

CERTIFICATION OF APPROVAL

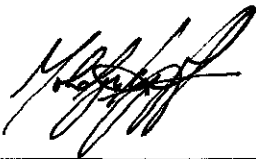
MICROCONTROLLER BASED SIGN LANGUAGE TRANSLATOR

By

MOHAMMED OBAIDALLAH ALHARBI

A project dissertation submitted to the
Electrical & Electronic Engineering Department
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Approved:



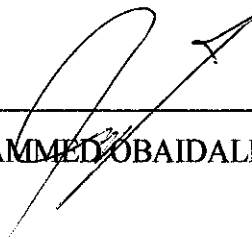
Dr. Mohd Zuki Bin Yusoff
Project supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein has not been undertaken or performed by unspecified sources or persons.



MOHAMMED OBAIDALLAH ALHARBI

ABSTRACT

The communities of vocal impaired and deaf people who use sign language face great communication difficulties with people who use vocal languages. This project, aims to contribute towards bringing the gap closer by offering a tool which translates sign languages to written messages on an LCD display. This report discusses the different development and implementation issues including gesture modeling, sensor interfacing, sign recognition and translation. American Sign Language is widely used in different part of the world including Malaysia; therefore it is considered in this project. The proposed method utilizes five potentiometers to emulate sensor output, a microcontroller to acquire, convert, recognize, translate and display the hand gesture on the LCD unit. The translator can recognize all 26 letters, 10 numbers, and some phrases and words. The presented work is believed to be an entry to more promising and rewarding sign language translation-applications in the future.

TABLE OF CONTENTS

| | |
|--|----|
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Background of Study | 1 |
| 1.2 Problem Statement | 1 |
| 1.3 Project Objectives and Scope | 2 |
| 1.4 Limitations | 2 |
| 1.5 Organization of Report | 3 |
| CHAPTER 2 LITERATURE REVIEW | 4 |
| 2.1 Preliminaries | 4 |
| 2.2 Related Work | 5 |
| 2.3 Recognition and Translation Algorithm: Fuzzy Logic | 6 |
| CHAPTER 3 METHODOLOGY | 7 |
| 3.1 Project Development Flow Chart | 7 |
| 3.2 System Identification and Tools | 8 |
| 3.2.1 Potentiometer | 8 |
| 3.2.2 PIC microcontroller | 10 |
| 3.2.3 CCS C compiler | 11 |
| 3.2.4 PIC development kit and programmer | 12 |
| 3.3 PIC Programming Methodology | 15 |
| 3.4 Sign Language Translation Procedure | 20 |
| 3.4.1 Sensor reading and display | 21 |
| 3.4.2 Signs gesture representation | 22 |
| 3.4.3 Letter matching algorithm | 24 |
| 3.4.4 Dictionary construction | 25 |
| 3.4.5 Translation of letters and numbers from sign language to written language | 29 |
| 3.5 Summary | 31 |
| CHAPTER 4 RESULTS AND DISCUSSION | 32 |
| 4.1 Sensor Reading and Display | 32 |
| 4.2 Basic Translation based on Sensors Reading | 33 |
| 4.3 Translation System Using a Set of Potentiometers | 35 |
| CHAPTER 5 CONCLUSION AND FUTURE DIRECTIONS | 40 |

| | |
|--|----|
| 5.1 Conclusion..... | 40 |
| 5.2 Future Directions..... | 40 |
| REFERENCES..... | 41 |
| Appendix A Project Gantt Chart..... | 43 |
| Appendix B Five Sensors Reading Display (code)..... | 45 |
| Appendix C Five Sensor Interface and Basic Translation System (code) | 46 |
| Appendix D Translation System with Sign Dictionary (main.c) | 47 |
| Appendix E Translation System with Sign Dictionary (main.h) | 53 |
| Appendix F Translation System with Sign Dictionary (lcd.h)..... | 54 |
| Appendix G Investigating the Interface with 5DT Data Glove: A Potential Future Work..... | 56 |
| G.1 5DT Data Glove | 56 |
| G.1.1 Getting started with USB interface | 58 |
| G.1.2 PIC18F4550 as a USB CDC device | 59 |
| G.1.3 PIC and data glove USB interface | 60 |
| G.2 Getting Started with USB..... | 61 |
| G.3 PIC as Serial Port via USB..... | 62 |
| G.4 USB Interface between PIC and Data Glove | 66 |
| G.5 Discussion | 67 |
| Appendix H Data Glove Sensor Map | 70 |
| Appendix I Getting Started with USB (USB CDC Code) | 71 |
| Appendix J 5DT Data Glove and PIC Interface via USB (code)..... | 73 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1: Letter "A" in American Sign Language [2]..... | 4 |
| Figure 2: General Flow Chart of Project Work..... | 7 |
| Figure 3: System Tools (PIC, Board, LCD and Potentiometers)..... | 8 |
| Figure 4: Potentiometer a) Isometric View b) Internal Configuration c) Schematic.... | 9 |
| Figure 5: PIC18F4550 Pin Diagram | 11 |
| Figure 6: SK40C PIC Start-up Kit | 13 |
| Figure 7: UIC00B USB ICSP PIC Programmer | 14 |
| Figure 8: PIC Board and Programmer Connection | 15 |
| Figure 9: Microcontroller Programming Flow Chart..... | 16 |
| Figure 10: Programming Step #1: PICC Project Wizard..... | 17 |
| Figure 11: Programming Step #2: Code edition | 18 |
| Figure 12: Programming Step #3: Building/Compilation..... | 18 |
| Figure 13: Programming Step #4: Simulation (Optional)..... | 19 |
| Figure 14: Programming Step #5: In Circuit Debugging (Optional) | 19 |
| Figure 15: Translation Flow Chart..... | 21 |
| Figure 16: Dictionary Construction Flow Chart, Bias value is 7 | 26 |
| Figure 17: Full Translation Procedure Example | 30 |
| Figure 18: Potentiometer connected to Analog input | 32 |
| Figure 19: 5 Potentiometer, LCD and PIC Interfacing Circuitry..... | 33 |
| Figure 20: Basic Translation based on Sensors Reading (a) Displaying "M. Alharbi" (b) Displaying "Dr. M Zuki" (c) Displaying "None" | 34 |
| Figure 21: Translation System Components: 5 Potentiometers, Main Board (SK40C board), and LCD | 35 |
| Figure 22: Translation System Startup, LCD is displaying the "Starting" message.. | 35 |
| Figure 23: Sensor Readings in First Line (thumb: 123, index:123, middle:238, ring:242, and little:237), Second Line Displaying the Translated Sign | 36 |
| Figure 24: Sensor Readings in Volts (thumb: 2.4V, index:2.4V, middle:4.6V, ring:4.7V and little:4.6V)..... | 37 |
| Figure 25: Translation System Recognizing the Sign for B Equivalent to (thumb: 64+/-7, index: 0, middle: 0, ring: 0 and little: 0) | 37 |
| Figure 26: Recognition and Translation of the Sign for 5 (thumb: 0, index: 64+/-7, middle: 0, ring: 64+/-7, little: 64+/-7) | 38 |

| | |
|--|----|
| Figure 27: Recognition of Phrases e.g. “I Love You” (thumb: 0, index: 0, middle: 224+/-7, ring: 224+/-7, little: 0)..... | 38 |
| Figure 28: “None” Message for any Other Unrecognized Signs | 39 |
| Figure 29: Data Glove with USB Connection..... | 56 |
| Figure 30: Data Gloves Sensor Mapping | 57 |
| Figure 31: Startup Circuit for USB Interface..... | 58 |
| Figure 32: PIC18F4550 Oscillator and Clock Diagram for the CPU and USB Peripheral | 59 |
| Figure 33: Code Snippet for Setting USB Clock | 60 |
| Figure 34: Block Diagram for the Proposed System | 60 |
| Figure 35: Circuit Components..... | 61 |
| Figure 36: PC to Microcontroller Interface via USB Port | 62 |
| Figure 37: GUI Interface to Read a Value from an Analog Device and Toggle the State of LEDs Attached to Microcontroller..... | 62 |
| Figure 38: PIC showing USB is Successfully Attached (Observe the small LED light indicator)..... | 63 |
| Figure 39: PIC showing USB is Successfully Enumerated (Observe the small 2 LED light indicator)..... | 64 |
| Figure 40: Setting up the Serial Communication to the Microcontroller Using Serial Monitor on CCS C Compiler | 65 |
| Figure 41: Display of Received Data from PC via USB Connection | 65 |
| Figure 42: 5DT data glove interface with PIC board..... | 66 |
| Figure 43: Interfacing the PIC and Data Glove (note all LED indicators are OFF) ... | 67 |
| Figure 44: Block Diagram for the System with the Proposed Serial Interface Kit..... | 69 |

LIST OF TABLES

| | |
|--|----|
| Table 1: PIC18F4550 Specification | 11 |
| Table 2: Basic Gesture Meaning | 23 |
| Table 3: Numerical Representation of Gestures | 23 |
| Table 4: Dictionary Table (A-Z, 1-10 and I love you)..... | 27 |
| Table 5: Sensor Mappings for the 5DT Data Glove 14 Ultra | 57 |
| Table 6: Data Packet Sent by the Glove..... | 68 |

CHAPTER 1

INTRODUCTION

This chapter discusses the background of the study, the problem statement, project objectives and scope.

1.1 Background of Study

A sign language is widely used by people who suffer from vocal impairment or hearing problems in which the communicators use visually transmitted signs to convey meanings. The deaf community which utilizes sign language is estimated to be 0.1% of total population, which means millions of people worldwide [1]. This large community faces great difficulties in communicating with normal people. Several attempts have been made to break this gap between sign language users and conventional vocal language communicators by introducing tools that can interpret the meaning for both sides. This project aims to deliver a prototype which interprets the signal made by a sign language communicator into a displayed message on LCD. This project is believed to be a base for future work in this area.

1.2 Problem Statement

Sign language generally utilizes manual movement to convey meanings. This language is not understood by average people. The majority of people understand visually written letters, while sign language users can only use manual signs. In order to break the gap, a set of sensors can be used on the signer to efficiently convert the signs made to electrical signals which in return can be understood by a personal computer (PC) and interpreted accordingly. However, the use of PC does not make the solution mobile and easy to carry. Therefore, a simple IC based circuitry interface (i.e. microcontroller) is required to replace the job of PC. In general, such replacement involves several challenges due to the limited resources which are normally found in conventional ICs (i.e. microcontrollers).

1.3 Project Objectives and Scope

The aim of this project is to construct a prototype which interprets basic signs into a readable text on an LCD. In order to realize such prototype, the ASL language is chosen for the implementation and the following objectives are considered:

1. Obtaining a numerical representation all gestures used in the sign language
2. Constructing a sign language dictionary
3. Prototyping a translation system using set of potentiometers, a microcontroller and LCD modules.

The project is envisaged to deliver a prototype which makes use of a **set of potentiometers to model the actual sign gestures** which can be later replaced to an accurate data glove. A microcontroller with different communication modules is to be used to acquire, manipulate and display the signs being detected by the sensors. A display unit which is as simple as a 2 line-LCD is to be used to display detected messages.

Throughout the implementation of the proposed translation system, only **motionless signs** (i.e. static signs which do not involve any movement of hands, arms, fingers, head or body to convey the meaning) are considered. The motions are essential for several sign vocabulary, however they require complex detection and processing system to detect and interpret them.

1.4 Limitations

The implementation of such project involves several challenges throughout the different development phases. This project ultimately requires a microprocessor system to interface with a sensor unit and to process the reading of the sensor to deliver the final written message on a display unit. The processing of data requires some noise filtering, data acquisition and recognition algorithms in order to robustly deliver the final output. These processes, in addition to sign dictionary data, require significant memory capacity which is not available on most of the commonly used

microcontroller systems.

Additionally, low power consumption for portable devices is a major concern. This is to ensure convenience of usage without compromising the performance of the proposed design. This makes an advantage of using optimized integrated components versus separated ones. For example, a microcontroller with embedded USB peripheral (PIC18F4550) consumes lower power than two separate units comprising a microcontroller and USB interface modules.

1.5 Organization of Report

This report contains five chapters including Chapter 1 which consists of introductory parts for the work and the rest is organized as follows; Chapter 2 lists some preliminaries on sign language and the most recent translation systems developed by researchers. The methodology adopted in this work throughout the development stages is described in Chapter 3. The methodology discusses the components used to develop the system, the software and tools used in the key milestones of the development, and the algorithms used. The results and discussion are reported in Chapter 4. Concluding remarks are reported in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

In this chapter, the previous work which is relevant to this project is presented.

2.1 Preliminaries

Sign language which is based on visual manipulation of hands and body is the language of deaf and vocally impaired people. It is interesting to know that sign language is not universal. Despite the fact that most vocabulary and grammar of sign languages worldwide are quite similar, they are not typically identical [1]. For example the particular word “women” have different sign representation in Auslan, Israeli and DSL sign languages [1]. However, studies indicate that most of world’s sign languages have a great portion of identical vocabulary.



Figure 1: Letter "A" in American Sign Language [2]

On the other side, sign language does not follow the same rules of grammar as for vocal languages [1]. The surrounding vocal language has a significant impact in shaping a particular sign language. This explains the difference in sign languages globally. In this project, we will consider the American Sign Language (ASL) [2] as it shares major similarity with Malaysian Sign Language (or in Malay: Bahasa Isyarat Malaysia : BIM) and is well documented. The letters and the first ten numbers will be tentatively considered in the proposed system. Figure 1 shows example of letter “A” in ASL.

2.2 Related Work

Several attempts have been made to translate sign language to vocal languages and vice versa. J.M. Allen *et al.* in [3] presented a system which translates spoken English to sign language. In this work, the authors discussed an algorithm implemented in personal computer which can automate the translation of spoken and written English language and displays the equivalent via an avatar animated sign interpreter.

P. Mekala *et al.* and R. Akmeliawati *et al.* in [4], [5] discussed an algorithm which utilizes neural network to capture the sign from a camera and process it accordingly to give the English translation. This method requires less expensive hardware but more complicated algorithm to interpret the signs. In order to translate a sign, the image is captured and tracked, then the hand posture is extracted and the corresponding meaning is matched using a learned neural network.

Implementing a recognition system on an ARM processor is discussed in [6]. In this work, the practical aspects of real time blabbering recognition and translation are discussed. The system shows different practical aspects of the implementation of language recognition in embedded systems.

Another interesting work is discussed by R.M. McGuire *et al.* in [7]. In this work, a mobile sign translator based on one hand data glove and a Hidden Markov Model are used. The proposed system shows 94% accuracy for a particular scenario whereby a signer is seeking an apartment.

N. El-Bendary *et al.* attempted to implement arSLAT which recognizes sign representation of Arabic letters and gives the written equivalence [8]. The system processes a video which contains series of image representations for the letters. The

best captured image from the video undergoes several phases including categorization, feature extraction and classification before the Arabic letter is finally recognized. Experimental results show 91% of recognition accuracy.

In summary, this short listing for some of the most relevant work all around the world, show the global potentiality of the problem. It also highlights different areas of focus for the implementation of sign language translators. This includes: sensing devices, processing platform (PC, embedded processors, etc.), recognition algorithms, and output forms. In this project, the focus will be in implementing the translation system in microcontroller processing environment.

2.3 Recognition and Translation Algorithm: Fuzzy Logic

Fuzzy logic is a form of many-valued logics; it conceptually deals with reasoning that is approximate rather than fixed and exact. In contrast with the traditional logic theory, where binary variables have two logic values: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth. In partial truth, the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions.

The fuzzy logic is similar to some extent with the human reasoning. It allows for approximate values and inferences as well as incomplete or ambiguous data (fuzzy data) as opposed to only relying on crisp data (binary yes/no choices). Fuzzy logic is able to process incomplete data and provide approximate solutions to problems other methods find difficult to solve. The terminology used in fuzzy logic but not used in other methods is: very high, increasing, somewhat decreased, reasonable and very low.

It is relevant to note that fuzzy logic and probabilistic logic are similar in a mathematical point of view, but conceptually distinct. Fuzzy logic corresponds to "degrees of truth", while probabilistic logic corresponds to "probability, likelihood"; as these differ, fuzzy logic and probabilistic logic yield different models of the same real-world situations [9].

CHAPTER 3

METHODOLOGY

In this part, the methodology used to realize the project is discussed.

3.1 Project Development Flow Chart

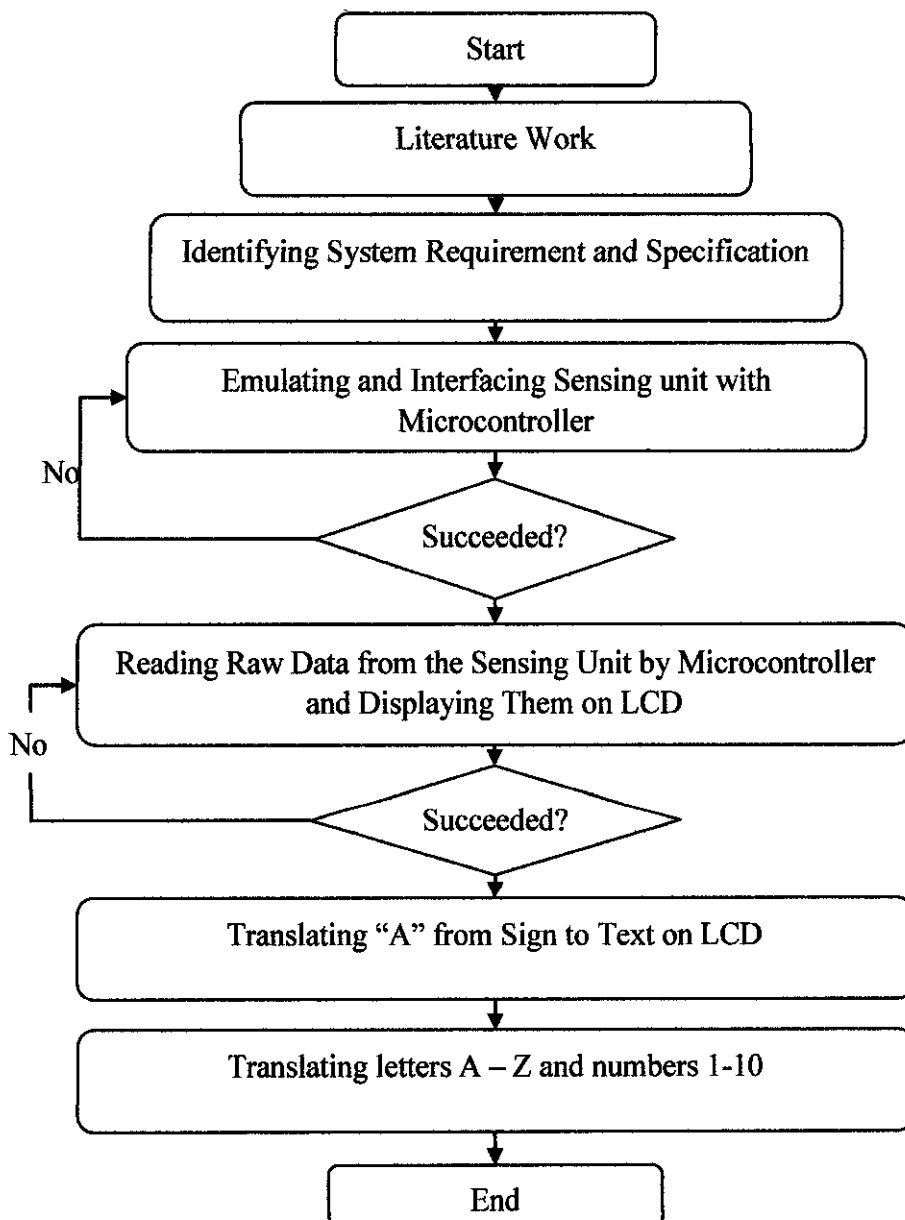


Figure 2: General Flow Chart of Project Work

The overall flow of the project can be divided into further detailed steps which are shown in Gantt chart in Appendix A)

3.2 System Identification and Tools

Throughout the development stage of the prototype of the project, several tools are potentially considered as shown in Figure 3.

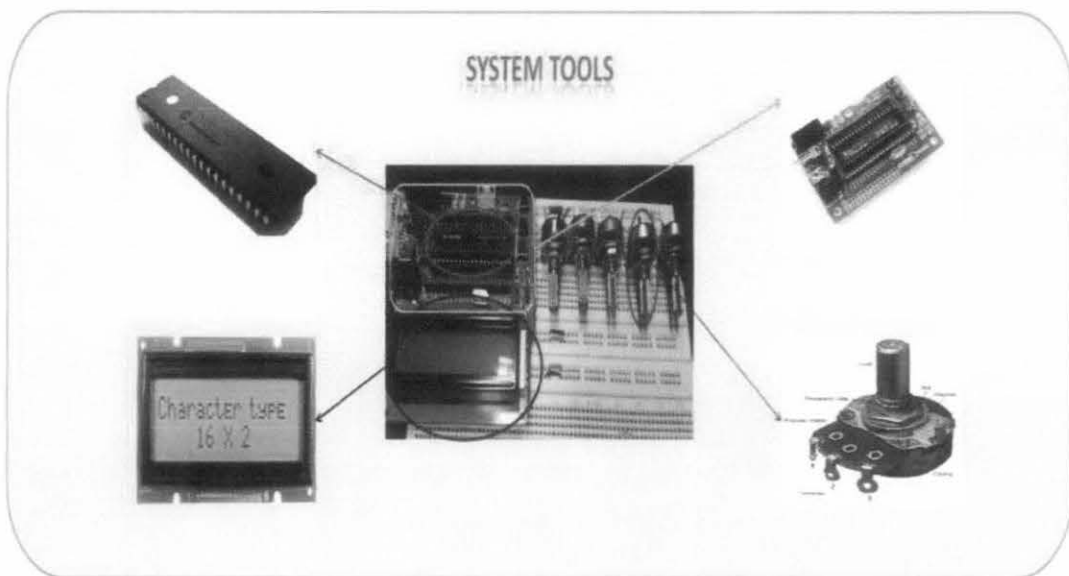
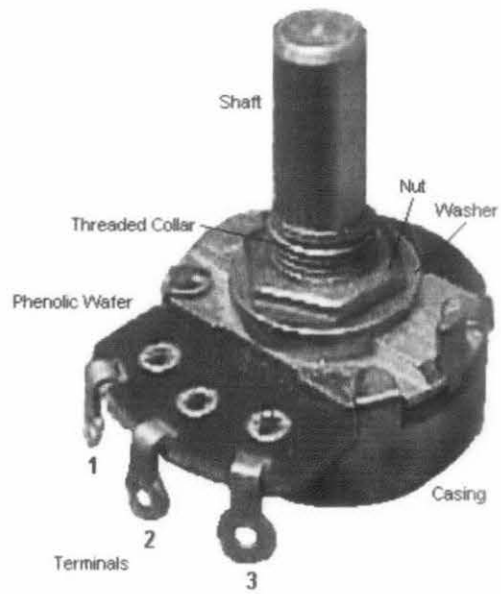


Figure 3: System Tools (PIC, Board, LCD and Potentiometers)

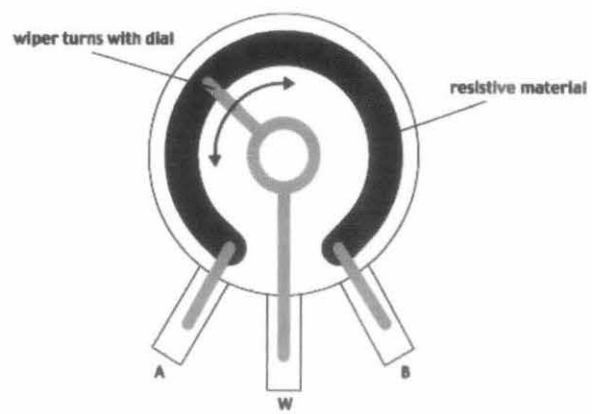
The tools used for the implementation of the translation system and the respective functional and technical details are discussed in the following sections.

3.2.1 *Potentiometer*

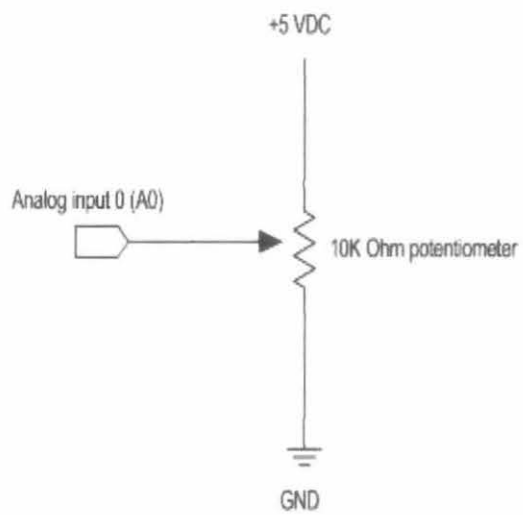
A potentiometer is a simple three terminals variable resistor. It comes in different values for the resistance across its ends. A third terminal in the middle is connected via a moving knob to adjust the resistance at this terminal from 0 to full value in relation to either ends. Figure 4 shows the isometric view, internal configuration and schematic of the potentiometer.



(a)



(b)



(c)

Figure 4: Potentiometer a) Isometric View b) Internal onfiguration c) schematic

The potentiometer is used to emulate and produce 0-5 V analog output. It has generally similar output range of a possible gesture sensor; this allows it to be used as a simplified model for gesture sensor. Therefore it can be used to emulate a fingers gesture sensing unit.

3.2.2 PIC microcontroller

A mid range microcontroller from Microchip is to be used. This selection enables the developers to deal with the prototype with more flexibility and efficiency.

The PIC18F4550 [11] is among the most commonly used Microchip microcontrollers barely because of its USB communication support capabilities. The PIC18F4550 is a 40-pin high performance microcontroller which is equipped with several built-in peripherals (Figure 5). The proposed system may require the USB support for advanced used, therefore, the selection is made to enable future development and flexibility of functionality expansion.

