P-2590

# INDOOR AIR QUALITY MONITORING SYSTEM

A PROJECT REPORT

*Submitted by*


| | |
|---|---|
| **PRIYANKAA VIJAYAKUMAR** | **71205205039** |
| **M.RAMYA** | **71205205044** |
| **I.REVATHI** | **71205205047** |


*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

**IN**

**INFORMATION TECHNOLOGY**




**KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY : CHENNAI 600 025**


**APRIL 2009**

P-2590

ANNA UNIVERSITY : CHENNAI 600 025

## BONAFIDE CERTIFICATE

Certified that this project report "**INDOOR AIR QUALITY MONITORING SYSTEM**" is the bonafide work of **"PRIYANKAA VIJAYAKUMAR, M.RAMYA and I.REVATHI "** who carried out the project work under my supervision.

SIGNATURE

**Mrs. L.S. Jayashree, M.E, Ph.D.,**
**ASSOCIATIVE PROFESSOR**
Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.

SIGNATURE

**Dr.Thangasamy,B.E(Hons).,Ph.D.,**
**DEAN**
Dept of Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore – 641 006.

The candidates with University Register No 71205205039, 71205205044 and 71205205047 were examined by us in the project viva-voice examination held on 28.4.2009

INTERNAL EXAMINER

EXTERNAL EXAMINER

# DECLARATION

We,

| | |
|---|---|
| **Priyankaa Vijayakumar** | **71205205039** |
| **M.Ramya** | **71205205044** |
| **I.Revathi** | **71205205047** |

hereby declare that the project entitled **"INDOOR AIR QUALITY MONITORING SYSTEM"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date:

Priyankaa. Vijayakumar M. Ramya I.Revathi

[Priyankaa Vijayakumar] [M.Ramya] [I.Revathi]

Project Guided by

29/4/09

**[ L.S Jayashree M.E., Ph.D]**

# ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc., F.I.E.,** Vice Chairman Dr.K. **Arumugam B.E., M.S., M.I.E.,** Correspondent **Shri.M.Balasubramaniam and Joint Correspondent Dr.A.Selvakumar** for all their support and ray of strengthening hope extended. We are immensely grateful to our Vice Principal **Prof.R.Annamalai,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy, B.E(Hons).,Ph.D.** Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-ordinator, **L.S Jayashree M.E., Ph.D.,** Asso.Prof, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide and extend our gratitude towards **L.S Jayashree M.E., Ph.D.,** Asso.Prof, Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

## ABSTRACT

Our project is designed to monitor the concentration of hazardous gases at a desired location and send the data to a centralized remote server. The Indoor Air Quality Monitoring System (IAQMS) is designed to monitor the concentration of hazardous gases which have been specified by the Environment Protection Agency (EPA) as the most hazardous, and the most common in an indoor environment. The system will enable building owners and managers to store electronic reports and data to easily maintain compliance with EPA record keeping and notification requirements. IAQMS also stores room specific information, threshold information, abatement information and generates reports and notifications automatically.

IAQMS consists of modules like *sensor module, data acquisition, data pre-processing, data storage, notification generation and report generation.* It operates in single user mode or can be configured in multi-user client-server operation mode. The concentration of gas in *parts per million* (ppm) in the indoor environment are first detected and converted to digital data. It is then displayed to the client side in intervals and sent to the server side for storage and report generation. The notifications that are generated can be used to set in motion the required corrective actions and the reports can be used for analysis to discover solutions to a recurring problem.

IAQMS is designed to be employed in any multi-storied building which employs a climate control system. It is an affordable, easy to use system for managing information regarding the concentrations of gas in a specific building or group of buildings. The server side is kept under the control of the building owner, while the client side and the sensor module will be installed at the tenant locations throughout the building. IAQMS can be easily extended to the commercial market to provide easy access to the concentrations of hazardous gases throughout a building.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

## LIST OF ABBREVATIONS

| | |
|---|---|
| IAQ: | Indoor Air Quality |
| EPA: | Environment Protection Agency |
| IAQMS: | Indoor Air Quality Monitoring System |
| ppm: | Parts per million |
| OSHA: | Occupational Safety and Health Administration |
| SQL: | Structured Query Language |
| HVAC: | Heating, Ventilation and Air Conditioning |

# *INTRODUCTION*

# 1. INTRODUCTION

## 1.1GENERAL

### 1.1.1 Indoor Air Quality Concerns

In recent years, the urbanization process has been accelerated with the development of economy; more and more modern architectures are emerged such as the top-grade office building, top-grade housing district, etc. Modern architecture is no longer the place where people take shelter from rain and wind. The more important function is to offer a healthy and comfortable environment for living and working. The oil crisis of the 1970s and the rise in oil prices motivated people to implement energy conservation strategies. Buildings were fitted with additional insulation and reduced ventilation rates. The reduction of forced and natural ventilation rate led to indoor pollutant concentrations increasing which resulted in high health risks of pollutants. People who are in the building for a long term perform more and more serious pathological reactions, including headaches; eye, nose, and throat irritation; a dry cough; dry or itchy skin; dizziness and nausea; difficulty in concentrating; fatigue; and sensitivity to odors. Therefore, the indoor environmental quality problems of modern architecture have become an increasing concern of the public.

In the last several years, a growing body of scientific evidence has indicated that the air within homes and other buildings can be more seriously polluted than the outdoor air in even the largest

and most industrialized cities. Other research indicates that people spend approximately 90 percent of their time indoors. Thus, for many people, the risks to health may be greater due to exposure to air pollution indoors than outdoors. The last three decades have seen a sharp rise in the number of complaints by office building occupants about indoor environments. Many of these complaints stem from poor Indoor Air Quality (IAQ). Poor IAQ can affect the comfort, productivity, and sometimes even the health of building occupants. Even when pollution levels are too low to pose a health risk, they can affect occupants' alertness and ability to concentrate and can lead to low morale.

### 1.1.2 Causes of Indoor Air Problems

Indoor pollution sources that release gases or particles into the air are the primary cause of indoor air quality problems in homes. Inadequate ventilation can increase indoor pollutant levels by not bringing in enough outdoor air to dilute emissions from indoor sources and by not carrying indoor air pollutants out of the home. High temperature and humidity levels can also increase concentrations of some pollutants

### 1.1.3 Pollutants and Sources of Indoor Air Pollution

- Asbestos
- Biological Pollutants
- Carbon Monooxide
- Formaldehyde/Pressed Wood Products
- Household Cleaning and Maintanence, Personal Care
- Lead
- Butane
- Nitrogen Dioxide
- Pesticides
- Radon
- Respirable Particles
- Secondhand Smoke/ Environmental Tobacco Smoke
- Stoves, heaters and chimmneys

## 1.2 PROBLEM DEFINITION

Our project is an Indoor Air Quality Monitoring System. The objective of the project is to obtain the concentration for various gases from various locations in a building and send it to a centralized location for analysis. It consists of a client side and a server side. The client side is includes the sensor module, data acquisition, data preprocessing. The server side consists of data storage, notification generation and report generation. A number of sensors are connected to one client. Multiple clients are in turn in connected with a single server.

The system begins with the sensor module which is involved with obtaining continuous real time information on the concentration of hazardous gases as voltage levels. The data is produced at rapid intervals. This analog voltage information is then preprocessed by converted it to the digital form which is acceptable to a computer system by manipulations in a microchip. This data is then passed to the client side in ASCII format.

At the client side the ASCII data is converted to decimal form. In a real time environment the gas levels have to be monitored only at intervals specified by the EPA. Hence the concentration of gas , at pre-specified intervals can be monitored, while the remaining data is discarded. The client side also does the manipulations by which the gas levels are converted to the required format of ppm(parts per million). The client side information is then sent to the server side.

At the server side, the first activity is data storage, where the gas history is automatically stored in a database. The server side is also involved with obtaining user input to store building details of the tenant locations. Different rooms have different monitoring requirements. These details along with the dimensions and type of room are saved in the server side system to facilitate easy management. The server side also stores corrective actions and threshold information. The notifications are generated when the gas concentrations , monitored by the client exceeds the threshold and sent back to the client side. The server side can also generate reports. The reports are generated upon selecting the required date. The activity for each date is stored in the gas history database along with the time. Upon selecting the date, a datasheet is generated calculating the average values of the gas for the date. Reports are then generated in the form of a line graph plotted between gas level in ppm along the y-axis and the time in the x-axis.

## 1.3 OBJECTIVE OF THE PROJECT

The objectives of the system are as follows

- To monitor the indoor air quality at regular intervals.
- To collect data from desired locations and store it in a centralized location in a systematic way.
- To generate notifications to be sent to the corresponding locations.
- To provide easy access to threshold information and corrective actions to be employed upon a request from the clients.
- To provide an effective management system which is capable of storing and retrieving information as per the user specification.
- To generate reports in the form of graphs and tables which can be used for trend analysis and interpretation.

# LITERATURE REVIEW

# 2. LITERATURE REVIEW

## 2.1 FEASIBILITY STUDY

### 2.1.1 Existing System

IAQ systems in the present day largely comprise of monitoring systems which gather information and store it without and give off alarms when required. An example of the above mentioned contemporary systems, is a product by the world renown Ecologic Systems, ADAM which is management system used to track gas concentration levels in specified locations. It can store and provides easy access to environmental, indoor air quality data, samples and documents. However it is not equipped to cope with the requirement of centralized data gathering which is a necessity in the contemporary fast paced world that exists. The existing system also does not generate statistical reports which will enhance understanding and make future improvements possible. Generating reports are necessary for analysis and investigation. Tracing and monitoring the gas levels in a building and generating timely notifications is highly sophisticated work, especially when the area in consideration is very large with larger possibilities of disaster. A monitoring system which is compatible to only a small area is not the solution for today's challenge. We need a large scale monitoring and management system which generates relevant information depending on the location where it is employed. We cannot use a one-size-fit- all strategy, and the management system must be capable of taking into consideration the specifications of the location where it is employed and provide user friendly reports.

### 2.1.2 The Proposed System – Indoor Air Quality Monitoring System

In this project, we place the sensor module in the strategic locations in the considered multi-storied building. A number of sensors are to be connected to a single client. The various sensors placed in strategic locations around the room are connected with the client computer and in turn the client software gets access to the continuous real time data which is generated. Whenever the pre-specified interval is completed the data is acquired and processed into the correct format and then sent to the server side. The server in turn stores the received information in a database along with the threshold information, the corrective actions that need to be taken in case of an emergency, and the room specific information in detail. It generates automatic notifications and sends it to the client side. On request from the user on the server side, the server side software can generate statistical reports and data sheets which can be used for analysis and investigation

**Advantages of the Proposed System**

- The proposed system saves time and reduces risk by providing easy access to records and information
- It also reduces the risk of litigation and citations by helping to ensure that IAQ is not inadvertently disturbed.
- The system saves time by producing reports and notifications directly to the user.
- It also enables quick logging in of drastic jumps in the gas levels and notification to begin quick relevant action when needed.

