Creating a Digital Musical Instrument for Children's Education and Expression

Joe Hathway

Music Technology Project Final Year Dissertation

BMus in Music Technology

Reid School of Music

University *of* Edinburgh

May 2021

Supervisor: Dr Michael Newton

Abstract

The aim for this project was to create an expressive and educational digital musical interface. Using research into other expressive digital instruments, a device was created that combined the use of audio, visual and haptic elements to provide the user with an expressive musical instrument enabled with additional non-sonic feedback. As the project progressed, there were changes made to the initially proposed hardware and software, which changed the scope if the device. The resulting device is an adaptable digital musical instrument that can be programmed to behave in a variety of ways.

Declaration

I do hereby declare that this dissertation was composed by myself and that the work described within is my own, except where explicitly stated otherwise.

Joe Hathway

May 2021

Acknowledgements

I would like to thank Dr Michael Newton and Matthew Hamilton for all their help over the course of this project. Your help and guidance has allowed me to create a project that I am proud of in an otherwise difficult year.

Contents

1 Introduction

The background for this project lies in research carried out into creating digital tools for use in education and digital instruments with expressive capabilities. The initial aim of the project – as laid out in its proposal – was to create an interactive musical controller that would aid the musical education and expression of young children, and so first research had to be carried out into existing products and expressive digital instrument design.

The most comparable existing product was one discovered early in the research stage: the Skoog. The Skoog was the product of a research project at the University of Edinburgh aimed at creating a musical instrument designed specifically for children with physical or learning disabilities [1]. The Skoog team achieved this by creating a tactile, touch-sensitive device that can be connected to the product app which in turn produces various different sounds. One area that the Skoog does not explore in its product, however, was providing a means of instant visual and haptic user feedback. The app receives data from the controller in order to produce sound; however, the controller does not provide any non-sonic feedback to indicate that a note has been successfully activated. Further research indicated that many other digital instruments do not possess this feature and so it was decided that it should be a key area to explore in this project.

Education and expressive performance were also important parts of the proposed project. It is easier to teach musical expression to a child than to an adult as research has shown that adults are less likely to unlearn bad performance habits than children [2]. This makes it important to introduce musical expression into a child's education from an early age. When it came to researching digital instruments that have been designed to aid musical expression, much material was available in the proceedings of past conferences of New Interfaces for Musical Expression (NIME). These proceedings contain many papers detailing the conception of various musical instruments that employ a wide variety of research and development techniques for creating expressive digital instruments. Some of these papers laid out recommendations for creating music technologies that are accessible for people of all ages and abilities [3-8]. A key aspect of most of these papers was to ensure anything being designed for widespread use should have accessibility in mind from the start [5-8]. Some of the intended users for this product may not possess full control over their physical or mental abilities, and so the controller should be designed to be accessible to as wide a range of abilities as possible. From the research stages of this project, a number of design factors were established that would outline how the project should be approached. These considerations were as follows:

- The device must be able to be used and operated by a child.
- It must remain accessible to those with impaired physical or mental capabilities.
- The device must allow for the ability to perform expressively.
- It should use visual and haptic elements to provide the user with instant feedback.

With these factors in mind, development was able to begin on creating an accessible and expressive digital musical instrument.

2 Hardware

2.1 Choosing the Hardware Platform

Upon embarking on this project, the first thing to decide was the matter of which hardware platform to use. The decision on which platform to use would be crucial in informing later decisions on both hardware and software. In the proposal for this project, it was suggested that the project may make use of an Arduino which would control patches created on Max MSP; however, upon further research another platform was found that appeared to be more suited to this project.

Bela is a platform that offers the ability for ultra-low latency sensors and audio [9]. While the Arduino utilises a microcontroller, Bela uses a powerful BeagleBone Black microprocessor, which works as a small computer capable of processing audio. The Bela also has audio output and even holds a small onboard audio amplifier. This allowed for the possibility of creating a stand-alone musical instrument, rather than the initially proposed controller that would have required connection to a computer fitted with Max MSP software. Bela also boasts much lower latency than Arduino [9-10], which would be crucial in creating a playable digital instrument and providing instant user feedback.

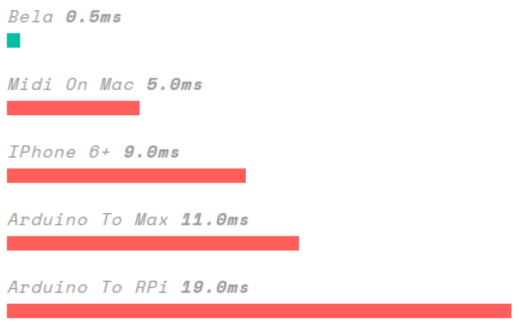


Figure 2.1: Action-to-sound latency across platforms. [source: Bela.io]

For these reasons, it was decided to go forward with this project using the Bela as its main platform.

This would change the original project scope, with the work going forward aiming to creating a digital musical instrument with onboard processing and audio. However, these options still met the original criteria laid out in this report’s Introduction, simply with the emphasis shifting from creating an interactive controller to creating a stand-alone device.

2.2 Input Device

2.2.1 Considerations

Arguably the most important part of designing an expressive interface is choosing how the user can interact with the device. Almost all musical interfaces act according to a cause-and-effect relationship between physical gesture and sound. When playing conventional musical instruments, some movements are used to initiate vibration – like the pluck of a string – while other, usually smaller, movements alter the vibration itself, such as vibrato, slides, bends etcetera [11, p. 46]. For a digital interface it can be difficult to get the same degree of control over these smaller, expressive variations in the sound; on acoustic instruments, the gestures used by musicians to alter the sound depend entirely on the physics of the instrument being played [12, p. 126]. This is an issue that should be considered when designing an expressive interface, as choosing the device's input device will define how it will be interacted with. Since the project was no longer simply sending MIDI messages to a computer to generate sound, the possibility of an interface that could control the sound directly, would give the device greater freedom when it came to mapping parameters [12, p. 126].

2.2.2 Possible Input Devices

To give the user more expressive control there had to be multiple degrees of control to the device. This could be a pressure sensitivity surface or velocity of a button push. Initially, the concept for this project involved using a pressure sensitive material to create capacitive sensing shapes. Research was carried out into the production of pressure sensor matrices using a common packaging material [13]. This option was very low cost and an entirely customisable method of creative capacitive sensors, as they would have had to be created from scratch. However, making a capacitive material from scratch would have been very time-consuming.

