12-2015

# EEG-Controlling Robotic Car and Alphabetic Display by Support Vector Machine for Aiding Amyotrophic Lateral Sclerosis Patients

Quang M. Le

# EEG-Controlling Robotic Car and Alphabetic Display by Support Vector Machine for Aiding Amyotrophic Lateral Sclerosis Patients

EEG-Controlling Robotic Car and Alphabetic Display by Support Vector Machine for Aiding Amyotrophic Lateral Sclerosis Patients

A thesis submitted in partial fulfillment
of the requirements for the degree of
Electrical Engineering

by

Quang Le

Graduation December and 2015
University of Arkansas

[illegible]
Dr. Baohua Li
Thesis Director

[illegible]
Dr. Jingxian Wu
Committee Member

Second Committee Member Name
Committee Member

Third Committee Member Name
Committee Member

Fourth Committee Member Name
Committee Member

## Abstract

This thesis presents the design and experiment of a system that can detect the human thinking such as driving directions and letters using the brainwave signals known as electroencephalogram (EEG) and a machine learning algorithm called support vector machine (SVM). This research is motivated by amyotrophic lateral sclerosis (ALS) disease which makes patients seriously lose mobility and speaking capabilities. The developed system in this thesis has three main steps. First, wearing EPOC headset from Emotiv Company, a user can record the EEG signals when he/she is thinking a direction or a letter, and also save the data in a personal computer wirelessly. Next, a large amount of EEG data carrying the information of different directions and letters from this user are used to train SVM classification model exhaustively. Finally, the well-trained SVM model will be used to detect any new thought about directions and letters from the user. The detection results from the SVM model will be transmitted wirelessly to a robotic car with LCD display built with Arduino microcontrollers to control its motions as well as the alphabetic display on LCD. One of the great potential applications of the developed system is to make an advanced brain control wheel chair system with LCD display for aiding ALS patients with their mobility and daily communications.

**Keywords: SVM (support vector machine), EEG (electroencephalogram), EPOC headset, ALS (amyotrophic lateral sclerosis).**

## Acknowledgement

Firstly, I would like to say many thanks to Dr. Baohua Li, my mentor for giving me a lot of supports not only in this research but also during my undergraduate study. Secondly, I want to say thanks to Qing Guo, a former master student in the electrical engineering department.  My research cannot be done without learning a lot from her experience. Also, I want to thank two freshmen students Mauricio Iglesias and Caleb Woodall, because they showed me how to use the Emotiv Epoc device at the very beginning of the research. Then, I want to say thanks to my girlfriend who help me to check the format and typos. Finally, I want to thank my parents for their caring and support during my four years of Education in the U.S., and my grandmother for being the one who always mentally supports and prays for my academic success.

## Table of Contents

## List of Figures

## List of Tables

# Chapter 1: Introduction

## 1.1 Background and motivation

An EEG signal measures summation of electrical activities of thousands or even millions of neurons that have the similar spatial orientation in the human brain. It can be acquired through a small, flat electrical disk (electrode sensor) sitting directly on a location at the scalp [1] and is normally represented as a waveform in microvolt unit. The neural populations in different brain locations may generate EEG signals with different statistical properties when human beings are thinking different tasks. Therefore, multi-channel EEG signals may have different patterns in terms of tasks. From this principle, EEG pattern recognition through machine learning algorithms will be able to identify thoughts of human beings. For example, when human beings plan two possible driving directions which are forward and backward, the generated EEG data can be classified using the support vector machine (SVM) [2&3] algorithm into two categories associated with the directions. Such EEG classification outcomes with high accuracy will be able to provide a method and technology for mobile motion control. When human beings imagine or think English words letter by letter in minds, the EEG patterns may vary with letters and therefore, the EEG data carrying the letter information can be even classified into different categories by the SVM algorithm [1&2]. The accurate classification results will be able to provide a new way for communication. In addition, for applications related to human brains, EEG acquisition technology can be low-cost, safe, portable, and wireless compared with other technologies such as functional magnetic resonance imaging and invasive surgery of implanting electrodes into brain for neural spikes. In our research, EPOC package from Emotiv Company has been used to record EEG signals.

Amyotrophic lateral sclerosis (ALS) is a progressive chronic motor neuron disease to make patients lose mobility and speaking capabilities [4]. This disease occurs when specific nerve cells in the brain and spinal cord that control voluntary movement gradually degenerate. The loss of these motor neurons causes the muscles under their control to weaken and waste away, leading to paralysis and difficulty in speaking [5]. As a result, ALS patients may not be able to move or speak normally. ALS affects about 20,000 Americans with 5,000 new cases occurring in the United States each year. Unfortunately, the cause of this disease process is unknown, there is no cure for ALS, and the patients may die within 5 years of the time the diagnosis of this disease is made.

Motivated by ALS disease and properties of human brain EEG signals, our research is to develop a EEG-based prototype system that can monitor the human brain activity, detect the human's thought such as moving directions and letters, and then control robotic car motion and alphabetic display on LCD. In the research for developing the system, we use Emotiv EPOC package, a personal computer, Matlab software, and microcontrollers (Arduino package). Such a system can also be called a brain computer interface (BCI) system. Based on the developed system, an application such as an advanced EEG-based brain control wheelchair with LCD display can be further developed to improve the lift quality of ALS patients with better mobility and communications.

## 1.2 EEG-based BCI system for robotic car and alphabetic display control

In this Honor thesis, we focus on creating a BCI system for controlling robotic car motion and alphabetic display on LCD through EEG signals only. The main challenge of developing such a system is that the data originated from the human brain are normally extremely noisy. Therefore, we need use some special or complicated signal processing and data analysis techniques to recognize the human thought with high accuracy. For example, we use SVM, a popular sophisticated nonlinear machine learning algorithm, to classify EEG patterns into different classes for decoding the human thinking. The main idea of our EEG-based BCI system is as follows. First, a user records the EEG signals when he/she is thinking a direction or a letter through EPOC headset, and also save these time-domain data in a personal computer (PC) wirelessly and automatically. Then, the time-domain EEG data are converted into power spectrum density (PSD) in the frequency domain as the input of the SVM algorithm. The output of the SVM algorithm would be the corresponding class of the input. In our research, the output would be a direction or a letter. Before the SVM model can be used to detect any new thought about the directions and the letters from the user, its parameters need to be optimized. To do that, a large amount of training data which are pairs of inputs and known outputs are generated by the user. In our project, the SVM model is extensively optimized through libSVM library [6]. After that, the well-trained SVM model can be used to classify any new EEG trial into a direction or a letter. The classification result from the well-trained SVM model is then sent to the robot through Arduino board, Matlab, and Xbee to control the driving direction and LCD alphabetic display remotely. When one EEG trial is processed completely, another EEG trial will be converted into the PSD in the frequency domain. Then, this PSD will be input into the well-trained SVM model. Next, the SVM output will be used to move the robot or control LCD display accordingly.

To test the performance of the designed BCI system, we recorded 6000 EEG trials, 5 seconds long each in two different categories: directional category and alphabetic category. In the directional category, we focused on 4 directions: left, right, forward, and backward. We took 1000 data for each direction. Then, in the alphabetic category, we focused on 2 letters: A and B.

We took 1000 data for each letter. Next, one part of these EEG trials with the corresponding direction or letters were used to train a SVM classification model for each category. The other parts of the EEG trials were used to show the performance of the developed BCI system.

## 1.3 Thesis outline

This thesis includes eight chapters. Chapter 1 presents the background and motivation for developing an EEG-based BCI system for robot motion and alphabetic display control and summarizes the main steps of such a system. In Chapter 2, the selected software and hardware will be discussed. Chapter 3 focuses on the optimization and implementation of the SVM model, which is the core of our BCI system. The details about SVM functions and their usage will be provided. Then, Chapter 4 will show how to use the programs developed for this project step by step. Next, Chapter 5 will discuss the rule and method of high-quality EEG data recording through our experience and the feature chosen as the input of the SVM model. We developed the system starting from the simple 2-class classification problem, then to the 3-class problem, then to the complicated 4-class problem. Chapter 6 will show the experimental results about 2-class classification, 3-class classification, and 4-class classification with different frequency ranges. In addition, since the SVM model is the core of our BCI system, a toy classification problem is created to verify the correction of the SVM program. Chapter 7 will show how to implement and test the BCI system on the robotic car with LCD. Finally, Chapter 8 will draw some conclusion of the project and propose some future work to further improve the system.

# Chapter 2: Hardware and Software for EEG-based BCI System

## 2.1 Hardware selection

### a) Emotiv EPOC

The Emotiv EPOC headset as shown in Figure 1 from Emotiv Company [7] is capable of recording human brain EEG signals. This device is chosen for two main reasons. Firstly, Emotiv EPOC is a well-known device in the market and research filed and so it is very safe for us to do experiment. Secondly, it is a very low cost device compared to other EEG-headsets in the market and therefore it makes our developed system low cost. The EPOC device provides 14 channels plus 2 references with the sampling frequency of 128 data per second. The 14 channels have the following names: 1-AF3, 2-F7, 3-F3, 4-FC5, 5-T7, 6-P7, 7-O1, 8-O2, 9-P8, 10-T8, 11-FC6, 12-F4, 13-F8, 14-AF4. Different channels have different characteristics, because they are corresponding to different brain areas. The names of channels can reflect their brain areas. Actually, F, P, O, and T represent frontal lobe, parietal lobe, occipital lobe, and temporal lobe, receptively. However, C does not mean center lobe because there is no center lobe at all. Instead, C means the center area of the brain. The odd numbers in the names represent left hemisphere and the even numbers represent right hemisphere. Please refer to 10-10 EEG electrode placement system and left and right brain hemispheres as shown in Figure 2 for specific channels and brain areas.

**Figure 1 Emotiv EPOC headset**[1]

Emotiv EPOC package also comes with an SDK called “EPOC control panel” that allows the user to monitor the EEG signal quality of each channel. As seen in Figure 3, channels with green color provide good signals while yellow are for medium, orange and red are for weak signals and black means no signal at all. This SDK is used every time the BCI is running, so that if the signal quality is low (yellow, orange, red or black), the user has to add some moisture (saline solution) to the channel to get green signal. This checkup may be done several times during the BCI running process to ensure the highest quality of the data.

[1] Figure 1:http://emotiv.com/upload/media/1_big.jpg

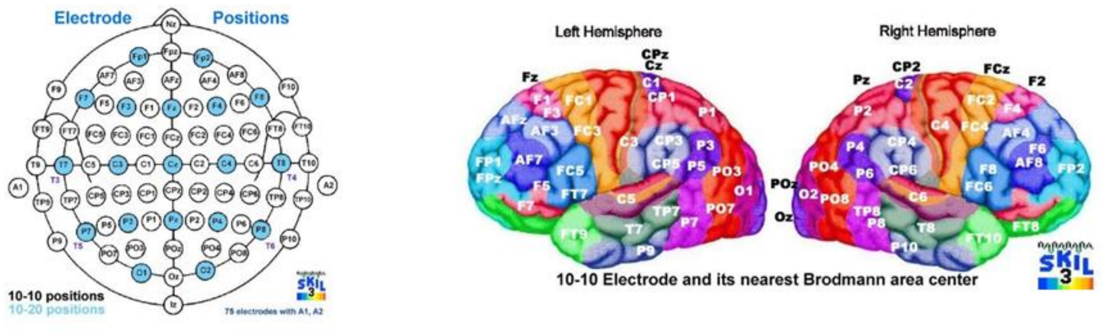
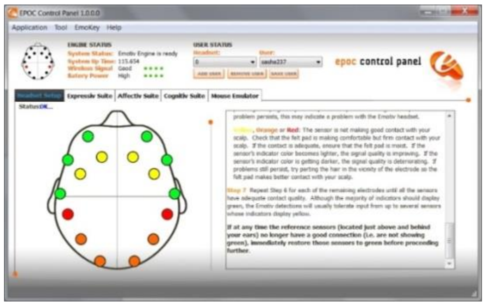


**Figure** 2 **10-10 EEG electrode system, left hemisphere, and right hemisphere** [2]



ENGINE STATUS
System Status: Emotiv Engine is ready
System Up Time: 115.654
Wireless Signal: Good
Battery Power: High

USER STATUS
Headset: 0
User: sasha237
ADD USER · REMOVE USER · SAVE USER

Status:OK

problem persists, this may indicate a problem with the Emotiv headset.

Yellow, Orange or Red: The sensor is not making good contact with your scalp. Check that the felt pad is making comfortable but firm contact with your scalp. If the contact is adequate, ensure that the felt pad is moist. If the sensor's indicator color becomes lighter, the signal quality is improving. If the sensor's indicator color is getting darker, the signal quality is deteriorating. If problems still persist, try parting the hair in the vicinity of the electrode so the felt pad makes better contact with your scalp.

Step 7 Repeat Step 6 for each of the remaining electrodes until all the sensors have adequate contact quality. Although the majority of indicators should display green, the Emotiv detections will usually tolerate input from up to several sensors whose indicators display yellow.

**If at any time the reference sensors (located just above and behind your ears) no longer have a good connection (i.e. are not showing green), immediately restore those sensors to green before proceeding further.**

**Figure 3 EEG EPOC control panel SDK**

## b) Arduino board

In this project, the Arduino Uno board as shown in Figure 4 is the main control board for the robotic car and the wireless robot controller. This board has 14 digital input/output pins with 6 analog inputs. It also has a 16 Mhz quartz crystal, USB connection, a power-jack, and a reset button. For this project, two Arduino boards will be needed for wireless communication between a personal computer (PC) and the robotic car. The first board is connected directly to the computer through a USB cable with the BCI software to create a Matlab-Arduino interface. All

[2] Figure 2: http://www.brainm.com/software/pubs/dg/BA_10-20_ROI_Talairach/nearesteeg.htm

of the results calculated from Matlab programs will be sent into this first board. The first board will be used to control a device called Xbee that can send signals wirelessly to the second board to control the robotic car. This will be discussed in more details in Chapter 7.

**Figure 4 Arduino Uno R3 board** [3]

### c) Xbee board

Xbee board as shown in Figure 5 is a simple antenna that is used in a lot of electronics applications. In this project, this board will serve as the main communication channel between the BCI and the robot car.

**Figure 5 Xbee antenna**[4]

[3]Figure 4: https://upload.wikimedia.org/wikipedia/commons/3/38/Arduino_Uno_-_R3.jpg

[4] Figure 5: https://cdn.sparkfun.com//assets/parts/1/8/2/0/08665-00.jpg

## 2.2 Software selection

To build the efficient BCI system for this project, some unique software and packages are chosen.

**a) Matlab**

Matlab is a multi-paradigm computing software that is used a lot in research area. The interfacing with Matlabs program can be built based on other languages such as C, C++, Java, and Python. It is also an open-source software that let the users adding many packages and functions. For this project, there are two main reasons to choose Matlab. Firstly, it does support the package from Libsvm library which is a very efficient toolbox to optimize and implement SVM model. Secondly, although Matlab is a numerical computing language, it does have a lot of packages that allow the user to control the hardware and microcontroller such as Arduino and Raspberry Pi. By this advantage, we will be able to use Matlab command window to control and send signals to an Arduino board.

**b) EPOC Simulink EEG importer**

In Qing Gou's Master thesis [8], to collect the data, Qing used software from Emotiv called "Emotiv Control Panel". This software developed from Emotiv allows users to collect the raw EEG data from the EPOC device and then save the data in CVS type used in Microsoft Excel. If we use Emotiv Control Panel in this project for EEG data collection, we have to manually convert the EEG data into Matlab format (.mat) to run the SVM algorithm. So, we could not develop an automatic system. Instead, we used "EPOC Simulink EEG Importer" which is a package that allows a user to collect the EEG data through Simulink into Matlab environment automatically. This package is also very easy to use. Firstly, an exe file is run in order to connect the headset with the Simulink as shown in Figure 6. After the connection is set, a Simulink file is called through some simple Matlab code and collect new data such as res.mat every single time as shown in Figure 7.

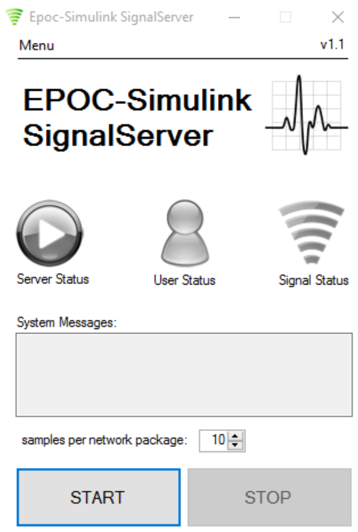

Epoc-Simulink SignalServer
Menu
v1.1
EPOC-Simulink SignalServer
Server Status
User Status
Signal Status
System Messages:
samples per network package:
10
START
STOP

**Figure 6 EPOC simulink signal server**

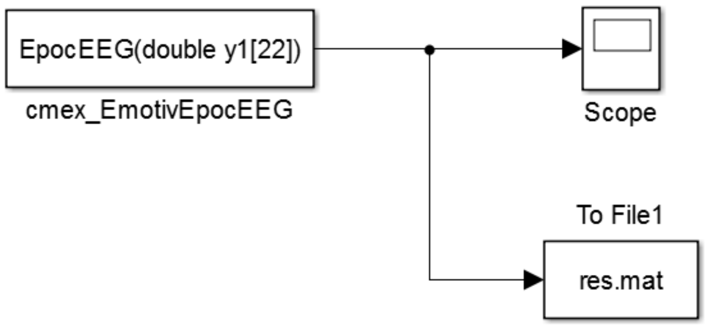


**Figure 7 A simple simulink model for EEG data collection**

# Chapter 3 Optimization and Implementation of Support Vector Machine

As briefly introduced in Chapter 1, Support vector machine or SVM is a set of machine learning methods which are used for classifying different data features, or also regression and outliers' detection [2]. Unlike other machine learning method such as logistic model tree (LMT) or Quadratic classifier [9&10], SVM is a kernel based non-linear, non-parametric classification technique which has shown good classification applications in medical analysis, character recognition, or electric load forecasting and other classification fields. SVM shows very good results on classifying data with large dimensionality and it can takes care of the "curse of dimensionality" [11]. In [11], the author pointed out that "the curse of dimensionality is a well-known but not entirely understood phenomenon. Too much data, in terms of the number of input variables, is not always a good thing". Also, for only one class, unknown class, or significantly unbalanced classes, the classification problem becomes very complicated. In this thesis, although we are not dealing with the unbalanced or one class type of classification, our problem is coming from the credibility of the data. Since the data is EEG type, the classification problem is normally high dimensional and complicated. Therefore, SVM method may be the optimal solution for the problem related to EEG data.

Plenty of research has been conducted for the applications of SVM into EEG data analysis. For instance, in [12] an experiment is set up to discriminate between imagination of right and left hand movement. To conduct this experiment, three subjects were asked to imagine about left and right movements for 1.5 seconds with a 10 seconds interval gap. There were 160 hand-movements data recorded. These data were then converted to power spectrum density (PSD) and frequency bands of 9-14 Hz are used for classification. The results showed that 80% of testing accuracy can be achieved. This has shown the possibility of using SVM as the classifier for EEG data. However, for the statistical point of view, the small size of training set such as a set of 160 data may be not very convincing to draw any conclusion. Therefore, in this project, we will create a large number of training data.

## 3.1 Mathematical view of SVM classifier

### a) Basic concept of SVM

We will use linear SVM classifier of two classes to show its basic concept. A two-class SVM classifier requires a set of training data of the input *x* and the output *y* to optimize the parameters in the SVM model. This set has to matrix X and Y for:

$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^n, y_i \in \{-1\,,1\}\}_{i=1}^{m} \qquad (3.1.1)$$

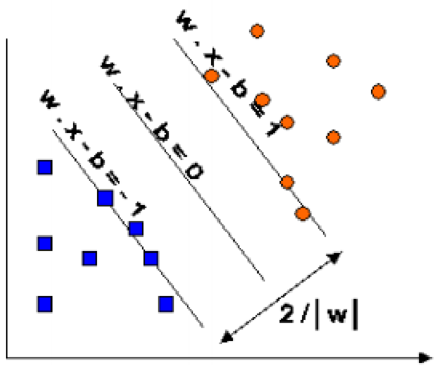


**Figure 8 A simple hyperplane dividing a set of data** [5]

Where $x_i$ is a row vector representing a value of the input  that is mapped to a label or a value of the output $y_i$ =1 or -1. A hyper lane can be written as a set of point x satisfying that:

[5] Figure 8: http://www.jatit.org/volumes/research-papers/Vol12No1/1Vol12No1.pdf or see [9]

$$\{x: f(x) = w^T x + b = 0\} \qquad (3.1.2)$$

This hyperplane separates the data set in two different regions. The upper region covers the positive set while the lower region covers the negative set [13] as seen in Figure 8.

**b) Margin and optimization**

There are many hyperplanes can be drawn to separate a set of training input *x<sub>i</sub>*. To decide the optimal solution and unique hyperplane, some optimizations are needed. As seen in Figure 8, the distance between two hyperplanes $w.x - b = 1$ and $w.x - b = -1$ is $\frac{2}{|w|}$. As such, we want to maximize this distance by simply minimizing the value of $|w|$ or mathematically conveniently minimizing $\frac{1}{2}\left||w|\right|^2$ [13] when at the same time ensuring:

$$\min_{w,b} \frac{1}{2}\left||w|\right|^2 \qquad (3.1.3)$$

subject to:

$$w.x_i - b \geq 1 \;\; or \;\; w.x_i - b \leq -1 \qquad (3.1.4)$$

$$or \;\; y_i(w.x_i + b) \geq 1 \qquad (3.1.5)$$

This optimization problem can be solved by the saddle point of the Lagrange's function:

$$L_p = L_{w,b,a} = \frac{1}{2}\left||w|\right|^2 - \sum_{i=1}^{m} \lambda_i [y_i(w.x_i + b) - 1] \quad (3.1.6)$$

To solve the extrema problem, we take the partial differentiation of *L* with respect to *w* and *b*. First for *w*, we have

$$\frac{d}{d_w} L_{w,b,a} = w - \sum_{i=1}^{m} \lambda_i y_i x_i = 0 \qquad (3.1.7)$$

which gives us:

$$w = \sum_{i=1}^{m} \lambda_i y_i x_i \qquad (3.1.8)$$

Then, taking the partial differentiation with respect to $b$ leads to:

$$\frac{d}{d_b} L_{w,b,a} = \sum_{i=1}^{m} \lambda_i y_i = 0 \qquad (3.1.9)$$

Plugging (3.1.8) and (3.1.9) in (3.1.6) gives us the extrema solution:

$$max_\lambda \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j x_i^T x_j \qquad (3.1.10)$$

where:

$$0 \le \lambda_i \le C \; for \; i = 1,2, \dots, m. \qquad (3.1.11)$$

Note that choosing the value of the penalty factor $C$ is important as stated in [14]. If it is too large, we have a high penalty for no separable points and we may store many support vectors to cause the overfitting problem. If the penalty factor $C$ is too small, we may have the under fitting problem.

When the optimized $\lambda$ is solved, we can get the decision function as follows:

$$f(x) = sgn(\sum_{i=1}^{m} \lambda_i y_i x_i^T x + b) \qquad (3.1.12)$$

Using (3.1.12), we can map any value of *x* into either 1 or -1.

