An Arduino-based MIDI Controller for Detecting Minimal Movements in Severely Disabled Children

Mattias Linder
Abstract

An Arduino-based MIDI Controller for Detecting Minimal Movements in Severely Disabled Children

Mattias Linder

In therapy, music has played an important role for children with physical and cognitive impairments. Due to the nature of different impairments, many traditional instruments can be very hard to play. This thesis describes the development of a product in which electrical sensors can be used as a way of creating sound. These sensors can be used to create specially crafted controllers and thus making it possible for children with different impairments to create music or sound. This thesis examines if it is possible to create such a device with the help of an Arduino micro controller, a smart phone and a computer. The end result is a product that can use several sensors simultaneously to either generate notes, change the volume of a note or controlling the pitch of a note. There are three inputs for specially crafted sensors and three static potentiometers which can also be used as specially crafted sensors. The sensor inputs for the device are mini tele (2.5mm) and any sensor can be used as long as it can be equipped with this connector. The product is used together with a smartphone application to upload different settings and a computer with a music work station which interprets the device as a MIDI synthesizer.
Contents

1 Introduction ......................................................... 3
  1.1 Background .................................................. 3
  1.2 Research question ......................................... 4
  1.3 Delimitation ................................................. 4

2 Similar and related products ...................................... 5
  2.1 Related work ................................................ 5
  2.2 Similar hardware ............................................ 5
  2.3 MIDI .......................................................... 6

3 Digital music instruments .......................................... 7

4 Overview of product ............................................... 9
  4.1 Music workstation ........................................... 9

5 Software .......................................................... 10
  5.1 Routing MIDI messages ..................................... 10
    5.1.1 Hairless-midiserial ..................................... 10
  5.2 iPhone Design ............................................... 11

6 Final prototype .................................................... 16

7 System ................................................................... 17
  7.1 Setup .......................................................... 17
  7.2 Calibration .................................................... 17
  7.3 Running ......................................................... 18

8 Hardware ................................................................ 19
  8.1 Arduino ........................................................ 19
  8.2 Bluetooth low energy ......................................... 19
  8.3 Sensors and similar components .............................. 20
  8.4 Development .................................................. 21

9 Summary ................................................................ 25
  9.1 Performance .................................................... 25
  9.2 Conclusion ...................................................... 25
  9.3 Future work and improvements ............................... 26
1 Introduction

Programming with different micro controllers has recently become more and more common for various types of tasks. Micro controllers and different modules for these chips become cheaper and more accessible \[28\]. With a cheap micro controller and some sensors there is much that can be done. This thesis project aims to use a collection of three components, namely a microchip, arbitrary analogue sensors and a smart phone \[11\], to allow severely disabled children to create sound and even music. The idea of this part of the project is that one should be able to connect a sensor to an input on a microchip, tell that microchip what that input regulates, via an application on a phone, and then start generating sound on software synthesizers with the input from the different sensors. This thesis project is made as a part of a larger project, MuMin (Maximum Output and Minimal Interaction) \[35\], for the VI2 unit at the department of Information Technology at Uppsala University. The main focus for the whole project is to enable children with very severe cognitive and/or physical disabilities to be able to create music on their own accord, with an electronic instrument. The project runs in close connection with the Arsta special school in Uppsala. The outcome of the project is intended to be used by the music teachers and special pedagogues at Arsta special needs school, and possibly also later in other schools around Sweden.

1.1 Background

Music has been a very important part of the therapy for children with severe cognitive and physical impairments. Learning to play an instrument can have many positive effects, including improved attention, improved social skills, better memory and more \[33\] \[31\]. However, due to the nature of the impairments music therapy becomes mainly a semi-passive activity for the children, that is, where the child either is listening to the music or follows the teachers piano playing by hitting a drum, or special single string guitars. Traditional music instruments, however, often require a relatively large physical effort and coordination, something that many of these children lack. The children of the target group are severely impaired, most of them with multiple impairments. The communication means are very limited, and few of them have a spoken language. The teachers use a simple sign language to communicate with the children, and they often have to rely on their personal experience of each individual child.

The overall intention with this project, for which this thesis is written, is to enable the children in this specific target group to take a more active part in generating music. The basic idea is that through the use of synthesizers and specially designed controllers, the children can use their individual abilities to produce music, instead of taking a passive role. Each child has certain physical and cognitive skills, which will be the focus of the design for the controller that is adapted to him or her.

