générateur de formes d'ondes Arduino Nano

Nano + code = générateur de fonctions

Michael J. Bauer (Australie)

Ceci n'est pas un projet d'amateur finalisé avec circuit imprimé. Le plus intéressant dans ce projet, ce sont sans doute le micrologiciel et la bibliothèque de code pour les périphériques avec leurs nombreuses techniques utiles pour les applications embarquées en temps réel, développées par un ingénieur professionnel ayant des dizaines d'années d'expérience dans l'industrie.

Les générateurs de formes d'ondes, aussi appelés « générateurs de fonctions », sont des projets régulièrement publiés depuis les tout débuts d'Elektor. À base d'Arduino Nano, ce générateur de formes d'ondes se situe plutôt dans le bas de gamme en termes de complexité et de coût de construction. Néanmoins, un microcontrôleur AVR 8 bits lui confère une grande puissance, ce qui en fait un instrument utile pour les tests d'audio et d'équipements numériques à faible vitesse. Les composants sont logés dans un petit boîtier en plastique (130 × 70 × 40 mm), qui rend l'unité facilement transportable. L'alimentation est en 5 V via USB. Tous les éléments nécessaires à la construction de ce générateur sont disponibles à des prix dérisoires.

Mode *Wave*

Formes d'ondes : sinus, triangle, carré, dent de scie et bruit
Fréquence : basse 1…100 Hz (12 pas) ; haute 80 Hz…8 kHz (18 pas) ou 50 Hz…5 kHz en continu
Bruit : Filtré à 50, 100, 200, 400, 800, 1 600, 3 200 Hz, ou non filtré (blanc)
Couplage de sortie : CA ou CC
Niveau de sortie : 100 mVPP, 200 mVPP, 500 mVPP, 1 VPP, 2 VPP ou 4 VPP
Résolution du CN/A : 8 bits

Mode *Pulse*

Fréquence : basse 1…1000 Hz (16 pas) ; haute 1 kHz…4 MHz (16 pas)
Rapport cyclique : 1 ~ 99 % variable (256 pas)
Niveau de sortie : 3,3 V ou 5 V ; 20 mA source/collecteur (à 5 V)

Circuit

Le projet est basé sur la carte Arduino Nano v3, un module LCD de 2 lignes × 16 caractères (type 1602A) avec rétro-éclairage LED, quatre boutons-poussoirs et un potentiomètre. Le potentiomètre commande soit la fréquence du signal, soit le rapport cyclique, en fonction du mode de sortie sélectionné. Les connecteurs de sortie sont des prises phono RCA montées sur panneau.

Le circuit se sert d'une minuterie intégrée à l'ATmega328P pour générer des formes d'ondes à rapport cyclique variable sur une broche de sortie (voir l'encadré **Modes**). En mode *Wave*, on utilise une sortie MLI à 32 kHz pour générer des signaux sur la plage audio (jusqu'à 8 kHz) avec une sélection de formes d'ondes (sinus, triangle, carré et dent de scie). Le micrologiciel produit ces signaux avec un algorithme d'oscillateur à table d'ondes. Le CN/A à 8 bits permet une résolution en amplitude de 0,4 % de la pleine échelle.

La fréquence d'échantillonnage de 32 kHz est éliminée du signal de sortie audio à l'aide d'un filtre passe-bas analogique, de fréquence de coupure à 8 kHz, construit autour d'un ampli-op (½ MCP602). Ce filtre étant du troisième ordre, sa caractéristique est de -18 dB/octave. La **figure 1** contient des captures d'écran de plusieurs formes d'ondes de ce générateur.

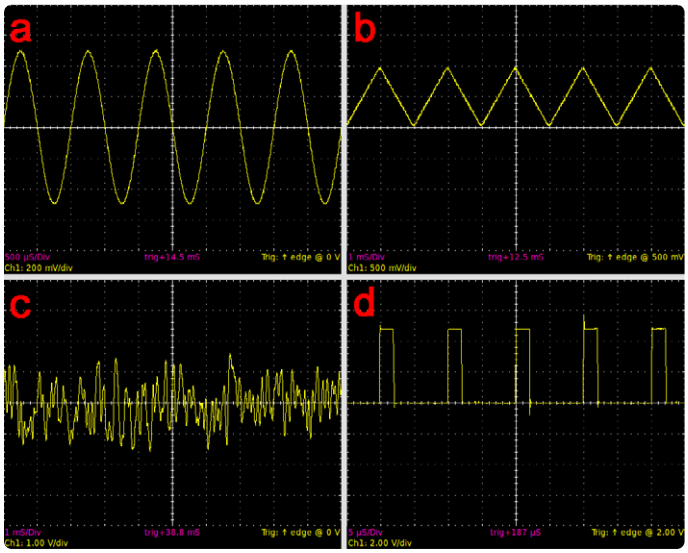

Figure 1. Captures d'écran des formes d'ondes : a) sinusoïdale 1 kHz, 1 V_{PP}, sortie CA ; b) triangulaire 500 Hz, 1 V_{PP}, sortie CC ; c) bruit, non filtré, 4 V_{PP}, sortie CA ; d) impulsion, 100 kHz, rapport cyclique = 20 %, sortie 5 V.

