

AIR POLLUTION MONITORING SYSTEM USING IOT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering
by

ANUMASU SATHVIK REDDY, GUDAVALLI CH K
SASIKIRAN (38110039, 38110174)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH – 2020

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my sincere thanks to **Dr. T. Sasikala, M.E., Ph.D.**, Dean, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D.**, and **Dr .L. Lakshmanan, M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.L.Lakshmanan, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ANUMASU SATHVIK REDDY, GUDAVALLI CH K SASIKIRAN (38110039, 38110174)** who carried out the project entitled “**AIR POLLUTION MONITORING SYSTEM USING IOT**” under the supervision from November 2019 to April 2020.

Internal Guide

Dr.L.Lakshmanan M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **ANANYA JAYARAM ORUGANTI** hereby declare that the Project Report entitled **“AIR POLLUTION MONITORING SYSTEM USING IOT”** done by me under the guidance of **DR. R. AROUL CANESSANE, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ABSTRACT

Air quality is a global provocation for governments, industries and citizens. Many governments are spending huge amount in improving air quality and they are educating people to improving air quality. Now a days, many industries and daily activities generate huge air pollutions, especially in metropolitan cities of the developing countries, such as china and India. In these cities, air quality of the urban environment has seriously degraded people"s work and life. Delhi, capital city of India, has to release fed alarm of air pollution and stop nearly all of the public outdoor activities several times per year. World Health Organization (WHO) reveals the fact that only in 2016, 575 thousand people died if air pollution, which contributes t0 12.5% of all the death, indicating that the air pollution is now the largest single environmental health risk. To eradicate this problem we need to sort out In order to implement effective policies and interventions there is an increasing focus on understanding the levels and causes of air pollution. Today, air quality monitoring is performed by large, expensive scientific instruments permanently installed and professionally maintained, at a relatively small number of fixed locations. For example London has around 100 monitoring stations. This makes it difficult for citizens to understand the levels of pollution they experience in their daily lives, as the monitoring data is not available in real time and is very sparse. Advances in sensors, IoT platforms and mobile communications technologies have led to the emergence of smaller, portable, low cost, sensors that can measure and report air quality in near real time.

TABLE OF CONTENTS

CHAPTER No.	CHAPTER NAME	PAGE No.
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF TABLES	x
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	01
	1.1 Arduino Uno	02
	1.2 Software and Programming tool	02
	1.3 Program Structure	03
	1.4 Power	03
	1.5 Memory	04
	1.6 Input and Output	04
	1.7 IDE	07
2	LITERATURE SURVEY	08
	Automated Control System	08
	Smart Vehicle Air Quality	09
	Management	
	Control Scheme for Smart	09
	Community	
	Home Mechanical Ventilation	10
	Micro-Scale Air Quality	10
	Management	
	Data Transferring Model for Urban	10
	Areas	
	Air Scope	11
	Parallel-Non Linear Optimization	11
	Grid Computing Exploitation	12
	Model for Air Quality	12
	Management and Assessment	
3	METHODOLOGY	13
	3.1 Block Diagram	13
	3.2 Receiver Side	13
	3.3 Working	13
	3.4 Requirements	14

	Hardware Requirements	14
	Software Requirements	14
	Pin Description	16
	Zigbee Transceiver	18
	Features	19
	Language Description	20
	Java	20
	Importance of Java to Internet	20
	Applications and Applets	
	Java Architecture	20
	Compilation of Code	20
	CSS	21
	Servlets	22
	Servlet Architecture	22
	Major Tasks of Servlets	23
	Apache Tomcat Server	23
	RMI	23
	Java Networking	23
	Socket Programming	24
	Uploading Data to Server	25
	Shelf life of Uploading	26
	files	26
	Request Methods	26
		26
	RESULTS AND DISCUSSIONS	
4	Screenshots	27
	Login Form	27
	Final Output	27
	Performance Analysis	28
	CONCLUSION AND FUTURE WORK	30
5	Conclusion	31
	Future Work	31
	REFERENCES	31
	APPENDIX	32
		33

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
1.1	Arduino Uno Compatible Board	3
1.2	Zigbee	6
3.1	Arduino Uno Block Diagram	13
3.2	Web Page Receiver Side	13
3.3	MQ135 Air Quality Sensor	14
3.4	Zigbee Sensor Value	17
3.5	Zigbee Transceiver	18
3.6	Zigbee Adaptor	18
3.7	Code Compilation	20
3.8	Servlet Architecture	22
3.9	CORBA Implementation	24
4.1	Active Arduino Board	27
4.2	Login Form	27
4.3	Arduino Uno	28
4.4	MQ2 Gas Sensor	28
4.5	Values of Gasses in Room	29
4.6	Values of Gasses in Office	29

LIST OF TABLES

TABLE No.	TABLE NAME	PAGE No.
3.1	Pin Description	15
3.2	Technical Specifications	16
4.1	PPM Values of Gasses	29

LIST OF ABBREVIATIONS

USB	Universal Serial Bus
V _{in}	Input Voltage
<u>Wi-Fi</u>	Wireless Fidelity
YIQ	Luminance In-Phase Quadrature
3-D	Three Dimension
3.3v	3.3 Volts
LCD	Liquid Crystal Display
HTML	Hyper Text Mark-up Language
IDE	Integrated Development Environment
I/O	Input/output

CHAPTER 1

INTRODUCTION

Air pollution can have a lasting effect on productivity in other ways, such as by stunting plant growth which reduces agricultural productivity. It can also make cities less attractive to talented workers, thereby reducing a cities competitiveness. For example, some Indian cities are reporting a reverse migration trend¹⁶ from the city back to the country as citizens take steps to avoid high pollution. Governments also face fines for non-compliance to regional air pollution legislation. Many countries across Europe including the UK, Germany, France, Italy and Spain face the prospect of huge fines arising from persistent failures to comply with European air pollution laws¹⁷. The UK has been threatened with a \$400M fine and cities including London, Berlin, Lyon and Barcelona have all been highlighted by regulators for their unacceptably high air pollution levels. The United States has legislated to control air pollution, with California demonstrating how effective regulatory approaches can be, with every dollar invested in air pollution control since 1970 reported to yield \$30 in benefits²⁰, giving an impressive \$1.5 trillion return for a \$65 billion investment.

Internet of things was first introduced in 1999 at autoID centre and first used by Kevin ashton. As evolving this latest burning technology, it promises to connect all our surrounding things to a network and communicating with each other with less human involvement. Still internet of things is in beginning stage and there is no common architecture exists till today. There is lot of researches and implementations are currently being going on in all the respective areas. Thus there is no guidelines or boundaries exists to define the definition of internet of things. So depending on the context, application the internet of things has different definitions. Shortly it is defined as the things present in the physical world or in an environment are attached with sensors or with any embedded systems and made connected to network via wired or wireless connections. And it consists of smart machines which communicating interacting with other machines, environment, objects etc. And also it incorporates to connect any two machines, machine to human and vice versa etc. this communication is called as M-M communication. As M-M communication is developing by the various standardization bodies such as Open Mobile Alliance (OMA), European Telecommunication Standards Institute (ETSI), Institute of Electrical and Electronic Engineers (IEEE), 3rd Generation Partnership Project (3GPP) organization have performed some activities on M-M communication . It makes daily life things to equip with transceivers, sensors, actuators and microcontrollers etc. for communication. Some important benefits of internet of things includes 1) tracking behaviour; 2) enhanced situational awareness; 3) sensor driven decision analytics; 4) instantaneous control and

response. Etc. IOT technology grows in various fields of smart applications but we have not yet found boundary constraints of this technology.

