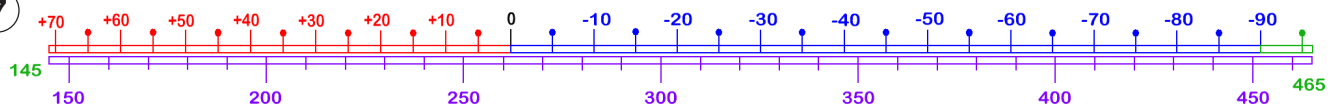
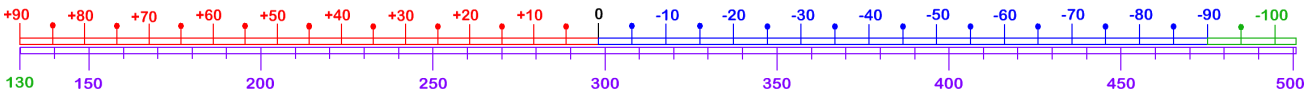
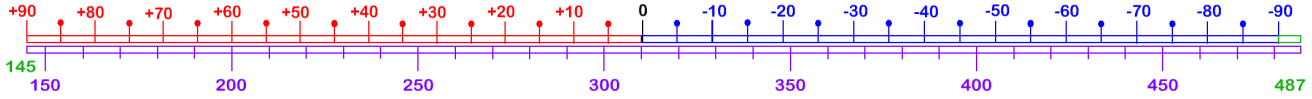
7 **Extérieur ou Avant** — **Hanche C** — **Intérieur ou Arrière**



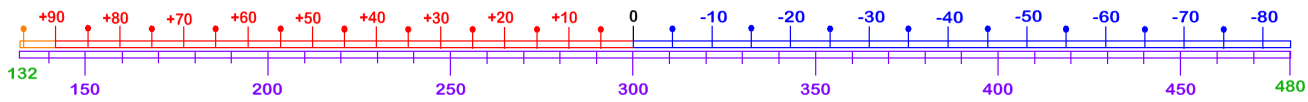
8 **Bas ou Intérieur** — **Genou C** — **Haut ou Extérieur**



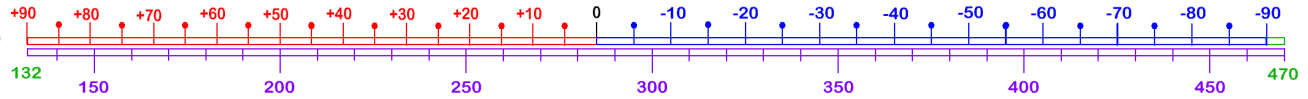
9 **Extérieur ou Haut** — **Pied C** — **Intérieur ou Bas**



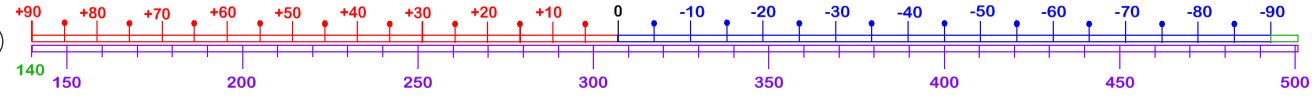
10 **Intérieur ou Avant** — **Hanche D** — **Extérieur ou Arrière**



11 **Haut ou Extérieur** — **Genou D** — **Bas ou Intérieur**

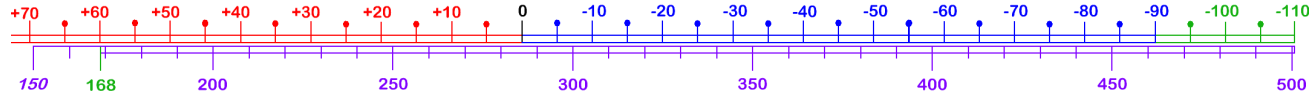


12 **Intérieur ou Bas** — **Pied D** — **Extérieur ou Haut**

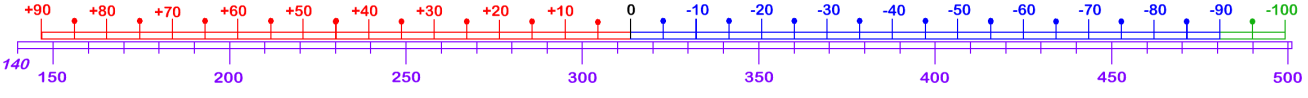


Fiche n° 10

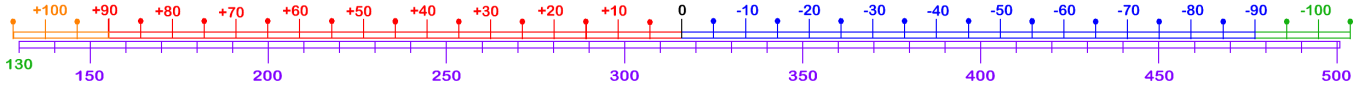
1 **Intérieur ou Avant** — **Hanche A** — **Extérieur ou Arrière**



