Design and Construction
Of A Paper Thin
Magnetostatic Loudspeaker
Using Arduino Mega 2560,
MusicShield v1.13 and MPR121

Minh Huyen Do Thi
Abstract

This focus of this report is on design and the aim is to build a magnetostatic speaker and a music loudspeaker from scratch by assembling the sensors together and thus, making a simple but very thin foil loudspeaker. The Arduino sensor is programmed in C language in order to make the capacitive sensor to have the ability to control the functions of the music player and LEDs.

At the end of the process, the product will be able to play music through the thin loudspeaker by touching the capacitive sensor.

Keywords: C, Arduino MEGA 2560, Foilspeaker, magnet, MusicShield, MPR121, amplifier.
Acknowledgements

Firstly, I want to thank my supervisor, Dr. Henrik Andersson, who has supported me and my co-worker with his patience, suggestions and knowledge about the sensors. I am grateful to Dr. Henrik Andersson for providing me with the opportunity to learn about Arduino sensors, to acquire knowledge about the workings of the speaker and for providing ideas in relation to the future design of a loudspeaker.

The other member of the thesis group, Chiang Tengyi, is an excellent team member and some of the hardware parts of the thesis work, the testing of the devices, analyzing the sound spectrum and some ideas in sensor connections were performed in co-operation with her.

I would also like to thank a researcher, who designed the thin foil loudspeaker thus providing us with the opportunity to conduct this research and thanks must also be given to Emmanuel Nsambu for his tutorials in software programming and to Xu Ye who gave us tutorials in relation to the hardware.
## Table of Contents

Abstract ........................................................................................................................................ ii
Acknowledgements .................................................................................................................. iii
Table of Contents ...................................................................................................................... iv
Terminology .................................................................................................................................. vi
Abbreviations ............................................................................................................................ vi

### 1 Introduction ........................................................................................................................ 1
1.1 Background and problem motivation ................................................................................. 1
1.2 Problem statement ............................................................................................................... 1
1.3 Scope ................................................................................................................................... 2
1.4 Outline .................................................................................................................................. 2
1.5 Contribution ......................................................................................................................... 2

### 2 Background Information .................................................................................................. 4
2.1 Arduino Mega 2560 ............................................................................................................. 4
2.2 Music shield v1.13 ............................................................................................................... 4
2.3 MPR121 capacitive touch chip ............................................................................................ 4
2.4 Audio amplifier .................................................................................................................... 4
2.5 Loudspeaker ...................................................................................................................... 5
2.6 Block Magnet ...................................................................................................................... 5
2.7 ............................................................................................................................................... 5
2.8 SD card ................................................................................................................................ 5
2.9 Arduino Software Platform ................................................................................................. 5

### 3 Theory/ Related work .......................................................................................................... 7
3.1 Spectrum .............................................................................................................................. 7
3.2 Type of loudspeaker .......................................................................................................... 7
3.2.1 Magnetostatic speaker ................................................................................................. 7
3.2.2 Thin foil speaker ........................................................................................................... 8
3.2.3 ITRI FleXpeaker ......................................................................................................... 8
3.2.4 Film speaker ................................................................................................................ 9

### 4 Methodology ...................................................................................................................... 10

### 5 Implementation ................................................................................................................ 11
5.1 Model ................................................................................................................................... 11
Table of Contents

5.2 The design of the speaker ................................................. 12
5.3 Configuration of the electrode pin MPR121 ........................... 16
5.4 Configuration for Amplifier ................................................. 17
5.5 Configuration for high voltage/current driver and LEDs ...... 17
5.6 Arduino Mega 2560 Programming in C .............................. 18
  5.6.1 General declaration and setup ...................................... 18
  5.6.2 ReadTouchInputs() Method ...................................... 20
  5.6.3 Play() method .......................................................... 23
  5.6.4 availableProcessorTime() Method ............................... 24

6 Result ................................................................................ 25
  6.1 The demo ................................................................. 25
  6.2 Result in simulation ...................................................... 26

7 Conclusion ......................................................................... 29

8 Further works ...................................................................... 30

Reference .............................................................................. 31

Appendix A: mpr121Edited_worked.ino .................................. 32
Appendix B: Components List ................................................. 40
Appendix C: ............................................................................. 41
## Terminology

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM</td>
<td>Pulse-width modulation</td>
</tr>
<tr>
<td>UART</td>
<td>Universal asynchronous receiver/transmitter</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>MHz</td>
<td>MegaHertz</td>
</tr>
<tr>
<td>ICSP</td>
<td>In-circuit serial programming</td>
</tr>
<tr>
<td>AC</td>
<td>Alternating current</td>
</tr>
<tr>
<td>DC</td>
<td>Direct current</td>
</tr>
<tr>
<td>GCC</td>
<td>GNU Compiler Collection</td>
</tr>
<tr>
<td>IRQ</td>
<td>Active Low Open-drain Interrupt Output</td>
</tr>
<tr>
<td>SDA</td>
<td>Serial data</td>
</tr>
<tr>
<td>SCL</td>
<td>Serial clock</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>ADD</td>
<td>Slave Address Pin Selects</td>
</tr>
<tr>
<td>MP3</td>
<td>MPEG Audio Layer III</td>
</tr>
<tr>
<td>PVDF</td>
<td>Polyvinylidene fluoride</td>
</tr>
</tbody>
</table>
# 1 Introduction

## 1.1 Background and problem motivation

Based on rapid developments in technology, items such as phones, laptops, TV, etc. have become thinner and have more advanced functions. Engineers now have ambitions to develop more advanced products, one of which is the magneto static speaker.

In the 19th century, the speaker was invented. At the beginning it was merely an electric loud speaker which was installed in a telephone, however after many improvements by several inventers, a fully functional speaker using electromagnet was created and produced as a commercial product. Two centuries later, the loud speaker appears as a part of almost all daily life in electrical products, such as: TVs, laptops, docks, phones, stereo, etc. With advances in technology, engineers are now planning on making all electronics items both portable and thinner, which will make them more convenient. A thin and advanced magneto static speaker can be placed on a wall with a low input voltage.

