Ethernet Communication for Control & Monitoring of Pole Top Remote Terminal Unit (RTU)

A Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Electronics Instrumentation and Control

Submitted by:

Prachi Sharma
(801151018)

UNDER THE GUIDANCE OF

Mr. Mandeep Singh
Asst. Professor
Electrical and Instrumentation Department
Thapar University, Patiala

Dr. Suraj Kumar Pardeshi
Sr. Manager - Technology
CG Global R&D Centre
Crompton Greaves Ltd, Mumbai

ELECTRICAL AND INSTRUMENTATION ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
(Established under the section 3 of UGC act, 1956)
PATIALA – 147004

JUNE 2013

Declaration

I hereby declare that the work which is being presented in this dissertation entitles as, **'Ethernet Communication for Control and Monitoring of Remote Terminal Unit'** in partial fulfillment of the requirements for the award of degree of Master of Engineering in Electronics Instrumentation and Control Engineering at Thapar University, Patiala is an authentic record of my own work carried out under the supervision and guidance of **Dr. Suraj Kumar Pardeshi** and **Mr. M.D Singh** and refer to the other researcher's work which are duly listed in the reference section.

The matter embodied in this thesis has not been submitted to any other University/Institute for the award of any Degree or Discipline.

Prachi Sharma

Date: 9/7/13 (Prachi Sharma)

Place: PATIALA (801151018)

It is certified that above statement made by the candidate is correct and true to the best of our knowledge.

9/7/13

(Mr. M.D. Singh)

Asst. Professor

Electrical and Instrumentation Dept.

Thapar University, Patiala

(Dr. Suraj Kumar Pardeshi)

Sr. Manager-technology

Global R&D

Crompton Greaves Ltd., Mumbai

S. Ghosh

(Dr. Samarjit Ghosh)

HOD

Electrical and Instrumentation Dept.

Thapar University, Patiala

(Dr. S.K. Mohapatra)

Dean of Academic Affairs

Thapar University,

Patiala

Acknowledgements

At the end of my dissertation, I want to thank God for always being with me and will pray him to always shower his blessings on me. I would like to express my sincere thanks to many people who have contributed to the fulfillment of my dissertation. First of all, I am extremely grateful to my dissertation guides, ***Dr. Suraj Kumar Pardeshi,*** *Sr. Manager-Technology, Crompton Greaves Ltd., Mumbai and* ***Mr. Mandeep Singh****, Assistant Professor, Department of Electrical & Instrumentation Engineering, Thapar University, Patiala, for their valuable guidance, scholarly inputs and consistent encouragement I received throughout the dissertation work.*

I am very thankful to ***Smarajit Ghosh****, Professor and Head, Department of Electrical & Instrumentation Engineering, Thapar University, Patiala, whose support always provoked me to do always better. I also want to thank* ***Dr. R.S. Kaler****, Dean of Resource Planning and Generation, Thapar University, Patiala for his support and guidance.*

Hearty thanks to all those who supported me, from Electronics Design Centre (EDC), Crompton Greaves Ltd., Mumbai that I may have called upon for assistance, as their suggestions have helped in the development of this dissertation.

A very special thanks to my parents ***Dr. M.P Sharma****, Associate Professor, IIT Roorkee, Roorkee and* ***Smt. Manju Sharma,*** *and my sister* ***Ms. Priyanka Sharma,*** *who always supported me in my good/bad times since the day of commencement of my training till its end. Without them I could do nothing. I owe my dissertation to them.*

Last but not the least, I want to thank my friends ***Ashish Maheshwari****,* ***Paramjeet Shahi****,* ***Surabhi Gangwar****,* ***Amandeep Cheema****,* ***Mohit Kumar*** *and* ***Rajeev Kumar*** *who helped me in every step of my dissertation and always motivated me to do the best.*

Place: Thapar University, Patiala (Prachi Sharma)

Date: (Roll No. 801151018)

Abstract

Remote terminal unit (RTU) is a microcontroller based device that communicates to a Distributed Control System (DCS) or supervisory control and data acquisition (SCADA) system by transmitting data to a master controller in a power distribution system. An RTU monitors the field's digital and analog parameters and transmits these to the central monitoring station. It contains setup software to connect data input streams to data output streams, defines communication protocols, and troubleshoots the problems. In this dissertation, the design and study of various sections of RTU has been explained. In RTU, exchange of information is very much essential to acquire complete information for further transmission to substations. Therefore, the main emphasis is put on Ethernet communication and the TCP/IP (Transmission Control Protocol/Internet Protocol) stack in embedded systems. The exchange of information takes place with the help of some communication media and protocols. A communication protocol is a set of rules that computers must follow to communicate with each other. The TCP/IP protocol suite refers to several separate protocols that computers make use to transfer the data across the networks. Two most important protocols in its suite: Transmission Control Protocol (TCP) and Internet Protocol (IP) are widely used because of their ability to switch packets from all shapes and sizes and varieties of networks. However, other protocols like ARP, HTTP and ICMP of this suite have also been discussed in the dissertation. A comparison between fiber optic and Ethernet has been made to provide the understanding of the best type of device and its optimal use. Moreover, the hardware and software implementation of this communication section has also been studied.

The main objective of the dissertation has been to implement TCP/IP stack to make interaction between Remote Terminal Unit and the web server. In this work, an Ethernet module using PIC microcontroller and micrel's Ethernet switch has been designed where TCP/IP stack has been used to make interactions. The analysis of network protocols and their interaction with each other has been carried out through simulations using Wireshark network protocol analyzer. The simulation has shown that transmission and reception of the packets are recognized by the sequence and the acknowledgement number. Also, analog section of the RTU is tested for its accuracy for all types of inputs. The future scope of the work includes the implementation of the fiber optic communication, used in the present RTU, successfully.

TABLE OF CONTENTS

PARTICULARS **PAGE NO.**

DECLARATION……i

ACKNOWLEDGEMENT……..ii

ABSTRACT……...iii

TABLE OF CONTENTS……...iv

LIST OF FIGURES……viii

LIST OF TABLES……..... x

RELATED PUBLICATIONS…….....xi

CHAPTERS

1 INTRODUCTION..... 1

1.1 INTRODUCTION 1

1.2 LITERATURE REVIEW 2

1.3 PREVIOUS DEVELOPED PRODUCTS 4

1.3.1 Control Wave® Micro Hybrid RTU/PLC 5

1.3.2 Siemens TG5700 RTU..... 5

1.3.3 Motorola's ACE3600 RTU..... 6

1.3.4 Kalkitech Sync 2100 series intelligent RTU..... 6

1.3.5 Comparison between various RTUs 7

1.4 RESEARCH GAPS 8

1.5 OBJECTIVE OF THESIS 8

1.6 OUTLINE OF THE DISSERTATION 8

2 POLE TOP REMOTE TERMINAL UNIT (RTU)..... 10

2.1 REMOTE TERMINAL UNIT 10

2.1.1 Transmission of Power to customers 11

2.1.2 Justification for Voltage Transformation..... 13

2.2 ARCHITECTURE OF REMOTE TERMINAL UNIT 13

2.3 BLOCK DIAGRAM OF REMOTE TERMINAL UNIT 14

2.4 ARM9 CONTROLLER 15
2.4.1 Features 15
2.5 DESCRIPTION OF RTU SECTIONS 16
2.5.1 Digital Inputs (DI) Section 16
2.5.2 Digital Outputs (DO) Section 18
2.5.3 Analog Inputs (AI) Section 19
2.5.4 Communications Section 22
3 ETHERNET COMMUNICATION 24
3.1 INTRODUCTION 24
3.2 PRINCIPLE OF OPERATION 24
3.3 BINARY EXPONENTIAL BACK OFF ALGORITHM 25
3.4 DIFFERENCE BETWEEN A SWITCH AND A HUB 25
3.5 ETHERNET FRAME FORMAT 26
3.5.1 MAC (Media Access Controller) Addresses 27
3.6 ETHERNET CABLING 28
3.7 VARIETIES OF ETHERNET 29
3.8 SIGNIFICANCE OF ETHERNET COMMUNICATION 29
3.9 ETHERNET MODULE 30
3.9.1 PIC32MX795F512H Micro Controller 31
3.9.2 Ethernet Switch 32
3.10 FIBER OPTIC COMMUNICATION 45
3.10.1 Serial Signal Transmission through Fiber equipment 47
3.11 FIBER VS. COPPER – CURRENT AND FUTURE CO-EXISTENCE 48
4 STUDY OF TCP/IP STACK 50
4.1 INTRODUCTION 50
4.2 NETWORK INTERFACE LAYER 50
4.2.1 Ethernet 50
4.2.2 Address Resolution Protocol 51
4.3 NETWORK LAYER PROTOCOLS 52
4.3.1 Internet Protocol (IP) 52

4.3.2 Internet Control Message Protocol (ICMP) 56
4.4 TRANSPORT LAYER PROTOCOLS 57
4.4.1 Transmission Control Protocol (TCP) 58
4.5 APPLICATION LAYER PROTOCOLS 61
4.5.1 Hypertext Transfer Protocol (HTTP) 61
4.6 FRAME/PACKET ENCAPSULATION 66
4.7 WEB PAGES 68
4.7.1 Static content web page 68
4.7.2 Dynamic content web page 68
5 IMPLEMENTATION OF HARDWARE AND SOFTWARE 69
5.1 IMPLEMENTATION OF HARDWARE 69
5.1.1 Ethernet Hardware Schematic 69
5.1.2 Understanding of circuit with the help of block diagram 71
5.2 IMPLEMENTATION OF SOFTWARE 72
5.3 IP (INTERNET PROTOCOL) FLOW DIAGRAM 73
5.4 HTTP (HYPER TEXT TERMINAL PROTOCOL) 78
5.4.1 HTTPExecuteGet Function 80
5.4.2 HTTPGetArg 80
5.4.3 HTTPGetROMArg 81
5.5 DYNAMIC WEB PAGE 81
6 METHODOLOGY 85
6.1 TEST 1: TESTING OF AUXILIARY INPUTS OF ANALOG SECTION 85
6.1.1 Channel 1 Measurements 85
6.2 TEST2: TESTING OF CURRENT INPUTS WITH THE HELP OF CURRENT TRANSFORMER 86
6.2.1 Measurement of 1A current 88
6.2.2 Measurement of 5A current 88
6.3 TEST 3: TESTING OF CURRENT INPUTS USING POTENTIAL TRANSFORMER (PT) 89
6.4 TESTING OF DC1 AND DC2 89
6.4.1 DC1 Measurements 90
6.4.2 DC2 Measurements 90

6.5 ANALYZING NETWORK PROTOCOLS 91

7 TESTING AND SIMULATION RESULTS 93

7.1 TEST 1: TESTING RESULTS OF AUXILIARY INPUTS OF ANALOG SECTION 93

7.1.1 Channel 1 Measurements 93

7.1.2 Channel 2 Measurements 95

7.1.3 Channel 3 Measurements 96

7.2 TEST2: TESTING RESULTS OF CURRENT INPUTS WITH THE HELP OF CURRENT TRANSFORMER 97

7.2.1 Measurement of 1A current 98

7.2.2 Measurement of 5A current 100

7.3 TEST 3: TESTING RESULTS OF VOLTAGE INPUTS WITH THE HELP OF POTENTIAL TRANSFORMER (PT). 101

7.4 TESTING RESULTS OF DC1 AND DC2 CHANNELS 102

7.4.1 DC1 Measurements 103

7.4.2 DC2 Measurements 105

7.5 SIMULATIONS OVER NETWORK PROTOCOLS 106

7.5.1 Simulation-1: ARP (Address Resolution Protocol) 107

7.5.2 Simulation-2: TCP (Transmission Control Protocol) 108

7.5.3 Simulation-3: HTTP (Hyper Text Terminal Protocol) 113

7.5.4 Simulation-4: Packets during interaction of web server with the board 114

8 CONCLUSION AND FUTURE SCOPE 115

8.1 CONCLUSION 115

8.2 FUTURE SCOPE 116

4 REFERENCES 117

LIST OF FIGURES

FIGURE 1.1 CONTROL WAVE MICRO HYBRID RTU/PLC AND SIEMENS TG5700 RTU 5
FIGURE 1.2 MOTOROLA'S ACE3600 RTU 6
FIGURE 1.3KALKITECH SYNC 2100 SERIES RTU 7
FIGURE 2.1 POLE TOP RTU 10
FIGURE 2.2 TYPICAL POWER TRANSFER AND DISTRIBUTION SCENARIO 12
FIGURE 2.3ARCHITECTURAL PLACEMENT DIAGRAM OF REMOTE TERMINAL UNIT 13
FIGURE 2.4 BLOCK DIAGRAM OF REMOTE TERMINAL UNIT 14
FIGURE 2.5 BLOCK DIAGRAM OF DI SECTION 17
FIGURE 2.6 SENSOR WIRING AND EQUIVALENT CIRCUIT 17
FIGURE 2.7 DIGITAL OUTPUTS SECTION 19
FIGURE 2.8 BLOCK DIAGRAM OF OVERALL ANALOG SECTION 20
FIGURE 2.9 UART-0 TO RS-232 LINK 22
FIGURE 2.10 UART-0 TO RS-485 LINK 23
FIGURE 2.11 USB PIN DIAGRAM 23
FIGURE 3.1 ETHERNET BY BOB METCALFE 24
FIGURE 3.2 ETHERNET FRAME FORMAT 26
FIGURE 3.3 MAC ADDRESSES 28
FIGURE 3.4 ETHERNET COMMUNICATION 31
FIGURE 3.5 100 MB/S LAYER DEFINITIONS 32
FIGURE 3.6 UNMANAGED SWITCH 34
FIGURE 3.7 MANAGED SWITCH 34
FIGURE 3.8 KSZ8863FLL/MLL/RLL FUNCTIONAL BLOCK DIAGRAM 35
FIGURE 3.9 SEQUENCE OF STEPS AT TRANMITTER SIDE 38
FIGURE 3.10 NRZ CODING 40
FIGURE 3.11 NRZI CODING 40
FIGURE 3.12 SEQUENCE OF STEPS AT RECEIVER SIDE 42
FIGURE 3.13 100BASE-TX BLW EVENT 43
FIGURE 3.14 MANCHESTER CODING 45
FIGURE 3.15 HFBR1414 AND HFBR 2412 47
FIGURE 3.16 CIRCUIT DIAGRAM FOR FIBER OPTIC TRANSMISSION 48

FIGURE 4.1 TCP/IP STACK 50
FIGURE 4.2 ARP: REQUEST/REPLY PACKET 51
FIGURE 4.3 IP: PACKET DELIVERY MODES 54
FIGURE 4.4 IP: FORMAT OF AN IP DATAGRAM HEADER 55
FIGURE 4.5 ICMP: MESSAGE FORMAT 57
FIGURE 4.6 TCP: SEGMENT FORMAT 59
FIGURE 4.7 TCP: CONNECTION ESTABLISHMENT 60
FIGURE 4.8 PROCESS BEHIND OPENING OF VARIOUS SITES 61
FIGURE 4.9 HTTP MESSAGE BODY 62
FIGURE 4.10 BODY OF HTTP REQUEST MESSAGE 62
FIGURE 4.11 AN EXAMPLE OF AN HTTP REQUEST MESSAGE 63
FIGURE 4.12 BODY OF HTTP RESPONSE MESSAGE 64
FIGURE 4.13 AN EXAMPLE OF AN HTTP REQUEST MESSAGE 65
FIGURE 4.14 DATA ENCAPSULATION EXAMPLE 67
FIGURE 5.1 PIC32MX795F512H MICROCONTROLLER 69
FIGURE 5.2 KSZ8863FLL/RLL/MLL ETHERNET SWITCH 70
FIGURE 5.3 POWER SUPPLY SECTION 70
FIGURE 5.4 BLOCK DIAGRAM OF THE IMPLEMENTED DESIGN 71
FIGURE 5.5 FLOW OF ETHERNET FRAME THROUGH VARIOUS PROTOCOLS 72
FIGURE 5.6 IP FLOW CHART 74
FIGURE 5.7 APPLICATION OF WEB FORMS 78
FIGURE 5.8 WEB PAGE FOR REMOTE TERMINAL UNIT 81
FIGURE 6.1 PROCESSING OF AUXILIARY INPUTS 85
FIGURE 6.3 CIRCUIT DIAGRAM FOR CURRENT TRANSFORMER 87
FIGURE 6.4 BLOCK DIAGRAM REPRESENTING PROCESSING OF VOLTAGE INPUTS 89
FIGURE 6.5 BLOCK DIAGRAM REPRESENTING PROCESSING OF DC INPUTS 90
FIGURE 6.6 WIRE SHARK AREAS 91
FIGURE 7.1 GRAPH BETWEEN I/P CURRENT & CALCULATED CURRENT FOR CHANNEL 1 94
FIGURE 7.2 GRAPH BETWEEN I/P CURRENT & CALCULATED CURRENT FOR CHANNEL 2 95
FIGURE 7.3 GRAPH BETWEEN I/P CURRENT & CALCULATED CURRENT FOR CHANNEL 3 97

LIST OF TABLES

TABLE 1.1 COMPARISON BETWEEN VARIOUS RTUS 7
TABLE 3.1 VARIETIES OF ETHERNET 29
TABLE 3.2 MII SIGNALS 36
TABLE 3.3 RMII SIGNALS 37
TABLE 3.4 4B/5B CODING 39
TABLE 3.5 LIST OF TRANSCEIVERS 46
TABLE 5.1: POSSIBLE VALUES FOR *TYPE [OUT]* PARAMETER 75
TABLE 5.2: POSSIBLE VALUES FOR PROTOCOL [OUT] PARAMETER 75
TABLE 5.3: POSSIBLE VALUES FOR PROTOCOL [IN] PARAMETER 77
TABLE 7.1 AUXILIARY INPUT CHANNEL 1 CALCULATIONS 93
TABLE 7.2 AUXILIARY INPUT CHANNEL 2 CALCULATIONS 95
TABLE 7.3 AUXILIARY INPUT CHANNEL 3 CALCULATIONS 96
TABLE 7.4 CALCULATIONS FOR 1A CURRENT 98
TABLE 7.5 CALCULATIONS FOR 5A CURRENT 100
TABLE 7.6 PT TESTING 101
TABLE 7.7 DC1MEASUREMENTS 103
TABLE 7.8 DC2 MEASUREMENTS 105

RELATED PUBLICATIONS

International Journal

[1] Prachi Sharma, Suraj Pardeshi, Rohit Kumar Arora, Mandeep Singh, ***"A Review of the Development in the Field of Fiber Optic Communication Systems"***, International Journal of Emerging Technology and Advanced Engineering (IJETAE), 3(5), May 2013, 113-119.

[2] Prachi Sharma, Rohit Kumar Arora, Suraj Pardeshi, Mandeep Singh, ***"Fiber Optic Communications: An Overview"***, International Journal of Emerging Technology and Advanced Engineering (IJETAE), 3(5), May 2013, 474-479.

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the present scenario, the world is moving towards automated systems. RTUs, in a system, gather information and are responsible for sending/receiving and executing the commands issued from the control center. Also, the industry is discovering the hazards of proprietary communications systems, (i.e., high cost, inflexibility, lack of support for older components, etc), so Ethernet is becoming the preferred communications vehicle within substations [1]. Better communications technology is an important component for implementing the utility systems that are going to cost-effectively meet today's needs and is also be easily upgradable as demand and technologies change.

To design hardware, which is able to interact via internet, is a challenging task in the design of any automation system. In the present design, the processor used is PIC32MX795F512H. This design is able to receive and transmit data and is able to respond accordingly. It also enables a user to communicate with the hardware via any web browser. There can be various applications of this kind of system, specially, related with automation and surveillance system. The investigated and developed design in this dissertation is based on automation.

A new approach is investigated and developed, which is able to measure, sense and respond according to various sensors attached to it and at the same time, the data measured can be distributed in various geographic locations via Ethernet network. Earlier, the system can only perform controlling action but cannot get the state of the system, but in this system, the current state of various devices like sensors etc. can be read. All the controls and the present readings of the attached sensors can be viewed in any geographic location with the internet facility [2].There is no requirement of any kind of software installation in mobile or computer to control or monitor this automation system.

There are many models used presently for communication. One of them is ISO/OSI (Open System Interconnect) model which was developed to standardize computer communication and is the first and most comprehensive networking model. It defines the seven layers of functionality and abstraction. Unfortunately, the model does not contain concrete protocols and the applicants did not have anything to follow. The model tries to solve almost all possible questions and uses interconnected computers. Therefore, it was far too complex to implement and spread protocols respecting it. However, a different approach succeeded- the TCP/IP model. Its model was based on the simple and concrete protocols which were really needed to build up. Its bottom–up approach leads to its current world domination. This protocol family is called the TCP/IP protocol suite, where the IP stands for the Inter–network Protocol and is the base protocol of the suite. It works on the principle of the best effort; the protocol tries to do its best to deliver the data from one computer to another. The word "try" is the key as nothing is guaranteed. On the other hand, TCP (Transmission Control Protocol) is probably the most used protocol on the top of the IP protocol and it offers reliable connection between two computers. The TCP/IP suite contains many protocols covering most aspects of the computer communication.

1.2 Literature Review

RTUs were once "dumb" devices that served as an extension of the central computer to the local Input/output. They had no intelligence but could only read the inputs and activate the control outputs. With the development and use of 8-bit, 16-bit and 32-bit microcontrollers, RTUs are now dedicated microcomputers with stand-alone capability [4]. With the RTUs sharing data base management with the central computer, the central computer becomes an operator interface device. The new protocols allow data-base synchronization functions between various parts of the data base.

Dr. Fred Steinhauser addressed some of the challenges arises in the planning, commissioning and operation of the installation of the communication networks and systems in substations [5]. This paper concluded that the communication infrastructure in substation networks is guided by the choice of Ethernet for the network layers.

Samaranayake.et.al presented the design and implementation based on an extensive research and development carried out to enhance the possibilities of using standard TCP/IP Ethernet protocol for condition monitoring and distributed real-time control of industrial drive systems via Ethernet [6, 7]. During this paper, the delay occurred due to connection establishment and connection termination phases of TCP sessions were measured and discussed. They were constant and their values were found depend on the configuration and the direction of traffic flow. The control delay and controller calculation time was evaluated off line using the system clock readings. The controller calculation time was negligible compared to control delay for low and high network traffic respectively. It was observed that the packets still be delayed or even lost if one of the following scenarios appears: If the total network load exceeds the switching capability of the switch engine or the output buffer capacity is not sufficient. The paper also gives solutions to these problems by using the following Ethernet techniques: Ethernet switch back pressure, flow control method and the method of putting the high priority Ethernet packets in a high priority queue.

Qureshi.et.al proposed EPON (Ethernet Passive Optical Networks) based architecture as the desired communication paradigm for the substation automation which was shipped with the two most demanding communication trends, i.e., Ethernet and fiber optic in a single package [8].

The author Clemens Hoga discussed one of the most crucial issues in the field of substation automation: the use of Industrial Ethernet [9].Compared to earlier communication systems, such as the DNP and IEC a 60870-5 protocol, Ethernet has its base not in the industrial sector but in the office environment. To be fully usable in the fields of substation and factory automation, it requires additional features in the following areas: reliability, hardness suitable for substation environment Conditions, power supplies suitable for substations, services enabling short response times and redundancy. This paper showed that during the last ten years, the enormous progress made in these areas has led to the decision to include Industrial Ethernet in the IEC 61850 project.

Skeie.et.al looked at the key issues and requirements for Ethernet in the substation environment and for substation automation applications requiring real-time performance [10].The paper concluded that Ethernet switches used in substation automation applications should comply with either IEC 61850-3 or IEEE P1613 standards for EMI immunity and environmental

requirements to ensure reliable operation of networking equipment in substation environments. Also, a variety of flexible network architectures offering different levels of performance, cost and redundancy are achievable using managed Ethernet switches.

FU.et.al proposed a design and implementation scheme of a general reduced web server protocol stack, which aimed at the limited resources characteristic of the embedded system [11]. It was based on single chip AT90S8515 that uses RISC technology and RTL-8019 network interface controller hardware platform. This single chip has only 8kb programming space and 512 bytes data storage space, while any of the TCP, IP or HTTP protocol is too large to be implemented. The paper evaluated the protocols in standard TCP/IP protocol stack carefully to decide which parts are necessary in the stack and which parts can be saved, using different implementation method for different protocols.

Ahmad Ashari described a research study using virtual IP on a PC that is connected with several microcontrollers, where each microcontroller is connected to a sensor device for monitoring temperature and a relay for controlling [12]. In this method, each set of the microcontroller can be accessed directly using the IP. The results of this study showed that the system can be a distributed monitoring and controlling system which has the ability to record and display the results in the form of graphs. This application is also accompanied by a web-based application that can be accessed from a Browser. The hardware used includes a microcontroller AT Mega 16, the IC MAX232, LM35 temperature sensor and the Lamp module. In this application, there are two levels of logic, i.e. RS232 levels and TTL levels. To bridge the two levels of logic, MAX232circuit is used.

1.3 Previous developed products

Traditional RTU builders are proud to offer different modular hardware building bricks: so they offer a data logger with optional products like analog inputs card, digital inputs card, digital outputs card, communication interfaces, telecom modems and power back-up devices. The customer has to make a choice and compose his own (expensive) solution if he wants the substation to exchange information with the central dispatching or scada system [13].Today, there are many and different manufacturer types and versions of RTUs and PLCs available. Each manufacturer

develops his products, to be acceptable, as per the market needs. Some of these products can be discussed as:

1.3.1 Control Wave® Micro Hybrid RTU/PLC

It is a combination of RTU/PLC Hybrid as shown in figure 1.1. It has high speed ARM 9 based processor. It uses up to eleven serial communication ports and has built-in modem and radio. It has very low power consumption and multiple protocol support. It consists of single and dual 10/100 Mbps Ethernet port and protocol used is IEC 61131-3 programming with ACCOL 3. It has onboard alarm &historical database. It works in the wide temperature range (-40 to +70ºC) [14].

1.3.2 Siemens TG5700 RTU

It has flexible yet powerful IED integration and documentation. It has secure remote maintenance access. It is more than an RTU or data concentrator [15]. It is easy to configure and easy to maintain. Its general I/O LAN Specifications includes baud Rate of 38.4 KB, 32 Nodes per LAN. In case of analog inputs, points per I/O Assembly: 24 bipolar or unipolar, selectable on a per-point basis and input Impedance greater than 108 Ω (DC). It is shown in figure 1.1 below.

Figure 1.1 Control Wave Micro Hybrid RTU/PLC and Siemens TG5700 RTU [14, 15]

1.3.3 Motorola's ACE3600 RTU

The ACE3600 RTU combines the local processing of a PLC with the superior communications of an RTU for an all-in-one high-performance unit [16] which is shown in figure 1.2. It allows seamless integration with multiple PLCs, RTUs and Intelligent Electronic Devices. A powerful processor combined with versatile input/output modules allows this RTU to be used for the most demanding SCADA applications.

Figure 1.2 Motorola's ACE3600 RTU [16]

1.3.4 Kalkitech Sync 2100 series intelligent RTU

These RTUs are ideal for low to medium count IO monitoring. With customizable DI, DO, AI card options, SYNC 2100 products are ideally suited for various Distribution Automation and IO monitoring requirements including as Feeder RTU (FRTU) in RMUs, auto-reclosers and sectionalizers [17]. The device also comes with real-time embedded Linux OS and supports a host of protocols like IEC 60870-101/104, Modbus as well as option to have IEC 61850, DNP3.0 and wireless communications like GPRS /CDMA and 3G over secure VPN connection. This series of RTU is shown in figure 1.3 below.

Figure 1.3Kalkitech Sync 2100 series RTU [17]

1.3.5 Comparison between various RTUs

The comparison between various RTUs of different manufactures with each other and with the RTU explained in this dissertation is given in table below.

Table 1.1 Comparison between various RTUs

	Control Wave Micro Hybrid	**Siemens TG5700 RTU**	**ACE3600RTU**	**Kalkitech Sync 2100 Series**	**Pole Top RTU**
Processor	32-bit ARM9 processor	Intel 80C186 16-Bit processor	Freescale Power PC II	____	32-bit ARM9 processor
Integrated Analog/ Digital Input	(AI): 1-5V, 4-20Ma (DI): 12-24Vdc	(AI): +/-5V or +/-10V (DI): 24Vdc	(AI): +/- 5V, +/-20Ma (DI): 24Vdc	(AI): 4-20Ma (DI): 24/48Vdc	(AI): 4-20Ma (DI): 24Vdc
Serial Ports	RS-232, RS-485, Ethernet **(10Mbps)**	RS-232, RS-485, Ethernet **(10Mbps)**	RS-232, RS-485, Ethernet **(10/100 Mbps)**	RS-232, RS-485, Ethernet **(10/100 Mbps)**	RS-232, RS-485, Ethernet **(10/100Mbps)**, USB (1 device + 2 host ports)
Bluetooth	No	No	No	No	Yes
Power Supply	9-30Vdc	40-60Vdc	18-70Vdc	19-58Vdc	9-36Vdc
Type	Modular	Modular	Modular	Modular	Stand-alone

1.4 Research Gaps

After analyzing the literature review, it has been found that the communication capabilities of the RTU, implemented in the dissertation, are more than the other RTUs present in the market. This RTU includes copper as well as fiber optic Ethernet ports, one device and two host ports of USB, Bluetooth wireless connection, RS-232 and RS-485 ports. Also, the RTUs that can afford such high communication capabilities are modular but the RTU implemented in the dissertation is stand-alone which has all analog and digital sections, communication ports on a single board. Thus, its size is small and very less wiring is needed. Not only this, a TCP/IP stack is also embedded in its processor that is helpful in making reliable interactions between the RTU and the web server. Hence, this RTU will prove to be the optimal choice for industries due to its these advantages.

1.5 Objective of Thesis

The following are the objectives of the dissertation:

i. To design and develop an RTU to enable an electric utility to remotely monitor, control and coordinate the distribution components installed in the substation.
ii. To study the protocols like ARP, ICMP, HTTP, TCP and IP out of several protocols and find the protocol that is the most reliable one for the application.
iii. To achieve Ethernet access in industrial control equipment using a suitable TCP/IP protocol embedded in the microcontroller used.

1.6 Outline of the Dissertation

The dissertation is organized as follows:

Chapter 1 gives the introduction and objective of this dissertation.

Chapter 2 presents the overview of the project "Pole Top Remote Terminal Unit (RTU)". RTU is made up of many sections: analog section, digital inputs section, digital outputs section, power supply section and communication section. All these sections are elaborated with their block diagrams.

Chapter 3 presents an overview of Ethernet communication relevant to this dissertation. Types of Ethernet communication: Copper and Fiber Optic, are also elaborated with their advantages and disadvantages.

Chapter 4 explains different layers of TCP/IP stack in depth with the involved protocols like TCP, IP, HTTP, UDP, and ARP.

Chapter 5 describes the implementation of hardware and software.

Chapter 6 gives the methodology for all the testing.

Chapter 7 provides the final results of the work done.

Finally, **Chapter 8** gives the conclusion on the basis of the dissertation work and the future scope.

CHAPTER 2

POLE TOP REMOTE TERMINAL UNIT (RTU)

2.1 Remote Terminal Unit

Remote Terminal Unit (RTU) is a controller controlled electronic device that interfaces objects in the physical world to a distributed control system or SCADA (supervisory control and data acquisition) system by transmitting telemetry data to a master system, and by using messages from the master supervisory system to control connected objects[18]. Pole top RTU can be depicted in figure 2.1.

Figure 2.1 Pole Top RTU [19]

The RTU monitors the field digital and analog parameters and transmits data to the substations or Central Monitoring Station. It contains setup software to connect data input streams to data output streams, define communication protocols, and troubleshoot installation problems.

A RTU may consist of one complex circuit card consisting of various sections needed to do a custom fitted function[20] or may consist of many circuit cards including CPU, communications interface(s), and one or more of the following: (AI) analog input, (DI) digital input, (DO/CO) digital or control (relay) output, or (AO) analog output card(s). This project includes all these sections on a single board.

The functions of RTU can be categorized as follows [21]:

- Acquisition of information such as measured values, signals and meter readings, etc.
- Transmit commands or instructions (binary type or continuous), set points, control variables etc. including their monitoring as a function of time.
- Recognition of changes in signal input states plus time data allocation and sequential recording of events by the central computer.
- Processing of information transmitted to and from the telecommunication equipment such as data compression, coding and protection.
- Communication with central computer.

2.1.1 Transmission of Power to customers

Electric power is normally generated at 11-25kV in a power station. To transmit over long distances, it is then stepped-up to 400kV, 220kV or 132kV as required. Power is carried through a transmission network of high voltage lines. Usually, these lines run into hundreds of kilometers and deliver the power into a common power pool called the grid. The grid is connected to load centers (cities) through a sub-transmission network of normally 33kV (or sometimes 66kV) lines. These lines terminate into a 33kV (or 66kV) substation, where the voltage is stepped-down to 11kV for power distribution to load points through a distribution network of lines at 11kV and lower [22].

