauteur**, l'axe **Y'Y** étant préalablement ajusté dans une orientation verticale parfaite au lieu d'observation considéré. En astronomie l'axe **Y'Y** est généralement orienté Nord/Sud pour constituer une monture équatoriale et le système de visée est purement optique. Pour une application simplifiée de "mesures topographiques" on peut remplacer l'optique de pointage par un faisceau LASER. On retrouve l'application décrite en page 68 mais avec une visée LASER possible à la fois horizontalement et en hauteur. Il suffit comme montré sur la Fig.2 de placer en "charge utile" le petit circuit imprimé de la diode LASER et de positionner comme précisé sur la Fig.3 l'axe de **ROULIS** verticalement. Compte tenu de la masse inerte du moteur **4** de nos articulations en cardans motorisés, la stabilité est largement suffisante. Il serait possible de compléter le tout d'un socle plus stable muni d'un niveau à bulle pour en faire un vrai théodolite, mais en ce qui me concerne, cette application n'est qu'une petite expérience ludique de plus, alors le matériel est simplifié au maximum. Seul le petit support adaptateur qui était visible sur la Fig.4 de la page 73 a été un peu modifié pour pouvoir être solidarisé avec la petite équerre métallique. Deux cotés ont été taillés en biais pour dégager au maximum cet adaptateur. Deux trous ont été percés pour passer les minuscules vis qui maintiennent le module LASER. Notre théodolite à LASER est opérationnel. La Fig.4 présente notre "maquette" en position verticale, la diode LASER étant allumée en mode éclairage constant. Les branchements électriques sont strictement identiques à ceux précisés en page 77 avec en plus l'alimentation du petit module de la diode LASER par la broche de sortie binaire **D9**. Une résistance de 100Ω est intercalée entre **D9** et le +5Vcc du module LASER pour optimiser la consommation de l'ensemble et augmenter la durée de vie du composant optronique actif. Le programme **THEODOLITE_LASER.ino** anime cette petite application et l'on observe, en consultant le tableau des commandes clavier via la ligne série USB, *(En haut de la page 79)* qu'un grand nombre de fonctions du programme précédent sont présentes. Toutefois, les vocables de **ROULIS** et de **TANGAGE** sont remplacés par HORIZONTAL et VERTICAL. S'ajoutent aux fonctions précédentes celles qui permettent de gérer la diode LASER, c'est à dire les directives déclenchées par "**C**" et "/". Le tableau des fonctions est toujours organisé par ordre alphabétique des caractères et respecte toujours les conventions de couleur suivantes :

