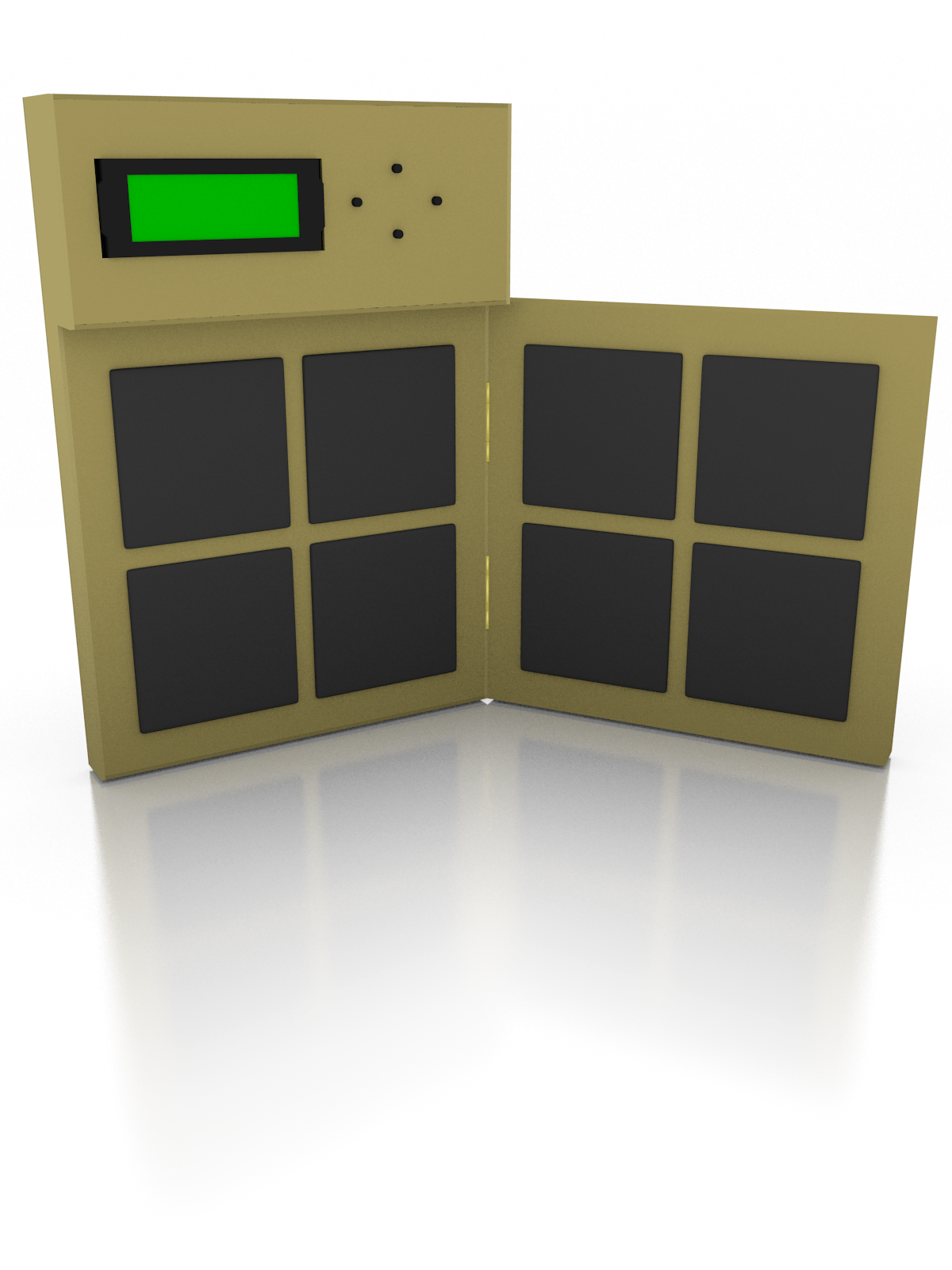
**Arduino Drum Machine**

**Batteria elettronica programmabile open source**



Progetto per l’**Esame di Stato** a.s. 2011-12

Candidato: Riccardo Miccini

Classe: 5°A Informatica

INDICE

1. Presentazione del progetto
2. Gestione dei pads
   1. Regolazione del segnale
      1. Trasduttore piezoelettrico
      2. Diodo di Zener
   2. Multiplexing degli ingressi
      1. Circuito integrato: 4051 Analog Mux/Demux
3. Gestione del display
4. Gestione dei bottoni
5. MIDI output
6. Audio output
   1. Filtraggio e regolazione
      1. Amplificatore oprazionale in modalità buffer
7. Studio e montaggio dell’hardware
8. Approfondimenti
   1. Arduino
   2. Musical Instrument Digital Interface (MIDI)
   3. Linguaggio C++/Wiring
9. English Corner
   1. What is Arduino?
   2. What is MIDI?

# Presentazione del progetto

Arduino è una piattaforma di prototipazione intuitiva e facile da utilizzare, che attira sempre più hobbisti e “makers”. Per questo motivo si è deciso di sfruttare la sua architettura per la costruzione di una batteria elettronica completamente programmabile e configurabile. Nonostante si trovino molteplici dispositivi simili, sia “fai-da-te” che professionali, raramente essi permettono una completa personalizzazione o l’integrazione di funzioni aggiuntive non previste.

Arduino invece, consente un’elevata versatilità, che rende il progetto continuamente aggiornabile e migliorabile. Il dispositivo si compone di 8 pad sensibili al tocco, collegati attraverso un multiplexer a un ingresso analogico. Tramite display e pulsanti di controllo è possibile navigare tra i menù e scegliere una delle 3 diverse modalità: drum (per la simulazione di batteria elettronica attraverso MIDI), piano (per l’output tramite sintesi sonora), e midi (per l’invio di dati MIDI). Ogni modalità consente la modifica di una vasta gamma di parametri (canale MIDI, tonalità, banco sonoro..), sempre accessibili tramite display.

Nel corso dello sviluppo del progetto, si sono riscontrate numerose difficoltà, soprattutto per quanto riguarda la gestione dei pad piezoelettrici e la sintesi di suoni con diverse forme d’onda. Nel primo caso, si sono riscontrati dei cali di tensione verso Arduino ogniqualvolta venivano collegate le schede di condizionamento del segnale dei sensori, probabilmente a causa della maggiore richiesta di corrente dei dispositivi. Temendo un danneggiamento della scheda Arduino, non si è provveduto in ulteriori verifiche, preferendo adottare dei comuni bottoni monostabili (tactile switch) montati su una scheda di prototipazione, a soli fini dimostrativi. Nonostante ciò, le schede sono state testate con risultati più che soddisfacenti, e il codice implementato permette la loro completa gestione.

