UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Triennale in Ingegneria dell'Informazione

TESI DI LAUREA TRIENNALE

Progettazione e sviluppo software per la riattivazione di installazioni artistiche multimediali interattive. Studio di caso: "Il caos delle sfere" di Carlo De Pirro

Relatore:
Sergio Canazza Targon
Correlatore:
Alessandro Fiordelmondo

Laureando:
Alvise Bolzonella

Anno Accademico 2021/2022

Data di Laurea 16/11/2022

Abstract

A partire dalla fine del XX secolo molte istituzioni, come università e ministeri culturali, hanno iniziato a promuovere azioni di conservazione e riattivazione di opere d'arte interattive con tecnologie, composte cioè da un ampio insieme di mezzi, anche digitali, in cui l'interazione persona-macchina è al centro dell'opera, con lo scopo di renderle fruibili per le generazioni future. L'obiettivo di questo studio è di descrivere il processo che porta alla riattivazione di un installazione artistica con tecnologie e a illustrare i passaggi necessari a una corretta conservazione, seguendo un modello multilivello definito dal Centro di Sonologia Computazionale dell'Università degli Studi di Padova, e utilizzando come studio di caso "Il Caos delle Sfere", un'opera ideata dal compositore Carlo De Pirro verso la fine del XX secolo. Dopo un'iniziale introduzione ai concetti di conservazione e riattivazione, viene descritta la caratterizzazione dell'hardware e del sofwtare utilizzati dall'opera d'arte nella sua concezione originaria, per poi spiegare i passaggi effettuati per la sua riattivazione, con un'attenzione particolare all'algoritmo implementato nell'opera, che è stato sviluppato da zero per sfruttare al meglio le nuove tecnologie utilizzate per la riattivazione e per la conservazione, i cui relativi metadati sono raccolti in un repository gestito dalla piattaforma GitLab.

Indice

Capitolo 1

Introduzione

Verso la fine del XX secolo la tecnologia iniziò a radicarsi sempre di più nella produzione di opere artistiche, dato l'aumento esponenziale del suo uso. Le possibilità d'interazione, che possono riguardare sia la performance (*performed based*) sia l'interazione tra il pubblico e l'installazione (*audience interaction*), spinse gli artisti ad approcciare l'ambito tecnologico attraverso esperimenti o giochi, cercando di creare un interazione uomo-macchina che potesse raggiungere nuovi orizzonti senza cui inimmaginabili senza l'utilizzo della tecnologia [11]. Questi studi sull'interazione diventarono un punto cardine nelle cosiddette *Interactive Multimedia Artworks*[1], che possono essere descritte come opere d'arte composte da un ampio insieme di mezzi in cui appunto l'interazione è al centro dell'installazione.
Questi lavori però venivano concepiti in un contesto molto sperimentale e spesso il loro ciclo vitale era breve, soprattutto a causa della rapida evoluzione della tecnologia negli anni, che rendendo obsoleti i dispositivi e i programmi con cui queste realizzazioni venivano concepite, faceva sì che molte di esse andassero perdute con il tempo.
A differenza di un dipinto, di una scultura o di qualunque altra tipologia di un'opera d'arte classica le realizzazioni artistiche multimediali presentano anche il problema della riproducibilità, dei possibili modi d'interagire con esse oppure della possibilità di un riutilizzo futuro per motivi di studio o implementazioni in altre opere [14].
Per questo si sentì sempre di più l'esigenza di conservare oppure riattivare queste opere d'arte con lo scopo di renderle usufruibili per le generazioni future. Con l'inizio del nuovo secolo molte università, musei così come alcuni ministeri culturali europei iniziarono a comprendere l'importanza di tutto ciò e così si vennero a creare molti progetti e archivi con lo scopo di collezionare opere d'arte contemporanee con diversi e originali approcci.
Ne sono un esempio importante lo ZKM[2] situato a Karlsruhe che conserva una grande collezione di opere audiovisive e progetti di conservazione, l'ADA[3] oppure il database ad accesso gratuito Basis Wien[4].
In Italia si può prendere come esempio il *Protocollo per l'Autenticità, la Cura e la Tutela dell'Arte contemporanea* che presenta delle linee guida e dei principi per la conservazione e l'installazione di opere artistiche multimediali ed è stato istituito dal Ministero per i Beni e le Attività Culturali e per il Turismo nel 2017.[5]
In questa tesi verrà trattato un caso specifico di conservazione e riattivazione basato sul progetto *Il Caos delle Sfere* di Carlo de Pirro attuata mediante una strategia chiamata *modello di conser-*

[1] Installazioni Artistiche Multimediali Interattive

[2] Zentrum fur Kunst und Medien; ZKM and conservation of Media Art https://zkm.de/en/keytopic/conservation-of-media-art

[3] Archive of Digital Art: https://www.digitalartarchive.at/nc/home.html

[4] Basis Wien database https://www.basis-wien.at/db/advsearch?show=advsearchlang=en

[5] https://storico.beniculturali.it/mibac/export/MiBAC/sitoMiBAC/Contenuti/Avvisi/visualizza_asset.html_1054867027.html

vazione multilivello sviluppata al CSC (Centro di Sonologia Computazionale), situato presso l'Università degli studi di Padova, durante dei lavori di conservazione sulle opere multimediali di Carlo de Pirro [1].
In questo primo capitolo verrà introdotto il modello originario di conservazione multilivello adottato al CSC e successivamente verrà trattato nello specifico il modello che è stato utilizzato per lo studio di caso in oggetto.

1.1 La conservazione e la riattivazione di opere con tecnologie

Ma perché è così importante la conservazione e la riattivazione di un'opera d'arte con tecnologia?
Uno degli aspetti fondamentali è sicuramente il fatto che un'artista per la realizzazione di un'idea potrebbe avere la necessità di ulteriori implementazioni nella tecnologia utilizzata, che lo potrebbero portare a far sviluppare quella macchina oltre le potenzialità che erano state ipotizzate fino a quel momento. Per questo le industrie artistiche e tecnologiche sono così correlate tra loro, rendendo necessario uno sviluppo e un supporto reciproco tra i due ambiti. Questo ambito è diventato fondamentale soprattutto negli ultimi anni, in particolare nell'Industria 5.0[6], in cui gli operai insieme alle macchine (sempre più automatizzate per essere "intelligenti") partecipano alla catena produttiva con la loro creatività. Proprio per questo l'interazione con le opere multimediali è uno studio di caso ideale della coesistenza tra uomo e macchina in cui il primo domina la seconda attraverso la propria fantasia e inventività.
In aggiunta la conservazione è fondamentale per tramandare alle generazioni future un background socioculturale di un determinato periodo storico; infatti la perdita di questi lavori potrebbe portare un rallentamento nella maturazione della creatività collettiva e dato che la durata vitale di questi progetti è molto precaria è bene attuare queste strategie di conservazione il prima possibile. È molto interessante in quest'ambito anche osservare come un'opera d'arte tecnologica possa venire, attraverso la conservazione e la riattivazione, continuamente aggiornata e modificata secondo nuove idee dell'artista o di chi la rielabora utilizzando nuove tecnologie ideate nel corso degli anni. Si può prendere come esempio l'opera *Very Nervous System* di David Rokeby, con l'artista che ha ripetutamente modificato la sua installazione (un sistema che genera dei suoni in base ai movimenti di uno spettatore in una stanza chiusa) nel corso dei decenni utilizzando diversi luoghi e configurazioni [13].
Infine è molto rilevante anche l'ambito economico, infatti come tutte le opere d'arte anche le installazioni tecnologiche una volta esibite richiamano molto pubblico, facendo girare l'economia dei vari stati in cui vengono esposte con un impatto molto positivo in tutto l'ambiente artistico. Per questo motivo negli ultimi anni le istituzioni stanno supportando molto lo sviluppo di nuovi progetti atti alla conservazione di queste opere.

[6] l'Industria 5.0 è un modello d'impresa caratterizzato dalla cooperazione uomo-macchina, con l'obiettivo di dare un valore aggiunto alla produzione creando prodotti personalizzati che rispettino le esigenze dei consumatori e anche l'ambiente. www.research-and-innovation.ec.europa.eu/research-area/industrial-research-and-innovation/industry-50_en

1.2 Introduzione al modello di conservazione del CSC

Il significato più radicato di un'installazione con tecnologia risiede quindi nell'interazione che si viene a creare tra le persone, l'ambiente in cui viene sviluppato e testato e i componenti che fanno sì che funzioni tutto a dovere.
A differenza quindi di un opera d'arte classica in cui basta semplicemente conservare la realizzazione stessa (sia essa un quadro, una statua ecc.), la conservazione di un'opera d'arte con tecnologie è molto differente.

Data la definizione di conservazione come insieme di tutti i passaggi necessari per assicurarsi che la realizzazione, insieme a tutta la sua documentazione, sia permanentemente accessibile [5], dobbiamo innanzitutto avere chiaro dove l'opera d'arte in se è situata nel caso di una realizzazione con tecnologie, cioè solitamente nel processo d'interazione tra l'uomo ed essa.
In questo senso quindi la conservazione è un continuo divenire, non può essere fatta in un determinato istante ma dev'essere una cosa che possa evolversi nel tempo. Il CSC per questo adotta un modello, definito multilivello, che si basa appunto su diversi livelli di conservazione in cui abbiamo due forme di conservazione: essa può essere statica, dove tutti i dati una volta salvati non vengono più modificati, oppure dinamica dove tutto può subire dei cambiamenti continui.
I vari livelli della conservazione, le cui caratteristiche fondamentali sono riepilogate nella figura 1.1, possono essere quindi descritti in questo modo:

Bit - Fanno parte di questo livello tutte le componenti del lavoro originale che possono essere conservate senza apporre modifiche, tutti i dati che ne fanno parte devono quindi essere mantenuti inalterati. Questa è una fase della conservazione fisica e statica, che può contenere solo oggetti fisici e digitali, e tutto il materiale digitale dev'essere conservato assieme a una descrizione riguardo al suo funzionamento e ai passaggi attraverso cui è avvenuta questa conservazione. Vanno poi fornite le informazioni su come questi oggetti si relazionano tra loro fisicamente e temporalmente nell'utilizzo dell'opera così che chiunque in un futuro possa essere in grado di ricomporla e riutilizzarla.

Data - In questo livello sono invece presenti tutte le informazioni tecniche riguardanti la realizzazione e sul suo funzionamento. Questo è uno stato fisico e dinamico e potrebbe tornare necessario per sviluppi futuri come ad esempio la riproposizione dell'opera con nuovi linguaggi.

Record - In questo livello bisogna conservare tutti gli elementi che nel tempo sono stati modificati o aggiornati rispetto all'opera d'arte concepita originariamente, incluse le interpretazioni che sono state date dallo sviluppatore nei riguardi della realizzazione originale. Questo è uno stato logico e statico, in cui ogni modifica prima di essere apportata definitivamente dev'essere verificata rigorosamente con gli strumenti necessari. Questo livello necessita anche di un ulteriore sottolivello contenente le informazioni sul funzionamento dell'opera in quel determinato momento dopo averci apportato le modifiche descritte.

Experience - In quest'ultimo livello va conservata tutta la documentazione che prova l'esistenza e il funzionamento dell'installazione. Vanno inclusi quindi hardware, software, interviste, registrazioni audiovisive dell'opera in funzione, i test effettuati sul sistema originale e tutte le informazioni riguardanti le persone che ci hanno contribuito (compositori, tecnici, artisti) specificando il ruolo in cui sono stati coinvolti nella realizzazione. Tutti questi documenti fanno si che si possa tenere traccia di tutta la storia dell'installazione, dalle sue origini fino al momento attuale in cui viene visionata. Questo stato è logico e dinamico, è necessario per l'utilizzo di un installazione dopo un lungo lasso di tempo (> 30 anni).

Questi livelli non richiedono una sequenzialità tra di loro dato che hanno come obiettivo la rappresentazione di diverse sfaccettature dell'opera, possono invece anche avere occasionalmente in comune tra loro alcuni elementi che fanno riferimento a più di un livello del modello.

Livello di preservazione	Statico/ Dinamico	Fisico/Logico	Aspettativa vitale (Anni)
Preservazione dei Bit	Statico	Fisico	5 - 10
Preservazione dei Data	Dinamico	Fisico	> 30
Preservazione dei Record	Statico	Logico	10 - 20
Preservazione delle Experience	Dinamico	Logico	> 30

Figura 1.1: *Caratteristiche dei livelli di conservazione del modello CSC*

1.3 Introduzione allo studio di caso: Il Caos delle Sfere

Il Caos delle sfere è un'installazione musicale interattiva che è stata presentata per la prima volta alla "Biennale of Young Artists from Europe and the Mediterranean" a Roma nel 1999; successivamente a questa mostra l'opera è stata esposta in molte altre manifestazioni artistiche fino al 2004. L'opera è stata concepita da Carlo de Pirro, con l'apporto scientifico e tecnico di Nicola Orio e Paolo Cogo, presso il CSC dell'università degli studi di Padova (nella figura 1.2 si può osservare l'opera nella sua completezza durante la sua fase di sviluppo); il lavoro si basa sui risultati di una ricerca sull'interattività della musica chiamata *"Controlled Refractions"* [8] riguardante l'iterazione tra un pianista e un computer attraverso la performance musicale.
Il concetto di base dell'opera è quello di utilizzare un comunissimo gioco elettronico, in questo caso un flipper, per controllare il comportamento di un pianoforte automatico meccanico, denominato Disklavier, collegato ad esso. L'utilizzo di un flipper rende molto aleatorio il tutto in quanto l'unico controllo che ha il giocatore durante l'esperienza è quello sulla pallina che in questo caso è un controllo abbastanza debole dato che c'è un alta imprevedibilità nei suoi movimenti. L'idea dei compositori era quella di evitare che i suoni venissero ripetuti, quindi evitando che il Disklavier suonasse la stessa serie di note ogni volta che veniva colpito un determinato elemento del gioco. In più lo scopo dell'opera è quello di premiare i giocatori più abili facendo si che man mano che si avanzasse nel gioco, e quindi nei livelli, le composizioni suonate dal Disklavier fossero sempre più articolate e meno banali. La partita per questo motivo partirà con delle sequenze prescritte, al raggiungimento di un nuovo livello il Disklavier inizierà a suonare delle composizioni generate automaticamente controllate parzialmente dal tipo di bersagli colpiti durante il gioco. Man mano che si cambiano i livelli il tipo di sequenze automatiche e il modo in cui queste vengono manipolate dal giocatore durante la partita variano aumentando la diversità delle composizioni.

Per l'installazione è stato scelto di utilizzare il flipper elettronico chiamato *"Creature from the Black Lagoon"* (rappresentato dalla figura 1.3) basato sull'omonimo film e prodotto dal 1992 dall'azienda Midway Games Inc [4]; questa scelta è stata fatta per il fatto che questo è uno dei primi flipper che introduceva il concetto di partita in più livelli, con questi che variavano in base alle azioni del giocatore durante la partita, caratteristica che è fondamentale per rispecchiare le caratteristiche dell'opera.

