POLITECNICO DI TORINO

Computer Engineering

Master's Degree in Embedded Systems

Master Thesis

B-Glove
a wireless MIDI instrument for disability

Overcoming communicative barriers: an application of the Globality of languages

Advisor:
Prof. Antonio Servetti

Candidate:
Sebastiano Franchina

Tutors
Paolo Cavagnolo
Riccardo Turino

October 2017

To my parents
(Ai miei genitori)

La musica, come la vita, si può fare in un solo modo: insieme.
La musica ci insegna la cosa più importante di tutte: ascoltare.
La musica è la nostra vera terapia.

Music, like the life, can be made in a single manner: together.
Music teaches us the most important thing: listening.
Music is our real therapy.

Ezio Bosso

Acknowledgements

I decided to write this section in Italian. If it is not important for the people which will read this thesis for a scientific interest, it is very important for me to be understood in the best way possible by the people I will cite.

Montale diceva che la nostra vita rassomiglia in gran parte al volo dei gabbiani: perlustrano il mare in cerca di pesce, si cibano, trovano una spiaggetta tranquilla e vi si appollaiano, per poi ripartire, come se su ogni spiaggia un cartello recasse scritto "Più in là". La nostra vita è fatta di bivi. Li troviamo, consapevolmente o meno, mentre percorriamo le nostre strade, ma soprattutto quando una strada che stavamo percorrendo finisce e si dirama in due o più direzioni.

Finita per me la strada dell'Università, del Politecnico di Torino, iniziata 5 anni fa (sembra sia ieri), desidero e ritengo sia opportuno rivolgere un pensiero a coloro i quali questa strada me l'hanno, magari non sempre resa più facile, nè più difficile, ma sicuramente riempita, o che hanno creato per me delle zone di sosta nelle quali trovare ristoro e sostentamento per proseguire nella vita e nel mio percorso di studi. Lo voglio fare, non con un grazie, ma, andando contro ciò che ho detto all'inizio, con un "acknowledgement", un grato riconoscimento, un "Reddite quae sunt Caesaris Caesari".

Desidero iniziare manifestando la mia profonda gratitudine a coloro i quali mi hanno dato la possibilità di realizzare questo lavoro che, come dirò nell'introduzione, rappresenta non solo il coronamento del mio percorso di studi, ma anche un'esperienza di crescita d'inestimabile valore.

Quindi, in ordine, la Prof.ssa Caire, per la sua gentilezza e disponibilità nell'indicarmi con probabilità di successo del 100 percento colui il quale mi ha proposto questo lavoro di tesi e ne è stato Relatore, il Prof. Servetti, che a sua volta ringrazio per avere impacchettato per me una tesi per come la desideravo, cioè una combinazione tra ciò che mi piace definire il mio pane, l'informatica, e ciò che mi piace definire la mia principale passione, la musica. Lo ringrazio inoltre per la sua disponibilità e gentilezza, rare a sentire esperienze d'altri laureandi. Ringrazio Paolo, il cui contagioso entusiasmo è stato un trampolino di lancio fondamentale per l'inizio di questo lavoro.

Un pensiero va sicuramente a Samir, ai suoi genitori ed agli altri ragazzi di "Tra silenzio e baccano". L'esperienza con loro mi riporta alla mente il mio film preferito, Patch Adams, e, in particolare, la scena in cui lui - accusato di praticare la medicina senza titolo - dichiara: "Tutti coloro che vengono al mio ranch sono pazienti, ma tutti coloro che vengono al mio ranch sono anche medici". Se è vero che nel mio piccolo, ho potuto essere d'aiuto o di conforto, o se ho potuto contribuire a creare qualche istante di benessere e/o divertimento per questi ragazzi, è anche vero che la loro voglia, l'imponenza con la quale si sono schierati ed hanno vinto contro i loro limiti, sono stati carburante per le mie piccole, in confronto, lotte quotidiane.

Come non rivolgere immensa gratitudine a Riccardo, che alla fine ha potuto vedere e mettere le mani nel costato del WiFi, ed Emanuela. Ringrazio entrambi per la loro buona volontà, il loro entusiasmo, la passione e la serietà con i quali portano avanti - lottando contro le mille difficoltà - questo meraviglioso progetto - Tra silenzio e baccano - che, insieme a loro, mi auguro possa ancora crescere. Li ringrazio per la pazienza nel concedermi di fare le mie prove tecniche, per le mille telefonate e per l'amicizia che mi hanno concesso.

Rivolgo uno smisurato grazie alla Dott.ssa Guerra Lisi, ideatrice della Globalità dei Linguaggi (GdL), cui questo lavoro di tesi si ispira, la quale mi ha proposto - sulla fiducia - di presentare davanti al pubblico del 22esimo Convegno Nazionale della GdL a Roma il mio guanto e che mi ha onorato accettando di essere autrice della prefazione a questa tesi.

Ringrazio Antonella, l'abile e paziente sarta che ha trasformato un prototipo dalle scocciose apparenze in un guanto figo.

Mischiando un po' gli ambiti, tra tesi, musica, lavoro, passioni, affetti, famiglia, vado al pazzo, scatenato, professore senza il quale l'esitazione che mi contraddistingue in gran parte delle scelte della mia vita non si sarebbe immediatamente trasformata in un si a questa tesi; senza il quale Torino sarebbe stata una città più brutta in cui vivere; un maestro senza il quale la mia passione per la musica ed il mio interesse per la composizione non sarebbero stati uguali; al quale devo tanto per avermi consolato, consigliato, sostenuto e sopportato in tutti quei momenti in cui mi sentivo solo e perso: il Pruf. Gabriele Sedino. Grazie Gab.

Conosciuta poco prima di entrare al Politecnico, ho condiviso con lei i cinque anni a Torino e gliene ho combinate tante. Credo di averglielo già detto in qualche altra occasione, ma val la pena di ripeterlo in un documento ufficiale: grazie alla mia paziente amica Fernanda Scalisi, Fernandina, la mia prima amica, colei che ha segnato un passaggio importante della mia vita e che ha visto per prima, in me, qualcosa di buono. La ringrazio per avermi dato una famiglia a Torino e per essere stata e rimasta, sempre e comunque, nella mia vita.

Ringrazio quindi il resto della mia famiglia di Torino, partendo da Antonio, il mio coinquilino preferito che mi ha insegnato a vivere lontano da casa. Fausto, complice nella creazione della DPS e fraternamente disponibile. Elda, la mia meravigliosa vicina. E poi Francesca, Angela, Rosa Giulia e Antonio; Antonio 2.0 e tutti gli altri della DPS; Davide, anche per i generosi prestiti elettronici, Silvia, per le rilassanti e mai negate pause caffè insieme, Alessandro; i miei colleghi. Tante le difficoltà affrontate insieme, tante le gioie condivise, tante le esperienze vissute con voi. Grazie.

Non posso non ringraziare don Pippo, paternamente disponibile negli anni a Torino; zio Hani e Pino come punti di riferimento di zona San Paolo.

Ringrazio coloro che, da lontano, si sono comunque resi presenti. Tra loro, i miei parenti: le mie nonne: Carmela, Annina e Maria, le mie zie ed i miei cugini, per avermi sempre aspettato, per aver sempre trattenuto il mio cuore in Sicilia. Ringrazio mio figlioccio Ivan e la pancanona Veronica per l'affetto e la vicinanza dimostratimi - anche salendo a Torino per festeggiare qui con me questo traguardo, lo zio Franco, il mio patrox: Andrea, il Prof. Schillaci, Antonella Raffa, Sanpippo, Jessica, Alessio e Simone, Bruno e Rita, Armando.

Ringrazio la mia famiglia: i Claudi e coloro cui è dedicato questo lavoro, i miei genitori, la mia forza, la mia critica, il mio principale riferimento, la mia sicurezza. I miei genitori senza i quali, probabilmente, non sarebbe esistito questo lavoro perchè non sarebbe esistita l'esperienza a Torino, neanche il Politecnico... Mio padre, perchè non sarebbe esistito Sebastiano.

Rivolgo un ultimo acknowledgement, senza far nomi, a tutti coloro che non ci sono o non ci sono più, ma che ci sono stati e continuano ad esserci, dentro. L'amico Gab mi ha insegnato che quando si brinda si "batte" per tenerli presenti ed è in questi momenti di festa, o comunque di fine-round nel ring della vita, che più frequentemente capita di brindare e quindi di volgere loro il pensiero.

A volte mi piace strimpellare al pianoforte e vi confesso che, quando suono, m'illudo che la musica - l'arte che meglio di tutte afferma il passare del tempo, "scomparendo" dopo un po' - se non raggiunge più le mie orecchie e svanisce, è perchè ha raggiunto i luoghi in cui si trovano coloro che non ho fisicamente accanto.

Così, se Torino mi ha insegnato a stare lontano dalle persone che amo, la musica mi insegna come raggiungerle. Così mi trovo anch'io a superare - sempre con lei, "la nostra vera terapia" - le distanze, i silenzi, le assenze.

Grazie a tutti

Seba

Foreword

Università Popolare di MusicArTerapia
Via SS. Quattro 36/B - 00184 Roma
Tel. 331.8907129 Fax. 0670450084 - www.centrogdl.org
info@centrogdl.org - segreteria@centrogdl.org

English version below

L'obiettivo – o piuttosto la scommessa – della Globalità dei Linguaggi, considerata in questa tesi, è di mostrare quanto la più incorporea e autonoma delle arti – la Musica – sia in realtà sostanziata delle materie e leggi del cosmo, dei principi costitutivi ed evolutivi dei viventi, delle articolazioni, emozioni e sensazioni della nostra realtà psicosomatica umana. Tutto ciò è inscritto nella musicalità di ciascun essere umano, oltre le differenze, anche diagnostiche, come dimostrato nella parte esperienziale di questa tesi. In concreto rivisita l'esperienza musicale facendone emergere le implicazioni sinestetiche, simboliche, interdisciplinari, nell'ottica della Globalità dei Linguaggi (GdL): una disciplina della comunicazione e dell'espressione con tutti i linguaggi, ideata da me, da circa 50 anni, che è stata formulata progressivamente in una serie di pubblicazioni e dà il nome alle Scuole Triennali e ad un Master universitario. Nell' esposizione della stessa trovo affinità di pensiero con il Paradigma della GdL.

Secondo il modello della Competenza Musicale Comune della GdL (Gino Stefani), sono le Pratiche Sociali a volere e dover istituire in modo riconoscibile come "musicali" una serie di pratiche estético-ludiche - il canto, il concerto, la composizione, la danza, ecc... - e una serie di Tecniche "Musicali" ossia codificate in funzione di tali pratiche. É in questo senso che, nella nostra società, un evento sonoro può essere detto "Musica" invece che "rumore", e che si può parlare di "specifico musicale", e di "più o meno Musica" in un dato evento, come nel caso esposto nella tesi: Tra silenzio e Baccano.

Nell'ottica antropologica dei POTENZIALI UMANI (in accordo con l'estetica radicale di un John Cage), "Musica" è "il suono vissuto esteticamente", e "Musicalità" è "vivere esteticamente il suono"; suono è qualunque evento acustico; in quest'ottica la distinzione suono/rumore o l'equivalente Musica/rumore non ha senso dal versante dell'oggetto-evento, ma solo dal versante del soggetto umano. Ciò è conforme al detto comune: "non è bello ciò che è bello, ma è bello ciò che piace", che si può riformulare: "Musica" è suono gradevole, rumore lo sgradevole.

Nel vissuto le due facce dell'esperienza – l'oggetto e il soggetto – sono in relazione di reciprocità.

"Esteticamente" implica due dimensioni, l'autonomia e la sensorialità. In un senso, "estetico" è vissuto in e per se stesso, ovvero come esperienza autonoma, non in funzione di qualche utilità ulteriore, interna o esterna alla persona (produzione, comunicazione, terapia, ecc.). Così , "bello" - per noi sinonimo di "estetico" - si oppone a "utile". La Musica non è cercata perché serve a qualcosa, ma perché è bella, per chi la produce con il B-Glove.

In altro senso, "estetico" è "vissuto sensorialmente", conforme al senso etimologico (aisthesis: sensazione). In questo senso, dire che "bello" è "ciò che piace", è parlare di un'esperienza sensorialmente positiva, euforizzante, che aumenta il piacere di vivere e di sentire.

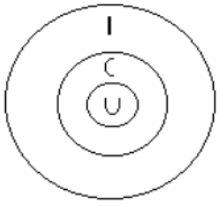
La GdL fonda e spiega quest'ottica con la sua ESTETICA PSICOFISIOLOGICA, mostrando in profondità le radici, gli inviluppi del musicale nel corpo: dalla sincronia-sintonia-sinfonia del feto nel grembo materno al flusso-sound vitale qualificato dalle emosazioni, alle modalità melodiche e ritmiche con cui si muovono le articolazioni; e poi dimostrando la continuità e lo sviluppo di questa costituzione protomusicale nella vita infantile e poi adulta, incluse le stereotipie sonore degli autistici e le sovversioni delle avanguardie.

Anche gli etnomusicologi "si sono resi conto che quanto più ci si avvicina all'individuazione degli universali in musica, tanto più i fenomeni con cui si ha a che fare sono radicati nella natura psicofisiologica dell'essere umano" (Baily).

"Come è musicale l'uomo?" La domanda, e più ancora le risposte, dell'etnomusicologo John Blacking (1973) erano sembrate rivoluzionarie per una cultura ancora fortemente eurocentrica. Ma le esigenze della nascente musicoterapia e poi i problemi dell' intercultura l'hanno progressivamente acclimatata e riformulata nel tema della Identità musicale.

Nella GdL questo tema viene articolato nel percorso di costruzione della Persona, dove la prima domanda è: "Chi sono io?"

Per rispondere, è utile anzitutto considerare che ogni Identità consta di più dimensioni o Identità. In sintesi: una dimensione o identità Universale (U), data a tutti gli uomini per natura; una Culturale (C), proveniente dal contesto storico e sociale; una Individuale (I), diversa per ogni storia personale. In questo modello la dimensione individuale include evidentemente quella culturale, ed entrambe l'universale: uno sviluppo di un avviluppo di potenziali profondi. In schema:



Rispetto all' uso del B-Glove, la domanda “Come è musicale l'uomo?” si può e si deve riformulare, nella GdL, in quella più generale: “Come è comunicativo ed espressivo l'essere umano?”. L'identità di homo musicus, la costituzione musicale universale dell'essere umano sarà uno sviluppo, una esplicitazione, una visione particolare dell'identità dell'homo sentiens, ludens, loquens.

Qui la GdL propone risposte articolate a diversi livelli; per ciascun livello c'è (almeno) un modello teorico ben formulato, ma che solo in parte è stato sviluppato nelle infinite applicazioni concrete, e che dunque sollecita la ricerca su una estetica psicofisiologica universale, musicale, che non si può perdere neanche nelle diversità patologiche più gravi, come si dimostra in questa tesi.

Il B-Glove di Sebastiano Franchina è di facile indossabilità, per cui è adeguato per soggetti poco collaboranti o anche spastici (di qualunque età anche alzheimer), e permette il "compiacimento nel produrre effetti" così importante per favorire la Comunicazione e l'Espressione nella contemporaneità dei Linguaggi: plastico-motorio-sonoro nelle sfumature propriocettive grafico-cromatiche. In particolare viene realizzata la musica spontanea di ciascuno secondo un "Progetto Persona nella GdL", che permette un dialogo MusicArTerapeutico secondo ritmi, melodie, glissando, funzionali allo sblocco bioenergetico del "corpo tripartito": Melodia, dalle ascelle in su, come bramosia; Sound, come emozionalità nel plesso solare; Ritmo, come scarica verso il basso, dai genitali agli arti inferiori. È importante sottolineare che la MANO, quindi il B-Glove può orientarsi verso l'alto (musica apollinea), o irradiando intorno al centro (musica panica), o scaricando verso il basso in modo catartico (musica dionisiaca).

Stefania Guerra Lisi, 2017
info@centrogdl.org, fb: Globalità dei Linguaggi-MusicArTerapia

The aim - or rather the bet - of the Globality of Languages, considered in this thesis, is to show how much the most incorporeal and autonomous of the arts - Music - consists of the substances and laws of the cosmos, of constitutive and evolutionary principles of the living, of the joints, emotions and sensations of our human psychosomatic reality. All this is inscribed in the musicality of each human being, apart from the differences, even diagnostics, as demonstrated in the experiential part of this thesis. In fact, the music experience has been revised by highlighting the sinesthetic, symbolic, interdisciplinary implications of the Global Language (GdL) perspective: a discipline of communication and expression with all the languages, conceived by me for about 50 years, which has been formulated progressively in a series of publications and gives the name to the three-years Schools and a Master's degree. In this thesis I find affinity with the GdL paradigm.

According to the model of GdL's Common Music Competence (Gino Stefani), the Social Practices are willing and need to recognize as "musicals" a series of esthetic-ludic practices - song, concert, composition, dance, etc ... - and a series of "Musical" Techniques that are coded according to such practices. It is in this sense that in our society, a sound event can be called "music" instead of "noise" and that we can talk about "music" and "more or less music" in a given event such as in the case mentioned in the thesis: Tra silenzio e baccano.

In the anthropological view of HUMAN POTENTIALS (in accordance with the radical aesthetics of a John Cage), "Music" is "aesthetically-lived sound", and "Musicality" is "aesthetically living the sound"; sound is any acoustic event; in this respect, the sound / noise distinction or the equivalent Music / Noise does not make sense from the side of the object-event, but only from the side of the human subject. This is in line with the common saying: "It is not beautiful what is beautiful, but it is beautiful what it likes", which can be rephrased: "Music" is pleasing sound, noise is an unpleasant sound.

The two faces of experience - the object and the subject - are in in a mutual relation.

"Aesthetically" involves two dimensions, autonomy and sensoryity. In a sense, "aesthetic" is lived in and for itself, or as an autonomous experience, not in function of any further usefulness, internal or external to the person (production, communication, therapy, etc.). So, "beautiful" - for us synonymous with "aesthetic" - it opposes "useful". Music is not sought because it serves something, but because it is beautiful for those who produce it with the B-Glove.

In other respects, "aesthetic" is "sensory lived", conforming to the etymological sense (aisthesis: sensation). In this sense, saying that "beautiful" is "what he likes" is talking of a sensitively positive, euphoric experience that increases the pleasure of living and feeling.

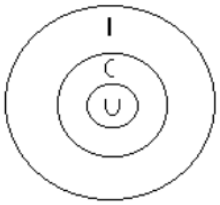
GdL sets and explains this view with its PSYCHOFISIOLOGICAL AESTHETICS, deepening the musical roots in the body: from synchrony-symphony-symphony of the fetus in the maternal womb to the vital flux-sound qualified by emos-actions, to melodic and rhythmic modes with which the joints move; and then demonstrating the continuity and development of this protomusical constitution in childhood and later life, including the sonor stereotypics of autistic people and avant-garde subversions.

Even ethnomusicologists "have realized that the closer they are to the identification of universals in music, the more the phenomena they are dealing with are rooted in the psychophysiological nature of the human being" (Baily).

"How is a man musical?" The question, and even more the answers of the ethno musicologist John Blacking (1973) seemed revolutionary for a still highly Eurocentric culture. But the demands of emerging music therapy and then the issues of interculturality have progressively acclimatized and reformulated into the theme of musical identity.

In the GdL this theme is articulated in the construction of the Person, where the first question is: "Who am I?"

To answer, it is useful first to consider that each Identity is of multiple dimensions or Identities. In summary: a Universal Dimension or Identity (U), given to all men by nature; a Cultural one (C), coming from the historical and social context; an Individual one (I), different for each personal story. In this model the individual dimension obviously includes the cultural one, and both the universal: a development of an envelopment of deep potentials. In scheme:



In relation to the use of the B-Glove, the question "How is a man musical?" can and should be reformulated, in the GdL, in the more general one: "How is the human being communicative and expressive?". The identity of homo musicus, the universal musical constitution of the human being, will be a development, an explication, a particular vision of the identity of homo sentiens, ludens, loquens.

Here the GdL proposes responses articulated to different levels; for each level there is (at least) a well-formulated theoretical model, only partially developed in the infinite number of concrete applications, and which urges research on a universal, musical psychophysiological aesthetic that can not be lost even in the bigger pathological diversities, as demonstrated in this thesis.

The B-Glove by Sebastiano Franchina is easy to wear, so it is suitable for non-cooperating or even spastic subjects (of any age even alzheimer's), and allows "feeling pleasure while producing effects" so important to promote communication and expression in the Contemporaneity of Languages: plastic-motor-sound in the chromatic proprioceptive nuances. In particular, the spontaneous music of each one is made according to a "Person Project in the GdL", which allows a MusicArTerapeutic dialogue according to rhythms, melodies, glissandos, functional to the bioenergetic unlock of the "tripartite body": Melody, from the armpits up, like craving; Sound, as emotionality in the solar plexus; Rhythm, as discharge down, from the genitals to the lower limbs. It is important to point out that the HAND, then the B-Glove can point upwards (Apollonian music), or radiating around the center, or cathartically discharging down (Dionysian music).

Stefania Guerra Lisi, 2017
info@centrogdl.org, fb: Globalità dei Linguaggi-MusicArTerapia

Contents

Introduction

If it is true that a thesis has to reflect the path of a student during his universitary experience, it should involve not only the capabilities the student acquired during his studies, but also his hobbies, his interests, his life.

In a world in which the engineer is ever more specialized, but where is requested to him to be extremely versatile, the thesis work of an engineer should be a tip in a sea seeming very far from his subject of study and, in some way, an opportunity for testing himself and for growing up in his transversal knowledges.

Animated by these considerations, which are not more than opinions of who is writing, it has been tried to find an argument requiring a bit of multidisciplinarity.

Working on a combination between the Computer Engineering and the Music, was the initial address of the author. The possibility to work on a combination between Computer Engineering, Music and disability gave to this experience, from the point of view of who is writing, an added value with respect to the already great importance of the thesis work: besides the degree, besides the thesis, besides the curriculum and the experiences in the Internet of Things field (of main interest for the author for his future career), they will have been collected the smiles of all those guys for which this work has been conducted, as well as life experiences that every man should do.

This thesis is part of a whole project of the Politecnico called Arduino Disability Orchestra (ADO). The main purpose of ADO is designing and prototyping electronic musical instruments for people with disabilities. The final (for the moment) product of this work is a MIDI glove able to produce music. It is connected through the WiFi to a PC which synthesizes the notes through the Ableton Live Software. It works in different modes, allowing to the performer to be part of an orchestra or to play a solo.

The glove has been designed for Samir (fantasy name), which gave it the name of Black Glove (because of the colour of the first prototype he played). Samir is a guy affected by an atypical form of Autism with mental retardation: as we will see later, despite he presents all the symptoms of Autism, Samir sometimes behaves differently from the majority of autistic people. He is part of "Tra silenzio e baccano", an ensembles of guys affected by cognitive and/or physical disabilities which directly use the ADO instruments.

In order to develop this thesis, a direct contact with these guys, and in particular with Samir, has been required. If in principle seemed strange, if in principle a sense of inutility was present thinking at a similar work for people in those conditions, after the first rehearsals it has been possible to focus the importance and the utility that a similar project can have for these people. If in a first time their disabilities seemed to make everyone unable to do something, after, their smiles and their wish of playing make the same people determined to address the same scope: having fun together.

In the first chapters of this thesis, is briefly described the MIDI protocol, some needed software and some concepts of 3D mechanics and avionics useful in the discussion of the work. Moreover, the motivations in support of making disabled people able to play music are discussed, with a general overview of the music-therapy history, its application to autistic people and a focus on some music-therapy methods involving also the movement.

In particular, the main aspects related to the Globality of languages (GdL) discipline (by Stefania Guerra Lisi, who wrote the foreword for this thesis) are discussed. In fact, the combination between movement expressiveness and musical feedback seems to completely match with the GdL paradigm. The B-Glove has been presented during the 22th Globality of languages National Convention in Rome.

Then, the real context is presented, starting by the history of "Tra silenzio e baccano", together with a general overview of its components and a particular focus on Samir and his pathology.

A chapter is dedicated to the analysis of the requirements that a new instrument, designed for disability, should have. In particular, it is discussed taking into account the components of "Tra silenzio e baccano" and their limits.

A technical presentation of the B-Glove is then proposed, focusing on hardware and software designs, showing some results of its usage and evaluating its performance in terms of cost, speed, usability, etc... A chapter is dedicated to the Instruments Manager software, a program developed for managing the B-Glove (and every instrument using the same protocol) from the PC.

After an analysis of the utility the B-Glove can have for Samir, who inaugurated it during a concert, some ideas for the future of "Tra silenzio e baccano" are proposed, together with some improvements for the B-Glove.

Music and MIDI

Music: levels of representation and digital production

Music is, in general, a sequence of notes that can be represented in different ways and at different levels [1]. Depending on the representation, different parameters are considered and used in order to transmit the information relative to the musical signal.

At the physical level, music is an audio signal, i.e. a waveform characterized and described in terms of amplitude and frequency. It would result difficult, for a musician, to reproduce a piece of music by looking at a graph representing its sonogram. Thus, what is to consider is the scope for which the music is represented.

The path of a piece of music includes a number of levels of representation that can be summarized as follow:

- the composer thinks at the piece of music from a structural point of view (structural);
- he writes it in symbols (symbolic);
- the performer converts the symbols in actions on the instrument (operative);
- the sound produced by the instrument propagates till reaching the ears of the listener and exciting its nervous system (physical);
- the listener perceives the music in terms of volume, duration, tone, etc... (perceptive).
- the listener can convert what he has listened in symbols (symbolic)

The composer, the performer, the instrument and the listener can be substituted by electronic devices or software on a PC. Like in the reality, they have to agree on the same musical representation. Then, a piece of music acquired through the microphone can be converted in that musical representation; a musical representation of a track can be converted in audio.

The device or software in charge of translating a sequence of notes in audio is called synthesizer. The synthesis operation generates electric signals that are converted to audio using amplifiers, loudspeakers and/or headphones.

Depending on the complexity of the synthesizer, the sequence of notes can be converted in order to resemble different musical instruments. The quality of the sound and its psico-acoustic properties directly depend on the quality of the synthesizer in the same way in which the performers and the instruments affect the sound during an acoustical performance.

The one in charge of storing and (re)generating musical events, instead, is the sequencer. It can be an electronic device or a software and it is able to acquire data from an input peripheral, editing them, storing them and send them to the synthesizer. If the data are sent in the same time they are collected (like in an electronic keyboard), you have a real-time sequencing.

The synthesizer can be embedded in the sequencer or not.

MIDI: Musical Instrument Digital Interface

The MIDI protocol is used as common standard music technology protocol for connecting and allowing the communication between products coming from different companies, like smart-phones, computers and musical instruments [2].

It defines the communication protocol and the digital and electrical interfaces for the devices and allows to establish a communication on 16 channels for each single link. Each channel can be associated to a musical instrument with a specific timbre and so no more than 16 timbres can be simultaneously synthesized [1].

MIDI Messages

The MIDI protocol works through messages that can be divided in system and channel messages: while the first category includes messages for configuration, synchronization and timing between multiple MIDI devices, all the musical event messages belong to the second one.

A MIDI message is composed by one or more words of 10 bits (8 + 2 for delimitation). The first word is called status byte, it has the MSb equal to 1 and it is used for indicating the kind of information contained in the following words, called data bytes. Each byte can be divided in two nibbles of 4 bits: the first indicating the most significant ones, the second indicating the least significant ones.

The first nibble of the status byte of a system message is always 1111. Thus, with the second nibble you can identify up to 16 system messages but only 11 are practically used. The structure of the status byte of a channel message is instead 1tttcccc, where ttt represents the kind of channel message and cccc the number of channel to which it is directed.

Both system and channel messages are divided into subcategories. The three most important and used messages belong to the Channel Voice Message category. They are called Note On, Note Off and Control Change and they are composed by two data bytes each.

Before going to the messages, it results to be useful to analyse a parameter which is used in the first two: the velocity.

