Palestine Polytechnic University

College of Engineering & Technology

Electrical and Computer Engineering Department

Graduation Project

Remote Controlling and Monitoring of Biological Incubators

Project Team

Suhair Sabarnah **Mirvat Al-Qawasmeh** **Samaher Abd Alraziq**

Project Supervisor

Eng. Yousef Salah

Hebron – Palestine

June-2009

كلية الهندسة والتكنولوجيا

دائرة الهندسة الكهربائية والحاسوب

اسم المشروع

Remote Controlling and Monitoring of Biological Incubators

أسماء الطلبة

مرفت عبد المنعم القواسمة   سهير وحيد صبارنة

سماهر عبد الرازق

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

[illegible] م. يوسف صلاح

....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

....

DEDICATION

To our families for their patience

To our brothers and sisters

ACKNOWLEDGMENT

All praise be to Allah the Ultimate Guide and The Cherisher who gave us the Courage and ability to work this project with a satisfying degree of perfection .The most deserving of our acknowledgments is our respected supervisor **ENG.YOUSEF SALAH** who helped us throughout the research of the present project and for his ideas supports and advices.

Special thanks to the Electro fun Company (hardware supplier) for all its cooperative with us and for all its valuable advices in case of device crash. Last but not least we could also like to thank the cooperative laboratory instructor **Waddah Sultan** for all his valuable advices and time.

Suhair Sabarnah

Mirvat Al-Qawasmeh

Samaher Abd Alraziq

Abstract

Remote Controlling and Monitoring of Biological Incubators

By:

Suhair Sabarnah
Mirvat Al- Qawasmeh
Samahir Abd Alraziq

Palestine Polytechnic University - 2009

Supervisor:
Eng. Yousef Salah

A biological incubator provides controlled environment for newborns needing special care. Managing Staff must walk around and check incubators one by one.

Fortunately remote control technologies incorporated with the World Wide Web have created the possibility of the web-based monitoring system which is important for people who live in remote areas where special care is required, and where geographical distance is a critical.

Thus, the project focuses on the design and implementation of biological incubator that can be accessed remotely to help manage and monitor conditions within the incubator using the Ethernet network to overcome space limitation and to enable immediate response to unusual circumstances.

الملخص

الحاضنات الطبية توفر جواً مناسبا لحديثي الولادة الذين هم بحاجة إلى رعاية خاصة، الفريق المراقب للحاضنات يحتاج إلى التواجد فيزيائيا وفحص كل حاضنة واحدة تلو الآخر.

لحسن الحظ في تكنولوجية التحكم عن بعد التي تتعامل مع الشبكة العالمية العنكبوتية (WWW) أتاحت الفرصة للمراقبة عن بعد، التي تعد مهمة جدا للأشخاص الذين يعيشون في مناطق بعيدة ويلزمهم رعاية خاصة.

لذلك في هذا المشروع يتم التركيز على تصميم وبناء حاضنة طبية بحيث يمكن الوصول إليها للتمكن من مراقبتها والتحكم بمتغيراتها عن بعد من خلال شبكة الانترنت للتغلب على عائق المسافة في الحالات الحرجة.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

Introduction

Chapter One
Introduction

1.1 Overview

Fortunately, Telemedicine is producing a great impact in the monitoring of creatures located in remote nonclinical environments such as homes, military bases, and the like. A number of applications, ranging from data collection, to chronic creature surveillance, and even to the control of therapeutic procedures, are being implemented in many parts of the world.

As part of this growing trend, in this project, we present a real-time remote patient monitoring service through World Wide Web (WWW), which allows physicians to monitor their patient in remote sites using popular Web browser.

The System implements the idea of device controlling through the Web. The system focuses on how devices that connected to a network could be accessed, controlled, and utilized. This criterion is useful to obtain the results a device dedicated to do remotely, as if it installed on front of a user. Basically, the system will control a biological incubator through a web using PIC web server development board.

1.2 Project Objectives

The project aims to design and build electrical system that controls the incubator parameters including (temperature, humidity, and oxygen) by using PIC web development board and special sensors to create the best environment inside the incubator.

1.3 Literature Review

There are several studies and projects working on remote control of biological incubator:

1- **Project Title" Web-based remote monitoring of infant incubators in the ICU".[9]**

This project was introduced and approved in the2003 academic year in the Department of Biomedical Engineering, College of Medicine, Han Yang University.

Project Description:

The aim of study is to help to manage and monitor conditions within incubators in the infant ICU using the hospital network to overcome space limits that restrict available room for controlling equipment and to enable immediate response to unusual circumstance.

A web-based real-time operating, management, and monitoring system for checking temperature and humidity within infant incubators using the Intranet has been developed and installed in the infant Intensive Care Unit (ICU). The project created a pilot system which has a temperature and humidity sensor and a measuring module in each incubator, which is connected to a web-server board via an RS485 port. The system transmits signals using standard web-based TCP/IP so that users can access the system from any Internet-connected personal computer in the hospital.

Using this method, the system gathers temperature and humidity data transmitted from the measuring modules via the RS485 port on the web-server board and creates a web document containing these data. The system manager can maintain centralized supervisory monitoring of the situations in all incubators while sitting within the infant ICU at a work space equipped with a personal computer. The system can be set to monitor unusual circumstances and to emit an alarm signal expressed as a sound or a light on a measuring module connected to the related incubator.

In conclusion, the study presents a solution to the need for centralized supervisory control and management, especially as use of various types of equipment used in hospitals increases in the future.

2- Project Title "CGI-based applications for distributed embedded systems for monitoring temperature and humidity "[10]

This project was introduced and approved in the 2003/2004 academic year in senior university.

Project Description:

Discusses the using of Common Gateway Interface in developing web-based distributed embedded systems. It shows the tree-layer model in developing client-server applications. An example using a BECK microcontroller SC12 in application for monitoring temperature and relative humidity in rooms with controlled microclimate is shown. SC12 is a system-on-chip with embedded RTOS, TCP/IP stack and CGI software.

In conclusion the represented system successfully integrates the functionality of the sensor, GSM gateway and embedded web server and RTOS. The client-server model guarantees the openness of the project and ability to expand.

1.4 Time Schedule

The time plan views the stages in designing and building the system components. This section includes two time schedules; the first one what is done in the first semester while the second shows the task scheduling for the second semester.

Table 1-1 Shows the first semester tasks; all task are referred to the theoretical background and the whole system analysis.

Table 1-1: Time planning (first semester)

Week / **Task**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Project Determination	🟦	🟦	🟦													
Data Gathering				🟦	🟦	🟦										
Design And Analysis							🟦	🟦	🟦							
Documentation				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦

Table 1-2 Shows the second semester tasks; all tasks are referred to the implementation and the whole system testing.

Table 1-2 Time Planning (second Semester).

Task/Week	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Software Design	🟦	🟦	🟦	🟦	🟦	🟦										
Hardware Implementation			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
Integrated System Testing											🟦	🟦	🟦	🟦	🟦	🟦
Documentation				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦

1.5 Estimated Cost

This section lists the overall cost of the project, the cost includes the hardware cost, and the software cost.

Hardware cost: includes the cost of the components that was used to implement the project .Table 1-3 shows these costs.

Table 1-3 Hardware Cost.

Component	**Price**
Pic web development board	250$
Olimex Programmer (PIC-PG1)	30$
Temperature sensor	5$
Humidity Sensor	35$
Heater	30$
Fan	5$
Incubator box	50 $
Cross over cable	10$
Serial Cable	15$
Breadboards &wires	5$
Resistors &Transistors &Relays	30$
Power supply	10$
Other Services(printing , …)	225$
Total	700$

Software Cost: includes the software used to implement the project .Table1-4shows the cost of each component.

Table 1-4: Software Cost

Software	Cost
Software Implementation	200$
Internet Domain	99$
Web Page Design	55$

1.6 Risk Management

The implementation of any project may face many risks during each stage of the project determining, analyzing the system requirements, designing, implementing and testing the whole system .This section illustrates what are the problems which occurred during the implementation.

1.6.1 Hardware Risk

The most important hardware part in the system is the PIC web development board, the risks are:

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

1.6.2 Group Risk

- Illness of one of group members
- Member of team becomes unavailable for any reason.

1.6.3 Requirements 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.

1.6.4 Project Risks:

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

Recovery:

- Taking care when using hardware components and using them according to their specifications.
- Work as team if any mission cannot be performed by any member for any reason .Other members should be helpful.
- Demand device at earlier time.
- Start working on the implementation earlier.
- Use alternate devices with the same functionality and lost cost.
- Try to build our own circuits that can replace unavailable device.

1.7 Report Contents

The documentation of this project is categorized into seven chapters. The followings summarize briefly what each chapter will explain.

- **Chapter one: Introduction**

The chapter demonstrates a general idea about the project and its objective, literature review, time planning, and project risk management.

- **Chapter Two: Theoretical Background**

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

- **Chapter Three: Project Conceptual Design**

The chapter describes the system in its abstract formula .It describes the project objectives, design option and justifies those that are chosen in The projects, a general block diagram, how the system works.

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

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

- **Chapter Five: Software System Design**

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

- **Chapter Six: 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 hardware testing.

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

This chapter will list the problems faced us in accomplishing the system and how did they resolved. Notes and Conclusions will help readers are also included. A future work is also proposed.

CHAPTER TWO

Theoretical Background

2.1 project Components

2.2 Hardware Theoretical background

2.3 Software Theoretical background

CHAPTER TWO

Theoretical Background

Chapter Two
Theoretical Background

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

2.1 Project Components

1. The incubator.
2. Temperature, humidity and Oxygen sensors.
3. Pic web Development Board
4. Required software to operate the system.

2.2 Hardware Theoretical Background

2.2.1 Incubator

A biological incubator is a device used during the care of in-risk creature and designed with the intent of producing environment conditions those suites to each unique creature particular need.

2.2.1.1 What Does It Do?

Biological incubator is used mainly to keep a creature's core temperature stable. Most incubators also humidify the air and can add extra oxygen.

2.2.1.2 How It Works

The mattress where the creature lies is completely enclosed by a clear plastic canopy. The temperature in the incubator is increased by a heater element below the mattress. A motor driven fan near the heater draws in fresh air through a filter and blows it past the heater, warming the air. The air is directed up through slots into the area above the mattress and circulated around. The air temperature is monitored by temperature sensors and is adjusted by controlling the current to the heater.

Supplementary oxygen can be taken in by an oxygen inlet connection where it is mixed with the fresh air through the filter.

The humidity can be increased by the use of water baths (passive humidification) or by dripping water on a heated element (active humidification). The creature is cared for through special access doors called arm ports. [1]

Table 2-1 summarizes the main incubator parameters that are managed and kept track during its operation.

Table 2-1: incubator parameters

Parameters	**Units of measurement**	**Typical values**
Air Temperature	degrees Celsius	32 to 38 C°
Total gas intake:	L/min	35 L/min
Relative humidity	%	50-100%

2.2.1.3 The Air Circulation In The Incubator

The forced air circulation system of the incubator:

- Permits stable temperature control
- Uniform heat distribution
- Humidification.
- Effective isolation of the infant from air born contaminants
- Control of oxygen concentration. [8]

2.2.1.4 Block diagram of Incubator

The following figure shows subpart of the incubator

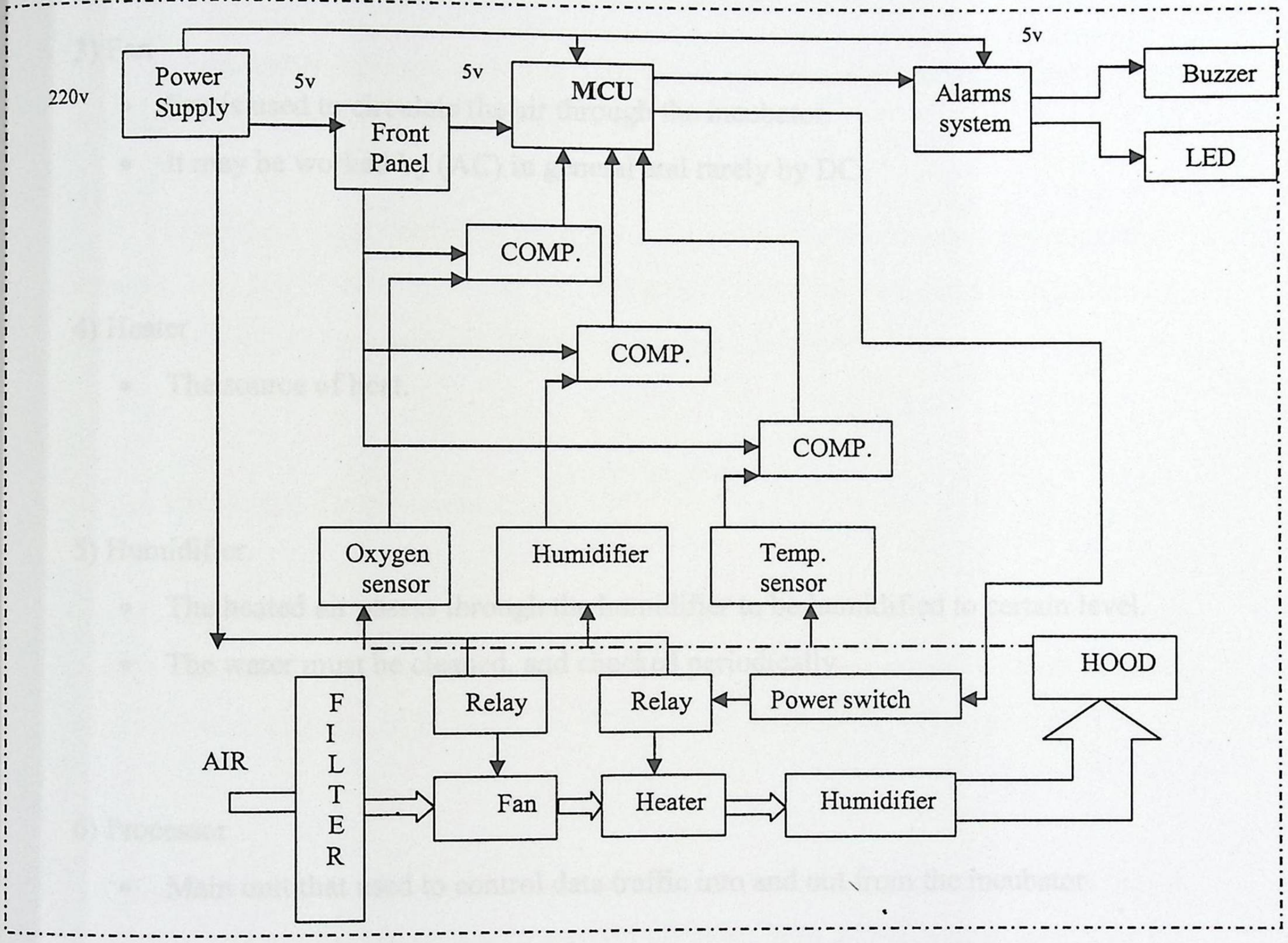


Figure 2-1: Block diagram of incubator

We'll briefly discuss the main characteristics of each subpart of the incubator:

1) Power supply

- Supplies DC voltage to feed the electrical circuit such as 5V, 12V, 18V, 24V for ICs.

2) Filter

- Filters are used for prevention the air born bacteria and any impurities.
- Filters must be exchanged every three months, Six months or at any time needed.

3) Fan

- Fan is used to circulate the air through the incubator.
- It may be worked by (AC) in general and rarely by DC.

4) Heater

- The source of heat.

5) Humidifier.

- The heated air passes through the humidifier to be humidified to certain level.
- The water must be cleaned, and checked periodically.

6) Processor

- Main unit that used to control data traffic into and out from the incubator.

7) Incubator

- The enclosed box which provides the creature with the controlled environment.
- It has doors for the physician in order to take care of the creature.[8]

2.2.2 Ethernet Microcontroller

2.2.2.1 Introduction

Microcontrollers are low-cost embedded systems that control and monitor consumer appliances, robots, machinery, etc.

Enabling a microcontroller to communicate to a ubiquitous data communication network, e.g., the Ethernet network, will allow developers and end-users to monitor and control microcontroller operated devices with greater flexibility.

2.2.2.2 PIC-WEB development board

Is compact board which is supported by Microchip's open source TCP-IP stack AN833 The board is designed with PIC18F452 microcontroller and ENC28J60 Ethernet controller and supports: SLIP, ARP, IP, ICMP, TCP, UDP, HTTP, DHCP, and FTP.

The Microchip stack is written on modular and flexible basis and you can enable or disable modules. The stack also supports dynamic web pages which give you the possibility to control all PIC resources remotely via FTP, HTTP, UDP, TCP etc. The on board 1Mbit serial flash is available for data storage.[7]

Board Feature:

- PIC18F452 microcontroller, ENC28J60 Ethernet controller
- 1Mbit on board serial flash for web pages storage
- ICSP/ICD connector for programming with PIC-MCP, PIC-MCP-USB and Programming and debugging with PIC-ICD2 and PIC-ICD2-POCKET.
- RS232 driver and connector
- Complete web server and TCP-IP stack support as per Microchip's open Source TCP-IP stack
- Voltage regulator +3.3V and filtering capacitors
- Extension header to connect to other boards

2.2.2.3 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).10 Base T is an implementation of Ethernet which allows station to be attached via twisted pair cable .the name 10 base 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 is present on the send and /or receive pair. 10Base T uses RJ45 Jacks.

The 10 Base T Ethernet standards use one wire pair for transition 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 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 10 Base-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).Crossover cable must also be used to connect two internally crossed devices (e.g. two hubs) as the internal crossovers cancel each other out.[11]

2.2.2.4 Serial Cable DB9 M/F

Description: The basic connecting cable for any development board. these cables use to hook any computer with a standard RS232 Serial port to any of our boards. the cable features [11]

- Male to Female
- Wired straight through

2.2.3 Sensors

A Sensor is a device, which responds to an input quantity by generating a functionally related output usually in the form of an electrical or optical signal. Sensor technology is applicable in a very broad domain including the automobiles, machines, medicine, commerce and industry.

The sensing system of the incubator consists of temperature, humidity and Oxygen sensor.

2.2.3.1 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 ±1⁄4°C at room temperature and ±3⁄4°C over full −55 to +150°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 µA 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°C temperature range.[4]

The main features of the LM35 temperature sensor are:

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale facto
- 0.5 °C accuracy guarantee able (at +25°C)
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts

Figure 2-4 Shows the Temperature Sensor

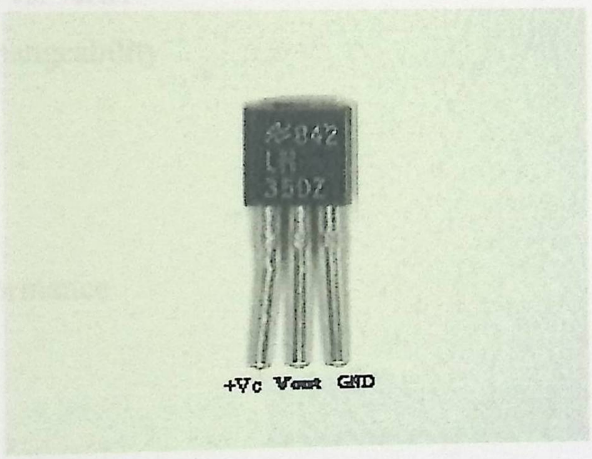


Figure 2-2: Temperature Sensor

2.2.3.2 HIH-4000 Humidity Sensor

The HIH-4000 Series Humidity Sensors are design specifically for high volume OEM (Original Equipment Manufacturer) users. Direct input to a controller or other devices made possible by this sensor's linear voltage output. With a typical current draw of only 200 µA, the HIH-4000 Series is often ideally suited for low drain, battery operated systems. Tight sensor interchangeability reduces or eliminatesOEMproduction calibration costs. Individual sensor calibration data is available.[5]

Feature:

- Molded thermo set plastic housing
- Linear voltage output vs. %RH
- Laser trimmed interchangeability
- Low power design
- High accuracy
- Fast response time
- Stable, low drift performance
- Chemically resistant

Figure 2-5 shows the Humidity Sensor

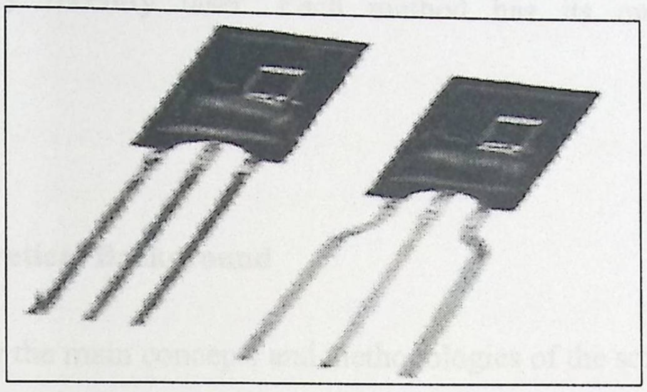


Figure 2-3: Humidity Sensor

2.2.3.3 Oxygen Sensor

An oxygen sensor is an electronic device that measures the proportion of oxygen (O_2) in the gas or liquid being analyzed. The original sensing element is made with a thimble-shaped zirconia ceramic coated on both the exhaust and reference sides with a thin layer of platinum and comes in both heated and unheated forms.

Scientists use oxygen sensors to measure respiration or production of oxygen and use a different approach. Oxygen sensors are used in oxygen analyzers which find a lot of use in medical applications such as anesthesia monitors, respirators and oxygen concentrators.

There are many different ways of measuring oxygen and these include technologies such as zirconia, electrochemical (also known as Galvanic), infrared, ultrasonic and very recently laser. Each method has its own advantages and disadvantages. [6]

2.3 Software Theoretical Background

We'll clarify the main concepts and methodologies of the software tools and protocols employed to perform the communication, interfacing and accessing aspects of the system.

2.3.1 The Microchip TCP/IP Stack

Is a suite of programs that provides services to standard TCP/IP-based applications (HTTP Server, Mail Client, etc.), or can be used in a custom TCP/IP-based application.