Figura 1.2: *L'installazione "Il Caos delle Sfere" durante il suo sviluppo presso l'abitazione di Carlo De Pirro nel 1999. Da sinistra a destra: Veniero Rizzardi, il compositore Carlo De Pirro e i collaboratori tecnici del CSC, Paolo Cogo e Nicola Orio. Il flipper si trova sulla destra, affianco al monitor del computer e al Disklavier.*

Figura 1.3: *Immagine dettagliata del flipper "Creature from the Black Lagoon"*

Capitolo 2

Studio di caso: Il caos delle sfere

2.1 Il modello tecnico dell'opera originale

Per poter presentare al meglio gli elementi che compongono questa installazione, li suddivideremo in nodi, cioè degli insiemi di elementi (hardware e software) che interagiscono tra di loro in una correlazione continua per raggiungere uno scopo comune. Nel nostro studio di caso possiamo identificare tre nodi cardine, rappresentati nella figura 2.1. Essi sono:

Nodo interattivo - Si può identificare con il flipper in quanto rappresenta tutti i segnali provenienti da esso e li elabora portando il sistema a selezionare le giuste sequenze che verranno poi riprodotte e generate.

Nodo comunicativo - Si può identificare con il computer, esso riceve (oppure genera in tempo reale) le sequenze dal nodo precedente e crea i corrispondenti eventi MIDI.

Nodo riproduttivo - Quest'ultimo nodo si può identificare con il Disklavier, che riceve gli eventi MIDI da quello precedente e li utilizza per riprodurre i suoni.

I tre nodi interagiscono tra di loro attraverso due diverse interfacce: una porta parallela DB25 attraverso cui comunicano i primi due nodi, e un interfaccia MIDI in cui comunicano il nodo comunicativo e quello riproduttivo.

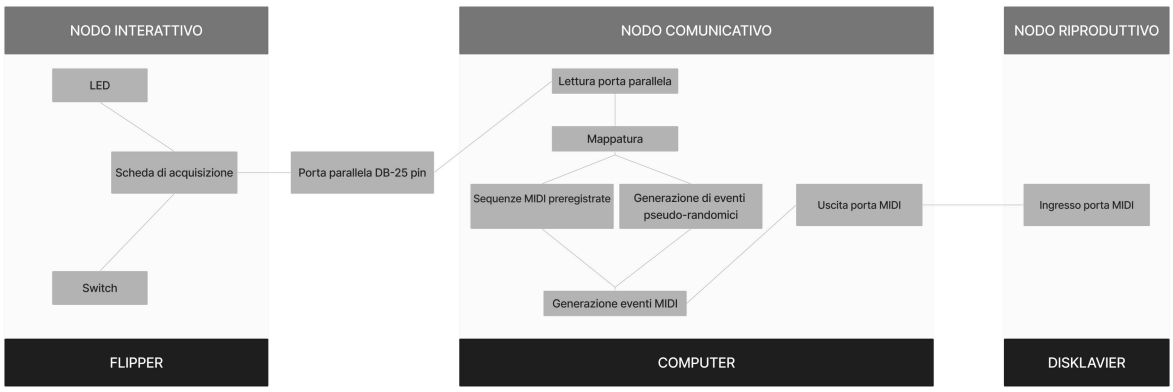


Figura 2.1: *Schema rappresentante la struttura a nodi dell'installazione "Il Caos delle Sfere"*

Figura 2.2: *Zoom sulla parte centrale del campo di gioco del flipper, in basso si può notare la scritta FILM che bisogna far illuminare per portare il gioco in modalità Extra-Ball.*

2.1.1 Il nodo interattivo

Il nodo interattivo prende il nome dal fatto che è l'insieme di tutti gli elementi con cui può interagire l'utilizzatore dell'opera in modo da raggiungere il risultato prefissato, cioè la produzione di sequenze musicali suonate dal Disklavier. Tutti gli elementi di questo nodo hanno come punto comune la presenza del pinball, dato che è proprio l'elemento d'interazione uomo-macchina e tutti i dati prodotti da esso vengono elaborati da una scheda d'acquisizione su misura che li rende usufruibili dalla porta DB25 e dal secondo nodo, quello comunicativo.
Come già introdotto precedentemente il flipper utilizzato, denominato *Creature from the Black Lagoon* è stato scelto per la sua struttura a più livelli. Nello specifico lo scopo del gioco è quello di attivare le quattro lettere che compongono la scritta "F-I-L-M" (posta come si può vedere nella figura 2.2 al centro del campo di gioco del flipper) colpendo i vari bersagli. A quel punto del gioco si raggiunge una fase del gioco chiamata *Extra-Ball* (EX) in cui si ha in gioco più di una pallina. In questa fase della partita il giocatore ha altri tre obiettivi: Deve inizialmente trovare (CS) la Creatura che può essere in tre diverse posizioni; successivamente deve salvare (RE) la ragazza per arrivare infine a conseguire il jackpot (JK) trovando la giusta posizione nel gioco dove la Creatura la tiene nascosta [4].

Sul piano tecnico il problema principale era il rilevare il punto della partita interfacciandosi alla tecnologia integrata nel flipper. Per questo Paolo Cogo sviluppò una scheda di acquisizione per far si che tracciasse i bersagli colpiti dalla pallina e le luci associate alle lettere della parola FILM e ai quattro eventi EB RC SR JK descritti poc'anzi. Si può notare però che in questo modo non si acquisiscono risultati riguardanti altre caratteristiche del flipper, come ad esempio il numero dei punti totali, ritenute non funzionali ai fini dell'installazione.

2.1.2 Il nodo comunicativo

Questo nodo viene chiamato così perché è quello che permette la comunicazione tra il flipper e il Disklavier. Esso è composto interamente dal computer originale dove gli sviluppatori del progetto hanno creato diversi programmi che avevano il compito di comunicare con il flipper, con il Disklavier e di processare, generare e modificare le sequenze musicali che poi verranno riprodotte. Quest'ultime in particolare riguardano l'ambiente *MidiShare* e si occupano di generare gli eventi MIDI.
Il computer originale presenta il sistema operativo Windows 95, I tecnici del CSC per farlo comunicare con il flipper hanno sviluppato una scheda dedicata che presentasse l'ingresso di una porta DB25, data l'assenza di quel tipo di connettore nei pc dell'epoca. In più è stata aggiunta anche una porta MIDI per poter trasferire i dati tra questo livello e quello riproduttivo. Il pc ha anche il compito di contenere una varia selezione di file *.wri[1] dove si trovano molte sequenze di eventi MIDI che erano state composte da Carlo De Pirro apposta per la composizione. Questi file venivano poi inseriti in un programma che generava a partire da essi le corrispondenti sequenze MIDI che venivano Inviate al Disklavier.
Tutti gli eventi MIDI devono essere processati per poter essere riprodotti. Per questo essi vengo inviati a un sistema open-source chiamato MidiShare[2] (la cui routine di esecuzione è rappresentata nella figura 2.3) che è in grado di attivare una comunicazione tra l'interfaccia MIDI e la porta MIDI. Le questo programma è stato utilizzato soprattutto per alcune sue importanti funzioni, in particolare quelle di poter controllare il flusso degli eventi MIDI in tempo reale con un refresh dei dati prossimo al millisecondo e quello di poter programmare gli eventi MIDI da far riprodurre al Disklavier indicandogli il momento esatto in cui essi dovevano iniziare [9].
Quest'ultima funzione è fondamentale per l'installazione in quanto abbiamo bisogno di programmare le diverse sequenze di eventi MIDI una dopo l'altra in modo da avere un susseguirsi fluido di suoni senza pause. Un'altra funzione di MidiShare è quella di poter svuotare in un qualsiasi istante la coda di eventi MIDI programmati per un tempo futuro in modo da riuscire, nel momento in cui cambia un livello oppure nel momento in cui la partita giunge al termine d'inviare nuovi eventi correlati al punto del gioco raggiunto da chi sta interagendo con l'opera o altrimenti nel secondo caso a fermare la riproduzione di suoni da parte del Disklavier [6].

2.1.3 Il nodo riproduttivo

Il nodo riproduttivo è dedicato alla riproduzione della performance musicale, che è comandata dall'algoritmo contenuto nel pc e dall'evoluzione della partita giocata con il flipper dall'utilizzatore dell'opera. Qui gli eventi MIDI inviati da MidiShare vengono convertiti in suoni udibili dal giocatore che può così ascoltare i risultati del suo operato. In questo caso l'unico elemento appartenente a questo nodo è il Disklavier, che è lo strumento scelto per riprodurre le sequenze musicali preregistrate o generate dall'algoritmo. Il tipico Disklavier può essere descritto come un pianoforte automatico motorizzato che contiene dei sensori elettronici utili alla registrazione e dei solenoidi con lo scopo di far suonare il Disklavier in playback, senza intervento umano. Il Disklavier può essere utilizzato quindi per registrare grazie ai sensori delle sequenze musicali che poi possono venire salvate in molti formati, principalmente seguendo i protocolli MIDI. Solitamente questi dispositivi vengono usati per un uso didattico, con gli studenti che registravano le proprie esercitazioni per poi riascoltarsi o far correggere errori dai propri insegnanti, oppure in ambito ingegneristico dato che il suo comportamento può essere controllato in molte maniere diverse, come ad esempio attraverso dispositivi portatili o da remoto attraverso dei controller wi-fi.

[1] Microsoft Write Documents
[2] Sviluppato da Grame nel 1989 per fornire un sistema di sviluppo per la musica in tempo reale

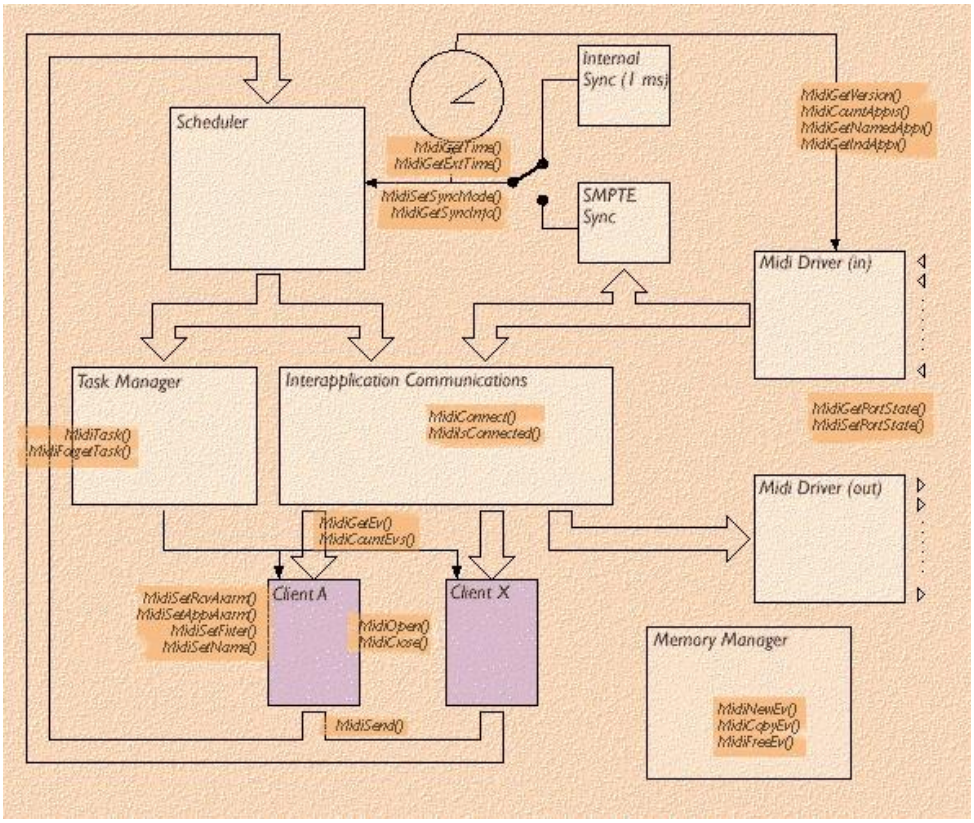


Figura 2.3: *Schema che rappresenta una routine eseguita da MidiShare così come viene rappresentata dal suo sviluppatore nella documentazione originale*

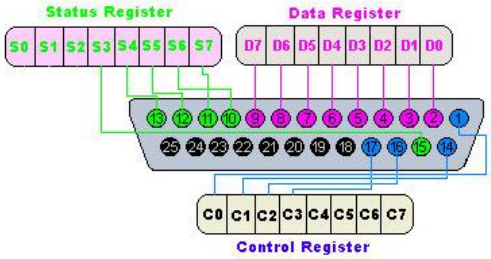
2.1.4 Interfacce di comunicazione tra i nodi

Il team di sviluppo ha successivamente dovuto affrontare il problema della comunicazione tra i vari nodi. Gli obiettivi a livello tecnico erano il far si che la scheda di acquisizione fosse il più semplice possibile (a livello elettronico) e l'evitare dei delay nella comunicazione tra i vari nodi per non far si che il Disklavier interrompesse le sequenze musicali per colpa di ritardi nella trasmissione dati. Per questi motivi si è scelto di adottare una porta parallela a 25 pin nella comunicazione nodo interattivo-nodo comunicativo, dato che il pc utilizzava il sistema operativo Windows 95 che dava la possibilità di accedere direttamente ai contenuti collegati a esso tramite porte seriali e parallele. Per la trasmissione d'informazioni tra il nodo comunicativo e quello riproduttivo è stata scelta invece una porta MIDI, ottima per sfruttare al meglio tutti i vantaggi dell'interfaccia MidiShare.

Porta parallela DB25

L'interfaccia di una porta parallela segue le linee guida stabilite dallo standard IEEE [12] nel 1994 in cui veniva definita come un metodo di trasmissione di segnale per comunicazioni in parallelo asincrone e bidirezionali tra due dispositivi (ad esempio pc e stampanti). Molte porte parallele utilizzano un connettore a 25 contatti, il cui schematico è rappresentato nella figura 2.4. Gli input ed output della porta parallela sono caratterizzati da dei TTL[3] La caratterizzazione di ogni pin, il cui nome rispecchia il tipo di dato che acquisisce, può essere osservata nella figura 2.4. Nella tabella la lettera "n" posta in testa al nome del segnale indica che il livello logico di quel segnale è invertito, cioè che si attiva alla tensione a cui gli altri bit si disattivano. Le porte

[3]Transistor-transistor logic, sono una famiglia di transistor a giunzione bipolare che hanno sia la funzione logica che di amplificazione. Sono stati introdotti nel 1963 dalla Sylvania Electric Products Inc.



