BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF FOREIGN LANGUAGES

ÚSTAV JAZYKŮ

AUTOMATIZOVANÝ XYLOFON

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR AUTOR PRÁCE	**David Seidler**
SUPERVISOR VEDOUCÍ PRÁCE	**Mgr. Jaromír Haupt, Ph.D.**

BRNO 2018

Bakalářská práce

bakalářský studijní obor **Angličtina v elektrotechnice a informatice**
Ústav jazyků

Student: David Seidler ***ID:*** 177381
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Automatizovaný xylofon

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti realizace automatizovaného xylofonu.
Popište výhody a nevýhody zvolené možnosti.
Navrhněte desku plošného spoje pro automatizovaný xylofon.
Navrhněte program pro ovládací prvek automatizovaného xylofonu.
Navržené rešení realizujte a otestujte.

DOPORUČENÁ LITERATURA:

Barrett, Steven Frank: Embedded systems design with the Atmel AVR microcontroller, LinkMorgan & Claypool Publishers, (2010)

Mike Banahan, Declan Brady and Mark Doran: The C Book, Addison Wesley, (1991)

Colonel McLyman, Wm. T.: Transformer and inductor design handbook, 3rd edition, Kg Magnetics Inc., Idyllwild, California, U.S.A., (2004)

Termín zadání: 9.2.2018 ***Termín odevzdání:*** 25.5.2018

Vedoucí práce: Mgr. Jaromír Haupt, Ph.D.
Konzultant: doc. Ing. Pavel Šteffan, Ph.D.

doc. PhDr. Milena Krhutová, Ph.D.
předseda oborové rady

Abstrakt

Tato bakalářská práce se zabývá návrhem a konstrukcí automatizovaného metalofonu. V teoretické části Bakalářská práce pojednává o možnostech realizace zařízení a popisuje použité komponenty. Tyto komponenty popisuje pomocí blokových diagramů a schémat. Praktická část popisuje konstrukci výsledného zařízení. Obsahuje popis návrhu a konstrukce desek plošných spojů a montáž výsledného zařízení.

Klíčová slova

Metalofon, solenoid, ATMega2560

Abstract

This Bachelor’s thesis deals with a design and construction process of an automated metallophone. In the theoretical part, the thesis covers possibilities of the device’s realization and describes its components. These components are described using block diagrams and schematics. The practical part illustrates the construction process of the device. It describes a process of designing and constructing the printed circuit boards as well as the assemblance of the device.

Key words

Metallophone, solenoid, ATMega2560

SEIDLER, D. *Automatizovaný xylofon.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 51 s. Vedoucí bakalářské práce Mgr. Jaromír Haupt, Ph.D..

Contents

List of figures

Introduction

The aim of this thesis is to describe the process of designing and constructing an automated xylophone. Rather than providing a step-by-step guide on the topic, it describes the comparison of various considered methods.

The thesis is divided into sections. *State of the art* section presents pre-existing examples of similar devices. *Previous attempts* and *motivation* sections sum up the author's motivation and the possible reasons of failures of the project in the past. *Basic overview* is a preface to the technical overview of the process as it provides background information. The device is then described from a strictly functional perspective in *Technical overview from control perspective section* illustrating the components as black boxes with a specific function and parameters. Next section - *Technical overview from electrotechnical perspective* - approaches the electrotechnical schematics and their components of the aforementioned black boxes. These sections represent the theoretical background of the device.

The practical part of the project is described in the *Construction process* section. The narrative shifts to a description of a process in *Construction process* subsections. The thesis address various methods of assembly of the metallophone in subsection *Actuators*. The subsection *The stabilizer circuit* involves a series of measurement that lead to the requirement of the stabilizer circuit. *The switching circuit* subsection illustrates the process of designing a printed circuit board. *The sensor sub*section covers the means of attaching the sensors to the walls of the metallophone to provide accurate results. The subsections - *The connectors, Connecting the resonance chambers* cover the aesthetics of the device.

The section *Programming* explains the possibilities of tests and trial runs using a programming language in *Programs for testing and adjusting* subsection. Lastly, the subsection *Final program* involves a description of the final program.

1 State of the art

This section evaluates the design and construction of various automated metallophones. It was found out that the process of automating a xylophone can be accomplished with emphasis on different aims.

One of such aims is to focus on the physics - specifically lateral vibrations of a bar. A solution by nerdkits team presents calculating the dimensions of the bar of the metallophone using a lateral vibrations formula and cutting aluminium bars accordingly with a milling machine. Their solution involves handcrafted solenoids with neodymium magnets on top of the core to increase efficiency. [1]

Another aim is to enable the metallophone to communicate with a computer in real time. The control element from this device drives the actuators via a MIDI connector. Due to a MIDI controller, the metallophone can be controlled via a notation rather than code. The components of this device are soldered on perfboards. Perfboard is a versatile type of a printed circuit board. [2]

The focus of the next evaluated device is interactivity. An automated metallophone constructed by a group of students from Northwestern University consists of a button pad and LCD display. Their device operates in two modes. *The real time* mode enables the user to play the metallophone in real time using the button pad. *The playback mode* provides a function of storing notes in a memory of a control element. User is able to edit the notes of the played sequence via the button pad and the LCD display. The circuits are realized on a breadboard rather than printed circuit board. This indicates that the device is meant to be temporary, because breadboards do not provide the level of fixation necessary for long lasting devices. Breadboards are usually used for prototypes. [3]

2 Motivation

This section describes the practical use of the automated metallophone. For the last few years, I have been working in an escape room industry. Escape rooms are used as a teambuilding activity. The participants of such room are locked within and have to figure a way out of the room. The way out usually follows some storyline, and the rooms themselves are decorated according to the theme of the room. The way out is usually complicated with some riddles, clues and puzzles. Therefore it is convenient to use ordinary objects, like a metallophone, and redesign them into functional parts of the room that yield progress for the participants. One example of a such procedure from a real room would be a rotary telephone that has the rotary input connected to a control element, which plays a message through a speaker of the receiver when the right number is dialed. This example can create an illusion of a dialogue.

These are not real puzzles - there is nothing puzzling about dialing a number - but rather ways to enhance the interactive aspect of the room. Such objects can help the participants to become more focused on the story.

3 Previous attempts

The first attempt to realize an automated metallophone was left unfinished. The attempt was based on a different metallophone. The frame of this metallophone caused a few of the bars to be inaccessible. The realization of the switching circuit and the process of attaching the solenoids would not be feasible without altering the frame of the metallophone. Therefore, a new metallophone with this particular feature in mind was chosen for this project.

Motivated by the craftiness of the nerdkits team, the previous attempt used homemade solenoids. It is feasible to construct solenoids homemade, but there is one particular feature that has to be considered carefully - the frame. In this case, the frame of the solenoids was loose and the core was then unstable.

The previous attempt used magnetic cores to increase effectivity. The space between the bars of the both metallophones is negligible. This caused the magnetic cores in the loose frame of the solenoids to alter the trajectory of the core. It was anticipated that to begin the project from the beginning but with these new experiences is the correct choice.

4 Basic overview

This section is a preface to the technical overview. Xylophone is an instrument that consists of bars that resonate on a frequency determined by the dimensions of the bars. Larger bars resonate on lower frequency.

From an etymological point of view, xylo- means of wood. Therefore it is wrong (but common) to call an instrument without wooden bars a xylophone. The instrument that is being dealt with has metal bars, thus the correct name for it would be a metallophone or a glockenspiel. Hereby I apologize for any inconveniences.

The number of bars defines the range of the notes. Commonly, metallophones consist of tones only, or of tones and semitones. This project involves a metallophone with tones and semitones to assure unlimited number of scales and melodies. The range of the metallophone covers twenty-five bars. The number of tones is fifteen and the number of semitones is ten. The tones and semitones are in separate resonance chambers.

