**An EMG based cost effective Prosthetic hand**

**Rajdeep Saha, Abhijit Biswas, Saradwat Sen and Sayani Dhali**

Department of Electronics and Communications Engineering, Supreme Knowledge Foundation Group of Institution, 1, Khan Road, Mankundu, DistrictWest Bengal, Hooghly 712139, India

**Abstract** In the era of assistive technology, the need for affordable and accessible prosthetic solutions has been roaming around forever. We introduce the solution by making an inexpensive prosthetic arm controlled by electromyographic (EMG) signals obtained from the upper arm muscles of amputees. This project addresses a major global issue which is, in an estimate over 100 million individuals worldwide require prosthetic limbs due to limb damage or amputation. Unfortunately, a large portion of these individuals (especially those from lower economic backgrounds) face problems accessing due to the high cost of traditional prosthetic options, which can range from Rs70,000 to Rs3.5 lakh for imported prosthetic hands. But this project is Remarkably cost-effective. The overall system costs around a few thousand rupees significantly reducing expenses compared to market alternatives . This project solves the solution by using cost-effective technologies in the most efficient ways. The key component in this prosthetic arm is the Muscle EMG sensor. The noise reduction and the amplification of the Bio-Potential signal is done in the IC of the EMG sensor, which helps in precise signal acquisition. By integrating this sensor into the prosthetic arm design, the component can accurately capture muscle signals and translate them into commands for prosthetic hand movements. The brain of this prosthetic arm is a microcontroller, which serves as the central processing unit for signal processing as Band-Pass Butterworth IIR digital filter and responsiveness. Through confined algorithms and real-time data analysis, the microcontroller reads EMG signals captured by the sensor, enabling full control of the prosthetic hand. This integration of hardware and software makes the project's goal of affordability without compromising performance possible . Beyond the technical aspects, this project solves the topmost prioritized problem which is affordability with efficiency. In third world countries where a large portion of people are suffering through poverty, surgery on amputation is nowhere near possible, let alone installing prosthetics. This cheap but effective alternative which costs around 1-3% of the market's product cost, can be affordable to the mass population with disabilities and enhance their quality of life.

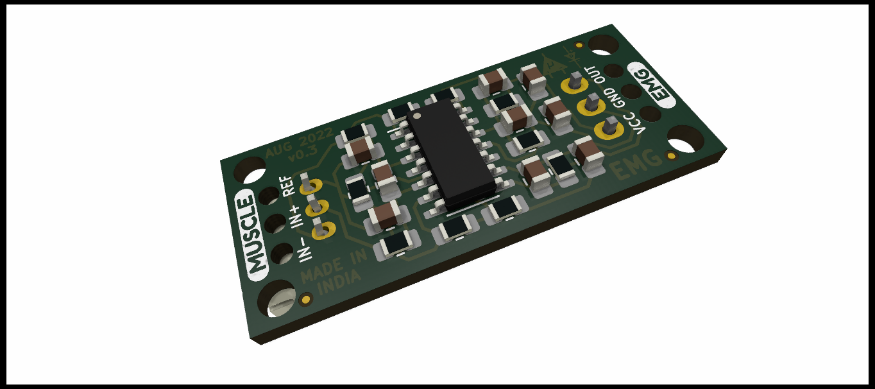
**Keywords:** EMG Signal, Bio-Potential Amplifier, Band-Pass Butterworth IIR digital filter

