Scuola Superiore Sant'Anna - Università di Pisa

Laurea Magistrale in Embedded Computing Sistems

Corso di Real Time Systems

Documentazione del progetto

LeapChord

Professore

Giorgio Buttazzo

Studenti

Antonio Di Guardo
Andrea Vincentini

Anno Accademico 2015-2016

Capitolo 1

Requisiti del progetto

Develop a program that maps hand gesture and positions into chords (major, minor, seventh, diminished, etc) and tonality, so you can play a song moving the hands.

1.1 Terminologia utilizzata

Benché il progetto faccia uso del sistema europeo basato sulla scala pitagorica, il significato dei termini utilizzati potrebbe in parte differire da quello tradizionalmente utilizzato in ambito musicale. Tali discordanze, talvolta sostanziali, sono state introdotte ai fini di evitare ogni tipo di ambiguità lessicale. Nel corso della documentazione verranno considerate valide le definizioni riportate di seguito:

Lessico musicale introdotto

- **semitono**: distanza tra due note consecutive nella scala cromatica.
- **tono**: distanza tra due note che distano tra loro due semitoni.
- **tonica** (t): nota della scala cromatica scelta dall'utente.
- **tonalità** (T): insieme di note ottenute sviluppando la scala maggiore dalla tonica sulle sue tre ottave successive.
- **grado di una nota** *g(n)*: sia $n \in T$, $g(n) = (t - n)\%8 + 1$.
- **accordo** (A): insieme di note appartenenti a T. Il nome dell'accordo deriva dalla prima nota dell'insieme, chiamata semplicemente *prima*. Sia p prima dell'accordo, l'accordo verrà talvolta detto *generato* sulla nota p.
- **grado di un accordo** *g(A)*: sia p *prima* dell'accordo A, $g(A) = (t - p)\%8 + 1$.

- **settima minore** $\flat m$: sia n nota, $\flat m$ è la nota della scala cromatica distante 5 *toni* da n.

Lessico proprio del progetto

- **sessione** (s): fase del programma in cui è possibile suonare.
- **inizializzazione**: intervallo di tempo tra una sessione ed un altra.
- **stato** ($S(t)$): insieme dell'accordo e della nota in esecusione all'istante t. L'insieme può avere cardinalità 0, 1 o 2.
- **gesto**: qualsiasi movimento del polso o delle dita della mano.

1.2 Requisiti scelti

1.2.1 Requisiti fondamentali

1. Durante tutta l'esecuzione dell'applicativo, l'utente può interagire con il processo in esecuzione unicamente tramite *gesti* riconoscibili dal dispositivo `Leap Motion`. È pertanto necessario attenersi ai vincoli spaziali ed ambientali (temperatura, luminosità ecc.) definiti dal costruttore. Per ulteriori dettagli si rimanda a https://www.leapmotion.com/.
2. Nel corso della stessa sessione è possibile eseguire solo note o accordi contenenti note appartenenti alla *tonalità* scelta.
3. Nel corso di una *sessione* l'utente potrà utilizzare la mano sinistra per suonare un *accordo* e la mano destra per suonare singole note (melodia). È possibile suonare con una sola mano o utilizzando contemporaneamente entrambe le mani.
4. Non è possibile suonare due accordi contemporaneamente. Durante la sessione sono possibili momenti in cui nessun accordo è in esecuzione.
5. Lo stesso comportamento descritto da Req. 4 è valido per l'esecuzione delle note.
6. Durante la sessione è prevista un'interfaccia grafica in grado di mostrare all'utente:
   - lo stato corrente;
   - gli stati raggiungibili partendo dallo stato attuale e delle indicazioni sui *gesti* necessari per effettuare le transizioni.
7. Durante la fase di inizializzazione l'utente può decidere alcuni parametri fondamentali, tra i quali la tonica della sessione. La scelta è guidata grazie all'ausilio di un'interfaccia grafica.

8. L'applicativo produce come output messaggi midi standard che il calcolatore deve essere in grado di trasformare in suono.

9. Il programma necessita una distribuzione *Debian-type* equivalente ad Ubuntu 14.04 o successivi.

1.2.2 Scelte progettuali aggiuntive

- È implementato un dispositivo di tipo *metronomo*, con volume non regolabile, che aiuta l'utente durante la sessione.

- Il volume della nota e degli accordi sono fissi.

- (aggiunta a Req. 2) - Insieme tonalità ampliato: durante l'esecuzione del generico accordo A, l'insieme *tonalità* è ampliato della *settima minore* relativa alla *prima* di A.

- (aggiunta a Req. 3) - Conflitto tra mani: l'utilizzo di più di una mano dello stesso tipo (destra o sinistra) da luogo a comportamenti indeterminati.

- (aggiunta a Req. 4) - Accordi eseguibili: durante la sessione è possibile in qualsiasi momento suonare un accordo di qualsiasi grado, anche quello corrente.

- (aggiunta a Req. 4) - Composizione accordi: ogni accordo è composto da quattro note le cui distanze reciproche sono decise dall'utente. È possibile eseguire la *prima* dell'accordo anche con uno strumento differente dalle altre (nota al basso).

- (aggiunta a Req. 5) - Note eseguibili: è possibile suonare con la mano destra una nota a scelta tra 15 appartenenti all'insieme *tonalità* ampliato (insieme *tastiera*). La modalità di scelta di *tastiera* è decisa dall'utente e dipende dall'accordo correntemente suonato. Nel caso nessuno accordo sia suonato, la dipendenza è nei confronti dell'ultimo accordo suonato. In fase di inizializzazione, la scelta dipende dall'accordo generato sulla tonica.