**c) Kernel functions and non-linear classification**

For some complicated problem, the training data cannot be simply classified by a linear SVM classifier. In order to solve such problems, the input $x$ can be mapped into higher dimensional space by the kernel function $\Phi$. Then, by SVM we will find how to select a linear hyperplane with the maximal margin in this new dimensional space. There are many popular kernal functions. For example:

$$Linear\ Kernel: K(x_i, x_j) = x_i^T x_j \qquad (3.1.13)$$

$$Polynomial\ Kernel: K(x_i, x_j) = (\gamma x_i^T x_j + r)^d\ for\ \gamma > 0 \quad (3.1.14)$$

$$RBF\ Kernel: K(x_i, x_j) = \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right), for\ \gamma > 0\ (3.1.15)$$

The RBF Kernel will be used in this thesis for the following main reasons:

1. The samples will be mapped to a higher dimensional space;
2. It is less computationally expensive compared to Polynomial Kernel since less parameters need to be found
3. It is the best Kernel that can deal with the “curse of dimensionality” problem.

In (3.1.10), the Linear Kernel is used. By simply switching this Kernel with (3.1.15), we have:

$$max_\lambda \sum_{i=1}^m \lambda_i - \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m \lambda_i \lambda_j y_i y_j \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right) \qquad (3.1.16)$$

where:

$$0 \le \lambda_i \le C\ for\ i = 1,2, \dots, m. \qquad (3.1.17)$$

To calculate the extrema solution of (3.1.16), first we need to find the optimal parameters $\gamma$ and $C$. It is normally time consuming and computationally expensive to find both optimal parameters. We will be discussed the method for finding them in Section 3.2.

## 3.2 LibSVM library and parameter optimization

### a) LibSVM library

LIBSVM is a library for Support Vector Machines (SVMs). This library has been developed since the year 2000. The main goals are to help users to apply SVM to their applications conveniently. This section will briefly present the usage of LibSVM in solving SVM problems. In this thesis, libsvm version 3.18 will be used [6].

LibSVM provides two most important functions in Matlab environment named as "svmtrain" and "svmpredict". First, we will introduce svmtrain function. A typical way to use this function is shown as follows:

model = svmtrain(y_train, x_train, Parameters)

This function will help us to generate a classification model, given the training data x_train and y_train and parameters, where x_train is a $m \times n$ matrix and y_train is a $m \times 1$ vector with each component corresponding to one row vector of x_train. The parameter has its own form as follows:

Parameters = ['-c ' **C_value** ' -g ' **gamma_value** '-w1 ' **w1_val** '-w2 ' **w2_value** '-b 0' ];

C_value and gamma_value represent the values of the parameters $C$ and $\gamma$ which we choose. (As discuss in 3.1, the RBF Kernel will be used in this project for the SVM classification method.) w1 and w2 are the weight of the classes. Usually, if there are 2 classes and the number of data in each class is the same, w1 and w2 are equal to 0.5. In this thesis, we only consider the case that each class has the same number of training data, i.e. the data set is balanced. As such, the weight values are all equal to each other using the rule $w_i = \frac{1}{n}\ for\ (i = 1, \dots, n)$ where $n$ is the number of all possible values of y or the number of the classes. '-b' is the probability estimate option and the default value is 0, which means we do not output any probability of each class in this thesis. Now, we will introduce svmpredict function. A typical way to use this function is shown as follows:

```
[group_train, acc_train, val_train] = svmpredict(y, x, model,'-b 0' );
```

In this function, “model” has been obtained from svmtrain function. This function is used to calculate which class a row vector in matrix x belongs to. If the true values of the output y are available, this function can also provide the accuracy result by the number of correctly classified rows in matrix x divided by the total number of data.

**b) Optimization of the parameters $C$ and $\gamma$**

As discussed both in Dr Lin’s paper [6] and Qing’s thesis [8], to have a higher classification accuracy, we have to optimize the parameters $C$ and $\gamma$ before we optimize $\lambda$ in the SVM model. It would be computationally expensive if we want to find the optimal $C$ and $\gamma$. To balance the computation amount and higher classification accuracy, we will use grid searching method five times for best $C$ and $\gamma$ in a range. Grid searching starts with a coarse set:

For $C$ : $\{2^{-5}, 2^{-5+h}, \dots., 2^{-5+20h}\}$

For $\gamma$: $\{2^{-10}, 2^{-10+h}, \dots., 2^{-10+15h}\}$

Where $h$ is the step size and it is equal to 1 at the beginning. At each time of search, a five-fold cross validation procedure is used to evaluate a pair of $C$ and $\gamma$. In the cross validation, the total training data are partitioned into five folds. Any four folds out of them are used to train the model while the remaining one fold is used to test the model under the given pair of $C$ and $\gamma$. Therefore, we have five testing accuracies and we take the average of them. To optimize the parameters and reduce computation cost, a bound value of training accuracy is set to be 80%. If the training accuracy of a pair of $C$ and $\gamma$ is less than this value, this pair cannot be best and we just discard this pair. At each time of searching, the best pair of $C$ and $\gamma$ with the highest average testing accuracy is picked out. After one search, the step size $h$ is divided by two and thus a finer grid searching at next time can be made within ten steps around $C$ and $\gamma$ obtained from the previous searching. Finally, the best pair of $C$ and $\gamma$ is found after five searches.

## 3.3 Feature as input of SVM for EEG-based BCI system

In our developed EEG-based BCI system, the feature or the input directly going into the SVM model is not time-domain EEG signals. Instead, we convert these signals into power spectrum density (PSD) in the frequency domain as the feature for classification. The typical algorithm used for this conversion from time domain to frequency domain is fast Fourier transform (FFT).

Actually, FFT is an algorithm to calculate the discrete Fourier transform (DFT) [16]. It can reduce the computations order from $2N^2$ to $2Nlog2(N)$, where $N$ is the number of computations needed.

The function below shows the DFT:

$$X(k) = \sum_{n=1}^{N} x[n]e^{-j\omega_k n} \qquad (3.3.1)$$

while

$$\omega_k = \frac{2\pi}{N}k \qquad (3.3.2)$$

PSD represents the power of the time-domain signal in different frequencies. The equation to compute the PSD of a discrete time signal is:

$$S_{xx}(k) = \frac{(\Delta t)^2}{T} \left| \sum_{n=1}^{N} x[n] e^{-j\omega_k n} \right|^2$$

$$= \frac{\Delta t}{N} \left| \sum_{n=1}^{N} x[n] e^{-j\omega_k n} \right|^2$$

$$= \frac{1}{F_s * N} \left| \sum_{n=1}^{N} x[n] e^{-j\omega_k n} \right|^2$$

$$= \frac{1}{F_s * N} |X(k)|^2 \tag{3.3.3}$$

where $x[n] = x(n * \Delta t), T = N * \Delta t, F_s = \Delta t$

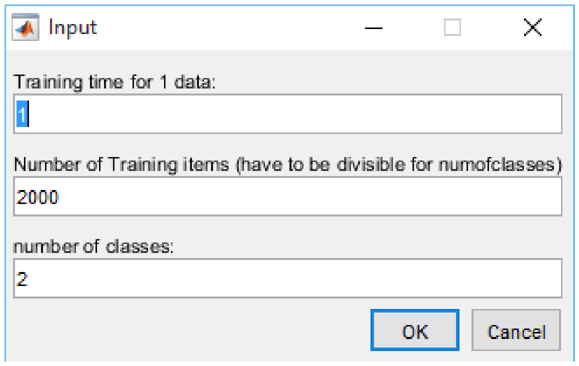
Equation (3.3.3) will be used in Matlab coding in Section 4.2 for FFT computation. The rectangular window is used in the above FFT computation. However, to avoid the power leakage in calculation, the hamming window will be used instead in our research, which has the function:

$$h(n) = 0.54 - 0.46 \cos\left(2\pi \frac{n}{N}\right) \tag{3.3.4}$$

# Chapter 4 Useful Self-developed Programs for EEG-based BCI System

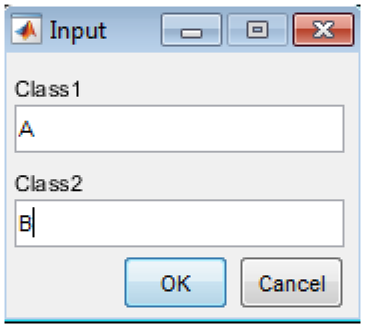
## 4.1 EEG time domain data collection program

In our research, a program is created for automatically collecting high-quality EEG time-domain data from EPOC headset to PC with the format compatible with Matlab. First, the raw EEG data is taken from EPOC headset. After the EPOC headset is connected to the EPOC Simulink Signal Sever. A Matlab program "TakingData.m" (see Appendix A) is run to start the data collecting process. Depending on where this program locates, it can automatically change the working directory into the directory of the folder that contains it. Firstly, the user will be asked to input some general information about the data type. The information includes: the number of training items, the number of data type, and the time in second for each data type. The code will then call the Simulink model as shown in Figure 7 and run the Simulink model for the same training time input from the user as seen in Figure 9. The training time for one data will be one second in the example in this figure. Depending on the number of classes, the user will be asked to input different names for classes (e.g. left, right) as seen in Figure 10. If the folders with the same names of these classes' names do not exist in the working directory, new folders will be created. The program will then ask the user to prepare to take the data in the random manner. For example, if there are two classes of the letters A and B, the program will ask the user to think for A and B randomly and pop up a message dialog as seen in Figure 11. After the user has thought about a class for a specific time, the program will double check the user's feeling about the data quality by using the question dialog as seen in Figure 12. If the answer is 'Yes', the program will collect the specific data and put it in a specific folder. However, if the answer is 'No', the program will ask the user to retake the specific data by calling the message box shown in Figure 11. The data corresponding to the folder name will be named in a numerical order (e.g. 'Time_dataLeft0' if the class name is 'Left'). By counting the number of files in each folder every time, the program will keep track of the numerical order. Even when the user closes the program to take a rest, the next time he or she wants to take new data, the program will resume the numerical order of the new data's name. This will avoid overlapping which leads to losing data. An example of the data taken for the letters A and B in their working directories is shown in Figure 13.



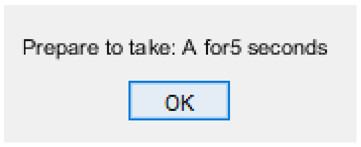
Training time for 1 data:
1
Number of Training items (have to be divisible for numofclasses)
2000
number of classes:
2

**Figure 9 The program asks the user to input general information**



Class1
A
Class2
B

**Figure 10 The program asks the user to input classes' names**



Prepare to take: A for5 seconds

**Figure 11 The program asks the user to use the EPOC and think for the specific class**

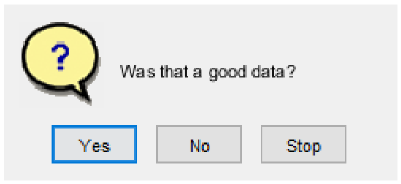
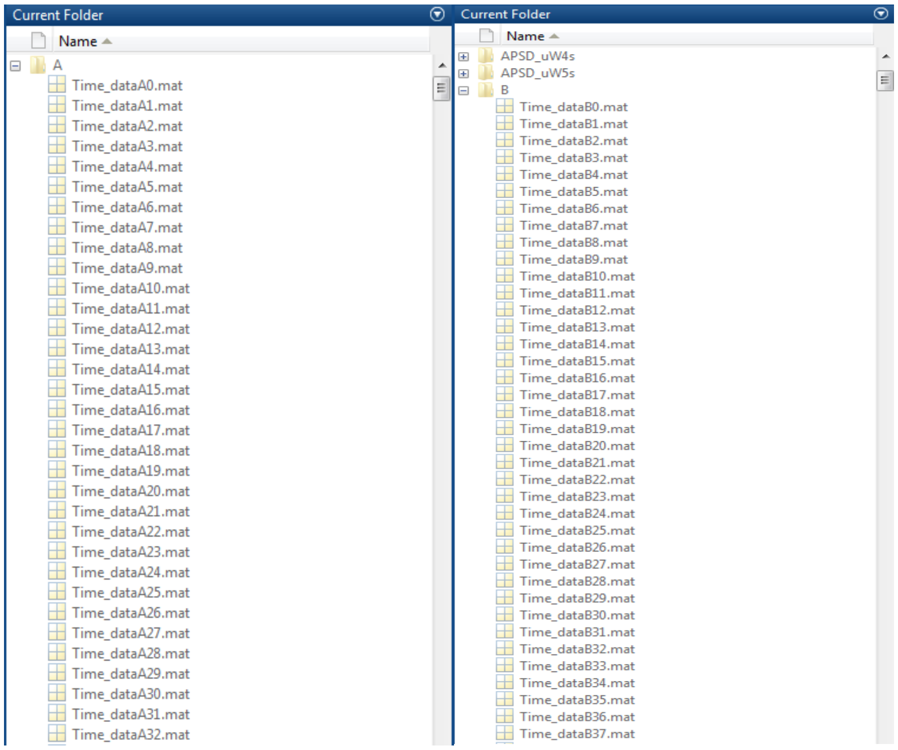


**Figure 12 The program double checks the user's feeling about the data quality**



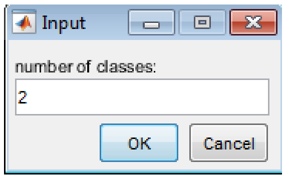
```
A
  Time_dataA0.mat
  Time_dataA1.mat
  Time_dataA2.mat
  Time_dataA3.mat
  Time_dataA4.mat
  Time_dataA5.mat
  Time_dataA6.mat
  Time_dataA7.mat
  Time_dataA8.mat
  Time_dataA9.mat
  Time_dataA10.mat
  Time_dataA11.mat
  Time_dataA12.mat
  Time_dataA13.mat
  Time_dataA14.mat
  Time_dataA15.mat
  Time_dataA16.mat
  Time_dataA17.mat
  Time_dataA18.mat
  Time_dataA19.mat
  Time_dataA20.mat
  Time_dataA21.mat
  Time_dataA22.mat
  Time_dataA23.mat
  Time_dataA24.mat
  Time_dataA25.mat
  Time_dataA26.mat
  Time_dataA27.mat
  Time_dataA28.mat
  Time_dataA29.mat
  Time_dataA30.mat
  Time_dataA31.mat
  Time_dataA32.mat
```

```
APSD_uW4s
APSD_uW5s
B
  Time_dataB0.mat
  Time_dataB1.mat
  Time_dataB2.mat
  Time_dataB3.mat
  Time_dataB4.mat
  Time_dataB5.mat
  Time_dataB6.mat
  Time_dataB7.mat
  Time_dataB8.mat
  Time_dataB9.mat
  Time_dataB10.mat
  Time_dataB11.mat
  Time_dataB12.mat
  Time_dataB13.mat
  Time_dataB14.mat
  Time_dataB15.mat
  Time_dataB16.mat
  Time_dataB17.mat
  Time_dataB18.mat
  Time_dataB19.mat
  Time_dataB20.mat
  Time_dataB21.mat
  Time_dataB22.mat
  Time_dataB23.mat
  Time_dataB24.mat
  Time_dataB25.mat
  Time_dataB26.mat
  Time_dataB27.mat
  Time_dataB28.mat
  Time_dataB29.mat
  Time_dataB30.mat
  Time_dataB31.mat
  Time_dataB32.mat
  Time_dataB33.mat
  Time_dataB34.mat
  Time_dataB35.mat
  Time_dataB36.mat
  Time_dataB37.mat
```

**Figure 13 Time-domain EEG data in the working directory**

## 4.2 Conversion program for EEG data from time domain to frequency domain

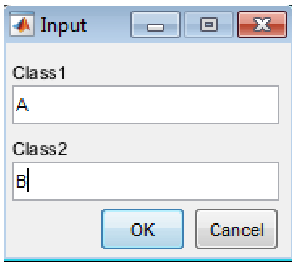
As discussed in Chapter 3, in this thesis, the EEG data will be converted into PSD format. To do this, the FFT function as shown in section 3.3 is used. In order to make the brain data converting job easier, the Matlab program “CovertData.m” (see Appendix A) is developed. The purpose of the code is converting all of the time domain data that is previously saved in the work directory to PSDs in the frequency domain. First of all, the user needs to place this Matlab code in the same directory where the time domain data locates. When the code is running, an input dialog will pop up and ask the user to input the number of classes as shown in Figure 14. Then, the user need to input the name of the folders where the time data are contained as shown in Figure 15. Here, “A” and “B” will be input here as an example. Note that “A” and “B” represent two folders that contain the time-domain EEG data for the letters A and B and locate in the same working directory. If these folders are not in the working directory, the program will show errors and close. Then another dialog will pop out and explain the user what to do next. At this point, just hit ‘OK’ to proceed. Another input dialog will appear as shown in Figure 16. Here, if the user changes the value of the input from 0 to 1, it means the user wants to convert the time-domain data into that specific data type. There are two type choices for data conversion available: “PSD data in Db” and “PSD data in uW” where the PSD data in Db is basically 10log10 of PSD data in uW. This input dialog serves for the same purpose as the check box, which is not included in Matlab basic dialog’s models. For this project, since all data are 5 seconds long, this program will serve for these data only. After the type of the converted data is chosen, the program will start running. Firstly, it will create the new directories as shown in Figure 17. For example, if “A” and “B” are the name of the folders containing time-domain data and the user wants to convert it into PSD data in uW, there will be 10 new directories created APSD_uW1s-APSD_uW5s and BPSD_uW1s-BPSD_uW5s. These are where the PSD data from 1s-5s for each time-domain data locates. This will be done by looping 5 times to create five different hamming windows and chopping down different information from one piece of time data. To be more specific, the 1s PSD data is converted from 0-1s time-domain data, the 2s PSD data is converted from 0-2s time-domain data until 5s PSD data is converted from 0-5s time-domain EEG data. Before converting data, the program will keep track of how many data are in each directory already. This will help the program to avoid calculating existed PSD data and thus saving a lot of time. The process of converting data normally takes about 15 minutes for 1000 time-domain data trials.



number of classes:

2

**Figure 14 Inserting the number of classes for conversion**



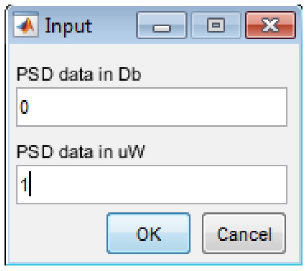
Class1

A

Class2

B

**Figure 15 Inserting the name of the time-domain data folders**



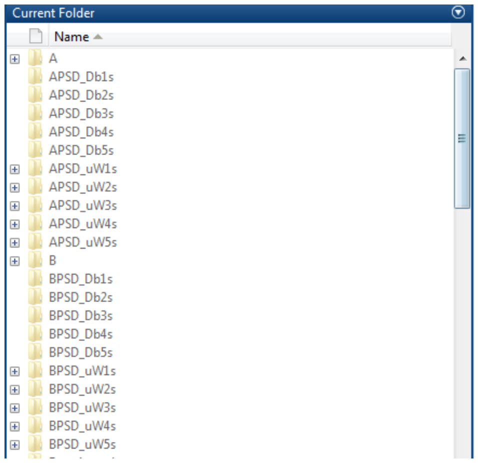
PSD data in Db

0

PSD data in uW

1

**Figure 16 Choosing the type of data to be converted into**



```
A
APSD_Db1s
APSD_Db2s
APSD_Db3s
APSD_Db4s
APSD_Db5s
APSD_uW1s
APSD_uW2s
APSD_uW3s
APSD_uW4s
APSD_uW5s
B
BPSD_Db1s
BPSD_Db2s
BPSD_Db3s
BPSD_Db4s
BPSD_Db5s
BPSD_uW1s
BPSD_uW2s
BPSD_uW3s
BPSD_uW4s
BPSD_uW5s
```

**Figure** 17 **New directories are created**

## 4.3 Program for training and creating SVM model

In this project, the most critical program is the one which can solve multi-class classification problems. We will present such a program in this section. This program is named "svm_train_n_categories.m" which is shown in Appendix A. First, since in our project, each class has the same amount of data, i.e. the data set is balanced, our program is developed to be capable of doing multiple-classes classification for this case. The program has two main functions. The first function is to train the SVM model by training data and output the training result in a text file. The second function is to create the well trained SVM model and save it into .mat format. This well-trained model can be used later to control the robot car. When the user runs this program, the dialog below will pop up. The first choice "seeding analysis" is for SVM training, while the second choice "Making predict model" is used to generate the well-trained SVM model. Next, we will describe these two choices

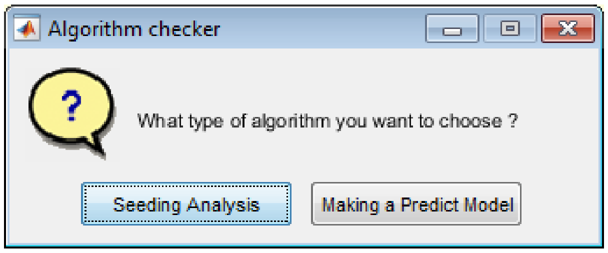
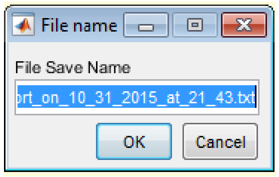


**Figure 18 The program asks the user to make choice between 2 options**

### a) Seeding analysis

If the user chooses "Seeding Analysis" from the dialog, the new dialog will pop up, asking for the name of the text file (Figure 19). This text file is located in the same working directory where the program is located and the result of the SVM training is stored. By default, the text file will be automatically generated based on the date and time of the simulation; the users can change the name if they wish to. When "Ok" button is hit, the text file will be generated in the working directory, the program will then ask for some information before the training process is run.

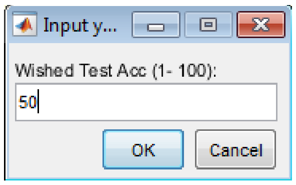

File name

File Save Name

ort_on_10_31_2015_at_21_43.txt

**Figure 19 The name of the text file**

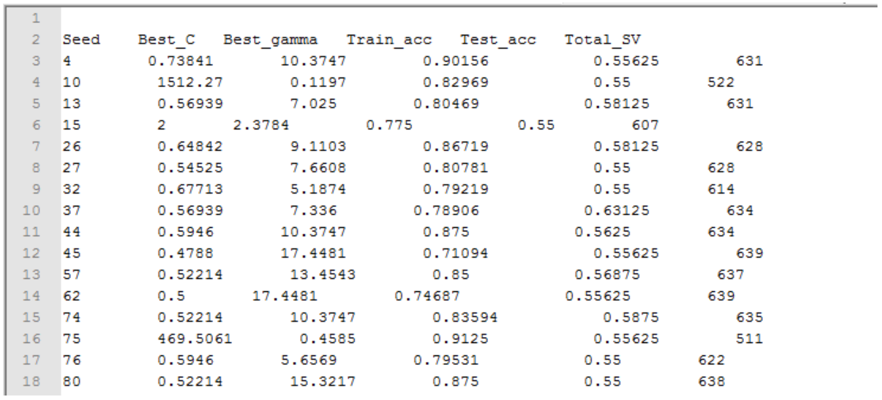
First, the program will ask for the testing accuracy bound that the user wants to see (Figure 20). If the testing accuracy of a pair of $C$ and $\gamma$ pair is lower than this bound, it will not show up in the text file (Figure 21). The output text file showed in Figure 21 has 6 columns. From right to left are the values of seed, best C parameter, best gamma parameter, training accuracy, testing accuracy, and finally the number of support vectors. In this sample output, only a few seeds which have the testing accuracy greater than 50% will be showed. Any results less than or equal to 50% will be ignored.



