FDI DATA ACQUISITION SYSTEM FOR GASEOUS POLLUTANTS MONITORING

BY

OFOEGBU OSITADINMA EDWARD

2011227004P

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

NNAMDI AZIKWE UNIVERSITY, AWKA

JULY, 2018

NNAMDI AZIKWE UNIVERSITY, AWKA ANAMBRA STATE.

DISCIPLINE & EXCELLENCE

SELF RELIANCE

FDI DATA ACQUISITION SYSTEM FOR GASEOUS POLLUTANTS MONITORING

SUBMITTED

BY

OFOEGBU OSITADINMA EDWARD

2011227004P

TO SCHOOL OF POSTGRADUATE STUDIES, NNAMDI AZIKWE UNIVERSITY, AWKA

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF DOCTOR OF PHILOSOPHY IN ELECTRONICS AND COMPUTER ENGINEERING

NNAMDI AZIKWE UNIVERSITY,

AWKA

JULY, 2018

CERTIFICATION PAGE

This is to certify that Ofoegbu Ositadinma Edward carried out this work in partial fulfillment for the award of Doctor of Philosophy Degree in Electronics and Computer Engineering from Nnamdi Azikwe University, Awka. The researcher affirms that this work has not been submitted elsewhere for the award of certificate, diploma or degree.

----------------------------	---------------------------	-------------------------
Student	Sign	Date

APPROVAL PAGE

This Dissertation has been read and approved, having met the requirements of the Department of Electronic and Computer Engineering for the award of Doctor of Philosophy (PhD) in Electronic and Computer Engineering.

………………………………… ……………………………….

Prof .H.C. Inyiama **Date**

(Supervisor)

…………………………… ………………………………..

Date

(External Examiner)

……………………………….. ………………………………..

Prof C.C Okezie **Date**

(HOD, ECE)

………………………………. ……………………………….

Prof V. E. Idigo **Date**

(Dean Faculty of Engineering)

…………………………. ……………………………….

(Dean School of Postgraduate) **Date**

DEDICATION

This doctoral dissertation is dedicated to God Almighty, the maker of the universe and the owner of my being who has sustained me during my earthly sojourns and by His grace was I able to successfully carry out this research.

ACKNOWLEDGEMENT

My immense gratitude goes to the Almighty God, for His mercies, favor and provision, who is the source and the giver of life, wisdom, understanding and knowledge, whose strength and enablement has been very crucial in the success of this dissertation.

In a special way, I want to thank my sincere and efficient supervisor, Professor H.C Inyiama for his understanding and perseverance in guiding and directing me through the period of this dissertation to ensure it was successful. God bless you sir.

Also, I really appreciate the support given to me by the head of department Prof C.C Okezie, Professor V.E Idigo, Dr C.Ohaneme and other members of staff of the Department of Electronics and Computer Engineering, also not forgetting my internal departmental and internal faculty examiners, Professor Okafor and Dr J.C Onuegbu for their candid assessments, support and encouragement.

My sincere appreciation also goes to my wonderful and loving parents – MR & MRS E.O OFOEGBU. Through thick and thin, you have never wavered in the discharge of your God given responsibilities. Your financial support, prayers and reminders to face my academics, while being there to look up to is something I greatly cherish. You are irreplaceable.

Also, I really appreciate the support given to me by my siblings. You are all blessings from God. I love you all.

A heartfelt appreciation also goes to my lovely wife, Mrs Ofoegbu Ann Onyinye, who has and always was by my side from the onset of this research pursuit till date. I do say a hearty thank you to everyone.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEFINITION OF TERMS

ECO- Electro-catalytic Oxidation

FTC- Fault Tolerant Control

FDI- Fault Detection and Isolation

UUV- Underwater Unmanned Vehicles

UAV- Unmanned Aerial Vehicles

MRAC- Model Reference Adaptive Control

LTI- Linear Time Invariant Systems

ICAP- Internal Configuration Access Port

RFID- Radio Frequency Identification

FLS- First Level Sensor

WSN- Wireless Sensor Network

PAN- Personal Area Network

JVM- Java Virtual Machine

AQMS- Air Quality Monitoring System

Q_o- Error Trend Factor

ACK- Acknowledgment

NACK – Not Acknowledged

FDI- Fault Detection and Isolation

DAS- Data Acquisition System

.

ABSTRACT

A Fault Detection and Isolation (FDI) data acquisition system for gaseous pollutants monitoring was developed in this dissertation. Missing measurement data due to sensor drop out faults, error in data measurement due to offset bias faults and variance degradation faults in a data acquisition system all pose serious challenges to overall system reliability when in a deployment. Thus a fault tolerant approach using redundant sensors and a supervisory controller deployed in a multi-tier data acquisition system architecture should detect the occurrence of these faults and have capability to isolate/accommodate the occurring fault. Additive and subtractive offset bias faults in data acquisition systems are majorly caused by environmental factors, while variance degradation faults are caused by ageing in sensors, which if unattended to would result in a drop out sensor fault. Thus, using a redundant sensor, gives room for switching between sensors in a given node when one gets old, and with the supervisory controller controlling environmental factors which could cause an offset in sensor reading. The FDI data acquisition system has a multi-tier architecture comprising of a data acquisition system unit, a central remote terminal unit and a data analytics unit. An FDI mathematical model based on data variance and deviation was implemented in the central terminal unit to detect the occurring faults, while feedback information of these faults was sent back to the data acquisition unit, whose fault isolation/accommodation behavior was modeled using an error residual generator. An Air Quality Monitoring System application (AQMS) provided the tool for the data analytics phase of the study. The AQMS application allowed for trend analysis, where results gotten from pre-analysis by the central remote terminal unit and post –analysis of the supervisory controller using the developed FDI, showed that CO and NO measurements of (40, 120) and (187,110) received on the 12/09/2014, were shown to be below the CO sensor lower threshold of 75mg/m^3 and above the NO sensor upper threshold of 107mg/m^3, thus indicating offset bias faults and data variance faults respectively. Data received afterwards didn't indicate a continuous occurrence of the fault situation. Data trend analysis for multiple hours, revealed a change in concentration of CO by $values < \pm 16.64$mg/m^3 and NO by $values < \pm 9.14$mg/m^3, where values outside this range indicate a fault situation. This research offered a simplistic approach to fault tolerant deployment with fewer components as opposed to the popular triple modular redundancy (TMR) methods. The new system can be applied to any fixed process application for pollution study of gaseous pollutants as it offers a more reliable operation than a generic data acquisition system.

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

Pollution is the introduction of contaminants into the natural environment that causes adverse changes to it. Where, such pollutants could take the form of a chemical substance or energy. Pollution is often classified as point source or non-point source (fenterstock, kurtzweg, & Ozolins, 1971) .

Air pollution has always accompanied civilizations and dates back to prehistoric times, when man first created the first fires. Environmental pollution was a direct result of the industrial revolution, where the industrial revolution era caused the emergence of great factories and increased consumption of immense quantities of coal and other fossil fuels (David, 1996). The resulting air pollution coupled with large volumes of industrial chemical discharges which when added to the growing load of untreated human waste constituted environmental pollution. Gaseous Pollutants from combustible processes have remained a major source of concern with respect to environmental pollution (Farrah, Mateen, & Brook, 2011) and this has fuelled the desire to monitor/measure these pollutants in order to determine their effects to the environment at large.

Data acquisition is the process of sampling signals that measure real world physical conditions and thus converts the resulting samples into digital numeric values that can be manipulated by a computer (Batra, 2005), its major components include sensors for converting physical parameters to electrical signals, signal conditioning circuitry for converting sensor signals into a form that can be converted to digital values and converters which could be Analog to Digital converters(ADC) or Digital to Analog converters (DAC). data acquisition applications are usually controlled by programs developed using various general purpose programming languages as well as open source software packages which provide necessary tools to acquire data from different hardware equipment (Mylopoulous, Lobcopolous, & Zicari, 2010). Diverse physical parameters can be measured e.g. gaseous pollutants in an air pollution study, temperature, light intensity, gas pressure, fluid flow and force etc. DAQ cards which act as hardware interfaces between sensors and the computer could contain multiplexers, ADC, DAC, high speed timers' etc. generally, these are interfaced to a microcontroller which runs small programs.

Modern technological systems such as data acquisition systems, now rely heavily on smart control system algorithms to meet increased performance and reliability requirements (Starowiecki, 2011)and this is particularly true in critical infrastructures and applications, where a minor or often benign fault could potentially develop into catastrophic events and inaccurate measurements, if left unattended or incorrectly responded to.

In spite of many advances and new concepts in the field of systems and controls design, complex systems sometimes do not render the services they were designed for, or they simply run out of control thus creating situations whose effects range from inaccurate measurements, energy wastage, damage to the environment and damage to human health.

Therefore to prevent a fault induced loss or performance drop while minimizing the potential risk resulting as an effect of faults in deployed systems, control techniques and design approaches need to be developed to cope with system component malfunction. these malfunctions are in themselves fault conditions resulting from many possible causes which can be classified into design errors, implementation errors, human operator errors, wear, ageing and environmental degradation (Alwi, Edwards, & Pin Tan, 2011) (Starowiecki, 2011).

An optimal control system which possesses the capability described above is often known as a fault tolerant control system (Jiang, 2014). Therefore fault tolerance is a concept comprising of Fault Tolerant Control (FTC) and Fault Detection and Isolation (FDI) which generally implies that when a fault occurs in a system, the main problem to be addressed is to ideally diagnose what fault has occurred and then decide how to deal with it, such that the problem of detecting a fault, finding the source/location and then taking appropriate action is the basis of fault tolerant control (Mahmood, Jiang, & Zhang, 2003).

1.2 Problem Statement

The issue of global warming has become a matter of great concern globally. Climate change is evident in our daily lives as floods, drought, natural disasters and rising ocean level as a result of the melting ice cap in the polar region all threaten to destroy civilization (Zoidis, 1996). Nations have identified this problem and its effects which are heavily linked to pollution. An integral issue to the success of any pollution control plan is to have a monitoring system that is informative, timely, fault tolerant and precise to enable regulating agencies determine pollution levels of processes whose operation contribute to air pollution, and for the operators themselves to have information relating to their pollution levels. All these go to ensure best practices are maintained before the consequences of our actions in the name of civilization and industrialization destroys the earth. Adverse air quality can kill many organisms not just humans, ozone pollution causes respiratory disease, cardiovascular disease, throat inflammation, chest pain and congestion etc (Spergler & Sexton, 1983) (Goldstein, Koven, & Heald, 2009). Water pollution has been attributed as the single cause of over 14,000 deaths per day mostly due to contamination of drinking water by untreated sewage in developing countries (Roman, 2010). Oil spills also can cause health problems such as skin irritation and rashes. Noise pollution induces hearing loss, high blood pressure, stress and sleep disturbance. Mercury poisoning, lead and other heavy metal ingestion have been identified as a leading cause of developmental deficiencies in children as well as other neurological problems (Spergler & Sexton, 1983) (Roman, 2010).

Carbon dioxide though vital for photosynthesis is sometimes referred to as a pollutant because raised levels of the gas in the atmosphere affects the earth's climate. Disruptions to the environment can also highlight the connection between areas of pollution that would normally be classified separately such as those of water and air. Recent studies have investigated the potential for long term rising levels of atmospheric carbon dioxide to cause slight but critical increases in the acidity of ocean waters and its possible effects on marine ecosystems (Duflo, Greenstone, & Hanna, 2008).

Thus improving the reliability of deployed data acquisition systems to pollution studies, would be highly welcome as it helps mitigate the effects of faulty systems (System component faults) and the resulting inaccurate/missing data used for analysis.

1.3 Aim and Objectives

The aim of this research work is to develop a Fault Detection and Isolation (FDI) Data Acquisition System for gaseous pollutants monitoring.

1.3.1 Objectives

In order to achieve this aim, the following objectives are pursued:

1. To develop a physical model for an FDI data acquisition and supervisory control unit.
2. To develop a fault detection algorithm using standard deviation and data variance in a remote terminal unit.
3. To develop an error-residual generator - based model for fault isolation and fault accommodation.
4. To perform data analysis and model validation using an Air Quality Management System (AQMS) software.

1.4 Scope of the Research

The dissertation is limited to the development of a fault detection and isolation (FDI) data acquisition system for gaseous pollutant monitoring. The FDI-DAS is capable of detecting and accommodating only two types of faults, namely off-set bias sensor fault (calibration offset) and variance degradation sensor fault. Standard data acquisition system architecture will be modified with the use of redundancies in order to make it fault tolerant to the identified faults of interest. The multi-tier architecture fault tolerant data acquisition system will be deployed to measure gaseous pollutants in an industrial location (Ife steel mill), with data measurement and subsequent data analysis proving the system's tolerance to variance degradation and offset bias faults.

1.5 Significance of the Study

The world we live in has evolved through many ages and periods. It has experienced a period of stability with regards its climate, which many generations of humans have gotten accustomed to. The aftermath of industrialization and nuclear studies was a change in climate, where the rainy/dry seasons or winter/summer seasons became uneven and in some cases drought and floods ravaged certain areas of the earth. Increases in sea water level as a direct result of the melting ice caps has thus threatened coastal areas and there are in fact fears that if such phenomenon isn't contained countries such as the Netherlands and much of Scandinavia which lie below sea level could be submerged. Studies have also shown that the climate change phenomenon has been liable for these changes we have been experiencing. It became imperative to discuss what caused the climate change in the first place. Air pollution which results from a direct release of green house gasses into the atmosphere depletes the ozone layer. This destruction of the earth's own immunity to unwanted rays from the sun has resulted in sunny days being much hotter than they were 100years ago. The fact that the day is much hotter not only causes discomfort for people in sunny and arid regions of the world, it melts the ice in the polar region specifically Antarctica, resulting in an increase in sea water levels. The hotter weather causes evaporation of more water from the sea, thus making the rainfall heavier, which leads to inland flooding. This unwanted gas emissions are products of numerous processes. Industrialization is the bane of modern development and cannot be substituted but resultant emissions can be reduced through better practices by the industries. It therefore highlights the need for efficient pollution monitoring systems and pollution control schemes. The aim of this dissertation is to develop a fault detection and isolation data acquisition system for industries to measure gaseous pollutant concentration within an industrial complex. The measured data is reported to a monitoring agency whose job is to ensure that industries are doing their utmost to cut down on emissions of unwanted air pollutants which are harmful to the environment.

1.6 Motivation

The accurate measurement of physical parameters is key to proper monitoring and control of larger processes. Fault tolerant control offers extensive techniques that could improve the robustness of any system to which it is applied. Fault tolerant data acquisition systems would provide less instances of missing data in a data set, ensure the possibility of occurrence of valid data in a data set of measurements and also provide a platform for machine to machine communication (feedback) ,which is aimed towards improving the overall system performance. The use of a fault tolerant data acquisition system in real life monitoring will result in longer system deployment needing less maintenance than its generic counterpart.

CHAPTER TWO

LITERATURE REVIEW

There are many systems and approaches employed for air pollution and environment monitoring. Some have been deployed at various scales such as countrywide monitoring, locality monitoring and general air quality monitoring. Some of these have resulted into commercial products or experimental research platforms. Most of the researches carried out belong to the following categories:

- Fault Tolerant Control
- Industrial Air Pollution Monitoring

2.1 Fault Tolerant Control

It has been described earlier as a control system that possesses the capability to cope with system component malfunctions, while maintaining a desirable degree of overall system stability and performance level. The key to any fault tolerant control system is the existence of system redundancies and different design methods which are a reflection of different philosophies in utilizing and managing such redundancies (Edwards & Pin Tan, 2006). Redundancies can be in hardware or analytical forms such as redundant sensors/actuators or fault detection/diagnosis schemes, such that a properly designed fault tolerant control system should operate satisfactorily not only in the absence but also in the presence of system faults.

2.1.2 Fault Detection and Isolation

FDI is a major component part of fault tolerant control systems, it is that part whose job is to detect and locate the source of the occurring fault condition. fault detection and isolation is present in active fault tolerant control systems but not in passive fault tolerant control systems simply that in the active fault tolerant control systems, the main function of the FDI is to detect a fault or failure, thus localizing it to enable corrective action to be implemented in order to eliminate or minimize the effect on the overall system performance (Edwards & Pin Tan, 2006) (Paton, 1997).

FDI can be classified in two ways or approaches namely:

- The model based FDI scheme
- The non-model based FDI scheme

2.1.3 Model Based FDI Scheme

The model based fault detection and isolation schemes has two categories namely fault detection and isolation using residual schemes (Edwards & Pin Tan, 2006) and fault detection and isolation that has the capability to estimate the fault.

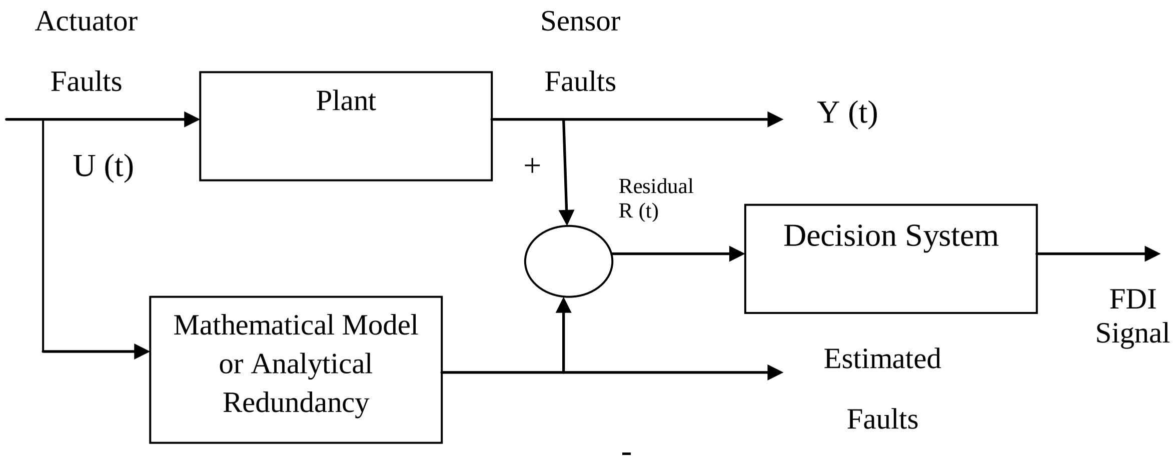


Figure 2.1: Residual-Based FDI (Edwards & Pin Tan, 2006).

Where, from the figure above representing the residual based FDI, signals from a mathematical model and hardware measurements are compared and the filter difference forms a residual signal. In nominal fault free conditions, the residuals should be zero and non zero when faults/failures occur, this residual signal is usually applied with a threshold to avoid false alarm from disturbances or uncertainties.

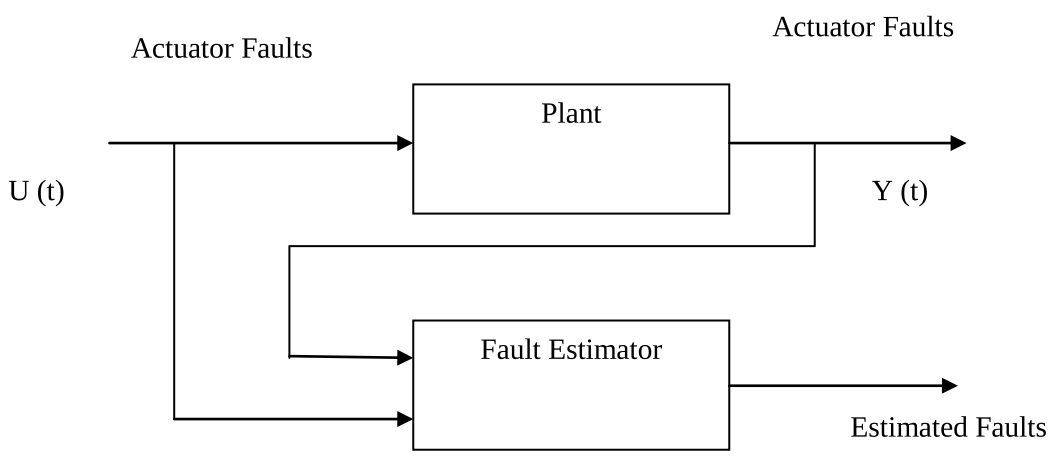


Figure 2.2: Fault Estimation Based FDI (Paton, 1997), (Edwards & Pin Tan, 2006).

The kalman filter falls under this estimation based fault detection and isolation scheme. It was conceived in the 1960 by Rudolf kalman and made famous by its application in the National Aeronautics and Space Administration (NASA) Apollo space program (Paton, 1997).it implements parameter estimation schemes, which provides a means of updating the system parameters online in real time for controller reconfiguration, and this has been applied largely to aircrafts.

2.1.4 Non-Model based FDI Scheme

The non-model based fault detection and isolation scheme typically involves schemes utilizing artificial intelligence and self computing approaches such as neural networks and fuzzy logic. The underlying concept is to structure the neural network in a fuzzy logic format, which allows residual generation through rapid and correct training of the neural network to model the non-linear dynamics of the system, and evaluation and diagnosis of the fault through fuzzy logic. This scheme has the benefit in its ability to model any non-linear function (Paton, 1997), (Edwards & Pin Tan, 2006) and (fekih, 2006).

2.1.5 Passive Fault Tolerant Control (FTC)

All efforts on passive fault tolerant control are mainly concentrated on not so reliable controllers to achieve a reliable control system. Thus, the redundancy lies in the employment of multiple controllers and actuators. The passive fault tolerant control system might sometimes rely on a fixed controller for all conceivable situations.

Decentralized state feedback approaches are used to synthesize these multiple controllers and thus naturally show that the reliability of the resulting control system was improved greatly as opposed to using an individual controller (Paton, 1997), (fekih, 2006). Hence fault tolerance was achieved resulting in two very important concepts namely

- Redundancy
- Decentralized control (or distributed control)

A necessary condition for using a multiple controller structure is that the system should possess multiple actuators, thus a multi-input system (Ahmad, 2011). Theoretically diverse existing control system design techniques such as linear quadratic optimal control, transfer function matrix approach, matrix lyapunov equation etc have all been applied to the passive control system to good measure (Mogens, Marcel, & Eva, 2001). Passive fault tolerant control is also known as “reliable control” where the passive fault tolerant control system design problem becomes to synthesize a controller so that the closed loop system is stable for any combination of the failure elements (Mogens, Marcel, & Eva, 2001).

2.1.6 Active Fault Tolerant Control

Research in active fault tolerant control system was motivated by flight control systems for aircraft as a safety critical system, whose main objective was to incorporate some “self repairing” capabilities into the on-board flight control system so that the aircraft can make a safe landing in the event of component failures. The active fault tolerant control system unlike the passive fault tolerant control system reacts to diagnosed faults/failures by exercising the controls accordingly through proper manipulation of redundancies, so that the system stability can be maintained and its performance still acceptable. In many circumstances a compromise has to be made to accept a degraded performance in the presence of failure due to limited amount of redundancies (Paton, 1997).

Historically two events encouraged the development of active systems. The first case was the delta airline flight 1080(on April 22, 1977) (fekih, 2006), one of the elevators became jammed at 19 degrees up and this malfunction was unknown to the pilot at the time. Fortunately based on his flying experience and the availability of other redundant actuators on this L-1011 aircraft, the pilot successfully reconfigured the remaining control elements and landed safely. The second case was the ill fated American airline DC-10 (Flight 191, on May 25, 1979) (fekih, 2006) where post accident analysis has indicated that the pilot had about 15seconds to react to the failure for which if corrective measures were put in place the situation would have been saved. The importance of having an active fault tolerant control system cannot be overemphasized especially in military aircrafts for dealing with battle damages over enemy territory (Cole & Keogh, 2012).

Therefore in contrast, passive fault tolerant control systems could sometimes rely on a fixed controller for all conceivable situations while an active fault tolerant control system reacts to diagnosed failures by exercising the controls accordingly through proper manipulation of redundancies, so that the system stability can be maintained and the performance still acceptable.

An active fault tolerant control system comprises of the following subsystems

- Fault Detection and Diagnosis scheme
- Controller Reconfiguration Mechanism
- Reconfigurable Controller

All these subsystems have to work in harmony within the real-time constraints to achieve optimal control. The reconfigurable controller in active fault tolerant control is usually a digital controller whose parameters can be easily varied as directed by the controller reconfiguration mechanism.

2.1.7 Hybrid Fault Tolerant Control

The bottleneck in any active fault tolerant control is the real time fault detection/diagnosis scheme, since it has to operate in a real time environment with a limited number of measurements which are often corrupted by noise. It is quite possible that a wrong decision could be made. While the bottlenecks in the passive fault tolerant control is the lack of robustness it has, as well as its scalability issues. Therefore an optimal solution to the bottleneck of the active FTCS would be to employ a passive FTCS to provide the stability cushion and then using an active FTCS to further improve the system performance in the same application such that the combination of the two would produce what is known as the hybrid fault tolerant control which if properly designed will be able to provide both stability and optimal performance (Neville, Kam, & Myron, 2006).The mechanisms for obtaining fault tolerance for different types of system ranging from conceptually simple modular redundancy schemes to more advanced model based fault diagnosis techniques require a rigorous theory of security in control systems as components of a system can fail due to a variety of factors ranging from wear and tear, adverse conditions, accidents or targeted attacks etc. Therefore incorporating fault tolerance into the system can allow the system degrade gracefully, while functioning under failures or could prevent faults from propagating to other parts of the system. Fault tolerance here can be broken down into two objectives namely

- Fault detection and identification (FDI): which determines whether a fault has occurred, thus isolating the failed component.
- Fault Accommodation: taking steps to correct the fault or reconfigure the system to avoid the faulty component entirely.

Requirement analysis for a fault tolerant system is regarded as the first stage for the development of fault tolerant control systems where any combination of redundancy methods is perceived as the key and way forward in optimal fault tolerant control (Cole & Keogh, 2012)

The pre-emptive/proactive actions embedded in control systems to improve reliability especially in critical processes has been predefined with this technology being applied in various fields and proposed as solutions to problems in safety critical applications. Fault detection and isolation, fault estimation, fault tolerance, fault accommodation and system reconfiguration as applied to robotics and automated control systems were discussed in (Neville, Kam, & Myron, 2006). The main approaches and concepts used for the design of fault tolerant systems were introduced, its impacts were analyzed and from the level of knowledge provided by the fault detection and isolation algorithm a clear distinction between system reconfiguration and fault accommodation was established such that when no solution exists to the problems it became necessary to change the system objectives.

Two control problems were investigated in that study namely model matching and optimal control, which detailed the conditions and means via which they can receive fault tolerant solutions. The original motivations in the use of redundancies showed that over reliance by modern technological systems on sophisticated control systems to meet increased safety and performance requirements particularly in safety critical applications such as aircraft, spacecraft, nuclear power plants etc cannot be overemphasized as a minor or often benign fault in these systems could potentially develop into catastrophic events if left unattended or incorrectly responded to (Tushar, Joseph, & Dominque, 2011).   The key to any FTCS is the existence of system redundancies, such that different design methods would be merely the reflection of diverse philosophies in utilizing and managing such redundancies.

A fault tolerant control system for application in unmanned underwater vehicles using the URRG-ROV test bed to focus on faults in the propulsion system was developed into two parts, (Sghair, Bonneral, Crowzet, & Brot, 2007) The first part focused on the detection and diagnosis of the thruster due to events of failure, such as a blocked ducting and power supply malfunction and the second part being a statistical method of design using experimental techniques to construct and analyze future data. The implementation of the fault tolerant control system in real time unmanned underwater vehicle operations showed during testing in manual maneuvering (open loop control) and auto pilot maneuvering(closed loop control) to have operated optimally (Sghair, Bonneral, Crowzet, & Brot, 2007).

A fault tolerant control strategy for actuator faults using LPV techniques was proposed in (Yixin & Passino, 2001) and applied to a two degree of freedom helicopter. The linear parameter varying FTC method was employed to adapt the faulty plant to the controller and not the controller to the faulty plant. This approach can be seen as a kind of virtual actuator, where an integrated design procedure for the fault identification and fault tolerant control schemes using LPV was presented. The FTC controller was implemented as a state feedback controller and designed using polytopic LPV techniques and linear matrix inequality (LMI) regions in such a way as to guarantee the closed loop behaviors in terms of several LMI constraints.

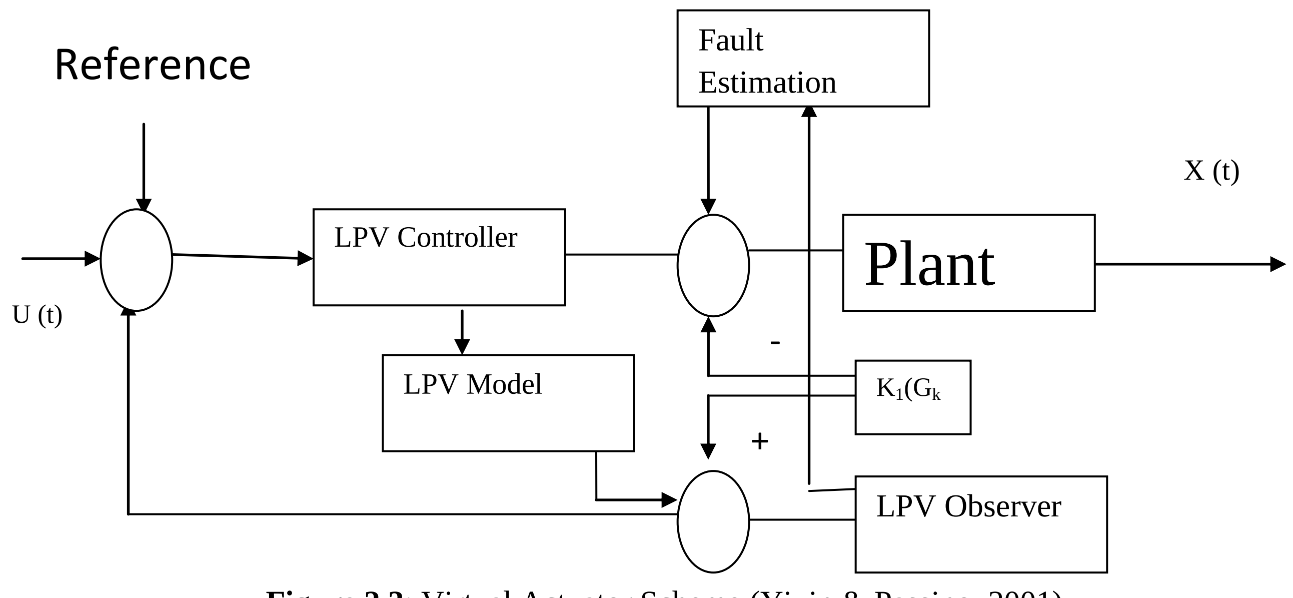


Figure 2.3: Virtual Actuator Scheme (Yixin & Passino, 2001).

The virtual actuator developed above, exhibited stabilization problems which were solved using a dynamic neural network trained for the fault trajectory, in order to determine the identity matrix in its discrete time representation, thus resulting in a negligible one-time step delay.

An adaptive approach to active fault tolerant control where faults or process failures acting or occurring in dynamic systems are estimated and compensated within an adaptive control scheme with required stability and performance robustness was developed in (Heiner, Collias, & Wirthin, 2007). A fault tolerant controller design with application in power systems and synthetic biology for linear time-invariant systems (LTI) with multiple actuators was discussed in (Jiang, 2014). The study based on the fact that every combination of actuators can fail as long as the set of the remaining actuators includes one of the failure subsets. Therefore to achieve stability under all permissible sets of faults and also better performance after clearing every subset of the existing faults in the system, it was shown that a state feedback controller that satisfies the above properties can be obtained. The resulting LMI condition can then be transformed into an optimal control condition. A hypothetical state system of

$$x(t) = Ax(t) + Bu(t) \qquad 2.1$$

Was considered where A is a 4*4 matrix whose entries where sampled and B is a 4*3 matrix and X (0) is a random variable of zero mean with an identity covariance matrix. The controller was thus designed so that

$$u(t) = Gx(t) \qquad 2.2$$

Where G is the transfer function and could also be expressed as

$$G = \sum \int \big(x(t) + U(t) \big) dt \qquad 2.3$$

A hybrid fault tolerant model for predictive control within the hybrid system framework was proposed in (Hassan, Dominque, Frederic, & Dieder, 2000). This method was applied to sewer networks and used a mixed logical dynamic form to represent a hybrid system, such that the hybrid model could be obtained, which would include inherent hybrid phenomena and possible modes caused by a fault occurrence. The proposed solution allows online adaptation of the system model while taking into account the fault information provided by a fault diagnosis and isolation module. The controller can therefore cope with the considered faults, while additionally implementing different schemes and fault tolerant evaluation procedures for the hybrid model in considering fault tolerant capabilities for the sewer network

Integrated fault tolerant robotic control systems for high reliability and safety were discussed in (Ocampo-Martinez & Vicenc, 2011). The system was developed for applications that require high dependability (reliability, availability and safety). It consisted of a fault tolerant controller and an operational work station, where the fault tolerant controller uses a strategy which allows for detection and recovery of hardware, operating system and application software. Protection against higher level unsafe events was provided by a software resident in a separate operator workstation, which included features to predict collisions and reduce human workload, thereby reducing errors and enhancing safety. The developed fault tolerant controller can be used by itself in a wide range of applications in industry, process control, and communication, while the controller combined with the operator workstation, can be applied to robotic applications such as space borne extravehicular activities, hazardous material handling e.t.c. every other task where a robotic system failure poses a significant risk to life or property is also an applicable area.

