Jitesh Aggarwal | Bharat Vaidhyanathan

COmputer science and Engineering 2016 | Shiv Nadar University

Feasibility Study

Final Year Project

**Mentored by**

**Dr. Debopam Acharya**

**Associate Professor, Dept. of Computer and Engineering**

**Shiv Nadar Univerity**

Contents

[2 EXECUTIVE SUMMARY 4](#_Toc448184927)

[3 PROJECT MANAGEMENT 5](#_Toc448184928)

[3.1 The Team 5](#_Toc448184929)

[3.1.1 Jitesh Aggarwal: Group Member 5](#_Toc448184930)

[3.1.2 Bharat Vaidhyanathan: Group Member 5](#_Toc448184931)

[3.1.3 Dr. Debopam Acharya: Project Mentor 6](#_Toc448184932)

[3.2 Methodology 6](#_Toc448184933)

[3.2.1 Software dev Process 6](#_Toc448184934)

[3.2.2 Schedule 6](#_Toc448184935)

[3.2.3 Work Breakdown 7](#_Toc448184936)

[3.3 Budget 7](#_Toc448184937)

[3.4 Milestones 8](#_Toc448184938)

[4 SCOPE OF THE PROJECT 11](#_Toc448184939)

[5 PROJECT DESCRIPTION 12](#_Toc448184940)

[5.1 Problem Specification 12](#_Toc448184941)

[5.2 Client 12](#_Toc448184942)

[5.3 Vehicular Emission 13](#_Toc448184943)

[5.3.1 The Process of Combustion 14](#_Toc448184944)

[5.3.2 By-products of Combustion 15](#_Toc448184945)

[5.3.3 Emission Standards in India 16](#_Toc448184946)

[5.4 Data Aggregation and Data Fusion 16](#_Toc448184947)

[6 Technical Requirements 18](#_Toc448184948)

[6.1 Performance Requirements 18](#_Toc448184949)

[6.1.1 Sensing Gas Requirements: 18](#_Toc448184950)

[6.1.2 Mobility Requirements 18](#_Toc448184951)

[6.2 Hardware Requirements 18](#_Toc448184952)

[6.2.1 Core Setup Requirements: 19](#_Toc448184953)

[6.2.2 Sensing Requirements: 19](#_Toc448184954)

[6.2.3 Communication Requirements: 20](#_Toc448184955)

[6.2.4 Server Machine/Cloud: 20](#_Toc448184956)

[6.2.5 Other Electronic Requirements: 20](#_Toc448184957)

[6.3 Software Requirements 21](#_Toc448184958)

[6.3.1 Java SE: 21](#_Toc448184959)

[6.3.2 Java EE: 21](#_Toc448184960)

[6.3.3 Python: 21](#_Toc448184961)

[6.3.4 Meteor (Full development stack suing Blaze and MongoDB): 21](#_Toc448184962)

[6.3.5 Android Studio: 22](#_Toc448184963)

[6.4 Deliverables 22](#_Toc448184964)

[6.4.1 Basic Setup: 22](#_Toc448184965)

[6.4.2 Software Deployed in the Raspberry Pi 2: 22](#_Toc448184966)

[6.4.3 Software Deployed on the Server: 23](#_Toc448184967)

[6.4.4 Documentation 23](#_Toc448184968)

[6.4.5 Full Working Prototype: 23](#_Toc448184969)

[6.4.6 The Website: 23](#_Toc448184970)

[6.4.7 Mobile Application: 23](#_Toc448184971)

[6.4.8 Project Poster: 23](#_Toc448184972)

[6.4.9 Completed Codes in a CD 23](#_Toc448184973)

[7 DOMAIN ANALYSIS 24](#_Toc448184974)

[7.1 Microcontroller Board 24](#_Toc448184975)

[7.1.1 About 24](#_Toc448184976)

[7.1.2 Raspberry Pi 24](#_Toc448184977)

[7.1.3 Arduino Mega 24](#_Toc448184978)

[7.1.4 Choice of On Board Computer 24](#_Toc448184979)

[7.2 Data Communication 25](#_Toc448184980)

[7.2.1 About 25](#_Toc448184981)

[7.2.2 WiFi 25](#_Toc448184982)

[7.2.3 Mobile Data 25](#_Toc448184983)

[7.2.4 Bluetooth 25](#_Toc448184984)

[7.2.5 Choice of Communication Mode 25](#_Toc448184985)

[7.3 Temperature and Humidity Sensor 26](#_Toc448184986)

[7.3.1 About 26](#_Toc448184987)

[7.3.2 Combined Temperature and Humidity Sensor 26](#_Toc448184988)

[7.3.3 Separate Temperature and Humidity Sensor 26](#_Toc448184989)

[7.3.4 Choice of Temperature and Humidity Sensor 26](#_Toc448184990)

[7.4 Gas Sensors 26](#_Toc448184991)

[7.4.1 About 26](#_Toc448184992)

[7.4.2 MQ-x Gas Sensor Modules 27](#_Toc448184993)

[7.4.3 MQ-x Gas Sensors 27](#_Toc448184994)

[7.4.4 Choice between sensors and modules 27](#_Toc448184995)

[7.5 Analog to Digital Converter 27](#_Toc448184996)

[7.5.1 About 27](#_Toc448184997)

[7.5.2 MCP3008 27](#_Toc448184998)

[7.5.3 ADC0809 27](#_Toc448184999)

[7.5.4 Arduino as ADC 28](#_Toc448185000)

[7.5.5 Choice of ADC 28](#_Toc448185001)

[7.6 Web User Interface & Database Technology 28](#_Toc448185002)

[7.6.1 About 28](#_Toc448185003)

[7.6.2 Meteor 28](#_Toc448185004)

[7.6.3 LAMP (Linux Apache MySQL PHP) 28](#_Toc448185005)

[7.6.4 Choice of Web Development Stack 29](#_Toc448185006)

[8 PROJECT SETUP 30](#_Toc448185007)

[8.1 Overview 30](#_Toc448185008)

[8.2 Hardware Setup 30](#_Toc448185009)

[8.2.1 Setting up the Raspberry Pi 30](#_Toc448185010)

[8.2.2 Interfacing the Sensors 32](#_Toc448185011)

[8.2.3 Interfacing the Analog to Digital Converter (MCP3008) 33](#_Toc448185012)

[8.2.4 Calibrating the MQ-x Sensors 34](#_Toc448185013)

[8.2.5 Data Collection 35](#_Toc448185014)

[8.3 Web System Setup 35](#_Toc448185015)

[8.3.1 Meteor 35](#_Toc448185016)

[8.3.2 Java Web Service 37](#_Toc448185017)

[9 RISK ANALYSIS 38](#_Toc448185018)

[9.1 Sensors 38](#_Toc448185019)

[9.2 Wireless Communication 38](#_Toc448185020)

[9.3 ADC Failover 39](#_Toc448185021)

[9.4 GSM/GPRS Module 39](#_Toc448185022)

[10 CONCLUSION 39](#_Toc448185023)

# **EXECUTIVE SUMMARY**

Air pollution in metropolitan cities is rising at an alarming rate. To tackle this issue, first step is to make people aware of the problem. It is required that we have an accurate and reliable system of measurement. Not only should the data be consistent but it should also be timely updated. There are various groups and organizations having their own models. Instead, if everybody used a similar way of sensing and had a simple and easy way to upload it to a common database, the data gathered could be very fruitful. Crowd Sensing is one way to address the problem. Using the Internet-of-Things system with participatory sensing, the project aims to collect several environmental data from a large subset of the pollutants in vehicular emission, analyse it and exhibit it in a visually appealing and meaningful manner. This may serve as a crowd sourced way to monitor the external environment which could be scaled later for greater environmental impact.

# PROJECT MANAGEMENT

This section discusses the Team members & their responsibilities, methodology adopted and the various milestones for the duration of the project.

## The Team

The team for this project comprised of two working members and one mentor. All the members add a lot to the team in their own right. The developmental work is equally divided between the two working members. A greater part of the project was done together, with the required guidance from the mentor.



### Jitesh Aggarwal: Group Member

Jitesh is a fourth year Computer Science and Engineering student. He was born and raised in the city of New Delhi. Jitesh adds a lot of value to the team as a Java Developer with his natural aptitude. His managerial side adds more than just technical depth to the team. Jitesh is a dedicated and hardworking member, always eager to learn new technologies and getting the things done. His past experience with Kalakriti and as a Graphic Designer also offers a different outlook to the team.