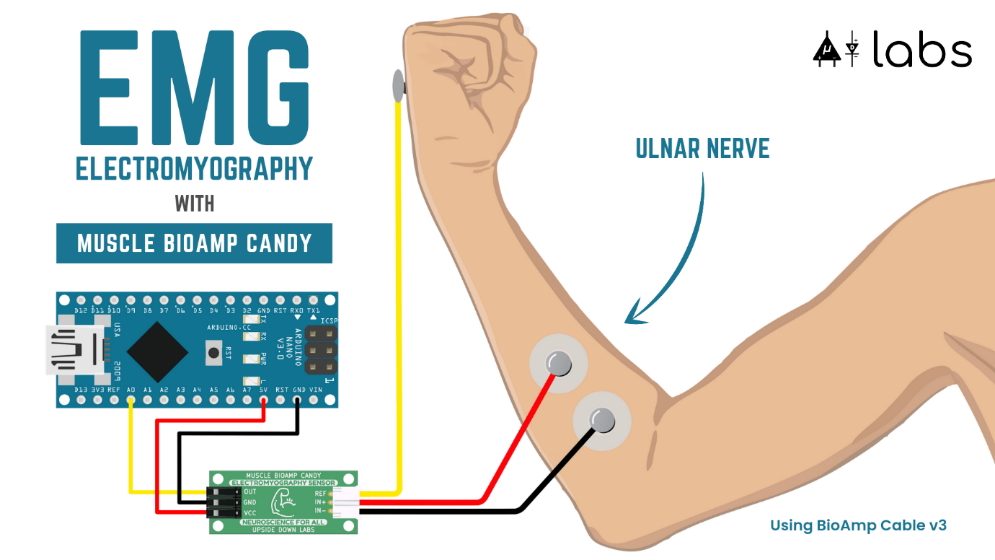
**Muscle-BioAmp-Candy**

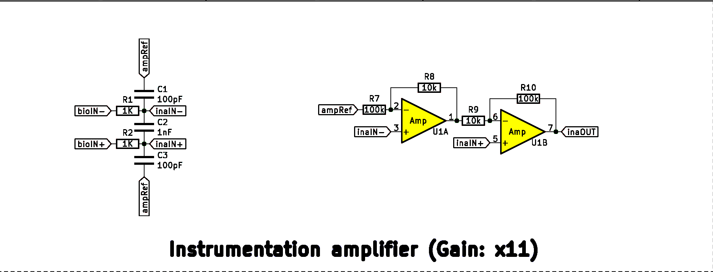
Muscle-BioAmp-Candy is a candy-sized Electro Myography (EMG) sensor designed to capture precise muscle bio-potential signals. With a compact size and functionality, this device can transform the way of monitoring muscle activity.

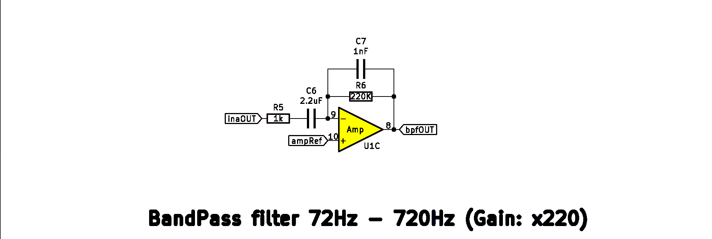
It has a fixed gain of x2420 and a Band Pass filter spanning from 72Hz to 720Hz, Muscle-BioAmp-Candy ensures signal clarity and accuracy.

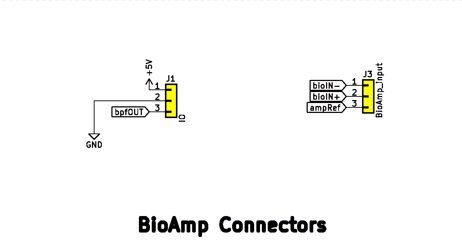
Compatible with standalone ADCs such as the ADS1115 or any microcontroller development board equipped with an ADC, Muscle-BioAmp-Candy offers seamless integration into your existing setup. With its plug-and-play design, recording EMG signals has never been easier.