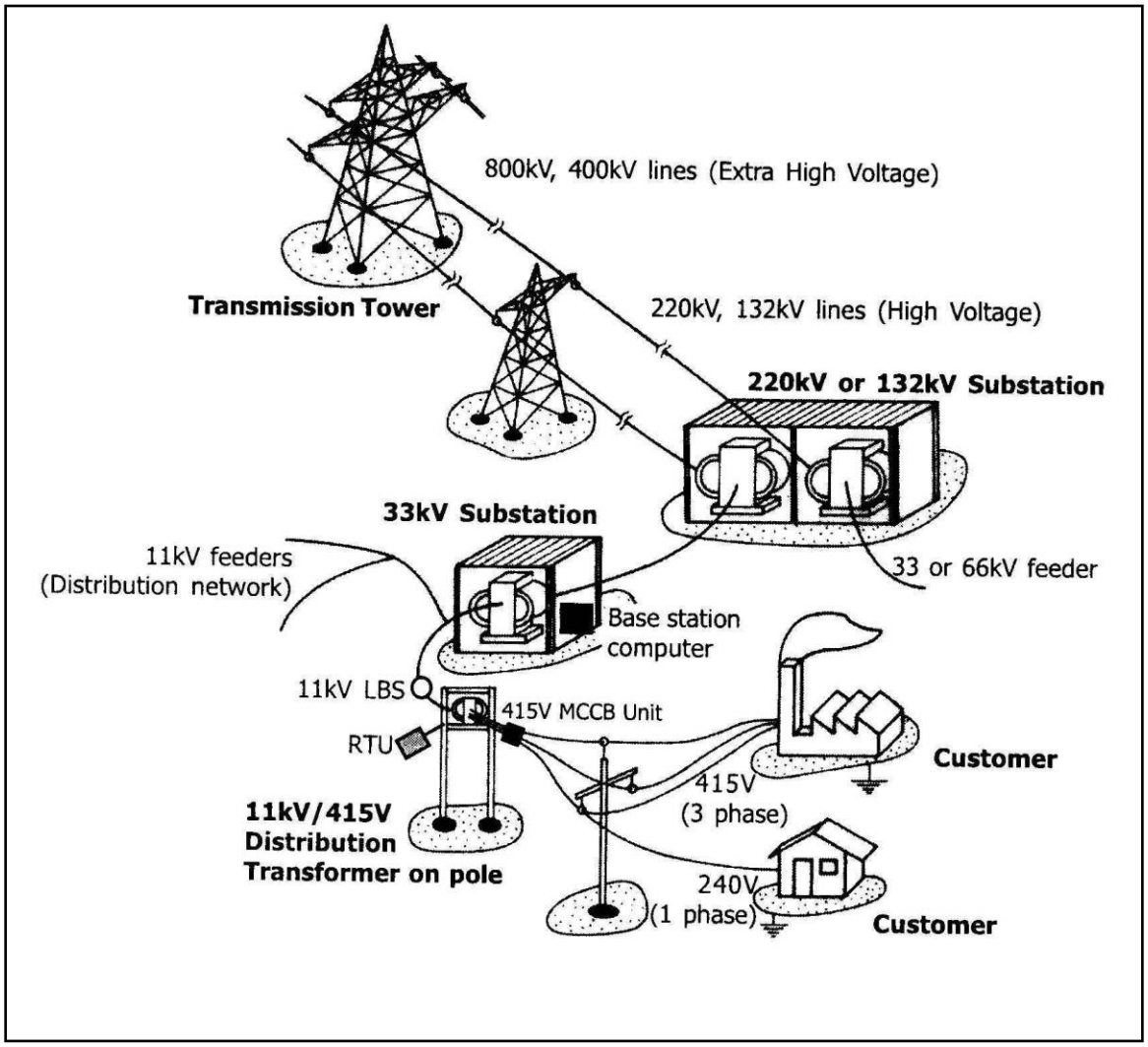

Figure 2.2 Typical Power transfer and distribution scenario [22]

The power network, which generally concerns with the common man, is the distribution network of 11kV lines or feeders downstream of the 33kV substation. Each 11kV feeder emanates from the 33kV substation branches further into several subsidiary 11kV feeders to carry power close to the load points (localities, industrial areas, villages, etc.,) as shown in figure 2.2. At these load points, a transformer further reduces the voltage from 11kV to 415V to provide the last-mile connection through 415V feeders (also called as Low Tension (LT) feeders) to individual customers, either at 240V (as single-phase supply) or at 415V (as three-phase supply). A feeder could be either an overhead line or an underground cable. A 415V feeder should normally be restricted to about 0.5-1.0 km. Unduly; long feeders lead to low voltage at the consumer end.

2.1.2 Justification for Voltage Transformation

As seen from figure 2.2, along the route from the source to the customer, the electricity is undergoing numerous transformations, with generating voltage getting stepped-up to transmission level with a follow-up decrease down to distribution and eventually, customer levels. One of the ways to reduce power and energy losses may be accomplished by raising the voltage level.

Another benefit from raising the voltage is a reduction of voltage drop. This is done by increasing the voltage for transmission of electrical energy. After it is delivered to the customers of a locality, the voltage is gradually lowered to the safe utilization level (220/120 V). Obviously, electrical transformer substations are playing a major role in accomplishing this task. The number of steps, in raising and lowering the voltage, are defined through optimization studies performed by utility company planners.

2.2 Architecture of Remote Terminal Unit

As the whole remote terminal unit is on a single board i.e. stand-alone, so placing all the sections on this single board properly is a very critical task. Figure2.3 shows basically the architectural placement diagram of remote terminal unit.

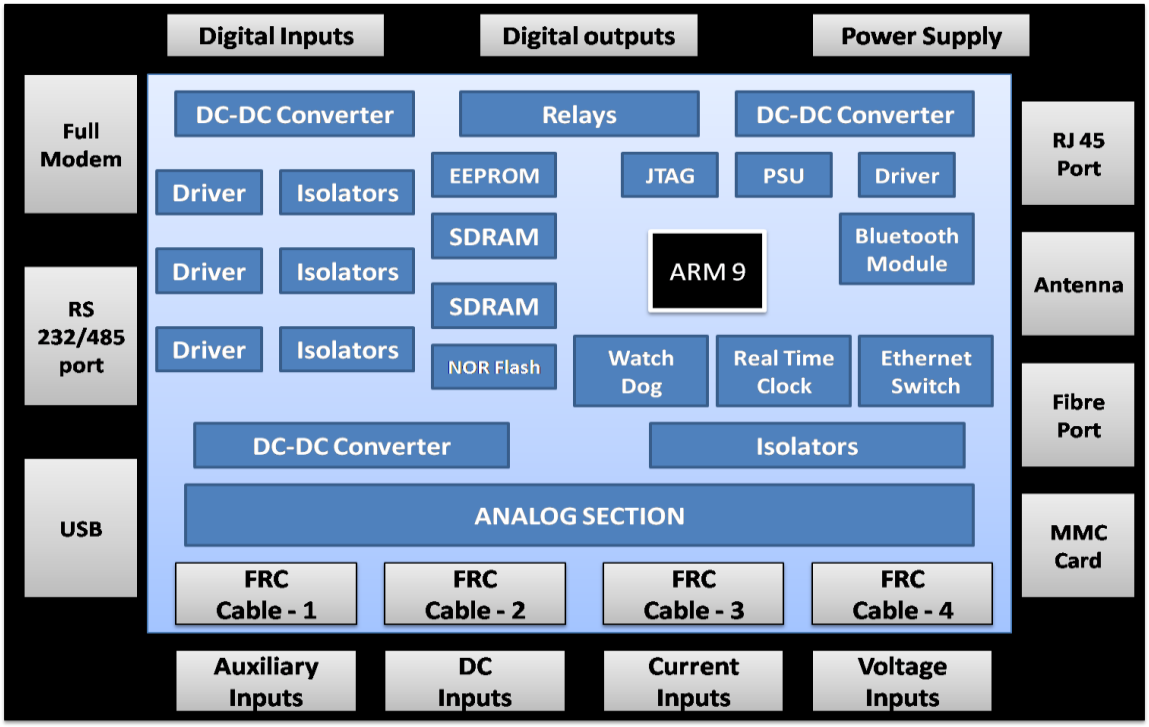

Figure 2.3Architectural Placement Diagram of Remote Terminal Unit

It can be seen from the figure that the connectors for all types of inputs and outputs like digital inputs, digital outputs, analog inputs, power supply input, are placed at the edge of the Printed Circuit Board (PCB). Also, MMC (Multi-Media Card) and all communication ports: full modem, RS-232/485, USB, RJ45, fiber, antenna, are placed at the edge of the board to provide easy connection with other devices. Analog section and digital section are isolated from each other, which is very necessary, with the help of various isolators. DC-DC converters are also placed for voltage conversions in different parts of the system as per the requirement. This critical task of placement of components should be done very accurately to avoid any failure or error in the system.

2.3 Block Diagram of Remote Terminal Unit

The block diagram shown in figure 2.4 describes the basic interconnection of controller with various sections.

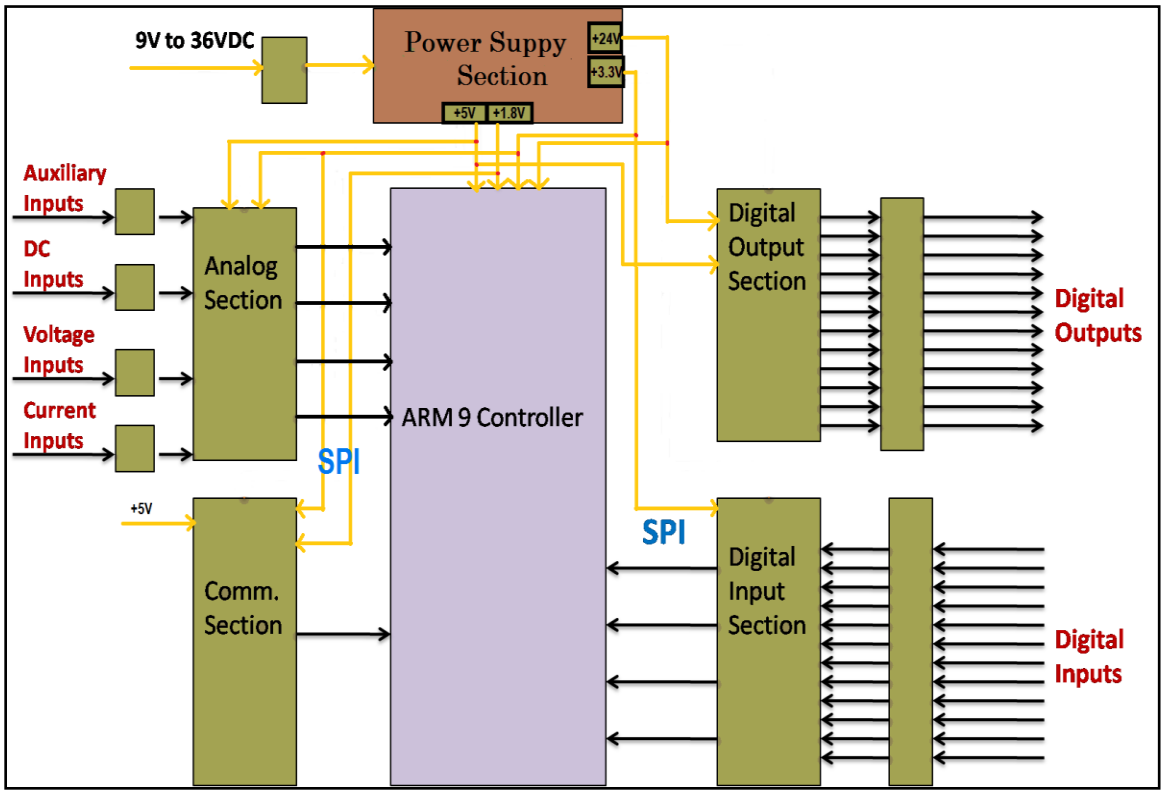

Figure 2.4 Block Diagram of Remote Terminal Unit

As shown in this figure, the main power supply to the board ranges from 9V to 36VDC. It is converted into +24V, +5V, +3.3V and +1.8V, with the help of several DC-DC converters, as different sections require different voltages as per their requirements. As shown, analog section works on +5V and +3.3 V, DO section works on +24V and +5V, DI section works on +3.3V and communication section takes +3.3V, +1.8V and +5V. +5V is given to it from another source to make it stand-alone from the whole system.

2.4 ARM9 Controller

The controller used in this single board remote terminal unit is AT91SAM9XE512 which is the heart of this whole project. It is the central processing unit of the whole system, which is responsible for controlling and maintaining synchronization between all the sections in this system. It is the master and all other sections are slaves. The main features of this controller are given below.

2.4.1 Features

Some of the important features of ARM9 controller are [23]:

1. Additional Embedded Memories
   - One 32 Kbyte Internal ROM.
   - One 32 Kbyte Internal SRAM.
   - 512 Kbytes of Internal High-speed Flash.
2. USB 2.0 Full Speed (12 Mbits per second) Device Port.
   - On-chip Transceiver and 2,688-byte Configurable Integrated DPRAM.
3. USB 2.0 Full Speed (12 Mbits per second) Host Double Port.
   - Single or Dual On-chip Transceivers.
   - Integrated FIFOs and Dedicated DMA Channels.
4. Ethernet MAC 10/100 Base-T
   - Media Independent Interface or Reduced Media Independent Interface.
   - 28-byte FIFOs and Dedicated DMA Channels for Receive and Transmit.
5. Fully-featured System Controller, including
   - Reset Controller, Shutdown Controller.
   - Four 32-bit Battery Backup Registers for a Total of 16 Bytes.

- Clock Generator and Power Management Controller.
- Advanced Interrupt Controller and Debug Unit.
- Periodic Interval Timer, Watchdog Timer and Real-time Timer.

6. Two-slot Multimedia Card Interface (MCI)
   - SD Card/SDIO and Multimedia Card™ Compliant.
   - Automatic Protocol Control and Fast Automatic Data Transfers with PDC (Peripheral DMA Controller).
7. Two Master/Slave Serial Peripheral Interfaces (SPI).
8. Required Power Supplies:
   - 1.65V to 1.95V for VDDBU, VDDCORE and VDDPLL.
   - 1.65V to 3.6V for VDDIOP1 (Peripheral I/Os).
   - 3.0V to 3.6V for VDDIOP0 and VDDANA (Analog-to-digital Converter).
   - Programmable 1.65V to 1.95V or 3.0V to 3.6V for VDDIOM (Memory I/Os).

2.5 Description of RTU Sections

2.5.1 Digital Inputs (DI) Section

This section is made to provide feedback to the controller about the status of various field devices. These digital inputs can be the status of circuit breaker, isolator and sequence of events. Hence, this section takes inputs from field relays that are connected to various devices. These inputs are either ‘1’ or ‘0’. Thus, these are called digital inputs. In this project, there are 12 digital inputs with 1 common terminal. This section works upon +24V and can also be called as Feedback Section.

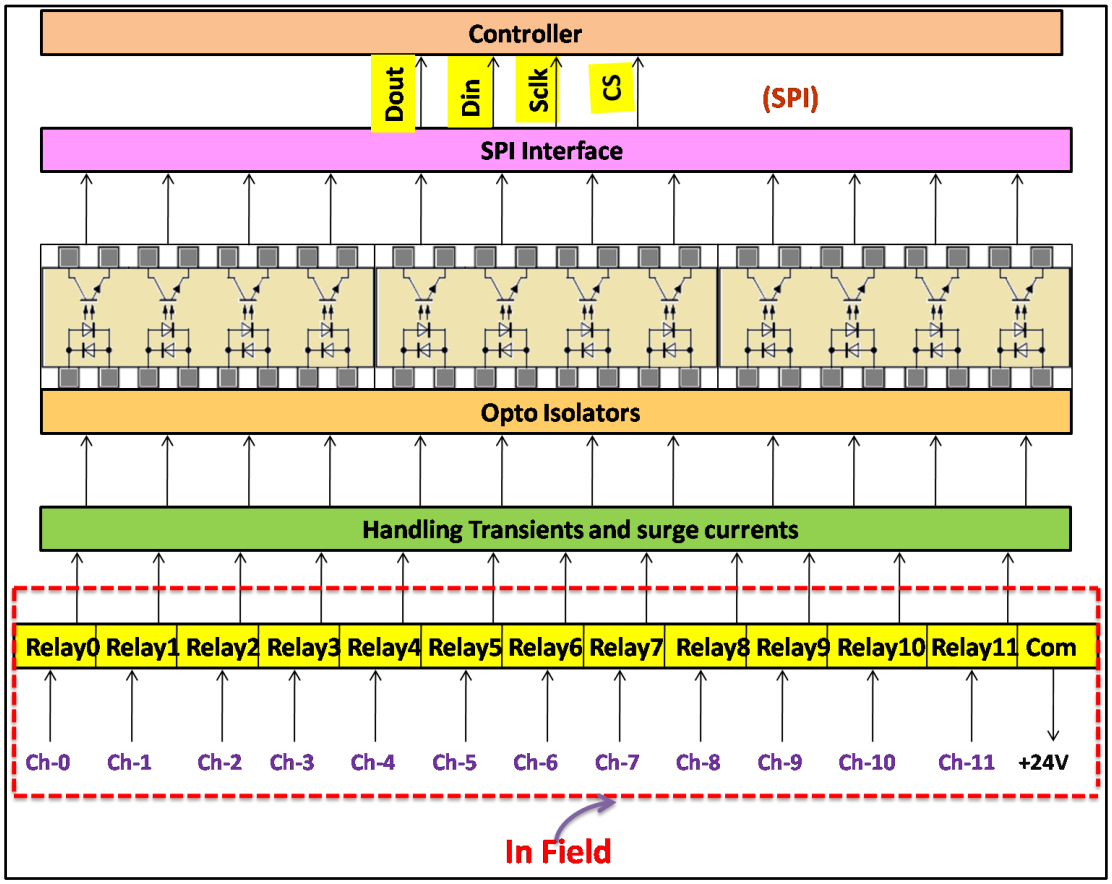

Figure 2.5 Block Diagram of DI Section

As can be seen in figure 2.5, there are a number of relays connected to various devices in the field. The outputs of these relays are given as digital inputs to the DI section. These inputs are passed through some protecting devices like MOV, TVS etc. for handling transients and surge currents. It is required to isolate controller from field signals, so optoisolators are necessary to provide such isolation. As shown, now, outputs of isolators are sent to the controller using Serial Peripheral Interface.

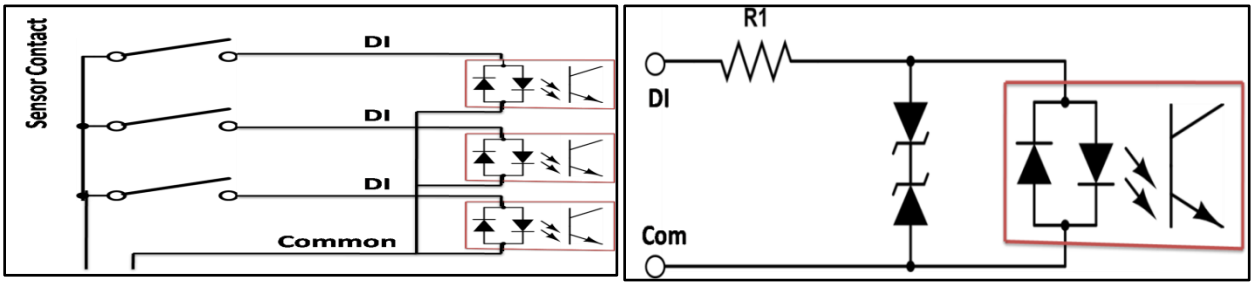

Figure 2.6 Sensor wiring and equivalent circuit [24]

The connection of field relays to sensors and to the DI section is shown in figure 2.6. Relays are operated according to the status of the sensors and are then given to the controller by passing them from optoisolators. The figure 2.6 also shows the equivalent circuit of DI section for only one channel of digital inputs. It shows the protection device (MOV or TVS diodes) and necessary passive components to be used in its circuitry.

2.5.2 Digital Outputs (DO) Section

As DI section was for monitoring, the purpose of DO section is for controlling the field devices like circuit breaker etc. After continuous monitoring of field devices by DI section, if there is any need arises to control any of the field devices at a particular time, the controller will give command to the corresponding relays in the DO section and then these relays will send commands, in the form of digital outputs, to the field devices connected to them and so, in this way, field devices are controlled.

In this section, 8 digital output channels are used. The relays used have 3 terminals: Common, NO (Normal Open), NC (Normal Closed). Hence, the relays are SPDT (Single Pole Double Throw).

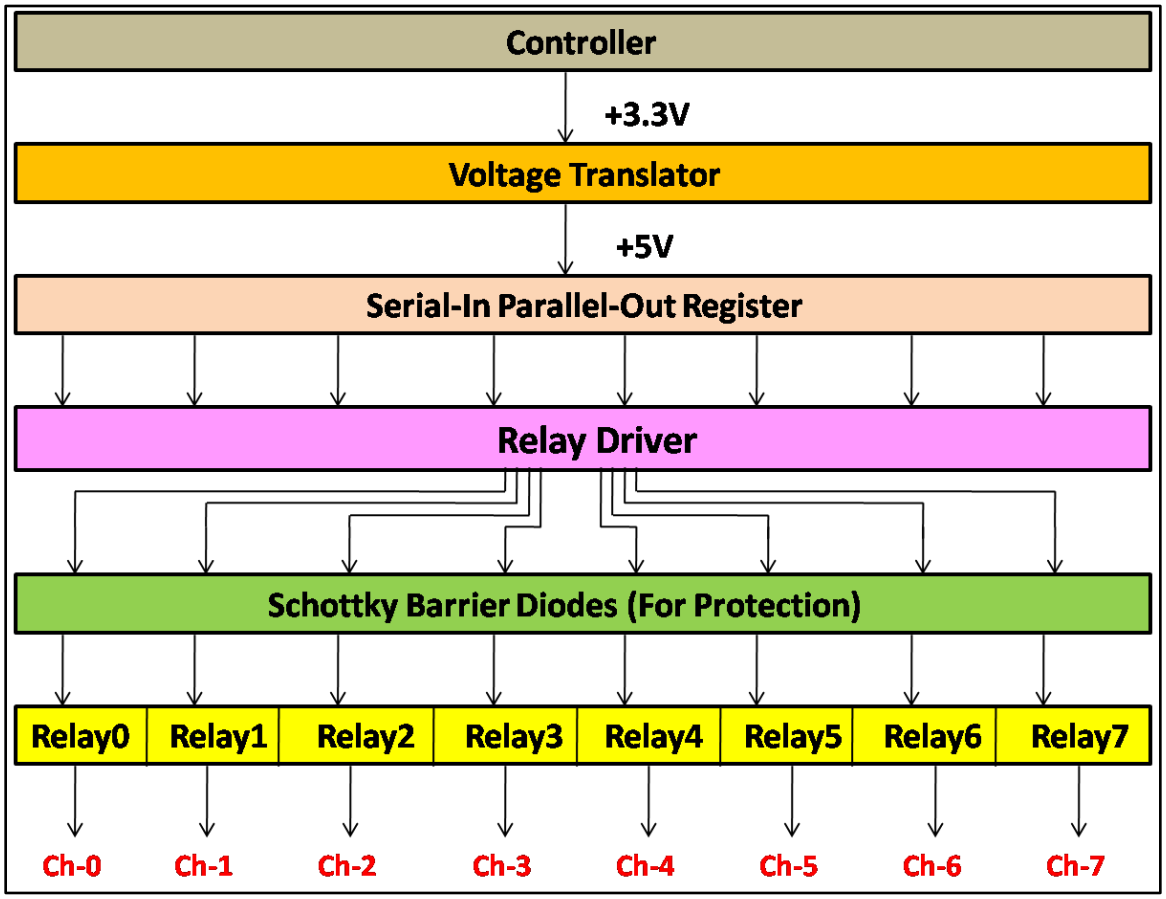

Figure 2.7 Digital Outputs Section

As shown in figure 2.7, after taking feedback from the DI section, controller wants to control the field devices directly by on-board relays. For this purpose, a Serial-In-Parallel-Out (SIPO) register is required to convert serial data from the controller into the parallel one for operating the relays connected in parallel. The Controller and SIPO register works on +3.3V and +5V respectively. Thus, a voltage translator is needed to convert these voltage levels. Now, as shown in this figure, a relay driver is always required for driving the relays which will then control the field devices connected to them. In this way, control action takes place.

2.5.3 Analog Inputs (AI) Section

This section is the most complex and critical one. It gives information about on site physical parameters, line voltages, line currents to the controller continuously. This section consists of various inputs according to application:

1. Auxiliary Inputs.

2. Voltage Inputs.
3. Current Inputs.
4. DC Inputs.

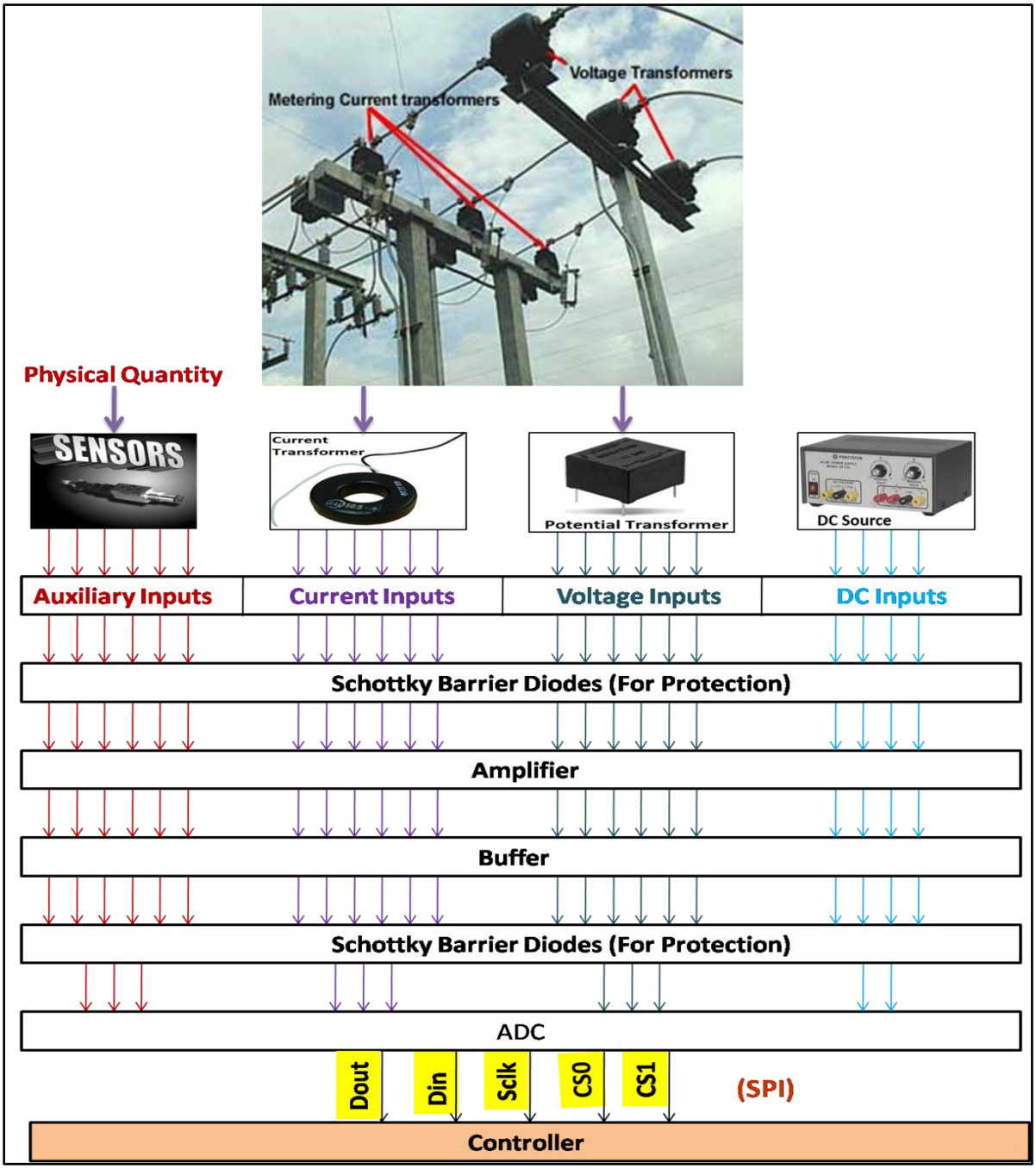

Figure 2.8 Block Diagram of overall Analog Section

In figure 2.8, processing of all four types of inputs is shown. According to the figure, processing of each type of input is explained one-by-one.

1. Auxiliary Inputs

In the field, physical parameters like pressure, temperature etc. is sensed by various sensors. These sensors provide their output in the range of 4-20 mA. After passing through resistor network, these current signals are converted into voltage signals. Here, 3 auxiliary input channels are used. From figure 2.8, these signals are given to schottky barrier diodes for protection against transients. Now, these signals are given to amplifiers which act as buffer and provide noise free signal. These analog signals are then digitized with the help of Analog-to-Digital Converter (ADC). These digitized signals are given to the controller with the help of serial peripheral interface as shown in the figure.

2. DC (Direct Current) Inputs

In this section, two DC input channels are used: DC1 and DC2. DC inputs are coming from DC source as shown in figure 2.8. Here, DC inputs range from 70V to 280V. These signals undergo the same procedure as the auxiliary inputs do. In this way, these DC inputs are continuously measured and monitored.

3. Voltage and Current Inputs

Voltages and currents in the transmission lines are of very high value. These high values can’t be measured directly. They have to be converted into some lower values which can be conveniently connected to measuring and recording instruments. For this purpose, current transformers (CT) and potential transformers (PT) are used for line currents and line voltages respectively.

Potential transformers (PT) (also called voltage transformers (VT) are a parallel connected type of instrument transformer. They are designed to present negligible load to the supply being measured and have an accurate voltage ratio and phase relationship to enable accurate secondary connected metering. The PT is typically described by its voltage ratio from primary to secondary

(E.g. 600:120). Burden and accuracy are usually stated as a combined parameter due to being dependent on each other. In current transformer, the primary winding is the main conductor passing through the center of the core. The secondary winding is uniformly distributed around the toroidal core. Essentially, all the flux which links the primary conductor also links the secondary winding.

The rating of current transformers used on transmission lines is approx. 800/1200A to 1A or 5A and rating of potential transformers is approx. 220 KV to 110 KV. The output of these transformers is then given to another current and voltage transformers of lower rating as shown in figure 2.8. The ratings of these current and voltage transformers are 1A/5A to ~ 200mV and 110 KVto ~ 200mV. According to the figure, the output of PT is amplified and then converted into digital form with the help of ADC. Finally, these signals are given to the controller.

2.5.4 Communications Section

It consists of many types of communication:

1. RS-485/RS-232
2. USB (Universal Synchronous Bus)
3. Ethernet

1. RS-485/RS-232

Electronic data communications between elements will generally fall into two broad categories: single-ended and differential. RS-232 is a single-ended communication and RS-485 is a differential communication. When communicating at high data rates, or over long distances in real world environments, single-ended methods are often inadequate. Differential data transmission (balanced differential signal) offers superior performance in most applications. Differential signals can help nullify the effects of ground shifts and induced noise signals that can appear as common mode voltages on a network.

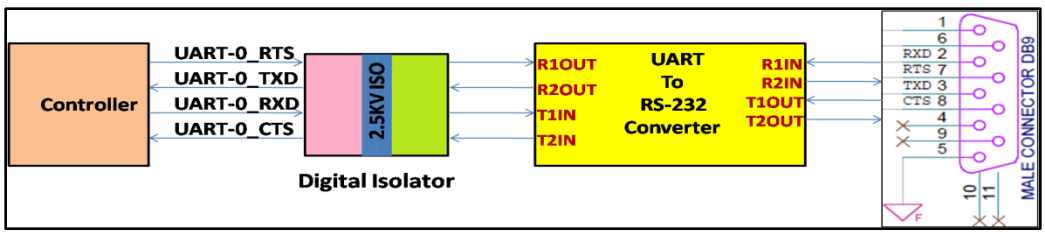

Figure 2.9 UART-0 to RS-232 link

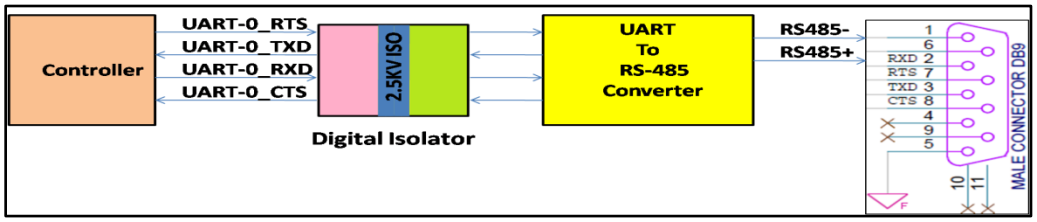

Figure 2.10 UART-0 to RS-485 link

In this section, the controller uses its two serial ports: UART-0 and UART-1. The signals from these ports are sent to RS-485 and RS-232 ports with the help of UART to RS-232 and UART to RS-485 converters as shown in figures 2.9 and 2.10. According to these figures, digital isolators are always placed between the data controller and the data converter. The digital isolator used here provides 2.5 KV of isolation.

2. USB (Universal Serial Bus)

A USB interface consists of 4 wires: Power, Ground, Data + (USBDP) and Data - (USBDM) as shown in figure 2.11.

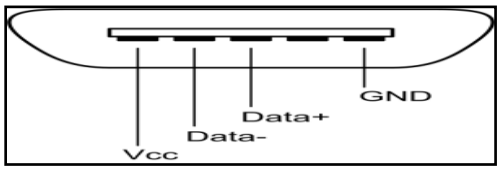

Figure 2.11 USB Pin Diagram [26]

A USB host port with no devices connected uses some high resistance to connect both USB DP and USB DM to GND. When a USB device (sometimes referred to as a slave) is plugged into a USB host there is a change on these USB data lines. It is this change that the USB host uses to detect a device has been connected. This change is also used to identify the speed of device attached.

In this chapter, the power transfer and distribution scenario is explained and so the significance of remote terminal unit in substation automation. Also, working and significance of various sections of a remote terminal unit like analog section, digital section, communications section etc. are discussed along with the proper diagrams. In this project, various types of communications are used but only those types are explained that are relevant to this dissertation. Here, the most important communication is Ethernet Communication which is discussed in the next chapter.

CHAPTER 3

ETHERNET COMMUNICATION

3.1 Introduction

Ethernet was developed by Bob Metcalfe and others at Xerox PARC in mid-1970s.It was standardized by Xerox, DEC, and Intel in 1978. The diagram for Ethernet, drawn by Metcalfe, is given as in figure 3.1.

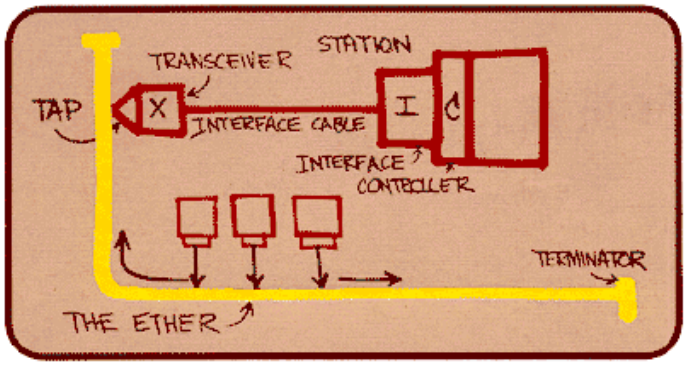

Figure 3.1 Ethernet by Bob Metcalfe [27]

As shown in this figure, there is a controller interfaced with a transceiver. This transceiver shares a common transmitting medium called Ether. Here, all the transceivers can transmit and receive data on this single shared medium.

3.2 Principle of Operation

Ethernet is a LAN protocol and follows the principle of CSMA/CD which stands for Carrier Sense Multiple Access/Collision Detect. This is explained as [28]:

- **Carrier Sense -** Before an Ethernet node can begin transmitting, it must first determine whether the medium is active or Idle (Carrier Sense). If the medium is active, then that node must wait until the medium becomes Idle, and then waits a predetermined amount of time after that before starting to transmit. This predetermined amount of time is called the Inter-Frame Gap (IFG), and is dependent on the speed of the bus.
- **Multiple Access -** This means that all machines on the network are free to use the network whenever they like as long as no one else is transmitting.

- **Collision Detection –** There is a collision, when two machines try to transmit at the same time. Hence, it is a means of ensuring that when two machines start to transmit data simultaneously, then that transfer of data is discarded and re-transmissions are generated at differing time intervals. So, each machine has to 'back off' for a random period of time before re-trying.

