Visualization of Musical Instruments through MIDI Interface

John Jaminet

Visualization of Musical Instruments through MIDI Interface

John Jaminet

Senior Honors Project

Submitted in partial fulfillment of the graduation requirements of the Westover Honors College

Westover Honors College

May, 2020

____
Will Briggs, PhD

____
Barry Lobb, PhD

____
Jennifer Styrsky, PhD

Abstract

We have created a Music Visualization system that controls an LED strip by parsing MIDI signals that are sent by a musical instrument. This is achieved through a program that parses the MIDI note signals into three-byte RGB signals which are then transferred over WIFI to four Arduino boards that control the LEDs. The system provides a musician the ability to add a dynamic light display that responds to the music in real-time. The system utilizes Windows Presentation Foundation because of its event handling and GUI capabilities. The board we are using is the Arduino MKR 1010 WIFI board as if offers WIFI transmission and it can control the LEDs using pulse width modulation (PWM). The algorithms we designed for the RGB parsing are driven by the pitch and velocity of the MIDI note signals.

1. INTRODUCTION

Using light displays to compliment audio performances has become a regular practice in the field of performing arts. With the introduction of high-performing computers that are able to control both audio and video came the synthesis of the two in innumerable ways, most notably music videos and visualizers [1]. The goal of this practice is to add another dimension of sensory stimulation so that the audience can react to and appreciate the musical elements as they occur [2]. Music visualization is also used to aid those with auditory issues by expressing the music in an accessible media.

The motivation for this project was to make music visualization technology accessible to instrumental performers who do not possess the knowledge or resources to create a lighting system and script for their performances. By using this system, the performer simply has to connect their instrument to the computer through a MIDI interface, which is supported by a wide range of electronic instruments.

In this paper, we will describe how we created a graphic display program and a LED array driven by Arduino boards. This system is able to respond to signals received from a musical instrument that is connected through a MIDI (Musical Instrument Digital Interface) connection. We will then describe the methods used to parse that data into RGB (red, green, and blue) data values. By using this array and display program, a musician can perform a piece of music and have the system generate an immersive visual display in real time without the need of complex lighting scripts that are written for each specific piece.

2. Related Works

The most notable and widely used application that closely relates to this project is the Windows Media Player developed by Microsoft [3]. It offers visualization methods that feature varying colors and shapes along with support for visualizing the volume of the audio [2]. This project goes a step further by tailoring the colors to match the value of the notes. In other articles, researchers sought to use graphics to represent different aspects of either individual pieces or a collection of pieces for the purpose of identification and comparison. Lopes and Tenreiro Machado sought to create three dimensional graphs of artist's time series complexity over the course of their career [4]. This allowed them to analyze an artist's large collection of works in one three dimensional graph. Similarly, many methods have been developed that can visually represent an entire piece of music for the sake of comparison and categorization [5][6]. However, the focus of this project is the visualization of a live performance for the sake of enjoyment. Our goal is similar to that of Taylor, Boulanger, and Torres [7]. Their goal was to create animations that are affected by the piano and vocal cues of the performer. Our project seeks to accomplish this goal while also bringing in the LED hardware that compliment the display and make the performance immersive for the audience member.

3. METHODS

Our system is designed around a central Windows 10 computer that drives two separate programs that receive input from the MIDI device and drive either the graphics display or the LED controllers (see Figure 1).