Since the controllers are individually designed, there is also a need to be able to connect them to the synthesizer, and this is where the current thesis comes in. Using an Arduino \[3\] board, the aim is to create a micro controller that converts the controller signals to MIDI (Musical Instrument Digital Interface) \[20\]. Ideally a new controller should be more or less plug-and-play, albeit with a possibility to make some smaller adjustments if needed. The application should be using an iPhone or iPad \[14\] for the interface, since this is a tool that is used at the school in question.

Initially the project used ”traditional” hardware controls, e.g., drum pads, ribbon controllers, and very sensitive keyboards. One special instrument used is the Theremin by Moog \[12\], which can be controlled with small movements of a single finger.
1.2 Research question

The challenges in this thesis project have been to make a MIDI controller, that is, a device that can generate MIDI messages, using a simple interface, that still can use any kind of control sensor without the user needing to know anything about the technical issues around the devices. This thesis aims to answer the following questions:

- Is it possible to create a small MIDI controller that receives signals from analogue sensors and then generates appropriate MIDI signals?
- What are the limitations of the Arduino for this kind of application?

Apart from the research questions, the project aims at producing a working prototype as an end result.

1.3 Delimitation

The focus of this project is to make a phone application only for iOS. Further, the control unit (the iPhone application) and the micro-controller will only communicate through bluetooth BLE (also referred to as BLE, bluetooth low energy, bluetooth LE or bluetooth smart). There will also be a limitation of how many different MIDI messages that can be generated by a normal user. If the user have more knowledge about MIDI messages, then there will be an option in the application where the user can input exactly what the microchip should send.
2 Similar and related products

2.1 Related work

There are many actors on the market who are creating technology in which computers control devices through different sensors. Below is a list of some of the bigger actors and their products.

TouchTone [6] is an electrical music instrument designed for children with Cerebral palsy [8]. It uses pressure sensitive pads to generate tones and a shift that is used to change the octave of the tones (see Figure 1). This makes it easier for the children to be creative when creating sound and music. Every pad has a red light which is used when a user wants to learn a song or follow an exercise.

Soundscraper (or Ljudskrapan) [23] is a system designed to help children with special needs to create and explore music. It does that by having a similar design as this project. It uses different controllers or cameras to catch the user's movement and then produce different sounds.

![TouchTone instrument](image)

Figure 1: TouchTone instrument.

2.2 Similar hardware

littleBits Electronics [19] is a company based in New York. Their product offers an easy way to build your own electrical solutions and inventions using pre-built electrical modules. littleBits has several modules that can be used to create a synthesizer. They offer several different modules, for example an oscillator and a pulse regulator. These modules are easy to connect, you just have to put them next to each other and a magnet will pull them together and create a connection between them.

Makey Makey [16] is a product invented by MIT Media Lab, Cambridge, U.S. Makey Makey is a microcontroller that can read analogue input from almost any material, the material just needs to be able to conduct a tiny bit of electricity.

A material (e.g., a fruit or a potato) is connected to the Makey Makey by an alligator clipper. Another material (e.g., the user's body) is connected to ground. When the two materials touch, a connection is made and is read by the Makey Makey board. When the materials touch, the input is translated into a digital signal, either 0 or 1. The signal is then mapped onto a key on the keyboard of a computer, making that signal controlling that key on the computer.

Microsoft kinect [3] is a motion sensor device created by Microsoft, U.S. It has the appearance of a web-camera with the difference that it can read its surrounding, making it possible to control a computer by using different gestures. The camera of the kinect should be faced against the object that is going to act as a controller. The kinect can then sense when gestures from that objects are made.
Leapmotion [32] is a company based in the U.S. Their product is similar to Microsoft kinect, the difference being that Leapmotion is placed on a desktop, with a sensor facing upwards, and then reads movement and gestures that are made above the device.

Myo [34] is a gesture control device made by Thalmic Labs located in Canada. It is an armband that lets the user wirelessly control technology with gestures and motion. It supports both motion and hand gesture detection through EMG (Electromyography) muscle detection.

2.3 MIDI

MIDI is a protocol used for different electronic instruments and computers to communicate with each other. The protocols describes what a MIDI message should contain and what different messages do. Every MIDI message is divided into two categories: the status and the data. The status byte indicates what type of information is being sent. For example the status byte can tell that there should be a note plying or there should be a pitch bending. The data portion of the message tells what values are associated with the status part [20].