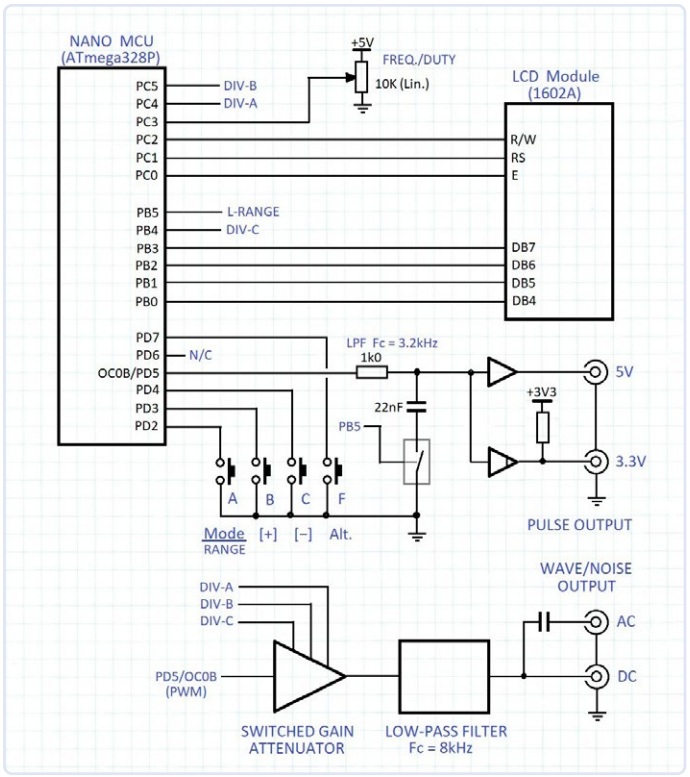

Figure 2. Circuit simplifié du générateur de fonctions basé sur un Arduino Nano.

Circuit simplifié

La **figure 2** présente un schéma fonctionnel simplifié du générateur. En mode *Wave*, la sortie est réglée sur l'un des six niveaux à l'aide d'un atténuateur à gain commuté interposé entre la broche de sortie MLI et le filtre passe-bas à base d'ampli-ops. L'atténuateur utilise trois interrupteurs analogiques (74HC4066) commandés par le microcontrôleur, suivis d'un ampli-op tampon (½ MCP602).

En mode *Low-frequency pulse*, le générateur utilise la même fréquence d'échantillonnage de 32 kHz qu'en mode *Wave* pour générer des impulsions avec des rapports cycliques variables. Un simple filtre RC du premier ordre avec une fréquence de coupure de 3,2 kHz élimine la plupart des résidus du signal de 32 kHz. Un tampon (74HC125) convertit le signal analogique en signal numérique, en éliminant tout le reste de 32 kHz. Le signal a maintenant des fronts de montée et de descente rapides et peut piloter directement des circuits numériques.

En mode *High-frequency pulse*, le module de minuterie du MCU génère directement un signal dont la fréquence et le rapport cyclique sont variables. Le filtre RC est déconnecté en ouvrant un interrupteur (¼ 74HC4066). Le signal de sortie est tamponné à l'aide d'un couple de portes câblées en parallèle (quadruple tampon à trois états 74HC125). Ceci permet à la sortie d'impulsion de 5 V de piloter un courant « élevé » (collecteur ou source de 20 mA). Les deux portes restantes fournissent un niveau de sortie en 3,3 V grâce à une résistance de rappel de 1 kΩ. La solution du rappel conduit à des fronts de montée non optimaux, bien visibles à fréquence élevée. La **figure 3** montre le schéma détaillé du circuit.

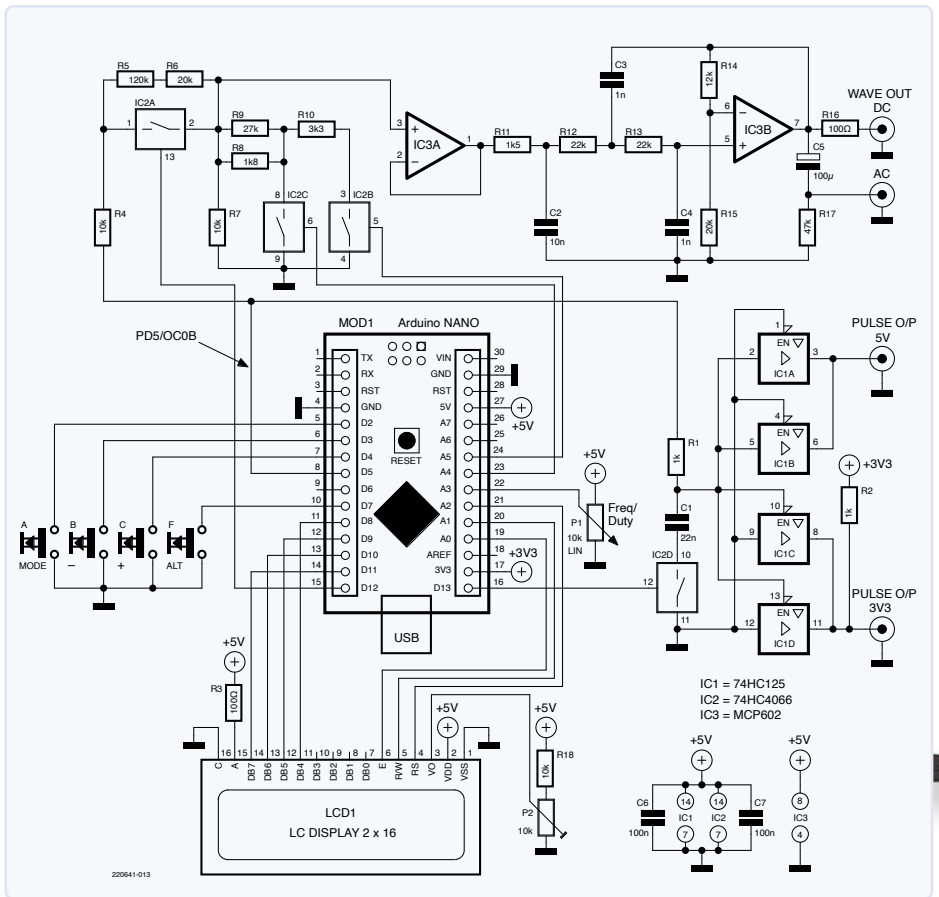

Figure 3. Le circuit détaillé n'est pas beaucoup plus compliqué que la version simplifiée de la figure 2.

