Générateur de notes aléatoires MIDI

Bonjour!

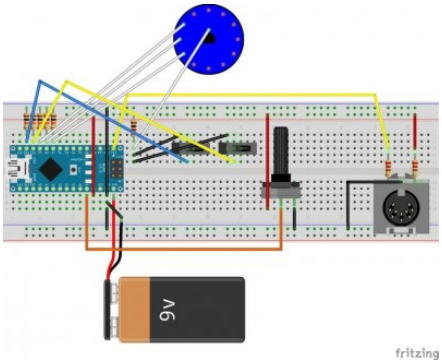
Pourquoi?

Il s'agit ici de fabriquer un géné...oui, bon, tu as déjà lu le titre... Ce sera un appareil physique, avec une sortie MIDI physique donc, pas USB mais en jack 3.5.

Ce projet est né suite au dépoussiérage d'un vieux sampler qui a une connectique MIDI complète.

Comment?

Donc, à force de fouiller l'intermouette, on fini par tomber sur des projets déjà existant, qui ressemblent comme deux gouttes d'eau à ce qu'on avait en tête. Et pour nous cela donne ceci: Midi Random Sequence Generator



Les crédits de cette image reviennent à l'auteur de la page Instructable.

C'est basé sur un Arduino Nano. L'idée du projet est de générer des séquences de notes MIDI aléatoires. Plusieurs longueurs pour ces séquences sont disponibles: 4, 7, 8 et 16 pas. Il est possible de stopper la génération de séquence avec le premier des interrupteurs et de faire tourner en boucle infinie la séquence en cours avec le second interrupteur.

Voici le code:

```
#include <MIDI.h>

#define redLed 13
#define pot A7
#define fourth 7
#define seventh 8
#define eighth 9
#define switchOne 11
#define switchTwo 10

MIDI_CREATE_DEFAULT_INSTANCE();

int notes[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int scale[] = {0, 2, 3, 5, 7, 9, 10, 12, 14, 15, 17, 19, 21, 22, 24, 26, 27, 29, 31, 33, 34, 36};
int seqLength;
int range = 21;
int bpm;
int rootNote = 35;
int potValue;
int four;
int seven;
int eight;
int hold;
int loops = 4;
int count = 1;

void setup() {
  pinMode(redLed, OUTPUT);
  pinMode(pot, INPUT);
  pinMode(fourth, INPUT);
```

```
  pinMode(seventh, INPUT);
  pinMode(eighth, INPUT);
  pinMode(switchOne, INPUT);
  pinMode(switchTwo, INPUT);
  digitalWrite(0, LOW);

  //Serial.begin(115200);
  MIDI.begin(MIDI_CHANNEL_OFF);

  potValue = analogRead(pot);
  randomSeed(potValue);

}

void loop() {
  //Serial.println(count);
  hold = digitalRead(switchOne);
  if (count == 1){

    digitalWrite(redLed, HIGH);
    four = digitalRead(fourth);
    seven = digitalRead(seventh);
    eight = digitalRead(eighth);
    hold = digitalRead(switchOne);
    if (four == 1){
      seqLength = 4;
    }
    else if (seven == 1){
      seqLength = 7;
    }
    else if (eight == 1){
      seqLength = 8;
    }
    else {
      seqLength = 16;
    }
    for (int i = 0; i < seqLength; i++){
      notes[i] = scale[random(0,range)];
    }

    digitalWrite(redLed, LOW);
  }
  if (count <= loops){
    for (int i = 0; i < seqLength; i++){
      potValue = analogRead(pot);
      bpm = 60 + potValue * 0.72265625;         // 60 - 800 bpm quarter notes

      int pause = digitalRead(switchTwo);
      while (pause == 1){
        delay(50);
        pause = digitalRead(switchTwo);
      }
      MIDI.sendNoteOn(rootNote + notes[i], 127, 1);
      digitalWrite(redLed, HIGH);
      //Serial.println(notes[i]);
      //Serial.println(potValue);
      //Serial.println(bpm);
      //Serial.println("");
      delay(60000/bpm*0.75);
      MIDI.sendNoteOff(rootNote + notes[i], 0, 1);
      digitalWrite(redLed, LOW);
      delay(60000/bpm*0.25);
    }
    //Serial.println("");

    hold = digitalRead(switchOne);
    if (count == loops && hold == 1){
      count = loops;
    }
    else{
      count += 1;
    }

  }



  if (count > loops && hold == 0){
    count = 1;

  }
}
```

Premiers essais

Après avoir testé le montage sur une plaque d'essais, il semble que les interrupteurs et le sélecteur de durée de séquence ne fonctionnent pas. Ce qui confirme le ressenti lors de l'écoute. Le reste roule parfaitement, donc on ne change rien de ce côté pour le moment...

Le matériel est donc testé physiquement en dehors du montage. Un simple test de continuité fera l'affaire dans ce cas et,

effectivement, pas de pannes de ce côté-ci. Les interrupteurs ouvrent et ferment le circuit, le sélecteur rotatif de même.

Donc, allons voir du coté de la perspicacité du montage en lui-même, à savoir si les branchements entre les différents composants font ce qu'ils doivent faire en fonction du code.

Test du circuit avec un bout de code

La méthode qui suit a été gentillement soufflée à l'oreille par une porte logique à barbe...