L’altro problema, riguardante la sintesi sonora polifonica e multi timbrica, richiedeva elevate conoscenze delle istruzioni assembly del microcontrollore, che non si è avuto modo di approfondire. Si è quindi deciso di adottare alcune delle librerie messe a disposizione degli sviluppatori di Arduino, tra cui una funzione che permette la generazione di onde quadre a una data frequenza. Parte del codice riguardante la creazione di wavetable e la sintesi sonora è però stato implementato e scritto.

Un altro aspetto che non è stato curato riguarda la creazione del supporto fisico contenente le varie schede, non strettamente fondamentale ai fini della presentazione del progetto. Per la sua realizzazione ci si è avvalsi dell’aiuto del prof. Roberto Gatta, che ha fornito idee e supporto nell’assemblaggio del contenitore, volutamente lasciato incompiuto per via delle problematiche riportate sopra.

Il progetto in sé vuole essere una dimostrazione di come la piattaforma Arduino possa essere impiegata in una vasta gamma di applicazioni, permettendo al programmatore (nonché utente) di implementare funzionalità spesso non presenti nelle versioni commerciali di prodotti analoghi. Inoltre, lo sviluppo della Arduino Drum Machine è stato una grande fonte di esperienza, che mi ha permesso di comprendere le fasi della progettazione di circuiti stampati e il workflow (flusso di lavoro) necessario per la realizzazione di un dispositivo. Ho inoltre avuto modo di affacciarmi al mondo dell’embedded programming, ovvero la programmazione di microcontrollori o altri sistemi integrati.

***Arduino Drum Machine*** *è un progetto di* ***Riccardo Miccini*** *e* ***Davide Montini****, seguito con particolare attenzione dall’assistente tecnico* ***Mauro Acangeli****. Il codice sorgente è prelevabile online presso il forum di Arduino, sotto licenza* ***GNU GPL v3****. Questa documentazione, nonché tutti gli schemi circuitali ed i circuiti stampati sono invece coperti da licenza* ***Creative Common BY-NC-ND v3****.*

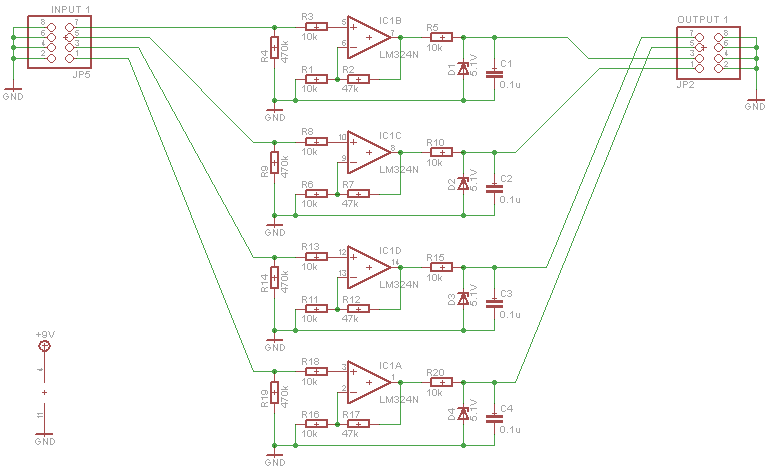
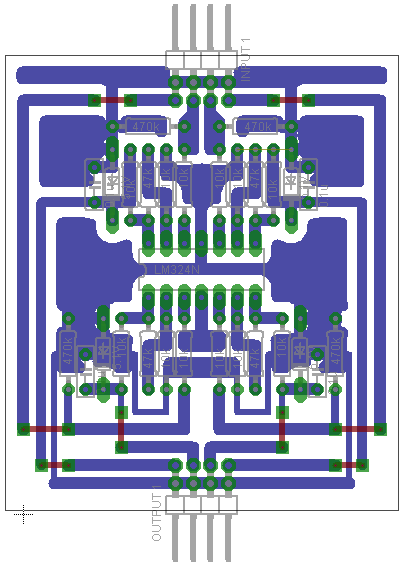
# Gestione dei pads

Una delle parti fondamentali, nonché uno dei primi problemi nel quale si è incappati nel corso della progettazione, è la gestione dei pad, ovvero i cuscinetti semirigidi che verranno suonati. Arduino infatti possiede un numero limitato di ingressi analogici (6 pin), e sfruttarli tutti per la ricezione dei pad non sembra una buona scelta né da un punto di vista progettuale (vista l’impossibilità di sfruttarli in seguito per eventuali upgrade) né pratico (6 pad non sono sufficienti per una drum machine). Si è quindi deciso di convogliare i segnali provenienti dai pad in un multiplexer, occupando solo 4 pin, di cui 3 I/O digitali e uno ingresso analogico. Inoltre, onde evitare escursioni di tensione pericolose o ingestibili, è stato sviluppato un apposito circuito.

## Regolazione del segnale

Nell’ambito del “fai-da-te” i sensori piezoelettrici sono ampiamente usati per rilevare svariati tipi di sollecitazioni meccaniche. Il segnale da essi prodotto però, a causa delle sue caratteristiche di instabilità ed incompatibilità con il range supportato dagli ADC, necessita di un’opportuna elaborazione al fine di renderlo fruibile da Arduino. È stato quindi ideato il seguente schema logico, che raccoglie tutte le fasi del processo e le relative scelte implementative:

Dallo schema logico è stato infine modellato lo schema circuitale, da cui si è ottenuto il seguente circuito stampato:

### Trasduttore piezoelettrico

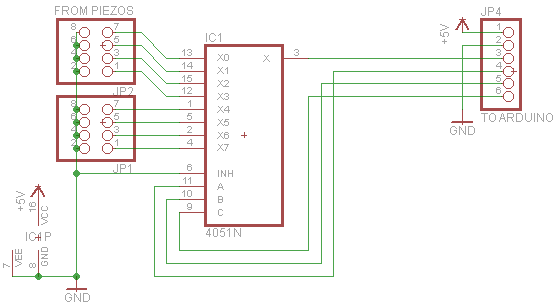
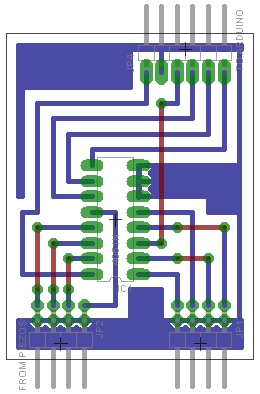
L’effetto piezoelettrico è un fenomeno scoperto da P. Curie e caratterizza alcuni tipi di cristalli (quarzo, tormalina, galena) e ceramiche, che sprigionano una differenza di potenziale quando sottoposti a forze meccaniche. I trasduttori piezoelettrici sono dei piccoli dischi metallici separati una speciale pellicola piezoelettrica che permette la generazione di una tensione proporzionale allo stimolo ricevuto. Le applicazioni di questi elementi sono centinaia: accendigas da cucina, rilevatori sismici, rudimentali altoparlanti, orologi, microfoni, segnalatori acustici, solo per citarne alcune. L’alta temperatura di lavoro (oltre 1000°C per gli elementi al fosfato di gallio), l’ampia banda passante e il costo relativamente basso hanno permesso ai trasduttori e sensori piezoelettrici di imporsi come standard nel mercato. I modelli usati nel progetto sono dei comunissimi “buzzer” dal diametro di 2,8 cm, generalmente usati come segnalatori acustici negli elettrodomestici.

### Diodo di Zener

Si tratta di un particolare tipo di diodo che sfrutta l’effetto Zener, un fenomeno che porta una giunzione P-N percorsa da un’elevata tensione inversa al breakdown, ovvero un repentino aumento di corrente inversa. L’elevato drogaggio cui sono sottoposti i cosiddetti diodi di Zener permette loro di condurre in zona di breakdown, senza che questo precluda il loro funzionamento, come avviene per dei comuni diodi. Questa caratteristica li rende molto apprezzati come stabilizzatori di tensione. Il parametro fondamentale nella scelta di questi diodi è la soglia di zener VZ, che stabilisce la tensione inversa oltre la quale essi tornano a condurre.

## Multiplexing degli ingressi

Avendo scelto di sfruttare uno dei 6 pin analogici per la gestione dei bottini e di dotare la drum machine di 8 pad, si è ritenuto necessario sfruttare un multiplexer esterno, al fine di incanalare tutti i segnali verso un’unica porta analogica. È stato utilizzato il circuito integrato 4051, in grado di multiplex are/demultiplexare fino a 8 canali analogici. Il chip è stato montato su un’apposita scheda, che permette una gestione più agevole de pin. Gli 8 ingressi sono stati collegati alle uscite del circuito dei pad, mentre l’uscita è stata collegata con uno dei 6 pin analogici di Arduino. Il multiplexer è controllato attraverso 3 ingressi, collegati a e controllati da 3 porte analogiche del microcontrollore. Di seguito, lo schema e il circuito stampato della scheda contenente il multiplexer:

### Circuito integrato: 4051 Analog Mux/Demux

L’integrato 4051 è un multiplexer/demultiplexer analogico a 8 canali. Permette di multiplare segnali analogici (quindi continui nel tempo) ampi fino a 15VPP controllando quale ingresso/uscita attivare attraverso 3 ingressi digitali che si attivano con segnali nel range 3-15V. I tre valori inviati a tali ingressi rappresentano il numero (codificato in binario) dell’ingresso/uscita X0-7 che si vuole andare a selezionare. Altre caratteristiche sono: bassa impedenza di uscita (80Ω) e basso assorbimento di corrente sulle alimentazioni e sugli ingressi.

# Gestione del display

Volendo dotare la drum machine di un display, utile per avere sempre sott’occhio le informazioni più importanti e per eseguire una configurazione visuale, si è deciso di valutare un compromesso tra costi e funzionalità. Sono state quindi prese in esame le seguenti caratteristiche:

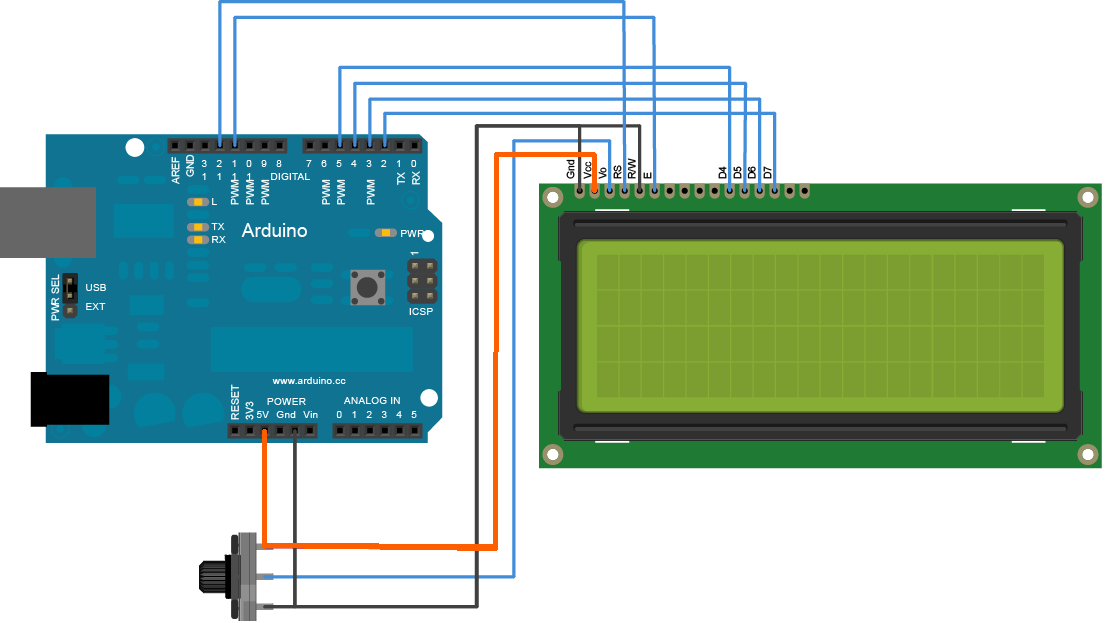
* Tipo di grafica: grafico, a matrice di caratteri, a 7 segmenti
* Tipo di bus: seriale, I2C, parallelo
* Formato: 20×4, 16×4, 16×2…

Dopo aver attentamente valutato queste caratteristiche, anche in relazione alla facilità d’uso con Arduino, si è deciso di usare un display LCD a matrice, con 4 righe e 20 colonne di caratteri, comunicante con il microcontrollore attraverso lo standard previsto dai driver universali HD44780. Per l’uso del display è stato necessario il collegamento ai seguenti 6 pin di Arduino:

* 12: pin *RS* del display
* 11: pin *enable* del display
* 2-5: *data* pin del display

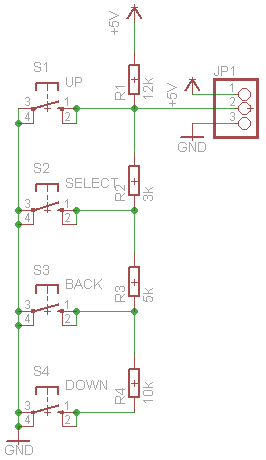
Per risparmiare risorse in Arduino, è stato omesso il collegamento del pin *RW* del display e delle 4 *data lines* facoltative. È stato però necessario alimentare il display, e collegarvi un trimmer per regolare il contrasto.

