05 la détection tactile en toute simplicité

un guide de fabrication pour n'importe quel microcontrôleur

Michael J. Bauer (Australie)

Tous les microcontrôleurs ne disposent pas d'entrées tactiles intégrées. Cet article présente une méthode pour activer la détection tactile capacitive en utilisant des broches d'E/S à usage général sur des dispositifs tels que l'AVR ATmega328P. Il couvre la configuration des broches pour détecter les changements de capacité et révèle le code développé avec Microchip Studio.

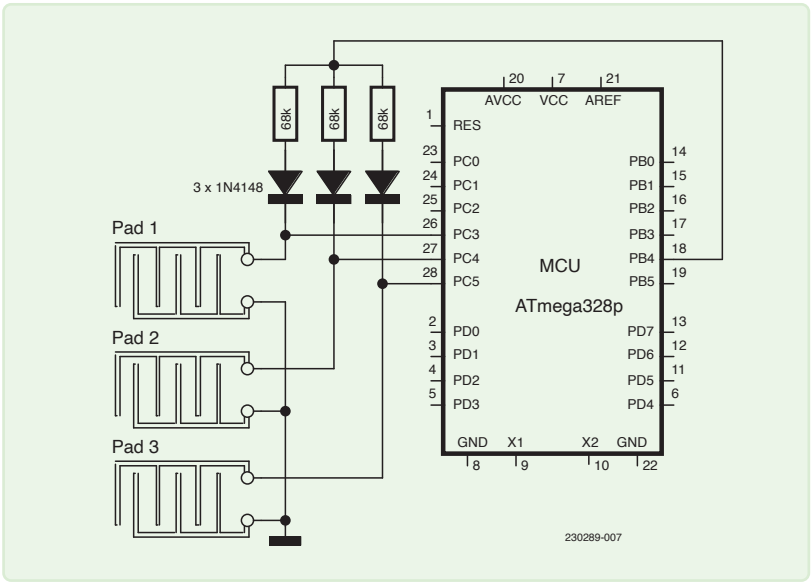


Figure 1. Schéma des connexions du pavé tactile au microcontrôleur.

Certains microcontrôleurs sont dotés d'un périphérique permettant de lire les contacts tactiles capacitifs. Ces dispositifs fournissent une mesure « automatique » de la capacité entre une broche d'E/S et la terre (GND). Cependant, un changement de capacité sur une broche d'entrée (analogique) peut être détecté facilement sans aucune intervention spéciale sur la puce. La technique présentée ici repose sur la mesure du taux de charge d'un condensateur (sous la forme d'une touche tactile). Pour être plus précis, on mesure la tension sur le condensateur après un temps de charge fixe. Lorsqu'une touche est touchée, sa capacité effective est plus élevée ; le taux de charge sera plus faible et donc la tension finale sera plus basse.

Principe

Le microcontrôleur doit disposer de broches d'E/S qui peuvent être configurées comme numériques (GPIO) ou analogiques (entrées CAN). Une telle broche est nécessaire pour chaque entrée tactile. Le nombre de touches tactiles peut être augmenté au-delà du nombre d'entrées ADC disponibles en utilisant un multiplexeur analogique externe. Une sortie numérique est nécessaire pour fournir une « source de courant ». En fait, il s'agit d'une source de tension, mais une résistance de grande valeur (une pour chaque entrée tactile) la transforme en source de courant. Se référer à la **figure 1** pour le schéma montrant comment les touches tactiles sont connectées au microcontrôleur.

Les touches tactiles peuvent être gravées sur un circuit imprimé avec des pistes de signal et de terre entrelacées pour former un condensateur. Lorsqu'elles sont touchées par un doigt, la capacité entre les traces de signal et de terre augmente. Les contacts tactiles peuvent également être de simples objets métalliques nus tels qu'une tête de vis, un rivet ou un morceau de ruban métallique. Le corps humain exerce une forte capacité à la terre lorsqu'il entre en contact avec une pastille tactile. Mais la détection tactile est plus efficace si le corps est connecté à la terre du système de microcontrôleur (GND) d'une manière ou d'une autre.