Panneau de commande

La forme d'onde de sortie souhaitée est sélectionnée en faisant défiler les options disponibles à l'aide du bouton *MODE* (voir **figure 4**). En modes sinus, triangle, dent de scie et carré, le potentiomètre *FREQ/DUTY* permet de régler la fréquence de sortie (FO). En mode Noise, le potentiomètre permet de régler la fréquence de coupure du filtre de bruit. En modes *Sinus*, *Triangle*, *Sawtooth*, *Square* et *Pulse*, les touches *FREQ+* et *FREQ-* permettent de faire défiler vers le haut ou vers le bas une gamme de fréquences préréglées. Cette méthode de sélection de la fréquence du signal permet d'obtenir un réglage plus précis qu'avec un potentiomètre. En outre, les valeurs de fréquence fixes ont été choisies afin que le signal de sortie ne présente pas d'artefacts de repliement.

Figure 4. Face avant du prototype de l'auteur avec écran, boutons et potentiomètre.

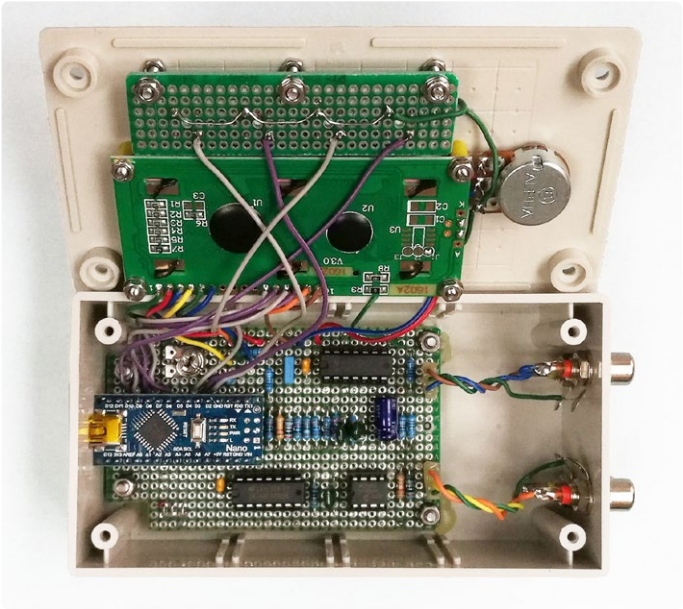

Figure 5. L'ensemble des composants électroniques montés dans un boîtier en plastique. La carte Arduino est calée sur un côté pour que le port USB soit accessible de l'extérieur.

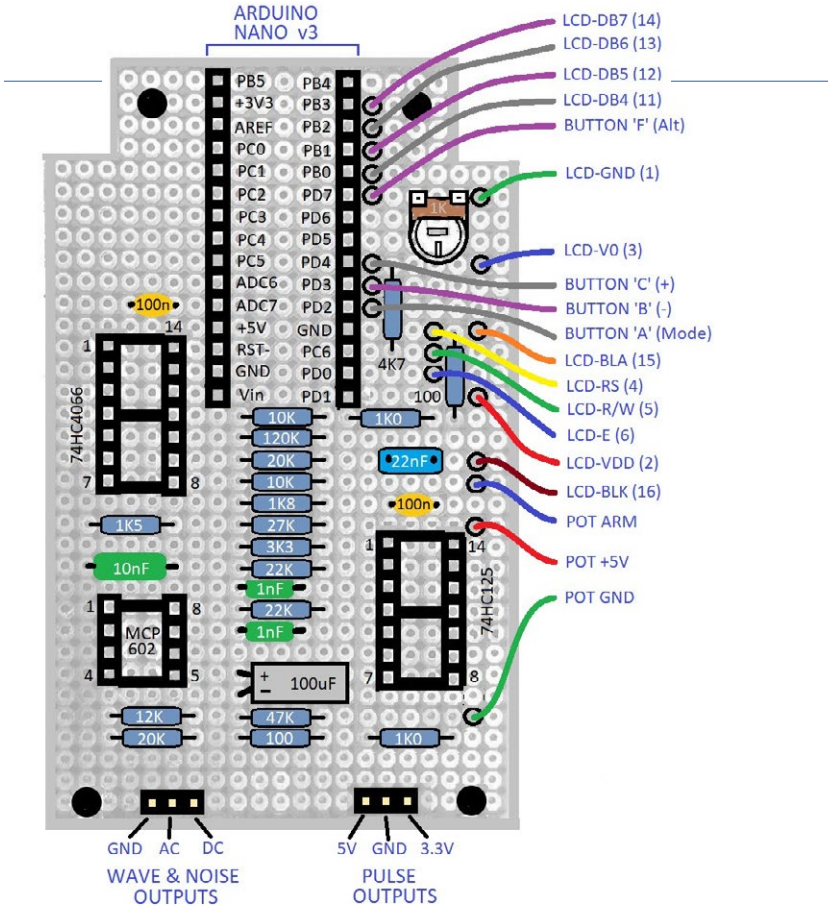

Figure 6. Face supérieure de la carte de prototypage avec la position exacte des composants et des supports.

En mode *Pulse*, la largeur d'impulsion est réglée à l'aide du potentiomètre. La valeur n'est pas ajustable en continu mais « arrondie » pour donner un réglage précis. La résolution dépend de la fréquence. Les valeurs de largeur d'impulsion sont espacées à haute fréquence et plus rapprochées à basse fréquence. La largeur d'impulsion est affichée en pourcentage de la période.

Si la touche *ALT* est maintenue enfoncée, elle active d'autres fonctions pour les trois autres touches. Pour sélectionner la gamme de fréquences, maintenez la touche *ALT* enfoncée et appuyez sur la touche *RANGE* pour basculer entre les gammes haute et basse. Pour régler le niveau du signal de sortie dans tous les modes, sauf *Pulse*, maintenez la touche *ALT* enfoncée et appuyez sur la touche *LEVEL+* ou *LEVEL-* pour faire défiler les niveaux disponibles.