Another available, low-cost option was the Trill sensor, produced by the creators of Bela. Trill sensors are a range of capacitive touch sensors that come in various shapes. The Trill square provides four degrees of control per sensor - x-position, y-position, touch-size and number of touches - which would make them ideal for mapping to different controls. Trill sensors use the I2C communication protocol that provides a method of sending and receiving data from the host to the sensors [14]. This feature allows the use of multiple Trill sensors, providing each sensor has a unique I2C address [14-15].

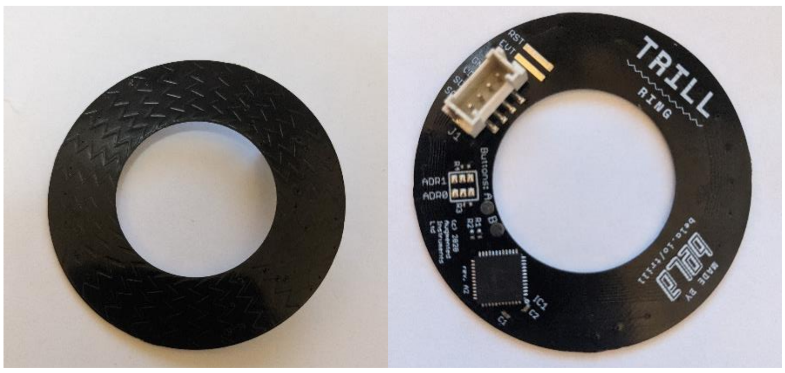


Figure 2.2.2: Front and back of a Bela Trill Ring

Due to the convenience of having a ready-made capacitive sensor and the possibilities it allowed for mapping, it was decided to go forward using Trill sensors as its primary input device.

2.2.3 Implementation

Connecting a single sensor required connecting the SCL, SDL, ground and voltage outputs to corresponding pins on an Arduino or Bela. However, connecting multiple Trill sensors to one host device required a little more work as each sensor needed a unique I2C address to be operated independently from each other. This was achieved by simply soldering a connection between two solder pads on the back of each sensor to manually change the address.

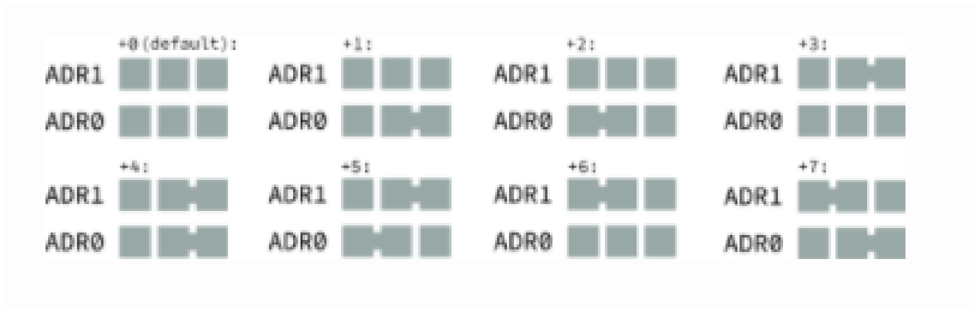


Figure 2.2.3 A diagram showing how to solder the pads on the back of a Trill to change its I2C address. [source, Bela.io]

2.3 Output Devices

2.3.1 Audio and Visuals

While selecting output devices, it was important to consider the parameters that would be most useful for the user to receive feedback on. As a musical instrument, it would obviously have to output audio, and using Bela, this was very simple. The audio output on the Bela can be driven using an onboard amplifier and just required connecting a small speaker to the audio output pins. However, when choosing other output devices, additional aspects of playing were necessary to consider.

Visual feedback can develop the interactive processes between a user and the instrument, as well as helping the audience to understand how the performers play their instrument [12, p. 129]. Using eye-catching visuals would also help the device become more engaging, especially for children. For this visual component, it was decided that a system of coloured LEDs would suffice in order to provide visual feedback. Once connected to the Bela’s analog output pins, they could be coded to give feedback on parameters such as pitch and envelope.

2.3.2 Haptics

Using Trill sensors as the input device presented an issue: using a smooth 2D surface to control the device removed the ability for instant haptic feedback. A difficulty would arise for the user with regards to the position of their finger on the sensor, especially if it is not in full view or multiple sensors are being used simultaneously. It was decided that a haptic component should be added to the device to provide this missing feedback.

For the haptic component, it was decided to make use of vibration motors attached to the Trill sensor to give user feedback on position. Initially, a DC vibration motor was connected to an Arduino, and its intensity controlled using the touch-size parameter of the Trill. Unfortunately, the DC motor was not able to power on/off fast enough to provide instant response. It was thus decided to make use of Linear Resonant Actuator (LRA) vibration motor, a much more responsive motor.

LRA motors employ an electromagnet connected to a spring that provides vibration across a linear axis [16]. This allows for an instant start and stop at high speeds and can even simulate a button-like 'click'. LRAs usually require external drivers to control them as they can only operate at a specific AC frequency. The driver chosen [17], automatically detected the resonant frequency of the LRA and could communicate with a host device using I2C protocol.

Once connected to an Arduino, the LRA motors provided a much faster haptic response. Connecting multiple LRA drivers to the same host device required using a I2C multiplexer to provide each driver with a unique address.

2.4 Connecting Hardware Components

A make-shift circuit board was created on a sheet of perforated board to secure and connect the internal components. Male header pins were soldered to the board to allow it to be attached and removed from the Bela. All of the wired connections to the board were made using male/female crimp connectors and were not soldered to allow for easy access and assembly.