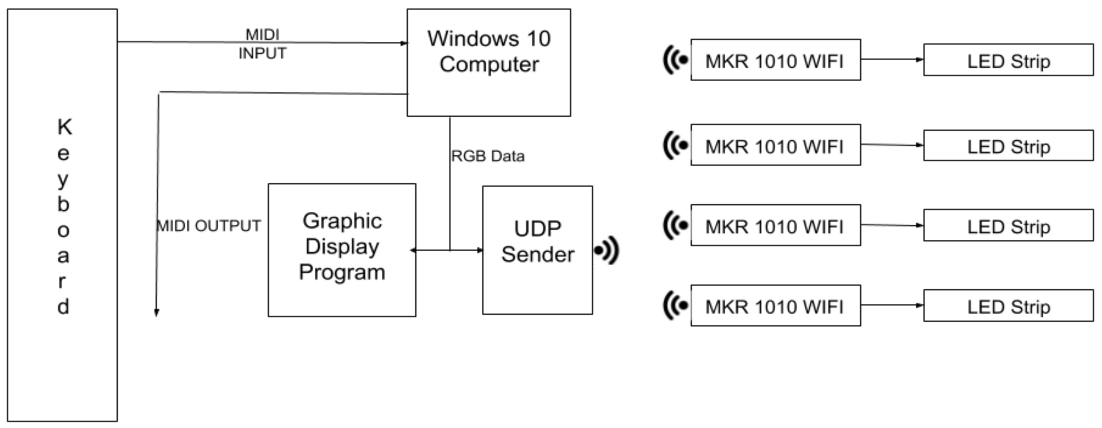

Figure 1 - Diagram of the MIDI to LED system

3.1 The need for Real-Time methods

With the goal of making this project something that performers can utilize without the need for prior setup or script writing, we had to rely on methods that provided immediate data that the system can respond to. Previous research have developed methods of music visualization that represent an entire piece of music such as the work by Hall. Similarly, the work by Kling and Roads promises to lead to visualizations that relate to the sound's time and frequency patterns, something that real-time visualizers struggle to do [8][9]. For this application, we relied on methods similar to those expressed by Fonteles, Rodrigues, and Basso where graphics objects are driven by the pitch and volume of each note [10]. The advantages and disadvantages of real-time vs full animation is described by Kubelka [11].

3.2 Basics of MIDI Signals

MIDI (Musical Instruments Digital Interface) is a standard for communication protocols used in digital instruments and computers for playing, editing, and recording music. The MIDI signals are received as 6 or 8-digit hexadecimal values where each 2-digit pair holds the value of a different piece of the message. The first pair says which channel the message is coming from, typically channel 0 when receiving from a single instrument. The second pair contains the type of message being communicated. For example, the number 90 would be the code for a note on message and 80 would be note off. Following the message type is the numerical value of the note which can range from 0 to 127. In the case of a note on message, an extra two digits will be added which tells the velocity or volume of the note that was played. By applying these standardized rules, the programs can interpret those signals into 8-bit RGB values. The program that runs the LED controllers then passes values to a UDP (User Datagram Protocol) package sender for broadcasting over WIFI. The graphic display program also receives the MIDI messages and interprets them in the same manner in order to create visual elements to reflect those signals.

3.3 LED Controllers

The LED controllers are built around an Arduino MKR 1010 WIFI microcontroller. We chose this board because it has built in WIFI capabilities which would allow us to control multiple controllers from a single computer at the same time. This board also provides pins with PWM (pulse width modulation) capabilities which was needed in order to control the brightness of each color on the LEDs [12]. Additionally, the MKR 1010 board is smaller than the standard Arduino UNO which made the controllers more portable and less costly (see Figure 2).

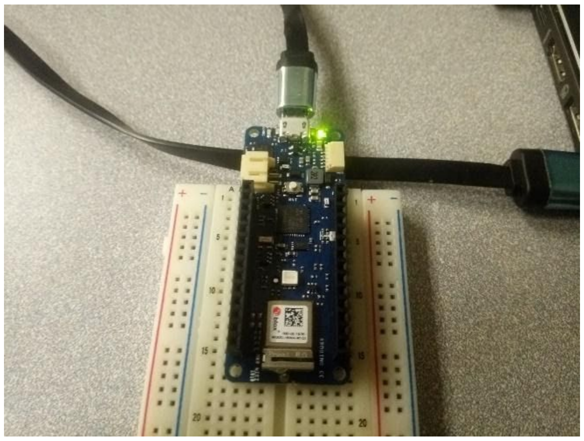

Figure 2 - Arduino MKR 1010 WIFI

The LEDs being used in this project are the RGB LED strip lights kit by Lighting Ever. We chose these lights because they feature exposed copper pads which allowed us to solder our own jumper wires to the strips so as to connect them to the Arduino board. The strips are also able to be cut into small segments which allow us to run tests on smaller scales. In order to boost the 3.3-volt signals from the Arduino to the 9-12 volts required for the LEDs, we are using N-Channel MOSFETs and 100-ohm resistors as recommended by Adafruit Learning System [13] (see Figure 3).

