

Recommandations pour réaliser les fiches.

Par Nulentout (Achevé le 7 Mars 2025.)

Mise à part cette première page d'explications, toutes les autres **sont prévues pour réaliser des fiches en imprimant les "pages" de ce document par "paire" Recto / Verso** sur du papier de qualité suffisante pour que les éléments situés d'un coté ne soient pas visibles de l'autre. Pour ne pas vous tromper, le tableau de la Fig.1 précise quelques paires constituant une même "fiche double" car chaque page au format A4 est relatif à deux fiches indépendantes.

Réaliser une "paire de fiche" n'est pas spécialement compliqué, toutefois le fait d'avoir à imprimer des deux cotés d'une feuille impose une procédure simple, mais rigoureuse. À titre d'exemple on va traiter le cas des fiches couplées de la FEUILLE ⑤ :

- 1) Imprimer la **page 10**. (Repère vertical en gris clair au centre.)
- 2) Remplacer la page sur votre périphérique et imprimer la **page 11**.
Logiquement, si c'est une imprimante classique, il suffit en principe de remplacer la feuille sur le dessus du bac à papier dans la position et l'orientation qu'elle occupe en sortie de la machine en la retournant coté non imprimé vers vous.
- 3) Étape non obligatoire, personnellement je protège toutes mes fiches, disposant d'une petite plastifieuse thermique pour P.C.
- 4) Il ne reste plus qu'à séparer les deux fiches en coupant la feuille par le milieu. Puis on découpe tout le tour de la fiche le cadre gris clair "en laissant vivre le trait", cadre qui en délimite la périphérie.

Il n'y a pas forcément un ordre logique dans la succession des fiches car elles ont été rédigées au cours des semaines durant le développement de ce petit projet. **Naturellement, il n'est absolument pas obligatoire d'imprimer toutes les fiches. Vous avez parfaitement le loisir de ne concrétiser que celles qui vous semblent indispensables ...**

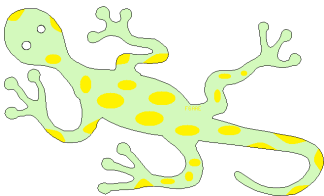
NOTE : Parfois la façon dont sont organisées les diverses fiches d'un même thème peut sembler curieuse. Elles sont paginées pour permettre au lecteur d'avoir simultanément deux faces relatives à des informations communes.

Personnellement, quand j'imprime de telles fiches, je complète leur réalisation en les plastifiant. Outre la belle apparence que cette pratique leur confère, elles sont bien plus rigides et agréables à manipuler. C'est d'autant plus facile qu'actuellement une petite plastifieuse au format A4 est d'un coût peu élevé et devenu vraiment banal dans les boutiques "informatiques".

Exemples de "Paires de pages" pour réaliser les fiches Recto / verso.




RECTO	VERSO	RECTO	VERSO
2	3	4	5
RECTO	VERSO	RECTO	VERSO
6	7	8	9
RECTO	VERSO	RECTO	VERSO
10	11	12	13
RECTO	VERSO	RECTO	VERSO
14	15	16	17
RECTO	VERSO	RECTO	VERSO
18	19	20	21

Fig.1



Fiche n°1.

TABRE DES CARACTÈRES ASCII affichables sur LCD

DEC	B	Car	DEC	B	Car	DEC	B	Car
32	00100000	SPC	64	01000000	@	96	01100000	
33	00100001	!	65	01000001	A	97	01100001	a
34	00100010	"	66	01000010	B	98	01100010	b
35	00100011	#	67	01000011	C	99	01100011	c
36	00100100	\$	68	01000100	D	100	01100100	d
37	00100101	%	69	01000101	E	101	01100101	e
38	00100110	&	70	01000110	F	102	01100110	f
39	00100111	'	71	01000111	G	103	01100111	g
40	00101000	(72	01001000	H	104	01101000	h
41	00101001)	73	01001001	I	105	01101001	i
42	00101010	*	74	01001010	J	106	01101010	j
43	00101011	+	75	01001011	K	107	01101011	k
44	00101100	,	76	01001100	L	108	01101100	l
45	00101101	-	77	01001101	M	109	01101101	m
46	00101110	.	78	01001110	N	110	01101110	n
47	00101111	/	79	01001111	O	111	01101111	o
48	00110000	0	80	01010000	P	112	01110000	p
49	00110001	1	81	01010001	Q	113	01110001	q
50	00110010	2	82	01010010	R	114	01110010	r
51	00110011	3	83	01010011	S	115	01110011	s
52	00110100	4	84	01010100	T	116	01110100	t
53	00110101	5	85	01010101	U	117	01110101	u
54	00110110	6	86	01010110	V	118	01110110	v
55	00110111	7	87	01010111	W	119	01110111	w
56	00111000	8	88	01011000	X	120	01111000	x
57	00111001	9	89	01011001	Y	121	01111001	y
58	00111010	:	90	01011010	Z	122	01111010	z
59	00111011	;	91	01011011	[123	01111011	{
60	00111100	<	92	01011100	\	124	01111100	
61	00111101	=	93	01011101]	125	01111101	}
62	00111110	>	94	01011110	^	126	01111110	
63	00111111	?	95	01011111	_	127	01111111	

Fiche n°3.

Méthodes de la bibliothèque **LiquidCrystal.h**.

LiquidCrystal.h permet à une carte Arduino de contrôler un afficheur LCD piloté par un "chipset" de type Hitachi HD44780 ou un compatible qui équipe sur la plupart des écrans LCD alphanumérique. Elle autorise en outre aussi bien la commande avec 4 bits que par le mode 8 bits.

LiquidCrystal lcd(RS, RW, Enable, d0, d1, d2, d3, d4, d5, d6, d7);

Déclaration globale pour créer un objet **lcd** de type **LiquidCrystal**.

RW : Paramètre facultatif. Dans ce cas relier RW à la masse d'Arduino.

d0, d1, d2, d3 sont également facultatifs. Si omis pilotage quatre bits. Les diverses syntaxes possibles sont :

LiquidCrystal lcd(RS, Enable, d4, d5, d6, d7);

LiquidCrystal lcd(RS, RW, Enable, d4, d5, d6, d7);

LiquidCrystal lcd(RS, Enable, d0, d1, d2, d3, d4, d5, d6, d7);

LiquidCrystal lcd(RS, RW, Enable, d0, d1, d2, d3, d4, d5, d6, d7);

(Dans cet exemple **lcd** est l'identificateur de l'objet déclaré)

lcd.begin(Cols,Lignes); (**Déclaration dans void setup**())

Précise les dimensions de la matrice de caractères de l'écran.

Doit être appelée avant les autres instructions de la bibliothèque.

lcd.setCursor(Col,Ligne);

Définit la position du curseur où commencera la prochaine écriture sur l'écran. (0 est la première ligne ou la première colonne)

lcd.cursor(); **lcd.noCursor**();

Valide ou supprime l'affichage du curseur de position d'écriture.

lcd.home();

Place le curseur de la prochaine écriture en haut et à gauche.

lcd.clear();

Efface tous les caractères de l'écran LCD.

Comme pour **home** positionne le curseur en haut

Curseur de Blink

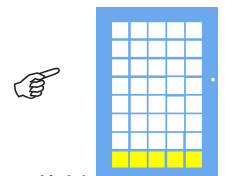
et à gauche de l'écran du module d'affichage.

lcd.blink(); **lcd.noBlink**();

Valide ou supprime l'affichage du gros curseur clignotant qui remplit intégralement la matrice 5/8.

Le curseur simple est visible en permanence s'il est validé.

... / ...



Fiche n°4.

`lcd.noDisplay();` `lcd.display();`

`noDisplay` efface l'écran et le curseur mais conserve l'état en mémoire.
`display` rétablit l'affichage du texte et du curseur s'il est validé.

`lcd.scrollDisplayLeft();` `lcd.scrollDisplayRight();`

Décale l'intégralité de l'affichage, y compris le curseur d'une position à gauche ou à droite. Les caractères "entrants sont des "espaces".

