
# PROGETTAZIONE DI UN ***CONTROLLER MIDI*** PER MEZZO DI ***SENSORI PIEZOELETTRICI*** INTERFACCIATI A UN MICROPROCESSORE ***ARDUINO DUE***.

*(progetto per il corso di **Tecniche Audiovisive**)*

*a cura di:*

***Gianluca Calabria***

***Claudio Mazzei***

# INDICE

# 1. INTRODUZIONE

Lo scopo di questo progetto è quello di realizzare un hardware che funzioni da controller MIDI. L'idea è quella di far suonare campioni audio di un determinato banco sonoro attraverso l'uso di uno o più piezoelettrici.

Per lo svolgimento del lavoro sono stati utilizzati: un microprocessore Arduino Due, un TinkerKit Sensor Shield V.2, una breadboard, un connettore MIDI out, un cavo USB-MIDI e due piezoelettrici con relative resistenze.

L'obiettivo primario del progetto è quello di convertire un segnale analogico, in questo caso una tensione proporzionale a una vibrazione, captato tramite un sensore piezoelettrico in un messaggio digitale che verrà inoltrato al connettore MIDI fornendone i dettagli necessari per la riproduzione di un qualsiasi effetto sonoro.

Prima di esporre la parte pratica analizzeremo i concetti fondamentali e la strumentazione utilizzata (e sopra citata) per la realizzazione del lavoro. Per prima cosa tratteremo una breve introduzione sul microprocessore Arduino Due, passando attraverso i concetti fondamentali del protocollo MIDI arriveremo a spiegare i passaggi caratterizzanti lo sviluppo del progetto. Verrà riportato per completezza l'intero codice utilizzato nel software di Arduino e verranno esplicate le possibili varie applicazioni attuabili con l'hardware progettato oltre ai potenziali sviluppi futuri.

# 2. IL PROTOCOLLO MIDI

Con l'acronimo MIDI (Musical Instrument Digital Interface) si indica il protocollo standard per l'interazione degli strumenti musicali elettronici, anche tramite un computer.

Il termine MIDI indica due cose: un linguaggio informatico, ossia una serie di specifiche che danno vita al protocollo, e un'interfaccia hardware, che consente il collegamento fisico tra dispositivi. Il MIDI è nato negli anni ottanta e, nonostante possa essere soppiantato da protocolli moderni dalle prestazioni superiori, è rimasto pressoché inalterato ed è tuttora intensamente utilizzato nella produzione di musica elettronica. I motivi risiedono probabilmente nel ruolo di standard pressoché incontrastato che esso ha assunto nell'ambito musicale e nella cura riposta dai progettisti nella stesura delle prime specifiche.Di fatto, il MIDI ha peculiarità interessanti su più fronti:

- la qualità e la praticità del sistema – l'integrazione tra eventi audio ed eventi MIDI ha dimostrato di essere una mossa vincente, confermando l'importanza di questo standard nella realizzazione di musica digitale.
- il peso dei file (nell'ordine dei kB) – tramite Internet e i software multimediali, il MIDI diventa un media di uso comune.
- la qualità delle basi.

- il costo – molti produttori hardware e software puntano sulla multimedialità dei propri prodotti. Con un investimento minimo o ricorrendo a programmi freeware, è possibile a chiunque disporre di un computer in grado di realizzare produzioni musicali di buon livello.

Lo standard MIDI consiste in un protocollo per lo scambio di messaggi progettato per l'uso con strumenti musicali elettronici e della relativa interfaccia fisica. Un collegamento MIDI consiste in una connessione seriale unidirezionale (simplex) a loop di corrente, che funziona a una velocità di trasmissione di 31,250 bps. Il loop è tenuto a massa solo da un lato, mentre l'altra estremità è libera (flottante) per evitare ronzii e interferenze audio indotte dalla formazione di anelli di massa. Due dispositivi collegati con interfaccia MIDI sono tra di loro optoisolati, cioè il loop di corrente dal lato del trasmettitore pilota il LED di un accoppiatore ottico dal lato del ricevitore. L'accoppiatore ottico dev'essere logicamente molto veloce (un dispositivo molto usato è lo Sharp PC900) e, comunque, i tempi di commutazione asimmetrici che caratterizzano la maggior parte degli accoppiatori sono in ogni caso causa di distorsioni. Nel caso di connessione in cascata di più dispositivi MIDI, poi, la distorsione e il ritardo del segnale si fa via via più rilevante fino a generare errori di trasmissione.