ARDUINO UNO:

Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License, permitting the manufacture of Arduino boards and software distribution by anyone.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (SHIELDS) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project. to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

SOFTWARE AND PROGRAMMING TOOLS

A program for Arduino may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

A program written with the IDE for Arduino is called a sketch. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE)

pre-1.0 saved sketches with the extension .

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub `main()` into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program `avr-dude` to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

PROGRAM STRUCTURE



FIG. 1.1 ARDUINO UNO COMPATIBLE BOARD

A minimal Arduino C/C++ program consist of only two functions

- ***setup()***: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.
- ***loop()***: After `setup()` has been called, function `loop()` is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

POWER:

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power

jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN:** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V:** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3:** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND:** Ground pins.

MEMORY:

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

INPUT AND OUTPUT:

Each of the 14 digital pins on the Uno can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- ***TWI: A4 or SDA pin and A5 or SCL pin.*** Support TWI communication using the Wire library
- ***External Interrupts: 2 and 3.*** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- ***PWM: 3, 5, 6, 9, 10, and 11.*** Provide 8-bit PWM output with the analogWrite() function.
- ***SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).*** These pins support SPI communication using the SPI library.
- ***LED: 13.*** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the analogReference () function. Additionally, some pins have specialized functionality:

- ***TWI: A4 or SDA pin and A5 or SCL pin.*** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- ***AREF.*** Reference voltage for the analog inputs. Used with analogReference().
- ***Reset.*** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB

connection to the computer (but not for serial communication on pins 0 and 1).

- A Software Serial library allows for serial communication on any of the Uno's digital pins.



FIG. 1.2. ZIGBEE

Zigbee is an [IEEE 802.15.4](#)-based [specification](#) for a suite of high-level communication protocols used to create with small, low-power [digital radios](#), such as for home automation, medical device data collection, and other low-power low-bandwidth needs, designed for small scale projects which need wireless connection. Hence, Zigbee is a low-power, low data rate, and close proximity (i.e., personal area) [wireless ad hoc network](#).

The technology defined by the Zigbee specification is intended to be simpler and less expensive than other [wireless personal area networks](#) (WPANs), such as [Bluetooth](#) or more general wireless networking such as [Wi-Fi](#). Applications include wireless light switches, [home energy monitors](#), traffic management systems, and other consumer and industrial equipment that requires short-range low-rate wireless data transfer.

Its low power consumption limits transmission distances to 10-100 meters [line-of-sight](#), depending on power output and environmental characteristics. Zigbee devices can transmit data over long distances by passing

data through a [mesh network](#) of intermediate devices to reach more distant ones. Zigbee is typically used in low data rate applications that require long battery life and secure networking (Zigbee networks are secured by 128 bit [symmetric encryption](#) keys.) Zigbee has a defined rate of 250 kbit/s, best suited for intermittent data transmissions from a sensor or input device.

IDE

An **integrated development environment (IDE)** or **interactive development environment** is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs offer intelligent code completion feature. Some IDEs contain a compiler, interpreter, or both, such as Net Beans and Eclipse; others do not, such as Sharp Develop and Lazarus. The boundary between an integrated development environment and other parts of the broader *software development environment* is not well-defined. Sometimes a version control system and various tools are integrated to simplify the construction of a Graphical User Interface (GUI). Many modern IDEs also have a class browser, an object browser, and class diagram, for use in object-oriented software development.

CHAPTER 2

LITERATURE SURVEY

AUTOMATED CONTROL SYSTEM

Air quality contributes a major role in human performance especially in close environment where it is crucial to have an air quality management & control system, implemented in places with high occupancy like schools and multi-story offices or buildings. The work modifies the existing air quality management equipment already placed in industries and offices with high occupancy. The modified system is cost effective yet with same results and response time compare to commercial systems. Beside industries it can be used in households or shops. The main objective of this paper is to design an autonomous system to detect smoke, carbon mono oxide and other toxic gas in close doors with high occupancy. The hazardous gases like liquid petroleum gas(LPG), Methane, Carbon-Monoxide, Propane and Smoke, in addition to over temperature, were sensed and displayed continuously on the LCD and a popup alarm will be generated if any of the said elements cross specified ranges. In addition to this, the information is also transmitted to the command and control center over a wireless link, where an alarm is generated. The benefit of the proposed automatic detection and responsive system over the traditional scheme is the swift reaction with precise recognition in case of emergency, that can help to quickly diffuse the serious situation if arise. Having experience of various industrial and commercial audits, it is concluded that such type of devices are much needed in both safety and performance aspect. Besides that, central HVAC systems required certain level of air changes as the loss of chilled air is directly related to the loss of energy in building so the proper monitoring of above mention gasses are very important.

This approach allows integrating portable after-market sensors and cloud-based analytics into the climate control loop. [Energy Conservation & Efficiency \(ICECE\), 2017 International Conference.](#)

SMART VEHICLE AIR QUALITY MANAGEMENT

Severe in-vehicle air quality may affect drivers and passengers' health and reduce customers' satisfaction or experience with the vehicles, especially in some areas with heavy pollutants. Nowadays, in-vehicle cabin air quality management draws more attention from automakers. The advances in climate control include improved filtering systems and advanced climate control management based on air quality parameters from the in-cabin and external sensors. The proliferation of connected car technologies and data analytics provides opportunities for an efficient and effective cloud-based architecture for vehicle climate control management. The paper introduces analytical models for air quality mapping and discusses the design of a scientific experiment to test the effectiveness of cloud-based analytics under different scenarios. In our experiments we leverage Smart Device Link™ (SDL) technology that seamlessly integrates smartphone applications with the Ford SYNC infotainment system and allows programmatic vehicle management of climate control settings. This approach allows integrating portable after-market sensors and cloud-based analytics into the climate control loop. [Advanced Information Management, Communicates, Electronic and Automation Control Conference \(IMCEC\), 2016 IEEE.](#)

CONTROL SCHEME FOR SMART COMMUNITY:

Indoor Air quality (IAQ) is an urgent topic that worsens because of several of pollutants in the product indoor. World Health Organization (WHO) has standardized the harmful level indicating the potential health risk caused by hazardous substance. The conventional schemes consider single substance only which have no significant contribution to IAQ practically as IAQ depends on lots of parameters. As a result, a new comprehensive control scheme which deal with the IAQ and indoor environmental controlling, is designed. The results show that the use of both methods increases the efficiency of analysis because it highlights a more complete and effective set of criticalities and vulnerabilities. [Industrial Informatics \(INDIN\), 2016 IEEE 14th International Conference](#)

HOME MECHANICAL VENTILATION:

Risk analysis (RA) plays an important role in the development of safety-critical healthcare systems and numerous methods have been specifically

designed for the risk analysis. As a matter of fact, these processes are complex, composed by many sub-processes and characterized by the presence of sophisticated technologies used in diagnoses and therapies. According to these considerations, the Clinical RA (CRA) techniques derive from those used in other complex industrial systems, adapted on concepts and settings typical of healthcare. Often the industrial risk analysis methods combine more risk analysis methods with the advantage of obtaining multi-views analysis exploiting the different focus points of each method. In this work, a particular critical aspect of home care service with high level of assistance has been analysed using the combination of Failure Mode Effects and Criticality Analysis (FMECA) and Healthcare Failure Mode Effects Analysis (HFMEATM). The results show that the use of both methods increases the efficiency of analysis because it highlights a more complete and effective set of criticalities and vulnerabilities. [Medical Measurements and Applications \(MeMeA\), 2014 IEEE International Symposium.](#)

MICRO-SCALE AIR QUALITY MANAGEMENT

A new concept of CFD-based virtual sensor data was implemented in a micro-scale air quality management system. This new concept is expected to bridging gaps between present and future USN application. In addition to the VS implementation, air quality data management and multi-modal scientific visualization module are discussed in the paper. We also describe about an extended actual system to apply in reality such as a small area in modern urban. [Computer Sciences and Convergence Information Technology \(ICCIT\), 2011 6th International Conference](#)

A DATA TRANSFERRING MODEL FOR URBAN AREAS

Today, the environmental observing systems are no longer strange to the people, and they are playing an important role in e-sciences, especially in researches about earthquake, climate, air pollution, etc. To be named as air scope [1], this micro-scale air quality management system deals with unexpected pollution events in small scope, by using CFD-based air quality modeling [2], USN-based sensor monitoring model, Data Turbine middleware [3], multi-model interaction platforms and devices. Because of these complicated

heterogeneity tools and devices, data transferring inside this project is very important and is a backbone for the whole system. In this paper, we will briefly introduce air scope project and its elements, our indoor testbed and especially the data flows inside. We also describe about an extended actual system to apply in reality such as a small area in modern urban. [New Trends in Information and Service Science, 2009. NISS '09. International Conference](#)

AIR SCOPE:

Monitoring air quality for daily life gets more and more attention. Especially in modern urban environment with high skylines and unexpected pollution events, it is important to have micro-scale monitoring in addition to traditional monitoring methods. We propose a micro-scale modeling system as well as a micro-scale air quality monitoring system, which comprises as a micro-scale air quality management system, named air scope. The Air Scope consists of CFD-based air quality modeling, USN-based sensor monitoring, and multi-modal interaction platform. In this paper, we present a brief overview of Air Scope and several aspects of constructed initial indoor test environment with a few validity tests. The proposed system will be extended to an outdoor real-world testbed with most of modern urban elements. [eScience, 2008. eScience '08. IEEE Fourth International Conference.](#)

PARELLEL-NON LINEAR OPTIMIZATION:

We address the optimization component of a decision support system for air quality management, and describes some benchmark studies assessing its effectiveness. Because of the computationally demanding nature of the objective function we are interested in parallelizing the quasi-Newton BFGS algorithm selected for initial study. This is achieved by concurrent evaluation of functions in finite difference approximations to the derivative and a method of interval subdivision in simple bound constrained line searching. In a realistic case study, use of the parallel optimization algorithm is shown to have significant performance gains over other methods of solution. [SC '97: Proceedings of the 1997 ACM/IEEE Conference on Supercomputing](#)

GRID COMPUTING EXPLOITATION:

Leveraging grid computing technology, i.e. the visualization of distributed

computing and data resources such as processing, network bandwidth and storage capacity to create a single system image, we present a Grid Air Quality Forecast System (G-AQFS). The modeling system consists of meteorological and dispersion models coupled in cascade. The computational workflow of the modeling system is defined by means of DAGs (direct acyclic graph). A simple system is presented to manage and schedule the computational grid resources. As case study the system has been applied over Salento area, in the Apulia region (South-eastern Italy), to simulate ground level ozone concentration. Model predictions have been compared with field measurements, with reasonable results [Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference .](#)

MODELS FOR AIR QUALITY MANAGEMENT AND ASSESSMENT:

Models which could be useful for estimating air pollutant concentrations as a function of the emission distribution and the attendant meteorological conditions have been investigated. The integration of this model into an emission control scheme for the control of the pollution sources may represent a very useful approach to air quality management and assessment. A novel black box approach for air pollution modeling has been proposed. The main target of this approach is the prediction, on the basis of meteorological forecasts, of the air pollution concentration as a function of the expected causes. [IEEE Transactions on Systems, Man, and Cybernetics, Part C \(Applications and Reviews\)](#) (Volume: 30, [Issue: 3](#), Aug 2000).

CHAPTER 3

METHODOLOGY

BLOCK DIAGRAM

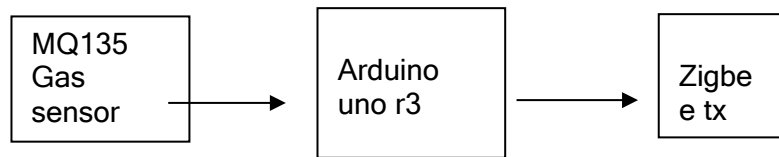


FIG. 3.1. ARDUINO UNO BLOCK DIAGRAM

RECEIVER SIDE

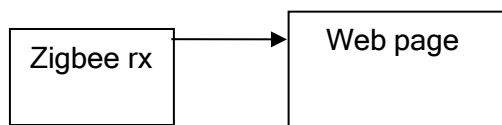


FIG. 3.2 WEB PAGE RECEIVER SIDE

WORKING :

The MQ135 sensor can sense NH₃, NO_x, alcohol, Benzene, smoke, CO₂ and some other gases, so it is perfect gas sensor for our **Air Quality Monitoring Project**. When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million). MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So for converting the output in PPM, here we have used a library for MQ135 sensor.

Sensor was giving us value of 90 when there was no gas near it and the safe level of air quality is 350 PPM and it should not exceed 1000 PPM. When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.

When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”. Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”. If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”.

REQUIREMENTS

HARDWARE REQUIREMENTS:

- MQ135 GAS SENSOR
- ARDUINO UNO
- ZIGBEE
- LCE

SOFTWARE REQUIREMENTS:

- JAVA
- PHP
- ARDUINO IDE
- MQ135 GAS SENSOR

In this project we are going to make an **IOT Based Air Pollution Monitoring System** in which we will **monitor the Air Quality over a webserver using internet** and will trigger a alarm when the air quality goes down beyond a certain level, means when there are sufficient amount of harmful gases are present in the air like CO₂, smoke, alcohol, benzene and NH₃. It will show the air quality in PPM on the LCD and as well as on webpage so that we can monitor it very easily.