The velocity

It can be easily confused with the volume of a note, but it is something different.

The MIDI protocol has been structured considering the piano keyboard. When pressing or releasing a key on the piano, the speed with which it is moved down or up affects the dynamics of the whole sound.

The relation between this speed, called velocity, and the dynamics of the sound is different than the effect of a static volume regulation. The velocity is something that can be set only at the beginning and at the end of a note, while the volume can be modified even during the period the note is reproducing.

It is immediate to observe that velocity affects the freedom in modifying the volume, but not vice versa. For example, playing a note with 0 velocity means being not able to listen anything, whatever it is the volume.

Another consideration can be done in relation to the kind of instrument you are playing (or synthesizing). In some instruments, like a piano, the velocity directly affects the duration of the note, while in others, like an organ, this is not true. If you press and leave pressed a key of a piano, the note starts with a certain velocity, it has a dynamics, then it ends. If you press a key of an organ and you leave it pressed, the note never ends.

For the first kind of instruments, setting the volume can become useless at a certain point and you have to play another note if you want to modify the dynamics; for the second ones, each time a note is played, you can only act on the volume for affecting the dynamics.

Although in the real world it is difficult to find yourself in the second situation, when synthesizing it is very common to find instruments with a sound which never ends.

Note On

The Note On message is characterized by a status byte equal to 1001cccc and two data bytes indicating the note and the velocity respectively.

Each data byte has a structure equal to 0xxxxxxx and so the meaningful value can go from 0 to 127 (2^7 - 1): you can represent 128 notes and 128 values of velocity. The central C (Do in Italian notation) is assigned to the value 60.

Note Off

The Note Off message is characterized by a status byte equal to 1000cccc and two data bytes indicating the note and the velocity respectively. It can be considered like a Note On message with velocity equal to 0. For this motivation, the sequences of Note On and Note Off messages are substituted with a single Note On status byte with couples of note/velocity.

Control Change

Through the so called controllers, it is possible to manage the expressiveness of a MIDI device. An example of controller is the piano pedal or the breath control for a flute.

The status byte of a Control Change message has the following structure: 1011cccc. The first data byte indicates the kind of control you want to act on. The meaning of the second data byte depends on the kind of control and can assume:

- continue values, from 0 to 127, like for controller 7: main volume
- discrete values, like for controller 64: pedal, which can be inserted (from 0 to 63) or not (from 64 to 127).

Software

As said before, the sequencer and the synthesizer can be substituted either by electronic devices or software on a PC. The software used for this thesis will be now briefly described.

Ableton Live

Running on Windows and MAC-OS, it is one of the most common and most used software DAWs, that is software able to record, edit and produce audio files, sound effects, songs, music.

Generally, with software DAW is indicated, not only the software, but the combination of a computer, an audio interface (or sound card), a digital audio editor software and one or more input devices.

Differently than the others DAWs, Ableton is designed for live performances. It comes with two views: the arrangement one, very similar to any kind of sequencer interface, and the session one (see the figure), allowing to arrange different scenes and to easily group instruments and effects.

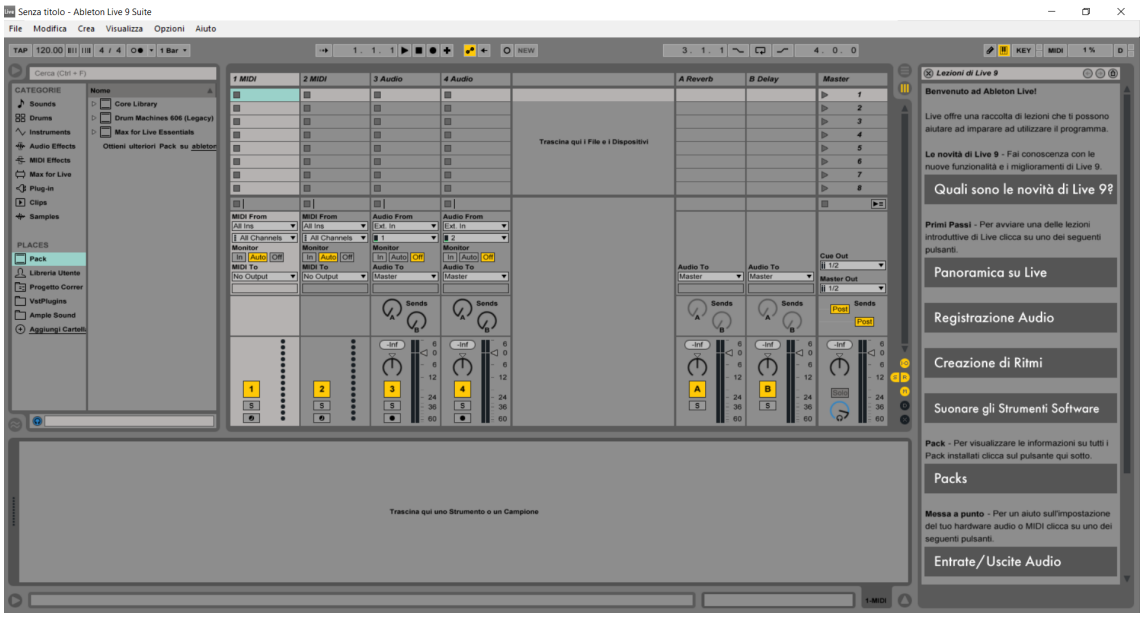


Figure 2.1: Ableton Live 9 Suite interface

Live can be controlled through external MIDI devices/surfaces: a determined MIDI Control message can be associated (mapped) to an Ableton function. This is possible

previously checking, in the MIDI section of the Ableton settings, the remote control for the corresponding input device, like showed in the figure.

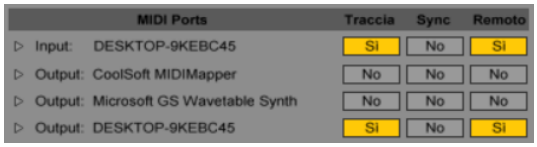


Figure 2.2: MIDI peripherals settings on Ableton Live Suite 9

rtpMIDI

rtpMIDI is a virtual-MIDI driver, allowing Windows systems to be connected to MIDI devices through the network. Thanks to this driver, it is possible to exploit the LAN for exchanging MIDI messages.

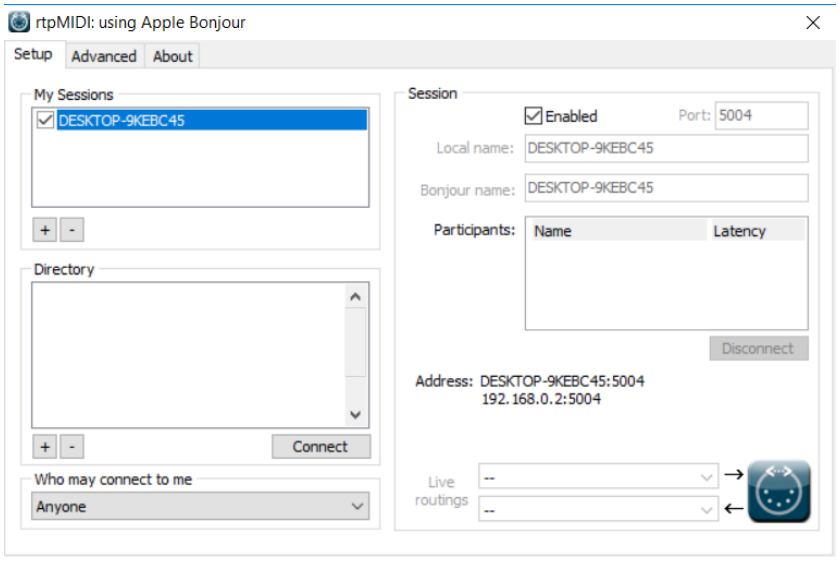


Figure 2.3: rtpMIDI interface

While a similar driver, called network-MIDI, is included in all the Apple operating systems, on Windows it needs to be installed. A PC mounting rtpMIDI and an Apple device can communicate, without problems, through the network: rtpMIDI and network-MIDI are fully compatible.

If using Wi-Fi connections, this driver allows to avoid long and heavy MIDI cables for interconnecting different DAWs between them or to other MIDI devices.

MIDI-OX

MIDI-OX is a very powerful MIDI utility for Windows systems. It is used both for monitoring purposes and for allowing the communication between external devices and the PC through SysEx messages [3].

It allows to have a look of input and output MIDI streams and to manage a mapping between them. Moreover, thanks to an included driver called MIDI Yoke, it is possible to generate almost each configuration of mapping between any Windows MIDI application input and any Windows MIDI application output.

In fact, MIDI Yoke provides up to 16 MIDI input ports and 16 MIDI output ports, each of them allowing up to 4 connections, which can be used for performing every kind of configuration [4].

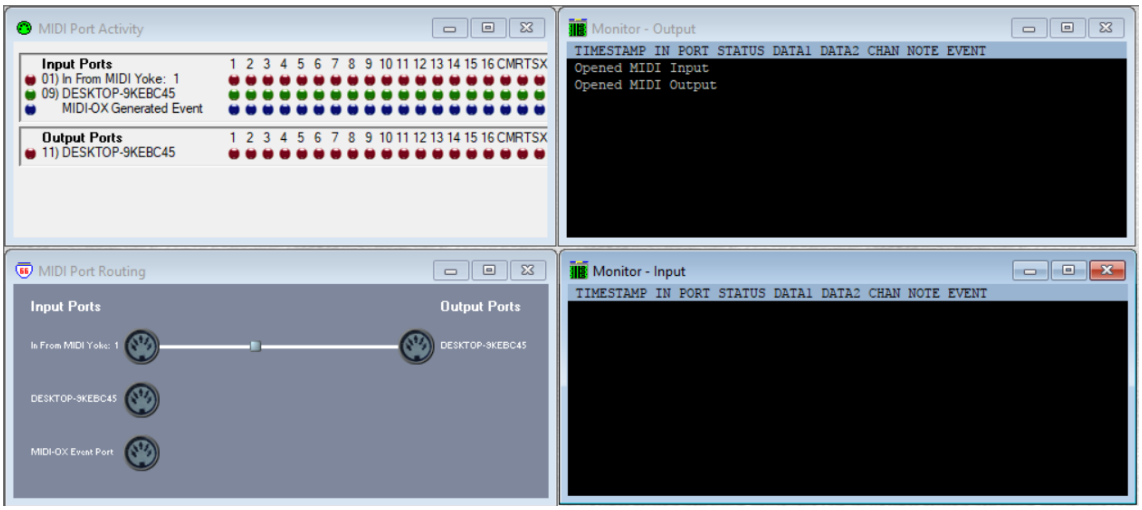


Figure 2.4: Part of the MIDI-OX interface

Using MIDI-OX it is also possible to generate MIDI messages either through the computer keyboard or the embedded control panel.

Finally, MIDI-OX owns a COM interface which can be used and accessed by the languages supporting COMs (for example Visual Basic).

Max for Live and MIDI Note Mapper

Max for Live is an Ableton-integrated version of Max. Max is a visual programming language for music and multimedia, used for instruments, audio effects, synthesizers and every kind of device you need in Ableton Live. In fact, Max is completely integrated in Ableton [5].

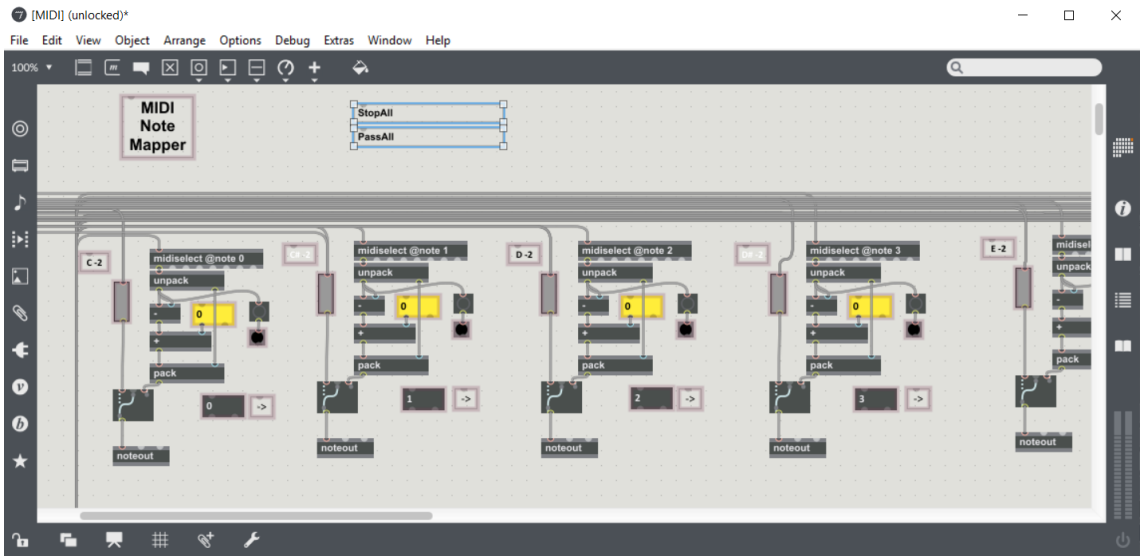


Figure 2.5: Max interface with an example

It can be used for building controllers in order to change the way in which external devices interact with Ableton. You can also add video and lights to your Ableton live set [6].

It comes with a collection of instruments, effects and tools and you can download more from the Max community where the users share their own Max devices.

MIDI Note Mapper It is a device for Max, using which you can filter incoming MIDI notes and remap them to different ones [7].

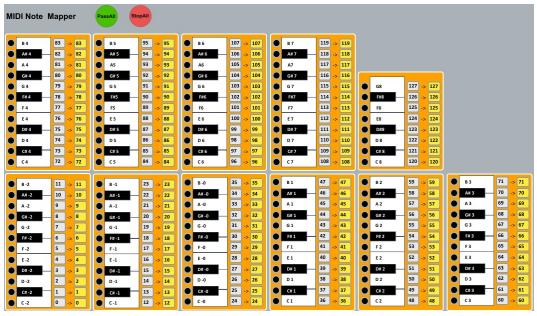


Figure 2.6: Midi Note Mapper

Music for therapy

The Music Therapy: history and definition

Music has been surely used as medicine in the past two ad a half millennia, but maybe even before [8]. What has changed in time has been the healer, that is the person in charge of using music for healing a patient [9]:

- in ancient societies, shamans and tribal musicians used particular rhythmic patterns for helping people possessed by evil spirits. The members of the patient's family and someone of the community contributed to produce the rhythms. These were slow for decreasing the temperature and expelling fever spirits, quick for increasing it and expelling rheumatoid spirits.
- in the Ancient Greece music was used by priests for restoring the harmony between body and soul. Music was considered able to influence entire populations: Plato wrote "Justice is to the soul as health is to the body. Through music, the soul learns harmony and rhythm and even a disposition to justice."
- even in the Christian era, priests used music, but this time as a message of hope, for creating the Heaven in the heart and make the patient feeling loved by God. Some instruments like trumpets, flutes etc... - commonly used for pagan music, were avoided.
- starting from the Renaissance and even in the Romantic Era, music was not more used by spiritual entities, but by physicians and musicians for helping patients in solving their emotional or physical problems. A theory called "of temperaments and body juices" was at the bases of these practices in which the music was used for affecting breathing, pulse, blood pressure and metabolic rate. Here, the patient became active part of the orchestra.
- in the first years of the 17th century, the interest for therapeutic properties of music rapidly grew and in 1940s music therapy started to be considered as a clinical profession, with a proper training of specialized music therapists [10].

What is in charge of a each music therapist is exploiting musical experiences and the relationships deriving from them in order to help the clients to promote their health [11].

The World Federation of Music Therapy (WFMT), bringing together the most important music therapy organizations, defines the Music Therapy as "the professional use of music and its elements as an intervention in medical, educational, and everyday environments with individuals, groups, families, or communities who seek to optimize their quality of life and improve their physical, social, communicative, emotional, intellectual, and spiritual health and well-being." [12].

Music for Autism

Autism

Autism has been categorized as a neurodevelopmental disorder, that is disturbs that manifest themselves during the first 5 years of a child and which affect learning-abilities, memory, self-control and emotions and which can cause intellectual disabilities (also known as mental retardation) [13], communication and language problems, etc...

Neurodevelopmental disturbs are categorized as follow:

- Intellectual disabilities, or mental retardation
- Autism spectrum disorders, including Asperger, Kanner, Autism
- Motor disorders, including coordination disorder and tics
- Traumatic brain injury
- Communication, speech and language disorders
- genetic disorders, including Down syndrome, hyperactivity disorder and schizophrenia
- disorders due to neurotoxicants caused by alcohols, heavy metals, drugs, etc..

Autism Spectrum Disorders (ASDs) are characterized by [13]:

- deficits in social communication and interaction;
- repetitive and restrictive behaviours and activities;
- difficulties in establishing, maintaining and understanding relationships.

Symptoms change with the development and can be masked through compensative mechanisms. Further, they can manifest themselves only when the social interests overcomes the limited capacities.

When diagnosing an autism spectrum disorder, it has to be specified if it is associated to other mental or behavioural neurodevelopmental disturbs; in fact, an ASD is often associated to a mental retardation and sensory perceptual and behavioural problems [14].

Music Therapy for children affected by Autism

Different medical, behavioural, developmental and educational therapies have been tested in order to try to contrast the disturbs caused by ASDs, whose results are discussed and analysed in [15] and [16].

It has been recognized that the quality of the interventions, medical and/or behavioural, depends on the specific areas, on the specific disturbs and that they are often not so effective.

In 1943, trying to give a first definition of Autism, Kenner noticed that many autistic children enjoy music very much [17]. Although children affected by ADSs manifest lack of interest for the outside world, the majority of them are attracted by music and some present a particular predisposition for the perception and discrimination of sounds [14]. Using music started to be an option for helping autistic people.

Music and movement: Orff Method

The Orff method for music therapy takes the name from its inventor, Gertrude Orff, and has been classified by Bruscia as a Developmental Music Therapy [18]. The method intends the music with the same sense intended by the great composer Carl Orff in his Orff-Schulwerk music-education approach: "musiké", a combination of words, sounds and movements. It is based on three main elements.

The first one is the improvisation: not necessarily "free improvisation"; the structure can be provided by music itself with sounds and silences. The use of improvisation provides creative stimuli to the client.

The second is the instrumentation: both musical instruments (like strings, keyboards and percussions) and non musical ones (like balls, hand puppets, etc..) can be used for play music in the sense of "musiké".

The last regards the possibility for the therapists of meeting the needs of the user using the multisensory properties of music: the sound can be felt in different ways.

Globality of languages

In the last 50 years, Stefania Guerra Lisi developed a discipline, called Globality of languages (or With all languages) [19], whose main purposes are the research, the education, the rehabilitation, the therapy in order to improve or develop communicational and expressive capacities.

The paradigm of this discipline is exploiting every kind of tool, meaning and language as possible - starting from the most used for human communication: the body language - for developing a cure for a particular person, not a generalized one. Expressing our-self, in any ways, is always therapeutic.

Then, from the paradigm derive a series of models and theories which are put in practice through specific methods, paths and operating styles. In every case, the methods observe the globality of a person: while some of them analyse the single person in her daily behaviours and in her globality, others exploit the group energy, realizing the integration, promoting an egalitarian communication between all the diversities. Fundamental in the Globality of languages is the therapy of smiles.

Figure 3.7: 22th National Convention poster

During the last National Convention of the Globality of languages (the 22th from the 1996) called "Comunicare per vivere" (Communicating for a living), a space has been given to the presentation of the B-Glove. After a brief improvisation with an accordionist, M° S. Panu, the B-Glove has been used by a volunteer from the crowd.

Some theory about 3D mechanics and flight dynamics

Quaternions

Quaternions have been defined by W. R. Hamilton in 1843 as the quotient of two vectors in the three-dimensional space [20], they extends the complex numbers and they are very used for calculations involving three-dimensional rotations in 3D mechanichs. They are represented in form a + bi + cj + dk, where a, b, c and d are real numbers and i, j and k are the quaternion units.

Yaw, Pitch and Roll

Yaw, pitch and roll are the coordinates of rotation of an aircraft. Considering this coordinate system, the z-axis positive direction is toward the Earth.

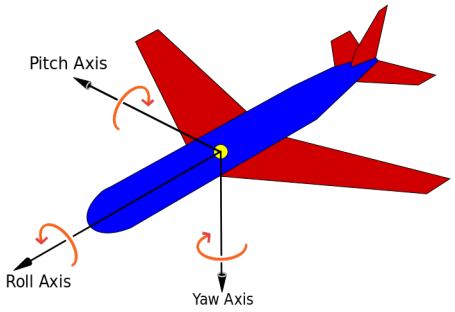


Figure 4.8: Max interface with an example

The Yaw is the rotation around the so called vertical axis and is computed exploiting the Earth magnetic North (or the true North). The Pitch is the angle between the x-axis of the aircraft and the Earth ground plane. The Roll is the angle between the y-axis of the aircraft and the Earth ground plane. Deriving Yaw, Pitch and Roll coordinates from a quaternion-based representation of the orientation is quite immediate.

Madgwick Filter

The Sebastian Magdwick's orientation filter is a sensor fusion algorithm which combines acceleration, rotation rate and magnetic moments in order to obtain the orientation of an object in a quaternion-based representation [21].

Compared with the conventional Kalman filter, it is able to reach comparable performance with a lower computational power and at a higher sampling rate. Moreover, it has been developed for being used with Inertial Measurement Units (IMU) [22].

The context

Tra silenzio e baccano

"Tra silenzio è baccano" (Between silence and noise) is a project born in 2013 from Riccardo Turino and Emanuela Badoglio. By chance, they have noticed that Matteo, Emanuela's son, responded to musical stimuli produced by Riccardo with its guitar.

Matteo was almost completely paralyzed: even now, he is able only to move its hands and its head; he is affected by physical and cognitive disabilities and he is not able to speak. When Riccardo played the guitar, Matteo replied with some sounds from its mouth and vibrating its hands. Moreover, he was visibly happy, he smiled.

Riccardo and Emanuela decided that Matteo should be made able to receive feedbacks from his movements, he should be made able to understand that those movements were producing something and to participate to the song he was listening with its movements. Therefore, with the help and the suggestion of Daniele, they positioned a chime close to Matteo in a way he could play it when vibrating his hands. Then, they would like to amplify the sound produced by Matteo and they have placed five piezos in the tubes of the chime, realizing an electronic instrument.

Figure 5.9: Matteo, playing the chime during the last concert of Tra silenzio e baccano

Today, Matteo plays the chime, synthesized with harp sounds, during the concerts of “Tra silenzio e baccano”. In fact, through the “Vivere” association – grouping families having kids with disabilities, Riccardo and Emanuela composed an orchestra in which each guy plays an instrument and/or sings.

In a typical performance of "Tra silenzio e baccano", Riccardo plays his guitar and drags the guys into the music. Some of them are volunteers, contributing to build a musical mood on which the solos of the guys affected by disability find foundation.

In the last concert the orchestra was composed by: a chime (Matteo), 5 iPads, 2 MIDI Keyboards, some digital and acoustic percussions, 2 bongos, a bass, violin, a flute, a harmonic, the Cavarin, the B-Glove (Samir) and the Riccardo’s guitar.

On the iPads, a particular app is installed: it allows to manage different iPads from a single master one, in order to set a common music scale and to set the instruments to play for each track. The Cavarin and, now, the B-Glove have been realized from scratch.

Samir

Samir is a 22 years old guy and is affected by an atypical Autistic Psychosis and by a mental retardation. When he was a child, he didn’t speak and he was, sometimes, very restless, repeating some problematic behaviours like running back and forth, jumping on the spot, sniffing food and people and touching the walls.

Today, Samir is a joyous person which loves to make contact with other people. Repetitive behaviours and hobby-horses are still present in his life, but in a less problematic way: he needs to program in advance how to spend his days and his weeks, marking on the calendar every thing he’s going to do; each time he talks with someone, he always brings up the same, often illogic, arguments.

Samir keeps his room in order and organizes his space, is autonomous in personal cleansing, showering and beating. He prepares his training bag and brings it near to the exit door in advance. He does some domestic jobs, such as washing dishes, ordering spaces, hanging the laundry and setting up the table.

He ended up the high school even if its school level is that of a 6 years old child. On the other hand, he loves and is also very good at playing the piano and painting. He draws with the bureau, without ever cancelling, and his works are very beautiful. Moreover, he has fun dancing.

He is enrolled to a sportive association for guys affected by disabilities and he plays basket, football and swim. Moreover, he sings in a choir of disabled people.

During the meetings and the rehearsals of "Tra silenzio e baccano" it has been noticed that Samir loves perform in front of a public. This makes it difficult to stop him when repeats his patterns at the piano or when sings at the microphone: you have to give him a precise time after which he will can perform again.

Arduino Disability Orchestra

Arduino Disability Orchestra (ADO) is a project born in collaboration with “Tra silenzio e baccano”, grouping engineers, makers, designers, artists in order to design and prototyping musical instruments for people with disabilities.

Cavarin

It is the first instrument prototyped by ADO, used in all the last concerts of "Tra silenzio e baccano". It consists of an Arduino Leonardo connected to an Ultrasonic Sensor which is in charge of measuring the distance between the instrument and the player.

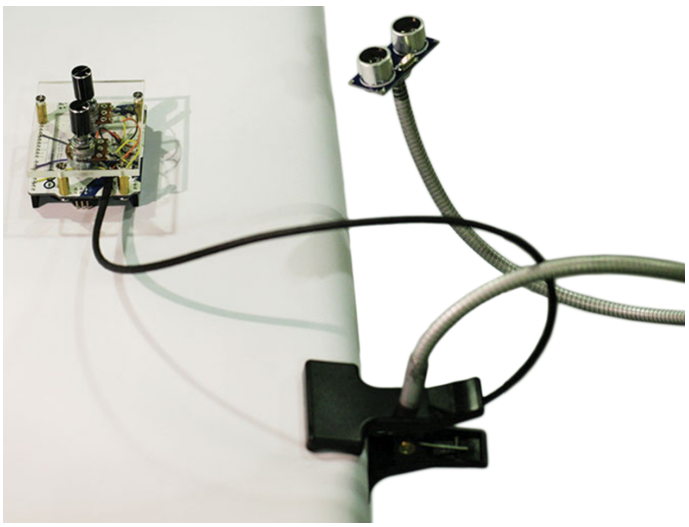


Figure 5.10: The Cavarin

Depending on the distance, a different note is synthesized: a different couple of MIDI messages (note-off of the previous note plus note-on of the new note) is serially sent to the PC that synthesizes and plays it through Ableton. When the distance is very low or very high, a note-off message is sent, producing the silence.

It has been used in two versions:

- with the hand
- with the wheelchair, with larger intervals for the notes

Requirements

Starting point and requirements

The first thing to try to avoid when speaking about people with disabilities is, as is known, any kind of limit, any kind of barrier. For allowing people affected by physical disabilities to enter in somewhere, architectural barriers have to be avoided. In the same way, allowing playing music can introduce a series of limits and of difficulties, not only physical, that should be bypassed.

Music, as said, can be defined as a sequence of sounds of determined frequencies: the notes. Whether notes are played simultaneously or consecutively, you can speak about musical consonance and dissonance, regarding the pleasantness that those sounds have for the ear. Despite in the XX century the usage of the dissonance is intensely practised, in musical composition are very used music scales, directly dependent on consonance.

Having the possibility of setting a music scale on a musical instrument, means that if who is playing this instrument is performing a solo over a base adopting the same scale, he will be always in consonance with it. This is fundamental if the consonance is requested and if the guy in charge to play the instrument is not able to remember or to understand which notes he have to play and in which order.

This is the road on which "Tra silenzio e baccano" is moving: using instruments (iPads or microcontroller-based devices) allowing to set a musical scale, in order to guarantee a consonant final music.

Playing a musical instruments means not only playing a sequence of notes, but playing it with a certain dynamics of volume. Something that in the instruments used by "Tra silenzio e baccano" was missing, was the possibility of giving a velocity to the MIDI notes: both the iPads and the Cavarin have not a mechanism allowing dynamics.