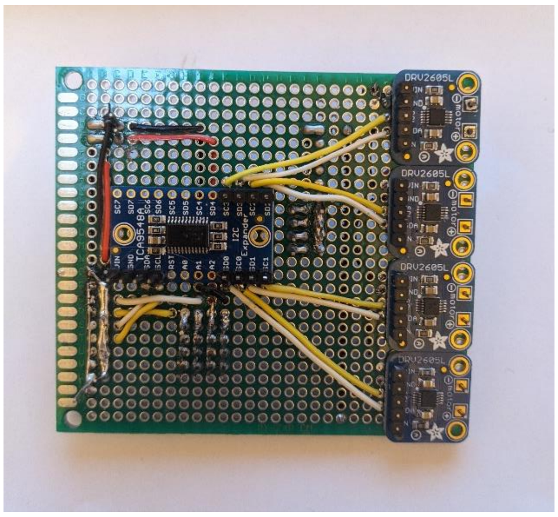


Figure 2.4: Top view of the circuit board which contains the haptic drivers (right side) and I2C Multiplexer (centre).

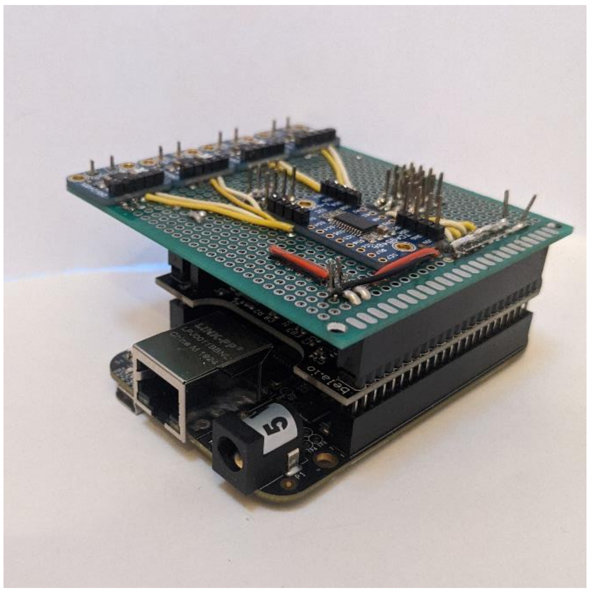


Figure 2.5: The circuit board connects to the Bela using male header pins soldered to underside and connects to the other components via the male pins on its top.

3 Software

3.1 Coding Using Bela

The Bela had the ability to be programmed using patches in Pure Data (Pd). Bela uses a Linux library (libPd) to run the DSP portion of Pd. This proved invaluable, as Pd offered a fast and reliable way to generate code for interactive audio. Using Pd allowed for a very visual workflow, and a gradual learning curve. Once generated, a patch can then be uploaded onto the Bela using the Bela IDE. Patches for controlling the Bela could be generated far more quickly this way than coding in C++.

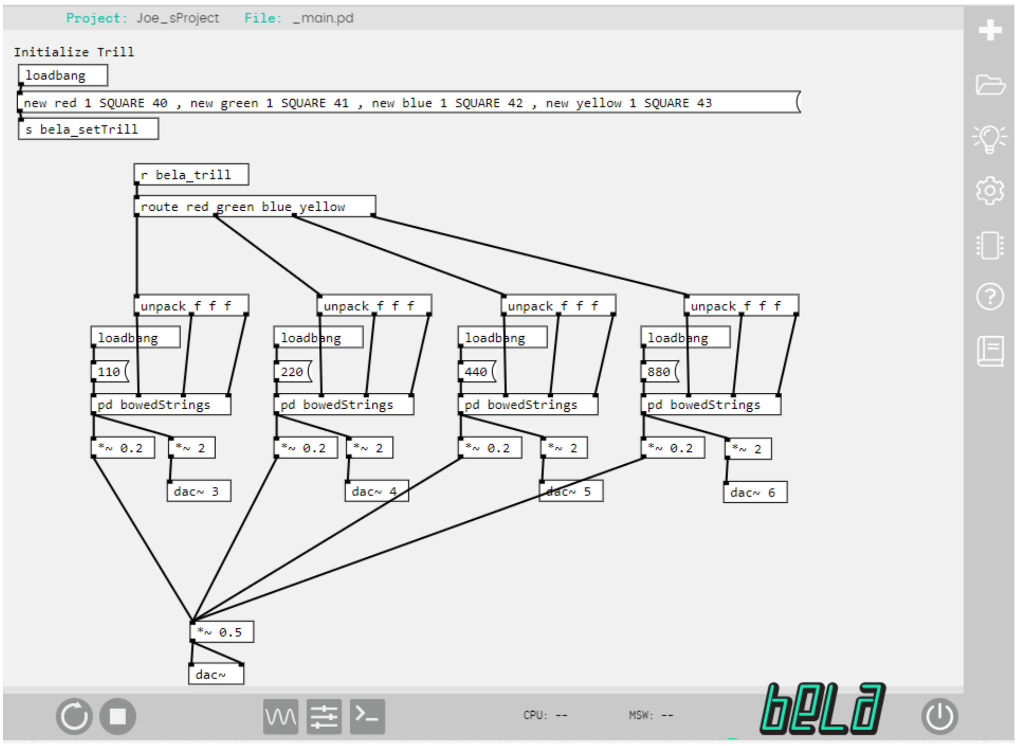
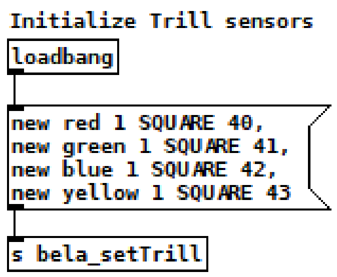


Figure 3.1: The Bela IDE.

3.2 Setting Up Input/Output Devices

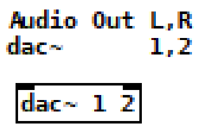
The Trill sensors were initialised upon loading the Pd patch using a message sent to the [s bela_setTrill] object. This message included a sensor ID and its unique address. By giving different sensors unique ID and addresses, they were able to be initialised, and read from, anywhere in the code. Reading from the sensors required using a [r bela_trill] object which could be routed using the sensor ID. Using the Trill Square, this outputted a stream of four numerical values that could be mapped to different parameters in the code: number of touches (integer value), x-position, y-position and touch-size (all floating-point values between 0 and 1).



```
Initialize Trill sensors
[loadbang]
[new red 1 SQUARE 40,
new green 1 SQUARE 41,
new blue 1 SQUARE 42,
new yellow 1 SQUARE 43(
[s bela_setTrill]
```

Figure 3.2: Multiple sensors can be initialised using the same message as long as there is a comma between each new initialisation.