## 2.2 HARDWARE REQUIREMENTS

| | |
|---|---|
| **Processor** | : Intel Pentium III 2.66 GHz |
| **Primary Memory** | : 128 MB |
| **Secondary Memory** | : 10 GB |
| **Other equipments** | : Standard 101 Keys Keyboard, Optical Mouse |
| **Sensor** | : MQ- 2 |
| **Microcontroller** | : PIC 16F877 Microcontroller |
| **Voltage Regulators** | : LM 7805 , LM 7812, LM 7912 |
| **Operational Amplifier** | : UA 741 |
| **Transformer** | : Step down transformer |
| **Cable Required** | : RS 232, 16-pin FRC cable |

## 2.3 SOFTWARE REQUIREMENTS

| | |
|---|---|
| **Operating System** | : Windows XP |
| **Front end** | : Visual Basic 6.0, Java 1.5 |
| **Software Required** | : J2SDK 1.5, Microsoft Visual Studio 6.0, Win X-Talk |
| **Back End** | : MS- Access |

## 2.4 HARDWARE OVERVIEW

### 2.4.1 MQ-2 SEMI-CONDUCTOR SENSOR FOR COMBUSTIBLE GAS

**Features:**

- Good sensitivity to Combustible gas in wide range
- High sensitivity to LPG, Propane, Hydrogen, Butane.
- Long life and low cost
- Simple drive circuit

**Application:**

- Domestic gas leakage detector
- Industrial Combustible gas detector
- Portable gas detector

**Basic Test Loop:**

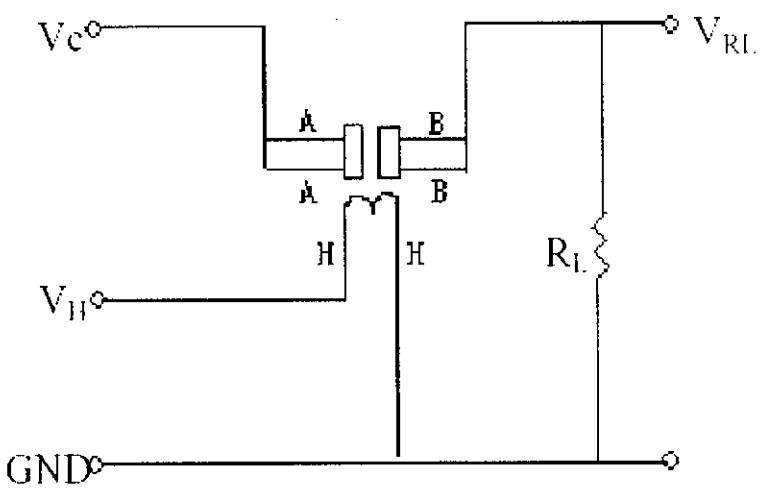



**Fig 2.4.1.1 Sensor's Testing circuit**

Heater voltage **(VH)** -- Supply certified working temperature to the sensor

Test voltage **(VC)** -- Detect voltage (VRL) on load resistance (RL)whom is in series with sensor.

Power of Sensitivity body**(Ps)**: Ps=Vc2×Rs/(Rs+RL)2

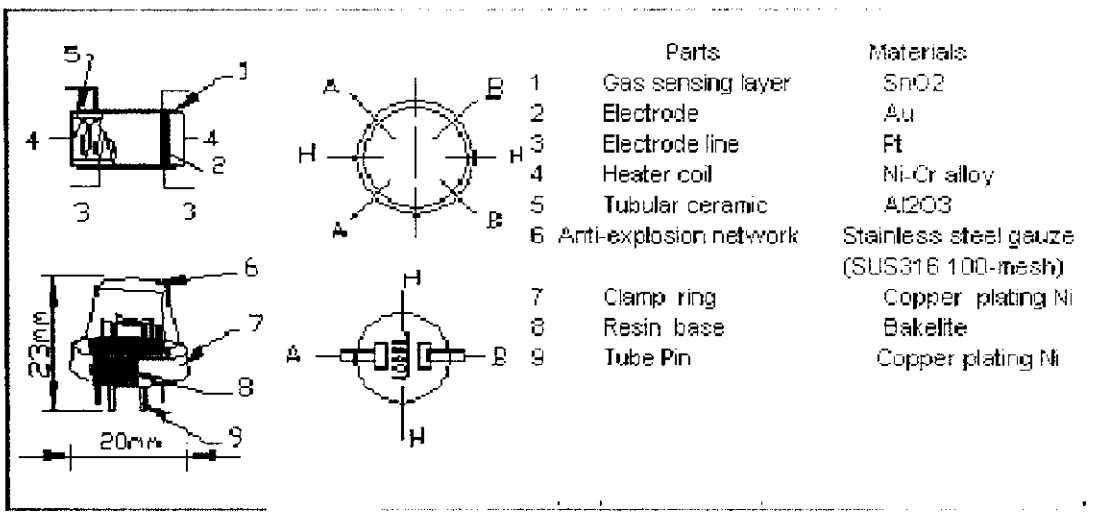
Resistance of sensor **(Rs)**: Rs=(Vc/VRL-1)×RL

**Technical Data**

| Model No. | | | MQ-2 |
|---|---|---|---|
| Sensor Type | | | Semiconductor |
| Standard Encapsulation | | | Bakelite (Black Bakelite) |
| Detection Gas | | | Combustible gas and smoke |
| Concentration | | | 300-10000ppm<br>( Combustible gas) |
| Circuit | Loop Voltage | $V_c$ | ≤24V DC |
| | Heater Voltage | $V_H$ | 5.0V±0.2V ACorDC |
| | Load Resistance | $R_L$ | Adjustable |
| Character | Heater Resistance | $R_H$ | 31Ω±3Ω (Room Tem.) |
| | Heater consumption | $P_H$ | ≤900mW |
| | Sensing Resistance | $R_s$ | 2KΩ-20KΩ(in 2000ppm [illegible]) |
| | Sensitivity | S | Rs(in air)/Rs(1000ppm isobutane)≥5 |
| | Slope | α | ≤0.6($R_{5000ppm}/R_{3000ppm}$ $CH_4$) |
| Condition | Tem. Humidity | | 20℃±2℃; 65%±5%RH |
| | Standard test circuit | | Vc:5.0V±0.1V;<br>$V_H$: 5.0V±0.1V |
| | Preheat time | | Over 48 hours |

**Tab. 2.4.1.1 Technical Data- MQ 2**

## Structure and Configuration




| | Parts | Materials |
|---|---|---|
| 1 | Gas sensing layer | SnO2 |
| 2 | Electrode | Au |
| 3 | Electrode line | Pt |
| 4 | Heater coil | Ni-Cr alloy |
| 5 | Tubular ceramic | Al2O3 |
| 6 | Anti-explosion network | Stainless steel gauze (SUS316 100-mesh) |
| 7 | Clamp ring | Copper plating Ni |
| 8 | Resin base | Bakelite |
| 9 | Tube Pin | Copper plating Ni |

**Fig. 2.4.1.2 Structure and Configuration of MQ 2**

Structure and configuration of MQ-2 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, TinDioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-2 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

### 2.4.2 PIC 16F877 MICRO CONTROLLER

**PIC** is a family of Harvard architecture microcontrollers made by Microchip Technology.The name PIC initially referred to "**Peripheral Interface Controller**". The PIC16F877A features 256 bytes of EEPROM data memory, self programming, an ICD, 2 Comparators, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 capture/compare/PWM functions, the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit ($I^2C$™) bus and a Universal Asynchronous Receiver Transmitter (USART).

**Applications:**

- A/D applications in automotive
- Industrial
- Appliances
- Consumer applications.

## Pin Diagram

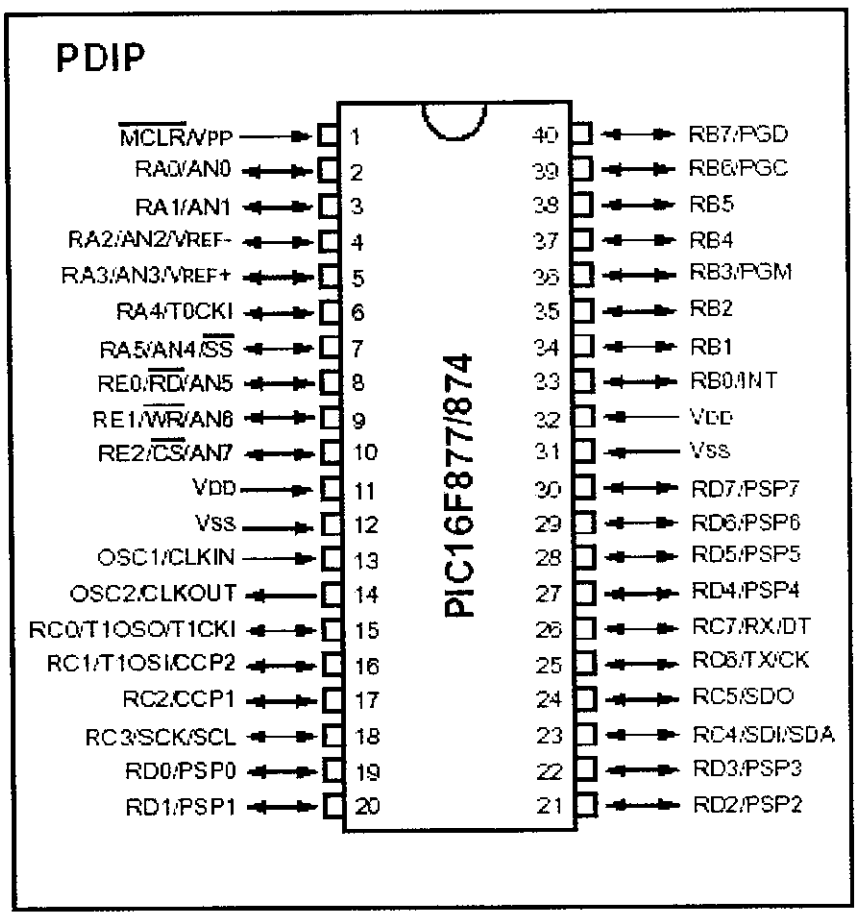



**Fig 2.4.2.1 Pin Diagram – PIC 16F877**

## 2.4.3 SERIES VOLTAGE REGULATORS

### General Description

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and currents.

### Features

- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

**Voltage Range**

- LM7805C 5V
- LM7812C +12V
- LM7912C -12V

**Connection Diagram**

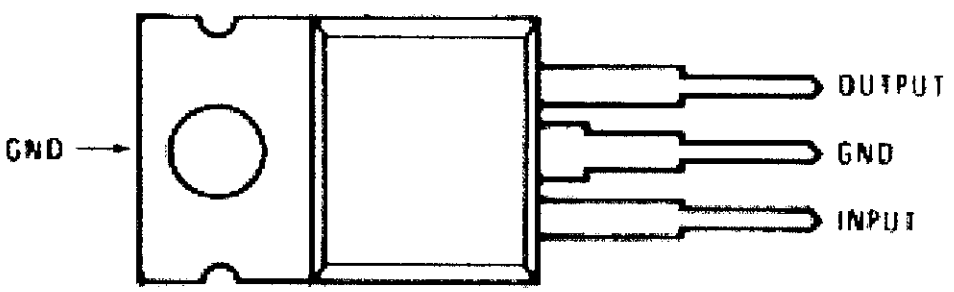
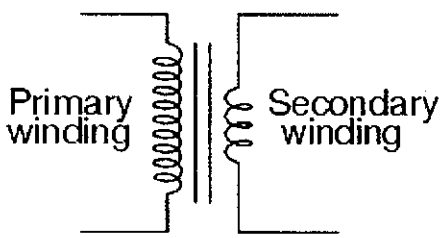



**Fig. 2.4.3.1 Pin Diagram – LM78XX**

### 2.4.4 STEP-DOWN TRANSFORMERS

Step down transformer is used divide voltage and current in AC circuits. A transformer designed to reduce voltage from primary to secondary is called a step-down transformer. Transformers "step down" voltage according to the ratios of primary to secondary wire turns




Voltage transformation ratio = $N_{secondary} / N_{primary}$

Current transformation ratio = $N_{primary} / N_{secondary}$

Where , N = number of turns in winding.

