26 ouverture de porte pour les musiciens

Jörg Trautmann (Allemagne)

Source : Adobe Stock

Vous avez encore oublié la clé de votre porte d'entrée ? Avec cet ouvre-porte, il vous suffira à l'avenir d'un peu de talent de musicien pour entrer dans votre appartement. Cette serrure de porte innovante est activée par une séquence de sons en do majeur. Jusqu'à cinq notes peuvent être chantées, sifflées ou jouées sur un instrument pour la déverrouiller. Le système reconnaît les mélodies sur quatre octaves, de 131 à 1 976 Hz, ce qui le rend compatible avec des voix et des instruments différents.

L'Arduino Nano est au coeur du circuit, qui utilise les périphériques et les composants suivants :

- Module LCD TC1604A-05 de Tinsharp
- Module Iduino ST1146 pour l'enregistrement des sons
- Relais Joy-IT COM-KY019RM pour activer l'ouvre-porte électrique
- Boutons vert, rouge et blanc pour le contrôle et la configuration
- Condensateur de 100 nF
- Résistance de 470 Ω
- Résistance de 10 kΩ
- Potentiomètre de 1 kΩ
- Diode 1N400

Le schéma de câblage de la **Figure 1** montre comment je l'ai branché. L'Arduino coordonne tout, connecté au module LCD via le bus 8 bits D5 à D12. D13 contrôle le module relais, tandis que l'entrée analogique A0 écoute le module microphone.
Il y a un point important à considérer en ce qui concerne l'alimentation électrique. Des tests ont montré que le port USB d'un ordinateur portable fournit une tension de 5 V assez propre, et convient parfaitement à cette application. Si l'ouvre-porte est utilisé avec un chargeur USB standard, la tension continue mal lissée provoque diverses fréquences d'interférence, de sorte qu'un fonctionnement fiable n'est pas garanti. C'est pourquoi je recommande vivement l'utilisation d'une alimentation séparée de bonne qualité.

Le réglage des fréquences

Le programme Arduino est disponible sur la page *Elektor Labs* de ce projet [1]. Le signal audio basse fréquence est envoyé à l'une des entrées analogiques de l'Arduino. La première étape consiste à déterminer la fréquence de ce signal d'entrée. La méthode éprouvée utilisée ici est celle de la transformée de Fourier rapide (FFT). Vous n'avez pas à vous préoccuper du processus laborieux de création des fonctions nécessaires, car la bibliothèque *arduinoFFT* les possède déjà.
Il a fallu expérimenter pour trouver la fonction de fenêtrage la mieux adaptée à la mesure de la fréquence. Les fonctions de fenêtre de Hamming et de Hann, qui ont toutes deux une forme sinusoïdale, ont posé des problèmes. Elles conduisent toutes deux à une valeur de crête large et à de faibles oscillations secondaires. La fenêtre de Hann s'approche de zéro aux deux extrémités, ce qui élimine les discontinuités. Avec la fenêtre de Hamming, le zéro n'est pas tout à fait atteint, de sorte que le signal peut présenter de légères discontinuités. En raison de cette différence, la fenêtre de Hamming supprime mieux le lobe latéral suivant, mais est moins efficace pour supprimer tous les autres. En fin de compte, de nombreux tests ont montré que la méthode Hann est la version la plus appropriée pour mon application.
La limite supérieure de la fréquence à mesurer étant de 1 976 Hz, il faut une fréquence d'échantillonnage de 4 096