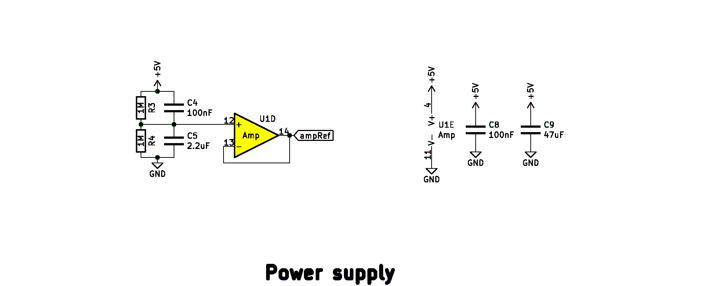




**Circuit Diagram**





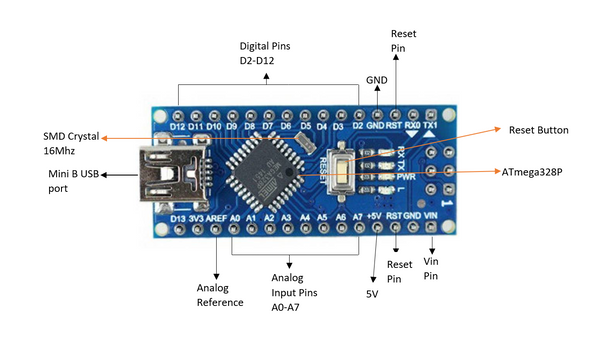


**Reference-**

Muscle-BioAmp\_Candy

<https://github.com/upsidedownlabs/Muscle-BioAmp-Candy>

**Arduino Nano**



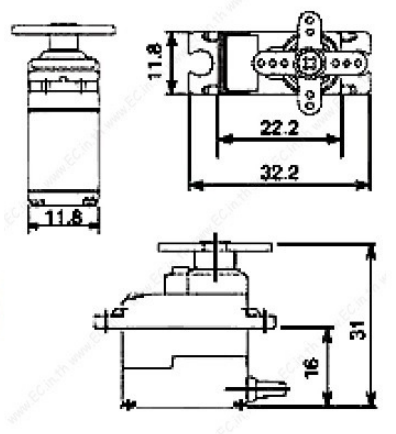
Arduino Nano is a small, breadboard-friendly development board based on an ATmega 328P SMD package microcontroller and offers the same connectivity and specs as Arduino Uno in a small package. In our project, we are using it because the size is the main factor. After all, the microcontroller needs to be fit into the 3d printed hand. We are supplying the power to the board with a Mini B USB port present on it.

**Specifications:**

* It has 22 I/O pins in total of which 14 are Digital (6 are PWM output) and 8 are Analog pins.
* Operating Voltage (Logic Level): 5V.
* Supports Serial, I2C, SPI Communication Protocols.
* Flash memory: 32KB in which 2KB is used by Bootloader
* Clock speed: 16MHz
* DC Current per I/O Pin: 40 mA
* SRAM: 2KB, EEPROM: 1KB
* DC Current per I/O pin: 40mA

Reference <https://quartzcomponents.com/products/arduino-nano-v3-0-soldered-without-cable?_pos=2&_sid=2bf3b65a9&_ss=r>

**SG90 9 g Micro Servo**

****

SG90 9 g Micro Servo is tiny and lightweight with high output power. The servo motors are used for the movement of the fingers in the hand. The servo motors usually provide control over the 180° range. This angular position control is controlled by the PWM technique so by varying its duty cycle we can control the angular position of the motor. This servo motor can lift a maximum of 1.6 kg when suspended at a 1cm distance from the shaft.

Specifications

• Weight: 9 g

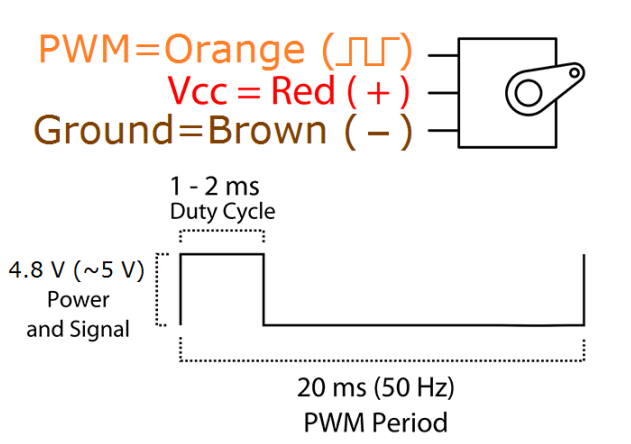
• Dimension: 22.2 x 11.8 x 31 mm approx.

• Stall torque: 1.8 kgf·cm

• Operating speed: 0.1 s/60 degree

• Operating voltage: 4.8 V (~5V)

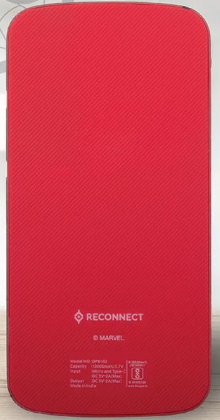
• Dead band width: 10 µs

• Temperature range: 0 ºC – 55 ºC 

Position "0" (1.5ms pulse) is middle, "90" (~2ms pulse) is all the way to the right, and -90" (~1ms pulse) is all the way to the left.

Reference <https://components101.com/sites/default/files/component_datasheet/SG90%20Servo%20Motor%20Datasheet.pdf>

**Power Bank**

****

Here we are using Reconnect Power Hub 10000 mAh Power Bank Series 100. It has a capacity of 10000 mAh, so it can be used for longer intervals. As the microcontroller, servo, and Muscle-BioAmp-Candy draw very little power on a full charge it lasts for about 10hrs.

