

ALTILIUM

Milella Mauro & Signorino Angelica

Indice documentazione:

1) Setup software necessari

2) Arduino

2.1) Il protocollo MIDI






2.2) Il sensore capacitivo

2.3) Il Circuito

2.4) Il Programma

3) Prototipo

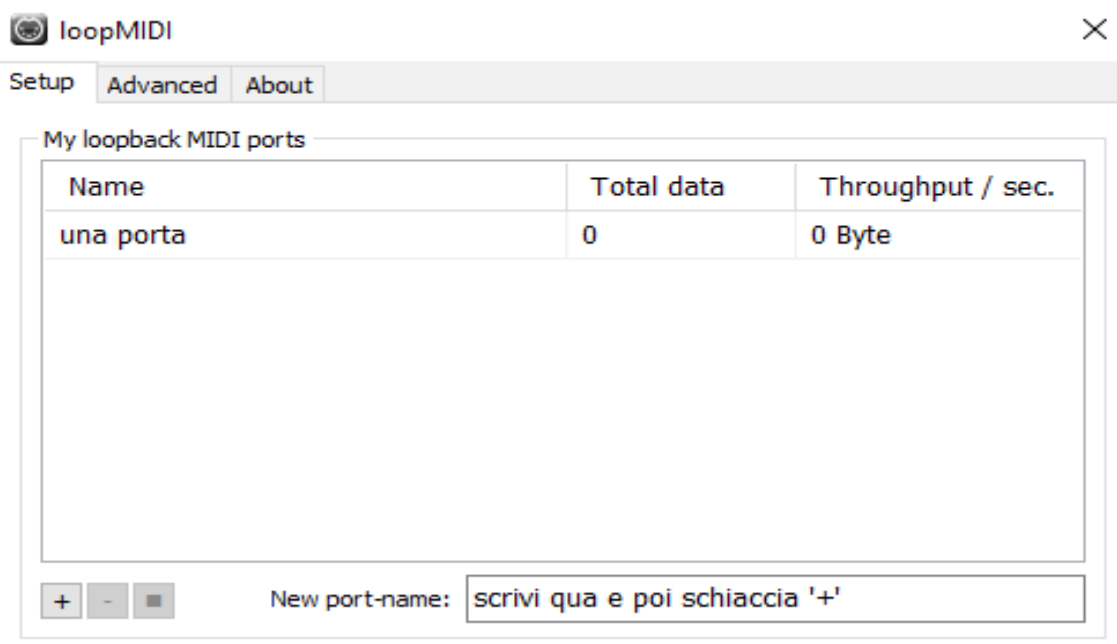
1) *Setup software necessari*

 ardBatteria	25/02/2019 21:50	Cartella di file	
 hairless-midiserial	23/02/2019 16:53	Cartella di file	
 loopMIDISetup.exe	21/12/2018 06:31	Applicazione	7.927 KB
 MTPDK-2.0.4-VST-64bit-Win-FULL.zip	23/02/2019 16:16	Archivio WinRAR ...	55.845 KB
 TracktionInstall_7_Windows_64Bit_latest....	23/02/2019 16:13	Applicazione	20.015 KB