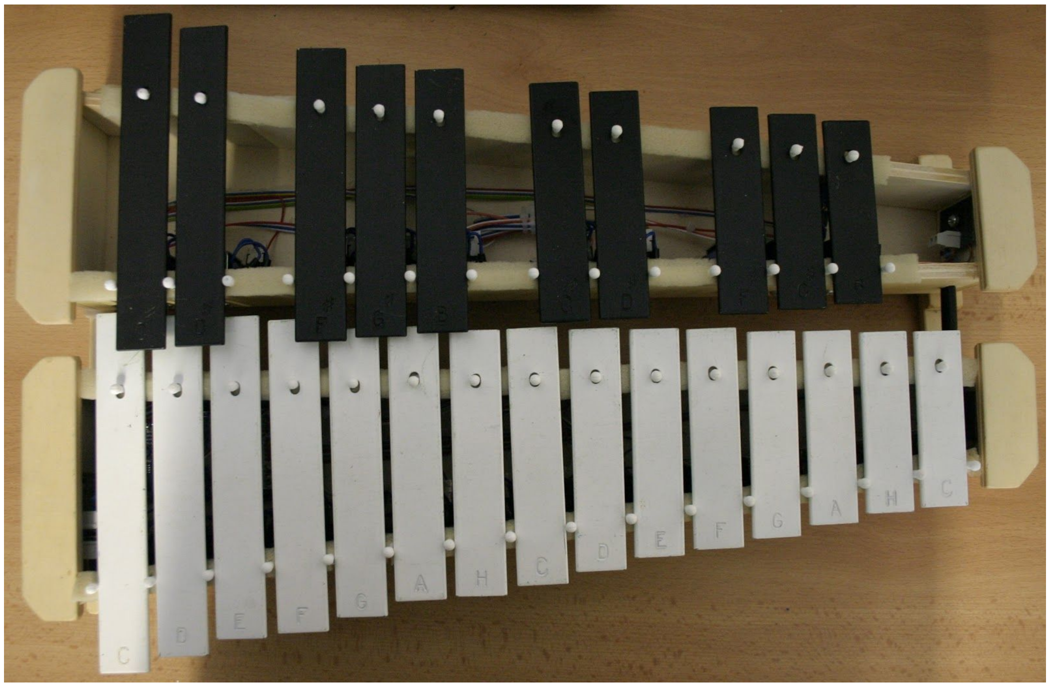

Figure 1 - The device

5 Technical overview from control perspective

This overview describes the device using block schematics. Block schematics are used to describe a function of a circuit rather describing a function of components of the circuit individually. The circuits are regarded as black boxes, where only the relation between input and output is known.

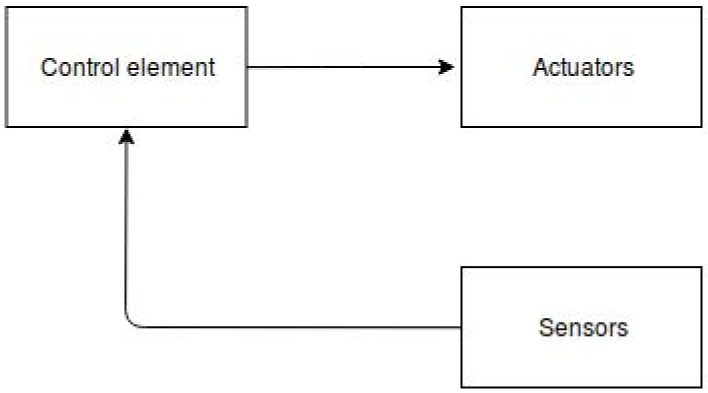

Figure 2 - A simplified block diagram

This is a simplified block diagram of the device. The control element drives actuators based on data from sensors. The control element switches between two phases. The first phase acquires data from the sensors and the second phase controls the actuators.

The actuator simulates a mallet. It is the visible part of the device. It should be able to resonate the bar of the metallophone.

The sensor detects when a bar is resonated by acquiring loudness of sound from its surroundings. If carefully positioned and tuned, it is able to determine which bar is played.

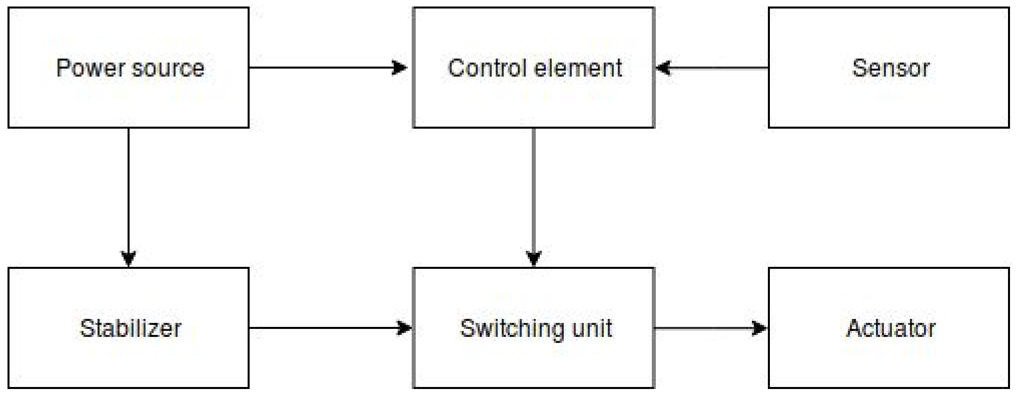

Figure 3 - An extended block diagram

This is an extended block diagram of the device. The control element does not control the actuator directly. The demand of current would be unsuppliable directly from the control element. The actuator is therefore controlled indirectly via a switching unit. Also, the control element and the actuators have different demands on the voltage level of the power source. The added stabilizer block allows the control element and the actuator to be connected to a shared power source.

This block diagram illustrates the data signal flow (from the control element to the switching unit and from the sensor to the control element) and the supply power flow.

6 Technical overview from electrotechnical perspective

This section depicts the device one layer deeper, focusing on the components of the circuits. This enables the description of features of the circuits.

6.1 The switching circuit

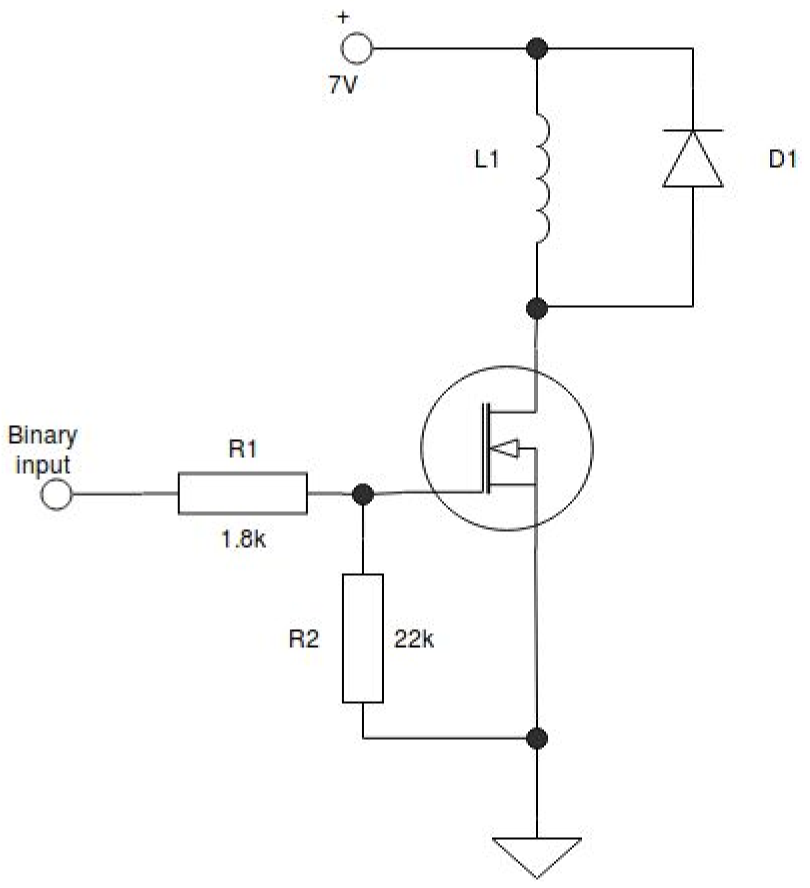

Figure 4 - The switching circuit schematic

The switching circuit controls the high current solenoids with low current data signal. The switching component is a MOSFET transistors. When logical one is sent to the gate terminal of the transistor, the transistor acts like a closed switch and allows current to flow through the solenoid. When logical zero is sent to the gate terminal of the transistor, the transistor acts like an open switch and no current is allowed to flow through the solenoid.

There are also some protective elements in the circuit. When the solenoid (or any other inductor) is disconnected from the power source, it induces a voltage spike. This voltage spike could damage the MOSFET transistor. The MOSFET transistor divides the data flow and the power source flow. If it becomes damaged, the power source would become directly connected to the control element pins. If that had happened, the pins of the control element would be inoperable. The D1 diode allows the induced power to dissipate in wires and the diode, protecting the MOSFET. A diode connected to an inductor in counter paralel is commonly known as a Flyback diode.

The resistor in series with the gate terminal of the transistor - R1 - protects the transistor by decreasing the current flow. The resistor that is connected from the gate terminal of the transistor to the ground - R2 - is grounding all of various noise signals that might be generated in wires or the control element.