Wished Test Acc (1- 100):

50

**Figure 20 Insert the lowest bound for the output**

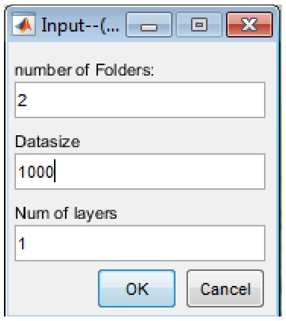


```

Seed     Best_C     Best_gamma    Train_acc    Test_acc    Total_SV
4          0.73841        10.3747        0.90156            0.55625         631
10          1512.27        0.1197        0.82969            0.55         522
13          0.56939        7.025        0.80469            0.58125         631
15          2        2.3784        0.775            0.55         607
26          0.64842        9.1103        0.86719            0.58125         628
27          0.54525        7.6608        0.80781            0.55         628
32          0.67713        5.1874        0.79219            0.55         614
37          0.56939        7.336        0.78906            0.63125         634
44          0.5946        10.3747        0.875            0.5625         634
45          0.4788        17.4481        0.71094            0.55625         639
57          0.52214        13.4543        0.85            0.56875         637
62          0.5        17.4481        0.74687            0.55625         639
74          0.52214        10.3747        0.83594            0.5875         635
75          469.5061        0.4585        0.9125            0.55625         511
76          0.5946        5.6569        0.79531            0.55         622
80          0.52214        15.3217        0.875            0.55         638
```

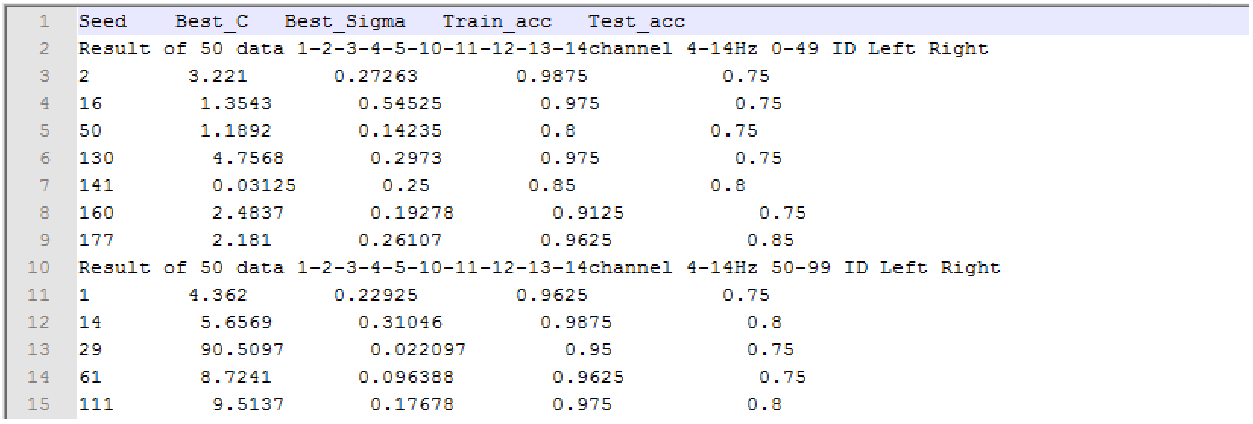
**Figure 21 The sample of an output file**

Then, the program will ask for the numbers of classes for training. The user needs to input the information as follow: number of folders (classes), data size (numbers of data in each folder) and number of layers (Figure 22). By default, the number of layers will be 1 which means that all of the data will be used for training. When the number of layers is changed, the data set will be divided into different groups. For example, if the data size is 1000, and the number of layers is 10, then for each layers there will be 100 data only. These 10 layers will be used to train SVM model separately, so that the user can compare the quality of data in each layer. This is important because in reality, it is hard to know if the brain data taken is good for the training or not. The user only knows the quality of his or her data after the training is done, which might take a lot of time. For example, it might take at least 4 hours to train the SVM model by the Left versus Right data, which are high quality data. If the quality of the data is low, it might take about 1 day of running to have one result. By running the SVM algorithm layers by layers, the user can have a better feeling about the data and remove a section of bad-quality data if necessary. Figure 23 shows the output file if the user chooses the number of layers different from 1. As seen in Figure 23, the first and second layers both have similar testing accuracy of 75-85% which means they are good data. Normally, the running time to get this output file is much shorter (1-2 hours) compared to the running time for the full data size (1000 data in this case).



number of Folders:
2
Datasize
1000
Num of layers
1

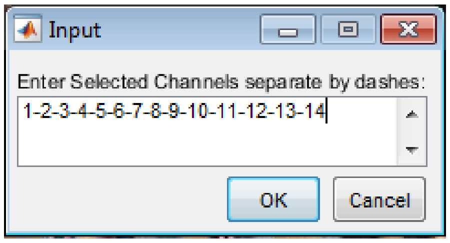
**Figure 22 Input number of folders, data size and number of layers**



```
Seed     Best_C    Best_Sigma    Train_acc    Test_acc
Result of 50 data 1-2-3-4-5-10-11-12-13-14channel 4-14Hz 0-49 ID Left Right
2         3.221       0.27263        0.9875          0.75
16         1.3543       0.54525        0.975          0.75
50         1.1892       0.14235        0.8          0.75
130         4.7568       0.2973        0.975          0.75
141         0.03125       0.25        0.85          0.8
160         2.4837       0.19278        0.9125          0.75
177         2.181       0.26107        0.9625          0.85
Result of 50 data 1-2-3-4-5-10-11-12-13-14channel 4-14Hz 50-99 ID Left Right
1         4.362       0.22925        0.9625          0.75
14         5.6569       0.31046        0.9875          0.8
29         90.5097       0.022097        0.95          0.75
61         8.7241       0.096388        0.9625          0.75
111         9.5137       0.17678        0.975          0.8
```

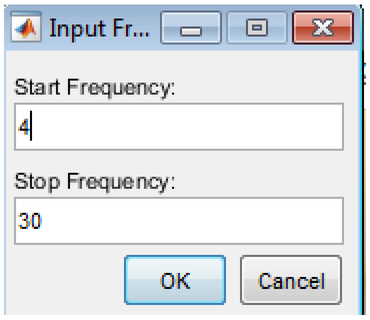
**Figure 23 the layers by layers output**

After the user hits “Ok” to proceed, the program will pop up the same dialog as seen in Figure 15. The user will need to input the name for the training data set which is PSD format data in our project. The program will then ask for the channel choices (Figure 24). Numbering from 1 to 14 are 14 physical channels on the EPOC headset. In the processes of data being taken and being converted, the raw EEG data from EPOC is used as input. As such, the information for all 14 channels is still contained in the PSD data but in frequency domain. To choose different channels for the training data set, the user need to place the dash “-“ between the number, the program will convert this string to the integers and take out the selected channels for the SVM training.  The inputs have to be between 1 and 14 only; if the user input is out of range, the program will report error and ask for the channel selection again. Finally, the program will then ask for the frequency range. In this project, since the EPOC with 128 sample rate is used, only 64 frequencies from 0 to 64 Hz will be analyzed. The dialog in Figure 25 will ask the user to input the start and stop frequencies to form the training data set. The inputs have to be between 0 and 64 Hz, if they are out of bound, the program will again report errors and ask for new inputs. Both of the options for channels and frequency range give the user flexibility in research. The user can compare testing accuracies o select the optimal frequency range and channels. Hitting “Ok” to proceed, the program will create the training data set and start training. The training process is shown in the block diagram below (Figure 26).


Enter Selected Channels separate by dashes:

1-2-3-4-5-6-7-8-9-10-11-12-13-14

**Figure 24 Input the channel range**



Start Frequency:
4

Stop Frequency:
30

**Figure 25 Input frequency range**

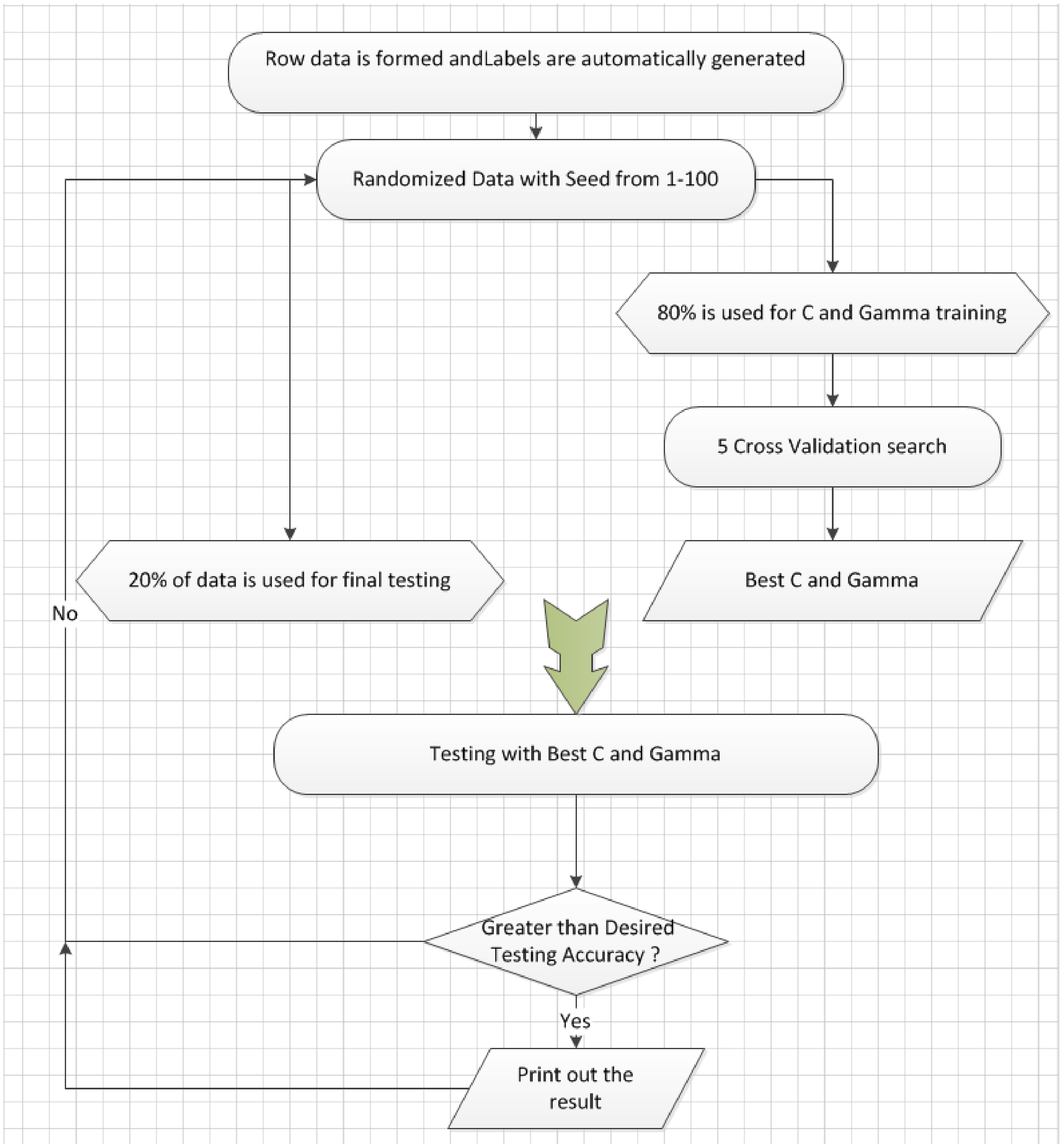

Row data is formed andLabels are automatically generated
Randomized Data with Seed from 1-100
80% is used for C and Gamma training
5 Cross Validation search
Best C and Gamma
20% of data is used for final testing
No
Testing with Best C and Gamma
Greater than Desired Testing Accuracy ?
Yes
Print out the result

**Figure 26 SVM training process**

The PSD data is saved in the form of a $m \times n$ matrix where the rows $m$ is the number of the channels and the columns $n$ is the number of the discrete frequencies from 0 to 64Hz. As discussed in section 3.3, each input in the training data has to be in a row vector. This vector will be formed as a $1 \times (m \times n)$ row vector by placing each rows of the matrix next to each other.

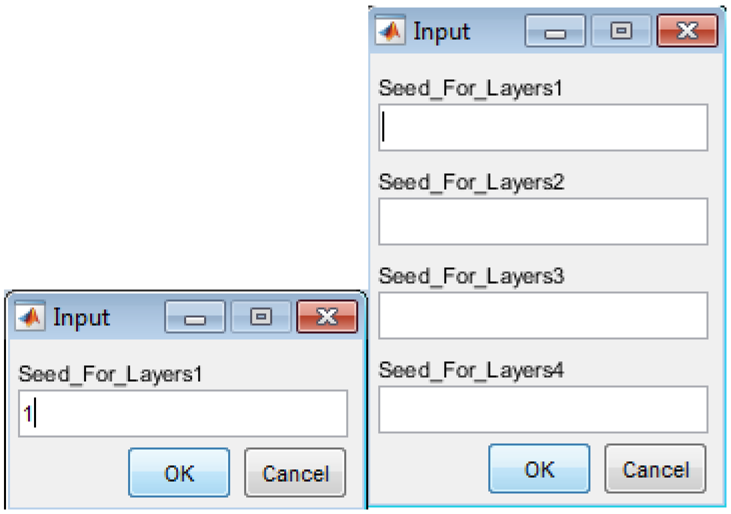
Each of this data row will correspond to a label. This label is automatically generated depending on the number of classes. The labels will depend on the order of the inputs, the first input is always mapped to 1 then the label will increase by 1 for the next classes. For example, as seen in Figure 15, class A will be mapped to label 1 and class B will be mapped to label 2. If there are C, D, and E, they will be mapped to labels 3, 4, and 5, respectively. The data will be then randomized using seeds' value from 1 to 100 (which can be changed in the code). The reason why the seeds are used is as follow: assume we have a data set of 1000 data, where only 80% of this data (800) is chosen for SVM training. To calculate all the possible cases to form this 800 data out of the total 1000 data set, we will use the combination equation:

$$\binom{1000}{800} = \frac{1000!}{800! * 200!}$$

where both 1000! and 800! are too big to calculate in Matlab, and the result could also be a large number. As a result, it is practically impossible to check all these cases and we need to randomize the data many times to get a best result out of all trials. The use of seeds in Matlab will help us to control the randomization and as such can reform the training data set for a given seed. The C and gamma searching will be done as discuss in Section 3.2. One problem here is the function "svmtrain" requires a string as its parameters. This string is written differently depending on the number of classes. To do this, normally we need to change this single line of code manually and this will reduces the flexibility of the program. The solution for this is to let the program write these lines of code itself. These special lines of the code are written in three other Matlab file named as writeFPara.m, writePara.m and writeweight.m. These files will be changed many times in the training process depending on the number of class, C value and gamma value. After the best C and Gamma parameters are found, the program will use these parameters for the final testing data set (20% of the data). If the testing accuracy is greater than the wished testing accuracy (Figure 20), this result will be printed to the text file. The program will proceed to the next seed and redo all of the above steps. The training will be finished after all the seeds are used. Practically, for some type of data, the time for training 1 seed might be hours or days. As such, the user need to control this manually by changing MySeed parameters in the code (Appendix A).

**b) Making predict model**

If the user chooses “Making Predict Model” from the dialog in Figure 18, the same dialog as seen in Figure 24 will pop out. The program will then ask for the folders where the data locate (Figure 15). Next, the program will ask for the seed used for each layer. If the layer value input is 1, only 1 seed is needed. However if layer value is a number *n*, more than 1, the user will be asked to input *n* seeds (Figure 27). This seed(s) is used to reconstruct the training data set as discuss in the previous section.  If this is a multilayer analysis, the best data in each layer will be taken out to form the training set. This is an extra implemented function but not necessarily used for this research.


Input

Seed_For_Layers1: 1

OK Cancel

Input

Seed_For_Layers1

Seed_For_Layers2

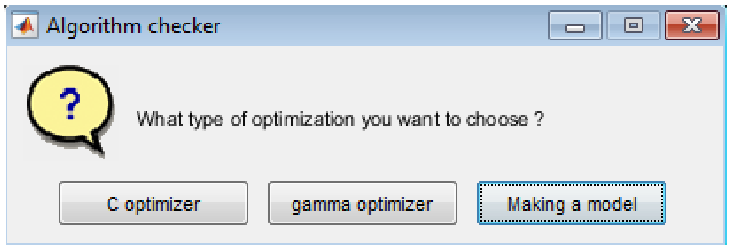
Seed_For_Layers3

Seed_For_Layers4

OK Cancel

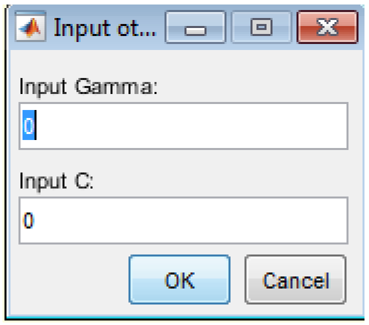
**Figure 27 Layer=1 on the left dialog, Layer =4 on the right dialog**

Then, the same dialogs in Figures 24 and 25 will show up. The user will input the same information that he or she uses for the seeding analysis case to reform the training data structure. It will take 1 to 2 minutes for the data reconstruction, depending on the data size. The program will then show three possible options (Figure 28). The first option C optimizer is used when the user knows the best value of gamma and search for C only. Conversely, the gamma optimizer is used when the user knows the best result for C and search for gamma. Again, these 2 options are built in this program but not used for this research. The last option “Making a model” will then ask the user to input the values for both best C and gamma parameters (Figure 29). These are the values shown in column 2 and 3 of the text file output (Figure 21). Hitting “Ok” to proceed, the program will start to recreate the well-trained SVM model which have the same training and testing accuracy shown in the text file. It would take several seconds to one minute for the model to be recreated. Finally, the program will ask the user to save the model in a desired directory (Figure 30). The default directory will be the Parameters folder located in the same working directory.


Algorithm checker

What type of optimization you want to choose ?

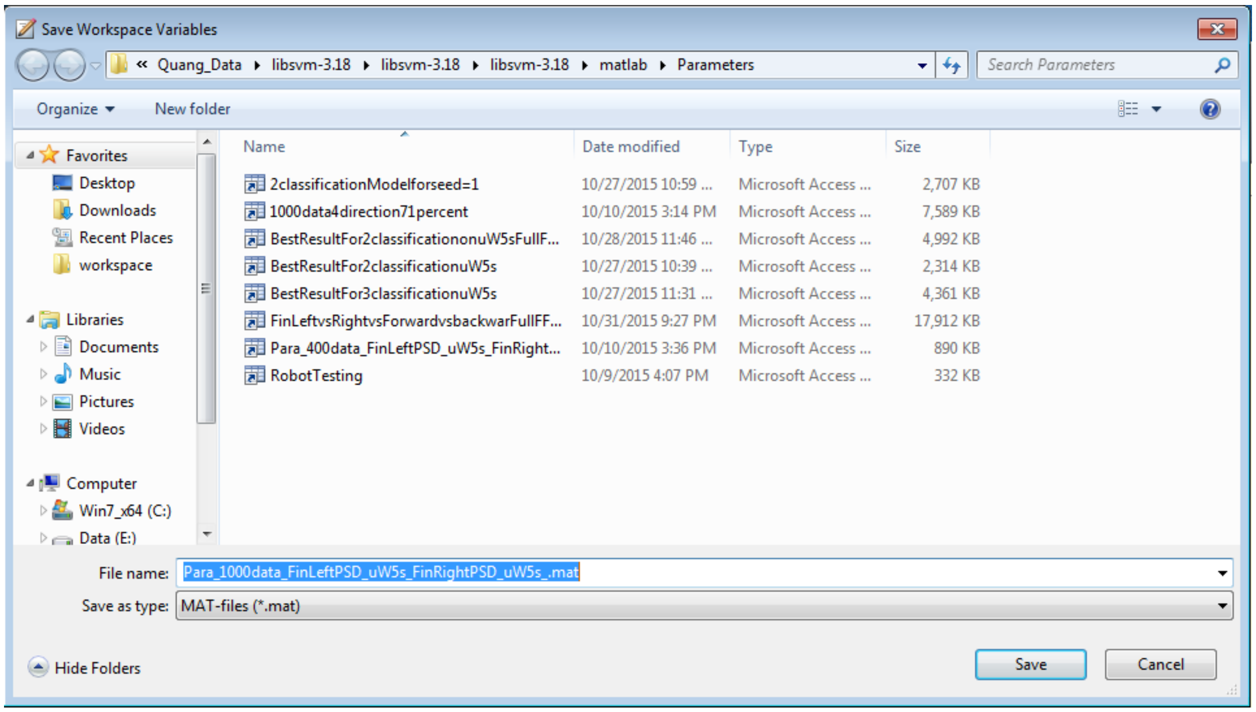
**Figure 28 Choosing between 3 options**


Input ot...

Input Gamma:
0

Input C:
0

**Figure 29 Inputting best C and gamma**


Save Workspace Variables

« Quang_Data ▸ libsvm-3.18 ▸ libsvm-3.18 ▸ libsvm-3.18 ▸ matlab ▸ Parameters

| Name | Date modified | Type | Size |
|---|---|---|---|
| 2classificationModelforseed=1 | 10/27/2015 10:59 ... | Microsoft Access ... | 2,707 KB |
| 1000data4direction71percent | 10/10/2015 3:14 PM | Microsoft Access ... | 7,589 KB |
| BestResultFor2classificationonuW5sFullF... | 10/28/2015 11:46 ... | Microsoft Access ... | 4,992 KB |
| BestResultFor2classificationuW5s | 10/27/2015 10:39 ... | Microsoft Access ... | 2,314 KB |
| BestResultFor3classificationuW5s | 10/27/2015 11:31 ... | Microsoft Access ... | 4,361 KB |
| FinLeftvsRightvsForwardvsbackwarFullFF... | 10/31/2015 9:27 PM | Microsoft Access ... | 17,912 KB |
| Para_400data_FinLeftPSD_uW5s_FinRight... | 10/10/2015 3:36 PM | Microsoft Access ... | 890 KB |
| RobotTesting | 10/9/2015 4:07 PM | Microsoft Access ... | 332 KB |

File name: Para_1000data_FinLeftPSD_uW5s_FinRightPSD_uW5s_.mat

Save as type: MAT-files (*.mat)

**Figure 30 Saving the final model to a location**

## 4.4 Programs for controlling robot

In this section, we will show the program “RobotControl.m” (see Appendix A) for establishing the interface between the computer and the robot. To do this, we need to use some of Matlab packages dealing with serial communication as well as Arduino control. Unlike other data types, creating a real time system for EEG is hard because there is no feedback between the user and the system. For example, in [16], the authors presented a method using libsvm for real time facial emotion detection. It has the similar purpose of our project which is to create a real time system for EEG signal detection. However, for facial recognition, the users can have the feedbacks (real-time webcam images) from the system and at the same time the results from the classifications. Thus, the users can have a way to control their facial emotion to get the good classification result. This is not the case for the EEG signal. Since there is no way to monitor the brain states at the same time controlling the robot, this model is just an open loop controller with no feedbacks. As such, it is very challenging to control the robot to move in a desired direction or control LCD to display a desired letter. Although the average classification accuracy is relatively high, there is mismatching between the individual performances of each class. For example, the robot always prefers to get to the direction with the highest individual performance. To solve this, we have two methods. The first method is to ask the user to firstly think “nothing” for about 20s before running the robot. Using this information as a reference, if the mean power of a thought is less than the mean of this reference value, the robot will stop. This will eliminate some of the misclassification for some data that is not related to directional or alphabetical categories and thus increase the number of correct guesses. The second method is using the correlation coefficient to eliminate bad EEG data. Depending on the modes chosen by the user, the program will calculate the correlation coefficient before or after the output from the model. The program will have two modes: practice mode and real time running mode (Figure 31).