`lcd.leftToRight();` `lcd.rightToLeft();`

Définit le sens de décalage du curseur de la gauche vers la droite (*Valeur par défaut*) ou de la droite vers la gauche lors d'un affichage.

`lcd.autoscroll();`

Chaque caractère sera écrit à la position actuelle du curseur, immédiatement suivi d'un décalage d'une position pour tout l'écran. Le décalage de l'écran est fonction du sens imposé par `scrollDisplayLeft` ou `Right`. Le recyclage interne des caractères n'est pas spécialement évident. L'instruction `lcd.noAutoscroll();` annule ce mode de fonctionnement et retourne au défilement classique.

`lcd.print();`

Cette fonction est analogue au `print` de la *bibliothèque Serial* et présente un comportement similaire pour les divers formats reconnus. Elle permet d'afficher des textes et des nombres représentatif des valeurs de variables sur un module LCD.

`lcd.print("Texte");` ou `lcd.print('Caractère');`

`lcd.print(Data);` Affiche la valeur numérique de la variable *Data*.

`lcd.print(Data, BASE);`

L'option *BASE*
est facultative.

BIN : Binaire pur.

DEC : Décimal.

HEX : Hexadécimal.

OCT : Octal.

Data : `char`, `byte`, `int`, `long`, `float` ou `string`

`lcd.write();`

Cette fonction est orientée caractère. Elle fait afficher au module LCD le caractère **dont on fournit le code ASCII en décimal**. Si la valeur du paramètre est précédée de "**B**" le code est alors exprimé en binaire, les zéros en tête peuvent être omis. Exemples de codage :

`lcd.write(57);` (Affiche "9" car par défaut paramètre en décimal)

`lcd.write(B111101);` (Affiche "=", paramètre en Binaire)

`lcd.write(B01000110);` (Octet complet, affiche "F")

Fiche n°2.

Différents types de données utilisables avec Arduino.

TYPE	Valeurs possibles	BITS	Octets
<code>char</code>	-128 à +127	8	1
<code>int</code>	-32768 à +32767	16	2
<code>long</code>	-2147483648 à +2147483647	32	4

Variables Décimales.

<code>float</code>	$3,4 * 10^{(-38)}$ à $+3,4 * 10^{(38)}$	32	4
<code>double</code>	$1,7 * 10^{(-308)}$ à $+1,7 * 10^{(308)}$	64	8

La gestion des variables de type double dans Arduino est exactement la même que celle des variables de type float, sans gain de précision.

Variables NON SIGNÉES : unsigned

<code>char</code>	0 à 255	8	1
<code>int</code>	0 à 65535	16	2
<code>long</code>	0 à 4294967295	32	4

Variables propres au langage C d'Arduino

<code>byte</code>	0 à +255	8	1
<code>word</code>	0 à +65535	16	2
<code>boolean</code>	0 ou 1	1	1

Variables booléennes

`boolean` Nom Variable = `FALSE` ou `TRUE`

Toute variable peut servir de variable booléenne :

- Si elle vaut 0 ce sera interprété comme `FALSE`.
- Si elle est $\neq 0$ ce sera interprété comme `TRUE`.

Conversion Numérique analogique pour les entrées A0 à A5

<code>CAN</code>	0 à 1023	10	2
------------------	----------	----	---

Fiche n°5.

Créer des caractères personnalisés pour un LCD.

C'est une spécificité particulièrement intéressante qui donne au programmeur la possibilité à tout moment dans le programme d'utiliser des caractères "graphiques" qu'il se sera créé dans une matrice de type 5 x 8 pixels qui est le format d'affichage des caractères standards.

`lcd.createChar(Numéro, Tableau);`

Procédure qui permet de générer simultanément jusqu'à huit caractères personnalisés désignés par Numéro dont l'ordre s'étend de 1 à 8. L'apparence de chaque caractère se définit par un tableau de huit octets chacun relatif à l'une des lignes. Seuls les cinq bits de poids faibles décrivent les pixels de la matrice du caractère. Pour afficher un caractère personnalisé, utiliser l'instruction :

`lcd.write(Numéro);`

L'exemple de construction d'un caractère spécifique donné en Fig.1 est relatif à la génération du caractère "ç". On peut coder le tableau de définition en binaire ou en décimal. (Et mélanger les deux) En binaire il est inutile de donner les zéros en tête. Ci-contre on détaille l'exemple pour le "ç".

// Définition des matrices de représentation avant Void SETUP().

// On peut en définir autant que l'on veut.

`byte Yminuscule[8] = {0,0,17,17,17,15,1,14};`

`byte Agrave[8] = {8,4,14,1,15,17,15,0};`

`byte Ugrave[8] = {8,4,17,17,17,19,13,0};`

`byte Acircconflexe[8] = {4,10,0,14,1,15,17,15};`

`byte Icircconflexe[8] = {4,10,0,4,4,4,14,0};`

`byte Ocircconflexe[8] = {4,10,0,14,17,17,14,0};`

`byte Eegu[8] = {2,4,14,17,31,16,14,0};`

`byte Egrave[8] = {8,4,14,17,31,16,14,0};`

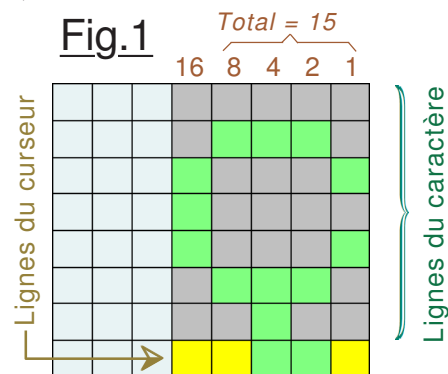
`byte Ecircconflexe[8] = {4,10,0,14,17,31,16,14};`

`byte Gminuscule[8] = {0,0,B01111,B10001,B10001,B01111,1,14};`

`byte Jminuscule[8] = {4,0,15,2,2,2,18,12};`

`byte Pminuscule[8] = {0,0,B11110,B10001,B10001,B11110,16,16};`

`byte Qminuscule[8] = {0,0,15,B10001,B10001,B01111,B00001,B00001};`

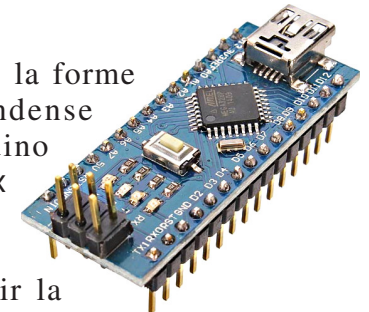


NOTE : "Sur Internet" Numéro est supposé varier entre 0 et 7 et non entre 1 et 8. Mais l'expérience semble démontrer le contraire.

Fiche n°7.

Carte Arduino NANO.

L'arduino NANO se présente sous la forme d'une minuscule carte qui condense l'intégralité des fonctions d'une Arduino UNO tout en ne mesurant que 1,9 cm x 4,5 cm. La NANO utilise l'ATmega328 en version CMS. Les broches d'utilisation sont séparées pour pouvoir la placer sur une platine d'essais classique. Arduino NANO peut être alimentée soit par le connecteur Mini-USB soit en externe avec +6V à +20V non régulé sur la broche VIN.



ATTENTION : Pas de VIN simultanément avec la liaison Mini-USB ou la carte électronique sera détruite.

Alimentée par le connecteur Mini-USB la carte fournit environ 4,2v sur la broche 5V pour alimenter des modules périphériques. Cette broche peut également être alimentée en +5Vcc simultanément avec la prise Mini-USB. On peut ainsi alimenter la carte par la broche 5V, sur son électronique d'application, tout en branchant en parallèle la ligne USB pour programmer sur site et dialoguer avec le Moniteur de l'IDE.

Caractéristiques de base :

14 broches binaires. (Dont 6 fonctionnant en PWM et deux servant à téléverser les programmes.)

