MyoEffect

Team Members:

- paulgm2 (netid)
- karans4 (netid)
- schanne3 (netid)

Problem

Most musicians working in digital audio workspaces (DAWs) desire fine control over the effects they use and the intensity at which those effects operate. Most of the time, in order to quickly assign effects to controls and to fine tune them, musicians will make use of MIDI controllers; however, these controllers are often both clunky to use and expensive to obtain.

Solution

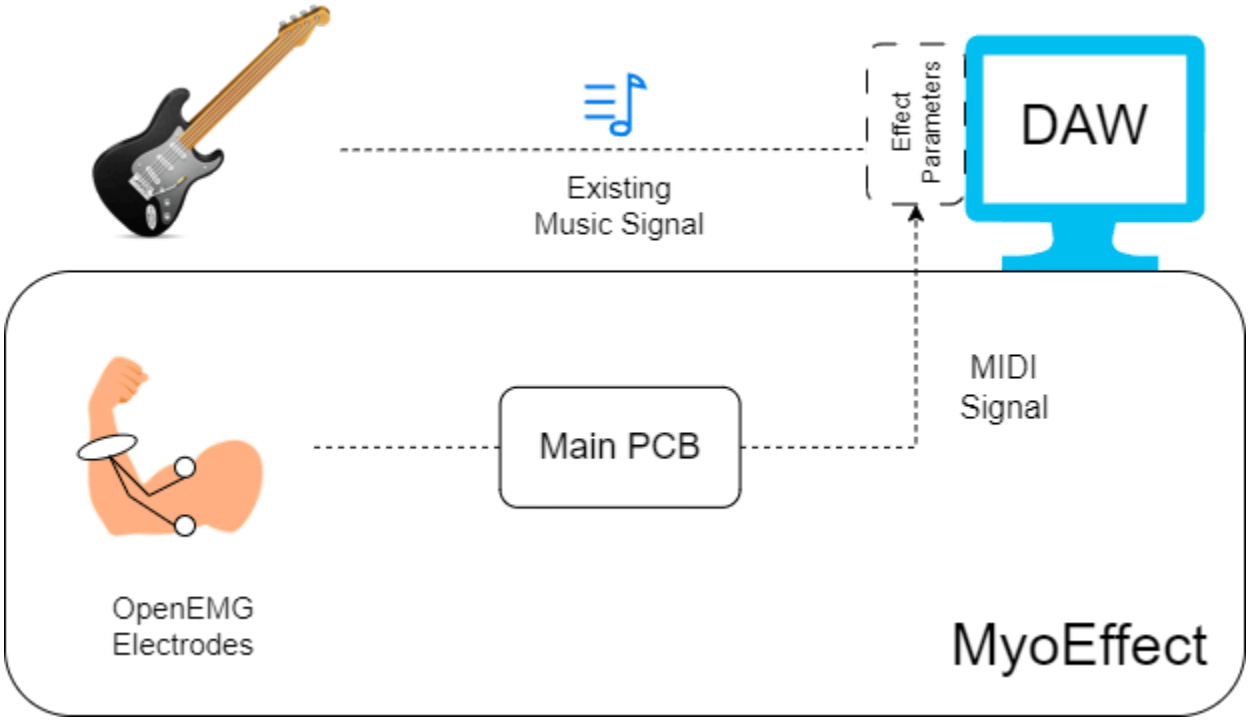
Our design offers a cheaper and more expressive method of using effects by bypassing the use of a separate controller entirely and controlling effects with the motion of the musician's own body. Instead of having to turn knobs or press buttons, a musician can simply close their fist, or extend their arm, or twist their leg; our design's goal is to give complete freedom to the musician as far as what motion they want to control their effects. Our design makes use of electromyography (EMG) sensors to map contractions of users' muscles to voltage readings; these readings will then be converted into corresponding MIDI parameters. Our device can then be plugged into a DAW and operate in exactly the same way as a normal MIDI controller.