Pin No (DB25)	Signal name	Direction	Register - bit	Inverted
1	nStrobe	Out	Control-0	Yes
2	Data0	In/Out	Data-0	No
3	Data1	In/Out	Data-1	No
4	Data2	In/Out	Data-2	No
5	Data3	In/Out	Data-3	No
6	Data4	In/Out	Data-4	No
7	Data5	In/Out	Data-5	No
8	Data6	In/Out	Data-6	No
9	Data7	In/Out	Data-7	No
10	nAck	In	Status-6	No
11	Busy	In	Status-7	Yes
12	Paper-Out	In	Status-5	No
13	Select	In	Status-4	No
14	Linefeed	Out	Control-1	Yes
15	nError	In	Status-3	No
16	nInitialize	Out	Control-2	No
17	nSelect-Printer	Out	Control-3	Yes
18-25	Ground	-	-	-

Figura 2.4: *Rappresentazione della configurazione dei pin di una porta parallela 25-DB*

parallele standard presentano tre registri a 8-bit definiti come *registro dati, registro degli stati e registro di controllo*. La caratterizzazione specifica di ogni pin può essere trovata nel seguente documento [10]. Il modello standardizzato presenta 5 diversi modi di comunicazione tra l'host e la periferica, chiamati *Compatibility Mode, Byte Mode, Nibble Mode, Extended Capabilieies Port Mode e Enhanced Parallel Port Mode*. Verrà trattato solo il *Nibble mode* in quanto è quello utilizzato per la comunicazione flipper-PC.

Nibble Mode Come detto prima il *Nibble Mode* è uno dei modi di comunicazione utilizzati dalla porta parallela, consiste nella lettura di 8 bit di dati senza però utilizzare i data bit (Data-0/Data-7) così da lasciare ai dispositivi collegati la possibilità di leggere e scrivere attraverso essi. Per questo implementa la lettura di gruppi di 4 bit (definiti *nibble*) alla volta. La Nibble mode consiste in due fasi: una fase di trasferimento dati (*Data Transfer phase*) e un secondo stadio denominato *Idle phase* che definisce gli stati delle porte quando non stanno avvenendo trasferimenti di dati. La fase di trasferimento dati può essere riassunta nei seguenti passaggi:

1. L'host abilita la ricezione dei dati settando il pin di controllo[4] a basso livello;
2. La periferica trasferisce il primo nibble attraverso i pin di stato S-3, S-4, S-5 ed S-7
3. Successivamente la periferica indica che il nibble è stato inserito con successo portando il pin S-6 a basso livello;
4. L'host quando ha ricevuto il nibble setta il pin Control-1 ad alto livello per indicare la fine della ricezione ma che non è ancora in grado di poter ricevere nuove informazioni;
5. La periferica setta il pin S-6 ad alto livello come conferma per l'host;
6. Vengono ripetuti i primi 5 passaggi per il secondo nibble.

Dopo aver ricevuto l'intero byte d'informazione l'host setterà *HostBusy* a basso livello se deve ricevere altri dati o ad alto livello se vuole evitare di ricevere ulteriori informazioni dalla periferica. Successivamente un software si occuperà di ricreare il byte a partire dai due nibble, per questo motivo la nibble mode, anche se molto più semplice da implementare, è più lenta rispetto ad altre modalità di comunicazione.

[4]Per definizione il pin n°14 denominato Control-1 e come si può leggere dalla figura 3.4 logicamente invertito

Nome del pin	Funzione del pin
Status-4	Se sta leggendo il primo nibble controlla lo stato della luce riferita alla lettera “F” se è il secondo riferita a “EX”
Status-5	Se sta leggendo il primo nibble controlla lo stato della luce riferita alla lettera “I” se è il secondo riferita a “JK”
Status-6	Se sta leggendo il primo nibble controlla lo stato della luce riferita alla lettera “L” se è il secondo riferita a “RE”
Status-7	Se sta leggendo il primo nibble controlla lo stato della luce riferita alla lettera “M” se è il secondo riferita a “CS”
Control-0	Determina quale dei due nibble è letto in un determinato momento
Data-0 / Data-7	Vengono utilizzati per determinare qual è l’ultimo bersaglio del flipper colpito dalla pallina

Figura 2.5: *Rappresentazione delle funzioni associate a ciascun pin della porta parallela 25-DB*

Applicazione della comunicazione nibble mode al flipper Come descritto precedentemente nella sezione 2.1.1, le due caratteristiche trasmesse dal pc al computer sono lo stato delle luci (accese o spente) che serve a determinare il livello della partita, e l’ultimo bersaglio colpiti dalla pallina. Analizzando il software dell’installazione originale si può notare come il metodo di acquisizione utilizzato seguisse la *Nibble Mode*, e che per inviare al computer i dati seguisse i seguenti passaggi:

1. Il PC acquisisce il primo nibble, che contiene i dati riguardanti lo stato delle luci che compongono la parola “FILM”;
2. Il PC porta al livello logico alto il primo pin (Control-0) per far si che venga predisposta la lettura del secondo nibble;
3. Il PC acquisisce il secondo nibble, che contiene lo stato delle luci che corrispondono alle fasi finali del gioco (“EX”, “CS”, “RE” e “JK”);
4. Il PC legge attraverso i pin che trasportano i dati (Data-0/Data-7) per acquisire il valore dell’ultimo oggetto colpito dalla pallina;
5. Il PC porta il primo pin al livello logico basso per far si che venga selezionata l’acquisizione del primo nibble relativo al successivo controllo delle luci.

Nella figura 2.5 si possono vedere tutte le funzioni dei singoli pin, il pin Status-7 essendo invertito logicamente deve però verificare la condizione opposta rispetto agli altri pin di stato per il controllo dell’accensione della luce associata.

Tecnologia MIDI

Per descrivere al meglio il funzionamento della porta MIDI, che come detto precedentemente governa la trasmissione d’informazioni tra il nodo comunicativo (rappresentato dal flipper) e quello riproduttivo (il Disklavier) è necessario descrivere tutti gli elementi che compongono la tecnologia MIDI cioè l’hardware da cui è composta, la struttura dei dati che caratterizzano un messaggio MIDI e la descrizione di un evento MIDI (elemento necessario per la temporizzazione dei messaggi MIDI).
La tecnologia MIDI è stata sviluppata e standardizzata dall’*MMA*[5] nel 1989 e oggi è ancora

[5]MIDI Manufacturers Association

alla base della maggior parte delle installazioni che prevedono una connezssione tra computer, dispositivi audio e strumenti musicali.
Le tecnologie MIDI sono essenziali in questo ambito perché forniscono una rappresentazione operazionale della musica, cosa fondamentale dato che essa è il mezzo multimediale maggiormente utilizzato per la creazione di questo specifico tipo di opere d'arte. Pertanto abbiamo bisogno di un mezzo di trasmissione per la musica che ci permetta di salvarla in un dispositivo per poi analizzarla, elaborarla e trasmetterla. Questo è possibile attraverso il salvataggio di campioni audio (che possono essere conservati ad esempio con le codifiche audio WAV o MPEG-1) oppure mediante una rappresentazione strutturale contenente tutte le informazioni sulla composizione di una determinata sequenza musicale. Questo metodo di rappresentazione può essere a sua volta di due tipologie:

la prima, che viene definita operazionale, in cui vengono descritti la sequenza temporale e la descrizione fisica dei suoni da cui è composta;

la seconda, definita simbolica, utilizza appunto dei simboli dettagliati per descrivere l'aspetto di questi suoni e permette molta più libertà d'interpretazione rispetto alla precedente.

Lo standard MIDI fornisce delle sequenze la cui composizione è definita in modo operazionale. Innanzitutto il protocollo MIDI fornisce le indicazioni riguardanti le modalità di collegamento tra tutti i possibili dispositivi, un setup per i circuiti elettronici che caratterizzano la porta MIDI e le specifiche riguardanti il bitrate e il tipo d'informazioni che caratterizzano il flusso di dati. Il punto cardine di questo standard è la caratterizzazione delle specifiche del messaggio MIDI, che viene utilizzato all'interno di ogni dispositivo messo in comunicazione attraverso la porta MIDI per generare la musica. La descrizione esatta di questo messaggio è determinata dal *General MIDI*, una standardizzazione delle specifiche necessarie agli strumenti musicali elettronici per poter ricevere e generare questo tipo di dati.
L'ultima parte del protocollo midi descrive la struttura degli SMF (*Standard MIDI File*), che determinano il modo in cui queste sequenze di dati possono venire immagazzinate all'interno dei vari dispositivi. Solitamente queste sequenze musicali vengono salvate all'interno dei *MIDI files*, contraddistinti dal formato (*.mid), che contengono le sequenze di messaggi MIDI insieme alle loro informazioni di riproduzione, come ad esempio i vari momenti temporali in cui devono essere eseguite e le modalità in cui devono essere riprodotte da uno strumento.

L'hardware dello standard MIDI Per quanto concerne l'ambito tecnologico, lo standard MIDI presenta un'interfaccia seriale attraverso cui i messaggi sono inviati come una sequenza di bit, con l'invio che avviene solamente quando questi dati sono disponibili nel dispositivo che li sta generando. Il bitrate impostato per l'invio dei messaggi è di 31250 $\frac{bit}{s}$, ed essendo essi delle parole composte da 10 bit, per una trasmissione completa sono necessari 320 μs. I Nella figura 2.6 si possono vedere le funzioni specifiche dei 5 pin che compongono i connettori che vanno poi a collegarsi a tre diversi tipi di porte MIDI (tutti presenti in un qualsiasi strumento compatibile con lo standard MIDI), che possono essere:

MIDI In - Porta attraverso cui si ricevono i messaggi MIDI, che successivamente verranno interpretati;
MIDI Out - Porta attraverso cui si inviano i messaggi MIDI ad altri dispositivi;
MIDI Thru - Porta attraverso cui si può inviare una copia del messaggio ricevuto attraverso il MIDI In a un altro dispositivo, è utile per collegare più strumenti in cascata.

Figura 2.6: *Rappresentazione delle funzioni associate a ciascuno dei cinque pin che compongono una porta MIDI*

Caratterizzazione di un messaggio MIDI Un messaggio MIDI non rappresenta l'evoluzione sequenza musicale, infatti ogni messaggio contiene una serie d'informazioni di controllo che indicano al dispositivo che lo riceve attraverso la porta MIDI Out l'evento musicale da eseguire, come ad esempio la pressione di una nota, o il settaggio di alcune funzioni dello strumento (come ad esempio la pressione di un pedale del Disklavier).
Come detto poco fa, i messaggi MIDI sono composti da una serie di sequenze lunghe 10 bit (definite *bytes*), la cui struttura si può suddividere così: un primo bit che indica l'inizio della sequenza, un bit finale che ne indica la conclusione e otto bit intermedi che trasportano le informazioni riferite al singolo byte. Il primo byte che compone un messaggio midi viene definito byte di stato, e si differenzia dai byte che trasportano i dati (*Data bytes*) per il valore del primo bit. I byte di stato infatti presentano sempre il bit più significativo uguale ad 1 e servono a fornire ai dispositivi delle istruzioni di controllo, come ad esempio il numero di byte che compongono il messaggio, oppure le informazioni riguardanti il numero del canale da utilizzare per la trasmissione dei dati.
I canali si possono definire come le diverse vie di comunicazione a disposizione del protocollo MIDI, infatti diversi canali vengono usati per comunicare con diversi strumenti. Lo standard MIDI specifica 16 possibili canali di comunicazione, tutti a disposizione attraverso un singolo cavo, e ciò rende possibile il controllo delle funzioni di diversi strumenti attraverso una sola piattaforma MidiShare. In aggiunta alla canalizzazione il byte di stato può classificare il messaggio in messaggio di sistema, che viene inviato a tutti i dispositivi connessi all'interfaccia qualunque sia il loro canale, oppure in messaggio di canale, che appunto viene inviato solo attraverso un determinato canale. Un messaggio di sistema trasporta quindi tutte le informazioni che non sono associate a un canale specifico, come ad esempio la temporizzazione della sequenza MIDI e le informazioni di setup iniziale per i dispositivi in ricezione. La struttura dei byte di stato e di trasporto dati è evidenziata nella figura 2.7, dove *S* sta a indicare il bit iniziale, ed *E* quello finale. Avendo 16 canali a disposizione, i bit che governano la selezione del canale saranno 4, mentre nei Data byte ho 7 bit a disposizione, per cui avrò 127 possibili scelte di parametri da inviare.

I messaggi di canale possono essere a loro volta *Channel Voice Messages* o *Channel Mode Messages*. I primi contengono le informazioni relative all'evento musicale da compiere[6]. Nel nostro studio di caso solo tre di questi tipi di messaggi vengono utilizzati, e sono:
Note On e Note Off - Per il protocollo MIDI la pressione di una nota e il rilascio di essa sono due eventi separati. Nel caso in cui volessi suonare una nota devo trasmettere il comando Note On attraverso la porta MIDI Out; il byte di stato del messaggio MIDI conterrà il valore 1 nei bit indicati con la lettera "x" e il numero di canale attraverso cui trasmettere nei bit indicati con la

[6]Un voice message può essere dei seguenti tipi: Note On, Note Off, Polyphonic Key Pressure, Channel Pressure, Pitch Bend Change, Program Change e Control Change

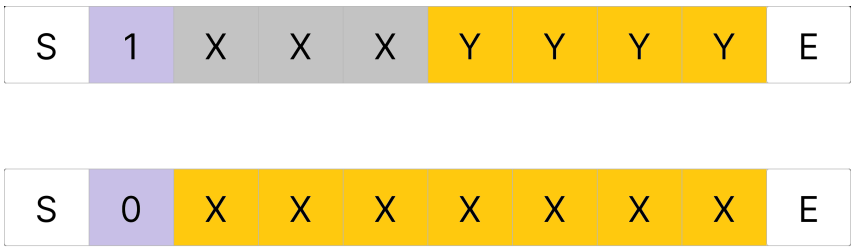


Figura 2.7: *Rappresentazione della composizione dei due tipi di byte che caratterizzano un messaggio MIDI. Nella tabella superiore è rappresentata la composizione di un byte di stato e in quella inferiore un Data byte.*

lettera "y". Il messaggio conterrà anche due Data byte che specificheranno la singola nota da suonare e la velocità a cui essa andrà suonata. Tre byte dello stesso tipo verranno inviati quando la nota dovrà essere rilasciata, con l'eccezione che per trasmettere il comando Note Off il byte di stato conterrà il valore 0 nei bit indicati dalla lettera "x";
Control Change - Se il byte di stato presenta il numero 11 nei tre bit indicati nella figura con la lettera "x" significa che quel determinato messaggio ha il compito d'impostare i parametri di un determinato oggetto che svolge determinate funzioni (sia esso una manopola, un cursore, un pedale ecc.) che si differenziano dal semplice azionamento di una nota; Il numero dell'oggetto da manipolare è contenuto nel primo Data byte e in aggiunta a esso nel secondo data byte è contenuta l'azione da eseguire, che può essere 0 (OFF) o 1 (ON) per uno switch oppure un valore continuo tra 0 e 127 per tutti gli altri tipi di oggetti.

