Palestine Polytechnic University

College of IT and Computer Engineering

Computer Systems Engineering

Graduation Project Report

Vocalizing Arabic Sign Language

Group:

Samah Zaro, Lubna Hashlamoun, Itimad Smairat

Supervisor:

Eng. Yousef A. Salah

Supervisor:

Dr. Hashem Tamimi

Co-Supervisor:

Eng. Bashar M. Altakrouri

"University of Luebeck, Germany"

2013/2014

الخليل – فلسطين
كليه تكنولوجيا المعلومات وهندسة الحاسوب
دائرة هندسة وعلوم الحاسوب

اسم المشروع
Vocalizing Arabic Sign Language

اسماء الطلبة:
سماح الزرو
لبنى الهشلمون
اعتماد سميرات

بناء على نظام كليه كليه تكنولوجيا المعلومات وهندسة الحاسوب وإشراف ومتابعة المشرفين على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع لاستيفاء متطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب

توقيع المشرف

_____

توقيع اللجنة الممتحنة

_____ _____

_____

توقيع رئيس الدائرة

_____

Dedication

To our families and our parents whose constant inspiration and encouragement drives us to become all that we can be.

Our appreciation goes to our teachers, and all the staff of the faculty of engineering.

We give thanks for them for being our mentors, and our friends; it is indeed a merit for us to be their students.

Samah Zaro

Itemad Smairat

Lubna Hashlamoun

Abstract

Social Cohesion is a main concern in Arabic culture; this concern has led to the idea of this project, which can help in better integrating for the mute and deaf people in the Arabic societies.

The aim of this project is being achieved by designing and implementing a system able to recognize Arabic sign language and translate it into text/voice represented in Arabic language. This system can perform immediate translation between mute and deaf people and hearing people.

The system reads the user's signs through a wearable sensory Data-Glove, then a Microcontroller reads the input data and performs recognition operations on them using some machine learning techniques, such as HMM (Hidden Markov Model). After that, the resultant text is displayed on a shielded LCD (Liquid Crystal Display) and, at the same time, converted into its corresponding voice through a TTS (Text to Speech Synthesizer) which, then, transmitted it to a speaker.

الملخص

التواصل الاجتماعي من أهم العناصر في الثقافة العربية؛والاهتمام بهذا العنصر هو الدافع لفكرة المشروع التي تهدف الى دمج الصم والبكم في المجتمعات العربية.

ويجري تحقيق الهدف من هذا المشروع من خلال تصميم وتنفيذ نظام عربي قادرا على التعرف على لغة الإشارة وترجمتها إلى نص / صوت الممثلة في اللغة العربية. يمكن لهذا النظام أن يقوم بالترجمة الفورية مابين الأفراد الصم وباقي أفراد المجتمع.

يقوم هذا النظام بقراءة اشارات المستخدم من خلال قفاز يحتوي على مستشعرات,يتم معالجة هذه البيانات من خلال المتحكم الدقيق مستخدما خوارزميات مثل نموذج ماركوف المخفي (HMM) ثم تظهر النتيجة على شاشة صغيرة (LCD) على شكل نص ,وفي الوقت نفسه,يتم تحويل النص الى الصوت المقابل له من خلال قطعة الكترونية تسمى (TTS) ثم يظهر الصوت من خلال السماعة.

Acknowledgment

We would like to express our gratitude to all people who supported us during the progress of this work. Especially we would like to thank:

Our supervisors, Eng. Yousef Salah, Dr. Hashem tamimi and Eng. Bashar Altakrouri, for their suggestions, the grateful comments and revisions of this document,

The Palestine Polytechnic University and College of Information Technology and Computer Engineering for giving us the opportunity to live this real experience,

Kayed Rajabi, Signal Language Interpreter at World Deaf Club Society- Hebron, for introducing us to deaf world and for the annotations of the sign language.

And finally our great parents for supporting us in every way.

Samah Zaro
Lubna Hashlamoun
Itimad Smerat

Table of Contents

List of Figures

List of Tables

Abbreviation and Acronyms

Abbreviation	Acronyms
ARSL	Arabic Sign Language
ASL	American Sign Language
HW	Hardware
SW	Software
PC	Personal Computer
MC	Microcontroller
AI	Artificial Intelligence
IC	Integrated Circuit
LCD	Liquid Crystal Display
GLCD	Graphical Liquid Crystal Display
USB	Universal Serial Bus
TTL	Transistor Transistor Logic
PPU	Palestine Polytechnic University
SLR	Sign Language Recognition
SPSL	Static Posture, Static hand Location
DPSL	Dynamic Posture, Static hand Location
SPDL	Static Posture, Dynamic hand Location
DPDL	Dynamic Posture, Dynamic hand Location
HMM	Hidden Markov Model
TTS	Text to Speech Synthesizer
AC/DC	Alternating Current/Direct Current
MHz	Mega Hertz
DB	Database
MicroSD	Micro Secure Digital
HCI	Human Computer Interaction
RBF	Radial Basis Function
OS	Operating System

Chapter One

Introduction

1.1 Problem Overview

1.2 Project Goals

1.3 Literature Review

1.4 Work Methodology

1.5 Feasibility Study

1.6 Tasks and Time Plan

1.7 Risk Identification and Planning

1.8 Report Overview

1.1 Problem Overview

Arabic Sign Language (ARSL) is the main mode of communication for most deaf people in the Arab world. However, most people outside the deaf community do not understand this special language!, for that, they live in isolation. This isolation prevents this large segment of the community from proper socialization, education, and aspiration to career growth. It hinders their talents and skills in benefiting the society.

This serious communication problem affects deaf people's lives and relationships negatively. Deaf people usually communicate with hearing people either through interpreters or text writing. Although interpreters can help the communication between deaf persons and hearing persons, they are often expensive and their use leads to loss of independence and privacy. Communication by writing is used by many deaf people with hearing people, but it is very inconvenient while walking, standing at a distance, or when more than two people are in a conversation.

It's proposed to build an easy-portable system which can automatically translate Sign Language into speech, in parallel with its corresponding text, using machine learning software algorithms for real environment. so that it will be much easier for deaf people to communicate with others.

1.2 Project Goals and Objectives

Building a system that have the ability to recognize continuous Arabic sign language gestures then translate them into their corresponding speech. The system must include the following objectives:

1. To promote applied projects relevant to the Arab World.
2. Implement real-time, somewhat, recognition software algorithms.
3. Make the recognition as accurate as possible.
4. To attempt to minimize cost and complexity.
5. Create dictionary of various known signs and storing them into database.

6. Gather samples, for each sign, from many different signers to achieve user-independent system.

1.3 Literature Review

The following related projects and papers were researched:

American Sign Language isolated word recognition using a sensory glove and motion tracker[2]

Describes two different American Sign Language (ASL) word recognition systems developed using artificial neural networks to translate the ASL words into English. The systems use a sensory glove, CybergloveTM, and a Flock of Birdss 3-D motion tracker to extract the gesture features. The data from these devices were processed by a velocity network, a feature extraction module, and a word recognition network in both systems.

The systems were trained and tested for single and multiple users for a vocabulary of 60 ASL words. Test results show that the accuracy of recognition is 92% and 95%.

Layered architecture for real time sign recognition[3]

This research presents a real time sign recognition architecture including both: gesture and movement recognition, using data gloves (5DT DataGlove 14 Ultra) and accelerometers (MTx) technologies. Due to the real time nature of the problem, the proposed approach works in two different tiers, the segmentation tier and the classification tier.

Researchers encounter a problem of difficulty in distinguishing very similar signs, due to problems such as glove calibration or sensor noise. and they see that recognition rates can be improved by creation of more complex classifiers.

Enable Talk Project[5]

a model of sensory gloves that includes a modern microcontroller, 15 flex sensors, accelerometer, gyroscope, and a compass in order to define the position of the glove in space, a Bluetooth module for data transmission from the gloves to a smartphone mobile device and a USB-port for the synchronization with the PC and for charging the Li-ion battery that provides power.

1.4 Work methodology

This section presents the work methodology in which this project will be processed with, it also defines the general process and characteristics of the work plan.

- **Development Process:**

Using the "Component-Based" Development Process model, the software will be defined as a set of modules that address specific and small well-defined issues.

After all of the modules required for the project are fully developed, the integration process begins to form the final product.

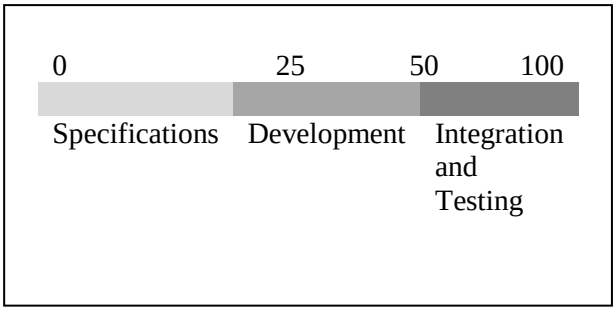


Figure 1.1 Component Based Developement

The main components required for software development are:

- **Hardware Driver module:**

  A TTL to USB cable which permits to immediately connect the data glove to a USB port of a PC or microcontroller.

- **Graphical LCD display module:**
  To show the output words of this system for the mute to be able to verify whether the signs translated correctly or not**.**

- **High Definition Speech Synthesizer module:**
  The Text-To-Speech processor converts ASCII text directly to voice with unlimited vocabulary.

- **Database module:**
  for saving the Models of each sign.

As for Hardware development, a "Spiral" Development Model will be used, which means that the development process will first address the basic requirements, then go through evolution by redefining the problem and upgrading the hardware gradually to get the final product.

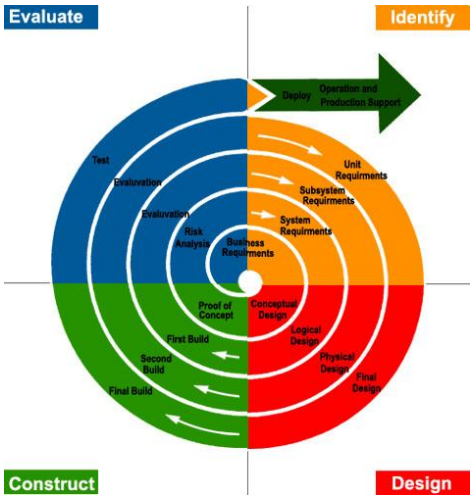


Figure 1.2 Spiral Model

An abstraction of the system's hardware components is the following:

- **DataGlove:** it provides the signs that would then be interpreted to associated Arabic meanings by the microcontroller.
- **Microcontroller:** it will run the software that performs the recognition of signs.
- **ASCII Text_to_Speech Synthesizer:** it converts ASCII text to the corresponding voice.
- **Graphical LCD Screen:** to display output as text.
- **DataBase:** to store the model of each sign.

- Proposed Technology:

The project will use the following list of technologies, characterized into two categories, Hardware and Software:

Hardware Technologies:

- Programmable Microcontroller (Arduino Mega 2560).
- TTL to USB adapter cable (VHand USB Cable).

Software Technologies:

- Visual C++, as the main language for software development.
- Matlab programming language for software development.
- Arduino software for microcontroller programming.
- Oracle database for sign dictionary.

1.5 Feasibility Study

The budget of the project is listed and characterized as following:

- **Hardware Components:**

The project estimated costs were calculated and found to be around 450$. Table 1.1 provides a detailed list of the components needed for the project

Table 1.1 Hardware Budget

Components	**Price($)**	**Quantity**	**Total($)**	**Description**
DG5 VHand 2.0 OEM	0 (PPU)	1	0	Right-hand data glove, with Flex Sensors and Accelerometers.
TTS-EM-HD2	580	1	580	ASCII Text to Speech Synthesizer Systems
Arduino Mega 2560	100	1	100	microcontroller board based on the ATmega2560
Rechargeable Battery	15	1	15	+5 +12 Power Supply
LM7805CT regulator	2	1	2	5V regulator
AC/DC adapter	10	1	10	220V/60Hz to 9V output
Graphical LCD Screen shield	60	1	60	16X250 LCD Display compatible with Arduino

Speaker 400mw (8ohm)	5	1	5	Connected to TTS module output
microSD shield	20	1	20	External memory to store sign language dictionary/database
1GB micro SD	8	1	8	
Total			**800$**	

- **Software Resources:**

The software resources are summarized in Table 1.2 as follows:

Table 1.2 Software Resources

Software	**Num**	**Costs($)**	**Notes**
Microsoft Visual Studio 2008	3	Free (PPU)	Professional Edition
Windows 7	3	70$	Ultimate Edition
Microsoft Office 2007	3	60$	Editors and Presentation
Microsoft Office Visio 2007	3	20$	Drawing tool
Matlab 2010	3	Free (PPU)	Machine Learning programming Tool
Arduino programming kit	1	Free	Microcontroller driver
Total		**150$**	

- **Human Resources:**

The human resources could be summarized as follows:

Table 1.3 Human Resources

Human resources	**Num**	**Fee**	**Notes**
Working Team	3	Free	
Samples providers	10	Free	
Total		None	

1.6 Task s and Time Plan

The work is divided into number of tasks, distributed at the overall time of the project, shown in the table bellow (see Table 1.4):

Table 1.4 Tasks Plan

Task No.	Task Name	Time (weeks)	Parallelism	Dependency
T1	**Information gathering**	6		
T1.1	• Previous studies	4		
T1.2	• Visiting deaf institutes	2		
T2	**Develop the working method**	1		T1
T3	**Project analysis**	2		T2
T3.1	• Time scheduling	1	T3.3,T3.4	
T3.2	• Determine the HW and SW needed	1		
T3.3	• Determine the cost of the project	1	T3.1,T3.4	
T3.4	Risk management	1	T3.1,T3.3	
T4	**Project analysis**	4		T3
T4.1	• Activity diag. class diag., state diag.	2		
T4.2	• HW components analysis	2	T4.3,T4.4	
T4.3	• training phase methods analysis	2	T4.2,T4.4	
T4.4	• sign recognition phase algorithms analysis	2	T4.2,T4.3	
T5	**Project design**	4		T4
T5.1	• Design block diagram	0.5		
T5.2	• Design the training and recognition phases SW	2		
T5.3	• Design the hardware schematic diagram	1.5		
T6	**Learning Arabic Sign Language**	12	T7	T4,T5
T7	**Hardware design and implementation**	12	T6	
T7.1	• System specification design	3	T7.2	
T7.2	• Collect the HW components	5	T7.1	
T7.3	• Implement the sensors and accelerometers	4		T7.1,T7.2
T7.4	• Connect the microcontrollers with peripherals	3		T7.3
T8	**Software implementation and execution**	16		T4,T5,T7

T8.1	• Collect the samples for training_ testing phases	2		
T8.2	• Perform the train phase using PC ,testing accuracy	8		
T8.3	• Write the MC program	2		
T8.4	• Test the system.	4		

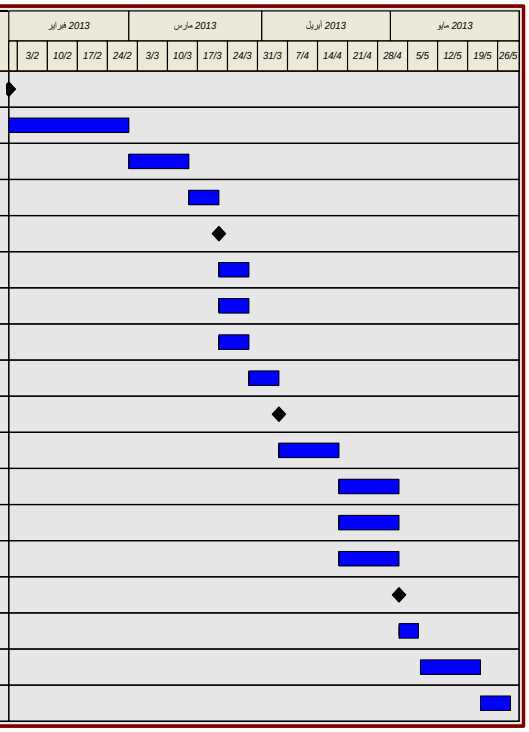
The preceding tasks are divided on three stages: Spring semester, Summer semester, and Fall semester.

Spring Semester:

In the Spring semester stage, an initial understanding of the problem was reached and an understanding of the system was achieved by gathering related information, develop the working strategy. After that the project planning, analysis and design were performed.

Table 1.5 Stage 1 - Spring Semester Schedule Chart

ID	Task Name	Start	Finish	Duration
1	T1	01/02/13	01/02/13	0w
2	T1.1	01/02/13	28/02/13	4w
3	T1.2	01/03/13	14/03/13	2w
4	T2	15/03/13	21/03/13	1w
5	T3	22/03/13	22/03/13	0w
6	T3.1	22/03/13	28/03/13	1w
7	T3.2	22/03/13	28/03/13	1w
8	T3.3	22/03/13	28/03/13	1w
9	T3.4	29/03/13	04/04/13	1w
10	T4	05/04/13	05/04/13	0w
11	T4.1	05/04/13	18/04/13	2w
12	T4.2	19/04/13	02/05/13	2w
13	T4.3	19/04/13	02/05/13	2w
14	T4.4	19/04/13	02/05/13	2w
15	T5	03/05/13	03/05/13	0w
16	T5.1	03/05/13	07/05/13	2.5d
17	T5.2	08/05/13	21/05/13	2w
18	T5.3	22/05/13	28/05/13	1w



Summer Semester:

In this stage the focus on learning Arabic sign language, to be familiar with its characteristics, also obtaining the hardware design and implementation to be use in the final stage.

Table 1.6 Stage 2 - Summer Semester Schedule Chart

ID	Task Name	Start	Finish	Duration
1	T6	03/06/13	25/08/13	12w
2	T7	03/06/13	03/06/13	0w
3	T7.1	03/06/13	23/06/13	3w
4	T7.2	03/06/13	07/07/13	5w
5	T7.3	08/07/13	04/08/13	4w
6	T7.4	05/08/13	25/08/13	3w

Fall Semester:

In this stage, the system begins to shape in its final form, the samples are collected, and training phase finalized, after the wanted accuracy being obtained the final software is written on MC then the system testing is performed.

Table 1.7 Stage 3 - Full Semester Schedule Chart

ID	Task Name	Start	Finish	Duration
1	T8	02/09/13	02/09/13	0w
2	T8.1	02/09/13	13/09/13	2w
3	T8.2	16/09/13	08/11/13	8w
4	T8.3	11/11/13	22/11/13	2w
5	T8.4	25/11/13	20/12/13	4w

1.7 Risk Management

There are some possible risks that may occur in our project in both hardware and software .in this section will briefly describe those risks and show their management.

The following table lists the types of risk which may impact the project development process, showing the risk and its type – Technology, people, organizational, tools, requirements, and estimation – and a short description about it.

Table 1.8 Risks Identification and Characterization

Type of Risk	**Possible Risks**	
Technology	**R1**	Hardware which is essential for the project may not be delivered on schedule.
	R2	Malfunction of hardware parts (IC's, Microcontroller, Sensors).
	R3	The inability to build an accurate classification in the system.
Requirements	**R4**	There may be changing the requirements than anticipated.
Estimations	**R5**	The time required developing the software and the hardware is underestimated.
	R6	The size of the software and the hardware is underestimated.
	R7	The Budget is not sufficient.
Tools	**R8**	Code of the software is damaged or deleted suddenly.
	R9	Malfunction in the Team's IC's Programmer.
Organizational	**R10**	Time of delivery of the project changed.
People	**R11**	Illness of one or more team member.

The following table lists the probability and effect of each risk:

Table 1.9 Risk Probability and Analysis

Risk ID	**Risk**	**Probability**	**Effects**
R1	Hardware which is essential for the project may not be delivered on schedule.	Moderate	Catastrophic
R2	Malfunction of hardware parts (IC's, Microcontroller, Sensors).	Low	Serious
R3	Building the accurate classifier.	Moderate	Serious
R4	There may be a larger number of changes to the requirements than anticipated.	Low	Tolerable
R5	The time required developing the software and Hardware is underestimated.	Low	Tolerable
R6	The size of the software and Hardware is underestimated.	Low	Tolerable
R7	The Budget is not sufficient.	Moderate	Catastrophic
R8	Code of the software is damaged or deleted suddenly.	Very low	Catastrophic
R9	Malfunction in the Team's IC's Programmer.	Moderate	Tolerable

R10	Time of delivery of the project changed.	Very low	Tolerable
R11	Illness of one or more team member.	Low	Tolerable

The following table shows the strategies that address each risk in this project:

Table 1.10 Risk Management Strategies

Risk ID	**Strategy**
R1	Identify the needed hardware as fast as possible and order them.
R2	Continue the project development and simulate the network operation.
R3	Simplifying the needed machine learning algorithms and total required recognition tasks.
R4	Derive traceability information to assess requirements change impact, and maximize information hiding in the design.
R5	Understand almost the exact time needed in developing the software and hardware.
R6	Understand almost the exact size for the software and hardware.
R7	Borrow the needed money.
R8	Always save backup copies of the software (on flash, CD's, hard disks)
R9	Depend on another team's Programmer, or buy a new one.
R10	No strategy
R11	Reorganize team so that there is more overlap in the work

1.8 Report overview

The following chapters will be included in this report:

Chapter One - Introduction: An introduction to the project, describing the problem, project idea, and its aims and the software engineering methodology followed.

Chapter Two – Theoretical Background : A theoretical background study on sign language, Arabic sign language and the recognition process, components and technologies needed for this project, description for HMM and its concepts, and explanation of its working methods.

Chapter Three – Conceptual Design and Analysis: This chapter describes the system in its abstract formula. It describes the project design options for its two categories; hardware and software, general block diagrams, system modeling, and how the system works.

Chapter Four – Detailed System Design: A detailed description about the project phases, subsystem design, and overall system design and user system interface.

Chapter Five – System Implementation and Testing: This chapter includes the testing and implementation for the system units and for the integrated system.

Chapter Six – Experiments and Results: This chapter includes the data set, methods, and final results.

Chapter Seven –Conclusions and Future Works: This chapter contains what should be done in the future to enhance the system.

Chapter Two

Theoretical Background

2.1 Sign Language

2.2 Human-Computer Interaction

2.3 Data Acquisition Approaches

2.4 Hardware Components.

2.5 Machine Learning

2.6 Hidden Markov Model.

2.1 Sign Language

Deaf persons use a sign language for communication with each other. Sign languages are self-contained languages with their own structure and grammar. They have developed in a natural way in deaf communities and independent from spoken languages. Therefore Sign language is not universal. Instead, a different type of sign language has evolved in every deaf community. Sign languages can be grouped into national languages, e.g. American Sign Language (ASL) and Arabic Sign Language (ARSL), which can have several dialects again.

A sign language usually provides signs for whole words. It also provides finger spelling, which depends on using signs of letters to perform words that don't have a corresponding sign in that sign language. So, although sentences can be made using the signs for letters performing with signs of words is faster.

In contrast to spoken languages, which use acoustic signals, sign language uses visual communication. Sign can be divided into two groups: manual components and non-manual components [6].

- **Manual Components**

There are three manual components: hand configuration, place of articulation and movement. These components were extended by hand orientation as fourth component.

- **Hand configuration:** The hand shape, i.e. the positions of the fingers, is called hand configuration. Due to the high degree of freedom of the fingers, a lot of hand shapes are possible, but not all of them are used in sign language. Which configurations are used, differs from language to language. However, six basic configurations can be found in most languages (see Figure 2.1). Each language defines additional hand shapes.

