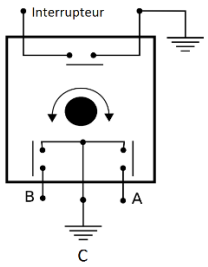
CommonBusEncoders

TUTORIEL

PRÉSENTATION

CommonBusEncoders (encodeurs reliés par des bus communs) est une librairie qui permet d'utiliser une grande quantité d'encodeurs, tout en maintenant le nombre de broches nécessaires sur l'Arduino à son minimum. Le premier encodeur requiert quatre (4) broches. Les autres encodeurs ne requièrent qu'une broche supplémentaire chacun.

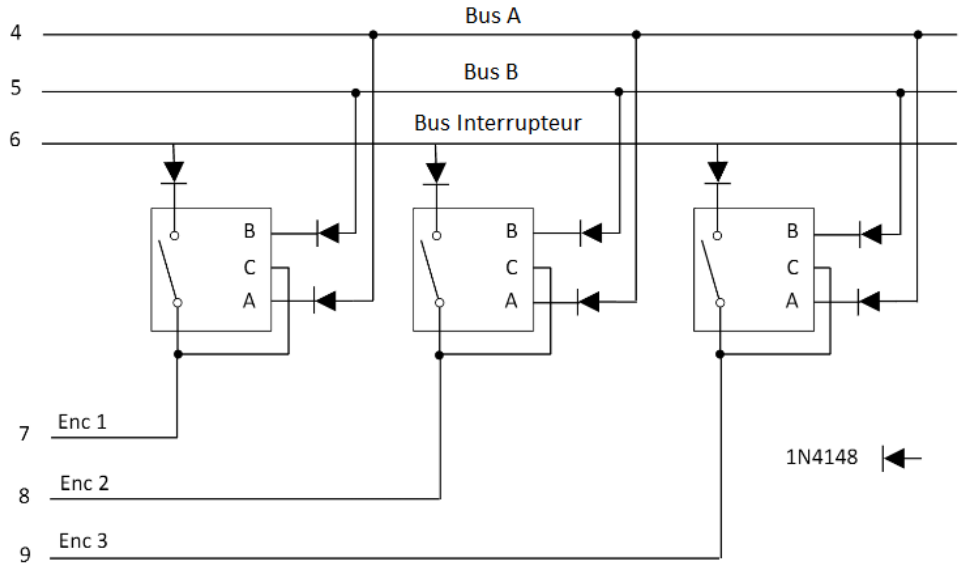
COMMENT RELIER LES ENCODEURS



Chaque encodeur possède 5 broches. Ces broches sont habituellement nommées A, B, C et deux broches servent à l'interrupteur si l'encodeur en possède un. [1]

Pour relier les encodeurs, toutes les broches **A** sont reliées ensemble, toutes les broches **B** sont reliées ensemble et toutes les broches **Interrupteur** sont reliées ensemble. Cet arrangement nous donne 3 bus qui seront reliés à 3 broches de l'Arduino.

De plus, les broches de retour à la terre de l'interrupteur et celui de l'encodeur (C) sont reliées entre elles et seront reliés à leur broche individuelle sur l'Arduino[2].



Afin d'isoler chaque encodeur, des diodes, dont les cathodes sont reliées à l'encodeur et les anodes sont reliées au bus sont nécessaires.

[1] Source de l'image : http://www.mon-club-elec.fr
[2] Merci à Vlad Sychev pour l'idée du montage : http://svglobe.com

LES TYPES D'ENCODEURS GÉRÉS

Les encodeurs qui peuvent être utilisés sont de type rotatif quadratique (quadrature encoders). Cette librairie ne gère que ceux qui sont conçus pour être tournés à la main. Pour les encodeurs qui sont attachés à des moteurs, il est préférable d'utiliser une librairie qui se concentre sur un seul encodeur et qui utilise des routines d'interruption de service. Plusieurs librairies sont offertes pour cet usage et elles sont très performantes.