- (aggiunta a Req. 5) - Conflitto tra note: nel caso in cui l'utente tenti di suonare due note contemporaneamente, non è possibile determinare a priori la nota che verrà suonata dal sistema. Prima di suonare una nota è necessario smettere di suonare la nota precedente.

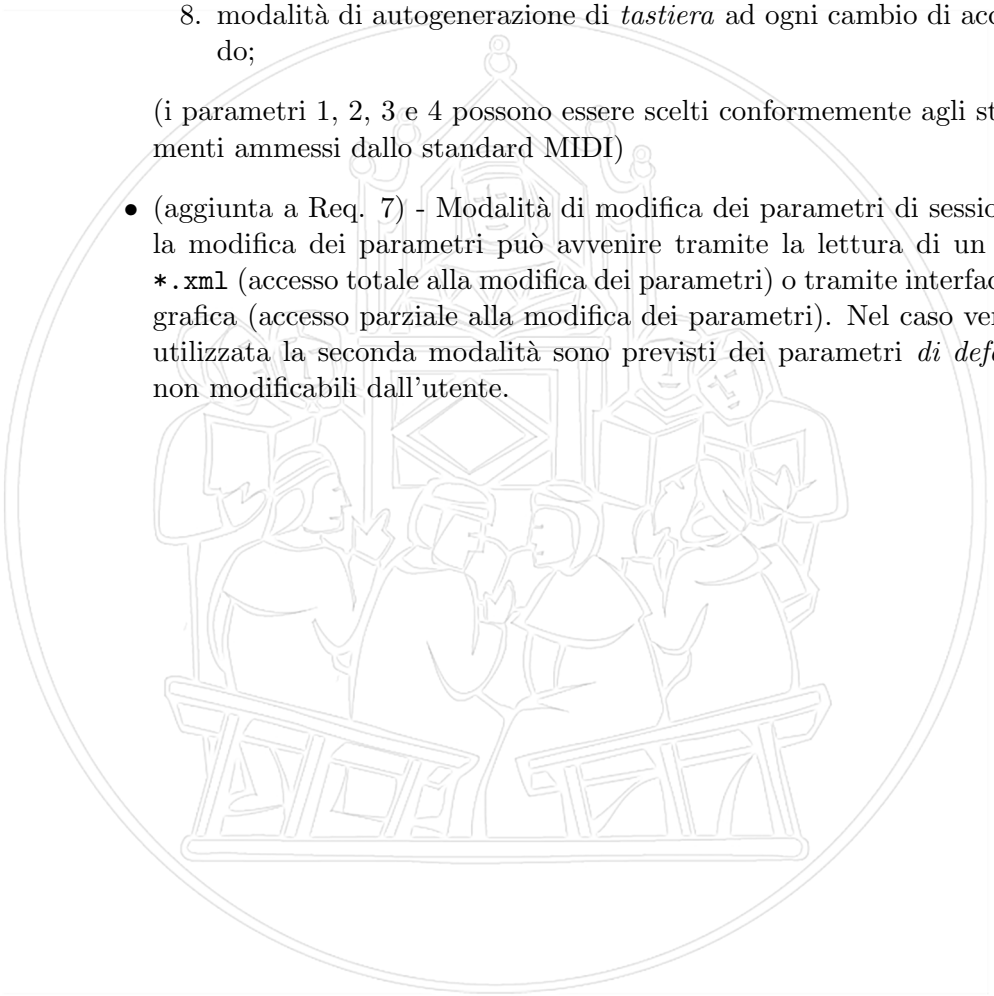
- (aggiunta a Req. 7) - Parametri di sessione: l'utente è in grado di settare i seguenti parametri:

    1. strumento con il quale suonare gli accordi;

2. strumento con il quale suonare la *prima* degli accordi;
3. strumento con il quale suonare le note della melodia;
4. strumento con il quale suonare le note del metronomo;
5. frequenza del metronomo;
6. tonica (una qualsiasi nota della scala cromatica);
7. modalità di costruzione degli accordi;
8. modalità di autogenerazione di *tastiera* ad ogni cambio di accordo;

(i parametri 1, 2, 3 e 4 possono essere scelti conformemente agli strumenti ammessi dallo standard MIDI)

- (aggiunta a Req. 7) - Modalità di modifica dei parametri di sessione: la modifica dei parametri può avvenire tramite la lettura di un file `*.xml` (accesso totale alla modifica dei parametri) o tramite interfaccia grafica (accesso parziale alla modifica dei parametri). Nel caso venga utilizzata la seconda modalità sono previsti dei parametri *di default* non modificabili dall'utente.



Capitolo 2

Design funzionale

2.1 Composizione del programma

Come mostrato in Figura 2.1, il programma può essere visto come un ciclo infinito di due fasi:

- inizializzazione;
- sessione.

Durante la fase di inizializzazione l'utente, guidato dall'interfaccia grafica e tramite l'ausilio della `Leap Motion` setta i parametri della sessione. Nel caso l'utente voglia uscire dal programma, viene posta a *true* una particolare variabile che, dopo l'uscita della fase di inizializzazione, permette la terminazione del programma.

