4. Software developed for the proposed system

Different software modules work simultaneously to achieve the defined goal of the proposed system. These programs are divided into four modules. Module 1 is burnt into the microcontroller of the monitoring board. This program is developed in Keil-C and responsible for displaying sense values of temperature and humidity [1]. It also makes the values available in RS232 port which itself is connected with the one COM port of the computer. Module 2 is burnt in to the microcontroller of the controlling board. This program is also developed in Keil-C and responsible for carrying out necessary control action by starting some hardware devices through relays. Module 3 is stored in computer, COM ports of which is connected with monitoring and controlling boards. Program written in Visual basic will read the values of COM port connected with monitoring board and display the same in interactive front end of the computer. MySQL database server is installed in the computer which stores the sense values in Temperature and Humidity tables of database with corresponding Date and Time. Another table MAXVALUES is constructed with corresponding limits of Temperature and Humidity at which control action has to be initiated. Program continuously monitor sense values with set values stored in MAXVALUES table. If it finds the sense values exceeding set value it passes signal through COM port of controlling board to start appropriate hardware device through relay to bring the parameter within the limit set. Update button provided in the interactive front end will enable users to change the limit value at desired level. Clients of the LAN will be provided with the same front end with all the features similar to server. Clients can monitor continuously sense values at its front end as per constant request for same to server at a regular interval. Client is capable of doing it as it is configured as fat client. Module 4 of the system is responsible for implementation of the system over Internet. It contains software for providing web services. It consist of IIS web server with .NET framework and web application, database server and a browser as a client. The web application is implemented in C#.NET. It sends requests to the database server to send updated data for sending the same to client. Client may send HTTP request to Web server for getting updated values of temperature and Humidity or control data for change of new set values of temperature and humidity. Next Web server send request to Embedded server for retrieving updated values from database or setting new values of temperature and
humidity for initiating control. Embedded server send updated values or new set values to web server. Web server in the form of HTTP response send the same to client. The advantage provided by such a web service is that the user can view and change the necessary values from any location. Above modules can combinedly represent in the following functional diagram of the overall system given in figure 4.1.

![Functional diagram of Embedded server process](image)

**Figure 4.1 : Functional diagram of Embedded server process**

Designed system is implemented in such a way, its users from both Internet and Intranet can access it simultaneously. Interactive front-end for both users designed separately [2]. Figure 4.2 with user and system interface can depict the simultaneous working of the system from Intranet and Internet.
4.1 Software processes for monitoring and controlling boards

In the proposed system, standalone unit will consist of connecting computer along with monitoring and controlling boards. Implementing the system as standalone will limit its functioning from only its place of installation. Still users will avail the user-friendly computer front end which can be operated from keyboard instead of normal liquid crystal display with push button keys. Software process of standalone system is explained in three units [3, 4]. Software process used with monitoring and controlling boards explained in the following section with reference to algorithmic and implementation detail. First unit being the software which has burnt in to the ROM of the monitoring board. Second unit contains the software which has burnt into the ROM of the controlling board.
4.1.1 Algorithm: Monitoring board software

Initialize microcontroller registers P0, P1, P2, P3, TL1, TH1, TH0, IE, SCON and baud rate fixed at 9600 baud.
ADC is used for converting two different sensor values belonging to ZONE1, and ZONE2 which stands for Humidity and Temperature respectively.

Procedure initializing the monitoring system

1. Initialize LCD port P0, P1, P2, P3.

2. IE register is set for masking and unmasking the interrupts. All interrupts will be acknowledged by unmasking the EA (IE.7) bit. The ES(IE.4) bit is unmasked for serial port interrupt and the ETO(IE.1) bit is enabled for Timer 0 overflow interrupt.

3. SCON register is set for start bit, stop bit, and data bits of data framing. Mask the SM0(SCON.7) bit and unmask the SM1(SCON.6) bit for serial mode 1, 8-bit data, 1 stop bit, 1 start bit to make data framing compatible to COM port of the computer.

4. TMOD register is set. C/T bit is zero to indicate that the timer is used as a delay generator. Mode is set to 2 for Timer 1 which means 8-bit auto reload timer and mode 2 for serial com port. Set the TH1 for baud rate. Make the Timer 0 to operate in mode 1 for delay. TH0 and TL0 are loaded some initial values and are cascaded.

Initializing the LCD display

1. Specify function set:

Send 38H for 8-bit, double line and 5x7 dot character format.

2. Display On-Off control:

Send 0CH for display on and cursor off.

3. Entry mode set:
Send 80H for forcing the cursor to beginning of first line

4. Clear display:
Send 01H to clear display and return cursor to home position.

Sending data to LCD for Display

1. Make R/W low

2. Make RS=0 ; if data byte is command

    RS=1 ; if data byte is data (ASCII value)

3. Place data byte on data register

4. Pulse E (HIGH to LOW)

5. Repeat the steps to send another data byte

Reading Data from ADC Chip

Different channels are used to read appropriate sensor data from ADC. The channel A,B are assigned to different pins for sending data to by this procedure for humidity and temperature respectively.

Pin 24 is attached to latch address, pin 13 for output enable to read data on ADC chip, pin 12 to monitor whether conversion (analog-to-digital) is complete. And pin 14 marks the start of conversion.

1. Select an analogue channel by providing bits to A and B addresses as in table below:

<table>
<thead>
<tr>
<th>Analog channel</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IN1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: Different analog channels

2. Activate ALE (Address latch Enable) pin. It needs L-to-H pulse to latch in the address.

3. Activate SC (Start conversion) by L-to-H pulse to initiate conversion.

4. Monitor EOC (End of Conversion) to see whether conversion is finished. H to L output indicates that the data converted is ready to be picked up.
5. Activate OE (output enable) to read data out of ADC chip. An L-to-H pulse to the OE pin will bring digital data out of the chip.

Main Procedure of the Monitoring board to display and transfer data serially

1. Call the Procedure initializing the monitoring system
2. Call the LCD initialization procedure
3. Initialize parameters for ZONE1
4. Read data for ZONE1. The character byte is serially written to SBUF register and store in a variable
5. The TI flag is monitored continuously by the timer interrupt, TF0, to see if the character has been transferred completely.
6. Call the procedure sending data to LCD for display
7. Repeat step 4 to 6 for a fixed period (millisecond) until user-defined external interrupt, INT1, forces to switch to ZONE2
8. Initialize parameters for ZONE2
9. Read data for ZONE2. The character byte is serially written to SBUF register and store in a variable
10. The TI flag is monitored continuously by the timer interrupt, TF0, to see if the character has been transferred completely.
11. Call the procedure sending data to LCD for display
12. Repeat step 9 to 11 for a fixed period (millisecond) until user-defined external interrupt, INT1, forces to switch to ZONE1
13. The TI flag is monitored continuously by the timer interrupt, TF0, to see if the character has been transferred completely.
14. Call the procedure to send data for display
15. Repeat step 13 to 15 for a fixed period (millisecond) until user-defined external interrupt, INT1, forces to switch to ZONE1
16. Repeat step 3 to 15 infinite number of times

procedure for Interrupt service routine to transfer data serially

1. TI is initially set to 0.
2. RI is initially set to 0.
3. The TI flag is raised at the beginning of stop bit to indicate that 8051 is ready to transfer another byte. If raised, make TI = 0;

4. RI flag is raised halfway through the stop bit and indicates that a byte has been received and places the byte in the SBUF register. If raised, make RI = 0;

**Procedure for Interrupt Service Routine for Delay in a Zone**

To generate time delay using timer 0 mode 2, so that each ZONE remains for a fixed number of delays, the following steps are taken:

1. Load the TMOD value register indicating which timer is used. This is done during system initialization procedure.
2. Load TH and TL registers with initial count value
3. Start the timer.
4. Keep monitoring the timer flag (TF0) to see whether it is raised. This is done using external interrupt (INT1)
5. Clear the TF flag.
6. Go Back to step 3 since mode 2 is auto-reload. Each reload should decrement a fixed unsigned value until it reaches 0. When it reaches zero, the 8051 should switch to different zone.

**4.1.2 Implementation of monitoring board algorithm**

The following section of code help each zone to remain for a fixed number of milliseconds before switching to another zone. It directs the interrupt generated by microcontroller to be sent to appropriate ISR (Interrupt Service Routine), such as timer interrupt and serial interrupt routines are executed whenever timer 0 times out, a byte is transferred to com port. Baud rate is fixed at 9600 bps. It initialize Microcontroller registers P0, P1, P2, P3, TL1, TH1, TH0, IE, SCON etc. It calls appropriate routines to convert the binary values generated by ADC to ASCII values and display. At the same time value is sent to COM port attached. It introduce delay for calibration of sensor values keeping track with fast microcontroller for giving average performance. Some part of the modules explained

The LCD routine use to display ASCII values is included as

```c
#include"led1.c"
```

Different constants used in the implementation

```c
#define reload_millisecond 80
```
#define reload_one_forth 20
#define adc_read_port P1 //P1 is ADC data pin to 8051

The common delay routine used is

```
void delay(unsigned int i)
{
    while(i--);
}
```

**Implementation to initialize the monitoring system**

The following routine initializes the system

```
void system_init()
{
    Initialize LCD port P0 and ADC port P1. Also initialize P2, P3.
    P0 = 0xFF; // initializing the LCD ports
    P1 = 0xFF; // make P1 input from ADC
    P2 = 0xFF;
    P3 = 0xFF;
    IE=0X92; // enable serial port interrupt and timer0 interrupt
    SCON=0x50;

    TMOD=0x21; // 21 FOR timer 1 autoreload
    // mode2 for serial com port and timer 0 in mode 1 (16 bit) for delay
    TH1=0xFD; // setting baud rate (9600 baud)
    TH0= 0x3C; // Load TH0
    TL0 = 0xB0; // Load TL0
    TR0=0; // stop timer 0
    TR1=0; // stop timer 1
}
```
Implementation of Reading Data from ADC chip