Along with the USB support, the microcontroller is featured with different processing modes, configurable internal oscillators, extendable instruction set which makes it a high performance yet power efficient microcontroller. The most important specifications to consider are reported in Table 1. The 32KB flash memory allows long programs (more than 16 thousands assembly code lines) to be executed. The data used along the execution of the program (i.e. variables' data) are saved in SRAM memory which is 2KB in size for the PIC18F4550 microcontroller.

Some of the peripherals of the microcontroller are not considered as the proposed system does not require them, however, it is likely that normal I/O operation are to be used instead to allow access to other direct digital transmission based devices such as LCD.

Table 1: PIC18F4550 Specification

| Program Memory | | Data Memory | | I/O | A/D | CCP/ ECCP PWM | SPP | MSSP | | EAUSART | Comparators | Timers |
|----------------|--------------|-------------|--------|-----|-----|---------------------|-----|------|-----|---------|-------------|--------|
| Bytes | Instructions | SRAM | EEPROM | | | | | SPI | I2C | | | |
| 32K | 16384 | 2048 | 256 | 35 | 13 | 1/1 | Yes | Yes | Yes | 1 | 2 | 1/3 |

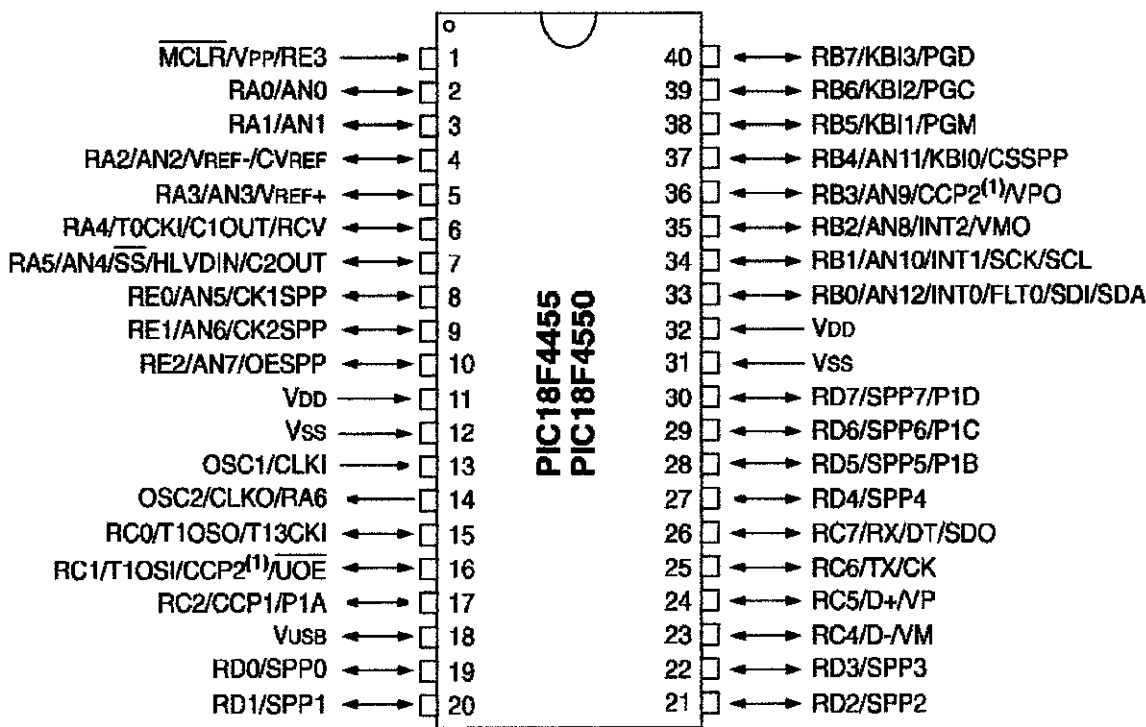


Figure 5: PIC18F4550 Pin Diagram

3.2.3 CCS C compiler

In order to program the microcontroller, a compiler is to be utilized. In this project, the PIC C compiler from CCS is to be used.

The CCS C compiler is easy to use, and almost immediate to get started due to the project wizard feature and the different startup codes which it offers.

Among the features of CCS C compiler:

- Automatic fuses configuration
- Extensive built-in functions providing direct access to PIC hardware
- Extensive source code driver library
- Arithmetic library
- Integrated development environment

3.2.4 PIC development kit and programmer

To speed up the development phase, a startup kit [12] is used as the platform of the microcontroller circuit. The use of this tool provides easier and more robust circuit to be built. The board in Figure 6 provides several functionalities and circuitry support. The kit is a robust development platform which offers:

- Voltage regulation circuitry (9 V input voltage to 5 V output voltage)
- Reset button
- USB port
- Connector to programmer
- Optional connection to LCD and UART
- 2 switches and 2 LEDs connected to Port B

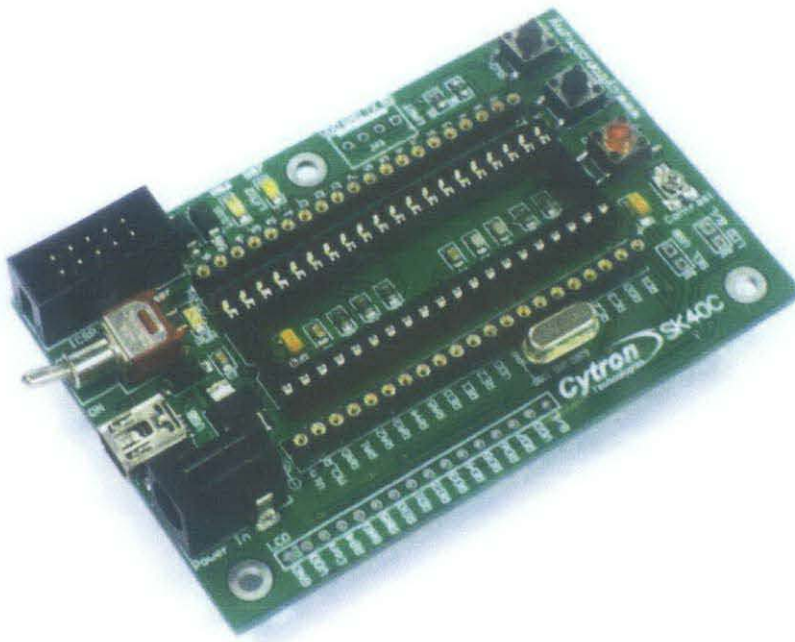


Figure 6: SK40C PIC Start-up Kit

In order to transfer the C codes to the program memory of the microcontroller, USB ICSB programmer (UIC00B) [13] is considered (see Figure 7). This programmer is a cheap programming solution and is highly compatible with the SK40C startup kit.

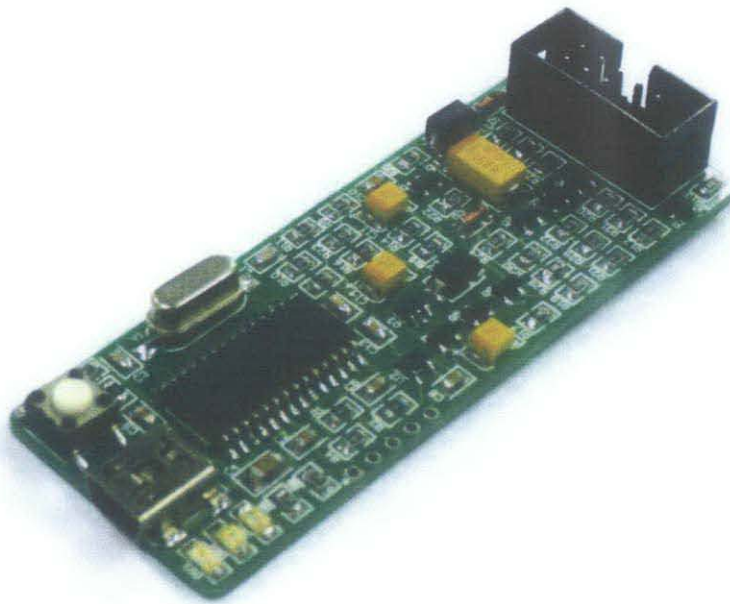


Figure 7: UIC00B USB ICSP PIC Programmer

The programming software (PIC kit 2) takes the hex file which is produced by CCS compiler and loads it to the microcontroller memory via the UIC00B programmer. The interface between the PIC board and the programmer is shown in Figure 8.

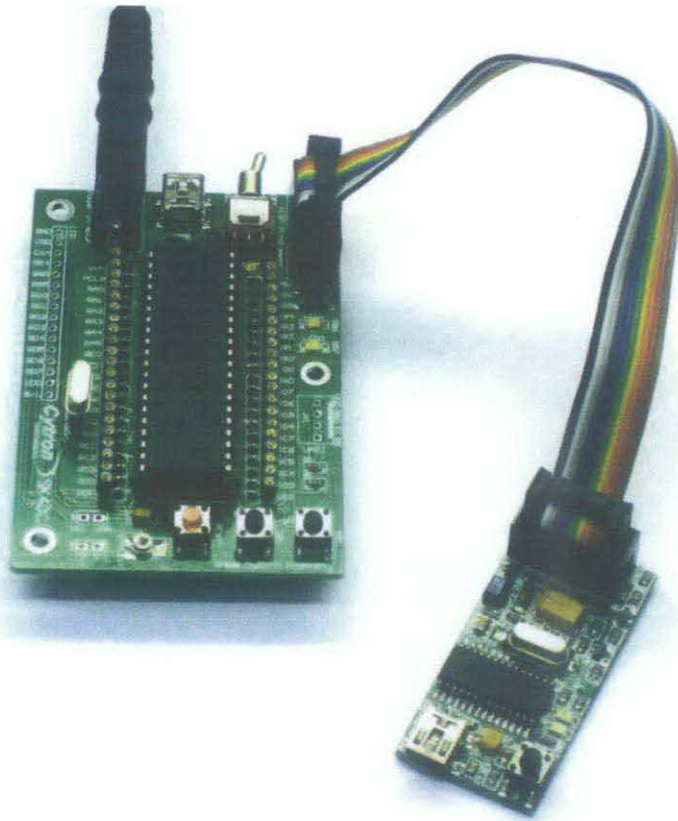


Figure 8: PIC Board and Programmer Connection

3.3 PIC Programming Methodology

The several steps required to implement and realize the developed C codes on the target board are discussed in this part. The steps are illustrated in the flow chart shown in Figure 9. The steps involve the following:

- **Project Creation using project wizard:** (see Figure 10) In CCS C compiler the best way to develop code is to start by project wizard. The project wizard provides a good utility which automatically generates startup codes with the proper fuses (configuration) setting, #include files, peripheral setups and main function. This is very useful for beginners and produces very robust codes.

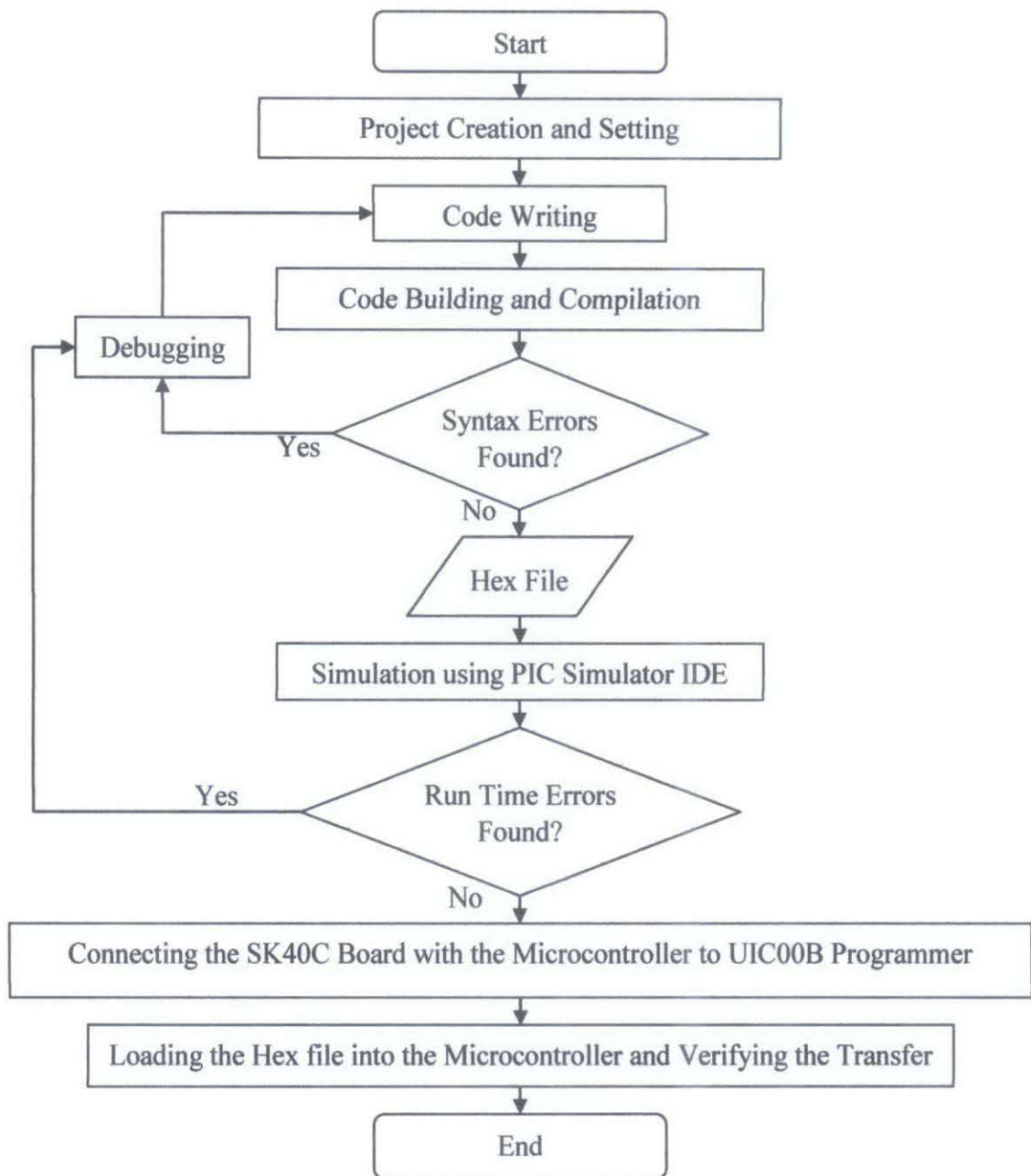


Figure 9: Microcontroller Programming Flow Chart

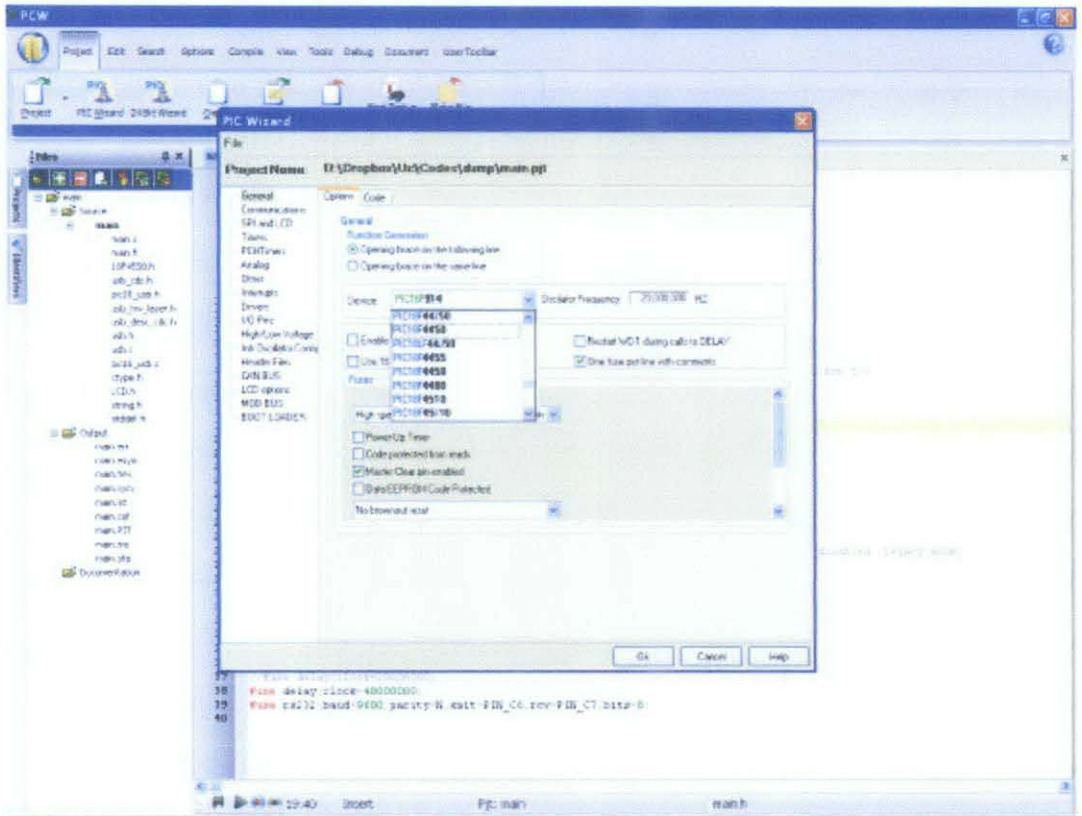


Figure 10: Programming Step #1: PICC Project Wizard

- **C Code edition:** This is where the developer writes the actual code by editing the main function and adding more functions according to the needs of the application (see Figure 11). The CCS C syntax follows the syntax of the standard ANSI C to a good extent.

- Simulation (optional):** Before loading the produced machine code (by the C compiler), it is worthy to see how the PIC would work when the code is to be loaded to the microcontroller. PIC simulator IDE offers realistic simulation interface whereby all the inputs and outputs can be monitored. Input pins can be easily stimulated and several configurable output devices are available.

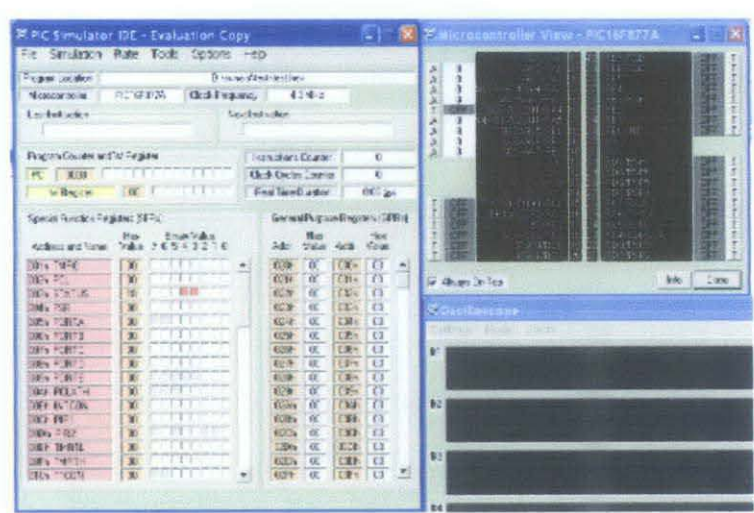


Figure 13: Programming Step #4: Simulation (Optional)

- Debugging (optional):** This is where the errors are rechecked and corrected by the programmer.

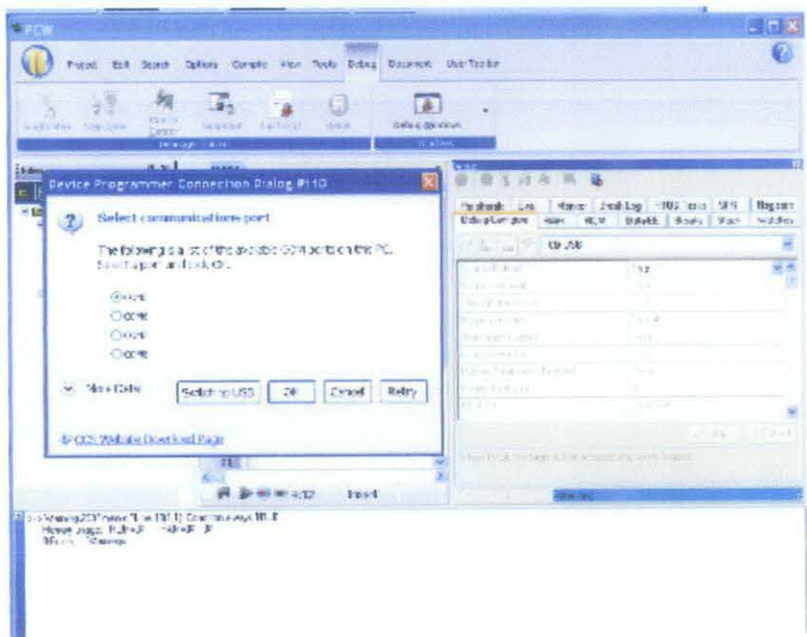


Figure 14: Programming Step #5: In Circuit Debugging (Optional)

- **Preparing the system for program loading:** The microcontroller has to be connected to the programmer (UIC00B) via certain pins as shown in the respective datasheet. In our case, fortunately the pins are accessible to the programmer via dedicated connection port and cable as shown in Figure 8. The programmer has to be known for the PC hosting the PICkit software by proper installation as described in the respective installation manual. The UIC00B programmer is USB device.
- **Loading the Hex file to the microcontroller:** The PICkit is used to load the hex file to the microcontroller. After this step the system is ready to run and the programmer can be disconnected from the board as the microcontroller does not require the connection anymore and in fact it may cause some malfunction to the circuit.

3.4 Sign Language Translation Procedure

The translation procedure involves several issues, tasks and algorithms. The following part discusses these challenges and explains the methodologies adopted in this work. The flow chart of the translation system is shown in Figure 15.

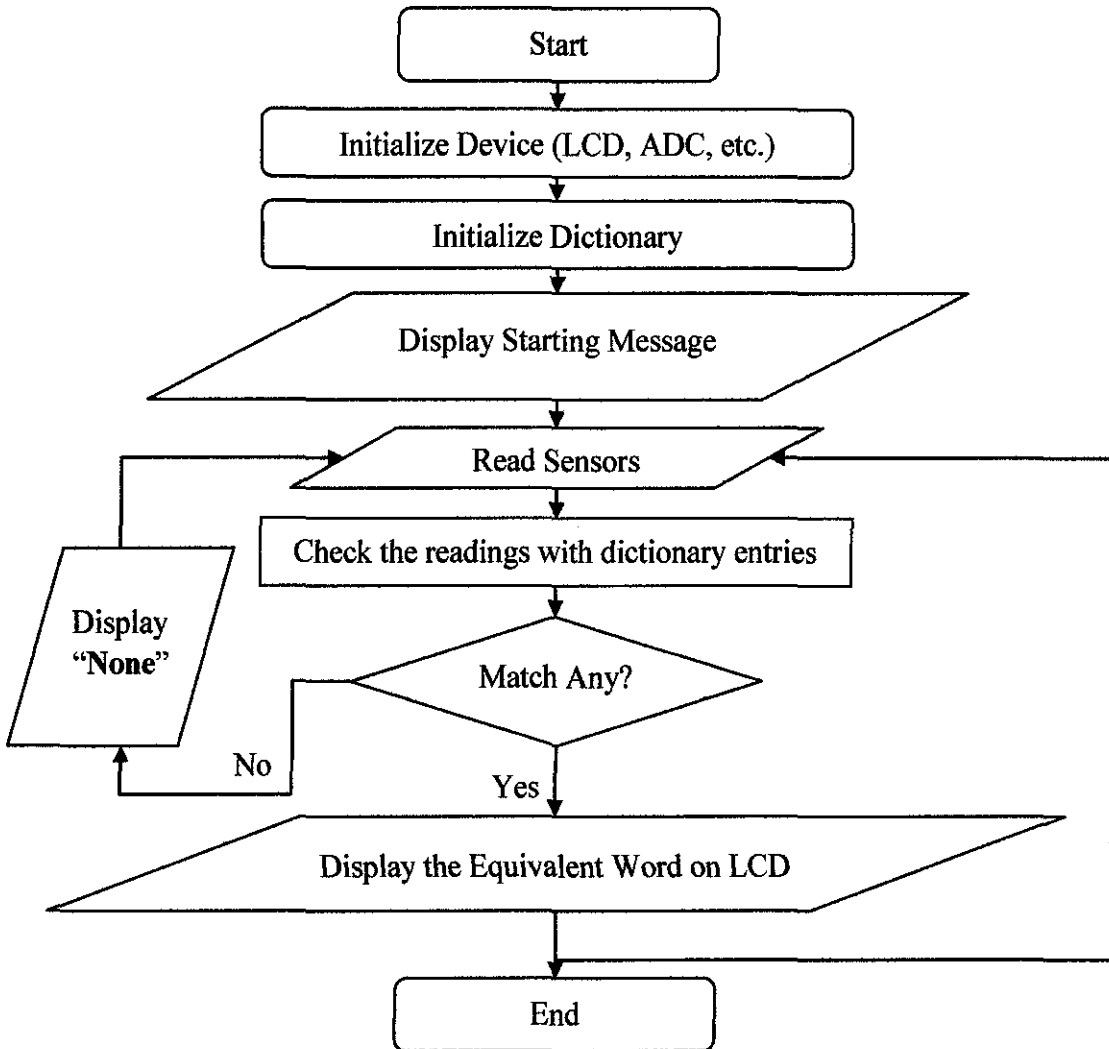


Figure 15: Translation Flow Chart

The translation is basically a closed loop in which the sensors are read, compared and if recognized, the results are displayed on the LCD unit. The reading of the sensors is compared to a dictionary which is created according to the ASL dictionary. The dictionary consists of a set of numerical representation to the gestures of the five fingers (thumb, index, middle, ring and little). The modeling and the numerical representation of the gesture are described in the following part.