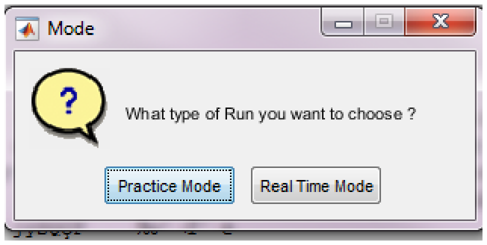
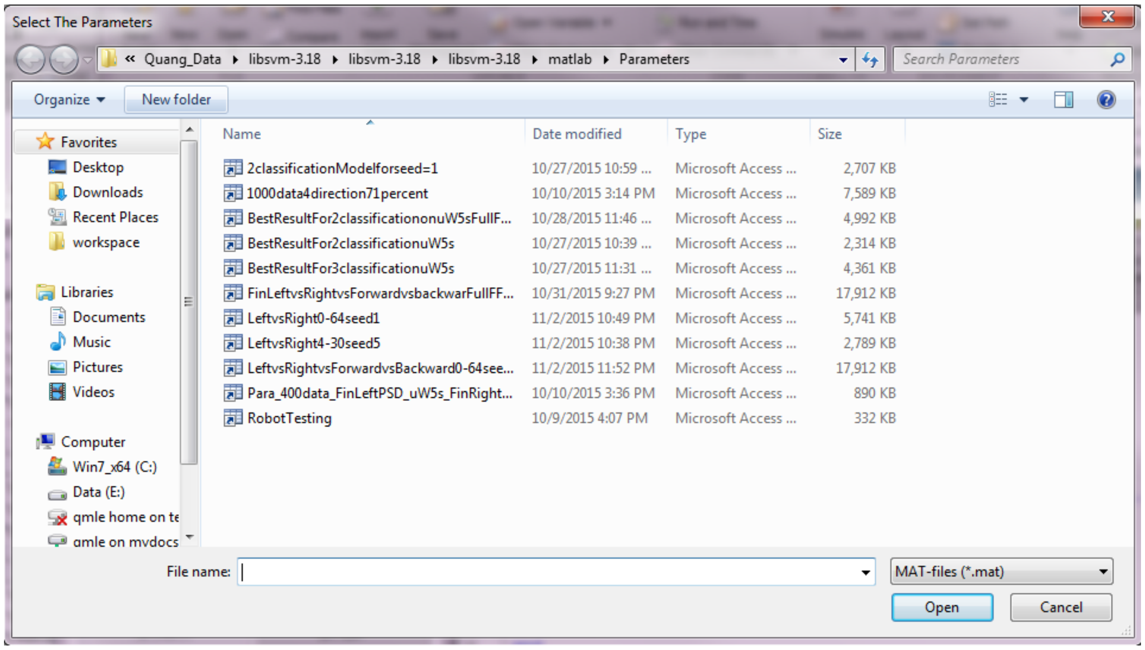


**Figure 31 Choosing the mode for the Program to run**

### a) Practice mode

In this mode, the program will help the user to get familiar with the SVM model. Firstly, the program will automatically open the Epoc control panel and EEG Simulink importer which are normally located in the Program Files of the computer. After the user put the head set on and make sure all of the sensors are in good condition (all green), the program will then ask the user to choose the well trained SVM model which is saved in the Parameters folder (figure 32). This is the same folder mentioned in Figure 30 above. The input dialog (Figure 33) will pop up, asking for general information before the training begins. The first three inputs have to be the same as the inputs while the model is created. For example, a model guessing 2 directions with 1000 data of 5 seconds long each is shown in the picture below. The points of input represents the least number of trials the program will run for each class. After the user hits “OK”, a dialog as seen in Figure 16 will appear, asking the user to provide the names of the PSD data folder. The reason why we need to take this information for the real time is because in the training process, the libsvm function requires all of the data to be normalized in the range between -1 and 1. Therefore, a single input by its own has no meaning to the real time detection. To solve this problem, we have to recall the training folders to calculate the normalized data. For example, if there is a total of N data in the training set, the new data will be then added into the group. The normalization of N+1 data is calculated and only the result for the real time data will be picked out. In fact, this calculation will reduce the response time of the system greatly since the normalized data will be recalculated every time the new data is input. To solve this problem, we only calculate the normalized data for N data and save the important values. These values include the array with the minimum value and the range (max-min). The new normalization of the single data will be calculated as followes:

$$realtime\_nomalized = \frac{realtime - \min}{\max - \min} \qquad (4.4.1)$$

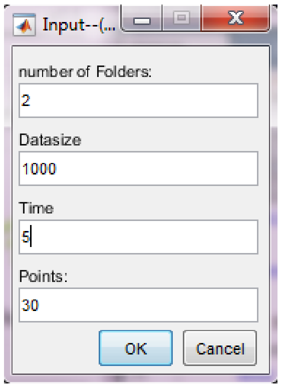


| Name | Date modified | Type | Size |
|---|---|---|---|
| 2classificationModelforseed=1 | 10/27/2015 10:59 ... | Microsoft Access ... | 2,707 KB |
| 1000data4direction71percent | 10/10/2015 3:14 PM | Microsoft Access ... | 7,589 KB |
| BestResultFor2classificationonuW5sFullF... | 10/28/2015 11:46 ... | Microsoft Access ... | 4,992 KB |
| BestResultFor2classificationuW5s | 10/27/2015 10:39 ... | Microsoft Access ... | 2,314 KB |
| BestResultFor3classificationuW5s | 10/27/2015 11:31 ... | Microsoft Access ... | 4,361 KB |
| FinLeftvsRightvsForwardvsbackwarFullFF... | 10/31/2015 9:27 PM | Microsoft Access ... | 17,912 KB |
| LeftvsRight0-64seed1 | 11/2/2015 10:49 PM | Microsoft Access ... | 5,741 KB |
| LeftvsRight4-30seed5 | 11/2/2015 10:38 PM | Microsoft Access ... | 2,789 KB |
| LeftvsRightvsForwardvsBackward0-64see... | 11/2/2015 11:52 PM | Microsoft Access ... | 17,912 KB |
| Para_400data_FinLeftPSD_uW5s_FinRight... | 10/10/2015 3:36 PM | Microsoft Access ... | 890 KB |
| RobotTesting | 10/9/2015 4:07 PM | Microsoft Access ... | 332 KB |

File name:

MAT-files (*.mat)

**Figure 32 Choosing a model**



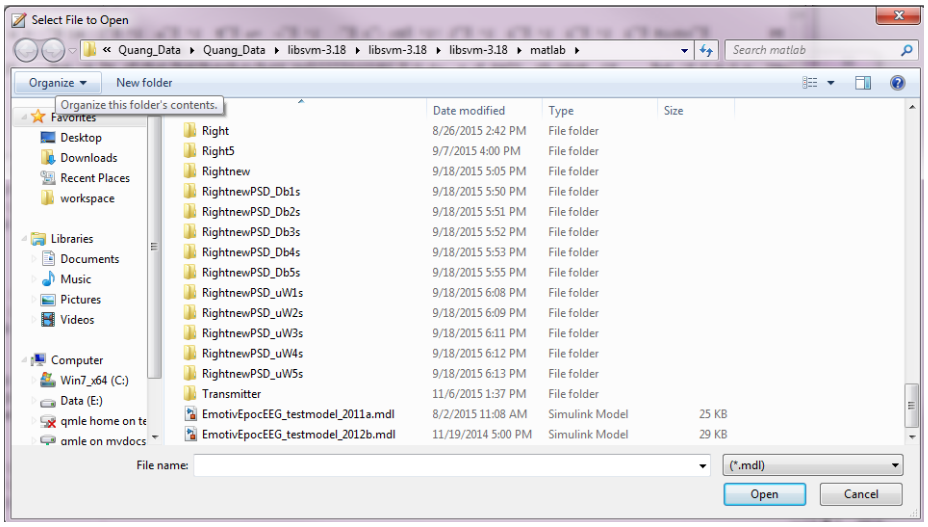
number of Folders: 2

Datasize: 1000

Time: 5

Points: 30

Figure 33 Information input for model-practice mode

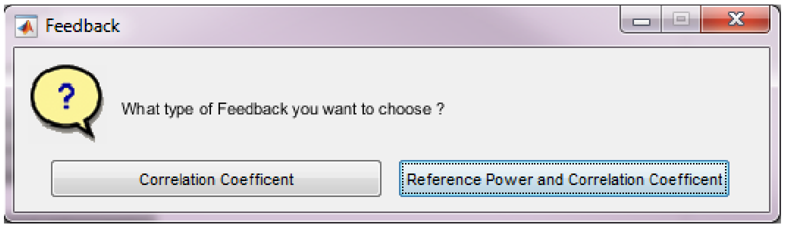


| | Date modified | Type | Size |
|---|---|---|---|
| Right | 8/26/2015 2:42 PM | File folder | |
| Right5 | 9/7/2015 4:00 PM | File folder | |
| Rightnew | 9/18/2015 5:05 PM | File folder | |
| RightnewPSD_Db1s | 9/18/2015 5:50 PM | File folder | |
| RightnewPSD_Db2s | 9/18/2015 5:51 PM | File folder | |
| RightnewPSD_Db3s | 9/18/2015 5:52 PM | File folder | |
| RightnewPSD_Db4s | 9/18/2015 5:53 PM | File folder | |
| RightnewPSD_Db5s | 9/18/2015 5:55 PM | File folder | |
| RightnewPSD_uW1s | 9/18/2015 6:08 PM | File folder | |
| RightnewPSD_uW2s | 9/18/2015 6:09 PM | File folder | |
| RightnewPSD_uW3s | 9/18/2015 6:11 PM | File folder | |
| RightnewPSD_uW4s | 9/18/2015 6:12 PM | File folder | |
| RightnewPSD_uW5s | 9/18/2015 6:13 PM | File folder | |
| Transmitter | 11/6/2015 1:37 PM | File folder | |
| EmotivEpocEEG_testmodel_2011a.mdl | 8/2/2015 11:08 AM | Simulink Model | 25 KB |
| EmotivEpocEEG_testmodel_2012b.mdl | 11/19/2014 5:00 PM | Simulink Model | 29 KB |

File name: (*.mdl)

Figure 34 choosing the simulink model

After the data folders are chosen, the user will set up the data size identical to the training data of the model. This includes choosing the frequency range (Figure 25) and channel range (Figure 24). After the information is taken, the user has to choose the Simulink model to take the data. This will be varied depending on the Matlab version in use. As seen in Figure 34 "EmotivEpocEEG_testmodel2011a.mdl" will be used. Then the program will ask the user to choose the feedback type (Figure 35). There are 2 options available in which the first will calculate the correlation coefficient only and the second one will calculate both the correlation coefficient and the reference power.


Feedback

What type of Feedback you want to choose ?

**Figure 35 input feedback type**

If the reference power is chosen, the program will ask the user to clear his or her mind for 20s and take this data (Figure 36). After the reference data is taken, it will be converted into PSD and the mean value of this PSD is also calculated. Then, the program will ask the user to take the reference data for the correlation coefficient calculation for each class. For example, if there are 2 classes left and right, 2 references will be asked to take. In taking the reference data, the program will use the model and ask the user to think a specific thought for 5s (Figure 37). If the detection result from the SVM model is correct, the program will use this as the reference for that class; if not, the program will ask the user to try again until one EEG trial with the correct detection result is obtained (Figure 38).

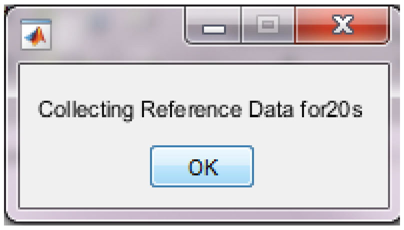
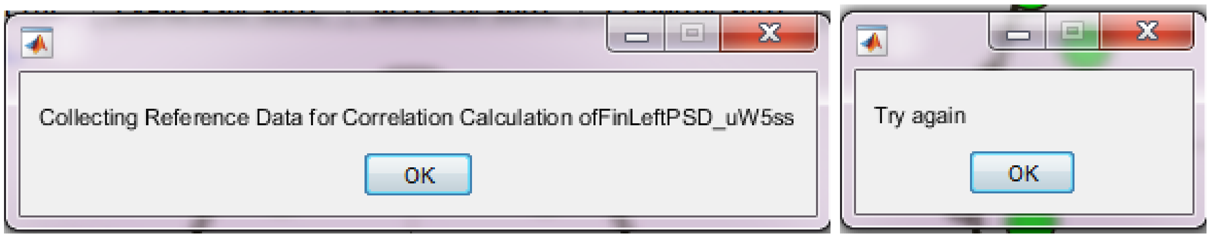

Collecting Reference Data for20s

**Figure 36 collecting the "no thinking" reference**


Collecting Reference Data for Correlation Calculation ofFinLeftPSD_uW5ss

Try again

**Figure 37 Colleting correlation coefficient reference data (left)-Retake data reference (right)**

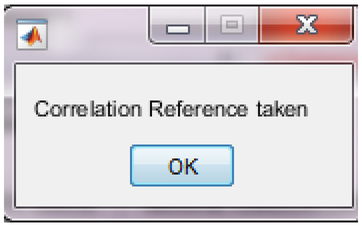

Correlation Reference taken

**Figure 38 Informing the user that the EEG data with correct detection result is taken**

The program will inform the user to begin the practice (Figure 49). Using the correlation references taken before, the program will calculate the correlation coefficient. If the result is less than 85% or the mean power of the input data is less than the reference mean (thinking nothing), the program will ask the user to retake the data (Figure 41). If both of the tests are passed, the model will classify the data into correct or incorrect answer (Figure 41). If it is the same with the test case (left or right), one point will be added to the total points. If not, the program will keep running.

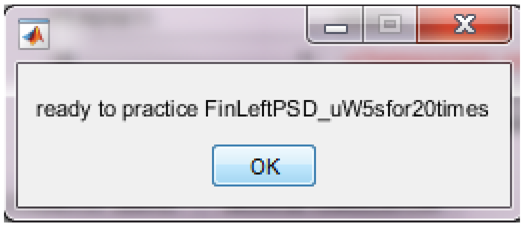

ready to practice FinLeftPSD_uW5sfor20times

**Figure 39 Start training for one class**

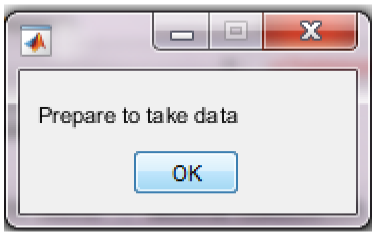
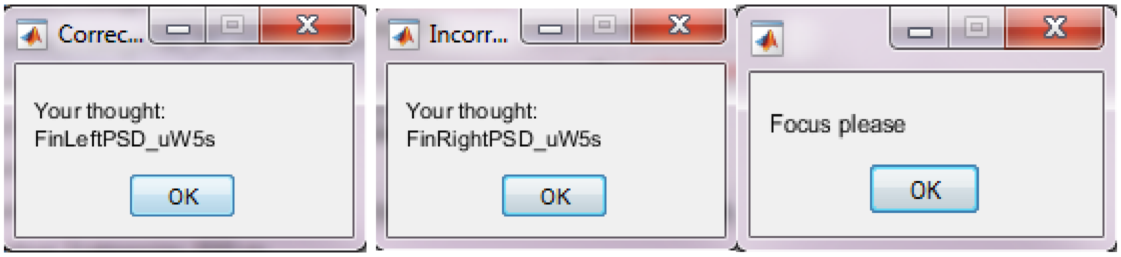

Prepare to take data

**Figure 40 Start taking a piece of data**


Correc...

Your thought:
FinLeftPSD_uW5s

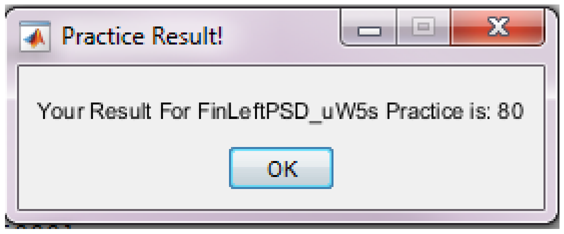
Incorr...

Your thought:
FinRightPSD_uW5s

Focus please

**Figure 41 Three types of possible answer from program-Correct (left)-Incorrect (middle)-Retake data (right)**

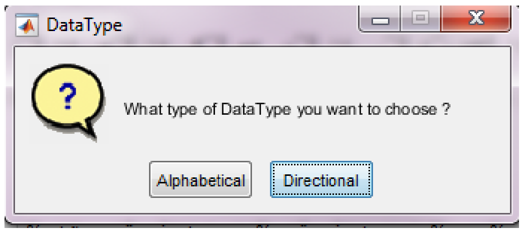
The dialogs in Figures 40 and 41 will keep showing up until the user gets all the points they indicate in the inputs. The total number of trials will be tracked so that the final accuracy can be calculated (Figure 42). In this mode, the results will not be illustrated on the robot car yet. When the user feels confident enough with the result, he or she can rerun the program and choose real time mode where the motion of the robot car will be controlled.


Practice Result!

Your Result For FinLeftPSD_uW5s Practice is: 80

**Figure 42 Final result for practice modeb) Real time mode**

In this mode, before the data taking begins, the user has to set up the BCI. The transmitter circuit (Arduino board) will be first connected to the computer, and the robot car power source will also be turned on. Then the steps from Figures 34 to 39 will be the same in the real time mode. After the references are taken, the user has to indicate the data type he or she wants to send to the robot. This is important because by default the directional SVM model will classify input PSD data into 4 integers (1-4) and the alphabetical SVM model will classify input PSD data into 2 integers (1-2). The answer the user has in this question dialog (Figure 44) will help the program to distinguish between the output data of alphabetical or directional type. The program will start taking the brainwave data and informing the user with the dialog in Figure 40. The EEG data taken is converted to PSD and then sent to the well-trained SVM model to do the classification. Then, depending on the classification output, the program will compare the classification result with a corresponding correlation reference. If the result is greater than 85%, this is a correct result and will be sent to the robot. To send the signal to the robot, the program has to set up the serial communication between Matlab and Arduino board. Its code is shown in “run.m” in Appendix A. The user has to manually change the ComPort variable in order to have the program function correctly. To figure out the right Comport, the user has to open “Device Manager” in Window and check for the usb connection tab. This will indicate which Comport the Transmitter Arduino is connected to. After the connection is set up correctly, Matlab’s command window will now be in serial mode with the Arduino board. Anything printed on this command window will be sent to the Arduino board. We will have 6 different signals in character type using the first letter of each classs. These signals are: ‘l’- Left, ‘r’ – Right, ‘f’-Forward, ‘b’-Backward, ‘A’- letter A and ‘B’-letter B. Depending on the classification result, these characters will be printed on the command window to send the signal to the board. The transmitter board will then send this information through Xbee to the robot. After each transmission, the serial connection will be closed between Matlab and Arduino, giving back the computational ability for Matlab.



DataType

What type of DataType you want to choose ?

**Figure 43 Indicating the type of signal sent to robot**

# Chapter 5 Analysis of Raw EEG Data and Feature as SVM Inputs

## 5.1 EEG data recording rule

In order to have the classifiable data set, the user has to follow a strictly rule for recording every single data. The thesis writer was the subject for the experiment in this project. The tricky part about taking EEG data in this research is the subject has to mimic the condition of an ALS patient. He has to limit his body motions and try to stay still.  For each of directional EEG data taking, the subject will imagine visually there is an object on the corresponding direction and also think about the meaning of the direction verbally in his language. The subject has to close his eyes to reduce the eye movement artefacts. For the alphabetical EEG data, the subject will also think about the letter visually in mind at the same time thinking how to pronounce the letter. To increase the quality of both data types, after taking every 100 data, the subject will take a rest and let his mind relax. In total, there will be 6000 5-seconds EEG data taken for 6 different classes. In estimation, it would take at least 4 hours to complete 1000 data taking which leads to the total of 24 hours for 6 classes.

## 5.2 EEG data recording method

To control the robot using BCI, normally, the user may think about any possible directions at any time to get the robot to a desired destination. As such, when the data recording program (Chapter 4) was developed, it was implemented so that the user can take the EEG data both randomly and continuously. If the user decides to take all time-domain data in different classes at the same time, the program will create a random list of labels and ask the user to think differently each time. The user can also decide to take the data continuously by choosing to take data for one class each time. At first, we thought that taking data randomly was a better approach since it can gives the user the same feeling as if they are controlling the robot. To test our idea with this approach, 100 5-second EEG data of 2 directions (Left and Right) were taken. The data was then converted to PSD format and going through the SVM training and testing procedure. The best testing accuracy was 70% with 92% of the training accuracy. This was an acceptable result for a small data set so we decided to run the training process with more data. 1000 5-second EEG data of 2 directions (Left and Right) were used for SVM training and testing. The best result however was only 55% for testing and around 90% for training. The libsvm used 1280 support vectors out of 1280 training vectors to create the SVM model, which means overfitting (also known as curse of dimensionality) has happened. This is normally because the data size (the dimension of each training vector) is too big, and the libsvm classifier will try to find the optimal model exhaustively which then leads to the overfitting problem. To solve this, we try to reduce the data size by reducing the channel range. We picked only 8 channels for the last training and see if the classification results could improve. However, the result ended up to be only 60% for testing and around 91% for training. The number of support vectors for this training is about 1000 which was still a high number. At this point, we decided to restart from data recording process, 100 5-second EEG data of Left and Right directions were taken continuously. After the first training process, both of the testing accuracy and training accuracy were above 90% which are very good. We continued to take 1000 5-seconds-data of each class in the same manner. For the first 8 seeds, the results are shown in Table 2 in Chapter 6. The best result showed that the test accuracy can be as high as 83 %. The number of support vectors corresponding to the best result is only 912 out of 1280 training data. This is a very good result which shows the model are well trained and good models for the real time testing. The same methods are used for 3 and 4 directional as well as 2 alphabetic classifications. The results for these classifications will be shown in Section 6.2 to 6.5 of Chapter 6.