An example of how a MIDI message could look like: the status part is the binary number 144, and the following data bytes could then be 61 followed by 100. The status byte 144 means that a note should be played. The first data byte 61 tells that the note C in octave 4 should be played and the following byte 100 tells at which volume the note should be played.
3 Digital music instruments

There are several theories about what makes a good instrument. Sergi Jorda suggests in his paper \[25\] that there are chances that absolute beginners will prefer an instrument that seems to offer more varied possibilities, meaning the instrument is very adaptable. However, professionals will be more satisfied with an instrument with a stronger personality, meaning a less adaptable instrument. He also notes that using a good instrument, the player should be capable of creating both good music and horrible music at will. There are apparently big differences between instruments.

Some instruments are constructed in a way that allows the user to actively play music rather than only \textit{playing with} music. These instruments tend to have fine-grained control over the nuances that can be produced. Jorda distinguishes instruments by three different criteria. An illustration of the instruments and criteria can be seen in Figure 2.

- **Macro-diversity** determines how flexible an instrument is by comparing how the instrument is performing in different contexts and roles. Instruments with high macro-diversity are generic instruments that fits many roles, like an electric-guitar, with its many sounds and style. A synthesizer could, with the help of a computer, also be used to create many different styles of music. An example of a low macro-diversity instrument is an instrument that is highly specialized in one field but hard to master. One example of this is the double bass.

- **Mid-diversity** determines how two \textit{different} musical pieces can differ on the same instrument. For example, instruments that can produce a large variety of tones and pitches have high mid-diversity. Instruments with low mid-diversity sound like they always play the same song. For example, percussions and other instruments without any pitch, have low mid-diversity while many traditional instruments such as guitars have high mid-diversity. An instrument that has high mid-diversity doesn’t necessarily have high macro-diversity, even though that is often the case. For example the double bass can have many different tones and pitches, but it is not a general purpose instrument because of the fact that it doesn’t fit many musical roles.

- **Micro-diversity** determines how two performances on the \textit{same} musical piece and with the same instrument can differ. This could be seen as how many possibilities the player has to create \textit{small nuances} and subtleties, or on digital instruments change variation in time, tempo, tone and so on. For example a kids-toy piano synthesizer that only has a certain set of sounds, can only be used to play this set of sounds. The tones themselves cannot be altered. This could be considered a low micro-diversity instrument. Traditional acoustic instruments have high micro-diversity, but also professional synthesizers with pitch control can have high micro-diversity.

![Figure 2: Macro/Micro/Mid-diversity instruments.](image-url)
Jorda argues that high macro-diversity instruments, i.e. general purpose instruments, are more suited for beginners. Mid-diversity is an essential component in order to turn a music-player into a music-performer. Micro-diversity is an important aspect in turning a musician into a virtuoso, since this diversity makes it possible to express oneself musically.
4 Overview of product

The core part of the project solution is called ArduInsto and takes usage of several other devices. A schematic picture that represents the different devices used can be seen in Figure 3. To use the device, an iPhone and a computer is needed.

![Figure 3: Schematic picture of the product](image)

To start using the prototype, the user needs an iPhone to upload settings to the ArduInsto device. These settings describes how the ArduInsto should interpret the signals from the sensors and when or how it should generate MIDI messages. The ArduInsto is routing the incoming information, from the sensors that is connected to it, through the USB (Universal Serial Bus) connection to the computer. The computer interprets these signals and creates audio output signals, which are generated from a music workstation. A later chapter describes what hardware and software was used when testing and creating this prototype.

4.1 Music workstation

A music workstation is a program where music can be produced. In this project, Apple’s Logic Pro X was used. It doesn’t matter which workstation is used when using ArduInsto, but it has to be able to receive MIDI signals. Other programs that have similar functionalities as Logic Pro X are Propellerheads Reason, Steinbergs Cubase, Apple GarageBand etc. The most important functionality of the workstation, for this project, is that it can produce different sounds (e.g. guitar sound, pad, violin) out of the received MIDI messages, and then outputs that sound on a chosen audio interface. The MIDI message itself is not an audio signal.
5 Software

The design of the prototype aims to give a high degree of usability to the user. Benyon writes that a system should have the following characteristics to have a high usability:

- It will be efficient in the sense that people will be able to do things using an appropriate amount of effort.
- It will be effective in that it contains the appropriate functions and information content, organized in an appropriate manner.
- It will be easy to learn how to do things and remember how to do them after a while.
- It will be safe to operate in the variety of contexts in which it will be used.
- It will have high utility in that it does the things that people wants to get done.