3.4.1 Sensor reading and display

The first step on the hardware development of the system is to realize the interface between the PIC and the sensing unit. Initially the sensing unit is assumed to be simple potentiometers and therefore, the PIC is required to establish a communication with the potentiometers. The ADC peripheral in the PIC is used to implement this

function. Five pins are devoted for this purposes which are: A0, A1, A2, A3, and A5. The first step in implementation is to set the proper configuration using the project wizard in CCS C compiler as follows:

- Selecting PIC18F4550 (as it is the target PIC)
- Oscillator frequency: 20,000,000 Hz (as used in the development board)
- For the oscillator fuses, choose the configuration: High speed Osc (> 4mhz, for PCM/PCH) (>10mhz for PCD)
- Unselect option : “PORTB pins are configured as analog input channels”
- For the analog configuration, the following is used:
 - A0, A1, A2, A3, A5
 - Range 0-Vdd
 - Units: 0-255
 - Internal 2-6 μ s for the clock

A startup code is generated upon making the above settings. An LCD is used to display the data. The full source code is listed in Appendix B.

3.4.2 Signs gesture representation

It is known that each letter or word in sign language is composed of gestures made by the five fingers of the hand. In order to obtain a numerical representation for each gesture, 16 gestures per finger are identified to be the basic building block for each sign as shown in Table 2.

Table 2: Basic Gesture Meaning

| Gesture Code | Gesture Description | Gesture Code | Gesture Description |
|--------------|--|--------------|---------------------------------------|
| G0000 | Inflexed finger | G0032 | Lower joint half bent |
| G1000 | Upper Joint flexed | G1032 | Upper+ Lower joint half bent |
| G0200 | Middle joint flexed | G0232 | Middle+ Lower joint half bent |
| G1200 | Middle + Upper joint flexed | G1232 | Upper+ Middle+ Lower joint half bent |
| G0031 | Lower joint tilts aside | G0033 | Lower joint fully bent |
| G1031 | Upper+ Lower joint tilts aside | G1033 | Upper+ Lower joint fully bent |
| G0231 | Middle+ Lower joint tilts aside | G0233 | Middle+ Lower joint fully bent |
| G1231 | Upper +Middle +Lower joint tilts aside | G1233 | Upper +Middle +Lower joint fully bent |

For each sign entry in the dictionary, the gesture represented by each finger is identified according to the table above and numerical values are assigned according to Table 3.

Each gesture is assigned to an arbitrary number from (0-255) with a 16 digits step. The gaps between the gestures are later exploited to identify fuzzy limits between the gestures.

Table 3: Numerical Representation of Gestures

| Gesture Code | Numerical Equivalent | Gesture Code | Numerical Equivalent |
|--------------|----------------------|--------------|----------------------|
| G0000 | 0 | G0032 | 128 |
| G1000 | 16 | G1032 | 144 |
| G0200 | 32 | G0232 | 160 |
| G1200 | 48 | G1232 | 176 |
| G0031 | 64 | G0033 | 192 |
| G1031 | 80 | G1033 | 208 |
| G0231 | 96 | G0233 | 224 |
| G1231 | 112 | G1233 | 240 |

3.4.3 Letter matching algorithm

The signals resembling letters and numbers do not have a strict set of Boolean values. This would suggest the use of Fuzzy logic based algorithm. The signs by nature are not exact and identical to all users. When detecting such signs, the detected signal for the same sign but from different users will vary but should still be close. This consequently leads us to select a fuzzy algorithm to store and match the sign language dictionary.

In fuzzy algorithm, the values of its variable are not in simple TRUE (1) and FALSE (0) patterns, however discrete values representing wide range of trueness and falseness ranging from extremely true to extremely false are typically considered.

In the case of sign language translator, the variables are the reading of sensors and the exact number of sensors depends on the type of sensor system. Each sensor is described by 8 bits value ranging from totally flexed to totally inflexed; however for more general case, the size of the variable (number of bits) depends on the accuracy of the sensor.

The reading of the sensor is to be later compared to find the similar letter which the gesture resembles. The letters, on the other hand, are to be represented by a set of values for each sensor. The fuzzy part comes here, whereby; the values representing each letter describe the upper limit and lower limit for each sensor value. This means, a typical letter or word, is represented by several variables describing the upper limit for the sensors and another set of variables describing the lower limit for the sensors. In addition to that another variable is required to store the equivalent word itself.

The data structure comprising of the lower limit for the sign from each sensor, the upper limit for the sign from each sensor and the equivalent word represents a single entry in the proposed sign dictionary. A look-up table is then to be made consisting of all data structures holding the entries for each sign and the corresponding meaning. The entries in the look up table are to be derived empirically.

A simplified version of the translation system is implemented using three words based on the reading of five sensors. The code is listed in Appendix C and the results are shown and discussed in next chapter.

3.4.4 Dictionary construction

As explained previously, in order to enable the translation, a dictionary holding the gesture and the equivalent word has to be constructed. The construction of the dictionary is shown in Figure 16.

The flow shows the steps adopted in realizing the dictionary. In C programming a **struct** data type is used to represent each dictionary entry. The dictionary is simply an array of “**struct**” data types. The maximum number of entries depends on the size of the data memory (RAM). In the future, a separate memory chip is required to store larger number of entries.

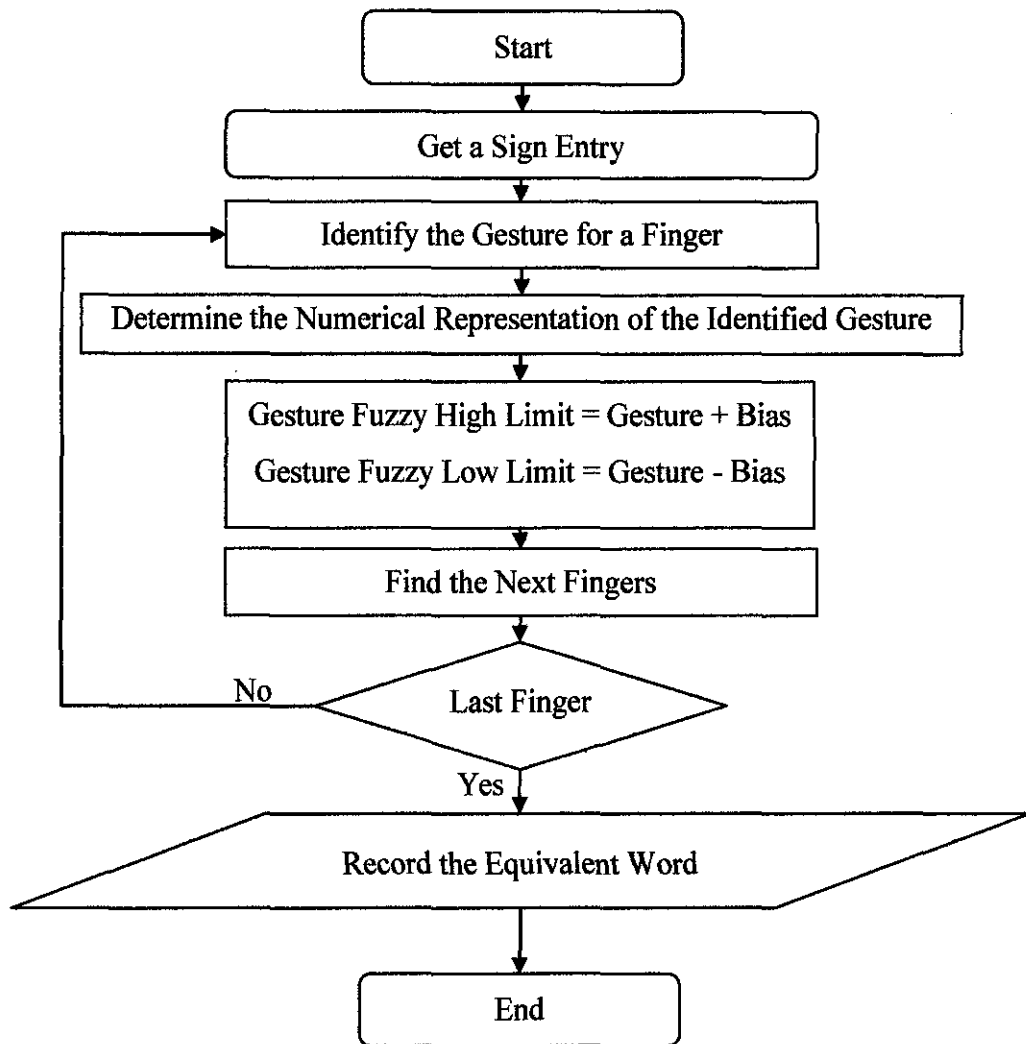


Figure 16: Dictionary Construction Flow Chart, Bias value is 7

The methodology shown in Figure 16 is used to obtain and construct the sign dictionary. The sign versions of the letters from A to Z and numbers from 1 to 10 and the “I love you” phrase are considered from [2] and consequently the sign dictionary is constructed in Table 4.

The entries of the table actually show the range of the five sensor values and the equivalent word. As example the sign which gives a value in the range of (G1031L to G1031) for the thumb sensor, (G0233L to G0233) for the index sensor, (G1031L to G1031) for the middle sensor, (G1233L to G1233) for the ring sensor and (G1233L to G1233) for the little sensor represents the letter A.

Table 4: Dictionary Table (A-Z, 1-10 and I love you)

| Fuzzy Range | Gesture Code | | | | | Equivalent Word |
|-------------------|--------------|--------|--------|--------|--------|-----------------|
| | Thumb | Index | Middle | Ring | Little | |
| Lower Fuzzy Limit | G1031L | G0233L | G0233L | G1233L | G1233L | A |
| Upper Fuzzy Limit | G1031H | G0233H | G0233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G0031L | G0000L | G0000L | G0000L | G0000L | B |
| Upper Fuzzy Limit | G0031H | G0000H | G0000H | G0000H | G0000H | |
| Lower Fuzzy Limit | G0000L | G0200L | G0200L | G0200L | G0200L | C |
| Upper Fuzzy Limit | G0000H | G0200H | G0200H | G0200H | G0200H | |
| Lower Fuzzy Limit | G1200L | G0000L | G1232L | G1232L | G1232L | D |
| Upper Fuzzy Limit | G1200H | G0000H | G1232H | G1232H | G1232H | |
| Lower Fuzzy Limit | G1231L | G1200L | G1200L | G1200L | G1200L | E |
| Upper Fuzzy Limit | G1231H | G1200H | G1200H | G1200H | G1200H | |
| Lower Fuzzy Limit | G1032L | G0232L | G0000L | G0031L | G0031L | F |
| Upper Fuzzy Limit | G1032H | G0232H | G0000H | G0031H | G0031H | |
| Lower Fuzzy Limit | G0032L | G0032L | G1233L | G1233L | G1233L | G |
| Upper Fuzzy Limit | G0032H | G0032H | G1233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G1232L | G0032L | G0032L | G0233L | G0233L | H |
| Upper Fuzzy Limit | G1232H | G0032H | G0032H | G0233H | G0233H | |
| Lower Fuzzy Limit | G1200L | G0233L | G0233L | G0233L | G0000L | I |
| Upper Fuzzy Limit | G1200H | G0233H | G0233H | G0233H | G0000H | |
| Lower Fuzzy Limit | G1200L | G0233L | G0233L | G0233L | G0000L | J |
| Upper Fuzzy Limit | G1200H | G0233H | G0233H | G0233H | G0000H | |
| Lower Fuzzy Limit | G0232L | G0000L | G0033L | G0233L | G0233L | K |
| Upper Fuzzy Limit | G0232H | G0000H | G0033H | G0233H | G0233H | |
| Lower Fuzzy Limit | G0000L | G0000L | G0233L | G0233L | G0233L | L |
| Upper Fuzzy Limit | G0000H | G0000H | G0233H | G0233H | G0233H | |
| Lower Fuzzy Limit | G0232L | G0033L | G0033L | G0033L | G1233L | M |
| Upper Fuzzy Limit | G0232H | G0033H | G0033H | G0033H | G1233H | |
| Lower Fuzzy Limit | G1232L | G0232L | G0232L | G0233L | G0233L | N |
| Upper Fuzzy Limit | G1232H | G0232H | G0232H | G0233H | G0233H | |
| Lower Fuzzy Limit | G1200L | G1232L | G1232L | G1232L | G1232L | O |

| | | | | | | |
|-------------------|--------|--------|--------|--------|--------|---|
| Upper Fuzzy Limit | G1200H | G1232H | G1232H | G1232H | G1232H | |
| Lower Fuzzy Limit | G0032L | G0000L | G0033L | G0233L | G0233L | P |
| Upper Fuzzy Limit | G0032H | G0000H | G0033H | G0233H | G0233H | |
| Lower Fuzzy Limit | G0032L | G0033L | G1233L | G1233L | G1233L | Q |
| Upper Fuzzy Limit | G0032H | G0033H | G1233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G0232L | G0000L | G0031L | G0232L | G0233L | R |
| Upper Fuzzy Limit | G0232H | G0000H | G0031H | G0232H | G0233H | |
| Lower Fuzzy Limit | G0232L | G1233L | G1233L | G1233L | G1233L | S |
| Upper Fuzzy Limit | G0232H | G1233H | G1233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G0232L | G0232L | G0233L | G0233L | G0233L | T |
| Upper Fuzzy Limit | G0232H | G0232H | G0233H | G0233H | G0233H | |
| Lower Fuzzy Limit | G0200L | G0000L | G0000L | G0232L | G0232L | U |
| Upper Fuzzy Limit | G0200H | G0000H | G0000H | G0232H | G0232H | |
| Lower Fuzzy Limit | G0200L | G0031L | G0000L | G0233L | G0233L | V |
| Upper Fuzzy Limit | G0200H | G0031H | G0000H | G0233H | G0233H | |
| Lower Fuzzy Limit | G0233L | G0031L | G0000L | G0031L | G0232L | W |
| Upper Fuzzy Limit | G0233H | G0031H | G0000H | G0031H | G0232H | |
| Lower Fuzzy Limit | G1232L | G1200L | G1233L | G1233L | G1233L | X |
| Upper Fuzzy Limit | G1232H | G1200H | G1233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G0000L | G1233L | G1233L | G1233L | G0000L | Y |
| Upper Fuzzy Limit | G0000H | G1233H | G1233H | G1233H | G0000H | |
| Lower Fuzzy Limit | G1232L | G0000L | G1233L | G1233L | G1233L | Z |
| Upper Fuzzy Limit | G1232H | G0000H | G1233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G1233L | G0000L | G1233L | G1233L | G1233L | 1 |
| Upper Fuzzy Limit | G1233H | G0000H | G1233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G1232L | G0031L | G0000L | G1233L | G1233L | 2 |
| Upper Fuzzy Limit | G1232H | G0031H | G0000H | G1233H | G1233H | |
| Lower Fuzzy Limit | G0000L | G0031L | G0000L | G1233L | G1233L | 3 |
| Upper Fuzzy Limit | G0000H | G0031H | G0000H | G1233H | G1233H | |
| Lower Fuzzy Limit | G0233L | G0031L | G0000L | G0031L | G0031L | 4 |
| Upper Fuzzy Limit | G0233H | G0031H | G0000H | G0031H | G0031H | |
| Lower Fuzzy Limit | G0000L | G0031L | G0000L | G0031L | G0031L | 5 |
| Upper Fuzzy Limit | G0000H | G0031H | G0000H | G0031H | G0031H | |

| | | | | | | |
|-------------------|--------|--------|--------|--------|--------|------------|
| Lower Fuzzy Limit | G0232L | G0031L | G0000L | G0031L | G0232L | 6 |
| Upper Fuzzy Limit | G0232H | G0031H | G0000H | G0031H | G0232H | |
| Lower Fuzzy Limit | G0232L | G0031L | G0000L | G1232L | G0031L | 7 |
| Upper Fuzzy Limit | G0232H | G0031H | G0000H | G1232H | G0031H | |
| Lower Fuzzy Limit | G0200L | G0031L | G1232L | G0031L | G0031L | 8 |
| Upper Fuzzy Limit | G0200H | G0031H | G1232H | G0031H | G0031H | |
| Lower Fuzzy Limit | G0200L | G1233L | G0000L | G0031L | G0031L | 9 |
| Upper Fuzzy Limit | G0200H | G1233H | G0000H | G0031H | G0031H | |
| Lower Fuzzy Limit | G0031L | G1233L | G1233L | G1233L | G1233L | 10 |
| Upper Fuzzy Limit | G0031H | G1233H | G1233H | G1233H | G1233H | |
| Lower Fuzzy Limit | G0000L | G0000L | G0233L | G0233L | G0000L | I Love You |
| Upper Fuzzy Limit | G0000H | G0000H | G0233H | G0233H | G0000H | |

3.4.5 Translation of letters and numbers from sign language to written language

Upon the construction of the sign dictionary, the recognition phase for the full entries would be developed. Since the aim of the project is to prove the capability of the system to translate, this level of implementation, by considering letters and ten numbers, is considered sufficient.

The translation is done by getting the values of the sensors and comparing with the corresponding lower and upper fuzzy limits for each entry in the dictionary table. Once the reading of the five sensors fall within all the corresponding lower and upper limits of a particular entry, the associated word is recognized as the equivalent word for the sign input.

The maximum size of entries which can be recognized is potentially governed by the memory size of the microcontroller. The complete source code implemented is listed and fully commented in Appendix D, Appendix E and Appendix F.

A full translation example is explained and shown in Figure 17.

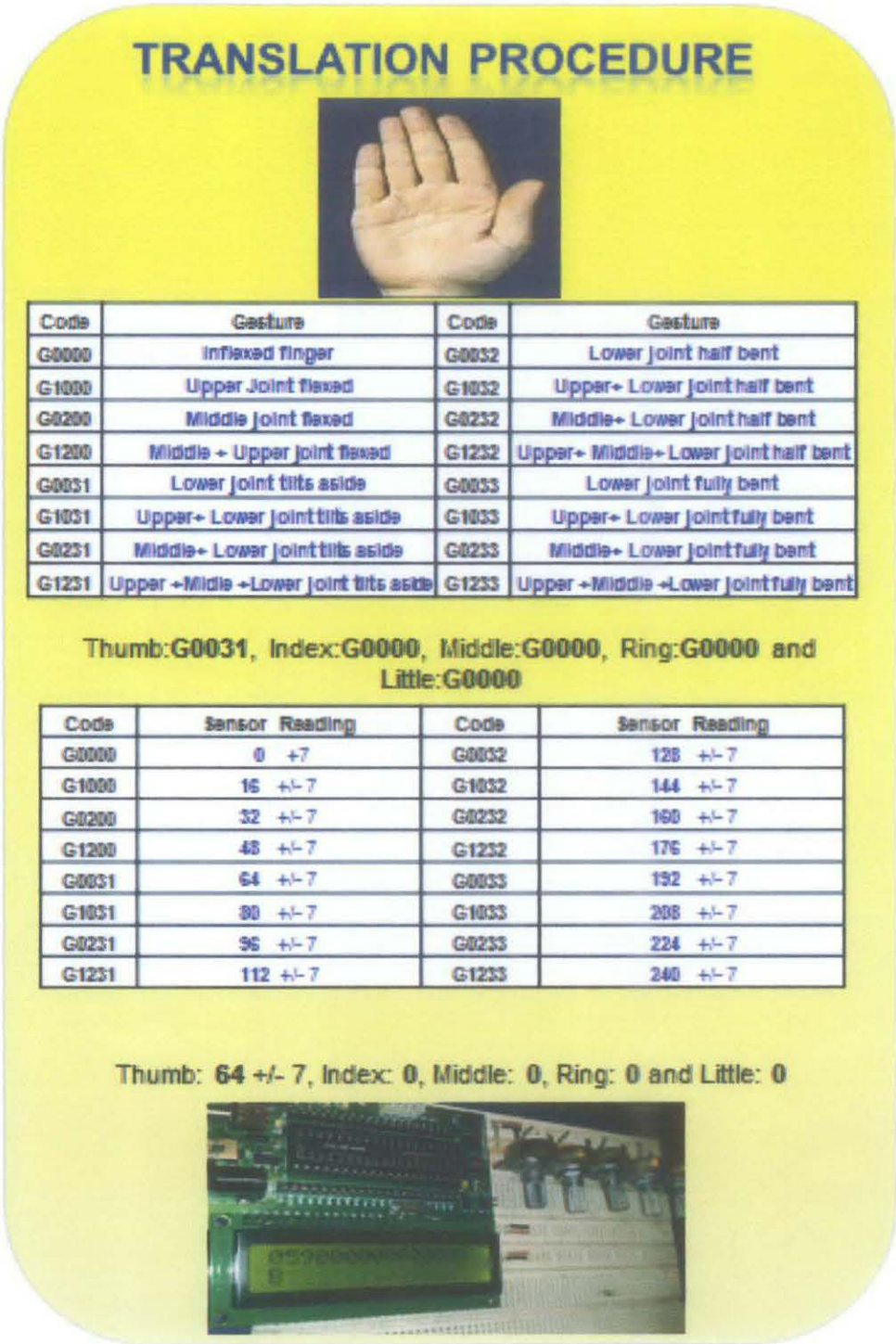


Figure 17: Full Translation Procedure Example

3.5 Summary

The methodology in implementing the translation system including the hardware, software and algorithm aspects is thoroughly discussed in this chapter. The results of some experimental work are reported in the next chapter.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter reports part of the results of the work followed by a section for discussion.

4.1 Sensor Reading and Display

In this experiment five potentiometers are used to emulate real sensor readings. The potentiometers have three terminals (see Figure 18); when connecting the first and the third terminals to VDD and GND, the output voltage will be in the range of 0-5 V depending on the position of the wiper. This range of voltages are converted using the built-in ADC in the PIC. The converted values are shown in the range of 0 to 255, whereby 0 is 0 V and 255 is 5 V.

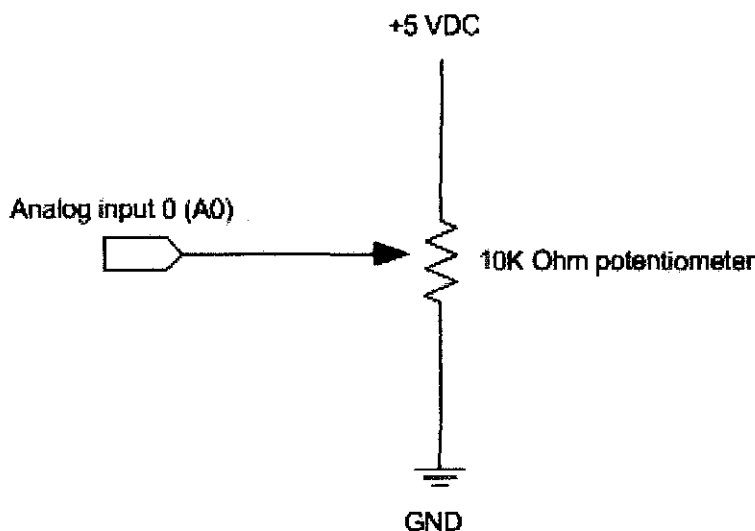


Figure 18: Potentiometer Connected to Analog input

A 2x16 LCD is used to display the values of sensors (potentiometers) readings. The working circuit with instantaneous conversion and display is shown in Figure 19. This figure shows the five potentiometers connected to the PIC to pins: A0, A1, A2, A3, and A5. The ADC unit converts the values and the PIC displays the values via the LCD.

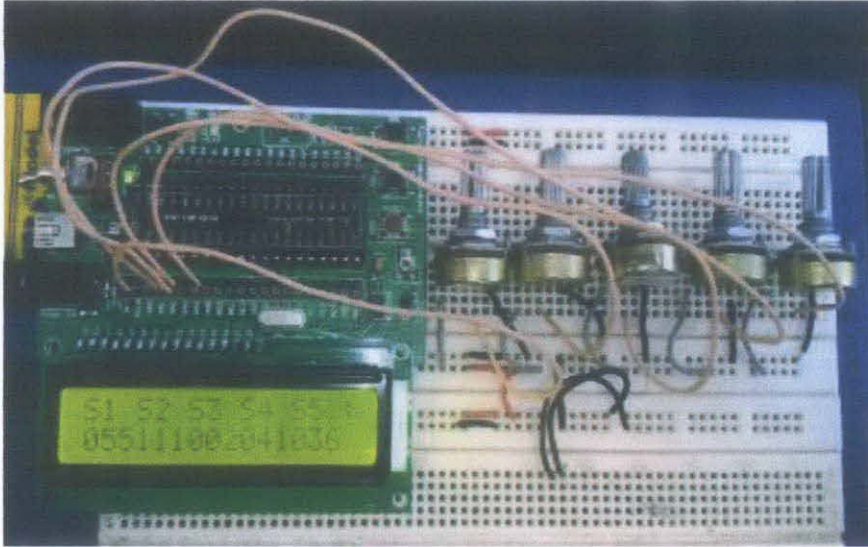
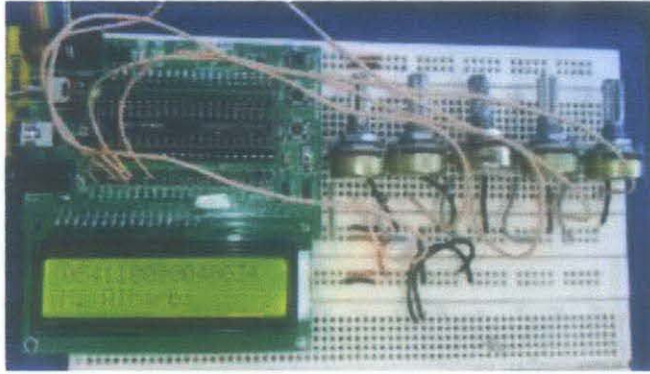


Figure 19: 5 Potentiometer, LCD and PIC interfacing circuitry

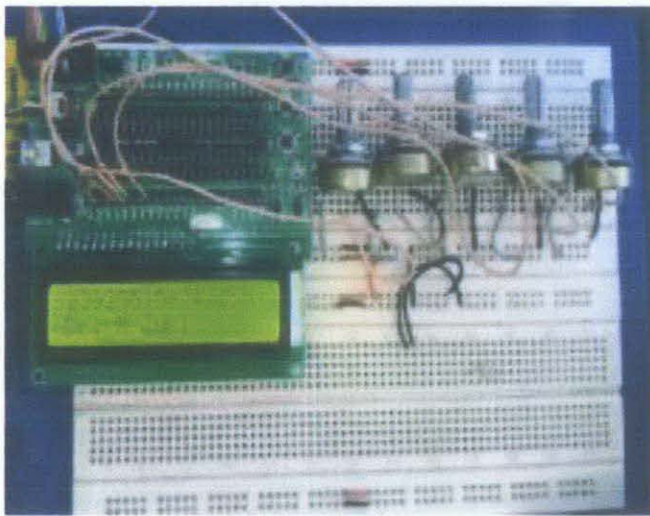
4.2 Basic Translation Based on Sensors Reading

In this experiment the readings from sensors are compared to different arbitrary values to show different messages accordingly. This experiment is an important start towards the full translation system. This is because; the translation likely comprises a lookup table and a set of comparisons with sensor readings to show a particular word.