This report focuses on the design for a magneto static speaker, built by using a thin foil speaker, which is a flexible membrane and which has a voice coil printed on it and which contains a number of magnets. The sound from the speaker depends on vibrations on the membrane, which are caused when the magnetic field, coming from the magnets, interacts with the current flow through the coil. In addition to this, in order to fully test the quality of the speaker, a music system is built. The system consists of an Arduino Mega 2560 sensor, an mp3 player Music Shield, a capacitive touch sensor MPR121, a high voltage/current driver, an audio amplifier, some LEDs and the speaker constructed for this project. The Arduino sensor controls the mp3 player in order to perform the basic functions of a music player and the capacitive touch sensor is also controlled by the Arduino sensor to assist the users to use the basic functions from the music player.

## 1.2 Problem statement

The thesis work aims to design a thin magneto static speaker. The foil speaker should be connected to the magnets which should be arranged in such a way that they can create a magneto static speaker. Consideration must also be given to the distance between the speaker itself and the magnets. The application is based on a connection between the sensors, the Arduino Mega 2560 with the Music Shield v1.13 and a capacitive touch sensor MPR121, together with an 18W amplifier, a relay, 3 LEDs, foil speaker and magnets.
The Arduino should be programmed to connect and control the Music Shield and capacitive touch sensor MPR121 together by using C programming. Moreover, the MPR121 should be able to control the Music Shield by touching pins with the basic functions: Play, Pause, Volume down, Volume up, Previous song and Next song; an extra function is included to control the LEDs. A relay should be used to provide power to the LEDs and the amplifier should be used to control the thin foil speaker.

Finally, the thin music speaker should be tested and analyzed by means of an oscilloscope and the program Audacity.

1.3 Scope
This report focuses on explaining how the music speaker is designed and assembled and on the arrangement of the magnets and the foil speaker. In addition information is provided in relation to how the Arduino was programmed, the results for simulations of the speaker and some explanation concerning the sound quality.

1.4 Outline
In Chapter 1, the main idea of the whole project is introduced and the problems which are to be solved during the research. Chapter 2 provides basic information about the components and the programming language used for the project. Chapter 3 explains the theory part regarding the concepts required for the design of the speaker, the type of speaker used in the project and some other types. Chapter 4 is concerned with the methodology of the project. Chapter 5 deals with the implementation of the project: the model of the whole project; the design of the speaker; the arrangement of the magnets; the configuration of the amplifier, the capacitive touch sensor, the high voltage/current driver and some explanations of the code used for the Arduino. Chapter 6 provides the results of the project. Chapter 7 offers the conclusion and states what has been achieved. Chapter 8 concerns future work and how to improve on this project in order to achieve a better result. All the sources used during this research are listed after chapter 8 in the References as is a list of components used. An appendix appears at the end of the report.

1.5 Contribution
The thesis has been completed by two students, Minh Huyen Do Thi and Chiang Tengyi, each of whom contributed different parts of the project with some parts being dealt with by both members of the group.

Minh Huyen Do Thi contributed the work associated with the speaker, some hardware parts and the programming part of the project.
Chiang Tengyi contributed the work associated with the speaker, some hardware parts and the measurement of the sound spectrum for the final product.
2 Background Information

2.1 Arduino Mega 2560
Arduino Mega 2560 is an updated version of Arduino Mega, also known as a microcontroller board that is used as the basis of the AVR Atmega2560. This is the reason why the number 2560 appeared in the name of the Arduino. The board has a total of 70 pins, 54 of which are digital input/output pins, 14 can be used as PWM output and the remaining 16 pins are for analog inputs. It also consists of 4 UARTS which are hardware serial ports, a 16MHz crystal oscillator, a USB connector, a power jack a reset button and an ICSP header. It can be powered up by means of an AC-to-DC adapter or by merely connecting it to a computer by means of a USB cable to power up or upload the program to the microcontroller. [1]

2.2 Music shield v1.13
Music shield is a Seeed product that can play music from a MicroSD card and it is a dock shield that is compatible with an Ipod/Iphone. It has 2 control-push buttons, 1 knob switch, an Sdcard reader, a line-in and a line-out. It is designed to work with any type of Atmega168 and ATmega328 based board, for example: Arduino board, Seeeduino v328...Music shield is based on a VS1053b chip that can play multi-format MP3 media files. [2]

2.3 MPR121 capacitive touch chip
An MPR121 capacitive touch chip is a board with 18 pins which are 3.3V, IRQ, SCL, SDA, ADD, GND and 12 electrode pins, of which 8 can be LED driving pins for occasions when the electrodes are not configured; 2 capacitors, 4 resisters and the MPR121 sensor. At the bottom of the board there are 4 jumpers, which are set to off as the default option and which are connected to ADD, SDA, SCL and IRQ.

MPR121 is also known as the second generation of capacitive touch sensor controllers, which follow a series of the MPR03x sensor controllers. It is a sensitive device that uses the I²C bus configuration in its hardware. It has an internal intelligent, increased electrode count and an expanded filtering system with debounce. It has completely independent electrodes with auto-configuration built in. [3]

2.4 Audio amplifier
An audio amplifier is an amplifier that increases the low power audio signals to the point at which the loudspeaker can be driven. In this specific project, a class AB audio power amplifier is used. It converts a low voltage
supply to a high power output by using the bridge tied load method and 55dB of high gain. [4]

2.5 **Loudspeaker**

A loudspeaker, or so called speaker, is an electro acoustic transducer that is used to create sound that responds to an electrical audio signal input. For ease of understanding, a loud speaker is a device that converts electrical energy from a power supply into sound that is amplified so that people can hear the sound from far away. It is commonly formed from a voice coil electromagnet, which is acting on a permanent magnet, supported by a paper cone. [5]

