MONASH University
Engineering

ECE4081 – Medical Instrumentation
Final Project Report
An EMG Video Game Controller

Siddhant Tandon (28677528)

Contents

1.0 Aim

Video games are enjoyed by a significant proportion of the population around the world. They form a popular and relaxing hobby that has also been linked to a multitude of social and cognitive benefits [1].

Presently, however, video games are largely inaccessible to individuals that have limited or no ability to use traditional controllers or keyboard/mouse inputs. This project seeks to investigate the potential for using EMG as a means of controlling a video game. EMG is well suited to this use case, as it is:

- **Versatile-** as the device measures muscle movements, it can be placed on a variety of limbs/muscle groups.
- **Fast-** Response times with EMG are still quite rapid, restricted by how fast the patient can control the muscle group being measured.
- **Non-invasive-** The device operates entirely outside the user's body, with only the need to stick electrodes on the skin over the muscle group being monitored.
- **Inexpensive-** The parts used in this experiment can all be purchased for under $200.

The developed device maps movements of a subject's bicep directly to a keyboard input, which is used to control three games: Flappy Bird, Pong, and Dinosaur. This toy example is designed to illustrate how feasible this technology is, and spur further development of such technologies to make gaming more accessible to all.

2.0 System Design

In this section, we will explore the overall system architecture of the developed medical device, with detailed calculation and discussion aimed at exploring the reasoning behind certain design choices.

2.1 Block Diagram:

A block diagram illustrating the overall system architecture is included in Figure 1.

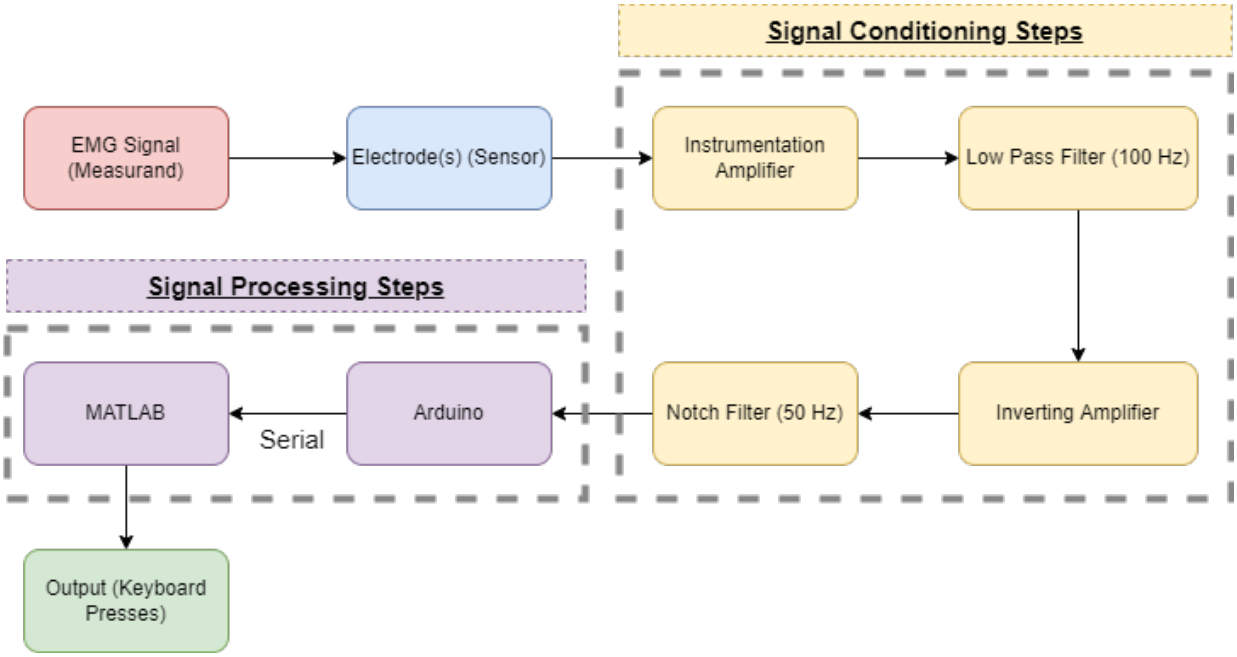

Figure 1 – Block diagram of overall system.

2.2 Schematic:

The circuit design schematic is also included for reference, in Figure 2.

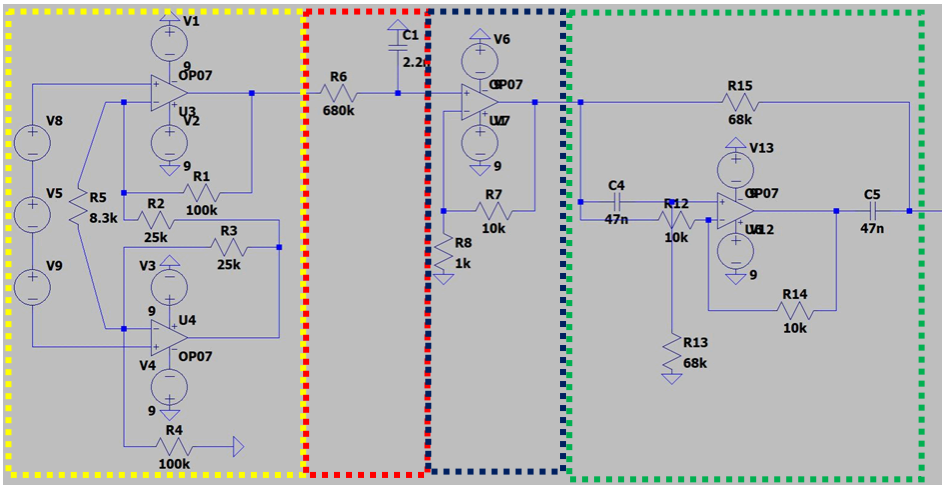

Figure 2 – Schematic of the overall circuit. The components in the yellow dotted box correspond with the instrumental amplifier, the red box bounds the low pass filter, the blue box bounds the amplifier, and lastly, the green box bounds the notch filter.

2.3 Calculations & Design Explanations:

This section will detail the calculations that informed decisions when constructing the various circuit subsystems in the instrument, as well as the design process and decisions behind the design of each subsystem.

