CENTRALIZED ENVIRONMENTAL NETWORK MONITORING AND REPORTING SYSTEM

By

NC RabyaMahmood
NC Muhammad Ahmed Mehmood
NC Javaria Khalid
NC AmalAtique

Project Supervisor

Major AsimRasheed

Submitted to the Faculty of Electrical Engineering, Military College of Signals National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E. Degree in Telecom Engineering

JULY 2012

ABSTRACT

CENTRALIZED ENVIRONMENTAL NETWORK MONITERING AND REPORTING SYSTEM

Traditional monitoring for un-manned sites is done by human operators from mountain tops to deserts. This requires huge number of labor operators and long working hours to achieve full time environmental conditions monitoring and their consequent reporting. Human operator monitoring is cumbersome as sometimes the un-manned site is a location which is not easily accessible. Reporting is sometimes also restricted and slowed down by bad weather conditions and poor communication. Computerized systemization is an effective means of monitoring that allows substantial saving through the reduction of continuous human labor, greater remote access, real time data processing, timelierreporting and alarm updates, and better deployment of human resource.

In this project, Client-Server model will be used for data acquisition and GSM for the reporting system. The Client-Server model takes advantage of the available Ethernet infrastructureto send data from the un-manned site to the server end and the Short Messaging Service (SMS) for the reporting of some unusual or unwanted event. An interface has been developed at the server end to receive and display the readings sent by the un-manned location and update the records in the database.

If commercially employed, this is a comprehensive system which accurately monitors an un-manned site via an efficient computerized monitoring system and also does the reporting in case of a mishap. The system also has the flexibility to be expanded for multiple sites with extensive control by the operator at the centralized server end.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

To Almighty Allah, for Whose greatness we do not have enough words,

To our parents and friends, without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

There is no success without the will of Allah. We are grateful to Allah, Who has given us guidance, strength and enabled us to accomplish this task. Whatever we have achieved, we owe it to Him, in totality.

We are also grateful to our parents, family, seniors and well-wishers for their admirable support. We would like to thank our supervisor Major AsimRasheed, for his help and motivation throughout the course of our project.

We are thoroughly obliged to Ufone for their courteous help in leading us to developing a practical approach to this project. We extend our gratitude to Mr. HasnainRiaz, for being a source of inspiration and innovation in our project selection phase.

We offer our regards and blessings to the University Administration especially our Lab Staffs for bearing with us throughout the course of our project and supporting us during the completion of this work.

Words will fail us in order to thank Flight Engineer PIA Mr. Syed Anwar Mahmood for his selfless help and unconditional support throughout our project. Without his patronage and guidance, this project would not have been possible.

Table of Contents

List of Figures

List of Abbreviations

CENMaRS	Centralized Environmental Network Monitoring and Reporting System
SIM	Subscriber Identity Module
GSM	Global System for Mobiles
GPS	Global Positioning System
BTS	Base Transceiver Station
USB	Universal Serial Bus
SMS	Short Messaging Service
Telco	Telecommunication Company
UART	Universal Asynchronous Receiver/Transmitter
TCP	Transmission Control Protocol
IP	Internet Protocol
GUI	Graphical User Interface
SNMP	Simple Network Management Protocol

CHAPTER 1

INTRODUCTION

1.1. Overview

Our world is facing serious financial crunch since last few years. This crisis has not only affected the common man but also service providers such as telecom operators. The need for efficient use of human resource, incorporating technology resources has risen manifolds. Due to peculiar nature of telecom infrastructure, operators are forced to deploy their resources at remote locations. Such infrastructure includes BTS stations, Microwave backhaul stations and repeater sites. These infrastructures are sometimes deployed in such a remote area where even human access is limited. Design and deployment requirements of these infrastructures include telecom resources as well as many administrative and environmental resources. Administrative and environmental resources include generators, UPS, heat sensors, smoke sensors, humidity sensors, entry sensors. Figure 1-1 is such a BTS site where there is dire need of a monitoring system.

Figure 1-1 BTS Site

The solution in the form of CENMaRS i-e Centralized Environmental Network Monitoring and Reporting System is a combination of hardware and software application. From hardware perspective, the solution includes the desired sensors, linked with industry grade controller, local audible and visual alarms. The controller is linked with the Centralized server through operator existing network resources. Centralized server, backed by secondary server one regional server will be part of system. A visual / audible reporting module is also attached with each server. However from software perspective, the system includes the controller application, frontend application at centralized, secondary and regional servers, backend database at secondary servers, encryption module for secure database, reporting module application and secure communication application for data sharing between controller to centralized server as well as among servers.

1.2. Problems

Rapidly increasing deployment of unmanned sites is often looked after through periodic visits. A nontechnical gatekeeper is normally deployed for protection purposes. In such scenario, timely reporting and subsequent response is big question mark, especially for far flung areas. Moreover, deployment of most of the telecom infrastructure is either decade old or was done in the financially stable periods. Electricity crisis, floods and earthquakes have imposed many new restrictions for routine functionalities. The segmented, unlinked and partially planned deployment of administrative and environmental resources has also played its negative role in overall efficiency. Most of the telecom operators have installed Network Management Systems linking their communication resources accordingly. However, the environmental and administrative resources lack such centralized management system. Localized management systems not only lengthen the reporting and response chain, but also prolong the outage duration increasing the intensity of losses. The problems even get worst with the deployment to multi-vendor equipment incorporating variety of proprietary management systems. Other than natural and artificial calamities, even routine functions lacks optimum use of resources due to incorrect or delayed reporting. Increase in unmanned sites due to financial burdens makes reporting even impossible in many cases.

1.3. Statement

In view of the problem stated, there is a dire need to efficiently use the available resources to hook all resources with the technical hub.

The project CENMaRS i.e."Centralized Environmental Network Monitoring and Reporting System" is aimed to develop a sensor based monitoring and reporting system for Telecom operators. The project targets to monitor the environmental changes in an unmanned site for centralized management through comprehensive database and reporting system.

1.4. Approach

Considering the severity of the problem and losses incurred due to lack of solution to the problem, a discrete and generic solution has been proposed[1]. The proposed solution incorporated collection of raw data through existing sensors or new deployed as per need. The data will be processed and filtered through localized controller installed at the remote site. The localized controller will generate the visible / audible alarms locally. The alarm data according to type of location, equipment, sensitivity, priority and type of response required will be sent to a centralized server through available network resources of the operator. Standard network management protocols will be used for data sharing between controllers and servers to enhance the target users and future expansion. Figure 1-2 is the architectural overview of the project.

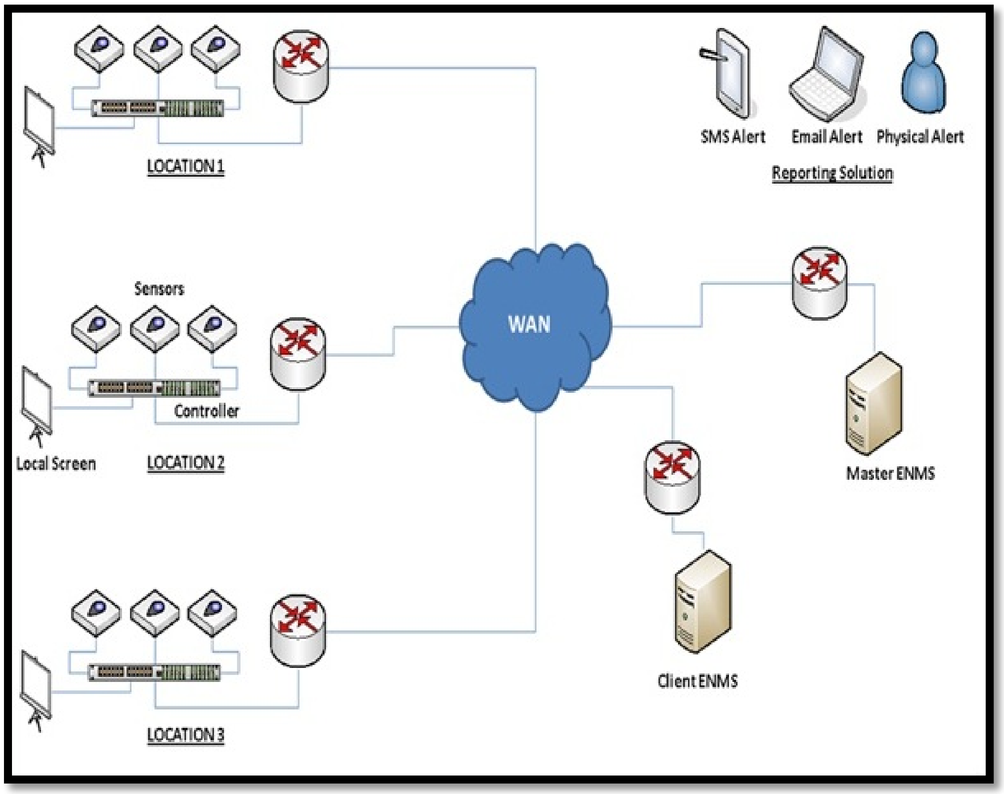


Figure 1-2 CENMaRS Architecture

The centralized server, backed by secondary and regional servers will be the hub of all automated management and reporting control procedure. Assuming large number of deployed sites, number of administrative and environmental sensors and frequency of alarms, a dedicated database will be linked with the server. A front end graphical user interface (GUI) will be developed to show the alarm and reporting logs for current and historical data. Efforts will be made to make the GUI as flexible and realistic as possible according to user requirements. The database will contain history of all linked equipment according to its usage and past records.