6.2 The stabilizer circuit

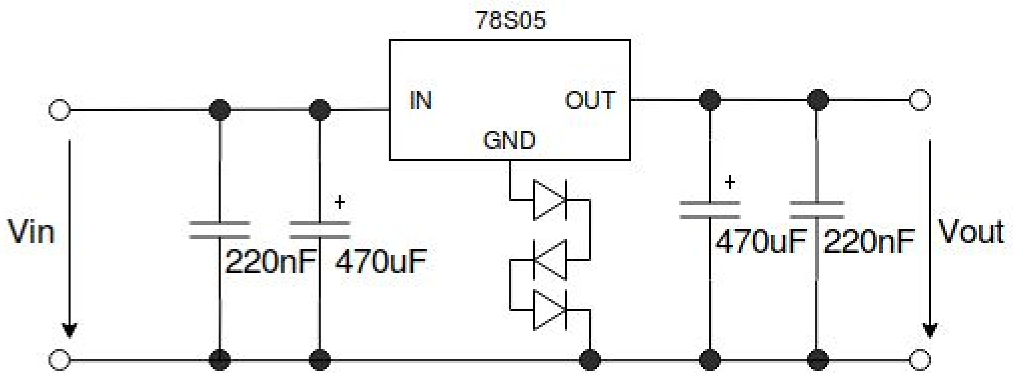

Figure 5 - The stabilizer circuit schematic

The stabilizer circuit adjusts the voltage of the source. The integrated circuit 78S05 decreases the voltage of the power source from 8-35V to 5V. The integrated circuit is depicted as a black box in this schematic. It consists of various components that are sealed in a package. The schematic of 78S05 is in its datasheet.[4] Due to the package, the components are prevented to be switched, removed or otherwise altered. Therefore the 78S05 is conceived as one component.

The 78S05 tends to oscillate. The common schematic of the 78S05 stabilizer recommends blocking capacitors to prevent oscillations. Two ceramic 220nF condenders and two 470uF electrolytic capacitors are attached to the circuit to comply with the recommendation.

According to trial runs mentioned in the construction process section, the desired output voltage is 7V. To increase the output voltage, diodes are used. These diodes increase the output voltage by lifting the ground of the integrated circuit. For this to function, the subsequent circuits have to be connected to the lifted ground of the integrated circuit and not the ground pin of the integrated circuit directly.

6.3 The sensor

A sensor should be capable of detecting when a bar is struck without delay and with precision. Due to a limited amount of space between the bars, the main source of noise is generated by the adjacent bars. The possibilities considered were: a microswitch, a microphone (analogue), a microphone (digital), a vibration sensor.

A microswitch would be immune to the noise since it does not acquire the sound but the direct vibrations of the bar. If the microswitch is directly touching the bar, it damps the vibrations of the bar. It is anticipated that the microswitch could be tuned in such a way that the damping would be negligible, but it was predicted to be a possible weak point of the device, since the microswitch would rely on a fixed position to a great extent.

A microphone module with digital output would be a logical choice, because the information acquired by the sensors should be binary. The microphone module can be tuned with an inbuilt potentiometer, partially fixing the problem with playing the adjacent bar. The dynamics of the strike remains to be a problem. Hard hit on the adjacent bar provides similar value as soft hit on the measured bar. To increase the focus on the measured bar, the microphone should be close to the bar. Then the measured bar is even more amplified, and if the potentiometer is tuned accordingly, the result is expected to be acceptable.

The microphone module with analogue output is more complicated to process. It creates another layer of control and as a result makes the process of data acquisition less straightforward, but more precise. The chosen control element has analogue inputs and is capable of converting analogue signal into 2^10 signal levels. Therefore, it is possible to obtain same signal qualities as with the digital microphone module, but in 1024 combination rather than 2. It is possible to create another level of detecting noise - an *IF* function can set a threshold value. This way it is possible to tune the sensor even more precisely, and therefore obtain superior results. That is why this method was chosen for the device.

The vibration sensor was considered, but not tested, since the microphone module with analogue output yields sufficient results.

6.4 The power source

Choosing the correct power source is inevitable. The requirements are: sufficient amount of power for the solenoids - the power demand increases with the number of solenoids enabled to be active simultaneously - and stable power supply for the control element. The options are: two separate sources with common ground, one for the control element and one for the actuators, or one source for both elements. Since the solenoids are controlled in pulses, the source should withstand the momentarily increase in power demand. Generally, when the current demand is exceeding the maximum value of the power source, the voltage starts to decrease. These voltage drops are not having a negative effect on the solenoids, but the control element should be provided enough power at all times. Insufficient power supply causes the control element to be momentarily inoperable. The control element in this device - Arduino Mega 2560 - can be powered in three ways. First one is via an USB connector, second is via the input supply connector and the third is via *Vin* pin.

One option would be to supply the control element with a stable 5V through the USB connector. Voltage of 5V is insufficient to drive the actuators, therefore this power source is sufficient only for the control element power consumption.

According to the manufacturer, the input supply connector should be powered by 7-12V. The resistance of the actuators is about 6 Ohms. The current flowing through the actuators can be computed as $I = V/R$. Switching on multiple actuators multiplies the current demand.

This problem is partially solved with the stabilizer circuit. Since the noise caused by the solenoids decreases with lower voltage, it is convenient to stabilize the voltage at the lower level. The Ohm's law demonstrates that with lower voltage, the current decreases. For a voltage level of 7V, the current flowing through the solenoid is 1.16667 amp. The lower demand of current means that the voltage drop of the power source decreases and the control element is supplied steadily.

Even though playing two bars at once is now acceptable, playing more than two bars at once still might cause instabilities of the control element.

7 Construction process

7.1 The actuators

To resonate the bar of the metallophone, the solenoids were fixed underneath the bars. Each switching circuit had been connected to source voltage and the common ground, and each solenoid was connected to its switching circuit. It was required to find a sufficient space for the control element and the stabilizer.

Two different approaches were considered. One was to construct a panel with the switching circuits and the solenoids. This panel would be easy to disassemble when necessary. Since the frame of the metallophone is wooden, the panel is wooden as well. However, after assembling the panel on the metallophone it was found to be aesthetically poor. The second approach was to put all of the components inside the resonance chamber of the metallophone. The space in the resonance chamber is limited. The dimensions of the printed circuit boards of stabilizing and switching circuits were measured to fit the resonance chamber, but the control element's dimensions were predetermined by the manufacturer. If the control element had not fit the resonance chamber, it would have been possible to build a control element printed circuit board with the necessary components only (ATMega2560 processor, bootloader, etc.). The space of the widest area of the resonance chamber is sufficient for the control element, therefore constructing a printed circuit board for the control element was not necessary. The stabilizer dissipates heat continuously, which demands a space for a cooler.