2.3.1 Filtration Circuitry:

As can be seen from Figure 1, the circuit is composed of two filtration sub-circuits: a low pass filter aimed at removing signals above 100 Hz, and an active notch filter, designed to remove noise signals of 50 Hz. The notch filter was employed to block mains power noise, which was observed to provide a noise signal of roughly 3mV peak-to-peak amplitude, while the low pass filter was designed with a cut-off frequency at roughly 100Hz- to cut of some high frequency noise signals also observed in the raw signal, with amplitudes of roughly 4mV amplitude. The mains noise only became visible after the low pass filter once some of the high frequency noise was removed from the signal.

The calculations used to determine the component values when designing both above filters are shown below.

Low pass filter:

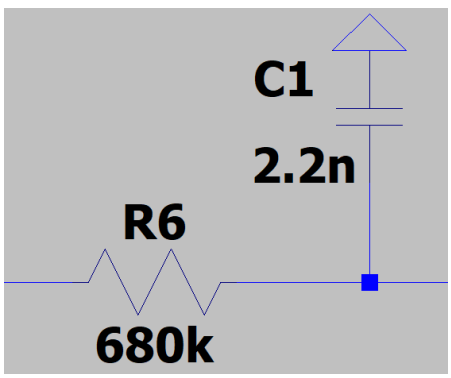

Figure 3- Passive low pass filter schematic (reproduced from [2])

$$f_{cut-off} = \frac{1}{2\pi R_6 C_1}$$

$$100 = \frac{1}{2\pi \times R_6 \times 2.2 * 10^{-9}}$$

$$R_6 = 723\,430\Omega$$

The closest resistor to this value was 680 kilo-ohms. Using this resistor value changes the cut-off frequency to 106.4 Hz instead of 100, but for the purposes of this prototype this is sufficient accuracy. A 2.2nF capacitance was fixed and used as a constraint in the design of this circuit to keep the capacitance in the filter low. As will eb discussed in the following segment, large capacitance values had negatively impacted the fidelity of the signal, by introducing substantial charging/discharging waveforms into the signal, which subsequently distorted the desired signal.

Notch Filter

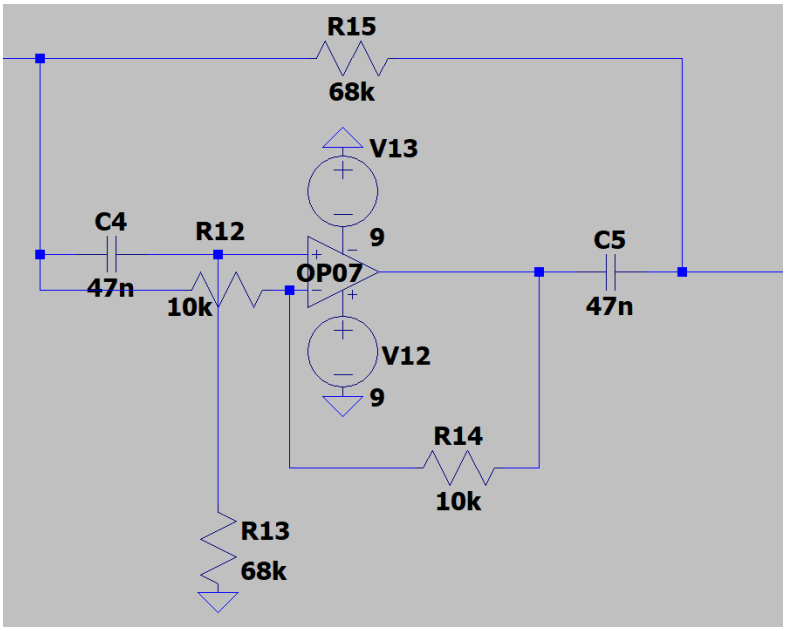

Figure 4- Notch filter schematic

The same equation as above can be used for both lowpass and high pass components in the notch filter. As we wish to filter 50 Hz frequencies with this notch filter, the high-pass and low-pass cut-off frequency is both 50 Hz, and accordingly the high pass and low pass capacitance/resistance values are the same:

$$f_{cut-off\ low\ pass} = f_{cut-off\ high\ pass} = f = 50 = \frac{1}{2\pi \times 68{,}000 \times C}$$

$$C_{HP\ \&\ LP} = 46.8\,nF$$

A fixed 10K resistance was used to find a matching capacitance value. Crucially, the capacitance was kept in the nano-farad range to prevent charging/discharging delay causing problems on the responsiveness of the circuit. The closest capacitance value to the calculated value available was 47nF- which was the component chosen (using this capacitor changes the stopping frequency to

49.8Hz, which is close enough to 50Hz to still filter the mains noise (as can be seen in Figure 17- the frequency response of this filter).

When designing this sub-system, the initial choice was to utilise a passive notch filter and have a subsequent amplification stage. This, however, was not the eventually selected approach, on account of the incredibly large capacitance values required to build this circuit (shown in Figure 5). When implemented the circuit displayed long capacitor charge discharge cycles, which masked the EMG signal being measured. The active alternative mitigated this somewhat as the capacitor values required were orders of magnitude lower.

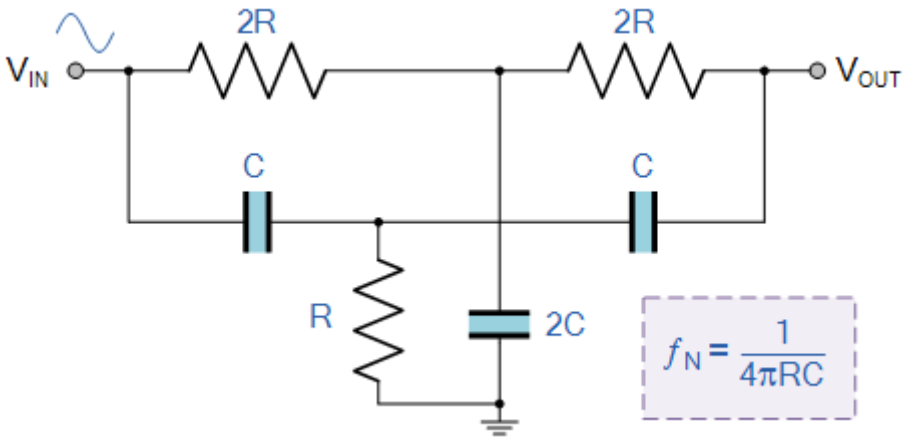

Figure 5- Passive notch pass filter schematic (reproduced from [2])

2.3.2 Amplification Circuit Calculations:

The circuit utilised two gain stages, as seen in Section 2.1. The first was an instrumentation amplifier, which was used to directly boost the base signal coming directly from the electrodes (the "differential" signal was calculated as the difference between the signal from the reference node- connected to the hip- and the signal from the bicep electrodes). This amplification stage was tuned to provide a gain value of around 29, as can be seen by the calculations below. The gain formula for this circuit was supplied by the data sheet [3]. Figure 4 illustrates the circuit schematic for this segment, noting R_G from the equation is equal to R_5 in the schematic.

