Ministro dell'Istruzione, dell'Università e della Ricerca Alta Formazione Artistica Musicale

Conservatorio di Musica "Giovanni Pierluigi da Palestrina" Cagliari

Triennio Ordinamentale di I Livello

Musica Elettronica

Kompas: progettazione di un sintetizzatore modulare Eurorack basato su Arduino

Prova Finale

Relatore: Prof. Daniele Ledda **Candidato: Stefano Manconi**

A.A 2017/2018

Dedicato a mio fratello Marco

Indice

Introduzione

L'oggetto di questa tesi è la progettazione di uno strumento musicale elettronico basato sullo standard dei sintetizzatori modulari Eurorack, attraverso l'implementazione della piattaforma di sviluppo Arduino.

Nell'anno accademico 2016/17 sono stato accettato come tirocinante presso una piccola ma ben nota azienda produttrice di strumenti musicali elettronici in Repubblica Ceca. Durante questa esperienza ho contribuito alla progettazione di diversi strumenti sviluppando delle competenze che, nutrite nel tempo libero, mi hanno portato a ideare e realizzare il modulo che presenterò in questa tesi.

Il percorso intrapreso in questi anni nel campo della Musica Elettronica, dal punto di vista accademico ed extra-accademico, ha coinvolto molteplici discipline, talvolta interconnesse al punto da costringermi ad un'analisi globale e ad una percentuale di approfondimento degli argomenti proporzionata agli obiettivi. Questa premessa è indispensabile se si vuole definire il contesto personale, dal quale la mia ricerca ha origine e se si vogliono tracciare i limiti nella stesura degli argomenti trattati. A seguito di tali considerazioni e presupposti, il testo si sviluppa formalmente come documentazione dei passi più significativi della progettazione dello strumento, con il duplice obiettivo di fornire risorse utili in materia di prototipazione e dalle quali poter trarre considerazioni generali sulla realizzazione di uno strumento musicale elettronico.

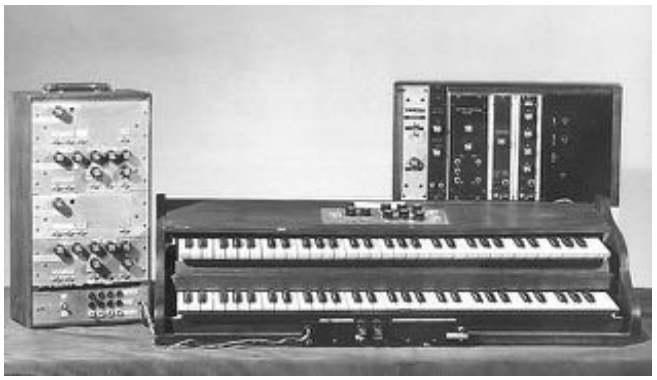
Con questo lavoro vorrei esprimere la mia volontà di condividere la conoscenza con la massima umiltà e onestà intellettuale, nella speranza di contribuire a un mondo migliore attraverso l'invito alla cultura e al rispetto collettivo. La mia tesi e i suoi contenuti saranno reperibili gratuitamente online e resi a disposizione dei colleghi del conservatorio e del mondo.

0. Il sintetizzatore modulare

Il sintetizzatore modulare è uno strumento musicale elettronico, composto per l'appunto di *moduli*, ossia sezioni indipendenti e intercambiabili, dedicate a svolgere un compito più o meno specifico all'interno del sistema.
Lo strumento presentato in questa tesi è un *modulo* realizzato appositamente per sintetizzatore modulare, secondo gli standard descritti nei capitoli a seguire.

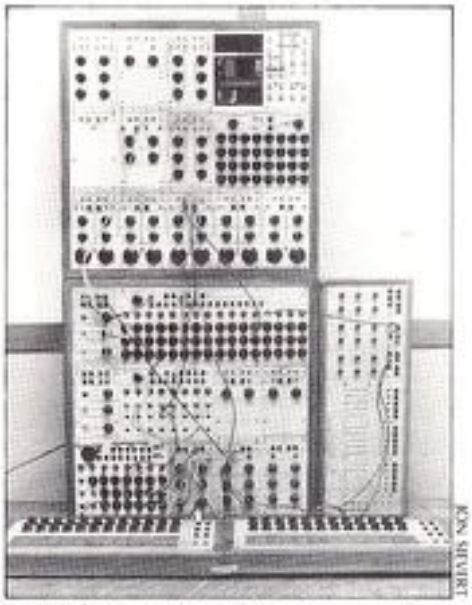
0.0 Breve excursus storico

Il sintetizzatore modulare nasce negli Stati Uniti a cavallo degli anni '60 ad opera di Robert Moog (New York 1934 - Asheville 2005) e Don Buchla (South Gate 1937 - Berkeley 2016) tuttavia, al contrario di quanto l'accostamento dei due nomi possa suggerire, i due ingegneri non furono collaboratori ma si trovarono a progettare lo stesso tipo di strumento nello stesso periodo, al punto da rendere difficile poter affermare a chi dei due appartenga il primato.
Il Moog Modular e il Buchla 100 Serie che prendono il nome dai loro costruttori, sono stati i primi sintetizzatori modulari che, pur trattandosi dello stesso strumento, presentavano delle differenze sostanziali che al giorno d'oggi continuano a contraddistinguere i prodotti delle due case e che raccontano molto della vita e degli approcci musicali dei loro inventori.



Robert Moog proveniva da una scena definibile oggi col termine "D.I.Y." (Fai da te) e i suoi primi progetti riguardavano la distribuzione del celebre Theremin, nel formato di kit di montaggio; Don Buchla apparteneva invece alla comunità Hippie e la progettazione del sintetizzatore gli venne commissionata da parte del San Francisco Tape Music Center. Se da una parte degli States Moog propose uno

strumento votato al temperamento equabile e integrato con una tastiera come quella del pianoforte, dall'altra parte, Buchla realizzò uno strumento dedicato alla sperimentazione in ambito elettroacustico che potesse convivere con le pratiche compositive di quella scuola.
Il suono dei due sintetizzatori può essere apprezzato nei brani tratti dagli album "Switched-On Bach" di Wendy Carlos e "Silver Apples of the Moon" di Morton Subotnick, nei quali le differenze sono radicali al punto che i due modelli di sintesi siano stati ribattezzati come East-Coast (Moog) e West-Coast (Buchla) synthesis.



Buchla's first complete modular system, presently at Mills College, Oakland.

Nel 1971, stavolta nel Regno Unito, fece comparsa un altro esemplare di sintetizzatore modulare, il Synthi 100 realizzato dalla compagnia EMS per il centro di produzione radiofonica della BBC. Il suono di questo sintetizzatore caratterizzò gran parte dei programmi "Sci-Fi", al punto che l'accostamento fra i timbri sintetici e questo genere di narrazione, diventò presto parte dell'immaginario collettivo. Fra gli esempi più conosciuti vi è senz'altro la serie "Doctor Who" le cui musiche sono state in parte curate da Delia Derbyshire (5 Maggio 1937 - 3 Luglio 2001).

Negli anni a seguire diverse case produttrici si affermarono nel mercato degli strumenti musicali, i sintetizzatori modulari e semi-modulari stavano acquisendo popolarità sino all'inizio degli anni '80, quando, con l'avvento del digitale, strumenti come il celebre sintetizzatore Yamaha DX7, fecero comparsa e grazie al loro rapporto funzionalità, portabilità e prezzo, conquistarono un grande successo, facendo diminuire l'interesse verso i sintetizzatori analogici.

Dopo questa fase di declino, la storia del sintetizzatore modulare riprende in Germania, negli anni ‘90, grazie a di Dieter Döpfer che, dopo aver realizzato diversi strumenti basati sul protocollo MIDI, voltò l’interesse verso i circuiti analogici e la distribuzione di strumenti in kit di montaggio, introducendo nel mercato un nuovo tipo di sintetizzatore modulare. Il system A-100 della Doepfer Musikelektronik fece rinascere l’interesse per i sintetizzatori modulari attraverso un nuovo standard di specifiche tecniche dal nome Eurorack, la cui architettura consisteva in un formato di dimensioni inferiori, in grado di migliorare la portabilità dello strumento e renderlo reperibile ad un prezzo decisamente inferiore. Grazie a questa formula, lo standard Eurorack incoraggiò negli anni a seguire la comparsa di numerosi costruttori, professionisti ed amatoriali, che continuano a crescere esponenzialmente sino al giorno d’oggi, nel quale se ne contano più di 270 per oltre 5000 moduli presenti sul mercato.