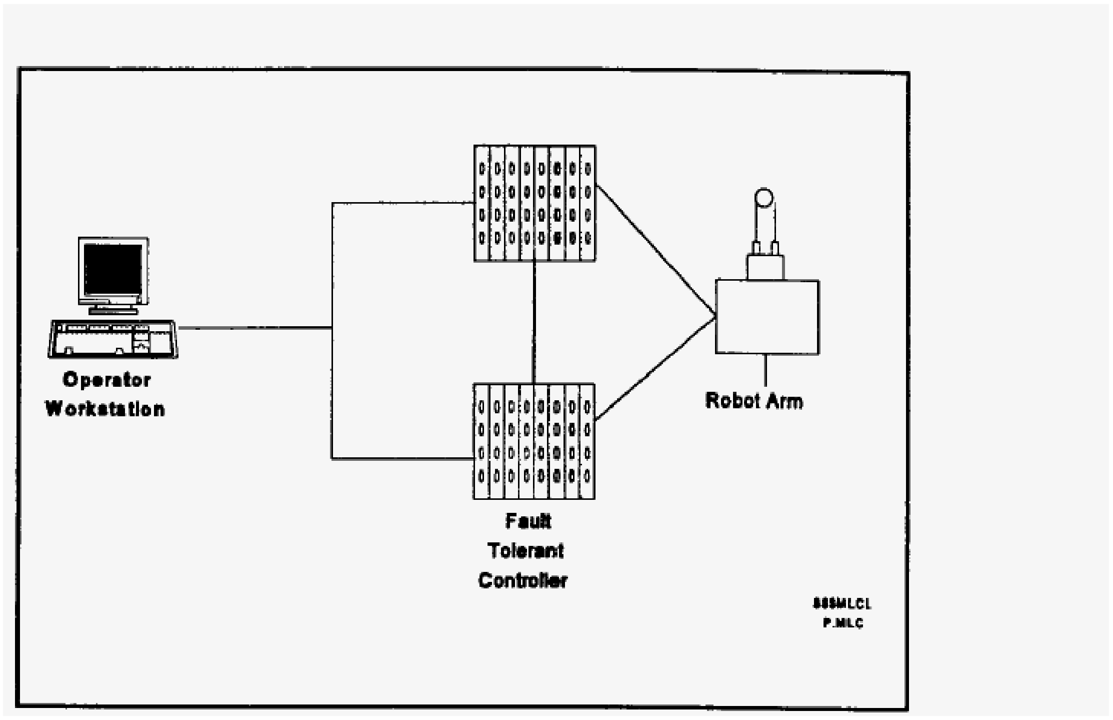


Figure 2.4: Top Level System Fault Tolerant Design (Ocampo-Martinez & Vicenc, 2011).

The solution for a closed loop fault tolerant control for uncertain nonlinear systems was based on an algebraic estimation technique of the derivatives of a time signal, which yields good estimates of the unknown parameters and of the residuals (Mohammed, Mohammed, & Ahmed, 2004) i.e. fault indicators was easily implementable in real time while being robust with respect to a large variety of noises. Numerical simulations were provided for a popular case-study in the diagnosis community which is the three-tank system, which may be characterized as a flat hybrid system.

It was demonstrated in (Pimentel & Salazar, 2011) that a performance based supervisory approach to achieve fault tolerance, does not require any explicit fault-diagnosis module using a novel switching logic.

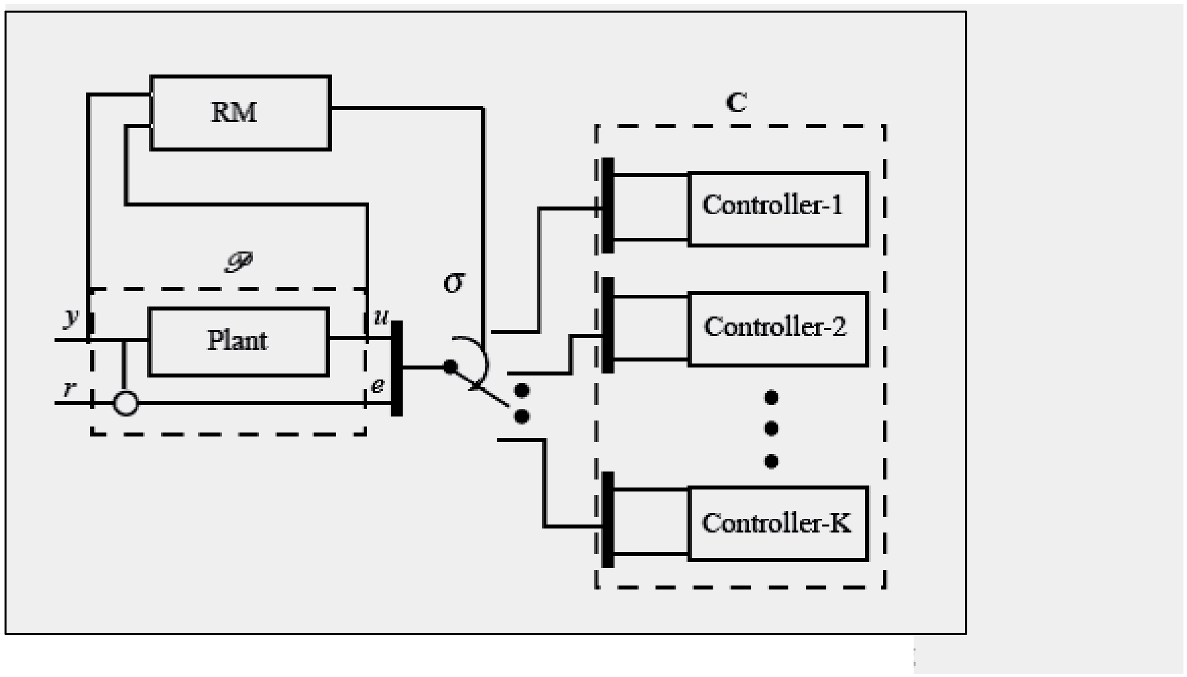


Figure 2.5: Switching Control Scheme for FTC in Behavioral Setting (Pimentel & Salazar, 2011).

2.2 Industrial Air Pollution and Pollutants

A substance in air that has adverse effect on human and environment is known as an air pollutant and these pollutants as earlier mentioned can be in the form of solid particles, liquid droplets or gases (Agajo & Inyiama, 2011). Pollutants are classified as either primary or secondary (Harish, Prahbur, Gadh, & Asad, 2007), where primary pollutants are directly produced from a process such as volcanic ash eruption, the carbon monoxide gas from a motor vehicle exhaust or nitrogen and sulphur oxides released from factories. Secondary pollutants are not emitted directly rather they form in the air when primary pollutants react or interact e.g ground level ozone and photochemical smog.

Primary pollutants produced by human activity include:

Sulphur Oxides (SO_x)

Sulphur dioxide, a chemical compound with the formula SO_x is produced by volcanoes and by various industrial processes such as power generation plants, coal incinerators, some industrial burners etc. Since coal and petroleum often contain sulphur compounds, their combustion generate SO_2 were further oxidation usually in the presence of a catalyst such as NO_2 forms H_2SO_4 , this results in acid rain, hence the concern over the environmental impact of the use of these fuels as power sources (Harish, Prahbur, Gadh, & Asad, 2007).

Nitrogen Oxides (NO_x)

The most prominent member of this family is the nitrogen oxide. This is expelled from high temperature combustion and is also produced naturally during thunderstorms by electric discharge. It can be seen over towns and cities as a brown haze dome. NO_2 is one of the most prominent air pollutants (Harish, Prahbur, Gadh, & Asad, 2007), (Markovic, Stanimovic, & Stoimenor, 2009).

Carbon Monoxides (CO)

This is a colorless, odorless, non-irritating but very poisonous gas. It is a product of incomplete combustion of fuel such as natural gas, coal or wood. Vehicular exhaust is a major source of carbon monoxide (Harish, Prahbur, Gadh, & Asad, 2007) .

Volatile Organic Compounds (VOC's)

This is an important outdoor air pollutant. They are divided into methane and non-methane family groups. The methane family derivates are extremely efficient green house gases which contribute to global warming (Markovic, Stanimovic, & Stoimenor, 2009).

Particulates

This is alternatively referred to as particulate matter (PM). Atmospheric particulate matter or fire particles are tiny particles or liquids suspended in gas e.g aerosol (Agajo & Inyiama, 2011).

We have many more as mentioned earlier such as toxic metals, CFC's, Ammonia, odors, radioactive pollutants which all constitute primary pollutants. For the secondary pollutants we have smog, ultraviolet light, ground level ozone, peroxyacetyl nitrate(PAN) (Agajo & Inyiama, 2011).

2.3 Review of Existing Pollution Monitoring System

A mobile air quality monitoring network (MAQUMON) which consisted of car mounted sensor nodes, measuring different pollutants in the air with the data points tagged with location and time with an on-board GPS. The periodic measurements made was updated to a server, and then processed before it being published on a sensor map portal (Arici & Altunbasak, 2003).

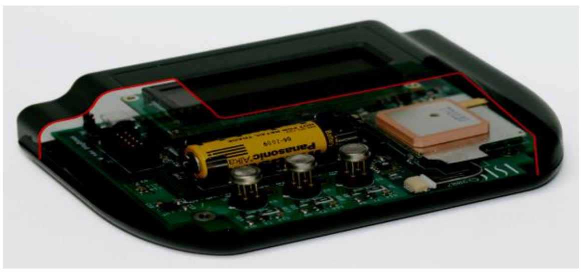


Figure 2.6: Sensor Node Prototype Renamed the MAQUMON Node (Arici & Altunbasak, 2003).

The deployment of wireless sensors for air quality monitoring in India was described in (Volgyesi, Nadas, Koutsoukos, & Ledeczi, 2008) ,where an environmental air pollution monitoring system that measures respirable suspended particulate matter, NO_x, SO_2 was designed with air quality index (AQI) reporting capabilities. Several sensor nodes which measure pollutant information were uniformly deployed in the networks, thus creating a sensing phenomenon. Low power strategies and hierarchical routing protocol were used to cause the motes to sleep during idle time to conserve power utilization.

Investigations on the use of wireless sensor networks for air pollution monitoring in Mauritius was described in (Zhao, 2011). Where an innovative system named the wireless sensor network air pollution monitoring system (WAPMS) was proposed to monitor air pollution in the island through the use of wireless sensors deployed in huge numbers. The system made use of an air quality index reporting capability in order to improve the efficiency of WAPMS. Data aggregation algorithm called recursive converging qualities (RCQ) was designed and implemented to enable the merging of data. The elimination and summary of these readings into a simpler form significantly reduced the amount of data to be transmitted to the sink. Hierarchal routing protocol was used in the WAPMS which caused the motes to also sleep during idle time.

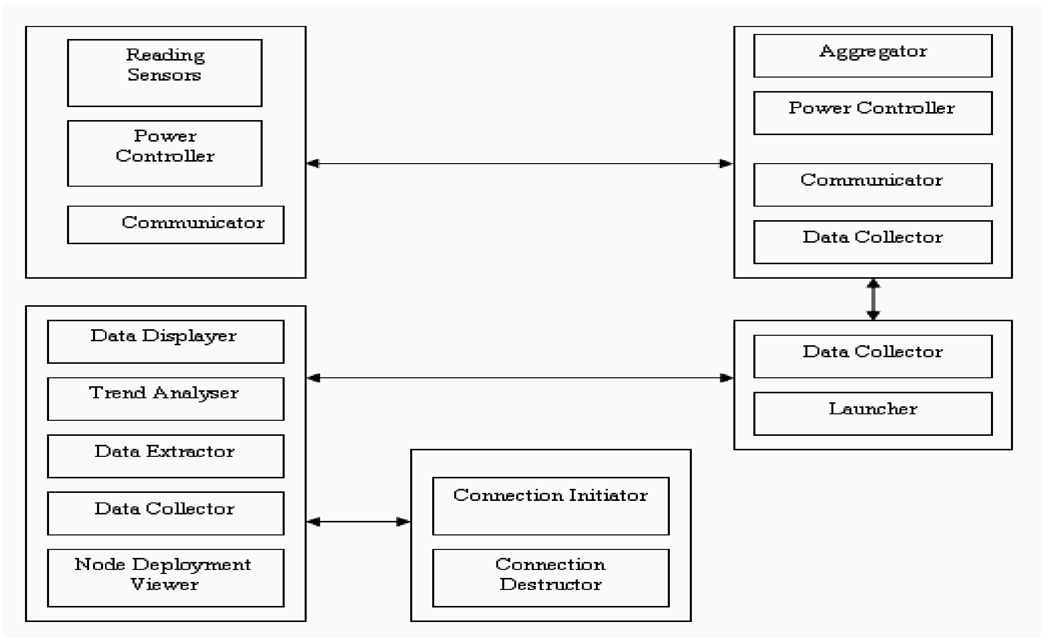


Figure 2.7: Architecture of a wireless sensor network air pollution monitoring system (WAPMS) (Zhao, 2011).

Air pollution monitoring systems, thus involves a context model and a flexible data acquisition policy (Mishra, Dhanashare, & Asutkar, 2011). The context model developed to prove the theory was also used to analyze the status of air pollution at any remote location, in order to provide an alarm or safety guideline depending on the condition of the context model. A flexible sampling interval change occurred for effective tradeoff between sampling rates and battery lifetimes. This interval change saved the batteries of the geo-sensors by reducing the number of data transmissions.

(Khedo, Perseedoss, & Mungur, 2010), simulated three air pollutant gases (carbon monoxide, carbon dioxide and sulphur dioxide) in air because these gases decide the degree of pollution levels. The pollution mode was designed by integrating the sensor associating circuitry, a low power microcontroller and Xbee communication module were integrated to perform functionalities such as signal conditioning, sensing changes in air, signal amplification and signal calibrations. The results showed the pollution concentrations in air.

(Young, Yang, Dongs, Keun, & Nittel, 2005), Developed a solution for pollution monitoring using wireless sensor networks on real time basis called the real time wireless air pollution monitoring system. Commercially available CO_2, NO_2, CO and O_2 sensors were calibrated using appropriate calibration technologies. The pre-calibrated gas sensors were integrated with the sensor motes and deployed in the field at Hyderabad city campus, using the multi hop data aggregation algorithm. A light weight middleware and web interface was developed to view the live pollution data in the form of numbers and charts from the test beds anywhere on the internet. Parameters such as temperature and humidity were also sensed along with the gas concentrations to enable data analysis through fusion techniques.

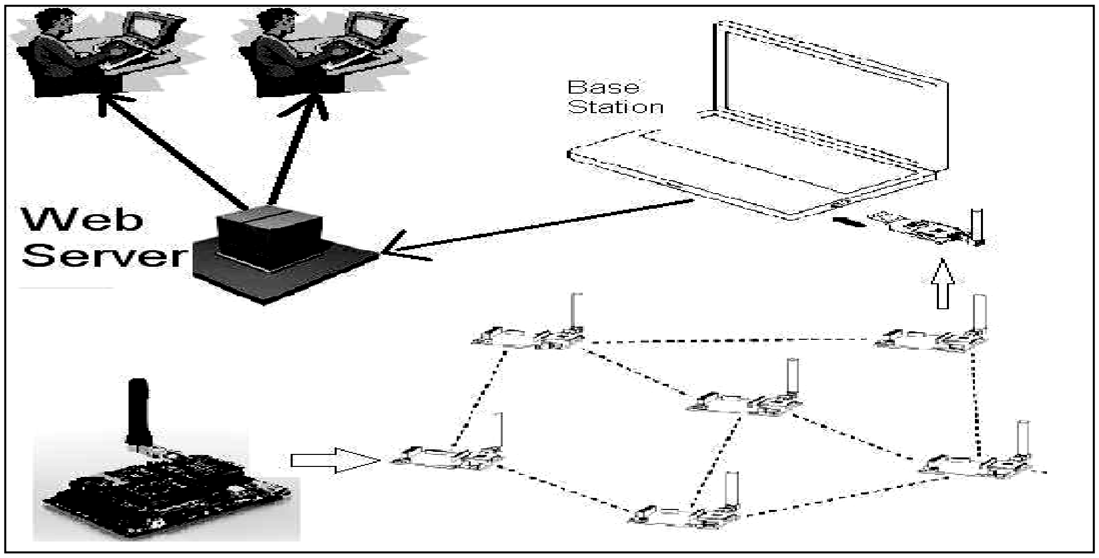


Figure 2.8: Multi Hop Mesh Network System Architecture for Real Time Wireless Pollution Monitoring System (Khedo, Perseedoss, & Mungur, 2010).

(Partheeban, Hemamalini, & Ragu, 2012) developed an air pollution monitoring system that provides alarm messages about potentially dangerous measurements from sensor data analysis. The data analysis step was designed to understand the detected air pollution regions and levels such that the analyzed data is used to track the pollution and to give an alarm. It was concluded that if the monitoring system is implemented it can be used to mitigate the damages caused by the air pollution.

A system that can read all parameters such as oxygen level, CO, CO2 and humidity in air and then send back to a Master unit by wireless zigbee protocol was designed and implemented in (Schreiner, Branzila, Trandabat, & Gobanu, 2006). The developed system operates in real time with inter-task communication after being scheduled by a scheduling algorithm. The system consists of transmitting and receiving sections which contain sensors, PIC18F52 microcontrollers, RFM70 zigbee modem and an alarm unit. The composition of air is computed and compared with polluted air and by this comparison; it is easy to deduce pollution levels.

In (Pravin, Deepak, & Angeline, 2013), a concise attempt was made to modify low cost available pollution devices to work for indoor and outdoor environments. Indoor carbon monoxide gas level monitoring using cheap alarm sensors was carried out while a computer program was designed to facilitate computer based monitoring process and logging of pollution data. In

2.4 Review of Pollution Sensors

A gas sensor **is** an electronic device which detects the presence of various gases within an area, often as part of a safety system. It interacts with a gas to measure its concentration.

There are many different types of gas sensors available commercially from metal oxide semiconductor to pellistor type sensors. This wide range of available sensors includes:

1. Infrared Gas Sensor
2. Pellistor-Catalytic Gas Sensor
3. Electromagnetic Gas Sensor
4. Thermal Conductive Gas sensors
5. Metal Oxide Semiconductor

The Infrared Sensor

The infrared sensor detection method is based upon the absorption of infrared radiation at specific wavelengths as it passes through a volume of gas. This method is based on the ability for certain gases to be able to absorb the infrared radiation, which limits the range of gases it detects. Its advantages included ability for fail-safe operations deployment, ability to operate in absence of oxygen and immunity to contamination. However the complexity of its calibrations for each particular gas and the added high costs makes it unsuitable for some designs.

The Pellistor-Catalytic Sensor

The Pellistor-Catalytic sensor is based on the principle that combustible gases and oxides produce heat, within the sensor are platinum heating coils, which change in resistance when the temperature varies. This change in resistance is linearly proportional to the gas it detects. However, the disadvantages for this method include its easy susceptibility to contamination, prolonged use may affect performance and complexity of calibration.

The Electromagnetic Sensor

The Electromagnetic Sensor detects via a chemical reaction within the sensor. Its advantages include its linearity, repeatability and its long life span. It could detect a wide range of gases and had proven results theoretically.

Metal Oxide Semiconductor

The phrase “metal oxide semiconductor” is a reference to the physical structure of certain Field Effect Transistors (FETs), having a metal gate electrode placed on top of an oxide insulator, which in turn is on top of a semiconductor material. Aluminum was once used but now the material is poly silicon. MICS-4514 is an example of a metal oxide sensor.

MICS-4514

The chosen sensors for this research is the MICS-4514, these sensors are solid-state devices composed of sintered metal oxides which detect gas through an increase in electrical conductivity when reducing gases are absorbed on the sensor's surface. These sensors are both in the same SMD package, and have a small size. The MICS-4514 includes two sensor chips with independent heaters and independent sensitive layers. One sensor chip detects oxidizing gases (OX) and the other sensor detects reducing gases (RED). In order to be able to make exact measurements with MICS-4514, it is necessary to take into account the effects that temperature has on the characteristic curve of the sensor resistance versus gas concentration. The sensor senses ambient air containing pollutants by gas diffusion. A change in electrical resistance of the sensing layer can be converted into a voltage change. This voltage change can then be used to calculate an equivalent concentration with logic control, to display the sensed equivalent value or to sound an alarm by comparing the voltage from the sensor with a pre-set threshold voltage

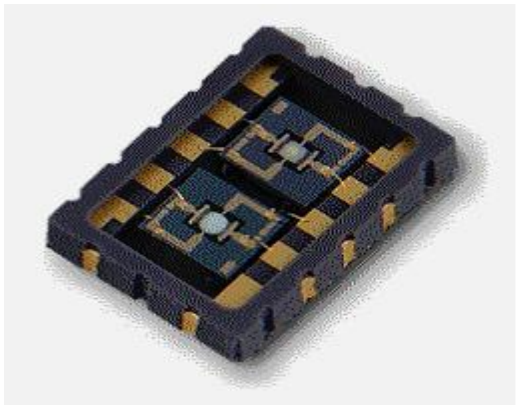


Figure 2.9: MICS 4514 Semiconductor Based Sensor.

Basic Sensor Parameters

The basic principle behind the metal oxide sensors is that the resistance of the detecting layer in the sensor changes, in the presence of the target gases. For oxidizing gases such as ozone or nitrogen oxides, the resistance will increase; while for reducing gases such as carbon monoxide or VOC's the resistance will reduce. In very simple terms reducing gases remove some of the 'insulative' oxygen species at the grain boundaries, thus causing the overall resistance to go down. Alternatively oxidizing gases add to the insulative oxygen species causing the resistance to increase. These reactions occur at elevated temperatures and hence the sensing layer needs to be heated with the integrated heater. One of the main advantages of the MICS sensor is that the power required to heat and run the sensor are less than other types on the market. The change in resistance with the change in gas concentration is not a linear response. This response can be measured and fitted to a polynomial relationship.

Sensor Signal Conditioning

The sensor module can be powered with 5V. In order to obtain a nominal heating power of 70 mW, a resistor Rs is connected in series with the heating resistor R heater. R_s can then be calculated by the following expression:

$$R_s = \frac{R_1}{(V_{cc} - V_s)V_s} \qquad 2.4$$

Heating the Resistor

The temperature of the sensing layer depends on the heating power and on the ambient temperature. To obtain good sensitivity and fast response times, the sensing layer temperature should stay within a temperature range of 350 °C to 400 °C. Below 350 °C the sensitivity decreases and the sensor response becomes significantly slower. Above 400 °C the sensitivity also decreases and at temperatures above 475 °C the sensor structure can deteriorate due to overheating.

2.5 Review of Related Literature

The presence of missing data faults in measurements, which rely on the deployment of data acquisition systems for environment monitoring applications, was handled using regression models to establish the existence of data/spatial correlation (Žliobaite, Hollmén, & and Junninen, 2014), the study characterized missing data fields and proposed a study based on regression model algorithms, which was able to show the distribution of missing sensors in a typical environmental monitoring applications and their effects to the overall aim of data gathering. it showed that 35% of all times, at least two(2) to four(4) sensors where missing (drop-out sensor faults) , with a mean number over an entire data set of about 2.4. The study argued that ignoring such missing data in a study will lead to inaccurate forecasting potential with such a data set. A correlation of the missing data set was thus established and using a simplified linear regression model was able to predict missing values from predefined constants.

Hardware redundancy can be used in different configurations to ensure fault tolerance, where a common method is the triple modular redundancy scheme (TMR) (Krstic, Stojcev, Djordjevic, & Andrejic, 2015). The study highlighted the problem in a TMR application for Fault Tolerant Data Acquisition systems, where it claimed that redundant sensors cannot be relied upon in a fault free state to match the normal output in such architecture. Thus, the study developed a VLSI fault tolerant voter with redundancy incorporated into the internal chip architecture. The study proposed an increase from three (3) redundant sensors to four (4). Most TMR systems make the basic assumption that in fault-free operations all three inputs to the voter are equal. In fault tolerant data acquisition systems (FTDASs), the remote sensor element (SE) is usually replicated at each location of the monitored technological process with the goal to allow tolerance of sensor element failure. However, it is well known that the outputs of redundant sensor elements cannot be guaranteed to match even in the absence of faults. Namely, the SE readings differs due to slight differences in their calibration, physical location (of SEs) in respect to reference, induced noise, inserted errors during data transfer between remote SEs and the central processing unit of the FTDAS (Georgiev Z, 1994). Thus the study modified the standard TMR approach with a coagulated arrangement of four sensors around a reference sensor, where a median value would be selected in the voter during a fault situation to ensure fault tolerance in the data acquisition system.

There is an increasing need to quantify the relationships between communications/ computer design with its run time parameters. (Chihaia, 2010) Proposed a novel real-time communication protocol based on the Time Division Multiple Access (TDMA) strategy, which has built-in tolerance against the network-induced effects like lost packets. This was proved true to assuring highly deterministic and reliable behavior of a networked control system. The experimental platform developed called the WiNC demonstrated the efficiency of the TDMA protocol as a fault tolerant communication platform for use in networked control systems.

Sensor array architectures are now a regular occurrence in data acquisition systems deployed for gaseous pollutant monitoring. Thus a fault tolerant supervisory control scheme for discrete event systems using a plant possessing both faulty and non-faulty parts was developed in (Claudio, Luca, & Alberto, 2008), where the goal of the fault tolerant supervisory control was to enforce a certain specification for the non-faulty plant and another (perhaps more liberal) specification for the overall plant, to further ensure that the plant recovers from any fault within a bounded delay, so that the recovery of the system state is equivalent to a non faulty state(as if no fault occurred). The notion of fault tolerant supervisory control to provide a necessary and sufficient condition for the existence of such a supervisor is therefore based on conditions involving the usual notions of controllability, observability and relative closure, together with the notion of state stability. Since in real life most systems are not able to achieve complete fault tolerance which would mean complete recovery of full functionality of the original system, a fault tolerant supervisor would thus allow the controlled system to achieve partial functionality without violating safety specification

(Aloul, Al-ali, & Zualkernan, 2010) Deployed a GPRS Sensor Array for outdoor air pollution monitoring, the system consisted of a mobile sensing unit, which was mounted on the public transportation infrastructure, and an Internet enabled server. Each sensing unit integrated a set of sensors (CO, NO2 and SO2), a GPS and GPRS modules. Data with location information were sent to the server through the cellular network (GPRS) for further processing and analysis.

Authorized air pollution information was made available to the public through a customized Web App which allowed for pollution mapping and analysis. The study showed the possibility of deploying multiple sensors on the same data acquisition system for measuring different gaseous pollutants at the same time.

(Henderson, Shilcrat, & Hansen, 1983) Introduced the concept of logical sensor processes which aims to ensure sensor reconfiguration is as easy as possible, thus aiding the implementation of a fault tolerant scheme such as FDI. The study reiterated the need for error determination and recovery using either replication (redundancy) for software and hardware or direct replacement of said faulty components. Thus it defined a logical sensor as a sensor whose output whether as a reference or a replication lies within the limits specified by a user for any specific application.

(Romano, Rughetti, Quaglia, & Ciciani, 2005) Developed a Posteriori Active Replication system (APART), where a novel active replication protocol was specifically tailored for multi-tier data acquisition systems. The study identified a common scenario in data acquisition systems, where sink processes, which filter and/or correlate incoming sensor data, produce output messages only if some application relevant event is detected. The use of a posteriori rather than a priori replication approach was shown to result in a deterministic system behavior while in a fault or faultless state.

(Werner & Peter, 2009) Worked on Eddy covariance flux measurements of methane and carbon dioxide, other trace gases, and fog droplets, which often involve the use of modern analyzers and digital signal processors. The study showed how eddy covariance could be applied to measurements carried out with data acquisition systems. The concept of eddy covariance is very much similar to basic correlation methods for determining errors in measurements due to sensor drop out or sensor bias faults. The study showed that the use of digital data acquisition systems for certain applications and analysis generated lesser values with bias faults as compared to their analog equivalent.

A real time monitoring scheme for air pollution using solid state gas sensors with an advanced RISC machine (ARM) module was developed in (Virjnatha, Aranvid, & Kumar, 2013). The system connects the measured air pollution levels to GIS information and transmits via the internet so that the pollution levels can be known in real time with one click from any place.

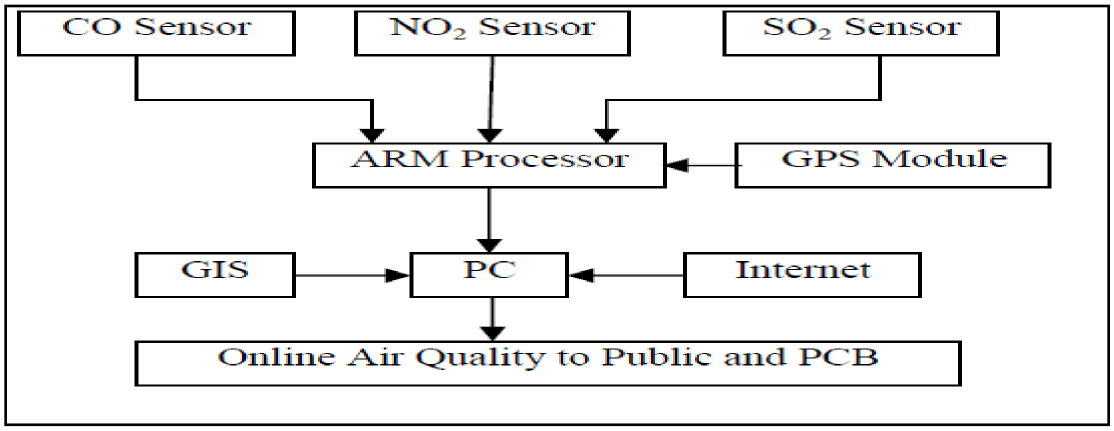


Figure 2.10: Block Diagram of the Air Quality Monitoring System (Virjnatha, Aranvid, & Kumar, 2013).

The study was meant for Chennai City and the vehicular air pollution was measured continuously and published on a website developed for the study. The Chennai city map was digitized with GIS & GPS software by moving around the city while the digitized map was fed into the internet. The three gas sensors which measured the gases namely, CO, NO2 and SO2 were linked with GPS through the ARM circuit and the data was transferred through laptop and then was uploaded to the internet so that people throughout the world can view the pollution level of the particular place in Chennai at any time using the website. It also helps in uploading the linked data to the internet. The ARM Processor, together with its outputs is collectively known as the ARM Module. It has 128-bit wide memory interface and unique accelerator architecture enabled 32-bit code execution at maximum clock rate. For critical code size applications, the alternative 16-bit Thumb Mode reduces code by more than 30 % with minimal performance penalty. With their compact 64 pin package, low power consumption, various 32-bit timers, 4-channel 10-bit ADC, 2 advanced CAN channels, PWM channels and 46 GPIO lines with up to 9 external interrupt pins these microcontrollers are particularly suitable for automotive and industrial control applications as well as medical systems and fault-tolerant maintenance buses.

The identification of faulty nodes on the transmission path of a data acquisition system plays a major role in energy conservation (Ding, Hu, Hao, & Cheng, 2015). With the dense deployment of sensor nodes, the failures in node and link are high that disrupts the entire communication of measured variables (Mahapatro & Khilar, 2013).

(Darwish & Elqafas, 2016) Developed an alternative fault free path prediction model to perform the communication amongst nodes in a sensor network. The study determined that the occurrence of faults in a sensor deployment lies in two ways, which could be internally occurring or externally occurring. The fault free path to prevent external faults and the possibility of it spreading to a healthy node was explored and determined.

(Wang & Hongyi, 2010) Focused on a Delay/Fault Tolerant Mobile Sensor Network (DFT-MSN) for pervasive information gathering. Simple and efficient data delivery schemes tailored for the DFT-MSN, which has several unique characteristics such as sensor mobility, loose connectivity, fault tolerability, delay tolerability, and buffer limit was developed in order to ensure a consistent fault free path for communication between two or more tier data acquisition system application. The study proved that a feedback approach with duplicate copies of data measurements existing between the tiers, would promote system robustness to potential faults and failures. Thus improving system overall reliability.

Gaseous pollutants and their health effects were presented in (Daudi S, 2017). The study used a multi-tier data acquisition system to monitor six (6) different gaseous pollutants at the same time from two industrial locations in Tanzania. The study deployed two sensor nodes which reported these gaseous pollutants to a central location using CDMA technology communication devices. The data gotten showed the high rate of pollution in the region of interest and was claimed to be used by government agencies to initiate prosecution of affected industries.

Fault tolerant frameworks for detecting routing failures in air pollution monitoring applications were developed and presented with promising results in (Kameswara & Chakravarthy, 2017), (Singla, Mukhdeep, & Manshahia, 2017) & (Manshahia, 2016).

(Ramon, 2007) Proposed fault accommodation techniques in software deployed for fault tolerant data acquisition systems. The study proposed the use of replication, diagnosis and backward recovery for fault tolerant implementation. The use of recovery blocks in a software programs written for fault tolerant systems also showed great promise to preventing software related faults and failure in deployed systems.

2.6 Discovered Gaps in Knowledge

The existence of missing data as a result of drop out sensor faults in a data set could lead to faulty analysis or prediction later on, thus it’s paramount that redundancy of sensors at the data acquisition system could have prevented this occurrence. The use of such redundancy is to ensure that when a drop out sensor fault occurs on a reference sensor in the Data acquisition system, the system still would have a redundant sensor it could switch operations to. Thus reduce the possible occurrence of missing data to an acceptable minimum.

The popular triple modular redundancy (TMR) approach to fault tolerance is an over-kill in many applications which ordinarily would require a simpler approach. The use of a voting algorithm to detect the presence of a fault using triple redundancies has been proven to cause additional errors in measurement. Since the data is always voted at its median value, subject to spatial distribution and calibration accuracy between the redundant sensors and the reference in TMR architecture. A cost effective vote-less system utilizing fewer redundancies with less spatial and calibration difference between a reference sensor and a redundant sensor deployed for the same application could produce a more reliable system.

The use of a TDMA communication type device, posteriori correlation methods for fault detection and recovery blocks in software fused with replicable logical sensors capable of operating within user specified ranges; in a data acquisition system for gaseous pollutant monitoring would offer better overall system reliability and robustness to handling variance degradation sensor faults and off set bias faults.

CHAPTER THREE

METHODOLOGY

This chapter outlines the steps taken in the research, the research questions, adopted design methods, sample selections, data collection and analysis. Most air pollution and quality monitoring schemes are based on sensors that report the pollutant levels to a server via wired modems, routers or short range wireless access points. The study thus proposes a fault tolerant system design utilizing redundancies, ranging from the integration of microcontrollers to several air pollution sensors, GSM-Modems and GPS modules that would utilize the wireless mobile public network. The system objectives are to be achieved in the following areas:

3.1 Sources of Data

The sources of data used in this study includes

1. Pollutant concentration level data measured from the factory environs at Ife steel Mill, Fashina bus-stop, Ile-Ife, Osun-State. Nigeria, these measurements will be carried out in the month of September, 2014 using the developed system.
2. Pollution data and survey information were sourced from air quality monitoring surveys and air quality reports. E.g. US-AQI and IND-AQI Standards Documentation in (Farrah, Mateen, & Brook, 2011).

