

Use Authorization

In presenting this dissertation in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my dissertation for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Signature: _____

Date: _____

**DEVELOPMENT OF A REAL – TIME,
ARTIFICIAL NEURAL NETWORK BASED SEMG CLASSIFICATION ALGORITHM
FOR MOTION IDENTIFICATION**

By

Asib Mahmud

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in the Department of

Measurement and Control Engineering

Idaho State University

May 2018

Committee Approval

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of **Asib Mahmud** find it satisfactory and recommend that it be accepted.

Dr. Marco P. Schoen,
Major Advisor

Dr. Alba Perez Gracia,
Committee Member

Dr. Steve C. Chiu,
Graduate Faculty Representative

Acknowledgment

Behind the scene of this achievement, there stand several people who made my success possible. I felt very privileged and humbled to have their support for this work. First, my most profound acknowledgment to my family, without their support, love, and motivation this accomplishment wouldn't be possible for me. No matter how hard I try, it seems impossible for me to see a way to return the favor to them. My mother, she never stopped me from chasing my goals with her warm wishes and beautiful smile for a long life of full of success even how hard the situation is. My father, he has never stopped me providing the support and guidance I needed to get through this stage of my life. My parents never failed to demonstrate an example of strength, dedication, and fondness. Also, I would like to express my love to my sister and brother in law whose support is always an encouragement to me.

My deepest gratitude to my major advisor, Professor Marco Schoen. Despite his busy schedule, he never hesitated me to provide guidance. In any of my stressful situations, I found him always encouraging, motivating and leading me throughout the degree. Here I am writing my thesis with great happiness because of his guidance and support. Dr. Schoen isn't only my major advisor; he is instead a role model and a good friend. I am honored to work under his supervision, without that I would not have learned the knowledge which is making me feel proud today.

I would like to extend special thanks to my committee members, Dr. Steve Chiu and Dr. Alba Perez. As I remember, from the beginning of my graduate school, I found Dr. Steve very friendly and inspiration while studying and working. He is one of the first guys who make me feel comfortable in the new environment. Besides, I cannot wrap up my acknowledgments, without thanking Dr. Alba Perez. The mechatronics course which I have taken under her helps me to give the knowledges which help me many ways while doing my thesis. Also, I would like to thank both

for providing me financial support through CPI. I also appreciate the help and co – operation of other graduate students and office staff at College of Science and Engineering, Idaho State University for this accomplishment.

TABLE OF CONTENTS

LIST OF FIGURES	IX
ABSTRACT.....	XII
CHAPTER 1: INTRODUCTION	1
1.1 PROBLEM STATEMENT	1
1.2 LITERATURE REVIEW	1
1.3 THESIS GOAL.....	3
1.4 ORGANIZATION OF THE THESIS	5
CHAPTER 2: BIOLOGICAL BACKGROUND	6
2.1 INTRODUCTION OF AN EMG SIGNAL:	6
2.2 SIGNAL ORIGIN	7
2.3 EMG ELECTRODES AND THEIR CATEGORIES:	8
2.3.1 Needle electrodes.....	9
2.3.2 Fine Wire Electrode.....	9
2.3.3 Surface EMG Electrode.....	10
2.3.3.1 Gelled EMG Electrodes	10
2.3.3.2 Dry EMG Electrodes.....	11
2.4 EMG ELECTRODE PLACEMENT	12
2.5 EMG SIGNAL ACQUISITION CIRCUITRY	14
2.5.1 Monopolar configuration	14
2.5.2 Bipolar Configuration	15
2.5.3 Multipolar Configurations.....	15
2.6 RAW EMG SIGNAL.....	16
CHAPTER 3: NEURAL NETWORK.....	17
3.1 OVERVIEW.....	17
3.2 ARTIFICIAL NEURAL NETWORK	17
3.3 BIOLOGICAL INSPIRATION	19
3.4 NEURON MODEL	20

3.5 Backpropagation Neural Network.....	23
3.5.1 Levenberg – Marquardt Backpropagation.....	24
3.5.2 Bayesian Regularization Backpropagation.....	32
3.5.3 Scaled Conjugate Gradient Backpropagation.....	36
CHAPTER 4: EXPERIMENTAL SETUP	39
4.1 SURFACE EMG CIRCUIT DESIGN.....	39
4.1.1 The first stage	39
4.1.2 Second Stage.....	40
4.1.3 Third Stage	42
4.2 SHORTCOMINGS OF PCB AND THE POWER SUPPLY	45
4.3 ELECTRODES PLACEMENT AND ESTABLISHING SEMG SIGNAL	48
CHAPTER 5: IMPLEMENTATION AND VALIDATION OF ANN	50
5.1 IMPLEMENTATION OF REAL – TIME CLASSIFICATION SIMULINK™ MODEL	50
5.1.1 Connecting Arduino to Simulink™ through serial port:.....	52
5.1.2 Overlapping Sliding Window:	54
5.1.3 Checking the Buffer blocks for creating delays:	54
5.1.4 Classification.....	56
5.1.4.1 Integrated EMG (IEMG).....	56
5.1.4.2 Mean Absolute Value (MAV)	57
5.1.4.3 Mean Absolute Value Slope (MAVSLP)	57
5.1.4.4 Simple Square Integral (SSI)	57
5.1.4.5 Root Mean Square (RMS)	57
5.1.4.6 Wavelength Length (WL)	58
5.1.4.8 Slope Sign Change (SSC)	58
5.1.4.9 Willison Amplitude (WAMP).....	59
5.1.4.10 Mean	59
5.2 CONFIGURATION PARAMETERS FOR RUNNING THE MODEL IN SIMULINK:	60
5.3 TRAINING ARTIFICIAL NEURAL NETWORK (ANN):.....	60
5.4 REAL – TIME IDENTIFICATION	69
CHAPTER 6: DISCUSSION AND RESULT	71

6.1 BUILDING REAL – TIME RANDOM OUTPUT VALUES	72
6.2 CLASSIFICATIONS	73
6.3 TRAINING ARTIFICIAL NEURAL NETWORK(ANN)	77
6.4 IDENTIFICATION AND RESULT	85
CHAPTER 7: CONCLUSION AND FUTURE WORKS.....	90
7.1 CONCLUSION	90
7.2 FUTURE WORK	91
REFERENCES	93
APPENDIX.....	97
SENSOR – BASED LED EXAMPLE EXPERIMENTATION SETUP.....	97
CLASSIFICATIONS	100
REAL – TIME SEMG IDENTIFICATION EXPERIMENTAL SETUP.....	109
ARDUINO CODE:.....	110
MATLAB® CODE TO TRAIN THE ANN:.....	111

LIST OF FIGURES

FIGURE 2.1: A SCHEMATIC IMAGE OF PRIMARY MOTOR CONTROL MECHANISMS, MOTOR UNIT AND ITS COMPONENTS [11]	8
FIGURE 2.2: A NEEDLE EMG ELECTRODE [14]	9
FIGURE 2.3: A FINE WIRE EMG ELECTRODE [15]	10
FIGURE 2.4: GELLED EMG ELECTRODES [15]	11
FIGURE 2.5: AN EXAMPLE OF DRY EMG ELECTRODE [15]	11
FIGURE 2.6: FRONTAL VIEW OF ANATOMICAL POSITIONS OF SELECTED SITES [17]	13
FIGURE 2.7: MONOPOLAR SIGNAL ACQUISITION TECHNIQUE [15]	14
FIGURE 2.8: BIPOLAR CONFIGURATION [15]	15
FIGURE 2.9: THE RAW EMG SIGNAL OF THREE CONSTRUCTION BURST OF THE BICEPS [17]	16
FIGURE 3.1: THE BASIC WORKING PRINCIPLE OF AN ANN [18]	18
FIGURE 3.2: SCHEMATIC DIAGRAM OF TWO BIOLOGICAL NEURONS [19]	19
FIGURE 3.3: SINGLE – INPUT NEURON MODEL [19]	20
FIGURE 3.4: MULTIPLE – INPUT NEURON [2]	21
FIGURE 3.5: A LAYER OF NEURONS EACH WITH MULTIPLE INPUTS AND ONE OUTPUT [19]	21
FIGURE 3.6: MULTIPLE – LAYERS OF NEURON MODEL [19]. HERE ONLY THREE LAYERS ARE SHOWN. HOWEVER, THE NUMBER OF LAYERS CAN BE EXTENDED AS DESIRED	22
FIGURE 3.7: AN EXAMPLE OF A RECURRENT NEURAL NETWORK(RNN) [19]	22
FIGURE 3.8: MECHANISM CHART OF BACKPROPAGATION [21]. HERE, AN ERROR VALUE IS CREATED BASED ON ‘N’ INPUT NEURONS AND ‘M’ OUTPUT NEURONS AND THEN BACKPROPAGATE(S) THROUGH THE HIDDEN LAYER. THROUGH THIS PROCESS, THE ERROR IS MINIMIZED BY UPDATING THE WEIGHTS OF THE HIDDEN AND OUTPUT NEURONS	23
FIGURE 4.1: INSTRUMENTATION AMPLIFIER [27]	39
FIGURE 4.2: UAF42 CIRCUIT [28]	40
FIGURE 4.3: A NOTCH FILTER [27]	41
FIGURE 4.4: HIGH – PASS FILTER [27]	42
FIGURE 4.5: LOW – PASS FILTER [27]	43
FIGURE 4.6: BUFFER AMPLIFIER ALONG VOLTAGE SHIFT CIRCUIT [27]	43
FIGURE 4.7: FINAL IMPLEMENTATION OF A PCB MODEL [27]	44

FIGURE 4.8: POWER SUPPLY DIAGRAM	45
FIGURE 4.9: POWER SUPPLY CIRCUIT CONNECTION	45
FIGURE 4.10: sEMG CHANNEL BOX CONTAINS TEN CHANNELS [29]	45
FIGURE 4.11: NEW POWER SUPPLY CONNECTION USING THE SOLDERABLE BOARD.....	46
FIGURE 4.12: NEW sEMG CHANNEL BOX SETUP	47
FIGURE 4.13: INNER FOREARM MUSCLE MOVEMENT POSITION	48
FIGURE 4.14: OUTER FOREARM MUSCLE MOVEMENT POSITION.....	48
FIGURE 4.15: sEMG SIGNAL FOR INNER FOREARM MUSCLE MOVEMENT.....	49
FIGURE 4.16: sEMG SIGNAL FOR OUTER FOREARM MUSCLE MOVEMENT	49
FIGURE 5.1: REAL – TIME CLASSIFICATION SIMULINK™ BLOCK MODEL.....	51
FIGURE 5.2: 4 BITS OF VALUE FROM THE SERIAL MONITOR	52
FIGURE 5.3: PARAMETERS SELECTION FOR (A) SERIAL RECEIVE BLOCK AND (B) SERIAL CONFIGURATION	53
FIGURE 5.4: CHECKING DELAYS USING BUFFER BLOCK	54
FIGURE 5.5: OUTPUT OF BUFFER DELAY BLOCK	54
FIGURE 5.6: CHECKING OVERLAPS USING BUFFER BLOCK	55
FIGURE 5.7: OUTPUT RESULTS OF BUFFER OVERLAPPING CALCULATION	55
FIGURE 5.8: SIMULINK™ DIAGRAM FOR STORING THE SIGNAL VALUES INTO THE MATLAB® WORKSPACE.....	61
FIGURE 5.9: CIRCUIT CONNECTION FOR A PHOTORESISTOR LED SENSOR	62
FIGURE 5.10: TARGETED OUTPUT VALUE	63
FIGURE 5.11: ANALOG OUTPUT FROM THE CHANNEL	63
FIGURE 5.12: TRAINING RESULT FOR NN TOOLBOX.....	66
FIGURE 5.13: COMPARING RESULT BETWEEN TARGETED OUTPUT AND PREDICTED OUTPUT	66
FIGURE 5.14: OBSERVING THE ERROR BETWEEN TWO OUTPUTS	67
FIGURE 5.15: SIMULINK™ MODEL FOR REAL – TIME IMPLEMENTATION.....	68
FIGURE 5.16: REAL – TIME IDENTIFICATION FROM TRAINED NN SIMULINK™ BLOCK. NUMBER THREE REPRESENTING FLASHING LIGHT ON RESISTORS, NUMBER ONE FOR NORMAL CONDITION AND NUMBER TWO FOR SHADOWING THE PHOTORESISTOR.	69
FIGURE 6.1: PUSHBUTTON VALUES AFFECTING THE NATURAL CHARACTERISTIC OF sEMG SIGNAL.....	72
FIGURE 6.2: ON/OFF SIMULINK BUTTONS AND THEIR CORRESPONDING OUTPUT VALUES	73

FIGURE 6.3: THE FIRST GRAPH IS SEMG SIGNAL, SECOND IEMG, THIRD MAV, FOURTH MAVSLP, FIFTH SSI, SIXTH RMS, SEVENTH WL CLASSIFICATIONS OUTPUTS	75
FIGURE 6.4: THE FIRST GRAPH IS SSI, SECOND RMS, THIRD WL, FOURTH ZC, FIFTH SSC, SIXTH WAMP AND THE LAST ONE IS MEAN CLASSIFICATIONS OUTPUTS.	76
FIGURE 6.5: FOUR DIFFERENT CLASSIFICATIONS SIMULINK™ BLOCK.....	78
FIGURE 6.6:SEMG SIGNAL FOR INNER FOREARM MUSCLE MOVEMENT	79
FIGURE 6.7:SEMG SIGNAL FOR OUTER FOREARM MUSCLE MOVEMENT.....	79
FIGURE 6.8:REAL – TIME PUSHBUTTON OUTPUT CONSIDERED AS A TARGETED OUTPUT	79
FIGURE 6.9: TRAINING RESULTS FOR NN TOOLBOX. (A) SLOPE SIGN CHANGE (SSC) (B) WAVEFORM LENGTH (WL) (C) ZERO CROSSING (ZC) (D) WILSON AMPLITUDE (WAMP). ...	82
FIGURE 6.10: FOUR DIFFERENT CLASSIFICATION SSC, WL, ZC AND WAMP TRAINED OUTPUT GRAPHS.....	83
FIGURE 6.11: ERROR DIFFERENCE BETWEEN TARGETED AND PREDICTED OUTPUTS FOR SSC AND WL.....	83
FIGURE 6.12: ERROR DIFFERENCE BETWEEN TARGETED AND PREDICTED OUTPUTS FOR ZC AND WAMP.....	84
FIGURE 6.13: TRAINED ANN SIMULINK™ BLOCK IMPLEMENTATION FOR REAL – TIME IDENTIFICATION	86
FIGURE 6.14: IDENTIFICATION USING SSC CLASSIFICATION	87
FIGURE 6.15: IDENTIFICATION USING WL CLASSIFICATION	87
FIGURE 6.16: IDENTIFICATION USING ZC CLASSIFICATION	88
FIGURE 6.17: IDENTIFICATION USING WAMP CLASSIFICATION	88
FIGURE 0.1: SIMULINK™ RUN TIME MODE SELECTED AS NORMAL	99
FIGURE 0.2: ARDUINO MEGA 2560 SETUP FOR SIMULINK™ MODEL	99
FIGURE 0.3: BUFFER BLOCK PARAMETER. HERE,50 OVERLAPPING IS CHOSEN	100
FIGURE 0.4: TARGETED OUTPUT SIMULINK BLOCK AND ITS CONFIGURATION.	105
FIGURE 0.5: GENERATED SIMULINK™ BLOCK FROM THE TRAINED NN	108
FIGURE 0.6: SATURATION BLOCK CONFIGURATION PARAMETERS	112

ABSTRACT

Development of a Real – Time, Artificial Neural Network based sEMG Classification Algorithm
for Motion Identification

Thesis Abstract--Idaho State University (2018)

This work deals with aiding and developing upper body rehabilitation engineering methods for stroke victims. In most rehabilitation cases, only partial success is accomplished after long training sessions, and the patient is left with a limited range of motion. The aim of this research is to develop a training tool utilizing an augmented reality device (ARWED) that addresses the rehabilitation of human hand and forearm motion. For this purpose, the research is accomplished in three steps. The first step focuses on the development of real – time sEMG classification methods where ten classification methods have been chosen based on their different characteristics. The second step explains the development of real – time Artificial Neural Network (ANN) models based on the signal classifications. This part consists of training the ANN model based on real – time classification. The third step discusses the development of identification algorithms for identifying motion intend using real – time ANN models by using Simulink™ block model. The results indicate preferences for the preferred classifier: Slope Sign Change (SSC), Waveform Length (WL), Zero Crossing (ZC), and Wilson Amplitude (WAMP) which can help in the design of the signal processing for real – time ANN implementation.

Key Words: Rehabilitation, surface Electromyography (sEMG) classification, real – time Artificial Neural Network (ANN), Backpropagation, Levenberg – Marquardt(LM), Arduino Mega 2560, Simulink™, MATLAB®, Motion identification, Slope Sign Change (SSC), Waveform Length (WL), Zero Crossing (ZC), and Wilson Amplitude (WAMP), Neural Network training.

CHAPTER 1: INTRODUCTION

1.1 Problem Statement

According to the Archives of Physical Medicine and Rehabilitation, nearly two million people are living in the United States with limb loss [1]. Current research shows that one in 190 Americans is living with the loss of limbs. This number has been increasing due to military engagements and vascular disease like strokes. It is estimated that the number of people living with limbs loss will be more than double to 3.6 million by 2050. There are many efforts to deal and improve with the advancement of prosthetic devices to help victims. Nonetheless, currently, there are no prosthetic devices available which can mimic the functionality of a human hand. Also, the available prosthetic devices are costly. Recent scientific and commercial advances in the man-machine interface field are promising and suggest that naturally controlled and simultaneous robotics prostheses can be a reality in the future for amputees. However, the framework of the situation in the market and the scientific ground is complicated, and the direction to naturally controlled prostheses still needs many improvements.

1.2 Literature Review

Upper – limb deficiency severely affects the ability to perform daily living activities. Myoelectric hand prostheses with many degrees of freedom are commercially available. Recent advances in rehabilitation robotics propose that their natural control can be performed in real life. These naturally controlled robotic prostheses can become a reality in everyday life. However, the path still needs many steps. In 2015, Atrzori et al.[2] proposed an overview of the advancements both in commercial and scientific domains to outline the current and future changes in this field.

Moreover, for upper limb amputees, the prosthesis control training is suggested before and after fitting. For functional monitoring, there are many different tests available. However, none can be used in the early phase of training. In 2015, Sturma et al. [3] proposed a tool for pre – evaluation of trainable voluntary muscle – activation skills before prosthetic fitting. The proposed tool supports planning of rehabilitation procedures for further monitoring where essential considerations for system development and the main features of the developed prototype are presented.

A human can learn new movements through imitation and other learning techniques. There are some structured rehabilitation methods, which includes imitation, repetition, and reinforcement learning to improve multifunctional prosthetic control. For this purpose, in 2015, Roche et al.[4] suggests a structured training protocol of imitation, repetition and supporting role in learning to control of a new prosthetic hand. While considering the repetitive movements for rehabilitation, video – game based therapies can increase patient motivation, effort, and performance. For this purpose, a clinically feasible and entertaining virtual rehabilitation intervention is established and evaluated for short – term improvement of EMG control engaging gameplay elements [5].

Furthermore, to improve upper – limb deficiency for independence and quality of life, surface Electromyography (sEMG) based control systems have been widely researched for several decades [6], [7]. However, advanced myoelectric prosthetic hands are limited due to the lack of real – time control performance and weak signal sources on amputation residual muscles. A novel human – machine interface is done for prosthesis manipulation which combines the advantages of surface Electromyography(sEMG) and near – infrared spectroscopy (NIRS) to conquer the limitations of myoelectric control.[8] This experiment in [8] evaluates both offline classification accuracy (CA) and online performance of the forearm motion recognition system based on three types of sensors:

EMG, NIRS, and hybrid EMG – NIRS where the result shows that combining EMG and NIRS signals gives better real – time performance. Considering sEMG signal is extensively studied and applied in clinic and engineering. However, the major drawback of sEMG pattern recognition is the poor recognition results because of the presence of noise. Thus, methods of reducing the noise influence become the most significant in EMG signal analysis. In 2009, Phinyomark et al. [9] presented a novel feature that can tolerate with White Gaussian Noise (WGN) without using a noise removal algorithm. As a result, the experiment result shows better recognition result in a noisy environment than other success feature candidates.

1.3 Thesis Goal

From the literature reviews, it can be said that natural and proficient controls of robotic hand prostheses have been studied for a long time by the scientific community. Among them, most of the methods rely on the use of sEMG signals and pattern recognition, imitating or different learning techniques. Nonetheless, the control robustness offered by the researchers is still not sufficient for many real – life applications. In the recent years, deep learning transformed several fields of machine learning, including speech and computer vision. The aim of this research to develop a real – time Artificial Neural Network (ANN) for motion identification. This base can then be extended to Deep learning. The primary concept of this research to make better upper body prosthetic devices and develop methods that facilitate stroke victim rehabilitation.

There are multiple types of hand prostheses. One of them is based on surface Electromyographic (sEMG) signals to initiate the actuation of the robotic hand [2]. sEMG signals are spatially distributed which pick up signals from other motor units stemming from different muscle groups.

The content of sEMG signal can be identified based on amplitude, frequency, and amplitude frequency.

The first objective of this research is to develop a real – time sEMG signal classification. Ten different types of classifications are chosen based on their different characteristics. The hypothesis is that using more classifications can extract a significant amount of information which are going to be used for training the ANN. Each of the classifications designed as a sliding window with a 0.067 sampling time (seconds). The classification and the size of the sliding window are picked randomly. All the real – time classifications are done in Simulink™.

The second objective of this work is to establish an ANN based on real – time signal classifications. For training purpose, two movements have chosen – inner and outer forearm movements. This part is done by extracting the signal values which are collected from the real – time Simulink™ classifications model and applied to ANN for offline training using a developed MATLAB® code. Backpropagation algorithm is used to train the proposed ANN. By following this method, the error can eventually be minimized, and the algorithm adjoined with an optimization result. Backpropagation has different training approaches which can optimize output results. In this part, the Levenberg-Marquardt, Scaled Conjugate Gradient and Bayesian Regularization methods are used for better optimization technique. However, for training and prediction, the Levenberg-Marquardt backpropagation method gives the better result compared to the others. Thus, Levenberg – Marquardt is chosen for observing the finals results.

Moreover, the third objective is to build motion identification intend using real – time ANN models. In this part, a Simulink™ block is built from the offline trained ANN model. Afterwards,

that block is placed to a Simulink™ model where all the classifications are used for extracting the sEMG signals in real – time.

1.4 Organization of the thesis

The thesis is organized as follows. Chapter 1 discusses the theme of the thesis and literature review. Chapter 2 provides the biological background of sEMG signals and its different categories. Chapter 3 focuses on the mathematical background of Artificial Neural Network (ANN), its different algorithm followed by an example of implementation of a real – time classification based ANN for motion identification. In Chapter 4, the experimental setup for the observation of sEMG signal is discussed. Chapter 5 focuses on the results and discussion based on the comprehensive sEMG example. Chapter 6 represents the conclusion and future works.

CHAPTER 2: BIOLOGICAL BACKGROUND

2.1 Introduction of an EMG signal:

Electromyography (EMG) is considered an experimental technique involving establishing, recording, and investigating of myoelectric signals. It is a study of muscle function through the analysis of the electrical signals originated during muscular contractions. There is a wide range of benefits for using EMG signal. It establishes evaluation tool for applied research, rehabilitation, sports science, and ergonomics. Besides EMG signals also have some common benefits which allow investigating the muscle directly, permit measurement of muscular performance, help patients to find and train their muscle, concede analysis to improve sports activities, and document treatment and training regimes.

There are two main types of EMG signals – clinical and kinesiological EMG. Clinical EMG, typically done by neurologists and physiatrists. Kinesiological EMG is the most preferred in the literature considering movement analysis. Again, kinesiological EMG is divided into two categories – surface and fine wire. Surface EMG determines muscle function by recording the muscle activity on the skin surface. It requires surface electrodes which provide a limited assessment of the muscle activity. Surface EMG can be recorded by using a pair of electrodes because EMG recordings display the potential difference between two separate electrodes. The advantages of using surface EMG are that they are easy to apply, cause minimal pain with the application, and the process is perfect for movement application. The disadvantage of sEMG is that it requires a significant pickup area. Therefore, it increases the possibility for crosstalk from adjacent muscles. On the other hand, fine wire EMG requires insertion of needles into the muscle. The advantages of using fine wire EMG are the ability to test deep muscle, separation of specific

muscle parts of large muscles, and capable of testing small muscles which might be difficult to detect with surface EMG, due to crosstalk. The disadvantages of wire EMG are the insertion of the needle may cause discomfort, can increase the tightness in the muscle, and the electrodes are less repeatable because it is difficult to place the needle in the same area of muscle for every research. For this research, surface EMG signals have been chosen because it is easy to apply, electrodes can be used for several times, and it is without much discomfort compared to fine wire EMG process.

2.2 Signal Origin

Electromyography measures the electrical signal correlated with voluntary or involuntary muscle contraction. This muscle contraction is related to tension for the EMG activity. The functional unit of the muscle contraction is called a motor unit (MU). An MU consist of an alpha motoneuron in the spinal cord and the muscle fibers it innervates. The alpha motoneuron is the final point of addition for all the descending and reflexing inputs. Various synaptic innervation sites determine the discharge pattern of the motor unit by inducing a current in the motoneuron. The number of MUs per muscle in a human body may range from 100 to 1,000 depending upon small hand muscle or large limb muscles [10]. This different number of MUs changes significantly based force generating capacity. Figure 2.1 represents the schematic of the basic motor control mechanisms, motor units, and its components.

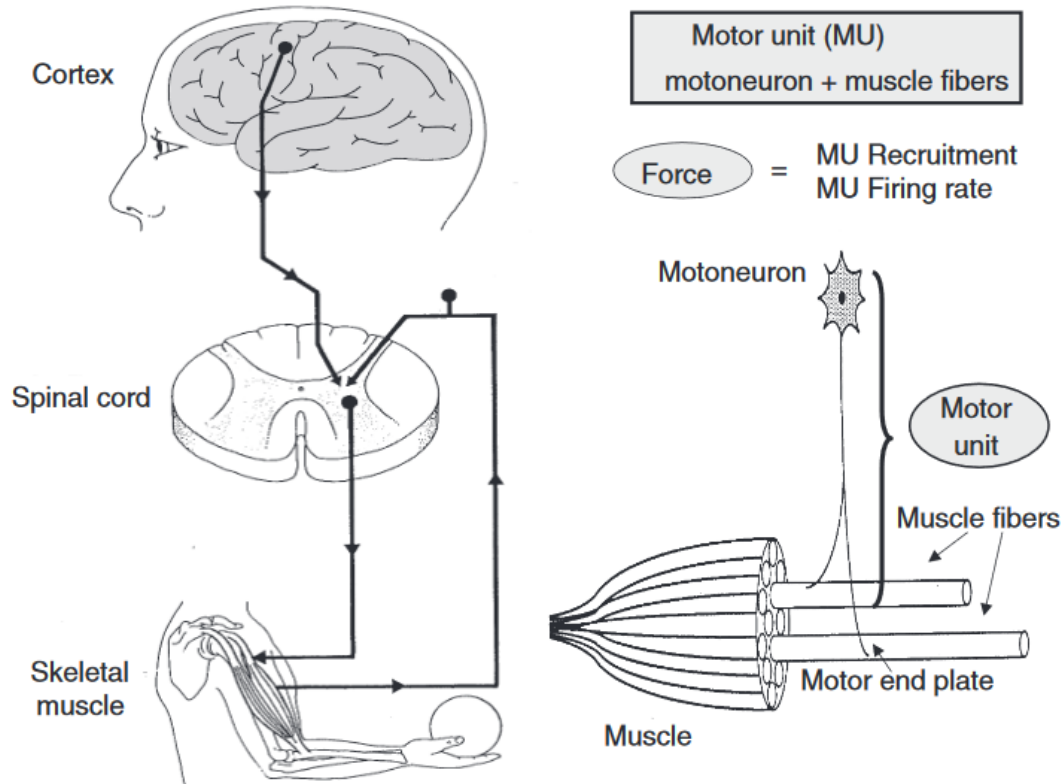


Figure 2.1: A schematic image of primary motor control mechanisms, motor unit and its components [11]

If there is an individual activity where the muscles of the body are to be recruited, the brain sends excitation signals through the Central Nervous System (CNS). In scientific terms, when there is a need for greater forces, the excitation from the CNS increases, which leads the motor units to increase activation level and of the firing rate. As a result, EMG signal shows up with high signal amplitudes [12,13].

2.3 EMG Electrodes and their categories:

EMG electrodes help to detect the bioelectrical activity inside the muscle of a human body. There are mainly two types of EMG electrodes – Surface and Inserted electrodes. Inserted electrodes are

also categorized into two different electrode types: needle and fine wire electrodes. However, since sEMG electrodes are exclusively used in this work, they are discussed separately.

2.3.1 Needle electrodes

Needle electrodes are commonly used in clinical procedure particularly in neuromuscular evaluations. The tip of the needle is called core which is bare and used for detection of a surface. There is an insulated wire in the cannula which is comparatively improved from another available category. It has relatively small pickup area and enables the electrodes to detect low force contractions. Figure 2.2 shows some details about the Needle electrodes.

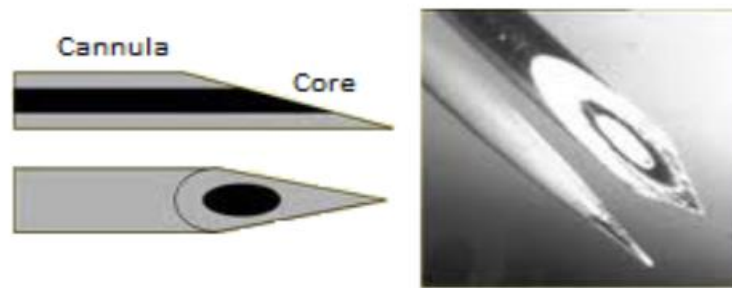


Figure 2.2: A Needle EMG Electrode [14]

2.3.2 Fine Wire Electrode

This kind of electrode is made of an alloy of platinum, silver, nickel, and chromium, which has a small diameter, is highly non-oxidizing, and constitutes a stiff wire with insulation. Fine wire electrodes are easily implanted and withdrawn from the muscle. They are less painful compared to needle electrodes. A fine wire electrode is shown in Figure 2.3.



Figure 2.3: A fine wire EMG electrode [15]

2.3.3 Surface EMG Electrode

Surface EMG electrodes implement a non-invasive process for measurement and detection of EMG signals. This technique increasingly used to detect activity to control device extensions to accomplish prosthesis for physically disabled and a population with amputations. There are two types of surface EMG Electrodes-Gelled and Dry EMG electrode. However, in categories, it is two types-Passive EMG electrodes and Active EMG electrodes. Passive electrodes should connect to an external amplification circuitry for proper acquisition of the EMG signal. On the other hand, active EMG electrodes consist of pre-amplifier attachment for surface electrodes.

2.3.3.1 Gelled EMG Electrodes

These types of electrodes contain a gelled electrolytic material as an interface between skin and electrodes. Oxidation and reduction reactions occur at the metal electrode junction. Silver-silver chloride (Ag-AgCl) is most by used for the metallic part of the gelled electrodes. It allows current from the muscle to pass freely across the intersection between the electrode and the electrolyte. Almost around 80% of Surface EMG applications use Ag – Agcl electrodes. Gelled electrodes

require special skin preparation [16] and precautions such as hair removal, prevention of sweat accumulation, proper gel concentration, etc. Gelled EMG electrodes are shown in Figure 2.4.