8 broches d'entrées Analogiques dont 6 pouvant fonctionner en E/S. Courant maximal par broche de sortie : 40 mA. (Total MAX : 200mA)

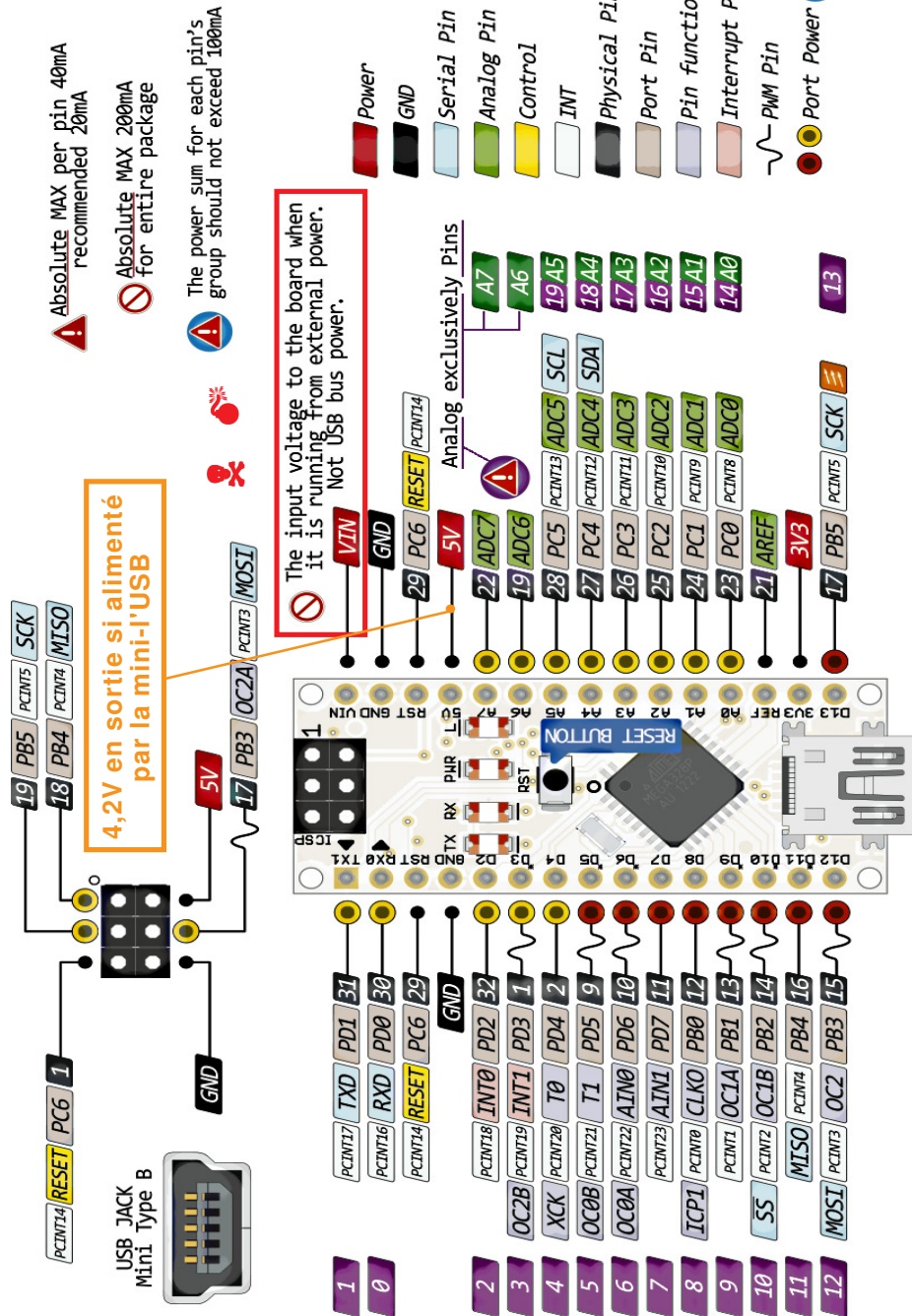
L'ATmega328 a 32 Ko, (Avec 2 KB utilisé pour le bootloader).

L' ATmega328 a 2 Ko de SRAM et 1 Ko de mémoire EEPROM.

Par rapport à la carte Arduino UNO la NANO présente deux entrées Analogiques supplémentaires A6 et A7. Elles ne peuvent pas être utilisées en E/S binaires, mais uniquement en entrées analogiques et ne disposent pas de résistances PUL-UP internes. Inutile de les déclarer en entrée, on les utilise directement avec la syntaxe standard `analogRead(20)` et `analogRead(21)`.

L'impédance d'une entrée logique est de 5MΩ et d'une entrée analogique de 100MΩ. la résistance de PUL-UP interne est comprise entre 20kΩ et 50kΩ. ... / ...

Fiche n°8.



Page 5

Fiche n°6.

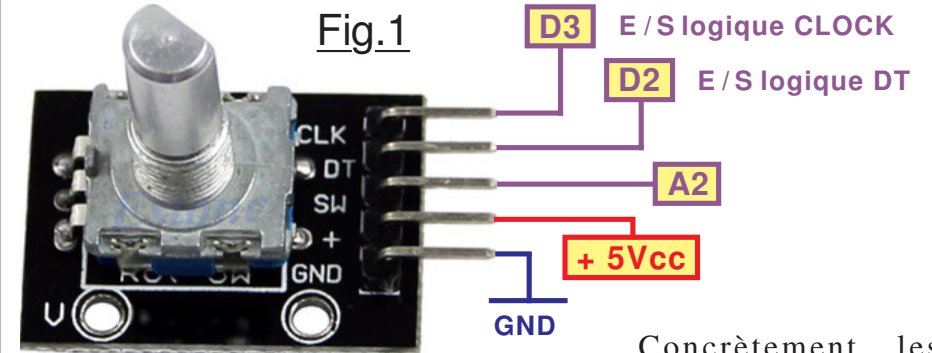
Encodeur rotatif KY-040.

Le **KY-040** est un codeur **sans butée** à 20 points par tour pourvu d'un "bouton poussoir" par appui sur la tige de commande centrale. La sortie se fait par deux lignes pilotées par des capteurs de type codage Gray. La Fig.1 donne le schéma des branchements à réaliser.

Caractéristiques techniques du module KY-040 :

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

Fonctionnement :



Concrètement les trois sorties sont alimentées au + 5Vcc par des résistances de 10 kΩ. La sortie **SW** est celle de l'inverseur piloté par appui sur la tige du rotor. Les deux sorties **CLK** et **DT** sont en réalité deux sorties classiques avec déphasage de 90° souvent nommées **A** et **B** pour ce type de capteur. Le déphasage de 90° électriques des signaux **CLK** et **DT** permet de déterminer le sens de rotation. (Voir Fig.2) *Le capteur est traité par interruptions.*

A change d'état avant B B change d'état avant A

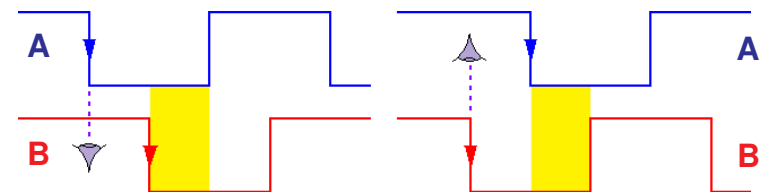


Fig.2

Fiche n°9.

Résumé de quelques conseils pour programmer de façon méthodique et structurée.

- Avant de programmer *il faut absolument consacrer du temps pour déterminer relativement finement ce que l'on désire exactement obtenir et en prouver la faisabilité.*

La première qualité d'un logiciel quel qu'il soit et quelle que soit sa complexité est sa LISIBILITÉ.