The centralized server will be linked with the reporting module to initiate alarms for response team using SMS, email and other audible / visual devices and interfaces. The automated reporting solution will increase the reporting level upto three tiers according to predefined rules. Efforts will be made to keep the reporting module open for future technologies as well.

The centralized server will be linked with secondary and regional servers for redundancy as well as regional reporting and control. The rights of regional server will be flexible according to user requirements.

The entire backend networking will be made secure using encryption techniques to avoid false alarm generation as well as data manipulation. The logs of each login and logout of controller and servers will be maintained accordingly.

Although few solutions have been proposed by some foreign vendors, but their cost and efficiency is a big hurdle for adaptation. Lack of generic approach and closed form solutions make the choice highly complicated in current financial crisis. Moreover neither any available solution covers entire equipment & sensor breed nor is scalable for thousands of sites.This approach is made as generic as possible, to enhance the scope and usage. The low cost controller and sensors will not only make an operator much relax

about the state of their unmanned site, but will also provide them relief financially and administratively.

1.5. Objectives

The project was developed with the drive of taking a step towards the amelioration of the current financial crisis inTelco’s due to the wastage of manpower. Various objectives were set and hence after, were successfully achieved. A brief overview of the project aims includes developing the project for multiple locations; the controller will be made scalable. It will be integrated with multiple sensors, to ensure the project efficiency, alarm data will be gathered at centralized server for timely reporting, efficient monitoring and response system, the aim is to grasp the present environmental condition through centralized environmental network monitoring, it will develop a cost effective project, having a positive impact, to have a better insight into how the sites and applications are performing and can better manage our resources, including staff, hardware and software, in the telecom industry, problems are being faced due to the overutilization of human resources as well as the continual monitoring of equipment. CENMaRS caters for the problems by developing a system for unmanned locations, reducing the wastage of manpower.

CHAPTER 2

LITERATURE REVIEW

The overview of the project describes sensors at remote sites / locations that would be connected to the controller which would be sending the raw data to the server over the Intranet. The comprehensive database will be maintained at the server end. The processed and filtered data will be displayed using a Graphical User Interface GUI. The reporting module would be connected with the database in case a value crosses the predefined threshold.

The project is categorized into the following parts, Sensor Network, Control system (Controller), Network System, Database server, Front-End Network Management System and Reporting System with the details and alternatives of each discussed later.

2.1. Sensor Network

2.1.1. Introduction

Sensor interface systems monitor the functioning and the conditions of the sites where they are being used. These sensors can perform many functions, e.g. monitor temperature, give outage detection and sense power quality etc. In this way, the control center without delay receives accurate information about the actual condition of the site.

A sensor network consists of multiple smalland portable detection stations called sensor nodes. Sensor is a detector that measures a physical quantity and converts it into a signal that can be observed. The sensors may be digital or analog in design. Every digital sensor node is equipped with a transducer, transceiver, microcomputer and power source. The same functionality for analog sensors is done using the sensor interface unit. The transducer generates electrical signals based on the sensed physical variations. The transceiver, which may be wired or wireless, then sends the data to a control system.

2.1.2. Sensor Sensitivity

The sensor sensitivity indicates how much the sensor's output changes when the quantity to be measured changes. For example, if the mercury in a thermometer moves 0.5 cm when the temperature changes by 1 °C, the sensitivity is 0.5 cm/°C. Sensors that measure very small changes must have very high sensitivities. Sensors also have an impact on what they measure; for instance, a room temperature thermometer inserted into a hot cupof liquid cools the liquid while the liquid heats the thermometer. Sensors need to be designed to have a small effect on what is measured; making the sensor smaller often improves this and may introduce other advantages.

2.1.3. Sensor Characteristics:

The characteristics of a good sensor are that it is only sensitive to the measured property, it is insensitive to any other property expected to be encountered and it does not affect the measured property.

A sensor takes in an input and converts it into an output quantity .The output signal of a sensor is linearly proportional to the observed value or it is a multiple of the measured property. The sensitivity is defined as ‘the ratio between output signal and measured property’. For example, if a sensor measures temperature and has anoutput as voltage, the sensitivity will be a constant with the unit [V/K]; this sensor is linear because the ratio of voltage and temperature is constant at all points.

2.1.4. Sensor Deviations

The deviations that can be observed in a sensor include the sensitivity maywhen using practically then from the specified value. This is called a sensitivity error.As the range of the output signal is always limited within a bracket, the output signal will reach a maximum or minimum when the measured property exceeds the limits.If the output signal is not zero when the measured property is zero, the sensor is said to have an offset or bias. This is known as the output of the sensor at zero input.If the sensitivity is variable over the defined range of the sensor, this effect is known asnonlinearity. This is defined on the basis of the output difference from ideal behavior over the full range of the sensor.If the deviation is caused by a sudden change of the measured property, there is a dynamic error. This behavior is explained with a Bode plot that shows sensitivity error and phase shift as a function of the frequency of a periodic input signal.If the output signal slowly varies irrespective of the measured property, this is defined as drift.A slow degradation of sensor properties is said to be occurring if long time drift is observed over a long period of time.Signal is randomly varied due to the noise present at all times.If the sensor being used has a digital output, the output is basically an approximatedvalue of the measured property. This is called digitization error.The sensor may be sensitive to variables other than the property being measured. For example, most sensors are affected by the temperature of their environment.

All these deviations in sensors can be classified as random errors or systematic errors. Systematic errors can be remunerated by means of some calibrationtechnique. Whereas noise is the main reason for random error and can be minimized by signal processing, such as filtering.

2.1.5. Sensor Output Values

In general, most following are the two categories in which the sensors fall are Analog Sensors and Digital Sensors.

An analog sensor, such as a CdS cell (Cadmium Sulfide cells measure light intensity), can be wired into a circuit so that it will have an analog output that ranges from 0 volts to 5 volts. The value can be any possible value between 0 and 5 volts. An Analog Signal works like a tuner on a tape recorder. It can be turned up or down in a continuous manner. Figure 2-1 is an analog signal of a sensor, which depicts that all the value are in a continuous form.

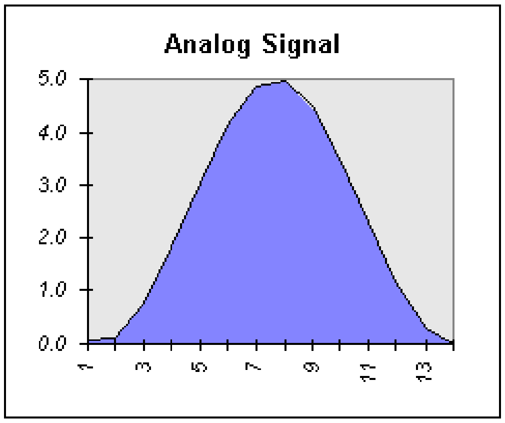


Figure 2-1 Analog Signal

Digital sensors generate discrete signals after measuring the property and converting them. This means that the sensor can give an output within a certain range, but the output value must increase in steps. 'Discrete Signals' usually have a stair step appearance when they are graphed on chart.

To use an analog sensor the data needs to be first converted into a digital form. Many microcontrollers have ADC ports that are capable of converting in to digital form. This form is mandatory as it eases further processing of the output. For example, converting the discrete output values to desired humidity or temperature values.But this is not built-in in all microcontroller boards so additional chips are required for this conversion. The Figure 2-2 is a digital signal showing discrete levels.

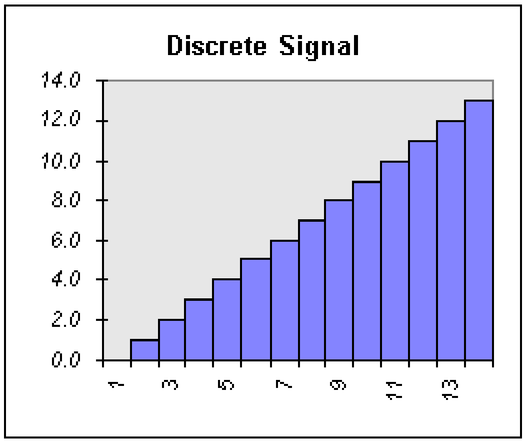


Figure 2-2 Discrete Signal

A sensor having digital output will produce only two values in its output, either on or off, as opposed to an analog output sensors which produce a continuous range for output value. The most common and simple example of a digital sensor is the touch switch. A typical touch switch is an open circuit with infinite resistance when it is not pressed and a short circuit with zero resistance when it is pressed. There are many types of digital sensors. Most of them are wired in the same form, using a pull-up resistor to force the line high, and to limit the amount of current that can flow. Analog sensors are preferred over digital sensors due to more accurate and precise results.

