UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

ECE PARIS ÉCOLE D'INGÉNIEURS

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

KTRL midi controller

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Pastor Bermúdez, Vicente

Tutor: Galiano Ronda, Isabel Remedios

2015-2016

Resumen

Estamos en una época en la que la tecnología es más que importante para la industria de la música. Antes dicha tecnología se utilizaba exclusivamente en los estudios de grabación, pero hoy en día podemos ver espectáculos en directo donde se produce música utilizando esta tecnología. ¿Realmente se puede ver cómo el músico moderno interactúa con la tecnología para producir su música? ¿Se puede considerar esto un verdadero espectáculo?

Mediante este proyecto se pretende cambiar la forma en la que se interactúa con la tecnología para producir música. El objetivo es que el público pueda apreciar de una forma más visual lo que el músico moderno realiza en un escenario mediante la tecnología. Para ello hemos creado una controladora MIDI con sensores de proximidad manejados por una placa Arduino.

Palabras clave: MIDI, Arduino, sensores de proximidad, controladora.

Abstract

We come to an era, where technology is more than important in music industry. First, music was written by humans. Then performed by humans, performed by musicians. Music spreads. It spreads via physicals, via vinyles, via CDs. Technologies were meant for recording studios : the compressors, the mixing desks, the microphones, the amplifiers, etc... Then, they spread on stage. Lights, visuals, performances. We come to a step, now, where electronic music shows are as important as music concerts. Music concerts have musicians playing their instruments and electronic music shows have DJs/Live performer mixing, playing their tunes with technology. These technologies can either be mixing console, turntables, controllers. But where is the performance in the way that the music is played by technology ? People go to a concert to see the musician performance. But when can we categorize an electronic music show as a performance ?

With KTRL, midi controller for Digital Audio Workstation, our goal is to provide entertainment to an electronic music show, to give to the audience some visuals, to give them stuff to watch, to show them what the performer is actually doing, also in an original way, a different way than normal. The research we made, comes from different statement, different point of view we, the team, had concerning electronic music shows. 3/4 of the team are into electronic music, and after going to severals concerts during our lifetime, we came to a point where it is necessary to the audience to know what the performer is doing. What is more boring than watching someone with his face down on his setup without even looking to the audience ?

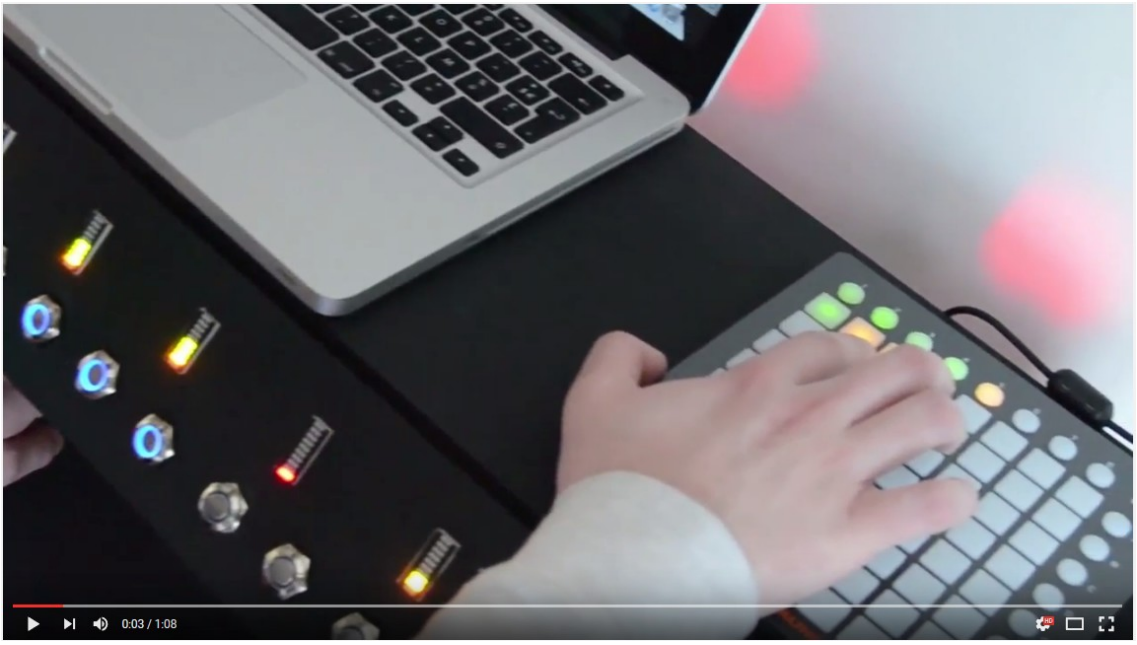
Another point for KTRL, as electronic music shows are very common, more and more people are into producing their own music and performing it. Also they are trying to reach a step where they can be more original than the others. One of our goal is to give access to the people, tutorials that we are putting in shape on our website so they can build, craft the controller. Our open vision, shares interest in the DIY (aka Do It Yourself) community. Making the controller yourself, gives you the impression of building an instrument that you care about, also the pride of owning something you did.

Keywords : midi, Arduino, proximity sensors, controller.

Table of Contents

The content is based on the documents required by ECE Paris for the realization of this project.

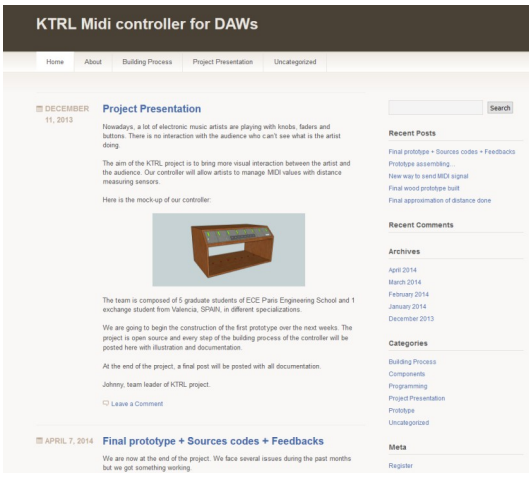
1. Introductory video



https://www.youtube.com/embed/D4qPhk1MOxs

2. Websites of the project

1.Wordpress:



http://ktrlmidi.wordpress.com

2.Facebook:

https://www.facebook.com/KTRLmidi

3.GoogleCode (Source code):

http://ktrl.googlecode.com

3. Project Roadmap

Document “Project Roadmap”

PPE13-33 Project Roadmap

Project Classification

Project name: KTRL

Cluster: Digital Entertainment

Valorization: Open Source

Project Description

Midi controller for live performance focused on hand motions using sensors. The controller will be more appealing to the audience because the crowd will see easily the artist controlling effects without touching anything. The project will allow the artist to manage more effects at the same time. KTRL will be a controller as a complement of a whole setup. It will be not the main controller of the artist for his live performance.

Project Valorization Strategy

The Open Source Valorization may be the best strategy for our project. Indeed, there is a huge community of people who are producing electronic music. Some of them are professionals. However the majority of music producer is composed of beginners and artist they listen inspires them. They all use MIDI controllers and some of them make live performances. All those people make a united community. They love trying new products in order to find a better way to create their music. We will use this community to ask questions and it will help developing our product. Moreover, we will create a website to show to people our project and they could send us feedbacks. We will also explain how we made the controller. Thanks to that anybody could build the product.

Project Team

Team leader: Johnny Lo

Team member roles

Name	Role
Pierre-Emile Boiron	Hardware, Arduino
Vincent Leboeuf	Sensors, display
Alexis Martin	Hardware, Arduino (functions, presets)
Vicente Pastor Bermudez	Arduino (link with computer), design
Stéphane Ruhlmann	Software (computer)

Project Actors

Mentor: Romain Da Costa
Email: romain.dacosta1@gmail.com
Telephone: 06 85 67 03 84

Cluster Director: Guillaume Jacquemin
Email: jacquemi@ece.fr
Telephone: 06 22 47 09 57

Valorization Director: Pierre Paradinas
Email: .....
Telephone: .....

Partner Mentor: .....
Email: .....
Telephone: .....

Project Deliverables

Name	Description
Maquette	The physical aspect of the product, with the components and all the technologies we will use
Response to sensors	Tests and choice of the sensors used
Buttons & LEDs	Interface with the user, buttons to choose the mode and LEDs to display different values from the DAW
Different modes	Different modes in which the system will respond to the sensors and the motions
Presets	Simple way to switch between configurations
Link with software	Ability to interact with DAWs and control other MIDI systems
Final prototype	Final product we will present
Website & Tutorials	Share the steps to build your own controller and how to program it in order of the open source valorization

Project Calendar

Date	Milestone / Deliverable
21/10/2013	Bill of specifications
4/11/2013	Purchases of equipment
9/12/2013	Validation maquette
27/1/2014	Sensors tests & electronic devices connections
3/2/2014	Validation SADv1
17/3/2014	Presets & Modes
7/4/2014	Final product & Website/Tutorials

4. Specifications document

Document “Specifications document”

2013/2014 | PPE : KTRL

Specifications document

Vicente Bermudez
Pierre-Emile Boiron
Johnny Lo
Vincent Leboeuf
Alexis Martin
Stéphane Ruhlmann

Table of Contents

1. Introduction

Nowadays, it is very easy to make electronic music, just using a computer and some electronic devices. But, performing it during a show can be painful as the possibilities are endless. You can for instance, perform with a band, or perform as a soloist using DJ setup or live setup.

DJ setup or Live setup ?

DJ setups are the most common as it requires less space. DJs just come with CDs, or a USB key, a computer with the software and they just have to plug everything and play as the venue is more likely to have turntables.

Live performances are different. The performer has to bring his whole setup at every venue he attends.

These two setups are different ways to perform

2. State of the art and existing solutions

Nowadays there are plenty of MIDI controllers for multiple purposes, they can be categorized that way :

- **DJ controllers :**

When DJs began to make mixes and play songs, the platform they used vinyl discs. They used two or more vinyl players to make their mixes and touched directly the vinyl discs to synchronize the songs and make the sound effects.

Nowadays DJs controllers have one or more shuttle jogs, this is a circle that tries to emulate the vinyl players so the DJ can touch this like if this was a vinyl disc.

These controllers are focused on mix and synchronization between songs and also have some effects and equalizations.

- **Live PA controllers:**

These controllers are used for live performances, it's basically a grid of buttons and every button has a sample recorded on it.

When the user presses a button, a sample starts to play, this sample can be played continuously or only once every time that the button is pressed. These controllers support multi-touch and some of them play the sample with different volumes according to the pressure the button is pressed. Also you can record a sequence of pressed buttons and play it every time you want. At the same time, you can press other buttons. With these features the artists can make complex songs during live performances using simple samples.

Other interesting feature is that these controllers can synchronize the samples, that means that they divide the time in ticks and if the user presses the button between one tick and the following the sample will be played on the most proximal correct tick.

Controllers for live performance are focused on playing samples and it's not usual that they have other kind of functionalities.

- **Instrument controllers:**

The instruments controllers have the same form as usual instrument and can be played like real ones, directly sending MIDI signals. The most common instrument controller used is keyboard but there are a lot of existing instrument controllers that can be used.

This kind of controllers is focused on very accurately emulate the instrument that they represent. So they need to be very precise and have a custom MIDI detector like airflow.

- **DAW controllers:**

These controllers are used for music creation in digital audio workstations (DAW). They need a lot of functionalities and a lot of replicate modules because on the DAWs the user needs to be able to manage a lot of different audio tracks and their effects so every track or group of tracks are mapped to one of this modules where you can control the volume and equalization separately.

These controllers also have other kind of modules to create or manage the effects. It must be very complete and it is usual that they have a big size. It can be said that these controllers are the most complete ones because they can use the others controllers explained before.

Critical analysis (strengths and weaknesses)

DJ controllers:

Strengths:

- Compact size compared to old vinyl players
- Emulate vinyl disks
- Don't need a physical device to store one or few songs like disks or similar

Weaknesses:

- Don't have a lot of functionalities and sometimes Djs need other controllers.
- The latency between the controller and the music can be significant.

Live PA controllers:

Strengths:

- Appearance is important on live performance and this kind of controllers usually has a lot of lights and attracts attention.
- Easy to understand how they work.

Weaknesses:

- To make complex songs, the user needs a lot of experience.
- The quality of the live performance depends very much on the samples quality.

Instrument controllers:

Strengths:

- The musician already knows how to play it.
- The controller sends a midi signal directly to the software and don't need a microphone.

Weaknesses:

- This controllers needs to emulate very well the instrument and the way that it is played or the musician won't be comfortable playing it.
- Need some special devices to convert the inputs on the midi signals.

DAWS controllers:

Strengths:

- This kind of controllers is very complete.
- Can be used for many purposes.

Weaknesses:

- The size of the controllers is very big.
- Need a powerful computer whit high quality sound devices

3. System requirements

Live performances are different. The performer has to bring his whole setup at every venue he attends.

The KTRL controller enters in the live performance category. It doesn't provide a whole control of proper software but it is meant to complete a setup and to have a specific function. By setup, we mean other controllers that may or may not complete each other.

The controller is meant for performers that are looking for an original way to perform and control the effects in their set. Using sensors, the performer will be free handed and just using the hand can be more intuitive and give more human feels to the set.

As the project is part of the open source valorization, we expect it to grow thanks to the community; people will probably add more functionality that will suit them and their setup.

The performers are the first to influence the system. We have to aim the artists to know if the controller can suit them. Also, the community influences the final product, they can provide new ideas, help in the development of the controller.

There is a large "Do It Yourself" community that can be helpful for the conception and the ergonomic of the controller.

But during a live show, only the artist will be in interaction with the controller. He will control the software "Ableton Live" on his computer.

4. Functional requirements

1. Requirements definitions (what is Live, Max for Live)

Ableton Live is a loop-based software music sequencer and digital audio workstation (DAW) for OS X and Windows. Live differs with many other DAWs in that it is designed to be an instrument for live performances as well as a tool for composing, recording, mixing and mastering. It is also used for beat match of tracks (matching the tempi of different songs) by DJs, as it offers a suite of controls for beat matching, cross fading, and a large choice of effects.

It also support VST (virtual instruments and effects plug-ins), Audio Units, ReWire (allows using other DAWs like Reason as plug-ins).

Live interface on Mac OSX

Max/MSP is a visual programming software that uses visual objects to implement code in order to create instruments, effects and even software prototypes.

Recently, Ableton and Cycling74 have co-developed Max for Live, a Max/MSP version that is ran as a plug-in in Live.

2. System context

Live has become the most used software for DJ sets and electronic lives as more and more MIDI controllers are specially designed for Ableton's software.

Our controller will have to be plug-and-play on Live so we can use it directly to control existing projects and sets.

We will design an interface in Max for Live to allow the user to have control on both instruments and effects.

3. Functions

Our controller will have to feature several key functions:

- Control DAWs and other MIDI software

The main purpose of the project is to provide MIDI data in order to control virtual instruments, effects or even DAWs. The micro controller will translate analogic variations into MIDI data used for macros in DAWs such as Live.

Live has become the most used software for DJ sets and electronic lives as more and more MIDI controllers are specially designed for Ableton's software. Our controller will have to be plug-and-play on Live so we can use it directly to control existing projects and sets.

- Being controlled by hand-motion

Here comes our project's key feature: hand-motion control. In order to have more fluidity, a better visual aspect and a different experience for both artist and audience, our controller will be totally contactless.

Distance sensors will acquire artist's hands motions and the system will translate it into MIDI variation. The data acquisition will be provided to Arduino, then to our scripts and finally to Ableton Live which will control some instruments or effects.

- Display current value of each parameter

The KTRL will display each parameter controlled by a sensor, in order to help the user to know where things are on a LED row. Also, it will provide a visual experience to the audience so it can result into an interactive show.

- Communicate with Max/MSP

In order to control Ableton Live macros we will implement a patch in Max/MSP that will allow the user to directly control some parameters in Max software and to create his own presets.

- Being both elegant and sophisticated

Though it has to be as user-friendly as it can, the KTRL is designed to be a part of a live show so we want it to stand out from the user's gear.

- Feature different control modes

The user will be able to choose a control mode, which will allow him to have absolute, relative and group modes so the microcontroller will interpret the motions and translate

4. Acceptance Criteria

Requirement	Acceptance	Flexibility
Control DAWs and MIDI software	Being plug and play	Negotiable High priority
Being controlled by hand-motion	Sensors respond properly to hand motion	Not negotiable Very high priority
Display values of parameters	Being plug and play or at least easy to configure	Negotiable Middle priority
Feature different control modes	Having several factory modes	Not negotiable Middle Priority
Feature different control modes	Easy user presets configuration	Negotiable Middle priority
Communicate with Max/MSP	Sending/receiving information from Max	Negotiable High priority
Communicate with Max/MSP	Factory patch	Not negotiable High priority

5. Recommended implementation architecture

In order to send a MIDI value to the computer starting from the sensors, we will design an architecture using a board with a microcontroller. The market is enormous and we have to focus on one system, which will respond to our requirements without useless interfaces. The board will control everything and it will just send the data to the computer. That is why the choice is very important.

