

**Integrated Process Control Automation Laboratory with Application of Internet
Based Distance Learning**

by

Lim Tsae Yng

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

June 2009

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL


**Integrated Process Control Automation Laboratory with Application of Internet
Based Distance Learning**

by

Lim Tsae Yng

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,



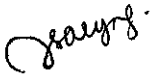
(Dr. Rosdiazli Ibrahim)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2009

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 have not been undertaken or done by unspecified sources or persons.



LIM TSAE YNG

ABSTRACT

This report presents the development of Final Year Project II titled Integrated Process Control Automation Laboratory with Application of Internet Based Distance Learning. The main objective of this project is to create a system that able to integrate the process control laboratory with internet for distance learning purposes. Considering the fact that cost, time and space are the 3 main issues faced by educators. Thus, a comprehensive web-based learning system is essential to provide students with adequate laboratory experience that will better prepare them for a corporate world where the need for engineers in the quality, service, and information technology industries. The web-based learning system must have the ability to control and monitor the conveyor system in the laboratory which can easily perform anytime anywhere. This report will explain how the web-based learning benefits the educators and students. Besides, the basic concept of computer networking has been briefly explained as well. The project work had been further broken down into 4 main parts. Communicating the system by using computer via RS232 serial port, controlling the conveyer by using OMRON PLC CQM1H, which the users can control the movement of the conveyor, sensing the presence of the object on the belt, design a page for students to access this system and lastly testing out the system and document a simple manual for maintenance and enhancement purposes. A project Gantt chart is attached at the Appendix I to illustrate the work flow and anticipated progress. The main page of user interface has been designed. The communication between PC and PLC has been created and the interface is designed. However, the functions of these interfaces will be added from time to time for users' benefit. Besides, the flow chart and source code are attached in the Appendix II and Appendix III.

ACKNOWLEDGEMENTS

First and foremost, the author would like to express her greatest gratitude to the project supervisor, Dr. Rosdiazli Ibrahim for his supervision, guidance and support in this project. His valuable assistant has enabled the author in completing the project successfully.

The author would like to express her greatest appreciation to Mr. Azhar Zainal Abidin for his assistance in understanding the basic concepts of Programmable Logic Controller (PLC).

Lastly, the author would like to thank my parents and friends who have been helping and motivating the author throughout the project.

TABLE OF CONTENTS

CERTIFICATION OF APPROVAL	i
CERTIFICATION OF ORIGINALITY	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	viii
LIST OF TABLES	ix
ABBREVIATIONS	x
CHAPTER 1 INTRODUCTION	1
1.1 Background of Study	1
1.2 Problem Statement	3
1.3 Objectives	5
1.4 Scope of Study	5
CHAPTER 2 LITERATURE REVIEW & THEORY	7
2.1 Application of Web-based Learning	7
2.2 Advantages Web-Based Learning	8
2.3 Disadvantages of Web-Based Learning	9
2.4 Basic Concepts of Computer Networking	9
2.4.1 Network	9
2.4.2 The Internet	10
2.4.3 Client-server Model	10
2.4.4 Protocol	11
2.4.5 Protocol Architecture	11

2.4.6	TCP/IP	11
2.4.7	Sockets	13
2.4.8	Bit Rate and Baud Rate	13
2.5	Programmable Logic Controller (PLC)	13
2.5.1	History of PLC.....	13
2.5.2	Introduction to PLC	14
2.5.3	Structure of PLC	15
2.5.4	Theory of Operation of PLC.....	18
CHAPTER 3 METHODOLOGY		20
3.1	Research Methodology.....	20
3.2	Projects Activities	21
3.2.1	Hardware Connection and Software Installation.....	21
3.2.2	Design the System.....	23
3.2.3	Upload the System Online.....	23
3.2.4	Test the System.....	23
3.3	Tool and Component.....	24
CHAPTER 4 RESULTS AND DISCUSSION.....		27
4.1	Ladder diagram	27
4.2	Comment for the Symbol and Address in PLC	28
4.3	Result and Discussion on TCP/IP	28
4.4	Result and Discussion on PC-PLC Communication.....	32
4.4.1	Serial Communication.....	32
4.4.2	PLC	33
4.4.3	PLC Control.....	33
4.4.4	PLC Status	34
CHAPTER 5 CONCLUSION AND RECOMMENDATIONS		35
5.1	Conclusion.....	35

5.2 Recommendations.....	36
REFERENCES	37
APPENDICES	45
APPENDIX I	46
GANTT CHART OF FINAL YEAR PROJECT II	46
APPENDIX II	48
Source Code for PC-PLC interface	48
Source code for TCP/IP.....	56
Library of TCP/IP Code	57
Source code for Socket Server.....	59
Source code for Socket Clients	67
APPENDIX III	74
Flow Chart for Ladder Diagram	74

LIST OF FIGURES

Figure 1: The concept of client-server model.....	10
Figure 2: A Comparison of the OSI and TCP/IP Protocol Architecture.....	12
Figure 3: Structure of PLC	16
Figure 4 : General Rule of PLC Programming.....	17
Figure 5: Scanning Process.....	18
Figure 6: Flow chart of research methodology.....	20
Figure 7: The plan of the conveyer in the laboratory.....	21
Figure 8: Circuit diagram of conveyor belt system.....	22
Figure 9: Conveyer system.....	24
Figure 10: Programmable Controller Training Kit.....	25
Figure 11: RS232 cable.....	25
Figure 12: RJ45 cable.....	25
Figure 13: Microsoft Visual Studio 2008.....	26
Figure 14: Virtual lab system architecture	26
Figure 15: Ladder diagram for the motion of the conveyor	28
Figure 16: Output of the ladder diagram	28
Figure 17: Interface of TCP/IP	29
Figure 18: The target IP address is typed in	29
Figure 19: The interface of the sender	30
Figure 20: The interface of the receiver	30
Figure 21 : Interface for Socket Server	31
Figure 22 : Interface for Socket Client.....	31
Figure 23 : Interface for PC-PLC.....	32

LIST OF TABLES

Table 1: Hardware and software	24
Table 2 : Comment for symbol and address in PLC	28

ABBREVIATIONS

2-D	2-dimensions
COM	Common
CPU	Central Processing Unit
DoD	Department of Defense
FWD	Forward
I/O	Input/Output
IP	Internet Protocol
LAN	Local Area Network
MAP	Manufacturing Automation Protocol
MIT	Massachusetts Institute of Technology
MODICON	Modular Digital Controller
NC	Normally Closed
NIC	Network Interface Card
NO	Normally Open
OSI	Open Systems Interface
PB	Push Button
PC	Personal Computer
PEC	Programmable Electronic Controller
PLC	Programmable Logic Controller
REV	Reverse
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
WAN	Wide Area Network

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Nowadays, many of us use internet as a source of information. The reasons of using internet are it can be accessed anytime and anywhere as long as there is an internet connection. Therefore, WIFI, Streamyx, Broadband and other internet connections have been introduced. The rapid spreading of broadband internet access has enabled a new delivery method for modern engineering application which is the remote laboratory. As the name suggests, remote laboratory allows the user to control the experiment and obtain the data outputs in an integrated browser-based user interface [1]. In more precise way, remote laboratory let the students to access automatic measuring setup and instruments via geographical network and directly carry out real experiments [38]. This concept allows measuring resources located at different geographical remote sites to be used by the students all around the world [38]. In this way, a more complete and economical education proposal is offered for educator globally [38]. In modern control concepts, blended learning is to encourage the simultaneous use of onsite teaching with online activities [45].

In engineering based education, laboratory experiences are very essential in providing the students with an adequate laboratory experience that will better prepare students for a corporate world where the need for engineers in the quality, service, and information technology industries is increasing [2]. Moreover, interactive experimentation to real world plants improves the motivation of the students as well as develops an engineering approach to solve realistic problems [11]. Besides, working in real

laboratories has become more and more expensive due to high price, involves supervision staff, required scheduling to fit in students' tight schedule, and space restriction [46].

Previously, simulation is one of the ways to complement control education. However, it cannot replace experiments on real plants because it is only a good model and experiment makes the user aware of phenomena that are hard to simulate [11]. By using this system, students can quickly use an experiment setup for a specific application problem and control the conveyer remotely through internet. In the remote delivery mode, students located at off site can control the equipment and collect data using a web-based interface [3]. Besides, unique and expensive equipment can be shared between different universities [11]. The purpose is to maximize their learning experience.

There are several university at other country apply this method in teaching and learning process and yielded positive testimonial from the students. For example, iLab experiment under the Massachusetts Institute of Technology (MIT) initiate involves a heat exchanger laboratory designed to demonstrate the principles of heat transfer [17]. In Chemical Engineering Department at Cambridge University, the remote laboratory of process control experiment has successfully sharing the experiment resources among universities [18]. In an assessment by Ogot et al, the paper has concluded that there is no significant difference between the educational outcomes from students who perform the experiment remotely, compared to those who carried out the experiment in person [49]. By using this approach, a flexible platform is provided. However, it is not imitate typical real work installations, such as networks with specific vendor equipment [48].

Programmable Logic Controller (PLC) is defined as the most inventive device which created to advance the field of manufacturing automation [34]. The world market demand of PLC will continue to grow. The unit size of PLC getting smaller with more function and able to work in tough environment [35]. The sales of PLC are about \$1 billion per year and there are more than 30 manufacturers around the world [36]. This has showed that the great need for engineers with very strong knowledge and skill in this area [37].

1.2 Problem Statement

Laboratory experience is very important to prepare an engineering student to industrial state. However, this concept sometimes has been neglected due to three main issues which are space, cost as well as time. Due to limited space of university, the bulky machine such as conveyer might not be able to fit into the lab. Secondly, many university or colleges might not agree to pay for the costly machine just for experiment purpose. Last but not least, the limited time issue has to be taken into consideration. Therefore, the possibility of lab session or experimental training for the student is reduced [21]. There are more and more educators recognize the importance of an educational experience including both theory and experience [20]. Thus, a lot of simulation tools are introduced to integrate traditional classroom lectures with experiment practice [21].

Although simulation is a cost-effective and safe way for the students to perform the experiment, the importance of physical realities is failed to include [21]. “There will also be an important place for simulation systems, but they cannot completely substitute for experience with actual system” [22]. Simulated environments are restricted as they generally do not imitate the real environment, require considerable effort to maintain up to date environment as well as fail to provide accurate systems perspectives [47]. Traditionally, students attend practical in university based laboratories at fixed time during the academic year [12]. During every lab session, the lab technician has to explain to the students about the procedures and details of the experiments. Due to a number of sessions for the same lab, this explanation will be repeated during each session which consume a lot of time and reduce the productivity of the lab technicians. On the other hand, looking back to the student time table, the slots for lab session might clash with other lectures and therefore more slot of lab sessions have to be planned. Therefore, this has restricts access to laboratory resources to normal working hours which meet the needs of students that is requiring more flexible attendance in line with their current lifestyle commitments [13]. In short, the development of remote laboratory is motivated by the factors of time saving, resources sharing especially costly equipment and last but not least individual access to experimentation [18]. In practice, the engineering laboratory has limitation. The schedule of laboratory has increase tremendously due to large number of

students and each of the session requires at least 2 hours continuous and adequately supervised access to the expensive laboratory equipment [19]. Traditional mode of delivery requires large amounts of resources for a high quality student experience since students must be supervised and equipment is expensive to be purchased and maintained [3]. Therefore, in electrical and electronic teaching and research projects are mostly oriented to supply theoretical educational background such as example of images or signals, hypertexts, exercises virtual instrumentations as well as simulation [41].

The disadvantages of simulation laboratory or virtual laboratory environment have led to the concept of remote laboratory of distance learning that user can accessed the lab remotely [23]. Most of the problems arise from laboratory and experiment teaching although there has been an increase in number of students as well as in instrument cost and complexity due to decrease in budget for laboratory technicians and equipment [38]. This has suggested the idea of activating laboratories remotely accessible via Internet [42, 43, 44].

PLCs are often offered as a subject that will be covered in undergraduate automation and control-related courses. However, most of the educational institutions lack of resources to help students to be proficient PLC users because of high faculty-to-student-ratios, limited access to laboratory, as well as limited equipment to support laboratory assignments [37]. In teaching electrical and electronics engineering, the problems such as limited time, teaching students with reduced means, physical disability, and broad distance are going to be solved in more effective way via using web-based learning [39, 40].

1.3 Objectives

The objectives of the project are:

- To optimize the usage of automation lab with minimal cost
- To let students access to automation lab via internet anytime and anywhere
- To improve the efficiency of control learning and industrial control practice
- To increase the students awareness about the importance of laboratory experiences

1.4 Scope of Study

The scope of this project is to control the conveyer by using CQM1H-CPU21 programmable Logic Controller (PLC). CX-Programmer is the software that will be using to develop the ladder diagram. Then, an interface will be created between the conveyer and personal computer by Visual C#. Therefore, the basic theory of C# language is very important especially to a beginner. Programmer must familiar with function of each symbol in the toolbox so that can write the code smoothly.

Before writing the code for any program, flow chart is very important in guiding the programmer. After that, the programmer can start coding with refer to the flow chart created. The purpose of flow chart is to ensure that the programmer program according to the function. Therefore, flow chart is drawn by using Microsoft Visio Studio.

The development of this remote laboratory involves developing the server application and client application. TCP/IP is the protocol that used in this project. Before writing the code for TCP/IP, the basic knowledge of socket programming and the basic theory of network can be obtained from reference book and reliable web site.

Lastly, a system that controls the conveyer via internet will be constructed. Thus, the scope of study for this project will be on interfacing both hardware (conveyer) and software (CX-Programmer). Besides, the encountered difficulties such as internet time delay and multiple users' collaboration will be studied as well in order to enhance the system's functionality.

CHAPTER 2

LITERATURE REVIEW & THEORY

2.1 Application of Web-based Learning

Web-based learning does not a new technology. There are many virtual labs, e-learning, e-conference and etc. have been introduced in many fields depending on the main functionality and research focus [6]. At the University of Catania, they have developed a Virtual Laboratory for distance learning which is realized in Java and uses the internet to provide complete access to the lab resources [7]. In China, the design of a low-cost Internet-based teleoperation system using a multimedia-rich human-computer interface is implemented [8]. Education fields are strongly influenced by the innovation of remote laboratory [38]. Besides, specific research project are in progress as well [38].

There are several areas for remote laboratories such as “Shared” remote laboratory, “Localized” remote laboratory, “Distant remote laboratory” and “Technical Review laboratory”.

- **“Shared” remote laboratory**

Normally, the laboratory equipment is very expensive and not many education institutions afford to have it. If this laboratory equipment is available at other university which is a distance away, remote laboratory may be one of the solutions in order to access that equipment [24, 25].