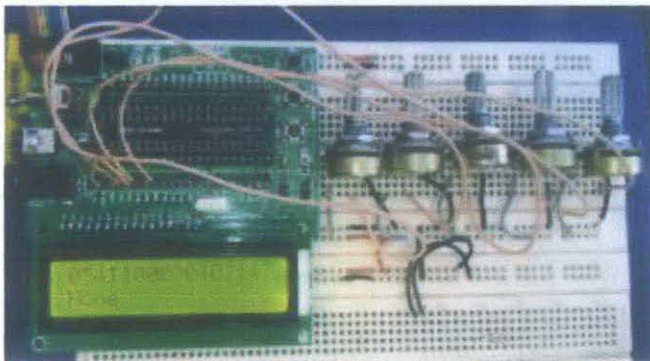
Three messages were used in this experiment: “M. Alharbi”, “Dr. Zuki” and “None” - when the sensors are all under 125, above 125, and otherwise, respectively. Figure 20 shows the three cases, whereby the LCD displays the reading of the sensors on the first line while displaying the equivalent message on the second line.



(a)



(b)



(c)

Figure 20: Basic Translation Based on Sensors Reading (a) Displaying “M. Alharbi” (b) Displaying “Dr. M Zuki” (c) Displaying “None”

4.3 Translation System Using a Set of Potentiometers

In this experiment, a translation system based on a set of potentiometers is considered. The system is implemented based on the codes listed in Appendix D, Appendix E and Appendix F. The system with the five potentiometers, LCD and mother board is shown in Figure 21. Figure 22 shows the system starting message.



Figure 21: Translation System Components: 5 Potentiometers, Main Board (SK40C board), and LCD

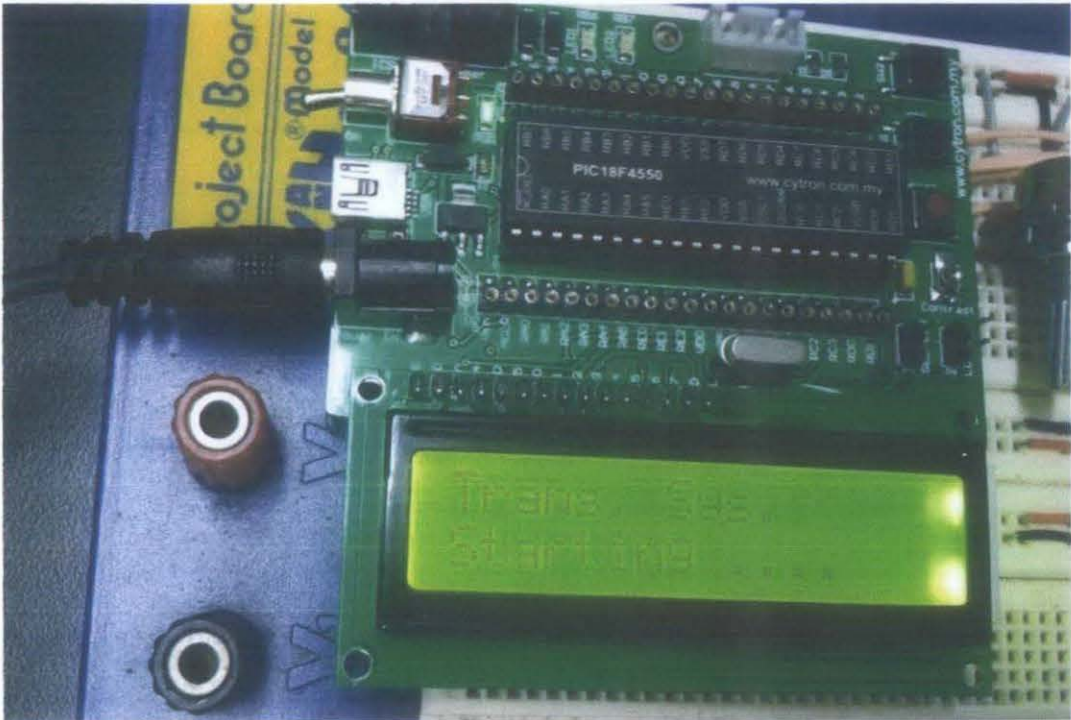


Figure 22: Translation System Startup, LCD is displaying the "Starting" message

The gestures for the letters, numbers and some words were obtained and implemented on the code listing. The reading of the sensors and the recognized gestures are configured to be displayed on the first and second lines of the LCD, respectively. Two modes of sensors reading display were shown on the LCD successfully. The two modes of display are the digitized (0-255) and in Volts (0-5 V) and are shown in Figure 23 and Figure 24, respectively.

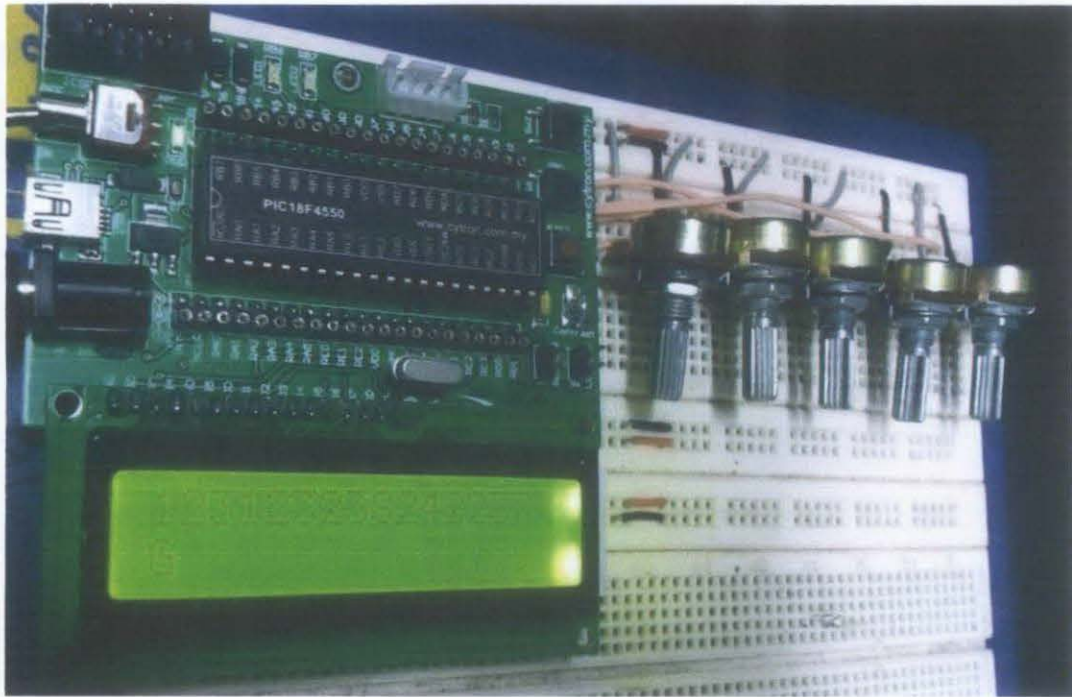


Figure 23: Sensor Readings in First Line (thumb: 123, index:123, middle:238, ring:242, and little:237), Second Line Displaying the Translated Sign

Several experiments had been conducted to test the ability of the system to recognize gestures. The experimental results for system show that it is able to recognize the whole 26 alphabetical letters.

Capital letters are used in coding as well as in display. This is to say that the LCD displays “B” instead of “b” to indicate the second alphabet. All the 26 letters did not involve motion except “Z”. Since it is assumed that only hand shapes are considered therefore the hand shape component of the letter is only considered in modeling.

Figure 23, Figure 24 and Figure 25 show some of the recognized letters.

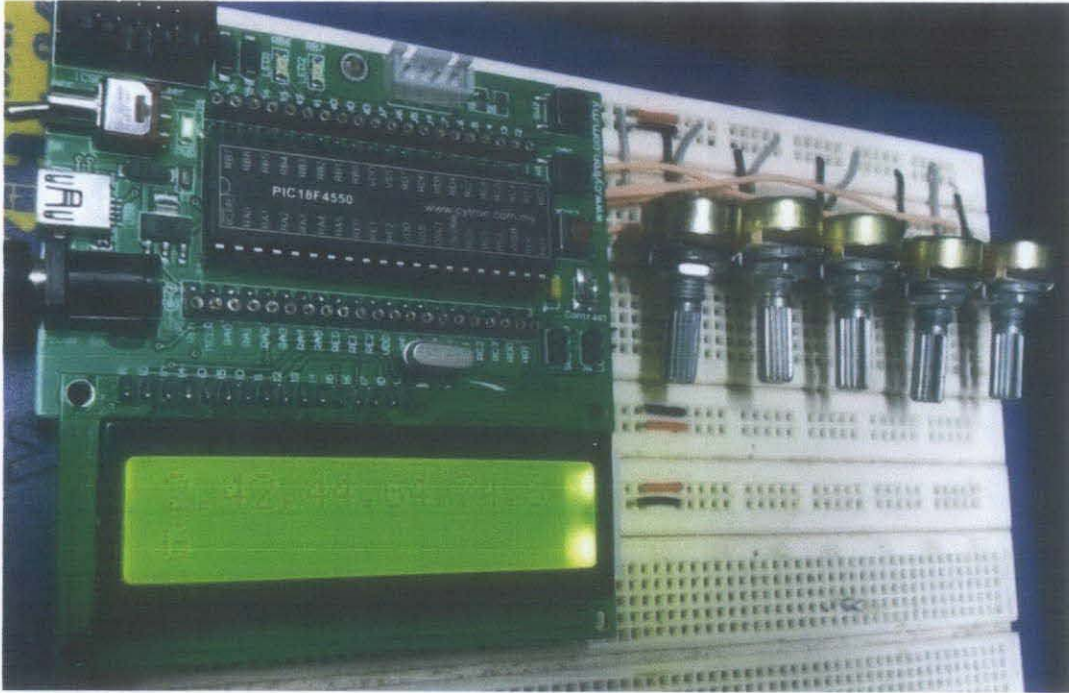


Figure 24: Sensor Readings in Volts (thumb: 2.4V, index:2.4V, middle:4.6V, ring:4.7V and little:4.6V)

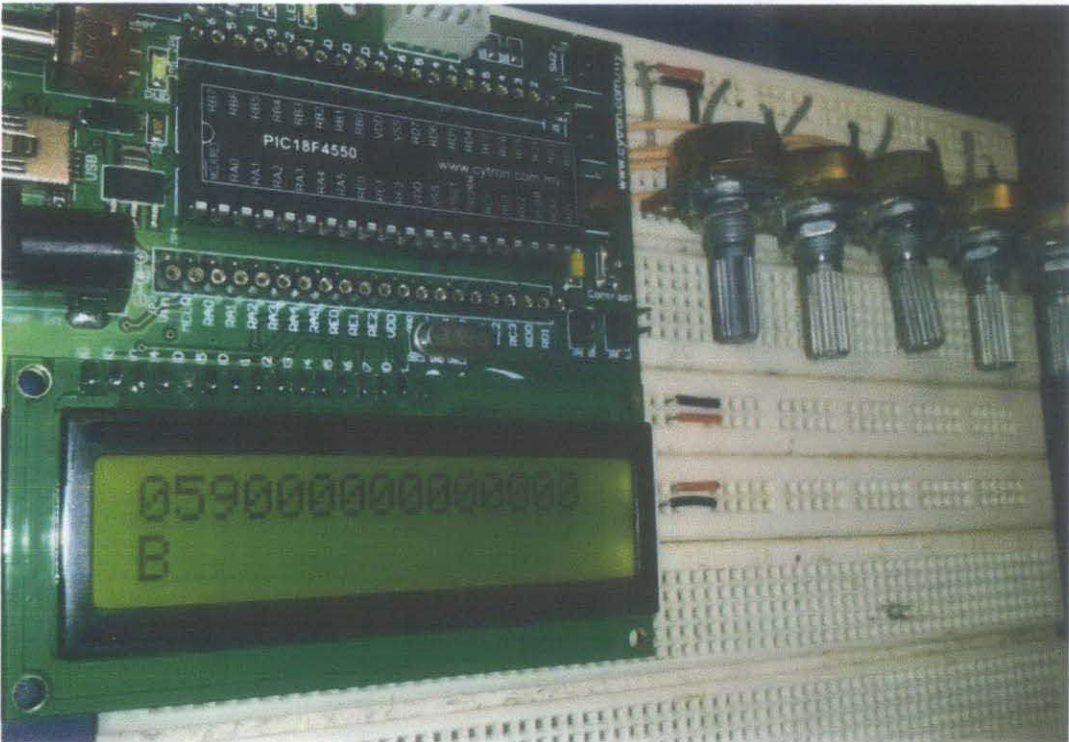


Figure 25: Translation System Recognizing the Sign for B Equivalent to (thumb: 64+/-7, index: 0, middle: 0, ring: 0 and little: 0)

Additionally numbers (0-10) are added and recognized successfully. Moreover, the system is able to recognize some phrases e.g. “I love you”. Figure 26 and Figure 27 show some of the obtained results.

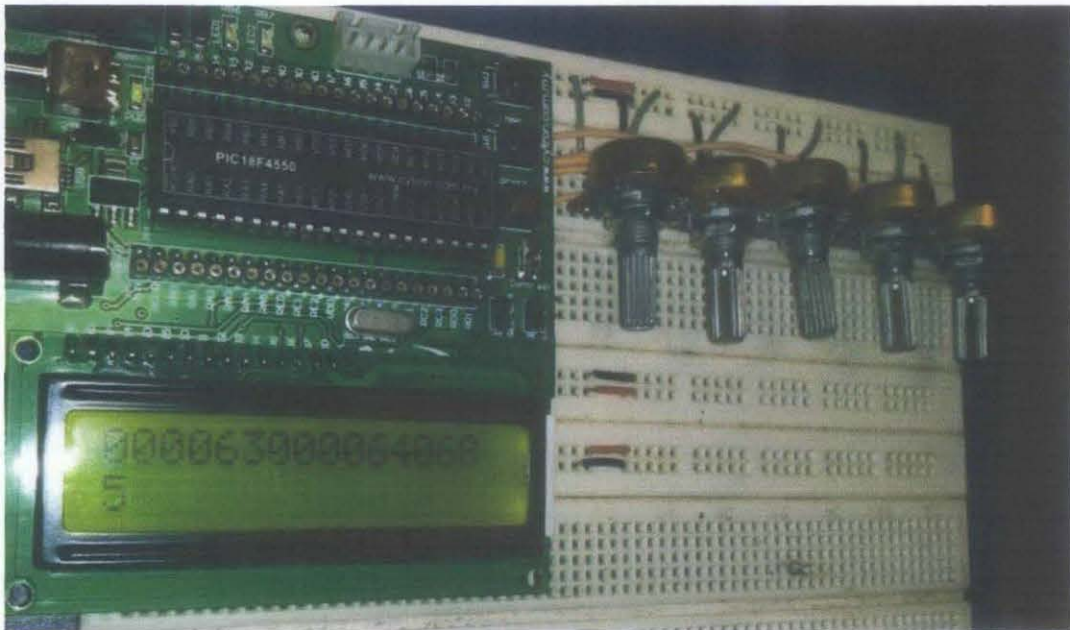


Figure 26: Recognition and Translation of the Sign of 5 (thumb: 0, index: 64+/-7, middle: 0, ring: 64+/-7, little: 64+/-7)

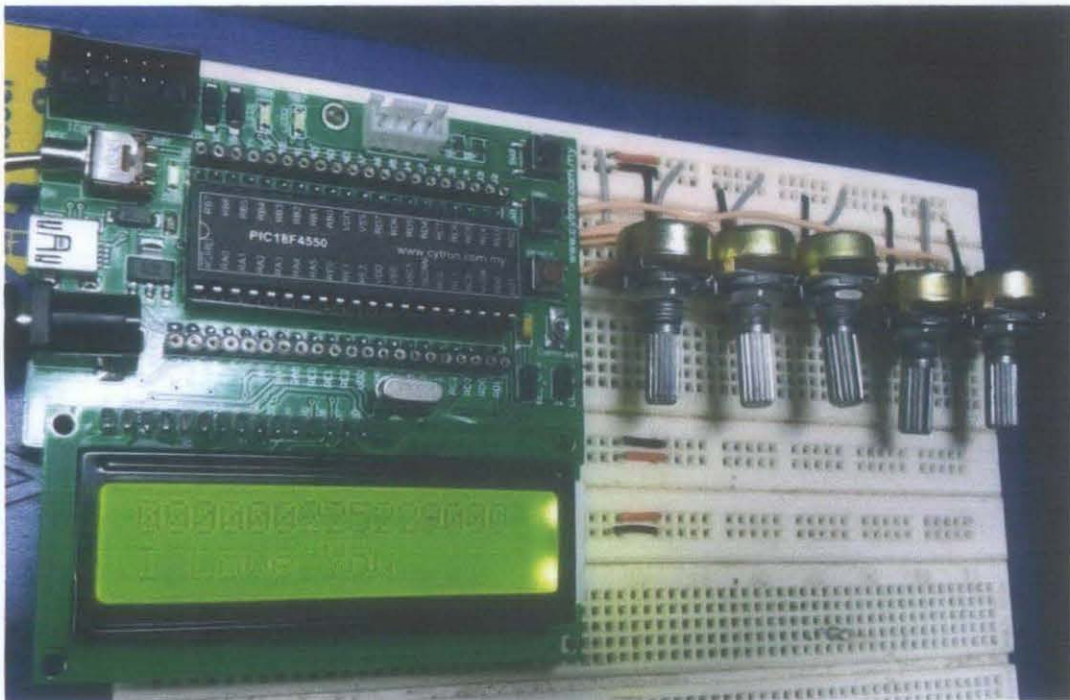


Figure 27: Recognition of Phrases e.g. “I Love You” (thumb: 0, index: 0, middle: 224+/-7, ring: 224+/-7, little: 0)

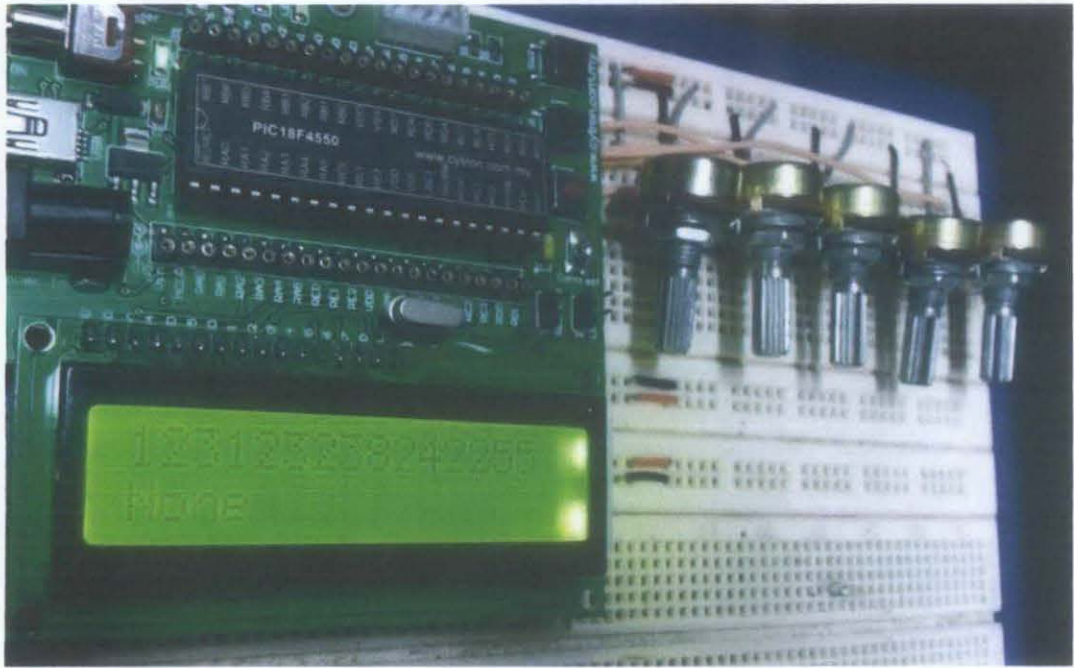


Figure 28: “None” Message for Any Other Unrecognized Signs

For unrecognized signs, the system displays by default “None” message as shown in Figure 28. This is later replaced in the coding by the message “Not recognized!”

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

In this chapter some conclusive statements on the progress of the proposed project and the expected future work are discussed.

5.1 Conclusion

A prototype incorporating five potentiometers – to simulate a realistic sensor reading, PIC microcontroller and LCD modules is proposed to aid sign language users to convey their messages in a more explicit way. The proposed prototype is based on ASL language and can support up to 75 signs and the equivalent words as a proof-of-concept. The project is envisaged to be an entry work for educational yet practical solutions which can potentially be extended for more functionality and portability.

5.2 Future Directions

Currently the system supports the translation of up to 75 signs/words. The size of the dictionary can be potentially extended considering the addition of memory chip to the system. To enable the portability of the design, a 9v battery module is to be added. Even though the system is tested without a realistic data glove, it is believed that the systematical methodology adopted in the project will ease the realization of the addition. A potential future work is to replace the five potentiometers by a data glove. An investigation for such interface is carried out as a part of this project and preliminary results are obtained and reported in Appendix G, to be exploited by future developers.

REFERENCES

- [1] Trevor Johnston and Adam Schembri, "Australian Sign Language (Auslan): An Introduction to Sign Language Linguistics", Cambridge university press, 2007
- [2] American Sign Language (ASL) dictionary, URL: <http://www.lifeprint.com/dictionary.htm>, retrieved: Nov 2011
- [3] Allen J.M., Foulds R.A., "An approach to animating sign language: A spoken english to sign english translator system", Proceedings of the Northeast Conference, 30, pp. 43-44, 2004
- [4] Akmeliawati, R.; Ooi, M.P.-L.; Ye Chow Kuang; , "Real-Time Malaysian Sign Language Translation using Colour Segmentation and Neural Network," Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE , vol., no., pp.1-6, 1-3 May 2007
- [5] Mekala, P.; Gao, Y.; Fan, J.; Davari, A.; , "Real-time sign language recognition based on neural network architecture," System Theory (SSST), 2011 IEEE 43rd Southeastern Symposium on , vol., no., pp.195-199, 14-16 March 2011
- [6] Nijusekar, C.; Brindhu Kumari, A.; , "Translating the sign of dumb person using ARM processor," Communication Control and Computing Technologies (ICCCCT), 2010 IEEE International Conference on , vol., no., pp.508-513, 7-9 Oct. 2010
- [7] McGuire, R.M.; Hernandez-Rebollar, J.; Starner, T.; Henderson, V.; Brashear, H.; Ross, D.S.; , "Towards a one-way American sign language translator," Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on , vol., no., pp. 620- 625, 17-19 May 2004
- [8] El-Bendary, N.; Zawbaa, H.M.; Daoud, M.S.; Hassanien, A.E.; Nakamatsu, K.; , "ArSLAT: Arabic Sign Language Alphabets Translator," Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on , vol., no., pp.590-595, 8-10 Oct. 2010
- [9] Fuzzy Logic, Wikipedia, URL: http://en.wikipedia.org/wiki/Fuzzy_logic, retrieved: Nov 2011
- [10] USB HID PC and PIC interface implementation code using C# and CCS compiler, Muhammad Rafique, URL: <http://www.pudn.com/downloads195/doc/project/detail916558.html>, retrieved: Nov 2011
- [11] Microchip Technology Inc., "PIC18F2455/2550/4455/4550 Data Sheet 28/40-Pin High-Performance: Enhanced Flash, USB Microcontrollers with nanoWatt Technology", U.S.A, 2006
- [12] Cytron Technologies, "SK40C PIC microcontroller start-up kit: User's Manual", Malaysia, November 2011.

[13] Cytron Technologies, "UIC00B USB ICSP PIC Programmer: User's Manual", Malaysia, November 2011.