Lo standard MIDI prevede l'uso di connettori circolari a standard DIN a 5 pin. Questi connettori erano molto diffusi in Europa per ogni genere di connessione audio fino agli anni novanta, quando sono stati sostituiti dai più pratici jack. Anche le comunissime tastiere dei personal computer in passato utilizzavano quel connettore, poi l'hanno abbandonato per il mini-DIN. Il MIDI è rimasto quindi una delle poche applicazioni per questo genere di connettori. Esiste anche una versione dello standard che sfrutta il collegamento USB ed è attualmente in sviluppo, da parte della IETF, una versione per il trasporto di segnali MIDI su Ethernet e Internet.

## 2.1 I MESSAGGI MIDI

Il linguaggio MIDI è usato per trasmettere informazioni in tempo reale per la riproduzione di un brano musicale. "Tempo reale" significa che ogni messaggio viene inviato esattamente nel momento in cui deve essere interpretato dal sintetizzatore di destinazione (che può essere un sintetizzatore hardware o software).

Vengono definiti vari messaggi per trasmettere le informazioni necessarie per eseguire la riproduzione della musica. Il punto fondamentale è che il linguaggio MIDI non definisce il suono stesso, ma solo la sequenza di istruzioni per creare il suono nel sintetizzatore di destinazione. I messaggi MIDI vengono inviati come una sequenza temporale di uno o più byte (8 bit). Il primo byte è uno Status Byte, spesso seguito da un Data Byte con parametri aggiuntivi. Uno Status Byte ha il settimo bit impostato a 1 e il Data Byte ha il settimo bit impostato a 0.

Lo Status Byte determina il tipo di messaggio. Il numero di Data Byte che seguono dipendono dal tipo di messaggio. Fatta eccezione per alcuni messaggi MIDI di sistema, lo Status Byte contiene il numero del canale MIDI. Ci sono 16 possibile canali MIDI, numerati da 0 a 15 in esadecimale. In

pratica, i musicisti e il software si riferiscono ai canali MIDI contando i canali da 1 a 16, in modo che ci sia una differenza di 1 quando li si programma in esadecimale (il canale ‘1’ è codificato come ‘0’, il canale ’10’ è codificato come ‘9’ e il canale ’16’ è codificato come ‘F’). Nello stesso cavo MIDI possono essere usati fino a 16 canali MIDI per controllare fino a 16 diversi strumenti che suonano in modo indipendente.

Durante la lettura dei byte provenienti da un messaggio MIDI, lo Status Byte può essere omesso (fatta eccezione per il primo messaggio di quel tipo). In tal caso è possibile ricevere un messaggio che ha solo Data Byte. Lo Status Byte dovrebbe essere lo stesso come l’ultimo Status Byte ricevuto. Questo si chiama MIDI Running Status. È utile ad esempio per ottimizzare la trasmissione quando viene inviata una lunga serie di stessi messaggi. Un esempio potrebbe essere un Pitch Bend o una curva di volume in crescendo.

## 2.2 I MESSAGGI NOTE

I messaggi principali sono quelli di Note On e Note Off. Il messaggio di Note On viene inviato quando l’esecutore preme un tasto della tastiera musicale; tale tasto contiene i parametri per specificare l’altezza della nota, così come la sua velocity (ovvero l’intensità con la quale viene premuto il tasto). Quando un sintetizzatore riceve questo messaggio inizia a suonare la rispettiva nota con la corretta intonazione e la giusta intensità. Quando invece viene ricevuto il messaggio di Note Off, il sintetizzatore cessa di suonare la nota corrispondente.

