INTERACTIVE MUSIC SYSTEMS USING NINTENDO SWITCH JOY-CONSS WITH MAX/MSP

|  |  |
| --- | --- |
|  | Sonia Shuk-Yin Ng |
|  | University of Victoriashukng@uvic.ca |

ABSTRACT

This project targets to develop an affordable and feasible alternative approach in generative music using gestural motion detecting devices readily available in the consumer market. The Nintendo Switch Joy-Cons will be the main analytical device used to obtain human gestures and analyzed through MAX to create an interactive music performance. A granular synthesizer is used to as the main sound generation unit for the project. Both manual changes to the synthesizer and automatic changes using a neural network are implemented in the creation of the synthesizer. By having the Joy-Cons be used to control a synthesizer created inside MAX, this will test the feasibility of a an easily accessible alternative approach to motion and gesture sensing for the use of performance. Further development to this project could lead to a wider audience of people being able to intuitively create and manipulate music with their movements.

1. INTRODUCTION

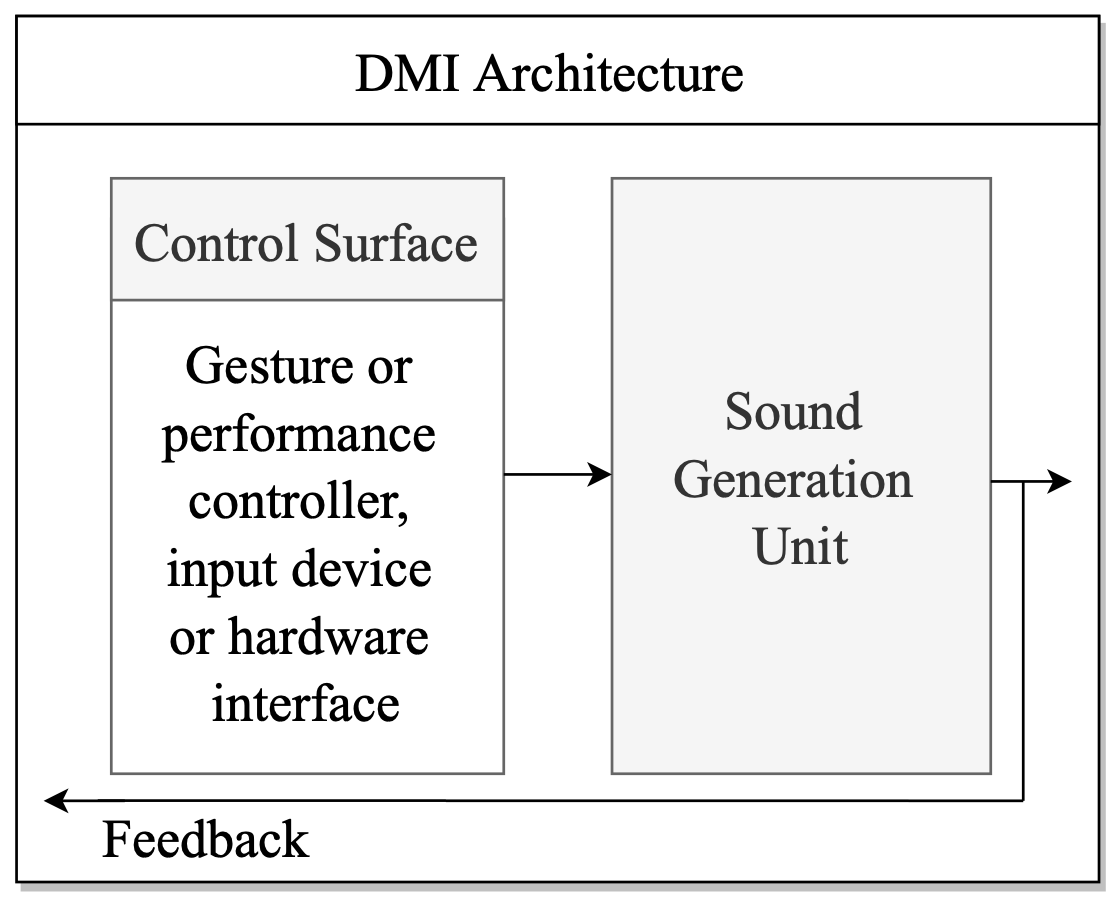
The rise of advanced devices capable of translating movement and gestures into computer data offers exciting potential for interactive music performances. There are several devices and technology currently available in the marketplace specifically designed for motion capturing [1]. However, the technology is often expensive and not accessible to the public. This created the alternative approach of musicians designing their own data sensors using devices not originally music but hacked to become musical interfaces. One of these devices being the Nintendo Switch Joy-Cons.

