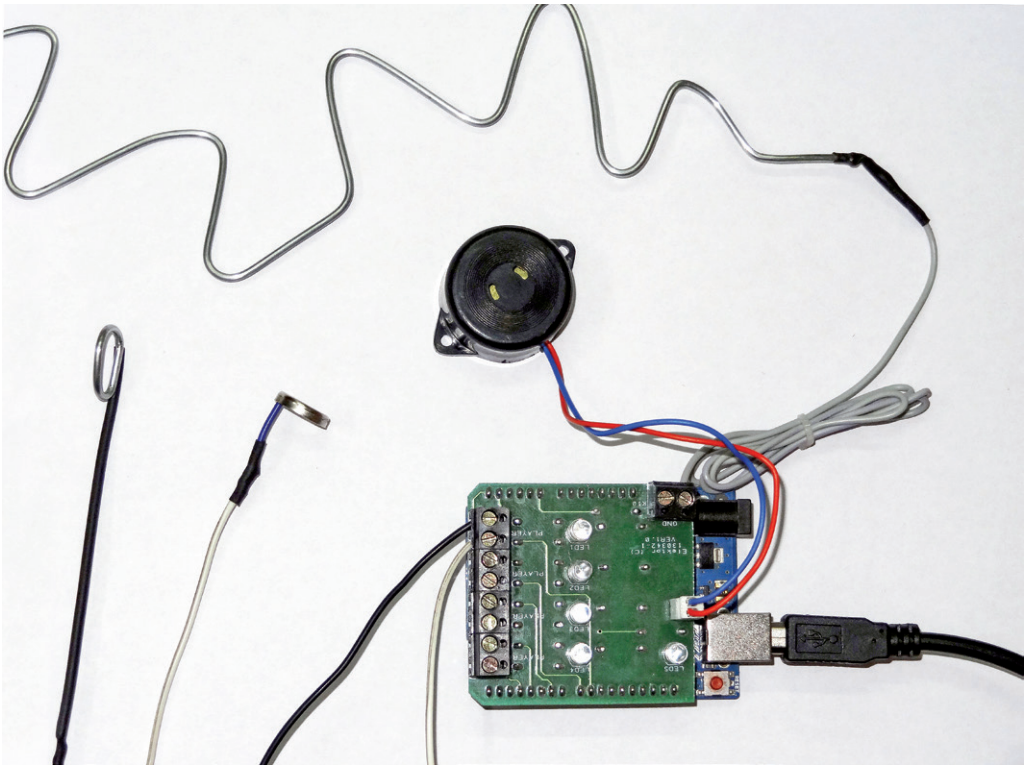
zigzag électrique
jeu d'adresse à base d'Arduino



Sunil Malekar
(Elektor Labs, Inde)

Le jeu d'adresse « zigzag électrique » consiste à déplacer une boucle métallique le long d'un fil biscornu sans jamais le toucher. Nous avons ici troqué contre une carte Arduino Uno les NE555 qui étaient au cœur d'une version de ce jeu [0] publiée par Elektor il y a quelques années. Dans cette déclinaison numérique et multijoueur, le vainqueur est le premier qui termine le parcours sans avoir déclenché le signal de faute.

Une victoire qui ne tient qu'à un fil

Oubliez intelligence, stratégie et autre don pour les coups fourrés, le zigzag électrique est un jeu de pure dextérité. Dans notre version Arduino, chaque joueur, quatre au plus, dispose de sa propre boucle. Le but du jeu est de terminer le parcours sans toucher le fil (ou tube) conducteur. Si la boucle entre en contact avec le fil, un son retentit et la LED associée au joueur s'allume. Le gagnant est celui qui touche le point d'arrivée après un parcours sans faute. Une brève mélodie célèbre la victoire et toutes les LED clignotent.