Following code sends the channel values to the ADC chip which converts the analog data to digital data and send it to the port P1. The various pins associated to channel A, B and C are below:

Pins are assigned to sbit variables

sbit select_channel_a = P2^0;//pin21
sbit select_channel_b =P2^1;//pin22
sbit adc_ale =P2^3; //pin 24 attached to latch address
sbit adc_oe = P3^3; //pin13 output enable to read data on ADC chip
sbit adc_eoc =P3^2; //pin 12 monitor whether conversion is complete
sbit adc_start = P3^4 ; //pin14 , start conversion

Code snippet below read the channel data to port after setting ALE,OE,EOC ,etc. The variable ADC_READ_PORT is assigned to read ADC data from port.

```
adc_read_port = OxFF; // initial assignment
adc_oe = 0; // making it low ,not reading data
adc_eoc =1; //signal conversion not complete
adc_start = 0; //conversion not initiated
adc_ale =0; //clear ALE , clear the address latch
adc_ale =1;
if(channel==0)
{
    // select analog channel
    select_channel_a = 0;
    select_channel_b =0;
    select_channel_c =0;
    delay(100);
}
```

Similarly other channels

Convert and then send data to display and for com port

```
adc_ale=1; // activate ALE , L to H
delay(100);
adc_start=1; // L to H pulse
delay(100);
```
adc_ale=0;          // deactivate ALE
delay(100);
adc_start=0;        //start conversion
delay(100);
while(!adc_eoc);    //stop while conversion not finished
delay(10);
adc_oe=1;           //make it high ,read digital data to port, enable RD
delay(50);
adc_value=adc_read_port;
adc_oe=0 ;          //stop reading, disable RD for next read
delay(3000);
return(adc_value);

Implementation for LCD routines

Following code is the code of “lcd1.c”. It is responsible for displaying values on
LCD screen. The LCD port is attached to port P0.

#define lcd_port P0
Bit Variable assigned to pins
sbit rs=P3^5;           //25 pin
sbit en=P3^6;           //pin 26

The following code initializes the LCD .RS,RW,EN are set appropriately.
void initialize_lcd()
{
    rs=0;
    rw=0;
    en=0;
    en=1;
    lcd_port=0X38;
    ......
    rs=0;
The following is the code to clear the LCD Screen.

```c
void lcd_clr()
{
    lcd_port=0X0E;  //forces cursor to beginning of first line
    en=0;
    en=1;
    ...........
    lcd_port=0X01;  //clear display
    rs=0;
    en=0;
    en=1;
    ...........
}
```

Write data to LCD
void lcd_putc(unsigned char datal)
{
    lcd_port=datal;
    rs=1;
    en=0;
    en=T;
}
The delay routine is written as given below. It takes an argument.
void delay(unsigned int i)
{
    while(i--);
}

**Implementation of interrupt service routine for delay in a zone**

Following code is an ISR. Executed whenever the TH0 bit flag is set or Timer 0 overflows.
Timer 0 overflow indicate that 1 unit elapsed. Every interrupt of Timer 0 decrements the loop
by one until it reaches zero. Reaching 0 (zero) microcontroller switches to another zone and
initialize necessary parameters for that zone. Delay introduced by it is 55 millisecond. So
each zone remains for \((80 \times 55) = 4400\) ms. This is a interrupt for timer interrupt 0 (TF0) that
uses the external interrupt 1 (INT1).

#define reload_milisecond 80    // Number of overflows for Timer 0
Void timer0 (void) interrupt 1 using 2
{
    TH0= 0x3C;                    //load the reload values in the timer
    TL0 = 0xB0;
    if( --milisecond==0)
    {
        milisecond=reload_milisecond;
        if(++count_zone>=2)
        {
            count_zone=0;
        }
    }
    TR0=1;                        // start timer 0
Implementation of Interrupt Service Routine for Serial Communication

Following code is ISR for serial COM port. The serial interrupt is used for receiving or sending data serially, invoked when a byte is sent or received to/from COM port, thereby setting SCON register. This is interrupt service routine for serial communication (RI + TI) that uses Timer interrupt 1 (TF0)

```c
void serialcom(void) interrupt 4 using 1
{
    unsigned char data_recieve=G;
    if(TI)
    {
        TI=0; // ready to sent another byte
        if(RI)
        {
            RI = 0; //ready to receive another byte
        }
    }
}
```

Implementation of the main procedure of the monitoring board algorithm

Initialize the microcontroller device. The Port 0,1,2,3 are initialized. The IE,SCON,TMOD,TH0,TL0 registers are loaded with values through the routine `system_init()`. The routine returns no value, so we write:

```c
system_init();
```

The LCD is initialize by calling the routine `initialize_lcd()`. LCD is attached to port P0.

```c
initialize_lcd();
```

Initially start from ZONE1. The variable `count_zone` keeps track of the zone we are in.

```c
count_zone=0;
```

Start the timer 0 and timer 1 by writing
Now continue to read data for three zones infinitely.

```c
while(1)
{
    Now it is in ZONE1 and it will remain in ZONE1 for some seconds. Read ZONE1 data from ADC, then display and send the data to LCD and COM port respectively, read_adc(unsigned char) uses an argument 0,1,2 for ZONE1,ZONE2 respectively to read data from three channels. Assigning the value to SBUF mean value written to COM port. The following code snippet does this:

    if(count_zone==0)
    {
        ........
        Clear the LCD first.
        lcd_clr();
        ....................
        sensor1=read_adc(0);//
        SBUF=sensor1;//write to serial COM port
        count=sensor1;//for read adc
    }

    Time interval expired for ZONE1, timer interrupt forced to switch to ZONE2 by incrementing count_zone. Refer to timer0 (void) interrupt service routine. The following code is for ZONE2.

    if(count_zone==1)
    {
        ......................
        lcd_clr();
        ......................
        sensor2=read_adc(1);
        SBUF=sensor2;
        count=sensor2;
    }
} // end of while
```
**Implementation to send data for display in LCD**

The following routine displays the ASCII values on the LCD screen. Following code uses the implementation routines of 4.2.2.3. This routine is written with the main routine.

```c
void display_on_lcd()
{
    lcd-port= 0x8b;
    rs = 0; // instruction command mode register is selected
    m = 0; // for H-to-L pulse
    en = 1; // strobe the enable pin LCD latches in the data present at the data pins. LCD ready for display data.

delay(100)

for(i = 1; i< 4 ; i++)
    switch(display_value[i])
    {
        case 0:
            lcd-putc('0')
            Break

        ............
        ............
        ..........
        lcd-putc('9')
        Break;
    }
    // end of switch
} // end of for loop clear display value buffer
delay(100)

for(l = 1; i< 4 ; i++)
    display-value[i] = 0
}
```
4.1.3 Algorithm: controlling board software

Main procedure of controlling board algorithm

Insert variables used in vb prog. For controlling

1. If received value is 1 then switch on relay for voltage
2. If received value is 2 at com port switch off the relay for voltage
3. If received value is 3 then switch on relay for voltage
4. If received value is 4 at com port switch off the relay for voltage
5. Repeat steps 1 to 5 whenever received interrupt is generated

Procedure for System Initialization of controlling board

1. Initialize LCD port P0. Also initialize P1, P2, P3 to FFH

2. IE register is set for masking and unmasking the interrupts. All interrupts will be acknowledged by unmasking the EA (IE.7) bit. The ES(IE.4) bit is unmasked for serial port interrupt and the ET0(IE.1) bit is enabled for Timer 0 overflow interrupt.

IE=0x92

3. SCON register is set for start bit, stop bit, and data bits of data framing. Mask the SM0(SCON.7) bit and unmask the SM1(SCON.6) bit for serial mode 1, 8-bit data, 1 stop bit, 1 start bit to make data framing compatible to COM port of the computer.

SCON=0x50

4. TMOD register is set. C/T bit is zero to indicate that the timer is used as a delay generator. Mode is set to 2 for Timer 1 which means 8-bit auto reload timer and mode 2 for serial com port. Set the TH1 for baud rate. Make the Timer 0 to operate in mode 1 for delay. TH0 and TL0 are loaded some initial values and are cascaded.

TMOD= 0x21
TH1=0xFD
TH0 = 0x3C
TL0=0xB0

5. Stop timer 0 and start the timer 1
Procedure for Interrupt Service Routine of Serial Communication

The following steps are executed to receive character bytes serially.

1. The TMOD register is loaded with value 21H indicating the use of Timer 1 in mode 2 (8-bit auto reload) to set to baud rate.
2. TH1 is loaded with value FDH to set the baud rate (taking XTAL = 11.0592 MHz)
3. The TH0 and TL0 are loaded with 3CH and B0H to check TF0 flag periodically.
4. The SCON register is loaded with value 50H, indicating serial mode 1, where 8-bit data is framed with start and stop bits and receive enable is turned on.
5. TR1 is set to 1 to start Timer 1.
6. RI is cleared with instruction RI = 0.
7. The RI flag bit is monitored by the Timer interrupt 1 to see if an entire character has been received yet.
8. When RI is raised, SBUF has a byte. Its contents are moved to a save place.
9. The moved contents of SBUF are set to appropriate relay.
10. To receive the next character, go to step 6.