Another possible limit of the Cavarin could be the constraint imposed by the sensor: the hand (or the wheelchair) has to move within the conical detection space of the sensor, otherwise no sound is produced. Thus, the guy in charge of playing the Cavarin is anchored to the place where it is mounted and this is not always convenient.

Regarding the architectural barriers, we can include and consider as such wires and cables. In "Tra silenzio e baccano" performances, there still be a huge amount of cables for the powering of the instruments and for their amplification.

The previous considerations conducted to a list of functional and non-functional requirements that a new instrument should have:

- Possibility of setting a music scale (F)
- Controllable volume/velocity (F)
- Avoid anchorages (NF)
- Wireless (NF)
  - Embedded powering -> Battery
  - Amplification (?)

Addressing the requirements: basic idea for a new instrument

The main goal was exploiting some capacities of the guys and making them able to produce music. The majority of them had the possibility of controlling the movements of one or both the hands, some of them were able to speak and to walk, Matteo only vibrated his hands and rotated his head.

The main idea was exploiting the speed of a movement in order to give a dynamics to a sound effect or to a note. This would have allowed us to address the first requirement.

Thinking at the exploitable movements, the options were represented by the rotatory movement of Matteo's head for triggering and piloting the dynamics of one or more (through its inclination) sound effects, or by the more controllable movements of the hands of any guy.

This second option has been chosen, deciding to realize a glove making able the performer to select the note through the inclination of the hand and to control the dynamics through the speed of the movement.

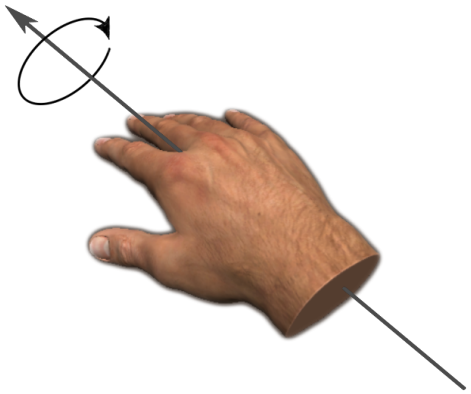


Figure 6.11: Rotation of the hand for note selection (AllHandModels with LeapMotion)

The new requirements became:

- Sensor unit able to measure both the speed of the hand in order to control volume/velocity and the inclination of the hand in order to select the note;
- Processing unit for managing the sensors;
- Radio Unit for wireless communicating with the PC;
- Battery for powering the system;
- Possibility to set a music scale (?);
- Amplification (?)

Thus, the requirement was a sensor node (as we will see, a wireless sensor node). While the selection of the kind of sensor has been rather easy, the selection of the kind of processing (and communication) unit has required to deal with different considerations and to take different decisions which are discussed in the following paragraphs.

Sensing subsystem

As said, the selection of the sensing subsystem has been quite easy. Instead of using different sensors for measuring the speed and the inclination, it has been decided to use a single IMU (Inertia Measurement Unit) and to exploit, not the speed, but the acceleration of the hand for the dynamics. Technical details are leaved to the Hardware and Software design sections.

Processing (and communication) subsystem

The two requirements affecting the selection of the processing subsystem typology were:

- Amplification
- Music scale selection

As written in a previous chapter, two main modules are involved in music production: the sequencer and the synthesizer. Avoiding cables from the hand to the amplifiers and/or loudspeakers, required the application of one of the following two options:

- placing the amplification circuit on the hand;
- sending data wireless to an amplification system

The first option was not feasible due to the context in which the instrument has to be used: mounting a loudspeaker on the hand means using a small loudspeaker, with a limited sound power; this wouldn't be audible during a live performance with other instruments.

This bore to the necessity of designing a wireless sensor node, able to transmit data. However, the solution of a second problem was now required: the kind of data to send.

Three solutions could be taken into account, depending on the placement of the sequencing and synthesizing operations:

1. sending the raw data of the sensor unit to a PC, deriving from them the MIDI sequence, synthesizing it with a software synthesizer and sending it to the amplification;
    - medium weight of data to be sent to the PC
    - no processing subsystem is required: only the radio unit for transmitting data to the PC
2. producing the MIDI sequence on the glove, sending it to the PC, synthesizing it and sending it to the amplification;
    - light data to be sent to the PC
    - medium effort for the processing subsystem
3. producing the MIDI sequence on the glove, synthesizing it on the glove, sending the audio data to the PC or to the amplification
    - heavy data to be sent to the PC
    - high effort for the processing subsystem: a DSP (Digital Signal Processor) could have been necessary.

On the other hand, the setting of the music scale have to be done between the sequencing and the synthesizing operation. For the solution 3, it should have been done on the glove, by sending some commands from the PC. For the solutions 1 and 2, instead, it was possible to decide if setting the scale on the glove or on the PC. This second possibility could be exploited by sending always the same MIDI notes to the PC and transposing them depending on the wanted music scale.

It has been decided to follow the solution 2 and to set the music scale on the PC. This allowed:

- using a microcontroller instead of a DSP
  - lower cost
  - simpler to program
- sending light data (MIDI messages) instead of heavy ones (raw sensor data or audio data)
  - less problems in wireless transmission and lower latencies
  - distributed elaboration
  - physical separation between sequencing and synthesizing operation: this will allow to reuse the synthesizer configuration on the PC for other similar instruments

The final Requirement Document

The target was composed by

- an electronic MIDI musical instrument having the appearances of a glove
- a software or a group of software on the PC handling the MIDI messages received by the glove and synthesizing them

Basic functionalities:

- data about inclination and acceleration of the hand have to be collected by a sensor
- raw sensor data have to be converted in MIDI messages
  - the rotation space (Roll) of the hand have to be divided into a predefined number of intervals, each of them corresponding to a note. Each time the hand has an inclination in a determined interval, a Note On message has to be produced (after a appropriate Note Off message for the previous note).

It has been decided that an appropriate rotatory space for the right hand is the one showed in the next figure: a range of about 220° to be divided into a predefined number of intervals.

– the velocity has to be computed depending on the acceleration of the hand: every cycle of the real-time software has to detect the acceleration and to send a Control-7 (Main volume) Change message with the new velocity as control value; the velocity of the Note On message has to be equal to the last Control Change message's one

- MIDI messages have to be sent to the PC wirelessly
- MIDI messages have to be handled by the PC, mapped to the correct music scale and synthesized with a determined instrument

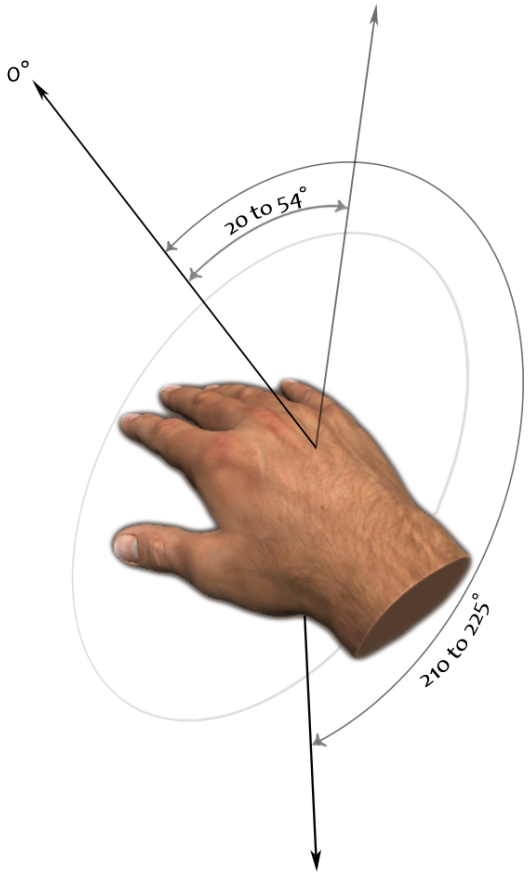


Figure 6.12: Rotation space of the hand for note selection (AllHandModels with Leap M.)

Other functionalities:

- the number of notes has to be variable: the PC software has to allow to send in a convenient manner the number of notes to the instrument, depending on the music track that is to be playing or decided on the fly by the user; on the other side, the instrument has to set itself the intervals in an appropriate way
- the PC software has to give the possibility of monitoring the incoming MIDI messages and of starting and stopping the loop of the real-time software running on the instrument

Constraints:

- the components have to be small and light (for wearability)
- real-time embedded system: low latencies (necessary if dealing with music)

Market analysis

Before starting thinking how to realize the glove, of course, a market analysis has been done, evaluating similar products in terms of characteristics and prices. The number of developed MIDI-gloves is pretty high, but the available solutions are quite different than the one discussed above.

The majority of them exploits the contact between the fingers or the movements of the hand for modulating a track and not for producing music from scratch. Some of them are students projects and are not yet on the market.

Only some of them are wireless, exploiting the Bluetooth or the WiFi. The following is the list of the most interesting MIDI gloves.

- Remidi T8 [23]
- Aura [24]: not for sale
- Tornado A1 [25]
- Glove midi controller [26]
- Midi Drum Glove [27]
- Gesture-based MIDI Glove [28]
- Mimu Glove [29]

The B-Glove

Hardware Design

Sensing part: the IMU

An IMU (Inertia Measurement Unit) is a device that combines different kinds of sensors, mainly accelerometers, gyroscopes and sometimes magnetometers. It is able to provide to the user information about its proper acceleration[1], its angular rate and sometimes about the magnetic field from which is surrounded.

The Adafruit LSM9DS0 IMU combines a 3-axis accelerometer, a 3-axis gyroscope, a 3-axis magnetometer and a temperature sensor. The first three sensors can be combined in order to obtain information about the 3-dimension acceleration and orientation.

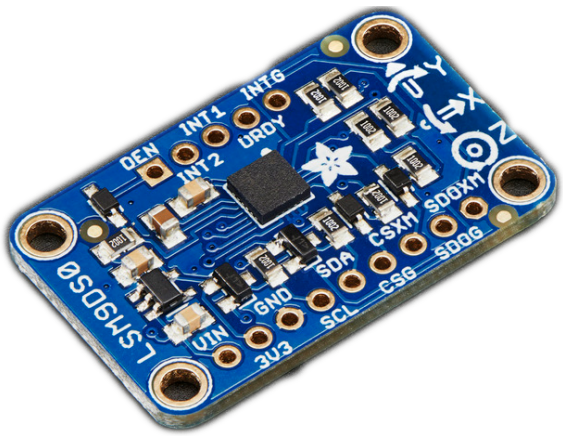


Figure 7.13: Adafruit LSM9DS0 IMU

[1]Proper acceleration (from Wikipedia): in relativity theory, proper acceleration is the physical acceleration (i.e., measurable acceleration as by an accelerometer) experienced by an object. It is thus acceleration relative to a free-fall, or inertial, observer who is momentarily at rest relative to the object being measured. Gravitation therefore does not cause proper acceleration, since gravity acts upon the inertial observer that any proper acceleration must depart from (accelerate from). A corollary is that all inertial observers always have a proper acceleration of zero.

The major advantages of this device were:

- an acceptable cost: around 30 €
- small dimensions and negligible weight: 33 x 20 x 2 mm on 2.3g (good for wearability)
- SPI and I2C interfaces
- easiness in using with Arduino-based platforms
  - available libraries
  - possibility of using both 3.3 V and 5 V for powering it, depending on the selected microcontroller

The microcontroller: Adafruit Feather Huzzah ESP8266 (AFH)

After a first test phase performed on an Arduino Due, during which the functionalities have been better defined (see the Software design section), the Adafruit Feather Huzzah ESP8266 microcontroller has been selected.

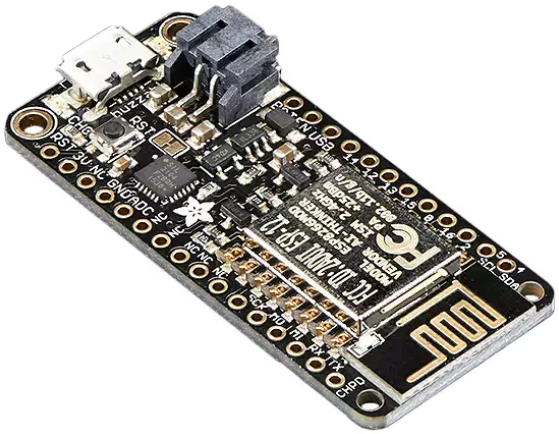


Figure 7.14: Adafruit Feather Huzzah ESP8266

This microcontroller can be programmed with the Arduino IDE, using the Arduino compiler and so the Arduino programming language. This allowed to re-use the code already written for the Arduino Due, changing the part relative to the transmission of the MIDI to the PC (from Serial to WiFi) and adding what was missing.

Moreover, it integrates the ESP8266, that is a WiFi module. This has avoided the necessity of providing a Radio Unit to be connected to the system for the communication with the PC. Thanks to the available libraries for the WiFi connection and for the MIDI over WiFi transmission, a lot of work has been simplified using this component.

Power

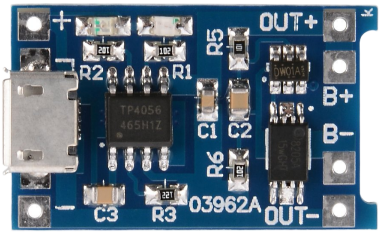
The power supplying is a substantial problem for every wireless sensor node; in particular, for the ones that have to be wearable and so as smaller and lighter as possible. The problem has been solved with a Li-Po Battery pack having:

- nominal capacity: 1500 mAh
- output voltage: 3.7 V
- maximum output power: 5.6 W

Figure 7.15: Li-Po 1500 mAh Battery

The average current requested by the system is around 100 mA, allowing a life time of about 15 hours with the battery completely charged.

Although it was possible, through the USB port of the AFH, to recharge the battery with a 100 mA current, a charger module from XCSOURCE has been added in order to decrease the charging time. This last embeds the TP4056 LI-Ion battery charger whose output current depends on the resistance value read on the R_{PROG} pin, like showed in the table taken from the data-sheet. The XCSOURCE module embeds a R_{PROG} of about 1.22 kΩ, resulting in an output current of 1 A and so in a charging time of about one hour and half. This module also allows to prevent complete discharge of the battery and so to increase its life.



R_{PROG} (k)	I_{BAT} (mA)
10	130
...	...
1.33	900
1.2	1000

Figure 7.16: XCSOURCE Battery Charger Module and TP4056 output current table

The glove: design

The final glove is not more totally black, like the prototype whose Samir gave the name (Black Glove). It is composed by two gloves: an internal one for separating the skin from the electronics and an external (blue or red) one for covering the electronics.

Figure 7.17: Internal (right) and external (left) gloves

Both the LSM9DS0 IMU and the AFH microcontroller are able to work up to a temperature of 85°C (look at the data-sheets): covering the electronics does not represent a problem.

Putting all together: the schematic

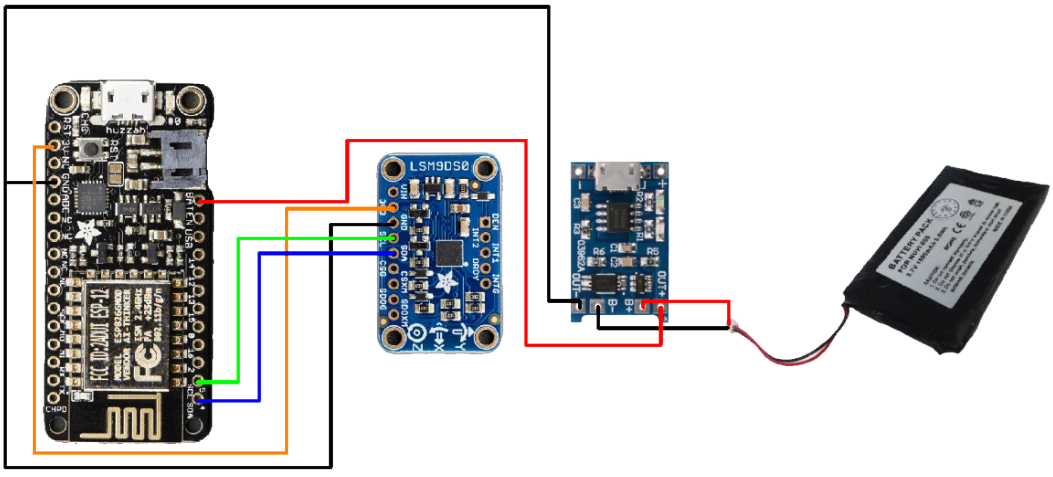


Figure 7.18: The schematic, made in Autocad

Putting all together: placement on the glove

The placement has been done by taking into account both functional and comfort aspects.

The IMU could be placed both on the palm and on the back of the hand, provided it was the most parallel to it as possible. It has been decided to place it on the back, like also the other components, in order to allow the hand to comfortably close. The final position is the one showed in the following figure, together with the final glove (even if not yet sewn):

Figure 7.19: Placement of the components on the internal glove an covering

For a correct initialization of the IMU, it has to be parallel to the floor when it is powered on. In order to give to the user the possibility to check this condition, a small bubble level has been placed on the external glove. Moreover, a button switch allows connecting and disconnecting the battery from the rest of the circuit. The components have been sewn on the internal glove as well as the external glove. Some holes have been created for the switch and for the micro-USB plugs.

Software Design and Testing

The software running on the microcontroller of the B-Glove is nothing more than a sequencer: it takes some values from the IMU and derives a sequence of Note On, Note Off and Control Change MIDI messages.

The first version of the software has been developed on an Arduino Due board, which was connected to the IMU through the I2C protocol exploiting the already predisposed ports of both the Arduino and the IMU. Moreover, it was connected via USB to the PC to which it transmitted the MIDI messages. Thus, the first version of the glove was not wireless.

We prefer to describe the final version, in order to immediately have the final overview. However, both the codes, the Arduino Due version and the AFH one, are attached to this thesis in the Appendix.

As described in the Hardware section, the final B-Glove software runs on the Adafruit Feather Huzzah microcontroller embedding the ESP8266 WiFi module. As said before, the AFH can be programmed using the Arduino IDE and the Arduino compiler, so with the Arduino code.

The structure of an Arduino program is composed by two main parts: the setup and the loop. Both the parts corresponds to two functions: the setup() one is executed only one time at the beginning, then, the loop() function is executed consecutively until the system is powered off.

The main operations executed by the software, can be mapped to the two functions as follow:

- setup()
  - Connection to the WiFi network
  - Connection to rtpMIDI
  - Initialization of the IMU
- loop()
  - Listening of incoming MIDI messages (used for starting and stopping the sequencing and for setting the number of notes/chords/notes per chord)
  - Checking of the PC availability for receiving MIDI
  - Reading IMU data
  - Data filtering
  - Derivation of avionic parameters
  - Computation of the musical note(s)/chord(s) depending on the interval
  - Delivering of the right MIDI message(s): couples of Note On and Note Off messages or a Control Change message

Needed libraries

Before showing the different part of the code, the following commented piece of code shows the libraries that have been included in order to use the IMU sensor and the ESP8266 on the microcontroller. Moreover, the AppleMIDI library is used for communicating MIDI on the network, with the same protocol used and understood by rtpMIDI (Windows) and network-MIDI (Apple).

```
1 //For I2C connection between AFH and IMU
2 #include <Wire.h>
3
4 //For IMU
5 #include <Adafruit_Sensor.h>
6 #include <SFE_LSM9DS0.h>
7 #include "MPU6050_6Axis_MotionApps20.h"
8
9 //For Wifi and MIDI
10 #include <ESP8266WiFi.h>
11 #include <ESP8266mDNS.h>
12 #include <AppleMidi.h>
```

Listing 7.1: Needed libraries

WiFi and rtpMIDI connections (Setup)

They connection to the WiFi is performed pre-storing in two variables the credentials and by using the WiFi.begin(ssid,password) function. Further, it is checked if the connection has taken place.

Regarding the WiFi connection, we had to consider the way in wich it was better to proceed: changing the router would mean changing the credentials. There were three possibilities:

- setting fixed credentials and leave to the user to configure the network for the glove;
- sending the credentials through the Serial Port (using the Instruments Manager software) each time a new router is used: they would be stored on the EEPROM and so they wouldn't be lost when the instrument is powered-off;
- embedding a micro-SD reader, storing on a micro-SD the credentials (through the Instruments Manager software) and read them from there.

The main obstacle of the second option is the limited number of writing/erasing cycles allowed by the EEPROM (some thousands). It is not so low, but still limited. Moreover, reading the IP which has been assigned to the instrument by the DHCP requires opening the Serial Port and reading it or access the router or the ARP table.

The third solution avoids using the cable and the EEPROM. However, if it is more simple w.r.t. the second, the IP address can be read only by checking the clients connected to the network.

The first option seems to be inconvenient because of the needing of having a network with fixed credentials. However, considering the possibility of each PC of using its network board as an access point, this solution becomes the easiest to apply. It does not require re-programming the glove and the IP address can be found on the same program setting up the access point (see the next chapter for more information). The credentials have been set like follows.

```
const char* ssid     = "glove";
const char* password = "blackglove";
```

Listing 7.2: Credentials for WiFi connection

The rtpMIDI connection is performed throug the AppleMIDI.begin("test") function, taking as parameter the name of the session which will appear on rtpMIDI when the connection will be established.

```
/* CONNECTION TO THE WIFI */
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
}

/* CONNECTION TO RTPMIDI */
// Create a session and wait for a remote host to connect to us
AppleMIDI.begin("test");

AppleMIDI.OnConnected(OnAppleMidiConnected);
AppleMIDI.OnDisconnected(OnAppleMidiDisconnected);

AppleMIDI.OnReceiveNoteOn(OnAppleMidiNoteOn);
AppleMIDI.OnReceiveNoteOff(OnAppleMidiNoteOff);
```

Listing 7.3: WiFi and rtpMIDI connection code

The last four instructions represent the definition of 4 functions in charge of handling four events: their name ar pretty meaningful and you can see the body of the function in the complete code in the Appendix. However, the code of the OnAppleMidiNoteOn function is presented also in the MIDI-management code explanation, in the following, because it is in charge of handling the commands received by the PC which, as we will see, sends them as MIDI Note On messages.

IMU initialization (Setup)

As already mentioned in the Hardware section, during the initialization (happening at the powering on of the circuit), the IMU board has to be parallel with respect to the floor. This is fundamental in order to have correct readings and so a correct production of MIDI messages.

The initialization consists in setting, for each sensor of the IMU, the wanted scale and the wanted update rate. Here, has been decided to have the maximum possible resolution and so to set the lowest ranges possible, correspondent to:

- 4G (gravity acceleration) for the accelerometer: 2G was low for hand acceleration
- 245 degrees per second (dps) for the Gyroscope
- 2 Gauss for the Magnetometer

The output data rates (ODR) are:

- 50 Hz for the accelerometer
- 190 Hz for the Gyroscope
- 12.5 Hz for the Magnetometer

Moreover, they have been set the Anti-Aliasing filter rate of the Accelerometer to 50 Hz and the Bandwidth for the Gyroscope to 12.5 Hz. Almost the totality of them are default values. Some other combination of values have been tested, but an absence of improvements has convinced to leave the following ones.

```
1  /* DEVICE INITIALIZATION */
2
3  //LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);
4  uint32_t status = dof.begin();
5
6  delay(1000);
7  dof.setAccelScale(dof.A_SCALE_4G);
8  dof.setGyroScale(dof.G_SCALE_245DPS);
9  dof.setMagScale(dof.M_SCALE_2GS);
10 dof.setAccelODR(dof.A_ODR_200);
11 dof.setAccelABW(dof.A_ABW_50);
12 dof.setGyroODR(dof.G_ODR_190_BW_125);
13 dof.setMagODR(dof.M_ODR_125);
14 delay(1000);
15 dof.calLSM9DS0(gbias, abias);
```

Listing 7.4: IMU initialization code

The dof.calLSM9DS0(gbias, abias) collects a number of samples from the Gyroscope and the Accelerometer, averages and scales them to g's and dps and computes the biases errors to subtract to the following data.

The loop: condition for looping and outside messages polling

As we will see in the MIDI-management code explanation, the "poweredon" boolean variable is set by the OnAppleMidiNoteOn function when the MIDI Note On message corresponding to the start command of the glove is received by the PC.

The "isConnected" boolean variable, instead, is set by the OnAppleMidiConnected function, each time a connection with rtpMIDI is established, and unset bu the OnAppleMidiDisconnected function.

```
1 void loop()
2 {
3   //Listen to incoming notes
4   AppleMIDI.run();
5
6   if (isConnected && poweredon){
7   ....
8   }
9 }
```

Listing 7.5: The loop: condition for looping and polling for outside messages

Thus, data from the IMU are collected if the glove is connected to the PC and the PC is ready for receiving MIDI data. Before entering the real body of the loop (inside the if), a polling for incoming MIDI messages is performed with the AppleMIDI.run() function.

When the rtpMIDI session disconnects, the OnAppleMidiDisconnected function calls a reset function in a way of retrying the connection.

Data collection and filtering

The data collection from the IMU sensors is performed through some functions of the SFE_LSM9DS0 library. When read, the data are corrected subtracting the biases errors and then are routed to specific variables.

At this point, the update time is computed in order to be used for the filtering and this last is applied, in order to have a certain continuity in data.

The last part of the code is in charge of producing the meaningful data that will be used for the MIDI messages fabrication. In particular, they are computed the three parameters of avionics: yaw, roll and pitch; and it is derived the 3D acceleration.

```
dof.readGyro();  // Read raw gyro data

// Convert to degrees per seconds, remove gyro biases
gx = dof.calcGyro(dof.gx) - gbias[0];
gy = dof.calcGyro(dof.gy) - gbias[1];
gz = dof.calcGyro(dof.gz) - gbias[2];

dof.readAccel();  // Read raw accelerometer data

// Convert to g's, remove accelerometer biases
ax = dof.calcAccel(dof.ax) - abias[0];
ay = dof.calcAccel(dof.ay) - abias[1];
az = dof.calcAccel(dof.az) - abias[2];

dof.readMag();  // Read raw magnetometer data

// Convert to Gauss and correct for calibration
mx = dof.calcMag(dof.mx);
my = dof.calcMag(dof.my);
mz = dof.calcMag(dof.mz);

Now = micros();
// set integration time, since last filter update
deltat = ((Now - lastUpdate)/1000000.0f);
lastUpdate = Now;
MadgwickQuaternionUpdate(ax, ay, az, gx*PI/180.0f, gy*PI/180.0f,
      gz*PI/180.0f, mx, my, mz);

//Computation of the avionic parameters
yaw   = atan2(2.0f * (q[1] * q[2] + q[0] * q[3]), q[0] * q[0]
     + q[1] * q[1] - q[2] * q[2] - q[3] * q[3]);
pitch = -asin(2.0f * (q[1] * q[3] - q[0] * q[2]));
roll  = atan2(2.0f * (q[0] * q[1] + q[2] * q[3]), q[0] * q[0]
     - q[1] * q[1] - q[2] * q[2] + q[3] * q[3]);
pitch *= 180.0f / PI;
yaw   *= 180.0f / PI;
// TO MODIFY FOR TURIN
yaw   -= 2.2;
roll  *= 180.0f / PI;

//Acceleration: components and computation of the total one
curraccx=ax*SENSORS_GRAVITY_STANDARD;
curraccy=ay*SENSORS_GRAVITY_STANDARD;
curraccz=az*SENSORS_GRAVITY_STANDARD;
curracc= sqrt(curraccx*curraccx+curraccy*curraccy
     +curraccz*curraccz);
```

Listing 7.6: Data collection and filtering

MIDI management

In this section we will see, not only the way in which the notes and the velocities are derived from the roll and the acceleration, but also the MIDI communication protocol allowing to control the glove from the PC.