- Les lignes qui doivent facilement se retrouver seront terminées par // @@@@.
- Placer un maximum de commentaires pour expliciter ou pour justifier certains calculs, certains choix etc.
- Pour une lisibilité maximale du listage, *toujours choisir avec beaucoup d'attention les noms des identificateurs des constantes et des variables* avec des noms qui "parlent".
- Toutes les déclarations sont placées en tête de programme.
- Éviter autant que possible les séquences trop longues imposant des défilements du listage sur l'écran de l'ordinateur. Une procédure ne devrait jamais dépasser la hauteur verticale possible.
- Préciser en tête de listage le but précis du programme, son utilisation, ainsi que le résumé de son comportement attendu.
- Indiquer en tête de listage les branchements à effectuer.
- Ajouter en tête de listage *la date de dernière modification* du programme, la taille du code et celle des données en mémoire dynamise. (*Fonction de la version à préciser du compilateur.*)
- Il est fortement recommandé de placer toute fonction ou procédure avant celle qui y fait appel. (*Et avant void setup et void loop.*)
- Pour structurer les constantes et les variables utilisées :
 - * Les classer par **type** et globalement par ordre alphabétique.
 - * Les lister par tailles croissante : **boolean, char, byte, int, long, float, double, tableaux** ...
- Affectation judicieuse des Entrées / Sorties** pour optimiser l'utilisation de l'ATmega328 et minimiser la taille du code.
- Dans **void setup** initialiser en premier les broches d'E/S. Vers la fin initialiser alors les variables en classant les instructions par l'ordre alphabétique des identificateurs. ... / ...

Fiche n°11.

Limitation du courant sur une sortie binaire.

Sur l'exemple de la Fig.1 on suppose que la sortie binaire **Dn** fournit un état "1" constant, c'est à dire environ **+5Vcc**. En **A** ... *ça chauffe* car le courant **I** débité par la sortie binaire **Dn** n'est pas limité et va atteindre une valeur déraisonnable. La diode électroluminescente **L** va probablement passer de vis à trépas et **Dn** être endommagée. En **B** tout ira bien si la valeur de la résistance **R** est suffisante. Supposons que l'on désire limiter le courant à 10mA : Nous savons que :

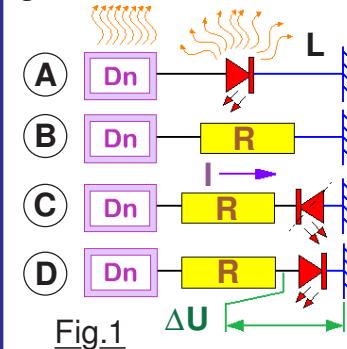


Fig.1

La diode électroluminescente **L** va probablement passer de vis à trépas et **Dn** être endommagée. En **B** tout ira bien si la valeur de la résistance **R** est suffisante. Supposons que l'on désire limiter le courant à 10mA : Nous savons que :

$$I = U / R \text{ donc } R = U / I$$

La valeur à adopter pour **R** sera de $5 / 0.01 = 500\Omega$. Mais dans la pratique une telle valeur n'est pas courante. On adoptera **560Ω** par exemple. Dans l'exemple **C** le courant est ... nul, car la *LED est avant tout une diode* et ne laisse passer le courant que dans le sens de la flèche de son symbole. (*Sens du courant conventionnel.*)

Enfin le cas D est typique du montage d'un pilotage d'une LED par une sortie binaire d'Arduino. Lorsqu'une diode électroluminescente est conductrice, la tension **ΔU** que l'on trouve à ses bornes est directement influencée par le courant qui la traverse (*Qui doit rester dans les limites imposées par sa fiche technique.*) et par sa couleur. Par exemple pour une LED de couleur rouge **ΔU** avoisine 0,8 à 1,5V. La valeur à adopter pour **R** sera de :

$$I = (U - \Delta U) / R \text{ donc } R = (U - \Delta U) / I$$

EXEMPLE avec **ΔU** ≈ 1,3V et **I** ≈ 10mA soit 0,01A :

R sera de $(5 - 1,3) / 0.01 = 370\Omega$. (*Valeur courante.*)

La théorie et les calculs c'est bien et seront souvent indispensables. Mais il faut savoir que les diodes électroluminescentes présentent des rendements très différents en fonction de leurs technologies et de leurs couleurs. Si l'approvisionnement se fait en ligne par lot

NOTE : **L** est ici utilisé pour **LED**. En français on devrait dire **DEL**, mais dans les catalogue c'est le sigle **LED** qui s'impose !

Fiche n°12.

de diverses références, il est assez rare que l'on dispose de leurs fiches techniques de caractéristiques. Aussi la détermination de **R** est à faire expérimentalement. *Le plus simple consiste à commencer par une valeur de 10kΩ sous +5Vcc par exemple, et de diminuer progressivement jusqu'à obtenir un bon compromis entre la consommation et la luminosité qui en résulte.*

➤ Cas d'un signal "rectangulaire".

Lorsque la sortie binaire **Dn** délivre un signal découpé du genre **PWM** par exemple, il faut tenir compte de son **Rapport cyclique**. Par définition c'est le temps de l'état "1" divisé par le temps de la période. La période, c'est le temps de l'état "1" plus le temps à l'état "0". Par exemple sur la Fig.2 le **Rapport cyclique** est de 1 / (1 + 3) soit 0.25 ou 1/4. *Si la fréquence du signal PWM dépasse les 20Hz, nous avons l'impression d'un éclairage continu.* Mais comme la LED ne s'éclaire qu'un quart du temps, sa luminosité est quatre fois moindre. Pour retrouver l'éclairage d'un courant qui serait continu il faut une valeur de résistance **R** égale à :

$$R = ((U - \Delta U) / I) * \text{Rapport cyclique}$$

EXEMPLE avec encore $\Delta U \approx 1,3V$:

R sera de $((5 - 1,3) / 0,01) * 0,25 = 92,5\Omega$. (Prendre 100Ω.)

⚠ DANGER ⚡ : *Si le rapport cyclique est faible, le courant calculé durant l'état "1" de la sortie **Dn** peut dépasser les 40mA MAXIMUM que peut ou doit fournir une sortie binaire* sur Arduino. Ne pas oublier non plus que si plusieurs LEDs sont allumées simultanément sur plusieurs sorties, le courant total ne doit pas dépasser 100mA. Il serait possible d'amplifier "la sortance" de **Dn** avec un transistor **T** utilisé en amplificateur, mais alors il faudrait veiller à ne pas dépasser le courant de point et la puissance moyenne maximale acceptable par la LED.

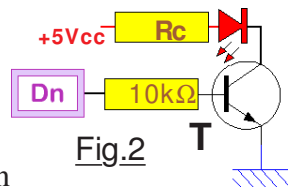


Fig.2

Fiche n°10.

- 📖 Dans **void setup** initialiser en premier les broches d'E/S. Vers la fin initialiser alors les variables en classant les instructions par l'ordre alphabétique des identificateurs.
- 📖 *Optimiser le code doit virer à l'obsession :*
 - * Optimiser le code c'est **Minimiser la taille du code** objet qui encombrera la mémoire,
 - * Optimiser le code c'est **Minimiser le temps d'exécution** d'une séquence critique. (Boucle d'affichage par exemple.)
- 📖 **Optimisation de la taille des variables : il est absolument indispensable de choisir le type de chaque variable pour en minimiser la place occupée en mémoire dynamique et en accélérer le traitement.** C'est un impératif absolu. Du bon choix du type des données dépend considérablement la taille occupée par le programme et le temps d'exécution. **ATTENTION**, les tableaux sont les données les plus voraces en espace utilisé et les textes sont intrinsèquement des tableaux. Aussi, *si la mémoire non volatile EEPROM n'est pas utilisée pour mémoriser des données* à conserver sur coupure alimentation, **y loger un maximum de textes** à afficher sur les écrans.
- 📖 Pour diverses raisons **la boucle de base doit "tourner" à grande vitesse.** (Rapidité de prise en compte d'une consigne opérateur, rapidité des affichages etc.) Le programme étant "terminé", il est fortement conseillé d'en mesurer la fréquence d'exécution.
- 📖 Un moyen incontournable pour diminuer les temps de développement consiste à **SIMULER LES DONNÉES** au lieu de les mesurer, et à diminuer certains délais de comportement.
- 📖 Il est conseillé en développement de placer dans la boucle de base un témoin logique pour s'assurer à tout moment que le logiciel n'est pas enlisé dans une séquence très longue ou infinie. (Il suffit de faire clignoter une LED par exemple.)
- 📖 **Si des paramètres dans des procédures sont susceptibles d'être modifiés** à la mise au point, il est de loin conseillé de **les définir dans des constantes en tête de listage.**
- 📖 **On devrait toujours placer la séquence de vérification de non collision de PILE en début de programme.**

Fiche n°13.