Construction

Comme déjà mentionné, aucun circuit imprimé n'a été conçu pour ce générateur. Au lieu de cela, l'électronique est construite sur une carte de prototypage avec des plots de soudure isolés (grille de 2,54 mm) comme le montre la **figure 5**. L'utilisation d'une carte double face à trous métallisés est parfaite pour cela. J'ai utilisé un morceau de 55 × 90 mm (avec 21 × 34 plots). Si le boîtier a des poteaux dans chaque coin pour les vis de fixation du couvercle, découpez deux coins de la carte pour que le côté court soit au ras de l'une des extrémités du boîtier. La prise USB de l'Arduino Nano est située sur ce côté.

La carte est fixée au fond du boîtier sur quatre entretoises de 6 mm. Mieux vaut les coller à l'intérieur du boîtier afin qu'elles restent en place lorsqu'on met les vis. Quatre boutons poussoirs sont soudés sur un autre morceau de carte de prototypage, également placé à l'intérieur du couvercle supérieur, sous le module LCD. Avant de souder les boutons, percez des trous de 2 mm dans la carte à l'emplacement central de chaque bouton.

La **figure 6** montre comment les composants sont soudés sur la partie supérieure de la carte. Le câblage sur la face inférieure de la carte est un peu plus compliqué : la **figure 7** montre le câblage de base à l'aide d'un fil de cuivre étamé dénudé tout autour de la carte, en suivant les pastilles les plus à l'extérieur et en évitant les trous de fixation. Il s'agit du rail GND. Reliez toutes les connexions à GND avec du fil de cuivre étamé dénudé, comme indiqué sur le schéma. Il vaut mieux utiliser un fil de section similaire à celle des pattes de résistances (~24 AWG,

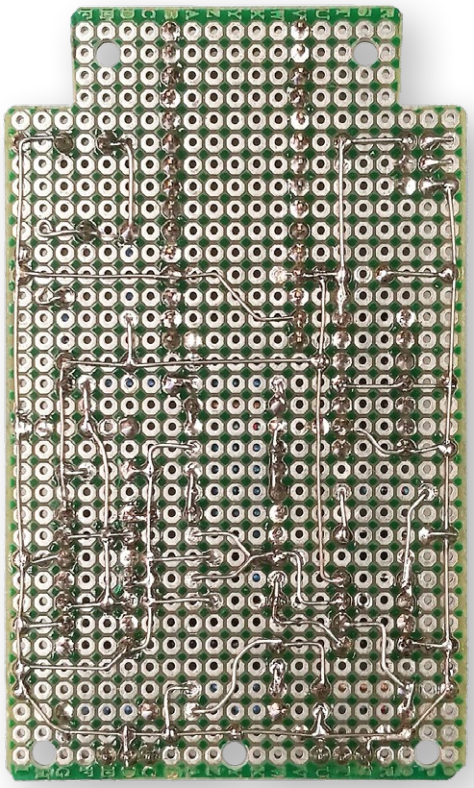

Figure 7. Face inférieure de la carte de prototypage avec câblage de base en fil de cuivre étamé.

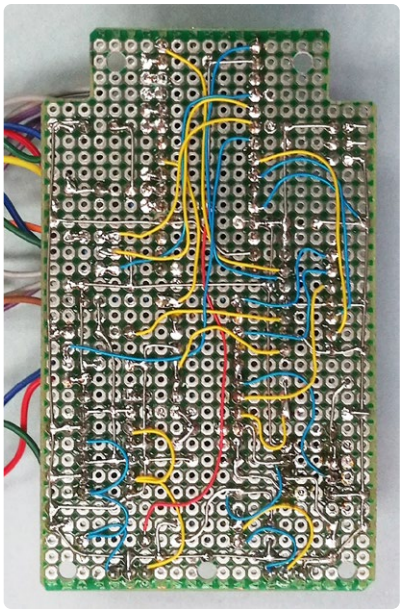

Figure 8: Face inférieure de la carte de prototypage avec des fils toronnés supplémentaires et des fils de connexion.

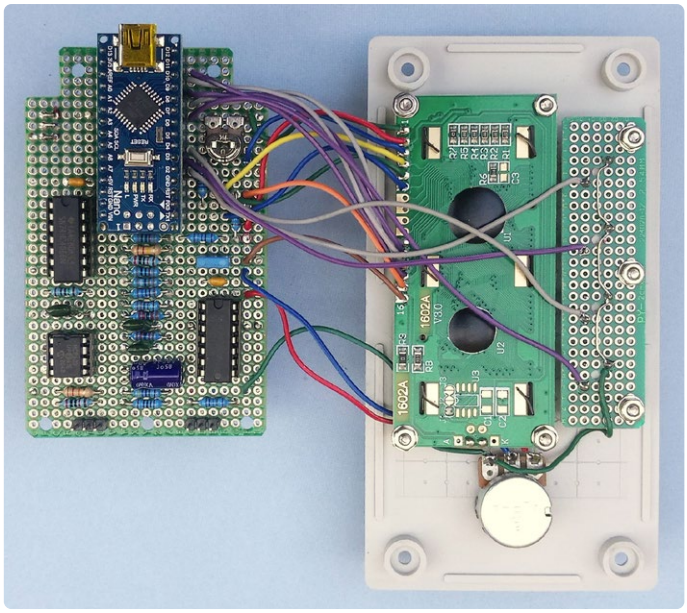

Figure 9. Prototype complet avec Arduino Nano connecté à l'écran, aux boutons et au potentiomètre, prêt pour un premier test.

