

Smart Sensor System

Project Team:

Asma M.Hisham Shaheen
Lubna Ahmad Al-Herbawi

Supervisor:

Eng. Elayan Abu Gharbyeh

Graduation Project Report

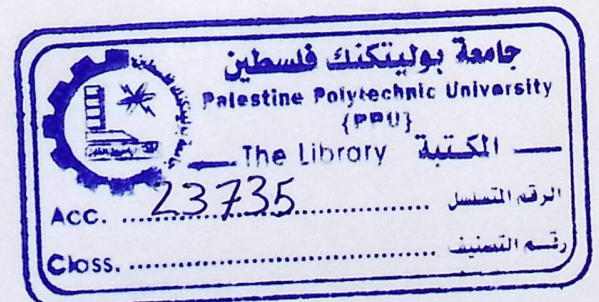
Submitted to the Department of Electrical and Computer Engineering in the College
of Engineering and Technology

Palestine Polytechnic University

This report is submitted in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Systems Engineering

Palestine Polytechnic University
Hebron – Palestine
May-2007

I





Smart Sensor System

Project Team:

Asma M.Hisham Shaheen

Lubna Ahmad Al-Herbawi

Supervisor:

Eng. Elayan Abu Gharbyeh

Graduation Project Report

Submitted to the Department of Electrical and Computer Engineering in the College
of Engineering and Technology

Palestine Polytechnic University

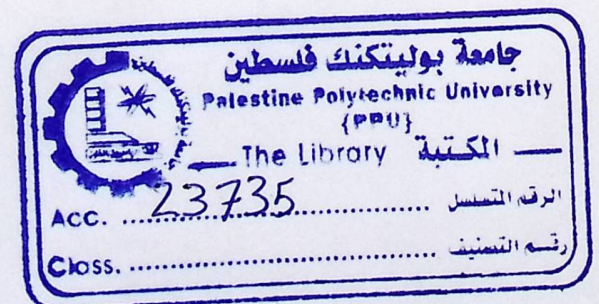
This report is submitted in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Systems Engineering

Palestine Polytechnic University

Hebron – Palestine

May-2007

I



جامعة بوليتكنك فلسطين
الخليل-فلسطين
كلية الهندسة و التكنولوجيا
دائرة الهندسة الكهربائية والحاسوب

اسم المشروع
Smart Sensor System

اسماء الطلبة

لبنى أحمد الحرباوي

أسماء "محمد هشام" شاهين

بناءً على نظام كلية الهندسة والتكنولوجيا وإشراف ومتابعة المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية والحاسوب، وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب.

توقيع المشرف

.....


توقيع اللجنة الممتحنة

.....

توقيع رئيس الدائرة

.....

DEDICATION

To those who give of themselves
so that others may live

To our families for their patience

To our brothers and sisters

To our colleagues for their support and encouragement

To our Supervisor Eng. Elayan Abu-Gharbyeh for his ideas supports
and advices

ACKNOWLEDGMENT

Abstract

All praise be to Allah The Ultimate Guide and The Cherisher who gave us the courage and ability to complete this project with a satisfying degree of perfection. The most deserving of our acknowledgments is our respected supervisor **ENG. ELAYAN ABU-GHARBEYE** who helped us throughout the research of the present project. Next we want to express our deepest gratitude to our friends **MOHAMMAD ABU-AJAMIA** for taking all the pains to help us in troubleshooting and give us some valuable tips. Thanks are also due to all those who by way of encouragement, prayer or advice might have contributed directly or indirectly towards the actualization of this work. Last but not the least we would also like to thank the cooperative laboratory instructor **SAMI AL-SALAMEEN** FOR letting us to use the facilities at the library freely.

Asma M.H Shaheen

Lubna A. Al-Hirbawi

Dated: 30/05/2007

Abstract

Smart Sensor System

Project Team:

Asma M.Hisham Shaheen.

Lubna Ahmad Al-Herbawi

Palestine Polytechnic University – 2006

Supervisor:

Eng. Elayan Abu-Gharbyeh

Smart Sensor System consists of multiple nodes connected to a network; each node consists of a controller and different kinds of sensors. The system implemented in this project has the ability to search for connecting nodes, one node is implemented and connected to 3 different types of sensors to transfer its value to the network, so any authorized user in the network will have the ability to read the sensors values from specific node, get charts defining sensors values during interval time and processing it.

المخلص

TABLE OF CONTENTS

يتكون النظام من مجموعه من العقد المرتبطة بشبكة الحواسيب، كل عقده تحتوي على معالج ومجموعه مختلفة من المجسات ، النظام التي تم تنفيذه لديه القدرة على البحث عن العقدة الموصولة، العقدة التي تم بناؤها واختبارها موصولة مع 3 أنواع مختلفة من المجسات، حيث تقوم بنقل قراءة هذه المجسات إلى الشبكة، وبذلك أي شخص مسؤول يستخدم هذه الشبكة يكون لديه القدرة على قراءة المجسات المختلفة من العقدة المبنية ، والحصول على رسم بياني للقراءات خلال فتره محده، و معالجتها .

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 General idea about the project and its importance	2
1.2 Literature Review	3
1.3 Estimated Cost	4
1.4 Time Schedule	5
1.4.1 The First Time Planning	5
1.4.2 The Second Time Planning	6
1.5 Risk Management	7
1.5.1 Hardware Risk	7
1.5.2 Green Risk	7
1.5.3 Requirement risk	8
1.5.4 Project risk	8
1.6 Report Contents	9

TABLE OF CONTENTS

DEDICATION	III
ACKNOWLEDGMENT	IV
ABSTRACT	V
TABLE OF CONTENTS	VII
LIST OF TABLES	XI
LIST OF FIGURES	XII
CHAPTER ONE	1
INTRODUCTION	1
1.1 General idea about the project and its importance	2
1.2 Literature Review	3
1.3 Estimated Cost.....	4
1.4 Time Schedule	5
1.4.1 The First Time Planning.....	5
1.4.2 The Second Time Planning.....	6
1.5 Risk Management.....	7
1.5.1 Hardware Risk.....	7
1.5.2 Group risk.....	7
1.5.3 Requirement risks.....	8
1.5.4 Project risks.....	8
1.6 Report Contents	9

TABLE OF CONTENTS

DEDICATION	III
ACKNOWLEDGMENT	IV
ABSTRACT	V
TABLE OF CONTENTS	VII
LIST OF TABLES	XI
LIST OF FIGURES	XII
CHAPTER ONE	1
INTRODUCTION	1
1.1 General idea about the project and its importance	2
1.2 Literature Review	3
1.3 Estimated Cost.....	4
1.4 Time Schedule	5
1.4.1 The First Time Planning.....	5
1.4.2 The Second Time Planning.....	6
1.5 Risk Management.....	7
1.5.1 Hardware Risk.....	7
1.5.2 Group risk.....	7
1.5.3 Requirement risks.....	8
1.5.4 Project risks.....	8
1.6 Report Contents	9

CHAPTER TWO	11
<hr/>	
THEORETICAL BACKGROUND	11
<hr/>	
2.1 Project Components.....	12
2.2 Software Theoretical Background.....	13
2.2.1 TCP/IP.....	13
2.3 Hardware Theoretical Background.....	15
2.3.1 Smart Sensor.....	15
2.3.2 IP μ 8930 Micro Web Server.....	18
2.3.2.1 General Description.....	18
2.3.2.2 IP μ 8930 Features.....	19
2.3.3 Sensors.....	20
2.3.3.1 Temperature Sensor (LM35).....	21
2.3.3.2 Light Sensor.....	22
2.3.3.3 Smoke Detector.....	24
2.3.4 10 Base T crossover Cable.....	25
CHAPTER THREE	27
<hr/>	
PROJECT CONCEPTUAL DESIGN	27
<hr/>	
3.1 Detailed project objectives.....	28
3.2 Smart Sensor design Options.....	29
3.3 General Block Diagram.....	32
3.4 How Does System Work?.....	35
CHAPTER FOUR	36
<hr/>	
DETAILED TECHNICAL PROJECT DESIGN	36
<hr/>	
4.1 Detailed description of the project phases.....	37
4.2 Subsystem Detailed Design.....	38
4.2.1 Micro Web Server (IP μ 8930).....	38
4.2.2 Sensing circuits.....	39
4.2.2.1 Light Sensing Circuit.....	40
4.2.2.2 Temperature Sensing Circuit.....	41

4.2.2.3 Smoke Detector Circuit.....	42
4.3 Overall System Design.....	43
4.4 User System Interface.....	44
4.4.1 System Login.....	44
4.4.2 Node selection.....	45
4.4.3 Sensors Values.....	46
4.4.4 Timer reading.....	47
4.4.5 System configuration.....	48
4.4.6 System Help.....	49

CHAPTER FIVE **50**

SOFTWARE SYSTEM DESIGN **50**

5.1 Communicating with Micro Web Server.....	51
5.2 Software Parameters.....	53
5.2.1 Software Tools.....	53
5.3 Micro Web Server Control Protocol	54
5.4 Software Implementation.....	57
5.4.1 Searching for node.....	58
5.4.2 Input data.....	60
5.4.3 Logging file.....	63
5.5 Flowcharts.....	64
5.5.1 Login flow chart (check password).....	64
5.5.2 Ping process flow chart.....	65
5.5.3 Search node flowchart.....	66
5.5.4 Input data flow chart.....	67

CHAPTER SIX **68**

SOFTWARE SYSTEM DESIGN **68**

6.1 Subsystems Testing.....	69
6.1.1 Testing the Micro Web Server.....	70
6.1.1.1 Testing TCP/IP Sockets.....	72
6.1.1.1.1 Testing Port A.....	72
6.1.1.1.2 Testing Parallel Port.....	75
6.1.2 Testing The Temperature Sensor.....	78
6.1.3 Testing The Light Sensor.....	82
6.1.4 Testing The Smoke Detector.....	84
6.2 Software Testing.....	85

CHAPTER SEVEN	89
PROBLEMS, CONCLUSIONS, AND FUTURE WORK	89
7.1 Problems	90
7.2 Conclusion.....	92
7.3 Future Work.....	93
REFERENCES	94
APPENDICES	95

LIST OF TABLES

TABLE 1-1: ESTIMATED COST.....	4
TABLE 5- 1: PACKET – READ REQUEST.....	55
TABLE 5- 2: ICP PACKET – READ RESPONSE.....	55
TABLE 5- 3: ICP PACKET – WRITE REQUEST.....	56
TABLE 5- 4: ICP PACKET – WRITE RESPONSE.....	57
TABLE 6-1: TEMPERATURE SENSORS READINGS.....	79
FIGURE 2-5: [Faint]	19
FIGURE 2-6: TEMPERATURE SENSOR	22
FIGURE 2-7: [Faint]	24
FIGURE 2-8: [Faint]	25
FIGURE 2-9: [Faint]	26
FIGURE 2-10: [Faint]	30
FIGURE 2-11: [Faint]	31
FIGURE 2-12: [Faint]	31
FIGURE 2-13: [Faint]	32
FIGURE 2-14: [Faint]	33
FIGURE 2-15: [Faint]	34
FIGURE 2-16: [Faint]	39
FIGURE 2-17: [Faint]	40
FIGURE 2-18: [Faint]	41
FIGURE 2-19: [Faint]	42
FIGURE 2-20: [Faint]	43
FIGURE 2-21: [Faint]	44
FIGURE 2-22: [Faint]	45
FIGURE 2-23: [Faint]	46
FIGURE 2-24: [Faint]	47
FIGURE 2-25: [Faint]	48
FIGURE 2-26: [Faint]	49
FIGURE 2-27: [Faint]	50
FIGURE 2-28: [Faint]	51
FIGURE 2-29: [Faint]	52
FIGURE 2-30: [Faint]	53
FIGURE 2-31: [Faint]	54
FIGURE 2-32: [Faint]	55
FIGURE 2-33: [Faint]	56
FIGURE 2-34: [Faint]	57
FIGURE 2-35: [Faint]	79

LIST OF FIGURES

FIGURE 1- 1: THE FIRST TIME PLANNING.....	5
FIGURE 1- 2: THE SECOND TIME PLANNING.....	6
FIGURE 2-1: TCP/IP WITH OSI.....	14
FIGURE 2-2: TCP/IP PROTOCOL STACK.....	14
FIGURE 2-3: SMART SENSOR SYSTEM.....	16
FIGURE 2-4: PROTOTYPE SMART SENSOR NODE.....	17
FIGURE 2-5: IP μ 8930.....	19
FIGURE 2-6: TEMPERATURE SENSOR.....	22
FIGURE 2-7: LDR SENSOR.....	24
FIGURE2-8: SMOKE DETECTOR.....	25
FIGURE 2-9: 10 BASE T CROSS CABLE.....	26
FIGURE 3-1: PIC 18F97J60 FAMILY.....	30
FIGURE 3-2: GENERAL I/O PIN OF IP μ 8930.....	31
FIGURE 3-3: IP μ 8930 BLOCK DIAGRAM.....	31
FIGURE 3-4: SMART SENSOR SYSTEM BLOCK DIAGRAM.....	32
FIGURE 3-5: BLOCK DIAGRAM SMART SENSOR NODE.....	33
FIGURE 3-6: DETAILED BLOCK DIAGRAM OF SMART SENSOR NODE.....	34
FIGURE 4-1: I/O PINS OF MICRO WEB SERVER.....	39
FIGURE 4-2: LIGHT SENSOR CIRCUIT.....	40
FIGURE 4-3: TEMPERATURE SENSOR CIRCUIT.....	41
FIGURE 4-4: SMOKE SENSOR CIRCUIT.....	42
FIGURE 4-5: SCHEMATIC OF THE SYSTEM.....	43
FIGURE 4-6: SYSTEM LOGIN.....	44
FIGURE 4-7: NODE SELECTION.....	45
FIGURE 4-8: SENSORS VALUES.....	46
FIGURE 4-9: TIMER READING.....	47
FIGURE 4-10: SYSTEM CONFIGURATION.....	48
FIGURE 4-11: SYSTEM HELP.....	49
FIGURE 5-1: IPSANDLOCS FILE FORMAT.....	60
FIGURE 5-2: PLOT OF TEMPERATURE SENSOR READINGS.....	61
FIGURE 5-3: PART OF THE LOG FILE.....	63
FIGURE 5-4: LOGIN FLOW CHART (CHECK PASSWORD).....	64
FIGURE 5-5: PING PROCESS FLOW CHART.....	65
FIGURE 5-6: SEARCH NODE FLOWCHART.....	66
FIGURE 5-7: INPUT DATA FLOW CHART.....	67
FIGURE 6-1: TEMPERATURE SENSOR CIRCUIT.....	79

FIGURE 6-2: PLOT OF TEMPERATURE SENSOR READINGS.....	80
FIGURE 6-3: EXAMPLE OF TEMPERATURE READING EQUATION.....	81
FIGURE 6-4 ANALOG-TO-DIGITAL (A/D) CONVERTER MODULES.....	82
FIGURE 6-5 INITIAL LIGHT SENSOR CIRCUIT.....	82
FIGURE 6-6 LDR SENSOR CIRCUIT (AT DAY).....	83
FIGURE 6-7 LDR SENSOR CIRCUIT (AT DARK).....	83
FIGURE 6-8: SMOKE DETECTOR (CLEAR ENVIRONMENT).....	84
FIGURE 6-9: SMOKE DETECTOR (SMOKE EXISTENCE).....	84
FIGURE 6-10: OUTPUT MODE.....	85
FIGURE 6-11: INPUT MODE.....	87
FIGURE 6-12: ADDRESSES OF MICRO WEB SERVER S' PINS.....	88

2.1 Chapter idea about the project and its importance.

2.2 Literature Review

2.3 Research Gap

2.4 Thesis Statement

2.5 Data Management

2.6 Report Structure

CHAPTER ONE

CHAPTER ONE

INTRODUCTION

INTRODUCTION

1.1 General idea about the project and its importance.

A smart sensor is simply one that acquires physical, biological or chemical input, converts the measured value into a digital format in the units of the measured quantity, processes, and transmits that measured information via the Ethernet to a

1.1 General idea about the project and its importance.

1.2 Literature Review

1.3 Estimated Cost

1.4 Time Schedule

1.5 Risk Management

1.6 Report Contents

transmitting the processed information ready to use via the Ethernet to the system monitoring point.

The design construction of our system will search for multiple nodes connecting to the network and get their IP addresses and locations. One node will be implemented in order to collect data from sensors of temperature, smoke, and light. Smart signals will be converted from analog to digital by a microprocessor which has an internal Analog to Digital Converter (ADC), then connect it to a network via a hub and switch. Therefore any one who uses any computer on that network can display and store sensors' view in a suitable format.

CHAPTER ONE

INTRODUCTION

1.1 General idea about the project and its importance

A smart sensor is simply one that acquires physical, biological or chemical input, converts the measured value into a digital format in the units of the measured attribute, process, and transmits that measured information via the Ethernet to a computer monitoring point.

A smart sensor does many things a sensor system was required to do to process a measurement and provide intelligent information. In addition, the smart sensor takes care of transmitting the processed information ready to use via the Ethernet to the system monitoring point.

The design construction of our system will search for multiple nodes connecting to the network and get their IP addresses and locations. One node will be implemented in order to collect data from sensors of temperature, smoke, and light. Sensors' signals will be converted from analog to digital by a microprocessor which has an internal Analog to Digital Converter (ADC), then connect it to a network via a hub and switch. Therefore any one who uses any computer on that network can display and store sensors' value in a suitable format.

The networking smart sensor enable high quality detection /measurements networks with low cost and easy deployment, and it provides new monitoring and control capability for wide range of applications industrial process monitoring, health care, safety and security.

By processing sensor data locally at the nodes, the smart sensor system may be capable of data manipulations too computationally intensive for a single computer to handle. So as the number of sensors in the system increases, the advantages of a smart sensor become more pronounced.

1.3 Estimated Cost

1.2 Literature Review

There are several studies and projects working on smart sensors:

1. Smart sensing using custom photo-application-specific integrated-circuits ^[11].

Monitor and collect sensor data during the curing process of a general material system. Data obtained from sensors are used to generate an expert processing knowledge base which automatically controls the composite cure state.

2. Acquisition and Analysis of Fire Sensor Signals in Selected Environments'.^[12]

Fire sensor signals: The integration of microprocessor, memory and modem at the installation site makes it possible to transmit signals from temperature and aerosol sensors to a security control center via the public telephone network. Reducing data volume. Only the relevant information concerning status changes in an environment are stored.

1.3 Estimated Cost

In this project we need the following equipments in order to implement the system.

Table 1-1: Estimated Cost

Component	Cost
Micro Web Server	\$200
Sensors	\$20
10 Base T Cable	\$5
Total Cost	\$225

1.4 Time Schedule

The time planning includes two time estimation schedules; the first one show what is done in the first semester and the second shows the expected scheduling time of the second semester.

1.4.1 The First Time Planning:

Tasks \ Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Data gathering analysis	■	■	■	■	■											
Study sensor					■	■	■	■								
Study PIC 18F67J60 Microcontroller							■	■	■	■						
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

Figure 1- 1: The First Time Planning.

1.4.2 The Second Time Planning

Weeks	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Task																
Design circuit	█	█	█													
Implement and test the sensors circuits			█	█	█	█										
Study Micro Web Server and NetBeans							█	█	█							
SW implementation and tested										█	█	█	█	█	█	
Test the project														█	█	█
Operate the project																█
Documentation				█	█	█	█	█	█	█	█	█	█	█	█	█

Figure 1- 2: The Second Time Planning.

1.5 Risk management

Many types of risks are taken into account in order to be ready to take recovery steps immediately as soon as they appear.

1.5.1 Hardware Risk

The most important hardware part in our project is the microcontroller. The predicted risks are:

- Device failure: the microcontroller may crash because of high voltage supply or other problems.
- The device operates differently than intended.
- Sensors don't react correctly due to a specific response that may result for many causes.

1.5.2 Group risk

- Illness of one of group members.
- Group meeting difficulties

1.5.3 Requirement risks

- Several requirements may be affected by other requirements due to the high cost of some components.
- Team fails to understand the impact of requirements changes.

1.4 Recovery:

- Cover budget of extra cost by ignoring unexpected requirements as possible.

1.5.4 Project risks:

- Some requirements and changes may arise lately.
- Schedule not accurate
- Budget not sufficient.
- Latency of devices arrival.

Recovery:

- Demand device at earlier time.
- Start working on the implementation earlier.
- Use alternate devices with the same functionality and less cost.
- Try to build our own circuits that can replace unavailable devices.

1.6 Report Contents

The documentation for this project is categorized into four chapters. Each chapter is concerned with a logical or physical part of the system. The followings summarize briefly what each chapter will explain:

- **Chapter One: Introduction**

This chapter demonstrates a general idea about the project and its importance, literature review, estimated cost, time planning, and project risk management.

- **Chapter Two: Theoretical Background**

This chapter focuses on theories and materials that are related to our system operation and behavior.

- **Chapter Three: Project Conceptual Design**

This chapter describes the system in its abstract formula. It describes the project objectives, design options and justifies those that are chosen in the project, a general block diagram, how does System work.

- **Chapter Four: Detailed Technical Project Design**

This chapter discusses a detailed description about project phases, subsystem design, overall system design and user system interface.

- **Chapter Five: Software System Design**

This chapter handles the software related to our system, depicts flow charts about system operation and illustrates different algorithms and techniques that will be considered in writing the software.

- **Chapter Six: System Implementation and Testing**

This chapter will manifest the implementation procedures to be acted so as to integrate the project. Then, a sequence of procedural testing will be listed. The testing comprises both software and hardware testing.

- **Chapter Seven: Conclusions and Future Work**

This chapter will list the problems facing us in accomplishing the system and how they were resolved. Conclusions and future work are also proposed.

CHAPTER TWO

CHAPTER TWO

THEORETICAL BACKGROUND

THEORETICAL BACKGROUND

2.1 Project Components

2.2 Software Theoretical Background

2.3 Hardware Theoretical Background

The components of this project include several parts, each of them is related to the other to form the whole system. These components are:

1. 486/500 Micro Web Server
2. Temperature, Light sensors and Smoke detector.
3. 10 Base T Cross Cable
4. Computer Network.
5. Software.

CHAPTER TWO

THEORETICAL BACKGROUND

This chapter focuses on theories and materials that are related to our system operation and behavior.

2.1 Project Components

The components of this project include several parts; each of them is related to the other to form the whole system. These components are:

1. IP μ 8930 Micro Web Server
2. Temperature, Light sensors and Smoke detector.
3. 10 Base T Cross Cable
4. Computer Network.
5. Software.

CHAPTER TWO

THEORETICAL BACKGROUND

This chapter focuses on theories and materials that are related to our system operation and behavior.

2.1 Project Components

The components of this project include several parts, each of them is related to the other to form the whole system. These components are:

1. IPμ8930 Micro Web Server
2. Temperature, Light sensors and Smoke detector.
3. 10 Base T Cross Cable
4. Computer Network.
5. Software.

The layers most closely affected by TCP/IP are Layer 7 (application), Layer 4 (transport), and Layer 3 (network). TCP/IP enables communication among any set of interconnected networks and is equally well suited for both LAN (Local Area Network) and WAN (Wide Area Network) communication. ^[1]

2.2 Software Theoretical Background

2.2.1 TCP/IP

The TCP/IP suite of protocols was developed as part of the research done by the Defense Advanced Research Projects Agency (DARPA). TCP/IP is the standard for internet work communications and serves as the transport protocol for the Internet, enabling millions of computers to communicate globally.

This Theoretical Background focuses on TCP/IP for several reasons:

- TCP/IP is a universally available protocol.
- TCP/IP is a useful reference for understanding other protocols because it includes elements that are representative of other protocols.
- TCP/IP is important because the router uses it as a configuration tool.

The function of the TCP/IP protocol stack, or suite, is the transfer of information from one network device to another. In doing so, it closely maps the OSI reference model in the lower layers, and supports all standard physical and data link protocols.

The layers most closely affected by TCP/IP are Layer 7 (application), Layer 4 (transport), and Layer 3 (network). TCP/IP enables communication among any set of interconnected networks and is equally well suited for both LAN (Local Area Network) and WAN (Wide Area Network) communication. ^[1]

2.2 Software Theoretical Background

2.2.1 TCP/IP

The TCP/IP suite of protocols was developed as part of the research done by the Defense Advanced Research Projects Agency (DARPA). TCP/IP is the standard for internet work communications and serves as the transport protocol for the Internet, enabling millions of computers to communicate globally.

This Theoretical Background focuses on TCP/IP for several reasons:

- TCP/IP is a universally available protocol.
- TCP/IP is a useful reference for understanding other protocols because it includes elements that are representative of other protocols.
- TCP/IP is important because the router uses it as a configuration tool.

The function of the TCP/IP protocol stack, or suite, is the transfer of information from one network device to another. In doing so, it closely maps the OSI reference model in the lower layers, and supports all standard physical and data link protocols.

The layers most closely affected by TCP/IP are Layer 7 (application), Layer 4 (transport), and Layer 3 (network). TCP/IP enables communication among any set of interconnected networks and is equally well suited for both LAN (Local Area Network) and WAN (Wide Area Network) communication. ^[1]

2.3 Hardware Theoretical Background

2.3.1 Stack

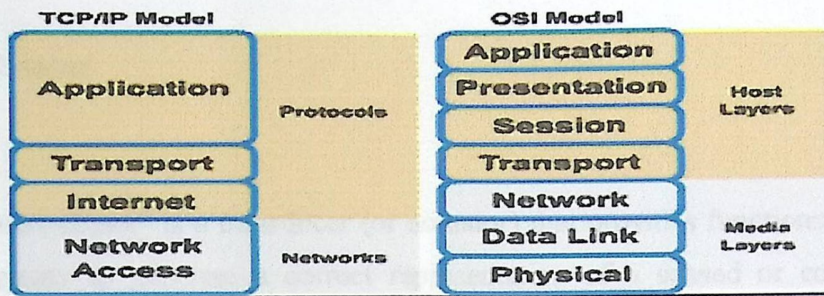


Figure 2- 1: TCP/IP With OSI

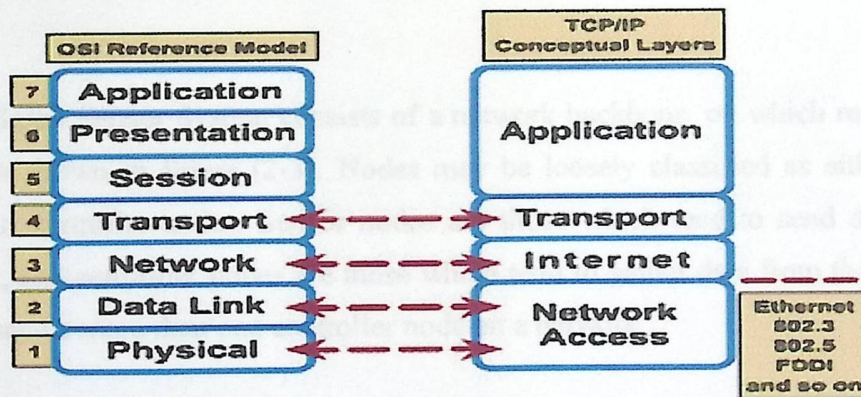


Figure 2- 2: TCP/IP Protocol Stack

2.3 Hardware Theoretical Background

2.3.1 Smart Sensor

A "smart sensor" is a transducer (or actuator) that provides functions beyond what is necessary to generate a correct representation of a sensed or controlled quantity (e.g., temperature, pressure, strain, flow, pH, etc.). The "smart sensor" functionality will typically simplify the integration of the transducer into applications in a networked environment. For example, a measurement from a temperature transducer requires the network controller to make a voltage-to-temperature conversion to represent the data in either degrees Fahrenheit or degrees Celsius. [3]

Smart Sensor System consists of a network backbone, on which reside many nodes as shown in figure (2-3). Nodes may be loosely classified as either sensor nodes or controller nodes. Sensor nodes are those which tend to send data to the network, and controller nodes are those which tend to gather data from the network. There may be more than one controller node on a network.

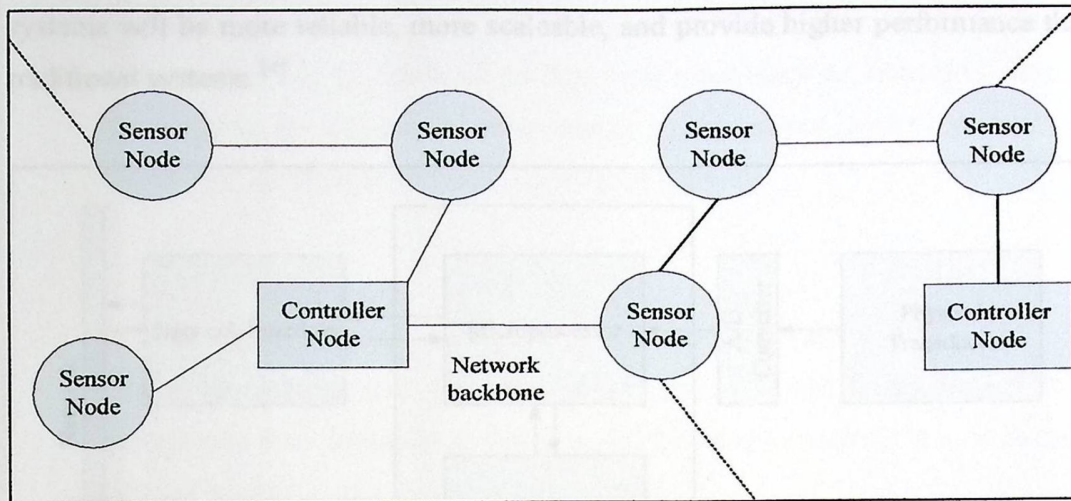


Figure 2-3: Smart Sensor System

A prototypical sensor node consists of three elements: a physical transducer, a network interface, and a processor/memory core. The transducer senses the physical quantity being measured and converts it into an electrical signal. Then the signal is fed to an A/D converter, and is now ready for use by the processor. The processor will perform some signal processing on the data, and depending on how it is programmed.

A prototypical controller node consists of processor/memory, a network interface, and input/output devices for communicating with human users. It is used to collect information from the sensor nodes, to program the sensor nodes, and to provide feed back to the user as shown on Figure (2-4).