## 5.3 Feature size for SVM input

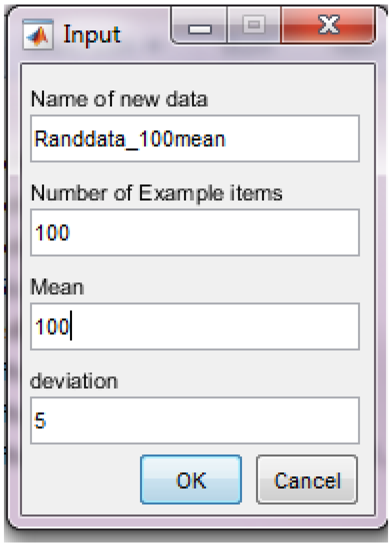
The feature as the input of the SVM model in our research is PSD in the frequency domain. The size of this feature is decided by the selected frequencies and channels. The Matlab programs introduced in Chapter 4 give us the flexibility in choosing the frequency range as well as channel range for the PSD data. The EPOC headset used in this project has the sample rate of 128 samples per second. As such, after being converted into the frequency domain, there are 64 different frequencies ranging from 0-64 Hz. This range can be cut down into 5 different bands: Delta band (1-3 Hz), Theta band (4-7 Hz), Alpha band (8-12 Hz), Beta band (13-30 Hz), and Gamma band (30-100 Hz) [17]. Some researchers only chose specific frequency bands to do the classification. For example in “EEG-based discrimination between imagination of right and left hand movement” [12], the authors found out that different frequency components in alpha (8-12 Hz) and beta bands (13-30Hz) provided best discrimination between left and right hand movement imagination. The classification results can reach to approximately 80 %. Another paper has also pointed out the EEG activity on theta band (4-7 Hz) in virtual maze navigation [18]. This research shows the important role of theta oscillators in human spatial navigation. Considering the alphabetical and language analysis, in [19] a broad analysis of different EEG frequency band can show different approaches to the language analysis. Specifically, Delta (1-4 Hz), Theta (4-7 Hz) and Alpha (8-12 Hz) band shows a respectful effect on auditory and visual of language perceptions. The nouns or verbs are then compared in two bands Alpha (8-12Hz) and Beta (13-30 Hz) to see their performances. This result is not very useful in this case since only alphabetical (A or B) data is analyzed in this paper. However, these papers pointed out a good frequency band (from Theta to Beta) and can help to lower the PSD data size concerned in this research. We decided to run 2 different trainings. In the first training, we only used frequency range from 4-30Hz and in the second we use all the possible frequency range from 0-64Hz.We tried to run the classification using all frequencies (0-64 Hz). This in fact increased the testing results for approximately 10% in each classification. However, the big drawback was the time for training SVM model has increased to as low as 30 hours to as high as 96 hours for each classification (depending on data type and the number of class). As such, only 1 seed for each classification will be shown in the experimental result in Section 6.6.

# Chapter 6 Experimental Results of SVM Models

In this chapter, we will summarize the results of the SVM models for different classification problems. First we will give the result of a very simple classification problem to verify the correctness of the SVM program since it is the core of our developed BCI system. And then, we will show the results of classifying directional and alphabetic EEG signals. Finally, we will present some classification results using the whole frequency range.

## 6.1 Classification of data from normal distributions

Since the SVM program "svm_train_n_catefories.m" is extremely critical in our developed BCI system, before using the SVM program to classify the real EEG data, we created a case to check if the program runs correctly. To do this, we create a $14 \times 64$ matrix and fill the matrix with random normal distribution data. We control the mean and standard deviation of the data so that the data will be classifiable. For each mean value of 1 to 4, 100 of matrices are created using the example_brain_data.m in Appendix A. This small program will let the user create the normal distribution randomized data. The user can control the standard deviation and mean values by modifying the inputs showed in Figure 44. These data will be numerated from 1 to 100 and pushed into folders with the same names (1, 2, 3, 4). Using the "svm_train_n_catefories.m", these data in the 4 folders will be classified. If the training accuracy and testing accuracy of the classification result is high, it means that the program created is capable of classifying data correctly. Table 1 below shows the classification results for the 4 different normal distribution randomized data. The results are close to 100% which proves that this program is working correctly.


Name of new data

Randdata_100mean

Number of Example items

100

Mean

100

deviation

5

**Figure 44 Input used in ExampleBrainDataMaker.m for training**

**Table 1 Classification result of data from normal distribution**

| **Seed** | **Best_C** | **Best_gamma** | **Train_acc** | **Test_acc** | **Total_nSV** |
|---|---|---|---|---|---|
| 1 | 0.03125 | 0.00097656 | 97.188 | 100 | 320 |
| 2 | 29.3441 | 0.00097656 | 100 | 100 | 316 |
| 3 | 0.03125 | 0.016317 | 98.125 | 100 | 320 |
| 4 | 19.8697 | 0.00097656 | 100 | 96.25 | 320 |
| 5 | 25.7678 | 0.00097656 | 100 | 92.5 | 317 |
| 6 | 1.4142 | 0.038808 | 100 | 92.5 | 320 |
| 7 | 0.03125 | 0.0040792 | 98.125 | 96.25 | 320 |
| 8 | 0.03125 | 0.0009756 | 97.188 | 100 | 320 |

## 6.2 Classification of 2 directional data

The classification results for 2 directions (left and right) are shown in Table 2. The best one appears at seed number 5 where the training accuracy is 98.188% and the testing accuracy is 83%. Using the training method in Figure 26, there were 1280 data (200 left, 200 right) used as the training data set and 400 data used for the final testing, where the best model only used 912 data out of 1280 training samples as support vectors. For this 2 directional classification, the training for each seed took at least 6 hours. The training time varied depending on the number of support vectors needed. In general, the classification with the least number of support vectors will take less time, but this does not always guarantee the optimal solutions. The program is capable of running more seeds. However, only the results of the first 8 seeds are obtained since it takes a lot of time and computational cost (running several Matlab windows at same time). Table 2 gives the overall testing accuracy at the fifth column. It is the number of EEG data classified correctly divided by the total number of testing data. Table 3 shows the testing accuracy of individual direction classification. For example, the second column labeled as Left is the number of left direction data classified correctly divided by the total number of left direction data. For most of the seeds, the individual testing accuracies are not balanced. The SVM model may detect one direction more successfully than the other. Seed number 3 shows the most balanced performance with the gap of only 0.5% between Left versus Right classification. However, there is a tradeoff of 0.75% in overall testing accuracy compared to the imbalanced result of seed 5.

**Table 2 Left vs Right Classification Result with Frequency range 4-30Hz**

| Seed | Best_C | Best_gamma | Train_acc | Test_acc | Total_nSV |
|---|---|---|---|---|---|
| 1 | 29.3441 | 0.088388 | 98.313 | 82.75 | 855 |
| 2 | 449.6006 | 0.010132 | 94.937 | 79.25 | 675 |
| 3 | 6.1688 | 0.13053 | 95.75 | 82.25 | 964 |
| 4 | 6.7272 | 0.125 | 96.437 | 79.5 | 934 |
| 5 | 14.6721 | 0.11463 | 98.188 | 83 | 912 |
| 6 | 122.5732 | 0.02116 | 94.688 | 82.75 | 744 |
| 7 | 18.2206 | 0.125 | 98.938 | 80.5 | 919 |
| 8 | 14.6721 | 0.1197 | 98.1880.0 | 82.5 | 912 |

**Table 3 Individual performances of 2 directions Left vs Right 4-30Hz**

| Seed | Left | Right | Test_accuracy |
|---|---|---|---|
| 1 | 84.5 | 81 | 82.75 |
| 2 | 80.5 | 78 | 79.25 |
| 3 | 82.5 | 82 | 82.25 |
| 4 | 75 | 84 | 79.5 |
| 5 | 81.5 | 84.5 | 83 |
| 6 | 84 | 81.5 | 82.75 |
| 7 | 79 | 82 | 80.5 |
| 8 | 80.5 | 84.5 | 82.5 |

## 6.3 Classification of 3 directional data

The classification results for 3 directions (left, right, and forward) are shown in Table 4. The best result in testing accuracy is 79.333 % for seed number 8 with the training accuracy of 95.208%. According to the training method (Figure 26), there were 600 data in the testing pool (200 left, 200 right, and 200 forward) and 1920 data in the training pool. The best model used only 1410 data out of 1920 data as support vectors. The training time for each seed was between 10 hours and 12 hours. So it took nearly 4 days for completing the calculation for all seeds. Table 5 then shows the individual testing accuracies. This table shows an even more imbalanced result compared to the 2 directional individual performances. The gap between the best and the worst individual performances in each seed varied from 4.5% (seed 4) to 12% (seed 3). The most balanced seed is seed number 4. However, there is a tradeoff of 5% overall accuracy compared to the imbalance result of seed 8.

**Table 4 Left vs Right vs Forward Classification Result with Frequency range 4-30Hz**

| Seed | Best_C | Best_gamma | Train_acc | Test_acc | Total_nSV |
|---|---|---|---|---|---|
| 1 | 10.834 | 0.25 | 98.375 | 77.333 | 1644 |
| 2 | 5.6569 | 0.17678 | 93.042 | 75.833 | 1602 |
| 3 | 18.2206 | 0.20131 | 98.542 | 76.5 | 1576 |
| 4 | 9.5137 | 0.17678 | 95.583 | 74.333 | 1555 |
| 5 | 7.336 | 0.2394 | 96.583 | 78 | 1649 |
| 6 | 69.7925 | 0.065267 | 96.333 | 79 | 1368 |
| 7 | 66.8335 | 0.081052 | 97.5 | 75.167 | 1369 |
| 8 | 32 | 0.084641 | 95.208 | 79.333 | 1410 |

**Table 5 Individual performances of 3 directions Left vs Right vs Forward 4-30Hz**

| Seed | Left | Right | Forward | Test_acc |
|---|---|---|---|---|
| 1 | 80 | 73.5 | 78.5 | 77.333 |
| 2 | 79 | 70.5 | 78 | 75.833 |
| 3 | 79 | 67 | 83.5 | 76.5 |
| 4 | 72.5 | 73.5 | 77 | 74.333 |
| 5 | 77 | 78.5 | 78.5 | 78 |
| 6 | 84 | 71.5 | 81.5 | 79 |
| 7 | 75 | 71 | 79.5 | 75.167 |
| 8 | 75.5 | 78.5 | 84 | 79.333 |

## 6.4 Classification of 4 directional data

Table 6 shows the classification results of 4 directions (left, right, forward, and backward). The best result in testing accuracy is 72 % for seed number 11 with the training accuracy of 92.75%. Seeds 2,4,10 are eliminated from the text file outputs since these seeds have less than 70% of testing result.  According to the training method (Figure 26) there were 800 data in the testing pool (200 left, 200 right, 200 forward, and 200 backward) and 2560 data in the training pool. The best model used only 2089 data out of 2560 data as support vectors. The estimated time for training of each seed varied between 1 to 1.5 days. As such, it took approximately 2 weeks to finish all of these calculations. Table 7 then shows the individual testing accuracy on each directional class. This is the most imbalanced individual classification in the directional classification. The gap between the best and worst individual performances of each seed can be as low as 7.5% for seed 9 and as high as 19.5% for seed 3. The most imbalanced seed is number 1 where the testing accuracy is only 70.875%. There is a tradeoff of 1.125% overall accuracy compared to the imbalance result of seed 11. In fact, during real time robot running, any classification model that has the testing accuracy lower than 80% may not be able to control robot well. As such, for this 4 directional classification problem, we expanded the frequency range to 0-64Hz. These classification results are shown in section 6.6.

**Table 6 Left vs Right vs Forward vs Backward Classification Result with Frequency range 4-30Hz**

| Seed | Best_C | Best_gamma | Train_accuracy | Test_accuracy | Total_SupportVectors |
|---|---|---|---|---|---|
| 1 | 14.6721 | 0.1621 | 94.813 | 70.875 | 2229 |
| 3 | 15.3217 | 0.19278 | 96.219 | 71.125 | 2277 |
| 5 | 6.1688 | 0.19278 | 90.625 | 71.125 | 2319 |
| 6 | 9.5137 | 0.20131 | 94.188 | 71.75 | 2299 |
| 7 | 6.442 | 0.17678 | 90.281 | 71.75 | 2322 |
| 8 | 18.2206 | 0.10511 | 91.031 | 71.125 | 2155 |
| 9 | 107.6347 | 0.048194 | 93.031 | 71.875 | 1976 |
| 11 | 34.8962 | 0.081052 | 92.75 | 72 | 2089 |

**Table 7 Individual performances of 4 directions Left vs Right vs Forward vs Backward 4-30Hz**

| Seed | Left | Right | Forward | Backward | Test_accuracy |
|---|---|---|---|---|---|
| 1 | 73.5 | 64 | 76.5 | 65 | 70.875 |
| 3 | 70.5 | 62 | 81.5 | 70.5 | 71.125 |
| 5 | 69 | 67.5 | 76 | 72 | 71.125 |
| 6 | 72 | 63.5 | 78 | 73.5 | 71.75 |
| 7 | 69 | 64.5 | 78.5 | 75 | 71.75 |
| 8 | 68 | 67 | 80.5 | 69 | 71.125 |
| 9 | 69 | 69 | 76.5 | 73 | 71.875 |
| 11 | 73 | 66 | 79 | 70 | 72 |

## 6.5 Classification of 2 alphabetical data

Table 8 shows the classification result for 2 letters (A and B). The best result appears at seed 1 where the testing accuracy is 78.75% and the corresponding training accuracy is 92.75%. Similar to the 2 directions classification of left versus right, this classification used 400 data in the final testing pool and 1280 data in the training pool. The best result used 834 data out of 1280 data for the support vectors. The training time required for each seed is a little bit longer than that of 2 directions classification. It took approximately 8 hours to complete the training for one seed which led to the total of nearly 4 days for the program to run. Table 9 will then show the individual testing accuracy of each letter's detection. The most balanced result has the difference of 1% at seed 7, while the most imbalanced result has the difference of 9.5% at seed 5. The best model has the gap of 4.5% between A's testing accuracy (81%) and B's testing accuracy (78.75%).

**Table 8 A vs B Classification Result with Frequency range 4-30Hz**

| **Seed** | **Best_C** | **Best_gamma** | **Train_accuracy** | **Test_accuracy** | **Total_SupportVectors** |
|---|---|---|---|---|---|
| 1 | 1961.1715 | 0.0060243 | 92.75 | 78.75 | 834 |
| 2 | 724.0773 | 0.013721 | 95.437 | 73 | 871 |
| 3 | 112.4001 | 0.046151 | 97.938 | 71.25 | 966 |
| 4 | 304.437 | 0.019404 | 93.875 | 75.75 | 917 |
| 5 | 24.6754 | 0.068157 | 93.875 | 73.75 | 1033 |
| 6 | 38.0546 | 0.10066 | 98.75 | 72.25 | 1091 |
| 7 | 939.0121 | 0.017794 | 98.125 | 75.5 | 917 |
| 8 | 122.5732 | 0.017794 | 88.813 | 74.75 | 958 |

**Table 9 Individual performances of 2 letters A vs B 4-30Hz**

| **Seed** | **A** | **B** | **Test_accuracy** |
|---|---|---|---|
| 1 | 81 | 76.5 | 78.75 |
| 2 | 74.5 | 71.5 | 73 |
| 3 | 68.5 | 74 | 71.25 |
| 4 | 76 | 75.5 | 75.75 |
| 5 | 69 | 78.5 | 73.75 |
| 6 | 68.5 | 76 | 72.25 |
| 7 | 75 | 76 | 75.5 |
| 8 | 74 | 75.5 | 74.75 |

## 6.6 Classification of 4 directional and 2 alphabetic data using the maximum frequency range

For the last training process, each classification case from 6.2 to 6.5 will have its frequency range expanded into 0-64Hz. The reason why only the result of the first seed is shown in this section for directional classification and alphabetic classification is that such classification problems need much more training time. For example, it may take approximately 1 day, 2 days, and 4 days for the 2, 3, and 4 directional classifications. For the 2 letter classification it may take about 2 days to complete. Although the computation time was increased, the testing accuracy has been increased approximately 4-11% in each of 4 cases. As shown in Table 10, each of the classification was run only for the first seed. For the directional classifications, the test accuracies for the 2, 3 and 4 directional classifications are 91.5%, 88.1667% and 81.75%

respectively. There is a gain of 8.75%, 10.8337%, and 10.875% of the 2, 3, and 4 directional classifications. Looking at the values of C and gamma parameters, we can see that the program uses very close values of C and gamma for these three cases. This has shown that the features with the expanded frequency range are more classifiable. For the 2 letter classification, the testing accuracy has an increase of 4.5%. The C and gamma parameters for the 2 letter classifications are also smaller compare to the results in the 4-30Hz frequency range. The individual performances in Table 11 show that these models are in fact more balanced than that of 4-30 Hz. The most imbalanced classification is the 4 directional one where the gap between the best and the worst performances is 4.5%. The most balanced result for the directional case is the Left versus Right classification where the gap is only 3%. This also shows that the 2 directional robot motions are easier to be controlled by EEG than 4 directional robot motion in Chapter 7. After expanding the frequency range, all of these classification testing results are greater than 80% which shows good SVM models for the real time testing with the robot.

**Table 10 Classification result of directional and alphabetical data for the first seed from 0 to 64 Hz**

| Classification | Best_C | Best_gamma | Train_acc | Test_acc | Total_nSV |
|---|---|---|---|---|---|
| L/R | 3.668 | 0.10997 | 98.875 | 91.5 | 827 |
| L/R/F | 4.1771 | 0.096388 | 97.0833 | 88.1667 | 1385 |
| L/R/F/B | 3.5125 | 0.21022 | 98.4063 | 81.75 | 2227 |
| A/B | 14.05 | 0.05985 | 98.625 | 83.25 | 998 |

**Table 11 Individual performances of directional and alphabetical data for the first seed from 0 to 64 Hz**

| **Classification** | **Left** | **Right** | **Forward** | **Backward** | **A** | **B** | **Testing Acc** |
|---|---|---|---|---|---|---|---|
| L/R | 93 | 90 | | | | | 91.5 |
| L/R/F | 92 | 86 | 86.5 | | | | 88.1667 |
| L/R/F/B | 79.5 | 82 | 84 | 81.5 | | | 81.75 |
| A/B | | | | | 84 | 82.5 | 83.25 |

# Chapter 7 Robot Control System Implementation and Testing

## 7.1 Block diagram of robot motion and alphabetic display control system

To control the robotic car motion and alphabetic LCD display, a lot of effort is required to communicate between different interfaces simultaneously. As seen in Figure 45, first of all, when a user is thinking about a direction or a letter, 14-channel EEG signals will be acquired by the EPOC device. Through Emotiv control panel and Simulink importer, these EEG signals will be saved with a Matlab format to be used easily in the Matlab SVM program. The parameters in the SVM algorithm will be optimized by  training process. In the SVM training process, training data sets will be needed. Training data sets mean pairs of inputs and outputs of the SVM. EEG signals last for 5 seconds and they will be converted in PSD data in the frequency domain in a desirable frequency range as the feature or input of the SVM model. The output would be a direction or a letter accordingly. To make the SVM well trained, 6000 training data for six classes (4 directions and 2 letters), 1000 data for each class, were generated for this project. As discussed in Chapter 3, in the training process, the cross validation and covex optimization methods will be used to optimize the SVM parameters. With all optimal paramters, the SVM will be well constructed and ready to classify any new EEG signals. This SVM training process and related steps are indicated by blue line in Figure 45.

New EEG signals generated by the user when thinking about a direction or a letter will be send to the constructed SVM model with the optimal parameters through Emotive EPOC, control panel, and Simulink importer as shown by red line in Figure 45. The SVM output is then fed into a control function. This function will convert the SVM output into a digital command such as 000,001,010, 011 and so on to represent each direction and letter. These digital commands will be sent wirelessly to the Adruino microcolltroller using Xbee. If the received digital command means a direction, the Adruino microcontroller will generate a specific wave pulse to control the servo motor to move the robotic car in the desired directions. If the received digital signal means a letter, the microcontroller will conrol the LCD installed in the robotic car to display the letter immediately. All steps related to new EEG signal process for conrolling robot motion and alphabetic display are indicated by red lines in Figure 45. In the previous chapters, we mainly focus on Steps 1-5. In the next sections of this chapter, we will focus on the remaining steps to discuss how to build wireless communication through PC, microcontrolloers, and Xbees in more details.

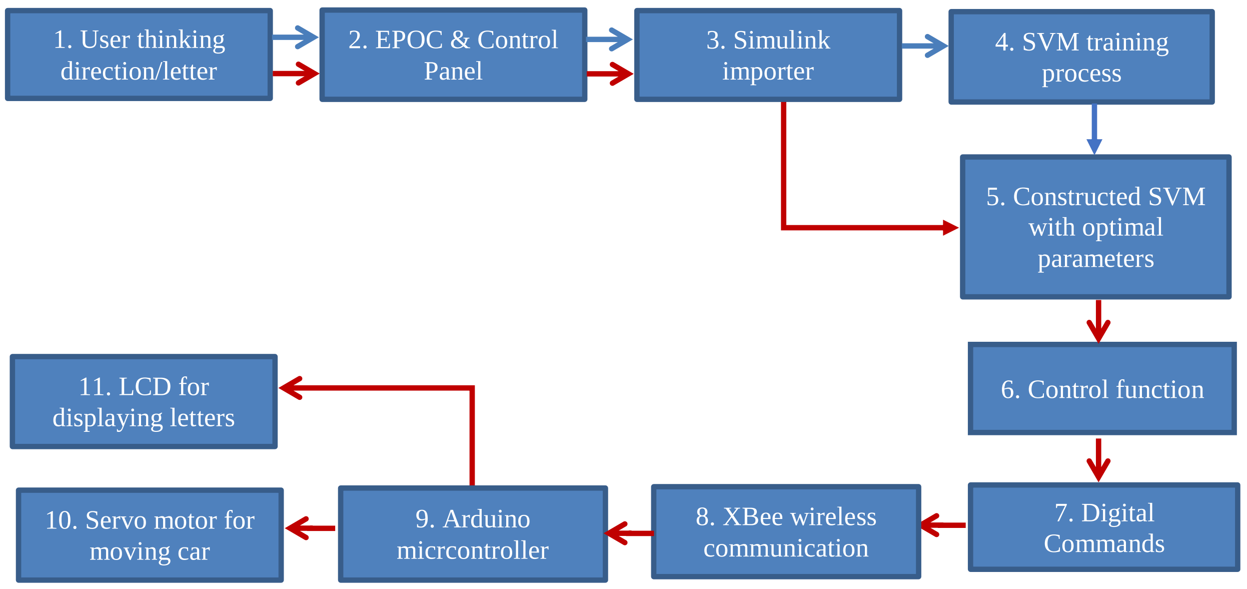
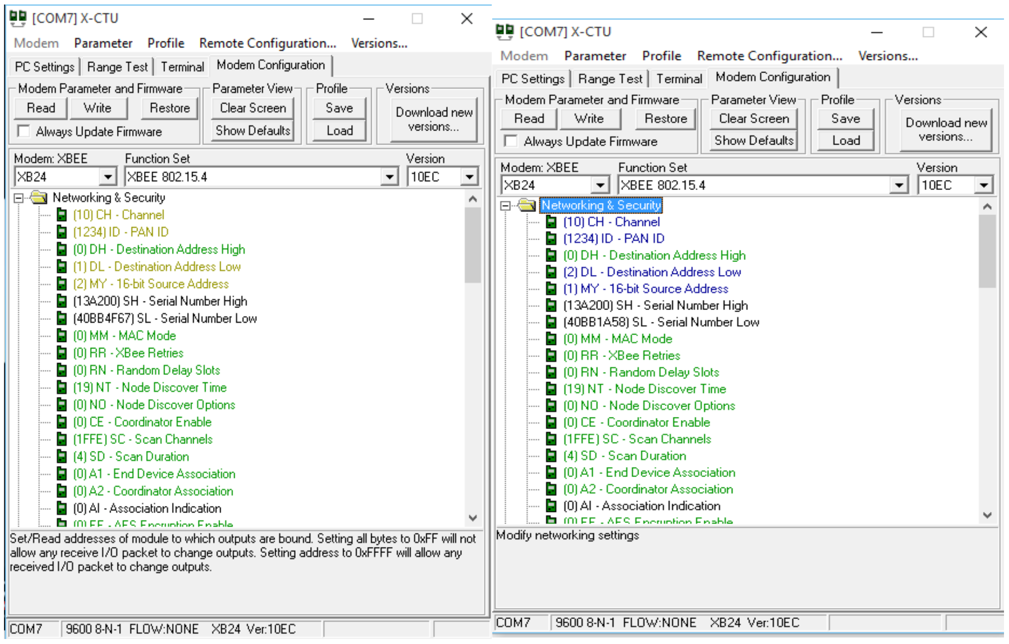