Afficheur graphique OLED 1,3 pouce.

Les références d'afficheurs graphiques sont kyrielles, avec des dimensions et des caractéristiques très différentes d'un modèle à l'autre. Leurs prix d'achat aussi incitent à éliminer ceux qui sont en couleur, les grandes définitions, les écrans tactiles etc. Le choix pour cette application s'est porté sur l'écran OLED de la Fig.1 monochrome blanc ou bleu pilotable par I2C de définition 128 x 64 Pixels et de diagonale 1,3 Pouces. Seulement deux broches de pilotage sont nécessaires pour le gérer. Muni d'un contrôleur SSD1306 ils sont lumineux, donc sans rétro-éclairage, de couleur blanche ou bleue selon la référence approvisionnée. (REMARQUE : Noter avant de passer commande qu'il existe aussi en 0,96 pouces de diagonale. C'est bien celui qui fait 1,3 pouces dont il est question dans cette fiche.) **ATTENTION : Certaines références ne sont pas compatibles avec U8glib.h** donc vérifier à la commande.

J'ai approvisionné le mien à l'adresse suivante :

https://www.amazon.fr/gp/product/B07FYG8MZN/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&psc=1



Fig.1

ATTENTION DANGER : L'afficheur de la Fig.1 est disponible sur plusieurs sources dans le commerce en ligne. Toutes sont équivalentes et compatibles avec U8glib.h. Toutefois il faut bien faire attention car en fonction des adresses d'approvisionnement **les deux broches GND et VCC sont inversées.**

La bibliothèque <U8glib.h> est fournie dans <Documents> et accompagnée du fichier Bibliothèque U8glib.pdf formaté pour être imprimé en Recto/Verso et réaliser un petit livret au format A5 qui en résume sur vingt pages les méthodes. Également fourni Réaliser un petit livret.pdf pour vous aider à l'assembler.

Fiche n°15.

BUS série au standard I2C. (1/2)

Développé initialement en 1982, le bus I2C s'est largement imposé dans le domaine des microprocesseurs, et des applications industrielles. Sa désignation dérive de **Inter-Integrated Circuit**. Il fut à l'origine conçu pour des applications de domotique et d'électronique domestique.

La norme I2C est basée sur un **bus série synchrone bidirectionnel** fonctionnant en "half-duplex", où plusieurs périphériques maîtres ou esclaves peuvent communiquer entre eux. Les dialogues ont toujours lieu entre un seul maître et un ou tous les esclaves présents et l'échange de données est toujours déclenché à l'initiative du maître. (Jamais de maître à maître ou d'esclave à esclave.)

➤ Constitution matérielle du bus I2C.

Outre GND la ligne n'utilise que deux fils "bidirectionnel" :

- **SDA** (**S**erial **D**ata **L**ine) : Ligne de données bidirectionnelle.
- **SCL** (**S**erial **C**lock **L**ine) : Ligne d'horloge pour la synchronisation également de type bidirectionnel.

Les deux lignes sont maintenues à l'état logique "1" par un niveau de tension +VDD à travers des résistances de forçage. (Pull-Up.) Dans le standard I2C le nombre maximal de périphériques est limité à 128 par le nombre d'adresses disponibles, 7 Bits d'adressage et un Bit R/W. (Lecture ou Écriture.) Dans le cas d'Arduino c'est la carte ATmega328 qui sera toujours le Maître et les modules périphériques les esclaves. Si le programme d'application doit intégrer une ligne I2C, comme représenté sur la Fig.1 ci-dessous ce sont **A4 et A5** qui sont respectivement affectées à **SDA et SCL** contrainte imposée par les bibliothèques qui accompagnent les modules du commerce.

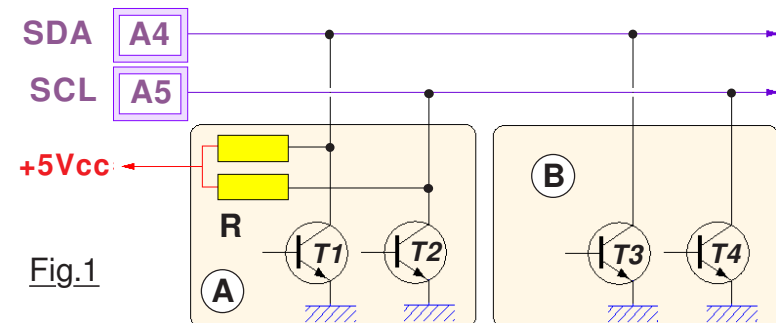
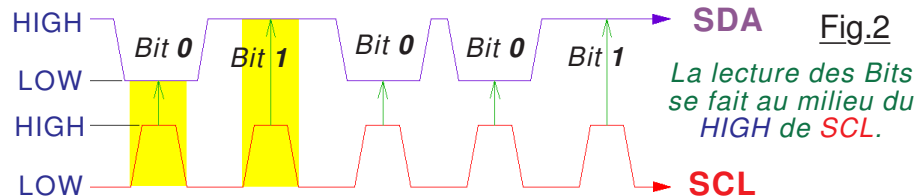


Fig.1

Fiche n°16. BUS série au standard I2C. (2/2)

➤ Signaux échangés sur un bus I2C.

Le niveau **HIGH** ou **LOW** de la ligne **SDA** doit être maintenu stable pendant le niveau **HIGH** sur la ligne **SCL** servant à déclencher la lecture successive des bits du protocole de dialogue. (Voir la Fig.2)



Comme montré sur la Fig.1 les équipements sont connectés au bus par des électroniques de type drain ouvert. (Ou collecteur ouvert.) Fonctionnant en ET câblés deux périphériques tels que **A** et **B** peuvent "parler" simultanément. Dans ce cas un état logique "0" "écrase" un état logique "1". Pour caractériser ce genre d'incident potentiel sur un bus I2C on utilise le vocable :

- L'état logique "0" **LOW** est un état dominant,
- L'état logique "1" **HIGH** est un état récessif.

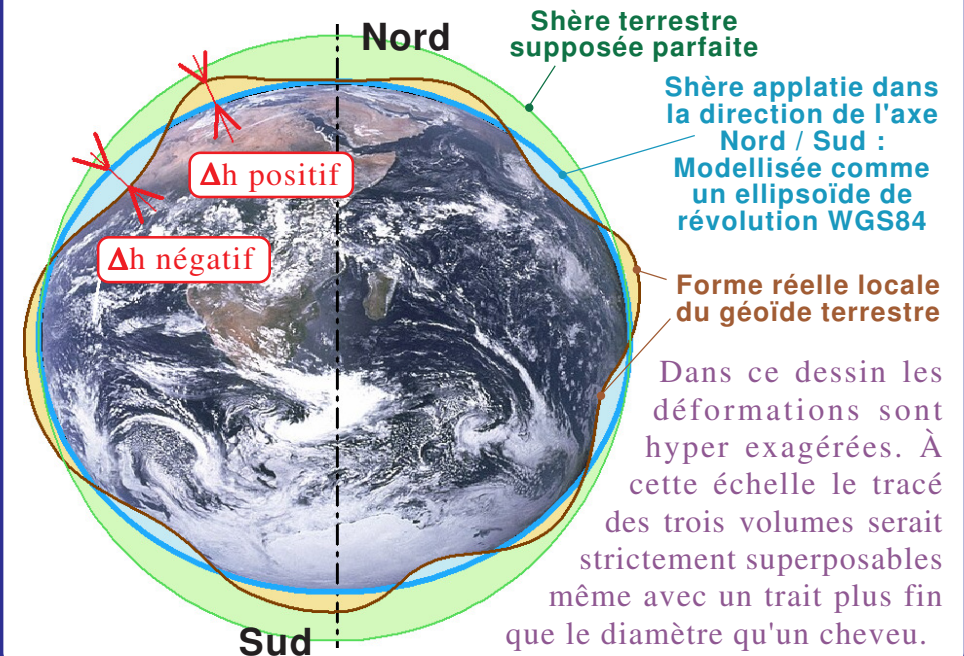
Lorsque le bus n'est pas utilisé, il est forcé niveau haut par les résistances telles que **R** de l'un des modules connectés. Il suffit d'un seul rappel à **+5Vcc** pour que la ligne fonctionne. Si plusieurs modules sont pourvus de résistances de forçage, le courant qui devra être drainé par l'ATmega328 et les électroniques branchées sera plus important mais reste généralement faible car le nombre de périphériques est classiquement faible pour des applications ordinaires. Pour les modules dédiés à Arduino les vitesses de transmission sont généralement comprises entre 100kb/s et 400kb/s.