Image 3.1



### Bharat Vaidhyanathan: Group Member

Bharat is a fourth year Computer Science and Engineering student. He was born and raised in the city of Mumbai. Bharat carries the reputation of one of the smartest coders in the Batch of 2016. His technical abilities and knack of picking up technical stuff quickly is essential to the team. His jovial and gregarious attitude brings much needed positive energy during the course of such a demanding project.

Image 3.2



### Dr. Debopam Acharya: Project Mentor

Dr. Debopam Acharya heads the Department of Computer Science and Engineering at Shiv Nadar University. His research interests are in various domains of Mobile Computing. He is the best person to mentor a project based on an IOT setup. He is a true asset to the team as a focussed guide and a motivator. His knowledge adds to the finer learning during the duration of project building.

Image 3.3

## Methodology

### Software dev Process

We used Agile methodology as the appropriate software development process for the project. The sprints comprised of bi-weekly scrums. These bi-weekly scrums were in coordination with the bi-weekly team meetings. As a result of this, the code development, testing and review occurred twice every week. The members stuck to the process to the best of their ability with certain exceptions. Since the development, testing and review happened simultaneously, the team benefitted in the longer run with many roadblocks being identified at earlier stages. The work done was incrementally, which allowed the members to review their progress and determine their path for the remainder of the project.

### Schedule

The project schedule was well structured. The meetings being of two types: Team meetings and Mentor meetings. The team had scheduled bi-weekly Team meetings and weekly Mentor meetings. Team meetings involved the team members coming together and discussing about current state of the project and its short term goals. The team meetings generally consisted of topics related to the immediate requirements of the project. Mentor meetings involved both the team members meeting with the mentor during a designated time at least once every week. Mentor meetings were used to evaluate the steps needed to be taken in the upcoming week and their impact in the longer run. The latter meetings were also useful for the mentor to keep a track over the progress of the students working under him.

### Work Breakdown

The Table 3.1 below shows the work breakdown between the two members. Although the work was equally divided and completed as a team, the below table only aims to serve as an indicator to highlight their major contributions respectively.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | H/W Design | Algo Design | Coding | Debugging | Documentation | Hours/Week |
| Bharat |  | ✓ | ✓ | ✓ | ✓ | 20 |
| Jitesh | ✓ |  | ✓ | ✓ | ✓ | 20 |

Table 3.1

## Budget

The budget for the project was determined by finding the best price available for each of the components needed for the project to work. These components were chosen depending on the cost and the level of necessity. Some degree of risk analysis was done before buying any of the products or components, some of which has been discussed in a later chapter. Once the components for the project were chosen, the team worked hard to get these components for the best available price and within least time possible. Table 3.2 shows the major component requirements for the project and their cost to the team.

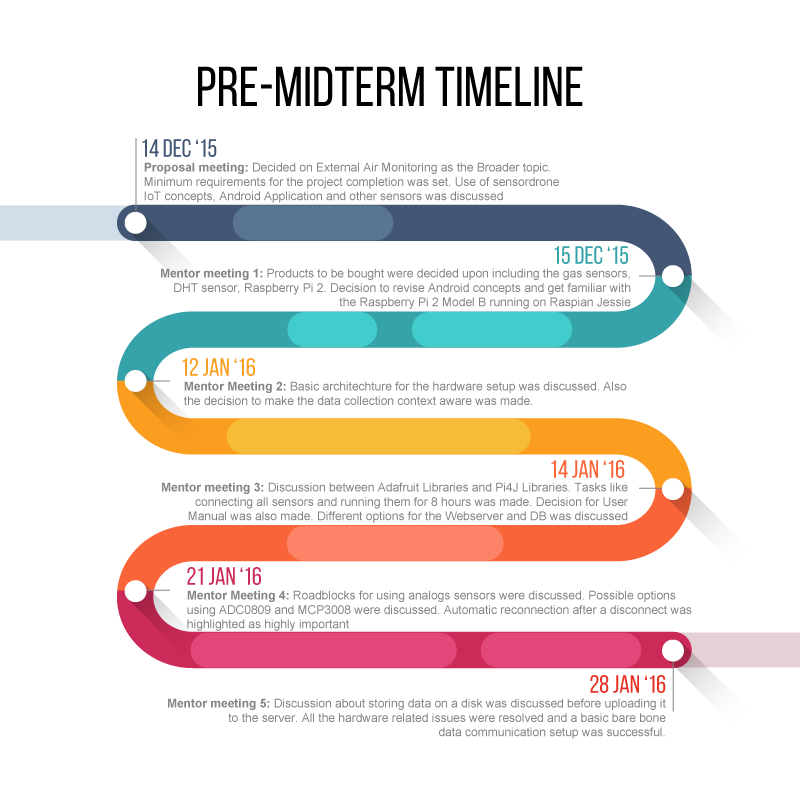
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO. | PRODUCT | QUANTITY | COST | TOTAL |
| 1 | Raspberry Pi 2 Mod B 1Gb  (Complete Kit with WiFi Module) | 1 | 5000 | 5000 |
| 2 | MQ-X sensors (135, 2, 4, 6, 7, 9 ) | 6 | 200 | 1200 |
| 3 | DHT22 Sensor | 1 | 350 | 350 |
| 4 | ADC MCP3008 | 1 | 250 | 250 |
| 4 | Power Bank 10000maH | 1 | 1200 | 1200 |
| 5 | Jumper Wires Set +Breadboard | 1 | 500 | 500 |
| 6 | MTS 3G Dongle | 1 | 1400 | 1400 |
| 7 | Development Machine | 1 | - | - |
| 8 | Server Machine | 1 | - | - |
|  |  |  |  | **9900** |

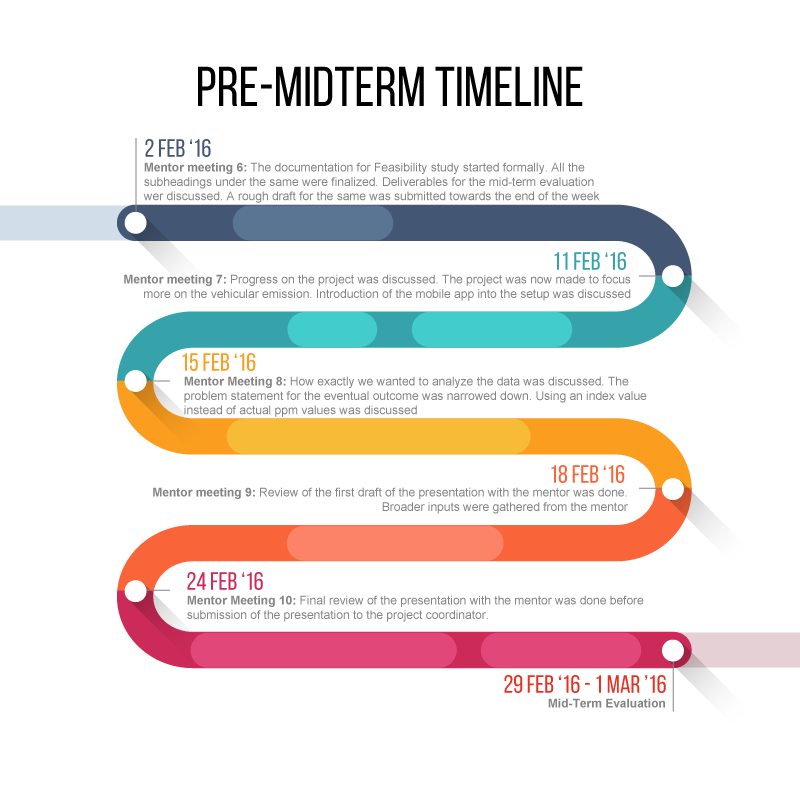
Table 3.2

## Milestones

The team used a list of specific tasks throughout the semester as milestones to judge completed progress and future goals. The milestones completed in the semester are laid out in this section along with the details of each of the mentor meetings.

[Date | Activity | Meeting and who attended it {Team meeting | Project Meets}]





Listed below were the key tasks that were completed during the stretch of the semester. While the first half milestones revolved around doing necessary background research, setting up hardware and software, the second half milestones were the ones which completed the loop. Greater part of the code development was done in the second half with supported documentation.