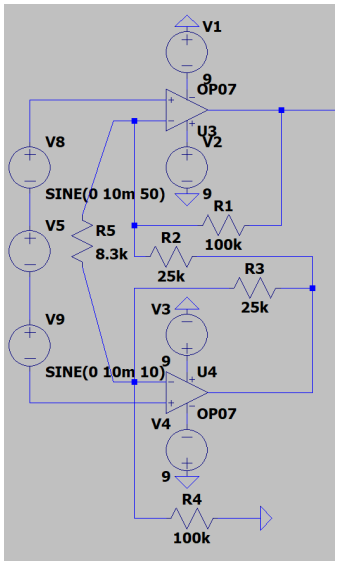

Figure 6- Instrumentation amplifier schematic

$$G = 5 + \frac{200000}{R_G}$$

$$G = 5 + \frac{200000}{8300} = 29$$

Initially, the gain for this stage was set high- at close to 200 (R_5 was set to 1K ohm resistance), to try boost the roughly 10mV EMG signal to around 2 volts. This caused the op-amp to saturate, with a clipped waveform and slight oscillation in the signal. For this reason, the gain was subsequently lowered by roughly a factor of 10 to prevent this unwanted behaviour.

A second gain stage also exists just before the notch filtration circuit. An inverting operational amplifier circuit design was employed for this part of the circuit, with the schematic of this segment provided in Figure 6:

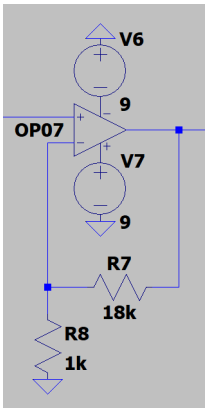

Figure 7- Schematic of the inverting op amp used to boost the signal amplitude.

The gain of this subcircuit is calculated below:

$$G = -\frac{R_7}{R_8} = -18$$

Accordingly, the overall gain of the circuit based on these two gain stages is given by -18 x 29 = -522. As discussed in section 3, an idealised muscle contraction can be estimated to provide stimulation of amplitude 10mVpp. Hence, assuming losses in the circuit are negligible, we would expect an output of the circuit of roughly a maximum of 5.2Vpp. As we will see later, this is not the case, our actual signal reaches at most roughly 2Vpp, due to the likely smaller inputs (10mV is a highly idealised model of standard EMG amplitude) as well as voltage drops across the circuit.

3.0 Signal Acquisition and Processing

This section will describe the methodology used to acquire and process the signals, in order to convert the filtered and amplified EMG- produced by the circuit- into usable outputs to control games. Arduino was used to read the voltage output of the circuit, and MATLAB R2020 was used as the software of choice, to process the signals, and control the selected games.

3.1 Signal Acquisition

The EMG was conducted over the bicep of the user. Three leads were connected- one on the inside of the elbow, the second midway up the same bicep (the placement at halfway along the muscle was selected for ergonomic reasons, as it made it easier to connect, and move around with the leads attached), and a third “reference’ electrode was attached to the hip on the other side of the body, to provide sufficient separation for the other electrodes, and accordingly provide an appropriate reference potential.

The filtered output of the EMG circuit was supplied directly into an Arduino, using the analogue voltage reading of the Arduino devices. This data was read by a simple script (see Appendix 2), which subsequently printed the data to a connected computer’s serial COM port.

Crucially, the electrodes had to be placed in a defined orientation based on the circuit constructed. The red electrode had to be placed on the bicep, with the white electrode on the inside of the elbow (with black always being reference). This was critical to ensure the Arduino was provided with a positive voltage signal. If a negative voltage was applied, the Arduino simply dropped the effected pin to ground, and accordingly no signal was detected.

The serial communication library provided with MATLAB [4] was used to connect with the Arduino over its serial port, and stream the data produced by the device. This data was subsequently processed to produce keyboard inputs, that are used to control the games.

The keyboard outputs were programmed using the Java AWT class, which can be used to emulate user keyboard presses in the operating system [5].

3.2 Signal Processing

The signal produced by the hardware suffered from two key defects, that made it difficult to work with in its raw form:

1. It was observed that the absolute signal produced by the circuit varied heavily depending on the condition of the subject the electrodes were attached too. Indeed, at different times of day, in different weather, the absolute signal varied dramatically from baselines of 3.5V down to 2.5V at other times. This can be attributed to varying skin impedance at different times, as factors like sweat and hydration can impact the signal produced.
2. The noise signature of the signal also varied heavily on the environmental noise signatures experimentation was conducted under. When taken to a student’s bedroom, a strong random noise signal was detected, which was absent in the lab environment.

These two complications meant that just setting static thresholds in the program was generally highly inaccurate, and frequently resulted in random misidentification of EMG tensing signals.

A solution developed by the research team was to use a “calibration” period at the start of every run, and the usage of variance to set thresholds that would detect muscle contractions in the EMG. When the program is initialised, the user is asked to keep the connected limb still for a period of roughly 10 seconds. Over this period, the quiescent noise levels can be observed. This data is used to compute a measure of the variability of the signal, by computing the standard deviation of the entire data series. Subsequently, this was multiplied by a scaling factor of 15, to create the thresholds used when determining whether an input was detected. The selection of the scaling factor of 15 was made following extensive fine-tuning and experimentation, to determine the value that stopped false triggers, whilst also not decreasing sensitivity to the point the patient needed to tense too hard to have their input registered.

The above calibration methodology served as an effective solution to point 2, and point 1 initially, as it reset the baseline, signal value each time the device was switched on.