Figure 2.4: Gelled EMG Electrodes [15]

2.3.3.2 Dry EMG Electrodes

As it is a Dry Electrode, it does not require a gel interface between skin and the detecting surface. Figure 2.5 shows an example of dry EMG electrode. It might contain more than one surface. They are usually heavier ($>20\text{g}$) compared to gelled electrodes ($<1\text{g}$) which cause for electrode fixation. Thus, a material for the stability of the electrode with the skin is needed



Figure 2.5: An Example of Dry EMG Electrode [15]

2.4 EMG Electrode Placement

Application of surface EMG electrodes needs proper skin preparation to obtain quality EMG signal. The skin's electrical resistance must be reduced. For this purpose, hair must be removed, and it is advisable to use an abrasive gel to reduce the dry layer of skin where EMG electrodes are to be placed. The surface EMG (sEMG) electrodes should be attached between the motor unit and the tendinous insertion of the muscle. The distance between the center electrodes should be only 1-2cm. It is essential to have a proper understanding of the particular muscles from where the EMG signal is being extracted in order to get the best result from sEMG. Crucial limb and trunk muscles activity can be investigated by using sEMG signals. For more in-depth, smaller or overlaid muscles, a fine wire application is required in order to safely and selectively detected the corresponding sEMG signals. Figure 2.6 shows a muscle map, a selection of muscles that commonly are investigated in kinesiological studies, where the two yellow dots of the surface muscles point out the orientation of the electrode.

Fine Wire Sites:

Surface Sites:

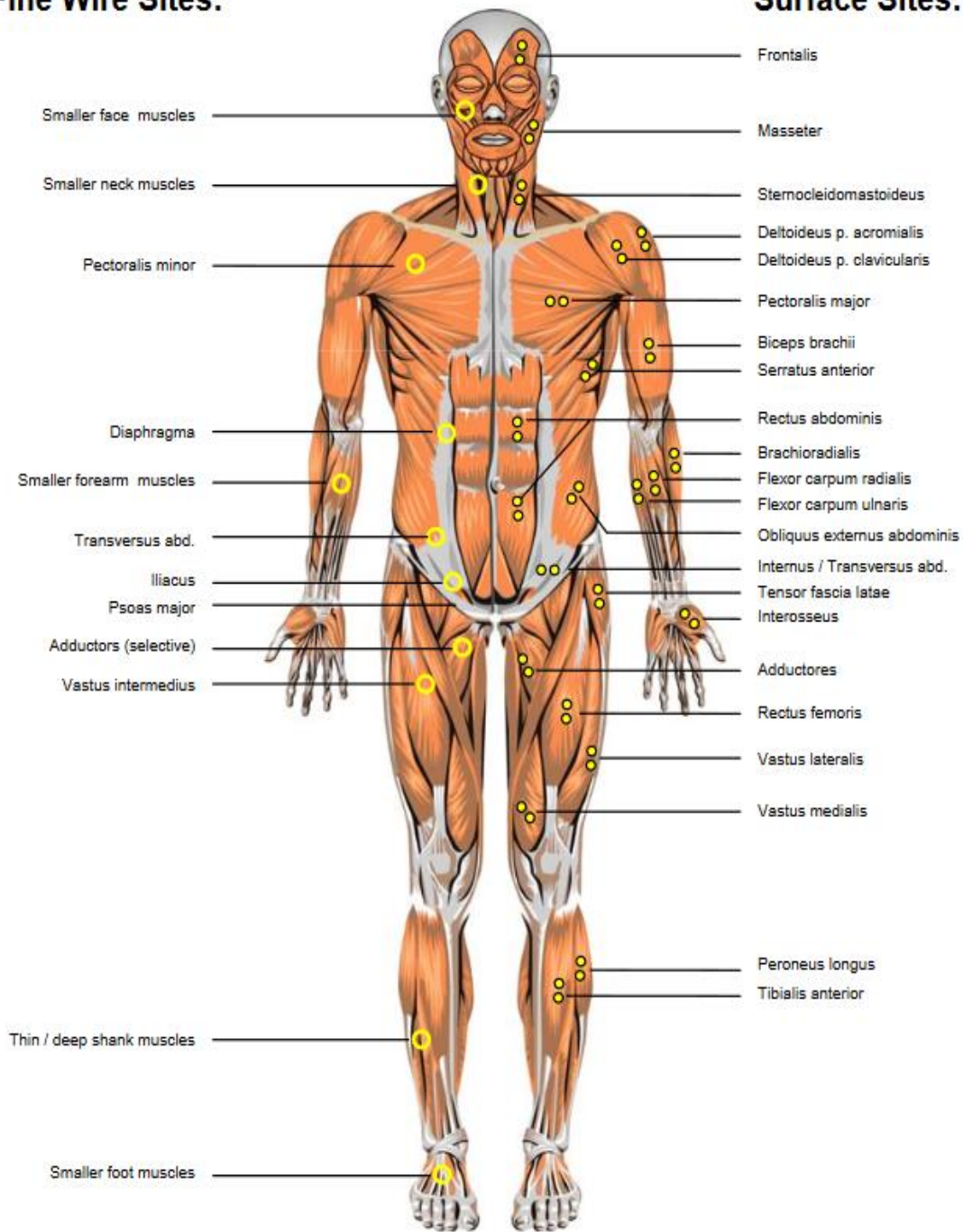


Figure 2.6: Frontal view of anatomical positions of selected sites [17]

2.5 EMG Signal Acquisition Circuitry

The method of differential amplification is utilized to acquire sEMG signals. This setup should have high input impedance and very low output impedance, which can be achieved with the help of instrumentation amplifiers. This instrumentation amplification accomplishes differential amplification by subtracting the voltages V_1 and V_2 (see Figure 2.7 and 2.8). By this way, the noise signal, which is common at V_1 and V_2 are eliminated. The habit of a differential amplification to reject signal is common to both inputs is regulated by common mode rejection ratio (CMRR) of 90dB. The placement of the sEMG and detecting surfaces are categorized in three different configurations: monopolar, bipolar, and multipolar.

2.5.1 Monopolar configuration

This configuration is carried out using only a single electrode on the skin against a reference electrode as shown in Figure 2.7. Though this method is used for its simplicity, however, it is not recommended as it detects all the electrical signals in the proximity of the detecting surface.

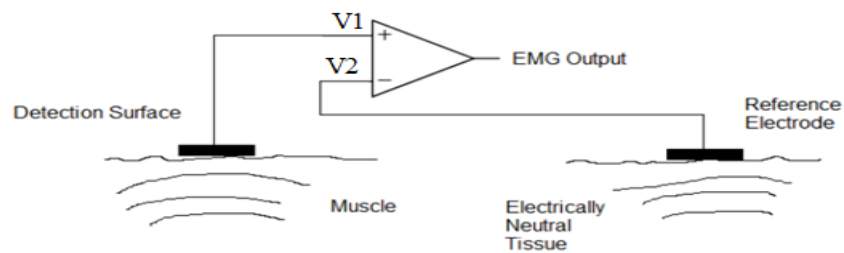


Figure 2.7: Monopolar Signal Acquisition technique [15]

2.5.2 Bipolar Configuration

This configuration is used to acquire sEMG signal using two EMG detecting surfaces with the support of a reference electrode. The two sEMG surfaces are placed only 1-2cm from each other and they are connected to a differential amplifier. The differential amplifier reduces the common noise signals to both inputs and amplifies the difference. It is the most commonly used electrode configuration which is also followed in this research work because it gives less noisy output compared to monopolar configuration and less circuit implementation method compared to multipolar configuration.

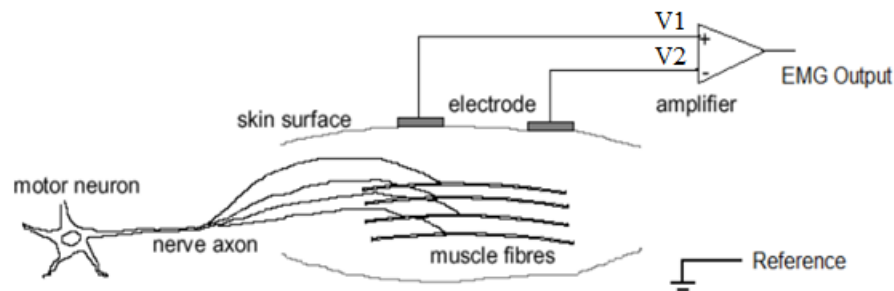


Figure 2.8: Bipolar Configuration [15]

2.5.3 Multipolar Configurations

This method uses more than two detecting surfaces to achieve the EMG signal with the support of a reference electrode. Three or more EMG detecting surfaces can be used, and they are placed from 1 to 2cm apart. This configuration needs to pass through more than two stages of differential amplification. By this way; it reduces crosstalk and noise concerns.

2.6 Raw EMG Signal

An unprocessed and unfiltered signal detecting the Motor Unit Action Potential called as a raw sEMG signal. Figure 2.9 shows a raw surface EMG recording for the three static contractions of the biceps brachii muscle.

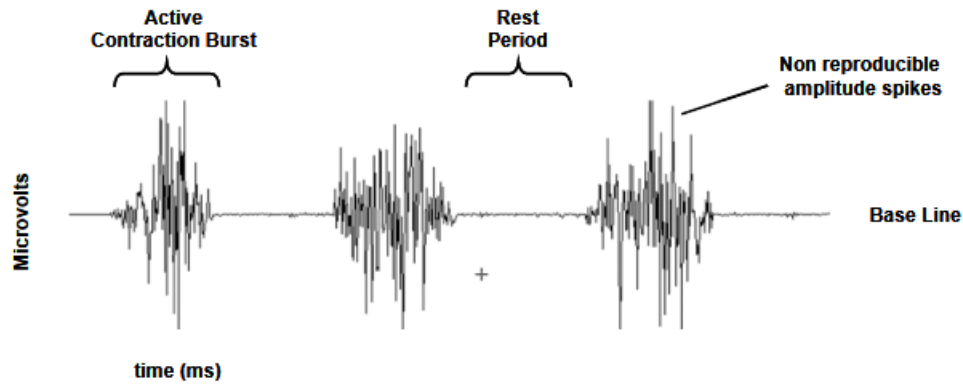


Figure 2.9: The raw EMG signal of three construction burst of the biceps [17]

The raw EMG baseline noise depends upon the environmental noise and the quality of the given detection. Usually, the average baseline noise considered not higher than 3-5 microvolts. By nature, raw sEMG spikes are of random shapes which cannot be reproduced in exact shape. The reason is the actual set of recruited motor units continually changes within the diameter of available motor units. Before the sEMG signal displayed and analyzed in the computer, it needs to be converted from analog to digital conversion (A/D). The resolution of an A/D board should be converted into the amplitude range of ± 5 volts. In this research work, an Arduino board is used which has $2^{10}=1024$ bits. The range of bits starts from 0 to 1023. Thus for this purpose the bits value is converted into the range ± 5 volts by computing $(5/1023) \times \text{voltage}$.

CHAPTER 3: NEURAL NETWORK

3.1 Overview

In the beginning of this chapter, mathematical background of Artificial Neural Network (ANN) and its different algorithm is explained. Furthermore, an example of implementation of a real – time classification – based ANN for motion identification is showed. For this purpose, implementation of Simulink [™] is discussed further which explains all the steps from Arduino to real – time ANN motion identification by showing a LED sensor light experimentation.

3.2 Artificial Neural Network

An Artificial Neural Network (ANN) is an analytical model based on the neural structure of the brain. The essential structure of every ANN is given by a simple mathematical model or function which has three simple sets of rules – multiplication, summation, and activation. At the beginning of artificial neuron, the inputs are weighted that means the individual weights multiplies every input value. The middle section of the model is a summing function that sums all weighted inputs and bias. At the end of an artificial neuron is the sum of previously weighted inputs and bias which is then passing through the activation function which is called a transfer function. Figure 3.1 shows the working principle of ANN.

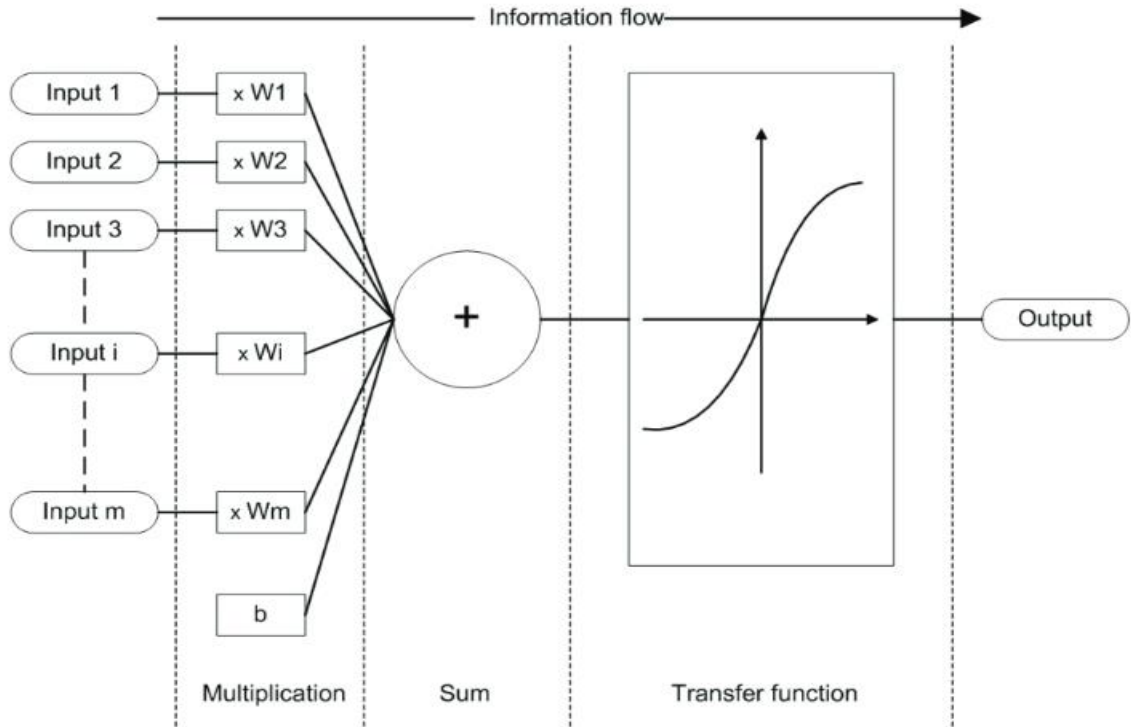


Figure 3.1: The Basic working principle of an ANN [18]

Animal brains are capable of functions which are currently impossible for computers. Computers can do routine things well such as keeping ledgers or performing complicated math computations method. However, computers have trouble recognizing even simple patterns. Therefore, this biologically inspired method of computing is a significant advancement for computing. Research shows that brains can store information as a pattern. Some of these patterns are very convoluted and permits us the ability to recognize individual faces from many different angles. The process of storing information as patterns, utilizing them, and afterward solving problems introduces a new field of computing which is ANN. This field does not utilize traditional programming, however, involves the creation of mostly parallel networks and the training of those networks to solve any specific problems.

3.3 Biological Inspiration

The brain consists of approximately 10^{11} of highly connected elements called neurons [19]. These neurons have three components – the dendrites, the cell body and the axon. From Figure 3.2 tree-like receptive networks are called dendrites which carry electrical signals into the cell body. The cell body efficiently sums and thresholds the incoming signals. A single long fiber that carries the signal from the cell body towards other neurons called the axon. The point of connection where an axon from one cell and a dendrite of another cell come together is called synapse. The whole process is determined by a complicated chemical process with the arrangement of neurons and the strength of the individual synapses, establishes the function of the neural network.

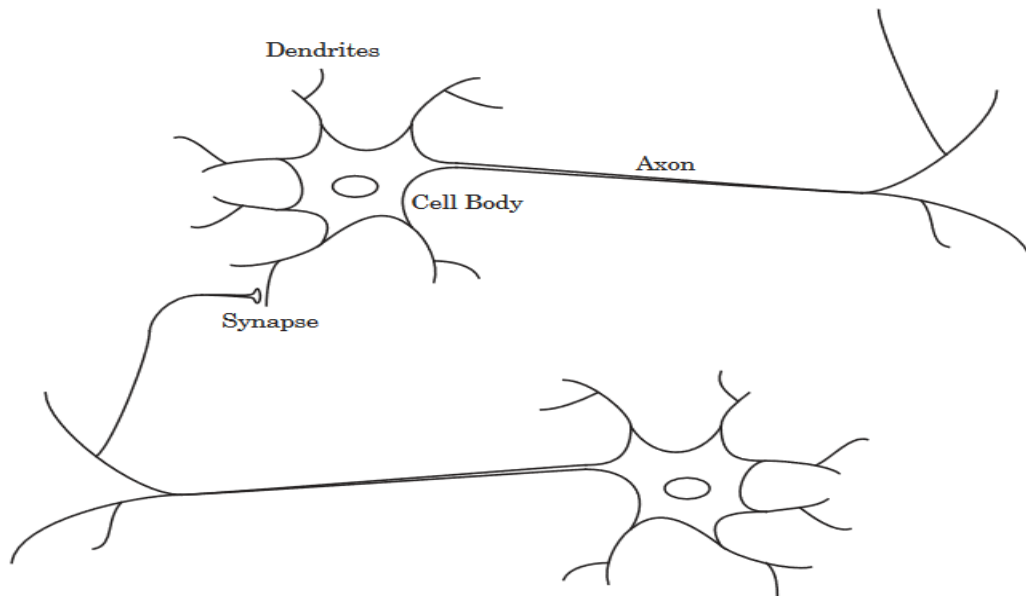


Figure 3.2: Schematic diagram of two biological neurons [19]

Some of the neural structure is defined at birth while other parts developed through learning [19]. By this process, neural structures continue to change throughout life. These changes contribute to the strengthening or weakening of synaptic junctions. Artificial neural networks do not follow the complexity of the brain. However, there are two similarities between biological and artificial neural networks. First, the highly connected building blocks of both networks are simple

computational devices. Second, the connections between neurons regulate the function of the network. The further discussion will introduce primary artificial neuron and will analyze how the combination of such neurons form networks.

3.4 Neuron Model

Figure 3.3 shows the fundamental building block of the single – input neuron for neural networks. In the building block, the scalar input p is multiplied by the scalar weight w to form the product wp . Afterward, the weighted input wp added to the scalar bias b which creates a net input n .

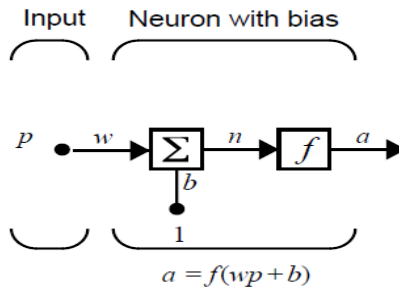


Figure 3.3: Single – input neuron model [19]

Finally, the net input passed through the transfer function f to produce a scalar output a . These three processes are named as – the weight function, the net function, and the transfer function.

A single – input neuron can be extended to a multiple – input neuron (Figure 3.4), where the input and the weights become vectors instead of scalars. For a single R – element input vector where the individual inputs are considered p_1, p_2, \dots, p_R multiplied by weights, the weighted values delivered to the summing junction is Wp . The bias b of the neuron is summed with the weighted inputs to form the net input n .

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \dots + w_{1,R}p_R + b \quad (1)$$

which can be expressed as follows:

$$n = w * p + b \quad (2)$$

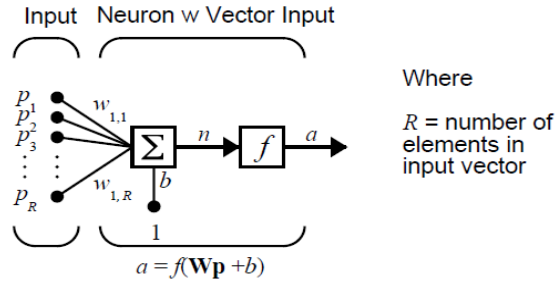


Figure 3.4: Multiple – input neuron [2]

A multiple – input can also extend to a single layer of neurons with multiple inputs (Figure 3.5). Furthermore, a single layer can be extended to multiple layers (see Figure 3.6). The ANN can be built up to perform more complicated tasks. However, too complicated structures can be troublesome in the training which might lead to a poor generalization on test data.

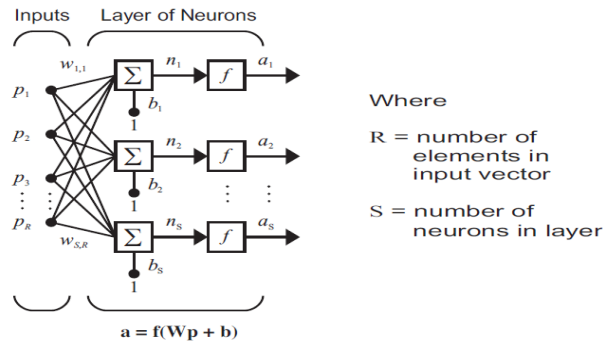


Figure 3.5: A layer of neurons each with multiple inputs and one output [19].

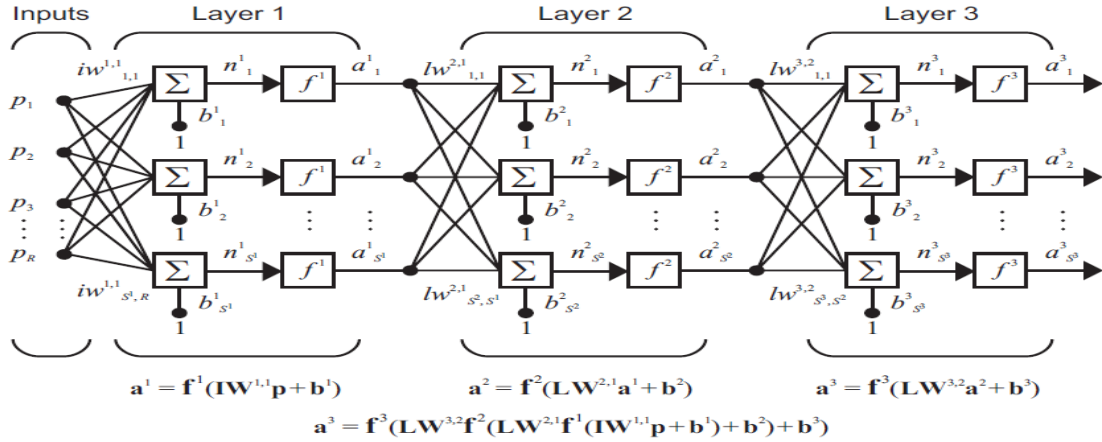


Figure 3.6: Multiple – layers of neuron model [19]. Here only three layers are shown. However, the number of layers can be extended as desired.

There are two major types of neural networks: feedforward and recurrent. All the examples are explained so far are feedforward networks. In this process, the input is piped all the way until the end where the values do not get reused at any stage of the process. On the other hand, recurrent Neural Network(RNN) are ones with feedback, where some of the outputs get connected to inputs throughout the process. Figure 3.7 represents an example of an RNN. In comparison, RNNs are more potent than feedforward NNs, because they acquire the ability to deal with dynamic systems while feedforward NN can only implement static input–output mapping. Feedforward NN, however, has been the focus of much research, resulting in a massive amount of literature.

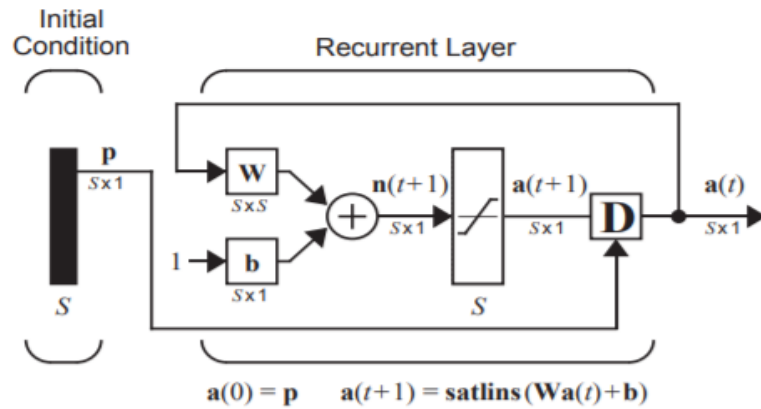


Figure 3.7: An example of a Recurrent neural network(RNN) [19]

3.5 Backpropagation Neural Network

Backpropagation is a learning algorithm of a multilayer neural network with a fixed architecture. It is an abbreviation of ‘Backpropagation Propagation Error’ [20] which is used to train ANN. It executes to minimize the sum squared error between the given target values and the network output values. Backpropagation training associates with two phases – the feedforward and the feedback propagation. In the first phase, the input values are fed to a network assembled with random weights. In this phase, the weights of each neuron remain unchanged. The output values produced by the hidden layer. In the second phase, the error is calculated by subtracting the network predicted output from the input values. Afterward, the error is back – propagated through the hidden layer(s) and minimized which results in updating the weights of the network. This is a repeated process until a satisfactory result is achieved. Figure 3.8 shows the mechanism chart of a backpropagation method.

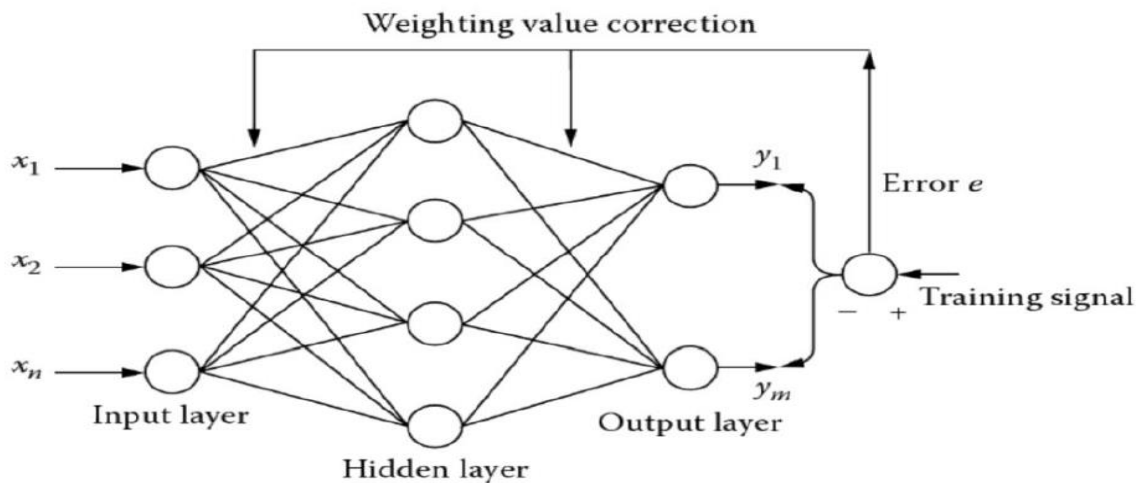


Figure 3.8: Mechanism chart of backpropagation [21]. Here, an error value is created based on ‘ n ’ input neurons and ‘ m ’ output neurons and then backpropagate(s) through the hidden layer. Through this process, the error is minimized by updating the weights of the hidden and output neurons

As explained in section 3.5, the backpropagation algorithm is used to train ANN. Backpropagation has different training processes to minimize the error : Levenberg – Marquardt (LM), Bayesian Regularization (BR), and Scaled Conjugate Gradient (SCG) backpropagations are nothing but different methods used for the optimization technique. Thus, LM, BR, and SCG are explained further in the following sections.

3.5.1 Levenberg – Marquardt Backpropagation

Levenberg – Marquardt(LM) is an optimization algorithm which solves the problem of minimizing non – linear functions [22]. It is also known as the Damped Least Square (DLS) algorithm [23]. LM merges the error backpropagation (EBP) and Gauss – Newton (GN) method. The features of both algorithms i.e. the speed of GN and the stability of EPB help to improve the performance. By comparison, LM is more robust than GN as well faster than the EBP. In this section, the LM algorithm is derived from the EBP and the GN methods. The following derivation is adopted from [22] where p represents the number of patterns i.e. multiple inputs or single inputs, m shows the number of outputs, i and j are weight indices, and k is the index iterations

Equation (3) is defined as the sum of square error which generalizes to all training patterns and network outputs

$$E(x, w) = \frac{1}{2} \sum_{p=1}^P \sum_{m=1}^M e_{p.m}^2 \quad (3)$$

In here, x is the input vector , w is the weight vector and $e_{p,m}$ is the training error which is defined as Equation (4)

$$e_{p.m} = d_{p.m} - o_{p.m} \quad (4)$$

where d is the target output and o is the predicted output vector from the NN.

The steepest descent algorithm uses a gradient g . For this, it uses the first – order derivative of the error function which is defined by Equation (5)

$$g = \frac{\partial^2 E(x, w)}{\partial w} = \left[\frac{\partial E}{\partial w_1} \frac{\partial E}{\partial w_2} \dots \frac{\partial E}{\partial w_N} \right] \quad (5)$$

The value from g can updates the weights by following:

$$w_{k+1} = w_k - \alpha g_k \quad (6)$$

Equation (6) shows the weighted equation where α is the learning constant. The gradient vector will have minimum values around the minimum point. Therefore, very tiny weight change will be completed.