As one of the main functions is to detect a hand motion, the selection of the sensors is crucial. We want an analog voltage in output of the sensor in order to do an A/D conversion. Moreover the measuring distance has to be appropriate for the project (~ 10–200 mm).

1. Alternatives

1. Development card

At first glance, there are several possible alternatives for the design of our project. The external architecture (inputs/outputs) is quite similar but the platform is different. Here are a couple of manufacturers that sell development cards (some of them are available at ECE):

Arduino	Cheap, open source, the largest community of developers, Atmel chips, a lot of products
Raspberry	Powerful embedded system, "Little computer" designed to run GNU/Linux platform, create some games
BeagleBoard	Powerful embedded system, open, "Little computer" to run Linux/Android systems
myAVR	Quite cheap, good development kit, few modules, almost no community
mikroelectronika	Very good documentation, starter-kit EasyPIC 7, several

	interfaces, big development cards
chipKIT	Compatible with many Arduino codes, PIC32 bits chips, brand-new community (not so big), cheap, several cards

Most of those solutions are not suitable for the project. Indeed, we don't need the more powerful microprocessor, or a card like Raspberry Pi, which can run GNU/Linux or small systems.

Arduino proposes the largest solution of development cards. Here are a couple of interesting cards suitable for our project:

Name	Processor	Operating Voltage/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [KB]	SRAM [KB]	Flash [KB]	USB	UART
Due	AT91SAM3X8E	3.3 V/7-12 V	84 Mhz	12/2	54/12	-	96	512	2 Micro	4
Mega 2560	ATmega2560	5 V/7-12 V	16 Mhz	16/0	54/15	4	8	256	Regular	4
Mega ADK	ATmega2560	5 V/7-12 V	16 Mhz	16/0	54/15	4	8	256	Regular	4

2. Sensors

The input information for the MIDI signal will come from sensors. The voltage value sent to the microcontroller will be converted into a MIDI signal, which will be transfer to the computer with a UBS link.

Several kinds of distance measuring sensors are available: infrared, ultrasonic, laser. The ultrasonic sensors have generally a response time very high. The laser sensors are more used in industry and do not meet the required features. Both of those types of sensors would not be acceptable for the project.

Sensors with the following features are required:

- 5V input voltage
- Ideal distance measuring between 10mm and 200mm
- Analog voltage output
- Great time response

Sharp provides a lot of distance measuring sensors using infrared light. Maxsonar makes also sensors but they have undesirable features and the datasheets are not very specific.

2. Recommendation

Arduino

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's indented for artists, designers, hobbyists and anyone interested in creating interactive objects or environments." (source arduino.cc)

Arduino is a good development platform. A lot of Arduino cards are available and you can (almost) program everything you want.

There is an enormous community of people who develop projects using the Arduino platform. We accessed to forums and blogs, which was helpful when we were looking for information. Furthermore, we can add extensions cards to the master card if we want new I/O pins.

We can analyze from the previous table that:

- The Due has a CPU Speed much higher than the two others (84 MHz) but contains less Analog In/Out (only 12 inputs and 2 outputs). But we need only 8 analog inputs for the sensors. This card runs at 3.3V; that could be a problem. Indeed, providing higher voltages, like 5V to an I/O pin could damage the board. Moreover most of the peripheral works at 5V.
- The Mega2560 and the Mega ADK are closely the same. The Mega ADK has a USB host interface to connect an Android device, which we will not use.

To conclude, we think that the Mega2560 seems to be the best solution for our project.

Sensors

An interesting sensor is the GP2Y0A41SK0F. It meets approximately the features required and will probably be used in our project to detect hand motion.

Other features

Moreover, other information on the microcontroller has to be seen by the user. For example, the states of all buttons will be displayed by turning on a simple LED inside them. A 7-segment display will show the number of the track the user is playing.

Design and ergonomics

The controller will be a box (dimensions around 1x0.3x0.3m) made of wood. The material is interesting because it is easy to work on it. The wood is also cheap and flexible.

Because of its huge size, all the control buttons and all the outputs for the artist (LEDs, 7-segment display) will be concentrated on one side of the controller. As a consequence there will be an extra place on the top of the controller. This will allow the user to put his computer on it and maybe an additional controller, which will be part of his whole set up.

For the audience, power LEDs on the back of the controller will display the levels of the tracks in order to create a great visual aspect. Usually, the audience can barely see the artist and what he is doing. Thanks to the power LEDs, the crowd could see more precisely how the artist manages effects.

Sources:

arduino.cc

www.conrad.fr/ce

microchip.com

www.lextronic.fr

6. Constraints

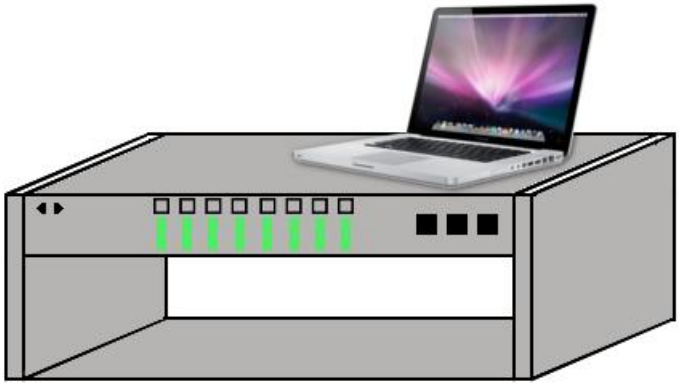
1. Design constraints

Security:

As we are using laser to point the position of the sensor, these lasers don't have to point somewhere where they can be in direct contact of the users and the audience’s eyes.

Ergonomic :

This is the expected aspect of our controller :



The sensors' positions have to be studied for a simple access to each of them. They also have to be well spaced to be accessible one by one.

We also need to be able to locate the sensors even in the dark because a lot of live performances are done in the dark. That's why, we are thinking about lasers pointing down to see the position of each sensor.

We need a way to see the midi value control for each sensor. We're going to put a column of LEDs corresponding to each sensor on the top of the controller.

We have to control different values in the same way at the same time. We'll develop a group function to group some sensors to make their MIDI value moving in the same way. This function will be implemented with eight buttons, one for each sensor. For example, if you group sensor one, two and four, you will just have to move one of them and the MIDI value of each of them will move in the same way.

Artists may want a function to switch between different preset on our controller. We want to implement a system of preset to switch between the auto-

mapping mode and a few other modes, created by the user. We're going to use some buttons to switch between those modes.

Finally, we'll need two buttons to switch between tracks and more particularly to switch between the effect-rack of each track.

We want to build the controller in wood. It also has to be light to be transportable by artists.

Platform:

Our controller has to be well recognized by all DAW. It also has to be automatically mapped on the Ableton effect-rack of the selected track. It also has to work on Windows and Mac OS.

2. Technology

We have to communicate with the computer by MIDI signal so we'll have to modulate the signal from the sensors in MIDI signal. Our controller also has to be connected by USB port to the computer because it's the easiest way to transfer MIDI signal in it.

The biggest constraint of our product is to be enough sensible and fast to allow a good control during live performance. Artists who play live music are used to have a physical control on their effects with knobs or buttons. The control is very responsive and communicates immediately with the software. We have to bring the same quality of control with our controller.

We will have to test different sensors to get the best concession between quality and price.

The sensors have to be really sensitive to detect short variations on a short-range distance.

The process of the electrical signal has also to be really fast to have the lowest possible latency. This latency shouldn't exist for the human perception.

Organizational constraints :

We need a USB port on the controller to connect it to a computer.

7. Glossary

Term	Definition
Analog signal	Is a signal that has a continuous variation in a range of values.
Artist	An artist is a person who creates, practices or demonstrates an art.
Beat	A beat is the sound/loop made by strokes.
Button	A button is a physical device that switch on/off a signal.
Chip	Is a set of electronic circuits encapsulated on a small case. Also called integrated circuit.
Controller	A controller is a device that generates, transmits and produces MIDI signals as input and/or output .
Cross fader	A cross fader is a physical device that increases / decreases the value of one signal, at the same time it decrease / increase the value of another one.
Digital signal	Is a signal that represents discrete values
Fader	A fader is a physical device that increases / decreases the value of one signal.
Hardware	The hardware is a group of physical components needed to run software instructions.
Integrated circuit	Is a set of electronic circuits encapsulated on a small case. Also called chip.
Knob	A knob is a physical device that controls the intensity of a signal. This device is moved circularly.
Live performance	A live performance is a concert where an artist creates music "on the fly".
Macro	A macro is a instruction that start a set of instructions to do automatic tasks or change between preset configurations.
Mastering	Mastering is the act of adjusting a song to his final form.
Microcontroller	A microcontroller is a small “computer” integrated in only one single chip.
Mix	Mix is when two or more sounds are combined in a new one.
Plug-and-play	Said of devices that don’t need configurations or specific software to work.
Plugin	A plugin is a complement for an application that extends the application functionality.
Rack	A rack is a metallic support for electronic equipment. Also

	it can refer a set of devices connected together, their can be virtual or physical devices.
Sequencer	It is a device used to generated MIDI signals in a programmed pattern.
Setup	A setup is the set of different controllers that one artist needs for a live performance.
Software	The software is a set of instructions running on hardware.
Track	A track can refer to an entire song or a subpart.

Acronym	**Meaning**	**Explanation**
BPM	Beats per Minute	It is a speed measure unit used in music.
CPU	Central Processing Unit	It is the main component of a computer where the software instructions are executed.
DAW	Digital Audio Workstation	This is a set of software solution focused on music generation, mixing and mastering.
DJ	Disc Jokey	Is a person who mixes songs.
LED	Light Emitting Diode	An electronic component that produces light.
MHz	Mega-Hertz	It is a measure of frequency.
MIDI	Musical Instrument Digital Interface	This is a protocol of serial communication used to send information related with the sound. This protocol can also be used for other purposes using the information that it send in a different way.
SPI	Serial Peripheral Interface	SPI is a standard of communications that is very common for transfer data between integrated circuits.
USB	Universal Serial Bus	USB is a standard bus to connect devices.
VST	Virtual Studio Technology	Software used to emulate instruments and physical devices used in traditional sound studios.

5. Technical Requirements Specification

Document “Technical Requirements Specification”

Technical Requirements Specification

for

KTRL

I4-TRS-PPE13-33-20131211

18-12-2013

Revision history

Name	Date	Changes	Version
Johnny Lo	13 Nov 2013	Starting from template: removed and arranged some parts of the plan.	0.1 draft
Johnny Lo	27 Nov 2013	Adding of external specification and first parts of the report	0.4
Johnny Lo	4 Déc 2013	Adding functional requirements and interface requirements	0.8
Johnny Lo	18 Déc 2013	Adding of the mock-up and review for final release	1.0 published

Table of Contents

1 Purpose

This document is the technical requirements specification. It provides the reader with all the specifications concerning the KTRL midi controller. The reader will have an inside look plus an outside look of the future final product.

2 Documentation and terminology

2.1 Reference documents

Document	Number	Attached	Application
Sharp GP2Y0A41SK0F	OP13008EN	No	Informations about sensors such as precision, range value, time response
ATmega2560/V	2549P–AVR–10/2012	No	Micro controller

2.2 Glossary

2.2.1 Terms

Term	Definition
Analog Signal	Is a signal that has a continuous variation in a range of values.
Ableton Live	Is a digital audio workstation. Used to produce and manage audio.
Controller	A controller is a device that generates, transmits and produces MIDI signals as input and/or output.
Digital signal	Is a signal that represents discrete values.
Midi channel	Whit one Midi wire can be controlled different units or parameters. To differentiate this units is used an identification. These identifications are the channels.
Software	The software is a set of instructions running on hardware.
MAC	An operating system developed by Apple.
Windows	An operating system developed by Microsoft.
Max for Live	Is a plugin for Ableton with preset instruments and the capacity for generates new ones.
Live performance	A live performance is a concert where an artist creates music "on the fly".
Bugs	It is an error on the software.
Hardware	The hardware is a group of physical components needed to run software instructions.
Arduino	Is a single-board microcontroller.
Pin	Is where the physical connections are done on the chips or electronic devices.
Protocol	It is a set of rules and regulations that determine how data is transmitted.
Plug-and-play	Said of devices that don’t need configurations or specific software to work.
Digital input	Is a signal that represents discrete values.
Setup	A setup is the set of different controllers that one artist needs for a live performance.
Hardware	The hardware is a group of physical components needed to run software instructions.
Artist	An artist is a person who creates, practices or demonstrates an art.
Track	A track can refer to an entire song or a subpart.
Rack	A rack is a metallic support for electronic equipment. Also it can refer a set of devices connected together, they can be virtual or physical devices.
Macro	A macro is an instruction that start a set of instructions to do automatic tasks or change between preset configurations.

2.2.2 Acronyms

Acronym	Meaning	Explanation
OS	Operating System	A collection of software that manages computer hardware and provides services for computer programs.
MIDI	Musical Instrument Digital Interface	This is a protocol of serial communication used to send information related with the sound. This protocol can also be used for other purposes using the information that it sends in a different way.
USB	Universal Serial Bus	USB is a standard bus to connect devices.
TTL	Transistor Logic	It is a class of digital circuits.

3 Product presentation

Our product is a midi controller with sensors. It delivers midi value and is the interface between the user and the software. We will first introduce the functional requirements such as the button used, then we will deal with the interface specification (ableton live 8 and Max). We want to have a live approach, so the controller has to be nice visually.

4 Functional Requirements

4.1 On/Off button (power)

This button represents a digital input of the system. It returns 1 when the button is turned on and 0 when it is turned off. It is circled in red on the following scheme:

This must be a “latching” button, because the system must be turned on as soon as the user pushes the button.

There are no specific physical requirements, because when an artist will use it, he will push it at the beginning of the show and won’t touch it during the performance. It should better be a light button because live performances are made in the dark.

Just under the on/off button, there is the usb hub.

4.2 The “modes” buttons

These buttons are represented on the following scheme:

The circled buttons selects the modes. When one button is pushed on, the mode is selected and the other modes are set to 0. So when we push a button, the other buttons are pushed off automatically.

We must have “latching” buttons too because the artist won’t keep his finger on the button to let the mode on. We have digital values that return 0 when the button is pushed off and 1 when it is pushed on, like a power button.

4.3 The “tracks” button

The buttons are circled in red in the following scheme:

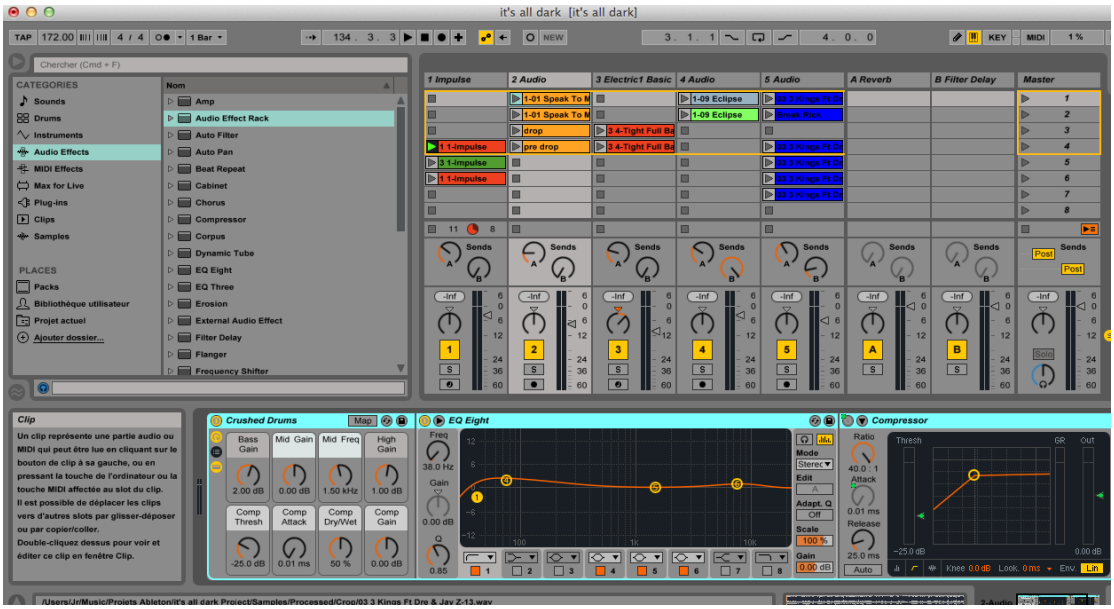
The 2 “tracks” buttons are used to switch between the tracks in the software. It could be done directly on the software but it’s easier to make it on the controller.