The Microchip TCP/IP Stack is implemented in a modular fashion, with all of its services creating highly abstracted layers. Potential users do not need to know
All the intricacies of the TCP/IP specifications to use it.

2.3.1.1 Stack Architecture

Many TCP/IP implementations follow a software architecture referred to as the "TCP/IP Reference model". Software based on this model is divided into multiple Layers, where layers are stacked on top of each other (thus the name "TCP/IP Stack") and each layer accesses services from one or more layers directly below it. A simple version of the TCP/IP Stack model is shown in Figure 2-4.

Per specifications, many of the TCP/IP layers are "live", in the sense that they do not only act when a service is requested but also when events like time-out or new packet arrival occurs. A system with plenty of data memory and program memory can easily incorporate these requirements.[3]

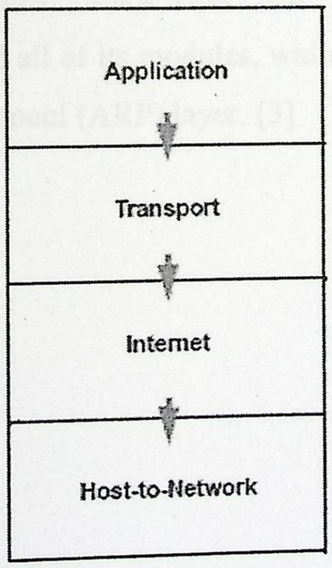


Figure2.4: layers of the TCP/IP reference model [3]

2.3.1.2 Stack Layers

Like the TCP/IP reference model, the Microchip TCP/IP Stack divides the TCP/IP Stack into multiple layers (Figure 2.5). The code implementing each layer resides in a separate source file, while the services and APIs (Application Programming Interfaces) are defined through header/include files.

Unlike the TCP/IP reference model, many of the layers in the Microchip TCP/IP Stack directly access one or more layers which are not directly below it. A decision as to when a layer would bypass its adjacent module for the services it needs, was made primarily on the amount of overhead and whether a given service needs intelligent processing before it can be passed to the next layer or not.

An additional major departure from traditional TCP/IP Stack implementation is the addition of two new modules: "Stack Task" and "ARP Task". Stack Task manages the operations of the stack and all of its modules, while ARP Task manages the services of the Address Resolution Protocol (ARP) layer. [3]

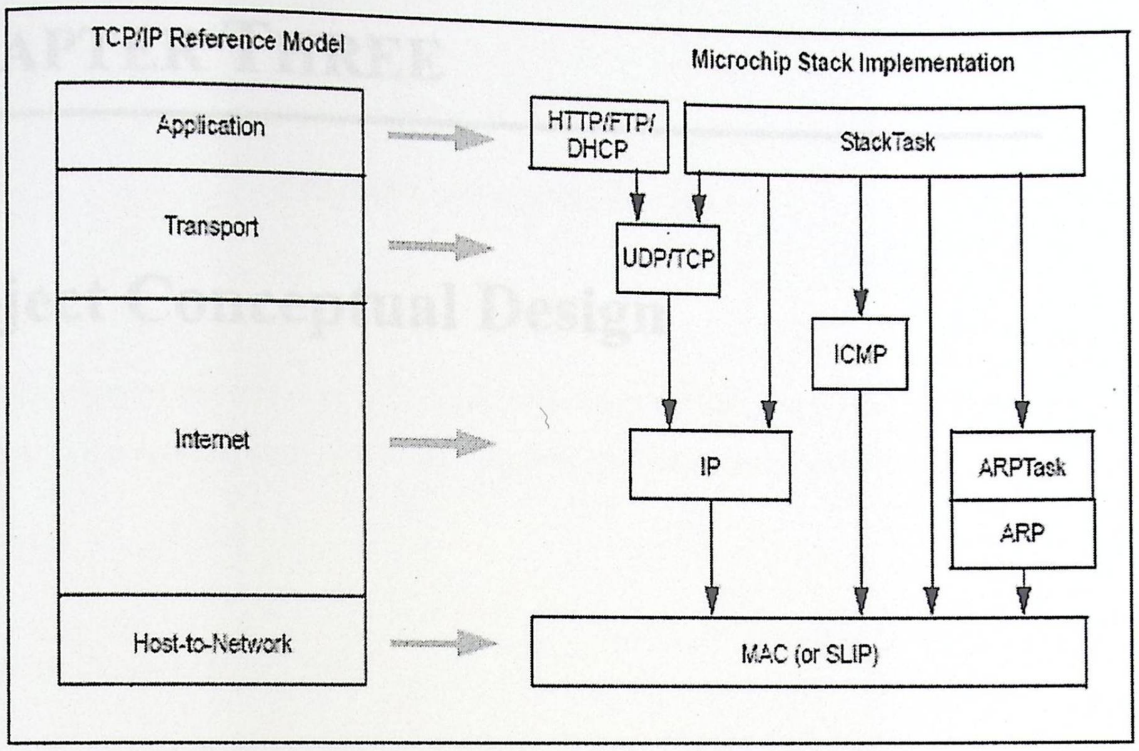


Figure 2.5: Comparing the microchip TCP/IP stack structure to the TCP/IP reference model [3]

CHAPTER THREE

Project Conceptual Design

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 the system works.

3.1 Detailed Project Objectives

1. Design and implement a biological incubator with the sensing circuits (Temperature, humidity, Oxygen .and the control circuits (Fan, Heater, Buzzer and Humidifier).

2. Simplify human daily activities by controlling some of them remotely. And to create a reliable and sensitive system .

3. The ability to program the Pic web development board and accessing its components and peripherals.

4. Design a web page to enable the user to monitor and control, the incubator parameters using web browser from any client machine that has an internet connection.

5. To provide a suitable alarm to the incubator administrator when certain malfunctionalities come up, and so to perform the proper procedures.

3.2 Hardware Design Options

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

Option 1: Using GPRS

Connect the incubator with its sensor system to the network by mobile phone by using GPRS technology. **GPRS (General Packet Radio Service)** is a method of enhancing 2G phones to enable them to send and receive data more rapidly.

With a GPRS connection, the phone is "always on" and can transfer data immediately. GPRS is now available on most new phones. But this method will need programming the mobile that look acceptable for monitoring but very hard for controlling the parameter in the incubator system.

Option 2: Using Bluetooth

Use the Bluetooth techniques to reach the incubator parameter. Bluetooth is a wireless communication protocol which uses radio waves to communicate on short distances.

For the latter characteristic, this is not the needed techniques the project attempt to reach the incubator system over the way by the internet.

Option 3: Using an Ethernet Microcontroller

Use the Ethernet microcontroller this technique will assign the incubator system an IP address and so we can reach from any where, then the monitoring will be easy and then the programming of the controlling process will be achieved.

3.2.1 Ethernet Microcontroller Options

There was a list of candidate commercial Ethernet Microcontrollers in the market. Here, a description of each one is discussed briefly.

Option1: use PIC SAM9L9260 Ethernet microcontroller

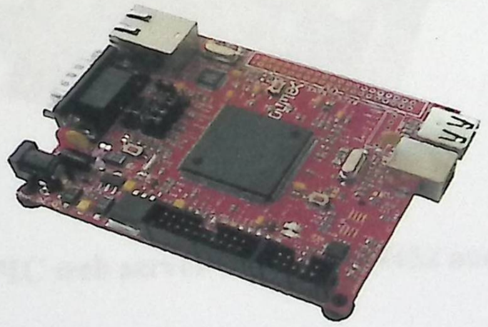


Figure 3-1: PIC SAM9L9260 Ethernet microcontroller

SAM9-L9260 is a low cost development platform with ARM9microcontroller, 64MB SDRAM and 512MB NAND Flash. The board has Ethernet 100Mbit controller, USB host, USB device, RS232 and a 40 pin extension port with all unused SAM9260 ports available for add-on boards.

SAM9-L9260 has vast amounts of Flash and RAM and runs Linux, Windows CE and other RTOS natively. There is an on-board RTC clock with 3V Li battery backup.

At first semester we studied all of the features of that board, 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. [2]

1-Option2: Use the Pic Web Server with PIC18F452 and ENC28J60

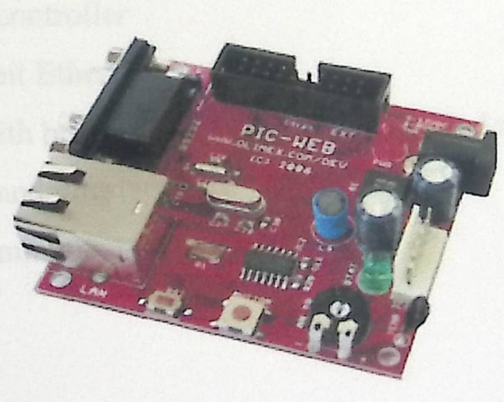


Figure 3-2 PIC web server with PIC18f452 and enc28j6

PIC-WEB is compact board which is supported by Microchip's open source TCP-IP stack AN833.The board is designed with PIC18F452 microcontroller and

ENC28J60 Ethernet controller and support: SLIP, ARP, IP, ICMP, TCP, UDP, HTTP, DHCP, and FTP.

The Microchip stack is written very modular and flexible and you can enable or disable modules and supports dynamic web pages which give you the possibility to control all PIC resources remotely via FTP, HTTP, UDP, TCP etc.

With this board you can implement web and server, send e-mails and almost everything what a big server can do. The on board 1Mbit serial flash is available for data storages. And so for this features we decided to use it. [7]

The Module Features Are:

- PIC18F452 microcontroller
- ENC28J60 10 M bit Ethernet controller
- RJ45 connector with build in Ethernet transformer and two status LEDs
- RS232 connector and interface
- ICSP for programming and debugging

3.3 General Block Diagram

The following diagrams define the general block diagram of the system:

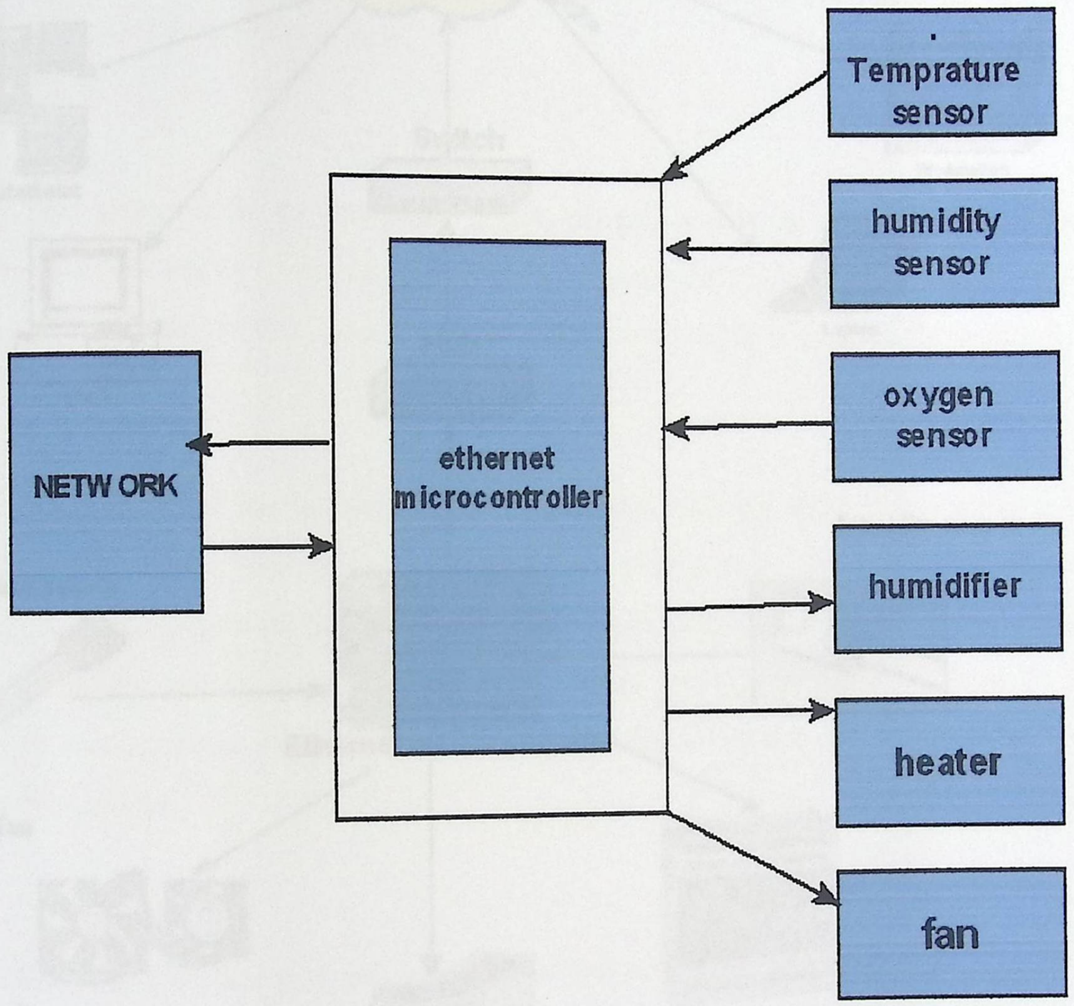


Figure 3-3: Incubator system general block diagram

The following is detailed block diagram

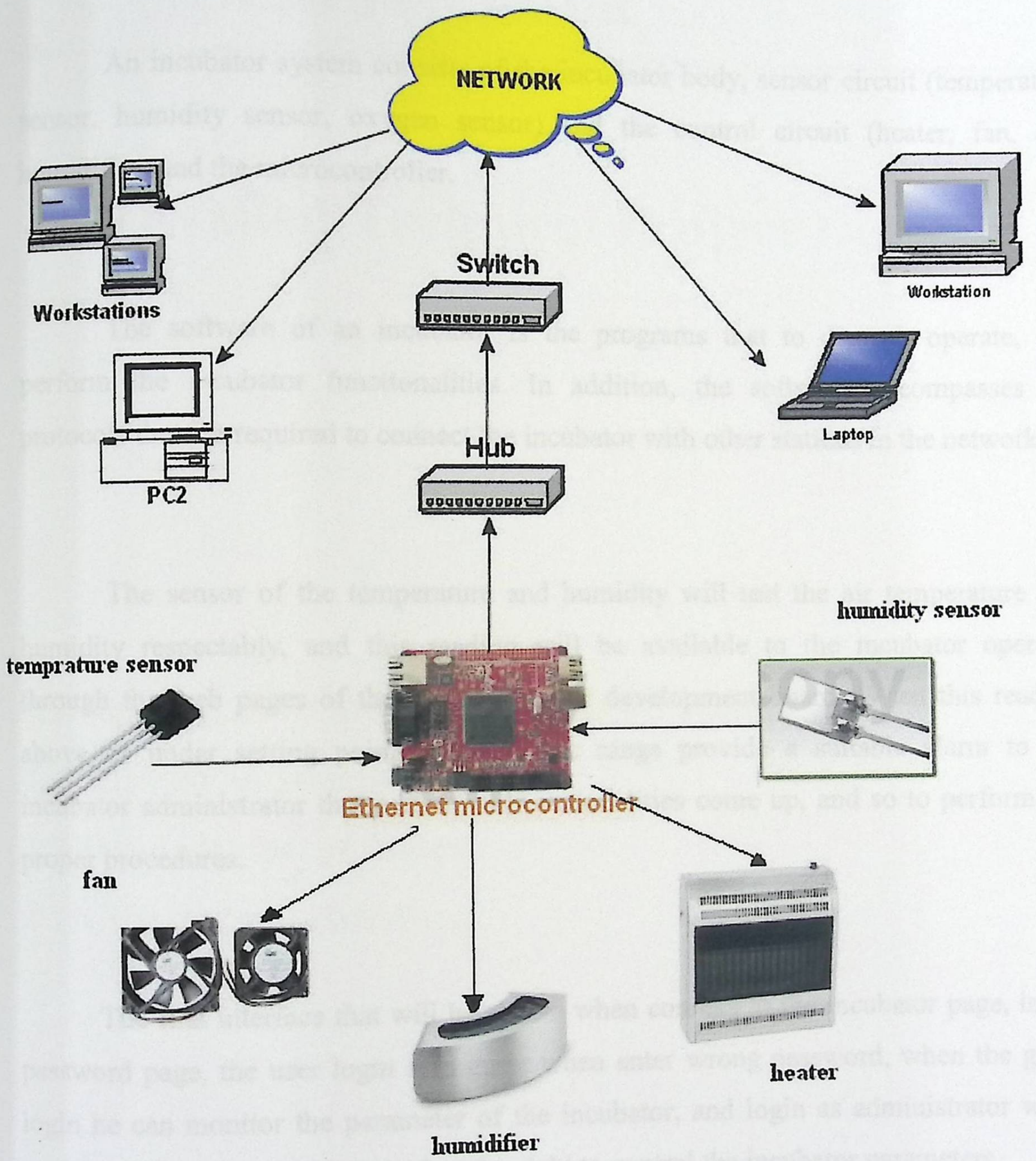


Figure 3-4: Detailed Block Diagram

3.4 How does System Work?

An incubator system consists of the incubator body, sensor circuit (temperature sensor, humidity sensor, oxygen sensor) and the control circuit (heater, fan, and humidifier) and the microcontroller.

The software of an incubator is the programs that to control, operate, and perform the incubator functionalities. In addition, the software encompasses the protocols that are required to connect the incubator with other stations in the network.

The sensor of the temperature and humidity will test the air temperature and humidity respectably, and this reading will be available to the incubator operator through the web pages of the pic web server development board. When this reading above or under setting point with specific range provide a suitable alarm to the incubator administrator that certain malfunctionalities come up, and so to perform the proper procedures.

The first interface that will be shown when connect to the incubator page, is the password page, the user login as a guest when enter wrong password, when the guest login he can monitor the parameter of the incubator, and login as administrator when enter correct password where he has the right to control the incubator parameters.

CHAPTER FOUR

Detailed Technical Project Design

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:

In this phase the PIC web development board was programmed to read the output of the sensing circuits and depending on this read turning on/off the heater, the humidifier.

Phase 2:

For data to be spread after being read from the sensing circuits, the pic web development board was connected to a network via hub and switch, so every PC in this network is able to read this data.

Phase 3:

To provide a suitable alarm to the incubator administrator when certain malfunctionalities come up, and so to perform the proper procedures.

4.2 Subsystem Detailed Design

4.2.1 PIC WEB (PIC18F452)

PIC WEB board use PIC18F452 which contains 8 channels (AN0-AN7) 10-bit Analog-to-Digital Converter and five Controllers (PORTA, PORTB, PORTC, PORTD, and PORTE). Each sensing circuit and other circuit (heater, fan, and humidifier) were connected to these pins as follow:

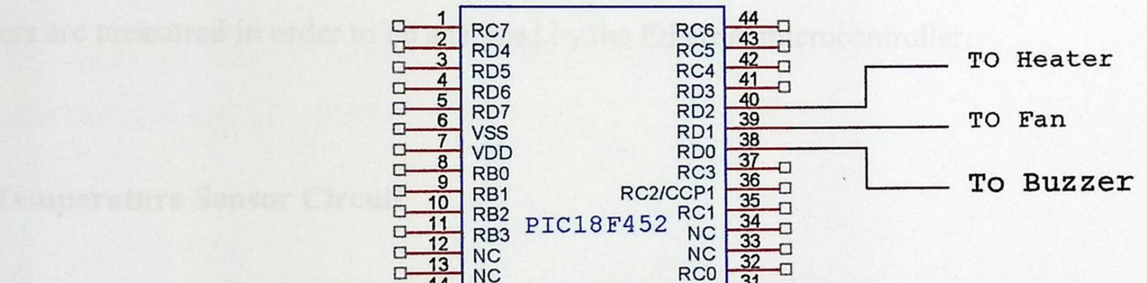


Figure 4-1: I/O pins of PIC WEB.

1. For LM35 temperature sensor circuit the output was directly connected to pin RA3 in PIC WEB.
2. For humidity sensor circuit the output was directly connected to pin RA5 in PIC WEB.

٣. For fan circuit we took the output from resister 20k ohm and connected it to pin RD1 in PIC WEB.

٤. For heater circuit we took the output from resister 15k ohm and connected it to pin RD2 in PIC WEB.

٥. For buzzer circuit we took the output from resister 20k ohm and connected it to pin RD0 in PIC WEB.

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 Ethernet microcontroller.

4.2.2.1 Temperature Sensor Circuit

The reason for choosing the LM35 temperature sensor stems from the following features of this sensor:

1- The change in voltage value is linear to heat. So it's easy to guarantee that the read values at various temperatures will be reasonable by knowing the voltage to heat ratio.

2- Low cost and availability.

Figure 4-2 shows the schematic circuit of temperature sensing, the LM 35 temperature sensor is configured to work on 5 volts power supply, this sensor is directly connected to the pic web development board A/D extension pin, the value of the measured temperature is converted internally to digital value.

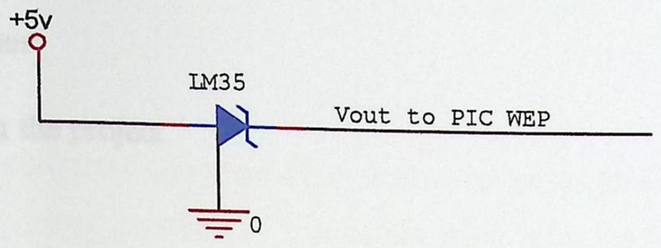


Figure4-2: Temperature sensor circuit

4.2.2.2 Humidity Sensor HIH-4000

We chose the humidity sensor HIH-4000 because it was designed for sensitive applications and it is available in the market.

Figure 4-3 shows the schematic circuit of humidity sensing, the humidity sensor is configured to work on 5 volts power supply, this sensor is directly connected to pic web development board A/D extension pin, the value of the measured humidity is converted internally to digital value.