0,5 mm). On peut aussi réaliser la plupart des connexions à VCC (+5 V) avec du fil dénudé. Le reste des connexions peut se faire avec de courts morceaux de fil isolé, comme le montre la **figure 8**. Des couleurs différentes peuvent faciliter le repérage.
Après avoir vérifié la conformité de la carte au schéma, connectez le module LCD, les boutons-poussoirs et le potentiomètre de commande. La méthode la plus simple et la plus fiable consiste à souder des fils toronnés directement sur la carte. Placez ensuite le module Arduino Nano et les circuits intégrés dans leurs supports. Réglez à environ mi-échelle le potentiomètre de réglage du contraste de l'écran LCD. Installez le circuit imprimé dans son boîtier comme le montre la **figure 9**. Voici venu le moment du « test de fumée ». Connectez l'Arduino Nano à une source d'alimentation de 5 V CC via le port USB. Le rétroéclairage de l'écran LCD doit être allumé. Si les tensions CC sont correctes et que vous ne voyez pas de fumée, votre carte est prête à être testée avec le micrologiciel.

Installation du micrologiciel

Le micrologiciel a été développé à l'aide de Microchip Studio for AVR and SAM Devices (anciennement Atmel Studio). La raison principale du choix de cet EDI par rapport à Arduino est que la conception matérielle du générateur de formes d'ondes est incompatible avec les bibliothèques de code Arduino disponibles. En particulier, la disposition d'interface du LCD 1602A (affectation des broches d'E/S avec le MCU) ne semble pas être prise en charge par une bibliothèque Arduino. On peut programmer l'ATmega328P cible sans Microchip Studio ni aucun outil de programmation matériel. Une méthode alternative est décrite dans l'encadré **Votre propre micrologiciel avec Microchip Studio**. L'Arduino Nano dispose d'un pont USB-série intégré et d'un chargeur de démarrage AVR résident. L'application Windows AVRDUDE [1] communique avec le chargeur de démarrage via USB pour programmer le micrologiciel dans la mémoire flash du MCU. Après avoir connecté votre Arduino Nano à un PC, ouvrez l'utilitaire *Windows Device Manager* et cliquez sur Ports (COM et LPT). Le périphérique série USB du Nano doit apparaître dans la liste. Notez le numéro du port COM associé et utilisez ce numéro lors de l'édition de la ligne de commande AVRDUDE. La programmation s'effectue à l'aide de l'invite de commande Windows. Une seule ligne de commande suffit. En revanche, la ligne de commande AVRDUDE requise est assez longue et doit être adaptée à la configuration logicielle particulière de votre PC et à l'allocation du port COM.
Copiez et collez (ou tapez) la ligne de commande ci-dessous dans un éditeur de texte tel que Bloc-notes. Veillez à supprimer tout saut de ligne afin que la commande soit sur une seule ligne. Le fichier *Program Nano.bat* contenant ces commandes se trouve dans le fichier d'archive, qu'on peut télécharger gratuitement sur le site Elektor de cet article [2]. Sélectionnez *Retour automatique à la ligne* dans le menu *Format* de *Bloc-notes* pour le rendre plus lisible. Voici la ligne de commande :

```
avrdude.exe -C avrdude.conf" -p atmega328p -c arduino
-P COM4 -b 115200
-U flash:w:nano-wave-gen-v1.5.hex:i
```

Certains des champs en gras devront être modifiés en fonction de la configuration de votre PC et de la révision du micrologiciel à programmer : remplacez COM4 par le port COM réel auquel votre Arduino Nano est connecté, tel que vous l'avez trouvé en utilisant le Gestionnaire de périphériques de Windows. Force est de constater que le port COM attribué peut changer lorsque le câble USB est débranché et rebranché. Ceci parce que la connexion USB-série est réalisée via un port série « virtuel » - et non un port « physique ». Le nom du fichier du micrologiciel de [2] sera quelque chose comme *nano-wave-gen-v1.5.hex*. Modifiez la ligne de commande AVRDUDE pour que le nom du fichier hex (en surbrillance) soit le même que celui qui a été téléchargé. Enregistrez le fichier de commande modifié sous *Program Nano.bat*.
Nota : Certains clones de cartes Arduino Nano bon marché utilisent un débit en bauds non standard pour le chargeur de démarrage série, qui peut être de 57 600 bauds. Si AVRDUDE affiche un message d'erreur, essayez d'utiliser une vitesse de transmission différente. Dans ce cas, remplacez « 115200 » par « 57600 » dans la ligne de commande.

Principe de fonctionnement

La technique employée est appelée « synthèse par table d'ondes ». Une table d'ondes est un tableau de données représentant un cycle complet (1 période) d'une forme d'onde. Les éléments du tableau contiennent les amplitudes de chaque échantillon du signal. La MCU émet les échantillons à une fréquence fixe appelée « fréquence d'échantillonnage ». Cette fréquence d'échantillonnage doit être (beaucoup)

plus élevée que la composante de fréquence maximale du signal de sortie. En règle générale, il convient d'utiliser un facteur de quatre. Pour générer des signaux audios jusqu'à 10 kHz, par exemple, il faut une fréquence d'échantillonnage de plus de 40 kHz. Par commodité, comme la fréquence d'horloge du MCU est de 16 MHz, on a choisi une fréquence d'échantillonnage plus faible à 32 kHz.
Le microprogramme utilise le module *Timer/Counter TC0* de l'ATmega328P en mode MLI pour générer des formes d'ondes arbitraires dans la gamme des fréquences audibles et des infrasons. La table d'ondes contient le rapport cyclique du signal de sortie MLI. Ce signal MLI encore numérique passe par un filtre passe-bas qui le moyenne à un niveau de tension analogique conforme à la forme d'onde stockée dans la table d'ondes, à un décalage de phase près.
La fréquence de coupure du filtre à 8 kHz élimine au mieux la fréquence d'impulsion de 32 kHz, ne laissant présentes dans le signal de sortie que les composantes de fréquence audio souhaitées. La sortie du signal MLI combinée au filtre passe-bas fonctionne comme un convertisseur numérique-analogique, ce qui permet d'économiser le coût d'une puce CN/A séparée. La **figure 10** montre le principe de fonctionnement en affichant le signal MLI à partir d'une table d'ondes contenant des données sinusoïdales dans la partie supérieure de l'écran et la forme d'onde résultante après le filtre passe-bas dans la partie inférieure de l'écran.