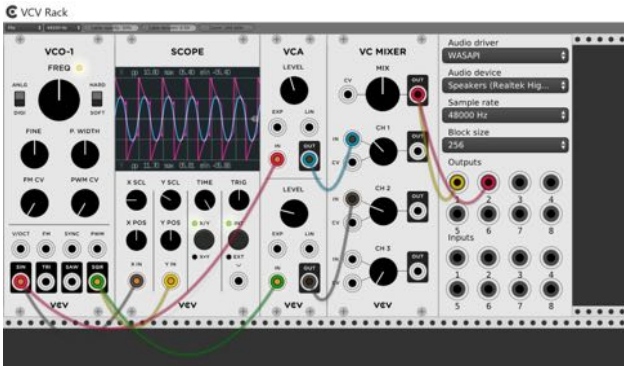
0.1 Il “virtual modular”

A metà degli anni ‘80, nel centro di ricerca IRCAM di Parigi, Miller Puckette iniziò a sviluppare Max, un ambiente di programmazione la cui interfaccia grafica richiamava quella del sintetizzatore modulare e che si configurava come un linguaggio orientato all’interfaccia con strumenti esterni e successivamente alla sintesi del suono. Nel ‘90 il pacchetto Max fu acquisito da David Zicarelli che iniziò a distribuirlo commercialmente nel ‘99 come software eseguibile dai computer domestici.

Nel 1996 fece comparsa il software Reaktor della Native instruments, il quale consentiva la simulazione del sintetizzatore modulare nel personal computer.

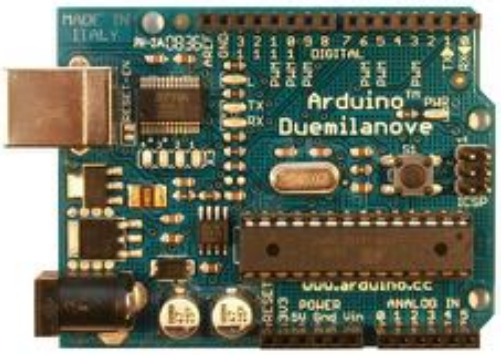
Nel 1998 la compagnia svedese Clavia progettò il Nord Modular G1, uno strumento hardware digitale ispirato ai sintetizzatori modulari classici. L'hardware dispone di un potente motore DSP, programmabile nel computer tramite un'interfaccia grafica, nella quale l'utente può accedere a una libreria di moduli virtuali, progettare la propria "patch" e memorizzarla all'interno dello strumento.

Negli anni a seguire diversi produttori di software hanno distribuito strumenti basati sull'architettura modulare. Il più recente e unico nella sua implementazione è senz'altro il software VCV Rack, sviluppato da Andrew Belt e distribuito gratuitamente nel 2017 come pacchetto open-source dedicato alla simulazione di un sistema modulare. La peculiarità di VCV Rack, oltre a contenere una libreria di moduli classici, è quella di essere espandibile con simulazioni di moduli hardware provenienti da diversi produttori, affermati nel mercato Eurorack.



0.2 Arduino come strumento del D.I.Y

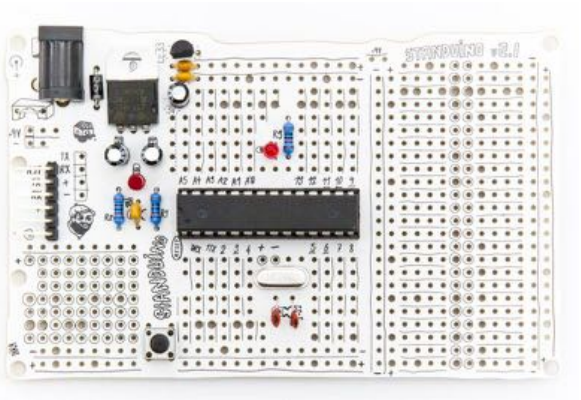
Arduino è una piattaforma elettronica open-source sviluppata nel 2005 da Massimo Banzi e David Cuartielles presso l'Interaction Design Institute di Ivrea. La piattaforma nasce come strumento di prototipazione rapida ed è costituita da una parte hardware, basata su microcontrollore e da una parte software ossia un ambiente di sviluppo, derivato dal linguaggio di programmazione Processing.



Con la distribuzione di uno strumento semplice e intuitivo ma allo stesso tempo capace di soddisfare applicazioni più complesse, Arduino ha contribuito allo sviluppo di una comunità mondiale di hobbisti, ricercatori e professionisti nel campo dell'elettronica D.I.Y. fra cui, le applicazioni in ambito musicale, non potevano tardare a fare la loro comparsa.

Standuino

In repubblica Ceca, verso la fine degli anni 2000, due studenti dell'Accademia di Belle Arti iniziarono a seguire l'ondata dei "makers" che si stava sviluppando nel resto dell'Europa, prendendola come spunto per investigare il passato della cultura D.I.Y. nella Cecoslovacchia.
Provenendo da un contesto nel quale le università non disponevano dei mezzi per insegnare tecnologie come Arduino e prendendo ispirazione da un pioniere del D.I.Y. come Standa Filip, Václav Peloušek e Ondřej Merta svilupparono insieme un clone della scheda, dando vita al progetto Standuino. Nel 2011, con la nascita ufficiale del progetto, il team Standuino iniziò un'intensa attività di workshop in Europa, rivolti a istituzioni e associazioni, nel quale venivano realizzati degli strumenti musicali basati su Arduino, attraverso la loro scheda di prototipazione. Nel 2013, la progettazione di strumenti musicali elettronici diventò specializzata al punto che il progetto Standuino venne trasformato nell'azienda, basata sul collettivo, chiamata Bastl Instruments.



1. Ideazione dello strumento

1.0 Concept

Lo strumento progettato in questa tesi è un sequencer probabilistico a tre voci, basato sul formato dei sintetizzatori modulari eurorack.
Kompas, che in lingua ceca significa bussola è dedicato alla pratica musicale dell'improvvisazione e dell'utilizzo dei processi "generativi", all'interno di uno strumento musicale storicamente affermato come il sintetizzatore modulare.
Gli algoritmi alla base di questo modulo sono stati implementati nell'ottica di ottenere un certo grado di imprevedibilità e della quale avere controllo dei margini. Per queste caratteristiche, Kompas rappresenta uno strumento di "esplorazione" nelle applicazioni tipiche della East e della West Coast synthesis.

Nell'universo dei sintetizzatori modulari, Kompas appartiene alla famiglia dei *sequencer*, introdotti per la prima volta nel *Buchla Series 100*.
Il sequencer è una tipologia di modulo che genera degli eventi e li ripete secondo una ciclicità paragonabile a un *loop* di nastro magnetico, dove gli eventi non sono altro che voltaggi utilizzabili come *parametri di controllo* per altri moduli del sistema. Il sequencer è composto da un numero finito di "eventi" chiamati *step* (tipicamente dagli 8 ai 16), ciascuno dei quali contiene un determinato valore di tipo *CV* (da 0 a 5 volts), *gate* (on/off - 0/5 volts) o *trigger* (impulso di 5 volts) impostabile manualmente. Gli step possono essere richiamati manualmente o tramite segnali esterni di tipo *clock* / *trigger* o *CV*, in questo modo, ogni volta che un determinato step viene richiamato, il suo valore memorizzato viene riportato all'uscita del modulo per essere indirizzato verso altre destinazioni.

Il sequencer in analisi viene inoltre catalogato come "trigger sequencer" in quanto produce degli eventi di tipo impulsivo.

