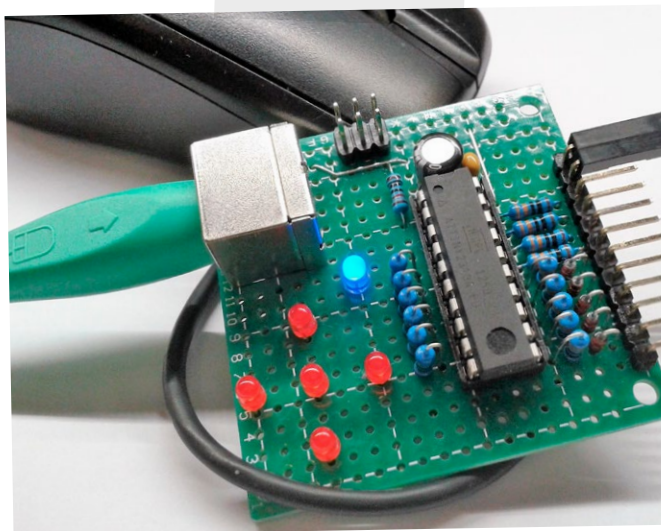


# 45 souris PS/2 comme codeur rotatif (et plus...)

Antonello Della Pia (Italie)

Reliez ce petit circuit au connecteur d'une souris PS/2 à molette, et vous avez en main l'équivalent d'un codeur rotatif à bouton. Le microcontrôleur ATtiny du circuit convertit les signaux du connecteur PS/2 et les distribue sur neuf sorties, dont deux servent à coder le sens de rotation de la molette.



L'idée de ce circuit m'est venue pendant que je travaillais sur un projet impliquant un codeur rotatif. J'utilisais une souris à molette, et je m'étais demandé si son mécanisme interne ne pourrait pas se substituer aux codeurs à signaux en quadrature habituellement utilisés avec les microcontrôleurs. Je ne voulais pas juste connecter une souris à un µC (la technique est connue), mais créer une sorte d'interface générique capable de remplacer un codeur rotatif et son bouton. J'ai récupéré une vieille souris PS/2, ai trouvé sur la toile une bibliothèque offrant également des fonctions pour la molette, et me suis mis au travail. Le résultat, obtenu avec l'aide d'un humble microcontrôleur ATtiny2313A de Microchip et de quelques composants courants, est une ancienne souris transformée en codeur rotatif à bouton. Il comprend deux boutons supplémentaires et quatre sorties « directionnelles » (haut, bas, gauche, droite), plus une optionnelle sortie LED signifiant « au repos » (sortie *IDLE\_LED*).

## Schéma

En dehors de l'alimentation, le circuit (**figure 1**) n'utilise que deux signaux du connecteur PS/2 : *Données* et *Horloge*. Les données reçues m'ont permis d'adresser neuf sorties, chacune ayant sa propre fonction et toutes référencées par rapport à la masse du circuit. Les sorties *EN\_A* et *EN\_B* fournissent le signal en quadrature représentant l'action des contacts mécaniques du codeur. La sortie *EN\_SWITCH* correspond au commutateur du codeur. *L\_CLICK* et *R\_CLICK* représentent les contacts des boutons gauche et droit de la souris. Les quatre autres

sorties (*LEFT*, *UP*, *RIGHT*, *DOWN*) sont au niveau Bas et passent à Haut lorsque la souris effectue les mouvements correspondants, se comportant ainsi comme un joystick.

Le niveau Haut de ces sorties est 5 V. Les diodes *D1-D5* permettent aux sorties associées d'apparaître comme des collecteurs ouverts, donc comme des contacts normalement ouverts. Ces cinq sorties peuvent aussi être reliées à des circuits à niveau logique Haut de 3,3 V. Les résistances de 330 Ω protègent les sorties contre les surintensités. La LED bleue s'allume lorsque la souris est au repos. Les cinq LED rouges indiquent les actions effectuées et ont pour résistances-talons *R2* à *R7*. *C1* et *C2* sont les habituels condensateurs de découplage. Le connecteur *JP1* autorise l'utilisation d'un programmeur *USBasp* pour flasher le µC. Le circuit s'alimente en 5 V mais, si la souris le permet, fonctionnera aussi sous 3,3 V.

## Logiciel

Le code source [1] est relativement simple. Il a été écrit et compilé avec l'EDI Arduino 1.8.19 et la version ATtinyCore 1.5.2 de Spence Konde. Plutôt que de recourir la fonction habituelle `digitalWrite()` d'Arduino, j'ai utilisé la directive `#define` pour créer des macros qui définissent l'état des broches. De même, au lieu des habituelles fonctions `pinMode()` et `digitalWrite()`, j'ai utilisé la manipulation directe des ports pour définir dans `setup()` les broches comme entrée/sortie ainsi que leur état initial. Le code compilé ne fait ainsi que 1830 octets, donc tient dans la petite mémoire flash de l'ATtiny2313A (2048 octets). Pour la

