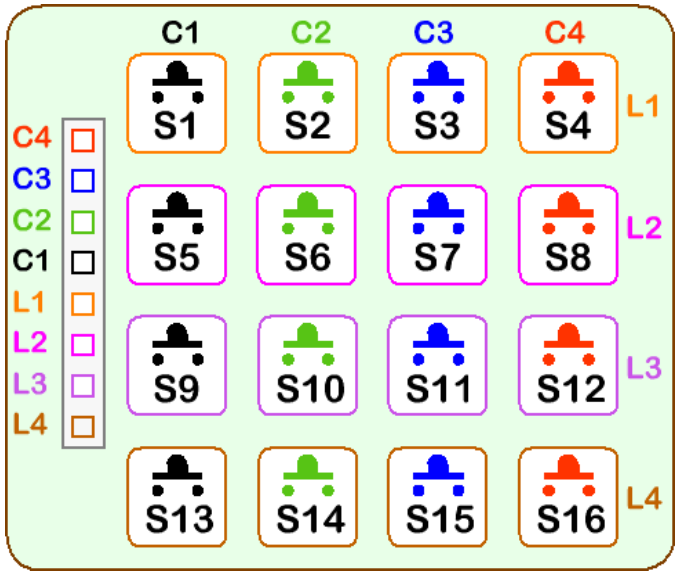
Clavier multiplexé 1/ 2

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

... / ...



Clavier multiplexé 2/ 2

Fig.1

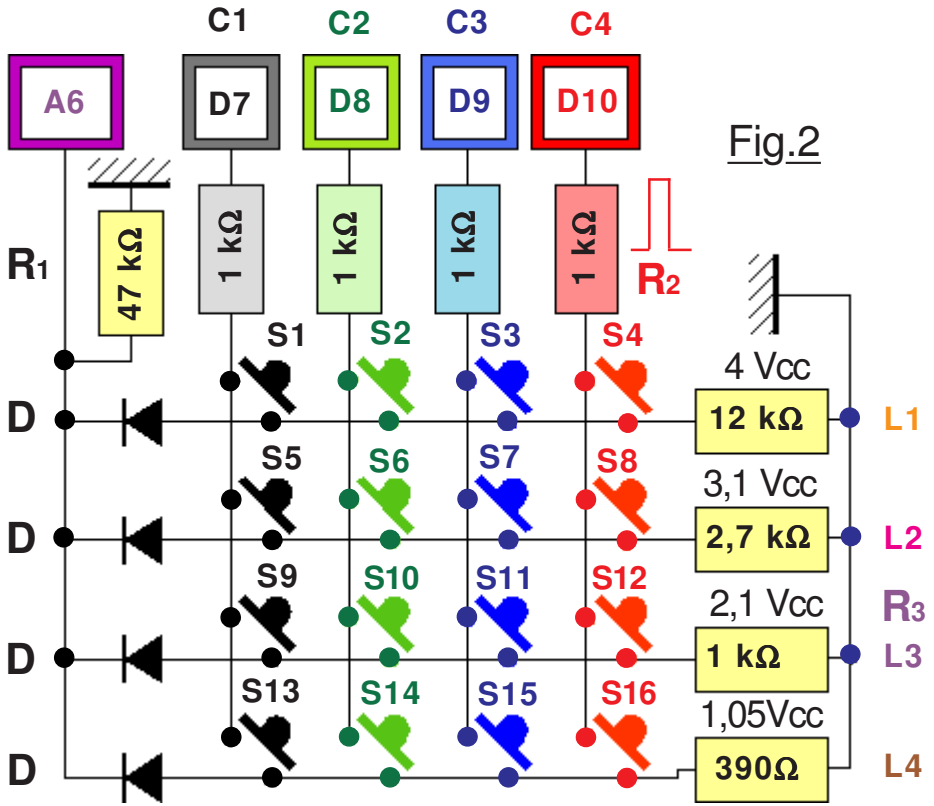
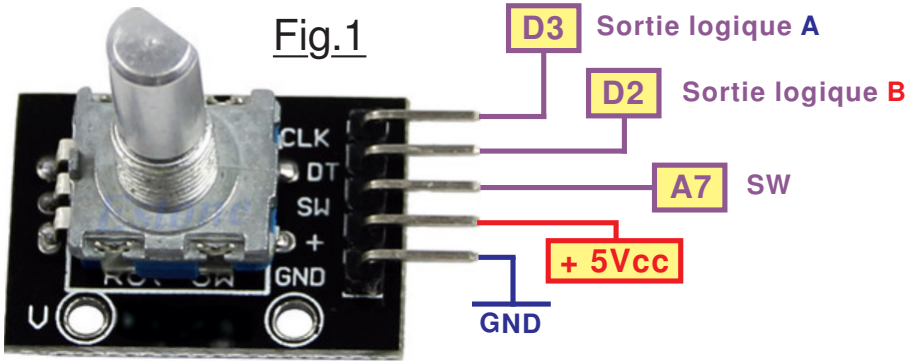