All the gradient components g_1, \dots, g_N are functions of weights where all of them are linearly independent.

$$\begin{cases} g_1 = F_1(w_1, w_2, \dots, w_N) \\ g_2 = F_2(w_1, w_2, \dots, w_N) \\ \cdot \\ \cdot \\ \cdot \\ g_N = F_N(w_1, w_2, \dots, w_N) \end{cases} \quad (7)$$

In Equation (7), F_1, \dots, F_N are nonlinear weights corresponding to the grading components. Expanding each component using the Taylor series and taking the first-order approximation yields Equation (8).

$$\left\{ \begin{array}{l} g_1 \approx g_{1,0} + \frac{\partial g_1}{\partial w_1} \Delta w_1 + \frac{\partial g_1}{\partial w_2} \Delta w_2 + \dots + \frac{\partial g_1}{\partial w_N} \Delta w_N \\ g_2 \approx g_{2,0} + \frac{\partial g_2}{\partial w_1} \Delta w_1 + \frac{\partial g_2}{\partial w_2} \Delta w_2 + \dots + \frac{\partial g_2}{\partial w_N} \Delta w_N \\ \vdots \\ g_N \approx g_{N,0} + \frac{\partial g_N}{\partial w_1} \Delta w_1 + \frac{\partial g_N}{\partial w_2} \Delta w_2 + \dots + \frac{\partial g_N}{\partial w_N} \Delta w_N \end{array} \right. \quad (8)$$

From Equation (5) it can be written as:

$$\frac{\partial g_i}{\partial w_j} = \frac{\partial \left(\frac{\partial E}{\partial w_j} \right)}{\partial w_i} = \frac{\partial^2 E}{\partial w_i \partial w_j} \quad (9)$$

Combining Equation (8) and Equation (9) can produce:

$$\left\{ \begin{array}{l} g_1 \approx g_{1,0} + \frac{\partial^2 E}{\partial w_1^2} \Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_1 \partial w_N} \Delta w_N \\ g_2 \approx g_{2,0} + \frac{\partial^2 E}{\partial w_2 \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_2^2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_2 \partial w_N} \Delta w_N \\ \vdots \\ g_N \approx g_{N,0} + \frac{\partial^2 E}{\partial w_N \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_N^2} \Delta w_N \end{array} \right. \quad (10)$$

For finding the minimum of a total error function E , each element of the gradient vector is equalized to zero value:

$$\left\{ \begin{array}{l} 0 \approx g_{1,0} + \frac{\partial^2 E}{\partial w_1^2} \Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_1 \partial w_N} \Delta w_N \\ 0 \approx g_{2,0} + \frac{\partial^2 E}{\partial w_2 \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_2^2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_2 \partial w_N} \Delta w_N \\ \vdots \\ 0 \approx g_{N,0} + \frac{\partial^2 E}{\partial w_N \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_N^2} \Delta w_N \end{array} \right. \quad (11)$$

Combining Equation (5) and Equation (11) can be written as:

$$\left\{ \begin{array}{l} -\frac{\partial E}{\partial w_1} = -g_{1,0} \approx \frac{\partial^2 E}{\partial w_1^2} \Delta w_1 + \frac{\partial^2 E}{\partial w_1 \partial w_2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_1 \partial w_N} \Delta w_N \\ -\frac{\partial E}{\partial w_2} = -g_{2,0} \approx \frac{\partial^2 E}{\partial w_2 \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_2^2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_2 \partial w_N} \Delta w_N \\ \vdots \\ -\frac{\partial E}{\partial w_N} = -g_{N,0} \approx \frac{\partial^2 E}{\partial w_N \partial w_1} \Delta w_1 + \frac{\partial^2 E}{\partial w_N \partial w_2} \Delta w_2 + \dots + \frac{\partial^2 E}{\partial w_N^2} \Delta w_N \end{array} \right. \quad (12)$$

There are N equations for N parameters. Thus, all the Δw_i can be calculated, and the weights can be updated iteratively. Equation (12) can be written as:

$$\begin{bmatrix} -g_1 \\ -g_2 \\ \vdots \\ -g_N \end{bmatrix} = \begin{bmatrix} -\frac{\partial E}{\partial w_1} \\ -\frac{\partial E}{\partial w_2} \\ \vdots \\ -\frac{\partial E}{\partial w_N} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \dots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \times \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_N \end{bmatrix} \quad (13)$$

The square matrix in Equation (13) is called the Hessian matrix:

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ & & \ddots & \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \quad (14)$$

From the above equations, one can get

$$-g = H \Delta w \quad (15)$$

and

$$\Delta w = -H^{-1} g \quad (16)$$

Thus, the weights can be updated using Equation (17)

$$w_{k+1} = w_k - H_k^{-1} g_k \quad (17)$$

This method needs computation of the second order derivative for every component which might be complicated. Therefore, the Gauss – Newton Algorithm simplifies the steps by introducing the Jacobian matrix J , which is shown as:

$$J = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \dots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \dots & \frac{\partial e_{1,2}}{\partial w_N} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \dots & \frac{\partial e_{1,M}}{\partial w_N} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\partial e_{P,1}}{\partial w_1} & \frac{\partial e_{P,1}}{\partial w_2} & \dots & \frac{\partial e_{P,1}}{\partial w_N} \\ \frac{\partial e_{P,2}}{\partial w_1} & \frac{\partial e_{P,2}}{\partial w_2} & \dots & \frac{\partial e_{P,2}}{\partial w_N} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\partial e_{P,M}}{\partial w_1} & \frac{\partial e_{P,M}}{\partial w_2} & \dots & \frac{\partial e_{P,M}}{\partial w_N} \end{bmatrix} \quad (18)$$

By integrating Equations (3) and (5) the following equation can be found:

$$g = \frac{\partial E}{\partial w_i} = \frac{\partial \left(\frac{1}{2} \sum_{p=1}^P \sum_{m=1}^M e_{p,m}^2 \right)}{\partial w_i} = \sum_{p=1}^P \sum_{m=1}^M \left(\frac{\partial e_{p,m}}{\partial w_i} e_{p,m} \right) \quad (19)$$

By combining Equations (18) and (19) one can produce the relationship between the gradient matrix and the Jacobian matrix.

$$g = Je \quad (20)$$

In Equation (20), e is the error vector which is given by the following equation:

$$e = \begin{bmatrix} e_{1,1} \\ e_{1,2} \\ \cdot \\ \cdot \\ e_{1,M} \\ \cdot \\ \cdot \\ e_{P,1} \\ e_{P,2} \\ \cdot \\ \cdot \\ e_{P,M} \end{bmatrix} \quad (21)$$

By subtracting Equations (3) and (4), the Hessian matrix can be formed. The equation is described by Equation (22)

$$h_{i,j} = \frac{\partial^2 E}{\partial w_i \partial w_j} = \frac{\partial^2 \left(\frac{1}{2} \sum_{p=1}^P \sum_{m=1}^M e_{p,m}^2 \right)}{\partial w_i \partial w_j} = \sum_{p=1}^P \sum_{m=1}^M \left(\frac{\partial e_{p,m}}{\partial w_i} \frac{\partial e_{p,m}}{\partial w_j} + S_{i,j} \right) \quad (22)$$

In here, $S_{i,j}$ is:

$$S_{i,j} = \sum_{p=1}^P \sum_{m=1}^M \left(\frac{\partial e_{p,m}}{\partial w_i} \frac{\partial e_{p,m}}{\partial w_j} \right) \quad (23)$$

$S_{i,j}$ is very small in the basic assumption of Newton's method. That is why the relationship

between the Hessian matrix H and the Jacobian matrix J is:

$$H \approx J^T J \quad (24)$$

Comparing Equations (17), (20), and (24), the updated weighted formula is given by:

$$w_{k+1} = w_k - (J_k^T J_k)^{-1} J_k e_k \quad (25)$$

The Levenberg – Marquardt algorithm proposes a new approximation to the Hessian matrix. In Equation (26), the approximation makes sure that $J^T J$ is always invertible, where μ is the combination coefficient and I is the identity matrix. The new approximation makes sure that the main diagonal of the Equation (26) is positive. Therefore, the approximation matrix will always be invertible.

$$H \approx J^T J + \mu I \quad (26)$$

Combining Equations (25) and (26) produces the updated weights for the Levenberg –Marquardt method.

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k \quad (27)$$

As it is already mentioned in this section, LM method blends both GN and EBP methods, this process can be observed when the combination coefficient is varied from very small to large values. If μ is close to zero, then Equations (27) turns into (25) which is the Gauss – Newton's algorithm. On the other hand, if μ is very large it becomes the learning coefficient of the steepest descent method.

LM is a fast and stable training method. It uses the Jacobian matrix; performance is a Mean of Squared Errors (MSE) or a Sum of Squared Errors (SSE). It limits the performance function which may not lead to the best generalization. In the following, the Bayesian regularization backpropagation method is explained.

3.5.2 Bayesian Regularization Backpropagation

Overfitting is one of the most significant issues of training neural networks. For NN's, there should be fewer parameters than there are data points in order to train a network [19]. Therefore, the concept is that if the network is too complicated, it might learn unnecessary patterns which might not be able to generalize. There are many approaches to this problem. However, the author of [19] suggests two methods for robust terms in practicality – early stopping and regularization. These two approaches restrict the complexity of the network by reducing the magnitude of the weights. Among them, regularization is the kind where Bayesian algorithm lie.

Thomas Bayes is the founder of the concept of the Bayesian Regularization (BR) algorithm. His famous Bayes' theorem computes the conditional probability of an event 'A' given a condition 'B' has occurred. The formula can recall as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (28)$$

In 1992, David MacKay developed an approach to use the Bayes' theorem for neural network training [24]. This approach brings neural network training to the probability framework. Beale, Demuth and Hagan divides this framework into two levels. The analysis described in the following section is adapted from [19].

Level I calculation starts with the assumption that the weights of an artificial neural network are random variables. Then the conditional probability of the weights form the data points. In the end, the weights which maximize the formula are found and chosen.

This probabilistic relation can be described as follows:

$$P(x|D, \alpha, \beta, M) = \frac{P(D|x, \beta, M)P(x|\alpha, M)}{P(D|\alpha, \beta, M)} \quad (29)$$

In Equation (29), x represents the vector containing all the weights and biases in the network, α and β are the parameters related with the density functions $P(D|x, \beta, M)$ and $P(x|\alpha, M)$, D is the input/output, and M is the architecture of the designed network.

To clarify this concept, each term in Equation (29) is further described. The probability density of the data $P(D|x, \beta, M)$, given the weight vector x , the parameter β and the network model M can be represented as given in Equation (30).

$$P(D|x, \beta, M) = \frac{1}{Z_D(\beta)} e^{-\beta E_D} \quad (30)$$

where

$$\beta = \frac{1}{2\sigma_\varepsilon^2} \quad (31)$$

Here, σ_ε^2 is the variance of the noise source, E_D is the squared error and

$$Z_D(\beta) = (2\pi\sigma_\varepsilon^2)^{\frac{N}{2}} = \left(\frac{\pi}{\beta}\right)^{\frac{N}{2}} \quad (32)$$

The weights are selected to maximize because maximizing the function requires minimizing the squared error E_D .

The second term in Equation (29) is $P(x|\alpha, M)$. It consists of the knowledge of the weights before collecting the data. This term is known as prior density. If the weights consider very small (close to zero), a zero – mean prior density function can be selected:

$$P(x|\alpha, M) = \frac{1}{Z_w(\alpha)} e^{-\beta E_w} \quad (33)$$

where

$$\alpha = \frac{1}{2\sigma_w^2} \quad (34)$$

Here, σ_w^2 is the variance of the weights, E_w is the sum squared of the weights and :

$$Z_w(\alpha) = (2\pi\sigma_w^2)^{\frac{n}{2}} = \left(\frac{\pi}{\alpha}\right)^{\frac{n}{2}} \quad (35)$$

where n is the number of weights and biases in the network.

The last term in Equation (29) is called the evidence $P(D|\alpha, \beta, M)$ which does not involve the weights vector x . However, the goal is to maximize Equation (28). Thus, it is not concerned at this point.

Putting Equations (29), (30) and (33) together, the density function can be rewritten as:

$$\begin{aligned} P(x|D, \alpha, \beta, M) &= \frac{\frac{1}{Z_w(\alpha)} \frac{1}{Z_D(\beta)} \exp[-(\beta E_D + \alpha E_w)]}{\text{Normalizing factor}} \\ &= \frac{1}{Z_w(\alpha, \beta)} \exp(-F(x)) \end{aligned} \quad (36)$$

Where $Z_F(\alpha, \beta)$ is the function α and β . $F(x)$ is the regularization performance index which can be defined as $F(x) = \beta E_D + \alpha E_w$.

Level II of the framework estimates the parameter α and β from the data for analysis. The level II framework equation can be written as:

$$P(\alpha, \beta | D, M) = \frac{P(D | \alpha, \beta, M) P(\alpha, \beta | M)}{P(D | M)} \quad (37)$$

This formula is almost the same as Equation (29). Following the same logic, Equation (36) can be maximized by maximizing $P(D | \alpha, \beta, M)$. However, this term is identical to the normalizing factor of Equation (29). Thus, $P(D | \alpha, \beta, M)$ can be solved as:

$$\begin{aligned} P(D | \alpha, \beta, M) &= \frac{\left[\frac{1}{Z_D(\beta)} \exp(-\beta E_D) \right] \left[\frac{1}{Z_w(\alpha)} \exp(-\alpha E_w) \right]}{\frac{1}{Z_F(\alpha, \beta)} \exp(-F(x))} \\ &= \frac{Z_F(\alpha, \beta)}{Z_D(\beta) Z_w(\alpha)} \end{aligned} \quad (38)$$

From Equations (32) and (35) $Z_D(\beta)$ and $Z_w(\alpha)$ can be found. However, $Z_F(\alpha, \beta)$ can be expressed utilizing the Taylor series expansion:

$$F(x) \approx F(x^{MP}) + \frac{1}{2} (x - x^{MP})^T H^{MP} (x - x^{MP}) \quad (39)$$

Here, H^{MP} is the Hessian matrix of $F(x)$ which is evaluated at x^{MP} . Substituting Equation (39) into Equation (36) yields the following equation:

$$Z_F(\alpha, \beta) \approx (2\pi)^{\frac{n}{2}} (\det((H^{MP})^{-1}))^{\frac{1}{2}} \exp(-F(x^{MP})) \quad (40)$$

Also, substituting Equations (40) and (38) brings the optimum solution for α and β at the minimum point:

$$\alpha^{MP} = \frac{\gamma}{2E_w(x^{MP})} \text{ and } \beta^{MP} = \frac{N - \gamma}{2E_D(x^{MP})} \quad (41)$$

where n is the total number of parameters in the network (weights and biases) and $\gamma = n - 2\alpha^{MP} \text{tr}(H^{MP})^{-1}$ is the efficient number performance.

3.5.3 Scaled Conjugate Gradient Backpropagation

The Scaled Conjugate Gradient (SCG) algorithm is based on ‘conjugate gradient methods’ where the large – scale problem is handled by a group of optimization techniques. SCG is similar to the LM method in the sense that it provides quadratic termination without computing the Hessian.

Considering the mathematical calculation, a set of vectors $\{p_k\}$ is mutually conjugate with regards to a positive definite Hessian Matrix H , if :

$$p_k^T H p_j = 0 \quad k \neq j \quad (42)$$

Identical to orthogonal vectors, there are an infinite number of conjugate vectors, includes the Hessian Matrix spanning an n -dimensional space. Thus, the conjugacy condition in Equation (42) must be stated without H . Recalling that for quadratic functions:

$$\nabla F(x) = Hx + d \quad (43)$$

$$\nabla^2 F(x) = H \quad (44)$$

By combining the formulas the change in gradient at iteration $k+1$:

$$\nabla g_k = g_{k+1} - g_k = (Hx_{k+1} + d) - (Hx_k + d) = H\Delta x_k \quad (45)$$

$$x_k = (x_{k+1} - x_k) = \alpha_k p_k \quad (46)$$

Here, α_k is minimizing $F(x)$ in the direction p_k .

The equation for conjugacy condition can be related as:

$$\alpha_k p_k^T H p_j = \Delta x_k^T H p_j = \Delta g_k^T p_j = 0 \quad k \neq j \quad (47)$$

Equation (47) represents the conjugacy condition with respect to change in the gradient at successive iterations. There are infinite number of conjugate vectors because p_0 is arbitrary and p_1 can be any vector that is orthogonal to Δg_0 . The equation can be expressed in the steepest decent direction:

$$p_0 = -g_0 \quad (48)$$

Next, a vector p_k that is orthogonal to $\{\Delta g_0, \Delta g_1, \dots, \Delta g_{k-1}\}$ can be simplified to iterations of the form [25]:

$$p_k = -g_k + \beta_k p_{k-1} \quad (49)$$

Here, β_k is scalar which produce equivalent results when it comes to quadratic functions [25].

The most common methods are [25]:

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{\Delta g_{k-1}^T p_{k-1}} \quad (50)$$

$$\beta_k = \frac{\Delta g_k^T g_k}{\Delta g_{k-1}^T g_{k-1}} \quad (51)$$

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{\Delta g_{k-1}^T g_{k-1}} \quad (52)$$

CHAPTER 4: EXPERIMENTAL SETUP

4.1 Surface EMG Circuit Design

This chapter describes building a circuit board to detect surface EMG (sEMG) signals. The circuit board consists of three different stages. All the stages are explained in the following documentation.

4.1.1 The first stage

This stage feeds the sEMG signal to the non-inverting terminal of the LM324 instrumentation amplifier. It compares two inputs and multiplies the difference 100 to increase the strength of the sEMG signal. Figure 4.1 shows the circuit diagram of the instrumentation amplifier. The figure contains four resistors which suggested values are $R1 - 2K\Omega$, $R2 - 100K\Omega$, $R3$, and $R4 - 10K\Omega$. The quad-operational amplifier model is a LM324NFS – ND. The output values from the first stage vary from ± 0.01 to ± 0.09 mV. The signal from the instrumentation amplifier is afterwards feed to the notch filter.

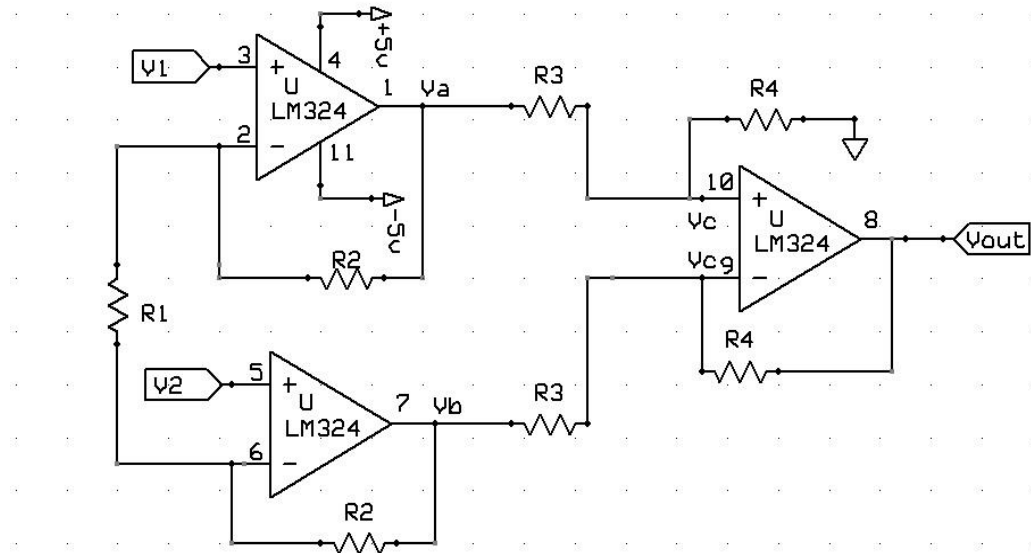


Figure 4.1: Instrumentation Amplifier [27]

4.1.2 Second Stage

The second stage consists of a notch filter which eliminates interference caused by the environment, i.e., electronic components using a power supply at 60Hz. A UAF42 circuit is used to build a notch filter by adding a set of corresponding resistors to tune the filter to a 60Hz frequency. The calculation of the resistor values shown below-

$$f_{Notch} = \sqrt{\left(\frac{A_{LP}}{A_{HP}} \cdot \frac{R_{Z2}}{R_{Z1}}\right)} * f_o \quad (53)$$

In the equation A_{LP} represents the gain from the input to the low-pass filter, A_{HP} indicates the gain from the input to high – pass filter and f_o is the fundamental frequency. R_{Z1} and R_{Z2} are the resistor values which are each $2K\Omega$. Consistently, the product of $\left(\frac{A_{LP}}{A_{HP}} * \frac{R_{Z2}}{R_{Z1}}\right)$ value is one, thus the notch filter frequency is equal to the fundamental frequency. Figure 4.2 reparents a UAF42 circuit diagram.

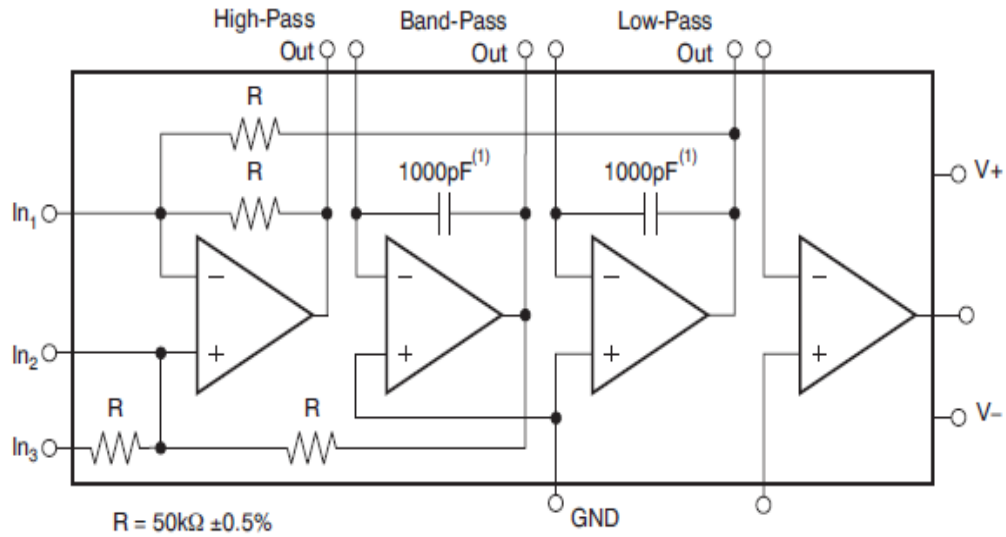


Figure 4.2: UAF42 circuit [28]

The -3dB drop-off occurs at the cut off frequency of the filter which correlates to half power level and the attenuation. The -3dB drop-off estimated as:

$$BW_{-3dB} = \frac{f_{notch}}{Q} \quad (54)$$

Here, Q represents the quality factor which can be adjusted by

$$R_Q = \frac{R}{Q-1} \quad (55)$$

Figure 4.3 shows the notch filter with the combination of resistors $R_q - 4.99K\Omega$, R_{F1} and $R_{F2} - 2.65 M\Omega$, $R_{Z3} - 12.1K\Omega$ and UAF42AP – ND – the universal active filter used for 60Hz compensation. The output from the notch filter is then feed to the bandpass filter, which encompass the next stage of the suggested design.

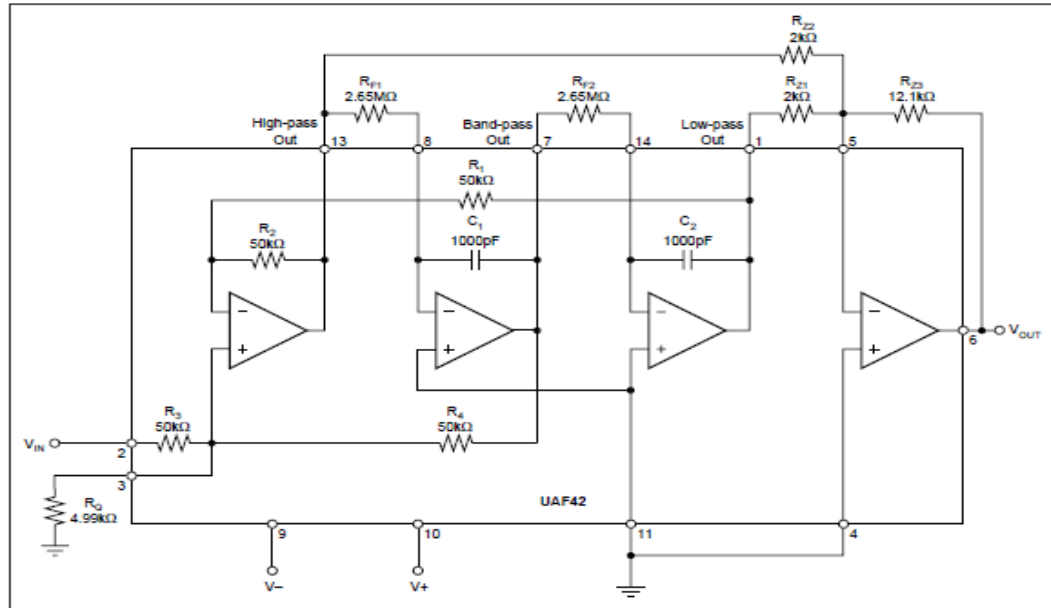


Figure 4.3: A notch filter [27]

4.1.3 Third Stage

A Chebyshev type II 0.1dB passband is used in this stage. It is an active filter for roll off used to describe the steepness of a transmission function with the frequency. This filter has a unique feature in eliminating the error between the idealized and the actual filter. The Chebyshev type II represents the stopband ripple which consists of low-pass and high-pass filter. Figure 4.4 shows a design frequency of 450Hz high – pass filter.

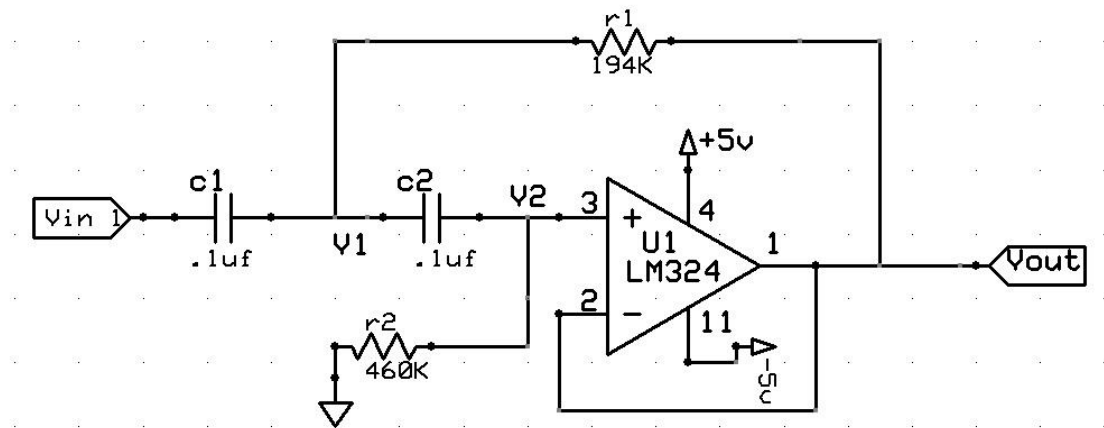


Figure 4.4: High – pass filter [27]

The output from the High – pass filter is passed through a 4Hz low – pass filter. The outputs of the low – pass filter range from $\pm 0.01V - \pm 0.09V$. Figure 4.5 represents the suggested low-pass filter. The resistors values and the operational amplifier model is shown in the figure.

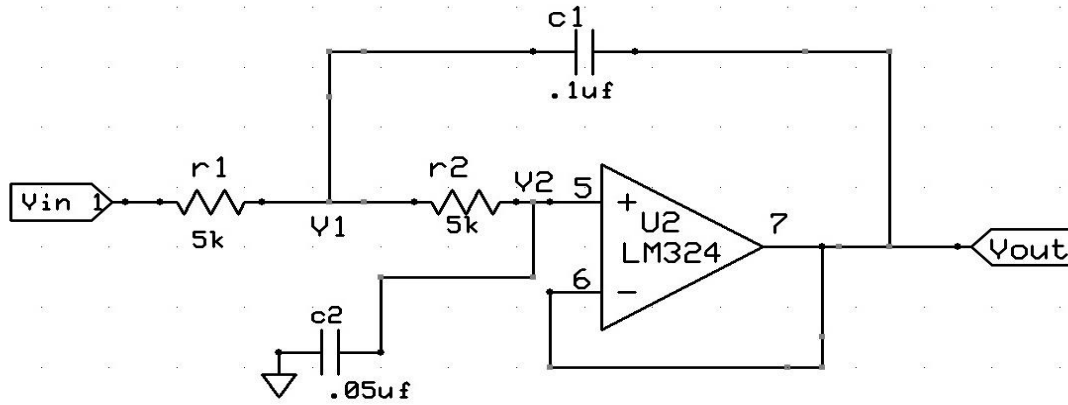


Figure 4.5: Low – pass filter [27]

The output from the lowpass filter is the final process to achieve sEMG signal. The output value from the low pass filter is not being able to measure regulate voltage because of Arduino. Therefore amplification of the signal is required. For this, a buffer amplifier is used in combination with voltage shift circuit to make suitable with a typical microcontroller. Figure 4.6 represents the buffer and voltage shift circuit where $R1$ & $R3 = 2K\Omega$, $R2 = 27K\Omega$, $R4$ & $R5 = 33K\Omega$, $R6$ is a potentiometer range of $10K\Omega$ and LM324NFS – ND is quad – operational amplifier.

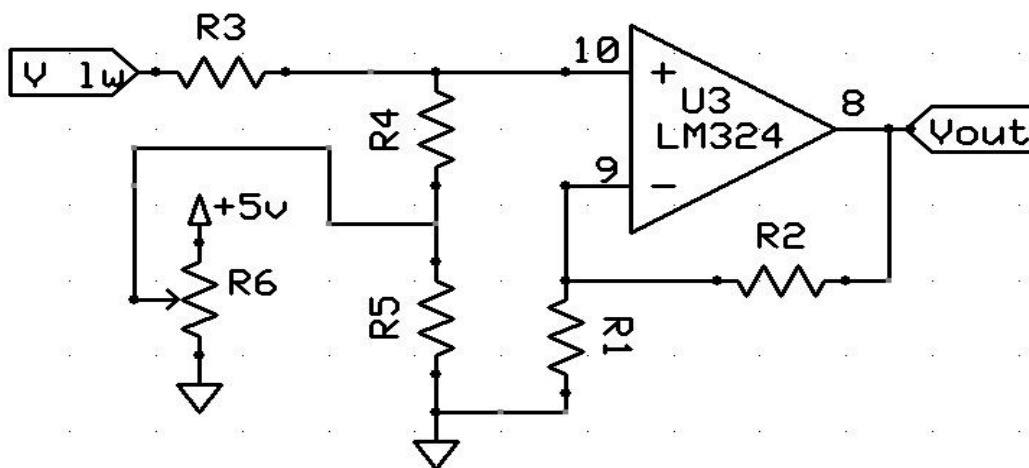


Figure 4.6: Buffer amplifier along voltage shift circuit [27]

The final implementation is achieved by doing a compact design of a PCB (Printed Circuit Board). This PCB design is small and portable for using mobile application. Figure 4.7 shows a final implementation of a PCB design for achieving sEMG signal.

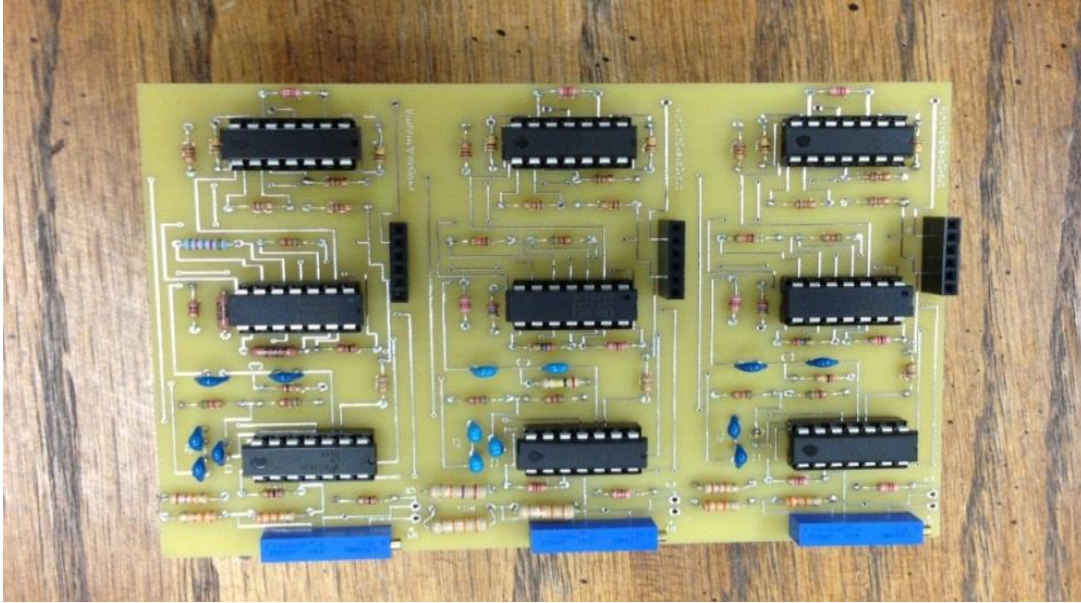


Figure 4.7: Final implementation of a PCB model [27]

The implemented PCB design needs $\pm 5V$ of battery supply to power the circuit. A power supply circuit diagram and connection are shown in Figure 4.8 and Figure 4.9. The power supply employed three regulators, U3 is a 10V regulator, U1 and U2 are $\pm 5V$ regulators. The battery from an old laptop lithium – ion batteries supplies approximately 23V.