Specifications and Features

* Over Charge Protection, Over Discharge Protection
* Over Current Protection, Short Circuit Protection
* Input: 5V 2A(max)
* Type C And Micro USB Dual Input Port
* USB A 5V/2A Output Port
* BIS Certified

Reference <https://www.jiomart.com/p/electronics/reconnect-marvel-spiderman-10000mah-powerbank-li-po-battery-1-usbb-output-port-micro-usb-type-c-dual-input-bis-approved-mobile-accessories-dpb102-sm/590041746>

**Algorithm**

1. **Initialization:**
   * Set up the system parameters including the header files, sampling rate, baud rate, and pin configurations.
   * Initialize the communication interfaces (e.g., Serial) for monitoring and debugging purposes.
2. **Setup:**
   * Attach the servo motors to their respective pins and set their initial positions.
   * Configure the EMG sensor input pin and any auxiliary pins required for system operation.
   * Define the threshold voltage
3. **Main Loop:**
   * Continuously sample the EMG signal at the defined sampling rate.
   * Apply a band-pass Butterworth IIR digital filter to the raw EMG signal to extract relevant muscle activity within the desired frequency range.
   * Compute the EMG envelope using an envelope detection algorithm to estimate the magnitude of muscle activation.
4. **Gesture Recognition:**
   * Compare the normalized envelope value with a predefined threshold to determine muscle activation and gesture recognition.
   * Implement a hysteresis mechanism to prevent rapid toggling of the arm due to noise or minor fluctuations in muscle activity.
   * Define specific thresholds for opening and closing gestures based on individual user characteristics and preferences.
5. **Servo Control:**
   * Based on the detected gesture:
     + If the muscle activation exceeds the closing threshold:
       - Close the arm by rotating the servo motors to the predefined closed position.
     + If the muscle activation falls below the opening threshold:
       - Open the arm by rotating the servo motors to the predefined open position.
   * Implement a gesture delay to prevent rapid and unintended toggling of the claw in response to minor fluctuations in muscle activity.

**Reference Github of Upsidedown Labs**

**Algorithm of EMG Band Pass Filter**

Algorithm for the Band-Pass Butterworth IIR digital filter:

1. **Initialization**:
   * Initialize the state variables **z1** and **z2** for each filter section to zero.
2. **Filtering Process**:
   * For each input sample:
     + For each filter section:
       - Calculate the intermediate variable **x** using the difference equation of a second-order IIR filter.
       - Update the output using the calculated **x** value and the previous state variables.
       - Update the state variables **z1** and **z2** for the next iteration.
3. **Output**:
   * Return the filtered output.

Here's a breakdown of the steps within the filtering process:

* For each filter section:
  1. Calculate the intermediate variable **x** using the difference equation:

x = input - a1 \* z1 - a2 \* z2

where **input** is the current input sample, **z1,** and **z2** are the previous state variables, and **a1** and **a2** are the filter coefficients.

* 1. Update the output using the calculated **x** value and the previous state variables:

output = b0 \* x + b1 \* z1 + b2 \* z2

where **b0**, **b1**, and **b2** are the filter coefficients for the output.

* 1. Update the state variables **z1** and **z2** for the next iteration:

z2 = z1 z1 = x

Repeating these steps for each input sample, obtained the filtered output of the Band-Pass Butterworth IIR digital filter.

**General Difference Equation**

The code implements a Band-Pass Butterworth IIR digital filter using second-order sections (biquads). Each biquad represents a second-order IIR filter section. Let's break down the mathematical expression for each biquad.

The general difference equation for a second-order IIR filter is:

*y*[*n*]=*b*0​*x*[*n*]+*b*1​*x*[*n*−1]+*b*2​*x*[*n*−2]−*a*1​*y*[*n*−1]−*a*2​*y*[*n*−2]

Where:

* *y*[*n*] is the output at time *n*
* *x*[*n*] is the input at time *n*
* *x*[*n*−1] and *x*[*n*−2] are the previous input samples
* *y*[*n*−1] and *y*[*n*−2] are the previous output samples
* *b*0​,*b*1​,*b*2​ are the feedforward (numerator) coefficients
* *a*1​,*a*2​ are the feedback (denominator) coefficients

Let's express each biquad in the code as difference equations:

**First Biquad:**

z1\_1[n] = x[n] - 0.05159732 \* z1\_1[n-1] - 0.36347401 \* z2\_1[n-1] z2\_1[n] = z1\_1[n-1] y\_1[n] = 0.01856301 \* z1\_1[n] + 0.03712602 \* z2\_1[n] + 0.01856301 \* z2\_1[n-1]

**Second Biquad:**

z1\_2[n] = y\_1[n] - (-0.53945795 \* z1\_2[n-1] - 0.39764934 \* z2\_2[n-1]) z2\_2[n] = z1\_2[n-1] y\_2[n] = 1.00000000 \* z1\_2[n] - 2.00000000 \* z1\_2[n-1] + 1.00000000 \* z2\_2[n-1]

**Third Biquad:**

z1\_3[n] = y\_2[n] - (0.47319594 \* z1\_3[n-1] - 0.70744137 \* z2\_3[n-1]) z2\_3[n] = z1\_3[n-1] y\_3[n] = 1.00000000 \* z1\_3[n] + 2.00000000 \* z1\_3[n-1] + 1.00000000 \* z2\_3[n-1]

