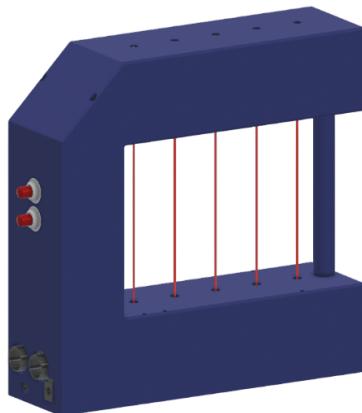


**Report progetto Laboratorio di Making
Università di Bologna
A.A. 2024-25**

MIDI Laser-Harp



Leonardo Vorabbi
0001186597

Email

leonardo.vorabbi@studio.unibo.it

Indice

1	Introduzione	2
1.1	Contesto del progetto	2
1.2	Controller e protocollo MIDI	2
1.3	Obiettivo del progetto	3
2	Stato dell'arte	3
2.1	Prodotti esistenti	3
2.2	Analisi progetti online	3
3	Materiali e componenti utilizzati	5
3.1	Tabella dei componenti elettronici	5
3.2	Tabella dei componenti di supporto	6
4	Progettazione	6
4.1	Prototipazione su breadboard	6
4.1.1	Schema del circuito preliminare	6
4.1.2	Codice Arduino per calibrazione e lettura dei sensori .	7
4.2	Comunicazione MIDI	8
4.2.1	Codice Arduino per comunicazione MIDI	8
4.2.2	Test con Ableton	10
4.3	Assemblaggio e saldatura	10
4.3.1	Trasferimento del circuito dalla breadboard a stripboard	10
4.4	Progettazione e modellazione 3D	11
4.4.1	Preparazione e misurazione	11
4.4.2	Disegno CAD e stampa FDM	11
4.5	Assemblaggio finale	12
4.5.1	Difficoltà riscontrate	12
4.6	Implementazione software di configurazione	14
5	Conclusioni	15
5.1	Risultati ottenuti	15
5.2	Possibili miglioramenti	15
6	Appendice	16
6.1	Codici sorgente	16
6.1.1	Codice Arduino	16
6.1.2	Codice software di configurazione	21
6.2	Immagini	28

1 Introduzione

1.1 Contesto del progetto

Negli ultimi anni il campo degli strumenti musicali elettronici ha conosciuto una rapida evoluzione, grazie alla diffusione di controller MIDI sempre più vari e innovativi. Accanto a tastiere, pad e superfici di controllo tradizionali, sono comparsi strumenti alternativi che sfruttano diverse tecnologie per offrire modalità di interazione più intuitive ed espressive.

Tra i marchi più innovativi nel settore dei controller MIDI spicca ROLI, che ha introdotto soluzioni capaci di ampliare notevolmente le possibilità espressive del musicista. Con la serie Seaboard, ROLI ha portato sul mercato superfici sensibili alla pressione e al movimento, basate sul protocollo MPE (MIDI Polyphonic Expression), che consentono di modulare parametri sonori in tempo reale con un approccio molto più fluido rispetto alle tastiere tradizionali. A queste si affiancano dispositivi come l'Airwave, che sfruttano la visione artificiale per interpretare i gesti del performer e tradurli in controlli musicali, aprendo la strada a un'interazione più naturale e gestuale con il suono.

In questo panorama si colloca il progetto MIDI Laser-Harp, che si ispira alle celebri arpe laser utilizzate in ambito performativo da artisti come Jean-Michel Jarre, ma con un approccio più accessibile e compatto. L'obiettivo è creare un dispositivo che permetta all'esecutore di interagire gestualmente con fasci di luce, traducendo l'interruzione dei laser in segnali MIDI compatibili con software musicali e sintetizzatori.

Il progetto si inserisce quindi nel filone della ricerca di nuove interfacce uomo-macchina per la musica elettronica, ponendo l'accento sull'aspetto performativo e sulla libertà gestuale.

1.2 Controller e protocollo MIDI

Un controller musicale non produce suoni autonomamente, ma invia comandi a un generatore sonoro esterno, che può essere un sintetizzatore hardware o un software di produzione musicale. Questi comandi sono trasmessi tramite il protocollo MIDI (Musical Instrument Digital Interface), uno standard introdotto negli anni '80 che consente a strumenti e computer di comunicare tra loro. Il MIDI non trasporta il suono, ma una serie di istruzioni digitali, come "nota ON/nota OFF", che vengono interpretate dal dispositivo ricevente per generare o modulare il suono. In questo modo, un controller come

l'arpa laser può diventare un'interfaccia che permette al musicista di suonare virtualmente strumenti molto diversi tra loro e delegando la sintesi del suono al dispositivo/software ricevente.

1.3 Obiettivo del progetto

Obiettivo del progetto è la realizzazione di un'arpa laser MIDI. L'intento è sviluppare uno strumento musicale alternativo capace di trasformare l'interruzione di fasci laser in segnali MIDI standard, in modo da poter interagire facilmente con software di produzione musicale e sintetizzatori hardware. La progettazione include non solo l'elettronica di rilevamento e comunicazione, ma anche la creazione di un contenitore stampato in 3D che integri in maniera ordinata laser, sensori e scheda di controllo. L'obiettivo finale è fornire un dispositivo completo che possa fungere da base per future estensioni.

2 Stato dell'arte

2.1 Prodotti esistenti

A seguito di una ricerca online, ho individuato un solo prodotto attualmente in vendita che possa essere definito a tutti gli effetti un'arpa laser: si tratta del modello proposto su laser-harp.com, commercializzato a 999 €. Questo strumento, tuttavia, si discosta molto dal progetto qui presentato: è concepito principalmente come dispositivo scenografico, pensato per spettacoli e performance dal forte impatto visivo, ma con funzionalità musicali piuttosto limitate e un costo che lo rende poco accessibile.

In parallelo esistono altri controller MIDI alternativi basati su sensori di movimento o di prossimità. Tra questi, ad esempio, il Leap Motion e l'Airwave di ROLI (basati su sensori ottici capaci di rilevare con precisione i gesti delle mani nello spazio). È importante sottolineare però che questi dispositivi sono accomunati all'arpa laser soltanto dall'obiettivo di rendere l'interazione con il suono più naturale e immediata rispetto a tastiere o controller tradizionali, offrendo tuttavia interfacce gestuali diverse.

2.2 Analisi progetti online

Online sono disponibili diversi progetti di arpe laser, come ad esempio <https://www.instructables.com/Laser-Harp-2/>. La maggior parte di essi si basa su un unico laser di potenza relativamente elevata (> 50 mW), il cui fascio

viene deviato da uno specchio montato su un motore passo-passo che ruota ad alta velocità. In questo modo si ottiene l'illusione di avere più raggi paralleli, corrispondenti a corde virtuali. Un LDR misura la luminosità e l'eventuale interruzione del fascio viene associata a una determinata nota, in base alla posizione del motore durante l'interruzione. Questo approccio, seppur scenografico, presenta diversi limiti: il costo del laser, la scarsa portabilità, il rischio per gli occhi dovuto all'uso di raggi potenti e la riduzione del controllo musicale a un semplice on/off per nota.