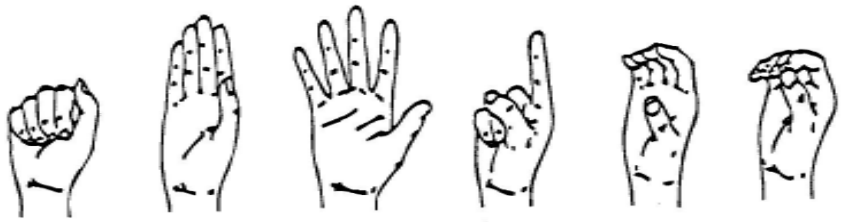


Figure 2.1 Basic hand shapes

- **Place of articulation:** The location in signing space, where the sign takes place, is referred to as place of articulation. The signing space is the area in front of the signer in which the signs are performed. It includes the face and the upper part of the body. Places of articulation can be grouped into lexical meaningful areas. The area in front of the face is used often, because signing persons look in the eyes of each other not on the hands while they are “listening”. The size of a sign can change depending on the “loudness”. Whispered signs are smaller than shouted ones.

- **Hand movement:** Hand movement refers to the trajectory of one or both hands in space. Some signs are performed with only one hand, called the dominant hand. In signs performed with two hands the non-dominant hand performs a similar movement as the dominant hand or has a supporting role. Deaf people recognize many signs already using only the hand movement.

- **Hand orientation:** The hand orientation is defined by the situation of wrist, elbow, and shoulder. Signs with equal hand configuration can differ in hand orientation. Differences in hand orientation become apparent by different views of the palm.

A pair of signs can be found for each component, such that both signs differ only in this component. The combination of manual components is subject to (national) rules. Not all combinations are allowed, similar to spoken languages, where not all possible phoneme combinations are used.

- **Non-Manual Components**

Non-manual components include facial expression, eye gaze, head tilt and body posture. They are used for grammatical purposes. For example, facial expression and head position can be used to indicate questions, negations, and subordinate clauses.

So, generally speaking, sign language is a communication skill that uses gestures instead of sound to convey meaning simultaneously combining hand shapes, orientations, and movement of the hands, arms or body and facial expressions to express fluidly a speaker's thoughts. Signs are used to deliver words and sentences to audience.

2.1.1 **Arabic Sign Language**

Arabic Sign Language had very limited research and projects. There has been no serious attention for this sign recognition. The sign language chosen for this project is the Arabic Sign Language.

Arabic Sign Languages (ARSLs) have been recently recognized and documented. Many efforts have been made to establish the sign language used in individual countries, including Jordan, Egypt, Libya, and the Gulf States, by trying to standardize the language and spread it among members of the deaf community and those concerned [7].

Two main books are published for Arabic sign language; namely The Arabic Sign Dictionary of Gestures for Deaf published by the Arabic League in Tunisia in 2001, and The Arabic Dictionary of Gestures for the Deaf II which was a result of collaboration between The Arab States League, the Arab Union for Deaf, and the Arab Organization for Culture and Sciences along with the support of the Arab deaf.

Arabic sign languages are not particularly different from other known sign languages, such as ASL. In fact, the Arabic varieties in use have undergone some lexical influence from other sign languages. ARSLs are basically manual languages

made from cheremes that involve the three recognized elements: configuration of hands (hand shape), placement/space (position of hand in relation to body), and movement (directions and contacts within space).1 In addition to these manual shapes, ARSLs make use of other non-manual features, like those of the face, mouth, and tongue (Figure 2.2).

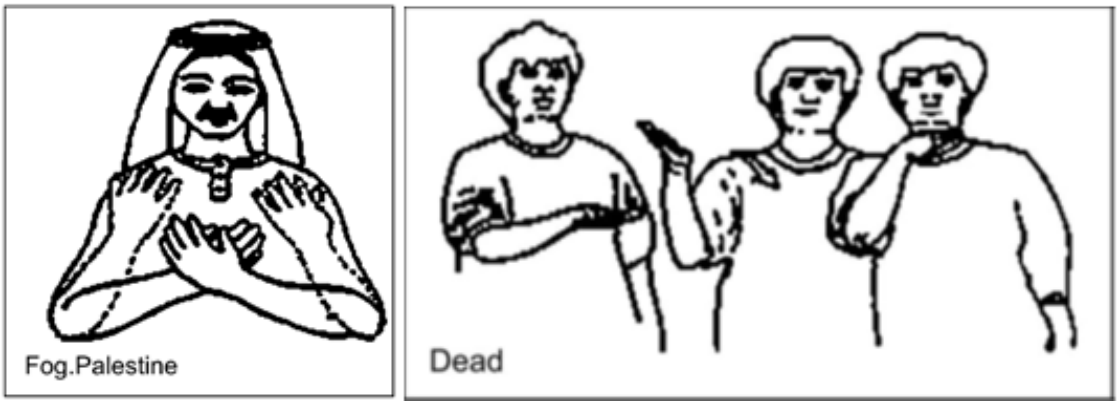


Figure 2.2 Using non-manual features. (Dead: lips spread together with hand movement; fog: eyes slightly closed.)

Arabic does have a finger spelling system that has become relatively standardized among the Deaf of different Arabic-speaking countries, although their Sign Languages differ (see Figure2.3).

Figure 2.3 Arabic Sign Alphabets

2.1.2**Sign Language Recognition**

Sign language is a structured form of visual communication, comparable to spoken languages in complexity and expressiveness. In most cases, the direct communication between deaf and hearing people is difficult, because either a sign language interpreter has to mediate or written communication has to be used.

Recent progress in image processing, speech recognition, machine translation gives a prospect of using computers to bridge this communication barrier by automatically translating sign language into spoken language and vice versa. Automatic sign language recognition plays an important role in this– translation system. Systems for the automatic recognition of sign language aim at recognizing single signs or sequences of signs in continuous signing. The translation of the

recognized signs to a spoken language can be done by an additional translation system.

Approaches to sign language recognition can be divided into isolated sign recognition and continuous sign language recognition. Continuous signs are sentences or phrases formed by sequencing a series of isolated signs. Each category can be further classified into signer-dependent and signer-independent according to the sensitivity to the signer. Also one may classify SLR systems as either glove-based (if the system relies on electromechanical devices for data collection), or none glove-based (if bare hands are used). These design options will be discussed later in the forth chapter (Chapter 4 _ Background).

- **Sign Language Recognition Applications**

Automatic sign language recognition can be applied, for example, to the following tasks:

- **Translation:** Translation systems can aid the communication between deaf and hearing people.

- **Sign language learning:** An application to teach sign language should control the progress of the user by examining the performed signs.

- **Dialog systems:** Dialog systems and information systems are usually operated by speech and are thus unusable for deaf people. Interfaces for barrier free dialog systems should allow sign language input.

- **Sign Language Recognition Requirements**

There are number of requirements to design manual sign language gesture recognizer:

- **Gesture Classes**

The first step in designing a gesture recognizer is to consider exactly what gestures are. For sign language recognition of the manual gestures, sign language gestures can be divided into four general classes, as in Table 2.1 (ordered from least to most complex), : SPSL, DPSL, SPDL, DPDL. gestures here are considered simply to be any possible movement that the human hand can make, including both movement of the fingers and of the hand in space.

Table 2.1 Classification of sign language gestures

Class	**Description**
SPSL	Static posture, static hand location
DPSL	Dynamic posture, static hand location
SPDL	Static posture, dynamic hand location
DPDL	Dynamic posture, dynamic hand location

Gesture Segmentation

One of the fundamental building blocks of a gesture recognizer is the ability to distinguish and recognize static hand shapes, i.e. gestures that belong to class SPSL (see Table 2.1). These postures can be recognized successfully when they occur singly. However, when postures occur one after another, posture recognition is more difficult because the recognizer also needs to determine the point where one posture begins and another ends in order to output a single symbolic token. These segments between signs, called movement epenthesis, are not part of either of the signs.

Classification

The final requirement for manual gesture recognition is classification. A wide variety of techniques have been applied to this. The most common ones are Hidden Markov Models (HMMs) and artificial neural networks. possible classification techniques will be explored and detailed later in the next chapter (Chapter 3_ Background).

2.2 Human–Computer Interaction: (HCI)

HCI is getting increasingly important as computer's influence on our lives is becoming more and more significant. With the advancement in the world of computers, the already-existing HCI devices (the mouse and the keyboard for example) are not satisfying the increasing demands anymore. Designers are trying to make HCI faster, easier, and more natural. To achieve this, Human-to-Human Interaction techniques are being introduced into the field of Human-Computer Interaction. One of the most fertile Human-to-Human Interaction fields is the use of hand gestures. People use hand gestures mainly to communicate and to express ideas.

The importance of using hand gestures for communication becomes clearer when sign language is considered. Instead of translating sign language through a human interpreter, a computerized translator is used to translate it into natural language and vice versa. In the recent years, sign language recognition became one of the important and attractive research areas of HCI.

2.3 Data Acquisition Approaches

In order to recognize Sign language symbols, this system must have some technique for data acquisition; in general there are three main approaches, each explained with its advantages and disadvantages as:

2.3.1 Vision-based Systems:

The first approach is to employ a camera in the system, such as those commonly found on many portable computing devices and phones, to capture body movements and hand shapes and rely on image processing and feature extraction techniques.

Advantages:

1. The camera captures the whole body movement, not just the hand, which enable the system to read facial expressions and hand locations, this will increase the features extracted from each sign.
2. Standard cameras are ubiquitous.
3. If the employed camera within a mobile or minicomputer the processing will be done on the same device without the need of external devices.

Disadvantages:

1. Image resolution depends on the position of the signer. If the signer is far away from the sensor, only a few pixels will be assigned to the hands that are insufficient for providing crucial details of finger positions.
2. Hand position and orientation have few geometric constraints and are therefore hard to locate.
3. There are similarities in handshapes in different signs that they can't be recognized accurately.
4. There are difficulties making the system signer independent.

2.3.2 Sensor-based Systems:

Constructing sensor-based sign recognition systems have become popular in the last decade as advances in human computer interfaces have fueled a new generation of devices sensory data-glove uses motion tracker for detecting hand movement.

Advantages:

1. The system would be flexible in measuring the changes in hand position, orientation, movement with employing various types of sensors, with different numbers.
2. It is much easier to locate finger positions using sensors located on a subject's fingers, so the accuracy regarding similarities in sign will increase.
3. It is suitable for urgent situations because it doesn`t need pre use preparations (like posturing in front of a camera).
4. It doesn't put any restriction regarding position.
5. It can achieve signer-independency; because its readings focusing on hand movement, not hand shape.
6. Its principle could be employed in other applications, like games or computer interfacing.

Disadvantages:

1. It doesn't read the facial expression.
2. Constraining the user interface through the use of additional sensors often conflicts with the goal of making sign recognition nonintrusive and natural.

2.3.3 Hybrid Systems:

To obtain hybrid system that utilizes a combination of sensors and vision systems. Hybrid systems often integrate data from a range of devices including sensors (often located on a subject's hands), conventional video cameras providing multiple angle views of a subject's hands, and thermo graphic cameras that operate outside the visible light band (e.g., infrared cameras).

Advantages:

1. Hybrid systems attempt to alleviate the need to use specialized sensors on the hands by employing more sophisticated imaging systems.

Disadvantages:

1. Sophisticated imaging systems often require a special peripheral (e.g., Kinect).
2. They are costly, and not as ubiquitous as standard cameras.

2.4 Hardware Components

This section mentions and explain hardware component to be used in this project.

2.4.1 DG5 VHand 2.0 OEM Data Glove

DG5 VHand data glove is a complete and innovative sensor. It contains five embedded bend sensors; it is possible to accurately measure the finger movements, while the embedded 3 axes accelerometer allows sensing both the hand movements and the hand orientation (roll and pitch). (see Figure 3.3 and 3.4 below)

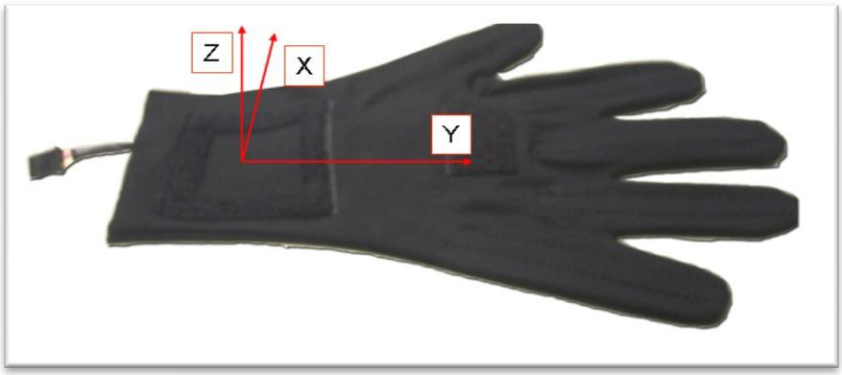


Figure 2.4 DG5 Data-Glove Data axes reference

Figure 2.5 Right DG5 dataglove: bend sensor reference

The glove communicates with external devices via a 4 wires connector. The transmission is made via TTL levels and it uses a standard RS232 protocol at 115200 bps, so interfacing the device to a PC or a microcontroller is really immediate.

The glove can be powered from 3.3V to 5V and it is really safe power, it consumes less than 20mA. It has been developed for wireless and autonomous operations and it can be powered with a battery guaranteeing a long operative period. For full datasheet, see Appendix A.

This data glove is the input device of this sensor-based system. The gesture will be transferred into electrical signals through its sensors and accelerometers and it will be connected to the other main component by throughout the output connector.

2.4.2 Arduino Mega 2560 Microcontroller

Microcontroller is a device which integrates a number of components of a microprocessor system into a single microchip, and it is optimized to interact with the outside world through on-board interfaces.

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply after connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery it will get started.

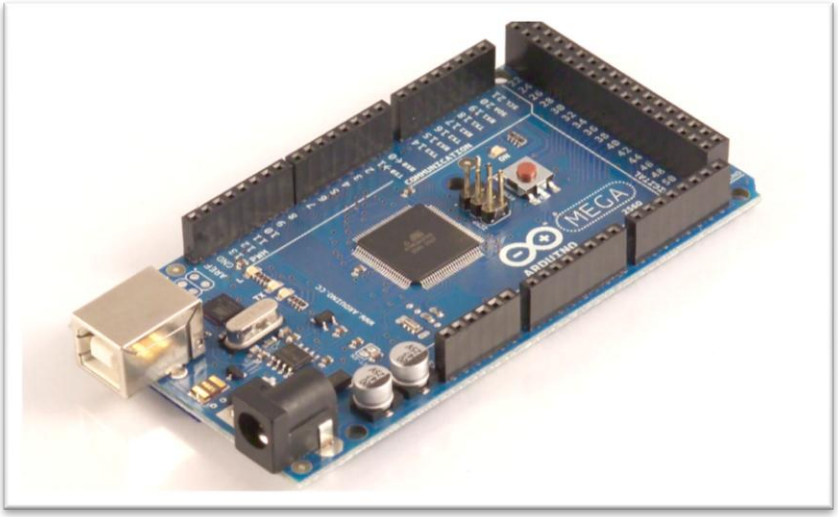


Figure 2.6 Arduino Mega 2560 Microcontroller

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer.

The Arduino Mega 2560 is the chosen microcontroller for this project. The designer chooses this type of microcontroller because it has features suitable for what is needed, such as:

1. The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the boot loader), 8 KB of 1 SRAM and 4 KB of EEPROM
2. The Arduino Mega2560 can be powered via the USB connection or with an external power supply.
3. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery.
4. The Atmega2560 on the Arduino Mega comes pre burned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer.
5. The Atmega2560 on the Arduino Mega can be programmed through high level languages such as C.

For full datasheet, see Appendix A.

2.4.3 High Definition Speech Synthesizer Systems (TTS-EM-HD2

The Text Speak 'EM' Text-To-Speech processor converts ASCII text directly to voice with unlimited vocabulary. The platform boasts multi-language capability on a sophisticated board level solution that accepts RS-232 and optional Ethernet input to provide high quality speech from text based data streams.

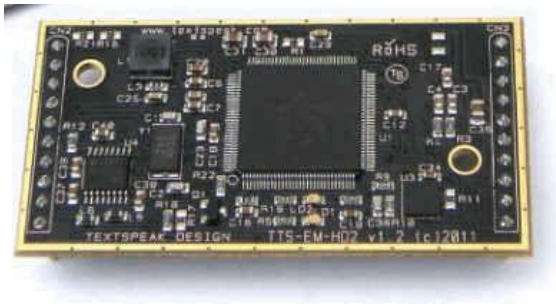


Figure 2.7 TTS-EM-HD2 Text-to-Speech Synthizer

This Text-to-Speech component has many features:

- Unlimited vocabulary Text to Speech generation**.**
- Multi-language capability with ‘a-la-carte ‘feature scalability.
- Integrated command language and in stream control codes**.**
- Dual audio output; Line Out and Speaker Amplifier (1w/8ohm).
- Switches seamlessly from TTS to pre-recorded files on the fly.
- Low power operation with auto power down and sleep modes**.**
- Flash upgradeable for both revision and language enhancements**.**
- RS-232 expandable to accept USB, Ethernet or 802.11 ASCII coded data.
- Small footprint, 1" x 2" board incorporating standard .1" 1x10 headers**.**
- Optional Ethernet, Automatic Level Control & integrated 25 watt amplifiers**.**
- RoHS and Industrial Temp standard.

The next figure shows the block diagram for the TTS-EM-HD2 version2 Text-To-Speech Synthesizer.

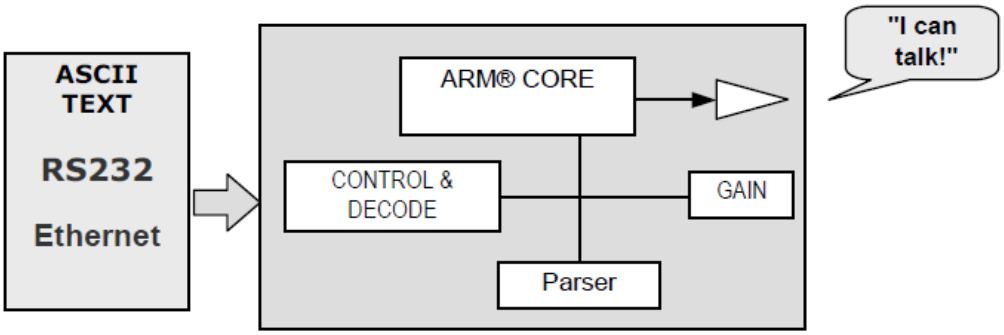


Figure 2.8 TTS-EM-HD2 Block Diagram

For further information see appendix A.

2.4.4 Graphical LCD Display

The project needs a display to show the output words of this system for the mute to be able to verify whether the signs translated correctly or not. The character LCD display not used here because it doesn’t support Arabic language and we chose

graphical LCD 128 X 64 that shown in the next picture. This can display several mid sentences. It display 8 lines each of them of 16 characters.

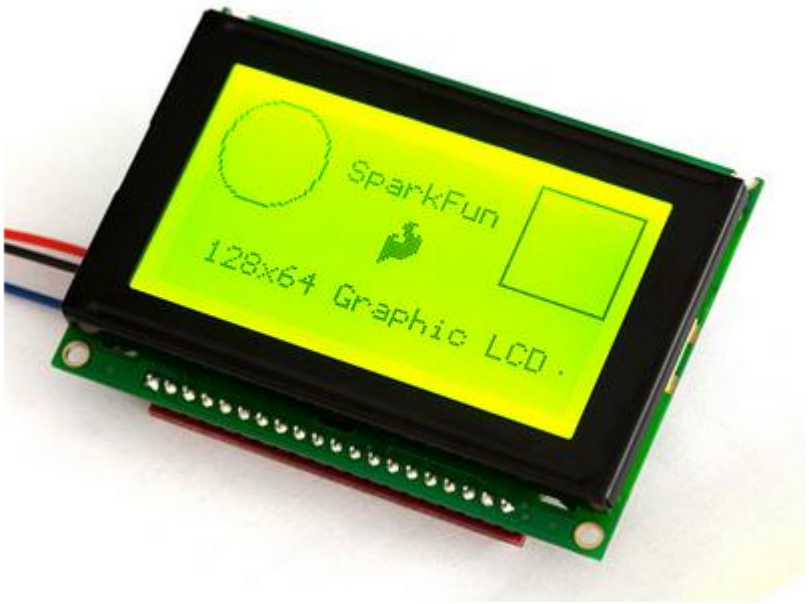


Figure 2.9 Graphical LCD

To make the GLCD serial the Serial Graphic LCD backpack with 160x128 pixels is used, the next figure presents it.

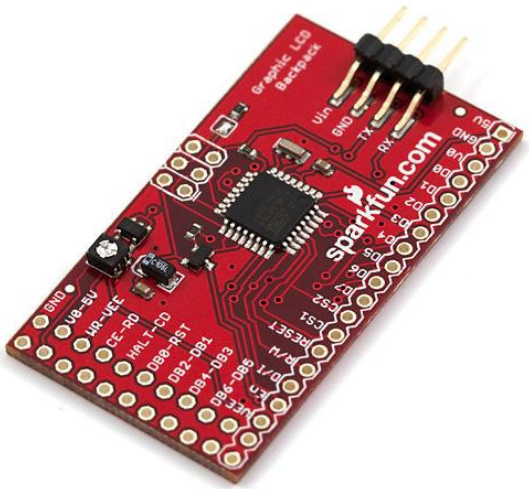


Figure 2.10 Serial Graphic LCD backpack

For further information see Appendix A.

2.5 Machine Learning

Machine learning is concerned with the design and development of algorithms and techniques that allow computers to "learn."

The machine learning field offers a wide range of algorithms capable of extracting rules and patterns from the available training dataset. This feature makes the use of these algorithms suitable for the problem of sign recognition. Even so, the real time nature of gesture recognition brings to light new issues not present in other kinds of classification problems.

2.6 Hidden Markov Model

There are some concepts concerning Hidden Markov Model, they are explained in this section.

- **Markov Chain (Markov Model):**

Mathematical system that consists of a sequence of random events based on probabilities instead of certainties, usually characterized as memory less, where the current state of a variable or system is independent of all past states, except the current (present) state, this concept is a form of something known as the Markov property.

- **Hidden Markov Model (HMM):**

Are stochastic methods to model temporal and sequence data ,which mean is a doubly embedded stochastic process with an underlying stochastic process that is

not observable (it is hidden),but can only be observed through another set of stochastic processes that produce the sequence of observations.

Hidden Markov Models attempt to model such systems, and allow among other things to understand the most likely sequence of states that produced a given output sequence, and to understand which will be the most likely next state (and thus predicting the next output), and calculate the probability that a given sequence of outputs originated from the system (allowing the use of hidden Markov models for sequence classification).

2.6.1 Elements of an HMM:

An HMM is characterized by the following elements [6]:

1. N, the number of states in the mode.
2. M, the number of distinct observation symbols per state.
   We denote the individual states as S= {S_1, $S_{2\ldots}$ S_N}, and the state at time t as q_t.
   We denote the individual symbols as V= {v_1, $v_2 \ldots$ v_M}.
3. The state transition probability distribution A={ a_{ij} } where
   $$a_{ij} = P\,[q_{t+1} = S_j | q_t = S_i], \qquad 1 \leq i, j \leq N \qquad (2.1)$$
4. The observation symbol probability distribution in state j, B = {$b_j(k)$}, where
   $$b_j(k) = P\,[v_k \text{ at } t | q_t = S_j], \qquad 1 \leq j \leq N \qquad (2.2)$$
   $$1 \leq k \leq M$$
5. The initial state distribution π= { π_i } where
   $$\pi_i = P[q_1 = S_i], \qquad 1 \leq i \leq N \qquad (2.3)$$

Given appropriate values of N, M, A, B, and, the HMM can be used as a generator to give an observation sequence $O=O_1 O_2 \ldots O_T$ Where each observation O_t is one of the symbols from V, and T is the number of observation in the sequence.