2.6 **Block Magnet**

Magnet is derived from a Greek word, “Magnesian stone”, which is a type of material or object that creates a magnetic field. There are many kinds of magnets but most common magnets are made of metals such as iron or metals that mix with other materials such as ceramics, rubber, etc. The block magnet, used in this project, is SALE-031 Neodymium Magnet (9.8x3.8x2.8mm) with a weight of 0.79gram. It is able to carry weights up to approximately 940grams and the maximum working temperature is 80°C. [6]

2.7 **C**

The C programming language is the language which assists users to create instructions for a computer to follow. Since it is a compiled language, it must be run on a compiler, so that an executable form can be created. A program using C programming consumes very little memory. All the executable code in C is surrounded by functions (subroutines). It is widely used, not merely for implementing system software, but also to create software applications. An extension of C, which is C++, is also a well known programming language but is not as popular as C. A C file has the extension .c and the header file has the extension .h. [7]

2.8 **SD card**

An SD card, also known as a Secure Digital card, is an integrated circuit stored in a compact and which is covered by rugged plastics. It is a non-volatile memory card which means that it can store data even when there is no power. An SD card is used in portable devices such as digital cameras, phones, music players, etc. There are 3 different sizes: original, mini and micro. It has high data transfer rate and is lightweight varying from 0.5 to 2.0 grams. [8]

2.9 **Arduino Software Platform**

The Arduino Software Platform is the environment used to write code for the Arduino microcontroller and to upload it to the i/o board. Even though
the environment is written in Java and is based on processing, avr-gcc, and other open source software, in general, the environment works with either the C or C++ programming languages. [9]
3 Theory/ Related work

3.1 Spectrum
The spectrum made from a recorded sound show the relative volume as a function of frequency. The frequency spectrum of a time-domain signal presents that signal in a frequency domain. It can be generated by using a Fourier transform of the signal and the result is shown as amplitude versus frequency and phase versus frequency. A sample of sound can contain a mixture of more than one frequency. Noise can be found in the environment and it has many different frequencies. [10]

3.2 Type of loudspeaker
In this project, the magnetostatic speaker is the type of speaker that is used. Additionally, in this report, three other types of thin loud speakers will be discussed: FleXpeaker, Piezoelectric and thin foil speaker.

3.2.1 Magnetostatic speaker

![Magnetostatic speaker](image)

**Figure 1. Magnetostatic speaker (from [11])**

The magnetostatic speaker is the type of speaker used in this project work. It is also known as a planar magnetic speaker, which, in some ways, described a ribbon speaker (a type of speaker made of thin metal-film ribbon which is suspended in a magnetic field) but this is not technically true. People call it a planar magnetic speaker because it is designed in a rectangular surface form that radiates on both sides. This type of speaker con-
sists of a flexible membrane which has a voice coil mounted or printed on it. The magnetic field comes from the magnets that are placed on either side of the diaphragm and which interact with the current flows through the coil, creating the vibration on the membrane without significant bending or wrinkling. The vibration is somewhat uniform. [11]

3.2.2 Thin foil speaker

A thin foil speaker is made by Mid Sweden University and is also known as a type of planar magnetostatic speaker. It consists of aluminum foil tape that sticks on a piece of plastic. The dimensions of the speaker are 30x40cm. It has a resistance from 2 to 3 Ohms. The material used in making this speaker is cheap and easy to find and thus the price could potentially be low, thus enabling it to be used in low cost applications or similar. A simple speaker can be constructed from this in combination with at least 50 thin and small magnets placed under it.

3.2.3 ITRI FleXpeaker

FleXpeaker is a product of the Research and Development Team of Industrial Technology Research Institute (ITRI), a nonprofit organization that engages in applied research and technical services. They arrived at the idea to make this speaker after conducting many experiments in order to choose the material which could form a thin and flexible speaker. Their idea was to use metals that are electrically conductive with soft electric substrate plastics and these were combined with a cheap and always available substance, namely, paper.
The top and bottom layer of the speaker are made of 2 pieces of paper (which is less than 1mm thick) which cover a thin layer of metallic membranes to allow electrode currents. In the middle of the speaker, there are membrane-covered sheets, which act as vibrating diaphragms with an electrostatic charge. Digital signals produce a soundscape, which attracts or repels the vibrating diaphragm and results in the compression of air to produce sound and this is how the FleXpeaker works. It reduces the size of a normal speaker structure and the other significant advantage of FleXpeaker is that the power to drive the speaker is less than 10% of a normal speaker.

The frequency of this product is between 200Hz and 20 KHz, which is very suitable for creating a voice range from middle to high frequencies, for example: bird’s chirpings. [12]

3.2.4 Film speaker

The film speaker is a product from the company FILS Film Speaker in Korea. It is made of PVDF film (which is a type of piezo film) which is both thin and flexible.

![Film Speaker Diagram](image)

**Figure 3. How film speaker is created and works (From [13])**

The figure above shows how a film speaker is constructed. The speaker has a wide frequency range, low impedance, low density and is very sensitive. The PVDF film is in between 2 layers of thin electrodes. It produce sound by means of vibration, which is generated by applying an electrical signal to the PVDF material; or, in simple terms, the explanation is that the speaker produces sound from the thin surface of the PVDF film. [13]
4 Methodology

The application will start with the creation of the speaker. The magnets will be arranged in such a way as to make the speaker work and the distance between the speaker and the magnets will also be discussed. The application will be using the Arduino Mega 2560 as the master device in order to configure and control the Music Shield together with the MPR121. The application will use the C programming language so as to configure the MPR121 in order to control the Music Shield with basic functions: Play, Pause, Volume down, Volume up, Previous song and Next song; an extra function is added in order to control the LEDs. A 12V power source will be used for 3 parts of the application: to provide power to the Arduino Mega 3560, the high voltage/current driver and the 18W amplifier. The relay will be used to provide power to the LEDs and the amplifier will be used for controlling the speaker. A capacitor will be added to the amplifier to make it fully functional. Finally, the software Audacity will be used to analyze the sound spectrum and an oscilloscope will be used to analyze the waveform data.
5 Implementation