Le circuit

Les composants principaux du circuit sont les boucles, le fil conducteur zigzagant, l'indicateur de faute et l'avertisseur sonore. La taille du circuit dépend du nombre de joueurs ; notre version en supporte jusqu'à quatre. La boucle de chaque participant est interfacée avec une broche Arduino à l'aide d'une résistance de rappel vers le haut (V_{CC}). Le point

d'arrivée d'un joueur et le microcontrôleur sont quant à eux interfacés avec une résistance de rappel vers le bas (GND).
La broche associée est normalement au niveau logique haut lorsque le joueur tient la boucle. Le fil rigide en cuivre ou en acier est relié à la masse. Le niveau haut passe au niveau bas si un joueur touche le fil avec la boucle. Arduino détecte le changement et répond instantanément en allumant les LED idoines et en activant le *buzzer*. À chaque joueur sont associés un son et un motif de LED particuliers. Le participant qui touche son point d'arrivée sans avoir touché le fil gagne la partie, victoire que fête le *buzzer* avec une courte fioriture.
La carte UNO révision 3 qui forme le cœur de ce projet exploite un microcontrôleur ATmega328P (**fig. 1**). Ses broches GPIO sont connectées à la carte d'extension (c.-à-d. le circuit de notre jeu) via K1, K2, K3 et K4. Ces connecteurs relient les points d'arrivée (Win1, Win2, Win3, Win4) et les boucles (Loop1, Loop2, Loop3, Loop4) en utilisant respectivement les broches de l'ATmega équipées de résistances de rappel au niveau bas (R7, R8, R9 et R10) et de résistances de rappel au niveau haut (R11, R12, R13 et R14). K10 sert à mettre à la masse le fil du parcours. Boucles et points d'arrivée sont mis en corrélation avec les broches GPIO 3, 4, 6, 7, 8, 9, A4 et A5 (selon le schéma de l'Arduino Uno R3). Les quatre LED indicatrices sont reliées aux broches 13, 12, 11 et 10 ; elles indiquent une faute lorsqu'une boucle (loop1, loop2, loop3 ou loop4) touche le fil (ou le tube). Le *buzzer* (avec oscillateur incorporé) émet alors un son personnalisé. La LED5 reliée à la broche A0 de l'Arduino indique soit une partie prête à être lancée, soit une partie gagnée.

Assemblage

La **figure 2** montre la carte à double face et trous métallisés sur laquelle a été implanté le circuit. Vous pouvez télécharger le dessin du CI depuis [1]. Il n'y a pas de mot pour rendre justice à la simplicité de cette carte d'extension qui ne contient que des composants traversants. Les résistances et les quatre