Both the iPhone application and the hardware are designed based on these principles.

5.1 Routing MIDI messages

As mentioned above, this project utilizes a music workstation to generate the final sound presented to the user. It also uses an open source software for routing MIDI messages from the USB connection on the computer to the music workstation. This software is called Hairless-midiserial (description follows later). Below is a guide on how to set up the Hairless midi-serial on a Macintosh.

1. In spotlight, search for "Audio MIDI SETUP" and press the icon that looks like a synthesizer. Navigate to Applications/Utilities and select Audio MIDI Setup application.
2. In the Audio MIDI Setup application, double click the IAC-driver (Inter-application communication). An IAC-driver properties window should appear.
3. Press the "+" icon on the bottom left of the window and choose an appropriate name.
4. Open Hairless-midiserial application. Now there should be an IAC-driver option in the MIDI out drop down menu in the Hairless application. Choose this option.
5. In the workstation, in this case logic pro X, select a MIDI instrument. Tell the MIDI instrument to listen to either IAC-driver out or to listen to all MIDI channels. This step is done differently in every music workstation program. In logic pro X, and if all previous steps where done successfully, this is done automatically when a new software instrument is added.

5.1.1 Hairless-midiserial

This project uses the application Hairless-midiserial. It is used to receive MIDI messages from the Arduino and redirect the messages to a MIDI port which a music workstation is listening to. It is a free cross platform application that talks to serial connected devices. It is able to send MIDI signals to a chosen MIDI port or receive MIDI signals from a chosen MIDI port or serial connected device. The Arduino is connected through USB to a computer where Hairless-midiserial is installed. The Arduino send all data through this USB connection, where Hairless-midiserial then receive the data and can redirect it to the intended port.
5.2 iPhone Design

The design of the app takes a human-centered perspective. The development started with a prototype that did not take a human-centered perspective, but rather the perspective of the developer. However, the design had flaws and a redesign of the whole application was needed. The first prototype can be seen in the left picture in Figure 4, and the redesign can be seen in the right picture in the same figure.

In the redesign the buttons are larger, taking up almost all the space of the screen. The buttons are marked with an icon and a title describing the functionality of the button.

The list of added instruments/controllers is moved to a new single page instead of being on the main page. This can be seen in the left picture in Figure 5. Also a left swipe function for deleting an item was added in the new prototype. There is a "button activator" setting on the main page of the old design. This setting is moved to a new page called Settings. This is where the user can change certain settings for how the prototype should work. Right now there is only the "Button activator" setting, as seen in Figure 5, but in the future there might be more settings added. When these changes were implemented, a grouping of the button layout was introduced. This is described later. The rest of this section will mostly focus on the new prototype design instead of comparing the two designs.

Figure 4: On the left is the first prototype of the main page and on the right is the redesigned prototype.

The new system design tries to be consistent in the way that the layout and buttons are designed. Every button that makes the same conceptual action has a similar look. For example the buttons that makes the user go back in the application have the same height, length,
describing title and position. Also the ”Apply” and ”Back to beginning and reset” buttons have these properties. The buttons on the main page, see the right picture in Figure 4, are grouped into small groups. The ”Add instrument” and ”Add custom instrument” buttons are grouped together on the first row side by side, and are of the same size and have similar icons and describing titles. They essentially have the same functionality, making it natural to group these two buttons together.

The two buttons beneath are ”Added instruments” and ”Settings”. These are harder to group together since they have different functionalities. However, both have the same conceptual function as they change the settings in either an added instrument or some setting for the overall system. To show that they still have some uncommon functionalities they are assigned with completely different icons and describing texts.

Figure 5: On the left is the page for added instruments and on the right is the page for settings.

These four buttons can also be considered a group since they all have the same size and are grouped like a square. This is intentional since these buttons control all the functions that change or add information that is later to be sent to the Arduino.