Di seguito, lo schema riassuntivo dei cablaggi:



# Gestione dei bottoni

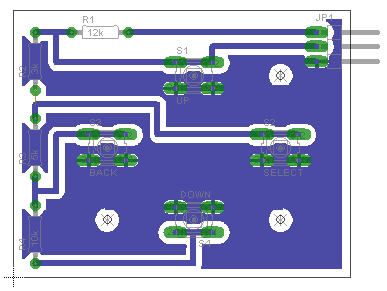
Ad ausilio della navigazione tramite display, si è deciso di inserire una pulsantiera composta di tre bottoni: ⇧, ⇩, ✓ e ⇦ (rispettivamente, scorri su, scorri giù, seleziona e indietro). Avendo la necessità di riservare la quasi totalità dei pin digitali ad altre funzioni si è deciso di adottare un unico ingresso analogico per gestire tutti quattro bottoni. Nel fare ciò è stato sfruttato il principio del partitore di tensione: i terminali degli switch, opportunamente collegati, al momento della pressione bypassano la serie di resistenze (connesse tra 5V e la massa) che li precedono, premettendo di ottenere determinati valori di cadute di tensione, ovvero di stati, rilevabili dall’ingresso analogico. Nel cercare di facilitare il lavoro di programmazione, sono stati adottati diversi valori per le resistenze: dopo una breve simulazione computerizzata è stato possibile vedere come l’uso di resistenze dal valore sempre più grande (seguendo una sorta di andamento geometrico) rendesse più facile la discriminazione dei diversi stati. Di seguito, lo schema del circuito sopra descritto e una tabella riassuntiva con i principali dati ottenuti:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bottone | Tensione  nominale  ΔVN (V) | Tensione  misurata  ΔV (V) | Valore  rilevato  *val* | Valore  software  *(val/100)* |
| *idle*  nessuno | 5 | 4.95 | 1023 | 10 |
| ⇧  scorri su | 0 | 0.00 | 0 | 0 |
| ⇩  scorri giù | 3 | 3.01 | 617 | 6 |
| ✓  seleziona | 1 | 1.01 | 204 | 2 |
| ⇦  indietro | 2 | 2.03 | 413 | 4 |

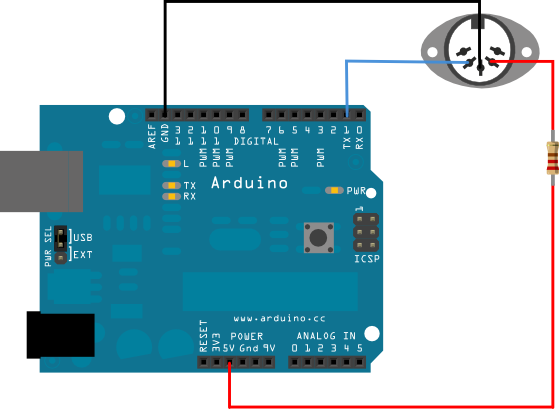
**NOTA:** Nella colonna “Valore software” è stato appuntato il valore usato dal software per riconoscere lo stato di ogni bottone più lo “stato di idle” (nessun bottone premuto). Avendo costatato che i valori rilevati dall’ingresso analogico di Arduino non erano costanti, ma oscillavano di qualche unità, si è ritenuto opportuno aggirare queste oscillazioni avvalendosi di una divisione intera per 100, in quanto l’ordine di grandezza dei dati rilevati si aggirava tra le centinaia.

Si è quindi ricavato un layout definitivo per i componenti, che ha permesso di trasferire il tutto su PCB:



# MIDI output

Tra le diverse funzionalità della drum machine, l’uscita midi è sicuramente la più utile ed interessante, nonché tra le più facili a livello di implementazione hardware. Prevede infatti un collegamento attraverso current-loop con corrente di circa 20mA, instaurato tra due cavi (trasmissione dati e massa, in quanto il dispositivo non prevede l’implementazione di una porta MIDI IN) e una resistenza da 220Ω, collegata tra il pin di output midi di Arduino (nel nostro caso, si tratta del pin TX, usato per la trasmissione seriale). La velocità di trasmissione è di 31.25kbit/s mentre tipo di connettore utilizzato è il DIN a 5 poli, già usato sin dai primi anni ’90 per la connessione di mouse, tastiere o dispositivi audio. Di seguito uno schema del collegamento:



Tuttavia, solo una minima parte dei computer moderni posseggono la porta di comunicazione sopra descritta, mentre gran parte dei dispositivi musicali si è avvicinata all’interfaccia USB, ben più comune e flessibile, permettendo la nascita dei cosiddetti dispositivi “USB-MIDI Compliant”, che incapsulano i dati MIDI dentro pacchetti USB.

Vista la presenza all’interno di Arduino di un chip dedicato alla trasmissione di dati tramite porta USB (il microcontrollore ATmega16U2, dotato di un bootloader DFU e di un firmware interno che emula la presenza di una porta seriale sul computer), si è deciso di riprogrammarlo caricando al suo interno un driver MIDI-over-USB sviluppato da terzi.

Per fare ciò è stato necessario attivare la modalità “DFU Programming” del microcontrollore mettendo in corto due pin di Arduino, e successivamente caricare il firmware precompilato tramite il software Flip, messo a disposizione dalla casa produttrice del circuito integrato. Una volta usciti dalla modalità DFU Programming, il computer riconosce Arduino come un controller MIDI a tutti gli effetti.

# Audio output

Nonostante Arduino provveda egregiamente alla sintesi sonora, il segnale risultante è ricco di armoniche sgradevoli e disturbi a frequenza elevata prodotti dalle commutazioni interne del microcontrollore. In particolare, collegando un carico di bassa impedenza si sarebbe rischiata un’eccessiva richiesta di corrente, con probabile danneggiamento del pin. Si è quindi deciso di implementare un circuito filtrante dotato di buffer di corrente e controllo del volume, per permettere l’utilizzo anche con cuffie auricolari.