Il modulo è composto di tre voci, o tre sequencer, ciascuno dei quali condivide lo stesso *clock* ossia lo stesso "ritmo" di avanzamento fra gli steps i quali, alla versione corrente, sono 16 per voce. A differenza dei sequencer classici, questo prototipo non offre la possibilità di programmare manualmente ogni step ma,

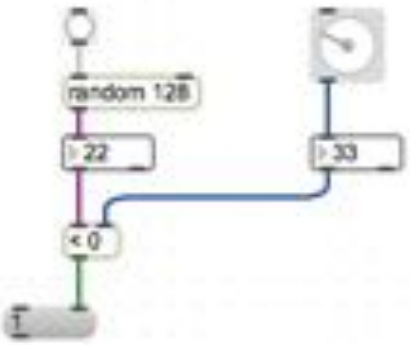
tramite un unico controllo per voce, l'utente può determinare la media degli eventi da generare.
Questa implementazione genera dei *pattern* secondo il modello probabilistico, riscontrato frequentemente all'interno di pratiche compositive d'avanguardia, basate sull'utilizzo dei *processi stocastici.*

1.1 Proof of concept

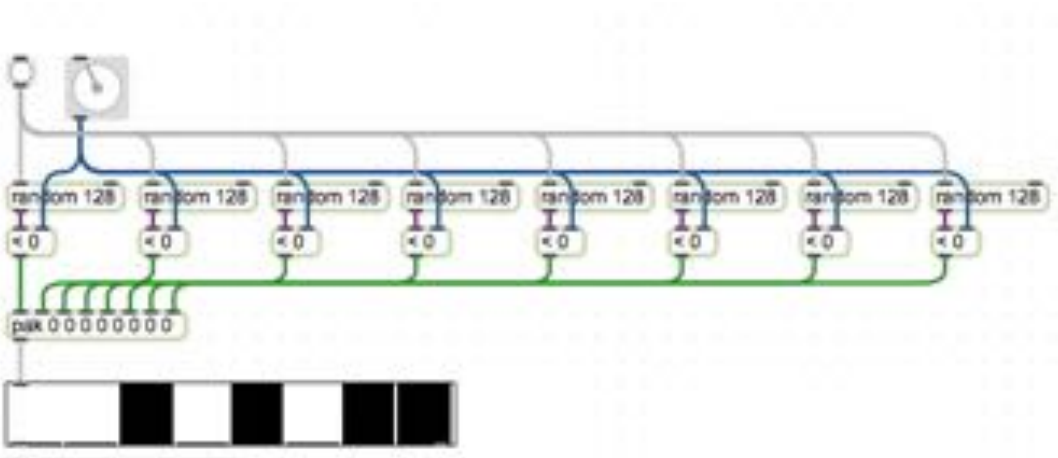
Nel lessico tecnico, con l'idioma inglese "proof of concept" si fa riferimento a una procedura di verifica dell'idea di partenza, al fine di dimostrare i risultati che potrebbero essere ottenuti nella fase di prototipazione. I sintetizzatori modulari "virtuali", accennati nel primo capitolo, possono essere una risorsa utile in quanto consentono di simulare il comportamento dello strumento e verificarne la validità in ambito musicale. Nel seguente capitolo vengono illustrati gli algoritmi alla base del sequencer probabilistico che, nel caso di questo strumento, sono stati verificati sull'ambiente di programmazione Max 6.

L'algoritmo alla base del sequencer probabilistico è costituito da un generatore numerico di tipo randomico e da un operatore di confronto.

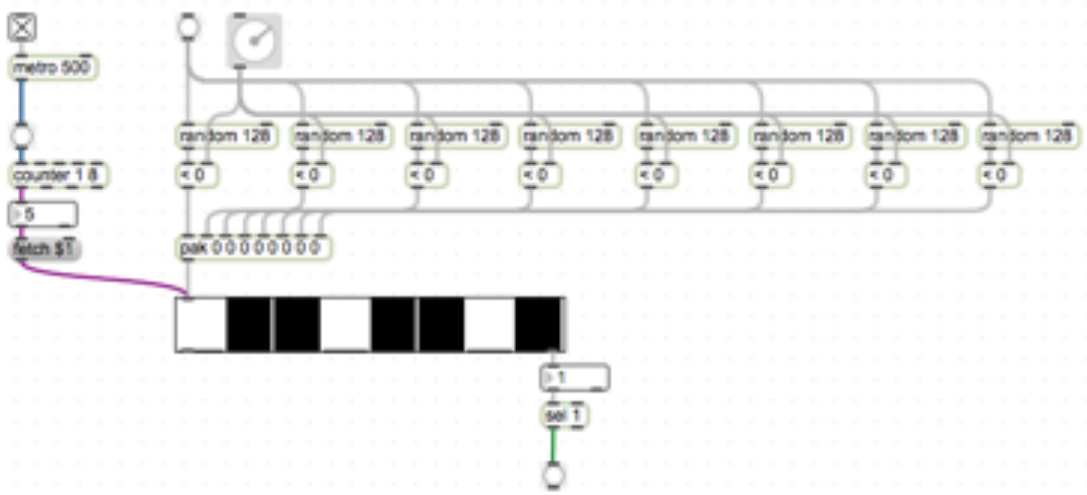
Nella patch raffigurata sulla destra, il numero generato randomicamente viene confrontato con un valore fisso immesso dall'utente: nel caso in cui il numero randomico sia inferiore a quello stabilito, il risultato dell'operazione sarà 1 ossia TRUE ("vero" in gergo dell'informatica booleana), nel caso contrario il risultato sarà 0 ossia FALSE (falso).



In base a questo principio di funzionamento,
se si ripete l'operazione per "n" volte, quante la durata in steps della nostra sequenza, per ogni step otterremo un responso "vero" o "falso" che equivarrà allo stato dello step ossia acceso o spento. Nella figura sottostante è riportata l'implementazione di un sequencer a 8 step secondo la stessa logica che costituisce una voce dello strumento.



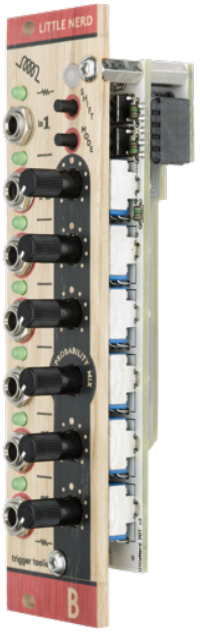
Dopo aver programmato l'algoritmo di generazione della sequenza, il passo successivo è quello di richiamare i singoli step ossia i valori memorizzati al suo interno. Per eseguire questo tipo di operazione è necessario utilizzare un "contatore" che, per ogni impulso ricevuto dall'esterno, faccia avanzare la lettura dello step successivo e al termine della sequenza riprenda il conteggio da capo, come in un loop. L'applicazione più diffusa nei sintetizzatori analogici è quella di utilizzare come impulso esterno una sorgente di tipo clock, simulabile nell'ambiente max tramite l'oggetto "metro". Per ogni step richiamato, se il suo valore è 1, verrà prodotto un impulso in uscita dall'oggetto "sel", con la stessa logica di comportamento del sequencer hardware.



Le patch mostrate in questo capitolo sono gli algoritmi essenziali alla base dello strumento e possono essere consultate online al link:

https://github.com/stziopa/kompas/tree/master/max

2. Componenti Elettroniche



Una volta stabilita la validità del concept, il passo successivo è quello di trovare delle soluzioni di tipo hardware che siano funzionali alle caratteristiche operative e di interfaccia dello strumento.

La ricerca delle componenti elettroniche e della loro implementazione sono state ispirate a moduli già presenti sul mercato Eurorack, come Little Nerd della Bastl-Instruments.

Little Nerd è un processore di segnali di tipo clock, trigger e gate basato su arduino. Gli schemi elettronici e il codice sorgente di questo modulo sono reperibili gratuitamente sulla piattaforma online Github, sotto la licenza Creative Commons CC-BY-SA.

2.0 Dal software verso l'hardware

Attraverso le linee guida sulla costruzione di "Arduino D.I.Y", reperibili nel sito ufficiale di arduino e riscontrabili nei diagrammi dei moduli sopra elencati, il primo passo è quello di implementare il microcontrollore sulla *breadboard* e fare in modo che possa essere alimentato da un sistema dello standard Eurorack.

Nei diagrammi seguenti, tratti dal modulo Little Nerd, sono evidenziati i blocchi relativi al microcontrollore, alla sua alimentazione e alle connessioni interne ed esterne.