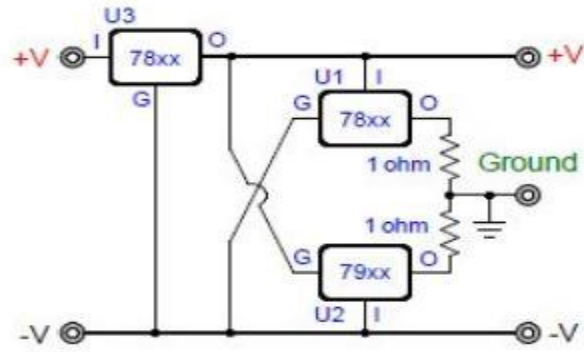


Figure 4.8: Power supply diagram

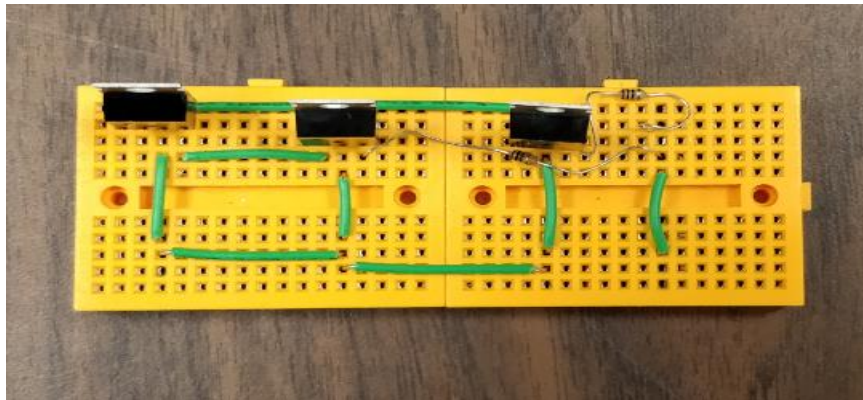


Figure 4.9: Power supply circuit connection

4.2 Shortcomings of PCB and the power supply

Through testing the sEMG channel box some shortcomings were noted. The sEMG sensor box has ten channel connections as shown in Figure 4.10.

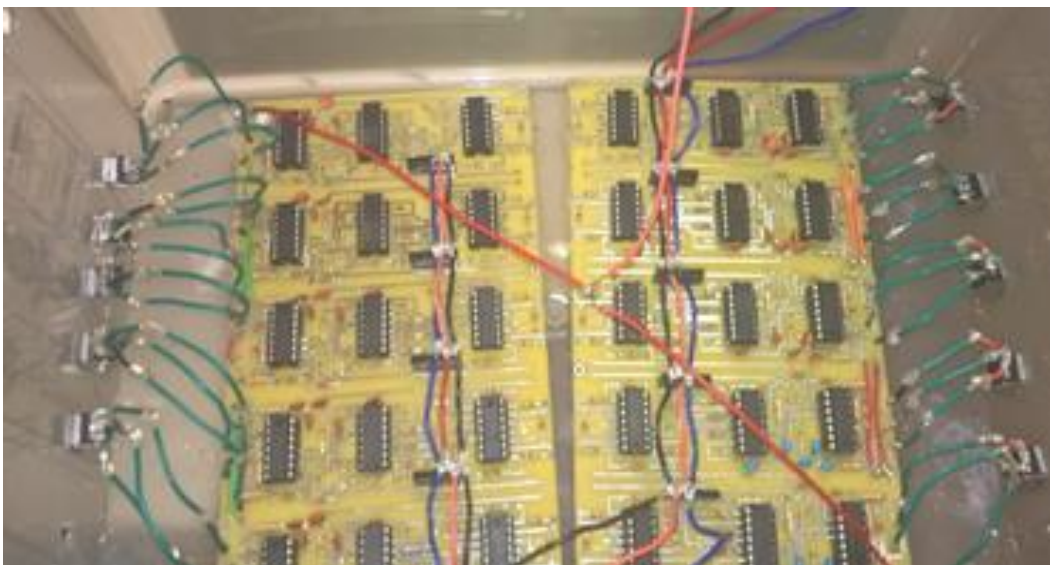


Figure 4.10: sEMG channel box contains ten channels [29]

However, not all the times the sEMG signals appeared in perfection. Most of the times the channel box showed some noisy output. Therefore, some additional tasks were made to acquire the desired sEMG signal. At first, there was only one power supply for the ten channels which is changed by adding a second two power supply resulting in one power supply for the right side five channels and the other one is connected to left side five channels. With the proposed change in power supplies, however, the output seemed to be the same as previously noted, which is a noisy signal. After investigating the circuit wire connections and the electrode connections, it was discovered that the channel box has ground connection issues. The breadboard ground connection might get loose while experimenting. Besides of that, it is also observed that the electrodes which are connected to the PCB through poles, are also creating signals that are noisy. Thus, for this purpose, a new power supply was built by using the solderable breadboard, which can provide for a robust ground connection. Figure 4.11 shows a new power supply using the solderable board. On the other hand, the poles on the sensor board have been removed and the electrodes were soldered directly to the PCB. Figure 4.12 represents the new set – up for achieving ideal sEMG signals without much noise.

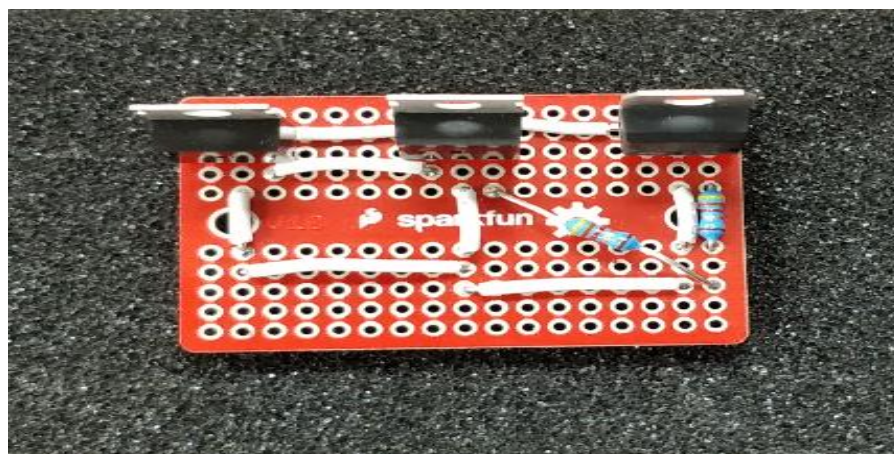


Figure 4.11: New power supply connection using the solderable board

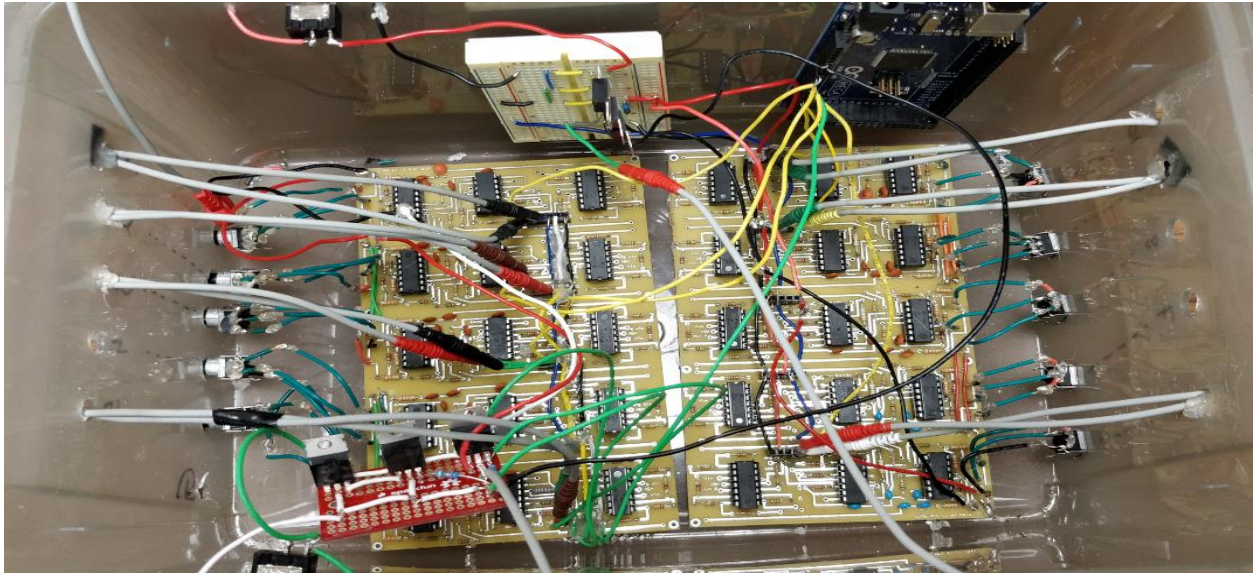


Figure 4.12: New sEMG channel box setup

The left side of the box contains five channels which has a separated solderable (red color) power supply box. Also, on the left side in Figure 4.12 shows the poles being removed and the electrodes being directly soldered on the PCB. On the other side, the right side which has another five channels connected to a breadboard (white color) power supply box. There wasn't any ground issue on the right side channels, thus the poles and power supply box did not change.

4.3 Electrodes Placement and establishing sEMG signal

Once the circuit setup is completed, the electrodes are confirmed working and wired correctly, the sEMG signal can be produced by controlling the hand muscles. Figure 4.13 and Figure 4.14 shows how the two hand motions should perform to activate the inside and outside forearm muscles with the least amount of effort. In standard muscle control method, it will be either open or close. As an example, if the outer forearm muscle is tensed the hand should open and when the inner forearm muscle has tensed the hand should close.



Figure 4.13: Inner forearm muscle movement position

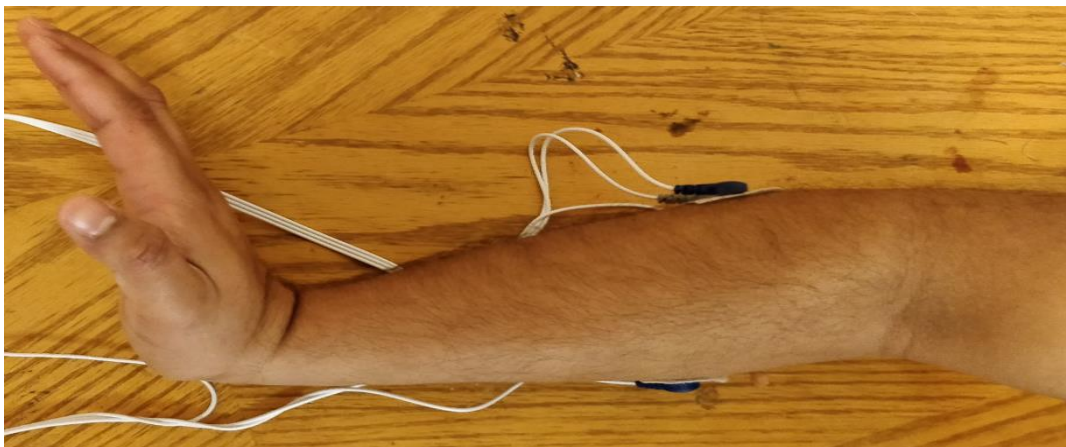


Figure 4.14: Outer forearm muscle movement position

By following the steps for Simulink™ setup which is discussed in chapter 3, the sEMG signals can easily be observed. Figure 4.15 and Figure 4.16 show the output from two channels for two

different motions. Figure 4.15 represents the inner forearm muscle movements when the hand is open. However, Figure 4.16 represents the outer forearm muscle movements while the hand close.

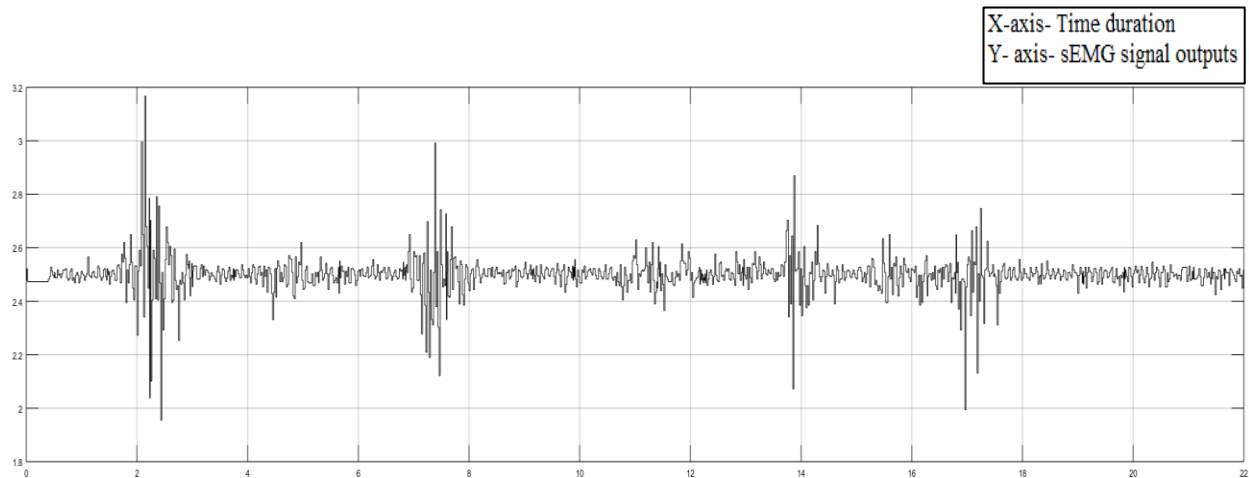


Figure 4.15: sEMG signal for inner forearm muscle movement

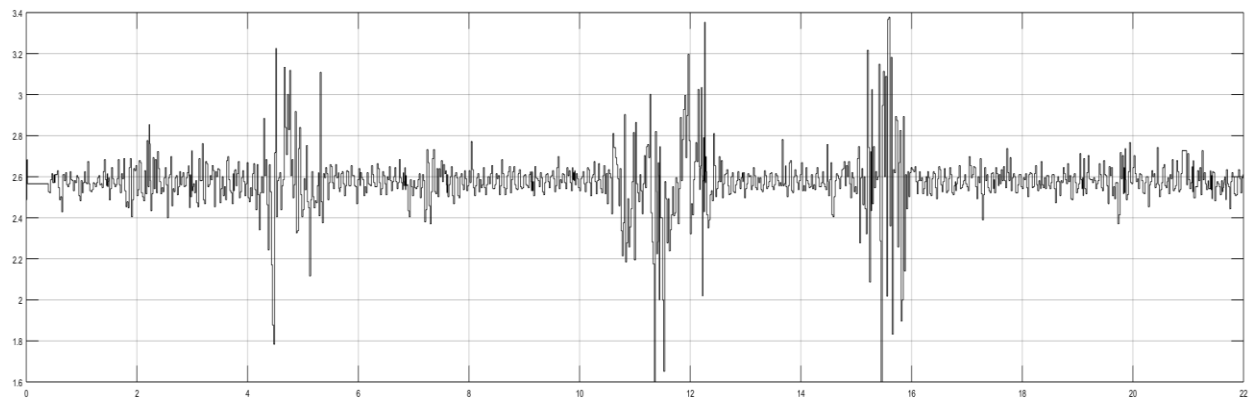


Figure 4.16: sEMG signal for outer forearm muscle movement

By observing the Figure 4.15 and Figure 4.16, it is noticeable the two sEMG signals have opposite pattern. The reason for this, the inner forearm muscle movement, and the outer forearm muscle movement contradicts each other movement positions. That is why in Figure 4.15 has the peak value at one instance in time in Figure 4.16 it does not show any changes.

CHAPTER 5: IMPLEMENTATION AND VALIDATION OF ANN

Chapter 3 gives the idea about the mathematical background of an Artificial Neural Network (ANN). In this chapter, ANN algorithm is going to be used for training of the network. For this purpose, implementation of Simulink™ models are explained, which is given in the following three parts: -

1. Implementation of real – time classification Simulink™ model.
2. Training the ANN model based on real – time classification.
3. Development of identification algorithm for identifying motion intend using real – time ANN models.

5.1 Implementation of real – time classification Simulink™ model

Figure 5.1 represents the real – time classification Simulink™ block model. Figure 5.1 consists of several implementation of the presented methods. The first part contains the connection from receiving the signal from Arduino to Simulink™ through serial port including the conversion of the ASCII code into a bit number between 0 to 1023 and afterward in the range of 0 to 5V. The second part shows the buffer blocks for calculating the overlapping sliding window. The third part in the block represents the different classification algorithms and their final outputs. All the parts are explained elaborately in the further sections.

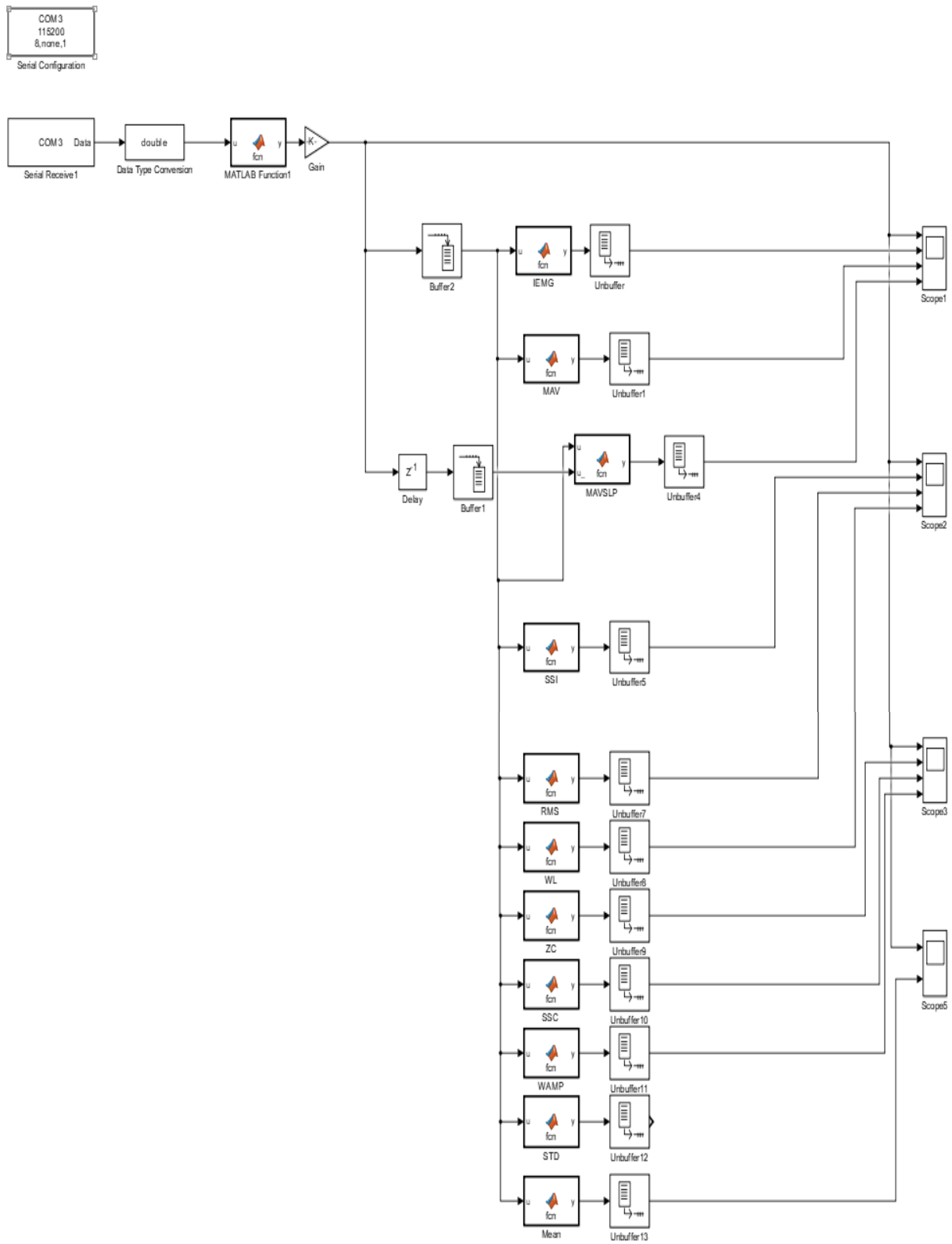


Figure 5.1:Real – time classification Simulink™ block model

5.1.1 Connecting Arduino to Simulink™ through serial port:

Instead of using traditional Arduino analog input Simulink™ block, an Arduino code has been created to consider the Arduino controls as a data acquisition system. By uploading the code, Arduino can pass the signal through serial communication to feed the Simulink™ block running on the PC. Usually, Arduino does not have much AVR memory, and because of its lower processor speed, it is tough to send a lot of data at average speed to the Simulink™ block running on the Arduino. Also, it seems impossible to pass the signal of average speed when the overlapping has increased. However, connecting the Arduino through the serial port helps with the system to speed up the signal processing. Thus, for this reason, at first, an Arduino code has been created. The code is made to read the 1024 bit from the one analog input. The analog input will show four bits which can be seen (Figure 5.2) in the serial monitor.

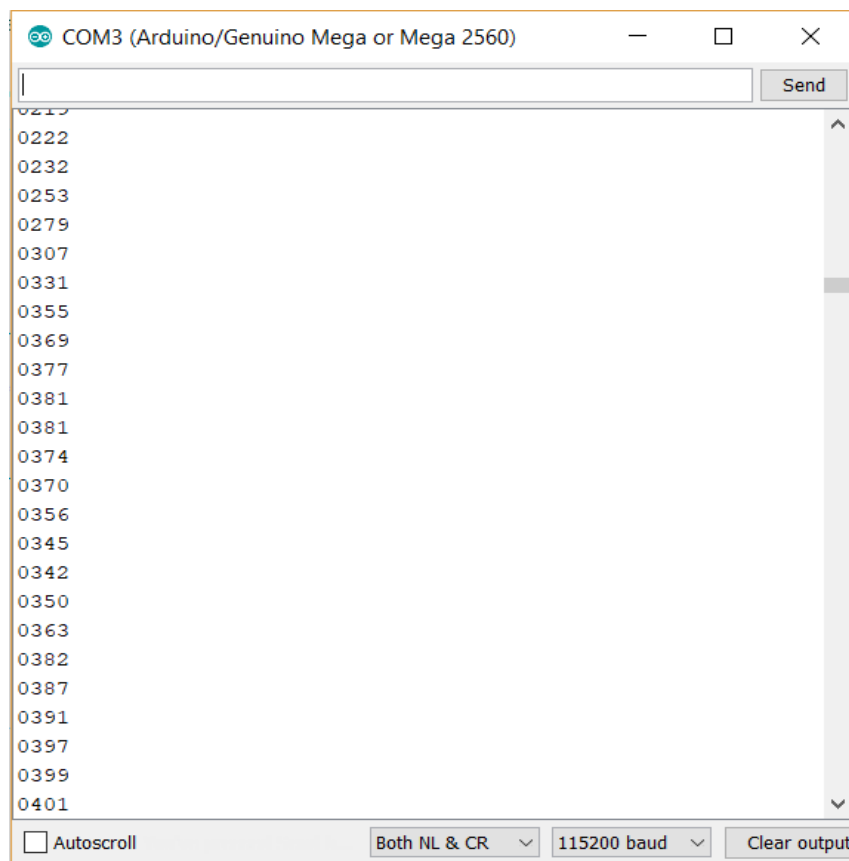


Figure 5.2: 4 bits of value from the serial monitor

The universal serial receive block has been chosen instead of the Arduino Serial receive block to receive the signal. Otherwise, the standard Simulink™ block will force to use the Arduino processor which makes the output slow and impossible to process the signals. For configuring the serial receiver block, the corresponding has been selected as [4 1] because there is a total of four bits which are coming through serial communication. These four bits considered for the bit range of 0 to 1023. The sampling time is selected as 0.067 to adjust the speed between Arduino and Simulink™ communication. Besides, another serial configuration has been added where the baud rate is selected as 115,200 and the number of data bits has been chosen as four bits. Figure 5.3 represents the configuration parameters for serial receive and serial configuration parameters. Afterwards, data type conversion block is used to convert the value. However, the output of real values can be observed when the ASCII code is converted into 0 to 1023 – bit values. Thus, for this, the values have been passed through a MATLAB® code to convert the ASCII value into bit values. Then those bit values are fed to a gain block to convert this signal into a voltage value which ranges from 0 to 5V. Through this process, the analog input signal can be achieved by a Simulink™ block running on the PC without using Arduino processor.

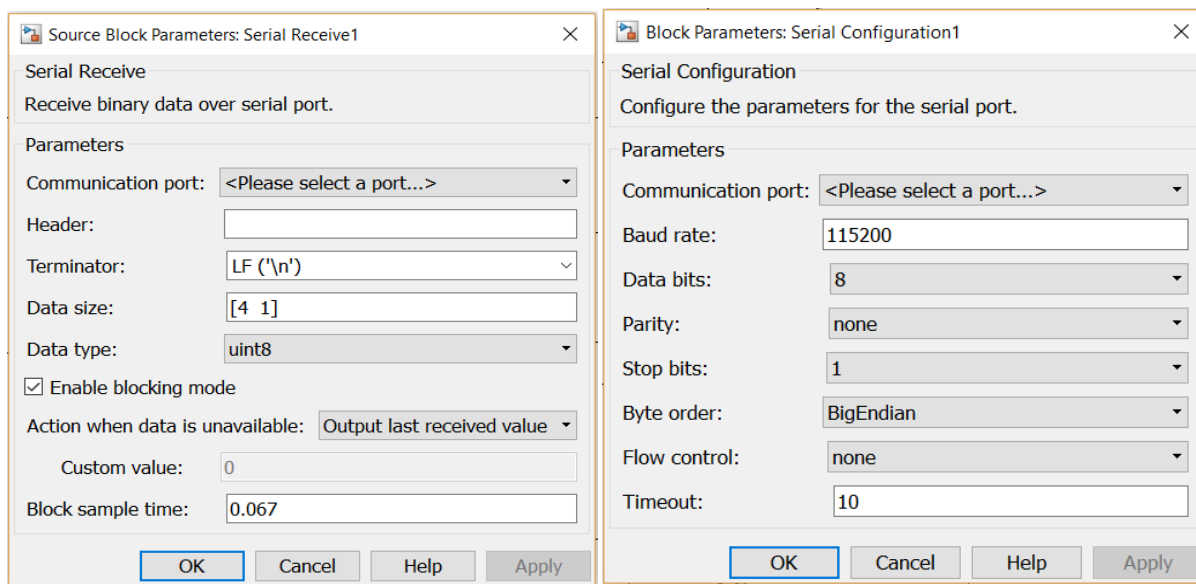


Figure 5.3: Parameters selection for (a) serial receive block and (b) serial configuration

5.1.2 Overlapping Sliding Window:

Before the classification process has begun, 50 blocks of delays are built with an overlapping sliding window. For doing so, a buffer block is used which will create 50 blocks of delay with overlapping sliding window. However, before doing that, the buffer block has been checked to confirm that it is working as a delay block and overlapping with each other. The examination of buffer block and overlapping calculation is described in further explanations.

5.1.3 Checking the Buffer blocks for creating delays:

For checking the buffer blocks, a triangular signal has been created with a sampling time of .01seconds. To observe the delays the output buffer size is selected as 100. Thus, this system can assume that after one second the buffer will start showing its output signal. The Simulink™ block & its output signal are shown in Figure 5.4 and Figure 5.5.

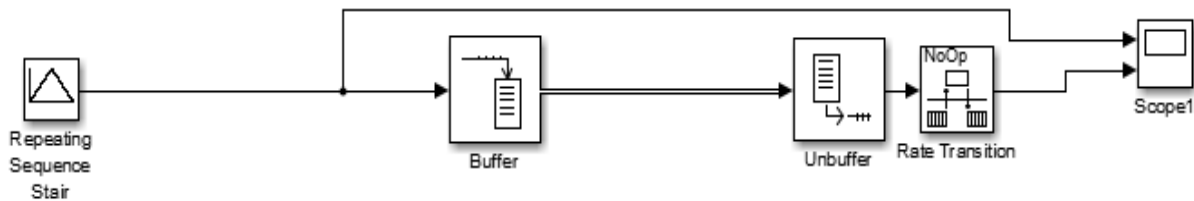


Figure 5.4: Checking delays using buffer block

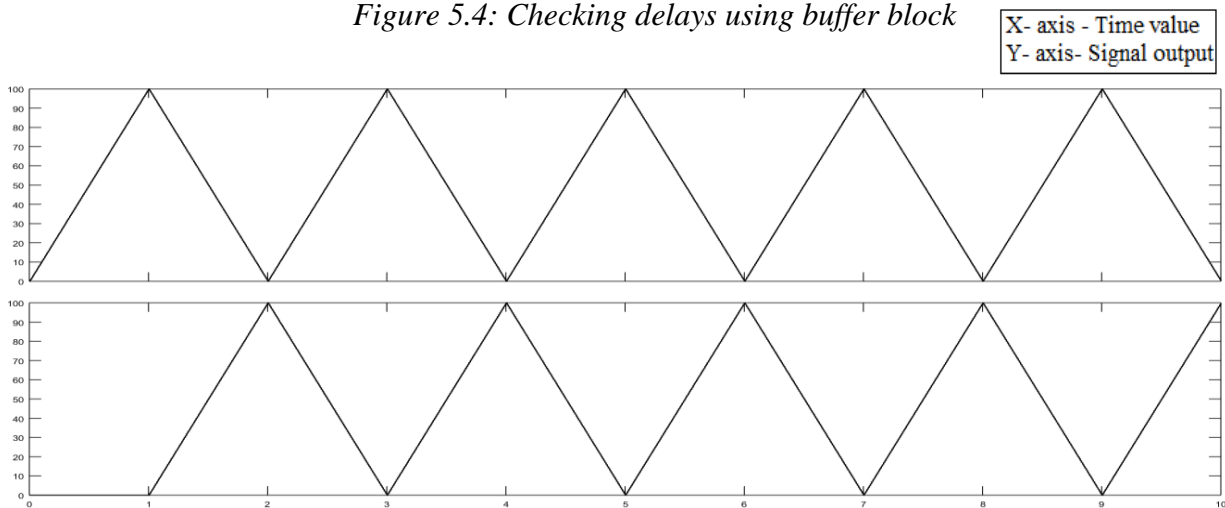


Figure 5.5: Output of buffer delay block

Now, for checking the overlaps, a ramp signal is being used. The signal is passed through the buffer block where the output buffer size is 100 and overlapping is 99. By observing the output signal, it can be easily seen that the output always shows the mean value by calculating its present value with the previous values which act as overlapping sliding window. For clearer concept, referred to Figure 5.6 and 5.7 which represent the block diagram & its output.