A complete specification of an HMM requires specification of two model parameters (N and M), specification of observation symbols, and the specification of the three probability measures A, B, and π.

The complete parameter set of the model is $\lambda = (A, B, \pi)$.

2.6.2 The basic problems for HMMs.

There are basic problems that must be solved for the model to be useful in real-world applications. These problems are the following:

- **The Evaluation Problem**

Namely given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model. We can also view the problem as one of scoring how well a given model matches a given observation sequence.

Given the observation sequence $O=O_1\ O_2\ \dots\ O_T$, and a model $\lambda = (A,B,\pi)$,how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

- **The Learning(training) Problem**

How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

Is the one in which we attempt to optimize the model parameters so as to best describe how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called a training sequence since it is used to "train" the HMM. The learning problem is the crucial one for most applications of HMMs, since it allows us to optimally adapt model parameters to observed training data i.e., to create best models for real phenomena.

2.6.3 Solutions to the basic problems of HMMs:

There are algorithms to solve the basic HMM problems. These solutions are the following:

- **Forward-backward algorithm for Evaluation problem[6]:**

We wish to calculate the probability of the observation sequence $O = O_1 O_2 \dots O_T$, given the model λ, i.e., $P(O|\lambda)$.

We have a model $\lambda = (A, B, \pi)$ and a sequence of observations $O = O_1 O_2 \dots O_T$, and $P(O|\lambda)$ must be found. We can calculate this quantity using simple probabilistic arguments. But this calculation involves number of operations in the order of N^T. This is very large even if the length of the sequence, *T* is moderate.

a. Forward Algorithm:

Therefore we have to look for another method for this calculation. Fortunately there exists one which has a considerably low complexity and makes use an auxiliary variable, $\alpha_t(i)$ called *forward variable*.

The forward variable is defined as the probability of the partial observation sequence, $O_1 O_2 \dots O_T$, (until time t) and state Si at time t , given the model λ. We can solve for $\alpha_t(i)$ inductively, as follows:

1- Initialization:

$$\alpha_1(i) = \pi_i \, b_i(O_1), \qquad 1 \le i \le N \qquad (2.4)$$

2- Induction :

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) \, a_{ij}] \, b_j(O_{t+1}), \qquad 1 \le t \le T-1 \qquad (2.5)$$

$$1 \le j \le N$$

3- Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \qquad (2.6)$$

The complexity of this method, known as the *forward algorithm* is proportional to N^2T, which is linear with T whereas the direct calculation mentioned earlier, had an exponential complexity.

b. Backward algorithm:

In a similar way we can define the $\beta_t(i)$ *backward variable*, $\beta_t(i) = P(O_{t+1} O_{t+2} \ldots O_T | q_t = S_i, \lambda)$ as the probability of the partial observation sequence $O_{t+1}, O_{t+2}, \ldots, O_T$, given that the S_i at time t and the model λ .

1- Initialization :

$$\beta_1(i) = 1, \qquad 1 \leq i \leq N \qquad (2.7)$$

2- Induction :

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \; b_j(O_{t+1}) \, \beta_{t+1}(j) \qquad (2.8)$$

$$t = T - 1, T - 2, \ldots\ldots, 1, \quad 1 \leq i \leq N$$

- **Baum-Welch Algorithm for learning problem[6]:**

This method can be derived using simple occurrence counting arguments or using calculus to maximize the auxiliary quantity.

$$Q(\lambda,\bar{\lambda}) = \sum_Q P(Q|O,\lambda)\log[P(O,Q|\bar{\lambda})] \quad (2.13)$$

A special feature of the algorithm is the guaranteed convergence. To describe the *Baum-Welch algorithm*, (also known as *Forward-Backward algorithm*), we need to define two more auxiliary variables, in addition to the forward and backward variables defined in a previous section. These variables can however be expressed in terms of the forward and backward variables.

First one of those variables is defined as the probability of being in state *i* at *t=t* and in state *j* at *t=t*+1. Formally,

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda). \quad (2.14)$$

Using forward and backward variables this can be expressed as,

$$\xi_t(i,j) = \frac{\alpha_t(i)\ a_{ij}\ b(O_{t+1})\ \beta_{t+1}(j)}{\sum_{i=1}^{N}\ \sum_{j=1}^{N}\ \alpha_t(i)\ a_{ij}\ b_j(O_{t+1})\ \beta_{t+1}(j)} \quad (2.15)$$

The second variable is the a posteriori probability,

$$\gamma_t(i) = P\{qt=i|O, \lambda\} \quad (2.16)$$

That is the probability of being in state *i* at *t=t*, given the observation sequence and the model. In forward and backward variables this can be expressed by,

$$\gamma_t(i) = \left[\frac{\alpha_t(i)\,\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\,\beta_t(i)}\right] \quad (2.17)$$

One can see that the relationship between $\gamma_t(i)$ and $\xi_t(i,j)$ is given by,

$$\gamma_t(i) = \sum_{j=1}^{N}\ \xi t\ (i,j), \qquad 1 \le i \le N,\ 1 \le t \le M \quad (2.18)$$

Using the concept of counting event occurrences we can give a method for re-estimation formulas for π, A , B are

$$\bar{\pi}_i = \gamma_1(i), \qquad 1 \le i \le N \quad (2.19)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi t\,(i,j)}{\sum_{t=1}^{T-1} \gamma 1(i)} \qquad 1 \le i \le N,\ 1 \le j \le N \qquad (2.20)$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}, \qquad 1 \le j \le N,\ 1 \le k \le M \qquad (2.21)$$

2.6.4 Weaknesses of HMM

- The Viterbi algorithm is expensive, both in terms of memory and compute time. For a sequence of length *n*, the dynamic programming for finding the best path through a model with *s* states and *e* edges takes memory proportional to *sn* and time proportional to *en*. Other algorithms for hidden Markov models, such as the forward-backward algorithm, are even more expensive.
- The HMM needs to be trained on a set of seed sequences and generally requires a larger seed than the simple Markov models. The training involves repeated iterations of the Viterbi algorithm, which can be quite slow.
- For a given set of seed sequences, there are many possible HMMs, and choosing one can be difficult. Smaller models are easier to understand, but larger models can fit the data better.

Chapter Three

Conceptual Design and Analysis

3.1 System Abstraction

3.2 System Hardware Design

3.3 System Software Design

3.1 System Abstraction

This section handles the proposed system, showing requirements, design concepts, and other criteria.

3.1.1 User requirements

The following requirements are summarized in the following points:

1. The system should be able to translate any represented sign, found in the dictionary, into its corresponding speech.
2. The translation processing of user signs should be as fast as possible to keep up with current situation of the user.
3. Recognition and translation process must be as accurate as possible.
4. The user must be able to test the performance of the system easily while using it; through a display to show the text of the generated voice and the device status.
5. The system must be user-independent; not restricted to a certain person.
6. The system should be easily-portable to anywhere.

3.1.2 Other design criteria

The design aspires to fulfill the functionalities and requirements assumed for the user, but it should also fulfill the following Criteria:

1. It should be affordable for any mute person to purchase.
2. It must be light-weighted for any user to carry.

3.1.3 Basic System Abstraction

Training phase is made on PC after collecting the samples, and then the resultant functions would be implemented within the program on the microcontroller. The recognition phase done by the microcontroller, which will map the sign to the corresponding letter, number or a word through software.

The Basic Block Diagram of the System is represented in the bellow (Figure 3.1):

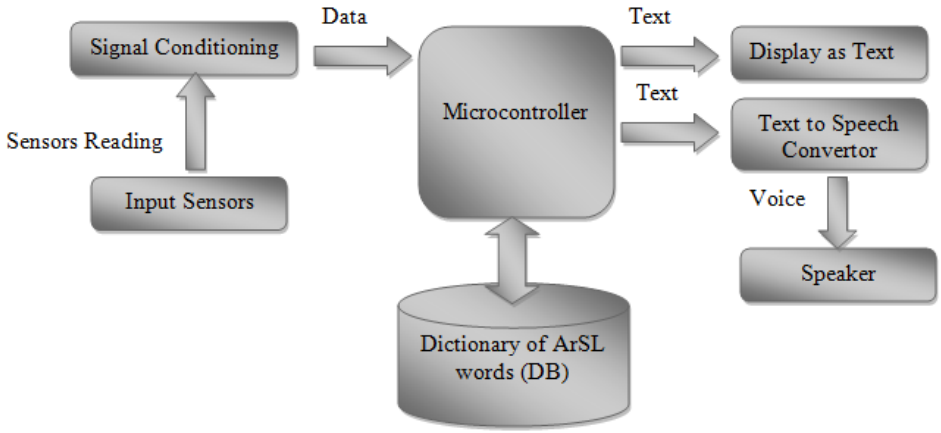


Figure 3.1 Block Diagram

3.1.4 System flowchart

The basic flowchart of the system is represented in figure below:

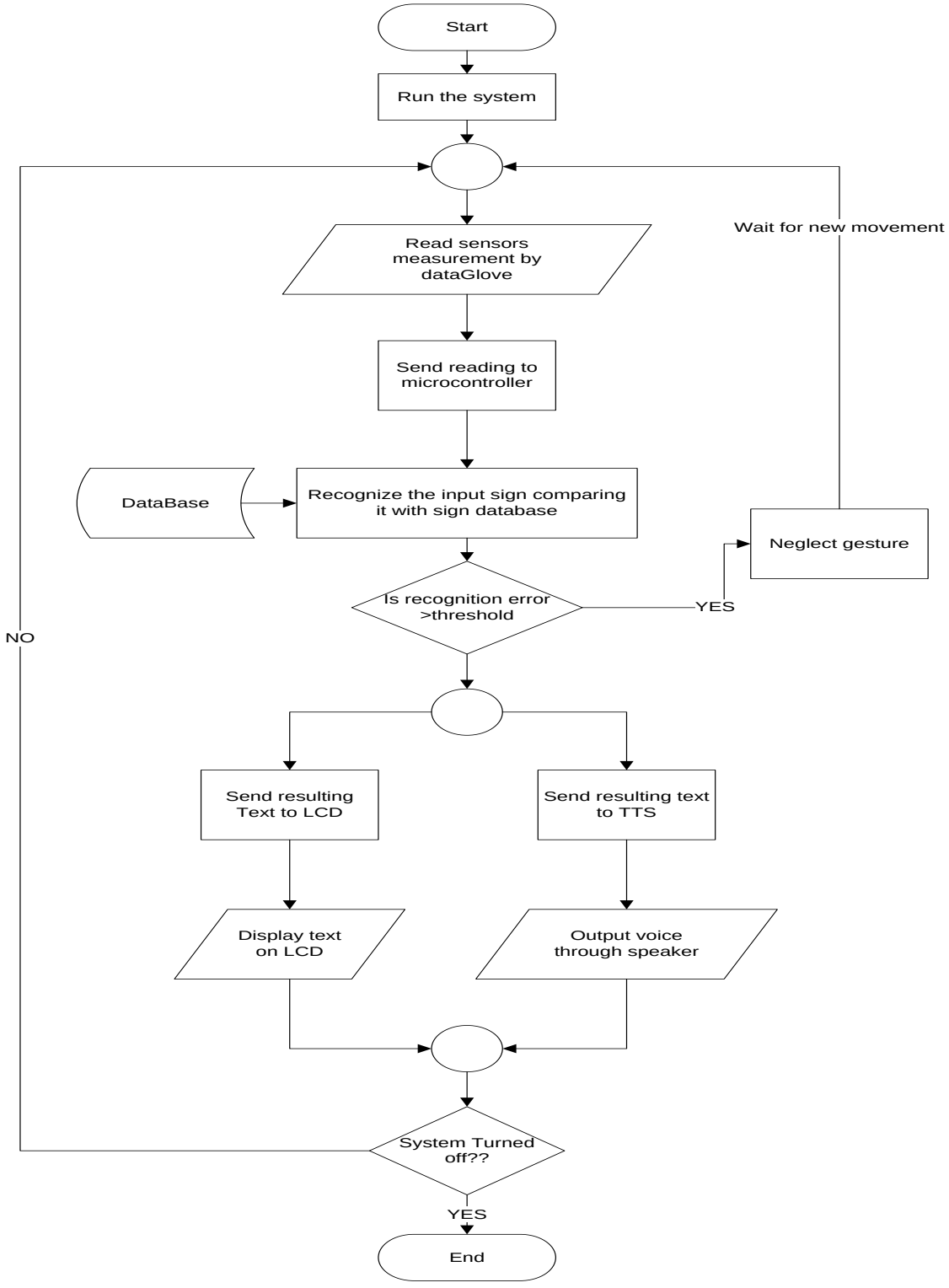


Figure 3.2 System General Flowchart

3.2 Software System Design

This section explains the expected design of the software components, classified and characterized according to function, and with the intention to keep dependency to the minimum.

The software of this system consists of two phases:

- **First Phase (Simulation Phase)**

We will do the next functional requirements using PC:

1. The software must be able to read the analog input sample from the data glove and their corresponding meanings, and then store them in dictionary.
2. Using Matlab to train HMM on the samples, testing the HMM accuracy on different set of samples until obtaining the required accuracy.

- **Second Phase (Recognition Phase)**

We will do the next functional requirements using microcontroller:

1. Programming the microcontroller by the function of models probabilities from the training phase.
2. The software must be able to connect between the data glove, microcontroller, LCD display, and TTS synthesizer.
3. The software must be able to read data from the glove and then match it with nearest data store in dictionary.
4. The software must be able to show the text on graphical LCD.
5. The software must be able to send the text to TTS synthesizer.
6. The software must be user independent.

- **Software System Design Abstraction**

The software can be modeled as a set of modules each performing a specific task to accomplish the desired main logic, so in general it can be can present as shown in figure 3.3 and figure 3.4:

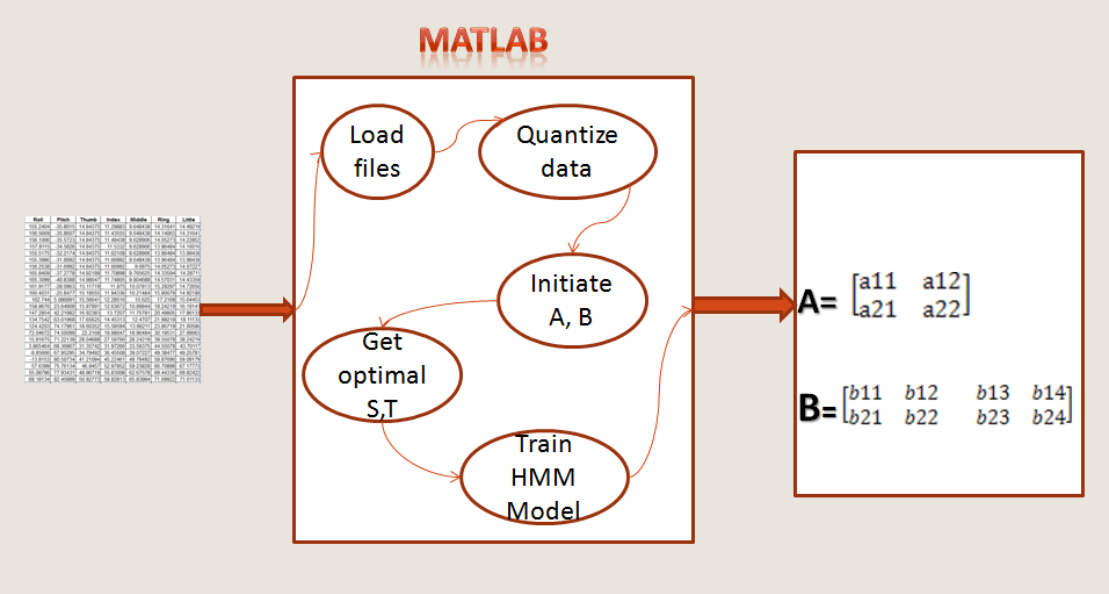


Figure 3.3 Software Block Diagram(simulation phase).

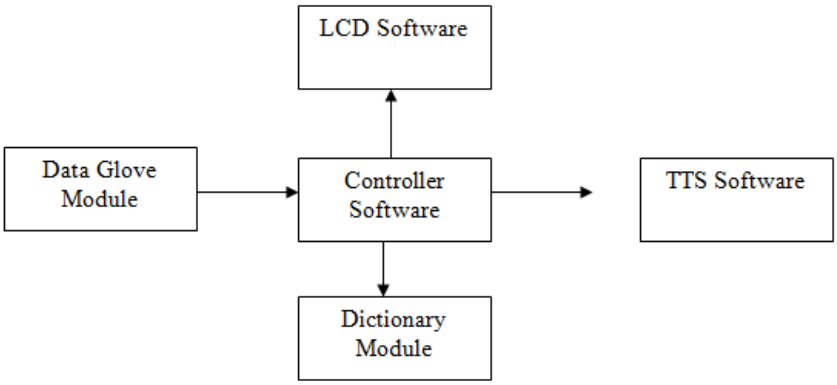


Figure 3.4 Software Block Diagram(recognation phase).

Basically, the software controller takes feature of the gesture from Data Glove module, then compare the feature with other stored in dictionary module ,then the microcontroller module process data and choose the corresponding data of this gesture ,then display the text on LCD Screen Shield module, and output the appropriate voice using TTS synthesizer module.

- **Sign Language Recognition System**

The functional requirements consist of two phases: train and recognition. The following figures explain the recognition phase:

To Recognize continuous signs, two major layers are needed: the first layer is segmentation; the input of this layer are 10 continuous glove signals (five signal determine bending degree for each right-hand finger, three signals from accelerometer XYZ and the last two signals are roll and pitch). The output from this layer is 10 signal glove lectures. Then, the next layer is the classification layer; it performs classification technique on the output of the preceding layer to get, finally, the recognized sign.

Chapter Four

Detailed System Design

4.1 Hardware Components Detailed Design

4.2 Software Components Detailed Design

4.1 Hardware Components Design

As shown in Figure 4.1, the hardware of the system consists of the following components, starting from the center towards the peripherals:

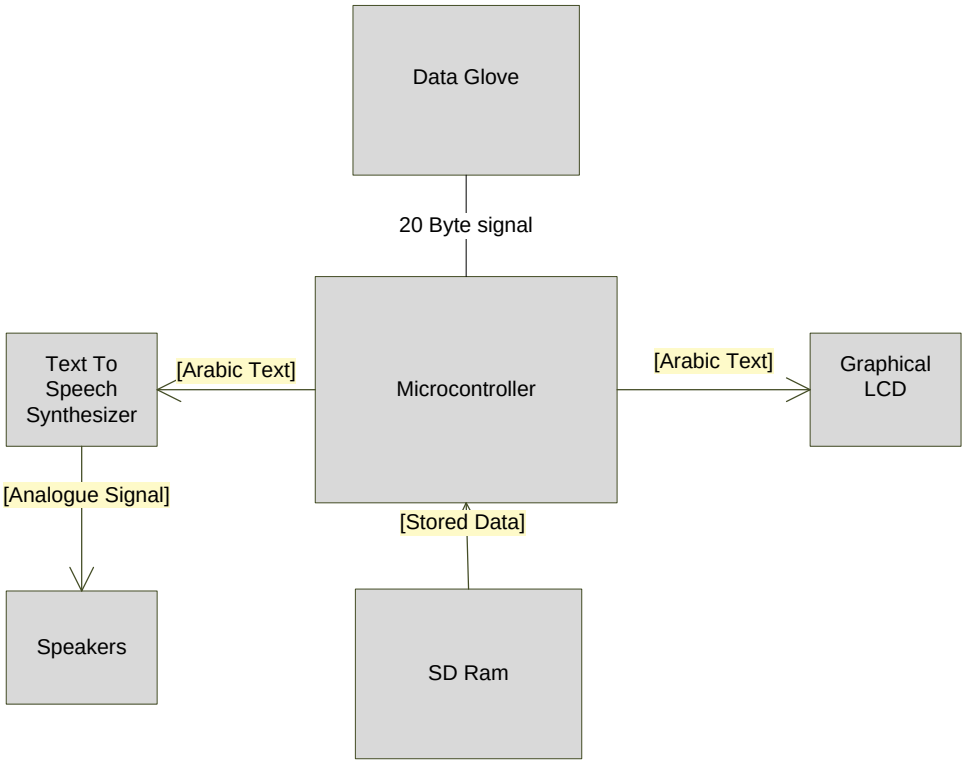


Figure 4.1 Detailed Hardware Block Diagram

- **Microcontroller**

The central element of the system is the microcontroller which is chosen to be Arduino. It will run the software that shall perform the recognition, and it will issue the commands for other elements to perform their duties.

This microcontroller must meet certain criteria to fit the application of this system, best summarized in the following requirements:

1. It must be programmable on a high level language for easier implementation.
2. It must contain USB port.

3. It must support SD memory card for memory capacity large enough to hold dictionary.
4. It should consume very little power and be able to withstand long hours of operation.
5. It must be small in physical size.
6. It must support multi parallel input (i.e. from sensors).

- **Data glove**

The data glove is a basic element for the system, it provides the signs that would then be interpreted to associated Arabic meanings by the microcontroller, and it must have certain characteristics, summarized as follow:

1. It must contain enough sensors to read differences in similar signs, for that it must have bend sensors, and orientation sensors, or accelerometers.
2. It must have USB port to be connected directly to PC especially during training phase and testing accuracy.
3. It must have Bluetooth adapter for future expansion of the system to support new words which need tow hands.
4. Its power consumption must be low to operate for long period without recharging.
5. Its size must fit a variety of hand sizes, it must be elastic.

Figure 4.2 illustrates the Data Glove components:

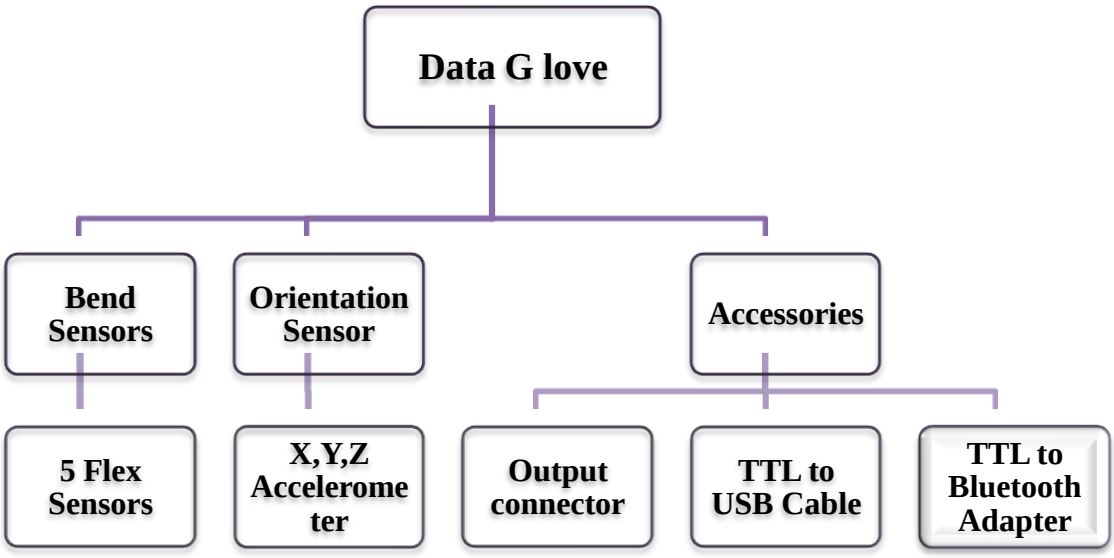


Figure 4.2 DataGlove Components

The data gloves would be connected to the microcontroller directly; each sensor should give output resolution with at least 20 Bytes; to get high sensitivity toward changes between different movements.