Alimentazione

L'alimentazione dei moduli, secondo lo standard Eurorack, è di +12 e -12 volt e rappresenta l'amplitudine massima dei segnali consentiti all'interno del sistema. Arduino Uno è basato sul microcontrollore ATMEGA328 che supporta voltaggi della portata massima di +5 volt.

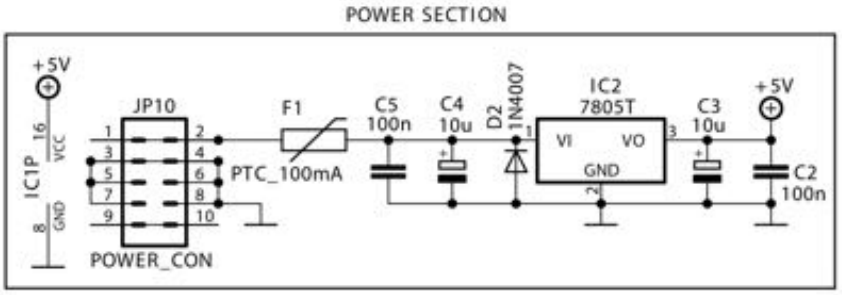
Per alimentare il microcontrollore è necessario dunque convertire la potenza da +12v a +5v attraverso un regolatore di voltaggio (IC2 - 7805).

Gli altri elementi presenti nel diagramma, come il fusibile (F1 - 100mA) e il diodo (D2 - 1N4007) servono a proteggere il modulo in caso di correnti superiori alla portata consentita e in caso di polarità inversa (-12V invece che +12V); le restanti componenti (C5, C4, C3, C2) sono dei condensatori dedicati al filtraggio dell'alimentazione con lo scopo di smussare i possibili picchi presenti nel flusso.

Elenco delle componenti:

1x Jumper 2x5 (JP10);
1x Fusibile 100mA (F1);
2x Condensatore ceramico 100nF (C5 e C2);
2x Condensatore elettrolitico 10uF (C4 e C3);
1x Diodo 1N4007 (D2);
1x regolatore di Voltaggio 7805T (IC2);

Diagramma elettronico alimentazione:

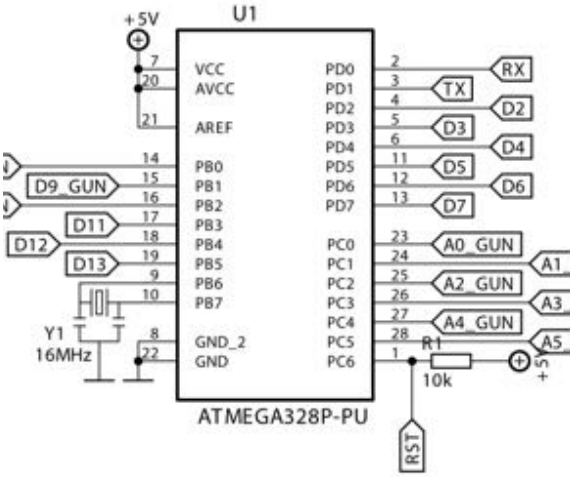


Microcontrollore e Porta Seriale

I blocchi successivi rappresentano il cablaggio del microcontrollore e della porta seriale secondo le istruzioni di Arduino D.I.Y. (diagrammi tratti da Little Nerd).

Componenti microcontrollore:

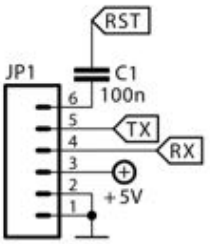
1x IC ATMEGA328-PU (U1);
1x Risonatore 16MHz (Y1);
1x Resistenza 10k (R1);



Per la connessione seriale, necessaria alla configurazione di Arduino, bisogna inoltre disporre di una scheda d'interfaccia esterna come la FTDI Basic del marchio Sparkfun che può essere collegata temporaneamente al pin header JP1.

Componenti porta seriale:

1x Pin header 1x6 (JP1);
1x Condensatore ceramico 100nF (C1);

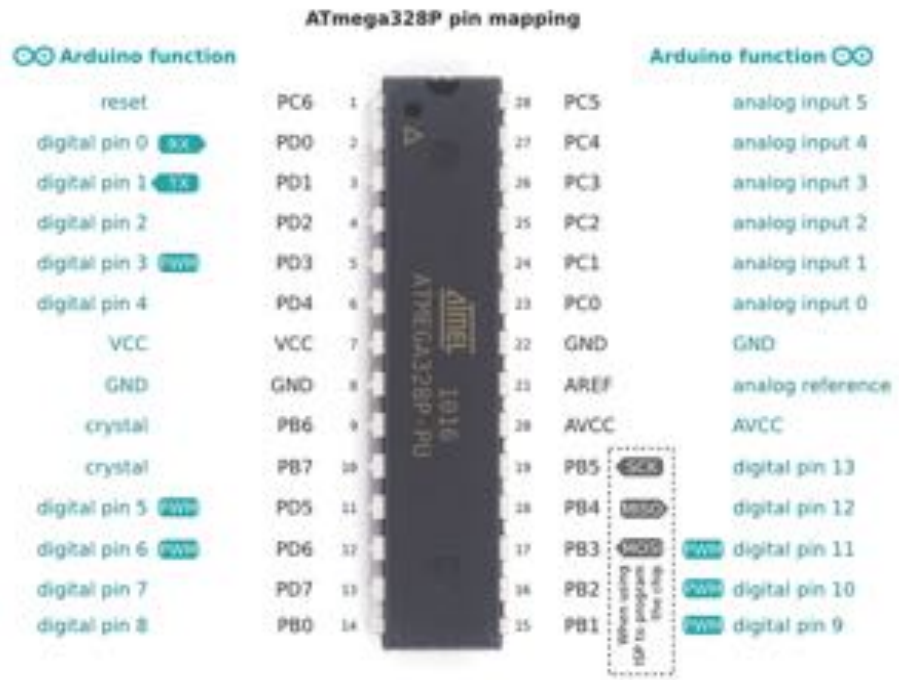


Assemblando i blocchi indicati si ottiene il clone di Arduino Uno.

Ingressi e uscite

Gli step successivi mostrano come implementare le sorgenti di input ed output necessarie al funzionamento dello strumento.

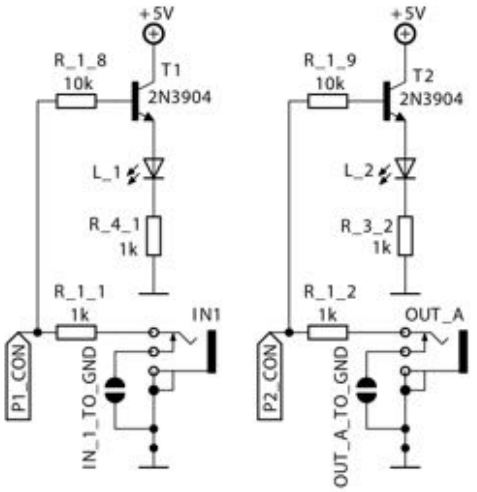
Lo scopo delle componenti elettroniche elencate di seguito, è principalmente quello di proteggere le porte del microcontrollore da voltaggi e correnti maggiori della sua portata. Poiché il microcontrollore Atmega328 è composto di 14 pin digitali e di 6 pin analogici, è necessario fare una distinzione fra le componenti, a seconda della porta che si andrà utilizzare.



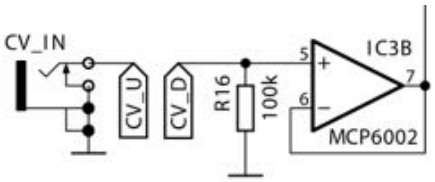
I pin digitali possono essere utilizzati come ingressi o come uscite di tipo on/off (fatta eccezione per le porte “PWM” - Pulse Width Modulation), per questo motivo si prestano particolarmente per ricevere o generate segnali di tipo trigger, gate e clock. Gli ingressi analogici sono invece dedicati alla lettura di segnali di tipo CV con una definizione di 10 bit.