4.1.4 Implementation of controlling board software

Contains the software burnt into the microcontroller of controller board. This program is responsible for running appropriate hardware devices through relay if sense temperature and humidity reaches the set value of temperature and humidity. relay1, relay2 are set to appropriate pins as below

```c
sbit relay1 = P2^4;
sbit relay2 = P2^5;
```

Initially no relay is sent or activated

```c
bit flag_relay1 = 0;
bit flag_relay2 = 0;
```

The variable received stores the value received from com port

```c
unsigned char received = 0;
```

Initially no relay will be sent.

```c
bit flag_on = 0;
```
The common delay routine used is

```c
void delay(unsigned int i)
{
    while(i--);
}
```

**Implementation of main procedure of controlling board algorithm**

The port 0,1,2,3 are initialized. The system_init() is called to initialize the system.

```c
void main()
{
    P0 = 0XFF;
P1 = 0XFF;
P2 = 0XFF;
P3 = 0XFF;

    The port P1 is attached to pull-up resistors to put up which helps to switch ON/OFF the device
    P1=0X00; 
    system_init();

    The following code ........
    while(1)
    {
        if(flag_relay1==1)
            relay1=1;
        else
            relay1=0;
        if(flag_relay2==1)
            relay2=1;
```
else
    relay2=0;

if(flag_on==1)
{
    P1=array1[0];
    P1=array1[1];
    delay(60000);
    flag_on=0;
    P1=0xff;
}
}

Implementation for System Initialization of controlling board

void system_init()
{
    P0 = 0xFF; //initializing ports
    P1 = 0xFF;
    P2 = 0xFF;
    P3 = 0xFF;
    IE=0X92; //enable timer0 and serial interrupt
    // IE=0X82;
    TMOD=0x21; //21 FOR timer 1 auto reload mode for serial comm and timer0 in mode 1
    TH1=0xFd; //setting baud rate
    TH0 = 0x3C; //load the reload values in the timer
    TL0 = 0xB0;

    SCON=0x50;
    TR0=0;
    TR1=1;
Implementation of Interrupt Service Routine for Serial Communication

Following code is ISR for serial COM port. The serial interrupt is used for receiving or sending data serially, invoked when a byte is sent or received to/from COM port, thereby setting SCON register. This is interrupt service routine for serial communication (RI + TI) that uses Timer interrupt 1 (TF0). Whenever TF0 flag is raised, there is an overflow, and RI is checked to see if an entire character is received. The contents of SBUF are saved and sent to the main procedure of controlling board for appropriate relay.

```c
void serialcom(void) interrupt 4 using 1
{
    if(TI)
    {
        TI=0;
    }

    if(RI)
    {
        received=SBUF;
        received= received & OxOf;
        Pl= received;
        if(recieved==1)
        {
            flag_relay1=1;
            array1[0]=received;
        }
        if(recieved==2)
        {
            flag_relay1=0;
        }
```
array[0] = received;
}
if(received==3) {
    flag_relay1=1;
    array1[1]=received;
}

if(received==4) {
    flag_relay1=0;
    array1[1]=received;
}

flag_on=1; //contents are saved, now take action for relay.
RI = 0;
}

4.2 Data Acquisition Software Details for Monitoring Board

Following section explains third module of the standalone implementation of the system where software is installed in the attached computer which can be viewed as an embedded server as well as database server. Contains the software that runs synchronously with monitoring and controlling board. It runs on Embedded server where database server is implemented in MySQL[5]. This program is implemented in Visual Basic 6 and interacts with the database server. It displays continuously the sense temperature and humidity in calibrated form. This software retrieves the sense data of temperature and humidity and stores the same in Temperature and Humidity tables of the database with corresponding date and time. It also stores user entered set value of temperature and humidity in a table based on which control action is initiated. This software enables its user to generate report on recorded temperature and humidity for a particular day as well as generate graph of same based on choice of user in the interactive front end.
Following section gives a brief description of the software with two parts: first part gives the algorithm and in the second part implementation of algorithm is given quoting and explaining some modules of program section.

4.2.1 Algorithm: Embedded Standalone VB Monitoring System

1. Establish database connection and create recordset of Humidity and Temperature to store data to database. Also populate the limit recordset.
2. Open the com port 1 to read data from the monitoring device.
3. Read COM port 1 and convert the acquired data to ASCII interger value.
4. Display the value in step 3.
5. Store the value in database.
6. Repeat step 3-5 till the display and store the value of the current zone till device remains in that particular zone after some fixed intervals.
7. Check if zone changes, then change the display area and repeat step 3-6.

4.2.2 Implementation of VB Standalone Monitoring System

The various recordsets used are:

a) pcnAP is a connection parameter recordset
b) prsHumidity is a humidity recordset
c) prsTemperature is a Temperature recordset
d) rsPrev is Recordset to hold previous scale to draw graph
e) prjdialogtitle is a constant that holds the string "INTERACTIVE FRONT-END FROM INTRANET"

Error messages are displayed with the following code:

```vbnet
MsgBox "Error code: " & Err.Number & vbCrLf & "Error description: " & _
     Err.Description & vbCrLf & "Error source: " & Err.Source, _
     vbOKOnly + vbCritical, prjdialogtitle
```

The ODBC Data Source Administrator within Windows enables you to create DSNs, check driver installation and configure ODBC systems such as tracing (used for debugging) and connection pooling. So a DSN is created named “mms”, and connection is setup to the MySQL database using the following code:
strconnect = "DSN=mms"
pcnAP.Open strconnect

Recordset are populated after establishing the connection to MySql database.
For example, the Humidity and Temperature recordsets are populated in the following way:

prsHumidity.Open "SELECT * FROM humidity", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
prsTemperature.Open "SELECT * FROM temperature", pcnAP, adOpenStatic, adLockOptimistic, adCmdText

The com port 1 was opened for reading data from the monitoring board:

    MSComml.PortOpen = True

The timer code executes after a fixed interval, inserts data into database whenever a new data arrives.

The timer clock is disabled with the following code. This is necessary for proper completion of instructions within the procedure. The Timer is disabled by the following code:

    Timer1.Enabled = False

The variable j stores the com input data and converts into character string format. After conversion the converted data is displayed.
    j = MSComml.Input
    j = Asc(j)
    Text6.Text = CStr(j)

Zone switching is checked out with following code snippet. It identifies which device is sending data.
1,2 indicates Humidity and Temperature respectively. When a particular zone switching is detected, the software synchronizes with the monitoring board for that zone data only.
If Text6.Text = "1" Then
    MSComml. Output = "1"
    count = 1
    Text6.Text = " "
Elself Text6.Text = "2" Then
    MSComml. Output = "2"
    count = 2
    Text6.Text = " "
End If

The following piece of code insert data into Temperature table if it is in Temperature zone.

First, create a new record in the recordset
    prsTemperature.AddNew
Load the Temperature values to the recordset
    Load Temperature
Finally write the Temperature values to the database
    prsTemperature.Update

To display, the values of Temperature in the Multiline text box, the vb code is:
Assume Text6 hold current data which is appended to Text1 to display:
    Text1.Text = Text6 + Text1.Text

Similar code fragments for Humidity.

4.2.3 **Algorithm: XY-Data for Standalone VB Monitoring System**

x, y = data to be plotted
x value is generally the time
y values can be the values of Temperature, Humidity.
    xmin, ymin, xmax, ymax = minimum and maximum values of x and y
n = number of data points
This procedure takes information on minimum, maximum values, major divisions (ticks) and minor divisions (sub-ticks) in the x and y axes of plot.
1. Connect to database and select any date of a particular recordset.
2. Populate the recordset for that date.
3. Select minimum and maximum values of x and y for that date.
4. Calculate the values of x and y as
   
   \[
   \text{ticks} = \frac{(\text{Difference of min. and max values})}{10} \\
   \text{subticks} = \frac{(\text{Difference of min and max values})}{100}
   \]
5. Save the minimum, maximum, ticks and subticks of x and y values in a file temp.txt.
6. Count the number of records, n, and write it to file, temp.txt
7. Take out all x and y values from the recordset and write it to file, temp.txt

4.2.4 Implementation of XY-Data for StandAlone VB Monitoring System

Code to connect to database is below. Here strconnect is a string variable.

```
strconnect = "DSN=mms"
pcnAP.Open strconnect
```

Select any recordset, say temperature, and group records by date.

```
Set rs = New Recordset
strSQL = "select TDate from temperature group by TDate"
rs.Open strSQL, cn, adOpenStatic, adLockOptimistic, adCmdText
```

Select the date and close the recordset.

```
Select minimum and maximum values of x and y for that date for temperature.
strSQL = "select min(TGTime) as MinX, max(TGTime) as MaxX, min(Temperature) as MinY," & _
        "max(Temperature) as MaxY from temperature where TDate = " & lstDate.Text & ""
rs.Open strSQL, cn, adOpenStatic, adLockOptimistic, adCmdText
```

Calculate the values of x and y as

\[
\text{ticks} = \frac{(\text{Difference of min. and max values})}{10} \\
\text{subticks} = \frac{(\text{Difference of min and max values})}{100}
\]

The code is:

```
With rs
    txtXmin = Format(!MinX, "###.###########")
    txtXmax = Format(!MaxX, "###.###########")
```

txtYmin = Format(!MinY, "###.############")
txtYmax = Format(!MaxY, "###.############")
txtXticks = Format((!MaxX - !MinX) / 10, "###.############")
txtYticks = Format((!MaxY - !MinY) / 10, "###.############")
ttxtXsubticks = Format((!MaxX - !MinX) / 100, "###.############")
ttxtYsubticks = Format((!MaxY - !MinY) / 100, "###.############")
.Close
End With

Save the minimum, maximum, ticks and subticks of x and y values in a file temp.txt

Ticks and sub-ticks of x are xtick and xstick.
Ticks and sub-ticks of y are ytick and ystick.
Min. and max. of subticks of x are xsmin and xsmax.
Min. and max. of subticks of y are ysmin and ysmx
xsmin = Val(txtXmin): xsmax = Val(txtXmax)
xtick = Val(txtXticks): xstick = Val(txtXsubticks)
ysmin = Val(txtYmin): ysmx = Val(txtYmax)
ytick = Val(txtYticks): ystick = Val(txtYsubticks)

Save the above data to file ‘temp.txt’
Open "C:\temp.txt" For Output As #1
Print #1, xsmin, xsmax, xtick, xstick
Print #1, ysmin, ysmx, ytick, ystick

Count the number of records in the recordset and write to C:\temp.txt
rs.MoveFirst
Do Until rs.EOF
k = k + 1
rs.MoveNext
Loop
n = k
Print #1, n

Take out all the x and y values from the recordset and write it to file "C:\temp.txt"
k = 1
With rs
  .MoveFirst
  Do Until .EOF
    x(k) = Val(Ixdata)
    y(k) = Val(lydata)
    Print #1, x(k), y(k)
    k = k + 1
  .MoveNext
Loop
.Close
end With

The data generated is used to plot xy-graph. The algorithm and implementation is given in section 1.1.5 and section 1.1.6.