The ”Upload settings and instruments” button is a group in its own. It has an essentially different functionality than the other buttons and is only to be used when all the settings are set and it is time to upload data to the Arduino.

The screen that appears when pressing the ”Add instrument” button can be seen in Figure 4. On the picture on the left, a controller that generates notes is created if the user presses ”Apply”. It is set to receive signals from input 3 on the Arduino chip. On the picture to the right, a fine pitch controller is added. Since the fine pitch only changes the pitch of a tone, we need to specify which note that should be changed in pitch. This is specified by the different
numbers under the "Output" label, above the "Apply" button.

As an example, you might add a controller that generates notes on input 1. You could then create a fine pitch controller on input 2 and then specify the output for that controller as 1. Then the system will generate notes from the sensor on input 1, and the sensor on input 2 will change the pitch on the notes created from the sensor on input 1. This works in the same way for the volume and octave controllers.

![Figure 6: On the left, a note controller is added. On the right, a pitch controller is added.](image)

The system has two reset/recovery buttons. They can be found in the main page, see the right picture in Figure 4, and in the calibration page seen in Figure 7. These buttons reset the bluetooth connection and reset all settings to their default values. This function should be useful if the user wants to start from the beginning or if something has gone wrong (for example if the bluetooth connection is lost or it isn’t able to create a bluetooth connection).

Some constraints are also added to the system. The user cannot upload an empty list of instruments to the Arduino. To be able to add an instrument, the user needs to fill in certain fields (e.g. the name of the instrument and an input needs to be specified). Further, the name of a controller cannot be longer than 20 characters.

These constraints are presented with an alert message. The alert message presented tries to explain what went wrong and how to recover from it. It does not try to accuse the user of making any faults but rather just informs about what happened and how to solve it.
When an instrument is added, there is a notification on the "Added instruments" button, showing a number on how many instruments that has been added this far. In this way the user gets feedback on how many instruments they have currently added. When the user presses the upload button, an alert comes up with a loading circle showing the application is busy. The user can either wait for it to finish or cancel the action to get back to the main page. When the loading is done the calibration page is shown.

There are also some things worth mentioning regarding the colour schema. The white background and the teal colours of buttons and tables are inspired by the Arduino homepage, which can be seen in Figure 8. Jonas Löwgren gives one short principle on how to choose a colour schema: to keep it simple. By that he means that a good design in colours is to choose low-intensity colours and use matching colours [27].

One might argue that the colour schema from the Arduino homepage is kept simple, since it has has both matching and not very intense colours. The app doesn’t rely on the users ability to differentiate between colours to be able to use it.

The main colours of the iPhone application are white, teal and blue. When an instrument is added, a red colour is used. That red colour has a higher brightness value than the teal colour behind it, making it possible for people with red/green or total colourblindness to differentiate the change in colour [17]. The different colours of the application have been tested at Vischecks.
The web-page. Each page/view in the application have been tested and verified according to Vischecks tests.

The final icon of the application can be seen in Figure 9.

Figure 8: Arduino homepage.

Figure 9: Final app icon.

---

2 Picture taken from https://www.Arduino.cc
3 Icon created by artist Jakob Lindh
6 Final prototype

The hardware prototype is a grey plastic and metallic box that has several different inputs. There is a USB connection on its right side which is used when connecting it to a computer. It has three audio inputs on top, three potentiometers on the front and three potentiometers on the back. There is a reset button on the left, a ground connection on the left and a light-emitting diode on the front (showing that it is powered on when lit, and off when not).

![Final product](image)

Figure 10: Final product.

The three audio inputs in Figure 10 are where the user connects a controller/sensor. In the beginning, a breadboard and jumper cables were used to connect sensors to the Arduino, but that requires previous knowledge about how electronics works. The hardware design was later changed so that a controller is connected to an audio cable, making it easier to connect sensors to the prototype. There is a screw that is connected to ground, which a sensor can use if it needs a ground connection (e.g. the proximity sensor used when developing this prototype).

The potentiometers on the back of the prototype regulate the resistance on the sensors that are connected on the top. Since some sensors require a high resistance and some low resistance, the potentiometer was added to make it possible to connect a wide range of different sensors.

The potentiometers on the front are static controllers. Their functionality is set in the iPhone application. They function as any other controller that is connected on the top, except that they are always connected to the prototype. They are added in the same way as any other controller (i.e. by using the iPhone application).