- **Graphical LCD Screen**

The LCD is significant to the system to display output as text so the mute people could understand, to make sure that the wanted meaning translated properly. The appropriate LCD has certain functional requirements demonstrated as follow:

1. It must have the ability to display Arabic language.
2. It must be small in size to be carried along with other components on the same hand, but large enough to show a complete sentence.

The chosen LCD for this system graphical LCD that can display Arabic it is 128 X 64 pixel.

- **ASCII Text to Speech Synthesizer Systems**

The system must give an output as voice; this is being done through the text to speech synthesizer, which will internally convert ASCII text to the corresponding voice. This system should have these characteristics:

1. It must translate text to voice in real time.
2. It must support Arabic language.
3. It mustn't consume much power.
4. It must be small in size.
5. It must be connected directly to microcontroller.

The text to speech synthesizer chosen for the system is the TTS-EM-HD2 version 2; because the previous characteristics are all implemented in it.

4.2 Software design

The system software has two phases :

- Simulation phase.
- Evaluation phase.

The quantization algorithm is needed for both phases, the following flowchart represents the algorithm:

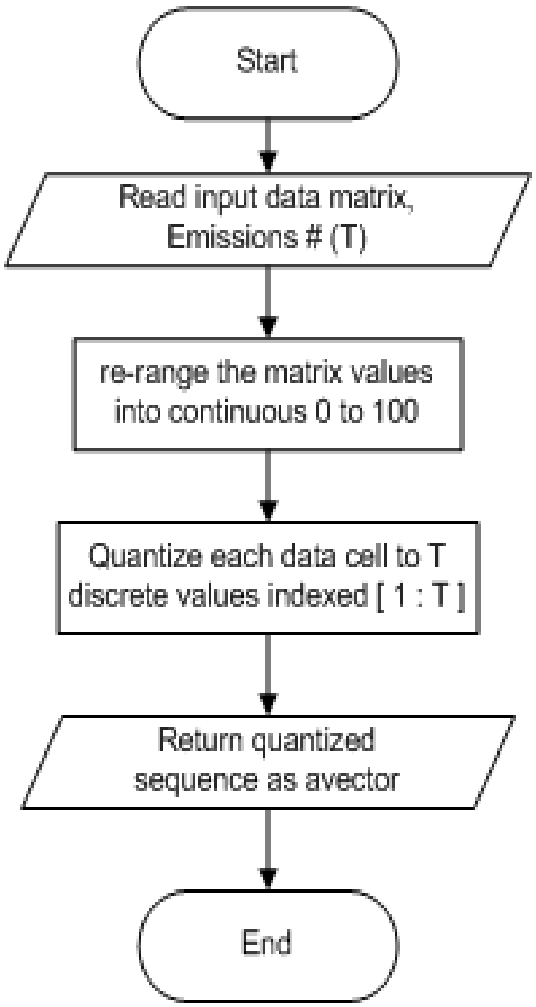


Figure 4.3 Quantization algorithm.

In simulation phase we need to optimized number of states and number of emission to improve the efficient of the system. The following figure shows the optimizing number of states algorithm.

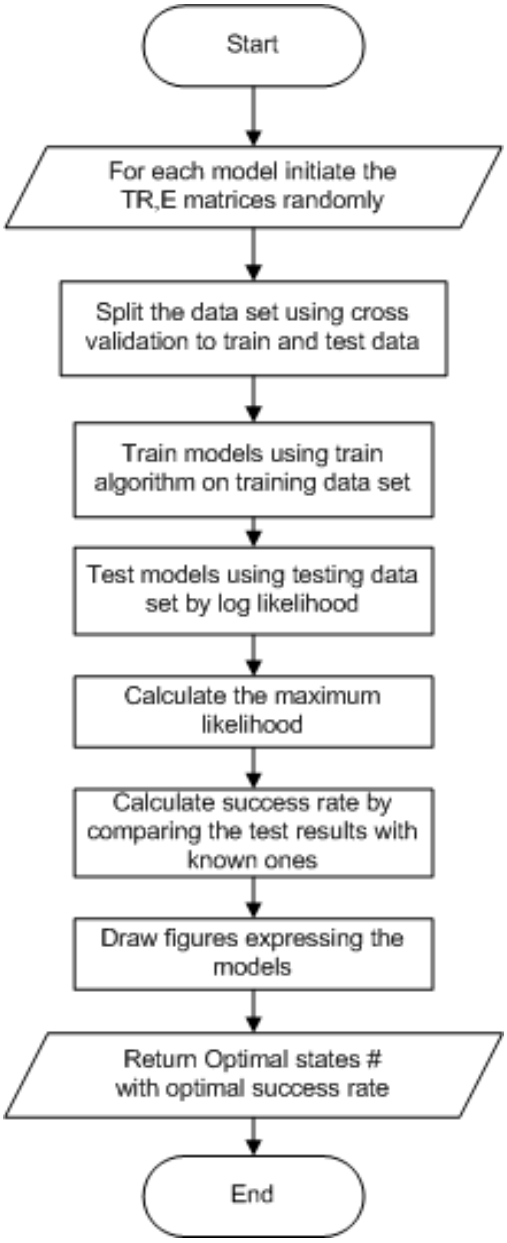


Figure 4.4 Optimizing # of States Algorithm.

The following figure shows the optimizing number of emissions algorithm.

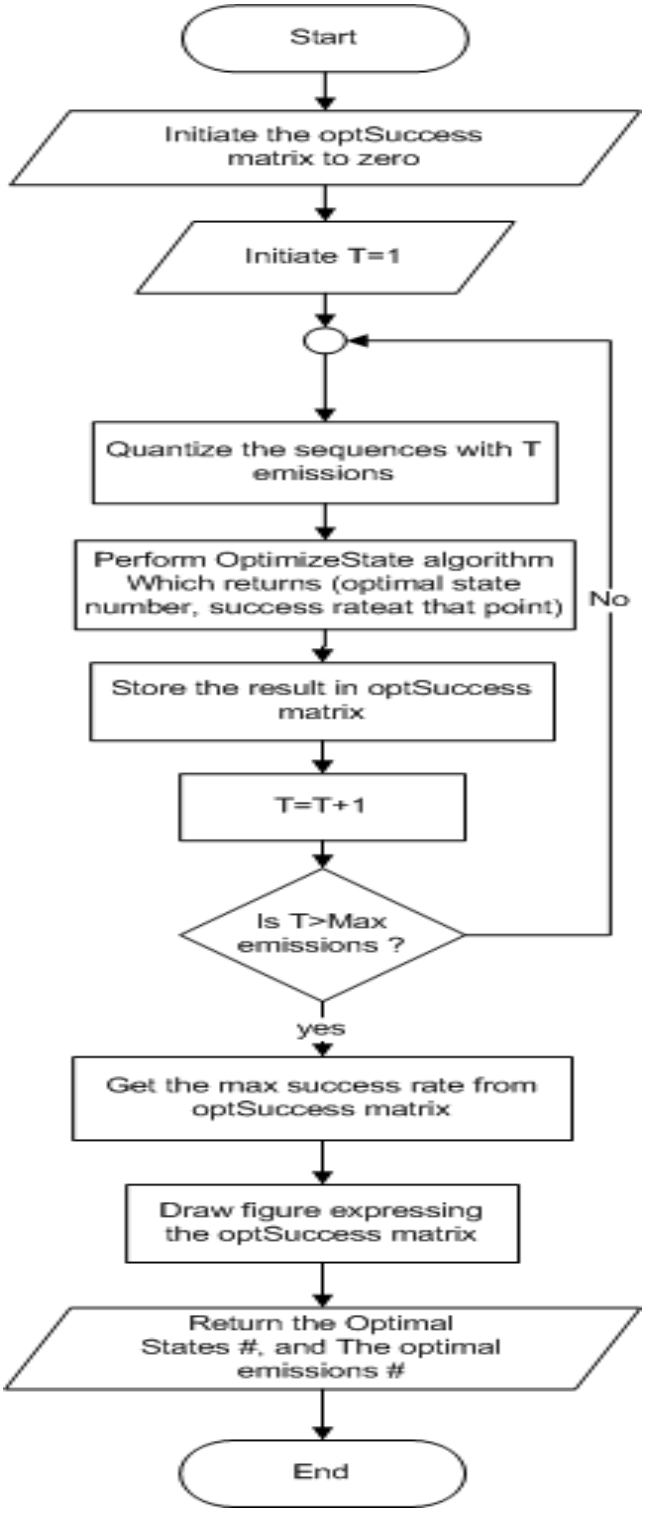


Figure 4.5 Optimizing # of Observations (EMISSIONS) Algorithm.

The evaluation phase needs the tow following algorithms; the recognition algorithm which used to determine the sign belongs to which model by using the evaluation algorithm that calculates the probability of the sign in each model.

The following figure shows the calculate alghorithm:

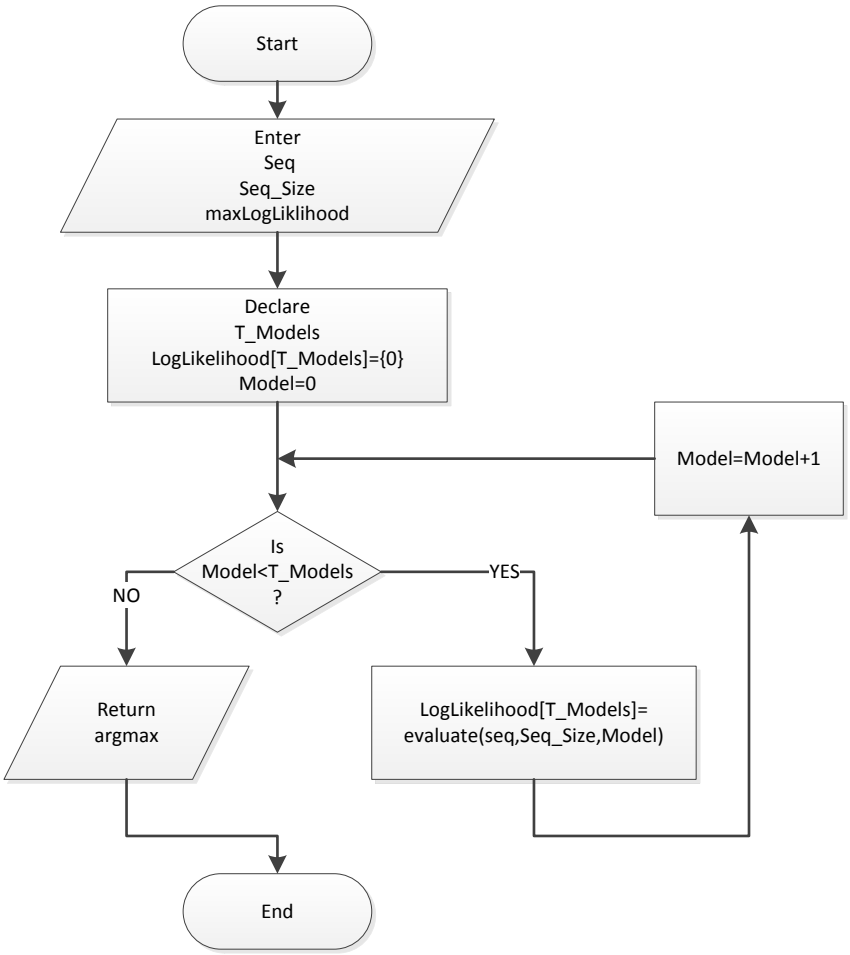


Figure 4.6 Calculate algorithm.

The following figure shows the evaluate algorithm:

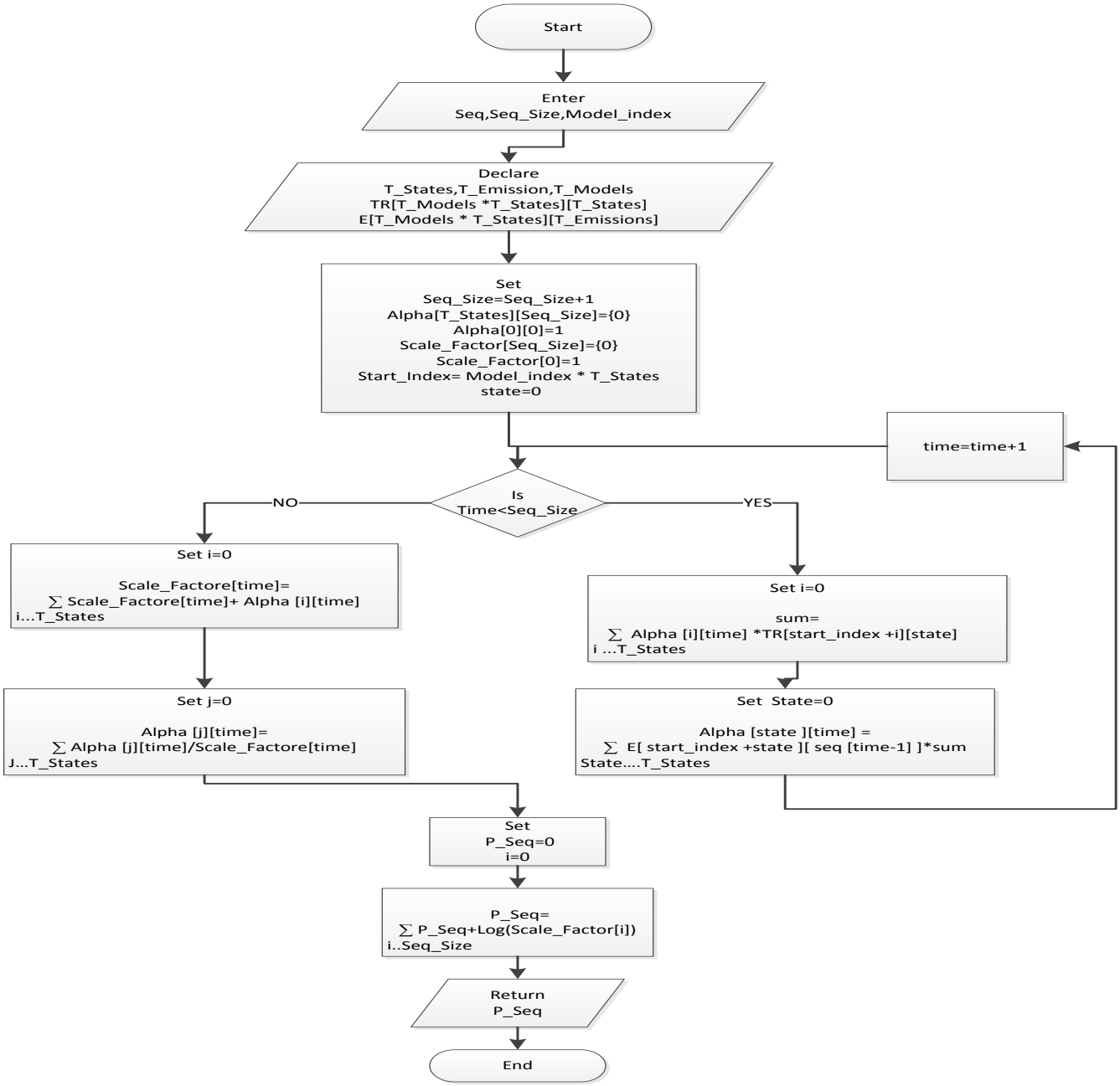


Figure 4.7 Evaluate algorithm.

Chapter Five

System Implementation and Testing

5.1 Overview:

This chapter introduces the development environment for the project it also clarify the system`s components implementation and testing, then it views the integrated system implementation and testing.

5.2 Development Environment Overview

The following is the set of programming tools used to develop the components specified in the design phase as they are presented here, with a small description to illustrate their use and their relevance to this project:

- **Microsoft Visual Studio 2010**

An integrated development environment (IDE) available from Microsoft is used to in the development of C++ Code, and the GUI application .As shown in Figure 5.1:

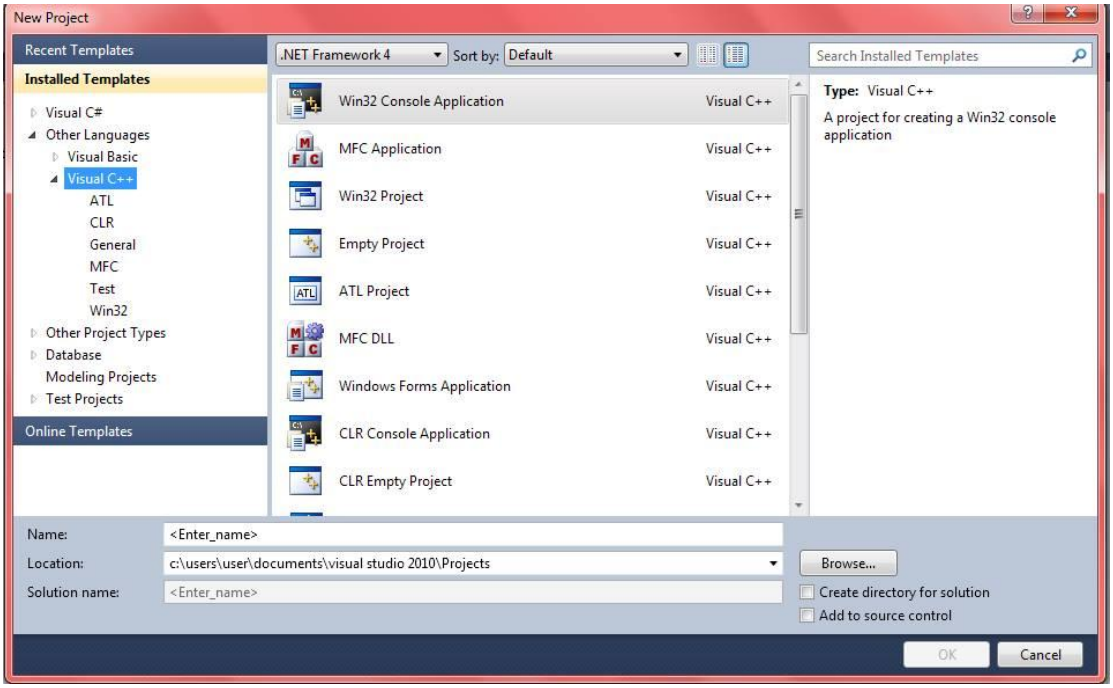


Figure 5.1 Microsoft Visual Studio 2010

The following are its main components:

- **Windows Forms Designer**

Used to build GUI applications using Windows Forms, where Data-bound controls can be created by dragging items from the Data Sources window onto a design surface. The UI is linked with code using an event-driven programming model.

- **Arduino Ide for Visual Studio**

It is a free Arduino plugin for Visual Studio 2008-2013 and Atmel Studio 6.1 allows an Arduino project (with code completion) to be developed, compiled and then uploaded to any Arduino micro-processor.

Visual macro has productivity features advanced over the original Arduino IDE, for example, include library sources directly in the projects and implement object oriented classes.

And the next picture shows the testing for the Ide.

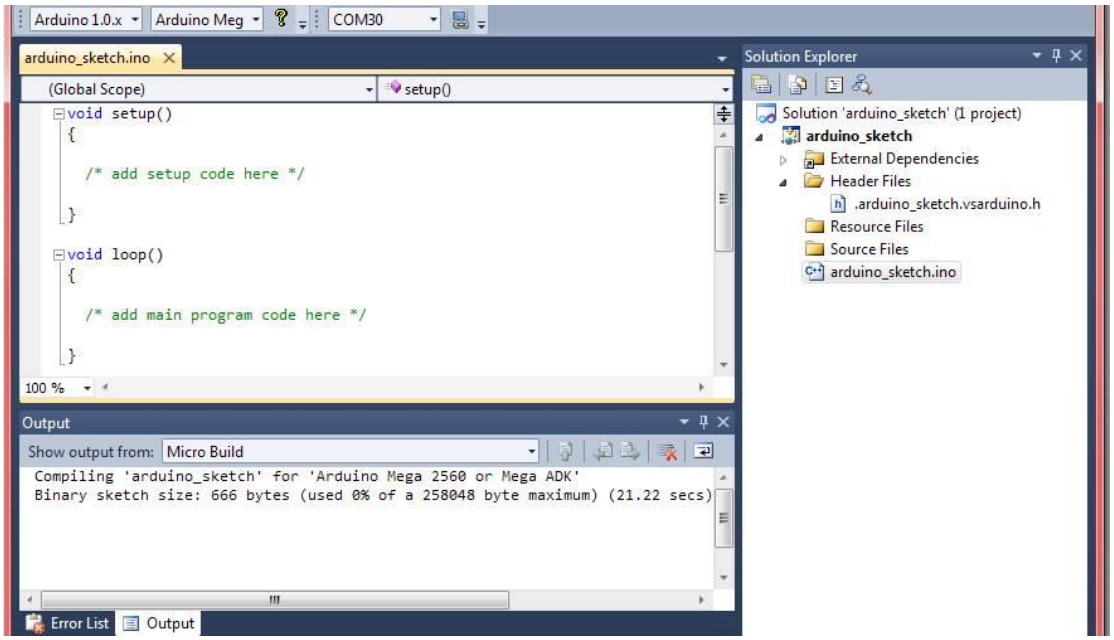


Figure 5.2 Arduino Idefor Visual Studio

- **Arduino IDE Software**

Arduino is a popular open-source , designed to make the process of using electronics in multidisciplinary projects more accessible it includes a code editor and is also capable of compiling and uploading programs to the Arduino board with a single click.

There is typically no need to edit make files or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools, Arduino programs are written in C/C++, although users only need define two functions to make a run able program. As shown in Figure 5.3.

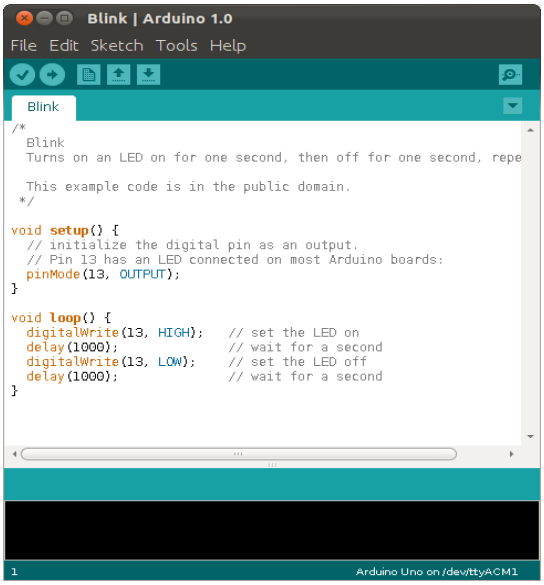


Figure 5.3 Arduino IDE

- **Real Term:**

Real Term has been created mostly to represent a better alternative to the HyperTerminal application, especially when it comes to debugging microcontrollers. It allows capturing, controlling and debugging binary data as well as other types of data streams. Data can be viewed as ASCII and various HEX and binary forms.