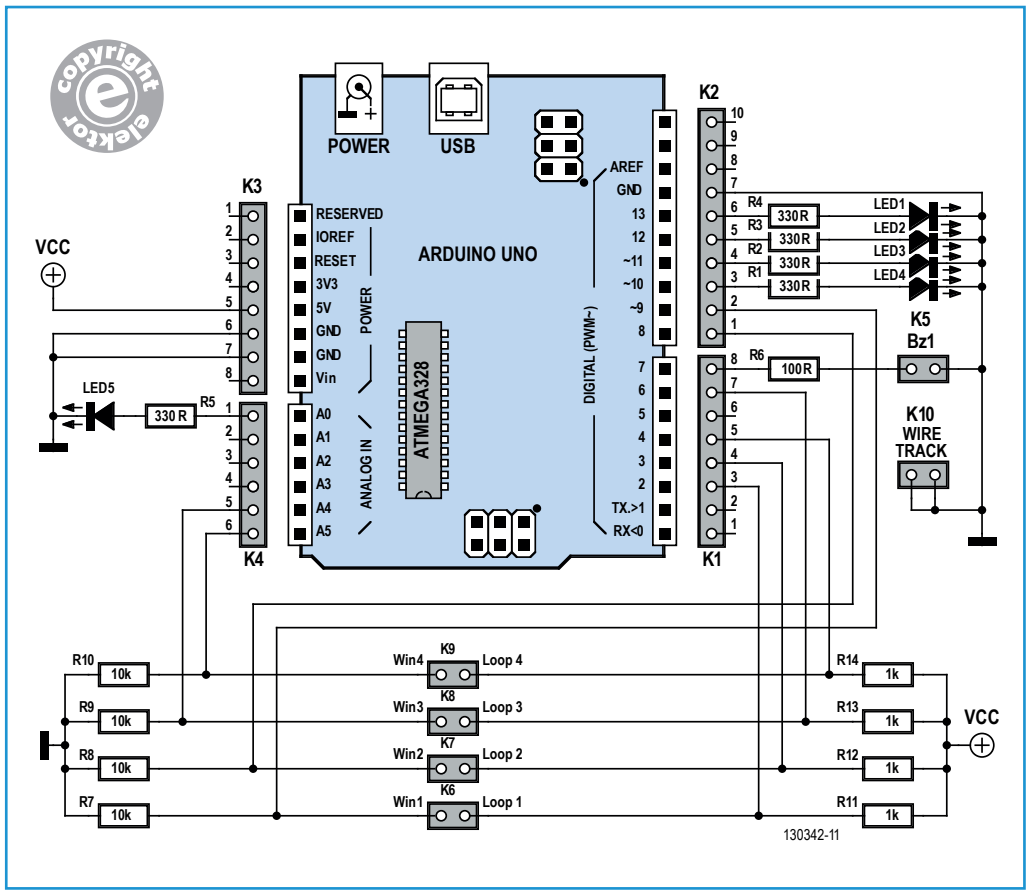


Figure 1.
Schéma du zigzag électrique, un *shield* unique pour la carte Arduino Uno.

embases sont placées côté cuivre, sachant que les pattes des embases ne peuvent être soudées que côté composants. Les fils conducteurs des boucles ainsi que le fil du jeu sont reliés à de robustes borniers à vis encartables. En passant, rappelons que les cartes d'extension pour Arduino sont souvent appelées *shields*, et qu'un *shield* s'enfiche dans les connecteurs d'extension prévus à cet effet. Peut-être a-t-on choisi de parler de bouclier (= *shield*) parce qu'il aide en quelque sorte Arduino à composer avec le monde réel.

Programme

Le zigzag électrique fait appel à une carte à microcontrôleur Arduino. Le code a été écrit avec l'EDI d'Arduino. Vous pouvez le télécharger gratuitement depuis [2].

La configuration du micrologiciel dépend du nombre de joueurs, quatre au plus dans notre version. Les points d'arrivée individuels, la LED qui indique une faute, de même que

Liste des composants

Résistances

R1 à R5 = 330 Ω
R6 = 100 Ω
R7 à R10 = 10 kΩ
R11 à R14 = 1 kΩ

Semi-conducteurs

LED1 à LED4 = LED, rouge, 5 mm, p. ex. Farnell 1780754
LED5 = LED, verte, 5 mm, p. ex. Farnell 2112108

Divers

K1, K3 = embase à 8 contacts, pas de 2,54 mm
K2 = embase à 10 contacts, pas de 2,54 mm
K4 = embase à 6 contacts, pas de 2,54 mm
K6 à K10 = bornier pour CI à 2 voies, au pas de 5 mm
buzzer, p. ex. Farnell 1022400
circuit imprimé 130342