3.3 Binary Exponential Back off Algorithm

The last requirement for our network protocol is a method by which each node determines when to retransmit. If every node tries to retransmit at the same time, collisions would continue. For this reason, Ethernet implements a *Binary Exponential Backoff Algorithm,* which works as follows:

Each node chooses a random delay (in the range from 0 to 1) before attempting its first retransmit. If another collision occurs, each node doubles the range of random delays (now from 0 to 3) and chooses a random delay again. This process repeats (with a range of 0 to 7, 0 to 15, etc.) until no collision occurs or until 10 attempts have been made. At this point, the defined range for each node will be 0 to 1023. In this manner, the range of back off times increases exponentially with each try and the probability of a collision rapidly decreases [29].Six more attempts (for a total of 16 attempts) will be made to retransmit. If a node is still unsuccessful at retransmitting, the frame is dropped, and an excessive collision error is reported. The application software must then detect the dropping of the frame and try to retransmit the dropped frame, if desired.

3.4 Difference between a Switch and a Hub

An Ethernet switch and an Ethernet hub perform the same task of sending and receiving data to and from the devices connected on their ports, but, both have a totally different way of doing this task.

Ethernet Switch

It is an intelligent device that sends data to only that port for which it intends to. It dramatically reduces the network congestion caused by using the CSMA/CD media access control over shared media[30].A switch will examine the incoming frame's MAC header to determine the physical address of the destination device. It examines each frame's Destination Address field and

forwards it only to the port which is attached to the destination device. In order to operate effectively, switches maintain their own MAC look-up tables. These tables illustrate which network device is at which port on the switch. The look-up tables are constantly updated. When a switch is newly installed, it will construct its table via an Address Resolution Protocol (ARP) broadcast.

Ethernet Hub

This device is not intelligent like Ethernet switch. It is a device for connecting multiple Ethernet devices together and making them act as a single network segment. It has multiple I/O ports, in which a signal introduced at the input of any port appears at the output of every port except the original transmitting port. A hub does not examine or manage any of the traffic that comes through it [31]. Also, it cannot support multiple rates or formats (e.g., 10 Mbps vs. 100 Mbps Ethernet).

3.5 Ethernet Frame Format

A basic 10/100 Ethernet frame consists of the following fields, as shown in figure 3.2:

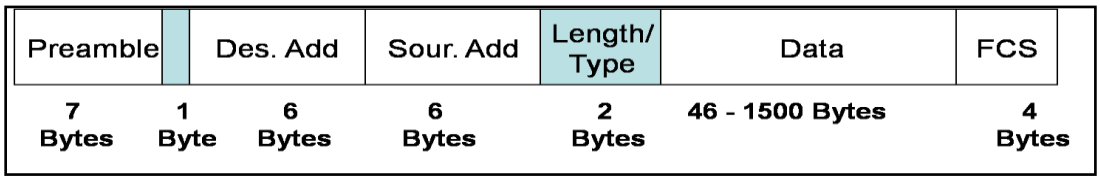

Figure 3.2 Ethernet Frame Format

- **Preamble:**

It consists of seven octets of 55h. The preamble is present to allow the receiver to lock onto the stream of data before the actual frame arrives. At the point, when the first bit of preamble is received, the receiver may be in an arbitrary state (i.e. have an arbitrary phase for its local clock). During the course of preamble, it learns the correct phase.

- **Start-of-Frame Delimiter (SFD):**

This field is only available in the IEEE 802.3 specification. It indicates an 8-bit value - 10101011b. SFD was considered to be a part of the preamble in earlier Ethernet frame format. It is used to indicate the beginning of the frame.

Preamble and SFD are not a part of the frame.

- **Destination Address (DA):**

It represents the 6-octet MAC address of the destination hardware. The MAC address is discussed in section 3.6.1.

- **Source Address (SA):**

It represents the 6-octet MAC address of the source hardware.

- **Length/Type:**

If the value in this 2-octet field is ≤ 1500 (decimal), this represents the number of octets in the payload. If the value is ≥ 1536, this represents the Ether Type (payload type). The following are the most common Ether Type values:

- IPv4 = 0800h
- IPv6 = 86DDh
- ARP = 0806h
- RARP = 8035h

- **Payload (Client Data):**

The next field is the Data field. The data field is split up into the Mac Client Data field and the Pad field. The Mac Client Data field holds the data that is sent in the frame. Although the minimum frame length is 46 bytes, but it might be possible that less than 46 bytes of data have to be sent. If that happens, the Pad field is used to ensure that the Data field contains 46 bytes. The total length of the data field should always be between 46 and 1500 octets.

- **Frame Check Sequence (FCS):**

The Frame Check Sequence field contains a Cyclic Redundancy Check. Both the receiver and transmitter calculate a value in function of the Source Address, Destination Address, Type/Length and Data Field. The transmitter places this value in the Frame Check Sequence field and the receiver checks the value in this field against the value it calculated itself. If these values don't match, the frame is considered as corrupted.

3.5.1 MAC (Media Access Controller) Addresses

A MAC address is a 48-bit (6-octet) number unique to every piece of Ethernet hardware. It consists of a 24-bit Organizationally Unique Identifier (OUI) and a 24-bit hardware identifier as

shown in figure 3.3. OUIs are assigned by the IEEE to a particular company or organization, while hardware IDs are assigned by the owner of that particular OUI. An example of microchip owned MAC address is given in figure 3.3.MAC address octets are transmitted with high-order octet (Octet #1) first, while bits within an octet are transmitted with low-order or Least Significant bit (LSB) first.

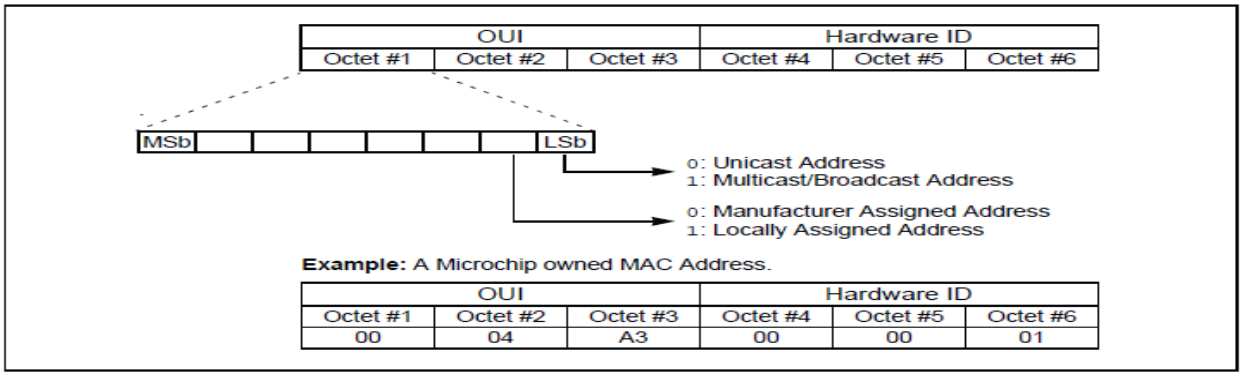

Figure 3.3 MAC Addresses [32]

As shown in this figure, in a MAC address, LSB of Octet #1 can be set as a multicast address, intended for one or more nodes, or a unicast address. As an example, a MAC address of FF-FF-FF-FF-FF-FF is a broadcast address, which is intended for all nodes.

3.6 Ethernet Cabling

The cabling of Ethernet has evolved over time and so today, a number of different Ethernet cablings are available. Ethernet cabling is of two types: using copper and using fiber optics. Earlier, coaxial cable was once the primary medium for Ethernet and other types of local area networks. Its design was very effective at preventing data loss. It had a maximum effective range of 500 meters and requires fewer repeaters. But, on the other side, it is more expensive to install and its thicker cable made it more difficult to work with. With the development of standards for Ethernet over twisted-pair, new installations of coaxial cable have totally disappeared [33].

Now, twisted pair cabling is a type of wiring in which two conductors of a single circuit are twisted together for the purposes of canceling out electromagnetic interference (EMI) from external sources. It can handle a data flow of up to approximately 1Mbps over several hundred feet. It is inexpensive and easy to install. Thus, it is the ideal choice for small LAN with a limited number of

users. But it will be incompatible, if talking to more than 1 computer by going through a switch or hub.

Nowadays, fiber optic communication is on hype. It has many advantages over previous Ethernet cablings like it has high noise resistance, less signal attenuation and higher bandwidth. But optical fibers are very expensive. These have high maintenance or installation costs also. But these disadvantages of fiber optics become negligible in front of its advantages.

3.7 Varieties of Ethernet

There are varieties of Ethernet available. These are described in table 3.1 as shown below.

Table 3.1 Varieties of Ethernet [34]

Ethernet Type	Speed	Cable Type	Distance
10Base-5	10Mbps	Co-axial	500m
10Base-2	10Mbps	Co-axial	185m
10Base-T	10Mbps	Copper	100m
10Base-F	10Mbps	Fiber	2Km
100Base-TX	100Mbps	Copper	100m
100Base-FX	100Mbps	Fiber	2Km
1000Base-T	1Gbps	Copper	25m
1000Base-X	1Gbps	Fiber	220m to 10Km

This table shows different Ethernet types based on speed, cable type and distance. According to the nomenclature explained above, these Ethernet types can be distinguished easily. E.g. 10Base-5 indicates a speed of 10Mbps, baseband modulation and a reach of 100m.

3.8 Significance of Ethernet Communication

Since the very beginning, communication infrastructure plays a vital role in mediating between physical and virtual worlds of substation. Recent developments in communication

technologies have enabled reliable remote control systems which have the capability of monitoring the real time operating conditions and performance of electric systems [35].These communication technologies can be classified into the following types [8]:

- *Power line communication (PLC):* This communication utilizes the existing infrastructure of electricity for communication, thereby, offering a broad coverage as well as a cost effective solution but, on the other hand, it is also suffered by noise, capacity, signal distortion and security issues.
- *Wireless communication:* Due to its rapid deployment and cost effective solution, wireless communication has dramatically changed the trend of communication paradigm but still need to address the crucial issues of capacity, security, and coverage in order to make it viable.
- *Ethernet Communication:* Ethernet, whether copper or fiber, is always on demand for industrial purposes. Where copper Ethernet communication is used for its good speed, cheap costs and easy installation properties, Optical fiber communication has also taken the major share of communication due to its high capacity and immunity characteristics for EMI/RFI (Electro Magnetic Interference / Radio Frequency Interference). Though cost of deploying fiber optic communication is expensive but it is overcome by the advantages offered.

Thus, Ethernet communication is always a preferred choice over other types of communication.

3.9 Ethernet Module

The Ethernet communication module is designed to transfer data to and from the Remote Terminal Unit for interacting with the substations. This data can be the status of various sensors installed in field like temperature sensor, pressure sensor (density meter), tap position sensor etc. or it can be the parameters like line voltages, currents, frequency, active energy, apparent energy etc.
This interaction is possible with the help of Ethernet switch which sends data to and from the controller to available ports in the switch IC used for different purposes. This Ethernet module can be shown in figure 3.4.

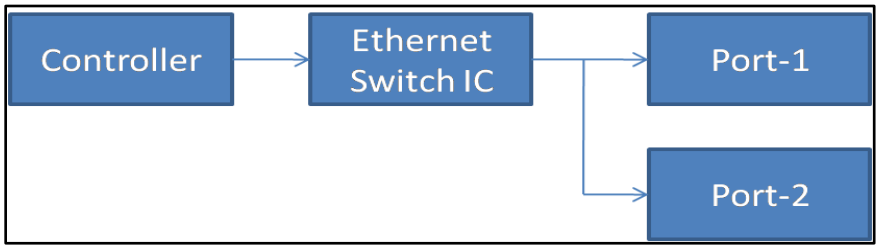

Figure 3.4 Ethernet Communication

This module uses PIC32MX795F512H controller which is interfaced with micrel's Ethernet switch KSZ8863FLL/RLL/MLL. From the above figure, this Ethernet switch consists of two ports, one for copper and one for fiber. The controller and the Ethernet switch IC are explained in sections 3.10.1 and 3.10.2 respectively.

3.9.1 PIC32MX795F512H Micro Controller

This PIC controller controls each and every transmission and reception of data to and from the devices connected on the two ports of the Ethernet switch. Some of the important features of this controller are as follows [36]:

i. Microcontroller Features:

- Operating voltage range of 2.3V to 3.6V.
- 64K to 512K Flash memory (plus an additional 12 KB of Boot Flash)
- 16K to 128K SRAM memory.
- Pin-compatible with most PIC24/dsPIC® DSC devices
- Multiple power management modes
- Multiple interrupts vectors with individually programmable priority
- Fail-Safe Clock Monitor mode
- Configurable Watchdog Timer with on-chip Low-Power RC oscillator for reliable operation.

ii. Peripheral Features:

- Up to 8-channels of hardware DMA with automatic data size detection
- USB 2.0-compliant full-speed device and On-The-Go (OTG) controller.
- 10/100 Mbps Ethernet MAC with MII and RMII interface.

- CAN module:
– 2.0B Active with DeviceNet™ addressing support.
– Dedicated DMA channels.
- 3 MHz to 25 MHz crystal oscillator.
- Internal 8 MHz and 32 kHz oscillators.
- Six UART modules withRS-232, RS-485 support.
- Up to four SPI modules.
- Up to five I2C™ modules.
- Separate PLLs for CPU and USB clocks.

iii. Debug Features:

- Two programming and debugging Interfaces:
– 2-wire interface with unintrusive access and real-time data exchange with application.
– 4-wire MIPS® standard enhanced Joint Test Action Group (JTAG) interface.

3.9.2 Ethernet Switch

Before explaining the working of Ethernet switch, it is necessary to understand some of its basic concepts that will be helpful in understanding the working of the switch easily.

3.9.2.1 Functions of Ethernet MAC and PHY

Functions of the Ethernet PHY and MAC, and the interfaces of each, are defined by the IEEE 802.3 specification as given in figure 3.5.

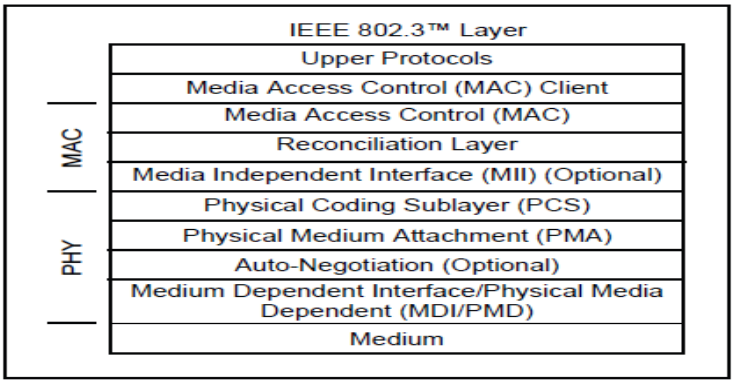

Figure 3.5 100 Mb/s Layer Definitions [32]

The physical interface to the transmission medium is called the MDI (Media Dependent Interface) which changes depending on which medium (twisted pair, fiber, etc.) is used. The interface between the PHY and the MAC is called the MII (Media Independent Interface), and is composed of a receive path, a transmit path and a management path, which is used to read and write PHY registers. The width of receive and transmit paths are the same, and is determined by the speed that the MAC and PHY are implementing, as follows:

- 10 Mb/s: 4 bits wide at 2.5 MHz
- 100 Mb/s: 4 bits wide at 25 MHz

Reconciliation Layer: Maps the physical status (carrier loss, collision, etc.) to the MAC layer.
Media Independent Interface (MII) (Optional): Provides an n-bit transmit/receive interface to the PHY.
Physical Coding Sub layer (PCS): Encoding, multiplexing and synchronization of outgoing symbol streams (4B/5B encoding, etc.).
Physical Medium Attachment (PMA): Signal transmitter/receiver (serialization/deserialization of symbol stream, clock recovery, etc.).
Auto-Negotiation (Optional): Negotiation to the highest mode supported by both hosts.
Medium Dependent Interface/Physical Media Dependent (MDI/PMD): RJ45, etc.
Medium: UTP, Fiber, etc.

3.9.2.2 Types of Switches

The KSZ8863MLL/RLL/FLL has the flexibility to reside in either a managed or unmanaged design. These designs are explained as:

1. Unmanaged Switch

An unmanaged design is achieved through I/O strapping and/or EEPROM programming at system reset time. It can be seen in figure 3.6.

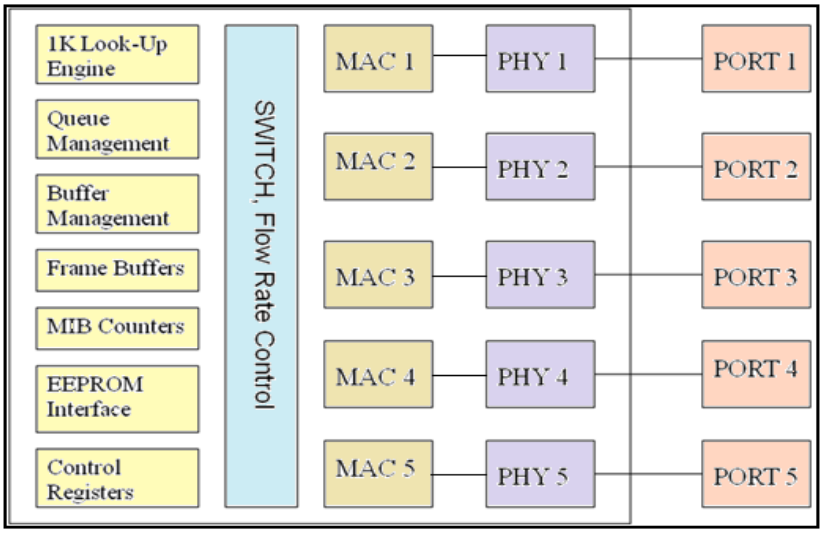

Figure 3.6 Unmanaged Switch [37]

From this figure, it can be understood that unmanaged switch is called so because there is no need of any controller to manage it. At the system reset time, the switch is programmed with desired or default configuration settings with the help of EEPROM and so the switch will work according to those settings.

2. Managed Switch

In a managed design, the host processor has complete control of the KSZ8863MLL/FLL/RLL via the SMI interface, MIIM interface, SPI bus, or I2C bus.

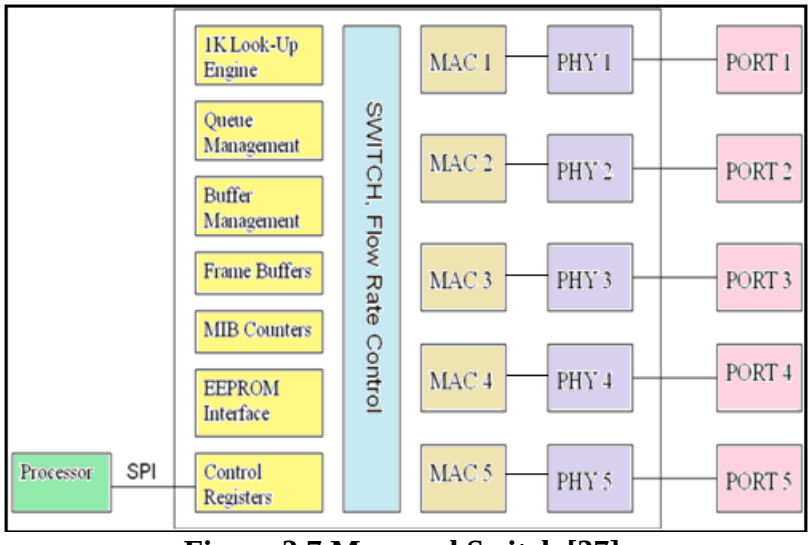

Figure 3.7 Managed Switch [37]

The figure 3.7 shows the managed switch. Here, processor controls the switch using the SPI bus. Using SPI bus, the processor can directly access global registers, port registers and advanced control registers of the switch.

3.9.2.3 Functional Block Diagram

The KSZ8863MLL/FLL/RLL contains two 10/100 physical layer transceivers and three MAC units with an integrated Layer2 managed switch.

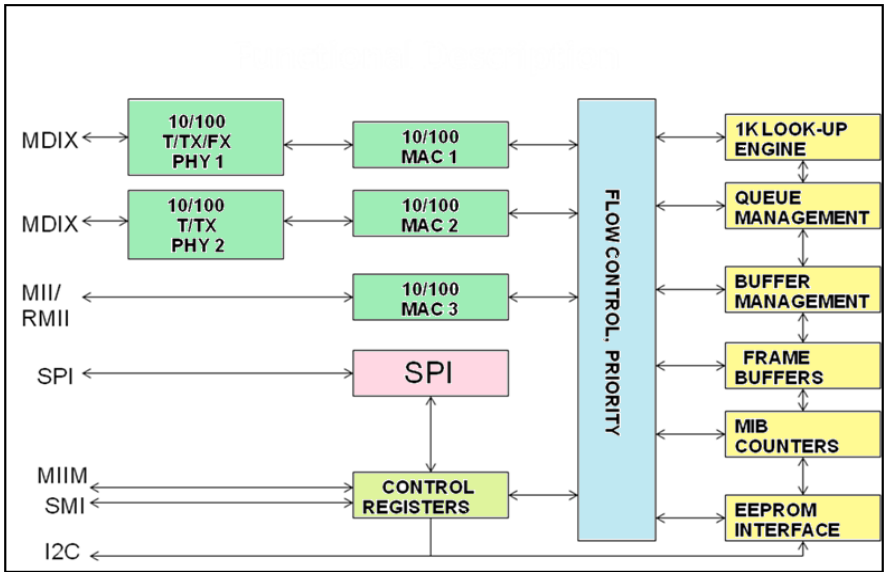

Figure 3.8 KSZ8863FLL/MLL/RLL Functional Block Diagram [38]

As can be seen from block diagram given in figure 3.8, on the media side, the KSZ8863MLL/FLL/RLL supports IEEE 802.3 10BASE-T and 100BASE-TX on both PHY ports (Figure 3.15). Physical signal transmission and reception are enhanced through the use of patented analog circuitries that make the design more efficient and allow for lower power consumption and smaller chip die size. This block diagram is explained properly in further sections.

3.9.2.4 Functional Overview: Media Independent Interface (MII) and Reduced MII (RMII)

i. Media Independent Interface

It provides a common interface between physical layer and MAC layer devices. The interface contains two distinct groups of signals: one for transmission and the other for reception. The following table 3.2 describes the signals used by the MII bus.

Table 3.2 MII Signals [38]

MII Signals	KSZ8863 Signals	Description
MTXEN	SMTXEN3	Transmit Enable
MTXER	SMTXER3	Transmit Error
MTXD3	SMTXD33	Transmit Data Bit 3
MTXD2	SMTXD32	Transmit Data Bit 2
MTXD1	SMTXD31	Transmit Data Bit 1
MTXD0	SMTXD30	Transmit Data Bit 0
MTXC	SMTXC3	Transmit Clock
MCOL	SCOL3	Collision Detection
MCRS	SCRS3	Carrier Sense
MRXDV	SMRXDV3	Receive Data Valid
MRXD3	SMRXD33	Receive Data Bit 3
MRXD2	SMRXD32	Receive Data Bit 2
MRXD1	SMRXD31	Receive Data Bit 1
MRXD0	SMRXD30	Receive Data Bit 0
MRXC	SMRXC3	Receive Clock

This MII provided by the KSZ8863MLL/FLL is connected to the device's third MAC. It operates in either PHY or MAC mode. The data interface is a nibble wide as the there are 4 transmit bits and 4 receive bits. Additional signals on the transmit side indicate when data is valid or when an error occurs during transmission. Similarly, the receive side has signals that convey when the data is valid and without physical layer errors. For half duplex operation, the SCOL signal indicates if a collision has occurred during transmission. Also, the MTXER signal indicates a

transmit error from the MAC device. Since the switch filters error frames, these MII error signals are not used by the KSZ8863FLL/MLL.

ii. Reduced MII (RMII)

The Reduced MII specifies a low pin count MII. RMII provides a common interface between physical layer and MAC layer devices, and has the following key characteristics:

- Ports 10Mbps and 100Mbps data rates.
- Uses a single 50 MHz clock reference (provided internally or externally).
- Provides independent 2-bit wide transmit and receive data paths.
- Contains two distinct groups of signals: one for transmission and one for reception.

The RMII provided by the KSZ8863RLL is connected to the device's third MAC. The following table 3.3 describes the signals used by the RMII bus.

Table 3.3 RMII Signals [38]

RMII Signals	KSZ8863 Signals	Description
REF_CLK	REFCLKI_3	Synchronous 50MHz clock reference
CRS_DV	SMRXDV3	Carrier Sense/Receive Data Valid
RXD1	SMRXD31	Receive Data Bit 1
RXD0	SMRXD30	Receive Data Bit 0
TX_EN	SMTXEN3	Transmit Enable
TXD1	SMTXD31	Transmit Data Bit 1
TXD0	SMTXD30	Transmit Data Bit 0
RX_ER	SMTXER3	Receiver Error

The KSZ8863RLL filters error frames, and thus does not implement the RX_ER output signal. As shown in table 3.4, to detect error frames from the RMII PHY devices, the SMTXER3 input signal of the KSZ8863RLL is connected to the RXER output signal of the RMII PHY device.

3.9.2.5 Functional Overview: Physical Layer Transceiver

1. 100Base-TX Transmitter

The 100BASE-TX transmitter consists of several functional blocks which convert synchronous 4-bit nibble data, as provided by the MII, to a scrambled MLT-3 125 Mb/s serial data stream. These whole sequences of steps are shown in figure 3.9.

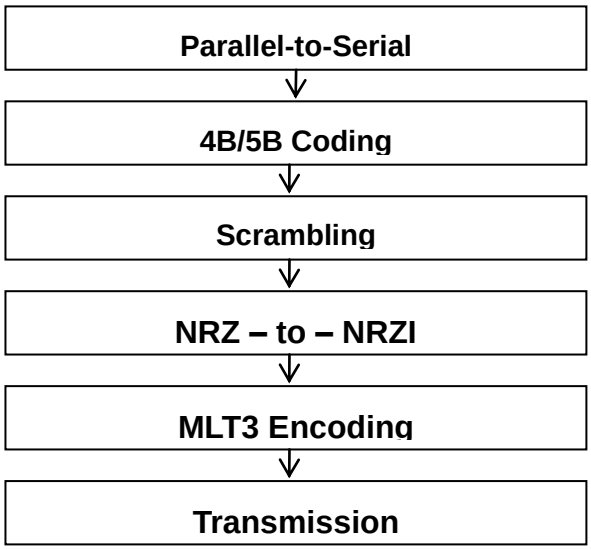

Figure 3.9 Sequence of steps at tranmitter side

Firstly, the parallel data is converted into the serial one as data is sent serially over the Ethernet cable to the other side. After this conversion, the steps followed, as shown in figure 3.16, are given below.

i. 4B/5B Coding

A run of 4 bits such as 0000 contains no transitions and so causes clocking problems for the receiver. This step solves this problem by assigning each block of 4 consecutive bits an equivalent word of 5 bits. The first 8-bits of the MAC preamble are replaced by the code-group pair 11000 10001 upon transmission. The encoder continues to replace subsequent 4B preamble and data nibbles with corresponding 5B code-groups. At the end of the transmit packet, upon the desertion of Transmit Enable signal from the MAC, the encoder injects the code-group pair 01101 00111 indicating the end of the frame. After this code-group pair, the encoder continuously injects IDLEs into the transmit data stream until the

next transmit packet is detected (reassertion of Transmit Enable). The 4B/5B codes from 0 to 9 are shown in table 3.4.

Table 3.4 4B/5B Coding [28]

Code	Value
0	11110
1	01001
2	10100
3	10101
4	01010
5	01011
6	01110
7	01111
8	10010
9	10011

In this table, 5 bit words are chosen so as to ensure that there will be at least two transitions per block of bits.

ii. Scrambler

It encodes a message at the transmitter to make the message unintelligible at the receiver not equipped with an appropriately set descrambling device.

iii. NRZ – to – NRZI Conversion

This conversion is done to get rid of the problem of long stream of ones. It can be explained as:

a. NRZ Encoding

The simplest digital signal representing a bit sequence uses just two voltage levels and represents a ‘1’ by the higher voltage and a ‘0’ by the lower voltage. This type of encoding is called NRZ (Non-Return-to-Zero).It is shown in figure 3.10.

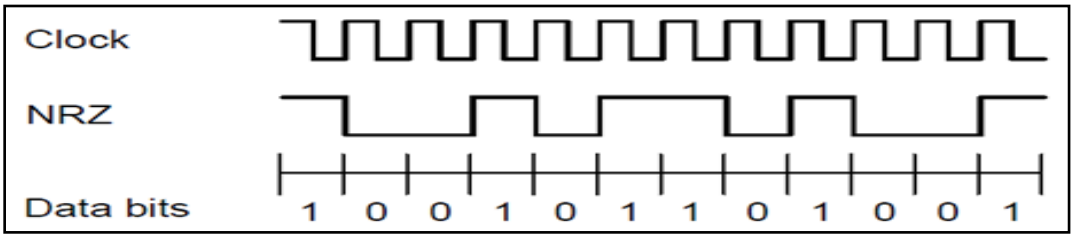

Figure 3.10 NRZ Coding [39]

Although simple, this method, for encoding digital information, has some serious drawbacks and is seldom used. First, it is difficult to keep the clocks of the source and receiver synchronized if there happen to be long sequences of ones or zeros. The receiver uses transitions in level to determine clock cycle boundaries. Second, it is impossible to distinguish between a long sequence of zeroes and the absence of a signal. Third, a long series of zeros or ones causes the average signal value, which is used to distinguish between high and low values, to drift. Thus, for many reasons, it is desirable to have frequent transitions between the high and low values [39].

b. NRZI Encoding

Another simple encoding method, called NRZI (Non-Return-to-Zero Inverted), changes level for a 1 bit and stays at the same level for a 0 bit. An example of an NRZI encoded signal is shown in figure 3.11.

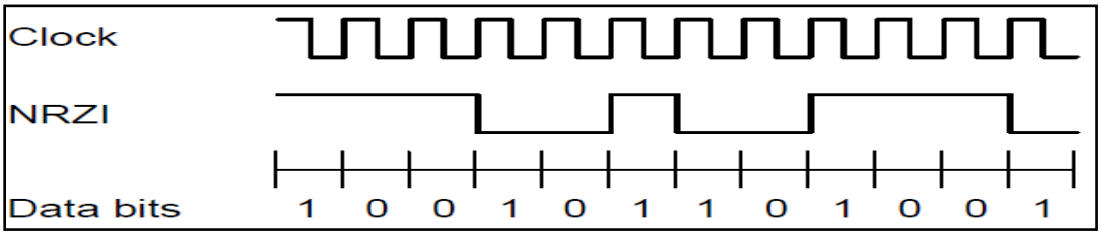

Figure 3.11 NRZI Coding [39]

This method gets rid of the problems associated with long strings of ones, but does nothing about long string of zeros.

iv. MLT-3 (Multi Level Threshold) Encoding

MLT-3encodingisusedtodecreasethehighfrequencycontentofthesignal.It uses three levels denoted by -1, 0, and 1 as shown in figure 3.12.The process cycles through the four

values -1, 0, +1, and 0. It moves to the next of the four states in a cyclical manner to transmit a 1 bit, and stays in the same state to transmit a 0 bit.

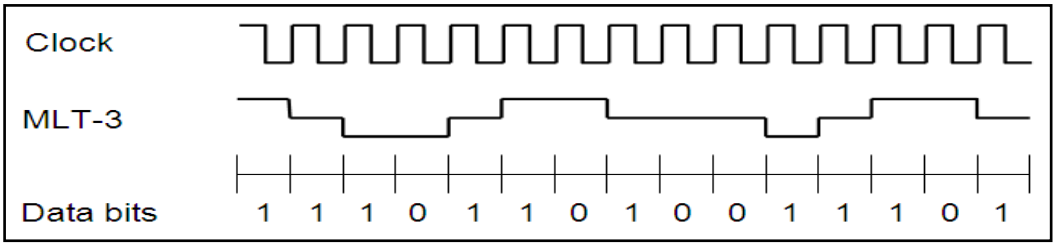

Figure 3.12 MLT-3 Encoding [39]

The fastest an MLT 3 signal can go through a complete cycle is four clock cycles. Thus, the high frequency limit of an MLT-3 signal will be about one-fourth that of a Manchester encoded signal. In Fast Ethernet (100Mbps), MLT-3 is applied to the signal generated by 4B/5B and NRZI. Higher order block codes such as 8B/10B and higher order multi-level methods such as MLT-5 are used in higher speed Ethernet networks [39]. Thus, binary to MLT-3 conversion is accomplished by converting the serial binary data stream output from the NRZI encoder into two binary data streams with alternately phased logic one events. These two binary streams are then fed to the twisted pair output driver which converts the voltage to current and alternately drives either side of the transmit transformer primary winding, resulting in a MLT-3signal.

2. 100Base-TX Receiver

The 100BASE-TX receiver consists of several functional blocks which convert the scrambled MLT-3 125 Mb/s serial data stream to synchronous 4-bit nibble data that is provided to the MII. Figure 3.13 shows all steps involved.

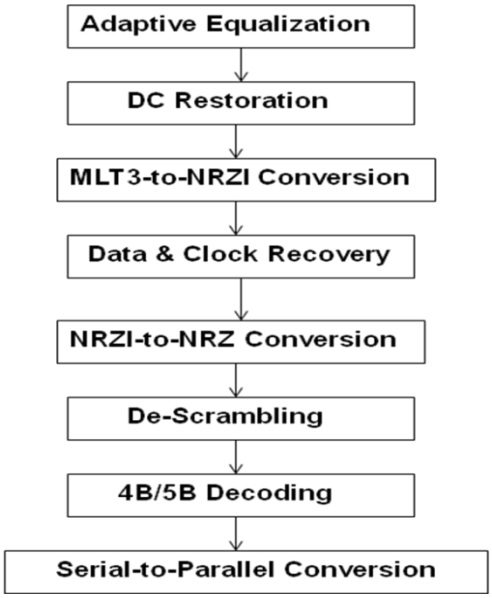

Figure 3.12 Sequence of steps at receiver side

All blocks of the above figure are explained one-by-one as:

i. Adaptive Equalization

In high-speed twisted pair signaling, the frequency content of the transmitted signal can vary greatly during normal operation, based primarily on the randomness of the scrambled data stream. This variation in signal attenuation, caused by frequency variations, must be compensated to ensure the integrity of the transmission. In order to ensure quality transmission when employing MLT-3 encoding, the compensation must be able to adapt to various cable lengths and cable types depending on the installed environment [40]. The selection of long cable lengths for a given implementation requires significant compensation which will over-compensate for shorter attenuating lengths. The Digital Equalizer removes ISI (inter symbol interference) from the receive data stream by continuously adapting to provide a filter with the inverse frequency response of the channel.

ii. DC Restoration

The DC restoration circuit is used to compensate for the effect of Base Line Wander (BLW) and to improve the dynamic range.