Unlike the other buttons, these ones must be “momentary buttons”, the value must stay at 1 only when the button is pushed on, and with a little delay. It returns the value 1 when it is pushed on and 0 when it is pushed off.

5 Interface requirements

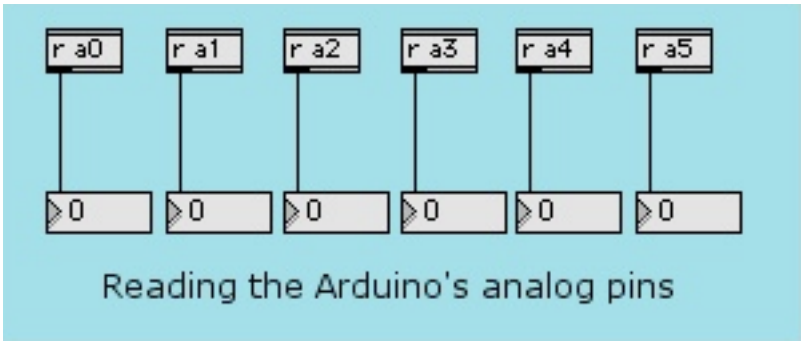
5.1 User interfaces

Here is Ableton Live, the software we will use to test our project. It’s a loop-based digital audio workstation that allows managing effects in a rack with macros controlling user-defined parameters.



Here is an example of an audio effect rack. Each parameter will be controlled by one of our sensors (in basic mode).

We will link our microcontroller inputs to the rack through a script in Max :



5.2 Hardware interface

The hardware interface will look like that:

The user will interact with the controller via eight buttons that will allow him to group the sensors to control one parameter.

The sensors will be under the top of the controller, getting the distance between the user's hand and their position.

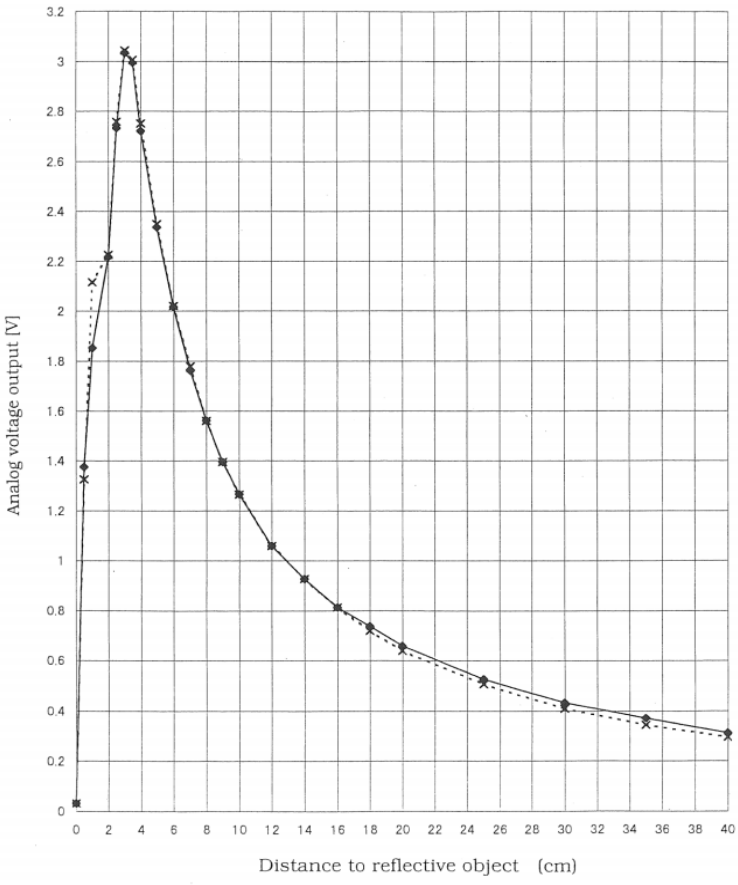
There will be some LED rows to display the parameters values to the user and to the audience.

6 Performance Requirements

6.1 Accuracy

We need a pretty high accuracy. The sensor has a measuring distance range from 4 cm to 30 cm. We need to manage a midi vale from 0 to 127.

So the maximum accuracy provided by our system is: Amax = (30 – 4) / 127 = 26 / 127 = 0,205 cm = 2,05 mm.



The Arduino card should manage the values of the analog voltage output by step to convert them in MIDI.

The accuracy will be increase around the actual value for a better control in the relative mode.

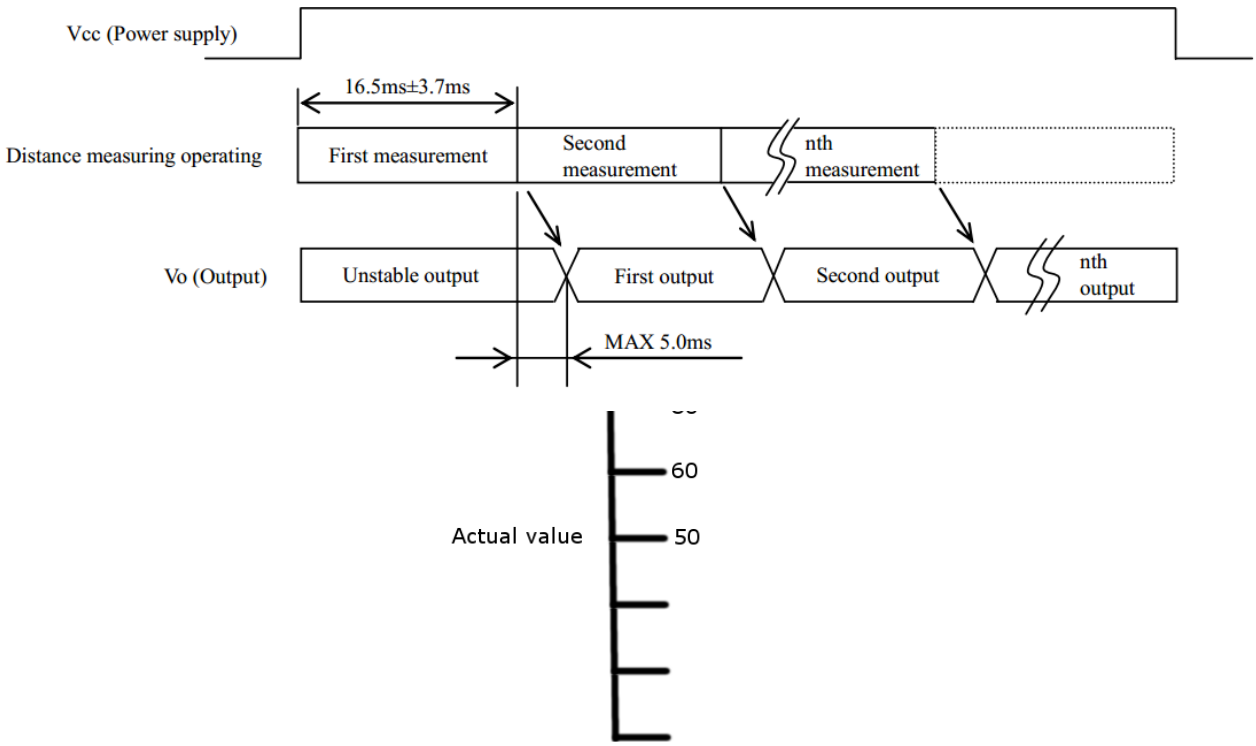
6.2 Response time

The response time of the sensors should be lower as possible to have the impression that everything is immediate when playing with the controller in live.

Something around 20 – 30 ms will be great.

We're going to use the Sharp GP2Y0A41SK0F.

Tmax = 16,5 + 3,7 + 5,0 = 25,2 ms



6.3 Stability

As our controller is oriented for live performance, we need it to be really stable and don't stop working in use. The Arduino card have to be able to manage several midi values of the sensors, blinking or not leds, get the action from the buttons without error.

The new version of Ableton Live allows connecting and disconnecting the controller when you want without the need to relaunch the software.

7 External requirements and constraints

7.1 Standards and compatibility requirements

Our product has to be compatible with Mac OS and Windows.

We will develop our solution using those versions, which are the last available:

- Ableton live 9: one advantage compared to Ableton Live 8 is that we can connect and disconnect a peripheral without closing the software. The 9.1 version provides a dual-screen mode in order to have a better experience.
- Max For Live: now the Max tool can be directly included into Ableton Live 9, which makes for instance the instruments configuration easier.

People that create electronic music or perform live have updated their software version. Indeed, the last version provides software improvement and always fixes bugs.

7.2 Hardware and software limitations

The Arduino Mega 2560 card will send a MIDI value to the computer throughout USB. The MIDI Data will be transferred using the ATmega16U2 USB-to-TTL Serial chip. Pin 0/19/17/15 (RX) and pin 1/18/16/14 (TX) of the card are used to receive and transmit TTL serial data. Pin 0 and 1 are also connected to the corresponding pins of the ATmega16U2.

Concerning the baud rate for the serial transmission, we have plenty of possibilities: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200 bps. The configuration of the baud rate is made in the code.

The controller will be huge in order to allow the user to put his computer on it. Indeed during live performance the artist, which could be a professional or not, does not have plenty of space around him. The space is very often limited. That is why the hardware part could fit easily in the controller's body.

7.3 Technology/Scientific Constraints

KTRL controller is based on the MIDI Protocol, using Max environment.

7.4 Physical requirements

The controller is made of wood, parallelepipedic. Concerning the size : 80x30x30cms. The weight will depend on the wood of the controller. There will be a metal plate on the front so the components can be fixed on.

7.5 Environmental requirements

The controller is live-oriented. But it can also be used for production purposes. It has to provide resistance against humidity as the live can be indoor or outdoor. Electronic devices are very sensitive towards humidity, so the electronic chip must be very isolated. For shocks, the controller has to be very solid as it can also be used as a desk for other midi controllers to sit on. Also, the controller will be very mobile, it has to be protected on the edges and the circuits must be fixed inside the box.

7.6 Documentation requirements

- List of the package content:
    - Security notices
    - User manual
        - Intro
        - Installation
        - Standard configuration
        - Ableton Live configuration
        - Basic usage
        - Advance usage
        - Fast support (FAQ technical)

7.7 Operations requirements

This controller is specifically designed for Ableton Live. So for full functionality the operator need to configure Ableton Live properly to work whit this controller. Also the operator must assign the midi channels from the controller to the midi channels where the controller is plugged.

7.8 Site adaptation requirements

The controller requires a flatbed to be placed. To work properly the controller needs to be plugged into some device that accepts midi signals or to get full functionality need to be plugged into a system whit the Ableton Live software.

Appendix A. Table des matières

6. Prototype Acceptance Plan

Document “Prototype Acceptance Plan”

Prototype Acceptance Plan

for

KTRL

I4-PAP- PPE13-33-20140408

04.08.2014

Revision history

Name	Date	Changes
Vincent Leboeuf	04.05.2014	Writing the document, defining tests
Johnny Lo	04.07.2014	Re-reading
Alexis Martin	04.08.2014	Last revision, final page layout

Table of Contents

1 Introduction

Nowadays, a lot of electronic music artists are playing with knobs, faders and buttons. The audience can't really see what the artist is doing.

KTRL is a MIDI controller using eight distance-measuring sensors to control functionality of DAW (Digital Audio Workstation). It was designed to be used with Ableton Live, especially to control effects.

The audience can see the artists' hands movement and catch the effects intensity with the power LEDs at the back of the controller.
It's also an intuitive and creative way to play music.

This document presents all the tests realized on our final prototype.

2 Test environment

2.1 Physical environment

To play with KTRL, you need to connect it to a computer via USB cable to give the power supply to the Arduino card.

You also need to connect the MIDI cable of the prototype to an external soundcard and then connect the soundcard to your computer and download the appropriate driver. This is to transfer MIDI data.

The prototype also has to be connected to an external power supply.

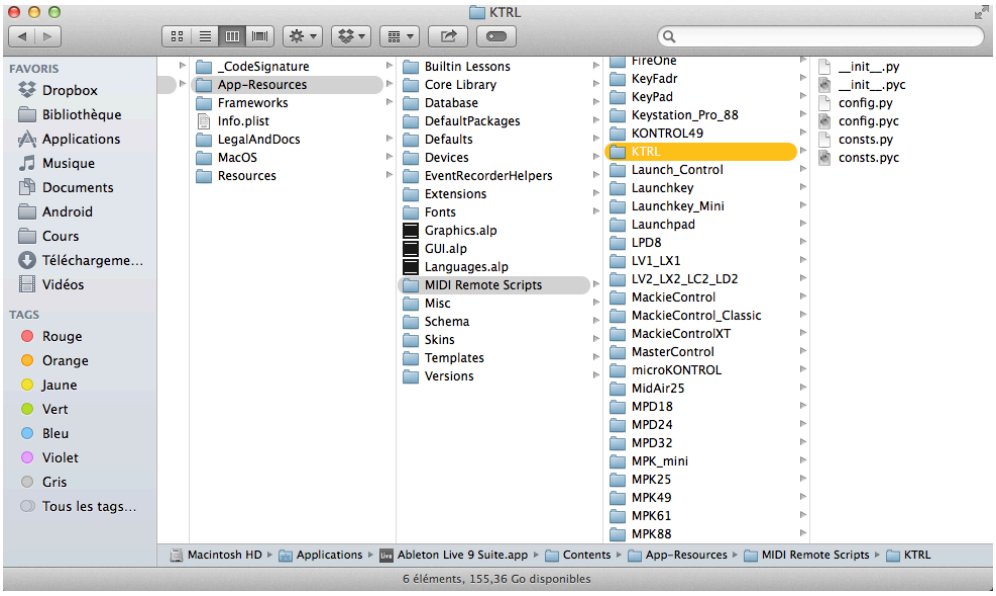
2.2 Hardware environment

KTRL can work on MAC OS and Windows. There is no minimal configuration required to play with KTRL.

2.3 Software environment

You need a version of Ableton Live software installed on your computer.

You also need to add the KTRL midi script in the appropriate folder.



You can download it here: http://ktrl.googlecode.com/svn/trunk/automapping/

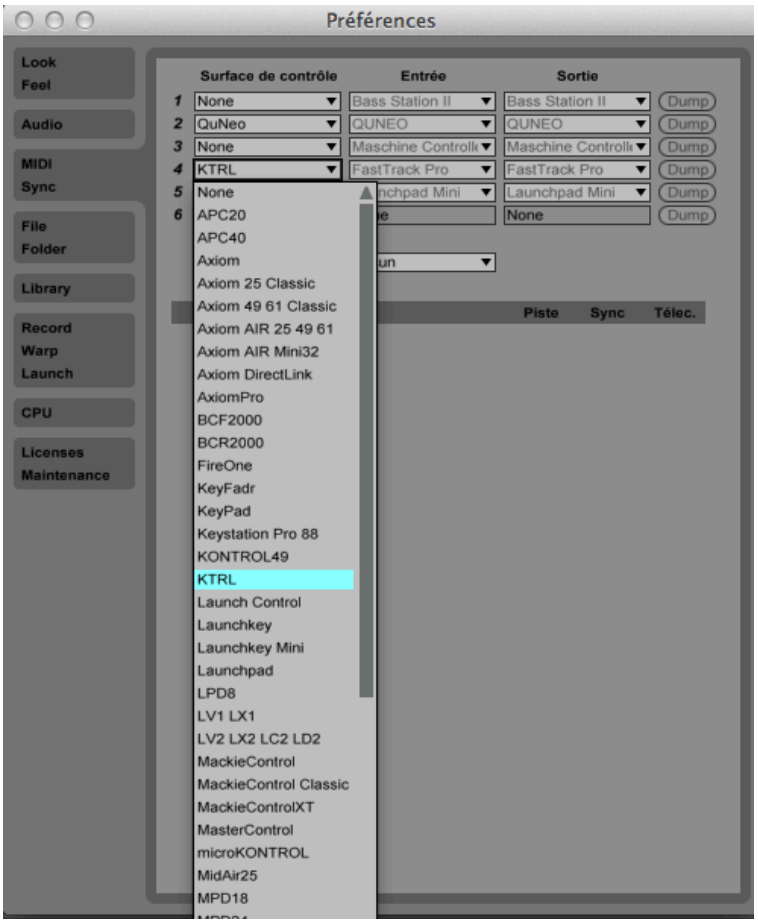
This script will make the prototype automatically map to parameters on Ableton.

The eight sensors will control the eight macros of the Audio Effect Rack of the selected audio track

2.4 Actor roles

This part is the goal of our project.

Any artist can come with is own configuration and is own controllers.
As it said before, the effects have to be added in an Audio Effect Rack and the parameters have to be assigned to the macros.
Then, you have to select the KTRL script in the Ableton preferences:



You have to allow Piste and Télec for the input.

The prototype is now ready to play, just launch an audio clip and play!

2.5 Data

The only external data file needed is the KTRL MIDI remote script.
You can download it at this link: http://ktrl.googlecode.com/svn/trunk/automapping/

Then see 2.3 section for the installation.