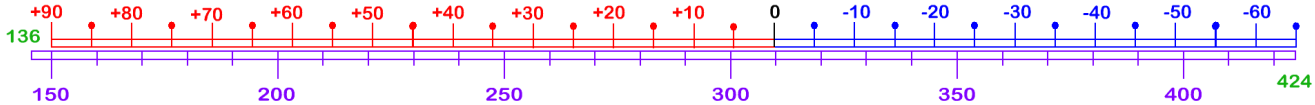
2 **Bas ou Intérieur** — **Genou A** — **Haut ou Extérieur**



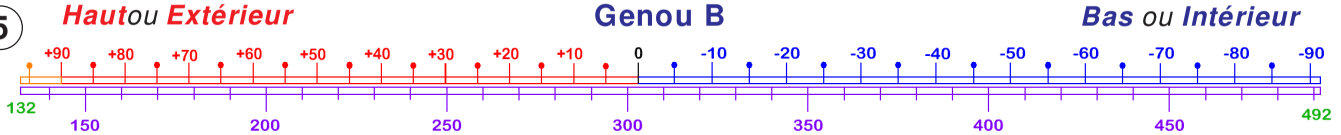
3 **Extérieur ou Bas** — **Pied** — **Intérieur ou Haut**



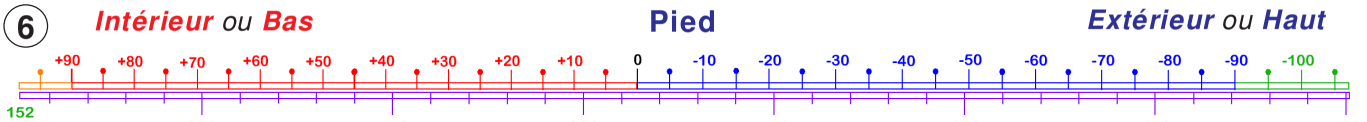
4 **Intérieur ou Arrière** — **Hanche B** — **Extérieur ou Avant**



5 **Hautou Extérieur** — **Genou B** — **Bas ou Intérieur**



6 **Intérieur ou Bas** — **Pied** — **Extérieur ou Haut**



Test du dialogue TX / RX entre deux Arduino. 1/ 2

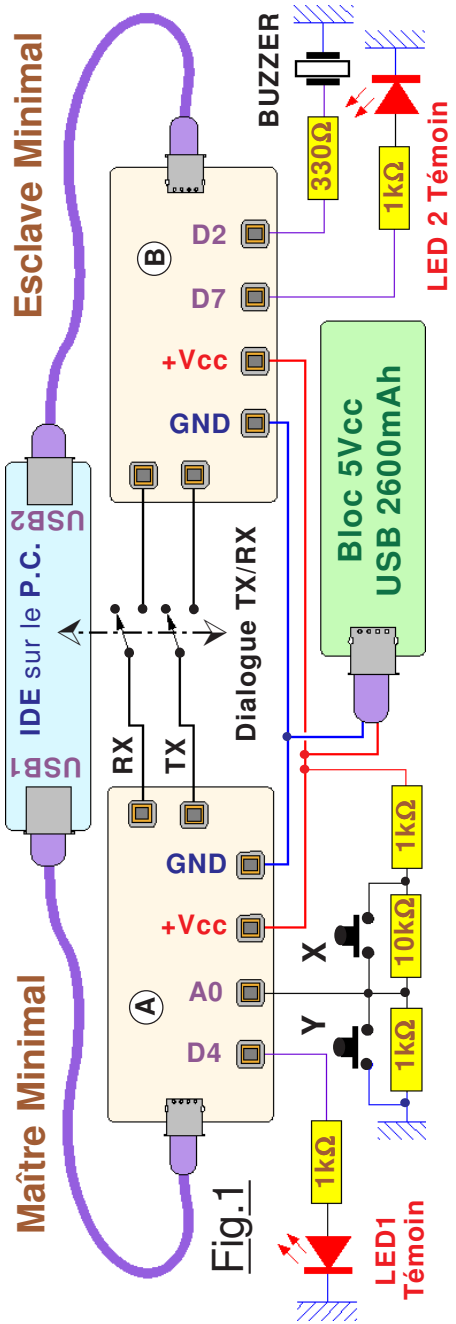


Fig.1

Pour tester la facilité d'échanger des données en mode alternat entre cartes Arduino, réaliser les branchements de la Fig.1 et téléverser le programme P18_PGM_Maitre_minimal.ino sur la carte **A**. Télécharger ensuite P19_PGM_Esclave_minimal.ino sur la carte **B**. Réaliser les essais en utilisant la procédure proposée sur le coté verso de cette fiche.