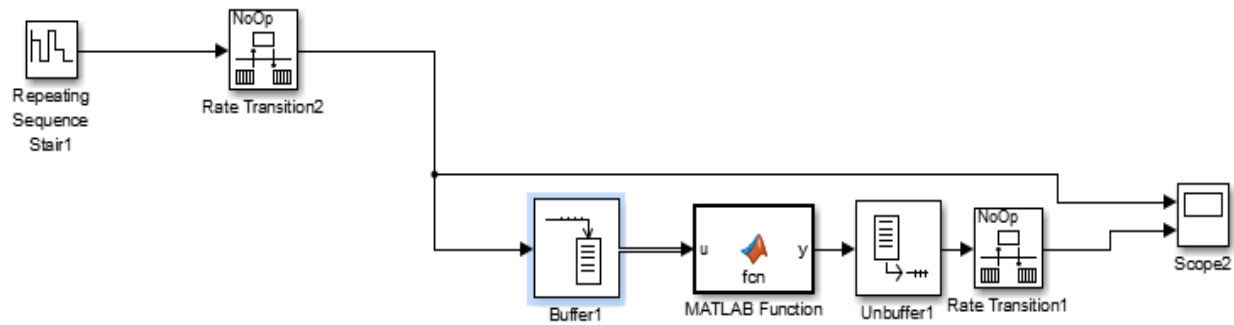


Figure 5.6: Checking overlaps using buffer block

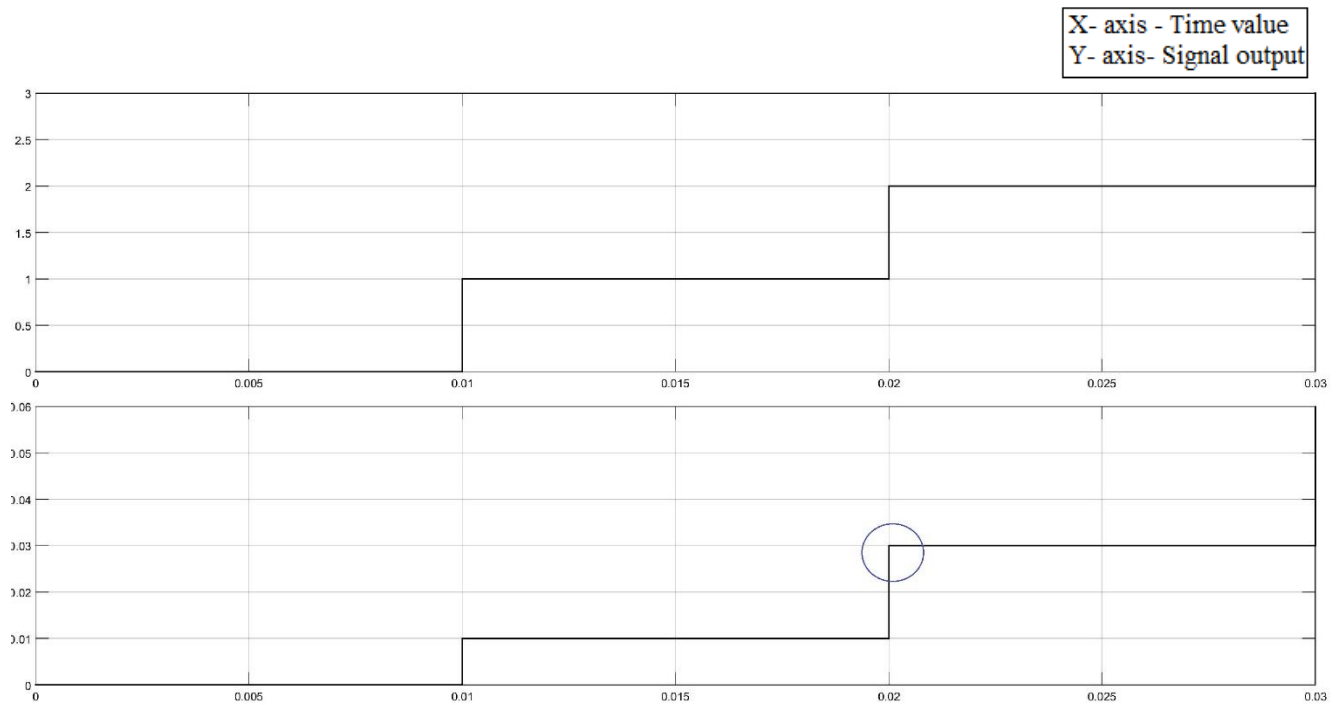


Figure 5.7: Output results of Buffer overlapping calculation

From Figure 5.7, the lower graph shows the overlapping result. For example, if the second value is observed one could notice that the output value is .03 at 2 seconds (circled as blue), which means that at first it counts 98 values as zeros for its previous values and then counts the two values for its present values which are 1 and 2 respectively.

As the Buffer block is used for creating delays and overlapping one needs an Unbuffer block before getting the outputs. After each classification, the Unbuffer block is being used. Thus, this way the analog signals can be passed through from Arduino to Simulink™ running on the PC without being interrupted or slowed down.

5.1.4 Classification

After creating buffer blocks for overlapping, the analog signals will be processed for classifications purposes. A numbers of different classification algorithms have been used in this work most of which are in the time domain. The hypothesis is that with more classifications, a larger amount of information can be extracted which is then used for training of ANNs. Each classification is designed as a sliding window with a width of 50 data points. The equations, and the Simulink™ Block diagram of the classifications are listed below.

5.1.4.1 Integrated EMG (IEMG)

Integrated EMG (IEMG) is the summation of the absolute values of the amplitude of sEMG signal. It is used as an onset index to detect the muscle activity which can be related to the sEMG signal sequence firing point [26]. IEMG can be expressed as

$$IEMG = \sum_{i=1}^N |X_i| \quad (56)$$

where N represents the length of the signal and X_i denotes the segment of sEMG signal.

5.1.4.2 Mean Absolute Value (MAV)

MAV calculates the average value of sEMG signal which is denoted as X_i . It is one of the popular classification used in myoelectric control application because of its simplicity [26]. It can be denoted as

$$MAV = \frac{1}{N} \sum_{i=1}^N |X_i| \quad (57)$$

5.1.4.3 Mean Absolute Value Slope (MAVSLP)

MAVSLP is an altered version of MAV. The contrast between them is that MAVs of adjacent segments are determined [9]. The equation for this classification is

$$MAVSLP_i = MAV_{i+1} - MAV_i \quad (58)$$

5.1.4.4 Simple Square Integral (SSI)

This classification uses the energy of the sEMG signal as a feature [26]. The equation for SSI represents as

$$SSI = \sum_{i=1}^N |X_i|^2 \quad (59)$$

5.1.4.5 Root Mean Square (RMS)

RMS uses the square root mean value signal which power raised to second. It is also called the quadratic mean. RMS is modeled as amplitude modulated Gaussian random process whose RMS is relevant to constant force and non – fatiguing contraction [9]. It associates to standard deviation, which can be expressed as:

$$RMS = \sqrt{\frac{1}{N} \sum_{i=1}^N X_i^2} \quad (60)$$

5.1.4.6 Wavelength Length (WL)

It is related to the waveform frequency, amplitude and time. WL is the increasing length of the waveform over time segment [9]. It can be defined as:

$$WL = \sum_{n=1}^{N-1} |X_{n+1} - X_n| \quad (61)$$

5.1.4.7 Zero Crossing (ZC)

ZC represents the number of times when the amplitude value of the sEMG signal crosses the zero y – axis. For EMG feature extraction, the threshold condition is used to refrain from the background noise or white noise. It provides an approximate estimation of frequency domain properties which can be formulated as:

$$ZC = \sum_{i=1}^N [\text{sgn} |x_i - x_{i+1}| \geq \text{threshold}] \quad (62)$$

where

$$\text{sgn}(x) = \begin{cases} 0.5, & \text{if } x \geq \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

5.1.4.8 Slope Sign Change (SSC)

SSC is similar to zero crossing. SSC is another way to represent the frequency information of sEMG signal. This method is an indicator of the slope sign change over three consecutive segments. This classification of sEMG signal is used for previous research [9]. SSC is described as follows:

$$SSC = \sum_{i=2}^{N-1} \{f[(X_i - X_{i-1})(X_i - X_{i+1})]\} \quad (63)$$

where,

$$f(x) = \begin{cases} 0.1, & \text{if } x \geq \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

5.1.4.9 Willison Amplitude (WAMP)

Willison amplitude detects the difference of the number of times between sEMG signal amplitude among two adjacent segment that crosses a predefined threshold to eliminate noise effects same as ZC and SSC. It is defined as:

$$WAMP = \sum_{i=1}^{N-1} f(|X_i - X_{i-1}|) \quad (64)$$

where,

$$f(x) = \begin{cases} 0.5, & \text{if } x \geq \text{threshold} \\ 0, & \text{otherwise} \end{cases}$$

5.1.4.10 Mean

It is the most common classification for measuring the central tendency of a random variable. It can denote by:

$$Mean = \frac{1}{N} \sum_{i=1}^N X_i \quad (65)$$

5.2 Configuration parameters for running the model in Simulink:

At first, the Arduino code must be uploaded. Then the Simulink™ model can be run. However, before running the Simulink™ model, some parameters need to be changed for running this model. The runtime option should be selected as normal mode, and in the options, tab target hardware should be chosen as Arduino Mega 2650. Enable overrun detection option should be checked on in the same tab. Also, the baud rate should be changed to 115,200 for Serial 0 & Serial 1 baud rate option. The reason behind changing the baud rate is because in the Arduino code the baud rate has been chosen as 115,200 for better speed.

5.3 Training Artificial Neural Network (ANN):

One channel of classified signal values are stored in the MATLAB® workspace by using Simulink™ blocks. Those stored classified values then processed by MATLAB® code for training the ANN. The corresponding Simulink™ block diagram for training the ANN is shown below.

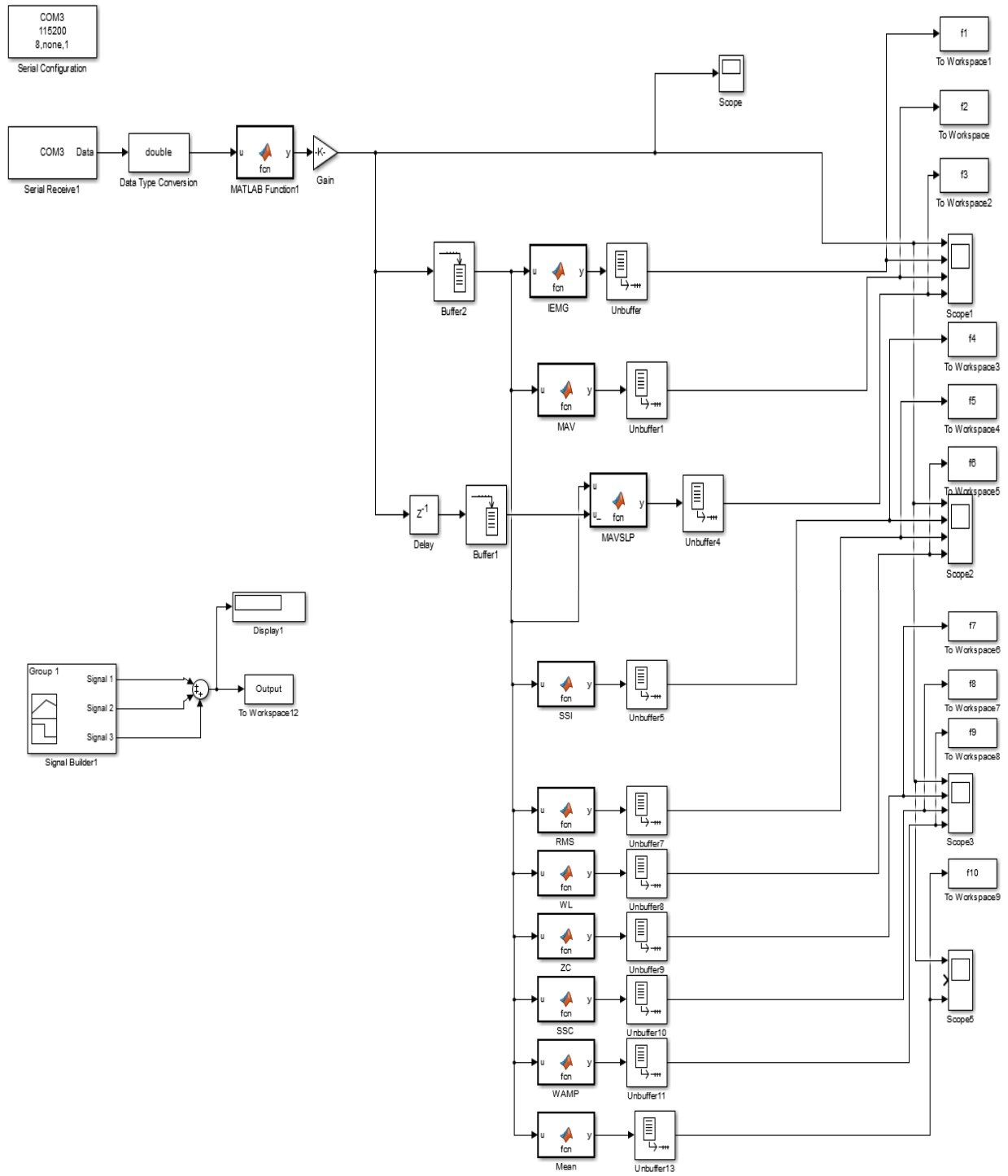


Figure 5.8: Simulink™ Diagram for storing the signal values into the MATLAB® Workspace

In this section, the classified signals from Channel 1 has been fed to the MATLAB[®] workspace. Afterwards, those classified signal values are processed for training the ANN. A different example is used, i.e. a photoresistor led sensor is tested before processing surface electromyography (sEMG) signals (see Figure 5.9).

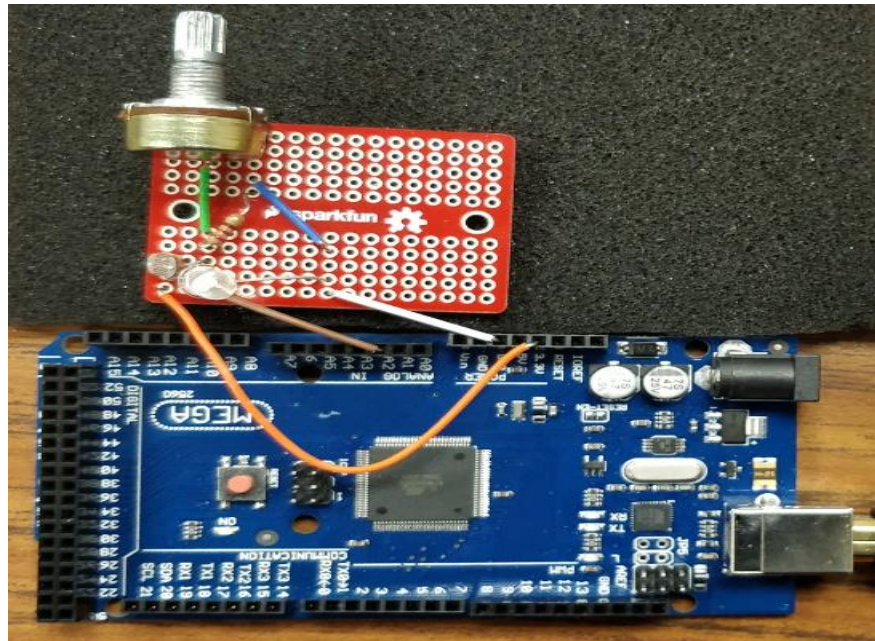


Figure 5.9: Circuit connection for a photoresistor led sensor

The photoresistor acts like a sensor. If there put some shadow it shows the lowest output values, in normal room light conditions it shows the highest value and if there is flashing an extra light it shows middle of the range value (see Figure 5.10). For experiment purpose, total thirty seconds time duration is chosen, where first ten seconds for shadow on photoresistor, next ten seconds for normal room light condition and last ten seconds selected as extra flashing light on photoresistor. For three different stages, three different target number is selected: first ten seconds is trained as number 3, next ten seconds is trained as number 1 and last ten seconds is trained as number 2 (see Figure 5.11).

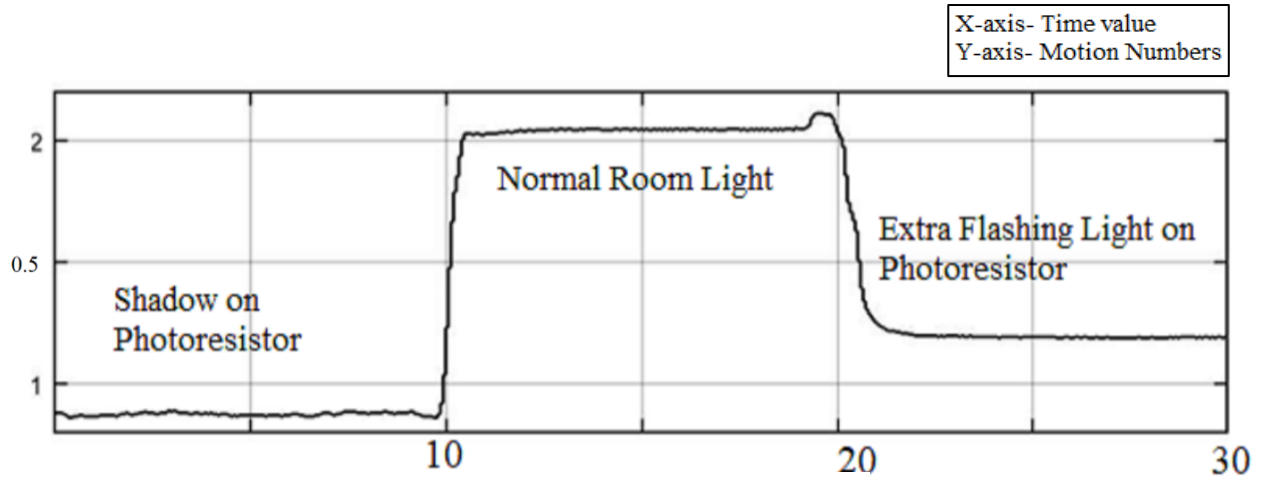


Figure 5.10: Targeted output value

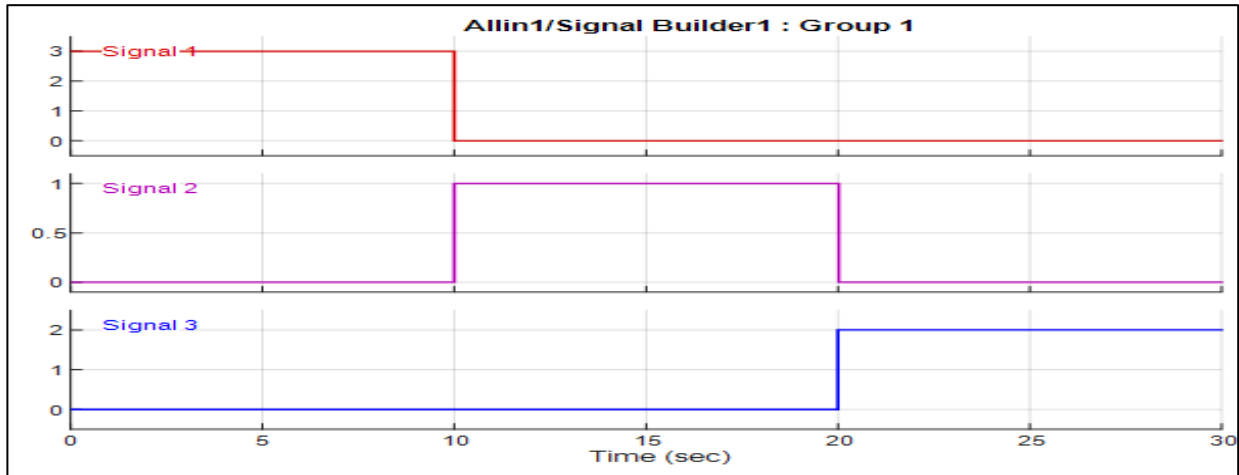


Figure 5.11: Analog output from the channel

For better training result, the value between changing the conditions are ignored. Because, there are some addition noise can be found while changing the conditions. That is why, after each nine seconds, one second values are ignored. Also, intial and last one second value is ignored for the same purpose. For making the gap, a function named stitch is built by using the MATLAB[®] code.

```
function[x] = stitch (x,t, i, j, k, l)
```

```
mask = ~(((t>=i(1)) & (t<=i(2))) | ((t>=j(1)) & (t<=j(2))) | ((t>=k(1)) & (t<=k(2))) | ((t>=l(1)) & (t<=l(2))));
```

```
x = x+1;
```

```
x = x.*mask;
```

```
x(x==0) = [];
```

```
x = x - 1;
```

Here, i , j , k , and l define the parameters of gap about how much values should be ignored. And t represents the time duration of the experiment These values are selected in the main MATLAB® as follows.

```
i=[0 1];
```

```
j=[10 11];
```

```
k=[20 21];
```

```
l=[29 30];
```

In the Simulink™ model, there is one channel which has ten classifications. These values are collected from the MATLAB® workspace. At first, all the obtained values are being squeezed to remove the singleton dimensions. Afterwards transposed the values, because for further calculations all values should be in one row. Also, the targeted output is collected from the signal builder blocks and transposed in order to make the same dimensions as inputs. Because, without the same dimensions of input values and targeted output ANN will show errors and as a result it cannot run.

Furthermore, all the processed values are trained through ANN network toolbox. Thus, for this feedforward neural network has been used. It consists a series of layers. The first layer shows the connection from the network inputs where the inputs are the classified values from the channels. And the final layer produces the network's output which follows some predetermined outputs. Inside the network each subsequent layer has a connection from the previous layer. There are forty

hidden layers, and for training function, 'trainlm' which is Levenberg Marquardt algorithm has been chosen. Total epochs are selected as 8000, and targeted performance is chosen as 1e-25.

% Training the ANN

```
net=newff(minmax(Channel1),[40,1],{'logsig','purelin','trainlm'});  
net.performFcn='msereg';  
net.performParam.ratio=0.5;  
net.trainparam.epochs=8000;  
net.trainparam.goal=1e-25;  
net.trainparam.lr=.067;  
net=train (net, Channel1, o_);  
v1=net ([x1; x2; x3; x4; x5; x6; x7; x8; x9; x10]);  
error=o-v1;
```

After training the NN, the output is plotted to compare the results between targeted output and trained output. Besides that, an error plot is also shown to observe the error difference between the two outputs. Figure 5.12 is shown the NN training toolbox to observe the number of epochs, targeted performance and gradient to achieve the training values for targeted output

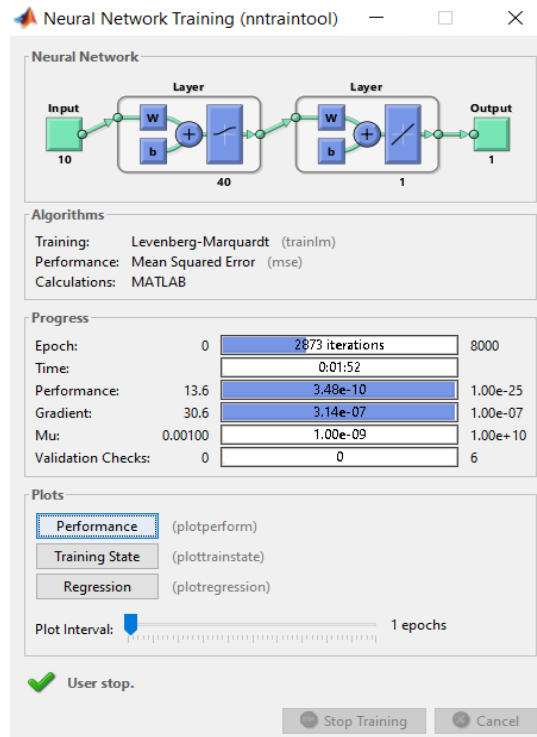


Figure 5.12: Training result for NN toolbox

And the figures for the outputs and error difference between the two outputs are shown in Figure 5.13 and 5.14.

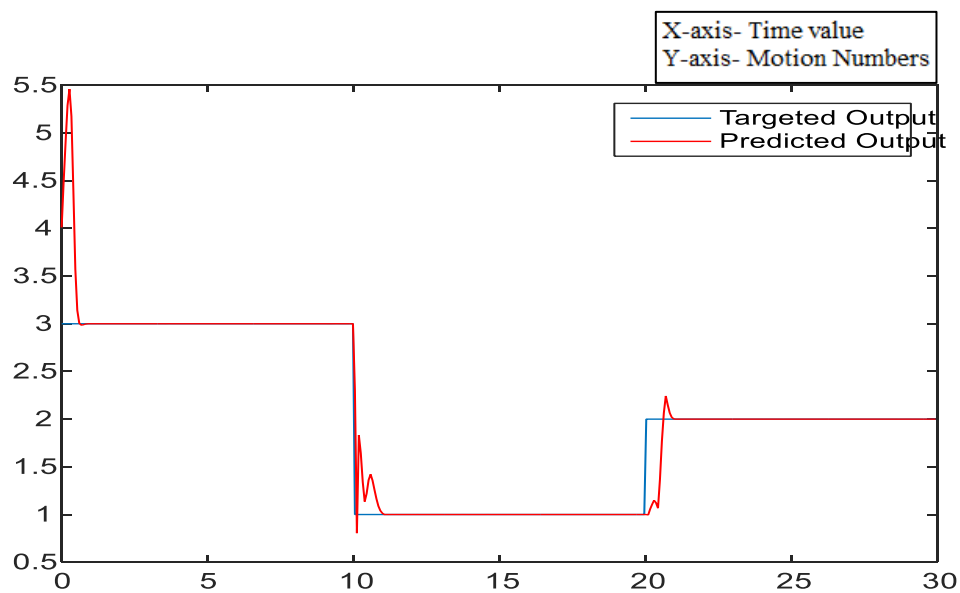


Figure 5.13: Comparing result between targeted output and predicted output

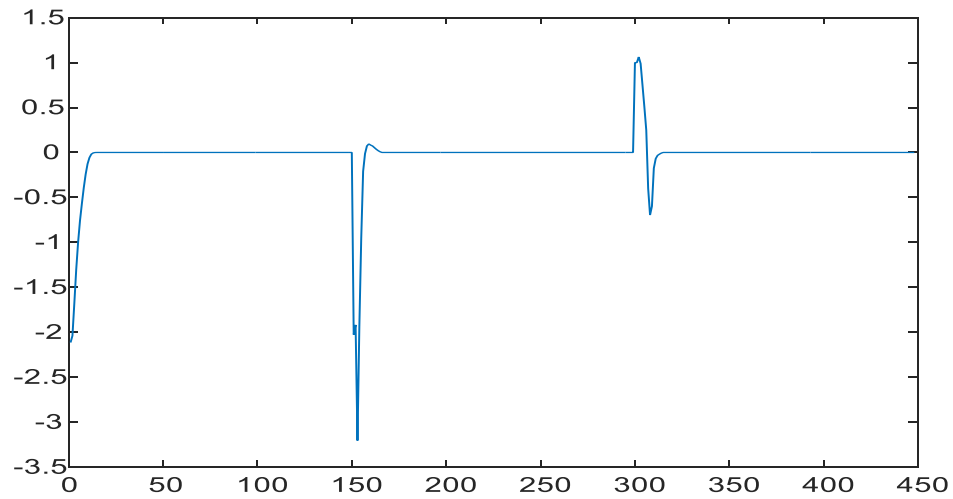


Figure 5.14: Observing the error between two outputs

Afterward, Simulink™ block created by using command `genism(net)`. The newly created trained ANN block then placed to the Simulink™ model to observe the identification of the targeted output signal. Figure 5.15 shows the Simulink™ implementation for real – time identification.

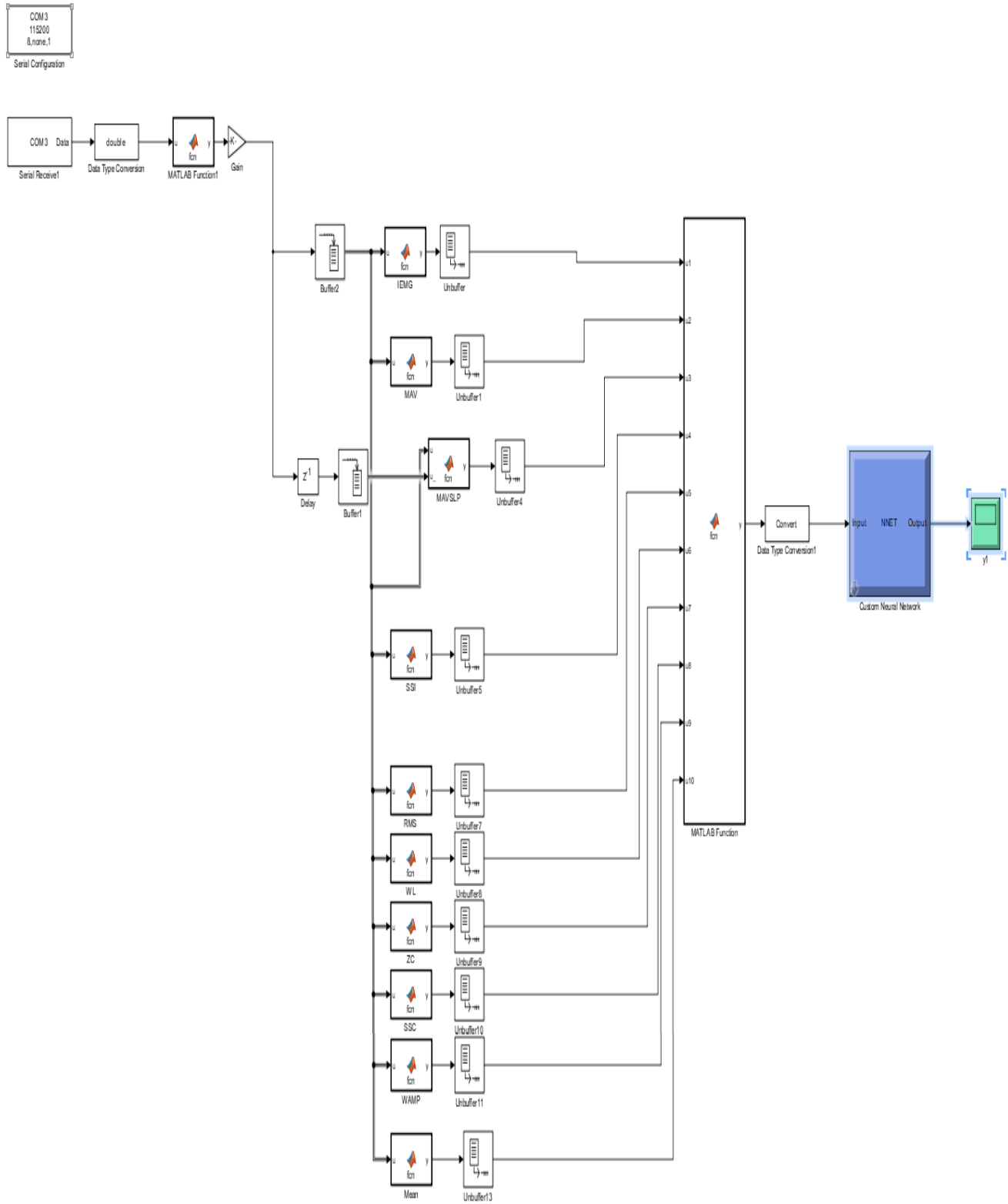


Figure 5.15: Simulink™ model for real – time implementation

5.4 Real – time identification

After running the Simulink™ in real – time, three different kinds of output can be observed. Figure 5.16 shows the output for real – time identifications. As already discussed a snitch function is used for reducing the additional noise values while changing the conditions. Though this function makes the training result much smoother than normal, however, for ignoring the gap still has some effects which is visible in the Figure 5.16. From the figure, it is noticeable that there are some extra noises while changing its real – time identification positions.