3 Test suite

3.1 Sensors

The sensors are used to control effects in Ableton Live. They have to return a value between 0 and 127, which corresponds to the position of the hand.

No.	*Description*	*Execution scenario*	*Expected results*	*OK/NOK*
1	*Use a sensor to control an effect*	*The user have to move his hand under the sensor*	*It should move the MIDI value of corresponding parameter in the Audio Effect Rack of the selected track.*	

3.2 Led bars

The led bars are showing the value of the effect of the corresponding sensors.

No.	*Description*	*Execution scenario*	*Expected results*	*OK/NOK*
2	*Display the effects value*	*The user has to move his hand under the sensor.*	*The leds should be blink in the same way as the effect is evolving.*	

3.3 Power leds

The power leds are blinked when a sensor is in use.

No.	*Description*	*Execution scenario*	*Expected results*	*OK/NOK*
3	*Display the active sensors*	*The user has to move his hand under the sensor.*	*The power led corresponding should be blink*	

3.4 Mode buttons

No.	*Description*	*Execution scenario*	*Expected results*	*OK/NOK*
4	*Activate memory mode*	*The user has to press the memory mode button*	*The sensors are now working in memory mode*	
5	*Activate free mode*	*The user has to press the free mode button*	*The sensors are now working in free mode*	
6	*Activate group mode*	*The user has to press the memory mode button, then several group buttons*	*The sensors are now working in group mode, which means that controlling one sensor could change several values*	
7	*Activate kill mode*	*The user has to press the kill mode button*	*The sensors who are not in group have to be reset*	

4 Test environment setup

To make it work, you just have to do the following tasks:

- The controller needs to be powered by plugging the electric supply cable to 220V.
- Then the MIDI cable must be connected to a sound card (or a MIDI to USB).
- Finally, the other USB cable should be plugged to a USB-220V adaptor or a computer.

Appendix A. Bibliography

http://arduino.cc/

http://learn.adafruit.com/

https://www.sparkfun.com/

http://www.seeedstudio.com/

https://forum.ableton.com

http://www.synthtopia.com/

http://en.wikipedia.org/

Appendix B. Glossary

See the following documents:

- CDC
- TRS
- SAD

7. System Architecture Document

Document “System Architecture Document”

System Architecture Document

for

KTRL

I4-SAD- PPE13-33-20140408

04.08.2014

Revision history

Name	Date	Changes
Vicent Leboeuf	03.15.2014	Definition of subsystems, external systems
Stephane Rulhman	03.13.2014	Ableton Live part added
Alexis Martin	03.15.2014	System global architecture, subsystems elements description
Pierre-Emile Boiron	03.15.2014	Introduction, schema of subsystems, hardware elements
Vicente Pastore	03.24.2014	Added components detailed part, schema of global system with hardware elements
Johnny Lo	03.26.2014	Automapping script part, re-read document
Alexis Martin	04.08.2014	Last revision, final page layout
Johnny Lo	04.08.2014	Final release

Table of Contents

1 Introduction

The purpose of this document is to explain all the parts of the project. We will introduce the global architecture of the product, divided in internal and external components, which are subsystems of the project.

The product is a MIDI controller based on 8 sensors that control different audio effects on any electronic music software, like Ableton Live. The values returned by the sensors are sent to the Arduino card, and send MIDI values to the software (from 0 to 127).

The modes are used to change the way of controlling the effects, and are managed by buttons on the front of the controller. Furthermore, power LEDs and LED bargraphs give a visual experience to our product.

First of all, the external systems have a direct influence on the product. The artist, who controls the controller, the computer, which receives the MIDI values and change the effects on the software, the other controllers and the sound system are external things that interact with the box.

The first subsystem is the Arduino card. It manages everything. It uses the sensor values to convert into MIDI; it controls all the modes and the group one. Furthermore, it delivers the power supply for the sensors.

The second one is the control subsystem that is represented by the sensors and the buttons and is managed only by human moves.

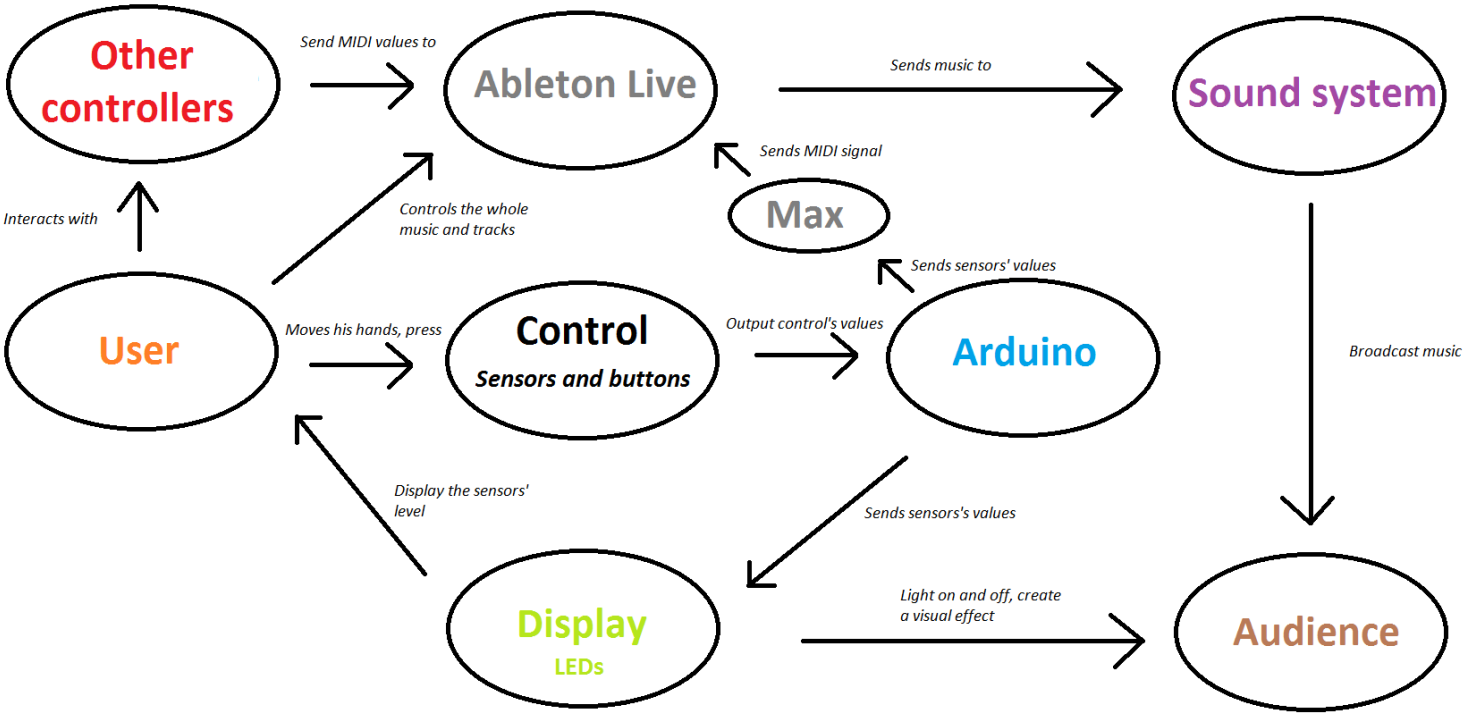
The last one is the “display” and is composed of the power LEDs and the LED bargraphs. The aim of this subsystem is to give a visual effect to our controller and to interact with the public.

1.1 Reference documents

See the following documents for a complete understanding of all terms:

- CDC
- TRS

2 System architecture



2.1 Subsystems

2.1.1 Arduino set

The Arduino card is a central element of our controller. We choose the Arduino MEGA 2560.

It does the communication between all the hardware elements and the computer : it gets in input the analog output of each elements (buttons, sensors).

It allows us to convert the analog voltage output of the sensors into MIDI signal and manage the different modes. It also allows blinking the different power leds and leds bar.

Then, we put an Arduino code on the card, which will use the input values to make them usable by Ableton. The program will convert the sensors values into MIDI values. It will also manage the different modes by reading the buttons values.

2.1.2 Control: sensors and buttons

The sensors are used to control an Audio Effect Rack in Ableton, which is composed of eight different MIDI parameters. They have to give the lowest response time as possible and the best accuracy.

We also have several buttons. Three of them are to control the global playing: free or memory. We also have one button to reset the values of the sensors that are not grouped. Eight buttons control which of the tracks is group.

2.1.3 Display

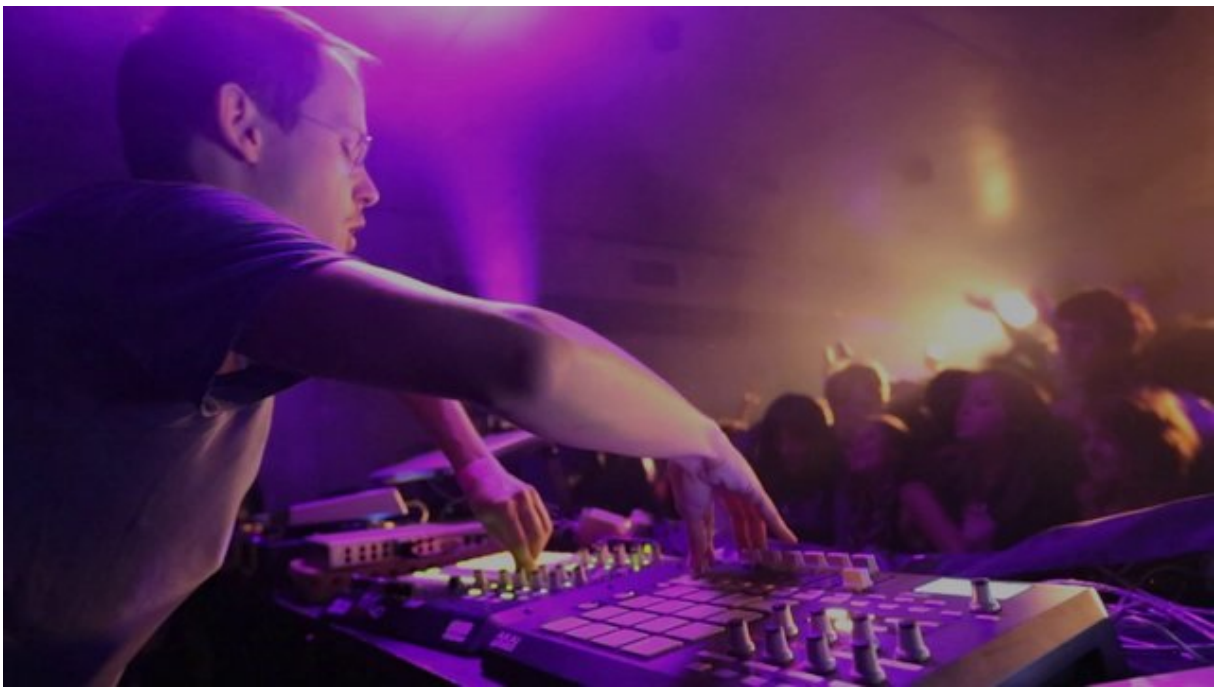
We use eight leds bar to display the value of each parameters control by the sensors.

We have also put eight power leds at the back of our controller (in front of the audience) to show which sensors are in use.

2.2 External systems

2.2.1 Artist

The user will be an electro music artist, eventually also producer. He will use our product during his live performance in front of a public, for example at a music festival.



Live performance of Rone

After discussing with professionals, we realize that artists have already their whole set of controllers, and they would not change all of them for one. Artists are often curious, and like trying new stuff. This is why our controller's purpose is to be a complementary to the others, not to be the main one. Our controller has a very precise use.

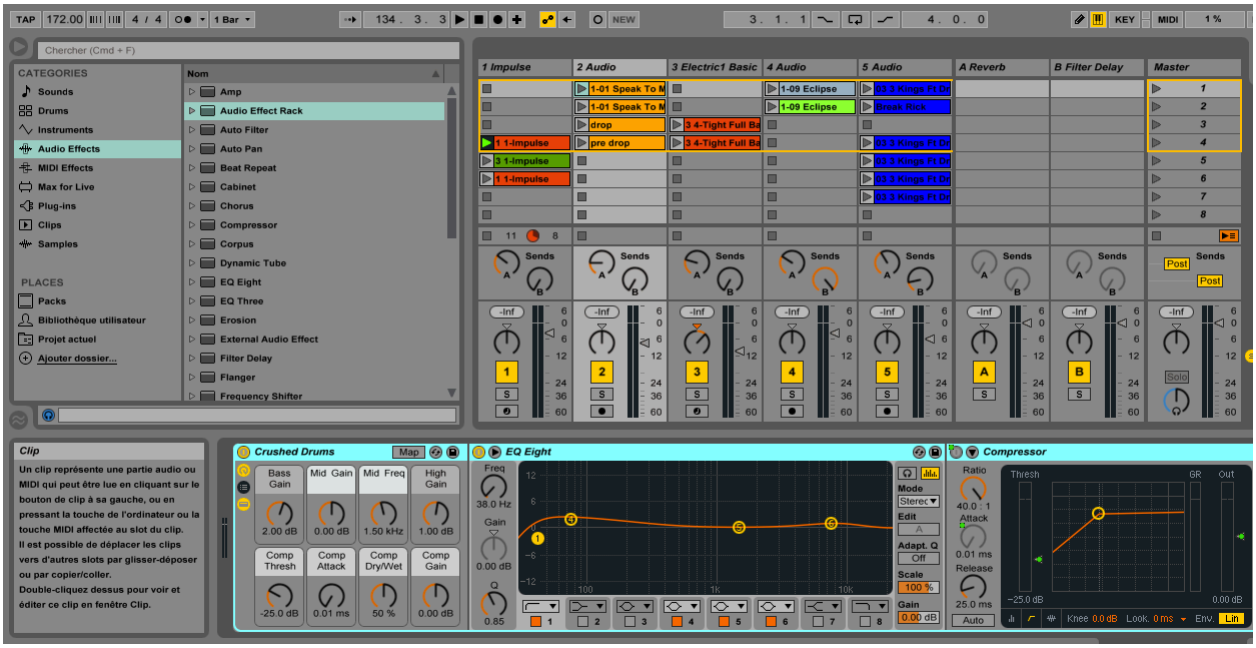
The user will interact with the controller by moving his hands and pressing buttons. All those actions will have an impact on the Ableton Live software on his computer, which generate the electronic music.

2.2.2 Computer (Max and Ableton Live)

Ableton Live is the most used software for electronic music live performances.

It allows connecting several MIDI controllers to control parameters.

Our controller is thought to work with the Ableton Audio Effect Rack but it can be used to control any other functions of any other software.



2.2.3 Other controllers

Nowadays, a lot of artists use several controllers in the same live session. They usually correspond to different part of the software. For example, one controller is going to launch audio clips, another one will control the mixer (volume, pan, …). Our controller is thought to control only one part of the Ableton live software: the Audio Effect Rack. It means that artists can still be playing with their actual setup and add our controller.

2.2.4 Sound System, audience

Usually in electronic music show, the audience can't really see and understand what the artist is doing.

The KTRL add a visual aspect for the audience during a live performance.

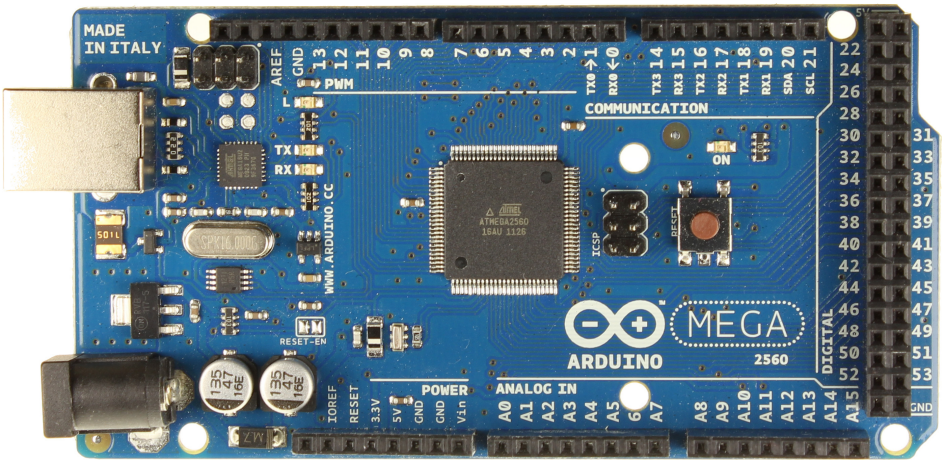
First the public can see the hands of the artist moving in the same way as the audio effect are playing.

Then, we also have power leds at the back of the controller, so in front of the audience, to show the evolution of the sensors value to the public.