The light-emitting diode on the front is used to show if the system is powered on. If it is emitting light the system is turned on, otherwise it is turned off. The reset button on the left restarts the prototype.
7 System

When the iPhone application is started the user can only see a button that is labeled connect. This button is used to connect the iPhone via Bluetooth to the Arduino Bluetooth chip. Both the Arduino and the chip need to be powered on for the connection to be established.

The Hairless-midiserial software needs to be running and needs to be redirecting all data coming from the Arduino to a virtual MIDI output port. A music workstation needs to be running and listening to the port that Hairless-midiserial redirects the MIDI messages to.

7.1 Setup

When a connection is established the user can start adding controllers/instruments. In order to add a controller/instrument, the user needs to set the input on which the sensor/potentiometer is connected to on the prototype, give it a name and set which type of controller it is (either it sends notes, pitch, octave or volume MIDI messages). If the type of the controller is either pitch, octave or volume, an output needs to be specified. That output describes which other sensor this sensor should regulate/alter (either volume, pitch or octave).

There is a button on the prototype used for controlling when notes should be played. There are three alternatives, listed below.

- **Drone:** There is always a note playing in the background. This is called drone [1].
- **Press:** Works like a switch, meaning that when the button is pressed notes are generated, and when the button is pressed again it stops generating notes.
- **Hold:** When the button is pressed, notes are generated.

7.2 Calibration

When everything is set and all controllers are added, these settings and controllers can be uploaded to the Arduino. When the settings are successfully uploaded, there is a need to calibrate the sensors that are connected to the prototype. This is due to the sensors giving different analogue values depending on the resistance of the sensors. To start calibrating, the user presses the calibrate button. When the button is pressed the user gets an alert message which tells the user to set the sensor to a state where its resistance will have a high value. When the user puts the sensor in a high resistance state and presses the “ok” button on the screen, the Arduino reads a value from that sensor and saves it. The same thing is done again but this time the sensor should be in a low value state.

There is no real need for the user to know when the sensor has a high or low value, only that it is in an state where it has maximized on minimized output. If the Arduino reads a low value when it expects a high value, the calibration will still work as intended but with the difference being that the values from the sensors will be read in an inverted way and the sensor will also work in an inverted way, meaning that it will output (from the Arduino) low values when it actually has a high value.

When all the sensors are calibrated, the user gets an alert message telling everything was either fine or something went wrong. If everything went fine, the user can start using the prototype after pressing the “ok” button. However, if something goes wrong, the user needs to go back and start over again.
7.3 Running

When everything is setup and calibrated, the user can start generating tones, changing volumes, change the octaves and/or pitch notes depending on controllers/instruments added. The preferred virtual instrument sound that should be generated by the MIDI signals can be set in the music workstation the user has chosen to use.
8 Hardware

Every MIDI message will be generated on the Arduino chip with the help of different sensors connected to the chip. The settings for how the sensors are used and when to calibrate the sensors are controlled by the iOS application. The communication between the iOS application and the Arduino chip is made via bluetooth low energy (BLE) technology.

8.1 Arduino

Arduino is a micro controller chip with open hardware made for the purpose to be easy-to-use. It can be used with Windows, Mac or Linux. This project uses the Arduino UNO model, which has 14 digital input/output pins and 6 analogue input/output pins. In this project it was especially important with the analogue pins because the sensors that are used are analogue sensors.

The chip is powered with either a USB cable or a DC (direct current) connector, with recommended voltage of 7-12 V. The analogue inputs on the chip take an input voltage from a device/sensor and convert it to a digital integer value between 0 and 1023. The digital pins can only read two values, either HIGH (integer value 1) or LOW (integer value 0) [4].

![Arduino UNO chip](https://www.Ardunio.cc/en/uploads/Main/ArdGen_UNO.jpg)

Figure 11: Arduino UNO chip

8.2 Bluetooth low energy

This project uses a BLE connection between the iPhone application and the Arduino chip. Since there is no bluetooth chip integrated on the Arduino, this project uses the nRF8001 chip [5]. BLE is a part of the bluetooth V4.0 protocol. BLE uses less power than regular bluetooth devices but it cannot exchange as much data as regular bluetooth [6]. BLE is used in this project because the iPhone application is only used to send a small amount of data to the Arduino UNO.