## 2.4.5 OPERATIONAL AMPLIFIERS

### UA741:

- Large input voltage range
- No latch-up
- High gain
- Short-circuit protection
- No frequency compensation
- Required
- Same pin configuration as the ua709

### Applications:

- Summing amplifier
- Voltage follower
- Integrator
- Active filter
- Function generator

### Pin Description:

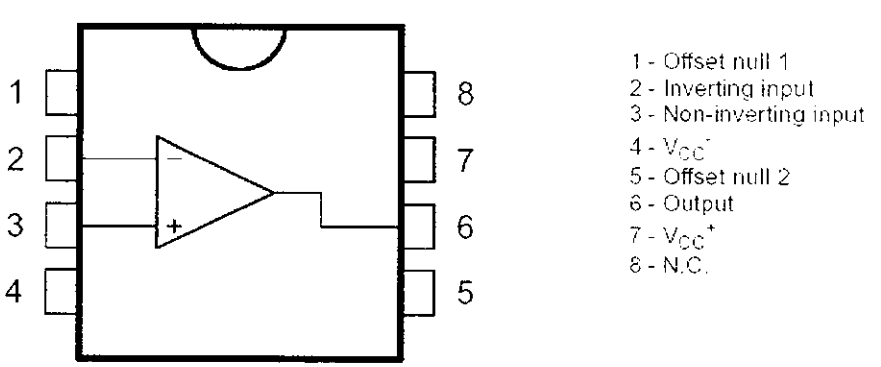



**Fig. 2.4.5 .1 Pin Diagram :UA741**

### 2.4.6 Bridge Rectifier

A bridge rectifier makes use of four diodes in a bridge arrangement to achieve full-wave rectification. This is a widely used configuration, both with individual diodes wired as shown and with single component bridges where the diode bridge is wired internally

**Circuit Diagram** **Diodes – IN 4007**

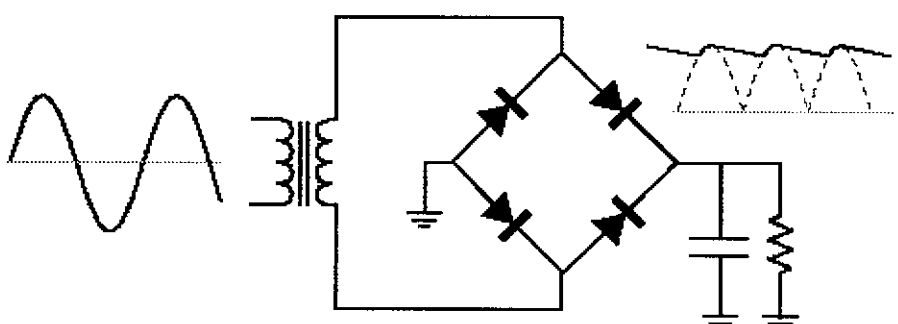


**Fig. 2.4.6.1 Bridge Rectifier**

## 2.5 SOFTWARE OVERVIEW

### 2.5.1 FRONT END SOFTWARE TOOLS

**Java**

Java is a programming language of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is simple, object oriented, familiar, robust , secure, architecture neutral , portable ,Interpreted, threaded, and dynamic. Portability achieved by compiling the Java language code, not to machine code but to Java byte code – instructions analogous to machine code but intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. Standardized libraries provide a generic way to access host specific features such as graphics, threading and networking. Just-in-time (JIT) compilation technique translates Java byte code into native code the first time that code is executed, then caches it.

**Visual Basic**

Visual Basic is the third-generation event-driven programming language and integrated development environment from Microsoft for its COM programming model. VB is also considered a relatively easy to learn and use programming language, because of its graphical development features and BASIC heritage. Visual Basic was derived from BASIC and

enables the rapid application development (RAD) of graphical user interface (GUI) applications, access to databases using Data Access Objects, Remote Data Objects, or ActiveX Data Objects, and creation of ActiveX controls and objects. Scripting languages such as VBA and VBScript are syntactically similar to Visual Basic, but perform differently.

### 2.5.2 EMBEDDED SOFTWARE TOOLS

#### CCS compiler

CCS provides complete, low cost, embedded software tools designed for Microchip PIC®MCU and dsPIC®DSCs. The CCS C Compiler is comprised of Standard C operators and built-in libraries that are specific to PIC®MCU registers, and access to hardware features from C. CCS offers 23 technology specific, all-inclusive Development Kits that bring hardware and software together to provide an innovative package of development tools. The CCS Development Kit allows engineers to design, develop, implement, and test applications directly on PIC® MCUs and dsPIC® DSCs. The CCS C Compiler features provide ample function for your development needs, including: standard C pre-processor directives, operators & statements, built-in libraries supporting all chips, MPLAB® IDE integration, source code drivers and automatic linking for multiple code pages . The compiler includes built-in functions to access the PIC® MCU hardware

**Win X-Talk**

X-talk program is a communications package using asynchronous protocols to provide a utility for transferring files to and from the E Multi-Access System (EMAS) via a Terminal Control Processor (TCP). Information passing between the IBM PC user and the remote host, is buffered by the XTALK program before display on the PC screen.

**PIC downloader**

It is compatible with the HI-Tech's for the PIC16F87x processors. It runs under Windows 9x/ME/NT/2000 .The baud rate selection from 2400 to 56000 Bd. The serial port selection COM1-6. It works with EEPROM data in the hex file (Shane Tolmie's or my boot loader). PIC downloader controls RESET and trigger pin of microcontrollers . It is inclusive of the assembler boot loader for all non "C" programmers. The boot loader is very easy recompilated for various microcontrollers, pin/time triggering, baud rates and quartz frequencies.

### 2.5.3 BACK END TOOLS

**Microsoft Office Access**

**Microsoft Access**, is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the Microsoft Office suite of applications and is included in the Professional and higher versions for Windows and also sold separately.

There is no version for MacOS or for Microsoft Office Mobile. Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other Access databases, Excel, SharePoint lists, text, XML, Outlook, HTML, dBase, Paradox, Lotus 1-2-3, or any ODBC-compliant data container including Microsoft SQL Server, Oracle, MySQL and PostgreSQL. Software developers and data architects can use it to develop application software and non-programmer "power users" can use it to build simple applications. It supports some object-oriented techniques but falls short of being a fully object-oriented development tool.

# DETAILS OF METHODOLOGY EMPLOYED

# 3. DETAILS OF METHODOLOGY EMPLOYED

## 3.1 Description of Modules

The methodology describes the flow of information from one module to another .The Sensor module involves the hardware design of the sensor. A number of sensor modules are connected to the data acquisition module. The data is then passed on to the data pre-processing module so that the data can be converted to the correct format. This ends the client side. The data is then passed form multiple clients to a single centralized server. The server side software beings with the automatic data storage module, which ensures no loss of data. The data storage module gives way to the notification generation module which involves notifying the client when the threshold is exceeded. The report generation module, which is called at user's request gives a visual representation of the gas levels in the form of graphs.

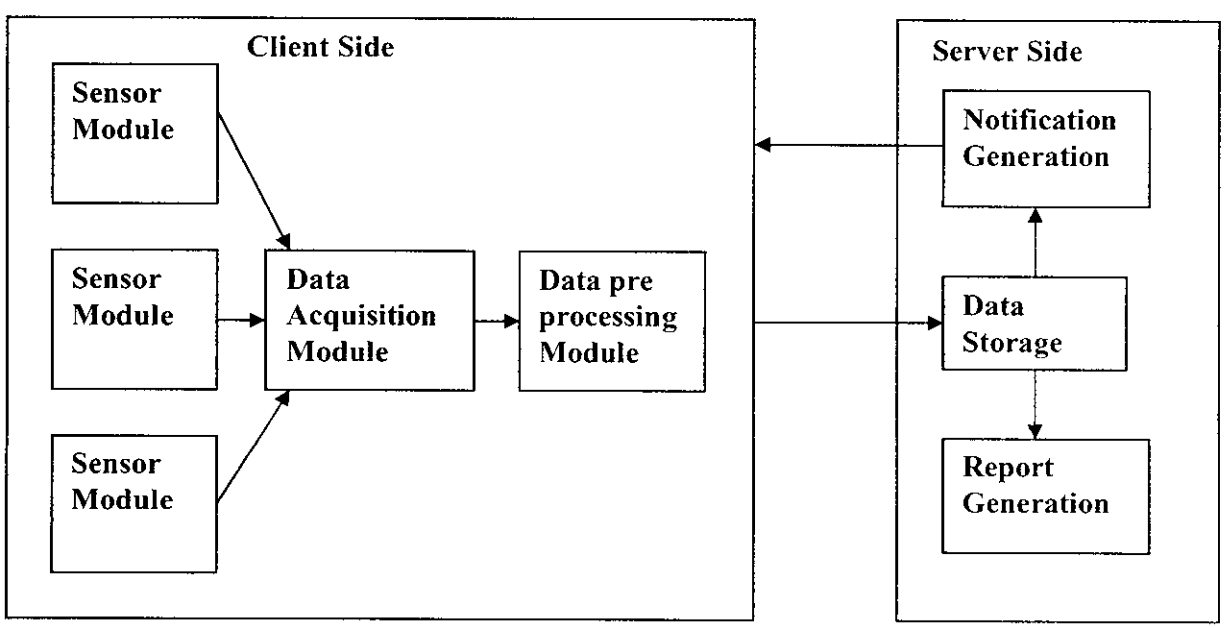



**Fig. 3.1.1 Modules in IAQMS**

## 3.2 DATA ACQUSITION SYSTEM

### 3.2.1 Sensor Module

IAQMS begins with sensors which are capable of monitoring the indoor air and retrieving the gas concentration levels. A number of sensors are available in the market today which are sensitive to numerous gases. We have created a prototype of the full fledged system by employing MQ-2 sensor which detects the concentration of Butane gas.

The power supply of 230V AC is first stepped down to 12V AC using a Step Down transformer. The Bridge rectifier circuit which is defined below, is used to convert the 12V AC to 12V DC.

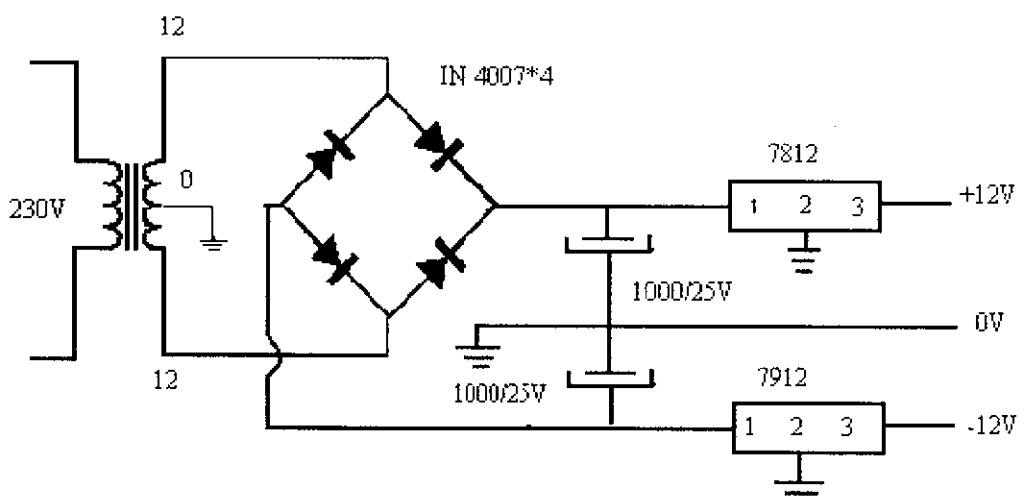



**Fig. 3.2.1.1 Circuit Diagram of Bridge Rectifier**

The above specified MQ-2 sensor, produces analog voltage corresponding to the gas concentration . The operational voltage of the MQ-2 is 12V DC. The output of the Bridge Rectifier circuit is given as input to the MQ-2 sensor as $V_c$. The sensor also obtains heater voltage ($V_H$) as input from the circuit. The output of the sensor is a feeble signal which is given to the voltage regulator and operational amplifier. This modified analog voltage signal is given to used the subsequent modules for manipulation.