3 Subsystem 1: Arduino set

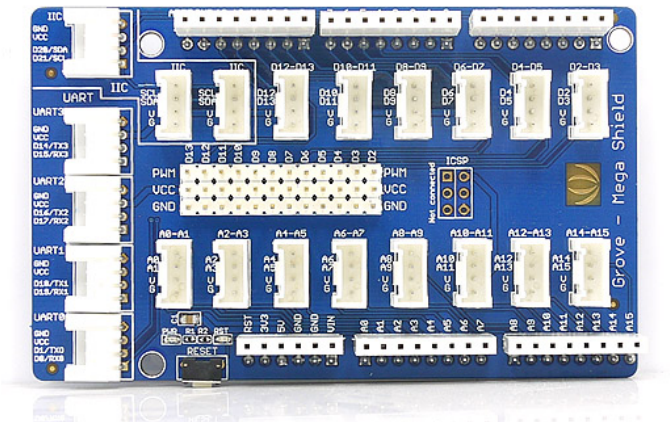
3.1 Hardware elements

3.1.1 Arduino Mega 2560



The Arduino Mega 2560 is the master of the project. All our components are connected to it, and the card communicates with the computer. Its main role is to manage the interaction between the user and the controller by receiving inputs value from sensors and buttons, and also by displaying the level of each track for the user and the visual effect for the audience.

3.1.2 Grove Mega Shield



Thanks to the Grove Mega shield, we can connect more easily the LEDs Bargraph and the sensors. There are connectors, which enable us to link directly the components to the card. Otherwise, we would have issues with the good connection of elements to the Mega 2560 because we use almost all the pins. Indeed, the Arduino card doesn't provide fixed and stable pins, and some of them will probably be in contact with each other.

3.2 Software elements

3.2.1 Arduino IDE

« The open-source Arduino environment makes it easy to write code and upload it to the i/o board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing, avr-gcc, and other open source software. »

We develop the project using Arduino IDE 1.0.5 on Mac and Windows. This software is hosted with Github, and librairies are written in C or C++.

On one hand, it is really fast to test the code on the Arduino and make changes thanks to this IDE. But on the other hand, it doesn't provide a great compiler and the implementation/readability of the code is not optimal. That is why we use another software.

3.2.2 Netbeans with Subversion and Google Code

The development of software for the arduino board has been done using the Subversion version control. Google code.google.com provide this service by free for the open source projects.

Using Subversion can be seen the growth of the project on the diferent versions. Also is a way to share the source code easily in a professional manner and of course all benefits provided by a version control are obtained.

The IDE (Integrated development environment) that has been used was the Arduino SDK (Software development kit) and also NetBeans with the Arduino plugin. NetBeans provide better developer tools and integration with the subversion server.

3.2.3 Library

We use one external library in order to implement the MIDI sending:

- The MIDI sending: <*Midi.h*>library

This library is of course open source, and available on the Arduino website.

3.3 Shared data definitions

3.3.1 Sensors and buttons values

The Arduino card gets back sensors RAW values and the state of the buttons.To have the exact distance between the user's hand, we implemented an approximation of the output function of the sensor. Otherwise, we would not have the "middle value" when the hand would be at half distance of the sensor.

Then, it computes the right algorithm with the combined functions (memory, free, release non grouped, group mode), in order to get the value that will be sent to the computer and to the LEDs.

All this data is stored in several arrays of NSENSORS (8 in our case).

3.3.2 MIDI signal

This is the most important part of the project. Indeed, the controller has to send a MIDI value to the computer and Ableton Live. To do that, we use the <*Midi.h*>prototypes, and the signal is sent through a MIDI cable, then a MIDI-to-USB interface. More details are in the MIDI part.

The communication between the hardware and software elements is a key point of our project. The Arduino will send the value we're interested in and it will be translated in MIDI to be therefore used by the user's digital audio workstation.

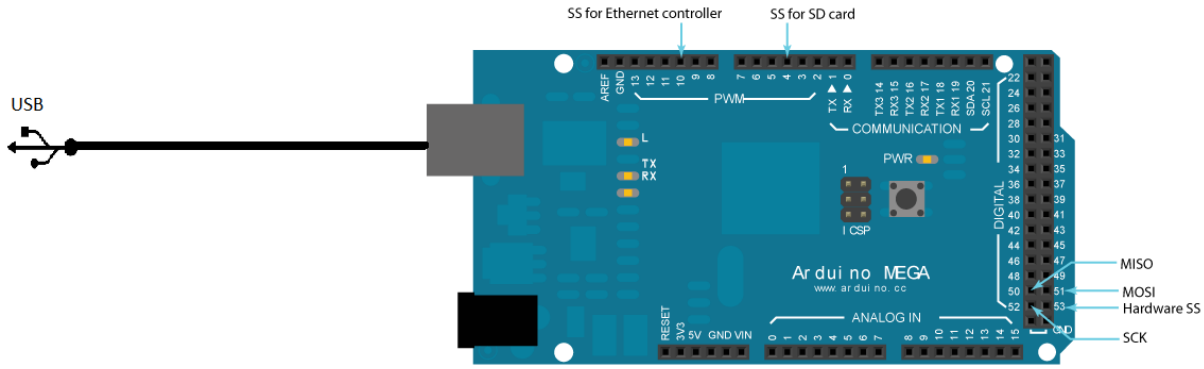
At first we considered two main solutions:

First, we planned to have the parameters values sent over the serial connection from Arduino to the computer and then be received by a Max patch.

This patch would parse the data and build a MIDI Control Change signal then send it.

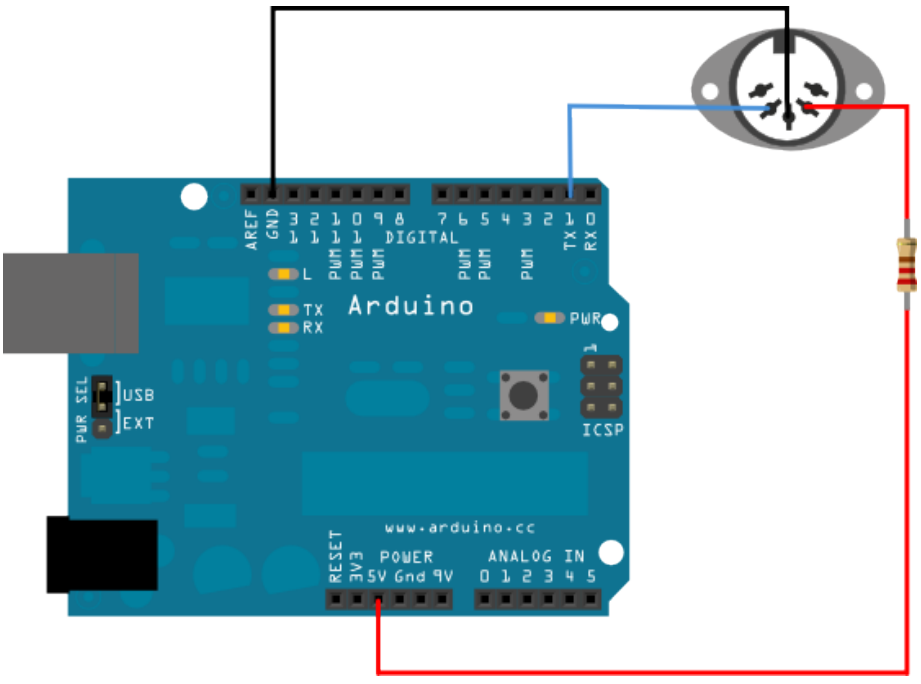
The user would then configure the DAW of his choice to use these MIDI CCs.

We decided to send MIDI directly via a MIDI connector/cable to the computer and receive it with whatever DAW user could use.



Plan B was to send MIDI information over the USB serial connection to be then directly received by the DAW. The MIDI over USB proceeding would be provided by a firmware installed on the Arduino.

The second solution is to send directly a MIDI signal through a MIDI cable, connected to our Arduino card via the following system:



We have to plug wires on the GND, the TX pin and the 5V pin, with a 220 ohm resistor.

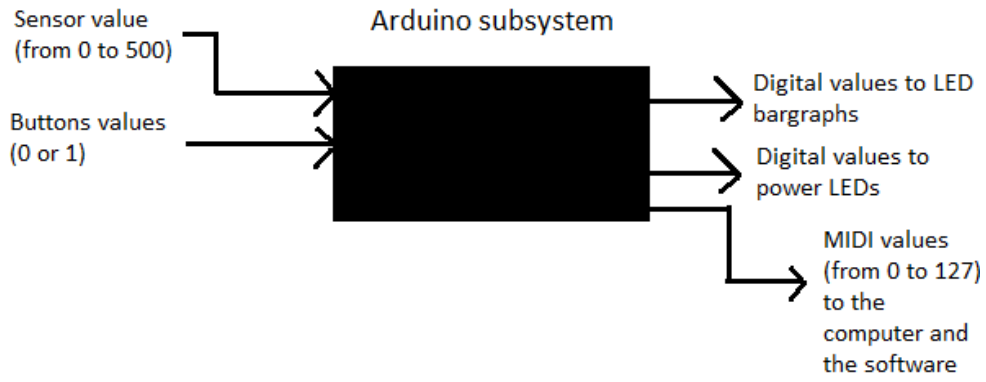
MIDI (Musical Instrument Digital Interface) is a protocol for controlling musical devices like our controller. This protocol operates at a 31250 baud rate (31250 bits per second). In our case, we

just receive data from the controller, but we send nothing in the MIDI cable, that's why we just plug the MIDI IN cable.

The data bytes are less than 127, that's we transform the sensors values into MIDI values. In the data, we have elements like the note to play and the velocity. In our case, we will just use the command ControlChange that will just change the value to send into the MIDI protocol.

3.4 Subsystem behavior

The following schema explains the interaction between the subsystem (the Arduino card), and the other elements. We made a "black box", because in this part we don't need to know what happens into the card, but how it interacts.



The card can receive different buttons values:

- From the mode
- From the sensors used for the group mode

It receives the values in the same time (from the sensor and from the buttons). For example, if the group is activated and the sensors 2,3 and 7 selected, the subsystem will receive the values from the buttons and from the sensors selected, and will send it to the different elements (bargraphs, power LEDs and the software via MIDI values).

4 Subsystem 2: Control (sensors and buttons)

4.1 Hardware elements

4.1.1 Sensors

The eight Sharp gp2y0a41sk0f sensors are the most important components for our project. They have a good range value and a correct response time. This is the best we found for a reasonable price.

They send back to the Arduino an analog voltage, which corresponds to the distance between the hand of the user and the sensor. This value will be handled by the card, and sent to the controller LEDs and the computer.

4.1.2 Buttons

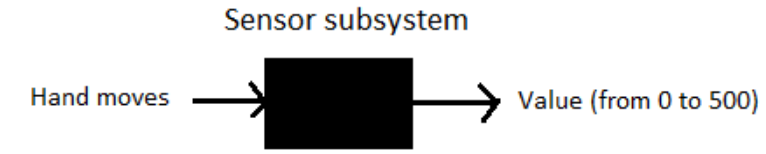
Figure 1

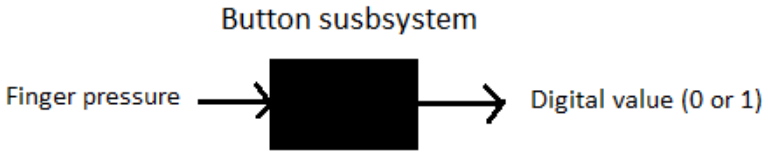
Figure 2

To choose which sensor the user wants to use, and to configure the controller, the user could use ON/OFF buttons (Figure 1) and pushbuttons (Figure 2). The ON/OFF buttons are essentially for the group mode, which enable the artist to gather together sensors. The pushbuttons are for the other functions of the controller, for example the mode selection.

4.2 Subsystem behavior

The following subsystems are controlled by human behaviors. They have a direct interaction with the user (the artist). They have a total control of the object.





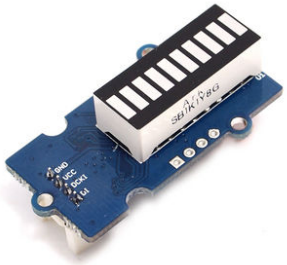
The sensor subsystem receives a move made by the artist (an up-and-down move to modify the value of the sensor) and will send the values defined by the sensor (from 0 to around 500).

The button system is very simple. It waits a pressure from the user, and then sends a digital value (0 or 1) to the Arduino card.

5 Subsystem 3: Display

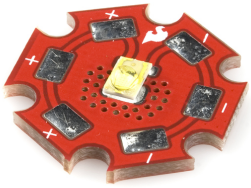
5.1 Hardware elements

5.1.1 LEDs bargraph



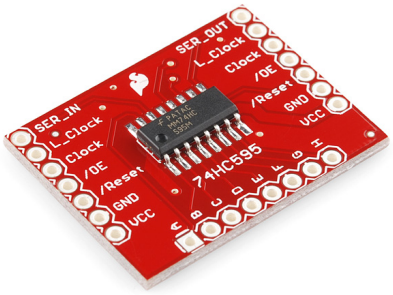
Grove LEDs bar are components made to work with the Grove shield. We use eight of them. Their goal is to display for the user the level of the actual track or effect.

5.1.2 Power LEDs



The eight power LEDs' purpose is to create a visual effect for the audience. They enhance the interaction between the artist and the crowd, which is one of the main goals of our project. The brightness of the eight power LEDs corresponds to the sensor value.

5.1.3 Shift Register 74HC595



This device is used to control the power LEDs described before. It can control several LEDs with fewer wires than the number of LEDs. The advantage is that we don't need 8 wires to control the 8 power LEDs. It has high-noise immunity and contains an eight-bit serial –in, parallel-out, shift register that feeds an eight-bit D-type storage register.

5.2 Software elements

Ableton Live is a digital audio workstation, allowing the performer to play, arrange, remix, mix sounds. It gives access to infinite effects for audio, such as reverb, distorsion, delay, glitch.

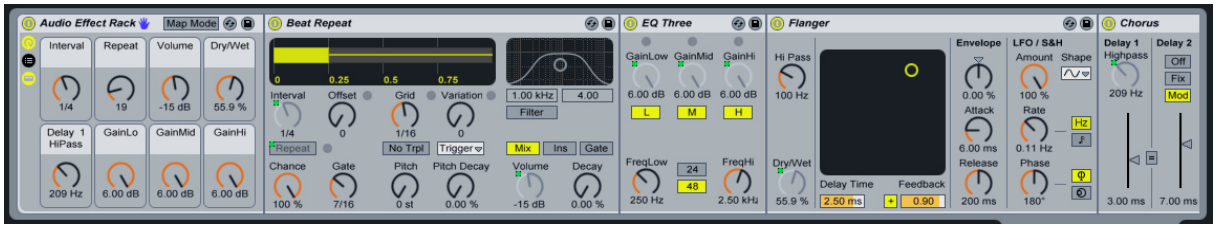
The software can be controlled with external hardwares, using midi communication. Every parameter can be assigned to a device. For example, the play button can be assigned to a button, same for the record button, the volume slider can be assigned to a fader.

Every parameter that is in blue is assignable.

The top left area shows the link from the hardware to the software. (in this example, nothing has been assigned).

KTRL midi controller provides 8 sensors, sending midi messages (we call that "CC message").

Macro parameters:



Macros are inside the Audio Effect Rack. It gives 8 parameters to be assigned to a specific parameter inside the specific effect. Here for example we have :

Macro 1 : Interval from the Beat Repeat Effect

Macro 2 : Repeat On/Off from the Beat Repeat Effect

Macro 3 : Volume from the Beat Repeat Effect

Macro 4 : Dry/Wet from the Flanger Effect

Macro 5 : High pass Filter from the Chorus Effect

Macro 6 : Low range from the Equalizer

Macro 7 : Mid range from the Equalizer

Macro 8 : High range from the Equalizer

The idea for KTRL is to map a proximity sensor to a macro.

For the instant mapping, we had 2 solutions with Ableton, which were the "User Remote Script" and the "Midi Remote Script".

5.2.1 User Remote Script

Situated in C:\Users\x\AppData\Roaming\Ableton\Live\Preferences\User Remote Scripts

It is a text document providing acess to the midi mapping.

```
[DeviceControls]
# The Encoders will control the device parameters (you can also
# use knobs or sliders). Replace the -1's with the CCs sent by
# the respective controls on your controller. You can also set
# the channel for each controller if it differs from the global
# channel (if you leave the channel of an encoder at -1, Live
# will assume that the encoder uses the global channel).
Encoder1: -1
Encoder2: -1
Encoder3: -1
Encoder4: -1
Encoder5: -1
Encoder6: -1
Encoder7: -1
Encoder8: -1
```

Here are the parameters concerning the macro. Encoder1 = Macro 1, Encoder2 = Macro 2, etc. To map the sensor to the macro: instead of "-1", you have to put the CC number from the sensor.