While the LED strips are intended to run at 12 volts, they are capable of running at 9 volts. We decided on this lower voltage because we are powering the LEDs with a 9-volt battery, thus eliminating the need for a 12-volt wall adapter. With the addition of a 9v to 5v step-down, we are able to power both the lights and the Arduino boards of each controller with the battery which eliminates the need for an external power source (see Figure 3).

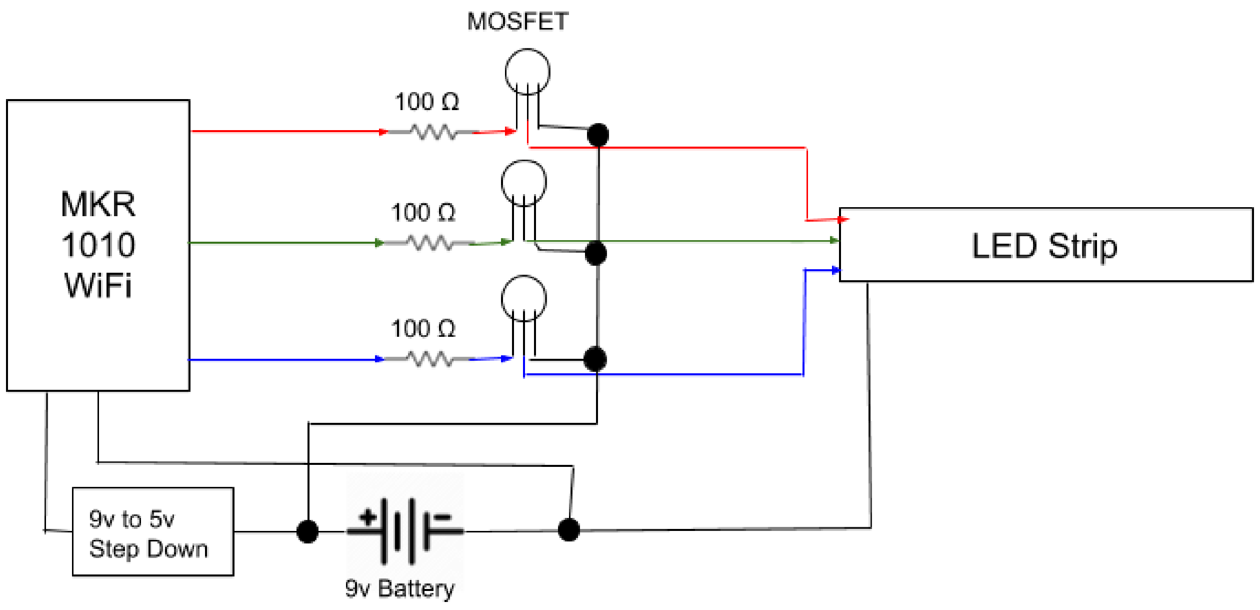

Figure 3 - Diagram of the Arduino to LED Circuit

3.4 MIDI signal retrieving

The instrument being used for generating MIDI messages for the system is the Roland ep-760 keyboard (see Figure 4). We are using a keyboard because it generates precise and predictable MIDI messages and it demonstrates how a musician would be able to control the system. The keyboard is connected to the computer using a M-audio Midisport Uno which provides one-to-one MIDI messaging and 16 MIDI channels. In order to control the MIDI data stream, we implement a couple high-level libraries that would make the data stream accessible and easy to parse. For the program that is driving the LED controllers, we include the N-Audio library because it is available for .Net and it is integrated with C# [14]. As for the graphic display program, we are utilizing the PortMidi library because N-Audio does not support C++ and it is a higher-level branch of its predecessor, PortAudio. This makes it attractive for implementation into our project [15][16].

