1.1Background

A greenhouse is a specially designed farm structure building to provide a controllable environment for better crop production, crop protection, crop seeding and transplanting. Moreover, the available space of land for cultivating crops has been significantly decreasing, since more space of land is heavily used for housing and industries in this modern area. In most tropical countries, the use of greenhouse has been growing for commercially horticulture (i.e. fruits, fresh flowers and vegetables) production .A greenhouse environment is an incredibly complex and dynamic environment and it strongly influences crop cultivation. The efficiency of plant production in greenhouses depends significantly on the adjustment of optimum climate growth conditions to achieve high yield at low expense, good quality and low environmental load. To achieve these goals several parameters such as air temperature, humidity, light intensity. Continuous monitoring and control of these environmental factors gives relevant information pertaining to the individual effects of the various factors towards obtaining maximum crop production.

Greenhouse environments present unique challenges to good control. Temperature changes occur rapidly and vary widely depending on solar radiation levels, outside temperatures and humidity levels, wind speed and direction and the amount of plant material in the greenhouse. Poor light intensity and high humidity often result in poor fruit set and quality. Proper control of plant disease is critical in greenhouse environments, where high temperatures and high humidity are ideal for diseases to develop.

1.2 Problem Statement

The traditional method used to run the greenhouses in so many places do not normally include any automatic control. The irrigation is done manually by human operator, and it is well known that such control is not optimum and always exposed to faults.

This thesis solved these problems by design full smart monitoring and control in Greenhouse that make the communication between Greenhouse stations and remote server easy and simple using database that facility to predict the environmental behaviors according to analytical results obtained from collected data.

1.3 Objectives

The objective of this Project is to design and implement simple, easy to install, microcontroller-based circuit to monitor and control the values of temperature, humidity and light of the natural environment that are continuously modified and controlled in order optimize them to achieve maximum plant growth and yield.

To work out the pre mentioned problem and find the best solution, different sensors such as humidity, temperature, and light sensors can be utilized to measure the related data, feed it back to a monitoring station and take relevant control actions at the same time without human intervention.

1.4 Methodology

To be able to make a full system design, a system model was done .and then simulated using microcontroller based system (Proteus & Micro C software's to simulate the system) to process data sensed and send this Information using virtual serial to remote server and displayed it by visual studio in server and

LCD . The database to be able to display and store a history data. Then make hardware implementation to the circuit .the circuit was tested and its satisfy the requirements.

1.5 Layout of the Thesis

This Thesis contains five chapters. Chapter one is the introduction. It presents the problem statement of the project, the methodology and objectives. Chapter two introduces the Literature review. Chapter three presents the research methodology. Chapter four includes the results and discussions. Finally Chapter five includes the conclusion and recommendations.

2.1 Introduction

This chapter reviews the fundamental concepts and principles that the project relies on; also it gives a brief and fast knowledge about the alternative technologies that precede to the same goals of the project as well as the basic theory of the project components.

2.2 Greenhouse

Monitoring and control of greenhouse environment play a significant role in greenhouse production and management. To monitor the greenhouse environment parameters effectively, it is necessary to design a control system. Here controlling process takes place effectively by both manual and automatic manner. For manual control purpose RS232 is used, which will send status of greenhouse environment automatic control process. To control room. There we can control the activities through PC and send to controller back which is in greenhouse environment. There it will activate the actuator according to our wish. The main objective is to design a simple, easy to install, microcontroller-based circuit to monitor and record the values of temperature, humidity, and sunlight of the natural environment that are continuously modified and controlled in order optimize them to achieve maximum plant growth and yield. PIC 16F877A controller is used. It communicates with the a variety of sensor modules in order to control the light, aeration and drainage process efficiently inside a greenhouse by actuating a cooler, fogger, dripper and lights respectively according to the necessary condition of the crops.[10]

2.3 Microcontroller

A microcontroller is a single-chip computer. Micro suggests that the device is small and controller suggests that it is used in control applications. Another term for microcontroller is embedded controller, since most of the microcontrollers are built into (or embedded in) the devices they control [1].

A microcontroller is a very powerful tool that allows a designer to create sophisticated input-output data manipulation under program control. Microcontrollers are classified by the number of bits they process. Microcontrollers with 8 bits are the most popular and are used in most microcontroller-based applications. Microcontrollers with 16 and 32

bits are much more powerful, but are usually more expensive and not required in most small-size or medium-size general purpose applications that call for microcontrollers. [1]

2.3.1 CPU

The CPU is the brain of the microcontroller and this is where all of the arithmetic and logic operations are performed. The CU controls the internal operations of the microprocessor and sends out control signals to other parts of the microcontroller to carry out the required instructions. [3]

2.3.2 Memory

Memory is an important part of a microcontroller system. Depending upon the type used us can classify memories into two groups: program memory and data memory. Program memory stores the program written by the programmer and this memory is usually non-volatile, i.e. data is not lost after the removal of power. Data memory is where the temporary data used in a program are Stored and this memory is usually volatile, i.e. data is lost after the removal of power. [3]

- **RAM**

  RAM means Random Access Memory. It is a general-purpose memory which usually stores the user data used in a program. RAM is volatile, i.e. data is lost after the removal of power. Most microcontrollers have some amount of internal RAM. 256 bytes is a common amount, although some microcontrollers have more, some less. In general it is possible to extend the memory by adding external memory chips. [3]

- **ROM**

  ROM is Read Only Memory. This type of memory usually holds program or fixed user data. ROM memories are programmed at factory during the manufacturing process and their contents cannot be changed by the user. ROM memories are only useful if you have developed a program and wish to order several thousand copies of it. [3]

- **EPROM**

  EPROM is erasable Programmable Read Only Memory. This is similar to ROM, but the EPROM can be programmed using a suitable programming device. EPROM memories have a small clear glass window on top of the chip where the data can be erased under UV light. Many development versions of microcontrollers are manufactured with EPROM memories where the user program can be stored. These memories are erased and re-programmed until the user is satisfied with the program. Some versions of EPROMs, known as OTP (One Time Programmable), can be programmed using a suitable programmer device but these memories cannot be erased. OTP memories cost much less than the EPROMs. OTP is useful after a project has been developed completely and it is required to make many copies of the program memory. [3]

- **EEPROM**

  EEPROM is Electrically Erasable Programmable Read Only Memory, which is a non-volatile memory. These memories can be erased and also be programmed under program control. EEPROMs are used to save configuration information, maximum and minimum values, identification data, etc. Some microcontrollers have built-in EEPROM memories (e.g. PIC16F84 contains a 64-byte EEPROM memory where each byte can be programmed and erased directly by software). EEPROM memories are usually very slow. [3]

- **Flash EEPROM**

  This is another version of EEPROM-type memory. This memory has become popular in microcontroller applications and is used to store the user program. Flash EEPROM is non-volatile and is usually very fast. The data is erased and then re-programmed using a programming device. The entire contents of the memory should be erased and then re-programmed. [3]

2.4 A/D Converter

Certain PIC pins can be set up as inputs to an analog-to-digital converter (ADC). The' 877 has eight analog inputs, which are connected to Port A and Port E. When used in this mode, they are referred to as AD0–AD7. The necessary control registers are initialized in CCS C using a set of functions that allow the ADC operating mode and inputs to be selected. An additional "device" directive at the top of the program sets the ADC resolution. An analog voltage presented at the input is then converted to binary and the value assigned to an integer variable when the function to read the ADC is invoked. The default input range is set by the supply (nominally 0–5 V). If a battery supply is used (which drops over time) or additional accuracy is needed, a separate reference voltage can be fed in at AN2 (_ V ref) and optionally AN3 (–Vr ef). If only _ V ref is used, the lower limit remains 0 V, while the upper is set by the reference voltage. This is typically supplied using a zener diode and voltage divider. The 2.56 V derived from a 2V7 zener gives a conversion factor of 10 mV per bit for an 8-bit conversion. For a 10-bit input, a reference of 4.096 V might be convenient, giving a resolution of 4 mV per bit. [4]

The interpolating and averaging ADC is based on the architecture of the flash ADC, which is the fastest of all ADC architectures. It tries to overcome the high power consumption disadvantage of the flash ADC by using analog preprocessing like pre-amplifying, interpolating, folding and averaging techniques [3].

As a result, lower input capacitance is seen by the input signal because the comparators are placed after the analog preprocessing. By making sure that the interpolation network does not load the preamplifiers, power can be saved [1].