## Filtraggio e regolazione

Il circuito che permette tali funzioni può essere suddiviso in quattro stadi principali: filtraggio, buffering, disaccoppiamento e regolazione del volume. I processi sono quindi stati schematizzati nel seguente grafico:

Ogni stadio è stato quindi sviluppato nel modo seguente:

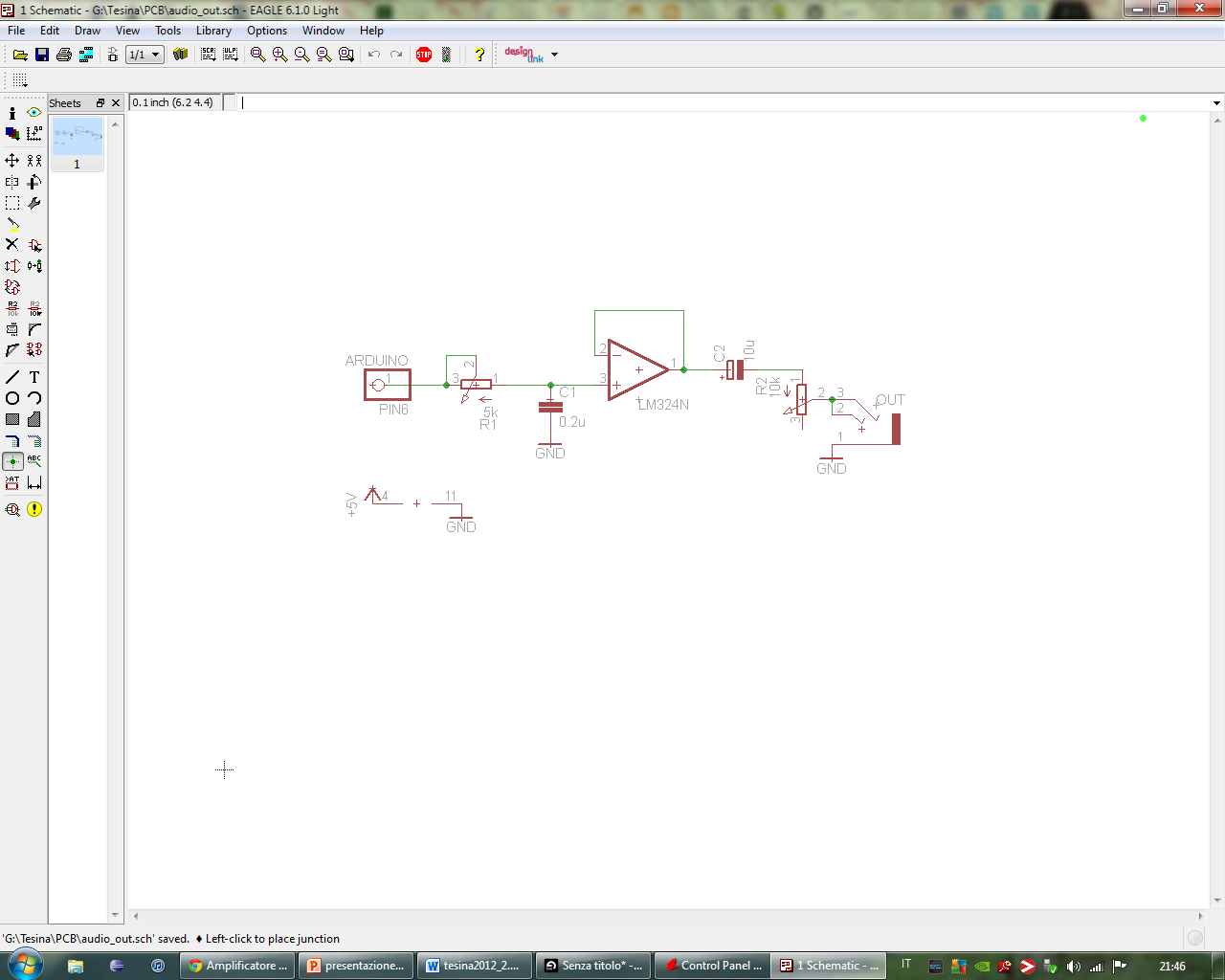
* **Filtro LP**: si tratta di un comune filtro passavo di primo ordine, realizzato tramite rete RC. Si è deciso di prendere come riferimento una frequenza di taglio appartenente allo spettro udibile. Pertanto:



Con una resistenza di 5KΩ e un condensatore di 0.2μF si avrà una frequenza di taglio di circa 160Hz, sufficiente a far percepire un calo delle armoniche. Sostituendo la resistenza con un potenziometro del medesimo valore, è possibile realizzare un filtro con frequenza di taglio variabile, in modo da ridurne l’effetto.

* **Buffer**: è possibile realizzare buffer con transistor BJT e Mosfet, ma si è deciso di impiegare un amplificatore operazionale LM324, alimentato a 5V tramite Arduino. La presenza di questo stadio permetterà il pilotaggio di carichi senza sforzare eccessivamente il pin del microcontrollore.
* **Disaccoppiamento**: Arduino è capace di inviare impulsi digitali a 0-5V, pertanto qualsiasi segnale periodico a onda quadra conterrà una componente continua (valor medio) di 2.5V. Per evitare il passaggio di questa parte del segnale, si è connesso un condensatore in serie ad esso, del valore relativamente alto di 10μF, per non interferire con la modellazione dell’onda..
* **Regolazione del volume**: infine, per permettere il controllo del volume, un potenziometro è stato cablato connettendo rispettivamente segnale di ingresso e massa ai due pin esterni, e segnale di uscita al cursore, operando quindi come un partitore di tensione variabile.

Dallo schema logico è stato realizzato il seguente schema circuitale, montato su breadboard:



### Amplificatore operazionale in modalità buffer

Talvolta risulta necessario dover adattare l’impedenza di un segnale o circuito a quella del dispositivo che lo segue: in particolare, quando si vuole passare da un circuito al alto livello di impedenza, da uno con con livello di impedenza inferiore, si adotta il buffer, anche conosciuto come amplificatore separatore o inseguitore di tensione.