Ogni messaggio di Note On richiede il corrispondente messaggio di Note Off, altrimenti la nota suonerà senza mai fermarsi. L’unica eccezione è fatta per gli strumenti a percussione per i quali può accadere che venga inviato solamente il messaggio di Note On poiché la nota, essendo di natura quasi impulsiva, cesserà di suonare da sola. E’ sempre preferibile, in ogni caso, inviare anche il messaggio di Note Off, perché non si sa come potrebbe essere interpretato il solo messaggio di Note On dal sintetizzatore che lo riceve. Il messaggio di Note On è strutturato come segue:

Status Byte: 1001 CCCC
Data Byte 1: 0PPP PPPP
Data Byte 2: 0VVV VVVV

dove:

‘CCCC’ è il canale MIDI (da 0 a 15)
‘PPP PPPP’ è il valore del pitch (intonazione) (da 0 a 127)
‘VVV VVVV’ è il valore della velocity (intensità) (da 0 a 127)

Il valore del pitch determina la frequenza della nota da suonare. Si va da 0 a 127, con il Do (C) centrale rappresentato dal valore 60. Il valore è rappresentato in semitoni, in modo che il Do# sarà 61, il Re sarà 62, ecc. Per trasporre una nota un’ottava più alta basterà aggiungere 12 al suo valore di intonazione. Utilizzando il MIDI, la trasposizione è molto semplice in quanto è fatta semplicemente

aggiungendo o sottraendo un valore fisso. Bisogna essere cauti però circa l'intervallo di note MIDI che va da 0 a 127. Aggiungendo ad esempio 4 ottave (+48) per un valore di nota 96, il totale è 144, che è al di fuori della gamma e può essere troncato a 16 (144-128) e finirà per suonerà una nota molto bassa. Il valore di velocity normalmente va da 1 a 127, che copre l'intervallo da una nota praticamente impercettibile fino ad un livello massimo.

Nei sintetizzatori più semplici il valore della velocity viene usato solo per determinare la forza con la quale la nota viene suonata, l'unico effetto è una nota che è più forte o più debole di volume. Nei sintetizzatori più sofisticati questo valore influisce anche sulla qualità del suono. Infatti, su un pianoforte vero, il colpire una nota in maniera forte interesserà non solo il suo volume ma anche la qualità del suono stesso, il timbro. Questo è il caso di ogni strumento reale. Vi è un caso particolare se la velocity è impostata a zero. Il messaggio di Note On assume lo stesso significato del messaggio di Note Off, mutando la nota.

Il messaggio di Note Off è strutturato come segue:

Status Byte: 1000 CCCC
Data Byte 1: 0PPP PPPP
Data Byte 2: 0VVV VVVV

dove CCCC e PPP PPPP hanno lo stesso significato detto sopra. Il VVV VVVV è la velocità di rilascio (release), che viene usata molto raramente e di default è impostata su '0'.

## 2.3 SUONARE NOTE E ACCORDI

Quando si invia un messaggio di Note On a un sintetizzatore, questa nota inizia a suonare. Nel frattempo, è possibile inviare altri messaggi di Note On, con diverse intonazioni, in modo da poter sentire un accordo. Tuttavia è necessario tenere traccia delle note che stanno suonando, in modo da poter inviare un messaggio corrispondente di Note Off per ogni nota, altrimenti ci sarebbero note bloccate che suonerebbero per sempre.

## 2.4 USARE I CANALI MIDI

Il protocollo MIDI gestisce fino a 16 canali MIDI diversi. Ogni canale ha il suo stato, ad esempio lo strumento corrente definito, le note che sono attualmente in riproduzione, così come altri valori quali il volume, la panoramica, ecc. Utilizzando diversi canali MIDI è possibile definire uno strumento specifico per ciascuno di essi. Inviando note sui corrispondenti canali MIDI queste note suoneranno con gli strumenti forniti.

## 2.5 CONTROLLER MIDI

Ci sono 128 controller MIDI definiti, ma solo pochi di essi vengono utilizzati nella pratica. Lo scopo di un controller MIDI è quello di impostare un valore di un parametro nel sintetizzatore che sta

suonando le note, come il volume, la panoramica (posizione stereo, da sinistra a destra), il livello di riverbero, ecc. Il messaggio è costruito come segue:

Status Byte: 1011 CCCC
Data Byte 1: 0NNN NNNN
Data Byte 2: 0VVV VVVV

dove CCCC è il canale MIDI, NNN NNNN rappresenta il numero di controllo (da 0 a 127) e VVV VVVV è il valore assegnato al controller (anch'esso da 0 a 127). I numeri più comuni dei controller sono i seguenti:

0 = selezione della Sound Bank (MSB);
1 = modulationwheel, spesso assegnata ad un effetto vibrato o tremolo;
7 = livello di volume dello strumento;
10 = panoramica (0 = sinistra; 64 = centro; 127 = destra);
11 = expression (talvolta utilizzato anche per il controllo del volume o simili, a seconda del sintetizzatore);
32 = selezione della Sound Bank (LSB)
64 = pedale di sustain (0 = nessun pedale; se >= 64 allora il pedale è attivo);
121 = tutti i controlli diventano off (questo messaggio cancella tutti i valori di regolazione per questo canale, si torna ai loro valori di default)
123 = tutte le note diventano off (questo messaggio interrompe tutte le note che stanno correntemente suonando).