The basic idea of an interpolating and averaging ADC is that the sampled input analog signal will go through a number of pre-amplification stages before the comparison and digitization actually takes place. In between the pre-amplification stages, interpolation will be done to get the required resolution of the digital output. Averaging at the output of the interpolation network with the help of passive elements can improve the accuracy of the digitization. [2]

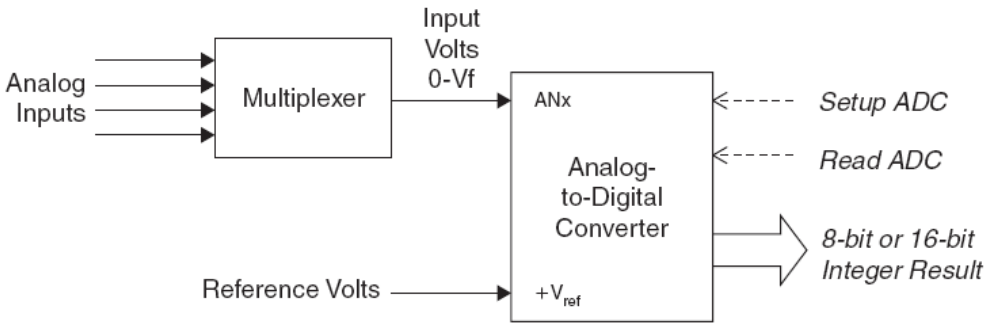

Figure 2.1**:** Analog To Digital Convertor

2.5 Oscillator

Crystal oscillators are widely used to generate accurate reference frequency in electronic systems. However, constant frequency comes at the expense of higher power consumption and thereby affecting the life of battery, especially in low-power microcontroller unit (MCU) and watch system. This issue is extremely important in the node of electronic system in mobile society. [5]

Low power CMOS crystal oscillators have either been optimized for low current or for have low supply voltage .But in most cases, low supply voltage cannot satisfy with applications, on the other hand LDO additional will also Cause the additional current. So the most important point for a low current consumption is an amplitude control, which reduces the supply current as soon as the oscillator amplitude reaches a reasonable value. [5]

- **Conventional Crystal Oscillator Circuit**

The conventional crystal oscillator circuit widely used in electronic system is based on structure of pierce. The schematic is shown in the Fig. The conventional pierce crystal oscillator consists of two parts. One is an inverting amplifier that supplies a voltage gain and 180 degree phase shift. The other is a frequency selective feedback path, which is out of the chip. The crystal Combined with C1 and C2 to form a feedback network that tends to stabilize the frequency and supply 180 degree phase shift to the feedback path because of the π network. These conditions conform with the Barkhausen criterion of Oscillation that overall phase shifts is zero and a closed loop

gain should be over or equal to one. The feedback resistance Rf, is used to bias the inverting Amplifier to stabilize the static operating point of amplifier. Generally the feedback resistance doesn't require precise resistor but large numerical value. And so we can use large length and small width transistor instead of Rf. [5]

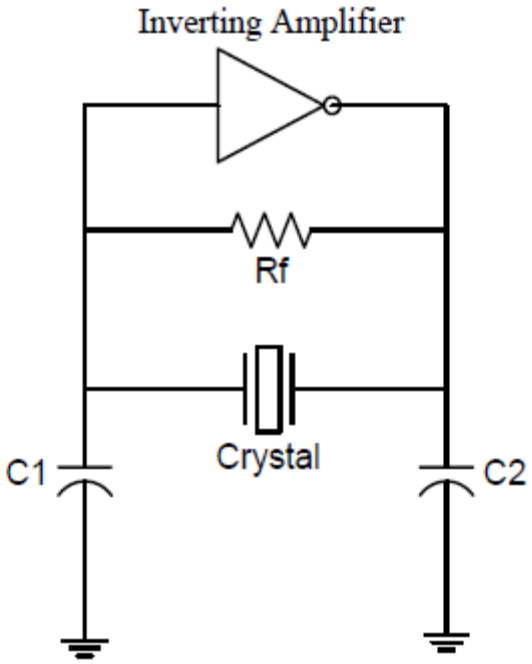

Figure 2.2**:** Schematic of conventional pierce crystal oscillator.

- **Crystal Model**

  Crystal is the main component of generating oscillation clock signal. However, in simulation we only use its equivalent circuit instead of a crystal. Fig shows the equivalent circuit of crystal. R is the effective series resistance in the crystal, as well as L and Cs are the motional inductance and capacitance of the crystal. Cp is the parasitic shunt capacitance due to the electrodes. In parallel resonant mode, the crystal will look and perform like a low resistance. For generating 32.768 kHz signal, we set L =47.22H, Cs=0.5pF, Cp=100Pf. [5]

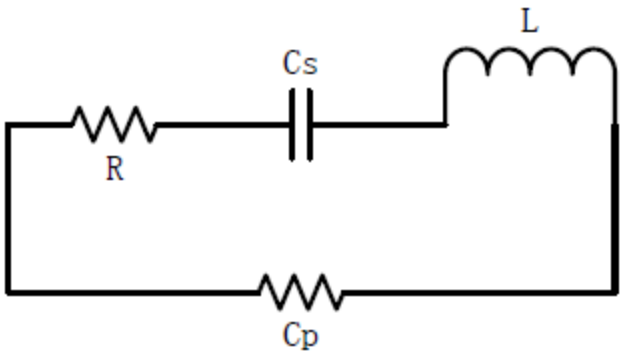

Figure 2.3**:** The equivalent circuit of crystal.

When the crystal is operating at series resonance, it looks. Purely resistive and the series resonance frequency is given by

Fs = 1/2∏√LC 2.1

When the crystal is operating at parallel resonance, it looks inductive. And the parallel resonance frequency is given by

Fs =1/2∏√L Cs Cp/Cs+Cp 2.2 [5]

2.6 Voltage Regulator (7805)

Industry is pushing towards complete system-on-chip (SoC) design solutions that include power management. The study of power management techniques has increased spectacularly within the last few years corresponding to a vast increase in the use of portable, handheld battery operated devices. Power management seeks to improve the device's power efficiency resulting

In prolonged battery life and operating time for the device. Power management system contains several subsystems including linear regulators, switching regulators, and control logic. The control logic changes the attributes of each subsystem; turning the outputs on and off as well as changing the output voltage levels, to optimize the power consumption of the device. [6]

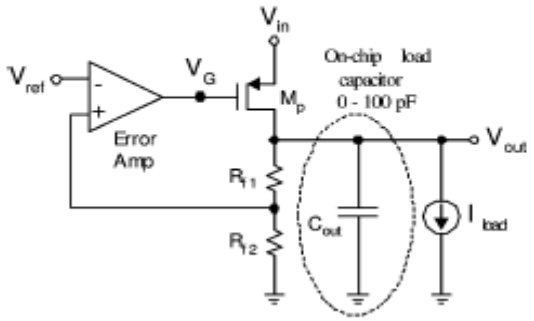

Figure 2.4**:** Voltage Regulator (7805)

2.7 Liquid Crystal Display

Liquid Crystal Display (LCD) has given a new demarcation to the display devices. The liquid crystals are used to display image in thin, light computer. LCD screens are used in most laptop computers as well as in flat panel monitors. It has replaced conventional cathode ray tube (CRT) monitors. The CRTs were preferred for their superior color presentation by graphics and photography professionals. The constant improvements in LCDs technology have, however, made the performance nearly comparable and the differences less noticeable. [7]

LCD panels are transmissive and could not emit light on their own and therefore require backlights to generate colors on LCD screen. Backlight structure is different in various applications. Edge type backlights are used for notebooks and monitors, whereas, direct types are used for LCD TVs. Light sources used in backlight are also various, which include cold cathode fluorescent lamp (CCFL), external electrode fluorescent lamp (EEFL), hot cathode fluorescent lamp (HCFL), flat fluorescent lamp (FFL) and light emitting diode (LED). CCFLs, employed in backlighting units (BLU) of LCD, have many drawbacks including high power consumption, using mercury to create vapor discharges etc. In tube based technology, CCFLs are usually susceptible to failures. The space occupied by CCFLs also constrains in slimming down the thickness of the LCD panel. [7]

Event limited to character-based modules, there is still wide variety of shapes and sizes available. Line lengths of 8,16,20,24 and 40 character are all standard, in one, two and four-line versions. [8]