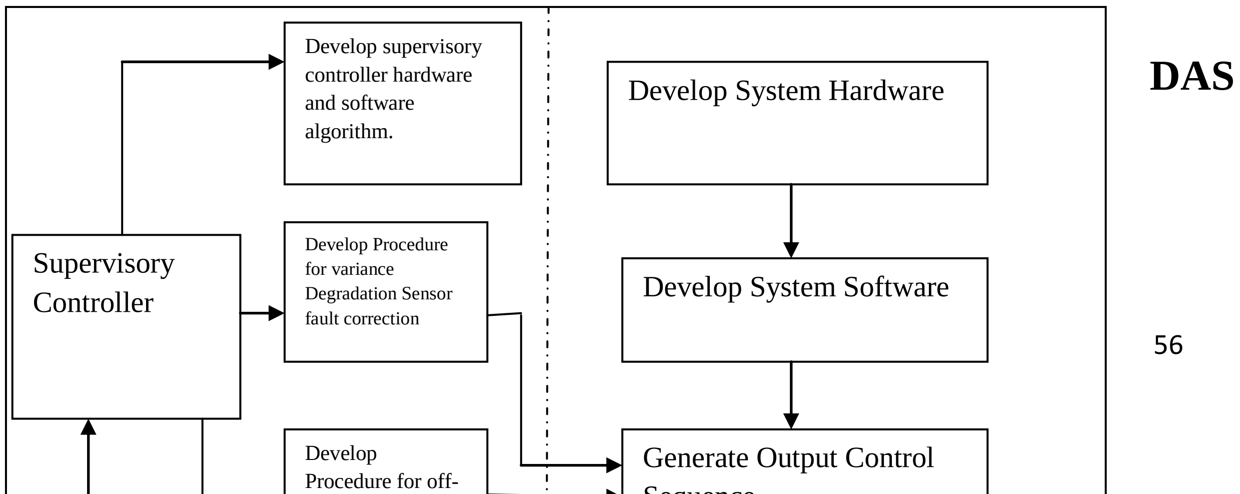
3.2 Choice of Design Methodology

The hardware for the data acquisition system was developed with a top- down design approach as shown in fig 3.1, which looked at the entire system in an overview of its constituent parts. The constituent parts were designed, analyzed and implemented individually, and then integrated to realize the final system. The figure 3.2 depicts the system hardware framework.

The C programming language is used in the controllers for implementing its control procedures as well as the FDI model to be discussed in other sections.

3.2.1 Top Down Design Methodology for the FDI Data Acquisition System

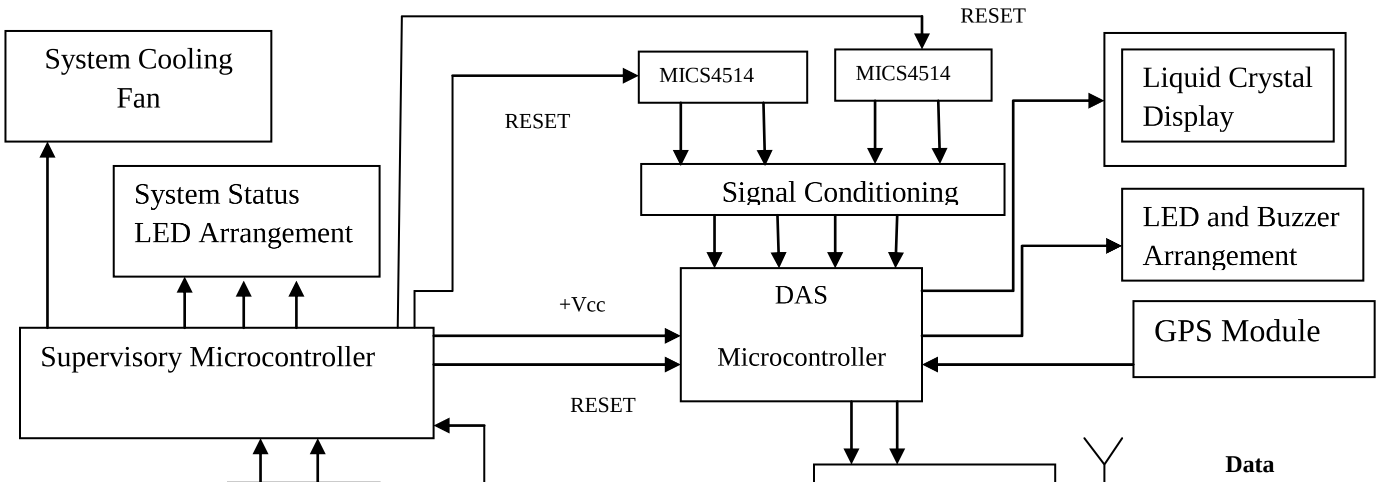
The overall framework for the FDI DAS is first presented showing its component parts, use -case and data flow diagrams. The system controller for the data acquisition system (DAS) and other attached I/O components such as the MICS 4514 sensors, LCD, buzzer, LED’s, are designed and the control algorithms presented using flow charts and pseudo-codes. The output data frame from this DAS module goes to the Central Remote Terminal Unit (CTU), whose role is designed, using flow charts and pseudo-codes to implement a fault detection model that generates output data frames to be sent in duplicate to a pollution database, as well as to the supervisory controller in the DAS module as a feedback. Algorithms for the supervisory controller are designed, which describes the error residual generator model employed for fault accommodation. An Air Quality Monitoring Software (AQMS) is used for data analysis and testing.


DAS
Develop supervisory controller hardware and software algorithm.
Develop System Hardware
Supervisory Controller
Develop Procedure for variance Degradation Sensor fault correction
Develop System Software
Develop Procedure for off-
Generate Output Control

Receive Fault Status
Stream from CTU

The physical model for the FDI data acquisition system shown in figure 3.2, exploits redundancy in design, by employing multiple sensors and a supervisory controller to measure the different gaseous pollutant concentration levels. The entire system consists of three parts; the first is the data acquisition system, then the central remote terminal unit and lastly the data analytics section. The data acquisition system consists of the DAS controller, supervisory controller, GPS module, MICS 4514 sensors, LCD display and a power supply circuit, while the central remote terminal unit and the data analytics unit consists of controllers, GPRS and GSM modems , and an AQMS software respectively.

Summarily, the system could be broken down to five (5) major processes requiring data transfer between the different parts of the system (Sensors, DAS controller, Supervisory controller, AQMS software, Central remote terminal unit) and these includes, the measure sensor reading, send data frame via modem, modify process (sensor reset etc) all existing between the DAS microcontroller and its sensor and supervisory controller, while the feedback process, FDI model implementation , send/receive data via modem are data transfer processes that occur between the supervisory controller and the central remote terminal unit via the GSM modem. The central remote terminal unit and the AQMS software share the database with SEND and RECIEVE data commands being possible data interactions between the two processes. Reconfiguration commands, pollution data information, sensor ID, fault status, are possible data streams that are transferred from one unit to another to achieve the overall system.



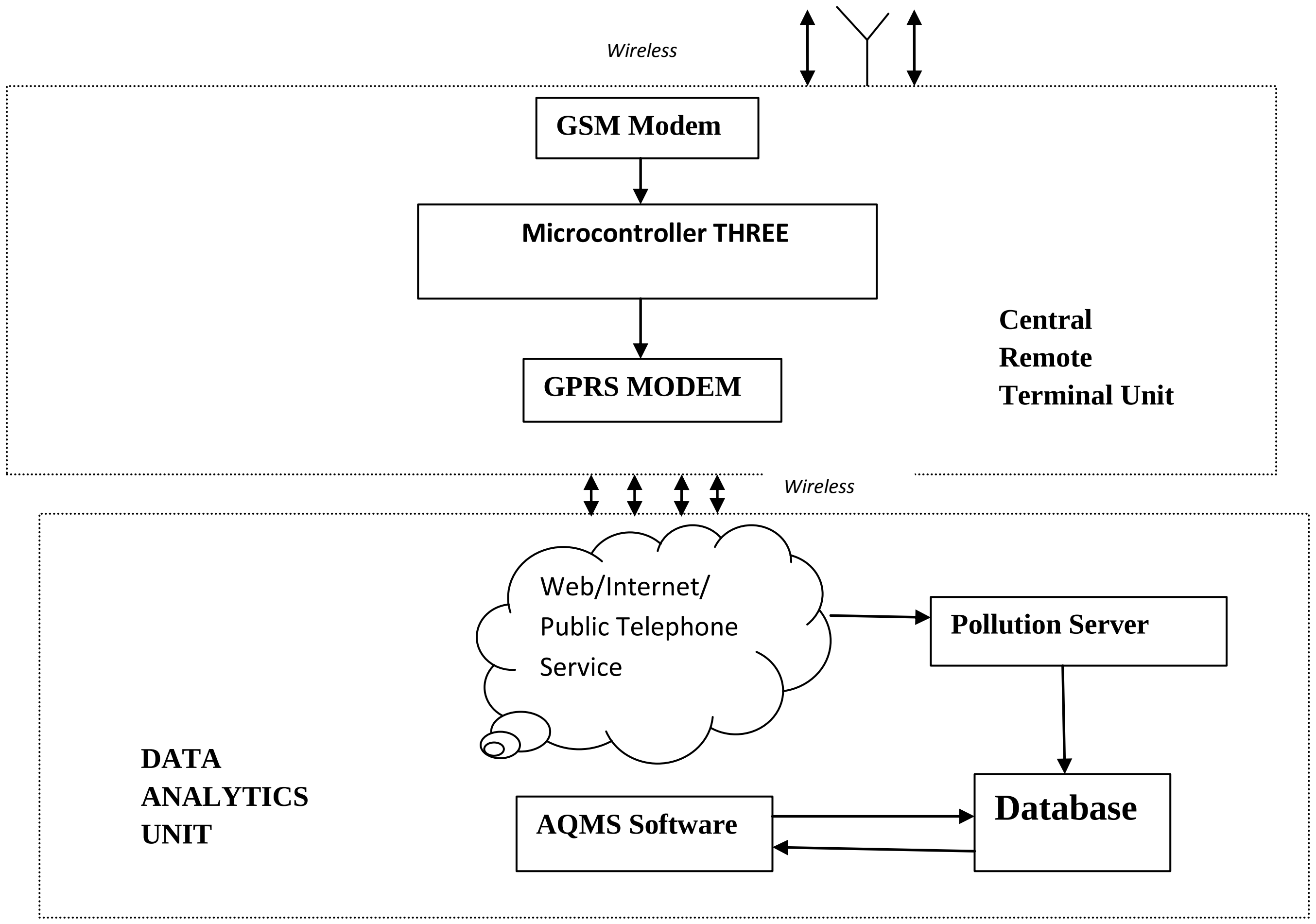
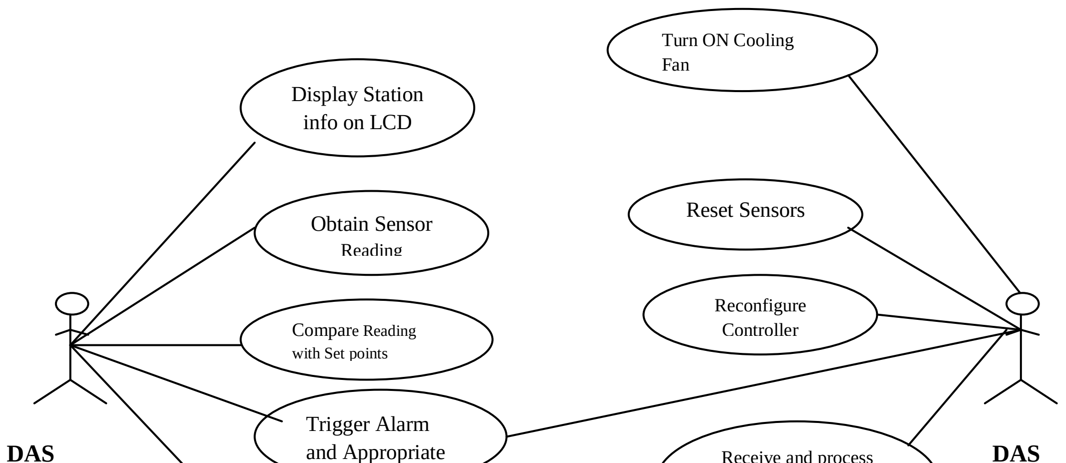


Figure 3.2: Complete FDI Data Acquisition System Architecture

The use case diagram depicted in figure 3.3 highlights the roles and functionalities of each module in the FDI data acquisition system.



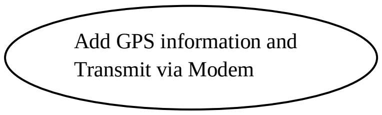
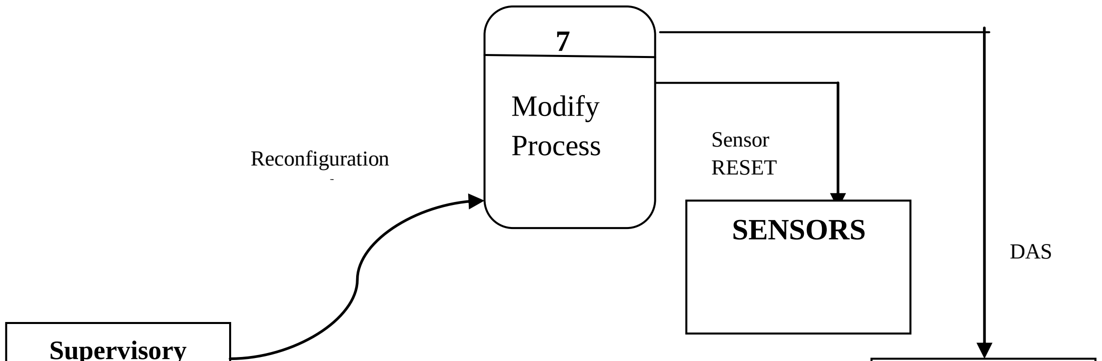


Figure 3.3: Use Case Diagram for the Fault Detection and Isolation Data Acquisition System.

The data flow diagram shown in figure 3.4 highlights the data flow between the different sections of the entire system. There are seven (7) different processes that play a role in the data flow (transmission and reception). These processes lie between the main system components which initiate these data transfer processes.



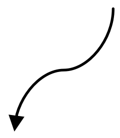


Figure 3.4: Data Flow Diagram of the FDI Data Acquisition System.

3.3 The Data Acquisition Unit

The data acquisition system unit is tasked with measuring data from sensors, comparing the measured data with set points, displaying these measured data on an LCD display, turning ON or OFF an alarm in the presence of an alarm condition and sending same data to a central terminal unit via an output data frame. The DAS contains the following components and sub sections: LCD, LED, Buzzer, GPS module, microcontrollers, sensors and a GSM modem.

3.3.1 Hardware Design

The FDI data acquisition system runs on a 5V and 12V DC power and this enables the TTL circuits and its associated circuitry to function, as well as providing power to the internal fan via the 12V output. The circuit in figure 3.5 was implemented for this purpose because its properties meet the current power needs. The circuit uses two IC's 7812(IC1) and 7805 (IC2) for obtaining the required voltages. The AC mains voltage will first be stepped down by the transformer T1, which is able to step down the 230v/220v AC power to a 15V AC. A 1A fuse is used as F1, while the switch S1 is an SPST ON/OFF switch. The bridge circuit B1, made up of four (4) 1N4007 diodes would perform the role of rectification, while C1 of 2200µF performs the filtering to obtain a steady DC level. The LED D1 acts as a power ON indicator, while additional capacitors C2, C3, C4, C5 and C6 provide filtering and smoothening of the signal. The 7812 regulates this voltage to obtain a steady 12V DC, while the output of the 7812 would be regulated by the 7805 to produce a steady 5V DC. This circuit was chosen for use because it offers a simplistic solution to multi-voltage power required by the FDI data acquisition system. The use of other IC's could generate other voltage levels, e.g. if 7809 was used instead of 7812, a 9V DC would be generated, while if a 7806 was used instead of a 7805, a 6V DC would be generated.

12V/5V Power Supply

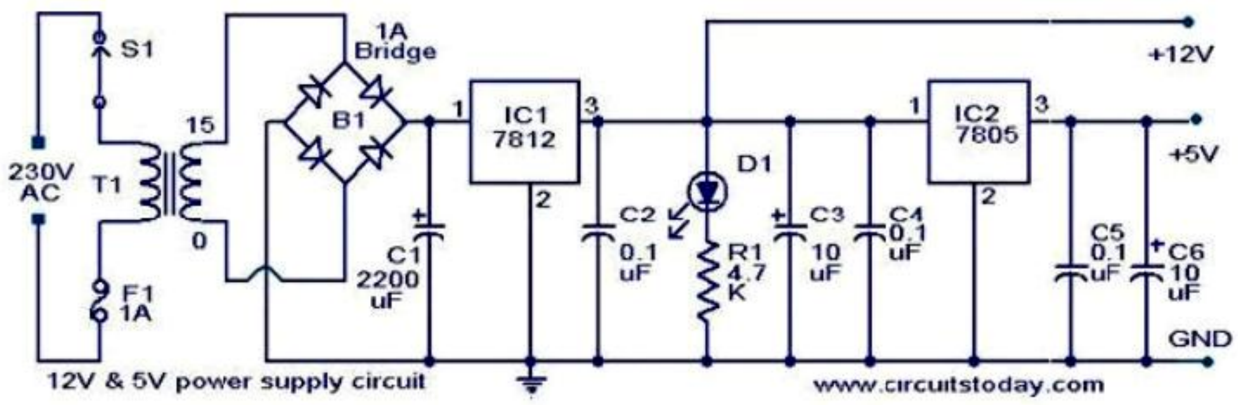


Figure 3.5: Power Supply Circuit (circuitstoday, 2017)

Supervisory Controller in the Data Acquisition System Module

The design for this unit utilizes a dual in line 40 pin ATMEL 89C52 microcontroller to accomplish the task, prior to using the controller; a 12MHz quartz crystal is connected with a 33pF capacitor to XTAL1 and XTAL2 on the microcontroller to provide the clock signal. The controller is powered by a 5V input (V_{cc}) provided by a power supply circuit in figure 3.5.

The reset pin to the controller which serves as the next input control signal for its operation is connected as shown in figure 3.6. When the 5V power is switched on, the capacitor shorts to 5V, and then gradually the RC circuit discharges to bring the reset pin to 0.

According to the 8051 documentation, the voltage across the reset pin should be at logic high for more than 2 machine cycles (where 1 machine cycle = 4 clock cycles).

$V_i = 5.0V$ 3.1

$V_f = 90\%\ of\ V_i$ (Very safe). This means $V_i . e^{\frac{-t}{RC}} > V_f$. 3.2

So $e^{\frac{-t}{RC}} > 0.9$ 3.3

So $e^{\frac{-t}{RC}} < -1.In0.9$ 3.4

When $t = \frac{10}{f_{xtal}}$ (which is safe, much higher than 2 machine cycles) 3.5

So $e^{\frac{\frac{10}{f_{xtal}}}{RC}} < 0.105$ 3.6

$RC > \frac{10}{0.105 * f_{xtal}}$ 3.7

$RC > \frac{100}{f_{xtal}}$ 3.8

If the value of resistor R =10k,

$C > \frac{100}{10k * f_{xtal}}$, then 3.9

$C > \frac{0.01}{f_{xtal}}$, the units of C will be in µF if F_{xtal} is in MHz since the value of the quartz frequency is

F_{xtal} = 11.0592 MHz, thus, $C > \frac{0.01}{11.0592}$ = 1nF, thus any value of C greater than 1nF and R=10k will be able to reset the microcontroller.

Since there is no requirement for an external memory in this application, thus the EA pin in the controller is set to HIGH by connecting it to Vcc. PSEN and ALE are all left alone (unconnected). The supervisory controller is also connected to a MAX232 IC as shown in figure 3.5 to enable data transfer from the GSM modem.

A 12V/5V external power supply circuit in figure 3.5 is used to provide the 12v needed to power the internal cooling fan; an OZ-5H-105D relay is used to perform the switching action between the fan and the supervisory controller. The system cooling fan is a 120mm*120mm*25mm, 12v DC fan. This will help eliminate dust and heat from the DAS as determined by the supervisory controller. This was not shown in figure 3.6 because it's an external fan. Four light emitting diodes are used to indicate in-process conditions. LED's are major components used in electronic design due to its numerous application areas. In this study a total of eight LED's are used for both the supervisory controller (U3) and the DAS microcontroller (U1). They are used for indicating different conditions/states of the DAS unit itself or the supervisory microcontroller in the DAS unit. LED's have a voltage drop of 1.7V and current of 10mA to glow at maximum intensity. This voltage is thus applied through the microcontroller output pins as shown in figure 3.7. The negative terminal of the LED is connected to the ground through a resistor. The value of this resistor is calculated using the following formula.

$$R = \frac{V-1.7}{10mA}, \text{ where V is the input voltage} \qquad 3.10$$

Generally, microcontrollers output a maximum voltage of 5V. This output serves as the input voltage to the LED, thus $R = \frac{5-1.7}{10mA} = 0.33k\Omega$

Thus the value of resistor calculated for use is 330 Ohms. This resistor can be connected either to the cathode or anode of the LED.

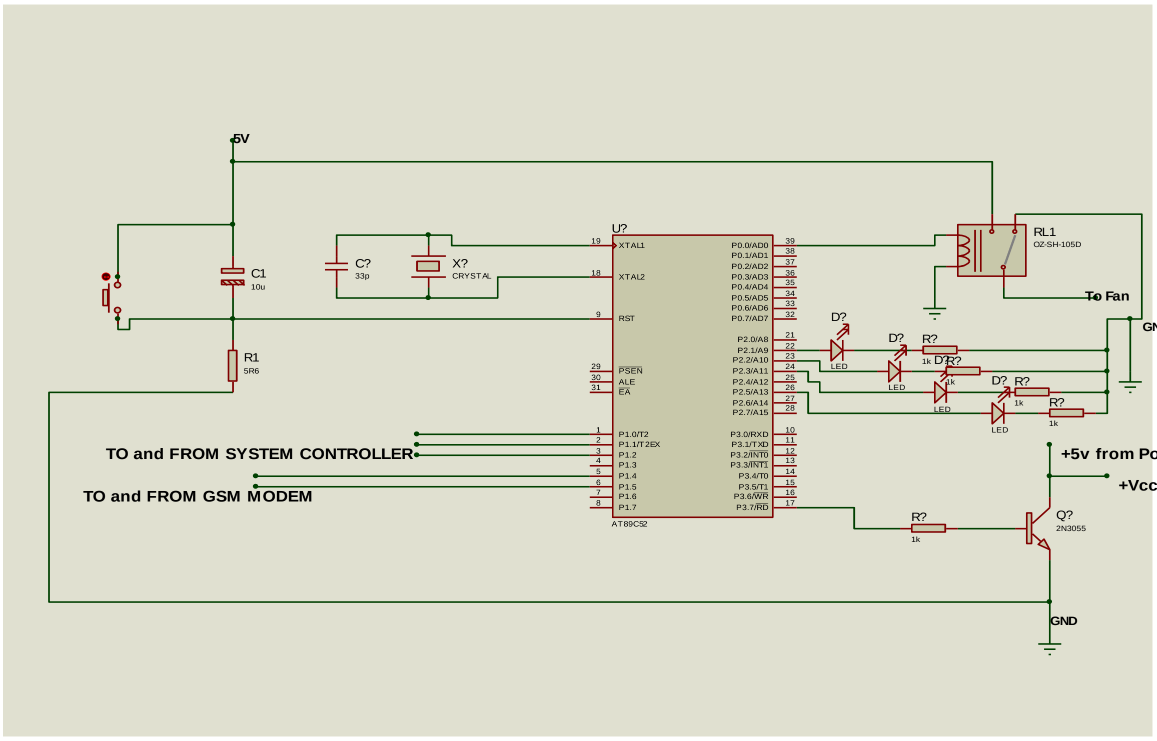


Figure 3.6: Supervisory Controller in the FDI DAS.

MICS4514 Sensor Module

The MICS-4514 sensor is a pre-calibrated sensor; Figure 3.7 is the standard method according to its data sheet (e2vtechnologies, 2008) for connecting the MICS 4514 sensor both for power and measurement. Two external load resistors are used to power both internal heaters for each sensor with a single power supply of 5V as shown in figure 3.7 (a). RDRED which is the resistance for the reducing type sensor in the circuit is 82Ω, while RDOX which represents the resistance for the oxidizing type sensor in the circuit is 133Ω. These resistors are necessary to obtain the right temperature of the two independent heaters while using a single 5V supply. This results in VHRED which represents the voltage drop in the reduction type sensor to be 2.4V and VHOX which represents the voltage in the oxidizing type sensor to be 1.7V. Load resistances of 820Ω are applied to the MICS 4514 sensor where the two voltages measured across them is linked to the resistance of the RED and OX sensor sensitive layers respectively. The detection of pollution gases is done by measuring the sensing resistance of the two sensors, where the resistance of the sensitive layer on the reduction type sensor reduces in the presence of the CO pollutant while the resistance of the sensitive layer on the oxidizing type sensor increases in the presence of the NO pollutant.

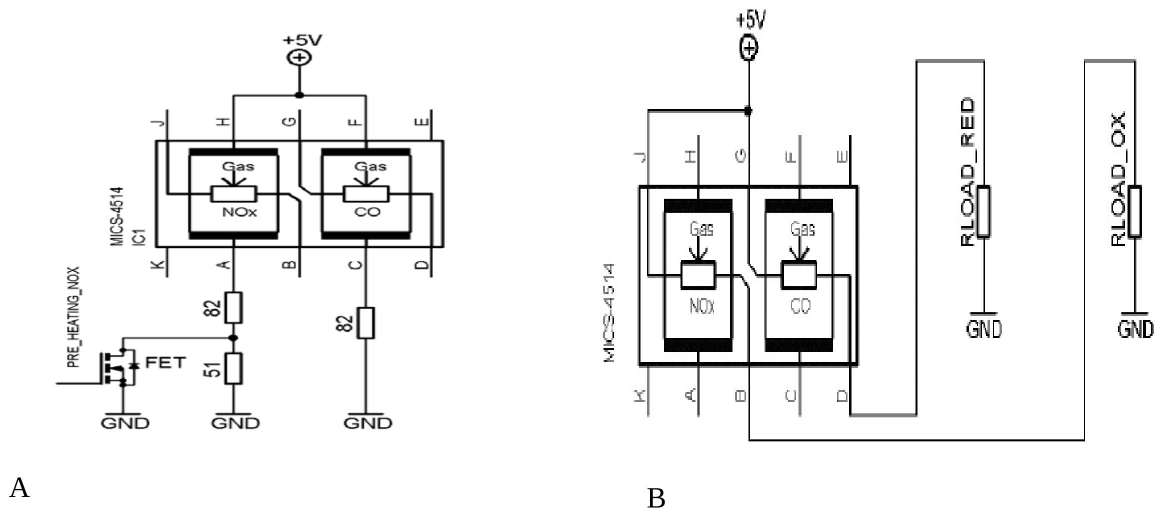


Figure 3.7: Measurement and Supply Circuit for the MICS-4514 Sensor.

GPS Module

A GPS module is a device that uses the Global Positioning System to determine the location of a vehicle or person or industry as in this case. GPS receivers are used to provide reliable navigation, positioning and timing services to the users at anytime and anywhere on the earth. This Global positioning system uses 24 to 32 satellites to provide the data to the receivers.

GPS module calculates the position by reading the signals that are transmitted by the satellites. Each satellite transmits the messages continuously which contain time sent. GPS receiver measures the distance to each satellite based on the arrival time of each message. This information is used to calculate the position of the GPS receiver. The received raw data is converted for the user as LATITUDE, LONGITUDE, ALTITUDE, SPEED and TIME.

The receiver pin of GPS module is connected to the 13th pin of max232 IC and GND pin is connected to ground. Controller RXD pin is connected to the 12th pin of max232. Here max232 IC is used for level conversion and is connected to the DAS microcontroller.

The GPS receiver continuously transmits the data as per the NMEA standards using RS232 protocol. In this NMEA format, the LATITUDE and LONGITUDE values of location are available in a GPRMC sentence. In this project LATITUDE and LONGITUDE values are extracted from NMEA format and used directly by the DAS microcontroller in creating the output data header frame.

The data is received by the controller from the GPS module serially using UART protocol (MAX232) and it now extracts the latitude and longitude values from the received messages.

FDI Data Acquisition System Hardware

The different modules were interconnected together with some additional components such as a buzzer, which is an electronic device that converts an electronic signal into a buzzing noise when applied to it. The study utilized a buzzer in the DAS unit to sound an alarm for a period of time as determined by the DAS controller algorithm. The buzzer used in this study was selected to ensure it has a maximum current rating of 20mA. A 1K ohm resistor is connected across to sink the buzzer to LOW state after the microcontroller would have switched its connecting port to HIGH to turn it ON. The GPS module was also connected to the DAS module in order to provide longitude and latitude coordinates, while additional Led's were connected to the controller to work in tandem with the buzzer, in order to flag alarm situations.

The figure 3.8 showed the major parts of the FDI data acquisition system which consists of the system controller, supervisory controller, LCD, LUOCEAN GPS module and Siemens' TC351 GSM modem. The interconnecting components to the system as well as the power supply circuit aren't shown. The entire system is powered by the +5v and +12V power supply circuit shown in figure 3.5.

Atmel 89C52 Family controllers are used for the DAS controller and supervisory controller, an LM032L (16*2) Alphanumeric LCD display is used together with generic LED's which indicate normal and alarm conditions respectively for the DAS system controller (Red LED comes ON when there is an alarm condition, green LED comes ON when in normal measurement range) and also for the supervisory controller (red LED comes ON when a fault exists on the DAS, while the yellow LED remains ON in normal working conditions)., 10WATT1K resistors, 2N3955A FET, 10WATT51R resistors, LCD, Buzzer,MICS4514 gas sensors, GSM and GPS Modules. The controller (U3) on the right is the supervisory microcontroller. The system controller is connected to the first MICS 4514 sensor via pins p0.0, p0.1, p0.2, p0.3, where a

2N3955A FET, 10WATT51R resistor and 7WATT8R2 connected to p0.0 enables connection to the internal heater in the NO sensor for pre-heating operations before measurement, while p0.1 connected to just a 7WATT8R2 resistor allows for preheating of the internal heater in the CO sensor. P0.2 and P0.3 when connected to a 10WATT10R resistor enables measurement of the resistance change with respect to analog voltage between the sensitive layer and the load resistance of 10WATT10R. The second MICS 4514 sensor is connected via pins p0.4, p0.5, p0.6, p0.7, where a 2N3955A FET, 10WATT51R resistor and 7WATT8R2 connected to p0.4 enables connection to the internal heater in the NO sensor for pre-heating operations before measurement, while p0.5 connected to just a 7WATT8R2 resistor allows for preheating of the internal heater in the CO sensor. P0.6 and P0.7 when connected to a 10WATT10R resistor enables measurement of the resistance change with respect to analog voltage between the sensitive layer and the load resistance of 10WATT10R.The controller is connected to four generic LED's via ports P2.0, P2.1, P2.2, P2.3 to enable it switch ON/OFF the LED's to indicate normal measurement range where the measured concentration is less than the minimum acceptable value (green is ON) or Red will be ON when the measured concentration exceeds the minimum accepted value and thus an alarm condition as required in the program code. The supervisory controller shown is connected to the system controller via port P1.5, P1.6, P1.7 to enable communication between both controllers with respect to which sensor should be used in any measurement cycle. Port P1.4 is connected to the reset pin of the system controller to enable the supervisory controller reset it when it chooses to.

The system controller connects to the LCD, buzzer, LED's, GSM Modem and GPS module via ports P0.0- P0.7, P2.4, P2.0 - P2.3, P1.1, P1.2 and P1.3- P1.4 respectively. The system is solely responsible for measuring the pollutant concentration by the sensor, displaying such value on the

LCD, triggering the buzzer to sound for a period of time, getting GPS information from the GPS modem and transmitting measured data to the central remote terminal unit via the GSM modem

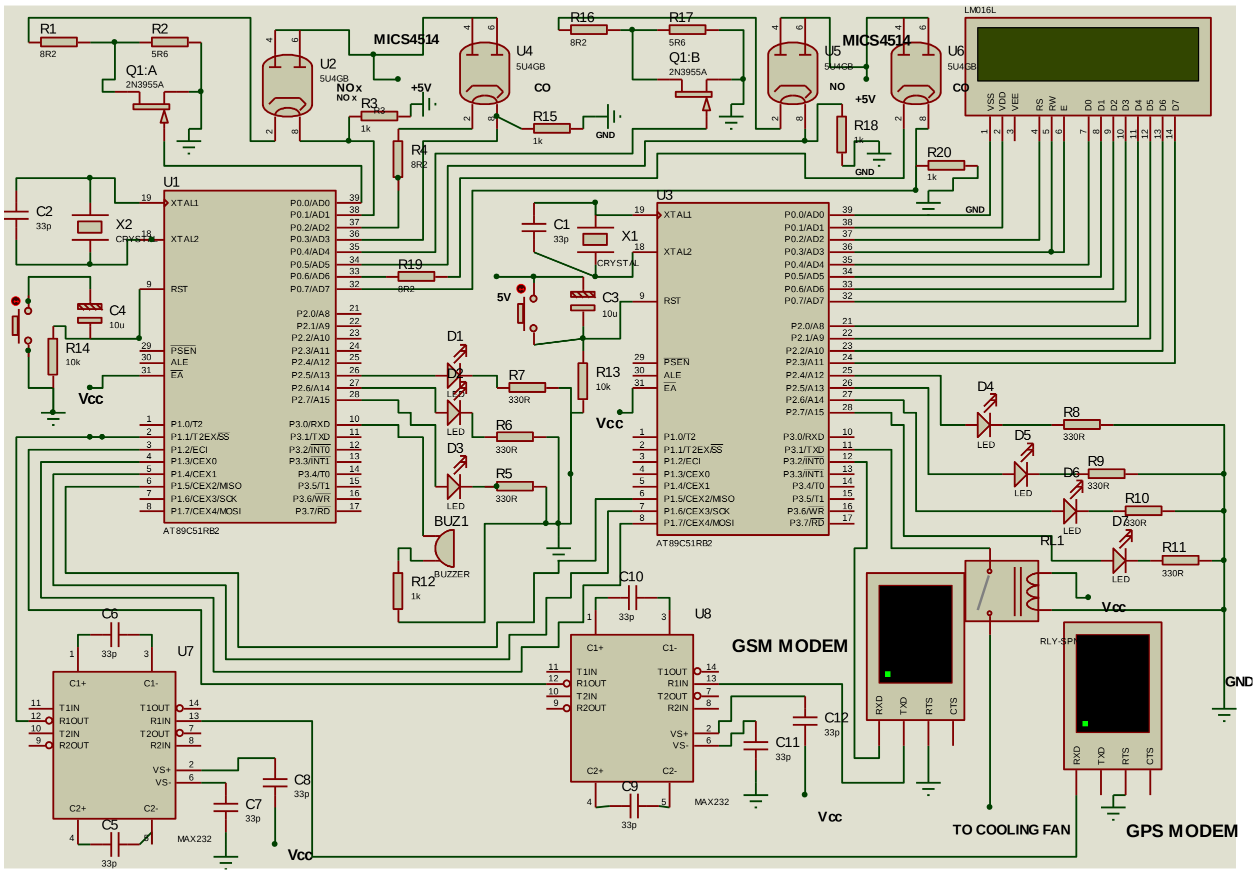


Figure 3.8: FDI Data Acquisition System Unit Hardware.

.

3.3.2 Software Design for Data Acquisition System

The system components in the hardware design in figures 3.6, 3.7 and 3.8 all require appropriate software to perform their given tasks towards meeting the objectives of the research.