Il progetto sviluppato adotta invece una logica differente: utilizza più laser indipendenti a bassa potenza ciascuno allineato con un proprio LDR dedicato. Questa soluzione risulta meno spettacolare dal punto di vista scenico, ma molto più funzionale e sicura, perché permette la realizzazione di un frame stampato in 3D in cui possono essere integrati altri sensori e controlli aggiuntivi, rendendo il sistema più versatile. Tra i limiti rimangono la necessità di un allineamento preciso tra laser e sensori e una resa visiva meno scenica rispetto alle soluzioni con fasci riflessi.

3 Materiali e componenti utilizzati

3.1 Tabella dei componenti elettronici

Categoria	Componente	Quantità	Note
Scheda di controllo	Arduino Leonardo (+ protoboard)	1	Compatibile con MIDI via USB grazie a ATmega32u4
Laser	HiLetgo 5 mW 650 nm Red DOT Laser Module (KY-008)	5	Moduli laser 5 V, luce rossa, 650 nm
Sensori ottici	LDR	5	Per rilevare l'interruzione del fascio laser
Resistenze	220 Ω	10	5 per le LDR, 5 per i LED
LED	LED 5 mm	5	Indicatori visivi per ogni "corda"
Sensori addizionali	Sensore a ultrasuoni	1	Per rilevare la distanza delle mani
Pulsanti	Pulsanti standard	2	Per cambio ottava
PCB	Schede perforate	2	Per assemblaggio e saldatura
Cablaggio	Jumper	vari	

Tabella 1: Componenti elettronici principali

3.2 Tabella dei componenti di supporto

Categoria	Componente	Quantità	Note
Assemblaggio	Viti M3 autofilettanti	varie	Per fissaggio delle parti stampate in PLA
Strumenti di supporto	“Terza mano” con pinze	1	Per facilitare le saldature
Materiali stampa 3D	Filamento PLA	variabile	Per la realizzazione del ”frame”
Stampante 3D	Ultimaker 3	1	Stampa del ”frame”
Software CAD/CAM	Fusion 360, Ultimaker Cura	—	Modellazione e slicing per stampa 3D

Tabella 2: Componenti di supporto e strumenti

4 Progettazione

4.1 Prototipazione su breadboard

4.1.1 Schema del circuito preliminare

Per la fase di prototipazione è stato realizzato un primo circuito di test su breadboard con lo scopo di verificare il corretto funzionamento del principio di base: l'interruzione di un fascio laser rilevata da un fotoresistore (LDR) viene tradotta in un segnale digitale visibile tramite l'accensione di un LED.

Nel prototipo sono stati utilizzati:

- Due moduli laser (650 nm, 5 mW), collegati direttamente all'alimentazione a 5V fornita da Arduino Leonardo.
- Due LDR, ognuno in partitore resistivo con una resistenza da $220\ \Omega$. Il punto centrale del partitore veniva letto da un ingresso analogico di Arduino per rilevare la variazione di luminosità.
- Due LED da 5 mm, ciascuno con una resistenza da $220\ \Omega$ in serie, collegati alle uscite digitali di Arduino. I LED fungevano da indicatori visivi dello “stato della corda”: spenti quando il fascio era attivo e accesi quando il fascio veniva interrotto.

- Due pulsanti momentanei, inizialmente collegati agli ingressi digitali di Arduino. La loro funzione era ancora in fase di definizione, ma sarebbero poi stati utilizzati per il controllo dell'ottava tramite messaggi MIDI.
 - Un sensore a ultrasuoni, aggiunto per future sperimentazioni: lo scopo era valutare la possibilità di utilizzarlo come ulteriore parametro di controllo MIDI (per esempio, modulare un effetto in base alla distanza della mano).

Il prototipo realizzato [figura 4] ha permesso di testare la sensibilità degli LDR rispetto ai moduli laser e di iniziare a scrivere il codice per la calibrazione degli LDR e la trasmissione dei vari segnali MIDI.

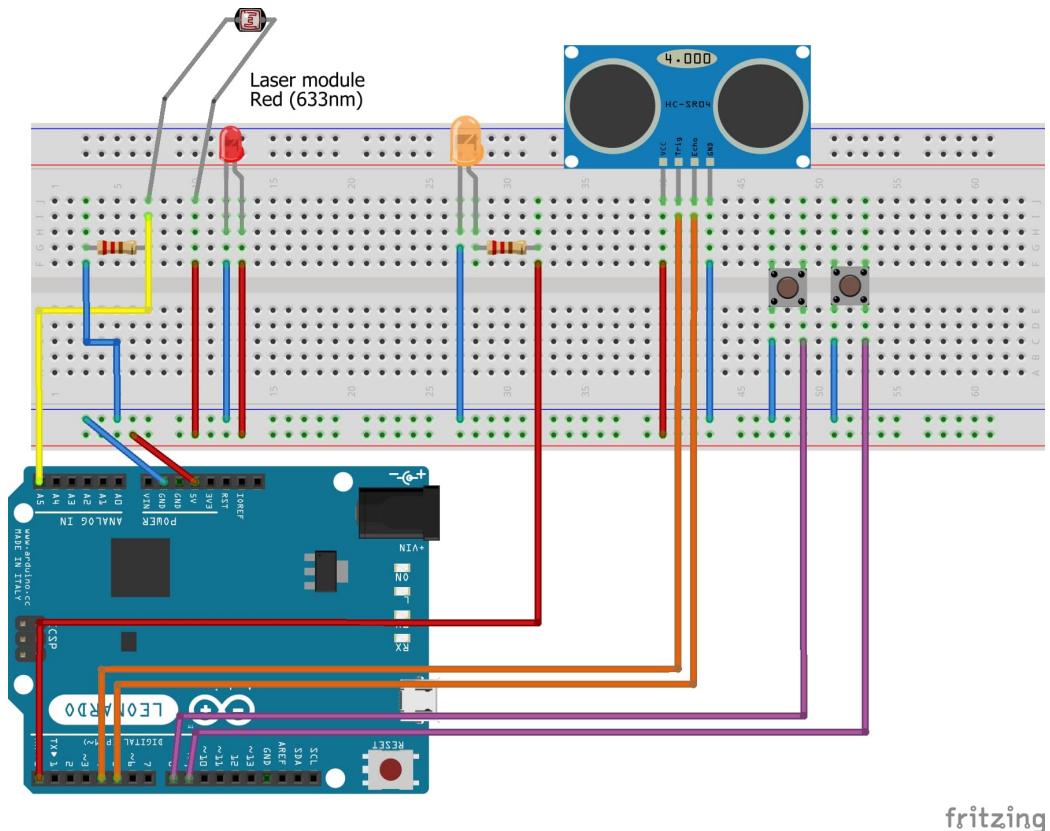


Figura 1: Schema circuito considerando un solo fascio laser

4.1.2 Codice Arduino per calibrazione e lettura dei sensori

Al fine di ridurre al minimo l'impatto della luce ambientale, variabile a seconda del contesto di utilizzo, sulla lettura dei sensori, 'è stata implementata

una procedura di calibrazione automatica dei singoli LDR. Durante la fase di setup, ciascun sensore viene campionato per 100 letture consecutive mentre il relativo fascio laser ‘e attivo. La media di questi valori viene utilizzata per calcolare una soglia dinamica (threshold), leggermente inferiore alla luminosità rilevata in condizioni normali. In questo modo il sistema ‘e in grado di adattarsi alle variazioni di luce ambientale e ridurre gli errori di lettura. Nel ciclo principale, i valori analogici provenienti dagli LDR vengono confrontati con la soglia calcolata. Quando il valore scende al di sotto della soglia, significa che il fascio ‘e stato interrotto, e il sistema attiva la nota corrispondente, mentre quando il valore torna al di sopra della soglia, la nota viene spenta. Questo meccanismo consente di rilevare in tempo reale l’interruzione dei fasci laser e di associare ogni “corda” luminosa a una specifica nota MIDI. L’aggiunta di LED di feedback collegati a ciascun sensore permette inoltre di verificare visivamente l’attivazione e il rilascio delle note durante i test su breadboard.