## 2.5.1 VOLUME E VELOCITY

Ad esempio, se si desidera impostare il volume su un valore di 100 (nell'intervallo di valori 0-127, 100 = 0x64) per lo strumento che suona sul canale 1 (codice 0), è possibile inviare il seguente messaggio:

0xB0 - 0x07 - 0x64

La variazione di volume ha un impatto su tutte le note che stanno correntemente suonando così come le note che inizieranno a suonare dopo. Il sintetizzatore mantiene questo livello di volume fino a quando viene inviato un altro livello di volume. Va ricordato che la velocity di una nota viene inviata con il messaggio di Note On stesso. La velocity non può essere cambiata una volta che la nota è stata avviata, ma è possibile utilizzare il controller del volume per cambiare il livello della nota in seguito alla sua esecuzione. Per creare un crescendo è necessario inviare una sequenza di valori crescenti di volume. È' sempre necessario bilanciare i valori di velocity e di volume da usare in modo che entrambi siano in un corretto intervallo di valori. Se uno dei due valori è troppo basso le note non possono essere sentite correttamente, anche se uno dei due è al suo massimo. L'effetto di entrambi i valori è moltiplicativo per determinare il vero volume della nota.

## 2.6 RESET DELLE NOTE

Come abbiamo visto, Il messaggio di Note On deve avere il suo corrispondente messaggio di Note Off, altrimenti la nota continuerà a suonare per sempre. Ci sono casi in cui si desidera ripristinare qualunque nota stia suonando. Ci sono fondamentalmente quattro modi per farlo. Alcuni sintetizzatori non le accettano tutte, però è interessante esporre le diverse possibilità:

1) Utilizzo del controller MIDI 123: se invia un controller MIDI 123 su un canale MIDI, il sintetizzatore fermerà tutte le note che stanno suonando su quel canale. Per ripristinare tutti i canali MIDI basterà inviare il messaggio per ciascun canale. Si prega di notare che alcuni sintetizzatori non rispondono a questo messaggio;

2) Messaggio di reset MIDI: questo è un messaggio di Status byte 0xFF, senza Data byte. Questo messaggio dovrebbe ripristinare il sintetizzatore per la sua impostazione predefinita all'accensione, quindi ferma anche tutte le note che stanno suonando. Bisogna utilizzare questo messaggio con parsimonia perché ripristina l'intero sintetizzatore, non solo le note che stanno suonando;

3) MIDI Note Off: è possibile inviare per ogni canale (da 0 a 15) e per ogni pitch di nota (da 0 a 127) un messaggio MIDI di Note Off. Questa è la soluzione totale ma richiede molti messaggi MIDI da inviare e può richiedere un certo tempo di reazione da parte del sistemo hardware MIDI utilizzato;

4) MIDI Note Off (ottimizzato): in questo caso si utilizza una tabella per tenere traccia dei messaggi di Note On e di Note Off che si inviano per ogni canale. Un buffer di 128 byte per ciascun canale, che rappresenta il numero di messaggi di Note On inviati a quella nota, potrebbe essere incrementato con un Note On o diminuito con un Note Off. Poi, quando si desidera ripristinare ogni nota, è sufficiente passare attraverso quella tabella e inviare un Note Off per ogni nota che sta ancora suonando.

# 3. ARDUINO

Arduino è una scheda elettronica di piccole dimensioni con un microcontrollore e circuiteria di contorno, utile per creare rapidamente prototipi e per scopi hobbistici e didattici.Il nome della scheda deriva da quello di un bar di Ivrea (che richiama a sua volta il nome di Arduino d'Ivrea, Re d'Italia nel 1002) frequentato da alcuni dei fondatori del progetto. Con Arduino si possono realizzare in maniera relativamente rapida e semplice piccoli dispositivi come controllori di luci, di velocità per motori, sensori di luce, temperatura e umidità e molti altri progetti che utilizzano sensori, attuatori e comunicazione con altri dispositivi. E' fornito di un semplice ambiente di sviluppo integrato per la programmazione. Tutto il software a corredo è libero, e gli schemi circuitali sono distribuiti come hardware libero.