As far as our implementation of this idea, we will create a bracelet or watch-like attachment to house the circuitry needed for our EMG sensors. These attachments will be designed to be easily adjustable and movable for reading inputs from arm and leg muscle groups. Later sections will talk about how many sensors each attachment will house and use, as well as the trade-offs for each implementation.

Our process for converting the analog voltage read from the EMGs to a corresponding MIDI signal is as follows: read the analog voltage from the EMG, convert it to a digital signal with an onboard ADC from our ESP32 chip, map that digital value to a corresponding MIDI parameter intensity, and then wrap that parameter in the correct

MIDI communication protocol. The specifics of exactly how this process will be implemented is explained in later sections.

Visual Aid

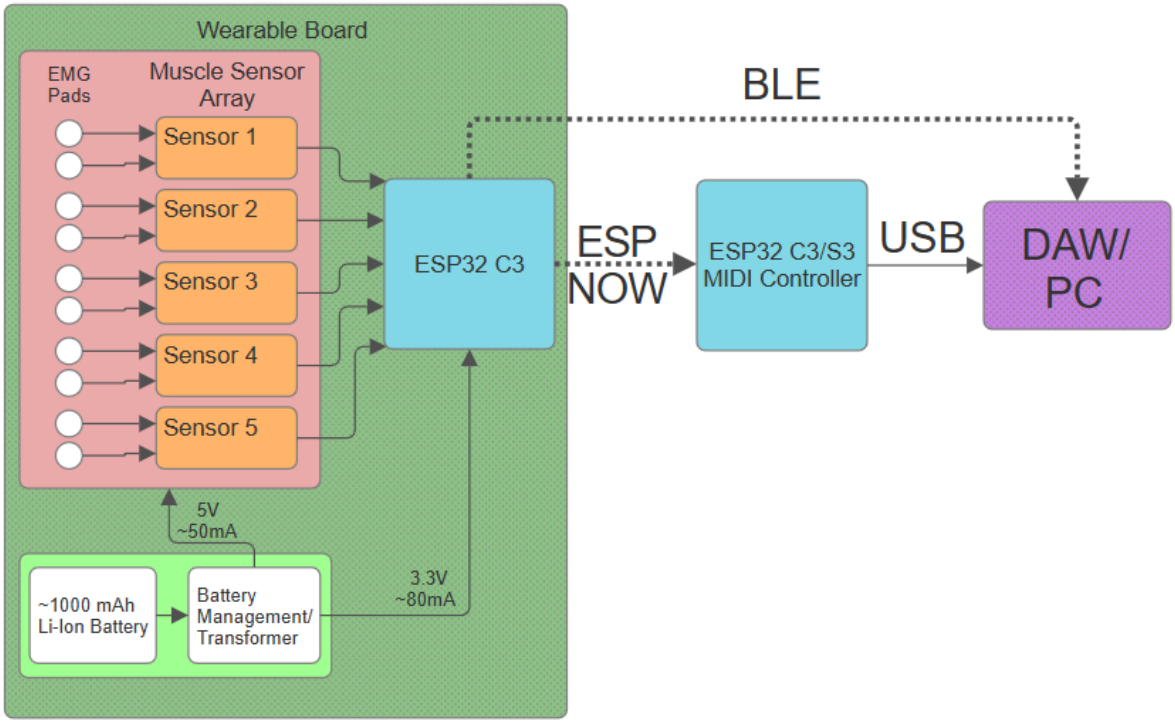


High-level requirements list

- Requirement 1: Trigger one effect (ex: MIDI code 0x90, play note) based on a contraction of any given muscle.
- Requirement 2: Device must be compliant to all communication protocols that we plan to use, including (but not limited to) ESP NOW, USB, and/or Bluetooth.
- Requirement 3: EMG sensors must be tunable to each muscle group. Each muscle group will exhibit different threshold voltages based on the size of the muscle. Therefore, the gain of our sensors must be adjustable either automatically or by the user in order to account for this.