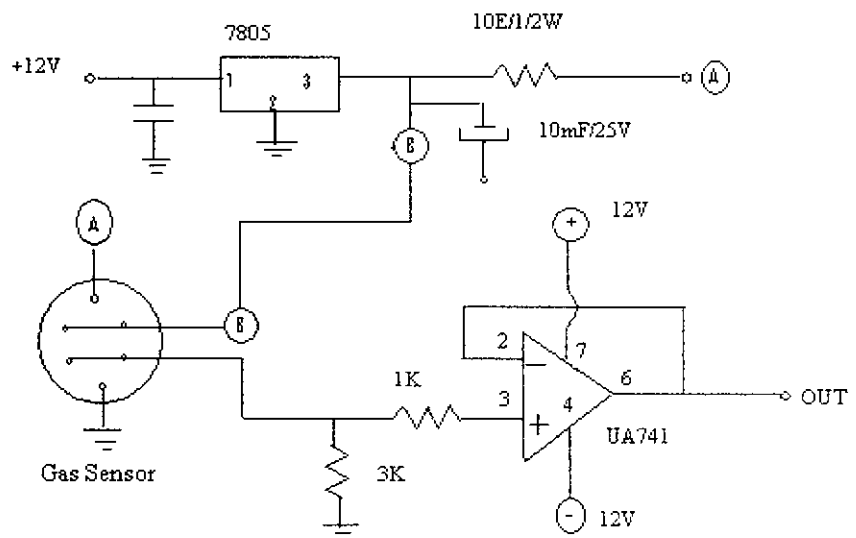



**Fig. 3.2.1.2 Circuit Diagram of Sensor Module**

## 3.3 DATA PREPROCESSING

### 3.3.1 Interfacing Sensor module and PIC 16F877

The analog output of the sensor module has to be given to the PIC 16F877. A 16 pin FRC cable is used for interfacing purpose.

### 3.3.2 A/D conversion

The analog output of the data acquisition module has to be converted to digital form. For this purpose, we use the Analog-to-Digital module of PIC 16F877 Micro-controller. The programming of the chip is done in the Vput Rev-01 kit. P12 of the above mentioned kit, corresponds to A/D conversion module of the PIC micro-controller. P12 of the kit has 16 pins of which pin 1-8 can be used for analog input and pin 16 is ground. An embedded C program written and complied using the CCS compiler enables us to obtain the output at the serial port of the kit.

### 3.3.3 Interfacing the PIC Micro-controller and the client PC

The data obtained after the analog to digital conversion has to be given as input to the client PC. To facilitate the above task, we use an RS232 cable to connect the serial port of the micro controller kit and the PC. Suitable coding and software allows the data, which is generated at the serial port of the kit and transmitted through the cable to the PC, to be displayed on the Win X-Talk Screen.

### 3.3.4 Client Side Software

The data acquisition and the data pre-processing modules give way to the client side application. The client side software is written in Visual Basic. The Spy++ software which is part of the Microsoft Visual Studio 6.0 Tools is used to obtain a handle on the Win X-Talk window which displays the information obtained from the serial port. Spy++ is a Win32 based utility which has a Finder Tool to select a window by mouse positioning. A handler in the hexadecimal format can be obtained. This hexadecimal format is converted to decimal. This 10 digit decimal handler is used as input to the VB code. This code is written to retrieve the contents of the Win X-Talk window. The digital values in ASCII format are converted integer to reflect the gas concentration in ppm. These values are retrieved and processed on completion of a pre-specified timer interval. This integer data is sent to the server side.

## 3.4 DATA STORAGE MODULE

The data storage module in the server side, takes care of storing the data which is sent from the client side. The gas concentration levels are stored in a database along with the room Id and the system date and time. The data storage module populates the HistoryTb. This data can be used as a basis for the report generation. The Data Storage module ensures that no resources need to be wasted on storage at the client side. It is a complete information repository which can take updates from numerous clients and store the data systematically. The data storage module is written in Java, using embedded SQL commands for updating the values in the database.

## 3.5 NOTIFICATION GENERATION MODULE

Notification Generation is also part of the server side software of the IAQMS. This module is responsible for generating client-specific notifications and sending it to the client side PC. The gas concentration levels that are sent from the client side to the server side, periodically, are compared with the threshold values. The threshold values are stored in the database in the ThresholdTb table. Threshold details are retrieved and compared with the incoming values. If the incoming gas concentration level is higher than the stored threshold, a notification is sent back to the client from whom the value was originally sent.

## 3.6 REPORT GENERATION MODULE

Report generation module is also part of the server side software. This module retrieves information from the HistoryTb to generate the graphs. The user selects the location using the Room ID and the date. Depending on this input the data is retrieved and the average gas concentration values for that date is calculated and displayed. The module also generates a bar graph depicting the average gas concentration in value in ppm. This module also generates a line graph for each gas separately which plts the gas concentration level in ppm along the y-axis against time in the x-axis.

## 3.7 DATABASES

A database in general is a structured collection of information. Databases are used in our applications, spanning virtually the entire range of the software. We have designed, developed and populated four tables to carry on the requirements of the project. They are explained in detail below.

### 3.7.1 Threshold Details Table (ThresholdTb)

| Field Name | Data Type |
|---|---|
| GasName | Text |
| PEL | Number |
| TLV | Number |
| STEL | Number |
| IDLH | Number |

**Tab 3.7.1.1 Design view of ThresholdTb**

The Threshold details of the six gases which are taken into consideration are stored in the ThresholdTb. The first field is a general field which is used to store the name of the gas. In our case the gases are CO2,CO,Pb,Nicotine,Butane,NH3. The remaining fields refer to the four accepted thresholds according to the EPA standard. Threshold limit Value(TLV) is an estimate of the average safe airborne concentration of a substance in representative conditions under which it is believed that nearly all people maybe repeatedly exposed day after day without any adverse effect. TLVs are published annually by the ACGIH (American Conference of Government Industrial Hygienists). PEL (Permissible Exposure Limit) is a legal limit in the United States for exposure of an employee to a chemical substance or physical agent. It is specified by OSHA. This means that, for

limited periods, a person may be exposed to concentrations higher than the PEL, so long as the average concentration over eight hours remains lower. STEL (Short term Exposure Limit) is one that addresses the average exposure over a 15-30 minute period of maximum exposure. IDLH(Immediately Dangerous To Life And Health) represents the maximum concentration level of a substance from which one could escape within 30 minutes without escape-impairing symptoms or any irreversible effects. The ThresholdTb is used to store all these gas concentration limits for a particular gas. This table is accessed every time concentration level is retrieved from the environment by the data acquisition module. The existing gas concentration is compared with the values in the table to give out notifications.

### 3.7.2 Corrective Actions Table (CorrectiveTb)

| Field Name | Data Type |
|---|---|
| ID | Number |
| Gas | Text |
| Action | Text |

**Tab 3.7.2.1 Design view of CorrectiveTb**

The details on the Corrective actions are stored in the CorrectiveTb. It consists of fields such as ID, gas and action. The first two fields are simply used to identify the gas. The corrective actions field contains details which can be implemented to address a sudden surge in a particular gas's concentration. These actions, when implemented, can prevent or mitigate the hazard.

### 3.7.3 Gas History Storage Table (HistoryTb)

| Field Name | Data Type |
|---|---|
| roomed | Text |
| Date1 | Text |
| Time1 | Text |
| Gas | Text |
| Value1 | Number |

**Tab 3.7.3.1 Design view of HistoryTb**

The details on the gas history are stored in the HistoryTb. It contains the day to day gas concentrations, each time it is retrieved. It is a complete information repository on the gas concentration characteristics of a client location. It is the end product of the data storage module. It consists of five fields, roomID, date, time, gas, value. The roomID field takes on a unique string which is used by the system to identify the location where the sensor module is employed. The date and the time field stores the system date and time when the gas concentration retrieval was carried out. The gas value is the most important and it stores the gas concentration levels in ppm, in the form of an Integer.

### 3.7.4 Client Location Information Table (RoomTb)

| Field Name | Data Type |
|---|---|
| ID | Number |
| RoomID | Text |
| RoomName | Text |
| RoomAddress | Text |
| RoomVolume | Text |
| UserID | Text |
| PhoneNumber | Number |
| Gas1 | Text |
| Gas2 | Text |
| Gas3 | Text |
| Gas4 | Text |
| Gas5 | Text |
| Gas6 | Text |

**Tab 3.7.4.1 Design view of RoomTb**

The details of the client location are stored in the RoomTb. It contains user specified information on the characteristics of the client locations. The fields include a roomID and a userID. Both values are unique. The userID is generated by the system. The roomID uniquely identifies the client's location. The RoomName field stores the colloquial name of the type of the room (e.g gymnasium, hotel, etc). The RoomVolume, RoomAddress, PhoneNumber store the details of the room as entered by the user. The table also stores the number of gases that are to be monitored and the names of the gases.

# CONCLUSION

# 4. CONCLUSION

Indoor Air Quality Monitoring System has been successfully developed. The IAQMS works well in any indoor environment and it generates accurate reports that reflect the conditions of the environment where it is deployed. The changes in the gas concentration levels are also detected instantly, which ensures that there will be no delay in generating the notifications if an erratic change in the gas concentration level occurs in the real time environment where it is employed.

The promising feature of the IAQMS is that it is linked with a Management system. The IAQMS uses economically feasible hardware. Since most locations already have computer systems it can easily be deployed in the existing infra structure. The software system is user friendly. The system is very easy to implement in a client location and has very few limitations.

# FUTURE ENHANCEMENTS

# 5. FUTURE ENHANCEMENTS

Although a lot of advancements have been done in the field of gas sensors to give more accurate and timely information, as well as in the field of HVAC (Heating, Ventilation and Air Conditioning) and demand controlled ventilation, very little has been achieved in the field of analysis. It is very important to analyze the causes for sudden or periodic rise in hazardous gases in a building. Pattern recognition and regression analysis can be used to see if there is any correlation between the location's features and the rise in the gas levels. Analysis can also be done to investigate if there is any correlation between sudden rise in gas concentration levels in a given location and the time of occurrence. If this analysis is completed successfully, demand controlled ventilation systems can be designed to become activated even before a rise in the gas concentration level occurs, to provide good ventilation ergonomics.

Hence the IAQMS can be extended in the future to do building specific pattern recognition. This can ensure prevention and control of erratic rise in gas concentration levels more effectively.