This type of "smart" sensor will revolutionize the design of sensor systems. It will become easier, cheaper, and faster to design a sensor system and the resulting

systems will be more reliable, more scaleable, and provide higher performance than traditional systems. [4]

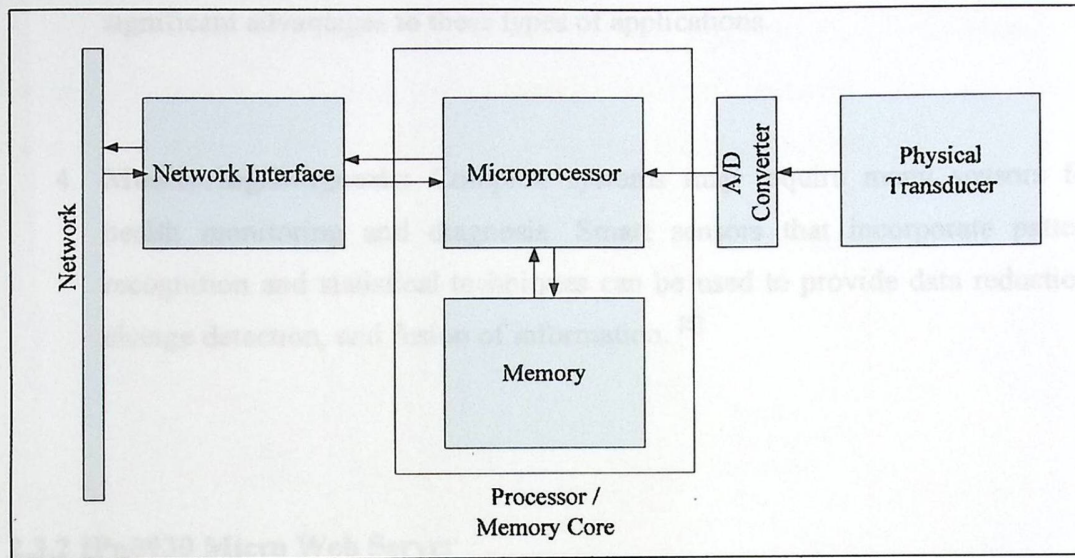


Figure 2-4: Prototype Smart Sensor node.

The role of smart sensors in applications can be grouped into five categories:

1. **Signal Conditioning:** The smart sensor serves to convert from a time-dependent, analog variable to a bus output. Functions such as linearization, temperature compensation, and signal processing may be included in the package.
2. **Data Reduction/Abstraction:** The smart sensor provides higher level information processing. The primary objective is to convert from the raw measurements to more abstract symbolic descriptions. Sensors that output "features", detections, or classifications fall in this category.

3. **Tightening Feedback Loops:** Communication delays can cause problems for systems that rely on feedback or that must react/adapt to their environment. By reducing the distance between sensor and processor, smart sensors bring significant advantages to these types of applications.

4. **Monitoring/Diagnosis:** Complex systems may require many sensors for health monitoring and diagnosis. Smart sensors that incorporate pattern recognition and statistical techniques can be used to provide data reduction, change detection, and fusion of information. [5]

2.3.2 IP μ 8930 Micro Web Server

2.3.2.1 General Description

The IP μ 8930 is a general purpose network controller and web server which makes it easy to monitor, control and communicate with remote sensors, actuators, and practically any devices with a serial port (via the onboard serial port) via a TCP/IP network such as the Internet.

At the heart of the IP μ 8930 Developer Kit is the IP μ 8930 Module, a compact TCP/IP network controller and web server module, which enables developers to rapidly add network connectivity to products. The IP μ 8930 combines a TCP/IP controller, HTTP compliant web server, Modbus TCP node and A/D converter into a single, small (1.3x1.4"/3.3x3.4cm) daughterboard. The IP μ 8930 is designed to enable remote monitoring and control over a TCP/IP-based network without the

overhead and complexity of traditional solutions which require the knowledge on how to support and program using a real-time operating systems.

The IP μ 8930 Module plugs easily into the IP μ 8930 Developer Board. The Developer Board is packaged with power, a 10BaseT network jack, and various connectors to make it easy to set up, program, and test the IP μ 8930. [13]

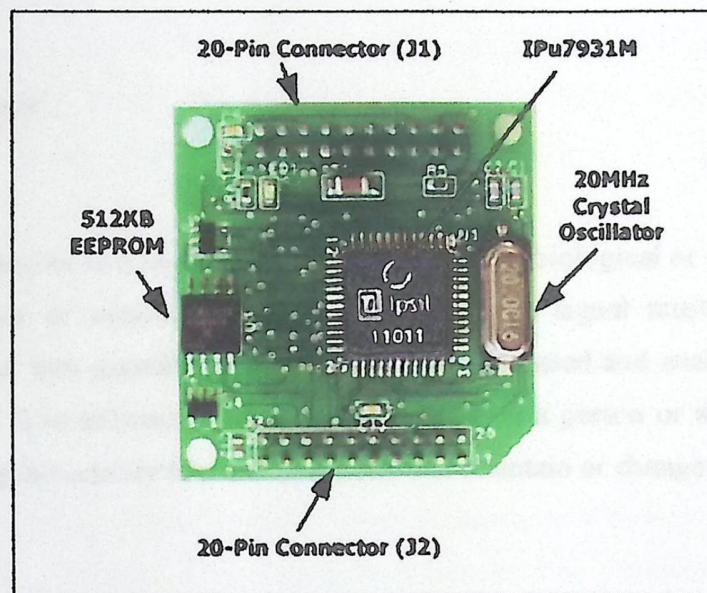


Figure 2-5: IP μ 8930

2.3.2.2 IP μ 8930 Features

The IP μ 8930 Board provides the following features:

- Supports both TCP/IP—read & write
- Up to 8 analog or digital ports for monitoring/control of external devices
- Built-in 10-bit ADC

- Ability to define WebHoles™ in HTML pages which get “filled” with values from pre-defined I/O port.
- Supports the following network applications:
 - ARP, DHCP, ICMP (ping), Modbus, and Ipsil Control Protocol (v1.0)

The data sheet for the Micro Web Server can be found in the appendix A

2.3.3 Sensors

A sensor is a device that converts physical, biological or chemical input into an electrical or optical signal. To be useful, the signal must be measured and transformed into digital format which can be processed and analyzed efficiently by computers. The information can be used by either a person or an intelligent device monitoring the activity to make decisions that maintain or change a course of action.

A sensor is a type of transducer. Direct indicating sensors, for example, a mercury thermometer, are human readable. Other sensors must be paired with an indicator or display, for instance a thermocouple.

Sensors are used in everyday life. Applications include automobiles, machines, aerospace, medicine, industry and robotics.

2.3.3.1 Temperature Sensor (LM35)

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of $\pm 1/4^\circ\text{C}$ at room temperature and $\pm 3/4^\circ\text{C}$ over a full -55 to $+150^\circ\text{C}$ temperature range. Low cost is assured by trimming and calibration at the wafer level.

The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy.

It can be used with single power supplies, or with plus and minus supplies. As it draws only $60\ \mu\text{A}$ from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to $+150^\circ\text{C}$ temperature range. [6]

Features:

1. Calibrated directly in ° Celsius (Centigrade)
2. Linear + 10.0 mV/°C scale factor
3. 0.5°C accuracy guarantee able (at +25°C)
4. Rated for full -55° to $+150^\circ\text{C}$ range
5. Suitable for remote applications

6. Low cost due to wafer-level trimming
7. Operates from 4 to 30 volts
8. Less than 60 μA current drain
9. Low self-heating, 0.08 $^{\circ}\text{C}$ in still air
10. Nonlinearity only $\pm 1/4^{\circ}\text{C}$ typical
11. Low impedance output, 0.1 W for 1 mA load

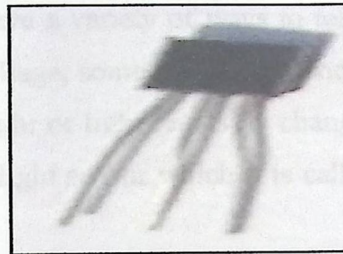


Figure2-6: Temperature Sensor

2.3.3.2 Light Sensor

Light sensors are one of the most common types of sensors. They are used in night lights, street lights, alarms, toys, cameras, etc

The light sensor is used to detect changes in the amount of light in its surroundings. The resistance of the light sensor decreases as the intensity of light falling on the sensor increases.

Light sensors have many different functions, and they come in different shapes, sizes, and with different price tags. Some sensors are designed to sense a particular color of light, such as blue, green, red, or infrared. Some sensors don't care what color the light is because they react to how bright the light is. Other sensors look for only special kinds of light given off by certain chemical reactions.

Light sensors also have a variety of ways to tell a microcontroller what they see. Some sensors send a voltage, some send a sequence of binary values, and others react to different kinds of light or light levels by changing resistance. In our project we want to use this kind of light sensor which is called: Light Dependant Resistor (LDR).

An LDR will have a resistance that varies according to the amount of visible light that falls on it. Against but it be the intensity of light that impact in the surface of the LDR smaller will be its resistance and against less light impact greater will be the resistance.

The light falling on the brown zigzag lines on the sensor causes the resistance of the device to fall. This is known as a negative co-efficient. There are some LDRs that work in the opposite way i.e. their resistance increases with light (called positive co-efficient).^[7]

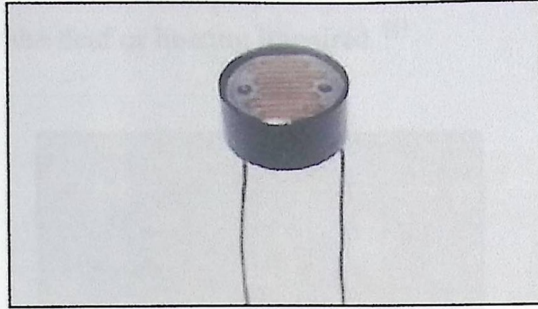


Figure2-7: LDR Sensor.

2.3.3.3 Smoke Detector

Smoke sensor is a device that detects smoke and issues an alarm to alert nearby people that there is a potential fire. A household smoke detector will typically be mounted in a disk shaped plastic enclosure about 150mm in diameter and 25mm thick, but the shape can vary by manufacturer.

Because smoke rises, most detectors are mounted on the ceiling or on a wall near the ceiling. To avoid the nuisance of false alarms, most smoke detectors are mounted away from kitchens. To increase the chances of waking sleeping occupants, most homes have at least one smoke detector near any bedrooms; ideally in a hallway as well as in the bedroom itself.

Smoke detectors are usually powered by one or more batteries but some can be connected directly to household wiring. Smoke detectors may operate alone, be interconnected to cause all detectors in an area to sound an alarm if one is triggered,

or be integrated into a fire alarm or security system. Smoke detectors with flashing lights are available for the deaf or hearing impaired. [8]



Figure2-8: Smoke Detector.

2.3.4 10 Base T crossover cable

10 Base T cable is one of several adaptations of the Ethernet (IEEE 802.3) standard for Local Area Networks (LANs). 10BASE-T is an implementation of Ethernet which allows stations to be attached via twisted pair cable. The name 10BASE-T is derived from several aspects of the physical medium. The *10* refers to the transmission speed of 10 Megabits per second (Mb/s). The *BASE* is short for baseband. This means only one Ethernet signal is present on the send and/or receive pair. 10 Base T uses RJ-45 Jacks.

The 10BASE-T Ethernet standards use one wire pair for transmission in each direction. This requires that the transmit pair of each device be connected to the

receive pair of the device on the other end. When a terminal device is connected to a switch or hub, this crossover is done internally in the latter.

One terminal device may be connected directly to another without the use of a switch or hub, but in that case the crossover must be done externally in the cable. Since 10BASE-T use pairs 2 and 3, these two pairs must be swapped in the cable. This is a crossover cable (which we use for testing process). A crossover cable must also be used to connect two internally crossed devices (e.g., two hubs) as the internal crossovers cancel each other out. ^[9]

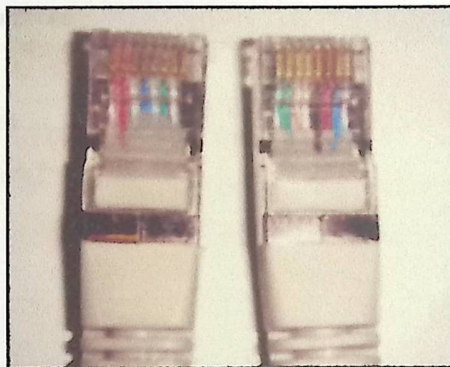


Figure 2-9:10 Base T cross cable

CHAPTER THREE

CHAPTER THREE

PROJECT CONCEPTUAL DESIGN

This chapter describes project objectives, design options and justifies those that are chosen in the project. It also explains how the system works.

PROJECT CONCEPTUAL DESIGN

3.1 Detailed project objectives

3.1 Detailed Project Objectives

3.2 Design options

3.3 General Block Diagram

3.4 How Does System Work?

1. Implement and test one node that read data from 3 different sensors (Temperature, Smoke, and light sensor) and connect it to the Micro Web Server after building the necessary circuits.
2. Connect this node with a hub through 10 base cable, to send data over the network, so that any person can get sensors' value from any computer in the system.

CHAPTER THREE

PROJECT CONCEPTUAL DESIGN

This chapter describes project objectives, design options and justifies those that are chosen in the project, a general block diagram, and how does system work.

3.1 Detailed project objectives

1. Design Smart Sensor System that consists of multiple nodes (as shown on figure 2-3) each one consists of sensor node (controller, sensors, Ethernet interface), and controller node (computers monitoring point, necessary software).
2. Implement and test one node that read data from 3 different sensors (Temperature, Smoke, and light sensor) and connect it to the Micro Web Server after building the necessary circuits.
3. Connect this node with a hub through 10 base cable, to send data over the network, so that any person can get sensors' value from any computer in the system.

4. Collect, display, analyze and store sensors ' value in a suitable format.

3.2 Smart Sensor design Options

In this system there are many designing options, we are checking these options to find which one is more reliable, available, and cheaper than others.

Option 1: Interface Microprocessor with Ethernet chip. This kind of interfacing required programming in low level language, so it will take a lot of time and effort.

Option 2: Use Peripheral Interface Controller (PIC Microcontroller) that contains an Ethernet chip. We choose this option for system construction, And After we search about some kind of PIC that contains an Ethernet chip; we find that PIC 18F67J60 have on-board Ethernet MAC/PHY interface with the following features:

1. 8K/4K bytes Ethernet buffer SRAM.
2. 39 (I/O)
3. 11 channel 10 bit A/D.
4. IEEE 802.3 compatible Ethernet Controller
5. I/O ports A, B, C, D, E, F, and G.
6. 128 K Program Memories (Bytes).
7. 65532 Program Memories (Instructions).

8. 3808 Data Memory (Bytes).
9. Ethernet communication (10 Base T).^[10]

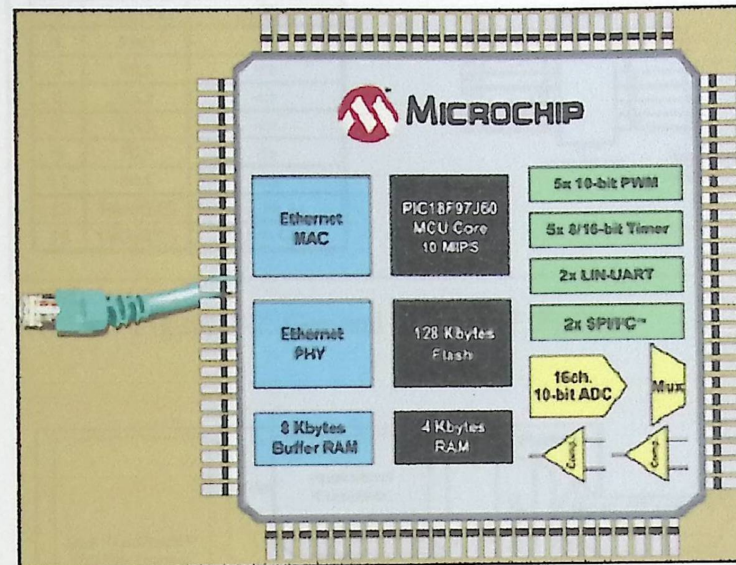


Figure 3-1: PIC 18F97J60 Family.

At the first semester we studied all the features of that PIC, and make a general image about the system based on it. But we faced a critical problem; that it was not available in the local market and also it was impossible to purchase it from other countries.

Option 3: Use the IP μ 8930 Micro Web Server. The IP μ 8930 contains a microcontroller and Ethernet interface that responsible for process, send and receive packet to or from the network. Also it has a general purpose IO pins (port A) which it is used to communicate with other system components. Each pin can be used for

input (set it to 1) or output (set it to 0), and can be configured as analog or digital. For these features we decided to use it instead of PIC microcontroller.

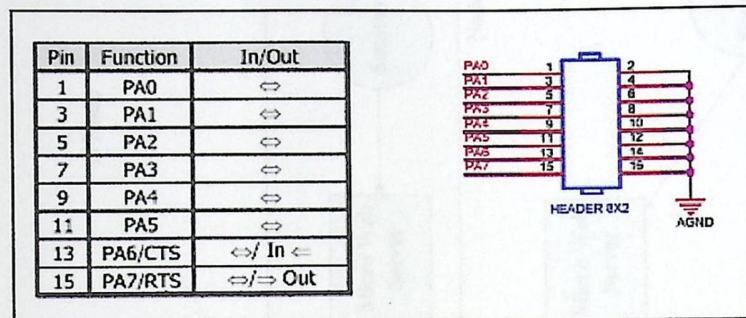


Figure 3-2: General I/O pin of IPμ8930

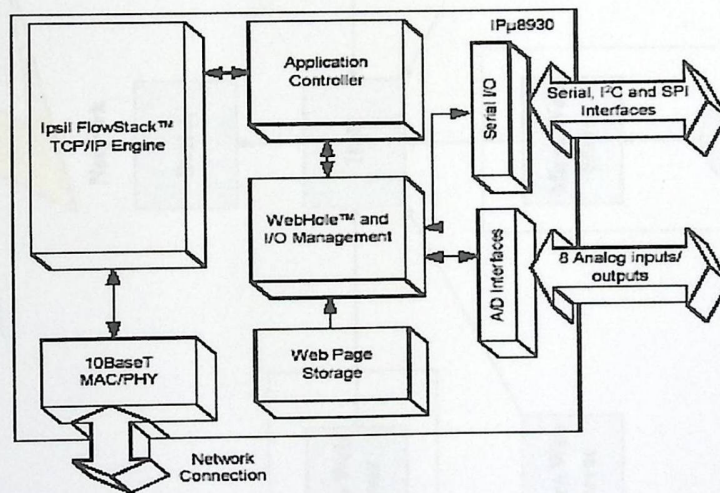


Figure 3-3: IPμ8930 Block Diagram.

Figure 3-4: Smart Sensor System block diagram

3.5 General Block Diagram

The following diagram defines general block diagram

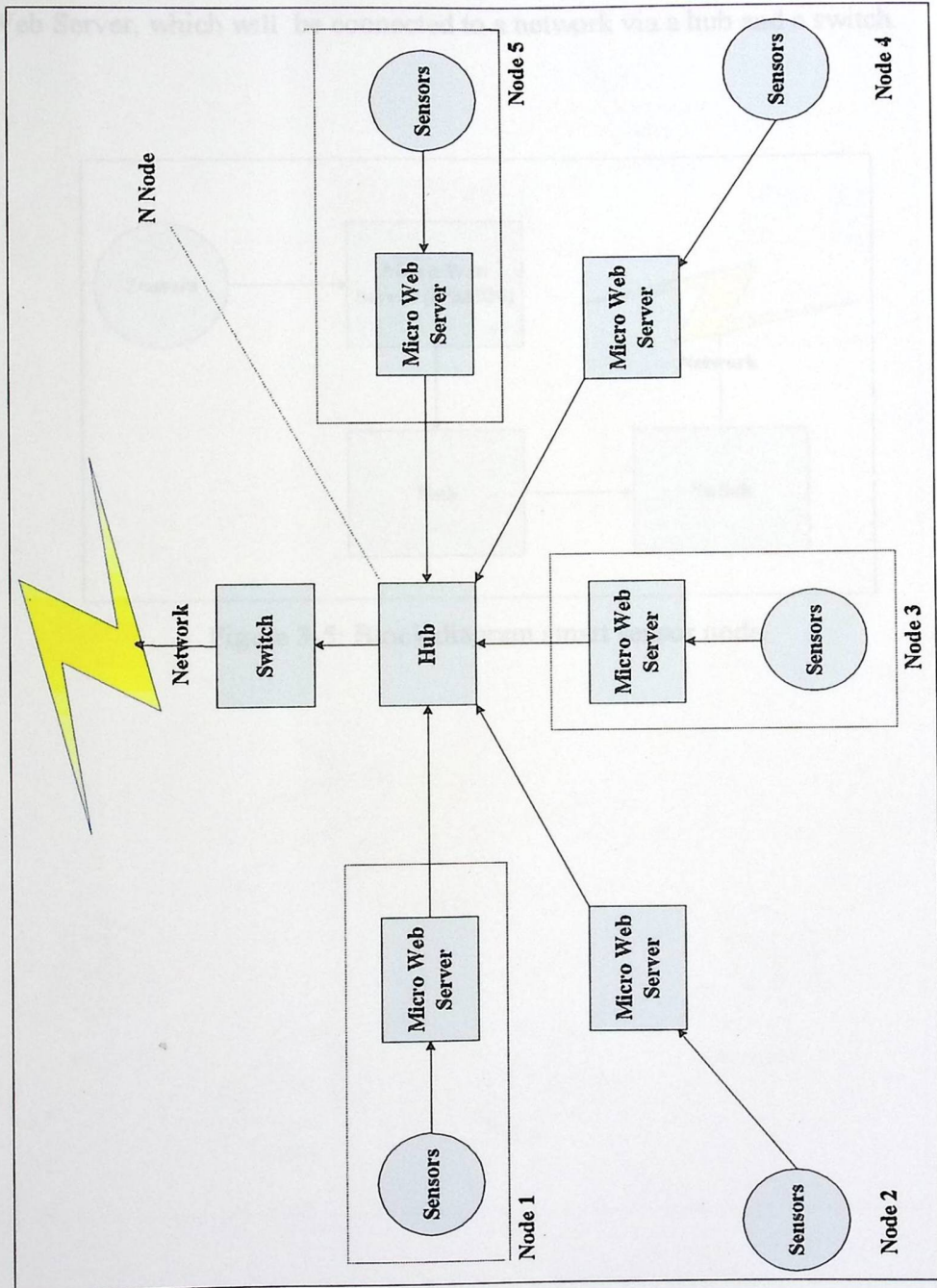


Figure 3-4: Smart Sensor System block diagram

Figures (3-5),(3-6) represents a block diagram and detailed block diagram of smart sensor node that has been implemented, as shown the sensors connected to the Micro Web Server, which will be connected to a network via a hub and a switch.

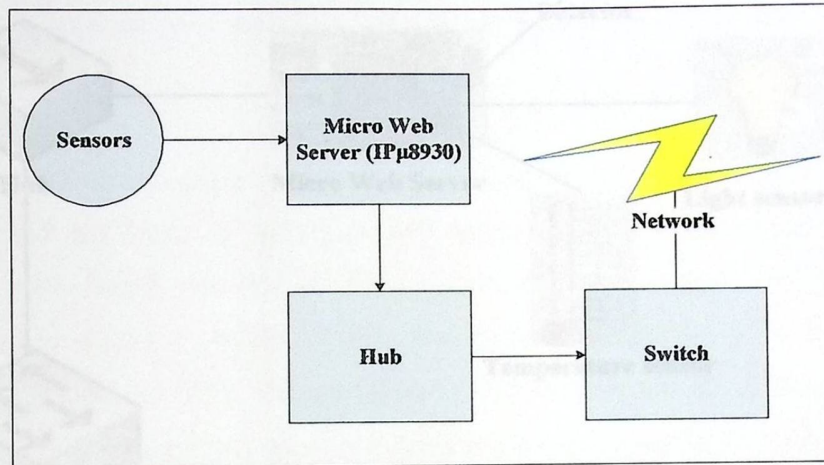


Figure 3-5: Block diagram smart sensor node.

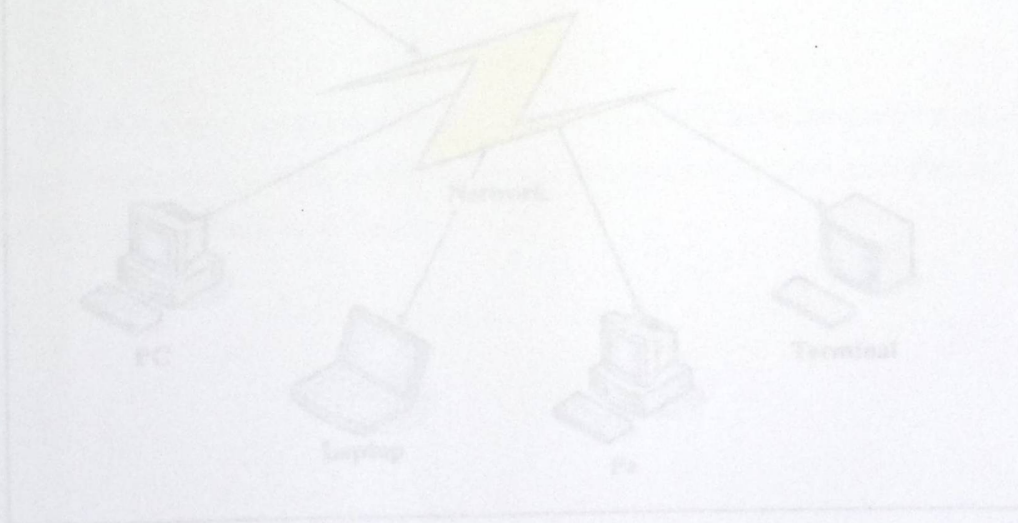


Figure 3-6: Detailed block diagram of smart sensor node.

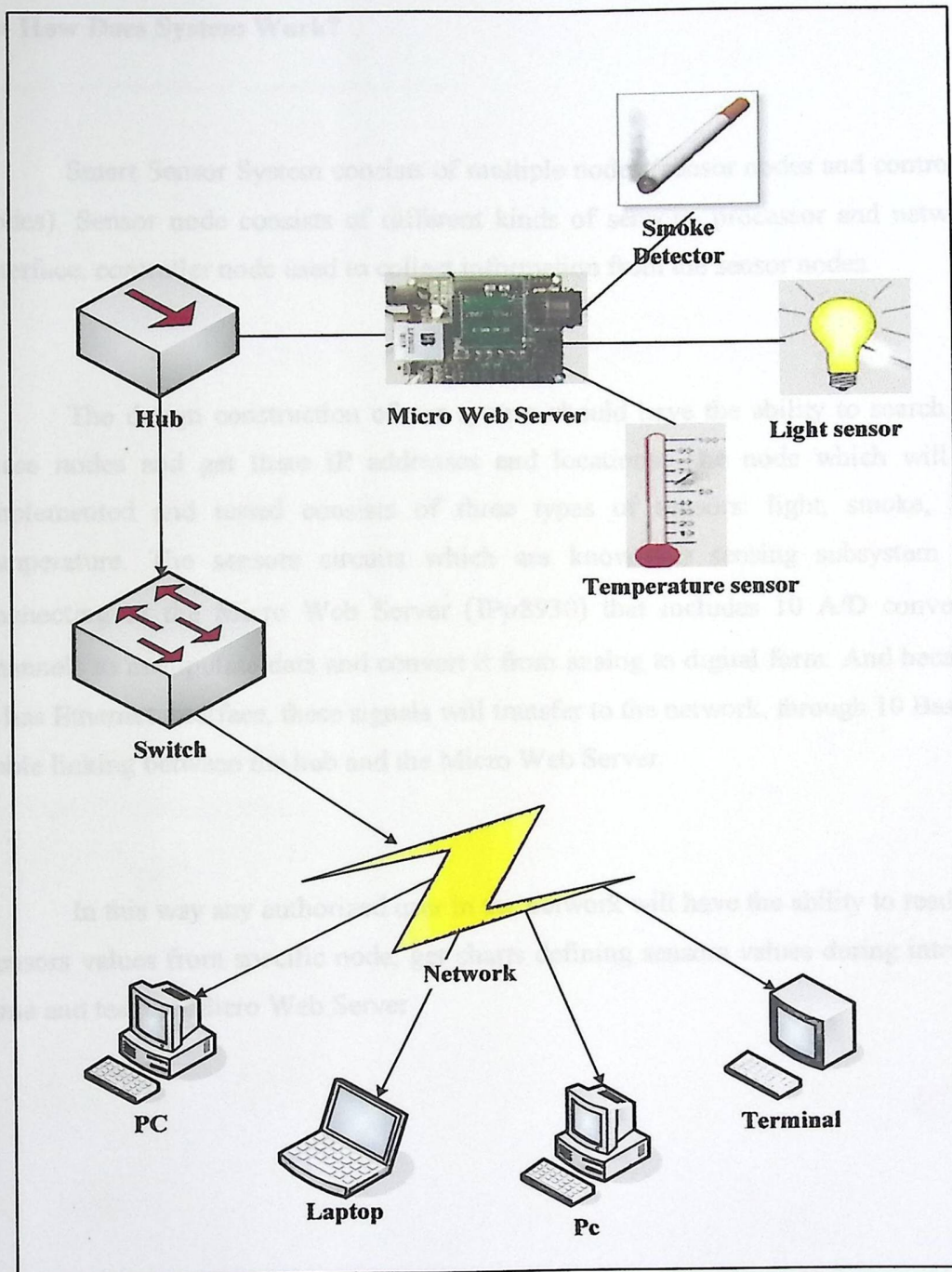


Figure 3-6: Detailed block diagram of smart sensor node.

3.4 How Does System Work?

Smart Sensor System consists of multiple nodes (sensor nodes and controller nodes). Sensor node consists of different kinds of sensors, processor and network interface, controller node used to collect information from the sensor nodes.

DETAILED TECHNICAL PROJECT DESIGN

The design construction of our system should have the ability to search for these nodes and get their IP addresses and locations. The node which will be implemented and tested consists of three types of sensors: light, smoke, and temperature. The sensors circuits which are known as sensing subsystem are connecting to the Micro Web Server (IP μ 8930) that includes 10 A/D converter channels to manipulate data and convert it from analog to digital form. And because it has Ethernet interface, these signals will transfer to the network, through 10 Base T cable linking between the hub and the Micro Web Server.

4.4 User-System Interface

In this way any authorized user in the network will have the ability to read the sensors values from specific node, get charts defining sensors values during interval time and test the Micro Web Server

CHAPTER FOUR

CHAPTER FOUR

DETAILED TECHNICAL PROJECT DESIGN

This chapter contains detailed description of the project phases, subsystems design, over

DETAILED TECHNICAL PROJECT DESIGN

4.1 Detailed description of the project phases

4.1 Detailed description of the project phases

4.2 Subsystem detailed design

4.3 Over all system design

4.4 User-System Interface

Each sensor in the system (temperature, light and smoke sensors) was connected in its own circuit and tested separately. After working on them they operated correctly and gave accurate values.

Phase II

For configuring the delay of each pin in the Micro Web Server, it was connected to a single PC by cross cable, so we were able to specify IO pins of PORT A, and test the read/write processes.

CHAPTER FOUR

DETAILED TECHNICAL PROJECT DESIGN

This chapter contains detailed description of the project phases, subsystems design, overall system design and user system interface.

4.1 Detailed description of the project phases

Phase 1:

Each sensor in the system (temperature, light and smoke sensors) was connected in its own circuit and tested separately. After working on them they operated correctly and gave accurate values.

Phase 2:

For configuring the duty of each pin in the Micro Web Server, it was connected to a single PC by cross cable, so we were able to specify I/O pins of PORT A, and test the read/write processes.

Phase 3:

In this phase the Micro Web Server was programmed to:

- Search for specific node on the network to pass this data to
- And read the output of the sensing circuits

Figure 4-1 I/O pins of Micro Web Server

Phase 4:

For data to be spread after being read from the sensing circuits, The Micro Web Sever was connected to a network via hub and switch, so every PC in this network is able to reach this data.