Design

Block Diagram



Subsystem Overviews

Wearable Sensor Array

Overview

The wearable sensor array will be a watch-type design with multiple muscle sensor interfaces aligned on the skin facing side of the device. Each individual sensor will

interface directly with its own channel of the ESP32 ADC. Sensors can either be distributed evenly on the same bracelet or be made into their own individual boards and bracelets, with the ability to connect or disconnect these “modular” sensors to the main ESP32 C3 board.

Requirements

In order to achieve full functionality of this system, we must implement individual muscle sensor units, capable of capturing the muscular contraction signals from a user’s arms and legs. The voltage output of this subsystem must not exceed 3.3V, as this is the maximum allowable voltage of the ESP32 ADC. It must also exceed a voltage of 100mV for a detected muscle pulse, as this is the practical minimum reading of the ESP32 ADC. Because of the ICL7660 voltage inverter and LM324 op-amp, this sensor system needs 5V power to run, which may require usage of separate small batteries for each sensor/array as the ESP32 main board takes 3.3V power. Each sensor draws around 10mA of power to run while powered on, so the batteries chosen must be able to sustain this power draw continuously for at least a few hours.

Tolerance Analysis

Using the 2 EMG pads, each sensor will first determine the differential in voltage at the sites of the pads and amplify the signal using a differential Op-Amp. After this, the signal passes through a second order active high pass filter with a cutoff frequency of ~22.575 Hz and a pass band gain of 1.55, and a second order low pass filter with a cutoff frequency of 482.288 Hz and pass band gain of 1.55. This step filters our raw signal into something more usable, as muscle impulses are known to occur in 0-500 Hz frequency band, with an amplitude of around 0.1 to 1 mV depending on the muscle being measured. After the signal is filtered, and at a voltage of around 10-100 mV, the final tunable opamp is used to scale up the output because different muscles have different sizes, and thus different voltage ranges. It should be noted that this may be too much gain, as the ESP32 can only read analog signals up to 3.3V, so we may opt to use less gain at this stage(lower value for R_{14}). The calculations for this section can be seen below.

$$\frac{47k}{1k} < A_{tuner} < \frac{147k}{1k} \sim 47 < A_{tuner} < 147$$

$$A_{Diff} = \frac{47k}{1k} = 47$$

$$A_{Filters} = 1 + \frac{10k}{18k} = 1.55$$

$$f_{HPF} = \frac{1}{2\pi\sqrt{R_5 R_6 C_1 C_2}} = \frac{1}{2\pi\sqrt{(15*10^3)^2(470*10^{-9})^2}} = 22.575\ Hz$$

$$f_{LPF} = \frac{1}{2\pi\sqrt{R_9 R_{10} C_3 C_4}} = \frac{1}{2\pi\sqrt{(1*10^3)^2(330*10^{-9})^2}} = 482.288\ Hz$$

Wearable ESP32 C3/ Sensor Controller

Overview

The ESP32 C3 will serve as the brains of the sensor array, and will continuously read the input it receives from the sensor array. It should be noted that the ESP32-C3 has 6 separate ADC channels, and the block diagram includes 5 sensors, but we are not planning on necessarily using 5 sensors per ESP32-C3. The amount of sensors placed in each array will ultimately come down to bandwidth of whatever communication protocol we settle on using. In the block diagram, it can be seen that there are two routes from the wearable main board to the DAW/PC. This is intentional, as we will test the advantages and disadvantages of both BLE and ESP-NOW.

Requirements

The wearable ESP32 module must read input continuously from at least one muscle sensor channel and translate these signals into either a MIDI command or a value that can be translated into a MIDI command by the receiver board. Each MIDI command consists of 3 bytes, the first of which is the command, followed by two parameter bytes. Some basic examples of midi commands are 0x80 (Turn note off), 0x90(Turn note on), or 0xC0 (Change sound shaping parameters). Our firmware must either trigger the command (such as note on or off) based on a threshold value, or vary the parameter bytes of the MIDI command such that an effect changes smoothly in intensity or a note varies in frequency depending on contraction. This system must also achieve sufficient data transfer speed and range in wireless communication.