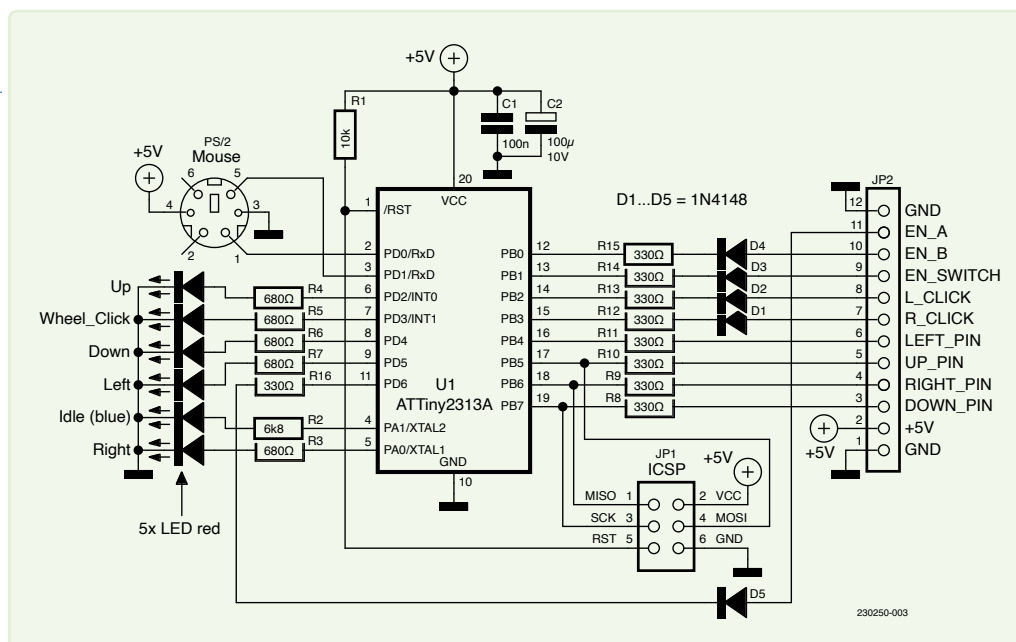


Figure 1. Schéma du circuit.

même raison, il est nécessaire d'activer LTO et de désactiver l'utilisation de `millis()/micros()` depuis le menu Outils de l'EDI Arduino. Dans la fonction `loop()`, les données traitées par la bibliothèque `arduino-ps2-mouse` sont évaluées par des expressions conditionnelles appelant trois fonctions. La première, `generateQuadratureSignal()`, produit le signal pour les sorties `EN_A` et `EN_B` lorsque la molette est actionnée (figure 2). J'ai obtenu le signal en quadrature en faisant basculer le niveau Haut/Bas des broches `PD6` et `PB0` à intervalles fixes et avec un déphasage de 90 degrés. C'est simple, mais efficace. La deuxième fonction, `mouseButtonHandling()`, achemine le signal des boutons vers leurs sorties respectives en utilisant une instruction `switch`. Enfin, `mouseShiftHandling()`, légèrement plus complexe, détecte les mouvements de la souris le long des axes principaux (X, Y, -X, -Y) et active les sorties appropriées. Ces trois fonctions traitent également les notifications d'événements par LED. S'il le faut, adaptez la réponse du programme à votre modèle de souris en modifiant les valeurs des fonctions `delay()` et des coordonnées définissant la position de la souris.

Les fichiers de bibliothèque nécessaires, `PS2Mouse.cpp` et `PS2Mouse.h`, sont inclus dans le dossier du programme. Il est donc inutile de les télécharger.

Le croquis `PS2_Mouse_As_Rotary_Encoder.ino` contient des commentaires et des informations supplémentaires. Il peut être facilement modifié et recompilé. Le fichier HEX est également disponible [1].

VF : Hervé Moreau — 230250-04

### À propos de l'auteur

Antonello Della Pia s'est senti attiré par les appareils électriques et électroniques dès l'enfance. Désormais adulte et technicien supérieur en génie électrique, il garde une passion pour l'électronique analogique et numérique. Ses intérêts du moment sont les microcontrôleurs et leur programmation, discipline pour laquelle il tente de s'améliorer. Il cherche à concevoir des projets aussi originaux que possible et, espère-t-il, intéressants.

### Des questions, des commentaires ?

Envoyez un courriel à l'auteur ([a.dellapia@elettronicaemake.it](mailto:a.dellapia@elettronicaemake.it)), ou contactez la rédaction d'Elektor ([redaction@elektor.fr](mailto:redaction@elektor.fr)).

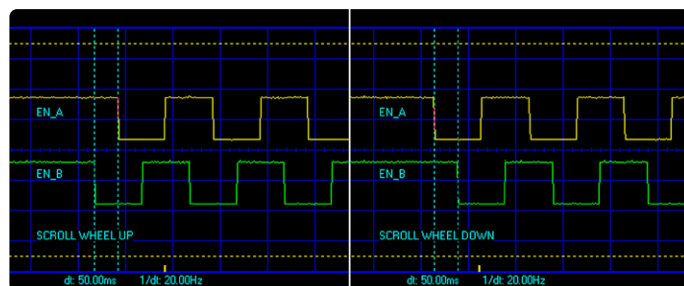


Figure 2. Le programme produit le signal pour les sorties `EN_A` et `EN_B` lorsque la molette tourne.

### LIEN

[1] Téléchargement du programme : <https://elektormagazine.fr/230250-04>



### Produit

> W. A. Smith, *Explore ATtiny Microcontrollers using C and Assembly Language*, Elektor, 2021  
<https://elektor.fr/20007>