Sebbene nel prototipo fossero già presenti i due pulsanti e il sensore a ultrasuoni, in questa fase preliminare il codice si è concentrato esclusivamente sulla calibrazione e sulla lettura degli LDR, rimandando l’implementazione delle funzionalità avanzate alla fase successiva.

Il codice completo è disponibile al link [github](#) del progetto oppure cliccando [qui](#).

4.2 Comunicazione MIDI

4.2.1 Codice Arduino per comunicazione MIDI

Per permettere all’arpa laser di interagire con software musicali e strumenti esterni, è stata implementata la comunicazione MIDI tramite la libreria **MIDIUSB** di Arduino. Il MIDI (Musical Instrument Digital Interface) è un protocollo digitale standardizzato che non trasmette audio, ma messaggi di controllo: istruzioni come “Nota On”, “Nota Off”, “Control Change” o “Program Change” che descrivono come uno strumento deve generare o modificare un suono.

Ogni messaggio MIDI tradizionale è composto da tre byte:

1. **Status byte**: indica il tipo di messaggio e il canale MIDI (1–16).
2. **Data byte 1**: numero della nota o ID del controller.
3. **Data byte 2**: velocità della nota o valore del controller (0–127).

Quando si utilizza la libreria **MIDIUSB**, questi tre byte vengono incapsulati in un pacchetto USB-MIDI a quattro byte (`midiEventPacket_t`), dove il primo byte (`cin`) indica al computer il tipo di messaggio USB-MIDI, mentre i successivi tre byte corrispondono ai classici messaggi MIDI.

La gestione avviene secondo due funzioni principali:

- **Nota ON:** quando il valore letto dall'LDR scende sotto la soglia di calibrazione, Arduino invia un messaggio Note On contenente il numero della nota corrispondente e la velocità massima (127). Contestualmente, viene inviato anche un messaggio di Control Change: questo non influisce sulla riproduzione musicale, ma viene utilizzato dal programma di configurazione dello strumento per aggiornare la grafica e monitorare lo stato dei sensori.
- **Nota OFF:** quando il valore torna sopra la soglia, Arduino invia il messaggio Note Off per la stessa nota, indicando così al software MIDI che il tasto virtuale è stato rilasciato. Anche in questo caso viene inviato un Control Change per aggiornare il monitoraggio visivo nell'interfaccia di configurazione.

L'elenco delle note assegnate agli LDR è gestito tramite un array (`noteBase[]`), che definisce la nota di riferimento per ciascun sensore. A questa nota di base può essere applicato uno **shift di ottava** (positivo o negativo), determinato dai pulsanti presenti sul prototipo.

Sensore	Nota MIDI	Nota musicale
LDR1	60	Do4
LDR2	62	Re4
LDR3	64	Mi4
LDR4	65	Fa4
LDR5	67	Sol4

Tabella 3: Note associate a ciascun LDR (Scala Do maggiore - prime 5 note)

Oltre ai messaggi di nota, il codice è predisposto anche per l'invio di **Control Change (CC)**, che permettono di mappare altri parametri musicali, come ad esempio il controllo di effetti.

Il codice completo è disponibile al link [github](#) del progetto oppure cliccando [qui](#).

4.2.2 Test con Ableton

Per testare il funzionamento musicale del prototipo, è stato utilizzato **Ableton Live**, una delle principali Digital Audio Workstation (DAW) che permette di registrare, riprodurre e modificare musica digitale, gestendo tracce audio e MIDI.

I primi test si sono rivelati relativamente semplici: è bastato collegare il prototipo al PC tramite USB, aprire Ableton, creare una traccia MIDI, selezionare uno strumento virtuale e attivarlo. Una volta fatto ciò, ogni interruzione di fascio laser corrispondeva all'invio di un messaggio Note On/Off, e il suono selezionato veniva riprodotto in tempo reale.

Per testare il controllo aggiuntivo fornito dal sensore a ultrasuoni, è stato sufficiente entrare in modalità *MIDI Map* di Ableton e selezionare il parametro desiderato da controllare (ad esempio il volume o un effetto di riverbero) associandolo al messaggio di Control Change inviato dal sensore. In questo modo il movimento della mano davanti al sensore modulava in tempo reale il parametro selezionato.

4.3 Assemblaggio e saldatura

4.3.1 Trasferimento del circuito dalla breadboard a stripboard

Dopo la fase di prototipazione su breadboard, è stato necessario realizzare un circuito più stabile e compatto, trasferendo il tutto su stripboard.

In particolare, sono stati saldati sulla stripboard il circuito dei fotoresistori e i relativi LED con resistenze, ovvero la sezione del circuito dedicata alla rilevazione delle interruzioni dei fasci laser e al feedback visivo, garantendo una maggiore robustezza.

Successivamente è stato saldato anche il sensore a ultrasuoni direttamente sulla protoboard di Arduino, insieme ad alcuni jumper necessari a facilitare i collegamenti con il resto del sistema.

Gli altri componenti (moduli laser, pulsanti, ecc.) non sono stati integrati direttamente sulla stripboard, ma fissati in altri modi sfruttando la struttura stampata in 3D.

Per completare l'intero circuito è stato inoltre necessario saldare numerosi cavi, al fine di unire tra loro le varie parti distribuite nel dispositivo.

4.4 Progettazione e modellazione 3D

4.4.1 Preparazione e misurazione

La progettazione della struttura dell'arpa laser è partita da un abbozzo a matita, utile per definire la disposizione generale dei componenti [figura 5]. Nello schema iniziale, l'Arduino Leonardo con il sensore a ultrasuoni era collocato in basso a sinistra, mentre i moduli laser occupavano la parte centrale e destra della base. La colonna verticale a sinistra era pensata come passaggio per il cablaggio verso la parte superiore, oltre che come alloggiamento dei pulsanti per il cambio di ottava. Nella parte superiore trovavano posto gli LDR e i LED: i primi rivolti verso il basso, in modo da allinearsi con i fasci laser, e i secondi orientati verso l'alto, per essere ben visibili all'utente.

Dopo aver definito questa disposizione, è stato necessario misurare accuratamente tutti i componenti elettronici per poterli riprodurre fedelmente in CAD e garantire un corretto alloggiamento.