In this system, it is used to test and communicate with the system serial devices as serial GLCD and the data glove, as shown in Figure 5.4 bellow.

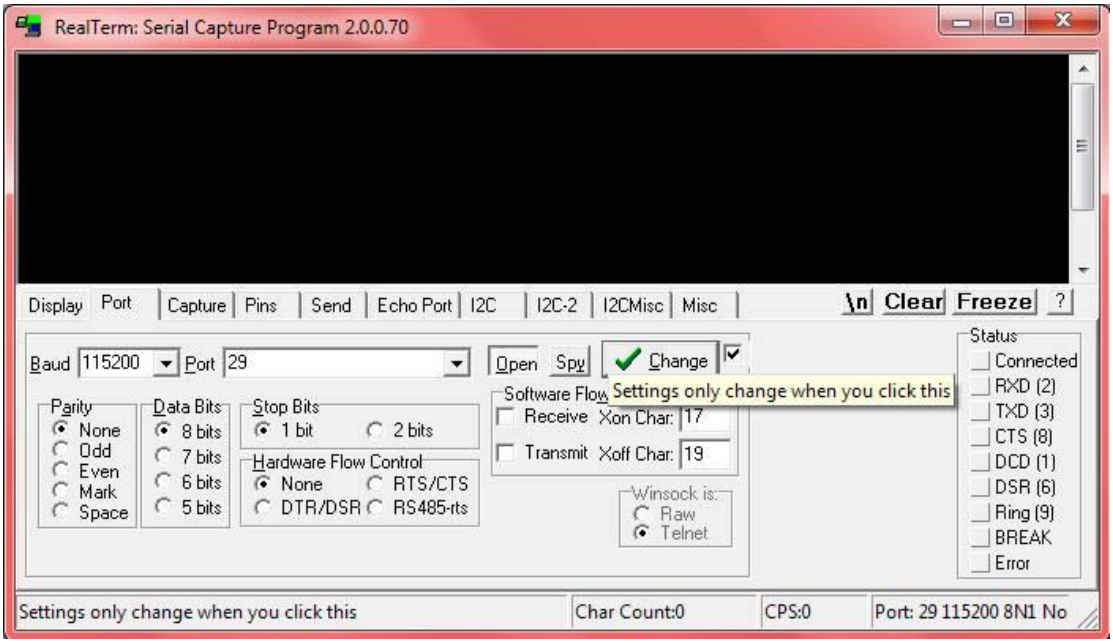


Figure 5.4 Real Term screen

- **MATLAB**

It is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces. It has statistics toolbox functions related to machine learning and especially hidden Markova model, as shown in Figure 5.5 bellow.

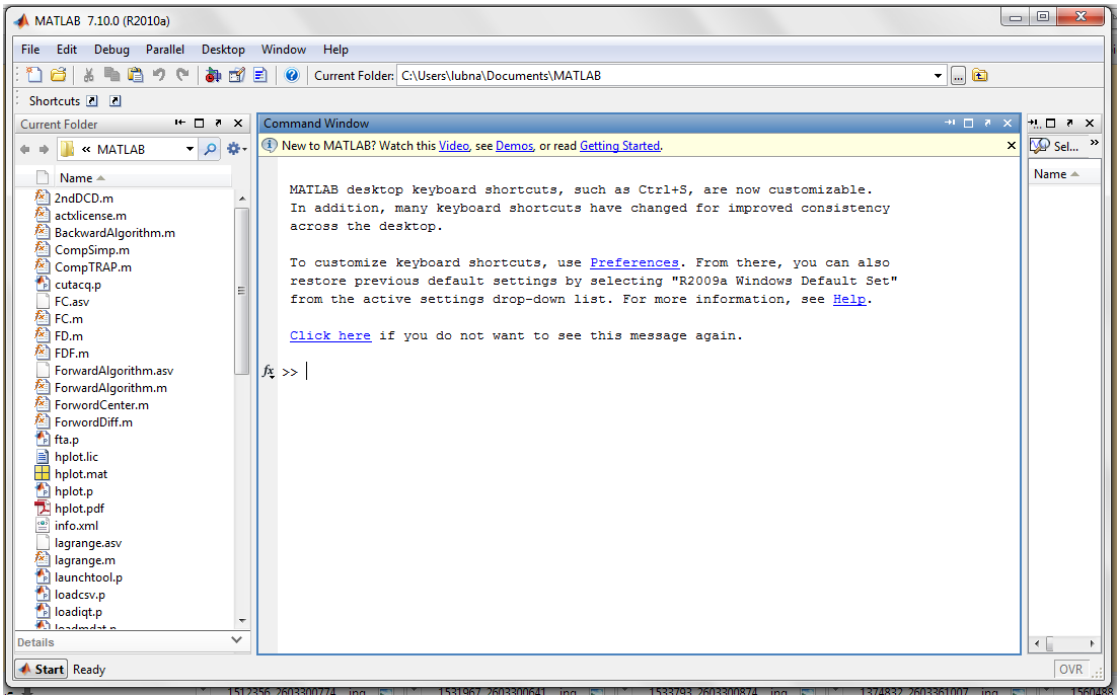


Figure 5.5 MATLAP

- **BTc Sound Encoder v3.0**

The BTC Sound Encoder is software that have been written for the purpose of converting sound (in wave files) to a 1-bit format that can be played back on very cheap hardware, like a microprocessor chip (ie PIC, Atmel, Motorola etc).

After the code is generated in bit format, the code must be loaded on the hardware like PIC microcontroller and then it can play the sound back with practically no other hardware is needed. It does not need a DAC (Digital to Analog Converter IC).

This is simple way to make the system generate voice for testing it, and make it more affective, the next figure views the program interface and how it works.

5.3 Unit Testing:

This section presents the testing for each individual unit of the system to determine if they are fit for the project requirements.

5.3.1 Data Glove Testing:

The glove communicates with external devices via a 4 wires connector. The transmission is made via TTL levels and it uses a standard RS232 protocol at 115200 bps, so interfacing the device to a PC is really immediate.

- **(PC_Phase)**

The next picture presents how the data glove is connected with pc using TTL to USB adapter cable:

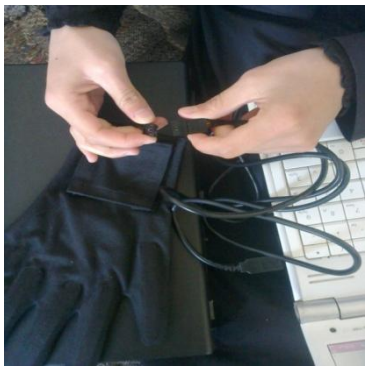


Figure 5.6 Connecting data glove with PC.

Get the COM Port number which the data glove is connected to, as shown in the next picture.

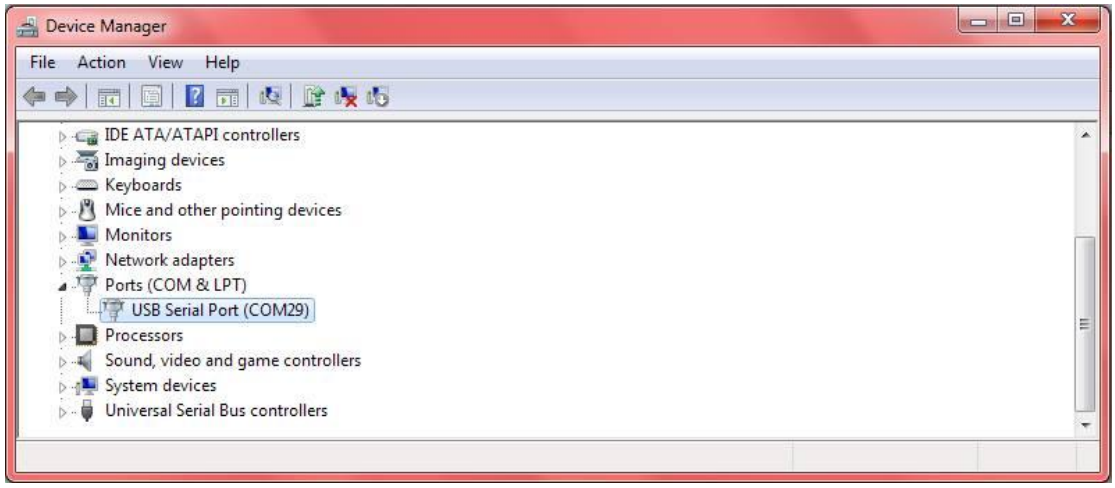


Figure 5.7 The Com Port Number

The Serial Port settings for the data glove are clarified in the next table.

Table 5.1 Serial Port Setting of data glove

Baud Rate	115200 BPS
Data Bit	8
Stop Bit	1
Parity	NONE

Adjust the baud rate to 115200 BPS and the port to 29 as shown in the next figure.

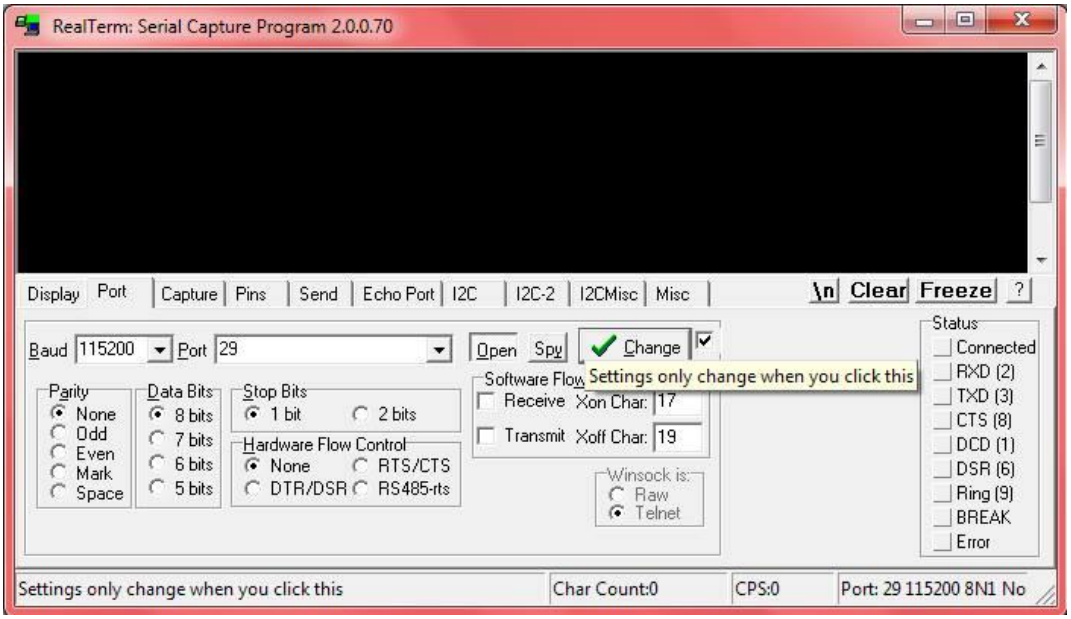


Figure 5.8 RealTerm configurations for the data glove.

Send 's' to the glove to test if it starts transmission, the glove must transmit packages continuously:

Figure 5.9 Send command by RealTerm Program.

The RealTerm program start reads the input data then present it as it shown in the next picture. The glove continuously transmits to the host device the following 20 byte package:

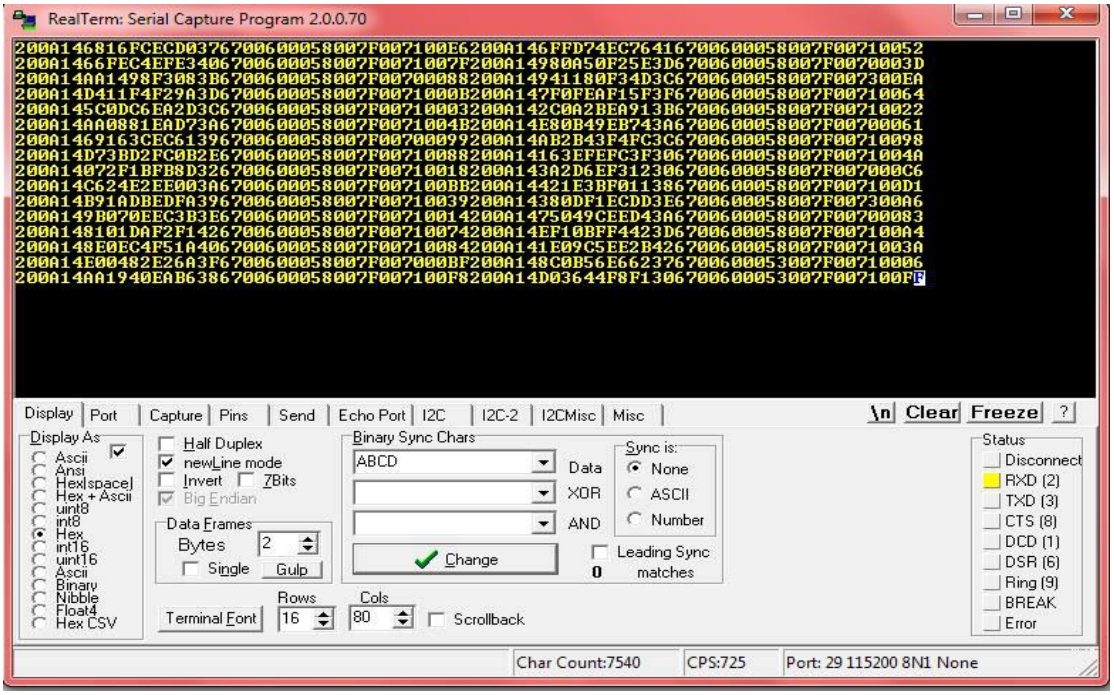


Figure 5.10 Reading data by RealTerm Program

After testing the data glove we noticed that it gave data ranges as expected, and communication with data glove was successful in PC phase.

- **(Arduino_Phase)**

The data glove connected to the arduino microcontroller using 4 wires connected through the TTL_Connector as shown in the next picture.

Figure 5.11 Connecting data glove to Arduino microcontroller

Then the arduino is connected to the PC via USB cable, the data glove sends the data to the arduino which process it then the data is viewed on the PC screen as shown in the picture bellow.

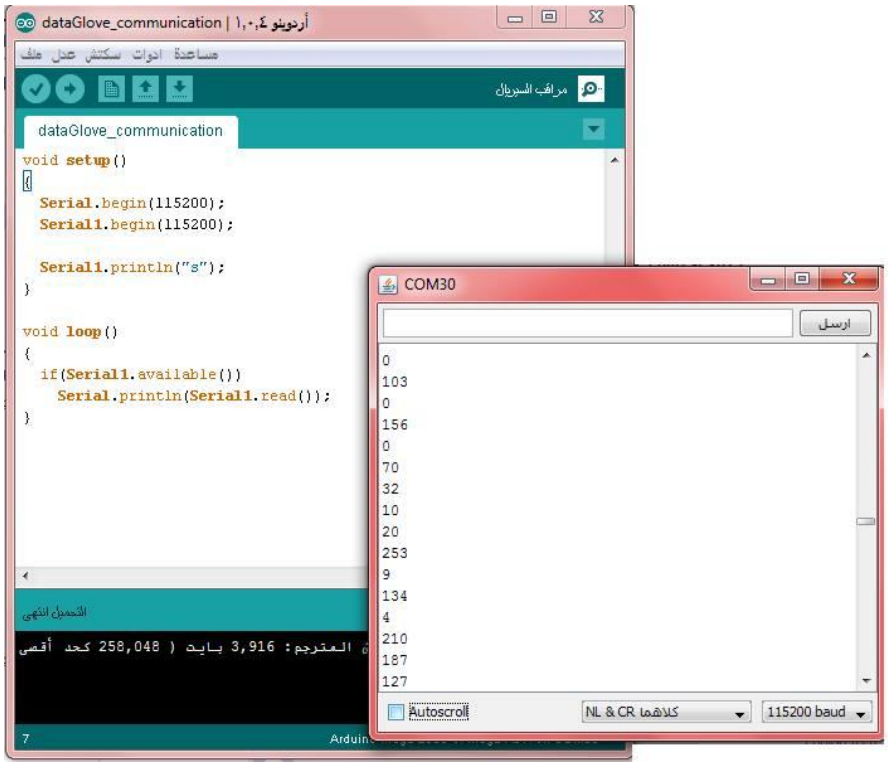


Figure 5.12 Testing data glove communication with Arduino

After testing the data glove we noticed that it gave data ranges as expected, and communication between arduino and data glove was successful in microcontroller phase.

5.3.2 Arduino Test:

We have tested our arduino mega board using a simple code that makes a led connected to one of the digital pins blinks.

Steps:

1. install the arduino IDE software
2. Connect the board to the computer using usb cable.
3. Install drivers for arduino mega 2560 and configure with windows 7 in this case.
4. Write the sketch (code) and upload it after selecting the correct com port number and the board name.

The testing code was as configured in the next picture.

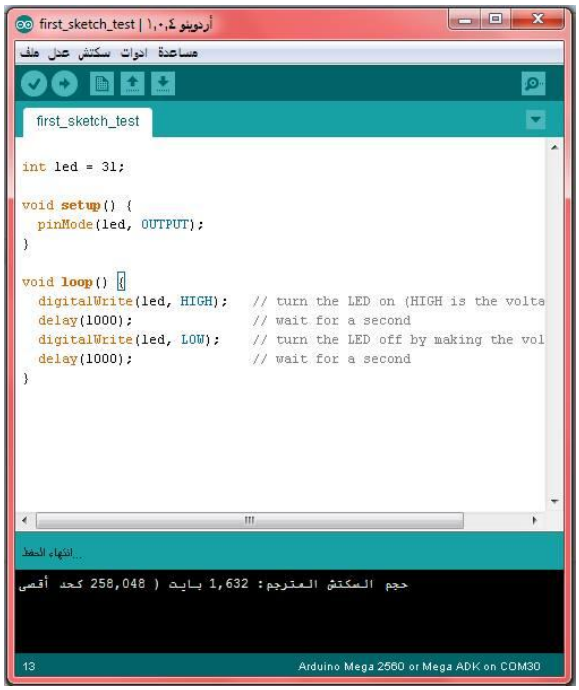


Figure 5.13 Testing code for arduino

The experiment was successful and the result was as shown in the next figure.

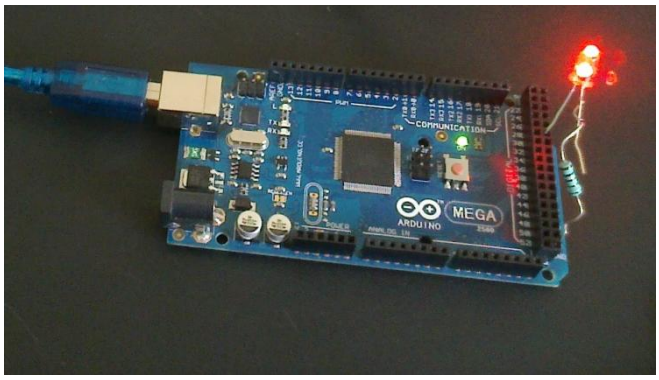


Figure 5.14 Arduino first experiment result.

5.3.3 GLCD Test:

The LCD was tested after downloaded the FirmWare first by giving it orders to display characters, then it was tested by sending words in English as shown in the next picture.

Figure 5.15 Testing the GLCD

The arduino code was written so that the GLCD can display English letters and Words through orders from arduino, the next picture shows the arduino code for the GLCD.

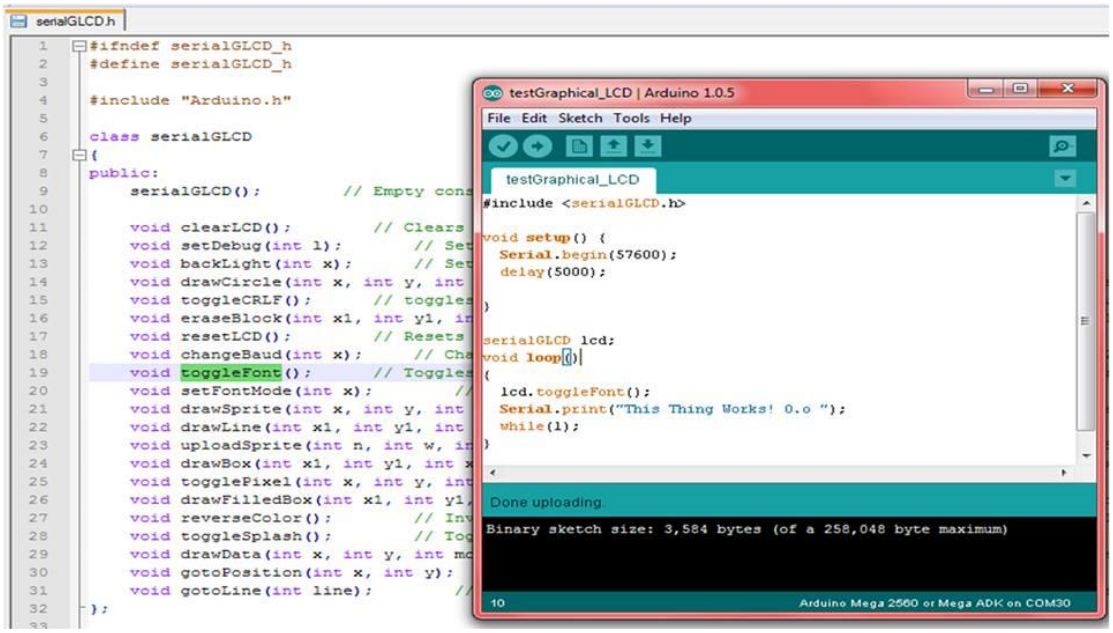


Figure 5.16 The GLCD arduino code.

5.4 Unit Implementation

This section presents the implementation for each individual unit of the system with the methods and algorithms that are needed for project requirements.

5.4.1 Data Glove Implementation

In this section, there is a description of how the data glove is implemented, to get the gestures readings, on both PC and Arduino microcontroller phases. The data glove used in this system is DG5 VHand 2.0 as mentioned in Hardware components section (2.3).

The glove uses a standard RS232 protocol at 115200 bps and it continuously transmits to the host device a 20 byte package ordered as following:

1 - header = 0x20
2 - header = 0x0A
3 - length = 0x14 (20 byte)
4 - acceleration axis ax_l
5 – acceleration axis ax_h
6 - acceleration axis ay_l
7 - acceleration axis ay_h
8 - acceleration axis az_l
9 - acceleration axis az_h
10- bend 0_l
11 - bend 0_h
12 - bend 1_l
13- bend 1_h
14- bend 2_l
15- bend 2_h
16 - bend 3_l
17 - bend 3_h
18 - bend 4_l
19 - bend 4_h
20 – crc

This is what we have seen before when tested the glove using RealTerm terminal; series of 20 byte package. The number of data samples collected per minute is around 1500 (25Hz sampling rate). The next step is to interpret those bytes, read raw values, and then convert those to their actual meaningful values.