5.1 Model

The model of the project can be described by the diagram below:

Diagram1. The concept of the project [14]

Firstly, the Music shield v1.13 and MPR121 sensor are connected to the Arduino Mega 2560 as shown in (a) and (b) in the diagram. In order to use the electrode pin of the MPR121, 6 pins are connected to the sensor as shown in (c). To control the Music shield and the capacitive sensor, a program from Arduino platform is transferred to the main sensor (d). A power supply is used to provide power to the Arduino sensor, the amplifier and the relay (e) (f) (g). The amplifier has the function to increase the signal, which in this particular project, is sound. The audio input of the amplifier is from the Music Shield (h) and the output is connected to the speaker (j) and under the speaker there are magnets (j) in order to create vibration by interacting with the current flow through the foil speaker which results in sound. As for the high voltage/current driver that is connected to the power supply in (g), a few LEDs are connected to the high voltage/current driver (k) in order to light up or blink according to the program in Arduino, because one pin in the main sensor is connected to the high voltage/current driver (l).
5.2 The design of the speaker

The magnetostatic speaker is a very important part of the project. It is based on a thin foil speaker with an amount of magnet. As for the speaker, the dimensions are 300x400mm and for the magnet, the dimensions are 9.8x3.8x2.8mm.

It must be designed in such a way that it can make the speaker vibrate and thus interact with the current flow through the foil speaker, which results in the creation of sound. In theory, the conductor between the magnets has same direction as the current flow. By using the right hand rule in physics, the force on the membrane is determined. When there is a change in the polarity, the force will be shifted 180°, which causes the membrane to move in and out. [16]

Following the theory principal, a design was drawn based on the design of the foil speaker and the magnet given in this project:

![Diagram of project's theory design](image)

**Figure 4. Project’s theory design**

As the figure above shows, the blue rectangular is the magnet that is placed at the South Pole and the red rectangular is the magnet that is placed at the North Pole. The red arrow shows the current flow through the speaker.
The connection is in green and the source represents the foil coil and the design of the foil speaker.

In this project, there are about 70 magnets used for the speaker. In order to spread them equally under the speaker, the dimensions of which are 40x30cm, 3 South Pole magnets are placed in a line and next to them are 2 North Pole magnets in a line. One reason for arranging the magnets in this manner is to prevent attraction between the south and north poles of the magnets. If the same number of magnets are in line with different poles placed next to each other, then, as the distance is short and the magnets are strong, the north and south poles will be attracted to each other which will cause the speaker to have magnets, which are stuck to each other (see figure below). In addition, the material used for sticking the magnets on the cardboard is not sufficiently strong, thus it is not a good idea to place the same number of magnets on all the lines. Even if a stronger tape is used to stick the magnets, the magnets will not create a strong magnetic field because part of the magnetic field is shared by the different poles attracting each other.

Figure 5. Same amount of magnets are placed in the lines
A better idea would be to have more magnets in order to create a stronger vibration on the speaker, but, in this special type of speaker, it is possible to have the same number magnets in line but, they should not be placed at the same position on each line. The reason is that, the coil line size is 4mm and the space between the coil is very small (0.2mm), while the size of the block magnet is 9.8x3.8x2.8mm; the magnet, instead of creating a magnetic field for the project, will merely attract the magnet next to it and the result is the magnetic field created by the magnets will not be strong.

By connecting the output from the amplifier to the speaker and placing a thin foil speaker, which has voice coil printed on it, above the magnets, the current will flow through the coil (presented as green connection in the figure above) and will interact with the magnetic field from the magnets placed on either the North or South Pole, thus creating the vibration on the speaker without significant bending or wrinkling. The vibrations on the speaker create the sound.

And this is the result in practical terms, namely to place the magnets according to the theory:

Figure 6. Practical result
The cover for this board, which has the magnet, is the foil speaker from figure 6.

Figure 7. Connecting amp to speaker

There is a small space between the speaker and the magnet around 0.5cm, which is shown in the figure below:

Figure 8. Design of the speaker

The reason why there is a small gap between them is to prevent the speaker from vibrating too much, which can create a disturbance.
5.3 Configuration of the electrode pin MPR121

As explained earlier, there are 6 electrode pins that are used in this project. The functions are listed below:

- Pin 0 (red cable): Start
- Pin 1 (Orange cable): Pause/Play
- Pin 2 (Dark green cable): Reduce volume
- Pin 3 (White cable): Increase volume
- Pin 4 (Yellow cable): Previous song
- Pin 5 (Light green cable): Next song

5.4 Configuration for Amplifier

![Amplifier and Music Shield](image)

**Figure 11. Amplifier and Music Shield (From [14])**

To configure the amplifier, the Audio input on the left is connected to the “Earphone” plug (marked in red). The speaker in the middle of the amplifier is connected to the loudspeaker and finally the supply on the right is connected to the power supply.

5.5 Configuration for high voltage/current driver and LEDs

![High voltage/current driver and LEDs](image)

**Figure 12. High voltage/current driver configuration (From [14])**
As shown in the figure above, the output from the Arduino MEGA 2560 is connected to input IN1 and, as a result, the output OUT1 connects to the LED. The LED must also be connected to the power source in order to have the power to light up. Above the connection for the LED is the connection from the power supply to the high voltage/current driver.

5.6 Arduino Mega 2560 Programming in C

5.6.1 General declaration and setup

```c
static unsigned int vu_cnt = 1000;//volume up interval
static unsigned int vd_cnt = 1000;//volume down interval
unsigned char g_volume = 40;//used for controlling the volume
int redPwm = 200;//used for controlling the brightness of red led
unsigned int greenFreq = 5000;//used for controlling the flash frequency of green led

unsigned char playStop = 1;// play or stop flag, 1-play, 0-stop

unsigned char currentFile = 0;
int ledPin = 53; // LED connected to digital pin 53
int irqpin = 52; // Digital 52
boolean touchStates[12];// to keep track of the previous touch states
int i;
```

The code above declares the variables that are used for the application. Note that:

- "int ledPin = 53;" declaring that the pin that used to light LEDs is digital pin number 53 in the Arduino Mega 2560 board.