4.2.5 Algorithm: XY-Plot for Standalone VB Monitoring System

The following algorithm produces the XY-plot after executing XY-Data algorithm.

1. Open the file temp.txt and read the min., max., ticks, subticks and n from file temp.txt and store the values to some temporary variable.
2. Read x and y values and save it to two one-dimensional array.
3. Repeat step 2, n steps
4. Determine the axes ranges
5. Determine a parameter r such that if R is minimum value of the difference of x subticks or y subticks, then
   \[ r = \frac{R}{200} \]
6. Set scale for plot, draw axes, ticks and sub-ticks.
7. Print the ticks values near the tick marks
8. Draw circles representing the data points and draw lines joining circles.
4.2.6 Implementation of XY-Plot for Standalone VB Monitoring System

Read the data from file "c:\templ.txt" passed from above XY-Data algorithm

Open "C:\temp.txt" For Input As #1
Input #1, xsmin, xsmax, xtick, xtick
Input #1, ysmin, ysmax, ytick, ytick
Input #1, n

Read the x,y data from the file temp.txt and store it to array x and y
for k = 1 To n
    Input #1, x(k), y(k)
Next k

Determine the axes ranges and take the absolute value of ranges. The following code does this:

    if xsmin <= 0 Then
        xaxis = 0#
    else
        xaxis = xsmin
    end if

    if ysmin <= 0 Then
        yaxis = 0#
    else
        yaxis = ysmin
    end if

Consider a parameter r such that if R is minimum value of the difference of x subticks or y subticks, then

    r = R / 200

The code snippet is self-explanatory.

    xrange = xsmax - xaxis: Yrange = ysmax - yaxis
    rx = Abs(xrange) / 200: ry = Abs(Yrange) / 200
    if Abs(xrange) < Abs(Yrange) then
\[ r = rx \]
\[ \text{else} \]
\[ r = ry \]
\[ \text{end if} \]

Set scale for plot, draw axes, ticks and sub-ticks. The following code prints the ticks values near the tick marks. \( nx \) is the number of \( x \) points along \( x \)-axis. Similarly for \( ny \). \( xt(k) \) and \( yt(k) \) are the saved ticks values along \( x \)-axis and \( y \)-axis respectively. These are one dimensional arrays. \( \text{picPlot} \) is the object where the plot is drawn. The code snippet is:

\[
\text{picPlot.Scale (xsmin - xtick, ysmax + ytick)-(xsmmax + xtick, ysmmin - ytick)}
\]
\[
\text{picPlot.Line (xaxis, ysmmin)-(xaxis, ysmmax)}
\]
\[
\text{picPlot.Line (xsmin, yaxis)-(xsmmax, yaxis)}
\]
\[
\text{picPlot.Line (xsmmax, ysmmin)-(xsmmax, ysmmax)}
\]

\[
\text{nx = xsrange / xtick + 1: ny = Ysrange / ytick + 1}
\]

\[
\text{for k = 1 To nx}
\]
\[
\text{xt(k) = xsmin + (k - 1) * xtick}
\]
\[
\text{picPlot.Line (xt(k), yaxis - ytick / 10)-(xt(k), yaxis + ytick / 10)}
\]
\[
\text{picPlot.Line (xt(k), ysmmin)-(xt(k), ysmmax)}
\]
\[
\text{next k}
\]

\[
\text{for k = 1 To ny}
\]
\[
\text{yt(k) = ysmmin + (k - 1) * ytick}
\]
\[
\text{picPlot.Line (xaxis - xtick / 10, yt(k)-(xaxis + xtick / 10, yt(k))}
\]
\[
\text{picPlot.Line (xsmin, yt(k)-(xsmmax, yt(k))}
\]
\[
\text{next k}
\]

\[
\text{nxx = xtick / xstick}
\]

\[
\text{for k = 1 To nx - 1}
\]
\[
\text{for l = 1 To nxx}
\]
\[
\text{xtt(l) = xt(k) + (l - 1) * xstick}
\]
\[
\text{picPlot.Line (xtt(l), yaxis - ytick / 20)-(xtt(l), yaxis + ytick / 20)}
\]
\[
\text{next l}
\]
\[
\text{next k}
\]
\[ nyy = \frac{ytick}{ystick} \]

for \( k = 1 \) to \( n - 1 \)

for \( l = 1 \) to \( nyy \)

\[ ytt(l) = yt(k) + (1 - 1) \times ystick \]

\[ \text{picPlot.Line (xaxis - xtick / 20, ytt(l))-(xaxis + xtick / 20, ytt(l))} \]

next \( l \)

next \( k \)

Draw circles representing the data points and draw lines joining circles

for \( k = 1 \) to \( n \)

\[ \text{picPlot.Circle (x(k), y(k)), r} \]

if \( k > 1 \) Then \[ \text{picPlot.Line (x(k - 1), y(k - 1))-(x(k), y(k))} \]

next \( k \)

Colour of the plot is given by the following code:

\[ \text{picPlot.ForeColor = RGB(255, 0, 255)} \]

4.3 Data Acquisition Software Details for Controlling Board

One objective of the application is to keep the physical parameters at desired level as stored in the control database. This procedure will perform the necessary operation by sending the appropriate signal to the controlling board. In case of exceeding the sense value than the limit value it will write the control data to the 2nd COM port. It determines type of operation to be performed (on/off) to the appropriate device through relay to keep the parameters at desired level. It also continuously display the status of the each device.

4.3.1 Algorithm: Embedded Standalone VB Controlling System

1. Establish database connection and create recordset for limit data. Also populate the limit recordset.

2. Set the control devices to a saved state and display the status of the controlling devices

3. Update the limits for Humidity and Temperature, if required and display in appropriate text box.

4. Open com port 2 to write control data to controlling device to control the monitoring device parameters

5. Check if the current zone data exceeded the zone limit. If exceeded, send control data to the com port 2 to control the appropriate sensor device
6. Repeat step 2-5 after fixed time intervals.

4.3.2 Implementation of VB Standalone Controlling System

The various recordsets used are

a) pcnAP is a connection parameter recordset
b) rsMaxValues is a recordset to hold maximum values of Humidity and Temperature
c) rsChkDevice is a recordset to hold the devices ON and OFF values
d) rsUpdate is a recordset to update the limits of Humidity and temperature
e) prjdialogtitle is a constant that holds the string "INTERACTIVE FRONT END FROM INTRANET"

Error messages are displayed with the following code:

```vba
MsgBox "Error code: " & Err.Number & vbCrLf & "Error description: " & _
    Err.Description & vbCrLf & "Error source: " & Err.Source, _
    vbOKOnly + vbCritical, prjdialogtitle
```

The limit recordset are populated in the following way:

```vba
rsMaxValues.Open "SELECT * FROM maxvalues", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
```

The com port 2 is opened for writing or sending signals to the controlling board by the following code:

```vba
MSComm2.PortOpen = True
```

The following code load the maximum limits on different text boxes for display:

```vba
With rsMaxValues
    Text12 = !Humidity
    Text13 = !Temperature
end with
```

The last status of the devices are displayed on the screen by checking appropriate check boxes:
With rsChkDevice
  if !Device1 = 0 Then
    Check1.Value = Unchecked
    Check2.Value = Checked
  else
    Check1.Value = Checked
    Check2.Value = Unchecked
  end if
  if !Device2 = 0 Then
    Check3.Value = Unchecked
    Check4.Value = Checked
  else
    Check3.Value = Checked
    Check4.Value = Unchecked
  end if
end With

Check for maximum temperature i.e. it is in temperature zone, check whether the current voltage sensor data exceeded the limit temperature data.

If YES then Switch ON appropriate device otherwise Switch OFF the device.

The procedure command1_Click is used send signals to the controlling board for necessary action on reading the parameters of monitoring board.

  if rsMaxValues! Temperature <= Val(Text6) then
    Label.Caption = "Temperature exceeded maximum range"
    Check1.Value = Checked
    Call Command1_Click
  else
    Check2.Value = Checked
    Call Command1_Click
  end if

The following code switches the controlling devices ON/OFF. This is procedure Command_Click(). If the device attached to a control is switch on then it sends the signal and save the signal for future notification. Device1 and Device2 represents control devices for
Humidity and Temperature respectively. The numbers assigned to MSComm2.Output is given in the table below:

<table>
<thead>
<tr>
<th>Number</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Temperature Control Device Turn ON</td>
</tr>
<tr>
<td>2</td>
<td>Temperature Control Device Turn OFF</td>
</tr>
<tr>
<td>3</td>
<td>Humidity Control Device Turn ON</td>
</tr>
<tr>
<td>4</td>
<td>Humidity control Device Turn OFF</td>
</tr>
</tbody>
</table>

Table 4.2: Controlling parameters for Device on/off

```plaintext
if Check1.Value = Checked And rsChkDevice1.Device1 = 0 then
    MSComm2.Output = "1"
    rsChkDevice1.Device1 = 1
elseif Check2.Value = Checked And rsChkDevice1.Device1 = 1 then
    MSComm2.Output = "2"
    rsChkDevice1.Device1 = 0
end if
```

rsChkDevice.Update

Similar check can be carried out for Humidity.