The primary goal for this project is to successfully obtain data from gestural movements of a user by having Nintendo Switch Joy-Con maneuvered by the user. The game controllers will be connected to the computer then, using the program MAX[[1]](#footnote-1), the received data becomes the dataset for music.

1. DESIGN

The design of this project will follow the process of creating a digital musical instrument (DMI), devices that are to be controlled for digital sound synthesis [2]. DMIs generate sound through a two-step process that begins with some form of input from the human performer, typically an action or gestural movement, that is executed on the instrument’s control element [3]. In this case, a ‘wearable’ controller attached to the performer’s hands.

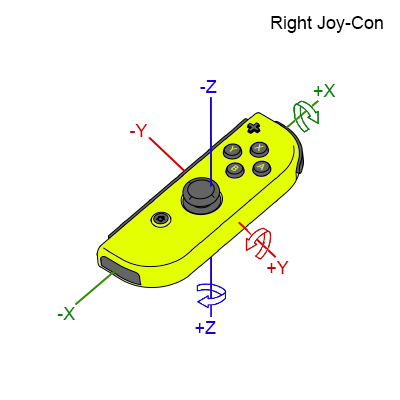
Input gestures would be linked to their respective output parameters. The audio output will be pre-coded in an audio programming environment on a laptop computer (e.g. MAX/MSP) and will be triggered by the gestural inputs.



**Figure 1.** Architecture of a typical DMI system [4].

* 1. **The Nintendo Switch Joy-Cons**

The Joy-Cons provides 6 degrees of freedom: 3 linear translation directions in X, Y, Z positions and 3 rotation angles, roll, pitch, and yaw. In addition, the controller also contains multiple buttons which can also be taken into consideration. By extracting these data inputs, this provides the basis for generating sound.