DAS Microcontroller Module

The microcontroller control logic used in achieving the system requirements is shown using the flowchart in fig 3.9

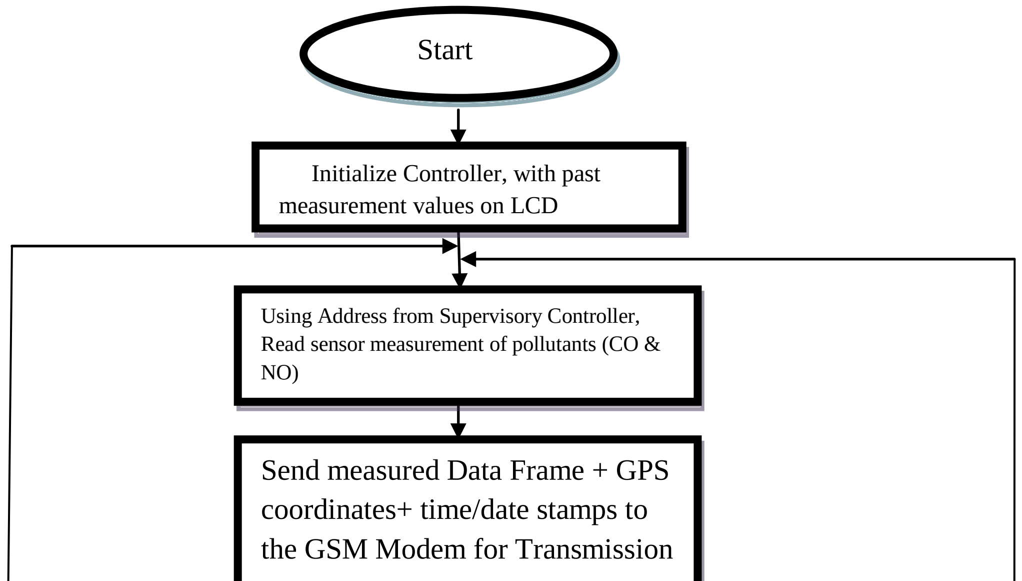

Start
Initialize Controller, with past measurement values on LCD
Using Address from Supervisory Controller, Read sensor measurement of pollutants (CO & NO)
Send measured Data Frame + GPS coordinates+ time/date stamps to the GSM Modem for Transmission

Fig 3.10 represents the pseudo code representing the system's operational sequence as elaborated through the flow chart in fig 3.9

```
INTITIALIZE Controller with dummy message of past measurement values on LCD
WHILE (SYSTEM = ON) DO
READ ADC Values (NO, CO)
COPY ADC Values (NO, CO) to Out Array
APPEND GPS coordinate, time stamp to Out Array
SEND String Based on Out Array to LCD
SEND Out Array to Modem
```

Figure 3.10: Pseudo-codes for the FDI-DAS microcontroller.

The DAS described by the flowcharts and pseudo-codes in figure 3.9 and 3.10 respectively, will display a dummy message on the LCD with past values of data earlier received, since the measurements are carried out hourly, The mode of operation is still the same implying that once data is measured , a data frame is created with longitude and latitude information gotten from the GPS module, This data stream is forwarded to the next phase (Central Remote Terminal Unit) via the GSM modem, The data representing the pollutant concentration D(i) is updated on the LCD screen and the system compares measured data and the set point D(ref) in order to trigger a buzzer and a danger (RED color) LED or to simply turn ON the green LED indicating the measured reading is not above the set point. Example, if the measured concentration for NO is 55mg/m^3, this value on comparison with a reference of 50mg/m^3 will result in an alarm situation because D(i) is higher than the reference D(ref), same comparison applies for lower values less than the reference point. Now,

$$E\ (t) = D\ (i) - \ D(ref) \qquad 3.11$$

Where E (t) is the error between values at time instant (t), D(i) is the current input Value and D(ref) is the reference value for that particular pollutant, For all positive E (t) the alarm buzzer will be triggered for 15seconds and no action will take place for all null or negative E (t).

Figure 3.11 shows the format for the transmitted output data header stream. Where ranges for the data headers were determined as follows: since $number\ of\ bits = 2^{number\ \ of\ digits}$

Then, a Controller ID range between 100 and 256 decimal which is 2^3= eight (8) bits is suitable for use.

CO and NO level ranges between 0 and 256 in value, thus 2^3 = eight (8) bit header is suitable.

The time header has a structure of 23:30 etc, where 24 represents the maximum hourly measurement and can be represented with 5bits, 60 represents the maximum minute measurement and can be represented by 6bits. Thus total number of bits for the time is 5+6=11bits. Thus we can use a 16bit header to represent this.

The date has a structure of 10/12/2017, thus 2017 for the year is represented by 2+2+1+4 = 9bits, 12 for the maximum month of December is represented by 4bits, while 31 which represents the highest number of dates in a month is also represented by 5bits. The total number of bits for the date header is 9+5+4 =18bits. Thus a 24bit representation is sufficient.

Longitude and latitude information are of the form 4.5566. With each digit represented by a maximum of 4bits, we have a total number of bits as 4+4+4+4+4 = 20bits.thus a 24bit header can be used to represent the longitude and latitude.

Controller ID	TIME	DATE	Latitude	Longitude	CO-Level	NO-Level

(8bits)	(16bits)	(24 bits)	(24bits)	(24 bits)	(8bits)	(8 bits)

Figure 3.11: Output Data Stream Produced By Data Acquisition System.

MICS 4514 Sensor Module

The sensor is interfaced with the DAS microcontroller which gets its readings by making the following calculations.

From figure 3.7 (B) the voltage drop across the load resistor (R_{load}) is used to determine the change in resistance of the sensitive layer in each sensor (RED and OX) using

$$V_{out} = \frac{R_{Load}}{R_s + R_{Load}} * V_{in} \qquad 3.12$$

This voltage divider method enables the microcontroller measure the analog voltage drop across the load resistance of 820Ω. This means that R_s can be deduced from eq 3.13. Its equivalent PPM value can be deduced by

$$PPM = \frac{R_s}{R_0} \qquad 3.13$$

Where sensing resistance in air (R_o) according to data sheet (e2vtechnologies, 2008), is 75KΩ for CO sensor sensitive layer and 66KΩ for NO sensor sensitive layer. The PPM value can be deduced for each of the pollutants from the method for any linear range on the engineering test bench sensor characteristics curve.

The study utilized measurements in mg/m^3 according to the Indian air quality index standard (IND-AQI) and thus the conversion formula from PPM to mg/m^3 is

$$concentration\ (mg\backslash m3) = concentration\ (ppm) * \frac{molecular\ \ mass\ (g\backslash mol)}{molar\ \ volume} \qquad 3.14$$

When molecular mass of CO is 28.01gmol^{-1}, and NO is 30.01gmol^{-1}. With a gas molar volume at 1atm of 22.4m^3, we can convert from ppm into mg/m^3. Where due to the non-linearity of the MICS-4514 sensor as observed on its data sheet. It was decided to observe the graph and pick lowest and maximum set points in regions of maximum tangential linearity. The measured voltage for the CO pollutant in this study across its load resistance was re-calibrated at between 0.3699V and 0.0707V for the maximum and minimum using a smoke exhaust calibration method. Therefore for V_{out} = 0.3699V, using the voltage divider method in eq 3.12

$$0.3699 = \frac{820}{R_s+820} * 2.4 \qquad 3.15$$

Where, V_{in} for the RED sensor (CO) is 2.4V.

$$R_s = 4500\Omega \qquad 3.16$$

Where , $PPM = \frac{4500}{75} = 60ppm$ 3.17

Using eq 3.14 , $mg/m^3 = 60 * \frac{28.01}{22.4} = 75.02$mg/m^3 3.18

Thus the lowest set point (L_{cp}) for the CO sensor is 75mg/m^3.

For V_{out} = 0.0707V

$$0.0707 = \frac{820}{R_s+820} * 2.4 \qquad 3.19$$

Where, V_{in} for the RED sensor is 2.4V.

$$R_s = 27000\Omega \qquad 3.20$$

Where , $PPM = \frac{27000}{75} = 360ppm$ 3.21

Using eq 3.14 , $mg/m^3 = 360 * \frac{28.01}{22.4} = 450.16$mg/m^3 3.22

Thus the highest set point (U_{cp}) for the CO sensor is 450mg/m^3.

While for the OX sensor the lowest and highest voltages in the acceptable linear range occurred at V_{out}= 0.5V and V_{out} = 0.2287V, thus for V_{out} = 0.5V

$$0.5 = \frac{820}{R_s+820} * 1.7 \qquad 3.23$$

Where, V_{in} for the OX sensor (NO) is 1.7V.

$$R_s = 1966.8\Omega \qquad 3.24$$

Where , $PPM = \frac{1966.8}{66} = 29.8ppm$ when R_o for OX sensor = 66KΩ

Using eq 3.14 , $mg/m^3 = 29.8 * \frac{30.01}{22.4} = 39.92$mg/m^3 3.25

Thus the lowest set point (L_{cp}) for the NO sensor is 40mg/m^3.

When V_{out} = 0.2287V

$$0.2287 = \frac{820}{R_s+820} * 1.7 \qquad 3.26$$

Where, V_{in} for the OX sensor (NO) is 1.7V.

$$R_s = 5273.4\Omega \qquad 3.27$$

Where , $PPM = \frac{5273.4}{66} = 79.9ppm$ when R_o for OX sensor = 66KΩ 3.28

Using eq 3.14 , $mg/m^3 = 79.9 * \frac{30.01}{22.4} = 107.04$mg/m^3 3.29

Thus the highest set point (U_{cp}) for the NO sensor is 107mg/m^3.

These values represented the maximum and minimum set point/threshold value used for the sensor in this study.

Supervisory Controller in the FDI-DAS Module

The supervisory microcontroller unit is the master controller in the entire architecture. The data header stream in fig 3.12 is the format of the feedback state information the supervisory microcontroller receives from the central remote terminal unit after its own analysis.

DAS ID	TIME	DATE	(D_{CO}, D_{NO})	Fault Set
(8bits)	(16bits)	(24bits)	(16bits)	(22bits)

Figure 3.12: Feedback Data Received by the Supervisory Control Unit

The bit length for each header is generated as follows

If D_{co} and D_{no} can have values not more than 256, and can be represented by 8bits each, thus total number of bits required is 8+8=16bits.

The fault set contains typically (F-1, F-3), if "f" is represented by a byte of 8bits and the numbers by 3bits. The total bits are thus 8+3+8+3=22bits.

The flowchart in fig 3.14 depicts the algorithm for the supervisory microcontroller in the data acquisition system. Its primary duty is to receive a feedback from the central remote terminal unit.

The feedback received indicates either an acknowledgement signal that the system is in fault free state or in fault condition. After identifying the fault state, the supervisory controller would generate a residual from the error residual generator, modify the residual if it is non-zero to zero, and then apply a control action based on its modification. This would correct the system malfunctions highlighted in this study.

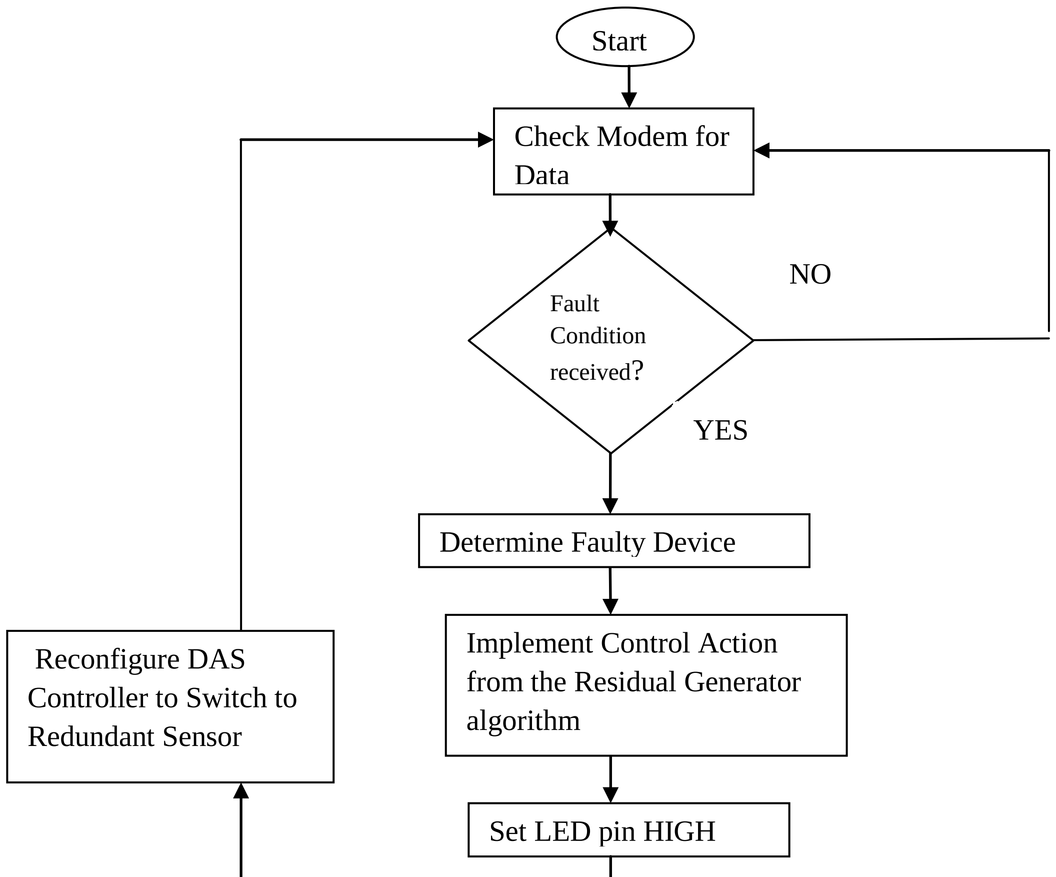

Start
Check Modem for Data
Fault Condition received?
NO
YES
Determine Faulty Device
Implement Control Action from the Residual Generator algorithm
Set LED pin HIGH
Reconfigure DAS Controller to Switch to Redundant Sensor

Figure 3.13: Flowchart showing the Supervisory Microcontroller Action.

The pseudo code below describes the sequence of operation of the supervisory controller

```
  START

SET V cc to Controller

READ data stream from Modem

IF an acknowledgement "ACK" is received

THEN Turn LED Green ON indicating system is working perfectly

IF a Fault Condition set is received

THEN classify fault condition

WHERE F-1indicates a negative offset bias sensor fault

AND F-2 indicates a Positive offset bias sensor fault

AND F-3 indicates a variance degradation sensor fault
```

Figure 3.15: Pseudo-Code for the Supervisory Microcontroller.

Error Residual Generator for Supervisory Controller in FDI-DAS Module

The FDI model implemented in the supervisory controller is based on the error residual generator for a simple input/output system.

$$r(s) = E(s)u(s) + F(s)y(s) \tag{3.30}$$

Where R(s) represents the residual matrix, E(s) and F(s) represent unit (3*3) input transfer matrices, U(s) represent a unit (3*3) matrix of measurement thresholds, and Y(s) represents a unit (3*3) matrix of measured values

The overall system objective of the FDI model is thus to modify the fault condition control law

$$< O, C(\Theta_p), U> \qquad 3.31$$

Where, O is the system objective, C (θ_p) are a set of constraints that affect the system from achieving its objectives and U is a set of algorithms that can handle the constraints in C (θ_p) to enable the system achieve its objective (O), a new control law with a degradable performance implements a new algorithm from the set of algorithms U, such that this FDI control law is modified to become

$$< O, C(\Theta_p), U(\phi)> \qquad 3.32$$

Where U (ϕ) is a set of modified algorithms that can handle the fault condition in C (Θ_p), thus ensuring that the initial control law of $< O, C(\Theta_p), U >$ in fault Free State is achieved.

The application of any isolative/ fault accommodation algorithm can only be possible when the particular fault condition in the constraints has been detected.

From the residual generator shown in (eq10) E (s) and F(s) are generally transfer matrices used to compare inputs and outputs in a control system. The choice of the matrices must capture the transfer function H(s).

When $$E(s) = -F(s)H(s) \qquad 3.33$$

This expression then implies that

$$H(s) = \frac{E(s)}{-F(s)} \qquad 3.34$$

This implies that the tranfer matrix function is representative of the negative output transfer matrix – F(s) and the input transfer matrix E(s)

In a fault free state, the residual generator should have a zero value. This zero value indicates that the actual output of the sensor in optimal condition is the same as the measured output of the sensor. A non-zero residual indicates a fault and thus causes an imbalance to the system.

The input and output transfer matrices are a matrix containing possible conditions that can affect the behaviour of the system. With respect to this study we will be looking at two types of sensors faults which can result in a non-zero residual.

- Off Set Bias Sensor Fault
- Variance Degradation Sensor Fault (Balzano, 2006)

The variance degradation sensor fault results due to ageing of a sensor or a factory design error. This could result in a drop out fault or in essence complete failure of the sensor to output any readings after a period of time which is a clear evidence of sensor degradation due to age and usage.

The off-set bias fault is an additive sensor fault that occurs when external factors modify the output of the sensor. The presence of dust and other environmental conditions can result in an off -set bias to a sensor reading as well as the presence of component / environmental noise. This condition is primarily responsible for fault conditions (F-1) which represents a negative or subtractive off set bias sensor fault arising when the measured value is less than the lowest threshold, (F-2) which represents positive or additive off-set bias faults arising when the sensor measurement is greater than the highest threshold value and the (F-3) which represents variance degradation faults arising when the measured values lies within the corrected range but is an outlier in comparison with previous measurements (non-correlative).

These two conditions will be represented in the input and output matrices to show how they affect the measured data by creating a non-zero residual, and also how the system can modify them to get a zero residual.

From (eq 3.30)

$$r(s) = E(s)u(s) + F(s)y(s)$$

When, $E_{(s)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $F_{(s)} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$

E(s) is a positive unit transfer matrix, while F(s) is a negative unit transfer matrix because to generate a residual between two terms, subtraction must occur amongst the two values being considered. The use of a (3*3) matrix for this analysis is because we have three fault scenarios namely F-1, F-2 and F-3 which have been previously defined as off-set bias faults and variance degradation faults respectively.

The first row in the matrix will be modified for fault condition (F-3), the second row in the matrix will be modified for fault condition (F-2), and while the last row in the matrix will be modified for fault condition (F-1), these are all due to possible relationships between the system components and possible failure states. The rows in the input transfer matrices contains the following information

$$\begin{pmatrix} Sensor \\ Controller \\ Environment \end{pmatrix}$$

For any value of U(s) and Y(s) which is also a 3*3 matrix as shown below, the residual generator will always give a zero reading. Let us assume for example that CO measurement was 75mg/m^3, the lowest measurable threshold for the CO sensor in this application is also 75mg/m^3. Thus

$$if,\ u_{(s)} = 75 * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, then \quad u_{(s)} = \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix} \qquad 3.31$$

And

$$y_{(s)} = 75 * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, then \quad y_{(s)} = \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix}$$

Where u(s) contains the lowest threshold value for the CO sensor which is 75mg/m^2, and y(s) contains the actual measured value for the CO pollutant which in this case is also 75mg/m^2, thus deducing from(eq3.30).

$$\text{Then,} \quad r_s = \left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix} \right) \qquad 3.32$$

$$, \quad r_s = \left(\begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix} + \begin{pmatrix} -75 & 0 & 0 \\ 0 & -75 & 0 \\ 0 & 0 & -75 \end{pmatrix} \right) \qquad 3.33$$

$$, \quad r_s = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad 3.34$$

Therefore, the residual from the generator is zero. In the first case we have a fault condition (F-1) when D_{CO} = 40. When the L_{CP} for CO is 75

The residual is thus represented below where only the affected row is input with faulty data.

$$r_s = \left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 40 \end{pmatrix} \right) \qquad 3.35$$

$$r_s = \left(\begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix} + \begin{pmatrix} -75 & 0 & 0 \\ 0 & -75 & 0 \\ 0 & 0 & -40 \end{pmatrix} \right) \qquad 3.36$$

$$r_s = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 35 \end{pmatrix} \qquad 3.37$$

Therefore to get zero –residual, the input matrix of the sensor should be modified as follows

$$r_s = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & .53333 \end{pmatrix} \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 75 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 40 \end{pmatrix} \Bigg) \qquad 3.38$$

$$r_s = \left(\begin{pmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 40 \end{pmatrix} + \begin{pmatrix} -75 & 0 & 0 \\ 0 & -75 & 0 \\ 0 & 0 & -40 \end{pmatrix} \right) \qquad 3.39$$

$$r_s = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad 3.40$$

Thus for the first detected fault case F-1, the system is able to determine that it didn't produce a zero residual R(s) as shown above and also it didn't produce a negative term which indicates that the measured value is greater than the lowest threshold for the sensor (L_{cp}). Thus in order to push the residual to zero, the controller deduces ($\frac{40}{75}$ = 0.53333). This value means a decrement in the unit transfer matrix value from (1 to 0.5333). This enables R(s) = 0. Thus the supervisory controller must behave in the following ways as represented in the flow chart in fig 3.16.

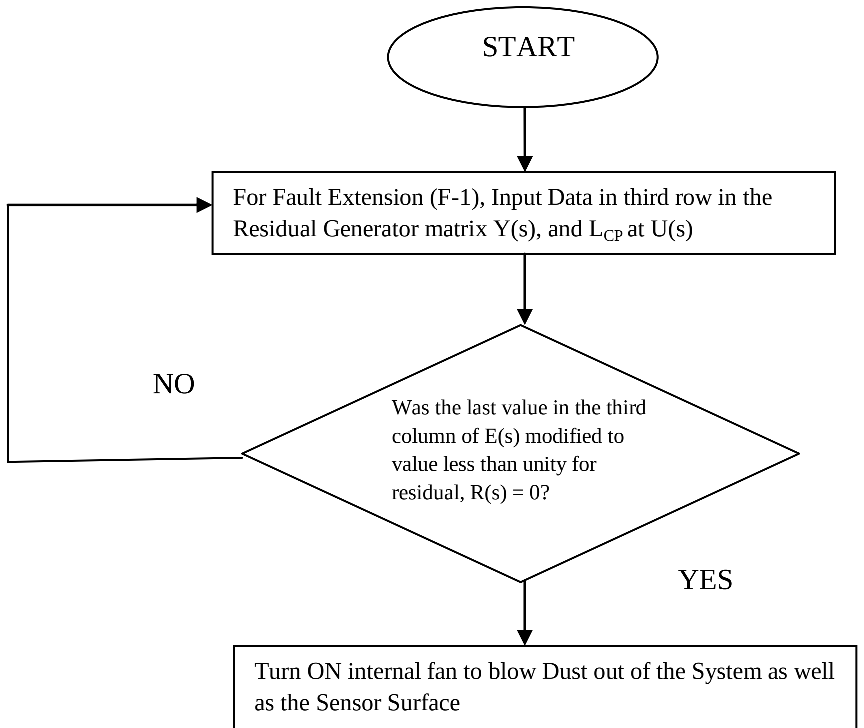

START
For Fault Extension (F-1), Input Data in third row in the Residual Generator matrix Y(s), and L_CP at U(s)
NO
Was the last value in the third column of E(s) modified to value less than unity for residual, R(s) = 0?
YES
Turn ON internal fan to blow Dust out of the System as well as the Sensor Surface

Figure 3.16: Flowchart Showing (F-1) Fault Handling.

Under the F-1 condition, the environment row was modified in the error residual generator because the presence of dust on the sensitive layer of the sensor might have affected its sensitivity and added a negative offset bias to the reading. The same applies to the heating temperature of the internal sensor heater being affected by the presence of dust. Thus the use of a cooling fan to blow off dust is an appropriate control action.

In the second case when we have the fault condition (F-2) occurring in table 4.6, D_{NO} = 120, U_{CP}= 107

The residual is thus

$$r_s = \left(\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}107 & 0 & 0\\ 0 & 107 & 0\\ 0 & 0 & 107\end{pmatrix} + \begin{pmatrix}-1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & -1\end{pmatrix}\begin{pmatrix}107 & 0 & 0\\ 0 & 120 & 0\\ 0 & 0 & 107\end{pmatrix}\right) \qquad 3.41$$

$$r_s = \left(\begin{pmatrix}107 & 0 & 0\\ 0 & 107 & 0\\ 0 & 0 & 107\end{pmatrix} + \begin{pmatrix}-107 & 0 & 0\\ 0 & -120 & 0\\ 0 & 0 & -107\end{pmatrix}\right) \qquad 3.42$$

$$r_s = \begin{pmatrix}0 & 0 & 0\\ 0 & -13 & 0\\ 0 & 0 & 0\end{pmatrix} \qquad 3.43$$

The negative sign indicates explicitly that the measured variable is higher than the highest measurable threshold for the study (U_{cp}). This means that an external noise was added to the actual value to the tune of 13. Therefore to get zero- residual, the system must modify the input matrix as follows

$$r_s = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1.121 & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 107 & 0 & 0 \\ 0 & 107 & 0 \\ 0 & 0 & 107 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 107 & 0 & 0 \\ 0 & 120 & 0 \\ 0 & 0 & 107 \end{pmatrix} \right) \qquad 3.44$$

$$r_s = \left(\begin{pmatrix} 107 & 0 & 0 \\ 0 & 120 & 0 \\ 0 & 0 & 107 \end{pmatrix} + \begin{pmatrix} -107 & 0 & 0 \\ 0 & -120 & 0 \\ 0 & 0 & -107 \end{pmatrix} \right) \qquad 3.45$$

$$r_s = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad 3.46$$

Thus in order to push the residual to zero, the controller deduces ($\frac{120}{107}$ = 1.121). This value means an increment in the unit transfer matrix value from (1 to 1.121). This enables R(s) = 0. Thus the supervisory controller must behave in the following ways in order to correct all possible anomalies as represented in the flow chart in fig 3.17

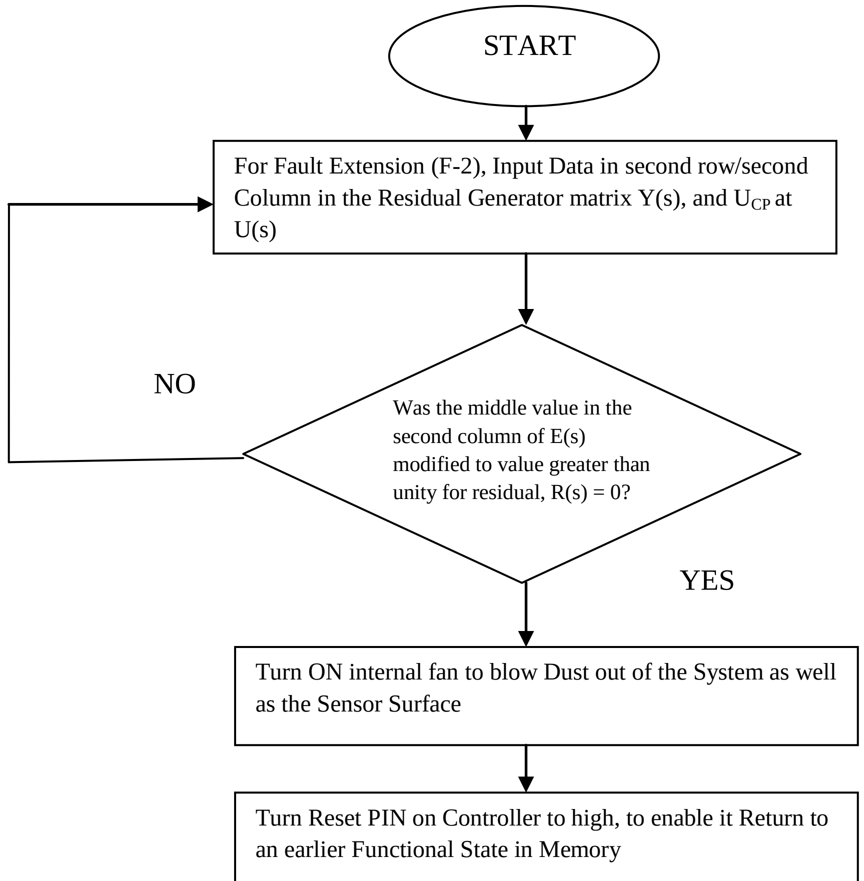

START
For Fault Extension (F-2), Input Data in second row/second Column in the Residual Generator matrix Y(s), and U_{CP} at U(s)
NO
Was the middle value in the second column of E(s) modified to value greater than unity for residual, R(s) = 0?
YES
Turn ON internal fan to blow Dust out of the System as well as the Sensor Surface
Turn Reset PIN on Controller to high, to enable it Return to an earlier Functional State in Memory

Figure 3.17: Flowchart Showing (F-2) Fault Handling.

The F-2 condition was analyzed in the controller column of the residual generator; because this problem is most probably a controller fault, due to improper exception handling in the software program impeding its ability to report correctly after a number of iterations or an additive noise in the measured reading due to improper calculations/conversions by the controller. A reading higher than U_{CP} is a perfect example of a malfunctioning control program or an additive offset bias fault.

The fault condition (F-3) occurred in table 4.5 when $D_{CO} = 187$, $L_{CP} = 75$, $U_{CP} = 450$,

When the span between the set points is 450-75 = 375, the difference between the measured value and the highest set point is 450-187=263, the difference between the measured value and the lowest set point is 187-75=112. The actual measured value is used in the matrix Y(s), while the differences between that value and the set points are used instead of zero, while the span between the set points is used in the matrix U(s) together with the upper and lowest threshold values of the CO pollutant for the sensor. The resulting expression due to eq 3.30 is thus

$$r_s = \left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 375 & 75 & 450 \\ 450 & 375 & 75 \\ 75 & 450 & 375 \end{pmatrix} + \begin{pmatrix} -1 & -1 & -1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 187 & 112 & 263 \\ 263 & 187 & 112 \\ 112 & 263 & 187 \end{pmatrix} \right) \qquad 3.47$$

The first role in F(s) are all made negative unity (-1), this is because the possible causes for why the data can lie between the sensor (threshold) set point values and yet is not correlative with previous weighted measurements due to change in process operation or variance degradation sensor fault. The matrices are generated by additions and subtraction between the measured data and the set points. 375 is chosen as the mid matrix value in the actual sensor value U(s) as it indicates the measurement span. The fault matrix Y(s) has values on the entire first row because this fault condition is likely dependent on the sensor, controller and environment.

$$r_s = \left(\begin{pmatrix} 375 & 0 & 0 \\ 0 & 375 & 0 \\ 0 & 0 & 375 \end{pmatrix} + \begin{pmatrix} -562 & -562 & -562 \\ -263 & -187 & -187 \\ -112 & -263 & -187 \end{pmatrix} \right) \qquad 3.48$$

$$r_s = \begin{pmatrix} -187 & -562 & -562 \\ -263 & 188 & -187 \\ -112 & -263 & 188 \end{pmatrix} \qquad 3.49$$

The residual is observed to be a non-zero term, to push the value to zero, the system must modify the input matrix E(s) as shown.

$$r_s =$$

$$\left(\begin{pmatrix} 0.624 & 0.624 & 0.624 \\ 0 & 0.498 & 0 \\ 0 & 0 & 0.498 \end{pmatrix} \begin{pmatrix} 375 & 75 & 450 \\ 450 & 375 & 75 \\ 75 & 450 & 375 \end{pmatrix} + \right.$$

$$\left. \begin{pmatrix} -1 & -1 & -1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 187 & 112 & 263 \\ 263 & 187 & 112 \\ 112 & 263 & 187 \end{pmatrix} \right) \qquad 3.50$$

The system simply modifies the first row on the matrix to get zero values, while it changes all zeros in the matrices below to zero. Thus

$$r_s = \left(\begin{pmatrix} 562 & 562 & 562 \\ 0 & 187 & 0 \\ 0 & 0 & 187 \end{pmatrix} + \begin{pmatrix} -562 & -562 & -562 \\ -263 & -187 & -187 \\ -112 & -263 & -187 \end{pmatrix} \right) \qquad 3.51$$

$$r_s = \begin{pmatrix} 0 & 0 & 0 \\ -263 & 0 & -187 \\ -112 & -263 & 0 \end{pmatrix} \qquad 3.52$$

Since there are still non-zero values in the matrix, the system will attempt to force the residual to zero by

$$r_s = \begin{pmatrix} 0 & 0 & 0 \\ 263 & 0 & 187 \\ 112 & 263 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ -263 & 0 & -187 \\ -112 & -263 & 0 \end{pmatrix} \qquad 3.53$$

$$r_s = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad 3.54$$

The added matrix $R_s = \begin{pmatrix} 0 & 0 & 0 \\ 263 & 0 & 187 \\ 112 & 263 & 0 \end{pmatrix}$ is a redundant version of the error residual, thus this implies the solution of this problem is for the system to switch to the redundant sensor in the data acquisition system.

In fault condition (F-3), the matrix is modified for there to be no residual as observed, But if there is a residual, then the system uses a mirror image of the residual to push it to a zero value. This action is the case with respect to using redundancy to solve a problem as shown in fig 3.18.