Gli eventi MIDI: come temporizzare una sequenza di messaggi Come detto precedentemente una funzionalità sostanziale che ha fatto optare per l'utilizzo della tecnologia MIDI è quella della temporizzazione delle sequenze inviate in output, potendo controllare il momento d'inizio della riproduzione in ogni momento. La caratteristica temporale però non è descritta nei dati che vengono inviati dai messaggi MIDI e per questo essi devono essere sempre inviati al dispositivo che dovrà riprodurli mediante l'interfaccia MIDI.
la temporizzazione degli eventi MIDI è caratterizzata dall'unità di tempo del *tick*, che deriva dal concetto di *beat*. Il *beat* è l'unita di tempo fondamentale nell'ambito musicale, descrive il modo in cui un musicista conta le note per tenerle a tempo, senza far si che si desincronizzino tra loro. Il tempo nella musica si può misurare in BPM (*beats per minute*), unità di misura che descrive la velocità della sequenza musicale in una composizione con la durata di una singola nota che può essere denotata con il numero di beat che la compongono. Quando lavoriamo con una sequenza di note con la tecnologia MIDI, il tempo musicale si denota in frazioni del beat, appunto i tick, con il loro numero per beat che può essere variato. Ad ogni messaggio MIDI che compone un SMF viene quindi assegnato il numero di tick che passano dal messaggio precedente, così da temporizzarlo nella sequenza di riproduzione. Quando lavoriamo con messaggi MIDI (come ad esempio le sequenze generate dall'algoritmo) che non compongono un SMF il tempo va specificato in microsecondi. Ad ogni messaggio MIDI va quindi associato un campo aggiuntivo (denominato MIDI *delta time*) che specifica il tempo in millisecondi che separa due eventi successivi, l'unione del delta time e del messaggio andranno poi a comporre l'evento MIDI. Tutti gli eventi contenuti nei file *.wri possono essere descritti dall'unione di quattro elementi: il delta time, il tipo di messaggio, la sua tonalità e la sua velocità. Tutte le sequenze preregistrate sono infatti composte da delle righe di quattro numeri interi che rappresentano questi quattro numeri. Gli eventi MIDI per andare in esecuzione vengono trattati da due funzioni di libreria denominate *MidiTask* e *MidiDTask*, che regola la loro trasmissione attraverso la porta MIDI facendo si che essi rispettino i tempi indicati.

2.2 L' algoritmo utilizzato per l'opera originale

Il punto cardine dell'opera si può sicuramente identificare con il nodo comunicativo, che contiene il pc sul quale è stato sviluppato l'algoritmo che governa la comunicazione tra flipper e Disklavier. Uno degli scopi principali dei ricercatori del CSC era quello di ridurre allo stretto necessario i mezzi digitali in grado di comunicare con l'utilizzatore esterno, per questo è stato deciso d'implementare un algoritmo che non avesse bisogno di nessuna interfaccia grafica.
Per fornire una descrizione dettagliata del funzionamento dell'algoritmo (che era stato scritto in linguaggio C) è stata necessaria un'analisi dettagliata del codice sorgente utilizzato a causa della mancanza di documentazione riguardante esso fornita originariamente dagli sviluppatori. La causa della mancanza di documentazione si può imputare al fatto che il codice è stato sviluppato a stretto contatto con il compositore Carlo De Pirro attraverso un approccio *try-and-error*[7], che dati i vari tentativi ha portato a non avere una traccia di come è stato creato il codice. Un ulteriore problema è dato dal fatto che nel disco fisso del computer originale abbiamo ritrovato più di dieci versioni del codice, generalmente che apportavano modifiche sostanziali l'una rispetto all'altra. Tutto ciò rende la riattivazione dell'opera *"Il Caos delle Sfere"* molto complessa dal punto di vista ingegneristico, come vedremo nel paragrafo successivo.
Elemento comune che abbiamo potuto evidenziare tra le diverse righe di codice è il fatto che l'algoritmo si comporta in base all'evoluzione della partita e ai diversi bersagli colpiti durante essa. L'algoritmo infatti è stato diviso in diversi livelli[8], ciascuno dei quali avanzando fa si che le melodie prodotte dal Disklavier fossero sempre più complesse e coinvolgenti per l'utilizzatore dell'opera. Gli eventi MIDI generati dal codice possono essere di due tipi:
Essi possono essere delle sequenze prescritte, generate da delle melodie del compositore Carlo de Pirro e contenute in dei file (*.wri). Alternativamente possono essere delle sequenze generate in tempo reale la cui composizione e durata è dipendente dallo stato di avanzamento nel gioco o dagli oggetti colpiti dalla pallina; per riferirsi a queste sequenze si utilizza il termine *Refractions*, che indica un insieme di accordi o sequenze musicali che possono essere eseguite in tempo reale da una qualsiasi installazione.
Queste *Refractions* nel nostro studio di caso possono essere dei seguenti tipi:

Trillo - Un trillo è un esecuzione molto veloce dell'alternanza di due note ravvicinate che distano fra di loro di un tono o semitono diatonico;
Arpa - Nel contesto delle *refractions* un'arpa rappresenta la melodia generata da uno strumento musicale che produce dei suoni grazie al passaggio del vento attraverso esso;
Bordone - un bordone rappresenta l'esecuzione continua (per la maggior parte di un esecuzione), di una oppure più note;
Canone - Un canone è una composizione in cui è presente l'unione tra una melodia e una o più sue imitazioni, che le si sovrappongono progressivamente.
Ventata - Il termine deriva dall'inglese *swing* e indica un modo di eseguire le note tipico della musica jazz, in cui l'andamento ritmico di esse è ondeggiante;
Accordo - Un accordo è un insieme di note differenti che vengono suonate nello stesso istante;
Accento armonico - Per accento armonico si definisce il mettere in risalto una determinata nota, cambiandone l'armonia.

[7] Letteralmente prova e sbaglia, si tratta di un metodo con cui si cerca di trovare una soluzione a un problema per tentativi fermandosi quando si ottiene il risultato desiderato

[8] I livelli che erano stati progettati nell'algoritmo non sono in relazione biunivoca con i livelli della storia del film "Creature from the Black Lagoon"che il flipper cerca di rappresentare

Il funzionamento del codice (di cui è rappresentato il diagramma di flusso nella figura 2.8) si può riassumere in questo modo: Ogni volta che la pallina colpisce un oggetto viene fornita in output al Disklavier una nuova sequenza, che dipende dal livello dell'algoritmo e dal numero di palline in gioco. In base al livello dell'algoritmo verranno quindi eseguite le seguenti istruzioni:

Livelli 0/1 - Livelli iniziali, corrispondono ai momenti in cui ci sono in gioco rispettivamente zero e una pallina e non è ancora stata raggiunta la condizione di accensione di una delle luci. In questo caso vengono riprodotte delle sequenze preregistrate dal compositore Carlo De Pirro;
Livello 2 - Viene raggiunto quando si accende solo una delle lettere della parola "FILM". Viene generato un trillo oppure una ventata, fino a un massimo totale di 15, ogni volta che viene colpita un oggetto dalla pallina. Superato il limite di 15 vengono generate delle sequenze di accordi finché non varia il livello;
Livello 3 - Viene raggiunto con l'accensione di una seconda luce, consiste nella generazione di un Arpa che viene riprodotta per un tempo compreso tra i 15 e i 20 secondi. In contemporanea vengono anche generati delle ventate e dei canoni a 6 voci. Se alla fine di questo tempo il livello è invariato vengono caricate nel Disklavier delle sequenze prescritte associate a quel determinato livello;
Livello 4 - Viene raggiunto con l'accensione di una terza luce, consiste nella generazione di un Arpa che viene riprodotta per un tempo compreso tra i 25 e i 30 secondi. In contemporanea vengono anche generati dei canoni a 6 voci e per ogni oggetto colpito dalla pallina viene suonata una sequenza di accordi. Anche in questo caso se alla fine di questo tempo il livello è invariato vengono caricate nel Disklavier delle sequenze prescritte associate a quel determinato livello;
Livello 5 - Appena si accendono tutte le luci che compongono la parola "FILM" si accede a questo livello; In questo livello vengono semplicemente caricate e riprodotte dal Disklavier delle sequenze preregistrate associate a questo livello;
Livello 6 - Questo livello viene raggiunto nel momento in cui si aziona la modalità extra ball (EX), vengono riprodotte delle continue sequenze di trilli che vengono centrati tra le note Sol#6 e Sol#7;
Livelli 7/8 - Questi due livelli finali vengono raggiunti quando nel gioco si raggiungono le fasi RE, JK o CS. Vengono riprodotti in continuazione dei trilli il cui centro trasla in base ad ogni target colpito.

Nei trilli presenti nei livelli 3 e 6 e nei bordoni è possibile che sia presente anche la generazione di alcuni accenti armonici. In entrambi i casi ciò accade solo se si verificano alcune condizioni imposte dall'algoritmo.

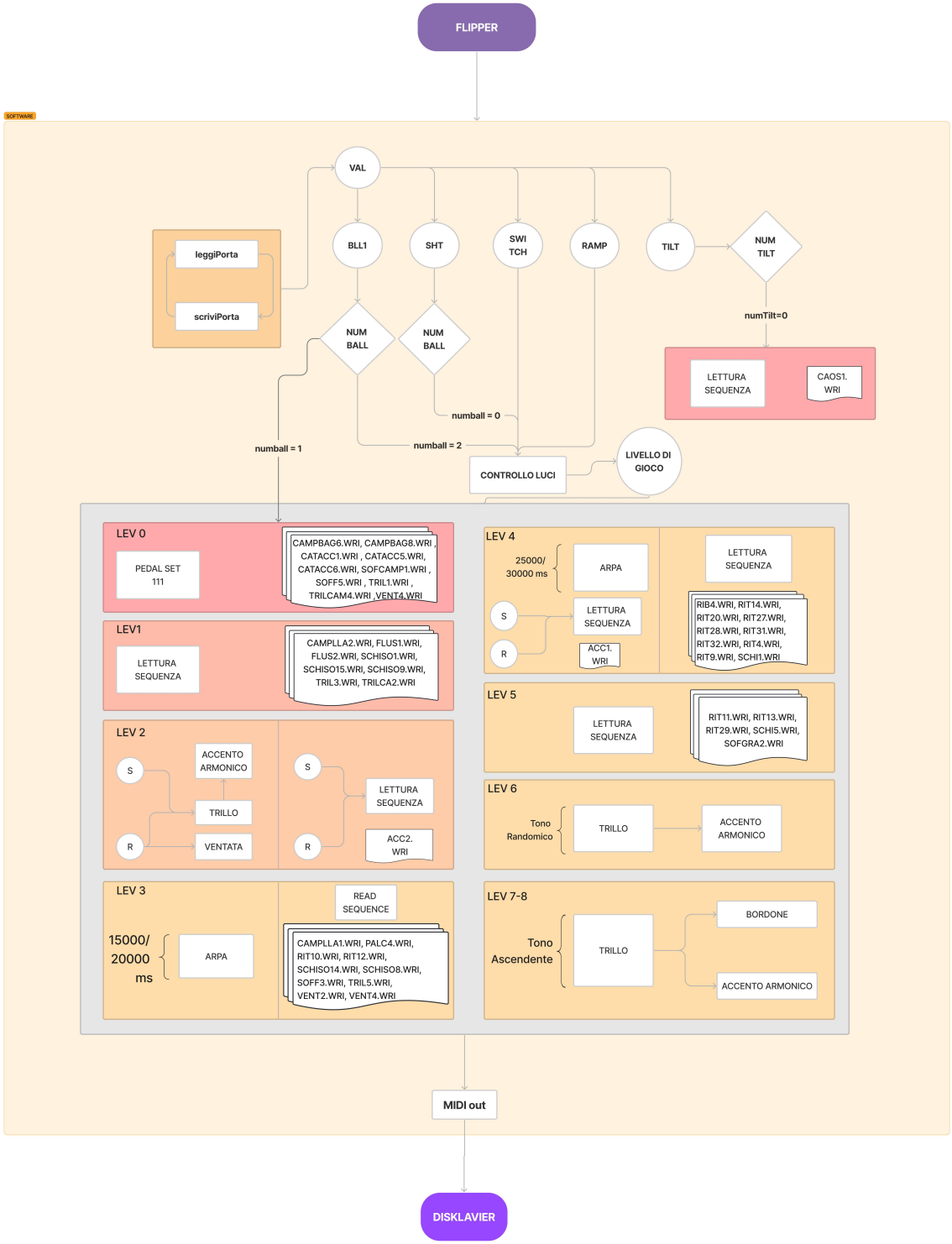


Figura 2.8: *Diagramma di flusso riferito all'algoritmo originale implementato nell'installazione*

2.3 Modello di conservazione applicato allo studio di caso

In questo paragrafo verrà trattata l'applicazione del modello di conservazione multilivello allo studio di caso in esame. Il modello di conservazione utilizzato è un espansione delle linee guida della metodologia di conservazione di documenti audiovisivi definita da (Bressan e Canazza 2013; Bressan 2016; Verde 2018) e presentata nel capitolo precedente. Fondamentale per questo metodo è il concetto della copia conservativa; essa consiste in un documento progettato per essere conservato e mantenuto aggiornato come elemento cardine di tutto il progetto conservativo; esso contiene un insieme organizzato efficacemente di tutti i dati e le informazioni a partire dalla documentazione originale, che viene poi completata dai metadati e dai documenti riguardanti tutto il processo di rigenerazione.
Per presentare il modello conservato dobbiamo inanzitutto introdurre il concetto di DPO (*Digital Preservation Object*). Un DPO riferito a un opera d'arte multimediale interattiva è un file digitale che raccoglie un insieme di elementi digitali ed analogici sempre interconnessi tra loro; essi sono correlati tra loro seguendo un architettura logica che ha il compito di rappresentare una singola esposizione dell'installazione (può essere la prima esibizione al pubblico dell'opera o una determinata riattivazione di essa). Lo scopo finale dello schema multilivello è quello di raccogliere e collegare tra loro tutti i diversi DPO per rappresentare l'opera d'arte più come una serie di processi che come un singolo lavoro immutabile.
La struttura del modello è stata basata sul paradigma ISAD(G)[9], cioè lo standard internazionale di descrizione archivistica, elaborato tra il 1988 e il 1993 dalla *Commissione per gli standard di descrizione* del Consiglio interazionale degli archivi; questo paradigma definisce una terna di livelli, correlati tra loro da una struttura gerarchica con il compito di rappresentare un archivio descrivendolo appunto su più livelli, da uno più generale fino ad arrivare ai sottolivelli più specifici. Nel nostro caso quindi avremo un livello generale, che rappresenta l'opera in se, che al suo interno racchiude in una serie di sottolivelli tutte le varie esibizioni. In analogia alle cartelle dell' ISAD(G) possiamo vedere quindi i DPO come dei raccoglitori per tutti i singoli oggetti che rappresentano l'installazione, classificando come oggetti dell'opera tutti i singoli file (sia analogici che digitali) che la compongono e che quindi descrivono il livello più basso del modello.
Ogni singolo DPO quindi contiene un'insieme di oggetti che, riprendendo le macrocategorie del modello di conservazione originario del CSC, possiamo raggruppare in base al loro ruolo all'interno dell'opera nelle tre classi (la cui struttura è stata descritta nel paragrafo 1.2) bit, data ed experience. Una caratteristica che abbiamo voluto implementare al modello, e che fa si che esso diverga dalla struttura classica dell' ISAD(G), è l'appartenenza multipla degli oggetti ai vari DPO. Infatti, come si può osservare dalla rappresentazione grafica del modello nella figura 2.9, gli oggetti contenuti nei livelli bit e data possono essere contenuti da più DPO. Ad esempio il flipper utilizzato nelle prime esposizioni dell'opera è lo stesso utilizzato nelle ultime e quindi quello apparterrà ad ogni singolo DPO rappresentante l'installazione. Gli unici elementi che sono sempre diversi tra i vari DPO sono quelli contenuti nel livello experience, dato che essi si riferiscono proprio a una singola esposizione e quindi non possono essere comuni tra 2 o più di esse.