2.1.6. List of Sensors

The project aims to monitor large variety of sensors for a telecom site such as acoustic, sound, vibration e.g. microphone, hydrophone. Automotive, transportation e.g. Air-fuel ratio meter, speedometer, parking sensor. Chemical e.g. carbon-dioxide sensor, smoke detector.Electric current, electric potential, magnetic, radio e.g. metal detector, current sensor, magnetometer.Environment, weather, moisture, humidity e.g. rain sensor, steam gauge.Flow, fluid velocity e.g. water meter, gas meter, air-flow meter. Navigation instruments e.g. variometer, altimeter. Position, angle, displacement, distance, speed, acceleration e.g. accelerometer[2]. Optical, light, imaging, photon e.g. flame detector, photoelectric sensor. Pressure e.g. barograph, barometer.Thermal, heat, temperature e.g. heat flux sensor, temperature gauge. Proximity, presence e.g. alarm sensor, motion detector.

2.2.Control System (Controller)

A control system is a device or set of devices to manage or regulate the behavior of other devices or systems. Through interfacing the sensors with the controllers, the control system can immediately receive accurate information about the actual condition of the siteand can transfer the data onto the server database. There are two common types of control systems but they further have many combinations and variations: ‘logic or sequential’ controls, and ‘feedback or linear’ controls. Along with these main two types, there is also ‘fuzzy logic’, which aims to combine the design simplicity of logic with the efficiency of linear control. The control system will be developed using industrial graded miniature mother board. The local development will allow scalability, generic deployment and flexibility.

2.2.1. Control System Functionality

The control system will have the following functions. It will act as an interface for the Sensor Network, scrutinize the data from the sensors, and then convert raw data into intelligible form, remotely process the data, determining if variables exceed certain defined thresholds, locally filter the alarms, generate local alarms, authenticate/Verify the user to ensure security in the network and act as an interface for the Network Management System.

2.2.2. Controller Boards

The following can be used as controller boards in the project:PIC microcontroller board, ARM-microcontroller board and Motherboard. The choice of the controller board was based on detailed analysis of the available options.

2.2.2.1. PIC microcontroller board

A PIC microcontroller is a processor can be used to develop control based projects and has built in memory and RAM. This saves the requirement of designing a circuit whichrequires separate external RAM, ROM and peripheral chips. It is a very powerfulcontroller board that has many useful built in modules, e.g.EEPROM, Timers, Analogue Comparators, UART.

The main task in developing an embedded system is to determine the number and type of inputs and outputs required. Once the hardware requirements have been determined, the program is needed to be written and tested. After writing the program, the chip memory size can be determined.

A few important features of PIC microcontroller board are a PIC microcontroller is a powerful processor which is fully featured with built-in internal RAM, EEROM FLASH memory. Itmay occupy the space required by a 555 timer but it has a 10bit ADC, 1k memory, 2 timers,a comparator, high current I/O portsand a watch dog timer.Theadvantage of PIC is its Flash memory which allows to burn to the memory unlimited number of times.A PIC Microcontroller is capable of controlling outputs and reacting to inputs, e.g. drive a relay or read input buttons. With the larger PIC boards, it is possible to even drive LCDs or seven segment displays with very few control lines.The USART is a useful module port and saves the user from codingup a software version, saving valuable memory. This is done by interfacing it to a PC serial port through a MAX232 (or equivalent).

2.2.2.2. Arm-Microcontroller Board

The ARM microcontroller is a general purpose 32-bit microprocessor that offers efficient performance with very low power consumption. Its architecture is developed on the RISC principles;with the instruction set and related decode mechanism much simpler than those of micro-programmed CISCs. This simplicity of ARM boards results in a high instruction throughput and impressive real-time interrupt response from a cost-effective and small processor core.

The ARM board hasa 5-stage pipeline employed so that all parts of the processing and memory systems can operate continuously. At any one instant in time, multiple operations are typically in progress. These 5 stages are subsequent instruction fetch, next instruction decode, instruction execution, memory access, and write-back respectively.

The combination of architectural enhancements gives the ARM9 about 30 % better performances than an ARM7 running at the same clock rate:

Approximately 1.3 clocks per instruction (1.9 clocks per instruction for ARM7)

Approximately 1.1 Dhrystone MIPS/MHz (0.9 Dhrystone MIPS/MHz for ARM7)

In addition to all this, the ARM9 board includes enhanced DSP instructions, as well as an enhanced 32-bit MAC block.

The features of ARM board are very much more efficient than other boards. The SDRAM memory controller provides an interface between the system bus and External (off-chip) memory devices**.**Some device pins that are not dedicated to a specific peripheral function have been designed to be general purpose inputs, outputs, or I/Os. Also, some pins may be configured either as a specific peripheral function or a general purpose input, output, or I/O.Bit-level set and clear registers allow a single instruction set or clear of any number of bits in one port.A single register selects direction for pins that support both input and output modes. For input/output pins, both the programmed output state and the actual pin state can be read.There are a total of 12 general purpose inputs, 24 general purpose outputs, and six general purpose input/outputs. Additionally, 13 SDRAM data lines may be used as GPIOs if a 16-bit SDRAM interface is used (rather than a 32-bit interface).

2.2.2.3. Motherboard

A motherboard is the main printed circuit board (PCB) in computers and holds many of the crucial components of the system, providing connections for other peripherals.

A motherboard is responsible for the communication of the different system components as it provides the electrical connections. Along with this,it also connects the central processing unit and hosts other subsystems and devices.

A typical computer has its main memory, microprocessor, and other essential components connected to the motherboard. The other components such as external storage, controllers for video display and sound, and peripheral devices may be attached to the motherboard as plug-in cards or via cables, although in modern computers it is increasingly common to integrate some of these peripherals into the motherboard itself.

An important component of a motherboard is the microprocessor's supporting chipset, which provides the supporting interfaces between the CPU and the various buses and external components. This chipset determines, to an extent, the features and capabilities of the motherboard.The modern motherboards include sockets[3](or slots) in which one or more microprocessors may be installed, slots into which the system's main memory is tobe installed (typically in the form of DIMM modules containing DRAM chips). Achipset which forms an interface between the CPU's front-side bus, main memory, and peripheral busesnon-volatile memory chips (usually Flash ROM in modern motherboards) containing the system's firmware or BIOS. Aclock generator which produces the system clock signal to synchronize the various components slots for expansion cards (these interface to the system via the buses supported by the chipset).

Power connectors, which receive electrical power from the computer power supply and distribute it to the CPU, chipset, main memory, and expansion cards.All motherboards include logic and connectors to support commonly used input devices, such as PS/2 connectors for a mouse and keyboard.

2.3. Network System

Network management refers to the activities, methods, procedures, and tools that pertain

This will include connectivity is the basic need of any networking component. From project perspective, we will be looking for seamless connectivity, incorporating network management protocols, such as SNMP and CORBA, operation of the network system or the operational needs of the servicesincluding quick fault finding and alarm generation, administration of the resource management and allocation. It will include all the housekeeping that is required to keep the system under control, maintenance to perform upkeep and upgrades according to the operational procedure, for example, when the equipment should be replaced,when a new switch is added to a network, when a router needs a patch for an operating system image. Maintenance will also involveremedial and precautionary measures to make the managed network run smoother, such as adjusting device configuration parameters, provisioning to configure resources in the system network to support a given check. For example, setting up the network for a new client or node and security management system will incorporate authentication as well encrypted data flow against possible network attacks.

2.3.1. Classification of Network Models/Architectures

The computer systems/networks can be categorized into centralized and distributed. The distributed systems have been further categorized into the Client-Server model and the Peer-to-Peer model.

2.3.1.1. Peer to Peer Architecture

P2P is a group of applications that takes benefit of resources - storage, cycles, content, human existence available at the limits of the Internet. Accessing the decentralized resources is similar to operating in an environment of unsteady connectivity and unpredictable IP addresses, P2P nodes must function outside the DNS scheme and have important or total independence from central servers.

Distributedsystem architecture may be called a Peer-to-Peer network, if the users share a part of their own hardware resources (storage capacity, network link capacity, processing power, printers). These common resources are essential to offer the Service and substance offered by the network (e.g. file sharing or shared workspaces for alliance). They are available by other peers openly, without passing mediator entities. The participants of such anassociation are thus resource (Service and content) providers as well as resource (Service and content) requestors (Servant-concept).

Peer-to-Peer computing is the distribution of computer resources and services by straight exchange between systems. These resources and services include the exchange of information, processing cycles, cache storing, and disk storage for files. Peer-to-Peer computing takes benefit of existing desktop computing power and networking connectivity, allowing reasonable clients to influence their collective power to help the entire project.