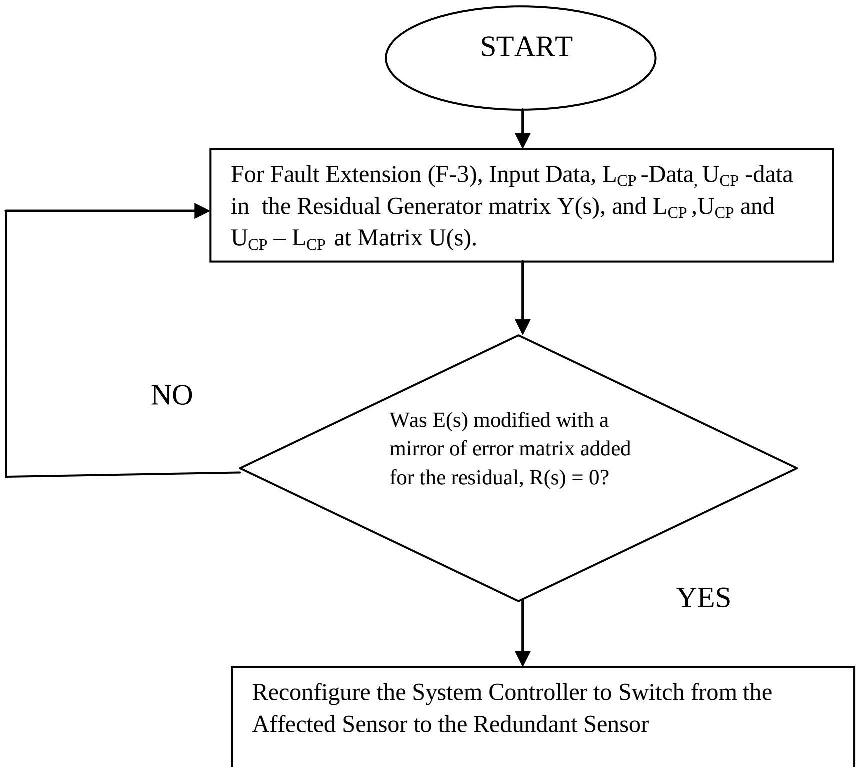

START
For Fault Extension (F-3), Input Data, L_{CP} -Data, U_{CP} -data in the Residual Generator matrix Y(s), and L_{CP} ,U_{CP} and $U_{CP} – L_{CP}$ at Matrix U(s).
NO
Was E(s) modified with a mirror of error matrix added for the residual, R(s) = 0?
YES
Reconfigure the System Controller to Switch from the Affected Sensor to the Redundant Sensor

Figure 3.18: Flowchart Showing (F-3) Fault Handling

3.4 Central Remote Terminal Unit

This section is simply saddled with the task of receiving the output data stream from the DAS unit, sending feedback data stream to the DAS and sending analyzed data to the data analytics unit. It performs a simple comparison to determine data correlation, before onward delivery to a database. This section contains a microcontroller, a GSM modem and a GPRS Modem. The block diagram shown in fig 3.16 details its structure

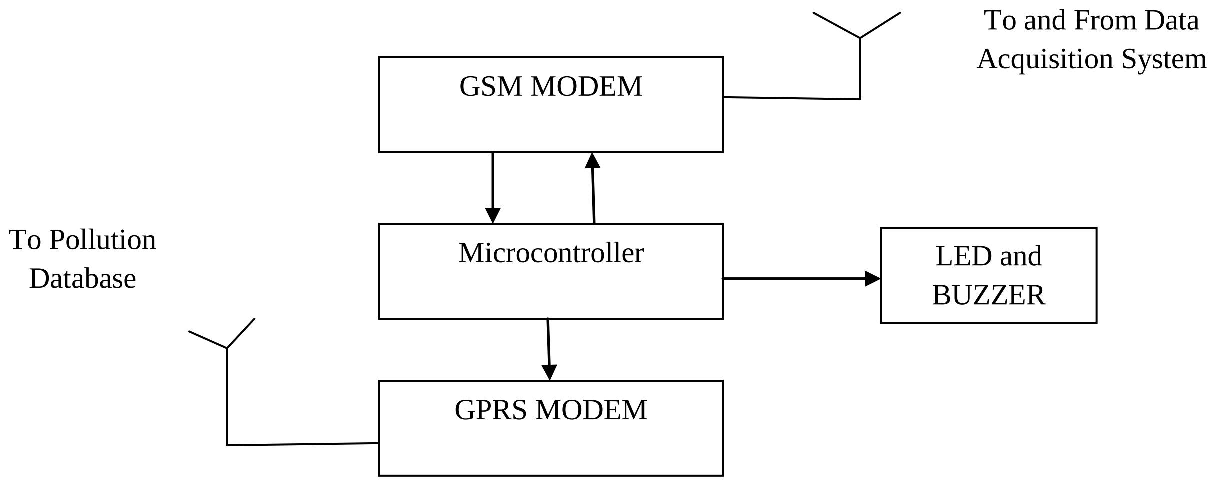


Figure 3.19: Block Diagram of the Remote Terminal Unit.

3.4.1 Hardware Design

The design for this unit utilizes a dual in line 40 pin microcontroller due to large number of I/O connection needed to accomplish its task, prior to using the controller; a 12MHz quartz crystal is connected with a 33pF capacitor to XTAL1 and XTAL2 on the microcontroller to provide the clock signal. U1 in figure 3.17 represents the CTU microcontroller, the controller is powered by a 5V input (V_{cc}) provided by a 5Vpower supply circuit shown in figure 3.5.

The reset pin to the controller which serves as the next input control signal for operation of the microcontroller is connected as shown in figure 3.20. When the 5V power is switched on, the capacitor shorts to 5V, and then gradually the RC circuit discharges to bring the reset pin to 0.

Since there is no requirement for an external memory in this application, thus the EA pin in the controller is set to HIGH by connecting it to Vcc. PSEN and ALE are all left alone (unconnected). The microcontroller is connected to two MAX232 ICs to enable data transfer from a GPRS and GSM modem. The LED, buzzer, GSM modem, GPRS modem were connected as described by figures 3.6, 3.7, 3.8 and 3.9.

The central terminal unit hardware was implemented using an ATMEL 89C52 controller, GPRS and GSM modules, a generic LED, generic buzzer and a 10WATT1K resistor. The system receives measured data from the data acquisition system via the GSM modem. It implements the first part of the FDI scheme of data variance and deviation analysis as described in chapter three. Depending on the result of the analysis, a feedback message is sent back to the data acquisition unit via the GSM modem while the result is sent to a database via the GPRS modem. The system is usually domiciled at a central point in a deployment area. The CTU controller is connected to the MAX232 IC's via ports P1.6 and P1.7 and it connects to the buzzer and LED via ports P0.4 and P2.6.

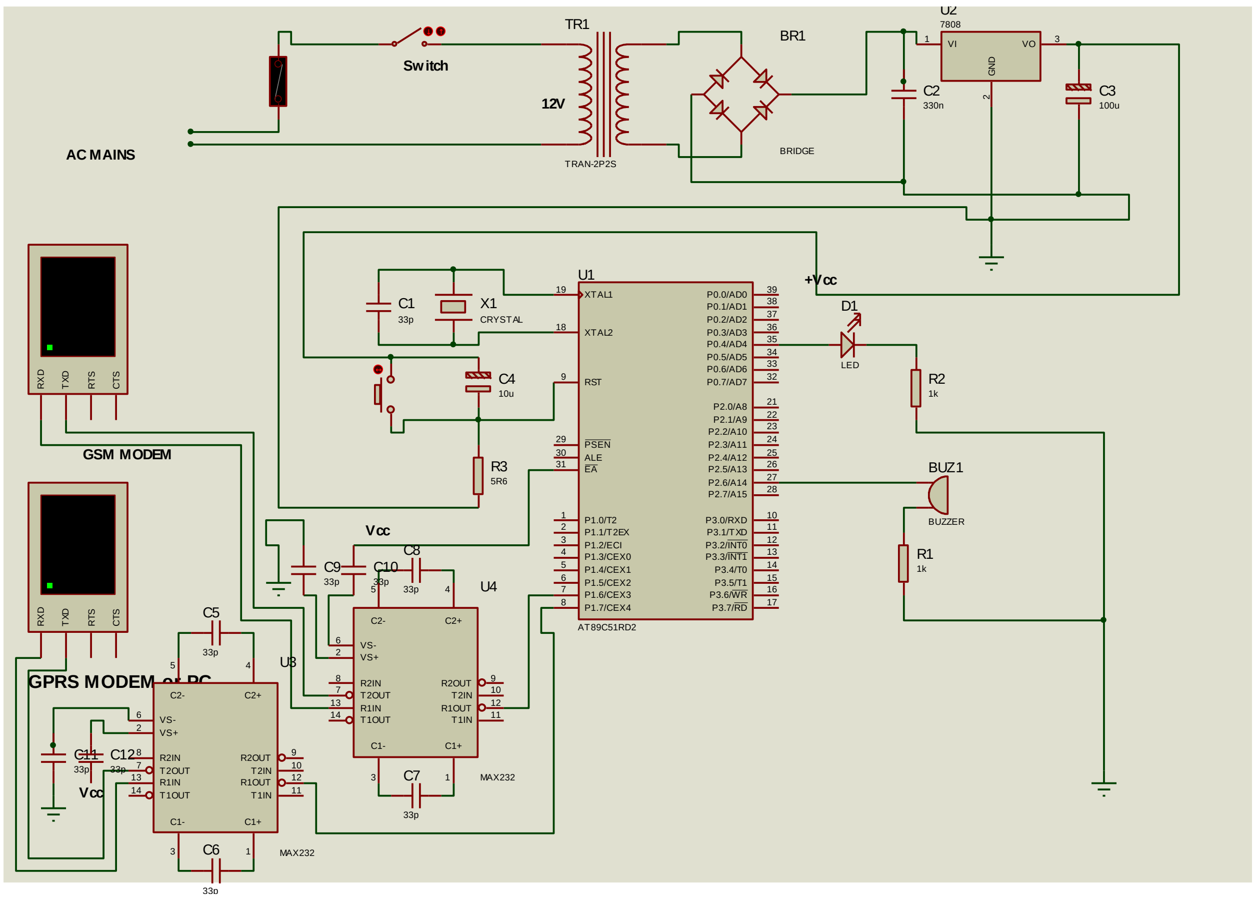


Figure 3.20: Central Remote Terminal Unit (CTU) Hardware.

3.4.2 Software Design for the Central Terminal Unit

CTU Microcontroller

The flow chart in fig 3.21 outlines the algorithm executed by the central remote terminal unit with the following terms defined as

D_{CO} = Measured CO Pollutant Concentration Data

D_{NO} = Measured NO Pollutant Concentration Data

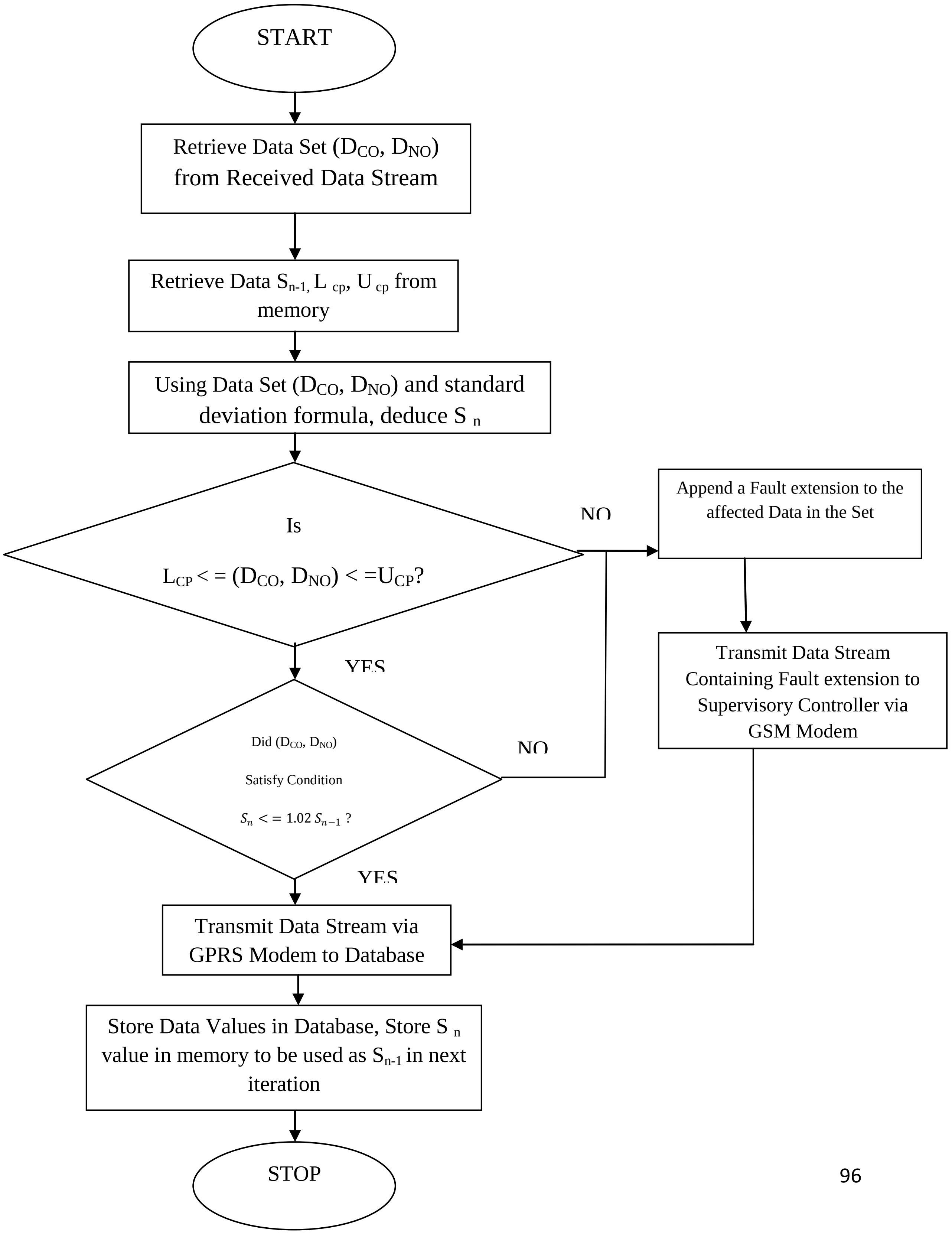

START
Retrieve Data Set (D_{CO}, D_{NO}) from Received Data Stream
Retrieve Data S_{n-1}, L_{cp}, U_{cp} from memory
Using Data Set (D_{CO}, D_{NO}) and standard deviation formula, deduce S_n
Is $L_{CP} <= (D_{CO}, D_{NO}) <= U_{CP}$?
NO
Append a Fault extension to the affected Data in the Set
YES
Transmit Data Stream Containing Fault extension to Supervisory Controller via GSM Modem
Did (D_{CO}, D_{NO}) Satisfy Condition $S_n <= 1.02\, S_{n-1}$?
NO
YES
Transmit Data Stream via GPRS Modem to Database
Store Data Values in Database, Store S_n value in memory to be used as S_{n-1} in next iteration
STOP

Figure 3.21: Flow Chart for the Central Remote Terminal.

Other terms are;

L_{CP} = Lowest Calibration Value for the Sensor According to Data Sheet

U_{CP} = Highest Calibration Value for the Sensor According to Data Sheet

(F1) = Fault Extension identifying Condition when (D_{CO}, D_{NO}) < L_{CP}

(F2) = Fault Extension Identifying Condition when (D_{CO}, D_{NO}) > U_{CP}

(F3) = Fault Extension Identifying Condition of Sensor Data un-correlativeness

S_n = Standard Deviation of measured data from mean

S_{n-1} = Standard Deviation of immediate previous measured data from mean

The pseudo code described in fig 3.22 shows exactly how the comparison were made

```
START
READ GSM Modem for Data Stream from GSM modem
READ $L_{cp}$, $U_{cp}$ and $S_{n-1}$ from memory
DEDUCE $S_n$ for Data ($D_{CO}$, $D_{NO}$) using Standard deviation/correlation formula
IF Data ($D_{CO}$, $D_{NO}$) Received is less than $L_{CP}$
THEN send Data ($D_{CO}$, $D_{NO}$) to Database with an (F1) extension added to affected value in set
ALSO send Data ($D_{CO}$, $D_{NO}$) with (F1) extension to supervisory control unit via GSM Modem
IF Data ($D_{CO}$, $D_{NO}$) Received is Greater than $U_{CP}$
THEN send Data ($D_{CO}$, $D_{NO}$) to Database with an (F2) extension added to affected value in set
ALSO send Data ($D_{CO}$, $D_{NO}$) with (F2) extension to supervisory control unit via GSM Modem
IF $L_{CP}$ < Data ($D_{CO}$, $D_{NO}$) < $U_{CP}$
AND Data ($D_{CO}$, $D_{NO}$) satisfies $S_n <= 1.02\, S_{n-1}$
THEN send Data ($D_{CO}$, $D_{NO}$) to Database through GPRS Modem
ELSE send Data ($D_{CO}$, $D_{NO}$) to Database through GPRS Modem with (F3) Extension
AND send Data ($D_{CO}$, $D_{NO}$) to supervisory controller through GSM Modem with (F3) Extension
```

Figure 3.22: Pseudo-Code for CTU Microcontroller.

The determination of sensor faults from measured data is achieved by considering the correlativeness of the measured value.

Fault Detection Model in the CTU

The first part of the FDI technique considered in this study was implemented in the central remote terminal unit

In the FDI model considered at this stage, at nominal condition, the control law for the data acquisition system is thus

$< O, C, U>$ 3.55

Where, U implies the algorithm to be implemented to achieve the objective of measuring pollutant concentration in a fault free state of the data acquisition system and the onward transmission of measured data to a pollution server.

O, represents the set of objectives while C, represents the constraints the system behavior will satisfy over a period of time

$< O, C (\Theta_i), U>$ represents the control law in a fault free state 3.56

$< O, C (\Theta_p), U>$ defines the control law in a fault condition 3.57

The set of objectives for this system with respect to functionality are as follows

- Is the data acquisition system functional
- Is the acquired data consistent with previous data (correlation)
- Can the system function despite the occurrence of a fault or failure?

The constraints set C contains the following rules for fault detection and isolation for the data acquisition system

1. If the Data acquisition system transmits no data (drop out sensor failure mode)
2. If the transmitted data value is below the lowest threshold value (L_{CP}) of the sensor
3. If the transmitted data value is higher than the highest threshold value (U_{CP}) of the sensor
4. If the data value does not correlate with previous data due to large changes or spikes in pollution levels above past measurements.

The use of the standard deviation and variance method ensures that the sensor values do not deviate too far from the mean point at any given time. This study is based on measurements of pollution information in a fixed industrial environment. Thus because of the fixed nature of the operation there would be a similarity in the data measured as the days goes by, this similarity can thus be thought of as a pattern. This pattern thus means that there is a variance range and standard deviation between any new data and the mean point of all previous past data. Thus we can use this method to reveal suspicious data resulting from its un-correlation with previous measurements.

The arithmetic mean equation is thus

$$\mu = \frac{\sum_{n=1}^{k} X_n}{k} \qquad 3.58$$

Where k is the total number of data values and X_n represents the individual data values themselves ranging from n=1 till n=n.

Thus the standard deviation and variance is thus

$$S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} \qquad 3.59$$

Where S is the standard deviation, $(x - \bar{x})$ is the deviation between a data value and the mean.

Where $\bar{x} = \mu$

The variance is thus $\sigma = s^2$ 3.60

This implies that the variance σ, is simply $\frac{\sum(x-\bar{x})^2}{n} = S^2$ 3.61

Thus the system calculates the mean continuously throughout its life span while in a specific deployment. A reset of the central remote terminal unit will return this mean value to zero. Therefore the pseudo code in fig 3.23 represents the application of these equations

```
START
INPUT $\sum_{n-1}^{k} X_{n-1}$
ADD $\sum_{n-1}^{k}(X_{n-1} + X_n)$
INCREMENT k= k+1
CALCULATE $\mu_n = \frac{\sum_{n}^{k}(X_{n-1}+X_n)}{k+1}$
STORE $\sum_{n}^{k}(X_{n-1} + X_n) = \sum_{n-1}^{k} X_{n-1}$ for next iteration when $X_{n+1}$
IF n=k
CALCULATE $\frac{\sum(x-\bar{x})^2}{n}$
WHEN $\bar{x} = \mu_n$
THEN $\sigma = \sigma_n$
THEN $S_n = \sqrt{\sigma_n}$
STOP
```

Figure 3.23: Pseudo-code for Standard Deviation and Correlation Model.

The arithmetic mean is additive such that the term $\sum_{n-1}^{k} X_{n-1}$ is always stored and used to add the new X_n gotten in future, with k incremented in value by one. This ensures that the arithmetic mean used to deduce the variance and standard deviation is based on a large data set which would most likely reflect the overall data pattern.

For every new measurement, after comparison with the L_{CP} and U_{CP} and it is determined that they lie within the range of the MICS 4514 sensor, the σ_n and S_n is deduced. The data is considered to be normal if the following condition holds

$$S_n <= 1.02\, S_{n-1} \qquad 3.63$$

The use of hourly measurements implies that we can have large changes in readings with an addition of maximum measurement error tolerance in sensors of between 2% and 5% (Iancu, 2011).

The following data streams shown in fig 3.24 is the output header frame sent from the central remote terminal unit to the data analytics unit for onward analysis.

DAS ID	TIME	DATE	Latitude	Longitude	CO -Level	NO-Level
(8bits)	(16bits)	(24bits)	(24bits)	(24bits)	(8bits)	(8bits)

Figure 3.24: Data Stream Output to Database via GPRS Modem.

The data stream shown in fig 3.24 contains the information received from the data acquisition system with an extension of a fault set. The bit length for each header is generated as follows

If D_{co} and D_{no} can have values not more than 256, and can be represented by 8bits each, thus total number of bits required is 8+8=16bits.

The fault set contains typically (F-1, F-3), if "f" is represented by a byte of 8bits and the numbers by 3bits. The total bits are thus 8+3+8+3=22bits.This is transmitted to the pollution server via the GPRS Modem for storage and analysis.

DAS ID	TIME	DATE	(D_{CO}, D_{NO})	Fault Set
(8bits)	(16bits)	(24bits)	(16bits)	(22bits)

Figure 3.25: Data Stream Output to Supervisory Controller in the DAS.

The data stream shown contains information sent to the supervisory controller in the data acquisition system, to enable it implement an FDI action.

Complete FDI Data Acquisition System

The two tier data acquisition system hardware was designed and implemented to measure oxides of nitrogen and oxides of carbon at an industrial location. C programming language was used in the software design of the different tiers as shown in fig 3.26.

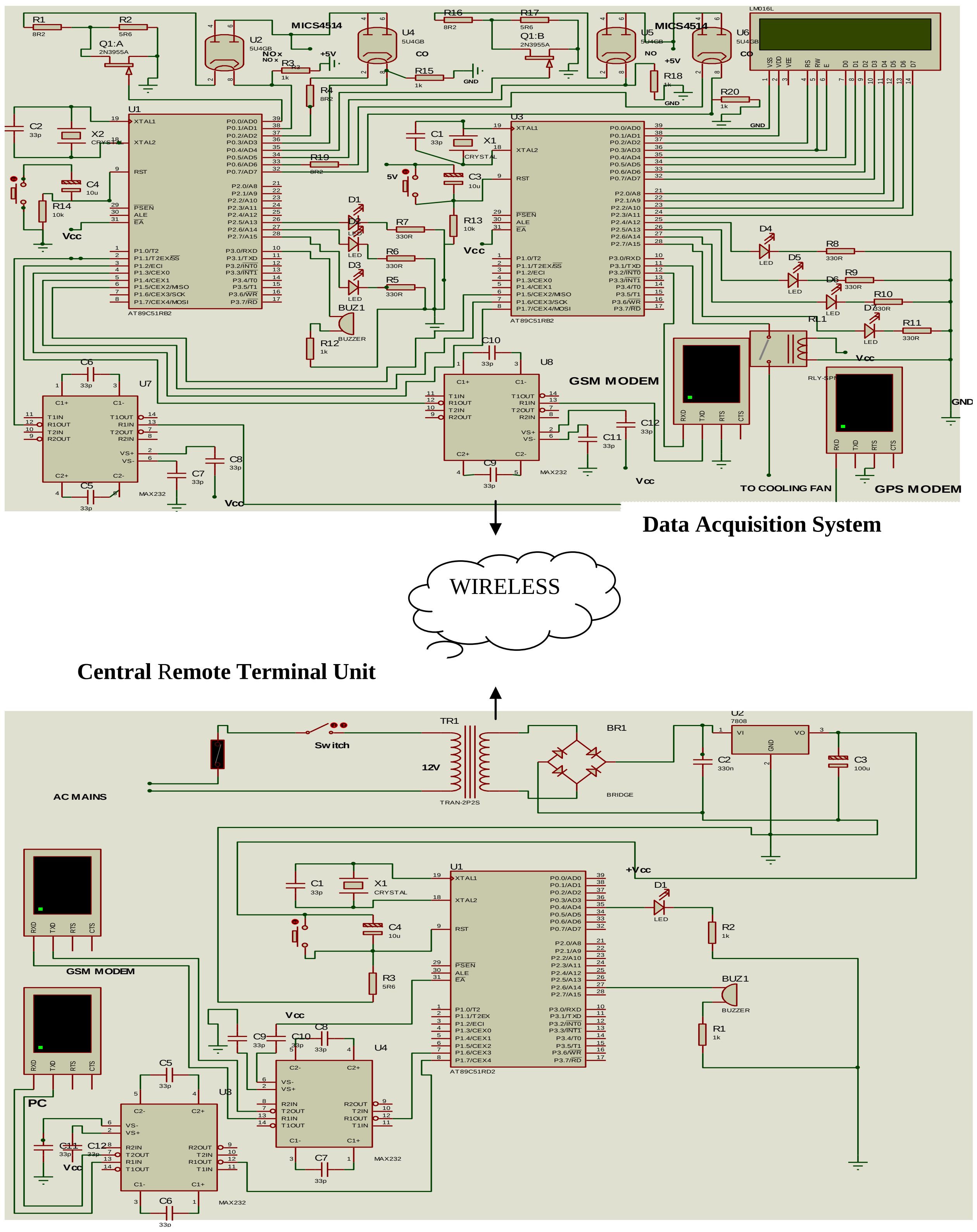


Figure 3.26: The Complete FDI Data Acquisition System for CO and NO Measurement.

3.5 Data Analytics Phase

An air pollution monitoring system would involve massive data collated over time where the primary challenge would now be how to convert the database of information from seemingly meaningless data into useful information. Data mining thus refers to the process of extracting or ‘mining’ knowledge from large amounts of data. It involves processes of exploration and analysis with the aim of discovering meaningful patterns and rules. Data mining can be viewed as an advanced stage of online analytical processing (OLAP) which in itself is a data summarization/aggregation tool that helps simplify data analysis. Data mining processes include human resource identification, problem specification, data prospecting, domain knowledge elicitation, methodology identification, data preprocessing, pattern discovery and knowledge post processing. The entire process in data mining can be summarized into the following actions taken:

- Retrieve the data from a large database.
- Select the relevant subset to work with.
- Decide on the appropriate sampling system.
- Clean the data and deal with missing fields and records.
- Apply the appropriate transformations, dimensionality reduction and projections.
- Fit models to the preprocessed data if necessary.

An air quality monitoring software (AQMS) was used in this research endeavor to mine the pollution data collated by the FDI data acquisition system deployed at an industrial location in Nigeria for trend analysis and air quality index assessment.

3.5.1 Hardware Specification

System Characteristics for AQMS Deployment

1. Memory size: It may not be too large, but a minimum of 128MB is recommended.
2. Hard Drive: large hard disk is required since it will serve as the main storage for the database.
3. Software Operating System: windows XP, window 2000, VISTA, 2003, 2007, 2008 can be used to install the program.

3.5.2 Software Specification

The AQMS Software

AQMS software is used in the analysis and forecasting of the pollutant data. The data used in the context of this research were samples of air pollution data from Ife steel plant. The collated data was grouped on an excel file based on the pollutant and the industry considered.

Software Functions for the AQMS Application

The AQMS application is meant to do the following:

i. Populate the database with air pollution data from Ife steel plant.
ii. Represent the results in a visual/graphical format using line graphs and scatter plots.

The flowchart in figure 3.27 shows the operation of the AQMS software.

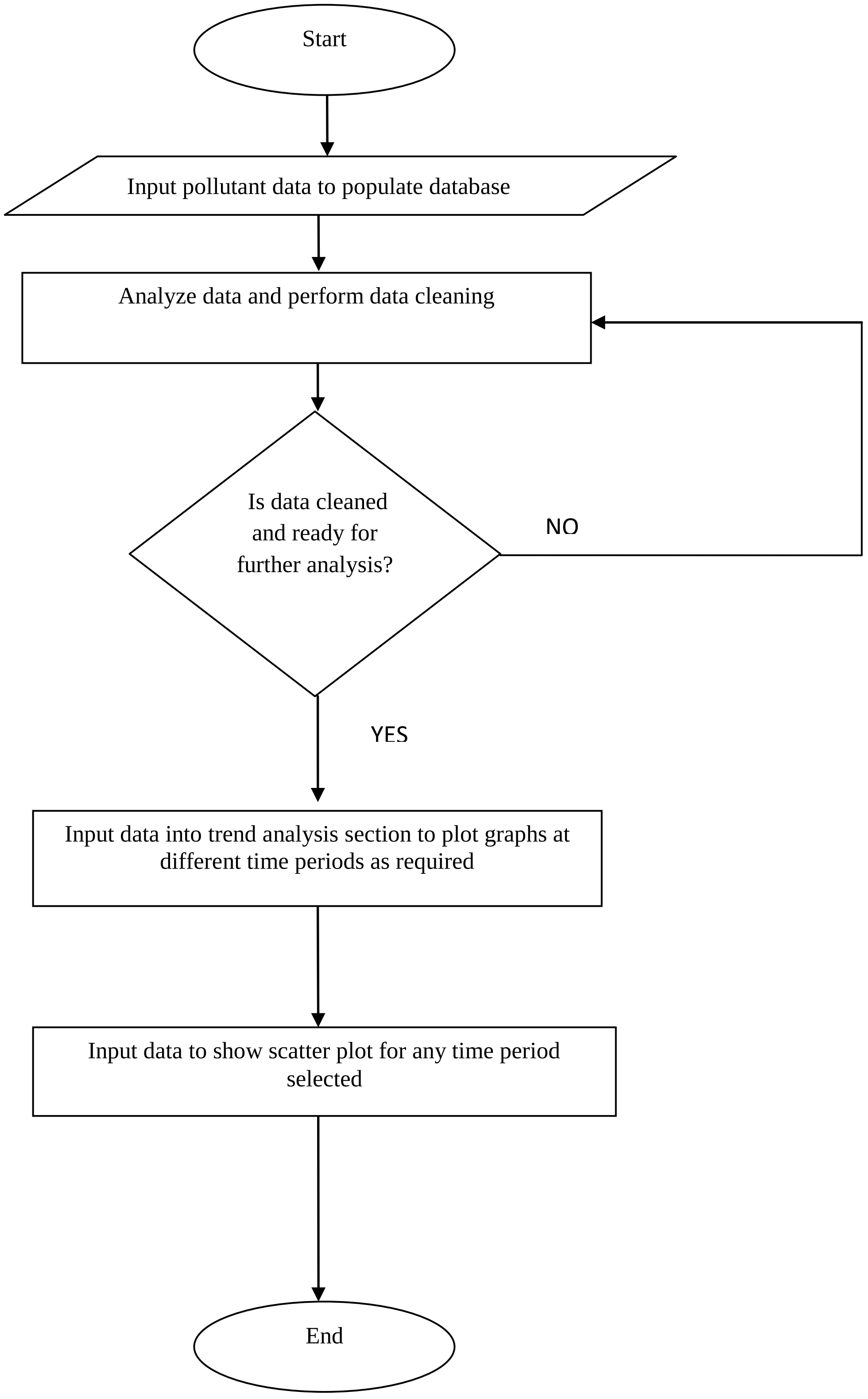

Start
Input pollutant data to populate database
Analyze data and perform data cleaning
Is data cleaned and ready for further analysis?
NO
YES
Input data into trend analysis section to plot graphs at different time periods as required
Input data to show scatter plot for any time period selected
End

Figure 3.27: AQMS software.

Figures 3.28-3.31 show the interfaces of the AQMS system used for analysis of the measured data and to show trend over any measurement period.

Configuration Section

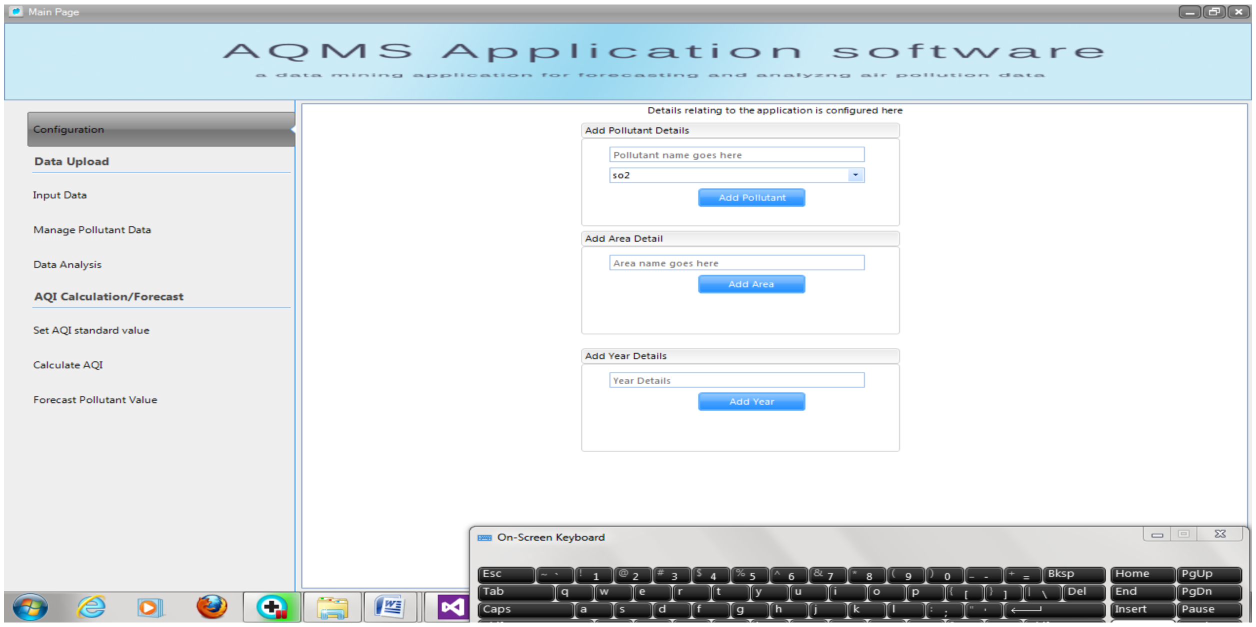


Figure 3.28: Configuration Section.

- Add Pollutant Details: This section enables the user to add pollutant details which would enable the system to determine the kind of pollutant data to analyze. Pollutant name could be NO etc while its unit could be ug/m^3 and mg/m^3.
- Add area/industry details: This section enables the user to add the details of the industry (name, location) whose pollution data would be considered later during data analysis.
- Add year details: This section enables the user to add the year which the measurement was taken.

