UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea Triennale in Informatica Musicale (f3x)

Sonificazione di Dati Meteorologici: Studio e Prototipazione Software

Relatore: Luca Andrea LUDOVICO

Correlatore: Giorgio PRESTI

Tesi di:

Andrea SCAPOLI

Matricola: 776991

A. A. 2013-2014

Indice

Elenco delle figure

Introduzione

La Sonificazione è una disciplina sviluppatasi da pochi decenni che utilizza segnali audio non vocali per rappresentare informazione. Il primo esempio di sonificazione risale agli inizi del Novecento con l'invenzione del contatore Geiger [1]; questo dispositivo permetteva infatti, attraverso brevi eventi sonori ripetuti, di rappresentare il livello di radiazioni nell'ambiente circostante l'utilizzatore.

Negli anni successivi si è posta sempre più attenzione allo sviluppo di tecniche di Sonificazione. In particolar modo, specialmente in ambito informatico, ha avuto un notevole sviluppo l'utilizzo della Sonificazione applicata a set di dati.

Lo sviluppo di questo elaborato riguarda l'utilizzo della Sonificazione applicata a dati meteorologici. Il lavoro svolto per lo studio e lo sviluppo di tale progetto non è quindi collocabile in una specifica area di ricerca, ma ricade nell'ambito multidisciplinare poiché rimanda a concetti presenti in meteorologia, informatica, elettronica e psicoacustica.

In letteratura [1] [2], è già stato ampiamente dimostrato il motivo per cui la Sonificazione risulta essere una tecnica utile nell'analisi di dati, questa infatti permette di rilevare caratteristiche e relazioni interne ai dati, come ad esempio l'andamento o l'individuazione di pattern, semplicemente ascoltando la Sonificazione prodotta. Questa tecnica è quindi da considerarsi alla pari di quelle tecniche visive che ci permettono di osservare tramite grafici il comportamento di un fenomeno o l'andamento di una curva, inoltre se queste due tipologie di tecniche, visive e uditive, cooperano l'una con l'altra, ecco che i vantaggi e le possibili applicazioni accrescono in maggior misura.

Nello sviluppare questo elaborato di tesi, si sono tenuti a mente questi aspetti positivi, che possono facilitare la comprensione di fenomeni come quelli meteorologici.

In [3] e [4] sono forniti notevoli esempi di come la Sonificazione applicata a dati meteorologici possa rendere più semplice, e anche più gradevole, la comprensione dell'evolversi di fenomeni meteorologici attraverso l'utilizzo di eventi sonori specifici. In questa applicazione, infatti, vengono sottoposti a Sonificazione dati che rappresentano l'andamento di vari parametri meteorologici come temperatura, umidità, intensità del vento e pressione atmosferica, permettendo così all'ascoltatore di avere un'idea più chiara

dell'andamento di un determinato parametro meteorologico nell'arco della giornata ascoltando pochi minuti di esecuzione sonora.

Considerando gli aspetti presentati dal progetto di Hermann, si è deciso di sviluppare questo elaborato sfruttando sia i vantaggi apportati dall'utilizzo di tecniche di Sonificazione come detto sopra, sia il diffondersi di piattaforme hardware, come Arduino, che rendono più semplice la realizzazione di prototipi per il trattamento di dati in tempo reale. Attraverso l'utilizzo congiunto di questi due elementi, si è sviluppato un prototipo per la produzione di suoni in tempo reale all'evolversi della situazione meteorologica analizzata.

Il lavoro di tesi è stato articolato in diverse fasi: dopo aver fornito nel Capitolo 2 una definizione precisa di Sonificazione e la collocazione di questa in una gerarchica interdisciplinare, si è provvisto ad elencare le principali tecniche di Sonificazione e a fornire una serie di esempi di utilizzo di queste tecniche in vari campi di applicazione. Nel Capitolo 3 sono presentate le varie tecnologie utilizzate, da Arduino per la parte di rilevazione fisica, all'uso di cSound per la sintesi sonora delle timbriche utilizzate nella Sonificazione.

Nel Capitolo 4 si presenta inizialmente un'introduzione alla meteorologia necessaria per comprendere a fondo il caso di studio proposto, successivamente si descrive lo sviluppo dei vari moduli software realizzati sfruttando l'IDE di Arduino e cSound per il trattamento dei dati raccolti, e infine sono analizzate le tecniche di sintesi sonora utilizzate per la realizzazione delle timbriche per la Sonificazione.

Nel Capitolo 5 sono presentati i possibili sviluppi e miglioramenti del prototipo realizzato e le considerazioni finali sul lavoro svolto.

Capitolo 2: Stato dell'arte

L'obiettivo di questo capitolo è quello di definire in modo esaustivo il significato e le caratteristiche del termine Sonificazione. Nella prima parte del capitolo viene presentata una struttura interdisciplinare dove è fornita una collocazione per la Sonificazione stessa e le sue differenti tecniche. Successivamente verranno presentate le differenti tecniche di Sonificazione ed infine verranno proposti degli esempi di applicazione di tali tecniche.

2.1 Collocamento nella disciplina dell'auditory display

Per Auditory Display si intende un processo che comprende sia l'utilizzo di tecniche per trasmettere informazione attraverso il suono, sia l'utilizzo di qualsiasi sistema in grado di generare e di trasmettere il suono stesso [1] e [5]. Questo settore di ricerca ha portato negli ultimi anni ad uno sviluppo di tecniche con grande applicazione in differenti ambiti.

L'organizzazione più rilevante che raggruppa tutte queste ricerche è l'ICAD (International Community for Auditory Display), che provvede all'organizzazione di incontri e seminari per la divulgazione di scoperte e ricerche in questo ambito [1s].

Tutte le tecniche che appartengono alla categoria dell'Auditory Display prevedono l'utilizzo del suono per comunicare informazione da un computer all'utente. La classificazione presentata non è tuttavia definita in modo netto, si possono infatti riscontrare alcune caratteristiche e modalità di operazione comuni a tutte. Le categorie principali di questa disciplina sono quattro: Audificazione, Icone sonore , Messaggistica vocale e Sonificazione [2s].

Per Audificazione [1] si intende una tecnica che permette l'analisi di una larga serie temporale(un grande numero di dati), mappando i valori a differenti livelli di pressione sonora.

In questa tecnica avremo quindi una serie di dati in ingresso che verranno associati a differenti livelli di pressione sonora in uscita; in questo modo è possibile identificare tramite l'ascolto la presenza di componenti periodiche nei dati, andando quindi a evidenziare le caratteristiche e l'andamento della serie temporale stessa.

La tecnica di Audificazione ha trovato applicazione in ambiti differenti, ad esempio nell'analisi di dati sismici o dei segnali neurofisiologici del cervello umano, dimostrando una

facilità di applicazione da parte dell'utente pari a quella ottenuta con l'utilizzo di tecniche di analisi visive (ad esempio l'analisi di un grafico o di uno spettrogramma).

La seconda categoria è quella delle Icone Sonore. Una Icona Sonora è un breve e distinto suono usato per rappresentare un evento specifico oppure per trasmettere informazione. In questa categoria è necessario distinguere due tipologie di icone chiamate rispettivamente Earcon e Auditory Icon. In [6] è chiarita la principale differenza tra le due categorie; la prima infatti utilizza elementi musicali, mentre la seconda utilizza suoni "quotidiani", quindi già conosciuti dall'ascoltatore, sfruttando l'associazione di un determinato suono ad una determinata fonte. Gli esempi più significativi di questa categoria si trovano nell'ambito dell'Informatica, in cui nello sviluppo di sistemi operativi molto conosciuti, coma ad esempio Windows, si è fatto largo uso di Icone Sonore per rappresentare azioni come lo svuotamento del cestino o la cancellazione di un file.

La terza categoria è quella della Messaggistica Vocale che utilizza tecniche di sintesi vocale o registrazioni del parlato per trasmettere informazione. Questa tecnica è spesso utilizzata per migliorare l'accessibilità nelle rubriche dei cellulari dove è possibile ascoltare la lettera alfabetica associata mentre si scorre la rubrica stessa.

La quarta categoria verrà descritta nel paragrafo seguente.

2.2 Definizione di Sonificazione

Una definizione generica di Sonificazione è stata fornita nell'Introduzione di questo elaborato, l'obiettivo di questo paragrafo è quello di fornirne una più chiara e precisa.

In [5] è riportata una definizione più esplicativa di Sonificazione, definita come "la trasformazione di relazioni nei dati in relazioni percepibili attraverso un segnale acustico con lo scopo di facilitarne la comunicazione e l'interpretazione". In questa descrizione è posta particolare attenzione alla funzione della Sonificazione, che è appunto quella di rendere più chiare all'ascoltatore le caratteristiche e gli attributi dei dati e delle relazioni fra questi, relazioni ben identificabili da chi ascolta, che si manifestano sotto forma di pattern ripetitivi, oscillazioni regolari o discontinuità nel suono.
Nel testo [5] di Hermann è fornita la definizione di Sonificazione tenendo conto dei requisiti che una qualsiasi tecnica, che utilizzi come input un set di dati e che generi in uscita eventi sonori, debba avere per rientrare nella categoria.

Questi requisiti sono riassumibili in quattro punti: il primo afferma che il suono generato dalla tecnica deve riflettere le proprietà o le relazioni dei dati in ingresso, ad esempio in un insieme di dati con un valore crescente sarà necessario scegliere un suono che rispecchi il crescere dei valori; il secondo afferma che la trasformazione deve essere sistematica, ovvero che esiste una definizione precisa di come i dati (e le relazioni fra essi) provochino cambiamenti nel suono.
Il terzo requisito afferma che la Sonificazione deve essere riproducibile, ovvero che forniti due set di dati identici nelle loro relazioni, il risultato sonoro deve essere strutturalmente identico.
L'ultimo requisito afferma che il sistema utilizzato per la Sonificazione deve poter essere utilizzato con dati differenti e anche in ripetizione con gli stessi dati.
In accordo con la definizione data sopra, la Sonificazione è quindi da considerarsi un metodo scientifico che porta a risultati riproducibili, utilizzando tuttavia, come canale percettivo, l'udito.

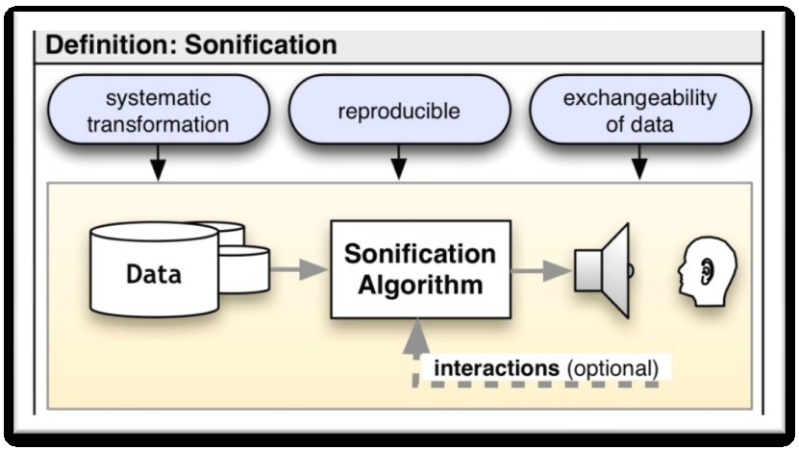


Fig. 2.1 Caratteristiche principali della Sonificazione

2.3 Tecniche di Sonificazione

In questo paragrafo sono presentate le principali tecniche di Sonificazione, queste tecniche presentano caratteristiche comuni alle altre appartenenti all'insieme più generale delle Auditory Displays; è normale quindi che alcune di queste vengano utilizzate in altri ambiti oltre a quello della Sonificazione.