* Setup Raspberry Pi with Raspbian OS with desired configuration
* Basic hardware setup
* A working webserver with a DB using both MySQL and Meteor
* Feasibility Study
* Necessary Background Research
* Mobile App Integration
* Making the data context aware
* Calibrate the sensors to sense the vehicular emission
* Report Documentation
* Completion of the IoT setup with non-WiFi connectivity through Internet Dongle
* A refined UI for the webserver
* Testing for a fully functional crowd-sensing platform

# SCOPE OF THE PROJECT

Building the monitoring system as discussed, although seems simple at first, is a challenging task. The architecture of the project widens the scope of the project as a developer. The data collection involves an IoT system including a Raspberry Pi setup with certain MQ-X gas sensors to sense various gas levels and a DHT22 sensor to sense ambient temperature and humidity. It also includes an Android phone paired with a device called sensordrone, which would behave as a neighbourhood sensor. A sensordrone is a small piece of hardware with several sensors built in. The entire hardware-setup is location aware and connected to the “IOT cloud” for storing the accumulated data. Web servers will make use of the final data and display it to the users on AirCheQ Web Platform. The aim of the team is to deliver a working prototype which aims to serve as a demonstration for such an IoT based participatory sensing setup.

# PROJECT DESCRIPTION

In order to get a focussed approach for development of the project, a basic scope for the project was decided between the mentor and the working members. This was essential to get the minimum expectations from the project. The team discussed and made a problem specification, which was followed by the members in order to complete the project before April 15, 2016. Once the basic understanding of the project was made, the team narrowed down on the requirements for the development of the project and a schedule was made for the same too.

## Problem Specification

This project is designed to serve as a prototype for a crowd sourced external environment monitoring system. The idea is to deliver a completely working system which could be scaled later for large scale use. The project involves using a Raspberry Pi 2 Mod B equipped with MQ-X sensors coupled with DHT22 sensor which work as the I/P source to sense various gases, temperature and humidity levels. A sensordrone connected to an Android smartphone serves as a neighbourhood sensor and an additional source of input for the setup. A complete list of the sensors used is given in the Sensing Requirements sub-section under the Hardware Requirements section. A central database server is also setup to which all the Raspberry Pi(s) upload their data. A web server is used to display the collected data in a meaningful and graphically pleasing manner in form of graphs which can be easily read. An Android mobile application would also be developed which would complement the web-server for quick display of data on mobile devices.

## Client

As a mobile crowd sensing platform, it could be adopted earliest by several government initiatives. It could even be installed in cars/taxis which are on the move throughout the day. This would enable us to collect data from all across the city from various sources, instead of relying on readings from a stationary sensor at a particular location.

* **Delhi Pollution Control Committee**: It is a government empowered committee which looks after the controlling pollution in Delhi. With the Odd-Even vehicle rule being exercised in Delhi, they could act as one of the earlier adopters by investing in the setup and letting people collect and view the data around them which was sensed by them and their peers. This would help to bring about behavioural change by making people aware of the hazardous gases around them.
* **Indian Pollution Control Association:** It is a not-for-profit, non-government organization (NGO) which has been certified by the Central Pollution Control Board to create awareness and implement pollution control systems over the country. They work both in the public and private sector to educate the people about an environmental approach. They could help spread the use of AirCheQ system in order to continue the same.

## Vehicular Emission

The exhaust of the vehicles generate a low amounts of smoke as compared to the thick dense black gas one usually imagines. That being said vehicular emission accounts for a great part of air pollution in many major cities around the globe. As there are millions of vehicles on the congested roads, each adds to the growing air pollution little by little. Apart from construction which generates a very high amount of dust particles in the air, the vehicular emissions from our everyday commute is what harms our air quality the most. The vehicular emission often combines with the fog to produce a dense gaseous state in the form of smog which is very injurious to health.

All vehicles ranging from two-wheelers to four-wheelers, from light motor vehicles to heavy transportation vehicles, all generate their own share of pollution while they move about. Some vehicles, like heavy duty transportation vehicles, produce more emissions than the other. But they all contribute. While the petrol and diesel driven vehicles contribute majorly towards CO and PM emissions respectively, the CNG driven vehicles are one of the major emission source of nitrogen oxides. According to vehicular emission inventory of Delhi the amount of CO, NOx and PM levels are roughly 509, 194 and 15 tons per day respectively for the calendar year 2008-2009.

The diagram below shows the percentage of major emission by different vehicles types in Delhi in calendar year 2008-2009 in percentage.



Image 5.1

One of the major source of these emission is the exhaust itself. Exhaust lets out the by-products generated by the combustion process of the engine. Although contrary to the belief that exhausts are the only source of pollutants, the emissions caused by evaporation of fuel while refuelling as well as otherwise also degrade the air quality.

### The Process of Combustion

Combustion of fuels leads to the exhaust gas being emitted. The fuels can be of various types such as natural gas, petrol, diesel, etc. But what’s more important is that each one of them is nothing but a combination of hydrocarbons i.e. compounds made up of Carbon and Hydrogen atoms.

During the combustion process, the vehicles lets out two kinds of components: toxic and non-toxic gases. The undesirable gases which are produced during the process include carbon monoxide (CO) from incomplete combustion, various hydrocarbons (HC) from unburnt fuel, nitrogen oxides (NOx) from high combustion temperatures and particulate matter (PM). Apart from the listed gases, nitrogen, water vapour and carbon dioxide constitute the components which aren’t as noxious as the rest, but CO2 does add up to the greenhouse gas content.

A perfect combustion process of a typical fuel results in the following reaction. Although this reaction is an ideal case scenario but doesn’t necessarily happen in real life.

**FUEL** + **OXYGEN** (2O2) -> **WATER** (2H2O) + **CARBON DIOXIDE** (CO2) + Energy

The fuel burns not just in pure oxygen but in a mixture of oxygen and nitrogen and hence producing nitrogen in a perfect reaction in “AIR”.

**FUEL**(hydrocarbons)**+ AIR**(oxygen and nitrogen)**= CARBON DIOXIDE**(CO2)**+ Water**(H2O)**+ Nitrogen**

The atmosphere is not pure oxygen as it contains other gases like nitrogen (78%), argon, hydrogen and other gases & compounds. These compounds have effect on the theoretical reaction mentioned earlier. They limit the concentration of oxygen which is effectively available and also acts as contaminants. Therefore when the process of combustion takes place in real life many other toxic and noxious by-products are also produced. The by-products of the mentioned process can be summarised by the reaction below. Partially unburnt hydrocarbons, nitrogen oxides, carbon monoxide, carbon dioxide and water vapour constitute the major resultants of the reaction.

**FUEL**(hydrocarbons)**+ AIR**(oxygen and nitrogen)**= UNBURNED or PARTIALLY BURNED HYDROCARBONS**(VOCs)**+ NITROGEN OXIDES** (NOx)**+ CARBON MONOXIDE**(CO)**+ CARBON DIOXIDE**(CO2)**+ Water**(H2O)

### By-products of Combustion

#### Hydrocarbons

Hydrocarbons are by-products which are produced during combustion emission as well as during the fuel evaporation emission. When the molecules in the fuel do not burn completely or burn partially, hydrocarbons are produced. These can combine with various other compounds in the atmosphere to form various noxious substances. Ozone is one such example which is bad for human health and puts them at risk.

#### Carbon Monoxide

Carbon Monoxide is a poisonous gas which is a result of combustion taking place in limited supply of oxygen. As carbon compounds do not get sufficient oxygen, they are unable to completely oxidize themselves into Carbon Dioxide and hence stay as Carbon Monoxide. It can greatly affect the bloodstream in humans and be critical to those suffering from cardiac problems.

#### Nitrogen Oxides

Nitrogen Oxides are similar to hydrocarbons in the way that they also contribute towards formation on ozone below at the surface. They are formed as a result of reaction between oxygen and nitrogen at high pressure and temperatures. They might also combine with water vapour to produce toxic liquids in the atmosphere.

#### Carbon Dioxide

Although not as poisonous as Carbon Monoxide, but Carbon Dioxide still is not considered to be the best of gases as it is one of the major greenhouse gas. It has contributed to the rising global temperatures as a result of trapping sun’s energy.

#### Soot

During the combustion process, particulate matter often gets collected which is referred to as soot. It is very dark and oily which is formed due to incomplete burning of compounds. Instead of oxidizing and forming CO2, it gets collected as amorphous carbon. It constitutes of large amounts of polycyclic aromatic hydrocarbons. Diesel exhaust vents out far more soot in the atmosphere than its other counter-parts. This is majorly due to Diesel being a less refined fuel.

### Emission Standards in India

India follows Bharat stage emission standards which are governed by Government of India and closely resemble Euro emission norms. It regulates the emissions standards from the combustion inside the motor vehicles. These norms have to be followed by all motor vehicle manufacturers. India has been slowly making its standards stringent and aim to closely follow Euro VI norms by year 2020.

These norms if implemented can potentially bring down the air pollution levels in the country. Although it adds to cost of manufacturing the vehicles following these standards. Continuous exposure to air pollution can result in various serious health issues, specially respiratory and cardiovascular diseases. The Bharat emission standards sets benchmarks for various categories of vehicles by limiting the emission of CO, Hydrocarbons and Nitrogen Oxides. Although similar to Euro standards, it continues to neglect Carbon Dioxide as a benchmark.