The B-Glove MIDI management regards both incoming and outcoming MIDI messages: the first ones are used for delivering some setting commands to the instrument from the PC; the second ones are used for the sequencing and also for acknowledging commands received by the PC. The sequencing operation is performed through three functions delivering Note On, Note Off and Control change messages respectively. They are:

- AppleMIDI.noteOn(note, velocity, channel)
- AppleMIDI.noteOff(note, velocity, channel)
- AppleMIDI.controlChange(controlNumber, controlValue, channel)

All the possible kinds of commands deliverable by the PC to the instrument using the Instruments Manager software, are specific MIDI Note On messages whose velocity is meaningful for understanding the kind of command. They are handled through the On-AppleMidiNoteOn function.

Before analysing incoming and outcoming MIDI messages, it is useful to explain the general idea about the behaviour. Extending the requirements, it has been programmed the instrument in order to deliver not only single notes, but also chords. Till now, it has been programmed only for chords up to 4 note each, but it results to be not difficult to extend the number of notes-per-chord.

Once the instrument has received by the PC the number of notes/chords it has to play in the whole rotation space of the hand, this last is divided into a correspondent number of intervals. When the roll falls in a determined interval, a specific note/chord is delivered to the PC.

The first interval (in a clockwise sort) always corresponds to the central C (Do in Italian notation), that is a MIDI height of 60. The following intervals are mapped to the following notes : 61, 62...up to 75. The maximum number of notes playable in a whole rotation has been decided to be 16.

When the instrument is set for playing a chord, each interval does not correspond to a single note, but to a series of 2, 3, or 4 notes having a MIDI height difference of 16, like showed in the following figure.

Interval	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Base-note	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
Notes	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91
	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107
	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123

Notes per chord (1 to 4)

Figure 7.20: Mapping roll-intervals to notes/chords

Even when a chord has to be played (more than one note at a time), only the base-note is computed and stored, depending on the roll. The others are eventually computed on the fly when the MIDI Note On messages are sent (see later). The base-note is stored into a variable called "notetoplay" at the end of each loop cycle. Before changing the "notetoplay" content, it is copied into a variable called "oldnote". The comparison between the two allows to understand if the angular interval of inclination of the hand has changed or not. Both "notetoplay" and "oldnote" are initialized to 0.

We are now ready for seeing how the incoming messages are handled, that is analysing the code of the OnAppleMidiNoteOn function:

```
1  void OnAppleMidiNoteOn(byte channel, byte note, byte velocity) {
2    if (velocity > 0 && velocity <= 16){//Changing number of notes/chords
3      numOfNotes=velocity;  prevelabroll=0;
4      for (int i=0; i<preinchord; i++){
5        AppleMIDI.noteOff(oldnote+i*16, 1, 1);
6        delay(20);
7      }
8      oldnote=0; notetoplay=0;
9      AppleMIDI.controlChange(9,velocity,1);
10   }
11   else if (velocity == 88){//Powering-on the sequencer
12     poweredon=true;
13     AppleMIDI.controlChange(9,88,1);
14   }
15   else if (velocity == 89){//Powering-off the sequencer
16     poweredon=false;
17     for (int i=0; i<preinchord; i++){
18       AppleMIDI.noteOff(oldnote+i*16, 1, 1);
19       delay(20);
20     }
21     prevelabroll=0;  oldnote=0;  notetoplay=0;
22   velocity=0;   prevvelocity=0;
23     AppleMIDI.controlChange(9,89,1);
24   }
25   else if (velocity > 89 && velocity < 94) {//Changing notes-per-chord
26     inchord=velocity-89;
27     AppleMIDI.controlChange(9,velocity,1);
28   }
29 }
```

Listing 7.7: OnAppleMidiNoteOn function code

It is pretty easy to understand from the code that the possible incoming messages are 22 and, as said, they are MIDI Note On messages on the channel 1 (starting counting from 0) distinguishable looking at the velocity. They can be grouped in:

- Setting poweredon: velocity = 88
- Unsetting poweredon: velocity = 89
- Setting the number of notes: velocity = 1 to 16
- Setting the number of notes-per-chord: velocity = 90 to 93

We can see that for each received command, an acknowledge is sent through a Control-9 Change message. Moreover, when is requested to change the number of notes-per-chord, a specific variable called "inchord" is set, while when it is requested to change the number of notes/chords playable with a full rotation, a variable called "numOfNotes" is set.

The delay of 20 or 30 milliseconds between two MIDI messages is used for avoiding losses. As we will see later, it does not imply any considerable latency problems.

Moving to the sequencing, in order to derive the base-note depending on the angular interval of inclination where the hand is, as said, the precomputed roll is exploited. In particular, depending on the total number of notes which has to be played in the whole rotation space of the hand, two parameters are defined:

- zero position: the angular distance between the yaw axis of the hand
- interval amplitude

Sixteen values (the total number of notes can go from 1 to 16) for the zero position and sixteen for the intervals amplitude are stored into two 16-element vectors, called zeroPosition and angle respectively and indexed through the numOfNotes variable set in the OnAppleMidiNoteOn function:

```
1 int zeroPosition[16]={54, 54, 36, 25, 20, 54, 48,
2         40.5, 36, 32.5, 50, 45, 42.5, 56, 52.5, 49};
3 int angle[16]={216, 108, 72, 50, 40, 36, 32, 27,
4         24, 21, 20, 18, 17, 16, 15, 14};
```

Listing 7.8: Declaration of zeroPosition and angle vectors

Varying the zero position allows to have a more intuitive interaction with the glove: the zero position is changed in order to always have a "central interval", parallel to the floor, moving in which you maintain the note. The following examples should clarify eventual doubts:

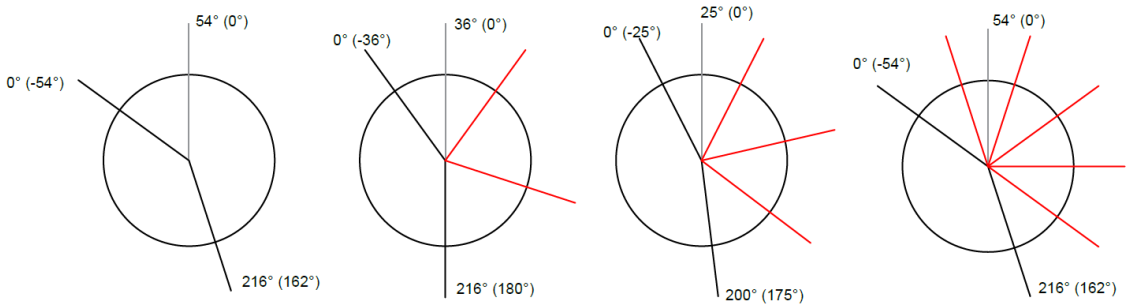


Figure 7.21: Examples of intervals placement depending on the total number of notes

In order to compute the note, the following operations are executed:

- the zero position is summed to the computed roll;
- it is checked if the hand has an admitted inclination (between the zero position and the end of the last interval)
- it is checked if the hand hasn't yet leaved the previous interval: on this check, an hysteresis of 2.5° is applied in order to avoid continual switches;
- in the case the hand is in a new interval, the new note is computed dividing the elaborated roll (roll + zero position) by the angle of an interval and summing 60 (the MIDI value corresponding to the central C, or Do in Italian notation).

```
1  double elabroll;
2  elabroll=roll+zeroPosition[numOfNotes];
3
4  if (elabroll < numOfNotes*angle[numOfNotes] && elabroll > 0){
5
6    if (elabroll < prevelabroll*angle[numOfNotes]-2.5
7              || elabroll > (1+prevelabroll)*angle[numOfNotes]+2.5){
8
9      elabroll=elabroll/angle[numOfNotes];
10     elabroll=floor(elabroll);
11     prevelabroll=elabroll;
12     notetoplay=60+elabroll;
13   }
14
15 }
```

Listing 7.9: Computation of the note to play depending on the hand inclination and on the total number of notes

It is clear from the algorithm that the notes which can be generated always correspond to MIDI heights from 60 up to a maximum of 75 (C4 to D#5), like desired: they will be mapped to a musical scale before being synthesized, as we will see in the following chapter.

Every time a new note is derived from the hand inclination, there can be two cases:

- the new note is equal to the previous one (notetoplay = oldnote)
- the new note is different than the previous one

In the second case, the velocity is computed, as we will see in a while, and a Control Change message is sent .In the first case, instead, a couple (or some couples if a chord has to be played) of a Note Off and a Note On MIDI messages is sent. The velocity of the Note On MIDI message is the last computed one.

The computation of the velocity is the most critical part of the MIDI management on the B-Glove. The way in which the velocity is computed affects the dynamics of the sound produced by the glove, the perception of a real-time influence of the user on the sound and so the pleasure of playing it.

Finding a good way for computing the velocity - as we said - from the acceleration, has been the highest effort-requiring operation because of the different attempts of approaching the problem and the different correction parameters that, along the way, it has been necessary to add for deriving the best correspondence between movement and sound perception.

The first idea was to derive the velocity, not from the acceleration, but from the speed of the hand. However, after different tests and attempts of integrating the acceleration, this approach has been abandoned: the non-negligible errors caused by the discrete integration were not acceptable.

The approach that has been follow and that is now used for computing the velocity is its derivation from the acceleration. In particular, the total acceleration, the one resulting by the quadratic sum of the three components. In this way, any movements in any directions can act on the velocity and you can move the hand in the most convenient and comfortable way.

It is to consider that the IMU always senses the gravity acceleration that therefore has to be subtracted to the resultant of the accelerations. From now on, we will refer to the acceleration to which the gravity component has been subtracted with the name of "corrected acceleration".

Another consideration affecting the selection of the best approach for computing the velocity regarded the instruments which it was desirable to use for synthesizing the sequence of MIDI messages produced by the B-Glove. In particular, we had to distinguish

between instruments for which the concept of velocity is equivalent to the volume (like an organ) and instruments for which the initial velocity affects the entire evolution of the sound (like for a piano).

The modulation of the velocity after the first one sent with the Note On message, make sense only for an instrument for which the volume variation has an impact on the sound. Considering a piano, after a while you pressed a key, the sound is so soft that a variation of volume does not produce effect. The following image should clarify any doubts:

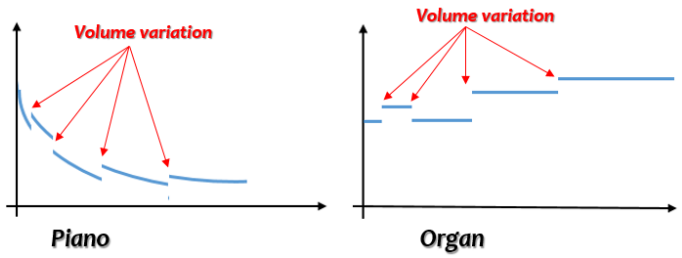


Figure 7.22: Instruments distinction for velocity computation

If you would like to synthesize the sequence with an instrument like a piano, you should intercept the maximum speed of a movement and use it for computing the velocity of a single Note On message.

Due to the difficulty of computing the speed and due to the difficulty of intercepting the "peak" of a movement, it has been decided to consider only the second group of instruments and to compute the velocity for altering the volume in time.

A first idea was to compute the velocity in an incremental way, following the steps below:

- The velocity is initialized at 0
- Each time the velocity has to be computed, the corrected acceleration is evaluated:
  - if the corrected acceleration is greater than 1 (if it is considerable), its integer part is added to the previous velocity value;
  - otherwise, the previous velocity is decremented by 1.

The code for implementing these steps is listed below. Moreover, a figure shows the results of a test, performed on this version of the B-Glove, during which the acceleration, the velocity, the note and the inclination (roll) have been monitored. It has been obtained connecting the glove to the PC using an USB cable and programming it in order to give in output on the Serial port the mentioned parameters. Then, they have been read by a Processing script (whose code is in the Appendix) and have been drawn in the 4 graphs. In all the cases, the instrument is set for playing 6 notes.

```
if (floor(curracc-G) >=1){
  if (volumeincreasing+floor(curracc-G) < 128 ) {
    volumeincreasing=volumeincreasing+floor(curracc-G);
  }
  else{
    volumeincreasing=127;
  }
  last=false;
}
else{
  if (volumeincreasing - 1 > 0) {
  volumeincreasing = volumeincreasing - 1;
  last=false;
  }
  else{
  volumeincreasing=0;
  }
}
prevvelocity=floor(velocity);
velocity=volumeincreasing;

if (prevvelocity != floor(velocity)){
  AppleMIDI.controlChange(7, floor(velocity), 1);
}
```

Listing 7.10: Version 1 of velocity computation

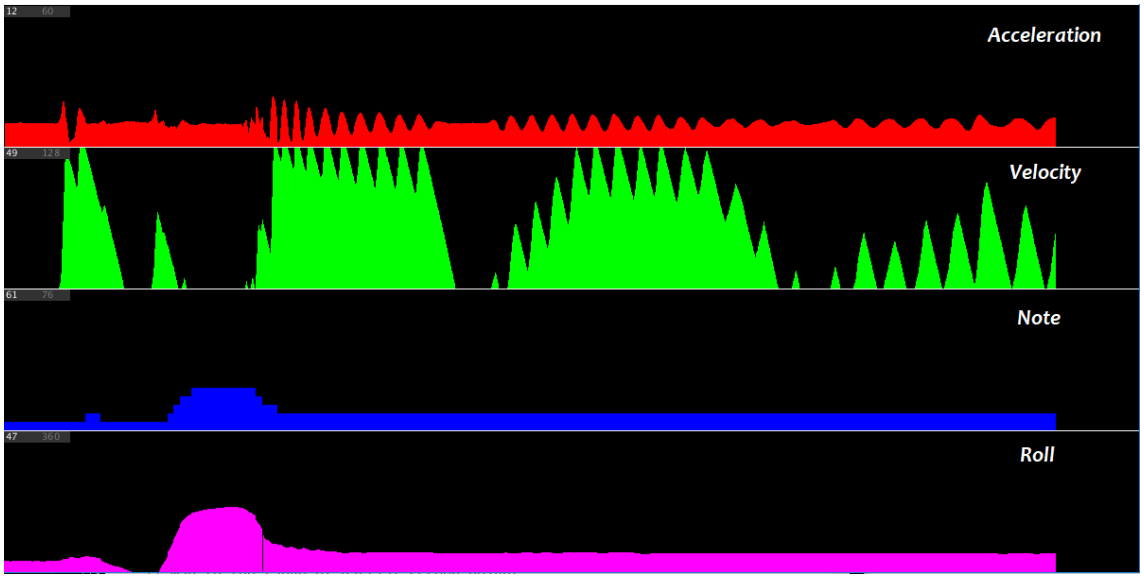


Figure 7.23: Test of the velocity-computation algorithm 1

This was, more or less, the code running on the first B-Glove prototype used by Samir during the concert of "Tra silenzio e baccano" in Poirino (TO). However, this very simple approach suffered of three main problems, which are pretty evident in the previous figure:

- a very high vibrato-effect
- difficult to play and to maintain a note at a low volume
- not so controllable velocity: to high for a medium acceleration

Both the first two problems were due to the fact that, each time the hand was not accelerating, the velocity was decreased by 1. It is to consider that, if you move the hand up and down (the only way for maintaining the note volume), each time it reaches the maximum and the minimum position, the acceleration is very low (even null) and so the velocity is decremented by 1. The idea was to decrease the velocity in a more complex way:

- decreasing it of a value always less than a factor A
- scaling the factor A considering the previous velocity: if lower velocity, then higher A, then slower decreasing; otherwise lower A and faster decreasing.
- Different tests bring to a value of 0.6 for the factor A.

The third problem, instead, was due to the complete proportionality between the acceleration and the velocity incrementing value: if, for example, we got a resultant acceleration equal to $2g = 19.6$, the velocity was incremented by 9. The idea for solving it was scaling the resultant acceleration with a correction factor. After different tests, 0.75 has been set: the previous example conducts now to a velocity incrementation of 4.

Below, you can find the results of the test of this second version and its code.

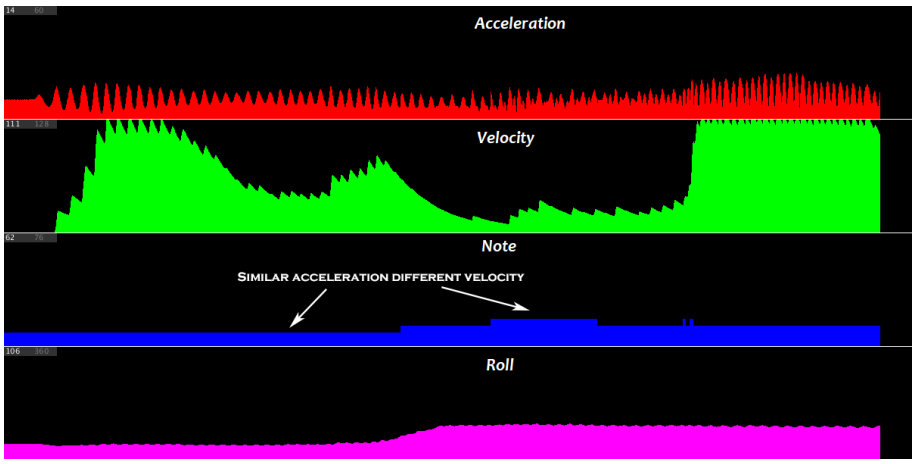


Figure 7.24: Test of the velocity-computation algorithm 2

```
float linearfactor=0.75;
float lineardecfactor=0.6;

if (floor((curracc*linearfactor)-G) >=0){
  if (volumeincreasing+floor((curracc*linearfactor)-G) < 128 ) {
    volumeincreasing=volumeincreasing+floor((curracc*linearfactor)-G);
  }
  else{
    volumeincreasing=127;
  }
  last=false;
}
else{
  if (volumeincreasing - lineardecfactor * (volumeincreasing / 127) > 0
    ) {
  volumeincreasing = volumeincreasing - lineardecfactor * (
    volumeincreasing / 127);
  last=false;
  }
  else{
  volumeincreasing=0;
  }
}
prevvelocity=floor(velocity);
velocity=volumeincreasing;


if (prevvelocity != floor(velocity)){
  AppleMIDI.controlChange(7, floor(velocity), 1);
}
```

Listing 7.11: Version 2 of velocity computation

Thanks to the second correction discussed before, it has been noticed that the impact of a movement on the velocity was maximum when the hand was parallel to the floor (in both positions) and minimum when it was orthogonal. Then, an ulterior correction factor has been added, scaling the impact of the acceleration on the velocity incrementation with respect to the roll. After different tests they have been derived a maximum scale factor of 1.2 if the roll is -90° or 90° and a minimum one of 1 if the roll is 0° or 180°.

The function allowing to perform this scaling operation has been mathematically derived as follow:

- computing the straight lines passing through the following two couples of points: (1.2, 90) (1, 180) and (1, 0) (1.2, 90);
- combining them with the absolute value
- applying the absolute value to the roll for including the case of roll < 0.

In Arduino language, the resulting operation is:

```
maxfactor=1.2;
minfactor=1;
rollfactor=maxfactor-(maxfactor-minfactor)*(abs(90-abs(roll))/90);
```

Listing 7.12: Roll factor computation

The following figure shows the results of the test of this version.

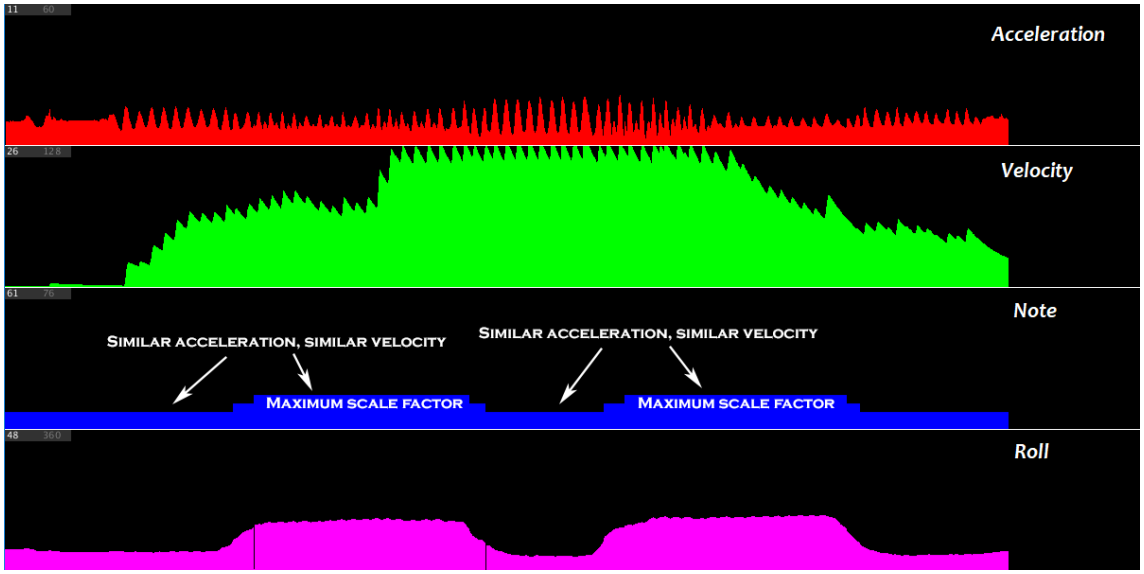


Figure 7.25: Test of the velocity-computation algorithm 3

The last improvement of the code, regards the introduction of an averaging operation on the velocity. The oscillations that still remained when trying maintaining the note, brought to the idea of considering the past of the velocity, not only in terms of the last computed one, but for a longer time. Initially, it has been performed the classical average on the last 50 values of velocity, memorizing them in a vector, using the following function:

```
float averaging(float Vi, float pastV){
  int iA = 0;
  float ret=pastV*N-velocityvector[0];

  for (iA=0; iA < N-1; iA++){
    velocityvector[iA]=velocityvector[iA+1];
  }
  velocityvector[N-1]=Vi;
  ret=ret+Vi;
  ret=ret/N;
  return ret;
}
```

Listing 7.13: Average-based velocity computation - version 1

Despite the problem of oscillations was almost solved, this solution was very time-consuming, causing the overall system slowdown. Moreover, as expected, the system response was not more real-time, especially for "instantaneous" movements.

The main causes of this problems were the high number of values in the velocity window and the average computation algorithm.

Great improvements have been obtained applying the following two corrections:

- reduction of the velocity window from 50 to 30 elements
- computation of an approximate average, using the following code which avoids the velocities storing considering as older value the average itself.

```
1 float averagingS(float Vi, float pastV, int numel){
2   float ret=pastV*numel-pastV;
3   ret=ret+Vi;
4   ret=ret/numel;
5   return ret;
6 }
```

Listing 7.14: Average-based velocity computation - version 2

Further, for a better response to instantaneous movements, the averaging function has been used two times, considering also a lower-wide velocity window and using this last as velocity value if very high with respect to the average on 30 values. This last version is the final one and the following are the test results and the full code:

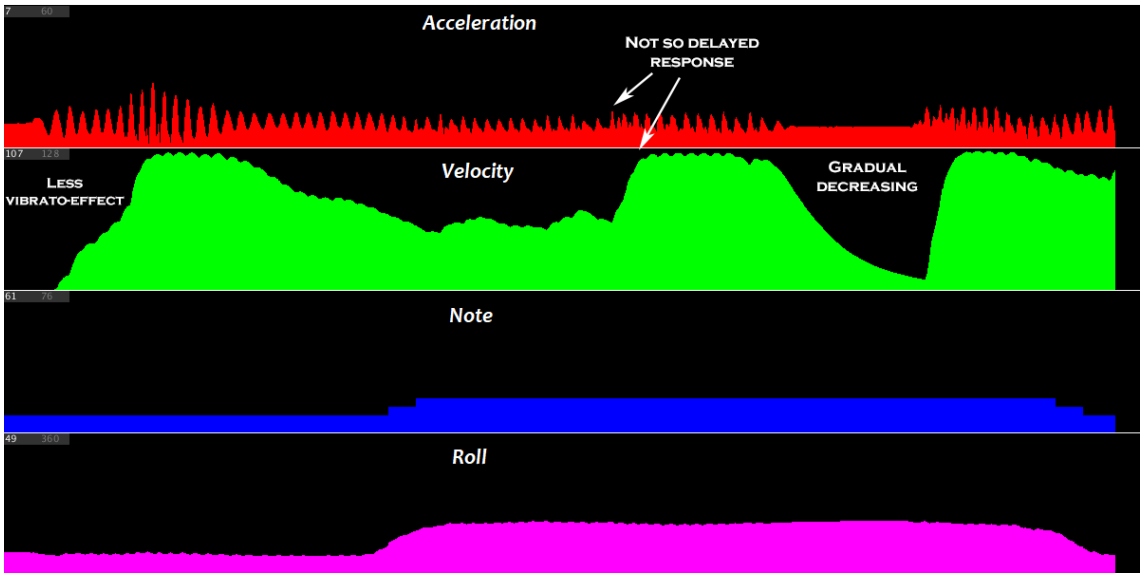


Figure 7.26: Test of the velocity-computation final algorithm

```
if (notetoplay != oldnote){
  for (int i=0; i<preinchord; i++){
  AppleMIDI.noteOff(oldnote+i*16, 1, 1);
  delay(20);
  }
  preinchord=inchord;
  for (int i=0; i<inchord; i++){
  AppleMIDI.noteOn(elabroll+60+i*16, velocity,1);
  delay(20);
  }
}
else{

  if (floor((curracc*linearfactor*rollfactor)-G) >=0){
    if (volumeincreasing+floor((curracc*linearfactor*rollfactor)-G) < 1
    28 ) {
    volumeincreasing=volumeincreasing+floor((curracc*linearfactor*
    rollfactor)-G);
    }
    else{
    volumeincreasing=127;
    }
    last=false;
  }

  else{
  if (volumeincreasing - lineardecfactor * (volumeincreasing / 127) > 0
    ) {
    volumeincreasing = volumeincreasing - lineardecfactor * (
    volumeincreasing / 127);
    last=false;
  }
  else{
    volumeincreasing=0;
  }
  }

  prevvelocity=floor(velocity);
  velocity=averagingS(volumeincreasing, velocity, N);

  float velocityonless=averagingS(volumeincreasing, velocity, 25);

  if abs(velocityonless-velocity > 4) {
  velocity=velocityonless;
  }

  if (prevvelocity != floor(velocity)){
  AppleMIDI.controlChange(7, floor(velocity), 1);
  }
}
oldnote=notetoplay;
```

Listing 7.15: Delivering of the MIDI messages to the PC

Performance and costs evaluation

Speed

Through the following functions it has been possible to compute the speed of the instrument in terms of how many MIDI messages it is able to send in a second. This rate is also the frequency at which the Madgwick filter has to work.

```
1 Now = micros();
2 deltat = ((Now - lastUpdate)/1000000.0f);
3
4 Serial.print("filter rate = "); Serial.println(1.0f/deltat, 1);
5 Serial.print("deltat = "); Serial.println(deltat, 6);
```

Listing 7.16: Computation of filter rate and time between consecutive MIDI messages

Considering single notes (not chords), we can assume as maximum deltat 4.5 ms, reached sometimes when changing the note (because of the delay between Note On and Off messages placed for avoiding losses). However, the medium time is about 4 ms. Further, we have a minimum rate of about 220 Hz and a rate of 250 Hz on average (500 MIDI messages per second, really faster with respect to human possibilities).

Considering also chords, we reach a minimum rate, of course, when the instrument is set for playing 16 chords of 4 notes: in this case, the minimum rate is 6 Hz (48 MIDI messages per second). 6 Hz means having a maximum time of 166 ms: this is lower than the time considered to be the maximum acceptable latency between audio and video (200 ms). The perception of a real-time behaviour still remain.

Usability and limits

The B-Glove seems to be pretty to use. Its design makes it comfortable and the results in terms of MIDI production are acceptable. It presents a very low latency and the response to the user movements is pretty rapid, thanks to the working velocity-computation algorithm. It is pretty intuitive to use and does not requires particular accuracy. It is perfect for improvisation and it is quite easy to use it for solos on musical bases. However, there are some limitations in its usability.

A first limitation regards the kind of instrument which can be used for synthesizing the MIDI sequence produced by the B-Glove. Considering that the MIDI Note On signal is sent only one time at the beginning, the B-Glove results to be usable almost exclusively if synthesized with instruments for which velocity is equivalent to volume, like an organ.