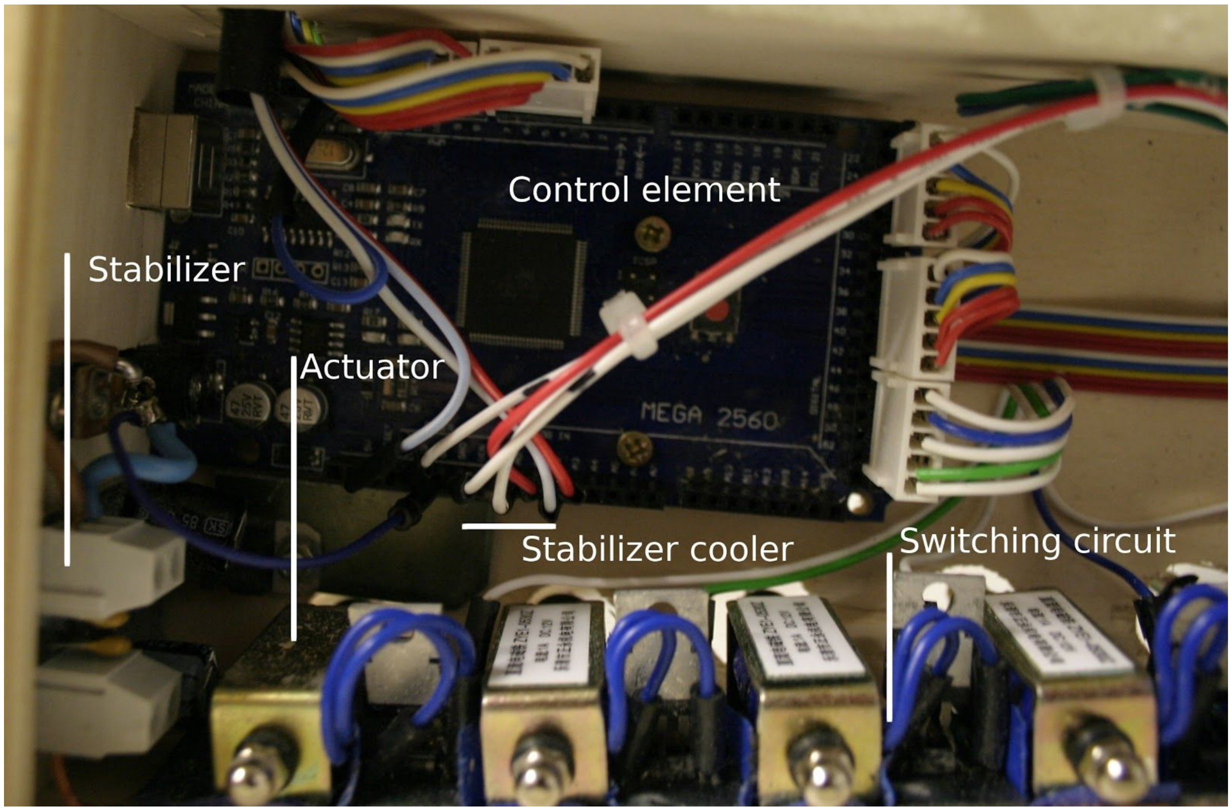

Figure 6 - The resonance chamber

7.2 The stabilizer circuit

To find the adequate voltage stabilizer, the adequate level of output voltage had to be discovered. There is no datasheet for the solenoids. The only parameters available in English are 12V and 1A. According to Ohm's law, the resistance of the coils should be equal to R = V / I. That is 12 Ohms. A control reading on an ohmmeter carried a value of 6 Ohms. According to the Ohm's law, the current consumed by the coil would be 2A. To find the correct value of source voltage, an adjustable power source aws used. With the adjustable power it was possible to run a series of trial runs and obtain information of the minimum voltage that the solenoid required to strike a bar and the level of noise caused by the solenoid in the process. After the series of trial runs, it was determined that a voltage level of around 7V yields optimal results. The L78S type of integrated circuits offers 5V and 7.5V voltage stabilizers. After minor adjustments via the stabilizer circuit, both of these integrated circuits would be suitable. In this device, a 5V voltage stabilizer - L78S05C - had been chosen and the output voltage was fixed to 6.8V.

During the later stages of the construction process it was discovered that the resistance of one of the solenoids is 12 Ohms. It is probable that this was the intended value of the resistance. Because the current flowing through the one solenoid is half of the current flowing through the rest of the solenoids, the one solenoid requires increased pulse width to operate.

7.3 The switching circuit

The recommended approach to designing the switching circuits was to be design a printed circuit board (PCB) and let a professional company build it. This would provide a possibility of a multi-layered PCB, which is more efficient in the limited space of the resonance chambers. Due to a lack of resources and a personal motivation to be a part of the process of PCB construction, this option was disregarded.

Another option was to avoid designing the PCB, but use a prototype shield instead.

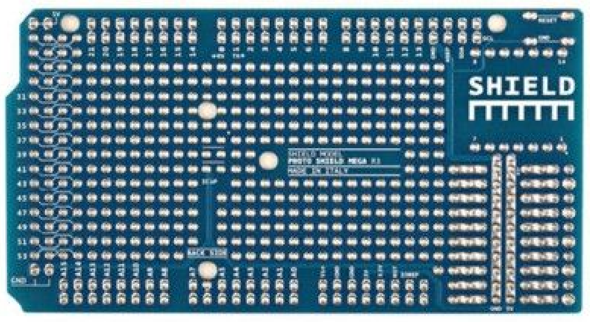

Figure 7 - A prototype shield

The layout of the prototype shield has been designed with the Arduino Mega microcontroller board in mind, thus it can be fixed on top of the control element. This solution would save a great amount of space. This solution was not chosen, because the design would probably be illegible - with a MOSFET transistor, two resistors and a diode for each actuator, the total amount of components is one hundred. More than one level of the prototype shield would probably need to be established, and that seems to be a difficult task for an option, which advantage lays in its lack of difficulty.

To construct a PCB, one has to design the PCB first. The PCB design is derived from the schematic. There are many great softwares for that (e.g. Eagle) that simplify the designing process. Constructing a separate switching circuit for each actuator was chosen, consequently the switching circuit is great in numbers, but not in complexity. Using a software was not necessary and the design has been sketched on paper.

The unprocessed PCB is a laminate board with a copper layer. The design is drawn on the copper layer - there are various technologies available, but in this case the design was drawn using wax. The board was then placed in a solution (FeCl3 - iron trichloride) that etched the copper that was unprotected by the wax. This approach is rather imprecise since it relies heavily on one's ability to draw with wax.

More precise - and the one intended - method to make a PCB is to use a photoplotter. The photoplotter irradiates the design onto the PCB board. This way, the conductive copper lines can be thinner comparing to the wax method. Unfortunately, the photoplotter was not available for the project.

After the PCB was soluted and cleaned of the remaining iron trichloride, it was further processed by drilling, and soldering the components. The components were soldered in order of their heat resistance, therefore the resistors were soldered first and the transistors last.

Additionally, the terminals of the diodes and one of the resistors were turned to the component side of the PCB to build pins for solenoid's wires and the data jumper.

Lastly, each switching circuit was connected to the ground terminal and the positive voltage terminal of the stabilizer via buses. The buses had been soldered to the switching circuit and were fixed to the walls of the resonance chamber by the actuators. The actuators were insulated from the buses to prevent short circuits.