1. User thinking direction/letter
2. EPOC & Control Panel
3. Simulink importer
4. SVM training process
5. Constructed SVM with optimal parameters
6. Control function
7. Digital Commands
8. XBee wireless communication
9. Arduino micrcontroller
10. Servo motor for moving car
11. LCD for displaying letters

**Figure** 45 **Robot motion and alphabetic display control system**

## 7.2 Setting up the Xbee antennas



```
Modem: XBEE   Function Set     Version
XB24          XBEE 802.15.4    10EC
Networking & Security
    (10) CH - Channel
    (1234) ID - PAN ID
    (0) DH - Destination Address High
    (1) DL - Destination Address Low
    (2) MY - 16-bit Source Address
    (13A200) SH - Serial Number High
    (40BB4F67) SL - Serial Number Low
    (0) MM - MAC Mode
    (0) RR - XBee Retries
    (0) RN - Random Delay Slots
    (19) NT - Node Discover Time
    (0) NO - Node Discover Options
    (0) CE - Coordinator Enable
    (1FFE) SC - Scan Channels
    (4) SD - Scan Duration
    (0) A1 - End Device Association
    (0) A2 - Coordinator Association
    (0) AI - Association Indication
    (0) EE - AES Encryption Enable
Set/Read addresses of module to which outputs are bound. Setting all bytes to 0xFF will not
allow any receive I/O packet to change outputs. Setting address to 0xFFFF will allow any
received I/O packet to change outputs.
COM7  9600 8-N-1  FLOW:NONE   XB24  Ver:10EC
```

```
Modem: XBEE   Function Set     Version
XB24          XBEE 802.15.4    10EC
Networking & Security
    (10) CH - Channel
    (1234) ID - PAN ID
    (0) DH - Destination Address High
    (2) DL - Destination Address Low
    (1) MY - 16-bit Source Address
    (13A200) SH - Serial Number High
    (40BB1A58) SL - Serial Number Low
    (0) MM - MAC Mode
    (0) RR - XBee Retries
    (0) RN - Random Delay Slots
    (19) NT - Node Discover Time
    (0) NO - Node Discover Options
    (0) CE - Coordinator Enable
    (1FFE) SC - Scan Channels
    (4) SD - Scan Duration
    (0) A1 - End Device Association
    (0) A2 - Coordinator Association
    (0) AI - Association Indication
    (0) EE - AES Encryption Enable
Modify networking settings
COM7  9600 8-N-1  FLOW:NONE   XB24  Ver:10EC
```

**Figure 46 Receiver vs Transmitter Xbee**

In this project, we will use 2 microcontrollers and 2 Xbee antennas to realize wireless communication between the computer and robot. First, we have to set up the connection between 2 Xbee antennas. This can be done using a program X-CTU as seen in Figure 46. This software will help the users to set up the pan ID as well as channels of multiple Xbee antennas. To set up the Xbee, the user needs to connect the Xbee board to the computer using a micro USB cord. Then, by choosing the “modem configuration” tab, the user can read, write or restore the Xbee to default set up. The window on the left hand side of Figure 46 shows the set up for the transmitter, while the one on the right hand side is for the receiver antenna. Both of these antenna will be connected to two Arduino boards. The first board is connected to the computer while the second board is used to control the robot. Normally, there will be a serial mode connection between the Xbee antenna and the Arduino board. This can be done by connecting the Tx and Rx pins on each Arduino board to the Rx and Tx pins on the Xbees. However, since there has already been a serial mode connection between Arduino board (which controls Xbee) and the Matlab command window (computer), these pins are always busy. We decided to use 3 pins D0, D1, and D2 on each Xbee antenna to set up the communication. The way this communication works is that if the Dx pin on the transmitter is high or low, the corresponding pins on the receiver will also be in the same state. Then these pins are connected to pins on the Arduino boards. The board will then process this 3-bit information and decide which action to perform.

## 7.3 Serial Mode Setup

The serial mode set up takes less than a second for each data transmission.  This can be done using the setupSerial.m and the Setup function in the transmitter.ino (Red Arduino board) shown in Appendix A. The setupSerial function will first connect the Matlab command window with a Comport where the Arduino is connected. Then, this function will decide the baud rate, the number of bits and the stop bit for the communication. The baud rate is set to 9600, there will be 8 data bits in each transmission and the stop bit is assigned as 1. Then, the setupSerial function will send a simple character to the Arduino board. The Arduino board will then take this data and print it back on the command window. After this point, any characters printed on the Matlab command window will be sent to the Arduino board. However, since we need the Matlab command window to do the classification of EEG data taking continuously, after each transmission is done, the serial mode is closed. The Matlab will then classify the incoming PSD data using libsvm library. The new serial connection will then be opened to send the new command to the Arduino board. This loop won't stop until the user decided to close the program or turn off the EPOC.

## 7.4 Robot control implementation

The robot with LCD has two main parts, the transmitter side that sending the signals from Matlab command window, and a receiver side that controlling the robot.

### a) Transmitter side

Figure 47 shows the connection between Xbee and an Arduino board. This is the Arduino board that is connected to the computer, taking care of the information from the Matlab command window. There will be 5 connections between the Arduino board and the Xbee antenna. The first 2 pins provide the power connections or VCC and GND. The last 3 pins 2, 3, and 4 on the board will be connected to D0, D1, and D2 on the Xbee board, respectively. There are 7 different characters can be sent to the board which are ‘s’-Stop, ‘l’-Left, ‘r’-Right, ‘f’-Forward, ‘b’-Backward, ‘A’, and ‘B’. The board will then encode these characters into 3-bit-sequences which are then sent to the Xbee Antenna. Because there are 3 bits, there are $2^3 = 8$ different sequences. However, the “000” and “111” sequences are both used for stop signal (Table 12)

**Table 12 Signal Sequences for different characters**

| Character | D0 | D1 | D2 |
|---|---|---|---|
| ‘s’ | 0/1 | 0/1 | 0/1 |
| ‘l’ | 0 | 0 | 1 |
| ‘r’ | 0 | 1 | 0 |
| ‘f’ | 0 | 1 | 1 |
| ‘b’ | 1 | 0 | 0 |
| ‘A’ | 1 | 0 | 1 |
| ‘B’ | 1 | 1 | 0 |

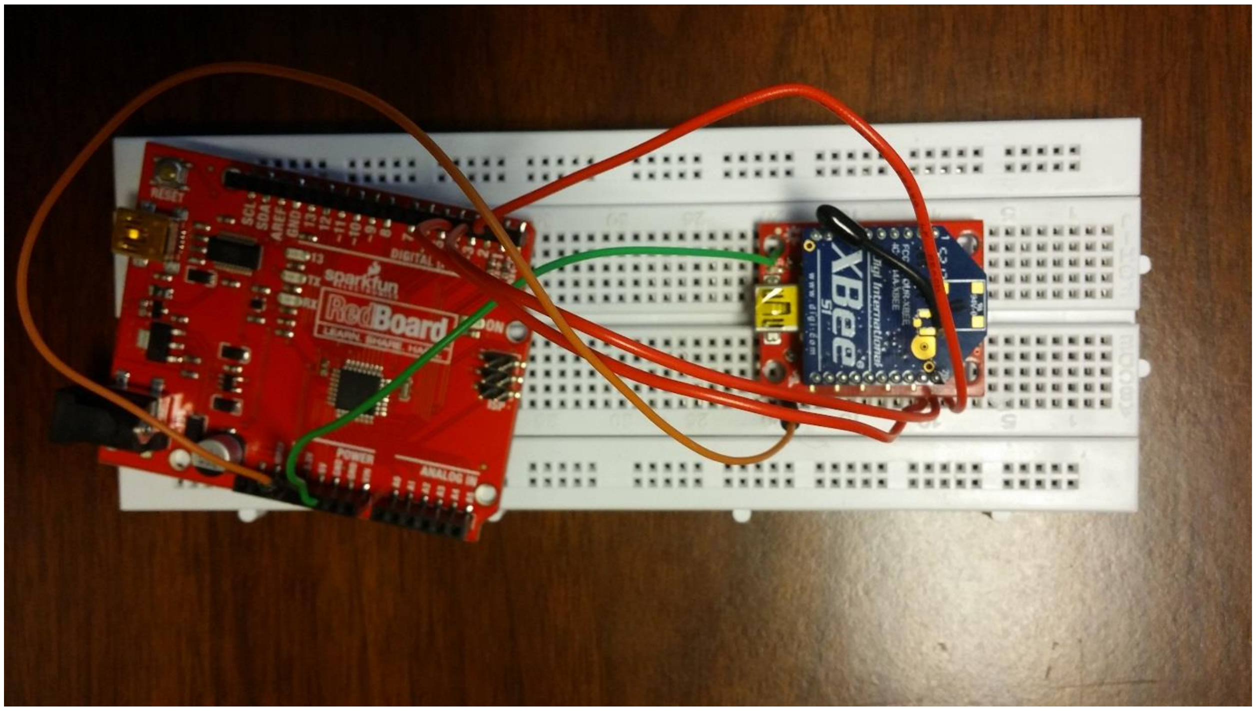


**Figure 47 The Transmitter Arduino and Xbee connection**

### b) Receiver side

The three-wheel robot as shown in Figure 48 is controlled by the other Arduino board. This Arduino board is taking care of more objectives compared to that of the transmitter. There are three main tasks this Arduino board has to complete. Firstly, after receiving signal in term of data bit sequences as illustrated in Table 12, the Arduino board will make a decision to control the robot. Each time a character indicating a directional movement is sent, the robot will move counter clockwise (turning left), clockwise (turning right), forward or backward accordingly. Then, at the end of the transmission, a Stop sequence will be sent to stop the robot completely, which is also the second task of the Arduino board where different pulses are sent to the servo motors. The last task is displaying the results on the LCD screen. Using only 4-bit sequences, the board will print a string on the LCD display illustrates both the directional movements and the two letters A, B. The Arduino code for the motor control and LCD display is shown in Receiver.ino in Appendix A.

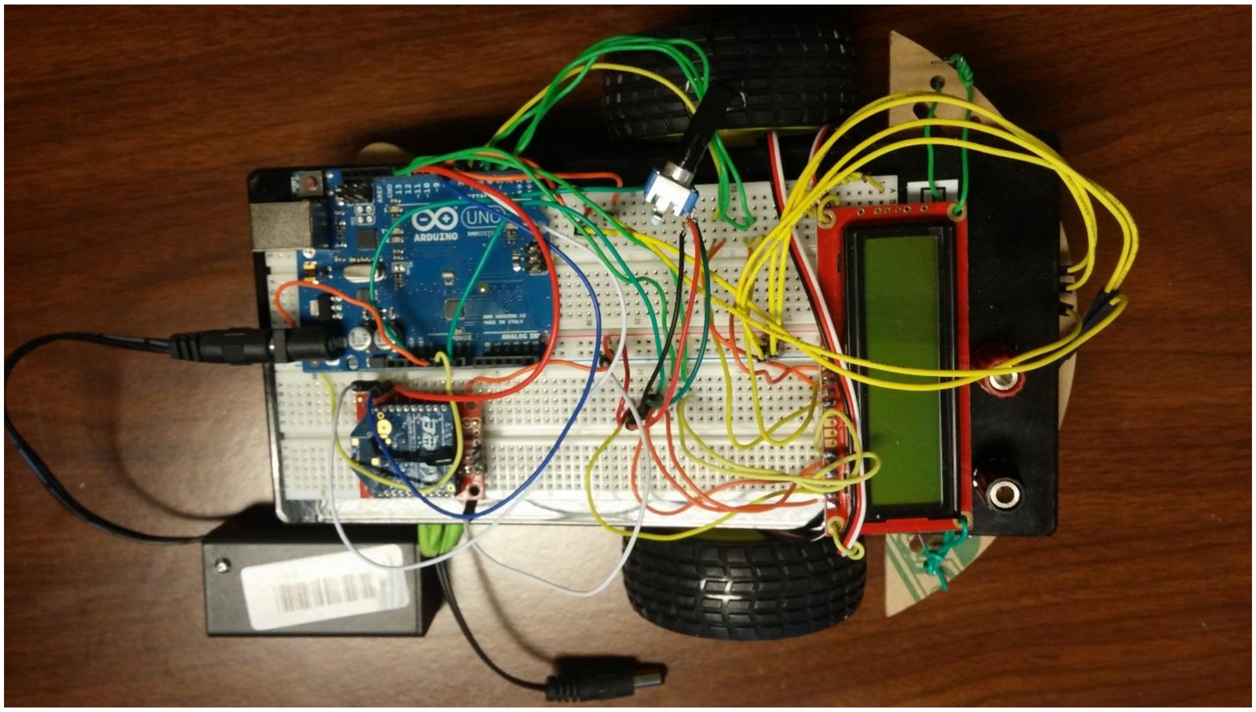


**Figure 48 The three-wheel robot**

## 7.4 Testing robot control system

Using the SVM models with the highest accuracies shown in Section 6.6 are used for the final test. After having a proper connection between the EPOC headset and the computer (all sensors are green), the “RobotControl.m” program was run in real time mode. There were three experiments. In the first experiment, we used the well-trained 2 directional classifier (Table 10) to detect 25 Left and 25 Right thinking. The robot moved in the correct directions with the accuracy of 80% (20/25 of both directions). This result showed only approximately 10% of mismatch between the testing accuracy reported in last chapter and the accuracy of the real time robot control. Then, in the second experiment, we used the well-trained 4 directional classifier (Table 10) to detect 25 left, 25 right, 25 forward and 25 backward thinking. In this experiment, it was easier to control the robot to move left and right directions, we got 17 correct classification results out of 25 for both directions. However, the accuracies of forward and backward are low. We got 14 correct results for Forward movement and 8 correct results for backward movement. The overall accuracy pf this real time testing is 56% which is 26% lower than the testing accuracy reported in Table 10. Finally, a well-trained SVM model for 2 letters A and B was tested. There were 20 correct results out of 25 for letter A while 15 correct results out of 25 for B. This led to a 70% real time testing overall accuracy, which was 13.25% less than the testing accuracy shown in Table 10.

# Chapter 8 Conclusion and Future work

## 8.1 Summary

Overall, the objective of the project was successfully achieved. The main contribution of the project is creating a brain computer interface system (BCI) to control the robotic car with alphabetic display only through human brain EEG signals only. Based on our BCI system, the raw EEG data can be collected and stored in folders automatically and wirelessly. The SVM model can be trained well to reach more than 80% testing accuracies. A detection result from the SVM model can be transmitted to robotic car with LCD to drive the car and show the letters remotely. Therefore, our BCI system has great potential to help people such as ALS patients who cannot move or even talk at all to gain better mobility and communication capability.

EEG-based BCI is a very challenging topic. It is not easy to generate a large amount of good quality EEG signals for better classification results. There is no standard to check for the quality of the signal before making classification. It required a lot of focuses and took a lot of time to generate 6000 EEG data in this project. The second challenging point is the optimization of the parameters in the classification model is time-consuming. Although a good personal computer is used to do this task, it still took days or weeks to complete the SVM training process for some cases. This is also one drawback of the program itself since the user can only see the final result after a long-time training process, it is very hard to figure out the mistakes and redo an experiment. Since the developed BCI system is just an open loop system now, it is hard to control the robot in the real time mode. Although the encouraging results have shown high possibility to control the robot to run in 2 directions, it is still very challenging for the 4 direction control.

## 8.2 Future work

We have several ways to further improve the EEG-based BCI system. First, the software implementation and interfaces can be improved. All of the interfaces between the user and the computer are Matlab built in dialog in this project now. But we can use Matlab GUI to create a more convenient interface. Second, we only focus on PSD data type as the feature of the classification model. PSD is the most basic and simple feature used for EEG data analysis. Some other more complicated features such as cross correlation and coherence have also been used widely. We can consider them in this project to improve the classification accuracy. Third, there is a new EPOC headset (EPOC+) that has a higher sampling rate. The higher density of data might reduce the time required (now 5 seconds) to achieve higher accuracy. Fourth, to improve the calculation efficiency, especially for the training parameters process, we might consider using Super Computer in the future. This will ensure the correctness of the result at the same time improve the calculation speed. Finally, instead of controlling the robot through a computer with Matlab, the EEG-based BIC system can be built in python language. The Libsvm model also supports the python language which is used commonly in Raspberry Pi board. A new interface can be built so that the user will control the Raspberry Pi board directly with EPOC. Then, this tiny board can have the capability of classifying the data and send signal pulse to control the servos and LCD display. Such a system will only need one Raspberry Pi board to run and eliminate the computer, Xbee antennas, and Arduino board, which will reduce the cost and shrink the size of the whole system.

As discussed in [20], the brain control wheelchair (BCW) is a robotic system which has been developed for people who are incapable of using the wheelchair control by joysticks, buttons, or even voice. Our BCI system development is in the early stage. For example, the response time of our system is not short enough for the real-time application in the market. However, the results shown in this project can prove that it is possible that our system can be further improved to make an advanced BWC in the future.

## References

1. F. Lopes da Silva, “EEG: Origin and Measurement”, University of Amsterdam, Amsterdam, Netherlands, 2010.
2. Cortes, C.; Vapnik, V. (1995). "Support-vector networks ". *Machine Learning* 20 (3): 273.
3. Stefan Rueping, “SVM Classifier Estimation from Group Probabilities”, Schloss Birlinghoven, Germany, ICML, 2010.
4. Mary Lyon, R.N., M.N., “Reasons for Living with ALS”. ALS Association [Online]. Available: http://www.alsa.org/als-care/resources/publications videos/factsheets/reasons-for-living-with-als.html
5. “Definition of Amyotrophic Lateral Sclerosis (ALS)”, Medicine Net [Online]. Available: http://www.medterms.com/script/main/art.asp?articlekey=2231
6. C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.
7. Emotiv Insight,” Simulink EEG Importer”, Emotiv Inc [Online] Available: https://emotiv.com/store/product_112.html
8. Gou, Qing “Improving Golf Putt Performance with Statistical Learning of EEG Signals.” Thesis. University of Arkansas, 2014
9. *Niels Landwehr, Mark Hall, and Eibe Frank (2003). Logistic model trees*
10. Ridella S, Rovetta S, Zunino R. (1997). "Circular backpropagation networks for classification". *IEEE Transactions on Neural Networks* 8 (1): 84–97.
11. Evangelista, Paul, Mark Embrechts, and Boleslaw Szymanski. "Taming the Curse of Dimensionality in Kernels and Novelty Detection."Applied Soft Computing Technologies:

The Challenge of Complexity. Vol. 34. Berlin: Springer Berlin Heidelberg, 2006. 425-438. Print

12. G. Pfurtscheller, Ch. Neuper, D. Flotzinger, M. Pregenzer, EEG-based discrimination between imagination of right and left hand movement, Electroencephalography and Clinical Neurophysiology, Volume 103, Issue 6, December 1997, Pages 642-651
13. Srivastava, Durgesh K., and Lekha Bhambhu. "DATA CLASSIFICATION USING SUPPORT VECTOR MACHINE." *Journal of Theoretical and Applied Information Technology* (2005 - 2009): n. pag. Web. Available: http://www.jatit.org/volumes/research-papers/Vol12No1/1Vol12No1.pdf
14. Apaydin, Ethem, 2004. Introduction to Machine Learning. books.google.com.
15. Weisstein, Eric W. "Fast Fourier Transform." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/FastFourierTransform.html
16. Bajpai, Anvita, and Knal Chada. “Real-time Facial Emotion Detection Using Support Vector Machines.” International Journal of Advanced Computer Science and Applications, 1 (2010): 37-40. Print.
17. Teplan, M. “Fundamentals of EEG Measurement.” Measurement of Science Review 2 (2002): n. pag. Print
18. Kahana, Michael J. Robert Sekuler, Jeremy B. Caplan, Mathew Kirschen, and Joseph Madsen. “Human Theta Oscillations Exhibit Task Dependence during Virtual Maze Navigation.” Letters to Nature 399 (1999): 781-84, Print.
19. Weiss, Sabine, and Horst M. Mueller. “The Contribution of EEG Coherence to the Investigation of Language.” Brain and Language 85 (2003) 325-43. Print

20. Brice Rebsamen. A Brain Controlled Wheelchair to Navigate in Familiar Environments. Human-Computer Interaction [cs.HC]. national university of singapore, 2009.