2.8 Serial communications system

Here have been several communication protocols in the embedded systems like RS-232, Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), Controller Area Network (CAN) and many more. But most of these protocols require prerequisite hardware and sometimes license for the copyrighted Code for the protocol. Hence in simple applications where this amount of complexity in terms of communication as well as rights to use is not essential and only some kind of basic communication is essential, there is a need to develop a simpler interface to overcome this existing complexity. The best Alternative would be to develop a new protocol which may not be the fastest or most efficient, but definitely the simplest, cheapest and open source. Making this protocol independent of the hardware platform/requirement would increase the flexibility of the protocol to a great extent. The development of such protocol has been brought out in this paper which elaborates the details of the protocol, the advantages, the limitations and further possible improvements of the protocol. [9]

2.9 Sensor

Sensor is a device which is used to convert physical quantity into electrical signal. 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's sensitivity indicates how much the sensor's output changes when the measured quantity changes. For instance, if the mercury in a thermometer moves 1 cm when the temperature changes by 1 °C, the sensitivity is 1 cm/°C (it is basically the slope Dy/Dx assuming a linear characteristic). Sensors that measure very small changes must have very High sensitivities. Sensors also have an impact on what they measure; for instance, a room temperature thermometer inserted into a hot cup of liquid cools the liquid while the liquid heats the thermometer. Sensors need to be designed to have a small effect on

what is measured; making the sensor smaller often improves this and may introduce other advantages. [10]

2.9.1 Light Sensor

Light Dependent Resistor (LDR) also known as photoconductor or photocell, is a device which has a resistance which varies according to the amount of light falling on its surface. Since LDR is extremely sensitive in visible light range, it is well suited for the proposed application. [11]

Figure 2.5**:** Light Sensor

2.9.2 Humidity Sensor

The humidity sensor HIH4000, manufactured by Honeywell is used for sensing the humidity. It delivers instrumentation quality RH (Relative Humidity) sensing performance in a low cost, solder able SIP (Single In-line Package) Relative humidity is a measure, in percentage, of the vapour in the air compared to the total amount of vapour that could be held in the air at a given temperature.[11]

Figure 2.6**:** Humidity Sensor

2.9.3 Temperature Sensor

National Semiconductor's LM35 IC has been used for sensing the temperature. It is an integrated circuit sensor that can be used to measure temperature with an electrical output proportional to the temperature. The temperature can be measured more accurately with it than using a thermistor. The sensor circuitry is sealed and not subject to oxidation, etc. [11]

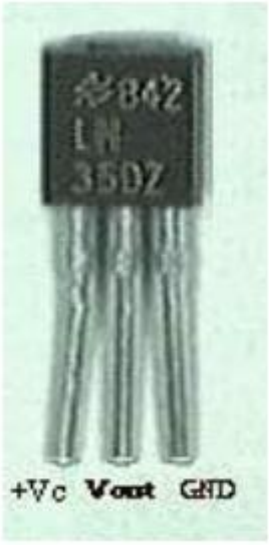

Figure 2.7**:** Temperature Sensor

2.10 Microsoft Visual studio

Visual Studio 2010 (VS) is an integrated development environment (IDE); a set of tools in a single application that helps you write programs. Without VS, you would need to open a text editor, write all of the code, and then run a command-line compiler to create an executable application. The issue with the text editor and command-line compiler is that you would lose a lot of productivity through manual processes. Fortunately, you have VS to automate many of the mundane tasks that are required to develop applications. The following sections explain what VS will do for you and why VS is all about developer productivity. [12]

VS include a suite of project types that you can choose from. Whenever you start a new project, VS will automatically generate skeleton code that can compile and run immediately. Each project type has project items that you can add, and project items include skeleton code. In the next chapter, you'll learn how to create projects, add project items, and view automatically generated code. V offers many premade controls, which include skeleton code, saving you from having to write your own code for repetitive tasks. Many of the more complex controls contain wizards that help you customize the control's havior, generating code based on wizard options you choose. [12]

2.11 DC Motor

Speed control of dc motor could be achieved using mechanical or electrical techniques. In the past, speed controls of dc drives are mostly mechanical and requiring large size hardware to implement. The development has launched these drives back to a position of formidable relevance, which were predicted to give way to ac drives. Some important applications are rolling mills, paper mills mine winders, hoists, machine tools, traction, printing presses, textile mills, excavators and cranes. This paper provides a system that can utilized to use DC motor for various applications. We can utilize the DC Motor for various applications by controlling the speed and orientation according to the field of interest. Pulse Width Modulation (PWM) is the technique of utilizing switching devices to produce. [13]

The effect of a continuously varying analog signal. This PWM conversion generally has very high electrical efficiency and can be used in controlling either a three-phase synchronous motor or a three-phase induction motor .It is desirable to create three perfectly sinusoidal current waveforms in the motor windings, with relative phase displacements of 120°. The production of sine wave power using a linear amplifier system would have low efficiency, maximum of 64%. Efficiency can be increase up to 95% if instead of the linear circuitry, fast electronic switching devices are used, depending on the properties of the semiconductor power switch. The result is a load current waveform that depends mainly on the modulation of the duty ratio. [13]

3.1 Introduction:

This chapter describe weather station monitoring system design that measure the environment statics (temperature-humidity- ambient light) then send this data dynamically after each (10 sec) frequently to the remote server (PC). Using serial connection to link weather station with the remote server. The overall architecture of the System is shown below:

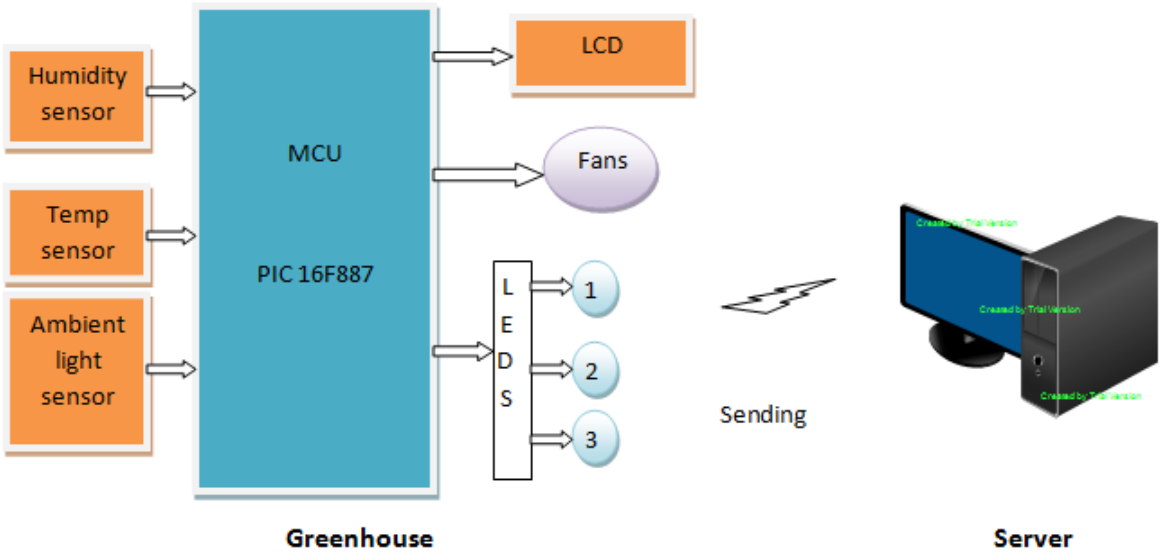

Fig 3.1: Overall system design.

3.2 Greenhouse:

The remote unit has been developed to be deployed at the location where the weather data has to be measured. Is designed to have fully automatic operation using a PIC16F877 microcontroller and is powered through a 5V battery.

3.3 Remote Server:

The main server is a PC that receives data sent by station using serial communication.
The user gets access to the system using visual studio based GUI. A full set of weather data can be received each 10 seconds. The weather data sent to the server saved to the SQLITE database and then displayed using LCD. This data updated dynamically.

3.4 System Specifications:

System Specifications divided for two section station side and Server Side:

3.4.1 Station side:

- System shall provide automatic monitoring of various weather conditions temperature, ambient light and humidity then send this data to remote server (PC).
- Data shall send form Greenhouse to the server each (10 sec) and will be updated dynamically.
- User shall be able to reset the system.
- The system shall have a display that continuously indicates all derived Measurements with their time.
- System shall link stations with the server using serial communication (RS232).
- System LEDs, Fans power (on/off) when measured data exceed specific values.