4.4.2 Disegno CAD e stampa FDM

La modellazione è stata svolta con Autodesk Fusion 360 e ha rappresentato una delle fasi più impegnative del progetto, trattandosi della mia prima esperienza in modellazione 3D. È stato necessario non solo progettare i singoli pezzi, ma anche pensare a come renderli assemblabili e solidi, introducendo alloggiamenti e fori per le viti di fissaggio. Tutti i file STL prodotti sono disponibili nel repository GitHub: github.com/leovora/MIDI-laser-harp.

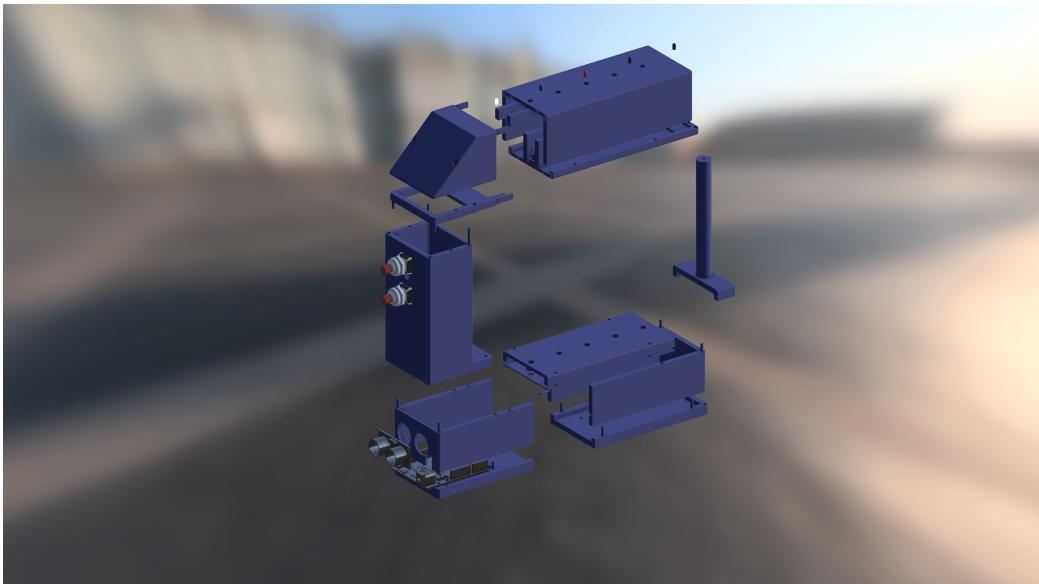


Figura 2: Mockup esploso con risalto sui singoli pezzi

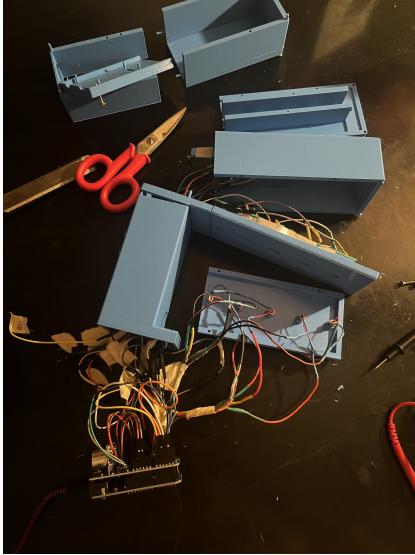
La stampa è stata effettuata con tecnologia FDM utilizzando Ultimaker Cura per il slicing e una Ultimaker 3 (ricevuta in prestito per un periodo limitato) per la realizzazione fisica dei componenti. Il processo è stato di natura fortemente iterativa: modellavo un pezzo, lo stampavo, e durante la stampa lavoravo al modello successivo. In diversi casi è stato necessario ristampare alcune parti dopo aver corretto errori di progettazione o piccole imprecisioni.

4.5 Assemblaggio finale

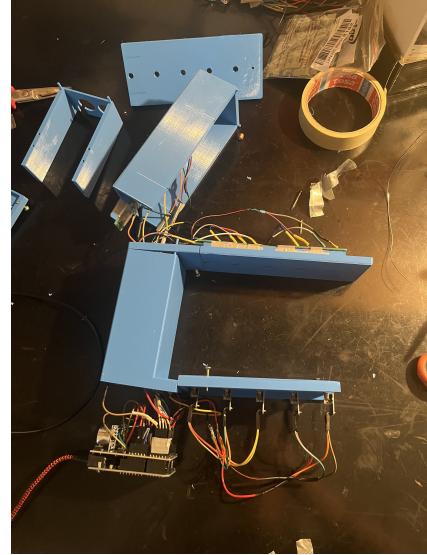
4.5.1 Difficoltà riscontrate

La fase di assemblaggio è stata senza dubbio quella in cui sono emerse le maggiori difficoltà, poiché solo mettendo insieme tutte le parti stampate e i componenti elettronici è stato possibile verificare concretamente eventuali errori progettuali.

Il primo problema rilevante è stato legato al cablaggio. Inizialmente avevo saldato ad ogni componente i propri collegamenti di alimentazione (5 V e GND), arrivando così a dover connettere oltre 30 cavi direttamente ad Arduino. Oltre alla complessità del cablaggio, il vero ostacolo era rappresentato dal fatto che tutti questi fili faticavano a passare all'interno della colonna sinistra. Per risolvere il problema ho dovuto tagliare e saldare molti cavi tra



(a) Prima delle correzioni



(b) Dopo le correzioni

Figura 3: Confronto prima e dopo correzioni assemblaggio

loro, riducendo notevolmente il numero di jumper e rendendo il circuito più gestibile.

Un'altra criticità è stata l'allineamento dei laser con gli LDR. Nonostante nel modello CAD i fori fossero perfettamente allineati, i moduli laser avevano troppo gioco all'interno delle sedi, causando disallineamenti. Ho quindi ristampato sia gli alloggi degli LDR, realizzando fori più larghi e conici per catturare il fascio anche in caso di leggera deviazione, sia la base dei laser, ridisegnata con fori calibrati sulle dimensioni reali dei moduli, per impedirne ogni movimento.

Nonostante queste correzioni, il problema non era del tutto risolto: i moduli laser stessi si sono rivelati poco affidabili. Ogni fascio aveva caratteristiche diverse (più largo, più stretto, inclinato) e, dopo pochi minuti di utilizzo, tendevano a surriscaldarsi, con conseguente calo di intensità luminosa. La soluzione definitiva è stata sostituirli con moduli di qualità superiore.

Superati questi ostacoli principali, il resto dell'assemblaggio è proceduto come previsto, consentendo di ottenere una struttura solida e un circuito ordinato e funzionale.

4.6 Implementazione software di configurazione

Poiché l'arpa laser dispone di soli 5 fasci luminosi, ho ritenuto importante sviluppare un software di configurazione che ne aumentasse la versatilità. L'applicazione è stata realizzata in Python, utilizzando la libreria Tkinter per l'interfaccia grafica e Mido per la gestione dei messaggi MIDI.