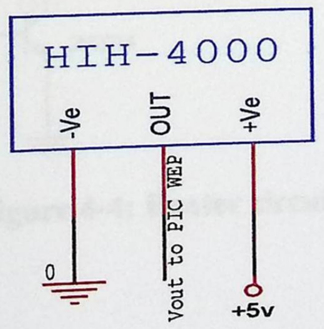


Figure 4-3: Humidity sensor circuit.

4.2.2.3 Oxygen Sensor

Not included in the project.

4.2.3 Heater circuit

In order to control the operation of incubator, heater circuit that shown in figure 4-4was designed. This circuit consists of a normally open relay, and 2N2219 transistor work as a switch , when the pin of the pic gives a 5 volt, the transistor turn to the saturation mode (close switch) , this closing for the switch connect the ground to the circuit ,then the voltage reach to the relay ,so the normally open relay close which close the circuit , so the 220V reach the heater circuit cause it to operate.

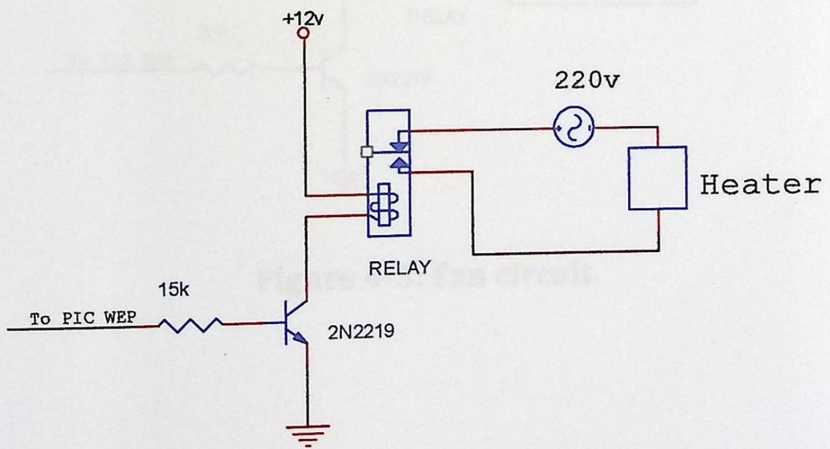


Figure 4-4: Heater circuit.

4.2.4 Fan circuit

In order to control the operation of incubator, fan circuit that shown in figure 4-5 was designed. This circuit consists of a normally open relay, and 2N2219 transistor work as a switch, when the pin of the pic gives a 5 volt, the transistor turn to the saturation mode (close switch) , this closing for the switch connect the ground to the circuit ,then the voltage reach to the relay ,so the normally open relay close which close the circuit so the 12V reach the fan circuit cause it to operate.

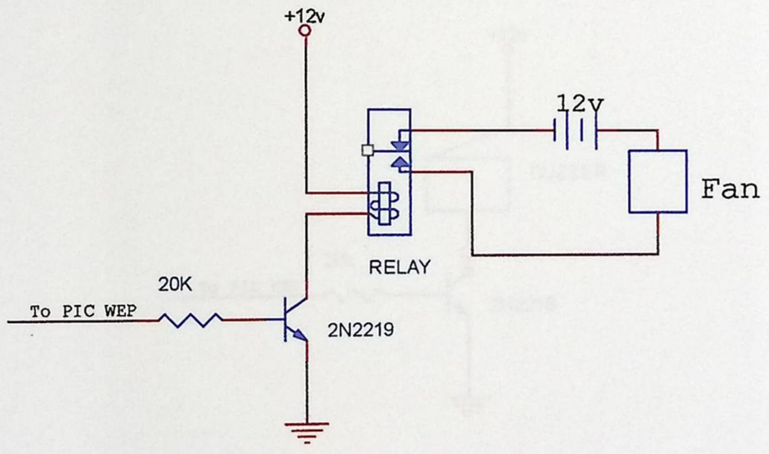


Figure 4-5: Fan circuit.

4.2.5 Buzzer circuit

In order to control the operation of buzzer circuit that shown in figure 4-6 was designed This circuit consists of 2N2219 transistor, the transistor work as a switch, when the pin of the pic gives a 5 volt, the transistor turn to the saturation mode (close switch) , this closing for the switch connect the ground to the circuit, then the voltage reach the circuit ,so the 12V reach the buzzer circuit cause it to operate.

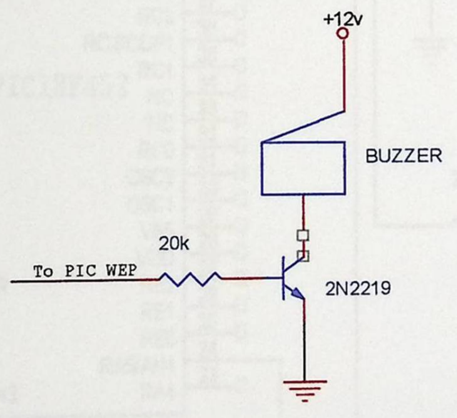


Figure 4-6: Buzzer circuit.

٤,٣ Overall system design

Figure (4-7) shows how the hardware components are connected to each other.

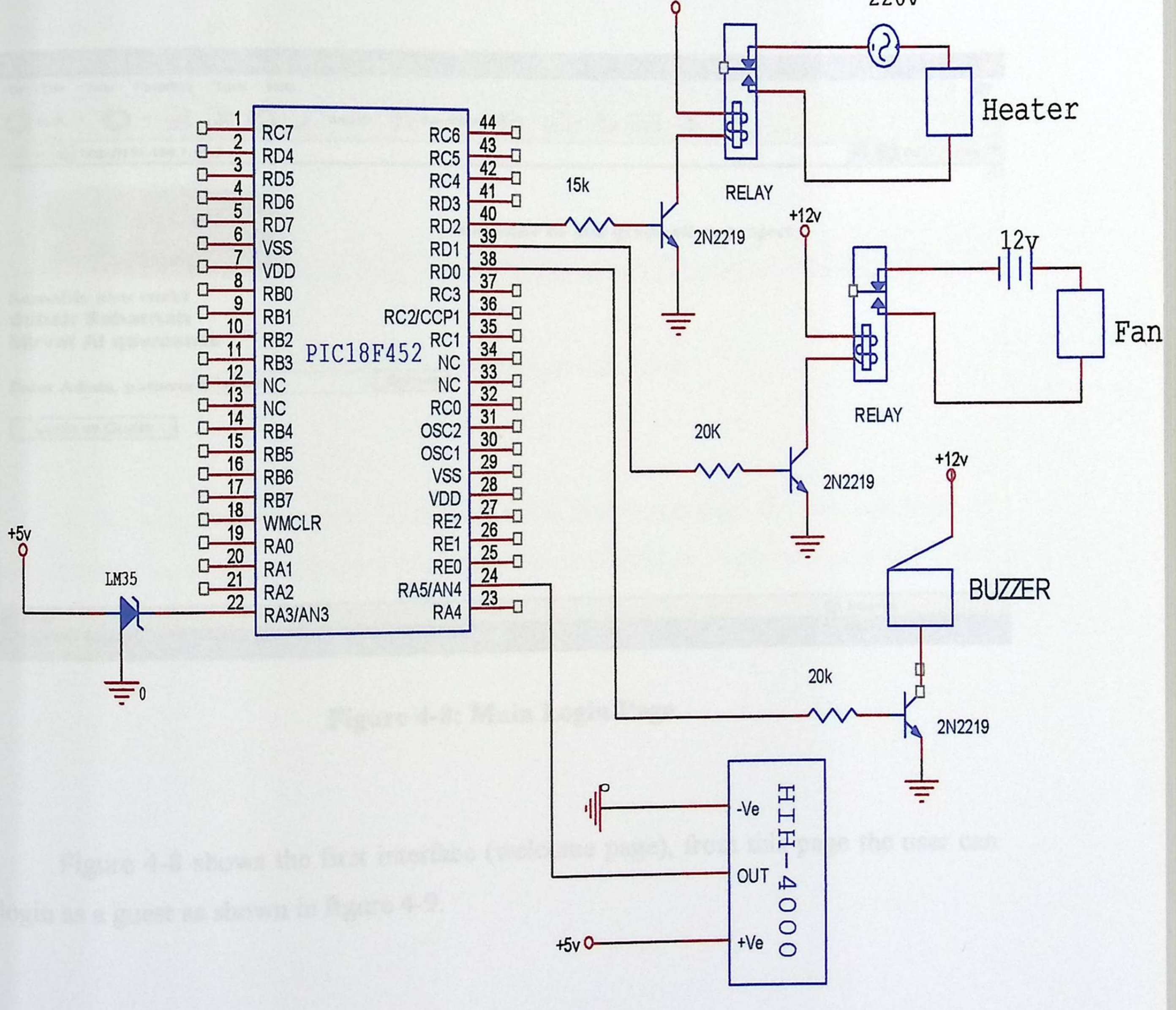
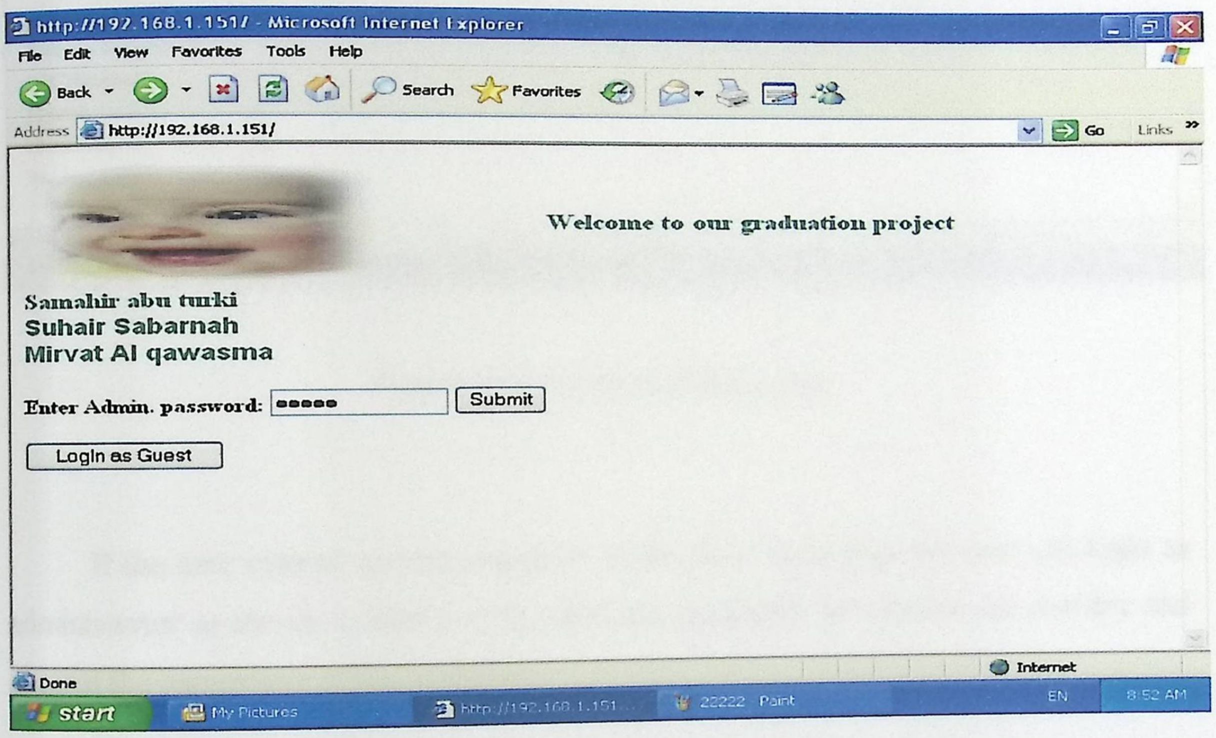


Figure 4-7: Schematic of the hardware components.

4.4 User System Interface

System interface is the screen which the user can interact with, and so communicate with the system friendly.

The following screen shots describe how our system is to be used.


http://192.168.1.151/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.1.151/ Go Links
Done Internet
start My Pictures 22222 Paint EN 8:52 AM

Welcome to our graduation project

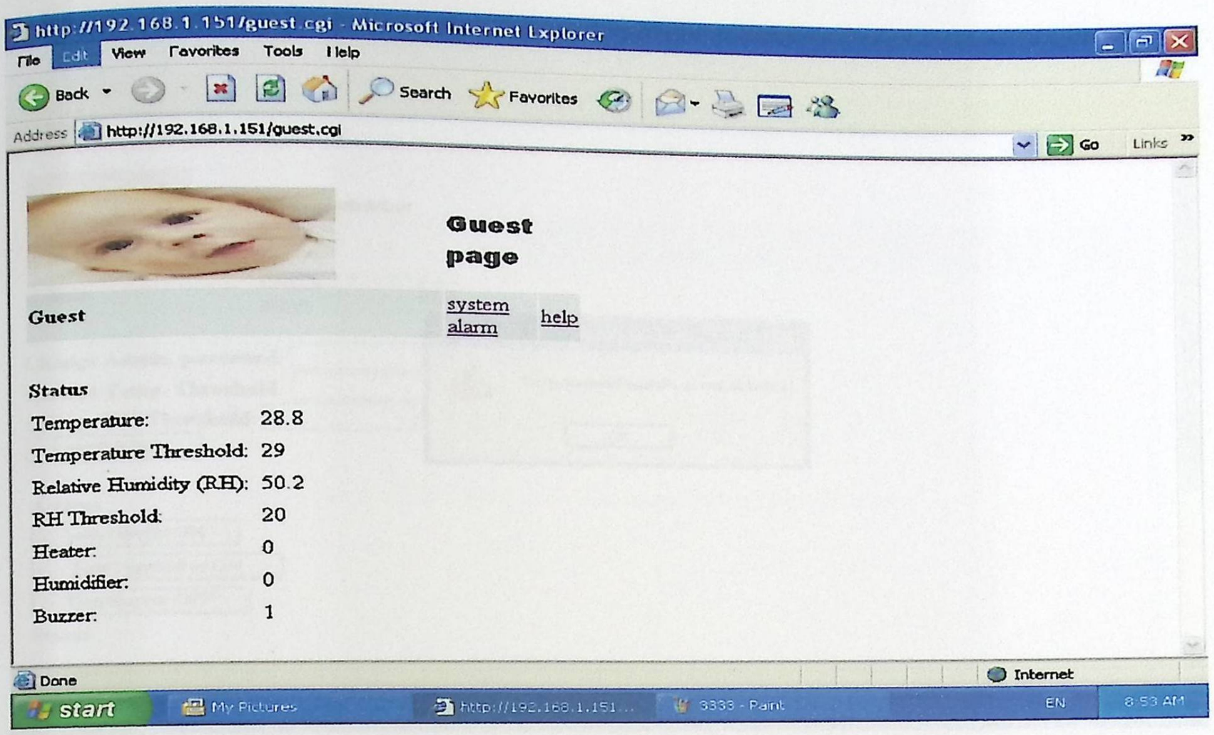
Samahir abu turki
Suhair Sabarnah
Mirvat Al qawasma

Enter Admin. password: ●●●●● Submit

Login as Guest

Figure 4-8: Main Login Page

Figure 4-8 shows the first interface (welcome page), from this page the user can login as a guest as shown in figure 4-9.

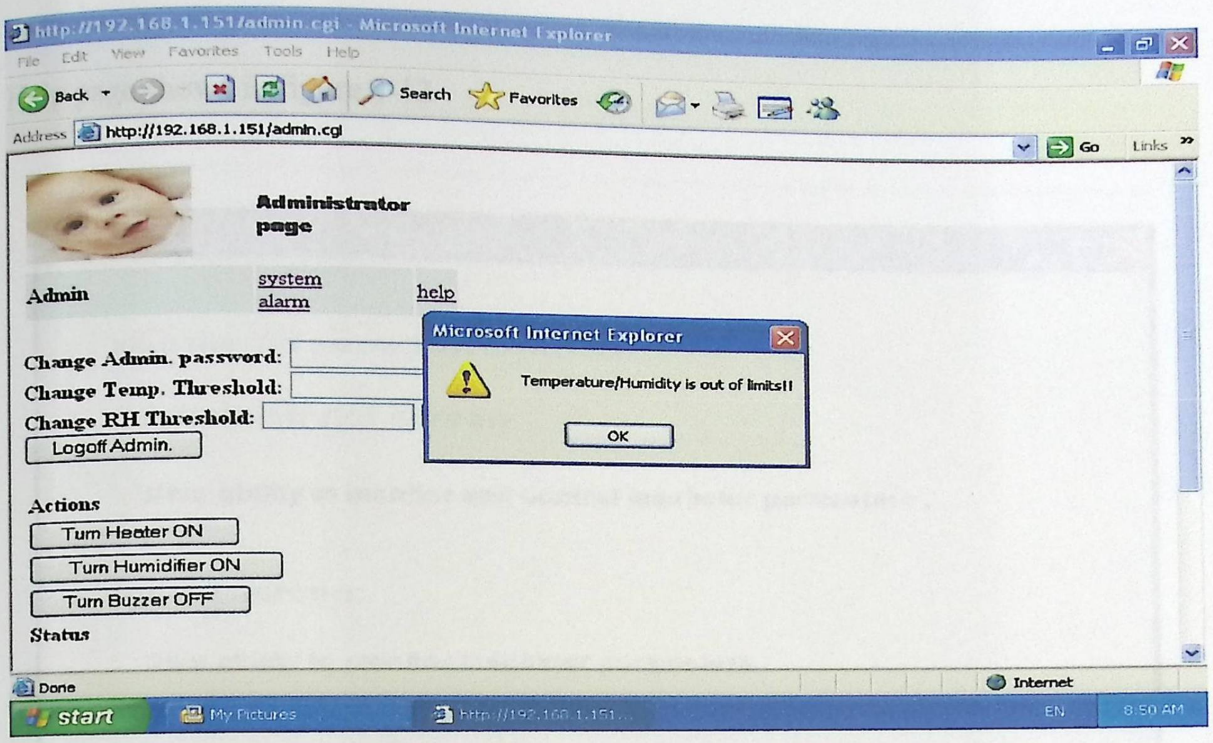

http://192.168.1.151/guest.cgi - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.1.151/guest.cgi
Go Links
Guest page
system alarm
help
Guest
Done
Internet
start
My Pictures
http://192.168.1.151 ...
3333 - Paint
EN
8:53 AM

Status

Temperature:	28.8
Temperature Threshold:	29
Relative Humidity (RH):	50.2
RH Threshold:	20
Heater:	0
Humidifier:	0
Buzzer:	1

Figure 4-9: Guest interface page

If the user entered correct password in the main login page the user can login as administrator as shown in figure 4-10, when the administrator login he can monitor and control the parameters of the incubator.


http://192.168.1.151/admin.cgi - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.1.151/admin.cgi
Go Links

Administrator page

Admin | system alarm | help

Change Admin. password:

Change Temp. Threshold:

Change RH Threshold:

Logoff Admin.

Actions

Turn Heater ON

Turn Humidifier ON

Turn Buzzer OFF

Status

Microsoft Internet Explorer

Temperature/Humidity is out of limits!!

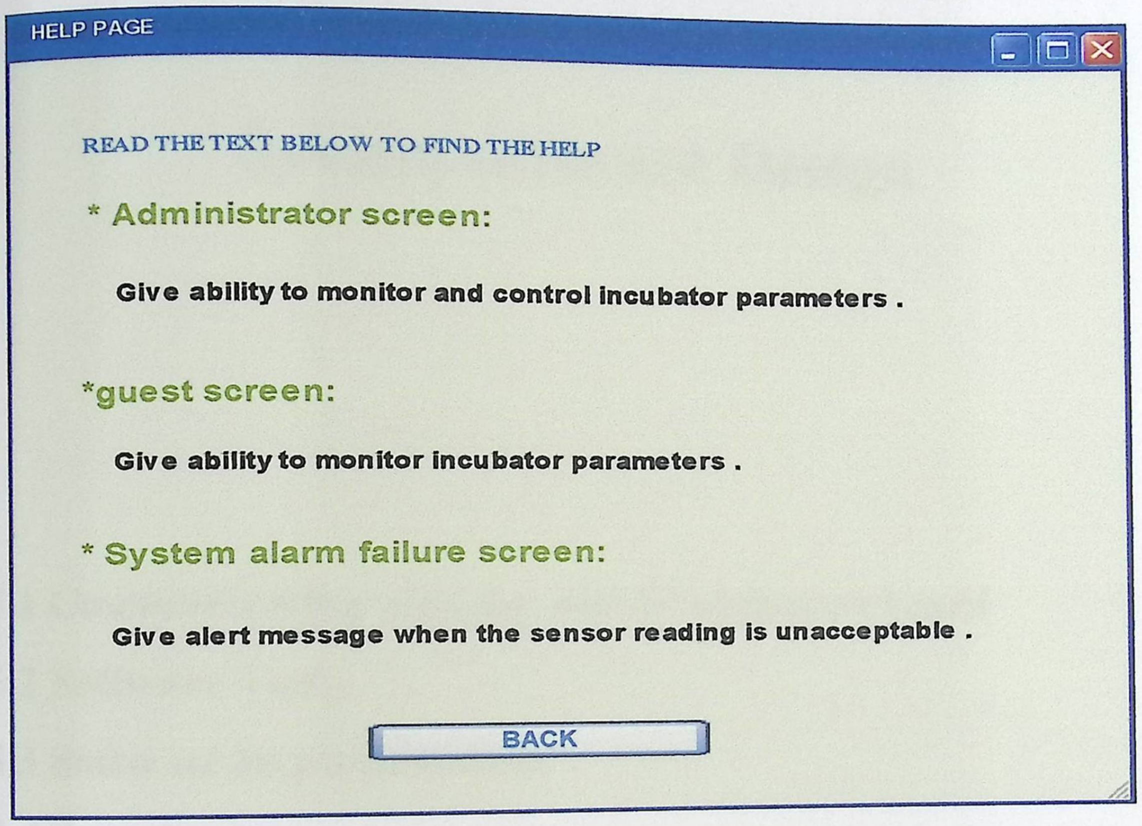
OK

Done
Internet
start
My Pictures
http://192.168.1.151...
EN
8:50 AM

Figure 4-10: Administrator interface page

When sensor reading is unacceptable, alert message appear on the client , and the administrator should do suitable actions to correct this situation.