Fig.1

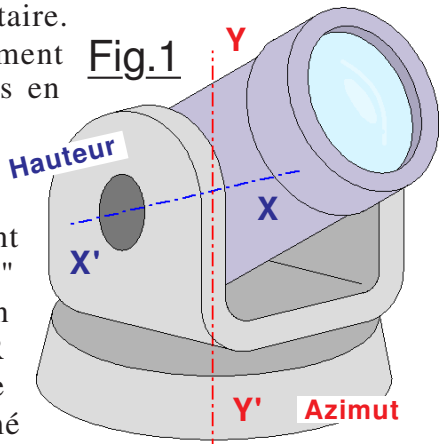


Fig.2

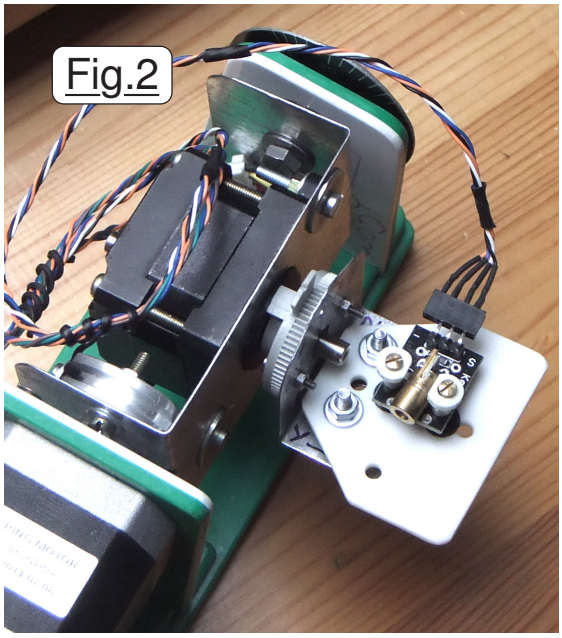
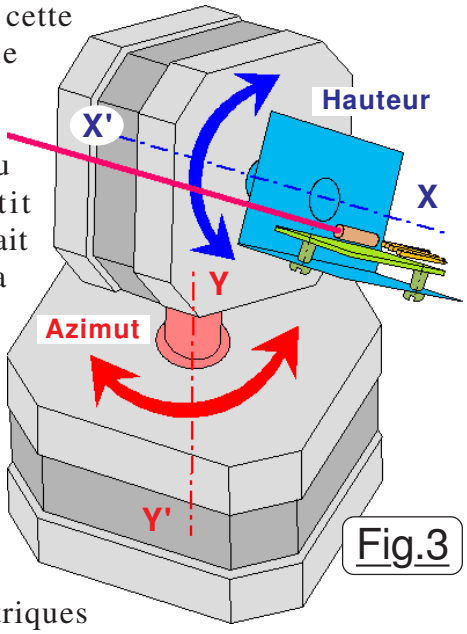


Fig.3

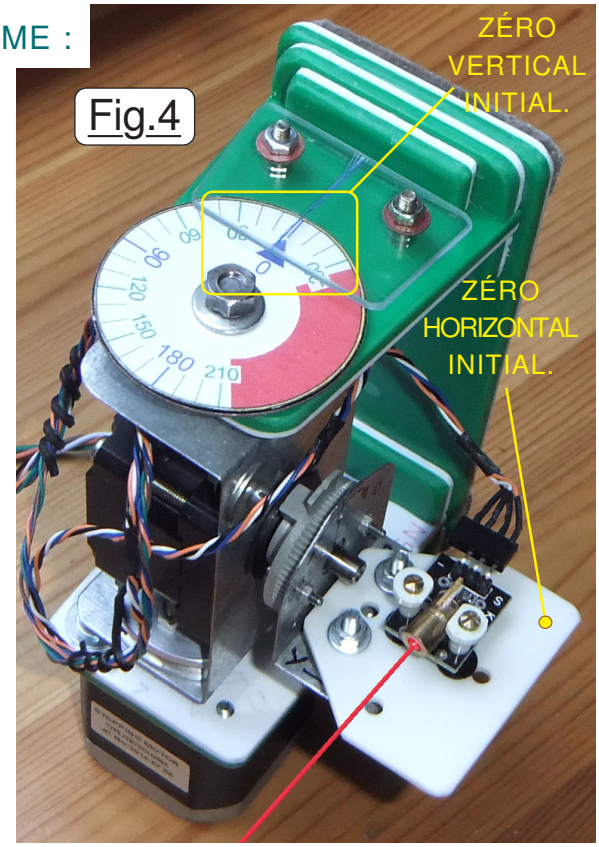


Les commandes en BLEU sont relatives à l'initialisation du zéro ABSOLU.
Les fonctions en MARRON imposent des déplacements angulaires.
Les directives en VERT imposent ORIGINE RELATIVE et JALON.
Les instructions en ROUGE imposent des retours sur des positions angulaires précises.
Les consignes en VIOLET sont dédiées aux requêtes d'informations concernant le système.
Les consignes en noir sont relatives à la gestion de la diode LASER. *(Et à la directive "U")*