Alors, pour cela nous allons garder le même montage mais changer le code dans le Nano. Nous allons

> tester que tous les petits bidules d'interaction (boutons, pots, etc) fonctionnent correctement hors de la logique du code, il faudrait tester avec un code de ce type qui permettra de le vérifier :

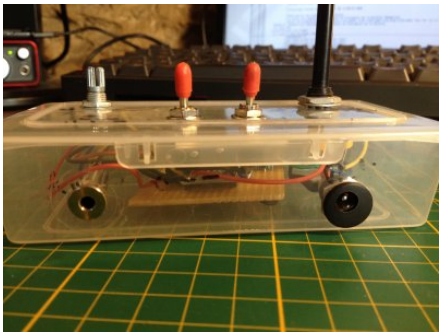
```
#define redLed 13
#define pot A7
#define fourth 7
#define seventh 8
#define eighth 9
#define switchOne 11
#define switchTwo 10

int count; // compteur utile pour ne pas surcharger d'info le moniteur série

void setup() {

  pinMode(pot, INPUT);
  pinMode(fourth, INPUT);
  pinMode(seventh, INPUT);
  pinMode(eighth, INPUT);
  pinMode(switchOne, INPUT);
  pinMode(switchTwo, INPUT);

  Serial.begin(9600);
}

void loop() {

  // Tester les boutons un par un
  if (digitalRead(switchOne)) Serial.println("switch one on");
  if (digitalRead(switchTwo)) Serial.println("switch two on");
  if (digitalRead(fourth)) Serial.println("fourth on");
  if (digitalRead(seventh)) Serial.println("seventh on");
  if (digitalRead(eighth)) Serial.println("eighth on");

  // Envoyer la valeur du pot une fois sur 50 ( = 50 x delay(10) = 2 fois par secondes
  if (count%50 == 0) Serial.println(analogRead(pot));

  count ++;  // augmenter le compteur
  delay(10); // petite pause 10 millisecondes

}
```

Une fois que le code est injecté dans le Nano, nous pouvons observer plusieurs choses dans le moniteur série:

1. qu'il y a des faux contacts, par exemple quand des affichages se font alors qu'aucune manipulation n'a eu lieu et que ces affichages sont aléatoires. Dans notre cas, les résistances en entrées sur D7 à D11 se touchent un peu, cela renvoie donc un peu n'importe quoi... Les pattes des résistances sont donc redressées et cela va déjà un peu mieux.
2. que nous avons la valeur 0 qui est renvoyée en permanence. Cela correspond au potentiomètre. Celui-ci est complètement tourné à gauche, donc c'est bon signe. En le tournant vers la droite, la valeur augmente au fur et à mesure jusqu'à la valeur maximale.
3. les interrupteurs et le sélecteurs rotatif ne font pas leur boulot. À la finale, il s'avère que l'idée de connecter ces trois “bidules” avec une patte commune sur le +5V au travers de la même résistance de 10kOhm n'est pas une bonne idée. Les interrupteurs interagissent entre eux. Quand l'un est ouvert, actionner le second fait croire à l’Arduino que c'est le premier qui est manipulé... Dans le montage final, il faudra donc séparer ces trois pattes pour les relier au +5V.

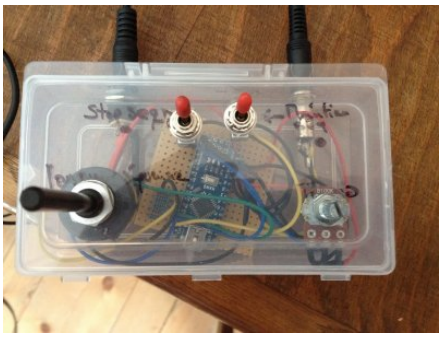
Reste à mettre en boite

Cela donnera pour cette première version du projet quelque chose comme ça:



Sur la photo ci-dessus nous pouvons voir plusieurs choses:

- le connecteur de gauche, argenté, est une sortie MIDI au format jack 3.5. Pour plus de renseignements sur ce format d'adaptateur, cf la page sur ce wiki qui lui sera prochainement consacré.
- le connecteur de droite est l'alimentation. Elle est câblée au standard des pédales d'effet guitare, à savoir le + sur l'extérieur et le - sur le centre. Comme cela, si besoin, on peut utiliser la même alimentation que celles de pédales d'effet.



Sur cette seconde photo, nous avons:

- Tout à gauche, le grand bâton noir est en fait le sélecteur de durée de longueur de séquence
- l'interrupteur de gauche, à capuchon rouge: la commande pour arrêter la séquence en cours. À noter, que quand cet interrupteur est actionné, un signal “note off” est envoyé, donc la séquence s'arrête. Il n'y a pas de note continue , pas comme lorsqu'on appuie au bon moment sur le reset de l'arduino. (note pour plus tard: implémenter un bouton de reset sur le boîtier pour permettre de créer des nappes sonores en envoyant un “note on”).
- l'interrupteur de droite, à capuchon rouge lui aussi, répète la séquence en cours de manière indéfinie.
- en bas à droite, le potentiomètre de contre du tempo. Plus on tourne vers la droite et plus la séquence défile rapidement, et inversement.