This was not enough, however, as though the noise signature stayed constant over time in the same environment- with extended use, the baseline voltage could change, as the patient skin impedance changed (possibly due to increased or decreased perspiration). Accordingly, a recalibration of the mean signal value, about which the thresholds would be applied, was conducted periodically, with best results seen when this process was completed roughly every 5 seconds.

3.3 Output

The keyboard presses generated by the EMG instrument were used to control three online games: Flappy Bird [6], Pong [7] & The Dinosaur Chrome Browser game [8]. The user could specify which game they wish to play using a dialogue box generated with a menu programmed in MATLAB, as seen in Figure 8.

Figure 8- UI developed to select game the signal to map to

The dinosaur game and Flappy Bird simply mapped EMG pulses to spacebar presses, as the methodology to control the game. Pong, on the other hand, used EMG pulses to invert the direction of travel of the pad, by toggling between the up and down arrow keys with each EMG input.

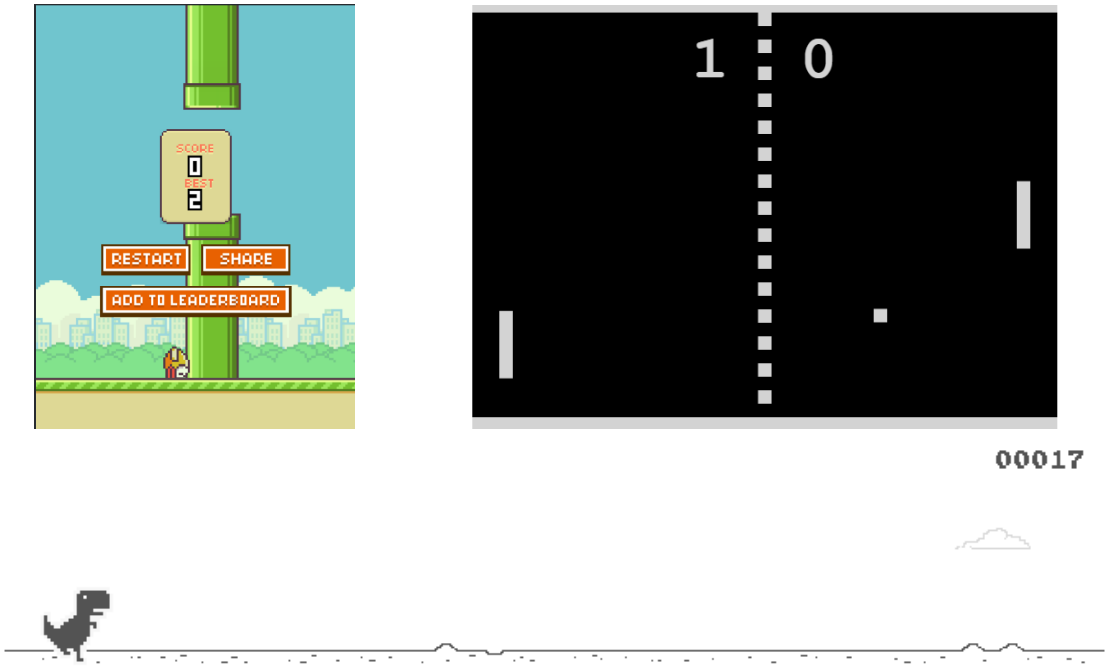

Figure 9- The game playable by the device (clockwise from top left- Flappy Bird [6], Pong [7] and Dinosaur [8].

4.0 Circuit Performance:

In this section, a deeper analysis of the performance characteristics of the circuit will be undertaken, including a description of the input and output simulated signals, and how they compared with calculated and real-world data, as well as the frequency response of the filter circuits.

4.1 Input Signal

For the purposes of the simulation, an input signal of the morphology shown in Figure 10 was applied. This signal approximates a muscle EMG input- which the literature suggests is usually approximated by a 10 mVpp signal [9]. A noise signal of 1kHz and amplitude 10mVpp was then added to this, alongside a 50Hz noise signal also with an amplitude of 10mVpp, to test the performance of the filters later in the circuit.

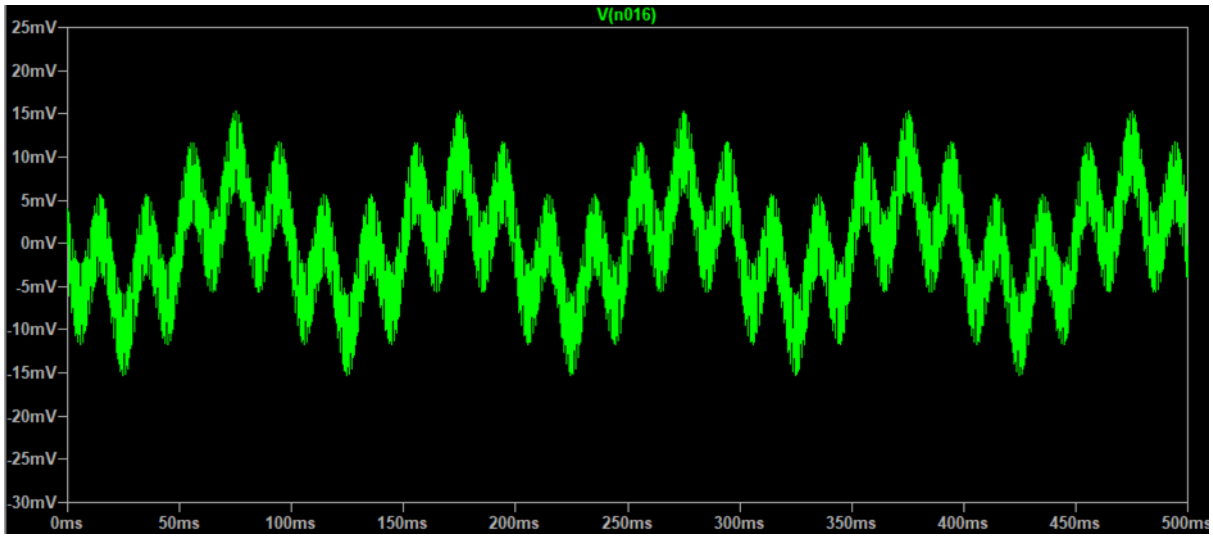

Figure 10- The input signal used for the simulation

4.1.1 Instrumentation Amplifier

The simulated output of the instrumentation amplifier is then provided in Figure 11.