➤ Les bibliothèques de programme.

Les concepteurs de modules électroniques dialoguant en I2C fournissent des bibliothèques dédiées à leurs produits. Outre les "library" propres à chaque référence commerciale, **Wire.h** est spécialisée pour gérer sur Arduino les protocoles I2C et devra parfois s'ajouter à celle qui accompagne un module électronique spécialisé. C'est elle qui impose l'usage de **A4** et **A5**.

Fiche n°14. Ellipsoïde WGS84.

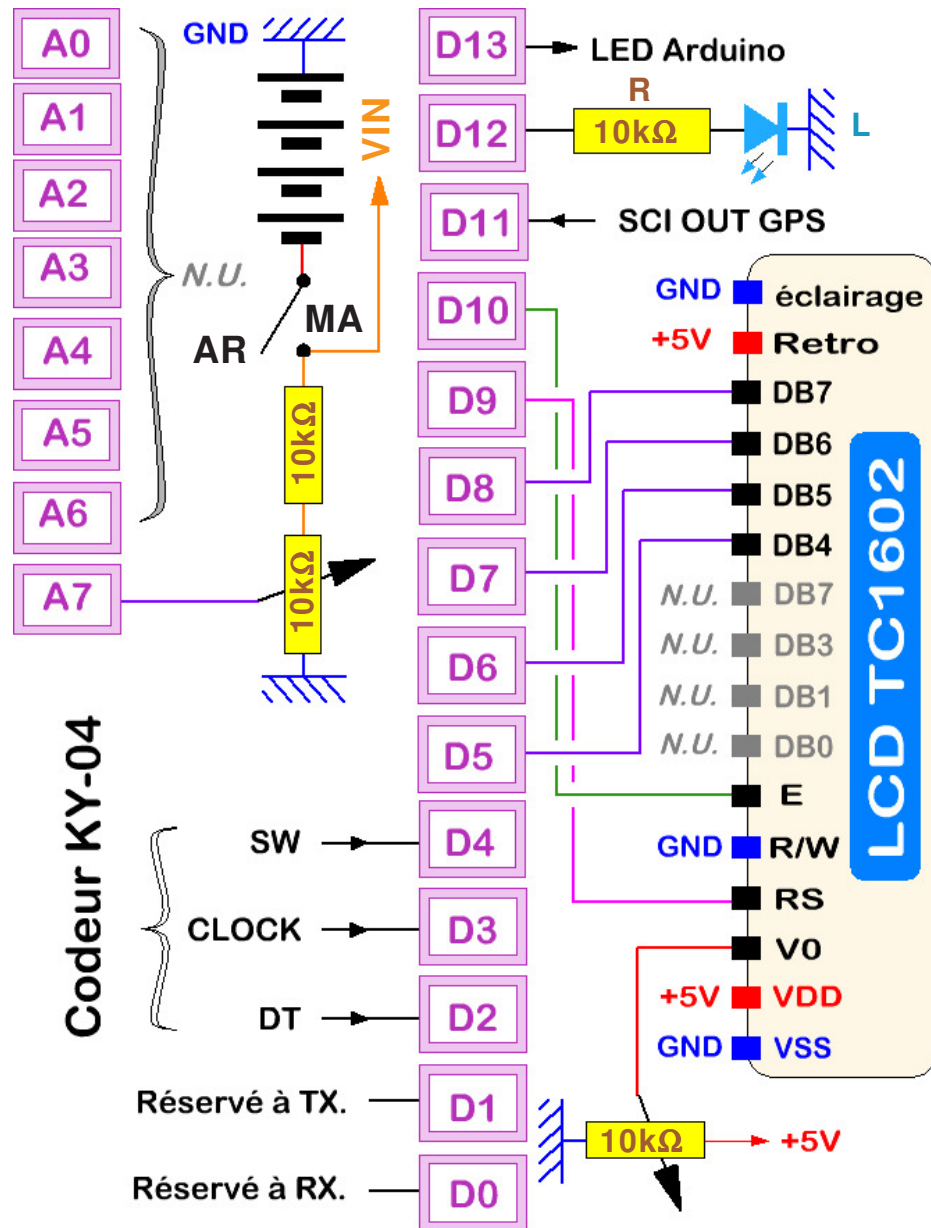
Globalement, la forme générale de la Terre s'apparente à une sphère avec un axe Nord/Sud légèrement inférieur à celui mesuré à l'équateur. De révolution, son volume général est très proche de celui d'un ellipsoïde d'axe Nord/Sud. WGS84 est un système de coordonnées terrestres, basé sur un géoïde de référence prenant la forme d'un ellipsoïde de révolution. (WGS84 : World Geodesic System révision de 1984) Le système est défini par un ensemble de paramètres primaires et secondaires **qui précisent la forme de l'ellipsoïde de la Terre**, sa vitesse angulaire, et sa masse et un modèle détaillé de la pesanteur terrestre. Les paramètres secondaires sont nécessaires pour déterminer les orbites des satellites de navigation GPS. **La position déterminée par un récepteur GPS est en fait une latitude, une longitude et une altitude dans le système WGS84.** Pour simplifier, on peut considérer que la correction de la hauteur du géoïde Δh de la Fig.36E en page 18 est le décalage qui existe entre la forme locale du géoïde terrestre et l'ellipsoïde de référence WGS84.



Fiche n°17.

Schéma de l'application n°1.

La résistance **R** est à choisir en fonction du rendement de **L**.



Fiche n°19.

Textes du dialogue H/M USB en EEPROM.

La zone rose contient les informations relatives aux 10 PI.

Les accentués dans les cadres jaunes sont codés spécifiquement.