### 4.4 Implementation of Database

Databases are created to keep track with sense data in proper format for future analysis and retrieval and to store limit value for carrying out necessary control. Sensor data from humidity and Temperature sensor stored in Humidity and Temperature tables of database with respect to time and date. These tables are having its attributes value, Date and time. Controlling value of humidity and temperature is stored in Maxvalue table for storing the current controlling value of temperature and humidity. This table is having its attribute Parameter name and value [6]. This value can be updated by user through front end designed whenever it is required to change. Detail discussion of same is made in section 4.6.
4.4.1 Creation of Database on StandAlone System

The configuration of database on a stand-alone implementation is same as the implementation of database on client-server concept. Refer to section 4.6.1 for details.

The port no is 3306 and is always same whatever implementation is chosen and is default port. It can be change for security reasons. Only the Database Server is assigned as 'localhost' on standalone system. For client-server configuration remote host name or ip address is given where database server is installed.

4.4.2 ODBC- 3.51 with MySQL

Open Database Connectivity (ODBC) refers to a software API method for using database management systems (DBMS). ODBC was created so as to be independent of programming languages or operational systems and offers access to different database systems. The standard ODBC consists of an ODBC core and the respective specific ODBC database drivers. The core, also known as Driver manager, is independent of the database and acts as an interpreter between the application and the database drivers. The database drivers, on the other hand, contain DBMS-specific details and offer a mechanism for connecting with different ODBS-enabled database systems.

This implementation uses MySQL Database server. The ODBC driver for MySQL is labeled MyODBC. It is also known as MySQL Connector /ODBC. It is developed by the MySQL team and at the moment, it has two versions available - 3.51 and 5.1. This application uses ODBC 3.51 DSN for windows.

Detail discussion is made on ODBC configuration in Section 4.6.3
4.5 Client-Server Implementation Details

Client – Server environment is provided in LAN implementation using Visual Basic. Same front-end is design for client as well as server as given in fig. 4.1. As already mentioned Embeddd boards are connected to the VB server which is also referred as embedded server. Necessary server configuration in Visual Basic is done in to the system [7 , 8]. Client configuration is made in other nodes of the system. Here client being a fat client send continuous request to server for updated values. Server in response to same send the updated values regularly. As database server is also installed in the same server used for keeping the record of parameter values in appropriate tables based on which report and graph of same can be displayed on user request.

Following two section gives the implementation detail of the client and server modules of the application.

4.5.1 Implementation: Embedded VB Server

1. Establish database connection and create recordset of Humidity and Temperature to store data to database. Also populate the limit recordset. The code snippet is:

```vba
Set pcnAP = New Connection 'new connections set
Set prsHumidity = New Recordset 'set the new recordset for humidity
Set prsTemperature = New Recordset ' set the new recordset for temperature
Set rsMaxValues = New Recordset 'set the new recordset for Limits
Set rsChkDevice = New Recordset 'set the new recordset for Device status
pcnAP.CursorLocation = adUseClient
establish the connection to MySql database and populate the recordsets
strconnect = "$DSN=mms"
pcnAP.Open strconnect
prsHumidity.Open "SELECT * FROM humidity", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
prsTemperature.Open "SELECT * FROM temperature", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
rsMaxValues.Open "SELECT * FROM maxvalues", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
rsChkDevice.Open "SELECT * FROM chkdevices", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
```
2. Open the com port 1 to read data from the monitoring device and com port 2 to write control data to controlling device to control the monitoring device parameters.

    MSComm1.PortOpen = True
    MSComm2.PortOpen = True

3. Set the control devices to a saved state and display the status of the controlling devices.

    The limit values are displayed by the following code snippet:
    
    With rsMaxValues
        Text12 = rsMaxiHumidity
        Text13 = rsMaxiTemperature
    End With

    The status of devices are shown on the display by the following code:

    With rsChkDevice
        if Device1 = 0 Then
            Check1.Value = Unchecked
            Check2.Value = Checked
        else
            Check1.Value = Checked
            Check2.Value = Unchecked
        end if
        if Device2 = 0 Then
            Check3.Value = Unchecked
            Check4.Value = Checked
        else
            Check3.Value = Checked
            Check4.Value = Unchecked
        end if
    end With
4. Update the limits for Humidity and Temperature, if required and display in appropriate text box. The update code is:

```vba
rsUpdate.Open "SELECT * FROM maxvalues", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
with rsUpdate
  !Humidity = Text2
  !Temperature = Text3
  .Update
end With
```

5. Read COM port 1 and convert the acquired data to ascii integer value.
6. Display the value in step 5.
7. Store the value in database.
8. Check if the current zone data exceeded the zone limit. If exceeded, send control data to the com port 2 to control the appropriate sensor device.
9. Repeat step 5-8 till the display and store the value of the current zone till device remains in that particular zone after some fixed intervals.
10. Check if zone changes, then change the display area and repeat step 5-9.

The code executing the above steps are done by a timer sub-routine:

```vba
Private Sub Timer1_Timer()
  On Error GoTo rc
  Dim a As String
  Static inserttemperaturecounter As Integer
  Static inserthumiditycounter As Integer
  Static count As Integer 'indicates the current zone are in
  Dim j As Variant 'accepts the com port input
  Static prevHumidity As Double 'stores the last Humidity
  Static prevTemperature As Double 'stores the last Temperature
```

Page 82 of 127
'Initialise the device counter if program was first run and data device counter if program was first run
if (Text9 = "-1") then
    Text9 = "0"
    count = -1
end if

rsMaxValues.Open "SELECT * FROM maxvalues", pcnAP, adOpenStatic, adLockOptimistic, adCmdText

Load the max. ranges into the form with rsMaxValues
    Text12 = !Humidity
    Text13 = !Temperature
end With
rsMaxValues.Close

The timer clock is disabled with the following code. This is necessary for proper completion of instructions within the procedure.

Timer1.Enabled = False
Text1.Text = Date
Text2.Text = Time

Stores the com input data and converts into character string format
j = MSComm1.Input
j = Asc(j)
Text6.Text = CStr(j)

Zone switching is checked out with following code snippet. It identifies which device is sending data.
1, 2 and 3 indicates Humidity and Temperature respectively. When a particular zone switching is detected, the software synchronizes with the monitoring board for that zone data only.
if Text6.Text = "1" Then  
    MSComm1.Output = "1"  
    count = 1  
    Text6.Text = " "  
elseIf Text6.Text = "2" Then  
    MSComm1.Output = "2"  
    count = 2  
    Text6.Text = " "  
end If  

'Shows the Temperature in the specified text box, highlight the labels and inserts data into Voltage table  
If (count = 1) Then  
    Label3.Font.Bold = True  
    Label4.Font.Bold = False  
    Label5.Font.Bold = False  
    Text3.Text = Text6 + Text3.Text  
    Text3.Text = " " + Text3.Text  
End If  

Reset the device counter if a Temperature is read in the com port  
if (Text6 <> " ") Then  
    count = -1  
end If  

The following piece of code insert data into Temperature table if current Temperature is not equal to the previous Temperature  
inserttemperaturecounter = inserttemperaturecounter + 1  
if (((Text6 <> CStr(prevVoltage) And Text6 <> " ")) ) Then  
    if (Text6 = " ") Then Text6 = CStr(prevTemperature)  
    if (Text6 <> " ") Then  
        prsTemperature.AddNew  
        Load Temperature  
        prs Temperature.Update
prevTemperature = Text6
end if
insertTemperaturecounter = 0 'reset if Temperature counter reaches a specified
end if

Call function MaxTemperature check for maximum Temperature
Call MaxTemperature
end if

'Show the humidity in the specified text box, highlight the labels and inserts data into
Humidity table
if (count = 2) then 'Show the humidity
    Label3.Font.Bold = False
    Label4.Font.Bold = True
    Label5.Font.Bold = False


reset the device counter if a Humidity is read in the com port
if (Text6 <> " ") then
    count = -1
end if

'the following piece of code insert data into Humidity table if current humidity is not equal
to the previous Humidity
inserthumiditycounter = inserthumiditycounter + 1
if (((Text6 <> CStr(prevHumidity) And Text6 <> " ") Or (inserthumiditycounter = 20))
then
    if (Text6 = " ") Then Text6 = CStr(prevHumidity)
    if (Text6 <> " ") Then
        prsHumidity.AddNew
        LoadHumidity
        prsHumidity.Update
        prevHumidity = Text6
insert humiditycounter = 0 'reset if humidity counter reaches a specified value

end if

Call function MaxHumidity check for maximum voltage
Call MaxHumidity
end if

All instructions finished. Repeat the loop again
    Timer1.Enabled = True
    Exit Sub
end if

rc: if any error occurs cancel all instructions and start the loop again
    Timer1.Enabled = True
end Sub

The procedures MaxTemperature, MaxHumidity check if the current zone data exceeded the zone limit. If exceeded, send control data to the com port 2 to control the appropriate sensor device. The procedure calls Command1_Click described later.