APPENDIX A

PROJECT GANTT CHART

| No. | Phase | FYP 1 | | | | | | | | | | | | | | FYP 2 | | | | | | | | | | | | | |
|-----|--|-------|---|---|---|---|---|---|---|---|----|----|----|----|----|-------|---|---|---|---|---|---|---|---|----|----|----|----|----|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | SELECTION OF PROJECT TOPIC | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | - Project Objectives/Problem formulation | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | LITERATURE WORK | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | - Study about related work | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | - Study about microcontrollers/compiler | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | - Extended Proposal | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | Methodology | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | System identification/ Tools to be used | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | -Interim report | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | Hardware development | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | - Circuit Interfacing / Testing | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | - Progress report | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | Software development | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | C code developing | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Programming/ Testing | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | Prototype testing and troubleshooting | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | Prototype finalization | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | Poster presentation/ Draft report | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | Final Report/Viva | | | | | | | | | | | | | | | | | | | | | | | | | | | | |



Reports Submissions



Process

APPENDIX B

FIVE SENSORS READING DISPLAY (CODE)

```
#include "main.h"
#include "LCD.h"

void main()
{
    setup_adc_ports(AN0_TO_AN5|VSS_VDD);
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    //Setup_Oscillator parameter not selected from Intr Oscillator Config tab

    char message1[] = "S1 S2 S3 S4 S5 :";
    char message2[] = "          ";
    int8 S1,S2,S3,S4,S5;

    lcd_init();

    lcd_display_str(0,message1 );

    while (TRUE) {
        set_adc_channel(0);
        delay_us(60);
        S1=read_adc();
        set_adc_channel(1);
        delay_us(60);
        S2=read_adc();
        set_adc_channel(2);
        delay_us(60);
        S3=read_adc();
        set_adc_channel(3);
        delay_us(60);
        S4=read_adc();
        set_adc_channel(4);
        delay_us(60);
        S5=read_adc();
        sprintf(message2, "%03u%03u%03u%03u%03u ",S1,S2,S3,S4,S5);
        lcd_display_str(1,message2 );
    }
}
```

APPENDIX C

FIVE SENSOR INTERFACE AND BASIC TRANSLATION SYSTEM (CODE)

```
void main()
{
    setup_adc_ports(AN0_TO_AN5|VSS_VDD);
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);

    //Setup_Oscillator parameter not selected from Intr Oscillator Config tab

    char message1[] = "S1 S2 S3 S4 S5 :";
    char message2[] = "          ";
    int8 S1,S2,S3,S4,S5;

    lcd_init();

    while (TRUE) {
        set_adc_channel(0);
        delay_us(60);
        S1=read_adc();
        set_adc_channel(1);
        delay_us(60);
        S2=read_adc();
        set_adc_channel(2);
        delay_us(60);
        S3=read_adc();
        set_adc_channel(3);
        delay_us(60);
        S4=read_adc();
        set_adc_channel(4);
        delay_us(60);
        S5=read_adc();
        sprintf(message1, "%03u%03u%03u%03u%03u ", S1,S2,S3,S4,S5);
        lcd_display_str(0,message1 );
        if ( (S1 < 128) && (S2 < 128) && (S3 < 128) && (S4 < 128) && (S5 < 128) )
        {
            message2="M. Alharbi          ";
            lcd_display_str(1,message2 );
        }
        else if ( (S1 > 128) && (S2 > 128) && (S3 > 128) && (S4 > 128) && (S5 > 128) )
        {
            message2="Dr. M Zuki          ";
            lcd_display_str(1,message2 );
        }
        else
        {
            message2="None          ";
            lcd_display_str(1,message2 );
        }
    }
}
```

APPENDIX D

TRANSLATION SYSTEM WITH SIGN DICTIONARY (MAIN.C)

```
#include "main.h"           //Setting of fuses are there
#include "LCD.h"             //Nonstandard made file to communicate with the LCD
#include <string.h>          //To enable the use of string comparison=>strcmp()

//Definition area
#define PB1 PIN_B0          //Push button connected to PIN_B0
#define PRESSED 0           //The value when PB is pressed
#define MAX_DICT_ENTRIES 50 //The maximum reserved number of dictionary entries

//Function prototype:       //Description:
void init_device();        //initialize the peripherals of the PIC
void read_sensors();       //Acquiring the readings from sensors at Port A
void dict_init();          //Filling in all dictionary entries
void welcome_msg();        //Display a welcoming message at system starting
void translate();          //Compare the reading and recognize the word

//Variables
char message[16];          //a variable for the use with LCD
char translated[16];       //a variable to contain the translated word
int8 tmb,idx,mdl,rng,ltl;  //sensors reading of thumb, index, middle, ring and little
float tmb_V,idx_V,mdl_V,rng_V,ltl_V; //sensors reading in voltage
int8 PB_state;             //a dummy variable to store the last state of the PB

//The main function
void main()
{
    init_device();          //initialize device
    dict_init();            //initialize the dictionary
    welcome_msg();          //display welcoming message

    PB_state=0;             //reset the state of the PB
    while (TRUE)
    {
        read_sensors();    //read the values of sensors

        //Toggle the state of PB when pressed
        if(input(PB1)==PRESSED) //if PB pressed
        {
            while(input(PB1)==PRESSED) delay_ms(50); //Keep looping while PB is pressed
            PB_state=!PB_state;                       //Toggle the state of the PB
        }
        //Display the sensors in 0-255 or 0-5v ranges
        //0-255 range
        if(PB_state==0) sprintf(message, "%03u%03u%03u%03u ",tmb,idx,mdl,rng,ltl);
        //0-5v range
        if(PB_state==1)                                     sprintf(message,
"%01.1f%01.1f%01.1f%01.1f%01.1f",tmb_V,idx_V,mdl_V,rng_V,ltl_V);
        lcd_display_str(0,message );

        translate();                                       //Translate the reading
        lcd_display_str(1,translated);
    }
}

//Initialize the device according to project wizard setting
void init_device ()
{
    setup_adc_ports(AN0_TO_AN5|VSS_VDD);
    setup_adc(ADC_CLOCK_DIV_2);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_spi2(SPI_SS_DISABLED);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
```

```

    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);
    set_tris_b(0x03); //PIN_B0-2: INPUTS, PIN_B3-7: OUTPUTS
    lcd_init();
}

//read sensors as follows:
//A0: thumb, A1: index, A2: middle, A3: ring, A4: little
void read_sensors()
{
    set_adc_channel(0);
    delay_us(60);
    tmb=read_adc();
    set_adc_channel(1);
    delay_us(60);
    idx=read_adc();
    set_adc_channel(2);
    delay_us(60);
    mdl=read_adc();
    set_adc_channel(3);
    delay_us(60);
    rng=read_adc();
    set_adc_channel(4);
    delay_us(60);
    ltl=read_adc();
    //convert to voltage
    tmb_V=(float)5/255*tmb;
    idx_V=(float)5/255*idx;
    mdl_V=(float)5/255*mdl;
    rng_V=(float)5/255*rng;
    ltl_V=(float)5/255*ltl;
}

// Display the welcoming messages
void welcome_msg()
{
    sprintf(message, "Trans. Sys.");
    lcd_display_str(0,message );
    sprintf(message, "Starting .");
    lcd_display_str(1,message);
    delay_ms(100);
    sprintf(message, "Starting ..");
    lcd_display_str(1,message);
    delay_ms(100);
    sprintf(message, "Starting ...");
    lcd_display_str(1,message);
    delay_ms(100);
    sprintf(message, "Starting ....");
    lcd_display_str(1,message);
    delay_ms(100);
    sprintf(message, "Starting .....");
    lcd_display_str(1,message);
    delay_ms(100);
    sprintf(message, "Starting .....");
    lcd_display_str(1,message);
    delay_ms(100);
}

//Defining the data structure of ONE single sign word and the equivalent:
//Sign representation:
//dictionary_enty.tmbFZL: the fuzzy low limit for thumb gesture
//dictionary_enty.tmbFZH: the fuzzy high limit for thumb gesture
//dictionary_enty.idxFZL: the fuzzy low limit for index gesture
//dictionary_enty.idxFZH: the fuzzy high limit for index gesture
//dictionary_enty.mdlFZL: the fuzzy low limit for middle gesture
//dictionary_enty.mdlFZH: the fuzzy high limit for middle gesture
//dictionary_enty.rngFZL: the fuzzy low limit for ring gesture
//dictionary_enty.rngFZH: the fuzzy high limit for ring gesture
//dictionary_enty.ltlFZL: the fuzzy low limit for little gesture
//dictionary_enty.ltlFZH: the fuzzy high limit for little gesture
//the Equivalent in written English:
//dictionary_enty.word: the equivalent word (max size is 16 characters)
typedef struct{
int8 tmbFZL;
int8 tmbFZH;
int8 idxFZL;
int8 idxFZH;
int8 mdlFZL;

```



```

int8 mdlFZH;
int8 rngFZL;
int8 rngFZH;
int8 ltlFZL;
int8 ltlFZH;
char word[16];
} dictionary_entry; //a dictionary containing 50 words

dictionary_entry dict[MAX_DICT_ENTRIES]; //Declare a array of dictionary_entry data structure

//Gesture lists and equivalent representation
#define BS 7
#define G0000 0
#define G0000L G0000
#define G0000H G0000+BS
#define G1000 16
#define G1000L G1000-BS
#define G1000H G1000+BS
#define G0200 32
#define G0200L G0200-BS
#define G0200H G0200+BS
#define G1200 48
#define G1200L G1200-BS
#define G1200H G1200+BS
#define G0031 64
#define G0031L G0031-BS
#define G0031H G0031+BS
#define G1031 80
#define G1031L G1031-BS
#define G1031H G1031+BS
#define G0231 96
#define G0231L G0231-BS
#define G0231H G0231+BS
#define G1231 112
#define G1231L G1231-BS
#define G1231H G1231+BS
#define G0032 128
#define G0032L G0032-BS
#define G0032H G0032+BS
#define G1032 144
#define G1032L G1032-BS
#define G1032H G1032+BS
#define G0232 160
#define G0232L G0232-BS
#define G0232H G0232+BS
#define G1232 176
#define G1232L G1232-BS
#define G1232H G1232+BS
#define G0033 192
#define G0033L G0033-BS
#define G0033H G0033+BS
#define G1033 208
#define G1033L G1033-BS
#define G1033H G1033+BS
#define G0233 224
#define G0233L G0233-BS
#define G0233H G0233+BS
#define G1233 240
#define G1233L G1233-BS
#define G1233H G1233+BS
#define _ MAX_DICT_ENTRIES-1

//The initialization of actual dictionary
void dict_init()
{
dict[ 0].word="A";
dict[ 0].tmbFZL=G1031L;dict[ 0].idxFZL=G0233L;dict[ 0].mdlFZL=G0233L;dict[
0].rngFZL=G1233L;dict[ 0].ltlFZL=G1233L;dict[ 0].idxFZH=G0233H;dict[ 0].mdlFZH=G0233H;dict[
0].rngFZH=G1233H;dict[ 0].ltlFZH=G1233H;

dict[ 1].word="B";
dict[ 1].tmbFZL=G0031L;dict[ 1].idxFZL=G0000L;dict[ 1].mdlFZL=G0000L;dict[
1].rngFZL=G0000L;dict[ 1].ltlFZL=G0000L;dict[ 1].idxFZH=G0000H;dict[ 1].mdlFZH=G0000H;dict[
1].rngFZH=G0000H;dict[ 1].ltlFZH=G0000H;

dict[ 2].word="C";

```

| | |
|--|---|
| dict[2].tmbFZL=G0032L;dict[2].rngFZL=G1200L;dict[2].ltlFZL=G1200L;dict[2].tmbFZH=G0032H;dict[2].rngFZH=G1200H;dict[2].ltlFZH=G1200H; | 2).idxFZL=G1200L;dict[2].idxFZH=G1200H;dict[2].mdlFZL=G1200L;dict[2].mdlFZH=G1200H;dict[2]. |
| dict[3].word="D";dict[3].tmbFZL=G1232L;dict[3].rngFZL=G1232L;dict[3].ltlFZL=G1232L;dict[3].tmbFZH=G1232H;dict[3].rngFZH=G1232H;dict[3].ltlFZH=G1232H; | 3).idxFZL=G0000L;dict[3].idxFZH=G0000H;dict[3].mdlFZL=G1232L;dict[3].mdlFZH=G1232H;dict[3]. |
| dict[4].word="E";dict[4].tmbFZL=G1231L;dict[4].rngFZL=G1200L;dict[4].ltlFZL=G1200L;dict[4].tmbFZH=G1231H;dict[4].rngFZH=G1200H;dict[4].ltlFZH=G1200H; | 4).idxFZL=G1200L;dict[4].idxFZH=G1200H;dict[4].mdlFZL=G1200L;dict[4].mdlFZH=G1200H;dict[4]. |
| dict[5].word="F";dict[5].tmbFZL=G1032L;dict[5].rngFZL=G0031L;dict[5].ltlFZL=G0031L;dict[5].tmbFZH=G1032H;dict[5].rngFZH=G0031H;dict[5].ltlFZH=G0031H; | 5).idxFZL=G0232L;dict[5].idxFZH=G0232H;dict[5].mdlFZL=G0000L;dict[5].mdlFZH=G0000H;dict[5]. |
| dict[6].word="G";dict[6].tmbFZL=G0032L;dict[6].rngFZL=G1233L;dict[6].ltlFZL=G1233L;dict[6].tmbFZH=G0032H;dict[6].rngFZH=G1233H;dict[6].ltlFZH=G1233H; | 6).idxFZL=G0032L;dict[6].idxFZH=G0032H;dict[6].mdlFZL=G1233L;dict[6].mdlFZH=G1233H;dict[6]. |
| dict[7].word="H";dict[7].tmbFZL=G1232L;dict[7].rngFZL=G0233L;dict[7].ltlFZL=G0233L;dict[7].tmbFZH=G1232H;dict[7].rngFZH=G0233H;dict[7].ltlFZH=G0233H; | 7).idxFZL=G0032L;dict[7].idxFZH=G0032H;dict[7].mdlFZL=G0032L;dict[7].mdlFZH=G0032H;dict[7]. |
| dict[8].word="I";dict[8].tmbFZL=G1200L;dict[8].rngFZL=G0233L;dict[8].ltlFZL=G0000L;dict[8].tmbFZH=G1200H;dict[8].rngFZH=G0233H;dict[8].ltlFZH=G0000H; | 8).idxFZL=G0233L;dict[8].idxFZH=G0233H;dict[8].mdlFZL=G0233L;dict[8].mdlFZH=G0233H;dict[8]. |
| dict[9].word="J";dict[9].tmbFZL=G0200L;dict[9].rngFZL=G0233L;dict[9].ltlFZL=G0000L;dict[9].tmbFZH=G0200H;dict[9].rngFZH=G0233H;dict[9].ltlFZH=G0000H; | 9).idxFZL=G0233L;dict[9].idxFZH=G0233H;dict[9].mdlFZL=G0233L;dict[9].mdlFZH=G0233H;dict[9]. |
| dict[10].word="K";dict[10].tmbFZL=G0232L;dict[10].idxFZL=G0000L;dict[10].mdlFZL=G0032L;dict[10].rngFZL=G0233L;dict[10].ltlFZL=G0233L;dict[10].tmbFZH=G0232H;dict[10].idxFZH=G0000H;dict[10].mdlFZH=G0032H;dict[10].rngFZH=G0233H;dict[10].ltlFZH=G0233H; | |
| dict[11].word="L";dict[11].tmbFZL=G0000L;dict[11].idxFZL=G0000L;dict[11].mdlFZL=G0233L;dict[11].rngFZL=G0233L;dict[11].ltlFZL=G0233L;dict[11].tmbFZH=G0000H;dict[11].idxFZH=G0000H;dict[11].mdlFZH=G0233H;dict[11].rngFZH=G0233H;dict[11].ltlFZH=G0233H; | |
| dict[12].word="M";dict[12].tmbFZL=G1233L;dict[12].idxFZL=G1232L;dict[12].mdlFZL=G1232L;dict[12].rngFZL=G1232L;dict[12].ltlFZL=G0233L;dict[12].tmbFZH=G1233H;dict[12].idxFZH=G1232H;dict[12].mdlFZH=G1232H;dict[12].rngFZH=G1232H;dict[12].ltlFZH=G0233H; | |
| dict[13].word="N";dict[13].tmbFZL=G1232L;dict[13].idxFZL=G0232L;dict[13].mdlFZL=G0232L;dict[13].rngFZL=G0233L;dict[13].ltlFZL=G0233L;dict[13].tmbFZH=G1232H;dict[13].idxFZH=G0232H;dict[13].mdlFZH=G0232H;dict[13].rngFZH=G0233H;dict[13].ltlFZH=G0233H; | |
| dict[14].word="O";dict[14].tmbFZL=G1232L;dict[14].idxFZL=G1232L;dict[14].mdlFZL=G1232L;dict[14].rngFZL=G1232L;dict[14].ltlFZL=G1232L;dict[14].tmbFZH=G1232H;dict[14].idxFZH=G1232H;dict[14].mdlFZH=G1232H;dict[14].rngFZH=G1232H;dict[14].ltlFZH=G1232H; | |
| dict[15].word="P"; | |

```

dict[15].tmbFZL=G0032L;dict[15].idxFZL=G0000L;dict[15].mdlFZL=G0033L;dict[15].rngFZL=G0233L;di
ct[15].ltlFZL=G0233L;
dict[15].tmbFZH=G0032H;dict[15].idxFZH=G0000H;dict[15].mdlFZH=G0033H;dict[15].rngFZH=G0233H;di
ct[15].ltlFZH=G0233H;

dict[16].word="Q";
dict[16].tmbFZL=G0032L;dict[16].idxFZL=G0033L;dict[16].mdlFZL=G1233L;dict[16].rngFZL=G1233L;di
ct[16].ltlFZL=G1233L;
dict[16].tmbFZH=G0032H;dict[16].idxFZH=G0033H;dict[16].mdlFZH=G1233H;dict[16].rngFZH=G1233H;di
ct[16].ltlFZH=G1233H;

dict[17].word="R";
dict[17].tmbFZL=G0232L;dict[17].idxFZL=G0031L;dict[17].mdlFZL=G0000L;dict[17].rngFZL=G0232L;di
ct[17].ltlFZL=G0233L;
dict[17].tmbFZH=G0232H;dict[17].idxFZH=G0031H;dict[17].mdlFZH=G0000H;dict[17].rngFZH=G0232H;di
ct[17].ltlFZH=G0233H;

dict[18].word="S";
dict[18].tmbFZL=G0232L;dict[18].idxFZL=G1233L;dict[18].mdlFZL=G1233L;dict[18].rngFZL=G1233L;di
ct[18].ltlFZL=G1233L;
dict[18].tmbFZH=G0232H;dict[18].idxFZH=G1233H;dict[18].mdlFZH=G1233H;dict[18].rngFZH=G1233H;di
ct[18].ltlFZH=G1233H;

dict[19].word="T";
dict[19].tmbFZL=G0232L;dict[19].idxFZL=G0232L;dict[19].mdlFZL=G0233L;dict[19].rngFZL=G0233L;di
ct[19].ltlFZL=G0233L;
dict[19].tmbFZH=G0232H;dict[19].idxFZH=G0232H;dict[19].mdlFZH=G0233H;dict[19].rngFZH=G0233H;di
ct[19].ltlFZH=G0233H;

dict[20].word="U";
dict[20].tmbFZL=G0232L;dict[20].idxFZL=G0000L;dict[20].mdlFZL=G0000L;dict[20].rngFZL=G0232L;di
ct[20].ltlFZL=G0232L;
dict[20].tmbFZH=G0232H;dict[20].idxFZH=G0000H;dict[20].mdlFZH=G0000H;dict[20].rngFZH=G0232H;di
ct[20].ltlFZH=G0232H;

dict[21].word="V";
dict[21].tmbFZL=G0232L;dict[21].idxFZL=G0031L;dict[21].mdlFZL=G0031L;dict[21].rngFZL=G0233L;di
ct[21].ltlFZL=G0233L;
dict[21].tmbFZH=G0232H;dict[21].idxFZH=G0031H;dict[21].mdlFZH=G0031H;dict[21].rngFZH=G0233H;di
ct[21].ltlFZH=G0233H;

dict[22].word="W";
dict[22].tmbFZL=G0233L;dict[22].idxFZL=G0031L;dict[22].mdlFZL=G0000L;dict[22].rngFZL=G0031L;di
ct[22].ltlFZL=G0232L;
dict[22].tmbFZH=G0233H;dict[22].idxFZH=G0031H;dict[22].mdlFZH=G0000H;dict[22].rngFZH=G0031H;di
ct[22].ltlFZH=G0232H;

dict[23].word="X";
dict[23].tmbFZL=G1232L;dict[23].idxFZL=G1200L;dict[23].mdlFZL=G1233L;dict[23].rngFZL=G1233L;di
ct[23].ltlFZL=G1233L;
dict[23].tmbFZH=G1232H;dict[23].idxFZH=G1200H;dict[23].mdlFZH=G1233H;dict[23].rngFZH=G1233H;di
ct[23].ltlFZH=G1233H;

dict[24].word="Y";
dict[24].tmbFZL=G0000L;dict[24].idxFZL=G1233L;dict[24].mdlFZL=G1233L;dict[24].rngFZL=G1233L;di
ct[24].ltlFZL=G0000L;
dict[24].tmbFZH=G0000H;dict[24].idxFZH=G1233H;dict[24].mdlFZH=G1233H;dict[24].rngFZH=G1233H;di
ct[24].ltlFZH=G0000H;

dict[25].word="Z"; //motionless Z
dict[25].tmbFZL=G1232L;dict[25].idxFZL=G0000L;dict[25].mdlFZL=G1233L;dict[25].rngFZL=G1233L;di
ct[25].ltlFZL=G1233L;
dict[25].tmbFZH=G1232H;dict[25].idxFZH=G0000H;dict[25].mdlFZH=G1233H;dict[25].rngFZH=G1233H;di
ct[25].ltlFZH=G1233H;

dict[26].word="1";
dict[26].tmbFZL=G1233L;dict[26].idxFZL=G0000L;dict[26].mdlFZL=G1233L;dict[26].rngFZL=G1233L;di
ct[26].ltlFZL=G1233L;
dict[26].tmbFZH=G1233H;dict[26].idxFZH=G0000H;dict[26].mdlFZH=G1233H;dict[26].rngFZH=G1233H;di
ct[26].ltlFZH=G1233H;

dict[27].word="2";
dict[27].tmbFZL=G1232L;dict[27].idxFZL=G0031L;dict[27].mdlFZL=G0000L;dict[27].rngFZL=G1233L;di
ct[27].ltlFZL=G1233L;
dict[27].tmbFZH=G1232H;dict[27].idxFZH=G0031H;dict[27].mdlFZH=G0000H;dict[27].rngFZH=G1233H;di
ct[27].ltlFZH=G1233H;

```

```
dict[28].word="3";
dict[28].tmbFZL=G0000L;dict[28].idxFZL=G0031L;dict[28].mdlFZL=G0000L;dict[28].rngFZL=G1233L;di
ct[28].ltlFZL=G1233L;
dict[28].tmbFZH=G0000H;dict[28].idxFZH=G0031H;dict[28].mdlFZH=G0000H;dict[28].rngFZH=G1233H;di
ct[28].ltlFZH=G1233H;
```

```
dict[29].word="4";
dict[29].tmbFZL=G0233L;dict[29].idxFZL=G0031L;dict[29].mdlFZL=G0000L;dict[29].rngFZL=G0031L;di
ct[29].ltlFZL=G0031L;
dict[29].tmbFZH=G0233H;dict[29].idxFZH=G0031H;dict[29].mdlFZH=G0000H;dict[29].rngFZH=G0031H;di
ct[29].ltlFZH=G0031H;
```

```
dict[30].word="5";
dict[30].tmbFZL=G0000L;dict[30].idxFZL=G0031L;dict[30].mdlFZL=G0000L;dict[30].rngFZL=G0031L;di
ct[30].ltlFZL=G0031L;
dict[30].tmbFZH=G0000H;dict[30].idxFZH=G0031H;dict[30].mdlFZH=G0000H;dict[30].rngFZH=G0031H;di
ct[30].ltlFZH=G0031H;
```

```
dict[31].word="6";
dict[31].tmbFZL=G0232L;dict[31].idxFZL=G0031L;dict[31].mdlFZL=G0000L;dict[31].rngFZL=G0031L;di
ct[31].ltlFZL=G0232L;
dict[31].tmbFZH=G0232H;dict[31].idxFZH=G0031H;dict[31].mdlFZH=G0000H;dict[31].rngFZH=G0031H;di
ct[31].ltlFZH=G0232H;
```

```
dict[32].word="7";
dict[32].tmbFZL=G0232L;dict[32].idxFZL=G0031L;dict[32].mdlFZL=G0000L;dict[32].rngFZL=G1232L;di
ct[32].ltlFZL=G0031L;
dict[32].tmbFZH=G0232H;dict[32].idxFZH=G0031H;dict[32].mdlFZH=G0000H;dict[32].rngFZH=G1232H;di
ct[32].ltlFZH=G0031H;
```

```
dict[33].word="8";
dict[33].tmbFZL=G0200L;dict[33].idxFZL=G0031L;dict[33].mdlFZL=G1232L;dict[33].rngFZL=G0031L;di
ct[33].ltlFZL=G0031L;
dict[33].tmbFZH=G0200H;dict[33].idxFZH=G0031H;dict[33].mdlFZH=G1232H;dict[33].rngFZH=G0031H;di
ct[33].ltlFZH=G0031H;
```

```
dict[34].word="9";
dict[34].tmbFZL=G0200L;dict[34].idxFZL=G1233L;dict[34].mdlFZL=G0000L;dict[34].rngFZL=G0031L;di
ct[34].ltlFZL=G0031L;
dict[34].tmbFZH=G0200H;dict[34].idxFZH=G1233H;dict[34].mdlFZH=G0000H;dict[34].rngFZH=G0031H;di
ct[34].ltlFZH=G0031H;
```

```
dict[35].word="10";
dict[35].tmbFZL=G0031L;dict[35].idxFZL=G1233L;dict[35].mdlFZL=G1233L;dict[35].rngFZL=G1233L;di
ct[35].ltlFZL=G1233L;
dict[35].tmbFZH=G0031H;dict[35].idxFZH=G1233H;dict[35].mdlFZH=G1233H;dict[35].rngFZH=G1233H;di
ct[35].ltlFZH=G1233H;
```

```
dict[36].word="I Love You";
dict[36].tmbFZL=G0000L;dict[36].idxFZL=G0000L;dict[36].mdlFZL=G0233L;dict[36].rngFZL=G0233L;di
ct[36].ltlFZL=G0000L;
dict[36].tmbFZH=G0000H;dict[36].idxFZH=G0000H;dict[36].mdlFZH=G0233H;dict[36].rngFZH=G0233H;di
ct[36].ltlFZH=G0000H;
```

```
dict[___].word="";
dict[___].tmbFZL=G0000L;dict[___].idxFZL=G0000L;dict[___].mdlFZL=G0000L;dict[___].rngFZL=G0000L;di
ct[___].ltlFZL=G0000L;
dict[___].tmbFZH=G0000H;dict[___].idxFZH=G0000H;dict[___].mdlFZH=G0000H;dict[___].rngFZH=G0000H;di
ct[___].ltlFZH=G0000H;
}
```