[9]General Instruction Standard for Archival Description, https://www.icar.beniculturali.it/index.php?id=54

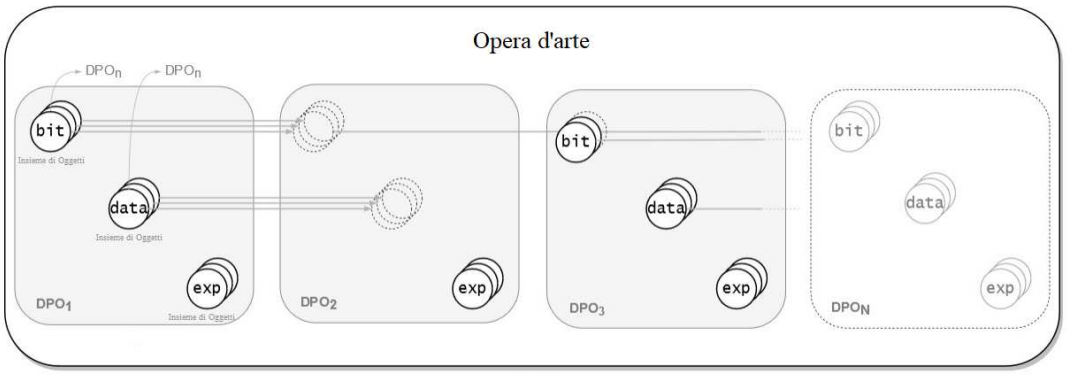


Figura 2.9: *Rappresentazione grafica del modello di conservazione multilivello.*

2.4 Migrazione tecnologica dello studio di caso

A partire dal modello di conservazione discusso nel paragrafo precedente è stata compiuta di recente, nei laboratori del CSC presso il dipartimento d'ingegneria dell'informazione dell'università di Padova, la migrazione tecnologica dello studio di caso *Il caos delle Sfere* ad opera di Alessandro Fiordelmondo, Mattia Pizzato e Luca Zecchinato. Per migrazione tecnologica si intende la riattivazione di un'opera utilizzando le nuove tecnologie, con l'obbiettivo primario di preservare l'identità dell'opera, specialmente nel rapporto interattivo che si viene ad instaurare tra l'uomo e la macchina.
La migrazione tecnologica è stata presa in considerazione dato che, a parte in una esibizione avvenuta nel 2012, l'opera "Il Caos delle Sfere" non è mai stata presentata al pubblico dal 2004 in poi. Questo ha fatto si che il PC non venisse mai aggiornato per quasi vent'anni, periodo in cui la tecnologia ha avuto un enorme evoluzione, tanto da rendere troppo obsoleto e inaffidabile il contenuto del computer utilizzato negli sviluppi originali dell'installazione. Il primo passo del processo di conservazione e di riattivazione è stato quello di prendere in esame tutti i documenti e gli elementi relativi all'opera che permettessero di stabilire un DPO delle prime esposizioni.
Per quanto concerne la parte dei bit, tutto il sistema utilizzato per l'installazione originale è ancora disponibile, e nel corso delle varie esposizioni dell'opera nel corso degli anni le uniche modifiche sono state a livello di codice. Tutti gli oggetti fisici utilizzati (la scheda di acquisizione, il computer e il flipper) sono stati conservati negli anni presso i laboratori del CSC con il Dislavier utilizzato per l'ultima esposizione nel 2012 che è stato utilizzato dal *Conservatorio Pollini* di Padova per scopi didattici. Sono state generate poi delle copie dell'hard disk del computer originale, contenente le sequenze musicali e le varie copie dei software utilizzati.
Purtroppo però non è stato altrettanto semplice raccogliere documentazione riguardante gli ambiti dei data e delle experience, dato che come spiegato precedentemente tutto il codice è stato sviluppato con una metodologia ed è stato conservato per buona parte senza documentazioni. Lo stesso vale per la scheda di acquisizione, di cui non è stato conservato nessuno schema che ne rappresentasse un diagramma elettrico. Per questo l'insieme di oggetti che appartengono alla categoria dei data è completamente incompleto.
Per quanto riguarda l'ultimo insieme, quello delle experience, sono state archiviate e sono disponibili online[10] solo alcune testimonianze fotografiche riguardanti l'aspetto dell'installazione e qualche azione che veniva eseguita durante il suo utilizzo. Purtroppo non è stato possibile trovare alcuna testimonianza audiovisiva. Nell'insieme delle experience sono state inserite anche

[10]Associazione Carlo De Pirro, Il Caos delle Sfere, http://csc.dei.unipd.it/depirro/ opere/composizioni/caos-delle-sfere

i diversi articoli scientifici che sono stati pubblicati per descrivere l'installazione e la conservazione multilivello, come ad esempio [1] oppure [2].
Ogni scelta riguardante il processo di riattivazione è stata presa purtroppo senza il consulto diretto con l'artista che l'aveva concepita, che è sfortunatamente deceduto nel 2008. Nel processo sono stati mantenuti inalterati i nodi interattivo e riproduttivo, rappresentati come esposto precedentemente dal flipper e dal Disklavier. Le modifiche sono state sostanzialmente apportate al nodo comunicativo con il PC che è stato sostituito, dopo svariati test per trovare il dispositivo più adatto, da una scheda Arduino Mega con l'algoritmo e il software che sono stati riprogammati appunto in uno sketch Arduino per adattarli al linguaggio di programmazione proprio della scheda.
Dopo una prima riattivazione, che è stata presentata durante l'esibizione *Science4All*[11] il progetto di conservazione ha subito ulteriori modifiche, coadiuvate anche dall'autore di questa tesi, con il fine di rendere più efficiente e leggibile lo sketch Arduino e che verranno esposte successivamente.

2.4.1 La scheda Arduino Mega

Come nell'opera originale, anche nell'installazione riattivata il nodo cardine è quello comunicativo, che regola l'interazione tra il flipper e il Disklavier, e che come detto poco fa è costituito da una scheda Arduino Mega 2560 che va a sostituire il PC presente nell'installazione originale. Arduino è una piattaforma elettronica sviluppata nel 2005 da alcuni membri dell'*Interaction Design Institute* per il design e la programmazione di microcontrollori. Le caratteristiche fondamentali di una scheda Arduino sono le seguenti:

- È in grado di leggere dati in input attraverso una serie di pin analogici o digitali che permettono all'Arduino eventualmente d'interfacciarsi anche ad altre schede;
- Permette la programmazione di microcontrollori utilizzando un proprio linguaggio di programmazione e un software di sviluppo integrato;
- Permette inoltre di programmare anche attraverso altri software open source, maggiormente utilizzabili per scopi didattici e basati sul linguaggio e sulle librerie C++;
- Anche l'hardware inoltre è open source, gli schematici delle varie schede sono disponibili online così che sviluppatori più esperti possano modificarli, apportandone aggiornamenti necessari ai propri scopi.

Il modello di scheda utilizzato per l'installazione (*Arduino Mega 2560*, figura 2.10) è stato scelto per l'elevato numero d'ingressi e uscite possibili, presenta infatti 54 pin che permettono ingressi ed uscite digitali, un numero abbastanza grande da permetterci il collegamento della porta DB25 presente nella scheda di acquisizione. Un'altra motivazione risiede nel fatto che questo modello di scheda viene usato solitamente per i casi in cui è necessaria una grande potenza di calcolo data la presenza del chip ATmega2560 (che garantisce una frequenza di 16 MHz) e a una dimensione della RAM notevolmente maggiore rispetto a quella presente nelle altre schede Arduino.
Come detto nel paragrafo precedente la riprogrammazione del codice è stata scritta in uno sketch Arduino. Questo ambiente di programmazione si basa sul linguaggio C++ e nel suo codice principale presenta sempre due funzioni: una funzione setup che inizializza e imposta alcune variabili iniziali utilizzate dal programma e una funzione loop che come indica il nome ripete, finchè la scheda non viene disattivata, un ciclo d'istruzioni.

[11] Convegno di divulgazione scientifica che si è tenuto presso l'Università di Padova il 30 settembre 2022

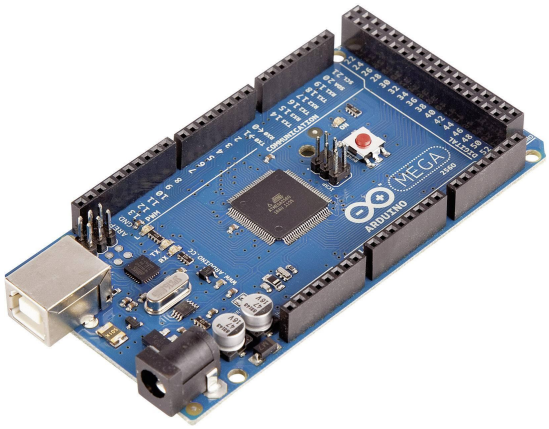


Figura 2.10: *Scheda Arduino Mega 2560 utilizzata per l'installazione.*

A differenza del codice originale, i file *.wri relativi alle sequenze preregistrate non sono contenute all'interno della memoria Arduino (dato lo spazio limitato rispetto al PC originale) ma sono contenute in una scheda SD inserita nella scheda Arduino.

2.4.2 Multithreading: che cos'è e perché è fondamentale per il nostro algoritmo

Come abbiamo esposto precedentemente una delle funzioni per noi fondamentali dell'ambiente MidiShare era quella che consentiva la temporizzazione degli eventi MIDI (che fossero essi preregistrati o generati in tempo reale) da riprodurre, così che noi potessimo pianificare l'inizio della riproduzione delle singole sequenze per far si che non avvenissero pause e che il flusso di suoni prodotto dal Disklavier fosse continuo. Questo però era possibile in quanto attraverso MidiShare la trasmissione di ogni singolo evento veniva compiuta in un *thread* separato. Avevamo quindi in questo caso un thread in cui veniva gestito il cuore pulsante del codice e dove venivano eseguite tutte le funzioni, e un thread per ogni funzione da temporizzare e che andava in un futuro riprodotta dal Disklavier.
Identificando come elemento base nell'esecuzione di un programma il processo, che è una singola azione che può anche venire eseguita in contemporanea ad altri processi, possiamo definire il thread come la rappresentazione di tutte le azioni compiute da una periferica per eseguire questo processo. I thread vengono presi in considerazione quando abbiamo la necessità di eseguire più azioni in contemporanea, il problema principale è che ciò è possibile solamente su dispositivi che presentano più di una unità di elaborazione[12]. Il codice originario operava eseguendo tutta la routine principale del programma in un thread, mentre in uno parallelo si occupava di programmare nel tempo l'esecuzione delle funzioni e la trasmissione degli eventi MIDI al Disklavier.
Tutto ciò nell'installazione originale era possibile in quanto il PC utilizzato presentava un processore *multicore*, cosa non possibile con la nostra scheda Arduino Mega 2560 in quanto il processore incluso in essa presenta un solo core e quindi può eseguire un solo thread alla volta e non più di uno in parallelo come sarebbe necessario. Per far si che il tutto sia eseguibile abbiamo bisogno d'implementare una struttura software in grado di simulare il comportamento di thread multipli.
L'algoritmo implementato nell'installazione è un esempio di programmazione ad eventi, un paradigma in cui le azioni eseguite da esso sono governate da messaggi provenienti da altri pro-