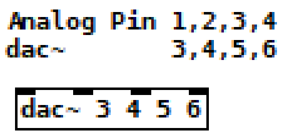
Bela's analog outputs could be used to drive audio and LED brightness. Driving the LEDs using an analog output allowed for easier control of brightness, as doing this in the code simply required sending a floating-point value between 0-1 to the one of the analog outputs on the Bela, using the [dac~] object. Analog outputs 1 and 2 corresponded to the Bela's left and right audio output; 3-10 controlled analog pins used for (in this instance) LEDs.



```
Audio Out L,R
dac~      1,2

[dac~ 1 2]
```

Figure 3.3: The Audio signal in Pd can be sent to the Bela's speakers using dac~ 1 and 2.

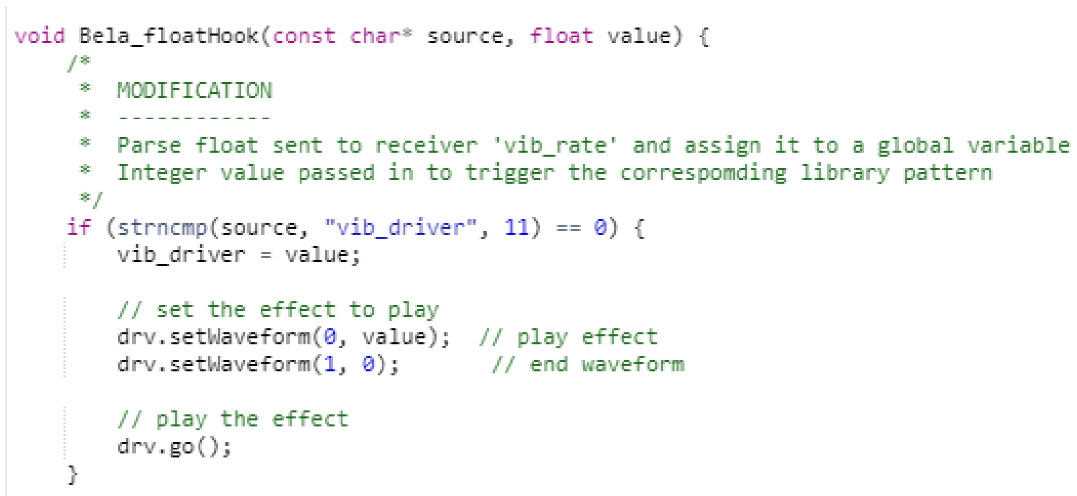


```
Analog Pin 1,2,3,4
dac~       3,4,5,6

[dac~ 3 4 5 6]
```

Figure 3.4: Dac~ 3-10 in Pd correspond to analog pins 1-7 on the Bela.

Calling upon the haptic drivers required a little more code, as they made use of an external library of vibration patterns downloaded from the manufacturer's website [17]. An object had to be created in order to communicate with the library from Pd. This could be done by coding a custom render.cpp file, which modified the Bela's default libPd wrapper template to combine C++ code with patches in Pd [18]. A receiver object named "vib_driver" was created, which could trigger the vibration patterns contained within the library.



```
void Bela_floatHook(const char* source, float value) {
    /*
     *  MODIFICATION
     *  ------------
     *  Parse float sent to receiver 'vib_rate' and assign it to a global variable
     *  Integer value passed in to trigger the correspomding library pattern
     */
    if (strncmp(source, "vib_driver", 11) == 0) {
        vib_driver = value;

        // set the effect to play
        drv.setWaveform(0, value);  // play effect
        drv.setWaveform(1, 0);       // end waveform

        // play the effect
        drv.go();
    }
```

Figure 3.5: This section of the render.cpp file receives a float value as input and uses it to initiate the vibration of the LRA motors using the vibration pattern library.

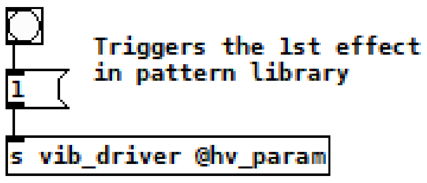


Figure 3.6: This is the created object that can be called upon in Pd to initiate a vibration from the pattern library.

Once the input and output devices were connected and ready to be called upon in Pd, they could be programmed to perform in a variety of ways.

3.3 Programming for Gesture Control

Musical gesture can be described as the conceptual mapping of one domain experience (change in sound) to another domain experience (physical movement) [19, p. 83]. For example, an increase in pitch or dynamics is often described as rising, and the decrease as falling. In order to design an interface that can map physical gesture to musical sound, these concepts should be kept in mind. Studies have shown that people respond to these kinds of musical gestures from infancy [20], therefore, for a device such as this, it was important a more aggressive movement should correspond to a rise in intensity of the device's audio and visual elements. Additionally, it was important to map the parameters of the Trill sensor in a way that would be intuitive for a child to operate.

3.3.1 Direct Mapping of Parameters

Arguably the simplest method of coding gesture control using Trill sensors was directly mapping the parameters of the Trill to an aspect of the sound - pitch, volume, timbre. The Trill outputted values for touch across x-position, y-position as well as a value for the touch-size (the surface area of the user's finger on the sensor). Each of these could be used as a separate degree of control over the sound.

One application of direct mapping is a form of subtracted synthesis. This was achieved by having a spectrally rich tone playing continuously upon touching the sensor's surface. By mapping the y-position to the volume control and the x-position to the cut-off frequency of a low-pass filter, the dynamic and timbral aspect of the sound were controlled entirely by the user. Further still, a detune parameter was connected to the touch-size value, allowing the user to generate a controllable vibrato.

3.3.2 Creating Trigger Points

Another way of programming gesture control with Trill involved create trigger points across its surface. This technique could be used to simulate the 'pluck' of a guitar string. This was achieved using the [moses] object in Pd. This splits the surface into multiple sections and moving from one section into another would produce a trigger. During development, these trigger points were used to trigger a Karplus-Strong synthesis sub-patch that generated a string sound, however, it could as easily be used to trigger a change in pitch or play a loaded sample.