- **“Localized” remote laboratory**

This idea is let student to access the laboratory over the internet in relax and time-free environment. It may allow the students to repeat a laboratory session that has been carried out earlier [26]. However, this method is limited to the Local Area Network in that particular university only.

- **“Distant” remote laboratory**

Remote laboratories can fulfill the requirement of practical sessions or experiment session in distance education which are getting popular nowadays [23, 27].

- **“Technical review” laboratory**

This laboratory is to allow industrial professionals to test and evaluate new instrument and device from other company. This is to replace the traditional workshop of evaluating a new instrument [26]. This method will help the company to save travelling cost because the company can test and evaluate the instrument online.

2.2 Advantages Web-Based Learning

Comparing to the traditional method, web-based learning offers a few advantages:

- It is easy to access to the lab and to understand the underlying concept. It requires no distribution of physical material. For example, user can reduce the usage of paper since the lab manual is no longer required to be printed out. This will not only help them to understand the concept well but also make the environment greener.

- It can be accessed anytime and anywhere around the world [10]. Students and researcher can access the lab from hostel, in the office or other places as long as the network connection is available.
- The costs are affordable for everyone [10]. There are not many institutions which can afford to buy the training kit for their students or researchers. If they can use the equipment online, they just have to pay for the internet connection which is affordable for everyone.

2.3 Disadvantages of Web-Based Learning

However, there are several limitations of web-based learning. Multi-user access and conflict resolution is one of the main issues of web-based learning. Besides, shortage of Web-control devices, Internet time delay and web related safety is the limitations of web-based learning as well [4]. Moreover, the computer of each user must be installed with suitable software [5] in order to view the pages, photos, video clips and etc.

2.4 Basic Concepts of Computer Networking

Before embarking to implementation or coding of the web-based learning system, there are several concepts and terminology that must be understood. These includes, network, internet, server-client model, protocol and protocol architecture.

2.4.1 Network

Network is a group of connected collection of devices such as computers and printers or systems that can communicate with each other [14]. The purpose of networking is for resource sharing such as share files, printers, network storage and backup device.

There are several types of network, such as Wide Area Network (WAN), Local Area Network (LAN), wireless network and etc [15].

2.4.2 *The Internet*

Internet is a collaboration of more than hundreds of thousand connected networks [14]. The purpose of internet is to interconnect end systems, called host. Host includes PCs, workstations, servers, mainframes and so on. It is built on the foundation of the Transmission Control Protocol / Internet Protocol (TCP/IP) suite.

2.4.3 *Client-server Model*

The client-server model as shown in Figure 1 describes the relationship between two computer programs which are the clients and the servers.

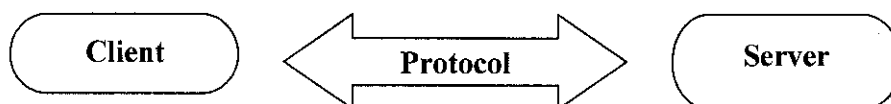


Figure 1: The concept of client-server model

A server is a program running on the remote machine providing service to the clients. When it starts, it runs infinitely unless a problem occurs. A client is a program running on the local machine requesting service from a server. It is started by the user and ended terminates when the service is complete. A client opens the communication channel by using IP address of the remote host or port address of the specific server program running on that machine [14].

2.4.4 Protocol

In computer networks, communications occurs between entities in different system. A protocol is a set of rules that governs data communication. A protocol defines what is communicated, how it is communicated and when it is communicated [14]. The key elements of protocol include syntax, semantics and timing. Syntax is the structure or format of the data and the order they are presented. Semantics is to control information for coordination and error handling. Timing includes speed matching and sequencing. It refers to two characteristics which are when data should be sent and how fast it can be sent [15].

2.4.5 Protocol Architecture

When computer, terminals or other data processing devices exchange data, the procedures involved is quite complex. Instead of implementing it as a single module, the task is broken up into subtasks and each of it is implementing separately. In protocol architecture, the module are arrange in vertical stacks. Each layer in a stack performs a related subset of the functions required to communicate with another system. it relies on the next lower layer to perform more primitive functions and to conceal the details of those functions. It provides services to the next higher layer. When there is some changes in one layer, it do not required changes in other layers [15].

2.4.6 TCP/IP

TCP/IP protocol suite was developed before the Open System Interface (OSI) model was published. As a result, it does not use the OSI model as a reference. TCP/IP was developed using the Department of Defense (DoD) reference model. Unlike OSI, DoD has 5 layers only. They are application, transport, internet, network interface and physical layer [16]. The physical layer covers the physical interface between the data transmission device and a transmission medium or network. The network access layer is concerned with the exchange of data between an end system and the network to which it is attached. Internet layer is to exchange data between two devices that are attached to two

different networks and procedures are needed to allow data to traverse multiple interconnected networks. Transport layer is to assure that all of the data arrive at the destination application and that the data arrive in same order in which they were sent. Lastly is application layer. Application layer is used to support the various user applications. The comparison between OSI model and TCP/IP protocol architecture is as shown in Figure 2.

OSI	TCP/IP
Application	Application
Presentation	
Session	
Transport	Transport
Network	Internet
Data Link	Network Access
Physical	Physical

Figure 2: A Comparison of the OSI and TCP/IP Protocol Architecture

2.4.7 Sockets

Internetworking Protocol (IP) address is the transmission mechanism used by the TCP/IP protocols. It provides no error checking or tracking. Transmission Control Protocol (TCP) provides full transport layer services to application. It is a reliable stream transport protocol. User Datagram Protocol (UDP) is the simpler of the two standard TCP/IP transport protocols. It is a process-to-process protocol that adds only port addresses, checks sum errors control and length information to the data from the upper layer. Port number is the second identifiers after IP address. In TCP/IP protocol suite, the port numbers are integers between 0 and 65,535 [14]. A socket combines three pieces of information which are IP address, TCP or UDP and port number [16]. Besides, it is an abstraction that represents an endpoint of communication.

2.4.8 Bit Rate and Baud Rate

Bit rate or data rate is the rate at which data can be communicated. The unit of bit rate is bits per second (bps) [15]. Baud rate or modulation rate is the rate for at which signal elements are generated [15]. Baud rate is used to determine the bandwidth required to send the signal [15]. For analogy, if 'bit' is a passenger, 'baud' will be the car that fit the passenger while bandwidth is the highway.

2.5 Programmable Logic Controller (PLC)

2.5.1 History of PLC

In the late of 1960's, PLCs (Programmable Logic Controller) were introduced. It was introduced in order to eliminate large cost involved in replacing the complicated relay based on machine control systems [28]. The reason of introducing PLC is because in the 1960's, the automobile industries and other high demand, high speed industries created a

demand for faster, smaller and more reliable control system and devices [28]. The system that needs not to rewire a system every time for just a minor change in the process was required [28].

Therefore, electronic industry responded with such a device and thus the birth of the PLC [28]. Bedford Associates proposed Modular Digital Controller (MODICON) to a major US car manufacturer [28]. The MODICON 084 brought the world's first PLC into commercial production [28]. In the mid70's, the dominant PLC technologies were sequencer state-machines and the bit-slice based CPU [28]. Communications abilities began to appear in 1973 and the system was Modicon's Modbus [28]. In the 80's, General Motor's manufacturing automation protocol (MAP) attempted to standardize communications. The size of PLC was reduced. They were software programmable through symbolic programming on personal computer [28].

In the 90's, there was a gradual reduction of new protocols and the modernization of the physical layers of some more popular protocols that survived the 1980's [28]. Nowadays, PLCs are programmable in function block diagrams, instruction lists, C and structured text all at the same time. Personal computers are being used to replace PLCs in some application [28].

2.5.2 Introduction to PLC

PLC (Programmable Logic Controller) sometime called PEC (Programmable Electronic Controller) is a solid state device that designed to perform logic functions [28]. In other word, PLC is a computer that has connection to external inputs and outputs. It is invented to replace the necessary sequential relay circuits for machine control. PLC is said to be similar to a computer because it is assembly of digital logic elements that are designed to make logical decisions based on input status compared to a user program. These logical decisions are used to provide output to field devices. By looking at the

inputs and depending upon their state which user program via software, it will produce the desired results [28].

The disadvantages of relay are:

- Relays and mechanical devices that have limited lifetime which required strict adherence to maintenance schedules.
- Troubleshooting is very tedious when so many relays are involved.
- Complicated wiring of relays cause the size of the machine control panel became very huge.

The advantages of PLC are:

- Easily programmed by maintenance and plant engineers.
- Its lifetime was long and programming changes can be easily performed.
- Troubleshooting is relatively easy.
- It was more compact and thus small in size.

2.5.3 Structure of PLC

Basically, PLC is divided into 3 parts which are input, program and output. The structure of PLC is showed in Figure 3. In more detail, it can be divided into CPU, Input/output system and programming device.

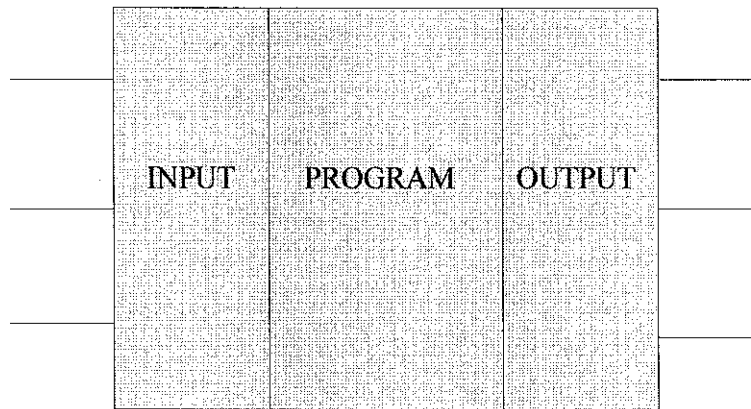


Figure 3: Structure of PLC

- **CPU (Central Processing Unit)**

CPU is the brain of PLC. It stores all of its data and does its entire computer processing [29].

- **Input/Output system**

Input/Output system is used to communicate signals to and from field devices [29]. Input interface is a collection of terminals that physically connects input devices to PLC. Input device provides data to the PLC while input interface translates data from the input into the form that the PLC's CPU can understand [29]. On the other hand, output interface is a collection of terminals that connects output devices to PLC. Output device receives control data from PLC while output interface translates data from the PLC's CPU into the form that the output devices can understand [29]. I/O system communicates information from input devices to CPU as well as communicates data from the CPU to the output devices [29].

- **Programming device [28]**

Programming device allows user to tell the PLC how to react to the input signal. In PLC, the general rule to program the PLC is shown in Figure 4.

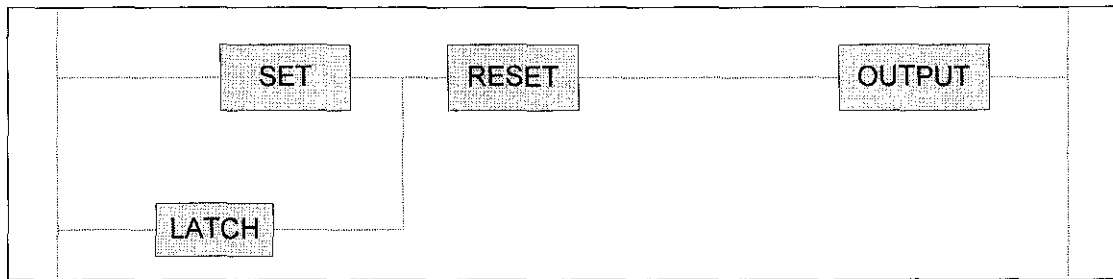


Figure 4 : General Rule of PLC Programming

There are several parts that involve in the process, such as:

Input relay

Typically they are not relay but rather transistor which is connected to the outside world. They physically exist and receive input signal from input device.

Internal utility relay

They do not physically exist and do not receive signal from the outside world. They are simulated relay. They enable PLC to eliminate external relay.

Counters

They are not physically exist. They are simulated counters and they can be programmed to count pulses. Since they are simulated they are limited in their counting speed.

Timers

They do not physically exist. They come in many varieties and increments. The increments vary from 1ms through 1s.

Output relay

Output relays are connected to the outside world. They physically exist and send on/off signals to output devices. They can be transistors, relays, or triacs.

Data Storage

They are register that assigned to store data. They used as temporary storage for math or data manipulation. They can also stored data when the power is removed from the PLC.

2.5.4 Theory of Operation of PLC

All PLCs perform three-step operation called a continually scan [30]. The scan consists of:

- **Reading / Checking input status**
Read the input data that the PLC receives from the input devices.
- **Executing program**
Execute the control program stored in memory. PLC executes one instruction at a time.
- **Writing/Updating output status**
Write the status of the output devices based on the outcome of the control program execution. A PLC performs the scan over and over again, constantly updating the outputs based on how new input conditions affect the control program.

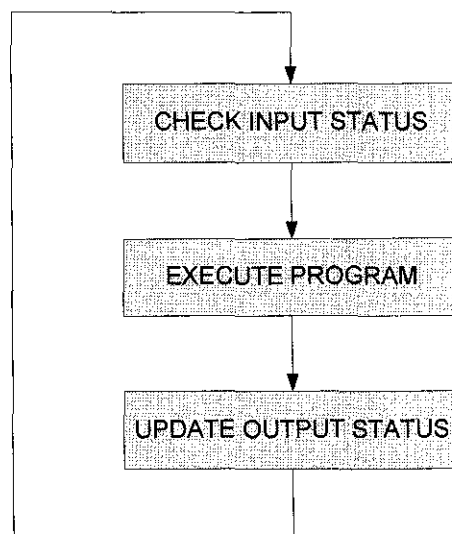


Figure 5: Scanning Process

After the third step the PLC goes back to step one and repeats the steps continuously. One scan time is defined as the time it takes to execute the 3 steps as shown in Figure 5 above.