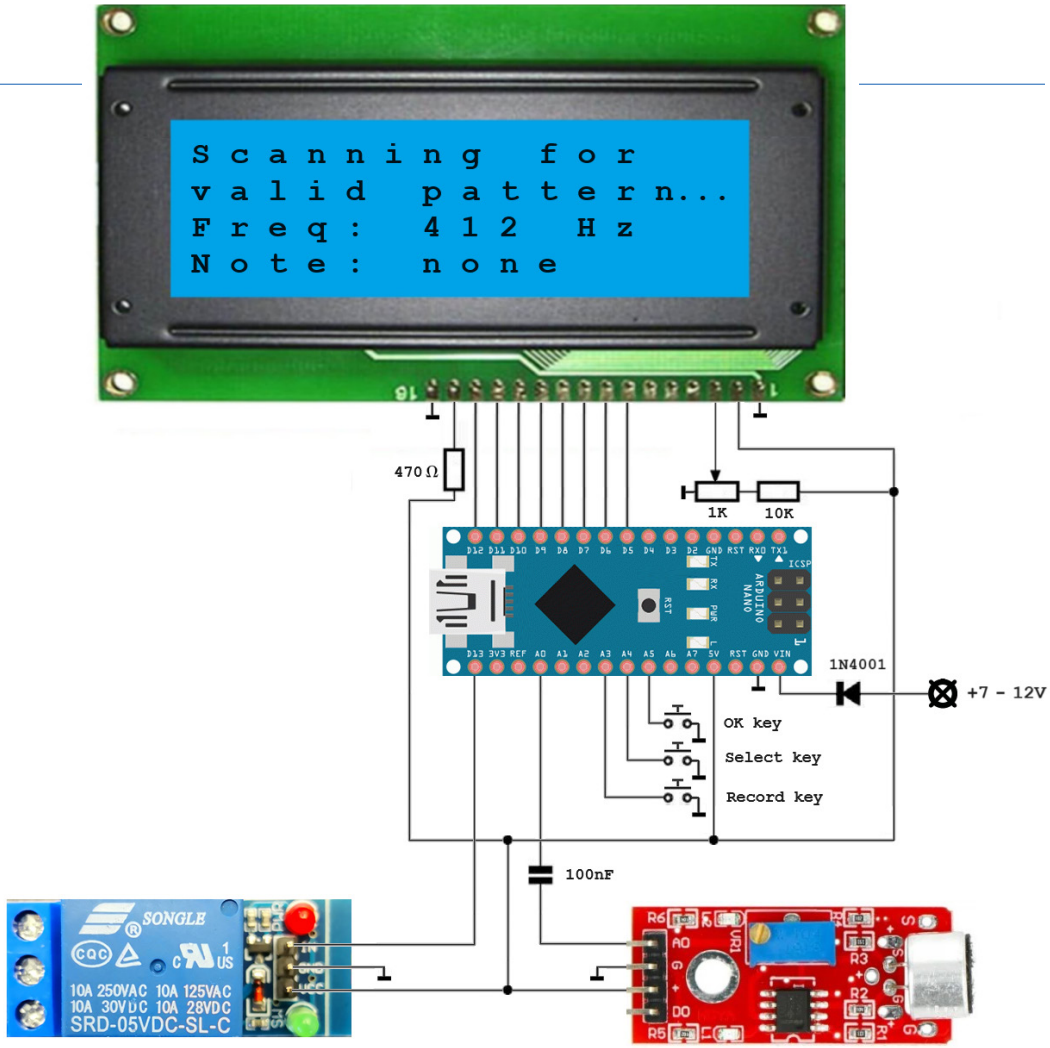


Figure 1. Schéma du projet.

Hz, ce qui donne une valeur supérieure théorique de 2 048 Hz. Pour obtenir une bonne précision de la fréquence définie, j'ai calculé expérimentalement un facteur de correction, qui est inclus dans le calcul, comme indiqué dans le **Listage 1**.

Menu structurel

Ensuite, nous devons créer une structure de menu intuitive pour faire fonctionner l'ouvre-porte. Après plusieurs versions, j'ai trouvé une solution, que vous pouvez voir dans la rubrique **figure 2**.