In a Peer-to-Peer architecture, computers that have been previously used as clients communicate directly between themselves and can act as clients and a server both. This helps to reduce the load on servers and allows them to carry out specialized services (such as mail-list generation, billing, etc.) more efficiently. At the same time, Peer-to-Peer computing can lessen the requirement of IT organizations to grow parts of their infrastructure in order to provide and support certain services, such as backup storage. Figure 2-3 is a comparison between the peer-to-peer and hybrid architecture.

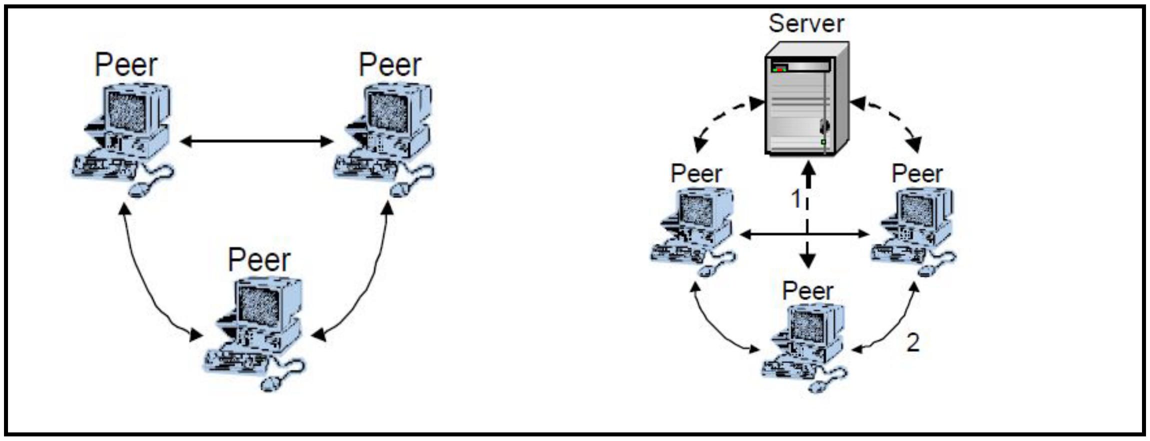


Figure 2-3 Pure peer-to-peer architecture vsHybrid peer-to-peer architecture

2.3.1.2. Advantages of peer to peer architecture In a pure Peer-to-Peer architecture,the system is stable and keeps on working even if one peer breaks down as this architecture is free from single-point failure. This enables other peers irrespective of each other's condition.

Peer-to-Peer gives the prospect to take gain of unused resources such as processing power for computations and storage capacity. In Client-Server architectures, the central system bears the greater part of the cost of the scheme. In Peer-to-Peer, all peers help extend the cost.

Peer-to-Peer prevents bottleneck such as traffic overload using centralserver architecture, because Peer-to-Peer can distribute data and balance request across the net without using a central server.

The system has better scalability due to distributed control and independent communication of peers.

2.3.1.3. Disadvantages of peer to peer architecture

The Peer-to-Peer solutions do not cater for high security standards desired by applications.

Even if the coverage of ADSL and Cable modem connections is increasing, the connections between the peers are usually not meant for high throughput.

A centralized system or a Client-Server system will only be able to communicate as long as the service provider keeps it running. If peers start to desert a Peer-to-Peer system, services will not be accessible to anybody.

AMeta search of peers is launched rather than searching a main database by the search engines. This problem is circumvented by the hybrid Peer-to-Peer architecture.

Administration is difficult as there is no centralized server.

2.3.1.4. Client-Server Architecture

A Client-Server architecturenetwork[4] is a distributed network which consists of one higher performance system, the server, and multiple lower performance systems, the clients. The server is the central processing unit as well as the only source of data and services. A client only requests content or the implementation of services, without giving out any of its own resources. The Client-Server architecture is a system in which each computer on the network is either a client or a server. Servers are controlling computers devoted to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are workstations on which users execute applications. Clients rely on servers for resources.

The most commonly used paradigm in developing distributed systems is the Client-Server model. The client and server involve a known set of conventions before they start communicating. This set of conventions contains a protocol, which must be implemented at both ends of the connection. Figure 2-4 is a visual comparison between the flat client-server and hierarchical client-server architecture.

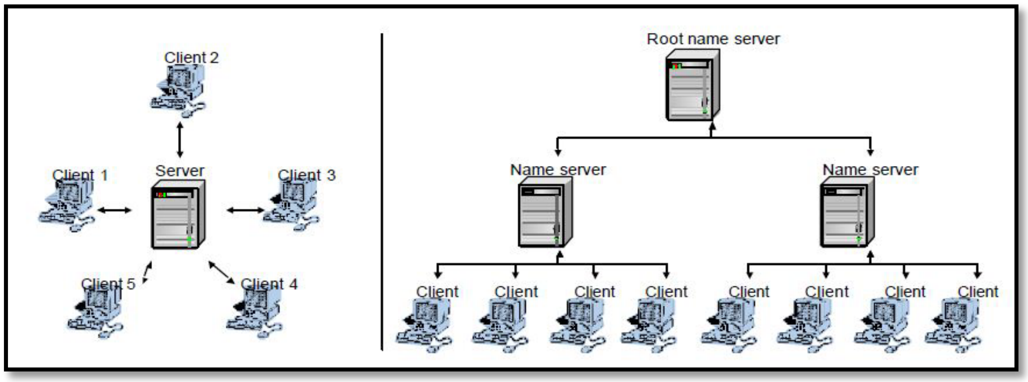


Figure 2-4Flat client-server architecture vsHierarchical Client-server architecture

2.3.1.5. Advantages of client-server architecture

The main advantage of this model is that the database is at one location allowing fast backups and proficient error management. The users can be prevented from damaging the files by the multiple layers of permissions required. The client only receives the end results as the processing is done at the back end only. This reduces the amount of network traffic amid the server and the client, enhancing network stability.

Thin client architectures allow a quick replacement of defect clients, because all data and applications are on the server.

2.3.1.6. Disadvantages of client-server architecture

Client-Server systems require a lot of maintenance and expenditure.

This architecture is dependent upon the server making it the central point of the system. If the server fails down, the whole system will suffer from delays as well long term break down, which can potentially block hundreds of clients from working with their data or their applications.

2.4. Database Server

The comprehensive database will be located at the server side of the network. The server is interfaced with the controller over the company intranet. The database server will be responsible for data acquisition, pre-processing, management, post processing and filtering. The raw data will be collected at the controller and will be passed to the server. From scalable perspective, there can be hundreds of controllers involved in the system.The efficient handling of database becomes a key issue to provide accuracy and scalability.

Data management will be the acquisition of environmental variables and then the management of the data from the sensors at remote locations in an efficientmethod.

2.4.1. Database Server Functionality

The Database Server performs the functions of collection of data from the controller, pre-processing of the received data, filtering according to some pre-defined requirements, logging the database with a front-end GUI, generation of alarms as per a defined threshold and sending the alarm information to the reporting system

2.4.2. Database Server Options

The following options can be used as databases in our project:MySQL database, Oracle database and Excel sheet.

2.4.2.1. MySQL Database

MySQL Database is a softwarewritten in C++ and C, which is tested with a broad range of compilers. APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available. It is multi-threaded using kernel threads. It is capable of using multiple CPUs. It provides transactional and non-transactional storage engines, uses very fast B-tree disk tables (MyISAM) with index compression and has a very fast thread-based memory allocation system. SQL uses high level class libraries in order to implement functions. Usually after the execution of inquiry, there is no memory requirement for storage. The MySQL code is tested with Purify (a commercial memory leakage detector) as well as with Valgrind, a GPL tool. The server is available as a separate program for use in a client/server networked environment. It is also available as a library that can be embedded (linked) into standalone applications. This is helpful in isolated or network-free environments.

MySQL Database isa very flexible and secure that allows host-based verification based onprivilege and password system. The password traffic is encrypted over the network providing password security. Itcan handle large databases, with numerous indexes per table. Each index may consist of 1 to 16 columns or parts of columns. The maximum index width is 1000 bytes. An index may use a prefix of a column for CHAR, VARCHAR, BLOB, or TEXT column types. Clients can connect to the MySQL server using TCP/IP sockets on any platform. On Windows systems in the NT family (NT, 2000, XP, or 2003), clients can connect using named pipes. On UNIX systems, clients can connect using UNIX domain socket files.

2.4.2.2. Oracle Database

The Oracle Database (which is commonly referred as Oracle RDBMS or simply as Oracle) is an object-relational database management system (ORDBMS) produced and marketed by Oracle Corporation.

The Oracle RDBMS stores data logically in the form of table spaces and physically in the form of data files.Tablespaces can contain various types of memory segments, such as Data Segments, Index Segments, etc. Segments in turn comprise one or more extents. Extents comprise groups of contiguous data blocks. The newer version also features a partitioning feature, which allows tables to be partitioned based on different set of keys.

Large data sets can be then easily managed by specifically dropping or selecting a certain key.