# APPENDIX

# 6. APPENDIX

## APPENDIX – A : SOURCE CODE

### SERVER SIDE:

```
/*
 * Main.java
 *
 * Created on March 29, 2009, 6:52 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
package javaapplication11;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Date;
class thread1 implements Runnable
{
  public void run()
  {
    try
```

```
{
    int k=0;
    ServerSocket ser=new ServerSocket(1237);
    Socket s1=new Socket();
    System.out.println("conn1");
    s1=ser.accept();
    System.out.println("conn2");
    BufferedReader in=new BufferedReader(new
InputStreamReader(s1.getInputStream()));
    char b[]=new char[4];
    int m;
    in.read(b);
    m=b.length;
    System.out.println("");
    for(int n=0;n<m;n++)
    {
      System.out.print(b[n]);
    }
    while(true)
    {
    in.read(b);
    m=b.length;
    System.out.println("");
    for(int n=0;n<m;n++)
    {
      System.out.print(b[n]);
    }
```

```
        String n3;
      n3="";
      String
n1=n3.valueOf(b[0])+n3.valueOf(b[1])+n3.valueOf(b[2])+n3.valueOf(b[3]);
      int n2=Integer.parseInt(n1);
            Date date=new Date();
            String str=date.toString();
            String time2=str.substring(11,19);
            String date2=str.substring(4,11)+str.substring(24,28);
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
               Connection con;
            con=null;
            con=DriverManager.getConnection("jdbc:odbc:iaqodbc");
               Statement sta;
            sta = con.createStatement();
               String str1="INSERT INTO HistoryTb"+ "
 roomID,date1,time1,gas,value1)"+" VALUES
'r12','"+date2+"','"+time2+"','Butane',"+n2+")";
             System.out.println(str1);
             sta.executeUpdate(str1);
                sta.close();
          }
       }
       catch(Exception e)
       {
       System.out.println(e);
       }
```

```
    }
  public thread1()
  {
    this.start();
  }
  protected void finalize() throws Throwable {
  }
  public void start()
  {
    Thread t= new Thread(this);
    t.start();
  }
}
public class Main {
  public Main() {
  }
  public static void main(String[] args) {
    // TODO code application logic here
    new thread1();
    java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
        new NewJFrame().setVisible(true);
      }
    });
  }
}
package javaapplication11;
```

```
import java.awt.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
public class NewClass extends JFrame{
      public NewClass() {
      Vector columnNames = new Vector();
      Vector data = new Vector();
 Connection connection;
      try
      {

 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          connection=DriverManager.getConnection("jdbc:odbc:iaqodbc");
          String sql = "Select * from ThresholdTb";
          Statement stmt = connection.createStatement();
          ResultSet rs = stmt.executeQuery( sql );
          ResultSetMetaData md = rs.getMetaData();
          int columns = md.getColumnCount();
           for (int i = 1; i <= columns; i++)
          {
             columnNames.addElement( md.getColumnName(i) );
          }
           while (rs.next())
          {
```

```
            Vector row = new Vector(columns);
             for (int i = 1; i <= columns; i++)
             {
                row.addElement( rs.getObject(i) );
             }
              data.addElement( row );
          }
           rs.close();
          stmt.close();
       }
       catch(Exception e)
       {
          System.out.println( e );
       }
        JTable table = new JTable(data, columnNames);
        JScrollPane scrollPane = new JScrollPane( table );
       getContentPane().add( scrollPane );
        JPanel buttonPanel = new JPanel();
       getContentPane().add( buttonPanel, BorderLayout.SOUTH );
    }
}
package javaapplication11;
import java.awt.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.swing.*;
```

```
import javax.swing.table.*;
public class NewClass1 extends JFrame {
   public NewClass1() {
      Vector columnNames = new Vector();
      Vector data = new Vector();
 Connection connection;
      try
      {
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        connection=DriverManager.getConnection("jdbc:odbc:iaqodbc");
        String sql = "Select * from CorrectiveTb";
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery( sql );
        ResultSetMetaData md = rs.getMetaData();
        int columns = md.getColumnCount();
         for (int i = 1; i <= columns; i++)
        {
           columnNames.addElement( md.getColumnName(i) );
        }
         while (rs.next())
        {
           Vector row = new Vector(columns);
            for (int i = 1; i <= columns; i++)
           {
              row.addElement( rs.getObject(i) );
           }
           data.addElement( row );
```

```
        }
         rs.close();
        stmt.close();
      }
      catch(Exception e)
      {
        System.out.println( e );
      }
       JTable table = new JTable(data, columnNames);
       JScrollPane scrollPane = new JScrollPane( table );
      getContentPane().add( scrollPane );
       JPanel buttonPanel = new JPanel();
      getContentPane().add( buttonPanel, BorderLayout.SOUTH );}}
```

```
package javaapplication11;
import java.awt.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
public class NewClass2 extends JFrame{
      public NewClass2() {
      Vector columnNames = new Vector();
      Vector data = new Vector();
```

```
Connection connection;
    try
    {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        connection=DriverManager.getConnection("jdbc:odbc:iaqodbc");
        String sql = "Select * from RoomTb";
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery( sql );
        ResultSetMetaData md = rs.getMetaData();
        int columns = md.getColumnCount();
         for (int i = 1; i <= columns; i++)
        {
           columnNames.addElement( md.getColumnName(i) );
        }
         while (rs.next())
        {
           Vector row = new Vector(columns);

           for (int i = 1; i <= columns; i++)
           {
              row.addElement( rs.getObject(i) );
           }
            data.addElement( row );
        }
         rs.close();
        stmt.close();
      }
```

```
        catch(Exception e)
        {
           System.out.println( e );
        }
        JTable table = new JTable(data, columnNames);
         JScrollPane scrollPane = new JScrollPane( table );
        getContentPane().add( scrollPane );
         JPanel buttonPanel = new JPanel();
        getContentPane().add( buttonPanel, BorderLayout.SOUTH );
    }
}
package javaapplication11;
import java.awt.BorderLayout;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.util.Vector;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
public class NewClass4 extends JFrame{
    public NewClass4() {
        Vector columnNames = new Vector();
        Vector data = new Vector();
```

```
Connection connection;
    try
    {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        connection=DriverManager.getConnection("jdbc:odbc:iaqodbc");
        String sql = "Select * from HistoryTb";
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery( sql );
        ResultSetMetaData md = rs.getMetaData();
        int columns = md.getColumnCount();
        for (int i = 1; i <= columns; i++)
        {
           columnNames.addElement( md.getColumnName(i) );
        }
         while (rs.next())
        {
           Vector row = new Vector(columns);

           for (int i = 1; i <= columns; i++)
           {
            row.addElement( rs.getObject(i) );
           }
            data.addElement( row );
        }
         rs.close();
        stmt.close();
```

```
        }
        catch(Exception e)
        {
           System.out.println( e );
        }
         JTable table = new JTable(data, columnNames);
         JScrollPane scrollPane = new JScrollPane( table );
        getContentPane().add( scrollPane );
         JPanel buttonPanel = new JPanel();
        getContentPane().add( buttonPanel, BorderLayout.SOUTH );
           }
    }
package javaapplication11;
import java.awt.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
public class NewClass5 extends JFrame{
    public NewClass5() {
        System.out.println("start");
        Vector columnNames = new Vector();
        Vector data = new Vector();
 Connection connection;
        try
        {
```

```
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  connection=DriverManager.getConnection("jdbc:odbc:iaqodbc");

  String sql = "Select * from ReportsTb";
  Statement stmt = connection.createStatement();
  ResultSet rs = stmt.executeQuery( sql );
  ResultSetMetaData md = rs.getMetaData();
  int columns = md.getColumnCount();
   for (int i = 1; i <= columns; i++)
  {
     columnNames.addElement( md.getColumnName(i) );
  }
   while (rs.next())
  {
     Vector row = new Vector(columns);
      for (int i = 1; i <= columns; i++)
     {
        row.addElement( rs.getObject(i) );
     }
      data.addElement( row );
  }
   rs.close();
  stmt.close();
}
catch(Exception e)
{
  System.out.println( e );
```

```
        }
        JTable table = new JTable(data, columnNames);
        JScrollPane scrollPane = new JScrollPane( table );
        getContentPane().add( scrollPane );
        JPanel buttonPanel = new JPanel();
        getContentPane().add( buttonPanel, BorderLayout.SOUTH );
        }
}
package javaapplication11;
import java.awt.Graphics;
import javax.swing.*;
 import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
public class LinePanel extends JPanel{
    public static void createAndShowGUI() {
        JFrame frame = new JFrame("Graph");
        frame.setContentPane(new LinePanel());
        frame.setSize(500,500);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
```

```
g.setColor(java.awt.Color.black);
g.drawLine(35, 25, 35, 450);
g.drawLine(35,450,475,450);
g.drawLine(32,25,38,25);
g.drawLine(32,45,38,45);
g.drawLine(32,65,38,65);
g.drawLine(32,85,38,85);
g.drawLine(32,105,38,105);
g.drawLine(32,125,38,125);
g.drawLine(32,145,38,145);
g.drawLine(32,165,38,165);
g.drawLine(32,185,38,185);
g.drawLine(32,205,38,205);
g.drawLine(32,225,38,225);
g.drawLine(32,245,38,245);
g.drawLine(32,265,38,265);
g.drawLine(32,285,38,285);
g.drawLine(32,305,38,305);
g.drawLine(32,325,38,325);
g.drawLine(32,345,38,345);
g.drawLine(32,365,38,365);
g.drawLine(32,385,38,385);
g.drawLine(32,405,38,405);
g.drawLine(32,425,38,425);
g.drawLine(32,445,38,445);
g.drawString("1700",0,25);
g.drawString("1620",0,45);
```

```
g.drawString("1540",0,65);
g.drawString("1460",0,85);
g.drawString("1380",0,105);
g.drawString("1300",0,125);
g.drawString("1220",0,145);
g.drawString("1140",0,165);
g.drawString("1060",0,185);
g.drawString("980",0,205);
g.drawString("900",0,225);
g.drawString("820",0,245);
g.drawString("740",0,265);
g.drawString("660",0,285);
g.drawString("580",0,305);
g.drawString("500",0,325);
g.drawString("420",0,345);
g.drawString("340",0,365);
g.drawString("260",0,385);
g.drawString("180",0,405);
g.drawString("100",0,425);
g.drawString("20",0,445);
g.drawString("CO2",62,465);
g.drawString("CO",137,465);
g.drawString("NH3",213,465);
g.drawString("Pb",288,465);
g.drawString("Butane",363,465);
g.drawString("Nicotine",438,465);
String str;
```

```
Connection connection;
  Statement statement;
  ResultSet resultSet;
  int y=100;
int ht=450-y;
try
{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

  connection=DriverManager.getConnection("jdbc:odbc:iaqodbc");
PreparedStatement pr=connection.prepareStatement("select * from
ReportsTb");
  resultSet = pr.executeQuery();
    if(resultSet!=null)
      {
      while(resultSet.next())
      {
        if("r12".equals(resultSet.getString("RoomID")))
        {
          //System.out.println(resultSet.getString("Butane"));
          str=resultSet.getString("CO2");
          if(!str.equals("-")&&!str.equals(""))
          {
          int g1=Integer.parseInt(str);
          g1=g1/4;
          System.out.println(g1);
          g.fillRect(62,450-g1,20,g1);
```

```
}
 str=resultSet.getString("CO");
if(!str.equals("-")&&!str.equals(""))
{
int g2=Integer.parseInt(str);
g2=g2/4;
System.out.println(g2);
g.fillRect(137,450-g2,20,g2);
}
 str=resultSet.getString("NH3");
if(!str.equals("-")&&!str.equals(""))
{
int g3=Integer.parseInt(str);
g3=g3/4;
System.out.println(g3);
g.fillRect(213,450-g3,20,g3);
}
 str=resultSet.getString("Pb");
if(!str.equals("-")&&!str.equals(""))
{
int g4=Integer.parseInt(str);
 g4=g4/4;
System.out.println(g4);
g.fillRect(288,450-g4,20,g4);
}
str=resultSet.getString("Butane");
if(!str.equals("-")&&!str.equals(""))
```

```
                    {
                    int g5=Integer.parseInt(str);
                    g5=g5/4;
                    System.out.println(g5);
                    g.fillRect(363,450-g5,20,g5);
                    }
                    str=resultSet.getString("Nicotine");
                    if(!str.equals("-")&&!str.equals(""))
                    {
                    int g6=Integer.parseInt(str);
                    g6=g6/4;
                    System.out.println(g6);
                    g.fillRect(438,450-g6,20,g6);
                }
            }
          }
          }
         }
    catch(Exception e)
    {
      System.out.println(e);
    }
  }
}
```