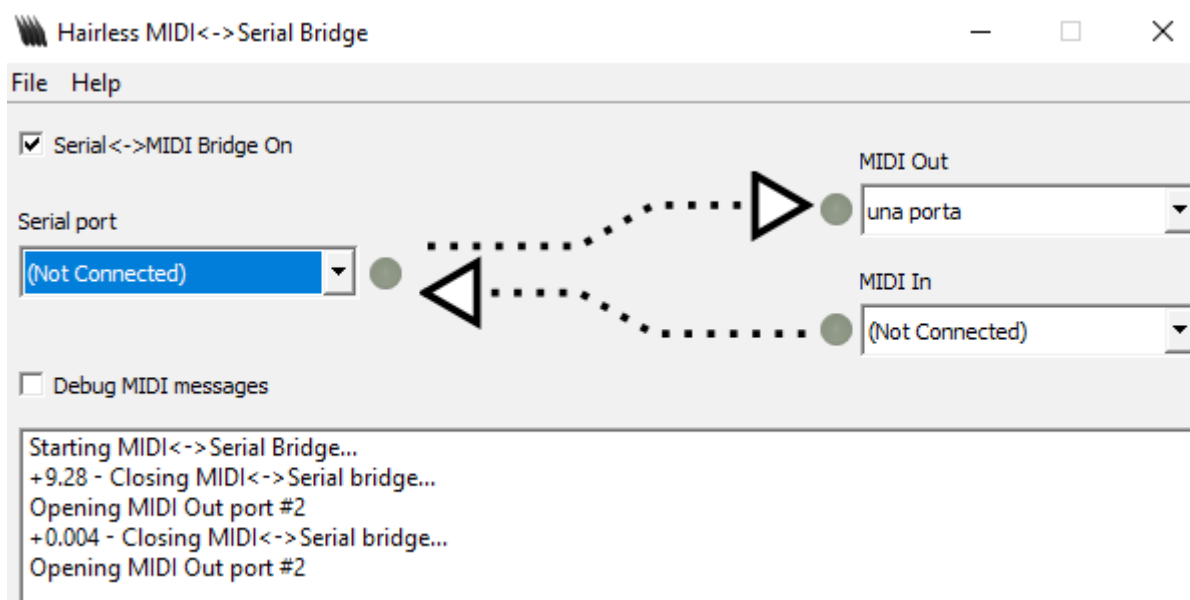
- ardBatteria: contiene lo sketch da caricare sulla scheda ArduinoUNO;
- hairless-midiserial: un software per permettere di indirizzare il flusso dati proveniente dalla porta seriale di Arduino ad un'altra porta virtuale
<http://projectgus.github.io/hairless-midiserial/>
- loopMIDI: permette di creare delle porte virtuali che traducono un segnale qualsiasi ricevuto in MIDI (vedi capitolo 2.2); queste porte possono essere collegate a programmi che si aspettano un input MIDI quindi, partendo da Arduino e passando per loopMIDI (grazie ad hairless-midiserial), terminiamo il ciclo su un qualsiasi programma di registrazione
<https://www.tobias-erichsen.de/software/loopmidi.html>
- MTPDK: il plugin per simulare una batteria sul computer <https://www.powerdrumkit.com/> ;
- Tracktion: un programma di registrazione
<https://www.tracktion.com/> ;

Una volta scaricato tutto il necessario, seguire i seguenti passaggi:

1. Aprire loopMIDI
2. Creare una nuova porta come da immagine:



3. Aprire hairless-midiserial e collegare Arduino al pc
4. Selezionare come Porta Seriale la scheda Arduino mentre come Output MIDI la porta creata nel punto 2