Oracle database management uses the information stored in its SYSTEM table space to track its computer data storage. The SYSTEM tablespace contains the data dictionaryand often (by default) clusters and indexes. A data dictionary consists of a special collection of tables that contains information about all user-objects in the database.

2.4.2.3. Excel Sheets

Excel provides a number of menu commands and functions that permit to use a list as a database. Menu commands can be used to find records in the database that match criteria that are defined, or you can use worksheet functions to extract numerical information from a database.

To use Excel's database commands and functions, the list must meet the following requirements:The list must be in column format: each column in the list is a field in the database, each row in the list is a record in the database and the top row of the list must contain the field names, which you will use to identify the information stored in each field.

The Excel will recognize any field as database if the curser is place anywhere on the sheet. By renaming it to database willautomatically permit Excel to detect the database. Another method can be giving reference to the cell ranges. To assign the range name Database to the list, select the entire range of cells in the list, including the field names. Then choose Set Database from the Data menu (available in Excel 97 only). Excel assigns the name Database to the selected range. One can also define a database by assigning the name Database to the selected range of cells by using Define Name.

2.5. Front-end Network Management System

The processed database is linked to the front end Graphical User Interface of Network Management System. For scalability and flexibility, the designing of the GUI is the most challenging task. The GUI will be able to display the status of entire network duly segregate able for location based view or equipment based view.

The GUI is capable of showing the following features:Real-time data updates, Real-time faults, updates, Data history and log files, Alarm indication, History of alarm generations, Location-wise categorizing remote sites, Sensor-wise categorizing remote sites i.e. w.r.t. the sensor values at various locations, To provide distributed control and visibility of the network, client servers will be developed. The client side Network Management System would be having the same functionalities as the Front-end GUI. However its functionalities would be limited and they would be in a controlled environment.

2.6. Reporting System

The database at the server will be responsible for determining if the processed data exceeds the threshold value or not or does it require any further action. If the value does exceed, the database updates the reporting module as part of the reporting system. The reporting module will be programmed to generate alarms based on certain defined thresholds. The alarms generated will be reported to the concerned authorities. According to predefined expected response, reporting system will efficiently raise the level (if required) or perform alternate action as defined.

CHAPTER 3

DESIGN AND DEVELPOMENT

The idea of this project is based upon the client server model to aid the development of a Centralized Environment Network Monitoring and Reporting System.

Design of the project consists of monitoring equipment at the remote site, which includes sensors and a controller for acquiring data and sending it to the server. The server side comprises of a database and website. The sensors at the remote site continuously monitor the ambient conditions of the site. While the database and website provides easy access of the status of the sensors and prevailing conditions at the remote site.

For prototype development, four sensors were used and first aim was to interface the sensors to the controller. The controller takes the reading from the sensors converts analog values to digital ones and then formulates a message including the reading sensor ID and sensor location. This message is then transmitted to the server over the WAN network.

At the Server end, data received would be recorded in a database and updated to a website and this website can be accessed by the authorized personnel. Any values exceedingset thresholds would generate alarms on the website and would be reported to concernedauthorities via SMS using a GSM modem incorporated at the server side.

Progress is made in individual modules of the project, depicted in Figure 3-1, and explained the modules as the Readings from the sensors employed, Controller board, Client server connection, Database Management System, SMS using GSM modem and Website.

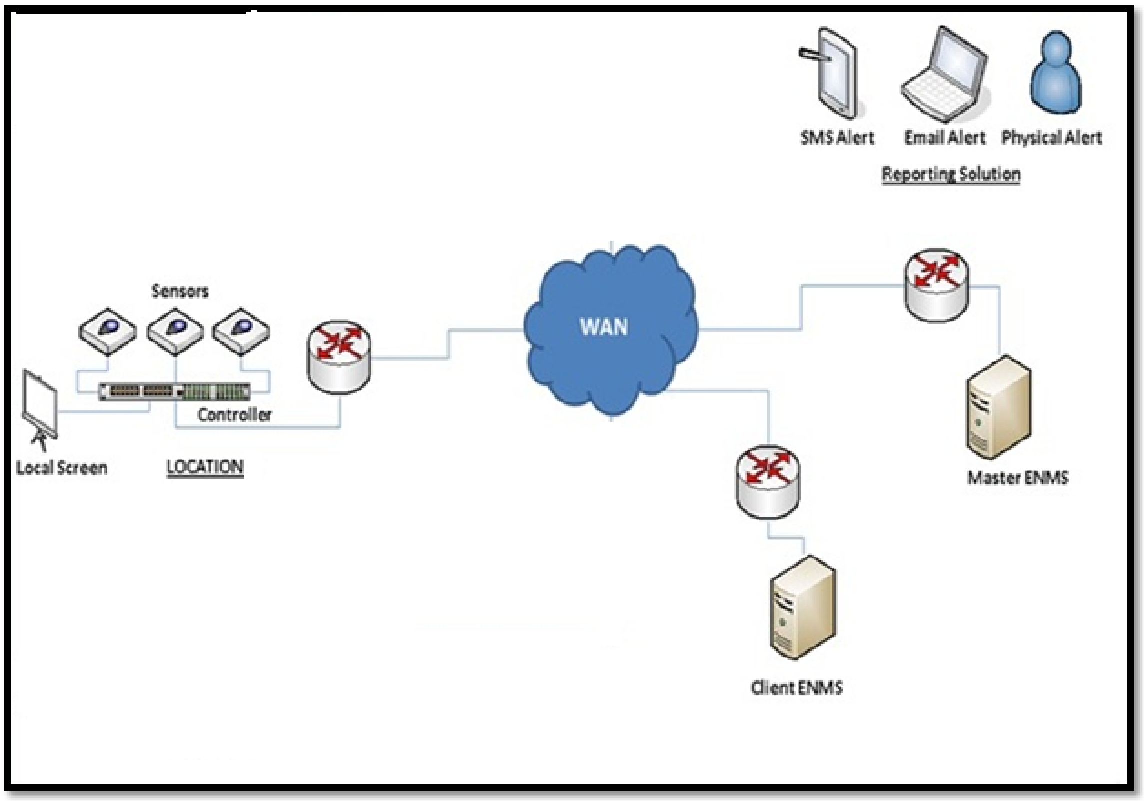


Figure 3-1 Modular Illustration of the Prototype

3.1. Readings from the sensors employed

The sensors being used aresmoke sensor, humidity sensor, temperature sensor and door sensor.Smoke sensor used is MQ-2. This flammable gas and smoke sensor detects the concentrations of combustible gas in the air and outputs its reading as an analog voltage. The sensor can measure concentrations of flammable gas of 300 to 10,000 ppm. Analog output from the sensor is converted to digital form ‘high’ and ‘low’ using the IC 1013 . Whenever slight amount of smoke is present in the air it is detected by the sensor resulting in ‘high’ voltage. This output is fed to the controller via wired connection

Humidity sensor used is HSU-041. It detects the relative humidity of the immediate environments in which it is placed. It measures both the temperature and moisture in the air and expresses relative humidity as a percentage of the ratio of moisture in the air to the maximum amount that can be held in the air at the current temperature. The sensor gives analog output , using the voltage to humidity curve given in the datasheet of HSU-041A. Equation 4.1 is the formula which was determined through Matlab software.

$$\textit{Humidity value}=((\textit{Vout}*0.00018148)-0.16)*161.29 \qquad (3.1)$$

This formula is used in the C code, run on the controller, to convert voltages output from ADC converter to humidity values.

Temperature sensor used is LM35DZ. This sensor also gives analog output. The output voltage is proportional to the Celsius temperature. Equation 3.2 is the conversion formula for obtaining temperature values.

$$\textit{Temperature}\ (^{o}C) = \textit{Vout} * (100\ ^{o}C/V) \qquad (3.2)$$

Output voltage from ADC converter is converted to temperature value. As in the case of the humidity sensor this conversion formula is also a part of the C code, run on the controller

A switch is being used as a door sensor. It changes its output to either ‘high’ or ‘low’ according to the state of the switch .It is implemented on the unhinged side of the door.

When the door is closed the switch is in on, the button being in pressed sate, the circuit is complete hence ‘high’ output. When the door is open the switch is in off state resulting in an incomplete circuit, hence, no output. Output from this sensor is fed to the controller. Figure 3-2 is the sensor connections with the ARM9 board as done in the project.

