Micro Controller Modeling On Digital FX Pedal For Guitar And Bass Using Arduino Pro Micro

Hanafi Dana Bastyan[1], Putri Febi Utami[2]

Departement of Informatic, Faculty of Computer Science, Universitas Amikom Purwokerto, Indonesia
Email: [1]vlamingvlaming0@gmail.com, [2]febiiutamii@gmail.com

ARTICLE INFO

History :

Submit on 16 May 2024
Review on 26 June 2024
Accepted on 14 July 2024

Keyword :

Micro Controller,
Design FX,
Sound Effect,
Arduino

ABSTRACT

A musician, especially a guitarist or a bassist, is certainly familiar with digital sounds or tones. Almost in the current technological era, they also use technology in music. Currently, there are many digital music devices. One of them is the use of digital effects for guitars and basses. These devices very widely, from large to compact. Sometimes guitarists and bassists often have difficulty controlling them. This is because the design of the device is made minimalist, so that with the large number of preset tones available with a small number of controls, it will complicate their performance on stage. The use of digital effects requires a large storage capacity, therefore, the role of the digital effect is like a processor in the CPU. If so much storage is not controlled, then it is possible to forget the configuration settings. Therefore, control is needed as a support and a tool to remind the preset memory. Therefore, additional devices are needed to control the preset tones. This paper will discuss how to create a model to control preset tones in the digital effects they use. The goal is to make it easier when they perform on stage. With good control features, and the arrangement of the appropriate tone arrangement, guitarists and bassists will find it easy to manage the presets stored in the digital effect memory.

Corresponding Author:

Hanafi Dana Bastyan
Departement of Informatic, Faculty of Computer Science, Universitas Amikom Purwokerto, Indonesia
Email: vlamingvlaming0@gmail.com

INTRODUCTION

Arduino is an open-source platform used to create electronic projects. Arduino consists of hardware in the form of a programmable microcontroller, and software in the form of Arduino IDE (Integrated Development Environment) which is used to write and upload code to the microcontroller[1]. Some of the main features of Arduino include Arduino is designed to make it easier for people from various backgrounds to create electronic projects without having to have in-depth knowledge of electronics or programming. Arduino is available in various models, each of which has certain specifications and features, such as Arduino Uno, Arduino Nano, Arduino Mega, and others[2]. Arduino has a large and active community, so there are many examples of projects, tutorials, and libraries that can help in project development[3]. Arduino IDE is available for various operating systems such as Windows, macOS, and Linux, making it more accessible to various users. Arduino is often used in various projects such as robotics, sensors, automation systems, MIDI controllers, and many more. This platform is very flexible and can be used for small-scale projects to more complex prototypes[4].

Digital effects on guitar and bass are devices or software that alter the original sound of a guitar or bass by electronically processing the audio signal to create a variety of sounds[5]. These effects allow musicians to explore textures, colour, and sonic characters that would be impossible to achieve with the instrument itself. Some examples of commonly used digital effects include[6]: (1) Distortion/Overdrive: This effect amplifies the audio signal to a certain extent, producing the "rough" or "cracked" sound typical of rock and metal music. (2) Chorus: The chorus effect creates the illusion that there are multiple instruments playing simultaneously with slight differences in timing and pitch, resulting in a thicker, richer sound. (3) Delay: Delay delays the resulting audio signal and then plays it back after a certain amount of time. This creates a timed echo effect. (4) Reverb: Reverb effects mimic the resonance of sound in a specific space, such as a hall or small room, providing a deeper, richer sound. These digital effects are usually controlled via foot-activated effects pedals (stomp-boxes), or via multi-effects devices that combine several effects in one unit[7][8]. Many modern digital effects can also be accessed and controlled via software or apps on computers and mobile devices, allowing for more advanced customization and saving of presets[9].