# Appendix A: Matlab Code

## A.1 TakingData.m

```
%% Step 1: collecting general information
open('C:\Program       Files       (x86)\Epoc       Simulink       EEG
Importer\EpocSignalServer.exe');
%open('C:\Program    Files    (x86)\Emotiv    EPOC    Brain    Activity
Map\EmoBrainMap.exe');
open('C:\Program    Files    (x86)\Emotiv    EPOC    Control    Panel
v2.0.0.21\Applications\ConsumerControlPanel.exe');
input_checking=0;
while input_checking==0
prompt = {'Training time for 1 data:','Number of Training items (have
to be divisible for numofclasses)'...
    ,'number of classes:'};
dlg_title = 'Input';
num_lines = 1;
def = {'1','2000','2'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
simstime=str2num(answer{1,:});
simitems=str2num(answer{2,:});
numofclasses=str2num(answer{3,:});
classesNames=cell(1,numofclasses);
for i=1:1:numofclasses
  classesNames{1,i}=strcat('Class',int2str(i));
end
input_checking = ~mod(simitems,numofclasses);
if input_checking==0
 msg1=   msgbox('Number of Training items have to be divisible for num
of classes'...
        , 'Warning');
while(1)
if ishandle(msg1)==0
   break;
end
pause(1);
end
end
end

input_checking=0;
while input_checking==0
Classes = inputdlg(classesNames,dlg_title,num_lines);
  for i=1:numofclasses-1
      cur_class=Classes{i};
      next_class=Classes{i+1};
      if strcmp(cur_class,next_class)
         input_checking=0;
         break;
      else
         input_checking=1;
      end
  end
  if numofclasses==1
      input_checking=1;
  end
if input_checking==0
msg2=   msgbox('Invalid Classes Name (they have to be different)'...
        , 'Warning');
while(1)
if ishandle(msg2)==0
   break;
end
pause(1);
end
end

end
%% Step 2: Making Destination Folders
Cur_dir=pwd;
FileDir_vetor=cell(numofclasses,1);
new_Folders=numofclasses
Cur_folders=dir;
Folder_mark=zeros(1,new_Folders);
for i=1:1:numofclasses
 if exist(Classes{i})==7
   new_Folders=new_Folders-1;
   Folder_mark(1,i)=1;
 end
end
if new_Folders~=0
msg3_string='The new Folders are: '
for i=1:1:numofclasses
  if Folder_mark(1,i)~=1
     msg3_string=strcat(msg3_string,', ',Classes{i,:});
  end

end
msg3=   msgbox(msg3_string...
        , 'Warning');
while(1)
if ishandle(msg3)==0
   break;
end
pause(1);
end
else
msg3=   msgbox('No new Folders will be made'...
        , 'Warning');
while(1)
if ishandle(msg3)==0
   break;
end
pause(1);
end
end
for i=1:1:new_Folders
   if Folder_mark(1,i)~=1
    New_Dir=strcat(Cur_dir,'\',Classes{i,:})
    mkdir(New_Dir);
   end
end
%% Step 3: Making random indexes vector
Data_indexes=zeros(1,simitems);
for i=0:1:(numofclasses-1)
   start_index=i/numofclasses*simitems+1;
   stop_index=(i+1)/numofclasses*simitems;
   for j= start_index:1:stop_index
       Data_indexes(1,j)=i+1;
   end
end
%taking current time to get different rand state every time
cur_state=sum(clock);
rand('state',cur_state); randn('state',cur_state);

Trainging_indexes=randperm(simitems);
Data_indexes=Data_indexes(1,Trainging_indexes)
%% Step 4: Taking New data
back=1;
for i=1:1:simitems
Cur_index=Data_indexes(back)
DataType=Classes{Cur_index,:};
Message=['Prepare  to  take:  ',  DataType,'  for',int2str(simstime),  '
seconds'];
Saving_Dir=strcat(Cur_dir,'\',DataType);
Data_index=length(dir(Saving_Dir))-2;
New_File_Dir=strcat(Saving_Dir,'\','Time_data',DataType
,int2str(Data_index), '.mat')
msg=msgbox(Message);
while(1)
if ishandle(msg)==0
   break;
end
pause(1);
end
Timespan=[0 simstime];
sim('EmotivEpocEEG_testmodel_2011a.mdl',Timespan);
S=load('res.mat');
A=S.ans;
size=length(A)-1;
```

```
X=A(3:16,1:size);
Taken_data=X;
choice     =     questdlg('Was     that     a     good     data?','Data
checker','Yes','No','Stop','Yes');
switch choice
    case 'Yes'
          save(New_File_Dir,'Taken_data')
          back=i+1
    case 'No'
          back=i;
    case 'Stop'
          break;
end
end
```

## A.2 ConvertData.m

```
input_checking=0;
while input_checking==0
prompt = {'number of classes:'};
dlg_title = 'Input';
num_lines = 1;
def = {'2'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
numofclasses=str2num(answer{1,:});
classesNames=cell(1,numofclasses);
for i=1:1:numofclasses
  classesNames{1,i}=strcat('Class',int2str(i));
end
input_checking=1;
end
input_checking=0;
while input_checking==0
Classes = inputdlg(classesNames,dlg_title,num_lines);
  for i=1:numofclasses-1
      cur_class=Classes{i};
      next_class=Classes{i+1};
      if strcmp(cur_class,next_class)
         input_checking=0;
         break;
      else
         input_checking=1;
      end
  end
if numofclasses==1
    input_checking=1;
end
if input_checking==0
msg2=   msgbox('Invalid Classes Name (they have to be different)'...
        , 'Warning');
while(1)
if ishandle(msg2)==0
   break;
end
pause(1);
end
end
end
Cur_dir=pwd;
FileDir_vetor=cell(numofclasses,1);
new_Folders=numofclasses
Cur_folders=dir;
Folder_mark=zeros(1,new_Folders);
for i=1:1:numofclasses
 if exist(Classes{i},'dir')==7
   new_Folders=new_Folders-1;
   Folder_mark(1,i)=1;
 end
end
if new_Folders==0
msg3_string='The Folders exist'
for i=1:1:numofclasses
  if Folder_mark(1,i)~=1
     msg3_string=strcat(msg3_string,', ',Classes{i,:});
  end

end
msg3=   msgbox(msg3_string...
        , 'Warning');
while(1)
if ishandle(msg3)==0
   break;
end
pause(1);
end
else
msg3=   msgbox('These Folders are not existed'...
        , 'Warning');
while(1)
if ishandle(msg3)==0
   break;
end
pause(1);
end
end
%% Count Data
CountSource=zeros(numofclasses,1);
for i=1:1:numofclasses
   if Folder_mark(1,i)==1
    FileDir{i}=strcat(Cur_dir,'\',Classes{i,:})
    CountSource(i)=length(dir(FileDir{i}))-2;
   end
end

msg=    msgbox('Ready to input 1==yes 0==no What Data Type You wanna
Create ? (New Dir will be made)'...
        , 'Warning');
while(1)
if ishandle(msg)==0
   break;
end
pause(1);
end
input_checking=0;
while input_checking==0
prompt = {'PSD data in Db','PSD data in uW'};
dlg_title = 'Input';
num_lines = 1;
```

```
def = {'0','0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
Db=str2num(answer{1,:});
uW=str2num(answer{2,:});
for i=1:1:2
   if str2num(answer{i,:})~=1&&str2num(answer{i,:})~=0
       msg=   msgbox('Plz input 1 or 0 only)'...
        , 'Warning');
       while(1)
       if ishandle(msg)==0
        break;
       end
       pause(1);
       end
      input_checking=0;
   else
      input_checking=1;
   end
end

end
New_Folders=2*numofclasses*5;
Folder_mark=zeros(1,New_Folders);
NewFolderNames=cell(New_Folders,1)
for i=0:1:numofclasses-1
    for j=0:1:2
        for k=1:1:5
          if j==0

NewFolderNames{i*10+5*j+k}=[Classes{i+1},'PSD_Db',num2str(k),'s'];
          elseif j==1

NewFolderNames{i*10+5*j+k}=[Classes{i+1},'PSD_uW',num2str(k),'s'];
        end
        end
    end

end

for i=1:1:New_Folders
 if exist(NewFolderNames{i})~=7
   Temp_Dir=strcat(Cur_dir,'\',NewFolderNames{i,:})
   mkdir(Temp_Dir)
 end
end
if Db==1
 i=1;
 Class=1;
 while(i<New_Folders)
   for sec=0:4
     Temp_Dir=strcat(Cur_dir,'\',NewFolderNames{i+sec,:})
     count_dir=length(dir(Temp_Dir))-2;
     for                             j=count_dir:1:CountSource(Class)-1
FileRead=[FileDir{Class},'\','Time_data',Classes{Class},num2str(j),'.m
at']
         X=load(FileRead);
         X=X.Taken_data;
         X=X(:,1:128*(sec+1));
         Fs=128;
         N=128*(sec+1);
         for k=1:1:14
         original_data=X;
         window=hamming(N);
         psd_data(k,:)                                                  =
10*log10(2*periodogram(original_data(k,:),window,0:Fs/2,Fs));
         end
         Taken_data=psd_data;

FileSave=[Temp_Dir,'\','PSD_data_of_',NewFolderNames{i+sec},num2str(j)
,'.mat']
         save(FileSave,'Taken_data');
     end
   end
    Class=Class+1;
     i=i+10;
 end
end

if uW==1
 i=6;
 Class=1;
 while(i<New_Folders)
   for sec=0:4
     Temp_Dir=strcat(Cur_dir,'\',NewFolderNames{i+sec,:})
     count_dir=length(dir(Temp_Dir))-2;
     for j=count_dir:1:CountSource(Class)-1

FileRead=[FileDir{Class},'\','Time_data',Classes{Class},num2str(j),'.m
at']
         X=load(FileRead);
         X=X.Taken_data;
         X=X(:,1:128*(sec+1));
         Fs=128;
         N=128*(sec+1);
         for k=1:1:14
         original_data=X;
         window=hamming(N);
         psd_data(k,:)                                                  =
2*periodogram(original_data(k,:),window,0:Fs/2,Fs);
         end
         Taken_data=psd_data;

FileSave=[Temp_Dir,'\','PSD_data_of_',NewFolderNames{i+sec},num2str(j)
,'.mat']
```

```
         save(FileSave,'Taken_data');
     end
   end
    Class=Class+1;
     i=i+10;
 end
end
```

## A.3 svm_train_n_categories.m

```
close all;
clear all;
clc;
Timer=clock();
Year=num2str(Timer(1));
Month=num2str(Timer(2));
Day=num2str(Timer(3));
Hour=num2str(Timer(4));
Min=num2str(Timer(5));
algo_choice=0;
choice = questdlg('What type of algorithm you want to choose
?','Algorithm checker','Seeding Analysis','Making a Predict
Model','Seeding Analysis');
switch choice
    case 'Seeding Analysis'
          algo_choice=0;
    case 'Making a Predict Model'
          algo_choice=1;
end
if algo_choice==0

ReportTxtName=['ResultReport','_on_',Month,'_',Day,'_',Year,'_at_',Hou
r,'_',Min,'.txt']
    prompt = {'File Save Name'};
    dlg_title = 'File name';
    num_lines = 1;
    def = {ReportTxtName};
    answer = inputdlg(prompt,dlg_title,num_lines,def);
    ReportTxtName=(answer{1,:});

    resultfile=fopen(ReportTxtName,'w')
    prompt = {'Wished Test Acc (1- 100):'};
    dlg_title = 'Input your choice';
    num_lines = 1;
    def = {'70'};
    answer = inputdlg(prompt,dlg_title,num_lines,def);
    Wished_Test_Acc=str2num(answer{1,:});
end
input_checking=0;
while input_checking==0
prompt = {'number of Folders:','Datasize','Num of layers'};
dlg_title = 'Input--(Datasize=0 for biggest Datasize will be taken';
num_lines = 1;
def = {'2','0','10'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
input_checking=1;
NumsofFolders=str2num(answer{1,:});
datasize=str2num(answer{2,:});
Layers=str2num(answer{3,:});
```

```
input_checking=1;
if mod(datasize,Layers)~=0 || datasize/Layers<50
   input_checking=0;
end
if input_checking==0
msg2=   msgbox('datasize is not divible for layers'...
        , 'Warning');
while(1)
if ishandle(msg2)==0
   break;
end
pause(1);
end
end
end
classesNames=cell(1,NumsofFolders);
for i=1:1:NumsofFolders
  classesNames{1,i}=strcat('Class',int2str(i));
end
input_checking=0;
while input_checking==0
dlg_title = 'Input';
num_lines = 1;
Classes = inputdlg(classesNames,dlg_title,num_lines);
 for i=1:NumsofFolders-1
      cur_class=Classes{i};
      next_class=Classes{i+1};
      if strcmp(cur_class,next_class)
         input_checking=0;
         break;
      else
         input_checking=1;
      end
 end
    if NumsofFolders==1
      input_checking=1;
    end
if input_checking==0
msg2=   msgbox('Invalid Classes Name (they have to be different)'...
        , 'Warning');
while(1)
if ishandle(msg2)==0
   break;
end
pause(1);
end
end
end
if algo_choice==1
% trained seed list here
LayersNames=cell(1,Layers);
for i=1:1:Layers
```

```
  LayersNames{1,i}=strcat('Seed_For_Layers',int2str(i));
end

dlg_title = 'Input';
num_lines = 1;
SeedsList = inputdlg(LayersNames,dlg_title,num_lines);

end
Cur_dir=pwd;
FileDir=cell(NumsofFolders,1);
NumofFiles=zeros(NumsofFolders,1);
for i=1:NumsofFolders
FileDir{i}=strcat(Cur_dir,'/',Classes{i})
NumofFiles(i,:)=length(dir(FileDir{i}))-2;
end
if datasize~=0
Best_Num_Train=datasize;
else
Best_Num_Train=min(NumofFiles);
end

input_checking=0
while input_checking==0
prompt = {'Enter Selected Channels separate by dashes:'};
dlg_title = 'Input';
num_lines = 2;
def = {'1-2-3-4-5-6-7-8-9-10-11-12-13-14'};
SelectedChannels = inputdlg(prompt,dlg_title,num_lines,def);
SelectedChannels=SelectedChannels{1};
counter=0;
for i=1:length(SelectedChannels)
    if strcmp(SelectedChannels(i),'-')
     counter=counter+1;
    end
end
numofChannels=counter+1;
choosing_vector=zeros(numofChannels,1)
String='';
num_count=0;
dash_last_index=0;
for i=1:length(SelectedChannels)
    if ~strcmp(SelectedChannels(i),'-')
     String=[String,SelectedChannels(i)];
    else
     if i>dash_last_index
       dash_last_index=i;
     end
     num_count=num_count+1;
     choosing_vector(num_count,:)=str2num(String);
     String='';
    end
```

```
end
choosing_vector(num_count+1,:)=str2num(SelectedChannels(dash_last_inde
x+1:length(SelectedChannels)))
if numofChannels<0 || numofChannels>14
msg3=   msgbox('Out of Bound','Warning');
while(1)
if ishandle(msg3)==0
   break;
end
pause(1);
end
input_checking=0;
else
input_checking=1;
end
end

input_checking=0;
while input_checking==0
prompt = {'Start Frequency: ','Stop Frequency: '};
dlg_title = 'Input Frequency From 0--64 Hz';
num_lines = 1;
def = {'4','30'};
answer = inputdlg(prompt,dlg_title,num_lines,def);

F_Start=str2num(answer{1,:});
F_Stop=str2num(answer{2,:});
if  F_Start==F_Stop  ||  F_Start>F_Stop  ||  F_Start<0  ||  F_Start>65  ||
F_Stop<0 || F_Stop>65
   input_checking=0;
else
   input_checking=1;
end

if input_checking==0
   msg4=   msgbox('Invalid Frequency Range'...
        , 'Warning');
while(1)
if ishandle(msg4)==0
   break;
end
pause(1);
end

end
end
F_range=F_Start:1:F_Stop;
input_checking=0;
Myseed=1:10;
Mylatch=0:1:Layers-1
Total_Num_Train=Best_Num_Train;
Best_Num_Train=Best_Num_Train/Layers;
```

```
Sectorlength=Best_Num_Train;

if algo_choice==1
  Wished_Test_Acc=0;
  Myseed=1;
  Mylatch=0;
  Mysector=0:1:Layers-1;
  Best_Num_Train=Total_Num_Train;
else
  Mysector=1;
end
Wished_Test_Acc=Wished_Test_Acc/100;
for latch=Mylatch

 for seed=Myseed

if algo_choice==0&&seed==1
fprintf(resultfile,'\n');
FirstLine=['Seed','              ','Best_C','           ','Best_gamma','
','Train_acc','   ','Test_acc','   ','Total_SV','\n']
fprintf(resultfile,FirstLine);
end

Data=zeros(Best_Num_Train*NumsofFolders,numofChannels*length(F_range))
;
 Data_row=zeros(1,numofChannels*length(F_range));
 Classify=zeros(Best_Num_Train*NumsofFolders,1);
if algo_choice==0||Layers==1
    I_Start=latch*Best_Num_Train;
    I_Stop=I_Start+Best_Num_Train-1;
 for i=0:NumsofFolders-1
 for j=I_Start:1:I_Stop

load_file=[FileDir{i+1},'\','PSD_data_of_',Classes{i+1},int2str(j),'.m
at']
   S=load(load_file)
   X=S.Taken_data(choosing_vector',F_range)

   for k=0:1:numofChannels-1

Data_row(length(F_range)*k+1:length(F_range)*k+length(F_range))=X(k+1,
:);
   end

  Classify(Best_Num_Train*i+j-I_Start+1,:)=i+1;
  Data(Best_Num_Train*i+j-I_Start+1,:)=Data_row;

  end
 end
else

for l=Mysector
```

```
    Jump=Best_Num_Train/Layers
    I_Start=l*Jump;
    I_Stop=I_Start+Jump-1;
 for i=0:NumsofFolders-1
    for j=I_Start:1:I_Stop

load_file=[FileDir{i+1},'\','PSD_data_of_',Classes{i+1},int2str(j),'.m
at']
   S=load(load_file)
   X=S.Taken_data(choosing_vector',F_range)
   for k=0:1:numofChannels-1

Data_row(length(F_range)*k+1:length(F_range)*k+length(F_range))=X(k+1,
:);
   end


  Classify(Jump*(i+l*NumsofFolders)+(j-I_Start)+1,:)=i+1;
  Data(Jump*(i+l*NumsofFolders)+(j-I_Start)+1+1,:)=Data_row;

  end
 end
end
end


% Start Training
tic
timedata_all = Data;
decision_data_all = Classify;
saved_minarray=min(timedata_all,[],1);
saved_ratio=spdiags(1./(max(timedata_all,[],1)-
min(timedata_all,[],1))',0,size(timedata_all,2),size(timedata_all,2));
timedata_all              =              (timedata_all                -
repmat(saved_minarray,size(timedata_all,1),1))*saved_ratio;
decision_data = decision_data_all(:,1);
range=length(decision_data');
timedata = timedata_all(1:1:range,:);
decision_data = decision_data(1:1:range,:);
x_all=timedata;
y_all = decision_data;
finaltest_index=zeros(1,length(y_all)/5);
for sector=Mysector
for i=0:1:NumsofFolders-1
    if algo_choice==1
       seed=str2num(SeedsList{sector+1});
       rand('state',seed); randn('state',seed);
       if Layers==1

finaltest_index(i*length(find(y_all==i+1))/5+1:1:(i+1)*length(find(y_a
ll==i+1))/5)=randsample(find(y_all==i+1),length(find(y_all==i+1))/5);
       else
```

```
        Jumper=Layers;
        Section_length=Best_Num_Train/Layers*NumsofFolders;

finaltest_index(sector*length(y_all)/5/Jumper+i*length(find(y_all==i+1
))/Jumper/5+1:1:sector*length(y_all)/5/Jumper+(i+1)*length(find(y_all=
=i+1))/Jumper/5)=Section_length*sector+randsample(find(y_all(Section_l
ength*sector+1:Section_length*(sector+1))==i+1),length(find(y_all==i+1
))/5/Jumper);
        end
        else
        rand('state',seed); randn('state',seed);

finaltest_index(i*length(find(y_all==i+1))/5+1:1:(i+1)*length(find(y_a
ll==i+1))/5)=randsample(find(y_all==i+1),length(find(y_all==i+1))/5);
    end
end
end
x_finaltest     = x_all(finaltest_index,:);
y_finaltest     = y_all(finaltest_index)';
Train_index     = setdiff(1:length(y_all),finaltest_index);
x_Train_all1     = x_all(Train_index,:);
y_Train_all1     = y_all(Train_index)';
[y_Train_all,right_order] = sort(y_Train_all1);
x_Train_all               = x_Train_all1(right_order,:);
alpha = 1/NumsofFolders;
%% start calculating
bond    = 80;  % this bond is to limit the training accuracy
bondgap=40
percentage_bound = bond ;
xValidationFolds = 5;
% train_rbfsvm_official is the function we use to train data for
getting
% best parameters C ,gamma ,and SVM structure
y= y_Train_all';
x= x_Train_all;
if algo_choice==1
y= y_Train_all1';
x= x_Train_all1;
end
indexes=zeros(NumsofFolders,length(y_Train_all)/NumsofFolders);
for i=0:1:NumsofFolders-1
indexes(i+1,:)=find(y==i+1);
end
fid=fopen('writeweight.m','w')
for i=1:NumsofFolders
write='w%i=%i ;\r\n';
text=[i 1]'
fprintf(fid,write,text)
end
fclose(fid);
writeweight
Col=floor(length(y_Train_all)/NumsofFolders/xValidationFolds);
```

```
%------------------------
Temp_indexes=indexes(1,:);
Temp_indexes_new=
reshape(Temp_indexes(randperm(xValidationFolds*Col)),xValidationFolds,
Col);
expected_range=length(Temp_indexes_new);
New_indexes=zeros(NumsofFolders*5,expected_range);
for sector=Mysector
for i=0:1:NumsofFolders-1
if algo_choice==1
seed=str2num(SeedsList{sector+1});
rand('state',seed); randn('state',seed);
lID=length(indexes)/Layers;
lID1=Col/Layers
Temp_indexes=indexes(i+1,sector*lID+1:(sector+1)*lID);
New_indexes(i*5+1:(i+1)*5,sector*lID1+1:(sector+1)*lID1)=reshape(Temp_
indexes(randperm(xValidationFolds*Col/Layers)),xValidationFolds,Col/La
yers);
else
rand('state',seed); randn('state',seed);
Temp_indexes=indexes(i+1,:);
New_indexes(i*5+1:(i+1)*5,:)=reshape(Temp_indexes(randperm(xValidation
Folds*Col)),xValidationFolds,Col);
end
end
end
% optimize C
% optimize gamma
if algo_choice==1
choice = questdlg('What type of optimization you want to choose
?','Algorithm checker','C optimizer','gamma optimizer','Making a
model','Making a model');
switch choice
    case 'gamma optimizer'
          option=0;
    case 'C optimizer'
          option=1;
    case 'Making a model'
          option=3;
end
else
    option=2
end
search=1:5
dlg_title='Input otimized para'
if option==0
prompt = {'Input C:'};
num_lines = 1;
def = {'0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
GuessBestC=str2num(answer{1});
search=1:5
```

```
elseif option==1
prompt = {'Input Gamma:'};
num_lines = 1;
def = {'0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
GuessBestGamma=str2num(answer{1});
search=1:5
elseif option==3
prompt = {'Input Gamma:','Input C:'};
num_lines = 1;
def = {'0','0'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
GuessBestGamma=str2num(answer{1});
GuessBestC=str2num(answer{2});
search=1;
xValidationFolds=1;
end
for search=search
if search==1
    step1 =1 ;
    if option==2
        min_c = -5; max_c =15;
        C_array = 2.^(min_c:step1:max_c);
        % original gamma step 1
        min_gamma = -10; max_gamma = 5;
        gamma_array = 2.^(min_gamma:step1:max_gamma);
    elseif option==0
        C_array=GuessBestC;
        min_gamma = -10; max_gamma = 5;
        gamma_array = 2.^(min_gamma:step1:max_gamma);
    elseif option==1
        gamma_array=GuessBestGamma;
        min_c = -5; max_c =15;
        C_array = 2.^(min_c:step1:max_c);
    elseif option==3
        gamma_array=GuessBestGamma
        C_array=GuessBestC;
    end
else
    step1=step1/2;
    if option==2
        C_range           =           max(min_c,           log2(best_C)-
step1*10):step1:min(log2(best_C)+step1*10,max_c);
        C_array = 2.^(C_range);
        gamma_range       =       max(min_gamma,       log2(best_gamma)-
step1*10):step1:min(log2(best_gamma)+step1*10,max_gamma);
        gamma_array=2.^(gamma_range);
    elseif option==0
        C_array=GuessBestC;
        min_gamma = -10; max_gamma = 5;
        gamma_array = 2.^(min_gamma:step1:max_gamma);
    elseif option==1
```

```
        gamma_array=GuessBestGamma;
        min_c = -5; max_c =15;
        C_array = 2.^(min_c:step1:max_c);
    elseif option==3
        gamma_array=GuessBestGamma
        C_array=GuessBestC;
    end
end
C_length = length(C_array);
G_length=length(gamma_array);
train_correct_rate=zeros(C_length, G_length);
test_correct_rate=zeros(C_length, G_length);
optimazation_criterion_2=zeros(C_length, G_length);

for k=1:C_length
 for j=1:G_length
     for fold=1:xValidationFolds
         testPortion=zeros(1,expected_range*NumsofFolders);
                for i=0:1:NumsofFolders-1
                    temp_indexes=New_indexes(i*5+1:(i+1)*5,:);

testPortion(1,i*expected_range+1:1:expected_range*(i+1))=temp_indexes(
fold,:);
                end

            trainPortion  = setdiff(1:length(y),testPortion);
            x_train = x(trainPortion, :);
            y_train = y(trainPortion, :);
            x_test= x(testPortion,:);
            y_test= y(testPortion,:);
            cur_C=C_array(k);
            cur_S=gamma_array(j);
            fid=fopen('writePara.m','w')
            write='Parameters    =  [''-c  ''  num2str(C_array(k))  ''  -g
'' num2str(gamma_array(j)) '
            for i=1:NumsofFolders
            write=[write,'''-w',num2str(i),'           ''           ','
','num2str(w',num2str(i),')',' '];
            if i==NumsofFolders
            write=[write,' ','''' -b 0''',' ];'];
            end
            end
            fprintf(fid,write)
            fclose(fid)
            writePara;
            model        = svmtrain(y_train,x_train,Parameters);
            [group_train,acc_train,val_train]   =   svmpredict(y_train,
x_train, model,'-b 0');
            [group_test,acc_test,val_test]        = svmpredict(y_test,
x_test, model,'-b 0');
```

```
train_correct_rate(k,j)=train_correct_rate(k,j)+100*sum(group_train==y
_train)/size(x_train,1);

test_correct_rate(k,j)=test_correct_rate(k,j)+100*sum(group_test==y_te
st)/size(x_test,1);
            Pro_test=zeros(NumsofFolders,1);

            for i=1:NumsofFolders
               Pro_test(i)=
length(intersect(find(y_test==i),find(group_test~=i)))/length(find(y_t
est==i));
                if isnan(Pro_test(i));
                Pro_test(i) = 0;
               end
            end
  for i=1:NumsofFolders
            optimazation_criterion_2(k,                             j)=
optimazation_criterion_2(k, j)+alpha*Pro_test(i);
  end
  end

train_correct_rate(k,j)=train_correct_rate(k,j)/xValidationFolds;

test_correct_rate(k,j)=test_correct_rate(k,j)/xValidationFolds;
        optimazation_criterion_2(k,   j)=   optimazation_criterion_2(k,
j)/xValidationFolds;

          if        train_correct_rate(k,j)<percentage_bound        ||
test_correct_rate(k,j)<train_correct_rate(k,j)-bondgap
           optimazation_criterion_2(k, j)=Inf;
          end
            end % end j
end %end k
train_correct_rate
optimazation_criterion_2
test_correct_rate
[best_rate1, index1]=min(optimazation_criterion_2)
[best_rate2, index2]=min(best_rate1)
if option==2
    best_C=C_array(index1(1, index2))
    best_gamma=gamma_array(index2)
elseif option==0
    best_C=C_array
    best_gamma=gamma_array(index2)
elseif option==1
    best_gamma=gamma_array
    best_C=C_array(index1(1, index2))
elseif option ==3
    best_C=C_array
    best_gamma=gamma_array
end
```

```
best_rate2
display('end')
display(search)
end
 fid=fopen('writeFPara.m','w')
            write='Final_Parameters     = [''-c '' num2str(best_C) '' -
g '' num2str(best_gamma) '
            for i=1:NumsofFolders
            write=[write,'''-w',num2str(i),'           ''           ','
','num2str(w',num2str(i),')',' '];
            if i==NumsofFolders
            write=[write,' ',''' -b 0''',' ];'];
            end
            end
            fprintf(fid,write)
            fclose(fid)
writeFPara
final_model                                                         =
svmtrain(y_Train_all',x_Train_all,Final_Parameters);
[class_train,~,p_train]                                             =
svmpredict(double(y_Train_all'),x_Train_all, final_model,'-b 0');
[class_test,~,p_test]                                               =
svmpredict(double(y_finaltest'),x_finaltest, final_model,'-b 0');
label_train=class_train;
label_test=class_test;
train_acc                       =  sum(label_train  ==  y_Train_all')  ./
numel(y_Train_all)    % Accuracy
test_acc                         =  sum(label_test  ==  y_finaltest')  ./
numel(y_finaltest)
C_train                   = confusionmat(y_Train_all,label_train)
C_test                    = confusionmat(y_finaltest,label_test)
numSV=final_model.totalSV
Cur_dir=pwd;
Para_dir=strcat(Cur_dir,'/','Parameters');
String=strcat(Para_dir,'/','Para_',int2str(Best_Num_Train),'data_');
for i=1:1:NumsofFolders
     String=strcat(String,Classes{i},'_');
end
if algo_choice==1
Name=strcat(int2str(numofChannels),'Channels','(',SelectedChannels
,')','_',num2str(test_acc),'.mat')
uisave('final_model',String);
else
if test_acc>=Wished_Test_Acc
ResultWrite=[num2str(seed),'                       ',num2str(best_C),'
',num2str(best_gamma),'                         ',num2str(train_acc),'
',num2str(test_acc),'        ',num2str(numSV),'\n']
fprintf(resultfile,ResultWrite);
end
F_rangestr=[num2str(F_range(1)),'-',num2str(F_range(length(F_range)))]
IndexRange=[num2str(I_Start),'-',num2str(I_Stop),'ID']
```

```
LastLine=['Result    of    ',    num2str(Best_Num_Train),'    data    ',
SelectedChannels, 'channel ',F_rangestr,'Hz ', IndexRange]
for i=1:NumsofFolders
LastLine=[LastLine,' ',Classes{i}]
end
LastLine=[LastLine,'\n']
fprintf(resultfile,LastLine);
end
end
end
Test_acc=num2str(Wished_Test_Acc)
VeryLastLine=['Result   of   ',   num2str(Total_Num_Train),'   data   ',
SelectedChannels, 'channel ',F_rangestr,'Hz ']
for i=1:NumsofFolders
VeryLastLine=[VeryLastLine,' ',Classes{i}]
end
VeryLastLine=[VeryLastLine,'with Wished_Test_Acc of: ',Test_acc]
fprintf(resultfile,VeryLastLine);
fclose(resultfile)
```