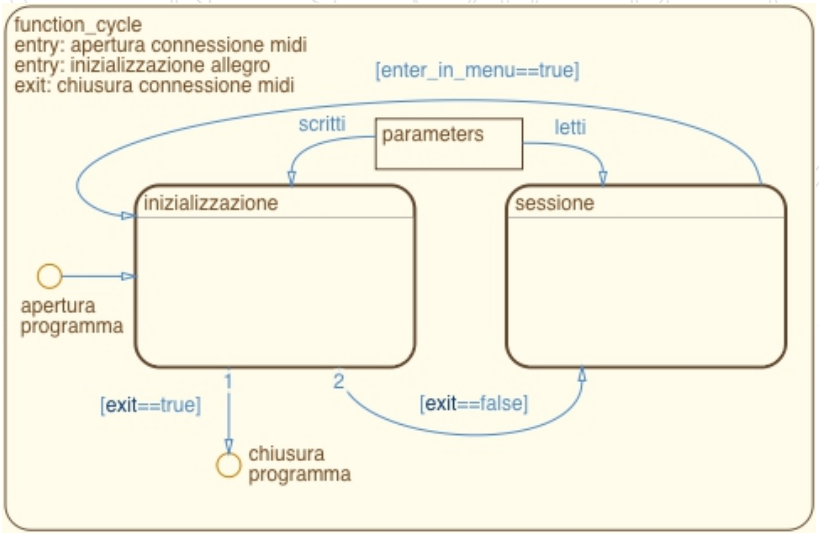


Figura 2.1: Funzione ciclica dell'applicazione.

Inizializzazione Nella prima fase dell'inizializzazione viene chiesto all'utente di indicare la modalità di acquizizione dei parametri (manuale o automatico da file xml). In questo frangente l'utente può decidere di uscire, bypassando dunque la fase di acquisizione dei parametri di sessione (si veda la Figura 2.2).

Nel caso venga scelta la modalità di inizializzazione di tipo *manuale*, dal punto di vista logico l'acquisizione dei parametri avviene tramite l'inserimento di informazione mediante *gesti*. L'utente è guidato dall'inserimento da un interfaccia grafica con funzione sia di guida che di feedback. Per ulteriori dettagli si rimanda al Par. 2.4.

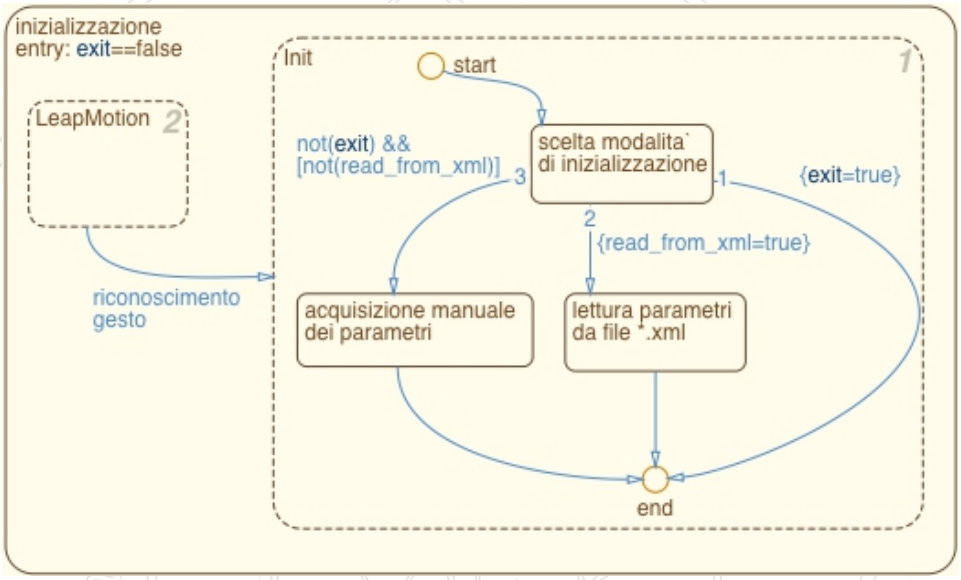


Figura 2.2: Design logico di *inizializzazione*.

Sessione Come mostrato in Figura 2.3, durante la sessione il funzionamento è dato dall'interazione dei seguenti blocchi logici:

- la **Leap Motion** viene utilizzata per riconoscere i gesti dell'utente;
- la **Logica** acquisice i dati prodotti dal blocco *LeapMotion* e, in base ai parametri forniti dall'utente, calcola lo stato corrente (si rimanda a Par. 2.2);
- il **Metronomo**, sulla base dei parametri scelti, comunica al blocco Midi gli istanti in cui *battere il tempo*;
- il **Midi**, sulla base dello stato del programma e degli strumenti scelti dall'utente, si occupa della riproduzione dei suoni (si rimanda a Par. 2.3);

- la **Grafica** si occupa di fornire un feedback visivo all'utente (si veda Par. 2.4).

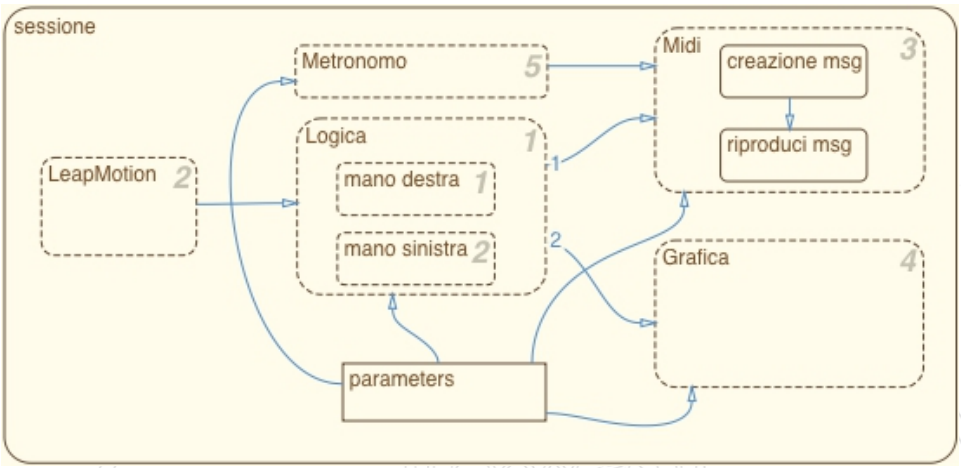


Figura 2.3: Design logico di *sessione*.