La prima tecnica è chiamata Audification e si presenta come la tecnica più semplice e immediata, questa, infatti, consiste nel mappare direttamente dei dati in ingresso con dei valori di pressione sonora.

Questo tipo di tecnica è molto spesso utilizzata in grandi set di dati con componenti periodiche.

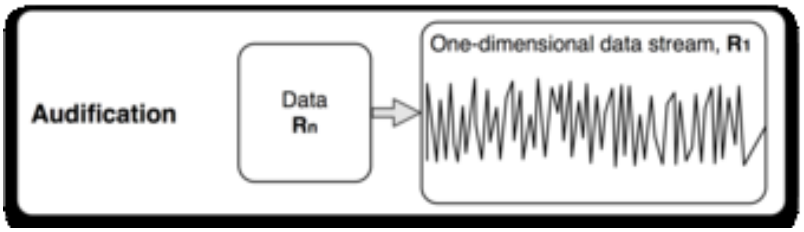


Fig. 2.2 Audification

La seconda tecnica di Sonificazione è chiamata Auditory Icons,che prevede l'utilizzo di uno specifico suono non vocale che deve essere facilmente riconoscibile e intuitivo poiché già conosciuto dall'ascoltatore. Questo tipo di tecnica è molto spesso utilizzata per rappresentare segnali di allarme oppure nei sistemi operativi, in cui si accompagnavano certe azioni con suoni caratteristici (ad esempio lo svuotamento del cestino con il suono di carta stracciata).

La terza tecnica è detta Earcons, questa tecnica subentra nel momento in cui viene a mancare la relazione fra il suono utilizzato ed un significato ben specifico.

Un Earcon può essere quindi definita come un messaggio musicale breve e strutturato, dove differenti proprietà musicali sono associate a differenti parametri dei dati che vengono rappresentati. La differenza principale tra le due tecniche di Sonificazione appena viste è che nell'utilizzo di una Earcon non è evidente la relazione fra il suono stesso e l'informazione che rappresenta ma deve essere in principio appresa dall'utente.
Attualmente questa tecnica viene sempre più spesso utilizzata, un classico esempio è il suono che viene emesso in un aereo di linea per la richiesta di assistenza da parte dei passeggeri.

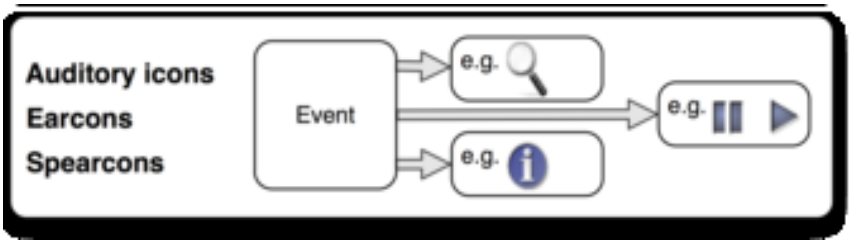


Fig. 2.3 Auditory Icons ed Earcons

La quarta tecnica è quella del Parameter Mapping, in cui si crea una stretta connessione fra le caratteristiche dei dati e i parametri di un evento sonoro (o di un generatore di suoni).

Per parametri sonori si intendono tutti quegli elementi che vanno a caratterizzare un determinato suono, ad esempio l'altezza, la durata o il timbro, parametri che nel loro variare avranno un comportamento simile a quello delle relazioni che si vogliono rappresentare nei dati analizzati. Per chiarire questa tecnica si può portare come esempio la Sonificazone di un grafico che descrive l'andamento di una funzione matematica; supponendo di associare all'andamento crescente della funzione l'aumento in altezza del suono, sarà possibile capirne l'andamento senza ricorrere alla visione del grafico stesso. Questa applicazione ha dato ottimi risultati in ambito scolastico, grazie alla quale è stato possibile facilitare la comprensione dei concetti spiegati a lezione a soggetti con disabilità visive.
Visto il gran numero di parametri sonori che è possibile utilizzare e le relazioni tra questi, questa tecnica è molto spesso scelta per l'analisi di dati multidimensionali [7].

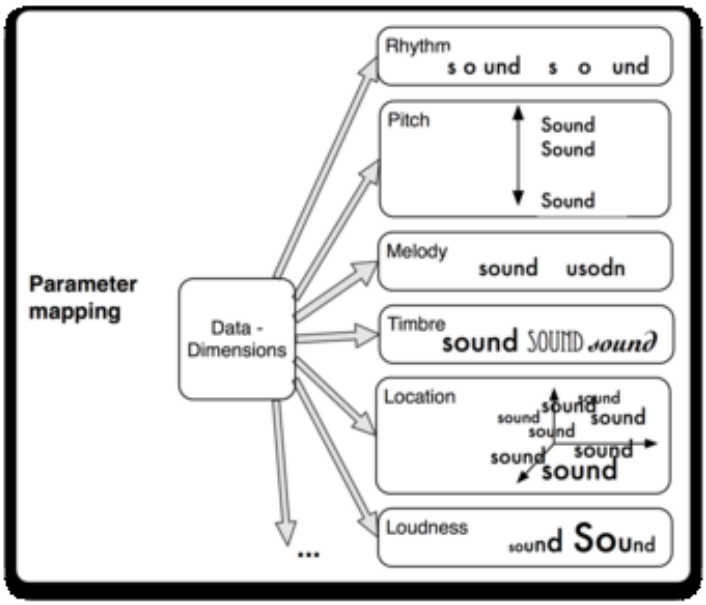


Fig. 2.4 Parameter Mapping

L'ultima tecnica è chiamata Model-Based Sonification. In questo tipo di tecnica si pone una particolare attenzione a come le risposte acustiche di un sistema vengano generate dalle azioni dell'utente [1] e [8]. In questo tipo di Sonificazione quindi, è necessario stabilire subito un modello dinamico di Sonificazione, ovvero la creazione di un "...[1]sistema in grado di generare suono virtuale" e come interagire con esso. L'interazione da parte dell'utente assume ,in questo tipo di tecnica, un ruolo fondamentale in quanto il modello stesso, se privo di interazione da parte dell'utente, non produce alcun suono. In alcuni casi non è necessario che l'utente sia una persona, infatti il modello dinamico creato per la Sonificazione può essere

messo in funzione anche da una variazione di dati percepiti tramite sensori o altro, esprimendo quindi la totale indipendenza del modello dal tipo di dati che gli viene fornito in ingresso, dati che possono provenire da qualsiasi ambito purché abbiano una struttura generica comune.

Fig 2.5 Model-based Sonification

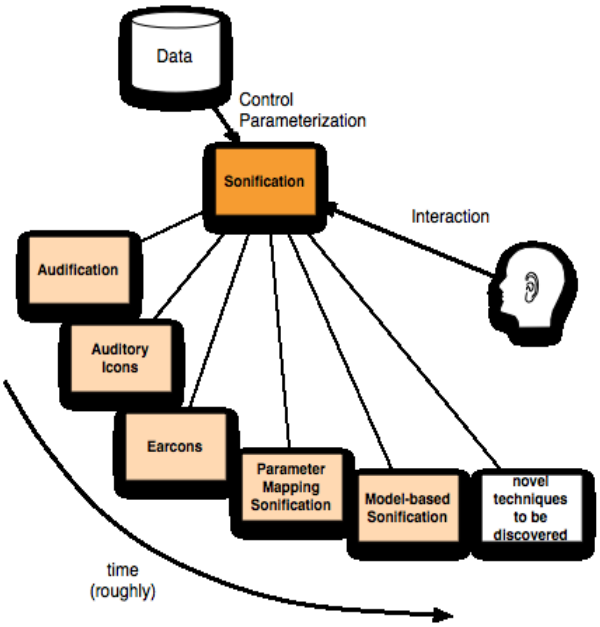


Fig. 2.6 Principali tecniche di Sonificazione

Tutte le tecniche citate presentano un possibile grado di interattività con l'utente.
Questo fattore è molto importante e come spiegato in [9], la Sonificazione interattiva può portare a risultati molto interessanti anche per un maggior coinvolgimento dell'utente stesso, ad esempio nella tecnica dell'Audificazione è possibile aumentare il grado di interattività fornendo all'utente una serie di controlli per spostarsi avanti o indietro nel file sonoro che rappresenta la Sonificazione.

2.4 Applicazioni

Le tecniche di Sonificazione hanno trovato un largo impiego nei più disparati campi, un primo esempio è quello della tecnologia assistiva. In questo ambito la Sonificazione ha trovato numerose applicazioni, anche se non è stata ancora sfruttata a pieno, ad esempio nell'accessibilità ai computer oppure come ausilio per la mobilità per soggetti con disabilità visive.

Per quanto riguarda l'accessibilità, la Sonificazione si propone come un canale di comunicazione alternativo rispetto allo schermo, in questo modo è possibile guidare l'utente nell'utilizzo di un sistema operativo o di un programma. In [1] vengono forniti i principali vantaggi dell'utilizzo di suoni per migliorare l'accessibilità rispetto all'utilizzo del braille nei computer; con l'utilizzo della Sonificazione infatti si evita tutto quel processo di apprendimento, che è invece obbligatorio nel braille, inoltre, essendo i computer moderni dotati di scheda audio, non vi è alcun costo aggiunto se non quello del software stesso.

L'accessibilità tramite la Sonificazione è raggiunta attraverso l'utilizzo di software chiamati "screen reader" [1], questi tipi di software si sono diffusi con la nascita dei primi sistemi operativi e offrono sostegno attraverso la riproduzione di suoni vocali e non per rappresentare ciò che è visualizzabile sullo schermo.

L'applicazione della Sonificazione come ausilio per la mobilità presenta due problematiche principali: la prima è la possibilità di evitare ostacoli a corto raggio, la seconda è quella più generica dell'ausilio alla navigazione.

Nella prima problematica presentata, la soluzione più efficiente prevede l'utilizzo di un sensore che generi segnali acustici a seconda della posizione dell'utente rispetto agli ostacoli che lo circondano, in questa situazione il suono deve essere percepito soltanto da chi deve essere guidato e non ascoltato da altri. Una possibile soluzione può essere l'utilizzo di cuffie che oltre a far percepire il suono solo all'ascoltatore che deve essere guidato, permettono di sfruttare le tecniche di spazializzazione che aggiungono informazioni sulla localizzazione degli ostacoli.

L'ausilio alla navigazione è invece fornito attraverso la creazione di un campo uditivo che descrive attraverso dei suoni la visuale dell'utente. La situazione visiva, in questo ambito, può essere registrata con una videocamera e poi tradotta opportunamente in linguaggio sonoro. Le problematiche che si presentano in questo caso sono quelle legate alla quantità di

informazione che è necessario rappresentare, infatti, attraverso l'utilizzo di una videocamera, sono trasmesse notevoli quantità di informazione che non devono essere totalmente rappresentate, poiché è importante fornire all'utente non il maggior numero di informazioni possibile ma quello necessario alla navigazione.