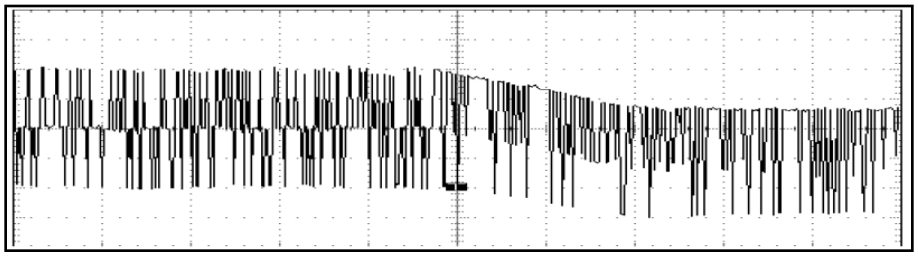

Figure 3.13 100BASE-TX BLW Event [40]

The digital oscilloscope plot provided above in Figure 3.14 illustrates the severity of the BLW event that can theoretically be generated during 100BASE-TX packet transmission.

BLW can generally be defined as the change in the average DC content, relatively short period over time, of an AC coupled digital transmission over a given transmission medium (i.e., copper wire). It results from the interaction between the low frequency components of a transmitted bit stream and the frequency response of the AC coupling component(s) within the transmission system. If the low frequency content of the digital bit stream goes below the low frequency pole of the AC coupling transformers then the droop characteristics of the transformers will dominate resulting in potentially serious BLW.

iii. MLT-3 to NRZI Decoder

The KSZ8863FLLI decodes the MLT-3 information from the Digital Adaptive Equalizer block to binary NRZI data.

iv. NRZI to NRZ

In a typical application, the NRZI to NRZ decoder is required in order to present NRZ formatted data to the descrambler.

v. Serial to Parallel

The 100BASE-TX receiver includes a Serial to Parallel converter which supplies 5-bit wide data symbols to the PCS (Personal Communication System) Rx state machine.

vi. Descrambler

A serial descrambler is used to de-scramble the received NRZ data. The descrambler has to generate an identical data scrambling sequence in order to recover the original

unscrambled data from the scrambled data. In order to maintain synchronization, the descrambler must continuously monitor the validity of the unscrambled data that it generates.

vii. 4B/5B Decoder

This step translates incoming 5B code-groups into 4B nibbles.

3. 100BASE-FX Operation

100BASE-FX operation is similar to 100BASE-TX operation with the differences being that the scrambler/de-scrambler and MLT3 encoder/decoder are bypassed on the transmission and reception.

4. 100BASE-FX Signal detection

In 100BASE-FX operation, FXSD (Fiber Signal Detect) is usually connected to the fiber transceiver SD (Signal Detect) output pin. The fiber signal threshold can be selected by register 192 for Port 1. When FXSD is less than threshold, no fiber signal is detected and a Far-End-Fault (FEF) is generated. When a FEF is detected, the KSZ8863FLL/MLL/RLL signals its fiber link partner that a FEF has occurred by sending 84 1's followed by a zero in the idle period between frames. When FXSD is over the threshold, fiber signal is detected. Alternatively, the designer may choose not to implement the FEF feature. In this case, FXSD input pin is tied high to force 100BASE-FX mode [39].

5. 10 Mb/s Stream Contents

There are distinct differences between a 10 Mb/s and a100 Mb/s stream. This section describes how 10 Mbps stream contents are actually transported over the physical medium (i.e., CAT5cable, etc.).For this purpose, these streams of data are transmitted using Manchester encoding.

In Ethernet, Manchester encoding encodes a logical ‘0’ as a high-to-low transition and logical ‘1’ as a low-to-high transition. An example of a Manchester encoded sequence is shown in figure 3.15 below.

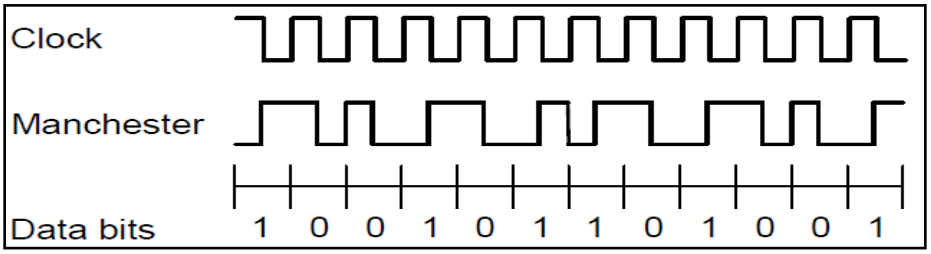

Figure 3.14 Manchester Coding [40]

Manchester encoding is used because it provides high reliability and the ability to extract the clock from the data stream. This encoding method is used in 10Mbps 10BASE-T Ethernet networks. It solves the problems mentioned previously in connection with NRZ encoding [40]. However, since the signal alternates level every clock cycle, Manchester encoding has a broader frequency spectrum than NRZ or requires double the bandwidth of the data to be transmitted.

3.10 Fiber Optic Communication

In the Ethernet module, it was desired to use fiber optic transceiver along with copper transceiver to get aware of this new technology. Here, the aim was to convert serial signal from controller’s UART to serial fiber optic signal. For this, the requirement was to use low link fiber optic transceivers with data rate of approximate 5MBd to 10MBd. Firstly, a list of transceivers was made after studying the transceivers of many companies as shown in table 3.5. This list categorizes the transceivers according to their cost, speed and reach. With the help of this study, it was easier to choose low link fiber equipment for serial signal transmission - HFBR-1414 (Transmitter) and HFBR-2412 (Receiver) of Agilent Corporation realizing electrical / optical and optical/electrical conversion.

Table 3.5 List of Transceivers

S.No.	Transceivers	Speed	Distance	Source Of Light	Wavelength Of Light	Type Of Cable	Type Of Connector	Cheapest One
1	Transmitter + Receiver	5MBd	Upto 30m	LED	660nm	POF	Versatile Link Connectors	HFBR-2521Z [R] HFBR-1521Z [T]
2	Transmitter + Receiver	5MBd	Upto 2Km	LED	820nm	Multi-mode Fiber	ST	HFBR-2412Z [R HFBR-1414Z [T
3	Transmitter + Receiver	10 MBd	Upto 50m (POF) Upto 500m (HCS)	LED	650 nm	1 mm POF, HCS	Versatile Link Connectors	HFBR-2528Z [R] HFBR-1528Z [T]
4	Transmitter + Receiver	20 to 160 MBd	2700m to 500m	LED	820nm	Multi-mc	ST	HFBR-2416Z [R HFBR-1414Z [T
5	Transmitter + Receiver	50 MBd	Upto 50m	LED	650 nm	1 mm POF	Versatile Link Connectors	AFBR-2624Z [R] AFBR-1629Z [T]
6	Transceiver	125MBd	Upto 2Km	LED	1300nm	Multi-mc Fiber	SC,ST	AFBR-5803Z
7	SFP	125MBd	Upto 10Km	LASER [100Base-LX]	1300nm	Single-mode Fiber	LC	AFCT-5971ALZ
8	Transceiver	155 MBd	Upto 2Km	LED	1300nm	Multimode Fiber	SC,ST	AFBR-5805Z
9	SFP	Upto 1.25Gbps	550 m	LASER [1000Base-SX]	850 nm	Multimode Fiber	LC	AFBR-5701APZ [Avago]
10	SFP	Upto 1.25Gbps	0.5 m to 550 m [M 0.5 m to 10 Km [S]	LASER [1000Base-LX]	1310 nm	Multimode Singlemode	LC	AFCT-5710PZ [Avago]
11	SFP	Upto 1.25Gbps	80 Km	LASER [1000Base-ZX]	1550 nm	Singlemode	LC	FTLF1518P1BTL [Finisar Corporat
12	SFP	2.457 Gb/s to 3.072 GB/s	500m to 300m	LASER [1000Base-SX]	850nm	Multimode	LC	AFBR-57J5APZ [Avago]
13	SFP	1.0625 GBd 2.125 GBd 4.25 GBd	500m [50 µm] 300m [62.5 µm] 300m [50 µm] 150m [62.5 µm] 150m [50 µm] 70m [62.5 µm]	LASER [1000Base-SX]	850 nm	Multimode	LC	AFBR-57R5APZ [Avago]
14	SFP	1.0625 GBd 2.125 GBd 4.25 GBd	10Km 10Km 4Km	LASER [1000Base-LX]	1310 nm	Singlemode	LC	AFCT-57R5APZ [Avago]
15	SFP+	10 GBd	300m to 500m	LASER [10GBASE-SR]	850 nm	Multimode	LC	AFBR-703SMZ [Avago]
16	SFP+	10 GBit/s	10Km	LASER [10GBASE-LR]	1310 nm	Multimode	LC	AFCT-701SDZ [Avago]

This list gives an overview of different types of transceivers used for different speeds and distances. From this list, an engineer can select any transceiver according to the application he is working upon.

3.10.1 Serial Signal Transmission through Fiber equipment

Here, the serial signal transmission through fiber optic cable is tested. To provide data in the form of pulses, a simple ON/OFF switch is used. The transmitter-receiver combination, HFBR-1414/2412 (Figure 3.16), allows data rate of 5MBd over a distance of 1500m. A *62.5/125µm multimode fiber cable* of 1m is used for only testing purpose. Also, the design is using *ST series multimode fiber connector* which exhibits an optimized cylindrical sleeve with a cross section designed to expand uniformly when the ferrules are inserted. Hence, the constant circumferential pressure provides accurate alignment even when the ferrule diameters differ slightly. The HFBR-1414 optical fiber transmitter contains a high efficiency light power excitation Gallium Aluminum Arsenide (GaAlAs) light emitter. The drive current of this light emitter is 48 mA. The operating temperature range is -40°C to 85°C and propagation delay per meter is 5ns[41].

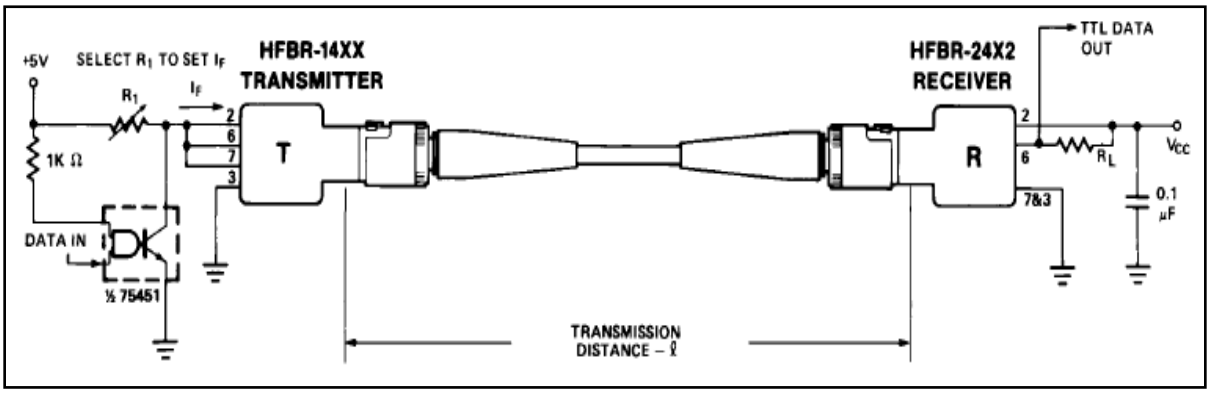

Figure 3.15 HFBR1414 and HFBR 2412 [42]

Ideally, when the switch send out data, it needs to pass the data through fiber optic transmitter for sending that signal, in light form, for long-distance transmission by optical fiber. The light signal passes through receiver at the other end of the optical fiber for restoring into electrical signal. Because the output signal power of the switch is weak, it could not be transmitted for a long distance. Thus, the figure 3.16 shows that before the optical/electrical and electrical/optical conversion a power drive chip SN75451/74ACT08 is required to increasing the drive capability of the input device. But in actual, many problems were faced while making

transmission possible between transmitter and receiver. Firstly, SN75451 was used as a power driver and all connections were made according to datasheet. The circuit showing all these connections is given in figure 3.17. But after repeated efforts, no signal was coming at the transmitter output and so at the receiver.

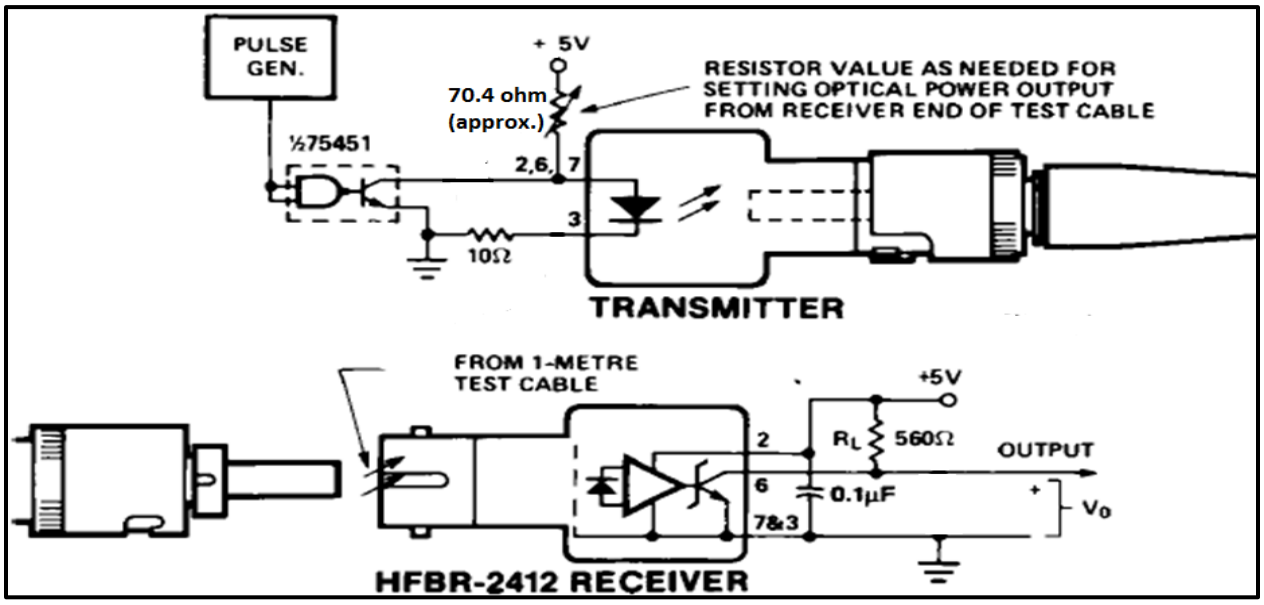

Figure 3.16 Circuit Diagram for Fiber Optic Transmission [42]

The factors causing this failure can be:

- Insufficient drive current.
- Over drive current resulting in the damage of the driver.

Secondly, another driver, 74ACT08, was used which is a good alternative for SN75451. In this case also, the same results were obtained. The efforts are being made to check these problems for correcting them as soon as possible.

3.11 Fiber vs. Copper – Current and Future Co-existence

While fiber cable is preferred for noise immunity, twisted pair cabling also has a role in substations. Within control room rack cabinets, copper cabling is safely used for short Ethernet interconnections. The same twisted pair cable and RJ-45 port connectors can be used for both 10 Mb and 100Mb speeds, simplifying installations. The assumption has been that copper cabling is less costly, so most RTU, PLC and IED manufacturers use RJ-45 ports on their products for both

lower cost and 10/100 Mb compatibility. A recent analysis of the installation costs of copper vs. fiber (typical for new substations and upgrades) shows surprisingly little difference. In a three-year study by the Fiber Optics LAN Section (FOLS) of the Telecommunications Industry Association (TIA) and Pearson Technologies, copper and fiber installations were modeled for initial cost and found to be about equal. Although the scope of the TIA study does not cover on-going maintenance costs, the argument can be made that fiber is actually less expensive to maintain than copper. Thus, a mix of copper and fiber cabling is present in latest substations, co-existing and complementing each other.

In this chapter, every concept regarding the Ethernet communication is explained be it, its principle, its frame, its cabling types or its nomenclature and significance. The designed Ethernet module is also explained in detail. The PIC micro controller used in this module is discussed with the help of its simple block diagram. Also, the whole working of the micrel's Ethernet switch is highlighted with every smallest detail important to understand it. Ethernet communication can be implemented using copper or fiber optic technology. Hence, lastly, the implementation circuits of fiber optic communication and the encountered problems are thoroughly discussed.

CHAPTER 4

STUDY OF TCP/IP STACK

4.1 Introduction

The stack is divided into multiple layers (Figure 4.1), where each layer accesses services from one or more layers directly below it. Per specifications, many of the TCP/IP layers are “live”, in the sense that they not only act when a service is requested, but also when events like time-out or new packet arrival occurs.

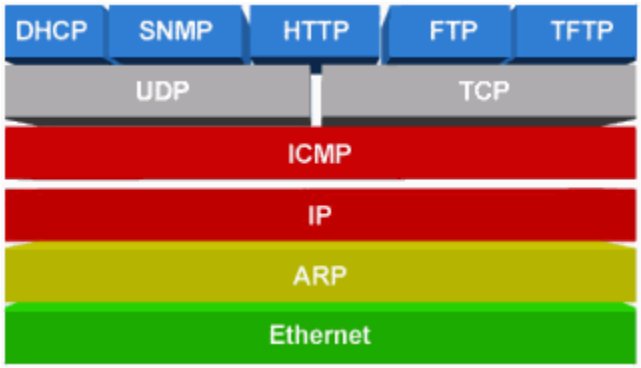

Figure 4.1 TCP/IP Stack [43]

The figure 4.1 shows the four layers of the TCP/IP model. These four layers are data link or network interface layer including Ethernet, network or internet layer including IP and ICMP protocols, transport layer including UDP or TCP protocols and the application layer consisting of various protocols like HTTP, FTP etc. All these layers are explained, in detail, in further sections.

4.2 Network Interface Layer

4.2.1 Ethernet

Ethernet is a data link and physical layer protocol defined by the IEEE 802.3 specification. It comes in many flavors, defined by:

- Maximum Bit Rate (Mbits/s): 10, 100, 1000, etc.
- Mode of Transmission: Broadband, Baseband
- Physical Transmission Medium: Coax, Fiber, UTP, etc.

4.2.2 Address Resolution Protocol

ARP is a protocol that will translate the IP address to the physical address of the destination host. It uses a lookup table (sometimes referred to as the ARP cache) to perform this translation. When the address is not found in the ARP cache, a broadcast is sent out in the network with a special format called the ARP request. If one of the machines in the network recognizes its own IP address in the request, it will send an ARP reply back to the requesting host. The reply will contain the physical hardware address of the host and source route. Both this address and the source route information are stored in the ARP cache of the requesting host. All subsequent datagrams to this destination IP address can now be translated to a physical address, which is used by the device driver to send out the datagram in the network.

4.2.2.1 ARP Packet

A general ARP packet is shown below in figure 4.2.

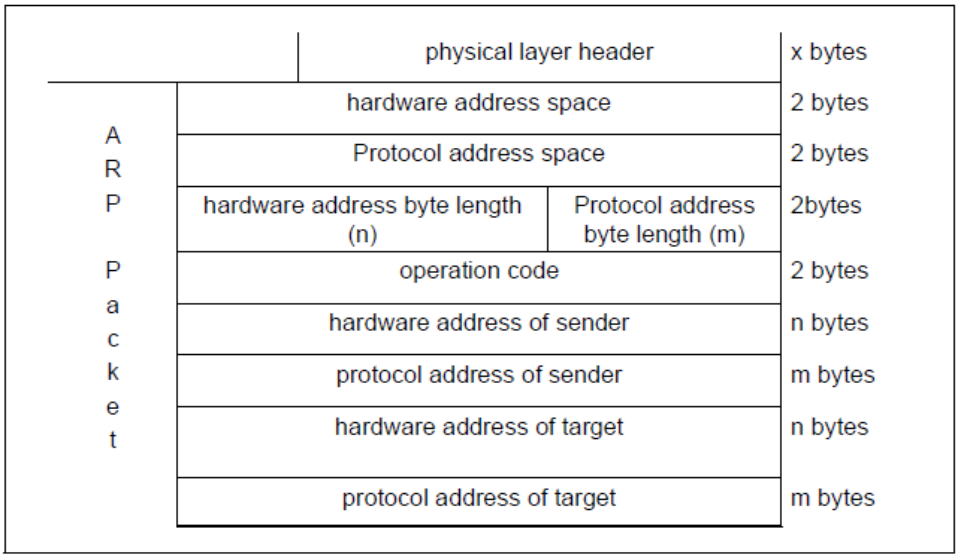

Figure 4.2 ARP: Request/reply packet [44]

Where:

- **Hardware address space:** Specifies the type of hardware.
- **Protocol address space:** Specifies the type of protocol.
- **Hardware address length:** Specifies the length (in bytes) of the hardware addresses in this packet.

- **Protocol address length:** Specifies the length (in bytes) of the protocol addresses in this packet.
- **Operation code:** Specifies whether this is an ARP request (1) or reply (2).
- **Source/target hardware address:** Contains the physical network hardware addresses.
- **Source/target protocol address:** Contains the protocol addresses.

For the ARP request packet, the target hardware address is the only undefined field in the packet.

4.3 Network Layer Protocols

This layer addresses messages and routes them across the network and exchanges data between the systems independent of the network topology and the media used. The protocols of this layer are explained below.

4.3.1 Internet Protocol (IP)

IP is the protocol that hides the underlying physical network by creating a virtual network view. It is an unreliable, best-effort, and connectionless packet delivery protocol. Note that best-effort means that the packets sent by IP might be lost, arrive out of order, or even be duplicated. IP assumes higher layer protocols will address these anomalies. One of the reasons for using a connectionless network protocol was to minimize the dependency on specific computing centers that used hierarchical connection-oriented networks [44].

IP addresses are represented by a 32-bit unsigned binary value. It is usually expressed in a dotted decimal format. For example, 9.167.5.8 is a valid IP address. The numeric form is used by IP software. The mapping between the IP address and an easier-to-read symbolic name, for example, myhost.ibm.com, is done by the *Domain Name System (DNS).*

4.3.1.1 The IP address

To identify a host on the Internet, each host is assigned an address, the *IP address*, or in some cases, the *Internet address*. The IP address consists of a pair of numbers [44]:

IP address = <network number><host number>

The *network number* portion of the IP address is administered by one of the three Regional Internet Registries (RIR): American Registry for Internet Numbers (ARIN), Reseaux IP Europeans (RIPE) and Asia Pacific Network Information Centre (APNIC).

IP addresses are 32-bit numbers represented in a *dotted decimal* form (as the decimal representation of four 8-bit values concatenated with dots). For example, 128.2.7.9 is an IP address with 128.2 being the network number and 7.9 being the host number. This numeric form is used by the IP software. The mapping between the IP address and an easier-to-read symbolic name, for example, myhost.ibm.com, is done by the Domain Name System (DNS).

4.3.1.2 IP subnets

Due to the explosive growth of the Internet, the principle of assigned IP addresses became too inflexible to allow easy changes to local network configurations. Those changes might occur when:

- A new type of physical network is installed at a location.
- Growth of the number of hosts requires splitting the local network into two or more separate networks.
- Growing distances require splitting a network into smaller networks, with gateways between them.

To avoid requesting additional IP network addresses, the concept of IP subnetting was introduced. The assignment of subnets is done locally. The entire network still appears as one IP network to the outside world. The subnet address is a bit mask that actually defines the scope of the network. The host number part of the IP address is subdivided into a second network number and a host number. This second network is termed a *sub network* or *subnet*. The main network now consists of a number of subnets. The IP address is interpreted as:

<network number><subnet number><host number>

The combination of subnet number and host number is often termed as the local address or the local portion of the IP address. Subnetting is implemented in a way that is transparent to remote

networks. A host within a network, that has subnets, is aware of the subnetting structure. A host in a different network is not. This remote host still regards the local part of the IP address as a host number. If the IP address is 192.168.5.100, and the subnet mask is specified as 255.255.255.0, the stack will assume that addresses in the range of 192.168.5.x are on the same subnet as the above specified IP address is, and that packets sent to any of those addresses won't have to be routed anywhere else.

4.3.1.3 Methods of delivery: Unicast, Broadcast, Multicast, and Anycast

The majority of IP addresses refer to a single recipient; this is called a *unicast* address. Unicast connections specify a one-to-one relationship between a single source and a single destination. Additionally, there are three special types of IP addresses used for addressing multiple recipients: broadcast addresses, multicast addresses, and anycast addresses. These addresses are shown in figure 4.3.

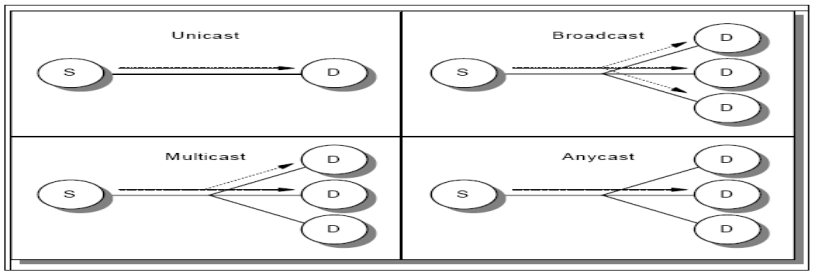

Figure 4.3 IP: Packet delivery modes [44]

A *connectionless* protocol can send unicast, broadcast, multicast, or anycast messages as shown above. A *connection-oriented* protocol can only use unicast addresses (a connection must exist between a specific pair of hosts).

4.3.1.4 IP datagram and its format

The unit of transfer in an IP network is called an IP datagram. It consists of an IP header and data relevant to higher-level protocols. The format of an IP datagram can be explained in figure 4.4. The header of this datagram has a minimum length of 20 octets as illustrated.

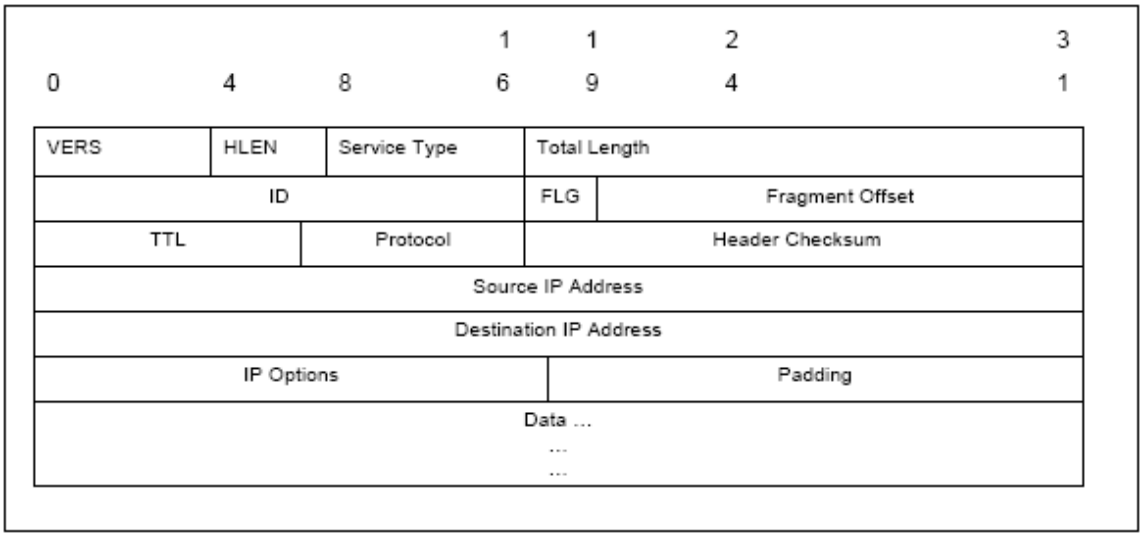

Figure 4.4 IP: Format of an IP datagram header [44]

The various fields of this datagram format are defined below as:

i. **VERS:** This field contains the IP protocol version. The current version is 4. Version 5 is an experimental version and version 6 is the version for IPv6.

ii. **HLEN:** It indicates the length of the IP header counted in 32-bit quantities. This does not include the data field.

iii. **Service Type:** The service type is an indication of the quality of service requested for this IP datagram.

iv. **Total Length:** It represents the total length of the datagram, header and data.

v. **Identification:** It is a unique number assigned by the sender to aid in reassembling a fragmented datagram. Each fragment of a datagram has the same identification number.

vi. **Flags:** This field contains control flags as:

***0*:** Reserved, must be zero.

***DF (Do not Fragment)*:**0 means allow fragmentation; 1 means do not allow fragmentation.

***MF (More Fragments)*:**0 means that this is the last fragment of the datagram; 1 means that additional fragments will follow.

vii. **Fragment Offset:** This is used to aid the reassembly of the full datagram.

viii. **Time to Live (TTL):** This field specifies the time (in seconds) the datagram is allowed to travel. Theoretically, each router, processing this datagram, is supposed to subtract its processing time from this field. In practice, a router processes the datagram in less than 1 second. Therefore, the router subtracts one from the value in this field. When the value, in this field, reaches zero, it is assumed that this datagram has been traveling in a closed loop and is discarded. The initial value should be set by the higher-level protocol that creates the datagram.

ix. **Protocol Number:** This field indicates the higher-level protocol to which IP should deliver the data in this datagram. Example: Internet Control Message Protocol (ICMP)**,** IP (IP encapsulation)**,** Transmission Control Protocol (TCP) or User Datagram Protocol (UDP).

x. **Header Checksum:** This field is a checksum for the information contained in the header. If the header checksum does not match the contents, the datagram is discarded.

xi. **Source IP Address:** It represents 32-bit IP address of the host sending this datagram.

xii. **Destination IP Address:** It represents 32-bit IP address of the destination host for this datagram.

xiii. **Options:** An IP implementation is not required to be capable of generating options in a datagram. However, all IP implementations are required to be able to process datagrams containing options. The Options field is variable in length (there can be zero or more options).

xiv. **Padding:** If an option is used, the datagram is padded with all-zero octets up to the next 32-bit boundary.

xv. **Data:** It indicates the data contained in the datagram. It is passed to the higher-level protocol specified in the protocol field.

4.3.2 Internet Control Message Protocol (ICMP)

When a router or a destination host must inform the source host about errors in datagram processing, it uses the Internet Control Message Protocol (ICMP). ICMP uses IP as though ICMP

were a higher-level protocol (that is, ICMP messages are encapsulated in IP datagrams). However, ICMP is an integral part of IP and must be implemented by every IP module. It is used to report errors, *not* to make IP reliable. Datagrams can still be undelivered without any report on their loss. Reliability must be implemented by the higher-level protocols using IP services.

4.3.2.1 ICMP messages

ICMP messages are sent in IP datagrams. The IP header has a protocol number of 1 (ICMP) and a type of service of zero (routine). The IP data field contains the ICMP message shown in Figure 4.5.

0 8	16	31
Identifier	Sequence number	Checksum
ICMP data (depending on the type of message)		

Figure 4.5 I CMP: Message format [44]

- **Type:** Specifies the type of the message. For example:

  0 Echo reply

  8 Echo

  Echo is used to detect, if another host is active in the network. It is used by the Ping command. The sender initializes the identifier, sequence number, and data field. The datagram is then sent to the destination host. The recipient changes the type to Echo Reply and returns the datagram to the sender.
- **Code** Contains the error code for the datagram reported by this ICMP message. The interpretation is dependent on the message type.
- **Checksum** Contains the checksum for the ICMP message starting with the ICMP Type field. If the checksum does not match the contents, the datagram is discarded.
- **Data** Contains information for this ICMP message. Typically, it will contain the portion of the original IP message for which this ICMP message was generated. Each of the ICMP messages is described individually.

4.4 Transport layer protocols

The most important and commonly used protocols of the TCP/IP transport layer are:

- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

By building on the functionality provided by the Internet Protocol (IP), the transport protocols deliver data to applications executing in the internet. This is done by making use of ports. The transport protocols can provide additional functionality such as congestion control, reliable data delivery, duplicate data suppression, and flow control as is done by TCP.

4.4.1 Transmission Control Protocol (TCP)

TCP is a *connection-oriented* protocol, unlike UDP, which is *connectionless*.TCP, provides considerably more facilities for applications than UDP. Specifically, this includes error recovery, flow control, and reliability. Most of the user application protocols, such as Telnet and FTP, use TCP. The primary purpose of TCP is to provide a reliable logical circuit or connection service between pairs of processes. It does *not* assume reliability from the lower-level protocols (such as IP), so TCP must guarantee this itself. TCP can be characterized by the following facilities it provides for the applications using it [45]:

- **Stream data transfer**: TCP transfers a continuous stream of bytes through the network. The application does not have to bother about chopping the data into basic blocks or datagrams. TCP does this by grouping the bytes into TCP segments, which are IP layer for transmission to the destination. Also, TCP itself decides how to segment the data, and it can forward the data at its own convenience.
- **Reliability:** TCP assigns a sequence number to each byte transmitted, and expects a positive acknowledgment (ACK) from the receiving TCP layer. If the ACK is not received within a timeout interval, the data is retransmitted. Because the data is transmitted in blocks (TCP segments), only the sequence number of the first data byte in the segment is sent to the destination host. The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order, and to eliminate duplicate segments.
- **Flow control:** The receiving TCP, when sending an ACK back to the sender, also indicates to the sender the number of bytes it can receive (beyond the last received TCP segment) without causing overrun and overflow in its internal buffers. This is sent in the ACK in the

form of the highest sequence number it can receive without problems. This mechanism is also referred to as a window-mechanism.

- **Logical connections:** The reliability and flow control mechanisms described here requires that TCP initializes and maintains certain status information for each data stream. The combination of this status, including sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is uniquely identified by the pair of sockets used by the sending and receiving processes.

4.4.1.1 TCP segment format

Figure 4.6 shows the TCP segment format. The various fields in this datagram format are defined as follows:

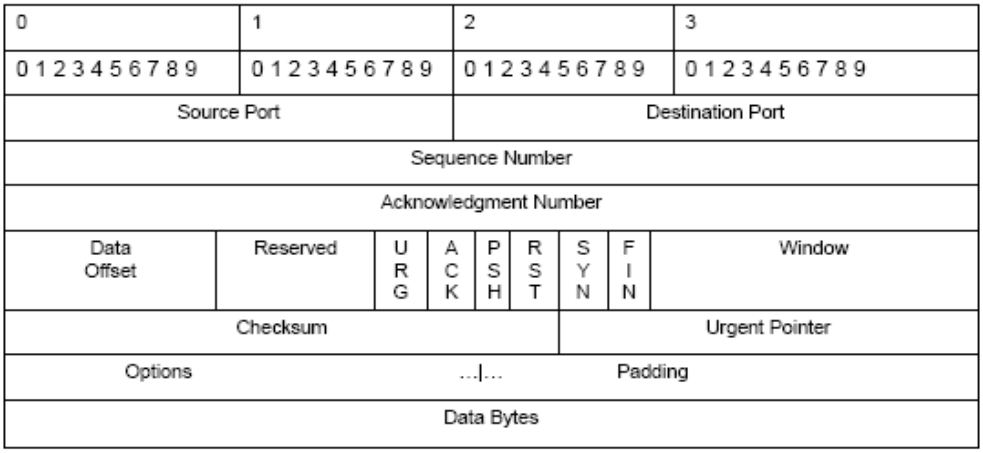

Figure 4.6 TCP: Segment format [44]

- **Source Port:** It is the 16-bit source port number, used by the receiver to reply.
- **Destination Port :** It represents the 16-bit destination port number.
- **Sequence Number:** It represents the sequence number of the first data byte in this segment. If the SYN control bit is set, the sequence number is the initial sequence number (n) and the first data byte is n+1.
- **Acknowledgment Number:** If the ACK control bit is set, this field contains the value of the next sequence number that the receiver is expecting to receive.
- **Data Offset:** It indicates the number of 32-bit words in the TCP header. It indicates where the data begins.
- **Reserved:** Six bits reserved for future use; must be zero.