A controller in digital effects acts as a device or interface that is used to control the various parameters and functions of the effect[10]. This allows musicians to change the sound and activate or deactivate effects in real-time, both in the studio and during live performances[11]. A controller allows the user to quickly activate or deactivate a digital effect. For example, a stomp-box pedal can be used to turn a distortion effect on or off during a performance. With a controller, the musician can change effect parameters such as the level of distortion, delay time, or modulation speed in real-time[12]. This provides flexibility in creating different nuances of sound while playing. Many digital effects and multi-effects devices allow the user to save presets (pre-programmed sound settings)[13]. A controller can be used to quickly select and move presets, allowing the musician to move between different sounds seamlessly while playing. Some controllers, such as an expression pedal, allow more dynamic control over a particular effect[14]. For example, an expression pedal can be used to control the volume, wah-wah, or level of an effect continuously while playing. In a looper, a controller allows the musician to record, overdub, and play back audio loops with complete control[15]. This is often used in live performances to create complex layers of music from a single instrument. On devices that support MIDI (Musical Instrument Digital Interface), controllers can be used to send MIDI signals that control various parameters and program switching on digital effects devices[16]. This allows for greater integration between devices in a musical setup. Some controllers can be used to automate parameter changes or to synchronize effects to a specific tempo, such as synchronizing a delay to the speed of a song. This is especially useful in both studio and live settings. In the digital world, controllers play a vital role in giving musicians complete control over various aspects of their sound effects, allowing for richer and more dynamic musical expression[14].

METHOD

In this article will discuss about how to model a digital effect controller for the guitar or bass. In this article will be discussed using Arduino Pro Micro with ATMega 32u4 processor. The stages that will be done can be seen in the following figure 1:

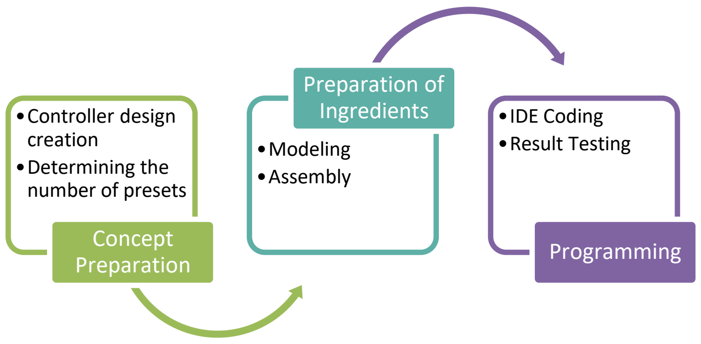

Controller design creation
Determining the number of presets
Concept Preparation
Preparation of Ingredients
Modeling
Assembly
IDE Coding
Result Testing
Programming

Figure 1, Design flow of digital effect controller model.

The first stage is concept preparation. This preparation requires materials including the number of buttons, markers using LED segments, and a USB MIDI port.

Then the second stage is modeling and assembling the materials that have been provided. The materials available include a 30cm x 12cm case, 8 push-on buttons, a Mini USB port and Type C port, and 1 9V battery socket for additional power. Another important material is 1 Arduino Pro Micro tool kit with an ATMega 32u4 processor as its main controller.

After the materials are ready, they are assembled according to the existing design. After that, programming is carried out for the functions of the buttons that have been arranged. This programming uses Arduino IDE version 2 software.

After the programming stage is complete, the next step is testing. The first thing to test is the MIDI connection function from the device to the processor response in the Arduino IDE. After being turned on, there are no code errors, then the connection with the device that has been prepared as a trial tool.

RESULT & DISCUSSION

The first step is to prepare the design model to be made. This design requires at least 1 casing to install and 1 electronic device complete with Arduino. Then do not forget to also prepare software such as Arduino IDE which will be needed in coding later. The materials prepared include those listed in the following table:

Table 1, Tools and Materials for Modeling Foot Controller.