Initialement, la source de courant (PB4) est désactivée et les broches du pavé tactile (PC3, PC4, PC5) sont configurées comme des sorties et mises à l'état bas (0 V). Cela permet de décharger les contacs. Chaque contact est interrogé à tour de rôle, un par un. Au début de l'interrogation, la broche d'E/S de la pastille est configurée comme une entrée analogique. La pastille est chargée à partir

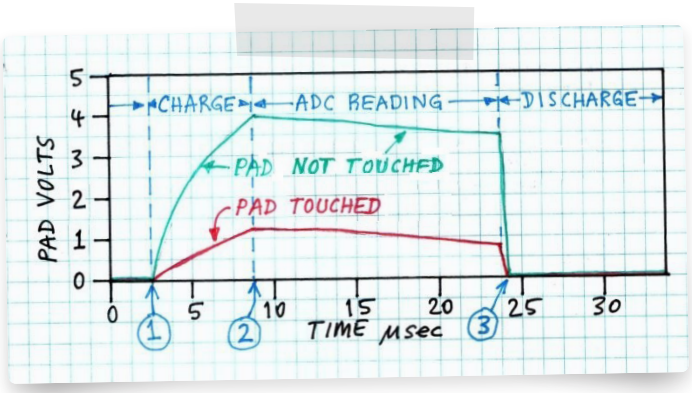


Figure 2. Tension en fonction du temps pour un contact tactile non touché.

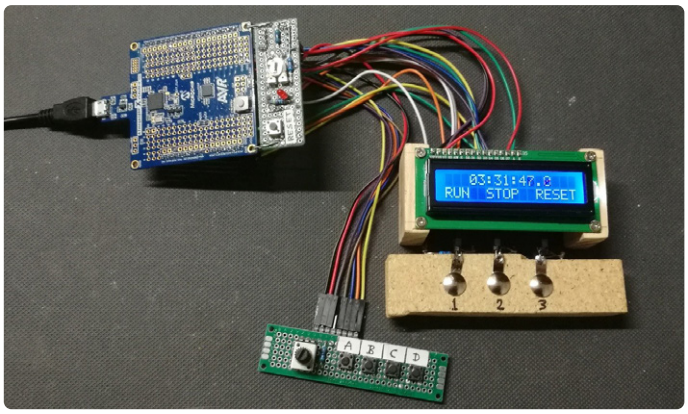


Figure 3. Configuration de Test & Demo avec la carte AVR X-Mini.

de la source de courant (résistance et diode) pendant une durée fixe, typiquement quelques microsecondes.

À la fin du temps de charge, la source de courant est désactivée. La diode empêche la décharge pendant que la tension de la pastille est mesurée par le CAN. Lorsque la conversion analogique-numérique est terminée, le résultat est lu et la broche d'E/S de la pastille est reconfigurée en sortie numérique et mise à l'état bas pour décharger la pastille.

Voir la **figure 2** pour le graphique illustrant la tension du signal en fonction du temps pendant la séquence de lecture. Le tracé vert correspond à la tension du contact, le contact n'étant pas touché. La capacité du contact étant très faible, elle se charge rapidement et atteint une tension relativement élevée. Au point de synchronisation 2, la source de courant est coupée et la séquence de conversion AN est lancée. Il y aura une décharge lente due à la fuite de courant dans la diode et dans les entrées du CAN. Ce n'est pas un problème car le CAN échantillonne la tension au début de la conversion (point 2) et la maintient constante pendant la conversion. À la fin de la séquence de conversion (point 3), la valeur du CAN est lue et sauvegardée dans un tableau. Enfin, le contact tactile est déchargé, en préparation d'une lecture ultérieure.

En fonction des exigences spécifiques de l'application, l'intervalle entre les lectures du pavé tactile peut être compris entre 100 µs et 5 ms, voire plus. Le temps d'exécution de la routine de service elle-même est généralement inférieur à 30 µs pour chaque entrée tactile. Ainsi, la charge de traitement pour l'entretien des contacts tactiles est très faible. Les interruptions doivent être désactivées pendant l'exécution de la routine de service car le temps est critique. Pour la plupart des applications embarquées, un retard de moins de 30 µs se produisant une fois par milliseconde (plus ou moins) ne serait pas considéré comme bloquant.