**CLIENT SIDE:**

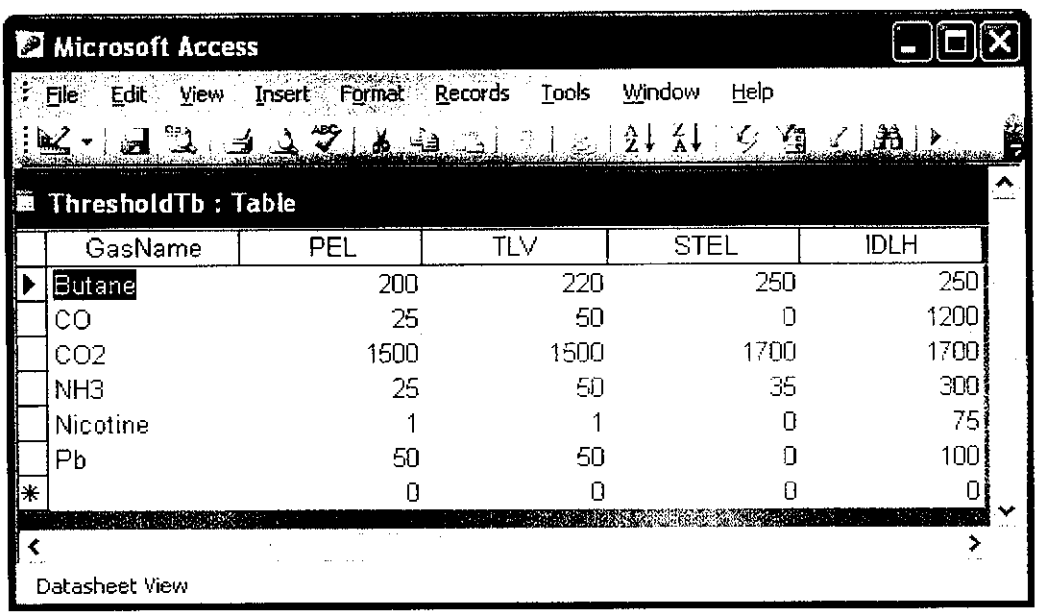
```
Dim i As Integer
Dim j As Integer
Dim s As String
Dim s1, s2 As String
Dim m, k As Integer

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA"
(ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long,
lParam As Any) As Long

Private Sub Form_Load()
If Form1.Winsock1.State = sckClosed Then ' if the socket is closed
   Form1.Winsock1.RemoteHost = "localhost"
   Form1.Winsock1.RemotePort = 1237
   Form1.Winsock1.Connect ' start connection attempt
 Else ' if the socket is open
   'Form1.Winsock1.Close ' close it
 End If
Label1.Caption = "System Time"
Label2.Caption = Time$
End Sub

Private Sub Timer1_Timer()
i = i + 1
j = j + 5
```

```
If (i = 2) Then
   i = 0
   k = SendMessage(393810, &HE, 0&, 0&)
   s = Space$(i + 1)
   Call SendMessage(393910, &HD, ByVal i + 1, ByVal s)
   s = Right(s, 20)
   For m = 1 To 20
      If (StrComp("=", Mid(s, m, 1))) Then
         s2 = Mid(s, m, 3)
   Next m
   k = Asc(Mid(s2, 1, 1)) + Asc(Mid(s2, 2, 1)) + Asc(Mid(s2, 3, 1))
   If Form1.Winsock1.State = sckConnected Then ' if there is a connection
        Form1.Winsock1.SendData s2  ' send data to the other side
        Label3.Caption = "sent"
   End If
End If
End Sub
```

# APPENDIX- B

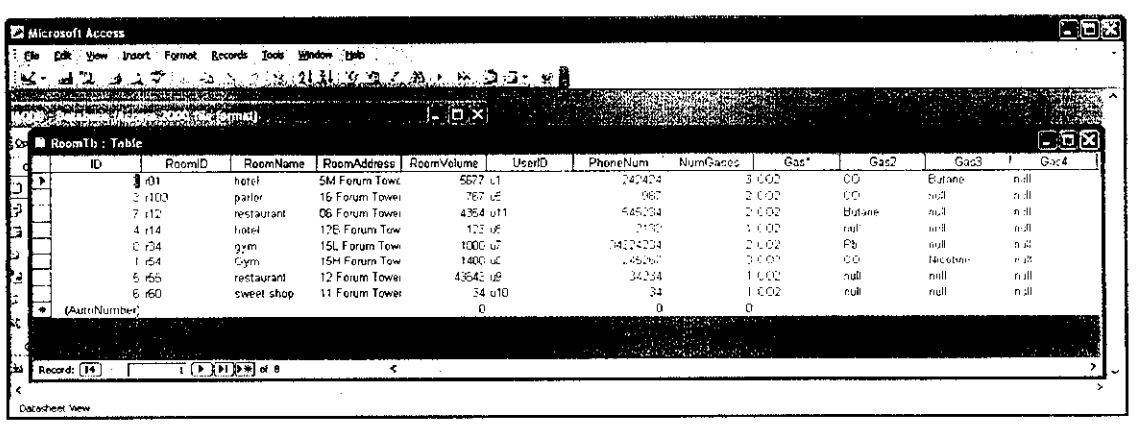
## SCREEN SHOTS

## DATABASES



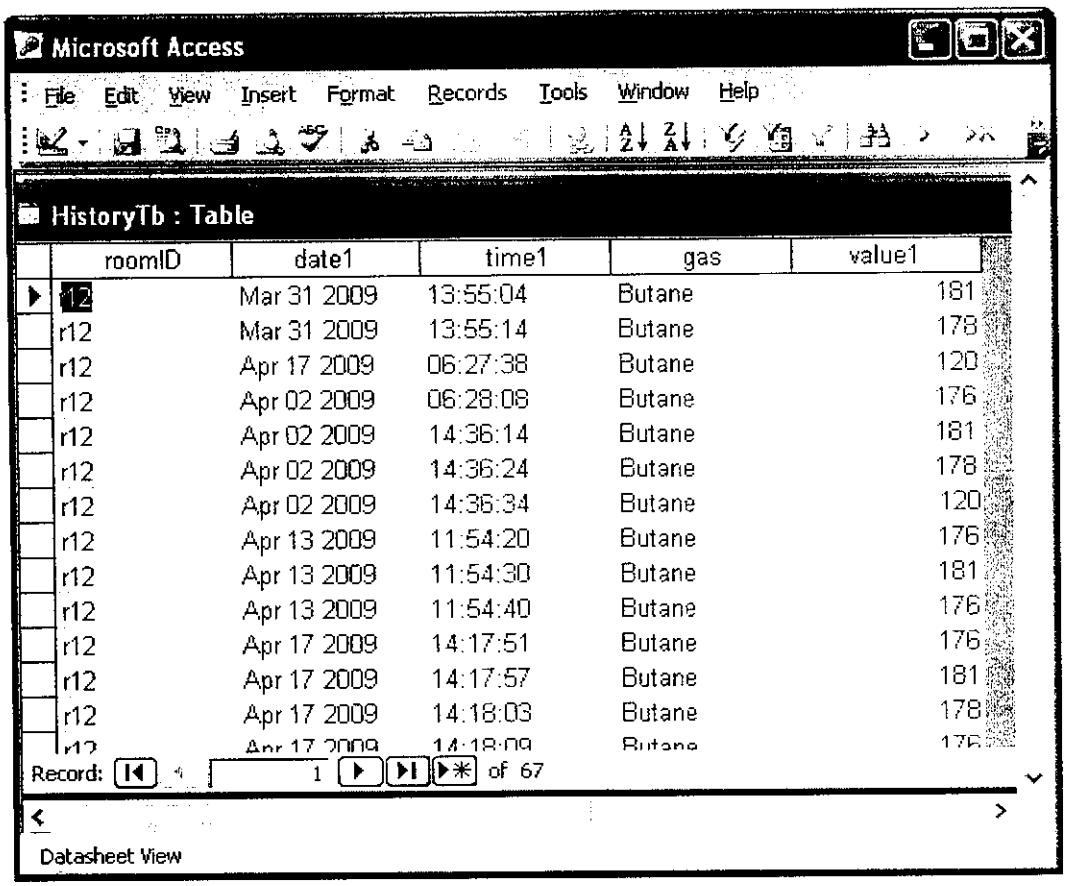

| GasName | PEL | TLV | STEL | IDLH |
|---|---|---|---|---|
| Butane | 200 | 220 | 250 | 250 |
| CO | 25 | 50 | 0 | 1200 |
| CO2 | 1500 | 1500 | 1700 | 1700 |
| NH3 | 25 | 50 | 35 | 300 |
| Nicotine | 1 | 1 | 0 | 75 |
| Pb | 50 | 50 | 0 | 100 |
| | 0 | 0 | 0 | 0 |

**Threshold Table**

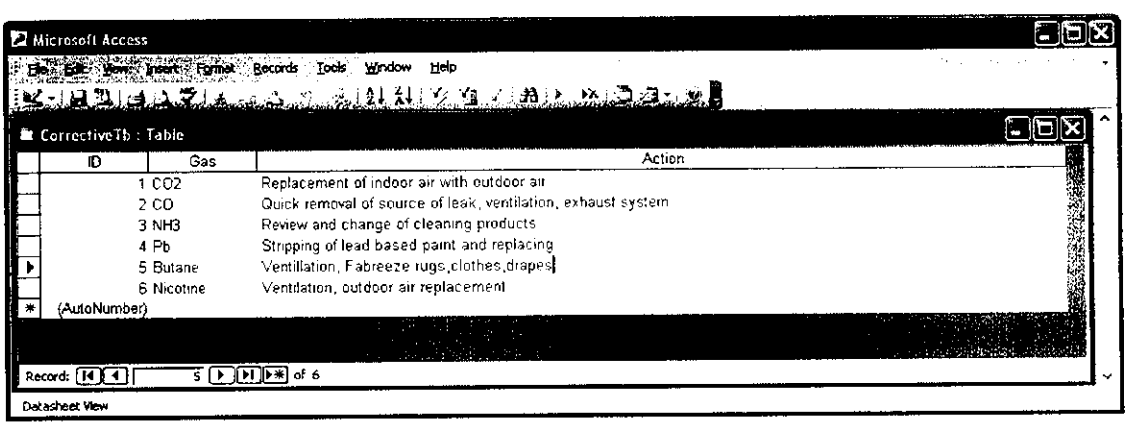



| ID | RoomID | RoomName | RoomAddress | RoomVolume | UserID | PhoneNum | NumGases | Gas1 | Gas2 | Gas3 | Gas4 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | r01 | hotel | 5M Forum Towe | 5677 | u1 | 242424 | 3 | CO2 | CO | Butane | null |
| 3 | r103 | parlor | 16 Forum Tower | 767 | u5 | 987 | 2 | CO2 | CO | null | null |
| 7 | r12 | restaurant | 06 Forum Tower | 4354 | u11 | 5452234 | 2 | CO2 | Butane | null | null |
| 4 | r14 | hotel | 12B Forum Tow | 123 | u6 | 2132 | 1 | CO2 | null | null | null |
| 8 | r34 | gym | 15L Forum Tow | 1000 | u7 | 34234234 | 2 | CO2 | Pb | null | null |
| 1 | r54 | Gym | 15H Forum Tow | 1400 | u5 | 345267 | 3 | CO2 | CO | Nicotine | null |
| 5 | r55 | restaurant | 12 Forum Tower | 43643 | u9 | 34234 | 1 | CO2 | null | null | null |
| 6 | r60 | sweet shop | 11 Forum Tower | 34 | u10 | 34 | 1 | CO2 | null | null | null |
| (AutoNumber) | | | | 0 | | 0 | 0 | | | | |