Devices	Qty	Function
Casing 30x12cm	1	As a container for the main device
Arduino Pro Micro ATMega 32u4	1	Processor
Push on	8	Controller Button
Segment LED	1	Controller screen
Socket Mini USB & Type C	2	Connector

To clarify the materials used, please see Figure 2 below:

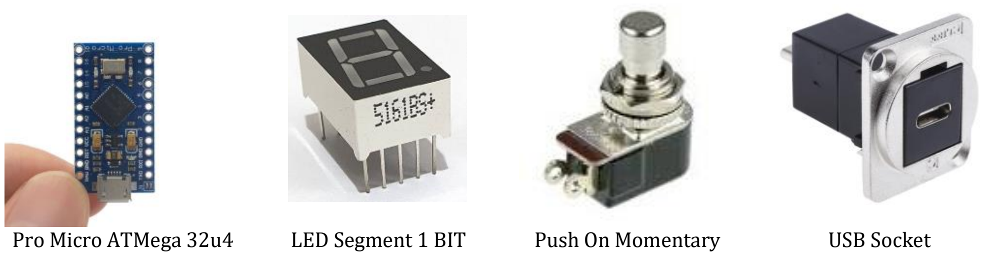


Figure 2, Modeling materials

Then for the design we have designed it in a form like in Figure 3 below:

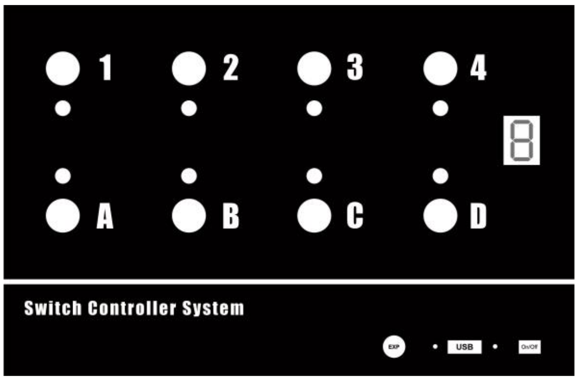


Figure 3, Controller design model.

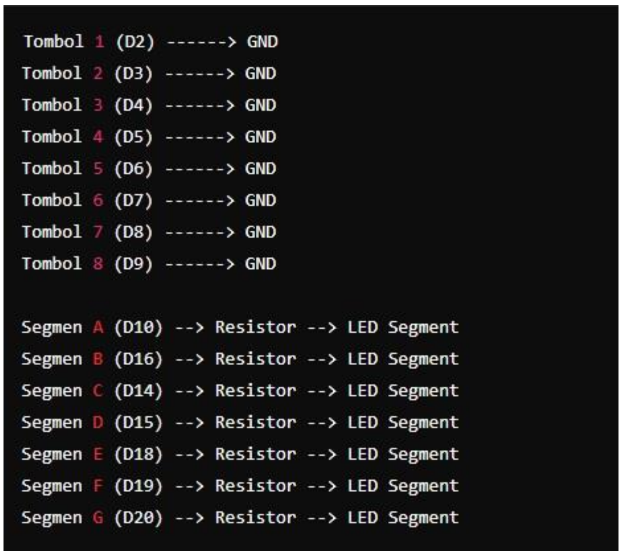
The second stage is to compile a concept with the materials that have been prepared. This process will determine how it will function in this digital effect control system. The first thing to do is to determine the function of the existing buttons. To connect 8 buttons and 1 digit LED segment to Arduino Pro Micro with ATmega32U4, you need to know the Arduino Pro Micro pin layout and set up the connections efficiently. Arduino Pro Micro has several digital and analog pins that you can use to connect buttons and LED segments. Here are the commonly used pins:

- Digital Pins: D2 to D21
- Analog Pins: A0 to A3 (also function as D18 to D21)
- Power Pins: VCC, RAW, GND
- USB Pins: USB +, USB –

Here is the layout of the model designed for the Pin arrangement for 8 Buttons and 8 Segment LED Input Buttons:

1. First connect 8 buttons to the digital pins. Each button is connected between the digital pin and ground, with the internal pull-up resistor enabled in the code. The arrangement can use pins D2 to D9 for the buttons.
2. The 1 digit segment LED has 7 segments (A-G) connected to other digital pins. The arrangement can use pins D10 to D16 to connect segments A to G.