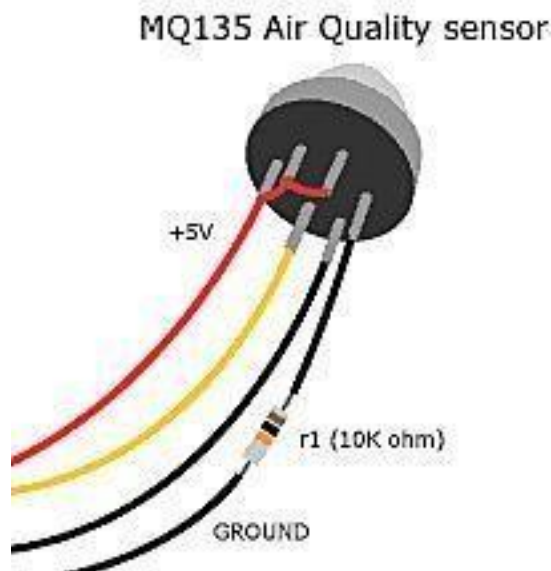


FIG. 3.3. MQ135 AIR QUALITY SENSOR

First of all we will connect the **ZIGBEE with the arduino**. ZIGBEE runs on 5V and if you will give it 5V from the Arduino The RX pin of ZIGBEE works on 5V and it will not communicate with the Arduino. This can be done by connecting

three resistors in series like we did in the circuit. Connect the TX pin of the ZIGBEE to the pin 1 of the Arduino and the RX pin of the ZIGBEE to the 1 of Arduino through the resistors.

ZIGBEE module gives your projects **access to Wi-Fi or internet**. It is a very cheap device and make your projects very powerful. It can communicate with any microcontroller and it is the most leading devices in the IOT platform.

Then we will connect the **MQ135 sensor with the Arduino**. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino. Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true. In last, we will connect LCD with the Arduino. The connections of the LCD are as follows

- Connect pin 1 (VEE) to the ground.
- Connect pin 2 (VDD or VCC) to the 5V.
- Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.
- Connect pin 4 (RS) to the pin 12 of the Arduino.
- Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so we will connect it to the ground.
- Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
- The following four pins are data pins which are used to communicate with the Arduino.

Connect pin 11 (D4) to pin 5 of Arduino.

PIN DESCRIPTION

TABLE 3.1 PIN DESCRIPTION

Pin Category	Pin Name	Details
--------------	----------	---------

Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/output Pins	Digital Pins 0 – 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10(SS),11(MOSI),12(MISO) and 13(SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
AREF	AREF	To provide reference voltage for input
TWI	A4 (SDA), A5	Used for TWI communication.

TABLE 3.2 TECHNICAL SPECIFICATIONS

Microcontroller	Microchip ATmega328P
Operating Voltage	5 Volts
Input Voltage	7 to 20 Volts
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	20 mA

DC Current for 3.3V Pin	50A
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Length	68.6 Mm
Width	53.4 mm
Weight	25 g

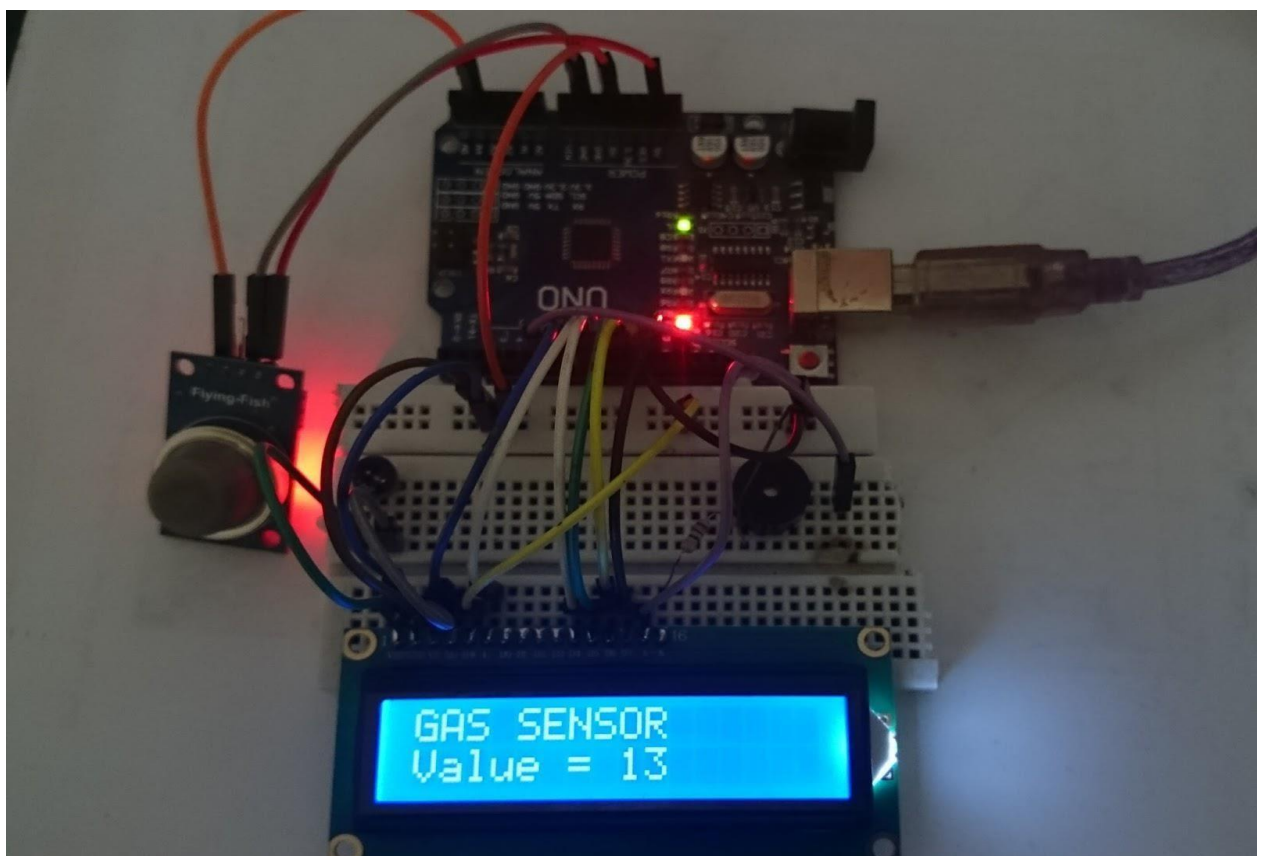


FIG. 3.4. ZIGBEE SENSOR VALUE

ZIGBEE TRANSCEIVER:

RFM75 data modem working at 2.4 Ghz frequency in half duplex mode with automatic switching of receive/transmit mode with LED indication. Receives and Transmits serial data of adjustable baud rate of 9600/4800/38400/19200 bps at RS232 level for direct interfacing to PC Serial port or such devices. This model can work with other 2.4 Ghz Sensor embedded models (TTL, 30 meter range) or RFM75 modem can be used for applications that need two way wireless data transmission.