CHAPTER 3

METHODOLOGY

3.1 Research Methodology

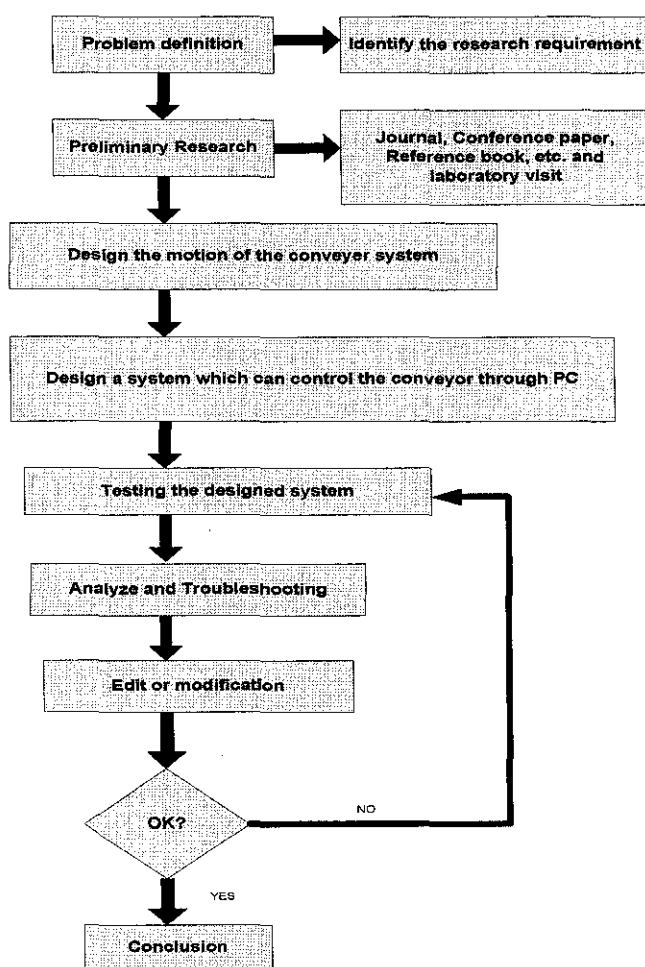


Figure 6: Flow chart of research methodology

3.2 Projects Activities

In order to develop a virtual lab to carry out experiments purpose based on web-based learning, the project work had been further broken down into several parts. There are four main parts, with the first two parts to be completed in FYP I and the rest are scheduled to be completed in FYP II. The project Gantt chart is attached in the Appendix I.

3.2.1 Hardware Connection and Software Installation

The main controller will be connected to personal computer by using RS232 connector. The computer is an intermediate device between user and the conveyer system. The plan of the conveyer system is shown at Figure 7. The conveyor belt is to be controlled by a PLC. The vision system shown in the figure is used to detect the presence and absent of an object.

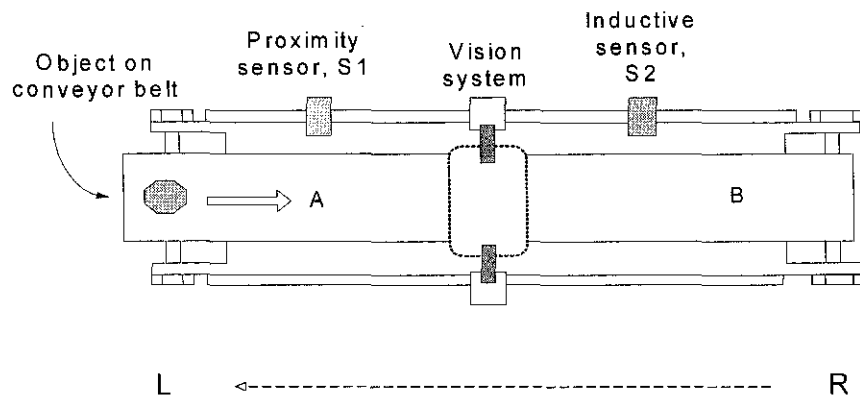


Figure 7: The plan of the conveyer in the laboratory

Figure 8 shows the circuit diagram of the conveyer system. It shows the connection between the PLC and the conveyor belt system which includes the motor. The motor is used to control the conveyor belt movement whether moving forward or moving reverse.

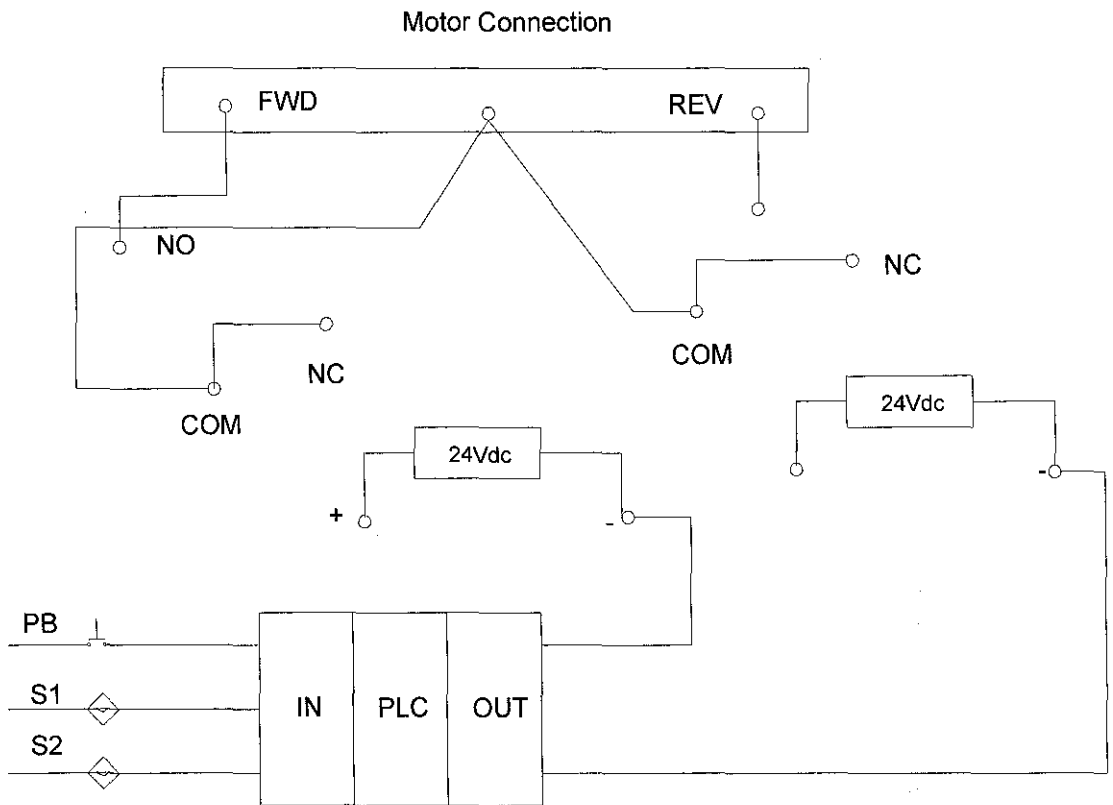


Figure 8: Circuit diagram of conveyor belt system

CX- Programmer software has been installed into the computer. In this stage, the ladder diagram will be designed in order to ensure that computer and conveyer can communicate with each other well.

3.2.2 Design the System

In the designing stage, there are several factors that have to be taken into consideration. First of all, it is important to ensure that the code which is going to be used is compatible with both PLC as well as internet. Then, the system will be planned by drawing the flow chart. Attempts to run the program will be made without any connectivity to internet. Basic user interface is designed in this stage by using C# programming language. Besides, it is imperative to ensure that the function can be run properly before proceeding to the next stage.

3.2.3 Upload the System Online

An interface for this Web-based learning automation laboratory is designed in this stage. Most important coding will be written in this stage. The coding includes writing a TCP/IP by using Visual Basic programming language. The computer in the lab will act as a server PC while student computers will act as client PC. All the interfaces that have been created will be uploaded online. The function of the web page will be added from time to time for the users' benefit.

3.2.4 Test the System

The next step is to test the designed system. This can be performed by accessing the lab by using the computer from different place such as computer lab, students' hostel, and lecturers' office and most probably from other state. This is a compulsory step to ensure the workability of the system. If there is any bug or error, the program should be edited. Then, a simple manual can be written for maintenance or even for enhancement purposes.

3.3 Tool and Component

The tools and components involve in this project include the controller for the conveyer and conveyer in the automation lab. The installer for CX-Programmer is needed to control the conveyer motion. The software such as Microsoft Visual Studio 2008 will be used in creating the interface between personal computer and the controller. The controller is connected to PC by using RS232 serial port. Then, Local Area Connection (LAN) cable is needed in order to connect the system to the internet.

Table 1: Hardware and software

Hardware	Software
Conveyer system	Microsoft Visual Studio 2008
CQM1H-CPU21 PLC	CX-Programmer
RS232 cable	
RJ45 cable	
Proximity sensor	
Relays	

The main hardware that involved in this project is the conveyer system that available in the Automation Laboratory at Block 22 as shown in Figure 9. This is the hardware which will be controlled by using server PC.

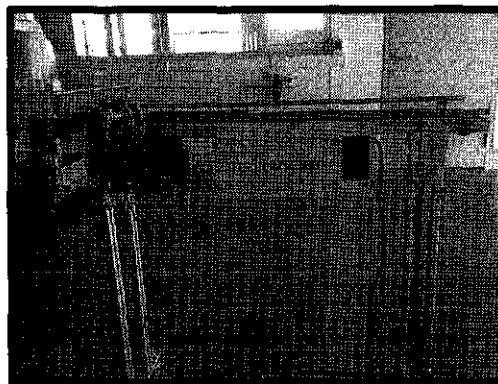


Figure 9: Conveyer system

The Programmable Controller Training Kit as shown in Figure 10 is the device that controls the conveyer system. It is build up of switches, internal relay and others. It is connected to PC by using RS232 serial port.

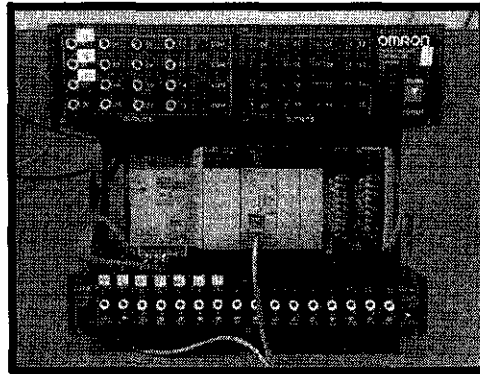


Figure 10: Programmable Controller Training Kit

RS232 cable which used to connect from the PLC to server PC is shown in Figure 11.

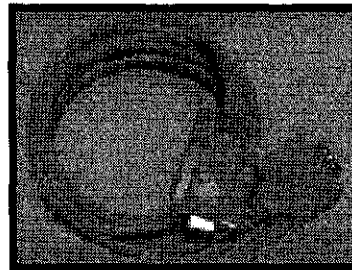


Figure 11: RS232 cable

RJ45 cable is showed in Figure 12. It is used to connect the server PC to internet port.

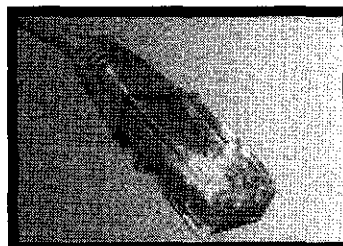


Figure 12: RJ45 cable

Microsoft Visual Studio 2008 is the compiler that used for this project. It can compile C#, C++, C, Visual Basic (VB) and others. However, only C# and VB language is used in this project. Figure 13 shows the symbol of the compiler.

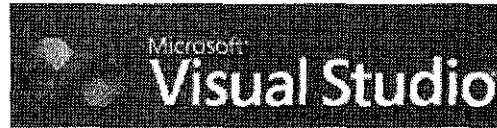


Figure 13: Microsoft Visual Studio 2008

CX-Programmer compiler is a PLC compiler for OMRON PLC. It can be used to create ladder diagram and run the simulation.

Figure 14 shows the basic relation between the server and client pc. The virtual lab architecture consists of 4 major parts which are server, client, internet, and the hardware to be controlled.

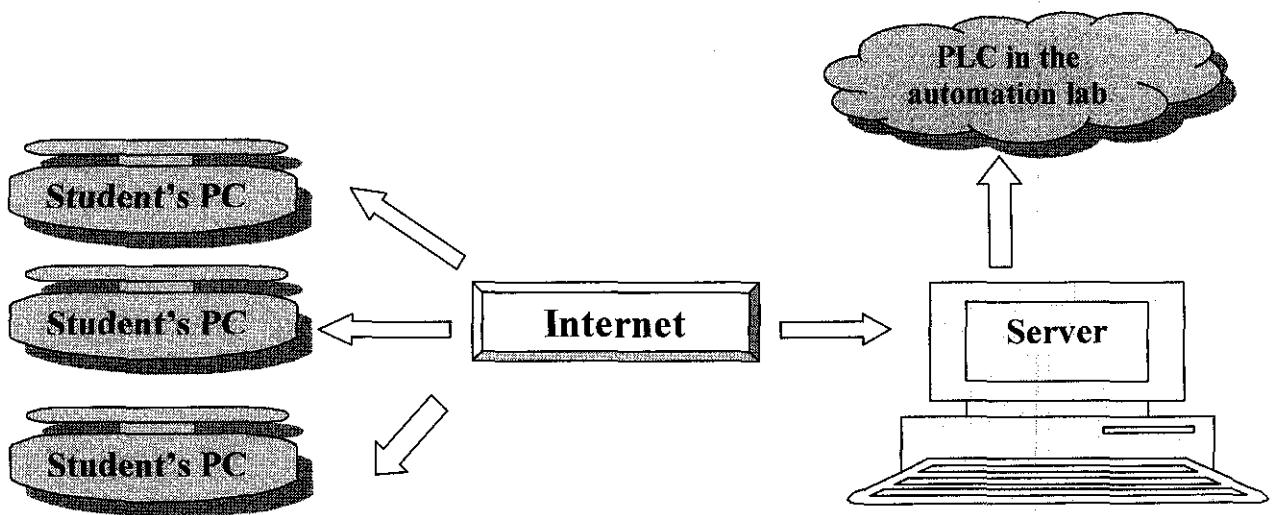


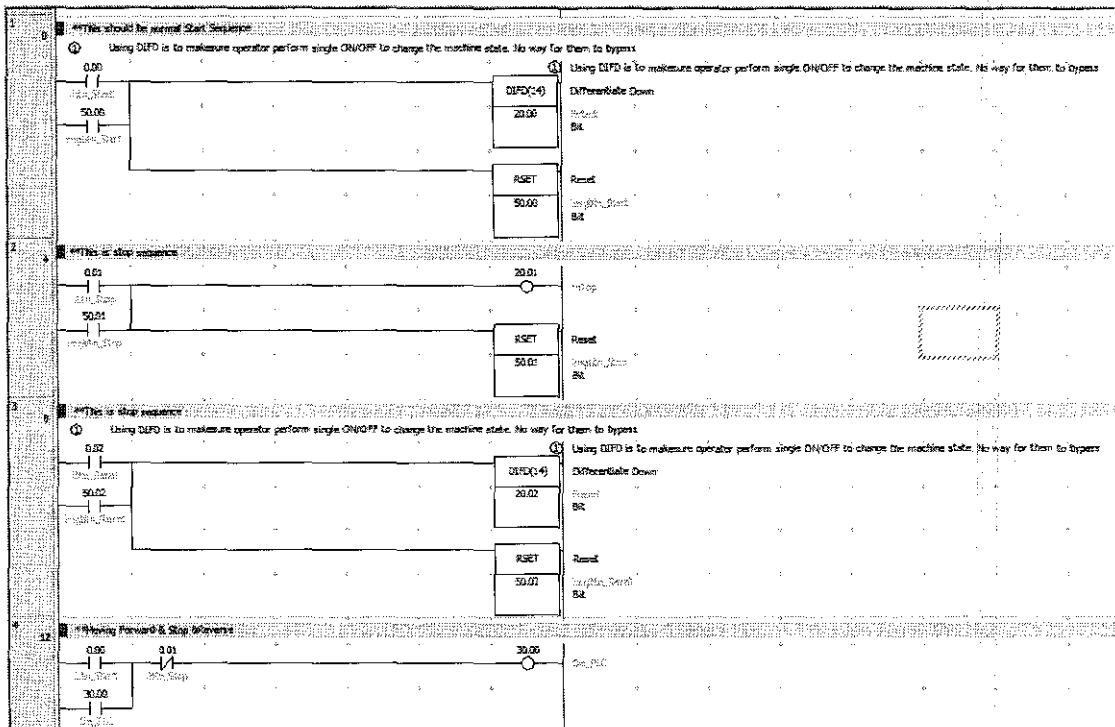
Figure 14: Virtual lab system architecture

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Ladder diagram

The ladder diagram is designed to preset the movement of the conveyor in the laboratory. First of all, the function of the ladder diagram is to turn on the PLC by turning on input 0 while stop the PLC motion by using input 2. When input 3 is turned on, the conveyor will be in forward motion. When input 4 is turned on, the conveyor will be in reverse motion. The ladder diagram of the input for the conveyor motion is shown in Figure 15 and the ladder diagram of the output for the conveyor motion is shown in Figure 16.