To clarify the settings on the pin, please see the following configuration image:



```
Tombol 1 (D2) ------> GND
Tombol 2 (D3) ------> GND
Tombol 3 (D4) ------> GND
Tombol 4 (D5) ------> GND
Tombol 5 (D6) ------> GND
Tombol 6 (D7) ------> GND
Tombol 7 (D8) ------> GND
Tombol 8 (D9) ------> GND

Segmen A (D10) --> Resistor --> LED Segment
Segmen B (D16) --> Resistor --> LED Segment
Segmen C (D14) --> Resistor --> LED Segment
Segmen D (D15) --> Resistor --> LED Segment
Segmen E (D18) --> Resistor --> LED Segment
Segmen F (D19) --> Resistor --> LED Segment
Segmen G (D20) --> Resistor --> LED Segment
```

Figure 4, Configuration and PIN settings for buttons and LED segments.

Special notes to note in installing the pins are, adding resistors for the LED Segment, namely by using resistors (~220Ω) on each segment pin to prevent damage to the LED. Then Internal Pull-up the button by using the INPUT_PULLUP command when defining the button pin to use the Arduino internal pull-up resistor. With the above setup, you can connect 8 buttons and 1 digit LED segment to Arduino Pro Micro ATmega32U4. The buttons are used as input to send MIDI commands, and the LED segment displays the status or number of the active button. This configuration allows you to create an efficient and functional foot controller.

The third stage is to use the Arduino IDE software for the coding process that will be uploaded into the ATMega 32u4 processor. The code written is as follows:

```
#include <MIDIUSB.h>
const int buttonPins[8] = {2, 3, 4, 5, 6, 7, 8, 9};
const int segPins[7] = {10, 16, 14, 15, 18, 19, 20};
const byte digitPatterns[10] = {
  0b0111111, // 0
  0b0000110, // 1
  0b1011011, // 2
  0b1001111, // 3
  0b1100110, // 4
  0b1101101, // 5
  0b1111101, // 6
  0b0000111, // 7
  0b1111111, // 8
  0b1101111  // 9
};
bool buttonState[8] = {0, 0, 0, 0, 0, 0, 0, 0};
bool lastButtonState[8] = {0, 0, 0, 0, 0, 0, 0, 0};
void setup() {
  for (int i = 0; i < 8; i++) {
    pinMode(buttonPins[i], INPUT_PULLUP);
  }for (int i = 0; i < 7; i++) {pinMode(segPins[i], OUTPUT);}}
void loop() {
  for (int i = 0; i < 8; i++) {
    buttonState[i] = !digitalRead(buttonPins[i]);
    if (buttonState[i] && !lastButtonState[i]) {
      midiEventPacket_t midiMsg = {0x09, 0x90, 60 + i, 127};
      MidiUSB.sendMIDI(midiMsg);
      MidiUSB.flush();
      displayDigit(i + 1);}
    lastButtonState[i] = buttonState[i];}}
void displayDigit(int digit) {
  byte pattern = digitPatterns[digit % 10];
  for (int i = 0; i < 7; i++) {
    digitalWrite(segPins[i], bitRead(pattern, 6 - i));}
}
```

The previous code explanation is as follows:

1. Pin Setup, buttonPins[]: Array to store pins connected to 8 control buttons. segPins[]: Array to store pins connected to 7 LED segments.
2. digitPatterns[] array, Binary patterns for each number (0-9) that determine which segments should light up to display that number.
3. Setup, The buttons are set as inputs with internal pull-ups. LED segments are set as outputs.
4. Loop, Reading Buttons: The program reads the status of each button. Sending MIDI Messages: When a button is pressed, a MIDI Note On message is sent. Setting LED Segments: The LED segments display the number of the active button.
5. Function displayDigit(int digit), Displays a number on the LED segment by activating the corresponding segment based on the binary pattern defined in the digitPatterns[] array.

The above code integrates 8 control buttons with 1 bit segment LED as an active indicator on a MIDI USB foot controller using Arduino Pro Micro. You can modify and extend this code according to the specific needs of the project.

After the coding process is complete, the next step is to test the code results. The first test is a simulation of the results process. The device used for the test sample is 1 unit of the ZOOM MS60B digital effect pedal with a Hairless MIDI bridge.