**Room Details Table**

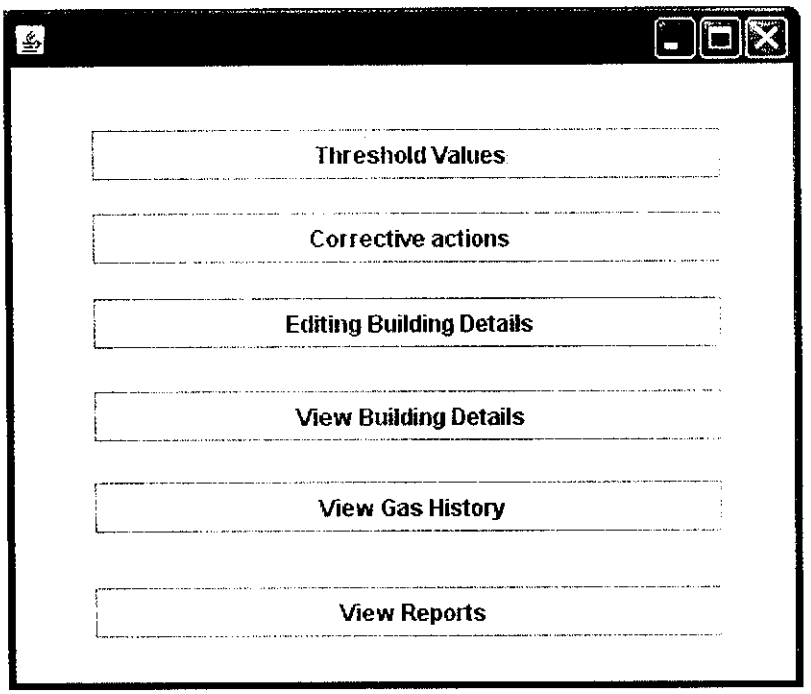



**Gas History Table**

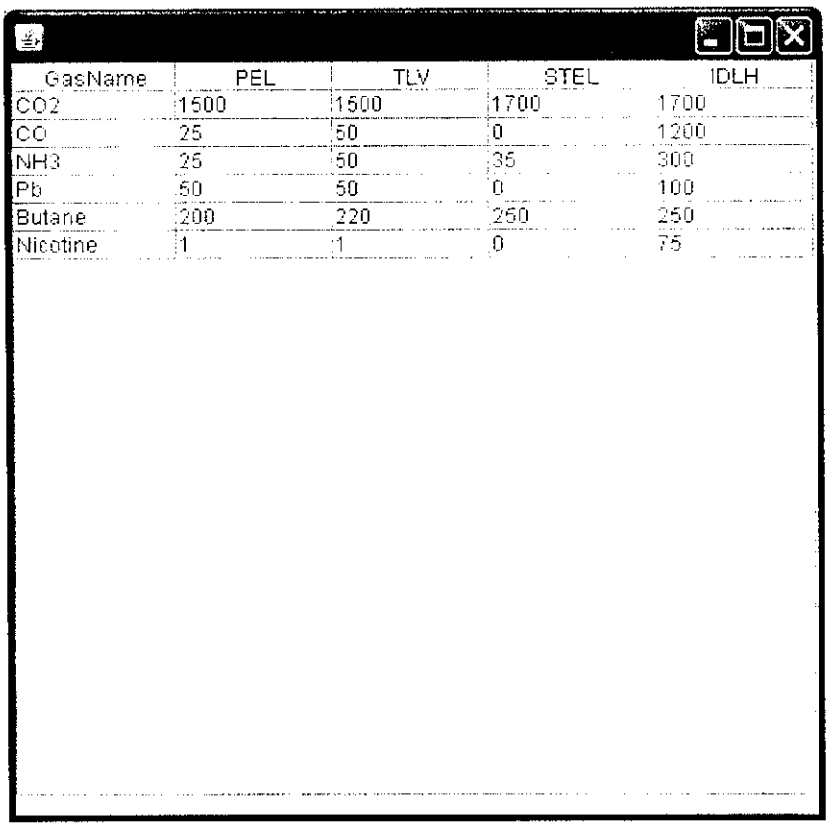



**Corrective Actions Table**

## Server Side

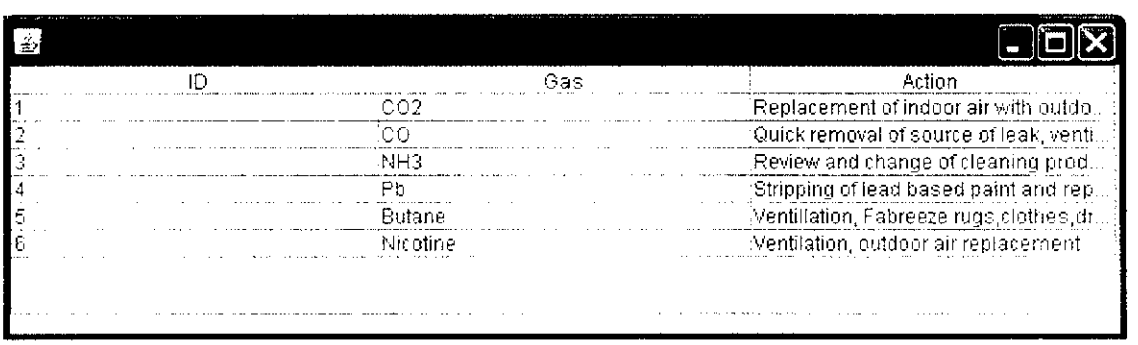



**Main Menu**



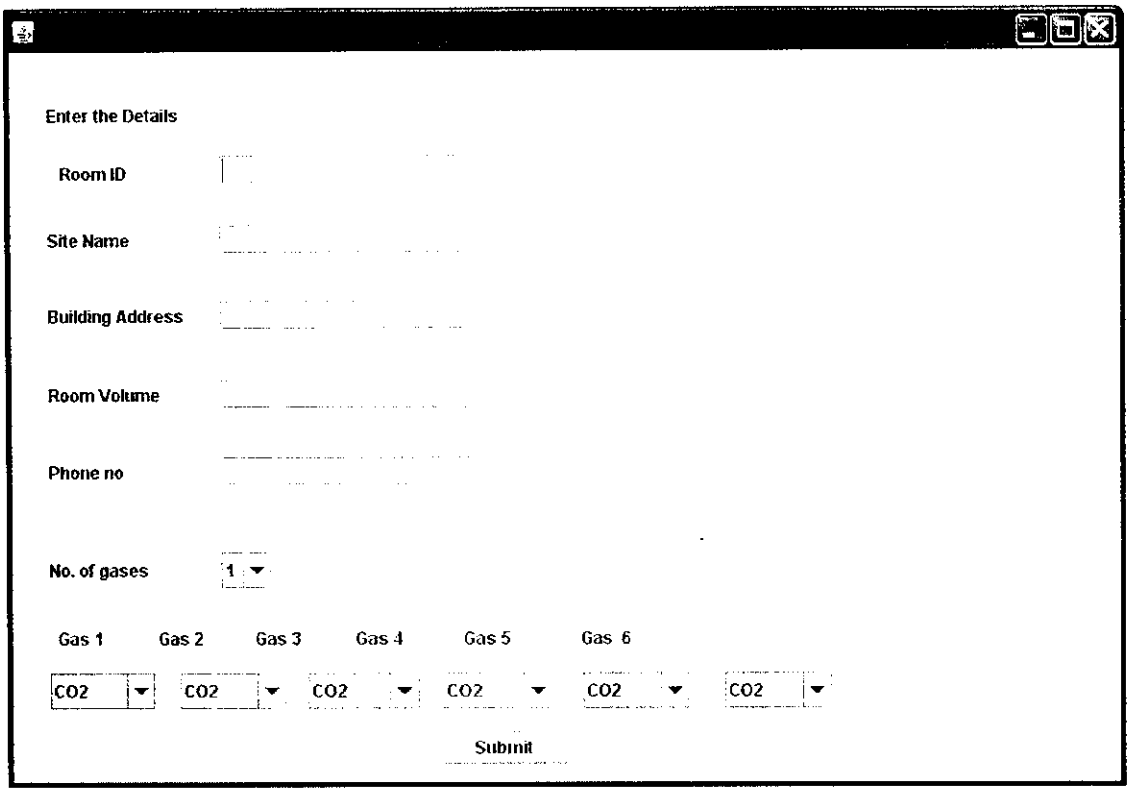
| GasName | PEL | TLV | STEL | IDLH |
|---|---|---|---|---|
| CO2 | 1500 | 1500 | 1700 | 1700 |
| CO | 25 | 50 | 0 | 1200 |
| NH3 | 25 | 50 | 35 | 300 |
| Pb | 50 | 50 | 0 | 100 |
| Butane | 200 | 220 | 250 | 250 |
| Nicotine | 1 | 1 | 0 | 75 |

**View Threshold Details**

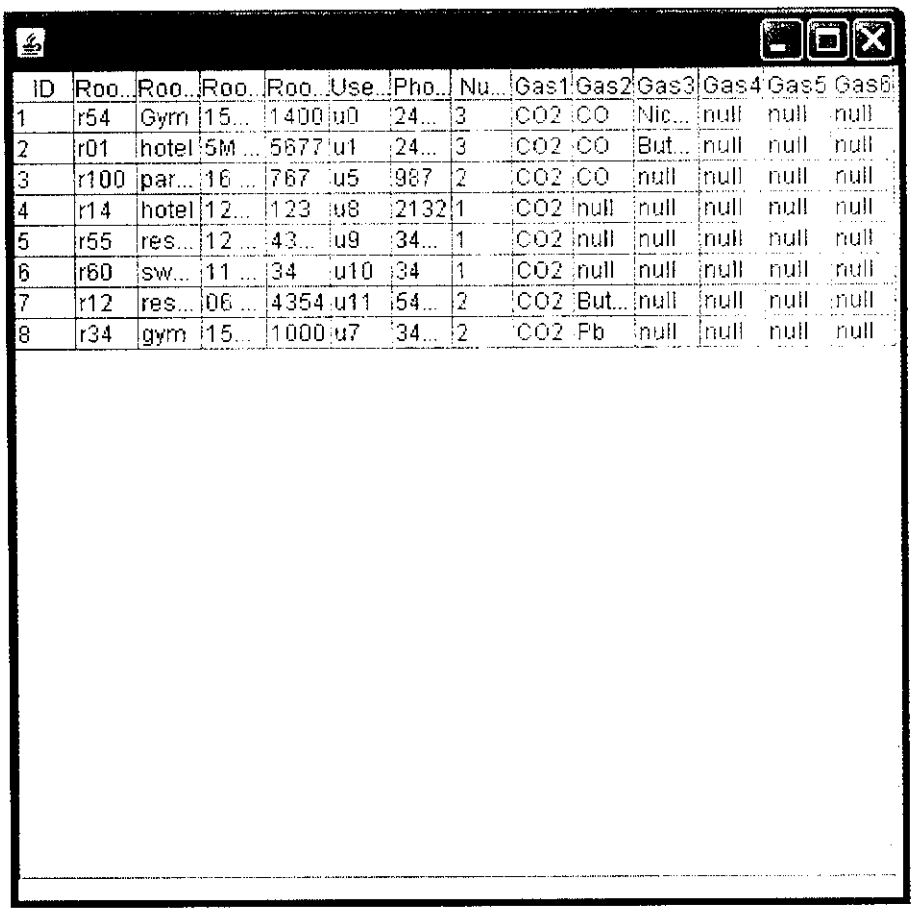


| ID | Gas | Action |
|---|---|---|
| 1 | CO2 | Replacement of indoor air with outdo... |
| 2 | CO | Quick removal of source of leak, venti... |
| 3 | NH3 | Review and change of cleaning prod... |
| 4 | Pb | Stripping of lead based paint and rep... |
| 5 | Butane | Ventillation, Fabreeze rugs,clothes,dr... |
| 6 | Nicotine | Ventilation, outdoor air replacement |