Pour une vérification de fiabilité dans les échanges de données entre les deux unités, téléverser le programme démonstrateur P20_Maitre_pour_dialogue_rapide.ino sur la carte **A**. Les échanges doivent se succéder sans interruption à une cadence rapide. Pour ce test débrancher le BUZZER de la carte **B** et le connecter sur **D6** de l'unité **A**. À tout moment on peut placer l'inverseur sur **IDE** sur le **P.C.** et téléverser de nouvelles versions de démonstrateurs, tant sur le Maître **A** que sur l'esclave **B**. Chaque carte Arduino est reliée à une prise USB séparée sur le P.C.

*Noter que sur le Maître **A** ou sur l'Esclave **B**, si l'on valide le Moniteur série de l'**IDE** on peut observer sur l'écran vidéo toutes les consignes envoyées par le programme.*

... / ...

Test du dialogue TX / RX entre deux Arduino. 2/ 2

P18_PGM_Maitre_minimal.ino et P19_PGM_Esclave_minimal.ino étant tous les deux téléversés sur leurs cartes Arduino respectives, quand l'inverseur est sur sur "Dialogue TX/RX" le comportement de l'ensemble est le suivant : *(Voir la Fig.1 au recto.)*

- Quand on clique sur **X** le Maître envoie "a*" sur **TX** et repasse immédiatement à l'écoute sur RX. Quand on clique sur **Y** le Maître envoie "1*" sur **TX** et repasse immédiatement à l'écoute sur **RX**. Si l'ACR est "!OK!" il allume LED 1 Témoin. Si l'ACR est "!ERR!" il éteint LED 1 Témoin.
- Chaque fois que l'Esclave reçoit une consigne sur **RX** il génère un BIP sonore. Si la consigne est "a*" il retourne "!OK!" sur **TX** et allume LED 2 Témoin. Si la consigne est "1*" il retourne "!ERR!" sur **TX** et éteint la LED 2 Témoin. Ce n'est pas une vraie erreur mais uniquement un ACR pour différencier les deux réponses.

Procéder à un essai logiciel :

1) L'inverseur est basculé sur "Dialogue TX/RX",
2) Sur l'une des deux liaisons USB 1 ou USB 2 téléverser le *(Ou sur les deux lignes.)* programme en cours de mise au point,
3) Faire un RESET sur les deux cartes Arduino,
4) Placer l'inverseur sur "Dialogue TX/RX",
5) Procéder aux essais,
6) **Replacer l'inverseur sur "Lignes TX/RX ouvertes".**

ATTENTION : Toute tentative de téléverser un programme sur l'une des cartes Arduino alors que l'inverseur est sur "Dialogue TX/RX" fait perdre la synchronisation avec l'**IDE**. Dans ce cas :

Reprise d'une synchronisation avec l'IDE :

1) Basculer l'inverseur sur **IDE** sur le **P.C**,
2) Débrancher la ligne USB incriminée,
3) Fermer l'éditeur sur le programme "en cause",
4) Invoquer à nouveau le programme.ino à tester,
5) Rebrancher la ligne USB vers le P.C,
6) Dans les outils valider le Port affecté par le P.C. à l'**IDE**,
7) Téléverser le programme.ino. Normalement le téléchargement doit s'effectuer de façon normale.
8) Reprendre les manipulations et Procéder à un essai logiciel.

... / ...

Clavier multiplexé 1/ 3

Comme montré sur le dessin de la Fig.1 on agence un clavier de type "multiplexé ligne / colonne". Une exploitation banale de type ligne / colonne imposerait de mobiliser 4 sorties et 4 entrées sur la carte Arduino NANO. Moyennant de faire appel à une entrée analogique et à une série de résistances placées en parallèle et constituant un diviseur de tension il devient possible de n'utiliser qu'une seule entrée. L'idée de base pour effectuer la différenciation des lignes se fait alors par l'application de valeurs de tensions différentes sur ces dernières. La Fig.2 donne le principe utilisé. Les résistances de 1 kΩ alimentent les colonnes avec le **+ 5Vcc** lorsque l'une des sorties binaires passe au niveau haut. Les tensions notées sur la Fig.2 sont celles obtenues quand on appui sur l'un des boutons poussoir situés sur la ligne testée. En réalité les valeurs de résistances utilisées privilégient des éléments très courants, les tensions mesurées sont fonction de la chute de potentiel d'environ 0,5 Vcc au passage du courant dans les diodes **D**. Dans la pratique, les tensions présentes seront légèrement plus faibles car les sorties binaires ne fournissent pas exactement la valeur de + 5 Vcc quand elles sont forcées au niveau "1" et surtout **R1** se trouve en parallèle sur **R3**. Pour l'entrée analogique on utilisera **A6** car elle ne peut fonctionner qu'en entrée analogique. Pour les sorties binaires, celles qui sont proposées sur le schéma de la Fig.2 sont voisines sur le connecteur HE14. On peut se le permettre car dans l'application il n'est pas fait beaucoup usage de la PWM. Si aucun des boutons poussoir de la colonne n'est appuyé, la tension présente et le courant consommé seront nuls. Cette technique ne permet pas de gérer l'appui simultanné de plusieurs touches.