**Fourth Biquad:**

z1\_4[n] = y\_3[n] - (-1.00211112 \* z1\_4[n-1] - 0.74520226 \* z2\_4[n-1]) z2\_4[n] = z1\_4[n-1] y\_4[n] = 1.00000000 \* z1\_4[n] - 2.00000000 \* z1\_4[n-1] + 1.00000000 \* z2\_4[n-1]

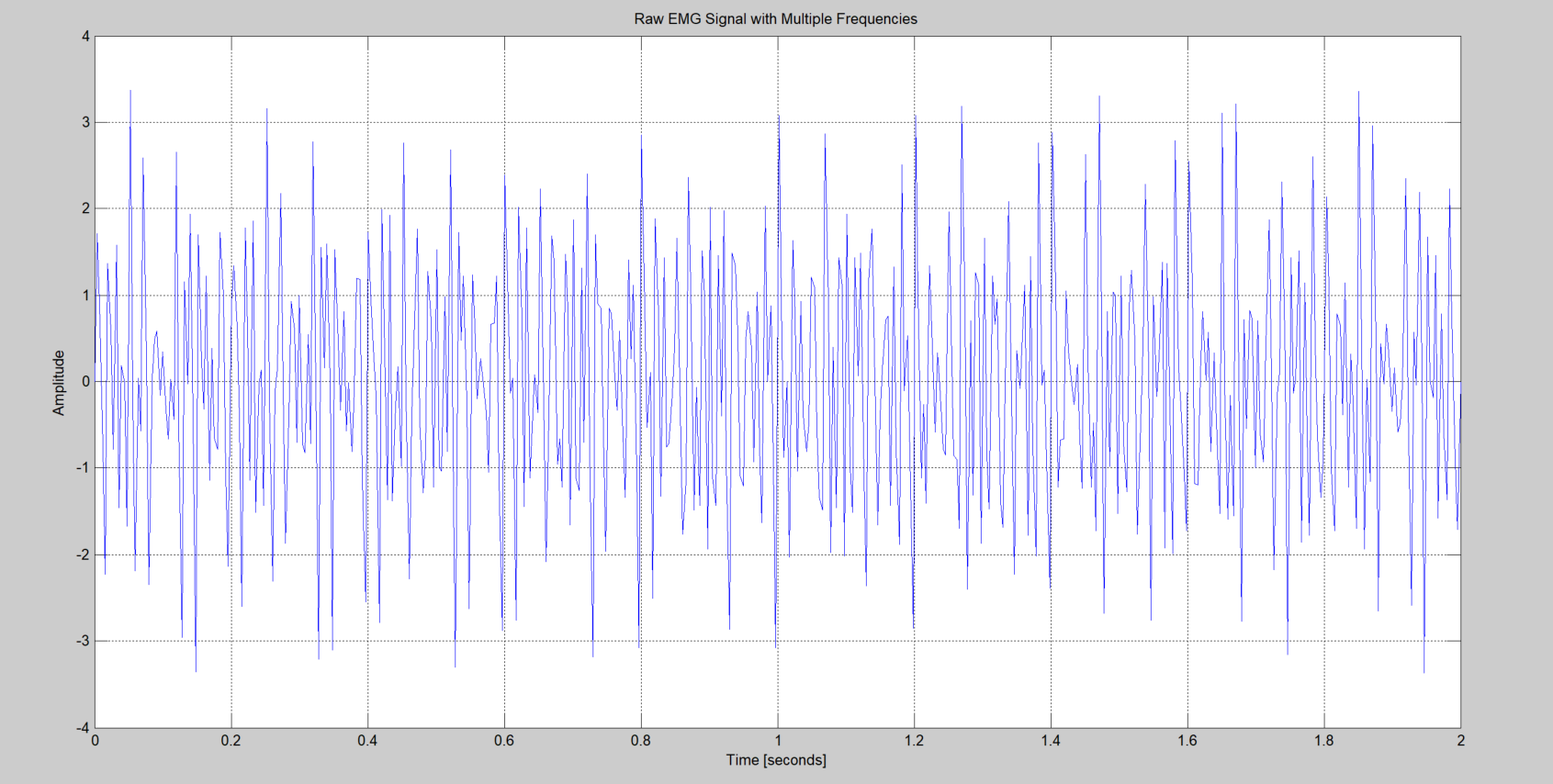
Where:

* *z*1*i*​[*n*] and *z*2*i*​[*n*] are the state variables for the *i*-th biquad at time *n*
* *x*[*n*] is the input at time *n*
* *yi*​[*n*] is the output of the *i*-th biquad at time *n*

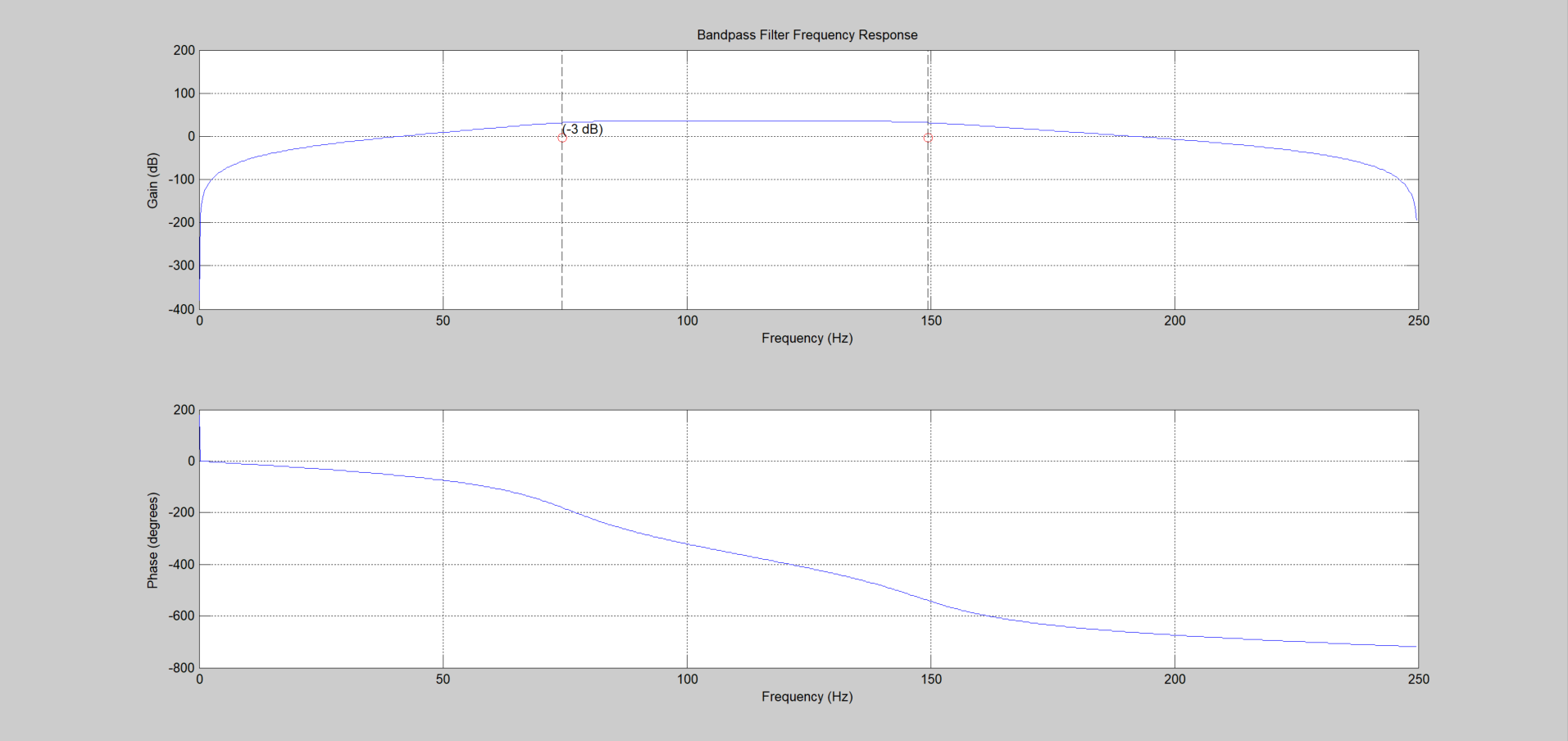
This set of equations describes the mathematical expression for the given Band-Pass Butterworth IIR digital filter implemented in the provided code. Each biquad contributes to the overall filter response, and the output of one biquad serves as the input to the next biquad in the chain.