**View Corrective Actions**

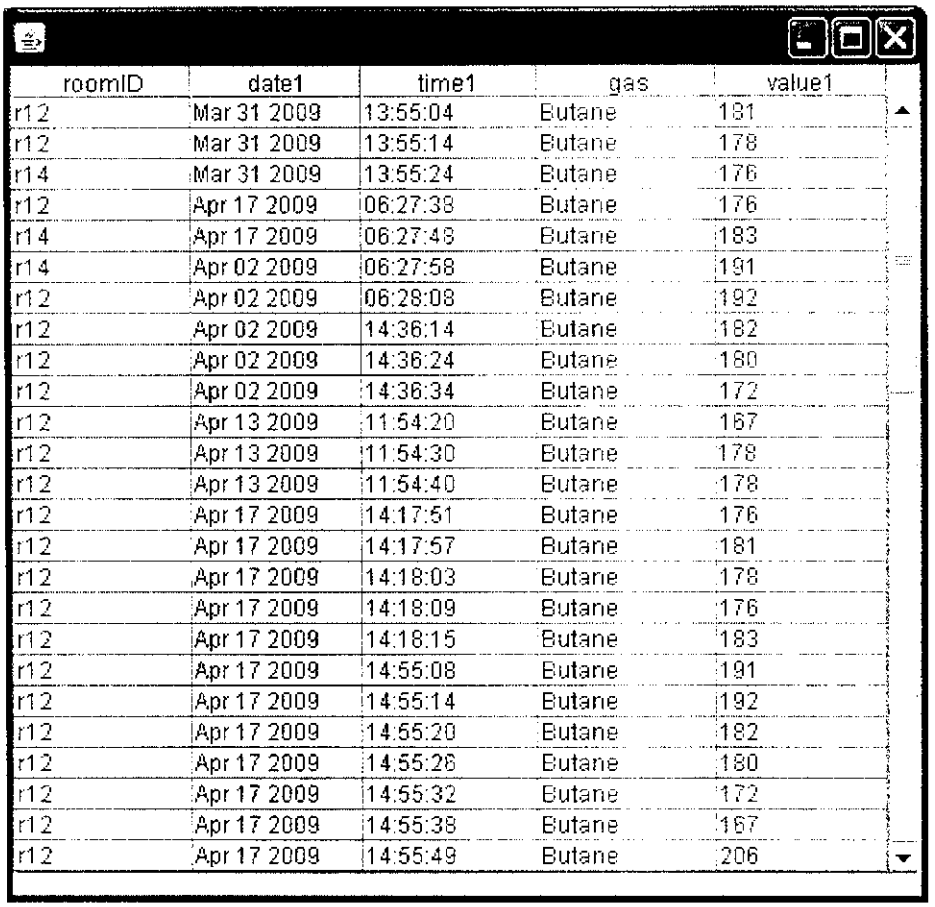



**Edit Room Details**



| ID | Roo... | Roo... | Roo... | Roo... | Use... | Pho... | Nu... | Gas1 | Gas2 | Gas3 | Gas4 | Gas5 | Gas6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | r54 | Gym | 15... | 1400 | u0 | 24... | 3 | CO2 | CO | Nic... | null | null | null |
| 2 | r01 | hotel | 5M ... | 5677 | u1 | 24... | 3 | CO2 | CO | But... | null | null | null |
| 3 | r100 | par... | 16 ... | 767 | u5 | 987 | 2 | CO2 | CO | null | null | null | null |
| 4 | r14 | hotel | 12... | 123 | u8 | 2132 | 1 | CO2 | null | null | null | null | null |
| 5 | r55 | res... | 12 ... | 43... | u9 | 34... | 1 | CO2 | null | null | null | null | null |
| 6 | r60 | sw... | 11 ... | 34 | u10 | 34 | 1 | CO2 | null | null | null | null | null |
| 7 | r12 | res... | 06 ... | 4354 | u11 | 54... | 2 | CO2 | But... | null | null | null | null |
| 8 | r34 | gym | 15... | 1000 | u7 | 34... | 2 | CO2 | Pb | null | null | null | null |

**View Room Details**

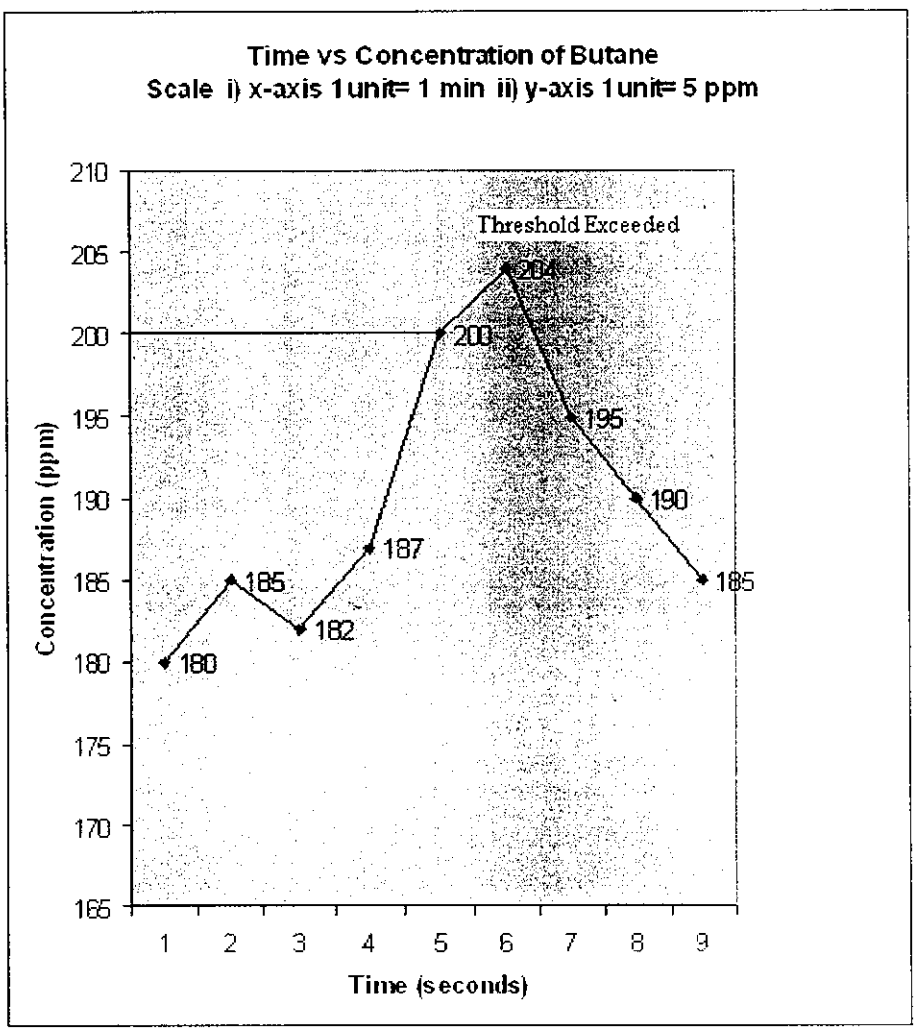


| roomID | date1 | time1 | gas | value1 |
|---|---|---|---|---|
| r12 | Mar 31 2009 | 13:55:04 | Butane | 181 |
| r12 | Mar 31 2009 | 13:55:14 | Butane | 178 |
| r14 | Mar 31 2009 | 13:55:24 | Butane | 176 |
| r12 | Apr 17 2009 | 06:27:38 | Butane | 176 |
| r14 | Apr 17 2009 | 06:27:48 | Butane | 183 |
| r14 | Apr 02 2009 | 06:27:58 | Butane | 191 |
| r12 | Apr 02 2009 | 06:28:08 | Butane | 192 |
| r12 | Apr 02 2009 | 14:36:14 | Butane | 182 |
| r12 | Apr 02 2009 | 14:36:24 | Butane | 180 |
| r12 | Apr 02 2009 | 14:36:34 | Butane | 172 |
| r12 | Apr 13 2009 | 11:54:20 | Butane | 167 |
| r12 | Apr 13 2009 | 11:54:30 | Butane | 178 |
| r12 | Apr 13 2009 | 11:54:40 | Butane | 178 |
| r12 | Apr 17 2009 | 14:17:51 | Butane | 176 |
| r12 | Apr 17 2009 | 14:17:57 | Butane | 181 |
| r12 | Apr 17 2009 | 14:18:03 | Butane | 178 |
| r12 | Apr 17 2009 | 14:18:09 | Butane | 176 |
| r12 | Apr 17 2009 | 14:18:15 | Butane | 183 |
| r12 | Apr 17 2009 | 14:55:08 | Butane | 191 |
| r12 | Apr 17 2009 | 14:55:14 | Butane | 192 |
| r12 | Apr 17 2009 | 14:55:20 | Butane | 182 |
| r12 | Apr 17 2009 | 14:55:26 | Butane | 180 |
| r12 | Apr 17 2009 | 14:55:32 | Butane | 172 |
| r12 | Apr 17 2009 | 14:55:38 | Butane | 167 |
| r12 | Apr 17 2009 | 14:55:49 | Butane | 206 |

**View Gas History**

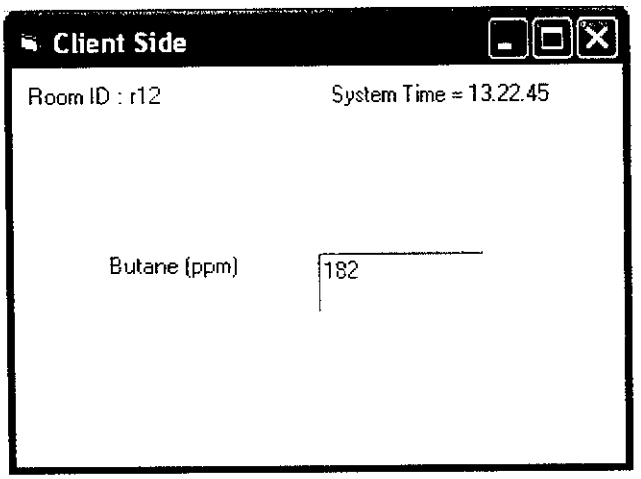
| RoomID | RoomA... | Date1 | CO2 | CO | NH3 | Pb | Butane | Nicotine |
|---|---|---|---|---|---|---|---|---|
| r54 | 15H F... | Apr 17 ... | | | - | - | - | |
| r01 | 5M For... | Apr 17 ... | | | - | - | | - |
| r100 | 16 For... | Apr 17 ... | | | - | - | - | - |
| r14 | 12B F... | Apr 17 ... | | - | - | - | - | - |
| r55 | 12 For... | Apr 17 ... | | - | - | - | - | - |
| r60 | 11 For... | Apr 17 ... | | - | - | - | - | - |
| r12 | 06 For... | Apr 17 ... | | - | - | - | 182 | - |
| r34 | 15L Fo... | Apr 17 ... | | - | - | - | - | - |

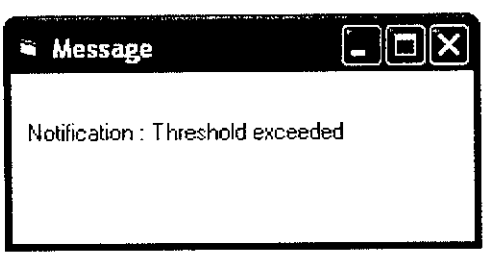
**Report- Data**




**Report- Line Graph**

## Client Side


Client Side
Room ID : r12
System Time = 13.22.45
Butane (ppm)
182


Message
Notification : Threshold exceeded

# REFERNCES

## 7. REFERENCES


1. Wan Rong, Kong Dequan, Xian University, "Analysis on Influencing Factors of Indoor Air Quality And Measures of Improvement on Modern buildings", The 2nd International Conference on Bioinformatics and Biomedical Engineering April 2008.

2. Ken Stevenson, "Air Quality Monitoring", 5th Meeting of National Air Quality Reference Laboratories, AQUILA, October 2004.

3. G.B. Shoom, "IAQES: An Expert System for Indoor Air Quality Problem Resolution", CompEngServ Conference, Ontario Canada.

4. Y. Zhang , "Indoor air quality engineering", CRC Press, Boca Raton, FL, 2005

5. Raj Shah, "Embedded Design Using PIC 16F877" , AMS Publishing, March 2004.

6. Herbert Schildt, "Java :The Complete Reference Book", Tata Mcgraw Hill Publishing Company Limited, Fifth Edition.

7. http://www.epa.gov/iaq as on Dec 23, 2008.

8. http://www.AmericanLungAssociation.com as on Dec 23,2008.

9. http://www.osha.gov/index.html as on Dec 25,2008.

10.http://www.canarina.com/indoorairpollution.html as on Jan 17,2009.

11.http://www.ecologicsystems.com as on Jan 20 2009

12.http://www.microchip.com as on Feb 10 2009.

13.http://www.dataarchive.com/PIC16F877.html as on Feb 17 2009

14.http://www.ccsinfo.com as on Feb 20 2009

15.http://www.alldatasheet.com as on Feb 25 2009