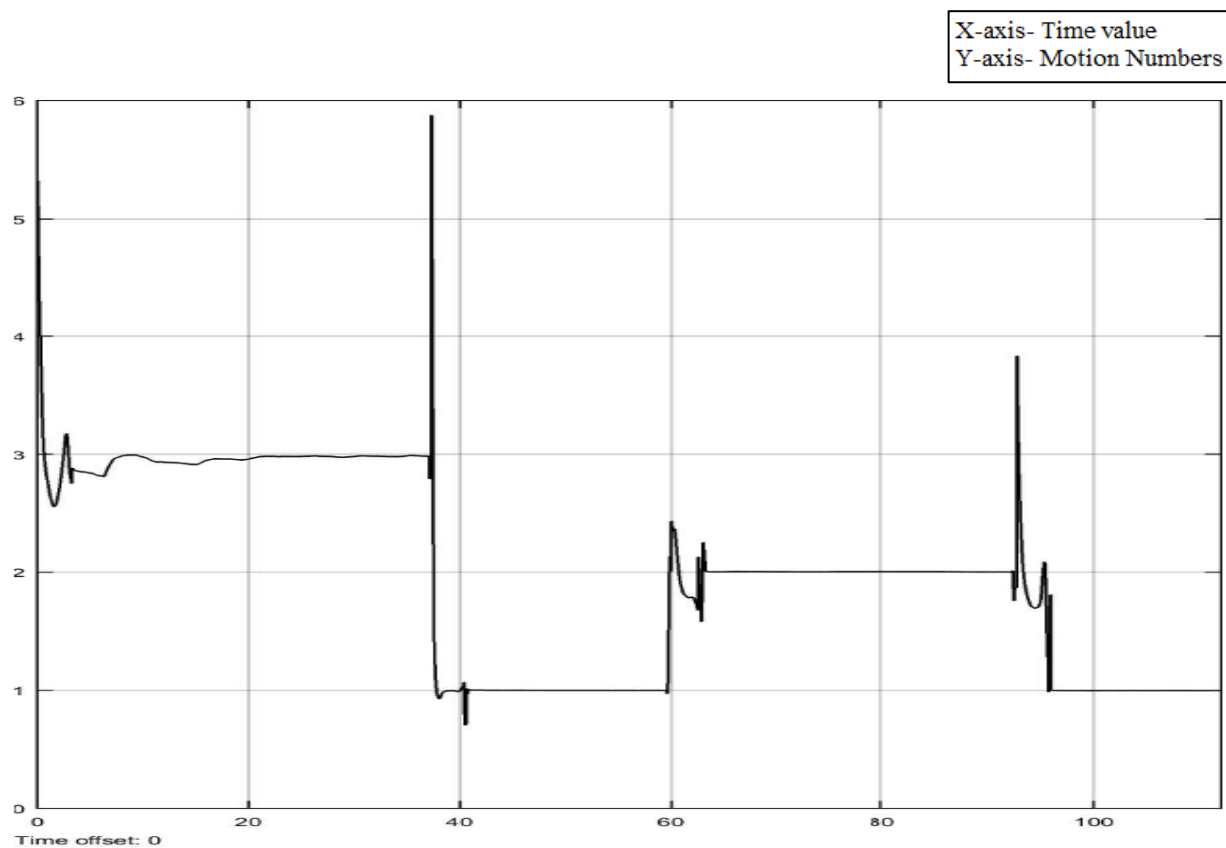


Figure 5.16: Real – time identification from trained NN Simulink™ block. Number three representing flashing light on resistors, number one for normal condition and number two for shadowing the photoresistor.

This implementation method gives the idea that the built-up Simulink™ models and all the followed steps can lead a fruitful result for real – time identification of any signal. This experiment is just a test before approaching the primary purpose of this research which is building real – time classifications and identifications for sEMG signal.

CHAPTER 6: DISCUSSION AND RESULT

This chapter consists of the discussion of the experiments using sEMG signal in real – time for training and identifications purpose. Implementation of Simulink™ models, Arduino code, and MATLAB® codes are introduced and discussed in Chapter 5. Experimenting with the sEMG signals for training the Artificial Neural Network (ANN) gives better training output, however, for real – time identification the output does not show the satisfying result as obtained with the experiments using the LED sensor lights. Therefore, some different approaches were followed for improving the identification results. As we know, sEMG is a complex signal which always gives random values based the subject's different movement. As a human being our movement are not exactly same as the previous movement even if we are doing the same activities in a repeated way. Subject's different muscle movement and activities can imply a lot affect the quality of the outputs while experimenting in real – time identification. Because in this research it is found that if the sEMG is trained for a specified period for different motions, the ANN can train unnecessary motion movement. For example, if motion one is trained for three seconds then directly move to action two and train for another four seconds without placing any gap, this will lead the ANN to train the random values. As a result, real – time identification will give some random noise signals which are not related to the identifier of the motion executed. That is why instead of using some continuous training pattern, random output numbers are chosen with random period gaps. This training pattern can help ANN to separate the additional movement values apart from the needed movement values.

6.1 Building real – time random output values

For this purpose, at first, a physical pushbutton is used to correlated motion initiation with the collected sEMG data. However, we know that pushbutton has some debouncing effect. Even if the debouncing impact is solved, the implementation of a pushbutton in Simulink™ faces some problems. First, it is not possible to use an Arduino analog input block because of the use of Arduino serial communication through USB for faster processing speed. Thus, the Arduino code needs to change in order to send the pushbutton values along with sEMG signal values to the PC. Nonetheless, the pushbutton values also affect the sEMG values as these values pass through the serial communication (Figure 6.1). From the figure, it is observable that when the pushbutton is pressed a sudden peak can also be seen in real – time sEMG signal which can eradicate the natural characteristic of the sEMG signal. Therefore, a different approach is being pursued for building the real – time implementation for sEMG experiments.

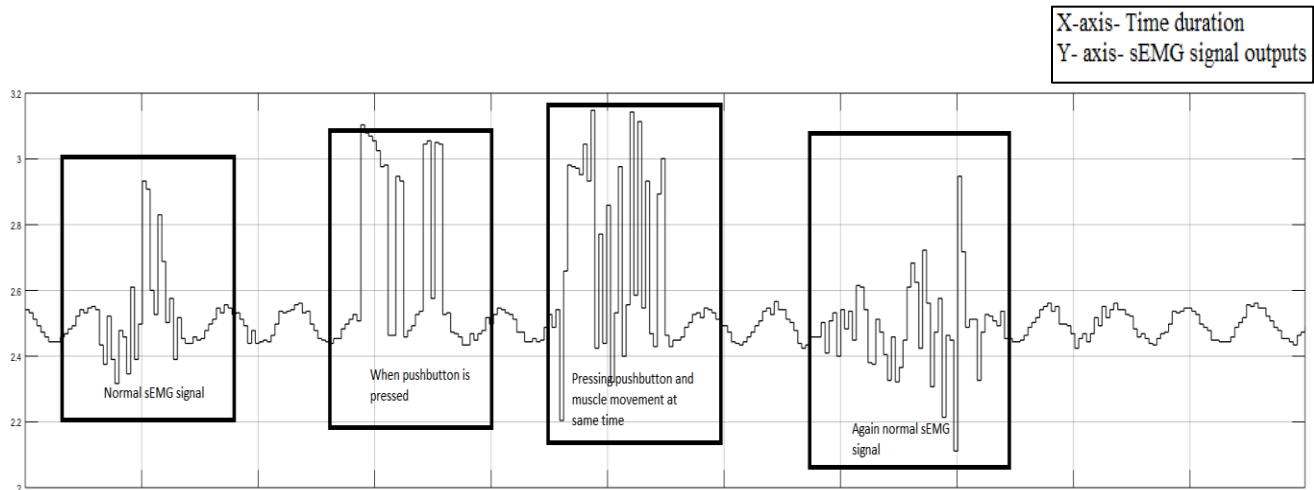


Figure 6.1: Pushbutton values affecting the natural characteristic of sEMG signal

In this work, a built – in Simulink™ block is used for giving a different number of outputs at random times [30]. Inside the Simulink™ block, a switch block and call – back functions are used to create an artificial pushbutton. The pushbutton block can have many different output numbers. However, for this research purpose, only two different numerical values are chosen. The first value is ‘three’ which is associated with the inner forearm muscle movement and the second value is the number ‘one’ which is associated with the outer forearm muscle movement values. Figure 6.2 shows the Simulink™ block for two different output numbers one and three. By double – clicking the block, the output values can change from 0 to 3 or 0 to 1 in real – time.

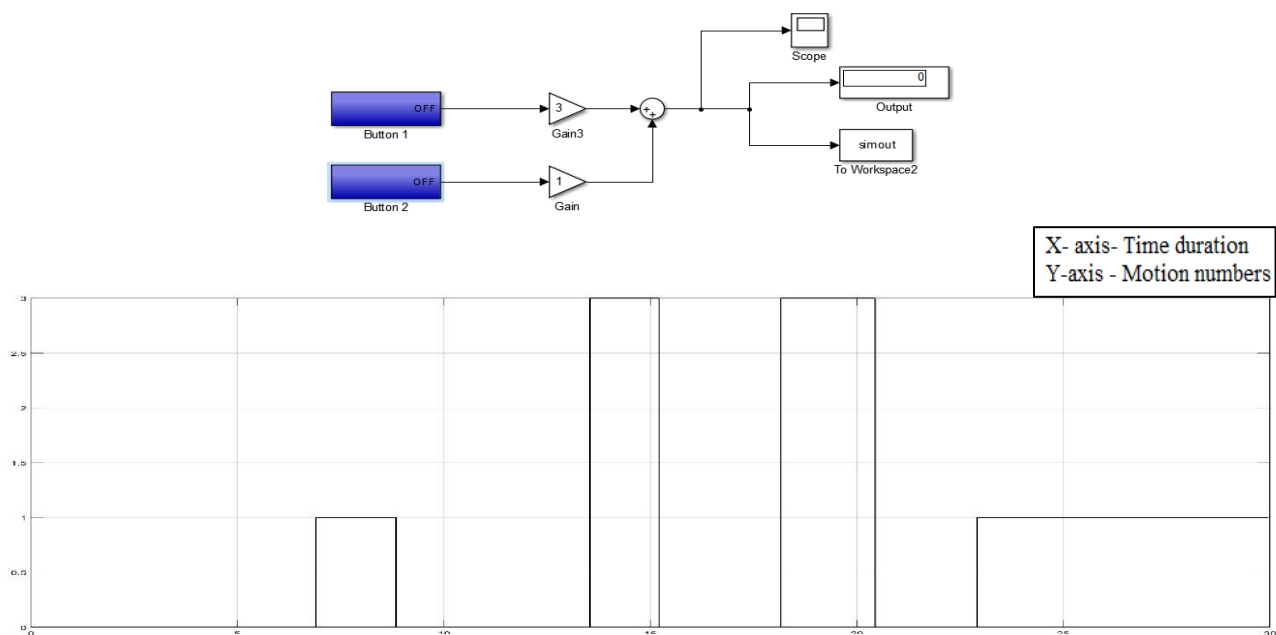


Figure 6.2: On/Off Simulink buttons and their corresponding output values

6.2 Classifications

Classification is implemented according to the material introduced in Chapter 5. In Chapter 5 there are ten classification algorithms introduced. These classification algorithms work well with the LED sensor experiment. However, the same classification algorithms give noisy output while

experimenting with sEMG signals. Later, it is found that some of the classifications create noisy output instead of improving the identification outputs. For this purpose, a different approach is attempted for better identification. The actual sEMG signal is too small (in millivolts) which can be observed by amplifying the signal. It is noticed that some of the classifications do not show a change in values for the smaller amount of signal change. On the other hand, some of them can give a wide variety of output values for a small change in the signal amplitude. At first, the Simulink™ blocks with classifications running in real – time for the inner forearm motion and outer forearm motion are used. Afterwards, these results are compared for the different classifications with each other to observe which classification can give a significant amount of output values for the small amount of change in the sEMG signals. Figure 6.3 and 6.4 show different classifications values for real – time forearm movements.

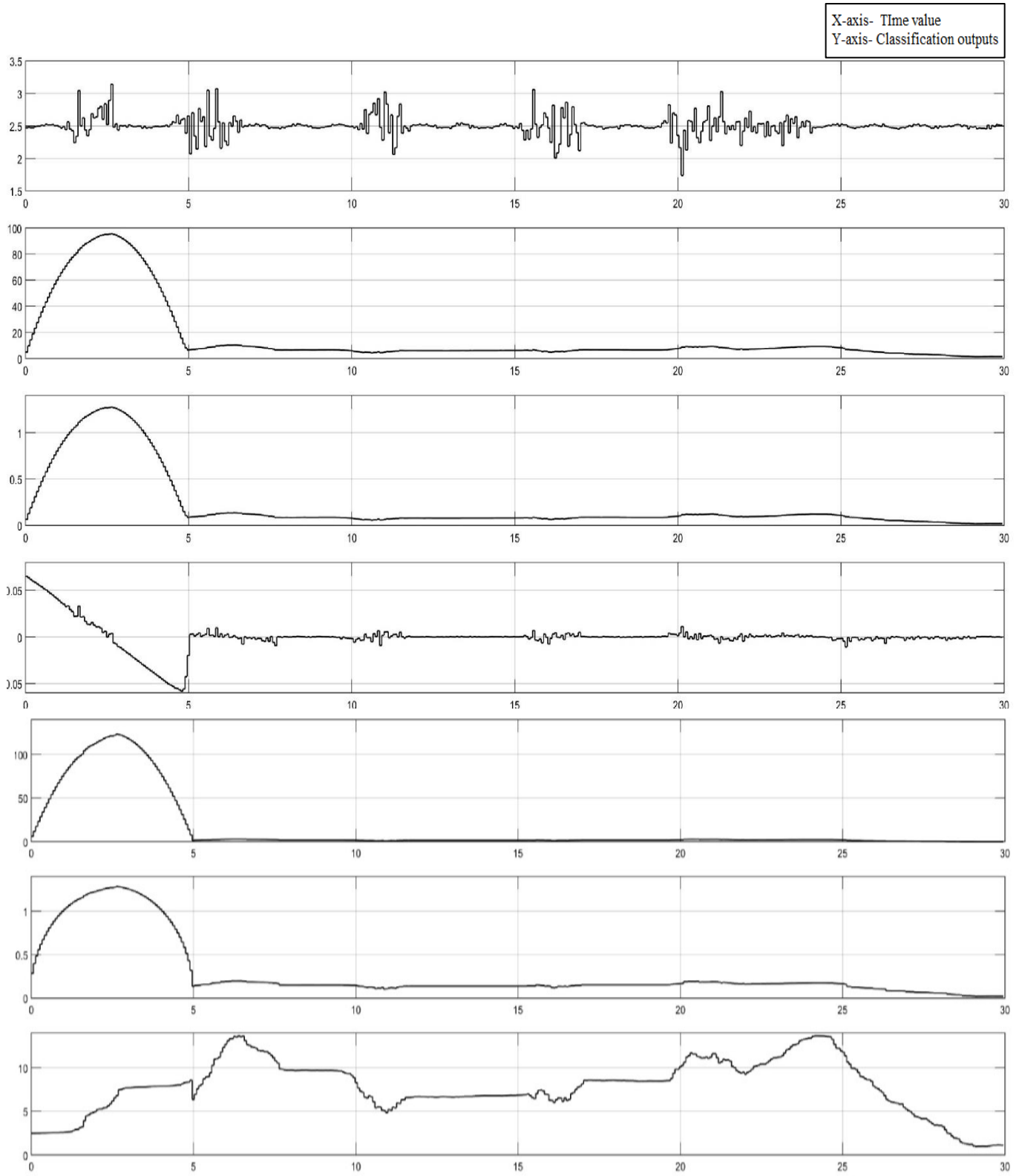


Figure 6.3: The First graph is sEMG signal, second IEMG, third MAV, fourth MAVSLP, fifth SSI, sixth RMS, seventh WL classifications outputs

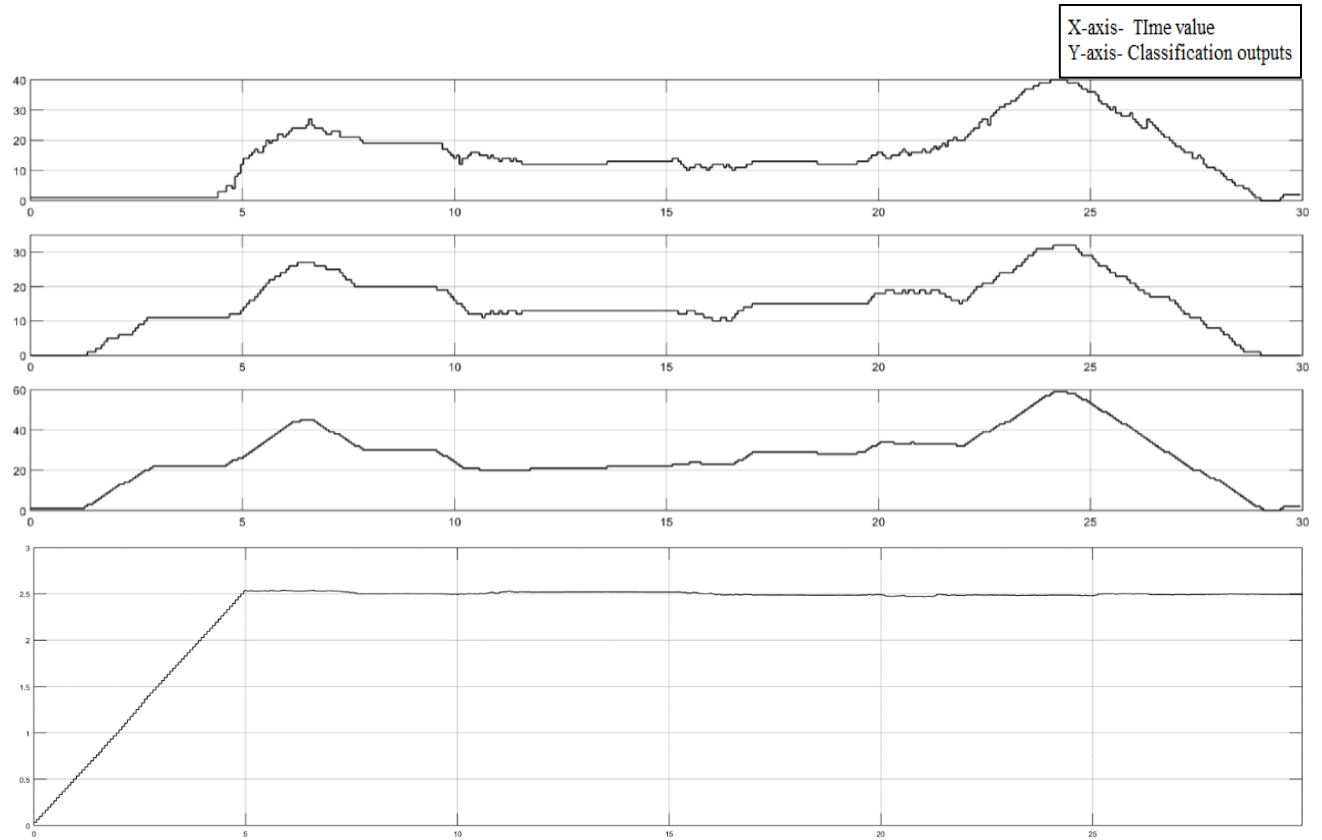


Figure 6.4: The First graph is SSI, second RMS, third WL, fourth ZC, fifth SSC, sixth WAMP and the last one is mean classifications outputs.

In Figure 6.3 the first graph shows the output of a real – time sEMG signal. The following graphs are showing the different classified value perspective of the sEMG signal. By observing the ten different classifications, it can be noticed that the classifications named waveform length (WL), zero crossing (ZC), slope sign change (SSC) and Wilson amplitude (WAMP) classification are showing significant output changes due to a slight change in the real – time sEMG signal. Thus, these four classifications are chosen and used to train the Artificial Neural Network for better identification results.

6.3 Training Artificial Neural Network(ANN)

In Chapter 5, the training is done by collecting all the classified values from the Simulink™ to MATLAB® workspace. However, instead of using all ten classifications together a few classifications are chosen based on their performance for better accuracy identification. These chosen classifications are used one by one instead of using them together, because it is also found that when two or more classifications together are used, they hamper the output values for better sEMG identifications. Thus, WL, ZC, SSC, WAMP classifications are used one by one for training purpose and later their trained value is compared to check which classification is giving better training and identification results. Figure 6.5 denotes the Simulink™ block for four different classifications. These classification values are stored in the workspace using different names so that each classification can be trained just by changing their stored variable names.

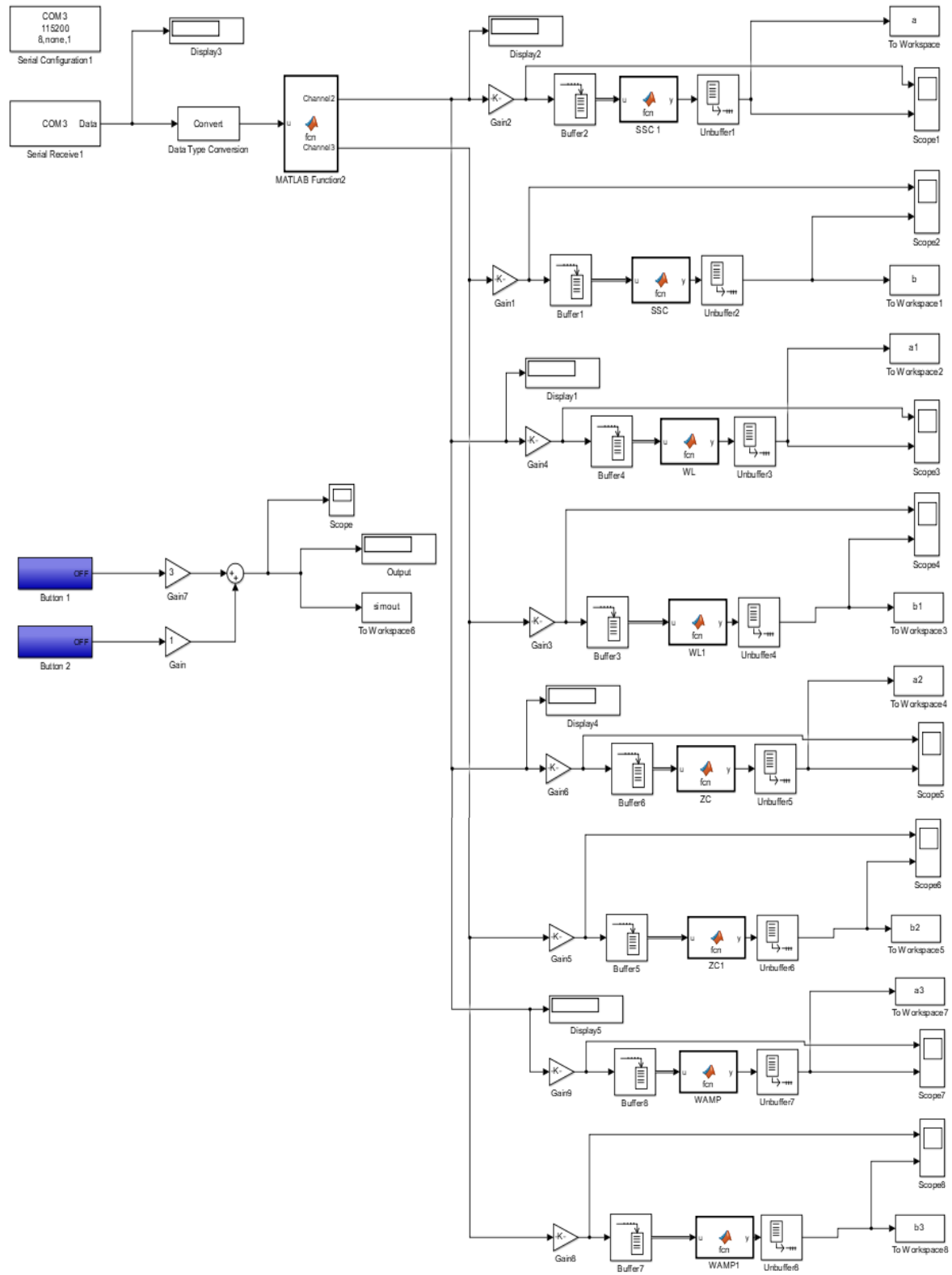


Figure 6.5: Four different classifications Simulink™ block

For training purpose, two movements are chosen – inner forearm movement and outer forearm movement which is shown in chapter 4 (See Figure 4.13 and 4.14). For training the targeted output values with the input values, the inner movement is set to number 3 and outer forearm movement is set to number 1. Figure 6.6, 6.7, and 6.8 show the inner and outer muscle movement

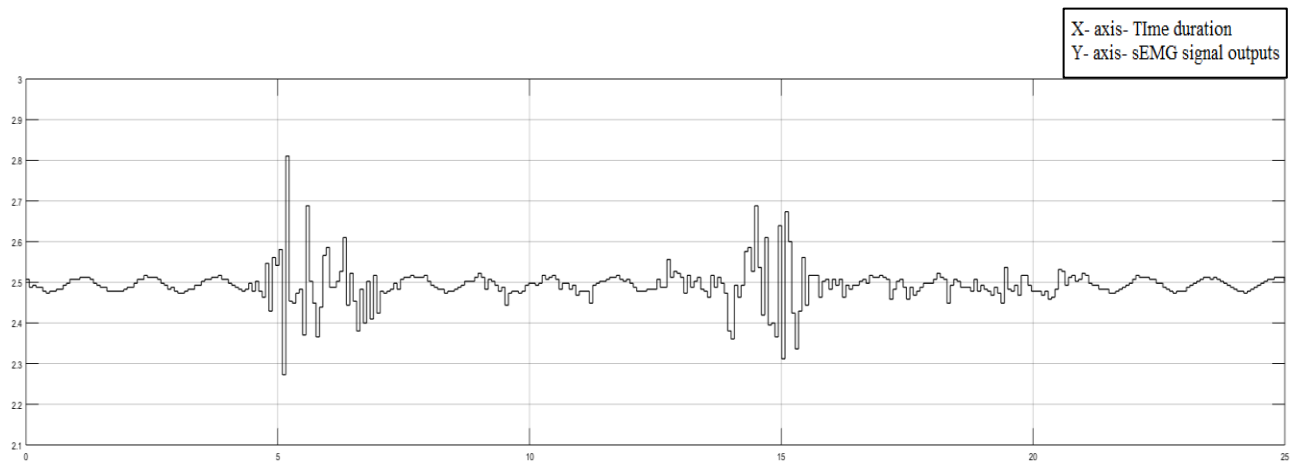


Figure 6.6:sEMG signal for inner forearm muscle movement

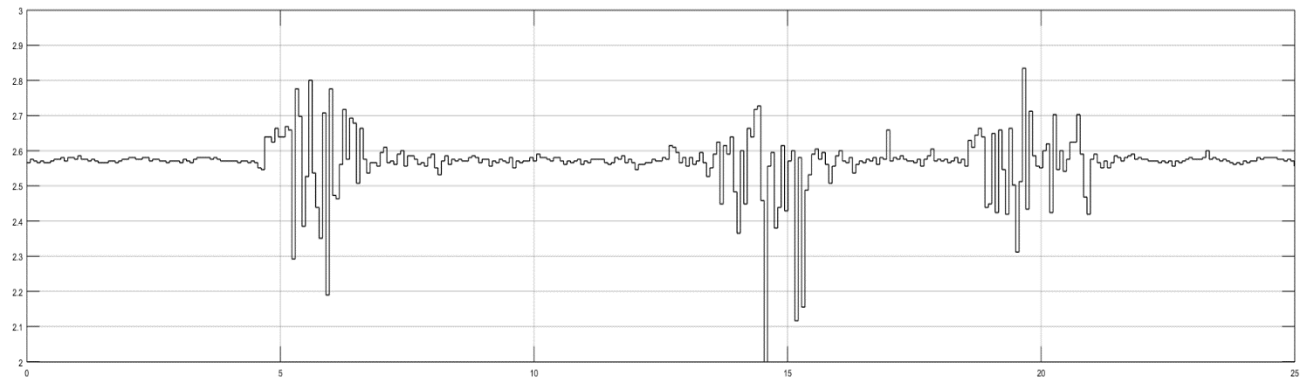


Figure 6.7:sEMG signal for outer forearm muscle movement

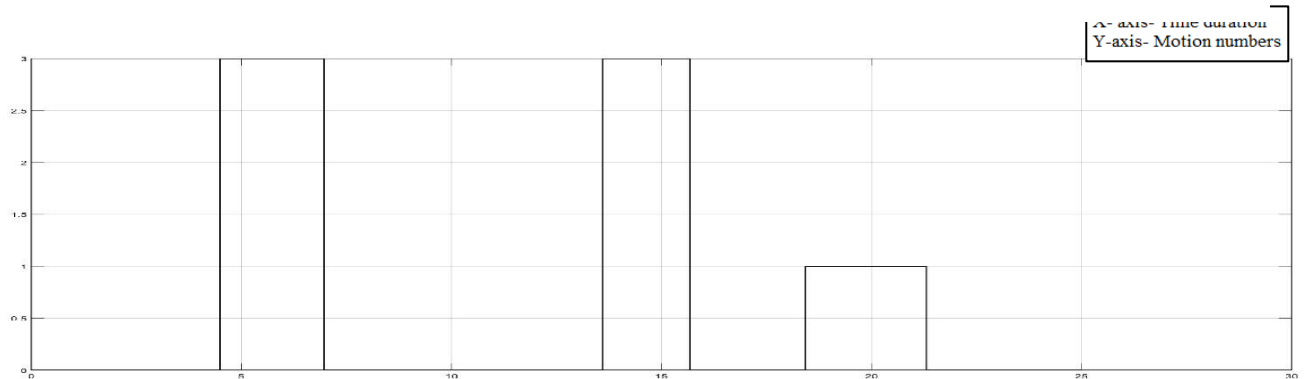


Figure 6.8:Real – time pushbutton output considered as a targeted output

corresponding to their target values 3 and 1. That means, when the target output shows a 3, the participant needs to do an inner forearm movement. On the other hand, when it is 1 the participant needs to perform outer forearm movement.

At first, the sampling time is chosen as 0.067 seconds which is also used in the Simulink™ block. The classified real – time values from two different channels are stored in the MATLAB® workspace to train the values through ANN. Later, time values are chosen from one of the channels which value is stored in the MATLAB® workspace to plot the output result between input values and target outputs. Afterwards, all the values are transposed into a row matrix to confirm all the dimensions to the same range. Because of the backpropagation training, the values of the input data and target output data should have the same dimensions.

There are many kinds of the backpropagation algorithm. Among them, for faster training purpose, the Levenberg – Marquardt (trainlm) is used [30]. There are several parameters associated with trainlm. A feed – forward network consisting of two – layer backpropagation is used. The following call to the MATLAB® function ‘newff’ creates a two – layer network with 40 neurons in the hidden layer:

```
net = newff(p,outputb,40,{'tansig','purelin'},'trainlm');
```

Here, p represents the input values which are the classification values from two channels, and $outputb$ represents the target output values. Tan – sigmoid and linear transfer functions are chosen as transfer functions. For better training results, the default training parameters are modified. The following part of the code shows the regularized performance functions. Here the performance ratio is set to .007. The learning rate is chosen to be 0.05; the number of epochs is 5000 and the parameter goal is chosen as 1e-15. It should be noted that if the learning rate is made too big, the algorithm becomes unstable [19]. On the other hand, if the learning rate is too small, the algorithm

takes a long time to converge [19]. Thus, it should be chosen wisely. The same considerations need to be applied for choosing the performance parameter ratio. If this parameter is too large, the results might get overfitting. Otherwise, if the ration is too small, the network does not sufficiently fit the training data.

```
net.divideFcn = '';
net.performFcn='msereg';
net.performParam.ratio=0.007;
net.trainParam.show = 500;
net.trainParam.lr = 0.05;
net.trainParam.epochs = 5000;
net.trainParam.goal = 1e-15;
```

The initialize command is used to reinitialize the weights and biases. This function works a network object as input and rebound a network object with all weights and biases initialized. Finally, the training has to be done with the combination of input values and targeted output values. Below is shown the coding lines for initializing the network and training command.

```
net = init(net);
[net,tr] = train(net,p,outputb);
y = sim(net,p);
```

Also, the error is calculated from the trained values and targeted outputs. Figure 6.9 shows the training results for four different classifications individually such as - WL, ZC, SSC, WAMP. The time duration for real – time classification is set to 25 seconds. Overlapping delay is chosen as 100 data points. Thus, in the buffer block the output buffer size is set to 100 and buffer overlap is set

to 99. Figure 6.10 represents the trained output values comparing the targeted output and predicted outputs. Moreover, Figure 6.11 and 6.12 produces the error values between targeted and predicted outputs.