Il software svolge tre funzioni principali:

- **Configurazione manuale:** l'utente può assegnare liberamente una nota MIDI a ciascun laser.
- **Preset di scala:** è possibile selezionare una tonica e una scala musicale predefinita (maggiore, minore, pentatonica, blues, ecc.), che vengono applicate automaticamente ai 5 fasci.
- **Monitoraggio in tempo reale:** l'interfaccia mostra lo stato del dispositivo, come note attive, ottava selezionata e distanza rilevata dal sensore a ultrasuoni.

La comunicazione tra il software di configurazione e Arduino si basa principalmente sull'uso di messaggi MIDI di tipo Control Change (CC).

Il flusso dei dati avviene in due direzioni:

- **Dal software verso Arduino:** quando l'utente assegna manualmente una nota a un laser o seleziona una scala predefinita, il programma invia un messaggio MIDI Control Change con numero di controllo compreso tra 20 e 24. Il valore del messaggio corrisponde al numero della nota MIDI da associare al relativo fascio. Arduino riceve il CC, aggiorna l'array `noteBase[]` e utilizza quel valore come nota di riferimento per gli eventi Note On/Off successivi.
- **Da Arduino verso il software:** la scheda, durante il normale funzionamento, invia a sua volta messaggi MIDI utili al monitoraggio dello stato:
 - CC 30–34: indicano l'accensione o lo spegnimento dei LED associati ai laser (nota attiva o rilasciata).
 - CC 40: comunica l'ottava corrente.
 - CC 91: trasmette il valore di distanza rilevato dal sensore a ultrasuoni, già scalato nell'intervallo 0–127.

Grazie a questo scambio, il software può non solo configurare le note ma anche visualizzare in tempo reale l'attività dello strumento (stato dei fasci, ottava, sensore).

5 Conclusioni

5.1 Risultati ottenuti

Il progetto ha portato alla realizzazione funzionante di un controller MIDI a forma di arpa laser, capace di interagire con software musicali e di essere configurato tramite un'apposita interfaccia software. L'hardware, composto da Arduino, sensori LDR, moduli laser e un sensore a ultrasuoni, è stato integrato in una struttura stampata in 3D, superando le difficoltà di assemblaggio e cablaggio. Il risultato è uno strumento che traduce i gesti dell'utente in segnali MIDI standard, permettendo di suonare note e controllare parametri in tempo reale. In questo modo è stato raggiunto l'obiettivo iniziale: un dispositivo originale, funzionale e in grado di unire elettronica, programmazione e musica in un'unica esperienza interattiva.

5.2 Possibili miglioramenti

Nonostante il funzionamento soddisfacente, il progetto presenta margini di evoluzione. Tra i miglioramenti futuri possibili si evidenziano:

- **Aumento del numero di laser:** aggiungendo altri moduli laser si potrebbe ottenere un'intera ottava completa, aumentando la gamma musicale disponibile e la flessibilità dell'arpa.
- **Miglioramento della struttura:** ottimizzando la base e gli alloggi dei laser si potrebbe facilitare l'allineamento, rendendo la proiezione dei fasci più precisa e stabile. È possibile pensare anche a soluzioni dinamiche che correggano automaticamente la posizione dei laser in tempo reale.
- **Software di configurazione avanzato:** trasformare il programma di configurazione in un plugin DAW permetterebbe di gestire note e parametri direttamente all'interno del software musicale, senza passare da un'applicazione esterna.
- **Estensione delle funzionalità MIDI:** aggiungere mapping configurabili, preset, salvataggio delle impostazioni e controllo di ulteriori pa-

rametri musicali per rendere il dispositivo più versatile e integrabile in contesti live o di produzione.

6 Appendice

6.1 Codici sorgente

6.1.1 Codice Arduino

```
#include "MIDIUSB.h"
#include <NewPing.h>

// ===== Pin LDR e LED =====
#define NUM_NOTE 5

const int LDRpin[NUM_NOTE] = {A5, A4, A3, A2, A1};
const int LEDpin[NUM_NOTE] = {2, 3, 4, 5, 6};

// ===== Sensore ultrasuoni =====
#define TRIG_PIN 10
#define ECHO_PIN 11
#define MAX_DISTANCE 15 // 15 cm max

NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);

// ===== Pulsanti Ottava =====
#define BTN_DOWN 8
#define BTN_UP 9

// ===== MIDI =====
byte noteBase[NUM_NOTE] = {60, 62, 64, 65, 67}; // C4, D4...
bool notaInviata[NUM_NOTE] = {false, false, false, false,
    false};

int THRESHOLD[NUM_NOTE];
int ottavaCorrente = 0;

// ===== Ultrasuono =====
int distanza = 0;
unsigned long lastPing = 0;
```

```

const unsigned long pingInterval = 50; // ogni 50 ms

// Antirimbalzo
unsigned long lastDebounce = 0;
const unsigned long debounceDelay = 200;

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

  for (int i = 0; i < NUM_NOTE; i++) {
    pinMode(LDRpin[i], INPUT);
    pinMode(LEDpin[i], OUTPUT);
    THRESHOLD[i] = calibraSingoloLaser(i);
  }

  pinMode(BTN_DOWN, INPUT_PULLUP);
  pinMode(BTN_UP, INPUT_PULLUP);
}

void loop() {
  // === Lettura LDR ===
  for (int i = 0; i < NUM_NOTE; i++) {
    int val = analogRead(LDRpin[i]);

    if (val < THRESHOLD[i] && !notaInviata[i]) {
      inviaNotaOn(noteBase[i] + (ottavaCorrente * 12), i);
      notaInviata[i] = true;
      digitalWrite(LEDpin[i], HIGH);
    }
    else if (val >= THRESHOLD[i] && notaInviata[i]) {
      inviaNotaOff(noteBase[i] + (ottavaCorrente * 12), i);
      notaInviata[i] = false;
      digitalWrite(LEDpin[i], LOW);
    }
  }

  // === Ultrasuono con intervallo ===
  if (millis() - lastPing >= pingInterval) {
    gestisciUltrasuono();
    lastPing = millis();
  }
}

```

```

// === Gestione Pulsanti ===
gestisciOttava();

// === Controllo messaggi da interfaccia ===
midiEventPacket_t rx;
do {
    rx = MidiUSB.read();
    if (rx.header != 0) {
        if ((rx.byte1 & 0xF0) == 0xB0) { // CC message
            byte controller = rx.byte2;
            byte value = rx.byte3;
            if (controller >= 20 && controller <= 24) {
                int index = controller - 20; // LDR 0..4
                noteBase[index] = value;
                Serial.print("Aggiornata nota LDR ");
                Serial.print(index);
                Serial.print(" -> ");
                Serial.println(value);
            }
        }
    }
} while (rx.header != 0);

delay(2);
}

// ===== Calibrazione LDR =====
int calibraSingoloLaser(int index) {
    long somma = 0;
    const int letture = 100;

    Serial.print("Calibrazione LDR ");
    Serial.print(index);
    Serial.println(" (laser attivo)...");

    for (int i = 0; i < letture; i++) {
        somma += analogRead(LDRpin[index]);
        delay(5);
    }
}