Application Test & Demo

La technique de détection tactile a été testée avec succès sur un microcontrôleur AVR 8 bits, l'Atmel ATmega328P, que l'on trouve dans de nombreuses cartes de développement populaires, notamment l'Arduino UNO R3 et Nano, et la carte ATmega328P (AVR) X-Mini de Microchip. J'ai choisi l'AVR X-Mini parce qu'il possède un outil de programmation embarqué, qu'il est très bon marché (~ 9€) et facilement disponible auprès des principaux fournisseurs, mais surtout parce que j'en avais un qui traînait dans ma boîte de composants inutilisés. La **figure 3** montre la carte X-Mini configurée et les périphériques connectés.

Aussi, je préfère de loin Microchip/Atmel Studio IDE pour AVR et SAM, comparé à l'Arduino IDE. Mon programme Test & Demo pourrait être migré vers Arduino IDE, mais les allocations de microcontrôleurs pin (pour un module 1602A LCD en particulier, voir ci-dessous) sont incompatibles avec les bibliothèques de code courantes d'Arduino. Il faudrait donc extraire le code source de certaines fonctions périphériques de la bibliothèque AVR X-Mini pour l'importer dans votre sketch Arduino.

Si vous souhaitez exécuter le programme Test & Demo sur une carte Arduino UNO ou Nano, le code objet préconstruit (fichier HEX) peut être programmé dans le microcontrôleur avec ou sans Microchip/Atmel Studio, et sans aucun outil de programmation supplémentaire (veuillez vous référer à l'encadré *Comment programmer un Arduino UNO…*.

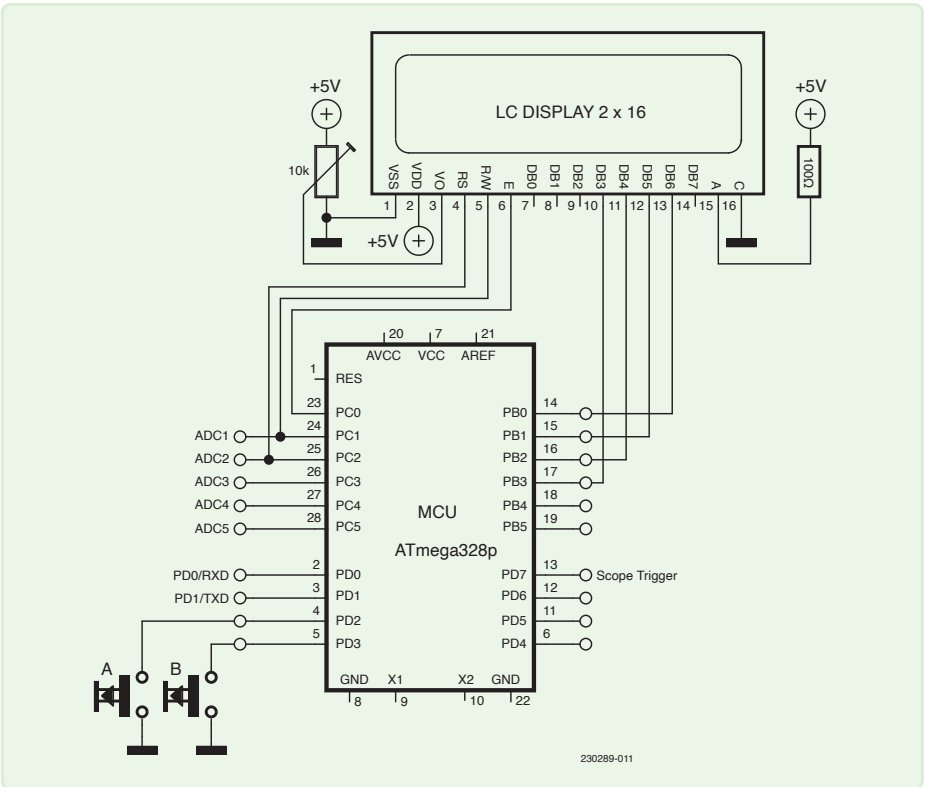


Figure 4. Schéma de l'application Test & Demo avec un ATmega328P.