A	Alimente les moteurs. *(Le disque gradué se positionne aléatoirement.)*
C	Mode CLIGNOTEMENT OUI/NON pour la diode LASER.
D	Définit sur les deux axes les coordonnées du JALON pour la position actuelle.
E	ÉTAT actuel du système. Précise l'état actuel des diverses options, ainsi que les coordonnées mémorisées pour l'ORIGINE RELATIVE et pour le JALON.
F	Rappel de toutes les FONCTIONS pilotées par la ligne USB.
H	Tourne HORIZONTALEMENT d'une valeur positive ou en négative en DEGRÉS ou en nombre de PAS en fonction de l'option "**U**" adoptée. *(En degrés par défaut.)*
J	Déplacement jusqu'à la position angulaire du JALON mémorisé par "**D**".
L	Joystick utilisé en vitesse LENTE.
M	Joystick utilisé en vitesse MAXIMALE.
N	La position actuelle devient la NOUVELLE ORIGINE RELATIVE.
O	Déplacement jusqu'à l'ORIGINE RELATIVE mémorisé par "**N**".
P	Affiche les orientations de la POSITION actuelle.
R	Moteurs mis au REPOS. *(On peut facilement orienter les deux axes à la main.)*
U	Utilisation : Affiche toutes les données en PAS ou en DEGRÉS.
V	Tourne en VERTICALEMENT d'une valeur positive ou en négative en DEGRÉS ou en nombre de PAS en fonction de l'option "**U**" adoptée. *(En degrés par défaut.)*
X	Positionne horizontalement en positif ou en négatif par rapport à l'ORIGINE RELATIVE. La position sera considérée en DEGRÉS ou en nombre de PAS fonction de l'option "**U**".
Y	Positionne verticalement en positif ou en négatif par rapport à l'ORIGINE RELATIVE. La position sera considérée en DEGRÉS ou en nombre de PAS fonction de l'option "**U**".
Z	Remise à zéro des coordonnées mémorisées sans faire tourner les moteurs.
+	Tourner d'un PAS horizontalement dans le sens direct. *(Affiner l'origine initiale.)*
-	Tourner d'un PAS horizontalement dans le sens rétrograde. *(Affiner l'origine initiale.)*
<	Tourner d'un PAS verticalement dans le sens direct. *(Affiner l'origine initiale.)*
W	Tourner d'un PAS verticalement dans le sens rétrograde. *(Affiner l'origine initiale.)*
/	Diode LASER allumée OUI/NON.
*****	Retour automatique sur l'origine ABSOLUE. *(Zéro enregistré avec la commande **Z**.)*

QUELQUES PARTICULARITÉS DE CE PROGRAMME :

- Les amplitudes de rotation sont différentes, et les valeurs des butées logicielles RlimiteMAX, RlimiteMIN, TlimiteMAX et TlimiteMIN ont été modifiées pour restreindre les deux mouvements car le support ajouté à l'équerre déborde davantage. Avec les valeurs du programme précédent certains mouvements allaient en butée mécanique et les moteurs "perdaient" des PAS.
- Les butées logicielles ne protègeront correctement le matériel que si la configuration adoptée est celle montrée sur la Fig.4 au moment de l'initialisation du ZÉRO de référence avec la commande "**C**". C'est à dire que le zéro en azimut correspond toujours à celui du petit disque gradué, mais le zéro en hauteur correspond à l'**orientation horizontale de la diode LASER**.
- Pour faciliter les orientations en mode manuel avec le petit joystick les deux vitesses lentes et rapides ont été considérablement diminuées par rapport à celles du programme précédent pour pouvoir aisément dégrossir des pointages précis. La procédure "Gigoter" invoquée par le caractère "**G**" n'est plus utile et a été enlevée.



Deuxième application : Cinéthéodolite optique.