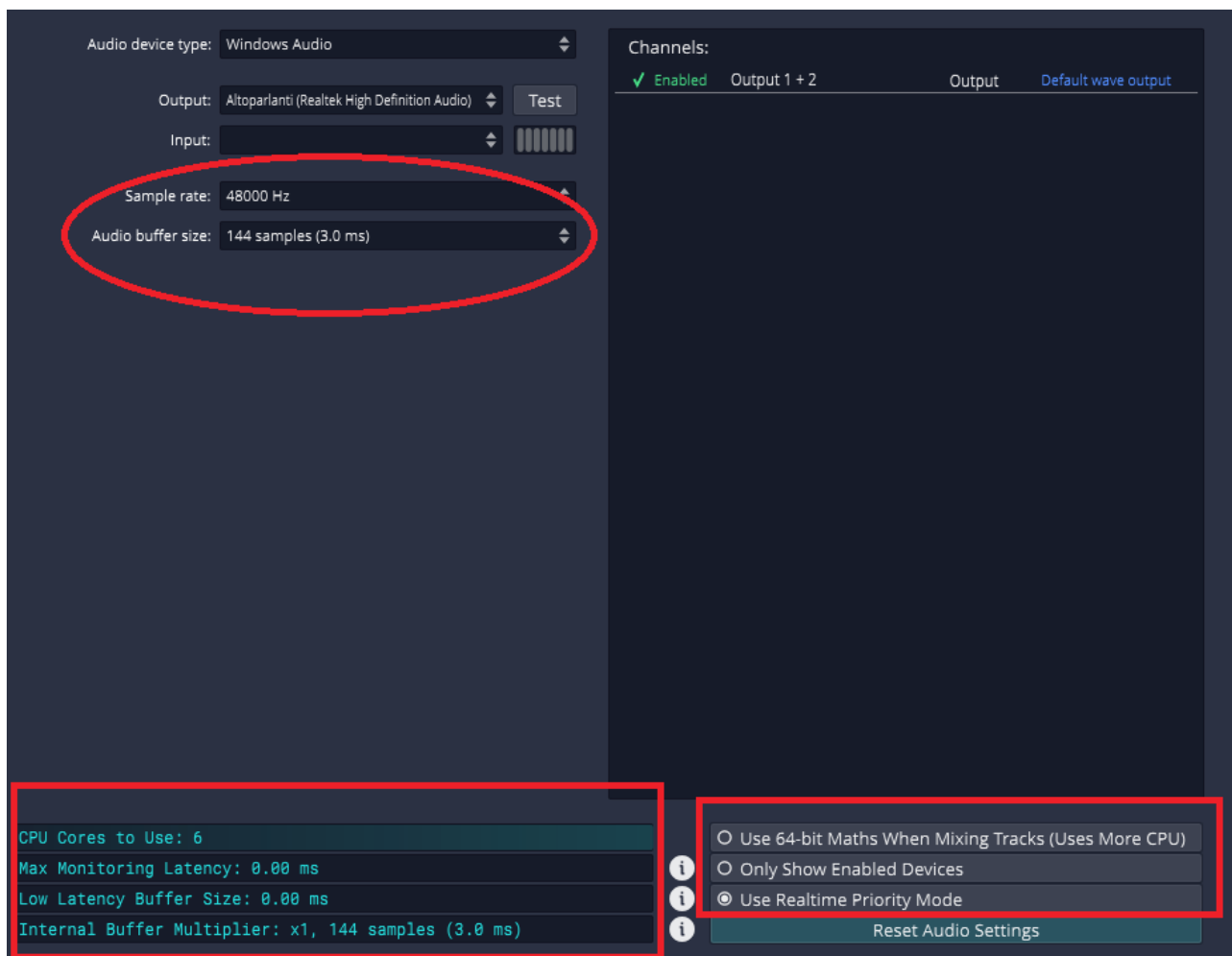


N.B! Nella cartella contenente lo sketch di Arduino è inizializzata la comunicazione seriale con il comando `Serial.begin(9600);` il numero contenuto all'interno della funzione identifica la velocità espressa in bps (baud per second) e deve coincidere con quella che si aspetta hairless-midiserial. E' necessario quindi, da quest'ultimo programma, navigare su File → Preferences e settare 9600 come baud rate. In realtà non è necessario che sia proprio 9600 (anche se è consigliabile): l'importante è che coincida quello dello sketch di Arduino con quello di hairless-midiserial.

5. Caricare lo sketch, contenuto nell'apposita cartella, sulla scheda ArduinoUNO (durante il caricamento deve essere tolta la spunta **SERIAL< - > MIDI Bridge On** da hairless-midiserial)
6. Aprire Tracktion
7. Estrarre dal file in formato .zip (MTPDK-2.0.4zip) "MT-PowerDrumKit.dll" e "MT-PowerDrumKit-Content.pdk"
8. In Tracktion, navigare in Settings → Plugins e qui trascinare nell'apposito spazio i due file estratto nel punto 7 (il plugin della batteria)
9. Poi Settings → MIDI Devices ed abilitare la porta creata nel punto 2



10. In Settings → Audio Devices è consigliabile utilizzare questo setup



11. Ora creare un nuovo progetto e seguire l'immagine



E' necessario trascinare il simbolo "+" nel punto 1 fino ad una qualsiasi delle "A+" dove ad esempio si trova il punto 2; si aprirà così un menù dal quale sarà possibile selezionare il plugin della batteria MTPowerDrum.

Bisogna poi schiacciare nella sezione Track della stessa riga scelta per il plugin e selezionare l'ormai amatissima porta del punto 2.

2.1) *Il protocollo MIDI*

MIDI è l'acronimo di Musical Instrument Digital Interface: è un "linguaggio" che permette di mettere in comunicazione tra loro computer, strumenti musicali, e altro hardware.