2.2 Logica dell'applicazione

Come illustrato in Figura 2.3, la logica dell'applicazione è suddivisa in due parti quasi del tutto indipendenti l'una dall'altra:

- la **logica della mano sinistra** sulla base delle informazioni fornite dalla `Leap Motion` determina il grado dell'accordo che deve essere eseguito;
- la **logica della mano destra** determina quale delle note di *tastiera* eseguire (si rimanda a Capitolo 1, Requisito 5).

2.2.1 Logica della mano sinistra

La logica di generazione degli accordi può essere vista come una macchina a stati governata dalle seguenti regole:

- nell'istante iniziale della sessione nessun accordo è in esecuzione.
- Un accordo può essere eseguito solo se in quel momento la mano sinistra è riconosciuta dalla `Leap Motion`. Se in qualsiasi momento la mano sinistra smette di essere rilevata, l'accordo corrente cessa immediatamente la sua esecuzione.
- Per eseguire un accordo l'utente può posizionare la mano sinistra in un punto dello spazio qualsiasi, a patto che possa essere rilevato dalla `Leap Motion`.

- Durante l'esecuzione di un accordo, l'utente può decidere di *passare* ad un accordo qualsiasi dei 7 possibili, compreso l'accordo corrente.

- Ad eccezione dell'accordo del settimo grado, per cambiare accordo è necessario:

  – rivolgere il palmo della mano verso il basso;

  – variare la velocità in modulo del proprio polso di un valore superiore a `v_limit` (si veda Fig. 2.5).

  Ognuna delle sei direzioni rispetto agli assi della `Leap Motion` corrisponde agli accordi dei primi 6 gradi (si veda Fig. 2.4).

- L'accordo del settimo grado viene eseguito nel caso il palmo venga rivolto verso l'alto.

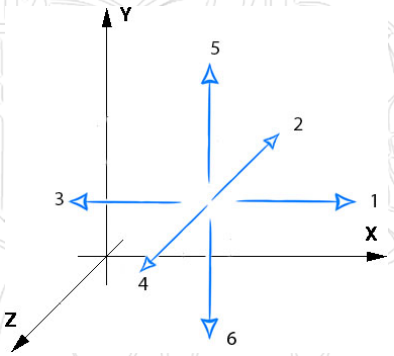


Figura 2.4: Mappatura degli accordi sviluppati sui primi sei gradi della scala.

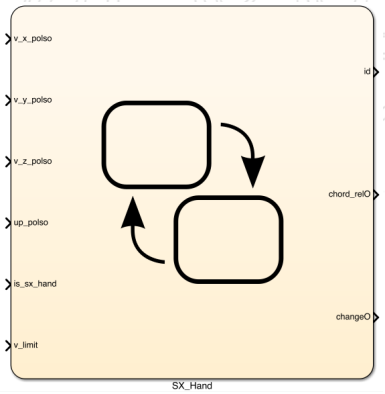


Figura 2.5: Input e output semplificati della *state machine* che governa la logica della mano sinistra.

Sulla base dei parametri inseriti dall'utente, il grado dell'accordo verrà poi mappato nell'accordo reale (insieme di note di *tonalità*) che verrà poi eseguito dal blocco logico Midi.

2.2.2 Logica della mano destra

La generazione delle note può essere vista come una macchina a stati governata dalle seguenti regole:

- nell'istante iniziale della sessione nessuna nota è in esecuzione. Solo una nota per volta può essere eseguita (si veda Cap. 1).

- Una nota può essere eseguita solo se in quel momento la mano destra è riconosciuta dalla `Leap Motion`. Se in qualsiasi momento la mano destra smette di essere rilevata la nota corrente cessa immediatamente la sua esecusione.

- Per eseguire una nota l'utente può posizionare la mano in un punto qualsiasi dello spazio, a patto che possa essere rilevato dalla `Leap Motion`.

- Durante tutta la sessione il palmo della mano deve essere rivolto verso il basso. Rivolgere il palmo verso l'alto comporta l'immediata terminazione della sessione corrente e l'ingresso nella fase di inizializzazione.

- Ad un istante t ad ogni dito della mano è associata una nota di *tastiera*. Pertanto è possibile suonare solo una delle 5 note della *quintupla corrente*.

- All'inizio della sessione la *quintupla corrente* è composta dalle prime 5 note di *tastiera*. Per cambiare quintupla è necessario muovere la mano verso l'alto (passare alla quintupla superiore) o verso il basso (passare alla quintupla inferiore) con velocità del polso in modulo maggiore di `limit_hand_velocity` (si veda Figura 2.6).

- Per suonare una delle cinque note della quintupla è necessario che la punta del dito al quale è associata si trovi più in basso del polso di un valore predefinito. Per facilitare l'esecuzione sono previste distanze limite differenziate a seconda delle diverse dita della mano.

Sulla base dei parametri scelti dall'utente, l'output della state machine verrà poi mappato in una nota di *tastiera* e quindi eseguita dal blocco logico Midi.