```

```

int media = somma / lettura;
int threshold = max(media - 100, 10);

Serial.print("Threshold LDR ");
Serial.print(index);
Serial.print(": ");
Serial.println(threshold);

return threshold;
}

// ===== MIDI =====
void inviaControlChange(byte controller, byte value, byte
channel = 0) {
midiEventPacket_t cc = {0x0B, (byte)(0xB0 | channel),
controller, value};
MidiUSB.sendMIDI(cc);
MidiUSB.flush();
}

void inviaNotaOn(byte nota, int index) {
midiEventPacket_t noteOn = {0x09, 0x90, nota, 127};
MidiUSB.sendMIDI(noteOn);

inviaControlChange(30 + index, 127); // LED ON

MidiUSB.flush();
Serial.print("Nota ON: ");
Serial.print(nota);
Serial.print(" (index ");
Serial.print(index);
Serial.println(")");
}

void inviaNotaOff(byte nota, int index) {
midiEventPacket_t noteOff = {0x08, 0x80, nota, 127};
MidiUSB.sendMIDI(noteOff);

inviaControlChange(30 + index, 0); // LED OFF

MidiUSB.flush();
}

```

```

    Serial.print("Nota OFF: ");
    Serial.print(nota);
    Serial.print(" (index ");
    Serial.print(index);
    Serial.println(")");
}

void inviaOttava(int ottava) {
    int val = 64 + ottava;
    inviaControlChange(40, val);
}

// ===== Gestione Pulsanti Ottava =====
void gestisciOttava() {
    unsigned long now = millis();

    if (now - lastDebounce > debounceDelay) {
        if (digitalRead(BTN_UP) == LOW) {
            ottavaCorrente = min(4, ottavaCorrente + 1);
            Serial.print("Ottava aumentata a: ");
            Serial.println(ottavaCorrente);
            inviaControlChange(31, ottavaCorrente + 64); // shift
                (+64 per evitare valori negativi)
            inviaOttava(ottavaCorrente);
            lastDebounce = now;
        }

        if (digitalRead(BTN_DOWN) == LOW) {
            ottavaCorrente = max(-3, ottavaCorrente - 1);
            Serial.print("Ottava diminuita a: ");
            Serial.println(ottavaCorrente);
            inviaControlChange(31, ottavaCorrente + 64);
            inviaOttava(ottavaCorrente);
            lastDebounce = now;
        }
    }
}

// ===== Gestione ultrasuoni con NewPing =====
void gestisciUltrasuono() {
    distanza = sonar.ping_cm();
}

```

```

if (distanza == 0) distanza = MAX_DISTANCE; // valore
massimo se nessuna risposta

int valoreMIDI = map(distanza, 2, MAX_DISTANCE, 127, 0);
valoreMIDI = constrain(valoreMIDI, 0, 127);

inviaControlChange(91, valoreMIDI);
}

```

6.1.2 Codice software di configurazione

```

import tkinter as tk
from tkinter import ttk
import mido

# === Setup MIDI ===
print("Porte MIDI disponibili:")
print(mido.get_output_names())

MIDI_PORT_NAME = "Arduino Leonardo"
outport = mido.open_output(MIDI_PORT_NAME)
inport = mido.open_input(MIDI_PORT_NAME)

# === Costanti ===
NOTES = {
    "Do (C)": 60, "Do# (C#)": 61, "Re (D)": 62, "Re# (D#)": 63,
    "Mi (E)": 64, "Fa (F)": 65, "Fa# (F#)": 66, "Sol (G)": 67,
    "Sol# (G#)": 68, "La (A)": 69, "La# (A#)": 70, "Si (B)": 71,
    "Do (C) 2": 72, "Do# (C#) 2": 73, "Re (D) 2": 74, "Re# (D#) 2": 75,
    "Mi (E) 2": 76, "Fa (F) 2": 77, "Fa# (F#) 2": 78, "Sol (G) 2": 79,
    "Sol# (G#) 2": 80, "La (A) 2": 81, "La# (A#) 2": 82, "Si (B) 2": 83
}

FIRST_OCTAVE = [ "Do (C)", "Do# (C#)", "Re (D)", "Re# (D#)",
    "Mi (E)", "Fa (F)", "Fa# (F#)", "Sol (G)",

```

```

    "Sol# (G#)", "La (A)", "La# (A#)", "Si (B)"]

SCALES = {
    "Maggiore (prime 5)": [0, 2, 4, 5, 7],
    "Minore (prime 5)": [0, 2, 3, 5, 7],
    "Maggiore Pentatonica": [0, 2, 4, 7, 9],
    "Minore Pentatonica": [0, 3, 5, 7, 10],
    "Blues": [0, 3, 5, 6, 7],
    "Diatonica sospesa (Sus4)": [0, 2, 5, 7, 9],
    "Orientale": [0, 1, 4, 5, 7],
    "Maggior armonica (prime 5)": [0, 2, 4, 7, 11],
    "Minore armonica (prime 5)": [0, 2, 3, 7, 10],
}
}

DEFAULT_COMBO_NOTES = ["Do (C)", "Re (D)", "Mi (E)", "Fa (F)",
    "Sol (G)"]

# === Funzioni utilitarie ===
def mostra_avviso(msg, widget=None, durata=1500):
    popup = tk.Toplevel()
    popup.overrideredirect(True)
    label = tk.Label(popup, text=msg, padx=10, pady=5,
        anchor="center")
    label.pack()
    popup.update_idletasks()
    w, h = popup.winfo_width(), popup.winfo_height()

    if widget:
        x = widget.winfo_rootx() + widget.winfo_width()//2 -
            w//2
        y = widget.winfo_rooty() + widget.winfo_height() + 5
    else:
        x = popup.winfo_screenwidth()//2 - w//2
        y = popup.winfo_screenheight()//2 - h//2
    popup.geometry(f"{w}x{h}+{x}+{y}")
    popup.after(durata, popup.destroy)

# === GUI setup ===
root = tk.Tk()
root.title("Configurazione Arpa Laser MIDI")
combos = []

```

```

def invia_config(mostra_popup=True):
    for i, combo in enumerate(combos):
        nota_val = NOTES[combo.get()]
        outport.send(mido.Message("control_change",
            control=20+i, value=nota_val))
        print(f"Invia LDR {i+1}: {combo.get()} ({nota_val})")
    if mostra_popup:
        mostra_avviso("Note applicate", widget=btn_invia)

def imposta_scala():
    tonica, scala_nome = root_note_combo.get(),
    scale_combo.get()
    if tonica not in NOTES or scala_nome not in SCALES:
        return

    tonica_val = NOTES[tonica]
    intervalli = SCALES[scala_nome]
    scale_notes = []

    for semitoni in intervalli:
        midi_val = tonica_val + semitoni
        nome = next((k for k, v in NOTES.items() if v ==
            midi_val), list(NOTES.keys())[0])
        scale_notes.append(nome)

    for i in range(5):
        combos[i].set(scale_notes[i])

    invia_config(mostra_popup=False)
    mostra_avviso("Scala applicata", widget=btn_scala)

# === Layout ===
# Colonna sinistra: configurazione manuale
label_left = ttk.Label(root, text="Configurazione manuale",
    font=("Roboto", 15))
frame_left = ttk.LabelFrame(root, labelwidget=label_left,
    padding=10)
frame_left.grid(row=0, column=0, padx=10, pady=10,
    sticky="nsew")