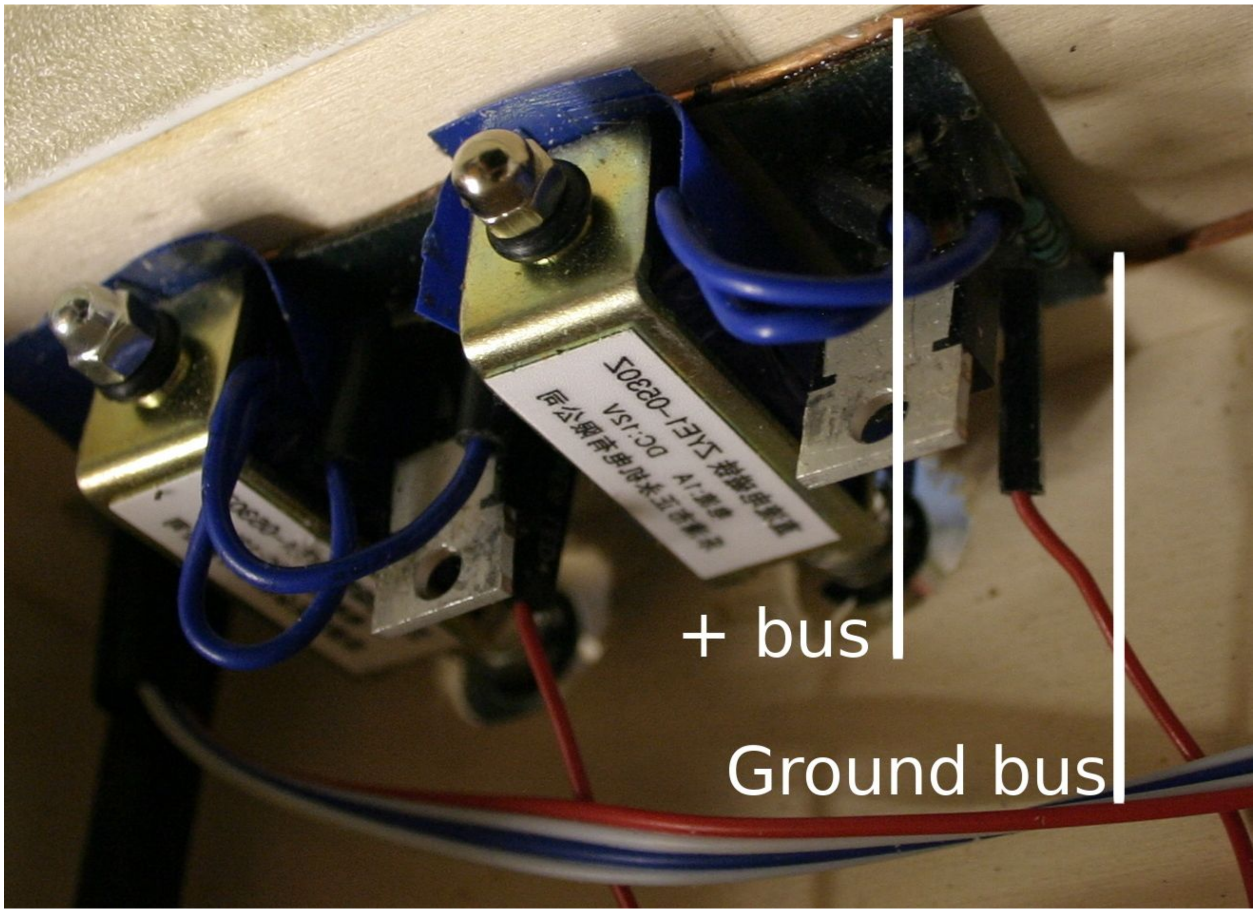

Figure 8 - The power buses

7.4 The sensors

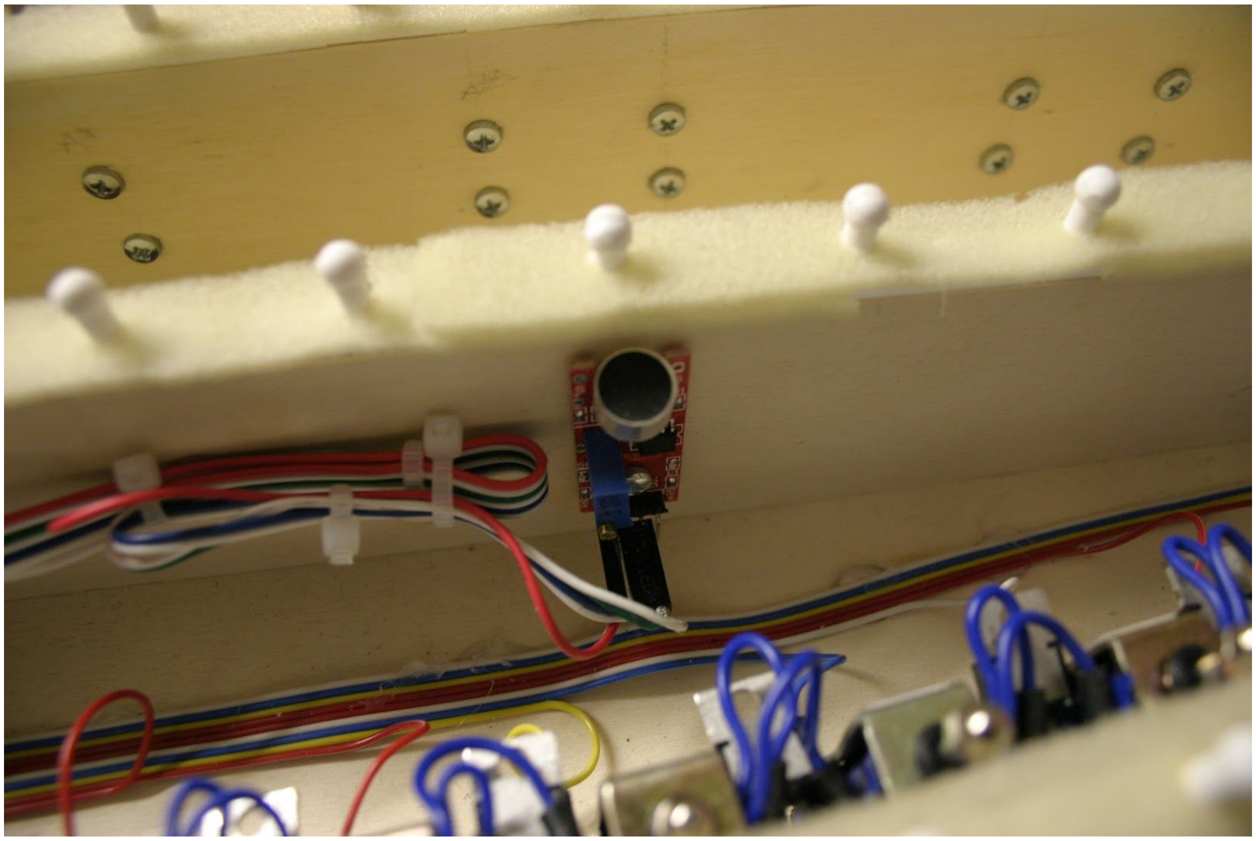

Figure 9 - The microphone module

The wall of the resonance chamber consists of: the wooden wall, plastic layer with holding elements for the bars and a foam layer for the undamped oscillation of the bars. The sensor was placed under the plastic layer. The distance between the measured bars prevents overlaps of the measured signal. Greater distance simplifies the adjustments of the microphone modules.

7.5 The connectors

The number of bars demanded a concise solution of connectivity. Each switching unit requires a data wire. That is fifteen wires for the tones chamber and ten wires for the semitones chamber. The length of the wires should be optimal to simplify further maintenance. Flat wires (fifteen and ten) were chosen for the data flow. The wires were soldered to a five pin connector at one side and clamped to jumper at the other.

The microphone module wires were clamped to jumpers at one side and to pins on the other. For the sake of flexibility of sensor's placement, their wires are prolonged.

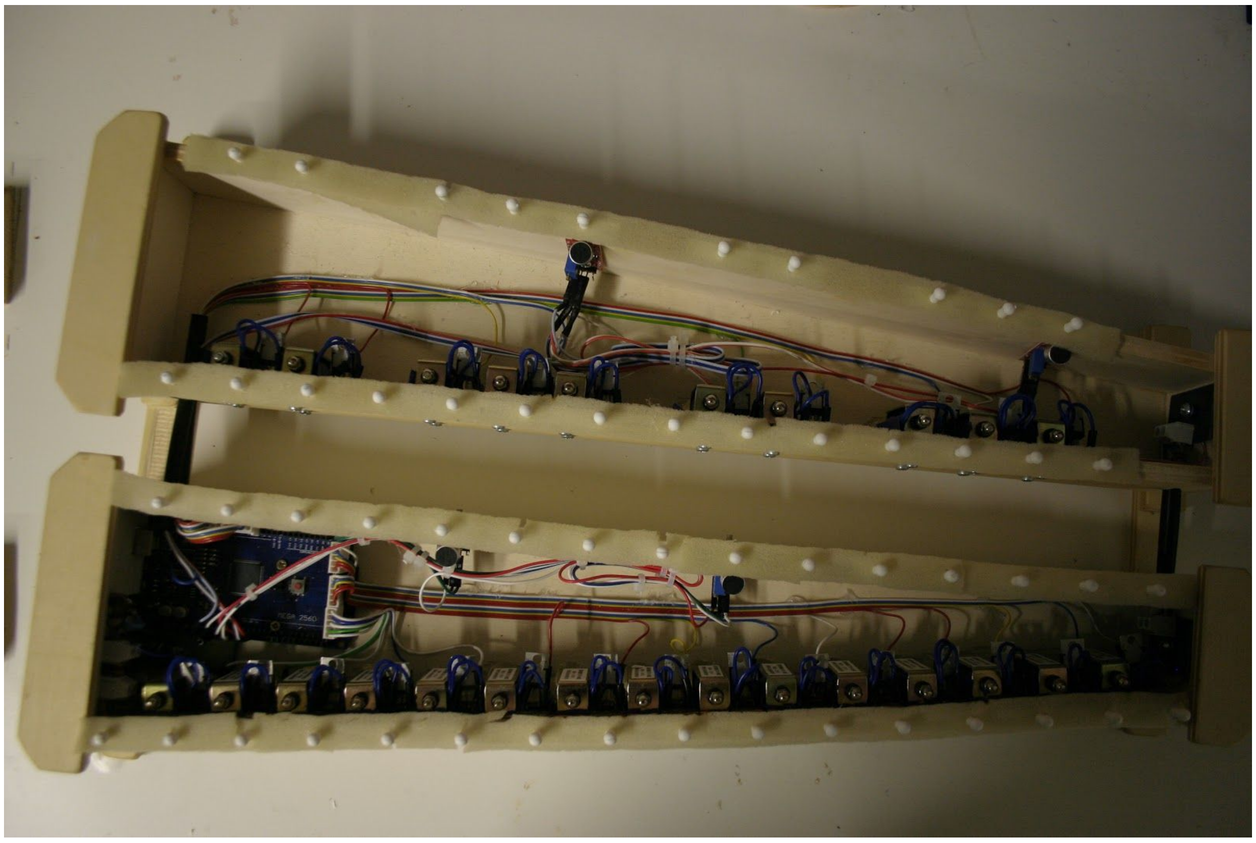

Figure 10 - The automated metallophone

7.6 Connecting the resonance chambers

The resonance chambers share the positive voltage terminal bus and the ground bus. These buses are connected via a wire. To consolidate the connection, a small PCB with terminals have been constructed.