- "int irqpin = 52;" declaring that the pin that used for interrupt pin of MPR121 is digital pin number 52 in the Arduino Mega 2560 board.
This is the setup of the whole application. In the method, the interrupt pin is set as the input and high, which means, turn on. The LEDs pin is also set as the output of the program.

The methods shown below enable the Music Shield to play.

```cpp
void setup(){
    pinMode(irqpin, INPUT);
    digitalWrite(irqpin, HIGH);  //enable pullup resistor
    pinMode(ledPin, OUTPUT); // sets the digital pin as output
    Serial.begin(9600);
    Wire.begin();
    Serial.println("touch Begin...");
    mpr121_setup();
    InitSPI();
    InitIOForVs10xx();
    InitIOForKeys();
    InitIOForLEDs();
    InitFileSystem();
    Mp3Reset();
}
```

This loop will keep running once the program has been started in order to check whether or not the pins from the MPR121 have been touched.

```cpp
void loop(){
    readTouchInputs();
}
```
5.6.2 **ReadTouchInputs() Method**

```
for (i=0; i < 12; i++) // Check what electrodes were pressed
    if(touched & (1<<i)){
        if(touchStates[i] == 0){ //pin i was just touched
            Serial.print("pin ");
            Serial.print(i);
            Serial.println(" was just touched");
        }
    }
```

The portion of code above explains how the electrodes are touched and what function is assigned to each of the pins. The program will check whether the pin is pressed or not by a “for” statement. If touchState = 0, which means true, then the pin has been touched and the print function will print out in the Serial Monitor to show the user which of the 12 pins is being touched. Integer "I" is assigned as the pin number.

```
//play song
    if(i == 0)
        {Play();}
    //pause and play
    if(i == 1)
        {
            Serial.println();
            Serial.println("Pause…");
            playStop = 1-playStop;
            delay(20);
            while(0 == PSKey); delay(20);
        }
```

If i = 0, which means pin 0 was touched, then the program will run the Play() function which will start the music.

If i = 1, pin 1 was touched, it will print out to the Serial Monitor that this is the Pause function. PSKey is defined as a register of function Play/pause in the Music Shield in config.h file.
The code above shows what happens if pin number 2 is touched and the corresponding red light that also changes when the volume is changed.

//increase volume
if (i == 3)
{
    Serial.println();
    Serial.println("Volumn increased...");
    --vu_cnt == 0;

    //Change + limit to 0 (maximum volume)
    if (g_volume-- == 0) g_volume = 0;
    Mp3SetVolume(g_volume,g_volume);
    redPwm = 305-(g_volume<<1);

    if(redPwm >255) {
        redPwm = 255;
    }
    if(redPwm < 0) {
        redPwm = 0;
    }

    vu_cnt = 1000;
}

//reduce volume
if (i == 2)
{
    Serial.println();
    Serial.println("Volumn reduced...");
    --vd_cnt == 0;

    //Change + limit to 254 (minimum vol)
    if (g_volume++ == 254) g_volume = 254;
    Mp3SetVolume(g_volume,g_volume);
    redPwm = 305-(g_volume<<1);

    if(redPwm >255) {
        redPwm = 255;
    }
    if(redPwm < 0) {
        redPwm = 0;
    }

    vd_cnt = 1000;
}
When pin number 3 is touched, the volume of the song will increase and it will be printed out in the Serial Monitor.

```c
if(i == 4)
{
    Serial.println();
    Serial.println("Go to previous song...");
    playingState = PS_PREVIOUS_SONG;
    delay(20);
    while(0 == BKKey);
    delay(20);
}
```

```c
if(i == 5)
{
    Serial.println();
    Serial.println("Go to next song...");
    playingState = PS_NEXT_SONG;
    delay(20);
    while(0 == NTKey);
    delay(20);
}
```

The code shows how to change to the previous song and to the next song.

The playingStage will change according to the pin that is being touched. BKKey and NTKey are defined as a register of the function Previous and Next song in the Music Shield in config.h file.
5.6.3 **Play() method**

```c
void Play()
{
    digitalWrite(ledPin, HIGH); // sets the LED on
    playingState = PS_NEXT_SONG;
    currentFile = 1;

    // cycle play
    while(1)
    {
        AvailableProcessorTime();
        if (1 == playStop)
        {
            if (OpenFile(currentFile)) {
                // if open failed, then try it again
                if (OpenFile(currentFile)) {
                    playStop = 0;
                    playingState = PS_NEXT_SONG;
                    currentFile = 1;
                    continue;
                }
            }
        }
        PlayCurrentFile();
        if (playingState == PS_PREVIOUS_SONG) currentFile--;
        if (playingState == PS_NEXT_SONG) currentFile++;
        if (currentFile == 0) currentFile = 1;
        if (playingState == PS_END_OF_SONG) playingState = PS_NORMAL;
       Mp3SoftReset();
    }
}
```

This is the Play() method which will be started when the user presses pin 0. Once it has started, the LEDs will light up (since it was set to HIGH). In addition, a “while” loop is used, which will play the song continuously until it is stopped, when the Next song pin is touched.

While the song is playing, the method `AvailableProcessorTime();` is called continuously and, in turn, this method will call the `readTouchInputs();` method that is used to check whether certain pins of the sensor MPR121 are being touched. If any is touched, then an action is performed while the song plays. For instance, if the user touched pin 1, the music will pause and if the user touched it again, the song will continue.
5.6.4 **AvailableProcessorTime() Method**

```c
void AvailableProcessorTime()
{
    do {
        digitalWrite(ledPin, LOW);  // sets the LED off
        readTouchInputs();
        if (0 == playStop) {
            GREEN_LED_ON();
        }
    } while (0 == playStop);
    // do other things
    ControlLed();
}
```

The code above shows how the method `AvailableProcessorTime()` is programmed. This method is a loop which keeps checking for the pin’s input by calling the `readTouchInputs()` method. Also the LEDs are turned on and off in order to obtain the blinking effect.
6 Result

6.1 The demo

Final product is shown in the picture:

![Figure 13. Final product](image)

When a user plugs in the power source, the LEDs light up.