Componenti di ingresso/uscita pin digitali:

1x Resistenza 10k (R_1_8);
2x Resistenza 1k (R_1_1 e R_4_1);
1x Transistor 2N3904 (T1);
1x Led (L_1);
1x Jack Mono 3,5mm (IN1);



Componenti di ingresso analogico (Bastl-Instruments GrandPA):

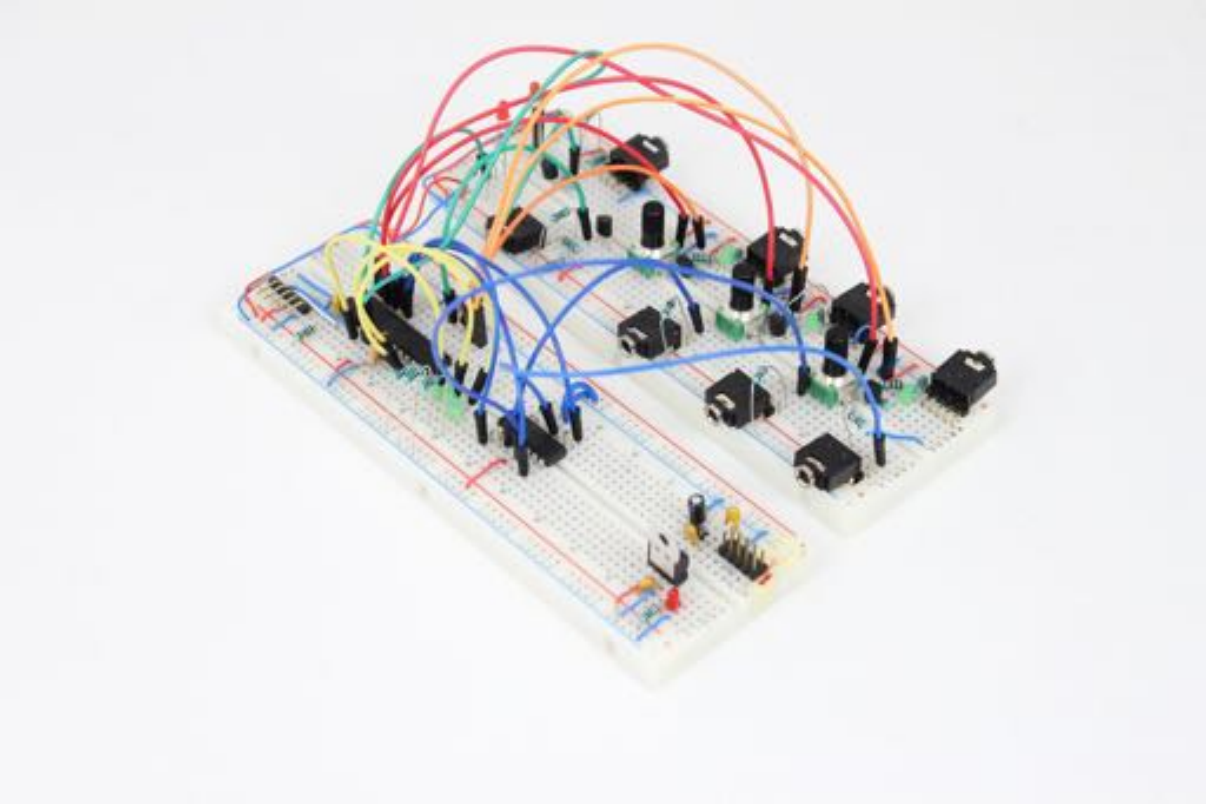


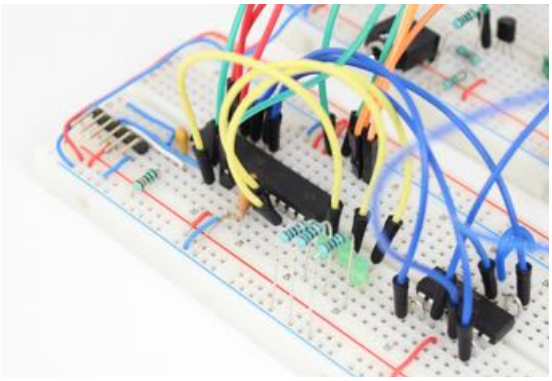
1x Resistenza 100k (R16);
1x IC Amp. Op. MCP6002 (IC3B);
1x Jack Mono 3,5mm (CV_IN);

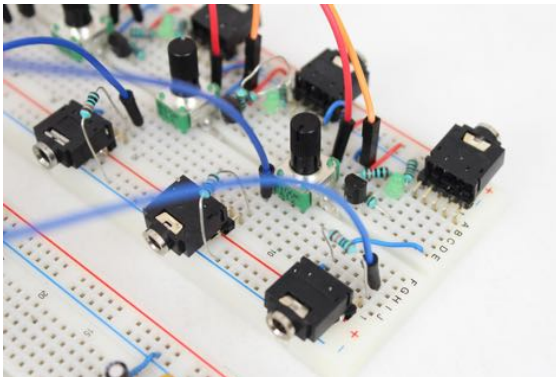
2.1 Implementazione dei diagrammi sulla breadboard

La *breadboard* è uno strumento di prototipazione di circuiti elettronici che offre il vantaggio di poter connettere le componenti senza che esse vengano “saldate” fra di loro. Le connessioni fra le parti sono temporanee e possono essere cambiate rapidamente tramite dei cavi chiamati “jumper”. Per questa sua caratteristica, la breadboard è lo strumento ideale per testare il proprio circuito e apportare modifiche durante le fasi di progettazione.

Dopo aver preso in esame i diagrammi elettronici e le componenti necessarie, si può procedere con l'assemblaggio del circuito sulla breadboard.
La prima versione del prototipo presentato in questa tesi è composta da tre ingressi CV, un ingresso Clock, un ingresso Reset, tre uscite Trigger e tre potenziometri per il controllo dei parametri interni.







2.2 Programmazione del Microcontrollore

Dopo aver cablato le componenti elettroniche sulla breadboard si procede con la programmazione del microcontrollore nell'ambiente Arduino.
Poiché nella fase di "concept" lo strumento è stato realizzato sul software Max, in questo capitolo viene trattata la "trasposizione" da un linguaggio di programmazione verso l'altro.

Arduino IDE

La schermata principale di Arduino, ossia dove si scrivono le *istruzioni* per l'hardware, si chiama "sketch" ed è costituito da diverse "parti" di codice che possono essere distinte in: *commenti*, *variabili*, *funzioni*, *setup* e *loop*, ciascuna delle quali contiene delle informazioni specifiche che possono mutare a seconda dell'ordine e della sintassi utilizzata.

Nei passaggi successivi verranno mostrate diverse sezioni del codice, tratte dallo strumento in analisi, che serviranno ad identificare l'algoritmo di generazione del pattern probabilistico, della sua esecuzione e della configurazione degli ingressi e delle uscite.

Algoritmo del sequencer probabilistico

Per semplificare la lettura del codice, si inizia col riportare la parte relativa alle variabili, ossia le informazioni da inserire prima della sezione "void setup".

In questa sezione vengono definiti i parametri dell'algoritmo per nome e tipologia (intero, booleano, matrice…), preceduti da un commento (//); lo scopo di questa parte è quello di definire dei parametri o delle funzioni che possono essere richiamate e modificate nelle funzioni "void setup" e "void loop", senza bisogno di essere ridichiarate:

```
//Probability sequencer algorithm

//define the pattern as an array of 8 bool values:
bool patternArray[8] = {0, 0, 0, 0, 0, 0, 0, 0};

//define an array of 8 random number generators:
int randomArray[8] = {0, 0, 0, 0, 0, 0, 0, 0};

//define the current step number
int stepCounter = 7;
int stepPosition = 0;

//step active (1 = true) or inactive (0 = false)
bool stepValue = 0;

//value of probability read
int probability = 0;

// void setup() { [...]
```

La sezione successiva contiene delle “istruzioni” che il microcontrollore eseguirà solamente una volta, precisamente al suo avvio o “accensione”.