Il circuito del buffer realizzato con l’amplificatore operazionale deriva da quello dell’amplificatore non invertente: infatti, si può assumere l’esistenza di una resistenza di valore infinito collegata tra l’ingresso invertente e la massa, e di una resistenza di valore nullo posta nel ramo della retroazione. Pertanto:

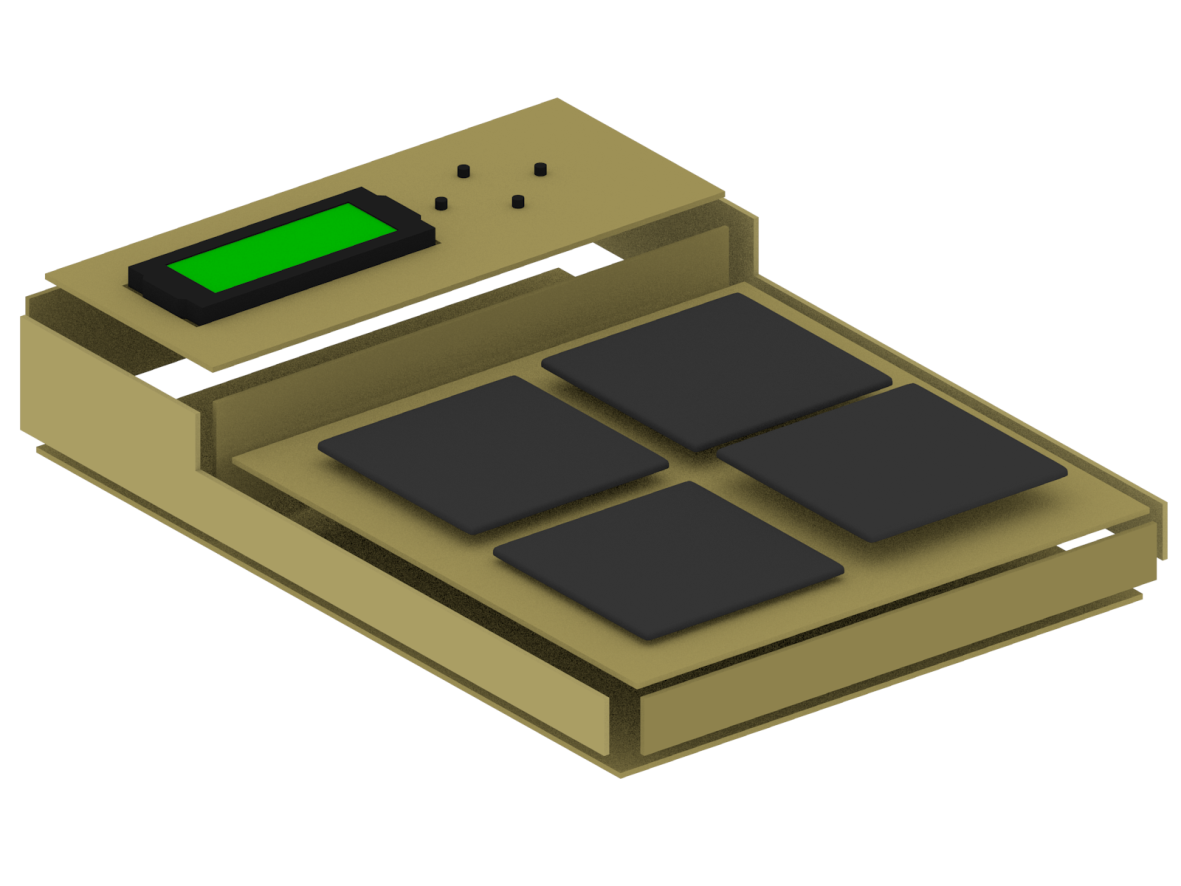


Con Z’=0 e Z=∞, si ha un guadagno di 1. Infatti tale dispositivo non fornisce un guadagno in termini di tensione, bensì di corrente, e quindi di potenza. È comunemente impiegato come stadio iniziale di amplificatori e preamplificatori.

# Studio e montaggio dell’hardware

Per dotare la drum machine di una struttura atta a contenere l’hardware e tutta la circuiteria, si è partiti dall’idea di utilizzare le proporzioni standard del formato A4, ovvero 210 × 297 mm, al fine di rendere lo strumento pratico da trasportare (può infatti essere accomodato ovunque ci sia spazio per dei quaderni, o per una pila di libri). Il primo problema sorto è stato quello della dimensione e disposizione degli 8 pad, che sarebbero diventati troppo piccoli e di conseguenza scomodi da raggiungere con delle bacchette. Si è ovviati a ciò disponendo i pad in due gruppi da quattro e ottenendo due sezioni dalla superfice avanzata al display e ai controlli. Tali sezioni, disposte in senso verticale, permettono un’ottimizzazione dello spazio, nonché una protezione al contatto per i pad. Il materiale usato è del comunissimo compensato, economico e facilmente lavorabile. Una volta ritagliate le sagome delle varie parti, sono state incollate e si è provveduto ad applicare degli spessori in prossimità delle aree destinate a contenere l’elettronica, lasciando i pannelli frontali rimuovibili per future modifiche o aggiornamenti.

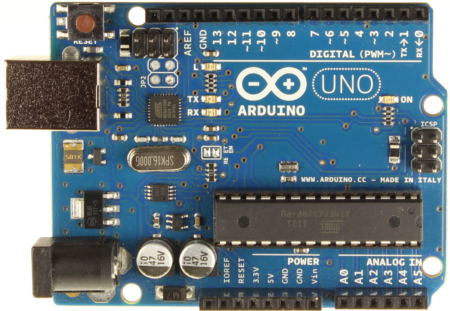
Nell’immagine seguente è possibile vedere una rappresentazione esplosa in grafica 3D del case:



# Approfondimenti

## Arduino

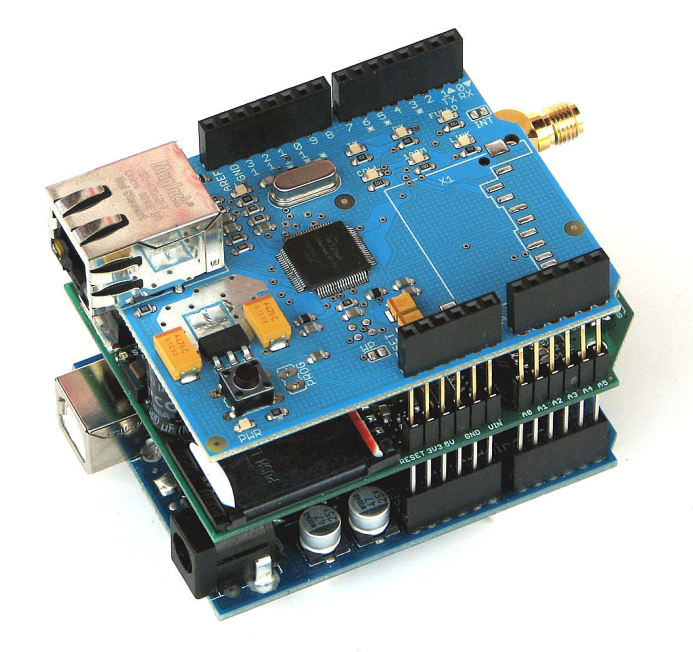
Arduino è un progetto open source nato in Italia nel 2006 con lo scopo di facilitare la prototipazione, la realizzazione di progetti DIY (fai-da-te) e l’apprendimento dell’elettronica. È sviluppato presso l'Interaction Design Institute che ha sede a Ivrea, un istituto di formazione fondato da Olivetti e Telecom Italia. Lo scopo era fornire agli studenti un dispositivo di controllo e sperimentazione che fosse più economico di quelli presenti nel mercato. Il nome della scheda deriva da quello di un bar di Ivrea frequentato da alcuni dei fondatori del progetto. Nel giro di pochi anni sono state vendute centinaia di migliaia di schede, è nata una vasta comunità internazionale di sviluppatori e hobbysti e il progetto originale si è evoluto: sono infatti 13 le versioni di Arduino sviluppate e tutt’ora presenti nel mercato.



Arduino è composto da una piattaforma hardware basata su un circuito stampato che integra un microcontrollore con PIN connessi alle porte I/O, un regolatore di tensione e un'interfaccia usb che permette la comunicazione con il computer. Per la programmazione del microcontrollore è stato rilasciato un ambiente di sviluppo integrato multipiattaforma (Linux, OS X e Windows) composto da un editor, un compilatore, una console seriale e una serie di librerie con le funzioni più comuni. Il linguaggio di programmazione usato dall’IDE è Wiring, derivato dal C++ ed estremamente flessibile e facile da apprendere.

Arduino può interfacciarsi con qualsiasi tipo di sensore e non solo: supporta, infatti, i più famosi protocolli di comunicazione seriale (I2C, SPI, 1-Wire…) e, grazie alla presenza di un chip di conversione TTL-USB riprogrammabile, può comunicare con il PC attraverso comunicazione seriale o MIDI. I principali software che supportano ed implementano la comunicazione con Arduino sono Adobe Flash, Processing, Max/MSP e LabView. Un’altra caratteristica di Arduino, tra l’altro una di quelle che ne hanno decretato il successo, è la possibilità di realizzare, nel pieno rispetto delle leggi sui diritti d’autore, delle copie o imitazioni basate sul progetto originale. Inoltre, per implementare funzioni più specifiche, sono state rese disponibili delle schede di espansione (“shield”) dal facile montaggio e utilizzo, corredate di librerie. Finora sono state realizzate schede dotate di connettore RJ45 (Ethernet shield), schede dotate di modulo ZigBee, alloggiamento SD o anche per il controllo di motori di potenza e led RGB.

La piattaforma hardware Arduino Uno (la versione usata nella drum machine) è basata sul microcontrollore Atmega328 prodotto dalla Atmel. Dispone di 14 PIN di I/O digitali di cui 2 dedicati alla comunicazione seriale e 6 capaci di fornire uscita PWM, usata per il controllo della velocità di motori o luminosità dei LED. Sono inoltre presenti 6 ingressi analogici a 10 bit. La tensione massima rivelabile dall’ADC interno è impostata a 5V per default, ma può assumere valori minori se applicata al pin AREF. Grazie al regolatore di tensione, la scheda è capace di erogare 5V o 3.3V per un massimo di 500mA. È inoltre presente un bottone di reset, un plug per alimentazione esterna e una porta USB gestita da un microcontrollore dedicato, l’Atmega8, anch’esso riprogrammabile per assolvere altri compiti.



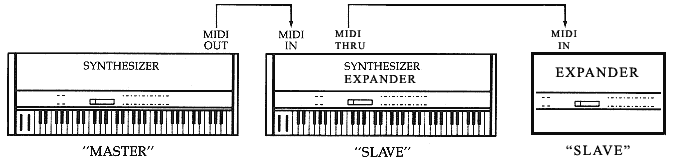
## Musical Instrument Digital Interface (MIDI)

Il Musical Instrument Digital Interface (universalmente conosciuto come MIDI) è un protocollo standard per la comunicazione e l’interazione tra strumenti musicali elettronici e, in generale, per hardware dedicato alla produzione musicale. Fornisce un sistema flessibile, affidabile e facile da implementare, tanto da essere rimasto sostanzialmente invariato per ben trent’anni.



Nasce nel 1981 per opera di due ingegneri della SCI, come alternativa ai sistemi proprietari proposti da altre aziende. A quel tempo la tecnologia limitava il numero di voci e di timbri (polifonia) di ogni strumento e i tastieristi dovevano svolgere manualmente i gravosi compiti di sincronizzazione e configurazione delle numerose apparecchiature di cui facevano uso. Diversi produttori mostrarono interesse verso il MIDI e nel 1985 fu fondata l’International Midi Association, che pubblicò le specifiche del protocollo v1.0 rendendolo uno standard.

La comunicazione può avvenire tra due o più dispositivi, seguendo diverse topologie. Ogni cavo trasporta un singolo stream di dati monodirezionale (simplex). In genere, si usa collegare due dispositivi tramite un cavo, in modo da averne uno che invia i comandi (“master”) e un altro che li interpreta ed esegue (“slave”). Grazie all’integrazione di sedici canali, è possibile adottare anche dei sistemi di trasmissione più complessi, definendo con precisione i segnali destinati a un dispositivo specifico, tra i tanti connessi in cascata.



(Nell’esempio in figura, i messaggi MIDI inviati dal primo sintetizzatore, raggiungono il secondo dispositivo e contemporaneamente anche il terzo, attraverso la porta “MIDI Thru” che ripete il segnale in ingresso. In questo modo è possibile comandare due expander.)