Tensions et numérisation.

SW	Tension	CAN
Aucun	0 Vcc	0
S1	3.95 Vcc	806
S2	3,1 Vcc	631
S3	1,95 Vcc	392
S4	0,95 Vcc	180

... / ...

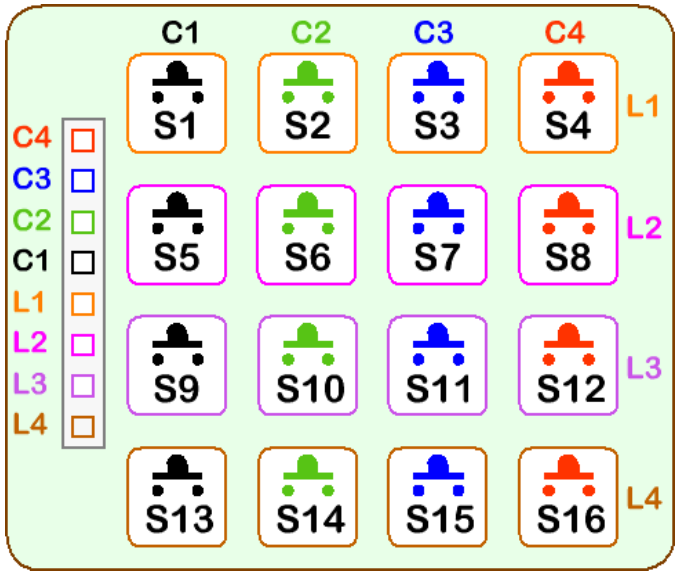


Fig.1

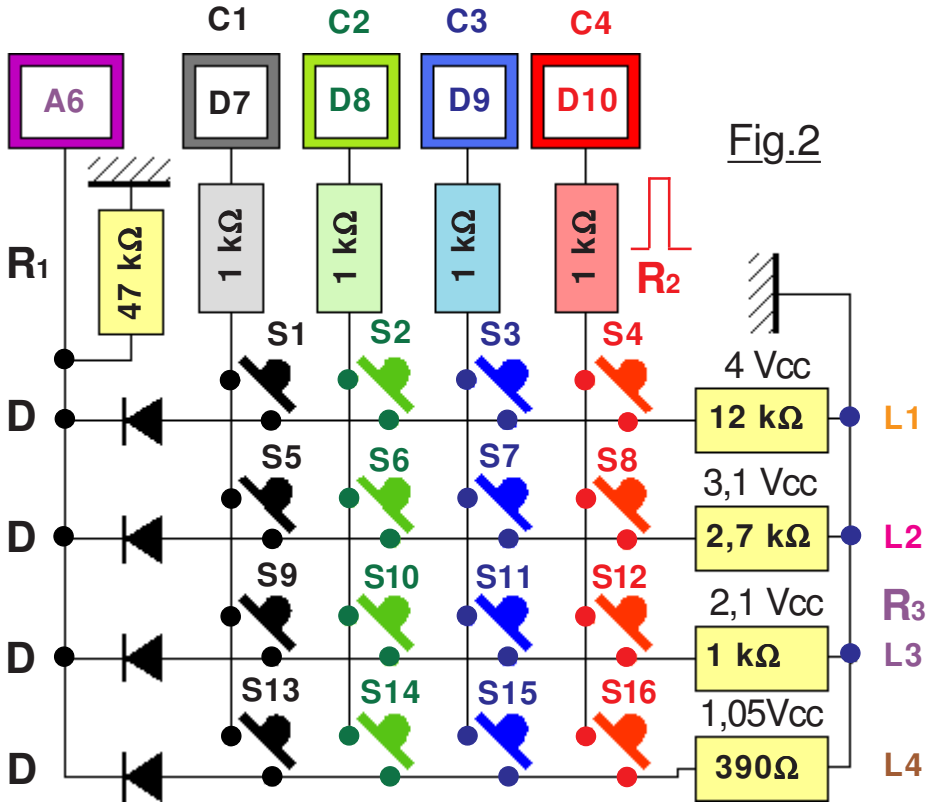


Fig.2

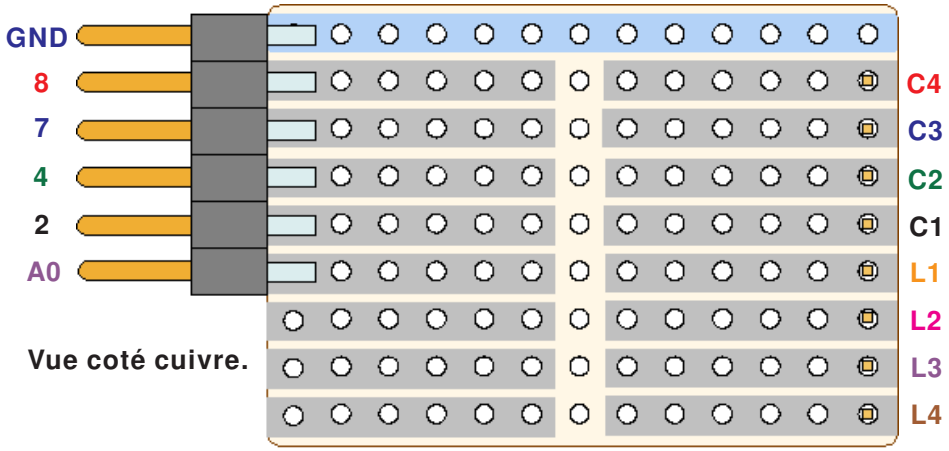
Clavier multiplexé 2/ 3

... / ...

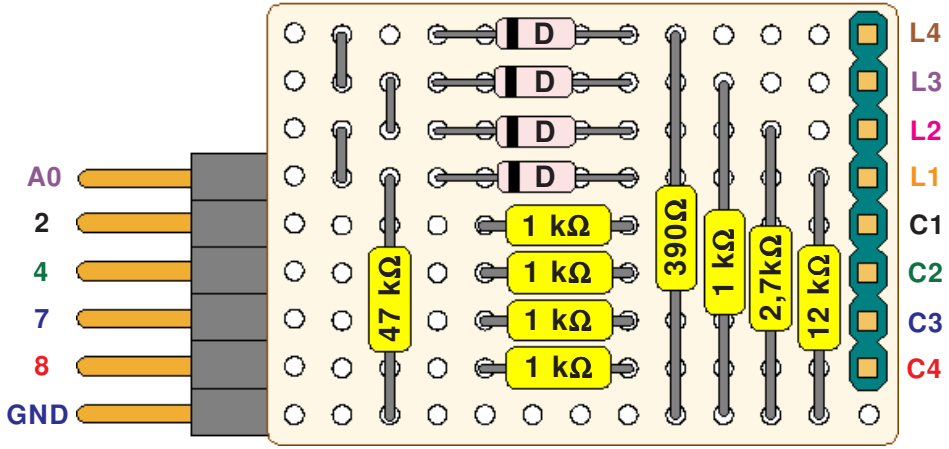
Clavier multiplexé 3/ 3

La conception de ce petit circuit imprimé est issue du schéma électrique donné en Fig.3 de la Fiche n°18 et fait partie des petits montages expérimentaux pour évaluer Arduino. Compte tenu du nombre de composants, il devenait rentable de réaliser un petit circuit imprimé sur lequel se branche directement le module mini clavier. Pour ne pas encombrer de façon peu commode en hauteur, une fois inséré sur le connecteur du mini clavier le circuit imprimé se trouve vers l'extérieur et à l'horizontale. Pour minimiser l'encombrement en hauteur tous les composants sont placés à plat.

Interface du Mini clavier matricé au format 4L x 4C.



Vue coté cuivre.



Vue coté composants.

Fiche n° 33

Codage des consignes. (3/3)

CODE	ACTION