The table bellows illustrate the massage format sent by the Greenhouse.

Table3.1: Message format.

SOM	time	humidity	light	temperature	EOM

3.4.2 Server Side:

- The server shall receive values sent from greenhouse and their time.
- Sever shall have SQLITE database to get information about the history Of the data sent by greenhouse and this data update dynamically.

- Server shall be able to display sent data and shall be able to search specific data item using data identification (ID).

❖ **PC system requirements:**

Table3.2: PC requirements.

Operating system	Windows 32 bit
Processor	Intel (R) Celeron (R) CPU
RAM	2GB
Hard disk	250 GB

3.5 Hardware Requirements:

- PIC16F877 microcontroller: as a processing unit, its function is determined by a program loaded in it.
- Sensors: LM35 as Temperature sensor, HS101 as Relative Humidity sensor and ambient light sensor (LDR).

All sensors shall be acquired by microcontroller; the signal from the sensor is then processed and sent to the processor board (MCU) to be translated into temperature, ambient light and humidity.

- Serial communication (RS232): as a data sender and receiver, from the processor board to the server side.
- MAX 232: the MAX232 is an IC that converts signals from anRS-232 serial port to signals suitable for use in TTL compatible digital logic circuits. The MAX232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS signals.
- ULN2003: as hardware driver.
- LCD: to display the data measured by station.

- Database: access database to store data sent from station.
- Server: a personal computer with Windows 7 operating is used as a weather information center (pc).
- LEDs, Fans to indicate the measure data exceed specific values.

3.6 Implantations tools:

- Micro C Compiler: is a full-featured ANSI C compiler for PIC devices from Microchip®. It is the best solution for developing code for PIC devices.
- Microsoft visual studio: is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms. Visual Studio supports different programming languages by means of language services. This tool used as Visual C++ to write C program or API to describe sever side.
- SQLITE: to create database.
- Virtual serial port: to make a virtual connection.
- Proteus: to simulate the overall circuit diagram.

3.7 Software design:

The fig bellow describe the finite state machine of the system

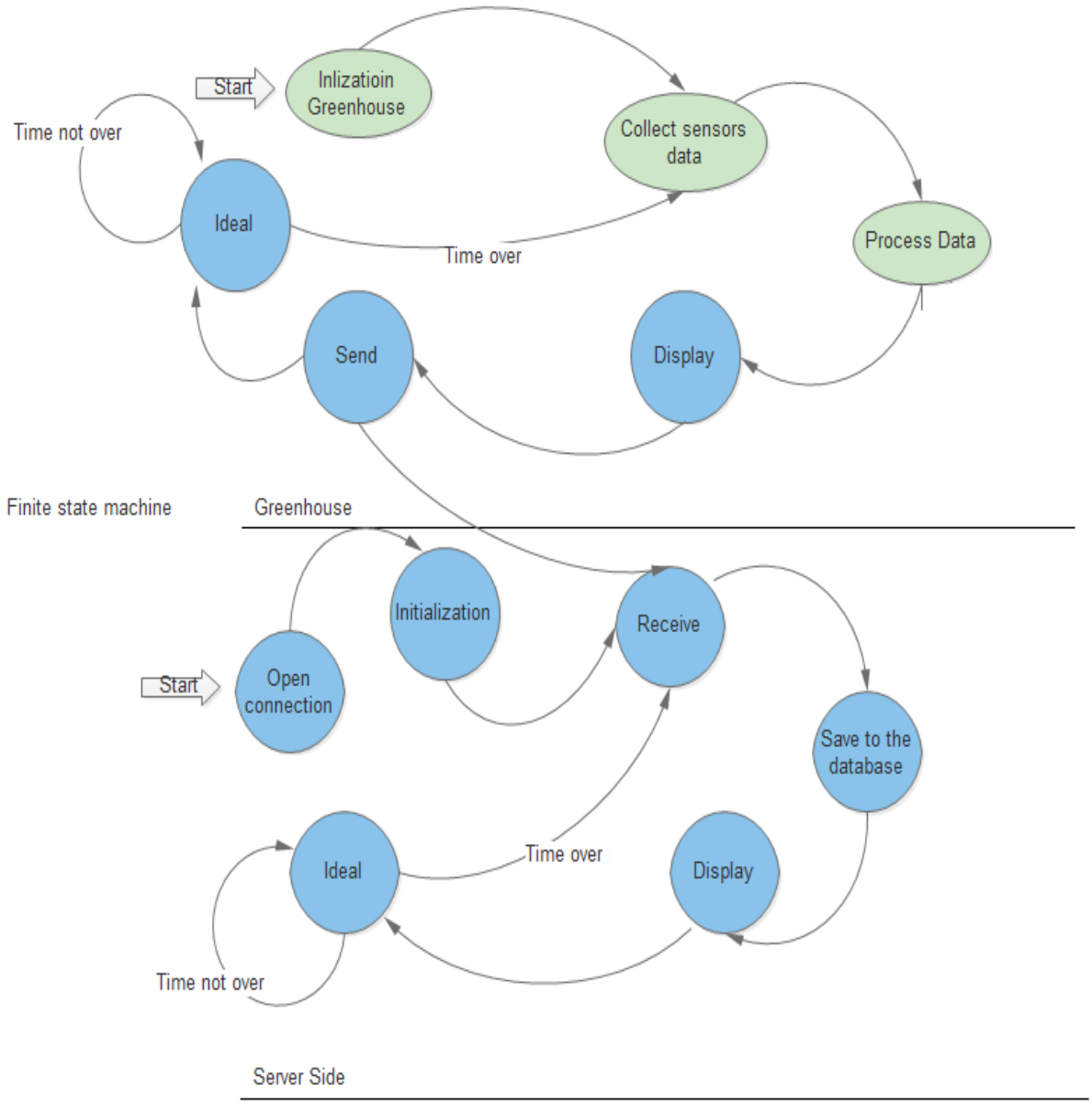

Fig 3.2: Overall finite state machine diagram.

3.8 Implementation:

Implementation divided for two section Software implementation and Hardware implementation:

3.8.1 Software implementation:

Fig(3.4) illustrate the software implementation for Greenhouse which collect data from different sensors and process it by MCU then display it with their time on Greenhouse LCD. Then send it to remote server illustrates in Fig (3.5) using RS232 connection. The two fans powered on according to MCU decision after temperature (above 25c) and humidity (above) reading when specific values exceed. The LEDs turned on or off according to light intensity.

Table 3.3 LEDs reading conditions.

Reading	**LED1**	**LED2**	**LED3**
X<50	on	on	on
50>X>100	on	on	off
X>100	on	off	off

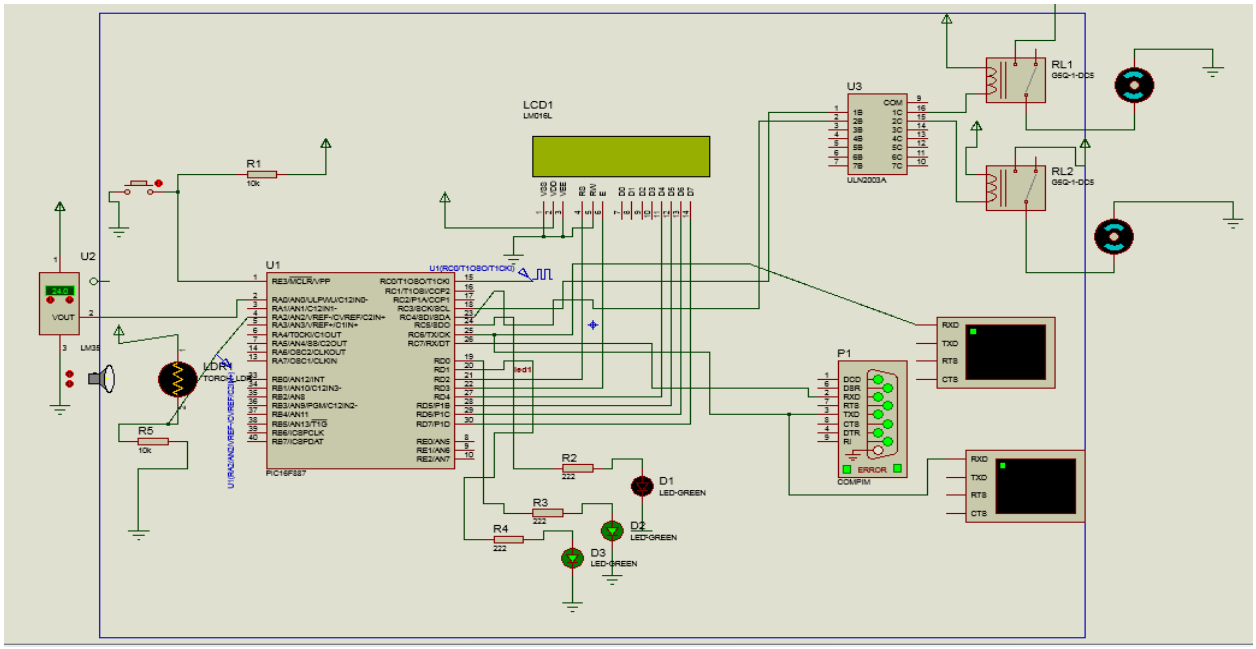

Fig 3.3: Greenhouse implementation in Proteus.