4.2 Subsystem Detailed Design

4.2.1 Micro Web Server (IP μ 8930)

Micro Web Server (IP μ 8930) contains 8 input/output pins, in our project we configured them as input pins, they are known as (PA0-PA7), each sensing circuit was connected to those pins as follow:

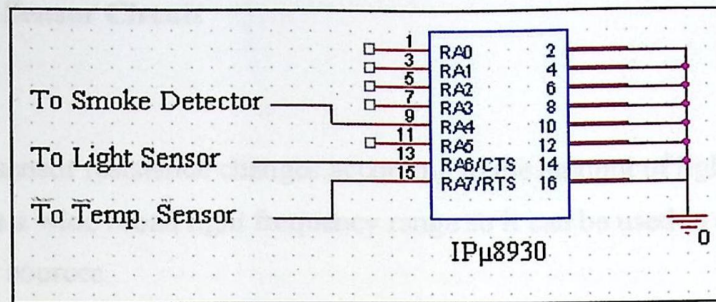


Figure 4-1: I/O pins of Micro Web Server

1. For LM35 temperature sensor circuit we took the output from pin 6 in the operational amplifier and connected it to pin 8 (PA7) in micro web server.
2. For LDR light sensor circuit, we took the output from pin 2 in 74H14 Schmitt trigger and connected it to pin 6 (PA5) in micro web server.
3. For smoke detector circuit the output was directly connected to pin 5 (PA4) in micro web server.

4.2.2 Sensing circuits

Sensing circuits are designed to realize the environment parameters; these parameters are measured in order to be analyzed by the Micro Web Server and displayed on a computer network.

4.2.2.1 Light Sensor Circuit

LDR sensor resistance changes according to the amount of light hitting its surface. It has a wide broad light frequency range so it can be used to sense a wide range of light sources.

Figure 4-2 shows the schematic circuit of light sensing, the LDR light sensor is manufactured to work on 5 volts power supply. In order to have a specific value of the output the 74HC14 Schmitt trigger logical gate was used. The sensing principle is to output 5 volts at night (light intensity is very low), and 0 volt at day (light intensity is high).

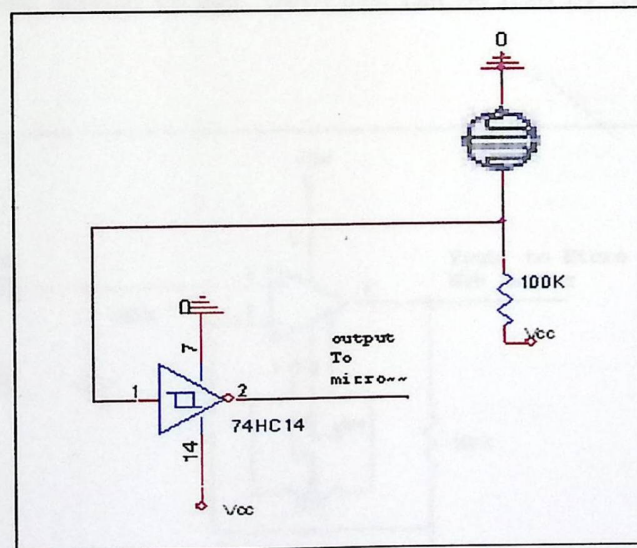


Figure 4-2: Light sensor circuit.

4.2.2.3 Smoke detector Circuit:

The smoke detector is one of the most popular alarm sensors used nowadays for fire alerting at kitchens, hotels, and hospitals. We choose this sensor because it is easy to use and deal with, and available on demand.

This detector consists of three legs, one is connected to the power supply (21V), the other is connected to the ground, and the third gives the output and is connected to the Micro Web Server. The principle on which this detector behavior works is to output (1.5V) when there is any smoke in the environment surrounding it, and output (21V) for clear environment, so we use voltage division principle with 2 resistances (3K Ω , 1K Ω), in order to out 5V. The schematic for smoke detector is shown at Figure (4-4).

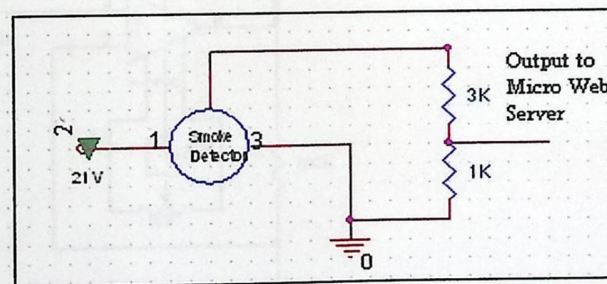


Figure 4-4: Smoke detector circuit.

4.3 Overall System Design:

4.4 User System Interface

Figure (4-5) show how is the hardware components are connected to each other.

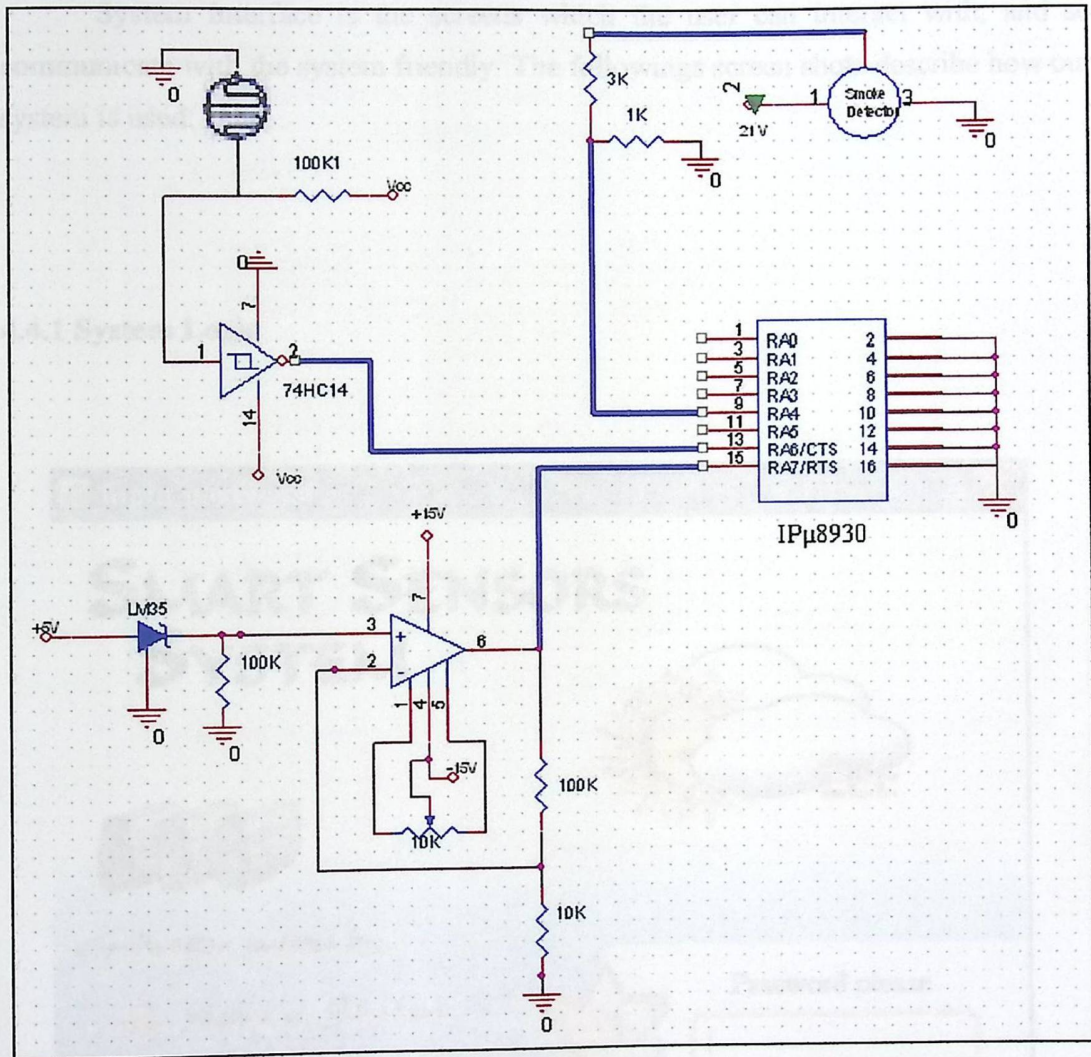


Figure 4-5: Schematic of smart sensor node.

Figure 4-6: System Logic

4.4 User System Interface

System Interface is the screens which the user can interact with, and so communicate with the system friendly. The followings screen shots describe how our system is used:

4.4.1 System Login

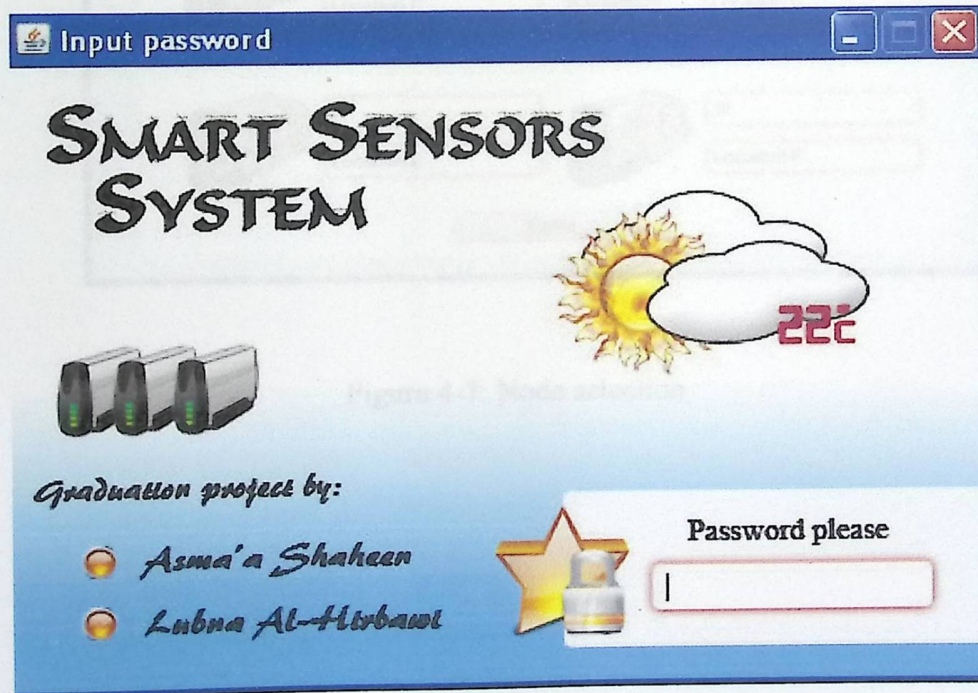


Figure 4-6: System Login

4.4.2 Node selection

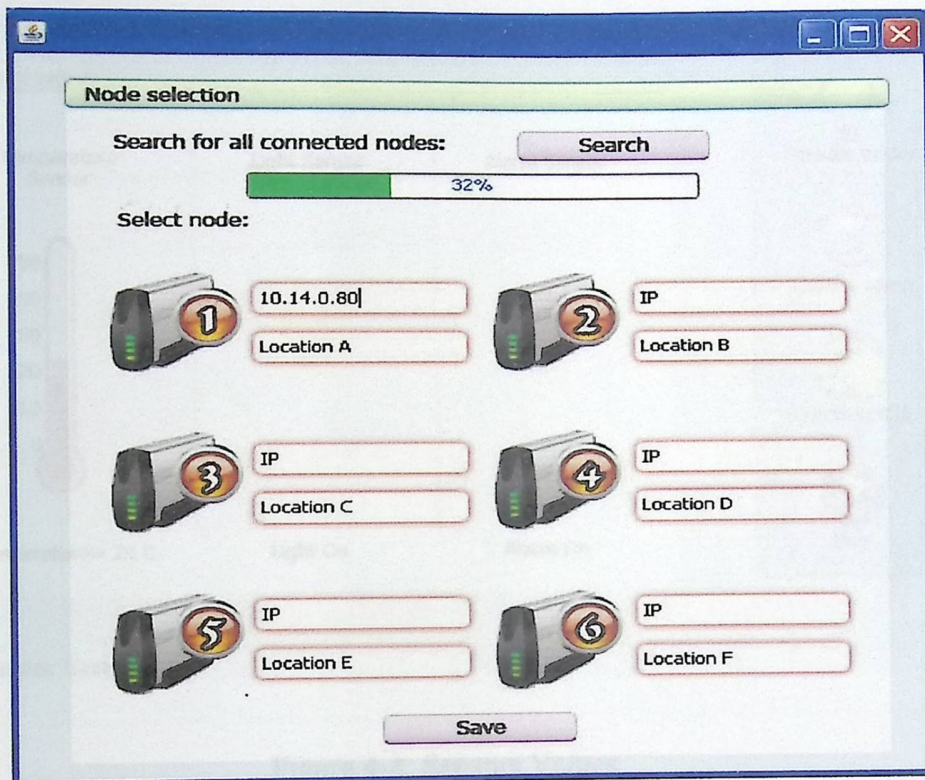


Figure 4-7: Node selection

4.4.3 Sensors Values

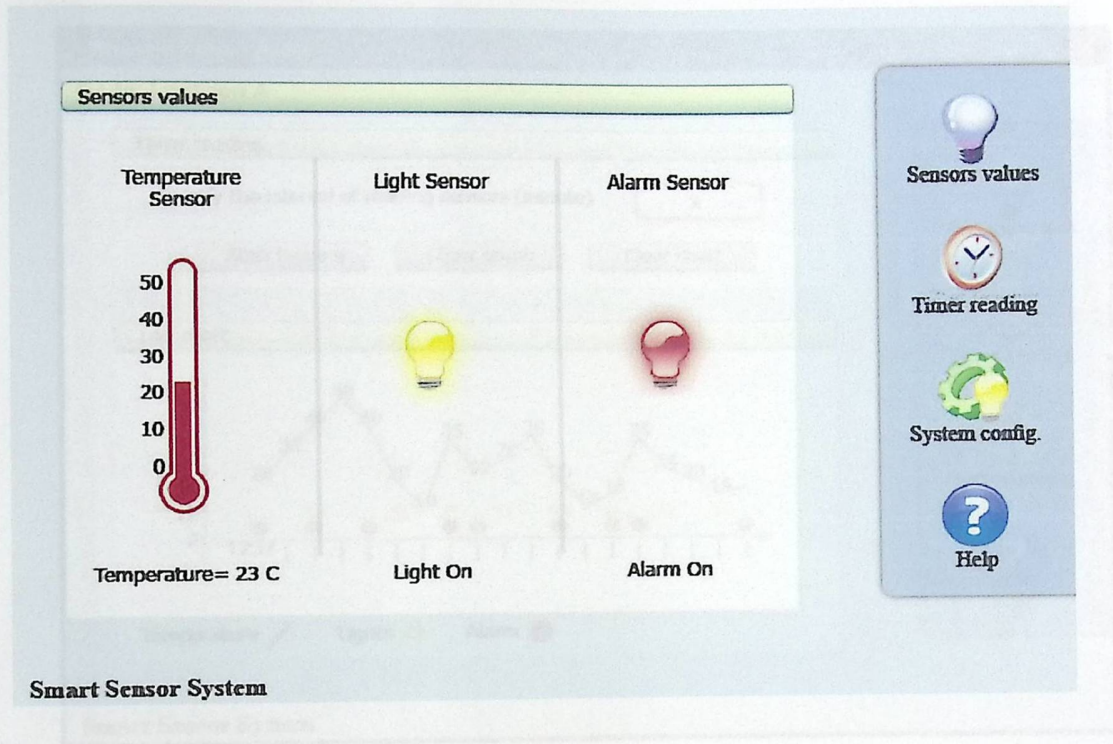


Figure 4-8: Sensors Values

4.4.4 Timer Reading

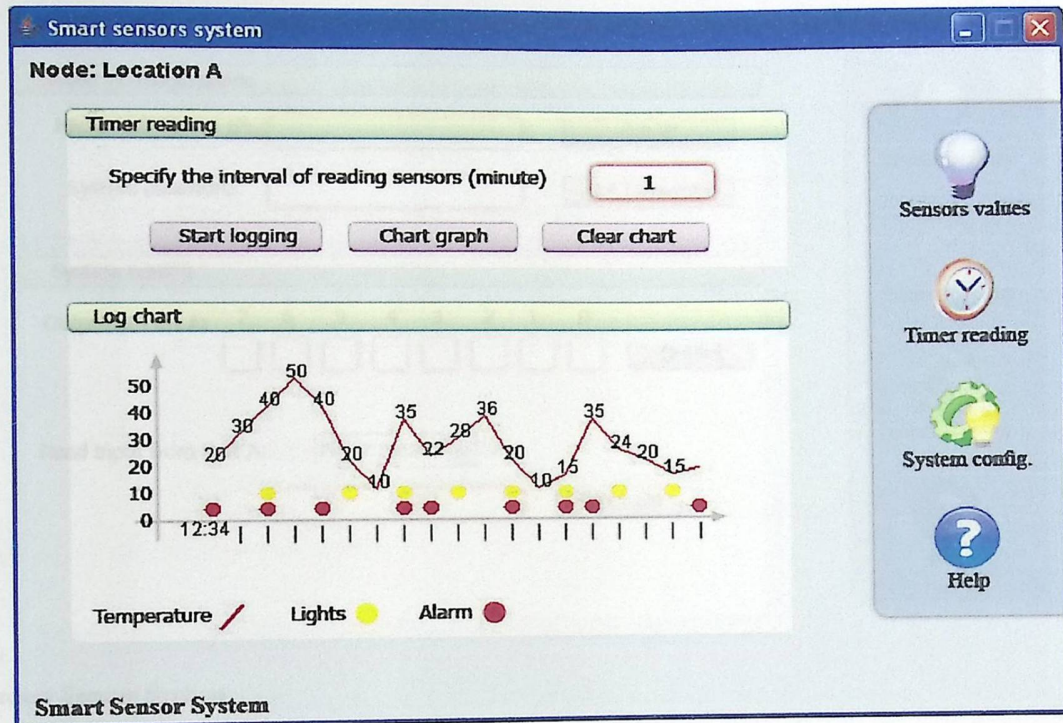
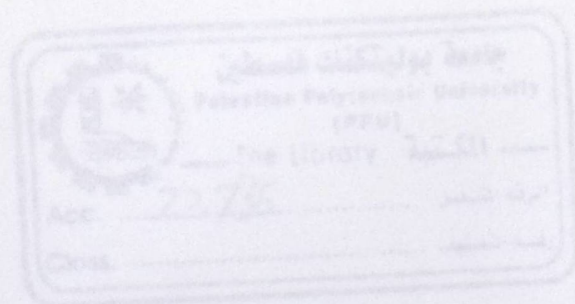


Figure 4-9: Timer Reading

4.4.5 System configuration


The screenshot displays the 'Smart Sensor System' web interface. It is divided into two main sections: 'System configurations' and 'System testing'.
System configurations: This section contains two rows of input fields and buttons. The first row has a text input for 'Micro web server IP:' followed by a 'Set IP' button. The second row has a text input for 'System password:' followed by a 'Set password' button.
System testing: This section includes an 'Output at Port A:' section with eight checkboxes labeled 7 through 0, and an 'Output' button. Below this is a 'Read input from Port A:' section with a dropdown menu currently set to 'Pin A7 0x128' and an 'Input pin' button.
Navigation sidebar: On the right side, there is a vertical sidebar with four icons and labels: a lightbulb for 'Sensors values', a clock for 'Timer reading', a gear for 'System config.', and a question mark for 'Help'.


Figure 4-10: System configuration



4.4.6 System Help

System Help

-  **Sensor values screen**
Give ability to monitor any sensor connected to the current node, currently set to monitor, temperature, light and smoke sensors, where sensors values are read every 1 second.

-  **Timer reading screen**
Read the sensor values every specified interval (in minutes) where a logging feature is also included.


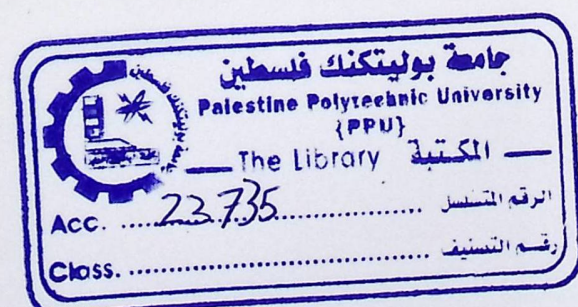
-  **System configuration screen**
Give user ability to configure the system, where he can change the password, node IP, and test the different pins in a specific node.

Figure 4-11: System Help



CHAPTER FIVE

SYSTEM SOFTWARE DESIGN

5.1 Communicating with Micro Web Server

5.2 Software Parameters

5.3 Micro Web Server Control Protocol

5.4 Software Implementation

5.5 Flowcharts

CHAPTER FIVE

SOFTWARE SYSTEM DESIGN

5.1 Communicating with Micro Web Server

To communicate with Micro web Server many methods can be used:

1. **Java applet with Jar files:** This method required to upload Java applets onto a web page on the IP μ 8930 web server. The IP μ 8930 can be used to read data via a web page served from its on-board file system. This web page in turn contains a Java applet archive (JAR file) with several Java applets that communicate with the IP μ 8930 using the Ipsil Control Protocol (ICP). ICP is a stateless protocol over TCP or UDP that provides programmatic access to all the I/O and storage resources of the IP μ 8930.
2. **Using Web Hole:** One of the unique features of the IP μ 8930 is the ability to define placeholders in a file stored on the IP μ 8930, which are called WebHoles. WebHoles allow IP μ 8930 developers to create dynamic web pages that are updated based on the status of devices hooked to the chip. So, for example, when a Web page stored on the IP μ 8930 is requested, the IP μ 8930 Web Server will insert the current value of the WebHole before sending the page. WebHoles can be defined for any type of file such as a text

file or a XML file. These WebHoles are linked to specific IP μ 8930 resources general I/O pins, RAM variables, the internal EEPROM, the external EEPROM the serial port, the I2C port or the SPI port. When the file is requested via the web server, the value from the I/O port is obtained and inserted into the HTML file at the pre-defined location.

3. **Direct communication:** communication oriented method; where the node must be always connected to the network. Those nodes contain different kind of sensors to read data from them, special software was implemented.

Why was direct communication method used?

There are many reasons for using this method rather than the others, which can be shortly defined as following:

1. Give the user a fully control over the Micro Web Server configuration.
2. Ability to search for the connected nodes, where their IP's and locations are unknown at the beginning.
3. Ability to observe all the sensors in any node at any time.
4. Give the user an option to read any node he wants, just by specifying the location and IP.

5.2 Software Parameters

Through programming of Micro-Web Server there are several steps that must be done relating to ICP to the Micro-Web Server:

1. Verifying the IP Address of the Micro-Web Server.
2. Port ID for the Socket path (Manufacturer to be 8930).
3. Parallel Port Address in the Micro-Web Server.
4. Operation Code (OPCODE) of the operation to be performed through the Socket (Read/Write).
5. Verifying the success of the specified operation.

5.2.1 Software Tools

The following components from the software point of view are used:

1. Window XP SP2
To acts as Client.
2. Netbeans 5.5
In order to design and implement the Java software application
3. Java 2SDK 1.4.1
Contains the Frame work of the java environment.
4. Java Run Time Environment (JRE)
To support the Java Virtual Machine for the client processing state.
5. Ipsil Configuration Utility (ICU)
Software to upload files into the Micro-Web Server.

5.3 Micro Web Server Control Protocol

The IP μ 8930 Control Protocol (ICP) provides programmatic access to all of the I/O and storage resources on the IP μ 8930 Board including the general I-O pins (like Port A), the Serial Port, IP μ 8930 RAM, the internal EEPROM, and the external EEPROM. Both read and write operations to IP μ 8930 resources are supported.

The ICP is a stateless protocol which supports reading and writing to the IP μ 8930 via port number 8930. Both TCP and UDP are supported. If the IP μ 8930 receives a request via UDP, it will reply to the request using UDP. Similarly, if a TCP request is received, the outgoing packet will be a TCP packet.

The ICP data structures described below should be written as the data contents of a standard TCP or UDP packet.

The flow control for ICP is lock-step. A program must, after writing a packet, wait for a reply before sending another packet.

The ICP packet format consists of several parts. First, every write packet starts with a password, which is required if secure mode is enabled—otherwise it is ignored. An OPCODE and address are next required. The remaining fields carry the information read from or to be written to the IP μ 8930. The contents of the data structure differ based on the operation and whether this is a request or response. ^[14]

The following tables detail the packet structure of an ICP read, an ICP write and their respective responses:

Table 5- 1: Packet – Read Request.

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	5	PASSWD	Five character ASCII password— always required. Will be ignored if secure mode (SECMODE) is not enabled. If SECMODE is disabled, it is recommended that this field be zero padded.
0x05	1	OPCODE	2 = Read from External EEPROM 4 = Read from Internal EEPROM 7 = Read from RAM location 9 = Read from serial port 10 = I2C Read (v1.1)

Table 5- 2: ICP Packet – Read Response.

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	4	ADDR	Address argument for the requested read. Same address as sent in the read request.
0x04	4	TIMESTAMP	MSB is first; LSB is last. This is the running count

			of time since last power up measured in 8 ms increments. These bytes can be used to attached a time reference to the data.
0x8	1-32	READDAT	Requested data

Table 5- 3: ICP Packet – Write Request

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	5	PASSWD	Five character ASCII password— always required. Will be ignored if secure mode (SECMODE) is not enabled. If SECMODE is disabled, it is recommended that this field be zero padded.
0x05	1	OPCODE	3 = Write to External EEPROM 5 = Write to External EEPROM 6 = Write to RAM location 8 = Write to serial port
0x06	4	ADDR	Address argument for the above operations. Addresses are in 'big endian' format, e.g., MSB comes first.
0x0A	1	WRITEDATA	Data to be written to the above address—must be from 1-30 bytes in length for UDP packets and 1-20 bytes for TCP packets.

Table 5- 4: ICP Packet – Write Response

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	1	STATUS	Will be equal to 0xFF if successful.
0x01	1	SEQNUM	Serial Sequence number. This is only present in response to op-codes 8 or 9. It contains the sequence number of the serial command just sent.

5.4 Software Implementation

The software is made in the system for the following objective:

1. Searching for connecting nodes: the software will be used to search for multiple nodes connecting to the network and getting their IP's and locations, then store it in specific file (see section 5.5 for more details).
2. Read sensor for getting a specific data: one of the main objectives of the project is to read data from different sensors at the same time or with interval reading.
3. Testing the Micro Web Server: through this software you can easily configure the Micro Web Server, change its IP address, and test it.
4. Log files: used to store the sensor readings every specified interval in minutes.
5. Secure system: using this system requires a specific password for more security.

5.4.1 Searching for nodes

In general, searching for nodes are done by getting the network ID's for all devices that are connected to the network. Then searching for the host ID's, if there is any response with [0xFF], then this node is a micro web server. The following code show how searching node process work.

```
public void ping(String IP){
    try {
        System.out.println("ping "+IP);

        isa= new InetSocketAddress(IP, IPSIL_PORT);
        sock=new Socket();
        sock.connect(isa,TIME_OUT);

        // Create input data stream
        sIn = sock.getInputStream();

        // Create output data stream
        sOut = sock.getOutputStream();

        //set all outputs
        sOut.write(outputData);

        // Read and ignore replay byte from the Ipsil device
        sIn.read(resp);

        if ((resp[0] & 0xff) == 0xff ){
            foundedIPs++;
            if (foundedIPs <= 6){
                IPs[foundedIPs-1]=IP;
                nodesFrame.IPArray[foundedIPs-1].setfText(IP);
            }
        }
    }
}
```

As seen in the code above, a *ping* process has been implemented where the program provides this method with the IP to check for, the ping process tries to open a new socket to this IP, if the IP is used by any other host then a connection will be established and a specific command word will be sent to that socket, in result if the response was [0xFF], then this IP is a micro web server, otherwise its not, an exception may happen here when the IP is not used at all, in this case a TIME_OUT parameter is used to terminate the connecting process, minimum TIME_OUT that can be used is one second.

Searching for connected nodes needs about 5 minutes where each IP in the network will be checked to see if it's an available node or not, checking each node requires at least one second = 1000 ms. Although some nodes can respond in less than 100 ms, other nodes may take longer time, so TIME_OUT parameter of the connection was set to 1000 ms. Two facts are known about the searching process which are:

- Takes 5 minutes each time at least
- Only required when a new node is connected or disconnected.

5.4.2 Input data

For the two facts above a storing method was implemented where all the IPs of the connected nodes and their specified locations are stored in a special file, and loaded each time the application is started.

The IPs and locations file (ipsAndLocs) stores the IP addresses and the locations as in figure (5-1).

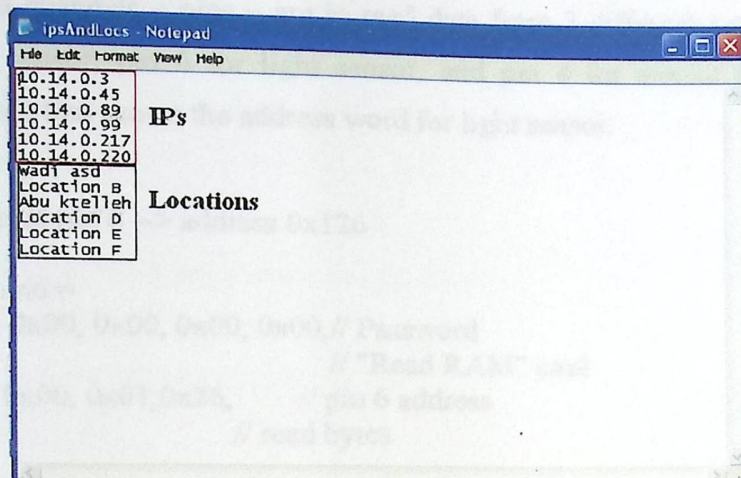


Figure 5-1: ipsAndLocs file format

This system consists of 6 node with 6 IP's and their locations .In searching for nodes process ,the IP address for the first node that have been discovered ,will be stored at the first line in the file, and for it's location ,it will be stored after all IP's and so on .

5.4.2 Input data

For input data mode all channels (eight pins) must be set as analog (0x80) and configured all pins as input (0xff) channel. See the following:

```

byte [] commandWord =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password
  0x06, // "Write EEPROM" cmd
  0x00, 0x00, 0x01, 0x13, // Start location
  (byte)0x80, // All channels Analog
  (byte)0xff, // All channels as inputs
};

```

These channels – pins – are to read data from 3 different sensors, pin 7 for temperature sensor, pin 6 for light sensor, and pin 4 for smoke sensor. See the following code that shows the address word for light sensor.

For example:

//light sensor at pin 6 --> address 0x126

```
byte[] pin6 =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password
  0x07, // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x26, // pin 6 address
  0x06, // read bytes
};
```

Reading sensors are done by sending one of the specific command words and then reading the response, light and smoke sensors produce [0 or 5] volts output, where the temperature sensor gives a range between [0-4.5] volts, this voltage is then read by the temperature pin and converted to digital value between (0-1023 hex values), this value is then processed to give the correct temperature. Processing the range from [0 to 1023] has been done using 8 linear equations; these equations were made by recording temperature and hex value at 9 different temperatures. Figure (5-2) shows the 8 linear relations obtained.