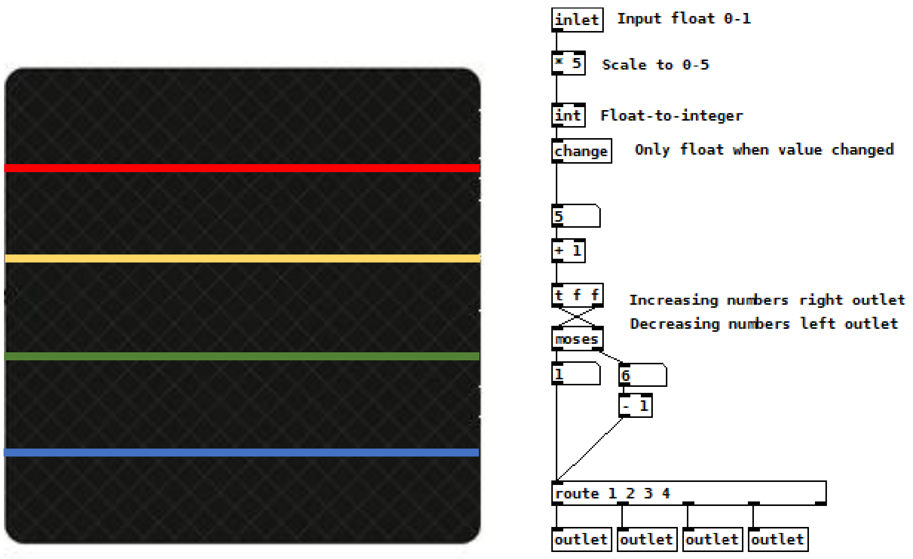


Figure 3.3.2: The diagram on the left indicates the position of the trigger points created by the code on the right.

A few improvements were made to this basic idea. Firstly, setting up multiple trigger points across a single Trill surface allowed them to be 'strummed', creating guitar-like chords. The x-axis was also mapped to values like volume and filter cut-off to make the string louder and brighter when 'plucked' in the middle. This simulation was further improved by adding a short 'click' from a connected vibration motor, which aided the user in locating the position of the trigger points through haptic feedback.

3.3.3 Using Speed as a Control

A main focus when working on the software for this device was finding a way to measure the speed of a stroke across the Trill surface. A sub-patch was found in the Pd forum for calculating this, called [pd difference] [21]. This stored the previous position values and calculated their difference to give a value for acceleration. If the result raises above or falls below zero, the direction of acceleration could also be measured and used as a parameter.

There were multiple applications of this for a device of this type. The sub-patch could be used as a speed limiter, preventing jumping too quickly between two values. Another example that was used during development was the creation of a bow-like gesture control. By mapping speed to volume control the user was able to control the dynamics of a sound by increasing and decreasing the speed at which their finger moved across the surface of the sensor. A continuous bowed string-like sound could be generated by continually and evenly circling the finger around the Trill surface.

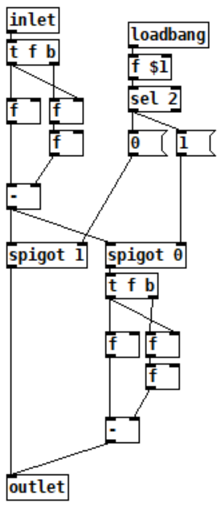


Figure 3.3.3: The [pd difference] sub-patch used to measure the speed of change between each input.

4 Fabrication

4.1 Design Considerations

Once the hardware and software components of the project had been finalised, the last step of prototyping was to fabricate a casing to house the components. Whilst designing a casing, the following considerations had to be taken into account:

- The casing must be large enough to house all of the hardware components yet small enough to be operated by a child.
- It must clearly display the LED components for continuous visual feedback.
- The layout of the Trill sensors must make their use intuitive and allow for a wide range of motion whilst operating them.
- It must not be hazardous to children, i.e., no small parts or sharp edges.

At this prototyping stage, the casing should also be easy to assemble/disassemble in order to have continued access to the internal components in case of the need for internal adjustments. For similar reasons, the Bela's USB input also had to be accessible in the case of any programming adjustments.

4.2 Research and Design

The first step of the design process was to research toys and technologies that have been designed for children. A mood board was created with pictures of existing products in order to see how other designers had tackled the concerns listed above. Sketches were made of some design ideas and these ideas were then broken down into pros and cons and narrowed down to a final design idea. After these stages, it was decided that a simple cube shaped casing would be suitable to house the components whilst being simple enough to prototype using accessible facilities.

Following this stage, a plan for designing the prototype was devised. For ease of assembly, the design would be broken down into an external skeleton and five removable panels, each holding a Trill sensors or speaker. The panels would be cut from an opaque material allowing the light from the LEDs to be evenly dispersed. A 3D model of the skeleton would then be drawn using CAD software and 3D printed. By printing each side individually and securing them using screws, the panels could be slid into place and were not permanently secured, allowing easy assembly and access to internal components.

Drawing a model of the skeleton for printing required accurately measuring the position of the Bela's inputs and screw holes and drawing them in Fusion 360. This way, a mounting could be created as well as a space for the USB cable. Using this mounting as a starting point, the skeleton was then modelled and cut into six printable sides.

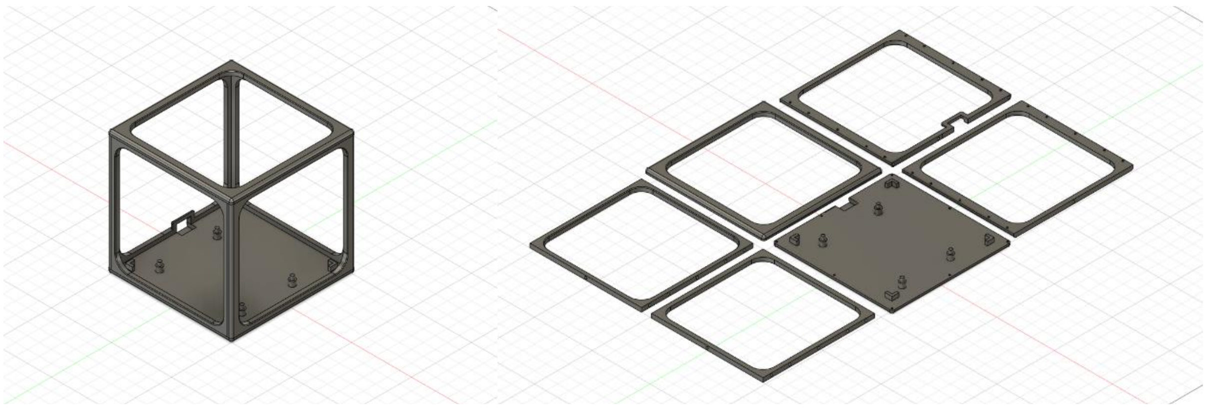


Figure 4.2: A 3D model of the devices casing drawn in Fusion 360. It can be assembled and disassembled using locating pins and holes at each connection point.