## Data Aggregation and Data Fusion

Often while setting up a hardware similar to the mentioned project, we need to have an energy constraint in mind. Since the sensors are mobile, energy consumption by the nodes in the network should be tried to minimize. One way to do the same if by limiting the data processing at the node and collecting the data at a common node and processing it there. Data aggregation and fusion are often confused for another, which is essentially not the case. There lies a difference between the two terms. Data fusion refers to the techniques used to gather data from numerous sources and then process it to deduce inferences, correlations and even more accurate data than what a simple source of data would have given. Data aggregation, however Data aggregation can be classified as a part of Data Fusion. It is nothing but compiling and summarizing data from multiples sources to reduce redundancy. Using Data Fusion and Aggregation techniques can help immensely in energy efficiency, scalability as well as data accuracy.

# Technical Requirements

## Performance Requirements

The project’s performance requirements determine how well it will function. The project functionality is designed taking into account the desired usability by having required performance measures as well as avoiding being overly expensive.

### **Sensing Gas Requirements:**

Mapping to the project requirements, one of the essential needs of the project is to sense the gases found in the external environment that might be harmful in some way. The gas sensors selected were on the basis that the sensors needed to be powerful enough to sense common level of concentrations. Also they required to be compatible with Raspberry Pi setup without being overly expensive. Also the project required the current supply needed by the sensors as they would be powered by the mobile Raspberry Pi 2 Mod B itself.

### Mobility Requirements

The project required mobility to inbuilt as one of the major traits. All the necessary equipment was purchased keeping the need of mobility in mind. Since this is intended to be used by an individual, it requires to be small enough to be carried around without much difficulty. Since it is a mobile setup, it is also powered by a 10400 maH battery pack. The setup also supports a Wi-Fi module in order to let the sensor values to be uploaded onto the server when a network area is reached. One can use a MTS 3G dongle is case there’s no internet connectivity.

## Hardware Requirements

The project’s system needs to be robust enough to work for prolonged hours. The various parts of the system need to be able to interact with other parts using the required technology. Each sensor must adhere to certain minimum specifications mentioned. A high level diagram of the entire IOT setup, which displays all the major I/Ps and O/Ps and also highlight how they interact with each other, is shown in Figure below.

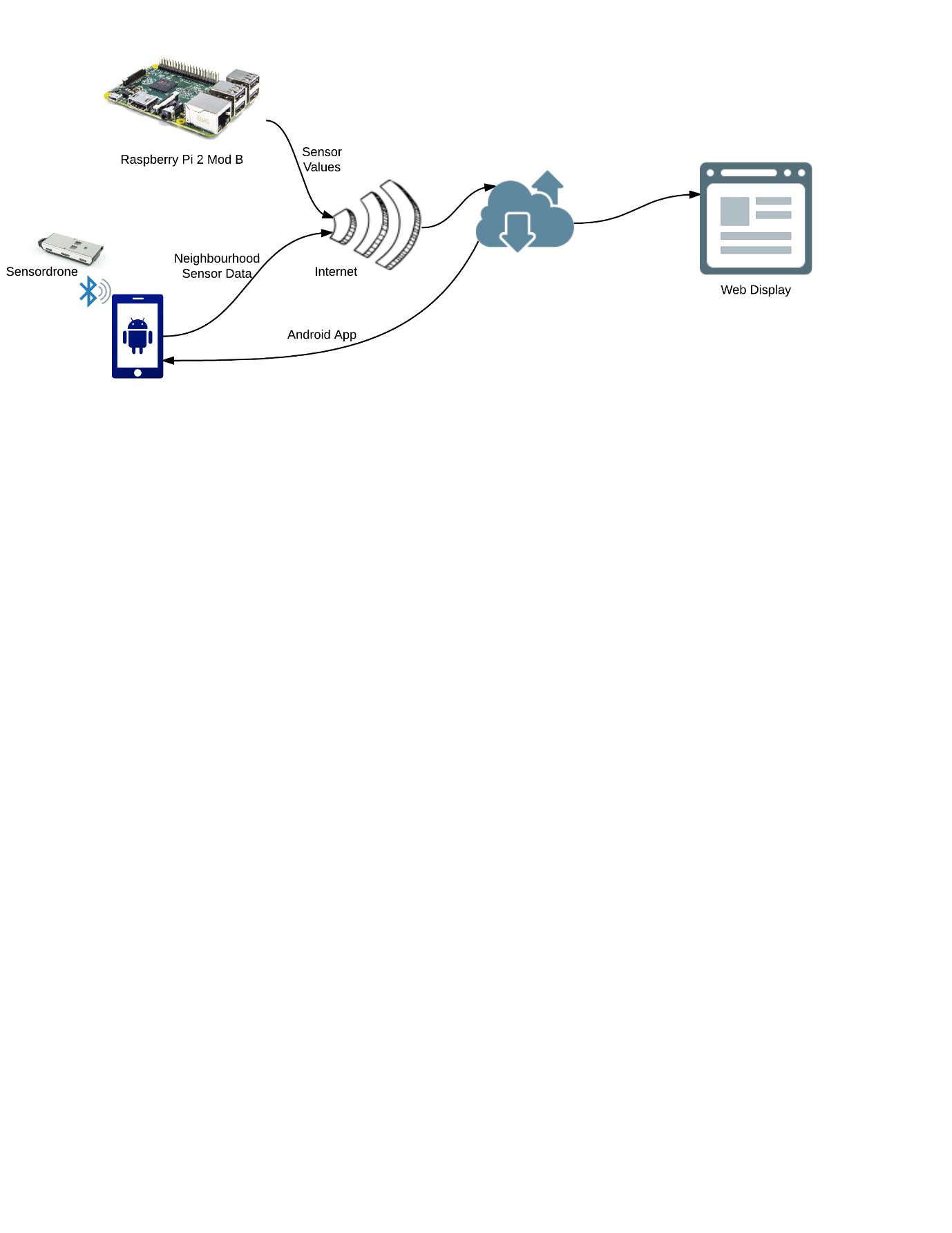


Image 5. 2 A high-level diagram of the IoT setup used

### Core Setup Requirements:

The Raspberry Pi is a single-board computer which is quite popular amongst the youth. It has one of the cheapest prices for single-board computers with adequate memory and processing power. In addition to its low price, Raspberry Pi has the largest support forum of any single-board computer on the market due to its open-source coding and wide fan base in the do-it-yourself market. This contributes to a great deal of support to troubleshooting and coding. The Raspberry Pi comes in 4 models: A, A+, B, and B+. All use a 700MHz ARM11 family CPU and a 250MHz Broadcom VideoCore IV GPU. A models have 256MB of SDRAM memory while the B models have 512MB. The plus models have a greater number of GPIO ports. We have selected the latest model which is Raspberry Pi 2 Mod B which has 4 USB-ports and comes at a price tag of ₹3000 without the kit and between ₹4000-5000 for a starter’s kit.

### Sensing Requirements:

Thevarious sensors like MQ-x sensors, with the necessary information, used in the setup are listed below in the table. Each of the MQ-x sensor can sense more than one gas, but it is always more suited to sense one particular gas. All the sensors listed below are used in the project to return value of only a single gas. However, MQ-7 and MQ-9 have both been used to sense CO with different sensitivity in order to get a complete result.

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | Sensor | Sensing Capability | Gas Sensed. |
| 1 | MQ-135 | NH3,NOx,alcohol, Benzene, Smoke, CO2 | **Carbon Dioxide** |
| 2 | MQ-2 | LPG, Propane, Hydrogen, Combustible Steam | **Smoke** |
| 3 | MQ-4 | CH4, Natural Gas, LNG | **Methane** |
| 4 | MQ-6 | LPG, iso-butane, propane, LNG | **LPG** |
| 5 | MQ-7 | Carbon Monoxide | **Carbon Monoxide** |
| 6 | MQ-9 | Carbon Monoxide, Methane and LPG | **Carbon Monoxide** |

Table 6.1

#### DHT22

DHT22 is a small temperature and humidity sensor that senses digitally. It is equivalent to the size of a coin and is very light. It is able to output humidity levels with a an error of not more than 2-5% with a temperature sensing range between -40 and 80 degrees Celsius. Its sampling rate is once every 2 seconds.