The Fig (3.5) illustrate GUI developed by visual studio 2008 .which receive data sent by greenhouse. And save it to internal database then display it as shown in Fig (3.5).

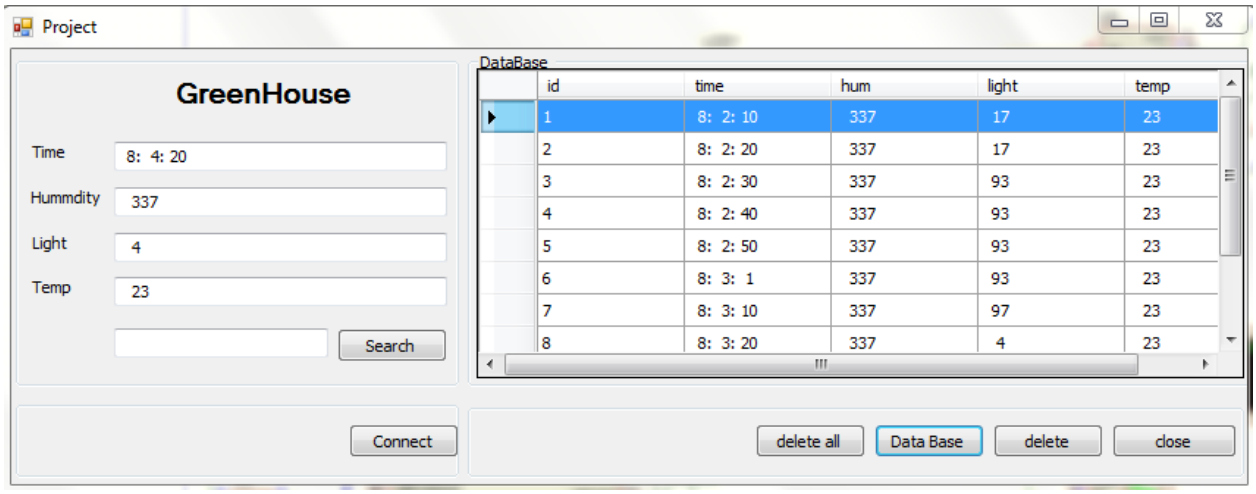

Fig 3.4: Server side GUI.

3.8.2 Hardware implementation:

Fig (3.5) illustrates the hardware implementation step in software.

Figure 3.5: Hardware Implementation Circuit

Results and Discussion:

4.1 Introduction:

This chapter presents discusses and the results obtained upon running the program designed using Proteus and micro c program in the Greenhouse side and visual studio at server side.

4.2 Greenhouse side:

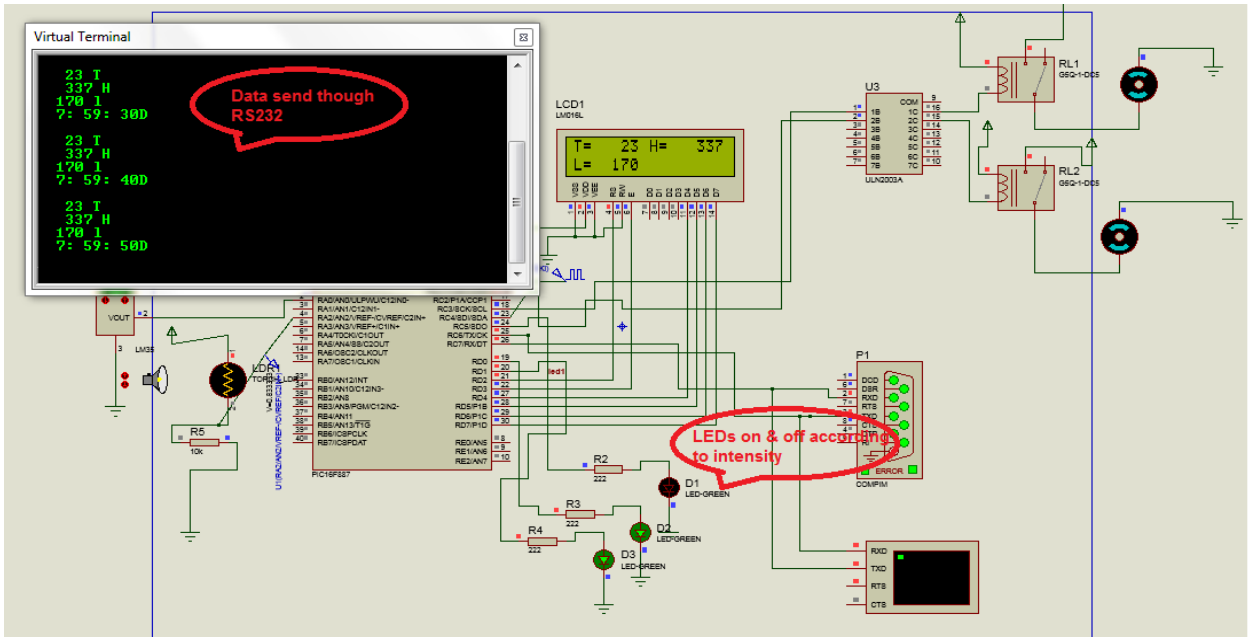

Fig 4.1: Greenhouse results in Proteus.

Fig (4.1) shows that the software simulation of the greenhouse side that collect data from the sensors and sent it through serial link to the remote server .also show data sent each (10 min).And when the temperature exceed the specific value the fan turn on .and the LED's indicate to light intensity as show in fig (4.1).LCD to show the recent data measurements with their time and date.

4.3 Server side.

- **Database description:**

Sever side contain database that save a history data sent by the greenhouse (identification of greenhouse number, time, humidity, light, and temperature). As shown in fig.

Weather system database created by QSLITE code in such a method that it can easily and quickly be re-created, additionally all data existing in the "WEATHER_PORJECT" was also created by the same method.

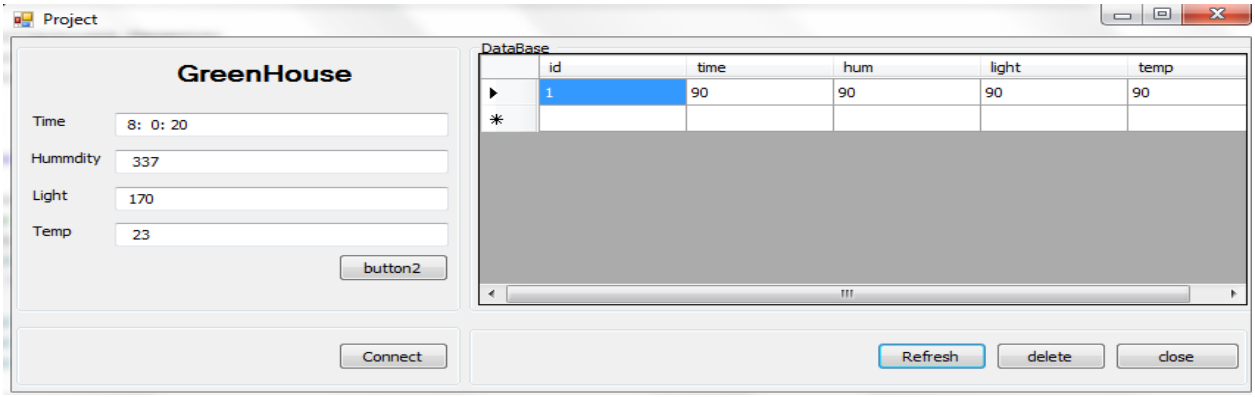

Fig 4.2: Server side GUI and database.