The data glove generates two types of sensor readings:

1. Fingers Readings.
2. Triple-Accelerometer Readings.

Here the explanation for each of them:

1. Fingers readings:

The data glove incorporates 5 sensors on each of the right-hand five fingers to be able to measure the fingers bending. Finger reading is a 10 bit resolution value that transmitted as two bytes (low and high). The maximum value of the sensor is 1024. So the finger values are from 0 to 1024. It was mapped to be from 0 to 100 using the following relations:

$$VALUE = RAW . \frac{RANGE}{2^{resolution}} \quad \text{,unsigned raw value} \qquad (5.1)$$

$bend_i = bend_{i\_low} + (finger_{i\_high} \,\&\, 0x03) \ll 8) \quad : 0 \leq i \leq 4$

$bend_i = bend_i * 100 / 1024$

Where i is finger index described it the table 5.2 below for the right-hand data glove used in the system.

Table 5.2 The fingers index for the data glove.

I	0	1	2	3	4
Finger	Thumb	Index	Middle	Ring	Little

2. Triple-accelerometer readings:

The glove incorporates a 3 axes accelerometer able to measure the hand acceleration along the three main axes. The acceleration values are from -32767 to +32767 that correspond to hand acceleration measurement from -2g to 2g.
The relative accelerations can be computed with the following relations:

$$VALUE = RAW \cdot \frac{RANGE}{2^{resolution\ -1}} \quad \text{,signed raw value} \qquad (5.2)$$

$accel_i = accel_{i\_low} + (accel_{i\_high} \,\&\, 0x7F) \ll 8)$: i = {x, y, z}

$accel_i = accel_i * sign(accel_{i\_high}) * 2 / 32767$

Here is the pseudo code of the main function to implement the glove:

- **Function main()**

```
{
   Port = create port connection
   Set baudrate and serial configurations of Port

   if connection fails
      print "fails to connect"
      exit
   end

   //comment:
   //start sampling
   Port.write_byte("s")
   while(data_is_available)
   {
      read_package_raw_values()
      convert_to_values()
      process_data()
   }end while

   //comment:
   //end sampling
   Port.write_byte("e")
}
```

- **Function read_package_raw_values()**

```
{
//comment:
//0x200A14 header must be received then 17 bytes must be received after it //to
complete the reading package.
Byte = Port.read_coming_byte
if Byte is 0x20
   Byte = Port.read_coming_byte()
   if Byte is 0x0A
      Byte = Port.read_coming_byte()
         if Byte is 0x14
            accel_xl = Port.read_coming_byte()
            accel_xh = Port.read_coming_byte()
            accel_yl = Port.read_coming_byte()
            accel_yh = Port.read_coming_byte()
            accel_zl = Port.read_coming_byte()
            accel_zh = Port.read_coming_byte()

            thumb_l = Port.read_coming_byte()
            thumb_h = Port.read_coming_byte()
            index_l = Port.read_coming_byte()
            index_h = Port.read_coming_byte()
            middle_l = Port.read_coming_byte()
            middle_h = Port.read_coming_byte()
            ring_l = Port.read_coming_byte()
            ring_h = Port.read_coming_byte()
            little_l = Port.read_coming_byte()
            little_h = Port.read_coming_byte()

         end
      end
   end
}
```

- **function convert_to_values()**

```
{
   //comment:
   //accelerometer value ranges from -2g to +2g
   accel_x = (accel_xl + (accel_xh ≪ 8)) * 2 / 32767
   accel_y = (accel_yl + (accel_yh ≪ 8)) * 2 / 32767
   accel_z = (accel_zl + (accel_zh ≪ 8)) * 2 / 32767

   //comment:
   //just 10-bit resolution per finger
   //finger value ranges from 0 to 100
   thumb = (thumb_l + ((thumb_h & 0x03)≪ 8)) * 100 / 1024
   index = (index_l + ((index_h & 0x03) ≪ 8)) * 100 / 1024
   middle = (middle_l + ((middle_h & 0x03) ≪ 8)) * 100 / 1024
   ring = (ring_l + ((ring_h &0x03) ≪ 8)) * 100 / 1024
   little = (little_l + ((little_h & 0x03) ≪ 8)) * 100 / 1024
}
```

In this system, it was assumed that roll and pitch in addition to the fingers bend will be good features representing the calculated gestures from the glove. In order to extract roll and pitch orientations, there some tests that need to be done to understand what the accelerometer reads exactly.

Testing accelerometer:

As we have searched and noticed, the accelerations can be divided in two types: dynamic and static. The 3 axes accelerometer measures both the types. A dynamic acceleration is a measure of the instantaneous movement of the hand along the three axes. Static accelerations are due to the gravity force, for example rolling the hand around the y axis (roll) will modify the acceleration measured along the x axis and along the z axis, since the component of the gravity along these axes is modified, then the pitch angle can be deduced, for example, by getting the static measure of the x acceleration. Vice versa, rotating the hand along the x axis (pitch) will modify the measured y acceleration, so the pitch angle can be measured [7] [8].

A check should be made while watching the raw accelerometer data from the glove to obtain right orientation relations:

First test:

The glove placed horizontally flat on the table. The z-axis accelerometer reads -1g and the x and y axes negligible values. Invert the glove so that the z-axis points downwards and verify that the z-axis accelerometer now indicates +1g. Repeat with the y-axis pointing downwards and then upwards to check that the y-axis reports -1g and then reports +1g. Repeat once more with the x-axis pointing downwards and then upwards to check that the x-axis reports -1g and then +1g.
Comparing the obtained results with data glove axes reference shown in its datasheet, it seemed that the accelerometer z-axis output Z, is correctly aligned, but the x-axis Z and y-axis Y signals are inverted in sign.

Second test:

A drop test was also performed. The glove was hold steadily, and then dropped from some height to apply free-fall on it. The result is shown in figure 5.17 bellow:

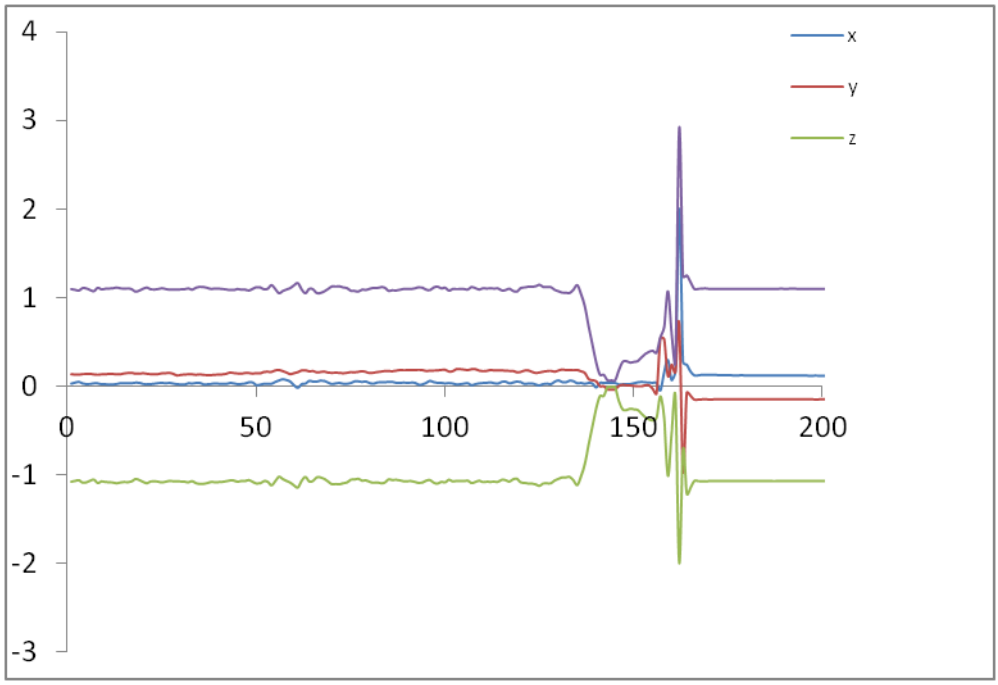


Figure 5.17 The result of the drop test

The glove read magnitude 1 before dropping, 0 during the drop, and a large spike for the large deceleration when the glove struck the ground.

Now all are clear, a positive pitch angle θ and positive roll angle φ are defined as clockwise rotations about the positive x- and positive y-axes respectively as in Figure 7.16 bellow. Roll and pitch relations will be as:

$$pitch = \tan^{-1}\frac{Gy}{\sqrt{Gz^2+Gx^2}} \quad (5.3)$$

$$roll = \tan^{-1}\frac{Gx}{-Gz} \quad (5.4)$$

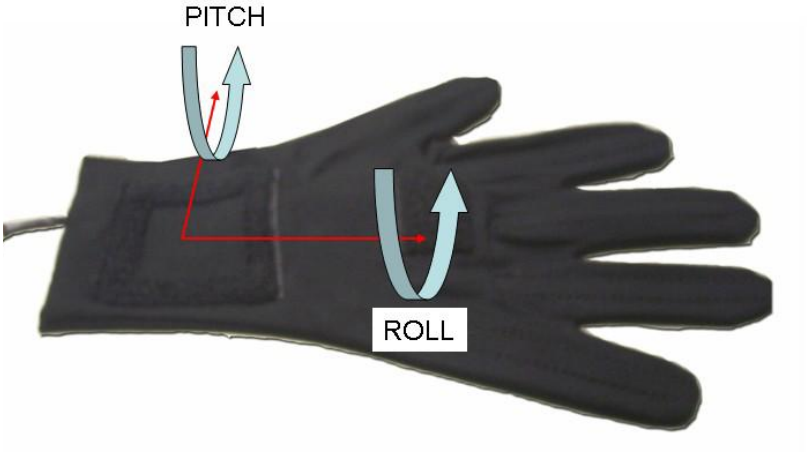


Figure 5.18 Roll and Pitch for the data glove

The denominator of the pitch equation is defined to be always positive, so the equation itself only provides [-90,+90]range, which is exactly what is expected for the pitch angle. In contrast, the roll equation provides [-180,+180]range. but a problem occurs when the pitch angle is *90º*, the y-axis (roll) is directly aligned with the gravity vector, thus we cannot measure the roll angle anymore.

Also, roll equation is undefined when both Gx and Gz are equal to zero, and that for each possible value of the calculation done inside the tan^{-1} function there are two valid solutions, not only on the roll but also on the pitch equation. These problems were solved in code by using the function *atan2*, which eliminates the angle calculation ambiguity by taking into account the quadrant. Atan2 function is found in <cmath> or <math.h> c++ library.

- **function extract_orientation_values()**

```
{
   roll = atan2( accel_x , -accel_z)
   pitch = atan2( accel_y , sqrt( accel_x * accel_x + accel_z * accel_z) )
}
```

- **PC phase:**

The data glove device is connected using a TTL to USB cable via a COM port to a computer, detected as "COM29" on the main used computer to implement the system, and at backhand a visual studio windows forms C++ application is used to read the data from sensors and record the represented signs, words, to a directory groups the training set files to train the system on it later.

```
DCB dcb;
BOOL fSuccess;
COMMTIMEOUTS CommTimeouts;
HANDLE hCom;
fp = fopen (pathname,"w");
int TimePakIn=0;
int TimePakfin=0;
int Time=0;
int TimeByte1Fin=0;
int TimeByteIn=0;
int TimeByteFin=0;

LPCWSTR pcCommPort = _T("\\\\.\\COM29");
hCom = CreateFile( pcCommPort,
   GENERIC_READ | GENERIC_WRITE ,
   0, // must be opened with exclusive-access
   NULL, // no security attributes
   OPEN_EXISTING, // must use OPEN_EXISTING
   0, // not overlapped I/O
   NULL // hTemplate must be NULL for comm devices
   );
if (hCom == INVALID_HANDLE_VALUE)
{
   // handle the error
   printf("cannot open serial port!");
   return 1;
}
// Build on the current configuration, and skip setting the size
// of the input and output buffers with SetupComm.
fSuccess = GetCommState(hCom, &dcb);
if (!fSuccess)
{
   printf("cannot open serial port!");
   return 1;
}
// Fill in DCB: 115,200 bps, 8 data bits, no parity, and 1 stop bit.
dcb.BaudRate = CBR_115200; // set the baud rate (sembra che non cambi niente)
dcb.ByteSize = 8; // data size, xmit, and rcv
dcb.Parity = NOPARITY; // no parity bit
dcb.StopBits = ONESTOPBIT; // one stop bit
fSuccess = SetCommState(hCom, &dcb);
if (!fSuccess)
{
   printf("cannot open serial port");
   return 1;
}
CommTimeouts.ReadIntervalTimeout=100;
```

```
CommTimeouts.ReadTotalTimeoutMultiplier=0;
CommTimeouts.ReadTotalTimeoutConstant=4000;
CommTimeouts.WriteTotalTimeoutConstant=0;
CommTimeouts.WriteTotalTimeoutMultiplier=0;
SetCommTimeouts(hCom,&CommTimeouts);
printf("Connected to port Successfully!");
DWORD dwBytesTransferred;
```

After that, when the port is opened successfully, transmit 's' ASCII character to the data glove to start sampling from it. As shown the next figure.

```
// START SAMPLING
WriteFile(hCom,"s",1,&dwBytesTransferred,0);
```

From the point clicking on 'start recording' button until clicking on 'stop recording' one, the application will repeatedly detect if new reading is available. For each reading, extract feature vector of the gesture and then append it as a comma-separated line to the file that created to record the represented sign some signer. Figures 5.19 to 5.21 show the graphical user interface of this application.

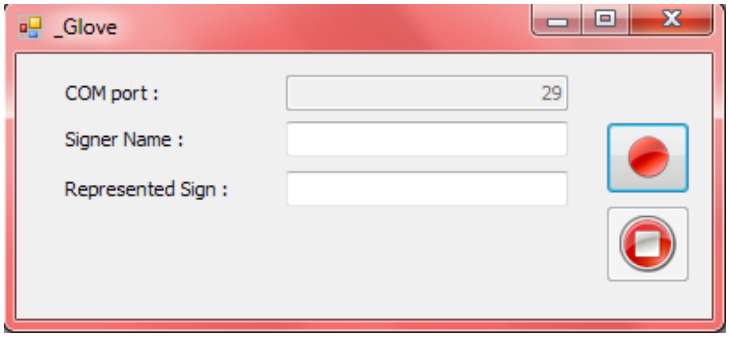


Figure 5.19 The Graphical Interface of the PC data glove application.

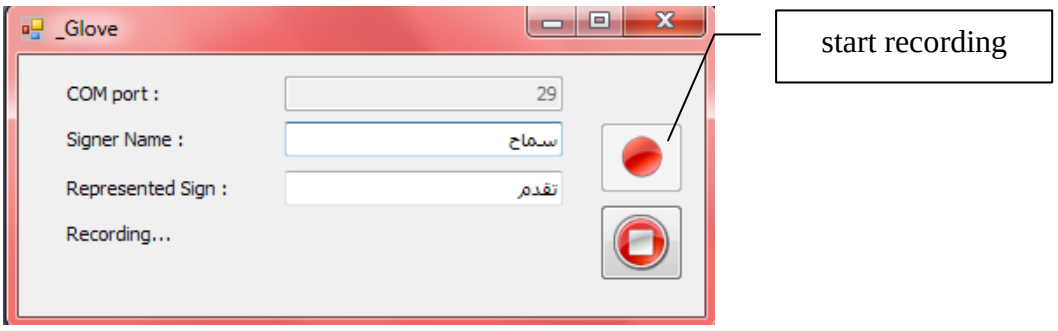


Figure 5.20 Start Recoding fro data glove.

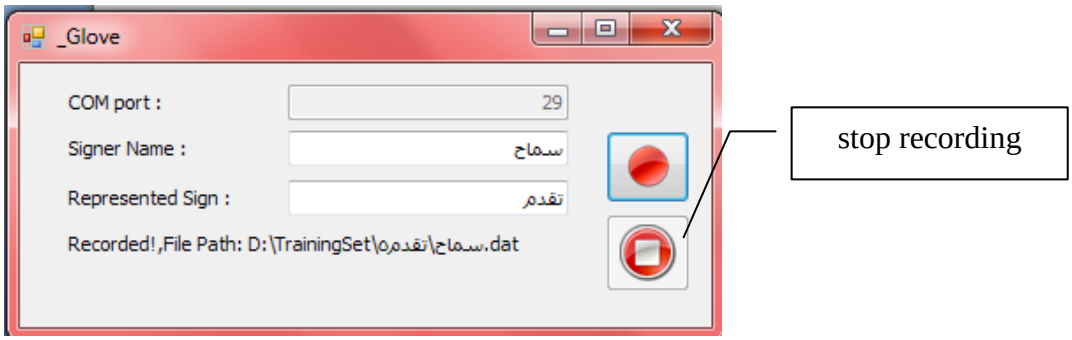


Figure 5.21 Stop Recording from data glove.

- **Arduino Microcontroller phase:**

The data glove device is connected serially to one of four the HardwareSerial Ports of the Mega2568 board, chosen to be Serial1 with Rx1 and Tx1, and at backhand an arduino sketch project was implemented to read the data from sensors and append their feature vector to the observed sequences stored as a circular buffer. These sequences will go through the recognition process afterward.

```
HardwareSerial& _PC=Serial;// RX0 TX0
HardwareSerial& _DataGlove=Serial1;// RX1 TX1

//Serial Communication Configuration
const unsigned long _BaudRate=115200;//Baud Rate: 115200 BPS
const uint8_t _DG_CONFIG=SERIAL_8N1;//Data Bit: 8 //Stop Bit: 1 //Parity: NONE

//variables
float f[5];
float ax, ay,az;
float roll,pitch;
int temp;
byte Byte[20];

//circular buffer
const int _BUFFER_SIZE = 1995;
float buffer[_BUFFER_SIZE];

void setup() {
   // initialize both serial ports:
   _PC.begin(_BaudRate);
   _DataGlove.begin(_BaudRate,_DG_CONFIG);
   // start sampling from dataGlove Serial by sending it 's'..
   _DataGlove.println("s");

   for (unsigned i=0;i<5;i++)
   {
```

```
      f[i]=0.0;
   }

   for (int i=0;i<_BUFFER_SIZE;i++)
   {
      buffer[i] = 0.0;
   }

   ax =  ay = az  = 0.0 ;
   roll =  pitch = 0.0 ;
   temp = 0;
   Byte[0]=0;
}//end of setup

void loop()
{
}

void serialEvent1()
{
    while (_DataGlove.available()) {
      _DataGlove.readBytes((char*)Byte,1);
     if (Byte[0]==0x20)
     {
        _DataGlove.readBytes((char*)&Byte[1],1);
        if (Byte[0]==0x0a)
        {
           _DataGlove.readBytes((char*)&Byte[2],1);
           if (Byte[0]==0x14)
           {
              _DataGlove.readBytes((char*)&Byte[3],17);
              //_PC.write("Header found\n");
              temp = Byte[3]+((Byte[4]&0x7f)<<8);
              if (Byte[4]&0x80)
                 temp = temp-32767 ;
              ax = temp * 2.0f / 32767.0f;

              temp = Byte[5]+((Byte[6]&0x7f)<<8);
              if (Byte[6]&0x80)
                 temp = temp-32767 ;
              ay = temp * 2.0f / 32767.0f;

              temp = Byte[7]+((Byte[8]&0x7f)<<8);
              if (Byte[8]&0x80)
                 temp = temp-32767 ;
              az = temp * 2.0f / 32767.0f;

              for (byte i =0 ;i<5 ;i++)
              {
                 temp = Byte[9 + 2*i] + ((Byte[10+2*i]&0x03)<<8);
                 f[i] = temp*100.0f/1024.0f;
              }

              roll = (atan2( ax, -az)*180.0f)/3.14f;
              pitch  = (atan2( ay, sqrt( ax * ax + az * az))*180.0f)/3.14f;

              buffer[7*(_BUFFER_SIZE-1)+0] = roll;
              buffer[7*(_BUFFER_SIZE-1)+1] = pitch;

              for(byte i=0 ; i<3 ; i++ )
                 buffer[7*(_BUFFER_SIZE-1)+2+i] = f[i];
```

```
                for (int i=0;i<_BUFFER_SIZE;i++)
                {
                   for (int j=0;j<7;j++)
                      buffer[7*i+j] = buffer[7*(i+1)+j];
                }


                _PC.print(roll);
                _PC.print("\t");
                _PC.print(pitch);
                _PC.print("\t");

                for(byte i =0 ; i<5 ; i++)
                {
                   _PC.print(f[i]);
                   _PC.print("\t");
                }
                _PC.print("\n");
             }
          }
       }
    }
}
```

5.4.2 GLCD Implementation

The Serial Graphic LCD backpack provides the user a simple serial interface to a full range of controls beside writing text, draw lines, circles and boxes ,set or reset individual pixels, erase specific blocks of the display and control the backlight.
In this system an Arabic text will be displayed on the lcd to show the series of words corresponding to the recognized gestures models that the user signs with a good response on the little screen. To do that, the firmware of that GLCD was updated to improve processing speed.

Fortunately, a good updated version of the firmware was found on the web [9] [10]. Then uploaded to the ATMega168 processor of the serial backpack. Arduino Mega board used to do this stage and imlemented to function as an AVR programmer.

The connection of Arduino as an In System Programmer (ISP) with the serial lcd is shown in figure 5.22 and figure 5.23 bellow. number of steps follwed and command on the terminal as shown in figure 5.24 must be done to initiallize AVR device and flash the hex file of the already compiled firmware.

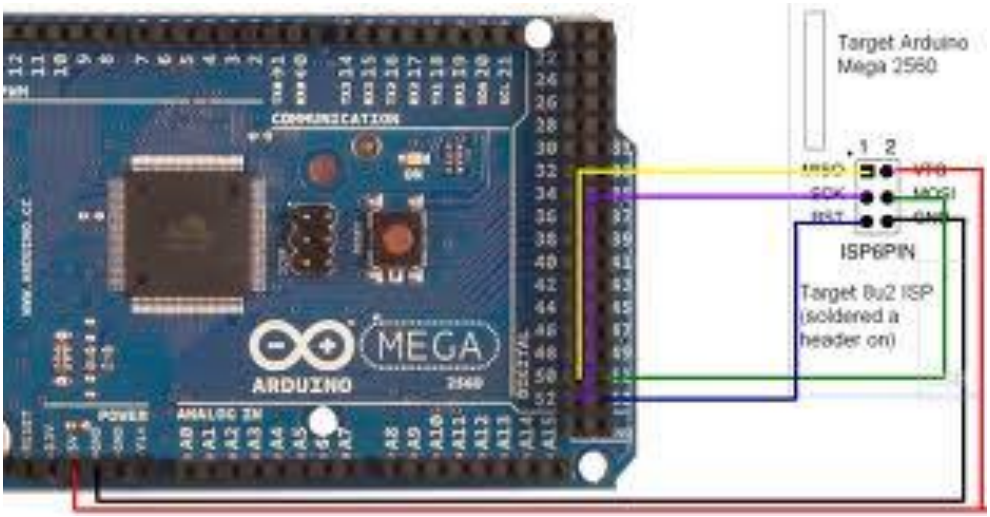


Figure 5.22 Connecting Arduino with the serial GLCD.

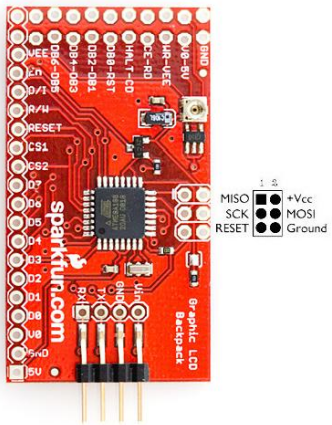


Figure 5.23 The connection of the serial GLCD with Arduino.