[12]La singola unità di elaborazione viene definita *core*

grammi o da valori provenienti dall'ambiente esterno (forniti dall'utente o recepiti attraverso sensori esterni). Solitamente questo paradigma è utilizzato per programmi che devono essere implementati su sistemi che non hanno una grande disponibilità in termini di memoria, e consiste in un ciclo principale che viene iterato continuamente con il compito di aspettare che accada o venga captato dall'esterno un evento che azioni una funzione specifica. Nel nostro studio di caso le funzioni sono gli invii di eventi MIDI al Disklavier oppure lo svolgersi di altre funzioni specifiche implementate nel codice.
Nell'utilizzo di una programmazione orientata agli eventi si presenta però una problematica, portata dal fatto che questo paradigma non supporta una funzione che permette di mettere in attesa un thread per poi eseguirlo nel momento in cui che quello precedente è terminato. Per questo motivo vengono solitamente utilizzate dagli sviluppatori delle macchine a stati, per progettare come nel nostro caso dei programmi che prevedono un comportamento dinamico dipendente dagli output forniti dagli stati precedenti a quello in esecuzione. Generalmente però questo tipo di approccio rende la programmazione orientata ad eventi molto complessa.
Abbiamo quindi deciso d'implementarne un paradigma che simula il multi-threading attraverso una programmazione simile a quella della macchina a stati finiti. Si basa sul fatto che molte volte all'interno di un singolo processo tra due azioni successive ci sia il bisogno di aspettare un certo intervallo temporale, durante il quale il programma risulta bloccato, per questo ci può essere utile separare il processo in singole funzioni da richiamare poi nel loop principale quando necessario. Questa è una tecnica utilizzabile però nei casi in cui le varie funzioni non sono troppo complesse, ma che però si adatta alla perfezione al nostro caso in cui dobbiamo leggere e scrivere in array, generare numeri casuali o trasmettere eventi MIDI, tutti compiti in cui non è necessaria una grande computazione.
Lo scopo principale dell'implementazione è il far si che tra un azione e l'altra il programma non debba restare in attesa, ad esempio se sto aspettando la ricezione di un messaggio attraverso la porta seriale, devo continuare comunque in parallelo l'esecuzione del programma. Si potrebbe infatti banalmente aspettare la completa ricezione del messaggio per poi continuare l'esecuzione ma ciò non sarebbe efficiente e non rispecchierebbe per nulla il concetto di multithreading. Dobbiamo quindi trovare un modo per cui il programma continui a generare e ad inviare al Disklavier sequenze di eventi MIDI in contemporanea all'esecuzione del loop principale dell'algoritmo. Ciò è stato implementato grazie alla presenza di molte variabili di stato, che in ogni momento sono in grado di definire se una determinata sequenza è oppure no in esecuzione e grazie a delle funzioni in grado di tenere traccia del tempo passato dall'inizio dell'esecuzione del programma e tra una funzione e l'altra in grado di azionare altre funzioni descritte all'interno del codice nel momento opportuno.
Questa seconda caratteristica è implementabile grazie all'uso della funzione *millis()*, che è in grado di fornire il numero di millisecondi trascorsi dal momento in cui la scheda Arduino ha iniziato l'esecuzione del programma e che viene richiamata ogni volta che il loop dello sketch Arduino viene eseguito. Abbiamo poi implementato alcune variabili che utilizzano questa funzione per tenere traccia della differenza di tempo tra l'istante in cui ci troviamo e l'ultimo momento in cui è iniziata l'esecuzione di una funzione. Se il tempo trascorso è maggiore o uguale del tempo necessario per il completamento di quella determinata funzione essa può essere completata, in caso contrario viene impedita l'esecuzione di qualsiasi altra azione prima del completamento di quella precedente. Questa differenza di tempo viene aggiornata ad ogni loop e successivamente viene ripetuto il controllo, quando esso viene soddisfatto viene poi azzerata in contemporanea all'inizio dell'esecuzione di una nuova funzione. E bene ricordare che per nel caso in cui una funzione non avesse finito il suo svolgimento non possiamo eseguire la funzione *delay()*, che fermerebbe l'esecuzione del programma per un intervallo tempo indicato e vanificherebbe i nostri tentativi di simulare un approccio multi-tasking.
Questa implementazione fa si che nel nostro studio di caso che non comprende funzioni dalla

una lunga durata, dato che esse sono tutte molto semplici dal punto di vista computazionale, il multithreading venga simulato efficacemente (a parte per qualche piccolo caso in cui si verificano dei ritardi, ma che è trascurabile dato che non modifica l'esito finale della computazione).

2.4.3 Applicazione della programmazione ad oggetti allo studio di caso

Per l'esibizione *Science4All* è stato quindi sviluppato un codice che seguisse le linee guida della programmazione ad eventi e che utilizzasse, come esposto nel paragrafo precedente, la funzione *millis()* per gestire la temporizzazione mantenendo inalterate tutte le considerazioni riguardanti l'accesso ai dati trasmessi dal flipper attraverso la porta parallela e la divisione del codice nei livelli algoritmici esposti precedentemente. Successivamente a questa esposizione è stato quindi deciso di proseguire la riattivazione dell'opera riscrivendo il codice mediante una programmazione orientata ad oggetti, dato che lo Sketch Arduino presentava una composizione molto confusionaria e complessa, che potrebbe renderne difficile la comprensione per chi volesse in un futuro riutilizzare l'installazione per nuovi sviluppi o per nuove esposizioni.
La programmazione orientata agli oggetti è un paradigma di programmazione che permette di rappresentare il programma non come un unico flusso d'istruzioni ma come un insieme di oggetti che interagiscono tra loro scambiandosi informazioni con lo scopo di raggiungere i risultati richiesti. Gli oggetti, seguendo la loro definizione fornita da Booch[13], sono degli elementi che possiedono tre caratteristiche:

Uno stato - Il loro stato è rappresentato dell'insieme dei loro attributi e delle loro proprietà, come ad esempio l'elenco di tutte le variabili che si riferiscono a quel determinato oggetto.

Un comportamento - E l'insieme delle funzioni che ne denotano le sue azioni all'interno dell'ambiente in cui è contenuto.

Un identità - E il loro nome univoco che fa si che si differenzino tra loro e che non vengano utilizzati al posto di altri nell'utilizzo del programma.

Gli oggetti hanno quindi la funzione di definire le entità che governano l'algoritmo e incorporano tutte le informazioni e i comportamenti che esse devono assumere [3].
E necessario poi introdurre il concetto di classe. Mentre un oggetto è come detto un'entità con una propria identità univoca una classe è lo schema a partire da cui i singoli oggetti vengono creati. A partire da una classe possono essere creati un numero infinito di oggetti, tutti con la stessa struttura, che possono presentare eventualmente un diverso contenuto degli attributi a essi riferiti.

La programmazione ad oggetti ci permette come detto di migliorare di molto la leggibilità del codice, andando a creare una classe per ogni tipo di sequenza musicale fornito in output dalla scheda Arduino e che poi verrà riprodotta dal Disklavier.
Tutto ciò è possibile nell'ambiente fornito da Arduino dato che gli sketch seguono di dettami del linguaggio di programmazione C++ che permette di creare un file *.cpp e uno *.h per ciascuna delle classi, i cui oggetti creati a partire da essa saranno poi utilizzati nel loop dell'algoritmo. Sono stati poi creati degli oggetti che gestiscono altre funzioni cardine all'interno dell'algoritmo, cioè per la gestione del tempo e l'invio in output degli eventi MIDI. Le classi predisposte per la riattivazione dell'algoritmo (di cui qua sotto vengono mostrate le righe di codice dei file *.h[14] e *.cpp[15] riferiti agli oggetti Time, Midi e Trillo) sono quindi:

[13]Grady Booch è un informatico statunitense che ha contribuito allo sviluppo dell'ULM (*Unified Modeling Language*) e per aver definito molti degli attuali principi della programmazione ad oggetti

[14]File di tipo header, è un file di testo in cui sono contenuti i prototipi delle funzioni e delle variabili riferite a quell'ambiente

[15]File di codice sorgente scritto in C++ che serve a definire i vari passaggi delle funzioni i cui prototipi sono stati forniti nei file *.h

Trillo - Questa classe presenta al suo interno la funzione *set* che ha il compito d'inizializzare le variabili proprie della classe e la funzione *play*, che viene richiamata dal loop nel momento in cui un trillo dev'essere riprodotto;

```
#include "Arduino.h"
#include "Trillo.h"
#include "Variabili.h"
#include "AccArm.h"

Trillo::Trillo(){
}

/* Inizializza le variabili dell'oggetto trillo in base al suo tipo
   Input:
   tipo = Tipo di trillo da settare
   Output: void
 */
void Trillo::set(int tipo){
  notaTri=72+random(32);
  tipoTrillo=tipo;
  tempoTri=75;
  varTri=-2+random(6);
  maxTri=3+random(4);
  actTri=0;
  fTri=0;
  actTri = 0;
}

/* Riproduzione di un evento trillo
   Input: void
   Output: void
 */
void Trillo::play(){
  ti.deltT = ti.t - t0t;
  if (actTri == 0)
    yNote = notaTri;      //Inizializza myNoteOn

  /*-------------------------------------*/
  // NOTE OFF TRILLO

  if (velTri > 0 && ti.deltT >= tempoTri-20){
    velTri = 0;
    mid.outMIDI(noteON, myNote, velTri);
    h_acc.play(myNote, velTri);
    ++actTri;
    ++actTri2;

    //cambio valori seconda nota
```

```
45
46    if(actTri%2==0){
47      myNote=notaTri;
48    } else {
49        if(actTri<maxTri){
50          myNote=notaTri+varTri;
51        } else {
52          actTri=0;
53          maxTri=3+random(4);
54          varTri=-2+random(6);
55          myNote=notaTri+varTri;
56          }
57      }
58
59    //BORDONE per Trillo
60
61    if(tipoTrillo){
62      if(fTri==1 || fTri==3){
63        bord.play1(myNote, velTri);
64      }
65      if(fTri>1){
66        bord.play2(myNote, velTri);
67      }
68      if(notaTri%20==0 && random(2)==0){
69        notaTri=notaTri-(8+random(13));
70        fTri=(++fTri)%4;
71      }
72    }
73  }
74
75  /*-------------------------------------*/
76  // NOTE ON TRILLO
77
78  if (velTri == 0 && ti.deltT >= tempoTri){
79    velTri = 80+random(30);
80    mid.outMIDI(noteON, myNote, velTri);
81    h_acc.play(myNote, velTri);
82    if(tipoTrillo){
83      if(fTri==1 || fTri==3){
84        bord.play1(myNote, velTri);
85      }
86      if(fTri>1){
87        bord.play2(myNote, velTri);
88      }
89    }
90    t0t = ti.t;
91  }
92
93  /*-------------------------------------*/
```

```
  // BARICENTRO E VENTATE

  if(SWITCH){
    if(tipoTrillo){
      notaTri+=1+random(2);
    } else {
        notaTri=80+random(25);
      }
  }
  if (RAMP && venON2==0){
    vent.venPit=myNote;
    venON2=1;
    if(random(4)==0){
      vent.pit=21+random(7);
    } else {
      vent.pit=55+random(7);
    }
    vent.t0v = ti.t+90;
  }
}
```

```
#ifndef Trillo_h
#define Trillo_h

#include "Arduino.h"

class Trillo {
  public:
    Trillo();
    void set(int tipo);
    void play();
    char numTotTri;         // Numero totale di trilli (solo per il livello
    2)    di tipo char in quanto durante la partita pu  assumere valori
   negativi (non considerati dall'algoritmo)
    uint8_t numTri2;        // E' di tipo uint8_t perch  assume sempre
   valori positivi
    uint8_t actTri2;        // Tnnesimo trillo (numTri2)
    int t0t;                // Tempo in cui    stato eseguito l'ultimo
   evento MIDI riguardante un trillo
  private:
    uint8_t myNote;         // Nota del trillo
    uint8_t maxTri;         // Massimo numero di trilli (da 3 a 7)
    uint8_t actTri;         // Ennesimo trillo
    char varTri;            // Variazione delle note, di tipo char perch
   p u  assumere anche valori negativi
    uint8_t velTri;         // Velocit  del trillo
    uint8_t notaTri;        // Nota di base del trillo
    uint8_t tempoTri;       // Tempo del trillo, di tipo uint8_t in quanto
      una costante fissa a 75
```

```
23     uint8_t tipoTrillo;      // Aggiunta di bordoni metto uint8_t perch
       vale 0 o 1
24     uint8_t fTri;            // Contatore per scegliere un bordone random,
       metto uint8_t perch  assume valori maggiori di 0 e viene sempre ridotto
        modulo 4
25 };
26
27  #endif
```

Arpa - Questa classe presenta al suo interno quattro funzioni: la funzione *ini* che ha il compito d'inizializzare le variabili proprie della classe, la funzione *set* che ha il compito di settare le variabili riferite al singolo oggetto arpa, la funzione *update* che aggiorna le caratteristiche dell'accento armonico basato sulla tipologia di arpa in riproduzione la funzione *play*, che viene richiamata dal loop nel momento in cui un'arpa dev'essere riprodotto;
Bordone - Questa classe presenta al suo interno quattro funzioni, esse si assomigliano a due a due nella struttura in quanto è possibile la riproduzione di due tipi di bordone. Le funzioni *set1* e *play1* gestiscono l'inizializzazione e la riproduzione del primo tipo di bordone mentre le funzioni *set2* e *play2* gestiscono le stesse operazioni riferite però alla seconda tipologia di bordone;
Canone - Questa classe presenta al suo interno la sola funzione *play* che viene richiamata dal loop nel momento in cui un canone dev'essere riprodotto;
Ventata - presenta al suo interno la sola funzione *set* che viene richiamata dal loop nel momento in cui una ventata dev'essere riprodotta;
Accento armonico - Questa classe presenta al suo interno quattro funzioni: la funzione *ini* che ha il compito d'inizializzare le variabili proprie della classe, le funzione *set* e *update* che hanno il compito di settare le variabili riferite al singolo oggetto arpa in base a un intervallo in cui esso deve essere generato e la funzione *play*, che viene richiamata dal loop nel momento in cui un'arpa dev'essere riprodotto;
Midi - Questa classe presenta la funzione *outMIDI* che una volta ricevute in input le informazioni riguardanti il tipo di azione da eseguire (cmd), la nota su cui compierla e la velocità con cui eseguirla invia al Disklavier l'evento MIDI da riprodurre;

```
1  #include "Arduino.h"
2  #include "Midi.h"
3  #include "Variabili.h"
4
5  Midi::Midi(){}
6
7  /* Invio di un messaggio MIDI attraverso la porta seriale
8    Input:
9    cmd  =   MIDI Status Byte
10   n    =   MIDI Data byte 1
11   vel  =   MIDI Data byte 2
12   Output: void
13    */
14 void Midi::outMIDI(int cmd, int n, int vel){
15   if(cmd == noteON){
16   vel = (int) vel/1.6;
17   }
18   Serial.write(cmd);              // invia il comando note on, note off o
      control change  (status byte)
```

```
19   Serial.write(n);                // invio tono della nota da suonare (Data
       byte 1)
20   Serial.write(vel);              // invio velocit  della nota (Data byte
       2)
21 }
```

```
1  #ifndef Midi_h
2  #define Midi_h
3
4  #include "Arduino.h"
5
6  class Midi {
7    public:
8      Midi();
9      void outMIDI(int cmd, int n, int vel);
10 };
11
12 #endif
```

Time - Questa classe contiene le variabili descritte nel paragrafo 2.4.2 e necessarie per governare la temporizzazione dell'esecuzione degli eventi MIDI. Presenta un costruttore in cui all'inizio dell'esecuzione del programma (cioè nel momento in cui viene creato l'oggetto di tipo Time) vengono inizializzate le variabili t0, tp e tluci la cui descrizione può essere letta nel codice sottostante. La funzione *updt* ha invece il compito di aggiornare il valore di t ad ogni iterazione del loop.

```
1  #ifndef Time_h
2  #define Time_h
3
4  #include "Arduino.h"
5
6  class Time {
7    public:
8      Time();
9      bool sqnz(int a_time);
10     void updt();
11     unsigned long t;              // Tempo trascorso dall'inizio del loop
       espresso in millisecondi
12     unsigned long t0;             // Tempo trascorso dall'inizio del loop
       all'ultimo evento MIDI espresso in millisecondi
13     unsigned long tp;             // Tempo trascorso dall'inizio del loop
       all'ultimo controllo dei valori inviati dalla porta parallela
14     unsigned long tluci;          // Tempo trascorso dall'inizio del loop
       all'ultimo controllo del valore delle luci
15     int deltT;                    // Tempo che separa due eventi successivi
        espresso in millisecondi
16   private:
17 };
18
19  #endif
```

```
#include "Arduino.h"
#include "Variabili.h"

Time::Time(){
  t = millis();
  t0 = t;
  tp = t;
  tluci = t;
}

/* Aggiorna il valore della variabile t all'inizio di ogni loop
   Input: void
   Output: void
 */
void Time::updt(){
  t = millis();
}
```

Per ciascuna classe è stato creato un singolo oggetto che permette, nel caso degli oggetti riferiti alle sequenze musicali, attraverso le sue variabili proprie di essere reinizializzato ogni volta in cui abbiamo la necessità di riprodurre quel determinato tipo di sequenza MIDI (attraverso la funzione set). All'interno della funzione *play* riferita ad ogni sequenza è presente un richiamo alla funzione outMIDI propria dell'oggetto Midi che permette d'inviare l'evento MIDI al Disklavier.
In aggiunta sono stati creati anche due file Variabili.cpp e Variabili.h con il compito di contenere tutte le variabili riferite allo svolgimento del gioco e che devono essere utilizzate da più di un oggetto. La struttura del codice è osservabile nel diagramma di flusso raffigurato nella figura 2.11.