Fonctions de temporisation dans le micrologiciel

Dans le fichier d'archive du micrologiciel à [2], on trouve non seulement le fichier hex prêt à programmer, mais aussi le code source complet. Si la façon dont ce logiciel transforme un Arduino Nano en générateur de formes d'ondes vous intéresse, jetons un coup d'œil au code. La fonction `TC0_setup()` initialise le *Timer-Counter* TC0 en mode MLI « *dual slope* » (c.à.d correction de phase) pour générer une forme d'onde d'impulsion avec un rapport cyclique variable sur la broche OC0B (= PD5). On se sert du registre de comparaison de sortie A du minuteur TC0 pour générer une interruption périodique à la fréquence d'échantillonnage de 32 kHz, de sorte que l'intervalle entre deux IRQ sera de 31,25 µs. La routine de service d'interruption (ISR) doit récupérer, traiter et émettre un échantillon toutes les 31,25 microsecondes. C'est un véritable défi pour un MCU 8 bits bas de gamme, mais avec une fréquence d'horloge de 16 MHz, jusqu'à 16 instructions peuvent être exécutées en 1 µs. Par conséquent, jusqu'à 16 × 31,25 = 500 instructions peuvent être exécutées dans l'ISR. Il y a assez de puissance de calcul pour ne pas avoir à utiliser l'assembleur.
Bien sûr, le MCU ne peut pas passer tout son temps dans l'ISR, sinon rien d'autre ne serait fait dans la boucle principale (en arrière-plan). Il faut donc s'assurer que le temps d'exécution maximum de l'ISR soit très inférieur à l'intervalle des IRQ. Un bon choix consiste à utiliser la moitié de l'intervalle des IRQ comme temps d'exécution de l'ISR.
Le diviseur d'horloge est désactivé (N = 1) pour obtenir la fréquence d'horloge la plus élevée. Avec une fréquence d'horloge de 16 MHz, le registre OCR0A est réglé sur 249 pour une période de 250 tops (dans chaque sens – haut et bas), de sorte que la fréquence de sortie MLI sera exactement de 32 kHz. On utilise le registre de comparaison de sortie B, OCR0B, pour générer une sortie MLI sur la broche PD5/OC0B. Le rapport cyclique du MLI varie proportionnellement à l'amplitude de l'échantillon, mis à jour à chaque échantillon (toutes les 31,25 µs) par l'ISR du registre de comparaison de sortie A du minuteur.
Lorsque le registre du compteur (TMR0) atteint son « *TOP count* » (OCR0A = 250), le sens du comptage est inversé et un drapeau IRQ (OCA) est levé. En même temps, la broche de sortie MLI, PD5/OC0B, est mise à l'état haut. Lorsque TMR0 atteint la valeur OCR0B, la broche PD5/OC0B est automatiquement mise à l'état bas. Le rapport cyclique de l'impulsion MLI est donc proportionnel à la valeur OCR0B. Le rapport cyclique ne peut évidemment pas dépasser la période, et donc le rapport cyclique maximal (100 %) est obtenu en écrivant 249 dans OCR0B. Pour plus de détails sur la génération de forme d'onde MLI à l'aide d'un microcontrôleur AVR, référez-vous à la fiche technique de l'ATmega328P [3], dans la section 8-bit *Timer/Counter TC0*.

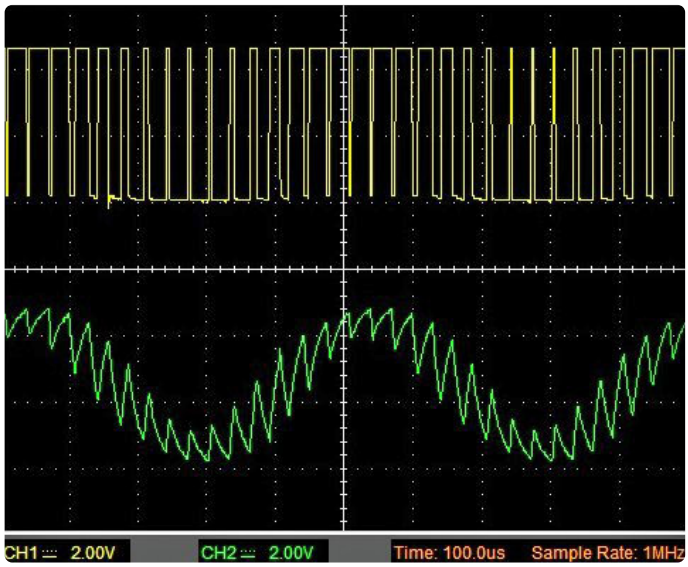

Figure 10. L'écran supérieur montre le signal MLI d'une onde sinusoïdale de 2 kHz. La forme d'onde sur l'écran inférieur est le résultat après filtrage passe-bas.

Algorithme de table d'ondes

L'algorithme de table d'ondes fonctionne en récupérant des échantillons d'une table d'ondes à la fréquence d'échantillonnage fixe de 32 kHz. La fréquence du signal de sortie est déterminée par la « distance » (angle de phase) entre les points extraits de la table. Plus la distance est courte, plus la fréquence de sortie est basse. Lorsque l'index dans la table de l'échantillon suivant à extraire dépasse la fin de la table, l'index est ajusté pour revenir sur un point de la table en maintenant la bonne distance d'échantillonnage.
Appelons « pas d'angle de phase » la distance entre les points d'échantillonnage. Pour des raisons pratiques, le pas d'angle de phase doit être un nombre réel, c.à.d. qu'il doit comporter à la fois une partie entière et une partie fractionnaire. L'arithmétique à virgule flottante faciliterait la tâche, mais elle est trop lente pour l'ATmega328P qui n'a pas d'unité à virgule flottante intégrée. C'est pourquoi on utilise l'arithmétique à virgule fixe à la place.
L'arithmétique à virgule fixe utilise l'arithmétique entière rapide avec des mots entiers (ou mots longs) composés de deux « champs de bits » représentant les parties « entières » et « fractionnaires » d'un nombre. Un nombre à virgule fixe de 16 bits peut être composé d'une partie entière de 8 bits et d'une partie fractionnaire de 8 bits. Pour les nombres non signés, la plage est comprise entre 0,0 et 255,99