## 3.1 ARDUINO DUE

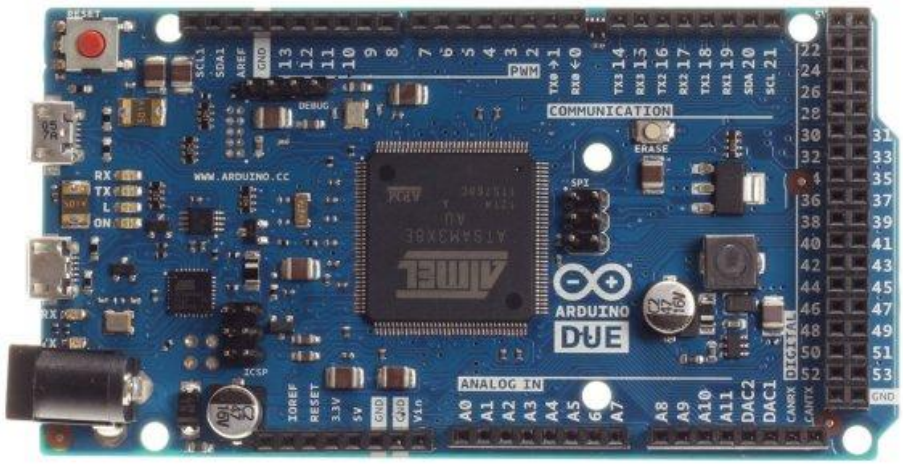
La scheda Arduino Due è basata sul microcontrollore Atmel SAM3X8E ARM Cortex-M3. Arduino Due è la prima scheda Arduino a 32 bit, è fornita di 54 piedini I/O digitali (di cui 12 utilizzati come uscite PWM), 12 input analogici, 4 UARTs (porte seriali hardware), ha un clock a 84 MHz, 2 DAC (digital/analog), 2 TWI (per la comunicazione I2C o TwoWire), jack di alimentazione da 2,1 mm con positivo centrale, una JTAG per la programmazione diretta del microcontrollore e per il debug, un bottone di reset ed uno di cancellazione.

A differenza di altre schede Arduino, la scheda Arduino Due funziona a 3.3V. La tensione massima che i pin I /O sono in grado di tollerare è di 3,3V. Fornire tensioni più elevate, come 5V a un pin I /O potrebbe danneggiare la scheda.Per poter programmare la scheda è sufficiente connetterla alla micro USB di programmazione. L'alimentazione viene fornita direttamente dalla USB oppure mediante un alimentatore esterno. Arduino Due è compatibile con tutti gli shield Arduino che funzionano a 3,3 V e compatibili con il pinout di Arduino Uno.

E' disponibile una porta USB nativa da usare quando si desidera usare Arduino Due come una qualsiasi periferica USB (così come avviene per un mouse o una tastiera) oppure quando si vuole usare la scheda come host in modo che altri dispositivi si possano collegare ad essa (mouse, tastiere o telefoni Android) per connettere mediante micro USB altri dispositivi esterni. Quando la scheda è utilizzata come host USB è indispensabile alimentarla tramite connettore di alimentazione. La USB nativa può essere usata come porta seriale virtuale usando l'oggetto "SerialUSB" nel linguaggio di programmazione di Arduino.

Caratteristiche tecniche:

- Microcontroller: AT91SAM3X8E
- Tensione di funzionamento: 3.3V
- Tensione di ingresso (raccomandata): 7-12V
- Tensione di ingresso (limiti): 6-20V
- Pin I/O digitali: 54 (di cui 12 utilizzati come uscite PWM)
- Pin di ingresso analogico: 12
- Uscite analogiche: 2 (DAC)
- Corrente totale sulle linee I/O: 130 mA
- Corrente sul Pin 3,3V: 800 mA
- Corrente sul Pin 5V: 800 mA
- Memoria Flash: 512 KB, totalmente disponibili per le applicazioni dell'utente
- SRAM: 96 KB (due banchi: 64 KB e 32 KB)
- Velocità di clock: 84 MHz




## 3.2 CONFIGURAZIONE HARDWARE

Elencheremo di seguito le componenti necessarie a completare l'hardware trattato:

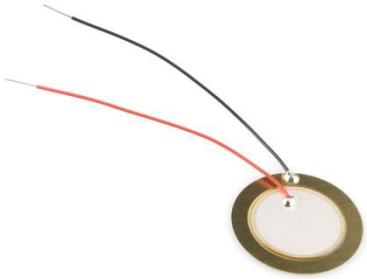
1) Un MIDI Jack: ha 5 pin di cui ne utilizzeremo solamente 3.