4.3 Manufacture and Assembly

A click-and-collect service had been put in place for 3D printing using University facilities. This required importing the 3D drawings as .STL files from into Ultimaker Cura to calculate the print time before submitting the files and print times on an online form.

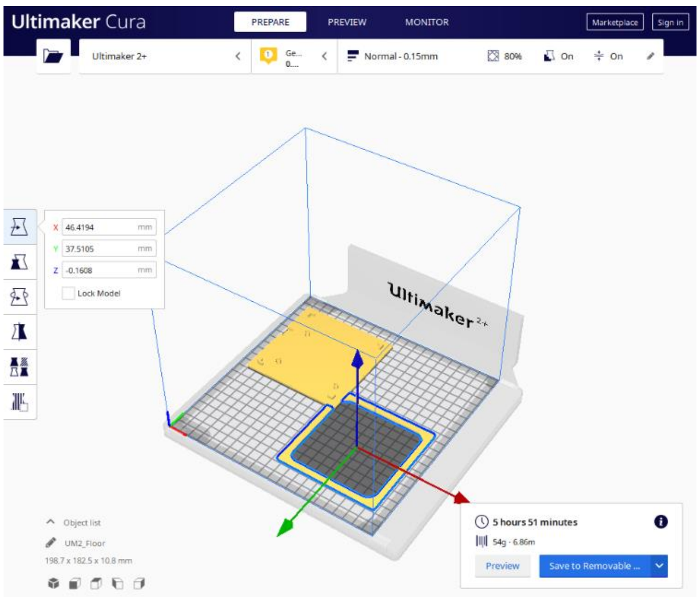


Figure 4.3: Each component could have its print time calculated using the Ultimaker Cura software.

Once all the parts had been printed and collected, the parts' edges were cleaned up using a Stanley knife and assembled with the four sided glued together and the top and base drilled and tapped for threaded screws. Upon assembly, some of the printed parts did not fit together due to the tolerance from the 3D printer being larger than anticipated. The parts had to be reprinted with this tolerance in mind.

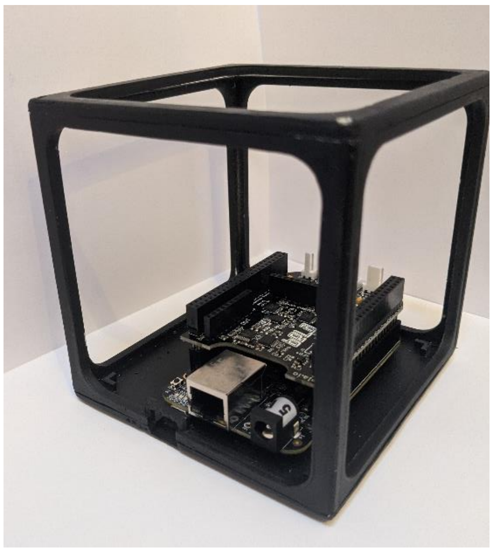


Figure 4.4: The second print of the skeleton casing, housing the Bela board.

Each of the panels that held the Trill sensors were cut to size and a hole made in the middle to feed through the connecting cables and attach the LRA motors to the back of the Trill. A sheet of coloured vinyl was applied to each of the sensors for the user to differentiate more easily between each side. The sensors – along with a string of corresponding, coloured straight LEDs – were secured to each panel using hot glue. A more elegant solution for securing the sensors and LEDs will need to be devised in further prototypes; however, for the purposes of securing the components for demonstration, hot gluing was an adequate, non-permanent method.

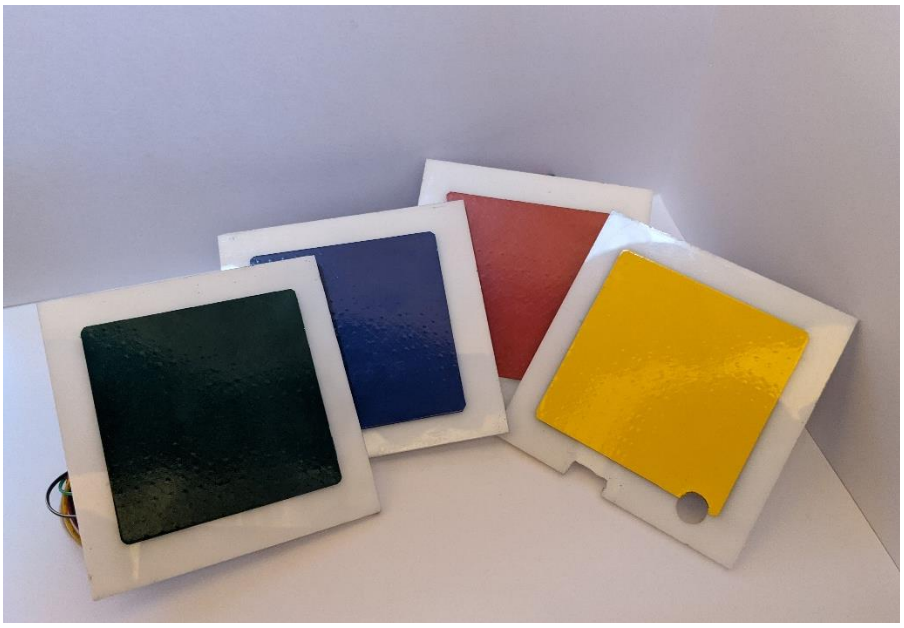


Figure 4.5: Each Trill sensor was mounted onto a panel of opaque plastic. A hole had to be drilled into the panel with the yellow Trill to allow the Bela to receive a 5V power cable.