Un altro campo di applicazione della Sonificazione è quello del monitoraggio di processi. In questo campo si fa utilizzo di suoni per comunicare particolari situazioni e avvenimenti in uno specifico ambito; un esempio è dato dall'utilizzo di eventi sonori nelle industrie, in cui ad ogni macchina con uno specifico compito è assegnato un set di suoni per poter comunicare il proprio stato (ad esempio se la produzione procede regolarmente oppure è presente qualche problematica). In questo tipo di applicazione il vantaggio sta nel poter comunicare un certo tipo di informazione nel minor tempo possibile sfruttando a pieno il canale di comunicazione uditivo.

La Sonificazione è stata inoltre utilizzata nei segnali di allarme, questo campo è sicuramente uno dei primi ad essere stato sviluppato. Con l'utilizzo della Sonificazione come segnale d'allarme, l'obiettivo è quello di creare un suono sintetico che soddisfi alcune caratteristiche principali: la capacità di indicare quanto sia urgente la situazione rappresentata, la capacità di fornire indizi su che cosa ha innescato l'allarme e fornire informazione sulla localizzazione dell'evento che ha attivato l'allarme.

Un ultimo esempio di Sonificazione è dato dall'utilizzo di eventi sonori per esplorare grandi quantità di dati (*Data Mining*) [1]. In questo ambito l'utilizzo della Sonificazione facilita l'identificazione di pattern, rendendo più rapida l'analisi di grandi quantità di dati. Gli esempi di utilizzo in questo ambito sono molteplici: dall'esplorazione di dati geologici, all'analisi di dati biomedici fino ad arrivare alla navigazione Web e altro ancora.

Capitolo 3: Tecnologie utilizzate e cenni di meteorologia

In questo capitolo verranno descritti i concetti di meteorologia utilizzati per lo sviluppo dell'elaborato e le tecnologie scelte per la sua realizzazione; verranno forniti, inoltre, i principali motivi della scelta e dell'utilizzo sia di Arduino per la raccolta dei dati, che di cSound per la generazione del suono.

3.1 Cenni di Meteorologia

La meteorologia [3s] è quel ramo della scienza dell'atmosfera (l'insieme delle discipline che studiano l'atmosfera in ogni ambito possibile) che studia i fenomeni fisici che avvengono in essa. Questi fenomeni fisici vengono studiati e compresi attraverso l'analisi dei parametri che li vanno a caratterizzare, questi parametri sono: pressione atmosferica, temperatura dell'aria, umidità atmosferica e vento. Per comprendere a fondo la formazione dei fenomeni atmosferici, non basta analizzare soltanto gli elementi citati attraverso rilevazioni e misurazioni, ma è anche necessario lo studio di leggi fisiche che descrivono i comportamenti di questi nella formazione dei fenomeni stessi.

Il primo parametro descritto è la pressione atmosferica che è definita come il peso della colonna d'aria che sovrasta una superficie di un metro quadrato. Un valore notevole per la pressione atmosferica è quello calcolato a livello del mare a 0 ° C e ad una latitudine di 45° che misura 1033 grammi per centimetro quadrato, pressione che è esercitata dal peso degli strati soprastanti dell'atmosfera.

L'unità di misura della pressione è il Pascal [4s] ed è equivalente ad un Newton (unità di misura della forza) su metro quadro; in meteorologia è più largamente usato l'ettopascal (hPa).

La variazione di pressione atmosferica nel tempo è indice di un cambiamento nella condizione meteorologica [10] [5s], questo perché precorre o accompagna importanti processi dinamici dell'atmosfera in quanto è la principale causa dello spostamento delle masse d'aria atmosferiche.

La pressione atmosferica è inoltre influenzata dalla temperatura e dall'umidità, infatti più l'aria è umida più è leggera ed inoltre una massa d'aria calda è più leggera rispetto ad una di aria fredda.

Per questi motivi si può affermare che la pressione atmosferica è un parametro molto importante per quanto riguarda la comprensione dei fenomeni meteorologici, ma è anche uno dei parametri che più difficilmente si possono sfruttare se non si è in possesso di apparecchiature adatte e di conoscenze approfondite nel campo della meteorologia. In linea generale, si è diffuso il principio che un valore basso di pressione sia associato ad una situazione di "brutto tempo", mentre un valore alto di pressione è associato ad una situazione di "bel tempo", in realtà è più corretto affermare che, tenendo in considerazione i valori di umidità e temperatura, la variazione barometrica può indicarci un miglioramento oppure un peggioramento nelle condizioni climatiche. Tenendo in considerazione l'importanza della variazione di pressione atmosferica, si possono delineare dei metodi generici che descrivono la relazione che intercorre tra variazione di pressione e cambiamento nella condizione meteorologica [11].

Un calo di 1-2 hPa (ettoPascal) in 3 ore di solito anticipa un peggioramento che si manifesta nelle successive 24-48 ore.

Un calo di 5-6 hPa in 3 ore, invece, indicano un peggioramento già in atto e suggerisce fenomeni violenti.

Queste variazioni sono da considerarsi come linea generale, è necessario infatti considerare che, nell'arco della giornata, la pressione ha una andamento proprio che raggiunge il valore massimo intorno alle ore 10:00, per poi decrescere fino alle 16:00 dove arresta la sua caduta e cresce fino alle 22:00, dove arresta la sua crescita e decresce raggiungendo un valore minimo intorno alle 4:00.

Un'altra caratteristica della pressione che va tenuta in considerazione è che, a seconda della zona geografica in cui ci si trova, la variazione di pressione è sempre abbastanza contenuta, in Italia, ad esempio, si può andare in casi estremi da un valore minimo di 980 hPa ad un massimo di 1045 hPa.

Il secondo parametro preso in considerazione è la temperatura dell'aria che si può esprimere in gradi Celsius oppure in gradi Fahrenheit. In meteorologia questa grandezza è considerata in maniera puramente quantitativa poiché le nostre impressioni sensoriali si rivelano insufficienti o inadeguate.

La temperatura dell'aria diminuisce con l'altitudine in quanto il riscaldamento dell'atmosfera avviene principalmente per irraggiamento termico terrestre, cioè cessione di calore da parte del suolo. Altri fattori che in generale influiscono sulla temperatura dell'aria sono:

- la latitudine: all'aumentare di questo fattore aumenta l'inclinazione dei raggi solari e diminuisce l'intensità della radiazione assorbita;
- l'ora del giorno: la radiazione aumenta durante il mattino fino a raggiungere l'apice nelle ore centrali per poi tornare a diminuire e azzerarsi subito dopo il tramonto e per tutta la notte;
- l'esposizione e l'inclinazione del terreno rispetto ai raggi solari.

Oltre a questi fattori la copertura nuvolosa, il pulviscolo atmosferico, il vapore acqueo e la vegetazione attenuano l'intensità della radiazione solare assorbita e quindi hanno effetto sulla temperatura.

Parlando di umidità atmosferica ci si può riferire a tre parametri significativi:

- Umidità assoluta: è la quantità di vapore acqueo espressa in grammi contenuta in un metro cubo d'aria. L'umidità assoluta aumenta all'aumentare della temperatura;
- Umidità specifica: è il rapporto della massa del vapore acqueo e la massa d'aria umida;
- Umidità relativa: è il rapporto percentuale tra la quantità di vapore contenuta da una massa d'aria e la quantità massima (livello di saturazione) che il volume d'aria può contenere nelle stesse condizioni di temperatura e pressione. L'umidità relativa è un parametro dato dal rapporto tra umidità assoluta e umidità di saturazione.

L'umidità relativa è il parametro che viene preso in considerazione in meteorologia, questo parametro è particolarmente legato al fenomeno della pioggia in quanto, durante una precipitazione,

l'umidità relativa dell'aria in ambiente esterno raggiunge tipicamente valori dell' 80-90%.

Il vento è il movimento di una massa d'aria atmosferica da un'area con alta pressione ad un'area con bassa pressione. In meteorologia sono di particolare importanza la velocità del vento e la direzione del vento. La velocità del vento può essere espressa in metri al secondo, chilometri orari o nodi.

3.2 Tecnologie utilizzate

In questo paragrafo verranno descritte le principali componenti hardware e software utilizzate per la realizzazione dell'elaborato.

3.2.1 Arduino, Weather Shield e sensori

Arduino è una piattaforma Hardware e Software opensource per la prototipazione [6s].

Gli ideatori di questa piattaforma, raggruppati sotto il nome di *Arduino Team*, hanno creato inizialmente questo dispositivo per offrire agli studenti di *Interactive Design Institute* di Ivrea una soluzione semplice e comprensibile per realizzare progetti interattivi elettronici senza dover utilizzare linguaggi di programmazione complessi e conoscenze avanzate che spesso non sono prerogativa dei professionisti in ambito creativo.

Poiché lo scopo finale era la massima diffusione di Arduino, il team di sviluppo decise di applicare all'hardware il concetto di "open" che negli anni precedenti aveva già dimostrato tutta la propria forza in ambito software, portando così Arduino ad essere conosciuto ed apprezzato a livello mondiale.

L'ambiente di sviluppo integrato (IDE) di Arduino è scritto in linguaggio Java, ed è derivato dall'IDE creato per il linguaggio di programmazione Processing (linguaggio di programmazione per lo sviluppo di giochi, animazioni e contenuti interattivi) e per il progetto Wiring (piattaforma di sviluppo open source).

Il linguaggio utilizzato dall'IDE di Arduino è il C/C++ e prevede uno schema generale simile per tutti i programmi, chiamati *sketch*, che è costituito dalla dichiarazione di due funzioni chiamate rispettivamente `setup()` e `loop()`.

La funzione `setup()` è invocata una sola volta ed al suo interno vengono definiti i settaggi iniziali come l'inizio della comunicazione seriale oppure la modalità ingresso o uscita dei pin analogici e digitali della scheda.

La funzione `loop( )` è invece invocata ripetutamente e la sua esecuzione si interrompe solo allo spegnimento della scheda, in questa funzione saranno descritte le operazioni svolte da Arduino.

Per la rilevazione dei dati, si è scelto di utilizzare la piattaforma hardware Arduino Uno Rev. 3, tra varie opzioni prese in considerazione questa versione di Arduino si è presentata come la migliore sia per il prezzo contenuto sia per le sue caratteristiche.

La scheda è basata sul microcontrollore Atmel ATmega 328 e possiede quattordici ingressi/uscite digitali, sei ingressi analogici e porta USB per la comunicazione seriale. Per la scrittura del codice si è utilizzata l'IDE 1.0.6, scaricabile gratuitamente dal sito degli sviluppatori.

Fig. 3.1 Arduino UNO Rev3

Per la rilevazione dei dati meteorologici è stato acquistato uno shield per Arduino chiamato Weather Shield acquistato dall'azienda di fabbricazione elettronica Sparkfun.

Questo shield permette la misurazione di:

- pressione atmosferica: attraverso il sensore MPL3115A2 e la sua libreria esprime la misura di pressione atmosferica in Pascal, è inoltre in grado di fornire l'altitudine in metri e in piedi dal livello del mare;
- umidità relativa: il sensore HTU21D e libreria corrispondente riportano il valore di umidità relativa attraverso la semplice chiamata di una funzione;
- luminosità: rilevata attraverso il sensore ALS-PT19, riporta una misurazione di luminosità espressa in Lux (unità di misura relativa alla luce visibile, e pertanto dipendente dalle caratteristiche dell'occhio umano attraverso la curva di sensibilità dell'occhio alla radiazione luminosa);

- temperatura: questo parametro è misurato sia dal sensore di pressione atmosferica sia dal sensore di umidità. Nello sviluppo di questo elaborato è stata scelta la temperatura fornita dal sensore di pressione atmosferica in quanto considerata più attendibile dopo dei test di confronto.