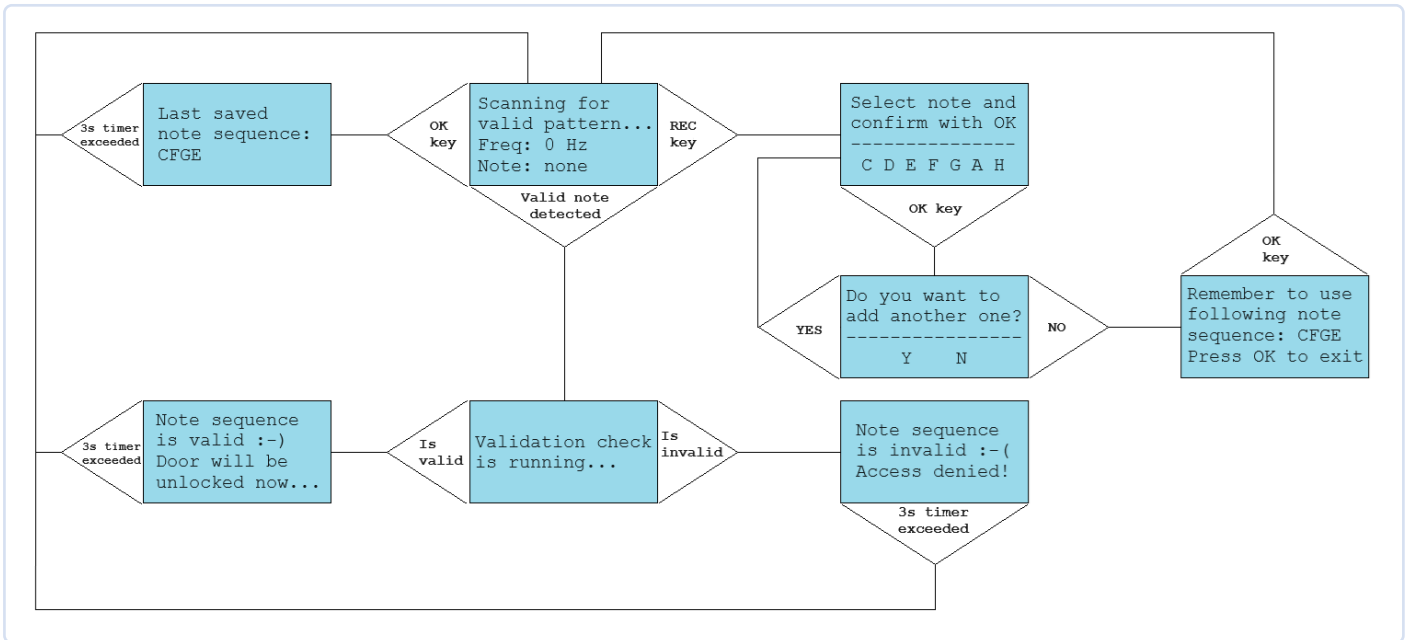


Figure 2. Organigramme du programme.

L'écran de démarrage initial reste en mode de balayage permanent et affiche la fréquence actuelle détectée et, si elle est disponible, la note de do majeur associée. Afin d'obtenir le taux de réussite le plus élevé possible, un écart de fréquence de ±9 Hz est accepté. Si la première note d'une séquence mémorisée est reconnue correctement, le programme passe en mode validation et vérifie la séquence complète. Chaque tonalité doit être maintenue pendant au moins 2 secondes avant que le son suivant ne soit répété. Si le test de la séquence est réussi, un message de confirmation est lancé, le relais est activé via un port de sortie Arduino et l'ouvre-porte libère la serrure pendant environ 3 secondes. Si la validation de la séquence échoue, une note correspondante est émise et, après environ 3 secondes, le système revient à l'écran de démarrage.

Pour définir la séquence de notes, il faut appuyer sur le bouton rouge *Record.* Une boîte de dialogue apparaît, dans laquelle une note peut être sélectionnée à l'aide du bouton blanc *Select.* La sélection est confirmée par le bouton vert *OK.* Une boîte de dialogue apparaît ensuite, demandant si une autre note doit être ajoutée. La touche *Select* permet de sélectionner « *Y* » ou « *N* » et la touche *OK.* permet d'appeler la boîte de dialogue suivante. Après cinq notes, la longueur maximale de la séquence est atteinte et l'écran de départ apparaît à nouveau. Si vous ne vous souvenez plus de la séquence enregistrée, vous pouvez l'afficher en appuyant sur le bouton *OK*.

Comme la séquence de notes sauvegardée doit rester disponible après une coupure de courant, elle est stockée dans l'EEPROM.

Installation et mise en service

L'affectation des broches de l'Arduino Nano est choisie de manière à ce qu'il y ait un câblage 1:1 avec le module LCD. Comme il n'y a que quelques composants, je n'ai pas utilisé de circuit imprimé et j'ai travaillé avec un câblage en fils « volants » . Le trimmer du module microphone est ajusté de façon à ce qu'une seule LED s'allume.