Il primo comando immesso (Serial.begin(9600);) serve ad aprire la comunicazione seriale tra microcontrollore e computer; questo passaggio è indispensabile per configurare l’hardware tramite l’Arduino IDE e per monitorare i parametri tramite il “serial monitor”:

```
void setup() {
  //enable serial communication
  Serial.begin(9600);
}

// void loop() { [...]
```

La parte finale dello “sketch” ossia la funzione “void loop”, contiene le istruzioni che il microcontrollore ripeterà “ciclicamente” dal momento in cui viene acceso sino a quando verrà spento. All’interno di questa sezione si andrà dunque a configurare l’algoritmo del sequencer, secondo i parametri dichiarati nella prima sezione.

In questa prima parte viene mostrato come creare un contatore che andrà a leggere il relativo step nella matrice del pattern (patternArray) e successivamente, stampare il suo valore (stepValue) nel serial monitor dell'Arduino IDE:

```
void loop() {
 //advance the sequence to the next step
 stepCounter++;

 //go back to the first step (0) at the end of the sequence
 if(stepCounter > 7) {
  stepCounter = 0;
 }

 //define which step to read from the pattern array
 stepPosition = stepCounter;
 stepValue = patternArray[stepPosition];

 //print on the serial monitor current step (active or inactive = 1 or 0)
 Serial.println(stepValue);
```

Quest'ultima parte di codice viene invece eseguita quando il contatore raggiunge l'ultimo step (stepCounter = 7) ed inizia con la funzione "Analog.Read" che legge il valore del potenziometro (probability).
Successivamente, viene generata una matrice di numeri casuali (random(1023)), precisamente uno per ogni step; la matrice viene poi confrontata con il valore del potenziometro (probability) e per ogni numero casuale che risulti inferiore alla soglia di probabilità (if(randomArray < probability)), il rispettivo step sarà salvato come "acceso" (patternArray[i] = 1) altrimenti come spento (patternArray[i] = 0).
Il codice termina con la funzione "delay" che abilita il ciclo "void loop" una volta ogni 250 millisecondi:

```
//generate new pattern at the end of the sequence
 if(stepCounter = 7) {
   //read value from potentiometer
   probability = analogRead(A5);

     //generate array of random numbers
     randomArray[0] = random(1023);
     randomArray[1] = random(1023);
     randomArray[2] = random(1023);
     randomArray[3] = random(1023);
```

```
    randomArray[4] = random(1023);
    randomArray[5] = random(1023);
    randomArray[6] = random(1023);
    randomArray[7] = random(1023);

    //define pattern (if random number < probability then step = 1)
    for(int i=0; i <=7; i++) {
      if(randomArray[i] < probability) {
        patternArray[i] = 1;
      } else {
        patternArray[i] = 0;
      }
    }
  }

  //advance sequence to next step every 250 milliseconds
  delay(250);
}
```

Una volta implementato questo sketch e accertato il suo funzionamento, il passo successivo è quello di controllare il sequencer tramite impulsi esterni e a sua volta generare un impulso in uscita per ogni step attivo.

Gli ingressi di tipo trigger, gate o clock possono essere assegnati ai *pin digitali* del microcontrollore. Questi pin si limitano a rilevare la presenza o l'assenza di un segnale (on/off) tuttavia, nel mondo analogico, i segnali non sono mai stabili e possono avere delle micro-fluttuazioni che causerebbero risultati diversi dal previsto, a seconda della configurazione.

Per questo motivo, all'interno dello sketch di Arduino, bisogna creare delle istruzioni su come interpretare i diversi "stati" del segnale. Nella parte relativa alle variabili si definisce il pin digitale dedicato alla ricezione del *clock* da un modulo esterno (#define clockInput PINB&B00100000); le altre variabili da dichiarare sono la "soglia di fluttuazione" del segnale ricevuto (#define bounce 5) e i parametri contenuti all'interno della funzione "void clockDetect" che determinano lo stato di "clock":

```
//INPUT DEBOUNCING

//port mapping clock input pin PB5
```

```
#define clockInput PINB&B00100000

//state bouncing threshold
#define bounce 5

//clock detection variables
bool clockBounce;
bool clockState;
byte clockBounceCounter;

//detect only one clock signal for every pulse received
void clockDetect() {
 clockState = false;
 if(!(clockInput)) {clockBounce = true;}
 if(clockBounce) {
  if(clockInput) {
   clockBounceCounter++;
  } else {
   clockBounceCounter = 0;
  }
  if(bounce == clockBounceCounter) {
   clockState = true;
   clockBounce = false;
  }
 }
}

// void setup() { [...]
```

Una volta stabilito il pin dedicato al clock, è necessario "registrarlo" come ingresso, all'interno della funzione "void setup" (DDRB = B11011111;). Questa operazione è chiamata "Port Manipulation" ed è descritta in dettaglio nella pagina "reference" del sito web di Arduino.

Nella sezione "void loop" è necessario abilitare la funzione di rilevamento del clock, semplicemente richiamandola (clockDetect();). Attraverso la funzione "Serial.println" è possibile monitorare lo stato del clock collegando un segnale esterno all'ingresso clock della nostra breadboard. Questa operazione va ripetuta per ogni ingresso digitale destinato alla ricezione di segnali esterni di tipo trigger/gate/clock.

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

```
  //set port PB5 as input
  DDRB = B11011111;
}

void loop() {

  //enable clock detection function
  clockDetect();

  //print the state of the clock (1 or 0)
  Serial.println(clockState);

  //just to make the serial monitor smoother
  delay(50);
}
```

Nell'esempio successivo viene riportata la procedura riguardante la programmazione dei segnali in uscita, seguendo la stessa logica adoperata per l'ingresso Clock. Per semplificare la comprensione del codice, in questo esempio si mostra come produrre un segnale di trigger in uscita per ogni segnale clock ricevuto dall'esterno.

Ripartendo dallo sketch precedente, si inizia col definire il pin dedicato all'uscita trigger (`#define trigOut0 PINB&B00001000`), viene poi stabilita la durata dell'impulso in millisecondi (`#define trigLength 5`) e infine viene introdotta una variabile che rappresenta il tempo cronometrico dell'ultimo stato di Clock (`unsigned long lastClock = 0;`):

```
//TRIGGER OUTPUT

//port mapping clock input pin PB5
#define clockInput PINB&B00100000
//port mapping trigger output pin PB3
#define trigOut0 PINB&B00001000

//state bouncing threshold
#define bounce 5

//length of trigger signal (ms)
#define trigLength 5

//clock detection variables
bool clockBounce;
bool clockState;
```

```
byte clockBounceCounter;

//keep time of last clock State
unsigned long lastClock = 0;

// void clockDetect() { [...]

// void setup() { [...]
```

Nella funzione void setup viene registrato il pin PB3 come output (DDRB = B11011111;) e viene inoltre inizializzato come spento (PORTB|B00000001;).

All'interno del ciclo void loop, ogni qualvolta viene rilevato lo stato di clock (if(clockState) {) il tempo attuale viene salvato all'interno della variabile (lastClock = millis();) e si imposta il pin in uscita come acceso (PORTB|=B00001000;); ad ogni ciclo viene misurato il tempo trascorso dall'ultimo clock e nel caso sia maggiore della durata dichiarata per il trigger (if(millis() >= lastClock + trigLength) {), il segnale in uscita dal pin viene interrotto (PORTB=B00000001;):

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

 //set port PB5 as input and PB3 as output
 DDRB = B11011111;
 //set port PB3 output as LOW
 PORTB|B00000001;
}

void loop() {

 //enable clock detection function
 clockDetect();

 //when a clock signal is detected:
 if(clockState) {
  //save time of last clock state
  lastClock = millis();
  //set pin PB3 as HIGH (begin of trigger signal)
  PORTB|=B00001000;
 }