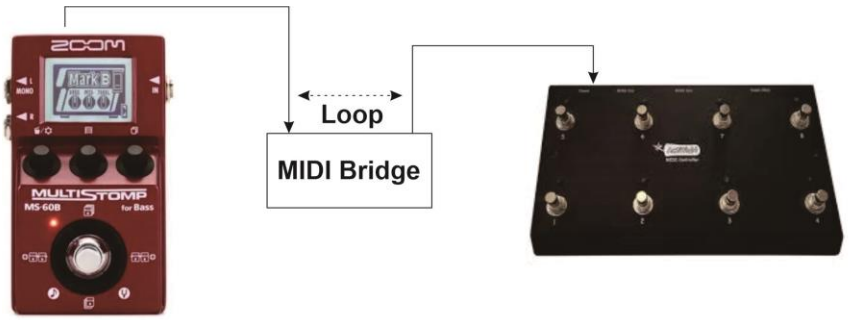


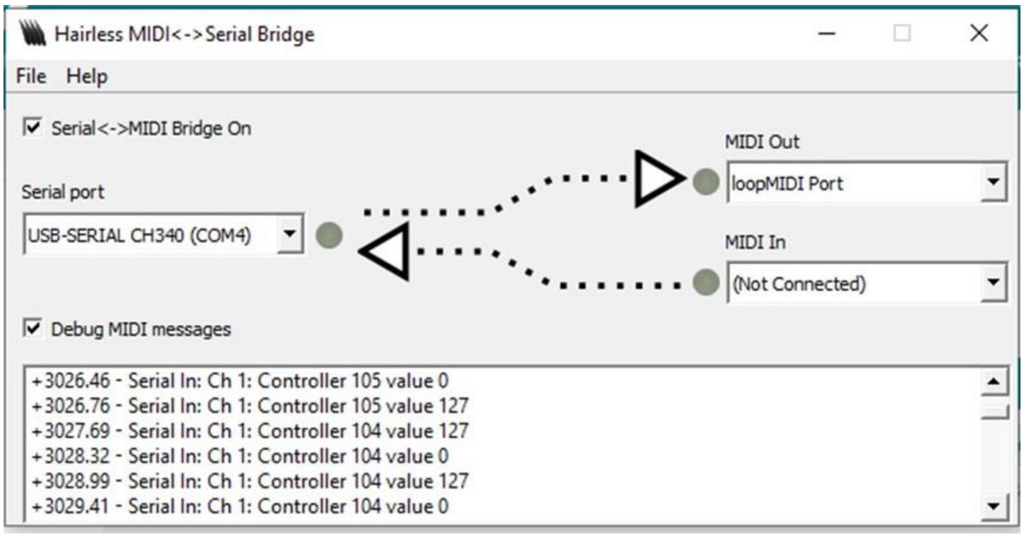
Figure 5, Control code function testing process

From the test results, the results obtained are as shown in Table 2 as follows:

Table 2, Button test results.

Code	Function	Control	Status
Button 1 (D2)	Push On	Control Preset 1	Ok
Button 2 (D3)	Push On	Control Preset 2	Ok
Button 3 (D4)	Push On	Control Preset 3	Ok
Button 4 (D5)	Push On	Control Preset 4	Ok
Button 5 (D6)	Push On	Control Preset 5	Ok
Button 6 (D7)	Push On	Control Preset 6	Ok
Button 7 (D8)	Push Up	Control Preset 7	Ok
Button 8 (D9)	Push Down	Control Preset 8	Ok

For the testing process in table 2 using the Hairless MIDI Bridge software, the results obtained are as shown in the following image:



Serial<->MIDI Bridge On ✓
Serial port: USB-SERIAL CH340 (COM4)
MIDI Out: loopMIDI Port
MIDI In: (Not Connected)
Debug MIDI messages ✓

```
+3026.46 - Serial In: Ch 1: Controller 105 value 0
+3026.76 - Serial In: Ch 1: Controller 105 value 127
+3027.69 - Serial In: Ch 1: Controller 104 value 127
+3028.32 - Serial In: Ch 1: Controller 104 value 0
+3028.99 - Serial In: Ch 1: Controller 104 value 127
+3029.41 - Serial In: Ch 1: Controller 104 value 0
```

Figure 6, Test results using Hairless MIDI Bridge.