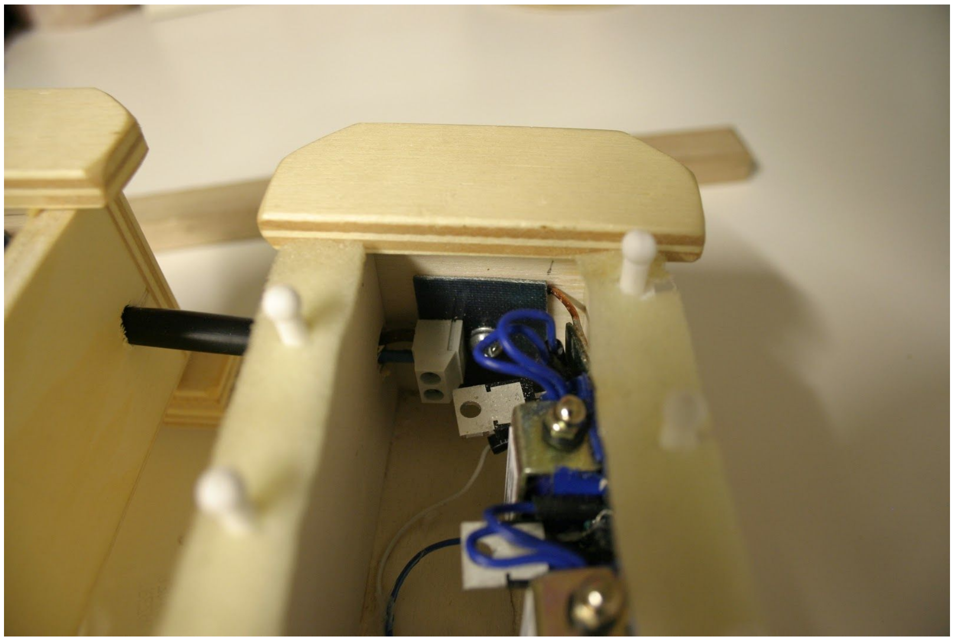

Figure 11 - An interconnection of the resonance chambers

7.7 Adjusting the actuators

To leave a space for adjustments, holes for the actuators were drilled with a one millimeter wider drill bit than necessary, enabling adjustments in four directions.

Also, the top of the solenoid core can be lifted up with a nut. This brings the core closer to the bar and decreases the necessary force to strike a bar.

The actuators can be further adjusted via the stabilizer. The ground of the stabilizer is lifted by diodes and it is possible the adjust the output voltage by increasing or decreasing the number of diodes. Greater values of source voltage cause increase of current flow, consequently increasing the force of the core of the solenoid.

8 Programming

The control element acquires data from the sensors and drives the actuators. To be able to perform this task, it is given instructions by a programming language. The control element of this device is a type of Arduino board, which are commonly programmed in Wiring. Wiring is a high level programming language based on C language with some additional functions.

Functions are blocks of code that provide an output value if given an input. Or set a parameter inside the control element. An example of a function providing output is an *analogue read*. This function has a syntax - *analogueRead(pin)*. The *pin* is an input variable, and is specified by the programmer. This function allows the programmer to read an analogue value from an analogue pin and convert it to a digital value on a scale of 2^10 values. An example of a function that provides no output is a pin mode function. A syntax *pinMode(pin, INPUT)* sets a pin to an input mode. Both of them are complex functions and the fact that they are accessible by one line of code clarifies the program.

A high level programming language means that the program is written in a human readable format and then interpreted into a machine code. It enables the programmer to focus on the functionalities of the program rather than the internal connections of the control element.

Programs written in Wiring consist of two functions. *Setup* function runs once and sets up e.g. the pins of the control elements or serial communication with a computer. *Loop* function runs continuously, and as a result the main part of the program is usually based here.

Also, variables needs to be set. There are two types of variables from the point of view of accessibility. First, there are global variables. Global variables are usually defined before the setup function and are shared everywhere in the program. Then there are local variables. Local variables are defined within a function and hold no value outside the function.

Variables are also defined by their datatype. The device uses integers and arrays. Integers are numbers in a range from -32 768 to 32 767. Arrays are sets of variables.

8.1 Programs for testing and adjustment

Testing programs assess the functionality of the circuits. Testing a prototype of a circuit can reveal possible failures, thus preventing a repetition of these failures.

The first program is a test of the prototype of the switching circuit. First of all, to prevent potential damage to the control element, the switching circuit was tested through with an adjustable power source with a current limiter. The switching circuit is controlled by voltage pulses. By placing a capacitor in series with a gate of a transistor, current was allowed to flow only until the capacitor had been charged. The pulse width derives from the capacitance. Since the circuit had been functional, the switching circuit was further tested using the control element. The testing program - *pulseWidthTuning.ino* - sends a pulse to the gate of the transistor of the switching circuit. The pulse width varies from ten to fifty milliseconds in step of two milliseconds to adjust the loudness of the bar and the actuator.

The second program - *barsTest.ino* - inspects the functionality of actuators and switching circuits. The code is similar to the *pulseWidthTuning.ino*, but instead of pulse width being the variable, the name of the bar is. The aim of this test was to find possible mistakes in assembling the switching circuits in the resonance chambers of the metallophone.

The third program - *testMicrophoneModules.ino* - was written to evaluate and adjust the potentiometers of the microphone modules. The program reads the value of the microphone module, converts it to one of the 2^10 values and prints it through the serial monitor on a computer screen. These values are lower if the potentiometer is set to higher resistance. Adjusting the potentiometer in a way that each microphone module provide similar outputs values simplifies later adjustments.

The fourth program - *testInteractivity.ino* - is a simplified version of the final program. The program drives an actuator of a defined bar when any of the microphones provide value exceeding a threshold constant. This program provides an environment for final adjustments of the potentiometers of the microphone modules and a threshold constant in the program. It demonstrates the influence of noise on the modules.

8.2 The final program

The final program can be divided into two sections. The first section consists of a *do .. while* cycle. The signal from the microphone modules is read continuously and if any of the modules provides a value higher than the threshold constant, it saves a number of the module to an array called *sequence*. If no microphone module provides a value that is higher than the threshold constant for a time exceeding the *sequenceTimer* variable, a zero is written to the *sequence* array. The *sequence* array is compared with a *sequenceTrigger* array at the end of each cycle. The cycle keeps repeating until the two arrays match.

When the program escapes the *do .. while* cycle, the second section of the program is active. In this part, the control element drives actuators defined in a *pitch* array. The delay between each *pitch* array variable is denoted by a *noteType* array. After driving the actuators, the numbers of the *sequence* array are changed to zeros, enabling the program to enter the *do .. while* cycle again.

10 Further development

The device is built with a specific purpose in mind. Throughout the construction process of this device, other possibilities of usage of an automated metallophone have been considered. Some of them are described in this section.

10.1 Bars of the metallophone functioning as the user interface

From the point of view of functionality, the measured bars act like buttons. This can be further developed by providing each bar a microphone module. Then, the bars could be used as an user interface. Providing a user interface introduces numerous possible enhancements.

Such as defining the active bars. One could set a function that would play a random note from a set of active bars. The user interface could have some added functionalities - e.g. play one bar twice for *enter* function.

Or as programming a library of melodies - a set of bars and a set of delays from the point of view of programming - and set a combinations of bars to trigger these melodies.

Or as a tool to find uncommon chords. A sketch of such program could be as follows - when a user plays any two bars, play two random bars. Then repeat the four notes.

Utilizing the bars of the metallophone as the user input would demand precision that the current device can not provide. To increase the precision of the microphone modules, there could be a tuning circuit for each module amplifying the signal on a frequency of the measured bar. The functionality of such circuit is similar to an audio equalizer.

10.2 LCD shield as user interface

The unused pins of the control element can be used for a traditional user interface e.g. LCD shield with buttons. The device could then be used as a tool to study and practice music theory. The LCD shield could display a set of scales and, if selected, play within the boundaries of the selected scale.

11 Conclusion

The purpose of this thesis was to describe the process of designing and constructing the automated metallophone. The thesis presented various options of the device design in the theoretical part of the thesis. The general background information were presented to the reader in the *basic overview* section. The device was described using block diagrams in the section *Technical overview from control perspective*. In this part, the importance of the functionality of the components was superior to the components themselves. The components and the circuits were the focus of the next section - *Technical overview from electrotechnical perspective.*

The practical part of this Bachelor's thesis depicted the construction process of the device. The general approach was illustrated in the section *The actuators.* Section *The stabilizer circuit* explained the need of a stabilizer, whose function had been described in the section *The stabilizer circuit* of the theoretical part of the thesis. The process of designing and constructing a printed circuit board was described in the section *The switching circuit*. The section *The Sensors* covered the assembly of the microphone modules. The aesthetics were covered in the sections *The connectors* and *Connecting the resonance chambers.*

The programming section of this Bachelor's thesis covered the programming background and various programs used for testing and adjustment. *The final program* section covered the functions used in the final program. The programs are included in the Appendix. The *Further development* section covered more possibilities of utilization of the automated metallophone.

To conclude, the design and construction of an automated metallophone was carefully evaluated, thus providing expected results. The description of the process would be more illustrative if a photo documentation of the process would be provided rather than photo documentation of details of the finished device.