Data Upload Section

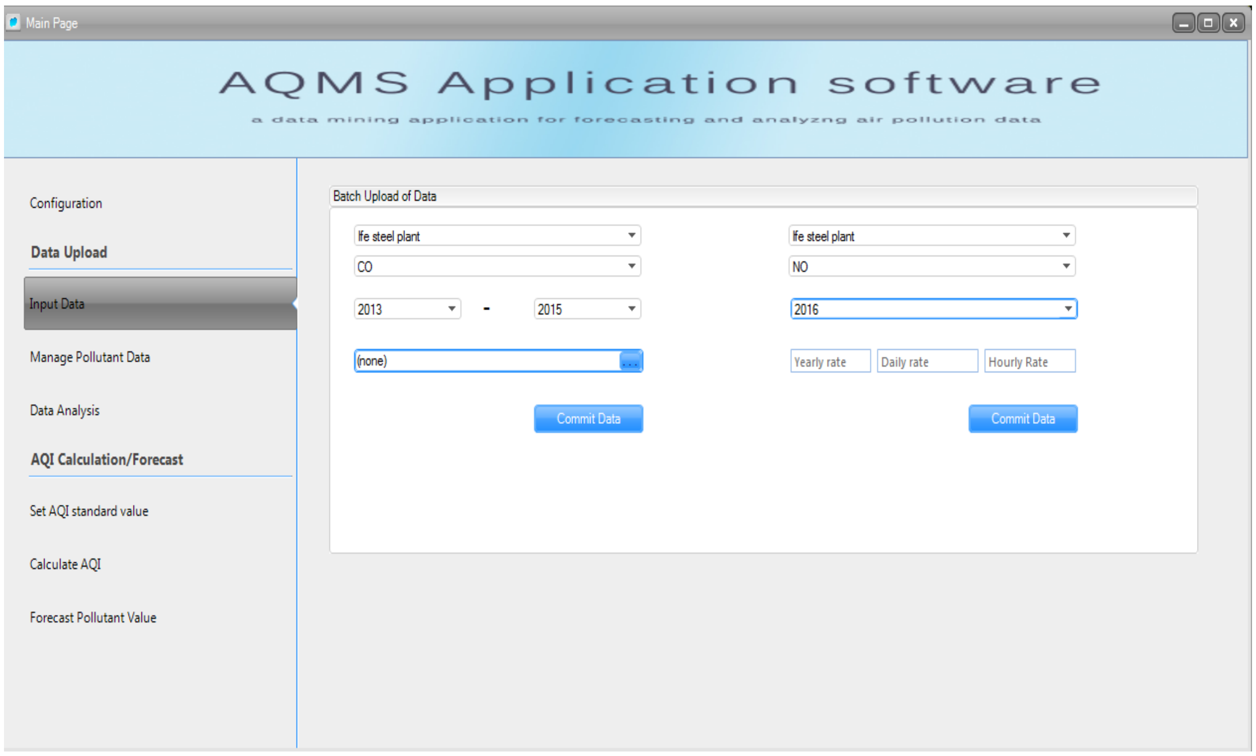


Figure 3.29: Uploading Pollutant Data.

Data upload section enables the user to supply pollutant data that would be used by the system to perform analysis and forecasting, Data to be uploaded can be in form of an excel file or supplied individually under a selected year, area and pollutant.

Manage Pollutant Section

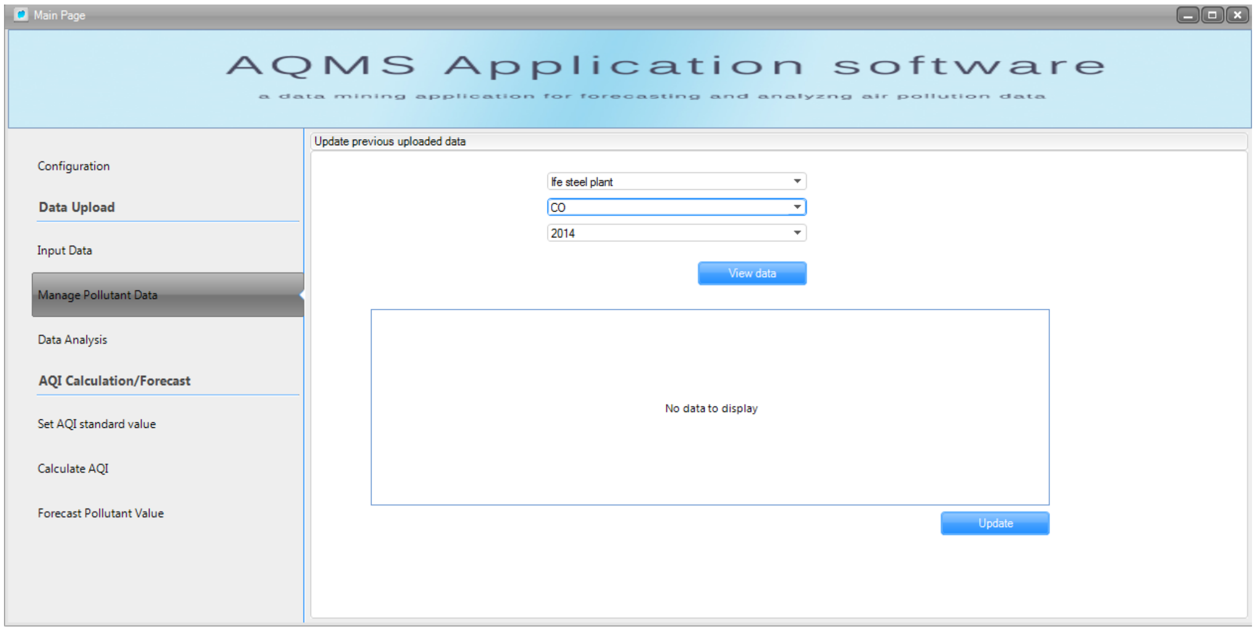


Figure 3.30: Manage Pollutant Section.

The manage pollutant section allows the user to view the daily, weekly, monthly or yearly average of measured pollutant data stored for that selected year.

Data Analysis

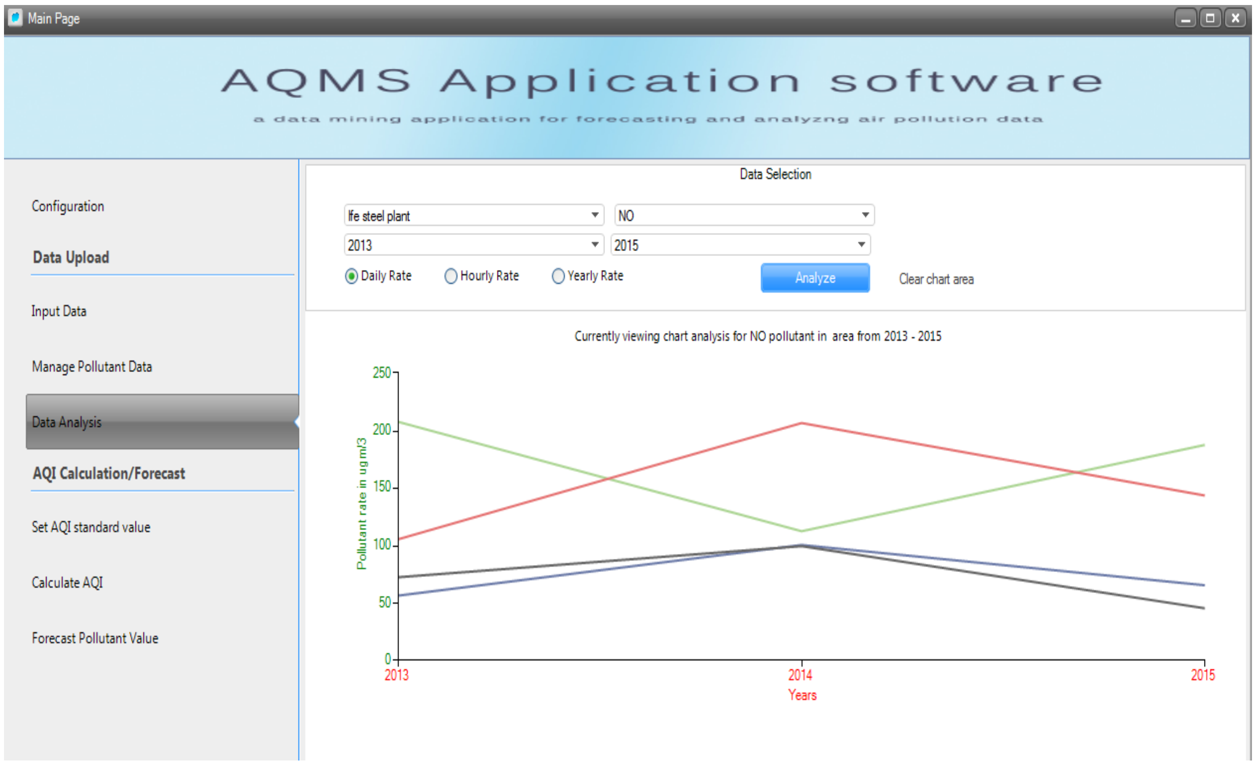


Figure 3.31: Data Selection and Analysis.

The data analysis section provides a graphical interface that helps to analyze pollutant data from a range of years selected and also area specified. The output is usually in graphical form and can cover numerous selections and time periods. Multiple graphs can be plotted at once on the same page, thus enabling comparison amongst the different sets of data being considered for the locations of study.

Data Cleaning and Transformation

Data cleaning and transformation is the most resource-intensive step in a data analysis endeavor. The purpose of data cleaning is to remove noise and irrelevant information out of the dataset. The purpose of data transformation is to modify the source data into different formats in terms of data types and values. There are various techniques that can be applied to perform data cleaning and data transformation, *Data Type* Transform: this is the simplest data transform. An example is transforming a Boolean column type to integer. The reason for this transform is that some data analysis algorithms perform better on integer data, while others prefer Boolean data.

Much of the raw data contained in excel sheet were un-pre-processed, incomplete, and noisy. For example, the data in excel format initially contained:

- Fields that are obsolete or redundant
- Missing values
- Outliers
- Data in a form not suitable for input to our analysis model
- Values with a fault condition extension which indicates low data confidence

Methodology used for Data Cleaning

Missing data is a problem that continues to plague data analysis methods. Even as our analysis methods gain sophistication, missing values in fields, especially in databases with a large number of fields are often encountered. The absence of information is rarely beneficial. All things being equal, more data is almost always better. Therefore, in order to handle the thorny issue of missing data, the following steps were taken such as replacing the missing value with some constant, replacing the missing value with the field mean (for numerical variables) or the mode (for categorical variables), replacing the missing values with a value generated at random from the variable distribution observed.

The other major factor was the transformation of data with a fault extension, the table below indicates the methodology used to handle data with fault extensions (F-1), (F-2) and (F-3)

TABLE 3.1: Faulty Data Cleaning Rules

Fault Conditioned Data	**Action**

(F-1)	Discard and ignore
(F-2)	Discard and ignore
(F-3)	Include if Occurrence is less than four times in an 11-hour shift

3.6 System Testing

3.6.1 System Hardware Testing

A drop out sensor fault was simulated by disconnecting one sensor from the circuit board during its transmit cycle. The DAS controller transmitted a null reading for D_{CO} and D_{NO} and this was detected by the central remote system's FDI model and fed back to the supervisory controller. This controller immediately redirected the DAS controller to measure from the redundant sensor while it attempted to resolve the sensor drop out fault.

Overall test revealed the following in table 3.2

Table 3.2: Black Body Test Result for System Hardware

COMPONENT/SUBSYSTEM	**STATUS**
DAS	**Sensors**: (1) Heater: Worked well with respect to design. **DAS microcontroller**: Performed its control function according to flow chart in fig 3.7 **Buzzer**: Sounded when alarm condition was met **LCD**: Displayed Measured Data Satisfactorily
Supervisory Controller	➢ Performed it oversight function on the

	overall DAS ➢ Implemented the error residual generator when a fault condition was observed
Central Remote Terminal unit	➢ Received and transmitted pollution data to and from DAS and Database respectively. ➢ Implemented its FDI model to received data in line with flow chart in fig 3.20
GPS Connection	Measured longitude and latitude information of location of measurement.
GSM modem Connectivity	➢ Used the MTN network service for transmission of data due to its availability. ➢ Modem performed satisfactorily in all modules, where it was implemented
GPRS Connectivity	➢ Used the MTN network service for transmission due to its availability ➢ Modem performed satisfactorily

3.6.2 System Software Testing

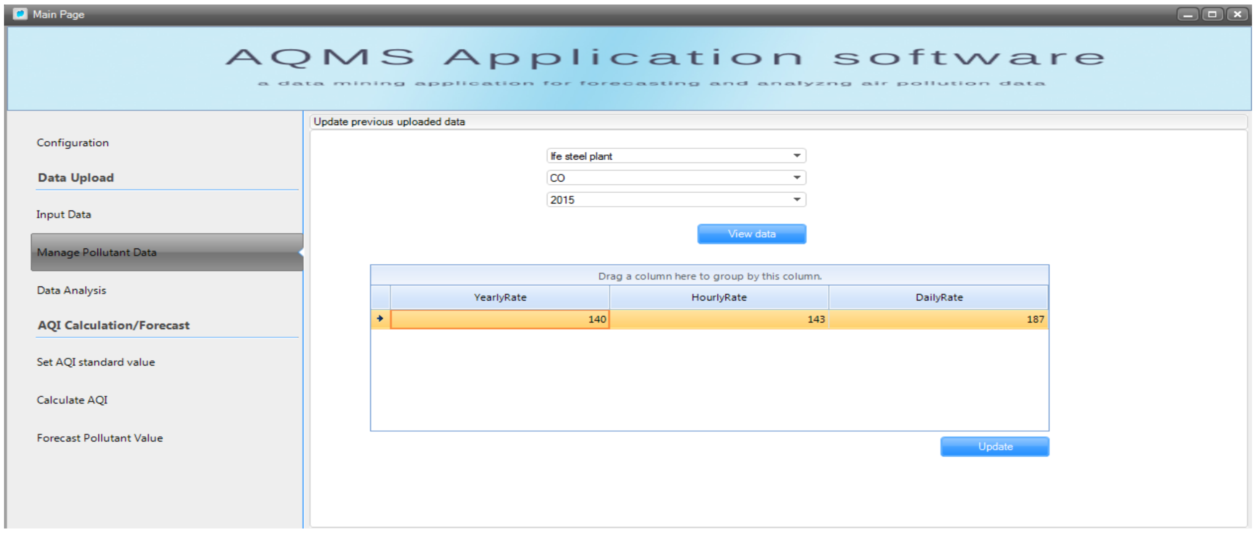
Data Acquisition Unit & Central Remote Terminal Unit

The software for the microcontrollers were written using the MIDE-51 editor and simulated on the Proteus 6 development platform. The source code was debugged many times during its development cycle to ensure that all possible sources for errors were eliminated. The modules were tested differently before integration. This integration resulted in system errors during runtime, but was corrected after repeated debugging cycles till no errors were observed.

Data Analytics Unit

Black body testing of the AQMS software revealed that the system was able to carry out its functions as detailed by the flowchart in figure 3.27. This check test considered the following in testing the performance of the AQMS

- **Accuracy Constraints**: The link to the database was checked and validated especially when the database is updated so as to have a current and updated report.



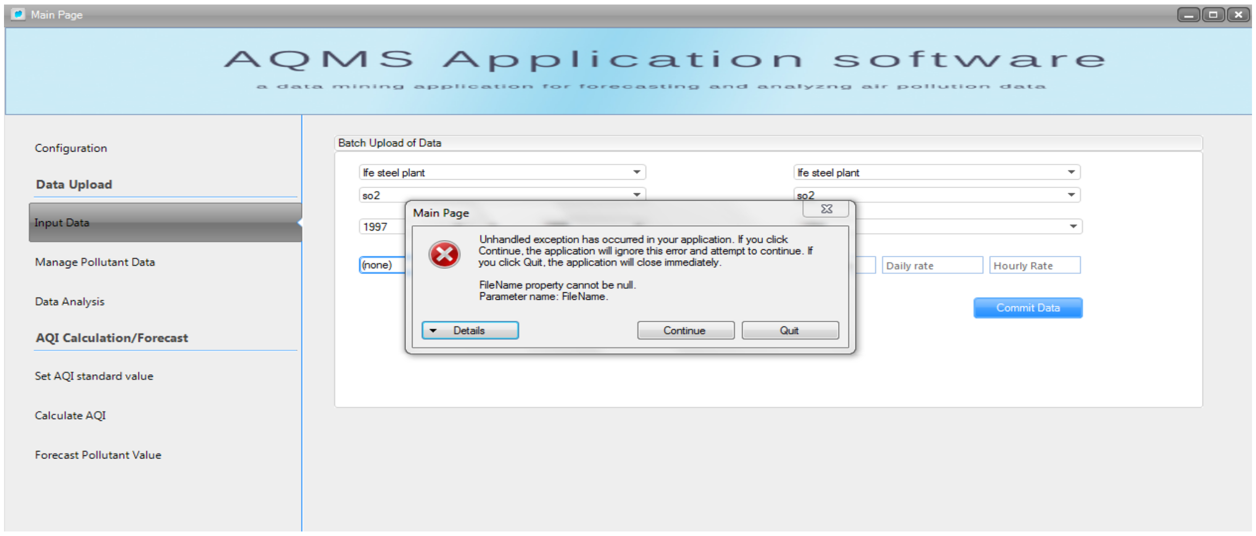
Ife steel plant
CO
2015

YearlyRate	HourlyRate	DailyRate
140	143	187

Figure 3.32: System Database linkage Update Check to Ensure Accuracy.

- **Response to undesired Events**: The software displayed a message for every wrong action on the software to notify the user of wrong operations.

➢ **Changes:** The software allows for modification in case of any changes to the information supplied by the user



Unhandled exception has occurred in your application. If you click Continue, the application will ignore this error and attempt to continue. If you click Quit, the application will close immediately.

FileName property cannot be null.
Parameter name: FileName.

Figure 3.33: System Error Response to Undesired Events and Wrong Operation

CHAPTER FOUR

RESULTS AND DISCUSSION

4.0Results and Discussion

Data Acquisition System Unit

Table 4.1 extracted from Appendix A, represents CO pollutant information (D(i)) measured on the 11/09/2014. Eq(6) was applied to the values in the table, where it depicted both alarm and non-alarm scenarios based on the comparison between the measured value and the defined set point for CO pollutant (IND-AQI Standard). By 11am and 12 noon, the E (t) was negative which showed that the measured values were less than the threshold and thus there was no alarm. Measurement from 1pm up till 4pm indicated an alarm situation.

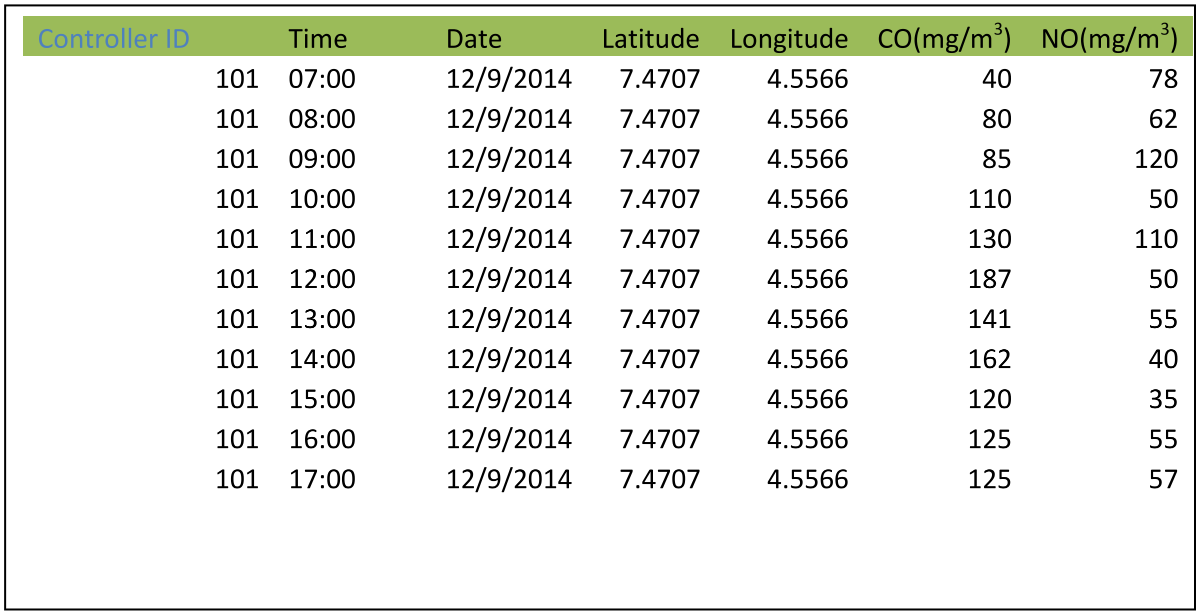
Table 4.1: Alarm Condition Testing

Date: 11/09/2014	D (i)	D (ref)	E (t)	ACTION
11:00	90mg/m^3	100mg/m^3	-10mg/m^3	NULL
12:00	99mg/m^3	100mg/m^3	-1mg/m^3	NULL
13:00	105mg/m^3	100mg/m^3	5mg/m^3	**ALARM**
14:00	120mg/m^3	100mg/m^3	20mg/m^3	**ALARM**
15:00	122mg/m^3	100mg/m^3	22mg/m^3	**ALARM**
16:00	110mg/m^3	100mg/m^3	10mg/m^3	**ALARM**

Table 4.2 represents data sent to the central remote terminal unit by the DAS. The same data is sent to the database for storage with an added fault extension. It can be observed from Table 4.2 that the measurements have no fault extension and this is what was measured by the DAS

directly from the sensors. The data contained the controller ID field, time of measurement, date of measurement, longitude and latitude information gotten from the GPS modem and lastly the pollutant concentrations for carbon monoxide and nitrogen oxide.

Table 4.2: Pollution Data Transmitted to Central Remote Terminal Unit on 12/09/2014



Controller ID	Time	Date	Latitude	Longitude	CO(mg/m^3)	NO(mg/m^3)
101	07:00	12/9/2014	7.4707	4.5566	40	78
101	08:00	12/9/2014	7.4707	4.5566	80	62
101	09:00	12/9/2014	7.4707	4.5566	85	120
101	10:00	12/9/2014	7.4707	4.5566	110	50
101	11:00	12/9/2014	7.4707	4.5566	130	110
101	12:00	12/9/2014	7.4707	4.5566	187	50
101	13:00	12/9/2014	7.4707	4.5566	141	55
101	14:00	12/9/2014	7.4707	4.5566	162	40
101	15:00	12/9/2014	7.4707	4.5566	120	35
101	16:00	12/9/2014	7.4707	4.5566	125	55
101	17:00	12/9/2014	7.4707	4.5566	125	57

Central Remote Terminal Unit

Table 4.3: Data Analysis of Measured Data with Respect to Thresholds

$V_{out\ (Volts)}$	R_s (KΩ)	PPM	(D_{CO}, D_{NO})	(L_{CP} < = D_{CO} < =	Output

			mg/m^3 12/09/2014	**U_{CP} , L_{CP} < = D_{NO} < = U_{CP})**	
(0.569, 0.207)	(2.6, 5.9)	(35.2,89.6)	(44, 120)	(L_{CP} > 44, U_{CP} < 120)	(44[F-1], 120 [F-2])
(0.35, 0.42)	(4.8, 2.5)	(64, 37.34)	(80,50)	(Valid, Valid)	(80, 50)
(0.332, 0.223)	(5.1, 5.4)	(68,82.2)	(85,110)	(Valid, Valid)	(85, 110)
(0.265, 0.42)	(6.6, 2.5)	(88, 37.34)	(110,50)	(Valid, Valid)	(110, 50)
(0.228, 0.394)	(7.8, 2.7)	(104, 41.07)	(130,55)	(Valid, Valid)	(130, 55)
(0.164, 0.499)	(11.2, 1.9)	(149.6, 29.9)	(187, 40)	(Valid, Valid)	(187,40)
(0.212, 0.54)	(8.5, 1.7)	(112.8, 26.13)	(141,35)	(Valid, L_{CP} > 35)	(141, 35[F-1])
(0.187,0.394)	(9.7, 2.7)	(129.6, 41.1)	(162,55)	(Valid, Valid)	(162, 55)
(0.245, 0.384)	(7.2,2.8)	(96, 42.5)	(120,57)	(Valid, Valid)	(120,57)
(0.237, 0.42)	(7.5, 2.5)	(100, 37.34)	(125,50)	(Valid, Valid)	(125,50)

Table 4.3 shows the analysis of measured readings on 12/09/2014 and the two defined set points (75, 450) for lower and upper set point respectively for CO and (40,107) for lower and upper set point for NO sensors. The condition stated in the fifth column was able to identify readings which fell under the three fault conditions earlier defined for variance degradation and offset bias sensor fault types. Data set (44, 120) for CO and NO when applied to the condition indicated

them to be under fault conditions F-1 and F-2 respectively, while data set (141, 35) only indicated an F-1 fault for the NO measurement. The rest of the measured data were all valid.

Table 4.4: Table Showing Un-Correlated CO Data from Variance Analysis

Date	**V_{out} (Volts)**	**R_s (KΩ)**	**PPM**	**D_{CO} (mg/m^3)**	**Variance (σ_{CO})**	***Standard Deviation* (S_{CO})**	***Immediate past Standard Deviation* $1.02\,S_{CO(n-1)}$**	**Fault Due to condition $S_{CO} \leq 1.02\,S_{CO(n-1)}$**
11/09/2014	0.2718	6.4	85.6	107	834.4014	28.88601		
11/09/2014	0.2886	6	80	100	827.1583	28.76036	29.3355	Valid
12/09/2014	0.6111	2.4	32	40	872.942	29.54559	29.3356	**(F-1)** $S_{CO} < 1.02\,S_{CO(n-1)}$
12/09/2014	0.3502	4.8	64	80	874.1256	29.56562	30.1635	Valid
12/09/2014	0.3324	5.1	68	85	872.1439	29.53208	30.1569	Valid
12/09/2014	0.265	6.6	88	110	863.2542	29.38119	30.1227	Valid
12/09/2014	0.2283	7.8	104	130	858.156	29.2943	29.9688	Valid

12/9/2014	0.1635	11.2	149.6	187	906.522	30.1085	29.8801	**(F-3)** $\boldsymbol{S_{CO}}$ $< 1.02\ \boldsymbol{S_{CO(n-1)}}$
12/09/2014	0.2121	8.5	112.8	141	905.8424	30.09722	30.7107	Valid
12/09/2014	0.1867	9.7	129.6	162	921.0434	30.3487	30.6991	Valid
12/09/2014	0.2454	7.2	96	120	912.6165	30.20954	30.9555	Valid
12/09/2014	0.2365	7.5	100	125	905.2705	30.08771	30.8137	Valid
12/09/2014	0.2365	7.5	100	125	898.0377	29.96728	30.6894	Valid

The table 4.4 shows clearly how the condition $S_n <= 1.02\ S_{n-1}$ is used to detect uncorrelated data that implies a fault condition from a weighted data pattern. At D_{co}= 40, it was observed that the condition did not hold and was flagged as fault condition F-1, while at D_{co}= 187, it was observed that the condition did not hold and was flagged as fault condition F-3. All other data values were deemed valid.

Table 4.5: Table Showing Un-Correlated NO Data from Variance Analysis

Date	**V_{out}** (volts)	**R_s(KΩ)**	**PPM**	**D_{NO} Mg/m^3**	**Variance** (σ_{NO})	$\boldsymbol{Standard\ Deviation}$ $(S_{NO}$	$\boldsymbol{Immediate\ past\ Standard\ Deviation}$ $\mathbf{1.02}\ \boldsymbol{S_{NO(n-1)}}$	**Fault** **Due to condition** $\boldsymbol{S_{nO}} \leq 1.02\ \boldsymbol{S_{NO(n-1)}}$
11/09/2014	0.42	2.5	37.3	50	143.8348	11.99312		
11/09/2014	0.369	2.9	44.8	60	142.3655	11.9317	12.2329	Valid
12/09/2014	0.298	3.8	58.3	78	145.193	12.04961	12.1703	Valid
12/09/2014	0.359	3.0	46.3	62	143.8595	11.99415	12.2906	Valid
12/09/2014	0.207	5.9	89.6	120	181.7576	13.48175	12.234	**(F-2)** $\boldsymbol{S_{NO}} > 1.02\ \boldsymbol{S_{NO(n-1)}}$
12/09/2014	0.42	2.5	37.3	50	180.6345	13.44003	13.7513	Valid
12/09/2014	0.223	5.4	82.2	110	205.4105	14.33215	13.7088	**(F-2)** $\boldsymbol{S_{NO}} > \mathbf{1.02}\ \boldsymbol{S_{NO(n-1)}}$
12/9/2014	0.42	2.5	37.3	50	204.1477	14.28803	14.6187	Valid
12/09/2014	0.394	2.7	41.1	55	202.2719	14.22223	14.5737	Valid
12/09/2014	0.499	1.9	29.9	40	203.7253	14.27324	14.5066	Valid
12/09/2014	0.54	1.7	26.1	35	207.1169	14.25829	14.5587	Valid
12/09/2014	0.394	2.7	41.1	55	205.2357	14.32605	14.5434	Valid
12/09/2014	0.384	2.8	42.6	57	203.2988	14.25829	14.6125	Valid

The table 4.5 shows clearly how the condition $S_n <= 1.02\, S_{n-1}$ is used to detect uncorrelated data that implies a fault condition from a weighted data pattern. At D_{no}= 120 (measured on 12/09/2014), it was observed that the condition did not hold and was flagged as fault condition F-2, while at D_{no}= 110(of same day), it was observed that the condition did not hold and was flagged as fault condition F-2.All other data values were deemed valid.

Table 4.6: Feedback Data Sent to Supervisory Controller in DAS unit by CTU on 12/09/2014

Controller ID	Time	Date	(Dco, Dno)	Fault
101	07:00	12/9/2014	40,78	F-1, ACK
101	08:00	12/9/2014	80,62	ACK, ACK
101	09:00	12/9/2014	85,120	ACK,F-2
101	10:00	12/9/2014	110,50	ACK,ACK
101	11:00	12/9/2014	130,110	ACK,F-2
101	12:00	12/9/2014	187,50	ACK,ACK
101	13:00	12/9/2014	141,55	F-3,ACK
101	14:00	12/9/2014	162,40	ACK, ACK
101	15:00	12/9/2014	120,35	ACK, ACK
101	16:00	12/9/2014	125,55	ACK, ACK

Tables 4.6 is an extract of data received on the 12/09/2014, details can be seen in appendix A. it depicts feedback data received by the supervisory controller in the DAS via the modem from the central remote terminal unit after its initial FDI analysis based on data variance and correlation. The table clearly showed that by 07:00, 09:00, 11:00, 13:00, the fault conditions F-1, F-2 and F-3, were all observed. It will also be necessary to observe that in the immediate next measurement cycle after a fault condition occurred (at 08:00, 10:00, 12:00, 14:00) there was no observed error

indicating a clear attempt by the supervisory controller to control the system and handle the occurring fault situation. The system performed reliably well during these measurement cycles.

Data Analytics Unit

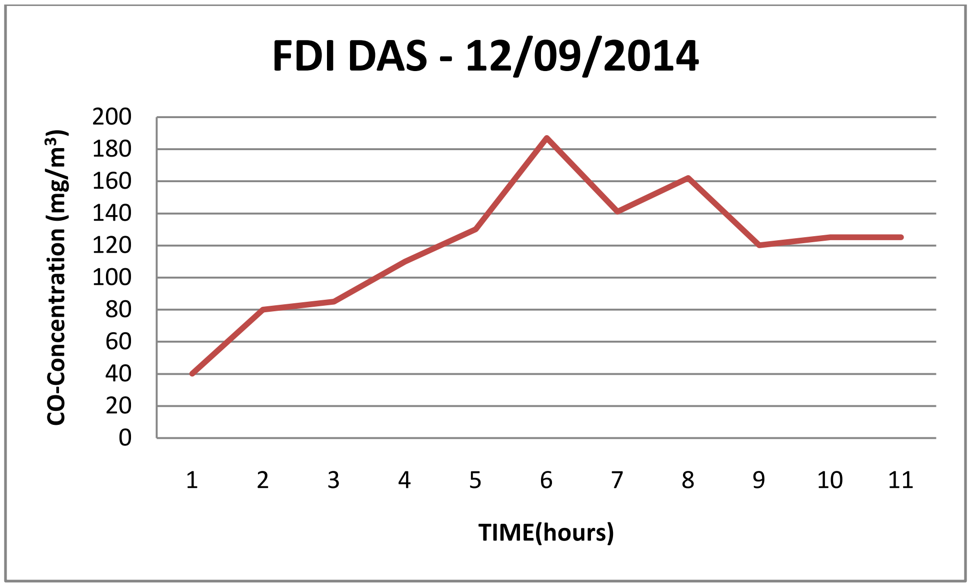


Figure 4.1: CO Concentration Chart for 12/09/2014.

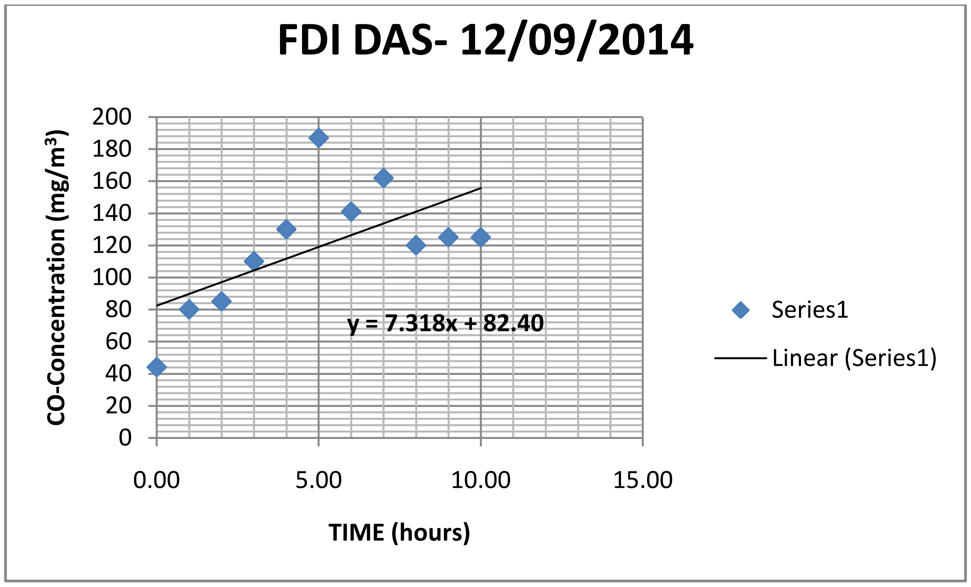


Figure 4.2: CO Concentration Chart Showing Trend Line for 12/09/2014.

Figure 4.1 and figure 4.2 show the linear relationship between the dependent and independent variables of concentration and time respectively. A view of the graph in figure 4.2 makes it easy to see the outlier values observed. Points D_{CO} = 187 and D_{CO} = 40 are so far away from the trend line which represents the weighted arithmetic mean of all data values measured to date. With respect to the FDI model implemented in the central remote terminal unit as described by Figure 3.21, the FDI-DAS system was able to identify the points as outliers after comparison with thresholds as well as the correlation model equation described in equation (20). The effects of the subsequent control action on the DAS by the supervisory controller (on feedback received from the central remote terminal unit as described in Figs 3.16, 3.17 and 3.18 respectively) can be seen on the chart as the readings returned to normal range (closer to the trend line in fig 4.2 and fig 4.4) in its next measurement cycle. For example, in Fig 4.2, the first measurement of 40 is observed to be farther from the trend line than the next measurement of 80, the measurement at 187 and 160 are very far away from the trend line shown on the graph but their immediate next measurements of 140 and 120 respectively are closer to the trend line. The closer a value is to the trend line; the less likely it's in error.