69*	Enregistrer un balayage télémétrique.
70*/	Afficher le balayage télémétrique enregistré.
71*/	Débuter un apprentissage.
72*/	Terminer un apprentissage.
73*	Mode Manuel début.
74*	Mode Manuel fin.
75*	Jambe A.
76*	Jambe B.
77*	Jambe C.
78*	Jambe D.
79*	Hanche.
80*	Genou.
81*	Griffe.
82*/	Retourner en ACR la configuration courante.
83*/	Retourner la configuration sauvée en EEPROM.
84*/	Fait retourner le n° du PGM actif sur la Sonde.
85*/	Impose le n° du PGM actif sur la Sonde.
86*/	Lister un programme logé en EEPROM.
87*/	Déclencher un programme logé en EEPROM.
88*/	Effacer le PGM indexé en EEPROM.
89*/	Effacer le dernier code d'un apprentissage.
90*	Générer une temporisation d'une seconde.
91*/	Activer l'affichage continu des données de NAV.
92*/	Stopper l'affichage continu des données de NAV.
93*/	Retourner la valeur du Roulis.
94*/	Tester le PGM indexé pour NON vide. *(E18 si vide.)*
95*/	Impose le nombre de PGM à chaîner.
96*/	Retourner la référence du CAP magnétique.
97*/	Impose comme Référence le CAP magnétique.
98*/	Impose comme Référence le Gyroscope de LACET.
99*	N° POSTURE EEPROM prend le n° du PGM actuel.