Help page shown in figure 4-12.



READ THE TEXT BELOW TO FIND THE HELP

* Administrator screen:

Give ability to monitor and control incubator parameters .

*guest screen:

Give ability to monitor incubator parameters .

* System alarm failure screen:

Give alert message when the sensor reading is unacceptable .

Figure 4-12: Help page.

CHAPTER FIVE

System software Design

5.1 Communicating with pic web development board

5.2 Software Tools

5.3 Software Implementation

5.4 flow charts

Chapter Five

Software System Design

This chapter includes implementing the system from the programming point of view. Software Implementation depends on many components.

5.1 Communication with Pic web development board

Microchip stack is written on modular and flexible basis and you can enable or disable modules. The stack also supports dynamic web pages which give you the possibility to control all PIC resources remotely viaFTP, HTTP, UDP, TCP etc.

There are two different http servers available in the stack HTTP and HTTP 2. Depending on the version used, the dynamic interaction between the stack and the web page displayed is handled in a different way.

In this project since we use pic_web the HTTP server is used.

5.1.1 The Microchip HTTP Server

The HTTP Server provided here does not implement all HTTP functionality; it is a minimal server targeted for embedded system. The user can easily add new functionality as required.

The HTTP Server incorporates these main features:

• Contains a simple file system (MPFS)

• Supports Web pages located in either internal program memory or external serial EEPROM

• Includes a PC-based program to create MPFSmages from a given directory

• Supports the HTTP method "GET".

• Supports a modified Common Gateway Interface (CGI) to invoke predefined functions from within the remote browser

• Supports dynamic web page content generation.[3]

5.2 Software Tools

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

1. Windows XP SP2

   To acts as client.

2. MPLAB Integrated Development Environment

MPLAB IDE is free software distributed by Microchip that includes a toolset for development of PIC microcontroller applications for windows OS. It basically integrates transparently other modules like C18 or PICC-18 compilers that strengthen the functionality of MPLAB and let us program in C language.

3. MPLAB C18 Compiler

MPLAB C18 is a C compiler intended for the PIC18 family from Microchip. In our case is the needed compiler because the PIC-WEB platform uses the PIC18F452. This software converts C code into PIC18 machine code and link them together in to a .HEX file with the proper memory mapping for the microcontroller just ready to be programmed on it.

4. Microchip AN833 TCP/IP Stack

The Stack is an application developed by Microchip for been used in its PIC18 family and intended for both Microchip C18and Hi-Tech PICC-18 compilers.

5. IC-Prog

A software compatible with JDM programmer.

6. Terminal program

We use HyperTerminal program because it is included on almost all windows machine. We configured it at 19200 bps, 8NA1and no flow control.

5.3 Software implementation

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

1. Reading sensor values.

2. To provide a suitable alarm when certain malfunctionalities come up and so the administrator can perform the proper procedures and actions.

4. Secure system: using this system requires a specific password for authorization.

5.3.1 Reading Sensor Values.

5.3.1.1 Process IO Function:

Reads the various sensors connected to the PIC-WEB, store the results in a series of variables that are referenced By HTTPGetVar and wait 5 seconds before next sample. This routine is called periodically by main ().

```
ADCON0 =b10011001;

ADCResult.v [0] = 100;

While (ADCResult.v [0]);

ADCON0bits.GO = 1;

While (ADCON0bits.GO);

ADCResult.v [0] = ADRESL;

ADCResult.v [1] = ADRESH;

lm35 = ADCResult.Val;

lm35 = (lm35 * 4.88)/10;
```

5.3.1.2 HTTPGetVar Function:

This is a callback function from HTTP Server main application Whenever a variable substitution is required on any html pages, HTTP Server calls this function 8-bit variable identifier, variable reference, which indicates whether this is a first call not. Application should return one character at a time as a variable value.

Since this function only allows one character to be returned at a time as part of variable value, HTTP Server() calls this function multiple times until main application indicates that there is no more value left for this variable. On beginning, HTTPGetVar () is called with ref = HTTP_START_OF_VAR to indicate that this is a first call.

Application should use this reference to start the variable value extraction and return updated reference. If there is no more values left for this variable application should send HTTP_END_OF_VAR. If there are any bytes to send, application should Return other than HTTP_START_OF_VAR and HTTP_END_OF_VAR reference. [3]

The following code shows this function:

```
WORD HTTPGetVar (BYTE var, WORD ref, BYTE* Val) {
        Static BYTE VarString [25];
        Switch (var) {
                // LM35
        Case VAR_ANAIN_AN3:
                *Val = AN3String [(BYTE) ref];
                if (AN3String[(BYTE)ref] == '\0')
                Return HTTP_END_OF_VAR;
                Else if (AN3String [(BYTE)++ref] == '\0')
```

```
Return HTTP_END_OF_VAR;
Return
                // Password
        Case VAR_PASSWORD:
                *Val = ADMIN_PASSWORD [(BYTE) ref];
                if (ADMIN_PASSWORD[(BYTE)ref] == '\0')
                Return HTTP_END_OF_VAR;
                Else if (ADMIN_PASSWORD [(BYTE) ++ref] == '\0')
                Return HTTP_END_OF_VAR;
                Return ref;
```

5.3.2 Perform Appropriate Actions.

5.3.2.1 HTTPExecCmd Function:

This function is a "callback" from HTTP Server task. Whenever a remote node performs interactive task on page that was served, HTTPServer calls this functions with action arguments info. Main application should interpret these arguments and act accordingly [3]. See the following

```
Void HTTPExecCmd (BYTE** argv, BYTE argc) {
        BYTE command;
        BYTE var;
        Int i = 0;
        Command = argv [0][0] - '0';
        Switch (command) {
        Case CGI_CMD_DIGOUT:
        // Convert string to integer value
```

```
Var = argv [1][0] - '0';
Switch (var) {

Case CMD_FAN:
          // Toggle FAN Relay
          FAN_RELAY_IO ^= 1;
          Break;
```

5.3.3 System Failure Alarm

The system failure alarm indicates that a critical fault condition has occurred. The air temperature may go above or under setting point with specific range. This may be caused by the following conditions

1- The incubator is direct sunlight.
2- The hood door has been left open.
3- A number of portholes have been left open.
4-The air temperature sensor is malfunctioning and the incubator is getting erroneous readings from this sensor.

The alarm mode

Which indicate a possibly dangerous condition, in this mode the air temperature message will appear on the client and the audible alarm will sound at the server.

5.4 System Flowcharts

5.4.1 Temperature Subroutine

The following subroutine is the temperature subroutine .This subroutine is programmed in order to control the heating system in the incubator.

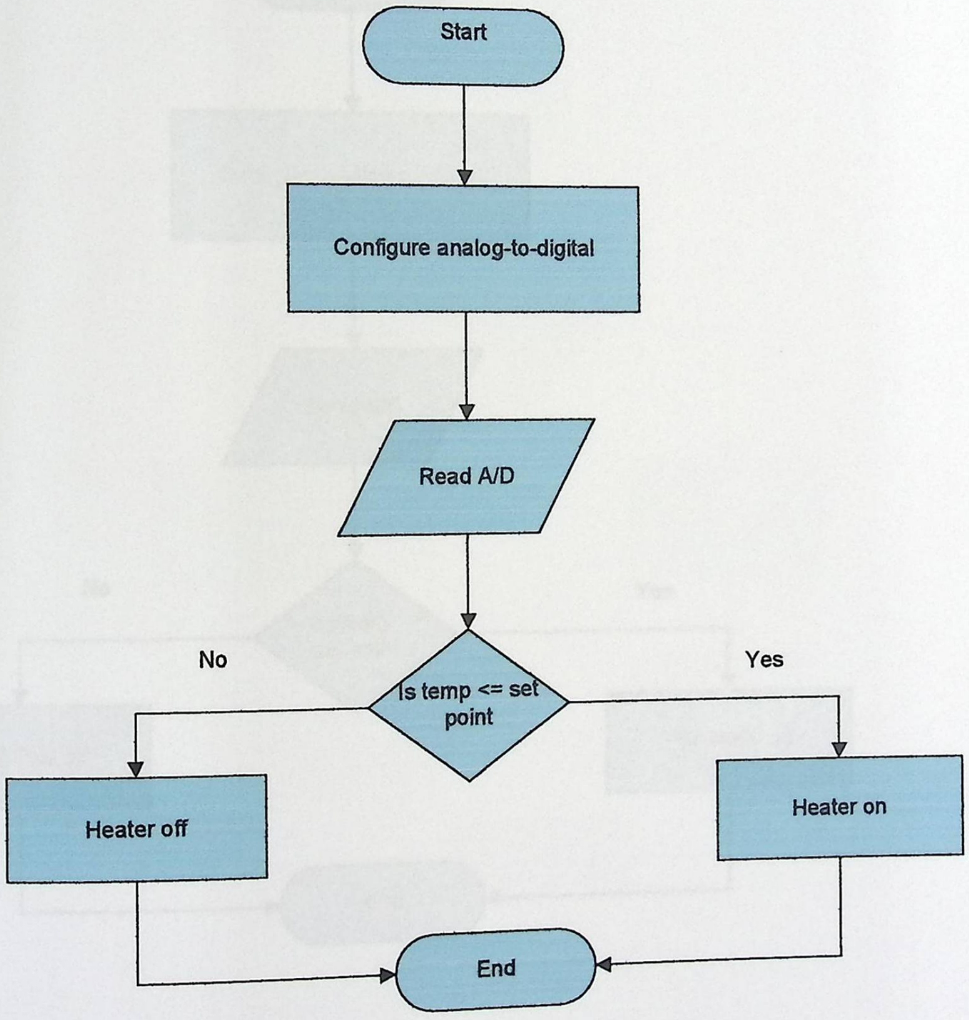


Figure 5-1 Temperature Subroutine Flow Chart

5.4 .2 Humidity Subroutine

The following subroutine is the humidity subroutine .This subroutine is programmed in order to control the humidity system in the incubator.

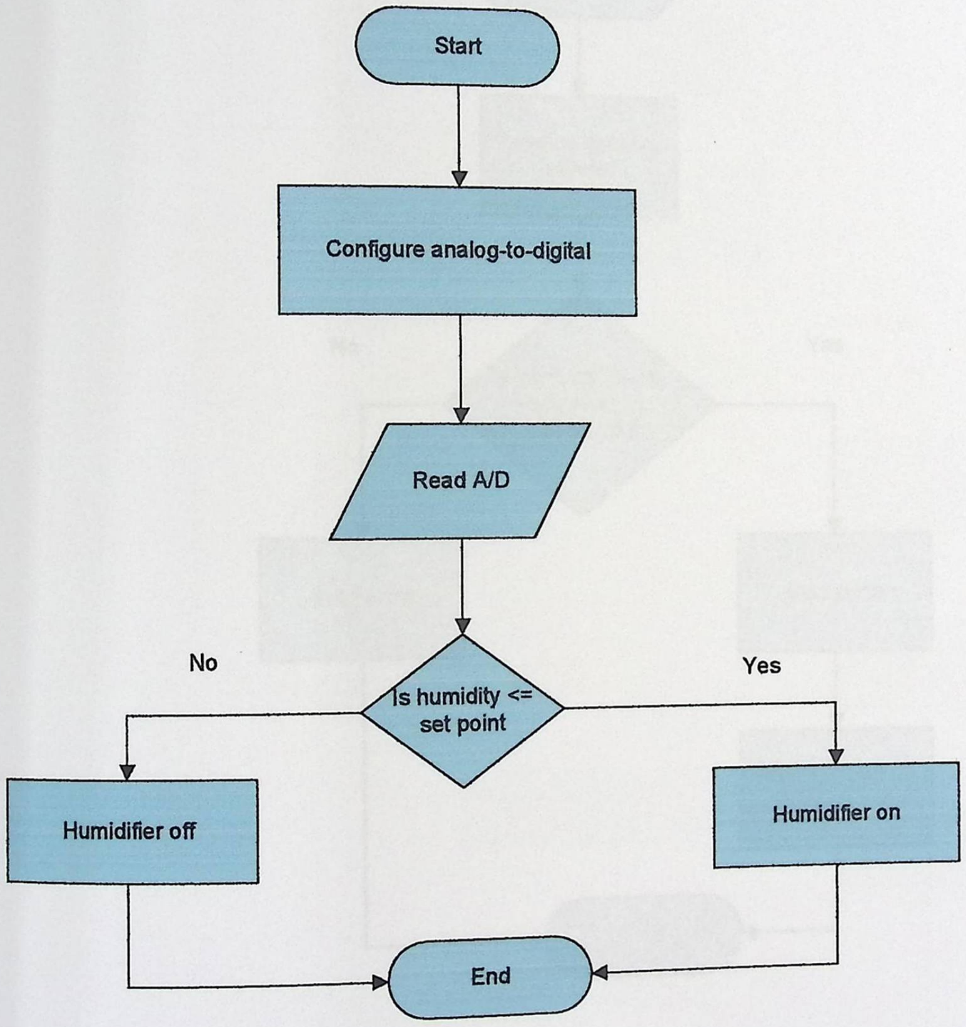


Figure 5-2 Humidity Subroutine Flow Chart

5.4.3 Alarm Flow chart

Buzzer flowchart is shown in the following.

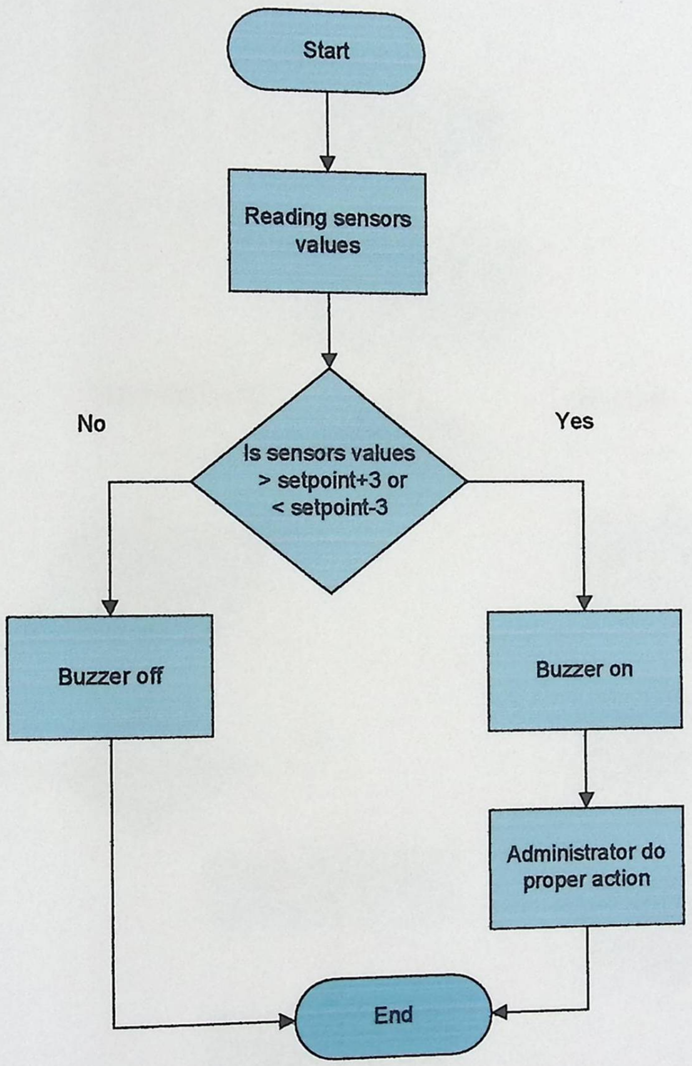

Start
Reading sensors values
No
Yes
Is sensors values > setpoint+3 or < setpoint-3
Buzzer off
Buzzer on
Administrator do proper action
End

Figure 5-3 Alarm Flow chart

5.4.4 Login flow chart (check password)

Password flowchart is shown in the following.

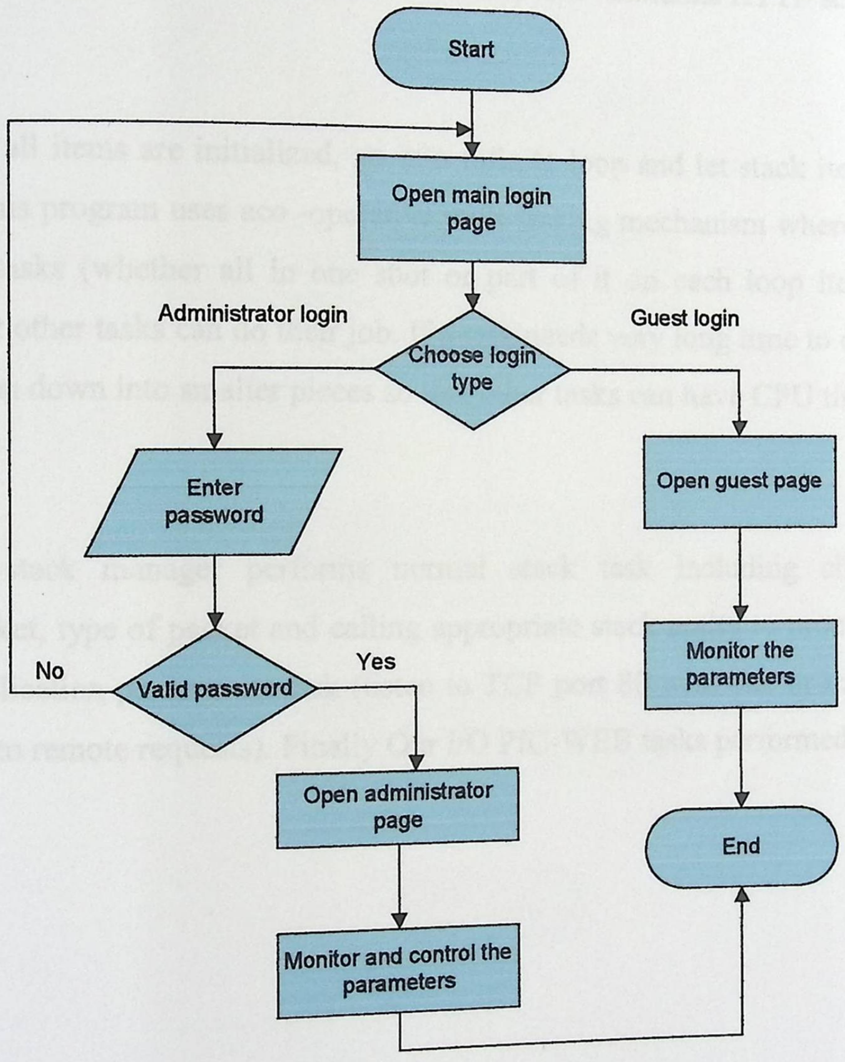


Figure 5-4 password flow chart

5.4.5 Main Entry Point Flow chart

At the begging initialize any application specific hardware then Initialize all stack related components. Which includes Initialize Microchip file system module, Read stack and application related NV variables from EEPROM into App Config, Initialize core stack layers (MAC, ARP, TCP, UDP) and Initialize HTTP server.

Once all items are initialized, go into infinite loop and let stack items execute their tasks. This program uses aco -operative mult-tasking mechanism where every task performs its tasks (whether all in one shot or part of it on each loop iteration) and returns so that other tasks can do their job. If a task needs very long time to do its job, it must be broken down into smaller pieces so that other tasks can have CPU time.

Then stack manager performs normal stack task including checking for incoming packet, type of packet and calling appropriate stack entity to process it. Then let HTTP application perform its task (listen to TCP port 80 with one or more sockets and responds to remote requests). Finally Our I/O PIC-WEB tasks performed.

The flowing flow charts show this.