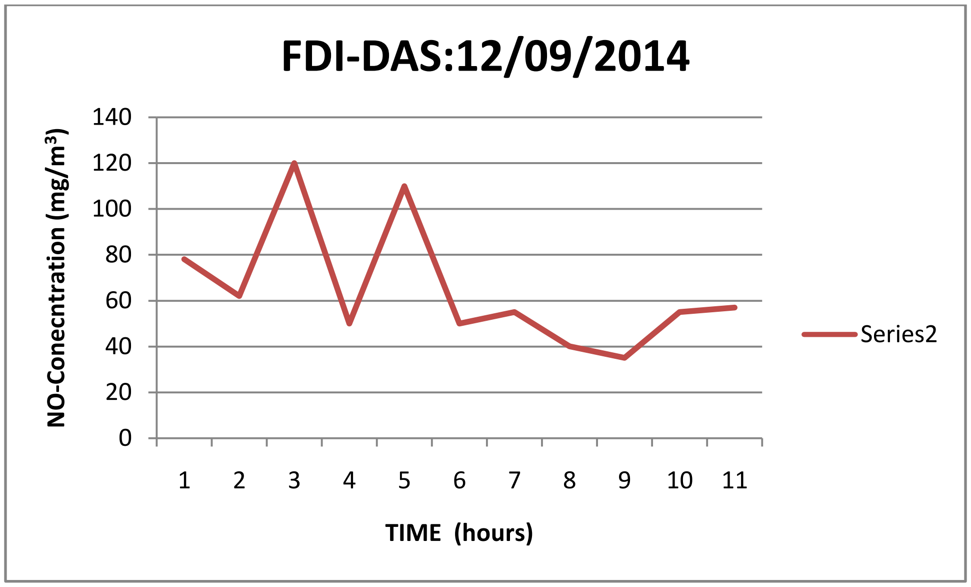


Figure 4.3: NO Concentration Chart for 12/09/2014.

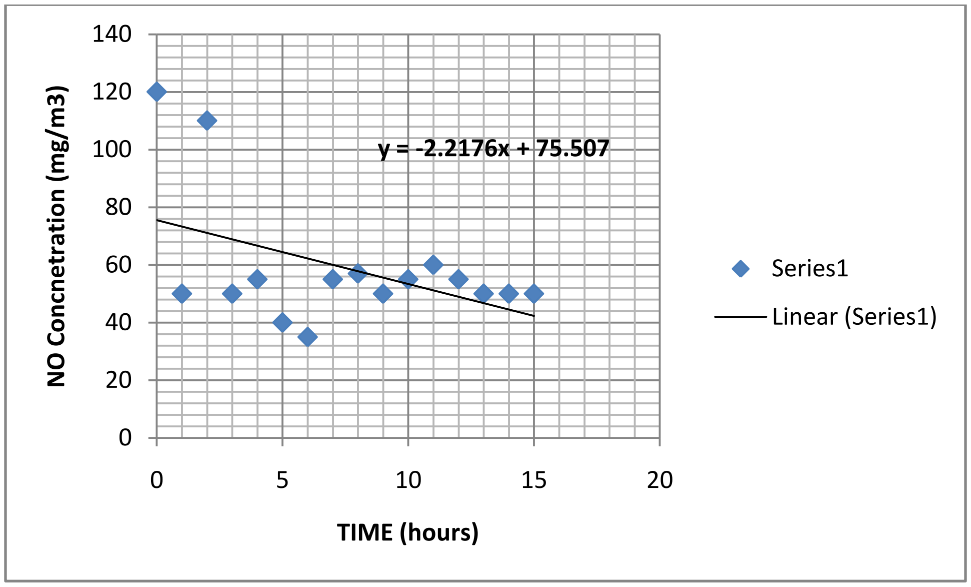


Figure 4.4: NO Concentration Chart Showing Trend Line for 12/09/2014.

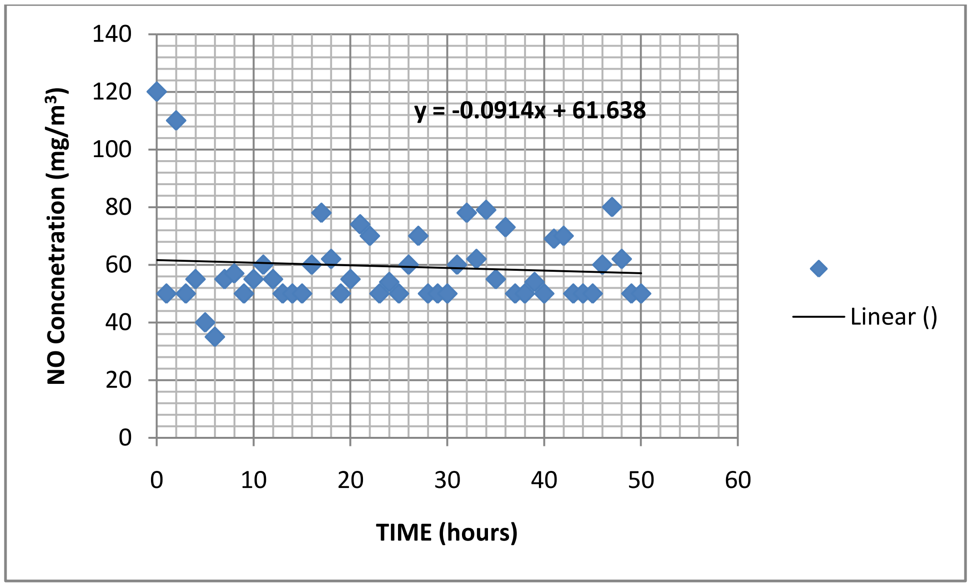


Figure 4.5: NO Concentration Chart Showing Trend Line for 50hours of Measurement.

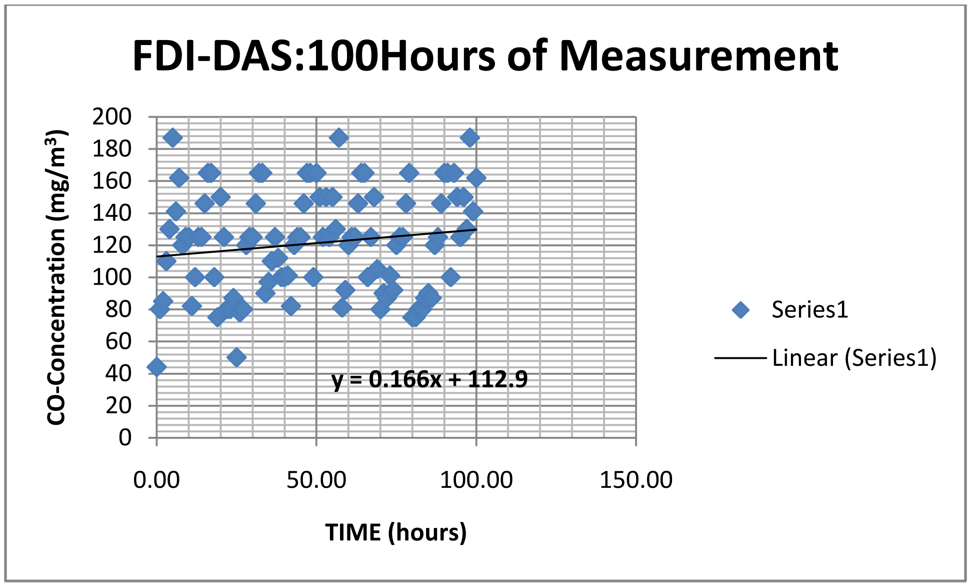


Figure 4.6: CO Concentration Chart Showing Trend Line for 100hours of Measurement.

Fig 4.5 and fig 4.6 show the scatter plot with trend line for 50 hours of NO measurement and 100hours of CO measurement sampled from the pollution database shown in appendix A. It can be observed on both charts the accumulation of most data points along the trend lines (weighted mean value). This nearness indicates the correlation between each individual data points and their adherence to the calibrated measurement range for the MICS 4514 used for this study. The absence of two concurrently occurring outlier measurements (where outliers are data measurements that are so far away from the trend line that depicts the weighted average mean of past pollutant measurements) in figs 4.2, 4.4, 4.5 and 4.6 proved that they were detected by the central remote terminal unit, FDI correlation model and corrected by the supervisory controller in the next measurement cycle. The analyses of outliers are clear indications of faults (F-1 and F-2). As these either lies too far above the upper threshold value (450mg/m^3 for CO and 107mg/m^3 for NO) or lie lower than the lower threshold value(75mg/m^3 for CO and 40mg/m^3 for NO). F-3 faults thus would occur amongst the data values which appear to be correlated with past readings as seen in the charts. Thus the supervisory controller's control action described in fig 3.16, 3.17

and 3.18 attempts to ensure all readings are tested in the fault model. Even when they appear correlative, they must satisfy the model equation described in equation 3.63.

The FDI-DAS performed satisfactorily on varying pollution data sets and utilized its fault detection and isolation features as described in Figures 3.10, 3.14.3.16, 3.17, 3.18 and 3.21.

CHAPTER FIVE

CONCLUSION, SUGGESTIONS AND RECOMMENDATIONS

5.1 Conclusion

The dissertation “An FDI data acquisition system for CO and NO monitoring” consisted of three distinct but interconnected parts, namely the Data acquisition system unit (DAS), the central remote terminal unit (CTU) and the Data analytics unit. Carbon monoxide (CO) and nitrogen oxide (NO) were both detected and measured using a redundant connection of two MICS-4514 sensors to measure pollutants from an industrial location in Nigeria (IFE steel mill). The DAS was able to perform on-site analysis of measured data when compared to set point ,where normal ranges for these pollutants , are the values of CO <= 100mg/m^3 and for NO values <= 55mg/m^3 (Wei Ying Yi Kin, Terrence, Kwong Sak Leung, & Yee, 2015) and sounded alarms as needed. The DAS transmits this data to a central remote terminal unit which implements a fault detection model based on standard deviation and data variance to determine if the measurements received are actually valid data. These faults were classified into various fault conditions namely F-1 (which represents when reading fall below the lowest measurable range of the sensor i.e. 75mg/m^3 for CO and 40mg/m^3 for NO), F-2 (which represents when the value is higher than the highest measureable range of the sensor i.e. 450mg/m^3 for CO and 107mg/m^3 for NO) and also the F-3 condition (which represents when the values lie between the measurable range but they are still in error due its non correlativeness with previous measurements). The fault status was sent as a feedback to the supervisory controller in the DAS; where an error residual generator based on matrix functions was used to determine possible fault isolation and corrective control actions the supervisory controller can perform on the DAS to enable it operate optimally. The data measured by the system was analyzed using an air quality monitoring software where graphical results in form of straight line graphs, scatter plots were used to determine the adherence of the system to working specifications.

5.2 Contribution to Knowledge

- The dissertation utilized a simple data correlation model based on standard deviation and data variance (as shown in equation 3.59, 3.60 and 3.61) to perform fault detection on measured pollution data in the central terminal unit. It proved that measurements in such applications could be regarded a fixed process application as the measurement data showed a correlative pattern over time, allowing for easy analysis for variance degradation sensor faults.

- The study showed the benefits of adding a feedback to a multitier data acquisition system deployment, where data received by the data acquisition module from the central remote terminal unit, enabled it perform corrective actions to improve measurement accuracy to identified faults.
- Passive fault tolerant systems designs do not implement FDI schemes and models, yet this dissertation developed a passive FTCS system that includes some level of fault detection and isolation (FDI). This not only improves the developed system's robustness to handling pre-known faults (passively), it also reduces it down time and troubleshooting time in event of a system failure.
- The use of a matrix- based approach in the error generator described by equation 3.30, (as opposed to the probability approach) is well thought out because this study considered two major sensor faults namely the off -set bias fault and the drop-out sensor fault. There exists other types of faults (albeit minor) that could occur in the sensor, controller or other components. The matrix approach can accommodate this increase as one only has to increase the order of the matrix used to represent the input and output transfer matrix described by equation 3.9.

5.3 Suggestions for Further Improvement

From the overall experience gathered during the course of this research, it was discovered that an FPGA approach would allow the use of analytical redundancy instead of physical redundancy (two sensors).

From the foregoing, further research may include applying other methodologies such as Active FTCS or hybrid FTCS which could eliminate the need for a redundant sensor, but rather attempt to solve the entire control problem analytically via predictive modeling.

References

Agajo, J., & Inyiama, H. (2011). Remote Monitoring and Estimation of Carbon Monoxide Pollution in indoor Enviornment using Wireless Sensor Network via Satelite. *Pacific Journal of Science and Technology* , 464-471.

Ahmad, A. (2011). Design of a Fault Tolerant Controller Based on Observers for a PMSM Drive. *IEEE Transactions on Industrial Electronics*, (pp. 1416-1427).

Aloul, F., Al-ali, A., & Zualkernan, I. (2010). A Mobile GPRS Sensor Array for Air Polltuion Monitoring. *IEEE Sensors Journal* .

Alwi, H., Edwards, C., & Pin Tan, C. (2011). Fault Detection and Fault Tolerant Control Using Sliding Mode. *Springer* .

Arici, T., & Altunbasak, Y. (2003). *Adaptive Sensing for Environment Monitoring.* Georgia: Centre for Signal and Image Processing.

Balzano, L. (2006). *Faults in Sensor Networks.* USA: UCLA.

Batra, D. (2005). Conceptual data modelling patterns. *journal od database management* , 84-106.

Chihaia, C. I. (2010). *Active Fault-Tolerance in Wireless Networked.* Lasi,Romania.

circuitstoday, B. (2017). *Power Supply Circuit.* Circuit today.

Claudio, P., Luca, C., & Alberto, L. (2008). Fault Tolerant Distributed Deplyment of Embedded Control Software. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems* .

Cole, M., & Keogh, P. B. (2012). Fault Tolerant Control of Rotor/Magnetic Bearing systems using Reconfigurable Control with inbuilt fault detection. *Journal of Mechanical Engineering Sciences* , 1445-1466.

Darwish, I., & Elqafas, S. (2016). Enhanced Algorithms for Fault Nodes Recovery in Wireless Sensors Network. *International Journal Sensor Network Data Communication* .

Daudi S, S. (2017). Industrial Air Pollution Monitoring System Based on Wireless Sensor Networks. *Journal of Information Sciences and Computing Technologies(JISCT)* , 612-624.

David, C. (1996). Health Effects of Outdoor Airr Pollution:Committee on Environment and Occupational Health Assembly of the American Thoracic Society. *American Journal of Respiratory and Critical Care Medicine* .

Ding, Y., Hu, Y., Hao, K., & Cheng, L. (2015). An intelligent routing recovery scheme for heterogeneous wireless sensor networks. *MPSICA* , 49-60.

Duflo, E., Greenstone, M., & Hanna, R. (2008). Indian Air pollution, Health and Economic Well Being. *sapiens (www.Sapiens.revens.org)* .

e2vtechnologies. (2008). *MICS 4514.* united kingdom: e2v technologies limited.

Edwards, C., & Pin Tan, C. (2006). Sensor Fault tolerant Control Using Sliding Mode Observer. *Control Engineering Practise* , 897-908.

ElectronicsHub. (2017). *Data Acqusition System.* Electronics Hub.

Farrah, J., Mateen, G., & Brook, R. (2011). Air Pollution as an Emerging Global Risk Factor for Stroke. *journal of air monitoring and assesment* , 1240-1243.

fekih, .. (2006). A Robust Fault Tolerant Control Strategy for a class of nonlinear uncertain systems. *proceedings of the American Control Conference*, (pp. 5474-5480). Mineapolis, Minnesota,USA.

fenterstock, J., kurtzweg, J., & Ozolins, G. (1971). Reduction of Air Pollution Potential through Environmental Planning. *Journalof the air pollution control association* , 395-399.

Gajski, D. (2009). Embedded System Design: Modeling, Synthesis and Verification,. *Springer Science + Business Media, LLC* , 35-47.

Georgiev Z, S. M. (1994). VLSI Common Voting Module for Fault-Tolerant TMR System in Industrial System Control Applications. *International Journal of Electronics* , 163-205.

Goldstein, A., Koven, C., & Heald, C. (2009). Biogenic Carbon and Anthropogenic Pollutants Combine to Form a cooling haze over southeastern United States. *National Academy of Sciences* .

Harish, R., Prahbur, B., Gadh, R., & Asad, M. (2007). Wireless Industrial Monitoring and Control Using a Smart Sensor Platform. *IEEE Sensors Journal* , 34-38.

Hassan, N., Dominque, S., Frederic, H., & Dieder, T. (2000). Fault Tolerant Control in Dynamic Systems:Application to a Winding Machine. *IEEE Systems Control Magazine* , pp. 1053-5888.

Heiner, J., Collias, N., & Wirthin, M. (2007). Fault Tolerant ICAP Controller for High Reliable Internal Security. *IEEE AC* .

Henderson, T., Shilcrat, E., & Hansen, C. (1983). A Fault Tolerant Sensor Scheme. *System Development Foundation and NSF Grants* (pp. 1-16). Utah, Salt Lake City: Department of Computer Science, University of Utah.

Iancu, E. (2011). Fault Detection and Analysis Using Spectral Analysis. *IFAC Conference.* Prague.

Jiang, J. (2014). *Fault Tolerant Control Systems:An Introductory Overview.* London: Department of electrical and computer engineering, University of Western Ontario.

Kameswara, R., & Chakravarthy, A. (2017). A Fault Tolerant Framework to Detect Routing Failures in Air Pollution Monitoring MANET Using 2ACK. *International Journal of Scientific & Engineering Research Volume 8, Issue 9, September-2017* , 300-303.

Khedo, K., Perseedoss, R., & Mungur, A. (2010). A Wireless Sensor Network Air Pollution Monitoring System. *International Journal of Wireless and Mobile Networks* , 35-45.

Krstic, M., Stojcev, M., Djordjevic, G., & Andrejic, I. (2015). *A Mid-Value Select Voter.* , Beogradska 14, 18000 Nis, Serbia & Montenegro: Faculty of Electronic Engineering, University of Nis.

Mahapatro, A., & Khilar, P. (2013). Energy-efficient distributed approach for clustering-based fault detection and diagnosis in image sensor networks. *IET Wireless Sensor Systems* , 26-36.

Mahmood, M., Jiang, J., & Zhang, Y. (2003). Active Fault Tolerant Control Systems:Stochastic Analysis and Synthesis. *SPringer-verlag* .

Manshahia, M. S. (2016). Wireless Sensor Networks:A survey. *International Journal of Scientific & Engineering Research , Vol. 7 , Issue 4* , 710-716.

Markovic, N., Stanimovic, A., & Stoimenor, L. (2009). Sensor Web for River Pollution Monitoring and ALert System. *AGILE International conference on Geographic Information Science,* (pp. 1-9).

Mishra, S., Dhanashare, T., & Asutkar, G. (2011). Design of Energy Aware Air Pollution Monitoring System Using WSN. *International Journal of Advances in Engineering and Technology* , 34-39.

Mogens, B., Marcel, S., & Eva, N. (2001). Concepts and Methods in Fault Tolerant Control. *Tutorial at American Control Conference, Paper 012.* USA.

Mohammed, O., Mohammed, C., & Ahmed. (2004). robust observer based fault tolerant control for vehicle lateral dynamics. *International Journal for Vehicle Design* , 12.

Mylopoulous, J., Lobcopolous, P., & Zicari, R. (2010). Modelling,Databases and a case of integrated view of information systems development. *citeseerx* , 49-68.

Neville, I., Kam, S., & Myron, H. (2006). *An Integrated Fault Tolerant Robotic Control System for High Reliability and safety.* pasadena: Jet Proplusion Laboratory.

Ocampo-Martinez, C., & Vicenc, P. (2011). *Fault Tolerant Model Predictive control within hybrid systems framework:application to server networks.* Barcelona,Spain: Institute de Robotica Informatics Industrial (IRI).

Partheeban, P., Hemamalini, R., & Ragu, P. (2012). Vehicular Emmision Monitoring Using Internet,GIS, GPS and Sensors. *International Conference on Environment Energy and Biotechnology.*

Paton, R. (1997). Fault Tolerant Control Systems:The 1997 Situation. *IFAC safe process*, (pp. 1033-1055). hull,united kingdom.

Pimentel, R., & Salazar, M. (2011). *Dependability of Distributed Control System Fault Tolerant Units.* Flint,Michigan: Department of Electrical and Computer Engineering,kettening University.

Pravin, J., Deepak, S., & Angeline, V. (2013). Industrial Air Pollution Monitoring System Using Labview and GSM. *International Journal of Advanced Research in ELectrical Electronics and Instrumentation Engineering* , 2685-2693.

Ramon, B. (2007). *Challenging Malicious Inputs with Fault Tolerance.* Black Hats Europe.

Roman, M. (2010). *Governing from the Middle:The CO Cities Leadership Group.* Coporate Governance.

Romano, P., Rughetti, D., Quaglia, F., & Ciciani, B. (2005). APART: Low Cost Active Replication for Multi-tier Data Acquisition Systems. *IEEE Transactions on Knowledge and Data Engineering,* , 1-8.

Schreiner, C., Branzila, M., Trandabat, A., & Gobanu, R. (2006). Air Quality and Pollution Mapping System Using Remote Measurements and GPS Technology. *Global NEST Journal* , 315-323.

Sghair, M., Bonneral, A., Crowzet, Y., & Brot, P. (2007). Challenges in building fault tolerant flight control systems for a civil aircraft. *International Journal of Computer Sciences* , 35.

Singla, T., Mukhdeep, S., & Manshahia, Z. (2017). Wireless Sensor Networks for Pollution Monitoring and Control. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* , 763-770.

Spergler, J., & Sexton, D. (1983). Indoor Air Pollution: A Public Health Perspective. *Science(New Series)* , 9-17.

Starowiecki, M. (2011). Fault Tolerant Systems. *Control Systems, Robotics and Automation* , 41-88.

Texasinstruments. (1995). *Microcontroller Based Data Acquisition Using the TLC2543 12-Bit Serial-Out ADC.* Texas.

Tushar, J., Joseph, Y., & Dominque, S. (2011). *A Novel supervisory based fault tolerant control:application to hydraulic processes.* hal:00576324: springer-verlag.

Virjnatha, R., Aranvid, R., & Kumar, B. (2013). Pollution Monitoring System Using Wireless SEnsore Networks in Visakhapatnam. *International Journal of Engineering Trends and Technology* , 45-51.

Volgyesi, P., Nadas, A., Koutsoukos, X., & Ledeczi, A. (2008). Air Quality Monitoring with Sensor Map. *International Conference on Information Processing in Sensor Networks,* (p. 529&530).

Wang, Y., & Hongyi, W. (2010). *DFT-MSN: The Delay Fault Tolerant Mobile Sensor Network for Pervasive Information Gathering.* P.O. Box 44330, Lafayette, LA: Center for Advanced Computer Studies,University of Louisiana at Lafayette,.

Wei Ying Yi Kin, M. L., Terrence, M., Kwong Sak Leung, 1., & Yee, L. (2015). A Survey ofWireless Sensor Network Based Air Pollution Monitoring Systems. *Sensors* , 31392–31427.

Werner, E., & Peter, B. (2009). A fault-tolerant eddy covariance system. *Institute of Plant, Animal and Agroecosystem Sciences,* (pp. 1-26). ETH Zurich, Universit¨atsstrasse 2, CH–8092 Zurich,: Agricultural and Forest Meteorology Special issue on CH4 and N2O fluxes.

Yixin, D., & Passino, K. (2001). Stable Fault Tolerant Adaptive Fuzzy Neural Control for a Turbine Engine. *IEEE Transactions on Control Systems Technology* .

Young, J., Yang, K., Dongs, G., Keun, H., & Nittel, S. (2005). Air Pollution Monitoring System Based on Geo Sensor Networks. *Geo Sensors Networks Journal* , 269.

Zhao, G. (2011). *Wireless Sensors Network for Industrial Process Monitoring and Control:A survey.* Macrothink Institute Network Protocols and Algorithms.

Žliobaite, I., Hollmén, J., & and Junninen, H. (2014). Regression Models Tolerant to Massively Missing Data: A Case Study of Solar Radiation Now-Casting. *European Geosciences Union* (pp. 4387-4399). Helsinki,Finland: Copernicus Publications.

Zoidis, J. (1996). The Impact of Air Pollution on COPD for Decision makers in Respiratory Care. *Springer-Verlag* .

APPENDIX A: Pollutant Concentration Data (IFE STEEL)

Date	TIME	CO_x	NO_x	Fault CO_x	Fault NO_x	Daily Average CO_x	Daily Average NO_x	Longitude	Latitude
02/09/2014	07:00	80	40	NIL	NIL			4.5566	7.4707
	08:00	84	40	NIL	NIL			4.5566	7.4707
	09:00	80	50	NIL	NIL			4.5566	7.4707
	10:00	92	60	NIL	NIL			4.5566	7.4707
	11:00	102	80	NIL	NIL			4.5566	7.4707
	12:00	140	75	NIL	NIL			4.5566	7.4707
	13:00	125	75	NIL	NIL			4.5566	7.4707
	14:00	150	50	NIL	NIL			4.5566	7.4707
	15:00	125	43	NIL	NIL			4.5566	7.4707
	16:00	102	50	NIL	NIL			4.5566	7.4707

	17:00	101	45	NIL	NIL	128	55.27	4.5566	7.4707
03/09/2014	07:00	80	54	NIL	NIL			4.5566	7.4707
	08:00	80	50	NIL	NIL			4.5566	7.4707
	09:00	82	57	NIL	NIL			4.5566	7.4707
	10:00	120	60	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**longitude**	**Latitude**
	11:00	125	50	NIL	NIL			4.5566	7.4707
	12:00	125	50	NIL	NIL			4.5566	7.4707
	13:00	146	50	NIL	NIL			4.5566	7.4707
	14:00	165	60	NIL	NIL			4.5566	7.4707
	15:00	165	80	NIL	NIL			4.5566	7.4707
	16:00	100	62	NIL	NIL			4.5566	7.4707
	17:00	75	50	NIL	NIL			4.5566	7.4707
	-			NIL	NIL			4.5566	7.4707
	-			NIL	NIL			4.5566	7.4707
				NIL	NIL			4.5566	7.4707

				NIL	NIL	99.81	53.14	4.5566	7.4707
04/09/2014	07:00	87	54	NIL	NIL			4.5566	7.4707
	8:00	81	50	NIL	NIL			4.5566	7.4707
	9:00	82	62	NIL	NIL			4.5566	7.4707
	10:00	120	80	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	11:00	125	50	NIL	NIL			4.5566	7.4707
	12:00	125	50	NIL	NIL			4.5566	7.4707
	13:00	184	50	NIL	NIL			4.5566	7.4707
	14:00	165	60	NIL	NIL			4.5566	7.4707
	15:00	165	78	NIL	NIL			4.5566	7.4707
	16:00	100	62	NIL	NIL			4.5566	7.4707
	17:00	75	50	NIL	NIL	104	55.24	4.5566	7.4707
05/09/2014	7:00	75	55	NIL	NIL			4.5566	7.4707
	8:00	85	74	NIL	NIL			4.5566	7.4707

	9:00	80	80	NIL	NIL			4.5566	7.4707
	10:00	80	85	NIL	NIL			4.5566	7.4707
	11:00	87	54	NIL	NIL			4.5566	7.4707
	12:00	91	50	NIL	NIL			4.5566	7.4707
	13:00	92	70	NIL	NIL			4.5566	7.4707
	14:00	120	80	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	15:00	125	50	NIL	NIL			4.5566	7.4707
	16:00	125	50	NIL	NIL			4.5566	7.4707
	17:00	146	50	NIL	NIL	118.56	50.02	4.5566	7.4707
06/09/2014	7:00	75	36	NIL	F2			4.5566	7.4707
	8:00	75	40	NIL	NIL			4.5566	7.4707
	9:00	100	62	NIL	NIL			4.5566	7.4707
	10:00	105	50	NIL	NIL			4.5566	7.4707
	11:00	150	55	NIL	NIL			4.5566	7.4707
	12:00	125	74	NIL	NIL			4.5566	7.4707
	13:00	150	80	NIL	NIL			4.5566	7.4707
	14:00	130	87	NIL	NIL			4.5566	7.4707

	15:00	107	54	NIL	NIL			4.5566	7.4707
	16:00	101	60	NIL	NIL			4.5566	7.4707
	17:00	92	70	NIL	NIL	100.78	55.9	4.5566	7.4707
07/09/2014	7:00	40	40	F2	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	15:00	125	50	NIL	NIL			4.5566	7.4707
	16:00	125	50	NIL	NIL			4.5566	7.4707
	17:00	146	50	NIL	NIL			4.5566	7.4707
08/09/2014	7:00	78	60	NIL	NIL			4.5566	7.4707
	8:00	80	78	NIL	NIL			4.5566	7.4707
	9;00	100	62	NIL	NIL			4.5566	7.4707
	10:00	105	50	NIL	NIL			4.5566	7.4707
	11:00	110	50	NIL	NIL			4.5566	7.4707
	12:00	125	43	NIL	NIL			4.5566	7.4707
	13:00	150	50	NIL	NIL			4.5566	7.4707
	14:00	130	50	NIL	NIL			4.5566	7.4707
	15:00	187	54	NIL	NIL			4.5566	7.4707

	16:00	141	50	NIL	NIL			4.5566	7.4707
	17:00	162	62	NIL	NIL	120.3	50.33	4.5566	7.4707
09/09/2014	7:00	120	80	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	8:00	125	50	NIL	NIL			4.5566	7.4707
	9:00	125	50	NIL	NIL			4.5566	7.4707
	10:00	146	50	NIL	NIL			4.5566	7.4707
	11:00	105	60	NIL	NIL			4.5566	7.4707
	12:00	105	78	NIL	NIL			4.5566	7.4707
	13:00	104	62	NIL	NIL			4.5566	7.4707
	14:00	85	50	NIL	NIL			4.5566	7.4707
	15:00	150	55	NIL	NIL			4.5566	7.4707
	16:00	125	74	NIL	NIL			4.5566	7.4707
	17:00	75	50	NIL	NIL	109.2	53.44	4.5566	7.4707
11/09/2014	7:00	80	50	NIL	NIL			4.5566	7.4707

	8:00	87	54	NIL	NIL			4.5566	7.4707
	9:00	94	53	NIL	NIL			4.5566	7.4707
	10:00	102	62	NIL	NIL			4.5566	7.4707
	11:00	90	70	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	12:00	99	50	NIL	NIL			4.5566	7.4707
	1:00	105	50	NIL	NIL			4.5566	7.4707
	14:00	120	50	NIL	NIL			4.5566	7.4707
	15:00	122	60	NIL	NIL			4.5566	7.4707
	16:00	110	78	NIL	NIL			4.5566	7.4707
	17:00	100	62	NIL	NIL	100.88	51.12	4.5566	7.4707
12/09/2014	7:00	44	120	F2	F3			4.5566	7.4707
	8:00	80	50	NIL	NIL			4.5566	7.4707
	9:00	85	110	NIL	F3			4.5566	7.4707
	10:00	110	50	NIL	NIL			4.5566	7.4707
	11:00	130	55	NIL	NIL			4.5566	7.4707
	12noon	187	40	NIL	NIL			4.5566	7.4707

	13:00	141	35	NIL	F1			4.5566	7.4707
	14:00	162	55	NIL	NIL			4.5566	7.4707
	15:00	120	57	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	16:00	125	50	NIL	NIL			4.5566	7.4707
	17:00	125	55	NIL	NIL	120	51	4.5566	7.4707
13/09/2014	7:00	…	...	F1	F1			4.5566	7.4707
	8:00	…	...	F1	F1			4.5566	7.4707
	9:00	…	....	F1	F1			4.5566	7.4707
	10:00	…	...	F1	F1			4.5566	7.4707
	11:00	…	...	F1	F1			4.5566	7.4707
	12noon	…	...	F1	F1			4.5566	7.4707
	13:00	…	...	F1	F1			4.5566	7.4707
	14:00	…	...	F1	F1			4.5566	7.4707
	15:00	…	...	F1	F1			4.5566	7.4707
	16:00	….	...	F1	F1			4.5566	7.4707
	17:00	…	...	F1	F1			4.5566	7.4707

14/09/2014	7:00	82	60	NIL	NIL			4.5566	7.4707
	8:00	100	55	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	9:00	125	50	NIL	NIL			4.5566	7.4707
	10:00	125	50	NIL	NIL			4.5566	7.4707
	11:00	146	50	NIL	NIL			4.5566	7.4707
	12noon	165	60	NIL	NIL			4.5566	7.4707
	13:00	165	78	NIL	NIL			4.5566	7.4707
	14:00	100	62	NIL	NIL			4.5566	7.4707
	15:00	75	50	NIL	NIL			4.5566	7.4707
	16:00	150	55	NIL	NIL			4.5566	7.4707
	17:00	125	74	NIL	NIL	121	54.1	4.5566	7.4707
15/09/2014	7:00	80	70	NIL	NIL			4.5566	7.4707
	8:00	80	50	NIL	NIL			4.5566	7.4707
	9:00	87	54	NIL	NIL			4.5566	7.4707
	10:00	78	50	NIL	NIL			4.5566	7.4707
	11:00	80	60	NIL	NIL			4.5566	7.4707

	12noon	120	70	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	13:00	125	50	NIL	NIL			4.5566	7.4707
	14:00	125	50	NIL	NIL			4.5566	7.4707
	15:00	146	50	NIL	NIL			4.5566	7.4707
	16:00	165	60	NIL	NIL			4.5566	7.4707
	17:00	165	78	NIL	NIL	130	60	4.5566	7.4707
16/09/2014	7:00	90	62	NIL	NIL			4.5566	7.4707
	8:00	97	79	NIL	NIL			4.5566	7.4707
	9:00	110	55	NIL	NIL			4.5566	7.4707
	10:00	125	73	NIL	NIL			4.5566	7.4707
	11:00	112	50	NIL	NIL			4.5566	7.4707
	12noon	100	50	NIL	NIL			4.5566	7.4707
	13:00	100	54	NIL	NIL			4.5566	7.4707
	14:00	101	50	NIL	NIL			4.5566	7.4707
	15:00	82	69	NIL	NIL			4.5566	7.4707
	16:00	120	70	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	17:00	125	50	NIL	NIL			4.5566	7.4707
18/09/2014	7:00	125	50	NIL	NIL			4.5566	7.4707
	8:00	146	50	NIL	NIL			4.5566	7.4707
	9:00	165	60	NIL	NIL			4.5566	7.4707
	10:00	165	80	NIL	NIL			4.5566	7.4707
	11:00	100	62	NIL	NIL			4.5566	7.4707
	12noon	165	50	NIL	NIL			4.5566	7.4707
	13:00	150	50	NIL	NIL			4.5566	7.4707
	14:00	125	74	NIL	NIL			4.5566	7.4707
	15:00	150	50	NIL	NIL			4.5566	7.4707
	16:00	130	50	NIL	NIL			4.5566	7.4707
	17:00	187	54	NIL	NIL	130	62	4.5566	7.4707
19/09/2014	7:00	81	50	NIL	NIL			4.5566	7.4707