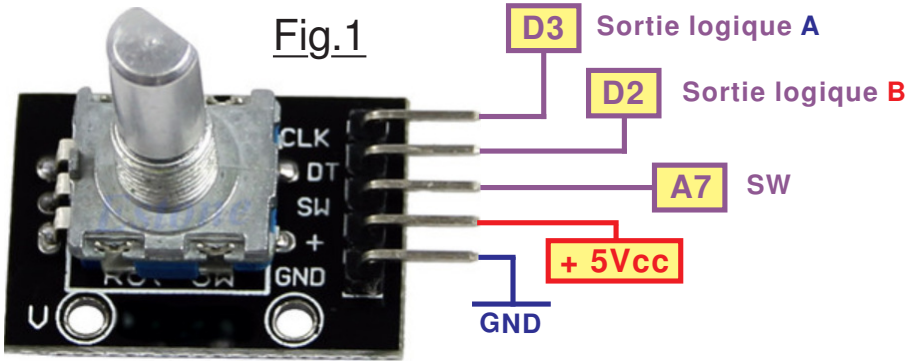
Penser à vérifier pour chaque nouveau code s'il faut l'inclure aux interdit d'un apprentissage.

(/ signifie que le code ne sera pas intégré durant l'apprentissage.)*

Encodeur rotatif KY-040. 1/2

Composant périphérique particulièrement bien adapté à l'environnement Arduino, le KY-40 est un codeur **sans butée** dont la définition est de 20 points par tour. Il est pourvu d'un "bouton poussoir" par appui sur la tige de commande qui permet d'envoyer une commande spécifique "hors" clavier à touches. La sortie se fait sur des contacts de type courtes impulsions à "0" avec déphasage de 90°. La Fig.1 donne le schéma des branchements à réaliser sur Arduino quand on dispose d'un modèle de qualité. Les broches sont ici affectées pour la compatibilité avec la Raquette de commande.

Fig.1

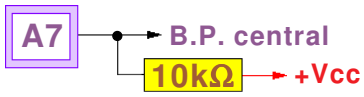


Caractéristiques techniques du module KY-040 :

- Consommation maximale : 10 mA sous 5 Vcc.
- Température de fonctionnement : - 30 à + 70 °C.
- Température de stockage : - 40 à + 85 °C.
- Durée de vie du capteur de rotation : Minimum 30 000 cycles.
- Durée de vie du contact B.P. central : Minimum 20 000 cycles.
- Résistance de passage du contact du B.P. central : 3 Ω maximum.

ATTENTION : On peut se procurer de tels capteurs compatibles que qualité un peu inférieure. Par exemple pour une "broutille" ils sont commercialisés par paquets de cinq. C'est ce genre de lot qui a été commandé pour JEKERT. Cette économie engendre deux petits inconvénients qu'il faut compenser. Bien que la sérigraphie du petit circuit imprimé y fait mention, la résistance de 10kΩ qui force un état logique "1" sur le B.P. central **SW** n'est pas installée. Comme montré sur la Fig.2 il

Fig.2

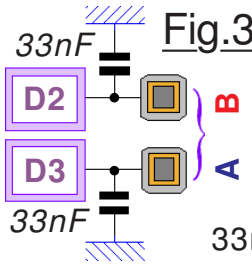


... / ...

Encodeur rotatif KY-040. 2/2

faudra l'ajouter sur l'électronique interne à la Raquette de commande. Par ailleurs, les contacts électriques de codage ne sont pas excellents et sont affectés d'un nombre important de rebonds parasites à la commutation. On va en partie éliminer cet inconvénient par logiciel. Toutefois, le codage sera simplifié si l'on améliore les conditions électriques en plaçant comme montré sur la Fig.3 un condensateur de 33nF entre chaque entrée de commutation et GND.

Fig.3



Fonctionnement :

La sortie **SW** est celle du bouton poussoir central piloté par appui sur la tige du rotor. Les deux broches de codage **CLK** et **DT** sont en réalité deux sorties impulsionnelles classiques avec déphasage de 90° souvent nommées **A** et **B** pour ce type de capteur. Le déphasage de 90° des signaux logique **CLK** et **DT** permet de déterminer le sens de rotation. *(Voir Fig.4 montrant un pas dans chaque sens)*

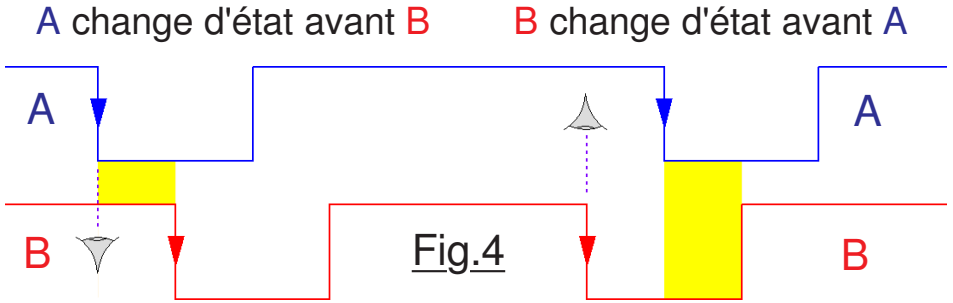


Fig.4

Comme montré Fig.4 la détermination du sens de rotation quand des impulsions se produisent se fait par observation de l'ordre chronologique des transitions du front descendant par exemple. Chaque encliquetage génère une impulsion sur les deux sorties. On détermine ainsi un pas et son sens de rotation.