Oltre a questi sensori già integrati nel Weather Shield, è possibile aggiungere un set di sensori per pioggia, velocità e direzione del vento; è inoltre possibile connettere un modulo GPS GP 635-T per la localizzazione e l'orario.

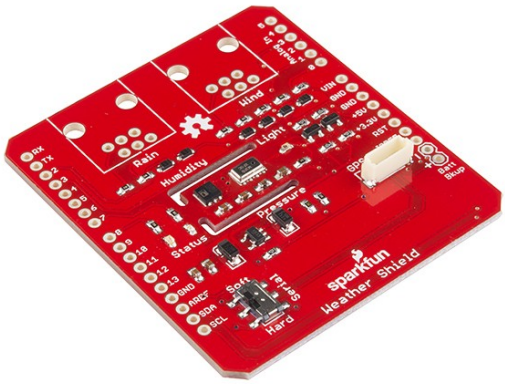


Fig. 3.2 Weather Shield di SparkFun

La velocità del vento è stata rilevata attraverso l'utilizzo di un anemometro collegato attraverso una breadboard ad Arduino stesso. L'anemometro considerato è dotato al suo interno di un sensore reed che chiude il contatto al passaggio di un piccolo magnete, il magnete è solidale alla parte in movimento (quella con le coppette) e ad ogni giro passa in prossimità del sensore chiudendo il circuito.

Fig. 3.3 L'anemometro a coppette utilizzato

3.2.2 cSound

Csound è un software per la sintesi digitale diretta del suono realizzato da Barry Vercoe allo M.I.T (Massachusetts Institute of Technology) [12]. Il software è in continuo sviluppo e aggiornamento: infatti si tratta di un software di pubblico dominio, e chiunque è libero di utilizzarlo, modificarlo e ampliarlo. Csound è scritto in linguaggio C e ha una propria sintassi che richiede la scrittura di due testi denominati rispettivamente Orchestra (file .orc) e Partitura (file .sco), a partire dalla versione 3.50 di CSound, è possibile includere questi due testi in un unico file con estensione .csd.

Nel file Orchestra si andranno a specificare tutte le caratteristiche dei suoni che si vogliono ottenere come, ad esempio, frequenza e ampiezza e i metodi di generazione dei suoni stessi.

Il file Partitura contiene le note che compongono l'esecuzione e altre caratteristiche legate alla composizione come ad esempio il tempo metronomico.

Per la realizzazione di questo elaborato sono state sfruttate le principali tecniche di sintesi come la sintesi additiva, modulazione di frequenza e sintesi per modelli fisici; inoltre sono state utilizzate delle funzioni che permettono di aprire una comunicazione su porta seriale rendendo possibile il dialogo con Arduino.

La descrizione fornita in questo paragrafo è tuttavia molto limitata e non fornisce un'idea chiara delle enormi potenzialità del software, con cSound infatti è possibile creare qualsiasi tipo di sintesi sonora, generare suono in real-time, supportare il protocollo di comunicazione MIDI e molte altre caratteristiche che lo rendono un software estremamente utile e versatile.

Fig. 3.4 cSound e alcune possibili applicazioni

Capitolo 4: Realizzazione del software

L'obiettivo di questo capitolo è quello di descrivere e spiegare la realizzazione del software sviluppato: a partire dalla rilevazione dei dati eseguita con Arduino, si forniranno tutti gli elementi necessari per comprendere l'analisi dei dati da parte di cSound con la formulazione di modelli per la Sonificazione, e come questi dati, attraverso una struttura precisa, verranno utilizzati per produrre i vari eventi sonori.

In generale il software realizzato presenta quattro ambienti sonori, ognuno caratterizzato da un determinato range di valori dei differenti parametri meteorologici.

Dopo aver iniziato la rilevazione dei dati, questi vengono analizzati per determinare in quale range di valori ricadono le rilevazioni, in questo modo verrà attivato solo l'ambiente sonoro corrispondente. Questa selezione funge da "macro modello" grazie al quale è possibile, una volta attivato un ambiente sonoro, avere una prima idea dei valori analizzati.

Successivamente all'interno di ogni ambiente sonoro sono presenti degli strumenti realizzati per ricordare una determinata situazione meteorologica; questi strumenti a loro volta subiranno delle variazioni secondo modelli predefiniti e basati sui parametri meteorologici, questo fornirà all'utente un'idea ancora più precisa dei parametri rilevati.

Infine si attua un controllo del volume per ogni ambiente sonoro attraverso una variabile calcolata dinamicamente a seconda dei parametri. Questa variabile avrà valore unitario (massimo passaggio di volume) se le condizioni rilevate corrispondono alle condizioni archetipiche dell'ambiente sonoro attivato, mentre avranno un valore tendente allo zero quanto più ci si allontana da questa situazione, rendendo così più dolci le transizioni fra due ambienti sonori.

Il software realizzato è composto da tre differenti moduli:

- primo modulo: rilevazione dei dati e invio da parte di Arduino;
- secondo modulo: ricezione ed elaborazione dei valori tramite cSound;
- terzo modulo: sonificazione dei dati con sintesi sonora in cSound.

4.1 Primo modulo: la rilevazione e la trasmissione dei dati tramite Arduino

Nel capitolo 2 di questo elaborato sono state elencate le caratteristiche delle schede Arduino e Weather Shield utilizzati per la rilevazione dei dati. In questo paragrafo si fornirà una spiegazione del codice implementato attraverso l'IDE di Arduino per la raccolta dei dati.

Il codice scritto prende spunto dalla gestione dei messaggi nel protocollo di comunicazione MIDI. In questo protocollo, infatti, due o più dispositivi MIDI si scambiano messaggi tra loro, messaggi che sono ben identificati da un codice che, a sua volta, va a definire le caratteristiche e le funzioni che ogni specifico messaggio inviato svolge.

Nella realizzazione del programma che invia i byte dati da Arduino a cSound, è stata implementata una struttura simile, che prevede un codice identificativo (sensor_ID) per ogni sensore, un messaggio con il numero di byte di dati che seguiranno (byte length) e un certo numero di byte dati inviati successivamente.

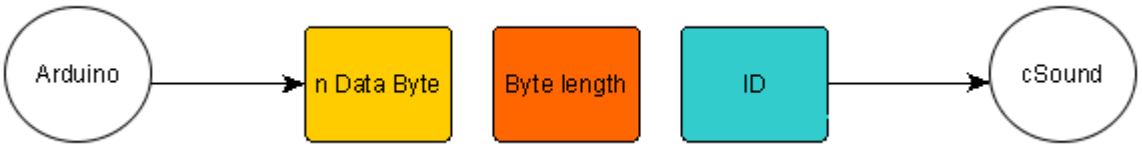


Fig. 4.1 Schema semplificato del messaggio

Il codice identificativo del sensore è il primo byte della serie di messaggi che verrà inviato; all'inizio del programma vengono dichiarati i vari ID per i sensori, questi devono essere uguali nel codice di cSound.

```
// Value IDs (must be between 128 and 255)

byte pressureID = 128;

byte humidityID = 129;

byte lightID = 130;

byte temperatureID = 131;

byte windID = 132;
```

Una volta definiti i codici identificativi è stata creata nella sezione `loop()` del codice di Arduino una struttura di controllo `switch`. Questa struttura, attraverso una variabile di tipo intero chiamata `select`, seleziona a turno i vari sensori, che interroga attraverso una funzione di libreria, ottenendo il valore letto dal sensore; questo valore, e l'identificatore del sensore corrispondente, vengono passati come argomenti della funzione `serial_send()` che al suo interno ha una chiamata alla funzione `serial_send_recursiv()`.

```
// Read one value at the time (determined by the select variable)
       switch (select) {
         case 0: {   //pressure
           float press_value = myPressure.readPressure();
           press_value = press_value / 100;  //conversione in ettopascal
           serial_send(pressureID, press_value);
         }
         break;
         ... //Altri sensori
// Update the select (0, 1 and 2)
       select = (select+1)%3;
```

Le funzioni `serial_send()` e `serial_send_recursiv()` hanno il compito di inviare, rispettando l'ordine, il byte ID, il byte length e i vari byte dati.

La funzione `serial_send()` ha come argomenti il codice identificativo e il valore letto dal sensore. Dopo essere stata chiamata, questa funzione esegue il comando `Serial.write(id)` che invia a cSound , tramite comunicazione seriale, il byte contenente l'identificatore del sensore. La seconda istruzione svolta da `serial_send()` è quella di chiamare `serial_send_recursiv()`, passandole come argomenti il valore preso dal sensore e un `byte length` uguale a 1.

```
void serial_send(byte id, int number)
{
  Serial.write(id);
  serial_send_recursiv(number, 1);
}
```

`Serial_send_recursiv` è una funzione ricorsiva che riceve come argomenti il valore del sensore e il byte length. Una volta ricevuti questi due numeri, la funzione procede con una struttura di controllo che verifica se il valore del sensore è trasmissibile in un unico byte (`if (number < 128 )`), se questa condizione è vera allora la funzione procede all'invio del byte length (che sarà uguale a 1) e dell'unico byte di dati contenente il valore raccolto dal sensore. Nel caso in cui il valore non sia trasmissibile in un unico byte di dati, la funzione comincia il processo di ricorsione che esegue ogni volta un'operazione di spostamento a destra di sette bit, in questo modo dopo aver incrementato il byte length di uno è possibile rappresentare valori più grandi attraverso l'utilizzo di più byte di dati.

```
// Recursiv function that sends the bytes in the right order
void serial_send_recursiv(int number, int bytePart)
{
  if (number < 128) {        // End of recursion
    Serial.write(bytePart);  // Send the number of bytes first
  }
  else {
    serial_send_recursiv((number >> 7), (bytePart + 1));
  }
  Serial.write(number % 128);  // Sends one byte
}
```

In conclusione si fornisce un esempio di simulazione del codice scritto per Arduino in cui le istruzioni di `Serial.write()` verso cSound vengono sostituite da istruzioni `Serial.print()` per stampare i risultati a schermo.

```
Valore press: 962.54 ID del sensore: 128 numero di byte:2 Byte: 111 Byte:
1000010


Valore humid: 44 ID del sensore: 129 numero di byte:1 Byte: 101100
Valore light: 675 ID del sensore: 130 numero di byte:2 Byte: 101 Byte:
100011
Valore temp: 16 ID del sensore: 131 numero di byte:1 Byte: 10000
```

4.1.1 Il codice per la rilevazione del vento tramite anemometro

La rilevazione della velocità del vento è stata sviluppata acquistando un comune anemometro dotato di un circuito reed (vedi Capitolo 3 paragrafo 3.2.2) che viene chiuso al passaggio di un magnete.

Questo sensore non fa parte dell'insieme dei sensori forniti con il Weather Shield e per questo non è presente una libreria specifica per poter gestire la lettura della velocità del vento.

Con l'ausilio della rete sono stati trovati degli sketch di esempio per Arduino che forniscono un possibile metodo di calcolo, tra questi ne è stato scelto uno che verrà persentato in seguito:

Velocità [m/s] = 2π * r [m] * f

formula fisica per calcolare la velocità noto il numero di giri al secondo f, dove r = raggio, ovvero la distanza tra l'asse dell'anemometro e le coppette.