- **Server side GUI description:**

Table 4.1: content all item within the GUI of the sever side and its work.

Item	**Description**
connect	To interface between the greenhouse and server side
Refresh	To restart the GUI
Delete	To delete measurements according to their ID
Close	To close connection between green house and server side

Fig (4.3) Shows that what is happen when the connection open between greenhouse and server side by clicking connect item.

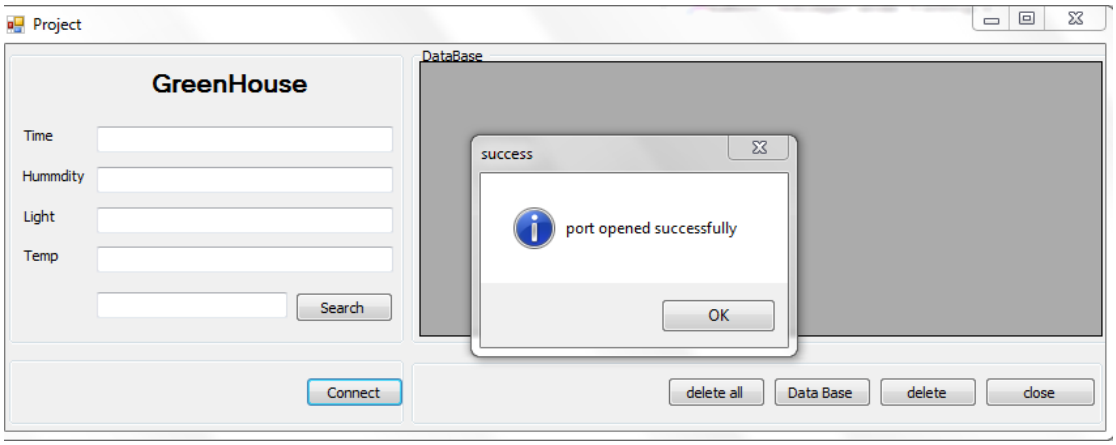

Fig 4.3: open connection.

When the connection failed it send massage to user content that he haven't ability to open connection as shown in fig below.

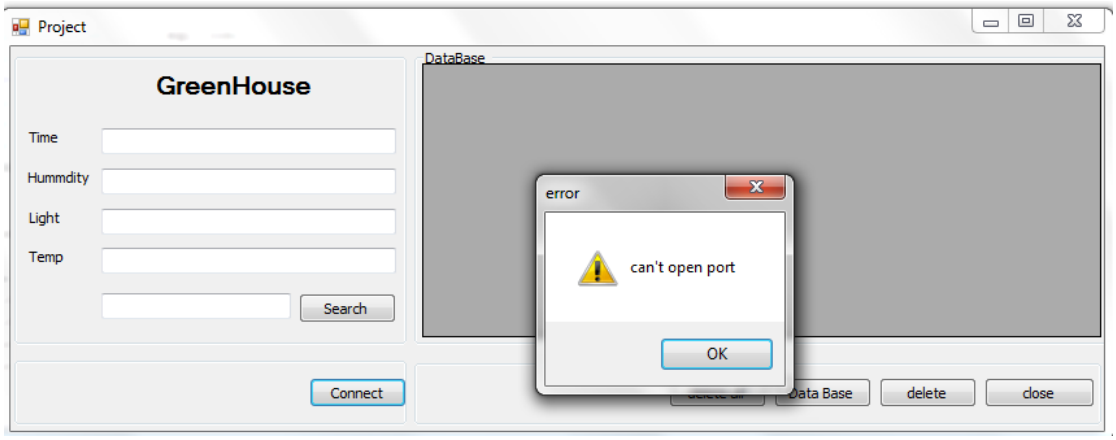

Fig 4.4: connection failed.

To delete such a measurements from database the item delete is used to this as shown below.

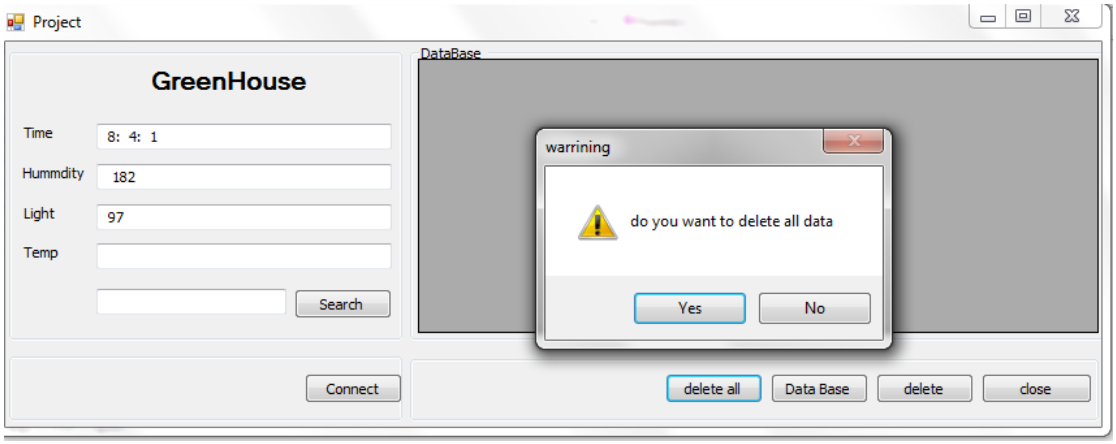

Fig 4.5: assurance massage for delete

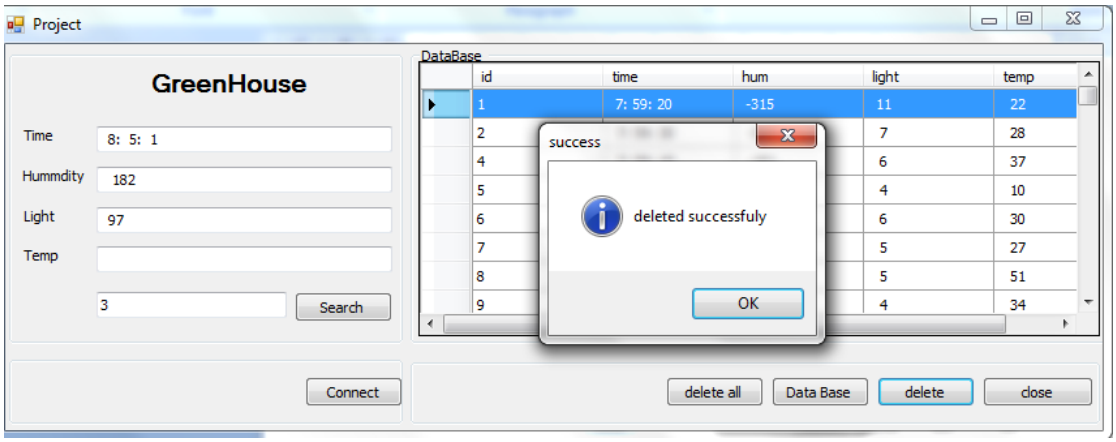

Fig 4.6: delete process.

For search measurements, the ID required from user to search a specific measurements as show in fig, the measurement had ID (3) was searched.

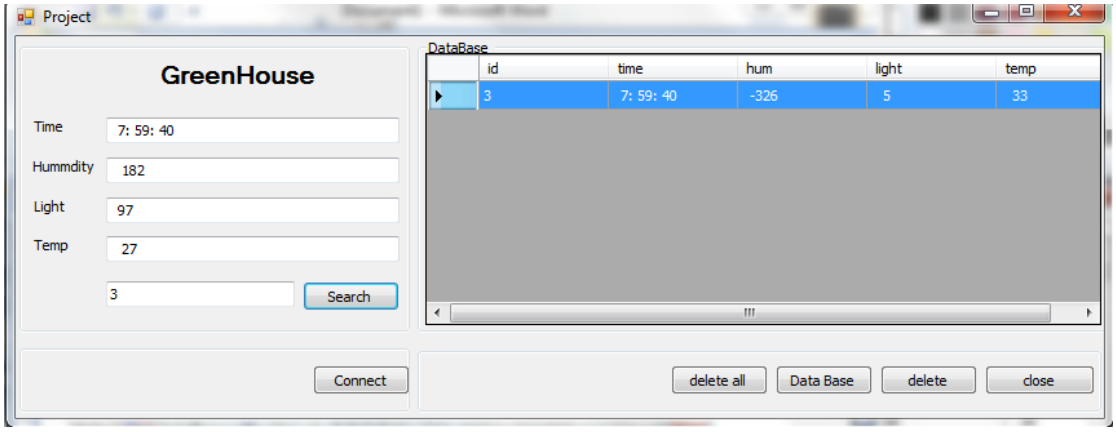

Fig 4.7: searching item.