Comment programmer l'Arduino UNO ou Nano

Le micrologiciel a été développé à l'aide de Microchip Studio for AVR and SAM Devices (anciennement Atmel Studio). L'une des nombreuses raisons pour lesquelles nous avons choisi cet IDE (Integrated Development Environment) au lieu d'Arduino est que la conception matérielle du projet est incompatible avec les bibliothèques de code Arduino disponibles. En particulier, le schéma d'interface de l'écran LCD 1602A (affectation des broches d'E/S du microcontrôleur) n'est pris en charge par aucune bibliothèque Arduino.

La programmation du dispositif cible (microcontrôleur ATmega328P) peut être réalisée sans l'EDI Arduino et sans aucun outil de programmation matériel. Les cartes UNO et Nano sont équipées d'un pont USB-série et d'un chargeur d'amorçage AVR résidant dans la mémoire flash. Une application PC Windows appelée *AVRdude* communique avec le bootloader via USB pour programmer le firmware dans la mémoire flash du microcontrôleur.

Vous devez donc télécharger certains fichiers pour faire fonctionner AVRdude sous Windows. Vous pouvez télécharger ces fichiers depuis le dépôt GitHub d'AVRdude [2]. Il devrait y avoir trois fichiers de distribution : *avrdude.exe*, *avrdude.conf* et *avrdude.pdb*. Copiez ces fichiers dans un nouveau dossier nommé *AVRdude* sur le disque local de votre PC, dans le répertoire principal (C:\).

Connectez votre carte UNO/Nano à un port USB de votre PC. Ouvrez l'utilitaire Windows Device Manager et cliquez sur *Ports* (*COM & LPT*). Vous devriez voir le périphérique USB-série UNO/Nano répertorié. Notez le numéro du port COM associé. Sachez que le numéro du port COM peut changer de temps à autre. Il s'agit d'une bizarrerie gênante de l'USB. N'oubliez pas de vérifier le port COM attribué lorsque vous reconnectez la carte UNO ou Nano à votre PC.

Vous pouvez exécuter AVRdude directement à partir de Microchip Studio, si vous créez un *Programming Tool* dans cet EDI :

Dans Microchip Studio, cliquez sur le menu *Tools/External tools*.

Vous devriez voir une boîte de dialogue vous demandant quelques paramètres, comme suit :

Dans *Title*, écrivez : *Program Nano* ou tout autre nom que vous préférez.

Dans *Command*, écrivez : *C:\AVRdude\AVRdude.exe*

Dans *Arguments*, écrivez (sur une seule ligne) :

```
-C "C:\AVRdude\avrdude.conf" -p atmega328p -c arduino -P COM# -b 115200
-U flash:w:"$(ProjectDir)Debug\$(TargetName).hex":i
```

Remplacez `COM#` (dans le champ Arguments) par le port COM réel auquel votre carte Nano est connectée, tel qu'il apparaît dans le gestionnaire de périphériques de Windows (exemple : `COM4`).

Cochez la case *Use output window*. Cliquez sur *OK*.

Terminé… Vous devriez voir une nouvelle option *Program Nano* dans le menu *Tools* (Outils). Une fois que votre code de programme est construit, il peut être programmé dans la carte Nano en cliquant simplement sur l'option *Program Nano* dans le menu *Tools*.

Remarque : certains clones chinois bon marché de la carte Nano utilisent un débit en bauds non standard pour le chargeur d'amorçage série, généralement 57600 bauds. Si Microchip Studio affiche un message d'erreur lors de l'exécution de l'outil de programmation, essayez de remplacer `115200` par `57600` dans le champ des arguments.