```

```

for i in range(5):
    ttk.Label(frame_left, text=f"LASER {i+1}").grid(row=i,
        column=0, padx=5, pady=5, sticky="w")
    combo = ttk.Combobox(frame_left, values=FIRST_OCTAVE,
        state="readonly")
    combo.set(DEFAULT_COMBO_NOTES[i])
    combo.grid(row=i, column=1, padx=5, pady=5)
    combos.append(combo)

btn_invia = ttk.Button(frame_left, text="Applica note",
    command=invia_config)
btn_invia.grid(row=6, column=0, columnspan=2, pady=10)

# Colonna destra: preset di scala
label_right = ttk.Label(root, text="Preset", font=("Roboto",
    15))
frame_right = ttk.LabelFrame(root, labelwidget=label_right,
    padding=10)
frame_right.grid(row=0, column=1, padx=10, pady=10,
    sticky="nsew")

ttk.Label(frame_right, text="Tonica:").grid(row=0, column=0,
    sticky="w", padx=5, pady=5)
root_note_combo = ttk.Combobox(frame_right,
    values=FIRST_OCTAVE, state="readonly")
root_note_combo.set("Do (C)")
root_note_combo.grid(row=0, column=1, padx=5, pady=5)

ttk.Label(frame_right, text="Scala:").grid(row=1, column=0,
    sticky="w", padx=5, pady=5)
scale_combo = ttk.Combobox(frame_right,
    values=list(SCALES.keys()), state="readonly")
scale_combo.set("Maggiore (prime 5)")
scale_combo.grid(row=1, column=1, padx=5, pady=5)

btn_scala = ttk.Button(frame_right, text="Applica Scala",
    command=imposta_scala)
btn_scala.grid(row=2, column=0, columnspan=2, pady=10)

# === Frame stato ===

```

```

status_label = ttk.Label(root, text="Stato arpa",
    font=("Roboto", 15))
status_frame = ttk.LabelFrame(root, labelwidget=status_label,
    padding=10)
status_frame.grid(row=1, column=0, columnspan=2, padx=10,
    pady=10, sticky="nsew")

# --- LED ---
canvas_width, canvas_height = 300, 80
led_frame = ttk.Frame(status_frame)
led_frame.grid(row=0, column=0, padx=5, pady=5)
ttk.Label(led_frame, text="LED").pack(pady=(0,5))
led_canvas = tk.Canvas(led_frame, width=canvas_width,
    height=canvas_height)
led_canvas.pack()

led_radius = min(canvas_height // 3, 20)
spacing = canvas_width // 6
led_widgets =
    [led_canvas.create_oval(spacing*(i+1)-led_radius,
        canvas_height//2-led_radius,
        spacing*(i+1)+led_radius,
        canvas_height//2+led_radius,
        fill="gray", outline="") for
        i in range(5)]

def aggiorna_led(index, acceso=True):
    if 0 <= index < len(led_widgets):
        led_canvas.itemconfig(led_widgets[index], fill="green"
            if acceso else "gray")

# --- Ottava ---
ottava_frame = ttk.Frame(status_frame)
ottava_frame.grid(row=0, column=1, padx=5, pady=5)
ttk.Label(ottava_frame, text="Ottava").pack(pady=(0,5))
ottava_label = tk.Label(ottava_frame, text="0", font=("Arial",
    18, "bold"))
ottava_label.pack()
def aggiorna_ottava(val): ottava_label.config(text=f"{val}")

# --- Ultrasuoni ---

```

```

ultra_frame = ttk.Frame(status_frame)
ultra_frame.grid(row=0, column=2, padx=5, pady=5)
ttk.Label(ultra_frame, text="Sensore").pack(pady=(0,5))
ultra_canvas = tk.Canvas(ultra_frame, width=60,
    height=canvas_height)
ultra_canvas.pack()
ultra_bar = ultra_canvas.create_rectangle(10, canvas_height,
    50, canvas_height, fill="blue")
def aggiorna_ultrasuoni(distanza, max_dist=100):
    altezza = max(10, min(canvas_height-10,
        canvas_height-10-(distanza/max_dist)*(canvas_height-20)))
    ultra_canvas.coords(ultra_bar, 10, altezza, 50,
        canvas_height-10)

# Centra frame interni
for i in range(3): status_frame.grid_columnconfigure(i,
    weight=1)
led_frame.grid(sticky="n"); ottava_frame.grid(sticky="n");
ultra_frame.grid(sticky="n")

# === Lettura MIDI ===
def leggi_midi():
    for msg in import.iter_pending():
        if msg.type == 'note_on':
            print(f"Nota ON ricevuta: {msg.note}")
        elif msg.type == 'note_off':
            print(f"Nota OFF ricevuta: {msg.note}")
        elif msg.type == 'control_change':
            if 30 <= msg.control <= 34:
                aggiorna_led(msg.control-30, msg.value>0)
            elif msg.control == 40:
                aggiorna_ottava(msg.value-64)
            elif msg.control == 91:
                aggiorna_ultrasuoni(int((127-msg.value)*100/127))
            else:
                print(f"CC sconosciuto: {msg.control}
                      value={msg.value}")
    root.after(50, leggi_midi)

leggi_midi()
root.mainloop()