8.3 Sensors and similar components

A sensor is an electrical component that can detect changes in the environment and provide that data as an electrical signal. Below is a list of different sensors, or similar components, used in this project. There will not be a detailed description on how they work, only a shorter description on how they measure their surroundings.

- **Flex sensor:** A sensor that measures a bending motion. It outputs different voltage levels when bent. [26].

- **Potentiometer:** It converts either rotary or linear motion into a change of resistance (this is not a sensor, but is used in this project).

---

• **Proximity sensor:** A sensor that is able to detect the presence of objects in front of the sensor without any physical contact.

![Proximity sensor](https://upload.wikimedia.org/wikipedia/commons/2/27/Sharp_GP2Y0A21YK_IR_proximity_sensor_cropped.jpg)

• **Piezoelectric sensor:** A sensor that converts pressure/force into electrical charge.

![Piezo sensor](https://cdn.sparkfun.com//assets/parts/2/9/6/7/09375-1.jpg)

### 8.4 Development

The development of this product has involved many different components. One part was the iPhone application and this is described in the Design section. Another part has been to solder and connect jumper wires, sensors, potentiometers and other electronics.

---

7 Picture taken from [http://sound.westhost.com/pots.htm#markings](http://sound.westhost.com/pots.htm#markings)

8 Picture taken from [https://upload.wikimedia.org/wikipedia/commons/2/27/Sharp_GP2Y0A21YK_IR_proximity_sensor_cropped.jpg](https://upload.wikimedia.org/wikipedia/commons/2/27/Sharp_GP2Y0A21YK_IR_proximity_sensor_cropped.jpg)

9 Picture taken from [https://cdn.sparkfun.com//assets/parts/2/9/6/7/09375-1.jpg](https://cdn.sparkfun.com//assets/parts/2/9/6/7/09375-1.jpg)
Figure 17: During development.

1. The Arduino Uno which is the heart of the ArduInsto.
2. A button used to reset the Arduino.
3. Several potentiometers connected the the Arduino.
4. The Ada Bluefruit chip.

Figure 17 is a picture taken during development when different connections and electronics still were being tested. In this picture nothing is soldered together and an breadboard was used to connect the different components to each other. Below are some pictures of different components/sensors used during development.
Figure 18: Mini tele connections were later added. This was done so it would be easier to connect sensors to the hardware without having any prior knowledge on how to connect electronics.

Figure 19: Proximity sensor used during testing. This sensor was hard to generate notes on, but it was better at controlling the pitch or volume of notes generated by other sensors.
Figure 20: A flex sensor used during testing. This sensor outputs different voltage levels when bent. In this picture it was used to generate notes. Having a high bent made it generate high notes, and when not bent it generated low notes.
9 Summary

9.1 Performance

While using the prototype to generate MIDI messages with different controllers, there has been no substantial lag or slowdowns. The Arduino has enough memory and CPU power to deliver good performance. The limitation of the prototype and Arduino lies in the amount of analogue inputs (i.e. sensors) that can be connected. The Arduino UNO device used in this thesis can only have 6 analogue inputs. There are other versions of the Arduino that have more memory and more analogue inputs. The downside on the other chips with more inputs is that they are much bigger (some are almost double the size of the UNO chip). It has been thoroughly tested with many different sensors connected to it with different functionalities (for example having a setup of four sensors where one is generating notes, one changes the pitch, one changes the volume and the last one changes the octave).

The setup for the whole product can be quite big and cumbersome and this is something that could be done in a better way. For example, if the user tries to upload new settings to the prototype from the iPhone application, but the upload fails, both the prototype and the Hairless midi-serial need to be restarted manually. This should be changed so that it happens automatically.

Some sensors perform certain tasks better than other. The proximity sensor (distance sensor) is a good controller for tuning fine pitch or volume, but not for generating different notes. This is due to the fact that it is so sensitive when reading its surroundings that when it generates notes, it changes notes all the time.

The conclusion from a technical point of view is that the Arduino is perfectly capable of implementing a device like this. With the added bluetooth connectivity and the iPhone app, it has a potential to be a practically usable tool at the selected school, but the prototype would highly benefit from having some improvements implemented (those are explained in the next chapter).

9.2 Conclusion

This thesis aimed to answer the following questions:

- Is it possible to create a small MIDI controller that takes a general analogue control input and generates the appropriate MIDI signals for a software synthesizer?