**Simulation of BPF in MATLAB**

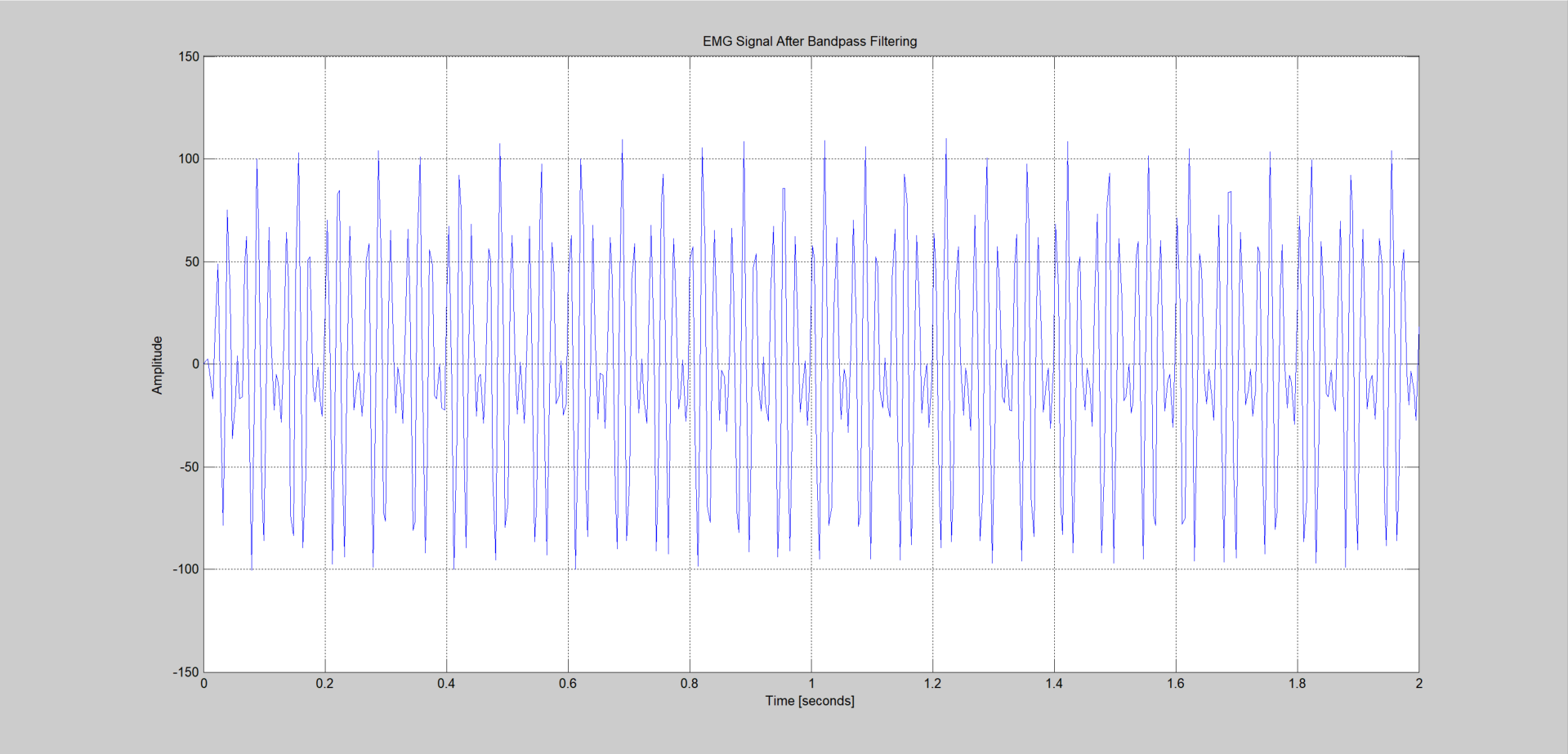
Raw EMG signal with multiple frequencies of 45Hz, 60Hz, 90Hz, 100Hz and 160Hz



Frequency response of the filter with -3db point



Filtered EMG Signal after Band Pass Filter consisting of frequencies 90Hz and 100Hz