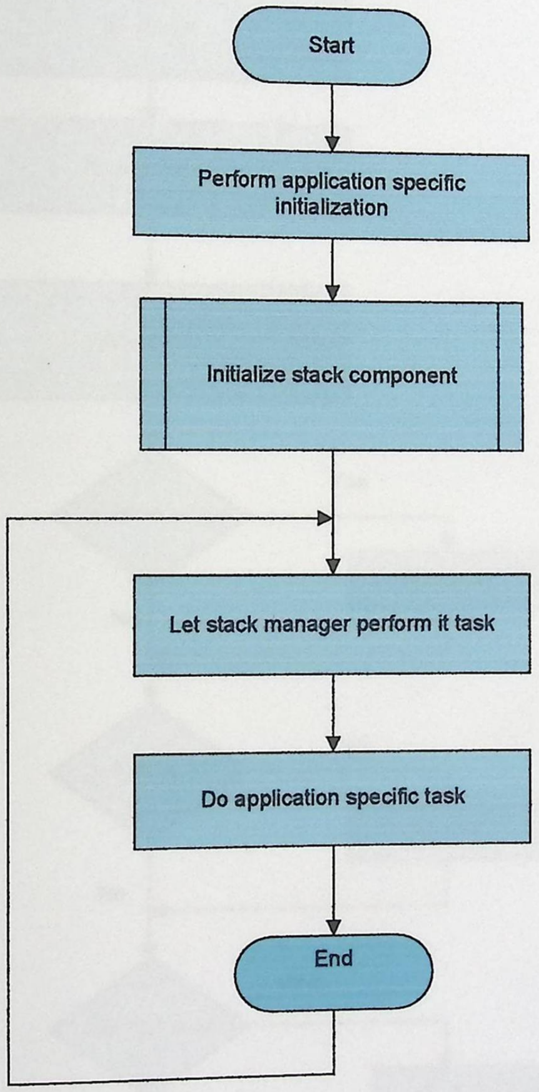

Start
Perform application specific initialization
Initialize stack component
Let stack manager perform it task
Do application specific task
End

Figure 5-5 Main Entry Point Flow chart

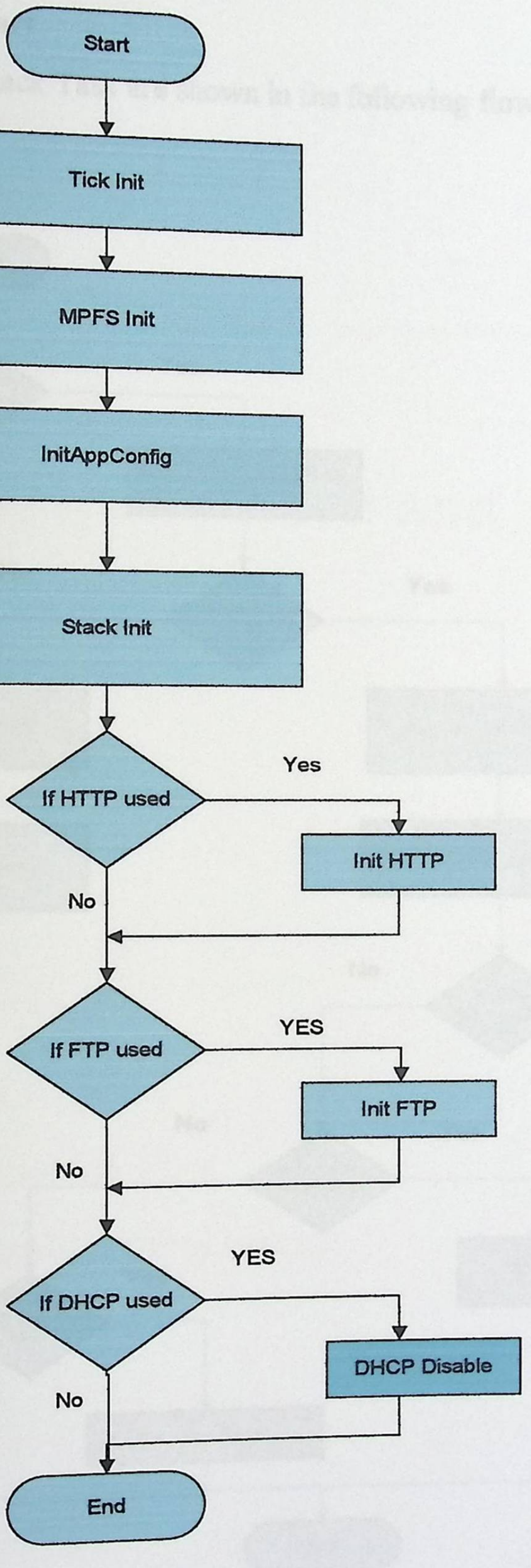


Figure 5-6 Initialize Stack Component

5.4.5 Stack Task Flow Chart

The exact steps used by Stack Task are shown in the following flow chart

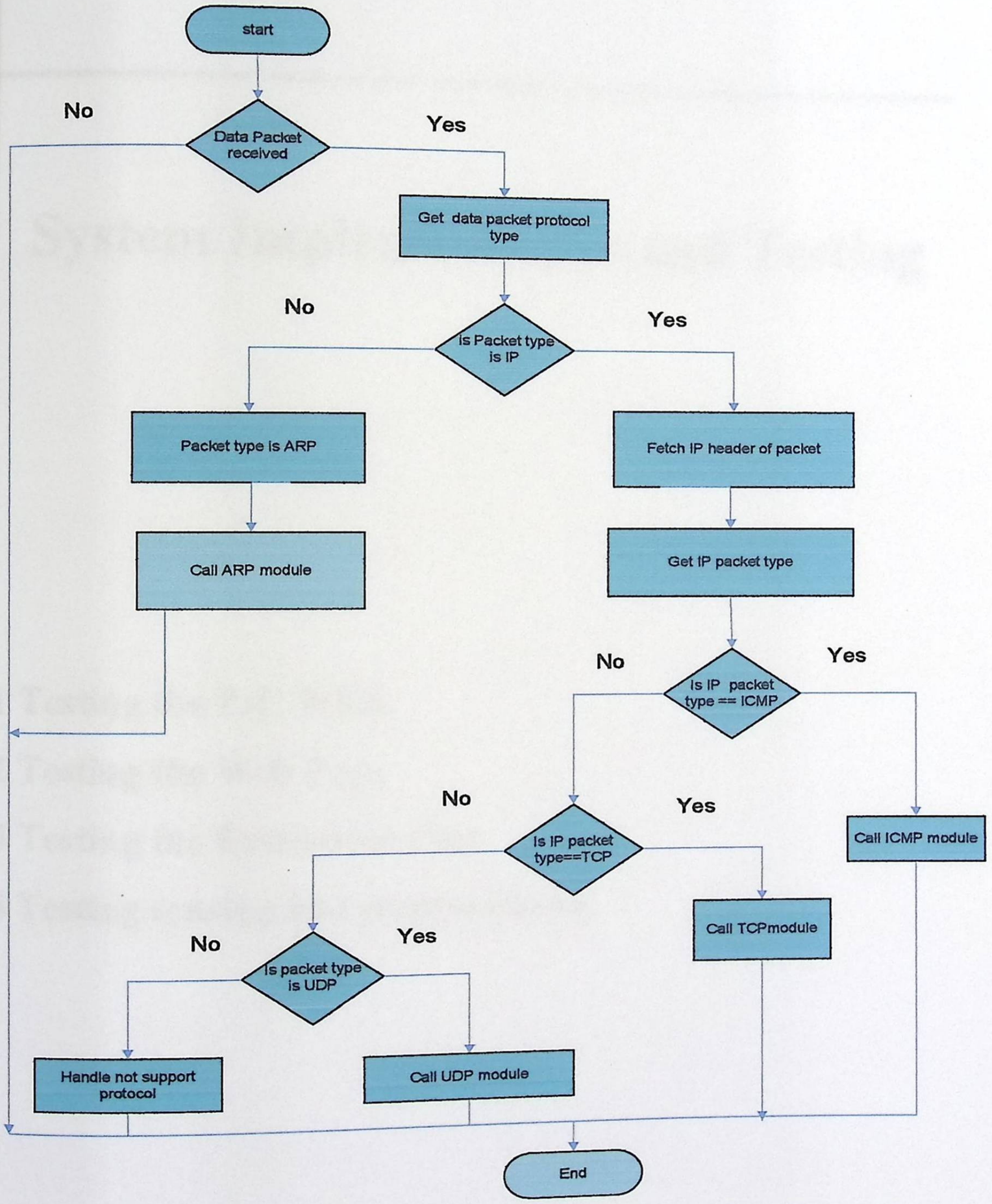


Figure5-7 Stack Task Flow Chart

CHAPTER SIX

System Implementation and Testing

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, hardware or networked 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.

Here are the testing issues. They are not ordered in any manner; rather they represent some way of system integrity and operation:

6.1 Testing the PIC WEB

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

- Structuring a cross over cable between a PC and the PIC WEB
- Connecting the PIC WEB to one of a Hub ports

After connecting the cable between the PIC WEB and the PC, power on the PIC WEB 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 PIC WEB already supports the TCP and UDP Protocols. The test is done simply by writing the following command on the Command Line Shell:

```
Ping 192.168.1.152
```

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

```
Pinging 192.168.1.152 with 32 bytes of data:
Reply from 192.168.1.152: bytes=32 time<1ms TTL=128
Reply from 192.168.1.152: bytes=32 time<1ms TTL=128
Reply from 192.168.1.152: bytes=32 time<1ms TTL=128
Reply from 192.168.1.152: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.152:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

2. Testing PIC WEB Reach ability

The PIC WEB 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 192.168.1.151 IP address is assigned to the PIC WEB.

Testing the PIC WEB assumes no DHCP server is found. So, the following command well tests the PIC WEB:

```
Ping 192.168.1.151
```

The following reply indicates the success of PIC WEB accessibility:

```
Pinging 192.168.1.151 with 32 bytes of data:
Reply from 192.168.1.151: bytes=32 time=1ms TTL=128
Reply from 192.168.1.151: bytes=32 time=1ms TTL=128
Reply from 192.168.1.151: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.1.151:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

6.2 Testing the Web Page

1-Convert the files to Microchip File System (MPFS)

The implemented HTTP server uses the MPFS simplified file system to store the pages on the microcontroller. Then it is necessary to transform the html files to that file system before upload them. This is done using a program included in PIC-WEB Software called MPFS.exe.

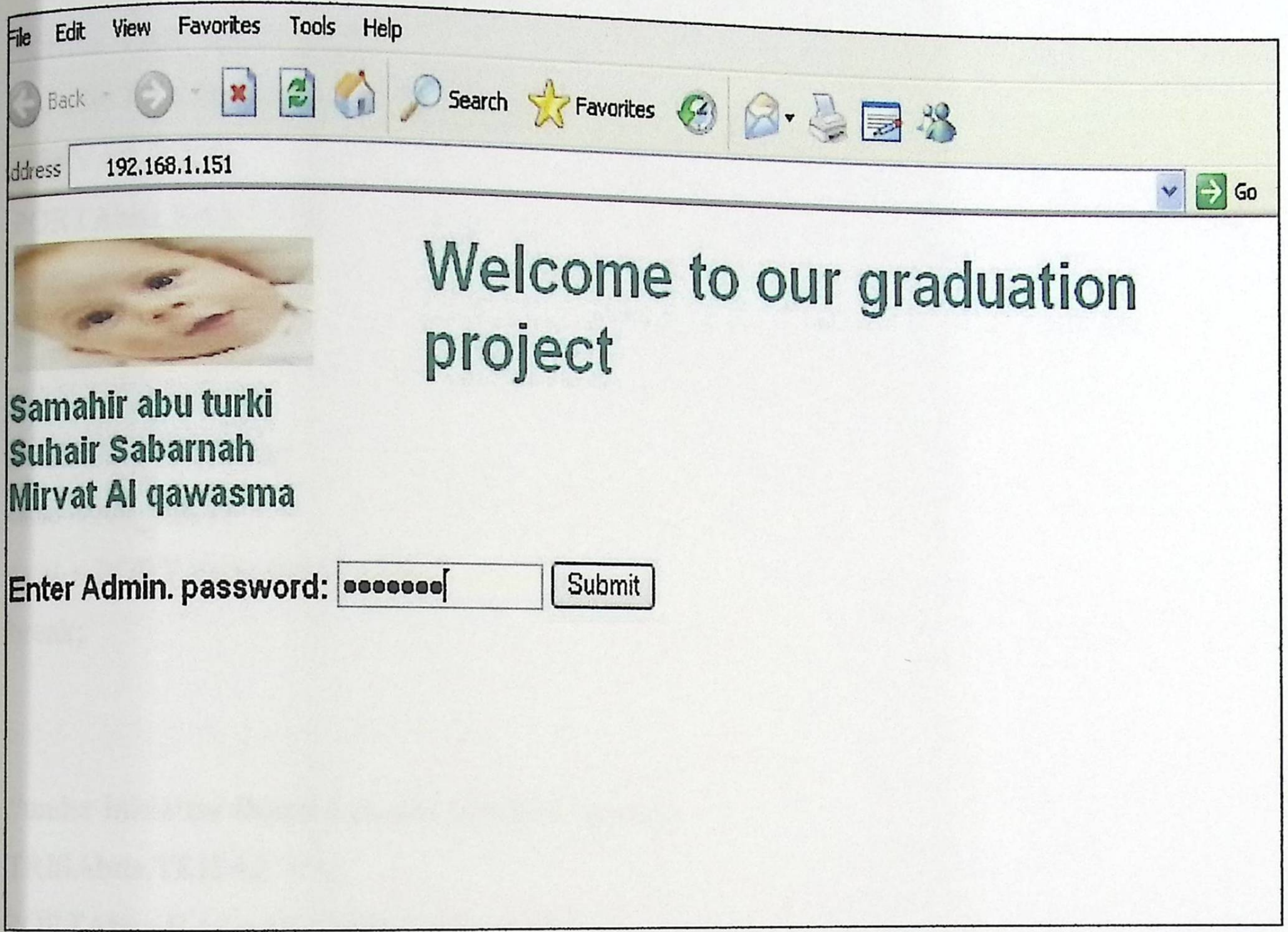
```
MPFS.exe <Web_page_directory> <Output_file.bin>
```

2-Uploading the page

There are two ways to upload the page to the board EEPROM. The first one is using the console, is faster but need direct, access to the PIC-WEB. The second is using the FTP Server that the TCP/IP Stack. During the implementing of the project we use the two way.

3-Check result

After uploading the page, the PIC-WEB board has web page on it. To access the web page, the server must be accessible from the computer. In this project we have set IP address of the board to 192.168.1.151. Then we get the page with any web browser writing the IP address of the board on the address bar as shown in the following Figure 6-1.



Welcome to our graduation project

Samahir abu turki
Suhair Sabarnah
Mirvat Al qawasma

Enter Admin. password: ●●●●●●● Submit

Figure 6-1 Welcome Web Page

6.3 Testing the Extensions Pins

The testing is implemented by connecting a led to one of the digital out pins available, in serial to a resistor and to ground, the Pin A2 corresponding to EXT 1 Pin is chosen. We add a button to changes the state of this pin. Also the actual state of that led will be displayed in the results using a simple java script code.

We add the following code in the main .c

```
#define VAR_EXT1 (0x05)
// Under HTTPExecCmd callback function
case VAR_EXT1
PORTAbits.RA2 ^ = 1;
break;


//under HTTPGetVar
case VAR_EXT1:
*val = PORTAbits.RA2? '1':'0';
break;


//under Initialize Board function callback function
TRISAbits.TRISA2  = 0;
PORTAbits.RA2 = 0; // Initial state is off
```

We add the following under the Html

```
//Test led form
<tr>
<td><input type=submit name=5 value="Toggle TEST led"></td>
</tr>


//Test led state result
<tr>
<td><font face="Arial" size="2"> Test Led: </font></td>
<td>
<SCRIPT LANGUAGE="JavaScript">
```

```
if(%05==1) document. write ("On".fontsize(2).fontcolor("green"));
else document.write("Off".fontsize(2).fontcolor("red"));
</SCRIPT>
</td>
</tr>
```

When we toggle the test led button we get the led on and that appear in results in words.

6.4 Testing sensing and control circuit.

6.4.1 Testing temperature sensor circuit

After connecting the LM5 sensor, the circuit was connected to the power supply and tested the output voltage of the circuit as shown in figure 6-2.

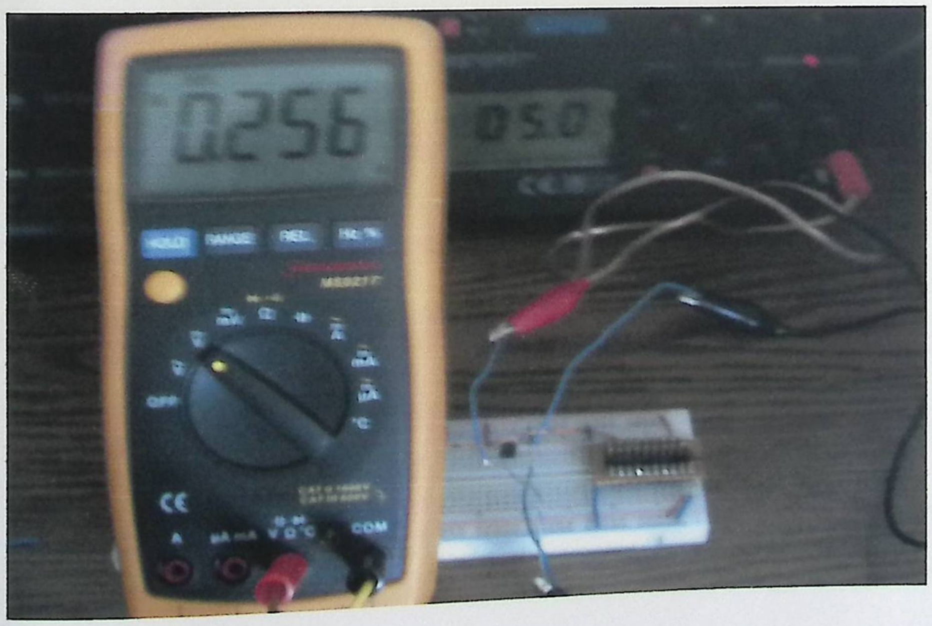


Figure 6-2Temperature sensor circuit.

The voltages obtained from LM35 passed to the Analog-to-Digital (A/D) port in the PIC WEB. PIC WEB reading's values were calculated depending on the following rules:

- The analog-to-digital (A/D) converter that allows conversion of an analog input signal to a corresponding 10-bit digital number.
- 10-bit digital number can be read as float number (N).

Vref+ =5V

Temp.=(Vref+/1024)*N *100

Temp=0.48 *N

6.4.2 Testing humidity sensor

Figure 6-3 shows the humidity sensor circuit that tested independently, and this circuit requires 5V power supply .The output from the humidity sensor goes to the PIC WEB.

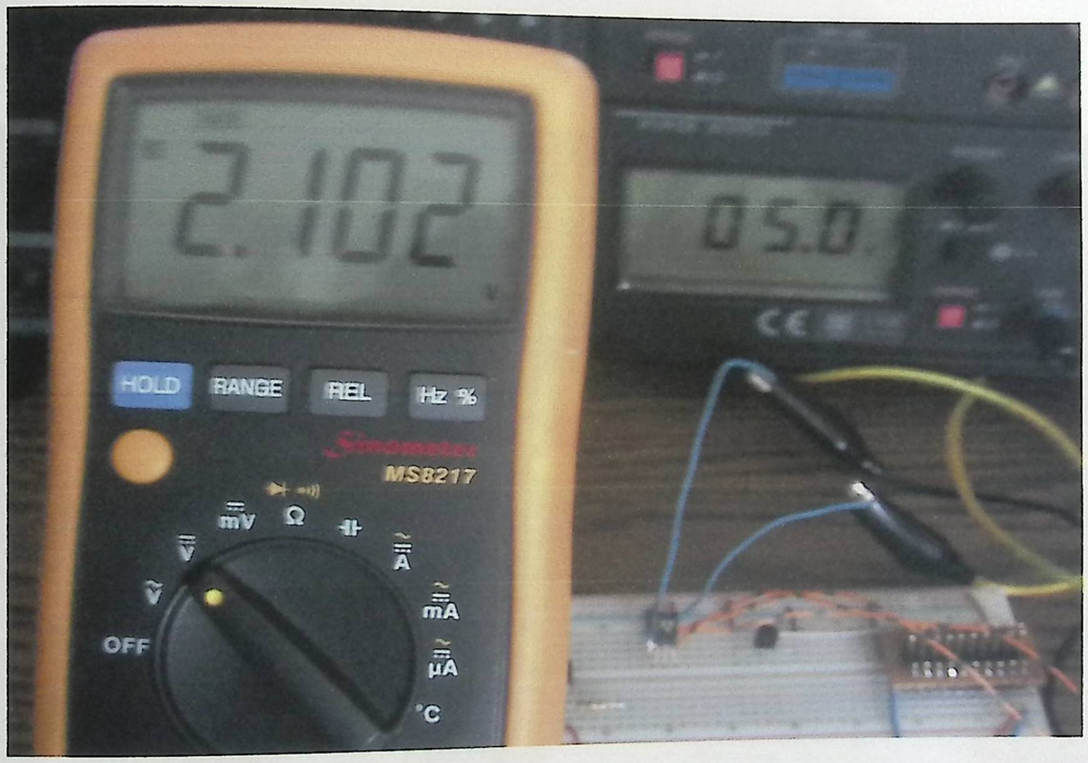


Figure 6-3Humidity sensor circuit

6.4.3 Testing heater circuit

The circuit that controls turning on/off the heater was build and tested as shown in figure 6-4 .The heater requires 220VAC power supply, this circuit operates as follows:

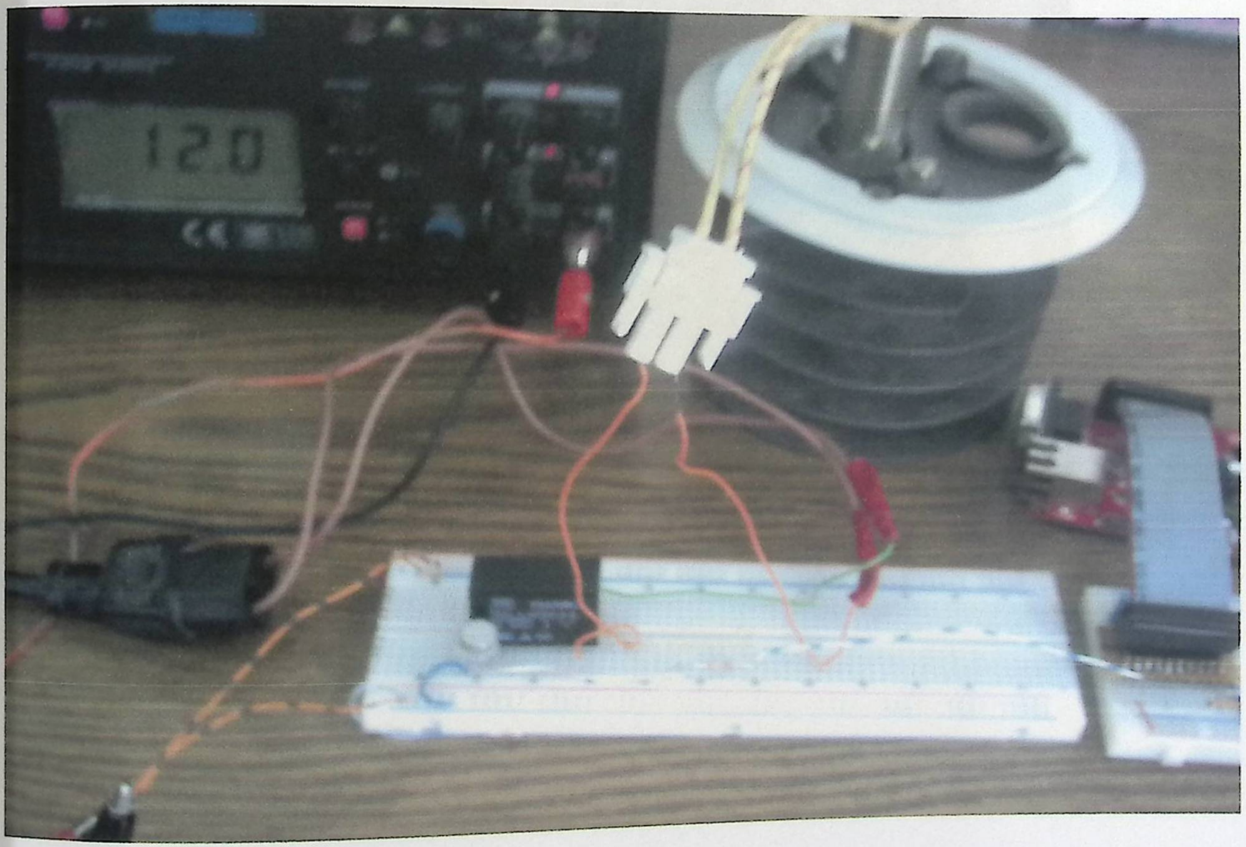


Figure 6-4 Heater circuit

6.4.5 Testing fan circuit

The circuit that controls turning on/off the fan was build and tested as shown in figure 6-5.The fan requires 12V power supply, this circuit operates as follows:

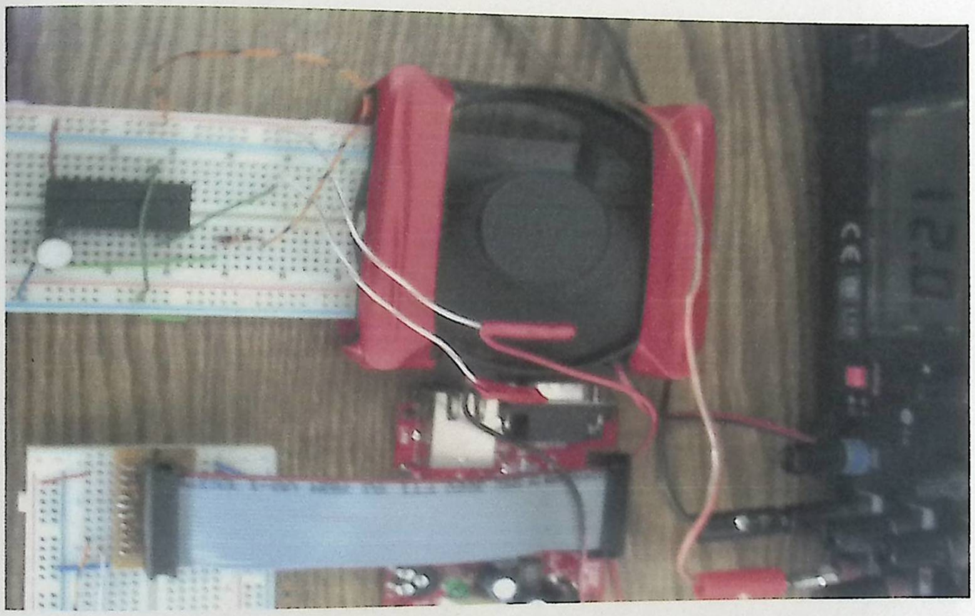


Figure 6-5 Fan circuit

6.4.6 Testing Buzzer circuit

The circuit shown in figure 6-6 is a buzzer circuit that tested independently, this circuit requires from 5V to 12V power supply.