 //set pin PB3 as LOW after it reaches its length (end of trigger signal)
 if(millis() >= lastClock + trigLength) {
  PORTB=B00000001;
 }
```

}
Assemblando gli esempi riportati, con le dovute integrazioni delle sorgenti di input ed output, si ottiene lo sketch completo del modulo, pronto ad essere interfacciato con altri sistemi modulari Eurorack e affinato al proprio scopo.
Le parti di codice illustrate sono disponibili online per essere consultate e testate come sketch indipendenti al link:

https://github.com/stziopa/kompas/tree/master/arduino

3. Primo prototipo dello strumento

Il circuito elettronico implementato sulla breadboard rappresenta il punto di partenza dal quale capire i margini di perfezionamento dello strumento in termini di operatività, interfacciabilità con altri moduli e stabilità nel funzionamento.
Una volta accertati questi margini ed apportate le eventuali modifiche, la configurazione hardware deve essere consolidata all'interno del sistema modulare, rispettando le specifiche tecniche dello standard e pensando alla disposizione dei controlli tenendo conto degli aspetti performativi.
Nello Standard Eurorack, le dimensioni dei pannelli rispettano delle unità di misura specifiche, pensate appositamente per garantire la compatibilità fra i diversi costruttori. La lunghezza verticale è fissata a 3U (1U = 44.45mm; 3U = 133.4 mm), quella orizzontale si misura in HP (1HP = 5.08mm) e procede per multipli di 1 a seconda della grandezza dell'interfaccia; la profondità non ha un'unità dedicata ma tipicamente varia da 30 a 130mm.

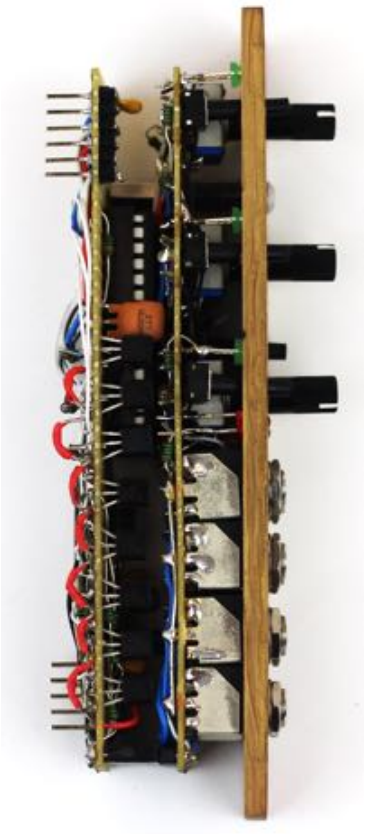
3.0 Assemblaggio su perfboard

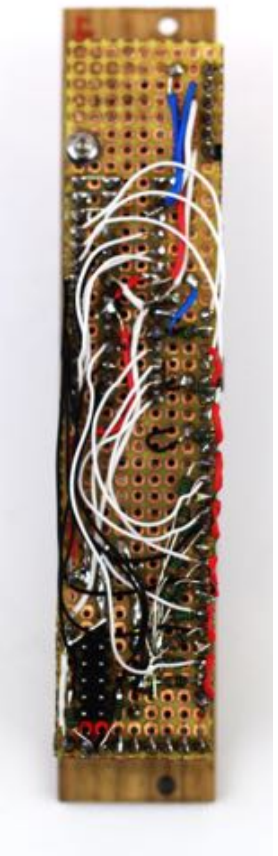
Il circuito realizzato precedentemente sulla breadboard deve essere riprodotto, nella maniera più fedele possibile, in una scheda di prototipazione chiamata *"perfboard"*. L'impiego della scheda perforata non è un passaggio indispensabile, a seconda del grado di esperienza nella progettazione di circuiti,

tuttavia, in questo capitolo è riportato il suo impiego in quanto rappresenta il mezzo meno costoso e relativamente semplice da modificare in caso di errori nel cablaggio, al contrario della PCB come verrà affrontato nel capitolo successivo.

Nel caso di questo modulo, la cui versione finale su breadboard consiste di 8 connettori ingresso/uscita e 3 potenziometri, si è deciso di sfruttare il pannello del modulo Noise Square della Bastl Instruments, in quanto utilizza lo stesso numero di controlli ed, essendo in legno, può essere all'occorrenza tagliato e perforato con attrezzi di semplice reperibilità.

Le immagini seguenti mostrano la veduta frontale, laterale e posteriore della prima implementazione del modulo sulla perfboard.





4. Strumento finale

Per ottenere uno strumento efficiente la regola dovrebbe essere quella di "comprimere" l'hardware nei minimi HP possibili, senza però compromettere l'ergonomicità dei controlli. Il vantaggio di un avere un modulo dalle dimensioni ridotte è principalmente quello di poter risparmiare risorse e ottenere il massimo in termini di portabilità ed efficienza del sistema modulare.

Per realizzare la versione finale di uno strumento che possa essere prodotto in più esemplari della stessa fattura è necessario consolidare il primo prototipo all'interno di un formato che garantisca la sua riproducibilità. Attraverso l'utilizzo di software CAD dedicati al design di circuiti elettronici è possibile disegnare il circuito dello strumento ed esportarlo in un formato che consenta la produzione della PCB presso aziende specializzate.

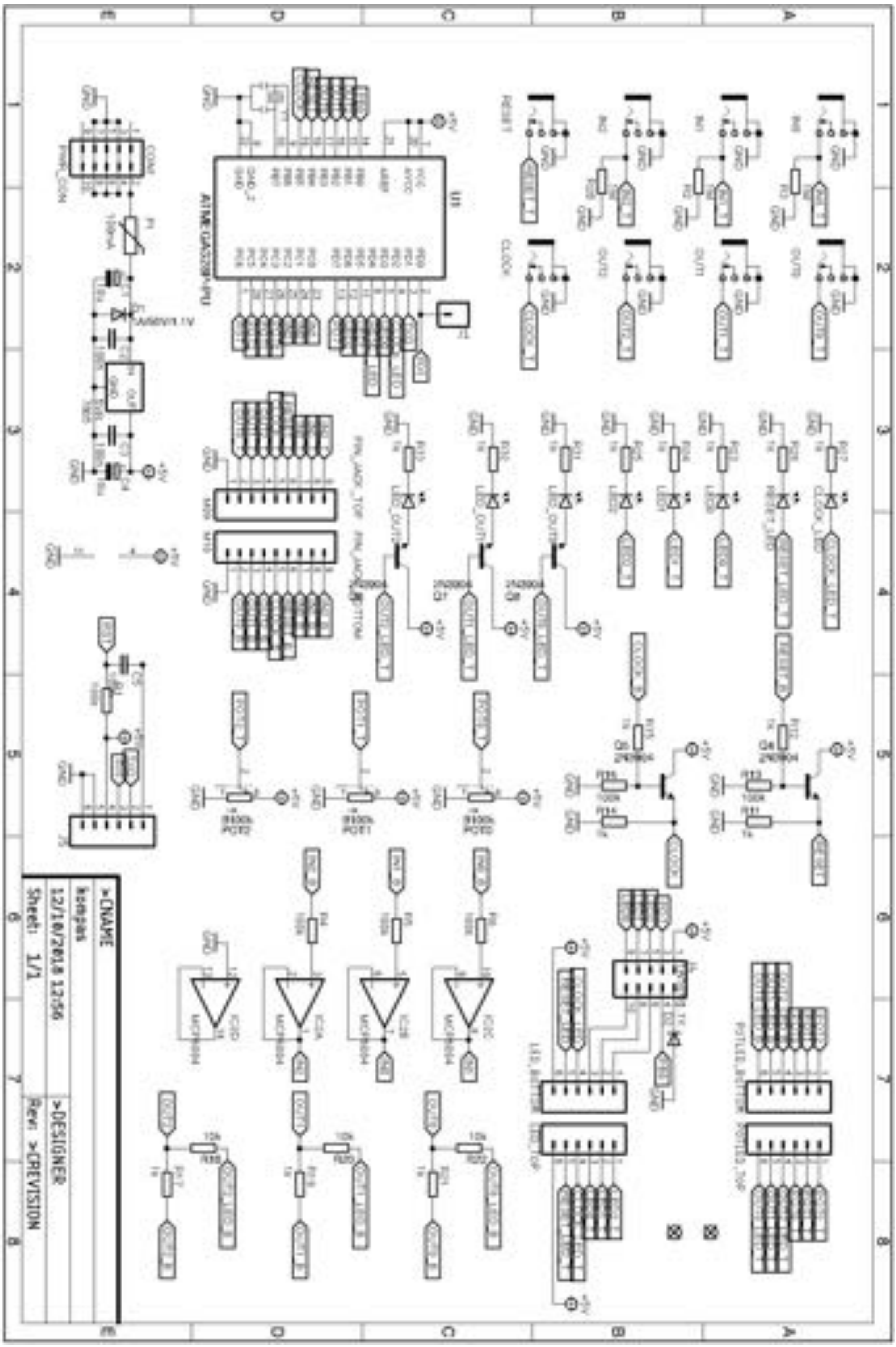
4.0 Progettazione su Eagle

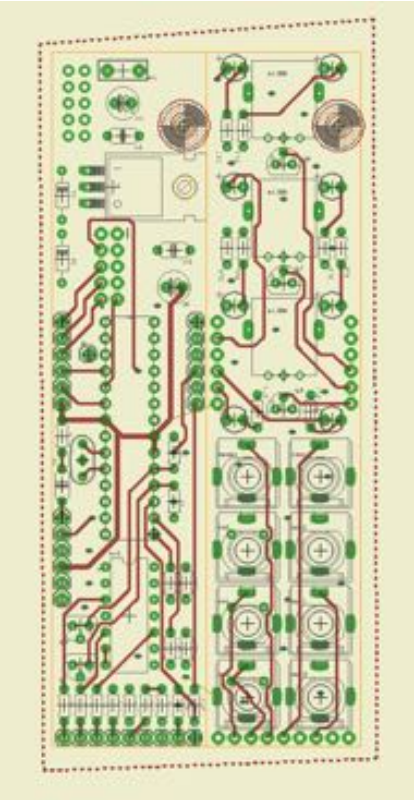