- What are the limitations of the Arduino for this kind of application?

I think that it is possible to play and create music using this product. However, in order to conclude how well the product is suited for the creating music, it needs to be tested in future work. But if we look at the theories of Sergi Jorda and interpret his perspective on digital instruments we can come to some conclusions.

In terms of mid-diversity, an instrument can have a big variety of tones and pitches. This product can have both a low and a high mid-diversity depending on which sensors are used and what they are set to do. It is possible to have a setup that creates notes from just a potentiometer, and then the instrument would have a low mid-diversity. However, if you set a proximity sensor to control the pitch, a pressure sensor to control the notes and another sensor to control the octave, the user will have a wide range of notes and pitches, giving the instrument a high mid-diversity.

The macro-diversity defines how the instrument performs in different contexts. Since it uses the help of a music workstation, it can produce many different sounds using software synthesizers.
A problem lies in the way that tones are generated. When using certain sensors to play music, the tones are generated in a linear fashion. This means that one cannot go from a low tone to a high tone without generating the tones in between, when using these particular sensors. This makes the instrument less generic as it is limited by the way the tones are generated. Although one could use the activation button to generate tones when the activation button is pressed and hold down, to get away from the linearity, but that would lead to a choppy sound where the notes are turned on and off all the time. Therefore I think that this prototype has a low macro-diversity.

The product can definitely generate music, but it does this in it’s own way. It is not designed to be a professional instrument, but should rather be regarded as a way of creating music for the sake of trying something new, or as a way to easily create music or sounds with small movements. It does not require the player to be a virtuoso.

9.3 Future work and improvements

The limitations of the Arduino in terms of overall performance was not a problem. The limitations of the Arduino version used in this project (the Uno model) were the number of analogue inputs on the board and the size of the board. There are versions of the Arduino that has many more analogue inputs, but they are also bigger (e.g. Arduino Mega). There are smaller Arduino chips (e.g. Arduino Nano) which run on the same clock speed as the UNO chip and have more analogue inputs, but those have much less memory than the UNO. The more analogue inputs the chip needs to control, the more complex the system will become, thus it might require more memory and a higher clock speed. The Arduino Uno is capable of controlling the 6 analogue inputs with the amount of memory and clock speed it has. It might be interesting to see if a Nano chip will perform as good, and if it does, if it would be a better chip to use in this prototype.

Instead of using a laptop to generate the final sound, an idea could be to use a small single-board computer (like the raspberry pi) and to place it inside the box of the prototype. Instead of having to connect the product to a laptop, one would then only need to power up the product. This would lead to less steps in setting the product up and it would make it more portable (instead of having to use a laptop/stationary computer together with the product). It would require the single-board computer to be controlled from the phone application, or you would also need to connect a screen with mouse and keyboard.

Other possibilities would be to implement some game like feature in the iPhone app. The game could be very simple, maybe letting the player generate certain tones and then gain points if the tones are successfully generated. This could be used for different exercises, learning to play certain songs or just encouraging to use the product.

Testing how well it performs and how easy it is to use for the intended users would be very useful for the future of this product. Doing this might give new insights in how to build a system like this, and what should be changed in future versions of the product.

One other big improvement would be to develop an application for other platforms, like Android, Windows phones and even Linux and Microsoft Windows. This would make the product more accessible since more users would be able to use it.

It would also be very beneficial to be able to improve the bridge between the Arduino and the music workstation. In this thesis the application Hairless MIDI is used. If the bridge application was built only for this system, it could generate more useful debugging tools, less options in the GUI section and give more control of the whole system to the programmers. In this thesis, the Hairless MIDI program was altered in order to generate more useful messages for debugging. This could be changed further to improve the relation between the Ardulnsto and Hairless MIDI.
As briefly noted in the performance section, some sensors perform better than others. This could be fixed by adding a delta mean shift, meaning that every read from a sensor is the mean value of a specified number of previous reads and the current read. In this way, the reading from some very sensitive sensors may become more stable.

Some improvements can be done in the interface of the iOS application. In Figure 6, the inputs should be numbered from 1-6 instead of 0-5. Also, when adding an instrument that is changing fine pitch or volume, we need to specify which other controller generating notes it should modify. Instead of having to choose manually between the different numbered outputs, we would get a list of the current controllers that generate notes and then be able to choose one of these alternatives.
References