Tolerance Analysis

The speed of ESP-NOW is in the range of 214-512kbps, so we can’t encode a recording quality audio signal from every sensor. For reference, a standard audio signal might be sampled at 22,050 Hz * 16 bits = 352800=352.8 kbps. It is for this reason that we need to be mindful of the sampling rate of each ADC. Using a sampling rate of 1kHz

should be more than sufficient to achieve what we need, giving us a rate of (3 bytes per MIDI command)*(1000 samples/s)=3000 Bytes/s = 24kbps. With long range mode enabled, the ESP-NOW protocol reportedly has a range of around 200 meters.

ESP32-C3/S3 Receiver

Overview

Depending on the wireless setup that we decide on this may or may not be necessary because BLE would not need a receiver board, but is much slower than ESP-NOW. Because of this, we are more likely to choose ESP NOW, which would require a separate receiver board. This board will only be responsible for establishing a connection to one or more sensor controllers and would interface to a PC through USB.

Requirements

This subsystem must act as a properly configured ESP-NOW router, able to handle one or more muscle sensor controllers simultaneously. It must also wrap the MIDI commands such that they comply with USB protocol standards.

Tolerance Analysis

This component will need to be compliant with ESP-NOW, and the feasibility has already been discussed in the earlier section about the other ESP32 board.

Ethics

Our team deeply values ethical behavior, and at every step of the way, we pause and ask ourselves, “Are we doing the right thing?” We regularly consult with both the IEEE and ACM code of ethics, but not only that, we strive to go beyond merely “complying” with them, and instead seek to live to the standard of a higher benchmark, holding the values of fairness and integrity deeply in our hearts.

As the IEEE code of ethics says, our team strives “**to treat all persons fairly and with respect, and to not engage in discrimination based on characteristics such as race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression. (IEEE Code of Ethics 2.7)**” We take steps in order to ensure that we include as many people as we can, regardless of their physical ability. Although our device will be tied to our end user’s movements, it is designed to work anywhere on the body, allowing anyone who can move at least one muscle to use it.

Also taking guidance from the ACM code of ethics, we must ensure that we "**respect the work required to produce new ideas, inventions, creative works, and computing artifacts (ACM Code of Ethics 1.5)"** We value the creative spirit and seek to give credit where credit is due. Our project makes use of many open source projects published online, and we made sure to properly cite them and abide by their licenses. By respecting copyright law, we promote fairness for creators, innovators, and their ideas. Additionally, we used Google's extensive patent database (patents.google.com) to verify that we are not infringing on anyone's ideas.

Another essential but often overlooked part of ethics is to **"respect privacy (ACM Code of Ethics 1.6)"** Our device is designed from the ground up to respect the users privacy. By keeping everything on the device local, we avoid many of the privacy issues involved with cloud services and the Internet of Things. We also use the ESP NOW protocol, which has built-in AES-128 encryption, to ensure user privacy throughout the entire experience.

It is important to preserve user control over their own device. We will use encryption and verification to ensure that only authorized users can communicate with the MIDI receiver. We also take steps to ensure that no foreign signals will interfere with the device while we use it. In order to maintain a culture of transparency, we will also make our source code and process available upon request, to allow our users to make extensions and come up with novel ways to use our device. This preserves user ownership of their own device.

One safety concern is the fact that we are using potentially flammable batteries on our device. If the batteries are improperly handled, they could catch fire and/or explode. We will enclose our batteries in flame resistant materials to reduce the harm they can cause. We will also include a warning of the dangers of flammable batteries in the manual to our device. Our batteries will also be user serviceable, allowing the user to exchange defective batteries for new, safer, and functioning ones.

Our team cares deeply about the environment. Batteries tend to make their way into landfills, where their dangerous chemicals can leak into the ground and possibly a drinking water source. To ensure the proper disposal of these dangerous chemicals, we will include a manual with our device, guiding the user to places where they can safely dispose of their batteries. We also made sure to use interchangeable and user serviceable parts to reduce e-waste, allowing our user to repair or send in their device for repair, instead of them throwing out the whole device when just one part is broken.