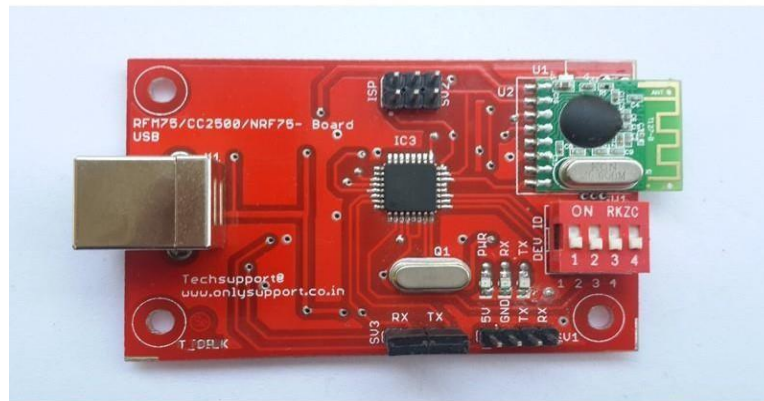


FIG. 3.5. ZIGBEE TRANSCEIVER

It features high data rate and longer transmission distance. The communication protocol is self controlled and completely transparent to user interface. The module can be embedded to your current design so that wireless communication can be set up easily.

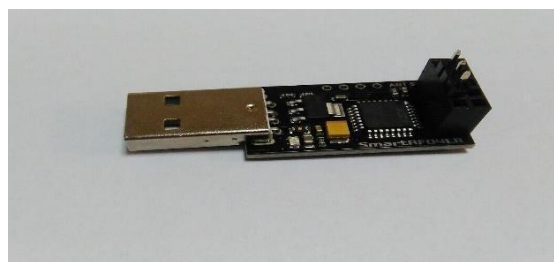


FIG. 3.6. ZIGBEE ADAPTOR

FEATURES:

- RS-232 direct interface
- TTL support
- Separate LED for RX, TX and power
- 30Mtr range

4 bit channel selection
Programmable output power
Variable payload length from 1 to 32bytes
Automatic packet processing
6 data pipes for 1: 6 star networks
4-pin SPI interface with maximum 8 MHz clock rate
20-pin 4x4mm QFN package

LANGUAGE DESCRIPTION

JAVA:

Initially the language was called as “oak” but it was renamed as “java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language
- Java is cohesive and consistent
- Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Finally Java is to Internet Programming where c was to System Programming.

IMPORTANCE OF JAVA TO THE INTERNET:

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

APPLICATIONS AND APPLETS:

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++ .Java’s ability to create Applets makes it important. An Applet is an application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

JAVA ARCHITECTURE:

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet. It is an intelligent program, not just a media file. It can react to the user input and dynamically change. Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet. Language that could be used to create software to be embedded in various consumer electronic devices. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine(JVM). The JVM is supposed to execute the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

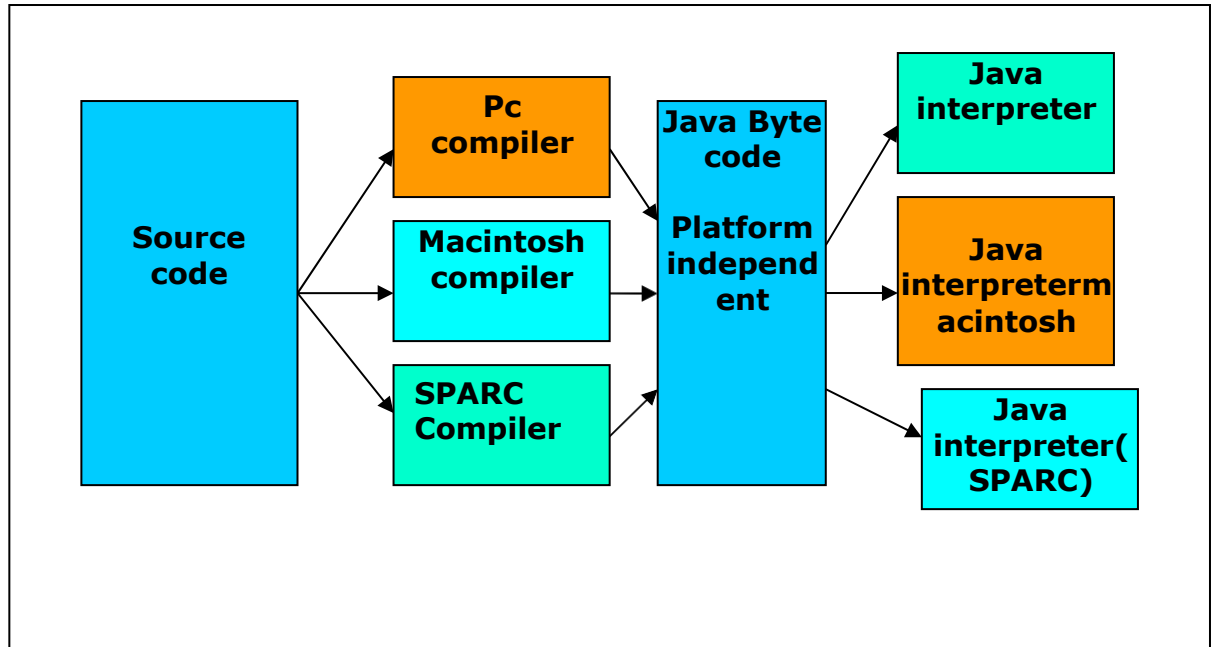


FIG. 3.7. CODE COMPILATION

CODE COMPILATION:

During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be an Intel Pentium windows 95 or sun SPARCstation running Solaris or Apple Macintosh running system and all could receive code from any computer through internet and run the Applets.

CSS:

While (X)HTML is used to describe the content in a web page, it is Cascading Style Sheets (CSS) that describe how you want that content to *look*. In the web design biz, the way the page looks is known as its presentation. CSS is now the official and standard mechanism for formatting text and page layouts. CSS also provides methods for controlling how documents will be presented in media other than the traditional browser on a screen, such as in print and on handheld devices. It also has rules for specifying the non-visual presentation of documents, such as how they will sound when read by a screen reader. Style sheets are also a great tool for automating production, because you can make changes to all the pages in your site by editing a single style sheet document. Style sheets are supported to some degree by all modern browsers.

Although it is possible to publish web pages using (X)HTML alone, you'll probably want to take on style sheets so you're not stuck with the browser's default styles. If you're looking into designing web sites professionally, proficiency at style sheets is mandatory.

SERVLETS:

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically. Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But Servlets offer several advantages in comparison with the CGI. Performance is significantly better. Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request. Servlets are platform-independent because they are written in Java. Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted. The full functionality of the Java class libraries is available to a servlet. It can

communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already. After the connections are established, communication can occur using I/O streams. Each socket has both an Output Stream and an Input Stream. The client's Output Stream is connected to the server's Input Stream, and the client's Input Stream is connected to the server's Output Stream. But web servers aren't limited to serving up static HTML pages; they can also run programs in response to user requests and return the dynamic results to the user's browser.

SERVLET ARCHITECTURE:

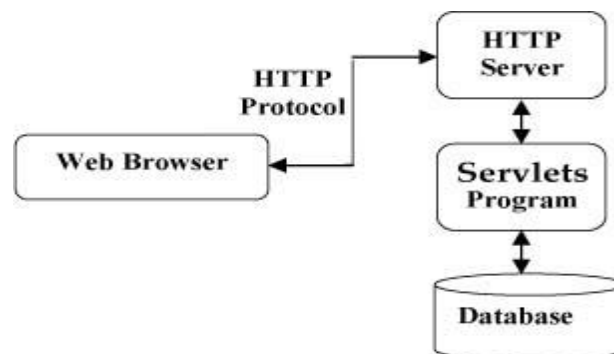


FIG. 3.8 . SERVLET ARCHITECTURE

MAJOR TASKS OF SERVLETS:

- Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.

- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.
- HTTP also provides methods for controlling how documents will be presented in media.

APACHE TOMCAT SERVER:

Tomcat is a Java servlet container and web server from the Jakarta project of the Apache Software Foundation (<http://jakarta.apache.org>). A web server is, of course, the program that dishes out web pages in response to requests from a user sitting at a web browser. But web servers aren't limited to serving up static HTML pages; they can also run programs in response to user requests and return the dynamic results to the user's browser. This is an aspect of the web that Apache's Tomcat is very good at because Tomcat provides both Java servlet and Java Server Pages (JSP) technologies (in addition to traditional static pages and external CGI programming). The result is that Tomcat is a good choice for use as a web server for many applications. And it's a very good choice if you want a free, open source (<http://opensource.org/>) servlet and JSP engine. Tomcat can be used stand-alone, but it is often used "behind" traditional web servers such as Apache http, with the traditional server serving static pages and Tomcat serving dynamic

Servlet and JSP requests. No matter what we call Tomcat, a Java servlet container or servlet and JSP engine, we mean Tomcat provides an environment in which servlets can run and JSP can be processed. Similarly, we can absolutely say a CGI-enabled Web server is a CGI program container or engine since the server can accommodate CGI programs and communicate with them according to CGI specification. Between Tomcat and the servlets and JSP code residing on it, there is also a standard regulating their interaction, servlet and JSP specification, which is in turn a part of Sun's J2EE (Java 2 Enterprise Edition). But what are these servlets and JSP? Why do we need them? Let's take a look at them in the following subsections before we cover them in much more detail in the future.

RMI:

The Java Remote Method Invocation (Java RMI) is a [Java API](#) that performs the object-oriented equivalent of [remote procedure calls](#) (RPC), with support for direct transfer of [serialized](#) Java classes and [distributed garbage collection](#).

1. The original implementation depends on [Java Virtual Machine](#) (JVM) class representation mechanisms and it thus only supports making calls from one JVM to another. The protocol underlying this Java-only implementation is known as [Java Remote Method Protocol](#) (JRMP).
2. In order to support code running in a non-JVM context, a [CORBA](#) version was later developed.

Usage of the term RMI may denote solely the programming interface or may signify both the API and [JRMP](#), whereas the term [RMI-IIOP](#) (read: RMI over [IIOP](#)) denotes the RMI interface delegating most of the functionality to the supporting [CORBA](#) implementation.

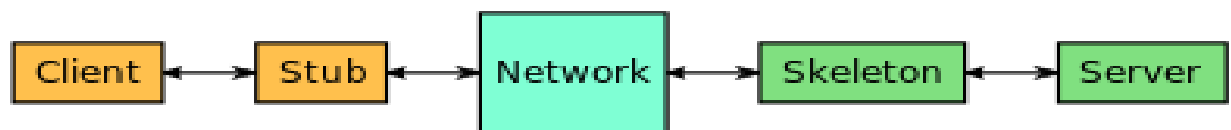


FIG. 3.9. CORBA IMPLEMENTATION

JAVA NETWORKING:

The term network programming refers to writing programs that execute across multiple devices (computers), in which the devices are all connected to each other using a network.

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.

The java.net package provides support for the two common network protocols:

- **TCP:** TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
- **UDP:** UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications. This tutorial gives good understanding on the following two subjects
- **SOCKET PROGRAMMING:** This is most widely used concept in Networking and it has been explained in very detail.
- **URL PROCESSING:** This would be covered separately. Click [here](#) to learn about URL Processing in Java language.

SOCKET PROGRAMMING:

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server. When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket. The `java.net.Socket` class represents a socket, and the `java.net.ServerSocket` class provides a mechanism for the server program to listen for clients and establish connections with them.

The following steps occur when establishing a TCP connection between two computers using sockets:

- The server instantiates a Server Socket object, denoting which port number communication is to occur on.
- The server invokes the `accept()` method of the Server Socket class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the `accept()` method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an `OutputStream` and an `InputStream`. The client's `Output Stream` is connected to the server's `InputStream`, and the client's `Input Stream` is connected to the server's `OutputStream`.

TCP is a two way communication protocol, so data can be sent across both streams at the same time. There are following useful classes providing complete set of methods to implement sockets.

UPLOADING DATA TO SERVER

Users can upload source data or other file-based resources for FME Server Workspaces. In the FME Server Web User Interface, File Upload is available for any workspace with a published parameter that represents a file used by the workspace. For more information, see [Uploading Data Files](#). Once data is uploaded, users can select the uploaded data for the workspace to use. Uploaded files are stored in a private location accessible to FME Server but not the user.

SHELF LIFE OF UPLOADED FILES

By default, files are deleted 1 day after they are uploaded. To change, update task `Delete_DataUpload_Files` on the System Cleanup page of the Web User Interface. For more information, see [Cleaning Up FME Server Logs and Other Files](#).
Request Elements Supported

REQUEST METHODS

This section describes the various methods available for interacting with the Data Upload service. The methods supported for sending files are PUT and POST. The HTTP GET method is supported for browsing files that are already uploaded. Remember that only volatile storage is supported and that all requests must be associated with an FME Server workspace.