Quels que soient la plateforme matérielle et les outils de développement logiciel que vous utilisez pour votre application, la technique de détection tactile peut être comprise en examinant le code du programme Test & Demo, en particulier la routine de service de détection tactile. Si vous souhaitez reproduire l'application sur une plateforme matérielle AVR compatible, connectez l'écran LCD et les autres périphériques comme indiqué sur la **figure 4**. Deux interrupteurs (étiquetés « A » et « B ») sont prévus au cas où les contacts tactiles ne fonctionneraient pas initialement. Les contacts tactiles sont câblés comme indiqué dans la figure 1. Si votre application n'a pas besoin du LCD ou du bus I²C (PC4, PC5), alors toutes les (huit) broches du Port C peuvent être utilisées pour les contacts tactiles ou d'autres entrées analogiques.

Le seuil de tension d'activation/désactivation du toucher peut nécessiter un ajustement dans le logiciel. Si l'on se réfère au code source de Test & Demo (fichier *main.c*), la ligne 29 définit le niveau de seuil pour déterminer si un contact est touché ou non. La valeur par défaut est de 150, ce qui devrait convenir à la majorité des dispositions physiques des touches tactiles. La valeur seuil optimale peut être trouvée en exécutant le programme en *mode test*, sélectionné par le bouton A. Cela affichera les lectures CAN (plage 0…255) pour les trois touches tactiles. Pour chaque touche, notez la lecture lorsque la touche est touchée et de nouveau lorsque la touche n'est pas touchée. La valeur seuil optimale se situe à peu près à mi-chemin entre ces deux lectures.

Le programme comprend également un mode démo, qui fonctionne comme un chronomètre primitif, utilisant les trois pavés tactiles pour démarrer, arrêter et réinitialiser le chronomètre.

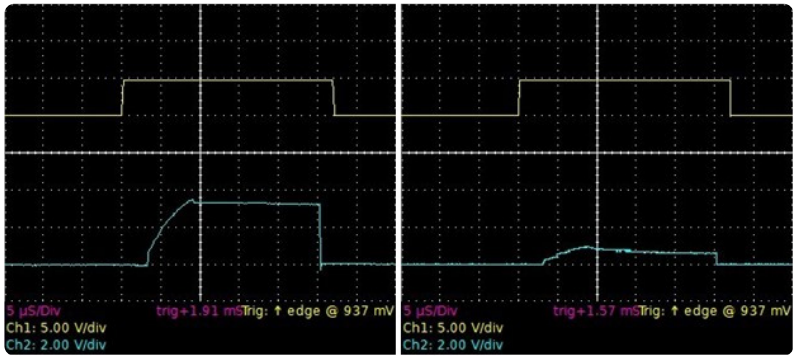


Figure 5. Résultat sur l'oscilloscope lorsque les contacts ne sont pas touchés (à gauche) et lorsqu'ils sont touchés (à droite).

Appuyez sur le bouton « B » pour accéder à ce mode. Le code de démonstration est implémenté sous la forme d'une machine à états avec trois états : *Réinitialisation*, *Marche* et *Arrêt*. Le fait d'appuyer sur un contact tactile change l'état (si la machine à états n'est pas déjà dans l'état sélectionné).

Captures d'écran de l'oscilloscope

La **figure 5** montre les captures d'écran d'oscilloscope obtenues pendant les tests, mettant en évidence le comportement du signal de la touche tactile. La trace supérieure (jaune) est un signal de sortie du point de test. Ce signal est mis à l'état haut au début de la routine de service et à l'état bas à la fin, juste avant le retour de la fonction. La sortie du point de test (broche PD7) a deux fonctions : d'une part, le déclenchement de l'oscilloscope et, d'autre part, la mesure de la durée d'exécution de la routine de service.

La trace inférieure (bleue) est le signal du contact tactile (entrée CAN). Dans ces tests, le temps de charge a été fixé à 6 µs. La valeur de la résistance source de courant (68 kΩ) et le temps de charge ont été choisis de manière à ce que le signal du contact atteigne presque (mais pas tout à fait) la tension de référence du CAN (+5 V) lorsque le contact n'est pas touché. Il en résulte une sensibilité tactile optimale.