### Communication Requirements:

In order to upload data to the server, the setup can make use of the WiFi as well as the Ethernet cable. Although the Raspberry Pi 2 Mod B comes inbuilt with an Ethernet port, a WiFi module is necessary in this mobile age. An Ethernet port may not be accessible everywhere, therefore equipping the setup with a small USB WiFi adapter gave ample support. The one used for testing is Tenda W311MI, an 802.11n compliant wireless USB Adapter that provides wireless reception over 802.11g products while staying backward compatible with 802.11g/b devices. It has an ultra-compact design and is nearly invisible. The USB Adapter supports WEP, WPA, and WPA2 encryption for connecting to a secure wireless network connection.

Apart from WiFi connectivity, the setup also enables the user to use a MTS 3G dongle when not near a WiFi source for ease of use. This will let the user collect data almost anywhere where he/she is likely to go in daily life.

### Server Machine/Cloud:

A dedicated server machine would be required where the server is hosted and also the DB. It should be able to support Windows 7 Operating system and above. The server machine will host the meteor server along with a Java Web Service. The meteor stack also combines with itself the MongoDB, so the database doesn’t have to be hosted separately.

### Other Electronic Requirements:

Other electronic/hardware requirements include a portable power-bank with a 2 Amp output rating. This is the easiest way to provide your setup the juice for its functioning. A sensordrone would also be required in the IOT setup which would work as the neighbourhood sensor while staying connected to the mobile platform. A couple of development machine/laptops would also be required for the coding and testing the system. Also a smartphone running on android would complete the major hardware requirement for the project.

## Software Requirements

The scope of the project requires working across multiple platforms. Building an IoT system would involve interacting between different platforms. The development of this project would require the developers to have a basic understanding of IoT as a concept. The most pivotal part of the whole setup will the various raspberry pi(s) which serve as the main source of data collection. Below is the list of technologies which are being used to bring the project to life.

### Java SE:

The Java SE platform is used by the developers to program the networking part of the setup. Reading the data from the terminal and performing file handling with it is done through Java SE. Also sending the data to the database is done using Java SE.

### Java EE:

The Java EE platform is used to develop RESTful web service for the web server to be able to fetch data from the DB securely. It consists of various GET methods which are used by the Web UI as and when required.

### Python:

This language is used for reading the GPIO pins of the Raspberry Pi. It is used to deliver the meaningful data to the terminal by interpreting the data from the sensors. Python was chosen as it was fast compared to Java SE and more readily available Raspberry Pi resources for it.

### Meteor (Full development stack suing Blaze and MongoDB):

Meteor is an open-source, full-stack JavaScript framework that makes it easy to write top-quality web apps. There are lots of different technology components that have to work together to get a full-fledged web app up and running; the particular components you choose constitute your "stack" (e.g. what kind of database you use, what server-side programming language you use). Meteor allows you to write both your client and server code in JavaScript. Meteor is real-time i.e. changes made in the code are reflected as soon as it’s saved, without having to refresh the server. It makes use of MongoDB which is lightweight and quick. It is a non-relational database.

### Android Studio:

The application for the android smartphones will be developed using Android Studio. Android Studio is the official IDE available for the platform. It makes android development a little easier with an interactive environment.

## Deliverables

Since this project serves as the Final Semester Project for the completion of Bachelors of Technology Degree from Shiv Nadar University, there are a number of hardware and software deliverables that are due to the faculty panel and project advisor for grading.

### Basic Setup:

A basic hardware setup complete with all the sensors able to write the data into the DB and display it on a web server.

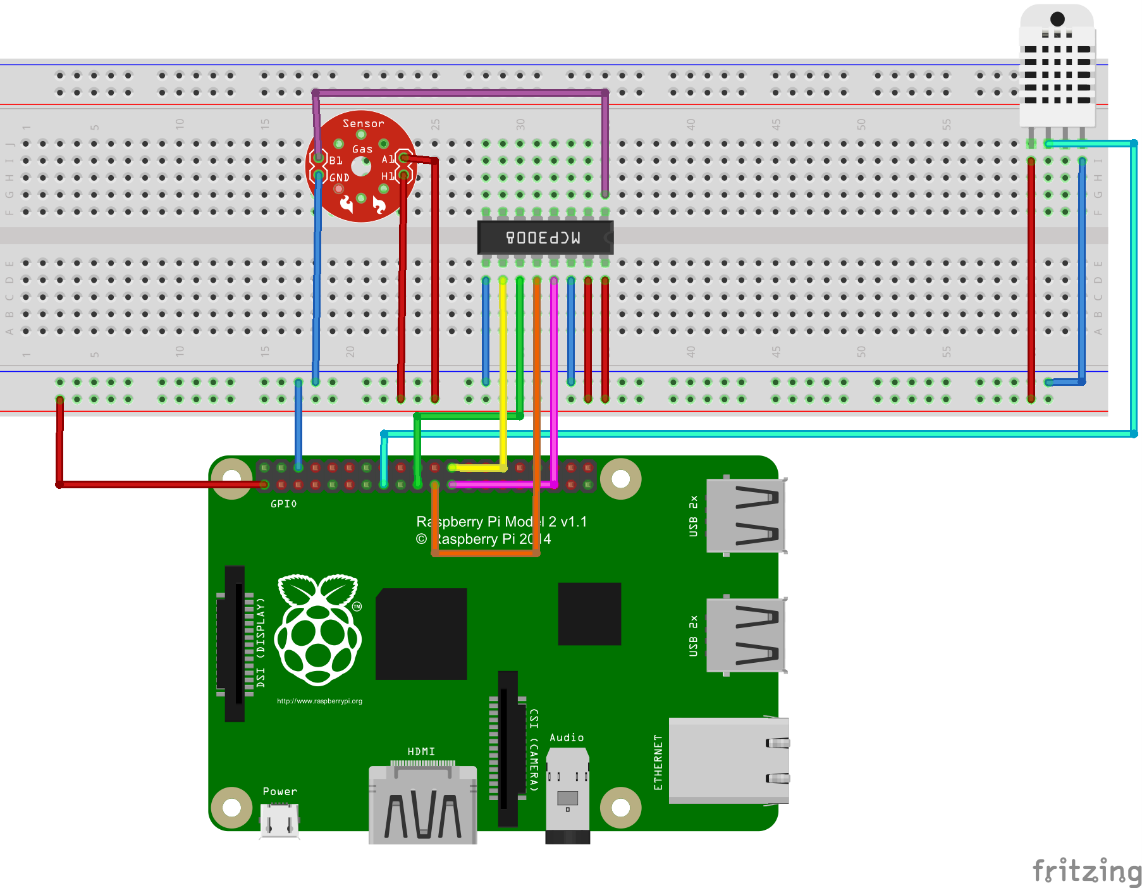


Image 6. 1 A basic hardware setup to read MQX sensor from channel 1 of ADC3008 and DHT22 readings directly from GPIO pins of Raspberry Pi 2 Mod B

### Software Deployed in the Raspberry Pi 2:

A fully functional Raspberry Pi with the latest Raspbian Jessie OS installed. All the necessary configurations should be complete and tested.

### Software Deployed on the Server:

Meteor and Web services deployed on the server machine for connecting from anywhere.

### Documentation

A project report stating in detail about the project. The project report will all the necessary details required for setting up the project hardware. It will be a detailed script about the project development.

### Full Working Prototype:

A fully functional setup for the IOT system with the necessary hardware and software support.

### The Website:

The finally deployed website which showcases the data collected from the sensors in a meaningful way in form of graphs with couple of sort options.

### Mobile Application:

An Android based mobile application for the user to quickly monitor the ppm level readings from the sensordrone.

### Project Poster:

A final poster to the panellist shall also be submitted summarising our project details.

### Completed Codes in a CD

A completed and in-depth code files which have been used in the entire project.

# DOMAIN ANALYSIS

The system to be developed in the project had various sub parts needed to be acquired or used. The team researched and did domain analysis on the available technologies that needed to be acquired or used.

## Microcontroller Board

### **About**

An onboard computer was necessary in order to run the sensors and push these values to the server. Apart from sending these values to the server, the system must also files to run various sensors.Since the project required a single board computer with GPIO pins to interact with the sensors, most popular single board computers in the market were researched.

### Raspberry Pi

The Raspberry Pi has one of the biggest communities online for projects involving sensors and home automation. It is the most popular single-board computer on the market. For its high memory capacity and huge processing power, it is relatively low priced. It’s wonderful support forums and huge fan base makes it one of the most versatile computers. The Raspberry Pi comes in the following versions 1A, 1A+, 1B, 1B+, 2B, zero. These models have varying specifications and hence are differently priced.

### Arduino Mega

The arduino family of microcontrollers are one of the most famous microcontroller series. The Arduino Mega 2560 and Arduino Uno are two of the more popular boards. They are wildly popular in the market for their compatibility with almost all sensors. These Arduino microcontrollers require less power than the raspberry pi’s but lack in GPU power and Graphical user interface.