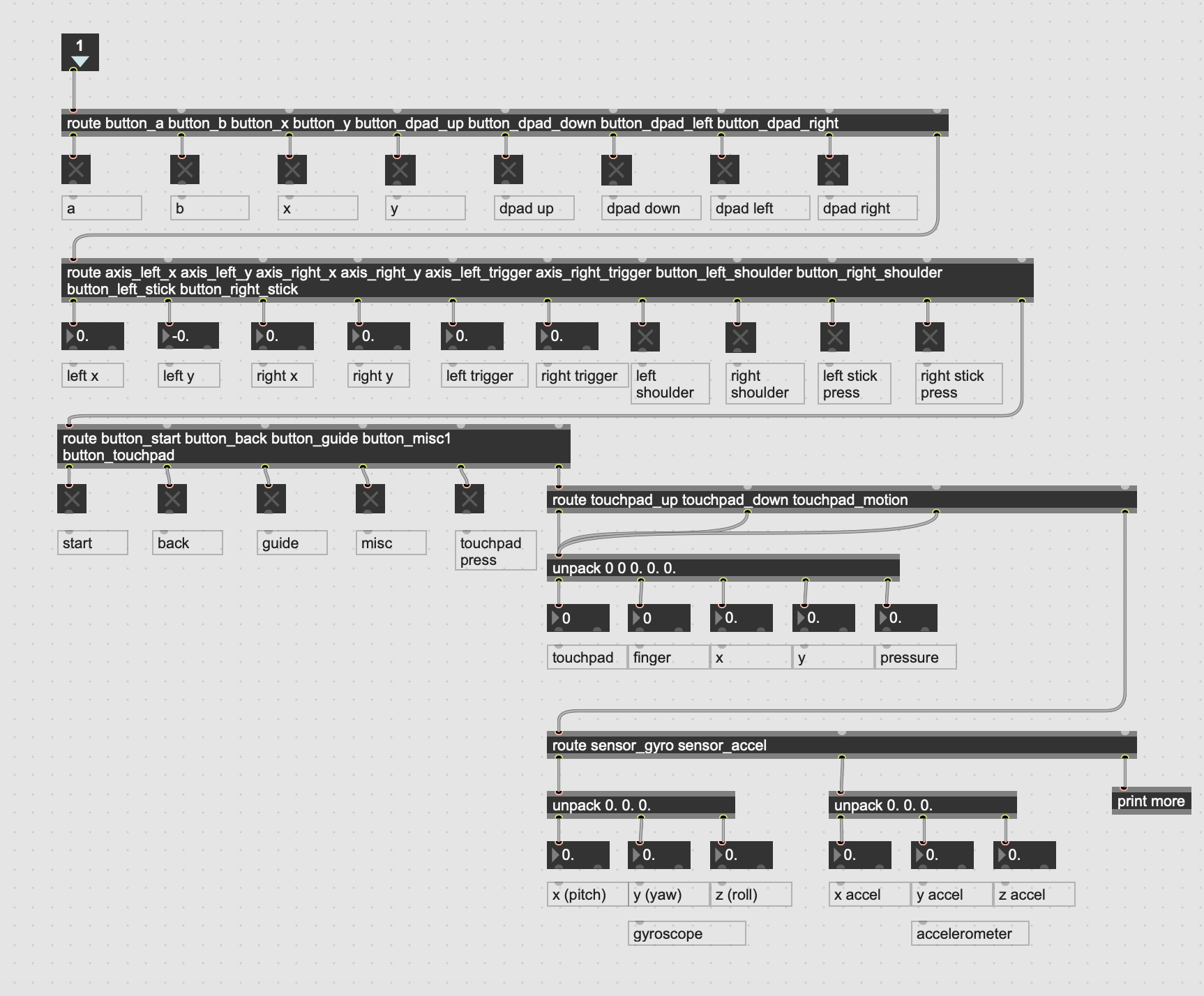
**Figure 2.** Accelerator and gyroscope axis for the right Joy-Con. For the left Joy-Con, mapping will be similar expect reversed. [5].

1. Implementation

Using a game controller with MAX is not a new subject. There have been several projects in the past which highlights the flexibility of integrating game controllers with MAX to create generative music [6] [7]. With recent versions of MAX have created more efficient ways in in integrating the controllers to its interface.

* 1. **MAX Objects for Using Game Controllers**

Introduced in MAX 8.6 is the [gamepad] object[[2]](#footnote-2). This object provides easier implementation for connecting gaming controllers to MAX. [gamepad] allows users to track the outputs from the button, joystick, trigger, and sensor events. By connecting the Nintendo Switch Joy-Cons via Bluetooth to the computer, [gamepad] should be able to detect the connection and parse out data to the MAX interface.