Although Arduino has limitations in audio processing, you can still try to create guitar effects that resemble digital sounds with a simple approach. However, for results closer to professional quality, you may need to use a platform with higher processing power or additional devices that support audio processing.

CONCLUTIONS

Creating a guitar effect control that produces a sound like a digital tone with Arduino is an interesting project, but also quite challenging because it requires real-time audio signal processing. Arduino has limitations in terms of audio processing, but there are several approaches that can be tried to achieve this effect. By creating additional media to control the tone, musicians, especially guitarists or bassists, will find it easier to perform on stage. This is certainly very helpful. For advice, the use of this Arduino media is specifically for those that support USB MIDI. Apart from the Arduino Pro Micro that we use, it may be necessary to add a traditional MIDI tool. This is necessary if the processor on the Arduino does not allow the use of a USB port.

REFERENCE

[1] A. C. Bento, E. M. L. da Silva, and E. M. de Souza, "An Experiment With NRF24L01+ and Arduino Pro Micro Data Transmission for IoT," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2019, pp. 1–6.
[2] Y. A. Badamasi, "The working principle of an Arduino," in *2014 11th international conference on electronics, computer and computation (ICECCO)*, 2014, pp. 1–4.
[3] A. J. Bianchi and M. Queiroz, "Real time digital audio processing using Arduino," in *Proceedings of the Sound and Music Computing Conference, Stockholm, Sweden*,

2013, pp. 538–545.
[4] R. Vieira and F. L. Schiavoni, "Fliperama: An affordable Arduino based MIDI Controller.," in *NIME*, 2020, pp. 375–379.
[5] R. P. Pirotti, M. Johann, and M. Pimenta, "Design and implementation of an open-source subtractive synthesizer on the Arduino Due platform," 2019.
[6] M. P. IRUNGU, "DESIGN AND CONSTRUCTION OF A MICROCONTROLLER-BASED HUMAN AUDIO NOISE MONITORING AND CONTROL SYSTEM," KENYATTA UNIVERSITY, 2020.
[7] L. De Lauretis, T. Lombardi, S. Costantini, and L. Clementini, "An Arduino-based Device to Detect Dangerous Audio Noises.," in *IoTBDS*, 2021, pp. 303–308.
[8] S. Kamble, R. Bhagat, S. Mahule, S. Choudhary, and R. A. Burange, "A REVIEW ON E-BIKE VIBRATION AND SOUND IMITATING MODULE," in *International Conference on Electronics \& Computer Exigencies ETECE*, 2023.
[9] A. Valle, "Audio physical computing," *Algorithms*, vol. 1, no. 2, p. 3, 2011.
[10] A. Ahlzén, V. Holma, A. Segerberg, S. Varahram, and M. Wiig, "Piano Hero: Interactive musical learning." 2024.
[11] E. Stifjell, "A FLexible musical instrument Augmentation that is Programmable, Integrated in a Box (FLAPIBox)," 2023.
[12] A. Romanchuk and V. Chyhin, "EXPERIMENTAL SYSTEM AND SOFTWARE: TWO-DIMENSIONAL RETURN OF THE PLATFORM TO THE SOUND SOURCE," 2023.
[13] F. Lelli, "PoliMotion: a gesture driven audio control system," 2019.
[14] P. Patel, N. Gupta, and S. Gajjar, "Real Time Voice Recognition System using Tiny ML on Arduino Nano 33 BLE," in *2023 IEEE International Symposium on Smart Electronic Systems (iSES)*, 2023, pp. 385–388.
[15] S. Dimitrov and S. Serafin, "Towards an open sound card: bare-bones FPGA board in context of PC-based digital audio: based on the AudioArduino open sound card system," in *Proceedings of the 6th Audio Mostly Conference: A Conference on Interaction with Sound*, 2011, pp. 47–54.
[16] R. Kjærbo, L. Fogadic, O. B. Winkel, and S. Serafin, "Daisy Dub: a modular and updateable real-time audio effect for music production and performance," in *Proceedings of the Sound and Music Computing Conference*, 2023, pp. 49–57.