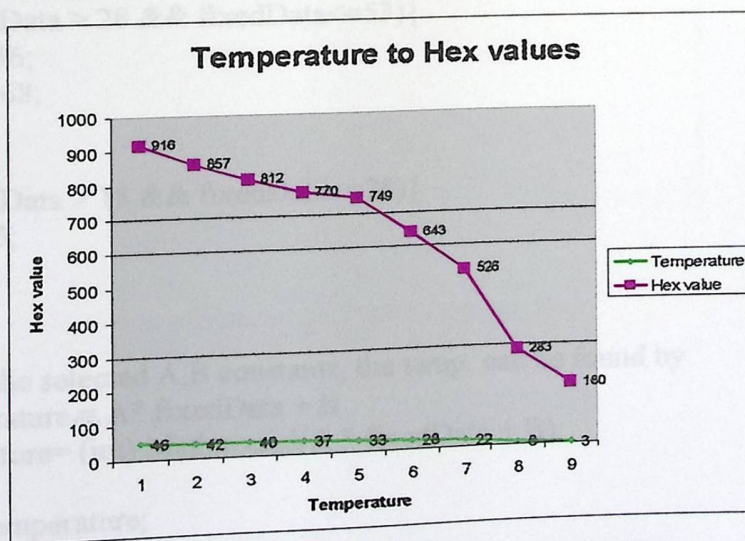


Figure 5-2: Plot of Temperature sensor readings.

5.4.3 Logging file

Those equations are in the form of $y-y_1=m(x-x_1)$, which m is the slope.

For example:

→ Lets take $x_1=916$, $x_2=875$ (Hex value)
and $y_1=46$, $y_2=42$ (Temperature value) .then:

$$y-y_1=m(x-x_1)$$

$$y-46=((42-46) / (857-916))(x-916)$$

$$y=Ax+B$$

Which mean → $A=\text{Slope}=(y_2-y_1) / (x_2-x_1)$.

Line 2: $B=(\text{Slope}) * (-x_1) + y_1$. And so on. (0-50)

See the following code that show 2 of those equations:

Line 4: smoke sensor reading (0 or 1)

```
public static int setTemperature(int data){  
  
    double fixedData=Math.round(data/10);  
    double A=0;  
    double B=0;  
    int temperature=0;  
  
    if (fixedData > 28 && fixedData<=53){  
        A=0.56;  
        B=-7.68;  
    }  
  
    if (fixedData > 18 && fixedData<=28){  
        A=0.5;  
        B=-6;  
    }  
  
    //using the selected A,B constants, the temp. can be found by  
    //temperature = A* fixedData + B  
    temperature= (int) Math.round(A * fixedData + B);  
  
    return temperature;  
}
```

Figure 5-3: Part of the log file

5.4.3 Logging file

File logging will be an additional feature in the software program, where all sensors will be read every X minutes-specified by the user, and then these values will be stored in a log file as shown in the figure (5-3).

Log file format:

Line 1: the time for first reading that we have obtained.

Line 2: the temperature value in celsius degrees (0-50).

Line 3: light sensor reading (0 or 1)

Line 4: smoke sensor reading (0 or 1)

And so on.

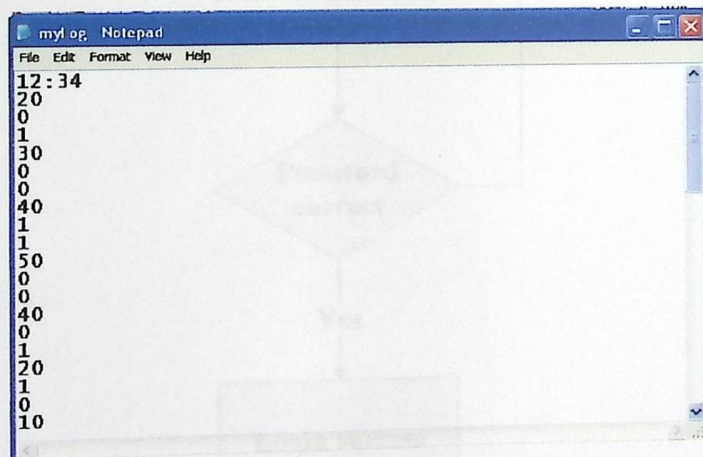


Figure 5-3: Part of the log file.

5.5 Flowcharts

5.5.1 Login flow chart (check password)

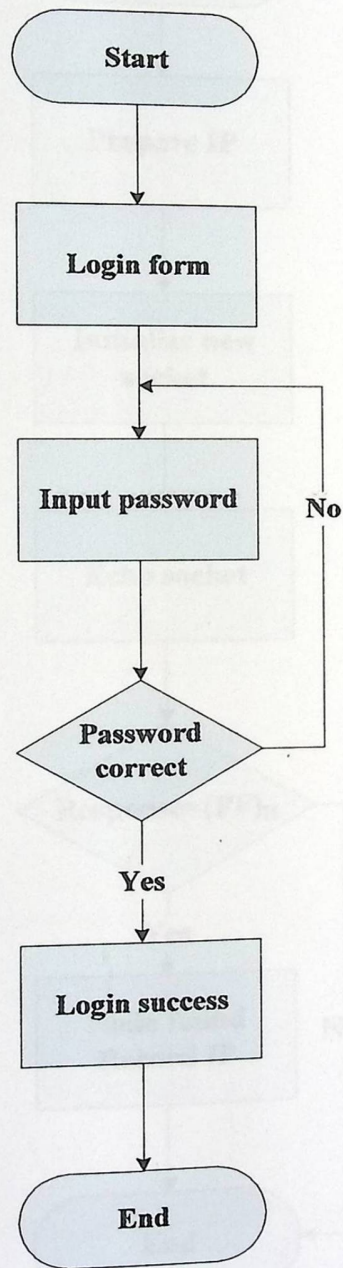


Figure 5-4: Login flow chart (check password)

5.5.2 Ping process flow chart

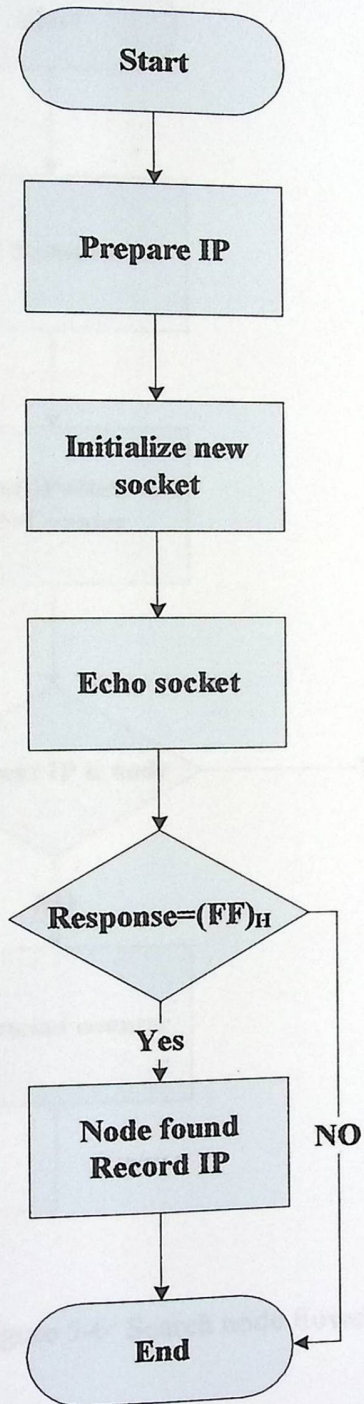


Figure 5-5: Ping process flow chart

5.5.3 Search node flowchart

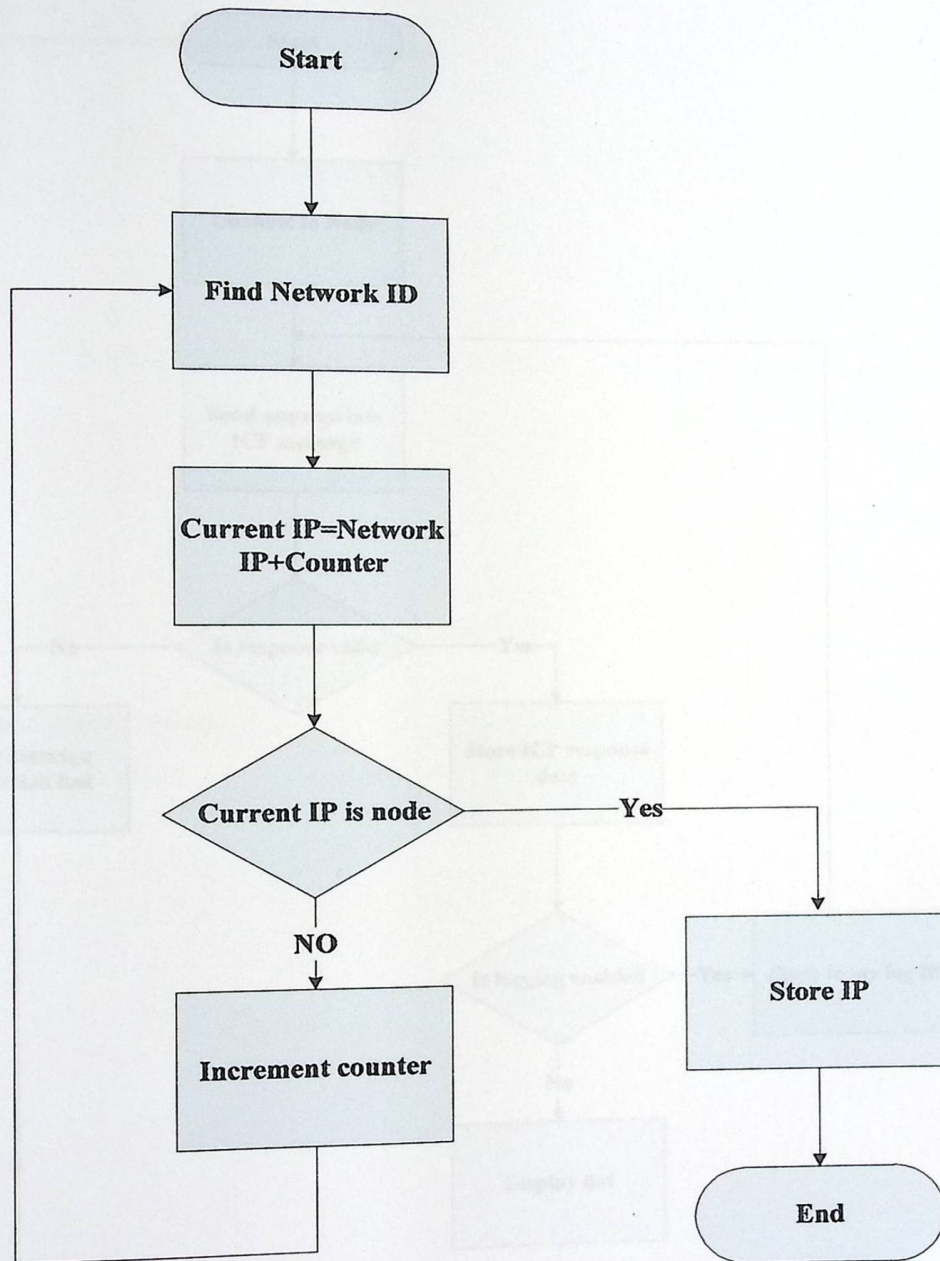


Figure 5-6: Search node flowchart.

5.5.4 Input data flow chart

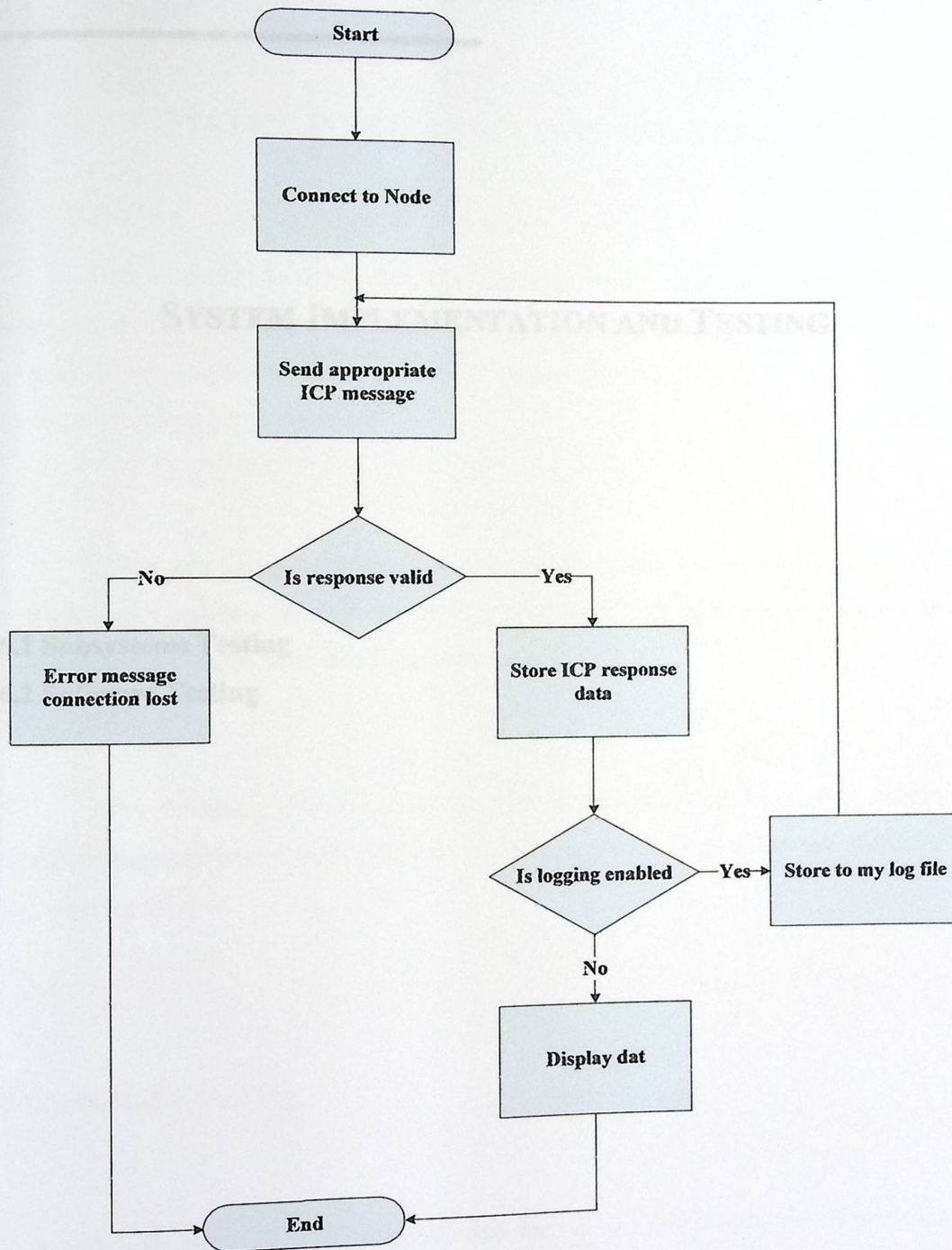


Figure 5-7: Input data flow chart

SYSTEM IMPLEMENTATION AND TESTING

This chapter demonstrates the methods and procedures used to test and examine the **SYSTEM IMPLEMENTATION AND TESTING** important and crucial step in implementing a system. It assesses the effectiveness of that system just before introducing it to its users.

This system has more than one issue to be tested. Some testing parts reflect a software, or hardware case. Also, testing procedures concentrate on a single device or whole system.

6.1 Subsystems Testing

6.2 Software Testing

After finishing the design of the system and drawing the system schematic, the next step was to test each part individually, and then implement the system using the wire rapping.

6.1 Subsystems Testing

At this stage each subsystem was tested independently, to ensure that the subsystem realizes its specified function. This way of testing simplifies trouble shooting detection.

CHAPTER SIX

SYSTEM IMPLEMENTATION AND TESTING

This chapter demonstrates the methods and procedures used to test and examine the system operation and behavior. System testing is an important and crucial step in implementing a system. It senses the effectiveness of that system just before introducing it to its users.

This system has more than one issue to be tested. Some testing parts reflect a software, or hardware case. Also, testing procedures concentrate on a single device independent from the over whole system.

After finishing the design of the system and drawing the system schematic, the next step was to test each part individually, and then implement the system using the wire rapping.

6.1 Subsystems Testing

At this stage each subsystem was tested independently, to ensure that the subsystem realizes its specified function. This way of testing simplifies trouble shooting detection.

6.1.1 Testing the Micro Web Server

Testing the Micro Web Server involves doing more than one step. In order to test it, there must be a simple installed network; this is accomplished by connecting the Micro Web Server with a simple client (PC) that has an Internet Browser (i.e. Internet Explorer, or Netscape navigator). Testing the Micro web server is done by simply asserting one of the following conditions:

- Structuring a cross over cable between a PC and the Micro Web Server
- Connecting the Micro Web Sever to one of a Hub ports

After connecting the cable between the Micro Web Server and the PC, power on the Micro Web Server to complete the network, some testing commands are done:

1. Testing the TCP/IP Protocol

This test is configured at the client side PC because the Micro Web Server already supports the TCP and UDP Protocols. The test is done simply by writing the following command on the Command Line Shell:

```
Ping 10.14.0.80
```

The following reply indicates the success of the installed TCP/IP protocol:

Pinging 10.14.0.80 with 32 bytes of data:

Reply from 10.14.0.80: bytes=32 time<1ms TTL=128

Reply from 10.14.0.80: bytes=32 time<1ms TTL=128

Reply from 10.14.0.80: bytes=32 time<1ms TTL=128

Reply from 10.14.0.80: bytes=32 time<1ms TTL=128

Ping statistics for 10.14.0.80:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 0ms, Average = 0ms

2. Testing Micro Web Server Reachability

The Micro Web Server when connected to a network is configured as a DHCP client. Detection packets are sent when sharing a network to have an IP Address from a DHCP server. When no server satisfies that, the 192.168.0.254 IP address is assigned to the Micro Web Server.

Testing the Micro Web Server assumes no DHCP server is found. So, the following command will test the Micro Web Server:

Ping 192.168.0.254

The following reply indicates the success of Micro Web Server accessibility:

Pinging 192.168.0.254 with 32 bytes of data:

Reply from 192.168.0.254: bytes=32 time=1ms TTL=128

Reply from 192.168.0.254: bytes=32 time=1ms TTL=128

Reply from 192.168.0.254: bytes=32 time=1ms TTL=128

Reply from 192.168.0.254: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.0.254:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 1ms, Maximum = 1ms, Average = 1ms

6.1.1.1 Testing TCP/IP Sockets

Testing win sockets needs applying some codes and executing it through a java applet. There are two main applets used to test both the Parallel port and the 8 General I/O pins.

6.1.1.1.1 Testing Port A

There are 8 general purpose I/O pins installed on the Micro Web Server. The following testing code is used to implement a socket through a java applet to flash 8 LEDs ON and OFF:

```

import java.applet.Applet;
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;

public class setDigitalHi extends Applet
{
    private int IPSIL_PORT = 8930;
    InetAddress ia = null;
    Socket sock = null;
    InputStream sIn = null;
    OutputStream sOut = null;
    StringBuffer buffer = new StringBuffer();
    byte [] setAllDigitalOutput =
    { 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
      0x06, // "Write EEPROM" cmd (1 byte len)
      0x00, 0x00, 0x01, 0x13, // Start location (4 bytes len)
      (byte)0x86, // All channels Digital (1 byte len)
      0x00 // All channels as Outputs (1 byte len)
    };

    byte [] flashLedsOn =
    { 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
      0x06, // "Write EEPROM" cmd (1 byte len)
      0x00, 0x00, 0x01, 0x17, // Start location (4 bytes len)
      (byte)0xff // All channels HI (1 byte len)
    };

    byte [] flashLedsOff =
    { 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
      0x06, // "Write EEPROM" cmd (1 byte len)

```

```

0x00, 0x00, 0x01, 0x17, // Start location      (4 bytes len)
(byte)0x00             // All channels Low    (1 byte len)
};

byte [] resp = new byte[1024];           // Network response buffer

public void init() {
try {
ia = InetAddress.getByName( getCodeBase().getHost() );
sock = new Socket(ia, IPSIL_PORT);
addItem(ia.toString());
sIn = sock.getInputStream();
sOut = sock.getOutputStream();
}
catch (Exception e)
{
    e.printStackTrace();
    addItem(" Init ");
}
}

private void addItem(String newWord)
{
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}

public void paint(Graphics g)
{
    g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);
    g.drawString(buffer.toString(), 5, 15);
}

public void start() {
try{
    sOut.write(setAllDigitalOutput);
}
}

```

```

sIn.read(resp);
sOut.write(flashLedsOn);
sIn.read (resp);
Thread.sleep( 1000 );
addItem("LEDs ON!");
sOut.write(flashLedsOff);
sIn.read (resp);
Thread.sleep( 1000 );
addItem(" LEDs OFF!");
}
catch (Exception e)
{
    e.printStackTrace();
    addItem(" Start ");
}
}
}
}

```

6.1.1.1.2 Testing Parallel Port:

The testing of the parallel port via a socket is implemented by connecting a LED to the transmit line of the parallel port. Then an applet is written to flash the led ON and OFF, here is the testing applet code:

```

import java.applet.Applet;
import java.io.*;
import java.net.*;
import java.util.*;

```

```

import java.awt.*;

public class Parallel extends Applet
{
    private int IPSIL_PORT = 8930;
    InetAddress ia = null;
    Socket sock = null;
    InputStream sIn = null;
    OutputStream sOut = null;
    StringBuffer buffer = new StringBuffer();

    byte []parallelHi =
    { 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
      0x08, // "Write Parallel" cmd (1 byte len)
      0x00, 0x00, 0x00, 0x00, // Start location (4 bytes len)
      (byte)0xff, (byte)0xff, (byte)0xff, (byte)0xff, (byte)0xff
    };

    byte [] parallelLo =
    { 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
      0x08, // "Write Parallel" cmd (1 byte len)
      0x00, 0x00, 0x00, 0x00, // Start location (4 bytes len)
      (byte)0x00, (byte)0x00, (byte)0x00, (byte)0x00, (byte)0x00
    };

    byte [] resp = new byte[1024]; // Network response buffer

    public void init()
    {
        try {

```

```
ia = InetAddress.getByName( getCodeBase().getHost() );
sock = new Socket(ia, IPSIL_PORT);
addItem(ia.toString());
sIn = sock.getInputStream();
sOut = sock.getOutputStream();
```

```
}
```

```
catch (Exception e)
```

```
{
    e.printStackTrace();
    addItem(" Init ");
}
```

```
}
```

```
private void addItem(String newWord)
```

```
{
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}
```

```
public void paint(Graphics g)
```

```
{
    g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);
    g.drawString(buffer.toString(), 5, 15);
}
```

```
public void start() {
```

```
try {
```

```

boolean n = true;
while( n ) {
    if( n % 2 == 0 ){

        sOut.write(parallelHii);
        sIn.read(resp);

    }

    else{ sOut.write(parallelLo);
        sIn.read (resp);
    }

}

catch (Exception e)
{
    e.printStackTrace();
    addItem(" Start ");
}
}
}

```

6.1.2 Testing temperature sensor circuit

After connecting the LM35 sensor to the Op Amp, the circuit was connected to the power supply and tested the output voltage of the circuit. As we see at the next figure the output on multimeter is 3.56 V which mean that the temperature is 32 °C according to the equations:

Gain= $\frac{R1+R2}{R1}$, Where the R1=10K, R2=100K.

R1

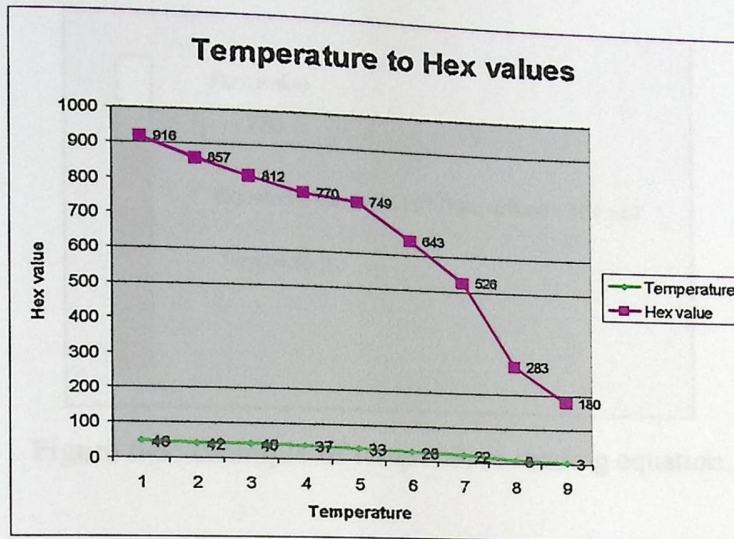


Figure 6-2: Plot of Temperature sensors' readings.

For calculating the temperature sensors' readings, we built 8 linear equations; each of these equations has a specific range. If the hexadecimal value we read verifies one range, then we choose the equation associated to this range.

Those equations are in the form of $y-y_1=m(x-x_1)$, which mean:

$$m=\text{slope}=\frac{y_2-y_1}{x_2-x_1}$$

x_1 & x_2 are Hexadecimal value

y_1 & y_2 are Temperature value then

$$\Rightarrow y - y_1 = m(x - x_1)$$

$$y = Ax + B$$

Where $\Rightarrow A = \text{Slope} = \frac{(y_2 - y_1)}{(x_2 - x_1)}$.

$$B = (\text{Slope}) * (-x_1) + y_1.$$

See Figure (6-3) that shows one of these equations.

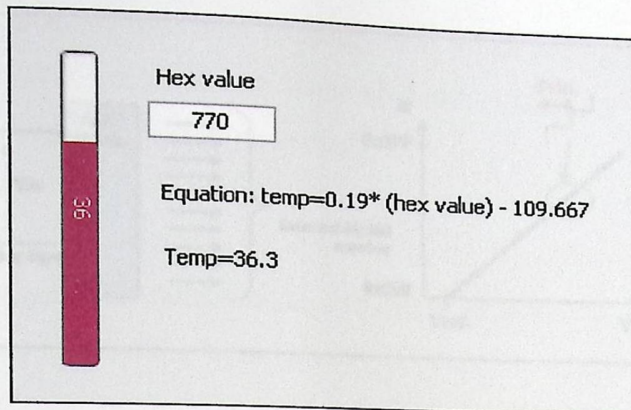


Figure 6-3: Example of temperature reading equation.

Micro web Server Reading's Values were calculated depending on the following rules: ^[2]

- The Analog-to-Digital (A/D) converter that allows conversion of an analog input signal to a corresponding 10-bit digital number as shown in figure (6-4).
- 10-bit digital number can be read as float number (N).
- N is converted to its corresponding voltage value using the following equation:

$$N = \frac{2^{10}}{(V_{ref+} - V_{ref-})} V_{in}$$

Where:

$V_{ref-} = 0$ and $V_{ref+} = 5$ Volt.

For example if the temperature is 7 C, then

$$N = 1024 / (5-0) * 0.07 = 14.3$$

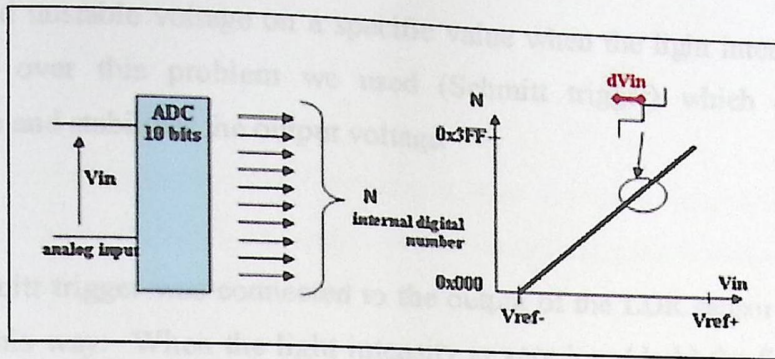


Figure 6-4: Analog-to-Digital (A/D) converter modules.

6.1.3 Testing Light Sensor circuit

The following Figure shows the initial light sensor circuit which was built in the laboratory. Testing this circuit needs the use of the multimeter to indicate the output voltage.

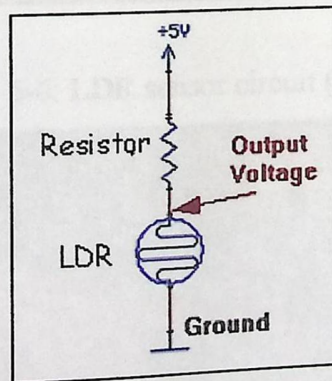


Figure 6-5: Initial light sensor circuit.

When connecting the first edge of the LDR sensor in series with 150Ω resistor and the second edge to the GND (0 volts), we face a problem of reading a

very low and unstable voltage on a specific value when the light intensity changes. For getting over this problem we used (Schmitt trigger) which enhances the performance and stabilizes the output voltage.

We connected the LDR sensor circuit as shown in figure (6-6). When the sensor detects any variation of voltage in the surrounding environment, it gives an alarm.

Schmitt trigger was connected to the output of the LDR sensor so the circuit worked in this way: When the light intensity is very low (dark) the Schmitt trigger output is almost 5 volt (figure 6-6), and when the light intensity is very high (day) the Schmitt trigger output is almost 0 volt (figure 6-7).

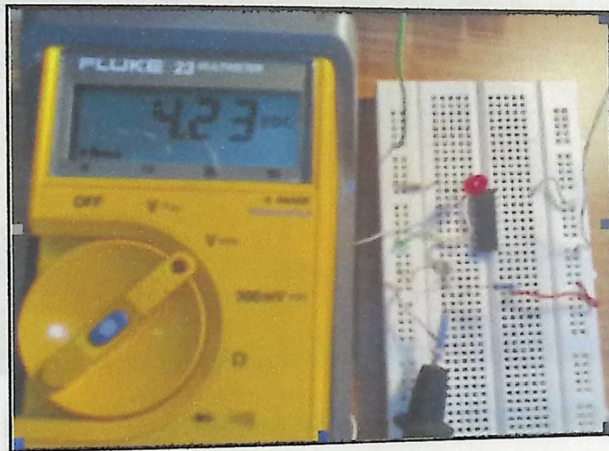


Figure 6-6: LDR sensor circuit (at day).