The modulation of the velocity of a note in time has effect only if the note produced by the synthesizer maintains its volume in that time. This is not true for a piano, for example, which can synthesize the B-Glove only if it is used for producing a very speedy sequence of notes, each having a considerable velocity.

Another limitation, which can be overcame with some training like for every instrument on the Earth, regards the possibility for the user of deciding exactly the sequence of notes he wants to play. This is mainly due to the fact that there is not a physical separation between them and so, without practice, is difficult to play the desired note. However, it's the same for a chord of a contrabass: there are not indicators showing the transition from one note to another.

The main nuisance speaking about the B-Glove regards the necessity of maintaining it straight during its ignition. This is an intrinsic limit of the IMU and results to be difficult to be overcome.

Reliablity

We can say that the overall system is reliable from the point of view of the communication between the instrument and the PC. Thanks to the programming strategies adopted both for the instrument and the Instruments Manager codes (see the following chapter), the communication is enough monitored and they are avoided losses of MIDI messages. Moreover, if the connection is lost, the instrument is reset in order to reconnect when it becomes available again.

Safety

The B-Glove can be considered a safe device, both for the user and from the instrument itself point of view. Mounting the electronics on an internal glove, allowed avoiding the contact with the user, eventual short circuits and wounds for the user. Moreover, they have been considered the operating conditions of the main parts of the B-Glove (AFH and IMU) and they are made for work at very high temperature (about 80°), which never be reached.

Costs

In terms of hardware, the B-Glove is not so expensive to realize. It is composed by the following components, listed with the respective prices:

Component	**Price**
Adafruit Feather Huzzah with ESP8266	21.62 €
Adafruit IMU LSM9DS0	29.21 €
LiPo Battery pack	8.25 €
Charger module	1.54 €
Switch	0.80 €
Level bubble	0.62 €
Gloves	~15 €
Total cost	~80 €

In terms of programming, it required the highest effort researching the best way for exploiting the acceleration for computing the velocity. Another considerable part of effort has been taken by the definition and the implementation of a good communication protocol between the B-Glove and the PC, especially for delivering MIDI messages from the PC to the outside. It can be said that the programming part required between 1 and 2 person months.

Instruments Manager

In order to use the B-Glove and to make it easier to manage it from the PC, a software has been developed, called Instruments Manager.

As the name says, this software is not devoted to the exclusively usage with the B-Glove, but it is meant for working also with other instruments, adopting the same communication protocol which has been established between the B-Glove and the PC, and with multiple instruments at a time.

For the moment, the Instruments Manager software is able to manage one instrument at a time, considering only the second (1 starting by 0) MIDI channel. However, it is not so difficult to extend the program for managing more (up to 16) instruments contemporary, exploiting all the channels.

Taking into account these considerations, the discussion about the Instruments Manager software will be as abstracted as possible from the instrument. The B-Glove will be used only as example.

The Instruments Manager software does not establish the connection with the instruments (this is done by rtpMIDI), but manages their MIDI settings. In particular, it is able of starting and stopping the sequencing operation, sending the number of notes/-chords the instruments have to play, opening and managing Ableton Live sets. So, it is an intermediate software between the driver (rtpMIDI) and the synthesizer (Ableton Live).

The main motivation driven us to the realization of this software (instead of using, for example, a Max device or plug-in), was the necessity of a easier and more intuitive way for managing the instrument and a dissociation from Ableton, maybe less intuitive and, in each case, replaceable with another synthesizer, perhaps cheaper.

In the first section of this chapter we will see how the PC is prepared for using Instruments Manager and how it works. Besides rtpMIDI and Ableton, Max for Live is used for mapping the notes coming from the instruments to a musical scale or to a sequence of note decided during the realization of the live set; moreover, we will see how MIDI-OX plays a fundamental role creating the communication channel between the output signals produced by Instruments Manager and the instruments.

PC preparation

Connection settings on the instrument

How we said above, the instrument is programmed for connecting to a WiFi network having SSID equal to "glove" and key equal to "blackglove". This has been done in order to avoid reprogramming it or sending new credentials when changing the network.

The setting-up of this network can be done by changing the parameters of the personal router or, in alternative, by using an hotspot software. This last solution has been tested; in particular, the MyPublicWiFi software has been used.

The network has been set and started like showed in the following image. This corresponds, on Windows 10, to starting the hostednetwork using the "netsh wlan start hostednetwork" command on the shell.

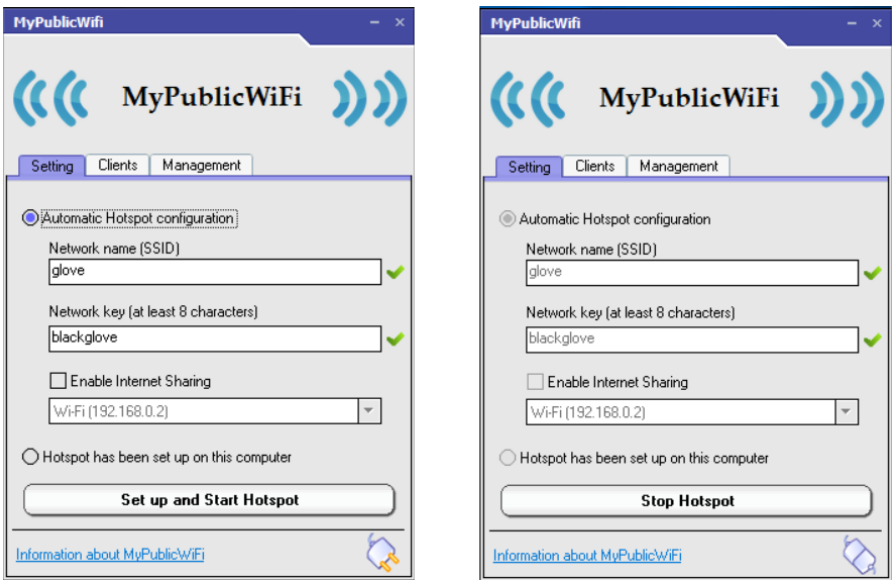


Figure 8.27: MyPublicWiFi - Glove-network setting-up

Then, the instrument has been powered-on and, after a while, it connected to the network. The IP address can be obtained by looking at the "Clients" tab and, as we will see in the next section, it will be useful for setting the MIDI communication on rtpMIDI. The following figure shows an example of a connected client.

rtpMIDI: setting up the connection

What is done with rtpMIDI is setting up the connection, through the network, between the MIDI instrument and the PC. Thanks to rtpMIDI, the MIDI messages arrives to the

Figure 8.28: MyPublicWiFi - Clients tab and IP getting

PC via WiFi. For doing that, you have to follow two steps:

1. setting up a session, clicking on the first [+] button in the interface and enabling it;
2. adding the instrument to the "Directory" space, by clicking on the second [+] button and writing a name and the IP address you get on MyPublicWiFi.

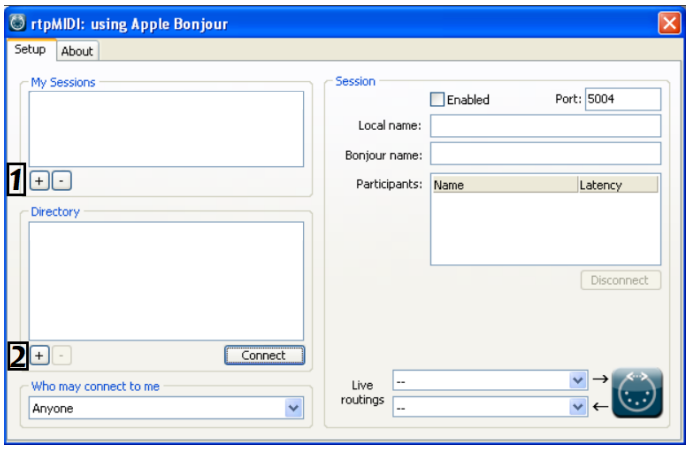


Figure 8.29: rtpMIDI empty interface

If you do not use a hotspot software, the IP address assigned to the instrument can be got through the router management page. In alternative, you can connect the instrument via USB to the PC and read the Serial port: in this way you can read the IP address which is printed once the instrument has joined the network.

MIDI-OX and MIDI-Yoke

After installed MIDI-OX and MIDI-Yoke, you can set up a channel from the output of the Instruments Manager software to the network. The output of this channel will be the network session created on rtpMIDI; its input will be one of the MIDI-Yoke ports which will be also used in output by Instruments Manager, like showed in the following figure.

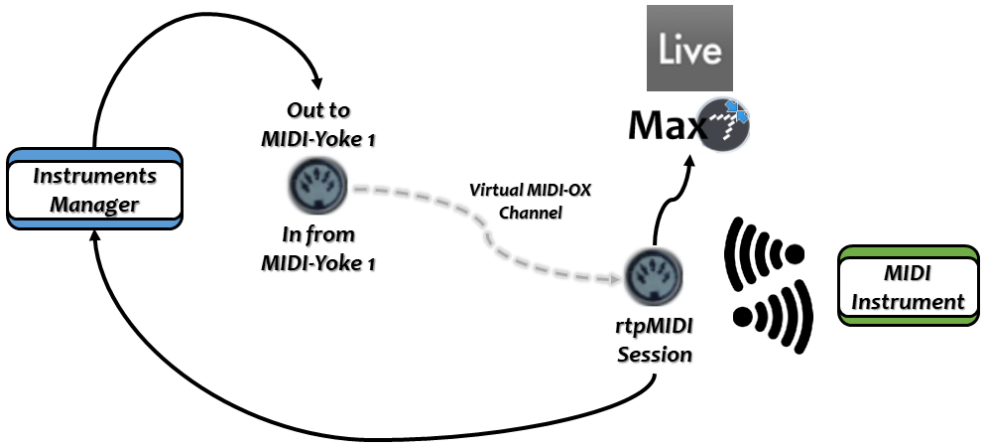


Figure 8.30: Connections

In order to set up this channel, it is sufficient to open the MIDI-OX software, open Options -> Midi Devices and select the inputs and outputs devices. Then, click on the Port Routings button in the upper sidebar and configure the connection like showed below.

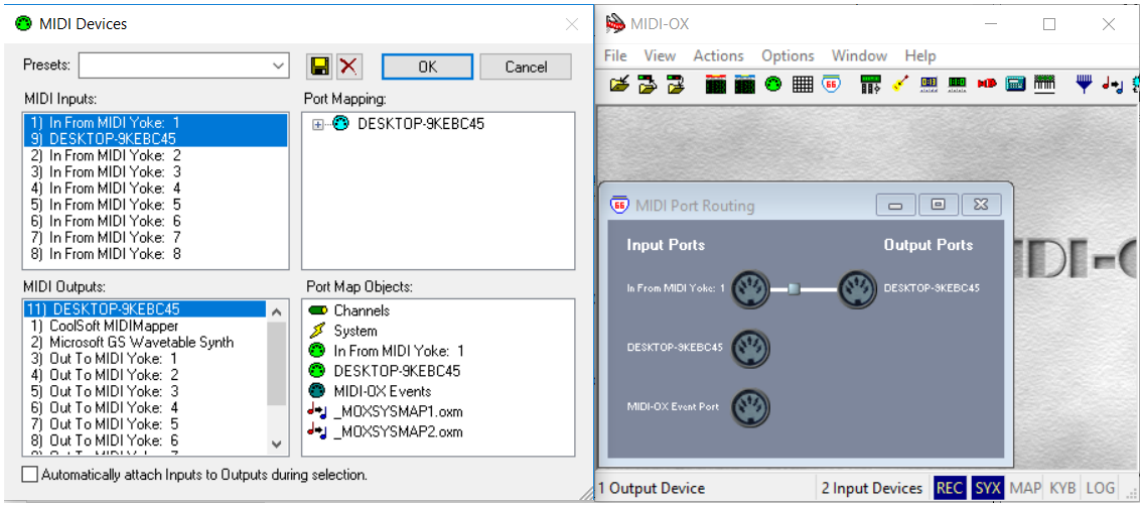


Figure 8.31: MIDI-OX Configuration

Ableton Live sets and Max MIDI Notes Mapper

The setting operation on Ableton consists of three steps. The first one is allowing Ableton to receive both notes (Note On and Off MIDI messages) and to be controlled from the outside (Control MIDI messages). This can be done by putting "Yes" in both the "Track" and "Remote" columns, on the line correspondent to the rtpMIDI session, in Preferences -> MIDI.

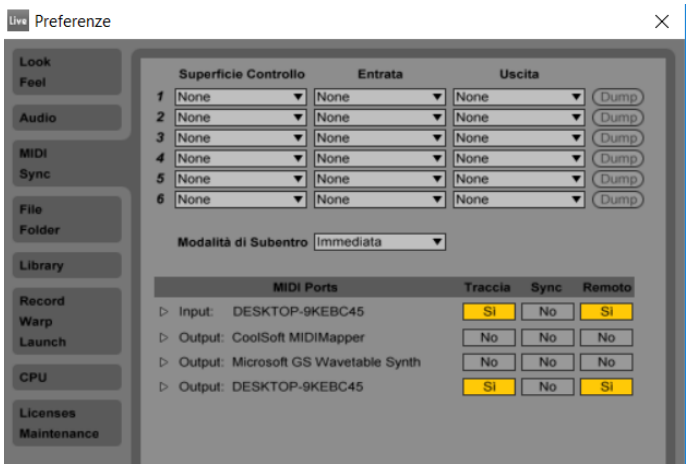


Figure 8.32: Enabling inputs on Ableton Live

The second step is mapping the notes to the wanted scale. This can be done using the Max for Live MIDI Notes Mapper device. Once you dragged the instrument on a column in the session view, you have to drag there also the MIDI Notes Mapper plug-in. Opening it, you got the interface below where you have to map the notes from 60 to 123 (the only ones played by the instrument as standard), or a portion of them, to the wanted ones.

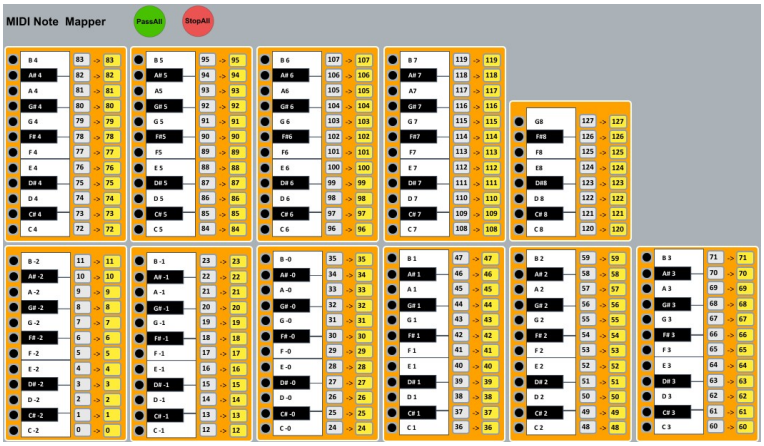


Figure 8.33: Midi Notes Mapper for music scale setting

Finally, you have to map the Control-7 Change messages to the Track Volume. For doing that you have to power on the instrument and to start the MIDI communication (see the software explanation section). At this point, you can perform the mapping by right-clicking on the Track-Volume and then clicking on "Edit MIDI Mapping".

Figure 8.34: Mapping to Track Volume control

Now, it is sufficient to produce with the instrument the event triggering the MIDI Control-7 Change message (for the B-Glove, moving the hand) and this command will be automatically mapped to the Track Volume control, like showed below.

Figure 8.35: Control-7 Change mapped to Track Volume control

Why Visual Basic?

The initial version of the software only was in charge of giving a visual feedback depending on the incoming MIDI messages. In particular, the background colour of the window should change depending on the musical note.

This results to be helpful if the glove is used for producing a determined sequence of note (less if improvising) because of the possibility of knowing the note that is to be played before it is actually played: changing the inclination of the hand, without a movement producing acceleration and so a velocity, you doesn't listen anything but you see the selected note.

For this purpose, it was sufficient to intercept the MIDI messages and to map each note to a different colour. It has been done starting from a VB-Net software found on the Internet [30], exploiting the COM port of MIDI-OX in order to read the incoming MIDI messages. It has been rewritten for Visual Basic and it has been added the colouring part.

Despite the lower level operations, as we will see in the following section, are performed in C#, it has been decided to continue using Visual Basic for developing the software interface, because of its simplicity of use.

The software

The final software is based on a C# package, downloaded from the Internet [31] as part of the VB-Net source code of MIDI utility. The package is called Toub.Sound.Midi, it has been written by Steve Toub in 2004 and it contains a MIDI player, a MIDI parser and other utilities exploiting the MIDI APIs for reading, writing, modifying, and playing MIDI events/tracks/sequences/files.

Instructions for use

The initial interface, appearing when starting the program, is the following:

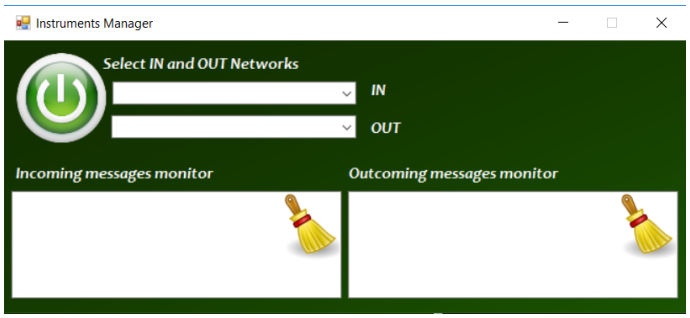


Figure 8.36: Instruments Maganer start-up GUI

The software is, for the moment, programmed for using only an instrument at a time. In order to use the Instruments Manager software, you have for first to:

- open the rtpMIDI and MIDI-OX software;
- configure them like is explained in the PC preparation section;
- power on and connect the instrument to rtpMIDI.

Only at this point you can open the IM software. If you forget to execute the first step, the program gives an error message and doesn't start. If you don't arrive to the last step, the power-on button of the software remains yellow for three seconds, then gives an error.

Further, in order to connect to the instrument, you have to select the right network sessions for incoming and outcoming MIDI messages. If you configure MIDI-OX and MIDI-Yoke like explained before, you have to select as output network the first "Out to M" (the complete names of the MIDI-Yoke ports are hidden because of the C# library). The input network, instead, is the session generated by rtpMIDI.

Once you have correctly set the networks, if the instrument is connected to rtpMIDI and the MIDI-OX mapping has been rightly configured, clicking on the power-on button of the software, it becomes red and the window enlarges, showing the rest of the GUI.

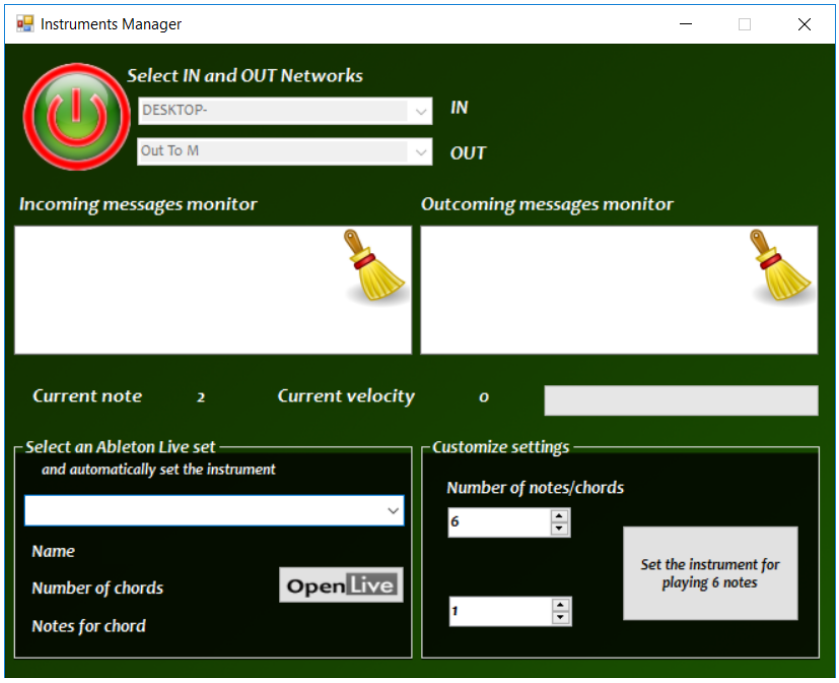


Figure 8.37: Instruments Maganer GUI

Last incoming and outcoming MIDI messages are automatically showed in the two monitors. These last can be cleared by clicking the relative clear buttons. Moreover, the parameters of MIDI-height and MIDI-velocity of all the incoming Note On and Control-7 Change MIDI messages are used for setting the current note and the velocity in the labels and in the progress bar.

The software can work in two modalities:

- with a predefined Live Set
- in a customized way

At the beginning, all the Live Sets contained in the software folder are uploaded in the corresponding drop down list. If there are some new files, they are requested the number of chords and the number of notes-per-chord which the instrument has to play when each of them is selected. When the program is closed, these last are memorized in a text file that is loaded the next time the software is opened.

If you want to use one of these Live Sets, you can select it in the part of the interface on the left, checking the right number of chords and notes-per-chord and opening Ableton Live, contemporary sending to the instrument the correct number of notes.

Otherwise, you can open Ableton Live on your own and send a customized number of chords (between 1 and 16) and a customized number of notes-per-chord (between 1 and 4) to the instrument, using the part of the interface on the right.

The code

In this section some important functional parts of the code are analysed. For some of them, the code is showed, others are analysed in terms of flow chart. The complete code is in the Appendix.

As mentioned when motivating the VB choice, this software exploits a C# library of MIDI functions. In particular, the Toub.Sound.Midi library is used for output-purposes. Moreover, in order to perform the input-monitoring operations, it exploits the "winmm.dll" Windows Multimedia APIs [32]: some of them are redeclared for being used in the program.

As an example, the following piece of code shows the declaration of two API-based functions: the midiInGetNumDevs, getting the number of input MIDI devices, and the midiInOpen, which opens a MIDI input device for monitoring. The MIDIINCAPS type is a structure able to hold some information about an input device, like showed below.

```
1 Public Declare Function midiInGetDevCaps Lib "winmm.dll" Alias "midiInGetDevCapsA"
      (ByVal uDeviceID As Integer, ByRef lpCaps As MIDIINCAPS, ByVal uSize As
      Integer) As Integer
2 Public Declare Function midiInOpen Lib "winmm.dll" (ByRef hMidiIn As Integer,
      ByVal uDeviceID As Integer, ByVal dwCallback As MidiInCallback, ByVal
      dwInstance As Integer, ByVal dwFlags As Integer) As Integer
```

Listing 8.17: winmm.dll declarations

```
1 Public Structure MIDIINCAPS
2   Dim wMid As Int16 ' Manufacturer ID
3   Dim wPid As Int16 ' Product ID
4   Dim vDriverVersion As Integer ' Driver version
5   <MarshalAs(UnmanagedType.ByValTStr, SizeConst:=32)> Dim szPname As String '
      Product Name
6   Dim dwSupport As Integer ' Reserved
7 End Structure
```

Listing 8.18: MIDIINCAPS structure

The midiInOpen function implies the declaration of one more function, the MidiInCallback, a callback function for handling the incoming MIDI messages which is declared as a Public Delegate function, that is a function which can be associated to a method having the same form in terms of parameters and returned value type. In this case, the function that is effectively called for handling the incoming MIDI messages is the MidiInProc, whose declaration and body are in the following piece of code.

Moreover, even the MidiInProc function needs to delegate another function for displaying data that have to be monitored. This is done through declaring the DisplayData function which, like you can see in the following piece of code, is called each time the MidiInProc function is called.

```
1 Public Delegate Function MidiInCallback(ByVal hMidiIn As Integer, ByVal wMsg As
      UInteger, ByVal dwInstance As Integer, ByVal dwParam1 As Integer, ByVal
      dwParam2 As Integer) As Integer
2 Public ptrCallback As New MidiInCallback(AddressOf MidiInProc)
3
4 Public Delegate Sub DisplayDataDelegate(dwParam1)
5
6 Function MidiInProc(ByVal hMidiIn As Integer, ByVal wMsg As UInteger, ByVal
      dwInstance As Integer, ByVal dwParam1 As Integer, ByVal dwParam2 As Integer)
      As Integer
7   If MonitorActive = True Then
8     txtMonitorIN.Invoke(New DisplayDataDelegate(AddressOf DisplayData), New Object
      () {dwParam1})
9   End If
10 End Function
```

Listing 8.19: MidiInProc function

The DisplayData function intercepts the incoming MIDI messages and, depending on the kind of messages, updates the input monitor. Looking at the code of the B-Glove, we can easily derive the kind of MIDI messages which can arrive:

- Note (60 to 76) On, Note (60 to 76) Off, Control-7 Change
- Control-9 Change for powering on/off and notes/chords-number changing acknowledgements.

Before analysing the DisplayData function, it is useful to give some more information about the general structure of the software. It works as a Finite State Machine, changing its state depending on the user actions and on the instrument behaviour.

Clicking on the power button, the user can initiate or stop receiving MIDI messages from the instrument and monitoring them. What happens on clicking, is a transition to a waiting state (one for each of the two possible transitions: ON/OFF, OFF/ON), during which the software waits for the ack MIDI message from the instrument, setting a 3-seconds timer. Elapsed this time, the software gives an error message and goes in the OFF state.

Something similar happens for the number of notes/chords message: the software waits for an ack like before and goes in the OFF state if, after 3 seconds, it has not arrived. The definition of the states and the FSM are showed below.

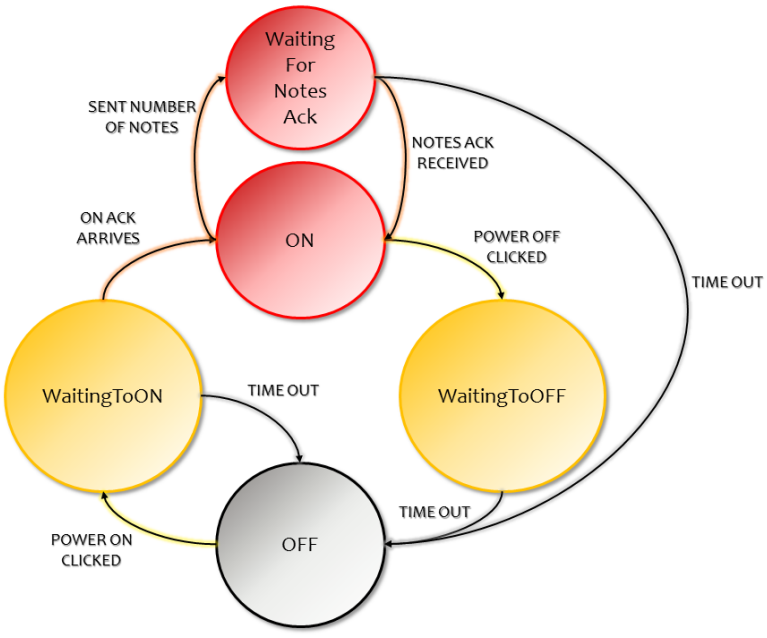


Figure 8.38: Instruments Maganer FSM

```
Public Enum instrState
  pOn
  pOff
  WaitingToOff
  WaitingToOn
End Enum
```

Listing 8.20: Instruments Manager FSM states definition

Further, we already said that the software gives the possibility of setting the instrument and sending the right numbers of chords and notes-per chords for a particular Ableton Live Set. This requires a preparatory phase which is executed at the beginning, during the initialization of the program.

What is done in practice, is loading into the structure defined below and into the related drop-down list, all the Ableton Live sets contained in the software folder. For each of them, it is checked if the numbers of chords and of notes-per-chord have been already memorized: if they have been not, they are requested to the user and they are stored in a text file. Further, the stored data of files which are no longer present in the folder are deleted.

```
Private Structure song
  Public Name As String
  Public NON As Integer
  Public IC As Integer
  Public ToDel As Boolean
End Structure

Dim allsongs() As song
```

Listing 8.21: Songs structure

At this point, the user can select a Live set from the ones showed in the drop-down list and then, through the openLive button, send to the instrument the corresponding number of chords and the number of notes-per-chord.