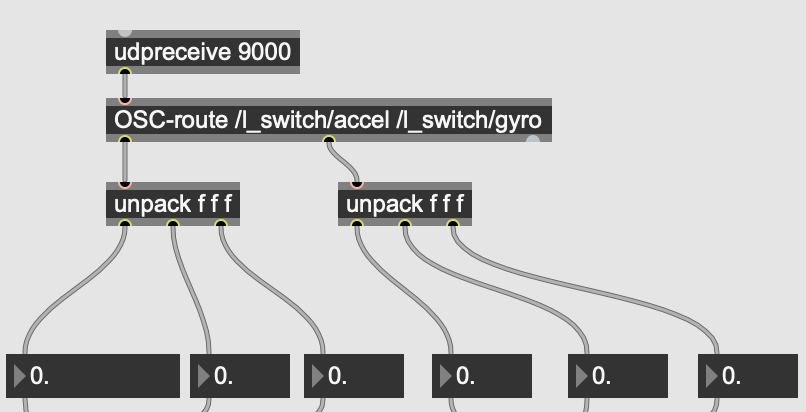
**Figure 3.** Possible data retrieval available from the gamepad.maxhelp patch for [gamepad].

While easily functional, the [gamepad] does not adapt well for more than one controller. The object outputs the accelerometer and gyroscope data into the same list. Using this object would not be a viable choice if independent data from both Joy-Cons is desired.

The previous used object from the community before this newer version of MAX is [hi][[3]](#footnote-3). The [hi] object functions similar to [gamepad], The [hi] object provides mapping for human interface peripherals (i.e. trackpad, keyboard, gaming devices, and others). The downside to using [hi] is the lack of processing accelerometer and gyroscope data. A workaround for this would be using a software to detect the movements outside of the MAX interface and feeding the received information to [hi] in MAX [8] [9].

* 1. **Python to MAX using OSC**

Simply using objects from MAX to receive data from the Joy-Con is not enough. A more hands-on approach is routing the data from the Joy-Cons to MAX yourself using available Python libraries, joycon-python [10] and python-osc [11]. Data will be transmitted via OpenSoundControl (OSC), a stable, 32-bit protocol used for interconnecting hardware controller devices to the computer, as well as software on one or more computers using local networks [12]. The routing of data from Joy-Cons to the MAX program will use python-osc then, with the CNMAT External[[4]](#footnote-4) [OSC-route] object, data should be received.



**Figure 4.** using the [OSC-route] object to dispatch OSC messages.

1. Generating sounds

Actual sound will be generated by a granular synthesizer using movements to control the parameters of the synthesizer. For performance potential, regression will be implemented.

* 1. **Granular Synthesizer**

Granular synthesizers generate sound by combining small fragments of audio which are defined as grains. These grains are often short in duration and are manipulated by the parameters given to the synthesizer to create complex and evolving sounds. It's also useful for exploring a sound's tonal content without its rhythmic structure. The main parameters the Joy-Cons will be manipulating and controlling are:

* **Position** – the area in the audio where the grain sample can start. Controlled by the X axis.
* **Duration** – how long the grain duration is. Controlled by the Y axis.
* **Transposition** – pitch manipulation of the grain. Controlled by the X axis.
* **Panning** – panning the grain between the left and right audio outputs. Controlled by the X axis by the right Joy-Con.