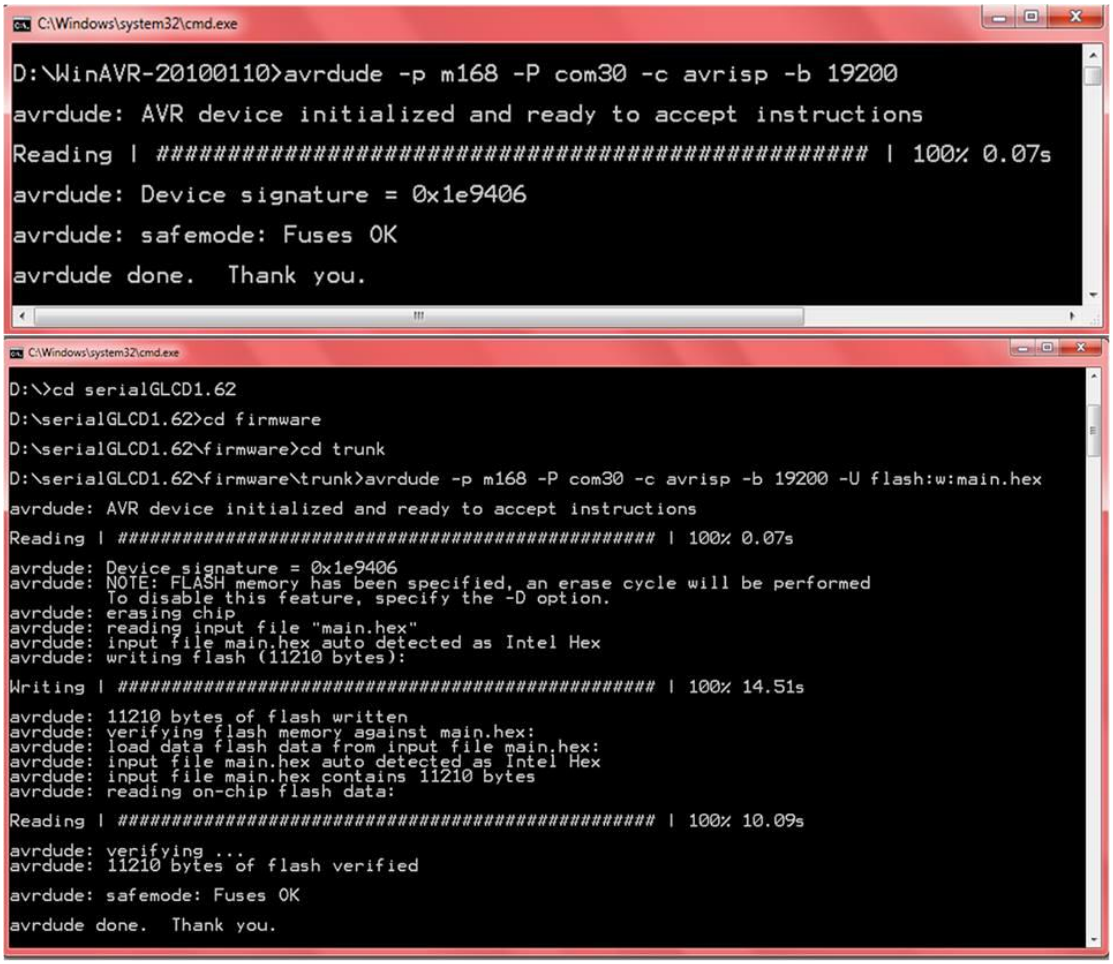


Figure 5.24 GLCD commands on the terminal

In this system, Arabic sign language recognition is the goal. so the text displayed on the screen is intended to be in Arabic. The first step to do that is to prepare an Arabic LCD font, this was done using some bitmap utility. Subpart of the font matrix of Arabic alphabets the LCD of the system would have is as follow:

```
0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x00, //199/ --> ا
0x00,0x30,0x28,0x20,0xA0,0x20,0x28,0x30, //200/ --> ب
0x00,0x00,0x30,0x2A,0x28,0x32,0x00,0x00, //201/ --> ق
0x00,0x30,0x28,0x22,0x20,0x22,0x28,0x30, //202/ --> ت
0x00,0x30,0x28,0x22,0x21,0x22,0x28,0x30, //203/ --> ث
0x00,0xC0,0xA8,0xA8,0x28,0xB0,0x20,0x20, //204/ --> ج
0x00,0xC0,0xA8,0xA8,0xA8,0x30,0x20,0x20, //205/ --> ح
0x00,0xC0,0xA8,0xAA,0x28,0x30,0x20,0x20, //206/ --> خ
0x00,0x00,0x24,0x24,0x24,0x38,0x00,0x00, //207/ --> د
0x00,0x00,0x24,0x25,0x24,0x38,0x00,0x00, //208/ --> ذ
0x00,0x80,0x80,0x40,0x30,0x00,0x00,0x00, //209/ --> ر
0x00,0x00,0x80,0x80,0x40,0x34,0x00,0x00, //210/ --> ز
0x60,0x80,0x80,0x78,0x20,0x38,0x20,0x18, //211/ --> س
0x60,0x80,0x80,0x78,0x22,0x39,0x22,0x18, //212/ --> ش
0x60,0x80,0x80,0x60,0x30,0x28,0x28,0x18, //213/ --> ص
0x60,0x80,0x80,0x60,0x30,0x28,0x2A,0x18, //214/ --> ض
0x00,0x22,0x14,0x08,0x14,0x22,0x00,0x00, //215/ --> ×
0x20,0x20,0x3E,0x30,0x28,0x28,0x18,0x00, //216/ --> ط
0x20,0x20,0x3E,0x30,0x28,0x2A,0x18,0x00, //217/ --> ظ
0x00,0x00,0x40,0xA0,0xB0,0x28,0x28,0x00, //218/ --> ع
0x00,0x00,0x40,0xA0,0xB0,0x2A,0x28,0x00, //219/ --> غ
0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20, //220/ --> ـ
0x00,0x18,0x20,0x20,0x30,0x28,0x2A,0x30, //221/ --> ف
0x00,0x60,0x80,0x80,0xB2,0xA8,0x7A,0x00, //222/ --> ق
0x00,0x30,0x28,0x2C,0x2A,0x20,0x3F,0x00, //223/ --> ك
0x00,0x40,0xA9,0xAA,0xA8,0xF0,0x00,0x00, //224/ --> à
0x00,0x00,0x60,0x80,0x80,0x7E,0x00,0x00, //225/ --> ل
0x00,0x40,0xAA,0xA9,0xAA,0xF0,0x00,0x00, //226/ --> â
0x00,0x00,0xC0,0x20,0x30,0x28,0x28,0x30, //227/ --> م
0x00,0x00,0x60,0x80,0x88,0x80,0x60,0x00, //228/ --> ن
0x00,0x00,0x30,0x28,0x28,0x30,0x00,0x00, //229/ --> ه
0x00,0x00,0x00,0xB0,0xA8,0x78,0x00,0x00, //230/ --> و
```

After implementing these codes; for GLCD the result was as shown in the next picture.

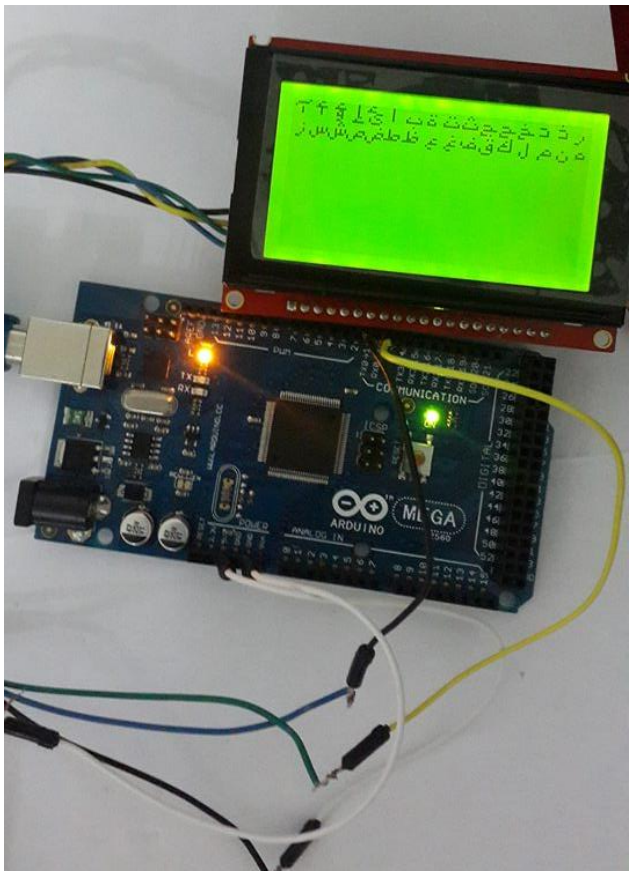


Figure 5.25 Arabic Letters implemented on the GLCD.

An arduino sketch was implemented to be able to display the text of each of the three models, when its gesture sign recognized by the system, on the LCD as follow:

```
HardwareSerial& _LCD=Serial;// RX0 TX0
#include <serialGLCD.h> // Include the library
byte model1[]={
0x00,0x18,0x20,0x20,0x30,0x28,0x2A,0x30,
0x20,0x20,0x20,0x20,0x30,0x2A,0x28,0x32
};
byte model2[]={
0x80,0x40,0x30,0x48,0x48,0x30,0x20,0x20,
0x00,0x24,0x24,0x24,0x38,0x20,0x20,0x20,
0x20,0x20,0x30,0x2A,0x28,0x32,0x20,0x20,
0x20,0x20,0x20,0x22,0x20,0x22,0x28,0x30
};
byte model3[]={
0x00,0x3F,0x20,0x20,
0x20,0x20,0x20,0x20,0x30,0x28,0x28,0x00,
0x00,0x00,0x00,0x3F,0x00,
0x00,0x00,0x24,0x24,0x24,0x38,0x00,
0x00,0x00,0x00,0xB0,0xA8,0x78,0x00,0x00,
};
```

```
char* models[]={"قف" , "تقدم" , "وداعا"};
void setup() {
  Serial.begin(57600); //default baudrate of the display, can be changed, //consult
  summoningdark's README that comes with the firmware to change it
    delay(5000);
  lcd.uploadSprite(1,16,8,model1);
  lcd.uploadSprite(2,32,8,model2);
  lcd.uploadSprite(3,32,8,model3);

}

serialGLCD lcd; // initialisation

void loop() {

  lcd.clearLCD();
   // Allow for a short delay after clearLCD since it takes some time for the
   //backpack to clear the entire screen
  delay(10);
  lcd.drawSprite(0,0,1,4);
  lcd.drawSprite(0,9,2,4);
  lcd.drawSprite(0,18,3,4);
  while(true);
}
```

The GLCD test was successful and it displays Arabic words as expected. The result is shown in the picture bellow.

Figure 5.26 Arabic words implementation on GLCD.

```
void showModel(int index)
{
   //this function is to show the text corresponding to model number of the
   //recognized sign (word) passed as index parameter on the LCD.
  switch(model_index)
  {
     case 0: lcd.drawSprite(0,0,1,4);break;
     case 1: lcd.drawSprite(0,9,2,4);break;
     case 2: lcd.drawSprite(0,18,3,4);break;
  }
}
```

5.4.3 Voice implementation

For this system, a TTS chip was needed to perform the translation from the Arabic text into its matching voice, this chip was ordered from its manufacturer in the United States of America, unfortunately this chip was not delivered because of the security examination by the occupation forces; that's why an alternative is needed. A dictionary of just three selected words was made and that was enough for now.

The alternative way was using the arduino mega to play those simple three audio waves:

1. MP3 audio files of the three words were downloaded from Google Translate TTS, these are the used links:

```
http://translate.google.com/translate_tts?ie=UTF-8&q=قَفَّدْمُ&tl=ar
```

```
http://translate.google.com/translate_tts?ie=UTF-8&q=قِفْ&tl=ar
```

```
http://translate.google.com/translate_tts?ie=UTF-8&q=وَدَاعَاً&tl=ar
```

2. The sounds converted to .wav sounds.
3. The BTc Sound Encoder v3.0is used to encode the .wav at a very low bit rate to conserve memory of the microcontroller using BTc16 1bit algorithm program option.
4. On arduino mega chip PCMAudio free source example used to play back the models speech on an external 8 ohm loud speaker.

5.4.4 Recognition Phase System Implementation

The recognition phase for the entire system that was performed on the arduino microcontroller is shown bellow.

```
import models
// HMMs.h file of trained models information (output of the system matlab
training library)

Dataglove glove
LCD lcd
Circular_buffer buffer[HMMs._BUFFER]
int buffersize = 0
```

```
float maxLogLikelihood
unsigned current_model_index
unsigned previous_model_index
int maxRepeat = 10
int repeat = 0

while glove.available()
   glove.elaboratoe_values()
   buffer.push(HMM_quantize(glove.f,glove))
   buffersize = buffersize + 1
   if buffersize == HMMs._BUFFER
      buffer.shift()
   end
   model_index = HMM_calculate(buffer,&maxLogLikelihood)
   if current_model_index != previous_model_index OR repeat == maxRepeat
      if maxLogLikelihood >= models[current_model_index].threshold
         txt = models[current_model_index].label
         showModel(txt)
         sayModel(txt)
      end
      repeat = 0
   end
end
```

5.5 HMM and Training Phase Methods

A Matlab library to do the training phase was implemented and it contains number of functions and methods to the process. See Appendix B for the full functions.

1] Quantization:

We united the status range from 0 to 100 for all columns including the roll and pitch columns. Then, converted the continuous set of values to a relatively small discrete set ranged from one to the given quantization number. And convert the data set matrix to vector of sequences. The quantization number will be assumed the same as number of emissions of the models.

Quantization number is optimized by repeatedly train and test the models on different cases. The case with less source error (misclassification) is relied. The quantization code is shown in the figure bellow.

```
function Q = SLR_quantize( data , T )
    %Quantization is the procedure of constraining something from a
    %continuous set of values (such as the real numbers) to a relatively small discrete set (such as the integers).

    %T : Quantization Number.
    % returns [ 1 X length(data) ] 'Q' Quantized Vector.

    Q = data;

    %the first two columns of 'data' should be [roll pitch] which ranges
    %from continuous -90 to +90 ..
    %re-range their values into continuous 0 to 100 as the range of other fingers columns.
    Q(:,1:2) = ( Q(:,1:2) + 90 ) * ( 100/180 );

    %Quantize each data cell to T discrete values indexed [ 0 : T-1 ].
    Q = ceil( Q*T/100 );
    Q( Q <= 0 ) = 1;
    Q( Q > T ) = T;
    Q = Q';
    Q = Q(:)';
end
```

Figure 5.27 Quantization Code.

2] HMM training:

Estimates the transition and emission probabilities for a Hidden Markov Model, from quantized training sequences, using the Baum-Welch algorithm. The initial transition and emission probability matrices are generated randomly. And the next figure presents the code for training algorithm.

```
function [ estTR, estE ] = SLR_train( seqs , tr ,e )
    %do hmmtrain on the given training sequences (seqs).
    [ estTR, estE ] = hmmtrain(seqs,tr,e);

    %then,
    %get rid of zeros and very small values (near zero) in either estTR or estE to deal with any
    %possible NaN values problem in the logLikeHood outputed from the evaluate algorithm.

    %min = small non-zero value
    %1e-323 : noticed, by testing, to be the minimum positive number that doesn't produce NaN when parsed to Matlab 'log(x)' function..

    min = 1e-323; %note for ==ME== :- if NaN problem still exist, replace min with min = 1e-10;
    estTR( estTR < min ) = min;
    estE( estE < min ) = min;

    %normalize estTR and estE.
    estTR = bsxfun(@rdivide, estTR, sum(estTR,2));
    estE = bsxfun(@rdivide, estE, sum(estE,2));

end

```

Figure 5.28 HMM training code.

3] Cross Validation:

Is a classification technique preventing over fitting and generalization error, we need these technique for optimizing number of states, We choose a 10- fold because for each class we have 15 sequences given in training set (5 sequences from each of the 3 signers) i.e. the k fold value depends on the number of signs represents

from the signer of the same word. So we choose 12 sequences for training and 3 for test.

And then calculate the source error with different number of state then choose the number of states corresponding with minimum source error.

4] HMM Evaluate:

Calculates the log likelihood given an observation sequence and a model specified by its parameters (transition and emission probabilities) using the scalar forward algorithm.

But NAN problem with hmm decode of matlab:

Using matlab HMM functions to train and test our recognition system ,a problem of getting 'NAN' values as log likelihood for some classes when using hmm decode function. It seems to return NAN if there are zeros or small values very near to zeros in either transition or emission matrices of this hmm classes.

Solution for this problem:

Use logical indices to replace each of these very small values with a small non zero values.

The code bellow was used on the arduino to evaluate the likelyhood for each model with the function Evaluate, and then it calculated the maximum likelyhood for all models with the function calculate.

Arduino Code for Evaluating the model likelyhood and the maximum likelyhood.

```
const byte _STATES = 3;
const byte _EMISSIONS = 10;
const byte _MODELS = 3;
const float TR[_MODELS*_STATES][_STATES];
const float E[_MODELS*_STATES][_EMISSIONS];

float getIndexOfMaximumValue(float array[], word size){
   byte maxIndex = 0;
   float max = array[maxIndex];
   for (byte i=1; i<size; i++){
      if (max<array[i]){
         max = array[i];
         maxIndex = i;
      }
   }
   return maxIndex;
}

float evaluate( byte seq[], word SIZE, byte HMMindex )
{
   /*
   evaluates sequence 'seq' to a given HMM with 'tr' and 'e' probabilities.
   */
   SIZE = SIZE +1;

   float fs[_STATES][SIZE]={0};
   fs[0][0]=1;
   float s[SIZE]={0};
   s[0]=1;

   byte start_index = HMMindex*_STATES;
   for(word count=1;count<SIZE;count++)
   {
      for(byte state=0;state<_STATES;state++)
      {
         float sum=0;
         for(byte i=0;i<_STATES;i++)
            sum+=(fs[i][count-1]*TR[start_index+i][state]);

         fs[state][count]=E[start_index+state][seq[count-1]*sum;

      }

      for(byte i=0;i<_STATES;i++)
         s[count]+=fs[i][count];
      for(byte i=0;i<_STATES;i++)
         fs[i][count]/=s[count];

   }

   float pSeq=0;
   for(word i=0;i<SIZE;i++)
   {
      pSeq+=log(s[i]);
   }
   return pSeq;
}

byte calculate( byte seq[], word SIZE, float& maxLogLikelihood )
{
   /*
   calculates the index of the model with maximum log likelihood
```

```
   which is the best of the HMM words that describes the observed sequence
'seq'.
   */

   float logLikelihood[_MODELS] = {0};
   for ( byte model =0; model<_MODELS; model++)
   {
      logLikelihood[model] = evaluate(seq, SIZE, model );
   }

   float argmax = getIndexOfMaximumValue(logLikelihood, _MODELS);
   maxLogLikelihood = logLikelihood[argmax];

   return argmax;
```

5.6 Integrated System Implementation

The whole system implemented together as shown in the diagram.

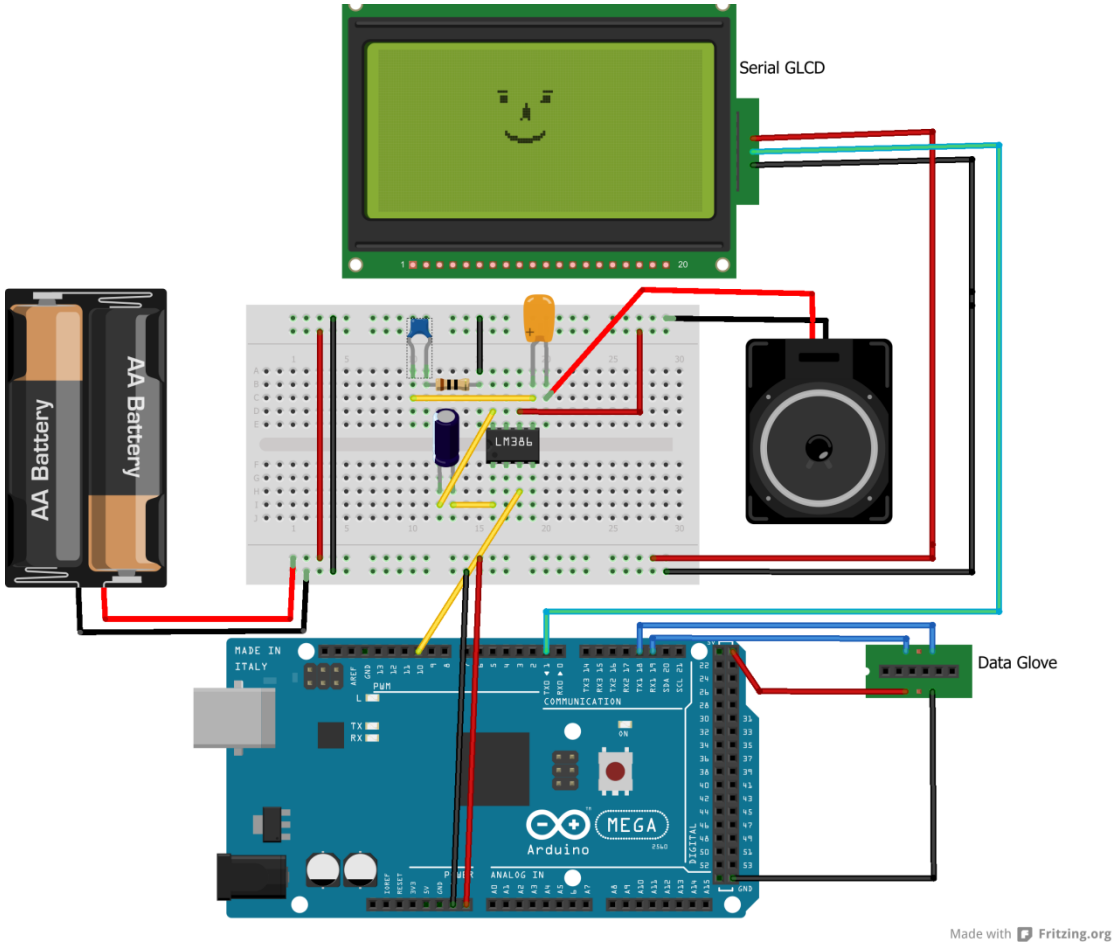


Figure 5.29 Integrated system schematic diagram

Chapter Six

Experiments and Results

6.1 Chapter Overview.

6.2 System Training Data Set.

6.3 Results.

6.1 Chapter Overview

This chapter describes in detail the data structure and feature vectors in the sequences of the gestures observation in the machine learning stage, especially the training phase. It also describes the methods used that were used during training and evaluation phases.