Encore un vocable bien savant pour désigner en fin de compte une simple "mirette informatico-électronique" ! Concrètement, un Cinéthéodolite est une instrument généralement optique qui permet de suivre automatiquement une cible mobile. Par exemple lors des vols lunaires, les images du départ de la colossale fusée Saturn V montrées à la télévision étaient issues de tels dispositifs. L'analyse d'image en temps réel permettait de gérer les rotations des axes de ces matériels qui supportaient les caméras vidéo. Pour notre compte, nous allons simplifier considérablement la technologie et nous contenter d'un suivi automatique d'une source lumineuse prépondérante dans l'environnement de notre maquette expérimentale. Nous retrouvons l'expérience abordée en page 72, mais cette fois le suivi optique ne sera plus limité au plan horizontal, mais permettra de diriger le capteur dans les trois dimensions du volume environnant notre dispositif. On a déjà compris qu'il suffit de remplacer la diode LASER qui constituait la "charge utile" de nos articulations en Cardan motorisés par un capteur lumineux analogue à celui décrit dans les pages 72 et 73. Dans la mesure où l'on doit piloter le suivi optique sur deux axes, il faut impérativement doubler le système d'acquisition lumineuse. En effet, le principe de fonctionnement du capteur optique montré sur la Fig.2 de la page 72 est basé sur la différence de lumière reçue entre les deux cellules photorésistances si la source est décalée latéralement. Comme on peut le constater sur la Fig.1 ci-contre, si le déplacement de la source lumineuse se fait verticalement, quand elle passe des positions **P1** à **P2**, le flux varie bien, mais simultanément sur les deux capteurs **C1** et **C2**. Pour notre application, il faut donc ajouter un ensemble analogue, mais disposé pour mesurer des différences dans le plan vertical. Le schéma électronique donné en Fig.2 est totalement copié sur celui de l'expérience précédente mais comporte quatre cellules optroniques au lieu de deux. La photographie en gros plan donnée en Fig.3 montre en détail la charge utile placée sur notre plateforme expérimentale. On peut remarquer en **A** que la fiche de la liaison filaire déborde du connecteur. C'est uniquement pour utiliser des câbles très souples disponibles, mais qui ici comportent systématiquement quatre fils. Deux sont donc employés, car la broche "n°5" qui double la masse n'est pas utilisée, les deux fiches étant un peu larges pour pouvoir être placées l'une contre l'autre. Il importe surtout d'observer l'orientation précise adoptée pour les diverses cellules qui ainsi matérialisent la "vue binoculaire artificielle". Sur cette image, le plateau support **P** est en orientation initiale ZÉRO.