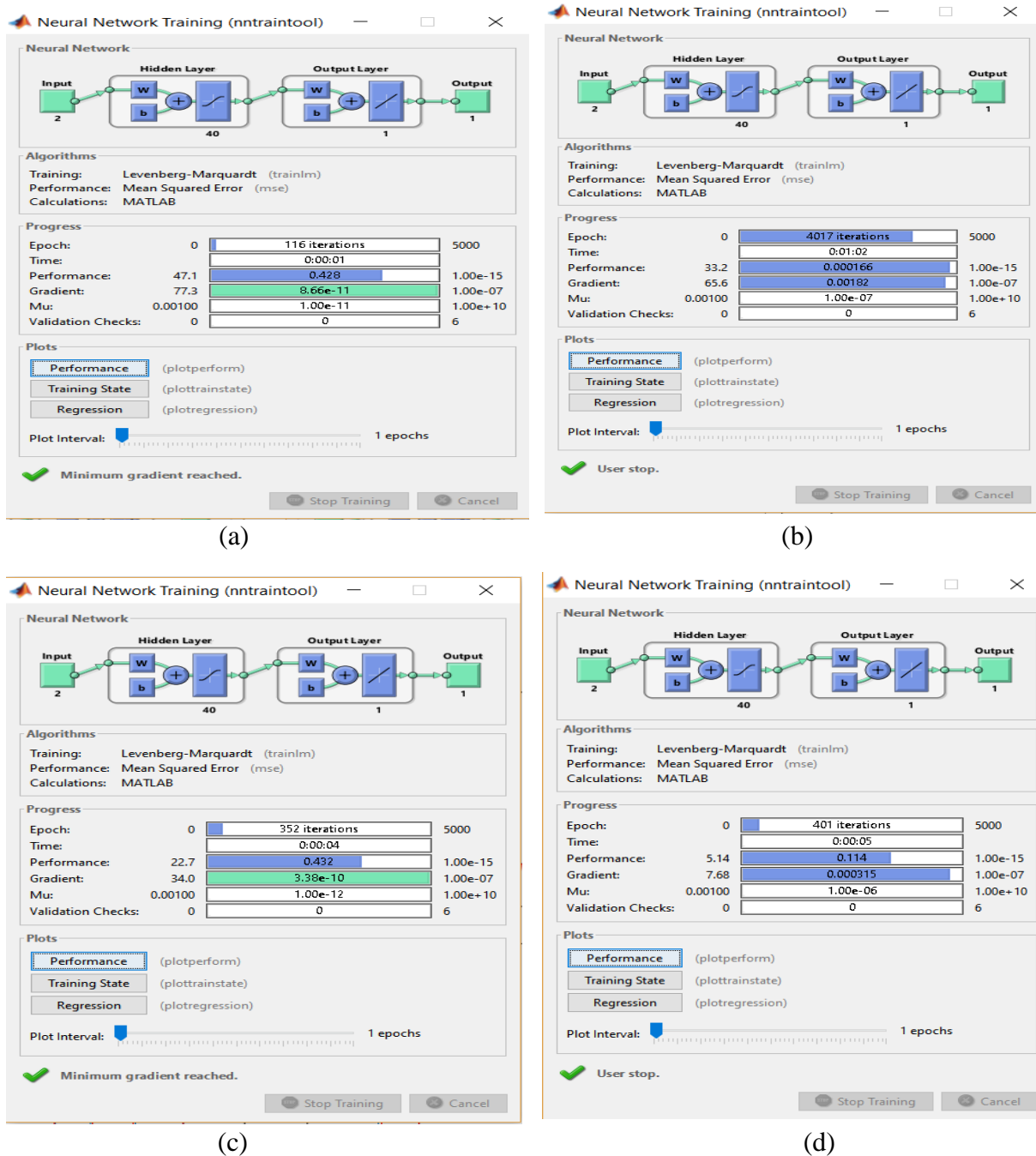


Figure 6.9: Training results for NN toolbox. (a) Slope Sign Change (SSC) (b) Waveform Length (WL) (c) Zero Crossing (ZC) (d) Wilson Amplitude (WAMP).

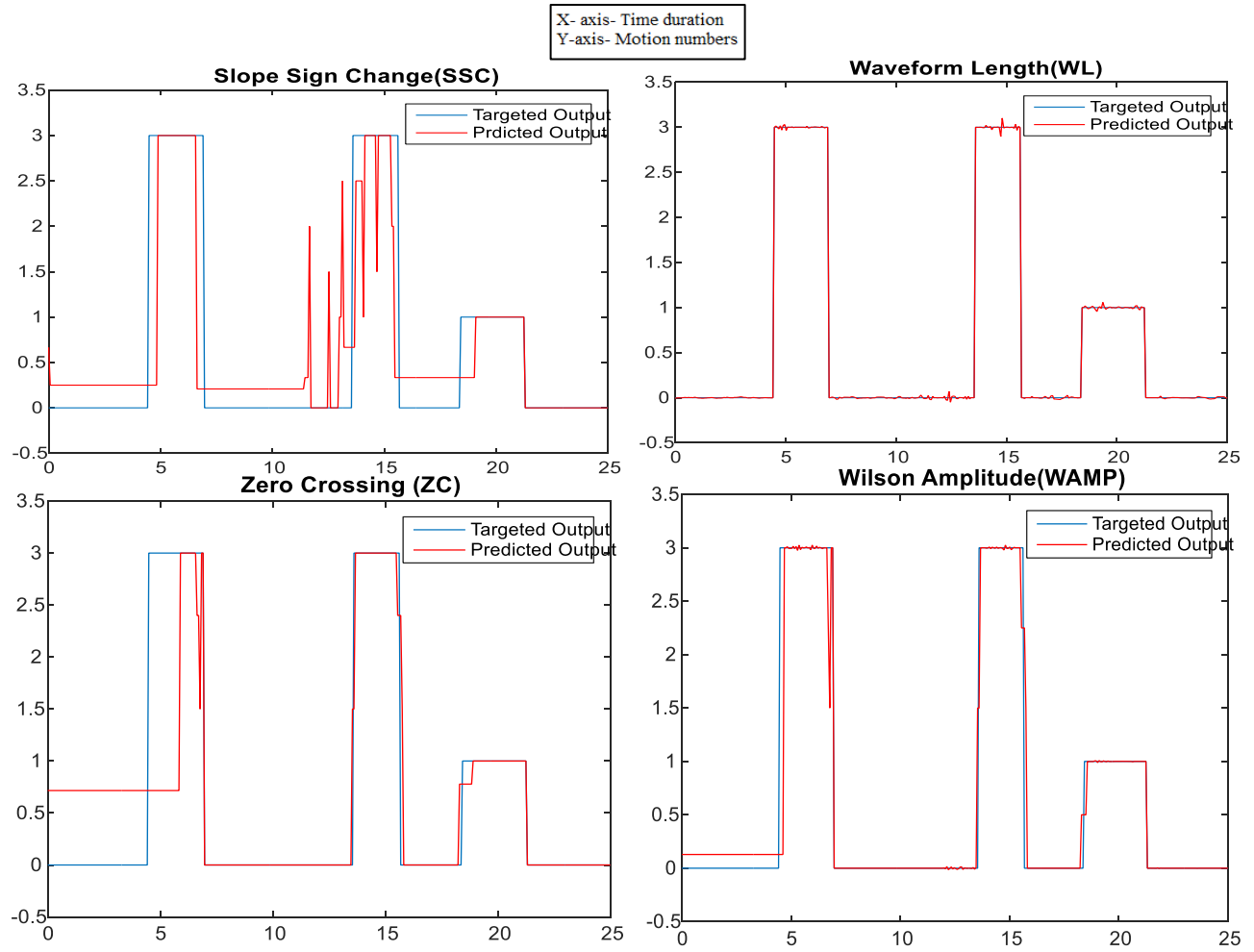


Figure 6.10: Four different classification SSC, WL, ZC and WAMP trained output graphs

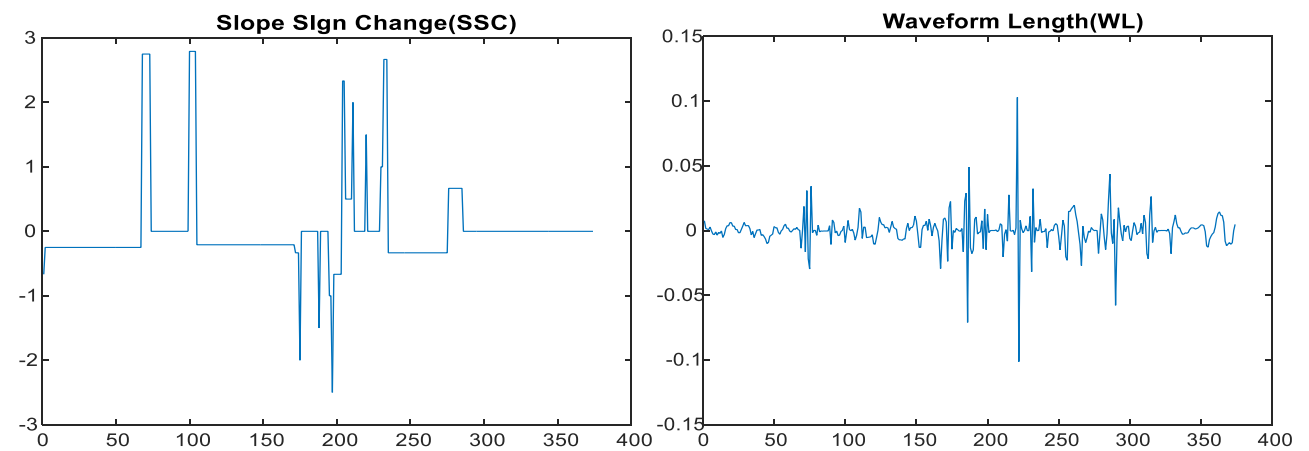


Figure 6.11: Error difference between targeted and predicted outputs for SSC and WL

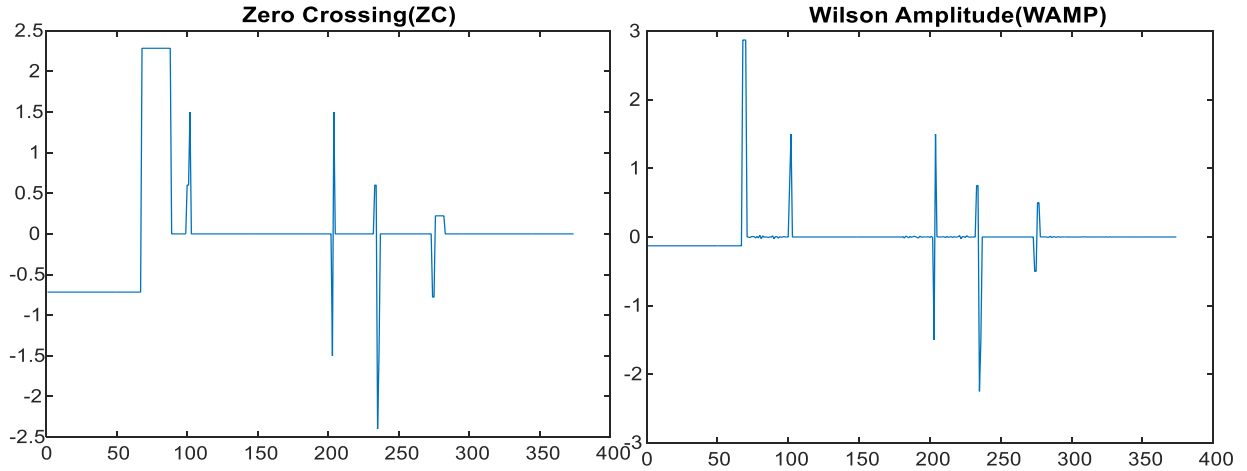


Figure 6.12: Error difference between targeted and predicted outputs for ZC and WAMP

Among the four classifications, waveform length (WL) shows the lowest error value which is 2.561%. Others such as the SSC, ZC, and WAMP error values are 115.58%, 109.84%, 29.74 % respectively. However, it should be noted that the error values for different classifications might vary depending upon how fast participants can do inner and outer forearm movements according to the real – time random pushbutton outputs. This pushbutton output is considered as a targeted output. If there is a delay between the pushbutton outputs and the participant's movements, it might create a large amount of difference in real – time identification. Thus, the timing between movements and pushbutton should be done cautiously.

6.4 Identification and result

Four different Simulink™ blocks are created from the trained ANN for four distinct classification – WL, ZC, SSC, WAMP. Afterwards, those blocks are placed in the Simulink™ model individually for observing real – time identifications. However, there is a change in the Simulink™ model for better results. A MATLAB® function block is used toward the trained ANN Simulink™ block for collecting the signal values from two different channels classifications. Besides, a saturation block is also used to fix the range, thus the identified output values cannot go beyond the predefined range. While training the network number 1 is chosen for outer forearm movement, and number 3 is chosen for inner forearm movement. For this reason, the lower limit range is set to 0 and upper limit range is set to 3, so the real – time identification values cannot go beyond this range. Figure 6.13 shows the Simulink™ block implementation for four different classifications. Also, Figure 6.14, 6.15, 6.16, 6.17 denote the real – time identifications for different classifications.

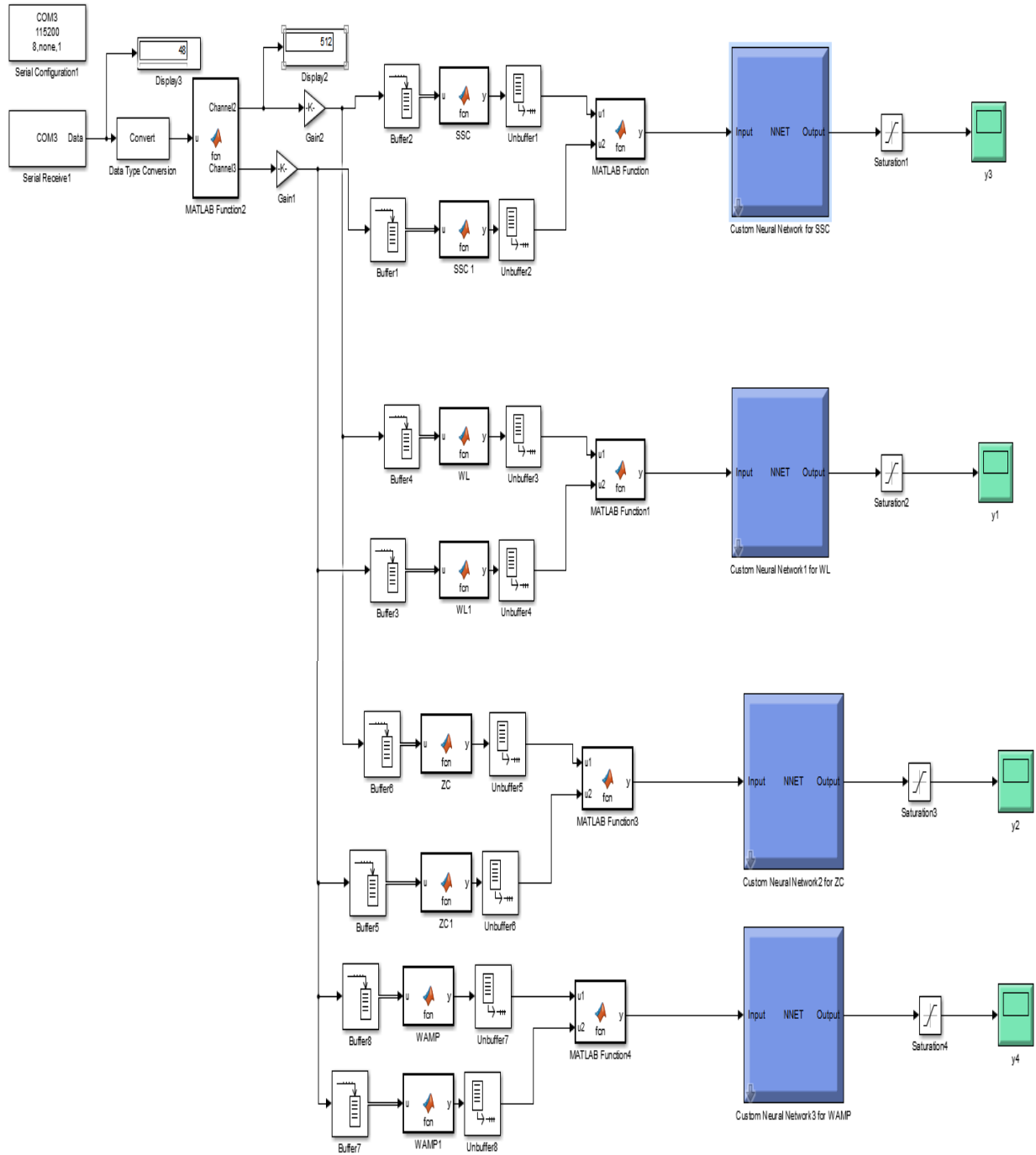


Figure 6.13: Trained ANN Simulink™ block implementation for real – time identification

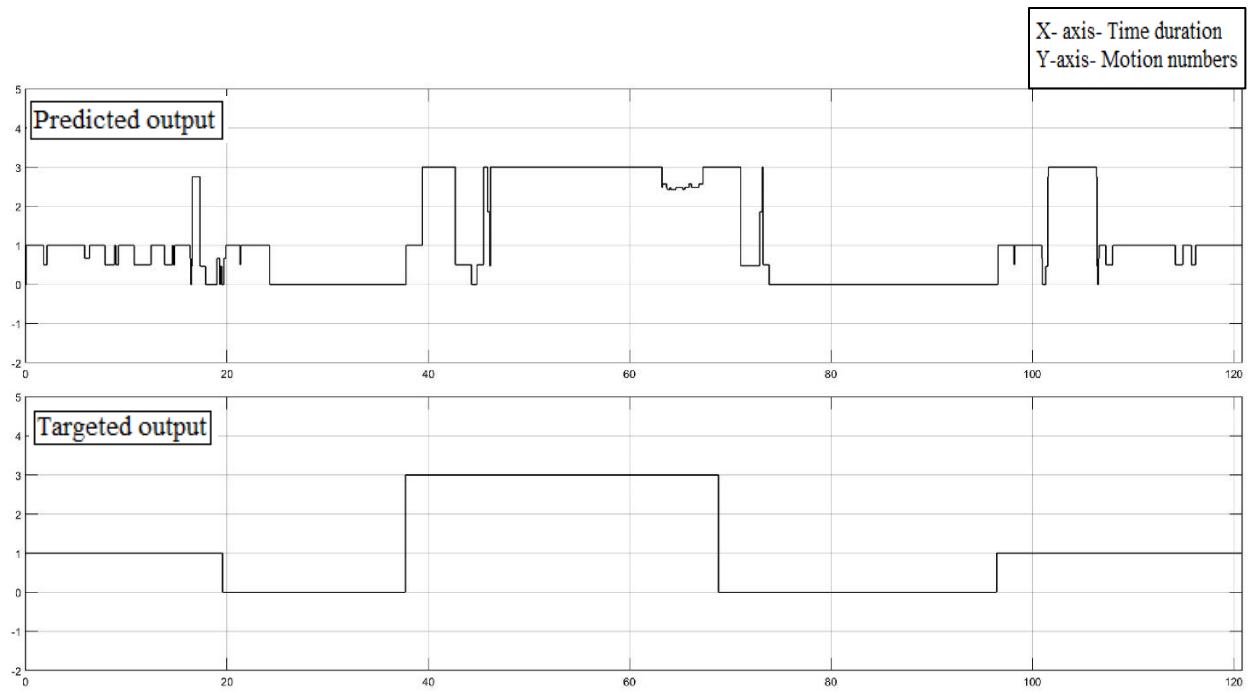


Figure 6.14: Identification using SSC classification

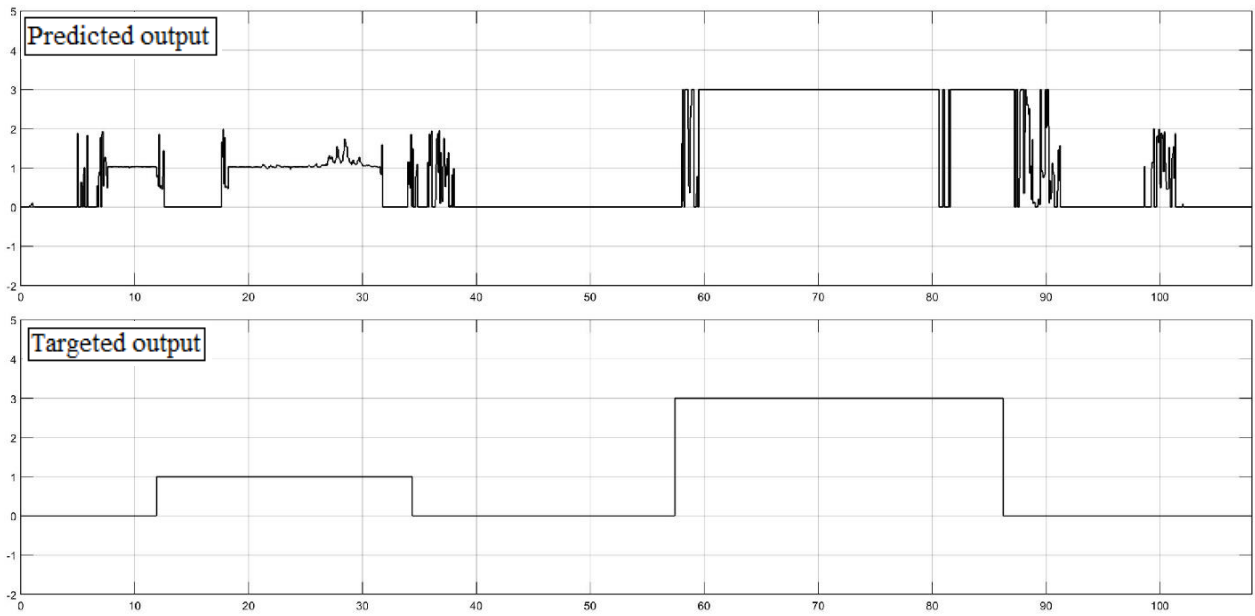


Figure 6.15: Identification using WL classification

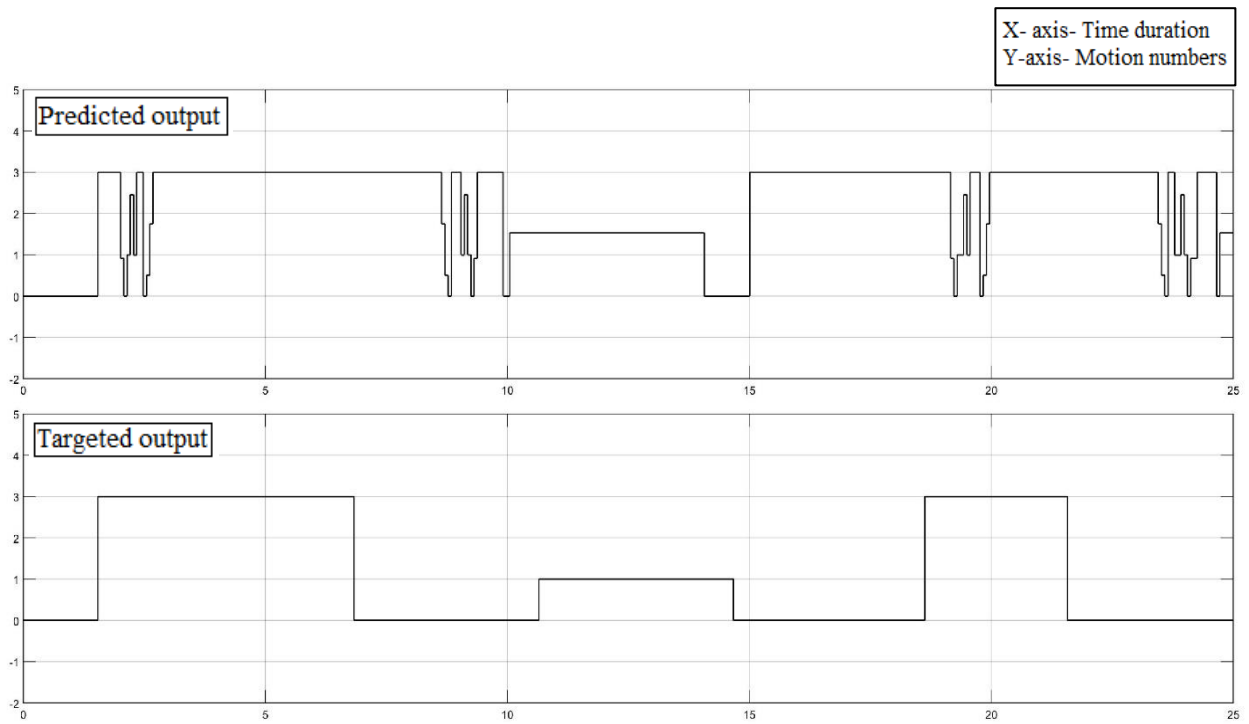


Figure 6.16: Identification using WAMP classification

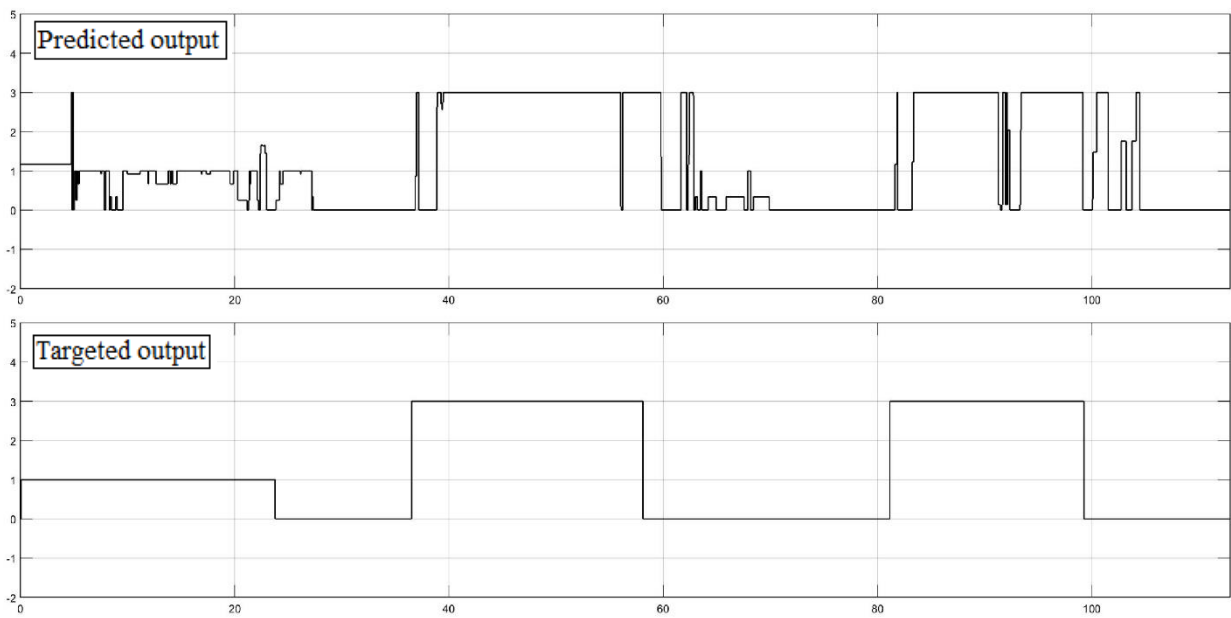


Figure 6.17: Identification using ZC classification

By noticing the identifications results from four different classifications, WAMP shows the best identifications result. Though the WL shows less error values (2.561%), however, it seems like the training values might be overfitting which creates some unstable output on real – time identification. Thus, it should be remind that overfitting values might also cause some noisy outputs. Nevertheless, it has been mentioned that the results of the different identifications may vary based on the participant performance. In this case, the delay between the participant reaction and the targeted output values play a vital role. There are some noises in the identification output which represent the error values from the trained ANN. The better the training the fewer noise content in the output scope. Thus, from all the steps derived in this chapter someone can conclude that the presented approach works for in real – time sEMG signal identifications. However, it should be pointed out that classification methods play an essential role in achieving the most accurate identification results.

CHAPTER 7: CONCLUSION AND FUTURE WORKS

7.1 Conclusion

The study of human apprehension through artificial intelligence recently has seen a rapid improvement because of soft computing abilities and the implementation of many concepts including Artificial Neural Network (ANN). Nowadays, the study of sEMG signals and their relation to human motion learning has become one of the hot topics of research. An abundant number of researchers study the sEMG signal relation to human motion through the application of ANN. Most of the studies include the use of different features/classifications to help the learning process of the ANN.

The outcome of this research shows some significance. This research explores the relationship between sEMG and the human forearm movement motions. However, before proceeding with the sEMG signal classification, a different example experiment is done in chapter 3 which includes the training and identification of signals in real – time. This experiment helps to justify the proposed procedure in order to give the desired result for this research. Nonetheless, there have been some slight changes between the experiment for the LED sensor light and the sEMG implementation. The experiment involving the LED sensors give a different range of output values based on in which conditions the sensor is placed. Based on the conditions, the output can give a full variety range which helps the ANN to identify each condition easily. As a result, the output shows very accurate identification result. On the other hand, sEMG signal is a complex signal which has a minimal range of output values. Besides of that, human motion does not match even when they are associated with the same tasks. The pressure of the muscles and angles of the movements impacts and changes the sEMG signals. Thus, for better results, this research is focused

on finding the classifications which can identify the motions even if there are slight changes in the output of sEMG signal. Besides of that, it is learned from this research; if any motion is trained for a specific range of time it might give a better – trained output, however, it might end up training the random values which can impact the real – time identification outputs. Thus, a pushbutton function is made which can give different numbers at random times. The pushbutton implementation helps to improve the ANN training. As a result, it shows better identification outputs without showing much noisy signals.

7.2 Future Work

Through this research, it is shown that real – time identification through ANN training is possible. Nonetheless, some improvement might end up with more accurate results.

First of all, there is some ground problem in the sEMG circuit which leads the output to be as a noise signal. Besides the connection of electrodes between the boards get loose sometimes, which also impacts the output results. Thus, a more compact PCB board design is suggested for the ground issue problem of the sEMG circuit.

Moreover, while working with an Arduino board, the author faced the limited memory problem while working with overlapping in real – time. This problem lead the author to follow different paths in order to use a more powerful processor and larger memory, which can calculate more significant overlapping calculation. Therefore, a more powerful microcontroller such as Beaglebone or a similar one is suggested for further research.

Last but not the least, classifications play the most vital role in identification. Choosing the wrong classification might lead the research results to be insufficient. In this research, noise influence was faced because of choosing the wrong classifications. Thus, it is recommended to choose a

better classification algorithm that will help the identification results. For this research purpose, ten classifications are chosen initially. However, while working with sEMG signals, only four classifications are selected because of the other classification methods cause problems to the real – time identification implementation. There are time – domain and frequency – domain classification methods that are used for this research work. Choosing the right classification method, should be considered the most important part in working with sEMG identifications.

REFERENCES

- [1] K. Ziegler-Graham, E.J. MacKenzie, P.L. Ephraim, T.G. Travison, R. Brookmeyer. Estimating the prevalence of limb loss in the United States: 2005 to 2050. *Arch Phys Med Rehabil*, 89 (2008), pp. 422-429
- [2] Atzori, M., and Muller, H. (2015). Control capabilities of myoelectric robotic prostheses by hand amputees: A scientific research and market overview. *Front. Syst. Neurosci.* 9:162. DOI: 10.3389/fnsys.2015.00162
- [3] Sturma A, Roche AD, Göbel P, Herceg M, Ge N, Fialka-Moser V, et al. A surface EMG test tool to measure proportional prosthetic control. *Biomed Tech (Berl)* 2015; 60: 207–2013
- [4] Roche AD, Vujaklija I, Amsüss Sebastian, Sturma A, Göbel Peter, Farina D, Aszmann OC. A structured rehabilitation protocol for improved multifunctional prosthetic control: a case study. *J Vis Exp.* 2015 Nov 06;(105): e52968. DOI: 10.3791/52968
- [5] C. Phram, F. Kayali, I. Vujalija (2017). Increasing motivation, Effort and performance through Game – based rehabilitation for Upper Limb Myoelectric Prosthesis Control. *IEEE*, preprint of the Publication[38]. DOI: 10.1109/ICVR.2017.8007517
- [6] G. N. Saridis, T. P. Gootee, "EMG pattern analysis and classification for a prosthetic arm", *IEEE Trans. Biomed. Eng.*, vol. BME-29, no. 6, pp. 403-412, Jun. 1982.
- [7] B. Hudgins, P. Parker, R. N. Scott, "A new strategy for multifunction myoelectric control", *IEEE Trans. Biomed. Eng.*, vol. 40, no. 1, pp. 82-94, Jan. 1993.