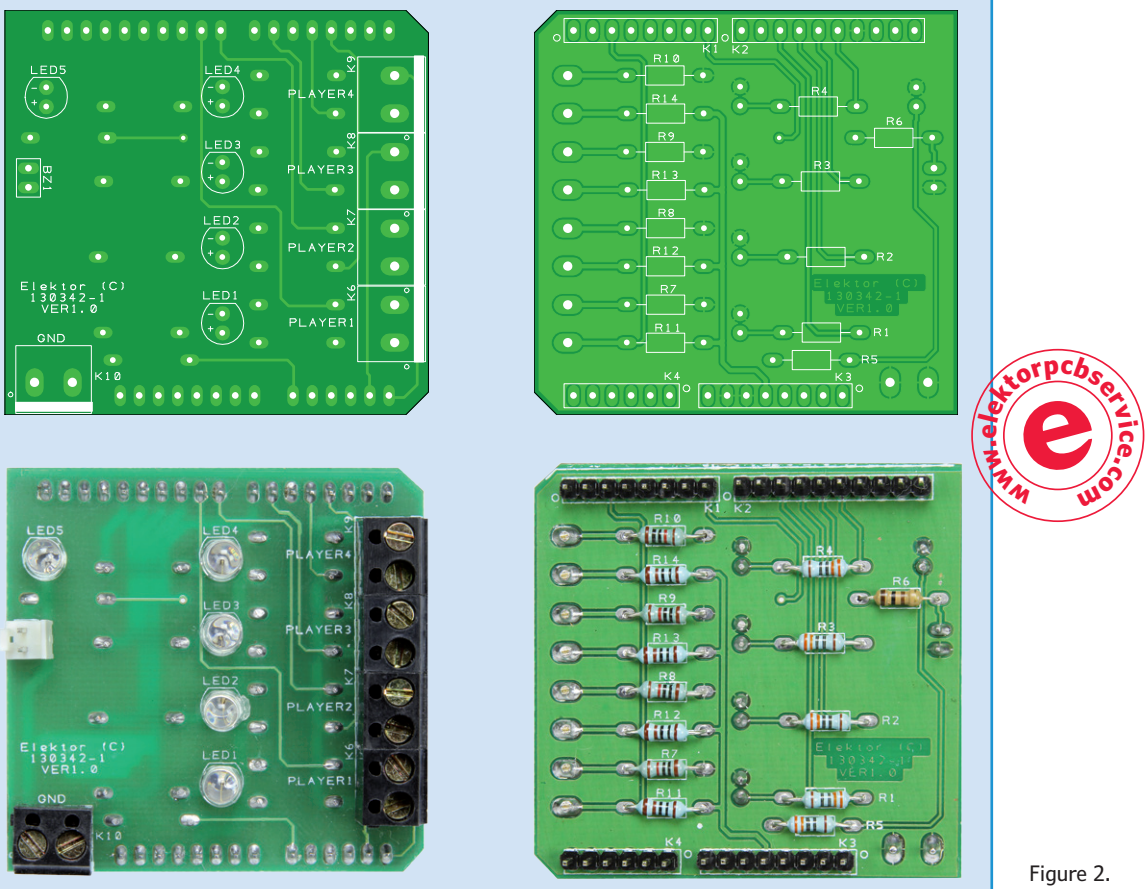
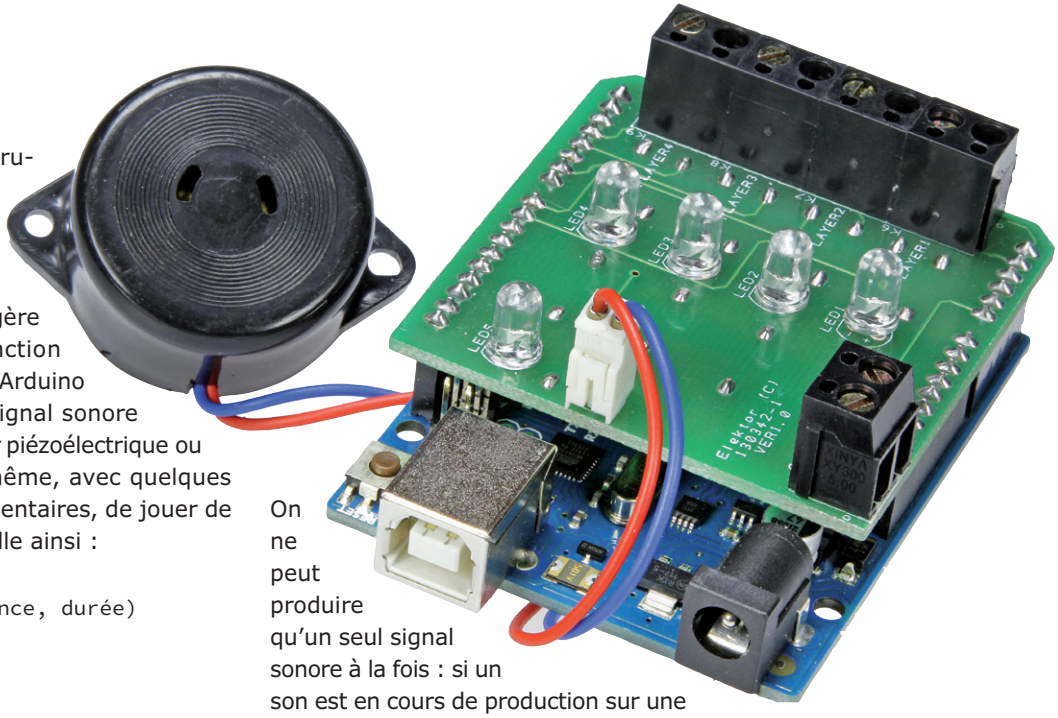


Figure 2.
Dessin du CI du zigzag électrique.



le *buzzer* qui signale bruyamment cette faute, sont configurés avec les broches GPIO normales de l'ATmega.
La partie du code qui gère le *buzzer* utilise la fonction `tone()`. Cette fonction Arduino permet d'envoyer un signal sonore sur la sortie d'un *buzzer* piézoélectrique ou d'un haut-parleur et même, avec quelques lignes de code supplémentaires, de jouer de la musique. Elle s'appelle ainsi :

```
tone (broche, fréquence, durée)
```

ou comme ceci :

```
tone (broche, fréquence)
```

Exemple :

```
tone (3,440)
```

Les arguments sont :
`broche` le numéro de la broche GPIO à laquelle est relié le buzzer/haut-parleur ;
`fréquence` la fréquence du signal sonore exprimée en Hz ;
`durée` la durée du signal sonore exprimée en ms.