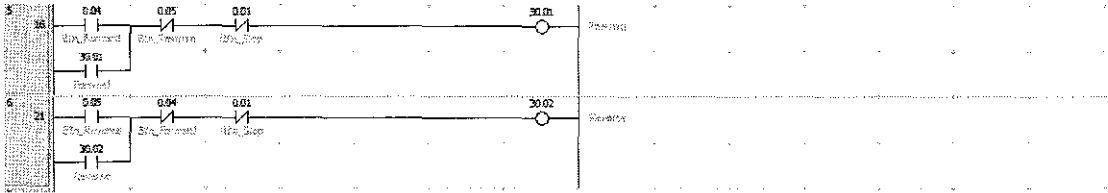


Figure 15: Ladder diagram for the motion of the conveyor

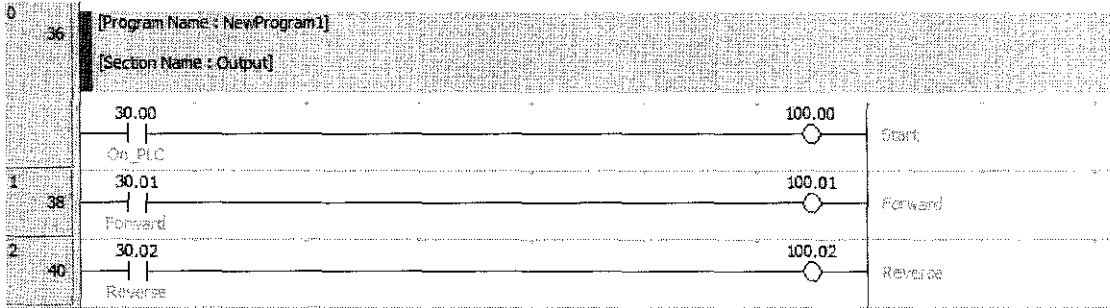


Figure 16: Output of the ladder diagram

4.2 Comment for the Symbol and Address in PLC

Table 2 : Comment for symbol and address in PLC

Symbol	Address	Comment
ibtn_Start	0.00	Start
ibtn_Stop	0.01	Stop
ibtn_Reset	0.02	Reset
Btn_Forward	0.03	Motor Conveyor moving forward
Btn_Reverse	0.04	Motor Conveyor moving reverse
F_Stop	20.01	Forward stop
imgbtn_Start	50.00	Start (activated by using created interface)
imgbtn_Stop	50.01	Stop (activated by using created interface)
imgbtn_Reset	50.02	Reset (activated by using created interface)

4.3 Result and Discussion on TCP/IP

The source code for TCP/IP has been attached in Appendix II. When this program is run, an interface as shown in Figure 17 will show on the screen. It requests user to key in target IP which is the IP address of the server. After the user types in the IP address as

shown in Figure 18, the connection is build between both server PC and client PC. When the user type in any command, it will be shown in another PC. This interface that seen by the sender is shown in Figure 19 and the interface seen by the receiver are shown in Figure 20.

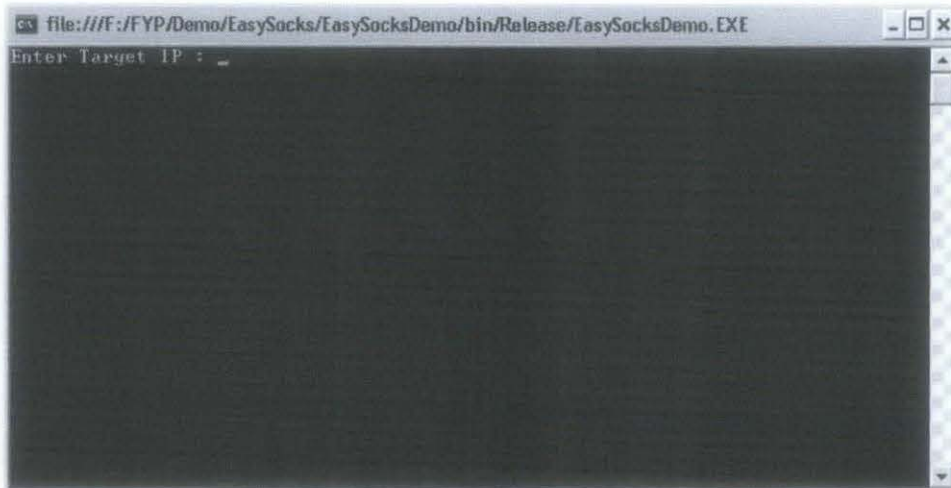


Figure 17: Interface of TCP/IP

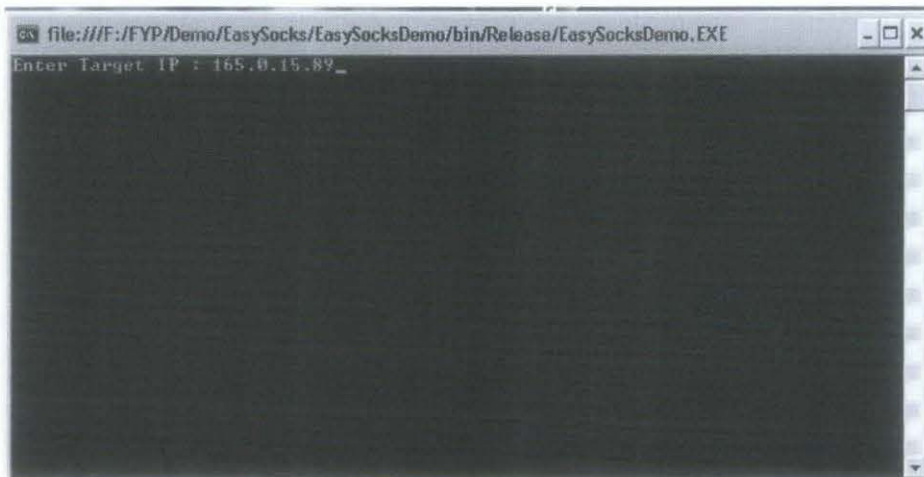


Figure 18: The target IP address is typed in

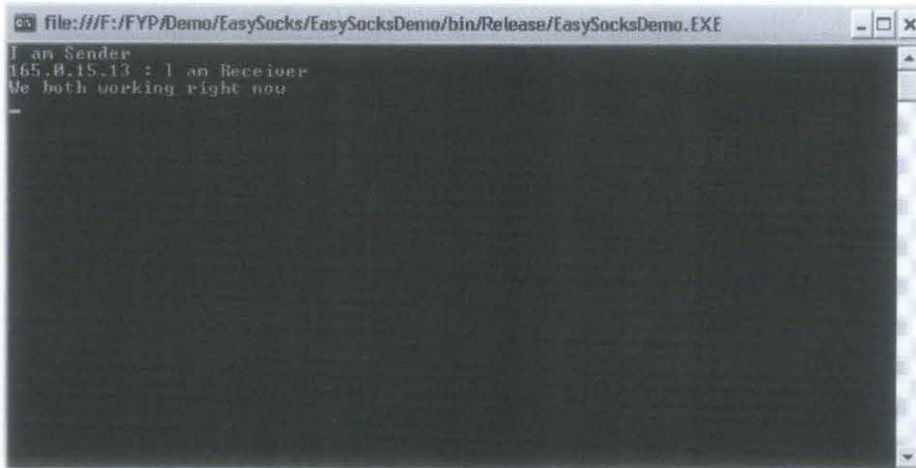


Figure 19: The interface of the sender

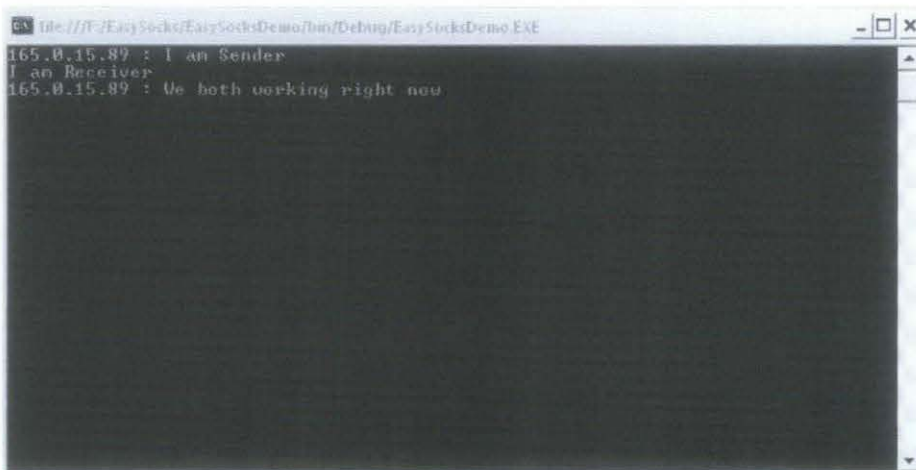


Figure 20: The interface of the receiver

The source code for TCP/IP with interface has been attached in Appendix II. When this program is run, an interface as shown in Figure 21 will show on the screen of server PC while interface Figure 22 will show on the screen of client PC. The IP address is auto detected by the program on the internet connection. When the button of “Connect To Server” is clicked, the communication path between client PC and server PC is created. Therefore, both of the PC can communicate with each other until the button of “Disconnect From Server” is clicked.

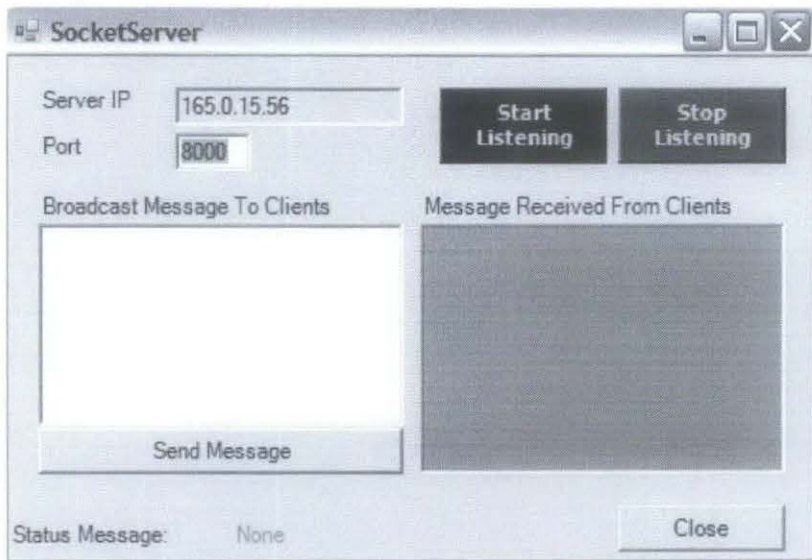


Figure 21 : Interface for Socket Server

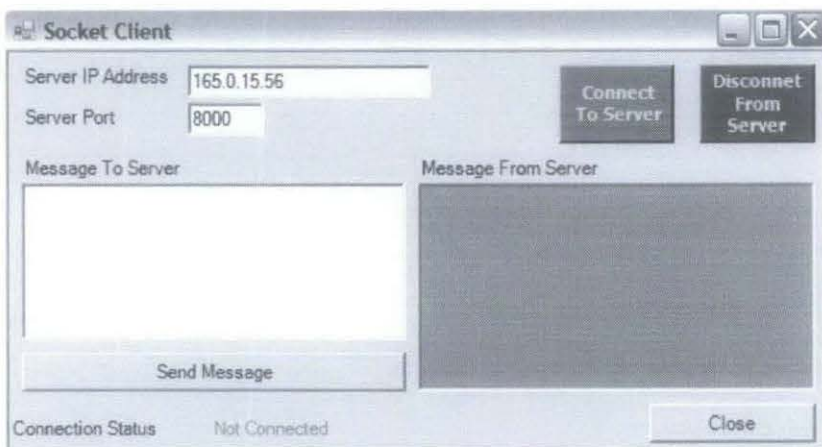


Figure 22 : Interface for Socket Client

4.4 Result and Discussion on PC-PLC Communication

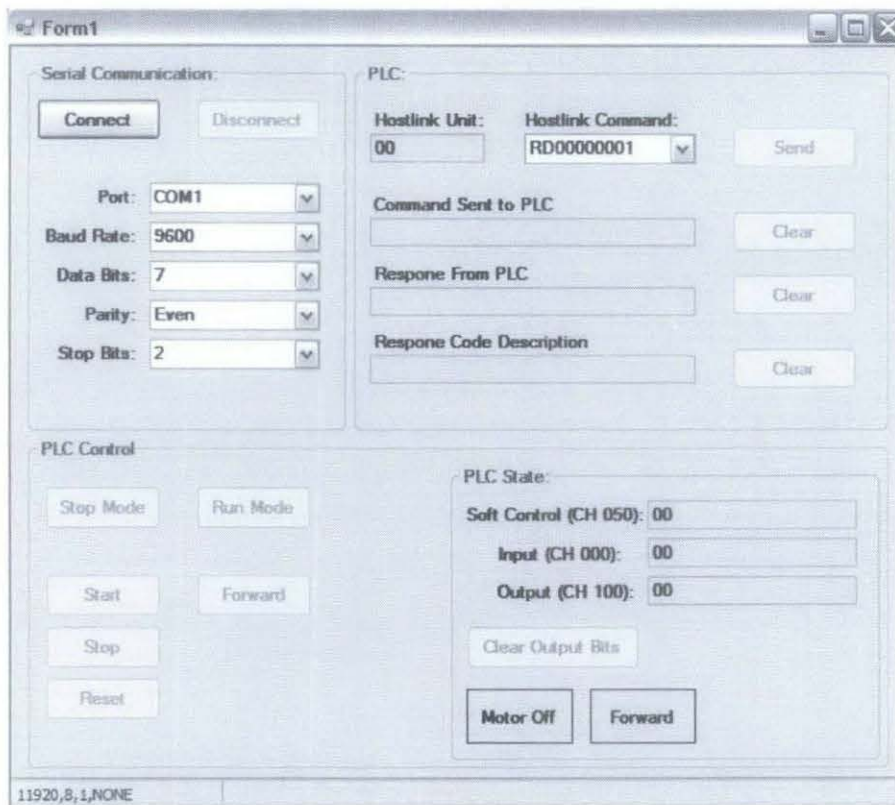


Figure 23 : Interface for PC-PLC

4.4.1 Serial Communication

In the drop down column of Port, it contains of COM1, COM2 and etc which is using to connect the PLC to the computer serial port. The COM number is auto detected from different computer. It is depends on the computer property. The Baud Rate is assigned to be 19200 or 9600 which is the speed. Baud rate of 19200 offers better refresh rate while doing online monitoring compared to baud rate of 9600. However, 9600 is the default baud rate for the PLC using in the lab. It is because when the PLC connects to touch screen (HMI Screen), usually the baud rate for the touch screen is 9600 only. Therefore, 9600 is chosen to be the baud rate for the program as well. The Data Bits is assigned to be 7. Data bit is depending on the hardware and 7 bits is true for ASCII. Parity is set to be even and lastly stop bit to be 2 (default for the PLC). It is because it allows the receiving hardware to detect the end character and to resynchronize with the character stream. This

setting is to set the bandwidth for the system. It is to ensure that this bandwidth is created uniquely for the PLC that is using in this particular project. Once the 'Connect' button is pressed, the communication path between the PLC and PC will be created.