### Choice of On Board Computer

The Raspberry Pi 2 B was chosen for use in the project. The decision was made for its cheap price and the processing power and networking capabilities it offers. Also, the raspberry pi was easy to acquire, hence the team started to work on the project immediately.

## Data Communication

### About

The system developed in the project would be equipped with gas sensors and would be providing meaningful values to the server. In order to establish this remote communication, a pre-existing wireless technology must be chosen. In order to choose the perfect technology needed for the project, the team decided to do research on the available technologies. The team initially decided that the technology used must be wireless as the system developed would be used at a varying distance from the server.

### WiFi

‘Wireless Fidelity’ was the first option that occurred in the team’s mind. It is one of the most used technologies. Transmitting data through WiFi is fast, reliable and secure. WiFi offers almost unlimited range as once the system is connected to a router, it can both transmit data to and receive data from almost any place on earth.

### Mobile Data

Mobile data through CDMA/GSM accessed by dongle was considered by the team in order to provide the user ease of uploading data when not near a WiFi source. This would let the data submitted be real-time instead of collecting and then uploading the data later.

### Bluetooth

The next immediate wireless technology that the team thought of was Bluetooth. This is also one of the most used wireless technologies currently used in the market. Bluetooth is both fast and easy to setup. The drawback with this technology though is that it does not have the range a WiFi router has and does not have a pre-existing intranet setup for its immediate implementation.

### Choice of Communication Mode

With the above research done in the respective fields, the team chose WiFi and 3G dongle as the appropriate means of remote communication between the system and the server. The campus has a pre-existing setup of WLAN by the name “Student” accessible to the team. This was chosen as the initial WiFi on which the setup would be run. Later, the team might shift the project to a safe and private WLAN. The team also tested a MTS 3G dongle to connect to the server successfully.

## Temperature and Humidity Sensor

### About

Sensing Temperature and Humidity was one of the projects various sources of data inputs. In order for the system to be developed in the project to run smoothly and efficiently, the technology used in sensing this must be wisely chosen. The team eventually had to decide between whether to choose an all in one sensor which would sense both temperature and humidity, or to use separate sensors for sensing both.

### Combined Temperature and Humidity Sensor

The DHT22 is a small digital temperature and humidity sensor. It is approximately the size of a quarter, and weighs very little. It is able to sense all humidity levels with a 2-5% accuracy and can read temperatures between 0 and 80 degrees Celsius. It is able to sample only once per 2 seconds. This sensor is widely available in the market and is budget friendly too.

### Separate Temperature and Humidity Sensor

When components are combined, they have the potential to lose functionality. In order to increase accuracy, it would be possible to implement separate humidity and temperature sensors. The HH10D can detect humidity levels with only a 3% error, marginally better than the combined system, 37 and the TMP36 can detect temperatures in the range of -40 to 125 degrees Celsius with only 2 degrees of error.

### Choice of Temperature and Humidity Sensor

The DHT22 was chosen because of its compact size, cost and its ease to use. The error in the DHT22 was acceptable. Also, since we chose the primary onboard computer as Raspberry Pi 2, choosing the DHT22 made more sense as the driver for the same was easily available and extremely stable.

## Gas Sensors

### About

Gas Sensors are the backbone of the project. Hence choosing the appropriate sensor was vital. The team decided to buy the MQ gas sensors as they are the most common and cost efficient gas sensors currently available in the market. The decision the team had to make was whether to buy the sensors only or buy the sensor modules.

### MQ-x Gas Sensor Modules

The MQ-x series of gas sensors are the most widely available gas sensors in the market. The usage of sensor modules in the project would mean reliable and uninterrupted readings to the system in the project. Also usage of sensor modules cuts down the usage of extra unnecessary resistors needed for the sensors to run individually.

### MQ-x Gas Sensors

The MQ-x Gas sensors could not be directly connected to the Raspberry Pi as then the readings would be erratic. This would make it difficult for the team to make sense of the readings.

### Choice between sensors and modules

Though the sensor modules were double the cost of the sensors in them, the team decided to choose them as they chose consistency over cost for the project. The entire project would depend on these readings, hence it was vital these readings were accurate and consistent.

## Analog to Digital Converter

### About

The project consists of a system of an on-board computer which reads values from the various gas sensors connected to the computer. Unfortunately, the on-board computer, in our case, Raspberry Pi 2 Mod B, cannot read analogy values, the default output of the numerous MQ-x gas sensors used. Hence, for the Raspberry Pi to read, these values must first be converted into digital values. This involves the usage of an Analog to digital converter. There were various ADC’s (Analog to Digital Converters) available in the market. All were available depending on the requirements.

### MCP3008

This was the ADC mentioned everywhere online. The ADC works perfectly with the Raspberry Pi. The ADC has 16 pins in all and supports upto 8 channels. The ADC was not available directly in India and hence had to be imported via a third party reseller.

### ADC0809

ADC0809 is one of the simplest available ADC’s. It has a very basic internal design structure. As a result it could not be directly connected to the GPIO pins of the Raspberry Pi. Hence a very complex circuit should be built in order to make it work. The ADC is abundantly available in Indian markets both online and offline.

### Arduino as ADC

A general trend in the market is to use an Arduino as an ADC. It was mentioned widely online and offline stores also recommended us. The Arduino Uno is best suited for this job. Though this might be quite expensive than a regular ADC, and the fact that an Arduino for this job might be an overkill, the problem would be solved by this method. The circuit for the same is very simple.

### Choice of ADC

The team decided to use the MCP3008 as the ADC in the project. This was mainly because of two reasons. Firstly, this was the most common solution online. The second being this method was both cost effective and simpler when comparing it to using ADC0809 as the ADC. We did not go forward with using an arduino as the ADC since it was a bit expensive and the fact that we earlier decided not to use Arduinos. However, we decided to keep this as our failover plan.

## Web User Interface & Database Technology

### About

Since our project involves data processing and then displaying meaningful results, a server would be hosting a website for the same. There are various technologies available to choose from. Since there are various technologies available to choose from, the team researched a lot about this. Eventually the team was stuck between using a new technology, Meteor, and using a well-established but older technology, LAMP.

### Meteor

Meteor is a relatively new technology which is emerging in the market. The setup of meteor is very simple as compared to older technologies. Meteor boasts excellent performance as well as wide scalability. The meteor stack primarily comprises of Blaze for its front end and MongoDB as its back end. Blaze has a smooth and stable UI template powered by JavaScript. It has bootstrap implemented within itself for automatic compatibility with mobile devices. MongoDB is a very fast and robust technology for managing large data.

### LAMP (Linux Apache MySQL PHP)

This is the industry standard in web hosting. An apache server would be hosting HTML, CSS and PHP pages with a SQL database. This is the most common technology in web development and hosting. It has a big community to discuss and solve problems.

### Choice of Web Development Stack

It was decided that the team would go on with Meteor as the server side of the project was relatively simple, and that the team hopes it would not encounter much errors when using newer technologies. Also, the faster development through Meteor stack strengthened the team’s resolve.

# PROJECT SETUP

## Overview

This chapter discusses the project setup in greater detail. It consists of two major sections, namely Hardware Setup and Web System Setup. The Hardware Setup section discusses how all the aforementioned components are made use of in the project. It also depicts the pin diagrams and explains the significance of each of the connections made in the hardware setup. The section also discusses the process of calibration of gas sensors.

The Web System Setup section discusses the software architecture of the system. It highlights how data is transferred from the circuitry to the Raspberry Pi, from Raspberry Pi to the Database and finally from Database to the UI via a Java Web Service which aggregates data and passes on for further use.