4.4 Hardware:

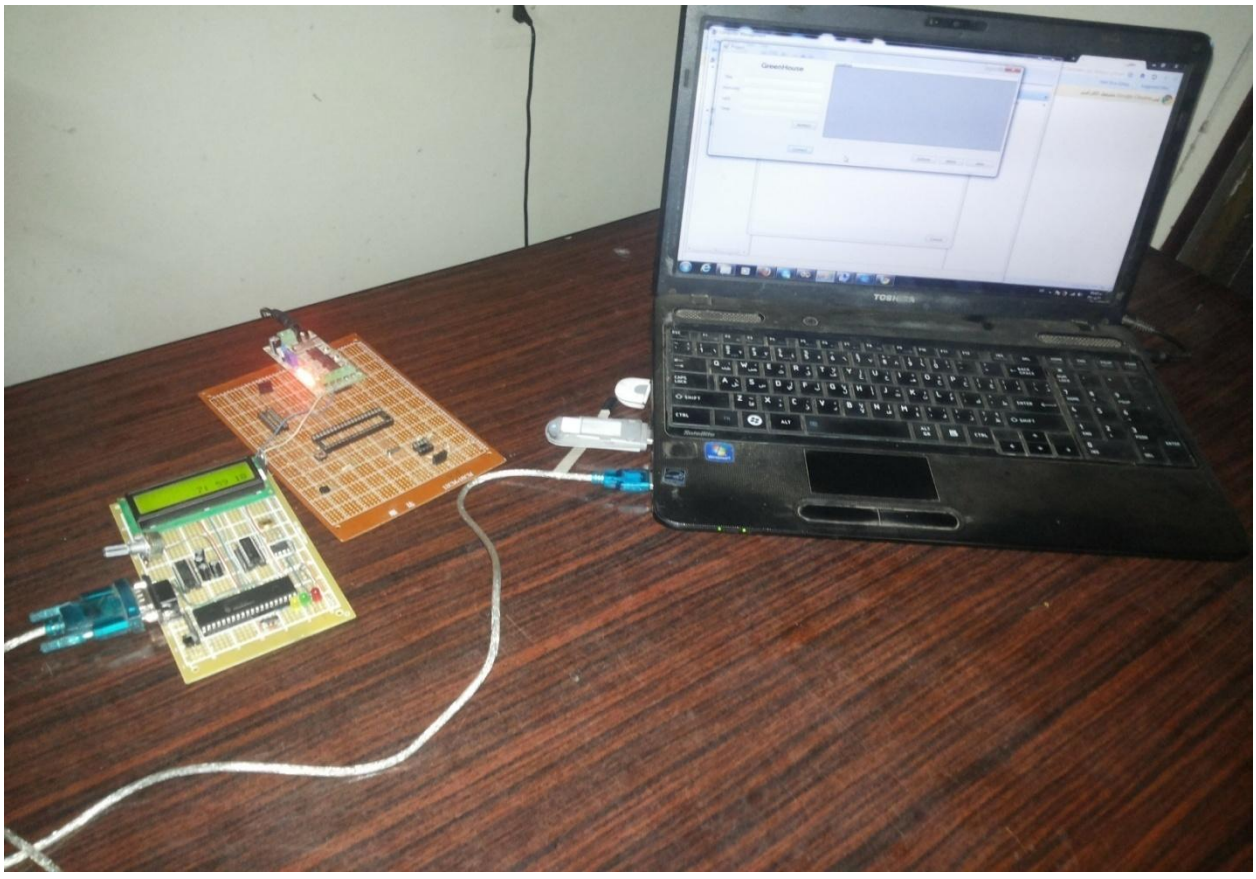

Fig 4.8: Hardware implantation results.

The module built in project acts as a server and greenhouse then displays the sensor data in the form of visual basic GUI in sever side. This module finds its practical implementation in the remote area and provides data through serial link fig (4:8) shows that implementation.

Conclusion & Recommendations

5.1 Conclusion

In this project a smart monitoring and control of environmental parameters in a greenhouse was developed using Proteus software to simulate full circuit design and Micro C as compiler.

In server side visual studio was used to develop GUI and SQLITE to develop database. The Proteus simulator was used to develop the simulation design of the circuit and the hardware was implemented according to the simulated model, a microcontroller form PIC family was used as the main the brain for the system and the resulting were very satisfying.

5.2 Recommendations

For further future development of this system, the following recommendation could be considered:

- Use different Greenhouses at different areas and connect them use more advanced connection. For instance this might be achieved using wireless media.
- A data logger unit can be added for more reliability
- Addition of event logging capabilities
- Adding self checking feature for the circuit may serve quite intelligence to the system and more reliability (possible faults and failure)
- Improve the server side to able the control and monitoring together

References

1- Dogan Ibrahim, 2008, "Advanced PIC Microcontroller Projects in C", Newness publications, USA.

2- Ricky Yuen, 2002," Analog to Digital Converter in Wireless Local Area Network IEEE 802.11a", November 19, 3-4.

3- Dogan Ibrahim, 2006, PIC BASIC Projects, Newnes publications, USA

4- Martin P.Bates, - ," Programming 8 bit PIC Microcontroller in C" Newnes publications, USA.

5- Qichao Zha & Tiejun Lu& Yu Zong & Jianhui Zhang & Shaoxian Qu, 2013, "Design of CMOS Crystal Oscillator with Low Power Consumption", International Journal of Information and Electronics Engineering, Vol. 3, No. 6,630-631.

6- Robert J. Milliken&Jose Silva-Martínez& Senior Member& IEEE& Edgar Sánchez-Sinencio , Fellow, IEEE, 2007," Full On-Chip CMOS Low-Dropout Voltage Regulator", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 54, NO. 9,1.

7- Ankita Tyagi1 & Dr. S. Chatterjee, 2013," Liquid Crystal Display: Environment & Technology**,** "International Journal of Environmental Engineering Science and Technology Research, Vol. 1, No. 7, 2.

8- Julyan Ilettm, 1977," How to use Intelligent L.C.D.s", everyday practical electronic magazine, 4.

9-Chetan Patil, 2011, "Development of a Simple Serial Communication Protocol for Microcontrollers (SSCPM)", International Journal of Scientific and Research Publications, Volume 1, Issue 1, ISSN 2250-3153.

10-S.Thenmozhi1 &M.M.Dhivya2 &R.Sudharsan& K.Nirmalakumari , 2014," Greenhouse Management Using Embedded System and Zigbee Technology",International Journal of Advanced Research in Electrical,
Electronics and Instrumentation Engineering, Vol. 3, Issue 2, 3.

11- Purna Prakash Dondapati& K. Govinda Rajulu, 2012" An Automated Multi Sensored Green House Management, "International Journal of Technological Exploration and Learning (IJTEL) Volume 1 Issue 1m1-3.

12- Joe Mayo, 2010," Microsoft ®Visual Studio® 2010A Beginner's Guide ", New York Chicago San Francisco Lisbon London Madrid Mexico City -Milan New Delhi San Juan-Seoul Singapore Sydney Toronto.

13- Jeetender Singh Chauhan & Sunil Semwal,2013," Microcontroller Based Speed Control of DC Geared Motor Through RS-232 Interface With PC ",.Jeetender Singh Chauhan, Sunil Semwal " ,International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622.