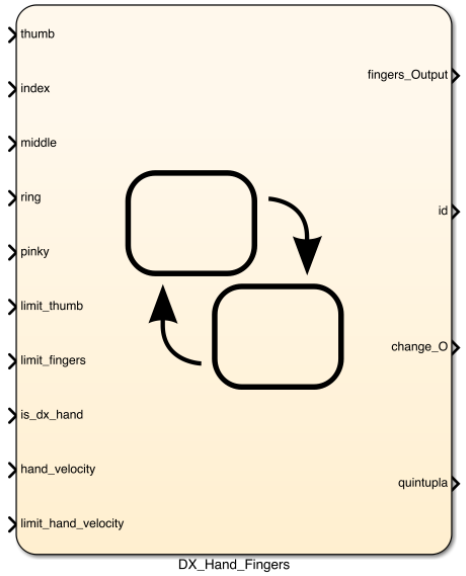


Figura 2.6: Input e output semplificati della *state machine* che governa la logica della mano destra.

2.3 Blocco logico MIDI

Visto nella sua interezza, il blocco *logica del programma* produce ad ogni iterazione uno stato che, per la definizione di accordo data, può essere visto come un insieme di quattro note. Il blocco logico **MIDI** si occupa dunque di riprodurre le note facenti parte lo stato del programma.

Il blocco Midi può essere suddiviso in due parti:

- la prima parte, che risiede all'interno del programma, si occupa di leggere lo stato del programma e, sulla base dei parametri inseriti dall'utente (strumenti di esecuzione scelti ecc), procuce messaggi conformi allo standard Midi.

- La seconda parte, che risiede nel calcolatore in cui è in esecuzione il programma, si occupa di trasformare in suoni i messaggi Midi generati dal programma. Per una visione più dettagliata delle configurazioni necessarie si veda il Cap. 4.

2.3.1 Generazione dei messaggi MIDI

Un messaggio Midi, a seconda della tipologia, è una sequenza di 8, 16 o 24 bit. All'interno del messaggio è sempre identificabile una parte di intestazione (1 byte) e una parte di informazione (0, 1 o 2 byte). Gli 8 bit di intestazione sono a loro volta suddivisibili in tre parti distinte:

- il **primo bit** è sempre pari ad 1. Grazie a questa informazione il sintetizzatore è in grado di distinguere i byte di intestazione da quelli di informazione (che hanno sempre il primo bit pari a 0).
- i **bit da 2, 3 e 4** servono per identificare la tipologia di messaggio midi. Le sequenze da 000 a 110 rappresentano le operazioni sui canali (inizio e fine nota, cambio degli strumenti ecc) mentre la sequenza 111 è riservata per i messaggi di sistema (clock di sincronizzazione tra dispositivi, stop dell'invio di messaggi, ecc).
- nel caso i primi 4 bit siano compresi tra 0x8 e 0xE, i restanti 4 bit stabiliscono il canale rispetto al quale l'operazione fa riferimento. È necessario distinguere il concetto di *porta midi*, ovvero la connessione con un sintetizzatore midi a livello di sistema, da quello di *canale* della connessione midi: una volta aperta una connessione essa prevede l'apertura di 16 canali indipendenti. Su ogni canale possono essere impostati degli attributi, tra cui il timbro che il sintetizzatore utilizzerà per riprodurre le note.
- Nel caso i primi 4 bit siano uguali a 0xF, i restanti 4 bit identificano la tipologia di messaggio di sistema desiderato (lo standard midi prevede la specifica di soli 12 messaggi).

A seconda della tipologia di messaggio (e dunque di operazione richiesta al sintetizzatore) è possibile allegare fino a 2 byte di informazione. Ogni byte di informazione ha il bit più significativo pari a 0.

I messaggi midi utilizzati dal programma sono i messaggi di inizializzazione dello strumento del canale ed i messaggi di riproduzione delle note.

L'invio dei messaggi è stato permesso grazie all'utilizzo di una libreria minimale (*RtMidi*) in grado di connettersi ad un socket aperto a livello di sistema. La classe utilizzata (*RtMidiOut*) permette inoltre di inviare byte in tempo reale ed *in maniera grezza* senza la necessità di alcuna sincronizzazione con il sintetizzatore. La latenza dell'esecuzione dipenderà dunque unicamente dalla reattività (e dunque dalla priorità di esecuzione) del sintetizzatore utilizzato. Per ulteriori approfondimenti sulla libreria utilizzata si rimanda a `https://www.music.mcgill.ca/~gary/rtmidi/`.

Messaggi di Inizializzazione dello strumento Questa tipologia di messaggi (nel gergo midi *program change*) è utilizzata al termine di ogni fase di inizializzazione che non preveda l'uscita dal programma, al fine di impostare il timbro dei canali midi utilizzati. Il programma prevede l'utilizzo di 4 canali:

- **canale 0001**: utilizzato per la riproduzione della melodia generata dalla mano destra;

- **canale 0010**: utilizzato per la riproduzione delle note dell'accordo generato dalla mano sinistra;
- **canale 0011**: utilizzato dal metronomo;
- **canale 0100**: utilizzato dal basso che suona la *prima* dell'accordo in esecuzione.

Il messaggio di inizializzazione dello strumento è dunque della forma:

- byte 1: 1010 | bit canale (4);
- byte 2: 0 | bit strumento (7).

I bit strumento sono un numero compreso tra 0 e 127. La corrispondenza tra numeri e strumenti, benchè sia almeno in parte standardizzata, non assicura che il sintetizzatore utilizzato sia all'atto pratico in grado di riprodurre il timbro voluto.

Lo standard midi prevede la possibilità di reimpostare lo strumento del canale durante l'esecuzione.

Riproduzione delle note La creazione e l'invio dei messaggi di riproduzione delle note rappresenta il cuore del blocco logico Midi presente nel design funzionale del programma.