Votre propre micrologiciel avec Microchip Studio

Pour adapter le micrologiciel du générateur de formes d'ondes à vos propres besoins, téléchargez et installez d'abord **Microchip Studio** for AVR and SAM Devices (EDI) sur votre PC Windows ou Mac [4]. Le tutoriel mentionné à [4] présuppose une « plate-forme cible » utilisant une carte de développement Microchip ou Atmel avec un outil de programmation matériel, par ex. Atmel AVRISP mk2 ou un dispositif similaire. Mais il n'est pas nécessaire d'avoir un de ces outils pour charger le micrologiciel dans le MCU Arduino Nano. Vous pouvez créer un *« Programming Tool »* dans **Microchip Studio** (voir « Option 2 » plus bas).
Après avoir téléchargé l'archive du micrologiciel à partir de [2] et l'avoir extraite, copiez tous les fichiers dans un nouveau dossier nommé *Nano-waveform-generator* sur un disque local, de préférence dans le dossier projects créé par **Microchip Studio** dans votre répertoire Documents. Connectez maintenant votre Arduino Nano à un port USB de votre PC. Ouvrez ensuite **Microchip Studio** et choisissez Open Project… à partir de la page d'accueil. Naviguez jusqu'au dossier de projet que vous avez créé et cliquez sur le nom de fichier *Nano-waveform-generator.atsln*.
Sur le côté droit de la fenêtre de l'EDI, trouver et cliquer sur l'onglet **Solution Explorer** (voir **figure 11**). Développez la liste Libraries et assurez-vous que la bibliothèque *lib_avrXmini.a* est présente. Le fichier de la bibliothèque contient des fonctions préconstruites pour prendre en charge divers périphériques couramment utilisés dans les applications du MCU ATmega328P.

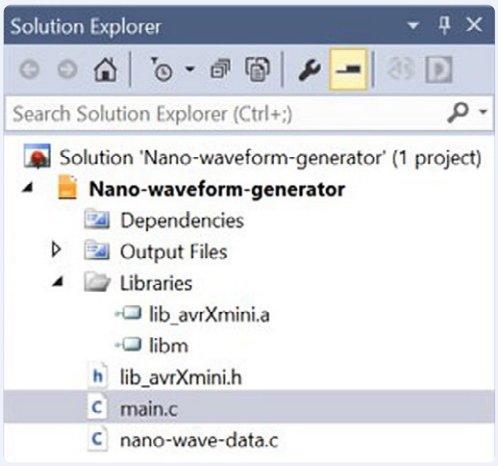

Figure 11. Capture d'écran du panneau de ***Solution Explorer****.*

Pour modifier un fichier source s'il n'est pas déjà ouvert dans l'éditeur, cliquez sur le nom du fichier dans le panneau de **Solution Explorer**. Pour garder le fichier ouvert dans l'éditeur, cliquez sur l'icône en épingle dans l'onglet de fichier en haut à droite de la fenêtre de l'éditeur. L'onglet de fichier se déplacera vers le côté gauche de la fenêtre et restera ouvert. N'hésitez pas à consulter le fichier d'en-tête de la bibliothèque, *lib_avrXmini.h*, mais ne le modifiez pas, sauf si vous avez l'intention d'apporter des modifications à la bibliothèque. Dans le code source C, on peut librement modifier n'importe quelle fonction de la bibliothèque ou en ajouter d'autres.
Lorsque vous avez terminé vos modifications, compilez votre solution, corrigez les éventuelles erreurs de compilation et compilez à nouveau. Suivez ensuite les instructions ci-dessous pour transférer le micrologiciel sur votre Arduino Nano. Il y a deux options ; dans les deux cas, il est supposé que les fichiers de distribution **AVRDUDE** se trouvent dans un dossier nommé *Nano Wave Gen* sur le disque local de votre PC.

Option 1 :

Utilisez l'invite de commande Windows comme précédemment. Vous pouvez utiliser la même méthode que celle décrite dans la rubrique **Installation du micrologiciel**. Il vous suffit de copier le fichier hex généré dans le dossier Nano Wave Gen. Chaque fois que vous appuyez sur *Build Solution*, **Microchip Studio** remplacera le fichier hex dans le dossier du projet, dans un sous-dossier nommé Debug. Si votre dossier de projet s'appelle *Nano-waveform-generator*, le fichier hexadécimal sera écrit dans le sous-dossier *Nano-waveform-generator\Debug*. Avant la programmation, donnez au fichier hex généré le nom de fichier contenu dans le fichier de commande *Windows Program Nano.bat*.

Option 2 :

Créez un « *Programming Tool* » dans **Microchip Studio**. Cliquez dans le menu *Tools* → *External tools*. Vous devriez voir une boîte de dialogue vous demandant quelques paramètres : dans *Title*, écrivez : *Program Nano* ou tout autre nom de votre choix. Dans *Command*, écrivez : *C:\Nano Wave Gen\avrdude.exe*. Dans *Arguments*, écrivez (sur une seule ligne) :

```
-C "C:\Nano Wave Gen\avrdude.conf" -p atmega328p -c arduino -P COM4 -b 115200 -U flash:w:"$(ProjectDir)Debug\$(TargetName).hex":i
```

Remplacez COM4 par le port COM réel auquel votre carte Nano est connectée, et sachez que certains clones d'Arduino Nano ne supportent que des débits en bauds inférieurs. Cochez la case *Use output window*. Cliquez sur *OK*. C'est fait.
Vous devriez voir apparaître une nouvelle option, *Program Nano*, dans le menu *Tools*. Une fois le code construit, il peut être programmé dans la carte Nano en cliquant simplement sur l'option *Program Nano* dans le menu *Tools*.