- **URG:** Indicates that the urgent pointer field is significant in this segment.
- **ACK:** Indicates that the acknowledgment field is significant in this segment.
- **PSH:** Push function.
- **RST:** Resets the connection.
- **SYN:** Synchronizes the sequence numbers.
- **FIN;** No more data from sender.
- **Window:** Used in ACK segments. It specifies the number of data bytes, beginning with the one indicated in the acknowledgment number field that the receiver is willing to accept.
- **Checksum:** The 16-bit one's complement of the one's complement sum of all 16 bit words in a pseudo-header, the TCP header, and the TCP data. While computing the checksum, the checksum field itself is considered zero. The pseudo-header is the same as that used by UDP for calculating the checksum. It is a pseudo-IP-header, only used for the checksum calculation, with the format.
- **Urgent Pointer:** It points to the first data octet following the urgent data. Only significant when the URG control bit is set.
- **Options:** Just as in the case of IP datagram options, options can be either
  → A single byte containing the option number.
  → A variable length option.

4.4.1.2 Establishing a TCP connection

Before any data can be transferred, a connection has to be established between the two processes. This process of making connection is known as a three-way handshake which is shown in figure 4.7.

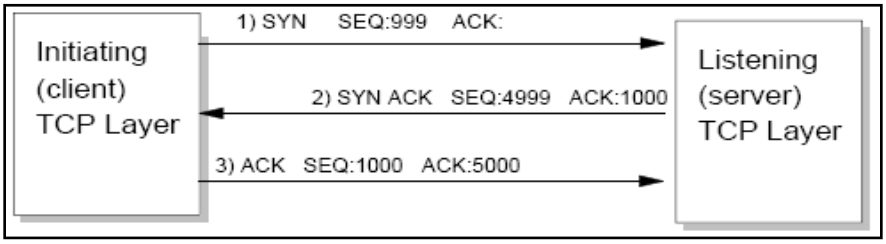

Figure 4.7 TCP: Connection establishment [44]

The exchanged TCP segments include the initial sequence numbers from both sides, to be used on subsequent data transfers. Closing the connection is done implicitly by sending a TCP segment with the FIN bit (no more data) set. Because the connection is full-duplex (that is, there are two independent data streams, one in each direction), the FIN segment only closes the data transfer in one direction. The other process will now send the remaining data; it still has to transmit and also ends with a TCP segment, where the FIN bit is set. The connection is deleted (status information on both sides) after the data stream is closed in both directions.

4.5 Application Layer Protocols

4.5.1 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol is a protocol designed to allow the transfer of Hypertext Markup Language (HTML) documents. HTML is a tag language used to create hypertext documents.

As shown in figure 4.8, HTTP is based on the request-response activity. An HTTP transaction can be divided into four steps:

- A client, running an application called a browser, establishes a connection with a server.
- The browser sends a request to the server in the form of a request method.
- The server responds the browser with a status line, including the message's protocol version and a success or error code, followed by a message containing server information, entity information, and possible body content.
- The connection is closed.

The working of the client-server model can be explained more thoroughly with the help of the following diagram given in figure 4.8.

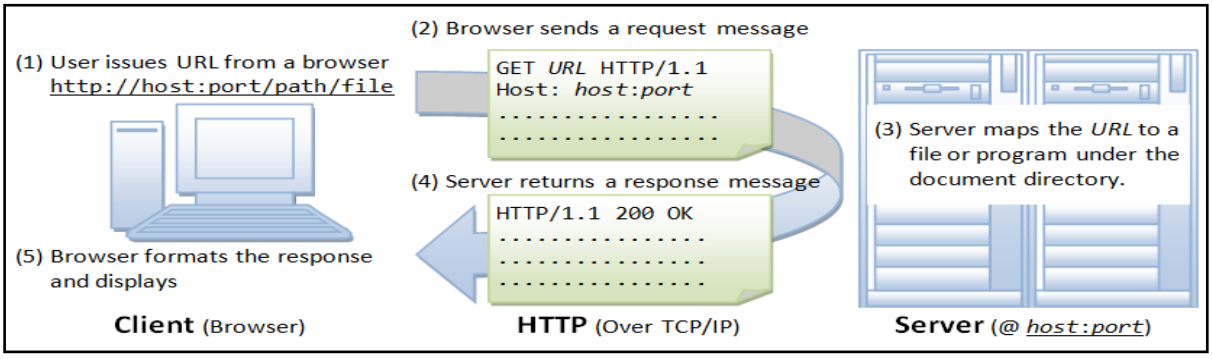

Figure 4.8 Process behind opening of various sites [46]

This figure shows that whenever a URL is issued from the browser to get a web resource using HTTP, e.g. http://www.test101.com/index.html, the browser turns the URL into a *request message* and sends it to the HTTP server. The HTTP server interprets the request message, and returns an appropriate response message, which is either the resource requested or an error message.

4.5.1.1 HTTP Request and Response Messages

HTTP client and server communicate by sending text messages. The client sends a *request message* to the server. The server, in turn, returns a *response message*.

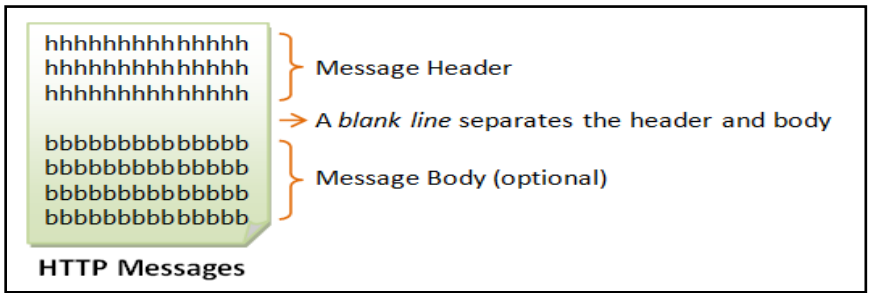

Figure 4.9 HTTP Message Body [46]

As shown in figure 4.9, an HTTP message consists of a *message header* and an optional *message body*, separated by a *blank line*. Both the HTTP request and response messages are explained individually as:

1. HTTP Request Message

The format of an HTTP request message is as shown in figure 4.10.

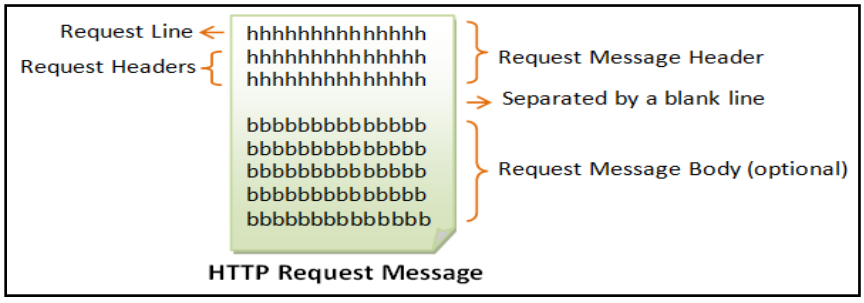

Figure 4.10 Body of HTTP request message [46]

The body parts of an HTTP request message, as shown in figure above, are explained as follows:

a. Request Line

The first line of the header is called the *request line*, followed by optional *request headers*. The request line has the following syntax:

request-method-name request-URI HTTP-version

- *request-method-name*: HTTP protocol defines a set of request methods, e.g., GET, POST, HEAD, and OPTIONS. The client can use one of these methods to send a request to the server.
- *request-URI: specifies the resource requested.*
- *HTTP-version*: Two versions are currently in use: HTTP/1.0 and HTTP/1.1.

b. Request Headers

The request headers are in the form of name: value pairs. Multiple values, separated by commas, can be specified as:

request-header-name: *request-header-value1*, *request-header-value2*, ...

c. Example

The following figure 4.11shows a sample HTTP request message:

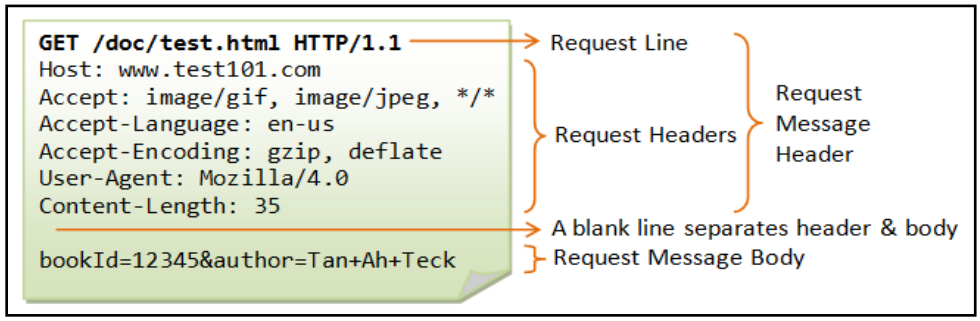

Figure 4.11 An example of an HTTP request message [46]

In this figure, each component of an HTTP request message is clearly indicated.

2. HTTP Response Message

The format of the HTTP response message is shown in figure 4.12.

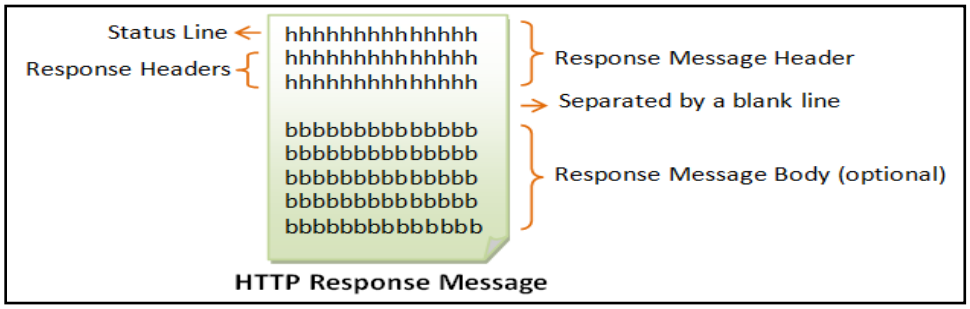

Figure 4.12Body of HTTP response message [46]

The body parts of an HTTP response message, as shown in figure above, are explained as follows:

a. Status Line

The first line is called the *status line*, followed by optional response header(s).The status line has the following syntax:

HTTP-version status-code reason-phrase

- *HTTP-version*: The HTTP version can be either HTTP/1.0 or HTTP/1.1.
- *status-code*: a 3-digit number generated by the server to reflect the outcome of the request.
- *reason-phrase*: gives a short explanation to the status code.
- Common status code and reason phrase are "200 OK", "404 Not Found", "403 Forbidden", "500 Internal Server Error".

b. Response Headers

The response headers are in the form name:value pairs:

response-header-name: *response-header-value1*, *response-header-value2*, ...

The response message body contains the resource data requested.

c. Example

The following figure 4.13 shows a sample response message:

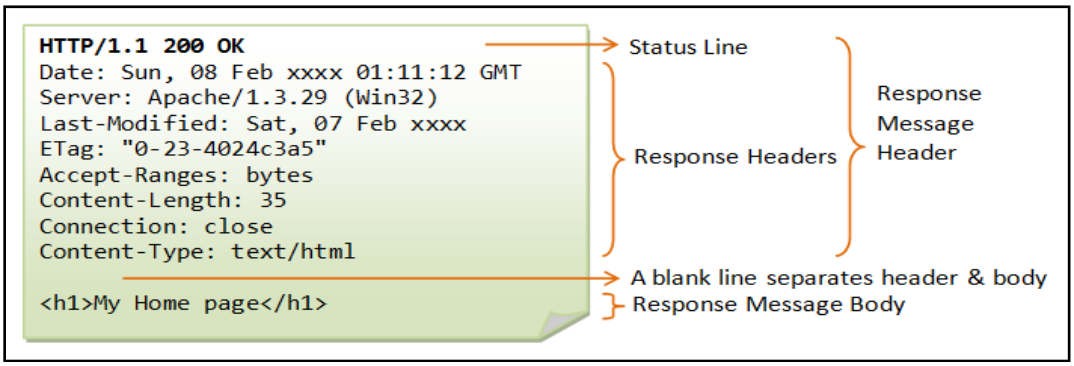

Figure 4.13 An example of an HTTP request message [46]

In this figure, each component of an HTTP request message is clearly indicated.

4.5.1.2 HTTP Methods

The two most common methods for HTTP/1.0 are GET and POST. These are defined below as:

i. The GET Method

GET is the most common HTTP request method. A client can use the GET request method to request (or "get") for a piece of resource from an HTTP server. A GET request message takes the following syntax:

GET *request-URIHTTP-version*
(optional request headers)
(blank line)
(optional request body)

- The keyword GET is case sensitive and must be in uppercase.
- *request-URI*: specifies the path of resource requested, which must begin from the root "/" of the document base directory.
- *HTTP-version*: Either HTTP/1.0 or HTTP/1.1. This client *negotiates* the protocol to be used for the current session. For example, the client may request to use HTTP/1.1. If the server does not support HTTP/1.1, it may inform the client in the response to use HTTP/1.0.

- The client uses the optional request headers (such as Accept, Accept-Language, and etc) to *negotiate* with the server and ask the server to deliver the preferred contents (e.g., in the language that the client preferred).
- GET request message has an optional request body which contains the query string.

ii. The POST Method

The POST request method is used to "post" additional data up to the server (e.g., submitting HTML form data or uploading a file). Issuing an HTTP URL from the browser always triggers a GET request. To trigger a POST request, an HTML form can be used with attribute method="post". For submitting HTML form data, POST request is the same as the GET request except that the URL-encoded query string is sent in the request body, rather than appended behind the *request-URI*. The POST request takes the following syntax:

POST *request-URIHTTP-version*
Content-Type: mime-type
Content-Length: number-of-bytes
(other optional request headers)
(URL-encoded query string)

In this format, MIME stands for Multipurpose Internet Mail Extension. It is the format of Internet Message bodies. Request header′s Content-Type and Content-Length is necessary in the POST request to inform the server `about` the media type and the length of the request body.

4.6 Frame/Packet Encapsulation

Each layer of the protocol stack is responsible for a particular level of functionality. As an example, the physical layer is concerned with the actual electrical transmission of bits across a medium. Each higher layer in the model utilizes the underlying layers in a somewhat independent fashion (meaning little or no overlap in functions between the layers). This layered approach is implemented through the use of encapsulation. This concept can best be explained using the example shown in Figure 4.14. This example shows how each layer associated with a web browser session maps to the protocol stack model. Starting at the application layer, the web browser would

generate an HTTP request using an application-specific command. This request would then be passed down to the TCP layer, which would construct a TCP packet consisting of a TCP header and TCP data. The TCP header contains information particular to the TCP protocol, such as packet sequencing information, checksum information and the source and destination port number (HTTP typically has a port number of 80).

At the IP protocol level, an IP datagram is constructed to hold the TCP packet. Similar to the TCP packet, the IP datagram consists of an IP header and IP data. The IP header contains information such as the type of service, checksum information, protocol type (06h for TCP), and the source and destination IP addresses. The data field of the IP datagram contains the complete TCP packet to be transmitted. At the data link/physical layer, the IP datagram is transported across the network using the IEEE 802.3 protocol. A MAC (IEEE 802.3) frame consists of a MAC header and a MAC payload (data). The MAC header contains information about the MAC frame, such as the source MAC address, the destination MAC address and the length of the frame. The payload field contains the complete IP datagram to be transported.

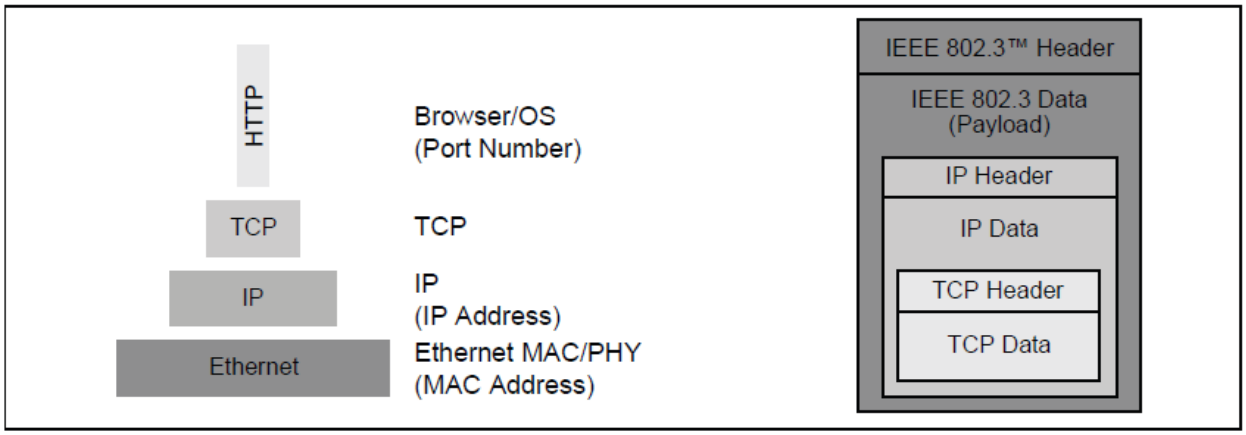

Figure 4.14 Data Encapsulation Example [32]

Note that the various addresses encapsulated within each protocol are different, and typically, have no fixed relationship to one another. In this example, the TCP packet uses a port number, which is typically assigned based on the application layer protocol (i.e., port 80 for HTTP). The IP datagram uses an IP address, which is statically or dynamically assigned out of a

pool of available internet addresses, and the MAC frame uses MAC addresses, which are assigned to the particular piece of hardware.

4.7 Web Pages

A Web server can serve static or dynamic (generated by a program upon invocation) content or web page [47]. This section discusses some commonly used technologies used to provide content and to facilitate interaction between a Web server and an application server that is not typically directly accessible to a client (for example, a Web browser).

4.7.1 Static content web page

Static content, or static pages, usually consist of data associated with a URL that does not change very often and does not depend on any client input. This content is usually in the form of some markup language, such as HTML or XML.

4.7.2 Dynamic content web page

Dynamic pages enable the Web to be used as a medium for database access, commercial transactions, gateways to other information systems, and even chat forums. These are the documents generated on-the-fly, typically by running a script through the Common Gateway Interface (CGI) protocol. It is a standard for interfacing external programs with information servers on the Internet. A CGI script can be written in any language that allows it to be executed.

Dynamic variables also plays a major role in building the dynamic web pages as they allow the web server module to take data from our system, such as the value from a sensor or data in memory, and combine it into a template web page. The completed web page is then transmitted through a network or the Internet, and the system data is displayed on the user's screen.

In this chapter, the different layers of the TCP/IP stack are explained with the protocols involved. The concept of IP address, IP datagram, TCP connection and its datagram are elaborated. Also, the application layer protocol, HTTP, is deeply studied. Its message structure, form methods are discussed. Finally, the static and dynamic web pages are briefly covered.

CHAPTER 5

IMPLEMENTATION OF HARDWARE AND SOFTWARE

5.1 Implementation of Hardware

5.1.1 Ethernet Hardware Schematic

The Ethernet schematic is designed using PIC family of microcontrollers and Ethernet switch IC. This schematic is divided into 3 sections: controller section, Ethernet switch section and power supply section. The whole module is designed with the help of OrCad software [48].Figure 5.1, 5.2 and 5.3shows all these three sections.

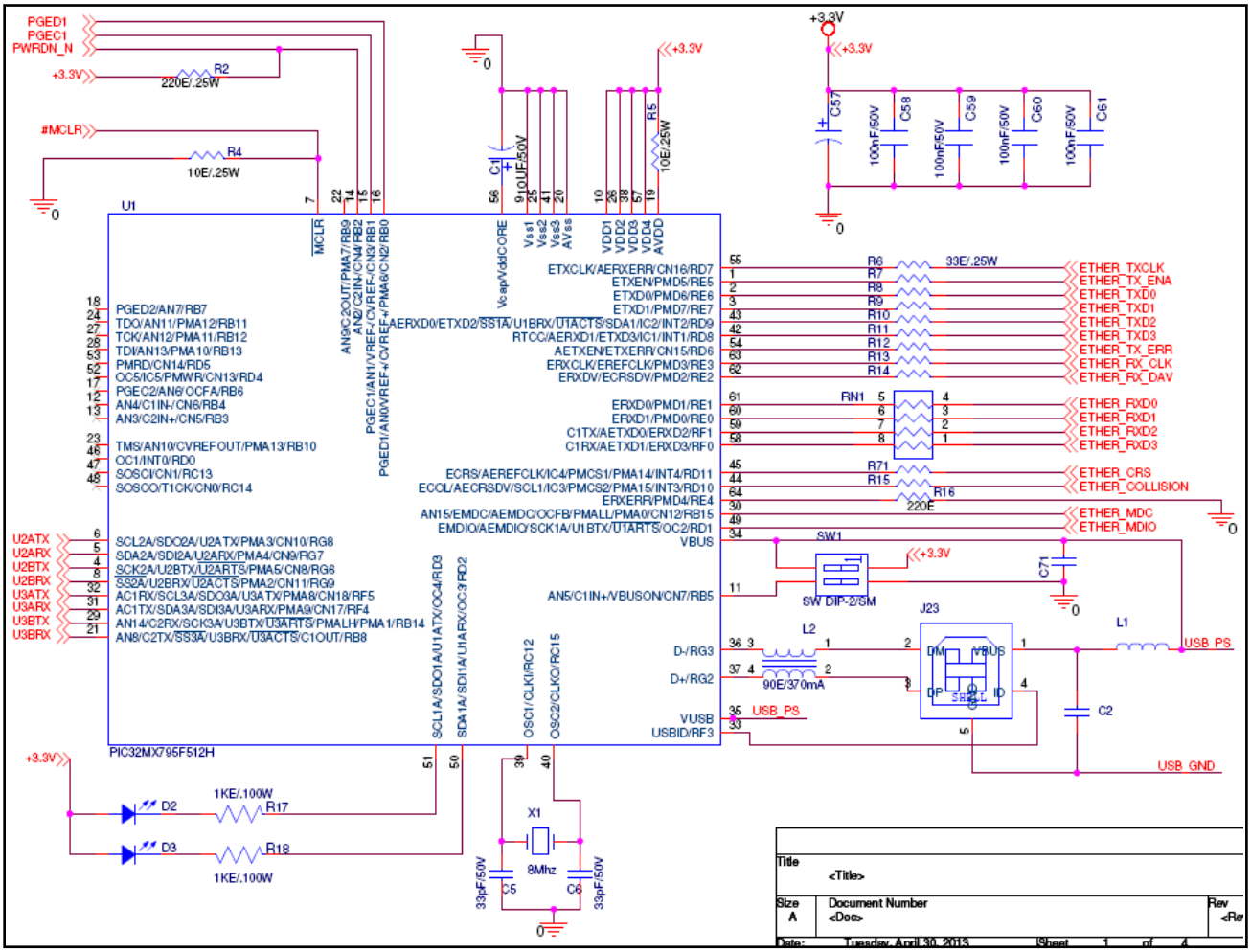

Figure 5.1 PIC32MX795F512H Microcontroller

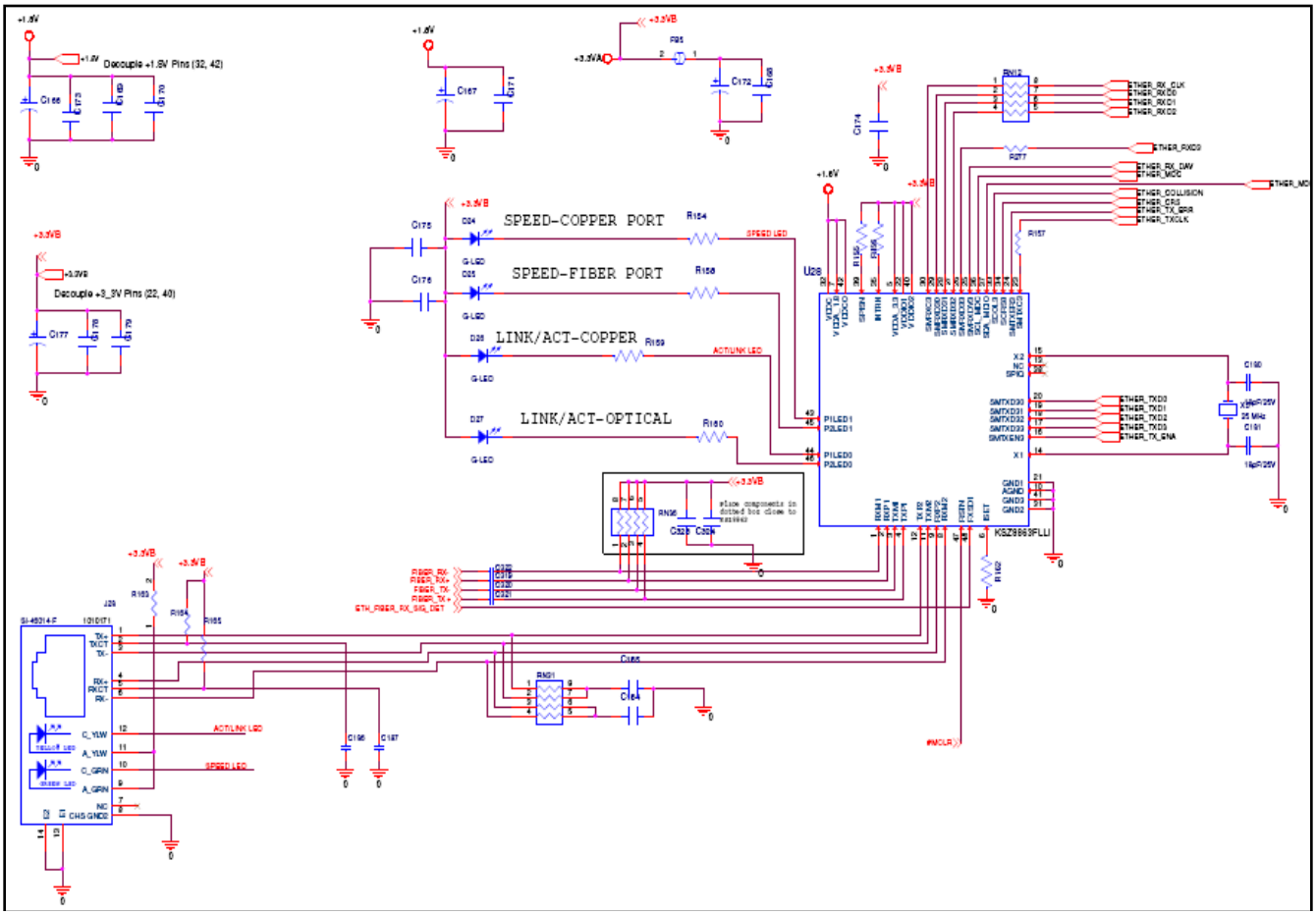

Figure 5.2 KSZ8863FLL/RLL/MLL Ethernet Switch

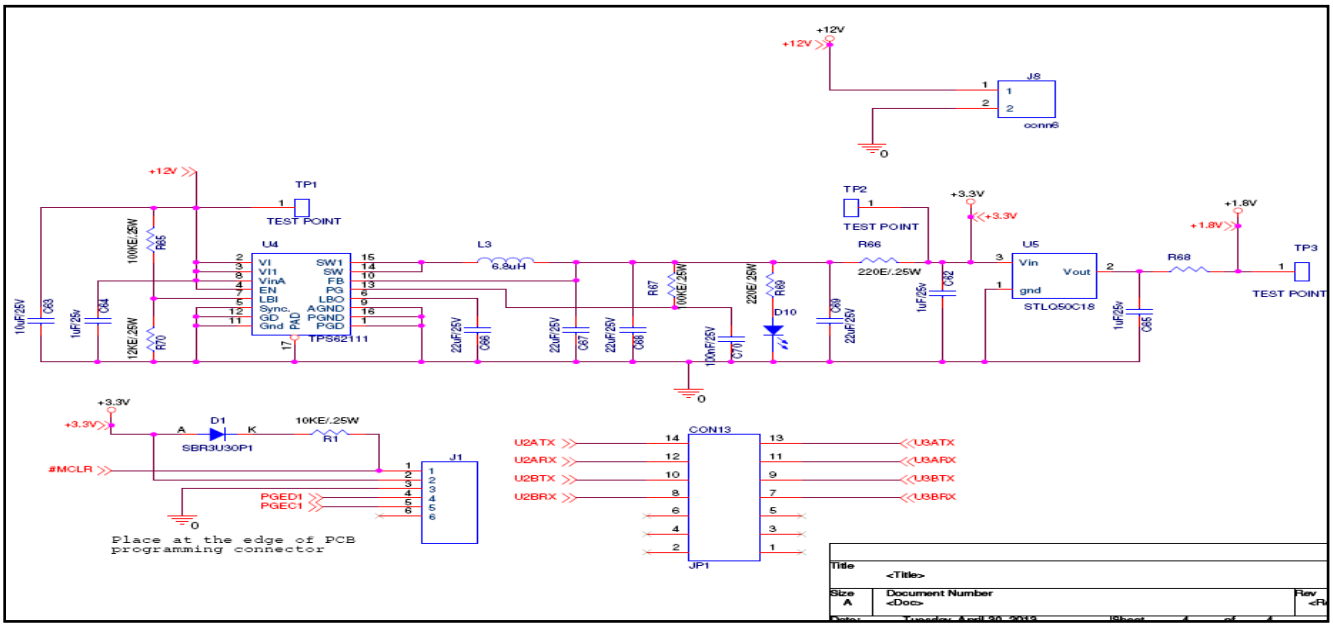

Figure 5.3 Power Supply Section

The first section is the controller section which is shown in figure 5.1. The controller used in the Ethernet module is PIC32MX795F512H. It has been explained in section 3.9.1.This controller is chosen because it has inbuilt Ethernet module which is the desirable.

Second section represents Ethernet switch which consists of 3 ports. It is shown in figure 5.2. One port is used as copper port, second as fiber port and third for MII interface. This Ethernet switch is explained in detail in section 3.9.2.

Third section is power supply section which is shown in figure 5.3. The main supply to Ethernet module is +12V. This +12V is then converted into +3.3V with the help of a synchronous step-down converter. This +3.3V is again converted into +1.8V using a low drop voltage regulator. Hence, three supplies are being used +12V, +3.3V and +1.8V. In the above schematic, J1 is a 6-pin programming connector and CON12 is a 14-pin connector on which all UART pins of controller are connected, to be used further in any application.

5.1.2 Understanding of circuit with the help of block diagram

The above implementation of hardware circuit can be understood with the help of block diagram shown in figure 5.4.

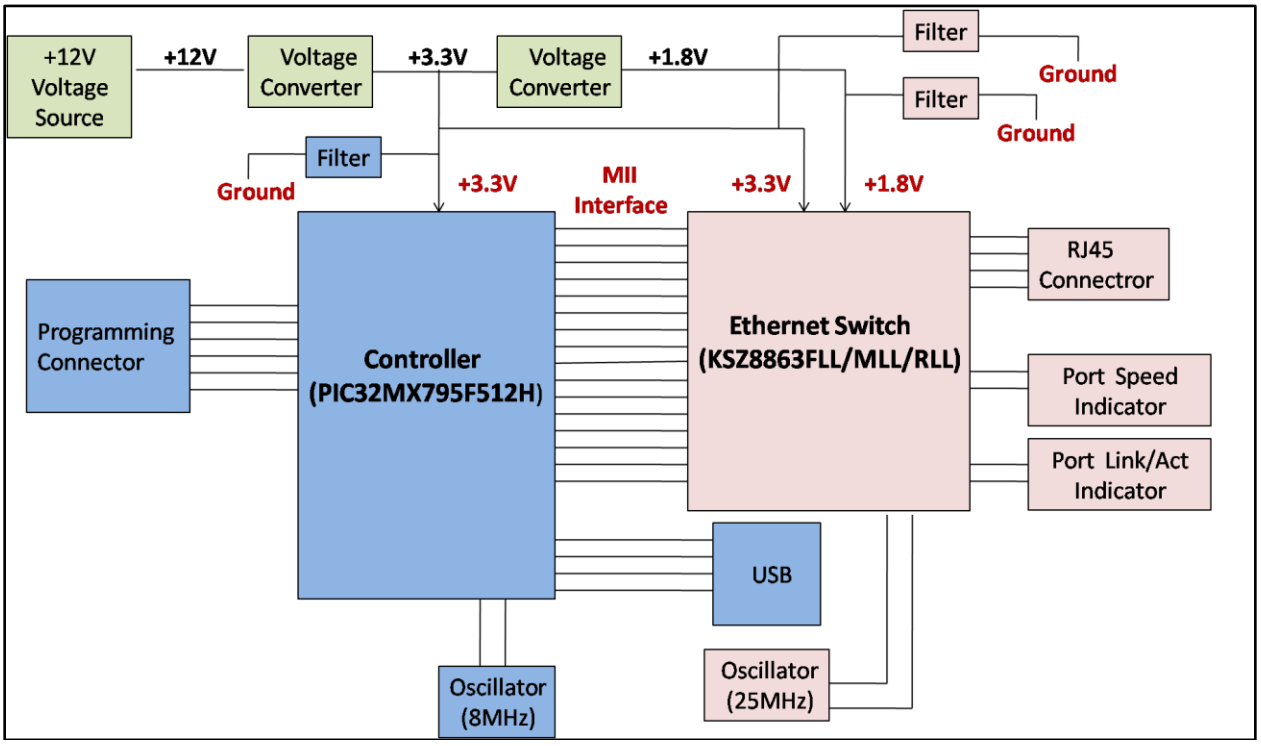

Figure 5.4 Block diagram of the implemented design

In this diagram, all the basic connections can be seen showing the interfacing of controller with the Ethernet switch and with USB, programming connector, oscillator etc.

5.2 Implementation of Software

The flow of an Ethernet frame through various protocol layers is represented in figure 5.5. As a message goes up the left side of the flow chart, two checksums are verified before reaching the TCP state machine. Web page requests generate an HTML page or image, and then two checksums are added going down the right side. ARP message flow is shown in the central portion of the chart.