- [8] Guo WC, Sheng XJ, Liu HG, Zhu XY (2017) Toward an enhanced human-machine interface for upper-limb prosthesis control with combined EMG and NIRS signals. *IEEE Transactions on Human-machine Systems* 47(4):564–575
- [9] A. Phinyomark, C. Limsakul, and P. Phukpattaranont, “A novel feature extraction for robust EMG pattern recognition,” *Journal of Computing*, vol. 1, issue 1, pp. 71-80, December 2009.
- [10] Henneman, E., and L. M. Mendell, “Functional organization of the motoneuron pool and its inputs,” in V. B. Brooks, ed., *Handbook of physiology: The nervous system*, American Physiological Society, Bethesda, 1981, pp. 423–507.
- [11] Moritani T, Stegeman D, Merletti R. Basic physiology and biophysics of EMG signal generation. In: Merletti R, Parker P, editors. *Electromyography physiology engineering and noninvasive applications*. Hoboken, NJ: Wiley-IEEE; 2004. P 1–25.
- [12] Carlo J. De Luca (1997) “Use of Surface Electromyography in Biomechanics” *Journal of Applied Biomechanics*, Vol.3
- [13] Carlo J. De Luca (2006) “Electromyography: Encyclopedia of Medical Devices and Instrumentation” (John G. Webster Ed.), John Wiley Publisher
- [14] Paul E. Barkhaus and Sanjeev D. Nandedkar (2000) “Electronic Atlas of Electromyographic Waveforms” Vol. 2, 2nd Edition
- [15] M. Z. Jamal, “Signal acquisition using surface EMG and circuit design considerations for robotic prosthesis,” 2012
- [16] Dr. Scott Day “Important Factors in Surface EMG Measurement,” Bortec Biomedical Incorporated, 2002.

- [17] Peter Konrad, The ABC of EMG: A Practical Introduction to Kinesiological Electromyography, Scottsdale: Noraxon U.S.A, Inc., 2006.
- [18] A Krenker, J Bešter and A Kos,"Introduction to the Artificial Neural Networks",Edited Kenji Suzuki, Published by InTech,, Janeza Trdine, Croatia,(2011), pp 1-18.
- [19] M. T. Hagan, H. B. Demuth and M. H. Be, Neural Network Design, Boston: PWS pub, 1996.
- [20] D. A. Winter, Biomechanics and Motor Control of Human Movement, New Jersey: Wiely & Sons Inc., 2009.
- [21] S. Kumar and A. Mital, Electromyography In Ergonomics, London: Taylor & Francis Inc., 1996.
- [22] C. R. Jeffrey and D. Maya, "The History of Muscle Dysfunction and SEMG,"*Journal of Electromyography and Kinesiology*, vol. 4, pp. 5-14, 1994.
- [23] Noraxon, "Manufacturers of professional electromyography products (emg/semg) and biomechanical sensors," 10 March 2006. [Online]. Available: ww.noraxon.com. [Accessed 315 March 2018].
- [24] G. L. X. Z. J. H. a. G. L. A. Xiong, "Feasibility of EMG-based ANN controller for a real-time virtual reality simulation," in *38th Annual Conference on IEEE Industrial Electronics Society*, Montreal, 2012.
- [25] A. M. S. P. D. a. B. R. S. Micera, "A hybrid approach to EMG pattern analysis for classification of arm movements using statistical and fuzzy techniques," *Med. Eng. Phys.*, vol. 21, pp. 303-311, 1999.
- [26] D. W. Marquardt, "An Algorithm for Least Square Estimation of Non-Linear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11,no. 2, pp. 431-441, 1963.

[27] MS thesis by Pavan Kumar Yarlagadda, " Real – time sEMG-based finger joint angle control for a smart prosthetic hand," ISU, May 2013

[28] <http://www.ti.com/lit/ds/symlink/uaf42.pdf> [Accessed 3/30/2018]

[29] MS thesis by Abduljaleel (AJ) Alriyadh, " EMG Feature-Based Approach toward a Robust Artificial Neural Networks Analysis of Human Finger Motion," ISU, May 2016

[30] MATHWORKS, "Simulink on/off switch" by Gleen Gomes, 2012 [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/39348-simulink-on-off-switch> [Accessed 3/30/2018]

APPENDIX

Sensor – based LED example experimentation setup

At first, after circuit setup connection is done, the Arduino should relate to PC through USB cable. The following Arduino code need to be uploaded for serial communication which has one Analog input channel configuration with a baud rate of 115,200.

```
// These constants won't change. They're used to give names
// to the pins used:

const int analogInPin = A2; // Analog input pin that the Channel is attached
to

int sensorValue = 0;          // value read from the Channel

void setup() {
    // initialize serial communications at 115200 bps:
    Serial.begin(115200);
}

void loop() {
    // read the analog in value:
    sensorValue = analogRead(analogInPin);
    // print the results to the serial monitor:
    //Serial.print("sensor = ");

    //If else loop showing 0 to 1023-bit output through serial port

    if (sensorValue > 999) {
        Serial.print(sensorValue); //When the bit value is greater than 999
    }
```

```

else if (sensorValue > 99) //When the bit value ranges from 100 to 999
{
    Serial.print(0);

    Serial.print(sensorValue);
}

else if (sensorValue > 9) //When the bit value ranges from 10 to 99
{
    Serial.print(0);

    Serial.print(0);

    Serial.print(sensorValue);
}

else
{
    Serial.print(0);

    Serial.print(0);

    Serial.print(0);

    Serial.print(sensorValue); //When the bit value ranges from 0 to 9
}

Serial.print('\n');

// for the analog-to-digital converter to settle
// after the last reading:
delay(50);
}

```

Afterward, Simulink™ model needs to set up for collecting the signal from the USB cable. Simulink™ model should be selected as normal mode. Run on hardware configuration should be chosen as Arduino Mega 2560 and serial 0 and 1 baud rate should be chosen as 115,200. See the below figures:



Figure 0.1: Simulink™ run time mode selected as normal

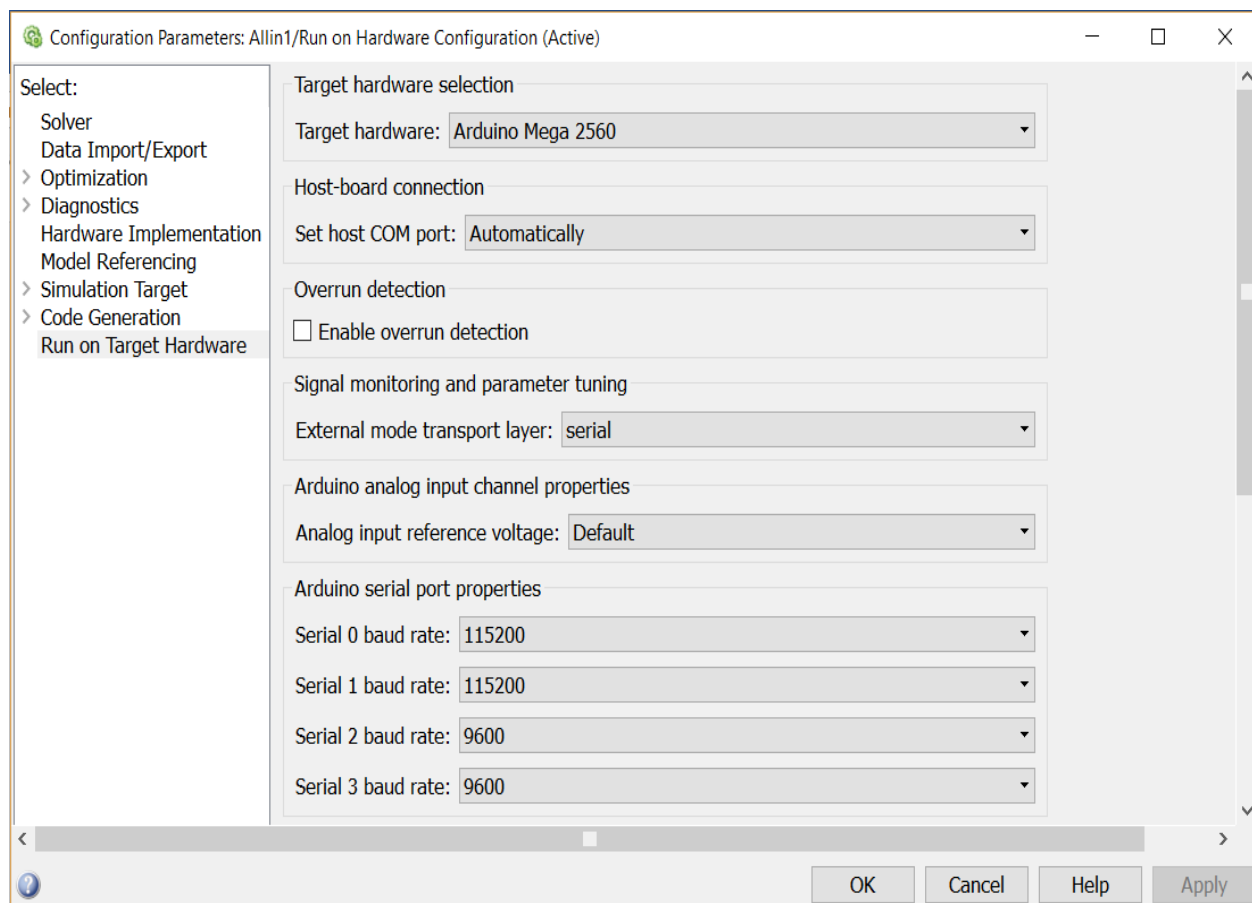


Figure 0.2: Arduino Mega 2560 setup for Simulink™ model

See Figure 5.8, which shows the Simulink™ model including the buffer (Overlapping purpose) and the classifications for real – time experiment. In that model, parameters need to be selected for serial receive block and serial configuration block (see Figure 5.3). Figure 0.3 shows the buffer parameters.

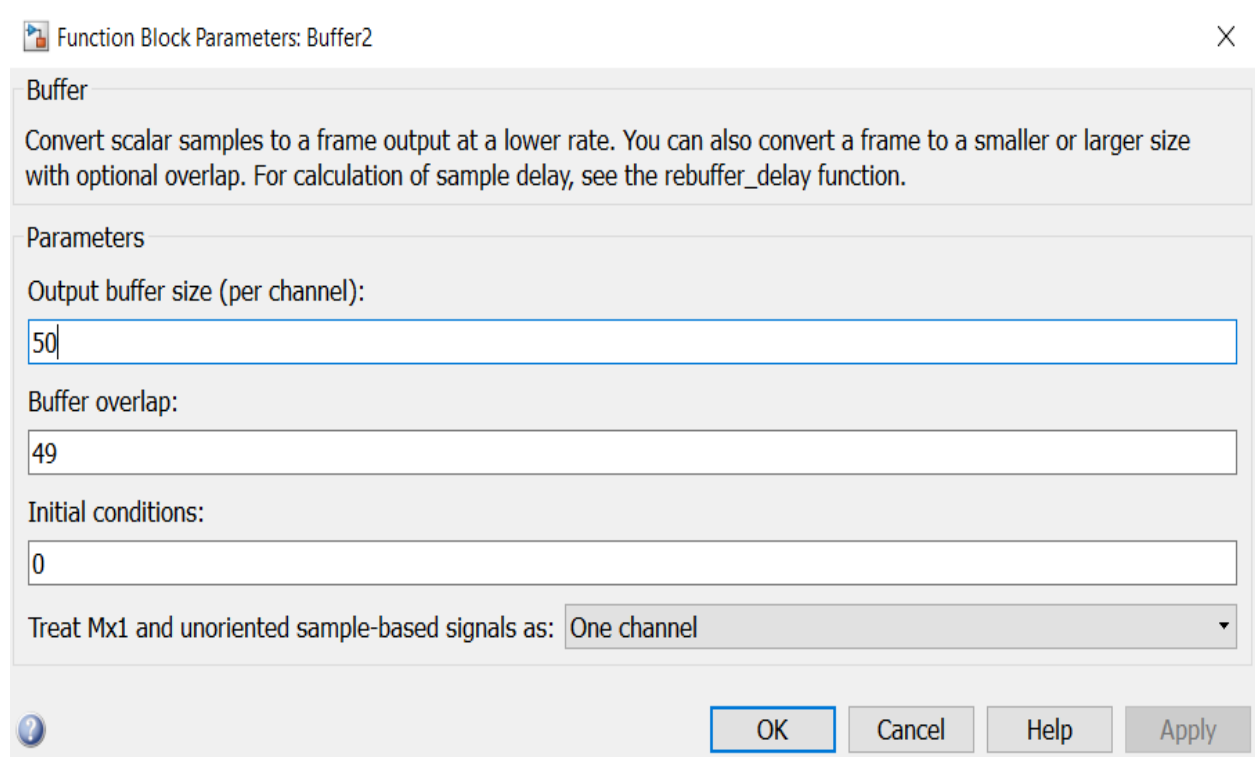


Figure 0.3: Buffer block parameter. Here, 50 overlapping is chosen

Classifications

As discussed earlier in this thesis, ten classifications are developed to extract the dynamics from a given real – time sEMG signal. All the ten classifications are programmed in MATLAB® as functions and grouped in a different MATLAB® functions Simulink™ blocks. The following MATLAB® codes are shown for ten different classifications.

Integrated EMG classification:

```
function y = fcn(u)
N=length(u);
IEMG=zeros(N,1);
for n=1:N
    IEMG(n)=abs(u(n)-mean(u));
end
```



```
y=sum(IEMG);
```

Mean Absolute Value classification:

```
function y = fcn(u)

N=length(u);

MAV = zeros(N,1);

for n=1:N

    MAV(n)=abs(u(n)-mean(u));

end

y=sum(MAV)/N;
```

Mean Absolute Value Slope classification:

```
function y = fcn(u, u_)

N=length(u);

MAV = zeros(N,1);

for n=1:N

    MAV(n)=abs(u(n)-mean(u));

end

y=sum(MAV)/N;

MAV_ = zeros(N,1);

for n=1:N

    MAV_(n)=abs(u_(n)-mean(u_));

end

y= y-(sum(MAV_)/N);
```

Simple Square Integral classification:

```
function y = fcn(u)
N=length(u);
SSI = zeros(N,1);
for n=1:N
    SSI(n)=abs(u(n)-mean(u)).^2;
end
y=sum(SSI);
```

Root Mean Square classification:

```
function y = fcn(u)
N=length(u);
x=zeros(N,1);
for n=1:N
    x(n)=u(n)-mean(u);
end
y=rms(x);
```

Wavelength Length classification:

```
function y = fcn(u)
N=length(u);
WL = zeros(N,1);
for n=1:N-1
    WL(n)=abs(u(n+1)-u(n));
end
y=sum(WL);
```

Zero Crossing classification:

```
function y = fcn(u)
N = length(u);
u = u - mean(u);
y = 0;
for i = 1:N-1
    if u(i)*u(i+1) < 0
        if abs(u(i)-u(i+1)) >= 0.05 % white noise/background noise
            y = y+1;
        end
    end
end
end
```

Slope Sign Change classification:

```
function y = fcn(u)
N = length(u);
y = 0;
for i = 2:N-1
    if (u(i)-u(i-1))*(u(i)-u(i+1)) > 0.01

        y = y+1;

    end
end
end
```

Willison Amplitude classification:

```
function y = fcn(u)
N=length(u);
```

```

y = 0;
for n=1:N-1
    if abs(u(n)-u(n+1)) > 0.05;
        y=y+1;
    end
end

```

Mean classifications classification:

```

function y = fcn(u)

y=mean(u);

```

For targeted output, a signal builder block is used which can generate the values one, two, and three. After real – time classifications, the stored MATLAB® classification values are used for offline training Artificial Neural Network (ANN). The training has been done by using classification values as input values and the targeted output. A function named stitch is used to skip the values while moving from one position to other positions by multiplying zero. The main purpose of using this function to reduce the additional noise. Figure 0.4 shows the block diagram of targeted output and its parameter configuration.

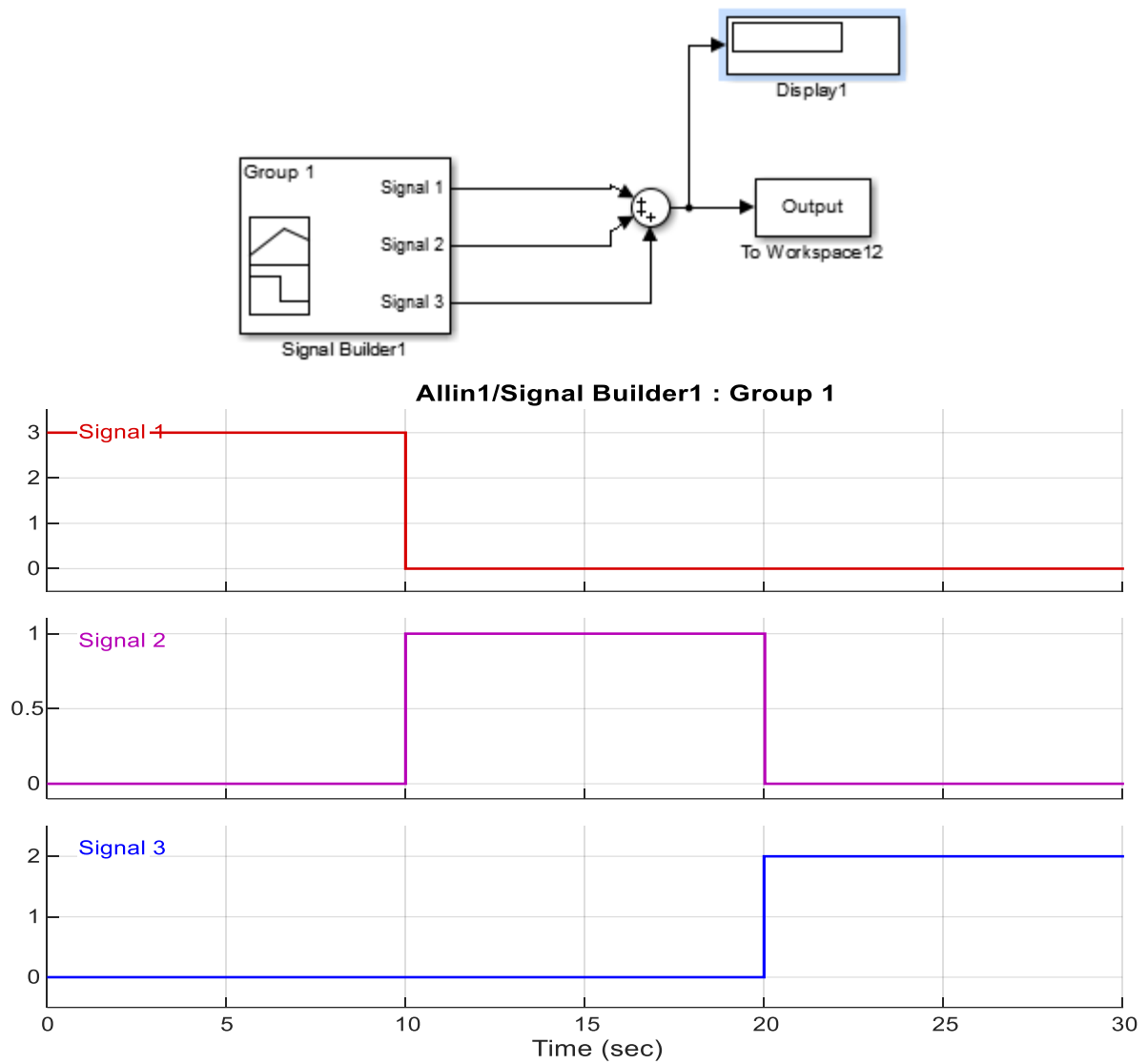


Figure 0.4: Targeted output Simulink block and its configuration.

The following MATLAB[®] code showed how stitch function MATLAB[®] code is made.

```
function[x] = stitch (x,t, i, j, k, l)

mask = ~(((t>=i(1)) & (t<=i(2))) | ((t>=j(1)) & (t<=j(2))) | ((t>=k(1)) & (t<=k(2)))
| ((t>=l(1)) & (t<=l(2))));

x = x+1;

x = x.*mask;

x(x==0) = [];
```

```
x = x - 1;
```

Afterward, extracted the classification values and converted the values in the same dimensions where targeted output has the same amount of matrix dimension. Later, both values are trained by using backpropagation algorithm. Also, the error values are calculated by comparing the targeted output and predicted output values. The following MATLAB® is shown for offline ANN training.

```
fs = 0.067;

% Stitch function parameters
i=[0 1];
j=[10 11 ];
k=[20 21];
l=[29 30];
t=f1.time;
t=t';

% Ten classification values collected from the workspace
x1=squeeze(f1.signals.values);
x2=squeeze(f2.signals.values);
x3=squeeze(f3.signals.values);
x4=squeeze(f4.signals.values);
x5=squeeze(f5.signals.values);
x6=squeeze(f6.signals.values);
x7=squeeze(f7.signals.values);
x8=squeeze(f8.signals.values);
x9=squeeze(f9.signals.values);
x10=squeeze(f10.signals.values);

% Transpose the classification values to meet up the same dimension with targeted
output
```

```

x1=x1';
x2=x2';
x3=x3';
x4=x4';
x5=x5';
x6=x6';
x7=x7';
x8=x8';
x9=x9';
x10=x10';

% Targeted Output value
o=squeeze(Output.signals.values);
o=o';

% Stitch function is used for reducing additional noise
x1_=stitch(x1,t,i,j,k,l);
x2_=stitch(x2,t,i,j,k,l);
x3_=stitch(x3,t,i,j,k,l);
x4_=stitch(x4,t,i,j,k,l);
x5_=stitch(x5,t,i,j,k,l);
x6_=stitch(x6,t,i,j,k,l);
x7_=stitch(x7,t,i,j,k,l);
x8_=stitch(x8,t,i,j,k,l);
x9_=stitch(x9,t,i,j,k,l);
x10_=stitch(x10,t,i,j,k,l);

o_=stitch(o,t,i,j,k,l);

Channel1=[x1_;x2_;x3_;x4_;x5_;x6_;x7_;x8_;x9_;x10_];

% Training the NN
net=newff(minmax(Channel1),[40,1],{'logsig','purelin','trainlm'});
net.performFcn='msereg';

```

```

net.performParam.ratio=0.5;
net.trainparam.epochs=8000;
net.trainparam.goal=1e-25;
net.trainparam.lr=.067;
net=train(net,Channel1,o_);
v1=net([x1;x2;x3;x4;x5;x6;x7;x8;x9;x10]);
error=o-v1;

% To plot the graph of targeted output and predicted output and showing the error
value
figure
plot(t,o,t,v1,'r')
figure(2)
plot(error)
error=sum(abs(error))

```

After training is done, a Simulink™ block is created based on the trained values by using the command ‘genism(net)’ in the command window. Figure 0.5 shows the Simulink™ block diagram generated from the trained value.

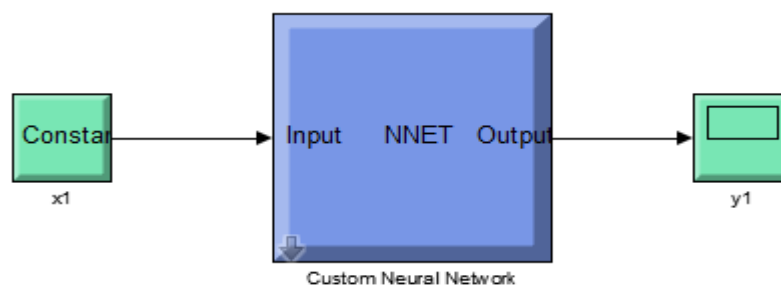


Figure 0.5: Generated Simulink™ block from the trained NN

From the Figure 0.5, only Custom Neural Network and y1 block is implemented in the different Simulink™ block where input values are collected from Arduino analog input through the USB cable through overlapping and classification in real – time. A MATLAB® function block is used

to convert the ten classification input values into one output value. For more understanding, referred to see Figure 5.15. Below the shown code is used for converting the ten inputs into one output value.

```
function y = fcn(u1,u2,u3,u4,u5,u6,u7,u8,u9,u10)
%#codegen
y = [u1;u2;u3;u4;u5;u6;u7;u8;u9;u10];
```

By following the above steps, it is possible to observe real – time identification for a sensor – based LED light.

Real – time sEMG identification experimental setup

For the sEMG identification, the followed steps almost the same as sensor – based LED light. However, for sEMG identification, two Arduino analog inputs are selected instead of one analog input. Thus, there has been made a little change in the Arduino code. Also, instead of using ten classifications only four classifications are used. Four classifications: Slope Sign Change (SSC), Waveform Length (WL), Zero Crossing (ZC), and Wilson Amplitude (WAMP) have same MATLAB® function codes which is used in the LED experiment. Buffer overlapping is selected as 100. And these four classifications are trained individually instead of using altogether as LED experiment. Also, avoided the stitch functions because, the targeted output is also done real – time (see Figure 6.2) in order to achieve better identification. Comparing the LED experiment with sEMG there are two major changes in the codes: one in Arduino code and another is for training the ANN. Also, a new Simulink™ block named saturation block is used to outline the real – time identification value which prevent to go beyond the targeted output values. Figure 0.6 shows the configuration parameters for saturation block. Otherwise, all the steps are same as LED experiment

just few changes have made in the Simulink™ block as individual classification is done. For better understanding recommended to see Figure 6.5 and 6.13. In the following the Arduino code and ANN training code are given which are used in sEMG experimentation.

Arduino Code:

```
void setup ()
{
    Serial.begin (115200);

} // end of setup

void loop ()
{
    for (int whichPort = A2; whichPort <= A3; whichPort++)// for collecting the
analog inputs from two different channels
    {
        analogRead (whichPort);

        int result = analogRead (whichPort);

        //Serial.println("Status: " + status);

        if (result>999)
        {
            Serial.print (result);

        }

        else if (result>99)
        {

            Serial.print (0);

            Serial.print (result);
```

```

    }

    else if (result>9)
    {
        Serial.print(0);
        Serial.print(0);
        Serial.print (result);
    }

    else
    {
        Serial.print(0);
        Serial.print(0);
        Serial.print(0);
        Serial.print (result);
    }
}

Serial.println();
delay (50);
} // end of loop

```

MATLAB® code to train the ANN:

```

fs = 0.067; % Sampling time

t=a.time;
t=t';

% Classification values from two analog inputs
i1=squeeze(a2.signals.values);
j1=squeeze(b2.signals.values);

% Targeted outptu value
outputa=squeeze(simout.Data);

```

```

o=outputa';

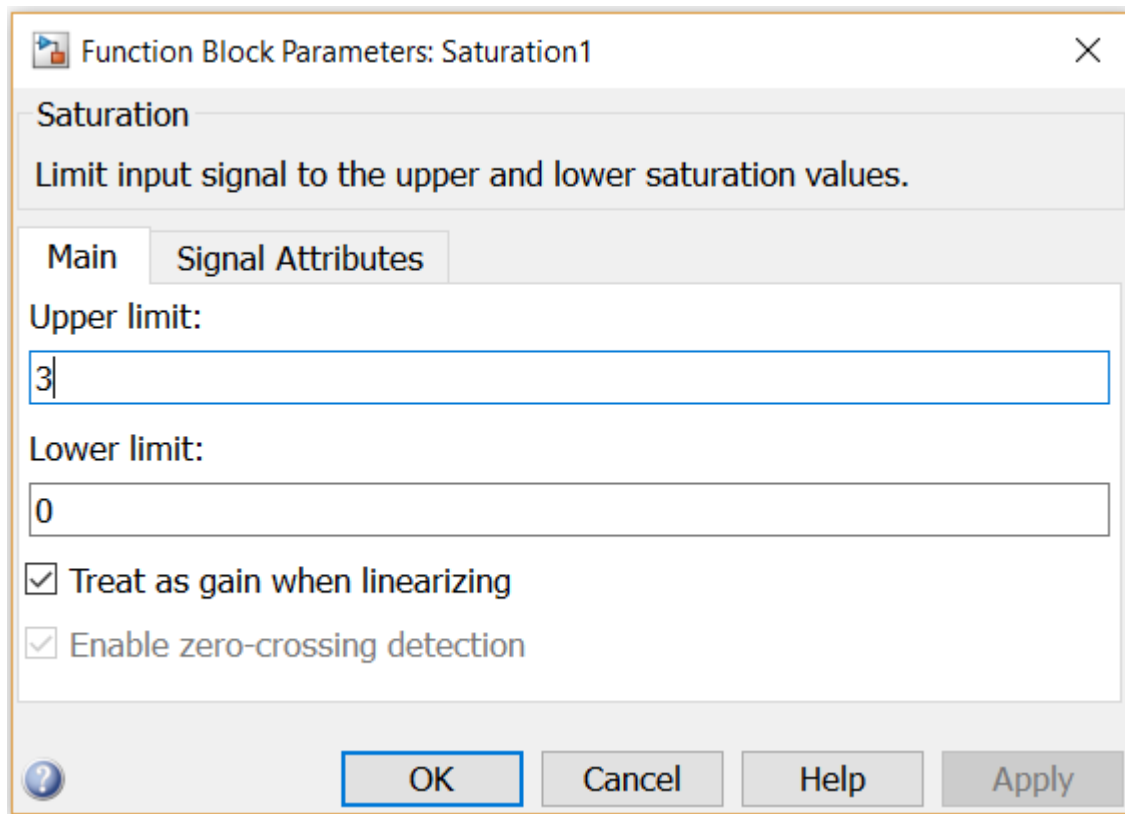
% Converting the input values and targeted output into same dimension
inputst=[i1 j1];
inputs=inputst';
p = inputs;
outputb = o;

% Traing the NN
net = newff(p,outputb,40,{'tansig','purelin'},'trainlm');
net.divideFcn = '';
net.performFcn='msereg';
net.performParam.ratio=0.07;
net.trainParam.show = 500;
net.trainParam.lr = 0.05;
net.trainParam.epochs = 5000;
net.trainParam.goal = 1e-15;
net = init(net);
[net,tr] = train(net,p,outputb);
y = sim(net,p);
v1=net(inputs);
error=o-v1;

% Plotting the graph
figure
plot(t,o,t,v1,'r')

figure(2)
plot(error)
error=sum(abs(error))

```



The image shows a software dialog box titled "Function Block Parameters: Saturation1". It has a close button (X) in the top right corner. The dialog is divided into two tabs: "Main" and "Signal Attributes". The "Main" tab is currently selected. Below the tabs, there is a description: "Limit input signal to the upper and lower saturation values." Under this, there are two input fields. The first is labeled "Upper limit:" and contains the value "3". The second is labeled "Lower limit:" and contains the value "0". Below these fields are two checked checkboxes: "Treat as gain when linearizing" and "Enable zero-crossing detection". At the bottom of the dialog, there is a help icon (question mark) on the left, and four buttons: "OK", "Cancel", "Help", and "Apply".

Function Block Parameters: Saturation1

Saturation

Limit input signal to the upper and lower saturation values.

Main Signal Attributes

Upper limit:

3

Lower limit:

0

☒ Treat as gain when linearizing

☒ Enable zero-crossing detection

? OK Cancel Help Apply

Figure 0.6: Saturation block configuration parameters