Also, for the recognition, the Input and the Output name should be the same as the controller displayed in the Ableton Midi Settings.

The macros are not the only parameter that can be mapped. In fact, it just give the capability to instant map without going in the midi menu. So, the 8 sensors can be mapped to for example, a volume slider, or a return knob, or the master crossfade, etc.

5.2.2 Midi Remote Script

Concerning the Midi Remote Script. It is composed of 3 python files. The three files are : init.py, config.py, consts.py.

init.py : this file provide the initiation and the implementation of the other files. It calls the script and import it in Live. It also create an instance with all the mapping implementation : concerning the device controls, the transport controls, the volume controls, the controller description and the mixer

options. In our case, the automap is provided with the device controls. It also import the Framework capabilities, which receive the midi message in input and output.

config.py : this file matches the parameters from const.py. It matches the generic encoders with the specific channel to Device controls.

```
DEVICE_CONTROLS = ((GENERIC_ENC1, 0),
(GENERIC_ENC2, 0),
(GENERIC_ENC3, 0),
(GENERIC_ENC4, 0),
(GENERIC_ENC5, 0),
(GENERIC_ENC6, 0),
(GENERIC_ENC7, 0),
(GENERIC_ENC8, 0))
```

Here, Device controls is the parameter for the audio effect rack with GENERIC_ENC1 is the macro 1, ENGERIC_ENC2 is the macro 2.

const.py : this file matches the parameters to the midi message.

```
GENERIC_ENC1 = 16
GENERIC_ENC2 = 17
GENERIC_ENC3 = 18
GENERIC_ENC4 = 19
GENERIC_ENC5 = 20
GENERIC_ENC6 = 21
GENERIC_ENC7 = 22
GENERIC_ENC8 = 23
GENERIC_ENCODERS = (GENERIC_ENC1,
GENERIC_ENC2,
GENERIC_ENC3,
GENERIC_ENC4,
GENERIC_ENC5,
GENERIC_ENC6,
GENERIC_ENC7,
GENERIC_ENC8)
```

This for example, matches the midi message CC16 to macro 1, CC17 to macro 2, etc... If each sensor provides a midi message CCX, this is the file to modify.

Also, for the recognition of the controller : modification are made in config.py :

```
CONTROLLER_DESCRIPTION = {'INPUTPORT': 'KTRL Input (KTRL)',
 'OUTPUTPORT':KTRL Output (KTRL)',
 'CHANNEL': 0,
```

same in init.py :

```
defget_capabilities():
```

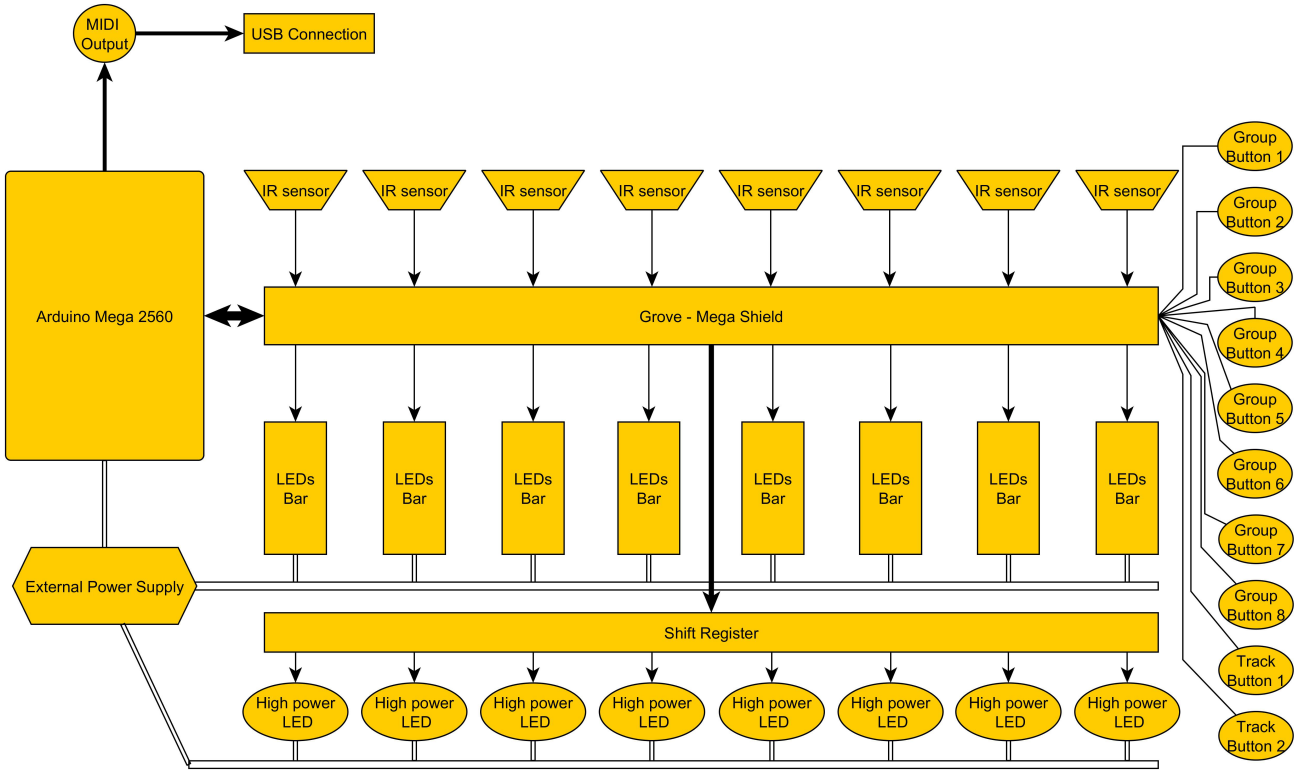
return {CONTROLLER_ID_KEY: controller_id(vendor_id=2536, product_ids=[108], model_name='KTRL'),

The three files are located in C:\ProgramData\Ableton\Live\Resources\MIDI Remote Script\KTRL

Same for the User Remote Script, it gives access to instant mapping with the same options.

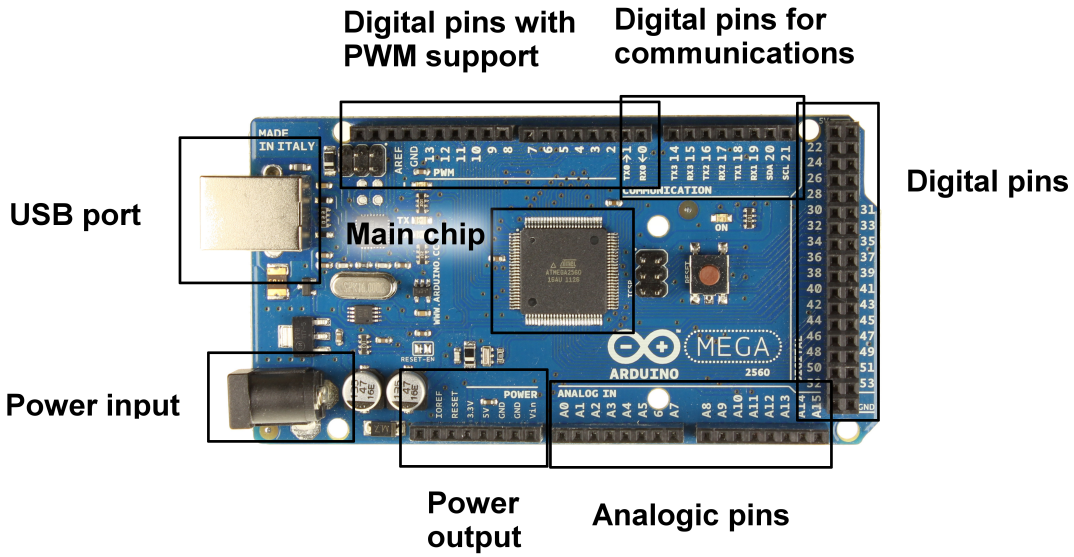
6 Hardware elements

Here is a diagram of all the components that are used for the project. One issue we had is the number of wires. That is why we use the Grove shield and its special connectors.



6.1 Components

6.1.1 Arduino Mega 2560



The Arduino Mega 2560 is a micro-controller board based on the ATmega2560.

6.1.1.1 USB port

This Universal Serial Bus can be used by different ways:

- Transfer the compiled software to the main chip.

    =>Useful for developing.

- Read and write using the USB protocol.

    =>USB normal use while running the software.

- Provide a small amount of energy to the board.

    =>If the board doesn’t need too much energy this can be enough.

    =>For this project this amount isn't enough due to the high number of LEDs and components.

6.1.1.2 Power input

When the board needs more energy than the amount that can be supply with the USB port, this power input can be used to give more energy.

The input must be between 6-20V or the board can be damaged or unstable.

For this project this power supply isn't enough either. So all the LEDs and buttons will use an external power supply and the sensors can be powered by the board (The board can use USB or this power input from the external power supply).

6.1.1.3 Power output

These pins provide power output for external devices.

On the project these pins will provide the power for the sensors.

6.1.1.4 Analog pins

These pins operate analogically between 0 and 5 Volts. They can be used for input and output.

Also these pins can be used like digital pins, to generate software PWM.

The Arduino get the voltage from these pins and convert it. It can also be performed in the reverse manner.

For the project these pins will be used to read the sensors data.

6.1.1.5 Digital pins

These pins operate digitally with 0 or 5 Volts values and a maximum of 40 mA.

The Arduino Mega 2560 model also have sets of specials digital pins with more integrated functionality.

These specials digital pins can:

-Receive and transmit TTL serial data.

-Tiger interrupts on the board depending of the input value.

-PWM (Pulse-Width Modulation) support.

-SPI (Serial Peripheral Interface) communication.

- TWI (Two Wire Interface) communication.

On the project the digital pins are used to check the buttons status, set the LEDs arrays and set the other LEDs on/off values.

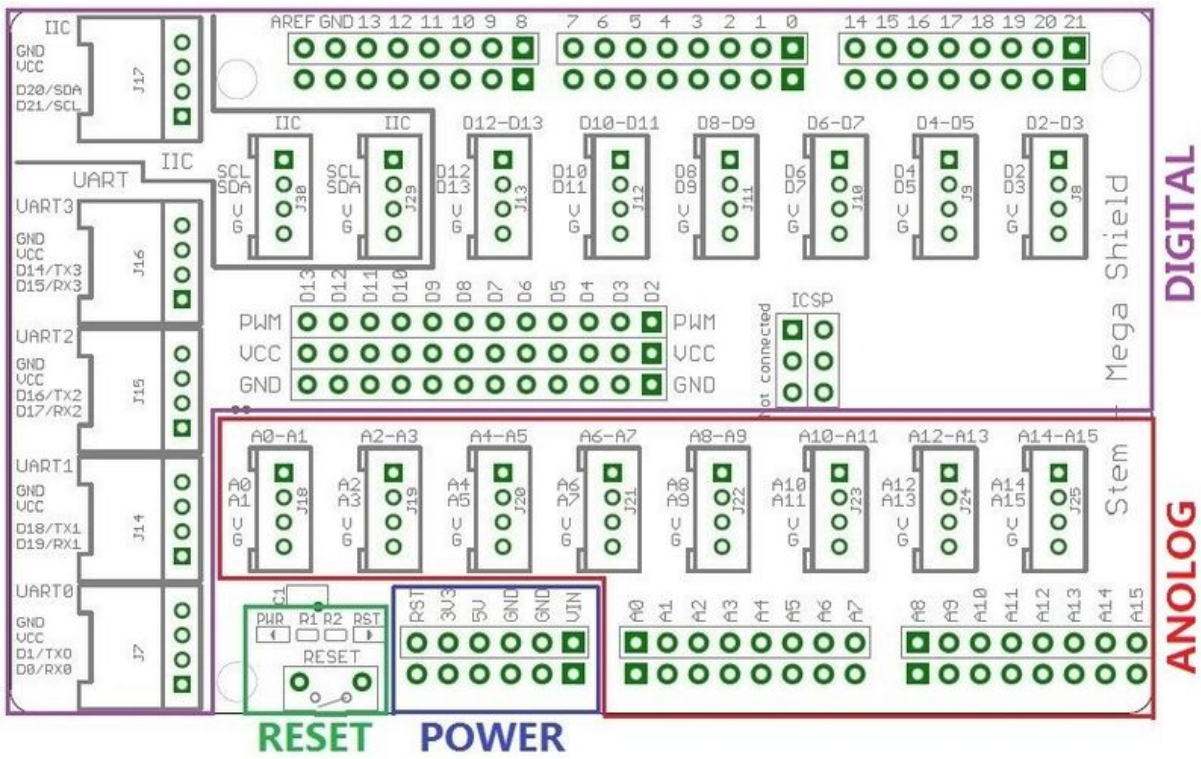
6.1.1.6 ATMega2560 chip.

Is aAtmel 8-bit AVR RISC-based microcontroller with 256KB ISP flash memory, 8KB SRAM, 4KB EEPROM working at 16 MHz can archive 1 MIPS per Mhz.

This chip is the heart of the board where all the operations are done. The rest of the components of the board are there to help or improve the communication with this chip.

6.1.2 Grove - Mega Shield for Arduino Mega 2560

Seeedstudio provides this board. This component is placed on the top of the Arduino board. It function is provide a different way to connect all the inputs of the board.



With this shield Grove connectors can be used to connect the analog and digital pins. The LEDs and sensors use Grove connectors so this shield helps to connect them in an easier and orderly manner.

6.1.3 Sensors

The sensors are one of the most important hardware elements because it does the particularity of our controller.

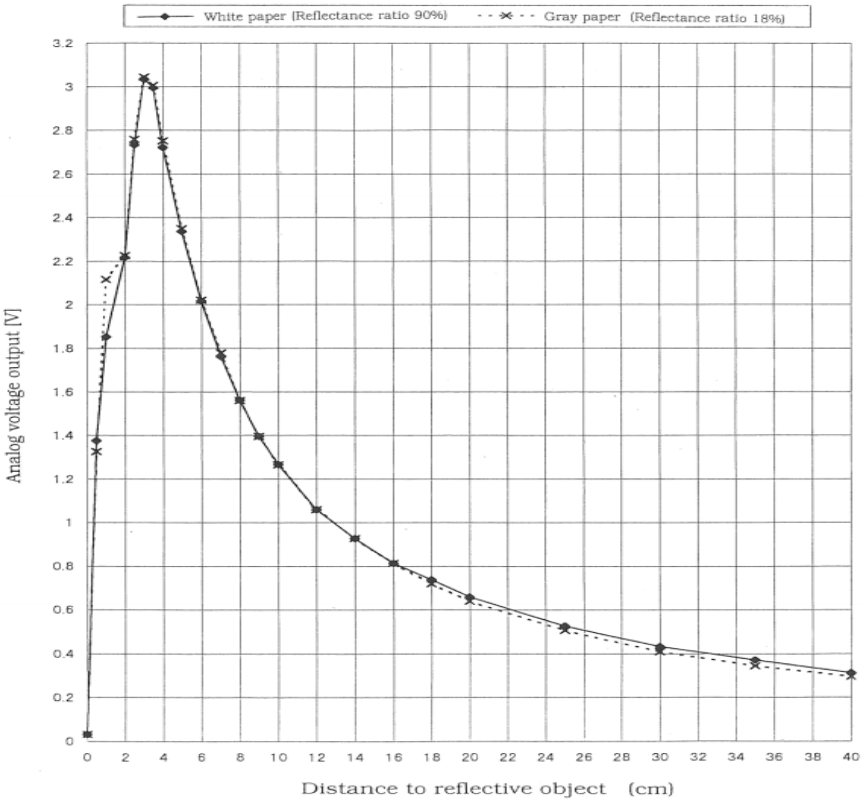
We use the sharp GP2YOA41SKOF.

We choose to put eight sensors on it because an audio effect rack in ableton allows the user to control eight parameters.

Our sensors are distance measurement sensors. They have a measuring distance range from 4 cm to 30 cm. We need to manage a midi value from 0 to 127.