I messaggi MIDI sono brevi descrizioni numeriche di un'azione e possono essere interpretati da un programma per generare, ad esempio, suoni.

Si potrebbe, nel caso appena descritto, considerare un segnale MIDI come una nota che viene ricevuta da un programma target specifico il quale restituisce la stessa nota di partenza, ad esempio dalle casse.

Il potenziale che offre questo sistema è enorme soprattutto nel campo della musica elettronica.

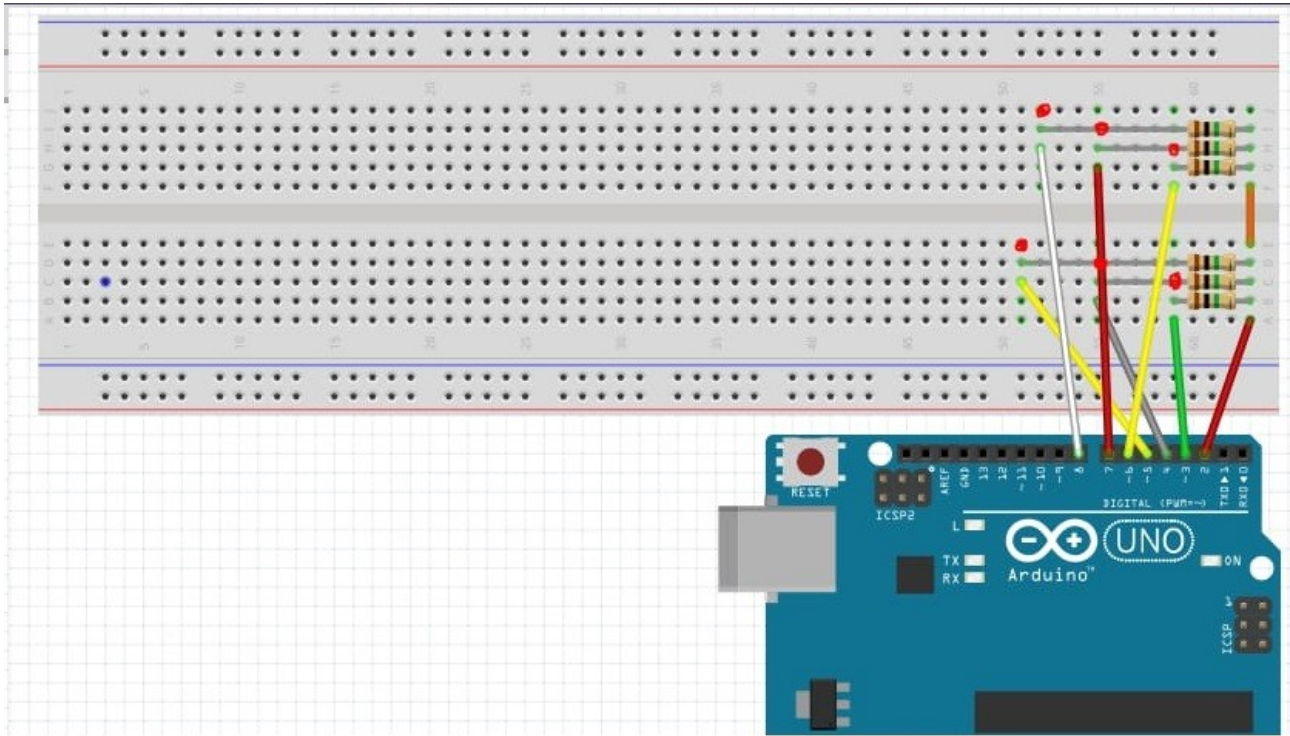
2.2) *Il sensore capacitivo*

La capacità misura la quantità di carica elettrica che un corpo è in grado di immagazzinare. La libreria Capacitive Sensor (di Paul Badger) che viene utilizzata nel programma (vedi argomento 2.4) controlla due piedini di Arduino, uno emettitore e l'altro ricevitore, e misura il tempo a loro richiesto per raggiungere lo stesso stato. Questi piedini sono collegati ad un oggetto metallico, come un foglio di alluminio nel nostro caso, che fa da sensore di tocco.