//Make the translation by comparing and fuzzy high and fuzzy low limit for each gesture

```
void translate()
{
    int8 i;
    for(i=0;i<MAX_DICT_ENTRIES;i++)
        if ( ( tmb >= dict[i].tmbFZL ) && ( tmb <= dict[i].tmbFZH ) \
            && ( idx >= dict[i].idxFZL ) && ( idx <= dict[i].idxFZH ) \
            && ( mdl >= dict[i].mdlFZL ) && ( mdl <= dict[i].mdlFZH ) \
            && ( rng >= dict[i].rngFZL ) && ( rng <= dict[i].rngFZH ) \
            && ( ltl >= dict[i].ltlFZL ) && ( ltl <= dict[i].ltlFZH ) )
            { strcpy(translated,dict[i].word); break;}
    else
        strcpy(translated,"*Not Recognized!"); //If nothing recognized display "*Not
Recognized!"
}
```

APPENDIX E

TRANSLATION SYSTEM WITH SIGN DICTIONARY (MAIN.H)

```
#include <18F4550.h>
#device adc=8

#FUSES NOWDT           //No Watch Dog Timer
#FUSES WDT128          //Watch Dog Timer uses 1:128 Postscale
#FUSES HS              //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPROTECT       //Code not protected from reading
#FUSES NOBROWNOUT      //No brownout reset
#FUSES BORV20          //Brownout reset at 2.0V
#FUSES NOPUT          //No Power Up Timer
#FUSES NOCPD           //No EE protection
#FUSES STVREN          //Stack full/underflow will cause reset
#FUSES NODEBUG         //No Debug mode for ICD
#FUSES NOLVP           //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOWRT           //Program memory not write protected
#FUSES NOWRTD          //Data EEPROM not write protected
#FUSES IESO            //Internal External Switch Over mode enabled
#FUSES FCMEN           //Fail-safe clock monitor enabled
#FUSES PBDEN           //PORTB pins are configured as analog input channels on RESET
#FUSES NOWRTC          //configuration not registers write protected
#FUSES NOWRTB          //Boot block not write protected
#FUSES NOEBTR          //Memory not protected from table reads
#FUSES NOEBTRB         //Boot block not protected from table reads
#FUSES NOCPB           //No Boot Block code protection
#FUSES MCLR            //Master Clear pin enabled
#FUSES LPT1OSC         //Timer1 configured for low-power operation
#FUSES NOXINST         //Extended set extension and Indexed Addressing mode disabled
(Legacy mode)
#FUSES PLL12           //Divide By 12(48MHz oscillator input)
#FUSES CPUDIV4         //System Clock by 4
#FUSES USBDIV          //USB clock source comes from PLL divide by 2
#FUSES VREGEN          //USB voltage regulator enabled
#FUSES ICPRT           //ICPRT enabled

#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

APPENDIX F

TRANSLATION SYSTEM WITH SIGN DICTIONARY (LCD.H)

```
/*=====
8-BIT LCD DRIVER FOR PIC18F4550 CCSC
=====*/
/////////////////////////////////////////////////////////////////
// CCS C Compiler
// LCD 16x2
//
// by Nisar Ahmed
// 2009/03/27
// lately edited by Alharbi
// 2011/11/30
/////////////////////////////////////////////////////////////////

#define NCHAR_PER_LINE      16           // max char numbers per line
#define LCD_RS              PIN_B4
#define LCD_RW              PIN_B3
#define LCD_E               PIN_B5
#define LCD_DAT             PORT_D

#define PORT_A              0           // define for function output()
#define PORT_B              1
#define PORT_C              2
#define PORT_D              3
#define PORT_E              4

///////////////////////////////////////////////////////////////// output()
//lcd data bus output
void output(int8 port, int8 dat)
{
    switch(port)
    {
        case PORT_A: output_a(dat);      break;
        case PORT_B: output_b(dat);      break;
        case PORT_C: output_c(dat);      break;
        case PORT_D: output_d(dat);      break;
        case PORT_E: output_e(dat);      break;
        default :      //??? port maybe error!
            break;
    }
}
//end output()

///////////////////////////////////////////////////////////////// lcd_write_cmd()
//
void lcd_write_cmd(int8 cmd)
{
    delay_us(400);
    output_low(LCD_RS);
    output_low(LCD_RW);
    output(LCD_DAT, cmd);

    output_high(LCD_E);
    delay_us(400);
    output_low(LCD_E);
}
//end lcd_write_cmd()

///////////////////////////////////////////////////////////////// lcd_write_dat()
//
void lcd_write_dat(int8 dat)
{
    delay_us(400);
    output_high(LCD_RS);
    output_low(LCD_RW);
    output(LCD_DAT, dat);

    output_high(LCD_E);
    delay_us(400);
    output_low(LCD_E);
}
//end lcd_write_dat()

///////////////////////////////////////////////////////////////// lcd_init()
//
```

```

void lcd_init(void)
{
    output_low(LCD_E);          // Let LCD E line low

    lcd_write_cmd(0x38);        // LCD 16x2, 5x7, 8bits data
    delay_ms(15);
    lcd_write_cmd(0x01);        // Clear LCD display
    delay_ms(10);
    lcd_write_cmd(0x0f);        // Open display & current
    delay_ms(10);
    lcd_write_cmd(0x06);        // Window fixed
    delay_ms(10);
}

//end lcd_init()

////////////////////////////////////////// lcd_display_char()
//
void lcd_display_char(int8 line, int8 pos, int8 ch)
{
    int8 tmp;

    line = (line==0) ? 0 : 1;
    pos = (pos > NCHAR_PER_LINE) ? NCHAR_PER_LINE : pos;

    tmp = 0x80 + 0x40*line + pos;
    lcd_write_cmd(tmp);
    lcd_write_dat(ch);
}

//end lcd_display_char()

////////////////////////////////////////// lcd_display_str()
//
void lcd_display_str(int8 line, char str[])
{
    int8 i;

    for(i=0; i<NCHAR_PER_LINE; i++)
    {
        if(str[i] == '\0') break;
        lcd_display_char(line, i, str[i]);
    }
    for( ; i<NCHAR_PER_LINE; i++)
        lcd_display_char(line, i, (char) ' ');
}

//end lcd_display_str()

////////////////////////////////////////// lcd_display_str()
//
void clear_lcd_line(int8 line)
{
    int8 i; char str[]=" ";
    for(i=0; i<NCHAR_PER_LINE; i++)
    {
        if(str[i] == '\0') break;
        lcd_display_char(line, i, str[i]);
    }
}

//end clear_lcd_display_line()

```

APPENDIX G

**INVESTIGATING THE INTERFACE WITH 5DT DATA GLOVE: A
POTENTIAL FUTURE WORK**

G.1 5DT Data Glove

A data glove is a device which detects the motions made by hand and converts it into electrical signals transmitted via a USB interface. The device incorporates 14 sensors mapped in different locations to detect realistic movements made by hand. (see Figure 30).

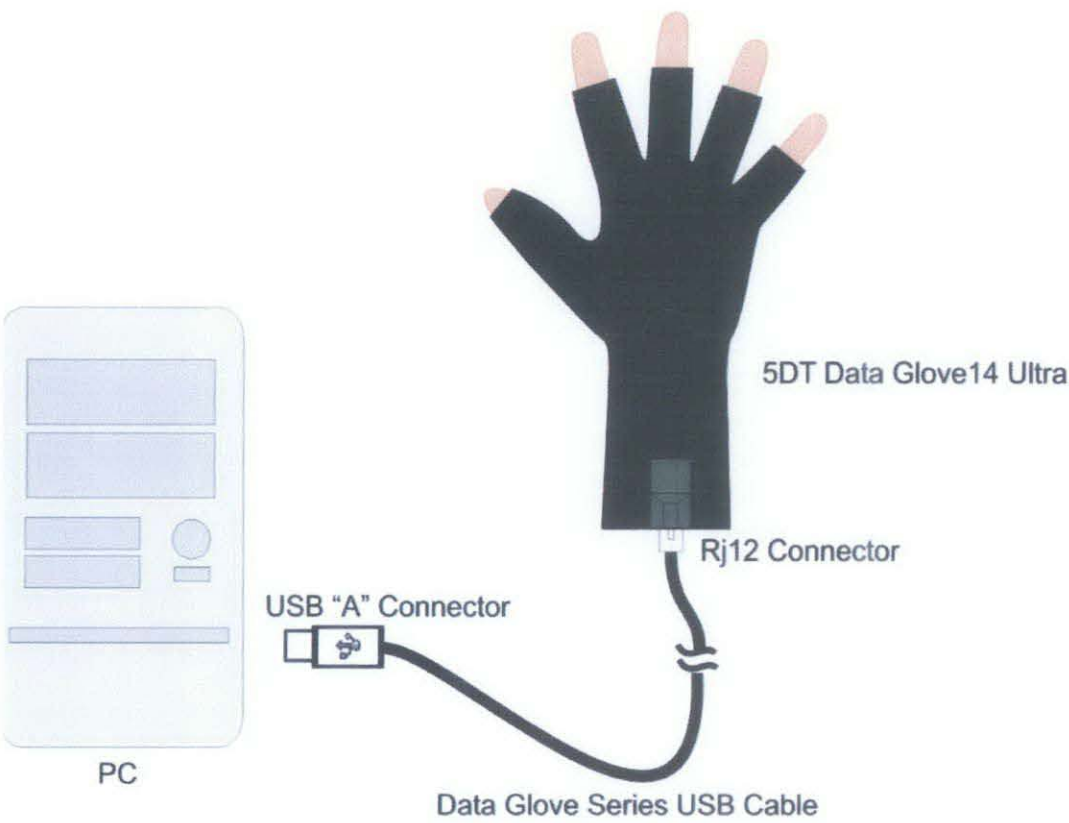


Figure 29: Data Glove with USB connection



Figure 30: Data Gloves Sensor Mapping

Figure 30 and Table 5 illustrate the sensor mapping of the 5DT data glove.

Table 5: Sensor Mappings for the 5DT Data Glove 14 Ultra

| <i>Sensor</i> | <i>Driver Sensor Index</i> | <i>Description</i> |
|---------------|----------------------------|--|
| 0 | 0 | Thumb flexure (lower joint) |
| 1 | 1 | Thumb flexure (second joint) |
| 2 | 2 | Thumb-index finger abduction |
| 3 | 3 | Index finger flexure (at knuckle) |
| 4 | 4 | Index finger flexure (second joint) |
| 5 | 5 | Index-middle finger abduction |
| 6 | 6 | Middle finger flexure (at knuckle) |
| 7 | 7 | Middle finger flexure (second joint) |
| 8 | 8 | Middle-ring finger abduction |
| 9 | 9 | Ring finger flexure (at knuckle) |
| 10 | 10 | Ring finger flexure (second joint) |
| 11 | 11 | Ring-little finger abduction |
| 12 | 12 | Little finger flexure (at knuckle) |
| 13 | 13 | Little finger flexure (second joint) |

G.1.1 Getting started with USB interface

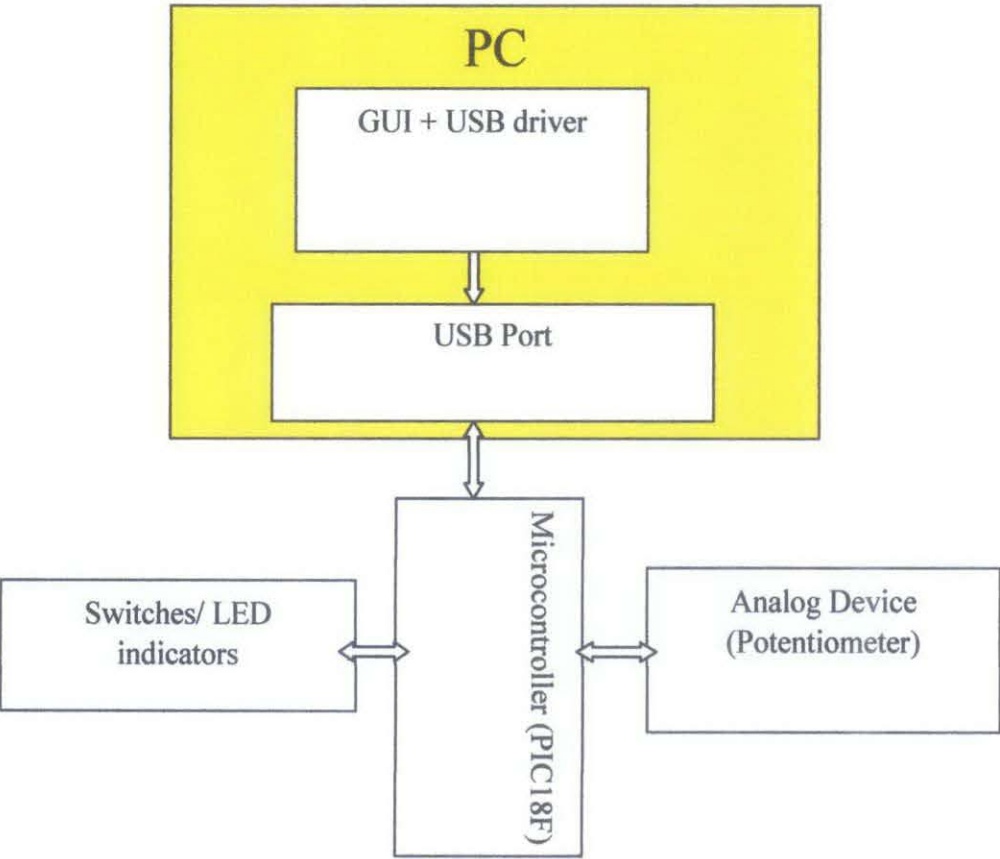


Figure 31: Startup Circuit for USB interface

In order to realize the interface between the microcontroller and the data glove, a prior step which is the familiarization with USB interface is required. Since USB communication is much complex compared to other serial protocols, the aim at this stage is to acquire the basic configuration for the USB interface to work. To get started with the USB interface, a previous project [10] which used the same microcontroller (PIC18F4550) is implemented in this work.

Figure 31 shows the block diagram for the proposed system. The SK40C board has a built-in USB port directly connected to the appropriate pins at the PIC18F4550 microcontroller. A USB cable is to be used to make the interface. However, having the connection while not defining the microcontroller to the PC would not allow any transmission of data from or to the microcontroller.

A visual C# project and dll files are used at the PC side to install required drivers and GUI interface. The microcontroller acts as a slave in this configuration, allowing responses when being acquired by the PC. A simple program reading analog values from AD conversion units and switches and LEDs are used in the program. The source codes are available in [10] and the results are shown in the following sections.

G.1.2 PIC18F4550 as a USB CDC device

USB devices can communicate differently according to the way they are defined to the host. In the previous section the PIC is defined as HID (Human Interface Device); however, it is more relevant to use it as a serial port to eliminate the need of using user defined application (like C# program). Therefore, an example file provided in the CCS compiler titled “ex_USBCDC.c” is used to realize this scenario. The code is used as is, but important modification is to set the right clocking options. This is essential because the USB peripheral and CPU of the PIC use the same oscillator but their clocking requirements are different. The schematic diagram for the clocking circuitry of the CPU and the USB is shown in the following figure.

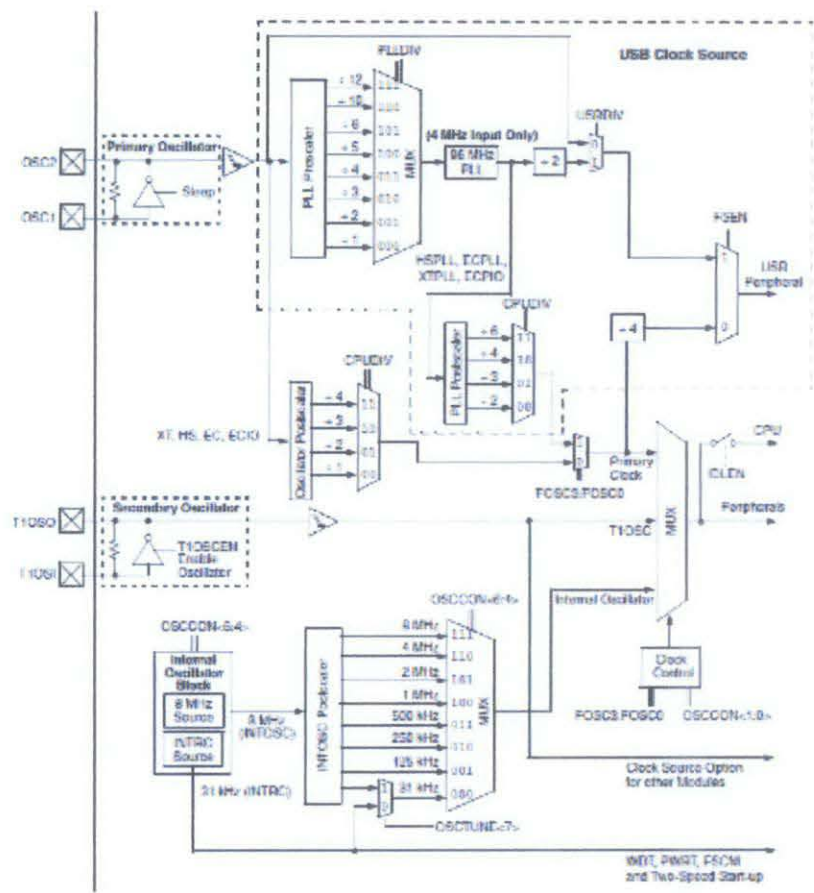


Figure 32: PIC18F4550 oscillator and clock diagram for the CPU and USB peripheral

As far as coding is concerned, some configuration bits have to be set to match the USB requirements. In the following figure, a code snippet describing the setting of some configuration/fuses bits is shown.