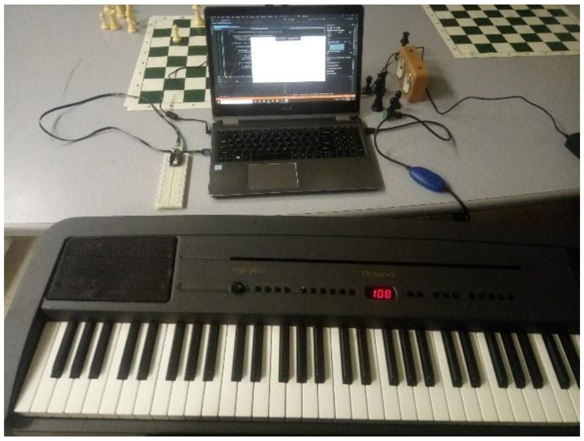

Figure 4 - The keyboard sends signals to the computer using a M-audio Midisport Uno

3.5 MIDI to RGB Parsing for the LEDs

The purpose of the computer in the system is to parse the incoming MIDI signals into three-byte RGB data that can be interpreted by the Arduino. This process is limited because we can only extract certain aspects of the music while maintaining our real-time requirement. Aspects such as tempo require some amount of time to pass before we can begin to calculate what the tempo is, so we cannot rely on this dimension of the music [17][18]. We wrote a program using Windows Presentation Foundation in C# because it is focused on event driven programs and it includes support for GUIs, both of which are necessary for this project. With the GUI, the user can decide on the method of color assignment that they would like to use and select which MIDI device that it should listen to along with the channel as most MIDI signals offer sixteen channels. We designed a few different methods for signal parsing that a user can choose from.

The first method relies solely on the pitch of the note. By picturing the colors blue, green, and red as a one-dimensional spectrum in that order, the lowest pitch played would correspond to blue and the highest pitch would correspond to red with all other pitches being assigned a color from the spectrum. The desired effect is that higher pitches will be assigned warmer colors that fall between green and red and lower pitches will get cooler colors from between blue and green. The second method adds in the variable of volume or velocity of the note. Since the pitch determines the values of either blue and green or green and red, the velocity controls the value of the third color which is either blue for high pitches or red for low pitches. The goal is so that as the volume increases, the colors become brighter as they shift towards white.

3.6 Music Visualizer

The other half of our system deals with generating a graphic display program that reacts to input it receives from the MIDI stream. Our program is based on a Game Engine developed by Sanjay Madhav for the purpose of teaching game programming which relies on the popular SDL library [19]. As our program does not require three dimensions or the use of complex models, we decided that this simple game engine would be sufficient. We decided that a game engine would be the best way to achieve our goals for the music visualizer because they feature a game loop that would allow us to continuously generate visual output. They also provide many methods and classes for handling user input and generating objects that can react to that input. From there, we implemented the PortMidi library that was mentioned previously which generates the input events for the program.

While this program does its MIDI parsing separately from the LED controller program, it follows the same methods described in section 2.3 so that the colors on the display and the LED are consistent. We then took these color values and utilized SDL's ability to modify the color of a texture as a way of rendering objects with the appropriate color assignment. While this method can be performed on any texture, we saw that this process delivered the most desirable results when using a white texture.

In addition to the piano input and color assignment, we include extra background effects such as fog and light trails and movement components to the piano-driven objects. These elements combined complete the basic scene of our visualizer. This scene features a white laser trails that spans the screen and move from right to left. As the piano input is received, a colored rectangular outline is drawn on the right side of the screen, on top of the white laser. The height of the rectangle is determined by the volume of the note while the color is determined by the assignment algorithm. As the note is played, the outline extends horizontally until the note is released in which the outline then travels down the laser until it goes off the screen.

4. RESULTS

The finished programs show that we were able to parse a MIDI stream on input from an electric piano into colors data and display that data in an immersive visual display. The color assignment algorithm was implemented into both the graphics display and the LED controllers' programs so that the two components complimented each other. For a demonstration of the working system, please see the link included in the reference section [20].

5. CONCLUSION

The system is designed to provide musical performers a format for creating immersive light display that respond to their playing through a MIDI interface. Our system provides a central graphics display and four LED controllers that can be arranged around a dark room or concert hall to provide an appealing light show. With the included MIDI parsing software, the lights are able to respond to the musician without the need for a traditional lighting script. We accomplished the other goal of this system which was to make it technologically accessible to performers in terms of a "plug-and-play" system where the only technical requirements for the user is to plug in their instrument and start the programs. In designing the system for simplicity, the user has little control over what the program displays beyond being able to choose the scene that is selected. With this limitation, any future designs would have to be hard coded directly into the visualizer. While this is a drawback, it does provide an opportunity for further work to develop a system that would allow users to have more control over each scene or even the possibility of creating user generated scenes by incorporating existing elements into a 'blank canvas.' Light and graphics displays continue to be used as a method of music visualization in nearly all professional performances. We hope that this system will become a solution for artists who want to add these elements to their performances.