A very similar approach is used for sending a customized number of chords and notes-per-chord through the sendNotes button. It is not showed here, but you can find it in the whole code in the Appendix.

The code of the openLive button is showed below: the Timer is set, as well as a flag that will be used in the DisplayData function for correctly interpreting the Ack. Note that the output monitor is used here for informing the user of the messages which are sent to the instrument. In the same way, it is used in the power on/off button.

```
Dim DeviceID As Integer = comboNetOUT.SelectedIndex
Dim device As MidiInterop.MidiDeviceHandle = MidiInterop.OpenMidiOut(DeviceID)
Dim output As Integer

Dim notesinchord As Integer = 89 + lblNotesIC.Text

numberOfNotes = lblNumberOfNotes.Text
waitingForNotesAck = True
fromLive = True
txtMonitorOUT.AppendText("Number of notes for Live set has been sent" & vbCrLf)
output = Convert.ToInt32(Hex(numberOfNotes) & "3D90", 16)
MidiInterop.SendMidiMessage(device, output)

txtMonitorOUT.AppendText("Notes per chord for Live set has been sent" & vbCrLf)
output = Convert.ToInt32(Hex(notesinchord) & "3D90", 16)
MidiInterop.SendMidiMessage(device, output)

timeTimer = timerSeconds
Timer1.Start()
device.Close()
```

Listing 8.22: openLive button code

The time has come to analyse the DisplayData function. It is nothing more than a multiple-branches if-else statement, which considers all the possible incoming MIDI messages and fills the input monitor.

The incoming MIDI messages are composed by three bytes which are parsed and stored in three variables: StatusByte, DataByte1, DataByte2. The StatusByte indicates the kind of message:

- 176 -> Control Change
- 144 -> Note On
- 128 -> Note Off

The DataByte1 of a Control Change message indicates the Control to change: if it is equal to 9, we have an ack and we have to distinguish between the four kinds of ack; if it is equal to 7, we have a velocity variation. The total number of possible input messages is 25: 22 for acknowledges and 3 for sequencing.

When an ack arrives, if it is related to a number-of-notes variation, it is checked if it has been requested in a customized way or for an Ableton Live set. In this second case, the right Live set is opened. Thus, this happen only when the ack has been received.

In the following, the code of the DisplayData function is showed, as well as a scheme representing the meaning of the incoming MIDI messages

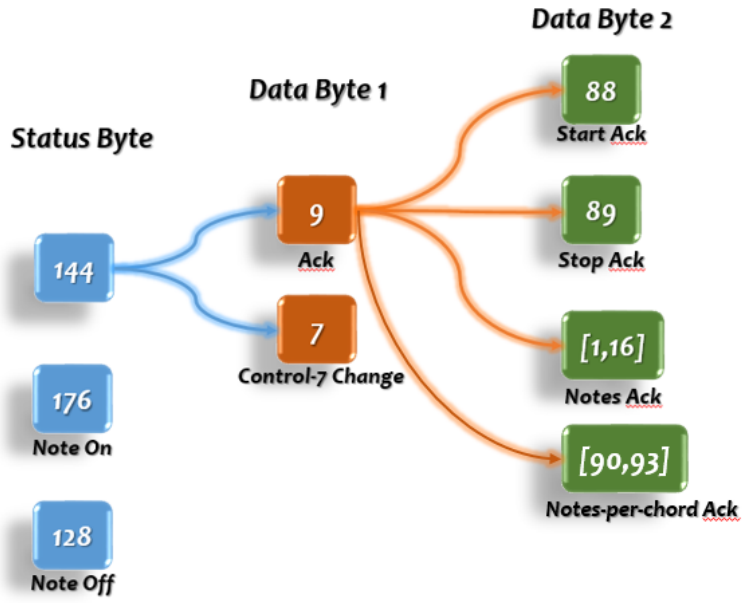


Figure 8.39: DisplayData scheme

```
1  Private Sub DisplayData(dwParam1)
2    StatusByte = (dwParam1 And &HFF)
3    DataByte1 = (dwParam1 And &HFF00) >> 8
4    DataByte2 = (dwParam1 And &HFF0000) >> 16
5
6
7    If poweredon = instrState.WaitingToOn And StatusByte = 176 And DataByte1 = 9 And
         DataByte2 = 88 Then
8      txtMonitorIN.AppendText("Start" & vbCrLf)
9      poweredon = instrState.pOn
10     btnPower.BackgroundImage = Image.FromFile("./images/ON.png")
11     Me.Height = 539
12     Me.Width = 670
13
14   ElseIf poweredon = instrState.WaitingToOff And StatusByte = 176 And DataByte1 =
       9 And DataByte2 = 89 Then
15     txtMonitorIN.AppendText("Stop" & vbCrLf)
16     poweredon = instrState.pOff
17     btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
18     Me.Height = 302
19     Me.Width = 670
20
21   ElseIf poweredon = instrState.pOn And StatusByte = 176 And DataByte1 = 7 Then
22     txtMonitorIN.AppendText("Control-7 Change: " & DataByte2 & vbCrLf)
23     ProgressBar1.Value = DataByte2
24     lblVelocity.Text = DataByte2
25
26   ElseIf poweredon = instrState.pOn And StatusByte = 144 Then
27     txtMonitorIN.AppendText("Note " & DataByte1 - 59 & " On, Velocity: " &
       DataByte2 & vbCrLf)
28     ProgressBar1.Value = DataByte2
29     lblVelocity.Text = DataByte2
30     lblNote.Text = (DataByte1 - 59) Mod 16
```

```
31
32    ElseIf poweredon = instrState.pOn And StatusByte = 128 Then
33      txtMonitorIN.AppendText("Note " & DataByte1 - 59 & " Off" & vbCrLf)
34
35    ElseIf waitingForNotesAck = True And StatusByte = 176 And DataByte1 = 9 And
        DataByte2 > 0 And DataByte2 <= 16 Then
36      waitingForNotesAck = False
37      txtMonitorIN.AppendText("Received Ack for number of notes" & vbCrLf)
38      Timer1.Stop()
39      If fromLive = True Then
40        fromLive = False
41        If (System.Diagnostics.Process.GetProcessesByName("Ableton Live 9 Suite").
        Length > 0) Then
42          System.Diagnostics.Process.GetProcessesByName("Ableton Live 9 Suite")(0).
        Close()
43        End If
44        System.Diagnostics.Process.Start(FileFolderPath & comboLiveSet.SelectedItem
        & " Project\ " & comboLiveSet.SelectedItem & ".als")
45      End If
46    ElseIf waitingForNotesAck = True And StatusByte = 176 And DataByte1 = 9 And
        DataByte2 > 89 And DataByte2 <= 94 Then
47      waitingForInChordAck = False
48      txtMonitorIN.AppendText("Received Ack for notes in chord" & vbCrLf)
49
50    End If
51
52    If poweredon = instrState.pOff Then
53      midiInStop(hMidiIn)
54      midiInClose(hMidiIn)
55      comboNetIN.Enabled = True
56      comboNetOUT.Enabled = True
57      MonitorActive = False
58    End If
59
60  End Sub
```

Listing 8.23: DisplayData function

As last aspect related to the code of the Instruments Manager software, the instructions executed when clicking on the power button are listed below. Here is the management of the MIDI input devices, the initialization of the input monitor and so the call to the midiInOpen function discussed above.

```
If comboNetOUT.SelectedIndex = -1 Or comboNetIN.SelectedIndex = -1 Then
  MsgBox("Select the networks.")
Else

  Dim DeviceID As Integer = comboNetOUT.SelectedIndex
  Dim output As Integer
  Dim ss As String = "Start"
  Dim nomessage As Boolean = False

  If poweredon = instrState.pOff Then
    output = 88
    ss = "Start"
    poweredon = instrState.WaitingToOn
    btnPower.BackgroundImage = Image.FromFile("./images/WAIT.png")
  ElseIf poweredon = instrState.pOn Then
    ss = "Stop"
    output = 89
    poweredon = instrState.WaitingToOff
    btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
  Else
    nomessage = True
    poweredon = instrState.pOff
    btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
    Me.Height = 302
  End If

  If nomessage = False Then
    Dim device As MidiInterop.MidiDeviceHandle = MidiInterop.OpenMidiOut(DeviceID)
    output = Convert.ToInt32(Hex(output) & "3D90", 16)
    MidiInterop.SendMidiMessage(device, output)
    txtMonitorOUT.AppendText(ss & " command sent" & vbCrLf)
    device.Close()
  End If

  If poweredon = instrState.WaitingToOn Or poweredon = instrState.WaitingToOff
    Then
    DeviceID = comboNetIN.SelectedIndex
    comboNetIN.Enabled = False
    comboNetOUT.Enabled = False
    midiInOpen(hMidiIn, DeviceID, ptrCallback, 0, CALLBACK_FUNCTION Or
    MIDI_IO_STATUS)
    midiInStart(hMidiIn)
    MonitorActive = True
  Else
    midiInStop(hMidiIn)
    midiInClose(hMidiIn)
    comboNetIN.Enabled = True
    comboNetOUT.Enabled = True
    MonitorActive = False
  End If

End If
```

Listing 8.24: Power button code

Samir and the B-Glove

Why Samir?

Samir joined Tra silenzio e baccano during the the B-Glove development. His desire of movement and the possibility of the B-Glove to be carried around, completely matched between them.

Moreover, thinking about his Autism, the usage of the glove seemed a good idea to perform, in some way, a music therapy work on him. Some suggestions of Riccardo traced a path for helping Samir, including three main steps:

1. Meeting the instrument:
    - let him to use the instrument without obstacles, to play with it, to find the best interaction, (hopefully) to like it;
    - let him receiving a feedback from the instrument, recognising and feeling himself as the producer of the music.
2. Setting him into a musical context:
    - allow him being part of a song, build a musical harmony with him;
    - give to his experience a musical sense.
3. Joining to an orchestra composed by other guys with disabilities:
    - give him the possibility to be part of a more complex musical experience in which he have to respect a certain schedule;
    - let him musical communicating with others, contrasting the communicative isolation proper of autistic people, respecting some communication constraints in terms of time and methods.

These steps can be followed with every instrument and by every guy. However, the B-Glove gives to Samir the possibility of musically performing in the sense of "musiké" (words, sounds and movements) and following the Globality of languages approach: exploiting all the possible communication languages. Not only, he is free to move, but his movements becomes the origin of the sound.

Moreover, it can be also part of a behavioural therapy in which Samir is trained for playing a particular pattern of notes with the B-Glove and this pattern can be gradually modified or totally changed in order to educate him to accept the alteration of his daily repetitive behaviours.

At the first appointment: the birth of the name

After the first idea of the B-Glove had been implemented with the AFH microcontroller, so in a wireless way, an appointment with Samir has been organized, in order to let him testing its new toy and to collect some ideas in order to better adapt the glove to the most natural interaction for Samir.

He dressed the B-Glove and, when Riccardo started playing his guitar, the hand of Samir started moving, rapidly, up and down: he had found the best way for playing the B-Glove, completely autonomously.

Then, Riccardo started playing Dabuè, a song composed by him for being played by Tra silenzio e baccano, the B-Glove has been set in order to play 6 notes of the "E major pentatonic" music scale and Samir improvised on that base, rotating the hand and moving it up and down.

It has been decided to repeat the same thing, with the whole orchestra, during the near concert that 15 days after would take place in Poirino (TO).

At the end of this first appointment, it has been asked to Samir which name he would like to give to its new toy. Looking at it, Samir said "Guanto" (glove), and then "Guanto nero" (black glove).

The concert in Poirino (TO)

Surely, it has been a great and significant test bench for the instrument:

- Samir has been requested to play the B-Glove in a public context, in front of a lot of people.
- The B-Glove, still in the prototyped version and without the possibility of managing it from the PC (the Instruments Manager software was not yet developed), has been subject to a never tested working period: Samir was placed on stage and it cannot be possible to manage the powering-on of the instrument, paying attention to the initialization constraints, during the concert. Thus, the glove, even if silenced for the majority of the time, transmitted MIDI signals for a lot of time.
- Even the network context was a novelty for the B-Glove: it has been connected to an Apple Router together with 5 iPads and a MAC, all transmitting MIDI signals.

The intervention of Samir with the B-Glove was programmed during the Dabuè song. After involving other guys for playing their solos on his base, Riccardo has gone up on stage and called Samir for playing with him. Then, they have come down from the stage and they have danced in front of the public for some seconds. Finally, Riccardo has said to Samir to come back to his place and Samir has run, up to the stage, to his chair. The B-Glove had worked.

Figure 9.40: Samir, using the B-Glove prototype during the rehearsals for the live - Poirino

Ideas for the future

"Tra silenzio e baccano" is a project whose noble purposes stimulates in doing more and more. The passion of the owners of the Project, Riccardo and Emanuela, in helping disabled children in expressing themselves with music is perfectly combined with a desire of having fun together, developing new instruments, composing new tracks, inventing new schemes of interventions of the guys.

B-Glove and Instruments Manager improvements

Starting from the B-Glove instrument, as said before, it can be considered a usable device, not needing the interventions of a technician every time it has to be used.

The possibilities of improvement directly come from its limitations. The first is without doubts find a way for switching it on without having the must of maintaining the hand straight.

Further, despite the instrument is quite intuitive to use thanks to the algorithms for the velocity-computation, it could be a good idea researching new sensors allowing to exploiting, not more the acceleration, but the speed of the hand. This could make the instrument even more intuitive and enjoyable.

For what regards the functional part, it can be tried to increase the admitted number of notes-per-chord. If we maintain the standard of the instrument settable for 1 to 16 notes, 128 notes (MIDI available number) can be exploited for sending chords of up to 8 notes at a time: more than what a human can do. However, this can increase the latency too much.

Moreover, it could be added an echo modality in which the note does not dead immediately when a new one has to be played, but its volume decrease less drastically.

Considering the limitation connected to the possibility of being synthesized only with a particular kind of instruments, it could be interesting to introduce a modality in which the velocity-computation and the delivered MIDI messages are performed in a way which allows synthesizing it also with other instruments.

A further functionality can be obtained by adding a flex sensor to a finger: you can evaluate the flex of the fingers and exploit it for varying the linear decreasing coefficient of the velocity and having a better controllability of the dynamics, for example speeding up the decreasing when the hand is closed. It could be thought also to add other sensors in order to exploit other movements of the hand and to extend the possibilities of the B-Glove, mapping each sensor to a particular kind, for example, of MIDI Control.

An interesting thing which can imply a considerable research and practical work can be finding a way for playing the instrument in time to a music. The current B-Glove, like we said, is perfect for improvisation. Reducing the amount of training needed for playing it in a more controlled way could be a good thing, even considering the context in which it has to be used.

From the side of the Instruments Manager software, it have to be extent for handling and managing different devices at a time, with a view to the conversion of all the instruments of "Tra silenzio e baccano" (older and newer) to the same standard of communication.

Thinking at the synthesis, it could be useful to look for alternatives to Ableton and Max in order to maintain low the price of the whole system.

General ideas for Tra silenzio e baccano

The main purpose connected to the orchestra of "Tra silenzio e baccano" is reducing the amount of cables and wires used for powering and amplifying the instruments.

The first step could be making all the instruments able to communicate with the PC through the WiFi and avoiding amplification and powering cables. This can be immediately (and has been partially) done with the Cavarin, using an AFH instead of the Arduino Mega, and the Chime of Matteo, connecting piezos to an AFH instead that directly to a sound card. This last requires a research on the best way for connecting multiple analogue sensors to the AFH (having only an analogue-in pin). Moreover, this strategy can be used in the development of new instruments.

Then, searching or developing some devices in order to acquire the MIDI signals from the MIDI keyboards and sending them to the Instruments Manager Software, for applying mapping and synthesizing operations at the base station and avoiding the amplification cables.

Developing a WiFi MIDI Network between the iPads and the base station, in order to perform, even here, the synthesis with a single device and avoiding amplification cables.

Conclusions

This thesis work can be considered as a starting point of a project that can become very wide. Developing only an electronic musical instrument can be considered as a work for its own sake. But the B-Glove finds its continuity at its origin, in the motivation that inspired us for its creation: the disability.

The path that has been traced by Riccardo for Samir can be useful for him, but maybe even more useful for other autistic or disabled people presenting more accentuated communication and interaction problems. Samir is a joyous person, which loves the human contact. However, different kinds of communication disturbs lead some people to isolate themselves and it could be important intercepting those languages which are congenial to them and exploiting these last for integrating these people.

The Globality of languages discipline, we think, completely matches with the proposals of the B-Glove: deriving music, smiles and fun from a glove, with a very simple interaction, as a stimulus for many disabled people. As explained by Stefania Guerra Lisi in the foreword, using the B-Glove allows a MusicArTerapeutic dialogue with a person and also to unlock his body, allowing him to express himself through his own spontaneous music.

The B-Glove is for sure a wireless MIDI instrument and its capacities in sequencing can be improved both in terms of hardware and software, as we saw above. This thesis work, however, did not propose only a wireless MIDI instrument, but also a communication protocol and some rules for developing an orchestra of electronic instruments, manageable from the PC. In the optic of disability, this last aspect is very important because of the freedom in realizing every kind of instrument exploiting every kind of interaction, using the same easy protocol of sequencing and synthesis.

Besides Tra silenzio e baccano, the participation to the National convention of the Globality of languages gave the possibility to the author - in addition to a personal enrichment in the musical, social and communication fields - to receive a feedback to his work by the external world. A great interest has been showed by different kinds of people, from the University Professors to the parents of disabled guys. Some information about the final price and the time-to-market of the B-Glove have been requested, giving to the author suggestions for thinking at a commercial version of the B-Glove.

The initial target of the author has been achieved: the sea in which he tipped was almost completely unknown for him and, despite it had happened to deal with disabled people, the author went through a path allowing him to get closer to their world and to their needs and to give a practical explanation to the words of Ezio Bosso cited at the beginning of this thesis.

The world needs communication, not only information.
The world needs listening people, not only hearing ones.

This thesis has given to the author the possibility to meet the Globality of Languages, recognizing it, not only as a way for treating disabled people, but also for re-conducting the human being on the road of humanity. On the other side, it has given the possibility of understanding the difficulties connected to the promotion of wonderful and useful projects like Tra silenzio e baccano, which often have to deal with economic and social shortcomings.

From the point of view of who is writing, there's a great satisfaction for the work which has been conducted. The results may be debatable, but the path driven there provided the author with the possibility of putting in practice the whole studies of the Embedded System Master course for a very practical, social and not abstracted goal.

Bibliography

[1] Vincenzo Lombardo and Andrea Valle. *Audio e Multimedia*. Ed. by Apogeo Education. 2005.

[2] MIDI. *What's MIDI?* URL: https://www.midi.org/ (visited on 08/30/2017).

[3] MIDI-OX. *Description*. URL: http://www.midiox.com/ (visited on 09/06/2017).

[4] MIDI Yoke. *Description*. URL: http://www.midiox.com/myoke.htm (visited on 09/06/2017).

[5] Ableton. *Max for Live*. URL: https://www.ableton.com/en/live/max-for-live/ (visited on 09/06/2017).

[6] Cycling '74. *Max for Live*. URL: https://cycling74.com/products/maxforlive (visited on 09/06/2017).

[7] Vertecs. *MIDI Note Mapper 1.0*. URL: http://www.maxforlive.com/library/device/2273/midi-note-mapper (visited on 09/14/2017).

[8] Peregrine Horden. *Music as Medicine: The History of Music Therapy Since Antiquity*. Ed. by Routledge. 2016.

[9] Debbie Carroll. *Historical roots of Music Therapy: a brief overview*. Ed. by Revista do Núcleo de Estudos e Pesquisas Interdisciplinares em Musicoterapia. 2011.

[10] American Music Therapy Association. *History of Music Therapy*. URL: https://www.musictherapy.org/about/history/ (visited on 08/30/2017).

[11] Kenneth E. Bruscia. *Defining Music Therapy*. Ed. by Barcellona Publischers. 2014.

[12] World Federation of Music Therapy. *What is Music Therapy?* URL: http://www.musictherapyworld.net/WFMT/Home.html (visited on 09/01/2017).

[13] American Psychiatric Association. *Manuale diagnostico e statistico dei disturbi mentali DSM-5*. Ed. by Raffaello Cortina Editore. 2014.

[14] Shi Zhi-Min and Qing Xie Gui-Hong Lin and. *Effects of music therapy on mood, language, behavior, and social skills in children with autism: A meta-analysis*. Ed. by Chinese Nursing Research. 2016.

[15] Z. Warren, J. Veenstra-VanderWeele, and W. Stone. *Therapies for Children With Autism Spectrum Disorder*. Ed. by Agency for Healthcare Research Comparative Effectiveness Reviews No. 26 and Quality (US). 2011.

[16] A.S. Weitlauf, M.L. McPheeters, and B. Peters. *Therapies for Children With Autism Spectrum Disorder: Behavioral Interventions Update*. Ed. by Agency for Healthcare Research Comparative Effectiveness Review No. 137 and Quality (US). 2014.

[17] Łucja Bieleninik et al. *Effects of Improvisational Music Therapy vs Enhanced Standard Care on Symptom Severity Among Children With Autism Spectrum Disorder*. Ed. by American Medical Association. 2017.

[18] Melanie Voigt. *Orff Music Therapy*. URL: https://www.voices.no/index.php/voices/article/view/134/110 (visited on 09/15/2017).

[19] Stefania Guerra Lisi. *Globality of languages*. URL: http://www.centrogdl.org (visited on 09/15/2017).

[20] A. S. Hardy. *Quaternions*. Ed. by Health 'Ginn and Company'. 1881.

[21] Sebastian O.H. Madgwick, Andrew J.L. Harrison, and Ravi Vaidyanathan. *Estimation of IMU and MARG orientation using a gradient descent algorithm*. Ed. by IEEE. 2011.

[22] X io Technologies. *Open source IMU and AHRS algorithms*. URL: http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/ (visited on 09/14/2017).

[23] Remidi. *Remidi T8*. URL: https://www.remidi-pro.com/ (visited on 09/14/2017).

[24] Ray Li (Student of Cornell University). *Aura*. URL: http://www.cornell.edu/video/ray-li-invents-electronic-musical-instrument-aura (visited on 09/14/2017).

[25] Global DJ. *Tornado A1*. URL: http://en.global-dj.com/ (visited on 09/14/2017).

[26] Anson Dorsey et al. *Glove Midi Controller*. URL: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2010/ecg35_ajd53_jps93/ecg35_ajd53_jps93/index.html (visited on 09/14/2017).

[27] Adafruit. *Midi Drum Glove*. URL: https://learn.adafruit.com/midi-drum-glove/overview (visited on 09/14/2017).

[28] Maxwell Dergosits, Richard Branciforte, and Students of Cornell University. *Gesture-based MIDI Glove*. URL: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/mad293_rjb297/mad293_rjb297/index.html (visited on 09/14/2017).

[29] Mimu Tech. *Mimu Glove*. URL: http://mimugloves.com/tech/ (visited on 09/15/2017).

[30] by julynessi Topic on the MIDI-OX website Forum section. *VB.net Code to Work with midi-ox*. URL: http://www.midiox.com/cgi-bin/yabb/YaBB.pl?board=MOXScript;action=display;num=1293921718 (visited on 09/10/2017).

[31] Steve Toub. *Toub.Sound.Midi*. URL: https://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=5525&lngWId=10 (visited on 09/10/2017).

[32] Microsoft. *winmm MIDI functions*. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/dd798495(v=vs.85).aspx (visited on 09/13/2017).

Appendices

B-Glove full code

```
/* LIBRARIES DECLARATIONS */

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <SFE_LSM9DS0.h>
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <AppleMidi.h>
#include "MPU6050_6Axis_MotionApps20.h"


void(* resetFunc) (void) = 0;//declare reset function at address 0


/* DEFINITIONS */

#define GyroMeasError PI * (40.0f / 180.0f)       // gyroscope
    measurement error in rads/s (shown as 3 deg/s)
#define beta sqrt(3.0f / 4.0f) * GyroMeasError   // compute beta

#define G SENSORS_GRAVITY_STANDARD


/* COMPONENTS DECLARATIONS */

MPU6050 mpu;

#define LSM9DS0_XM  0x1D // Would be 0x1E if SDO_XM is LOW
#define LSM9DS0_G   0x6B // Would be 0x6A if SDO_G is LOW
LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);


/* INTEGRATION TIME VARIABLES */

float deltat = 0.0f;          // integration interval for both filter
    schemes
uint32_t lastUpdate = 0;      // used to calculate integration interval
uint32_t Now = 0;             // used to calculate integration interval


/* CONNECTION TO THE WIFI */

const char* ssid     = "glove";
const char* password = "blackglove";

const char* host = "192.168.2.110";

bool isConnected = false;
APPLEMIDI_CREATE_INSTANCE(WiFiUDP, AppleMIDI); // see definition in
    AppleMidi_Defs.h