CHAPTER 4

RESULTS AND DISCUSSIONS

SCREENSHOTS

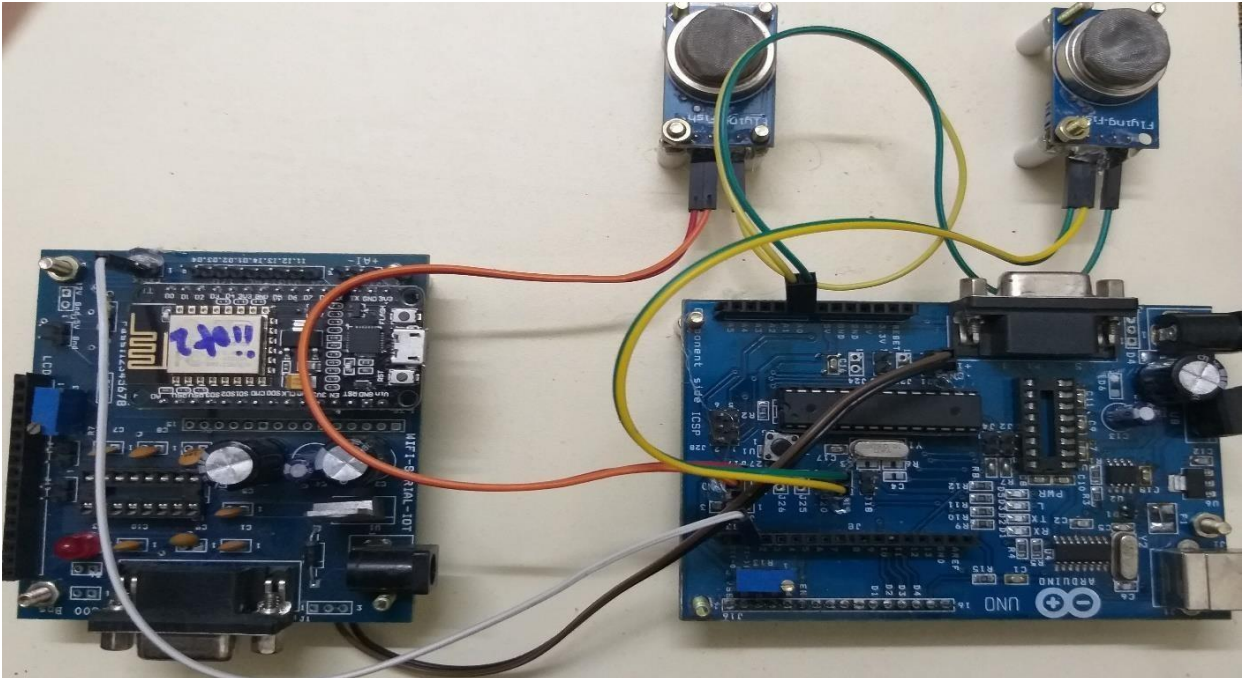


Fig:4.1 Active Arduino Board

In Fig:4.1 the Arduino board is connected to DC Supply that ensures that there is steady supply of electricity to the board. Whenever there is a power supply connected to the board there is a blinking of light on the adapter.

Login Form

The Login Form shows the procedure by which the user can login into the web page for viewing the data that is recorded from the sensor unit. Make sure that the user had entered the correct login details for the web page to be displayed to the user.

Login Form


Username

Password

Fig:4.2 Login Form

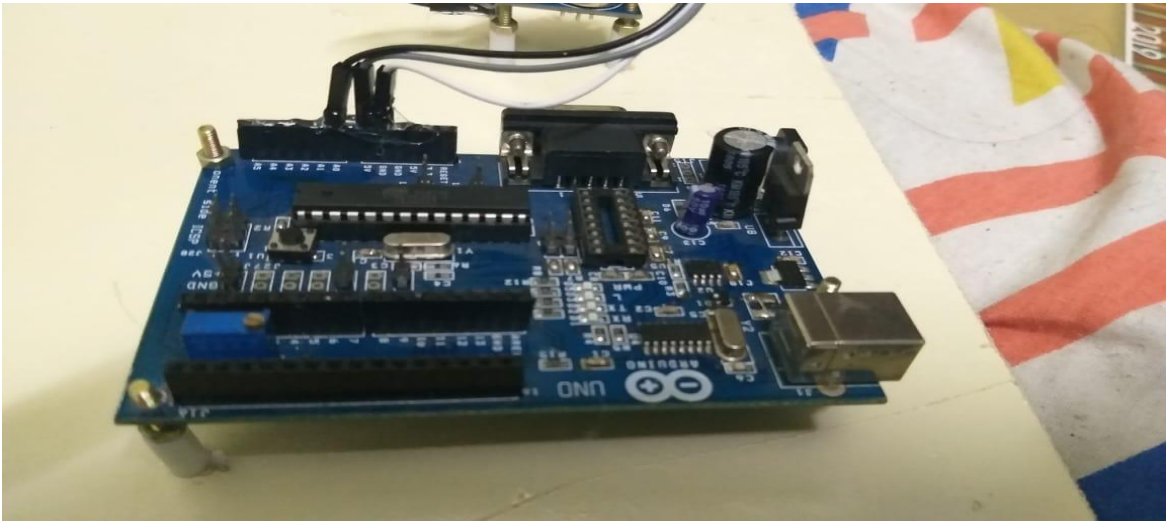


Fig:4.3 Arduino uno

We can monitor any electronic equipment in homes and industries via the Internet of Things (IOT). You can also read a data from any sensor, and graphically analyze it from anywhere in the world. Here, we can use Arduino Uno and ESP8266-01 module to read the sum of CO and CO₂ data from MQ2 sensor and upload it to cloud server. Arduino Uno is MCU, it collects and processes the amount of CO and CO₂ from the MQ2 sensor and a module ESP8266. ESP8266 is a WiFi module, and one of the leading Internet of Things platforms. It can transfer a data on to the IOT cloud.



Fig:4.4 MQ2 Gas Sensor

MQ2 gas sensor is an electronic sensor used to detect airborne gasses such as methane, hydrogen, alcohol, smoke, and carbon monoxide.

FINAL OUTPUT

Fig shows the measured readings of the sensor values with date and time with gas values that are displayed on the screen in closed room.

Air Quality Management System

Show 10 entries Search:

Sno	Sensor 1	Sensor 2	Date
1	208	62	2020-02-22 08:14:00
2	209	62q1	2020-02-22 08:13:56
3	209	62q1	2020-02-22 08:13:54
4	209	62	2020-02-22 08:13:54
5	210	62q1	2020-02-22 08:13:32
6	210	62	2020-02-22 08:13:30
7	210	62q1	2020-02-22 08:13:28
8	211	63	2020-02-22 08:13:27
9	211	63q1	2020-02-22 08:13:25
10	211	63	2020-02-22 08:13:22

Showing 1 to 10 of 479 entries Previous 1 2 3 4 5 ... 48 Next

Fig.4.5 Values of Gasses in Room

Air Quality Management System

Show 10 entries Search:

Sno	Sensor 1	Sensor 2	Date
11	212	63q1	2020-02-22 08:13:21
12	212	63	2020-02-22 08:13:19
13	212	63q1	2020-02-22 08:13:17
14	214	63q1	2020-02-22 08:13:14
15	213	63	2020-02-22 08:13:13
16	214	64q1	2020-02-22 08:13:11
17	221	66q1	2020-02-22 08:12:00
18	222	66	2020-02-22 08:11:41
19	222	66q1	2020-02-22 08:11:20
20	222	67q1	2020-02-22 08:11:14

Showing 11 to 20 of 479 entries Previous 1 2 3 4 5 ... 48 Next

Fig.4.6 Values of Gasses in Office

Fig shows the measured readings of the sensor values with date and time with gas values that are displayed on the screen in office.

TABLE 4.1 PPM VALUES OF GASSES

S.NO	SENSOR 1	SENSOR 2	DATE
1	205	64	2020-02-20 09:52:23
2	206	64	2020-02-20 09:52:21
3	205	63	2020-02-20 09:52:15
4	201	62	2020-02-20 09:52:11

5	197	61	2020-02-20 09:52:08
6	199	62	2020-02-20 09:52:03

PERFORMANCE ANALYSIS

These are the ppm values of gasses which are obtained from the gas sensor. These values are sent to Arduino uno board and from that the values are sent to WI-FI serial IOT kit in which values are sent to server.

Measuring the change in the amount of gases time to time depending upon the place.

The parameters can be observed which results in closed room. The speed at which input is processed can be modulated.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (SHIELDS) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION

There is increasing interest from the public, city administrators and regulators in the air quality within cities, there is also much greater awareness of the costs and impact of poor air quality. There are strong market forecasts for continued growth in this sector, in part driven by the needs of governments to reduce air pollution. Governments, cities and entrepreneurs will continue to invest in this sector, as without safe, clean air, cities and communities cannot flourish.