4.4.2 PLC

This column is for debugging purpose. According to the default setting of CX-Programmer, the Hostlink Unit number is 0. Hostlink command can be chosen from the drop down list. For example:

RD	0000	0001
⏟	⏟	⏟
Command	Address	Data

After the hostlink command is sent to PLC, command send to PLC, response from PLC and response code description will be displayed on the text box.

4.4.3 PLC Control

This is the part that user can manipulate to control the motion of the conveyor. First of all, users have to click on "Run Mode" button to turn the PLC to run mode while "Stop Mode" button is to off the PLC from run mode. The "Start" button is to make the conveyor start moving in forward direction. When the motor is in forward direction, the "Forward" button will change to "Reverse" button and vice versa. Off site user is not able to turn on the physical push button to control the PLC. Therefore, this imaginary button is to replace the physical push button. The "Stop" button is to stop the forward motion or reverse motion of the conveyor.

4.4.4 *PLC Status*

This column is to show the status of the output. When the motor is turn on, the text will change to “Motor On” and it will change to lime green color. If the motor is turned off, the text will change to “Motor Off” and it will be in grey color. Besides, this column can indicate the motion of the motor as well. When the motor is in forward direction, the text will be in “Forward” and it will turn to lime green color. If the motor is in reverse direction, the text will show “Reverse” and it will turn to grey color.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project report has presented the main idea of web-based learning or remote laboratory. Basically, this project can be divided into 2 parts which are PLC-PC and PC-network. RS-232 cable is used to connect server computer and programmable controller training kit. An interface to control the PLC remotely has been created. From the interface, users are able to control the PLC by clicking on the imaginary button instead of turn on the physical push button. Then, the communication path between server computer and client computer is created by using TCP/IP protocol. It works as a messenger where client computer can communicate with server computer and vice versa. By integrating both of these programs, it lets the users to control the PLC anytime and anywhere without attending the laboratory. This can help the students to optimize their learning process with flexible time table accessing to the laboratory. Moreover, laboratory technician can optimize their productivity because do not need to explain the procedure repeatedly during each laboratory session. With reference to the project Gantt chart, the development of remote laboratory is considered successful. All the activities are carried out although there are some problems occurs in between. However, the online booking and auto generated login ID and password part is not able to be created due to time constraint.

5.2 Recommendations

Future work will be devoted in creating the system for online scheduling and reservation that able to auto generate login ID and password to the respective students during pre-registration session. This is to ensure that multi-user conflict can be solved since the PLC interface system is created for single user at a time. Beside, various factors will be taken into consideration including the speed of the server PC and the stability of the LAN in the university.

Moreover, the user must be able to observe the movement of the conveyer in real time basis. Web camera can be integrated in the future project to ensure the user can observe the real plant instead of words that indicate the conveyor status.

REFERENCES

- [1] Li,Y. , Esche,S. , Chassapis,C.
An architecture for real-time remote laboratories
(2007) Proceedings of the 2007 ASEE Annual Conference and Exposition
Hoboken, NJ, United States
- [2] Chiou,R. , Kwon,Y. , Rauniar,S. , Sosa,H.
Internet-based robotics and mechatronics experiments for remote laboratory development
(2007) Proceeding of 2007 ASEE Annual Conference and Exposition
Drexel University
- [3] Milo D. Koretsky, Danielle Amatore, Connelly Barnes and Sho Kimura
Enhancement of student learning in experimental design using a virtual laboratory
IEEE Transaction on Education, VOL. 52, No. 1, February 2008
- [4] Maher Chaabene, Kamel Mkaouar, Rachid Souissi
A web-based control of real laboratory for process engineering education
Applied Computer Science Research Group (ISET Sfax), Tunisia
- [5] Suzana Uran and Karel Jezernik
Virtual laboratory for creative control design experiments
IEEE Transaction on Education, VOL. 51, No.1, February 2008

- [6] Dongil Shin, En Sup Yoon, Kyung Yong Lee, Euy Soo Lee
A web-based, interactive virtual laboratory system for unit operations and process systems engineering education: issues, design and implementation
School of Chemical Engineering, Seoul National University, Seoul 151-742, South Korea
Department of Chemical Engineering, Dongguk University, Seoul 100-715, South Korea
- [7] A.Di sefano, F.Fazzino, L.Lo Bello, O. Mira bella
Virtual lab: a Java application for distance learning
Catania, Italy
- [8] Song You, TianMiao Wang, Roy Eagleson, Cai Meng, Qixian Zhang
A low-cost internet-based telerobotic system for access to remote laboratories
University of Western Ontario, London, Canada
Beijing University of Aeronautics and Astronautics, Beijing, People's Republic of China
- [9] Clinical review Web based learning, *BMJ*. 6 August 2008
<<http://www.bmj.com/cgi/content/full/326/7394/870>>.
- [10] Beginner basics, *e-LearningGuru*. 6 August 2008
<http://www.e-learningguru.com/articles/art1_9.htm>.
- [11] Christof Rogrig, Andreas Jochheim
The Virtual Lab for Controlling Real Experiments via Internet
(1999) Proceedings of the 1999 IEEE, International Symposium on Computer Aided Control System Design, Kohala Coast-Island of Hawaii, Hawaii, USA

- [12] M.J. Callaghan, J. Harkin, E. McColgan, T.M. McGinnity, L.P. Maguire
Client-server architecture for collaborative remote experimentation
Intelligent Systems Engineering Laboratory, Faculty of Engineering, University of
Ulster, Magee Campus, Morthland RD, Derry, N.Ireland, UK
- [13] Chung Ko C, Chen BM, Hu S, Ramakrishnan V, Dong Cheng C, Zhuang Y
Web-based virtual laboratory for frequency modulation experiment
IEEE Transactions on Systems, Man and Cybernetics
- [14] Behrouz A. Forouzan, Sophia Chung Fegan, TCP/IP Protocol Suite, Second
Edition, McGraw-Hill Company, Inc., 2003. ISBN 0-07-246060-1.
- [15] William Stallings, Data and Computer Communications, Eighth Edition, Pearson
Education, Inc., 2007. ISBN 0-13-238195-8.
- [16] Andrew G. Blank, TCP/IP JumpStart, SYBEX Inc., 2000. ISBN 0-7821-2644-8
- [17] Colton, C., K., Knight, R., A., Ibrahim, S., West, R.
A Web-accessible Heat Exchanger Experiment
International Conference of Engineering Education, Valencia, Spain, July 21-
25,2003
- [18] P.K. Imbrie and Seetha Raghavan
**Work In Progress – A remote e-Laboratory for Student Investigation,
Manipulation and Learning**
35th ASEE/IEEE Frontiers in Education Conference

- [19] Euan Lindsay, Phil Long and PK Imbrie
Workshop – Remote Laboratories: Approaches for the Future
37th ASEE/IEEE Frontiers in Education Conference
- [20] Yorkovich, S. and k.M. Passino
An intelligent control laboratory course
Proc. 13th IFAC World Congress, s. Francisco, California, 1996
- [21] M. L. Corradini, G. Ippoliti, T. Leo, S. Longhi
An Internet based Laboratory for Control Education
Proceeding of the 40th IEEE Conference on Decision and Control Orlando, Florida
USA, December 2001
- [22] Bohus, C., L. A. Crowl, B. Aktan and M.H. Shor
Running Control Engineering experiments over the Internet
Proc. 13th IFAC World Congress, s. Francisco, California, 1996
- [23] Shen H., Xu Z., Dalager B., Kristiansen V., Strom O., et al,
Conducting Laboratory Experiments over the Internet
IEEE Trans. On Education, Vol. 42, No.3, 1999
- [24] Tuttas J. and Wagner B
Distributed Online Laboratories
International Conference on Engineering Education 2001, Oslo, Norway, August
6-10, 2001

- [25] Soderlund A., Ingvarson F., Lundgren P. and Jeppson K.
Laboratory – A New Complement in Engineering Education
International Conference on Engineering Education 2002, Manchester, U.K.,
August 18-21, 2002
- [26] Dervis Z. Deniz, Atilla Bulancak and Gokhan Ozcan
A Novel Approach to Remote Laboratories
33rd ASEE/IEEE Frontiers in Education Conference
- [27] Fernandez R., Kinguti E. and Ramirez-Fernandez F.
A Virtual Laboratory to Perform Electronic Experiments by Internet
International Conference on Engineering Education 2002, Manchester, U.K.,
August 18-21, 2002
- [28] Programmable Logic Controller, PLCS.net . 8 October 2008
< <http://www.plcs.net/contents.shtml> >
- [29] K. Clements-Jewery and W. Jeffcoat, The PLC Workbook, Programmable Logic
Controllers made easy, Pearson Education, Inc., 2007. ISBN 0-13-489840-0.
- [30] Curtis D. Johnson, Process Control Instrumentation Technology, Eight Edition,
Pearson Education, Inc., 2006. ISBN 0-13-197669-9.
- [31] SIEMENS SIMATIC.NET Manual
- [32] Training Kit Training Manual

- [33] Training Kit Operation Manual
- [34] Asfahl, C.Ray, Robotics and Manufacturing Automation, John Wiley & Sons, Inc. New York, NY, 1992.
- [35] World Programmable Logic Controller Market, Frost & Sullivan Research. Publication 7450-01-00-00-00, Publication Date: 18 July 2001
- [36] Filer, R. and Leinonen, G. Programmable Controllers Using Allen-Bradley SLC 500 and ControlLogix, Pearson Education, Inc., Upper Saddle River, NJ, 2002.
- [37] Sheng-Jen Hsieh, Patricia Yee Hsieh
Web-Based Programmable Logic Controller Learning System
32rd ASEE/IEEE Frontiers in Education Conference
- [38] Pasquale Arpaia, Aldo Baccigalupi, Felice Cennamo, Pasquale Daponte
A Measurement Laboratory on Geographic Network for Remote Test Experiments
IEEE Transaction Instrumentation and Measurement, VOL. 49, No. 5 OCTOBER 2000.
- [39] J.A. Orr, B. A. Eisenstein
Summary of innovations in electrical engineering curricula
IEEE Transaction Education, VOL. 37, No. 2, 1994.

- [40] M. Cobby, D.Nicol, T.S.Durrani, W.A. Sandham
Teaching Electronic Engineering via the World Wide Web
In Proc. IEE Colloquium Computer Based Learning in Electronic Education,
London, U.K., May 10, 1995.
- [41] F. Alegria, H.G. Ramos
A Remote Controlled Automated Measurement Systems
In Proc. Instr. and Meas. Tech. Conf., IEEE IMTC, Ottawa, May 19-21, 1997.
- [42] M. Parvis, U.Pisani
Multimediality and Remote Training in Measurement Electronic Courses
In Proc. XII GMEE Congress, Bologna, Italy, September 1995.
- [43] M. Bertocco, F. Ferraris, C. Offelli, M. Parvis
Training of Programmable Instrumentation: A Student Laboratory
In Proc. XIV IMEKO World Congr., Tampere, Finland, June 1-6, 1997.
- [44] P. Arpaia, F. Cennamo, P. Daponte, M. Savastano
A Distributed Laboratory Based on Object-Oriented Systems
Measurement, VOL.19, Nov/Dec. 1996.
- [45] A. Heinze and C. Procter
Reflections on the use of blended learning
In Proc. Educ. Changing Environ. Conf., Salford, U.K., 2004.

- [46] Rui Marques, Jaime Rocha, Silvano Rafael, and J.F. Martins, Senior Member, IEEE
Design and Implementing of a Reconfigurable Remote Laboratory, Using Oscilloscope/PLC Network for WWW Access
IEEE Transactions on Industrial Electronics, VOL.55, No.6, June 2008.
- [47] Douglas C. Sicker, Tom Lookabaugh, Jose Santos, Frank Barnes
Assessing the Effectiveness of Remote Networking Laboratories
35th ASEE/IEEE Frontiers in Education Conference
- [48] Baumgartner, F., Braun, T., Kurt, E., Weyland, A.
Virtual Routers: a tool for networking research and education
Computer Communication Review, 33(3): 127-135 (2003).
- [49] Ogot. M., G. Elliott, and N. Glumac
An Assessment of In-person and Remotely Operated Laboratories
Journal of Engineering Education, VOL. 92, No.1, 2003.