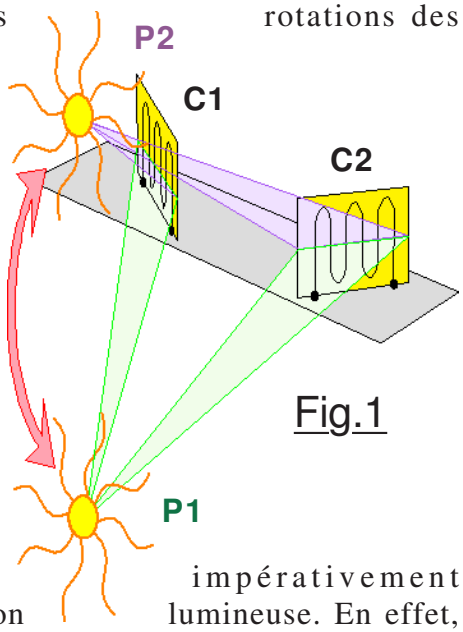


Fig.1

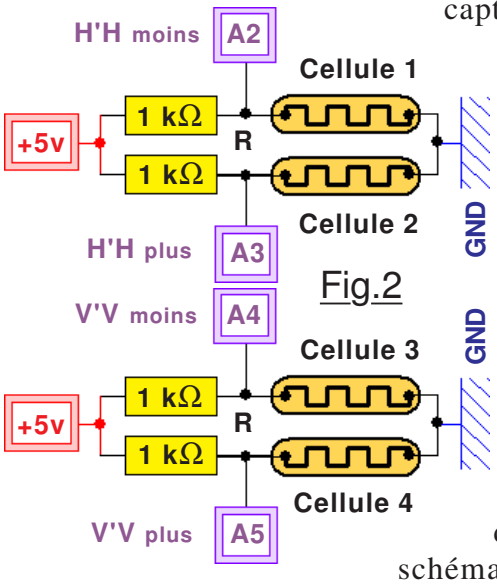


Fig.2

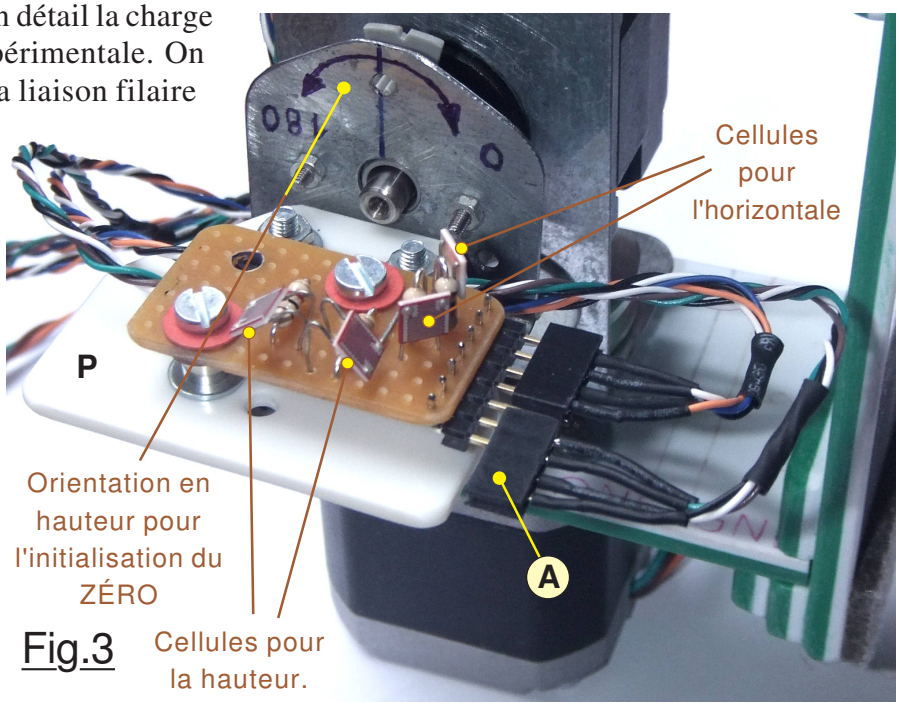
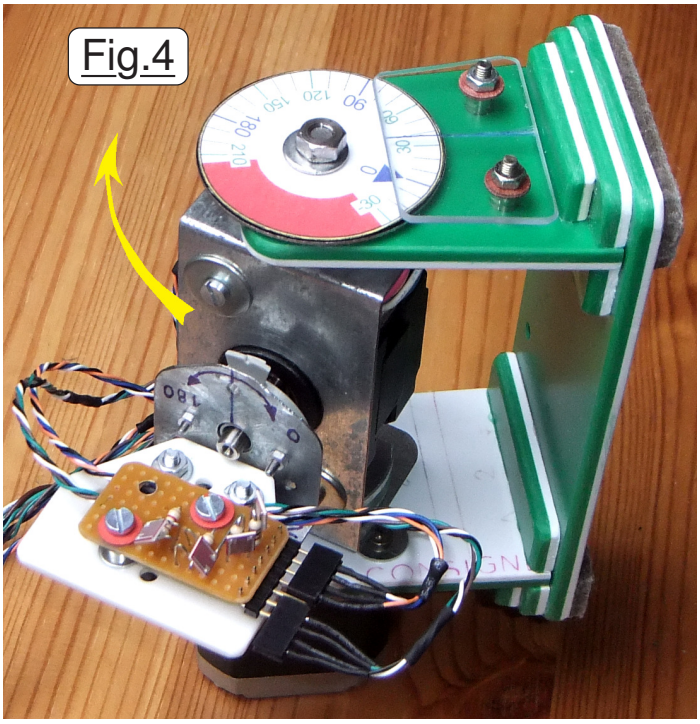


Fig.3

Pour mieux comprendre l'agencement du dispositif expérimental ainsi que la posture verticale adoptée par la structure de base des articulations croisées, la Fig.4 présente une vue d'ensemble du petit banc expérimental. Cette configuration correspond "au zéro initial" pour l'axe de hauteur, mais l'orientation horizontale a tourné d'un angle de 30° positif comme mis en évidence par la flèche jaune. Le capteur optique est bien visible avec ses quatre cellules photorésistantes. Ce sont des modèles identiques à ceux du capteur montré en page 73 car elles étaient disponibles. Pour ce montage, tout modèle de type LDR conviendra, inutile de focaliser sur une référence précise. En revanche, pour pouvoir facilement adopter les orientations visibles sur les images de la Fig.3 et de la Fig.4 il faudra choisir des modèles dont les fils de branchement sont situés dans le prolongement du plan contenant le substrat.