**Reference:**

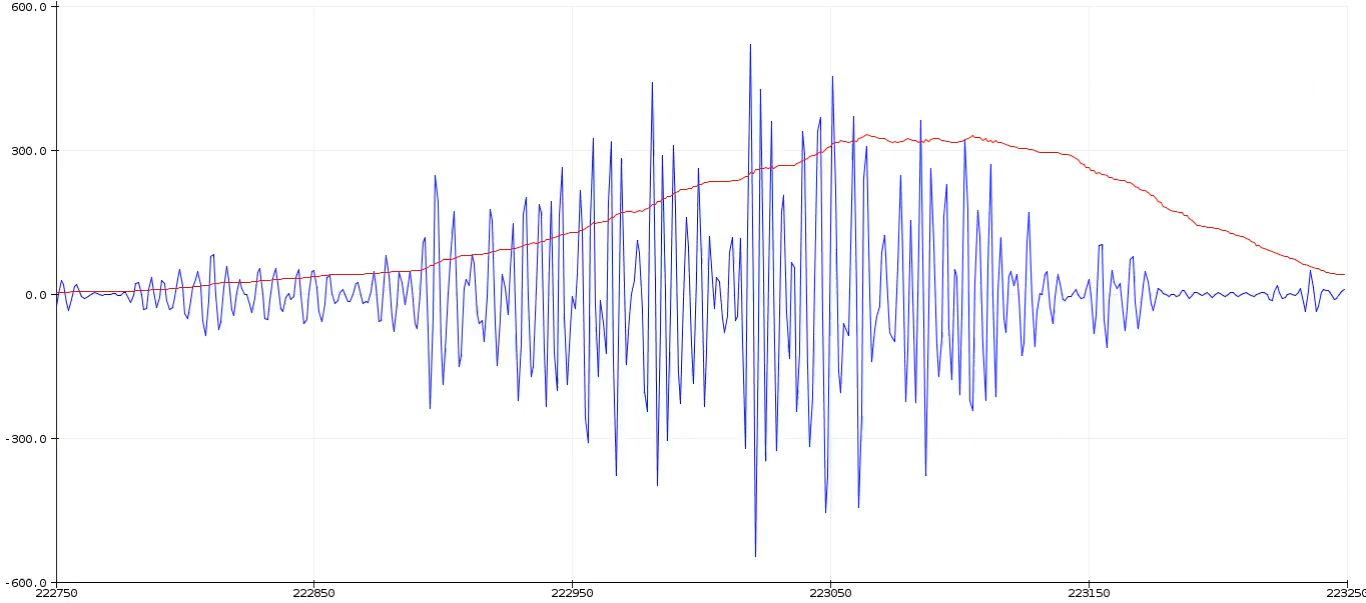
<https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html>

<https://courses.ideate.cmu.edu/16-223/f2020/Arduino/FilterDemos/filter_gen.py>

**Algorithm for Envelope Detection-**

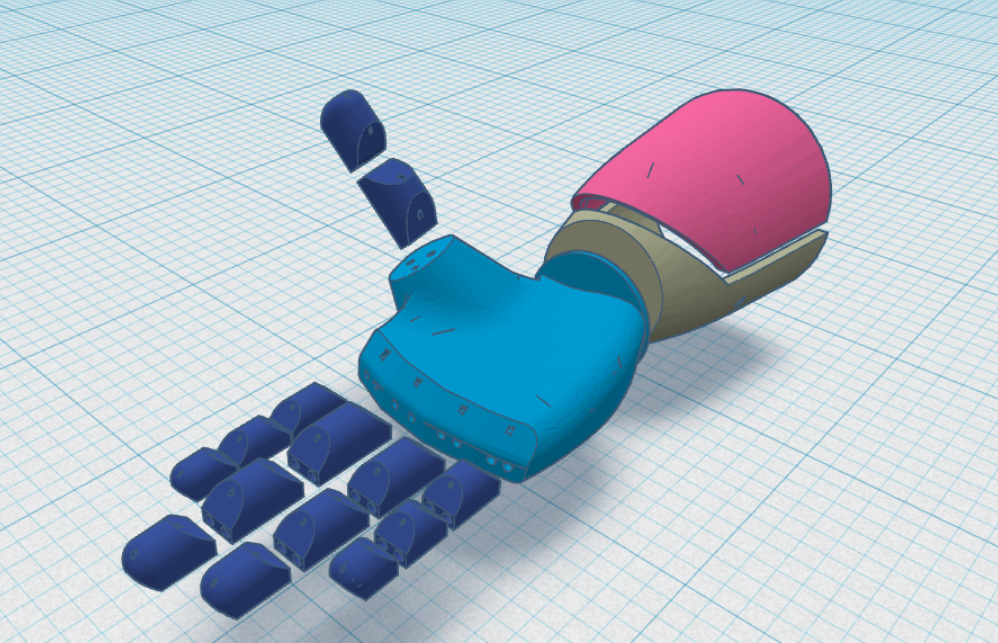
1. Subtract the previous EMG signal value from the sum.
2. Add the absolute value of the current EMG signal to the sum.
3. Store the absolute value of the current EMG signal in the circular buffer at the current index.
4. Update the data index to point to the next position in the circular buffer, wrapping around to the beginning if necessary.
5. Compute the average of the EMG signal values in the circular buffer by dividing the sum by the buffer size.
6. Multiply the average by 2 to scale the envelope signal.
7. Return the computed envelope signal.

**EMG Signal After Filter With The Detected Envelope**



**3D Printed Hand**

**CAD Files:**

****

**After assembling all the parts of the hand**