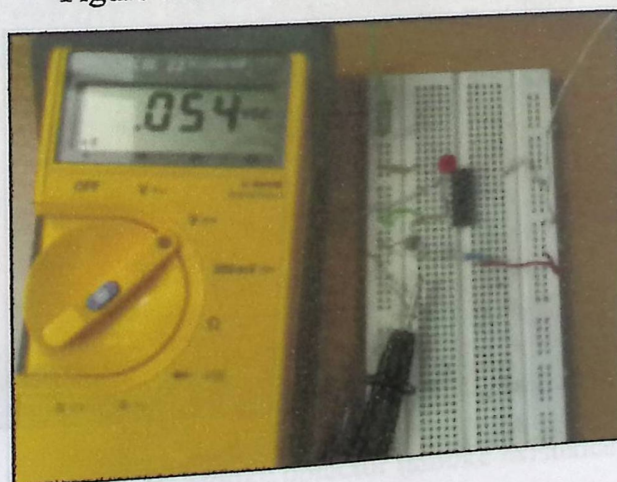


Figure 6-7: LDR sensor circuit (at dark)

6.1.4 Testing the Smoke detector circuit

We connected the smoke detector circuit as shown in figure (6-8). When the sensor detects any existence of smoke in the surrounding environment, it gives an alarm and the measured voltage on the output leg will be almost (1.5V).

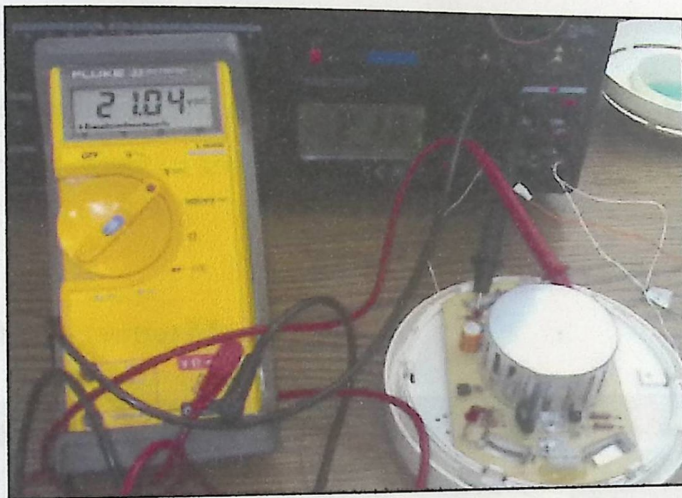


Figure 6-8: Smoke detector (clear environment).

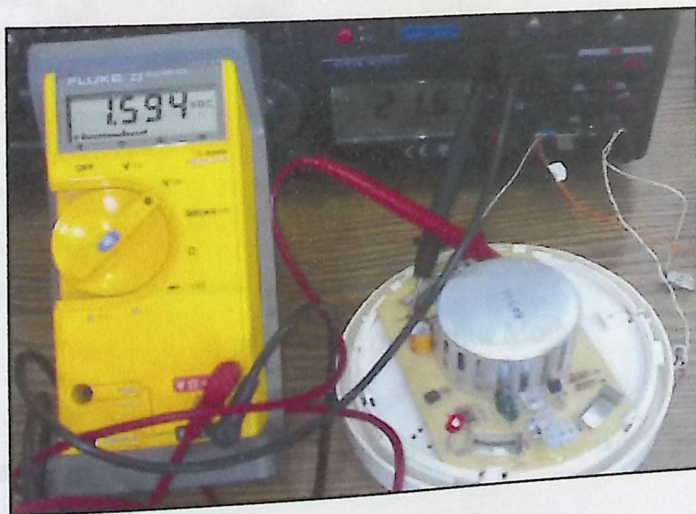


Figure 6-9: Smoke detector (smoke existence).

6.2 Software Testing

To test the project by means of software we designed a program that is responsible for configuring and testing the micro web server, controlling its inputs and outputs and processing them, and determining the means of passing the gained data all over the network.

The following screen shots show how this program works and each number on each figure represents a specific meaning:

- For Output (write) Mode:

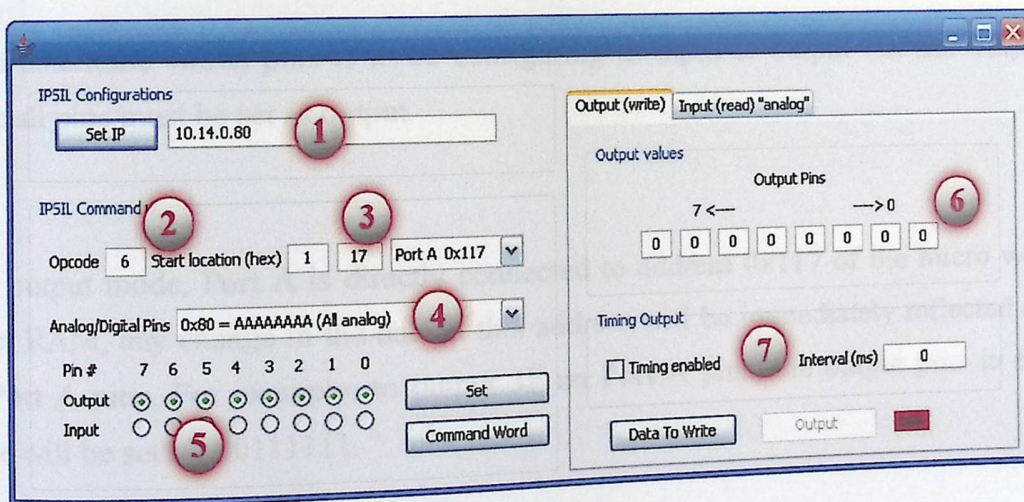


Figure 6-10: Output Mode

1. Set IP: to set the IP address of the micro web server, this way allows us to change it whenever we want.

2. Opcode: for determining command word we must know the operation code (opcode) and the start location of PORT A. at this mode the opcode must be 6 which mean: Write to RAM location.

3. Start location (hex): Address of the memory location, to read or write to, where each address has LSB and MSB in the command word ,address 0x113 is used to write the command word ,where address (0x117) to read or write digital values to/from Port A.

4. Analog/Digital Pins: to determine which pin is to be configuring as analog or digital Pins. At this mode all pins must be set as analog.

5. To determine which pins is to be configuring as input or output Pin .for output mode all pins must be set as output.

6. In output mode, Port A is directly connected to address 0x117 of the micro web server RAM, any change in the data at this address will be immediately reflected on the Port A pins. For example: to output 3F on Port A pins, the output pins in the figure will be set as: 00111111.

7. Configuration: Time interval is used to keep reading the sensors' output automatically each period of time, this period is specified by the interval variable, and it's measured by millisecond.

- For Input(read analog)Mode:

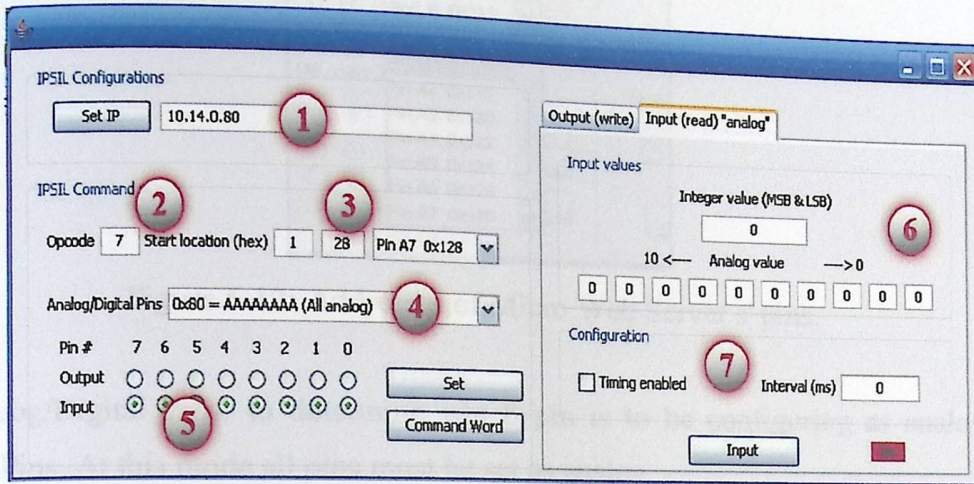


Figure 6-11: Input Mode

1. Set IP: to set the IP address of the micro web server.

2. Opcode: the opcode for this mode will be 7 which mean: Read from RAM location.

3. Addresses (0x11A) to (0x128) are the addresses of Port A pins, this means that (0x11A) is the address of pin 1(PA 0), and (0x128) is the address of pin 8(PA 7), for input mode we must determine the address of which pin is to be used .As we notice in figure (6-8):

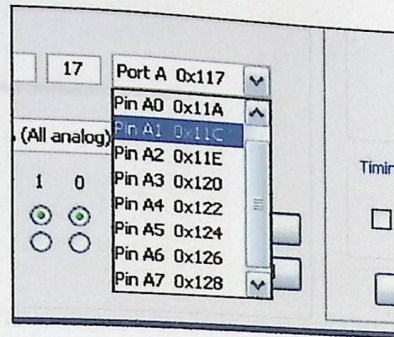


Figure 6-12: Addresses of Micro Web Server s' pins.

4. Analog/Digital Pins: to determine which pin is to be configuring as analog or digital Pins. At this mode all pins must be set as analog.

5. For input (read analog) mode all pins must be set as input.

6.1 Conclusions

6. In input Mode: The micro web server has two input modes:

- Digital input mode: where each pin of Port A is connected to a digital device that produces +5 v or 0v, which will be translated to digital 1, and 0 at Port A
- Analog input mode (as seen in figure): in this mode each pin of Port A is connected to a 10 bit Analog to Digital converter, (10 bit A/D), that mean the minimum value being read is 0 and the maximum is 3FF.

In other words, at input 0v the digital equivalent will be 0, and at input +5v the digital equivalent will be 3FF (1023 decimal), as shown in the figure, the top part represents the decimal converted value and the below part represents the same value but in bitwise mode.

7. Configuration: as mentioned before.

PROBLEMS, CONCLUSIONS AND
FUTURE WORKS

PROBLEMS, CONCLUSIONS AND
FUTURE WORK

7.1 Conclusions

7.2 Problems

7.3 Future Work

CHAPTER SEVEN

PROBLEMS, CONCLUSIONS AND FUTURE WORKS

This chapter will list the problems faced us in accomplishing the system, conclusions and a future work are also proposed.

7.1 Conclusions

In this project, we have navigated through many experiences that we have never gone through before; we have learned different approaches and experiences especially the way of thinking and how to develop an approach to solve problems. Many conclusions can be stated here, but only significant and important ones are described here:

- By the end of this project, we have finished designing and implementing Smart Sensor node using a Micro Web Server, this system enables the user to search for connecting nodes, read sensor for getting a specific data and testing the Micro Web Server.

- The main device in the system is ipu9830 Micro Web Server; for programming it we used IDE NetBeans.
- This project challenged us as engineers and it was very demanding as the team spent 30-40 hours a week in the university lab working on this project in the last month. We learned a lot and used everything we had learned in our classes to solve the problems and come up with solutions to make this system work.
- This project proved that our ideas are viable for implementation in a project that resembles a real-life problem. We mostly implemented our main objectives during the semester.
- Each device was tested individually in its own circuit by means of Hardware and Software to study its behavior and make sure it works properly and can do its expected job.
- Then the whole system was tested and gave the expected results.

7.2 Problems

System completion in regard to its objectives is an implementation dependent issue. Problems are natural things. Skipping these problems is a success. No

degradation affects the system if problems appear. Here are problems faced the project team during the system implementation

1. The PIC Microcontroller which we decided to use was not available.
2. Instead of using a PIC microcontroller we decided to use a micro web sever after two months of semester's beginning, so a lot of time are wasted.
3. Only one Micro Web Server is available in hand. So one node on the system are implemented and tested.
4. Damaging in some devices, because of wrong connections or high voltages, supplied to the devices during the implementation.
5. Searching for nodes process take almost 5 minutes for complete searching, we can't reduce it.

7.3 Future Works

We will be proud if the continuity base on our project is done in order to have more and more elaborated work. Here are some suggested projects:

- Our project seems to have a very promising future, talking about the SMS and E-mails technology .These technologies can be used to make it easier, faster, and more convenient to detect and convey the information if any smoke is detected . It also can send all the data regarding the environmental status like the temperature and the light.

- Design a distributed smart sensor system which contains multiple nodes, with different types of processing.

Web Sites:

- [1] <http://www.smartensensorsystems.com>
- [4] <http://www.eecs.berkeley.edu/~jmg/research/sensors/ANPAsummary.html>
- [5] http://www.ig.ni.nraa.gov/publications/home/bur/smart_sensors.html
- [6] http://www.x-robotics.com/sensors_ig.html
- [7] <http://www.mtracoy.binternet.co.uk/technical/Theory/theorysensors.htm>
- [8] http://en.wikipedia.org/wiki/Smoke_detector
- [9] http://en.wikipedia.org/wiki/Ethernet_crossover_cable
- [10] <http://www1.microchip.com/downloads/en/DeviceDoc/39762a.pdf>

Other Sources:

- [11] P. Seitz, T. Spring, O. Vietze, and K. Engelhardt (1999 Aug)
- [12] P. Ryster and D. Wicker (2002)
- [13] IPu0920 Developer Guide

REFERENCES

Books:

- [1] Data Communications and Networking, 2nd edition update, Behrouz A. Forouzan, McGraw-Hill, ISBN: 0-07-282294-5.
- [2] Microprocessor architecture, programming and applications with 8085, fifth edition, ramesh S. Gaonker

Web Sites:

- [3] <http://www.smartsensorsystems.com>
- [4] <http://www.eecs.berkeley.edu/~jimmy/research/sensors/ARPAsummary.html>
- [5] http://www-aig.jpl.nasa.gov/public/mls/home/burl/smart_sensors.html#Gat94
- [6] http://www.x-robotics.com/sensores_ing.htm#LM35
- [7] <http://www.mstracey.btinternet.co.uk/technical/Theory/theorysensors.htm>
- [8] http://en.wikipedia.org/wiki/Smoke_detector
- [9] http://en.wikipedia.org/wiki/Ethernet_crossover_cable
- [10] <http://ww1.microchip.com/downloads/en/DeviceDoc/39762b.pdf>

Other Sources:

- [11] P. Seitz, T. Spirig, O. Vietze, and K. Engelhardt (1999 Aug)
- [12] P. Ryser and D. Wieser (2002)
- [13] IPu8930 Developer Guide.

APPENDICES

APPENDIX A: IP μ 8930 Micro Web Server Datasheet.

APPENDIX B: Project ICs Datasheets.

APPENDIX C: source code.

1. Introduction & Overview

1.1 General Description

The IP μ 8930 is a general purpose network controller and web server which makes it easy to monitor, control and communicate with remote sensors, actuators, and practically any devices with a serial port (via the onboard serial port) via a TCP/IP network such as the Internet. The IP μ 8930 is designed to either work standalone or in conjunction with a "master" MCU.

The IP μ 8930 Developer Kit consists of the following:

- IP μ 8930 Module
- Product Information Sheet (which includes the device's MAC address, default IP address, serial number, and device write password)
- Regulated +5V DC Power plug/transformer
- RJ-45 to D69 adapter cable

Note: A copy of this manual is available for download from the Insil website.

Figure 1-1: Insil IP μ 8930 Developer Kit Contents

At the heart of the IP μ 8930 Developer Kit is the IP μ 8930 Module, a compact TCP/IP network controller and webserver module, which enables developers to readily add network connectivity to products. The IP μ 8930 combines a TCP/IP controller, HTTP-compliant webserver, Modbus TCP node and A/D converter into a single, small (1.3x1.4"/3.3x3.4cm) daughterboard. The IP μ 8930 is designed to enable remote monitoring and control over a TCP/IP-based network without the overhead and complexity of traditional solutions which require the knowledge on how to support and program using a real-time operating system. The IP μ 8930 Module is capable of operating standalone or as a TCP/IP peripheral for a separate MCU. Figure 1-2 contains a top-side photograph (actual size) of the IP μ 8930.

In addition to being able to store and retrieve web pages on the IP μ 8930, developers can access the I/O ports on the IP μ 8930 using Modbus TCP, a popular standard for

1. Introduction & Overview

1.1 General Description

The IP μ 8930 is a general purpose network controller and web server which makes it easy to monitor, control and communicate with remote sensors, actuators, and practically any devices with a serial port (via the onboard serial port) via a TCP/IP network such as the Internet. The IP μ 8930 is designed to either work standalone or in conjunction with a "master" MCU.

The IP μ 8930 Developer Kit consists of the following:

- IP μ 8930 Module
- IP μ 8930 Developer Board
- QuickStart Guide (hardcopy)
- Product Information Sheet (which includes the device's MAC address, default IP address, serial number, and device write password)
- Regulated +5V DC Power plug/transformer
- RJ-45 to DB9 adapter cable

Note: A copy of this manual is available for download from the Ipsil website.

Figure 1-1: Ipsil IP μ 8930 Developer Kit Contents

At the heart of the IP μ 8930 Developer Kit is the IP μ 8930 Module, a compact TCP/IP network controller and webserver module, which enables developers to rapidly add network connectivity to products. The IP μ 8930 combines a TCP/IP controller, HTTP-compliant webserver, Modbus TCP node and A/D converter into a single, small (1.3x1.4"/3.3x3.4cm) daughterboard. The IP μ 8930 is designed to enable remote monitoring and control over a TCP/IP-based network without the overhead and complexity of traditional solutions which require the knowledge on how to support and program using a real-time operating systems. The IP μ 8930 Module is capable of operating standalone or as a TCP/IP peripheral for a separate MCU. Figure 1-2 contains a top-side photograph (actual size) of the IP μ 8930.

In addition to being able to store and retrieve web pages on the IP μ 8930, developers can access the I/O ports on the IP μ 8930 using Modbus TCP, a popular standard for

accessing devices on a network in the industrial automation, building automation, and utilities industries.

Also included in the IP μ 8930 Developer Kit is an Ipsil IP μ 8930 Developer Board. The IP μ 8930 Module plugs easily into the IP μ 8930 Developer Board. The Developer Board is packaged with power, a 10BaseT network jack, and various connectors to make it easy to set up, program, and test the IP μ 8930.

The following sections provide additional detail on the features of both the IP μ 8930 and the IP μ 8930 Developer Board.

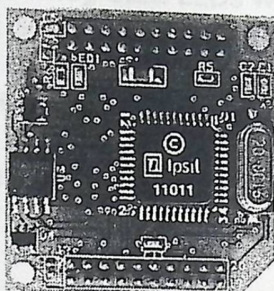


Figure 1-2: IP μ 8930 Top View (Actual Size)

1.2 IP μ 8930 Features

The IP μ 8930 Board provides the following features:

- Supports both TCP/IP and UDP—read & write
- Includes HTTP v1.0 compliant embedded webserver
- Up to 8 analog or digital ports for monitoring/control of external devices
- Built-in 10-bit ADC
- Serial port
 - With flow control (CTS/RTS, XON/XOFF, or None)
 - Can be used to communicate with external serial devices
- Ability to define WebHoles™ in HTML pages which get "filled" with values from pre-defined I/O ports
- Can operate standalone or as a peripheral to an external MCU using a serial interface
- Full set of user-configurable network and system settings including
 - Support for static & dynamic IP addresses
 - User-definable default fixed address
- Compliant with RFC-1122, the standard for TCP/IP hosts on the Internet
- Supports the following network applications:
 - ARP, DHCP, ICMP (ping), Modbus, and Ipsil Control Protocol (v1.0)
- 4 megabit (512KB) of on-board memory for web pages.
- Time stamp functionality
- File system with over 500 unique, variable length files

Without any additional hardware, a user can download web pages from the IP μ 8930 Board and control the analog/digital general I/O ports on the Developer Board. With additional devices hooked into the Serial ports, a user can control and monitor a wide variety of devices.

1.3 IP μ 8930 Developer Board Features

The IP μ 8930 Developer Board is designed to help you quickly develop applications using the IP μ 8930 module. The IP μ 8930 Developer Board contains the following features:

- Ethernet connector with two integrated LEDs which shows Rx/Tx activity
- RS-232 connectivity thru a serial RJ-45 jack (RJ-45 to DB9 cable included) with flow control
- 8 general I/O channels accessible through developer board header
- Service LEDs 5V, 3.3V, Tx, Rx, RTS, CTS activity
- Integrated I²C temperature sensor
- Reset button
- Power and collision activity LED
- External power jack

1.4 IP μ 8930 Applications

Applications for IP μ 8930 Modbus include:

- Remote monitoring
- Industrial automation and process control
- Home automation
- HVAC control
- Lighting control
- Environmental monitoring
- Remote telemetry
- Test and lab equipment monitoring

1.5 Documentation Conventions

When reading this guide, you will see that different words are in different fonts and sizes. They represent inclusion in a specific category such as a button to be clicked. The different types of notations are:

Command

Buttons to be clicked are represented in boldface. The indicated button should be pressed to invoke the function under discussion.

Screen Field/Area

3. Architectural Overview

The purpose of this chapter is to describe the functional blocks that compose the IP μ 8930 Module. Figure 3-1 contains a block diagram of the Ipsil IP μ 8930 Board:

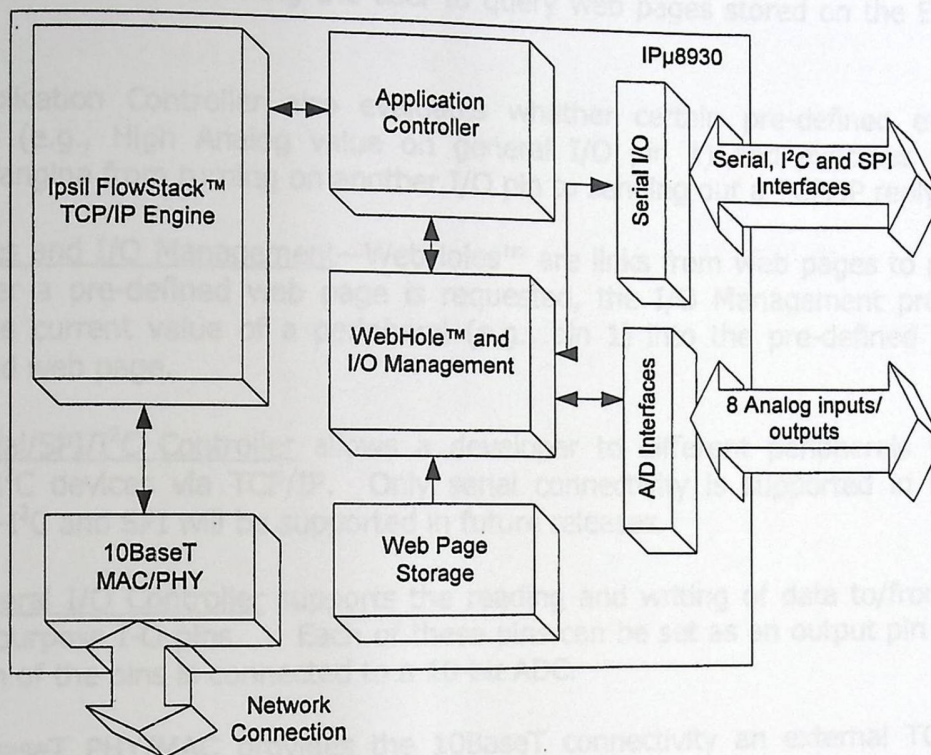


Figure 3-1: IP μ 8930 Block Diagram

The IP μ 8930 Board consists of the IP μ 7931M chip, a 512K byte EEPROM for web content, a Realtek 8019AS Ethernet controller and two female, 20-pin connectors for mating to a master board (such as the IP μ 8930 Developer Board) which provides power, a physical network connection, and access to the general purpose I/O pins.

The TCP/IP Engine performs complete TCP/IP protocol functions, processing all packets received using the built-in FlowStack™ engine. FlowStack is Ipsil's method for compressing and delayering TCP/IP while maintaining full standards compliance.

The TCP/IP Controller performs all of the functions specified in RFC-1122¹, the document that defines the functional requirements of TCP/IP-connected hosts on the Internet. The TCP/IP Controller can also read and write UDP packets as well. In addition, the TCP/IP Controller also executes the TCP/IP "applications" supported by Ipsil including DHCP and ICMP (ping).

The TCP/IP Controller supports IPv4; IPv6 will be supported in a future release.

¹ A copy of RFC-1122 can be found at <http://www.ietf.org/rfc/rfc1122.txt?number=1122>.

The Application Controller implements the user-defined monitoring and control functionality, including decoding I-O requests received from the network and translating and sending these requests over the I²C, SPI, serial, or general I-O controllers, as appropriate.

The Application Controller also contains the web server functionality that processes HTTP 1.0 commands allowing the user to query web pages stored on the EEPROM file system.

The Application Controller also evaluates whether certain pre-defined events have occurred (e.g., High Analog value on general I/O pin 1) and executes pre-defined actions ranging from turning on another I/O pin to sending out a TCP/IP reply.

WebHoles and I/O Management—WebHoles™ are links from web pages to peripherals. Whenever a pre-defined web page is requested, the I/O Management processor will insert the current value of a peripheral (e.g., pin 1) into the pre-defined location on requested web page.

The Serial/SPI/I²C Controller allows a developer to connect different peripherals via TCP/IP. control I²C devices via TCP/IP. Only serial connectivity is supported in the current version—I²C and SPI will be supported in future releases.

The General I/O Controller supports the reading and writing of data to/from the eight general-purpose I-O pins. Each of these pins can be set as an output pin or an input pin. Each of the pins is connected to a 10-bit ADC.

The 10BaseT PHY/MAC provides the 10BaseT connectivity an external TCP/IP-based network.

4. Hardware Overview

4.1 IP μ 8930 Module Layout

The Ipsil IP μ 8930 Board consists of an Ipsil IP μ 7931M TCP/IP controller chip with a 512KB (4Mbit) EEPROM for web page and other user data and a Realtek 8019AS Ethernet controller connectors. The front side of the IP μ 8930 Board is shown in Figure 4-1.

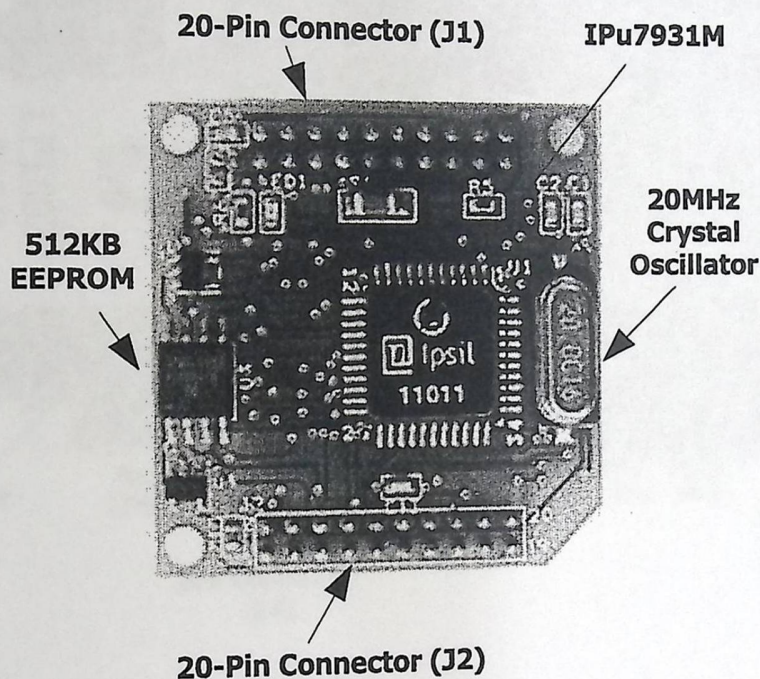


Figure 4-1: IP μ 8930 Module (Front)

The main clock on the board is implemented with a 20 MHz crystal oscillator generated by the IP μ 7931M chip. The output of the oscillator also feeds the Realtek Ethernet controller through a 33-ohm resistor.

The pin layout for connectors J1 and J2 are described in Section 4.6.

4.2 IP μ 8930 Developer Board Layout

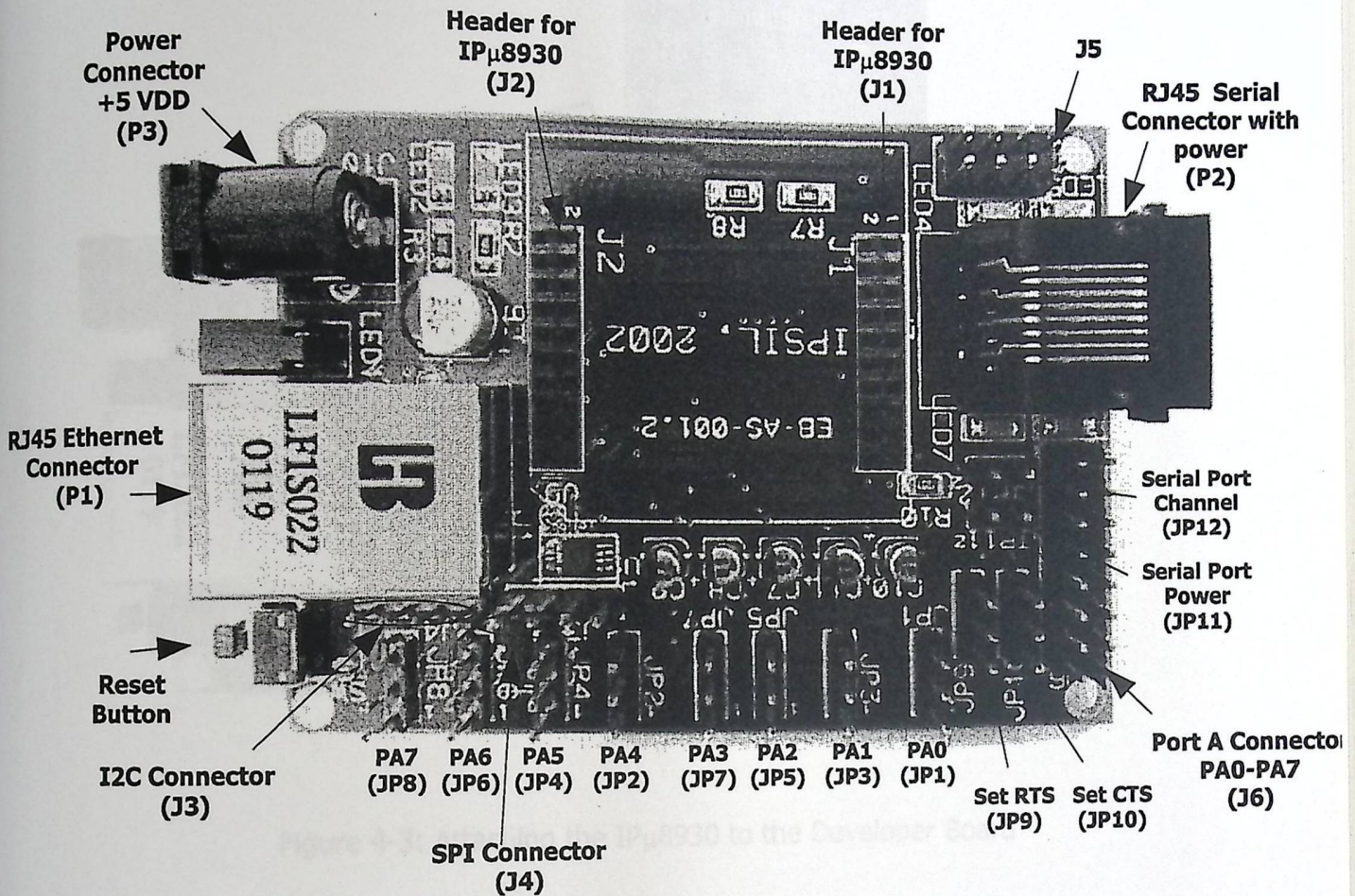


Figure 4-2: IP μ 8930 Developer Board (Front)

The IP μ 8930 Developer Board and its major components are identified in Figure 4-2 and described in more detail in the following sections.