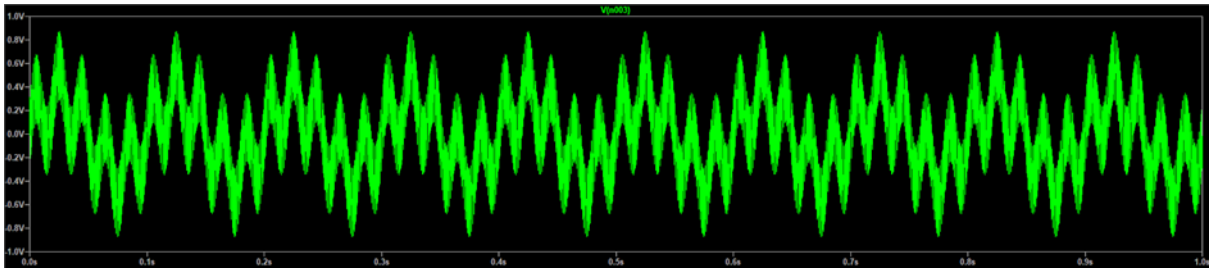

Figure 11- Simulated output of instrumental amplifier

Notably, the signal's amplitude has been boosted by a factor of roughly 56.7. This does not match the calculated gain of the instrument amplifier- which was expected to be roughly 29. A possible explanation for this is that idealised op-amp models were used that do not meet the exact specifications of the amplifiers used within the actual IC, which may have resulted in the gain difference.

Furthermore, interestingly, when conducted in the laboratory, signals with an amplitude of around 150 mVpp detected at the output of this op-amp. Assuming the calculated gain is correct, it can be assumed that this means the signals detected by the electrodes were likely to have an amplitude of

5mVpp- significantly lower than the predicted the input of 10mVpp asserted by [9]. This is likely because this source would have assumed an intramuscular EMG would be conducted to get these values (the paper does not specify). As this device uses a surface EMG, the smaller detected signals are understandable, due to the extra impedance of the skin.

4.2 Low Pass Filter

The output of the low pass filter is shown in Figure 12

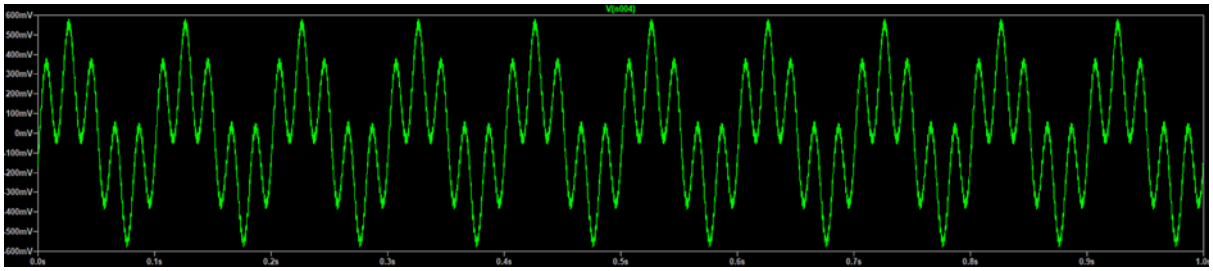

Figure 12- Simulated output of low pass filter

As expected, the high frequency noise is removed by the low pass filter, and a cleaner signal can be seen as a result. The 50Hz noise appears untouched, while the amplitude is also attenuated as a result of the voltage drop over the resistor in the circuit.

The frequency characteristics are highlighted in the bode plot- Figure 13.

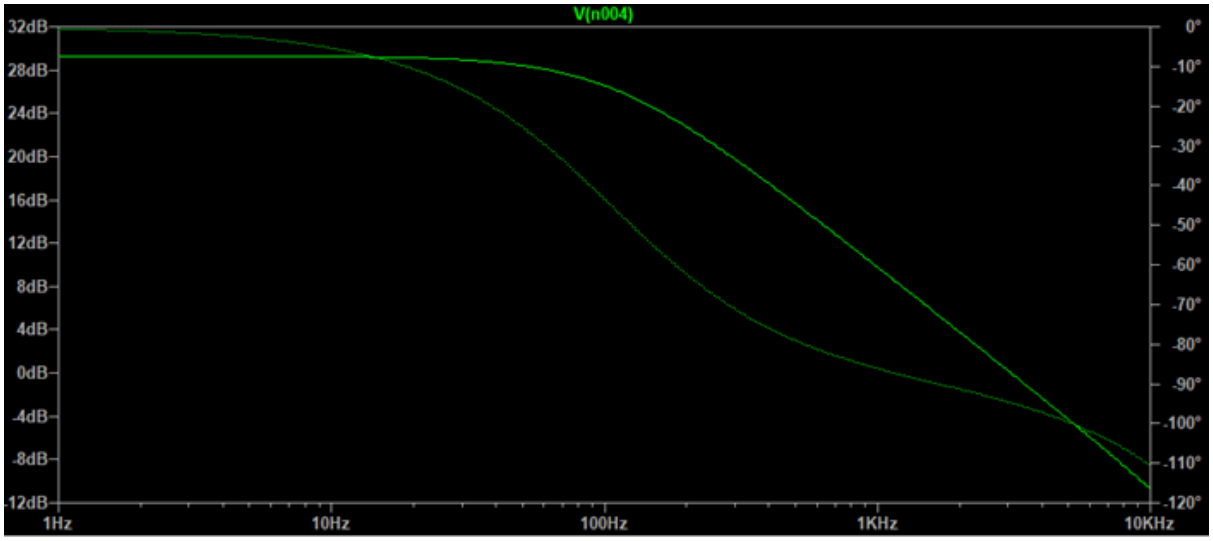

Figure 13- Simulated bode plot of low pass filter (dark line is magnitude, light line is phase)

As expected, the cut-off frequency can be seen at roughly 100 Hz, with a gradual decline in magnitude after this point.

4.3 Inverting Amplifier

Figure 14 depicts the simulated output of the inverting amplifier.