## Hardware Setup

### Setting up the Raspberry Pi

1. **Operating System:** Before reading the sensors, the Raspberry Pi 2 Mob B has to be configured. A system OS needs to be installed on the microSD card that gets plugged in to the Pi. The system OS used for this project’s development was the Raspbian Jessie OS, which needs to be timely updated. Users can either directly flash the microSD card with the OS image file or use a software called NOOBS which directly downloads and installs the selected OS on booting up the Raspberry Pi for the first time. Make sure to update and upgrade the system on installation by using the “sudo apt-get update” and “sudo apt-get upgrade” commands on the terminal. This will require internet connectivity, so make sure to use an Ethernet cable.
2. **WiFi Connectivity:** The Raspberry Pi 2 Mod B does not come in with an in-built WiFi setting, so one has to use a WiFi USB module. Once plugged in, the raspberry pi should automatically install its drivers. Although few of the files have to be configured in order to get the WiFi module up and running. Use command “sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf” to add the network details to the Raspberry Pi. The details to be added would vary depending on the WiFi network being used. Image below shows the wpa\_supplicant.conf file configured for Shiv Nadar University’s “Student” network. Once finished editing the details, use “sudo reboot” command to let the changes take effect.
3. **Oracle Java:** Installing Java is essential to run Java programs on the Raspberry Pi. Although the latest Raspian images ship with Oracle Java by default. But if Java is missing, one can use the command “sudo apt-get update && sudo apt-get install oracle-java7-jdk” on the terminal to setup the Oracle Java on the Raspberry Pi.
4. **Python and Adafruit Library:** To get the Adafruit Library working for reading the Digital Humidity & Temperature sensor, you will need to execute the command “git clone <https://github.com/adafruit/Adafruit_Python_DHT.git>” from /home/pi. Once finished, use execute “cd Adafruit\_Python\_DHT”. Before moving forward and using the library python will need to be installed by using the following commands “sudo apt-get update” followed by “sudo apt-get install build-essential python-dev python-openssl”. Finally execute “sudo python setup.py install” and you are ready to use the Python and the Adafruit Libraries. Change directory to examples and use “**s**udo ./AdafruitDHT.py 22 4” command to read the DHT22 sensor connected to GPIO4 pin.
5. **Spidev:** Since the MCP3008 is an SPI based device, we need to enable Spidev in the Raspberry Pi in order to make it work. Use the command “sudo nano /etc/modprobe.d/raspi-blacklist.conf” to open the raspi-blacklist.conf file and comment out “blacklistspi-bcm2708” by placing a “#” in front of it. Reboot the Raspberry Pi and type “ls /dev/spidev\*” to check if the SPI driver has been enabled. If the terminal output is similar to the one in the image below, the driver is working.

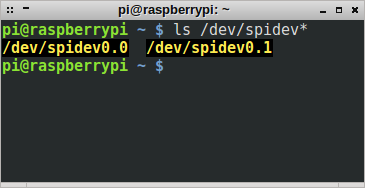


Image 8.1 Output of the terminal for successfully enables SPI driver

### Interfacing the Sensors

The hardware setup makes use of MQ-135, MQ-2, MQ-4, MQ-6, MQ-7, MQ-9 and DHT22 sensors. While the MQ-x sensors are analog sensors i.e. they give analog outputs, the DHT22 sensor is a digital sensor which outputs temperature and humidity values through the digital GPIO pins in Raspberry Pi 2. All of the sensors require three connections to be made to the Raspberry Pi.

1. The 5V GPIO pins of the Raspberry Pi is used to supply Vcc to the sensors. Raspberry Pi 2 has 5V sources, at Pin 2 and Pin 4. Refer to Image 8.1 for your reference.
2. The GND pins of the Raspberry Pi is used to ground the sensors. Raspberry Pi 2 has several ground pins which can be used as per user’s convenience.
3. The third connection required is the one used by the sensor to transmit data to the Raspberry Pi. Since the GPIO pins of Raspberry Pi 2 Mod B reads only digitally, the data pin of the DHT22 sensor can be connected to any of the GPIO pins directly. Only thing to be taken care of here is that the pin number used in the sensor library to read the data should mention the logical GPIO pin number and not the physical pin. The logical pin numbers are mentioned in the rectangular boxes adjacent to the pins in Image 8.1. Although MQ-x sensors cannot be interfaced so easily and requires an Analog to Digital Converter which is discussed in the following section.

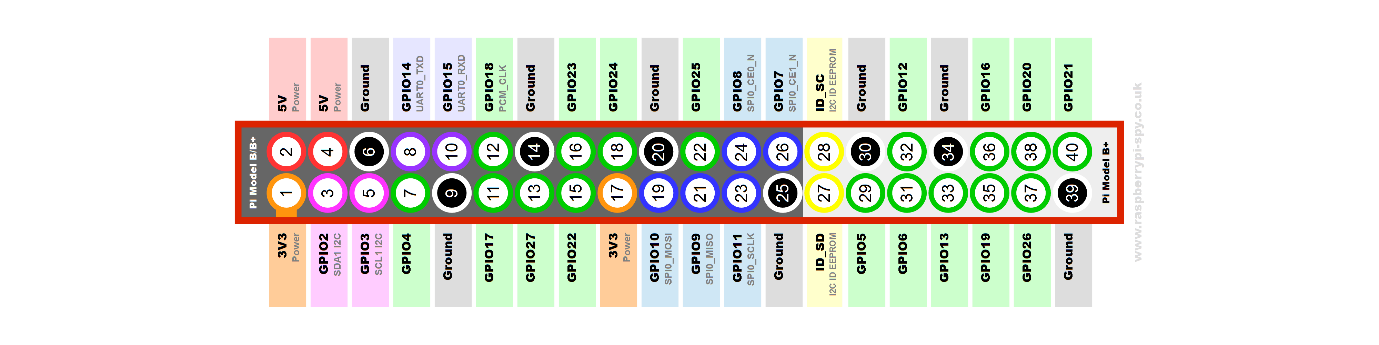


Image 8.2 The GPIO pin diagram for Raspberry Pi 2 Mod B

### Interfacing the Analog to Digital Converter (MCP3008)

To read the values from MQ-x sensors, we need an ADC. There are many possible options for an ADC but this section discusses the one used in the project: MCP3008. Image 8.2 shows the pin diagram for the MCP3008. The 8 in the end of the name refers to the number of input channels the IC supports. MCP3004 is also available, it has 4 input channels. The analog data pin of each of the sensor module is connected to any one of the input channels. The IC can support all 8 channels simultaneously with a 5V power supply which again can be sourced through the Raspberry Pi GPIO pin power source. The AGND and DGND pins are connected the ground reference of the Raspberry Pi similar to the sensors as discussed in the previous sub-section. The VREF pin depicts the reference voltage, which is the largest possible voltage that MCP3008 can interpret. Although VDD supplied in the setup is same as VREF at 5 Volts. The CLK pin is connected to the GPIO11 (Pin23), DOUT to the GPIO9 (Pin21), DIN to GPIO10 (Pin19) and finally CS is connected to GPIO8 (Pin 24).

Since it is a 10-bit ADC, the maximum value it can output is 210-1 which is numerically equal to 1023. The value given by the sensor is then processed by the node (Raspberry Pi) before pushing to the database to get the values in ppm. A big part of getting the values in ppm is the process of calibration which is discussed in the subsequent sub-section.

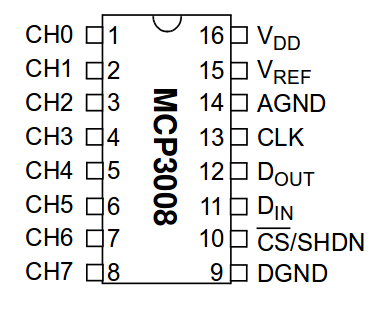


Image 8.3 Pin Diagram of ADC MCP3008

### Calibrating the MQ-x Sensors

Calibrating the sensors is an important as well as one of the most difficult part of the hardware setup. MQ-x sensors are majorly used for gas detection beyond a specific value and calibrating them to sense the real ppm value is a tough ask. The datasheet of the various MQ-x sensors depict a curve between RS/RO and ppm measure of the gases it can sense. The relationship is non-linear and plotted on a logarithmic scale. RS is the resistance of the sensor that can be calculated by the output that the sensor gives to the Raspberry Pi. RO is the sensor resistance at standard temperature and humidity conditions mentioned in the datasheet. The aim of the calibration process is to determine the RO value in order to get the correct ppm readings. Since the relationship is non-linear with no direct formula that can be used, the ppm measure of the gas to be sensed has to be approximated. First step involves determining any two points on the gas curve that has to be sensed and subsequently calculate the slope for it. Then once RO has been determined, the formula below can be used to approximate the ppm values.

It is advisable to sense only one gas through one sensor even when a sensor can sense multiple gases at a time. This is because presence of one gas can affect the reading taken for the second gas. The sensors get heated up when they sense the respective gases and offer a lower voltage output to the Raspberry Pi. Since this voltage is used to approximate the ppm values of other gases as well, change in one can directly affect the other leading to faulty values.

### Data Collection

## Web System Setup

### Meteor

Meteor is a javascript web framework. It is open source and free to use. It is built on Node JS. Meteor makes web development very fast and easy. The structure of meteor is a binding of various packages and libraries. Meteor has a big community supporting it; hence this offers developers a plethora of third party packages which they can integrate into their application. The meteor stack consist of Blaze as its front end and MongoDB (MiniMongo) as its database server. It uses tracker for maintaining its reactive front end nature. Much of the stack can be modified as needed, like, Angular can be used instead of Blaze for the front end part. One of the numerous advantages of using meteor is that it allows users to quickly deploy their app and produces cross platform code in Android, iOS and web. Hence meteor is perfect for rapid application development.