Figure 4-3 also shows, orientation-wise, how the IP μ 8930 Module should be seated on the Developer Board.

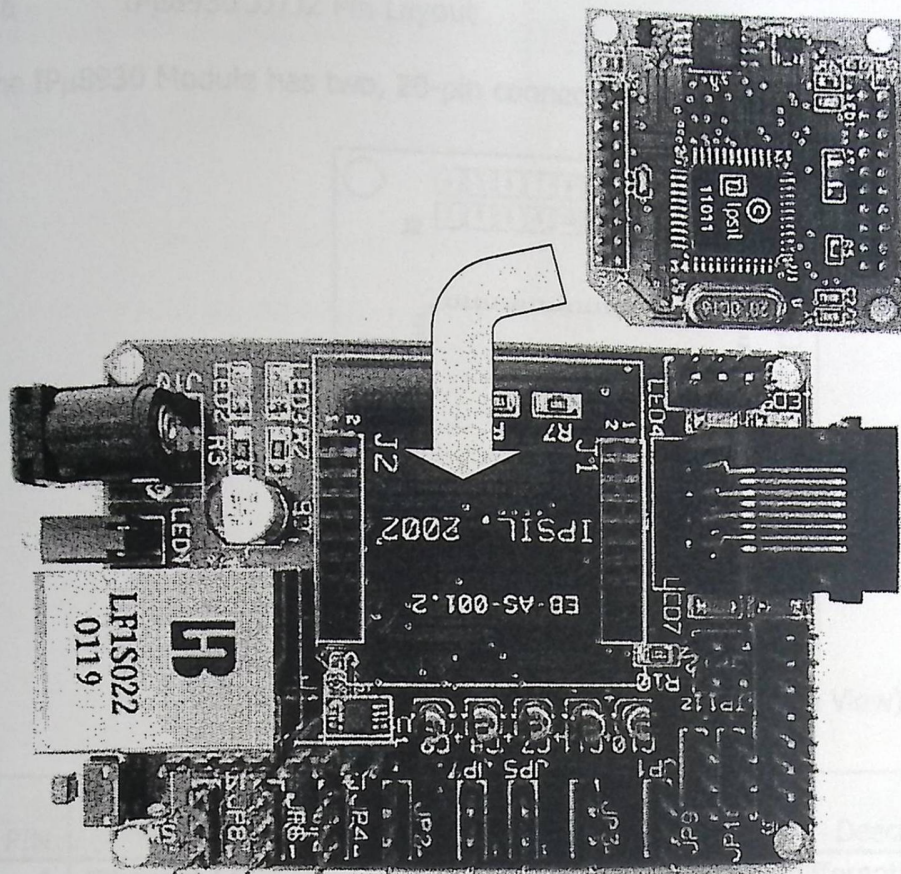


Figure 4-3: Attaching the IP μ 8930 to the Developer Board

4.3 Headers and Connectors

IP μ 8930 Module Headers (J1 and J2)

Junctions J1 and J2 are female connectors, which connect to a two male 20-pin connectors on the IP μ 8930 module. The specific pin assignments are described in Section 4.6.

I²C Connector (J3)

The I²C Connector occupies the J3 junction. Figure 4-4 contains the I²C pin-out summary. Programmatic access to I²C peripherals will be supported in future releases.

The I²C bus physically consists of two active wires and a ground connection. The active wires, called SDA (Serial Data line) and the SCL (Serial Clock line) are both bi-directional. For more information on I²C, a copy of the I²C specification can be found at:

<http://www.semiconductors.philips.com/i2c/facts/#specification>

4.6 IP μ 8930 J1/J2 Pin Layout

The IP μ 8930 Module has two, 20-pin connectors whose pin assignments are defined below:

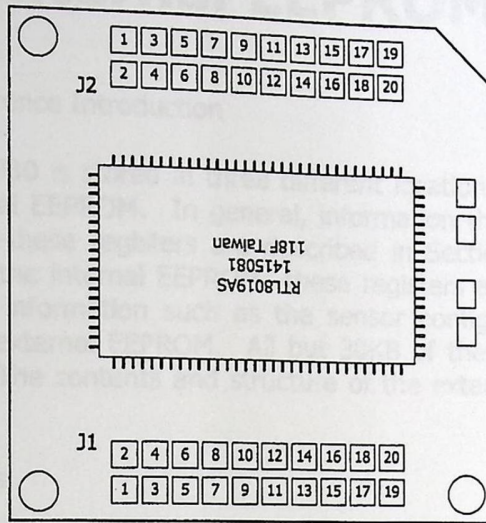


Figure 4-12: IP μ 8930 Module (Bottom View)

PIN	Function	Source @5V	Sink @5V	Description
1	Ethernet LD			Twisted pair differential output (10Mps)
2	Ethernet TPIN-			Twisted pair differential input (10Mps)
3	Ethernet HD			Twisted pair differential output (10Mps)
4	Ethernet TPIN+			Twisted pair differential input (10Mps)
5	+3.3V	35ma		3.3V supply output
6	AUI			RTL chip configuration input
7	GND			Signal Ground
8	GND			Signal Ground
9	RESERVED			RESERVED
10	+5V			Power input to 8930
11	RESERVED			RESERVED
12	+5V			Power input to 8930
13	RESERVED			RESERVED
14	GND			Signal Ground
15	RESERVED			
16	LED0/Collision		25ma	Collision/Link LED (open collector) Normally high. Pulling low will reset the device.
17	MCLR/Master reset			
18	LED1/Receive		25ma	Receive LED (open collector)
19	GND			Signal Ground
20	LED2/Sending		25ma	Send/Transmit LED (open collector)

Figure 4-13: J1 Pin Assignments

5. RAM/Internal EEPROM Reference

5.1 Memory Reference Introduction

Information in the IP μ 8930 is stored in three different locations: internal RAM, an internal EEPROM, and an external EEPROM. In general, information that is accessed frequently is stored in internal RAM—these registers are described in Section 5.2. Most configuration information is stored in the internal EEPROM—these registers are described in Section 5.3. Additional configuration information such as the sensor configuration table and WebHole table are stored in the external EEPROM. All but 30KB of the 512KB external EEPROM is available for user use. The contents and structure of the external EEPROM is described in section 6.

5.2 RAM Variables

Address	Name	Description	Length (Bytes)	Data Type	Valid Values
0x026	IPADR	Current IP Address	4	B	
0x046	TIMER	Timestamp	4	B	Running counter, in units of 8 ms
0x113	PATYPE	I/O Port A Analog/Digital Configuration Byte	1	B	The following combinations of analog and digital configurations are supported. The leftmost column is PA7; the rightmost column is PA0: <div style="text-align: center;"> 76543210 0x80 = AAAAAAAAA (All analog) 0x82 = DDDAAAAA 0x84 = DDDDADAA 0x86 = DDDDDDDD (All digital) 0x89 = DDAAAAAA 0x8E = DDDDDDDA </div>
0x114	PADIR	I/O Port A Directions	1	B	1 = Input Only affects pins 0 = Output configured as digital
0x115	--	Reserved for system use	1	B	
0x116	--	Reserved for system use	1	B	
0x117	PORTA	I/O Port A (digital value)	1	B	Status of Port A pins. Only valid for Digital pins. Analog bits will read as "0"
0x118	I2CTMPL	I2C Temp Value LSB	1	B	Only high bit is valid. If 1, add 0.5C to reading.
0x119	I2CTMPM	I2C Temp Value MSB	1	B	Reading is in degrees C
0x11A	PA0LSB	PA0 LSB	1	B	LSB of analog reading (0-1023)
0x11B	PA0MSB	PA0 MSB	1	B	MSB of analog reading (0-1023)

Address	Name	Description	Length (Bytes)	Data Type	Valid Values
0x11C	PA1LSB	PA1 LSB	1	B	LSB of analog reading (0-1023)
0x11D	PA1MSB	PA1 MSB	1	B	MSB of analog reading (0-1023)
0x11E	PA2LSB	PA2 LSB	1	B	LSB of analog reading (0-1023)
0x11F	PA2MSB	PA2 MSB	1	B	MSB of analog reading (0-1023)
0x120	PA3LSB	PA3 LSB	1	B	LSB of analog reading (0-1023)
0x121	PA3MSB	PA3 MSB	1	B	MSB of analog reading (0-1023)
0x122	PA4LSB	PA4 LSB	1	B	LSB of analog reading (0-1023)
0x123	PA4MSB	PA4 MSB	1	B	MSB of analog reading (0-1023)
0x124	PA5LSB	PA5 LSB	1	B	LSB of analog reading (0-1023)
0x125	PA5MSB	PA5 MSB	1	B	MSB of analog reading (0-1023)
0x126	PA6LSB	PA6 LSB	1	B	LSB of analog reading (0-1023)
0x127	PA6MSB	PA6 MSB	1	B	MSB of analog reading (0-1023)
0x128	PA7LSB	PA7 LSB	1	B	LSB of analog reading (0-1023)
0x129	PA7MSB	PA7 MSB	1	B	MSB of analog reading (0-1023)
0x130-135	--	Reserved for system use	6		
0x136	IPADRA	Current IP Address	15	A	15 bytes, trailing spaces
0x146	SPISTAT	Status byte from SPI chip	1	B	Can be used to determine size of Flash chip.
0x147	I2CCTRL	I ² C Control Byte	1	B	Bit 0 = 1 (Write), 0(Read) Bit 1-3 = Device Address Bits 4-7 = Device Type
0x148	I2LENSTP	I ² C Length & Stop Byte	1	B	Bits: 0-6. Length of data to read or write (max. of MSB: 0=No stop signal at end of I ² C operation; 1=Stop condition sent at end of operation
0x149	I2CIN	I ² C Input	2	B	Input buffer for I ² C operation
0x15A	I2COUT	I ² C Output	2	B	Output buffer for I ² C operation

Key: B=Binary; A=ASCII

Figure 5-1: RAM Variables Sorted By Address

Note on Digital/Analog Pin Assignments:

Digital In/Out is defined as OFF= < 0.5V and ON= >4.5V on for a 0-5V device. No analog output is supported.

Address	Name	Description	Length (Bytes)	Data Type	Valid Values
0x11C	PA1LSB	PA1 LSB	1	B	LSB of analog reading (0-1023)
0x11D	PA1MSB	PA1 MSB	1	B	MSB of analog reading (0-1023)
0x11E	PA2LSB	PA2 LSB	1	B	LSB of analog reading (0-1023)
0x11F	PA2MSB	PA2 MSB	1	B	MSB of analog reading (0-1023)
0x120	PA3LSB	PA3 LSB	1	B	LSB of analog reading (0-1023)
0x121	PA3MSB	PA3 MSB	1	B	MSB of analog reading (0-1023)
0x122	PA4LSB	PA4 LSB	1	B	LSB of analog reading (0-1023)
0x123	PA4MSB	PA4 MSB	1	B	MSB of analog reading (0-1023)
0x124	PA5LSB	PA5 LSB	1	B	LSB of analog reading (0-1023)
0x125	PA5MSB	PA5 MSB	1	B	MSB of analog reading (0-1023)
0x126	PA6LSB	PA6 LSB	1	B	LSB of analog reading (0-1023)
0x127	PA6MSB	PA6 MSB	1	B	MSB of analog reading (0-1023)
0x128	PA7LSB	PA7 LSB	1	B	LSB of analog reading (0-1023)
0x129	PA7MSB	PA7 MSB	1	B	MSB of analog reading (0-1023)
0x130-135	--	Reserved for system use	6		
0x136	IPADRA	Current IP Address	15	A	15 bytes, trailing zeros
0x146	SPISTAT	Status byte from SPI chip	1	B	Can be used to determine size of Flash chip.
0x147	I2CCTRL	I ² C Control Byte	1	B	Bit 0 = 1 (Write) 0 (Read) Bit 1-3 = Device ADDRESS Bits 4-7 = Device Type
0x148	I2LENSTP	I ² C Length & Stop Byte	1	B	Bits: 0-6. Length of data to read or write (max. of 255) MSB: 0=No stop signal at end of operation; 1=Stop condition
0x149	I2CIN	I ² C Input	2	B	Input buffer for I ² C (0-1023)
0x15A	I2COUT	I ² C Output	2	B	Output buffer for I ² C (0-1023)

Key: B=Binary; A=ASCII

Figure 5-1: RAM Variables Sorted by Address

Note on Digital/Analog Pin Assignments:

Digital In/Out is defined as OFF = < 0.5V and ON = > 0.5V output is supported.

Address	Name	Description	Length (Bytes)	Data Type	Valid Values
0x147	I2CTRL	I ² C Control Byte	1	B	Bit 0 = 1 (Write), 0(Read) Bit 1-3 = Device Address Bits 4-7 = Device Type
0x149	I2CIN	I ² C Input	2	B	Input buffer for I ² C operation
0x15A	I2COUT	I ² C Output	2	B	Output buffer for I ² C operation
0x118	I2CTMPL	I ² C Temp Value LSB	1	B	Only high bit is valid. If 1, add 0.5C to reading.
0x119	I2CTMPM	I ² C Temp Value MSB	1	B	Reading is in degrees C
0x148	I2LENSTP	I ² C Length & Stop Byte	1	B	Bits: 0-6. Length of data to read or write (max. of MSB: 0=No stop signal at end of I ² C operation; 1=Stop condition sent at end of operation
0x026	IPADR	Current IP Address	4	B	
0x136	IPADRA	Current IP Address	15	A	15 bytes, trailing spaces
0x11A	PA0LSB	PA0 LSB	1	B	LSB of analog reading (0-1023)
0x11B	PA0MSB	PA0 MSB	1	B	MSB of analog reading (0-1023)
0x11C	PA1LSB	PA1 LSB	1	B	LSB of analog reading (0-1023)
0x11D	PA1MSB	PA1 MSB	1	B	MSB of analog reading (0-1023)
0x11E	PA2LSB	PA2 LSB	1	B	LSB of analog reading (0-1023)
0x11F	PA2MSB	PA2 MSB	1	B	MSB of analog reading (0-1023)
0x120	PA3LSB	PA3 LSB	1	B	LSB of analog reading (0-1023)
0x121	PA3MSB	PA3 MSB	1	B	MSB of analog reading (0-1023)
0x122	PA4LSB	PA4 LSB	1	B	LSB of analog reading (0-1023)
0x123	PA4MSB	PA4 MSB	1	B	MSB of analog reading (0-1023)
0x124	PA5LSB	PA5 LSB	1	B	LSB of analog reading (0-1023)
0x125	PA5MSB	PA5 MSB	1	B	MSB of analog reading (0-1023)
0x126	PA6LSB	PA6 LSB	1	B	LSB of analog reading (0-1023)
0x127	PA6MSB	PA6 MSB	1	B	MSB of analog reading (0-1023)
0x128	PA7LSB	PA7 LSB	1	B	LSB of analog reading (0-1023)
0x129	PA7MSB	PA7 MSB	1	B	MSB of analog reading (0-1023)
0x12A	PA8LSB	PA8 LSB	1	B	Shadow of I2C Temp Sensor
0x12B	PA8MSB	PA8 MSB	1	B	Shadow of I2C Temp Sensor
0x114	PADIR	I/O Port A Directions	1	B	1 = Input Only affects pins 0 = Output configured as digital

Address	Name	Description	Length (Bytes)	Data Type	Valid Values
0x113	PATYPE	I/O Port A Analog/Digital Configuration Byte	1	B	The following combinations of analog and digital configurations are supported. The leftmost column is PA7; the rightmost column is PA0: 76543210 0x80 = AAAAAAAAA (All analog) 0x82 = DDDAAAAA 0x84 = DDDDADAA 0x86 = DDDDDDDD (All digital) 0x89 = DDAAAAAA 0x8E = DDDDDDDA
0x116	PDDIR	I/O Port D Direction (I/O control mask)	1	B	1 = Input Only low 3 bits 0 = Output. are valid
0x117	PORTA	I/O Port A (digital value)	1	B	Status of Port A pins. Only valid for Digital pins. Analog bits will read as "0"
0x146	SPISTAT	Status byte from SPI chip	1	B	Can be used to determine size of Flash chip.
0x046	TIMER	Timestamp	4	B	In units of 8 ms

Key: B=Binary; A=ASCII

Figure 5-2: RAM Variables Sorted By Name

5.3 Internal EEPROM

Address	Name	Contents	Length (Bytes)	Data Type	Valid Values
0x00	MACADR	MAC Address	6	B	
0x06	IPADRDF	Default IP address	4	B	
0x0A	PRDVER	Product/Version String	32	A	
0x2A	DHCPSW	DHCP Enable/Disable	1	B	0 = Disabled 1 = Enabled
0x2B	SECMODE	Secure Mode Enable/Disable	1	B	0 = Disabled 1 = Enabled
0x2C	PACFG	Analog/Digital Config byte	1	B	0 = Analog 1 = Digital
0x2D	PADIR	Port A Direction byte	1	B	0 = Output 1 = Input
0x2F	DEVNAM	Device Name	32	A	User assigned device name
0x4F	DEVLOC	Device Location	32	A	User assigned device location
0x6F	PADF	Power-on value for PA	1	B	Port A power up settings
0x70	SERBAUD	Serial Baud Rate	1	B	129 = 9600 64 = 19,200 42 = 28,800 36 = 33,600 20 = 57,600 10 = 115,200 0 = 1,250,000
0x71	SERCTRL	Serial Flow Control	1	B	0 = RTS/CTS 1 = XON/XOFF 2 = None
0x72	SERCTTO	Serial CTS Timeout	1	B	Units of 39ms (max. 255 = 9.95 secs). This is how long the 8930 waits when the serial device it's talking to has its flow control (CTS) pin on. After this timeout interval, the packet is dropped.
0x73	SERSNTM	Serial Send Timer	1	B	Units of 8ms (max. 255 = 2.04 secs). This is how long the 8930 waits after receiving serial data before sending it over the network.
0x74	TIMER1	Timer1 Interval	2	B	
0x76	TIMER2	Timer2 Interval	2	B	
0x78	FEACON	Feature Configuration	1	B	<u>bit 0</u> : 0=Peripheral Mode; 1=Serial Tunneling Mode <u>bit 1</u> : Analog Read Enable <u>bit 2</u> : Temperature Read Enable <u>bit 3</u> : Event/Action Table Enable <u>bit 4</u> : MCU Command Mode Enable Turning off bits 1-4 can be done to improve performance
0xFA	PASSWD	Password	5	A	Password for secure mode

Key: B=Binary; A=ASCII

Figure 5-3: Internal EEPROM Variables Sorted By Location

Address	Name	Contents	Length (Bytes)	Data Type	Valid Values
0x4F	DEVLOC	Device Location	32	A	User assigned device location
0x2F	DEVNAM	Device Name	32	A	User assigned device name
0x2A	DHCPSW	DHCP Enable/Disable	1	B	0 = Disabled 1 = Enabled
0x78	FEACON	Feature Configuration	1	B	<u>bit 0</u> : 0=Peripheral Mode; 1=Serial Tunneling Mode <u>bit 1</u> : Analog Read Enable <u>bit 2</u> : Temperature Read Enable <u>bit 3</u> : Event/Action Table Enable <u>bit 4</u> : MCU Command Mode Enable Turning off bits 1-4 can be done to improve performance
0x06	IPADRDF	Default IP address	4	B	
0x00	MACADR	MAC Address	6	B	
0x2C	PACFG	Analog/Digital Config byte	1	B	0 = Analog 1 = Digital
0x6F	PADF	Power-on value for PA	1	B	Port A power up settings
0x2D	PADIR	Port A Direction byte	1	B	0 = Output 1 = Input
0xFA	PASSWD	Password	5	A	Password for secure mode
0x0A	PRDVER	Product/Version String	32	A	
0x2B	SECMODE	Secure Mode Enable/Disable	1	B	0 = Disabled 1 = Enabled
0x70	SERBAUD	Serial Baud Rate	1	B	129 = 9600 64 = 19,200 42 = 28,800 36 = 33,600 20 = 57,600 10 = 115,200 0 = 1,250,000
0x71	SERCTRL	Serial Flow Control	1	B	0 = RTS/CTS 1 = XON/XOFF 2 = None
0x72	SERCTTO	Serial CTS Timeout	1	B	Units of 39ms (max. 255 = 9.95 secs). This is how long the 8930 waits when the serial device it's talking to has its flow control (CTS) pin on. After this timeout interval, the packet is dropped.
0x73	SERSNTM	Serial Send Timer	1	B	Units of 8ms (max. 255 = 2.04 secs). This is how long the 8930 waits after receiving serial data before sending it over the network.
0x74	TIMER1	Timer1 Interval	2	B	
0x76	TIMER2	Timer2 Interval	2	B	

Key: B=Binary; A=ASCII

Figure 5-4: Internal EEPROM Variables Sorted By Name

7. Peripherals

7.1 General Purpose I/O Pins

The IP μ 8930 contains one 8-bit port (Port A). PA0 is assigned to Port A's LSB; PA7 to the MSB. These port pins can be configured as input or output. Moreover, this direction can be changed dynamically using the Ipsil Application Interface.

The Port A direction byte, PORTADIR, is located at 0x2D in the IP μ 8930's internal EEPROM. A pin is designated as output if its direction bit is set to 0; conversely, a direction bit of 1 identifies the pin as input. For example, to set pins 0-3 to inputs, and pins 4-7 to outputs on Port A, PORTADIR should be set to 00001111b = 0Fh.

Several of the I/O pins have dual functions. PA6 and PA7 are used for CTS and RTS respectively when hardware flow control is enabled (see section 5.3 under SERCTRL)

The RTS pin can source a maximum of 20ma or sink a maximum of 25 ma. When connecting external devices to these pins, the cable length should not exceed 15 feet. Both inputs and outputs support a 0-5V signal.

It is recommended that a pin be set to output if it is unused.

Each pin can also be designated as digital or analog. If defined as digital, the definition of digital ON is 4.5-5.5V and OFF 0-0.5V.

7.2 Serial Port

The IP μ 8930 has a standard UART serial port, which used to communicate with or control a variety of external serial devices. The signal levels on the RX, TX, RTS, and CTS pins of the 8930 module are 5V TTL levels, not true RS-232 levels. To connect the 8930 to a PC or other standard RS-232 device, a level-shifting device must be used. The IP μ 8930 Developer Board has this chip (e.g., MAX232A) built in, and routed to the RJ-45 serial connector.

The baud rate (SERBAUD) and flow control type (SERCTRL) are user-settable and located in the internal EEPROM (see Figure 5-3). The data bits, parity, and stop bits are fixed at 8,N, and 1.

The IP μ 8930 serial port can operate in one of two modes: Serial Tunneling mode and Peripheral mode.

Serial Tunneling Mode

Serial Tunneling mode sends all data that is received from the serial port to a pre-defined IP address. Similarly, any data that is received to port 8930 with an OPCODE of 8 will be written to the serial port. The Serial Tunneling mode is intended to be used

either with a pair of IP μ 8930-powered devices or with an application written to received messages on port 8930.

The Serial Tunneling mode uses the UDP packet file (File #18—see section 6.7). This file has a series of WebHoles (see section 9) for IP packet length, UDP packet length, and payload data (which is provided by the serial buffer), which are filled dynamically by the IP μ 8930. The destination IP and MAC addresses fields within the UDP packet file must already be filled in with the appropriate data.

The UDP file can be updated using the ICU by uploading a new UDP packet file to file #18, or by using the Serial Settings dialog window. ARP'ing is not performed to resolve IP addresses—therefore, the Serial Tunneling mode can only work on devices on the same subnet (no routers or gateways between devices). Future releases will support serial pass-through connectivity across WANs.

When the Serial Tunneling mode is enabled, any incoming data on the serial port will be sent out using this UDP packet and another packet will not be sent until a UDP "ACK" packet is received by the IP μ 8930, to ensure lockstep operation.

Each packet sent also has a sequence number, to ensure the packets are not interpreted out of sequence, and to detect missing or duplicate packets.

Flow Control/Lost Packet Handling: If no response is received from the partner device after a timeout period (2 seconds), then the last data packet will be resent. This will repeat forever, or until a partner device responds.

Peripheral Mode

The serial port can be accessed from PC or other device by sending Ipsil commands (Op Code 8) over the network to send data out the serial port, and to read the serial input buffer (Op Code 9). This mode works using TCP or UDP, so it is not limited in any way to the same subnet or to any particular MAC address. In this way, any program capable of opening a TCP socket can read and write to the IP μ 8930's serial port over the network.

8. Program Interfaces

8.1 IP μ 8930 Ipsil Control Protocol

The IP μ 8930 Control Protocol (ICP) provides programmatic access to all of the I/O and storage resources on the IP μ 8930 Board including the general I-O pins (a.k.a., Port A), the Serial Port, IP μ 8930 RAM, the internal EEPROM, and the external EEPROM. Read/write access to the SPI and I²C ports is noted here but will not be supported until IP μ 8930 v1.1. Both read and write operations to IP μ 8930 resources are supported.

The ICP is a stateless protocol which supports reading and writing to the IP μ 8930 via port number 8930. Both TCP and UDP are supported. If the IP μ 8930 receives a request via UDP, it will reply to the request using UDP. Similarly, if a TCP request is received, the outgoing packet will be a TCP packet.

Note

The ICP data structures described below should be written as the data contents of a standard TCP or UDP packet.

The flow control for ICP is lock-step. A program must, after writing a packet, wait for a reply before sending another packet.

The ICP packet format consists of several parts. First, every write packet starts with a password, which is required if secure mode is enabled—otherwise it is ignored. An OPCODE and address are next required. The remaining fields carry the information read from or to be written to the IP μ 8930. The contents of the data structure differs based on the operation and whether this is a request or response.

The following tables detail the packet structure of an ICP read, an ICP write and their respective responses:

- Figure 8-1: ICP Packet – Read Request
- Figure 8-2: ICP Packet – Read Response
- Figure 8-3: ICP Packet – Write Request
- Figure 8-4: ICP Packet – Write Response
- Figure 8-5: Reset Device Request
- Figure 8-6: Complete ICP OPCODE List

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	5	PASSWD	Five character ASCII password—always required. Will be ignored if secure mode (SECMODE) is not enabled. If SECMODE is disabled, it is recommended that this field be zero padded.
0x05	1	OPCODE	2 = Read from External EEPROM 4 = Read from Internal EEPROM 7 = Read from RAM location 9 = Read from serial port 10 = I ² C Read (v1.1) 12 = SPI Read (v1.1)
0x06	4	ADDR	Address argument for the above operations. Addresses are in 'big-endian' format, e.g., MSB comes first.
0x0A	1	READLEN	Number of bytes to read. Binary number—must be from 1-32 for UDP packets and 1-20 for TCP packets

Figure 8-1: ICP Packet – Read Request

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	4	ADDR	Address argument for the requested read. Same address as sent in the read request
0x04	4	TIMESTAMP	MSB is first; LSB is last. This is the running count of time since last power up measured in 8 ms increments. These bytes can be used to attached a time reference to the data
0x8	1-32	READDATA	Requested data

Figure 8-2: ICP Packet – Read Response

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	5	PASSWD	Five character ASCII password—always required. Will be ignored if secure mode (SECMODE) is not enabled. If SECMODE is disabled, it is recommended that this field be zero padded.
0x05	1	OPCODE	3 = Write to External EEPROM 5 = Write to External EEPROM 6 = Write to RAM location 8 = Write to serial port 11 = I ² C Write (v1.1) 13 = SPI Write (v1.1)
0x06	4	ADDR	Address argument for the above operations. Addresses are in 'big-endian' format, e.g., MSB comes first.
0x0A	1	WRITEDATA	Data to be written to the above address—must be from 1-30 bytes in length for UDP packets and 1-20 bytes for TCP packets

Figure 8-3: ICP Packet – Write Request

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	1	STATUS	Will be equal to 0xFF if successful. Future release will contain additional error codes.
0x01	1	SEQNUM	Serial Sequence number. This is only present in response to op-codes 8 or 9. It contains the sequence number of the serial command just sent.

Figure 8-4: ICP Packet – Write Response

A programmer may also request that a soft boot be executed on the IP μ 8930. This will cause the IP μ 8930 to revert to its power-up values that are primarily stored in the internal EEPROM.

10. IP μ 8930 Config Utility

10.1 Connecting to the IP μ 8930

The Ipsil Config Utility or ICU is used to configure the IP μ 8930. This section explains how to configure some of the most common IP μ 8930 parameters (e.g., the IP μ 8930 user-assigned name or default IP address).

When you first launch the Config Utility, you will be presented with the following screen (see Figure 10-1). Alternatively, you can also access this screen by selecting the File > Connect menu option.

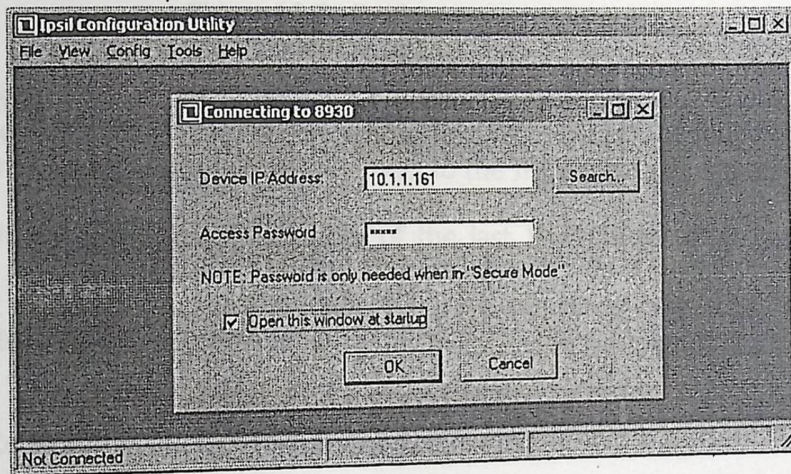


Figure 10-1: Connecting to the IP μ 8930

The default access password is "ipsil".

If you know the IP address of the IP μ 8930 that you want to connect to, you can enter it in the Device IP Address text box and click the OK button. However, if you do not know the IP address for the IP μ 8930 you can click the Search button to search for IP μ 8930s within the same subnet as your computer. If no IP μ 8930 devices are found in the same subnet, an error message will be displayed.

Note

The "Search for IP μ 8930s" feature only lists IP μ 8930s that are within the same subnet as your computer. To access IP μ 8930s using the ICU outside your local subnet, you will need to know the exact IP address of the IP μ 8930 and enter it directly in the *Connect to IP μ 8930* dialog window.

If one or more IP μ 8930s are found, the following *Scan for Ipsil Devices* dialog window will be displayed (see below). This dialog window lists all the IP μ 8930 devices found on the same subnet as your computer.