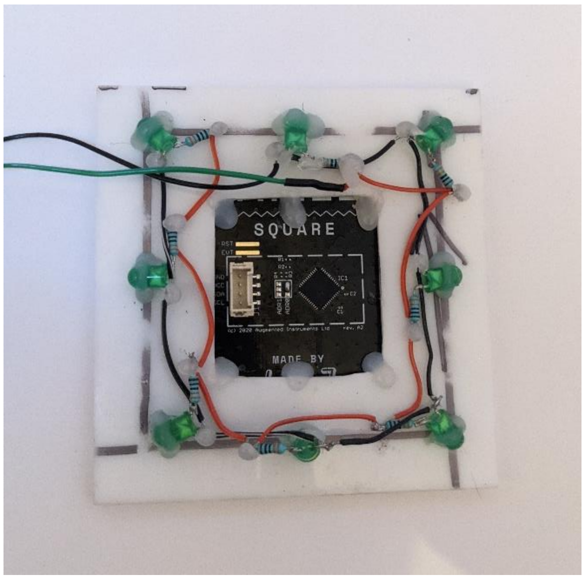


Figure 4.6: This photograph shows the back of the panel holding the green Trill. The string of LEDs was also hot glued to the back and the LRA vibration motor attached to the back of the Trill.

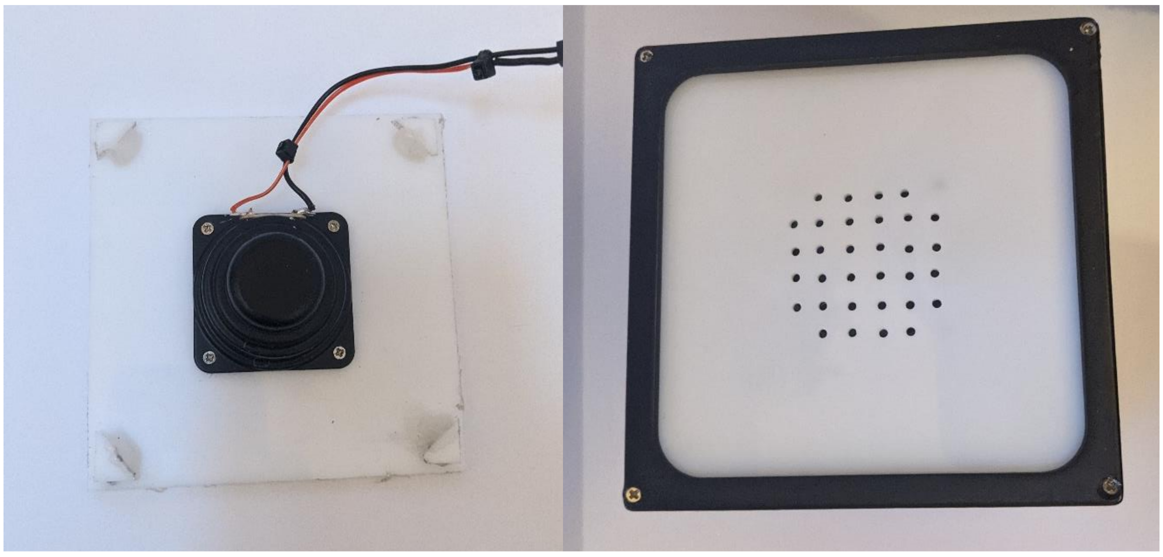


Figure 4.7: The top panel held the small 3W speaker that would output the device's audio. Four small triangular pieces of plastic were glued to the corners to the panel could be centred properly without it being permanently secured to the skeleton.

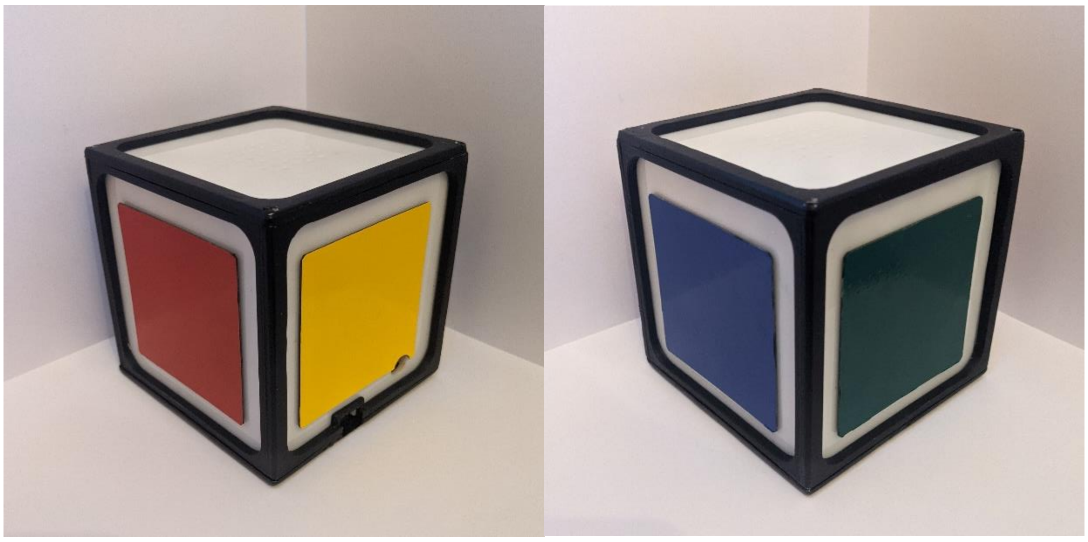


Figure 4.8: A view of both sides of the completed device housing.

5 Results and Areas for Further Development

As the device was developed and built, it became clear that it could be adapted for numerous applications, depending on how it is programmed to operate. With the ability to generate sound and call upon visual and haptic elements anywhere in the code, the device became very customisable in its functionality. This final chapter will explore some of the uses the device may have in real-world applications, as well as suggesting next steps for its development.

The created device is of an adaptable nature, and as such can be implemented in a variety of educational settings. For younger children (between 3-6 years old), it can be used to explore the connection between gesture and sound. This could be especially useful for children of this age who do not yet possess the physical strength to play acoustic instruments, or those with disabilities that prevent them from having a wide range of motion. As result, the device can be used to explore musical concepts such as melodic, rhythmic and timbral control from an early age.