Appendix

The appendix contains the programs aforementioned in the section *Programming*.

Pulse width tuning

```
//Pulse Width tuning


int barPin = 4;   //the bar that is tuned
int ledPin = 13;  //there is an inbuilt LED on pin 13, here it serves as an indicator of the pulse
width
int pulseWidth = 0; //Defines the variable pulseWidth. It will be provided a value later
void setup() {
  pinMode(barPin, OUTPUT); //defines the barPin and the ledPin as output pins
  pinMode(ledPin, OUTPUT);
}


void loop() {
   for(pulseWidth = 10; pulseWidth < 50; pulseWidth = pulseWidth + 2){ //pulseWidth in ms, it
goes //from 10 to 50 in a step of 2
  digitalWrite(barPin, HIGH);   //sends a logical one to the bar
  digitalWrite(ledPin, HIGH);   //and to the indicator
  delay(pulseWidth);            //the logical one is send for the time equal to pulseWidth
  digitalWrite(barPin, LOW); //then a logical zero is sent to the bar
  digitalWrite(ledPin, LOW);
  delay(1000);                  //one second delay to increase clarity
  }
  delay(5000);                  //after the for cycle is finished, the program waits for 5 seconds
and then starts from the beginning
}
```

Bars test

```
int pulseWidth = 20;



int C = 53;     //the pins are provided the names of the bar
int D = 51;     //for the purpose of clarity
int E = 49;
int F = 47;
int G = 45;

int A = 43;
int B = 41;
int C2 = 39;
int D2 = 37;
int E2 = 35;

int F2 = 31;
int G2 = 29;
int barA2 = 27;          //A2 is reserved for AnaloguePin2
int B2 = 25;
int C3 = 23;

int Cis = 7;
int Dis = 6;
int Fis = 5;
int Gis = 4;
int Ais = 3;
int Cis2 = 12;
int Dis2 = 11;
int Fis2 = 10;
int Gis2 = 9;
int Ais2 = 8;

void setup() {
  pinMode(C, OUTPUT);        //the bar pins are then defined as output pins
  pinMode(Cis, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(Dis, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(Fis, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(Gis, OUTPUT);
  pinMode(A, OUTPUT);
  pinMode(Ais, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C2, OUTPUT);
  pinMode(Cis2, OUTPUT);
  pinMode(D2, OUTPUT);
  pinMode(Dis2, OUTPUT);
  pinMode(E2, OUTPUT);
  pinMode(F2, OUTPUT);
  pinMode(Fis2, OUTPUT);
  pinMode(G2, OUTPUT);
  pinMode(Gis2, OUTPUT);
pinMode(barA2, OUTPUT);
pinMode(Ais2, OUTPUT);
pinMode(B2, OUTPUT);
pinMode(C3, OUTPUT);

digitalWrite(C, HIGH);        //and each bar is tested
delay(pulseWidth);
digitalWrite(C, LOW);
delay(500);



digitalWrite(Cis, HIGH);
delay(pulseWidth);
digitalWrite(Cis, LOW);
delay(500);



digitalWrite(D, HIGH);
delay(pulseWidth);
digitalWrite(D, LOW);
delay(500);



digitalWrite(Dis, HIGH);
delay(pulseWidth);
digitalWrite(Dis, LOW);
delay(500);
```

```
digitalWrite(E, HIGH);
delay(pulseWidth);
digitalWrite(E, LOW);
delay(500);


digitalWrite(F, HIGH);
delay(pulseWidth);
digitalWrite(F, LOW);
delay(500);

digitalWrite(Fis, HIGH);
delay(pulseWidth);
digitalWrite(Fis, LOW);
delay(500);


digitalWrite(G, HIGH);
delay(pulseWidth);
digitalWrite(G, LOW);
delay(500);


digitalWrite(Gis, HIGH);
delay(pulseWidth);
digitalWrite(Gis, LOW);
delay(500);
```

```
digitalWrite(A, HIGH);
delay(pulseWidth);
digitalWrite(A, LOW);
delay(500);


digitalWrite(Ais, HIGH);
delay(pulseWidth);
digitalWrite(Ais, LOW);
delay(500);


digitalWrite(B, HIGH);
delay(pulseWidth);
digitalWrite(B, LOW);
delay(500);


digitalWrite(C2, HIGH);
delay(pulseWidth);
digitalWrite(C2, LOW);
delay(500);


digitalWrite(Cis2, HIGH);
delay(pulseWidth);
digitalWrite(Cis2, LOW);
delay(500);
```

```
  digitalWrite(D2, HIGH);
  delay(pulseWidth);
  digitalWrite(D2, LOW);
  delay(500);


  digitalWrite(Dis2, HIGH);
  delay(pulseWidth);
  digitalWrite(Dis2, LOW);
  delay(500);


  digitalWrite(E2, HIGH);
  delay(pulseWidth);
  digitalWrite(E2, LOW);
  delay(500);


  digitalWrite(F2, HIGH);
  delay(pulseWidth);
  digitalWrite(F2, LOW);
  delay(500);


   digitalWrite(Fis2, HIGH); //the coil on Fis2 has 12Ohms instead of 6Ohm as the rest of the
coils
  delay(30);                 //by setting higher pulse width to the bar, the solenoid is able
  digitalWrite(Fis2, LOW);   //to hit the bar
```

```
delay(500);


digitalWrite(G2, HIGH);
delay(pulseWidth);
digitalWrite(G2, LOW);
delay(500);


digitalWrite(Gis2, HIGH);
delay(pulseWidth);
digitalWrite(Gis2, LOW);
delay(500);


digitalWrite(barA2, HIGH);
delay(pulseWidth);
digitalWrite(barA2, LOW);
delay(500);


digitalWrite(Ais2, HIGH);
delay(pulseWidth);
digitalWrite(Ais2, LOW);
delay(500);


digitalWrite(B2, HIGH);
delay(pulseWidth);
```

```
  digitalWrite(B2, LOW);
  delay(500);


  digitalWrite(C3, HIGH);
  delay(pulseWidth);
  digitalWrite(C3, LOW);
  delay(500);

}
void loop() {               //loop is empty, because it is sufficient to run the test once
}
```

Microphone modules test

```
int mic1 = A0; //defines the pins that the microphones modules are connected to
int mic2 = A1;
int mic3 = A2;
int mic4 = A3;

int mic1Value = 0; //defines a variable for the microphone modules value
int mic2Value = 0;
int mic3Value = 0;
int mic4Value = 0;

void setup() {
  pinMode(mic1, INPUT); //defines the microphone modules as input pins
  pinMode(mic2, INPUT);
  pinMode(mic3, INPUT);
  pinMode(mic4, INPUT);
```

```
 Serial.begin(9600); // starts a communication via serial monitor (9600 baud rate)
}

void loop() {
mic1Value = analogRead(mic1); //reads a value from the pin and save it to the mic1Value
variable
mic2Value = analogRead(mic2);
mic3Value = analogRead(mic3);
mic4Value = analogRead(mic4);

Serial.println("____________________________________");
Serial.println("Microphone on A0");
Serial.println(mic1Value);                  //prints the variable to the serial monitor
Serial.println("Microphone on A1");
Serial.println(mic2Value);
Serial.println("Microphone on A2");
Serial.println(mic3Value);
Serial.println("Microphone on A3");
Serial.println(mic4Value);
Serial.println("____________________________________");
delay(500);                          //150 ms delay to prevent flooding the serial monitor
}
```