ACKNOWLEDGEMENTS

I would like to thank my thesis committee, Dr. Will Briggs (committee chair), Dr. Barry Lobb, and Dr. Jennifer Styrsky for their assistance and feedback. I would like to thank Dr. F. Johnson Scott and Dr. Kara Dean of the Music College for providing the music hardware. I would like to also thank the Schewel Student-Faculty Research Fund for their financial support of this project. I would also like to thank the Westover Honors College, the Computer Science department of the School of Sciences, and the University of Lynchburg.

REFERENCES

[1] M. Bain, “Real Time Music Visualization: A Study in the Visual Extension of Music,”M.F.A. thesis, Grad. School of Fine Arts, Ohio State University, Columbus, OH, USA, 2008.

[2] M. Vieira, “Interactive Music Visualization - Implementation, Realization, and Evaluation,” Master Dissertation, Universidade da Madeira, Funchal, Madeira, Portugal, 2012.

[3] Microsoft Corporation, *Windows Media Player.* 2019.

[4] A. Lopes, J. Tenreiro Machado, “On the Complexity Analysis and Visualization of Musical Information,” *Entropy,* vol. 21, no. 669, 2019.

[5] M. Cooper, J. Foote, E. Pampalk, and G. Tzanetakis, “Visualization in Audio-Based Music Information Retrieval,” *Computer Music Journal*, vol. 30, no. 2, pp. 42–62, 2006.

[6] H. Wu, J. Bello, “Audio-Based Music Visualization for Music Structure Analysis,” Music and Audio Research Library, New York University, 2010.

[7] R. Taylor, P. Boulanger, D. Torres, “Real-Time Music Visualization Using Responsive Imagery,” Department of Computer Science, University of Alberta, 2006.

[8] R. Hall, “Geometrical Music Theory,” *Science,* vol. 320, pp. 328-329, April 12, 2008.

[9] G. Kling, C. Roads, “Audio Analysis, Visualization, and Transformation with the Matching Pursuit Algorithm,” in *Proc. of the 7th International Conference on Digital Audio Effects,* Naples. Italy. October 2004, pp. 33-37.

[10] J. Fonteles, M. Rodrigues, V. Basso, “Creating and Evaluating a Particle System for Music Visualization,” *Journal of Visual Languages and Computing,* vol. 24, pp. 472-482, 2013.

[11] O. Kubelka, “Interactive Music Visualization,” Department of Computer Science and Engineering, Czech Technical University, 2000.

[12] J. Okumura, Y. Kozawa, Y. Umeda, H. Habuchi, “Hybrid PWM/DPAM Dimming Control for Digital Color Shift Keying Using RGB-LED Array,” *IEEE Journal on Selected Areas in Communications.* vol. 36, no. 1, pp. 45-52, Jan. 2018.

[13] Adafruit Learning Systems “RGB LED Strips,” Adafruit Learning Systems. 2018. [Online] Available: https://cdn-learn.adafruit.com/downloads/pdf/rgb-led-strips.pdf. [Accessed: Nov. 16, 2019].

[14] M. Heath, NAudio. 2019.

[15] R. Bencina, P. Burk, and R. Dannenburg, PortMidi. 2009.

[16] R. Bencina, P. Burk, PortAudio. 2011.

[17] M. Gainza, E. Coyle, “Tempo Detection Using a Hybrid Multiband Approach,” *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 19, no. 1, pp. 57-68, Jan. 2011.

[18] F. Gouyan, S. Dixon, “A Review of Automatic Rhythm Description Systems,” *Computer Music Journal,* vol. 29, no.1, pp. 34-54, 2005.

[19] S. Madhav, *Game Programming in C++.* Pearson Education Inc. 2018.

[20] J. Jaminet, *Demonstration of Visualization of Musical Instruments Through MIDI Interface.* https://youtu.be/2dkYsXfp0HQ.