The following code checks if maximum humidity reached or exceeded limit and switches ON/OFF appropriate device. Current value of Humidity is stored in Text6

if rsMaxValues!Humidity <= Val(Text6) then
    Label20.Caption = "Humidity exceeded maximum range"
    Check1.Value = Checked
    Call Command1_Click
else
    Label20.Caption = ""
    Check2.Value = Checked
    Call Command1_Click
end if
The following code checks if maximum temperature reached or exceeded limit and switches ON/OFF appropriate device. Current value of temperature is stored in Text6

if rsMaxValues!Temperature <= Val(Text6) then
    Label20.Caption = "Temperature exceeded maximum range"
    Check3.Value = Checked
    Call Command1_Click
else
    Label20.Caption = ""
    Check4.Value = Checked
    Call Command1_Click
end if

If any sensor value exceeded, send control data to the com port 2 to control the appropriate sensor device. The procedure calls Command1_Click sends the data to the controlling board. The folowing code switches the devices ON/OFF. It sends the signal and saves the signal for future notification. The data written to com port 2 is given earlier.

if Check1.Value = Checked And rsChkDevice!Device1 = 0 then
    MSComm1.Output = "1"
    rsChkDevice!Device1 = 1
elseif Check2.Value = Checked And rsChkDevice!Device1 = 1 then
    MSComm1.Output = "2"
    rsChkDevice!Device1 = 0
end if

if Check3.Value = Checked And rsChkDevice!Device2 = 0 then
    MSComm1.Output = "3"
    rsChkDevice!Device2 = 1
elseif Check4.Value = Checked And rsChkDevice!Device2 = 1 then
    MSComm1.Output = "4"
    rsChkDevice!Device2 = 0
end if

rsChkDevice.Update if any change happened, update the status of controlling devices
4.5.2 Implementation: Embedded VB Client

Establish connection to database and create, but do not populate the recordsets for Voltage, Humidity and Temperature device data and also do not populate the limit recordset. The codes are:

```
Set pcnAP = New Connection 'new connections set
Set prsHumidity = New Recordset 'set the new recordset for humidity
Set prsTemperature = New Recordset 'set the new recordset for temperature
Set rsMaxValues = New Recordset 'set the new recordset for Limits
Set rsChkDevice = New Recordset 'set the new recordset for Device status
```

```
pcnAP.CursorLocation = adUseClient
```

Establish the connection to MySql database and populate the recordsets
```
strconnect = "DSN=mms"
```
```
pcnAP.Open strconnect
```

1. Initialize the limits for Voltage, Humidity and Temperature devices for controlling purposes and display in the appropriate textbox.
2. Set the control devices to last close state and display the state of the devices.

The following code snippet executes the above two steps:

```
The limit values are displayed by the following code snippet:
with rsMaxValues
    Text12 = !Humidity
    Text13 = !Temperature
end with
```
```
The status of devices are shown on the display by the following code:
With rsChkDevice
    if !Device1 = 0 then
        Check1.Value = Unchecked
        Check2.Value = Checked
    else
        Check1.Value = Checked
        Check2.Value = Unchecked
    end If
```
if !Device2 = 0 then
    Check3.Value = Unchecked
    Check4.Value = Checked
else
    Check3.Value = Checked
    Check4.Value = Unchecked
end if

end with

Update the limits for Humidity and Temperature, if required and display in appropriate text box. The update code is:

rsUpdate.Open "SELECT * FROM maxvalues", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
With rsUpdate
    !Humidity = Text2
    !Temperature = Text3
    .Update
End With

3. Populate the Humidity and Temperature recordset for display the values of Humidity and Temperature in appropriate Multiline Text Box.
4. Repeat steps 2-4 after fixed intervals.

The code for these code for executing the above two steps are done in a timer sub-routine:

Private Sub Timer1_Timer()
Dim prevHumidity As Double
Dim prevTemperature As Double
Static inserthumiditycounter As Integer
Static inserttemperaturecounter As Integer
The timer clock is disabled with the following code. This is necessary for proper completion of instructions within the procedure.

Timer1.Enabled = False
prsHumidity.Open "SELECT Humidity FROM humidity", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
prsTemperature.Open "SELECT Temperature FROM temperature", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
rsMaxValues.Open "SELECT * FROM maxvalues", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
rsChkDevice.Open "SELECT * FROM chkdevices", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
rsMaxValues.Close
rsMaxValues.Open "SELECT * FROM maxvalues", pcnAP, adOpenStatic, adLockOptimistic, adCmdText
Load the max. ranges into the form
with rsMaxValues
    Text12 = !Humidity
    Text13 = !Temperature
end with
rsMaxValues.Close
Text1.Text = Date display the current date and time values
Text2.Text = Time

Move to the last record of the Humidity recordset and save the value in Text6. Display the value by appending the value with the values in Text4

prsHumidity.MoveLast
Text6 = CStr(prsHumidity!Humidity)

Call function MaxHumidity check for maximum Humidity. It does not send signals to the com port 2 directly. It compares the last Humidity value with the limit values and display the status of the devices of the controlling board.
    Call MaxHumidity
prsHumidity.Close

Move to the last record of the Temperature recordset and save the value in Text6. Display the value by appending the value with the values in Text5
prsTemperature.MoveLast
Text6 = CStr(prsTemperature!Temperature)
Text5.Text = Text5.Text + " " + Text6

Call function MaxTemperature check for maximum Temperature. It does not send signals to the com port 2 directly. It compares the last Temperature value with the limit values and display the status of the devices of the controlling board.

Call MaxTemperature
prsTemperature.Close

All instructions finished. Repeat the loop again

Timer1.Enabled = True

End Sub

4.6 ODBC/Database Process Details of MySQL

MySQL database is the world's most popular open source database because of its fast performance, high reliability, ease of use, and dramatic cost savings. The most widely used and popular SQL type database in World Wide Web today is MySQL. As the most popular Open Source SQL database management system, MySQL is developed, distributed, and supported by MySQL AB. MySQL databases are preferred over the proprietary database systems due to their reliability and speed of performance. They are most commonly used for both regular and embedded Web applications running on UNIX, Windows and Mac OS. MySQL is the perfect cross platform database software for both averagely loaded websites and high-traffic web portals [9].

There are actually four versions of MySQL:

- **MySQL Standard** includes the standard storage engine, as well as the InnoDB storage engine, which is touted as a "transaction-safe, ACID-compliant database" with some additional features over the standard version.
- **MySQL Pro** is the commercial version.
- **MySQL Max** includes the more technologically advanced features that are available during early access programs.
- **MySQL Classic** is the standard storage engine without the InnoDB engine. This is another commercial version.
As with securing a network, securing a database by looking at the various layers that are involved is an effective approach. Security of databases can be defined as preventing unauthorized or accidental disclosure, alteration, or destruction of data. In addition to assigning firewall in different levels Internet, Application etc. authentication as well as security can be achieved through Access control through read, insert, update and delete commands must also be assigned appropriately within those views. Role-based authentication should be considered when adding access to any database. Typical roles for access include administrator, user, programmer and operator. Integrity and encryption are the other ways to achieve the security. This implementation uses the MySQL standard version 5.0 in default configuration.

### 4.6.1 Database Implementation Details

Table name: Maxvalues

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humidity</td>
<td>Stores the maximum humidity value</td>
</tr>
<tr>
<td>Temperature</td>
<td>Stores the maximum temperature value</td>
</tr>
</tbody>
</table>

Table 4.3: Table for controlling parameter

Table name: Humidity

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humidity</td>
<td>Stores the sense humidity value</td>
</tr>
<tr>
<td>Date</td>
<td>Stores the current date</td>
</tr>
<tr>
<td>Time</td>
<td>Stores the current date</td>
</tr>
</tbody>
</table>

Table 4.4: Table for keeping Humidity values

Table name: Temperature

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>Stores the sense Temperature value</td>
</tr>
<tr>
<td>Date</td>
<td>Stores the current date</td>
</tr>
<tr>
<td>Time</td>
<td>Stores the current date</td>
</tr>
</tbody>
</table>

Table 4.5: Table for keeping Temperature values
Create a database using MySQL database
create database mms;
Allow MySQL to use the mms instance :
use mms;

drop table if exists users;
drop table if exists Temperature;
drop table if exists Humidity;
drop table if exists Maxvalues;
drop table if exists Chkdevices;

Users table for storing user-id in MySQL database:
create table users ( username char(30) primary key, passwords char(40) not null );
create table Temperature ( Temperature double, TDate char(11), TTime char(11), TGTime double );
create table Humidity ( Humidity double, HDate char(11), HTime char(11), HGTime double );
Limits tables are created by :
create table Maxvalues ( Humidity double, Temperature double, );
The table used to keep track of previous status of controlling devices are:
create table Chkdevices (}
Device1 integer,
Device2 integer,
);
The limit table consists of only one tuple. The row is initialized by putting zero on every field of the table:
insert into Maxvalues values ('0','0');
The status of devices is initialized when the table is created. This table consists of only one row:
insert into Chkdevices values ('0','0');

4.6.2 ODBC Process Details

The ODBC Data Source Administrator within Windows enables to create DSNs, check driver installation and configure ODBC systems such as tracing (used for debugging) and connection pooling.

To open the ODBC Data Source Administrator on Windows XP:

1. On the Start menu, click Control Panel.
2. In the Control Panel when in Category View click Performance and Maintenance and then click Administrative Tools. If you are viewing the Control Panel in Classic View, click Administrative Tools.
3. In Administrative Tools, click Data Sources (ODBC).

Irrespective of Windows version, following ODBC Data Source Administrator window will appear:
Within Windows XP, add the Administrative Tools folder Start menu to make it easier to locate the ODBC Data Source Administrator. To do this:

1. Right click on the Start menu.
2. Select Properties.
3. Click Customize....
4. Select the Advanced tab.
5. Within Start menu items, within the System Administrative Tools section, select Display on the All Programs menu.

4.6.3 Configuring a Connector/ODBC 3.51 DSN on Windows

To add and configure a new Connector/ODBC data source on Windows, using the ODBC Data Source Administrator:

1. Open the ODBC Data Source Administrator.
2. To create a System DSN (which will be available to all users), select the System DSN tab. To create a User DSN, which will be unique only to the current user, click the Add... button.

You will need to select the ODBC driver for this DSN.
Figure 4.4: Create New Data Source

- Select MySQL ODBC 3.51 Driver, then click Finish.