Pour arrêter le signal, il suffit d'appeler la fonction `noTone` :

```
noTone (pin)
```

La fonction `tone()` produit sur la broche indiquée un signal carré qui a la fréquence spécifiée (et un rapport cyclique de 50 %). Si aucune durée n'est passée à la fonction, le signal continue tant que `noTone()` n'a pas été appelée.

On ne peut produire qu'un seul signal sonore à la fois : si un son est en cours de production sur une autre broche, l'appel de la fonction `tone()` est sans effet. Si la fonction `tone()` est appelée une nouvelle fois pour la même broche, la fréquence du signal est celle passée en argument.

Pour construire rapidement une mélodie très simple à l'aide de notes de différentes hauteurs, il existe un fichier d'en-tête très pratique : *pitches.h*. Ce fichier contient sous forme de constantes symboliques les fréquences de nombreuses notes ; NOTE_FS4 correspond p. ex. à un fa dièse de 370 Hz.

Bibliothèque Tone

Les notes ont des durées, hauteurs, intensités et timbres bien définis [4]. Leur durée et leur hauteur (les octaves) peuvent être modifiées dans le fichier *pitches.h*.
Les fonctions de la bibliothèque *Tone* facilitent la production des notes de musique. Elle peut être téléchargée depuis [3]. Le lien [5] illustre le fonctionnement de la fonction `tone()` et du fichier *pitches.h*.

Liens

[0] www.elektor-magazine.fr/110100
[1] www.elektor-magazine.fr/130342
[2] www.elektor-magazine.fr/post
[3] http://code.google.com/p/rogue-code/wiki/ToneLibraryDocumentation
[4] http://fr.wikipedia.org/wiki/Note_de_musique
[5] http://arduino.cc/en/Tutorial/Tone

Tableau 1. Constantes symboliques	
Constante	**Fréquence (Hz)**
NOTE_B2	123
NOTE_C3	131
NOTE_CS3	139
NOTE_D3	147

Le micrologiciel du jeu produit plusieurs notes à l'aide des constantes définies dans le fichier *pitches.h*. Quelques-unes sont listées dans le **tableau 1**.

Programmation Arduino

La Uno peut être programmée depuis l'EDI Arduino grâce au chargeur de démarrage interne de la carte. Pour ce projet, nous avons utilisé un programmateur externe. Si vous procédez comme nous, veillez à paramétrer correctement les bits de configuration (**fig. 3**).

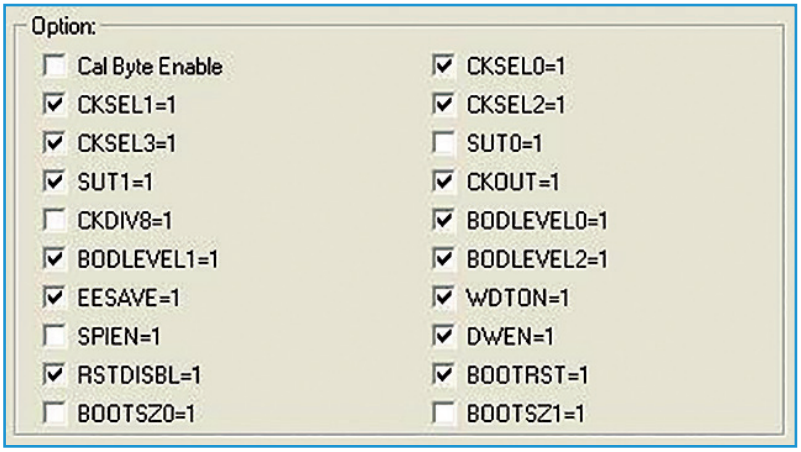


Figure 3. Configurez tels quels les fusibles si vous utilisez un programmateur externe.

Et maintenant boucle-la !

Le montage final est alimenté par l'adaptateur secteur USB/CC de 12 V. À la mise sous tension de la carte, la LED5 clignote en permanence pour indiquer que la partie peut commencer. Le joueur qui boucle son parcours sans avoir touché le fil est honoré par un *son et lumière* orchestré par l'Arduino. Pour commencer une nouvelle partie, appuyez sur le bouton d'initialisation de la carte Arduino.

(130342 – version française : Hervé Moreau)