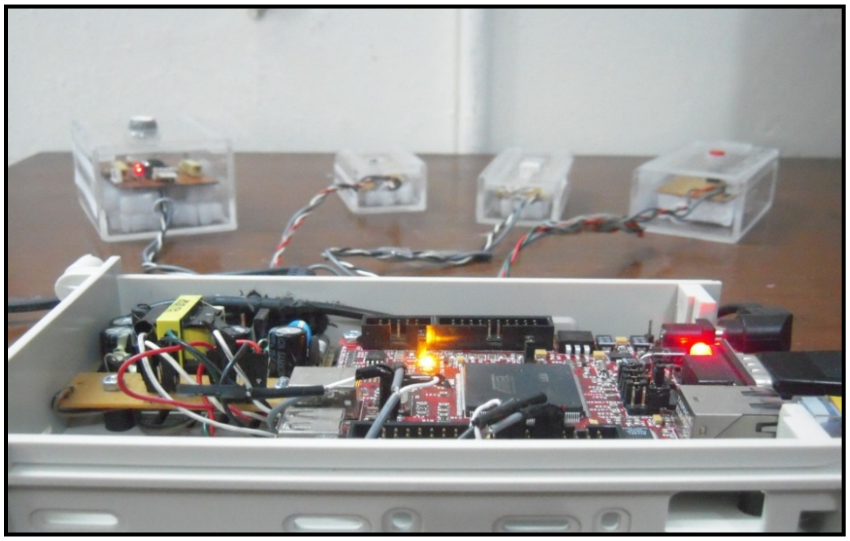


Figure 3-2 Controller board connected with the Sensors

3.2. Controller board

The controller board selected for the project is Olimex SAM9-L9260 development board. Figure 3-3 is the detailed view of the ARM9 controller board.

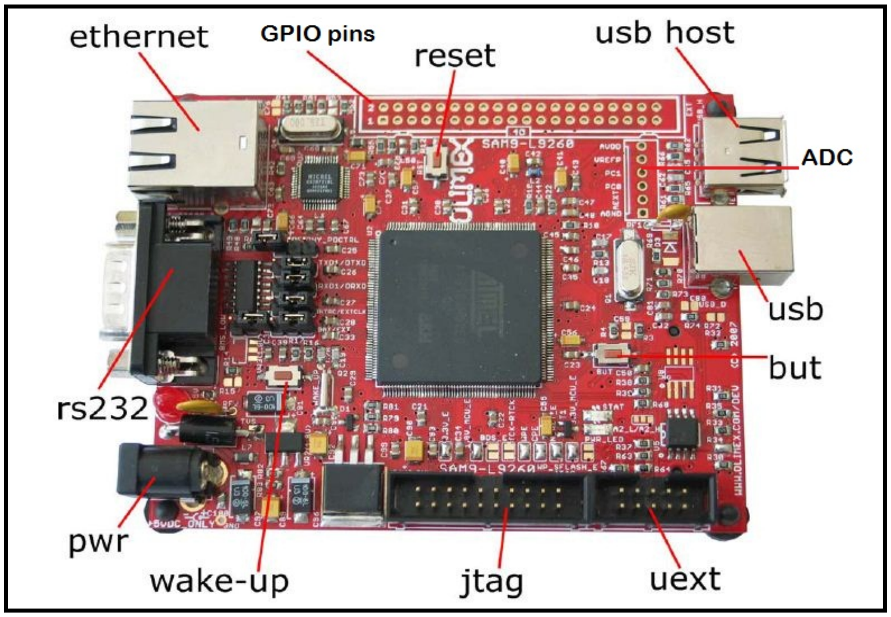
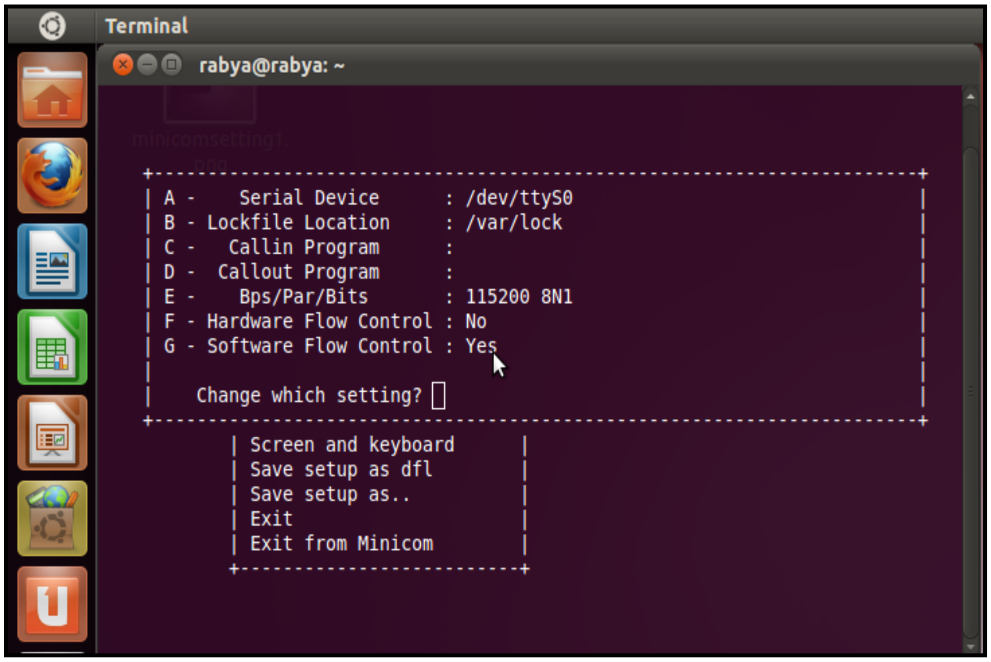


Figure 3-3 SAM9-L9260 Development board

3.2.1. Operating the controller board

The RS-232 serial port is used to interface the board with the laptop. The board is accessed through minicom which is similar to terminal in Linux environment. The serial port settings done on minicom for communication with the board are as follows:Bps/par/bits: 115200 8N1, Hardware flow control: No and Software flow control: Yes.

All the C codes to be run on the SAM9 board were originally tested through terminal in Ubuntu and later transferred to the development board using USB device. After mounting the USB device code was copied to the /home directory of the board. Using the gcc-arm-linux-gneaubi compiler code was compiled and the executable file was then run. Figure 3-4 displays the minicom settings for communicating with the ARM9 board.



```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyS0                                 |
| B - Lockfile Location     : /var/lock                                  |
| C -   Callin Program      :                                            |
| D -  Callout Program      :                                            |
| E -    Bps/Par/Bits       : 115200 8N1                                 |
| F - Hardware Flow Control : No                                         |
| G - Software Flow Control : Yes                                        |
|                                                                        |
|    Change which setting?                                               |
+-----------------------------------------------------------------------+
        | Screen and keyboard       |
        | Save setup as dfl         |
        | Save setup as..           |
        | Exit                      |
        | Exit from Minicom         |
        +---------------------------+
```

Figure 3-4Minicom settings for communication with controller

3.2.2. Interfacing sensors with the controller

The readings from the sensors are provided to the controller through wired connection. The sensors are connected to the programmable pins of the SAM9 board and the ADC Pins. The smoke sensor output pin and ground pin is connected to the PC6 and PC7 pins of the EXT connector on the board. The door sensor output pin is connected to the PC9 and PC10 pins of the board. The humidity sensor output pin is connected to the AD0 pin of the board and the temperature sensor output pin is connected to the AD1 pin of the board.The I/O pins are programmed such that whenever any voltage is output it is translated as ‘open’ in case of door sensor and ‘smoke’ in case of smoke sensor[9]. ADC pins are programmed to convert the analog output from the temperature and humidity sensor to the respective temperature and humidity values. Figure 3-5 is the detailed view of the pins of the SAM9L9260 microcontroller embedded on the ARM9 board.