```
uses HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV1, VREGEN, NOBROWNOUT, MCLR
The 20 mhz external crystal is pre scaled by div 5 (PLL5) to 4mhz
PLL multiplies by 16 to 96Mhz USBDIV post scales with div by 2 to 48MHZ
CPU clock is post scaled with CPUDIV1 by with div by 2 for 48 MHZ
se delay(clock=48000000)
```

Figure 33: Code snippet for setting USB Clock

The modified codes based on USB CDC example is attached in Appendix I.

G.1.3 PIC and data glove USB interface

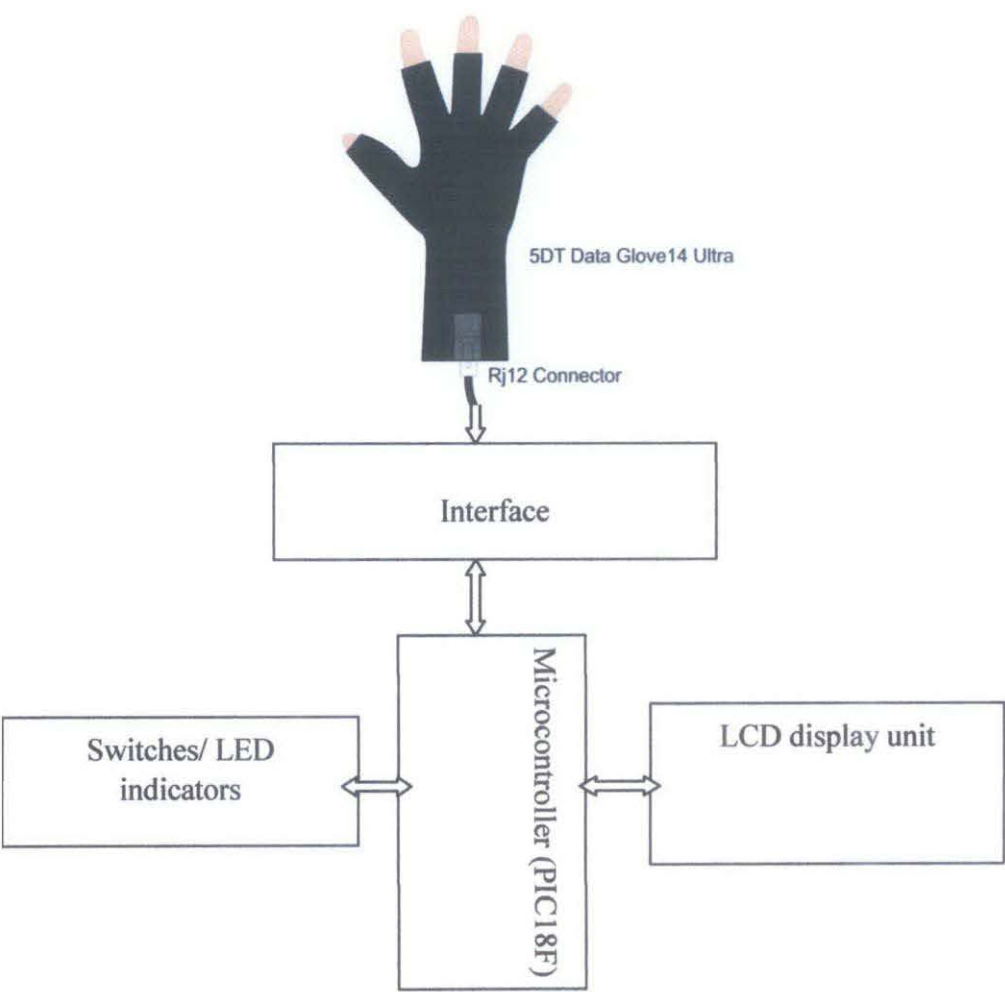


Figure 34: Block Diagram for the Proposed System

When the PIC is configured successfully to utilize the USB peripheral, then the possibility of interfacing the data glove can be verified. In order to do that, the PIC is configured to continuously read data from the attached device and display them on an LCD. This enables the monitoring of data transmission between the two devices. The code used to do that is provided in Appendix J. After loading the code into the PIC, a direct connection using USB cables is used between the PIC and the data glove. The result of the interface is shown and discussed in the next sections.

G.2 Getting Started with USB

In this experiment a USB communication between the PIC and the PC is to be established. This is because the ultimate goal of the work on USB is to establish a communication between the PIC and data glove. Since the data glove is designed to communicate via USB, the ability to use USB by PIC could help in establishing a connection with the data glove. The components used are shown in Figure 35 and Figure 36.



Figure 35: Circuit Components

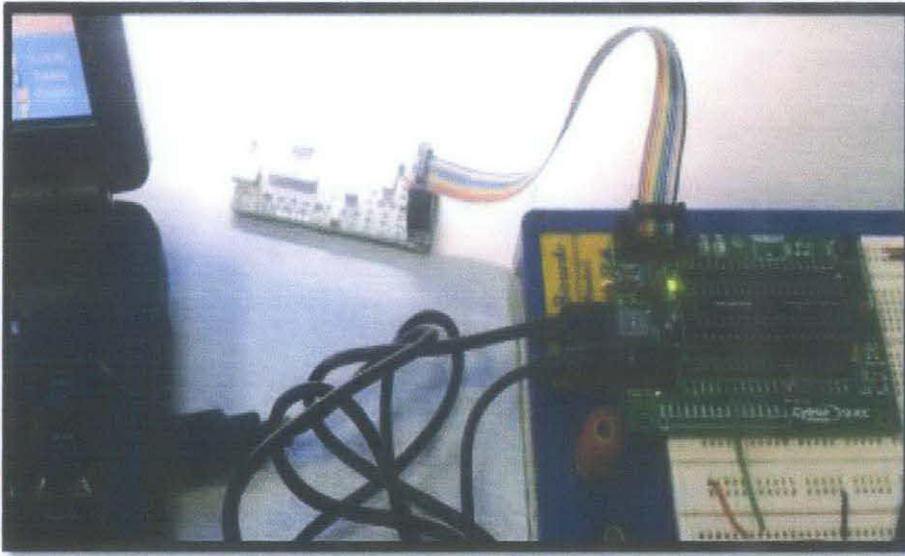


Figure 36: PC to Microcontroller Interface via USB Port

The PIC is programmed to communicate as HID (Human Interface Device) class. A GUI windows application is used to read the ADC values and switch status from the microcontroller and to toggle the status of LED as shown in Figure 37.

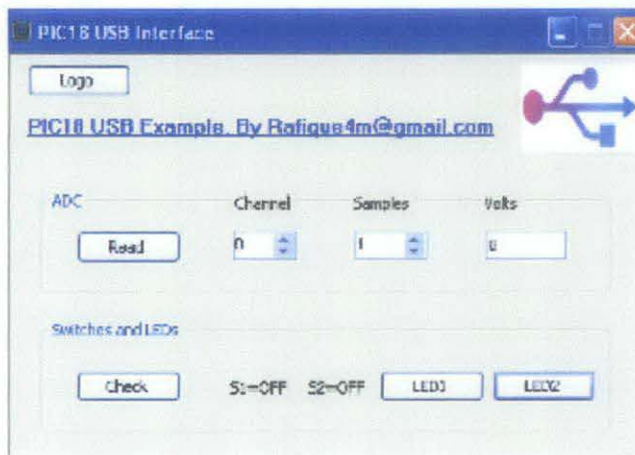


Figure 37: GUI Interface to Read a Value from an Analog Device and Toggle the State of LEDs Attached to Microcontroller

G.3 PIC as Serial Port via USB

The previous experiment requires some installation of USB drivers, whereas in the case of data glove, there is no room for installing any driver, so we need to rely on a supported

configuration without the need for installing additional drivers. In this experiment, the PIC is to be configured to work as a serial port which is more familiar to work which requires the use of hyper-terminal-like programs.

USB CDC example provided by CCS is used to make the configuration. The clocking setting for USB and CPU are made as explained in previous section.

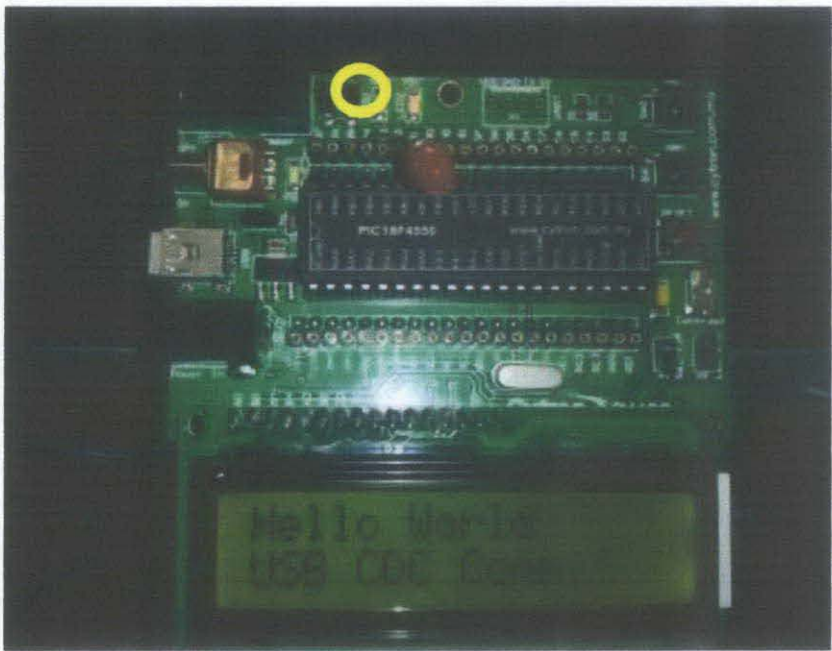


Figure 38: PIC Showing USB Is Successfully Attached (Observe the small LED light indicator)

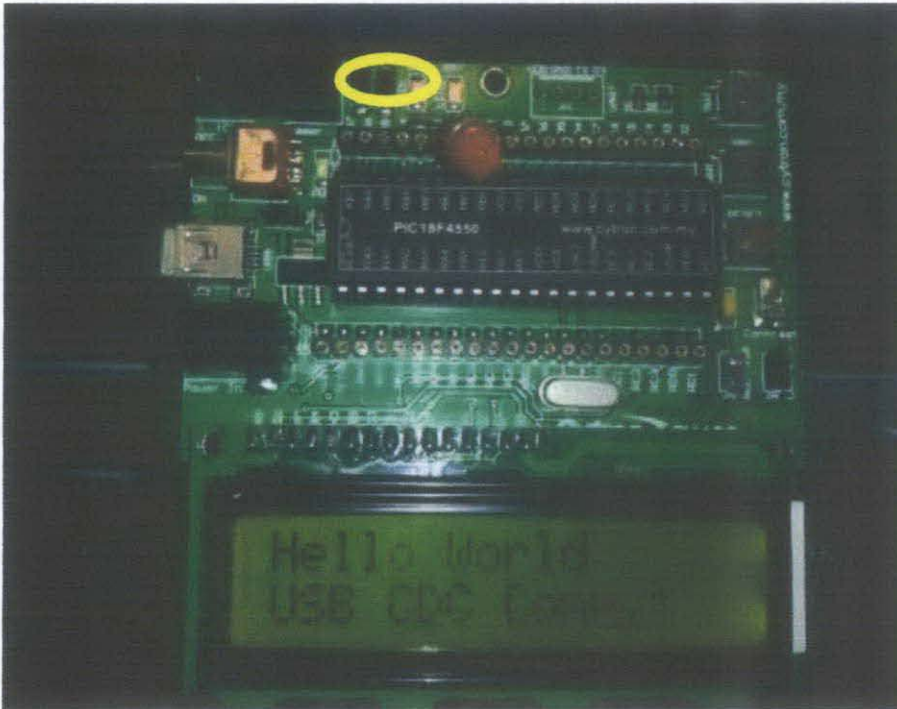


Figure 39: PIC Showing USB Is Successfully Enumerated (Observe the small 2 LED light indicator)

The proper setup was made and the PIC was successfully recognized/attached and enumerated by the PC as shown in Figure 38 and Figure 39 respectively. In order to send and receive data to and from the PC to the PIC, hyper terminal or any similar tool which communicates with serial port can be used; however the right setting has to be made. A window showing the COM port options is shown in Figure 40.

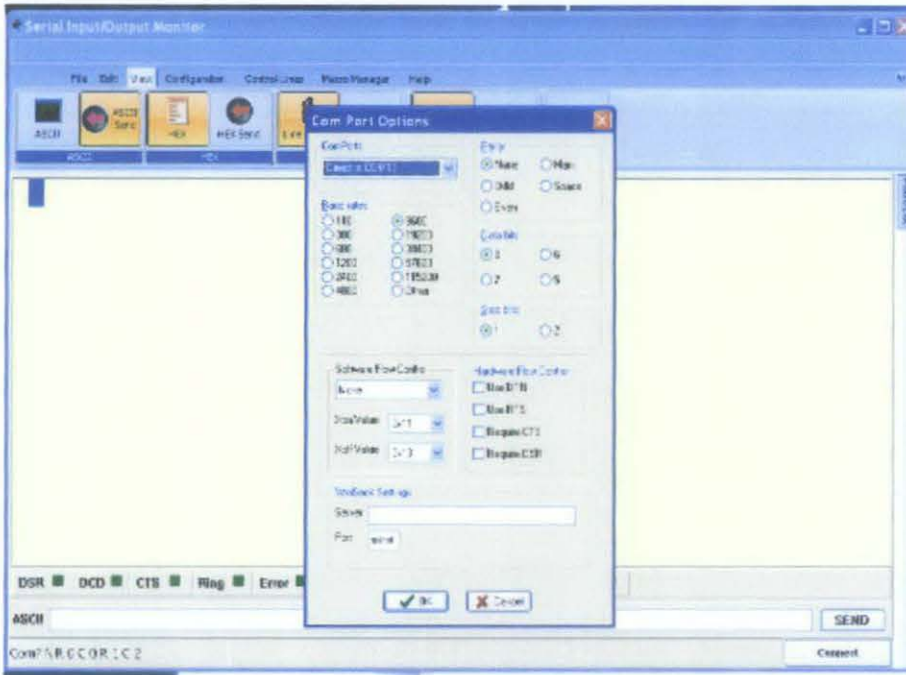


Figure 40: Setting up the Serial Communication to the Microcontroller Using Serial Monitor on CCS C Compiler

After making the setting, the communication is successfully established and the user can send data and receive it on LCD attached to the PIC as the PIC has been programmed to work. Figure 41 shows the working circuitry.

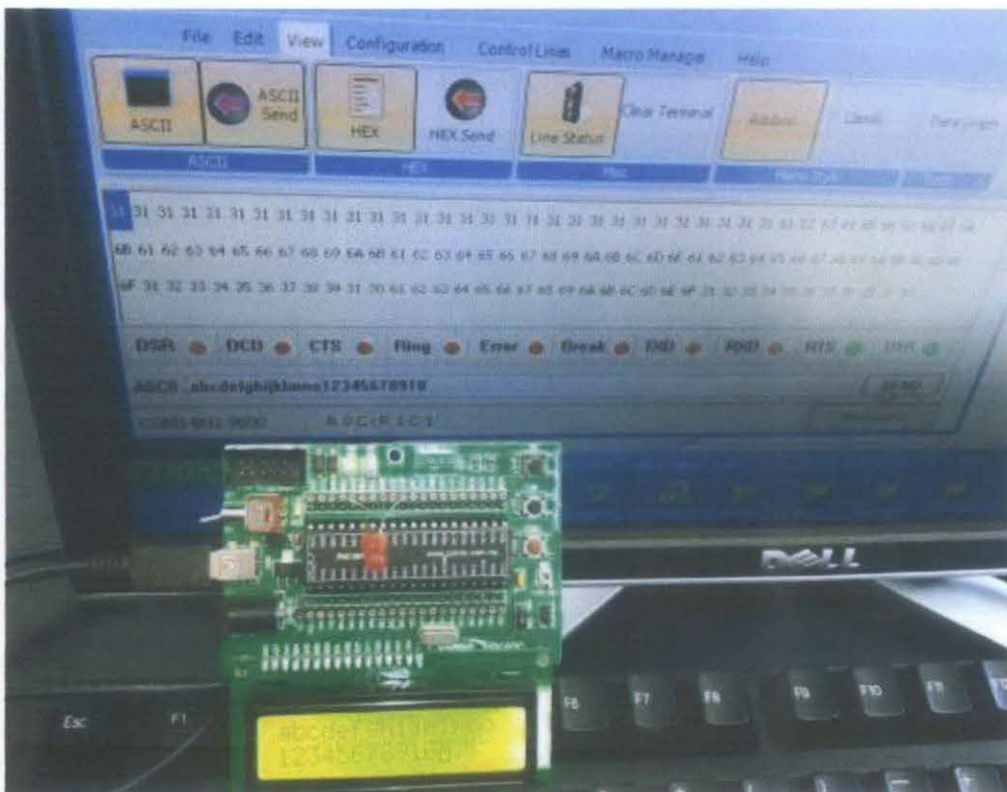


Figure 41: Display of Received Data from PC via USB Connection

As the figure shows, the PIC continuously read data from the USB bus and continuously displaying in on the LCD screen. This setting is made to assist in reading data from the data glove, regardless what it represents.

G.4 USB Interface between PIC and Data Glove

After doing the previous experiment, we feel confident to try the interfacing via USB cable. As PIC board is using mini USB female board and the data glove uses normal male USB cable, a converter cable was acquired and used in the experiment.

The PIC and glove are connected via the converter cable and the circuit is powered and the status of the connection is monitored on the LCD and the LED indicators. As shown in Figure 42 and Figure 43, the LED indicators are OFF which implies that the devices were not able to establish connection.

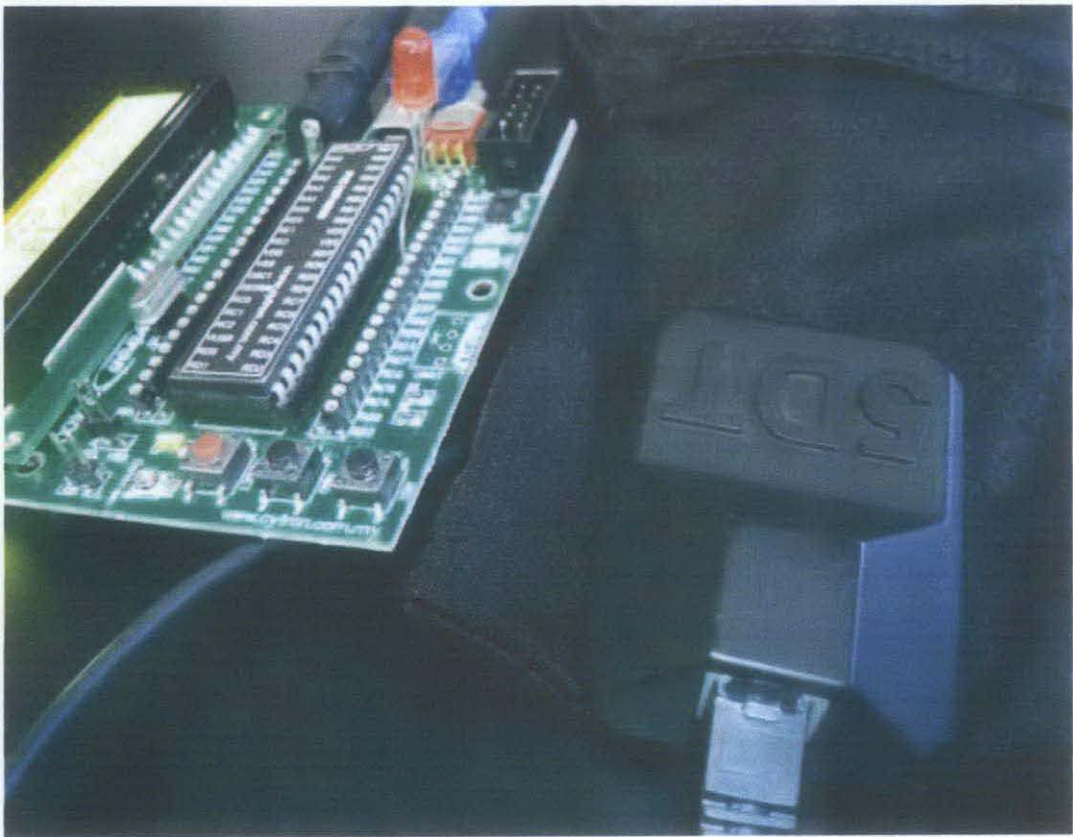


Figure 42: SDT Data Glove Interface with PIC Board



Figure 43: Interfacing the PIC and Data Glove (note all LED indicators are OFF)

G.5 Discussion

As far as interfacing the data glove is concerned, after conducting the pre-mentioned set of experiments, we realized that there is a great distinction between USB as serial communication and conventional RS232 protocol. The concept of slave and master in USB protocol and the required hardware support makes it significantly different than RS232 protocol which does not have this concept.

The USB communication has a special protocol which primarily enables up to 127 devices to be connected to the same bus. However it is important to note that the communication via USB is Host controlled (requires host device not like the normal RS232 protocol) and its topology can typically have: Host, Hub, and device. Therefore, the hardware and software specifications of the Host and device are different. Interestingly, to enable USB devices to communicate to other USB devices the USB On-The-Go devices were developed with some

limitations, but anyway they have different hardware specifications.

The PIC18F4550 is a slave device in a USB protocol, and it is technically impossible to initiate and control data communication to other slave devices like the 5DT data gloves. Additionally, if another microcontroller is used to take the role of a master in the proposed project (PIC24F series), some function and drivers have to be defined to the microcontroller in the form of hex, assembly or C files which is not provided by the 5DT company.

These reasons make it impossible to realize the interface using a USB connection. As a solution, a serial interface kit provided by the same company has to be acquired. The serial interface kit utilizes the conventional RS232 protocol with clearly defined packet data as specified in the data sheet of the product (see Table 6). Figure 44 shows the proposed modification of the system with the addition of the serial interface kit in the design.

Table 6: Data Packet Sent by the Glove

| Byte No. | Byte | Byte No. | Byte |
|----------|-----------|----------|-------------|
| 1 | Start | 17 | 9LI 10HI |
| 2 | Type Byte | 18 | 10Lu 10LI |
| 3 | Version | 19 | 11HI 11Lu |
| 4 | 1HI 1Lh | 20 | 11LI 12HI |
| 5 | 1LI 2HI | 21 | 12Lu 12LI |
| 6 | 2Lu 2LI | 22 | 13HI 13Lu |
| 7 | 3HI 3Lu | 23 | 13LI 14HI |
| 8 | 3LI 4HI | 24 | 14Lu 14LI |
| 9 | 4Lu 4LI | 25 | 15HI 15Lu |
| 10 | 5HI 5Lu | 26 | 15LI 16HI |
| 11 | 5LI 6HI | 27 | 16Lu 16LI |
| 12 | 6Lu 6LI | 28 | Checksum |
| 13 | 7HI 7Lu | 29 | Footer |
| 14 | 7LI 8HI | | |
| 15 | 8Lu 8LI | | |
| 16 | 9HI 9Lu | | |

Interfacing the 5DT data glove with the PIC is essential to realize the sign language trainer. However, the USB communication is not a viable option and requires the use of serial interface kit. Instead, a prototype incorporating five potentiometers – to simulate a data glove, PIC microcontroller and LCD modules is considered for implementation.

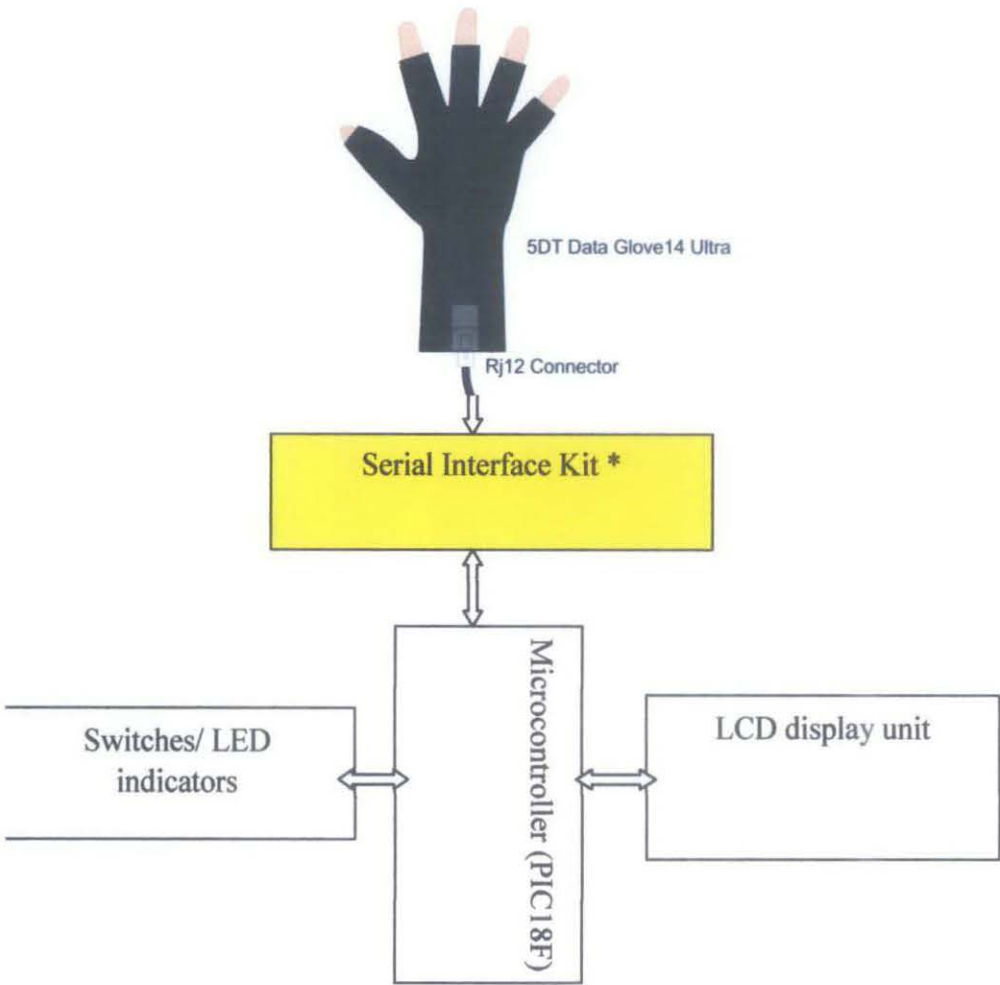


Figure 44: Block Diagram for the System with the Proposed Serial Interface Kit

The implementation of a translation system is applicable to other sign languages as long as the gestures in that language are sensible by the sensing device. The system is also extendable to accommodate the use of data glove as long as the proper interfacing devices are used.

The limitation of the current implementation of the system is that it does not support or make any consideration to the motion of the hand and gesture of the face during conversation. Since most of the signs involve many motions, this will severely limit the performance of the system. However, it is believed that with the proper wearable sensing technology, the motions can be interpreted, modeled and translated using similar concepts described in the project.

APPENDIX H

DATA GLOVE SENSOR MAP



| Gesture Number | Flexure (0=flexed, =unflexed) | | | | Gesture Description | Fig. |
|----------------|-------------------------------|---|---|---|-------------------------------|------|
| | | | | | | |
| 0 | 0 | 0 | 0 | 0 | Fist | 0 |
| 1 | 0 | 0 | 0 | 1 | Index finger point | 1 |
| 2 | 0 | 0 | 1 | 0 | Middle finger point | 2 |
| 3 | 0 | 0 | 1 | 1 | Two finger point | 3 |
| 4 | 0 | 1 | 0 | 0 | Ring finger point | 4 |
| 5 | 0 | 1 | 0 | 1 | Ring index point | 5 |
| 6 | 0 | 1 | 1 | 0 | Ring middle point | 6 |
| 7 | 0 | 1 | 1 | 1 | Three finger point | 7 |
| 8 | 1 | 0 | 0 | 0 | Little finger point | 8 |
| 9 | 1 | 0 | 0 | 1 | Index and little finger point | 9 |
| 10 | 1 | 0 | 1 | 0 | Little middle point | 10 |
| 11 | 1 | 0 | 1 | 1 | Not ring finger point | 11 |
| 12 | 1 | 1 | 0 | 0 | Little ring point | 12 |
| 13 | 1 | 1 | 0 | 1 | Not middle finger point | 13 |
| 14 | 1 | 1 | 1 | 0 | Not index finger point | 14 |
| 15 | 1 | 1 | 1 | 1 | Flat hand | 15 |

APPENDIX I

GETTING STARTED WITH USB (USB CDC CODE)

```
#include "main.h"
#include <usb_cdc.h>
#define LED1 PIN_B7
#define LED2 PIN_B6
#define LED3 PIN_B5
#define LED_ON(x) output_high(x)
#define LED_OFF(x) output_low(x)
#define BUTTON_PRESSED() !input(PIN_B0)
#include "LCD.h"
#include <string.h>
// #define USB_CON_SENSE_PIN PIN_B2

void usb_debug_task(void)
{
    static int8 last_connected;
    static int8 last_enumerated;
    int8 new_connected;
    int8 new_enumerated;
    static int8 last_cdc;
    int8 new_cdc;

    new_connected=usb_attached();
    new_enumerated=usb_enumerated();
    new_cdc=usb_cdc_connected();

    if (new_enumerated)
        LED_ON(LED1);
    else
        LED_OFF(LED1);

    if (new_cdc)
        LED_ON(LED2);
    else
        LED_OFF(LED2);

    if (usb_cdc_carrier.dte_present)
        LED_ON(LED3);
    else
        LED_OFF(LED3);

    if (new_connected && !last_connected)
        printf("USB connected, waiting for enumeration...\r\n\n");
    if (!new_connected && last_connected)
        printf("USB disconnected, waiting for connection...\r\n\n");
    if (new_enumerated && !last_enumerated)
        printf("USB enumerated by PC/HOST\r\n\n");
    if (!new_enumerated && last_enumerated)
        printf("USB unenumerated by PC/HOST, waiting for enumeration...\r\n\n");
    if (new_cdc && !last_cdc)
        printf("Serial program initiated on USB<->UART COM Port\r\n\n");

    last_connected=new_connected;
    last_enumerated=new_enumerated;
    last_cdc=new_cdc;
}

void main(void)
{
    setup_adc_ports(AN0_TO_AN4|VSS_VDD);
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);
    lcd_init();
    char c;
```

```

LED_OFF(LED1);
LED_OFF(LED2);
LED_OFF(LED3);

printf("\r\n\nCCS CDC (Virtual RS232) Example\r\n");

printf("\r\nPCH: v");
printf(__PCH__);
printf("\r\n");

usb_init_cs();
char message1[16], message2[16];
strcpy(message1, "Hello World");
lcd_display_str(0, message1);
strcpy(message2, "Line2");
lcd_display_str(1, message2);

char poll_command[16];

while (TRUE)
{
    usb_task();
    usb_debug_task();

    if (usb_cdc_kbhit())
    {
        c=usb_cdc_getc();
        if (c=='d') printf(usb_cdc_putc, "\r\nportd is a digital output port\r\n");
        else
        if (c=='a') printf(usb_cdc_putc, "\r\nporta is an analog input port\r\n");
        else
        if (c=='b') printf(usb_cdc_putc, "\r\nportb is a digital output port\r\n");
        else
        if (c=='c') printf(usb_cdc_putc, "\r\nportc is unavailable\r\n");
        else
        if (c=='!') printf(usb_cdc_putc, "\r\nHELLO-WORLD-HELLO-WORLD-HELLO-WORLD-HELLO-WORLD-HELLO-WORLD\r\n");
        else
        printf(usb_cdc_putc, c);
    }
}
}

```


APPENDIX J

5DT DATA GLOVE AND PIC INTERFACE VIA USB (CODE)

```
#include "main.h"
#include <usb_cdc.h>
#define LED1 PIN_B7
#define LED2 PIN_B6
#define LED3 PIN_B5
#define LED_ON(x) output_high(x)
#define LED_OFF(x) output_low(x)
#define BUTTON_PRESSED() !input(PIN_B0)
#include "LCD.h"
#include <string.h>

void usb_debug_task(void)
{
    static int8 last_connected;
    static int8 last_enumerated;
    int8 new_connected;
    int8 new_enumerated;
    static int8 last_cdc;
    int8 new_cdc;

    new_connected=usb_attached();
    new_enumerated=usb_enumerated();
    new_cdc=usb_cdc_connected();

    if (new_enumerated)
        LED_ON(LED1);
    else
        LED_OFF(LED1);

    if (new_cdc)
        LED_ON(LED2);
    else
        LED_OFF(LED2);

    if (usb_cdc_carrier.dte_present)
        LED_ON(LED3);
    else
        LED_OFF(LED3);

    if (new_connected && !last_connected)
        printf("USB connected, waiting for enumeration...\r\n\n");
    if (!new_connected && last_connected)
        printf("USB disconnected, waiting for connection...\r\n\n");
    if (new_enumerated && !last_enumerated)
        printf("USB enumerated by PC/HOST\r\n\n");
    if (!new_enumerated && last_enumerated)
        printf("USB unenumerated by PC/HOST, waiting for enumeration...\r\n\n");
    if (new_cdc && !last_cdc)
        printf("Serial program initiated on USB<->UART COM Port\r\n\n");

    last_connected=new_connected;
    last_enumerated=new_enumerated;
    last_cdc=new_cdc;
}

void main(void)
{
    setup_adc_ports(AN0_TO_AN4|VSS_VDD);
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    setup_comparator(NC_NC_NC_NC);
    setup_vref(FALSE);
    lcd_init();
    char c;
```