ADRS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	1	2	3	4	4	4	4	N	1	2	3	4	5	6	7	8
0016	W	M	a		M	a	i	s	o	n	.					
0032		1	2	3	4	5	6	7	N	1	2	3	4	5	6	7
0048	8	E	>	>	>	>	>	>		F	I	N				
0064			1	2	3	4	5	6	7	N	1	2	3	4	5	6
0080	7	8	W	T	E	S	T									
0096				7	6	5	4	3	2	1	S	8	7	6	5	4
0112	3	2	1	E	t	e	s	t		d	e		c	o	r	r
0128	e	c	t	i	M	M	M	M	M	M	M	S	5	5	5	5
0144	5	5	5	5	W	C	i	b	l	e		?	?	?		n
0160	u	1	1	e	.	1	2	3	4	5	6	7	S	1	2	3
0176	4	5	6	7	8	W	T	o	t	o	c	h	e	.		
0192							1	2	3	4	5	6	7	N	1	2
0208	3	4	5	6	7	8	W	w	w	w	w	w	w	w	w	w
0224	w	w	w	w	w	w	w	1	2	3	4	5	6	7	S	1
0240	2	3	4	5	6	7	8	E	u	u	u	u	u	u	u	u
0256	u	u	u	u	u	u	u	u	1	2	3	4	5	6	7	S
0272	1	2	3	4	5	6	7	8	W	<	<	<	<	<	<	<
0288	<	<	<	<	<	<	<	<	<	1	2	3	4	5	6	7
0304	S	1	2	3	4	5	6	7	8	W	p	p	p	p	p	p
0320	p	p	p	p	p	p	p	p	p	p		V	e	r	s	i
0336	o	n		d	u		P	I	L	E		-		T	A	S
0352		=			P	I		d	'	a	r	r	i	v	é	e
0368	?		P	I		o	r	i	g	i	n	e		?		
0384	C	o	m	m	a	n	d	e	:	P	I		à			
0400	e	f	f	a	c	e	r		?		L	a	t	i	t	u
0416	d	e	?		L	o	n	g	i	t	u	d	e	?		
0432		N	o	m	d	e		l	a		c	i	b	l	e	
0448	?	?			o	u	,		:		C	e	M			
0464	E	N	U	.		P	I		s	u	r		R	E	S	E
0480	T		?		A		o	u		a	:		A	j		
0496	o	u	t	e	r		u	n		P	I	.		D	o	

Fiche n°20.

Textes du dialogue H/M USB en EEPROM.

Les accentués dans les cadres jaunes sont codés spécifiquement. Les options du programme sont en zone bleue. Il ne reste plus qu'un seul octet de disponible repéré par '*'.
 Page 11

ADRS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0512	u	d	:		D	é	p	l	a	c	e	r		u		
0528	n	P	I	.	E		o	u		e	:		E			
0544	f	f	a	c	e	r		u	n	P	I	.		L		
0560	o	u		l	:		L	i	s	t	a	g	e	d		
0576	e	s		P	I		e	n		E	E	P	R	O	M	.
0592		M		o	u		m	:	M	é	m	o	i	r		
0608	e		d	y	n	a	m	i	q	u	e	.		Q		o
0624	u	q	:		Q	u	i	t	t	e	r		l	e		
0640	s	s	a	i	s	i	e	s	.		S		o	u		
0656	s	:		S	i	l	e	n	c	i	e	u	x	.		
0672	N	u	m		N	o	m	b	r	e		i	n	c	o	r
0688	r	e	c	t	.		D	é	p	l	a	c	e	r		
0704	u	n		P	I	:		E	c	h	a	n	g	e		
0720	a	v	e	c		?	?	?		I	l		n	e	r	
0736	e	s	t	e		q	u	'	u	n		s	e	u	l	
0752	P	.	I	.		E	f	f	a	c	e	m	e	n	t	
0768	e	n		c	o	u	r	s	.		P	I		à		e
0784	f	f	a	c	e	r		?		V	a	l	e	u	r	
0800	i	n	c	o	r	r	e	c	t	e		!		E	E	P
0816	R	O	M		p	l	e	i	n	e		!		F	i	n
0832		d	u		d	i	a	l	o	g	u	e		U	S	B
0848	.		B	I	P	s		O	U	I	N	O	N		C	o
0864	m	m	a	n	d	e		i	n	c	o	r	r	e	c	t
0880	e	.		L	o	n	g	u	e	u	r		0		o	u
0896	>		l		!		'	O	'		p	o	u	r		
0912	O	U	I	:		E	f	f	a	c	e	r		v	r	
0928	a	i	m	e	n	t		t	o	u	s		l	e	s	
0944	P	I		?		G		o	u		g	:		G	o	
0960	m	m	e	r	-		l	.		I	t	e	m		L	A
0976		L	G		L	B		(L	i	b	e	l	l	é)
0992	?		C	o	u		c	:		P	I		à			
1008		C	o	r	r	i	g	e	r		?		*	00	01	01 05

Fiche n°18.

Bruiteur actif.

ATTENTION : Ces éléments vendus généralement par lots de cinq existent en modèle de type *Actif* ou *Passif* et leur utilisation avec Arduino n'est pas du tout la même. Leur apparence est strictement identique, aussi lors d'une commande bien vérifier leur type. Pour simplifier et compacter le code objet, *c'est*

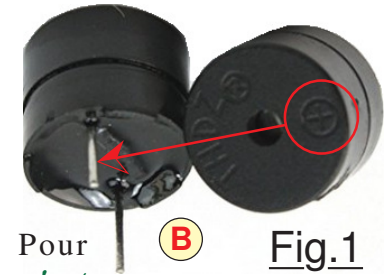


Fig.1

la version active qui s'impose dans notre application. Son seul inconvénient est de n'offrir qu'une seule tonalité d'environ 3kHz. Qu'ils soient actifs ou passifs, ils sont polarisés et la Fig.1 précise la borne positive en broche **B**, l'autre devant aller sur **GND**. D'une façon générale ces composants sont à mon sens bien trop "nerveux" et un BIP inattendu peut faire sursauter. Aussi, en Fig.2B j'ajoute en série une résistance de faible valeur pour limiter les décibels. Le tableau ci-contre précise le courant que doit fournir la sortie d'Arduino en fonction du circuit adopté. Avec **100Ω** on divise pratiquement par deux le courant consommé. *Si la valeur de la résistance est trop élevée le buzzer devient inactif.*

5Vcc	
0	100Ω
20mA	12mA

Pour notre application, la limitation en courant est un peu différente, car le schéma adopté et montré en Fig.2A met en série une LED orange et une résistance de **47Ω**. Le niveau sonore est proche de celui obtenu avec la résistance de **100Ω** avec pour avantage d'ajouter l'information visuelle.

C'est la première broche analogique qui est utilisée en sortie binaire et utilisée pour piloter le buzzer actif.

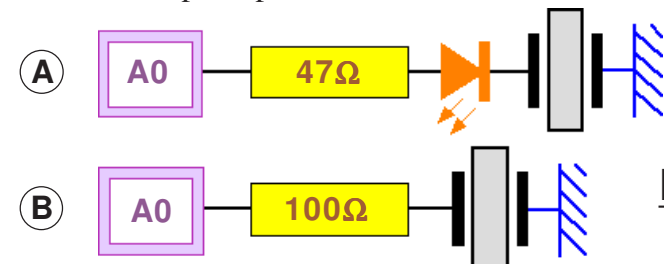
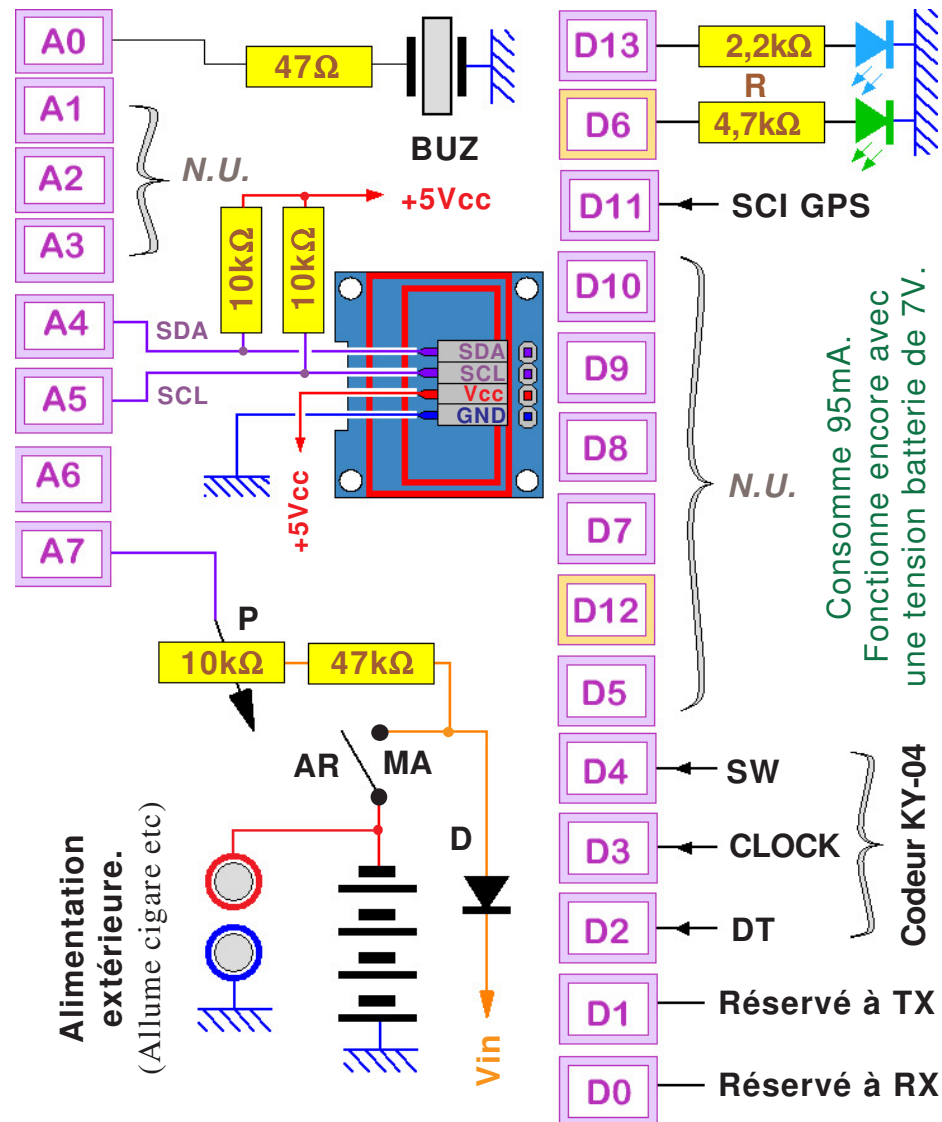