(équivalent décimal approximatif). Pour les nombres signés, la plage est comprise entre -127,99 et +127,99. La résolution est de 1/256 ≈ 0,004, ce qui est suffisant pour les applications de faible précision.
Un nombre à virgule fixe de 32 bits est souvent composé d'une partie entière de 16 bits et d'une partie fractionnaire de 16 bits. Pour les nombres signés, la plage est approximativement comprise entre -32 000 et +32 000. La résolution (précision) dans ce cas est d'environ 1/65 000 ≈ 0,00001, ce qui est plus qu'adéquat pour notre générateur de formes d'ondes. La précision de la fréquence du signal de sortie n'est limitée que par l'horloge du MCU.
Dans notre oscillateur à table d'ondes, le pas d'angle de phase et l'angle de phase (point d'échantillonnage de l'onde) sont représentés par des nombres à virgule fixe de 32 bits. La partie entière de l'angle de phase est utilisée comme index de tableau pour extraire un point d'échantillonnage de la table. La formule reliant le « pas d'angle de phase » à la fréquence de l'oscillateur est la suivante :

$$PhaseStep = OscFreq \times (TableSize / SampleRate)$$

où *PhaseStep* est la distance entre les points d'échantillonnage dans la table d'ondes, OscFreq est la fréquence de sortie requise (Hz), *TableSize* est le nombre total d'échantillons dans la table d'ondes et *SampleRate* est la fréquence d'échantillonnage (32 kHz).
Bien que les tables d'ondes puissent avoir une taille arbitraire, dans les limites de la mémoire du programme, la taille de la table doit être cohérente avec la résolution du signal de sortie. En règle générale, la taille doit être du même ordre que la valeur maximale des échantillons de la table d'ondes. Par exemple, si les valeurs des échantillons de sortie sont des nombres non signés de 8 bits, une taille de table d'ondes appropriée serait de 256 échantillons. Une taille de table plus importante offre le même avantage que l'interpolation entre les points d'échantillonnage, c'est-à-dire une réduction de la distorsion de la forme d'onde.

Repliement de signal

Si vous avez quelques connaissances en matière de traitement numérique du signal, vous connaissez peut-être le terme « repliement », ainsi que ses causes et ses effets. Le repliement est la distorsion d'une forme d'onde échantillonnée qui se produit lorsqu'une composante de fréquence de la forme d'onde d'entrée analogique est trop élevée par rapport à la fréquence d'échantillonnage. La distorsion se manifeste par l'introduction de composantes de fréquence indésirables dans le signal échantillonné.
Par exemple, si un signal contient une composante de fréquence de 16 kHz et est échantillonné à 20 kéch/s, il y aura de nouvelles composantes indésirables dans le signal de sortie (reconstruit) aux fréquences somme et différence, c'est-à-dire à 4 et 36 kHz. La composante à 4 kHz se situe bien dans la plage audible et constituerait un artefact perceptible. Les ondes sinusoïdales pures à des fréquences inférieures à la moitié de la fréquence d'échantillonnage ne subiront aucun effet de repliement. Les formes d'ondes riches en harmoniques d'ordre élevé, telles que les ondes carrées, impulsions et dents de scie, sont très sensibles à l'effet de repliement.
De même, le repliement se produit avec les oscillateurs à table d'ondes lorsqu'une forme d'onde représentée par des points d'échantillonnage d'une table d'ondes contient des composantes de fréquences supérieures à la moitié de la fréquence d'échantillonnage, sauf lorsque le rapport entre la fréquence de sortie et la fréquence d'échantillonnage est un nombre rationnel. Un choix judicieux de la taille de la table d'ondes par rapport à la fréquence d'échantillonnage peut éliminer le repliement à certaines fréquences de sortie. Dans ce générateur de formes d'ondes, la table contient 640 échantillons et la sélection de fréquences de sortie fixes a été choisie pour éviter les effets de repliement.

Pour finir

Comme indiqué précédemment, il ne s'agit pas d'un projet finalisé mais plutôt d'une démonstration technique de ce qui peut être fait avec un Arduino Nano bon marché. N'hésitez pas à modifier le code et le circuit selon vos besoins ou à l'utiliser comme source d'inspiration pour vos propres projets.

VF : Denis Lafourcade – 220641-04

Produits

- **OWON HDS1021M-N Oscilloscope à 1 voie (20 MHz) + Multimètre**
  https://elektor.fr/18778
- **Robert Sontheimer,** *Arduino & Co. — Measure, Control, and Hack*
  https://elektor.fr/20243

À propos de l'auteur

Michael Bauer est un ingénieur chevronné qui s'intéresse à la technologie de la musique électronique (par exemple, les instruments à vent et la synthèse sonore DSP). Il vit à Victoria, en Australie.

Des questions, des commentaires ?

Envoyez un courriel à Elektor (redaction@elektor.fr) ou contactez l'auteur (mjbauer@iprimus.com.au).

LIENS

[1] Téléchargements du projet sur Elektor Labs : https://elektormagazine.fr/labs/nano-waveform-generator
[2] Téléchargements sur Elektor Labs : https://elektormagazine.fr/labs/nano-waveform-generator
[3] Fiche technique de l'ATmega328P : https://tinyurl.com/4w2uzpy2
[4] Microchip Studio : https://microchip.com/en-us/tools-resources/develop/microchip-studio
[5] Getting Started with Microchip Studio : https://tinyurl.com/39bsmzbd

Nous vous aidons à affiner les grandes lignes de votre idée pour la concrétiser.

Quand vous rêvez d'avenir et d'innovation, les services d'ingénierie d'Arrow vous aident à faire évoluer votre idée de l'imagination à la réalité.

Plus d'informations sur **arrow.com/fiveyearsout**

ARROW
Five Years Out