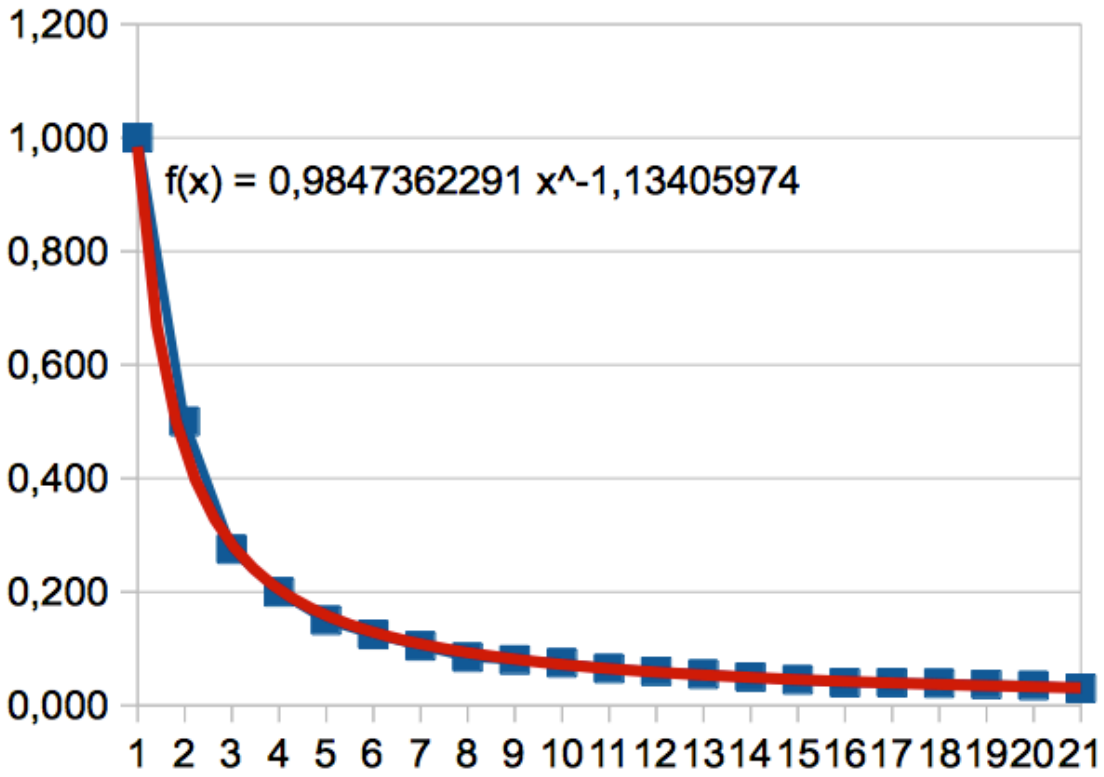
So the accuracy provided by our system is :

A = (30-4) / 127 = 2,05 mm



The sensors analog voltage output will be convert in MIDI by the Arduino card.

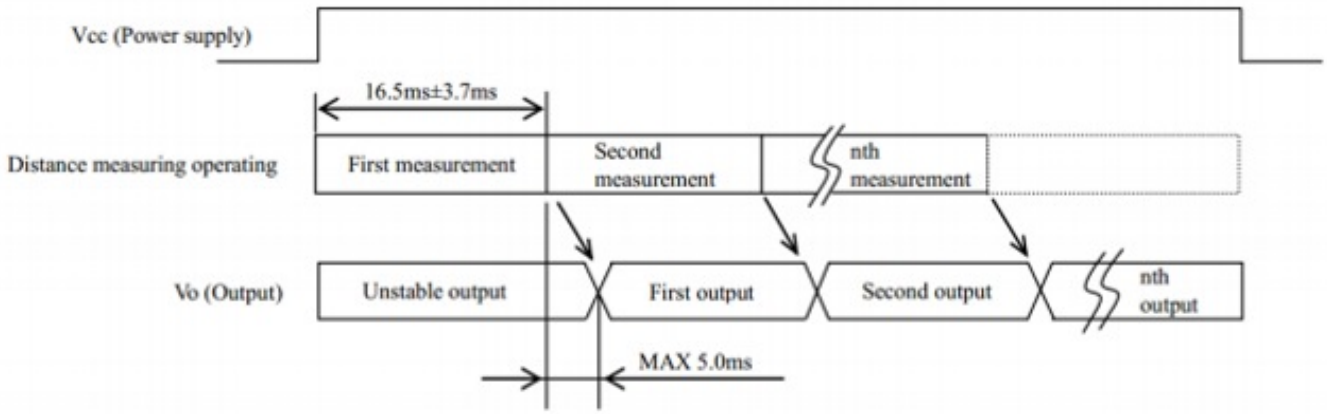
To have to exact distance between the hand and the sensor, we implement an approximation of this function in the conversion algorithm (in **red**the output curve, in **blue** the approximated function used in the code).



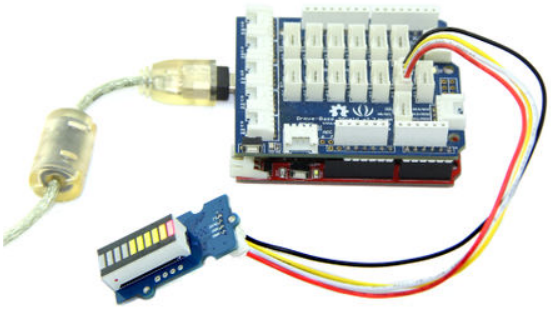
Each sensor is oriented from the top to the bottom; the highest value will be at the top so the nearest from the sensors.

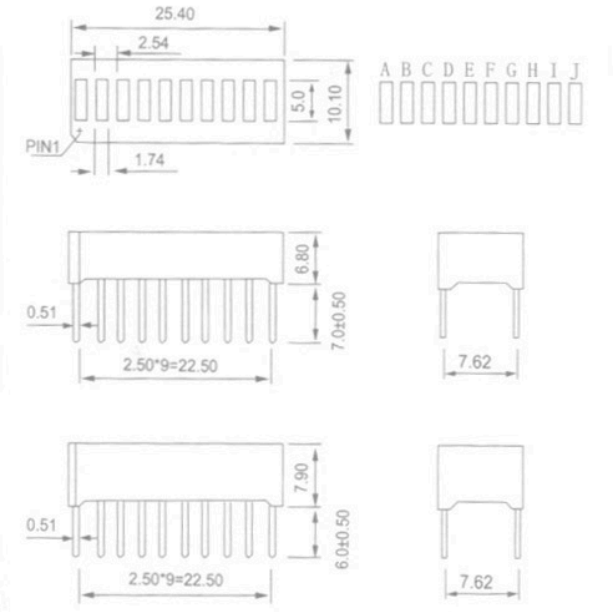
It's also essential to have the lower response time as possible.

Tmax = 16,5 + 3,7 + 5,0 =25,2 ms



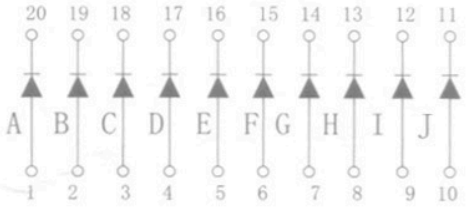
6.1.4 Grove - LED Bar





Seeedstudio provides this LED Bar. This LEDs bars will display the value for his correspondent sensor. This component comeswith a MY9221 LED (by My-Semi) controlling chip. Grove connectors are used to connect this bargraphs.

When all the LEDs are on (8 bars on this project = 80 LEDs) 2A are needed. To support this an external power supply will be used.



The main difference between the MY9221and the famous 74HC595 is that the first one can control up to 12 LEDs, and the command data is on 16 bits. To choose which LED we want to light on, a command on 10 bits is sent (because we are using 10 LEDs).

For example, if we want to light up the last 3 LEDs, we have to send: 1110000000= 896.

6.1.5 Metal Pushbutton

We use stainless Steel Body button with latch switch mode. Also they are equipped with a LED that controllable by separated.

This buttons will control the groups for the sensors. And also can be assigned to do more jobs if the software is configured to do it. The Arduino board can read the state and make changes according to it.

6.1.6 Luxeon Rebel High Power LED

This high power LED can afford a max forward current of 1A in a 2.55V to 3.99V range. Philips provides it.

For the project will be used 8 high power LEDs and the power will be supplied by the same source of the LEDs bars. This LED will have a resistance to avoid the use of so much current but still will be brighter than the usual LEDs

The electrical characteristics at 350 mA and at a 25°C thermal pad temperature are:

Nominal CCT	Forward Voltage V_f [1] (V) Min.	Typ.	Max.	Typical Temperature Coefficient of Forward Voltage [2] (mV/°C) $\Delta V_f / \Delta T_J$	Typical Thermal Resistance Junction to Thermal Pad (°C/W) $R\theta_{J-C}$
2700K, 3000K, 3500K, 4000K, 5000K, 5700K, 6500K	2.55	3.00	3.99	-2.0 to -4.0	10

The current characteristics are:

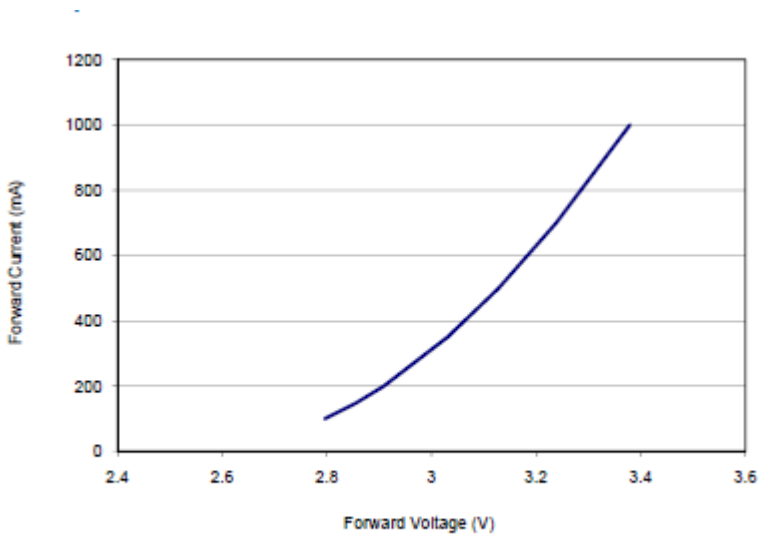
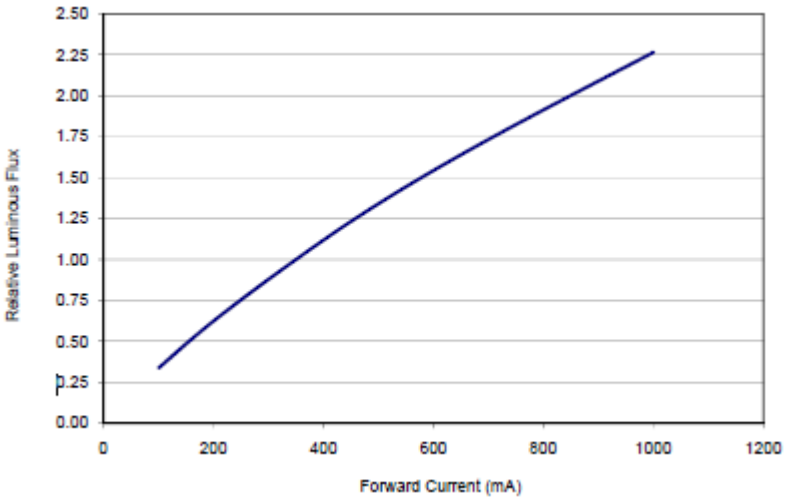


Figure 13. Forward current vs. forward voltage.

Furthermore, the luminous flux is represented by the following curve:



The 74HC595 driver will control these LEDs.

6.1.7 74HC595 Shift Register

The supply voltage must be between 2 and 6V. The input and the output voltage must be more than 0 and less than the supply voltage.

Here is the pinning of the shift register:

- OutputEnable: connected to the ground to active all the output.
- Master Reset: connected to +5V to avoid a reset of the chip.
- Q0 to Q7: connected to the 8 LEDs
- Ds: serial data input, it will command the outputs of the chip
- Two clocks, connected to digital pins of the Arduino card.

PINNING

PIN	SYMBOL	DESCRIPTION
1, 2, 3, 4, 5, 6, 7 and 15	Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7 and Q_0	parallel data output
8	GND	ground (0 V)
9	Q_7'	serial data output
10	$\overline{MR}$	master reset (active LOW)
11	SH_{CP}	shift register clock input
12	ST_{CP}	storage register clock input
13	$\overline{OE}$	output enable input (active LOW)
14	D_S	serial data input
16	V_{CC}	DC supply voltage

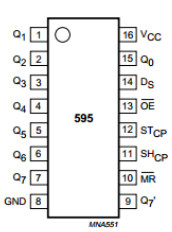


Fig.1 Pin configuration.

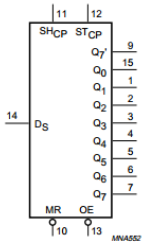


Fig.2 Logic symbol.

An order to the 74HC595 is composed of 8 bits. For example the order 00110111 will light up 5 LEDs. In this case, several logical '1' and '0' are following each other. The shift register can't know if this is a single '1', or several. That is why there is a clock signal. It has the same frequency as the other clock input. Both of them send square signal. At every rising edge of the SHcp input, the 74HC595 knows that there is a new bit to check on the STcp input, which is the one who receives the order.

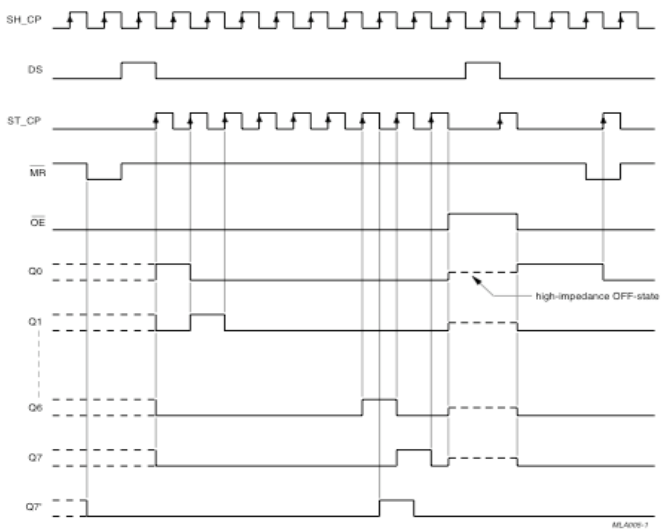
FUNCTION TABLE

See note 1.

INPUT					OUTPUT		FUNCTION
SH_CP	ST_CP	$\overline{OE}$	$\overline{MR}$	DS	Q7'	Qn	
X	X	L	L	X	L	n.c.	a LOW level on $\overline{MR}$ only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6'	n.c.	logic high level shifted into shift register stage 0; contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6') appears on the serial output (Q7')
X	↑	L	H	X	n.c.	Qn'	contents of shift register stages (internal Qn') are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6'	Qn'	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

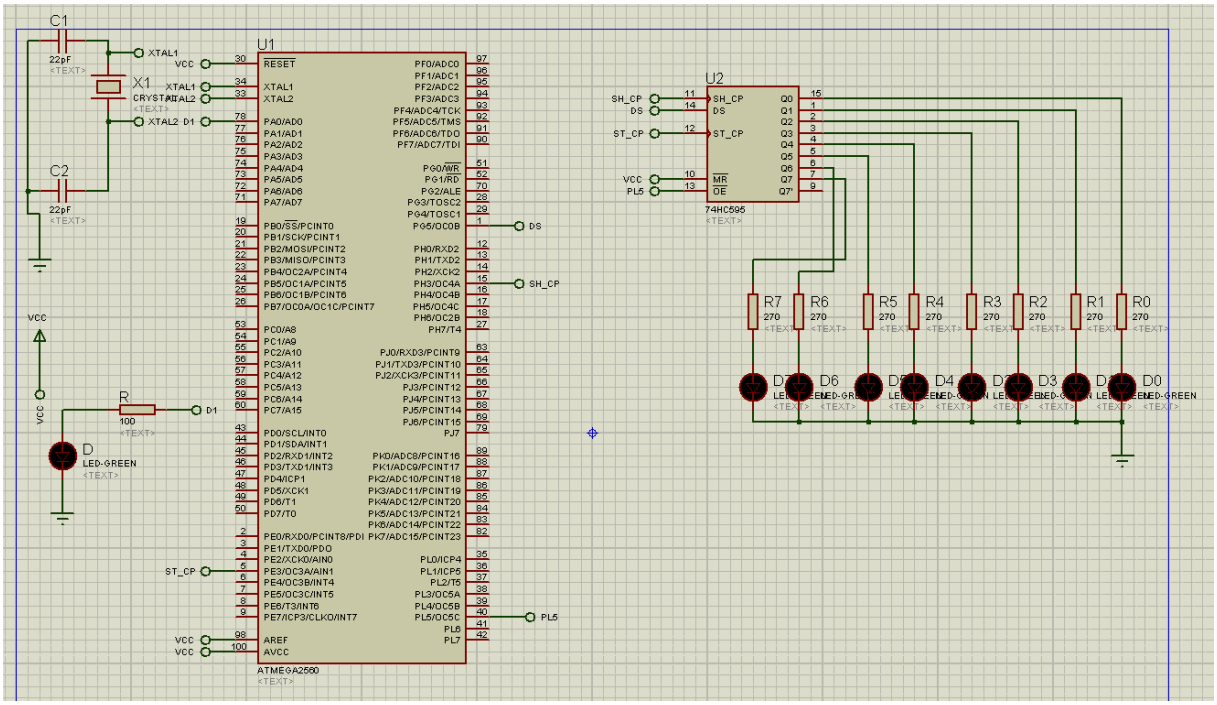
Note

1. H = HIGH voltage level;
   L = LOW voltage level;
   ↑ = LOW-to-HIGH transition;
   ↓ = HIGH-to-LOW transition;
   Z = high-impedance OFF-state;
   n.c. = no change;
   X = don't care.