Si vous avez une bonne voix, vous pouvez maintenant essayer la gamme de do majeur. Si une note est reconnue, elle s'affiche avec la fréquence associée. Bien entendu, il est également possible d'utiliser un générateur de sons et d'enregistrer la séquence correspondante, par exemple sur votre smartphone, afin de pouvoir la réécouter ultérieurement. Cependant, la lecture dépend beaucoup du volume du smartphone utilisé et de la qualité du haut-parleur. En chantant directement, le volume sonore n'a pas posé de problème et la reconnaissance des notes fonctionne de manière très fiable.

Une autre remarque à propos du module LCD utilisé. Au lieu du module TC1604A-05, une autre version peut également être utilisée, mais vous devrez alors adapter la ligne suivante du programme en conséquence, si le nombre de colonnes est différent :

```
lcd.begin(16, 4);
// Initialize LCD with 16 columns and 4 rows
```

Trucs et astuces

Ce montage convient également parfaitement pour stimuler la bonne humeur lors d'une fête. Tout le monde essaie d'être un chanteur pour faire sauter le verrou acoustique récalcitrant. Et par exemple en cas de succès, une sirène retentit.

Pour illustrer le fonctionnement de ce circuit, j'ai créé une courte vidéo de démonstration [2].

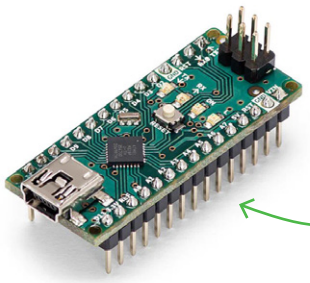
VF : Laurent Rauber — 240066-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (joerg.trautmann@gmx.de), ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Jörg Trautmann, passionné d'électronique depuis toujours, a commencé son parcours avec une TI-99/4A de Texas Instruments au début des années 1980. Après des décennies en tant que développeur de matériel puis de logiciel chez OpenText, il a pris sa retraite en 2023. Aujourd'hui, Monsieur Trautmann partage sa passion en enseignant la soudure et la construction de circuits à des élèves du primaire. Sa carrière couvre l'évolution de l'électronique, des tubes à vide à la technologie moderne, et continue d'inspirer la génération future.



Produits

- **Arduino Nano**
  www.elektor.fr/17002

Listage 1. Exemple de code.

```
// Create Fast Fourier Transformation object
arduinoFFT FFT = arduinoFFT();
// Sampling period
int samplingPeriod = 0;

…

// Count of samples
const int SAMPLES = 128;
// Sampling frequency - frequencies up to 2048 Hz can be handled
const int SAMPLING_FREQUENCY = 4096;
// Correction factor to be used for frequency calculation
const double FREQUENCY_CORRECTION_FACTOR = 0.033;
// Frequency measurement starts at 100 Hz
const int MIN_FREQUENCY = 100;

…

int getInputFrequency() {
  double vReal[SAMPLES]; // Vector for real values
  double vImag[SAMPLES]; // Vector for imaginary values

  // Cycle through the number of samples
  for(int i  = 0; i < SAMPLES; i++)
  {
      // Current microseconds value
      long microSeconds = micros();
      // Save microphone input value
      vReal[i] = analogRead(ANALOG_INPUT);
      vImag[i] = 0; // Save static imaginary value
      while(micros() < (microSeconds + samplingPeriod)) {
        // Wait until time for next sample is reached
      }
  }

  // Execute Fast Fourier Transform calculations on samples
  FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HANN, FFT_FORWARD);
  FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
  FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);
  double peakFrequency =
    FFT.MajorPeak(vReal, SAMPLES, SAMPLING_FREQUENCY);
  double correctedFrequency =
    peakFrequency - peakFrequency * FREQUENCY_CORRECTION_FACTOR;

  // Convert value to integer value
  return correctedFrequency < MIN_FREQUENCY ?
    0 : round(correctedFrequency);
}
```

LIENS

[1] Code source sur Elektor Labs :
https://www.elektormagazine.fr/labs/door-opener-for-musical-talents
[2] Video de démonstration du projet : https://youtu.be/iI9sYQETmpw