Per ottenere la velocità in chilometri orari:

Velocità [Km/h] = Velocità [m/s] * 3.6

La variabile f è ottenuta come:

Conteggio * (1 / (2 * T))

dove Conteggio indica il numero di impulsi, ovvero passaggi del magnete, e T è il periodo trascorso tra il primo passaggio del magnete (primo impulso) e il primo passaggio successivo che rispetti la condizione `if (Tempo >= TempoMax)`, ovvero che abbia una differenza di tempo maggiore di due secondi (valore preimpostato in `TempoMax`) per evitare il conteggio di brevi folate di vento. Il periodo T viene moltiplicato per due tenendo conto che l'anemometro è dotato di due magneti, quindi si avranno due impulsi ogni giro completo.

Estratto del codice:

```
void loop()
{
  Statoreed = digitalRead(reedPin);


  if (Statoreed != Statoreed_old)
  {
```

```
    Statoreed_old = Statoreed;

   if (Statoreed == HIGH)
    {
      if (Conteggio == 0){ TempoStart =  millis();}


      Conteggio = Conteggio + 1;

      Tempo = ( millis() - TempoStart);

      if (Tempo >=  TempoMax)

      {
            float deltaTempo = ( Tempo/1000.0);

            float Metris= (Conteggio*Pi*raggio)/deltaTempo;

            float Kmora = (3.6*Conteggio*Pi*raggio)/deltaTempo;

           Conteggio = 0; // azzeriamo il conteggio per nuova lettura
           delay(5000);// attesa per altra lettura
      }
    }
  }

}
```

Questo metodo fornisce il risultato corretto in presenza di vento, ma non fornisce alcun risultato se , in assenza di vento, le coppette dell'anemometro non girano; tuttavia per poter includere la rilevazione del vento nella parte di codice di Arduino è necessario restituire un valore ogni volta che il sensore viene interrogato.

Per questo motivo sono state apportate delle modifiche al codice base.

4.1.2 Prima variante del codice

Per far si che il codice restituisca un valore di velocità anche qualora questo sia nullo, è stata aggiunta una condizione `else` nella quale si ricade se lo strumento non comunica alcuna variazione di stato (passaggio del magnete).

```
void loop()
{
  Statoreed = digitalRead(reedPin);

  if (Statoreed != Statoreed_old){

      //calcolo velocità vento

}
```

```
else {
      float vento_sound = 0;
      serial_send(windID, vento_sound);
}
```

Con questo metodo si è riscontrata una problematica di rilevazione della velocità in quanto, a causa delle limitazioni dello strumento stesso, anche se in presenza di vento l'anemometro restituiva più volte un valore nullo.

4.1.3 Variante definitiva

Per ovviare alle problematiche riscontrate nelle rilevazione della velocità del vento, si è deciso di introdurre una variabile contatore `i`. Questa variabile contatore è utilizzata per fornire un limite di tempo entro il quale l'anemometro sia in grado di fornire un valore corretto, scaduto questo lasso di tempo si ricade nella condizione `else` che restituisce un valore nullo.

```
case 4: {  //vento
           i = 1;
           while( 1 == 1 ){
             if( i < 10000 && i != 0 ){
                Statoreed = digitalRead(reedPin); // legge il contatto reed
                if (Statoreed != Statoreed_old) //cambiamento stato
                 { //calcola velocità vento invia e i = 0 }
             } else {
                  i++;
                  vento_sound = 0;
                  serial_send(windID, vento_sound);
              }
          }
```

Dopo una serie di prove è stato definito un valore ottimale di `i = 3000` entro il quale l'anemometro, nonostante le sue limitazioni, è in grado di misurare una velocità in presenza effettiva di vento, altrimenti, per valori di `i > 3000`, si può essere ragionevolmente certi dell'assenza di vento.

4.2 Secondo modulo: ricezione ed elaborazione dei valori tramite cSound

La ricezione dei valori da parte di cSound è possibile attraverso l'utilizzo di tre opcodes: `serialBegin`, `serialRead` e `serialWrite`:

- `SerialBegin`: apre una porta seriale per Arduino, prendendo come argomenti il nome della porta seriale, il valore di baudrate (numero di simboli trasmessi al secondo in un sistema di trasmissione digitale) e restituendo un numero di porta che verrà utilizzato da `serialRead`;
- `serialRead`: legge dei dati dalla porta ottenuta da `serialBegin` e restituisce un byte dati che rappresenta il valore letto;
- `serialWrite`: scrive dei dati sulla porta specificata come argomento.

Il primo passo nello sviluppo del codice è stato quello di dichiarare delle variabili globali che immagazzinano i valori dei differenti sensori ottenuti da Arduino.

```
gkPressure init 0
gkHumidity init 0
gkLight init 0
gkTemperature init 0
gkWind init 0
```

Successivamente si procede all'apertura di una porta di comunicazione tra cSound e Arduino.

```
iPort serialBegin "/COM6", 9600 ;connect to the arduino baudrate = 9600
if( iPort != 1) then ;ARDUINO CONNESSO ( NON SONO IN MODALITà SIMULAZIONE )
serialWrite iPort, 1 ;Triggering the Arduino (k-rate)
```

Dopo aver connesso Arduino e cSound tramite serialBegin, viene eseguito un controllo sul valore restituito da questa funzione (`iPort`), un valore diverso da uno comunica che l'esecuzione è andata a buon fine. Una volta stabilita la connessione si esegue `serialWrite` che ha la funzione di mandare un byte ad Arduino che, una volta ricevuto, comincerà ad inviare dati (fase di triggering).

Dopo questo processo di collegamento fra i due dispositivi inizia la fase di trasmissione dati.

```
kValue = 0
kType serialRead iPort ; Read type of data (press, humd, light, temp)

if (kType >= 128) then

kIndex = 0
kSize serialRead iPort ;legge la length del messaggio da Arduino
```

Questa fase consiste in tre parti principali ripetute per ogni sensore: la prima è quella di lettura dell'ID del sensore, che sarà il primo byte ad essere inviato da Arduino, la seconda lettura sarà quella del `byte_length` ,ovvero il numero di byte di dati che seguiranno per la rappresentazione del valore, la terza fase è costituita da un ciclo che ricostruisce il valore del sensore stesso.

```
loopStart:
kValue = kValue << 7
kByte serialRead iPort ;legge i dati dei sensori da Arduino
kValue = kValue + kByte ;ricostruisce il byte....
loop_lt kIndex, 1, kSize, loopStart ;...iterando per kIndex volte, sommando
1 ogni volta, arrivando fino a kSize (numero di byte inviati da Arduino)
endif
```

Il ciclo `loopStart:` esegue un'operazione di shift a sinistra dei valori ricevuti sommando il risultato in `kValue` e riottenendo così il valore originale del sensore; questo ciclo verrà eseguito `kSize` volte ovvero il numero di byte di dati indicato dal `byte_length`.

Dopo aver ricostruito il byte si assegna il valore del sensore alla corrispettiva variabile globale, dichiarata all'inizio del programma, tramite una serie di istruzioni condizionali che leggono la variabile `kType` contenente l'ID del sensore.

```
if (kType == 128) then
gkPressure = kValue
elseif (kType == 129) then ; This is the humd
gkHumidity = kValue
elseif (kType == 130) then ; This is the light
gkLight = kValue
elseif (kType == 131) then ;This is the temp
gkTemperature = kValue endif
```

4.2.1 Elaborazione dei dati e fromulazione dei modelli

Dopo aver ricevuto e ricostruito i dati rilevati dai vari sensori, questi vengono utilizzati all'interno di modelli per effettuare la Sonificazione.

I modelli definiti nel software mettono in relazione i parametri meteorologici e restituiscono un valore che verrà opportunamente riscalato, attraverso l'opcode `Scalk`, per poter essere utilizzato all'interno di uno strumento del terzo modulo.

L'opcode `Scalk` è più precisamente un opcode definito da un utente (UDO User Defined Opcode) e si differenzia dagli opcode standard di cSound poiché il suo funzionamento viene definito nel file Orchestra del codice che lo utilizza.

Questo opcode riceve cinque argomenti, il primo è il valore da riscalare, che sarà il risultato del modello fisico, il secondo e il terzo valore sono il massimo e il minimo valore che raggiunge la variabile da riscalare, gli ultimi due valori sono gli estremi del range che da definire.

Il calcolo del range di riscalamento è definito a seconda delle caratteristiche di ogni singolo strumento mentre i valori massimi e minimi dei vari modelli sono stati calcolati utilizzando i valori già definiti per la scelta dei timbri (vedi paragrafo successivo).

Di seguito viene resentato un esemio di definizione di un modello:

```
kval_3 = gkLight / gkHumidity ; modello
imax_piano = gi_bello_luce_upper / gi_bello_umid_down ;valore massimo del
modello
imin_piano = gi_bello_luce_down / gi_bello_umid_upper ;valore minimo del
modello
gkmod_piano Scalk kval_3, imin_piano, imax_piano, 50, 300 ;riscalamento
valori
```

I criteri considerati nella creazione dei modelli, che verranno esposti nel paragrafo 4.3.4, dipendono dalle caratteristiche dei timbri e dalle variazioni dei parametri meteorologici di ogni ambiente sonoro.

4.3 Terzo modulo: sonificazione dei dati con sintesi sonora in cSound

Il terzo modulo sviluppato si occupa della sintesi sonora dei dati ricevuti dal secondo modulo, la sintesi è ottenuta attraverso vari "strumenti tematici" che sono utilizzati in un determinato scenario sonoro (ad esempio una giornata soleggiata o una nuvolosa).

Di seguito è presentato uno schema generale del terzo modulo:

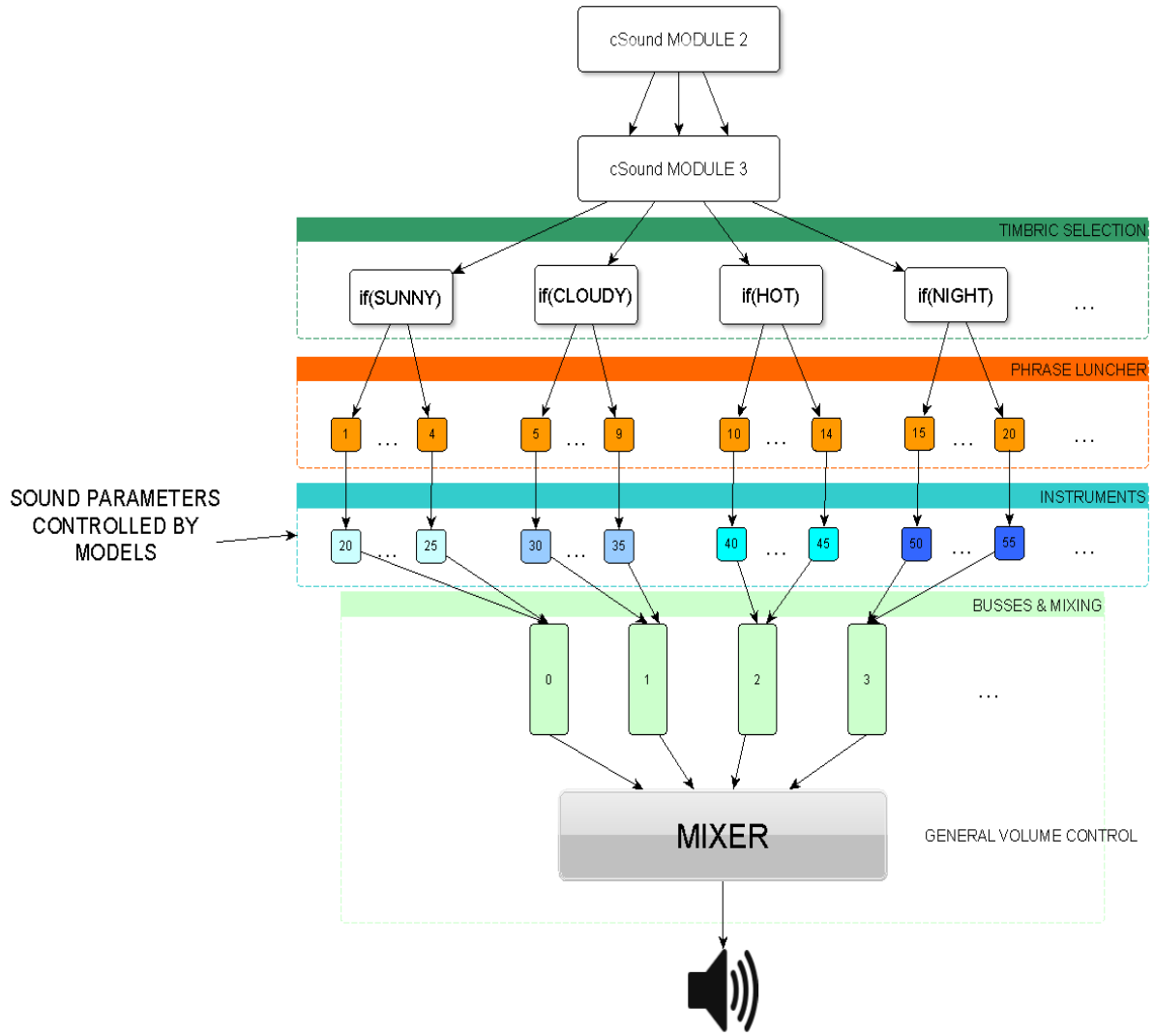


Fig. 4.2 Schema generale terzo modulo

4.3.1 Selezione timbriche

Il primo compito svolto dal terzo modulo è quello di analizzare i dati presi da Ardunio e selezionare, attraverso una serie di condizioni `if`, quali strumenti devono produrre suono; gli

strumenti selezionati genereranno quindi un ambiente sonoro consono alla situazione climatica rilevata da Arduino.

Ogni ambiente sonoro è definito da un range di valori per ogni parametro rilevato, ad esempio sono stati impostati i seguenti valori per definire un ambiente notturno sereno:

```
;IF NOTTE

if( gkLight >= gi_notte_luce_down && gkLight <= gi_notte_luce_upper

&& gkPressure >= gi_notte_press_down && gkPressure <= gi_notte_press_upper
&& gkHumidity >= gi_notte_umid_down && gkHumidity <= gi_notte_umid_upper

&& gkTemperature >= gi_notte_temp_down && gkTemperature <=
gi_notte_temp_upper ) then
```

dove `gi_notte_luce_down` e `gi_notte_luce_upper` sono i valori limite che verranno confrontati col parametro `gkLight` rilevato da Arduino. I valori limite per ogni parametro sono stabiliti in seguito a numerose prove di rilevazione dei parametri meteorologici, è necessario considerare che questi parametri variano a seconda della posizione geografica in cui ci si trova, ad esempio il valore di pressione in millibar associato al "bello" o "brutto" tempo è differente a seconda dell'altitudine.

4.3.2 Phrase launcher

Per far si che ogni strumento generi un determinato evento sonoro è stato sviluppato un blocco di codice denominato Phrase launcher [7s].

La logica di questo strumento è quella di leggere dei valori da tabelle e variabili preimpostate e generare un evento sonoro attraverso l'opcode `event`.

Le tabelle contengono i pitch delle note da eseguire (`giPitches`), le durate delle note (`giDurations`) e il tempo di attesa tra una nota e l'altra (`giStepLen`).

```
;TABELLE NOTE E DURATE PER NOTTE
giPitches ftgen 0, 0, -7, -2, 7.00, 7.00, 7.00, 7.00
giStepLen ftgen 0, 0, -7, -2, 4, 2, 3 , 4
giDurations ftgen 0, 0, -7, -2, 5, 5, 5, 5
```

Per gestire i vari strumenti sono stati creati altrettanti phrase launcher, questi sono tutti sincronizzati con una variabile `gkMetro` che definisce attraverso l'opcode `metro` il tempo metronomico.

Successivamente si procede con l'estrazione dei valori pitch e durata dalle tabelle dichiarate inizialmente e si genera, attraverso l'opcode event, un evento sonoro (nota `i` dello score) specificando tutti gli argomenti di cui necessita.

Il passo successivo è quello di estrarre il primo valore dalla tabella `giStepLen`, in questo modo si va ad associare un valore alla variabile `kwait` che servirà a mettere in attesa il phrase luncher per un certo periodo specificato appunto da `ksteplen`. Una volta scaduto il tempo di attesa si procede estraendo dalle tabelle i valori successivi (identificati tramite la variabile `kstepnum` che funge da indice per le tabelle), fino a quando non si verifica la condizione `if (kstepnum >= iseqlength)` che indica la fine della sequenza melodica e l'inizio della sua ripetizione attivata dall'istruzione `kstepnum = 0`.

```
if( // condizione meteo ) then
      if (gkMetro == 1) then
            if (kwait == 0) then
                  kpitch_1 table kstepnum, giPitches
                  kdur table kstepnum, giDurations
                  event "i", 11, 0, kdur, kamp, kpitch
                  ksteplen table kstepnum, giStepLen
                  kwait = ksteplen - 1
                  kstepnum = kstepnum + 1
                  if (kstepnum >= iseqlength) then
                        kstepnum = 0
                  endif
            else
                  kwait = kwait - 1
            endif
      endif
else
      kstepnum = 0
      kwait = 1
endif
```

Per gli instruments che necessitano di polifonia, è sufficiente impostare più tabelle con i vari pitch e durate, specificando nel phrase luncher la creazione di due note contemporanee tramite `event`.

```
kpitch_1 table kstepnum, giPitches_1
```

```
kpitch_2 table kstepnum, giPitches_2
kdur table kstepnum, giDurations
event "i", 11, 0, kdur, kamp, kpitch_1
event "i", 11, 0, kdur, kamp, kpitch_2
```

4.3.3 Instruments

Per la generazione dei suoni che eseguiranno la Sonificazione si sono utilizzate molteplici tecniche di sintesi offerte da cSound.

Gli strumenti creati sono raggruppabili secondo l'ambiente sonoro di cui fanno parte, di seguito verranno elencati i paesaggi sonori e le timbriche create:

- **Giorno sereno**: pianoforte con opcode `prepiano`, synth brass con modulazione di frequenza (FM) e lettura campioni;
- **Giorno nuvoloso**: sintesi additiva, sintesi FM e lettura campioni;
- **Notte serena**: synth brass con FM, sintesi sonora FM di una Celesta (strumento idiofono che produce suono attraverso lamelle di metallo) e lettura campioni
- **Giorno caldo**: sintesi additiva, sintesi sonora di un flauto ottenuto attraverso modellazione fisica e lettura campioni.

Le letture dei campioni sono state eseguite principalmente tramite l'opcode `diskin`, ma sono stati utilizzati anche gli opcode `soundin` e `loscil`.

Gli instruments che prevedono l'utilizzo della sintesi additiva sono stati utilizzati per creare un sottofondo sonoro, il suono creato è ottenuto attraverso la tecnica additiva tradizionale che prevede la creazione di differenti oscillatori per le varie armoniche, in cui ognuno con il proprio inviluppo e la somma dei segnali risultanti.

L'opcode `prepiano` per la timbrica del pianoforte è basato su un modello fisico e richiede numerosi argomenti tra cui la frequenza di base, il numero di corde simulate per ogni nota, parametri dimensionali, la posizione iniziale del martelletto e la velocità di pressione dei tasti.

Il synth brass è stato creato attraverso modulazione FM creando tre oscillatori per l'inviluppo, la modulante e la portante.

Il suono del flauto è, come quello del pianoforte, molto complesso essendo basato su un modello fisico che tiene in considerazione parametri come l' *embouchure* (la maggior o

minore pressione della bocca intorno all'imboccatura) e la quantità d'aria immessa inizialmente.

4.3.4 Variazione dei parametri sonori in base ai modelli

I parametri calcolati attraverso i modelli definiti nel secondo modulo, vengono utilizzati per variare alcuni parametri degli strumenti che generano i timbri.

In questo paragrafo verranno elencati gli strumenti e i parametri modificati, raggruppati per ambienti sonori, ponendo particolare attenzione alla relazione fra i modelli (e quindi le variazioni dei parametri meteorologici) e i cambiamenti nel suono dei vari strumenti; in questo modo si definiranno le idee e i concetti utilizzati a livello percettivo per comunicare le variazioni della situazione climatica.

- **Notte serena:** Questo ambiente sonoro rappresenta una situazione di condizioni stabili in un ambiente notturno.

  L'idea generale per questo ambiente è quella di comunicare un senso di calma e serenità.

  Il primo strumento definito simula il suono *brass* tipico di alcuni sintetizzatori generato attraverso sintesi FM.

  Il modello fisico definito è **(Temperatura * Pressione) / Umidità** e il valore, opportunamente scalato, in uscita viene utilizzato per variare il parametro kmax dello strumento che rappresenta un fattore moltiplicatore per l'onda modulante.

  Attraverso la variazione di questo parametro si ottiene un suono più "cupo" e "chiuso" in presenza di un basso valore di temperatura e pressione e un alto valore di umidità, al contrario un alto valore di temperatura e pressione e un basso valore di umidità corrispondono ad una situazione climatica migliore e il suono risulterà più "chiaro" e "aperto".

  Il secondo strumento simula il suono di una celesta, il modello fisico ha come variabile il valore di luce che rende il suono più o meno squillante variando il pitch delle note prodotte attraverso l'opcode octave() che calcola un fattore per portare l'esecuzione un'ottava più bassa.

  Il terzo strumento si occupa della lettura di un campione di arpa ed è direttamente legato alla presenza di vento o assenza di vento.
- **Giorno sereno:** questo ambiente sonoro è connesso ad una situazione meteorologica stabile durante il giorno.

Il primo strumento produce il timbro di un pianoforte attraverso l'opcode `prepiano`.

Il modello definito è **Luce / Umidità** che corrisponde ad una situazione serena per alti valori di luce e bassi di umidità.

Il parametro controllato nella simulazione del piano è `ivel` che rappresenta la velocità con cui vengono premuti i tasti del pianoforte; in questo modo avremo un suono meno incisivo per valori bassi di luce e alti di umidità e più incisivo per il contrario, rendendo così lo strumento sensibile alle variazioni di questi due parametri.

Il secondo strumento utilizzato simula un `brass` come nell'ambiente sonoro notturno e condivide lo stesso modello e le stesse modalità di controllo del suono discusse per l'ambiente notturno.

Per quanto riguarda la presenza o no di vento, anche in questo caso è stato utilizzato un campione di arpa letto a seconda del valore di vento rilevato.

- **Giorno caldo:** questo ambiente sonoro rappresenta una situazione di alta temperatura e pressione con bassi valori di umidità, generando sonorità basate sulla scala minore armonica per ricordare ambienti estremamente caldi e aridi come il deserto.

  Il primo strumento sintetizza il suono di un flauto a cui è stato associato il seguente modello: **Umidità / (Temperatura * Luce)**. Il parametro variato è il pitch delle note prodotte, in questo modo si otterranno dei suoni più alti se il livello di umidità è basso e i livelli di temperatura e luce sono alti, associando così una maggiore presenza del timbro per situazioni che ricordano giornate estive afose e secche.