6.2 System Training Data Set

Each data file containing seven column in continuous time, the roll, pitch ,and the fingers status ,the last five column expressed about the finger status in percentage (0.0 to 100.0),because the glove is of right type the f[0] is the thumb while f[4] represents the small finger, also the first and second column is Roll and pitch fields which report the hand inclinations in degrees, from -90 to +90, while ax,ay, anz az represent the instantaneous accelerations of the hand, which is The hand orientations (roll and pitch) can be extracted from these information. As shown in table below:

Roll	Pich	Thumb	Index	Middle	Ring	Little
99.67018	22.02497	16.66504	15.83008	11.25	12.09961	15.03418
100.7829	22.89264	16.54297	15.69336	11.35254	12.06543	14.95606
101.3102	23.92291	16.45996	15.69336	11.41113	12.0459	14.95606
100.8353	25.62749	16.37695	15.64453	11.40625	12.08496	15.06348
100.1341	26.08893	16.29395	15.60547	11.40625	12.08496	15.13184
98.98463	26.39299	16.25488	15.60059	11.3623	12.08496	15.10254
94.23426	25.96808	16.25488	15.53711	11.25977	12.0166	15.09277
91.70952	25.5764	16.29883	15.59082	11.30371	12.0166	15.05859
87.55821	24.34919	16.29883	15.66895	11.33789	11.98242	15.03418
82.30851	22.26063	16.21582	15.69336	11.25488	11.92871	15.0293
79.13583	20.58654	16.21582	15.64941	11.25977	11.96289	15.0293
76.26502	18.90832	16.13281	15.62988	11.25488	11.94824	14.99512
70.03091	15.02179	16.11328	15.51758	11.20605	11.875	14.79492
63.29993	9.924803	16.09375	15.25391	10.99121	11.9043	14.5752
59.47163	6.715598	16.13281	15.09766	10.9082	11.78711	14.49219
55.39448	3.320444	16.13281	14.92188	10.75195	11.68457	14.34082
47.93234	-4.40977	16.02051	14.54102	10.46387	11.46973	13.99902
41.88198	-10.9941	15.91309	14.13574	10.10742	11.14258	13.5791
39.60555	-13.852	15.83496	13.91113	9.995117	11.02539	13.42285
38.51704	-16.1507	15.71777	13.6377	9.755859	10.87402	13.2666
36.52159	-20.4253	15.43945	13.08105	9.482422	10.60059	12.81738
35.52483	-22.0491	15.27832	12.76855	9.326172	10.4834	12.6709

Figure 6.1 Data sample from data glove.

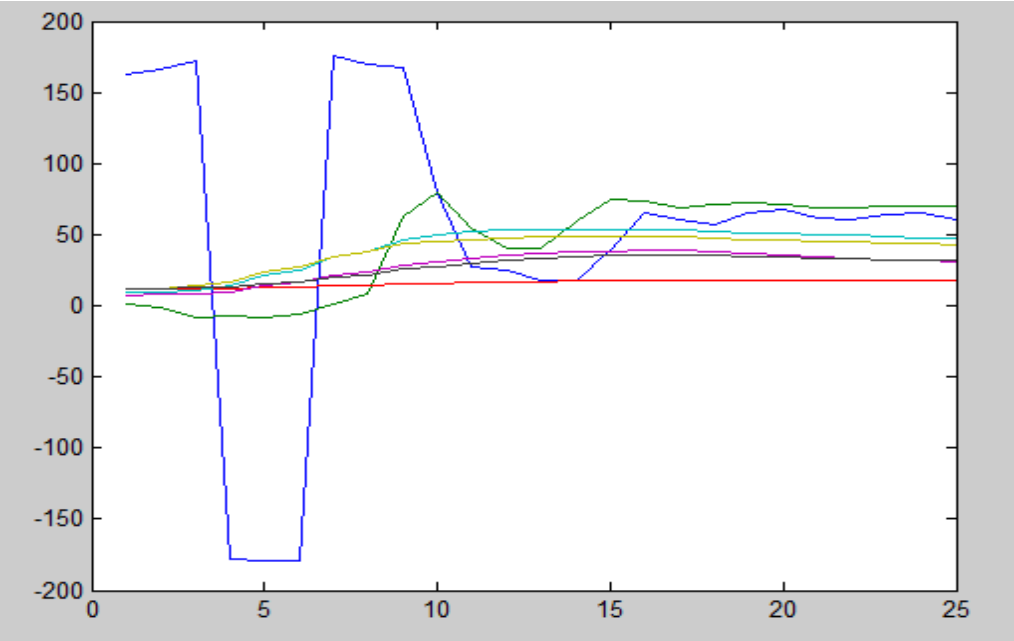


Figure 6.2 Graph representation for the data set sample.

We take data set from three different signers ,each signer gave a five data set about each Model(word) ,so the training set contain 45 data set for our three virtual models(قف/تقدم/وداعا). The full training data set files attached to Appendix C.

The figures bellow demonstrate how each of the sign models represented.

قف:

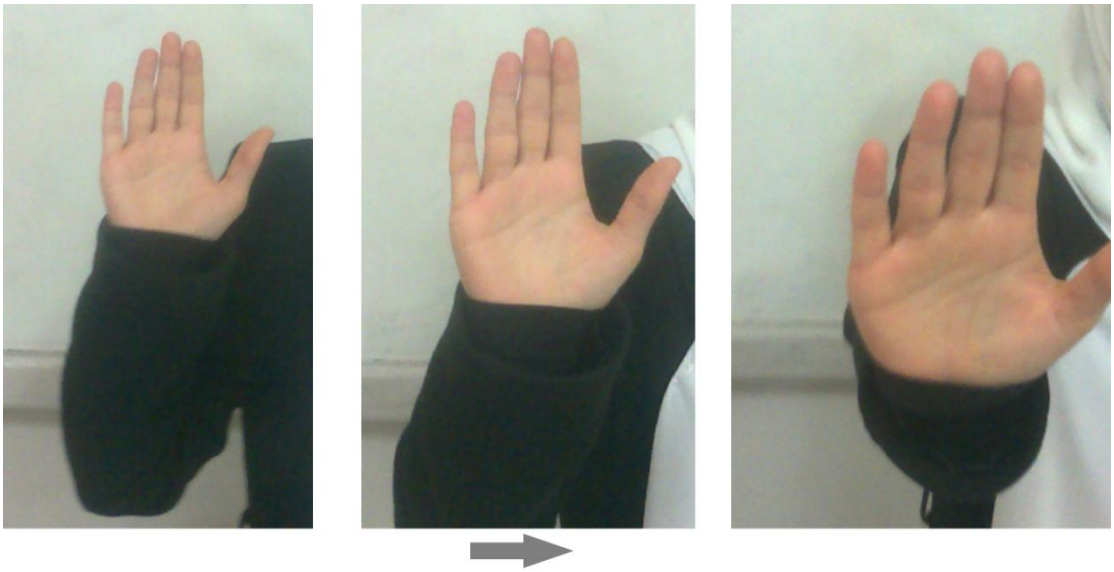


Figure 6.3 The representation of first model (قف)

تقدم:

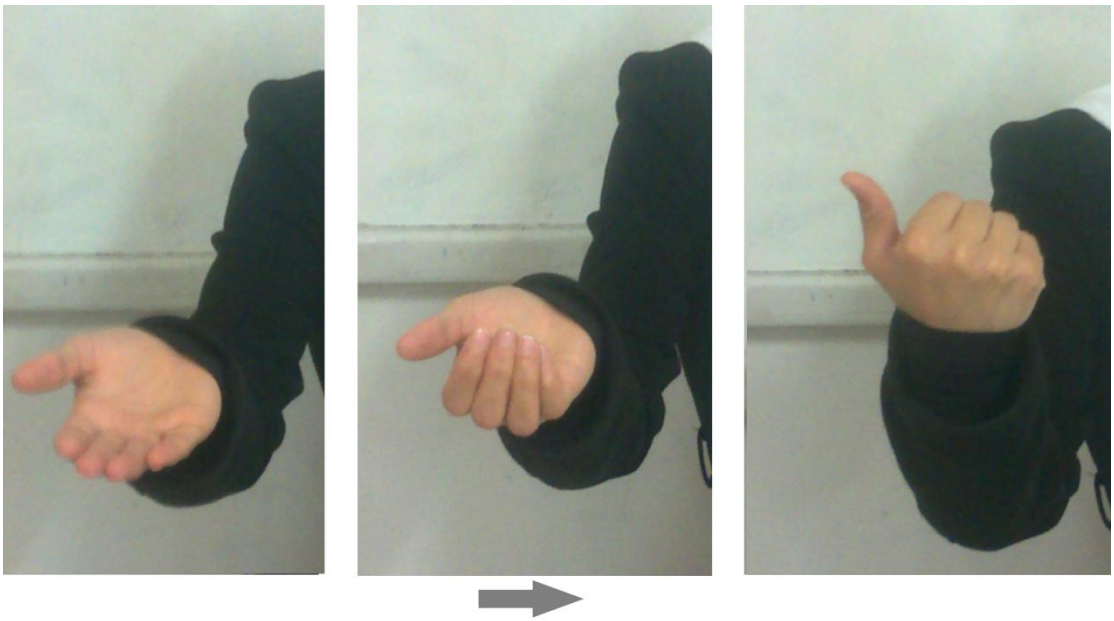


Figure 6.4 The representation of second model (تقدم)

وداعا:

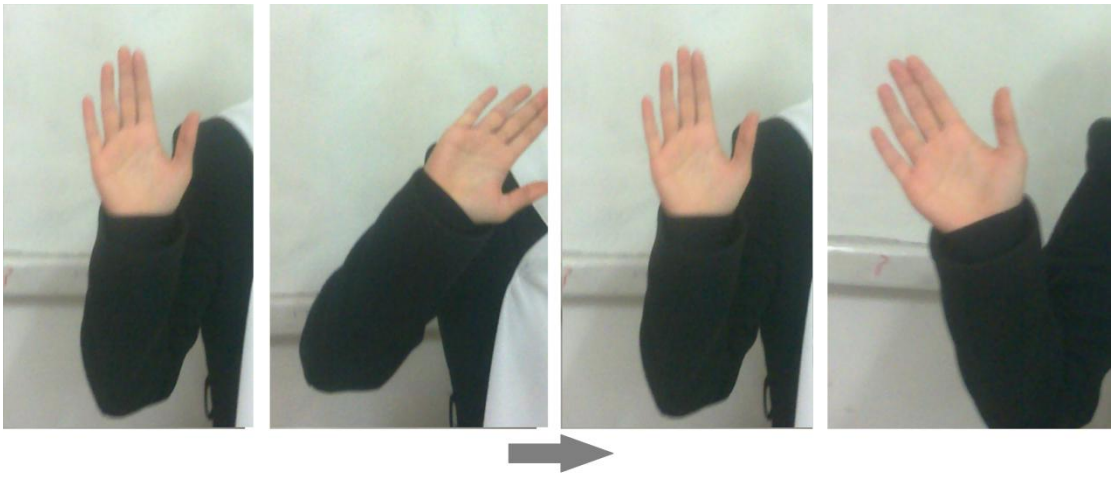


Figure 6.5 The representation of third model (وداعا)

6.3 **Results**:

After performing experiments to optimize number of states and emissions, the following results have been reached.

The following figures represent the success rate at different states and emissions number.

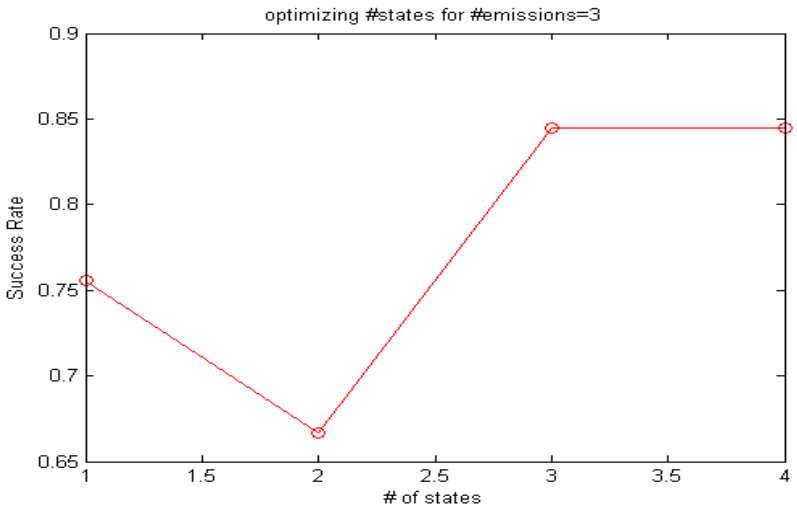


Figure 6.6 Success rate at emission=3

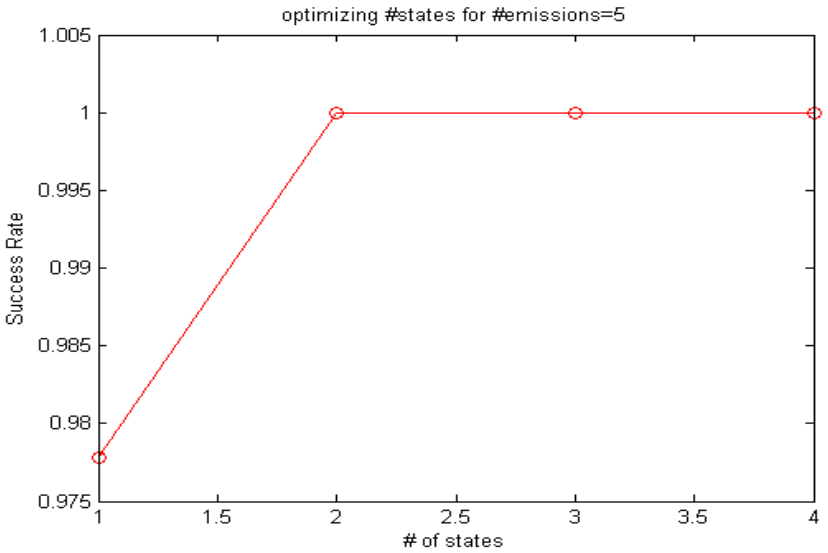


Figure 6.7 Success rate at emission=5

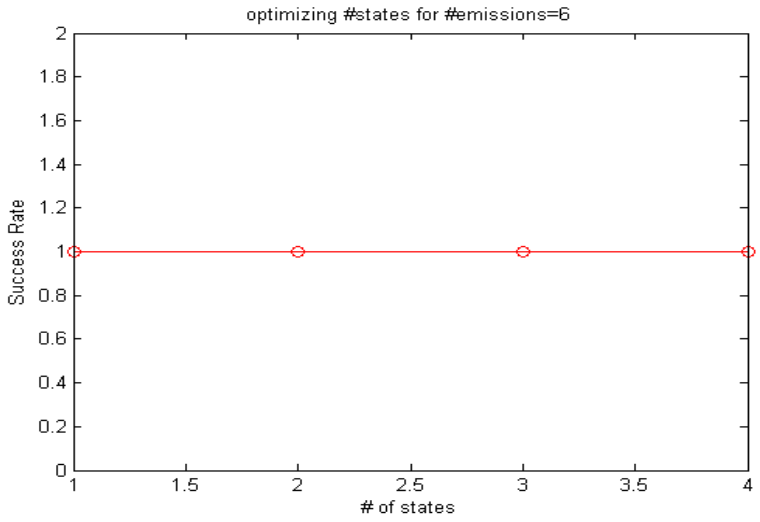


Figure 6.8 Success rate at emission=6

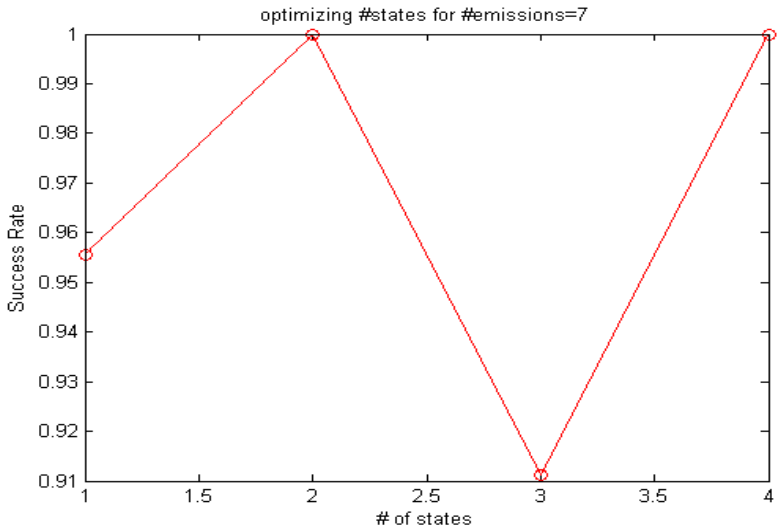


Figure 6.9 Success rate at emission=7

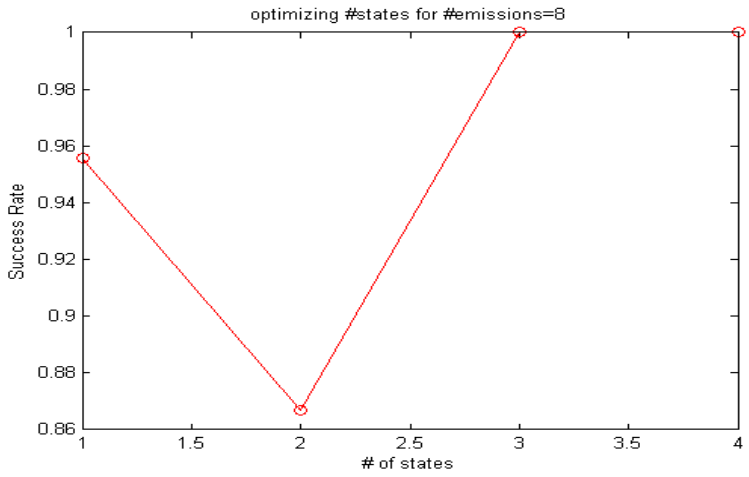


Figure 6.10 Success rate at emission=8

The final results shows that the optimal number of states is one when the number of emissions is four at that point the success rate is almost 100%. As shown in the figure below:

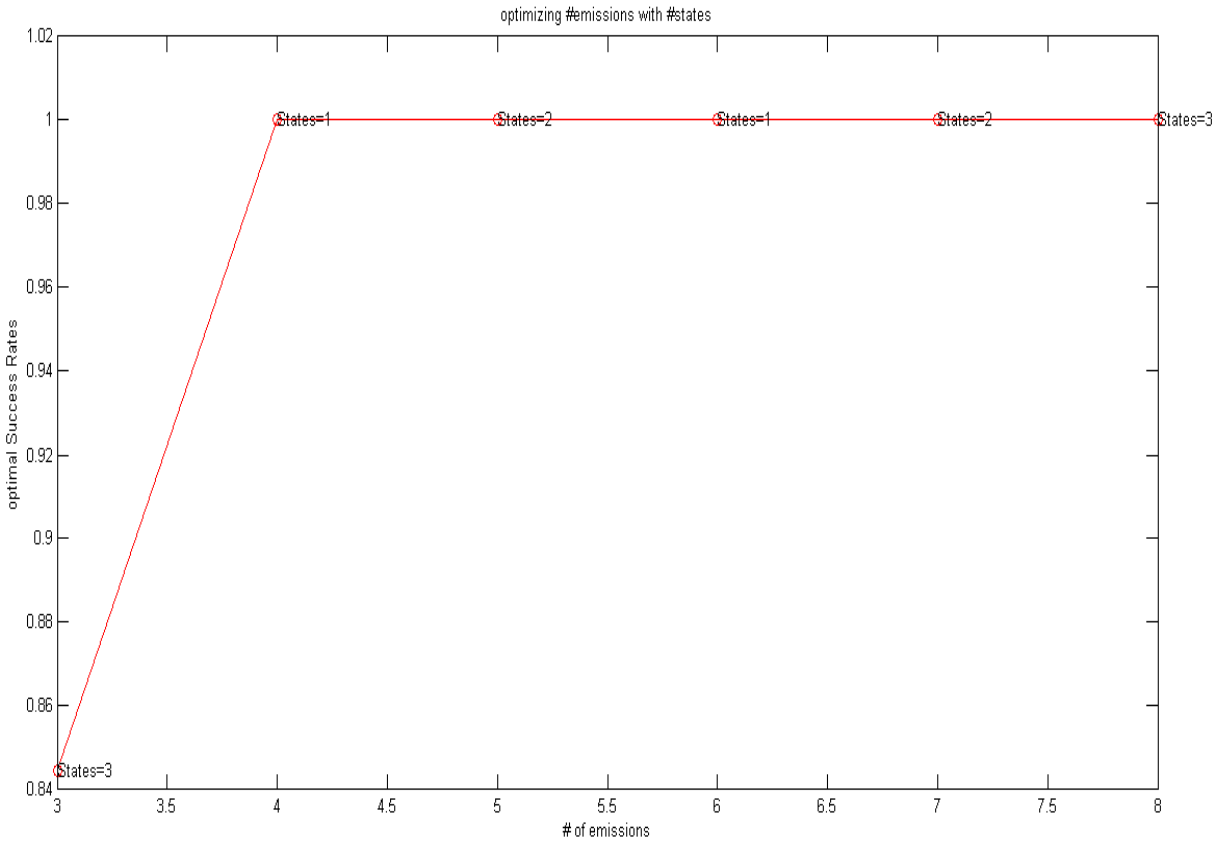


Figure 6.11Optimizing # of emissions with # of states.

The following table represented the transition and emission for the three models:

Table 6.1 The three models Transition and Emission matrices.

Sign	**تقدم**	**قف**	**وداعا**
Transition	[1.0]1x1	[1.0] 1x1	[1.0]1x1
Emission	[0.29386,0.17815, 0.37292,0.15507]1x4	[0.71910,0.00000, 0.14693,0.13397]1x4	[0.71456,0.05465, 0.10347,0.12732]1x4

Chapter Seven

Conclusions and Future Works

7.1 Conclusions.

7.2 Future Works.

7.1 Conclusions

After working on this system many conclusions were concluded and this section contains a description of these conclusions will be pointed:

1. Working on this project increases the knowledge of machine learning algorithms especially HMM.
2. Working on this project increases the knowledge of the sign language.
3. Working on this project increases the knowledge of Arduino programming.
4. This system able o recognize signs from any signer because training data from different signers.
5. Adding new words to the system could be done very easy.

7.2 Future Works

1. This system could be improved so that it becomes able to adjust the outgoing voice, its volume level, and its type (male/female).
2. This system could be improved so that it becomes able to implements full sentences in real time instead of words.
3. This system could be improved so that it success rate increases above the ranges that it reaches.

References

[1] A. Ibarguren, I. Maurtua and B. Sierra, "Layered architecture for real time sign recognition: Hand gesture and movement," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 7, pp. 1216-1228, October 2010.

[2] J. Won, C. Sunyoung, C. Jaeseong and B. Hyeran, "Enhancing Hand Gesture Recognition usingFuzzy Clustering-based Mixture-of-Experts Model," in *5th International Conference on Ubiquitous Information Management and Communication*, New York, NY, USA, 2011.

[3] *Enable Talk*, 5/10/2014. http://www.enabletalk.com/.

[4] R. David, "Appearance-Based Features for Automatic Continuous Sign Language Recognition," Aachen, North Rhine-Westphalia, Germany, 2006.

[5] M. A. Abdel-Fattah, "Arabic Sign Language: A Perspective," *Deaf Studies and Deaf Education*, vol. 10, no. 2, 2005.

[6] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *IEEE*, vol. 77, no. 2 , pp. 257 - 286, Feb 1989.

[7] P. Mark, "Tilt Sensing Using a Three-Axis Accelerometer," Freescale Semiconductor, 2013.

[8] I. Tom, "How to Locate (Almost) Anything," in *Making Things Talk*, O'Reilly Media, Inc., 2007, pp. 291-298.

[9] *SerialGLCD*, 20/11/2014. http://serialglcd.sourceforge.net/

[10] *serialGLCD 1.62*, 20/11/2014. http://sourceforge.net/projects/serialglcd/