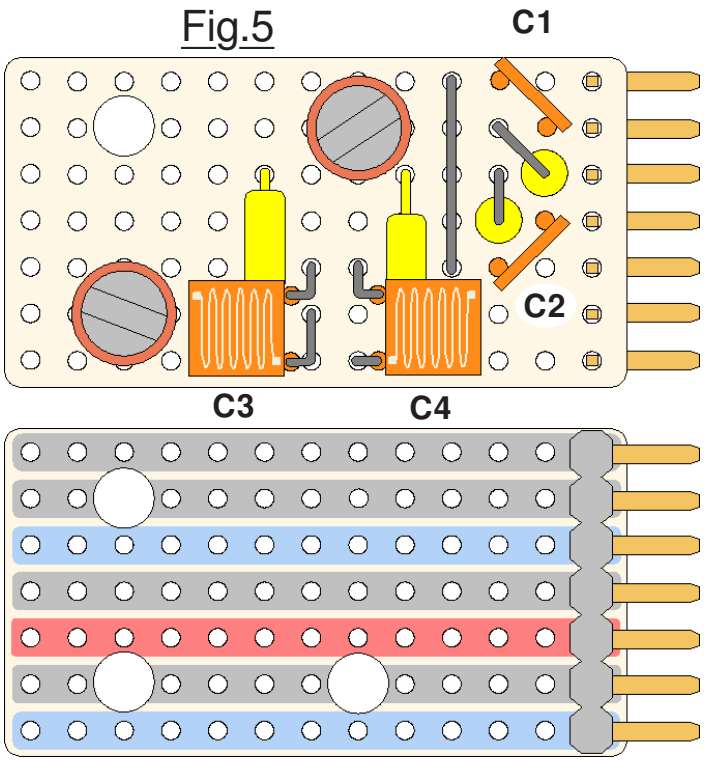
NOTE : Ceux qui désirent expérimenter cette application n'ont pas à réaliser le capteur montré en page 73, car celui-ci convient parfaitement en n'utilisant que les deux cellules **C1** et **C2**.

RÉPARTITION DES ENTRÉES / SORTIES D'ARDUINO :

Résumée dans le tableau ci-dessous, elle ressemble à s'y méprendre à celle de la deuxième expérience décrite en page 77 pour laquelle la LED d'alerte débordement et celle du mode REPOS des moteurs ont été placées respectivement sur les sorties binaires **D10** et **D9** pour libérer les deux entrées analogiques A2 et A3. Dans cette application les six entrées analogiques sont utilisées.

AFFECTATION DES ENTRÉES / SORTIES		
Broche	**Utilisation**	
D0	Réservée en standard pour la liaison série USB. (RX)	
D1	Réservée en standard pour la liaison série USB. (TX)	
D2	*Disponible F.E.*	
D3	Pilotage V3bis & 4bis du module électronique Shield ADAFRUIT.	En bleu les broches utilisées par le module électronique Shield ADAFRUIT.
D4	Pilotage SHIFT du module électronique Shield ADAFRUIT.	
D5	Pilotage V1 & 2 du module électronique Shield ADAFRUIT.	
D6	Pilotage V3 & 4 du module électronique Shield ADAFRUIT.	
D7	Pilotage OUTPUT du module électronique Shield ADAFRUIT.	
D8	Pilotage INPUT du module électronique Shield ADAFRUIT.	
D9	Sortie binaire pour piloter la LED signalant le mode VEILLE.	
D10	Sortie binaire pour piloter la LED signalant l'approche des butées.	
D11	Pilotage V1bis & 2bis du module électronique Shield ADAFRUIT.	
D12	Pilotage LATCH du module électronique Shield ADAFRUIT.	
D13	LED Arduino utilisée pour accuser réception d'une consigne USB.	
A0	Entrée analogique du JOYSTICK pour le TANGAGE.	
A1	Entrée analogique du JOYSTICK pour le ROULIS.	
A2	Entrée analogique de la cellule photorésistante H'H moins : **C1**.	
A3	Entrée analogique de la cellule photorésistante H'H plus : **C2**.	
A4	Entrée analogique de la cellule photorésistante V'V moins : **C3**.	
A5	Entrée analogique de la cellule photorésistante V'V plus : **C4**.	