  Il secondo strumento è ottenuto tramite sintesi additiva controllato dal seguente modello: **(Temperatura * Pressione) / Umidità**.

  Le variazioni nel suono di questo strumento sono operate da un filtro passa basso `butterlp` che varia la frequenza di taglio a seconda del modello definito, generando così un suono più o meno ricco di armoniche, e quindi più o meno presente, a seconda dei valori considerati.

  Anche in questo caso per il trattamento del vento si è scelto di utilizzare un campione di sitar.

- **Giorno nuvoloso:** l'ambiente sonoro in questione rappresenta una situazione meteorologica instabile con bassi valori di pressione, luce e temperatura e un alto valore di umidità.

  Anche in questo ambiente è stato utilizzato il suono di pianoforte che produce, al contrario dell'ambiente *giorno sereno*, delle note singole che variano il parametro

`ivel` (velocità di pressione tasti) a seconda del seguente modello: **Umidità** / **Pressione**.

Il secondo strumento è generato tramite sintesi FM, in questo strumento sono stati associati due differenti parametri, rispettivamente indice di modulazione e pitch delle note, a due differenti modelli: **Temperatura / Luce** e **Umidità / Luce**.

La variazione di questi tre parametri porta ad effetti particolari legati allo strumento che in generale possono essere descritti come suoni che esprimono uno stato di tensione a seconda di quanto i parametri considerati si avvicinino ad una situazione di tempo instabile.

I restanti strumenti di questo ambiente utilizzano campioni di fenomeni temporaleschi o di precipitazioni a seconda del valore di Umidità.

Tabella strumenti, funzioni di mapping e possibli stati d'animo

	Strumento	Funzione Mapping	Stati d'animo
1	BRASS	Dyn_mod = (Temperatura * Pressione) / Umidità dyn_mod = dinamica ampiezza modulante	Calma, Tranquillità, Serenità
2	CELESTA	Pitch = Luce	Calma, Tranquillità, Serenità
3	ARPA	Lancio campione = Presenza di vento	La comparsa di vento è accompagnata da un accordo di arpa
4	PIANO	Velocity = Luce / Umidità	Calma e Serenità con accordi a bassa velocity Tensione per note singole con Velocity crescente
5	MODULAZIONE FM	Indice di Modulazione = Temperatura / Luce Pitch = Umidità / Luce	Maggiore tensione al variare di pitch e Indice di modulazione
6	FLAUTO	Pitch = Umidità / (Temperatura * Luce)	Sensazione di afa, caldo all'aumentare del pitch, serenità
7	ADDITIVA	Frequenza di taglio = Temperatura * (Pressione / Umidità)	Afa, caldo all'aumentare della frequenza di taglio
8	SITAR	Lancio campione = Presenza di vento	La comparsa di vento è accompagnata da un accorodo di sitar
9	CAMPIONI AMBIENTALI		Specifici per ogni ambiente sonoro

Fig. 4.3 Elenco strumenti associati alle funzioni di mapping e possibili stati d'animo

	Ambienti Sonori		
	Scena	Strumenti	Mood
1	NOTTE SERENA	Brass, Celesta, Arpa, Campioni ambiente	Calma, Tranquillità, Serenità
2	GIORNO SERENO	Piano, Brass, Arpa, Campioni ambiente	Calma, Tranquillità, Serenità
3	GIORNO CALDO	Flauto, Additiva, Sitar Campioni ambiente	Afa, Caldo, Disagio
4	GIORNO NUVOLOSO	Modulazione FM, Piano Campioni ambiente	Tensione, Disagio

Fig. 4.4 Elenco strumenti per ogni ambiente sonoro con i possibili stati d'animo

4.3.5 Calcolo volume generale e mixaggio

L'ultima sezione del software realizzato svolge la duplice funzione di controllo del volume predefinito per ogni singolo ambiente sonoro e controllo dinamico del volume a seconda dei parametri meteorologici.

Per quanto riguarda il controllo del volume predefinito ogni gruppo di strumenti utilizzati per descrivere una determinata situazione meteorologica è inviato ad un buss del mixer utilizzando gli opcode `MixerSetLevel` e `MixerSend`, che hanno permesso la regolazione dei volumi in fase di creazione dei timbri:

- `MixerSetLevel`: ha come argomenti il numero di strumento da cui proviene il suono (`isend`), il numero di buss a cui inviare il segnale e una variabile kgain che esprime il valore di guadagno dello strumento;
- `MixerSend`: ha come argomenti il segnale stesso, `isend` come numero dello strumento, `ibuss` come numero del buss e il canale specifico del buss che si desidera associare allo strumento.

Oltre all'utilizzo di questi opcode, per la fase di regolazione del volume dinamico è stata dichiarata una variabile chiamata `aTOT` che è la somma di tutti i singoli ambienti sonori trattati (`atot_notte`, `atot_caldo`, `atoto_bello` e `atot_brutto`) moltiplicati per un coefficiente

chiamato `gkVol_nome` (ad esempio `gkVol_caldo`) ,definito per ogni singolo ambiente, che controlla il volume in base al valore dei parametri rilevati.

Il calcolo dei differenti coefficienti `gkVol_nome` è preceduto dalle stesse istruzioni condizionali utilizzate per l'avvio dei phrase launcher (paragrafo 4.3.2), in questo modo il calcolo stesso inizia solo se ci si trova nella corrispondente situazione meteorologica, evitando così che i differenti ambienti sonori suonino in contemporanea.

Variabile aTOT:

```
aTOT = atot_notte * gkVol_notte + atot_caldo * gkVol_caldo + atot_bello * gkVol_bello + atot_brutto * gkVol_brutto
```

I differenti coefficienti sono calcolati secondo la seguente formula:

gkVol_nome = (n - abs(gi_nome_parametro1_mean - gkParametro) / (gi_nome_parametro1_mean - gi_nome_parametro1_down) - abs(gi_nome_parametro_n_mean - gkParametro) / (gi_nome_parametro_n_mean - gi_nome_parametro_n_down) / n

dove:

- nome: identifica l'ambiente sonoro per cui è calcolato il volume;
- parametro_n: identifica uno dei parametri meteorologici rilevati;
- gi_nome_parametro_n_mean: è la media dei due valori estremi gi_nome che definiscono il range descritto nel paragrafo 4.3.1;
- abs: è una funzione che restituisce il valore assoluto dell'argomento fornito.

Applicando questa formula e definendo degli opportuni range di valori, è possibile ottenere un volume che sarà minimo in situazioni di transito tra due ambienti sonori e massimo nel caso in cui i parametri meteorologici coincidano con i valori medi del range definito.

Capitolo 5: conclusioni e sviluppi futuri

Il seguente capitolo riassume il lavoro svolto in questo elaborato, ponendo particolare attenzione allo scopo prefissato all'inizio della stesura, i contributi apportati, le problematiche affrontate e gli sviluppi futuri.

5.1 Scopo della ricerca e problematiche risolte

Lo scopo principale di questo elaborato è quello di fornire un'applicazione che, a seconda dei parametri rilevati, produce una serie di ambienti sonori con caratteristiche che si evolvono a seconda della situazione meteorologica letta in tempo reale mediante Arduino ed alcuni sensori.

In generale, il funzionamento di questo software è basato sulla generazione di ambienti sonori predefiniti che a seconda dei parametri meteorologici rilevati inizieranno o interromperanno la propria esecuzione; inoltre, all'interno di ogni ambiente sonoro, verranno variati alcuni parametri degli strumenti che li compongono in base ai modelli stabiliti, ottenendo così la Sonificazione dei dati.

La prima fase dello sviluppo è stata dedicata allo studio delle principali discipline coinvolte (Sonificazione di dati, Informatica, Psicoacustica e Meteorologia) e alla comprensione dei software utilizzati per la realizzazione (cSound e l'IDE di Arduino).

Successivamente si è reso necessario sviluppare un metodo per rendere possibile la comunicazione seriale fra Arduino e cSound, compito effettuato dalla prima parte del software contenuta nel primo e nel secondo modulo. In questa parte è risultata di fondamentale importanza la comprensione dei principi di comunicazione seriale di Arduino e lo studio degli opcode necessari in cSound per la ricezione dei dati, sviluppando un metodo di trasmissione dati in parte ispirato allo scambio di messaggi del protocollo MIDI.

Dopo aver interfacciato Arduino a cSound, si è ricercato un metodo in cSound per poter generare eventi sonori "automatci", ovvero generare suono in base ai parametri meteo senza dover scrivere partiture nel file Score del software. Questo obiettivo è stato raggiunto attraverso lo sviluppo dei *phrase player*, a loro volta basati sull'opcode `event` che genera eventi direttamente dal file Orchestra.

Il lavoro descritto fino a questo punto costituisce la base funzionale del software, dopo aver raggiunto questo risultato, lo sviluppo si è concentrato sulla dichiarazione dei modelli di Sonificazione e sulla creazione dei timbri adatti per ogni ambiente sonoro.

Per quanto riguarda la seconda problematica elencata, il lavoro si è basato sulle nozioni acquisite nella prima parte dello studio di cSound e sull'idea generale di creare un ambiente sonoro che rimandi ad una specifica situazione meteorologica secondo interpretazione personale.

Dopo aver sviluppato i timbri, si sono definiti i modelli di Sonificazione, tenendo conto dei parametri controllabili nei vari strumenti e associandoli a variazioni nei parametri meteorologici.

5.2 Sviluppi futuri

I possibili sviluppi riguardano sia la parte hardware che la parte software.

Per la parte hardware un possibile miglioramento è l'aggiunta di ulteriori sensori per basare la Sonificazione su un numero maggiore di parametri, un esempio è il sensore per la rilevazione della pioggia (rain gauge).

Per la parte software un importante miglioramento riguarda la comunicazione seriale tra Arduino e cSound e la generazione di suoni; nelle prove effettuate si è infatti presentata una problematica nella sintesi sonora che risulta essere, soprattutto con l'aggiunta del sensore per la velocità del vento (vedi paragrafo 4.1.1, 4.1.2 e 4.1.3), interrotta per brevi periodi. Per poter risolvere questa problematica è necessario uno studio più approfondito sulla comunicazione tra Arduino e cSound in real-time, rendendo chiare le meccaniche di trasmissione che dipendono da variabili come frequenze di campionamento, buffer di cSound e baudrate di Arduino.

Un altro importante sviluppo riguarda l'aggiunta di ambienti sonori per altre situazioni meteorologiche non prese in considerazione, come ad esempio una giornata di nebbia o la presenza di neve, rendendo così possibile la sonificazione in qualsiasi set di dati venga fornito.

L'ultimo sviluppo proposto si pone come obiettivo quello di migliorare la struttura del codice, soprattutto nella dichiarazione dei differenti phrase players, questi infatti sono dichiarati per ognuno degli strumenti creati. Il miglioramento proposto riguarda la riduzione del numero di

phrase players, rendendo quelli presenti adattabili a differenti parametri, come ad esempio il numero di voci o il tipo di strumento da controllare e scegliendo dinamicamente a seconda dell'ambiente sonoro le tabelle da cui estrarre i valori di durata e pitch delle note.

Bibliografia

[1] T. Hermann, A. Hunt, J. G. Neuhoff: *The Sonification Handbook*, Logos Publishing House, Berlin, 2011.