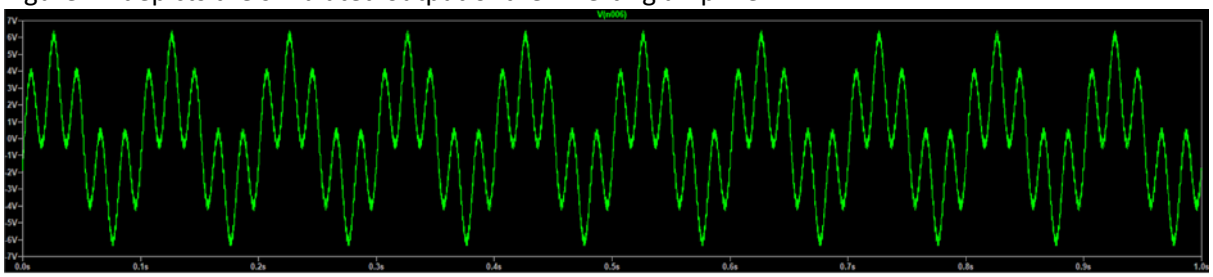

Figure 14- Simulated output from inverting amplifier.

This subsystem now amplifies the amplitude of the signal coming out of the low pass filter.

Earlier in this report, in Section 2, this amplifier was calculated to produce a gain of roughly -18. But based on the simulated input to this subsystem (a circuit with 1.2Vpp signal) the output here reflects a gain of roughly 10.8. This difference, once again, is likely a product of the different op-amp model used in simulation.

In the real world, this signal exhibited roughly a peak-to-peak amplitude of roughly 2.2 Volts. This is roughly the same as the expected value (as we would assume a 2.9V output given the roughly 100mV output of the instrumental amplifier, excluding the voltage drop in the filter circuit).

4.4 Notch Filter

The simulated output from the notch filter is provided in Figure 15.

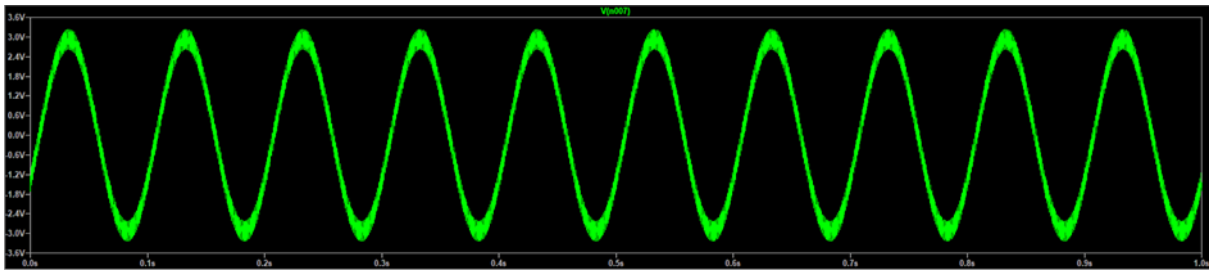

Figure 15- Simulated output from notch filter

As expected, the 50 Hz noise is now gone. The signal also experiences a significant voltage drop from the output of the inverting amplifier, due to voltage drops over the circuitry.

This signal is also the output of the circuit-side of this instrument. In the real world, a signal with peak-to-peak voltages of around 2V was observed, as show in Figure 16. This is roughly 1V off the output presented by the simulations. This variance between the two values can be explained by the usage of idealized models for the op-amps in the earlier subparts, as well as the lower input from the electrodes themselves.

Assuming the computation, in Section 2.5.2 was correct, and the actual input into the electrodes is roughly 5mVpp, we would find that- assuming there was no other voltage drop in the circuit- the expected output has an amplitude of roughly 2.699Vpp. This is within reason when compared to the actual value, as this amplitude loss could realistically be a product of the voltage drop across the circuits.

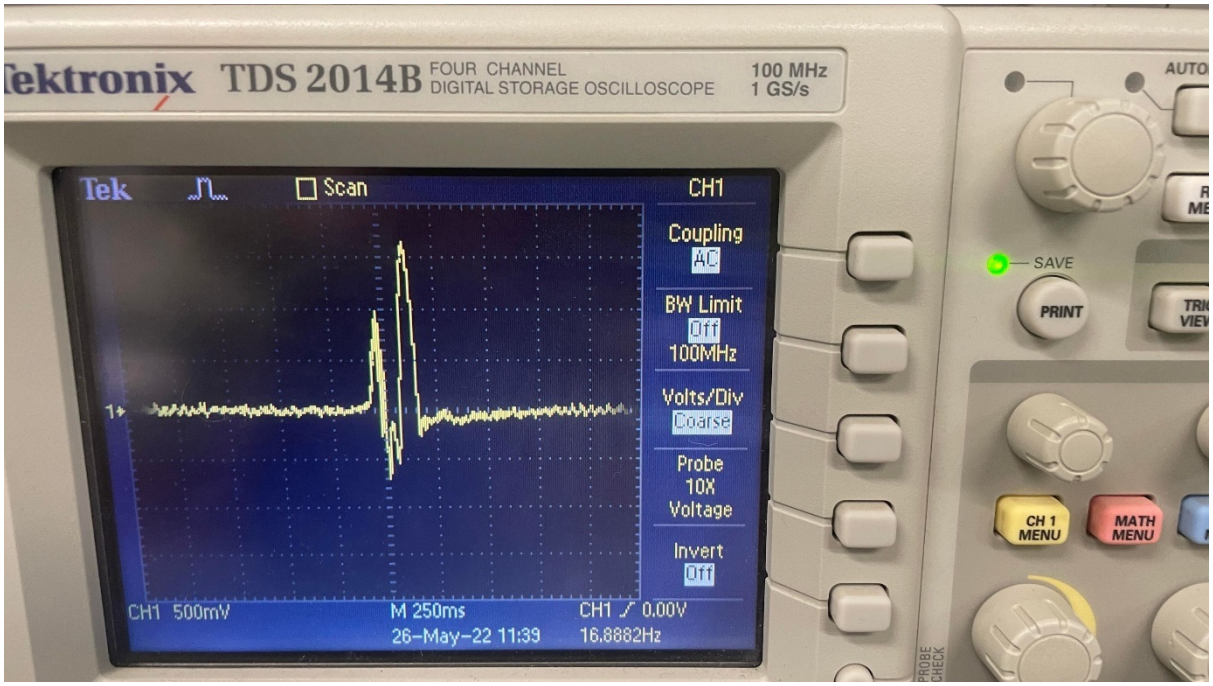

Figure 16- Output from notch filter

As can be seen from figure 16, the output is also quite clean, there appears to be minimal noise from mains power or other sources obfuscating the EMG waveform.