L'écran de droite montre ce qui se passe lorsque l'on touche le contact. Le taux de charge étant beaucoup plus lent, la tension sur le contact à la fin du temps de charge est plus faible. Le logiciel définit une tension intermédiaire comme étant le seuil « Touch ON ». Notez que le temps de conversion du CAN est d'environ 15 µs. Cela représente la majeure partie du temps d'exécution de la routine de service. La fréquence d'horloge du CAN de l'ATmega328P peut être augmentée pour réduire le temps d'exécution de la routine de service. Il y a bien sûr un compromis entre la vitesse du CAN et la précision de la conversion. Cependant, la fréquence d'horloge du CAN utilisée dans le programme de test (2 MHz) donne une précision tout à fait acceptable, de sorte qu'elle pourrait peut-être être augmentée davantage.

Développement de micrologiciels

Si vous décidez de développer votre propre application à l'aide de l'IDE Microchip/Atmel Studio, en supposant que vous ne l'ayez jamais utilisé auparavant, un bon point de départ est mon programme Test & Demo. Une fois que l'application fonctionne sur votre plateforme cible, vous pouvez modifier et étendre le code source pour vos besoins.

Tout d'abord, téléchargez les fichiers du projet depuis GitHub [1]. Créez un dossier de projet sur le disque dur de votre ordinateur et copiez tous les fichiers dans ce dossier. Dans la page de démarrage d'Microchip Studio, sélectionnez *Open Existing Project* (Ouvrir un projet existant). Naviguez jusqu'à votre dossier de projet et cliquez sur le fichier nommé *X-mini-touch-sense.atsln*. Cette opération est plus facile que la création d'un nouveau projet et permet de s'assurer que le projet contient tous les composants requis, en particulier les fichiers de la bibliothèque de périphériques, *lib_avrXmini.**.

Si vous souhaitez créer un nouveau projet utilisant la bibliothèque X-Mini, veillez à linké correctement les fichiers de la bibliothèque. Ceci peut être vérifié dans le panneau *Solution Explorer* sur le côté droit de la fenêtre de l'éditeur IDE.

Pour des étapes de programmation détaillées et des configurations spécifiques, la page Elektor Labs [3] fournit des conseils utiles.

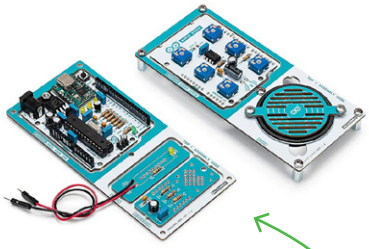
VF : Maxime Valens — 230289-04

Questions ou commentaires ?

Si vous avez des questions concernant cet article, n'hésitez pas à envoyer un courriel à l'auteur à l'adresse suivante mjbauer@iprimus.com.au, ou à l'équipe éditoriale d'Elektor à redaction@elektor.fr.

À propos de l'auteur

Michael Bauer a étudié l'ingénierie électrique et électronique et a ensuite enseigné l'informatique à l'université Deakin (Australie). Au cours de sa carrière, il a notamment co-développé ce qui était peut-être le premier « compteur intelligent » commercialement viable (compteur de kWh à base de microcontrôleur) ; il a conçu des systèmes de contrôle pour l'automatisation des scènes de théâtre ; il a conçu de l'électronique et développé des logiciels pour des instruments d'analyse scientifique et biomédicale. Aujourd'hui à la retraite, Mike s'intéresse à la technologie de la musique électronique, au cyclisme et au ski de fond. Sa principale fierté est un projet de montage d'ordinateur connu sous le nom de « DREAM-6800 » (publié en 1979), toujours populaire parmi les amateurs de micro-ordinateurs anciens.



Produits

- **Kit Arduino Make-Your-Uno**
  www.elektor.fr/20330
- **Seeed Studio Grove LCD RGB Backlight**
  www.elektor.fr/20014

LIENS

[1] M. J. Bauer | Touch sense on any MCU | Dépôt Github : https://github.com/M-J-Bauer/Touch-sense-on-any-MCU
[2] AVR Dude | Dépôt GitHub : https://github.com/mariusgreuel/avrdude/releases
[3] Page web d'Elektor Labs pour ce projet : https://elektormagazine.fr/labs/touch-sense-technique-for-any-mcu