Fig.2

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

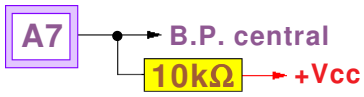


Caractéristiques techniques du module KY-040 :

- 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

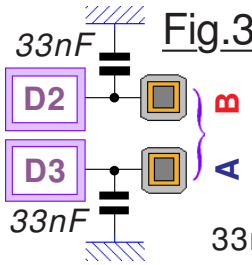


... / ...

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)*

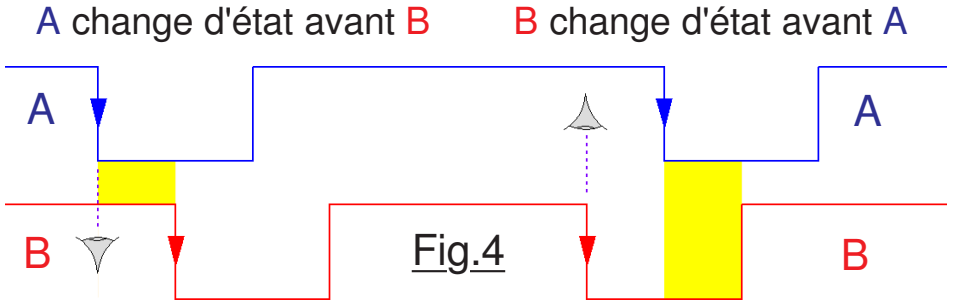


Fig.4

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".

... / ...

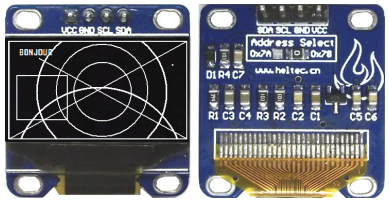
Méthodes de la bibliothèque Adafruit_ssd1306syp.h.

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.)*

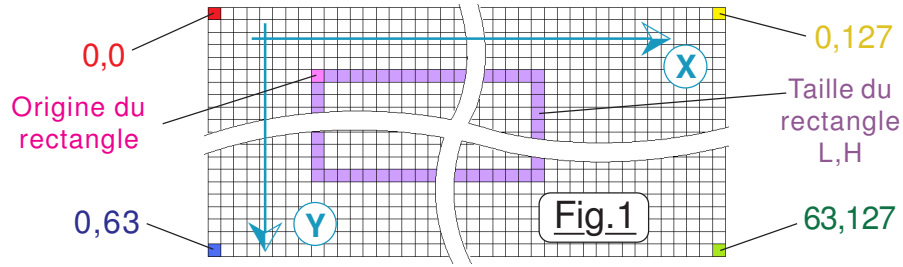


Principe des d'affichages.

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.)

... / ...

SCHÉMAS ÉLECTRONIQUES.