```

LED_OFF(LED1);
LED_OFF(LED2);
LED_OFF(LED3);

printf("\r\n\nCCS CDC (Virtual RS232) Example\r\n");

printf("\r\nPCH: v");
printf(_PCH_);
printf("\r\n");

usb_init_cs();
char message1[16], message2[16];
strcpy(message1, "Hello World");
lcd_display_str(0, message1 );
strcpy(message2, "USB CDC Comm.!");
lcd_display_str(1, message2 );

unsigned char recieved_packet[29];
int8 i;

for(i=0; i<29; i++)
{
    recieved_packet[i] = '\0';
}

i=0;

while (TRUE)
{
    usb_task();
    usb_debug_task();

    if (usb_cdc_kbhit())
    {
        recieved_packet[i]=usb_cdc_getc();
        lcd_display_char((int8) (i/15), (int8) (i%15), recieved_packet[i]);
        i++;
        if( i>=29) {lcd_display_char( 1,14, ' '); i=0;}
    }
}
}

```

Microcontroller Based Sign Language Translator

Mohammed Obaidallah Alharbi¹, Mohd Zuki Bin Yusoff²

¹Student Support Services

Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia

Email: mohammed_jd_sa@hotmail.com

²Electrical and Electronics Engineering Programme

Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia

Tel: +60-5-3687807, Email: mzuki_yusoff@petronas.com.my

ACT

unities of vocal impaired and deaf people who use uage face great communication difficulties with ho use vocal languages. This project, aims to : towards bringing the gap closer by offering a tool nslates sign languages to written messages on an lay. This report discusses the different development ementation issues including gesture modeling, nterfacing, sign recognition and translation.

Sign Language is widely used in different part of 'including Malaysia; therefore it is considered in ject. The proposed method utilizes five eters to emulate sensor output, a microcontroller e, convert, recognize, translate and display the ure on the LCD unit. The translator can recognize ers, 10 numbers, and some phrases and words. The work is believed to be an entry to more promising rding sign language translation-applications in the

ORDS

uage, ASL, Microcontroller, Gesture detection, y.

DUCTION

nguage is widely used by people who suffer from ipairment or hearing problems in which the ators use visually transmitted signs to convey . The deaf community which utilizes sign language ed to be 0.1% of total population, which means of people worldwide [1]. This large community at difficulties in communicating with normal everal attempts have been made to break this gap sign language users and conventional vocal communicators by introducing tools that can the meaning for both sides. This project aims to prototype which interprets the signal made by a

sign language communicator into a displayed message on LCD. This project is believed to be a base for future work in this area.

Sign language generally utilizes manual movement to convey meanings. This language is not understood by average people. The majority of people understand visually written letters, while sign language users can only use manual signs. In order to break the gap, a set of sensors can be used on the signer to efficiently convert the signs made to electrical signals which in return can be understood by a personal computer (PC) and interpreted accordingly. However, the use of PC does not make the solution mobile and easy to carry. Therefore, a simple IC based circuitry interface (i.e. microcontroller) is required to replace the job of PC. In general, such replacement involves several challenges due to the limited resources which are normally found in conventional ICs (i.e. microcontrollers).

Therefore, the aim of this project is to construct a prototype which interprets basic signs into a readable text on an LCD. In order to realize such prototype, the ASL language is chosen for the implementation and the following objectives are considered: obtaining a numerical representation all gestures used in the sign language, constructing a sign language dictionary, and prototyping a translation system using set of potentiometers, a microcontroller and LCD modules.

The project is envisaged to deliver a prototype which makes use of a set of potentiometers to model the actual sign gestures which can be later replaced to an accurate data glove. A microcontroller with different communication modules is to be used to acquire, manipulate and display the signs being detected by the sensors. A display unit which is as simple as a 2 line=LCD is to be used to display detected messages.

PRELIMINARIES

Sign language which is based on visual manipulation of hands and body is the language of deaf and vocally impaired people. It is interesting to know that sign language is not universal. Despite the fact that most vocabulary and grammar of sign languages worldwide are quite similar, they

typically identical [1]. For example the particular "men" have different sign representation in Auslan, and DSL sign languages [1]. However, studies show that most of world's sign languages have a great identical vocabulary.

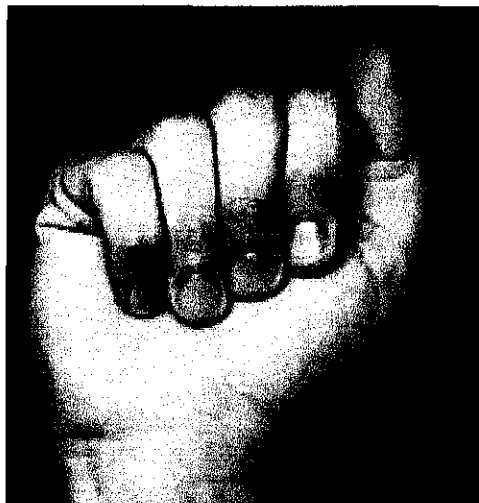


Figure 1: Letter "A" in American Sign Language [2]

On the other side, sign language does not follow the same grammar as for vocal languages [1]. The different vocal language has a significant impact in a particular sign language. This explains the difference in sign languages globally. In this project, we will use the American Sign Language (ASL) [2] as it shares similarity with Malaysian Sign Language (or in Bahasa Isyarat Malaysia : BIM) and is well understood. The letters and the first ten numbers will be highly considered in the proposed system. **Figure 1** is a sample of letter "A" in ASL.

RELATED WORK

Attempts have been made to translate sign language to spoken languages and vice versa. J.M. Allen *et al.* in [3] proposed a system which translates spoken English to sign language. In this work, the authors discussed an algorithm implemented in personal computer which can automate the conversion of spoken and written English language and sign language to the equivalent via an avatar animated sign language.

Ala *et al.* and R. Akmeliawati *et al.* in [4], [5] proposed an algorithm which utilizes neural network to recognize sign from a camera and process it accordingly to produce English translation. This method requires less expensive hardware but more complicated algorithm to process the signs. In order to translate a sign, the image is captured and tracked, then the hand posture is extracted and the corresponding meaning is matched using a learned neural

network. Implementing a recognition system on an ARM processor is discussed in [6]. In this work, the practical aspects of real time blabbering recognition and translation are discussed. The system shows different practical aspects of the implementation of language recognition in embedded systems.

Another interesting work is discussed by R.M. McGuire *et al.* in [7]. In this work, a mobile sign translator based on one hand data glove and a Hidden Markov Model are used. The proposed system shows 94% accuracy for a particular scenario whereby a signer is seeking an apartment.

N. El-Bendary *et al.* attempted to implement arSLAT which recognizes sign representation of Arabic letters and gives the written equivalence [8]. The system processes a video which contains series of image representations for the letters. The best captured image from the video undergoes several phases including categorization, feature extraction and classification before the Arabic letter is finally recognized. Experimental results show 91% of recognition accuracy.

In summary, this short listing for some of the most relevant work all around the world, show the global potentiality of the problem. It also highlights different areas of focus for the implementation of sign language translators. This includes: sensing devices, processing platform (PC, embedded processors, etc.), recognition algorithms, and output forms. In this project, the focus will be in implementing the translation system in microcontroller processing environment.

An essential component of the translation system is the recognition algorithm. Several recognition algorithms have been used in previous studies. In this work fuzzy logic based algorithm is considered for implementation.

Fuzzy logic is a form of many-valued logics; it conceptually deals with reasoning that is approximate rather than fixed and exact [9]. In contrast with the traditional logic theory, where binary variables have two logic values: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth. In partial truth, the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions [9].

METHODOLOGY

1. System Identification and Tools

Throughout the development stage of the prototype of the project, several tools are potentially considered. The tools used for the implementation of the translation system and the respective functional and technical details are discussed as follows.

Potentiometer

A potentiometer is a simple three terminals variable resistor. It provides different values for the resistance across its ends. The central terminal in the middle is connected via a moving wiper to adjust the resistance at this terminal from 0 to full resistance to either ends. The potentiometer is used to produce 0-5 V analog output. It has generally a wide input range of a possible gesture sensor; this allows it to be used as a simplified model for gesture sensor. It can be used to emulate a fingers gesture sensing

Microcontroller

The PIC18F4550 microcontroller from Microchip is to be used. This choice enables the developers to deal with the system with more flexibility and efficiency. The PIC18F4550 [11] is among the most commonly used 8-bit microcontrollers barely because of its USB programming support capabilities. The PIC18F4550 is a high performance microcontroller which is equipped with several built-in peripherals. The proposed system may use the USB support for advanced use, therefore, the system is made to enable future development and expansion of functionality. With the USB support, the microcontroller is featured with different processing modes, configurable internal registers, extendable instruction set which makes it a high speed yet power efficient microcontroller. The 32KB program memory allows long programs (more than 16,000 assembly code lines) to be executed. The data memory for the execution of the program (i.e. variables' data) is in SRAM memory which is 2KB in size for the PIC18F4550 microcontroller. The peripherals of the microcontroller are not used as the proposed system does not require them, it is likely that normal I/O operation are to be used to allow access to other direct digital transmission devices such as LCD.

C compiler

To program the microcontroller, a compiler is to be used. In this project, the PIC C compiler from CCS is to be used.

The PIC C compiler is easy to use, and almost immediate to learn due to the project wizard feature and the different modes which it offers.

- Features of CCS C compiler:
 - Automatic fuses configuration
 - Extensive built-in functions providing direct access to PIC hardware
 - Extensive source code driver library
 - Arithmetic library
 - Integrated development environment

Development kit and programmer

During the development phase, a startup kit [12] is used in the form of the microcontroller circuit. The use of this

tool provides easier and more robust circuit to be built. The board provides several functionalities and circuitry support. The kit is a robust development platform which offers:

- Voltage regulation circuitry (9 V input voltage to 5 V output voltage)
- Reset button
- USB port
- Connector to programmer
- Optional connection to LCD and UART
- 2 switches and 2 LEDs connected to Port B

In order to transfer the C codes to the program memory of the microcontroller, USB ICSP programmer (UIC00B) [13] is considered. This programmer is a cheap programming solution and is highly compatible with the SK40C startup kit.

The programming software (PIC kit 2) takes the hex file which is produced by CCS compiler and loads it to the microcontroller memory via the UIC00B programmer.

2. Sign Language Translation Procedure

The translation procedure involves several issues, tasks and algorithms. The following part discusses these challenges and explains the methodologies adopted in this work. The flow chart of the translation system is shown in Figure 2.

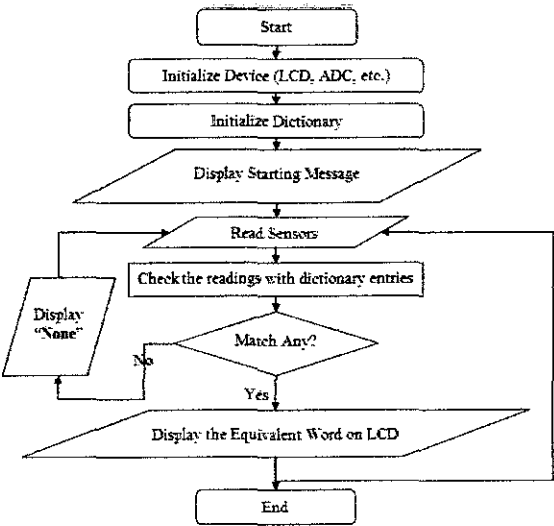


Figure 2: Translation Flow Chart

The translation is basically a closed loop in which the sensors are read, compared and if recognized, the results are displayed on the LCD unit. The reading of the sensors is compared to a dictionary which is created according to the ASL dictionary. The dictionary consists of a set of numerical representation to the gestures of the five fingers (thumb, index, middle, ring and little). The modeling and the numerical representation of the gesture are described in the following part.

Sensor reading and display

step on the hardware development of the system is the interface between the PIC and the sensing unit. the sensing unit is assumed to be simple meters and therefore, the PIC is required to establish communication with the potentiometers. The ADC in the PIC is used to implement this function. Five levels are provided for this purposes which are: A0, A1, A2, A3, A5. The first step in implementation is to set the configuration using the project wizard in CCS C as follows:

1. Selecting PIC18F4550 (as it is the target PIC)
2. Oscillator frequency: 20,000,000 Hz (as used in the development board)

3. For the oscillator fuses, choose the configuration: High speed Osc (> 4mhz, for PCM/PCH) (>10mhz for PCD)

4. Inselect the option : "PORTB pins are configured as analog input channels"

5. For the analog configuration, the following is used:

- A0, A1, A2, A3, A5
- Range 0-Vdd
- Units: 0-255
- Internal 2-6u for the clock

6. The code is generated upon making the above settings. 7. The code is used to display the data.

Signs Gesture Representation

Assume that each letter or word in sign language is composed of gestures made by the five fingers of the hand. To obtain a numerical representation for each letter, 6 gestures per finger are identified to be the basic blocks for each sign as shown in Table 1.

Table 1: Basic Gesture Meaning

| Code | Gesture Description |
|------|--|
| | Inflexed finger |
| | Upper Joint flexed |
| | Middle joint flexed |
| | Middle + Upper joint flexed |
| | Lower joint tilts aside |
| | Upper+ Lower joint tilts aside |
| | Middle+ Lower joint tilts aside |
| | Upper +Middle +Lower joint tilts aside |
| | Lower joint half bent |
| | Upper+ Lower joint half bent |
| | Middle+ Lower joint half bent |
| | Upper+ Middle+ Lower joint half bent |
| | Lower joint fully bent |
| | Upper+ Lower joint fully bent |
| | Middle+ Lower joint fully bent |
| | Upper +Middle +Lower joint fully bent |

For each sign entry in the dictionary, the gesture represented by the sign is identified according to the table above and numerical values are assigned according to Table 2.

Each gesture is assigned to an arbitrary number from (0-255) with a 16 digits step. The gaps between the gestures are later exploited to identify fuzzy limits between the gestures.

Table 2: Numerical Representation of Gestures

| Gesture Code | Numerical Equivalent |
|--------------|----------------------|
| G0000 | 0 |
| G1000 | 16 |
| G0200 | 32 |
| G1200 | 48 |
| G0031 | 64 |
| G1031 | 80 |
| G0231 | 96 |
| G1231 | 112 |
| G0032 | 128 |
| G1032 | 144 |
| G0232 | 160 |
| G1232 | 176 |
| G0033 | 192 |
| G1033 | 208 |
| G0233 | 224 |
| G1233 | 240 |

C. Letter Matching Algorithm

The signals resembling letters and numbers do not have a strict set of Boolean values. This would suggest the use of Fuzzy logic based algorithm. The signs by nature are not exact and identical to all users. When detecting such signs, the detected signal for the same sign but from different users will vary but should still be close. This consequently leads us to select a fuzzy algorithm to store and match the sign language dictionary.

In fuzzy algorithm, the values of its variable are not in simple TRUE (1) and FALSE (0) patterns, however discrete values representing wide range of trueness and falseness ranging from extremely true to extremely false are typically considered.

In the case of sign language translator, the variables are the reading of sensors and the exact number of sensors depends on the type of sensor system. Each sensor is described by 8 bits value ranging from totally flexed to totally inflexed; however for more general case, the size of the variable (number of bits) depends on the accuracy of the sensor.

The reading of the sensor is to be later compared to find the similar letter which the gesture resembles. The letters, on the other hand, are to be represented by a set of values for each sensor. The fuzzy part comes here, whereby; the values representing each letter describe the upper limit and lower limit for each sensor value. This means, a typical letter or word, is represented by several variables describing the upper limit for the sensors and another set of variables describing the lower limit for the sensors. In addition to that another variable is required to store the equivalent word itself.

structure comprising of the lower limit for the sign sensor, the upper limit for the sign from each and the equivalent word represents a single entry in the sign dictionary. A look-up table is then consisting of all data structures holding the entries for and the corresponding meaning. The entries in the table are to be derived empirically.

ified version of the translation system is tested using three words based on the reading of five the code is developed and the results are shown and in next sections.

Dictionary Construction

ned previously, in order to enable the translation, a holding the gesture and the equivalent word has to be created. The construction of the dictionary is shown in Figure 3.

shows the steps adopted in realizing the dictionary. In programming a struct data type is used to represent a dictionary entry. The dictionary is simply an array of struct data types. The maximum number of entries is limited by the size of the data memory (RAM). In the absence of a separate memory chip is required to store larger dictionaries.

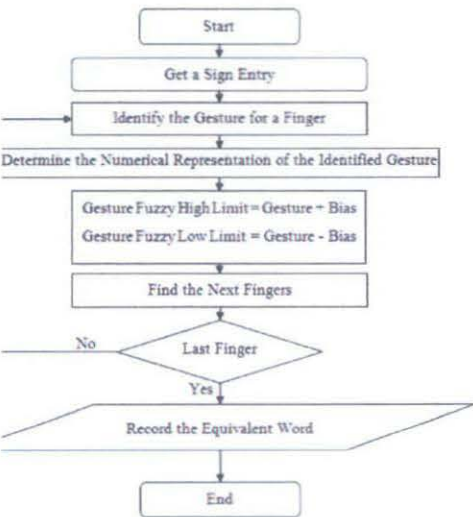


Figure 3: Dictionary Construction Flow Chart, Bias value is 7

Methodology shown in Figure 3 is used to obtain and create the sign dictionary. The sign versions of the letters A to Z and numbers from 1 to 10 and the "I love you" gesture are considered from [2] and consequently the sign dictionary is constructed.

The entries of the table actually show the range of the five sensor values and the equivalent word. As example the sign for the letter 'A' has a value in the range of (G1031L to G1031) for

the thumb sensor, (G0233L to G0233) for the index sensor, (G1031L to G1031) for the middle sensor, (G1233L to G1233) for the ring sensor and (G1233L to G1233) for the little sensor represents the letter A.

RESULTS

The translation system based on a set of potentiometers is constructed. The system is implemented by developing C codes based on the methodology explained previously. The system with the five potentiometers, LCD and mother board is shown in Figure 4. Figure 5 shows the system starting message.

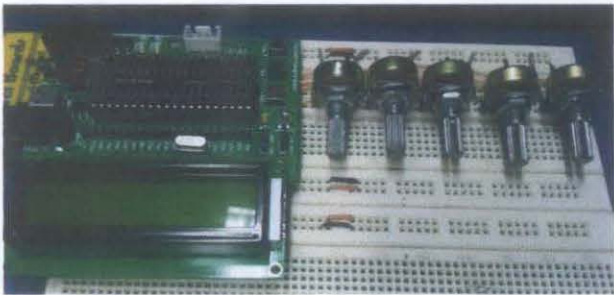


Figure 4: Translation System Components: 5 Potentiometers, Main Board (SK40C board), and LCD

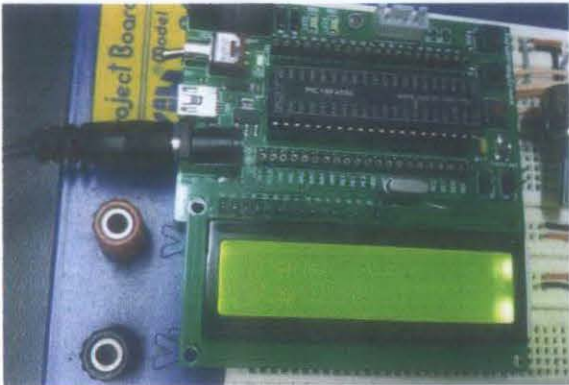


Figure 5: Translation System Startup, LCD is displaying the "Starting" message

The gestures for the letters, numbers and some words were obtained and implemented on the code listing. The reading of the sensors and the recognized gestures are configured to be displayed on the first and second lines of the LCD, respectively. Two modes of sensors reading display were shown on the LCD successfully. The two modes of display are the digitized (0-255) and in Volts (0-5 V) and are shown in Figure 6 and Figure 7, respectively.



Figure 7: Sensor readings in Volts (thumb: 2.4V, middle: 4.6V, ring: 4.7V and little: 4.6V)

Experiments had been conducted to test the ability of the system to recognize gestures. The experimental results show that it is able to recognize the whole 26 alphabetical letters.

Letters are used in coding as well as in display. This is what the LCD displays "B" instead of "b" to indicate the 1st alphabet. All the 26 letters did not involve motion. Since it is assumed that only hand shapes are considered in modeling.

Figure 7 and Figure 8 show some of the recognized

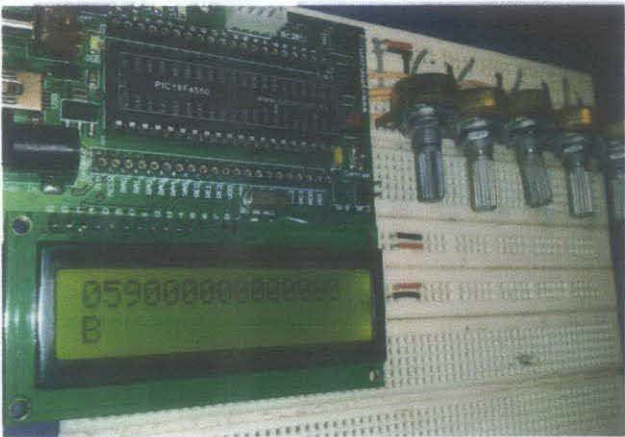


Figure 8: Translation system recognizing the sign for B equivalent to (thumb: 64+/-7, index: 0, middle: 0, ring: 0 and little: 0)

Additionally numbers (0-10) are added and recognized successfully. Moreover, the system is able to recognize some phrases e.g. "I love you". Figure 9 and Figure 10 show some of the obtained results.

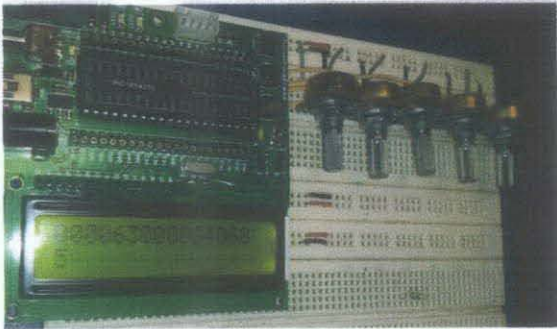


Figure 9: Recognition and translation of the sign for 5 (thumb: 0, index: 64+/-7, middle: 0, ring: 64+/-7, little: 64+/-7)

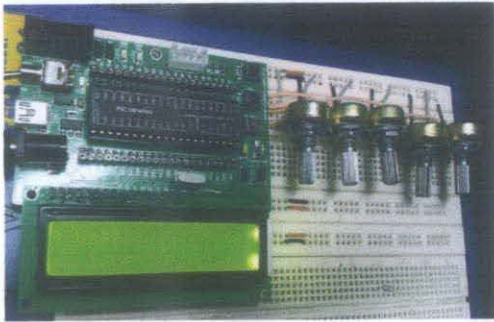


Figure 10: Recognition of phrases e.g. "I Love You" (thumb: 0, index: 0, middle: 224+/-7, ring: 224+/-7, little: 0)

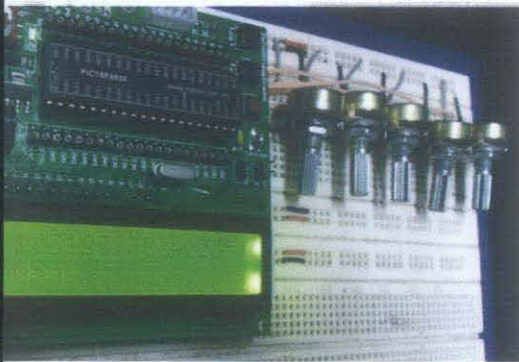


Figure 11: "None" message for any other unrecognized signs

unrecognized signs, the system displays by default "None" message as shown in Figure 11. This is later coded in the coding by the message "Not recognized!".

CONCLUSION

Prototype incorporating five potentiometers – to simulate a tactile sensor reading, PIC microcontroller and LCD display is proposed to aid sign language users to convey messages in a more explicit way. The proposed prototype is based on ASL language and can support up to 75 signs and the equivalent words as a proof-of-concept. The project is envisaged to be an entry work for educational yet practical solutions which can potentially be extended for functionality and portability.

FUTURE DIRECTIONS

Currently the system supports the translation of up to 75 words. The size of the dictionary can be potentially increased considering the addition of memory chip to the system. To enable the portability of the design, a 9v battery pack is to be added. Even though the system is tested without a realistic data glove, it is believed that the systematic methodology adopted in the project will ease the realization of the addition. A potential future work is to replace the five potentiometers by a data glove.

REFERENCES

Frederick Johnston and Adam Schembri, "Australian Sign Language (Auslan): An Introduction to Sign Language Linguistics", Cambridge university press, 2007
American Sign Language (ASL) dictionary, URL: <http://www.lifeprint.com/dictionary.htm>, retrieved: Nov 2011

Allen J.M., Foulds R.A., "An approach to animating sign language: A spoken english to sign english translator

system", Proceedings of the Northeast Conference, 30, pp. 43-44, 2004

[4] Akmeliawati, R.; Ooi, M.P.-L.; Ye Chow Kuang; , "Real-Time Malaysian Sign Language Translation using Colour Segmentation and Neural Network," Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE , vol., no., pp.1-6, 1-3 May 2007

[5] Mekala, P.; Gao, Y.; Fan, J.; Davari, A.; , "Real-time sign language recognition based on neural network architecture," System Theory (SSST), 2011 IEEE 43rd Southeastern Symposium on , vol., no., pp.195-199, 14-16 March 2011

[6] Nijusekar, C.; Brindhu Kumari, A.; , "Translating the sign of dumb person using ARM processor," Communication Control and Computing Technologies (ICCCCT), 2010 IEEE International Conference on , vol., no., pp.508-513, 7-9 Oct. 2010

[7] McGuire, R.M.; Hernandez-Rebollar, J.; Starnes, T.; Henderson, V.; Brashear, H.; Ross, D.S.; , "Towards a one-way American sign language translator," Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on , vol., no., pp. 620- 625, 17-19 May 2004

[8] El-Bendary, N.; Zawbaa, H.M.; Daoud, M.S.; Hassanien, A.E.; Nakamatsu, K.; , "ArSLAT: Arabic Sign Language Alphabets Translator," Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on , vol., no., pp.590-595, 8-10 Oct. 2010

[9] Fuzzy Logic, Wikipedia, URL: http://en.wikipedia.org/wiki/Fuzzy_logic, retrieved: Nov 2011

[10] USB HID PC and PIC interface implementation code using C# and CCS compiler, Muhammad Rafique, URL: <http://www.pudn.com/downloads195/doc/project/detail916558.html>, retrieved: Nov 2011

[11] Microchip Technology Inc., "PIC18F2455/2550/4455/4550 Data Sheet 28/40-Pin High-Performance: Enhanced Flash, USB Microcontrollers with nanoWatt Technology", U.S.A, 2006

[12] Cytron Technologies, "SK40C PIC microcontroller start-up kit: User's Manual", Malaysia, November 2011.

[13] Cytron Technologies, "UIC00B USB ICSP PIC Programmer: User's Manual", Malaysia, November 2011.