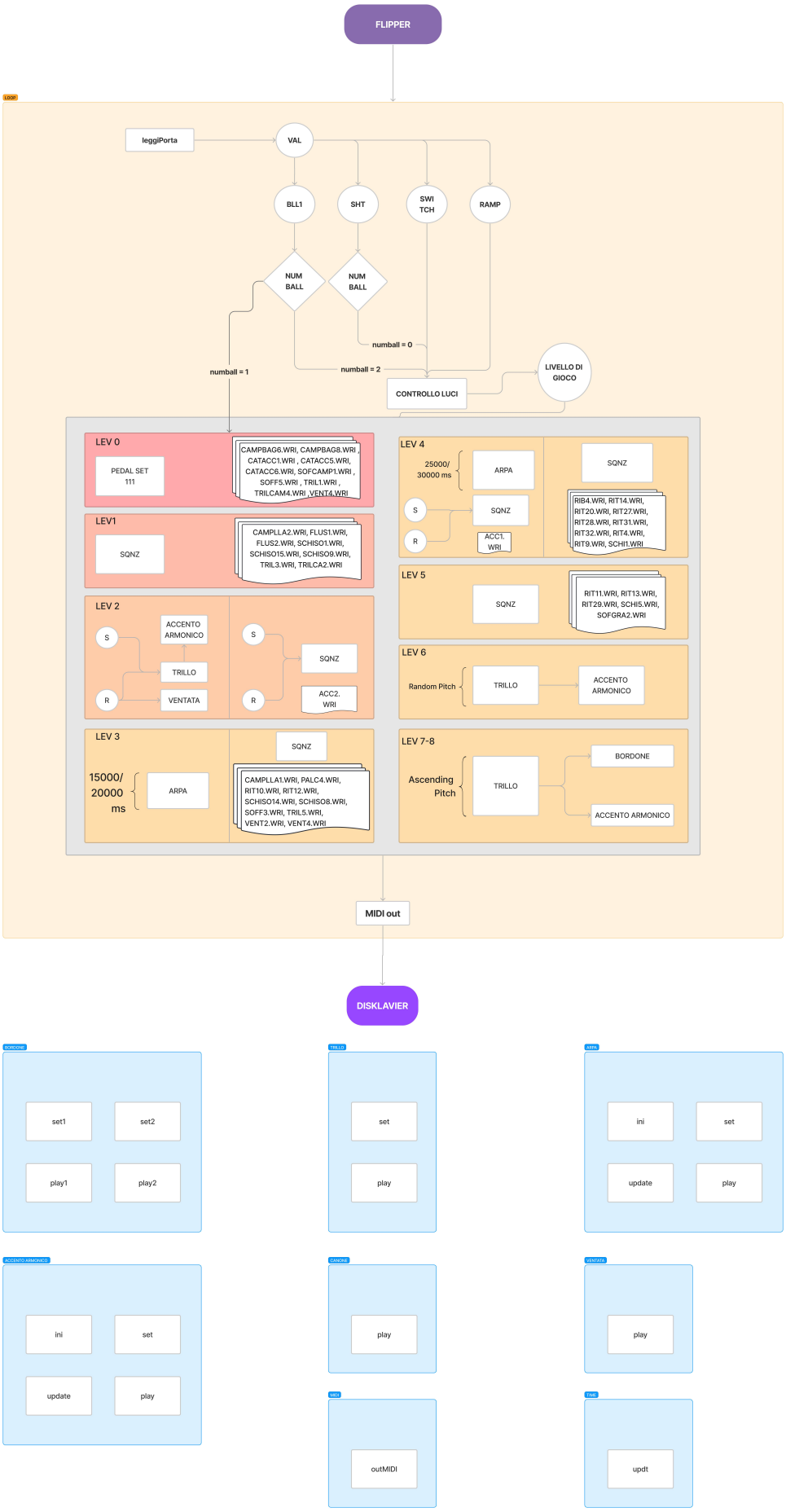


Figura 2.11: *Diagramma di flusso dell'algoritmo implementato con la programmazione ad oggetti.*

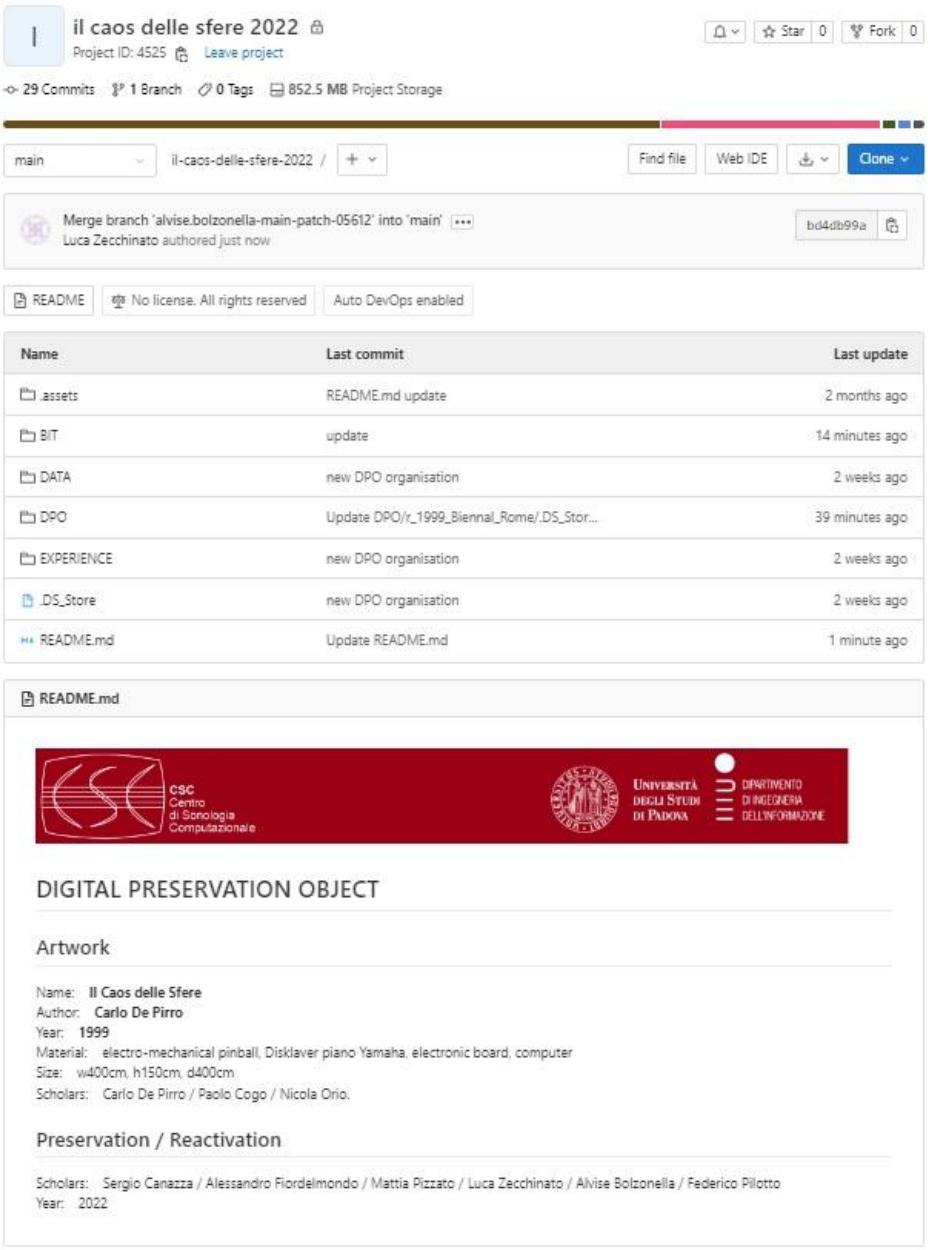


Figura 2.12: *Composizione della repository contenuta nell'ambiente GitLab.*

2.5 Compilazione DPO (Digital Preservation Object) REPO

Per quanto riguarda l'ambito conservativo abbiamo optato per un'implementazione di una repository gestita attraverso la piattaforma web GitLab, un progetto open-source nato nel 2011 con lo scopo di permettere una condivisione di progetti tra sviluppatori, e che ci consente di conservare tutti i documenti riguardanti l'hardware e software dell'opera riattivata in un ambiente online disponibile a chiunque voglia in un futuro osservare questi documenti per ulteriori sviluppi. La repository, la cui composizione è visualizzabile nella figura 2.12 è stata organizzata nel seguente modo: sono presenti al suo interno quattro cartelle BIT, DATA, EXPERIENCE E DPO al cui interno sono presenti una serie di metadati[16] e di documenti che rappresentano il software e l'hardware dell'opera che vengono suddivisi come descritto nel paragrafo 1.2.

[16]Sistema di dati con lo scopo di descrivere altri dati, viene utilizzato per digitalizzare le informazioni in Internet o negli archivi elettronici.

ELEMENTO	DESCRIZIONE
TITOLO	Nome dato alla risorsa
CREATORE	L'ente primario responsabile della creazione della risorsa
SOGGETTO	Argomento della risorsa
DESCRIZIONE	Riassunto dei contenuti della risorsa
EDITORE	Ente responsabile della condivisione della risorsa
COLLABORATORI	Enti che hanno contribuito alla creazione della risorsa
DATA	La data in cui la risorsa è stata pubblicata o altre date importanti riguardanti essa
TIPO	La natura del contenuto della risorsa
FORMATO	Il formato fisico o digitale della risorsa
IDENTIFICATORE	Un riferimento non ambiguo
FONTE	Riferimenti ad altre risorse da cui quella descritta deriva
LINGUA	Il linguaggio del contenuto della risorsa
RELAZIONE	Riferimenti ad una risorsa correlata
COPERTURA	L'obiettivo del contenuto della risorsa
DIRITTI	Informazioni sul detentore dei diritti della risorsa

Figura 2.13: *Modello di base del Dublin Core*

All'interno della cartella DPO invece sono presenti una serie di sottocartelle, ciascuna rappresentante un singolo DPO e quindi una singola esibizione, nel nostro caso quindi avremo cinque cartelle rappresentanti le cinque esibizioni dell'installazione di cui siamo a conoscenza (Rappresentate con i loro DPO nella figura 2.14).
All'interno di ogni cartella riguardante un singolo DPO è contenuto un metadato descrittivo che prende il nome di Dublin Core, in quanto il suo sviluppo è iniziato presso il OCLC (Online Computer Library Center) di Dublin, in Ohio [7]. Questo tipo di metadato è costituito da un nucleo contenente tutti gli elementi essenziali per la descrizione di qualsiasi materiale digitale (video, immagini, pagine Web, ecc.), risorse fisiche o oggetti come ad esempio opere d'arte, via rete informatica. Questo nucleo, nel suo modello di base costituito nel 1996 era composto da 15 elementi cardine rappresentati dalla tabella illustrata nella figura 2.13. In aggiunta a questi elementi ogni metadato presenta anche dei link che rimandano ai metadati dei componenti delle cartelle BIT, DATA ed EXPERIENCE che sono stati utilizzati in una specifica esposizione a cui si riferisce il singolo DPO.

Gli elementi che sono stati collegati ai cinque diversi DPO sono descritti dalla figura 2.14. Nelle prime 4 esposizioni in esame non abbiamo nessun elemento di tipo Data in quanto, come descritto precedentemente nel paragrafo 2.2, non abbiamo la documentazione necessaria per definirli; per quanto riguarda invece l'ambito dei bit le uniche variazioni in queste esposizioni riguardano il codice sorgente, dato che ne sono state utilizzate tre versioni differenti, e i file *.wri che a partire dalla seconda esibizione del 1999 tenutasi a Zagarolo sono stati variati per poi rimanere inalterati nelle due installazioni successive. Per quanto riguarda invece l'installazione riattivata da noi per il convegno *Science4All* sono stati mantenuti inalterati come detto precedentemente i bit riguardanti il flipper, la scheda di acquisizione e il Disklavier modificando tutti quelli relativi al nodo comunicativo dato il passaggio dal pc alla scheda Arduino.