Una delle peculiarità del protocollo midi è che ogni nota suonata necessita di un messaggio di attivazione a partire dal quale comincia l'esecuzione della nota, ed uno per la disattivazione, grazie al quale il suono viene terminato. Questa caratteristica fa del midi uno strumento ideale per l'esecuzione di suoni dei quali non è possibile conoscere a priori la durata.

Il blocco midi ha bisogno dunque di un proprio stato interno, poiché ogni qual volta avviene un cambiamento dello stato del programma è necessario disattivare le note in esecuzione prima di attivarne di nuove. Sono pertanto implementati due tipi di messaggi:

- avvio nota (*note on*): è composto da 3 byte così suddivisi:
  - **primo byte**: 1001 | bit canale (4);
  - **secondo byte**: 0 | bit nota (7);
  - **terzo byte**: 0 | bit volume (7).
- Terminazione nota (*note off*): è anch'esso composto da 3 byte:
  - **primo byte**: 1000 | bit canale (4);
  - **secondo byte**: 0 | bit nota (7);
  - **terzo byte**: raramente utilizzato, di default è impostato a 0x00.

In entrambi i messaggi i **bit nota** codificano un numero compreso tra 0 e 127 indicante la nota da attivare o disattivare. La distanza unitaria tra due note è corrispondente alla loro distanza cromatica, pertanto è possibile eseguire note su poco più di 10 ottave. Il do centrale ($\sim$ 261 Hz) è identificato dal numero 60 (0x3C).

I **bit volume** danno una misura indicativa del volume con il quale vengono riprodotte le note (l'andamento di tale volume dipende molto dal timbro scelto per il canale). Tra i valori comunemente utilizzati, 42 corrisponde approssimativamente al *p* musicale, 80 al *f* e 127 al *ffff*.

Per una spiegazione orientata alla programmazione dello standard Midi si rimanda a `http://www.music-software-development.com/midi-tutorial.html` (italiano) e `http://www.istitutobellini.cl.it/file.php/1/Master/Informatica_Relandini_2012/protocollo_MIDI.pdf` (inglese).

2.4 Grafica

2.4.1 Libreria grafica "allegro"

Il progetto prevede un'interfaccia con l'ausilio della libreria grafica ***allegro***. Nel caso la libreria non fosse già presente nel sistema sarà necessario scaricare il pacchetto tramite il comando:

```
sudo apt-get install liballegro4.2 liballegro4.2-dev
```

MakeFile All'interno del makefile sono stati aggiunte i parametri necessari in fase di compilazione per includere la libreria allegro.

```
ALLEGRO_OPT := `allegro-config --libs`
```

2.4.2 Interfaccia grafica

Nell'esecuzione del progetto è possibile individuare 3 interfacce grafiche aggiornate tramite i dati letti dalla Leap Motion.

Menù HomePage La prima schermata che viene visualizzata permette di scegliere tra:

- una configurazione di default impostata tramite la lettura dei dati da un file xml;
- l'ingresso in un menù di configurazione;
- uscita dal programma.

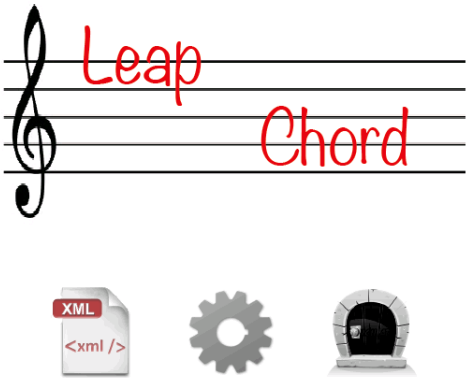


Figura 2.7: Menu HomePage

Come è possibile vedere in Figura 2.7 la schermata è composta da 3 bitmap stampate in scala di grigio sopra le quali viene stampata una sola icona a colori per discriminare la posizione all'interno del menù (per capire come avviene l'aggiornamento o la selezione si rimanda al paragrafo sui thread).

Menù Configurazione Il menù di configurazione si contrappone alla configurazione da file poichè permette all'utente di decidere manualmente le impostazioni.
Per ragioni di praticità alcune impostazioni sono state semplificate, rispetto alla configurazione da file xml, fornendo all'utente un numero limitato di opzioni selezionabili; rimane tuttavia la possibilità di ampliare questo tipo di menù in modo non troppo dispendioso.
Come mostrato in Figura 2.8 questa schermata è suddivisa in 5 sottomenù che permettono all'utente di configurare:

- modalità per la mano destra;
- tonalità;
- accordi;
- strumento per la mano destra;
- strumento per la mano sinistra.

Un feedback fornito all'utente per identificare il sottomenù selezionato viene dato dal rettangolo rosso.

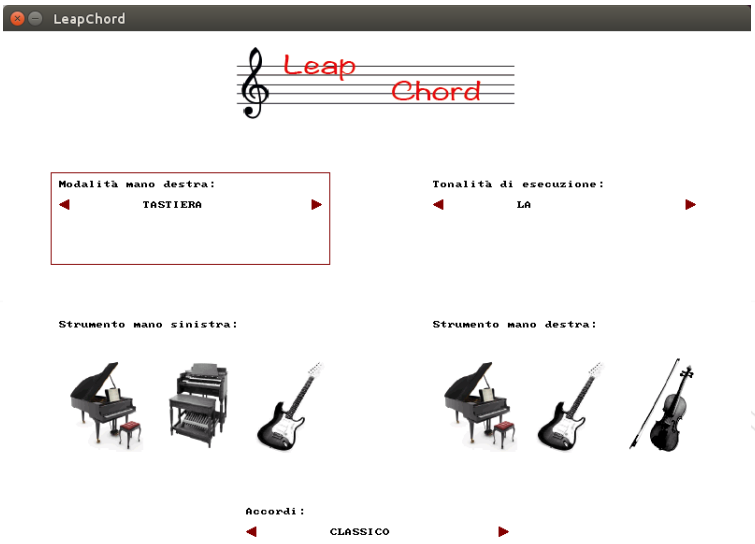


Figura 2.8: Menu Configurazione

I 5 sottomenù posso essere a loro volta suddivisi in 2 gruppi in base al tipo di selezione: *grafica* (sottomenù per la scelta degli strumenti) e *testuali* (sottomenù per la scelta della modalità, tonalità e accordi).