2) Una resistenza da 220 Ohm; la resistenza limiterà la tensione che arriva dall'uscita della porta di Arduino.

3) Cavo USB-MIDI: avremo bisogno di questo cavo per testare il controller. Difatti tramite questo

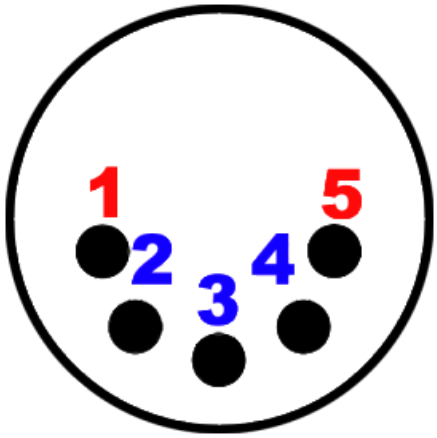
cavo collegheremo il controller all'entrata USB del computer, utilizzando plug-in audio interni al calcolatore sotto forma di software. Se volessimo attaccare il nostro apparecchio ad una tastiera che possiede dei banchi sonori basterà utilizzare un cavo midi maschio-maschio.

4) Due sensori piezoelettrici e relativa resistenza da 1 MOhm.



5) Cavi per i connettori:

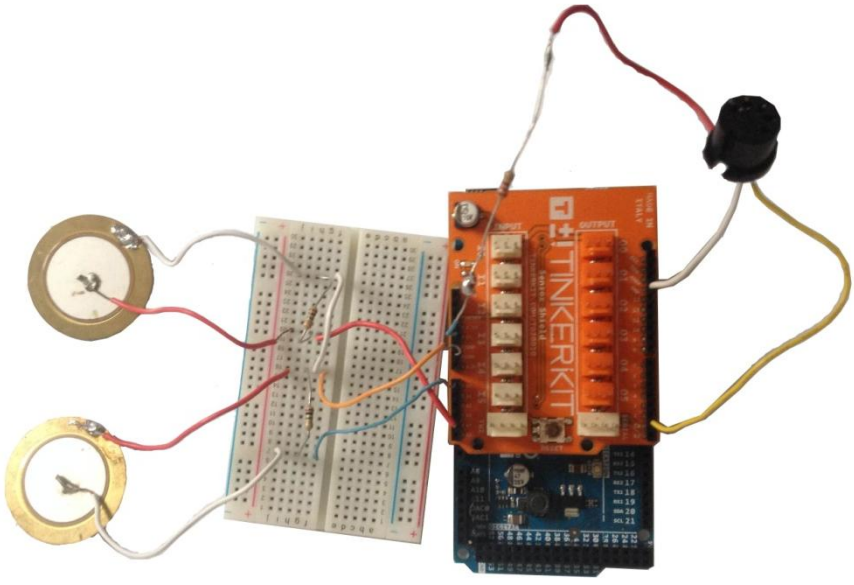
## 3.2.1 EFFETTUARE I COLLEGAMENTI




Verranno utilizzati solamente i connettori 2, 3 e 4. I connettori 1 e 5 non vengono quasi mai utilizzati nelle applicazioni.

Connettore 2: è la sincronizzazione della corrente;
Connettore 3: è la schermatura;
Connettore 4: è il generatore di corrente.

Si può pensare il connettore 4 come quello positivo, il 3 come la massa e il 2 come l'input dei dati.




Come si può vedere il connettore 4 va ai 5V positivi e il connettore 3 va al ground. Il connettore 2 va al serial 1 (TX1).