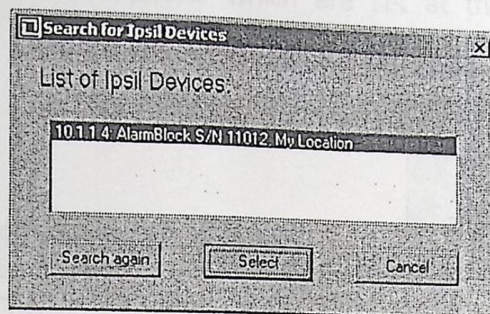


Figure 10-2: Devices Found Message

Click on Select to select an IP μ 8930. This will close the *Scan for Ipsil Devices* window and copy the selected IP address to the *Configure Connection* window. Then click OK in the *Configure Connection* window to connect to the IP μ 8930.

The ICU will remember the IP address of the last IP μ 8930 to which you successfully connected. If you de-select the ***Open this window at startup*** checkbox, the ICU will use the last IP address used when executing any requested operation (e.g., configure IP μ 8930) after startup.

10.2 Configuring the Network Settings

To change the network settings for the IP μ 8930, select the Config > Network menu option in the Ipsil Config Utility. The following dialog window will appear (see below):

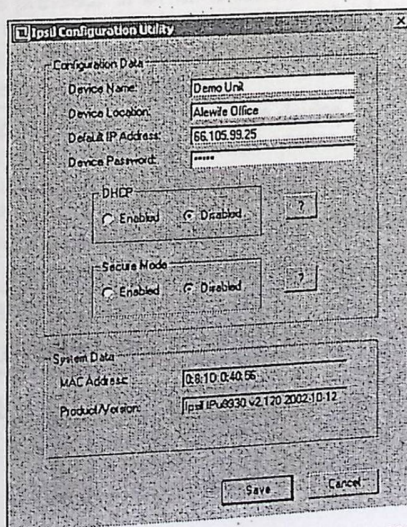


Figure 10-3: Network Configuration Window

The following table describes all the fields in this window. The fields fall into two groups. The *Configuration Data* group are those fields which are updateable by the user and affect the operation of the IP μ 8930. Conversely, the *System Data* group of fields document key data about the IP μ 8930 which are set at the factory and are not changeable.

Field	Description	User Updateable?	Required?	Default Value
<i>8930 Name</i>	User-customizable name (32 characters max.) for IP μ 8930 that can be displayed in sensor reporting web pages and WebLogger software application.	Yes	No	IPu8930
<i>8930 Location</i>	User-customizable location (32 characters max.) for IP μ 8930 that can be displayed in the sensor web pages and WebLogger.	Yes	No	None
<i>Default IP Address</i>	IP address for IP μ 8930 if DHCP is disabled or if the DHCP server cannot be reached when enabled.	Yes	Yes	10.1.1.1
<i>Device Password</i>	When "secure mode" is enabled (see below), this password string must be entered to both Write and write to device (e.g., view sensor web pages, change configuration settings). Max. of 5 chars.	Yes	Yes	ipsil
<i>DHCP</i>	Switch to enable or disable DHCP. When DHCP is enabled, the IP μ 8930 will attempt to obtain an IP address from a local network DHCP server every time it is powered up, or when the hardware reset button is pressed. If DHCP is disabled or a DHCP cannot be reached when the switch is enabled, the pre-defined Default IP Address is used (see above).	Yes	Yes	Enabled
<i>Secure Mode</i>	Switch to set security level for the device. When enabled all Write and write access to the device requires a password (see Device Password). When disabled, no password is required Write or write to the device.	Yes	Yes	Disabled
<i>MAC Address</i>	Unique MAC address for device.	No	Yes	Set at Factory
<i>Product / Version</i>	IP μ 8930 firmware version	No	Yes	Set at Factory

Figure 10-4: Network Configuration Parameters

10.3 Configuring the I/O Channels

To enable/disable I/O features and channel configurations on the IP μ 8930, select the Config > I/O menu option. The following dialog window will appear (see below):

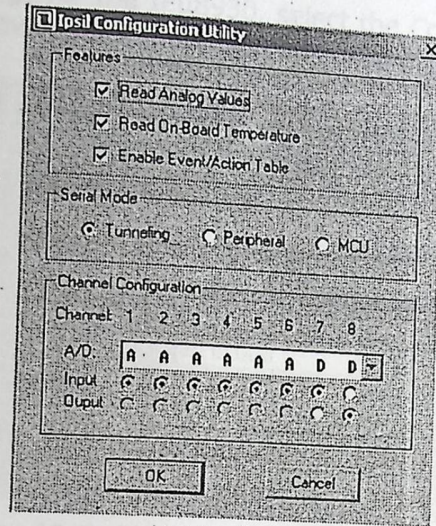


Figure 10-5: I/O Configuration Window

The following table describes all the fields in this window

Field	Description	User Updateable?	Required?	Default Value
<i>Read Analog Values</i>	Defines whether the reading of analog values is enabled	Yes	Yes	Enabled
<i>Read On-Board Temperature</i>	Defines whether the reading of the on-board temperature sensor is enabled	Yes	Yes	Enabled
<i>Enable Event/Action table</i>	Defines whether the event/action table is enabled on the IP μ 8930	Yes	Yes	Enabled
<i>Serial Mode</i>	Defines the serial mode for the IP μ 8930. In the Tunneling mode, two IP μ 8930s can be used to connect two serial devices over the local Ethernet network. In the Peripheral mode, a single IP μ 8930 can be used to send and receive data from a serial device using TCP/IP or UDP calls. In MCU mode, the IP μ 8930 can communicate with other micro-controllers	Yes	Yes	Serial Tunnel
<i>Channel Configuration</i>	Depending on the features selected above, defines the possible combinations of analog/digital channels and input/output channels available in the IP μ 8930	Yes	Yes	

Figure 10-6: I/O Configuration Parameters

10.4 Configuring the Serial Interface

To change the serial settings for the IP μ 8930, select the Config > Serial menu option in the Config Utility. The following dialog window will appear (see below):

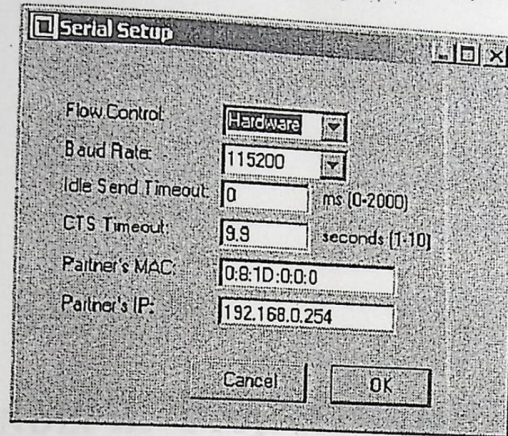


Figure 10-7: Serial Configuration Window

The following table describes all the fields in this window

Field	Description	User Updateable?	Required?	Default Value
<i>Flow Control</i>	Type of flow control used for serial communications. The options are Xon/Xoff, CTS/RTS and None	Yes	Yes	CTS/RTS
<i>Baud Rate</i>	Throughput used for serial communications. The options are 9600, 19200, 28800, 33600, 57600 and 115200 baud	Yes	Yes	115200
<i>Idle Send Timeout</i>	Idle timeout interval on serial interface after which an IP packet that encapsulates the serial data is generated by the IP μ 8930	Yes	Yes	0
<i>CTS Timeout</i>	Timeout interval for CTS	Yes	Yes	9.9
<i>Partner's MAC</i>	MAC address for partner IP μ 8930. Only required when the IP μ 8930 is being used in the serial tunneling mode	Yes	No	
<i>Partner's IP</i>	IP address for partner IP μ 8930. Only required when the IP μ 8930 is being used in the serial tunneling mode	Yes	No	

Figure 10-8: Serial Configuration Parameters

LM741 Operational Amplifier

General Description

The LM741 is a general purpose operational amplifier which offers excellent performance and industry standard pinout. They are direct drop-in replacements for the LM301, LM302, LM303 and 748 in most applications. In addition, they offer many features which make them significantly superior devices. Input protection diodes on the input and

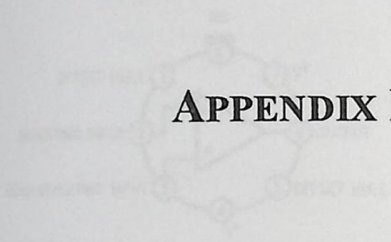
output, so loading when the output mode is not required, as well as protection from oscillation.

The LM741C is identical to the LM741, LM741A except that the LM741C has their performance guaranteed over a DC to 170°C temperature range, instead of -55°C to +125°C.

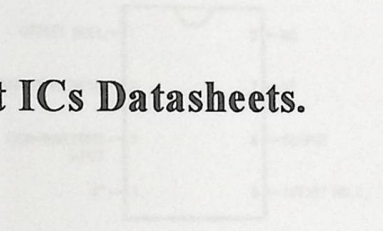
Features

Connection Diagrams

Small Cell Package



Dual In-Line or S.O. Package

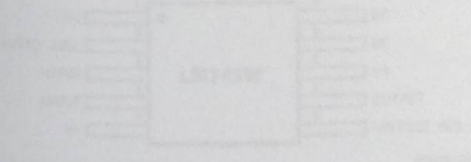


APPENDIX B: Project ICs Datasheets.

Order Number LM741, LM741C, LM741A, LM741B, LM741C, LM741D, LM741E, LM741F, LM741G, LM741H, LM741I, LM741J, LM741K, LM741L, LM741M, LM741N, LM741P, LM741Q, LM741R, LM741S, LM741T, LM741U, LM741V, LM741W, LM741X, LM741Y, LM741Z

Order Number LM741, LM741C, LM741A, LM741B, LM741C, LM741D, LM741E, LM741F, LM741G, LM741H, LM741I, LM741J, LM741K, LM741L, LM741M, LM741N, LM741P, LM741Q, LM741R, LM741S, LM741T, LM741U, LM741V, LM741W, LM741X, LM741Y, LM741Z

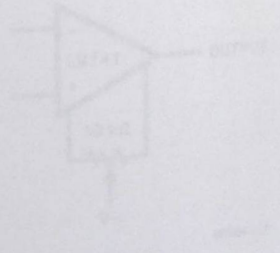
Carrier Package



Order Number LM741, LM741C, LM741A, LM741B, LM741C, LM741D, LM741E, LM741F, LM741G, LM741H, LM741I, LM741J, LM741K, LM741L, LM741M, LM741N, LM741P, LM741Q, LM741R, LM741S, LM741T, LM741U, LM741V, LM741W, LM741X, LM741Y, LM741Z

Typical Application

Offset Nulling Circuit



LM741 Operational Amplifier

General Description

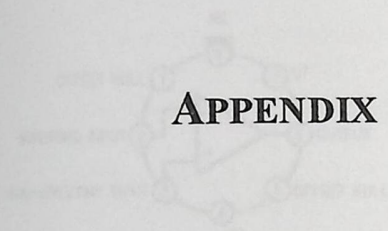
The LM741 is a general purpose operational amplifier with high performance and low standby current. It is ideal for use in instrumentation and data acquisition systems. The LM741 is available in a variety of packages and is designed for use in a wide range of applications. It features a wide bandwidth, high slew rate, and low offset voltage. The LM741 is also available in a low power version, the LM741C.

Output, no locking when the common mode range is exceeded, and will be immune from oscillation. The LM741C is identical to the LM741 except that the LM741C has their performance guaranteed over a 0°C to +70°C temperature range, instead of -55°C to +125°C.

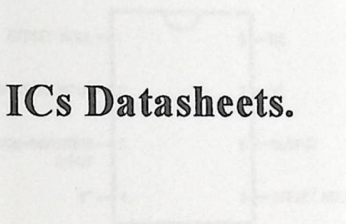
Features

Connection Diagrams

8-Pin DIP Package



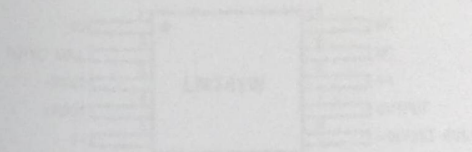
8-Pin DIP or TO-9 Package



APPENDIX B: Project ICs Datasheets.

Order Number LM741, LM741C, LM741N
See NS Package Number 7000, 7001 or 7002

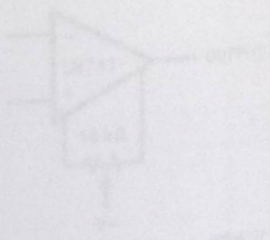
Cartridge Package



Order Number LM741N853
See NS Package Number 7000

Typical Application

Offset Nulling Circuit



LM741 Operational Amplifier

General Description

The LM741 series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439 and 748 in most applications. The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and

output, no latch-up when the common mode range is exceeded, as well as freedom from oscillations.

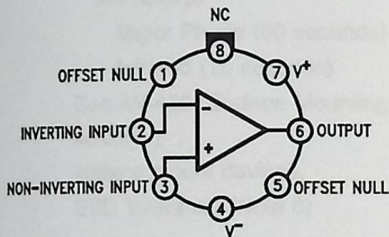
The LM741C is identical to the LM741/LM741A except that the LM741C has their performance guaranteed over a 0°C to +70°C temperature range, instead of -55°C to +125°C.

Features

LM741 Operational Amplifier

Connection Diagrams

Metal Can Package

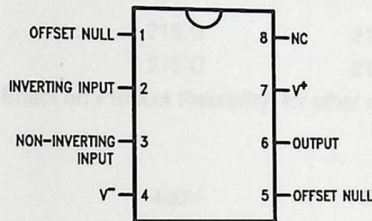


00934102

Note 1: LM741H is available per JM38510/10101

Order Number **LM741H, LM741H/883** (Note 1),
LM741AH/883 or **LM741CH**
See NS Package Number **H08C**

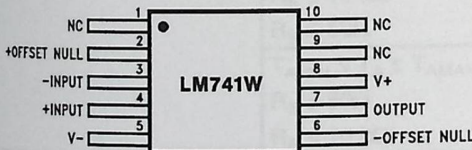
Dual-In-Line or S.O. Package



00934103

Order Number **LM741J, LM741J/883, LM741CN**
See NS Package Number **J08A, M08A** or **N08E**

Ceramic Flatpak

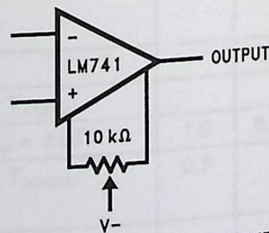


00934106

Order Number **LM741W/883**
See NS Package Number **W10A**

Typical Application

Offset Nulling Circuit



00934107

LM741 Operational Amplifier

General Description

The LM741 series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439 and 748 in most applications. The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and

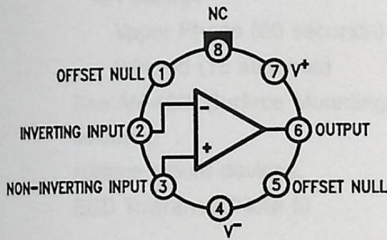
output, no latch-up when the common mode range is exceeded, as well as freedom from oscillations. The LM741C is identical to the LM741/LM741A except that the LM741C has their performance guaranteed over a 0°C to +70°C temperature range, instead of -55°C to +125°C.

Features

LM741 Operational Amplifier

Connection Diagrams

Metal Can Package

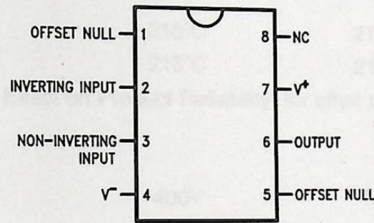


00934102

Note 1: LM741H is available per JM38510/10101

Order Number **LM741H, LM741H/883** (Note 1),
LM741AH/883 or **LM741CH**
See NS Package Number **H08C**

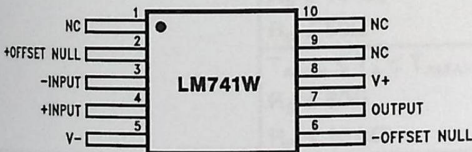
Dual-In-Line or S.O. Package



00934103

Order Number **LM741J, LM741J/883, LM741CN**
See NS Package Number **J08A, M08A** or **N08E**

Ceramic Flatpak

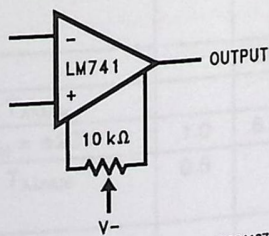


00934106

Order Number **LM741W/883**
See NS Package Number **W10A**

Typical Application

Offset Nulling Circuit



00934107

LM741

Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

(Note 7)

	LM741A	LM741	LM741C
Supply Voltage	±22V	±22V	±18V
Power Dissipation (Note 3)	500 mW	500 mW	500 mW
Differential Input Voltage	±30V	±30V	±30V
Input Voltage (Note 4)	±15V	±15V	±15V
Output Short Circuit Duration	Continuous	Continuous	Continuous
Operating Temperature Range	-55°C to +125°C	-55°C to +125°C	0°C to +70°C
Storage Temperature Range	-65°C to +150°C	-65°C to +150°C	-65°C to +150°C
Junction Temperature	150°C	150°C	100°C
Soldering Information			
N-Package (10 seconds)	260°C	260°C	260°C
J- or H-Package (10 seconds)	300°C	300°C	300°C
M-Package			
Vapor Phase (60 seconds)	215°C	215°C	215°C
Infrared (15 seconds)	215°C	215°C	215°C
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.			
ESD Tolerance (Note 8)	400V	400V	400V

Electrical Characteristics (Note 5)

Parameter	Conditions	LM741A			LM741			LM741C			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage	$T_A = 25^\circ\text{C}$					1.0	5.0		2.0	6.0	mV
	$R_S \leq 10\text{ k}\Omega$		0.8	3.0							mV
	$R_S \leq 50\Omega$										mV
Average Input Offset Voltage Drift	$T_{AMIN} \leq T_A \leq T_{AMAX}$			4.0							mV
	$R_S \leq 50\Omega$						6.0			7.5	mV
	$R_S \leq 10\text{ k}\Omega$										$\mu\text{V}/^\circ\text{C}$
Average Input Offset Current Drift			15								$\mu\text{V}/^\circ\text{C}$
Input Offset Voltage Adjustment Range	$T_A = 25^\circ\text{C}, V_S = \pm 20\text{V}$	±10				±15			±15		mV
Input Offset Current	$T_A = 25^\circ\text{C}$		3.0	30		20	200		20	200	nA
	$T_{AMIN} \leq T_A \leq T_{AMAX}$			70		85	500			300	nA
Average Input Offset Current Drift				0.5							nA/°C
Input Bias Current	$T_A = 25^\circ\text{C}$		30	80		80	500		80	500	nA
	$T_{AMIN} \leq T_A \leq T_{AMAX}$			0.210			1.5		0.3	2.0	nA
Input Resistance	$T_A = 25^\circ\text{C}, V_S = \pm 20\text{V}$	1.0	6.0		0.3	2.0		0.3	2.0		MΩ
	$T_{AMIN} \leq T_A \leq T_{AMAX}, V_S = \pm 20\text{V}$	0.5									MΩ
Input Voltage Range	$T_A = 25^\circ\text{C}$				±12	±13					V
	$T_{AMIN} \leq T_A \leq T_{AMAX}$										V

Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

(Note 7)

	LM741A	LM741	LM741C
Supply Voltage	±22V	±22V	±18V
Power Dissipation (Note 3)	500 mW	500 mW	500 mW
Differential Input Voltage	±30V	±30V	±30V
Input Voltage (Note 4)	±15V	±15V	±15V
Output Short Circuit Duration	Continuous	Continuous	Continuous
Operating Temperature Range	-55°C to +125°C	-55°C to +125°C	0°C to +70°C
Storage Temperature Range	-65°C to +150°C	-65°C to +150°C	-65°C to +150°C
Junction Temperature	150°C	150°C	100°C
Soldering Information			
N-Package (10 seconds)	260°C	260°C	260°C
J- or H-Package (10 seconds)	300°C	300°C	300°C
M-Package			
Vapor Phase (60 seconds)	215°C	215°C	215°C
Infrared (15 seconds)	215°C	215°C	215°C
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.			
ESD Tolerance (Note 8)	400V	400V	400V

Electrical Characteristics (Note 5)

Parameter	Conditions	LM741A			LM741			LM741C			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage	$T_A = 25^\circ\text{C}$ $R_S \leq 10\text{ k}\Omega$ $R_S \leq 50\Omega$		0.8	3.0		1.0	5.0		2.0	6.0	mV mV
	$T_{AMIN} \leq T_A \leq T_{AMAX}$ $R_S \leq 50\Omega$ $R_S \leq 10\text{ k}\Omega$			4.0			6.0			7.5	mV mV
				15							$\mu\text{V}/^\circ\text{C}$
Average Input Offset Voltage Drift						±15			±15		mV
Input Offset Voltage Adjustment Range	$T_A = 25^\circ\text{C}, V_S = \pm 20\text{V}$	±10									nA
Input Offset Current	$T_A = 25^\circ\text{C}$		3.0	30		20	200		20	200	nA
	$T_{AMIN} \leq T_A \leq T_{AMAX}$			70		85	500			300	nA nA/°C
Average Input Offset Current Drift				0.5							nA
Input Bias Current	$T_A = 25^\circ\text{C}$		30	80		80	500		80	500	nA
	$T_{AMIN} \leq T_A \leq T_{AMAX}$			0.210			1.5		0.3	2.0	μA
Input Resistance	$T_A = 25^\circ\text{C}, V_S = \pm 20\text{V}$	1.0	6.0								M Ω
	$T_{AMIN} \leq T_A \leq T_{AMAX}, V_S = \pm 20\text{V}$	0.5							±12	±13	V V
Input Voltage Range	$T_A = 25^\circ\text{C}$				±12	±13					
	$T_{AMIN} \leq T_A \leq T_{AMAX}$										

Electrical Characteristics (Note 5) (Continued)

LM741

Parameter	Conditions	LM741A			LM741			LM741C			Units	
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
Large Signal Voltage Gain	$T_A = 25^\circ\text{C}$, $R_L \geq 2\text{ k}\Omega$ $V_S = \pm 20\text{V}$, $V_O = \pm 15\text{V}$ $V_S = \pm 15\text{V}$, $V_O = \pm 10\text{V}$	50									V/mV	
	$T_{AMIN} \leq T_A \leq T_{AMAX}$, $R_L \geq 2\text{ k}\Omega$, $V_S = \pm 20\text{V}$, $V_O = \pm 15\text{V}$				50	200		20	200		V/mV	
	$V_S = \pm 15\text{V}$, $V_O = \pm 10\text{V}$ $V_S = \pm 5\text{V}$, $V_O = \pm 2\text{V}$	32			25			15			V/mV V/mV V/mV	
Output Voltage Swing	$V_S = \pm 20\text{V}$ $R_L \geq 10\text{ k}\Omega$ $R_L \geq 2\text{ k}\Omega$	± 16									V V	
	$V_S = \pm 15\text{V}$ $R_L \geq 10\text{ k}\Omega$ $R_L \geq 2\text{ k}\Omega$				± 12 ± 10	± 14 ± 13		± 12 ± 10	± 14 ± 13		V V	
	$T_A = 25^\circ\text{C}$	10	25	35		25			25		mA	
	$T_{AMIN} \leq T_A \leq T_{AMAX}$	10		40							mA	
Common-Mode Rejection Ratio	$T_{AMIN} \leq T_A \leq T_{AMAX}$ $R_S \leq 10\text{ k}\Omega$, $V_{CM} = \pm 12\text{V}$				70	90		70	90		dB	
	$R_S \leq 50\Omega$, $V_{CM} = \pm 12\text{V}$	80	95								dB	
Supply Voltage Rejection Ratio	$T_{AMIN} \leq T_A \leq T_{AMAX}$, $V_S = \pm 20\text{V}$ to $V_S = \pm 5\text{V}$										dB	
	$R_S \leq 50\Omega$ $R_S \leq 10\text{ k}\Omega$	86	96		77	96		77	96		dB dB	
Transient Response	$T_A = 25^\circ\text{C}$, Unity Gain	Rise Time	0.25	0.8		0.3			0.3		μs	
		Overshoot	6.0	20		5			5		%	
Bandwidth (Note 6)	$T_A = 25^\circ\text{C}$	0.437	1.5								MHz	
Slew Rate	$T_A = 25^\circ\text{C}$, Unity Gain	0.3	0.7			0.5			0.5		V/ μs	
Supply Current	$T_A = 25^\circ\text{C}$					1.7	2.8		1.7	2.8	mA	
Power Consumption	$T_A = 25^\circ\text{C}$ $V_S = \pm 20\text{V}$ $V_S = \pm 15\text{V}$		80	150							mW mW	
							50	85		50	85	mW mW
	LM741A	$V_S = \pm 20\text{V}$ $T_A = T_{AMIN}$			165							mW
		$T_A = T_{AMAX}$			135							mW
LM741	$V_S = \pm 15\text{V}$ $T_A = T_{AMIN}$					60	100				mW mW	
	$T_A = T_{AMAX}$					45	75				mW	

Note 2: "Absolute Maximum Ratings" indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits.

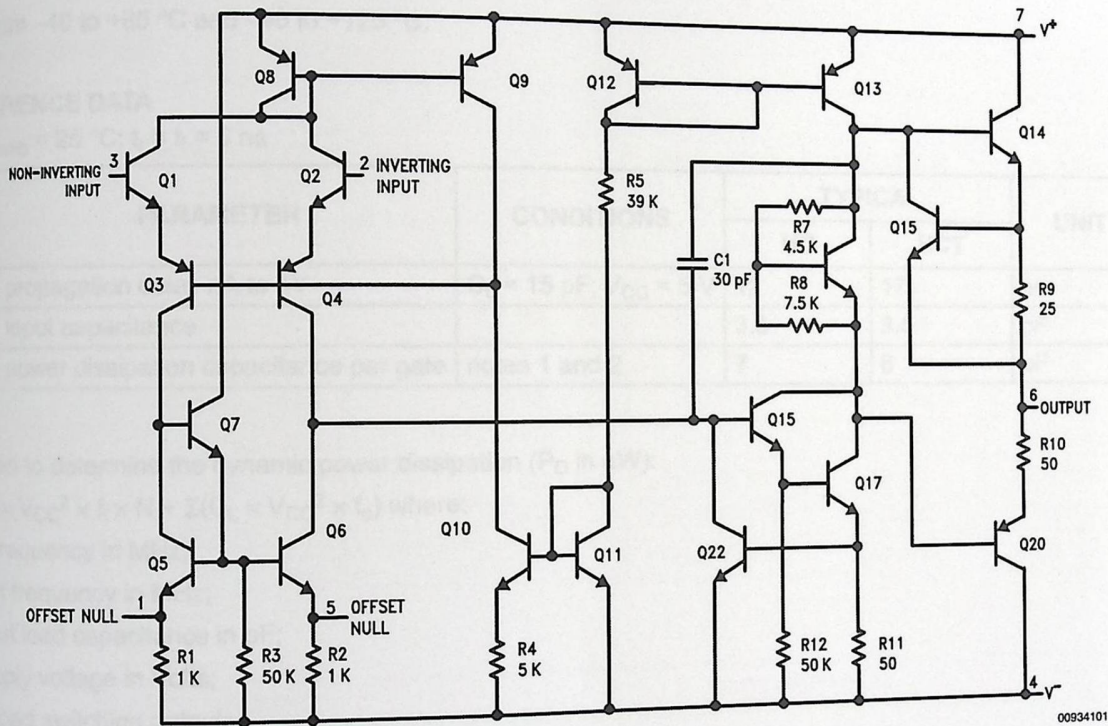
Electrical Characteristics (Note 5) (Continued)

Note 3: For operation at elevated temperatures, these devices must be derated based on thermal resistance, and T_j max. (listed under "Absolute Maximum Ratings"). $T_j = T_A + (\theta_{JA} P_D)$.

Thermal Resistance	Cerdip (J)	DIP (N)	HO8 (H)	SO-8 (M)
θ_{JA} (Junction to Ambient)	100°C/W	100°C/W	170°C/W	195°C/W
θ_{JC} (Junction to Case)	N/A	N/A	25°C/W	N/A

- Note 4: For supply voltages less than $\pm 15V$, the absolute maximum input voltage is equal to the supply voltage.
- Note 5: Unless otherwise specified, these specifications apply for $V_S = \pm 15V$, $-55^\circ C \leq T_A \leq +125^\circ C$ (LM741/LM741A). For the LM741C/LM741E, these specifications are limited to $0^\circ C \leq T_A \leq +70^\circ C$.
- Note 6: Calculated value from: BW (MHz) = $0.35/\text{Rise Time}(\mu s)$.
- Note 7: For military specifications see RETS741X for LM741 and RETS741AX for LM741A.
- Note 8: Human body model, $1.5\text{ k}\Omega$ in series with 100 pF .

Schematic Diagram



00934101

Hex inverting Schmitt trigger

74HC14; 74HCT14

FEATURES

- Applications:
 - Wave and pulse shapers
 - Astable multivibrators
 - Monostable multivibrators.
- Complies with JEDEC standard no. 7A
- ESD protection:
 - HBM EIA/JESD22-A114-A exceeds 2000 V
 - MM EIA/JESD22-A115-A exceeds 200 V.
- Specified from -40 to +85 °C and -40 to +125 °C.

DESCRIPTION

The 74HC14 and 74HCT14 are high-speed Si-gate CMOS devices and are pin compatible with low power Schottky TTL (LSTTL). They are specified in compliance with JEDEC standard no. 7A.

The 74HC14 and 74HCT14 provide six inverting buffers with Schmitt-trigger action. They are capable of transforming slowly changing input signals into sharply defined, jitter-free output signals.

QUICK REFERENCE DATA

GND = 0 V; T_{amb} = 25 °C; t_r = t_f = 6 ns

SYMBOL	PARAMETER	CONDITIONS	TYPICAL		UNIT
			HC	HCT	
t _{PHL} /t _{PLH}	propagation delay nA to nY	C _L = 15 pF; V _{CC} = 5 V	12	17	ns
C _I	input capacitance		3.5	3.5	pF
C _{PD}	power dissipation capacitance per gate	notes 1 and 2	7	8	pF

Notes

1. C_{PD} is used to determine the dynamic power dissipation (P_D in μW):

$$P_D = C_{PD} \times V_{CC}^2 \times f_i \times N + \Sigma(C_L \times V_{CC}^2 \times f_o)$$
 where:
 f_i = input frequency in MHz;
 f_o = output frequency in MHz;
 C_L = output load capacitance in pF;
 V_{CC} = supply voltage in Volts;
 N = total load switching outputs;
 $\Sigma(C_L \times V_{CC}^2 \times f_o)$ = sum of the outputs.
2. For type 74HC14 the condition is V_I = GND to V_{CC}.
 For type 74HCT14 the condition is V_I = GND to V_{CC} - 1.5 V.