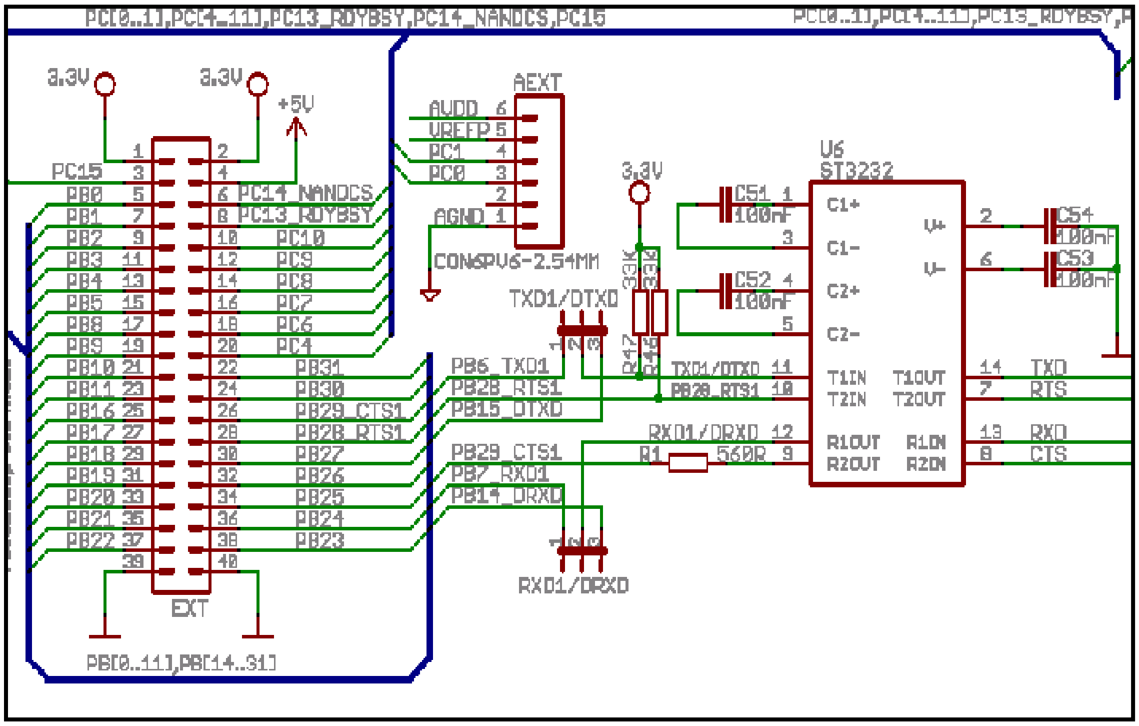
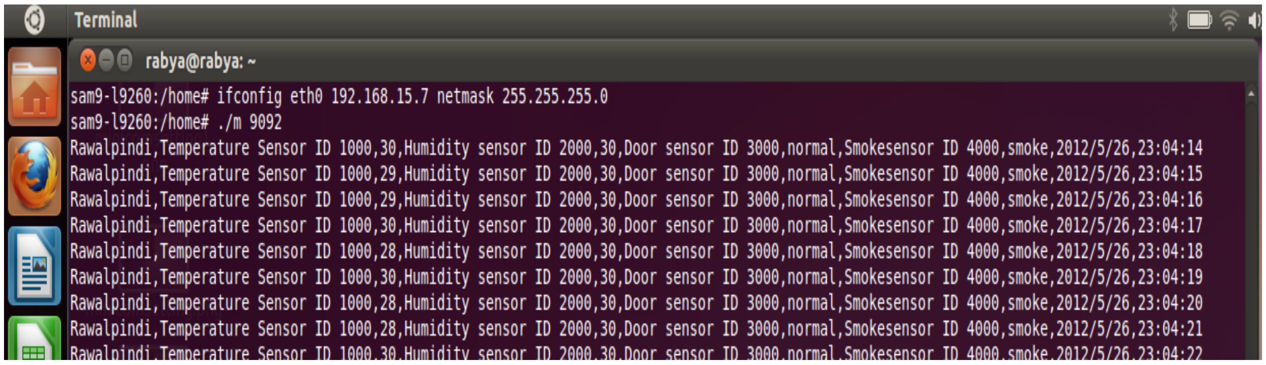


Figure 3-5 EXT and ADC connector for Sensor connection

3.2.3. Processing on the controller board

The code written runs on the controller board converts analog data from sensors todigital and then to humidity and temperature values .It also adds location time and dateinformation to the values continuously coming from the sensors. In the case of door sensor the ‘high’ and ‘low’ output is translated to ‘normal’ and ‘open’ state respectively. For smoke sensor ‘high’ output is translated as ‘smoke’ and no output is translated as ‘normal’. Sensor ID for each sensor is also added to the data through the C code. This data in Figure 3-6 is sent line by line to the database using the client server model. All data is sent to the database with a one second delay.



```
sam9-l9260:/home# ifconfig eth0 192.168.15.7 netmask 255.255.255.0
sam9-l9260:/home# ./m 9092
Rawalpindi,Temperature Sensor ID 1000,30,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:14
Rawalpindi,Temperature Sensor ID 1000,29,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:15
Rawalpindi,Temperature Sensor ID 1000,29,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:16
Rawalpindi,Temperature Sensor ID 1000,30,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:17
Rawalpindi,Temperature Sensor ID 1000,28,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:18
Rawalpindi,Temperature Sensor ID 1000,30,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:19
Rawalpindi,Temperature Sensor ID 1000,28,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:20
Rawalpindi,Temperature Sensor ID 1000,28,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:21
Rawalpindi,Temperature Sensor ID 1000,30,Humidity sensor ID 2000,30,Door sensor ID 3000,normal,Smokesensor ID 4000,smoke,2012/5/26,23:04:22
```

Figure 3-6 Processed Data from the Sensors

3.3. Client Server connection

The transmission of processed data from the controller board to the database server is done using the client server model. Socket programming is done to open ports on both the client and server side and for communication between the two. Server for our model is the system where the database is maintained. The controller board is the client for the model as it initiates the communication process. UDP is used as the underlying transport protocol for communication. UDP is preferred for real time applications as there is no overhead processing or delay due to missed packets. The socket programming client code is run on the system where database is maintained and server code is run on the controller board. IPs used to communicate between the client and server are 192.168.15.1 and 192.168.15.7 respectively.

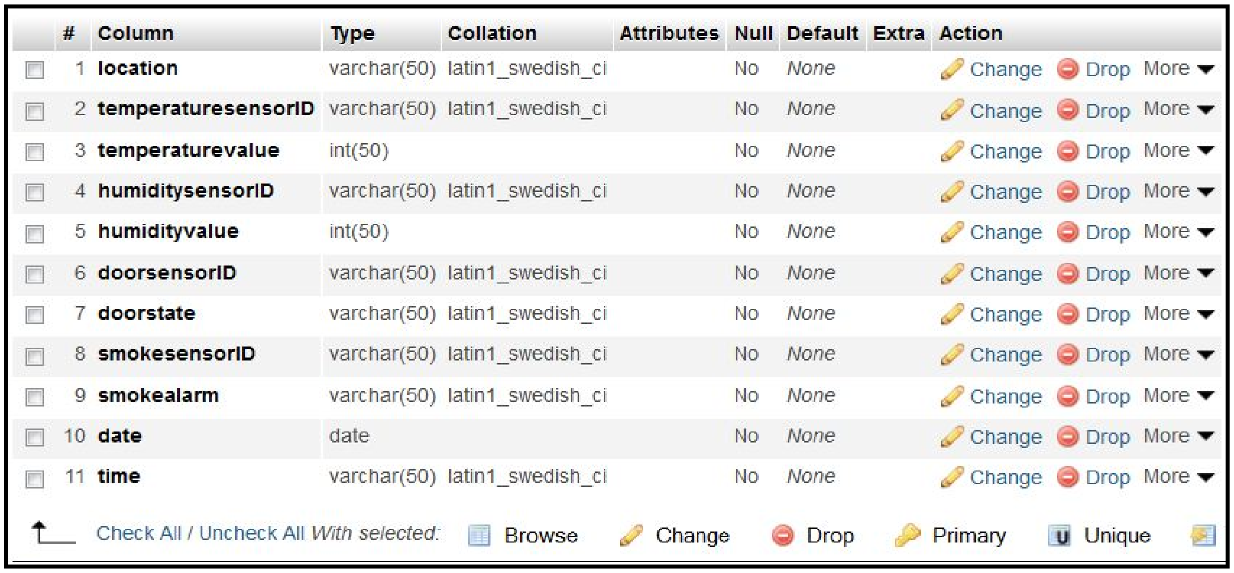
3.4. Receiving System at the Server

Data received from the controller board is stored in a folder specified in the code(C directory is used for this purpose). Files are made in the folder with an interval of one second which stores readings from the sensor. A windows form application in Visual

Studio 2010 C# has been developed for the project. The code runs such that data from these files is uploaded to MySQL database. The content of the file that is the sensor readings and the sender ID the time and date is sent and all saved in the database.

3.4.1 MySQL database

Figure 3-7 is the database made in Mysql server , it has the following columns Location, Temperature sensor ID, Temperature sensor value, Humidity sensor ID, Humidity sensor value, Door sensor ID, Door state, Smoke sensor value, Smoke alarm, date and time.



#	Column	Type	Collation	Attributes	Null	Default	Extra	Action
1	location	varchar(50)	latin1_swedish_ci		No	None		Change Drop More
2	temperaturesensorID	varchar(50)	latin1_swedish_ci		No	None		Change Drop More
3	temperaturevalue	int(50)			No	None		Change Drop More
4	humiditysensorID	varchar(50)	latin1_swedish_ci		No	None		Change Drop More
5	humidityvalue	int(50)			No	None		Change Drop More
6	doorsensorID	varchar(50)	latin1_swedish_ci		No	None		Change Drop More
7	doorstate	varchar(50)	latin1_swedish_ci		No	None		Change Drop More
8	smokesensorID	varchar(50)	latin1_swedish_ci		No	None		Change Drop More
9	smokealarm	varchar(50)	latin1_swedish_ci		No	None		Change Drop More
10	date	date			No	None		Change Drop More
11	time	varchar(50)	latin1_swedish_ci		No	None		Change Drop More

Figure 3-7Database in MySQL(wamp server)

3.5Sending SMS from the GSM modem

In the C# application thresholds were defined for each environmental parameter being monitored. [5] For temperature a threshold of 50 °C is set, for humidity 45%, for door ‘open’ state and for smoke sensor ‘smoke’ state. The thresholds for temperature and humidity were selected according to the prevailing conditions. These can be varied according to the unmanned site requirements. Emant SMS library has been used for sending SMS through the GSM modem. The modem is connected to COM port 4 of the laptop/PC where windows application is running and database is maintained. A fixed number is written in the code to which SMS are sent when thresholds are crossed. When temperature exceeds 50 °C an SMS is sent to the number specified in the code. The SMS message to be sent is also specified in the code for temperature it is 'temperature alert'. When humidity exceeds 45% an SMS is sent to the same number. The SMS message is 'humidity alert'. In case of 'smoke' a 'smoke alert' SMS is sent and in case of door state 'open' an SMS message 'intrusion alert' is sent to the same number. The number specified in the code can be changed and SMS can also be sent to multiple operators.