## 3.3 IL CODICE

Descriveremo di seguito il codice utilizzato per sviluppare il progetto:

```
int E3=0x40;
int G3=0x43;
int piezo=A0;
int piezo2= A4;
int velocity;
int velocity2;

byte valpiezo=0;
byte valpiezo2=0;

int THRESHOLD = 150;
```

La prima parte del codice riguarda l'inizializzazione delle variabili. In particolare sono state definite due variabili che rappresenteranno le due note che verranno successivamente assegnate ai piezoelettrici (in questo caso sono state generate un E, o Mi, alla terza ottava e un G, o Sol, anch'esso alla terza ottava). Il valore delle note è espresso tramite valore esadecimale e corrisponde alla sua intonazione (o pitch). La terza e la quarta variabile rappresentano gli input fisici ai quali sono collegati i due piezoelettrici; la quinta e la sesta rappresentano le variabili velocity che saranno proporzionali all'intensità della pressione esercitata sul sensore piezoelettrico e verranno memorizzate all'interno della settima e dell'ottava variabile. Il threshold, infine, rappresenta la soglia al di sopra della quale Arduino permetterà il rilevamento della pressione ed è stata settata opportunamente.

```
void setup() {
  //  Set MIDI baud rate:
  Serial.begin(31250);
}
```

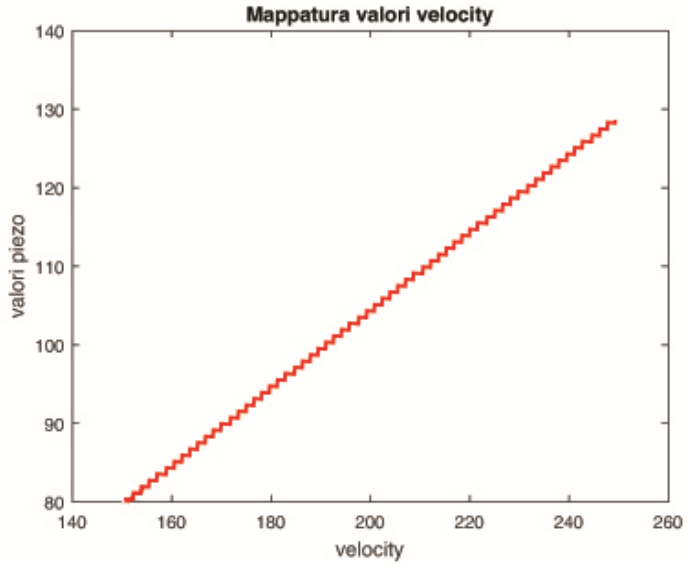
La funzione setup() è la prima ad essere chiamata quando parte il codice. Viene utilizzata per inizializzare le variabili, per impostare lo stato dei pin, per far partire le librerie da usare, per l'impostazione delle comunicazioni seriali. Nel nostro caso è stata impostata la velocità di comunicazione seriale con il computer a 31250 bit per secondo (baud) essendo il MIDI un protocollo seriale che richiede quel rate.

```
void loop() {

    valpiezo=analogRead(piezo);
    delayMicroseconds(10);
    valpiezo2=analogRead(piezo2);
    delayMicroseconds(10);
   if(valpiezo2 >= THRESHOLD &&  valpiezo >= THRESHOLD)
 {
        velocity=map(valpiezo,150,255,80,127);
        velocity2=map(valpiezo2,150,255,80,127);
        noteOn(0x90,E3,velocity);
        noteOn(0x90,G3,velocity2);
      delay(50);
      noteOn(0x90,E3,0x00);
      noteOn(0x90,G3,0x00);
      delay(50);
 }
```

```
  else if (valpiezo2 >= THRESHOLD && valpiezo<THRESHOLD) {
    velocity2=map(valpiezo2,150,255,80,127);

    noteOn(0x90,G3,velocity2);
    delay(50);
    noteOn(0x90,G3,0x00);
    delay(50);
  }

 else if (valpiezo >= THRESHOLD && valpiezo2<THRESHOLD) {
    velocity=map(valpiezo,150,255,80,127);

    noteOn(0x90,E3,velocity);
    delay(50);
    noteOn(0x90,E3,0x00);
    delay(50);
  }
}
```

In questa sezione Arduino svolge le operazioni principali. Innanzitutto, tramite la funzione analogRead, legge eventuali vibrazioni captate dal sensore piezoelettrico. È importante sottolineare una limitazione di Arduino che non permette di leggere due segnali analogici, provenienti dai due sensori distinti, nello istante temporale. Questo problema è stato risolto imponendo un leggero ritardo (o delay) tra i due analogRead.