Per realizzare la PCB di questo modulo è stato utilizzato il software Eagle della Autodesk. All'interno del software si inizia col disegnare il diagramma elettronico, basandosi sul cablaggio effettuato precedentemente nella breadboard. La maggior parte delle componenti elettroniche, utilizzate nel prototipo, sono presenti nel pacchetto di Eagle mentre le altre possono essere scaricate come librerie aggiuntive dal sito web della Sparkfun Electronics al link:

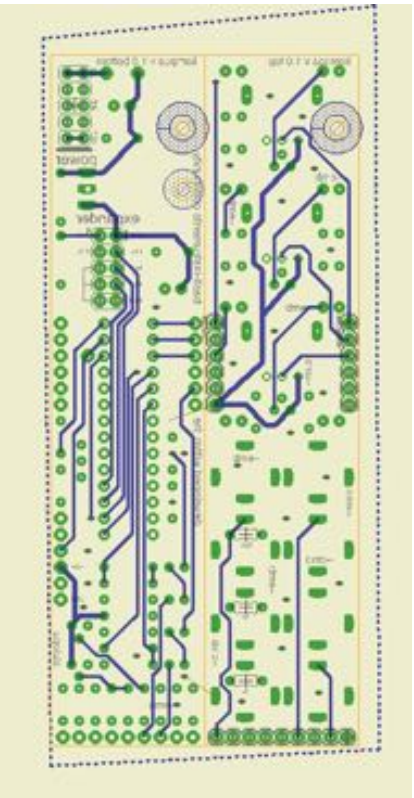
https://www.sparkfun.com/pages/eagle_lib_lightbox

Dopo aver disegnato il diagramma elettronico si procede con il design della scheda, il cui circuito deve essere ridimensionato in accordo con lo standard Eurorack.

Nelle pagine seguenti vengono riportati il diagramma elettronico e i disegni della PCB.


>CNAME
>DESIGNER
Sheet: 1/1
Rev: >CREVISION
ATMEGA328P-PU

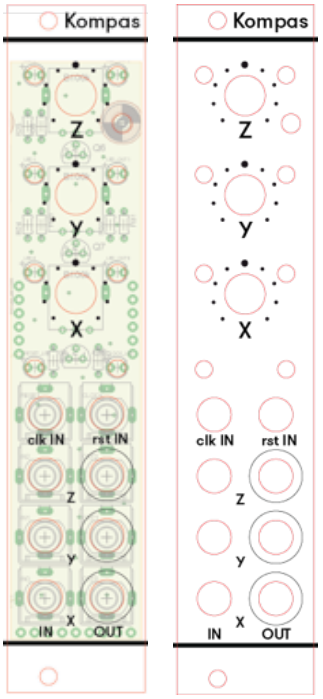




4.1 Design del pannello frontale

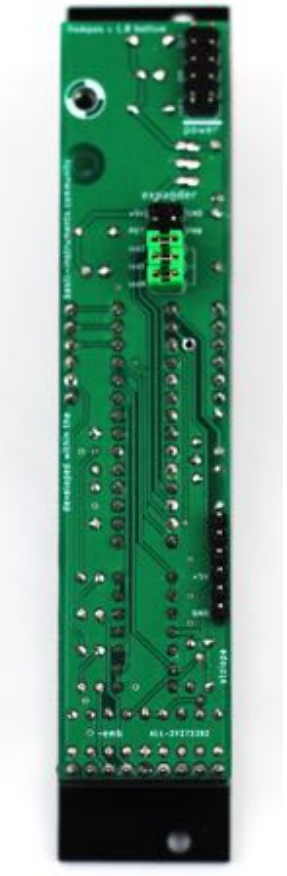
Per la versione finale del modulo, il pannello frontale è stato disegnato sul software Adobe Illustrator con la finalità di poter essere stampato attraverso una macchina dedicata al taglio laser dei pannelli acrilici.

All'interno del disegno, si specificano le aree del pannello da tagliare (in rosso: dimensioni, alloggio per potenziometri, connettori jack e led) e quelle da scolpire (in nero: grafiche e testi). Questi codici di colore permettono alla stampante di distinguere le aree di lavoro e applicare il tipo stampa opportuno.
La scheda disegnata su Eagle può essere esportata in formato .dxf (Drawing Interchange Format) ed essere utilizzata come livello all'interno di Illustrator.



4.2 Assemblaggio dello strumento finale

Dopo aver completato i passaggi affrontati nei capitoli precedenti, il modulo è pronto per essere presentato nella sua versione finale e si presenta come nelle immagini seguenti:



Conclusioni

Nella stesura di questa tesi accademica, la documentazione del procedimento creativo, mi ha posto davanti a delle problematiche legate al trasferimento di nozioni, che talvolta, non possono essere comprese se non attraverso la pratica delle stesse. Nelle fasi di “razionalizzazione” dei procedimenti, sono stato costretto a mantenere una direzione generica, evitando di addentrarmi in territori specializzati che richiederebbero l’analisi e la valutazione da parte di altri settori di competenza. Operando sempre con la massima onestà intellettuale, ho cercato di elaborare un testo che possa essere una risorsa utile a chi, come me, provenendo da un Conservatorio di Musica - indirizzato alla Musica Elettronica, volesse orientare la propria ricerca verso lo studio degli strumenti musicali elettronici.

Ringraziamenti

Fra le persone coinvolte, in maniera diretta o indiretta, nella realizzazione di questa tesi vorrei ringraziare in primis la mia famiglia per il supporto incessante durante questi anni di studio, i miei docenti Daniele Ledda, Alessandro Olla e Fabrizio Casti, Ondřej e Václav per avermi accolto nella loro comunità e avermi fatto assorbire esperienza dal loro lavoro e dalle loro visioni, tutti i miei amici, Emanuele e Rachele, Claudio, Alberto, Danilo e Travis, Lisa e Marina, Pavel e la sua famiglia, Petr, Martin, Peter, David, John, František e tutto il collettivo Bastl.

Elenco delle fonti

Testi:

ENRICO COSIMI, *Manuale di Musica Elettronica*. *Teoria e Tecnica dei Sintetizzatori*, Tecniche Nuove, 2011.

JULIEN BAYLE, *C Programming for Arduino*, Packt Publishing, 2013.

NICK COLLINS, JULIO D'ESCRIVAN, *The Cambridge Companion to Electronic Music*, Cambridge University Press, 2011.

PAUL HOROWITZ, WINFIELD HILL, *The Art of Electronics (3rd edition)*, Cambridge University Press, 2015.

Documentari:

ROBERT FANTINATTO, *I dream of Wires*, 2014.

Siti web:

doepfer.de
bastl-instruments.com
wiring.org
cdm.link
musicradar.com
theransomnote.com
keyboardmag.com
120years.net