![Figure 14. LEDs light up](image)
When the user pressed pin 0 (Start), the LEDs start to blink.

![Figure 15. Blinking LEDs](image)

In a real experiment for the project, the speaker still creates noise because there is a mismatch in impedance between the speaker (2 to 3 Ohm) and the amplifier (4 to 8 Ohm). However, the speaker can be used to play music because it reaches the human hearing frequency range (from 12Hz to 20KHz).

### 6.2 Result in simulation

Since it is not possible to bring sound into the report, the simulation result shown on a computer will be used instead.

Once the power source is given, on the screen will appear “Touch begin...” which can be understood as the Music application is ready.

![Figure 16. Begin](image)

When the user touches pin 0, the music application starts and it will play the first song as it appears in the figure below.
Figure 17. Start

If the user wants to pause the song, he/she can simply touch pin 1 for Pause and to continue with the song, touch pin 1 one more time.

Figure 18. Pause/play function

Pin 2 has the function of reducing the volume, when the user touches it, the volume will reduce.

Figure 19. Volume reduced

To increase the volume again, the user can touch pin 3

Figure 20. Volume increased

To change to a previous song, pin 4 can be touched

Figure 21. Previous song
Once the user wants to change to the next song, pin 5 can be touched in order for this to occur.

```
pin 5 was just touched
Go to next song...
pin 5 is no longer being touched

Found file name is DEAD-A-1MP3
File size: 7186048 bytes.
```

**Figure 22. Next song**
7 Conclusion

The overall aim of this thesis work is to conduct research on the speaker provided by Dr. Henrik Andersson, and the main focus was on the design of the speaker, and based on that, it was combined with a few sensors, magnets, a thin foil speaker and some other tools to develop a complete music thin loudspeaker as a demo version for future research.

The results above show that almost all the initial goals for developing this application were attained. The magnetostatic speaker was built from scratch. The music speaker controller can perform basic functions: Play/Start-up, Pause, and Volume down, Volume up, Previous song and Next song. To replace that function, whenever the music starts to play, the LEDs blink according to the beat of the song being played. The relay performed well and gave power to the LEDs. The speaker can produce sound even though there is still some noise because of the mismatch impedance between the speaker and the amplifier. The sound quality of the speaker, designed in this project, is not really the best because noise still exists, but users can still enjoy the music from the speaker.

With all the setting and programming above, users can easily use the music thin loudspeaker without any problems.
8 Further works

To improve on this thesis project, additional works can be added, for example: improve the sound quality by using a different amplifier, program the LEDs so that by touching the pin, the LEDs can go off or totally control the LEDs to turn on and off by using the MPR121 pin. The Demo box can be designed to make a better layout of the product.

As this would involve a significant amount of time to implement it is thus reserved for future work.
Reference

1. Arduino Mega 2560 datasheet/ ATmega 2560
2. Music shield datasheet
3. MPR121 breakout board
   http://www.sparkfun.com/datasheets/Components/MPR121.pdf
8. SD card: http://en.wikipedia.org/wiki/SD_card
10. Spectrum:
   http://en.wikipedia.org/wiki/Frequency_spectrum
11. Magnetostatic speaker:
    http://en.wikipedia.org/wiki/Loudspeaker#Ribbon_and_planar_magnetic_loudspeakers
13. Film speaker: http://film.koreasme.com/fils_03.htm
14. List of the components:
    http://www.cksinfo.com/electronics/computers/laptops/index.html,
    http://www.tuxbrain.com/en/oscommerce/products/100,
    http://www.flickr.com/photos/seeedstudio/4925513587/,
    http://www.quasarelectronics.com/3105-18w-btl-mono-amplifier-module-ha13118.htm,
    http://www.electrokit.com/en/power-pack-312v-500ma-unregulated.42680,
Appendix A: mpr121_edited_worked.ino

#include <avr/io.h>
#include "config.h"
#include "filesys.h"
#include "vs10xx.h"
#include "record.h"
#include <SoftwareSerial.h>
#include <Wire.h>

//define registers for MPR121
#define MHD_R 0x2B
#define NHD_R 0x2C
#define NCL_R 0x2D
#define FDL_R 0x2E
#define MHD_F 0x2F
#define NHD_F 0x30
#define NCL_F 0x31
#define FDL_F 0x32
#define ELE0_T 0x41
#define ELE0_R 0x42
#define ELE1_T 0x43
#define ELE1_R 0x44
#define ELE2_T 0x45
#define ELE2_R 0x46
#define ELE3_T 0x47
#define ELE3_R 0x48
#define ELE4_T 0x49
#define ELE4_R 0x4A
#define ELE5_T 0x4B
#define ELE5_R 0x4C
#define ELE6_T 0x4D
#define ELE6_R 0x4E
#define ELE7_T 0x4F
#define ELE7_R 0x50
#define ELE8_T 0x51
#define ELE8_R 0x52
#define ELE9_T 0x53
#define ELE9_R 0x54
#define ELE10_T 0x55
#define ELE10_R 0x56
#define ELE11_T 0x57
#define ELE11_R 0x58
#define FIL_CFG 0x5D
#define ELE_CFG 0x5E
#define GPIO_CTRL0 0x73
#define GPIO_CTRL1 0x74
#define GPIO_DATA 0x75
#define GPIO_DIR 0x76
#define GPIO_EN 0x77
#define GPIO_SET 0x78
```cpp
#define GPIO_CLEAR 0x79
#define GPIO_TOGGLE 0x7A
#define ATO_CFG0 0x7B
#define ATO_CFGU 0x7D
#define ATO_CFGL 0x7E
#define ATO_CFGT 0x7F
#define TOU_THRESH 0x06
#define REL_THRESH 0x0A

SoftwareSerial mySerial(2, 3); // pin2-Rx, pin3-Tx (note: pin3 is actually later used as volume down input)

/** Playing State Global */
playingstatetype playingState = PS_NORMAL;

static unsigned int vu_cnt = 1000; // volume up interval
static unsigned int vd_cnt = 1000; // volume down interval
unsigned char g_volume = 40; // used for controlling the volume
int redPwm = 200; // used for controlling the brightness of red led
unsigned int greenFreq = 5000; // used for controlling the flash frequency of green led

unsigned char playStop = 1; // play or stop flag, 1-play, 0-stop

unsigned char currentFile = 0;
int ledPin = 53; // LED connected to digital pin 53
int irqpin = 52; // Digital 52
boolean touchStates[12]; // to keep track of the previous touch states
int i;

void setup(){
  pinMode(irqpin, INPUT);
  digitalWrite(irqpin, HIGH); // enable pullup resistor
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
  Serial.begin(9600);
  Wire.begin();
  Serial.println("touch Begin...");
  mpr121_setup();
  InitSPI();
  InitIOForVs10xx();
  InitIOForKeys();
  InitIOForLEDs();
  InitFileSystem();
  Mp3Reset();
}

void loop(){
  readTouchInputs();
}

void readTouchInputs(){
  if(!checkInterrupt()){
    // read the touch state from the MPR121
    Wire.requestFrom(0x5A, 2);
  }
}
```
byte LSB = Wire.read();
byte MSB = Wire.read();