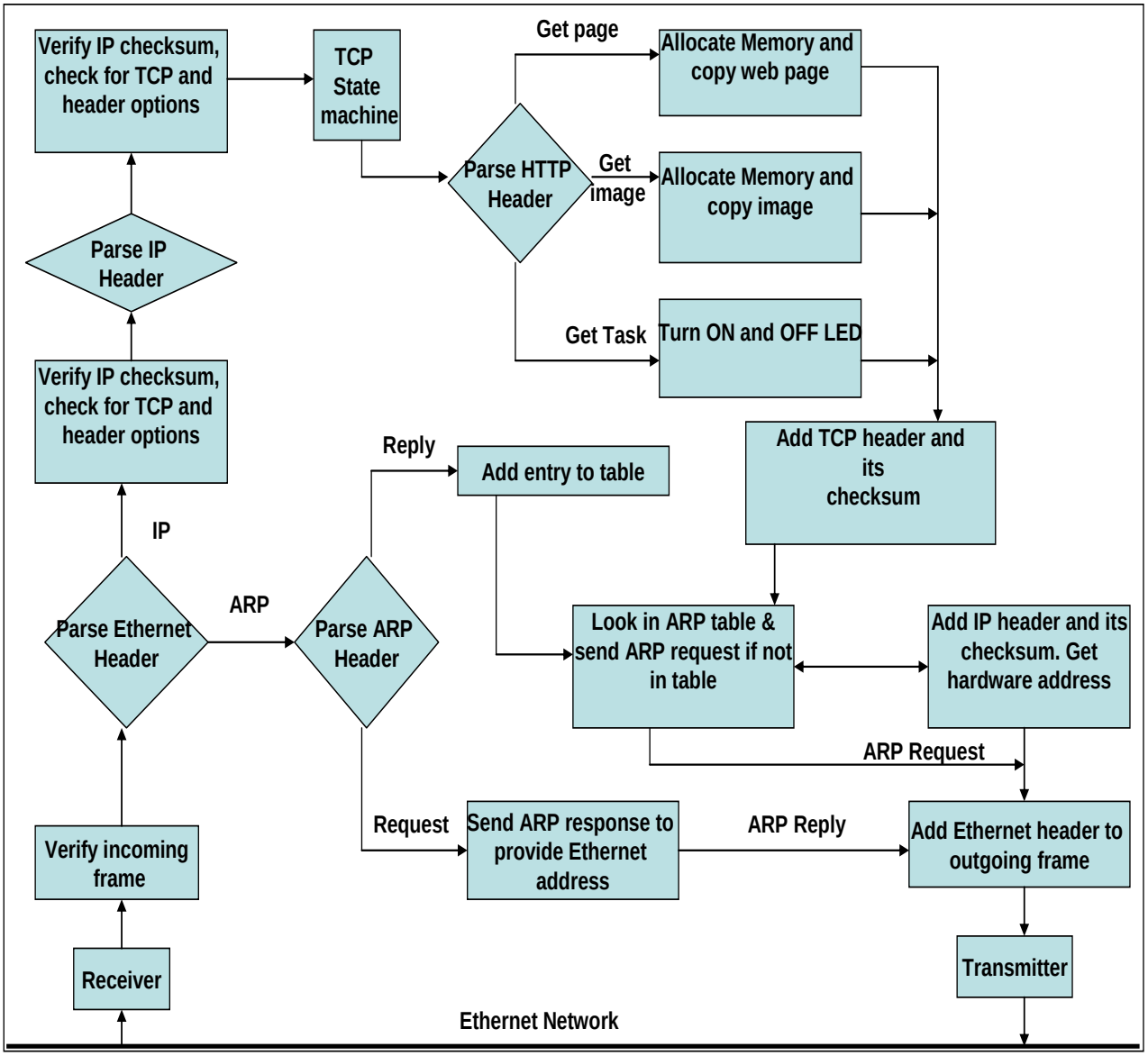

Figure 5.5 Flow of Ethernet frame through various protocols

When a new packet arrives to the system, an Ethernet transceiver chip converts the analog signals of Ethernet to 8-bit width digital signals. The design reads the data and verifies its validity. When the signal RX_DV (data valid) goes high, then the state machine goes to read preamble state and waits until the end of the preamble. Next, state machine examines the local MAC address. If the address's first byte is FF, this means it is a group address and so the ARP module is activated. Thus, at the same time, decoding of protocol's header is done to check which upper layer protocol is requested to perform the desired function. Here, the requested upper layer protocol is IP. Hence, after parsing Ethernet header, it forwards IP datagram further and attach to it a calculated checksum. Now, while receiving IP datagram, again checksum is calculated which is compared with the previous checksum and also next requested protocol i.e. TCP is inquired. Then, the same process is repeated. IP header is parsed and TCP datagram is moved forward. Their also checksum is calculated and compared, and then the next upper layer protocol is inquired which is HTTP in the present application. After getting HTTP datagram, its header is parsed and the method used for the desired function is executed like Get page, Get image and Get task.

Now, reverse process is performed. Instead of parsing, addition of lower layer protocol's header takes place. TCP header is added, and its checksum is calculated and compared. Then, IP header is added. After this, ARP table is looked upon for hardware or MAC address where data has to be sent. If it is not found in ARP table, an ARP request is generated and ARP table is immediately updated. Finally, Ethernet header is added and is moved to its destination (hardware). It is to be noted that for both the incoming as well as outgoing frames, two checksums are computed: one for the IP header and the other for the TCP segment.

5.3 IP (Internet Protocol) Flow Diagram

The IP layer of the TCP/IP Stack is implemented by the file "*IP.c*". The header file "*IP.h*" defines the services provided by the layer. In this architecture, the IP layer is passive; it does not respond to IP data packets. Instead, higher level layers use IP primitives and fetch the IP packet, interpret it and take appropriate action. The flow of IP is shown in figure 5.6and it is explained as:

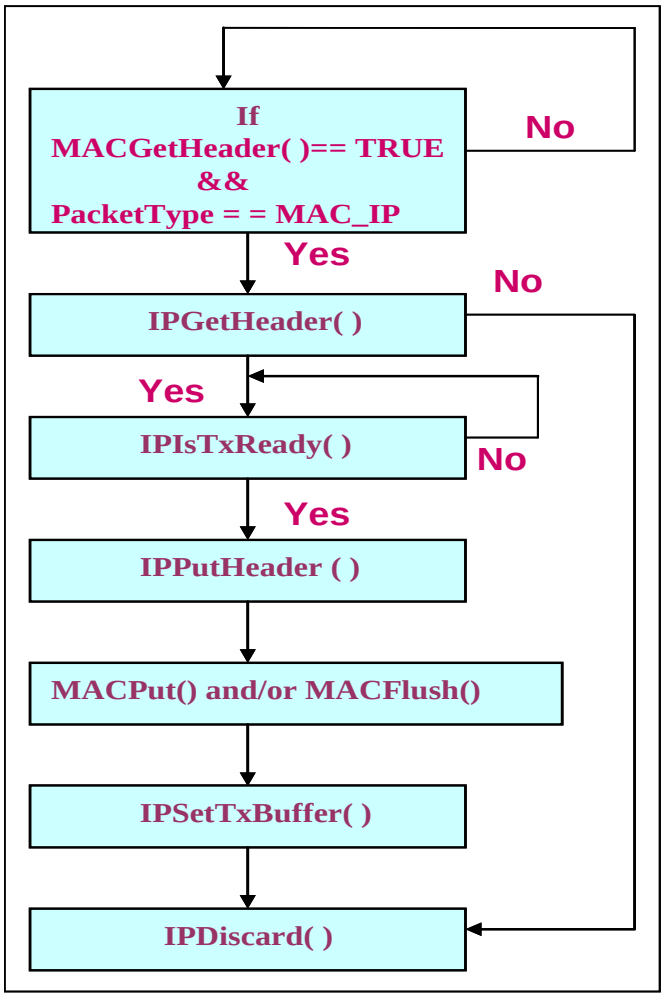

Figure 5.6 IP Flow chart

i. MACGetHeader ( )

This function checks the MAC receive buffer; if any packet is found, it returns the remote host and data packet information. Once a data packet is fetched by calling MACGetHeader, the complete data packet must be fetched (using MACGet) and discarded (using MACDiscardRx). MACGetHeader cannot be called multiple times to receive multiple packets and fetch data later on.

Syntax: *BOOL MACGetHeader (MAC_ADDR *remote, BYTE *type)*

Parameters:

Remote [out]: Remote MAC address

type [out]: Data packet type (Table 5.1)

Table 5.1: Possible values for *type [out]* parameter [49]

Value	Meaning
MAC_IP	An IP data packet is received
MAC_ARP	An ARP data packet is received
MAC_UNKNOWN	An unknown or unsupported data packet is received

Return Values:

TRUE: If a data packet is received and found to be valid. All parameters are populated.

FALSE: If no data packet is received or found to be invalid.

Pre-Condition: None

ii. IPGetHeader ( )

This function fetches the IP header from the active transmit buffer and validates it. It assumes that the active receive buffer access pointer is positioned to the beginning of the MAC Data area. In order to satisfy this condition, the higher level layer must perform the following checks before calling this function: *If MACGetHeader == TRUE* and *PacketType == MAC_IP, call IPGetHeader*. Else IPGetHeader is not called. Once the IP packet is processed and no longer needed, the caller must discard it from the MAC buffer by calling the IPDiscard function.

Syntax: *BOOL IPGetHeader (IP_ADDR *localIP, NODE_INFO *remote, BYTE *protocol, WORD *len)*

Parameters:

localIP[out]:Local node information such as MAC and IP addresses

remote[out]: Remote node information such as MAC and IP addresses

protocol[out]:Protocol associated with this IP packet (Table 5.2)

Table 5.2: Possible values for protocol [out] parameter [49]

Value	Meaning
IP_PROT_ICMP	This is an ICMP packet
IP_PROT_TCP	This is a TCP packet
IP_PROT_UDP	This is a UDP packet
All others	Unknown protocol

len[out]: It is the total length of IP data in this packet.

Return Values

TRUE: A valid IP packet was received. Remote IP address, packet protocol and packet length parameters are populated.

FALSE: An invalid IP packet was received. Parameters are not populated.

iii. IPIsTxReady ( )

This is a macro that calls MACIsTxReady in turn. This macro is provided to create an abstraction only. Rather than calling MACIsTxReady directly, the upper layer that uses IP services should call this macro.

Syntax: *BOOL IPIsTxReady ( )*

Parameters: None

Return Values:

TRUE: If there is at least one transmit buffer empty.

FALSE: If there is no empty transmit buffer.

iv. IPPutHeader

This function assembles a valid IP header and loads it into active transmit buffer. It assembles an IP packet complete with header checksum and correct network byte order. After this function, the active transmit buffer access pointer points to the beginning of the IP data area. This function does not initiate IP packet transmission. The caller must either load IP data by calling the MACPut function and/or calling MACFlush to mark the buffer as ready to transmit.

Syntax: *WORD IPPutHeader (NODE_INFO *remote, BYTE protocol, WORD len)*

Parameters:

remote[in]:Remote node information such as MAC and IP addresses.

protocol[in]:Protocol to use for this data packet (Table 5.3)

Table 5.3: Possible values for protocol [in] parameter [49]

Value	Meaning
IP_PROT_ICMP	Assemble this packet as ICMP
IP_PROT_TCP	Assemble this packet as TCP segment
IP_PROT_UDP	Assemble this packet as UDP segment

len[in]:Total length of IP data bytes, excluding IP header

Return Values: None

Pre-Condition: IPIsTxReady == TRUE

v. MACPut ( )

This function loads the given data byte into an active transmit or receive buffer. It can be used to modify either a transmit buffer or receive buffer – whichever is currently active.

Syntax: *void MACPut(BYTE val)*

Parameters:

val[in]:Data byte to be written.

Return Values: None

Pre-Condition: MACGetHeader, MACPutHeader, MACSetRxBuffer or MACSetTxBuffer must have been called.

vi. IPSetTxBuffer

This is a macro that calls MACIsTxReady () in turn. It provides next Read/Write access to transmit buffer 'a' set to offset 'b'.

Syntax: *IPSetTxBuffer (a, b)*

Parameters:

a: Buffer identifier

b: Offset

PreCondition: None.

Return Values: None.

vii. IPDiscard ( )

This is a macro which calls MACDiscard ( ) in turn. Here, current packet is discarded and buffer is freed-up.

- **MACDiscardTx ( )**

  This function discards given transmit buffer content and marks it as free.

Syntax: *void MACDiscardTx(BUFFER buffer)*

Parameters:

buffer[in]:Buffer to be discarded

Return Values: None

- **MACDiscardRx ( )**

  This function discards the active receive buffer data and marks that buffer as free. It marks the last received packet (obtained using *MACGetHeader ( )*)as being processed and frees the buffer memory associated with it.

Syntax: *void MACDiscardRx ( )*

Parameters: None

Return Values: None

Pre-Condition: MACGetHeader( ) = = TRUE.

5.4 HTTP (Hyper Text Terminal Protocol)

- **Web Forms**

  Web forms allow the web server module to accept data from users through a network as pictured in figure 5.7. This data can then be used to control system memory or outputs. In our project, the microcontroller accepts the command of "*On/Off*" from the web page and responds by turning the LED on.

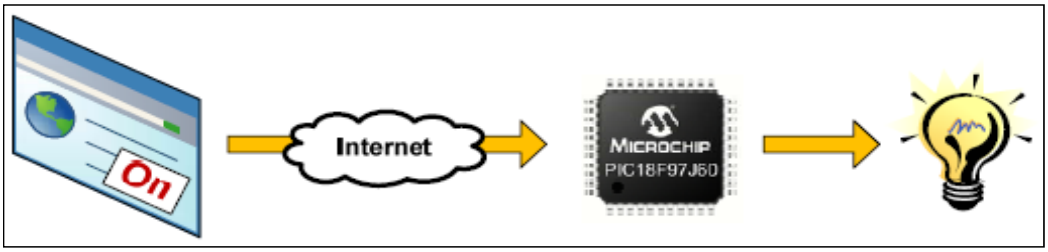

Figure 5.7 Application of Web Forms[50]

Web forms, like the rest of any web page, are designed in HTML. They are recognizable as the sections contained within <form> tags. Within the *<form>* tags, there is a series of *<input>* tags. Each input tag represents some user-interface element (such as a text field, checkbox, or a button). It is assigned a "name" parameter, which controls the name of the field when it gets returned to the browser.

When variables are submitted to the server, they are encoded as a series of name/value pairs. Each field is separated by an ampersand symbol (*&*) and names are separated from values with an equal sign (=). Non-alphanumeric characters, if present, are escaped by a percent sign (*%*) and then followed by two characters representing the ASCII value in hexadecimal.

- **Form Methods**

Each form tag has a parameter called the "method". This parameter is set to either "*Get*" or "*post*", which controls how the data is submitted to the server.

<form method="get" ...>

The GET method appends all submitted data to the URL.

http://rtuboard/forms.htm?led4=1&led3=1&led2=0&led1=0

The data string follows a question mark in the URL requested by the client. In this, the board LEDs are set to values '1' or '0' i.e. whether "*On*" or "*Off*".

In HTTP web server module, data submitted in this fashion is easiest to process. All data submitted via GET is automatically decoded and placed in the curHTTP.data byte array all at once. Since it is in memory, searching for parameters is simple and efficient. The length of all names, values, and delimiting characters is limited to the size of the available buffer – generally 100 bytes. The drawback of course, is that memory is limited, so data submitted in this manner must fit completely in this space allocated to each HTTP data buffer.

The HTTPExecuteGet ( ) callback function is called when data is submitted using this method. Like the dynamic variable callbacks, this function is also user implemented and is defined in CustomHTTPApp.c.

Steps to be executed during this process:

i. Handled in *HTTPExecuteGet ( )*

ii. Data stored in *curHTTP.data*

iii. Locate values with:

- *HTTPGetArg ( )*
- *HTTPGetROMArg ( )*

iv. Process input values

v. Perform necessary actions

5.4.1 HTTPExecuteGet Function

- The purpose of this function is to parse the data received from URL parameters (GET method forms) and perform any application-specific tasks in response to these inputs.
- When this function is called, curHTTP.data(Current HTTP Connection State), which is a structure, contains sequential name/value pairs of strings representing the data received.
- In this format, HTTPGetArg and HTTPGetROMArg can be used to search for specific variables in the input. If data buffer space associated with this connection is required, curHTTP.data may be overwritten here once the application is done with the values.
- This function is only called if variables are received via URL parameters. This function may not write to the TCP buffer.
- This function may service multiple HTTP requests simultaneously.

5.4.2 HTTPGetArg

This function Searches through a data array to find the value associated with a given argument. It can be used to find form field values in data received over GET. The end of data is assumed to be reached when a null name parameter is encountered. This requires the string to have an even number of null-terminated strings, followed by an additional null terminator.

Syntax: *BYTE* HTTPGetArg(BYTE* cData, BYTE* cArg);*

Parameters:

data: It represents the buffer to search.

arg: It represents the name of the argument to find.

Preconditions: The data array has a valid series of null terminated name/value pairs.

5.4.3 HTTPGetROMArg

This function searches through a data array to find the value associated with a given argument. It can be used to find form field values in data received over GET. The end of data is assumed to be reached when a null name parameter is encountered. This requires the string to have an even number of null-terminated strings, followed by an additional null terminator.

Syntax: *BYTE* HTTPGetROMArg(BYTE* cData, ROM BYTE* cArg);*

Parameters:

data: It represents the buffer to search.

arg: It represents the name of the argument to find.

Returns: A pointer to the argument value, or NULL if not found.

Preconditions: The data array has a valid series of null terminated name/value pairs.

5.5 Dynamic Web Page

The web page which is interacting with the hardware is shown below in figure 5.8.

Figure 5.8 Web Page for Remote Terminal Unit

In this page, a set of radio buttons are intended to let us control whether the board's lights are on or off. On our development board, we'd like this setting to control two of the LEDs: LED1 and LED2.

Coding of this web page is written in JavaScript and HTML language. Java Script is out of the scope of this project but HTML code is given as:

```
</head>
  <body bgcolor="yellow" ">
  <table border="0" width="100%">
  <tr>
  <td>
  <imgsrc="thapar.gif/CG.gif" />
  </td>
  <td width="100%">
  <p align="right">
  <font size="6" face="MS Sans Serif"><b>Remote Terminal Unit </b></font>
  </p>
  </td>
  </tr>
  </table>
  <table border="0" cellpadding="0" cellspacing="0" width="100%" fgcolor="yellow">
  <trbgcolor="#FFFFFF" fgcolor="yellow">
  <td width="25%">
  <font color="#000000"></font>
  </td>
  </tr>
  <form>
  <tr>
  <td>
  <b>Monitoring and Control</b>
  </td>
```

```
</tr>
<tr>
<td>
                    Toggle LED1:</td>
<td>
<input type="radio" value="on" ~LED_chk(1)~onclick="GetServerFile('0?1=LED2',')" />
on
<input type="radio" value="off" ~LED_chk(0)~onclick="GetServerFile('0?1=LED2',')" />
off
</td>
<td>
  Toggle LED2:</td>
<td>
<input type="radio" value="on" ~LED_chk(1)~onclick="GetServerFile('0?0=LED1',')" />
on
<input type="radio" value="off" ~LED_chk(0)~onclick="GetServerFile('0?0=LED1',')" />
off
</td>
</tr>
</body>
</html>
```

It can be noticed that a dynamic variable, highlighted in red, placed inside each tag. These dynamic variables control which of the two settings are selected by default when the form loads. Dynamic variables are created by enclosing any name inside a pair of tilde (~) characters and placing that variable in your web pages' HTML code. When the HTTP web server module encounters this variable, it will execute a callback function. This callback is implemented in our application's C code, and controls what is transmitted to the browser and ultimately display it on screen.

As can be seen in the coding, on clicking the LED button, a function is called *"GetServerFile* ( )" which is defined in JavaScript, which will help in accessing data from the server.

This chapter gives an overview of the hardware and software implemented. All the three sections: power supply section, microcontroller section and Ethernet switch section are explained with the help of their circuit diagrams. In this, implementation of software is explained thoroughly with the help of flow chart showing the interaction of various layers of the TCP/IP stack. Also, the working of each protocol, used in the communication, is discussed with the help of their individual flow charts. The C functions called during this whole process of communication are defined with their every parameter explained.

CHAPTER 6

METHODOLOGY

6.1 Test 1: Testing of Auxiliary Inputs of Analog Section

There are 3 auxiliary input channels. Figure 6.1 shows whole processing of these auxiliary input channels. This test is performed to know how accurately the status of field sensors is measured and detected by the controller using ADC.

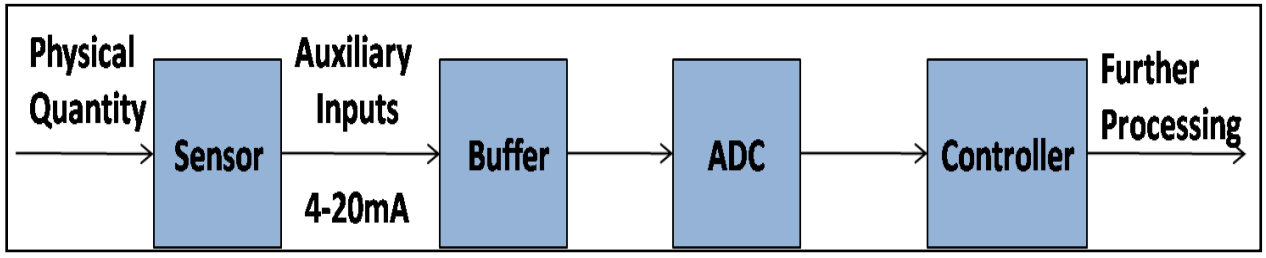

Figure 6.1 Processing of auxiliary inputs

Here, the physical quantity can be temperature, pressure etc. These physical quantities are sensed by field sensors which gives their output in the range of 4 to 20mA. These signals, after converting them to voltage signals, are given to a buffer to get a noise free signal. These signals are then applied as inputs to ADC which sample and digitize these signals as the controller accept only digital signals. The procedure for performing the tests for each channel is given in further sections.

6.1.1 Channel 1 Measurements

The steps for the testing of input channel 1 are concluded as:

Step1: A variable resistor is used, which works on +24V and has various resistances to provide current in the range of 4 to 20mA. The input channel 1of AI card is tested by connecting it to the variable resistor and set its resistance accordingly to provide 4mA.

Step2: The 4mA current is passed to the resistor network in the AI card which converts it into voltage (mV).

Step3: To make the signal noise free, this voltage is then passed through a buffer.

Step4: The output of the buffer is an analog signal. This analog signal is given as an input to the ADC.

Step5: The ADC count corresponding to 4mA is calculated according to the formula given in the datasheet as:

$$ADC\ Count = (2^N \times A_{IN} \times GAIN)/V_{REF}$$

Step6: After applying voltage to the ADC, the analog signal is converted into the digital one by taking samples and then, ADC calculates the voltage from these samples. The applied formula is:

$$Calculated\ Voltage = (ADC\ count \times 2.5) / (2^{24} \times 2)$$

Step7: Applied input is 4mA and voltage is also calculated with the help of samples, so now resistance can be calculated with the help of ohm's law.

Step8: Now, according to this calculated resistance, current is calculated. It should be equal to the input current which tells the accuracy of the AI card.

Step9: In the end, error percentage between input current and calculated current is determined by using the following formula:

$$Error\ \% = ((Calculated\ current - Input\ current) / Calculated\ current) \times 100$$

Step10: All the above steps are followed for 8mA, 12mA, 16mA and 20mA for the same input channel.

The same steps are followed for the remaining two channels: channel 2 and channel 3.

6.2 Test2: Testing of Current Inputs with the help of Current Transformer

This testing is done to know at which value of input current, CT gives accurate output voltage (~ 200mV) and at which input, the output voltage of CT starts saturating. Below 200mV,

the controller can't detect the input. Thus it is the threshold value. There are 3 current input channels in AI section. Figure 6.2 explains the whole processing of current inputs.

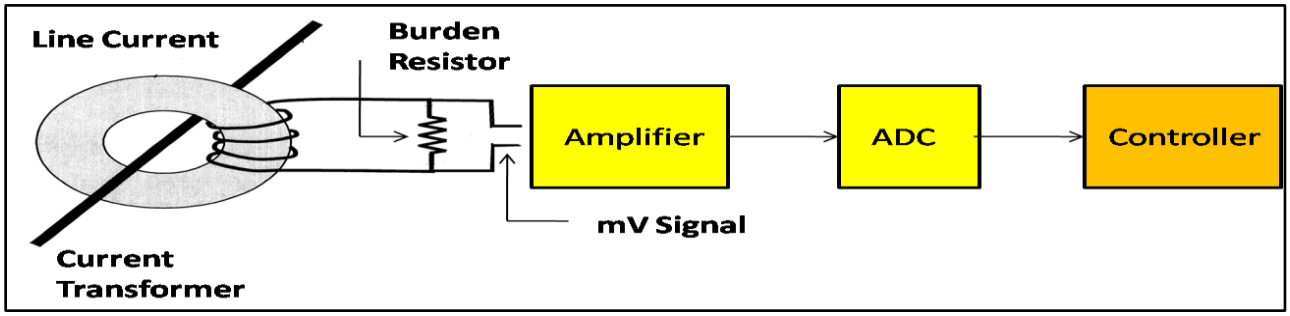

Figure 6.2 Block Diagram representing processing of current inputs

This figure shows that the 1A/5A current coming from the high rating current transformers, installed in series with the transmission line, are converted into some lower valued signal. The value of this signal depends upon the burden resistor. Now, the obtained mvolt signal is passed through a unity gain amplifier or a buffer to get a noise free signal. This analog signal is then digitized using ADC and the output of that ADC is sent to the controller. Figure 6.3 shows the circuit diagram for the current transformer.

In this testing, current of 1 A and 5 A is given to the CT inputs with the help of Omicron. Omicron is a device which provides current, voltage and frequency at different ranges.

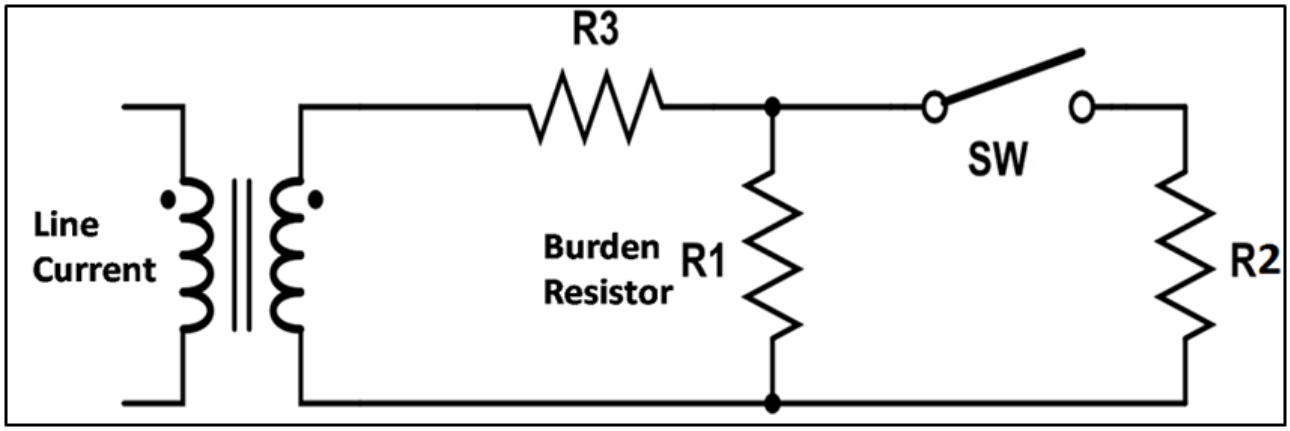

Figure 6.2 Circuit Diagram for Current Transformer

As shown in this figure, the output of current transformer depends upon burden resistor (R1). If input is 1A, then only R1 has to be set. If we want to measure 5A current, then the switch (SW) should be closed and resistor R2 has to be set. These resistors are set to get the required output of 200mV. After certain experimentation, the value of R1 comes out to be 150Ω and value of R2 is 36Ω.

6.2.1 Measurement of 1A current

The steps for this testing are concluded as follows:

Step1: At the input of CT, the current starting from 0.2A is applied with the help of Omicron.

Step2: The output voltage, across a burden resistor, corresponding to the applied input current is measured.

Step3: The expected voltage across the burden resistor is also calculated for each value of input current.

Step4: Here, these calculations are continued up to the input current of 4.5A.

Step5: As can be seen from the table, the accurate output voltage of CT is obtained at 1A input current and the output waveform of CT starts saturating at the input current of 3A when seen through the oscilloscope.

6.2.2 Measurement of 5A current

This testing follows the same steps as those for 1A input current. The range of input current taken is from 5A to 10A. The only difference, in this case, is that the output is calculated across the parallel combination of R1 and R2 rather than only R1.

6.3 Test 3: Testing of Current Inputs using Potential Transformer (PT)

This testing is done to check at which particular input voltage, output voltage at the secondary side is accurate and from which input voltage, output voltage of PT starts to saturate. There are 3 voltage channels in this project. Figure 6.4 shows the processing of voltage inputs.

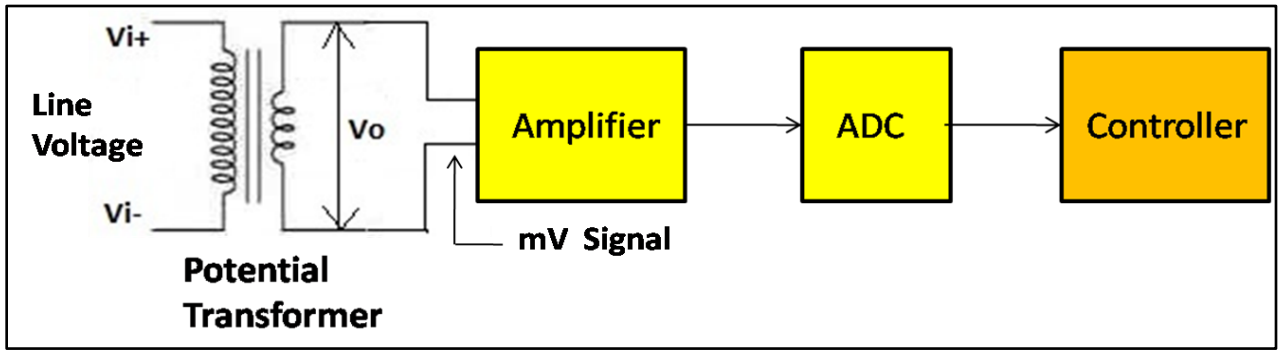

Figure 6.3 Block Diagram representing processing of voltage inputs

In this figure, the PT converts 110V, coming from the high rating potential transformer installed in parallel with the transmission lines, into a low-valued signal of some mV range. This signal is then given to a unity gain amplifier or a buffer which results in a noise free signal. The ADC then digitizes the incoming analog signal and sends it to the controller. The steps regarding this test can be concluded as follows:

Step1: At the primary side of PT, a number of voltages are applied starting from 0V until that input voltage at which the output waveform starts saturating.

Step2: As can be seen from the above table, after 80V input current, the output voltage starts saturating and at 63.6V input, accurate result is obtained.

6.4 Testing of DC1 and DC2

DC measurement is very important as it has to be continuously measured that what dc is being received from the dc source and what dc value is required for the particular operation. Accordingly to this measurement and monitoring, dc value, coming from the source, can be changed. That's why, this testing is done. Here, there are two DC input channels: DC1 and DC2. These two input channels are processed through the procedure given in figure 6.5 below.

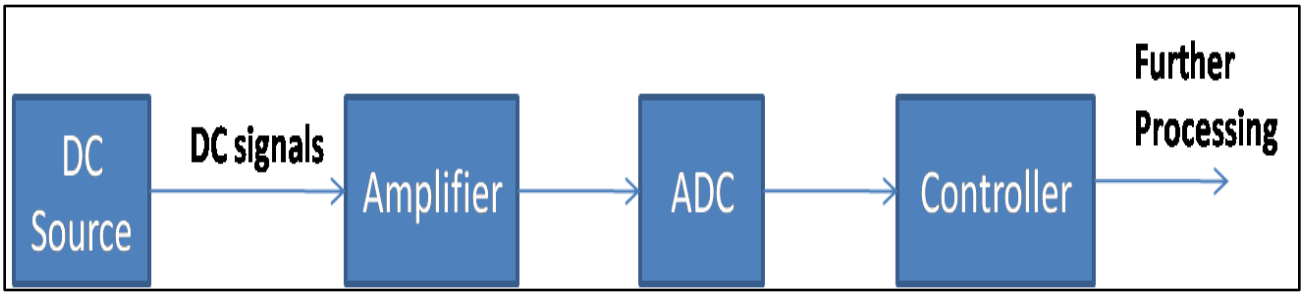

Figure 6.4 Block Diagram representing processing of DC inputs

This figure shows that a DC source is continuously providing DC signal to the AI section. These signals are then passed through buffer and digitized with the help of ADC. These digitized signals are then sent to the controller. These signals are continuously measured and monitored.

6.4.1 DC1 Measurements

This testing is done to check whether the DC signal supplied by the DC source is measured correctly by the controller or not. The range of input DC signal taken is 70V to 280V.The corresponding steps for performing this test are as follows:

Step1: A number of DC input voltages are applied at the input of AI section from DC source.

Step2: The ADC count, corresponding to the applied input, is calculated by using the following formula:

$$ADC\ Count = (2^N \times A_{IN} \times GAIN)/V_{REF}$$

Step3: At the output of the ADC, the DC voltage is observed which should be equal or nearly equal to the applied input voltage.

Step4: The %age of error is calculated between applied input DC voltage and the observed DC output voltage for each input case.

6.4.2 DC2 Measurements

This DC2 measurement follows the same steps as those for DC1 measurement.

6.5 Analyzing Network Protocols

To know how the data packets move in a network, a Network Protocol Analyzer WIRESHARK is used. It is an open-source protocol analyzer designed by Gerald Combs that runs on Windows and UNIX platforms. For start capturing, select the interface from the menu *Capture >> Interfaces.*

A brief description of the most interesting areas that Wireshark displays, once data capture starts, is shown in figure 6.6.