A screenshot of a computer

Description automatically generated

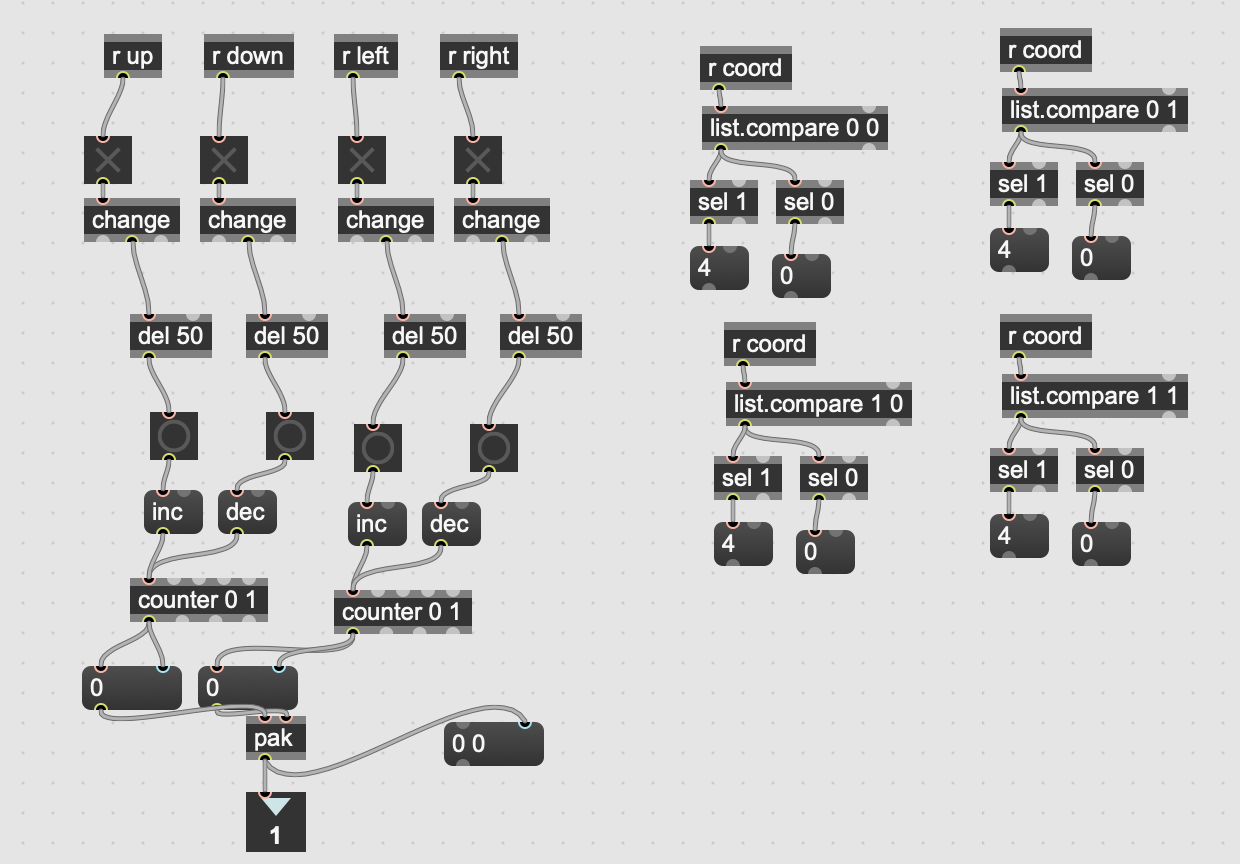
**Figure 7.** Screenshot of JoyousGrain.maxpat of the parameters interface.

Controllers will determine the minimum and maximum range for each parameter. The granular synthesizer will then randomly choose a number within that range to generate sound.

Other parameters that are not yet mapped to a control for the Joy-Cons are the gain for the synthesizer and the time between each grain. If the range for the duration parameter is set to be longer than the actual time between each grain, the synthesizer will prioritize the time for each grain over the duration.

* + 1. Changing Between Parameters

Since some of the parameters share the same movement controls (i.e. both Duration and Transposition use the X axis) there are controls in place which only allow you to control one parameter at a time. Using the *up, down, left,* and *right* buttons on the left Joy-Con, this will navigate between the parameters and only allow the user to manipulate the parameter they selected. This is performed by simulating a nested list where *up* and *down* buttons will change the first element and *left* and *right* control the second.



**Figure .** Depending on what button is pressed, list will change to be either [0 0], [0 1], [1 0], or [1 1]. List is then compared to see which parameter corresponds to that list.

* 1. **Regression**

Using multi-layer perceptron, a neural network process containing a number of layers, to control a synthesizer with 10 control parameters and 2 input controls of the XY parameters from the Nintendo Switch Joy-Cons. This will be implemented using regression, the neural network will predict output values that could go to anything for a set of input values that can come from anything. To train the regressor, we must collect a set of input and output example pairs to demonstrate when given these input values, it’s expected to return this output value. [*show an example of input pairs to an output pair. Each input/output pair share a common ID to show which input goes to what output*.].