In base ai differenti eventi che si possono verificare (pressione di un unico piezo o pressione di entrambi contemporaneamente), Arduino sceglie uno dei tre cicli imposti dal codice. Ognuno di questi cicli effettuerà una ri-mappatura del valore analogico letto dal piezo e lo distribuirà nell'intervallo tra 80 e 127 (valori di velocity, ovvero di intensità della nota).



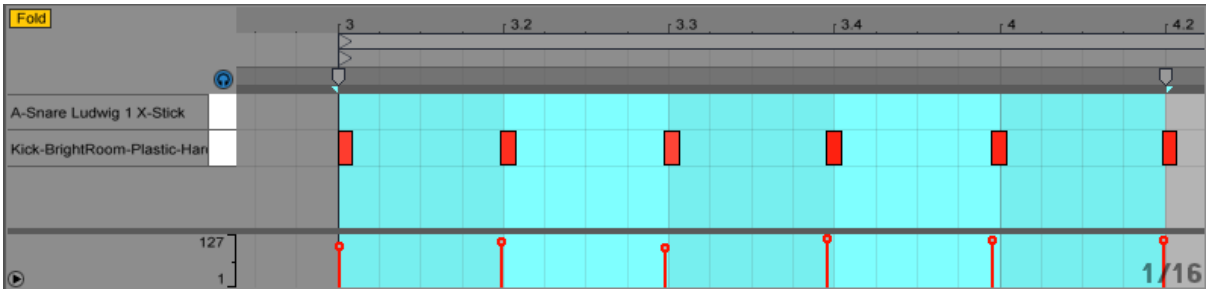

Successivamente verrà richiamata la funzione noteOn la quale richiederà in ingresso i tre valori tipici richiesti dal protocollo MIDI per la trasmissione di informazione: canale, nota, velocity. Nel nostro caso verrà utilizzato esclusivamente il canale 1. Il secondo valore sarà identico al pitch assegnato al relativo piezo che è stato percosso. Il terzo valore corrisponde all'intensità impressa sul sensore. Il messaggio di noteOff è espresso da un messaggio di noteOn avente velocity 0x00, ovvero nulla.

La funziona noteOn è descritta dal seguente blocco di codice:

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

}
```

# 4. CONCLUSIONI

Il progetto sviluppato risulta essere molto robusto e con valori di latenza decisamente accettabili essendo inferiore ai 10 ms (che è un valore di ritardo non percepibile dall'orecchio umano). Tuttavia per quanto riguarda la pressione simultanea dei sensori piezoelettrici, Arduino non risulta un'efficace soluzione poichè ciò richiederebbe una gestione avanzata dei meccanismi di interrupt. Sono stati effettuati dei test sulla latenza con una Digital Audio Workstation (nel nostro caso Ableton Live 9 Suite). I risultati mostrano che l'unica imprecisione rilevabile è dovuta all'errore umano (infatti i colpi risultano spesso anticipati rispetto al battito del metronomo), ciò permette di assumere il controller MIDI a latenza zero.




È possibile aumentare il numero di sensori piezoelettrici pur rientrando in spese contenute (un piezoelettrico si può trovare in qualsiasi negozio di elettronica a meno di due euro).
Tuttavia per ottenere valori di dinamica più precisi converrebbe acquistare dei sensori piezoelettrici più sensibili e performanti. Ciononostante anche i piezo a nostra disposizione ci consentono livelli di sensibilità tali da poter essere applicarti su qualsiasi superficie facendo diventare la superficie stessa un materiale risonante che provocherebbe vibrazioni rivelabili dal sensore. Per questo motivo il progetto sviluppato è ampliabile a svariati ambiti e non solo a quello musicale. Basti pensare ad un possibile collegamento wireless fra sensore piezoelettrico e microprocessore Arduino. Questo permetterebbe ad esempio l'interazione con un computer e la possibilità di gestire l'apparecchio come fosse un telecomando per presentazioni o per filmati. Potrebbe anche essere applicato ad un sistema di illuminazione per gestire in remoto le dinamiche di luminosità in un determinato ambiente.Con l'equivalente di 40 euro si sono poste le basi per la progettazione di un sistema sensibile alla dinamica dalle molteplici applicazioni nell'ambito multimediale.