```

```
50 bool poweredon=false;
51
52
53 /* VARIABLES FOR MIDI MANAGEMENT */
54
55 float volumeincreasing=0;
56 int oldnote=0;
57 int notetoplay=0;
58 double prevelabroll=0;
59
60 int numOfNotes=6;
61 int zeroPosition[16]={54, 54, 36, 25, 20, 54, 48, 40.5, 36, 32.5, 50, 4
       5, 42.5, 56, 52.5, 49};
62 int angle[16]={216, 108, 72, 50, 40, 36, 32, 27, 24, 21, 20, 18, 17, 16
       , 15, 14};
63
64 bool last=false;
65
66 #define N 30
67
68 float velocityvector[N]={0};
69 float velocity=0;
70 float prevvelocity=0;
71
72 int preinchord=1;
73 int inchord=1;
74
75
76 /* VARIABLES FOR DATA */
77
78 float abias[3] = {0, 0, 0}, gbias[3] = {0, 0, 0};
79 float ax, ay, az, gx, gy, gz, mx, my, mz; // variables to hold latest
       sensor data values
80 float q[4] = {1.0f, 0.0f, 0.0f, 0.0f};    // vector to hold quaternion
81
82 double curraccx=G;
83 double curraccy=G;
84 double curraccz=G;
85 double curracc, oldacc;
86
87 float pitch, yaw, roll;
88
89
90
91 void setup()
92 {
93   Serial.begin(115200);
94
95
96   /* CONNECTION TO THE WIFI */
97
98   Serial.print("Getting IP address...");
99   WiFi.begin(ssid, password);
100  while (WiFi.status() != WL_CONNECTED) {
```

```
101       delay(500);
102       Serial.print(".");
103     }
104     Serial.println("");
105     Serial.println("WiFi connected");
106     Serial.println();
107     Serial.print("IP address is ");
108     Serial.println(WiFi.localIP());
109
110
111     /* CONNECTION TO RTPMIDI */
112
113     Serial.println("OK, now make sure you an rtpMIDI session that is
          Enabled");
114     Serial.print("Add device named Arduino with Host/Port ");
115     Serial.print(WiFi.localIP());
116     Serial.println(":5004");
117     Serial.println("Then press the Connect button");
118     Serial.println("Then open a MIDI listener (eg MIDI-OX) and monitor
          incoming notes");
119
120     // Create a session and wait for a remote host to connect to us
121     AppleMIDI.begin("test");
122
123     AppleMIDI.OnConnected(OnAppleMidiConnected);
124     AppleMIDI.OnDisconnected(OnAppleMidiDisconnected);
125
126     AppleMIDI.OnReceiveNoteOn(OnAppleMidiNoteOn);
127     AppleMIDI.OnReceiveNoteOff(OnAppleMidiNoteOff);
128
129
130     /* DEVICE INITIALIZATION */
131
132     uint32_t status = dof.begin();
133
134     delay(1000);
135     dof.setAccelScale(dof.A_SCALE_4G);
136     //delay(500);
137     dof.setGyroScale(dof.G_SCALE_245DPS);
138     //delay(500);
139     dof.setMagScale(dof.M_SCALE_2GS);
140     //delay(500);
141     dof.setAccelODR(dof.A_ODR_200);
142     //delay(500);
143     dof.setAccelABW(dof.A_ABW_50);
144     //delay(500);
145     dof.setGyroODR(dof.G_ODR_190_BW_125);
146     //delay(500);
147     dof.setMagODR(dof.M_ODR_125);
148     //delay(500);
149     delay(1000);
150     dof.calLSM9DS0(gbias, abias);
151
152 }
```

```
153
154
155
156
157 void loop()
158 {
159
160   //Check for incoming signals
161   AppleMIDI.run();
162
163   if (isConnected && poweredon){
164
165     /* READING DATA AND PRODUCING INFO FROM DEVICE */
166     dof.readGyro();
167     gx = dof.calcGyro(dof.gx) - gbias[0];
168     gy = dof.calcGyro(dof.gy) - gbias[1];
169     gz = dof.calcGyro(dof.gz) - gbias[2];
170
171     dof.readAccel();
172     ax = dof.calcAccel(dof.ax) - abias[0];
173     ay = dof.calcAccel(dof.ay) - abias[1];
174     az = dof.calcAccel(dof.az) - abias[2];
175
176     dof.readMag();
177     mx = dof.calcMag(dof.mx);
178     my = dof.calcMag(dof.my);
179     mz = dof.calcMag(dof.mz);
180
181     /* COMPUTING INTEGRATION TIME AND APPLYING THE FILTERING */
182     Now = micros();
183     deltat = ((Now - lastUpdate)/1000000.0f); // set integration time
      by time elapsed since last filter update
184     lastUpdate = Now;
185     MadgwickQuaternionUpdate(ax, ay, az, gx*PI/180.0f, gy*PI/180.0f, gz
      *PI/180.0f, mx, my, mz);
186
187     /* DERIVING AVIONIC PARAMETERS */
188     yaw   = atan2(2.0f * (q[1] * q[2] + q[0] * q[3]), q[0] * q[0] + q[1
      ] * q[1] - q[2] * q[2] - q[3] * q[3]);
189     pitch = -asin(2.0f * (q[1] * q[3] - q[0] * q[2]));
190     roll  = atan2(2.0f * (q[0] * q[1] + q[2] * q[3]), q[0] * q[0] - q[1
      ] * q[1] - q[2] * q[2] + q[3] * q[3]);
191     pitch *= 180.0f / PI;
192     yaw   *= 180.0f / PI;
193     yaw   -= 2.2; // MODIFIED FOR TURIN Declination at Danville,
      California is 13 degrees 48 minutes and 47 seconds on 2014-04-04
194     roll  *= 180.0f / PI;
195
196     /* DERIVING DIRECTIONAL AND RESULTANT ACCELERATIONS */
197     curraccx=ax*G;
198     curraccy=ay*G;
199     curraccz=az*G;
200     curracc= sqrt(curraccx*curraccx+curraccy*curraccy+curraccz*curraccz
      );
```

```
201
202
203
204     /* BEHAVIOUR DEFINING */
205
206     /* DERIVING THE NOTE */
207     double elabroll;
208     elabroll=roll+zeroPosition[numOfNotes-1];
209     if (elabroll < numOfNotes*angle[numOfNotes-1] && elabroll > 0){
210       if (elabroll < prevelabroll*angle[numOfNotes-1]-2.5 || elabroll >
       (1+prevelabroll)*angle[numOfNotes-1]+2.5){
211         elabroll=elabroll/angle[numOfNotes-1];
212         elabroll=floor(elabroll);
213         prevelabroll=elabroll;
214         notetoplay=60+elabroll;
215       }
216     }
217
218
219     /* COMPUTING THE VELOCITY AND SENDING THE MIDI MESSAGES */
220     float rollfactor=1;
221     float maxfactor=1.2;
222     float minfactor=1;
223     rollfactor=maxfactor-(maxfactor-minfactor)*(abs(90-abs(roll))/90);
224
225     float linearfactor=0.75;
226     float lineardecfactor=0.6;
227
228     if (notetoplay != oldnote){
229       int need=oldnote-60;
230       for (int i=0; i<preinchord; i++){
231         AppleMIDI.noteOff(need+60+i*16, 1, 1);
232         delay(20);
233       }
234       preinchord=inchord;
235       for (int i=0; i<inchord; i++){
236         AppleMIDI.noteOn(elabroll+60+i*16, velocity,1);
237         delay(20);
238       }
239
240
241     }
242     else{
243
244       if (floor((curracc*linearfactor*rollfactor)-G) >=0){
245           if (volumeincreasing+floor((curracc*linearfactor*rollfactor)-
       G) < 128 ) {
246             volumeincreasing=volumeincreasing+floor((curracc*
       linearfactor*rollfactor)-G);
247           }
248           else{
249             volumeincreasing=127;
250           }
251           last=false;
```

```
252       }
253       else{
254         if (volumeincreasing - lineardecfactor * (volumeincreasing / 12
      7) > 0) {
255           volumeincreasing = volumeincreasing - lineardecfactor * (
      volumeincreasing / 127);
256           last=false;
257         }
258         else{
259           volumeincreasing=0;
260         }
261       }
262       prevvelocity=floor(velocity);
263       velocity=averagingS(volumeincreasing, velocity, N);
264 
265       float velocityonless=averagingS(volumeincreasing, velocity, 25);
266 
267       if abs(velocityonless-velocity > 4) {
268         velocity=velocityonless;
269       }
270 
271       if (prevvelocity != floor(velocity)){
272         AppleMIDI.controlChange(7, floor(velocity), 1);
273       }
274     }
275     oldnote=notetoplay;
276   }
277 }
278 
279 
280 float averagingS(float Vi, float pastV, int numel){
281   float ret=pastV*numel-pastV;
282   ret=ret+Vi;
283   ret=ret/numel;
284   return ret;
285 
286 }
287 
288 
289 
290 /* Event handlers for incoming MIDI messages */
291 
292 
293 void OnAppleMidiConnected(uint32_t ssrc, char* name) {
294   isConnected  = true;
295   Serial.print("Connected to session ");
296   Serial.println(name);
297 }
298 
299 
300 void OnAppleMidiDisconnected(uint32_t ssrc) {
301   isConnected  = false;
302   Serial.println("Disconnected");
303   resetFunc(); //call reset
```

```
304 }
305
306
307 void OnAppleMidiNoteOn(byte channel, byte note, byte velocity) {
308
309   Serial.print("velocity:");
310   Serial.print(velocity);
311   Serial.println();
312   if (velocity > 0 && velocity <= 16){
313     numOfNotes=velocity;
314     prevelabroll=0;
315     int need=oldnote-60;
316     for (int i=0; i<preinchord; i++){
317       AppleMIDI.noteOff(need+60+i*16, 1, 1);
318       delay(20);
319     }
320     oldnote=0;
321     notetoplay=0;
322     AppleMIDI.controlChange(9,velocity,1);
323   }
324   else if (velocity == 88){
325     poweredon=true;
326     AppleMIDI.controlChange(9,88,1);
327   }
328   else if (velocity == 89){
329     poweredon=false;
330     int need=oldnote-60;
331     for (int i=0; i<preinchord; i++){
332       AppleMIDI.noteOff(need+60+i*16, 1, 1);
333       delay(20);
334     }
335     prevelabroll=0;
336     oldnote=0;
337     notetoplay=0;
338     velocity=0;
339     prevvelocity=0;
340     AppleMIDI.controlChange(9,89,1);
341   }
342   else if (velocity > 89 && velocity < 94) {
343     inchord=velocity-89;
344     AppleMIDI.controlChange(9,velocity,1);
345   }
346
347
348 }
349
350
351 void OnAppleMidiNoteOff(byte channel, byte note, byte velocity) {
352   Serial.print("Incoming NoteOff from channel:");
353   Serial.print(channel);
354   Serial.print(" note:");
355   Serial.print(note);
356   Serial.print(" velocity:");
357   Serial.print(velocity);
```

```
358   Serial.println();
359 }
360
361
362 /* MADGWICK FILTER FUNCTION */
363
364 void MadgwickQuaternionUpdate(float ax, float ay, float az, float gx,
        float gy, float gz, float mx, float my, float mz)
365 {
366     float q1 = q[0], q2 = q[1], q3 = q[2], q4 = q[3];    // short name
        local variable for readability
367     float norm;
368     float hx, hy, _2bx, _2bz;
369     float s1, s2, s3, s4;
370     float qDot1, qDot2, qDot3, qDot4;
371
372     // Auxiliary variables to avoid repeated arithmetic
373     float _2q1mx;
374     float _2q1my;
375     float _2q1mz;
376     float _2q2mx;
377     float _4bx;
378     float _4bz;
379     float _2q1 = 2.0f * q1;
380     float _2q2 = 2.0f * q2;
381     float _2q3 = 2.0f * q3;
382     float _2q4 = 2.0f * q4;
383     float _2q1q3 = 2.0f * q1 * q3;
384     float _2q3q4 = 2.0f * q3 * q4;
385     float q1q1 = q1 * q1;
386     float q1q2 = q1 * q2;
387     float q1q3 = q1 * q3;
388     float q1q4 = q1 * q4;
389     float q2q2 = q2 * q2;
390     float q2q3 = q2 * q3;
391     float q2q4 = q2 * q4;
392     float q3q3 = q3 * q3;
393     float q3q4 = q3 * q4;
394     float q4q4 = q4 * q4;
395
396     // Normalise accelerometer measurement
397     norm = sqrt(ax * ax + ay * ay + az * az);
398     if (norm == 0.0f) return; // handle NaN
399     norm = 1.0f/norm;
400     ax *= norm;
401     ay *= norm;
402     az *= norm;
403
404     // Normalise magnetometer measurement
405     norm = sqrt(mx * mx + my * my + mz * mz);
406     if (norm == 0.0f) return; // handle NaN
407     norm = 1.0f/norm;
408     mx *= norm;
409     my *= norm;
```

```
410     mz *= norm;
411
412     // Reference direction of Earth's magnetic field
413     _2q1mx = 2.0f * q1 * mx;
414     _2q1my = 2.0f * q1 * my;
415     _2q1mz = 2.0f * q1 * mz;
416     _2q2mx = 2.0f * q2 * mx;
417     hx = mx * q1q1 - _2q1my * q4 + _2q1mz * q3 + mx * q2q2 + _2q2 * my
        * q3 + _2q2 * mz * q4 - mx * q3q3 - mx * q4q4;
418     hy = _2q1mx * q4 + my * q1q1 - _2q1mz * q2 + _2q2mx * q3 - my * q2q
        2 + my * q3q3 + _2q3 * mz * q4 - my * q4q4;
419     _2bx = sqrt(hx * hx + hy * hy);
420     _2bz = -_2q1mx * q3 + _2q1my * q2 + mz * q1q1 + _2q2mx * q4 - mz *
        q2q2 + _2q3 * my * q4 - mz * q3q3 + mz * q4q4;
421     _4bx = 2.0f * _2bx;
422     _4bz = 2.0f * _2bz;
423
424     // Gradient decent algorithm corrective step
425     s1 = -_2q3 * (2.0f * q2q4 - _2q1q3 - ax) + _2q2 * (2.0f * q1q2 + _2
        q3q4 - ay) - _2bz * q3 * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4
         - q1q3) - mx) + (-_2bx * q4 + _2bz * q2) * (_2bx * (q2q3 - q1q4) +
        _2bz * (q1q2 + q3q4) - my) + _2bx * q3 * (_2bx * (q1q3 + q2q4) + _2
        bz * (0.5f - q2q2 - q3q3) - mz);
426     s2 = _2q4 * (2.0f * q2q4 - _2q1q3 - ax) + _2q1 * (2.0f * q1q2 + _2q
        3q4 - ay) - 4.0f * q2 * (1.0f - 2.0f * q2q2 - 2.0f * q3q3 - az) + _2
        bz * q4 * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx)
        + (_2bx * q3 + _2bz * q1) * (_2bx * (q2q3 - q1q4) + _2bz * (q1q2 + q
        3q4) - my) + (_2bx * q4 - _4bz * q2) * (_2bx * (q1q3 + q2q4) + _2bz
        * (0.5f - q2q2 - q3q3) - mz);
427     s3 = -_2q1 * (2.0f * q2q4 - _2q1q3 - ax) + _2q4 * (2.0f * q1q2 + _2
        q3q4 - ay) - 4.0f * q3 * (1.0f - 2.0f * q2q2 - 2.0f * q3q3 - az) +
        (-_4bx * q3 - _2bz * q1) * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2
        q4 - q1q3) - mx) + (_2bx * q2 + _2bz * q4) * (_2bx * (q2q3 - q1q4) +
         _2bz * (q1q2 + q3q4) - my) + (_2bx * q1 - _4bz * q3) * (_2bx * (q1q
        3 + q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
428     s4 = _2q2 * (2.0f * q2q4 - _2q1q3 - ax) + _2q3 * (2.0f * q1q2 + _2q
        3q4 - ay) + (-_4bx * q4 + _2bz * q2) * (_2bx * (0.5f - q3q3 - q4q4)
        + _2bz * (q2q4 - q1q3) - mx) + (-_2bx * q1 + _2bz * q3) * (_2bx * (q
        2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) + _2bx * q2 * (_2bx * (q1q3
         + q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
429     norm = sqrt(s1 * s1 + s2 * s2 + s3 * s3 + s4 * s4);    // normalise
         step magnitude
430     norm = 1.0f/norm;
431     s1 *= norm;
432     s2 *= norm;
433     s3 *= norm;
434     s4 *= norm;
435
436     // Compute rate of change of quaternion
437     qDot1 = 0.5f * (-q2 * gx - q3 * gy - q4 * gz) - beta * s1;
438     qDot2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy) - beta * s2;
439     qDot3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx) - beta * s3;
440     qDot4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx) - beta * s4;
441
```

```
442     // Integrate to yield quaternion
443     q1 += qDot1 * deltat;
444     q2 += qDot2 * deltat;
445     q3 += qDot3 * deltat;
446     q4 += qDot4 * deltat;
447     norm = sqrt(q1 * q1 + q2 * q2 + q3 * q3 + q4 * q4);    // normalise
        quaternion
448     norm = 1.0f/norm;
449     q[0] = q1 * norm;
450     q[1] = q2 * norm;
451     q[2] = q3 * norm;
452     q[3] = q4 * norm;
453
454 }
```

Listing 25: B-Glove full code

Instruments Manager full code

```
1  Imports System
2  Imports System.IO.Ports
3  Imports System.Threading
4  Imports System.Runtime.InteropServices
5  Imports Toub.Sound.Midi
6  Imports Microsoft.VisualBasic.Constants
7  Imports System.Text
8  Imports NativeWifi
9
10
11
12 Public Class Form1
13     'Acks waiting time
14     Public Const timerSeconds = 3
15
16     'States of the software
17     Public Enum instrState
18         pOn
19         pOff
20         WaitingToOff
21         WaitingToOn
22     End Enum
23
24     'Structure holding a live set
25     Private Structure song
26         Public Name As String
27         Public NON As Integer
28         Public IC As Integer
29         Public ToDel As Boolean
30     End Structure
31
32     'Structure for MIDI devices
33     Public Structure MIDIINCAPS
34         Dim wMid As Int16 ' Manufacturer ID
35         Dim wPid As Int16 ' Product ID
36         Dim vDriverVersion As Integer ' Driver version
37         <MarshalAs(UnmanagedType.ByValTStr, SizeConst:=32)> Dim szPname As String
       ' Product Name
38         Dim dwSupport As Integer ' Reserved
39     End Structure
40
41     'MIDI management functions
42     Public Declare Function midiInGetNumDevs Lib "winmm.dll" () As Integer
43     Public Declare Function midiInGetDevCaps Lib "winmm.dll" Alias "
       midiInGetDevCapsA" (ByVal uDeviceID As Integer, ByRef lpCaps As MIDIINCAPS,
       ByVal uSize As Integer) As Integer
44     Public Declare Function midiInOpen Lib "winmm.dll" (ByRef hMidiIn As Integer,
       ByVal uDeviceID As Integer, ByVal dwCallback As MidiInCallback, ByVal
       dwInstance As Integer, ByVal dwFlags As Integer) As Integer
45     Public Declare Function midiInStart Lib "winmm.dll" (ByVal hMidiIn As Integer)
        As Integer
46     Public Declare Function midiInStop Lib "winmm.dll" (ByVal hMidiIn As Integer)
       As Integer
47     Public Declare Function midiInReset Lib "winmm.dll" (ByVal hMidiIn As Integer)
        As Integer
48     Public Declare Function midiInClose Lib "winmm.dll" (ByVal hMidiIn As Integer)
        As Integer
49     Public Delegate Sub DisplayDataDelegate(dwParam1)
50     Public Declare Function midiOutGetNumDevs Lib "winmm.dll" () As Integer
```

```
51      Public Declare Function midiOutGetDevCaps Lib "winmm.dll" Alias "
        midiOutGetDevCapsA" (ByVal uDeviceID As Integer, ByRef lpMidiOutCaps As
        MIDIINCAPS, ByVal cbMidiOutCaps As Integer) As Integer
52      Public Delegate Function MidiInCallback(ByVal hMidiIn As Integer, ByVal wMsg
        As UInteger, ByVal dwInstance As Integer, ByVal dwParam1 As Integer, ByVal
        dwParam2 As Integer) As Integer
53      Public ptrCallback As New MidiInCallback(AddressOf MidiInProc)
54      Public Const CALLBACK_FUNCTION As Integer = &H30000
55      Public Const MIDI_IO_STATUS = &H20
56
57      'Holds the state of the software
58      Dim poweredon As instrState = instrState.pOff
59
60      'Path of the softeare and of the folder containing the Live Sets
61      Dim AppFolderPath As String = IO.Path.Combine(IO.Directory.GetParent(
        Application.ExecutablePath).FullName, "")
62      Dim FileFolderPath As String = IO.Path.Combine(IO.Directory.GetParent(
        Application.ExecutablePath).FullName, "AblFiles\ ")
63
64      'Acks management
65      Dim waitingForNotesAck = False
66      Dim waitingForInChordAck = False
67      Dim timeTimer As Integer
68
69      'MIDI management
70      Dim hMidiIn As Integer
71      Dim hmo As Integer
72      Dim numberOfNotes As Integer = 0
73      Dim fromLive As Boolean = False
74      Dim StatusByte As Byte
75      Dim DataByte1 As Byte
76      Dim DataByte2 As Byte
77      Dim MonitorActive As Boolean = False
78
79      'Structure of the live sets
80      Dim allsongs() As song
81
82
83
84
85
86
87      Function MidiInProc(ByVal hMidiIn As Integer, ByVal wMsg As UInteger, ByVal
        dwInstance As Integer, ByVal dwParam1 As Integer, ByVal dwParam2 As Integer)
        As Integer
88          If MonitorActive = True Then
89              txtMonitorIN.Invoke(New DisplayDataDelegate(AddressOf DisplayData),
        New Object() {dwParam1})
90          End If
91      End Function
92
93      Private Sub DisplayData(dwParam1)
94          StatusByte = (dwParam1 And &HFF)
95          DataByte1 = (dwParam1 And &HFF00) >> 8
96          DataByte2 = (dwParam1 And &HFF0000) >> 16
97
98
99          If poweredon = instrState.WaitingToOn And StatusByte = 176 And DataByte1 =
         9 And DataByte2 = 88 Then
100             txtMonitorIN.AppendText("Start" & vbCrLf)
101             poweredon = instrState.pOn
102             btnPower.BackgroundImage = Image.FromFile("./images/ON.png")
```

```
103             Me.Height = 539
104             Me.Width = 670
105 
106         ElseIf poweredon = instrState.WaitingToOff And StatusByte = 176 And
    DataByte1 = 9 And DataByte2 = 89 Then
107             txtMonitorIN.AppendText("Stop" & vbCrLf)
108             poweredon = instrState.pOff
109             btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
110             Me.Height = 302
111             Me.Width = 670
112 
113         ElseIf poweredon = instrState.pOn And StatusByte = 176 And DataByte1 = 7
    Then
114             txtMonitorIN.AppendText("Control-7 Change: " & DataByte2 & vbCrLf)
115             ProgressBar1.Value = DataByte2
116             lblVelocity.Text = DataByte2
117 
118         ElseIf poweredon = instrState.pOn And StatusByte = 144 Then
119             txtMonitorIN.AppendText("Note " & DataByte1 - 59 & " On, Velocity: " &
     DataByte2 & vbCrLf)
120             ProgressBar1.Value = DataByte2
121             lblVelocity.Text = DataByte2
122             lblNote.Text = (DataByte1 - 59) Mod 16
123 
124         ElseIf poweredon = instrState.pOn And StatusByte = 128 Then
125             txtMonitorIN.AppendText("Note " & DataByte1 - 59 & " Off" & vbCrLf)
126 
127         ElseIf waitingForNotesAck = True And StatusByte = 176 And DataByte1 = 9
    And DataByte2 > 0 And DataByte2 <= 16 Then
128             waitingForNotesAck = False
129             txtMonitorIN.AppendText("Received Ack for number of notes" & vbCrLf)
130             Timer1.Stop()
131             If fromLive = True Then
132                 fromLive = False
133                 If (System.Diagnostics.Process.GetProcessesByName("Ableton Live 9
    Suite").Length > 0) Then
134                     System.Diagnostics.Process.GetProcessesByName("Ableton Live 9
    Suite")(0).Close()
135                 End If
136                 System.Diagnostics.Process.Start(FileFolderPath & comboLiveSet.
    SelectedItem & " Project\ " & comboLiveSet.SelectedItem & ".als")
137             End If
138         ElseIf waitingForNotesAck = True And StatusByte = 176 And DataByte1 = 9
    And DataByte2 > 89 And DataByte2 <= 94 Then
139             waitingForInChordAck = False
140             txtMonitorIN.AppendText("Received Ack for notes in chord" & vbCrLf)
141 
142         End If
143 
144         If poweredon = instrState.pOff Then
145             midiInStop(hMidiIn)
146             midiInClose(hMidiIn)
147             comboNetIN.Enabled = True
148             comboNetOUT.Enabled = True
149             MonitorActive = False
150         End If
151 
152     End Sub
153 
154     Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles Me.Load
155 
```

```
156         Me.Height = 302
157         Me.Width = 670
158         Me.Show()
159 
160         If (System.Diagnostics.Process.GetProcessesByName("rtpMIDI").Length <= 0
    Or System.Diagnostics.Process.GetProcessesByName("midiox").Length <= 0) Then
161             MsgBox("Before starting this program, MIDI-OX and rtpMIDI have to be
    opened.")
162             End
163         End If
164 
165         picClearIN.Parent = txtMonitorIN
166         picClearOUT.Parent = txtMonitorOUT
167         picClearIN.Location = New Point(252, 0)
168         picClearOUT.Location = New Point(252, 0)
169 
170         noteSelection.SelectedItem = noteSelection.Items(10)
171         inChord.SelectedItem = inChord.Items(3)
172 
173         'LOAD OF FILES
174         For Each s As String In System.IO.Directory.GetDirectories(FileFolderPath,
     "* Project")
175             Dim provs() As String = s.Split("\ ".ToCharArray, StringSplitOptions.
    RemoveEmptyEntries)
176             comboLiveSet.Items.Add(provs(provs.Length - 1).Split(" ".ToCharArray,
    StringSplitOptions.RemoveEmptyEntries)(0))
177         Next
178 
179 
180         Dim testo As String = ""
181 
182         'ALREADY NUMBERED FILES
183         testo = My.Computer.FileSystem.ReadAllText(AppFolderPath & "\NumOfNotes.
    txt")
184 
185         Dim rows() As String = testo.Split(vbCrLf.ToCharArray, StringSplitOptions.
    RemoveEmptyEntries)
186         Dim songs(rows.Length - 1) As song
187         Dim c As Integer = 0
188         For Each row As String In rows
189             Dim data() As String = row.Split(" ".ToCharArray, StringSplitOptions.
    RemoveEmptyEntries)
190             If data.Length = 3 Then
191                 songs(c).Name = data(0)
192                 songs(c).NON = data(1)
193                 songs(c).IC = data(2)
194                 songs(c).ToDel = False
195                 c = c + 1
196             End If
197         Next
198 
199         Dim i As Integer = 0
200         Dim nmore As Integer = 0
201         Dim songsmore(100) As song
202 
203 
204         If Not (testo = "") Then
205 
206 
207             'CHECK IF STILL PRESENT
208             For Each song As song In songs
209                 If Not (comboLiveSet.Items.Contains(song.Name)) Then
```

```
210                     If MsgBox(song.Name & " non è ùpi presente tra i file Ableton.
    Eliminare?", vbYesNo) = 6 Then
211                         song.ToDel = True
212                     End If
213                 End If
214             Next
215 
216             'CHECK FOR NEW PRESENCES
217 
218             For Each name As String In comboLiveSet.Items
219                 Dim found As Boolean = False
220                 For i = 0 To songs.Length - 1
221                     If songs(i).Name = name Then
222                         found = True
223                         Exit For
224                     End If
225                 Next
226                 If found = False Then
227                     songsmore(nmore).Name = name
228                     Do
229                         Dim s As String = InputBox("New Live set: " & name & "." &
    vbCrLf & "How many notes/chords ( 1 - 16 )?")
230                         If IsNumeric(s) Then
231                             If s > 0 And s < 17 Then
232                                 songsmore(nmore).NON = CInt(s)
233                             End If
234                         End If
235                         If songsmore(nmore).NON > 1 Then
236                             s = InputBox("How many notes per chord ( 1 - 4 )?")
237                             If IsNumeric(s) Then
238                                 If s > 0 And s < 5 Then
239                                     songsmore(nmore).IC = CInt(s)
240                                 End If
241                             End If
242                         Else
243                             songsmore(nmore).IC = 0
244                         End If
245                     Loop Until songsmore(nmore).NON > 0
246                     nmore = nmore + 1
247                 End If
248             Next
249 
250         Else
251 
252             'ADDING NEW SONGS
253             For Each name As String In comboLiveSet.Items
254                 songsmore(nmore).Name = name
255                 Do
256                     Dim s As String = InputBox("Nuova canzone: " & name & ".
    Inserisci numero di note ( 1 - 16 ).")
257                     If IsNumeric(s) Then
258                         If s > 0 And s < 17 Then
259                             songsmore(nmore).NON = CInt(s)
260                         End If
261                     End If
262                     If songsmore(nmore).NON > 1 Then
263                         s = InputBox("How many notes per chord ( 1 - 4 )?")
264                         If IsNumeric(s) Then
265                             If s > 0 And s < 5 Then
266                                 songsmore(nmore).IC = CInt(s)
267                             End If
268                         End If
```

```
                    Else
                        songsmore(nmore).IC = 0
                    End If
                Loop Until songsmore(nmore).NON > 0
                nmore = nmore + 1
            Next


        End If


        'COMPOSING OF THE FINAL STRUCTURE
        Dim dimension As Integer = 0
        If testo = "" Then
            dimension = nmore
        Else
            dimension = nmore + songs.Length
        End If
        ReDim allsongs(dimension - 1)

        c = 0

        If Not (testo = "") Then
            For i = 0 To songs.Length - 1
                If songs(i).ToDel = False Then
                    allsongs(c) = songs(i)
                    c = c + 1
                End If
            Next
        End If
        For j As Integer = c To nmore + c - 1
            allsongs(j) = songsmore(j - c)
        Next




        'MIDI STUFF


        If midiInGetNumDevs() = 0 Then
            MsgBox("No MIDI devices connected")
            'Application.Exit()
        End If

        Dim InCaps As New MIDIINCAPS
        Dim DevCnt As Integer

        For DevCnt = 0 To (midiInGetNumDevs - 1)
            midiInGetDevCaps(DevCnt, InCaps, Len(InCaps))
            comboNetIN.Items.Add(InCaps.szPname)
        Next DevCnt


        'OUTPUT
        If midiOutGetNumDevs() = 0 Then
            MsgBox("No MIDI devices connected")
            'Application.Exit()
        End If
        For DevCnt = 0 To (midiInGetNumDevs - 1)
            midiOutGetDevCaps(DevCnt, InCaps, Len(InCaps))
            comboNetOUT.Items.Add(InCaps.szPname)
```

```
331         Next DevCnt
332 
333     End Sub
334 
335 
336 
337 
338     Private Sub Form1_FormClosed(ByVal sender As Object, ByVal e As System.Windows
        .Forms.FormClosedEventArgs) Handles Me.FormClosed
339 
340         Dim DeviceID As Integer = comboNetOUT.SelectedIndex
341         Dim device As MidiInterop.MidiDeviceHandle = MidiInterop.OpenMidiOut(
        DeviceID)
342         Dim output As Integer = 89
343         output = Convert.ToInt32(Hex(output) & "3D90", 16)
344         MidiInterop.SendMidiMessage(device, output)
345         device.Close()
346 
347         MonitorActive = False
348         midiInStop(hMidiIn)
349         midiInReset(hMidiIn)
350         'midiInClose(hMidiIn)
351 
352         'WRITING FINAL FILE
353         Dim textToWrite As String = ""
354         For Each song As song In allsongs
355             textToWrite = textToWrite & song.Name & " " & song.NON & " " & song.IC
         & vbCrLf
356         Next
357         My.Computer.FileSystem.WriteAllText(AppFolderPath & "\NumOfNotes.txt",
        textToWrite, False)
358 
359         Dim stopN As New ProcessStartInfo
360 
361         stopN.FileName = "cmd.exe"
362         stopN.Arguments = "/c netsh wlan stop hostednetwork"
363         stopN.UseShellExecute = False
364         stopN.CreateNoWindow = True
365 
366         Process.Start(stopN)
367 
368         Application.Exit()
369     End Sub
370 
371 
372     Private Sub Button7_Click(sender As Object, e As EventArgs) Handles btnPower.
        Click
373 
374         If comboNetOUT.SelectedIndex = -1 Or comboNetIN.SelectedIndex = -1 Then
375             MsgBox("Select the networks.")
376         Else
377             Dim DeviceID As Integer = comboNetOUT.SelectedIndex
378             Dim output As Integer
379             Dim ss As String = "Start"
380 
381             Dim nomessage As Boolean = False
382 
383             If poweredon = instrState.pOff Then
384                 output = 88
385                 ss = "Start"
386                 poweredon = instrState.WaitingToOn
387                 btnPower.BackgroundImage = Image.FromFile("./images/WAIT.png")
```

```
388
389             ElseIf poweredon = instrState.pOn Then
390                 ss = "Stop"
391                 output = 89
392                 poweredon = instrState.WaitingToOff
393                 btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
394
395             Else
396                 nomessage = True
397                 poweredon = instrState.pOff
398                 btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
399                 Me.Height = 302
400                 Me.Width = 670
401             End If
402
403             If nomessage = False Then
404                 Dim device As MidiInterop.MidiDeviceHandle = MidiInterop.
    OpenMidiOut(DeviceID)
405                 output = Convert.ToInt32(Hex(output) & "3D90", 16)
406                 MidiInterop.SendMidiMessage(device, output)
407                 txtMonitorOUT.AppendText(ss & " command sent" & vbCrLf)
408                 device.Close()
409             End If
410
411             If poweredon = instrState.WaitingToOn Or poweredon = instrState.
    WaitingToOff Then
412                 DeviceID = comboNetIN.SelectedIndex
413                 comboNetIN.Enabled = False
414                 comboNetOUT.Enabled = False
415                 midiInOpen(hMidiIn, DeviceID, ptrCallback, 0, CALLBACK_FUNCTION Or
     MIDI_IO_STATUS)
416                 midiInStart(hMidiIn)
417                 MonitorActive = True
418             Else
419                 midiInStop(hMidiIn)
420                 midiInClose(hMidiIn)
421                 comboNetIN.Enabled = True
422                 comboNetOUT.Enabled = True
423                 MonitorActive = False
424             End If
425
426         End If
427
428     End Sub
429
430     Private Sub btnPower_BackgroundImageChanged(sender As Object, e As EventArgs)
    Handles btnPower.BackgroundImageChanged
431         If poweredon = instrState.WaitingToOn Or poweredon = instrState.
    WaitingToOff Then
432             timeTimer = timerSeconds
433             Timer1.Start()
434         Else
435             Timer1.Stop()
436         End If
437
438     End Sub
439
440     Private Sub Button7_MouseDown(sender As Object, e As MouseEventArgs) Handles
    btnPower.MouseDown
441         If poweredon = instrState.pOn Then
442             btnPower.BackgroundImage = Image.FromFile("./images/MOUSEON.png")
443         ElseIf poweredon = instrState.pOff Then
```

```
444             btnPower.BackgroundImage = Image.FromFile("./images/MOUSEOFF.png")
445         Else
446             btnPower.BackgroundImage = Image.FromFile("./images/WAITs.png")
447         End If
448     End Sub
449
450     Private Sub Button7_MouseUp(sender As Object, e As MouseEventArgs) Handles
        btnPower.MouseUp
451         If poweredon = instrState.pOn Then
452             btnPower.BackgroundImage = Image.FromFile("./images/ON.png")
453         ElseIf poweredon = instrState.pOff Then
454             btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
455         Else
456             btnPower.BackgroundImage = Image.FromFile("./images/WAIT.png")
457         End If
458     End Sub
459
460     Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
461         If timeTimer = 0 Then
462             Timer1.Stop()
463             poweredon = instrState.pOff
464             btnPower.BackgroundImage = Image.FromFile("./images/OFF.png")
465             Me.Height = 302
466             Me.Width = 670
467             midiInStop(hMidiIn)
468             midiInClose(hMidiIn)
469             comboNetIN.Enabled = True
470             comboNetOUT.Enabled = True
471             MonitorActive = False
472             MsgBox("The istrument is not reachable. Please check the rtpMIDI
        connection.")
473         Else
474             timeTimer = timeTimer - 1
475         End If
476
477     End Sub
478
479
480     Private Sub Button5_Click(sender As Object, e As EventArgs) Handles
        btnSendNotes.Click
481
482         numberOfNotes = noteSelection.Text
483         waitingForNotesAck = True
484         waitingForInChordAck = True
485         Dim notesinchord As Integer = 89 + inChord.Text
486
487         Dim DeviceID As Integer = comboNetOUT.SelectedIndex
488         Dim device As MidiInterop.MidiDeviceHandle = MidiInterop.OpenMidiOut(
        DeviceID)
489         Dim output As Integer
490
491         txtMonitorOUT.AppendText("Notes in chord has been sent" & vbCrLf)
492         output = Convert.ToInt32(Hex(notesinchord) & "3D90", 16)
493         MidiInterop.SendMidiMessage(device, output)
494
495
496         txtMonitorOUT.AppendText("Number of notes has been sent" & vbCrLf)
497         output = Convert.ToInt32(Hex(numberOfNotes) & "3D90", 16)
498         MidiInterop.SendMidiMessage(device, output)
499         timeTimer = timerSeconds
500         Timer1.Start()
501         device.Close()
```

```
502
503     End Sub
504
505
506     Private Sub Button4_Click(sender As Object, e As EventArgs) Handles
        btnOpenLive.Click
507
508         If lblNumberOfNotes.Text = "" Then
509             MsgBox("Select a Live Set.")
510         Else
511             Dim DeviceID As Integer = comboNetOUT.SelectedIndex
512             Dim device As MidiInterop.MidiDeviceHandle = MidiInterop.OpenMidiOut(
        DeviceID)
513             Dim output As Integer
514
515             Dim notesinchord As Integer = 89 + lblNotesIC.Text
516
517             numberOfNotes = lblNumberOfNotes.Text
518             waitingForNotesAck = True
519             fromLive = True
520             txtMonitorOUT.AppendText("Number of notes for Live set has been sent"
        & vbCrLf)
521             output = Convert.ToInt32(Hex(numberOfNotes) & "3D90", 16)
522             MidiInterop.SendMidiMessage(device, output)
523
524             txtMonitorOUT.AppendText("Notes per chord for Live set has been sent"
        & vbCrLf)
525             output = Convert.ToInt32(Hex(notesinchord) & "3D90", 16)
526             MidiInterop.SendMidiMessage(device, output)
527
528
529             timeTimer = timerSeconds
530             Timer1.Start()
531             device.Close()
532         End If
533
534     End Sub
535
536     Private Sub ComboBox3_SelectedIndexChanged(sender As Object, e As EventArgs)
        Handles comboLiveSet.SelectedIndexChanged
537         For Each song As song In allsongs
538             If song.Name = comboLiveSet.SelectedItem Then
539                 If song.IC > 1 Then
540                     Label2.Text = "Number of chords"
541                     Label12.Visible = True
542                     lblNotesIC.Visible = True
543                 Else
544                     Label2.Text = "Number of notes"
545                     Label12.Visible = False
546                     lblNotesIC.Visible = False
547                 End If
548                 lblName.Text = song.Name
549                 lblNumberOfNotes.Text = song.NON
550                 lblNotesIC.Text = song.IC
551                 Exit For
552             End If
553         Next
554     End Sub
555
556
557     Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles
        picClearOUT.Click
```

```
558         txtMonitorOUT.Clear()
559     End Sub
560 
561     Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles
        picClearIN.Click
562         txtMonitorIN.Clear()
563     End Sub
564 
565     Private Sub PictureBox1_MouseDown(sender As Object, e As MouseEventArgs)
        Handles picClearIN.MouseDown
566         picClearIN.Image = Image.FromFile("./images/sclear.png")
567     End Sub
568 
569     Private Sub PictureBox1_MouseUp(sender As Object, e As MouseEventArgs) Handles
         picClearIN.MouseUp
570         picClearIN.Image = Image.FromFile("./images/clear.png")
571     End Sub
572 
573     Private Sub PictureBox2_MouseDown(sender As Object, e As MouseEventArgs)
        Handles picClearOUT.MouseDown
574         picClearOUT.Image = Image.FromFile("./images/sclear.png")
575     End Sub
576 
577     Private Sub PictureBox2_MouseUp(sender As Object, e As MouseEventArgs) Handles
         picClearOUT.MouseUp
578         picClearOUT.Image = Image.FromFile("./images/clear.png")
579     End Sub
580 
581     Private Sub Button4_MouseDown(sender As Object, e As MouseEventArgs) Handles
        btnOpenLive.MouseDown
582         btnOpenLive.BackgroundImage = Image.FromFile("./images/OLs.png")
583     End Sub
584 
585     Private Sub Button4_MouseUp(sender As Object, e As MouseEventArgs) Handles
        btnOpenLive.MouseUp
586         btnOpenLive.BackgroundImage = Image.FromFile("./images/OL.png")
587     End Sub
588 
589     Private Sub DomainUpDown1_TextChanged(sender As Object, e As EventArgs)
        Handles noteSelection.TextChanged
590         btnSendNotes.Text = "Set the instrument for playing " & noteSelection.Text
         & " "
591         If inChord.Text > 1 Then
592             btnSendNotes.Text = btnSendNotes.Text & "chord"
593         Else
594             btnSendNotes.Text = btnSendNotes.Text & "note"
595         End If
596 
597         If noteSelection.Text > 1 Then
598             btnSendNotes.Text = btnSendNotes.Text & "s"
599         End If
600     End Sub
601 
602 
603 
604     Private Sub inChord_TextChanged(sender As Object, e As EventArgs) Handles
        inChord.TextChanged
605         btnSendNotes.Text = "Set the instrument for playing " & noteSelection.Text
         & " "
606         If inChord.Text > 1 Then
607             btnSendNotes.Text = btnSendNotes.Text & "chord"
608         Else
```

```
609             btnSendNotes.Text = btnSendNotes.Text & "note"
610         End If
611         If Not (noteSelection.Text = "") Then
612             If noteSelection.Text > 1 Then
613                 btnSendNotes.Text = btnSendNotes.Text & "s"
614             End If
615         End If
616     End Sub
617
618 End Class
```