APPENDICES

- APPENDIX I Gantt Chart of Final Year Project 2
- APPENDIX II Source Code
- APPENDIX III Flow Chart

APPENDIX I

GANTT CHART OF FINAL YEAR PROJECT II

No	Detail/Week	1	2	3	4	5	6	7	8	9	Mid Semester Break				10	11	12	13	14				
1	Hardware connection <ul style="list-style-type: none"> Connect the conveyer to the computer 																						
2	Ladder Diagram Design <ul style="list-style-type: none"> Moving forward ladder diagram Moving reverse ladder diagram Moving forward and reverse ladder diagram 																						
3	Submission of Progress Report 1																						
4	Computer Interface Design <ul style="list-style-type: none"> Interface at server PC 																						
5	Submission of Progress Report 2																						
6	TCP/IP code <ul style="list-style-type: none"> Connecting Client and Server PC through internet 																						

7	Seminar																		
8	Poster Exhibition																		
9	TCP/IP code (continue)																		
10	Submission of Dissertation																		
11	Oral Presentation																		
12	Submission of Project Dissertation																		
13	Submission of Project Technical Paper																		

APPENDIX II

Source Code for PC-PLC interface

```
using System;
using System:
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Net;
using System.Net.Sockets;

namespace OmronPLC
{
    public partial class frmMain : Form
    {
        SerialPort m_OmronPLC = new SerialPort();
        private delegate void CrossThreadHandler();
        private String m_Serial_data = String.Empty;
        private Int32 m_Seq = -1;
        const string CR = "\x13";
        private Int32 m_Direction = 0;

        public frmMain()
        {
            InitializeComponent();
            m_OmronPLC.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.DataReceived);
        }

        private void frmMain_Load(object sender, EventArgs e)
        {
            tmrUpdatePLC.Stop();
            cboPort.Enabled = true;
            cboBaud_Rate.Enabled = true;
            cboData_Bits.Enabled = true;
            cboParity.Enabled = true;
            cboStop_Bits.Enabled = true;
            btnConnect.Enabled = true;
            btnDisconnect.Enabled = false;
            btnSend.Enabled = false;
            btnClear_2PLC.Enabled = false;
            btnClear_FromPLC.Enabled = false;
            btnClear_CodeFromPLC.Enabled = false;

            cboPort.Items.Clear(); //clear list view
            foreach (String s in SerialPort.GetPortNames())
            {
                cboPort.Items.Add(s);
            }
        }
    }
}
```

```

}

private void btnConnect_Click(object sender, EventArgs e)
{
    StringBuilder _sb = new StringBuilder();

    m_OmronPLC.PortName = cboPort.Text;
    m_OmronPLC.BaudRate = Int32.Parse(cboBaud_Rate.Text);
    m_OmronPLC.DataBits = Int32.Parse(cboData_Bits.Text);
    m_OmronPLC.Parity = (Parity)cboParity.SelectedIndex;
    m_OmronPLC.StopBits = (StopBits)Int32.Parse(cboStop_Bits.Text);

    //open port
    if (!m_OmronPLC.IsOpen)
    {
        m_OmronPLC.Open();
        //enable
        cboPort.Enabled = false;
        cboBaud_Rate.Enabled = false;
        cboData_Bits.Enabled = false;
        cboParity.Enabled = false;
        cboStop_Bits.Enabled = false;
        btnConnect.Enabled = false;
        btnDisconnect.Enabled = true;
        btnSend.Enabled = true;
        btnClear_2PLC.Enabled = true;
        btnClear_FromPLC.Enabled = true;
        btnClear_CodeFromPLC.Enabled = true;

        //display at status bar
        _sb.Append(cboPort.Text).Append(", ").
            Append(cboBaud_Rate.Text).Append(", ").
            Append(cboData_Bits.Text).Append(", ").
            Append(cboParity.Text).Append(", ").
            Append(cboStop_Bits.Text);
        tssSerial.Text = _sb.ToString();
        btnStopMode.Enabled = false;
        btnRunMode.Enabled = true;
    }
}

private void btnDisconnect_Click(object sender, EventArgs e)
{
    tmrUpdatePLC.Stop();
    if (m_OmronPLC.IsOpen)
    {
        m_OmronPLC.Close();
    }
    cboPort.Enabled = true;
    cboBaud_Rate.Enabled = true;
    cboData_Bits.Enabled = true;
    cboParity.Enabled = true;
    cboStop_Bits.Enabled = true;
    btnConnect.Enabled = true;
    btnDisconnect.Enabled = false;
    btnSend.Enabled = false;
    btnClear_2PLC.Enabled = false;
    btnClear_FromPLC.Enabled = false;
    btnClear_CodeFromPLC.Enabled = false;
}

```

```

        //set default mode
        btnStopMode.Enabled = false;
        btnRunMode.Enabled = false;
        EnablePLC_Control(false);
    }

    private void btnClear_FromPLC_Click(object sender, EventArgs e)
    {
        txtFromPLC.Text = String.Empty;
    }

    private void btnClear_2PLC_Click(object sender, EventArgs e)
    {
        txt2PLC.Text = String.Empty;
    }

    private void btnClear_CodeFromPLC_Click(object sender, EventArgs e)
    {
        txtCodeFrmPLC.Text = String.Empty;
    }

    private void btnSend_Click(object sender, EventArgs e)
    {
        String _cmd = String.Empty;
        txt2PLC.Text = String.Empty;
        txtFromPLC.Text = String.Empty;
        txtCodeFrmPLC.Text = String.Empty;
        tmrUpdatePLC.Stop();

        _cmd = "@" + txtHostLink_Unit.Text + cboPLC_Cmd.Text;
        _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
        txt2PLC.Text = _cmd;
        m_OmronPLC.Write(_cmd);
        tmrUpdatePLC.Start();
    }

    private String CheckSum(String Command)
    {
        string _hexstring = String.Empty;
        char[] chars = Command.ToCharArray();
        Int32 _cks = 0;

        for (int i = 0; i < Command.Length; i++)
        {
            _cks = _cks ^ (Int32)chars[i];
        }
        _hexstring = _cks.ToString("X");

        if (_hexstring.Length < 2)
        {
            _hexstring = "0" + _hexstring;
        }
        return _hexstring;
    }

    public String EndCode(String Command)
    {
        String _end_Code = String.Empty;
        switch (Command)

```

```

{
    case "1C":
        _end_Code = "Unknow Command";
        break;

    case "00":
        _end_Code = "Normal Completion";
        break;

    case "13":
        _end_Code = "FCS Error";
        break;

    case "14":
        _end_Code = "Format Error";
        break;

    case "18":
        _end_Code = "Frame Length Error";
        break;

    case "01":
        _end_Code = "Not Executable in Run Mode";
        break;

    case "02":
        _end_Code = "Not Executable in Monitor Mode";
        break;

    case "23":
        _end_Code = "User Memory Protected";
        break;

    case "15":
        _end_Code = "Entry Number Data Error";
        break;
}
return _end_Code;
}

private void DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
{
    String _input_data = String.Empty;
    String _resp = String.Empty;
    m_Serial_data = String.Empty;

    for (Int32 i = 0; i < 1000; i++)
    {
        System.Threading.Thread.Sleep(100);
        _input_data += m_OmronPLC.ReadExisting();
        _resp = _input_data.Substring(_input_data.Length - 1, 1);
        if (_resp.CompareTo("\r") == 0)
        {
            m_Serial_data = _input_data.Substring(0, _input_data.Length - 1);
            ProcessshPortData();
            break;
        }
    }
}
}

```

```

private void ProcessshPortData()
{
    String _ret_code = String.Empty;

    if (this.InvokeRequired)
    {
        CrossThreadHandler d = new CrossThreadHandler(ProcessshPortData);
        this.Invoke(d);
    }
    else
    {
        txtFromPLC.Text = m_Serial_data;
        if (m_Serial_data.Length < 10)
        {
            _ret_code = m_Serial_data.Substring(4, 2);
        }
        else
        {
            _ret_code = m_Serial_data.Substring(5, 2);
        }
        txtCodeFrmPLC.Text = EndCode(_ret_code);
    }
}

private void btnSoftStart_Click(object sender, EventArgs e)
{
    String _cmd = String.Empty;
    txt2PLC.Text = String.Empty;
    //@      00      WR      0050      FFFF
    //header  hostlink  Command  address  data

    _cmd = "@" + txtHostLink_Unit.Text + "WR00500001";
    _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
    txt2PLC.Text = _cmd;
    m_OmronPLC.Write(_cmd);
}

private void btnSoftStop_Click(object sender, EventArgs e)
{
    String _cmd = String.Empty;
    txt2PLC.Text = String.Empty;
    //@      00      WR      0050      FFFF
    //header  hostlink  Command  address  data

    _cmd = "@" + txtHostLink_Unit.Text + "WR00500002";
    _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
    txt2PLC.Text = _cmd;
    m_OmronPLC.Write(_cmd);
}

private void btnSoftReset_Click(object sender, EventArgs e)
{
    String _cmd = String.Empty;
    txt2PLC.Text = String.Empty;
    //@      00      WR      0050      FFFF
    //header  hostlink  Command  address  data

    _cmd = "@" + txtHostLink_Unit.Text + "WR00500004";
}

```

```

    _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
    txt2PLC.Text = _cmd;
    m_OmronPLC.Write(_cmd);
}

private void btnStopMode_Click(object sender, EventArgs e)
{
    String _cmd = String.Empty;
    txt2PLC.Text = String.Empty;
    tmrUpdatePLC.Stop();
    _cmd = "@" + txtHostLink_Unit.Text + "SC00";
    _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
    txt2PLC.Text = _cmd;
    m_OmronPLC.Write(_cmd);
    btnStopMode.Enabled = false;
    btnRunMode.Enabled = true;
    EnablePLC_Control(false);
    tmrUpdatePLC.Start();
    m_Seq = -1;
}

private void btnRunMode_Click(object sender, EventArgs e)
{
    String _cmd = String.Empty;
    txt2PLC.Text = String.Empty;
    tmrUpdatePLC.Stop();
    _cmd = "@" + txtHostLink_Unit.Text + "SC02";
    _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
    txt2PLC.Text = _cmd;
    m_OmronPLC.Write(_cmd);
    btnStopMode.Enabled = true;
    btnRunMode.Enabled = false;
    EnablePLC_Control(true);
    tmrUpdatePLC.Start();
    m_Seq = 0;
}

private void EnablePLC_Control(Boolean State)
{
    btnSoftStart.Enabled = State;
    btnSoftStop.Enabled = State;
    btnSoftReset.Enabled = State;
    btnClearOutputBits.Enabled = State;
    btnDirection.Enabled = State;
}

private void tmrUpdatePLC_Tick(object sender, EventArgs e)
{
    String _cmd = String.Empty;
    Int32 _value = 0;

    switch (m_Seq)
    {
        case 0:
            //channel 100 - output
            _cmd = String.Empty;
            _cmd = "@" + txtHostLink_Unit.Text + "RR01000001";
            _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
            m_OmronPLC.Write(_cmd);

```



```

System.Threading.Thread.Sleep(150);
txt_100.Text = m_Serial_data.Substring(7, 4);
System.Threading.Thread.Sleep(150);
//update the UI
_value = int.Parse(txt_100.Text);
//
_value = 1; // int.Parse(txt_100.Text);
// motor on/off
// mask the result
if ((_value & 1) == 1)
{
    lblMtrStatus.Text = "Motor On";
    lblMtrStatus.BackColor = Color.LimeGreen;
}
else
{
    lblMtrStatus.Text = "Motor Off";
    lblMtrStatus.BackColor = Color.LightGray;
}
// motor direction on/off
if ((_value & 2) == 1)
{
    lblMtrDirection.Text = "Forward";
    lblMtrDirection.BackColor = Color.LimeGreen;
}
else
{
    lblMtrDirection.Text = "Reverse";
    lblMtrDirection.BackColor = Color.LightGray;
}

m_Seq = 1;
break;

case 1:
//channel 000 - input
_cmd = String.Empty;
_cmd = "@" + txtHostLink_Unit.Text + "RR00000001";
_cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
m_OmronPLC.Write(_cmd);
System.Threading.Thread.Sleep(150);
txt_000.Text = m_Serial_data.Substring(7, 4);
System.Threading.Thread.Sleep(150);
m_Seq = 2;
break;

case 2:
//channel 050 - soft control
_cmd = String.Empty;
_cmd = "@" + txtHostLink_Unit.Text + "RR00500001";
_cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
m_OmronPLC.Write(_cmd);
System.Threading.Thread.Sleep(150);
txt_050.Text = m_Serial_data.Substring(7, 4);
System.Threading.Thread.Sleep(150);
m_Seq = 0;
break;
}

```

```

}

private void btnClearOutputBits_Click(object sender, EventArgs e)
{
    String _cmd = String.Empty;
    txt2PLC.Text = String.Empty;
    //@      00      WR      0100      0000
    //header  hostlink  Command  address  data

    _cmd = "@" + txtHostLink_Unit.Text + "WR01000000";
    _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
    txt2PLC.Text = _cmd;
    m_OmronPLC.Write(_cmd);
}

private void btnDirection_Click(object sender, EventArgs e)
{
    m_Direction ^= 1;

    if (m_Direction == 0)
    {
        String _cmd = String.Empty;
        txt2PLC.Text = String.Empty;
        //@      00      WR      0050      FFFF
        //header  hostlink  Command  address  data

        _cmd = "@" + txtHostLink_Unit.Text + "WR00600000";
        _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
        txt2PLC.Text = _cmd;
        m_OmronPLC.Write(_cmd);
        btnDirection.Text = "Forward";
    }
    else // reverse
    {
        String _cmd = String.Empty;
        txt2PLC.Text = String.Empty;
        //@      00      WR      0050      FFFF
        //header  hostlink  Command  address  data

        _cmd = "@" + txtHostLink_Unit.Text + "WR00600001";
        _cmd = _cmd + CheckSum(_cmd) + "*" + (char)13;
        txt2PLC.Text = _cmd;
        m_OmronPLC.Write(_cmd);
        btnDirection.Text = "Reverse";
    }
}
}
}

```

Source code for TCP/IP

Module EasySocksDemo

```
Dim WithEvents ezySocks As New NERVLabs.EasySocks
Sub Main()
    Console.WriteLine("Enter Target IP : ")
    Dim IP As String = Console.ReadLine
    Console.Clear()
    ezySocks.SetTargetIP(IP)
    ezySocks.Start()
    Do
        Dim strInput As String = Console.ReadLine()
        Select Case strInput
            Case "exit"
                End
            Case Else
                ezySocks.SendData(strInput)
        End Select
    Loop
End Sub

Private Sub ezySocks_IncomingData(ByVal data As String, ByVal IP As String) Handles ezySocks.IncomingData
    Select Case data
        Case "slap"
            Console.WriteLine(IP & " slaps you")
        Case Else
            Console.WriteLine(IP & " : " & data.Trim(Chr(10)).Trim(Chr(11)))
        End Select
    End Sub
End Module
```