Fig.5



Circuit imprimé vu coté pistes.

L'ASPECT MATÉRIEL :

L'électronique spécifique à cette application se résume encore à un tout petit morceau de circuit imprimé réunissant le connecteur et les quelques composants idoines. **Le connecteur est placé coté piste sous le circuit imprimé pour être abaissé et ainsi ne pas masquer l'éclairement des cellules C1 et C2**. Les résistances sont toutes de 1kΩ. Bien prendre en compte le fait que les cellules **C3** et **C4** ne sont pas soudées bien à plat sur le circuit imprimé, **mais inclinées à 45°** par rapport à ce dernier. Entre elles on trouve un angle droit. La Fig.3 montre bien cet impératif à respecter sans quoi la poursuite de la source lumineuse ne serait pas conforme à nos prévisions.

L'ASPECT LOGICIEL :

Le programme **CINE_THEODOLITE_OPTIQUE.ino** qui assure l'animation de cette petite application utilise des commandes similaires à celles du THÉODOLITE LASEZ. Les commandes clavier via la ligne série USB sont donc pratiquement les mêmes mis à part quelques menu différences. Quand on frappe la consigne "**C**" le programme liste sur l'écran du terminal série virtuel la liste des commandes. On remarquera les différences suivantes :

- La commande "/" n'est plus pertinente et a été supprimée. Comme la commande "**C**" qui était dédiée au clignotement de la diode LASER n'est plus utile, le rappel des commandes se fait à nouveau avec le caractère "**C**" au lieu de la lettre "**F**". *("C" pour COMMANDE au lieu de "F" pour FONCTION.)*
- La commande "**S**" permet dans ce programme de valider ou de suspendre le suivi optique automatique de la source lumineuse prépondérante. C'est la même commande pour les deux axes qui doit être suivie de "**V**" ou de "**H**" si l'on désire inverser l'état sur l'axe VERTICAL ou sur le mouvement HORIZONTAL. Cette commande doit se composer d'exactement deux caractères, le deuxième étant un "**V**" ou un "**H**". Si ce n'est pas le cas un message d'erreur sera affiché sur la ligne série.
- L'amplitude de la rotation horizontale est différente et la valeur de la butée logicielle HlimiteMIN passe de -78 PAS à -22 PAS car les fils de liaison dépassent davantage avec le nouveau circuit imprimé.
- Les amplitudes de rotation en hauteur sont augmentées pour pouvoir orienter verticalement vers le haut et vers le bas. Les valeurs des deux butées logicielles VlimiteMAX et VlimiteMIN passent à +200 PAS et -200 PAS. Les plages angulaires ainsi balayées sont largement suffisantes pour suivre le Soleil dans sa course diurne par exemple. *(En été il faut placer un masque pour diminuer la lumière sur les cellules car un pointage direct vers l'astre du jour aboutit à la saturation des cellules photorésistantes.)*
- Les butées logicielles ne protègeront correctement le matériel que si la configuration adoptée est celle déjà montrée sur la Fig.4 de la page 79 au moment de l'initialisation du ZÉRO de référence avec la commande "**C**". C'est à dire que le zéro en azimut correspond toujours à celui du petit disque gradué, et le zéro en hauteur correspond à l'**orientation horizontale du circuit électronique**.

NOTE IMPORTANTE : Au cours de nombreuses expériences, tant avec le disque gradué qu'avec les articulations en Cardan motorisées, par moment les moteurs PAS À PAS se mettaient à tourner à environ un pas par seconde, et ce sans aucune raison apparente. Il suffit de réunir la masse d'Arduino à la terre du secteur pour que ce phénomène disparaisse.