Chaque révolution (360°) est marquée d'une certaine quantité de clics. De plus, certains encodeurs fournissent 2 ou 4 pas par clic (steps per detent). Cette librairie peut gérer les deux modèles. Si la plupart des encodeurs mentionnent le nombre de pas par clic, ce n'est pas toujours le cas. En cas de doute, utiliser le modèle à 4 pas par clic. Par la suite, si vous tournez l'encodeur et qu'il ne répond qu'une fois sur deux, c'est qu'il s'agit d'un modèle à 2 pas par clic.

Il est donc possible de reconnaître les rotations d'un demi ou d'un quart de clic. Cette librairie ne répond qu'aux moments où l'encodeur produit un clic, car c'est un moment où il est stable et que la main peut le relâcher sans risque qu'il change de position.

Cette librairie est donc conçue pour pouvoir modifier des valeurs discrètes (entiers, lettres de l'alphabet, angle de rotation en degrés, sélection d'un choix de menu…).

Il existe des encodeurs de type mécanique et de type optique. Les encodeurs optiques n'ont pas besoin de solution anti-rebonds (debouncing). Pour les encodeurs mécaniques, cette librairie offre une solution anti-rebonds logicielle[3]. Si le montage comprend à la fois des encodeurs mécaniques et des encodeurs optiques, il est fortement suggérer d'activer la détection des rebonds. Il est possible d'ajuster la sensibilité du détecteur de rebonds avec des valeurs allant de 1 (aucune détection) avec un temps de réponse d'environ 5µs, jusqu'à 32 avec un temps de réponse d'environ 160µs. Par défaut, la sensibilité est réglée à 16 (80µs).

FONCTIONNEMENT

Cette librairie permet d'utiliser plusieurs encodeurs. Afin de connaître quel encodeur a été utilisé et dans quelle direction il a été tourné, il est nécessaire de lui attribuer un code. Chaque encodeur aura un code qui lui est propre. De plus, afin de connaître la direction dans laquelle il a tourné, la convention suivante a été établie :

- Si l'encodeur a été tourné en direction horaire, c'est le code de l'encodeur qui est retourné;
- Si l'encodeur a été tourné en direction antihoraire, c'est le code de l'encodeur plus 1 qui est retourné.

Exemple : Si le code attribué à l'encodeur est 100 : Sens horaire = 100; Sens antihoraire = 101.

[3] Merci à : http://www.eng.utah.edu/~cs5780/debouncing.pdf. (en anglais)

Pour ce qui est de l'interrupteur, deux fonctions peuvent lui être attribuées :

- Servir à octroyer plusieurs modes à l'encodeur ;
- Servir comme un interrupteur indépendant de l'encodeur.

Dans le premier cas, il faut indiquer quel est le nombre de modes que devra gérer l'encodeur.

Supposons que l'on veuille ajuster la valeur d'un entier dont les valeurs admissibles se situent entre 0 et 999. En utilisant un encodeur avec un seul mode, l'encodeur devra avoir tourné pendant 1000 clics pour passer de 0 à 999, avec une moyenne de 500 clics pour ses valeurs possibles. Une autre avenue serait d'utiliser l'encodeur avec 3 modes : un pour changer les centaines, un pour changer les dizaines et un pour changer les unités.

L'interrupteur sert à changer de mode. Dans l'exemple, l'encodeur est tourné pour ajuster les centaines, clic sur l'interrupteur, les dizaines sont ajustées, nouveau clic sur l'interrupteur et ce sont les unités qui sont changées. Un nouveau clic sur l'interrupteur et ce sont à nouveau les centaines qui sont changées.

Pour un encodeur à 3 modes, les valeurs retournées seront les suivantes, en supposant que le code de l'encodeur est 100 :

- Centaines en sens horaire : 100;
- Centaines en sens antihoraire : 101;
- Dizaines en sens horaire : 102;
- Dizaines en sens antihoraire : 103;
- Unités en sens horaire : 104;
- Unités en sens antihoraire : 105.