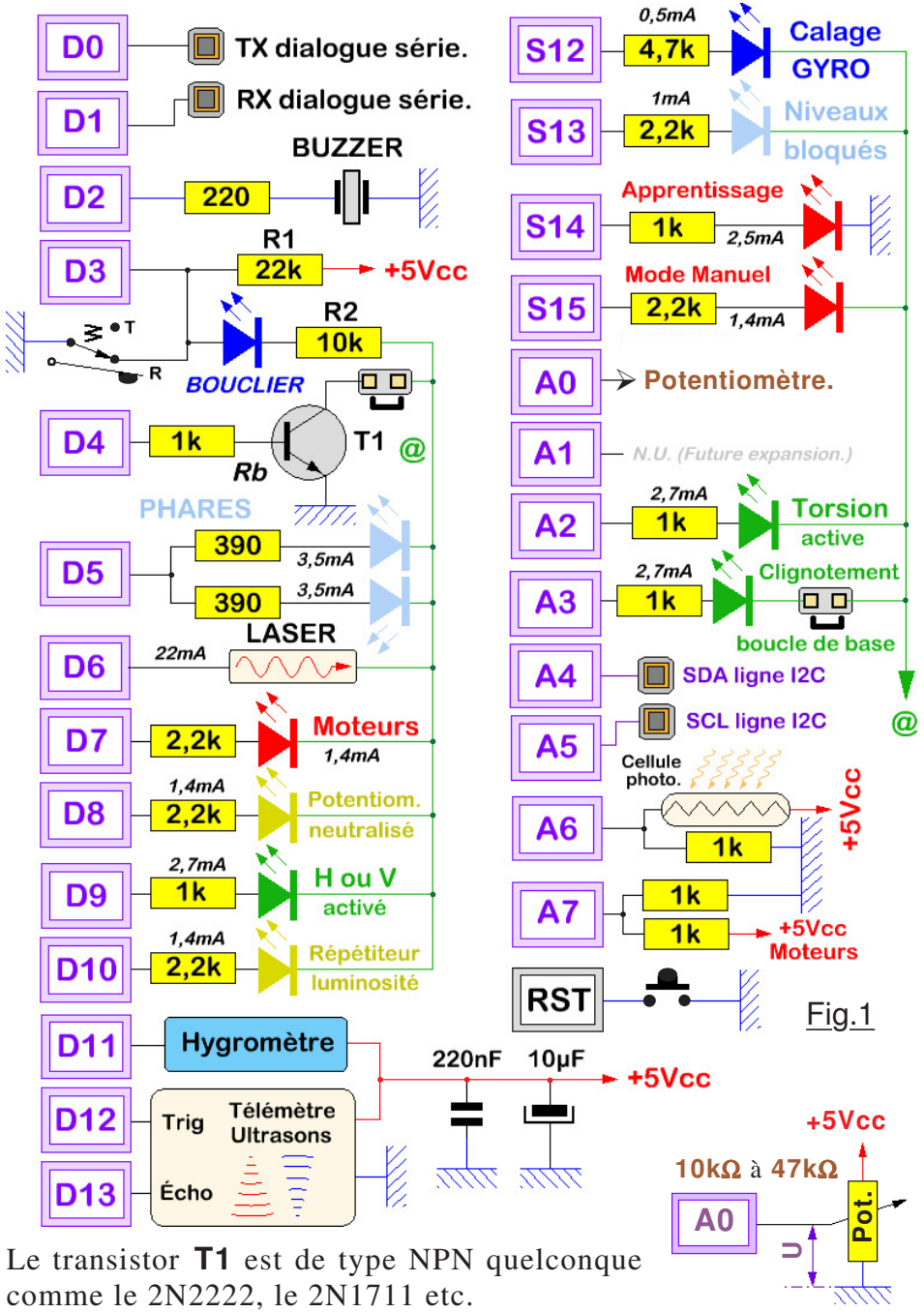


Fig.1

Le transistor **T1** est de type NPN quelconque comme le 2N2222, le 2N1711 etc.

... / ...

Fiche n° 4

Affectation des Entrées / Sorties.

D0 : *Dialogue série avec le moniteur de l'IDE.*
D1 : *Dialogue série avec le moniteur de l'IDE.*
D2 : Pilotage du BUZZER.
≈ D3 : Détection du contact du bouclier avec le sol.
D4 : Coupure de masse de toutes les LED. *(Pilotage de la base du transistor de commutation.)*
≈ D5 : Éclairage analogique *(PWM)* des phares.
≈ D6 : Pilotage du LASER.
D7 : Pilotage LED témoin Moteurs figés.
D8 : Pilotage LED témoin Potentiomètre neutralisé.
≈ D9 : Pilotage LED témoin mode H ou V.
≈ D10 : Pilote la LED répétiteur du CAN potentiomètre.
≈ D11 : Capteur Humidistance et de température.
D12 : Transmetteur ultrasons.
D13 : Récepteur ultrasons.

A0 : Sur la version P.C : Commande analogique.
A1 : Libre pour F.E. car non utilisée.
A2 : Pilotage LED de torsion active.
A3 : Clignotement de la LED de "boucle de PGM active".
A4 : SDA pour la ligne I2C.
A5 : SCL pour la ligne I2C.
A6 : Mesure de la lumière pour le luxmètre.
A7 : Mesure de la tension d'alimentation des moteurs.

Sortie S12 Multiplexeur : LED de la commande '=*' armée.
(Mémorisation du Gyroscope sur le Lacet actuel.)
Sortie S13 Multiplexeur : LED de l'option 'B*' active.
(Niveau éclairages des phares et du LASER Bloqués.)
Sortie S14 Multiplexeur : LED du mode 'D*' actif. .....
(Enregistrement d'une suite de déplacements en cours.)
Sortie S15 Multiplexeur : LED du mode "P9n*" en cours.
(Pilotage des moteurs manuellement et individuellement.)

... / ...