The visuals and haptics elements can also be mapped to any parameter, meaning it can (as research and development continues) be adapted to be more engaging for children and useful for education. An important part of research that could not be carried out due to current lockdown restrictions was user testing and feedback. By holding focus groups with the intended user demographic, more information can be gathered on the effectiveness of the device as an educational tool. Through a cycle of testing, feedback, analysis and adjustment, this would allow the product to become more tailored to this purpose.

For children of primary school age (roughly 5-11), the device becomes a useful tool for exploring musical expression, particularly through the audio and visual elements. As discussed in section 3.3, it can be programmed to be performed in different ways, even simulating conventional instruments. Children old enough to have more control over their gestures can use this device in a similar way to acoustic instruments. In order to prepare the device for use as an expressive musical instrument, more work needs to be carried out into creating sound generated patches in Pure Data, as the focus of this project so far has been in developing the interface and control aspects.

Although the device has been designed with the musical expression of children in mind, its nature as an adaptable digital instrument could see uses in other practical applications. Its combination of audio, visual and haptic elements could find use in work involving children with sensory issues. Further research and development could lead to adapting the device into a tool that enables audio, visuals and haptics, in order to help those with sensory issues to explore sound, light and touch in a controlled way. Any further research into this area would require contacting developmental psychologists, teachers, and other experts in this field to gain a greater understanding and develop the device to this purpose.

Bibliography

[1] "About - SkoogMusic", SkoogMusic.com, 2021. [Online]. Available: https://skoogmusic.com/about-2/#1470320958783-5694e999-201d. [Accessed: 06- Apr- 2021].

[2] B. Brenner and K. Strand, "A Case Study of Teaching Musical Expression to Young Performers", Journal of Research in Music Education, vol. 61, no. 1, pp. 80-96, 2013.

[3] A. Asha, L. Woodbury and T. Davis, "Design Considerations for Instruments for Users with Complex Needs in SEN Settings", in Conf. NIME, Copenhagen, Denmark, 2017, pp. 216-221.

[4] M. Grierson and C. Kiefer, "NoiseBear: A Malleable Wireless Controller Designed In Participation with Disabled Children", in Conf. NIME, Daejeon, Republic of Korea, 2013, pp. 413-416.

[5] C. Trappe, "Making Sound Synthesis Accessible for Children", in Conf. NIME, Ann Arbor, Michigan, USA, 2012.

[6] J. Wright and J. Dooley, "On the Inclusivity of Constraint: Creative Appropriation in Instruments for Neurodiverse Children and Young People", in Conf. NIME, Porto Alegre, Brazil, 2019, pp. 162-167.

[7] A. Skuse, S. Knotts. “Creating an Online Ensemble for Home Based Disabled Musicians: why disabled people must be at the heart of developing technology.,” in Conf. NIME, Birmingham, United Kingdom, 2020, pp 115-120.

[8] Q. Jarvis-Holland, C. Cortez and F. Botello, "EXPANDING ACCESS TO MUSIC TECHNOLOGY Rapid Prototyping Accessible Instrument Solutions F or Musicians With Intellectual Disabilities", in Conf. NIME, Birmingham, United Kingdom, 2020, pp 149-153.

[9] Bela.io. (n.d). What is Bela? [Online]. https://bela.io/about.html

[10] A. McPherson, R. Jack, G. Moro. “Action-Sound Latency: Are Our Tools Fast Enough?”, in Conf. NIME, Brisbane, Australia, 2016, pp 20-25.

[11] W. Windsor, "Gestures in Music-making: Action, Information and Perception", in New Perspectives on Music and Gesture, E. King, A. Gritten and G. Welch, Ed. Abingdon, United Kingdom: Taylor & Francis Group, 2011, pp. 45-66.

[12] D. Arfib, J. Couturier and L. Kessous, "Expressiveness and digital musical instrument design", Journal of New Music Research, vol. 34, no. 1, pp. 125-136, 2005. [Accessed 5 May 2021].

[13] S. Suprapto, A. Setiawan, H. Zakaria and W. Adiprawita, "Low-Cost Pressure Sensor Matrix Using Velostat", in International Conference on Instrumentation, Bandung, Indonesia, 2017.

[14] Bela.io. (n.d). About Trill [Online]. https://learn.bela.io/products/trill/about-trill/ [Accessed: 24- Nov- 2020].

[15] Bela.io. (n.d). All About I2C [Online]. https://learn.bela.io/products/trill/all-about-i2c/ [Accessed: 24- Nov- 2020].

[16] "AB-020 : Understanding Linear Resonant Actuator Characteristics - Precision Microdrives", Precisionmicrodrives.com. [Online]. Available: https://www.precisionmicrodrives.com/content/ab-020-understanding-linear-resonant-actuator-characteristics/. [Accessed: 17- Dec- 2020].

[17] T. DiCola and K. Rembor, "Adafruit DRV2605L Haptic Controller Breakout", Adafruit.com, 2014. [Online]. Available: https://learn.adafruit.com/adafruit-drv2605-haptic-controller-breakout. [Accessed: 10- Feb- 2021].

[18] Bela.io. (n.d). Custom Render [Online]. https://learn.bela.io/tutorials/pure-data/advanced/custom-render/ [Accessed: 10- Mar- 2021].

[19] L. Zbikowski, "Musical Gesture and Musical Grammar: A Cognitive Approach", in New Perspectives on Music and Gesture, E. King, A. Gritten and G. Welch, Ed. Abingdon, United Kingdom: Taylor & Francis Group, 2011, pp. 83-98.

[20] C. Trevarthen, J. Delafield-Butt, B. Schogler. "Psychobiology of Music Gesture: Innate Rhythm, Harmony and Melody in Movements of Narration", in New Perspectives on Music and Gesture, E. King, A. Gritten and G. Welch, Ed. Abingdon, United Kingdom: Taylor & Francis Group, 2011, pp. 11-43.

[21] "Trigger Bang from change in sign of acceleration", PdPatchRepo.info, 2016. [Online]. Available: https://forum.pdpatchrepo.info/topic/9985/trigger-bang-from-change-in-sign-of-acceleration/4. [Accessed: 29- Jan- 2021].