Accordi e Tastiera Una volta scelta la configurazione di default o tramite il menù di configurazione l'utente dovrà interagire con l'ultima schermata relativa alla produzione del suono. L'aspetto di questa finestra viene mostrato in Figura 2.9.
È possibile individuare nella parte sinistra della schermata l'elenco degli accordi e nella parte destra la tastiera delle note.
Ad ogni accordo è affiancato un triangolo per indicare all'utente il movimento da eseguire con la mano sinistra per suonare l'accordo desiderato. Un rettangolo verde fornisce un feedback visivo per indicare l'accordo che si sta suonando. Ad ogni movimento effettuato viene ristampata la bitmap contenente l'elenco degli accordi più un serie di rettangoli di colore nero intorno ad ogni accordo non suonato e un rettangolo verde intorno all'accordo suonato.
La tastiera contiene le note relative all'accordo suonato e viene quindi aggiornata ad ogni cambio di accordo. L'aggiornamento consiste nella riscrittura della bitmap contenente la griglia dei tasti con conseguente scrittura delle nuove note. Sulla tastiera è possibile individuare 2 tipi di feedback:

- un cerchio verde per evidenziare la nota suonata;
- un cerchio bianco per indicare la tastiera che stiamo usando.

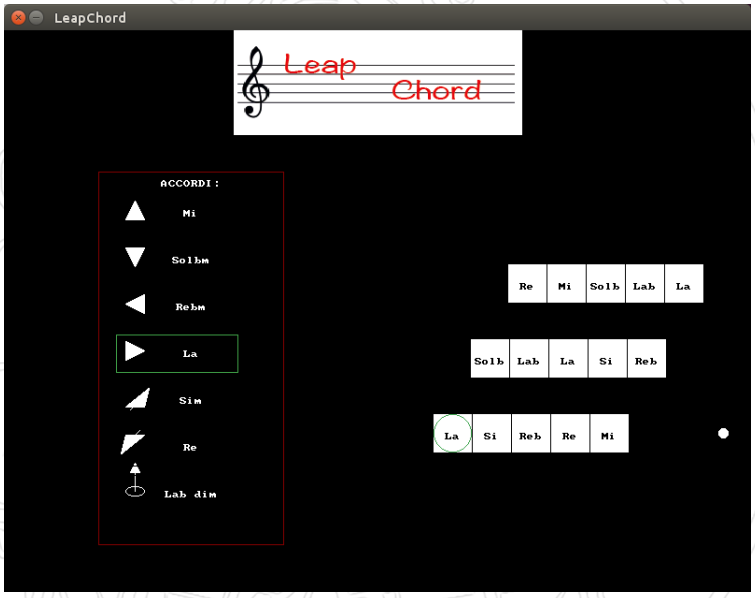


Figura 2.9: Accordi e Tastiera

Capitolo 3

Implementazione

In questo capitolo verranno descritti i thread e le funzioni principali necessarie per gestire l'esecuzione del programma.

3.1 Main()

La funzione ***main()*** ha una struttura molto semplice all'interno della quale è possibile individuare l'inizializzazione per il *midi* e per *allegro* e il corpo vero e proprio del programma (***function_cycle()***) eseguito fino al verificarsi del settaggio del parametro di uscita.

3.2 Function_cycle()

Nel corpo della funzione ***function_cycle()*** è possibile individuare 2 fasi distinte di funzionamento:

- inizializzazione delle strutture
- generazione del suono

Nella fase iniziale viene invocata la funzione ***inizialize()*** con quale si avvia il ***thread_allegro_init()*** per l'inizializzazione dei parametri della struttura:

Listing 3.1: Struttura InitType

```
typedef struct {
 InstrumentType bassInstrument;
 InstrumentType sxInstrument;
 InstrumentType dxInstrument;
 InstrumentType drumInstrument;
 int bpmDrums;
 Tonality tonality;
 Modality modality;
```

```
  boolean_T exit;
  boolean_T read_from_xml;
  int _1_;
  int _2_;
  int _3_;
  int _4_;
} InitType;
```

Una volta inizializzati i parametri della struttura di tipo InitType vengono avviati i 3 thread necessarri per la generazione del suono:

- ***thread_hands_sx()***;
- ***thread_hands_dx()***;
- ***thread_drums()***.

3.3 Thread: *thread_allegro_init()*

Il thread ***thread_allegro_init()*** ha una struttura molto complessa in quanto si occupa di gestire entrambe le fasi di inizializzazione tramite 2 schermate (vedi capitolo *Grafica*).
Il comportamento di questo thread prevede 3 fasi:

1. gestione della schermata *Home*
2. gestione della schermata *Configurazione*
3. predisposizione della schermata *Acccordi e Tastiera*

La fase 2 è opzionale e vi si accede solamente nel caso venga selezionato l'icona del menù *Configurazione* nella fase 1 e si occupa della gestione dei sottomenù grafici.
NB. Prima di accedere alla parte ciclica del thread è presente uno **sleep(1)** *strettamente necessario* per evitare di selezionare involontariamente la prima opzione del primo sottomenù.
Nelle fasi 1 e 2 ogni volta che viene eseguito uno spostamento tra le icone del menù o tra le opzioni di un sottomenù, per permettere ad allegro di aggiornare la schermata, viene eseguita una postpone_deadline di 500 ms rispetto alla deadline originale.