4. You now need to configure the specific fields for the DSN you are creating through the Add Data Source Name dialog.
4. In the Data Source Name box, enter the name of the data source you want to access. It can be any valid name that you choose.

5. In the Description box, enter some text to help identify the connection.

6. In the Server field, enter the name of the MySQL server host that you want to access. By default, it is localhost. If Database server is on remote, then we need to supply the MySQL Database Server Hostname or IP address.

This is how the Server and the client interact remotely through this database. No need to set or change other parameters in the VB Client or VB Server program as long as the DSN name never changes.

7. In the User field, enter the user name to use for this connection.

8. In the Password field, enter the corresponding password for this connection.

9. The Database pop-up should automatically populate with the list of databases that the user has permissions to access.

10. Click OK to save the DSN.

A completed DSN configuration may look like this:
You can verify the connection using the parameters you have entered by clicking the Test button. If the connection could be made successfully, you will be notified with a Success; connection was made! dialog.

This is how the Server and the client interact remotely through this database using ODBC connectivity. We have created a DSN named “mms” in the entire project. We do not need to set or change other parameters in the VB Client or VB Server program as long as the DSN never changes.

4.7. Web Implementation for Data Acquisition

Installation of web server is the primary task in the web implementation of the data acquisition system. Since the application is developed in Windows environment, IIS (Internet Information Server) is used as a web server. Web server performs functions.

Authentication: Process username and password before satisfying web request.
URL mapping: Maps logical to physical URLs.

CGI/ISAPI environment: Provides Interfaces for execution of Common Gateway Interface (CGI) and Internet Server application programming interface (ISAPI) applications

Sending Response Data: Replies to HTTP requests.

Logging: Creates log entries regarding resource use.

This module implements IIS web server with .NET framework and web application, database server and a browser as a client [10]. The web application is implemented in C#.NET [11]. The whole process is implemented in a three tire model as stated in the following fig. It sends requests to the database server to send updated data for sending the same to client as requested from client through its browser. Client may send HTTP request to Web server for getting updated values of temperature and Humidity or control data for change of new set values of temperature and humidity. Next the Web server sends request to Embedded server for retrieving updated values from database or setting new values of temperature and humidity for initiating control. Embedded server sends updated values or new set values to web server. Web server in the form of HTTP response sends the same to client. These interactions are depicted in the following three tire model given in the Figure 4.7. The application program running at embedded server initiates necessary control by looking at the changed values in the database server. The user can view the maximum values of sensed data and the current set values upon request from the database. The advantage provided by such a web service is that the user can view and change the necessary values from any location.
4.7.1 Implementation of Home/Index page

Some modules of the index/home page (reference to Figure 5.4) with explanations is given below.

The various namespaces used are:

```csharp
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.Odbc;
```

The variables used are:

```csharp
string sql;
OdbcConnection conn;
OdbcCommand comm;
OdbcDataReader dr;
```

Connection to database is established by using the following code:

```csharp
string connectionString = "Driver={MySQL ODBC 3.51 Driver};Server=192.168.0.18;Database=mms;User=root;Password=mysql;";
    conn = new OdbcConnection(connectionString);
    conn.Open();
```

Populating the ODBC Data Reader object with say, Temperature recordset. We first combine the sql statement with the Database connection object, as

```csharp
    comm = new OdbcCommand(sql, conn);
    dr = comm.ExecuteReader();
```
If session is set for a user, then execute the following code snippet if the page is reloaded:

```csharp
if (((string)Session["username"] == "1"))
{
    lblogin.Text = "Log Out";
    lblwelcome.Text = "hello admin!";
    txtHumidityLimit.ReadOnly = true;
    txtTemperatureLimit.ReadOnly = true;
    txtVoltageLimit.ReadOnly = true;
}
```

Set the current date and time for display on the web page, as:

```csharp
txtTime.Text = DateTime.Now.ToLongTimeStringO;
txtDate.Text = DateTime.Today.ToLongDateStringO;
```

Going to the last record of the recordset as last record is the current Temperature data is the most recent data recorded in the database and then display the value as:

```csharp
txtCurrentTemperature.Text = dr.GetValue(0).ToString();
```

Closing of the opened objects is done as:

```csharp
conn.Close();
dr.Close();
comm.Dispose();
conn.Dispose();
```

Similarly, it displays the most recently stored value of Humidity.

Add the values of dates to the list box in the web page. This is required as to know the maximum and minimum values of Humidity and Temperature.

```csharp
sql = "SELECT HDate FROM humidity group by HDate";
conn = new OdbcConnection(connectionString);
```
After selecting a date from the list box, the following code displays the Maximum and minimum values of Humidity and Temperature on that date.

Code for Humidity for maximum and minimum values:

```csharp
class OdbcQuery
{
    string connectionString;
    void OpenO()
    {
        conn.Open();
        comm = new OdbcCommand(sql, conn);
        dr = comm.ExecuteReader();
        while (dr.Read())
        {
            lstboxDates.Items.Add(dr.GetValue(0).ToString());
        }
        conn.Close();
        dr.Close();
        comm.Dispose();
        conn.Dispose();
    }
    string sql = "select max(Humidity) as maxv, min(Humidity) as minv from humidity where HDate ="" + IblMaxMinDate.Text + " group by HDate ";
    void ExecuteQueryO()
    {
        conn = new OdbcConnection(connectionString);
        conn.Open();
        sql = "select max(Humidity) as maxv, min(Humidity) as minv from humidity where HDate ="" + IblMaxMinDate.Text + " group by HDate ";
        comm = new OdbcCommand(sql, conn);
        dr = comm.ExecuteReader();
        while (dr.Read())
        {
            txtMaxHumidity.Text = dr.GetValue(0).ToString();
            txtMinHumidity.Text = dr.GetValue(1).ToString();
            conn.Close();
            dr.Close();
            comm.Dispose();
            conn.Dispose();
        }
    }
}
```

Code for Temperature for maximum and minimum values:
conn = new OdbcConnection(connectionString);
conn.Open();

sql = "select max(Temperature) as maxv,min(Temperature) as minv from temperature
where TDate =" + IblMaxMinDate.Text + ", group by TDate ";
comm = new OdbcCommand(sql, conn);

dr = comm.ExecuteReader();
dr.Read();
txtMaxTemperature.Text = dr.GetValue(0).ToString();
txtMinTemperature.Text = dr.GetValue(1).ToString();
conn.Close();
dr.Close();
comm.Dispose();
conn.Dispose();

The index/home page displays the limits values for Humidity and Temperature.

string sql = "SELECT Temperature,Humidity FROM maxvalues";
OdbcCommand comm = new OdbcCommand(sql, conn);
OdbcDataReader dr = comm.ExecuteReader();
while (dr.Read())
{
    txtTemperatureLimit.Text = dr.GetValue(0).ToString();
    txtHumidityLimit.Text = dr.GetValue(1).ToString();
}
conn.Close();
dr.Close();
comm.Dispose();
conn.Dispose();

The following code displays the current temperature when user clicks button:

string sql = "SELECT Temperature FROM temperature";
OdbcConnection conn = new OdbcConnection(connectionString);
conn.OpenQ;
OdbcCommand comm = new OdbcCommand(sql, conn);
OdbcDataReader dr = comm.ExecuteReaderQ;

while (dr.ReadQ)
{
    txtCurrentTemperature.Text = dr.GetValue(0).ToStringQ;
}
conn.CloseQ;
dr.CloseQ;
comm.DisposeQ;
conn.DisposeQ;

The following code displays the current humidity when user clicks button

sql = "SELECT Humidity FROM humidity";
conn = new OdbcConnection(connectionString);
conn.OpenQ;
comm = new OdbcCommand(sql, conn);
dr = comm.ExecuteReaderQ;

while (dr.ReadQ)
{
    txtCurrentHumidity.Text = dr.GetValue(0).ToStringQ;
}
conn.CloseQ;
dr.CloseQ;
comm.DisposeQ;
conn.DisposeQ;
The following code snippet is executed if the admin user logs for the first time. The index /home page labels and buttons are set accordingly. If the admin session is already set, then the limit values cannot be updated in this section of code:

```csharp
if (((string)Session["username"] == "1") && (btnSetLimit.Text == "Set Limit"))
{
    txtVoltageLimit.ReadOnly = false;
    txtHumidityLimitReadOnly = false;
    txtTemperatureLimit.ReadOnIy = false;
    btnSetLimit.Text = "Update";
    lbnlogin.Text = "Log Out";
    lblwelcome.Text = "hello admin!";
}
```

The following code snippet is executed if admin session is already set. It updates the limit database:

```csharp
if (((string)Session["username"] == "1") && (btnSetLimit.Text == "Update"))
{
    txtHumidityLimit.ReadOnly = true;
    txtTemperatureLimit.ReadOnIy = true;
    txtVoltageLimit.ReadOnIy = true;

    btnSetLimit.Text = "Set Limit";
    lbnlogin.Text = "Log Out";
    lblwelcome.Text = "hello admin!";

    string sql = "UPDATE maxvalues SET Temperature= + txtTemperatureLimit.Text + ",Humidity= + txtHumidityLimit.Text + "";

    OdbcConnection conn = new OdbcConnection(connectionString);
    conn.Open();
```
OdbcCommand comm = new OdbcCommand(sql, conn);

    comm.ExecuteNonQuery();
    conn.Close();
    comm.Dispose();
    conn.Dispose();

Redirection to login page can be done with the following code:

    Response.Redirect("./login.aspx");

Date can be refreshed in the list box to check for new dates, by the following code:
Selecting date from any table will work.
    string sql = "SELECT TDate FROM Temperature group by TDate";
    OdbcConnection conn = new OdbcConnection(connectionString);
    conn.Open();
    OdbcCommand comm = new OdbcCommand(sql, conn);
    OdbcDataReader dr = comm.ExecuteReader();
    lstboxDates.Items.Clear();
    while (dr.Read())
    {
        lstboxDates.Items.Add(dr.GetValue(0).ToString());
    }
    conn.Close();
    dr.Close();
    comm.Dispose();
    conn.Dispose();
4.7.2 Implementing Login Page

Authentication is needed to secure a web page. The updation of controlling limits is granted to authorized users. The passwords are first hashed and compared with the encrypted password in the database. Refer to figure of login page 5.5

4.7.3 Encrypting passwords

ASP.NET has included some hashing functions to encrypt passwords. The two hashing functions that SHA1 or MD5. Both hashing functions are suppose to not let malicious users take the hashed password and get the original password. Passwords in the Database file are stored as their hashed values. Then when a user tries to login, they will send their password to the webserver and the webserver will hash the password and compare it to the hashed password in the database file. If the two hashes match, then the password is correct.