```

6.2 Immagini

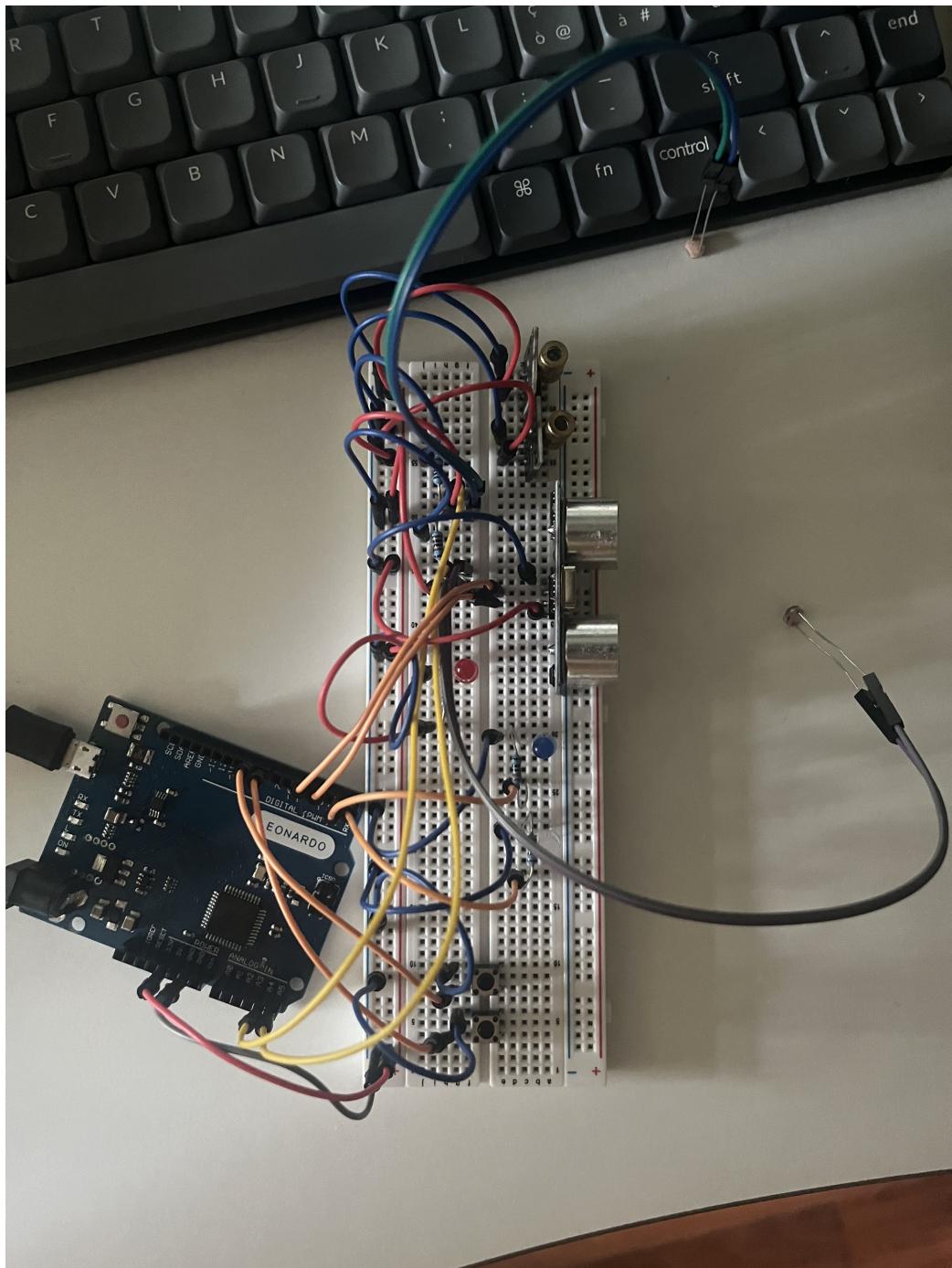


Figura 4: Primo prototipo su breadboard

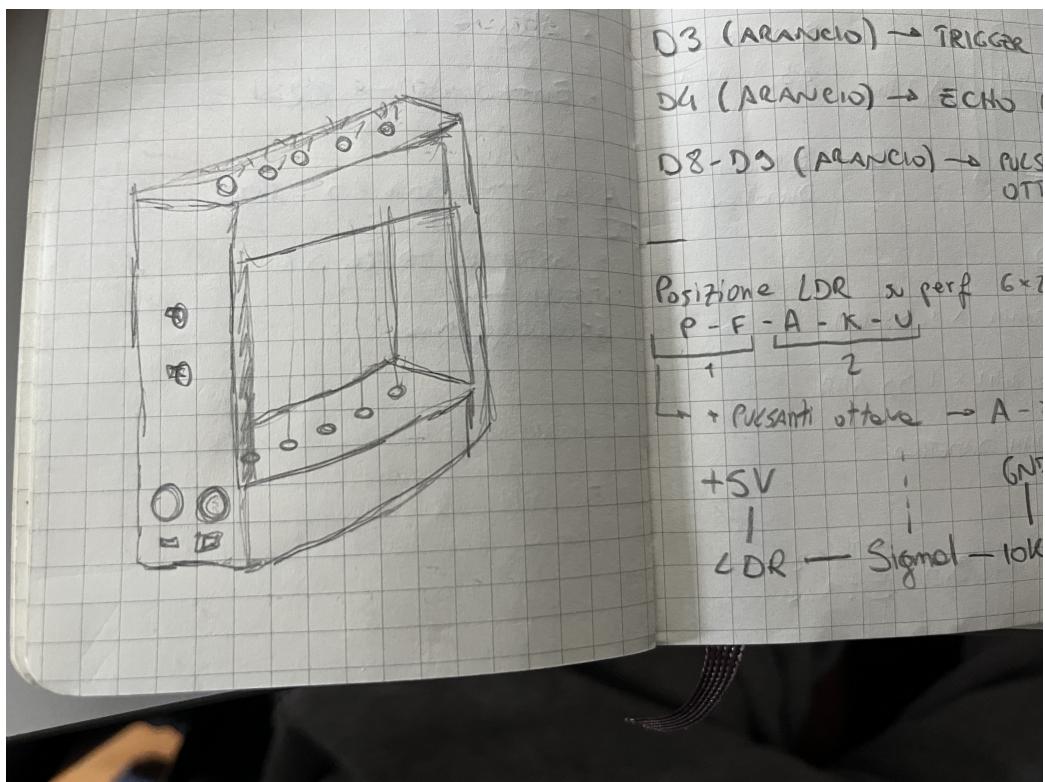


Figura 5: Primo schizzo dell'arpa laser

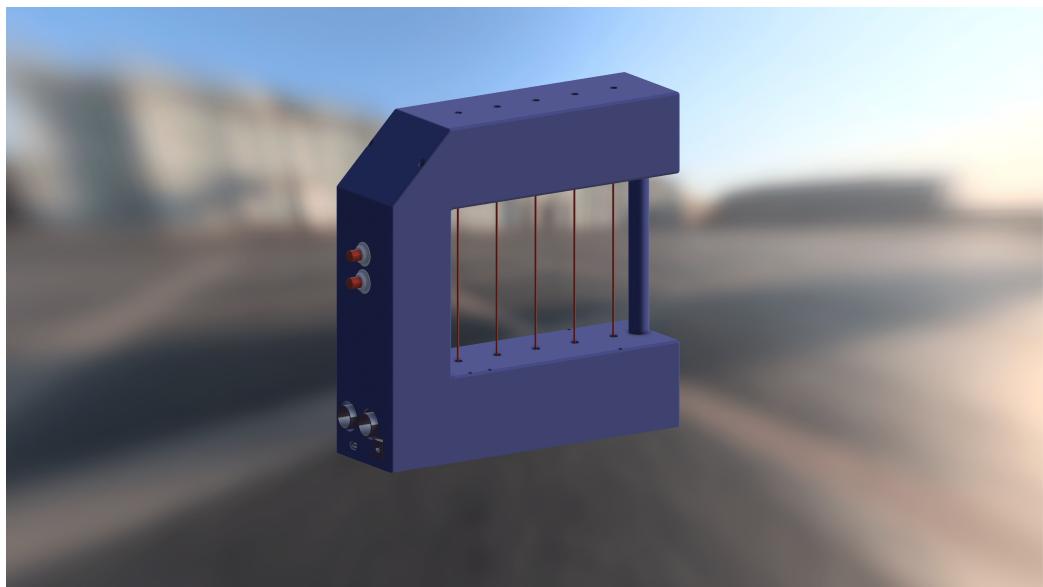


Figura 6: Mockup dell'arpa completata



Figura 7: Primo pezzo stampato - contenitore arduino