Si l'interrupteur sert à un encodeur multimodes, lui attribuer le code 0. S'il est indépendant de l'encodeur (nombre de modes à 1), on peut lui attribuer un code qui lui est propre.

UTILISATION DANS UN SKETCH

La librairie doit d'abord être incluse dans le sketch :

```
#include <CommonBusEncoders.h>
```

Puis, les broches des 3 bus ainsi que le nombre d'encodeurs du circuit sont déclarés lors de la création de l'objet de la classe CommonBusEncoders. Ainsi, avec l'exemple présenté ci-dessus, la déclaration sera :

```
CommonBusEncoders encodeurs(4, 5, 6, 3);
```

Cette ligne est habituellement insérée tout de suite après les déclarations `#include` et doit apparaître avant la section `setup()` du sketch.

L'objet `encodeurs` qui vient d'être créé possède un tableau [1..3]. Pour chaque encodeur les informations suivantes doivent être fournies, dans l'ordre :

- Le numéro de l'encodeur (Dans quelle ligne du tableau mettre l'information);
- Le modèle d'encodeur : 2 pour le modèle à 2 pas par clic et 4 pour le modèle à 4 pas par clic;
- Le numéro de la broche de l'Arduino sur laquelle la broche commune (C) de l'encodeur a été reliée;
- Le nombre de modes de l'encodeur;
- Le code qui sera retourné par l'encodeur;
- Le code qui sera retourné par l'interrupteur de l'encodeur (Mettre à zéro si l'encodeur a plus d'un un mode).

Ces informations doivent être fournies dans la section `setup()` du sketch.

Exemple :

Le montage est réalisé grâce à trois encodeurs d'un modèle à 4 pas par clic est branché tel que montré à la première page de ce document.

- Le premier servira à ajuster un entier de 0..999 (3 modes);
- Le second servira à ajuster un entier de 0 à 9999 (4 modes);
- Le troisième servira à choisir une lettre de l'alphabet et son interrupteur servira à indiquer que le choix de la lettre est terminé.

Voici le code tel qu'il pourrait apparaître dans la section `setup()` du sketch :

```
Void setup() {
  encoders.addEncoder(1, 4, 7, 3, 100,   0);
  encoders.addEncoder(2, 4, 8, 4, 200,   0);
  encoders.addEncoder(3, 4, 9, 1, 300, 399);
}
```

Le montage est maintenant décrit à la librairie. Il ne reste qu'à lui demander de vérifier à chaque passage d'une boucle (dans la section `loop()` du sketch) si un encodeur a été utilisé et d'en faire état.

```
Void loop() {
  int code = encoders.readAll();
  if (code > 0) {
    //Agir sur ce code;
  }
  //Faire d'autres tâches.
  //REMARQUE : les encodeurs ne seront pas lus durant ces tâches
}
```

MÉTHODES SUPPLÉMENTAIRES

```
encoders.setDebounce(8);
```

Cette méthode permet d'ajuster la sensibilité du détecteur de rebonds. Le nombre fourni en argument peut prendre une valeur entre 1 et 32.

Un interrupteur est soit ouvert ou fermé. Cela est vrai la plupart du temps. Cependant, au moment de la transition, l'interrupteur verra sa valeur fluctuer rapidement d'ouvert à fermé jusqu'à ce qu'il atteigne un nouvel état stable. C'est ce qu'on appelle le rebondissement[4]. La durée pendant laquelle ces fluctuations perdurent varie d'un encodeur à un autre, mais aussi dépendent de la température ambiante, de l'humidité et de beaucoup d'autres facteurs. Cette librairie permet de détecter des fluctuations pouvant durer de 5µs, jusqu'à 160µs.

Selon la valeur de la sensibilité fournie en argument, l'algorithme effectue de 1 à 32 lectures sur la broche du Bus A de l'encodeur. Si toutes les lectures sont identiques, c'est que l'interrupteur est dans un état considéré stable et peut être comparé à la lecture précédente faite sur cette broche pour déterminer s'il a changé entretemps. Si une seule lecture n'est pas identique aux autres, c'est que l'interrupteur de la broche A est instable, signe qu'il est en transition entre le mode ouvert et le mode fermé ou vice-versa. Si c'est le cas, l'algorithme refera une autre série de lectures, et ceci jusqu'à obtenir une lecture stable.