#### Setup

The installation of meteor is simple as it does not require any other software to be pre-installed for hosting, deploying, etc. For Microsoft Windows, meteor developers provide a windows installer which sets up the meteor directory. For Linux and OSX users, meteor can be installed directly from the terminal using the command : 'curl [https://install.meteor.com/](https://www.google.com/url?q=https%3A%2F%2Finstall.meteor.com%2F&sa=D&sntz=1&usg=AFQjCNHP5_YdJ4q8xH6vTbT5qGSTUFYZJQ) | sh'. Once meteor is setup, third party packages must be added for the code to work properly. The following command installs all the necessary packages required by meteor to process the code; 'meteor add d3 sergeyt:dimple templates:tabs richsilv:pikaday iron:router'. The meteor directory contains the html, css and javascript files of the project. It also contains a public folder which contains necessary images and javascript libraries for proper functioning of the meteor application. Once meteor is setup and all the necessary files are present, meteor can be deployed through an elevated command prompt or in the terminal by typing the command 'meteor' in the respective project folder. Meteor runs at port 3000. Further the MongoDB database created by the meteor application can be viewed and managed by the command 'meteor mongo'. The mongo server is hosted at port 3001 by default.

#### Front End

The AirCheQ web application consists of two views. The first view is the map view which consists of the AirCheQ logo at the top followed by the rest of the page filled by a google maps instance. The markers on the map are loaded by the result of an HTTP Get request web service hosted in the same server. This GET request returns the unique latitude and longitudes where the data is dispersed. Clicking on any one of the markers loads the second view of the web application: the data view. In this, the data for the respective latitude and longitude is neatly presented in the form of graphs. The data in the graph is queried with the respective date selected in the date picker field. The graph can be viewed either for the day or for the month, depending on the radio button selected. Hovering over a data point in the graph results in dotted lines being drawn from the data point to its x and y intercepts respectively. Clicking the back button takes back to the map view without any session errors.

#### Backend

Meteor uses MongoDB as its database technology. Mongo is very fast when compared to other database technologies. It is very easy to scale. One of the biggest advantages of mongo is that its stores its data in a schema-less fashion. Mongo is hugely preferred by startups as it is very easy to setup and provides excellent deep search speeds. The AirCheQ Web application sets up a MongoDB database 'meteor'. Inside a database, there are multiple collections, which are equivalent for tables. The 'meteor' database holds the collections: 'values' and 'counts', which are used in the web application to store and display data. The collection 'values' holds the gas sensor data, while the collection 'counts' is used to store a variable to maintain the active count of the entries in the 'values' collection. The object structure of entries in the collection 'values' is as follows: {[AutoGenerated]String \_id, [ScriptGenerated]in seq, String date, String time, String latlong, Float temp, Float hum, Float carbon-monoxide, Float carbon-dioxide, Float smoke, Float methane, Float LPG}.

The data is pushed to the server from a java program which collects sensor readings on a raspberry pi. The data is outputted in the meteor UI data view.

### Java Web Service

The Java Web Service serves dual functions. It not only secures the database call but also it helps in aggregating the data which has to be displayed on the UI. It sorts the data according to the given options, averages it and sends it to the Web UI in the form of JSON objects.

#### Setup and Structure

Setting up and developing the Java web service involves installing Java EE platform in the Netbeans IDE. A glassfish server also needs to be installed in order to deploy the web service. Make sure to use the latest version of the Netbeans while developing or else REFTful Jersey jar file will have to be imported in the library folder for the creation of the RESTful web service. The web-service has several GET methods, which can be accessed on any platform via an HTTP.GET request, provided the user has the web-service URL. Two of the methods in the web service accepts parameters during the time of the method call in the form of the URL extension.

For example [http://localhost:8080/AirCheQWeb/webresources/getJSON/month/{param}](http://localhost:8080/AirCheQWeb/webresources/getJSON/month/%7bparam%7d) is such an URL, where {param} is replaced by “2016-04-16XX 28.525564,77.571531”. The string before “XX” serves as the date in “yyyy-mm-dd” format and the string after “XX” serves as the latitude and longitude values in “latitude,longitude” format. After receiving the parameters, the web-service process the desired request and returns a JSON response, either as an object or as an array of objects.

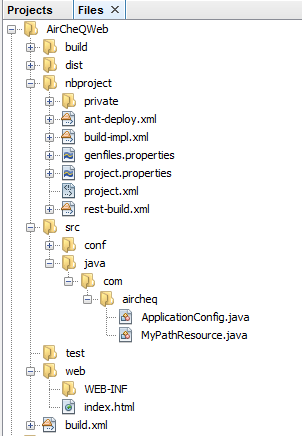


Image 8.4 Project Structure of the AirCheQWeb RESTful Java Web Service

# RISK ANALYSIS

The team, upon designing the system, examined various risks that the team might face in the future. Hence the team developed solutions to counter the same.

## Sensors

The project involves development of a system using numerous sensors that would be reading data for prolonged period of time. Hence it was essential that the team tested the prototype for long durations. The team tested the prototype for 1, 2, 4 and 8 hours respectively. The prototype worked perfectly with no heating issues whatsoever, and the accuracy unaffected.

## Wireless Communication

Since the system to be developed in the project would involve a raspberry pi sending data to a server, it was essential to test the wireless communication between them. The data was sent using TCP and hence would be secure and reliable. Although TCP handles connection problems well, the team devised various reconnecting techniques for stable functioning of the system.

## ADC Failover

The system to be developed in the project used the MCP3008 as the ADC of choice. The ADC worked as expected. The team had kept the Arduino as a backup ADC in case the first fails, wherein the inbuilt analog read from Arduino would have been interfaced to the Raspberry Pi.

## GSM/GPRS Module

The team initially thought of using a GSM/GPRS module with a GSM sim card in order to transfer the data from the Raspberry Pi to the DB. Although this was then replaced by a 3G dongle due to three reasons. GSM/GPRS module’s surface area was greater than the Raspberry Pi itself, making it inconvenient to carry around. The module also had to be powered by a 12V source which would reduce the time for which the setup can run by drawing greater power. It also offered slower GPRS speeds. Hence the idea of using it was dropped by the team.

# CONCLUSION

Upon extensive research and careful analysis, the team feels it is possible to design a fully functional IoT participatory data collection system within the financial and physical constraints. The team expects their fully working IOT system prototype, along with server and mobile application to scale smoothly to production. Although the sensors used during production would have to be better suited to sense gases more accurately as MQ-x sensors are more suitable for demonstration purposes. The whole setup runs smoothly with the configured hardware and software support.

There are various live projects in the concerned field but none too mature enough to be used by masses. The cost of such hardware needs to come down in order to get the majority audience to participate in the sensing process. Another possible solution to get penetration into the target audience would be the use of incentive schemes, be it monetary or non-monetary. Future projects could lay more focus on incentive schemes in such IoT based participatory sensing models to bring about a welcome change.

Definitely this field requires a lot of work to be done, both in technicality and marketing, till the time we get a mature enough platform which is accepted and widely used by the people themselves.

[Image 3.1 5](file:///C:\Users\Jitesh%20Aggarwal\Downloads\Draft.docx#_Toc448177255)

[Image 3.2 5](file:///C:\Users\Jitesh%20Aggarwal\Downloads\Draft.docx#_Toc448177256)

[Image 3.3 6](file:///C:\Users\Jitesh%20Aggarwal\Downloads\Draft.docx#_Toc448177257)

[Table 3.1 7](#_Toc448177258)

[Table 3.2 7](#_Toc448177259)

[Image 5.1 14](#_Toc448177260)

[Image 5. 2 A high-level diagram of the IoT setup used 19](#_Toc448177261)

[Table 6.1 20](#_Toc448177262)

[Image 6. 1 A basic hardware setup to read MQX sensor from channel 1 of ADC3008 and DHT22 readings directly from GPIO pins of Raspberry Pi 2 Mod B 22](#_Toc448177263)

[Image 8.1 The GPIO pin diagram for Raspberry Pi 2 Mod B 31](file:///C:\Users\Jitesh%20Aggarwal\Downloads\Draft.docx#_Toc448177264)

[Image 8.2 Pin Diagram of ADC MCP3008 32](#_Toc448177265)

[Image 8.3 Project Structure of the AirCheQWeb RESTful Java Web Service 34](#_Toc448177266)