Library of TCP/IP Code

```
Imports System.Net
Imports System.Net.Sockets
Imports System.Threading
Imports System.Text

Public Class EasySocks
    Dim sockServer As New TcpListener(New IPEndPoint(IPAddress.Any, 10037))
    Dim sockClient As New TcpClient()
    Dim targetIP As String = "127.0.0.1"

    Dim sockServerClientConnected As New ManualResetEvent(False)

    Public Sub SetTargetIP(ByVal IP As String)
        targetIP = IP
    End Sub

    Public Sub Start()
        Dim srvThread As New Thread(AddressOf StartListener)
        srvThread.SetApartmentState(ApartmentState.STA)
        srvThread.IsBackground = True
        srvThread.Start()
    End Sub

    Private Sub StartListener()
        sockServerClientConnected.Reset()
        sockServer.Start()
        sockServer.BeginAcceptTcpClient(New AsyncCallback(AddressOf
DoAcceptSockServerCallback), sockServer)
        sockServerClientConnected.WaitOne()
    End Sub

    Public Sub DoAcceptSockServerCallback(ByVal ar As IAsyncResult)
        Try
            Dim listener As TcpListener = CType(ar.AsyncState, TcpListener)
            Dim client As TcpClient = listener.EndAcceptTcpClient(ar)

            Dim netStream As NetworkStream = client.GetStream()
            If netStream.CanRead Then
                Dim bytes(client.ReceiveBufferSize) As Byte
                netStream.Read(bytes, 0, CInt(client.ReceiveBufferSize))
                Dim returndata As String = Encoding.ASCII.GetString(bytes).Trim(Chr(0))
                RaiseEvent IncomingData(returndata, CType(client.Client.RemoteEndPoint,
IPEndPoint).Address.ToString)
            Else
                client.Close()
                netStream.Close()
                Return
            End If
            client.Close()
            sockServer.BeginAcceptTcpClient(New AsyncCallback(AddressOf
DoAcceptSockServerCallback), sockServer)
            sockServerClientConnected.Set()
        Catch
            sockServer.BeginAcceptTcpClient(New AsyncCallback(AddressOf
DoAcceptSockServerCallback), sockServer)
        End Try
    End Sub

    Public Event IncomingData(ByVal data As String, ByVal IP As String)

    Public Sub SendData(ByVal data As String, Optional ByVal IP As String = "NotSet")
        If IP = "NotSet" Then
            IP = targetIP
        End If
        Try
            sockClient.Connect(IP, 10037)
            If sockClient.Connected Then
                Dim netStream As NetworkStream = sockClient.GetStream()
                If netStream.CanWrite Then
                    Dim sendBytes As [Byte]() = Encoding.UTF8.GetBytes(data)
                    netStream.Write(sendBytes, 0, sendBytes.Length)
                End If
            End If
        End If
    End Sub
End Class
```

Source code for Socket Server

```
using System;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace DefaultNamespace
{
    /// <summary>
    /// Description of SocketServer.
    /// </summary>
    public class SocketServer : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.RichTextBox richTextBoxReceivedMsg;
        private System.Windows.Forms.TextBox textBoxPort;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TextBox textBoxMsg;
        private System.Windows.Forms.Button buttonStopListen;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.RichTextBox richTextBoxSendMsg;
        private System.Windows.Forms.TextBox textBoxIP;
        private System.Windows.Forms.Button buttonStartListen;
        private System.Windows.Forms.Button buttonSendMsg;
        private System.Windows.Forms.Button buttonClose;

        const int MAX_CLIENTS = 10;

        public AsyncCallback pfnWorkerCallBack ;
        private Socket m_mainSocket;
        private Socket [] m_workerSocket = new Socket[10];
        private int m_clientCount = 0;

        public SocketServer()
        {
            //
            // The InitializeComponent() call is required for Windows Forms designer
            support.
            //
            InitializeComponent();

            // Display the local IP address on the GUI
            textBoxIP.Text = GetIP();
        }

        [STAThread]
        public static void Main(string[] args)
        {
            Application.Run(new SocketServer());
        }

        #region Windows Forms Designer generated code
        /// <summary>
        /// This method is required for Windows Forms designer support.

```

```

    /// Do not change the method contents inside the source code editor. The Forms
designer might
    /// not be able to load this method if it was changed manually.
    /// </summary>
private void InitializeComponent() {
    this.buttonClose = new System.Windows.Forms.Button();
    this.buttonSendMsg = new System.Windows.Forms.Button();
    this.buttonStartListen = new System.Windows.Forms.Button();
    this.textBoxIP = new System.Windows.Forms.TextBox();
    this.richTextBoxSendMsg = new System.Windows.Forms.RichTextBox();
    this.label1 = new System.Windows.Forms.Label();
    this.buttonStopListen = new System.Windows.Forms.Button();
    this.textBoxMsg = new System.Windows.Forms.TextBox();
    this.label4 = new System.Windows.Forms.Label();
    this.label5 = new System.Windows.Forms.Label();
    this.textBoxPort = new System.Windows.Forms.TextBox();
    this.richTextBoxReceivedMsg = new System.Windows.Forms.RichTextBox();
    this.label2 = new System.Windows.Forms.Label();
    this.label3 = new System.Windows.Forms.Label();
    this.SuspendLayout();
    //
    // buttonClose
    //
    this.buttonClose.Location = new System.Drawing.Point(321, 232);
    this.buttonClose.Name = "buttonClose";
    this.buttonClose.Size = new System.Drawing.Size(88, 24);
    this.buttonClose.TabIndex = 11;
    this.buttonClose.Text = "Close";
    this.buttonClose.Click += new System.EventHandler(this.ButtonCloseClick);
    //
    // buttonSendMsg
    //
    this.buttonSendMsg.Location = new System.Drawing.Point(16, 192);
    this.buttonSendMsg.Name = "buttonSendMsg";
    this.buttonSendMsg.Size = new System.Drawing.Size(192, 24);
    this.buttonSendMsg.TabIndex = 7;
    this.buttonSendMsg.Text = "Send Message";
    this.buttonSendMsg.Click += new System.EventHandler(this.ButtonSendMsgClick);
    //
    // buttonStartListen
    //
    this.buttonStartListen.BackColor = System.Drawing.Color.Blue;
    this.buttonStartListen.Font = new System.Drawing.Font("Tahoma", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
    this.buttonStartListen.ForeColor = System.Drawing.Color.Yellow;
    this.buttonStartListen.Location = new System.Drawing.Point(227, 16);
    this.buttonStartListen.Name = "buttonStartListen";
    this.buttonStartListen.Size = new System.Drawing.Size(88, 40);
    this.buttonStartListen.TabIndex = 4;
    this.buttonStartListen.Text = "Start Listening";
    this.buttonStartListen.Click += new
System.EventHandler(this.ButtonStartListenClick);
    //
    // textBoxIP
    //
    this.textBoxIP.Location = new System.Drawing.Point(88, 16);
    this.textBoxIP.Name = "textBoxIP";
    this.textBoxIP.ReadOnly = true;
    this.textBoxIP.Size = new System.Drawing.Size(120, 20);

```

```

this.textBoxIP.TabIndex = 12;
this.textBoxIP.Text = "";
//
// richTextBoxSendMsg
//
this.richTextBoxSendMsg.Location = new System.Drawing.Point(16, 87);
this.richTextBoxSendMsg.Name = "richTextBoxSendMsg";
this.richTextBoxSendMsg.Size = new System.Drawing.Size(192, 104);
this.richTextBoxSendMsg.TabIndex = 6;
this.richTextBoxSendMsg.Text = "";
//
// label1
//
this.label1.Location = new System.Drawing.Point(16, 40);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(48, 16);
this.label1.TabIndex = 1;
this.label1.Text = "Port";
//
// buttonStopListen
//
this.buttonStopListen.BackColor = System.Drawing.Color.Red;
this.buttonStopListen.Font = new System.Drawing.Font("Tahoma", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte) 0));
this.buttonStopListen.ForeColor = System.Drawing.Color.Yellow;
this.buttonStopListen.Location = new System.Drawing.Point(321, 16);
this.buttonStopListen.Name = "buttonStopListen";
this.buttonStopListen.Size = new System.Drawing.Size(88, 40);
this.buttonStopListen.TabIndex = 5;
this.buttonStopListen.Text = "Stop Listening";
this.buttonStopListen.Click += new
System.EventHandler(this.ButtonStopListenClick);
//
// textBoxMsg
//
this.textBoxMsg.BackColor = System.Drawing.SystemColors.Control;
this.textBoxMsg.BorderStyle = System.Windows.Forms.BorderStyle.None;
this.textBoxMsg.ForeColor = System.Drawing.SystemColors.HotTrack;
this.textBoxMsg.Location = new System.Drawing.Point(120, 240);
this.textBoxMsg.Name = "textBoxMsg";
this.textBoxMsg.ReadOnly = true;
this.textBoxMsg.Size = new System.Drawing.Size(192, 13);
this.textBoxMsg.TabIndex = 14;
this.textBoxMsg.Text = "None";
//
// label4
//
this.label4.Location = new System.Drawing.Point(16, 71);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(192, 16);
this.label4.TabIndex = 8;
this.label4.Text = "Broadcast Message To Clients";
//
// label5
//
this.label5.Location = new System.Drawing.Point(217, 71);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(192, 16);
this.label5.TabIndex = 10;

```

```

this.label5.Text = "Message Received From Clients";
//
// textBoxPort
//
this.textBoxPort.Location = new System.Drawing.Point(88, 40);
this.textBoxPort.Name = "textBoxPort";
this.textBoxPort.Size = new System.Drawing.Size(40, 20);
this.textBoxPort.TabIndex = 0;
this.textBoxPort.Text = "8000";
//
// richTextBoxReceivedMsg
//
this.richTextBoxReceivedMsg.BackColor =
System.Drawing.SystemColors.InactiveCaptionText;
this.richTextBoxReceivedMsg.Location = new System.Drawing.Point(217, 87);
this.richTextBoxReceivedMsg.Name = "richTextBoxReceivedMsg";
this.richTextBoxReceivedMsg.ReadOnly = true;
this.richTextBoxReceivedMsg.Size = new System.Drawing.Size(192, 129);
this.richTextBoxReceivedMsg.TabIndex = 9;
this.richTextBoxReceivedMsg.Text = "";
//
// label2
//
this.label2.Location = new System.Drawing.Point(16, 16);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(56, 16);
this.label2.TabIndex = 2;
this.label2.Text = "Server IP";
//
// label3
//
this.label3.Location = new System.Drawing.Point(0, 240);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(112, 16);
this.label3.TabIndex = 13;
this.label3.Text = "Status Message:";
//
// SocketServer
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(424, 260);
this.Controls.Add(this.textBoxMsg);
this.Controls.Add(this.label3);
this.Controls.Add(this.textBoxIP);
this.Controls.Add(this.buttonClose);
this.Controls.Add(this.label5);
this.Controls.Add(this.richTextBoxReceivedMsg);
this.Controls.Add(this.label4);
this.Controls.Add(this.buttonSendMsg);
this.Controls.Add(this.richTextBoxSendMsg);
this.Controls.Add(this.buttonStopListen);
this.Controls.Add(this.buttonStartListen);
this.Controls.Add(this.label2);
this.Controls.Add(this.label1);
this.Controls.Add(this.textBoxPort);
this.Name = "SocketServer";
this.Text = "SocketServer";
this.ResumeLayout(false);
}

```



```

#endregion
void ButtonStartListenClick(object sender, System.EventArgs e)
{
    try
    {
        // Check the port value
        if(textBoxPort.Text == ""){
            MessageBox.Show("Please enter a Port Number");
            return;
        }
        string portStr = textBoxPort.Text;
        int port = System.Convert.ToInt32(portStr);
        // Create the listening socket...
        m_mainSocket = new Socket(AddressFamily.InterNetwork,
            SocketType.Stream,
            ProtocolType.Tcp);
        IPEndPoint ipLocal = new IPEndPoint (IPAddress.Any, port);
        // Bind to local IP Address...
        m_mainSocket.Bind( ipLocal );
        // Start listening...
        m_mainSocket.Listen (4);
        // Create the call back for any client connections...
        m_mainSocket.BeginAccept(new AsyncCallback (OnClientConnect), null);

        UpdateControls(true);
    }
    catch(SocketException se)
    {
        MessageBox.Show ( se.Message );
    }
}

private void UpdateControls( bool listening )
{
    buttonStartListen.Enabled    = !listening;
    buttonStopListen.Enabled    = listening;
}

// This is the call back function, which will be invoked when a client is connected
public void OnClientConnect(IAsyncResult asyn)
{
    try
    {
        // Here we complete/end the BeginAccept() asynchronous call
        // by calling EndAccept() - which returns the reference to
        // a new Socket object
        m_workerSocket[m_clientCount] = m_mainSocket.EndAccept (asyn);
        // Let the worker Socket do the further processing for the
        // just connected client
        WaitForData(m_workerSocket[m_clientCount]);
        // Now increment the client count
        ++m_clientCount;
        // Display this client connection as a status message on the GUI

        String str = String.Format("Client # {0} connected", m_clientCount);
        textBoxMsg.Text = str;

        // Since the main Socket is now free, it can go back and wait for
        // other clients who are attempting to connect
    }
}

```

```

        m_mainSocket.BeginAccept(new AsyncCallback ( OnClientConnect ), null);
    }
    catch(ObjectDisposedException)
    {
        System.Diagnostics.Debugger.Log(0, "1", "\n OnClientConnection: Socket
has been closed\n");
    }
    catch(SocketException se)
    {
        MessageBox.Show ( se.Message );
    }
}
public class SocketPacket
{
    public System.Net.Sockets.Socket m_currentSocket;
    public byte[] dataBuffer = new byte[1];
}
// Start waiting for data from the client
public void WaitForData(System.Net.Sockets.Socket soc)
{
    try
    {
        if ( pfnWorkerCallBack == null ){
            // Specify the call back function which is to be
            // invoked when there is any write activity by the
            // connected client
            pfnWorkerCallBack = new AsyncCallback (OnDataReceived);
        }
        SocketPacket theSocPkt = new SocketPacket ();
        theSocPkt.m_currentSocket = soc;
        // Start receiving any data written by the connected client
        // asynchronously
        soc.BeginReceive (theSocPkt.dataBuffer, 0,
            theSocPkt.dataBuffer.Length,
            SocketFlags.None,
            pfnWorkerCallBack,
            theSocPkt);
    }
    catch(SocketException se)
    {
        MessageBox.Show (se.Message );
    }
}
// This the call back function which will be invoked when the socket
// detects any client writing of data on the stream
public void OnDataReceived(IAsyncResult asyn)
{
    try
    {
        SocketPacket socketData = (SocketPacket)asyn.AsyncState ;

        int iRx = 0 ;
        // Complete the BeginReceive() asynchronous call by EndReceive()
method
        // which will return the number of characters written to the stream
        // by the client
        iRx = socketData.m_currentSocket.EndReceive (asyn);
    }
}