FUTURE WORK

Advances in computing power and the IoT, coupled with emerging Mobile IoT communications technologies, create opportunities for mobile operators to develop new revenue streams from air quality data products, services and solutions. In Future the common server will be created and every iot board will be connected to common server by the use of this we can access many private servers.

REFERENCES:

- [1] Arden Pope III, Majid Ezzati, and Douglas W Dockery. 2018. Fine-particulate air pollution and life expectancy in the United States. *New England Journal of Medicine* 2018, 360 (2018), 376-386.
- [2] Cheng.Y, X. Li, Z. Li, S. Jiang, and Y. Li, J Jia Air Cloud: a cloud-based air-quality monitoring system for everyone. in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. 2018. ACM, pp.148-213.
- [3] Devarakona, S. P. Sevusu, and H. Liu, Real-time air quality monitoring through mobile sensing in metropolitan areas. in *Proceedings of the 2nd ACM SIGKDD international workshop on urban computing*. 2018. ACM, pp.659-672.
- [4] Marjovi, A. Arfire, and A. Martinoli, High resolution air pollution maps in urban environments using mobile sensor networks. in *Distributed Computing in Sensor Systems, 2015 International Con. on*. 2018. IEEE, pp.216-224.
- [5] Jung-Yoon Kim, Chao-Hsien Chu, and Sang-Moon Shin. 2014. ISSAQ: An integrated sensing systems for real-time indoor air quality monitoring. *IEEE Sensors Journal* 14, 12 (2018), 4230-4244
- [6] M. Á. Olvera-García, J. J. Carbajal-Hernández, L. P. Sánchez-Fernández, and I. Hernández- Bautista, „„Air quality assessment using a weighted fuzzy inference system,““ *Ecol. Inf.*, vol. 33, pp. 57-74, May. 2018.
- [7] pan .L, B. Sun, and W. Wang, „„City air quality forecasting and impact factors analysis based on grey model,““ in *Proc. Procedia Eng.*, 2018, pp. 74-79
- [8] Y. Zheng, F. Liu, and HP Hsieh, U-air: When urban air quality inference meets big data. in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2018. ACM, pp.78-96.
- [9] Zheng,F .Y. Liu, and H. P. Hsieh,„„ U-Air: When urban air quality inference meets big data, ““in *Proc. ACM SIGKDD Int. Conf. Knowledge. Discovery Data Mining*, 2018, pp. 1436-1444.
- [10]Y.Zhengetal.,„„Forecasting fine-grained air quality based on bigdata,““ in *Proc.21thACMSIGKDDInt.Conf.Knowl.DiscoveryDataMining,2018*, pp. 2267-2276.

APPENDIX

ARDUINO CODE:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(13,12,11,10,9,8); //12=RS,E=11,D7=2,D6=3,D5=4,D4=5;

const int gasPin = A0; //GAS sensor output pin to Arduino analog A0 pin
const int gas1Pin = A1;

float gas,ags1;

void setup()
{
  Serial.begin(9600); //Initialize serial port - 9600 bps
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("AIR QUALITY SYS");
  lcd.setCursor(0,1);
  lcd.print("G : ");
  lcd.setCursor(0,1);
  lcd.print("AIR QUALITY SYS");
  lcd.setCursor(0,1);
  lcd.print("G1 : ");
}

void loop()
{
  float* values= mq2.read(true); //set it false if you don't want to print the values in
the Serial
  //lpg = values[0];
  lpg = mq2.readLPG();
  //co = values[1];
  co = mq2.readCO();
  float sensor_volt;
  float RS_air; // Rs in clean air
  float R0; // R0 in 1000 ppm LPG
  float sensorValue;
  //Average
```

```

    for(int x = 0 ; x < 100 ; x++)
    {
        sensorValue = sensorValue + analogRead(A0);
    }
    sensorValue = sensorValue/100.0;
    // _____/
    sensor_volt = (sensorValue/1024)*5.0;
    RS_air = (5.0-sensor_volt)/sensor_volt; // Depend on RL on yor module
    R0 = RS_air/9.9;
    Serial.print("sensor_volt = ");
    Serial.print(sensor_volt);
    Serial.println("V");
    Serial.print("R0 = ");
    Serial.println(R0);
    delay(1000);
    Serial.println(analogRead(gasPin));
    gas = analogRead(A0);
    lcd.setCursor(4,1);
    lcd.print(gas);
    delay(1000); // Print value every 1 sec.
    Serial.println(analogRead(gas1Pin));
    gas = analogRead(A0);
    lcd.setCursor(4,1);
    lcd.print(gas);
    delay(1000); // Print value every 1 sec.
    Serial.print("");
    Serial.print("G:");
    Serial.print(gas);
    Serial.print(",");

    Serial.print("G1:");
    Serial.print(gas1);
    Serial.println("#");
    //Serial.print(ms);

```

```
delay(3000);  
}
```

Java code :

```
import java.io.*;  
import java.sql.*;  
import javax.servlet.ServletException;  
import javax.servlet.http.*;  
  
public class Register extends HttpServlet {  
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
  
        String n=request.getParameter("userName");  
        String p=request.getParameter("userPass");  
        String e=request.getParameter("userEmail");  
        String c=request.getParameter("userCountry");  
  
        try{  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            Connection con=DriverManager.getConnection(  
                "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");  
  
            PreparedStatement ps=con.prepareStatement(  
                "insert into registeruser values(?,?,?,?)");  
  
            ps.setString(1,n);  
            ps.setString(2,p);  
            ps.setString(3,e);  
            ps.setString(4,c);  
  
            int i=ps.executeUpdate();  
            if(i>0)  
                out.print("You are successfully registered...");  
  
        }catch (Exception e2) {System.out.println(e2);}  
  
        out.close();  
    }  
  
}
```

```
import java.io.IOException;
```

```

import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 *
 * @author arul
 */
public class autho extends HttpServlet {
    /**
     * Processes requests for both HTTP GET and
     POST
     * methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String u2="",p3="",block="";
        try {
            String user =request.getParameter("username");
            String pass =request.getParameter("password");
            System.out.println(user+pass);

```

```

        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/medical","root","");
        Statement stmt=con.createStatement();
        ResultSet rs=stmt.executeQuery("select * from member where useid='"+user+"'");
        while(rs.next()) {
        u2=rs.getString("useid");
        p3=rs.getString("password");
        block=rs.getString("Action"); }
        con.close();

        System.out.println(u2+p3);
        if((u2.equals(user))
        (p3.equals(pass))&&((block=="no")||block.equals("no"))){
        request.setAttribute("user", user);
        Cookie loginCookie = new Cookie("user",user);
        //setting cookie to expiry in 30 mins
        loginCookie.setMaxAge(30*60);
        response.addCookie(loginCookie);
        request.getRequestDispatcher("/login").include(request, response);
        }else if(user.equals(pass)){
        request.getRequestDispatcher("/admin.jsp").forward(request, response);
        }else{
        if((block!="no")||!block.equals("no")){
        request.setAttribute("errorMessage", "****blocked****");
        }else{
        request.setAttribute("errorMessage", "Invalid user id or password");
        }
        request.getRequestDispatcher("/index.jsp").forward(request, response);
        }
        }catch(Exception e){
        request.getRequestDispatcher("/index.jsp").include(request, response);
        }
        }

```