Interactivity test

```
int mic1 = A0; //defines the pins that the microphones modules are connected to
int mic2 = A1;
int mic3 = A2;
int mic4 = A3;
int mic1Value = 0; //defines a variable for the microphone modules value
int mic2Value = 0;
int mic3Value = 0;
int mic4Value = 0;

int struckLimit = 800; //value from 0 to 1023 that determines whether a bar was strucked or not
int pulseWidth = 25;

int C = 53;      //the pins are named after the bars
int D = 51;      //for the purpose of clarity
int E = 49;
int F = 47;
int G = 45;

int A = 43;
int B = 41;
int C2 = 39;
int D2 = 37;
int E2 = 35;

int F2 = 31;
int G2 = 29;
int barA2 = 27;          //A2 is reserved for AnaloguePin2
int B2 = 25;
int C3 = 23;

int Cis = 7;
int Dis = 6;
int Fis = 5;
int Gis = 4;
int Ais = 3;

int Cis2 = 12;
int Dis2 = 11;
int Fis2 = 10;
int Gis2 = 9;
int Ais2 = 8;

void setup() {
  pinMode(mic1, INPUT); //defines the microphone modules as input pins
  pinMode(mic2, INPUT);
  pinMode(mic3, INPUT);
  pinMode(mic4, INPUT);

  pinMode(C, OUTPUT);       //the bar pins are then defined as output pins
  pinMode(Cis, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(Dis, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(Fis, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(Gis, OUTPUT);
  pinMode(A, OUTPUT);
  pinMode(Ais, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C2, OUTPUT);
  pinMode(Cis2, OUTPUT);
  pinMode(D2, OUTPUT);
  pinMode(Dis2, OUTPUT);
  pinMode(E2, OUTPUT);
  pinMode(F2, OUTPUT);
  pinMode(Fis2, OUTPUT);
  pinMode(G2, OUTPUT);
  pinMode(Gis2, OUTPUT);
  pinMode(barA2, OUTPUT);
  pinMode(Ais2, OUTPUT);
  pinMode(B2, OUTPUT);
  pinMode(C3, OUTPUT);
}

void loop() {
delay(10);

mic1Value = analogRead(mic1); //reads a value from the pin and save it to the mic1Value
variable
mic2Value = analogRead(mic2);
mic3Value = analogRead(mic3);
mic4Value = analogRead(mic4);

if(mic1Value > struckLimit){ //if the mic1Value is lower than the struckLimit it is thought of as
noise
  delay(500);
  digitalWrite(G2, HIGH); //mic1 is on G1
  delay(pulseWidth);
  digitalWrite(G2, LOW);
  delay(100);
}
else if(mic2Value > struckLimit){
  delay(500);
  digitalWrite(G2, HIGH); //mic2 is on D2
  delay(pulseWidth);
  digitalWrite(G2, LOW);
  delay(100);
}
else if(mic3Value > struckLimit){
  delay(500);
  digitalWrite(G2, HIGH);
  delay(pulseWidth);
  digitalWrite(G2, LOW);
  delay(100);
}
else if(mic4Value > struckLimit){
  delay(500);
  digitalWrite(G2, HIGH);
  delay(pulseWidth);
  digitalWrite(G2, LOW);
  delay(100);
}
}
```

Final program

```
//____________________________________________
//Defining microphone values, pins

int mic1 = A0;
int mic2 = A1;
int mic3 = A2;
int mic4 = A3;

int mic1Value = 0;
int mic2Value = 0;
int mic3Value = 0;
int mic4Value = 0;
//____________________________________________

//____________________________________________
//Defining the pins of the bars
int C = 53;
int D = 51;
int E = 49;
int F = 47;
int G = 45;

int A = 43;
int B = 41;
int C2 = 39;
int D2 = 37;
int E2 = 35;

int F2 = 31;
int G2 = 29;
int barA2 = 27;          //A2 is reserved for AnaloguePin2
int B2 = 25;
int C3 = 23;

int Cis = 7;
int Dis = 6;
int Fis = 5;
int Gis = 4;
int Ais = 3;

int Cis2 = 12;
int Dis2 = 11;
int Fis2 = 10;
int Gis2 = 9;
int Ais2 = 8;
//____________________________________________

//____________________________________________
//Defining the pulseWidth and the threshold
int pulseWidth = 22;
int threshold = 750;
//____________________________________________

//____________________________________________
//Sequence variables
int sequence[4] = {0, 0, 0, 0};
int sequenceTrigger[4] = {1, 2, 3, 4};
int sequenceTimer = 1000;
int sequencePosition = 0;
unsigned long sequenceTime = 0;
int miniWait = 350;
//________________________________________________
//________________________________________________
//Notes sequence
int note = 500;
int halfnote = note / 2;
int quaternote = note / 4;
int pitch[7] = {D, D, Dis, D, D, G2, Ais2};
int noteType[7] = {note, halfnote, note, note, halfnote, note, note};
//________________________________________________

void setup() {
  Serial.begin(9600);
  pinMode(mic1, INPUT); //defines the microphone modules as input pins
  pinMode(mic2, INPUT);
  pinMode(mic3, INPUT);
  pinMode(mic4, INPUT);

  pinMode(C, OUTPUT);        //the bar pins are then defined as output pins
  pinMode(Cis, OUTPUT);
  pinMode(D, OUTPUT);
  pinMode(Dis, OUTPUT);
  pinMode(E, OUTPUT);
  pinMode(F, OUTPUT);
  pinMode(Fis, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(Gis, OUTPUT);
  pinMode(A, OUTPUT);
  pinMode(Ais, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C2, OUTPUT);
  pinMode(Cis2, OUTPUT);
  pinMode(D2, OUTPUT);
  pinMode(Dis2, OUTPUT);
  pinMode(E2, OUTPUT);
  pinMode(F2, OUTPUT);
  pinMode(Fis2, OUTPUT);
  pinMode(G2, OUTPUT);
  pinMode(Gis2, OUTPUT);
  pinMode(barA2, OUTPUT);
  pinMode(Ais2, OUTPUT);
  pinMode(B2, OUTPUT);
  pinMode(C3, OUTPUT);
}

void loop() {
do{
  mic1Value = analogRead(mic1); //read a 2^10 value from mic1 pin and save it to mic1value
  mic2Value = analogRead(mic2);
  mic3Value = analogRead(mic3);
  mic4Value = analogRead(mic4);

  if (mic1Value >= threshold){        //if the read value exceeds the threshold constant
        for (int a = 0; a < 3; a++){
        sequence[a] = sequence[a + 1];      //move the values of the sequence array by one to
//the left
```

```
        }                         //a[2] = a[3], a[3] = a[4]...
        sequence[3] = 1;          //write 1 on the last position
        sequenceTime = millis();          //and write the time of the strike to the
sequenceTime //variable
        for (int a = 0; a < 4; a++){
        Serial.print(sequence[a]);    //also, send the sequence array through the serial monitor
        }
        Serial.println();         //add an empty line to keep the serial monitor legible
        delay(miniwait);          //add a small delay after strike to not assess one strike
twice
  }

  if (mic2Value >= threshold){
        for (int a = 0; a < 3; a++){
        sequence[a] = sequence[a + 1];
        }
        sequence[3] = 2;
        sequenceTime = millis();
        for (int a = 0; a < 4; a++){
        Serial.print(sequence[a]);
        }
        Serial.println();
        delay(miniWait);
  }

  if (mic3Value >= threshold){
        for (int a = 0; a < 3; a++){
        sequence[a] = sequence[a + 1];
        }
        sequence[3] = 3;
        sequenceTime = millis();
        for (int a = 0; a < 4; a++){
        Serial.print(sequence[a]);
        }
        Serial.println();
        delay(miniWait);
  }

  if (mic4Value >= threshold){
        for (int a = 0; a < 3; a++){
        sequence[a] = sequence[a + 1];
        }
        sequence[3] = 4;
        sequenceTime = millis();
        for (int a = 0; a < 4; a++){
        Serial.print(sequence[a]);
        }
        Serial.println();
        delay(miniWait);
  }

  if (millis() - sequenceTime == sequenceTimer){              //if nothing was played for a time
that equals //sequenceTimer
        for (int a = 0; a < 3; a++){
        sequence[a] = sequence[a + 1];             //move the sequence by 1 to left
        }
        sequence[3] = 0;                           //and send 0
        sequenceTime = millis();                   //rewrite time variable
```

```
 }
   }while((sequence[0]  !=  sequenceTrigger[0])  ||  (sequence[1]  !=  sequenceTrigger[1])  ||
(sequence[2] != sequenceTrigger[2]) || (sequence[3] != sequenceTrigger[3]));
//if the sequence equals the sequenceTrigger, break the loop

 for (int a = 0; a < 7; a++){
        digitalWrite(pitch[a], HIGH); //play the notes from the pitch array
        delay(pulseWidth);
        digitalWrite(pitch[a], LOW);
        delay(noteType[a]);             //the type of the note - quarter note, half note
 }

 for(int a = 0; a < 3; a++){
        sequence[a] = 0;                //write  zeros  to  the  played  sequence  array  to  prevent
//skipping the loop
 }
}
```

References

[1] *Robotic xylophone* [online]. [cit. 2018-05-23]. Available at: http://www.nerdkits.com/videos/robotic_xylophone/

[2] *Automated MIDI Xylophone* [online]. [cit. 2018-05-23]. Available at: http://www.instructables.com/id/Automated-MIDI-Xylophone/

[3] ROBINS, Brandon, Neil TIWARI and Jenny YONG. *Automated Xylophone* [online]. [cit. 2018-05-23]. Available at: http://hades.mech.northwestern.edu/index.php/Automated_Xylophone

[4] *2 A positive voltage regulator IC* [online]. [cit. 2018-05-25]. Available at: http://www.st.com/content/ccc/resource/technical/document/datasheet/e9/be/53/a3/1f/6f/4f/75/CD00000449.pdf/files/CD00000449.pdf/jcr:content/translations/en.CD00000449.pdf