Autre possibilité de détermination du sens de rotation :

- Dans un sens de rotation pendant le front descendant du signal **A**, le signal **B** est à l'état logique "1".
- Dans l'autre sens de rotation pendant le front descendant du signal **A**, le signal **B** est à l'état logique "0".

... / ...

Codage des consignes. (1/3)

CODE	ACTION
01*	Posture posée sur le sol.
02*	Posture Stable Transversal.
03*	Avancer de N pas.
04*	Reculer de N pas.
05*	Tourner à Droite de N pas.
06*	Tourner à Gauche de N pas.
07*	Décaler à Droite de N pas.
08*	Décaler à Gauche de N pas.
09*	Simuler un TIR LASER.
10*	Configuration Hauteur Maximale.
11*	Retour de Configuration Hauteur Maximale.
12*/	Impose le n° du PGM actuel comme SOURCE.
13*/	Déplace la SOURCE dans le "PGM actuel".
14*	Libérer les efforts. *(Touche permanente.)*
15*	Passer tous les moteurs au Neutre Opérationnel.
16*	Configuration Stable Raisonnable.
17*	Retour de Configuration Stable Raisonnable.
18*/	Enregistrer en EEPROM la posture actuelle.
19*	Utiliser l'enregistrement EEPROM de la posture.
20*/	Ajouter des codes à un un apprentissage.
21*	Déposer la sonde.
22*	Configuration Atterrissage.
23*/	Configuration Décollage.
24*/	Retourne la version du PGM de la Sonde.
25*	Couper les énergies. *(Couper ☼ OUI.)*
26*	Rétablir les énergies. *(Couper ☼ NON.)*
27*	Allumer les phares.
28*	Éteindre les phares.
29*	Allumer le LASER.
30*	Éteindre le LASER.
31*	Figer les Moteurs.
32*	Rétablir les Moteurs.
33*	Coordinations RAPIDES.
34*	Coordinations LENTES.

(/ signifie que le code ne sera pas intégré durant l'apprentissage.)*

Codage des consignes. (2/3)

CODE	ACTION
35*	Stabilisation Gyroscopique Active.
36*	Stabilisation Gyroscopique suspendue.
37*/	Retourner la configuration de posture actuelle.
38*	Débuter un mouvement de Torsion.
39*	Terminer un mouvement de Torsion.
40*	Incrémenter la valeur du codeur rotatif.
41*	Décrémenter la valeur du codeur rotatif.
42*	Le codeur rotatif agit sur la puissance du LASER.
43*	Le codeur rotatif agit sur l'éclairage des PHARES.
44*/	Retourner l'état des booléens système.
45*/	Retourner la tension sur la Motorisation.
46*/	Retourner la puissance des PHARES.
47*/	Retourner la puissance du LASER.
48*/	Retourner la valeur de la Température.
49*/	Retourner la valeur de l'Hygrométrie relative.
50*/	Retourne l'état "Hibernation".
51*/	Retourner la distance télémétrique.
52*/	Retourner le CAP MAGNÉTIQUE.
53*	Fin du Gradateur.
54*	LASER et Phares forcés à 127.
55*	Réveiller JEKERT. (Sortie de l'Hibernation.)
56*/	Retourner la valeur du Tangage.
57*/	Retourner la valeur du Calage Gyroscopique.
58*/	Tester le fonctionnement de la centrale inertielle.
59*/	Calage gyroscope sur sa référence interne actuelle.
60*/	Retourner l'écart de route gyroscopique.
61*	Passer en mode Hibernation.. *(Ancien "Quitter".)*
62*	Début d'utilisation du LASER. *(Pointage.)*
63*	FIN d'utilisation du LASER. *(Pointage.)*
64*	Pointage Vertical Début.
65*	Pointage Vertical FIN.
66*	Inverse débattements angulaires Petits / Grands.
67*	Enregistrer un spectre colorimétrique.
68*/	Retourner les valeurs du spectre colorimétrique.

(/ signifie que le code ne sera pas intégré durant l'apprentissage.)*

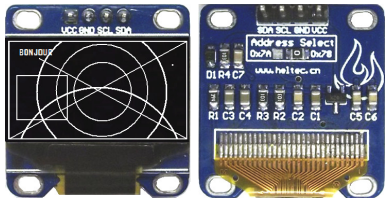
Méthodes de la bibliothèque Adafruit_ssd1306syp.h.