When the clockPin goes from low to high, the shift register reads the state of the data pin. As the data gets shifted in it is saved in an internal memory register. When the latchPin goes from low to high the sent data gets moved from the shift registers (memory register) into the output pins, which light up the LEDs.

Here is a simulation of the Atmega 2560, with the 74HC595 shift register connected to 8 LEDs:



6.1.8 Mode buttons

Generic buttons without latch.

7 Software element: Max Patch

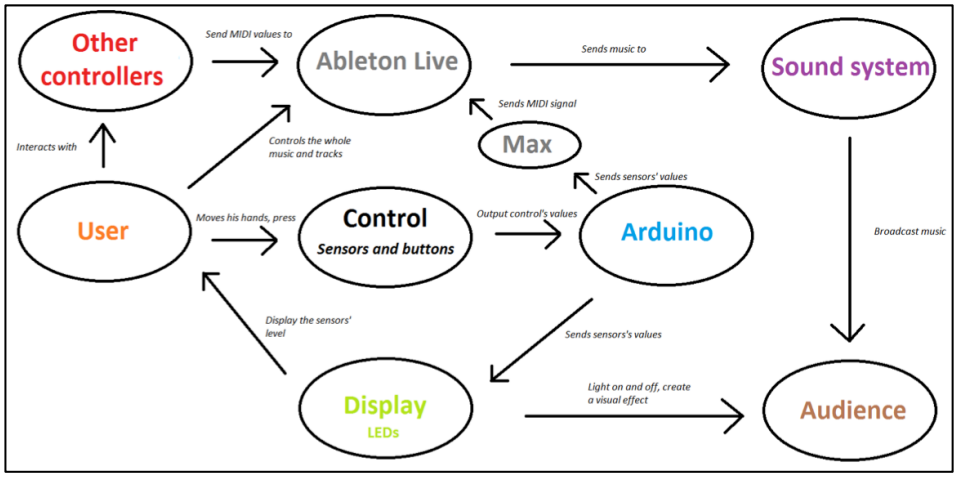
The communication between the hardware and software elements is a key point of our project. The Arduino will send the value we're interested in and it will be translated in MIDI to be therefore used by the user's digital audio workstation.

At first we considered three architectures:

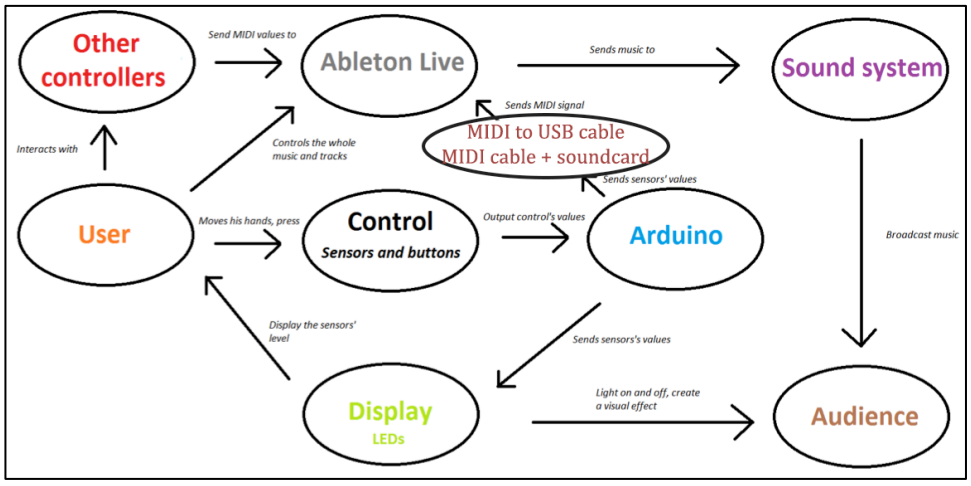
First, we planned to have the parameters values sent over the serial connection from Arduino to the computer and then be received by a Max patch.
This patch would parse the data and build a MIDI Control Change signal then send it.

The user would then configure the DAW of his choice to use these MIDI CCs.

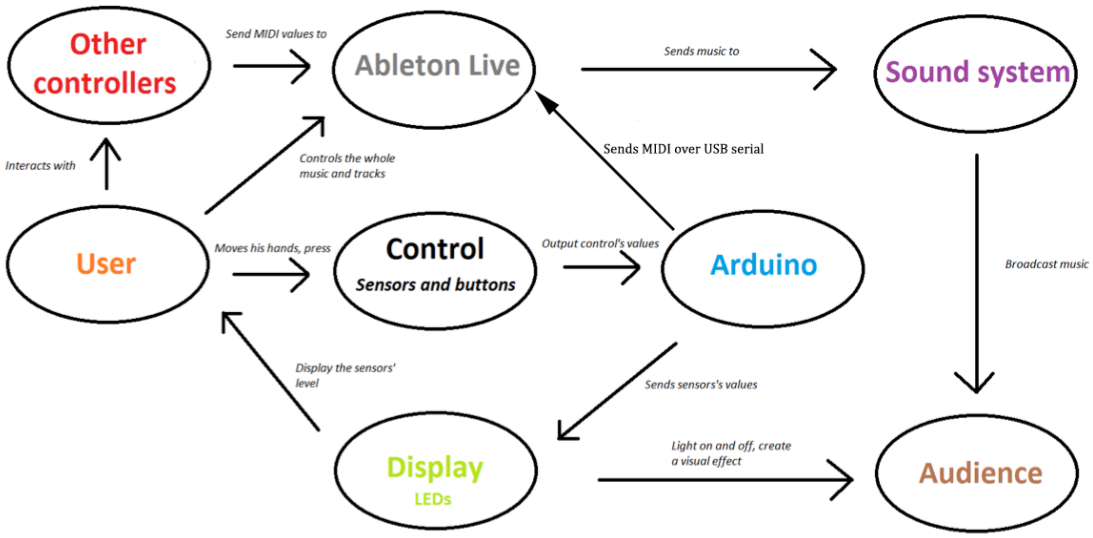
7.1 Architecture



- We decided to send MIDI directly via a MIDI connector/cable to the computer and receive it with whatever DAW user could use.



- Plan C was to send MIDI information over the USB serial connection to be then directly received by the DAW. The MIDI over USB proceeding would be provided by a firmware installed on the Arduino.



7.2 Max Patch

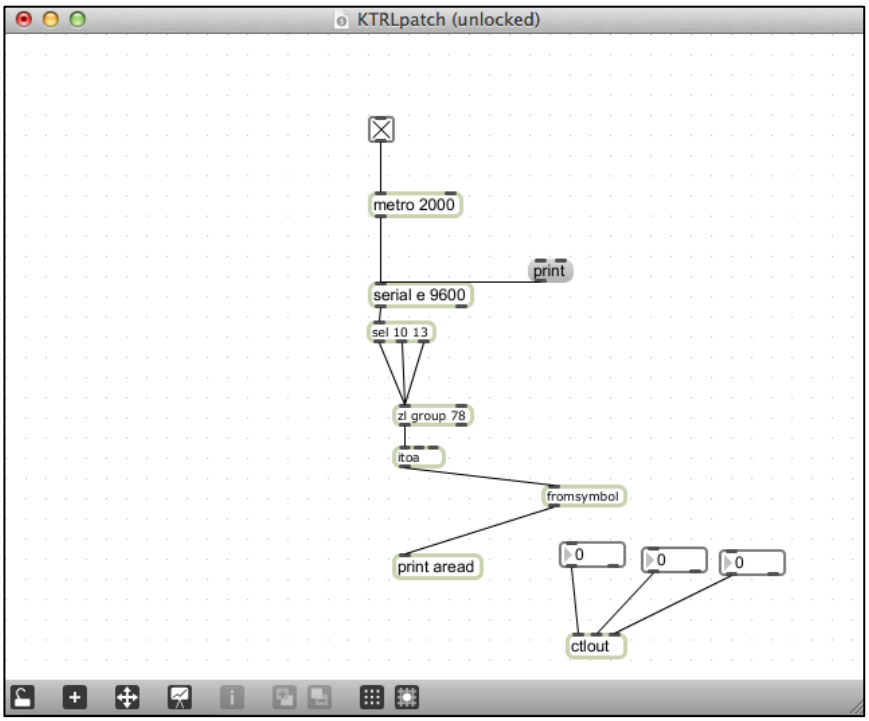
Max is a visual programming language for music and multimedia developed by Cycling'74. During its 20-years history, it has been used by composers, performers, software designers, researchers, and artists for creating recordings, performances, and installations.

The **Max** program itself is modular and has a large user base of programmers not affiliated with Cycling '74 who enhance the software with commercial and non-commercial extensions. Because of its extensible design and GUI, Max is now known as a standard for developing interactive music performance software.

It can send several type of information to and through the computer and so we considered it to be a good candidate to convert our data into MIDI.

The idea was to send information to the computer through the USB serial port,**Max**would then process it and convert it into a MIDI signal to be used by any DAW.

Here is the patch we made to parse the serial information, then convert it into MIDI:



This solution worked well but it needed the user to download the patch and run **Max** to have the controller working. It would have added one more link in the process chain.

We finally decided to give up this idea because we wanted the controller to be the more plug-and-play we could make it so the user would simply have to plug it and… play.

Appendix A. Bibliography

http://arduino.cc/

http://learn.adafruit.com/

https://www.sparkfun.com/

http://www.seeedstudio.com/

https://forum.ableton.com

http://www.synthtopia.com/

http://en.wikipedia.org/

Appendix B. Glossary

See the following documents for a complete understanding of all terms:

- CDC
- TRS

8. Valuation Report

Document “Rapport de valorisation”

Rapport de valorisation
Projet KTRL

Le projet

Le KTRL est un contrôleur MIDI composé de 8 capteurs capables de gérer 8 effets sonores sur un logiciel de musique électronique (comme Ableton Live), via la transmission MIDI (Musical Instrument Digital Interface).

Il possède différents modes, contrôlés par des boutons, permettant de gérer différemment les capteurs utilisés (par exemple un mode permet de reprendre la valeur précédente lors qu'on utilise un capteur). Des barres de LED sont disposés sur le contrôleur afin que l'utilisateur puisse savoir la valeur du capteur (et donc de l'effet) utilisé, et des LED de puissance sont fixées devant le KTRL pour offrir au public un effet visuel intéressant.

L'objectif de ce projet est tout d'abord d'offrir à l'artiste un contrôle plus simple et intuitif d'un contrôleur MIDI, peu d'entre eux utilisant actuellement des capteurs de distance. De plus, nous souhaitons, à travers ce produit, offrir une expérience visuelle au public grâce aux LED frontales, ainsi qu'avec les mouvements de l'artiste qui procurent un dynamisme à la prestation.

Le choix de la valorisation

La valorisation Logiciel Libre nous semblait la plus adaptée pour ce projet. Après avoir effectué des recherches, nous nous sommes rendu compte que le produit réalisé ne serait pas brevetable. Très peu de brevets existent autour des contrôleurs MIDI, car peu de grandes entreprises en produisent. La majorité des projets comme le nôtre sont développés en Open Source.

Nous voulions créer un projet LL pour plusieurs raisons. Tout d'abord, il existe une communauté importante d'utilisateurs de Digital Audio Workstation. Il s'agit de logiciels qui permettent la création de musique électronique, ainsi que la performance live. Le plus célèbre et le plus utilisé est Ableton Live. Nous avons donc posté sur plusieurs forums (Ableton, Synthopia) une présentation de notre projet, ainsi qu'un lien vers notre site qui contient toutes les sources. Nous avons obtenu une quinzaine de réponses (une dizaine d'utilisateurs différents) qui étaient assez partagées. Certains étaient curieux voire enthousiastes, d'autres plus sceptiques, ce qui montre bien l'étendue et la diversité de la communauté.

Les clients

Nous voulions partager notre projet afin d'obtenir des feedbacks. Ils nous permettaient d'avoir des avis extérieurs sur le projet afin d'y apporter d'éventuelles modifications. De plus, créer de la musique électronique est devenu très populaire et assez facile d'accès. Des nombreux petits artistes indépendants se produisent en live. Ils sont parfois rattachés à des labels qui regroupent plusieurs artistes, dans le but de les produire à plus grande échelle. Ils sont nos potentiels premiers utilisateurs. Ils feraient la vitrine de notre produit.

Nous avons contacté dans un premier temps FATCAT et GEDEON, deux artistes français signés sur le label parisien POINCARE RECORDS, encore mineur dans le milieu de la musique électronique française. Ils étaient les invités d'un festival de musique organisé par une association de l'école, No Larsen. Leurs retours étaient positifs et étaient impatients de tester notre produit.

Le retour le plus concluant fut celui de STEPHAN BODZIN, artiste allemand de techno progressive, dont il est un des plus grands représentants. Fondateur du label HERZBLUT, sa renommée est mondiale (125 000 likes sur Facebook). Nous l'avons contacté car c'est un artiste qui propose des performances live qui sortent de l'ordinaire. Il est en quête de créativité et de nouveauté. Nous avons eu un retour positif de sa part, il nous a dit qu'il était « impatient de mettre ses mains dedans ». Nous avons donc eu la confirmation que notre produit pourrait avoir un futur intéressant.

La place de KTRL

Beaucoup de personnes se lancent dans la création de contrôleurs DIY (« Do It Yourself » = objet qui peut être réalisé facilement soi-même). Ils sont plus ou moins aboutis, et il est difficile de rencontrer un grand succès. Les sources sont parfois partagées, et cela nous a été d'une grande utilisé. En tant que bénéficiaires du logiciel libre, nous devons par conséquent en être contributeurs.

Il existe des centaines de contrôleurs sur le marché, qui sont bien sûr tous différents. Alors comment se démarquer ? Chacun d'entre eux ajoute des fonctionnalités différentes pour permettre à l'artiste d'avoir son set de contrôleur sur mesure. C'est en partie dans cette optique là que nous avons créé KTRL. Généralement, un artiste a déjà ses habitudes de marques, et il a sa propre façon de jouer. Certains sont plus originaux et sortent des « standards », c'est-à-dire qu'ils aiment jouer avec des contrôleurs uniques dans leur genre. STEPHAN BODZIN fait partie de ce genre d'artistes là, et c'est pour cela que notre projet l'a intéressé. Les artistes sont aussi généralement curieux et en quête d'inspiration.

La musique électronique est en pleine effervescence ces dernières années. Nous constatons qu'il y a de plus en plus de festivals de ce genre de musique partout dans le monde. Ils sont extrêmement populaires sur les réseaux sociaux. En allant à un concert de musique électronique, on remarque facilement que l'artiste est à moitié « caché » derrière une table sur la scène. On se doute qu'il y dispose tous ses contrôleurs qui lui permettent de piloter, modifier, créer la musique. Mais il n'existe aucune interaction

entre les gestes de l'artiste et la modification du son. On peut faire l'analogie avec les concerts de rock avec des instruments tels que la guitare. Dans ce cas, le public voit très bien l'action du guitariste sur les cordes.

KTRL s'inscrit dans cette lignée, en proposant une nouvelle interaction entre l'artiste et le public à l'aide des capteurs de mouvement.

Distribution du code et avenir

Le code est distribué sur Google Code, plateforme d'échange de codes en ligne. Nous diffusons le lien par le biais de notre site Wordpress sur lequel nous avons publié plusieurs articles expliquant l'état d'avancement du projet. A la fin de celui-ci, nous avons posté un article final récapitulant toutes nos sources, nos impressions sur les composants que nous avons utilisé. Il est préférable de faire comme ça plutôt que de poster plusieurs articles, pour une meilleure lisibilité.

L'open source est un très bon départ pour un projet, l'aide de la communauté et les nombreuses sources sont mises à contribution pour le projet. Cependant, si nous voulions manufacturer notre contrôleur à plus grande échelle, il faudrait revoir totalement le projet. En effet, nous ne vendrions pas un objet à l'intérieur duquel il y aurait une carte Arduino. Bien sûr nous réutiliserions certains composants, mais il faudrait réfléchir à une nouvelle architecture hardware. Ceci permettrait notamment d'optimiser le comportement du système, mais aussi d'en diminuer les coûts pour une production. De plus, il y a toujours la possibilité qu'une compagnie ayant de plus importants moyens financiers tombe sur notre idée. Elle pourrait commencer à la produire et la commercialiser avant nous. Aucun recours ne serait possible.

Liens

Facebook : https://www.facebook.com/KTRLmidi?fref=ts
Wordpress : http://ktrlmidi.wordpress.com/