Avvicinandoci al nostro sensore il nostro corpo è in grado di

assorbire parte della carica, allungando i tempi necessari ai due piedini per raggiungere lo stesso stato.

2.3) Il Circuito



Il pin 2 è sempre emettitore mentre il 3,4,5,6,7 ed 8 sono ricevitori; i sensori sono da inserire in corrispondenza dei pallini rossi come da figura: ecco due esempi di sensori



2.4) Il Programma

```
#include <CapacitiveSensor.h>
```

```
CapacitiveSensor capSensor1 = CapacitiveSensor(2,3);  
CapacitiveSensor capSensor2 = CapacitiveSensor(2,4);  
CapacitiveSensor capSensor3 = CapacitiveSensor(2,5);  
CapacitiveSensor capSensor4 = CapacitiveSensor(2,6);  
CapacitiveSensor capSensor5 = CapacitiveSensor(2,7);  
CapacitiveSensor capSensor6 = CapacitiveSensor(2,8);
```

```
bool sound1Open = true;  
bool sound2Open = true;  
bool sound3Open = true;  
bool sound4Open = true;  
bool sound5Open = true;  
bool sound6Open = true;
```

```
int threshold = 100;
```

1. Includere la libreria CapacitiveSensor ;
2. Creare 6 oggetti CapacitiveSensor e passare al loro costruttore il numero del pin utilizzato come emettitore e il numero di quello utilizzato come ricevitore;
3. Dichiarare 6 variabili booleane inizializzate a true (una per ogni suono) ed una variabile intera che segnerà il limite di ritardo tra emissione e ricezione dei pin: se questa soglia viene superata, allora stiamo toccando il sensore.

```

void setup() {
  Serial.begin(9600);
}

void loop() {

  long sensorValue1 = capSensor1.capacitiveSensor(30);
  long sensorValue2 = capSensor2.capacitiveSensor(30);
  long sensorValue3 = capSensor3.capacitiveSensor(30);
  long sensorValue4 = capSensor4.capacitiveSensor(30);
  long sensorValue5 = capSensor5.capacitiveSensor(30);
  long sensorValue6 = capSensor6.capacitiveSensor(30);

  //primo suono 0x21 = Kick
  if (sensorValue1 > threshold && sound1Open == true){
    noteOn(0x90, 0x21, 0x70);
    sound1Open = false;
  }else if(sensorValue1 < threshold )
    sound1Open = true;

```

4. Nel setup, inizializzare la comunicazione seriale (Argomento 1) – punto 4);
5. Nel loop, creare una variabile di tipo long per ogni oggetto CapacitiveSensor precedentemente dichiarato: il valore della variabile è ottenuto dalla lettura del sensore. Leggere il sensore troppe volte può introdurre ritardi, farlo poco non è preciso. Controllare 30 campioni è un'ottima via di mezzo;

Ora analizzeremo quando e come viene inviato il segnale del primo suono (la grancassa della batteria) : il procedimento è uguale per gli altri 5, che non verranno quindi trattati.

6. Quando il valore del sensore ha superato la soglia da noi stabilita come ritardo limite, e quando la variabile booleana `sound1Open` è `true`, allora inviamo alla porta seriale un “suono” (non è ancora tale) per mezzo della funzione `noteOn` (punto 7). Una volta che il segnale è stato inviato, la variabile booleana passa a `false` e tornerà `true` solo quando il dito sarà rimosso dal sensore (e quindi il valore di `sensorValue1` sarà minore del ritardo limite);

```
void noteOn(int cmd, int pitch, int velocity)
{
    Serial.write(cmd);
    Serial.write(pitch);
    Serial.write(velocity);
}
```

7. La funzione `noteOn` accetta 3 parametri interi: il canale MIDI (tecnicamente sarebbe uno per strumento musicale ma a noi non importa) , la nota e l'intensità con il quale viene eseguita (cambia il volume); Queste informazioni vengono scritte sulla porta seriale e verranno tradotte da altri programmi in MIDI. Ogni nota corrisponde ad un numero esadecimale.
<https://www.wavosaur.com/download/midi-note-hex.php>

3) *Prototipo*