Spécifique à l'afficheur graphique monochrome miniature OLED 128 x 64, cette bibliothèque est suffisante pour mettre facilement en œuvre ce produit aussi bien pour du tracé géométrique que pour de l'affichage de textes.

Initialisations diverses.

Adafruit_ssd1306syp display(broche_SDA, broche_SCL);
Cette directive doit être placée avant void **setup**() et précise les deux sorties binaires d'Arduino qui seront utilisées pour interfacer l'I2C. Il n'est pas utile de configurer ces deux broches en sorties.

display.initialize(); delay(100);
Instruction à placer dans void **setup**() avant toute utilisation de l'afficheur. Cette instruction se charge de définir les deux broches affectées à la ligne I2C en sorties. Les broches A0 à A5 sont parfaitement utilisables en sorties. *(Déclarations 14 à 19.)*

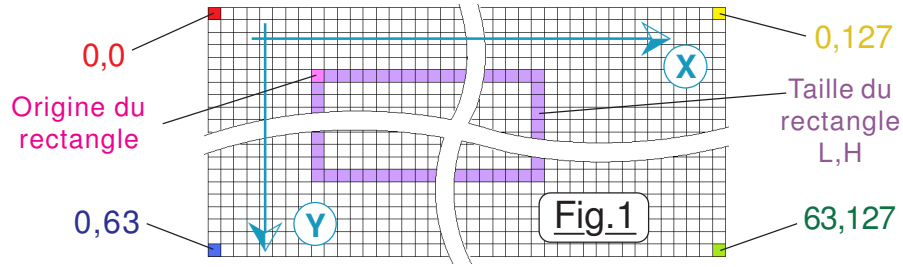


Principe des d'affichages.

Qu'il soit graphique ou texte, un affichage se fait toujours en deux phases. La première action consiste à placer les "pixels" dans la mémoire tampon 128 x 64 de l'afficheur. La deuxième action consiste à transférer l'affichage sur l'écran LCD avec l'instruction display.update(); cette ligne étant indispensable même pour l'instruction display.clear(); d'effacement de l'écran.
La Fig.1 précise les valeurs des coordonnées pour les pixels.

Affichages graphiques.

display.drawPixel(X, Y, WHITE);
Procédure qui permet de gérer l'afficheur point par point. Elle allume le pixel de coordonnées X et Y si le paramètre de lumière est WHITE ou éteint le point si le paramètre est BLACK.



display.drawLine(Xd, Yd, Xf, Yf,WHITE);
Procédure qui trace une ligne entre les deux points dont on précise les coordonnées. WHITE ou BLACK précise si on trace en blanc ou en noir. L'ordre de définition des extrémités est quelconque.

display.drawRect(Xo, Yo, Largeur, Hauteur,WHITE); *(Voir Fig.1)*
Procédure qui trace un rectangle dont on précise les **coordonnées de l'Origine**, la Largeur et la Hauteur. WHITE ou BLACK précise si on trace en blanc ou en noir. Largeur et hauteur peuvent avoir des valeurs négatives et sont alors tracés vers la gauche et vers le haut.

display.drawCircle(Xo, Yo, Rayon, WHITE);
Procédure qui trace un cercle dont on précise les **coordonnées du centre** et la valeur du Rayon. WHITE ou BLACK précise si on trace en blanc ou en noir. Gère les débordements de 127 x 63.

Affichages des textes.

display.setTextColor(WHITE);
Procédure qui définit la "couleur" du texte. (Noir si BLACK.)

display.setTextColor(TEXTE,FOND);
Procédure qui définit la couleur du texte puis du fond. Par exemple display.setTextColor(BLACK, WHITE); fait passer le texte en "surbrillance" par inversion écriture en noir sur fond blanc.

display.setCursor(X, Y);
Positionne le curseur de la prochaine écriture textuelle.

display.setTextSize(Taille);
Définit la taille des caractères, la plus petite valeur étant **1.**
1 : Matrice 5x7, **2** : Le double, **3** : Le triple etc.

display.print(Donnée); display.println(Donnée);
Affiche la donnée à partir de la position actuelle du curseur. Gère les débordements de la fenêtre 127 x 63, le texte dépassant du domaine affichable sera ignoré. Le format de la Donnée peut être :
display.println("Texte quelconque."); *(1)*
display.print(PI,8); Valeur et en option le nombre de décimales.
display.println(1234, BIN); Valeur affichée en Binaire.
display.print(1234, HEX); Valeur affichée en Hexadécimal.
display.println(1234, OCT); Valeur affichée en Octal.

(1) : Les lettres, les chiffres et la ponctuation standard sont correctement affichés ainsi que 128 autres caractères particuliers. (Matrices standard ASCII au format 5x7 pour le coefficient 1.)