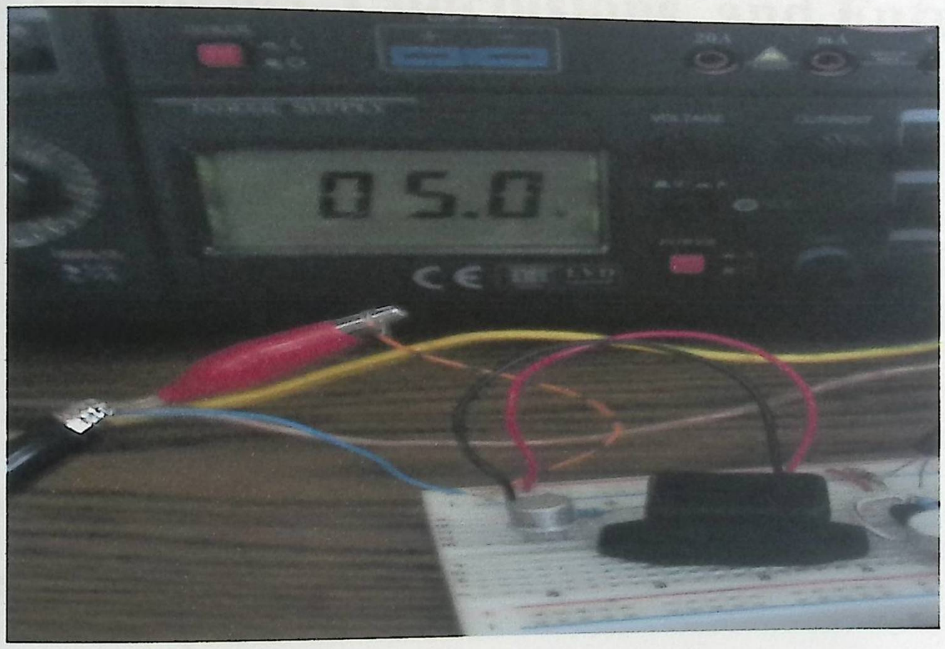


Figure 6-6 Buzzer Circuit

CHAPTER SEVEN

Problems, Conclusions, and Future Work

Chapter Seven

Conclusions, Problems and Future Work

This chapter presents some conclusions that resulted from implementing and testing the project, also we will list the problems faced us in accomplishing the system 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 a biological incubator that can be accessed remotely using a Pic web development board, this system enables the user to manage and monitor conditions within the incubator and to enable immediate response to unusual circumstances.

- The main device in the system is PIC web development board .for programming it we used MPLAB IDE.

- This project challenged us as engineers and it was very demanding as the team spent 30-40hours a week in the university lab working on this project in the last month .We learned a lot and used every thing 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 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 it 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 objectivists an implementation dependent issue. Problems are natural things. Skipping these problems is a success. No degradation affects the system if problems appear. Here are the problems faced the project team during the system implementation.

1. The development board which we decided to use was not available.

2. Instead of using SAM9-L9260 we decided to use a pic web development board after two months of semester's begging, so a lot of time was wasted.

3. Damaging in some devices, because of wrong connections or high voltages, supplied to the devices during the implementation.

4. One only pic web development board is available in hand .So one incubator on the system is implemented and tested.

5. The humidity sensor which we decided to use is low sensitivity, and so we faced a problem in the implementation of the humidity circuit.

6. Instead of use HS1101 relative humidity sensor we use the HIH4000 in the last week of the project.

6. Oxygen sensor is not available because it is very expensive, above $1000.

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:

- Develop the system by central monitoring to control a group of incubator at the same time.

- Interfacing a USB Webcam to the Pic Web-Server and provide a full remote monitoring views of the incubator.

REFFRENCES

[1] July 13, 2006 .Incubator - Basics Retrieved october15, 2008,
website:http://www.fbe.org.au/Clin/BasicEquipment/InfantIncubator.htm.

[2] April 2008. SAM9-L9260 development board. Retrieved November 20, 2008,
From olimex
website: http://www.olimex.com/dev/pdf/ARM/ATMEL/SAM9-L9260.pdf

[3] Nilesh Rajbharti, Augest2008. Microchip TCP/IP Stack Application Note .Retrieved may 20, 2009, from Microchip.
Website: http://ww1.microchip.com/downloads/en/AppNotes/00833c.pdf.

[4] December 1994. LM35 Precision Centigrade Temperature Sensors. Retrieved november10, 2008, from datasheet catalog
Website: http://www.datasheetcatalog.org/datasheet/nationalsemiconductor /DS005516.PDF

[5] March 2005.HIH-4000 Series .Retrieved may 28, 2009, from. Honeywell
Website: http://www.phanderson.com/hih-4000.pdf

[6] Lang, M.A 7 June 2009. Oxygen sensor. Retrieved December 2, 2008, 200, Website: http://en.wikipedia.org/wiki/Oxygen_sensor from Wikipedia

[7] July 2008. PIC-WEB development board. Retrieved may 15, 2009, from olimex website : http://www.olimex.com/dev/pdf/pic/pic_web.pdf

[8] Jacksonville Rd.,Hatboro ,Air _shields Isolate infant incubator, CAT.NO.67 990 16_11

[9] D.I. Shina, S.J. Huha,*, T.S. Leeb, I.Y. Kimc March 2003 .Web-based remote monitoring of infant incubators in the. ICU.

[10] Grisha Spasov, Nikolay Kakanakov 2004.CGI-based applications for distributed systems for monitoring temperature and humidity.

[11]Product Info. Retrieved march 15,2009.from spark fun
Website: http://www.sparkfun.com

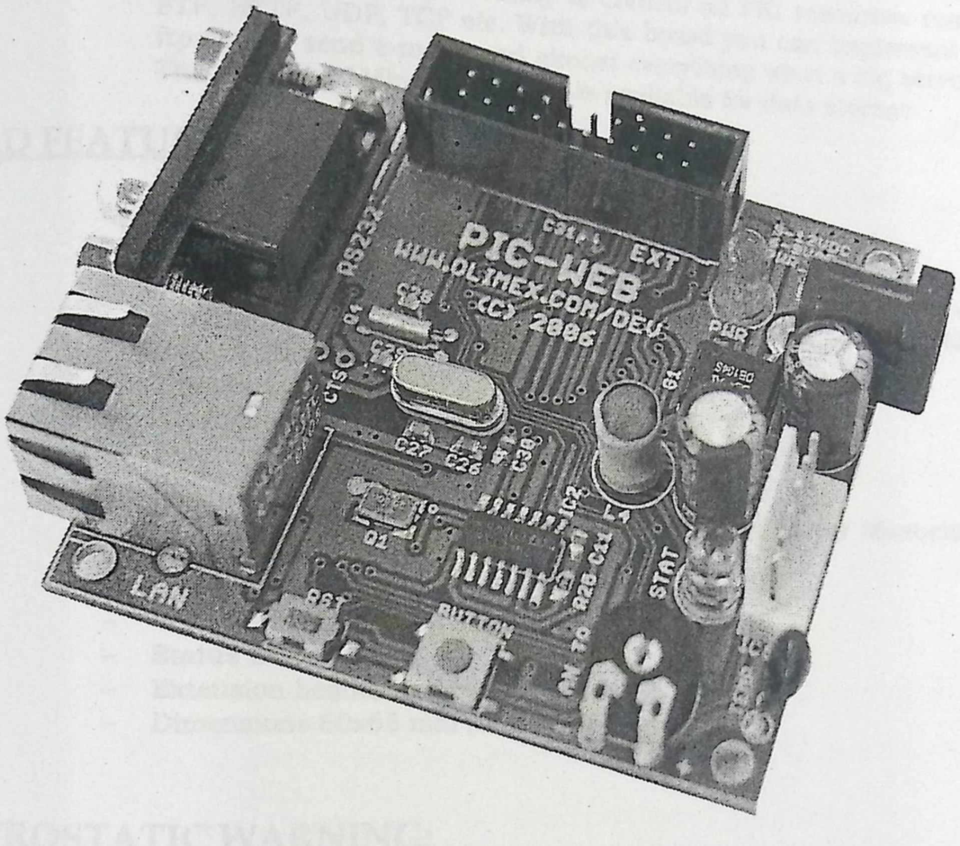
APPENDICES

APPENDEX A: PIC _WEB DEVELOPMENT BOARD Datasheet.

APPENDEX B: PROJECT ICs Datasheet.

APPENDEX C: SOURCE CODE.

APPENDIX A:PIC-WEP DEVELOPMENT BOARD Datasheet.



PIC-WEB development board
Users Manual

Rev.A, July 2008

INTRODUCTION:

PIC-WEB is compact board with 65x60 mm size which is supported by Microchip's open source TCP-IP stack AN833. The board is designed with PIC18F452 microcontroller and ENC28J60 Ethernet controller and supports: SLIP, ARP, IP, ICMP, TCP, UDP, HTTP, DHCP, FTP. The Microchip stack is written on modular and flexible basis and you can enable or disable modules. The stack also supports dynamic web pages which give you the possibility to control all PIC resources remotely via FTP, HTTP, UDP, TCP etc. With this board you can implement web and ftp server, send e-mails and almost everything what a big server can do. The on board 1Mbit serial flash is available for data storage

BOARD FEATURES:

- PIC18F452 microcontroller, ENC28J60 Ethernet controller
- 1Mbit on board serial flash for web pages storage
- ICSP/ICD connector for programming with PIC-MCP, PIC-MCP-USB and programming and debugging with PIC-ICD2 and PIC-ICD2-POCKET.
- Reset button
- User event button
- Analogue trimmer potentiometer
- Thermistor for temperature monitoring
- RS232 driver and connector
- Complete web server and TCP-IP stack support as per Microchip's open source TCP-IP stack
- Power plug-in jack for +5VDC power supply
- Voltage regulator +3.3V and filtering capacitors
- Status LED
- Extension header to connect to other boards
- Dimensions 60x65 mm (2.36x2.55")

ELECTROSTATIC WARNING:

The PIC-WEB board is shipped in protective anti-static packaging. The board must not be subject to high electrostatic potentials. General practice for working with static sensitive devices should be applied when working with this board.

BOARD USE REQUIREMENTS:

Cables: Depends on the used programming/debugging tool. It could be 1.8 meter USB A-B cable to connect PIC-MCP-USB, PIC-ICD2 or PIC-ICD2-POCKET to USB host on PC or RS232 cable in case of PIC-MCP or other programming/debugging tools. You will need a serial cable if not for programming, than for configuring the board. You will also need a LAN cable.

Hardware: Programmer/Debugger – most of Olimex programmers are applicable, for example **PIC-MCP, PIC-MCP-USB, PIC-ICD2, PIC-ICD2-POCKET** or other compatible programming/debugging tool.

Software: PIC-WEB is tested with **MPLAB IDE v.7.62 + MPLAB C18** C compiler. It is possible that the stack might not function properly if used with later versions of MPLAB IDE.
You will also need a terminal program configured at 19 200 bps, 8N1 and no flow control.

PROCESSOR FEATURES:

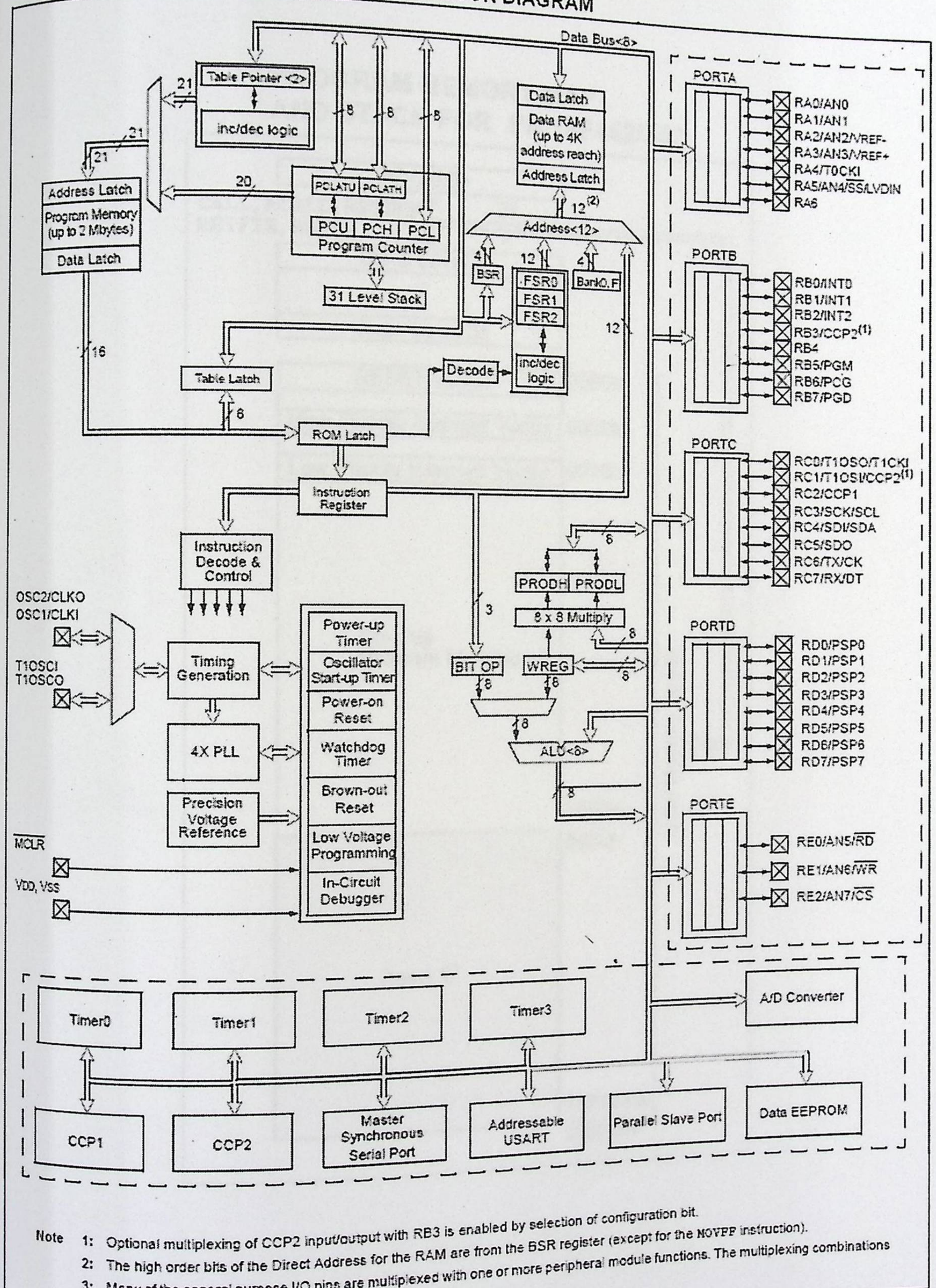
PIC-WEB board uses microcontroller **PIC18F452** from Microchip with these features:

- C compiler optimized architecture/instruction set
  - o Source code compatible with the PIC16C, PIC17C and PIC18C instruction sets
- 32 Kbytes FLASH, 1536 bytes RAM and 256 bytes EEPROM on board
- Up to 10 MIPs operation:
  - o DC - 40 MHz osc./clock input
  - o 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier
- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules. CCP pins that can be configured as:
  - o Capture input: capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - o Compare is 16-bit, max. resolution 100 ns (TCY)
  - o PWM output: PWM resolution is 1- to 10-bit, Max. PWM freq. @: 8-bit resolution = 156 kHz and 10-bit resolution = 39 kHz
- Master Synchronous Serial Port (MSSP) module, Two modes of operation:
  - o 3-wire SPI™ (supports all 4 SPI modes)
  - o I2C™ Master and Slave mode
- Addressable USART module:
  - o Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module
- Compatible 10-bit Analog-to-Digital Converter module (A/D) with:
  - o Fast sampling rate
  - o Conversion available during SLEEP
  - o DNL = ±1 LSb, INL = ±1 LSb
- Programmable Low Voltage Detection (PLVD)
  - o Supports interrupt on-Low Voltage Detection
- Programmable Brown-out Reset (BOR)
- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory
- FLASH/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)

- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options including:
    - 4X Phase Lock Loop (of primary oscillator)
    - Secondary Oscillator (32 kHz) clock input
- Single supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges

BLOCK DIAGRAM:

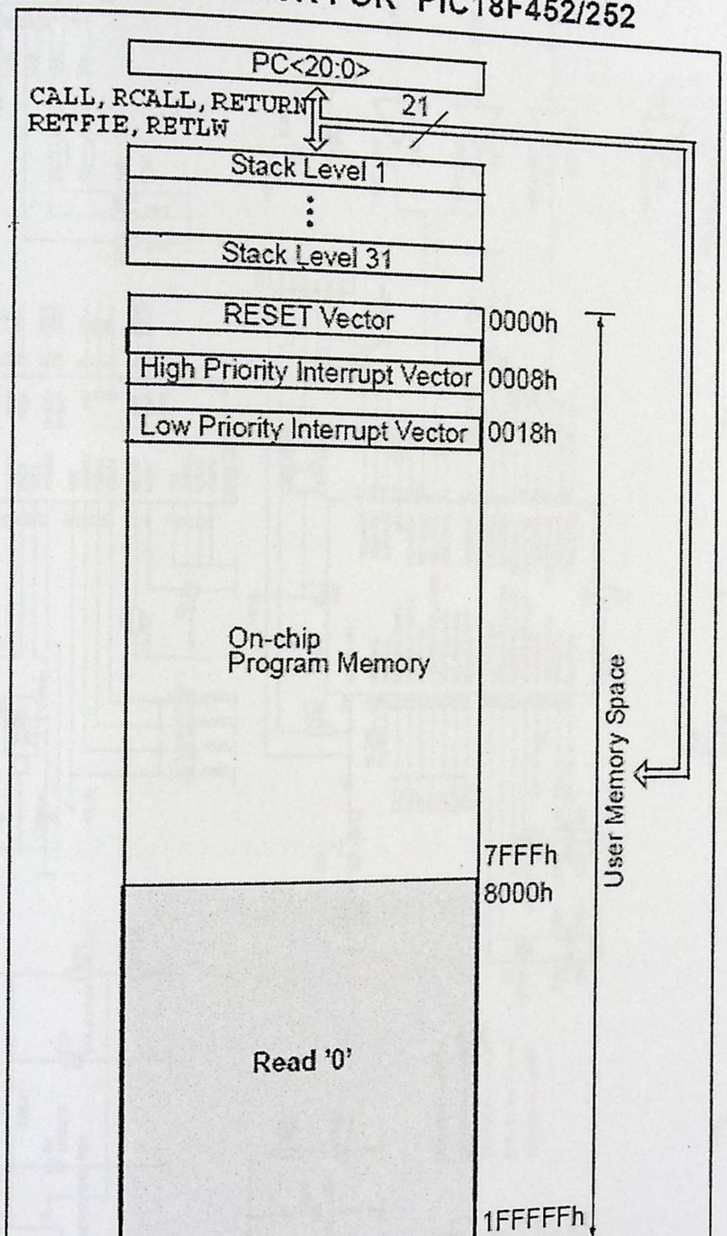
PIC18F4X2 BLOCK DIAGRAM



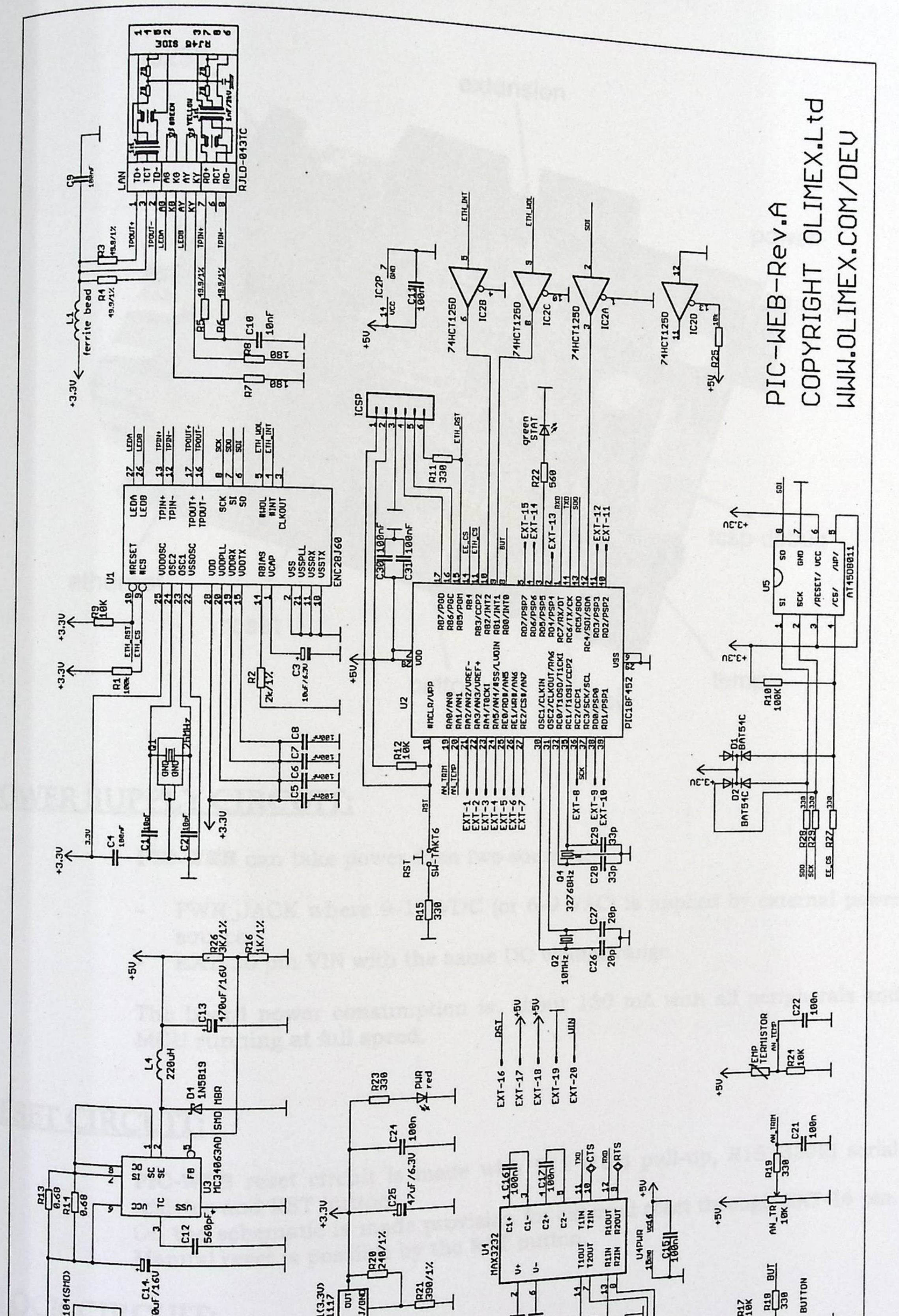
Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.
2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the MOVFF instruction).
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

MEMORY MAP:

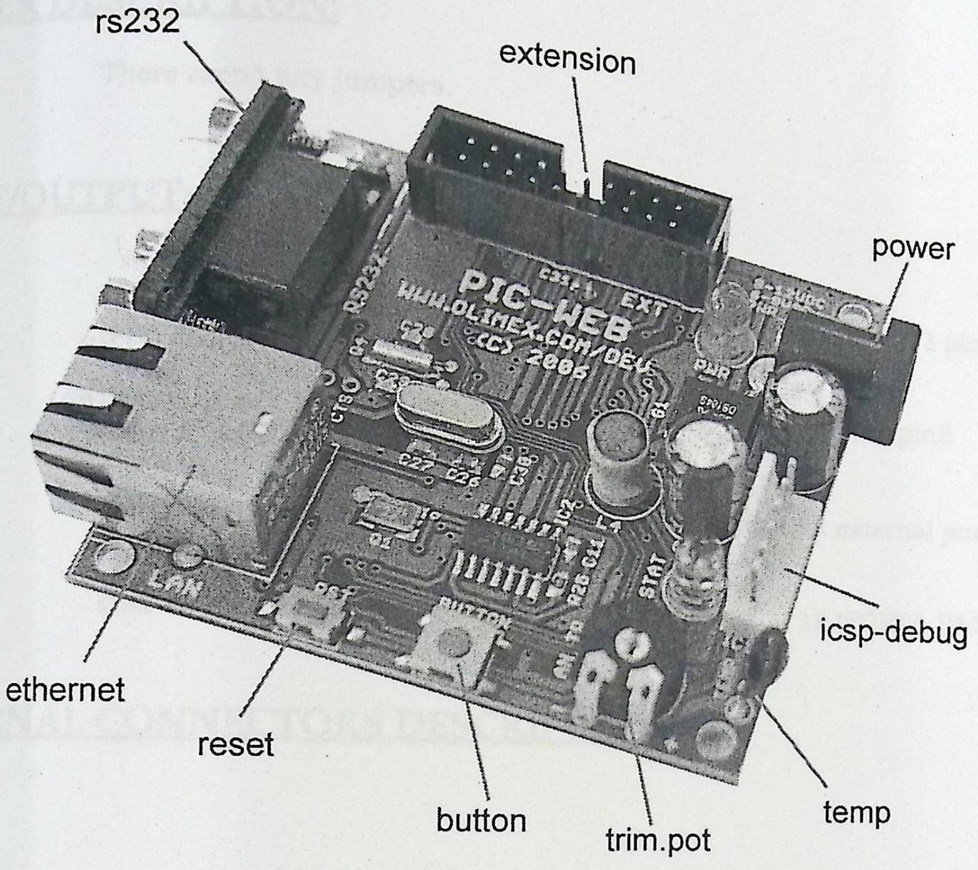
PROGRAM MEMORY MAP AND STACK FOR PIC18F452/252



SCHEMATIC:


PIC-WEB-Rev.A
COPYRIGHT OLIMEX.Ltd
WWW.OLIMEX.COM/DEV
ENC28J60
PIC18F452
MAX3232
MC34063AD
LM1117
AT45DB011
74HCT125D
RJLD-043TC
ICSP
RS232
DB9_female
PWR_JACK
9-12VDC
6-9VAC

BOARD LAYOUT:



POWER SUPPLY CIRCUIT:

PIC-WEB can take power from two sources:

- PWR_JACK where 9-12 VDC (or 6-9 VAC) is applied by external power source.
- EXT-20 pin VIN with the same DC voltage range.

The board power consumption is: about 130 mA with all peripherals and MCU running at full speed.

RESET CIRCUIT:

PIC-WEB reset circuit is made with R12 (10k) pull-up, R15 (330R) serial resistor and RST button.
On the schematic is made provision for external reset through EXT-16 pin.
Manual reset is possible by the RST button.

CLOCK CIRCUIT:

Quartz crystal 10 MHz is connected to **PIC18F452** clock in (OSC1/CLKIN) and clock out (OSC2/CLKOUT).

32.768 KHz quartz crystal is connected to **PIC18F452** T1OSO and T1OSI pins for its internal Real Time Clock.

JUMPER DESCRIPTION:

There aren't any jumpers.

INPUT/OUTPUT:

One **User button** with name **BUTTON** – connected to PIC18F452 pin 8 (PORTB.RB0/INT0);

Status green LED with name **STAT** connected to PIC18F452 pin3 (PORTD.RD5/PSP5).

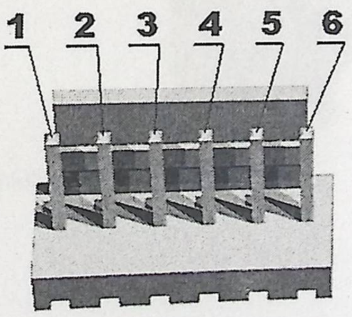
Power supply red LED with name **PWR** – indicates that external powers source is applied and board power supply is applied;

One trimmer AN_TR is connected to PIC18F452 pin 19 (PORTA.RA0/AN0).

EXTERNAL CONNECTORS DESCRIPTION:

ICSP:

Pin #	Signal Name
1	RST
2	+5V
3	GND
4	PGD
5	PGC
6	PGM

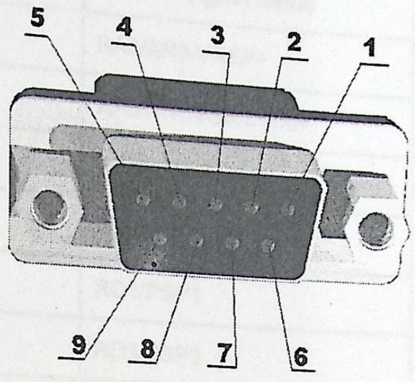


PGD I/O **Program Data.** Serial data for programming.
PGC Input **Program Clock.** Clock used for transferring the serial data (output from ICSP, input for the MCU).
PGM Input **Program Enable** (output from ICSP, input for the MCU).

RS232:

Pin #	Signal Name

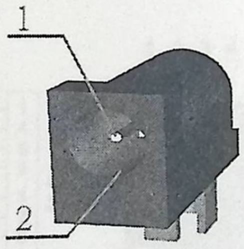
1	NC
2	TXD
3	RXD
4	NC
5	GND
6	NC
7	RTS
8	CTS
9	NC



TXD Output **Transmit Data.** This is the asynchronous serial data output (RS232) for the shift register on the UART controller.
RXD Input **Receive Data.** This is the asynchronous serial data input (RS232) for the shift register on the UART controller.
RTS Pin **Request To Send.** This is the RST pin on the board which is not connected to the PIC18F452 MCU.
CTS Pin **Clear To Send.** This is the CTS pin on the board which is not connected to the PIC18F452 MCU.

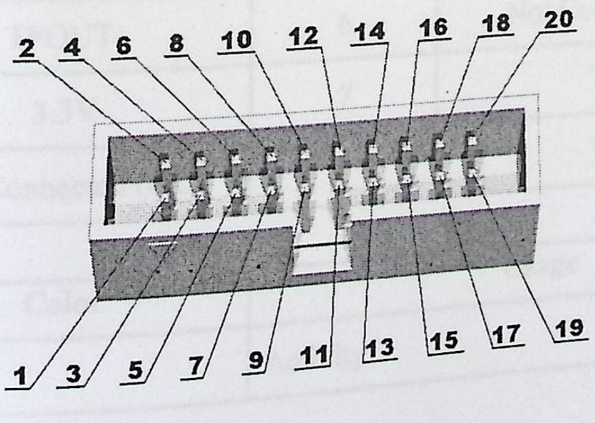
PWR JACK:

Pin #	Signal Name
1	Power Input
2	GND



The power input should be +9VDC/6VAC.

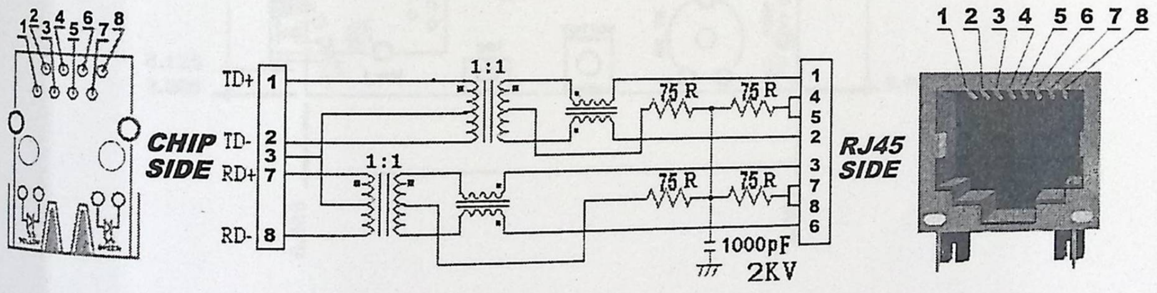
EXT:



ATTENTION!!!: EXT-18 pin is not 3.3V but 5V!!!

Pin #	Signal Name	Pin #	Signal Name
1	RA2/AN2/VREF–	2	RA3/AN3/VREF+
3	RA4/T0CKI	4	RA5/AN4/#SS/LVDIN
5	RE0/RD#/AN5	6	RE1/WR#/AN6
7	RE2/CS#/AN7	8	RC2/CCP1
9	RD0/PSP0	10	RD1/PSP1
11	RD2/PSP2	12	RD3/PSP3
13	RD4/PSP4	14	RD6/PSP6
15	RD7/PSP7	16	RST
17	+5V	18	+5V!!!
19	GND	20	VIN

LAN:



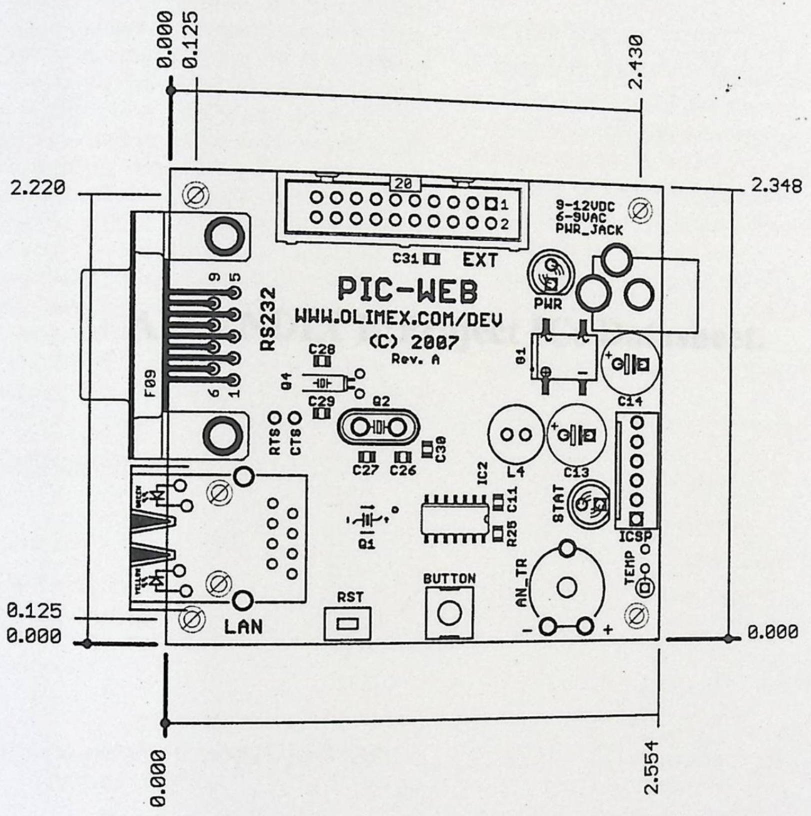
Pin #	Signal Name Chip Side	Pin #	Signal Name Chip Side
1	TPOUT+	5	Not Connected (NC)
2	TPOUT-	6	Not Connected (NC)
3	3.3V	7	TPIN+
4	Not Connected (NC)	8	TPIN-

LED	Color	Usage
Right	Yellow	Activity

Left	Green	100MBits/s (Half/Full duplex)

TPOUT-	Output	Differential signal output.
TPOUT+	Output	Differential signal output.
TPIN-	Input	Differential signal input.
TPIN+	Input	Differential signal input.

MECHANICAL DIMENSIONS:



All measures are in Inches.

AVAILABLE DEMO SOFTWARE:

You could find information about PIC-WEB board, Microchip TCP/IP stack and how to change and configure the software in Understanding PIC WEB boards on www.olimex.com/dev.

APPENDIX B:Project ICs Datasheet.

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

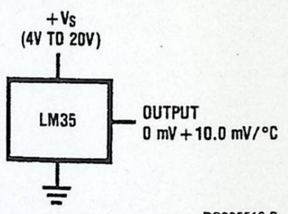
General Description

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 ±¼˚C at room temperature and ±¾˚C over a full −55 to +150˚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 µA 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˚C temperature range, while the LM35C is rated for a −40˚ to +110˚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

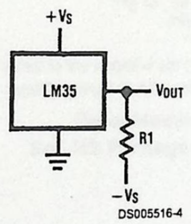
- Calibrated directly in ˚ Celsius (Centigrade)
- Linear + 10.0 mV/˚C scale factor
- 0.5˚C accuracy guaranteeable (at +25˚C)
- Rated for full −55˚ to +150˚C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08˚C in still air
- Nonlinearity only ±¼˚C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications



DS005516-3

FIGURE 1. Basic Centigrade Temperature Sensor (+2˚C to +150˚C)



DS005516-4

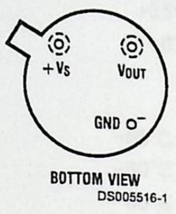
Choose $R_1 = -V_S/50\ \mu A$

V_{OUT} = +1,500 mV at +150˚C
= +250 mV at +25˚C
= −550 mV at −55˚C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

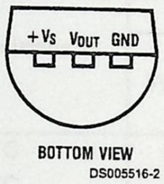
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

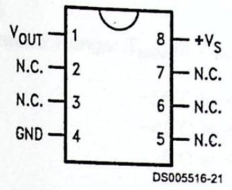
Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

TO-92
Plastic Package



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

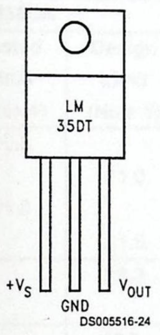
SO-8
Small Outline Molded Package



N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

TO-220
Plastic Package*



*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 Typical	LM35A Tested Limit (Note 4)	LM35A Design Limit (Note 5)	LM35CA Typical	LM35CA Tested Limit (Note 4)	LM35CA Design Limit (Note 5)	Units (Max.)
Accuracy (Note 7)	T_A=+25˚C	±0.2	±0.5		±0.2	±0.5		˚C
	T_A=−10˚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}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	67		56	67		μA
	V_S=+5V	**105**		**131**	**91**		**114**	μA
	V_S=+30V, +25˚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≤V_S≤30V, +25˚C	0.2	1.0		0.2	1.0		μA
	4V≤V_S≤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

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	(Max.)
Accuracy, LM35, LM35C (Note 7)	$T_A=+25°C$	±0.4	±1.0		±0.4	±1.0		°C
	$T_A=-10°C$	±0.5			±0.5		±1.5	°C
	$T_A=T_{MAX}$	±0.8	±1.5		±0.8		±1.5	°C
	$T_A=T_{MIN}$	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D (Note 7)	$T_A=+25°C$				±0.6	±1.5		°C
	$T_A=T_{MAX}$				±0.9		±2.0	°C
	$T_A=T_{MIN}$				±0.9		±2.0	°C
Nonlinearity (Note 8)	$T_{MIN}\le T_A\le T_{MAX}$	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
Sensor Gain (Average Slope)	$T_{MIN}\le T_A\le T_{MAX}$	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
Load Regulation (Note 3) $0\le I_L\le 1$ mA	$T_A=+25°C$	±0.4	±2.0		±0.4	±2.0		mV/mA
	$T_{MIN}\le T_A\le T_{MAX}$	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation (Note 3)	$T_A=+25°C$	±0.01	±0.1		±0.01	±0.1		mV/V
	$4V\le V_S\le 30V$	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current (Note 9)	$V_S=+5V$, +25°C	56	80		56	80		μA
	$V_S=+5V$	**105**		**158**	**91**		**138**	μA
	$V_S=+30V$, +25°C	56.2	82		56.2	82		μA
	$V_S=+30V$	**105.5**		**161**	**91.5**		**141**	μA
Change of Quiescent Current (Note 3)	$4V\le V_S\le 30V$, +25°C	0.2	2.0		0.2	2.0		μA
	$4V\le V_S\le 30V$	**0.5**		**3.0**	**0.5**		**3.0**	μA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	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

Note 1: Unless otherwise noted, these specifications apply: $-55°C\le T_J\le +150°C$ for the LM35 and LM35A; $-40°C\le T_J\le +110°C$ for the LM35C and LM35CA; and $0°\le T_J\le +100°C$ for the LM35D. V_S=+5Vdc and I_{LOAD}=50 μA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

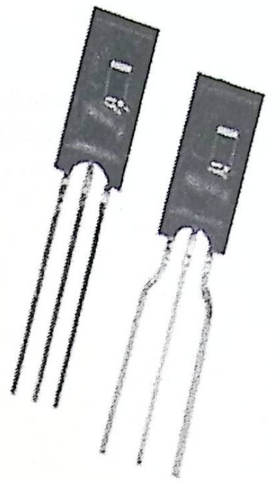
Note 9: Quiescent current is defined in the circuit of *Figure 1*.

Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

Honeywell



HIH-4000 Series

Humidity Sensors

The HIH-4000 Series Humidity Sensors are designed specifically for high volume OEM (Original Equipment Manufacturer) users. Direct input to a controller or other device is made possible by this sensor's linear voltage output. With a typical current draw of only 200 µA, the HIH-4000 Series is often ideally suited for low drain, battery operated systems. Tight sensor interchangeability reduces or eliminates OEM production calibration costs. Individual sensor calibration data is available.