Fig.2

Schéma de l'application n°3.

Les résistances **R** sont à choisir en fonction du rendement des diodes électroluminescente utilisées. La diode **D** protège d'une inversion de tension extérieure. **P** ajuste la précision de la tension affichée en fonction de celle de l'accumulateur rechargeable.



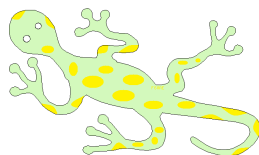
Texte EEPROM du dialogue USB d'Application n°5.

ADRS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	4	4	4	5	5	4	4	N	0	0	1	5	2	1	9	2
0016	E	M	a	M	a	i	s	o	n	.						
0032	4	7	3	7	3	5	9	N	0	0	3	1	2	6	3	
0048	4	E	D	o	n	z	y	.								
0064	4	4	3	7	0	5	1	N	0	0	4	8	0	7		
0080	4	5	E	C	l	a	n	s	a	y	e	s	.			
0096																
0112	3	7	8	E	N	I	M	E	S	.						
0128																
0144	5	0	2	6	E	P	A	R	I	S	(N	T	r		
0160	D	a	m	e)	3	3	8	3	8	6	8	5	1	5	1
0176	0	2	6	6	8	E	S	y	d	n	e	y	.			
0192																
0208	8	0	0	5	2	1	E	J	o	h	a	n	n	e	s	b
0224	u	r	g	.			2	7	1	8	6	2	4	5	1	
0240	0	9	4	3	4	8	1	W	I	l	e	d	e	P		
0256	a	q	u	e	s	.		4	8	5	8	6	9	8	N	
0272	0	0	7	7	4	1	5	4	E	S	t	r	a	s	b	o
0288	u	r	g	.				9	0	0	0	0	0	0		
0304	S	9	0	0	0	0	0	0	0	W	P	o	l	e	S	
0320	U	D	.													
0336	o	n	d	u	P	I	L	E	-	I	A	S				
0352	=	<	?	:	C	e	e	M	E	N	U	A				
0368	:	A	J	o	u	t	e	r	:	C	o	r	r	i		
0384	B	:	D	:	D	e	p	a	c	e	r	:				
0400	g	r	e	D	:	E	f	f	a	c	e	r	:			
0416	u	n	E	:	E	f	f	a	c	e	r	:				
0432	L	i	s	t	e	r	:	f	a	c	e	r	:			
0448	I	s	u	r	R	E	S	E	T	Q	:					
0464	u	i	t	t	e	r	H	:	H	i	v	e	r	K		
0480	:	K	m	x	x	x	x	x	N	u	m	N	o			
0496	m	b	r	e	i	n	c	o	r	r	e	c	t	E	C	

Fiche n°24.

Liste des diverses fiches.

- Fiche n°1 : **TABRE DES CARACTÈRES ASCII** affichables sur LCD.
- Fiche n°2 : Différents types de données utilisables avec Arduino.
- Fiche n°3 : Méthodes de la bibliothèque **LiquidCrystal.h**.
- Fiche n°4 : Méthodes de la bibliothèque **LiquidCrystal.h**.
- Fiche n°5 : Créer des caractères personnalisés pour un LCD.
- Fiche n°6 : *Encodeur rotatif KY-040.*
- Fiche n°7 : *Carte Arduino NANO.*
- Fiche n°8 : *Carte Arduino NANO.*
- Fiche n°9 : *Programmation méthodique et structurée.*
- Fiche n°10 : *Programmation méthodique et structurée.*
- Fiche n°11 : *Limitation du courant sur une sortie binaire.*
- Fiche n°12 : *Limitation du courant sur une sortie binaire.*
- Fiche n°13 : *Afficheur graphique OLED 1,3 pouce.*
- Fiche n°14 : *Ellipsoïde WGS84.*
- Fiche n°15 : *BUS série au standard I2C.*
- Fiche n°16 : *BUS série au standard I2C.*
- Fiche n°17 : *Schéma de l'application n°1.*
- Fiche n°18 : *Bruiteur actif.*
- Fiche n°19 : *Textes du dialogue H/M USB en EEPROM.*
- Fiche n°20 : *Textes du dialogue H/M USB en EEPROM.*
- Fiche n°21 : *Schéma de l'application n°3.*
- Fiche n°22 : Texte du dialogue USB d'Application n°5.
- Fiche n°23 : Texte du dialogue USB d'Application n°5.
- Fiche n°24 : **Table des matères.**



Fiche n°22.

Texte EEPROM du dialogue USB d'Application n°5.

```

4445544N00152192E Ma Maison.
4737359N00312634E Donzy.
4437051N00480745E Clansayes.
4383763N00438378E NIMES.
4885365N00235026E PARIS (Ntr Dame)
3383868S15102668E Sydney.
2619940S02800521E Johannesburg.
2718624S10943481W Ile de Paques.
4858698N00774154E Strasbourg.
9000000S90000000W Pole SUD.
Version du PILE - TAS = <
? : Ce MENUA : Ajouter un PIB : BIP
C : CorrigerD : Déplacer un
E : EffacerL : Lister les
P : PI sur RESETQ : Quitter
H : HiverK : Kmxxxxx
Num Nombre incorrectEchange avec ???
Il ne reste qu'un seulValeur incorrecte !
Commande incorrecteLongueur 0 ou > 1 !
EEPROM pleine !Commande
BIPsPI OriginexUI.NONFin du DIALOGUE
Distance/20NmKmLe []
LongitudeLatitudeNom de la cible
Item LA LG LiBlPI d'arrivéePI à
CorrigerPI à effacerAltitude Dh
dePide POSITIONGPS :Cible :DEBUT ?
ARRIVEE ?U batterie
= en Heure d'ETEHIVEREtat satellites.
poursuivis / NominaleExcellente
BonneNon fiablePoint non caleMode
differentielPoint estimeDilution H Marge

```

Cinq
emplacements
non utilisés et
disponibles

Erreur !

Encadré en rouge les accentués acceptés par le Moniteur de l'IDE.
Encadrés verts les accentués obtenus par PIXELs sur OLED.