uint16_t touched = ((MSB << 8) | LSB); //16bits that make up the touch states

for (i=0; i < 12; i++){  // Check what electrodes were pressed
  if(touched & (1<<i)){
    if(touchStates[i] == 0){
      //pin i was just touched
      Serial.print("pin ");
      Serial.print(i);
      Serial.println(" was just touched");
      
      //play song
      if(i == 0)
      {
        Play();
      }
    }
    //pause and play
    if(i == 1)
    {
      Serial.println();
      Serial.println("Pause...");
      playStop = 1-playStop;
      delay(20);
      while(0 == PSKey);
      delay(20);
    }
    //increase volume
    if(i == 2)
    {
      Serial.println();
      Serial.println("Volumn reduced...");
      --vd_cnt == 0;
      if (g_volume++ == 254) g_volume = 254; //Change + limit to 254 (minimum vol)
    }

  }
}
//reduce volume
if(i == 3)
{
    Serial.println();
    Serial.println("Volumn increased...");
    --vu_cnt == 0;
    if (g_volume -- == 0) g_volume = 0; //Change + limit to 0
    (maximum volume)
    Mp3SetVolume(g_volume,g_volume);
    redPwm = (175-g_volume)*3>>1;
    if(redPwm > 255)
    {
        redPwm = 255;
    }
    if(redPwm < 0)
    {
        redPwm = 0;
    }
    vu_cnt = 1000;
}
if(i == 4)
{
    Serial.println();
    Serial.println("Go to previous song...");
    playingState = PS_PREVIOUS_SONG;
    delay(20);
    while(0 == BKKey);
    delay(20);
}
if(i == 5)
{
    Serial.println();
    Serial.println("Go to next song...");
    playingState = PS_NEXT_SONG;
    delay(20);
    while(0 == NTKey);
    delay(20);
}
}else if(touchStates[i] == 1){
}
touchStates[i] = 1;
}else{
    if(touchStates[i] == 1){
        Serial.print("pin ");
        Serial.print(i);
        Serial.println(" is no longer being touched");
    }
void mpr121_setup(void){
  set_register(0x5A, ELE_CFG, 0x00);
  // Section A - Controls filtering when data is > baseline.
  set_register(0x5A, MHD_R, 0x01);
  set_register(0x5A, NHD_R, 0x01);
  set_register(0x5A, NCL_R, 0x00);
  set_register(0x5A, FDL_R, 0x00);
  // Section B - Controls filtering when data is < baseline.
  set_register(0x5A, MHD_F, 0x01);
  set_register(0x5A, NHD_F, 0x01);
  set_register(0x5A, NCL_F, 0xFF);
  set_register(0x5A, FDL_F, 0x02);
  // Section C - Sets touch and release thresholds for each electrode
  set_register(0x5A, ELE0_T, TOU_THRESH);
  set_register(0x5A, ELE0_R, REL_THRESH);
  set_register(0x5A, ELE1_T, TOU_THRESH);
  set_register(0x5A, ELE1_R, REL_THRESH);
  set_register(0x5A, ELE2_T, TOU_THRESH);
  set_register(0x5A, ELE2_R, REL_THRESH);
  set_register(0x5A, ELE3_T, TOU_THRESH);
  set_register(0x5A, ELE3_R, REL_THRESH);
  set_register(0x5A, ELE4_T, TOU_THRESH);
  set_register(0x5A, ELE4_R, REL_THRESH);
  set_register(0x5A, ELE5_T, TOU_THRESH);
  set_register(0x5A, ELE5_R, REL_THRESH);
  set_register(0x5A, ELE6_T, TOU_THRESH);
  set_register(0x5A, ELE6_R, REL_THRESH);
  set_register(0x5A, ELE7_T, TOU_THRESH);
  set_register(0x5A, ELE7_R, REL_THRESH);
  set_register(0x5A, ELE8_T, TOU_THRESH);
  set_register(0x5A, ELE8_R, REL_THRESH);
  set_register(0x5A, ELE9_T, TOU_THRESH);
  set_register(0x5A, ELE9_R, REL_THRESH);
}

public int readADC(int pin) {
  return analogRead(pin);
}

float readTouch(int i) {
  return analogRead(touchStates[i]);
}

int main() {
  touchStates[0] = 0;
  touchStates[1] = 0;
  touchStates[2] = 0;
  touchStates[3] = 0;
  return 0;
}
set_register(0x5A, ELE10_T, TOU_THRESH);
set_register(0x5A, ELE10_R, REL_THRESH);

set_register(0x5A, ELE11_T, TOU_THRESH);
set_register(0x5A, ELE11_R, REL_THRESH);