Entrambi le parti per gestire l'aggiornamento e la selezione delle opzioni sfruttano le funzioni:

- ***change_modality()***;
- ***confirm_modality()***

Funzione change_modality() Questa funzione legge i dati dalla Leap Motion e serve per riconoscere un eventuale movimento del dito indice della mano destra lungo uno dei semiassi dell'asse x con velocità in modulo superiore a 500 mm/s. Nel caso di spostamento lungo il semiasse positivo si passerà alla voce successiva mentre nel caso di spostamento lungo il semiasse negativo si passerà alla voce precedente.

Funzione confirm_modality() Questa funzione legge i dati dalla Leap Motion e serve per riconoscere un eventuale movimento del dito indice della mano destra lungo il semiasse negativo dell'asse z con velocità in modulo superiore a 800 mm/s. Nel caso venga riconosciuto un movimento di questo genere verrà selezionata l'opzione corrente del menù.

3.4 Thread "sonori"

Il thread *thread_drums()* non è altro che un metronomo di sostegno all'utente che produce un suono con periodo e volume costante dove non è presente niente di significante di cui parlare.
I thread ***thread_hands_dx()*** e ***thread_hands_sx()*** hanno un comportamento simile dovuto a scelte progettuali effettuate per ridurre al minimo le operazioni di aggiornamento del suono e della grafica. Gli aggiornamenti del suono e della grafica avvengono *solamente* quando il bit di aggiornamento **change0** restituito dalla state machine non è *false*.
I thread effettuano le seguenti operazioni:

1. lettura del frame della Leap Motion e salvataggio nella struttura dati delle informazioni necessarie;
2. invio delle informazioni alla state machine;
3. lettura dei valori restituiti dalla stase machine;
4. eventuale aggiornamento del suono e dalla parte grafica.

NB. Il codice rappresentante il funzionamento della state machine è stato autogenerato tramite Matlab.
Ogni volta che si verifica un aggiornamento viene prodotta una postpone_deadline().

Struttura: *ControllerSxMsgType* Il thread ***thread_hands_sx()*** si occupa inizialmente di riempire una struttura dati di tipo

Listing 3.2: Struttura ControllerSxMsgType

```
typedef struct
{
```

```
  int id_frame;
  PointType position;
  PointType velocity;
  boolean_T is_sx_hand;
  int palm_normal;
} ControllerSxMsgType
```

dove le informazioni sono relative al palmo della mano sinistra.

Struttura: ***ControllerDxMsgType*** Il thread ***thread_hands_dx()*** si occupa inizialmente di riempire una struttura dati di tipo

Listing 3.3: Struttura ControllerdxMsgType

```
typedef struct
{
  int id_frame;
  int thumb;
  int index;
  int middle;
  int ring;
  int pinky;
  int hand;
  int hand_speed;
  boolean_T init;
} ControllerDxMsgType;
```

dove le informazioni sono relative alla posizione lungo y della dita e alla velocità e alla normale del palmo della mano destra.

Capitolo 4

HOWTO: Riprodurre midi su sistemi Linux con Alsa

4.1 Strumenti necessari

Per permettere la riproduzione realtime di suoni midi nei sistemi Linux è necessario predisporre la macchina attraverso la configurazione degli strumenti necessari. Gli strumenti necessari sono:

- la libreria ***asound***;
- un sintetizzatore audio ***Timidity***;
- un server audio ***jack***;
- una applicazione che generi suoni midi.

4.2 Configurare la macchina Linux

Nel seguito verranno riportati i passaggi per configurare la macchina in modo corretto.

Installare gli strumenti Per installare gli strumenti necessari è sufficiente aprire un terminale e dare i seguenti comandi:
libreria asound di ALSA:

```
sudo apt-get install libasound2
```

sintetizzatore timidity

```
sudo apt-get install timidity
```

server audio jack

```
sudo apt-get install jackd
```

Configurazione Real Time per Jack Per ottenere i suoni in *real time* è necessario fare 3 ulteriori passaggi per completare la configurazione:

1. modificare il file di configurazione;
2. creare il gruppo "realtime";
3. riavviare

Per il *file di configurazione* (punto 1) posizionarsi nella cartella */etc/security/limits.d* e creare il file ***/etc/security/limits.d/99-realtime.conf***. Questo file dovrà contenere almeno le seguenti istruzioni:

```
@realtime   -   rtprio      99
@realtime   -   memlock     unlimited
```

Per *creare il gruppo "realtime"* (punto 2) aprire un terminale e come utente root dare i seguenti comandi:

```
groupadd realtime
usermod -a -G realtime ``yourUserID"
```

dove l'utente deve sostiture yourUserID con il nome del gruppo a cui appartiene.

Avviare Timidity in modalità realtime Una volta completata con successo la configurazione l'utente può avviare una applicazione in grado di sfruttare timidity per la riproduzione real time di suoni midi.
Tramite il comando:

```
timidity --realtime-priority=100 -iA -B2,8 -Os1l -s 44100
```

è possibile abilitare la creazione di un canale midi con priotità real time che sfrutta la libreria alsa.