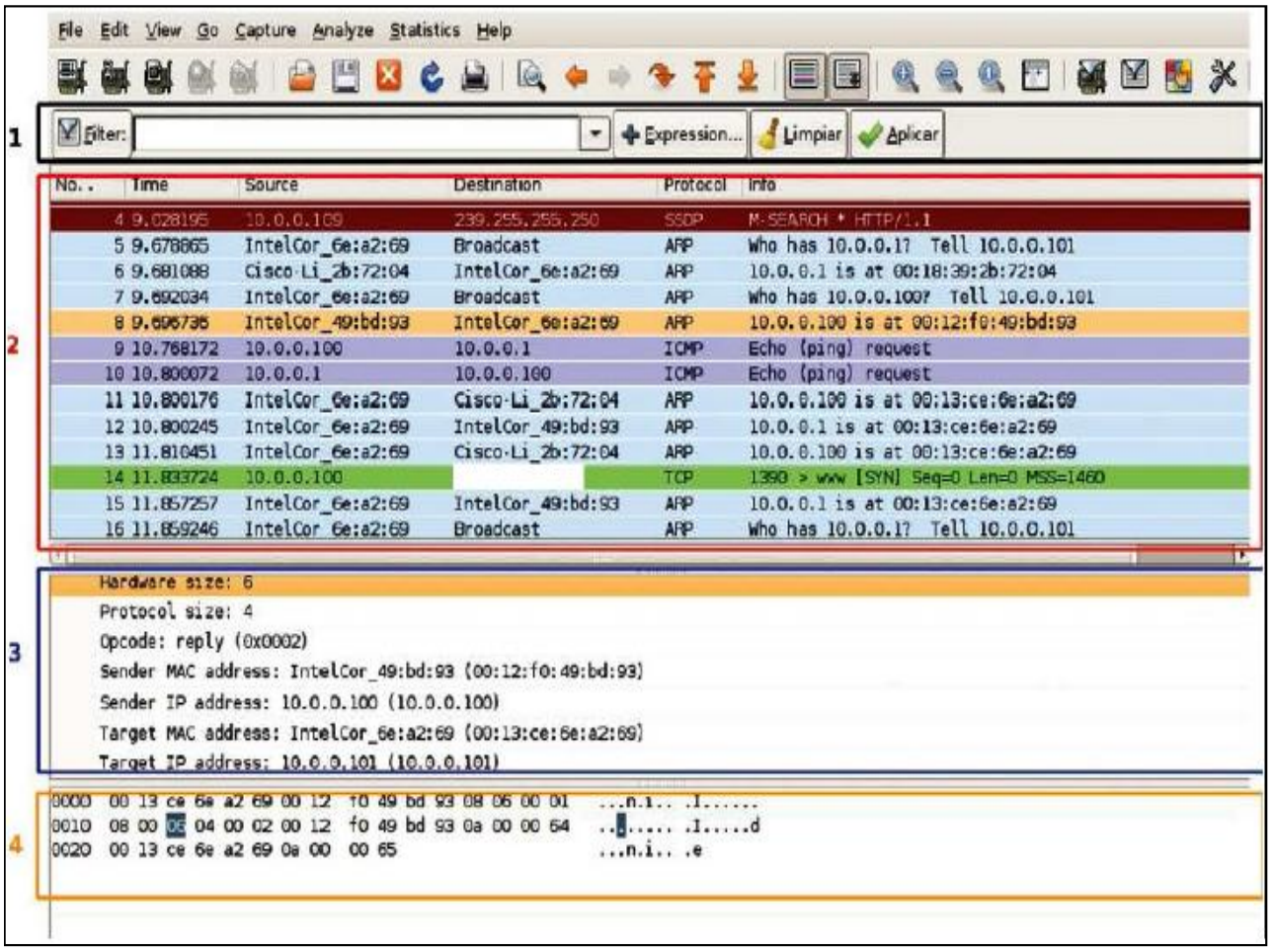

Figure 6.5 Wire shark Areas [51]

This figure shows the areas displayed by the Wireshark. These areas are divided into 4 zones as explained below:

- **Zone 1** is the area where filters are defined and enables us to define search patterns to view those packets or protocols that are of interest to us.
- **Zone 2** corresponds to a list to view of all packets being captured in real time. Knowing how to interpret the data given in this zone correctly (protocol type, number sequence, flags, time stamps, ports, etc.) enables us to, under certain circumstances; identify the problem without having to perform a detailed audit.
- **Zone 3** enables you to classify, by layer, each header of the packets selected in zone 2 and we can navigate through each field of the same.
- Lastly, **Zone 4** represents, in hexadecimal format, the packet in the state in which it was captured by your network card.

CHAPTER 7

TESTING AND SIMULATION RESULTS

7.1 Test 1: Testing results of Auxiliary Inputs of Analog Section

For monitoring the analog parameters, there are various sensors in the field. The following calculations are done for each channel of the auxiliary inputs:

7.1.1 Channel 1 Measurements

Channel 1 auxiliary input calculations are shown in below in table 7.1.

Table 7.1 Auxiliary Input Channel 1 Calculations

ADC Count	Current (mA)	Calculated Voltage (V)	Calculated Resistance (E)	Calculated Current (mA)	Error %
3337008	4	0.2486	62.1566	4.0026	0.0646
6666528	8	0.4967	62.0869	7.9962	-0.0477
9974860	12	0.7432	61.9321	11.9644	-0.2978
13381997	16	0.9970	62.3148	16.0511	0.3182
16667722	20	1.2418	62.0921	19.9921	-0.0393

These calculations are done based on the procedure explained in figure 6.1 and steps given in section 6.1.1. A graph is plotted between input current and calculated output current which is shown in figure 7.1.

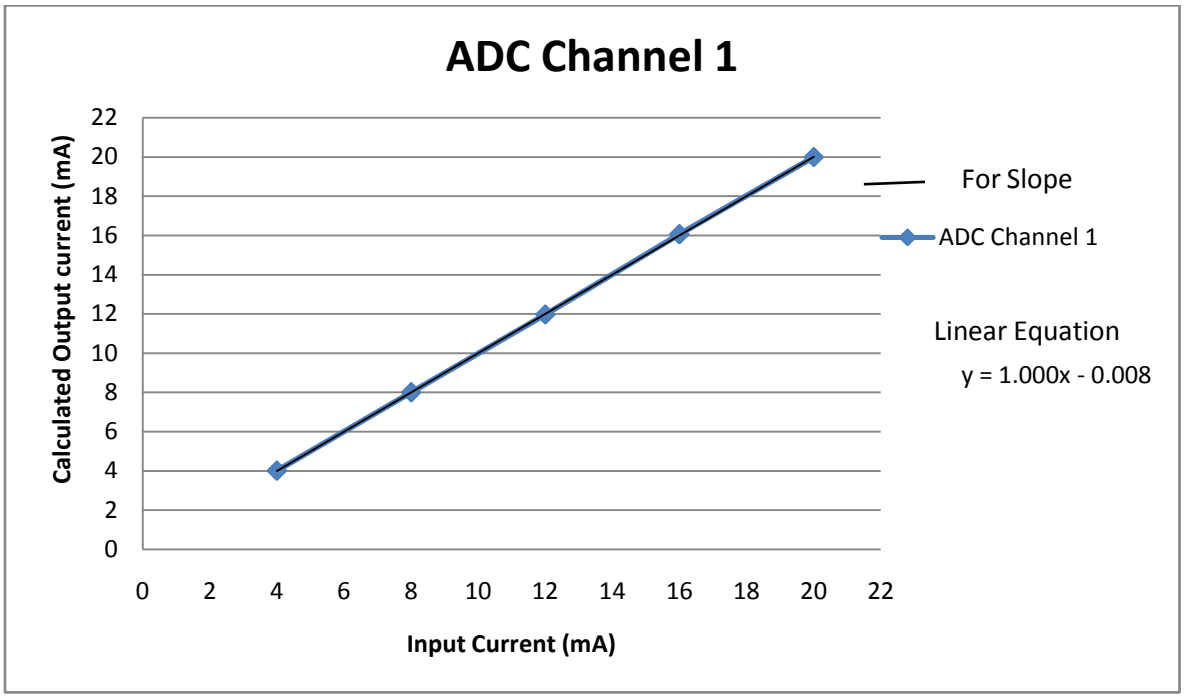

Figure 7.1 Graph between I/p current & Calculated current for channel 1

This figure shows a linear graph between input current and calculated output current. It means that output follows the input accurately which is the desired result. As indicated on the graph, the line equation

$$y = 1.000x - 0.008$$

shows that

- **slope of the line = 1,** which shows that for every increase of 1 in '*x'*, there is also an increase of 1 (same amount) in '*y'*.
- **y-intercept = -0.008**, which shows that the line intercepts the *y-axis* at *(0,-0.008)*.

7.1.2 Channel 2 Measurements

Channel 2 auxiliary input calculations are shown in below in table 7.2.

Table 7.2 Auxiliary Input Channel 2 Calculations

ADC Count	Current (mA)	Calculated Voltage (V)	Calculated Resistance (E)	Calculated Current (mA)	Error %
3330010	4	0.2481	62.0263	4.0143	0.3551
6659666	8	0.4962	62.0230	8.0281	0.3498
9941990	12	0.7407	61.7280	11.9849	-0.1264
13258377	16	0.9878	61.7391	15.9827	-0.1083
16513764	20	1.2304	61.5186	19.9070	-0.4672

The same steps are followed for this channel 2 as described for channel 1 in section 6.1.1. The graph obtained for this input channel is shown below in figure 7.2.

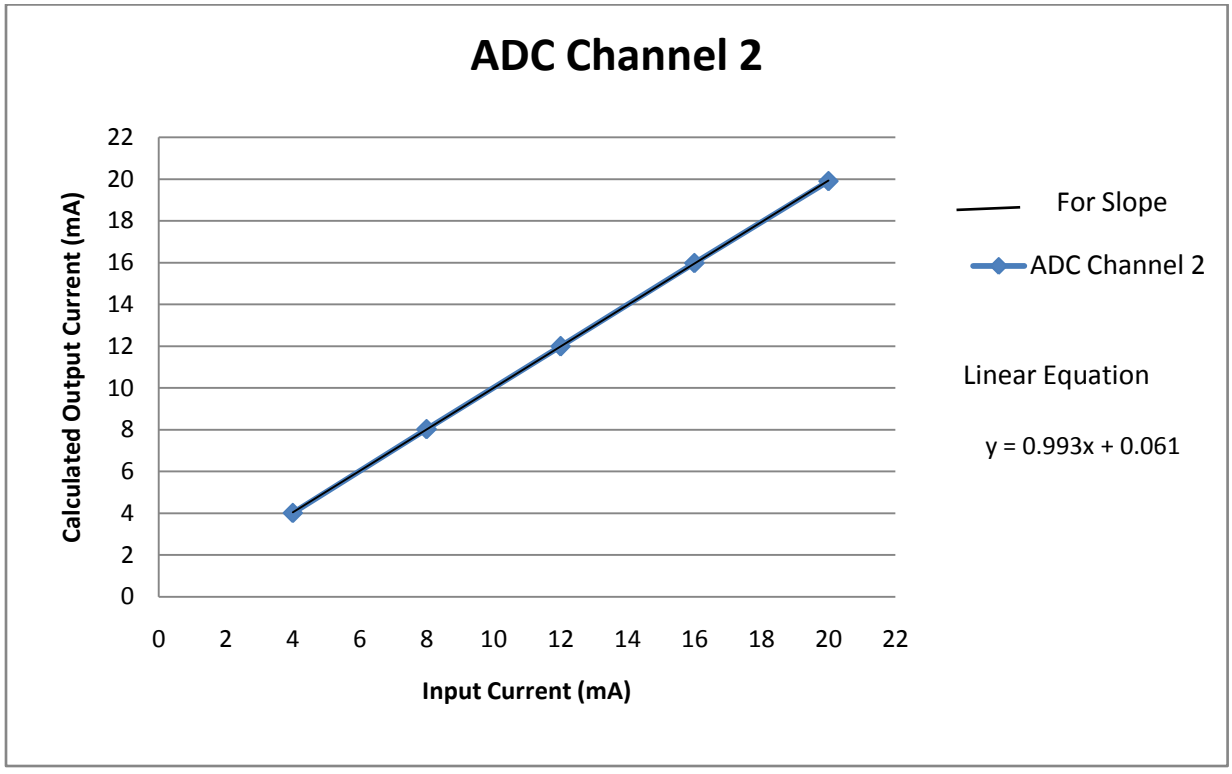

Figure 7.2 Graph between I/p current & Calculated current for channel 2

This figure also shows a linear graph between input current and calculated output current. It means the output current follows the input current accurately which is the desired result. As indicated on the graph, the line equation

$$y = 0.993x + 0.061$$

shows that

- **slope of the line = 0.993 (~ 1),** which shows that for every increase of 1 in '*x*', there is also an increase of 1 (approximately same amount) in '*y*'.
- **y-intercept = 0.061**, which shows that the line intercepts the *y-axis* at *(0,0.061)*.

7.1.3 Channel 3 Measurements

Channel 3 auxiliary input calculations are shown in below in table 7.3.

Table 7.3 Auxiliary Input Channel 3 Calculations

ADC Count	Current (mA)	Calculated Voltage (V)	Calculated Resistance (E)	Calculated Current (mA)	Error %
3330010	4	0.2481	62.0263	4.0148	0.3675
6659666	8	0.4962	124.0459	8.0291	0.3622
9941990	12	0.7407	185.1840	11.9863	-0.1140
13258377	16	0.9878	246.9565	15.9847	-0.0959
16513764	20	1.2304	307.5928	19.9095	-0.4548

The same steps are followed for this channel 3 as followed for channel 1 and 2. The graph obtained for this input channel is shown below in figure 7.3.

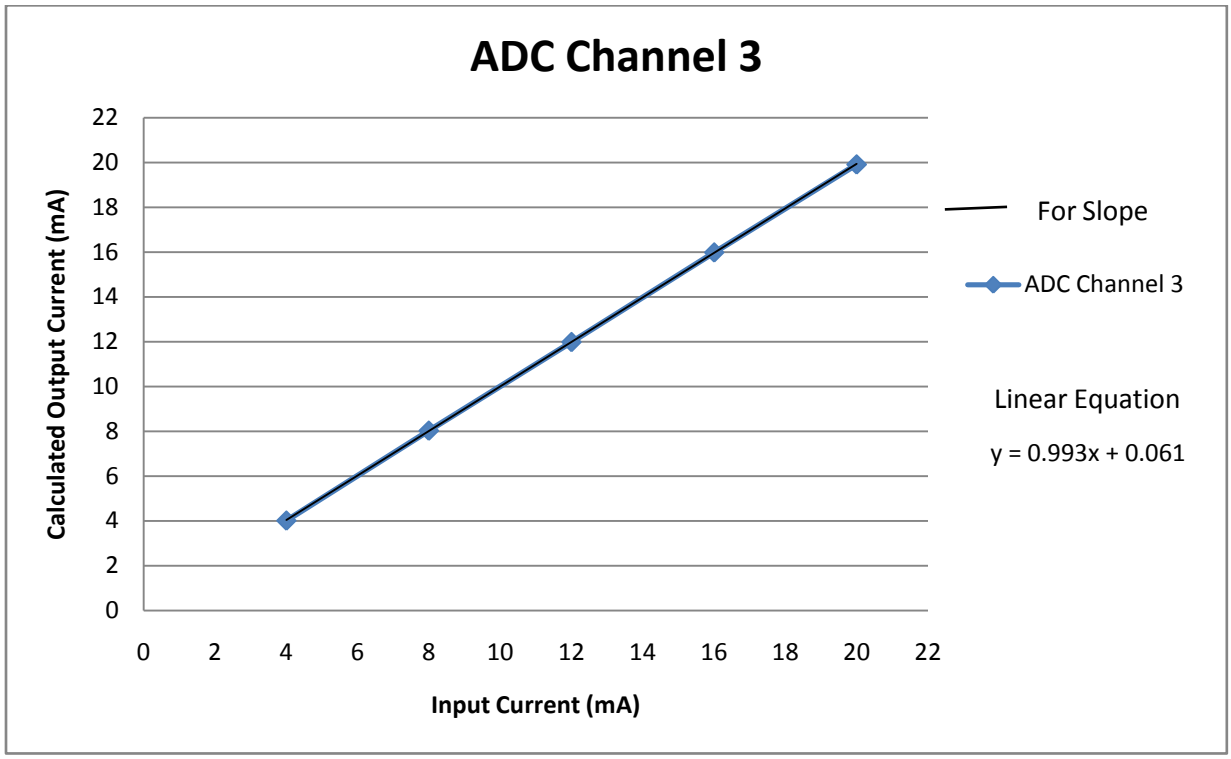

Figure 7.3 Graph between I/p current & Calculated current for channel 3

This figure also shows a linear graph between input current and calculated output current. It means that the output follows the input accurately which is the desired result. As indicated on the graph, the line equation

$$y = 0.993x + 0.061$$

shows that

- **slope of the line = 0.993 (~ 1),** which shows that for every increase of 1 in '*x*', there is also an increase of 1 (same amount) in '*y*'.
- **y-intercept = 0.061**, which shows that the line intercepts the *y-axis* at *(0,0.061)*.

7.2 Test2: Testing results of Current Inputs with the help of Current Transformer

Based on the figures and the testing procedure given in section 6.2.1, the following calculations can be done for 1A and 5A current inputs as given below.

7.2.1 Measurement of 1A current

The calculations for 1A current are shown in table 7.4.

Table 7.4 Calculations for 1A current

Current (A)	Voltage(across 150E)	Expected Voltage (Vr)
0.2	0.041	0.042
0.5	0.104	0.105
1	**0.207**	0.21
1.5	0.307	0.315
2	0.408	0.42
2.5	0.513	0.525
3	**0.605**	**0.63**
3.5	0.671	0.735
4	0.722	0.84
4.5	0.762	0.945

Accurate

Saturate

According to these calculations, a graph is plotted between input current and the output voltage as shown in figure 7.7.

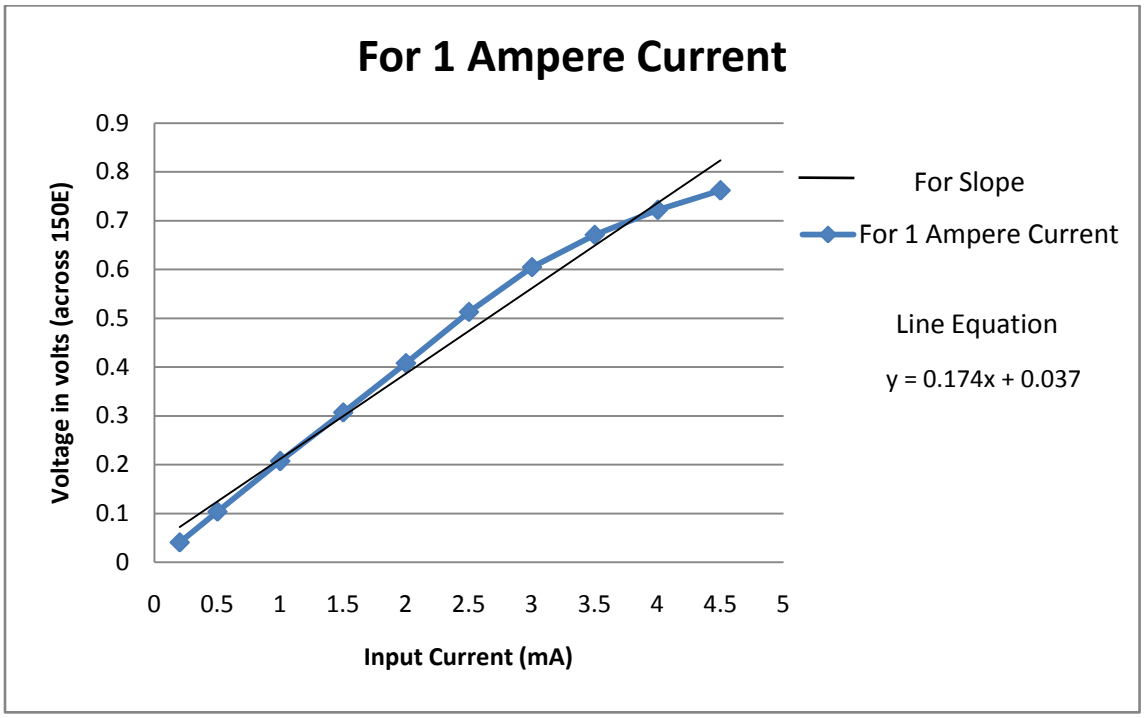

Figure 7.7 Graph between I/p Current & Measured Voltage for 1A Current

This figure shows that the graph between input current and the measured voltage is linear up to an input current of 3A, after that is starts saturating. As indicated on the graph, the line equation

$$y = 0.174x + 0.037$$

shows that

- **slope of the line = 0.174,** which shows that for every increase of 1 in '*x*', there is an increase of 0.174in '*y*'.
- **y-intercept = 0.037**, which shows that the line intercepts the *y-axis* at *(0,0.037)*.

7.2.2 Measurement of 5A current

The calculations for 5A current are shown in table 7.5.

Table 7.5 Calculations for 5A current

Current (A)	Voltage (across 150//36 E)	Expected Voltage (Vr)
5	0.208	0.203
5.5	0.226	0.223
6	0.247	0.243
6.5	0.268	0.264
7	0.289	0.284
7.5	0.31	0.304
8	0.33	0.325
8.5	0.35	0.345
9	0.37	0.365
9.5	**0.389**	0.386
10	0.405	0.406

Accurate

Saturate

Here also, a graph is plotted between input current and the output voltage as shown in figure 7.8.

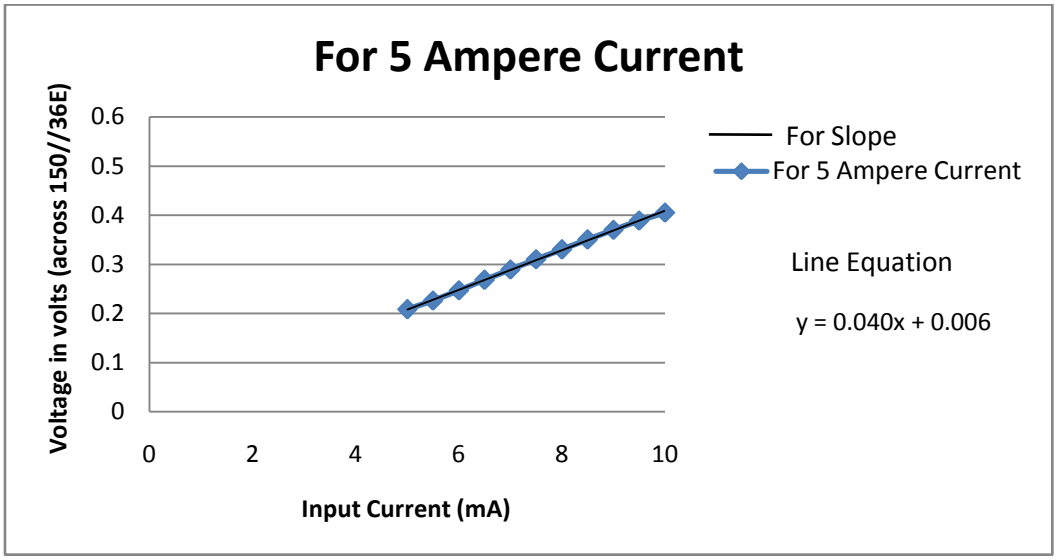

Figure 7.8 Graph between I/p Current & Measured Voltage for 5A Current

This figure shows that, in this case, the graph between input current and the measured voltage is a straight line which is the desired result. Here, the starting input current is high (i.e. 5A) and the output is in mV, so it can not be a linear graph as was expected. As indicated on the graph, the line equation

$$y = 0.040x + 0.006$$

shows that

- **slope of the line = 0.040,** which shows that for every increase of 1 in '*x*', there is also an increase of 0.04 in '*y*'.
- **y-intercept = 0.006**, which shows that the line intercepts the *y-axis* at *(0,0.006)*.

7.3 Test 3: Testing results of Voltage Inputs with the help of Potential Transformer (PT)

Based on the figure and the testing procedure described in section, the following calculations can be done for the testing of PT as shown below in table 7.6.

Table 7.6 PT Testing

Vprimary	Vsecondary
10.14	0.189
20.2	0.387
30.0	0.574
40.1	0.767
50.2	0.956
60.2	1.151
63.6	**1.207**
70.3	1.326
80.0	1.490

Accurate

According to these calculations, a graph is plotted between primary and secondary voltage of the PT as shown in figure 7.10.

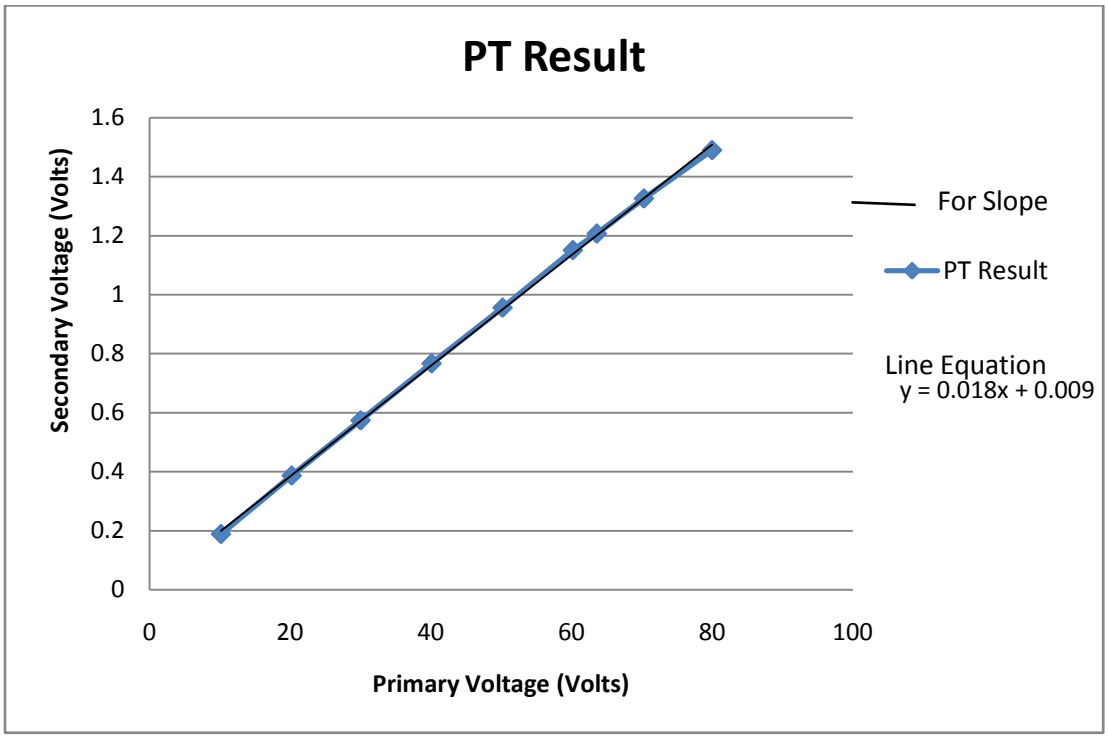

Figure 7.10 Graph between Vsecondary & Vprimary of PT

This figure shows that the graph between primary and secondary voltages of a PT is linear. It means the secondary output voltage is following the primary input voltage which is the desired result. As indicated on the graph, the line equation

$$y = 0.018x + 0.009$$

shows that

- **slope of the line = 0.018,** which shows that for every increase of 1 in '*x*', there is also an increase of 0.018 in '*y*'.
- **y-intercept = 0.009**, which shows that the line intercepts the *y-axis* at *(0,0.009)*.

7.4 Testing results of DC1 and DC2 channels

Based on the figure and the testing procedure described in previous chapter, the following calculations can be done for both the channels, DC1 and DC2, as shown below.

7.4.1 DC1 Measurements

The DC1 measurements are shown in table 7.7.

Table 7.7 DC1Measurements

DC1 (volts)	ADC Count (DC1)	Observed DC1 (volts)	DC1 Voltage Error (%)
70	16408	70.0560731	-0.080104523
80	18764	80.11531911	-0.144148958
90	21126	90.20018294	-0.22242549
100	23415	99.9733638	0.026636203
110	25741	109.9045209	0.086799158
120	28080	119.8911832	0.090680639
130	30371	129.6729033	0.25161281
140	32703	139.6296983	0.264515535
150	35000	149.4370161	0.375322574
160	37297	159.244354	0.472278733
170	39635	169.2267467	0.45485488
180	42303	180.618117	-0.343398313
190	44673	190.7371378	-0.387967248
200	47026	200.7835749	-0.391787442
210	49300	210.4927113	-0.234624431
220	51627	220.4281361	-0.19460821
230	53962	230.3977219	-0.172922545
240	56273	240.2048345	-0.110347728
250	58585	250.1362169	-0.054486749
260	60910	260.0631044	-0.024270912
270	63221	269.9302171	0.025845531
280	65384	279.1654247	0.29806262

According to these calculations, a graph is plotted between Input Voltage and output voltage as shown in figure 7.12.

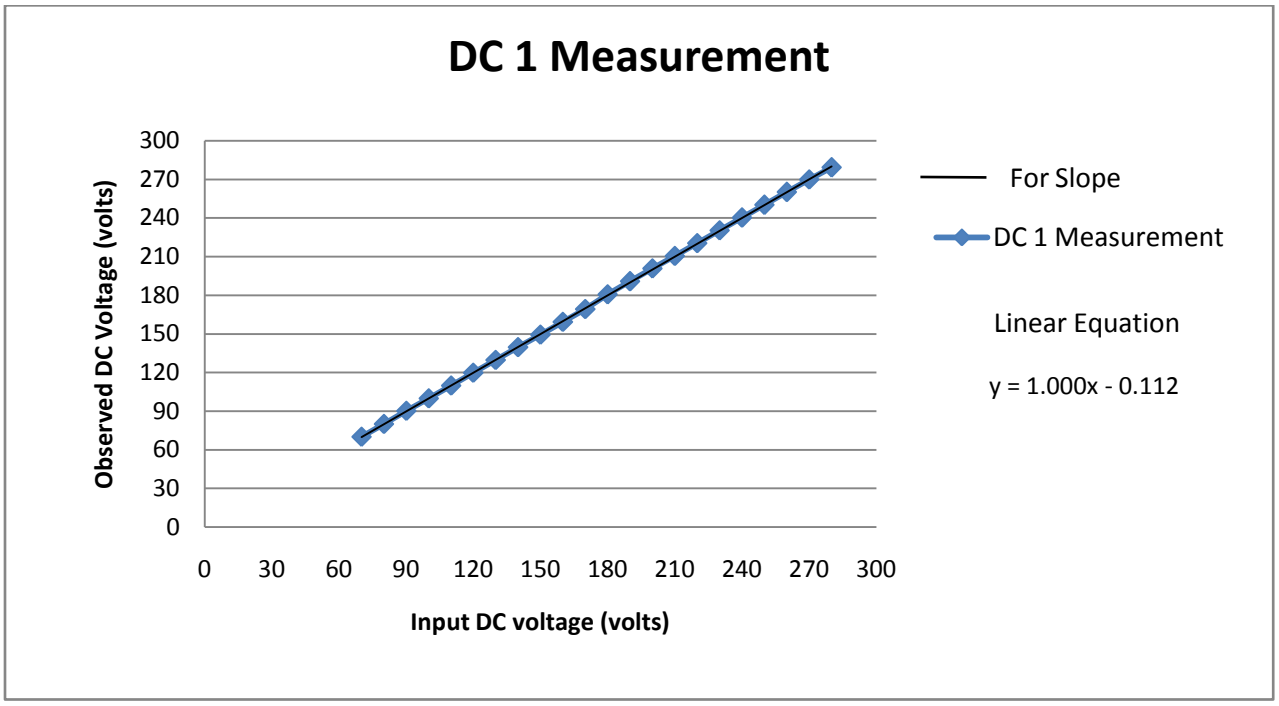

Figure 7.12 Graph between Observed & I/p voltage for DC1 Measurement.

This figure shows that the graph between applied input voltage and the observed output voltage is linear. It means the observed DC output voltage follows the applied DC input voltage which is the desired result. As indicated on the graph, the line equation

$$y = 1.000x + 0.112$$

shows that

- **slope of the line = 1,** which shows that for every increase of 1 in '*x*', there is also an increase of 1 in '*y*'.
- **y-intercept = 0.112**, which shows that the line intercepts the *y-axis* at *(0,0.112)*.

7.4.2 DC2 Measurements

The DC2 measurements are shown below in table 7.8.

Table 7.8 DC2 Measurements

DC2 (volts)	ADC Count (DC2)	Observed DC2 (volts)	DC2 Voltage Error (%)
70	16158	70.02290993	-0.032728464
80	18481	80.08994915	-0.112436434
90	20820	90.22632657	-0.251473961
100	23076	100.0030121	-0.003012095
110	25396	110.0570504	-0.051864005
120	27709	120.0807533	-0.06729438
130	29981	129.926777	0.056325399
140	32276	139.8724744	0.091089741
150	34580	149.8571745	0.095217015
160	36838	159.6425273	0.223420449
170	39157	169.692232	0.181040026
180	41575	180.1709667	-0.094981507
190	43947	190.4503542	-0.237028504
200	46266	200.5000588	-0.250029416
210	48496	210.1640698	-0.078128473
220	50785	220.0837654	-0.03807516
230	53079	230.0151291	-0.010925693
240	55372	239.9621592	0.015766999
250	57612	249.6695045	0.132197386
260	59951	259.805884	0.074660018
270	62233	269.6952441	0.112872574
280	64455	279.3245859	0.241219316

Same steps are followed as for DC1 measurements and a graph is plotted between input and output as shown in figure 7.13.

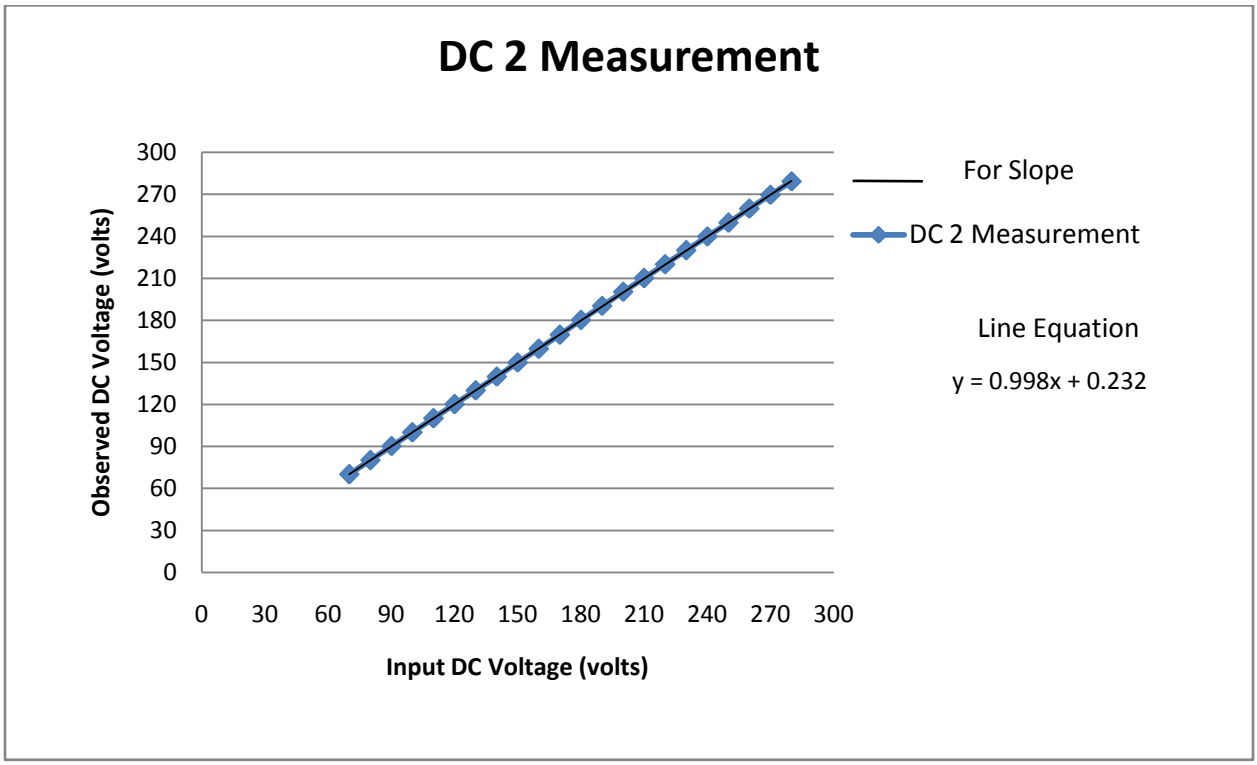

Figure 7.13 Graph between Observed & I/p voltage for DC2 Measurement

This figure shows that the graph between applied input voltage and the observed output voltage is linear. It means the observed DC output voltage follows the applied DC input voltage. It is the desired result. As indicated on the graph, the line equation

$$y = 0.998x + 0.232$$

shows that

- **slope of the line = 0.998 (~ 1),** which shows that for every increase of 1 in '*x*', there is also an increase of 1 in '*y*'.
- **y-intercept = 0.232**, which shows that the line intercepts the *y-axis* at *(0,0.232)*.