The frequency response of this subsystem is provided in Figure 17.

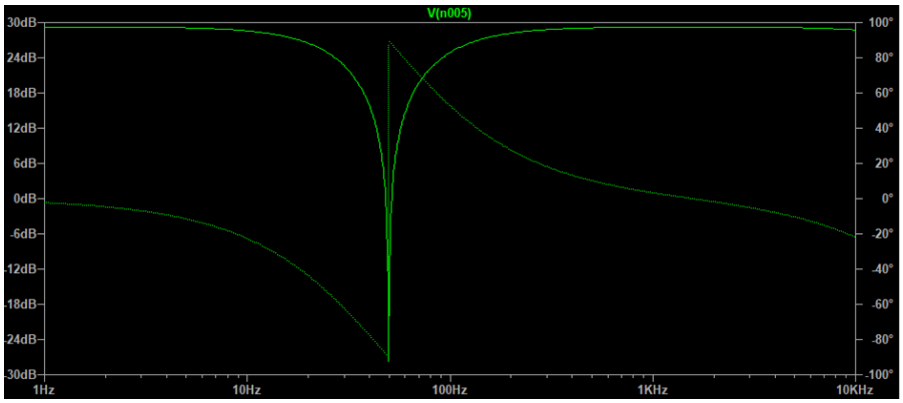

Figure 17- Simulated bode plot of notch filter (dark line is magnitude, light line is phase)

As expected, the signal is significantly attenuated around the 50Hz band- which was the dsired band-stop frequency. Lastly, the overall frequency response of this circuit is included in Figure 18.

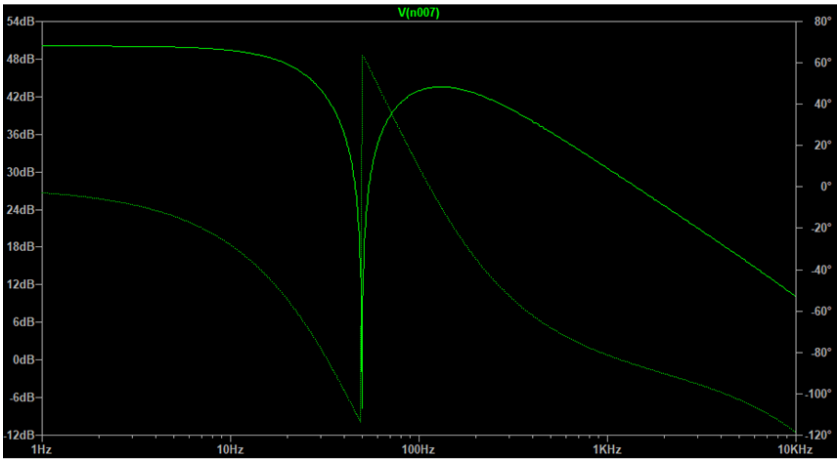

Figure 18- Simulated bode plot of overall circuit (dark line is magnitude, light line is phase)

As expected, here, we can see the shape of the frequency responses of the two filters essentially superposed over one another. The magnitudes in this plot, however, are higher than the bode plots before, reflecting the gain of the instrumental amplifier which is now also included.

5.0 Conclusions & Future Work

In this project, an EMG device capable of registering bicep contractions was developed. The device successfully and reliably processed the EMG signals to output keyboard shortcuts, which were used to control three games: Pong, Dinosaur, and Flappy Bird.

There is, however, still significant room for further development of this technology, if it is to form the foundation of new, accessible video game control technologies. Firstly, at this stage, only one EMG source is being recorded, significantly restricting the complexity of the apps the device can control (only single input software can be controlled). Furthermore, the device has only been tested on a single muscle group- the bicep- and its performance must be characterised and tested for a broader number of body parts/areas. Work could also be done to investigate how the EMG can detect movements of large muscle groups, or whole limbs, to increase the complexity of the inputs users could elicit. As such, future work should focus on extending the current functionality of the device, to increase its versatility, and the possible contexts it can be applied to.

6.0 References

[1] I. Granic. "Video games play may provide learning, health, social benefits." American Psychological Association. https://www.apa.org/news/press/releases/2013/11/video-games (accessed.

[2] E. Tutorials. "Low Pass Filter - Passive RC Filter Tutorial." AspenCore Inc. https://www.electronics-tutorials.ws/filter/filter_2.html (accessed.

[3] B. Burr, "Single Supply, MicroPower INSTRUMENTATION AMPLIFIER," vol. PD-1388B, ed. USA: Burr-Brown, 1997.

[4] *Instrument Control Toolbox*. (2022). MathWorks. [Online]. Available: https://au.mathworks.com/products/instrument.html

[5] *java.awt (Java Platform SE 7)*. (2020). Oracle. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html

[6] Mxmcd. "Play Flappy Bird." MXMCD. https://flappybird.io/ (accessed.

[7] PongGame.org. "Pong Game." PongGame.org. https://www.ponggame.org/ (accessed.

[8] G. D. Game. "Chrome Dino Game Online." Unknown. https://dino-chrome.com/en (accessed.

[9] M. B. I. Raez, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis: detection, processing, classification and applications," (in eng), *Biol Proced Online,* vol. 8, pp. 11-35, 2006, doi: 10.1251/bpo115.