// Section D
// Set the Filter Configuration
// Set ESI2
set_register(0x5A, FIL_CFG, 0x04);

// Section E
// Electrode Configuration
// Set ELE_CFG to 0x00 to return to standby mode
set_register(0x5A, ELE_CFG, 0x0C); // Enables all 12 Electrodes

// Section F
// Enable Auto Config and auto Reconfig
/*set_register(0x5A, ATO_CFG0, 0x0B);
set_register(0x5A, ATO_CFGU, 0xC9); // USL = (Vdd-0.7)/vdd*256 = 0xC9 @3.3V
set_register(0x5A, ATO_CFGL, 0x82); // LSL = 0.65*USL = 0x82 @3.3V
set_register(0x5A, ATO_CFGT, 0xB5);*/ // Target = 0.9*USL = 0xB5 @3.3V
set_register(0x5A, ELE_CFG, 0x0C);
}

boolean checkInterrupt(){
    return digitalRead(irqpin);
}

void set_register(int address, unsigned char r, unsigned char v){
    Wire.beginTransmission(address);
    Wire.write(r);
    Wire.write(v);
    Wire.endTransmission();
}

/** Plays a disk file. Returns 1) if the file ends or 2) if the global
variable playingState is not PS_NORMAL i.e. user has requested
stop or next or previous.*/
void PlayCurrentFile()
{
    char c, nFragments;

    playingState = PS_NORMAL; /* Request to play normally*/
    //uiMode = UI_SPEC; /* User interface: show title SPECANA FOR VS1003*/

    //LcdLocateHome();
    //LcdPutConstantString("Opening ");

    //Serial.print("\r\nBuilding file fragment table...");
delay(100); // delay here is very important, give some time to sd card. --- by Icing
nFragments = BuildFragmentTable(); /* Too slow, rewrite */
// Serial.print("Fragments: ");
// Serial.print(nFragments, DEC);

// LcdLocateHome();
// LcdPutConstantString("Playing ");

for (c=0; c<nFragments; c++) {
    sectorAddress.l = fragment[c].start;
    // ConsoleWrite ("\r\n\nPlayer: Playing from sector ");
    // ConsolePutUInt (sectorAddress.l);
    if (PlayDiskSectors(fragment[c].length) != 0) {
        Mp3WriteRegister(SPI_MODE, 0, SM_OUTOFWAV);
        SendZerosToVS10xx();
        return; // return without touching the value of playingState
    }
}
SendZerosToVS10xx();

// After finishing normally default to requesting to play next song
playingState = PS_NEXT_SONG;
}

void ControlLed()
{
    static unsigned char greenOnOff = 1;
    
    if (0 == greenFreq -- )
    {
        greenOnOff = 1 - greenOnOff;
        greenFreq = 5000;
    }
    if (greenOnOff)
    {
        GREEN_LED_ON();
    }
    else
    {
        GREEN_LED_OFF();
    }
    
    analogWrite(9, redPwm);
}

/** This function is called when the player is playing a song and there is free processor time. The basic task of this function is to implement the player user interface. */
void AvailableProcessorTime()
{
    do
    {
        digitalWrite(ledPin, LOW); // sets the LED off
readTouchInputs();

    if(0 == playStop)
    {
        GREEN_LED_ON();
    }
}while(0 == playStop);
//do other things
ControlLed();

void Play()
{

digitalWrite(ledPin, HIGH); // sets the LED on
playingState = PS_NEXT_SONG;
currentFile = 1;

//cyclely play
while(1)
{
    AvailableProcessorTime();

    if(1 == playStop)
    {
        if(OpenFile(currentFile))
        {
            //if open failed, then try it again
            if(OpenFile(currentFile))
            {
                playStop = 0;
                playingState = PS_NEXT_SONG;
                currentFile = 1;
                continue;
            }
        }
    }

    PlayCurrentFile();

    if (playingState == PS_PREVIOUS_SONG) currentFile--;
    if (playingState == PS_NEXT_SONG) currentFile++;
    if (currentFile==0) currentFile = 1;
    if (playingState == PS_END_OF_SONG) playingState = PS_NORMAL;
    Mp3SoftReset();
}
}
### Appendix B: Components List

<table>
<thead>
<tr>
<th>No.</th>
<th>Component Name</th>
<th>Type of component</th>
<th>Quantity</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arduino MEGA2560</td>
<td>Sensor</td>
<td>1</td>
<td>Main sensor</td>
</tr>
<tr>
<td>2</td>
<td>Music Shield v1.13</td>
<td>Sensor</td>
<td>1</td>
<td>5.5V</td>
</tr>
<tr>
<td>3</td>
<td>MPR121 Capacitive Touch Sensor Breakout Board</td>
<td>Sensor</td>
<td>1</td>
<td>3.3V</td>
</tr>
<tr>
<td>4</td>
<td>Power supply</td>
<td>Power supply</td>
<td>1</td>
<td>3V to 12V</td>
</tr>
<tr>
<td>5</td>
<td>18W amplifier</td>
<td>Amplifier</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1GB SD card</td>
<td>Memory card</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>LEDs</td>
<td>Light</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>High voltage/Current Driver</td>
<td>Relay</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Foil speaker</td>
<td>Speaker</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Block magnet</td>
<td>Magnet</td>
<td>75</td>
<td>9.8x3.8x2.8mm</td>
</tr>
<tr>
<td>11</td>
<td>10k - 50 k log Pot</td>
<td>Sliding resistor</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Connecting cable</td>
<td>Cable</td>
<td>Several</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Circuit board</td>
<td>Board</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C:

Mid Sweden University
The Department of Information Technology and Media (ITM)
B.Sc. Thesis report
International Bachelor Programme in Electronics, 180 Credits
Design and Construction
Of A Paper Thin Magnetostatic Loudspeaker
Minh Huyen Do Thi