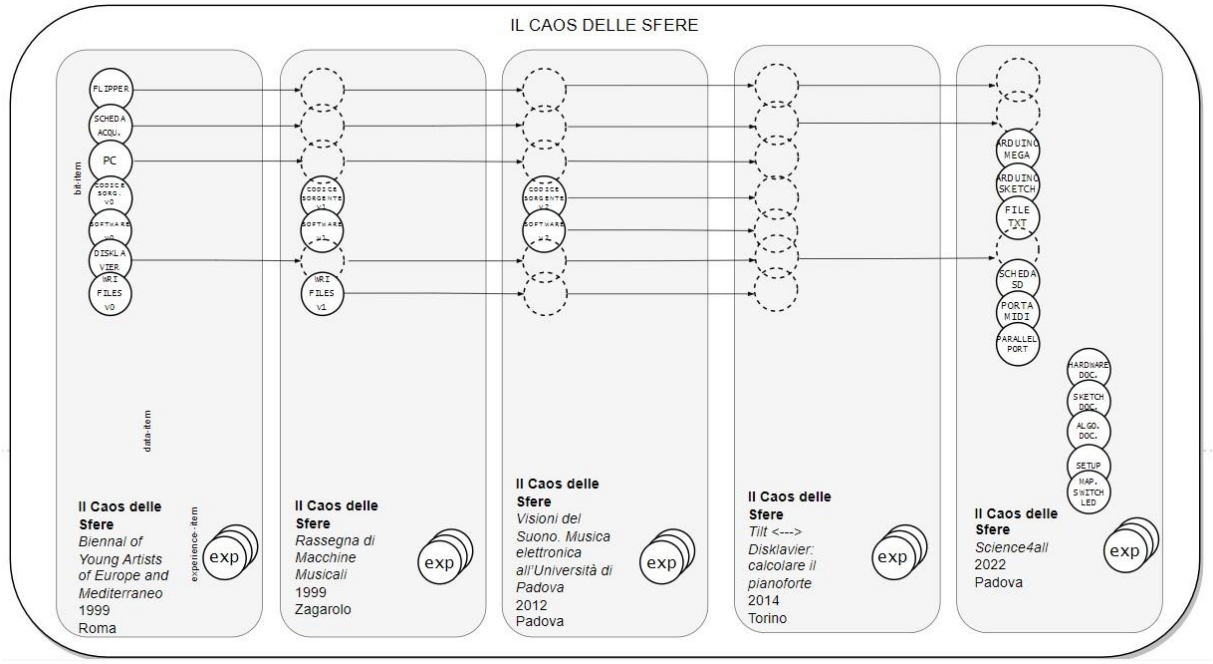
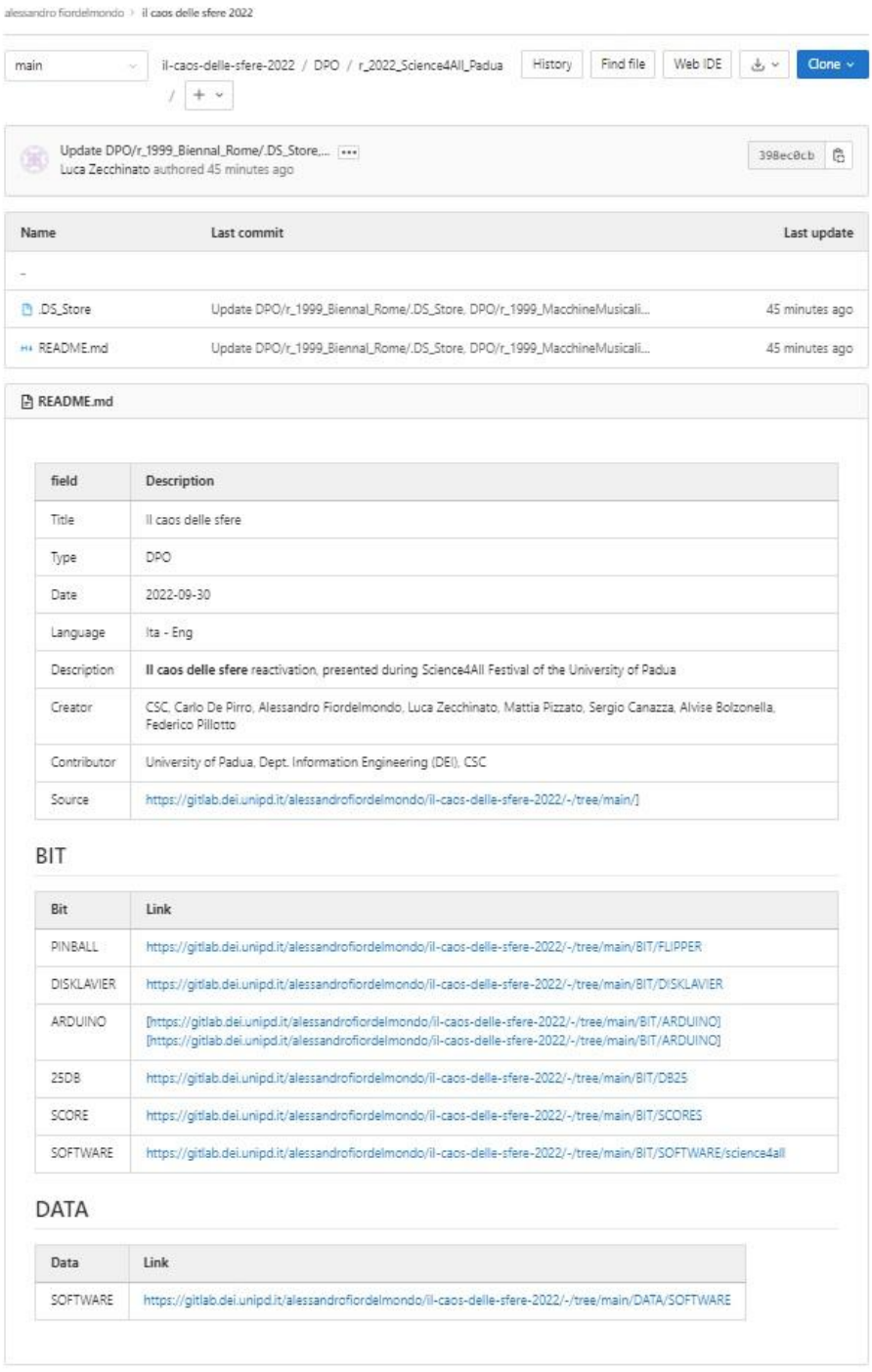


Figura 2.14: *Descrizione del contenuto dei singoli DPO riguardanti l'opera "Il Caos delle Sfere"*

Nelle due figure seguenti sono rappresentati il metadato riferito all'installazione del 2022 presso il convegno *Science4All* (figura 2.15) ed il metadato che riporta gli elementi essenziali per la descrizione del flipper (figura 2.16); si possono notare i metadati sotto forma di tabella che riprendono lo schematico del Dublin Core di cui però non sono stati descritti tutti e 15 gli elementi dato che alcuni erano trascurabili per il caso in esame.



alessandro fiordelmondo › il caos delle sfere 2022

main | il-caos-delle-sfere-2022 / DPO / r_2022_Science4All_Padua | History | Find file | Web IDE | Clone

Update DPO/r_1999_Biennal_Rome/.DS_Store,... — 398ec0cb
Luca Zecchinato authored 45 minutes ago

Name	Last commit	Last update
-		
.DS_Store	Update DPO/r_1999_Biennal_Rome/.DS_Store, DPO/r_1999_MacchineMusicali...	45 minutes ago
README.md	Update DPO/r_1999_Biennal_Rome/.DS_Store, DPO/r_1999_MacchineMusicali...	45 minutes ago

README.md

field	Description
Title	Il caos delle sfere
Type	DPO
Date	2022-09-30
Language	Ita - Eng
Description	**Il caos delle sfere** reactivation, presented during Science4All Festival of the University of Padua
Creator	CSC, Carlo De Pirro, Alessandro Fiordelmondo, Luca Zecchinato, Mattia Pizzato, Sergio Canazza, Alvise Bolzonella, Federico Pillotto
Contributor	University of Padua, Dept. Information Engineering (DEI), CSC
Source	https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/]

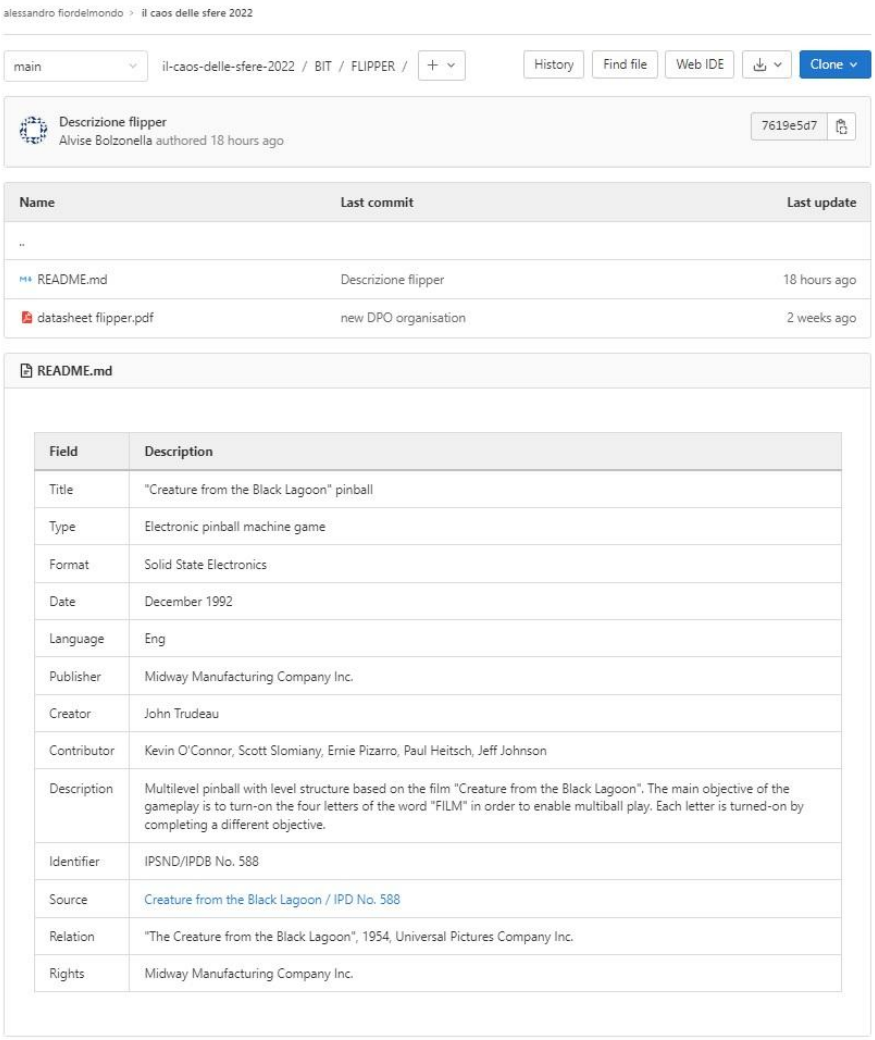
BIT

Bit	Link
PINBALL	https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/BIT/FLIPPER
DISKLAVIER	https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/BIT/DISKLAVIER
ARDUINO	[https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/BIT/ARDUINO] [https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/BIT/ARDUINO]
25DB	https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/BIT/DB25
SCORE	https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/BIT/SCORES
SOFTWARE	https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/BIT/SOFTWARE/science4all

DATA

Data	Link
SOFTWARE	https://gitlab.dei.unipd.it/alessandrofiordelmondo/il-caos-delle-sfere-2022/-/tree/main/DATA/SOFTWARE

Figura 2.15: *Metadato riferito al convegno Science4All tenutosi a Padova il 30-09-2022*



alessandro fiordelmondo > il caos delle sfere 2022

main | il-caos-delle-sfere-2022 / BIT / FLIPPER / | History | Find file | Web IDE | Clone

Descrizione flipper
Alvise Bolzonella authored 18 hours ago — 7619e5d7

Name	Last commit	Last update
..		
README.md	Descrizione flipper	18 hours ago
datasheet flipper.pdf	new DPO organisation	2 weeks ago

README.md

Field	Description
Title	"Creature from the Black Lagoon" pinball
Type	Electronic pinball machine game
Format	Solid State Electronics
Date	December 1992
Language	Eng
Publisher	Midway Manufacturing Company Inc.
Creator	John Trudeau
Contributor	Kevin O'Connor, Scott Slomiany, Ernie Pizarro, Paul Heitsch, Jeff Johnson
Description	Multilevel pinball with level structure based on the film "Creature from the Black Lagoon". The main objective of the gameplay is to turn-on the four letters of the word "FILM" in order to enable multiball play. Each letter is turned-on by completing a different objective.
Identifier	IPSND/IPDB No. 588
Source	Creature from the Black Lagoon / IPD No. 588
Relation	"The Creature from the Black Lagoon", 1954, Universal Pictures Company Inc.
Rights	Midway Manufacturing Company Inc.

Figura 2.16: *Cartella della repository contentente i metadati riguardanti il flipper.*

Capitolo 3

Conclusioni

Questo studio ha cercato d'introdurre ai principi della conservazione e della riattivazione di opere d'arte interattive con tecnologie, fornendone un esempio pratico di applicazione. Riattivazione che si è concentrata sull'ambito algoritmico dell'opera, riuscendo a migliorarne sensibilmente la leggibilità attraverso il paradigma della programmazione a oggetti, con la suddivisione delle varie funzioni del codice originario in base al loro ambiente di utilizzo. Sviluppi futuri potranno cercare di migliorare l'occupazione di memoria del programma, dato che al momento non è ottimizzata al meglio, cercando di partizionare al meglio la memoria assegnata per le funzioni e per le variabili globali all'interno della scheda Arduino.
Per quanto riguarda la conservazione dell'opera è invece stato fornito un modello che è implementabile anche per altre installazioni, data la sua struttura molto semplice e molto chiara, e di facile reperibilità dato il suo contenuto in metadati conservati interamente in un repository online.
In futuro inoltre si potrebbe ripartire dal nostro lavoro per sviluppare una nuova scheda di acquisizione, in grado di elaborare attraverso un microprocessore contenuto al suo interno le informazioni ricevute dal flipper, e che implementi in uscita una porta MIDI, così da non rendere necessario l'utilizzo di periferiche esterne dato che il lavoro verrà gestito interamente dai nodi interattivo e riproduttivo.

Bibliografia

[1] F. Bressan and S. Canazza. The challenge of preserving interactive sound art: a multilevel approach. *International Journal of Arts and Technology*, 7(4):294–315, 2014.

[2] F. Bressan, S. Canazza, A. Rodà, and N. Orio. Preserving today for tomorrow: A case study of an archive of interactive music installations. *Proceedings of WEMIS-Workshop on Exploring Musical Information Spaces*, 2009.

[3] P. Camagni and R. Nikolassy. *Corso di Java, Dalla programmazione ad oggetti alle applicazioni grafiche*. Hoepli, 2013.

[4] MIDWAY MANUFACTURING COMPAMY. Creature from the black lagoon. 1993.

[5] R. Edmonson. Memory of the world: general guidelines to safeguard documentary heritage. *UNESCO*, 2002.

[6] GRAME. Midishare realtime music operating system. Last accessed 04 November 2022.

[7] D. Haynes. *Metadata for Information Management and Retrieval. Understanding metadata and its use*. Facet, 2018.

[8] N. Orio and C. De Pirro. Controlled refractions: A two-levels coding of musical gestures for interactive live performances. *Proceedings of the International Computer Music Conference.*, Ann Arbor, MI: ICMC:88–91, 1998.

[9] Y. Orlarey and Lequay H. Midishare : a real time multi-tasks software module for midi applications. *International Computer Music Conference, Colombus, United States.*, pages 234–237, 1989.

[10] C. Peacock. Interfacing the standard parallel port. 1998.

[11] C. Post. Preservation practices of new media artists: Challenges, strategies, and attitudes in the personal management of artworks. *Journal of Documentation*, 73(4):716–732, 2017.

[12] IEEE Std. Ieee standard signaling method for a bidirectional parallel peripheral interface for personal computersm. 1284-1994, 1994.

[13] P. Waelder. Keep it alive or let it die: new media art, curating and the art market. *ETC Media*, 109, 2016.

[14] G. Wijers. Ethics and practices of media art conservation, a work-in-progress. 2010.