Appendices

Appendix A

C code

```
#line 1 "C:/Users/hamamoo24/Desktop/microc/hwai.c"
sbit LCD_RS at Rd2_bit؛
sbit LCD_EN at Rd3_bit؛
sbit LCD_D4 at Rd4_bit؛
sbit LCD_D5 at Rd5_bit؛
sbit LCD_D6 at Rd6_bit؛
sbit LCD_D7 at Rd7_bit؛
 sbit LCD_RS_Direction at TRISd2_bit؛
 sbit LCD_EN_Direction at TRISd3_bit؛
sbit LCD_D4_Direction at TRISd4_bit؛
sbit LCD_D5_Direction at TRISd5_bit؛
sbit LCD_D6_Direction at TRISd6_bit؛
sbit LCD_D7_Direction at TRISd7_bit؛
  int i; // unsigned int light؛
  char temp[10؛[
 unsigned temps؛
 char lights[10؛[
 unsigned light؛
  void newline()
}
```

```
 UART1_Write(13ؚ(
 UART1_Write(10ؚ(
 delay_ms(100ؚ(
{
float Humidity; char Humiditys[10ؚ[
 char MM,DD,HHؚ
char MMS[4],DDS[4],HHS[4ؚ[
int T1ؚ
void main} ()
trisd.f0=0ؚ
portd.f0=0ؚ
trisd.f1=0ؚ
portd.f1=0ؚ
trisc.f5=0ؚ
portc.f5=0ؚ
   trisc.f3=0ؚ
   trisc.f4=0ؚ
   portc.f3=0ؚ
   portc.f4=0ؚ
   trisc.f1=0ؚ
Lcd_Initؚ()
    UART1_Init(9600ؚ(
   ANSEL=0xFF;              // Configure AN2 pin as analog
  ANSELH=0X00ؚ
```

```
 T1CON = 0b00000011؛
 adc_init؛()
Lcd_Cmd(_lcd_cursor_off؛(
 Lcd_out(1,4,"DESIGNED BY؛("
 Delay_ms(1000؛(
 Lcd_out(2,4,"ELSHIKH؛("
 Delay_ms(1000؛(
 for(i=0; i<15; i} (++
 Lcd_Cmd(_LCD_SHIFT_RIGHT؛(
 Delay_ms(100؛(
{
 MM=59;DD=8;HH=7؛
 for(؛؛)
}
 portc.f1=1؛
 DD؛++
if(DD==60){DD=1;MM{؛++
if(MM==60){MM=0;HH{؛++
if(HH==24) {HH=1{؛
ByteToStr(DD,DDS؛(
ByteToStr(MM,MMS؛(
ByteToStr(HH,HHS؛(
Lcd_Out(2,7,HHS؛(
 Lcd_Out_Cp؛(":")
```

```
 Lcd_Out(2,11,MMS؛(
 Lcd_Out_Cp؛(":")
 Lcd_Out(2,14,DDS؛(
 delay_ms(1000؛(
 lcd_cmd(_lcd_clear؛(
 if(DD==10|| DD==20||DD==30||DD==40|| DD==50|| DD==1(
}
 Temps = ADC_Read(0؛(
  Temps=Temps*0.488؛
  wordToStr(Temps,Temp؛(
    lcd_out(1,1,"T؛("=
  lcd_out(1,3,Temp؛(
  UART1_Write_Text(Temp؛(
UART1_Write_Text(" T؛("
newline؛()
// delay_ms(1000؛(
*/TMR1L=0؛
  TMR1H=0؛
T1= TMR1H؛
  T1=T1<<8؛
  T1=T1|TMR1L;  // OR OPERATION
  Humidity=311.91301651-0.0775253895*T1/*؛
  TMR1L=0؛
  TMR1H=0؛
```

```
 delay_ms(1000꞉(
 T1= TMR1H꞉
 T1=T1<<8꞉
 T1=T1|TMR1L;  // OR OPERATION
 Humidity=569.91301651-0.0775253895*T1꞉
intToStr(Humidity, Humiditys꞉(
 Lcd_Out(1,9,"H꞉("=
Lcd_Out(1,11,Humiditys꞉(
UART1_Write_Text(Humiditys꞉(
UART1_Write_Text(" H꞉("
 newline꞉()
 light = ADC_Read(2꞉(

  wordToStr(light,lights꞉(
  UART1_Write_Text(lights꞉(
   Lcd_Out(2,1,"L꞉("=
Lcd_Out(2,3,lights꞉(
UART1_write_text("l꞉("
    if(light<50}(
portd.f1=1꞉
portd.f0=1꞉
portc.f5=1꞉
{
 if(light>50&&light<=100}(
```

```
portd.f1=1꞉
portd.f0=1꞉
portc.f5=0꞉
{
if(light>100}(
portc.f5=1꞉
portd.f0=0;portd.f1=0꞉
{
newline꞉()
   UART1_Write_Text(HHS꞉(
UART1_Write_Text꞉(":")
    UART1_Write_Text(MMS꞉(
UART1_Write_Text꞉(":")
 UART1_Write_Text(DDS꞉(
 UART1_Write_Text("D꞉("
   newline();   newline꞉()
  DELAY_MS(5000꞉(
  if(Temps>25(
}
    portc.f3=1꞉
{
   if(Temps<25(
}
    portc.f3=0꞉
```

```
{
*/  if(Humiditys > 125(
}
    portc.f4=1;
/*{
*/   if(Humiditys <75(
}
    portc.f4=0;
/*{
  if(Humidity > 125(
}
   portc.f4=1;
{
  if(Humidity <75(
}
  portc.f4=0;
{
  UART1_Write_Text;("$")
{
{
{
```

Appendix B

SQLITE &C# CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
namespace test
{
   public partial class Form1 : Form
   {
      string RXDATA;
      DAl.DataAccessLayer dal = new test.DAl.DataAccessLayer();
      public Form1()
      {
         InitializeComponent();
         serialPort1.BaudRate = 9600;
         serialPort1.PortName = "COM1";
      }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void groupBox1_Enter(object sender, EventArgs e)
        {
        }
        private void button2_Click(object sender, EventArgs e)
        {
            //dal.TestConnecion();
            //dal.ExecuteQuery("create table first (id integer primary key,time
varchar(50),hum varchar(50)"
                //+ ",light varchar(50),temp varchar(50))");
          //dal.ExecuteQuery("insert into first(time ,hum ,light ,temp ) values('" + 10 + "','" +
50 + "','" + 70 + "','" + 80 + "')");
           DataTable dt = dal.LoadData("select *from first");
           dataGridView1.DataSource = dt;
        // MessageBox.Show("Success");
        }
        private void button3_Click(object sender, EventArgs e)
        {
              try
            {
            if (!serialPort1.IsOpen)
            {
```

```
            serialPort1.Open();

            MessageBox.Show("port opened successfully", "success",
MessageBoxButtons.OK, MessageBoxIcon.Information);



          }

        }

         catch

        {

            MessageBox.Show("can't open port", "error", MessageBoxButtons.OK,
MessageBoxIcon.Warning);

         }

        serialPort1.DataReceived += new
SerialDataReceivedEventHandler(receiveData);//==while(true)

        }

        private void receiveData(object sender, SerialDataReceivedEventArgs e)

        {

           RXDATA = serialPort1.ReadLine();

           this.Invoke(new EventHandler(display));

        }

        private void display(object sender, EventArgs e)

        {

           if (RXDATA.Contains("T") && !RXDATA.Contains("l") &&
!RXDATA.Contains("H") && !RXDATA.Contains("D"))

           {

              txtTemp.Clear();

              txtTemp.AppendText(RXDATA.Remove(6));
```

```
            //dal.ExecuteQuery("insert into first(time ,hum ,light ,temp ) values('"
+txtTemp.AppendText(RXDATA.Remove(6))+"','"+
txtHum.AppendText(RXDATA.Remove(6)) + "','" +
txtPress.AppendText(RXDATA.Remove(6)) + "',"

            //+ "'" + txtTime.AppendText(RXDATA.Remove(11)) + "')");

        }

        if (!RXDATA.Contains("T") && !RXDATA.Contains("l") &&
RXDATA.Contains("H") && !RXDATA.Contains("D"))

        {

            txtHum.Clear();

            txtHum.AppendText(RXDATA.Remove(6));

        }

        if (!RXDATA.Contains("T") && RXDATA.Contains("l") &&
!RXDATA.Contains("H") && !RXDATA.Contains("D"))

        {

            txtPress.Clear();

            txtPress.AppendText(RXDATA.Remove(6));

        }

        if (!RXDATA.Contains("T") && !RXDATA.Contains("l") &&
!RXDATA.Contains("H") && RXDATA.Contains("D"))

        {

            txtTime.Clear();



            txtTime.AppendText(RXDATA.Remove(11));

        }

    }

    private void textBox4_TextChanged(object sender, EventArgs e)
```

```
        {
        }
        private void groupBox2_Enter(object sender, EventArgs e)
        {
        }
        private void button5_Click(object sender, EventArgs e)
        {
            DataTable dt = dal.LoadData("select *from first where id='"+1+"'");
            dataGridView1.DataSource = dt;
        }
        private void txtPress_TextChanged(object sender, EventArgs e)
        {
        }
    }
}
```