The HIH-4000 Series delivers instrumentation-quality RH (Relative Humidity) sensing performance in a competitively priced, solderable SIP (Single In-line Package). Available in two lead spacing configurations, the RH sensor is a laser trimmed, thermoset polymer capacitive sensing element with on-chip integrated signal conditioning. The sensing element's multilayer construction provides excellent resistance to most application hazards such as wetting, dust, dirt, oils and common environmental chemicals.

FEATURES

- Molded thermoset plastic housing
- Linear voltage output vs %RH
- Laser trimmed interchangeability
- Low power design
- High accuracy
- Fast response time
- Stable, low drift performance
- Chemically resistant

TYPICAL APPLICATIONS

- Refrigeration equipment
- HVAC equipment
- Medical equipment
- Drying
- Metrology
- Battery-powered systems
- OEM assemblies

HIH-4000 Series

TABLE 1. PERFORMANCE SPECIFICATIONS (At 5 Vdc supply and 25 °C [77 °F] unless otherwise noted.) (%RH performance specifications include test system measurement errors (±0.5 % typical.)

Parameter	Minimum	Typical	Maximum	Unit
Interchangeability (best fit straight line)	–	–	–	–
0 % to 60 %	-5	–	5	%RH
60 % to 100 %	-8	–	8	%RH
Interchangeability (2nd order curve)	–	±3.5	–	%RH
Accuracy[1] (best fit straight line)	–	±3.5	–	%RH
Accuracy (2nd order curve)	–	±2.5	–	%RH
Hysteresis	–	3	–	%RH
Repeatability	–	±0.5	–	%RH
Settling time	–	–	70	ms
Response time (1/e in slow moving air)	–	15	–	s
Stability[2] (@ 50 %RH)	–	±1.2 (per year)	–	%RH
Stability[3] (@ 50 %RH)	–	±0.5 (per year)	–	%RH
Voltage supply	4	–	5.8	Vdc
Current supply	–	–	500	µA
Voltage output (1st order fit)	$V_{out}=V_{supply}(0.0062(\text{sensor RH})+0.16)$			
Voltage output (2nd order curve fit)	$V_{out}=0.00003(\text{sensor RH})^2+0.0281(\text{sensor RH})+0.820$, typical @ 25 °C			
Temperature compensation	$V_{out}=(0.0305+0.000044T-0.0000011T^2)(\text{Sensor RH})+(0.9237-0.0041T+0.000040T^2)$, T=Temperature in °C			
Operating temperature	-40[-40]	See Figure 1.	85[185]	°C[°F]
Operating humidity	0	See Figure 2.	100	%RH
Storage temperature	-40[-40]	–	125[257]	°C[°F]
Storage humidity		See Figure 2.		%RH

Notes:
1. For HIH-4000-003 and -004 only.
2. Specification includes testing outside of recommended operating zone.
3. Specification includes testing for recommended operating zone only.

NOTICE
- Do not expose sensor to condensing environments. Exposure to condensing environments will cause sensor output to indicate 0 %RH.
- Sensor is light sensitive. For best performance, shield sensor from bright light.
- Sensor is static sensitive. Sensor connection protected to 15 kV maximum.
- Sensor output is ratiometric to supply voltage.

Failure to comply with these instructions could result in death or serious injury.

CAUTION
ELECTROSTATIC SENSITIVE DEVICES
DO NOT OPEN OR HANDLE EXCEPT AT A STATIC FREE WORKSTATION

ESD SENSITIVITY: CLASS 3

FACTORY CALIBRATION DATA

HIH-4000 Sensors may be ordered with a calibration and data printout (Table 2). See order guide on back page.

TABLE 2. EXAMPLE DATA PRINTOUT

Model	HIH-4000-001
Channel	92
Wafer	030996M
MRP	337313
Calculated values at 5 V V_{out} @ 0 %RH V_{out} @ 75.3 %RH	 0.958 V 3.268 V
Linear output for 2 %RH accuracy @ 25 °C Zero offset Slope RH	 0.958 V 30.680 mV/%RH (V_{out}-zero offset)/slope (V_{out}-0.958)/0.0307
Ratiometric response for 0 % to 100 %RH V_{out}	V_{supply} (0.1915 to 0.8130)

FIGURE 1. RECOMMENDED OPERATING CONDITIONS

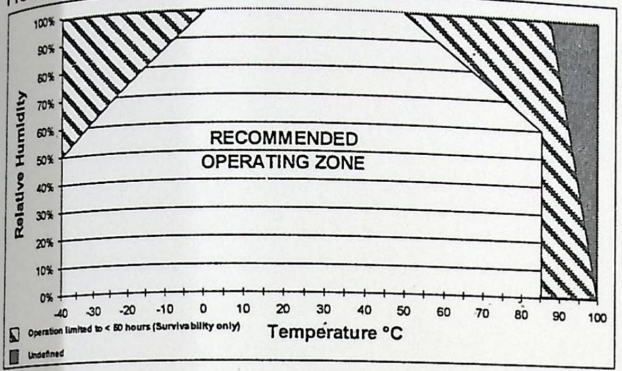


FIGURE 2. STORAGE ENVIRONMENT

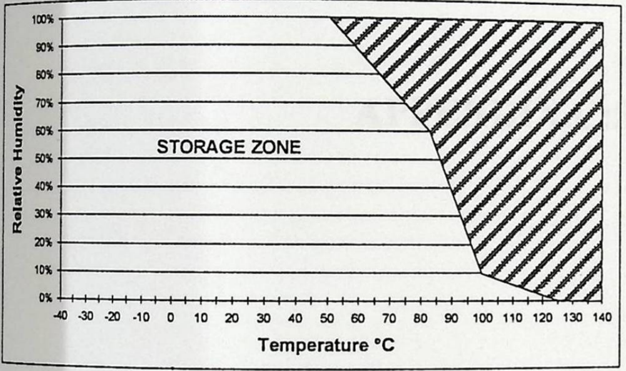


FIGURE 3. MOUNTING DIMENSIONS
for reference only mm/[in]

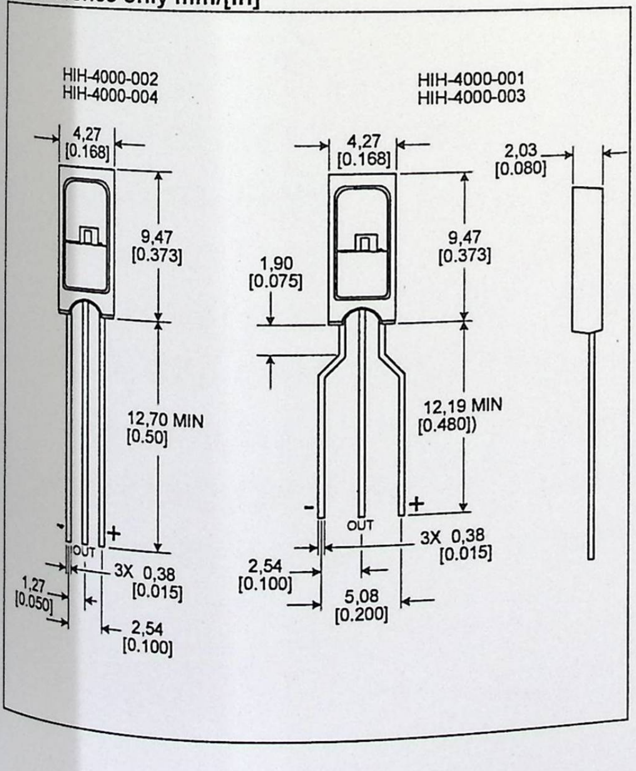


FIGURE 4. TYPICAL BEST FIT STRAIGHT LINE

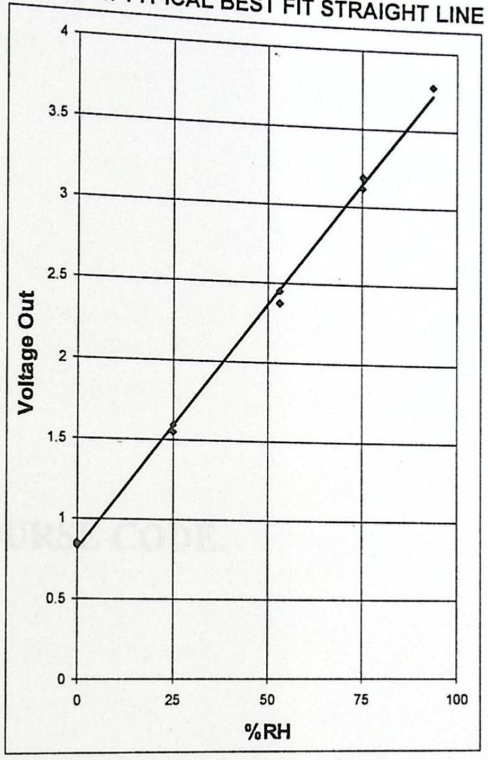
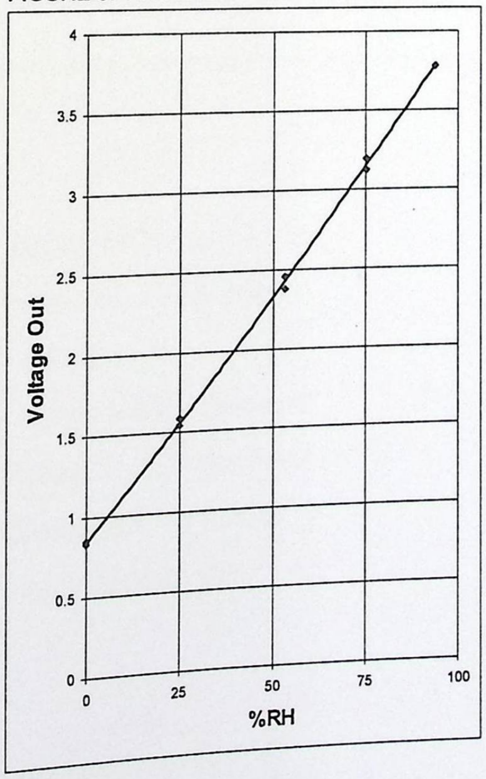


FIGURE 5. TYPICAL 2nd ORDER CURVE FIT



APPENDIX C:SOURSE CODE.

```
#define THIS_IS_STACK_APPLICATION
#define VERSION              " v4.02"
#define BAUD_RATE            (115200)
#define MAX_USER_RESPONSE_LEN   (20u  )
#include "TCPIP Stack/TCPIP.h"
#include "1wire.h"
#include "ftoa.h"

#define VAR_ANAIN_AN3            (0 x03)  // Analog input (LM35)
#define VAR_ANAIN_AN4            (0x04)  // Analog input (HIH400)
#define VAR_TEMP_THRSHOLD  (0x05)  // Temperature Threshold
#define VAR_RH_THRSHOLD          (0x06)  // RH Threshold

#define VAR_FAN_RELAY            ( 0x21)  // Status (on/off) of Fan relay
#define VAR_HEATER_RELAY         (0x22)  // Status (on/off) of Heater relay
#define VAR_BUZZER                (0x23)  // Status (on/off) of Buzzer
#define VAR_PASSWORD             (0x29)  // Admin Password
#define VAR_CURR_USER             (0 x30)  // Guest or Admin

************************************************************************/
 *CGI Command codes (CGI_CMD_DIGOUT)
/************************************************************************

#define CGI_CMD_DIGOUT           (0)

#define CMD_FAN                  (0x01)  // Toggle FAN relay

#define CMD_HEATER               (0x02)  // Toggle HEATER relay

#define CMD_ADMIN_LOGIN0         (0 x03)  // Admin login

#define CMD_CHNG_PASS            (0x04)  // Change Admin Password

#define CMD_CHNG_TEMP            (0x05)  // Change Temp. Threshold

#define CMD_CHNG_RH              (0x06)  // Change RH Threshold

#define CMD_BUZZER               (0x07)  // Toggle Buzzer relay
```

```
static void InitializeBoard(void);
static void ProcessIO(void);
static void ProcessTemp(float);
static void ProcessRH(float);

#if defined(__18CXX)
#if defined(__18F45)
#if defined(__DEBUG)
#pragma config OSC=HSPLL, WDT=OFF, LVP=OFF, PWRT=OFF
#else
#pragma config OSC=HSPLL, WDT=OFF, LVP=OFF, PWRT=ON
#endif
#endif
#endif


BYTE Temp_Threshold[8] = "29;"
float TempLimit = 29;
BYTE RH_Threshold[8]   = "20;"
float RHLimit = 20;



BYTE ADMIN_PASSWORD[10] = "admin;"
BYTE ADMIN_PAGE[20] = "admin.cgi;"
BYTE GUEST_PAGE[20] = "guest.cgi;"
BYTE CURR_USER[20];  // Guest or Admin
```

```
//Buffers for the two ADC readings
BYTE AN3String[8];
BYTE AN4String[8];


//Variables for sampling timer
TICK lastTickSample;
TICK currentTickSample;
TICK tickTimeout;



*****************************************************************/
 *Main application entry point
/*****************************************************************


void main(void){
        Delay10KTCYx(0); // Delay of about 250mS
        InitializeBoard();
        TickInit();
        MPFSInit();
        InitAppConfig();

        if (BUTTON_IO == 0u){

                SetConfig() ;

        {

        StackInit () ;
        HTTPInit ();
        putrsUSART("\r\nIP address");
        putrsUSART("\r\n");


        while (1){

                StackTask () ;
                HTTPServer () ;
                //Our I/O PIC-WEB tasks
                ProcessIO () ;
                }


    }


static void ProcessIO(void){
        float f;
        unsigned char buf[10];
        signed int temp;
        float lm35, HIH4000;
        WORD_VAL ADCResult;



        //Read AN3 (LM35)
        ADCON0      0 =  b10011001; //  Select AN3 channel
        ADCResult.v[0] = 100;
        while (ADCResult.v[0 ]--) ;
        ADCON0bits.GO = 1 ;
        while (ADCON0bits.GO);
        ADCResult.v[0] = ADRESL;
        ADCResult.v[1] = ADRESH;


        //Calculate the temperature based on the LM35 reading
        lm35 = ADCResult.Val;
        lm35 = (lm35 * 4.88)/10;
        ProcessTemp(lm35 );


        //Read AN4 (HIH4000)
        ADCON0      0 =  b10100001;         //  Select AN4 channel
        ADCResult.v[0] = 100;
        while (ADCResult.v[0 ]--);
        ADCON0bits.GO = 1 ;
        while (ADCON0bits.GO );
        ADCResult.v[0] = ADRESL;
        ADCResult.v[1] = ADRESH;
        //Calculate the RH based on the HIH400 reading
        HIH4000 = ADCResult.Val;
        HIH4000 = ((0.00488 * HIH4000) - 0.958)/0.0307;
        ProcessRH(HIH4000 );



        currentTickSample = TickGet () ;

        if (TickGetDiff(currentTickSample, lastTickSample) >= tickTimeout)
        {
                lastTickSample = TickGet();

                //Read the temperature and store is displayable form
                ftoa(HIH4000, AN4String, 1, 'f');      // Store HIH4000 ASCII value
                ftoa(lm35, AN3String, 1, 'f'); // Store LM35 ASCII value

                tickTimeout = TICK_SECOND * 2;  // 5 seconds before next sample

        }
}

static void InitializeBoard(void){

        FAN_RELAY_TRIS = 0;
        FAN_RELAY_IO   = 0;
        HEATER_RELAY_TRIS = 0;
        HEATER_RELAY_IO   = 0;
        BUZZER_TRIS = 0;
        BUZZER_IO   = 0;
        RELAY4_TRIS = 0;
        RELAY4_IO   = 0;
        #if defined(__18F452)

        TRISA  = 0x2F;

        ADCON1=0b11000100;

}

ROM char COMMANDS_OK_PAGE[]   = "INDEX.CGI;"
ROM char CMD_UNKNOWN_PAGE[]   = "INDEX.CGI;"
ROM char CMD_ADMIN_PAGE[]     = "admin.CGI;"


#define COMMANDS_OK_PAGE_LEN      (sizeof(COMMANDS_OK_PAGE)
#define CMD_UNKNOWN_PAGE_LEN  (sizeof(CMD_UNKNOWN_PAGE)
#define CMD_ADMIN_PAGE_LEN            (sizeof(CMD_ADMIN_PAGE)



void HTTPExecCmd(BYTE** argv, BYTE argc){
        BYTE command;
        BYTE var;
        BYTE CurrentArg;
        BYTE* TmpPointer;
        char* TmpChar;
        WORD_VAL TmpWord;
        int i = 0;

        command = argv[0][0] - '0;'

        switch (command){


        case CGI_CMD_DIGOUT:
                //Convert string to integer value
                var = argv[1][0] - '0;'
                switch (var)


                case CMD_HEATER:
                        //Turn on HEATER Relay
                        HEATER_RELAY_IO = 1;
                        break;
```

```
case CMD_BUZZER:
        //Turn off Buzzer Relay
        BUZZER_IO = 0;
        break;
case CMD_ADMIN_LOGIN:
        //Admin Login
        TmpChar = argv[2];
        if(strcmp(TmpChar, ADMIN_PASSWORD) == 0)

{
                //Correct Password

                 i = 0;
                while( i < 20)
                {
                        CURR_USER[i] = ADMIN_PAGE[I ];
                        i ++;
                }


          else
        {       //Wrong Password
                i = 0;
                while(i < 20)
                {
                        CURR_USER[i] = GUEST_PAGE[I ];
                        i++ ;

                }

          }
        break;


case CMD_CHNG_TEMP:

        //Change Temp. Threshold
        i = 0;
        while(argv[2][i] != '\0' && i < 5)
        {
                Temp_Threshold[i] = argv[2][i];
                i ++ ;
        }
```

```
                    TempLimit = Temp_Threshold[0] - '0;'
                    TempLimit += (Temp_Threshold[1] - '0')*10;

                    break;


                case CMD_CHNG_RH:
                    //Change RH Threshold
                    i = 0;
                    while(argv[2][i] != '\0' && i < 5)
                    {
                            RH_Threshold[i] = argv[2][i];
                            i++ ;
                    }
                    RHLimit = RH_Threshold[0] - '0;'
                    RHLimit = (RH_Threshold[1] - '0')*10;
                    break;


                case CMD_CHNG_PASS:
                    //Change Admin. Password
                    i = 0;
                    while(i < 8)
                    {
                            ADMIN_PASSWORD[i] = argv[2][i];
                            i ++ ;
                    }
                    break;
}


    memcpypgm2ram((void*)argv[0], (ROM void*)COMMANDS_OK_PAGE,
COMMANDS_OK_PAGE_LEN );
            break;


    default:
            memcpypgm2ram((void*)argv[0], (ROM
void*)COMMANDS_OK_PAGE, COMMANDS_OK_PAGE_LEN) ;

            break;
    }
```

```
WORD HTTPGetVar(BYTE var, WORD ref, BYTE* val){

    static BYTE VarString[25];

        switch (var){



            //LM35
    case VAR_ANAIN_AN3:
            *val = AN3String[(BYTE)ref];
            if (AN3String[(BYTE)ref] == '\0' )
            return HTTP_END_OF_VAR;
            else if (AN3String[(BYTE)++ref] == '\0')
            return HTTP_END_OF_VAR;
            return ref;



            //HIH4000
    case VAR_ANAIN_AN4:
            *val = AN4String[(BYTE)ref];
            if (AN4String[(BYTE)ref] == '\0')
            return HTTP_END_OF_VAR;
            else if (AN4String[(BYTE)++ref] == '\0')
            return HTTP_END_OF_VAR;
            return ref;



            //FAN relay
    case VAR_FAN_RELAY:
            *val = FAN_RELAY_IO ? '1':'0;'
            break;



            //Heater Relay
    case VAR_HEATER_RELAY:
            *val = HEATER_RELAY_IO ? '1':'0;'
            break;
        //Buzzer
case VAR_BUZZER:
        *val = BUZZER_IO ? '1':'0;'
        break;


                //Temp Threshold
        case VAR_TEMP_THRSHOLD:
        *val = Temp_Threshold[(BYTE)ref];
        if (Temp_Threshold[(BYTE)ref] == '\0 ')
        return HTTP_END_OF_VAR;
        else if (Temp_Threshold[(BYTE)++ref] == '\0')
        return HTTP_END_OF_VAR;
        return ref;


        //RH Threshold
case VAR_RH_THRSHOLD:
        *val = RH_Threshold[(BYTE)ref];
        if (RH_Threshold[(BYTE)ref] == '\0 ')
        return HTTP_END_OF_VAR;
        else if (RH_Threshold[(BYTE)++ref] == '\0 ')
        return HTTP_END_OF_VAR;
        return ref;


        //Password
case  VAR_PASSWORD:
        *val = ADMIN_PASSWORD[(BYTE)ref];
        if (ADMIN_PASSWORD[(BYTE)ref] == '\0')
        return HTTP_END_OF_VAR;
        else if (ADMIN_PASSWORD[(BYTE)++ref] == '\0')
        return HTTP_END_OF_VAR;
        return ref;
```

```
            //Current User
        case VAR_CURR_USER:
            *val = CURR_USER[(BYTE)ref];
            if (CURR_USER[(BYTE)ref] == '\0'')
            return HTTP_END_OF_VAR;
            else if (CURR_USER[(BYTE)++ref] == '\0')
            return HTTP_END_OF_VAR;
            return ref;



static void ProcessTemp(float temp)
{
        if(temp > TempLimit + 1)
        {
                //Turn heater off
                HEATER_RELAY_IO = 0;
        }

        if((temp > (TempLimit + 3)) || (temp < (TempLimit – 3)))
        {
                //Turn on Buzzer
                BUZZER_IO = 1;
        }
}



static void ProcessRH(float R)
{       if((RH > (RHLimit + 3)) || (RH < (RHLimit – 3)))
        {
                //Turn on Buzzer
                BUZZER_IO = 1;
        }
}
```