## A.4 RobotControl.m

```
clc
clear all
open('C:\Program Files (x86)\Epoc Simulink EEG Importer\EpocSignalServer.exe');
open('C:\Program Files (x86)\Emotiv EPOC Control Panel v2.0.0.21\Applications\ConsumerControlPanel.exe');
choice = questdlg('What type of Run you want to choose ?','Mode','Practice Mode','Real Time Mode','Practice Mode');
switch choice
    case 'Practice Mode'
          M=0;
    case 'Real Time Mode'
          M=1;
end
Cur_dir=pwd;
Para_folder=[Cur_dir,'\','Parameters']
Parachoice=0
while Parachoice==0
[Parachoice,ParaPath] = uigetfile('*.mat','Select The Parameters',Para_folder)
RunModel=load([ParaPath,Parachoice])
RunModel=RunModel.final_model
end
if M==0
input_checking=0;
while input_checking==0
prompt = {'number of Folders:','Datasize','Time','Points:'};
dlg_title = 'Input--(Datasize=0 for biggest Datasize will be taken';
num_lines = 1;
def = {'2','1000','10','30'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
NumsofFolders=str2num(answer{1,:});
datasize=str2num(answer{2,:});
TimeTrain=str2num(answer{3,:});
Points=str2num(answer{4,:});
input_checking=1;
end
else
input_checking=0;
while input_checking==0
prompt = {'number of Folders:','Datasize','Time'};
dlg_title = 'Input--(Datasize=0 for biggest Datasize will be taken';
num_lines = 1;
def = {'2','1000','5'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
input_checking=1;
NumsofFolders=str2num(answer{1,:});
datasize=str2num(answer{2,:});
TimeTrain=str2num(answer{3,:});
```

```
input_checking=1;
end
end

classesNames=cell(1,NumsofFolders);
for i=1:1:NumsofFolders
  classesNames{1,i}=strcat('Class',int2str(i));
end

input_checking=0;

while input_checking==0
dlg_title = 'Input';
num_lines = 1;
Classes = inputdlg(classesNames,dlg_title,num_lines);
 for i=1:NumsofFolders-1
      cur_class=Classes{i};
      next_class=Classes{i+1};
      if strcmp(cur_class,next_class)
         input_checking=0;
         break;
      else
         input_checking=1;
      end
 end
    if NumsofFolders==1
      input_checking=1;
    end
if input_checking==0
msg2=   msgbox('Invalid Classes Name (they have to be different)'...
        , 'Warning');
while(1)
if ishandle(msg2)==0
   break;
end
pause(1);
end
end

end
FileDir=cell(NumsofFolders,1);
NumofFiles=zeros(NumsofFolders,1);
for i=1:NumsofFolders
FileDir{i}=strcat(Cur_dir,'/',Classes{i})
NumofFiles(i,:)=length(dir(FileDir{i}))-2;
end
if datasize~=0
Best_Num_Train=datasize;
else
Best_Num_Train=min(NumofFiles);
end
```

```
input_checking=0
while input_checking==0
prompt = {'Enter Selected Channels separate by dashes:'};
dlg_title = 'Input';
num_lines = 2;
def = {'1-2-3-4-5-6-7-8-9-10-11-12-13-14'};
SelectedChannels = inputdlg(prompt,dlg_title,num_lines,def);
SelectedChannels=SelectedChannels{1};
counter=0;
for i=1:length(SelectedChannels)
    if strcmp(SelectedChannels(i),'-')
     counter=counter+1;
    end
end
numofChannels=counter+1;
choosing_vector=zeros(numofChannels,1)
String='';
num_count=0;
dash_last_index=0;
for i=1:length(SelectedChannels)
    if ~strcmp(SelectedChannels(i),'-')
     String=[String,SelectedChannels(i)];
    else
     if i>dash_last_index
       dash_last_index=i;
     end
     num_count=num_count+1;
     choosing_vector(num_count,:)=str2num(String);
     String='';
    end

end
choosing_vector(num_count+1,:)=str2num(SelectedChannels(dash_last_inde
x+1:length(SelectedChannels)))
if numofChannels<0 || numofChannels>14
msg3=   msgbox('Out of Bound','Warning');
while(1)
if ishandle(msg3)==0
   break;
end
pause(1);
end
input_checking=0;
else
input_checking=1;
end
end

input_checking=0;
while input_checking==0
prompt = {'Start Frequency: ','Stop Frequency: '};
dlg_title = 'Input Frequency From 0--64 Hz';
```

```
num_lines = 1;
def = {'4','30'};
answer = inputdlg(prompt,dlg_title,num_lines,def);

F_Start=str2num(answer{1,:});
F_Stop=str2num(answer{2,:});
if  F_Start==F_Stop  ||  F_Start>F_Stop  ||  F_Start<0  ||  F_Start>64  ||
F_Stop<0 || F_Stop>65
   input_checking=0;
else
   input_checking=1;
end

if input_checking==0
   msg4=   msgbox('Invalid Frequency Range'...
        , 'Warning');
while(1)
if ishandle(msg4)==0
   break;
end
pause(1);
end

end
end
F_range=F_Start:1:F_Stop;
Rows=Best_Num_Train*NumsofFolders
Collumns= numofChannels*length(F_range)
Data=zeros(Rows,Collumns);
Data_row=zeros(1,Collumns);
TakingDatamodel=uigetfile('*.mdl')
TakingDatamodel=TakingDatamodel(1:length(TakingDatamodel)-4)
choice  =  questdlg('What  type  of  Feedback  you  want  to  choose
?','Feedback','Correlation     Coefficent','Reference     Power     and
Correlation Coefficent','Reference Power and Correlation Coefficent');
switch choice
    case 'Correlation Coefficent'
          Feedback=0;
    case 'Reference Power and Correlation Coefficent'
          Feedback=1;
end
if Feedback==1
   msg=msgbox(['Collecting Reference Data for',num2str(20),'s'])
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
   sim(TakingDatamodel,20)
   S=load('res.mat');
   A=S.ans;
   X=A(3:16,1:128*TimeTrain)
   Fs=128;
   N=128*TimeTrain;
 for k=1:1:14
         original_data=X;
         window=hamming(N);
         psd_data(k,:)                                        =
2*periodogram(original_data(k,:),window,0:Fs/2,Fs);
 end
   psd_data_ref=psd_data(choosing_vector,F_range)
   Temp=psd_data_ref';
   Data_Ref=Temp(:)'
 msg=msgbox('Reference taken')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
else
    Data_Ref=0;
end
 for i=0:NumsofFolders-1
 for j=0:1:Best_Num_Train-1

load_file=[FileDir{i+1},'\','PSD_data_of_',Classes{i+1},int2str(j),'.m
at']
   S=load(load_file)
   X=S.Taken_data(choosing_vector',F_range)

   for k=0:1:numofChannels-1

Data_row(length(F_range)*k+1:length(F_range)*k+length(F_range))=X(k+1,
:);
   end

  Data(Best_Num_Train*i+j+1,:)=Data_row;

  end
 end
timedata_all=zeros(Rows,Collumns)
timedata_all(1:Rows,:)=Data;
%Normalizing
minarray=min(timedata_all,[],1);
 ratio=spdiags(1./(max(timedata_all,[],1)-
min(timedata_all,[],1))',0,size(timedata_all,2),size(timedata_all,2));
 timedata_all              =              (timedata_all              -
repmat(minarray,size(timedata_all,1),1))*ratio;
Timespan=[0 TimeTrain]
```

```
check=0;
if M==0
for i = 1:NumsofFolders
Correct=0;
Choice=i;
cccheck=0;
Loop=0;
msg=msgbox(['Collecting  Reference  Data  for  Correlation  Calculation
of',Classes{i},'s'])
while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
end
while(cccheck~=1)
   sim(TakingDatamodel,TimeTrain)
   S=load('res.mat');
   A=S.ans;
    X=A(3:16,1:128*TimeTrain)
   Fs=128;
   N=128*TimeTrain;
 for k=1:1:14
         original_data=X;
         window=hamming(N);
         psd_data(k,:)                                          =
2*periodogram(original_data(k,:),window,0:Fs/2,Fs);
 end
   psd_data_ref=psd_data(choosing_vector,F_range)
   Temp=psd_data_ref';
   Corr_Ref=Temp(:)';
   Corr_Ref=(Corr_Ref-minarray)*ratio;
   [Result_Corr,~,p_train]    = svmpredict(Choice,Corr_Ref, RunModel,'-
b 0');
  if Result_Corr==i
    cccheck=1;
    Time_corr=X;
   msg=msgbox('Correlation Reference taken')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
  else
    cccheck=0
   msg=msgbox('Try again')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
  end
end


   msg=msgbox(['ready                    to                    practice
',Classes{i},'for',num2str(Points),'times'])
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end

while Correct<Points
Loop=Loop+1;
msg=msgbox(['Prepare to take data ')
while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
end
check=0;
while check~=1
sim(TakingDatamodel,Timespan)
S=load('res.mat');
A=S.ans;
X=A(3:16,1:128*TimeTrain);
Timedata=X;
Fs=128;
N=128*TimeTrain;
 for k=1:1:14
         original_data=X;
         window=hamming(N);
         psd_data(k,:)                                        =
2*periodogram(original_data(k,:),window,0:Fs/2,Fs);
 end
psd_data_use=psd_data(choosing_vector,F_range);
Temp=psd_data_use';
Predict_data=Temp(:)';
corr_coeff=corrcoef(Timedata(:),Time_corr(:))
if mean(Predict_data)<=mean(Data_Ref) || corr_coeff(1,2)<0.80
   check=0;
   msg=msgbox('Focus please');
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
else
```

```
   check=1;
end
end
Predict_data=(Predict_data-minarray)*ratio;
[Result,~,p_train]      =  svmpredict(Choice,Predict_data,  RunModel,'-b
0');
if Result==Choice
   msg=msgbox(['Your thought:',Classes(Result)],'Correct !')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
   Correct=Correct+1;
else
   msg=msgbox(['Your thought:',Classes(Result)],'Incorrect !')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
end
end
   msg=msgbox(['Your   Result   For   ',Classes{i},'   Practice   is:
',num2str(Correct/Loop/NumsofFolders*100)],'Practice Result!')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
end
else
choice   =   questdlg('What   type   of   DataType   you   want   to   choose
?','DataType','Alphabetical','Directional','Directional');
switch choice
    case 'Alphabetical'
          Type=0;
    case 'Directional'
          Type=1;
end
Timedata_Corr=zeros(NumsofFolders,numofChannels*128*TimeTrain);

for i = 1:NumsofFolders
    cccheck=0;
    msg=msgbox(['Collecting Reference Data for Correlation Calculation
of',Classes{i},'s'])
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
    while(cccheck~=1)
   sim(TakingDatamodel,TimeTrain)
   S=load('res.mat');
   A=S.ans;
    X=A(3:16,1:128*TimeTrain)
   Fs=128;
   N=128*TimeTrain;
 for k=1:1:14
         original_data=X;
         window=hamming(N);
         psd_data(k,:)                                               =
2*periodogram(original_data(k,:),window,0:Fs/2,Fs);
 end
   psd_data_ref=psd_data(choosing_vector,F_range)
   Temp=psd_data_ref';
   Corr_Ref=Temp(:)';
   Corr_Ref=(Corr_Ref-minarray)*ratio;
   [Result_Corr,~,p_train]      =  svmpredict(i,Corr_Ref,  RunModel,'-b
0');
  if Result_Corr==i
    cccheck=1;
Timedata_Corr(i,:)=X(:);
   msg=msgbox('Correlation Reference taken')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
  else
    cccheck=0
   msg=msgbox('Try again')
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
  end
end
end
while(1)
msg=msgbox('Prepare to take data ')
while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
end
```

```
check=0;
while check~=1
sim(TakingDatamodel,Timespan)
S=load('res.mat');
A=S.ans;
X=A(3:16,1:128*TimeTrain);
Timedata=X;
Fs=128;
N=128*TimeTrain;
 for k=1:1:14
         original_data=X;
         window=hamming(N);
         psd_data(k,:)                                                   =
2*periodogram(original_data(k,:),window,0:Fs/2,Fs);
 end
psd_data_use=psd_data(choosing_vector,F_range);
Temp=psd_data_use';
Predict_data=Temp(:)';

Predict_data=(Predict_data-minarray)*ratio;
[Result,~,p_train]   = svmpredict(1,Predict_data, RunModel,'-b 0');
Time_corr=Timedata_Corr(Result,:)
corr_coeff=corrcoef(Timedata(:),Time_corr)
if mean(Predict_data)<=mean(Data_Ref) || corr_coeff(1,2)<0.85
   check=0;
   msg=msgbox('Focus please');
   while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
   end
else
if Type==0
   if Result==1
       string='A';
   else
       string='B';
   end
else
   if Result==1
       string='l';
   elseif Result==2
       string='r';
   elseif Result==3
       string='f';
   elseif Result==4
       string='b';
   end
end
    run ;
    fprintf(s,string)
    display(string)
    pause(2);
    if Type==1
      string='s';
    fprintf(s,string)
    display(string)

    end
    fclose(instrfind);
msg=msgbox('Prepare to take data ')
while(1)
   if ishandle(msg)==0
   break;
   end
   pause(1);
end
end
end
end
end
```

## A.5 ExampleBrainDataMaker.m

```
input_checking=0;
while input_checking==0
prompt = {'Name of new data','Number of Example items'...
    ,'Mean','deviation'};
dlg_title = 'Input';
num_lines = 1;
def = {'Randdata_100mean','1000','100','5'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
input_checking=1;
end
cur_dir=pwd;
des_dir=[cur_dir,'/',answer{1}]
mkdir(des_dir)
a = str2num(answer{4});
b = str2num(answer{3});
c = str2num(answer{2})
for i=0:1:c-1
   rng(i,'twister');
   Taken_data=a.*randn(14,64)+b;
   new_name=['PSD_data_of_',answer{1},int2str(i),'.mat']
   newdes=[des_dir,'/',new_name]
   save(newdes,'Taken_data')
end
```

## A.6 Serial mode function and run.m

### a) run.m

```
comPort='COM5';
[s,flag]=setupSerial(comPort);
```

### b) setupSerial.m

```
function[s,flag]=setupSerial(comPort)
flag=1;
s = serial(comPort);
set(s,'DataBits',8);
set(s,'Stopbits',1);
set(s,'BaudRate',9600);
set(s,'Parity','none');
fopen(s);

a='b';
while(a~='a')
    a=fread(s,1,'uchar');
end
if(a=='a')
    disp('serial read');
end
   fprintf(s,'%c','a');
   fscanf(s,'%u');
end
```

# Appendix B: Arduino code

## B.1 Transmitter. ino

```
char myData;
void setup(){
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  Serial.begin(9600);
  Serial.println('a');
  char a= 'b';
while (a !='a')
{
 a= Serial.read();
}
}
void loop(){
  if(Serial.available() > 0){
    myData = Serial.read();
             if(myData=='f')
        {
          digitalWrite(2, LOW);
          digitalWrite(3, LOW);
          digitalWrite(4, HIGH);
        }
        else if(myData=='b')
        {
          digitalWrite(2, LOW);
          digitalWrite(3, HIGH);
          digitalWrite(4, LOW);
        }
        else if(myData=='l')
        {
          digitalWrite(2, LOW);
          digitalWrite(3, HIGH);
          digitalWrite(4, HIGH);
        }
        else if(myData=='r')
        {
          digitalWrite(2, HIGH);
          digitalWrite(3, LOW);
          digitalWrite(4, LOW);
        }
        else if(myData=='A')
        {
          digitalWrite(2, HIGH);
          digitalWrite(3, LOW);
          digitalWrite(4, HIGH);
        }
        else if(myData=='B')
        {
          digitalWrite(2, HIGH);
          digitalWrite(3, HIGH);
          digitalWrite(4, LOW);
        }
        Serial.print(myData);
   }
}
```

## B.2 Receiver.ino

```
#include <Servo.h>
#include <LiquidCrystal.h>
int motorL=8;
int motorR=9;
Servo myservoR;
Servo myservoL;
LiquidCrystal lcd(12,11,5,4,3,2);
char myData;
void setup(){
  Serial.begin(9600);
  pinMode(motorL, OUTPUT);
  pinMode(motorR, OUTPUT);
  myservoR.attach(9);
  myservoL.attach(8);
  lcd.begin(16,2);
  lcd.clear();
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
}
  void loop(){
  int a=A0;
  int b=A1;
  int c=A2;
  if (a==0&&b==0&&c==1)
  {
          lcd.print("FORWARD");
    myservoL.writeMicroseconds(1000);
    myservoR.writeMicroseconds(2000);
    delay(100);
    myservoL.writeMicroseconds(1500);
    myservoR.writeMicroseconds(1500);
  }
    if (a==0&&b==1&&c==0)
  {
         lcd.print("BACkWARD");
    myservoL.writeMicroseconds(2000);
    myservoR.writeMicroseconds(1000);
    delay(100);
    myservoL.writeMicroseconds(1500);
    myservoR.writeMicroseconds(1500);
  }
    if (a==0&&b==1&&c==1)
  {
          lcd.print("LEFT");
    myservoL.writeMicroseconds(1000);
    myservoR.writeMicroseconds(1000);
  delay(700);
  myservoL.writeMicroseconds(1500);
  myservoR.writeMicroseconds(1500);
}
  if (a==1&&b==0&&c==0)
{
         lcd.print("RIGHT");
  myservoL.writeMicroseconds(2000);
  myservoR.writeMicroseconds(2000);
  delay(700);
  myservoL.writeMicroseconds(1500);
  myservoR.writeMicroseconds(1500);
}
  if (a==1&&b==0&&c==1)
{
   lcd.print("A");
}
  if (a==1&&b==1&&c==0)
{
   lcd.print("B");
}
    if (a==0&&b==0&&c==0)
{
   lcd.print("STOP");
}

}
```