A screenshot of a computer program

Description automatically generated

**Figure 5.** An example of input pairs to an output pair. Each input/output pair share a common ID to show which input goes to what output.

Each time the neural network receives an input example, it makes a guess for what the output values should be and compares the output values provided to determine how wrong it. Depending how wrong it is, it will correct itself to be more accurate next time.

* + 1. Fluid Corpus Manipulation

The objects used to implement the neural network are provided by Fluid Corpus Manipulation (FluCoMa)[[5]](#footnote-5). Following the example page provided by FluCoMa [13]. Changing the input parameters to receive the left stick movements, horizontal and vertical movements with the left stick is mapped as the input pairs for the regressor. The output pairs are the parameters to the granular synthesizer.

A screenshot of a computer

Description automatically generated

**Figure 6.** Screenshot of JoyousGrain.maxpat of the regression interface.

* 1. **Using The System**

When the user finds a desired sound and would like to save it, They will move the stick on the left Joy-Con to control the position on the [pictslider]. When I position is chosen, pressing the *plus* button on the right Joy-Con will add that point and parameter values as a pair in a buffer. This can be repeated multiple times. When finished, press the *minus* button to begin training the system. The message box will then change to say ‘fit’ followed by a decimal point number. The number corresponds to the error value meaning how inaccurate the neural network still is. In general, a lower value is better. In order to lower the error value, continuous pressing of the *minus* button will keep training it until the number starts flattening out. Now, moving the left stick will activate the system to change the parameters of the synthesizer without having to manually adjust the parameters.

1. RESULTS and Continued Development

A screenshot of a computer

Description automatically generated

**Figure 6.** Screenshot of the main MAX patch, joyousgrain.maxpat.

In order for the Max patch to receive OSC messages, a separate python code must be running concurrently. Data from the Nintendo Joy-Cons have successfully been mapped between the physical controllers to MAX. Once routed, data is smoothed and then scaled to be -1 to 1 in order to be compared between the other data sets on a visual aspect. Actual data received are then scaled again for the synthesizer parameters.

Switching between manually controlling the granular synthesizer and going to neural network to control the synthesizer appears to be in working condition with audible differences being heard immediately.

There are still minimal tasks that do not involve using the Joy-Cons such as importing an audio file to be used for the granular synthesizer, changing the gain levels and the time between each grain. The latter two could be an easy fix using another set of buttons and controls from the Joy-Cons while for the importing of an audio file, this can be mitigated by having an already preset sample on standby. Henceforth this could have the possibility of being completely autonomous with only Nintendo Joy-Cons.

One major issue that has arisen is the panning parameter. A stack overflow occurs immediately when doing anything to the received data coming from the Regressor-Switch.maxpat which is odd when none of the other data received from the patch seems affected. Various changing of the panning implementation does not resolve the issue. Current workarounds for this issue is having the panning be controlled manually by the right Joy-Con while the left Joy-Con controls the regression.

Future additions to this project could include a larger usage of the right Joy-Con while activating the neural network. This could be done by incorporating another audio processing (i.e. reverberation) element into the patch where the user can manipulate concurrently.

1. CONCLUSION

The project was successful in incorporating easy access game controllers for music generation. It shows potential for independent music systems that is easily replicated and readily available for performing.

Although the maxpatch is able to run and output results, further testing of the maxpatch is encourage in order to fine tune any errors that could occur. This should be done by spending more time using the granular synthesizer (i.e. different parameter settings, changing between audio samples, switching between manual and automatic usage of the synthesizer and vice versa).