	8:00	92	70	NIL	NIL			4.5566	7.4707
	9:00	120	60	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	10:00	125	50	NIL	NIL			4.5566	7.4707
	11:00	125	50	NIL	NIL			4.5566	7.4707
	12noon	146	50	NIL	NIL			4.5566	7.4707
	13:00	165	60	NIL	NIL			4.5566	7.4707
	14:00	165	80	NIL	NIL			4.5566	7.4707
	15:00	100	62	NIL	NIL			4.5566	7.4707
	16:00	125	90	NIL	NIL			4.5566	7.4707
	17:00	150	50	NIL	NIL	101.2	64	4.5566	7.4707
20/09/2014	7:00	105	43	NIL	NIL			4.5566	7.4707
	8:00	80	50	NIL	NIL			4.5566	7.4707
	9:00	90	50	NIL	NIL			4.5566	7.4707
	10:00	87	54	NIL	NIL			4.5566	7.4707
	11:00	101	50	NIL	NIL			4.5566	7.4707
	12noon	92	70	NIL	NIL			4.5566	7.4707

	13:00	120	70	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	14:00	125	50	NIL	NIL			4.5566	7.4707
	15:00	125	50	NIL	NIL			4.5566	7.4707
	16:00	146	50	NIL	NIL			4.5566	7.4707
	17:00	165	60	NIL	NIL	122	55	4.5566	7.4707
21/09/2014	7:00	……	……	F1	F1			4.5566	7.4707
	8:00	……	……	F1	F1			4.5566	7.4707
	9:00	……	……	F1	F1			4.5566	7.4707
	10:00	……	……	F1	F1			4.5566	7.4707
	11:00	……	……	F1	F1			4.5566	7.4707
	12noon	……	……	F1	F1			4.5566	7.4707
	13:00	……	……	F1	F1			4.5566	7.4707
	14:00	……	……	F1	F1			4.5566	7.4707
	15:00	……	……	F1	F1			4.5566	7.4707

	16:00	……	……	F1	F1			4.5566	7.4707
	17:00	……	……	F1	F1			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
22/09/2014	7:00	…..	…..	F1	F1			4.5566	7.4707
	8:00	…..	…..	F1	F1			4.5566	7.4707
	9:00	…..	…..	F1	F1			4.5566	7.4707
	10:00	…..	…..	F1	F1			4.5566	7.4707
	11:00	…..	…..	F1	F1			4.5566	7.4707
	12noon	…..	…..	F1	F1			4.5566	7.4707
	13:00	…..	…..	F1	F1			4.5566	7.4707
	14:00	…..	…..	F1	F1			4.5566	7.4707
	15:00	…..	…..	F1	F1			4.5566	7.4707
	16:00	…..	…..	F1	F1			4.5566	7.4707
	17:00	…..	…..	F1	F1			4.5566	7.4707
23/09/2014	7:00	…..	…..	F1	F1			4.5566	7.4707

	8:00	…..	…..	F1	F1			4.5566	7.4707
	9:00	…..	…..	F1	F1			4.5566	7.4707
	10:00	…..	…..	F1	F1			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	11:00	……	……	F1	F1			4.5566	7.4707
	12noon	……	……	F1	F1			4.5566	7.4707
	13:00	……	……	F1	F1			4.5566	7.4707
	14:00	……	……	F1	F1			4.5566	7.4707
	15:00	……	……	F1	F1			4.5566	7.4707
	16:00	……	……	F1	F1			4.5566	7.4707
	17:00	……	……	F1	F1			4.5566	7.4707
25/09/2014	7:00	75	55	NIL	NIL			4.5566	7.4707
	8:00	75	74	NIL	NIL			4.5566	7.4707
	9:00	80	50	NIL	NIL			4.5566	7.4707
	10:00	80	50	NIL	NIL			4.5566	7.4707

	11:00	87	54	NIL	NIL			4.5566	7.4707
	12noon	90	50	NIL	NIL			4.5566	7.4707
	13:00	87	60	NIL	NIL			4.5566	7.4707
	14:00	120	70	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	15:00	125	50	NIL	NIL			4.5566	7.4707
	16:00	125	50	NIL	NIL			4.5566	7.4707
	17:00	146	50	NIL	NIL			4.5566	7.4707
26/09/2014	7:00	165	60	NIL	NIL			4.5566	7.4707
	8:00	165	78	NIL	NIL			4.5566	7.4707
	9:00	100	62	NIL	NIL			4.5566	7.4707
	10:00	165	50	NIL	NIL			4.5566	7.4707
	11:00	150	55	NIL	NIL			4.5566	7.4707
	12noon	125	74	NIL	NIL			4.5566	7.4707
	13:00	150	50	NIL	NIL			4.5566	7.4707

	14:00	130	50	NIL	NIL			4.5566	7.4707
	15:00	187	54	NIL	NIL			4.5566	7.4707
	16:00	141	50	NIL	NIL			4.5566	7.4707
	17:00	162	75	NIL	NIL	134	63	4.5566	7.4707
27/09/2014	7:00	120	51	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	8:00	105	60	NIL	NIL			4.5566	7.4707
	9:00	115	50	NIL	NIL			4.5566	7.4707
	10:00	106	50	NIL	NIL			4.5566	7.4707
	11:00	125	60	NIL	NIL			4.5566	7.4707
	12noon	115	78	NIL	NIL			4.5566	7.4707
	13:00	120	70	NIL	NIL			4.5566	7.4707
	14:00	122	50	NIL	NIL			4.5566	7.4707
	15:00	123	50	NIL	NIL			4.5566	7.4707
	16:00	155	74	NIL	NIL			4.5566	7.4707

	17:00	150	50	NIL	NIL	149	65	4.5566	7.4707
28/09/2014	7:00	130	50	NIL	NIL			4.5566	7.4707
	8:00	107	54	NIL	NIL			4.5566	7.4707
	9:00	121	50	NIL	NIL			4.5566	7.4707
	10:00	162	70	NIL	NIL			4.5566	7.4707
	11:00	120	70	NIL	NIL			4.5566	7.4707

Date	**TIME**	**CO**	**NO**	**Fault CO**	**Fault NO**	**Daily Average CO**	**Daily Average NO**	**Longitude**	**Latitude**
	12noon	100	50	NIL	NIL			4.5566	7.4707
	13:00	105	50	NIL	NIL			4.5566	7.4707
	14:00	90	50	NIL	NIL			4.5566	7.4707
	15:00	85	60	NIL	NIL			4.5566	7.4707
	16:00	105	78	NIL	NIL			4.5566	7.4707
	17:00	100	60	NIL	NIL	90.55	55	4.5566	7.4707
29/09/2014	7:00	85	50	NIL	NIL			4.5566	7.4707
	8:00	100	55	NIL	NIL			4.5566	7.4707
	9:00	125	74	NIL	NIL			4.5566	7.4707
	10:00	150	50	NIL	NIL			4.5566	7.4707

	11:00	130	50	NIL	NIL			4.5566	7.4707
	12noon	187	54	NIL	NIL			4.5566	7.4707
	13:00	141	50	NIL	NIL			4.5566	7.4707
	14:00	162	50	NIL	NIL			4.5566	7.4707
	15:00	120	50	NIL	NIL			4.5566	7.4707

Date	**TIME**	**COx**	**NO$_x$**	**Fault CO$_x$**	**Fault NO$_x$**	**Daily Average CO$_x$**	**Daily Average NO$_x$**	**Longitude**	**Latitude**
	16:00	100	50	NIL	NIL			4.5566	7.4707
	17:00	125	50	NIL	NIL			4.5566	7.4707
30/09/2014	7:00	114	50	NIL	NIL			4.5566	7.4707
	8:00	109	60	NIL	NIL			4.5566	7.4707
	9:00	100	78	NIL	NIL			4.5566	7.4707
	10:00	100	62	NIL	NIL			4.5566	7.4707
	11:00	115	60	NIL	NIL			4.5566	7.4707
	12noon	120	55	NIL	NIL			4.5566	7.4707
	13:00	125	74	NIL	NIL			4.5566	7.4707

	14:00	150	50	NIL	NIL			4.5566	7.4707
	15:00	130	50	NIL	NIL			4.5566	7.4707
	16:00	187	54	NIL	NIL			4.5566	7.4707
	17:00	141	50	NIL	NIL			4.5566	7.4707

APPENDIX B

Bill Of Materials For Complete FDI DAS.DSN

Design Title	: Complete FDI DAS.DSN
Author	: Ofoegbu Ositadinma Edward
Design Last Modified	: Wednesday, March 23, 2016
Total Parts In Design	: 31

12 Resistors

Quantity:	References	Value	Cost
12	R1-R12	1k	N200

2 Capacitors

Quantity:	References	Value	Cost
2	C1, C2	33p	N250

2 Integrated Circuits

Quantity:	References	Value	Cost
2	U1, U3	AT89C51RB2	N5000

4 Transistors

Quantity:	References	Value	Cost
4	Q1-Q4	2N2926	N180

7 Diodes

Quantity:	References	Value	Cost
7	D1-D7	LED	N70

4 Miscellaneous

Quantity:	References	Value	Cost
1	BUZ1	BUZZER	N150
1	LCD1	LM016L	N800
2	X1, X2	CRYSTAL	N300
2	S1, S2	MICS-4514	N35000

Bill Of Materials For central remote terminal unit.DSN

Design Title	:central remote terminal unit.DSN
Author	: Ofoegbu Ositadinma Edward
Revision	:
Design Created	:Tuesday, March 22, 2016
Design Last Modified	:Tuesday, March 22, 2016
Total Parts In Design	:7

2 Resistors

Quantity:	References	Value	Cost
2	R1, R2	1k	N100

1 Capacitors

Quantity:	References	Value	Cost
1	C1	33p	N50

1 Integrated Circuits

Quantity:	References	Value	Cost

1	U1	AT89C51RD2	N700

1 Diodes

Quantity:	References	Value	Cost
1	D1	LED	N10

2 Miscellaneous

Quantity:	References	Value	Cost
1	BUZ1	BUZZER	N150
1	X1	CRYSTAL	N100

APPENDIX C: Source Codes for System Controller

```
#define RAND_MAX10

#include <reg52.h >
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <Lib_UART.h>


#define LCD_PRINTF
#include"Lcd_4.h"


#define true     1
#definefalse     0

bit Data_Complete;
sbit GPS_PIN =P1^2; //soft ware uart recieve pin

sbit Sensor1 =P1^1;
sbit Sensor2 =P1^2;

/*some useful definitions  */
#define CR         0x0d
```

```
#define LF        0x0a

#define SPACE                   0x20
#define COMMA                   0x2C
#define FULLSTOP  0x2E
#define SIZE              70
#define MAXSIZE   66                    /* GPS at most, sends 80 or so chars per message string.
So set maximum to 100 */

                unsigned char idata latitudeString[13];             //array to hold the latitude string
        unsigned char idata longitudeString[13];                 //array to hold the longitude string
        unsigned char  *pChar;
        short count;
         // bit network_flag;

                 unsigned  char lastCommaPosition;

  unsigned int   j, k;                                                  /* dummy variable */
  unsigned int   i;                                                     /*  Number  of  chars  read  per  GPS
message string */
  unsigned int   numLinesRead;                    /* Number of GPS strings read */

        ////////////////////////////////////////////////////////

        //////////////////////////////////////////////////////////


unsigned char Command;
short bit_no=0;
sbit RX_PIN =P3^2;   //soft ware uart transmit pin
sbit TX_PIN =P3^3;   //soft ware uart transmit pin


#include <Soft_Uart.h>
        ////////////////////////////////////////////////////////////////////////////////////////////
//converts hex
//////////////////////////////////////////////////////////////////////////////////////////
char Char_to_hex(char Char)
{
         char byte;       if(isalpha (Char))
         {
                Char-=48;
                byte=(((Char>>1)&0x0F)|(Char&0x0F))+0x01;
        }
        else
        byte=(Char&0x0F);
        return byte;
}
////////////////////////////////////////////////////////////////////////////////
/*check for errors that might have occur during transmission*/
char Cr_Check_Error(char *statement)
{
          int val,a;
          char checksum,checkbyte;
          char idata checktemp [3];
          val=1;
          a=0;
          checksum=0x00;
           while(*(statement+val)!='*'){                        //manuall calculation of checksum
                checksum ^=*(statement+val);                          //the checksum is calculated
by exorring
                val++;                                                              //all
characters between '$' and '*' character of the gps string
                }
                val=1;
                while(*(statement+val)!='*')val++;                //the checksum byte is after the '*'
character of the gps string
                 val++;

          while(*(statement+val)!=CR && a<3)                  //extract checksum from gps string
           {checktemp[a]=*(statement+val) ;
           val++;
           a++; }
          checkbyte=((Char_to_hex(checktemp[0]))<<4)|(Char_to_hex(checktemp[1]));
//convert gps ascii to hex format
          if(checksum==checkbyte) return false;                           //compare              manually
calculated checksum and gps checksum value
          else    return true;                                                       //The two must have
same values if a valid gps string is recieved else error...
}



////////////////////////////////////////////////////////////////////////////////
unsigned char *strncmpy(unsigned char *Source,unsigned char *destination, short index, short
length)
{
        short i=0;
        for(i=0;i=length;i++)destination[i]=Source[index+i];
        return destination;
}
```

```
/*****************************************************************************
*/
/*  ******************************************    */
/*  Function:     void Get_Gps_Cordinates(void)                                  */
/*                                                  */
/*  Purpose:      extract the gps cordinates from the gps reciever           */
/*                                                  */
/*  CallParams:   0.                                         */
/*                                                  */
/*  ReturnValues: 0
                                                                          */
/*                                                  */
/*  Requires :    Built in Uart function.                                 */
/*                                                  */
/*  Example  :    void Get_Gps_Cordinates();
                              */
/*                                                  */
/****************************       CHANGE LOG      **************************/
/* Version | ACTION                              |  DATE  | SIG  */
/* --------|------------------------------------------------|--------|----- */
/*         |                                      |       |      */
/*    0.00 | Created function                     | 200512 | ST   */
/*         |                                      |       |      */
/*****************************************************************************
*/
void Get_Gps_Cordinates(void)
{
            unsigned char  idata stringRead[SIZE];          /* Buffer collects chars read
from GPS */
                  unsigned char  charRead;                  /* char read from COM port
*/

                  char Satelite_Fixed_Status;
                  Data_Complete=false;
                  do{
                  do{
                  charRead =Soft_Uart9600_Read();    /* read char from serial port */
            }while(charRead != '$') ;          /* GPS messages start with $ char */
                  /* otherwise not a $ character... so loop back until one arrives */
                  i = 0;
                  numLinesRead++;
                  stringRead[i] = charRead;

      ////////////////////////////////////////////////////////////////////////////
                  do {
                  charRead =Soft_Uart9600_Read();
```

```
                        if(  (charRead  !='\0')  &&  (isalnum(charRead)  ||    isspace(charRead)  ||
ispunct(charRead)) ) {
                                i++;
                                stringRead[i] = charRead;
                        }
                        } while(charRead != COMMA && i<8);
                /* Check if string we collected is the $GPRMC message */

        //$GPRMC,104706.000,A,1026.1408,N,00723.7895,E,0.00,177.47,071213,,,A*63
                /*$GPRMC,142213.000,A,1030.2170,N,00726.0835,E,0.00,0.0,090212,,*37
        */
                }while(stringRead[3] != 'R' && stringRead[4] != 'M' && stringRead[5] != 'C');
         /*extract the gps cordinates and save them into a temp  storage*/
         do {
           charRead =Soft_Uart9600_Read();
           if(   (charRead    !='\0')    &&    (isalnum(charRead)    ||       isspace(charRead)    ||
ispunct(charRead)) ) {
                                i++;
                                stringRead[i] = charRead;
           }
         } while(charRead != CR && i<MAXSIZE);
                                /* By this point, a complete GPS string has been read so save it to
stringRead*/
         /* Append the null terminator to the string read */
          stringRead[i+1] = '\0';
                pChar = stringRead;
                GPS_PIN=~GPS_PIN;
                //extract the
                 Satelite_Fixed_Status=stringRead[18];
                /*check if gps module has found a reference point*/
                //if (Satelite_Fixed_Status!='A')        return;
                /*check for errors in reception*/
                if(!Cr_Check_Error(stringRead))        //
                {

                        GPS_PIN=true;
                 /* Get lattitude: ddmm.mmmmm */
           strncmpy(stringRead,latitudeString,21,11);
            /* Get longitude: dddmm.mmmm */
           strncmpy(stringRead,longitudeString,33,12);
                                Data_Complete=true;
/////////////////////////////////////////////////////////////////////
                }
}

unsigned int Sensor1_=0;
unsigned int Sensor2_=0;

void feedback(){
	unsigned char threshold=0;
if(gas>threshold){
return adjustdown;
	elseif(gas<threshold){
return adjustup;
}


}
void GetConcentration()
{
if(Sensor1)Sensor1_=rand()%10; else Sensor1_=55;
if(Sensor2)Sensor2_=rand()%10; else Sensor2_=55;
}

sbit Buzzer = P1^0;


void SendData()
{
Out_Path=Serial_Com_Port;
	GetConcentration();
	printf("LAT:        %s,        LNG:        %s#        SNR1:        %i,        SNR2:
%i\n",latitudeString,longitudeString,Sensor1_,Sensor2_);
	Delay_ms(2000);
	Out_Path=Lcd;
}


void main(void)
{
	unsigned char adjustdown, adjustup;
	P3=0xFF;
	P1=0xFF;
	Delay_ms(2000);
	Uart_Init(9600);
	Out_Path=Lcd;
	Lcd_Init();
	printf("\f\1 Welcome!!!\n Please Wait...");
	//printf("DATA ACQUISITION\n SYSTEM");
	Delay_ms(2000);
	//printf("DATA AQUISITION SYSTEM");
	printf("\f\1  Invalid Data\nOr No Reception! ");
```

```
        while(1)
        {
        Get_Gps_Cordinates();
                if(Data_Complete==true)
                printf("\1LAT: %s    \nLN: %s   ",latitudeString,longitudeString);
                else printf("\f\1  nvalid Data\nOr No Reception! ");
                SendData();
        }
        if(P3^4==0){
        feedback();
        }
```

APPENDIX D: Source Code for Supervisory Microcontroller in DAS

```
#include <stdio.h>>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <Lib_UART.h>
#include <math.h>

#define OUTPUTMATRIX y       0x0d
#define INPUTMATRIX u        0x0a

#define OUTPUTMATRIX f            0x20
#define INPUTMATRIX e             0x2C
#define Lcp        0x0e
#define Ucp        0x0f

unsigned int e;
unsigned int f;
unsigned int u;
unsigned int y;
unsigned int lcp_ co_=75;
unsigned int ucp_co_= 450;
unsigned int lcp_ no_=40;
unsigned int ucp_no_= 107;

void Get_cO_data_dco()
void Get_cO_data_dno()

int main()
{
   int r,c,e[3][3],f[3][3],u[3][3],y[3][3]sum[3][3], mul[3] [3],i,j;
```

```
 getchar( “Lcp ,e");
 for  r [1][ 0][0]
 for c[ 0][1][0]
  for e[0][ 0][1]
   scanf("%d",&r);
 putchar(“e");
  for e [-1] e[0],e[0]
 for e [0] e [-1], e[0]
  for e[0], e [0], e[ -1]
 scanf("%d",&c);
 printf("\n elements of 1st matrix:\n , e");

 getchar("fault set string: ,f");
 if ( f ==-1, f== 0,f==0)
 if  (f== 0, f==-1, f==0)
  if (f==0, f== 0, f==-1)
   scanf("%d",&r);
 putchar("f");
  if ( f == -1, f== 0,f==0)
 if  (f== 0, f== -1, f==0)
  if (f==0, f== 0, f== -1)
 scanf("%d",&c);
 printf("\n elements of 2nd matrix:\n , f");

getchar(" fault set string ,y");
 if ( y ==1, y== 0,y==0)
 if  (y== 0, y==1, y==0)
  if (y==0, y== 0, y==1)
   scanf("%d",&r);
 putchar("y");
  if ( y == 1, y== 0,y==0)
 if  (y== 0, y== 1, y==0)
  if (y==0, y== 0, y== 1)
 scanf("%d",&c);
 printf("\n elements of 3rd matrix:\n , y");

getchar(": ,u");
 if ( u ==1, u== 0,u==0)
 if  (u== 0, u==1, u==0)
  if (u==0, u== 0, u==1)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( u== 1, u== 0,u==0)
 if  (u== 0, u== 1, u==0)
  if (u==0, u== 0, u== 1)
 scanf("%d",&c);
 printf("\n elements of 4th matrix:\n , u");
  r = e * u - f *y
   if (r=0)
   Printf("ACK, r");
   putchar(getkey(), n_F1, n_f2, n_f3, m_F1, m_f2, m_f3);

   else {

  for(n = dco;n < 75; n = n+1){
       printf("Enter number of rows (between 1 and 3): ,u");
 if ( u == 75, u== 0,u==0)
 if  (u== 0, u== 75, u==0)
  if (u==0, u== 0, u==75)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( u== 75, u== 0,u==0)
 if  (u== 0, u== 75, u==0)
  if (u==0, u== 0, u== 75)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");

for(n = dco;n < 75; n = n+1){
       printf("Enter number of rows (between 1 and 3): ,y");
 if ( y== 75, y== 0,y==0)
 if  (y== 0, y== 75, y==0)
  if (y==0, y== 0, y== n)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( y== 75, y== 0,y==0)
 if  (y== 0, y== 75, y==0)
  if (y==0, y== 0, y== n)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");
  r= e*u + f*y
  if (r=0)
  printf("ack,r");
  putchar(getkey(), n,)
else {

 printf("Enter number of rows (between 1 and 3): ,e");
 if ( e ==1, e== 0,e==0)
 if  (e== 0, e==1, e==0)
  if (e==0, e== 0, e==n /75)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,e");
  if ( e == 1, e== 0,e==0)
 if  (e== 0, e== 1, e==0)
  if (e==0, e== 0, e== n/75)
 scanf("%d",&c);
 printf("\n elements of 1st matrix:\n , e");

 printf("Enter number of rows (between 1 and 3): ,f");
 if ( f ==-1, f== 0,f==0)
 if  (f== 0, f==-1, f==0)
  if (f==0, f== 0, f==-1)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,f");
  if ( f == -1, f== 0,f==0)
 if  (f== 0, f== -1, f==0)
  if (f==0, f== 0, f== -1)
 scanf("%d",&c);
 printf("\n elements of 2nd matrix:\n , f");

 r= e*u + f *y
  if (r=0)
 printf("ack,r");
  putchar(getkey(), n,)
   }


  else {

  for(n = dco;n > 450; n = n+1){
       printf("Enter number of rows (between 1 and 3): ,u");
 if ( u == 450, u== 0,u==0)
 if  (u== 0, u== 450, u==0)
  if (u==0, u== 0, u==450)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( u== 450, u== 0,u==0)
 if  (u== 0, u== 450, u==0)
  if (u==0, u== 0, u== 450)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");

for(n = dco;n < 75; n = n+1){
       printf("Enter number of rows (between 1 and 3): ,y");
 if ( y== 450, y== 0,y==0)
 if  (y== 0, y== 450, y==0)
  if (y==0, y== 0, y== n)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( y== 75, y== 0,y==0)
 if  (y== 0, y== 75, y==0)
  if (y==0, y== 0, y== n)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");
  r= e*u + f*y
  if (r=0)
  printf("ack,r");
  putchar(getkey(), n,)
else {

 printf("Enter number of rows (between 1 and 3): ,e");
 if ( e ==1, e== 0,e==0)
 if  (e== 0, e==1, e==0)
  if (e==0, e== 0, e==450/n)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,e");
  if ( e == 1, e== 0,e==0)
 if  (e== 0, e== 1, e==0)
  if (e==0, e== 0, e== 450/n)
 scanf("%d",&c);
 printf("\n elements of 1st matrix:\n , e");

 printf("Enter number of rows (between 1 and 3): ,f");
 if ( f ==-1, f== 0,f==0)
 if  (f== 0, f==-1, f==0)
  if (f==0, f== 0, f==-1)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,f");
  if ( f == -1, f== 0,f==0)
 if  (f== 0, f== -1, f==0)
  if (f==0, f== 0, f== -1)
 scanf("%d",&c);
 printf("\n elements of 2nd matrix:\n , f");

 r= e*u + f *y
  if (r=0)
 printf("ack,r");
  putchar(getkey(), n,)


  for(m = dno;m < 40; m = m+1){
      printf("Enter number of rows (between 1 and 3): ,u");
 if ( u == 40, u== 0,u==0)
 if  (u== 0, u== 40, u==0)
```

```
  if (u==0, u== 0, u==40)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( u== 40, u== 0,u==0)
 if  (u== 0, u== 40, u==0)
  if (u==0, u== 0, u== 40)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");

for(m = dno;m < 40; m = m+1){
       printf("Enter number of rows (between 1 and 3): ,y");
 if ( y== 40, y== 0,y==0)
 if  (y== 0, y== 40, y==0)
  if (y==0, y== 0, y== m)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( y== 40, y== 0,y==0)
 if  (y== 0, y== 40, y==0)
  if (y==0, y== 0, y== m)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");
  r= e*u + f*y
  if (r=0)
  printf("ack,r");
  putchar(getkey(), m,)
else {

 printf("Enter number of rows (between 1 and 3): ,e");
 if ( e ==1, e== 0,e==0)
 if  (e== 0, e==1, e==0)
  if (e==0, e== 0, e==m /40)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,e");
  if ( e == 1, e== 0,e==0)
 if  (e== 0, e== 1, e==0)
  if (e==0, e== 0, e== m/40)
 scanf("%d",&c);
 printf("\n elements of 1st matrix:\n , e");

 printf("Enter number of rows (between 1 and 3): ,f");
 if ( f ==-1, f== 0,f==0)
 if  (f== 0, f==-1, f==0)
  if (f==0, f== 0, f==-1)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,f");
  if ( f == -1, f== 0,f==0)
 if  (f== 0, f== -1, f==0)
  if (f==0, f== 0, f== -1)
 scanf("%d",&c);
 printf("\n elements of 2nd matrix:\n , f");

 r= e*u + f *y
  if (r=0)
 printf("ack,r");
  putchar(getkey(), m,)
   }


  else {

  for(m = dno;m > 107; m = m+1){
       printf("Enter number of rows (between 1 and 3): ,u");
 if ( u == 107, u== 0,u==0)
 if  (u== 0, u== 107, u==0)
  if (u==0, u== 0, u==107)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( u== 107, u== 0,u==0)
 if  (u== 0, u== 107, u==0)
  if (u==0, u== 0, u== 107)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");

for(m = dco;m < 107; m = m+1){
       printf("Enter number of rows (between 1 and 3): ,y");
 if ( y== 107, y== 0,y==0)
 if  (y== 0, y== 107, y==0)
  if (y==0, y== 0, y== m)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( y== 107, y== 0,y==0)
 if  (y== 0, y== 107, y==0)
  if (y==0, y== 0, y== m)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");
  r= e*u + f*y
  if (r=0)
  printf("ack,r");
  putchar(getkey(), m,)
else {

 printf("Enter number of rows (between 1 and 3): ,e");
 if ( e ==1, e== 0,e==0)
 if  (e== 0, e==1, e==0)
  if (e==0, e== 0, e==107/m)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,e");
  if ( e == 1, e== 0,e==0)
 if  (e== 0, e== 1, e==0)
  if (e==0, e== 0, e== 107/m)
 scanf("%d",&c);
 printf("\n elements of 1st matrix:\n , e");

 printf("Enter number of rows (between 1 and 3): ,f");
 if ( f ==-1, f== 0,f==0)
 if  (f== 0, f==-1, f==0)
  if (f==0, f== 0, f==-1)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,f");
  if ( f == -1, f== 0,f==0)
 if  (f== 0, f== -1, f==0)
  if (f==0, f== 0, f== -1)
 scanf("%d",&c);
 printf("\n elements of 2nd matrix:\n , f");

 r= e*u + f *y
  if (r=0)
 printf("ack,r");
  putchar(getkey(), m,)

  for(n = dco; m= dno; n > 75; n<450; m > 40; m<107; m=m+1; n = n+1){
       printf("Enter number of rows (between 1 and 3): ,u");
 if ( u == 375, u== 75 ,u==450)
 if  (u== 450, u== 75, u==375)
  if (u==75, u== 450, u==375)
   scanf("%d",&r);
 printf("Enter number of columns (between 1 and 3): ,u");
  if ( u== 375, u== 75,u==450)
 if  (u== 450, u== 75, u==375)
  if (u==75, u== 450, u== 375)
 scanf("%d",&c);
 printf("\n elements of matrix:\n , u");

for(n = dco;n < 75; m= dno; n > 75; n<450; m > 40; m < 107; m=m+1; n = n+1){
       printf("Enter number of rows (between 1 and 3): ,y");
 if ( y== n-75, y== 450-n ,y==263)
 if  (y== 450-n, y== 263, y==n-75)
  if (y==263, y== n-75, y== 450-n)
```

```
      scanf("%d",&r);
    printf("Enter number of columns (between 1 and 3): ,u");
     if ( y== n-75, y== 450-n,y==263)
    if  (y== 450-n, y== 263, y==n-75)
     if (y==263, y== n-75, y== 450-n)
    scanf("%d",&c);
    printf("\n elements of matrix:\n , u");
     r= e*u + f*y
     if (r=0)
     printf("ack,r");
     putchar(getkey(), n,)
   else {

    printf("Enter number of rows (between 1 and 3): ,e");
    if ( e ==1, e== 0,e==0)
    if  (e== 0, e==1, e==0)
     if (e==0, e== 0, e==n/75)
      scanf("%d",&r);
    printf("Enter number of columns (between 1 and 3): ,e");
     if ( e == 1, e== 0,e==0)
    if  (e== 0, e== 1, e==0)
     if (e==0, e== 0, e== n/75)
    scanf("%d",&c);
    printf("\n elements of 1st matrix:\n , e");

    printf("Enter number of rows (between 1 and 3): ,f");
    if ( f ==-1, f== 0,f==0)
    if  (f== 0, f==-1, f==0)
     if (f==0, f== 0, f==-1)
      scanf("%d",&r);
    printf("Enter number of columns (between 1 and 3): ,f");
     if ( f == -1, f== 0,f==0)
    if  (f== 0, f== -1, f==0)
     if (f==0, f== 0, f== -1)
    scanf("%d",&c);
    printf("\n elements of 2nd matrix:\n , f");

    r= e*u + f *y
     if (r=0)
    printf("ack,r");
     putchar(getkey(), n,)

sbit P1^0=normal;
sbit P1^1=above;
sbit P1^2=below;
sbit dbrate=ACC^7;
void init_serial();

unsigned char q
            P1=0X00;
init_serial();
ACC=PCON;
dbrate=1;
PCON=ACC;
while(1){
while(RI==0);
q=SBUF;
RI=0;
if(q=='A'){
normal=1;
}elseif(q=='B'){
above=1;
}
elseif(q=='C'){
above=1;}else
{P1=0x00;}
}
}

void init_serial(){
SCON=0x50;
TMOD=0x20;
TH1=0xfa;
TR1=1;
TI=0;
RI=0;
}

    return 0;
}
```

APPENDIX E: Source Code for Remote Terminal Unit

```
#include <reg52.h>
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <Lib_UART.h>
#include <Lib_Delay.h>
#include <math.h>
void main(void)
{
Uart_Init(9600);
Delay_ms(2000);
printf("Initialising devices..\n");
while(1)putchar(getkey());
#define Lcp        0x0d
#define Ucp        0x0a

#define Dco              0x20
#define Dno              0x2C
#define Sco
#define Sno
float standard_deviation(float data[], int n);
float standard_deviation(float data[], int m);
unsigned int lcp_ co_=75;
unsigned int ucp_co_= 450;
unsigned int lcp_ no_=40;
unsigned int ucp_no_= 107;

void Get_cO_data_dco()
void Get_cO_data_dno()


int main()
{
   int n, dco;
   for(dco = n;dco > 75;dco < 450;Dco = n+1){
   if (n > = 75)
   printf("%d\n", n);
```

```
    else
        printf("%d\n", n_f1);
    if (n < = 450)
    printf("%d\n", n);
    else
        printf("%d\n", n_f2);
    }

    int n, dno;
     for(dno = m;dno > 40;dco < 107;Dco = n+1){
    if (m > = 40)
    printf("%d\n", m);
    else
        printf("%d\n", m_f1);
    if (m < = 450)
    printf("%d\n", m);
    else
        printf("%d\n", m_f2);
    }
int n;/*variable for CO*/
  int m;/*Varaible for NO*/
  float s_1;/*standard Deviation*/
float s_2;/*standard Deviation*/
  float u_1;/*Arthimetic mean*/
float u_2;/*Arthimetic mean*/
int k;/*number of data*/

   for (n = dco; n < 75; n > 75; n < 450; n > 450;k=k+1; n = n+1){
        u = (n + n -1 + n - 2+ .. . .n-n)/k ;
       scanf("%f",&u)
        s_1 = sqrt(){
                   (sqr ( n-u) + sqr (n-1-u) + sqr (n-2-u) + ......Sqr (n-n-u)/k;
        }

    printf("\n");
    printf("Standard Deviation = %.2f", standard_deviation(data,n));

   }

     for (m = dno;m < 75; m > 75; m < 450; m > 450;k=k+1; m = m+1){
        u = (m + m -1 + m - 2+ .. . .m-m)/k ;
       scanf("%f",&u)
        s_2 = sqrt(){
                   (sqr ( m-u) + sqr (m-1-u) + sqr (m-2-u) + ......Sqr (m-m-u)/k;
        }
    printf("\n");
    printf("Standard Deviation = %.2f", standard_deviation(data,m));

   }
  if (s_1 < 1.02*S_1 -1)
  printf("%d\n" ,n);
  else
       printf ("%d\n" , n_F1, n_f2, n_f3);

if (s_2 < 1.02*S_2 -1)
  printf("%d\n" ,m);
  else
       printf ("%d\n" , m_F1, m_f2, m_f3)

  unsigned char Command;
  short bit_no=0;
  sbit RX_PIN =P3^2; //soft ware uart transmit pin
  sbit TX_PIN =P3^3; //soft ware uart transmit pin
while(1)putchar(getkey(), n_F1, n_f2, n_f3, m_F1, m_f2, m_f3);

}
```