7.5 Simulations over network protocols

These simulations are represented in a way as shown in figure 6.6 in section 6.5.

7.5.1 Simulation-1: ARP (Address Resolution Protocol)

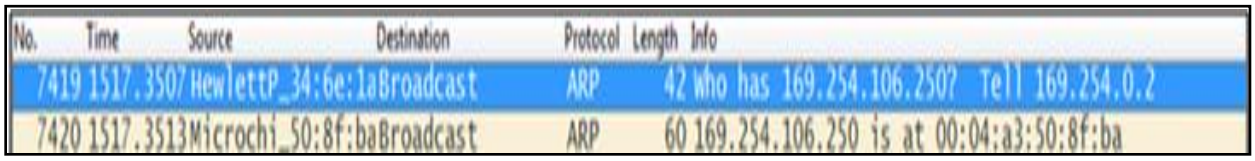

Figure 7.15 ARP Request

From figure 7.15, we can see how the machine with IP *169.254.0.2*, and a Message Authentication Code (MAC) *HewlettP_34:6e:1a*, has launched an ARP request to the broadcast address asking for the MAC of the IP *169.254.106.250* (our network gateway). Immediately afterwards, we get a response with an ARP reply indicating the MAC address.

Figure 7.16 shows the raw data format of an ARP request generated by the machine.

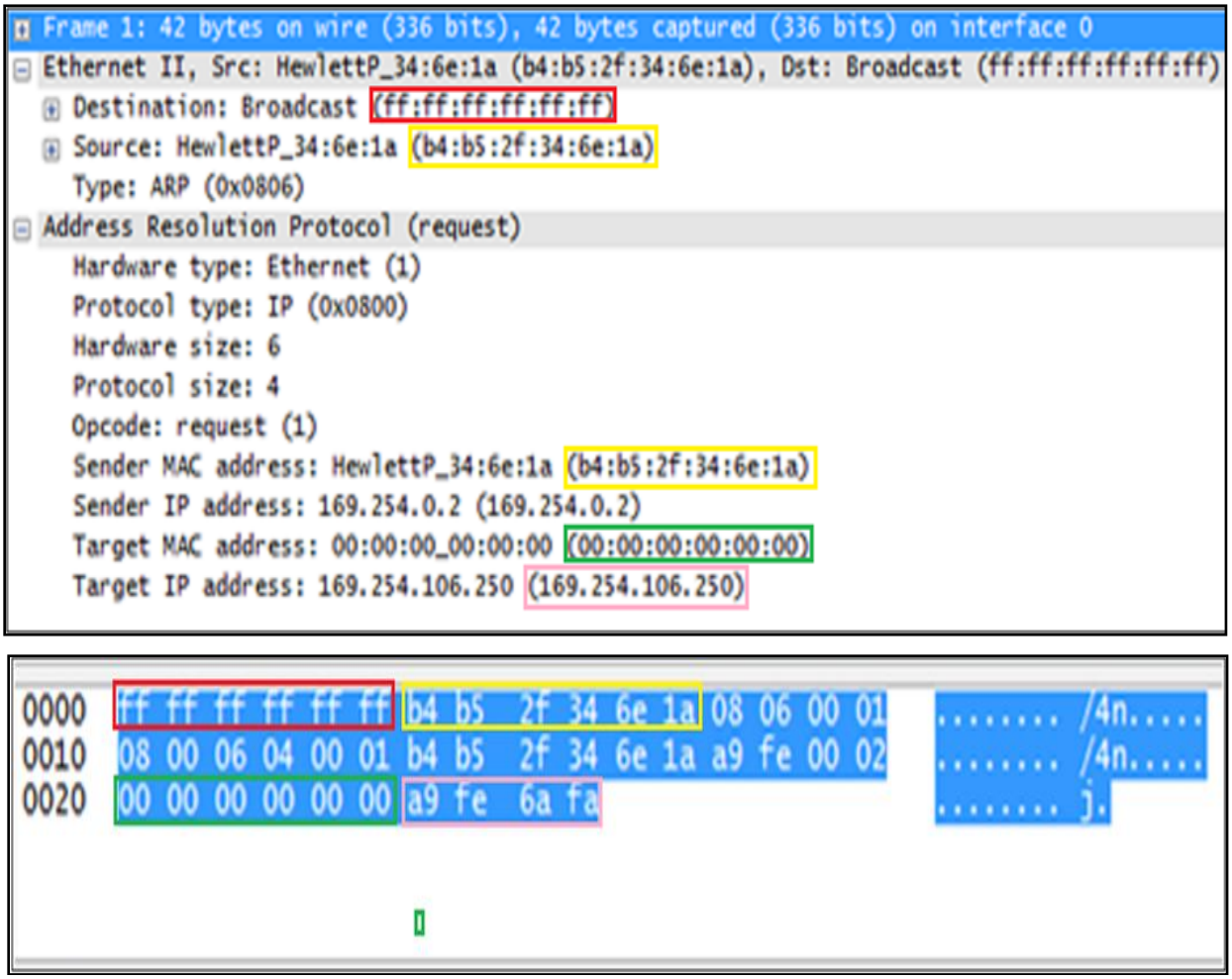

Figure 7.16 ARP Spoof

This figure shows zone3 and zone4 for ARP. The hexadecimal text shown in the lower portion corresponds to the segment transmitted by the network. It is difficult to understand that what these hexadecimal values indicates. It is described by zone3. Following things can be understood from the figure 7.16:

- The red marked values describe destination address which is a broadcast address. As ARP request is sent to all the hosts in the network.
- The yellow marked values describe sender's MAC address.
- The pink marked values describe the target IP address.
- The green marked values describe target MAC address which is 00:00:00:00:00:00 because this ARP request is sent to know target MAC address, so it is unknown.

7.5.2 Simulation-2: TCP (Transmission Control Protocol)

TCP is the most reliable protocol of TCP/IP suite. Its detailed working flow graph details are given below.

7.5.2.1 Flow Graph

The packet sequence can be seen graphically by selecting from the menu *Statistics, >> Flow Graph*. This tool enables to track the behavior of TCP connections because, as can be seen in the image shown in figure 7.17, it intuitively illustrates, using arrows, the source and target of each packet, highlighting the active flags that intervene in each connection flow.

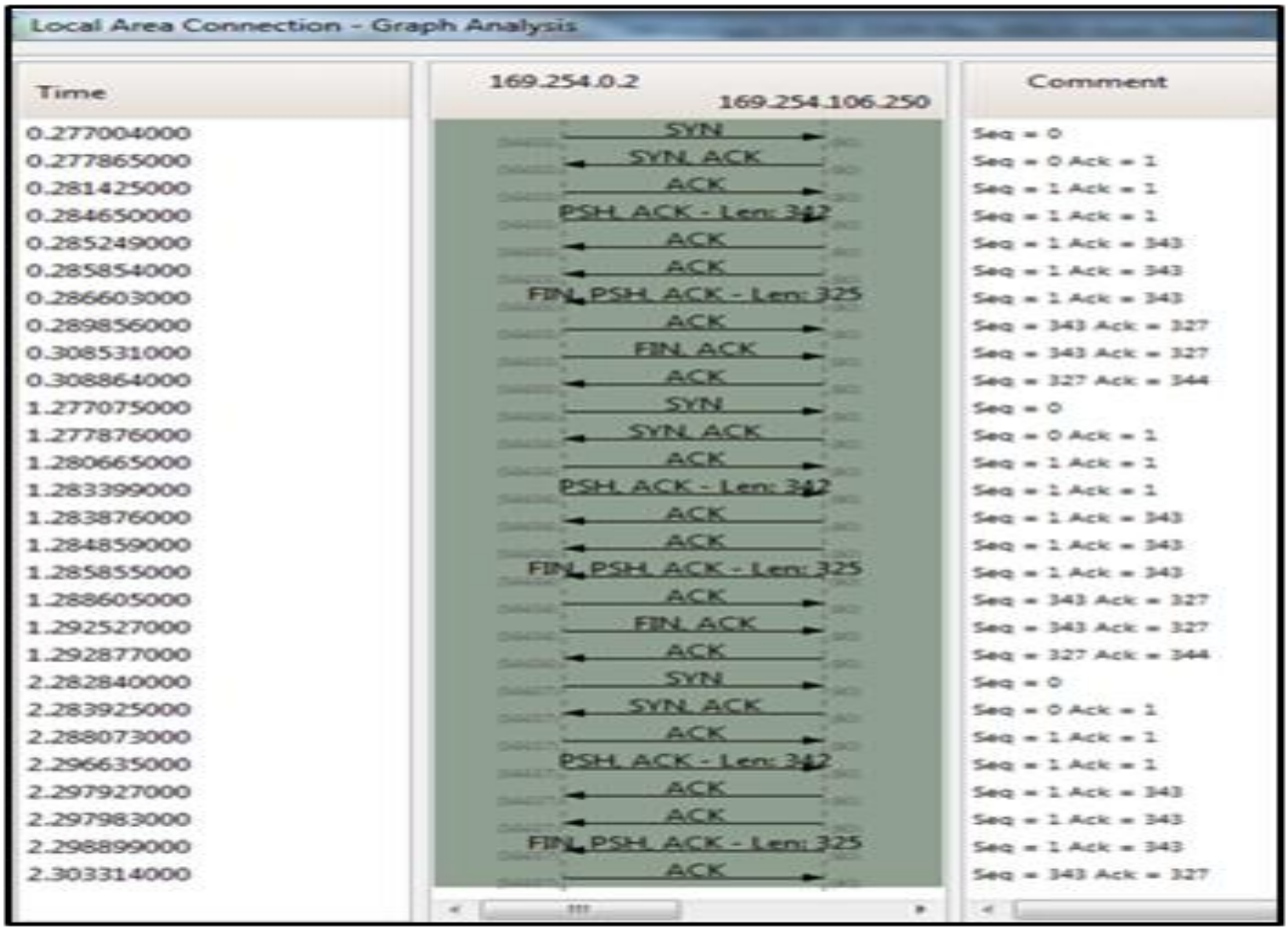

Figure 7.17 TCP Flow Graph

Here, each row represents a single TCP packet. The left column indicates the direction of the packet, TCP ports, segment length, and the flag(s) set. The column at right lists the relative sequence and acknowledgement numbers in decimal.

Understanding this flow graph as:

Packet #1

Each side of a TCP session starts out with a (relative) sequence number of zero. Likewise, the acknowledgement number is also zero, as there is not yet a complementary side of the conversation to acknowledge.

Packet #2

The server responds to the client with a sequence number of zero, as this is its first packet in this TCP session, and a relative acknowledgement number of 1. The acknowledgement number is set to 1 to indicate the receipt of the client's SYN flag in packet #1.

Here, the acknowledgement number has been increased by 1 although no payload data has yet been sent by the client. This is because the presence of the SYN or FIN flag in a received packet triggers an increase of 1 in the sequence.

Packet #3

Like in packet #2, the client responds to the server's sequence number of zero with an acknowledgement number of 1. The client includes its own sequence number of 1 (incremented from zero because of the SYN).
At this point, the sequence number for both hosts is 1. This initial increment of 1 on both hosts' sequence numbers occurs during the establishment of all TCP sessions.

Packet #4

This is the first packet in the stream which carries an actual payload (specifically, the client's HTTP request). The sequence number is left at 1, since no data has been transmitted since the last packet in this stream. The acknowledgement number is also left at 1, since no data has been received from the server, either. This packet's payload is 342 bytes in length.

Packet #5

This packet is sent by the server solely to acknowledge the data sent by the client in packet #4 while upper layers process the HTTP request. Notice that the acknowledgement number has increased by 342 (the length of the payload in packet #4) to 343; e.g., "I have received 343 bytes so far." The server's sequence number remains at 1.

Packet #7

This packet marks the beginning of the server's HTTP response. Its sequence number is still 1, since none of its packets prior to this one have carried a payload. This packet carries a payload of 325 bytes.

Packet #8

The sequence number of the client has been increased to 342 because of the last packet it sent. Having received 325 bytes of data from the server, the client increases its acknowledgement number from 1 to 327.

For the majority of the capture, this cycle is repeated. The client's sequence number will remain steady at 343, because it has no data to transmit beyond the initial 342 byte request. The server's sequence number, in contrast, continues to grow as it sends more segments of the HTTP response.

The last session of this flow graph is not shown in this window, but can be summarized as: After acknowledging the last segment of data from the server, the client processes the HTTP response as a whole and decides no further communication is needed and so in this packet, FIN flag is set by the client and acknowledgement number remains the same as in the prior packet. The server acknowledges the client's desire to terminate the connection by increasing the acknowledgement number by one and setting the FIN flag as well. At this point, both hosts have terminated the session and can release the software resources dedicated to its maintenance.

7.5.2.2 TCP Frame

Similar to the ARP frame, the TCP frame can also be visualized properly with the help of the used network protocol analyzer as shown in figure 7.18.

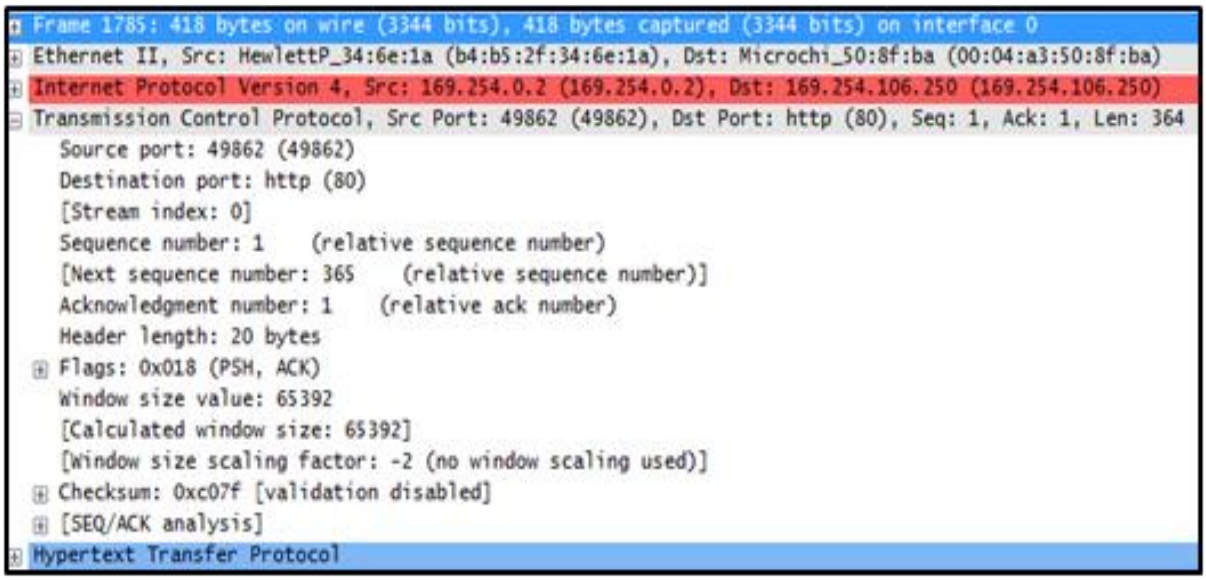

Figure 7.18 TCP Frame

This figure clearly shows every parameter of the frame as- source port, destination port, sequence number, acknowledgement number and header length. From this, the actual information required for requesting data by TCP to the application layer protocols (HTTP) can be understood.

When a host initiates a TCP session, its initial sequence number is effectively random; it may be any value between 0 and 4,294,967,295, inclusive. However, protocol analyzers like Wireshark will typically display *relative* sequence and acknowledgement number in place of the field's actual value as shown in figure 7.18. These values are relative to the initial sequence number of that stream. This is handy, as it is much easier to keep track of relatively small, predictable numbers rather than the actual numbers sent on the wire.

7.5.3 Simulation-3: HTTP (Hyper Text Terminal Protocol)

In this simulation, information about HTTP protocol can be seen as given in figure 7.19.

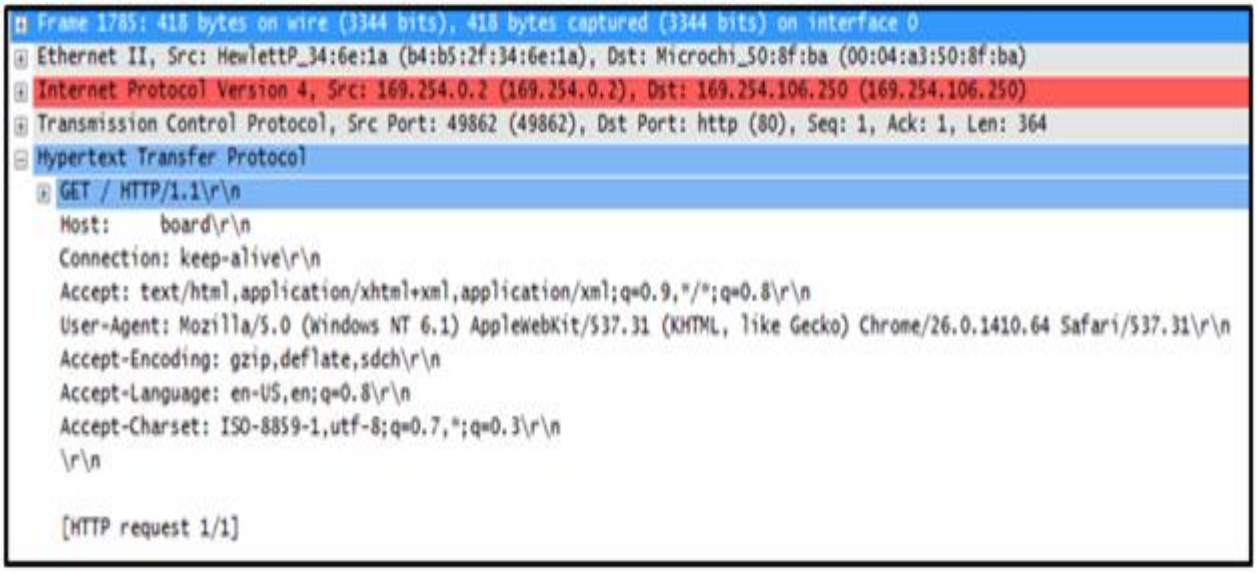

Figure 7.19 HTTP Frame

From this snapshot, following things can be understood:

- First line "GET /HTTP/1.1\r\n" shows the used form method and the HTTP version.
- Here, the "User-Agent" acts as a client within this client-server distributed computing system. Its format is:
  User-Agent: Mozilla/[version]([system and browser information])[platform]([platform details])[extensions]
- Accept Encoding field indicates the compression schemas supported to the server as HTTP data is compressed before it is sent from the server to make better use of bandwidth, and provide greater transmission speeds between both. Here, these schemas are gzip, deflate and sdch.

7.5.4 Simulation-4: Packets during interaction of web server with the board

The packets captured during the interaction of web server with the board can be seen through the snapshot given in figure 7.20.

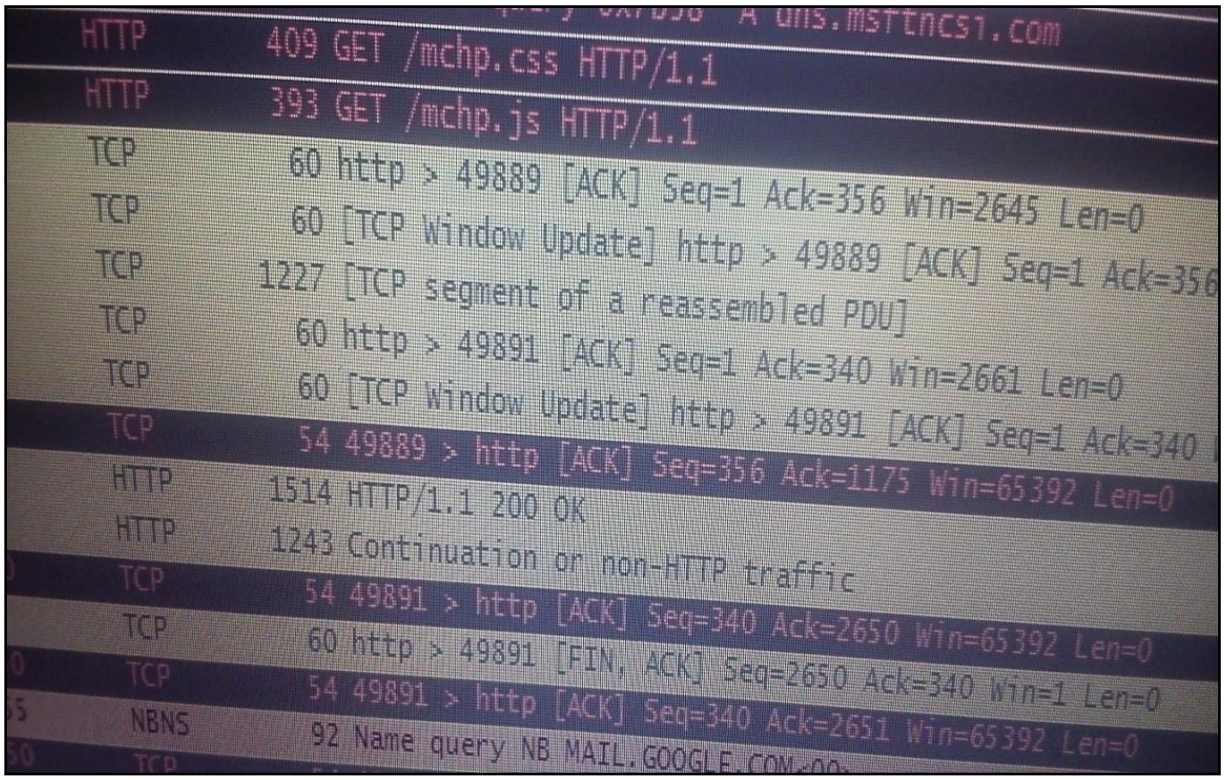

Figure 7.20 Packets during interaction of web server with the board

In this figure, a HTTP request is sent to a web server, in which the client web server requests the dynamic web page by requesting its ".css (Cascading Style Sheets)" and ".js (JavaScript)" files. Finally, the server returns an HTTP/1.1 200 OK response which includes the files requested. In between, TCP helps in sending request to the receiver so that the server can reliably send the requested data to the client.

This chapter describes all the testing and simulation results related to this dissertation. The testing, for checking how accurately controller detects the inputs coming from the sensors as their status, transmission line parameters and DC inputs, is done and also got the desired results which are discussed in this chapter. Also, many network protocols, ARP, TCP, IP, HTTP, are analyzed with the help of Wireshark and so, the snapshots of this analysis are described as the simulation results in this chapter.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

The dissertation deals with the design and implementation of an Ethernet module. Ethernet, which is the data link layer protocol in the TCP/IP stack, is used to make interaction between the RTU and the substations. The analysis of network protocols and their interaction with each other was carried out through simulation using Wireshark network protocol analyzer. A stand-alone RTU has also been proposed. This RTU contains various sections: analog inputs section, digital inputs section, digital outputs section, communication and power supply section, . The testing of analog section has been emphasized in this study. The field parameters like status of field devices, transmission line parameters and DC voltage has been measured by the controller using, an external ADC. Testing has also been done to make sure the measurement of correct parameters by the controller. The following conclusions are drawn based on the testing and simulations:

- The testing of auxiliary inputs showed that the controller detects these inputs accurately for all the 3 input channels with minute difference that can be neglected.

- The testing of voltage inputs showed that the accurate output (threshold value which a controller can detect) of PT is obtained at 63.5V.

- Testing has been done for both 1A and 5A current inputs using CTs. The results indicate that with 1A of current input, the accurate output (threshold value which a controller can detect) of the CT is obtained at 1A and at 5A for 5A current input.

- The testing of both the DC1 and DC2 channels showed that the controller detects these inputs accurately for both the 2 input channels with minute difference that can be neglected.

- The actual frames of network protocols, like ARP, IP, TCP and HTTP has been shown through simulation and so the information like destination address, source address, sequence and acknowledgement number of packets (in case of TCP), reply packet or response packet (in case of ARP) and checksum validity can be extracted.

- TCP flow diagram has also been generated to know each step of packet transmission. It is concluded that every transmission should get an acknowledgement for reliable communication.

- The packets, during the interaction of web server with the board, were also captured. From this, it is concluded that firstly, the client web server can request the dynamic web page by requesting its .css and .js files and secondly, the server can return an HTTP/1.1 200 OK response including the requested files.

8.2 Future Scope

Optical fiber communication systems have attracted the attention of researchers as a desired mode of communication all over the world due to its high capacity, efficiency and security. In this study, only 5MBd (Mega Baud) link was tried using 1 m length multimode fiber cable with HFBR-1414/2412 as transmitter and receiver respectively. There is a need to move from such speed links to gigabits and terabits fiber optic technology for using high speed communication.

Out of many fiber optic gigabit transceivers available in the market e.g. XENPAK, X2, XFP and SFP, the later is Small Form-factor Pluggable Transceiver having advantages of less power consumption ($\leq$1W), having the smallest form factor and its ability to reach up to hundreds of kilometers even at higher speeds of up to 5Gbps compared to other gigabit trans receiver. Lot of scope exists in the implementation of the present low link fiber optic communication used in the present RTU, and the implementation of the gigabit fiber optic communication link using SFP transceiver in the same RTU.

REFERENCES

[1] GarrettCom, Inc.,"*Ethernet in Power Utilities Substations – The Changing Role of Fiber Media*", A White Paper for Network Engineers.

[2] Sanath Alahakoon, Lilantha Samaranayake, Thilakasiri Vijayananda, Mats Leksell, *"Remote Monitoring And Distributed Real-Time Control Via Ethernet",* Proceedings of the 11th National Conference on Machines and Mechanisms held at the Indian Institute of Technology Delhi, New Delhi, December 18-19, 2003.

[3] Deepa Chekka and Ravi Kanth, "*Design and development of embedded web server based on Arm9 and Linux",* World Journal of Science and Technology, 94-97, 2012.

[4] Hairulzawan Hashim, Zainal Alam Haron, "*A Study on Industrial Communication Networking: Ethernet Based Implementation",* International Conference On Intelligent And Advanced Systems (Icias2007), KLConvetion Centre, Kuala Lumpur, 25 - 28 November 2007.

[5] Fred Steinhauser*, "New challenges with substations utilizing communication networks*," Proceedings of IEEE Power Technology Conference, vol.2, 2003.

[6] Lilantha Samaranayake, Sanath Alahakoon, *"Condition Monitoring and Distributed Real-Time Control of Industrial Drive Systems via Ethernet"* , Proceedings of the Invited lecture for the annual general meeting, Central province of the Institute of Engineers Sri Lanka (IESL), Kandy, Sri Lanka, 32-38, September 2003.

[7] Ammar Ibrahim Majeed, "*Distributed Real-Time Monitoring and Control of Industrial Drive Systems via Ethernet",* Journal of Engineering and Development, 16(2), ISSN 1813-7822, June 2012.

[8] Mahmood Qureshi, Ali Raza, Dileep Kumar, Sang–Sig Kim, Un–Sig Song, Min–Woo Park, Hyuk–Soo Jang,Hyo–Sik Yang, Byung–Seok Park, "*A Survey of Communication Network Paradigms for Substation Automation",* IEEE International Symposium on Power Line Communications and Its Applications (ISPLC), 2-4 April 2008.

[9] Clemens Hoga, "*New Ethernet Technologies for Substation Automation",* IEEE Power Tech Conference, 707 – 712, Lausanne, 2007.

[10] Skeie T, Johannessen S, Brunner C, *"Ethernet in Substation Automation*", IEEE Control Systems Magazine, 43-51, June 2002.

[11] Chong FU, Zhi-liang ZHU, Xiao-xing GAO and Pei-rong WANG, "The Design and Implementation of a General Reduced TCP/IP Protocol Stack for Embedded Web Server", IEEE International Conference on Industrial Electronics and Control Applications (ICIECA), Quito, Ecuador, 2005.

[12] Ahmad Ashari, "*Distributed Monitoring and Controlling Using Microcontroller and Virtual Internet Protocol",* Telkomnika Indonesian Journal of Electrical Engineering, 8(3), 285 – 292, ISSN: 1693-6930, December 2010.

[13] Lucia Lo Bello, Orazio Mirabella, *"Design issues for Ethernet in automation",* Proceedings of8th IEEE International Conference on Emerging Technologies and Factory Automation, 213 – 221vol.1, 15-18, Antibes-Juan les Pins, France Oct. 2001.

[14] Control Wave, *"Bristol® ControlWave® Micro"*, Product Overview**,** June 26, 2007.

[15] Siemens, "*TG5700 RTU Technical Manual",* Document Number 1015717000, Document Revision F, April 2004.

[16] *Motorola,* "*High Performance Monitoring and Control-ACE3600 Remote Terminal Unit", 2009.*

[17] Kalki Communication Technology Pvt. Ltd., "*Sync 2100 series"*, Intelligent RTU/Substation Controller, Banglore.

[18] Kyairul Azmi Bin Baharin, "*Remote Terminal Unit (RTU) for Distribution Automation System (DAS)",* Faculty of Electrical Engineering, National Technical University College of Malaysia, November, 2005.

[19] http://www.hubbellpowersystems.com/switching/dist/overhead/motor/, referred on 23rd May, 2013.

[20] Salvadori F, Gehrke C.S, Campos, Sausen P.S, Oliveira A.C, "A Hybrid Network Architecture Applied to Smart Grid", International Journal of Computing and Network Technology 1, No. 1, 45-59, SPC, University of Bahrain, 2013.

[21] Power Finance Corporation, *"Technical Requirements of RTU",* Section 3, Chapter-1, SCADA/DMS, system under Part A–R-APDRP, Model Technical specification, MGVCL/RAPDRP/SIA/1140.

[22] http://www.iitk.ac.in/infocell/Archive/dirmar1/power _ distribution.html, referred on 3rd May, 2013.

[23] Atmel Corporation, "*AT91 ARM Thumb Microcontroller",* 2008.

[24] Motorola Inc., "*A 16 Digital Input 110V Module for the MOSCAD-L RTU"*, 1999.

[25] http://www.maximintegrated.com/app-notes/index.mvp/id/723, referred on 20th May, 2013.

[26] https://forum.sparkfun.com/viewtopic.php?f=19&t=33227, referred on 20th May, 2013

[27] http://labspace.open.ac.uk/mod/resource/view.php?id=372342, referred on 20th May, 2013.

[28] Marzio Pozzuoli, *"Ethernet in Substation Automation Applications-Issues and Requirements",* Proceedings of Western Power Delivery Automation Conference, 2003.

[29] Vahid Shakir, *"Modelling and Simulation of Ethernet Based Networked Mechanical Systems",* Department of Mechanical and Industrial Engineering, Concordia University, Montreal, Quebec, Canada, February 2004.

[30] GE Fanuc Embedded Systems, "Ethernet Switching", 2007.

[31] Veselin Skendzic and Armando Guzmia, "*Enhancing Power System Automation Through The Use Of Real-Time Ethernet", pp.* 480 – 495, Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources, Clemson, SC, 2006.

[32] Michael Simmons, "*Ethernet Theory of Operation*", Microchip Technology Inc., 2008.

[33] JDS Uniphase Corporation, *"White Paper: Fundamentals of Ethernet",* March 2010.

[34] Data Centre, Brocade,"*The Evolution of Ethernet Nomenclature*".

[35] Yu Hsiang Poon, *"Ethernet Speakers",* School of Information Technology and Electrical Engineering, The University Of Queensland, 2003.

[36] Microchip,"*PIC32MX5XX/6XX/7XX Family Data Sheet*", High-Performance, USB, CAN and Ethernet32-bit Flash Microcontrollers.

[37] Micrel Inc.,"*Ethernet Applications"*, Innovation Through Technology.

[38] Micrel Inc., "*KSZ8863MLL/FLL/RLL Integrated 3-Port 10/100 Managed Switch with PHYs"*, Rev. 1.4, September 2011.

[39] George W Benthien, *"Digital Encoding and Decoding",* Revised March 30, 2010.

[40] National Semiconductor Corporation, "*DP83848C PHYTER Commercial Temperature Single Port 10/100 Mb/s Ethernet Physical Layer Transceiver",* Literature Number: SNOSAT2E, May 2008.

[41] Ng Kok Leong, *"Digital Fiber-Optic System",* School of Information Technology and Electrical Engineering, The University Of Queensland.

[42] Avago Technologies,"*HFBR-0400Z, HFBR-14xxZ and HFBR-24xxZ Series Data Sheet"*.

[43] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&d DocName=en537041, referred on 20th May, 2013.

[44] Lydia Parziale, David T. Britt, Chuck Davis, Jason Forrester, Wei Liu, Carolyn Matthews, Nicolas Rosselot, "*TCP/IP Tutorial and Technical Overview,* International Technical Support Organization, December 2006.

[45] Tao Lin, Hai Zhao, Jiyong Wang, Guangjie Han, Jindong Wang, *"An Embedded Web Server for Equipments",* IEEE Computer Society, Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04), 345 – 350, 2004.

[46] http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html, referred on 16th June, 2013.

[47] Jim Brady, "*Build Your Own 8051 Web Server",* Circuit Cellular, The Magazine For Computer Applications, Issue 146, September 2002.

[48] OrCAD Inc., *"OrCAD Capture User's Guide",* First Edition, 30th November, 1998.

[49] Microchip Technology Inc., "*Microchip TCP/IP Stack Help"*, 2011.

[50] Microchip Technology Incorporated, "*TCP/IP Networking",* Part 2: Web-Based Control, Microchip TCP/IP Stack HTTP2 Module, Webinars, 2007.

[51] Borja Merino Febrero, "*Traffic Analysis with Wireshark",* The National Communications Technology Institute, February 2011.