Hex inverting Schmitt trigger

74HC14; 74HCT14

FUNCTION TABLE

INPUT	OUTPUT
nA	nY
L	H
H	L

Note

1. H = HIGH voltage level;
L = LOW voltage level.

ORDERING INFORMATION

TYPE NUMBER	PACKAGE				
	TEMPERATURE RANGE	PINS	PACKAGE	MATERIAL	CODE
74HC14D	-40 to +125 °C	14	SO14	plastic	SOT108-1
74HCT14D	-40 to +125 °C	14	SO14	plastic	SOT108-1
74HC14DB	-40 to +125 °C	14	SSOP14	plastic	SOT337-1
74HCT14DB	-40 to +125 °C	14	SSOP14	plastic	SOT337-1
74HC14N	-40 to +125 °C	14	DIP14	plastic	SOT27-1
74HCT14N	-40 to +125 °C	14	DIP14	plastic	SOT27-1
74HC14PW	-40 to +125 °C	14	TSSOP14	plastic	SOT402-1
74HCT14PW	-40 to +125 °C	14	TSSOP14	plastic	SOT402-1
74HC14BQ	-40 to +125 °C	14	DHVQFN14	plastic	SOT762-1
74HCT14BQ	-40 to +125 °C	14	DHVQFN14	plastic	SOT762-1

PINNING

PIN	SYMBOL	DESCRIPTION
1	1A	data input
2	1Y	data output
3	2A	data input
4	2Y	data output
5	3A	data input
6	3Y	data output
7	GND	ground (0 V)
8	4Y	data output
9	4A	data input
10	5Y	data output
11	5A	data input
12	6Y	data output
13	6A	data input
14	V _{CC}	supply voltage

Hex inverting Schmitt trigger

74HC14; 74HCT14

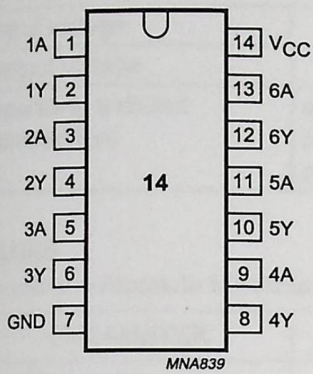
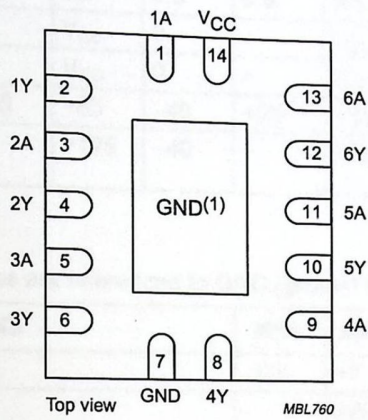


Fig.1 Pin configuration.



(1) The die substrate is attached to this pad using conductive die attach material. It can not be used as a supply pin or input.

Fig.2 Pin configuration DHVQFN14.

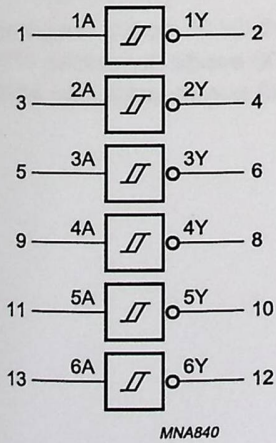


Fig.3 Logic symbol.

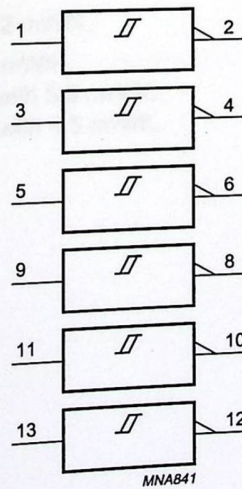


Fig.4 IEC logic symbol.

Hex inverting Schmitt trigger

74HC14; 74HCT14

RECOMMENDED OPERATING CONDITIONS

SYMBOL	PARAMETER	CONDITIONS	74HC14			74HCT14			UNIT
			MIN.	TYP.	MAX.	MIN.	TYP.	MAX.	
V_{CC}	supply voltage		2.0	5.0	6.0	4.5	5.0	5.5	V
V_I	input voltage		0	–	V_{CC}	0	–	V_{CC}	V
V_O	output voltage		0	–	V_{CC}	0	–	V_{CC}	V
T_{amb}	operating ambient temperature	see DC and AC characteristics per device	–40	+25	+85	–40	+25	+85	°C
			–40	–	+125	–40	–	+125	°C

LIMITING VALUES

In accordance with the Absolute Maximum System (IEC 60134); voltages are referenced to GND (ground = 0 V).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CC}	supply voltage		–0.5	+7	V
I_{IK}	input diode current	$V_I < -0.5 \text{ V}$ or $V_I > V_{CC} + 0.5 \text{ V}$	–	±20	mA
I_{OK}	output diode current	$V_O < -0.5 \text{ V}$ or $V_O > V_{CC} + 0.5 \text{ V}$	–	±20	mA
I_O	output source or sink current	$-0.5 \text{ V} < V_O < V_{CC} + 0.5 \text{ V}$	–	±25	mA
I_{CC}, I_{GND}	V_{CC} or GND current		–	50	mA
T_{stg}	storage temperature		–65	+150	°C
P_{tot}	power dissipation	$T_{amb} = -40 \text{ to } +125 \text{ °C}$			
		DIP14 packages; note 1	–	750	mW
		Other packages; note 2	–	500	mW

Notes

- For DIP14 packages: above 70 °C the value of P_D derates linearly with 12 mW/K.
- For SO14 packages: above 70 °C the value of P_D derates linearly with 8 mW/K.
For (T)SSOP14 packages: above 60 °C the value of P_D derates linearly with 5.5 mW/K.
For DHVQFN14 packages: above 60 °C the value of P_D derates linearly with 4.5 mW/K.

LM35 Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of $\pm 1/4^\circ\text{C}$ at room temperature and $\pm 3/4^\circ\text{C}$ over a full -55 to $+150^\circ\text{C}$ temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only $60\ \mu\text{A}$ from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to $+150^\circ\text{C}$ temperature range, while the LM35C is rated for a -40° to $+110^\circ\text{C}$ range (-10° with improved accuracy). The LM35 series is available packaged in

hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear $+ 10.0\ \text{mV}/^\circ\text{C}$ scale factor
- 0.5°C accuracy guaranteeable (at $+25^\circ\text{C}$)
- Rated for full -55° to $+150^\circ\text{C}$ range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than $60\ \mu\text{A}$ current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only $\pm 1/4^\circ\text{C}$ typical
- Low impedance output, $0.1\ \Omega$ for 1 mA load

Typical Applications

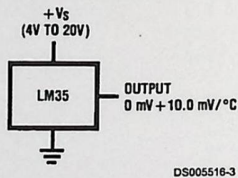
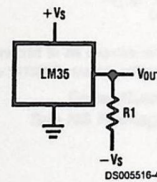


FIGURE 1. Basic Centigrade Temperature Sensor
($+2^\circ\text{C}$ to $+150^\circ\text{C}$)

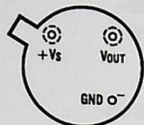


Choose $R_1 = -V_S/50\ \mu\text{A}$
 $V_{\text{OUT}} = +1,500\ \text{mV}$ at $+150^\circ\text{C}$
 $= +250\ \text{mV}$ at $+25^\circ\text{C}$
 $= -550\ \text{mV}$ at -55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

**TO-46
Metal Can Package***



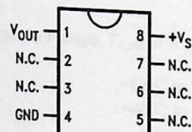
BOTTOM VIEW
DS005516-1

*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH

See NS Package Number H03H

**SO-8
Small Outline Molded Package**

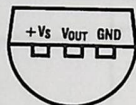


DS005516-21

N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

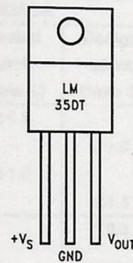
**TO-92
Plastic Package**



BOTTOM VIEW
DS005516-2

Order Number LM35CZ,
LM35CAZ or LM35DZ
See NS Package Number Z03A

**TO-220
Plastic Package***



DS005516-24

*Tab is connected to the negative pin (GND).

Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	+35V to -0.2V
Output Voltage	+6V to -1.0V
Output Current	10 mA
Storage Temp.:	
TO-46 Package,	-60°C to +180°C
TO-92 Package,	-60°C to +150°C
SO-8 Package,	-65°C to +150°C
TO-220 Package,	-65°C to +150°C
Lead Temp.:	
TO-46 Package,	
(Soldering, 10 seconds)	300°C

TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	-55°C to +150°C
LM35C, LM35CA	-40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	$T_A = +25^\circ\text{C}$	± 0.2	± 0.5		± 0.2	± 0.5		$^\circ\text{C}$
	$T_A = -10^\circ\text{C}$	± 0.3			± 0.3		± 1.0	$^\circ\text{C}$
	$T_A = T_{MAX}$	± 0.4	± 1.0		± 0.4	± 1.0		$^\circ\text{C}$
	$T_A = T_{MIN}$	± 0.4	± 1.0		± 0.4		± 1.5	$^\circ\text{C}$
Nonlinearity (Note 8)	$T_{MIN} \leq T_A \leq T_{MAX}$	± 0.18		± 0.35	± 0.15		± 0.3	$^\circ\text{C}$
Sensor Gain (Average Slope)	$T_{MIN} \leq T_A \leq T_{MAX}$	$+10.0$	$+9.9,$ $+10.1$		$+10.0$		$+9.9,$ $+10.1$	mV/ $^\circ\text{C}$
Load Regulation (Note 3) $0 \leq I_L \leq 1$ mA	$T_A = +25^\circ\text{C}$	± 0.4	± 1.0		± 0.4	± 1.0		mV/mA
	$T_{MIN} \leq T_A \leq T_{MAX}$	± 0.5		± 3.0	± 0.5		± 3.0	mV/mA
Line Regulation (Note 3)	$T_A = +25^\circ\text{C}$	± 0.01	± 0.05		± 0.01	± 0.05		mV/V
	$4V \leq V_S \leq 30V$	± 0.02		± 0.1	± 0.02		± 0.1	mV/V
Quiescent Current (Note 9)	$V_S = +5V, +25^\circ\text{C}$	56	67		56	67		μA
	$V_S = +5V$	105		131	91		114	μA
	$V_S = +30V, +25^\circ\text{C}$	56.2	68		56.2	68		μA
	$V_S = +30V$	105.5		133	91.5		116	μA
Change of Quiescent Current (Note 3)	$4V \leq V_S \leq 30V, +25^\circ\text{C}$	0.2	1.0		0.2	1.0		μA
	$4V \leq V_S \leq 30V$	0.5		2.0	0.5		2.0	μA
Temperature Coefficient of Quiescent Current		$+0.39$		$+0.5$	$+0.39$		$+0.5$	$\mu\text{A}/^\circ\text{C}$
Minimum Temperature for Rated Accuracy	In circuit of Figure 1, $I_L = 0$	+1.5		+2.0	+1.5		+2.0	$^\circ\text{C}$
Long Term Stability	$T_J = T_{MAX}$, for 1000 hours	± 0.08			± 0.08			$^\circ\text{C}$

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	+35V to -0.2V
Output Voltage	+6V to -1.0V
Output Current	10 mA
Storage Temp.:	
TO-46 Package,	-60°C to +180°C
TO-92 Package,	-60°C to +150°C
SO-8 Package,	-65°C to +150°C
TO-220 Package,	-65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C

TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	-55°C to +150°C
LM35C, LM35CA	-40°C to +110°C
LM35D	0°C to +100°C

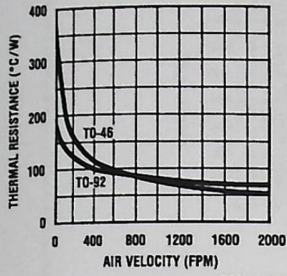
Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	$T_A = +25^\circ\text{C}$	± 0.2	± 0.5		± 0.2	± 0.5	± 1.0	°C
	$T_A = -10^\circ\text{C}$	± 0.3			± 0.3		± 1.0	°C
	$T_A = T_{MAX}$	± 0.4	± 1.0		± 0.4	± 1.0		°C
	$T_A = T_{MIN}$	± 0.4	± 1.0		± 0.4		± 1.5	°C
Nonlinearity (Note 8)	$T_{MIN} \leq T_A \leq T_{MAX}$	± 0.18		± 0.35	± 0.15		± 0.3	°C
Sensor Gain (Average Slope)	$T_{MIN} \leq T_A \leq T_{MAX}$	+10.0	+9.9, +10.1		+10.0		+9.9, +10.1	mV/°C
Load Regulation (Note 3) $0 \leq I_L \leq 1$ mA	$T_A = +25^\circ\text{C}$	± 0.4	± 1.0		± 0.4	± 1.0		mV/mA
	$T_{MIN} \leq T_A \leq T_{MAX}$	± 0.5		± 3.0	± 0.5		± 3.0	mV/mA
Line Regulation (Note 3)	$T_A = +25^\circ\text{C}$	± 0.01	± 0.05		± 0.01	± 0.05		mV/V
	$4V \leq V_S \leq 30V$	± 0.02		± 0.1	± 0.02		± 0.1	mV/V
Quiescent Current (Note 9)	$V_S = +5V, +25^\circ\text{C}$	56	67		56	67		μA
	$V_S = +5V$	105		131	91		114	μA
	$V_S = +30V, +25^\circ\text{C}$	56.2	68		56.2	68		μA
	$V_S = +30V$	105.5		133	91.5		116	μA
Change of Quiescent Current (Note 3)	$4V \leq V_S \leq 30V, +25^\circ\text{C}$	0.2	1.0		0.2	1.0		μA
	$4V \leq V_S \leq 30V$	0.5		2.0	0.5		2.0	μA
Temperature Coefficient of Quiescent Current		+0.39		+0.5	+0.39		+0.5	μA/°C
Minimum Temperature for Rated Accuracy	In circuit of Figure 1, $I_L = 0$	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	$T_J = T_{MAX}$, for 1000 hours	± 0.08			± 0.08			°C

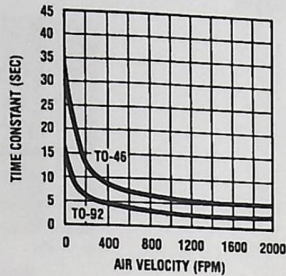
Typical Performance Characteristics

**Thermal Resistance
Junction to Air**



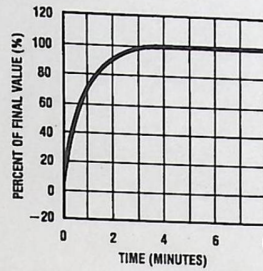
DS005516-25

Thermal Time Constant



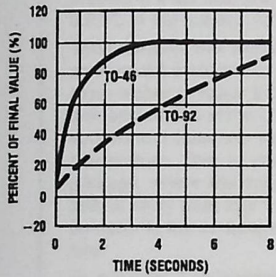
DS005516-26

**Thermal Response
in Still Air**



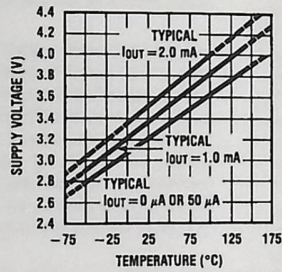
DS005516-27

**Thermal Response in
Stirred Oil Bath**



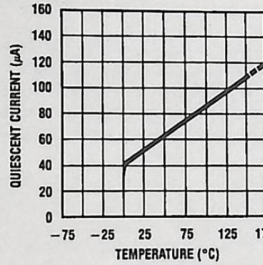
DS005516-28

**Minimum Supply
Voltage vs. Temperature**



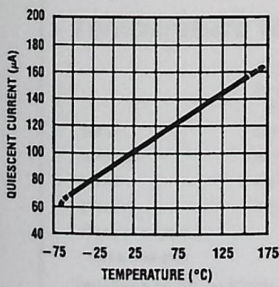
DS005516-29

**Quiescent Current
vs. Temperature
(In Circuit of Figure 1.)**



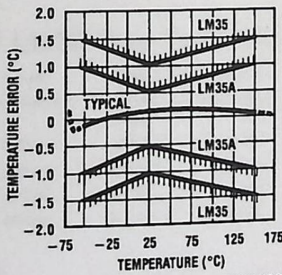
DS005516-30

**Quiescent Current
vs. Temperature
(In Circuit of Figure 2.)**



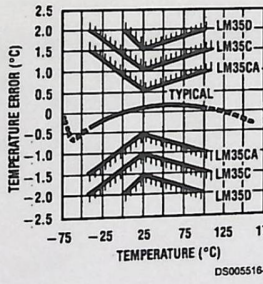
DS005516-31

**Accuracy vs. Temperature
(Guaranteed)**



DS005516-32

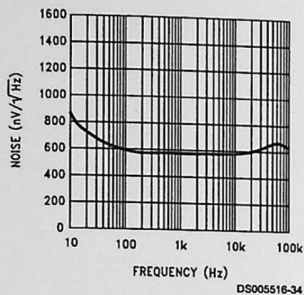
**Accuracy vs. Temperature
(Guaranteed)**



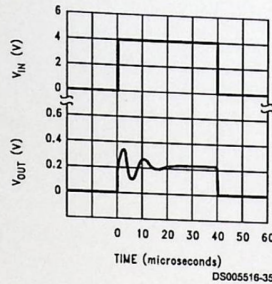
DS005516-33

Typical Performance Characteristics (Continued)

Noise Voltage



Start-Up Response



Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01°C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is especially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V- terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			

(Clamped to metal,
infinite heat sink)

(24°C/W)

(55°C/W)

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.

**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

APPENDIX C: Source Code.

```
import java.io.*;
import java.net.*;
import java.awt.*;
import javax.swing.JOptionPane;
```

```
public class searchNodes extends Thread{
```

```
    // Constants
```

```
    private int IPSIL_PORT = 8930;           // Ipsil device port number
```

```
    // Variables
```

```
    InetAddress ia = null;
```

```
    Socket sock = null;                    // Socket
```

```
    InputStream sIn = null;               // Input data stream
```

```
    OutputStream sOut = null;             // Output data stream instance
```

```
    InetAddress isa;
```

```
    Thread t;
```

```
    byte [] outputData;
```

```
    String IPs[]=new String[20];;
```

```
    byte [] resp = new byte[1024];
```

```
    int foundedIPs=0;
```

```
    int TIME_OUT=1000;
```

```
    public searchNodes() {
```

```
        //set the output data (testing data)
```

```
        byte[] outData =
```

```
        { 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
          0x06,
```

```
          0x00, 0x00, 0x01, 0x17,
```

```
          0x00,
```

```
        };
```

```
        // Password (5 bytes len)
```

```
        // "Write EEPROM" cmd (1 byte len)
```

```
        // Start location (4 bytes len)
```

```
        outputData=outData;
```

```
        t=new Thread(this);
```

```
        t.start();
```

```
    }
```

```
    public void run(){
```

```
        String hostIP="";
```

```
        String networkID="";
```

```
        try {
```

```
            hostIP=InetAddress.getLocalHost().getHostAddress();
```

```
            int index=hostIP.lastIndexOf(".");
```

```

networkID=hostIP.substring(0,index);

} catch (Exception e) {

    JOptionPane.showMessageDialog(null,"Cant find the LAN IP");
}

for (int i = 1; i < 255; i++) {

    nodesFrame.searchProgress.setValue(i);

    String currentIP=networkID+"."+i;

    try{

        ping(currentIP);

        this.sleep(50);

    } catch (Exception e) {
        System.out.println("-->" + e.toString());
    }
}

}

public void ping(String IP){
    try {

        System.out.println("ping "+IP);

        isa= new InetSocketAddress(IP, IPSIL_PORT);

        sock=new Socket();
        sock.connect(isa,TIME_OUT);

        // Create input data stream
        sIn = sock.getInputStream();

        // Create output data stream
        sOut = sock.getOutputStream();

        //set all outputs
        sOut.write(outputData);

        // Read and ignore replay byte from the Ipsil device
        sIn.read(resp);

        if ((resp[0] & 0xff) == 0xff){
            foundedIPs++;
            if (foundedIPs <= 6){
                IPs[foundedIPs-1]=IP;
                nodesFrame.IPArray[foundedIPs-1].setText(IP);
            }
        }
    }
}

```



```

    }catch(Exception e){
        System.out.println(e.toString());
    }
}

}

////////////////////////////////////
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.util.Calendar;

public class readThread extends Thread {

    Calendar cc;
    int interval=1;
    String deviceIP="";

    File f1;
    FileWriter fWrite;
    BufferedWriter writer=null;

    public readThread(int interval,String IP) {
        cc=Calendar.getInstance();

        try {
            String path="C:\\myLog.txt";

            f1=new File(path);
            fWrite=new FileWriter(f1);
            writer=new BufferedWriter(fWrite);
        }catch (Exception e){}

        deviceIP=IP;

        this.interval=interval;

        this.start();

    }

    public void run(){
        int minutesBefore=cc.get(Calendar.MINUTE);
        int minutesAfter=cc.get(Calendar.MINUTE);

        inputData readData;

        //write the time to the file (first line)

```

```

writeToFile(cc.get(Calendar.HOUR_OF_DAY)+":"+cc.get(Calendar.MINUTE));

for (int i = 0; i < 19; ) {

    try {
        cc=Calendar.getInstance();
        minutesAfter=cc.get(Calendar.MINUTE);

        int diff=minutesAfter-minutesBefore;

        if (diff<0) diff+=60;

        if (diff >= this.interval){
            //read all sensors/////
            readData=new inputData(deviceIP,0);

            writeToFile(readData.getTemperature()+"");
            writeToFile(readData.getLights()+"");
            writeToFile(readData.getAlarm()+"");

            minutesBefore=minutesAfter;
            i++;
        }

        this.sleep(1000);
    } catch (Exception e) {

    }

}

System.out.println("Logging done");

try {
    writer.close();
} catch (Exception e) {

}

}

public void writeToFile(String line){

    try {

        writer.write(line);
        writer.write("\r\n");

    } catch (Exception e) {
        System.out.println(e.toString());
    }

}

}

]

```

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
```

```
public class inputData extends Thread{
```

```
    // Constants
    private int IPSIL_PORT = 8930;           // Ipsil device port number
```

```
    // Variables
    InetAddress ia = null;
    Socket sock = null;                     // Socket
    InputStream sIn = null;                 // Input data stream
    OutputStream sOut = null;               // Output data stream instance
    InetSocketAddress isa;
```

```
    Thread t;
    int interval=0;
    byte[] cmdWord;
    byte [] pin7Address;
    byte [] pin6Address;
    byte [] pin4Address;
```

```
    byte [] readPinX;
```

```
    String deviceIP="";
```

```
    static byte [] resp = new byte[1024];   // Network response buffer
```

```
    private static final int TEMPERATURE_SENSOR=0;
    private static final int LIGHT_SENSOR=1;
    private static final int ALARM_SENSOR=2;
```

```
    int currentTemp=0;
    boolean currentLights=false;
    boolean currentAlarm=false;
```

```
    public inputData(String ip) {
```

```
        //get command word from Configurations
        deviceIP=ip;
```

```
        byte [] commandWord =
        { 0x00, 0x00, 0x00, 0x00, 0x00,           // Password
          0x06,                                 // "Write EEPROM" cmd
          0x00, 0x00, 0x01, 0x13,               // Start location
          (byte)0x80,                            // All channels Analog
          (byte)0xff,                            // All channels as inputs
        };
```

```
        cmdWord=commandWord;
```

```
        //temperature sensor at pin 7 --> address 0x128
```

```
        byte[] pin7 =
        { 0x00, 0x00, 0x00, 0x00, 0x00,           // Password
          0x07,                                 // "Read RAM" cmd
          0x00, 0x00, 0x01,0x28,               // pin 8 address
          0x06,                               // read bytes
```

```

};

pin7Address=pin7;

//light sensor at pin 6 --> address 0x126
byte[] pin6 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,      // Password
  0x07,                               // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x26,          // pin 6 address
  0x06,                               // read bytes
};

pin6Address=pin6;

//alarm sensor at pin 4 --> address 0x122
byte[] pin4 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,      // Password
  0x07,                               // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x22,          // pin 4 address
  0x06,                               // read bytes
};

pin4Address=pin4;

t=new Thread(this);
this.interval=1000; //read each 1 second
t.start();

}

public inputData(String ip,int x) {

//get command word from Configurations
deviceIP=ip;

byte [] commandWord =
{ 0x00, 0x00, 0x00, 0x00, 0x00,      // Password
  0x06,                               // "Write EEPROM" cmd
  0x00, 0x00, 0x01, 0x13,          // Start location
  (byte)0x80,                         // All channels Analog
  (byte)0xff,                         // All channels as inputs
};

cmdWord=commandWord;

//temperature sensor at pin 7 --> address 0x128
byte[] pin7 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,      // Password
  0x07,                               // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x28,          // pin 8 address
  0x06,                               // read bytes
};

pin7Address=pin7;

//light sensor at pin 6 --> address 0x126
byte[] pin6 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,      // Password
  0x07,                               // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x26,          // pin 6 address
};

```

```

};

pin7Address=pin7;

//light sensor at pin 6 --> address 0x126
byte[] pin6 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x07,                                // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x26,              // pin 6 address
  0x06,                                // read bytes
};

pin6Address=pin6;

//alarm sensor at pin 4 --> address 0x122
byte[] pin4 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x07,                                // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x22,              // pin 4 address
  0x06,                                // read bytes
};

pin4Address=pin4;

t=new Thread(this);
this.interval=1000; //read each 1 second
t.start();

}

public inputData(String ip,int x) {

//get command word from Configurations
deviceIP=ip;

byte [] commandWord =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x06,                                // "Write EEPROM" cmd
  0x00, 0x00, 0x01, 0x13,              // Start location
  (byte)0x80,                           // All channels Analog
  (byte)0xff,                            // All channels as inputs
};

cmdWord=commandWord;

//temperature sensor at pin 7 --> address 0x128
byte[] pin7 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x07,                                // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x28,              // pin 8 address
  0x06,                                // read bytes
};

pin7Address=pin7;

//light sensor at pin 6 --> address 0x126
byte[] pin6 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x07,                                // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x26,              // pin 6 address
};

```

```

    0x06,          // read bytes
};

pin6Address=pin6;

//alarm sensor at pin 4 --> address 0x122
byte[] pin4 =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x07,          // "Read RAM" cmd
  0x00, 0x00, 0x01, 0x22,          // pin 4 address
  0x06,          // read bytes
};

pin4Address=pin4;

connectMe();
readDataAndStore();

}

public inputData(String ip, byte addressLSB) {

//get command word from Configurations
deviceIP=ip;

byte [] commandWord =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x06,          // "Write EEPROM" cmd
  0x00, 0x00, 0x01, 0x13,          // Start location
  (byte)0x80,          // All channels Analog
  (byte)0xff,          // All channels as inputs
};

cmdWord=commandWord;

//set the reading cmd word
byte[] pin =
{ 0x00, 0x00, 0x00, 0x00, 0x00,          // Password
  0x07,          // "Read RAM" cmd
  0x00, 0x00, 0x01, addressLSB, // set the address of the pin
  0x06,          // read bytes
};

readPinX=pin;

connectMe();
readDataPin();

}

public void connectMe(){
    try {

        isa= new InetSocketAddress(deviceIP, IPSIL_PORT);

        // Init TCP socket. Connect to the device itself on Ipsil port
        sock=new Socket();
        sock.connect(isa,1500); //1.5 second time out
    }
}

```

```

// Create input data stream
sIn = sock.getInputStream();

// Create output data stream
sOut = sock.getOutputStream();

} catch (Exception e) {
    System.out.println(e.toString());
}
}

public void run(){

//check if connects right
connectMe();

while(true){

    try {
        //read date from sensors (if error occurs then break reading)
        if(readDataAndStore())
            mainFrame.displaySensors(currentTemp,currentLights,currentAlarm);
        else
            throw new Exception();

        this.sleep(this.interval);

    } catch (Exception e) {
        javax.swing.JOptionPane.showMessageDialog(null,"Error in reading sensors");
        break;
    }
}

}

public boolean readDataAndStore(){

    try {

        // Send ICP command - Configure channels
        sOut.write(cmdWord);

        // Read and ignore replay byte from the Ipsil device
        sIn.read(resp);

        //*****READ TEMPERATURE PIN 7*****
        // Send ICP command - Read pin 7 (temperature)
        sOut.write(this.pin7Address);

        // Read response from the Ipsil device
        sIn.read(resp);
        storeRecevedData(TEMPERATURE_SENSOR,resp);

        //*****READ LIGHTS PIN 6 *****
        // Send ICP command - Read pin 6 (lights)
        sOut.write(this.pin6Address);

        // Read response from the Ipsil device
        sIn.read(resp);
    }
}
}

```

```

storeReceiviedData(LIGHT_SENSOR,resp);

//*****READ ALARM PIN 4 *****
// Send ICP command - Read pin 4 (alarm)
sOut.write(this.pin4Address);

// Read response from the Ipsil device
sIn.read(resp);
storeReceiviedData(ALARM_SENSOR,resp);

} catch (Exception e) {
    System.out.println(e.toString());
    return false;
}

return true;
}

public void readDataPin0{

    try {

        // Send ICP command - Configure channels
        sOut.write(cmdWord);

        // Read and ignore replay byte from the Ipsil device
        sIn.read(resp);

        //*****READ SPECIFIED PIN *****
        // Send ICP command - Read pin
        sOut.write(this.readPinX);

        // Read response from the Ipsil device
        sIn.read(resp);

        int data=((resp[9]<<8) & 0xff00) | (resp[8] & 0xff);

        mainFrame.setReadValue(data);

    } catch (Exception e) {
        System.out.println(e.toString());
    }

}

private void storeReceiviedData(int sensorID, byte[] response){
    String str="";

    int data=((response[9]<<8) & 0xff00) | (response[8] & 0xff);

    switch (sensorID){
        case TEMPERATURE_SENSOR:
            currentTemp=data;
            break;

        case LIGHT_SENSOR:
            if (data >600)
                currentLights=true;
            else

```



```

        currentLights=false;
        break;

    case ALARM_SENSOR:
        if (data >600)
            currentAlarm=true;
        else
            currentAlarm=false;
        break;

    }

}

public int getTemperature(){
    return setTemperature(currentTemp);
}

public int getLights(){
    if (currentLights)
        return 1;
    else
        return 0;
}

public int getAlarm(){
    if (currentAlarm)
        return 1;
    else
        return 0;
}

public static int setTemperature(int data){

    double fixedData=Math.round(data/10);
    double A=0;
    double B=0;
    int temperature=0;

    if (fixedData > 86 && fixedData<=92){
        A=0.6666666;
        B=-15.333333;
    }

    if (fixedData > 81 && fixedData<=86){
        A=0.4;
        B=7.6;
    }

    if (fixedData > 77 && fixedData<=81){
        A=0.75;
        B=-20.75;
    }

    if (fixedData > 75 && fixedData<=77){
        A=2.0;
        B=-117.0;
    }
}

```

```
if (fixedData > 64 && fixedData<=75){
    A=0.454545455;
    B=-1.090909091;
}

if (fixedData > 53 && fixedData<=64){
    A=0.545454545;
    B=-6.909090909;
}

if (fixedData > 28 && fixedData<=53){
    A=0.56;
    B=-7.68;
}

if (fixedData > 18 && fixedData<=28){
    A=0.5;
    B=-6;
}

//using the selected A,B constants, the temp. can be found by
//temperature = A* fixedData + B
temperature= (int) Math.round(A * fixedData + B);

return temperature;
}
}
```