[2] B. N. Walker, M. A. Nees: *Theory of Sonification*. In T. Hermann, A. Hunt & J. Neuhoff, Principles of Sonification: An Introduction to Auditory Display and Sonification, 2001.

[3] T. Hermann, J. M. Drees, H. Ritter: *Broadcasting Auditory Weather Reports – A Pilot Project*. In atti della conferenza internazionale su Auditory Display (ICAD), 2003.

[4] J. H. Flowers, L. E. Whitwer, D. C. Grafel, C. A. Kotan: *Sonification of Daily Weather Records: Issues of Perception, Attention and Memory*. In Design Choices (2001). *Faculty Publications, Department of Psychology.* Paper 432.

[5] T. Hermann: *Taxonomy and Definitions for Sonification and Auditory Display*. In atti della conferenza internazionale su Auditory Display (ICAD), 2008.

[6] R. Minghim, A.R. Forrest: *An Illustrated Analysis of Sonification for Scientific Visualisation* 1995, *IEEE*.

[7] J. H. Flowers: *Thirteen Years of reflection on auditory graphing: Promises, pitfalls, and potential new directions*. In atti del Simposio su Auditory Graph, 2005.

[8] T. Hermann, H. Ritter: *Listen to your Data: Model-Based Sonification for Data Analysis*. In Advances in intelligent computing and multimedia systems, G. E. Lasker, Ed., Baden-Baden, Germany, 08 1999, pp. 189–194, Int. Inst. for Advanced Studies in System research and cybernetics.

[9] T. Hermann, A. Hunt: *An Introduction to Interactive Sonification*, In atti della conferenza internazionale su Auditory Display (ICAD), 2008.

[10] M. Giuliacci, A. Giuliacci, P. Corazzon: *Manuale di Meteorologia*, Milano, Alpha Test, 2010.

[11] M. Giuliacci, A. Giuliacci, P. Corazzon: *Prevedere il tempo con Internet*, Milano, Alpha Test, 2007.

[12] R. Bianchini, A. Cipriani: *Il Suono Virtuale: Sintesi ed Eleaborazione del Suono - Teoria e Pratica con cSound*, Roma, ConTempoNet, 2011.

Sitografia

[1s] http://www.icad.org/

[2s] http://en.wikipedia.org/wiki/Auditory_display

[3s] http://it.wikipedia.org/wiki/Meteorologia

[4s] http://it.wikipedia.org/wiki/Pascal_%28unit%C3%A0_di_misura%29

[5s] http://www.centrometeo.com/articoli-reportage-approfondimenti/fisica-atmosferica/4624-alta-bassa-pressione-bello-cattivo-tempo

[6s] http://it.wikipedia.org/wiki/Arduino_%28hardware%29

[7s] http://www.csounds.com/journal/issue15/phrase_loops.html

Appendice A: DataSheet sensori Arduino

EVERLIGHT

DATASHEET

Ambient Light Sensor
Surface - Mount
ALS-PT19-315C/L177/TR8

Features

- Close responsively to the human eye spectrum
- Light to Current, analog output
- Good output linearity across wide illumination range
- Low sensitivity variation across various light sources
- Guaranteed temperature performance, -40°C to 85°C
- Wide supply voltage range, 2.5V to 5.5V
- Size : 1.7mm(L)*0.8mm(W)*0.6mm(H)
- RoHS compliant and Pb Free package

Description

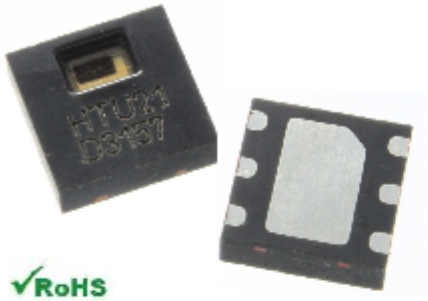
The ALS-PT19-315C/L177/TR8 is a low cost ambient light sensor, consisting of phototransistor in miniature SMD. EVERLIGHT ALS series product are a good effective solution to the power saving of display backlighting of mobile appliances, such as the mobile phones, NB and PDAs. Due to the high rejection ratio of infrared radiation, the spectral response of the ambient light sensor is close to that of human eyes.

Applications

- Detection of ambient light to control display backlighting
  - Mobile devices – mobile phones, PDAs
  - Computing device – TFT LCD monitor for Notebook computer
  - Consumer device – TFT LCD TV, plasma TV, video camera, digital camera, toys
- Automatic residential and commercial management
- Automatic contrast enhancement for electronic signboard
- Ambient light monitoring device for daylight and artificial light
  - Street light, CCD/CCTV

HTU21D(F) Sensor

Digital Relative Humidity sensor with Temperature output



- **DFN type package**
- **Relative Humidity and Temperature Digital Output, I²C interface**
- **Fully calibrated**
- **Lead free sensor, reflow solderable**
- **Low power consumption**
- **Fast response time**

DESCRIPTION

HTU21D(F), the new digital humidity sensor with temperature output of MEAS is about to set new standards in terms of size and intelligence: Embedded in a reflow solderable Dual Flat No leads (DFN) package of 3 x 3 mm foot print and 0.9 mm height it provides calibrated, linearized signals in digital, I²C format.

HTU21D(F) digital humidity sensors are dedicated humidity and temperature plug and play transducers for OEM applications where reliable and accurate measurements are needed. Direct interface with a micro-controller is made possible with the module humidity and temperature digital outputs. HTU21D(F) digital humidity sensors are low power consumption designed for high volume and cost sensitive applications with tight space constraints.

Every sensor is individually calibrated and tested. Lot identification is printed on the sensor and an electronic identification code is stored on the chip – which can be read out by command. Furthermore, the resolution of HTU21D(F) digital humidity sensor can be changed by command (8/12bit up to 12/14bit for RH/T), low battery can be detected and a checksum improves communication reliability.

With made improvements and the miniaturization of the sensor the performance-to-price ratio has been improved – and eventually, any device should benefit from the cutting edge energy saving operation mode.

Optional PTFE filter/membrane (F) protects HTU21D digital humidity sensors with temperature output against dust, water immersion as well as against contamination by particles. PTFE filter/membrane preserves a high response time. The white PTFE filter/membrane is directly stuck on the sensor housing.

FEATURES

- Full interchangeability with no calibration required in standard conditions
- Instantaneous desaturation after long periods in saturation phase
- Compatible with automatized assembly processes, including Pb free and reflow processes
- Individual marking for compliance to stringent traceability requirements

APPLICATIONS

- Automotive: defogging, HVAC
- Home Appliance
- Medical
- Printers
- Humidifier

Freescale Semiconductor
Data Sheet: Advance Information
An Energy Efficient Solution by Freescale

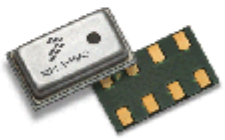
Document Number: MPL3115A2
Rev 2.2, 07/2012

I²C Precision Altimeter

The MPL3115A2 employs a MEMS pressure sensor with an I²C interface to provide accurate Pressure/Altitude and Temperature data. The sensor outputs are digitized by a high resolution 24-bit ADC. Internal processing removes compensation tasks from the host MCU system. Multiple user-programmable, power saving, interrupt and autonomous data acquisition modes are available, including programmed acquisition cycle timing, and poll-only modes. Typical active supply current is 40 μA per measurement-second for a stable 30 cm output resolution. Pressure output can be resolved with output in fractions of a Pascal, and Altitude can be resolved in fractions of a meter.

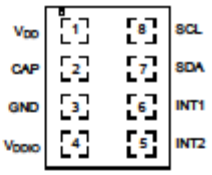
The MPL3115A2 is offered in a 5 mm by 3 mm by 1.1 mm LGA package and specified for operation from -40°C to 85°C. Package is surface mount with a stainless steel lid and is RoHS compliant.

MPL3115A2
50 to 110 kPa



LGA PACKAGE
5.0 mm by 3.0 mm by 1.1 mm

Top View



Pin Connections

Features

- 1.95V to 3.6V Supply Voltage, internally regulated by LDO
- 1.6V to 3.6V Digital Interface Supply Voltage
- Fully Compensated internally
- Direct Reading, Compensated
  - Pressure: 20-bit measurement (Pascals)
  - Altitude: 20-bit measurement (meters)
  - Temperature: 12-bit measurement (degrees Celsius)
- Programmable Events
- Autonomous Data Acquisition
- Resolution down to 1 ft. / 30 cm
- 32 Sample FIFO
- Ability to log data up to 12 days using the FIFO
- 1 second to 9 hour data acquisition rate
- I²C digital output interface (operates up to 400 kHz)

Application Examples

- High Accuracy Altimetry
- Smartphones/Tablets
- Personal Electronics Altimetry
- GPS Dead Reckoning
- GPS Enhancement for Emergency Services
- Map Assist, Navigation
- Weather Station Equipment

ORDERING INFORMATION

Device Name	Package Options	Case No.	# of Ports			Pressure Type			Digital Interface
			None	Single	Dual	Gauge	Differential	Absolute	
MPL3115A2	Tray	2153	•					•	•
MPL3115A2R1	Tape & Reel (1000)	2153	•					•	•

This document contains information on a new product. Specifications and information herein are subject to change without notice.

freescale Xtrinsic

Ringraziamenti

Ringrazio i miei genitori per il supporto ricevuto durante il mio percorso di studi, per la loro pazienza e per avermi dato la possibilità di raggiungere questo importante risultato.

Ringrazio Marco e Valeria per gli incoraggiamenti e per i bellissimi momenti passati insieme.

Ringrazio il professor Luca Andrea Ludovico e il professor Giorgio Presti per tutto l'aiuto ricevuto durante la stesura di questo elaborato.

Un enorme ringraziamento a tutti i compagni di Università: Teo, Walter, Fill, Paga, Gio, Stic, Budd, Opo, Nava, Gibbo, Fuji, Kako, Case, Massa, Simo, Marci, Ciodo, Albe e Pisan (chi ho dimenticato vince una birra) per le risate e per tutto il tempo passato insieme durante questo percorso.

Un ringraziamento speciale a Marco Piffa, Dani Spoldi, Fabietto Festa e Beppone Frisicaro con cui ho passato la maggior parte del tempo in Università, per gli incoraggiamenti ricevuti (sono un bullo) e per i risultati che assieme abbiamo raggiunto, siete tutti dei bulli.

Grazie anche al mitico Izza, per le suonate e per il suo fondamentale aiuto nell'ultimo periodo di studi.

Un enorme grazie a tutti i miei compagni delle superiori: Alessandro "Sale" Indoni, Gabriele "Fari John Joseba Farinjao" Farina, Andrea "Ferruz Truce" Ferraro, Riccardo "Riki" Montanari, Luca "Resco Jam" Resconi, Nicolò "Calsciadeur" Berton, Michele "Mich" D'Andria, Emanuele "Dino" Di Natale , Walter "Issimo" Gazzotti, "Pocciolone National Geographical Channel Cuore Pulsante" (di cui non si conosce il vero nome) e alle splendide Giada Goffredo, Jessica Pistoletti, Caterina Zanzi, Evelina Bianchi, Ester Gandini e Silvia Riboni (sono andato in ordine di banco non litigate!), per tutti i momenti passati assieme e per la pazienza nel sopportare il mio bellissimo carattere.

L'ultimo ringraziamento va a Daniele "Salvo" Salvato, per i chilometri fatti insieme in macchina, per le serate passate e future e per tutto il supporto che mi ha fornito, grazie per questa splendida amicizia.