Appendices

MATLAB Code:

There are 2 MATLAB files used. GUI_Window.m (below) generates a GUI to select which game they wish to play, and also starts running the serial port object which reads the data in-coming from the Arduino. The code is included below:

```
% Adapted from https://au.mathworks.com/help/matlab/creating_guis/create-and-run-a-simple-programmatic-app.html
% By Siddhant Tandon (28677528)
% 25/05/2022

function GUI_Window
    % SIMPLEAPP Interactively explore plotting functions
    %   Choose the function used to plot the sample data to see the
    %   differences between surface plots, mesh plots, and waterfall plots

    arduinoObj = serialport("COM7",9600); % Create Serial Object
    configureTerminator(arduinoObj,"CR/LF");
    flush(arduinoObj); % Clean serial buffer
    arduinoObj.UserData = struct("Data",[],"Count",1);


    % Create figure window
    fig = uifigure;
    fig.Name = "EMG Game Manager";

    % Manage app layout
    gl = uigridlayout(fig,[1 1]);
    gl.RowHeight = {50,'1x'};
    gl.ColumnWidth = {'fit','1x'};

    % Create UI components
    title = uilabel(gl);
    lbl = uilabel(gl);
    dd = uidropdown(gl);

    % Lay out UI components
    % Position title
    title.Layout.Row = 1;
    title.Layout.Column = 1;
    % Position label
    lbl.Layout.Row = 2;
    lbl.Layout.Column = 1;
    % Position drop-down
    dd.Layout.Row = 2;
    dd.Layout.Column = 2;

    % Configure UI component appearance
    lbl.Text = "Which Game Would you Like to Play";
    title.Text = "Welcome to the Game Manager";
    dd.Items = ["Dinosaur" "Flappy Bird" "Pong"];
    dd.Value = "Dinosaur";
    arduinoObj.UserData.GameName = "Dinosaur";

    arduinoObj.UserData.Calib_period = 1000; % Number of data enteries to wait for when Calibrating
    arduinoObj.UserData.Continuous_Calib_Period = 200; % Also calibrate the system every 100 seconds or so

    arduinoObj.UserData.Calibrated_avg = 0; % Initialise the field
    arduinoObj.UserData.std_dev = 0;
    arduinoObj.UserData.sens_offset = 0;
    arduinoObj.UserData.last_trig = 0;
```

```
    wait_time = 1000/110; % Compute the estimate calibration time

    arduinoObj.UserData.KeyState = "Up";

    warning("Please wait roughly %f seconds for calibration before moving (sit still)",
wait_time) % Print informational message to the command window
    configureCallback(arduinoObj, "terminator", @Read_Filter); % Setup callback

    % Assign callback function to drop-down
    dd.ValueChangedFcn = {@changePlotType,arduinoObj};

end

% Program app behavior
function changePlotType(src,event,obj)
    type = event.Value;
    switch type
        case "Dinosaur"
            obj.UserData.GameName = "Dinosaur";
        case "Flappy Bird"
            obj.UserData.GameName = "Flappy Bird";
        case "Pong"
            obj.UserData.GameName = "Pong";
    end
end
```

Read_filter.m is the function file that perofrms the calibrations steps, and programs the keyboard outputs of the device.

```
% Adapted from https://au.mathworks.com/help/matlab/creating_guis/create-and-run-a-simple-
programmatic-app.html
% By Siddhant Tandon (28677528)
% 25/05/2022

function Read_Filter(src, ~)
    import java.awt.*;
    import java.awt.event.*;
    % Read the ASCII data from the serialport object.
    data = readline(src);

    % Convert the string data to numeric type and save it in the UserData
    % property of the serialport object.
    dub_dat = str2double(data);
    src.UserData.Data(end+1) = str2double(data);
    % disp(dub_dat);

    % Program key inputs
    if src.UserData.GameName == "Dinosaur" || src.UserData.GameName == "Flappy Bird"

        if dub_dat == 0 && src.UserData.Count > src.UserData.Calib_period &&
(src.UserData.Count - src.UserData.last_trig) >= 50
            rob=Robot;
            %rob.keyPress(KeyEvent.VK_SPACE)
            %rob.keyRelease(KeyEvent.VK_SPACE)
            rob.keyPress(KeyEvent.VK_SPACE)
            rob.keyRelease(KeyEvent.VK_SPACE)
            display("DOUBLE SPACE!!!")
            src.UserData.last_trig = src.UserData.Count;

        elseif (abs(dub_dat - src.UserData.Calibrated_avg) > src.UserData.sens_offset) &&
(src.UserData.Count > src.UserData.Calib_period) && (src.UserData.Count -
src.UserData.last_trig) >= 50
            rob=Robot;
            rob.keyPress(KeyEvent.VK_SPACE)
            rob.keyRelease(KeyEvent.VK_SPACE)
            display("SINGLE SPACE!!!")
```

```
            src.UserData.last_trig = src.UserData.Count;
        end
    elseif src.UserData.GameName == "Pong";
        rob=Robot;

        if (abs(dub_dat - src.UserData.Calibrated_avg) > src.UserData.sens_offset) &&
(src.UserData.Count > src.UserData.Calib_period) && (src.UserData.Count -
src.UserData.last_trig) >= 40
            rob=Robot;
            if (src.UserData.KeyState == "UP")
                rob.keyRelease(KeyEvent.VK_UP)
                rob.keyPress(KeyEvent.VK_DOWN)
                src.UserData.KeyState = "DOWN"
            else
                rob.keyRelease(KeyEvent.VK_DOWN)
                rob.keyPress(KeyEvent.VK_UP)
                src.UserData.KeyState = "UP"
            end
            display("SWITCHED")
            src.UserData.last_trig = src.UserData.Count;
        end

    end
    % Update the Count value of the serialport object.
    src.UserData.Count = src.UserData.Count + 1;
    % Calibration Sequence- to accomodate for changes in impedance (due to
    % more or less sweaty skin, hydration etc.)
    if src.UserData.Count ==  src.UserData.Calib_period
        src.UserData.Calibrated_avg = mean(src.UserData.Data);
        display("Calibrated & Ready to Go!!!")
        src.UserData.std_dev = std(src.UserData.Data);
        src.UserData.sens_offset =  15 * src.UserData.std_dev;
    end

    if rem(src.UserData.Count, src.UserData.Continuous_Calib_Period) == 0
         src.UserData.Calibrated_avg = mean(src.UserData.Data(end-50:end));
    end
```

Arduino Code:

The code used by the RAduinoi to poll the voltage values, and print them to the serial port is included below.

```
int analogPin = A0; //Electrode receiving Output

double val = 0;  // variable to store the value read

void setup() {

 Serial.begin(9600);         //  setup serial

}

void loop() {

  val = analogRead(analogPin);  // read the input pin

  //val = val * scaling_fact * 1000;

  Serial.print(val);

  Serial.write(13); // Print next line chracter also next line

  Serial.write(10);

}
```