Listing 26: Instruments Manager whole code

Processing code for graphing roll, acceleration, base-note and velocity

```
import processing.serial.*;
Serial myPort;

int numValues = 4;

float[] values = new float[numValues];
int[] min = new int[numValues];
int[] max = new int[numValues];
color[] valColor = new color[numValues];

float partH;

int xPos = 0;
boolean clearScreen = true;


void setup() {
  size(1200, 600);
  partH = height / numValues;

  myPort = new Serial(this, Serial.list()[0], 115200);
  myPort.bufferUntil('\n');

  textSize(10);

  background(0);
  noStroke();

  values[0] = 0;
  min[0] = 0;
  max[0] = 60;
  valColor[0] = color(255, 0, 0);

  values[1] = 0;
  min[1] = 0;
  max[1] = 128;
  valColor[1] = color(0, 255, 0);

  values[2] = 0;
  min[2] = 59;
  max[2] = 76;
  valColor[2] = color(0, 0, 255);

  values[3] = 0;
  min[3] = 0;
  max[3] = 360; // custom range example
  valColor[3] = color(255, 0, 255); // purple

}

```

```
51 void draw() {
52
53   if (clearScreen) {
54     background(0);
55     clearScreen = false; // reset flag
56   }
57
58   for (int i=0; i<numValues; i++) {
59     float mappedVal = map(values[i], min[i], max[i], 0, partH);
60     stroke(valColor[i]);
61     line(xPos, partH*(i+1), xPos, partH*(i+1) - mappedVal);
62     stroke(255);
63     line(0, partH*(i+1), width, partH*(i+1));
64     fill(50);
65     noStroke();
66     rect(0, partH*i+1, 70, 12);
67     fill(255);
68     text(round(values[i]), 2, partH*i+10);
69     fill(125);
70     text(max[i], 40, partH*i+10);
71   }
72   xPos++;
73   if (xPos > width) {
74     xPos = 0;
75     clearScreen = true;
76   }
77
78 }
79
80 void serialEvent(Serial myPort) {
81   try {
82     String inString = myPort.readStringUntil('\n');
83
84     if (inString != null) {
85       inString = trim(inString);
86       values = float(splitTokens(inString, ", \t"));
87     }
88   }
89   catch(RuntimeException e) {
90     e.printStackTrace();
91   }
92 }
```

Listing 27: Processing code for graphing

Arduino Due code - initial tests

```
#include <SPI.h> // Included for SFE_LSM9DS0 library
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include "MIDIUSB.h"
#include <SFE_LSM9DS0.h>

#define LSM9DS0_XM  0x1D // Would be 0x1E if SDO_XM is LOW
#define LSM9DS0_G   0x6B // Would be 0x6A if SDO_G is LOW
LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);

#define beta sqrt(3.0f / 4.0f) * GyroMeasError   // compute beta
#define zeta sqrt(3.0f / 4.0f) * GyroMeasDrift   // compute zeta, the other free parameter in the Madgwick scheme usually set to a small or zero value
#define Kp 2.0f * 5.0f // these are the free parameters in the Mahony filter and fusion scheme, Kp for proportional feedback, Ki for integral
#define Ki 0.0f

uint32_t count = 0;  // used to control display output rate
uint32_t delt_t = 0; // used to control display output rate
float pitch, yaw, roll, heading;
float deltat = 0.0f;        // integration interval for both filter schemes
uint32_t lastUpdate = 0;    // used to calculate integration interval
uint32_t Now = 0;           // used to calculate integration interval

float abias[3] = {0, 0, 0}, gbias[3] = {0, 0, 0};
float ax, ay, az, gx, gy, gz, mx, my, mz; // variables to hold latest sensor data values
float q[4] = {1.0f, 0.0f, 0.0f, 0.0f};    // vector to hold quaternion
float eInt[3] = {0.0f, 0.0f, 0.0f};       // vector to hold integral error for Mahony method
float temperature;

int volumeincreasing=0;
double oldnote=0;

void noteOn(byte channel, byte pitch, byte velocity) {
  midiEventPacket_t noteOn = {0x09, 0x90 | channel, pitch, velocity};
  MidiUSB.sendMIDI(noteOn);
}

void noteOff(byte channel, byte pitch, byte velocity) {
  midiEventPacket_t noteOff = {0x08, 0x80 | channel, pitch, velocity};
  MidiUSB.sendMIDI(noteOff);
}

void controlChange(byte channel, byte control, byte value) {
  midiEventPacket_t event = {0x0B, 0xB0 | channel, control, value};
  MidiUSB.sendMIDI(event);
}
```

```
46
47 void setup()
48 {
49   Serial.begin(115200); // Start serial at 115200 bps
50
51
52   uint32_t status = dof.begin();
53
54     dof.setAccelScale(dof.A_SCALE_2G);
55     dof.setGyroScale(dof.G_SCALE_245DPS);
56     dof.setMagScale(dof.M_SCALE_2GS);
57     dof.setAccelODR(dof.A_ODR_200);
58     dof.setAccelABW(dof.A_ABW_50);
59     dof.setGyroODR(dof.G_ODR_190_BW_125);
60     dof.setMagODR(dof.M_ODR_125);
61     dof.calLSM9DS0(gbias, abias);
62 }
63
64 void MadgwickQuaternionUpdate(float ax, float ay, float az, float gx,
       float gy, float gz, float mx, float my, float mz)
65 {
66     float q1 = q[0], q2 = q[1], q3 = q[2], q4 = q[3];    // short name
       local variable for readability
67     float norm;
68     float hx, hy, _2bx, _2bz;
69     float s1, s2, s3, s4;
70     float qDot1, qDot2, qDot3, qDot4;
71
72     // Auxiliary variables to avoid repeated arithmetic
73     float _2q1mx;
74     float _2q1my;
75     float _2q1mz;
76     float _2q2mx;
77     float _4bx;
78     float _4bz;
79     float _2q1 = 2.0f * q1;
80     float _2q2 = 2.0f * q2;
81     float _2q3 = 2.0f * q3;
82     float _2q4 = 2.0f * q4;
83     float _2q1q3 = 2.0f * q1 * q3;
84     float _2q3q4 = 2.0f * q3 * q4;
85     float q1q1 = q1 * q1;
86     float q1q2 = q1 * q2;
87     float q1q3 = q1 * q3;
88     float q1q4 = q1 * q4;
89     float q2q2 = q2 * q2;
90     float q2q3 = q2 * q3;
91     float q2q4 = q2 * q4;
92     float q3q3 = q3 * q3;
93     float q3q4 = q3 * q4;
94     float q4q4 = q4 * q4;
95
96     // Normalise accelerometer measurement
97     norm = sqrt(ax * ax + ay * ay + az * az);
```

```
98      if (norm == 0.0f) return; // handle NaN
99      norm = 1.0f/norm;
100     ax *= norm;
101     ay *= norm;
102     az *= norm;
103
104     // Normalise magnetometer measurement
105     norm = sqrt(mx * mx + my * my + mz * mz);
106     if (norm == 0.0f) return; // handle NaN
107     norm = 1.0f/norm;
108     mx *= norm;
109     my *= norm;
110     mz *= norm;
111
112     // Reference direction of Earth's magnetic field
113     _2q1mx = 2.0f * q1 * mx;
114     _2q1my = 2.0f * q1 * my;
115     _2q1mz = 2.0f * q1 * mz;
116     _2q2mx = 2.0f * q2 * mx;
117     hx = mx * q1q1 - _2q1my * q4 + _2q1mz * q3 + mx * q2q2 + _2q2 * my
        * q3 + _2q2 * mz * q4 - mx * q3q3 - mx * q4q4;
118     hy = _2q1mx * q4 + my * q1q1 - _2q1mz * q2 + _2q2mx * q3 - my * q2q
        2 + my * q3q3 + _2q3 * mz * q4 - my * q4q4;
119     _2bx = sqrt(hx * hx + hy * hy);
120     _2bz = -_2q1mx * q3 + _2q1my * q2 + mz * q1q1 + _2q2mx * q4 - mz *
        q2q2 + _2q3 * my * q4 - mz * q3q3 + mz * q4q4;
121     _4bx = 2.0f * _2bx;
122     _4bz = 2.0f * _2bz;
123
124     // Gradient decent algorithm corrective step
125     s1 = -_2q3 * (2.0f * q2q4 - _2q1q3 - ax) + _2q2 * (2.0f * q1q2 + _2
        q3q4 - ay) - _2bz * q3 * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4
         - q1q3) - mx) + (-_2bx * q4 + _2bz * q2) * (_2bx * (q2q3 - q1q4) +
        _2bz * (q1q2 + q3q4) - my) + _2bx * q3 * (_2bx * (q1q3 + q2q4) + _2
        bz * (0.5f - q2q2 - q3q3) - mz);
126     s2 = _2q4 * (2.0f * q2q4 - _2q1q3 - ax) + _2q1 * (2.0f * q1q2 + _2q
        3q4 - ay) - 4.0f * q2 * (1.0f - 2.0f * q2q2 - 2.0f * q3q3 - az) + _2
        bz * q4 * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx)
        + (_2bx * q3 + _2bz * q1) * (_2bx * (q2q3 - q1q4) + _2bz * (q1q2 + q
        3q4) - my) + (_2bx * q4 - _4bz * q2) * (_2bx * (q1q3 + q2q4) + _2bz
        * (0.5f - q2q2 - q3q3) - mz);
127     s3 = -_2q1 * (2.0f * q2q4 - _2q1q3 - ax) + _2q4 * (2.0f * q1q2 + _2
        q3q4 - ay) - 4.0f * q3 * (1.0f - 2.0f * q2q2 - 2.0f * q3q3 - az) +
        (-_4bx * q3 - _2bz * q1) * (_2bx * (0.5f - q3q3 - q4q4) + _2bz * (q2
        q4 - q1q3) - mx) + (_2bx * q2 + _2bz * q4) * (_2bx * (q2q3 - q1q4) +
         _2bz * (q1q2 + q3q4) - my) + (_2bx * q1 - _4bz * q3) * (_2bx * (q1q
        3 + q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
128     s4 = _2q2 * (2.0f * q2q4 - _2q1q3 - ax) + _2q3 * (2.0f * q1q2 + _2q
        3q4 - ay) + (-_4bx * q4 + _2bz * q2) * (_2bx * (0.5f - q3q3 - q4q4)
        + _2bz * (q2q4 - q1q3) - mx) + (-_2bx * q1 + _2bz * q3) * (_2bx * (q
        2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) + _2bx * q2 * (_2bx * (q1q3
         + q2q4) + _2bz * (0.5f - q2q2 - q3q3) - mz);
129     norm = sqrt(s1 * s1 + s2 * s2 + s3 * s3 + s4 * s4);    // normalise
         step magnitude
```

```
130     norm = 1.0f/norm;
131     s1 *= norm;
132     s2 *= norm;
133     s3 *= norm;
134     s4 *= norm;
135 
136     // Compute rate of change of quaternion
137     qDot1 = 0.5f * (-q2 * gx - q3 * gy - q4 * gz) - beta * s1;
138     qDot2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy) - beta * s2;
139     qDot3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx) - beta * s3;
140     qDot4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx) - beta * s4;
141 
142     // Integrate to yield quaternion
143     q1 += qDot1 * deltat;
144     q2 += qDot2 * deltat;
145     q3 += qDot3 * deltat;
146     q4 += qDot4 * deltat;
147     norm = sqrt(q1 * q1 + q2 * q2 + q3 * q3 + q4 * q4);    // normalise
       quaternion
148     norm = 1.0f/norm;
149     q[0] = q1 * norm;
150     q[1] = q2 * norm;
151     q[2] = q3 * norm;
152     q[3] = q4 * norm;
153 
154 }
155 
156 
157 
158 
159 void loop()
160 {
161   dof.readGyro();           // Read raw gyro data
162     gx = dof.calcGyro(dof.gx) - gbias[0];   // Convert to degrees per
      seconds, remove gyro biases
163     gy = dof.calcGyro(dof.gy) - gbias[1];
164     gz = dof.calcGyro(dof.gz) - gbias[2];
165 
166     dof.readAccel();          // Read raw accelerometer data
167     ax = dof.calcAccel(dof.ax) - abias[0];   // Convert to g's, remove
      accelerometer biases
168     ay = dof.calcAccel(dof.ay) - abias[1];
169     az = dof.calcAccel(dof.az) - abias[2];
170 
171     dof.readMag();            // Read raw magnetometer data
172     mx = dof.calcMag(dof.mx);     // Convert to Gauss and correct for
      calibration
173     my = dof.calcMag(dof.my);
174     mz = dof.calcMag(dof.mz);
175 
176     Now = micros();
177     deltat = ((Now - lastUpdate)/1000000.0f); // set integration time
      by time elapsed since last filter update
178     lastUpdate = Now;
```

```
179
180     MadgwickQuaternionUpdate(ax, ay, az, gx*PI/180.0f, gy*PI/180.0f, gz
        *PI/180.0f, mx, my, mz);
181
182     yaw   = atan2(2.0f * (q[1] * q[2] + q[0] * q[3]), q[0] * q[0] + q[1
        ] * q[1] - q[2] * q[2] - q[3] * q[3]);
183     pitch = -asin(2.0f * (q[1] * q[3] - q[0] * q[2]));
184     roll  = atan2(2.0f * (q[0] * q[1] + q[2] * q[3]), q[0] * q[0] - q[1
        ] * q[1] - q[2] * q[2] + q[3] * q[3]);
185     pitch *= 180.0f / PI;
186     yaw   *= 180.0f / PI;
187     yaw   -= 2.2; // MODIFIED FOR TURIN Declination at Danville,
        California is 13 degrees 48 minutes and 47 seconds on 2014-04-04
188     roll  *= 180.0f / PI;
189
190     Serial.println("New note or volume");
191     double elabroll;
192     elabroll=roll+90;
193     if (elabroll < 180 && elabroll > 0){
194
195       elabroll=elabroll/30;
196       if (elabroll - floor(elabroll)>0.5){
197           elabroll=ceil(elabroll);
198       }
199       else elabroll=floor(elabroll);
200
201
202       int notetoplay=elabroll+60;
203
204       #ifdef DEBUG
205       Serial.print("Note to play: "); Serial.println(notetoplay);
206       #endif
207
208       if (notetoplay != oldnote){
209         volumeincreasing=0;
210         Serial.println("Sending note off");
211         noteOff(1, oldnote, 64);  // Channel 0, middle C, normal
        velocity
212         MidiUSB.flush();
213         //delay(100);
214         Serial.println("Sending note on");
215         noteOn(1, notetoplay, 64);   // Channel 0, middle C, normal
        velocity
216         MidiUSB.flush();
217
218         //delay(200);
219       }
220       else{
221
222         double curraccx=ax*SENSORS_GRAVITY_STANDARD;
223         double curraccy=ay*SENSORS_GRAVITY_STANDARD;
224         double curraccz=az*SENSORS_GRAVITY_STANDARD;
225
226         #ifdef DEBUG
```

```
227           Serial.print("Curracx");Serial.println(curraccx);
228           Serial.print("Curracy");Serial.println(curraccy);
229           Serial.print("Curracz");Serial.println(curraccz);
230           #endif
231
232           if (sqrt(curraccx*curraccx+curraccy*curraccy+curraccz*curraccz)
      -SENSORS_GRAVITY_STANDARD > 0.05){
233             if (64+volumeincreasing+floor(sqrt(curraccx*curraccx+
      curraccy*curraccy+curraccz*curraccz)-9.8) < 128) {
234               volumeincreasing=volumeincreasing+floor(sqrt(curraccx*
      curraccx+curraccy*curraccy+curraccz*curraccz)-9.8);
235               controlChange(1, 7, 64+volumeincreasing);
236               MidiUSB.flush();
237             }
238             else{
239               controlChange(1, 7, 128);
240               MidiUSB.flush();
241             }
242           }
243           else{
244             if (volumeincreasing>0) {
245               volumeincreasing=volumeincreasing-1;
246               controlChange(1, 7, 64+volumeincreasing);
247               MidiUSB.flush();
248             }
249           }
250           //delay (60);
251         }
252         oldnote=notetoplay;
253       }
254 }
```

Listing 28: Arduino Due code - initial tests

List of Figures

Listings