I messaggi MIDI sono composti da due parti: uno “status byte” ed uno o più “data byte”. Lo status byte, identificato dal primo bit posto a 1, definisce il tipo di messaggio midi, ovvero l’azione che lo slave dovrà compiere. Il resto del nibble più significativo (3 bit) permette di ottenere 8 differenti messaggi:

* 000 NOTE OFF
* 001 NOTE ON
* 010 POLY KEY PRESSURE
* 011 CONTROL CHANGE
* 100 PROGRAM CHANGE
* 101 CHANNEL PRESSURE
* 110 PITCH BEND PRESSURE
* 111 SYSTEM MESSAGES

I primi 7 messaggi vengono definiti “Channel Voice Message” in quanto i successivi 4 bit indicano il canale al quale il comando è stato trasmesso. La combinazione 111 identifica invece una serie di messaggi speciali, usati per la sincronizzazione e la configurazione, chiamati “System Message”. Dei possibili 16 messaggi di sistema, solo 12 vengono adottati. Uno di questi, identificato dallo status byte 11110000 e chiamato “System Exclusive Message”, è divenuto molto importante con l’avvento dei computer, in quanto permette, ad esempio, l’aggiornamento del firmware di un dispositivo tramite collegamento MIDI, o il download e l’upload di presets e impostazioni.

I “data byte” vengono invece utilizzati per specificare uno o più parametri assegnati ad uno status byte. Ogni comando necessita di un determinato numero di data byte: ad esempio, NOTE ON presenta due data byte, uno contenente la nota da suonare ed un altro contenente la forza con cui è stato premuto il testo. Ogni data byte permette 128 valori diversi. In alcuni casi, due data byte vengono usati assieme, permettendo una risoluzione di 16384 valori.

## Linguaggio C++/Wiring

Arduino viene programmato tramite linguaggio Wiring derivato dal C++. In realtà Wiring è solo un set di librerie e funzioni fornite allo scopo di facilitare lo sviluppo da parte degli utenti meno esperti. Tali funzioni ricoprono una vasta gamma di scopi e applicazioni tipiche dei microcontrollori, permettendone un completo, ma al tempo stesso facile ed intuitivo, controllo.

Wiring perciò poggia sulle solide basi del C++, e come tale, prevede la programmazione a oggetti, permettendo all’utente di implementare le proprie classi con metodi e attributi personalizzati. Consente la gestione diretta dei puntatori, l’esecuzione di operazioni booleane, tra bit, o ricorsive, e la creazione di strutture. Offre inoltre delle funzioni integrate per la gestione del tempo, per la conversione tra diversi tipi di dato, la manipolazione dei singoli bit di una variabile e la generazione di numeri pseudo-casuali, nonché funzioni matematiche, trigonometriche, di input/output e di controllo degli interrupt. Nonostante molte di queste funzionalità siano già presenti nel linguaggio C++, i loro algoritmi sono stati riscritti ed ottimizzati per l’esecuzione in microntrollori.

Un programma scritto tramite linguaggio Wiring è chiamato sketch. Ogni sketch si compone di due funzioni principali, implementabili dall’utente:

* **setup()**: viene eseguita appena il bootloader carica lo sketch. Contiene, in genere, l’inizializzazione dei pin di Arduino e la dichiarazione di variabili o impostazioni non modificabili.
* **Loop()**: funzione reiterata fintanto che Arduino è in funzione. Contiene le azioni da compiere.

Il codice Wiring è compilato tramite una versione del famoso compilatore open-source GCC, ottimizzata per microcontrollori basati su tecnologia AVR della Atmel.

# English Corner

## What is Arduino?

Arduino is an electronic prototyping platform, developed by an Italian team with the “open-source” philosophy, meaning that every part of the project is free and easily accessible and modifiable. Arduino is focused on ease to use, so it gained a lot of popularity and became a standard for learning electronics. The Arduino project has released two part: the hardware and the software. The first part consists of a printed circuit board containing the microcontroller with its I/O ports, and a USB socket. The software part is the development environment (IDE) used to program the firmware. It uses a programming language based on C++, called Wiring. It also provides several libraries and examples ready to be tested.

Arduino can communicate with many devices: it can get information from the outer world, reading values from proximity or temperature or humidity sensors, manipulate data, drive LEDs, small servo motors and even show information on a screen. Furthermore, several boards with an easy mounting system called “shields” are available, each one with its own characteristics and intent. There are shields designed to communicate through Ethernet, other ones that has a SD Card slot, and so on.



Of course, communication is also possible between Arduino and the computer, as there are many software that completely support it. The most notable are: Adobe Flash, Processing and LabView. In fact, Arduino come with a USB connector that emulates a serial port on a computer. In this way, the board can be fully controlled from the PC, while the data coming from it are presented with a more attractive graphic.

The board comes with several input/output areas called “pins”. Each pin can be either analog or digital, and can send or receive data. In order to use Arduino, you have to link all your sensors and stuff to the proper pins, and then to write and upload the code into it.

## What is MIDI?

The “Music Instrument Digital Interface” is a standard protocol used to transfer musical data among digital instruments or PCs.

It has been developed in 1981 by two engineers of SCI. Their aim was to create an easier way to allow different synthesizers and expanders to communicate each other, in order to play polyphonic and more complex sounds. So, when in 1983 the first MIDI-compliant device came out, many manufacturers became interested and formed the MMA (Midi Manufacturers Association) that ratified the first version of the protocol. It is so reliable and easy to implement that it’s still used nowadays, 30 years later.



The protocol itself is a collection of messages, each one with its own role. Every message is made of 2 parts: a “status byte”, containing the command identifier, and one or more “data bytes”, containing the parameters used by the command. The most important MIDI messages are: NOTE ON (plays a note), NOTE OFF, (stops a note), PITCH BENDER (change the pitch of the currently played notes) and PROGRAM CHANGE (switches among several sound banks). MIDI also allows changing of about 127 other parameters, such as volume, panning, or expression. It also manages the synchronization between two devices, for example a PC and a keyboard. In a physical MIDI connection, the device that sends the command assumes the role of “master”, the other one is the “slave”.

The most notable kinds of devices that use MIDI are:

* **Synthesizers**: they convert MIDI signal coming from an “IN” port into pitched or percussion sounds. They can provide a keyboard, a series of plastic pads (drum machines) or none of them: in this case they are called “expanders” and they can be rack-mounted.
* **Controllers**: they are the instruments physically played by the musician in order to make a MIDI data stream. They don’t produce any sounds and need a synthesizer to generate it. They can take the form of existing instruments, like pianos, guitars or drum sets. They can also contain various switches, buttons or sliders used to control any aspect of the performance, such as volume, reverb, distortion…
* **Sequencers**: They allow the performer to store and play any MIDI sequence he wants. They can be either hardware or software. Hardware sequencers are tiny devices with a display and a keyboard to scroll the sequence or save a new one. The software version is a program running on a PC provided with a MIDI port. They usually allow time-stretching and key-transposing of the recorded sequence.

mauroarca@alice.it