To hash a password, we can use the built in method called Hash Password For Storing In ConfigFile. It can be referred as: Forms Authentication. Hash Password For Storing In Config File ("password", "md5"). The first parameter is the password to be hashed. The second parameter is either "md5" or "sha1" depending on which hashing function used.

Encryption is done in the following way:

String hashedPassword =
FormsAuthentication.HashPasswordForStoringInConfigFile(password.Text,"MD5");

4.7.4 User accounts stored in MySQL Database

Storing user accounts in the web.config file has it's limitations. A lot of larger website will prefer to store the passwords in an SQL database or maybe a separate XML/text file. So in this case, we won't have those optional <credential> and <user> tags in web.config file.

Authorization of user is saved in a session variable. The code for authorizing such login is ::

String username;

String sHashedPassword =
FormsAuthentication.HashPasswordForStoringInConfigFile(password.Text,"MD5");

String sql = "Select username from Users where username=" + UserName.Text + and 
password=" + sHashedPassword +";";}
OdbcConnection conn = new OdbcConnection(connectionString);
conn.Open();
OdbcCommand comm = new OdbcCommand(sql, conn);
OdbcDataReader dr = comm.ExecuteReader();

DR.Read();

    username = dr.GetValue(0).ToString();
    Session["username"] = username;
    if (username == "0")
        Response.Redirect("./login.aspx");
    else
        Response.Redirect("./index.aspx");

In this function, the FormsAuthentication.Authenticate function is used with some SQL code that will query the database for a username/password pair. If such a pair if found, then the credentials are correct and the user session parameters are set and saved in web server main memory for an entire user session and remains there until the user invalids the session parameter by logging out.

The session can be abandoned by writing:

if (((string)Session["username"] == "1"))
{
    Session.Abandon();
    lblogin.Text = "Login";
    btnSetLimit.Text = "Set Limit";
    ......
    ......
}

Where “username” is the session parameter set for a user session. Each user will have a separate session value.

Once login is successful, the user is redirected back to the index webpage. The HashPasswordForStoringInConfigFile method (described in the section above for encrypting password) is used. This function is used because the database will have the passwords hashed [12]
4.7.5 Implementation of forgot password form

If an authorized user forget his password same can be retrieved by using forgot password form. Detail is discussed in section 4.9.1.

4.7.6 Implementation of change password form

The sole purpose of granting authority to the web application lies in the administrator. The administrator has to insert a new user id (generally an email) into the users table database. He then gives a password to the inserted user(s) through user’s email-id or through any communication media and request the users to change their password, (so as not to suspect the administrator putting wrong data on their behalf) as shown in figure 5.7.

The user should be a registered/authorized user and should have his session set prior to changing his password. He should insert his old password which is required for further verification and security enhancement. If the old password match with the password in the database, the updation of password field takes place.

The code for changing password is given below:

```csharp
string count;
String sHashedPassword =
FormsAuthentication.HashPasswordForStoringInConfigFile(password.Text,"MD5");
conn = new OdbcConnection(connectionString);
conn.OpenQ;
sql = "select count(*) from users where username=" + UserName.Text + " and 
passwords=" + sHashedPassword + "";
comm = new OdbcCommand(sql, conn);
dr = comm.ExecuteReaderO;
dr.ReadQ;
count = dr.GetValue(0).ToString();
conn.CloseQ;
dr.CloseO;
comm.DisposeO;
conn.DisposeQ;
```
The TextBox3 and Textbox4 are new password and confirm new password respectively as in change password web page. Update new password now:

```csharp
if (TextBox3.Text == TextBox4.Text)
{
    if (count == "1")
    {
        sql = "UPDATE users SET passwords=" + TextBox3.Text + "" where username =" + UserName.Text + "";
        conn = new OdbcConnection(connectionString);
        conn.Open();
        comm = new OdbcCommand(sql, conn);
        comm.ExecuteNonQuery();
        conn.Close();
        comm.Dispose();
        conn.Dispose();
    }
}
```

There is no way to change the email-id or userid of a registered user. If session is not set, the change password form will redirect the user to the login page and request the user to authenticate with the web application first. The following code performs this:

```csharp
if (((string)Session["username"] == "0")
    Response.Redirect("./login.aspx");
```

### 4.8 Web-Client Process

A Web Client is also called a web browser which send/recieve and display data and web pages in a computer by sending request to a web server, generally a computer, kept at a remote location.
A web browser is the software program used to access the **World Wide Web**, the graphical portion of the Internet. Web implementation of the system is discussed in the following sections.

### 4.8.1 Web Client Anatomy

When the web browser was first launched, usually by double-clicking on the icon on the desktop, a predefined web page appears. This page is referred to as **home page**. Next, typing the URL of the embedded web application server [13] will open the home page of the web application as given in the figure 5.4.

### 4.8.2 The Toolbar of web client

The row of buttons at the top of the browser, known as the **toolbar**, helps to move through the web of possibilities, keeping track of web pages. Since the toolbars for Internet Explorer, Firefox and Navigator, differ slightly, there are a few things common:

- **Back** button (the arrow pointing to the left) returns to the previous web page visited.
- Use the **Forward** button to return to the page we just came from.
- **Home** takes to whichever chosen home page. (If haven't selected one, it will return the default home page, usually the Microsoft, Netscape or Google website.)
- **Reload** or **Refresh** loads the web page again. Sometimes all of the elements of a web page haven't loaded the first time, because the file transfer was interrupted. Also when a web page is downloaded the data is **cached**, meaning it is stored temporarily in client's computer memory. The next time if that page is required, instead of requesting the file from the web server, web browser accesses it from the cache. By reloading the page, this timely data is updated from the web server.
- **Print** lets us make a hard copy of the current page loaded in the browser.
- The **Stop** button stops the browser from loading the current page.
- **Search** connects to directories and search tools on the Microsoft, Netscape or Google websites.
- **Bookmarks** or **Favourites** can record the addresses of websites we want to revisit. Once URL is added to list, we can return to that web page simply by clicking on the link in the list, instead of retyping the entire address.
4.8.3 The Access Indicator

![Access Indicator](image)

Figure 4.8: Access Indicator

All three browsers have small graphics that indicate what the browser is doing. When this image is animated, it means that the browser software, known as a **client**, is accessing data from a remote computer, called a **server**. The server can be located across town or on another continent. The browser downloads these remote files to the computer, then displays them on the screen.

4.8.4 The Status Bar

![Status Bar](image)

Figure 4.9: Status bar

At the bottom of the web browser there is **status bar**. It indicates the progress of web page transactions, such as the address of the site we are contacting, whether the host computer has been contacted and the size and number of the files to be downloaded.

4.8.5 The Scroll Bar

The vertical bar to the right of the browser enable to scroll down and up a web page. If a web page is too wide to fit in the screen, a horizontal scroll bar will appear just above the status bar.

So the web browser is a web client and is the gateway to the Internet to access the monitoring sensor data and to control the sensor data through web pages.
4.9 E-mail process and implementation

Developed System implements a mail server for sending an email message with authentication using ASP.NET 2.0 and C# .NET.

In the first step it is necessary to import the System.Net.Mail namespace. The System.Net.Mail namespace contains the SmtpClient and MailMessage classes which is needed to send the email and specify the user credentials necessary to send authenticated email.

        Using System.Net.Mail;

EmailClient is called to, send the message using the variables from ASP.NET coded page. Next instantiate a System.Net, network Credential object with the necessary authentication info and assigning that object to the Credentials property of SmtpClient object.

For a user to be able to read his mail, it is required to connect his mail server. This is generally not the same machine as the web server. A mail server has to design to interact with the user’s mailbox.

4.9.1 Details of email implementation

When password of a particular user is forgotten, the password can be recovered by sending the password to the user’s email account. The user’s email id is registered by the creating a new user for the application. Figure 5.6 show a typical web form for when the user clicks the forgot password link.

        The user on entering the valid email-id in the email Text box gets a password in his email account. Mails are not sent to invalid email-id or that are not registered.

The protocol used for sending mail is Simple Mail Transfer Protocol. A Mail Transfer Agent (MTA), generally SendMail is used, for sending messages to the mail server. System uses the ASP.NET send() method of the Mail class.

        The code behind the implementation is as follows:

        Private bool SendMail ( string to, string subject, string message, string senderName )
        {
        
Page 113 of 127
Try
{

    Put the username and password of your mail server account

    mail.From = new System.Net.MailAddress ("jecmca@mail.com", senderName);
    mail.IsBodyHtml = true;
    mail.Body = message;

    Put the sender address here
    System.Net.Mail.SmtpClient("mail.com");
    Smtp.UseDefaultCredentials = false;
    Smtp.EnableSsl = false;
    Smtp.Credentials = cred;
    Finally send the mail
    Smtp.Send(mail);
}

Catch (Exception )
{
    Return false ;
}

Return true;
}
References


[8] Prince Anne , Lowe Doug , “ Murach VB.NET database programming with ADO.NET”, SPD New Delhi