Par défaut (si cette méthode n'est pas utilisée), la sensibilité est réglée à un niveau 16, et la lecture se fera pendant 80µs. Si la librairie retourne des valeurs erratiques, c'est ici qu'il faut agir. En supposant que l'encodeur est tourné manuellement et de manière raisonnable et que des valeurs erronées surviennent, augmenter sa sensibilité pourrait aider. S'il faut absolument tourner les encodeurs très rapidement et que des valeurs erronées surviennent, diminuer sa sensibilité serait une manière d'accélérer la cadence des lectures. La sensibilité détermine la vitesse des lectures et, inversement, la fiabilité des résultats.

```
encoders.resetChronoAfter(1000);
```

La plupart du temps, la librairie vérifie s'il y a une utilisation d'un des encodeurs en examinant l'état de tous les encodeurs. Toutefois, si une activité est détectée chez un encodeur particulier, la librairie lui accorde l'exclusivité et lui consacre toutes les lectures subséquentes, ignorant les autres encodeurs.

La prémisse est que lorsque la main est posée sur cet encodeur, elle ne le sera sur aucun autre.

Un chronomètre mesure le temps qui s'est écoulé entre un état de l'encodeur et le suivant. Ce chronomètre est remis à zéro à chaque changement d'état de l'encodeur. Lorsque le chronomètre atteint une certaine limite, la librairie considère que la main n'est plus posée sur l'encodeur et termine son exclusivité pour retourner vérifier l'ensemble des encodeurs du montage. Cette limite, en millièmes de secondes, est ajustée par cette méthode. Par défaut (si cette méthode n'est pas utilisée), elle est de 500 ms.

[4] Je n'ai trouvé aucune traduction officielle de « bouncing ».

```
if (!encoders.focussed()) { //Do something else; }
```

Si le sketch doit accomplir d'autres tâches indépendantes de l'état des encodeurs, il faut réaliser que pendant le temps requis pour accomplir ces tâches, il n'y aura aucune lecture des encodeurs. Il est préférable de limiter au maximum le temps nécessaire pour accomplir ces tâches pour éviter que les encodeurs soient utilisés sans réponse du montage.

Ceci est particulièrement vrai si l'encodeur est présentement utilisé. La librairie met en exclusivité un encodeur pendant son utilisation. Cette méthode permet de vérifier si un encodeur est en cours d'utilisation et de déférer à plus tard les autres tâches du sketch. Elle retourne VRAI si un encodeur est en cours d'usage. L'exemple démontre comment désactiver les autres tâches pendant la boucle en cours si un encodeur est actif.

UNE DERNIÈRE NOTE

Comme précisé précédemment, la librairie gère un tableau d'entiers contenant les caractéristiques de chaque encodeur. Étant donné que le nombre d'encodeurs peut varier d'un montage à un autre, la librairie ne réserve dans la partie de la mémoire dédiée aux variables que ce qui est nécessaire pour le nombre d'encodeurs utilisés.

Cependant, l'IDE de l'Arduino ne peut connaître la taille de ce tableau, puisqu'il n'est créé qu'après la compilation du sketch. Lorsque l'IDE de l'Arduino déclare après la compilation du sketch qu'une certaine quantité de mémoire est utilisée par les variables, la taille du tableau n'est pas incluse.

Chaque ligne du tableau nécessite 14 octets. De plus, 14 octets sont réservés par la librairie.

Exemple : L'IDE de l'Arduino vous rapporte que l'espace requis pour les variables est de 430 octets et votre montage compte 6 encodeurs, l'espace réel occupé dans la mémoire réservée aux variables est de : 430 + (6 x 14) + 14 = 528 octets.

J'espère sincèrement que cette librairie vous sera être utile dans vos projets.
Jacques Bellavance