```

```

        char[] chars = new char[iRx + 1];
        System.Text.Decoder d = System.Text.Encoding.UTF8.GetDecoder();
        int charLen = d.GetChars(socketData.dataBuffer,
                                0, iRx, chars, 0);
        System.String szData = new System.String(chars);
        richTextBoxReceivedMsg.AppendText(szData);

        // Continue the waiting for data on the Socket
        WaitForData( socketData.m_currentSocket );
    }
    catch (ObjectDisposedException )
    {
        System.Diagnostics.Debugger.Log(0, "1", "\nOnDataReceived: Socket has
been closed\n");
    }
    catch(SocketException se)
    {
        MessageBox.Show (se.Message );
    }
}

void ButtonSendMsgClick(object sender, System.EventArgs e)
{
    try
    {
        Object objData = richTextBoxSendMsg.Text;
        byte[] byData = System.Text.Encoding.ASCII.GetBytes(objData.ToString
());

        for(int i = 0; i < m_clientCount; i++){
            if(m_workerSocket[i] != null){
                if(m_workerSocket[i].Connected){
                    m_workerSocket[i].Send (byData);
                }
            }
        }

    }
    catch(SocketException se)
    {
        MessageBox.Show (se.Message );
    }
}

void ButtonStopListenClick(object sender, System.EventArgs e)
{
    CloseSockets();
    UpdateControls(false);
}

String GetIP()
{
    String strHostName = Dns.GetHostName();

    // Find host by name
    IPHostEntry iphostentry = Dns.GetHostByName(strHostName);

    // Grab the first IP addresses
    String IPStr = "";
    foreach(IPAddress ipaddress in iphostentry.AddressList){
        IPStr = ipaddress.ToString();
    }
}

```

```

        return IPStr;
    }
    return IPStr;
}
void ButtonCloseClick(object sender, System.EventArgs e)
{
    CloseSockets();
    Close();
}
void CloseSockets()
{
    if(m_mainSocket != null){
        m_mainSocket.Close();
    }
    for(int i = 0; i < m_clientCount; i++){
        if(m_workerSocket[i] != null){
            m_workerSocket[i].Close();
            m_workerSocket[i] = null;
        }
    }
}
}
}
}

```

Source code for Socket Clients

```
using System;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;

namespace DefaultNamespace
{
    /// <summary>
    /// Description of SocketClient.
    /// </summary>
    public class SocketClient : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button buttonDisconnect;
        private System.Windows.Forms.TextBox textBoxIP;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Button buttonConnect;
        private System.Windows.Forms.TextBox textBoxPort;
        private System.Windows.Forms.RichTextBox richTextRxMessage;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TextBox textBoxConnectStatus;
        private System.Windows.Forms.RichTextBox richTextTxMessage;
        private System.Windows.Forms.Button buttonSendMessage;
        private System.Windows.Forms.Button buttonClose;

        byte[] m_dataBuffer = new byte [10];
        IAsyncResult m_result;
        public AsyncCallback m_pfnCallBack ;
        public Socket m_clientSocket;

        public SocketClient()
        {
            //
            // The InitializeComponent() call is required for Windows Forms designer
            support.
            //
            InitializeComponent();

            textBoxIP.Text = GetIP();
        }

        [STAThread]
        public static void Main(string[] args)
        {
            Application.Run(new SocketClient());
        }

        #region Windows Forms Designer generated code
        /// <summary>
        /// This method is required for Windows Forms designer support.

```

```

designer might
    /// Do not change the method contents inside the source code editor. The Forms
    /// not be able to load this method if it was changed manually.
    /// </summary>
private void InitializeComponent() {
    this.buttonClose = new System.Windows.Forms.Button();
    this.buttonSendMessage = new System.Windows.Forms.Button();
    this.richTextTxMessage = new System.Windows.Forms.RichTextBox();
    this.textBoxConnectStatus = new System.Windows.Forms.TextBox();
    this.label4 = new System.Windows.Forms.Label();
    this.richTextRxMessage = new System.Windows.Forms.RichTextBox();
    this.textBoxPort = new System.Windows.Forms.TextBox();
    this.buttonConnect = new System.Windows.Forms.Button();
    this.label5 = new System.Windows.Forms.Label();
    this.textBoxIP = new System.Windows.Forms.TextBox();
    this.buttonDisconnect = new System.Windows.Forms.Button();
    this.label11 = new System.Windows.Forms.Label();
    this.label12 = new System.Windows.Forms.Label();
    this.label13 = new System.Windows.Forms.Label();
    this.SuspendLayout();
    //
    // buttonClose
    //
    this.buttonClose.Location = new System.Drawing.Point(400, 216);
    this.buttonClose.Name = "buttonClose";
    this.buttonClose.Size = new System.Drawing.Size(104, 24);
    this.buttonClose.TabIndex = 11;
    this.buttonClose.Text = "Close";
    this.buttonClose.Click += new System.EventHandler(this.ButtonCloseClick);
    //
    // buttonSendMessage
    //
    this.buttonSendMessage.Location = new System.Drawing.Point(8, 184);
    this.buttonSendMessage.Name = "buttonSendMessage";
    this.buttonSendMessage.Size = new System.Drawing.Size(240, 24);
    this.buttonSendMessage.TabIndex = 14;
    this.buttonSendMessage.Text = "Send Message";
    this.buttonSendMessage.Click += new
System.EventHandler(this.ButtonSendMessageClick);
    //
    // richTextTxMessage
    //
    this.richTextTxMessage.Location = new System.Drawing.Point(8, 80);
    this.richTextTxMessage.Name = "richTextTxMessage";
    this.richTextTxMessage.Size = new System.Drawing.Size(240, 96);
    this.richTextTxMessage.TabIndex = 2;
    this.richTextTxMessage.Text = "";
    //
    // textBoxConnectStatus
    //
    this.textBoxConnectStatus.BackColor = System.Drawing.SystemColors.Control;
    this.textBoxConnectStatus.BorderStyle =
System.Windows.Forms.BorderStyle.None;
    this.textBoxConnectStatus.ForeColor = System.Drawing.SystemColors.HotTrack;
    this.textBoxConnectStatus.Location = new System.Drawing.Point(128, 224);
    this.textBoxConnectStatus.Name = "textBoxConnectStatus";
    this.textBoxConnectStatus.ReadOnly = true;
    this.textBoxConnectStatus.Size = new System.Drawing.Size(240, 13);
    this.textBoxConnectStatus.TabIndex = 10;

```

```

this.textBoxConnectStatus.Text = "Not Connected";
//
// label4
//
this.label4.Location = new System.Drawing.Point(8, 64);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(120, 16);
this.label4.TabIndex = 9;
this.label4.Text = "Message To Server";
//
// richTextRxMessage
//
this.richTextRxMessage.BackColor =
System.Drawing.SystemColors.InactiveCaptionText;
this.richTextRxMessage.Location = new System.Drawing.Point(256, 80);
this.richTextRxMessage.Name = "richTextRxMessage";
this.richTextRxMessage.ReadOnly = true;
this.richTextRxMessage.Size = new System.Drawing.Size(248, 128);
this.richTextRxMessage.TabIndex = 1;
this.richTextRxMessage.Text = "";
//
// textBoxPort
//
this.textBoxPort.Location = new System.Drawing.Point(112, 31);
this.textBoxPort.Name = "textBoxPort";
this.textBoxPort.Size = new System.Drawing.Size(48, 20);
this.textBoxPort.TabIndex = 6;
this.textBoxPort.Text = "8000";
//
// buttonConnect
//
this.buttonConnect.BackColor = System.Drawing.SystemColors.HotTrack;
this.buttonConnect.Font = new System.Drawing.Font("Tahoma", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.buttonConnect.ForeColor = System.Drawing.Color.Yellow;
this.buttonConnect.Location = new System.Drawing.Point(344, 8);
this.buttonConnect.Name = "buttonConnect";
this.buttonConnect.Size = new System.Drawing.Size(72, 48);
this.buttonConnect.TabIndex = 7;
this.buttonConnect.Text = "Connect To Server";
this.buttonConnect.Click += new System.EventHandler(this.ButtonConnectClick);
//
// label5
//
this.label5.Location = new System.Drawing.Point(0, 224);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(104, 16);
this.label5.TabIndex = 13;
this.label5.Text = "Connection Status";
//
// textBoxIP
//
this.textBoxIP.Location = new System.Drawing.Point(112, 8);
this.textBoxIP.Name = "textBoxIP";
this.textBoxIP.Size = new System.Drawing.Size(152, 20);
this.textBoxIP.TabIndex = 3;
this.textBoxIP.Text = "";
//
// buttonDisconnect

```

```

//
this.buttonDisconnect.BackColor = System.Drawing.Color.Red;
this.buttonDisconnect.Font = new System.Drawing.Font("Tahoma", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.buttonDisconnect.ForeColor = System.Drawing.Color.Yellow;
this.buttonDisconnect.Location = new System.Drawing.Point(432, 8);
this.buttonDisconnect.Name = "buttonDisconnect";
this.buttonDisconnect.Size = new System.Drawing.Size(72, 48);
this.buttonDisconnect.TabIndex = 15;
this.buttonDisconnect.Text = "Disconnect From Server";
this.buttonDisconnect.Click += new
System.EventHandler(this.ButtonDisconnectClick);
//
// label1
//
this.label1.Location = new System.Drawing.Point(8, 8);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(96, 16);
this.label1.TabIndex = 4;
this.label1.Text = "Server IP Address";
//
// label2
//
this.label2.Location = new System.Drawing.Point(8, 33);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(64, 16);
this.label2.TabIndex = 5;
this.label2.Text = "Server Port";
//
// label3
//
this.label3.Location = new System.Drawing.Point(256, 64);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(192, 16);
this.label3.TabIndex = 8;
this.label3.Text = "Message From Server";
//
// SocketClient
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(512, 244);
this.Controls.Add(this.buttonDisconnect);
this.Controls.Add(this.buttonSendMessage);
this.Controls.Add(this.label5);
this.Controls.Add(this.buttonClose);
this.Controls.Add(this.textBoxConnectStatus);
this.Controls.Add(this.label4);
this.Controls.Add(this.label3);
this.Controls.Add(this.buttonConnect);
this.Controls.Add(this.textBoxPort);
this.Controls.Add(this.label2);
this.Controls.Add(this.label1);
this.Controls.Add(this.textBoxIP);
this.Controls.Add(this.richTextTxMessage);
this.Controls.Add(this.richTextRxMessage);
this.Name = "SocketClient";
this.Text = "Socket Client";
this.ResumeLayout(false);
}

```



```

#endregion
void ButtonCloseClick(object sender, System.EventArgs e)
{
    if ( m_clientSocket != null )
    {
        m_clientSocket.Close ();
        m_clientSocket = null;
    }
    Close();
}

void ButtonConnectClick(object sender, System.EventArgs e)
{
    // See if we have text on the IP and Port text fields
    if(textBoxIP.Text == "" || textBoxPort.Text == ""){
        MessageBox.Show("IP Address and Port Number are required to connect
to the Server\n");
        return;
    }
    try
    {
        UpdateControls(false);
        // Create the socket instance
        m_clientSocket = new Socket (AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp );

        // Get the remote IP address
        IPAddress ip = IPAddress.Parse (textBoxIP.Text);
        int iPortNo = System.Convert.ToInt16 ( textBoxPort.Text);
        // Create the end point
        IPEndPoint ipEnd = new IPEndPoint (ip, iPortNo);
        // Connect to the remote host
        m_clientSocket.Connect ( ipEnd );
        if(m_clientSocket.Connected) {

            UpdateControls(true);
            //Wait for data asynchronously
            WaitForData();
        }
    }
    catch(SocketException se)
    {
        string str;
        str = "\nConnection failed, is the server running?\n" + se.Message;
        MessageBox.Show (str);
        UpdateControls(false);
    }
}

void ButtonSendMessageClick(object sender, System.EventArgs e)
{
    try
    {
        Object objData = richTextBoxMessage.Text;
        byte[] byData = System.Text.Encoding.ASCII.GetBytes(objData.ToString
());

        if(m_clientSocket != null){
            m_clientSocket.Send (byData);
        }
    }
}

```

```

        catch(SocketException se)
        {
            MessageBox.Show (se.Message );
        }
    }
    public void WaitForData()
    {
        try
        {
            if ( m_pfnCallBack == null )
            {
                m_pfnCallBack = new AsyncCallback (OnDataReceived);
            }
            SocketPacket theSocPkt = new SocketPacket ();
            theSocPkt.thisSocket = m_clientSocket;
            // Start listening to the data asynchronously
            m_result = m_clientSocket.BeginReceive (theSocPkt.dataBuffer,
theSocPkt.dataBuffer.Length,
0,
SocketFlags.None,
m_pfnCallBack,
theSocPkt);
        }
        catch(SocketException se)
        {
            MessageBox.Show (se.Message );
        }
    }
}
public class SocketPacket
{
    public System.Net.Sockets.Socket thisSocket;
    public byte[] dataBuffer = new byte[1];
}

public void OnDataReceived(IAsyncResult asyn)
{
    try
    {
        SocketPacket theSockId = (SocketPacket)asyn.AsyncState ;
        int iRx = theSockId.thisSocket.EndReceive (asyn);
        char[] chars = new char[iRx + 1];
        System.Text.Decoder d = System.Text.Encoding.UTF8.GetDecoder();
        int charLen = d.GetChars(theSockId.dataBuffer, 0, iRx, chars, 0);
        System.String szData = new System.String(chars);
        richTextRxMessage.Text = richTextRxMessage.Text + szData;
        WaitForData();
    }
    catch (ObjectDisposedException )
    {
        System.Diagnostics.Debugger.Log(0,"1","\\nOnDataReceived: Socket has
been closed\\n");
    }
    catch(SocketException se)
    {
        MessageBox.Show (se.Message );
    }
}
private void UpdateControls( bool connected )

```

```

    {
        buttonConnect.Enabled = !connected;
        buttonDisconnect.Enabled = connected;
        string connectStatus = connected? "Connected" : "Not Connected";
        textBoxConnectStatus.Text = connectStatus;
    }
    void ButtonDisconnectClick(object sender, System.EventArgs e)
    {
        if ( m_clientSocket != null )
        {
            m_clientSocket.Close();
            m_clientSocket = null;
            UpdateControls(false);
        }
    }
}
//-----
// This is a helper function used (for convenience) to
// get the IP address of the local machine
//-----
String GetIP()
{
    String strHostName = Dns.GetHostName();

    // Find host by name
    IPHostEntry iphostentry = Dns.GetHostByName(strHostName);

    // Grab the first IP addresses
    String IPStr = "";
    foreach(IPAddress ipaddress in iphostentry.AddressList){
        IPStr = ipaddress.ToString();
        return IPStr;
    }
    return IPStr;
}
}
}

```

APPENDIX III

Flow Chart for Ladder Diagram