1. REFERENCES
2. C. Ko, “The GLOBE: Evaluating Sensors for Interac-tive Computer Music Performance”, M.S. thesis, University of Calgary, Calgary, Cana-da, 2020. [Online]. Available: https://prism.ucalgary.ca/items/668ef7e7-f9b8-4bdb-8e11-5dbd60e926c3. [Accessed 27 Feb. 2024].
3. D. Wessel, M. Wright, and J. Schott, “Intimate Musical Control of Computers with a Variety of Controllers and Gesture Mapping Metaphors,” in *The International Conference on New Interfaces for Musical Expression*, May 1, 2002 [online]. Available: https://dl.acm.org/doi/10.5555/1085171.1085213. [Accessed 28 Feb. 2024].
4. G. Emerson and H. Egermann, “Exploring the Motivations for Building New Digital Musical Instruments,” Exploring the motivations for building new digital musical instruments, *Musicae Scientiae*, vol. 24, no. 3, pp. 313-329, Oct. 16, 2018. [Online]. Available: https://pure.york.ac.uk/portal/en/publications/exploring-the-motivations-for-building-new-digital-musical-instru. [Accessed 28 Feb. 2024].
5. M. Pessoa, C. Cláudio*,* P. Luís, I. Corintha, and G. Bernardes, “Examining Temporal Trends and Design Goals of Digital Music Instruments for Education in NIME: A Proposed Taxonomy,” in *The International Conference on New Interfaces for Musical Expression*, June 1, 2020. [Online]. Available: https://zenodo.org/records/4813210. [Accessed 28 Feb. 2024].
6. "My Nintendo Switch reverse engineering attempts," Aug. 2021. [Online]. Available: https://github.com/dekuNukem/Nintendo\_Switch\_Reverse\_Engineering. [Accessed 3 Mar. 2024].
7. Y. Seznec, “The Wii Loop Machine Musical Software Development for the Nintendo Wii Remote,” 2007. [Online]. Available: https://www.yannseznec.com/blog/ys\_Wii\_Loop\_Machine.pdf. [Accessed 29 Feb. 2024].
8. H. Fan, “Metamorphosis,” in *The International Conference on New Interfaces for Musical Expression*, May, 24, 2021. [Performance]. Available: https://nime.pubpub.org/pub/kjjp3zcs/release/1#program-notes. [Accessed 1 Mar. 2024].
9. M. Fosse, "JoyCon-Driver," 2018. [Online]. Available: https://github.com/fossephate/JoyCon-Driver. [Accessed 28 Feb. 2024].
10. B. Rieke, "joycon piece 1," 18 Mar. 2021. [Online]. Available: https://www.youtube.com/watch?v=jn4zqCtYzlk. [Accessed 28 Feb. 2024].
11. "joycon-python 0.2.4," 18 May 2020. [Online]. Available: https://pypi.org/project/joycon-python/. [Accessed 28 Feb 2024].
12. "python-osc 1.8.3," 5 Aug. 2023. [Online]. Available: https://pypi.org/project/python-osc/. [Accessed 28 Feb. 2024].
13. "What is OSC?," 13 Aug. 2021. [Online]. Available: https://opensoundcontrol.stanford.edu. [Accessed 28 Feb. 2024].
14. "Controlling a Synth using a Neural Network," [Online]. Available: https://learn.flucoma.org/learn/ regression-neural-network/.[Accessed 14 Apr. 2024].

1. Max is a visual programming language for multimedia projects. More can be found online at: https://cycling74.com/products/max. [↑](#footnote-ref-1)
2. More information on the [gamepad] object available on the MAX 8.6.1 documentation page found online at: https://docs.cycling74.com/max8/refpages/gamepad?q=gamepad. [↑](#footnote-ref-2)
3. More information on the [hi] object available on the MAX 8.6.1 documentation page found online at: https://docs.cycling74.com/max8/refpages/hi. [↑](#footnote-ref-3)
4. A collection of tools created by researchers by the Center For New Music And Audio Technologies (CNMAT). More information on https://cnmat.berkeley.edu. [↑](#footnote-ref-4)
5. External toolkit that focuses on contemporary machine listening and machine learning techniques. More information on https://learn.flucoma.org. [↑](#footnote-ref-5)