3.6. Graphical User Interface Development

The MySQL database is used to store data in passive form. In order to attain useful and illustrative information from it, a website was developed. The purpose of the website is to generate the graphs for different sensors, to view sensor state with time and to show alarms generated when thresholds are exceeded. The content development was done in php scripting language, which gets, inserts or updates data into the MySQL database through SQL queries.[5]

The Login Page

Figure 3-8 is the login page. For verification of the user, username and password is entered here. Correct entry opens the website.

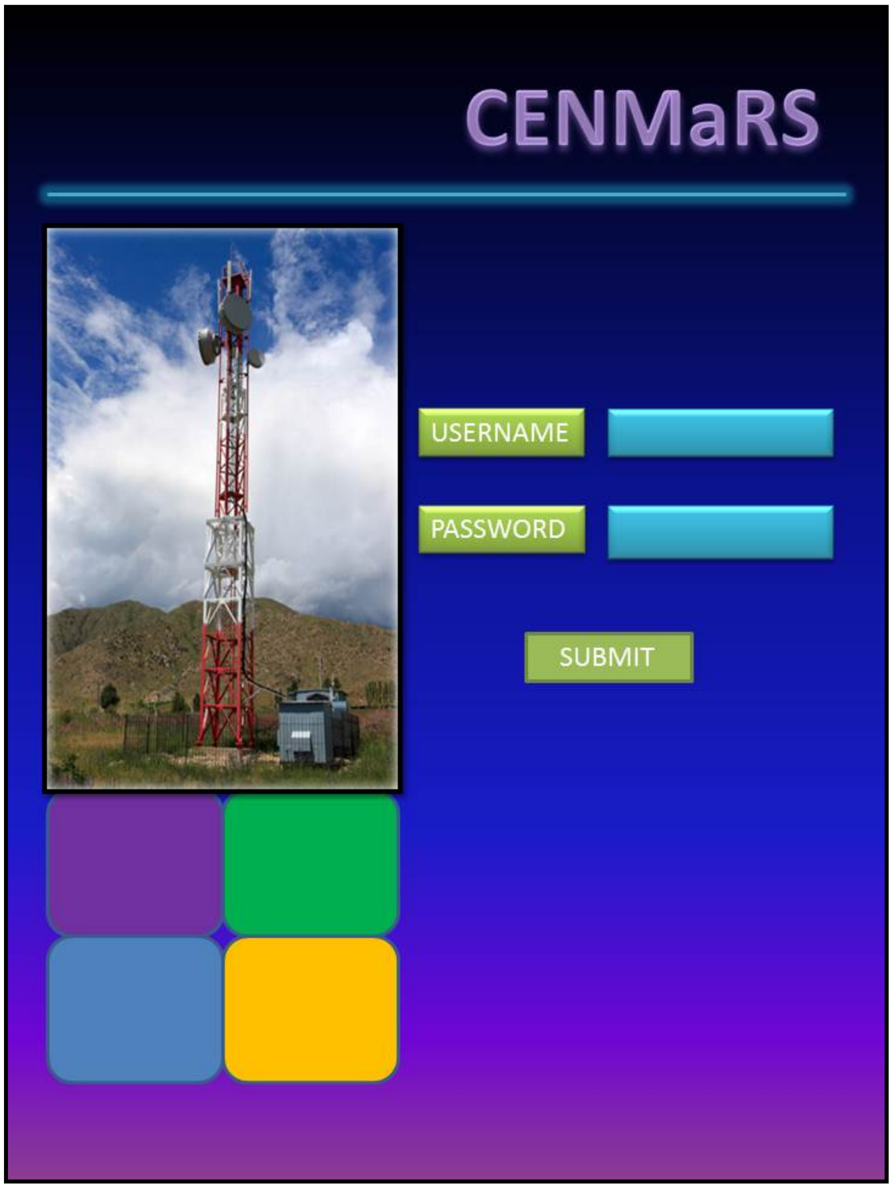


Figure 3-8 Login Page

The administrator logs into the website through the "**admin**" account. When the username and password is entered, a SELECT query is run at the back end. All data from the “users” table is retrieved, and the entered username and password is matched. If either username and/or password is incorrect, login attempt is rendered unsuccessful and an error is returned.

The initial connection with the database is established using the **mysql_connect**command in PHP.

```
mysql_connect("$sql_host","$sql_username","$sql_pass")
```

The hostname, username and password for the database are specified in the mysql_connect command. The PHP web pages were tested using local host.

CENMaRS Home page

Figure 3-9 is the home page as seen after logging in**.** The homepagedisplays general information about the sensorlocation, type of sensors and alarm summary information .From the homepage, there is the option for navigating to the data page, particular sensor information, log files, graph for the sensor data. The Location chart shows the location of the sensors employed in various cities. Any alarm generated at a specific location is indicated on the map with a change in color. Equipment chart shows the details of sensors employed at each location , the website developed is such that each location has four sensors. The alarm summary in the left column shows the number of alarms generated according to priority. Smoke has been given highest priority then intrusion alarm and temperature alarm. The sensor buttons are hyperlinked to the pages where sensor details can be viewed.

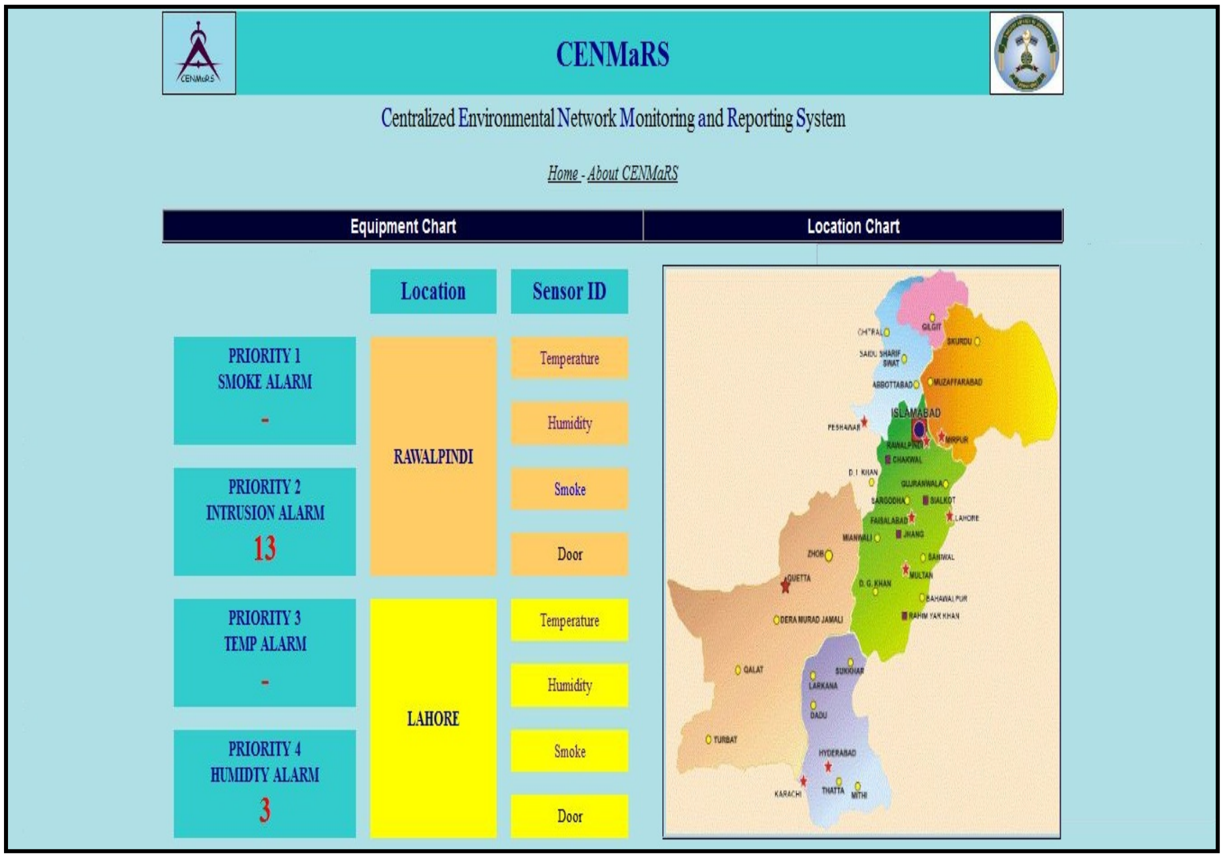
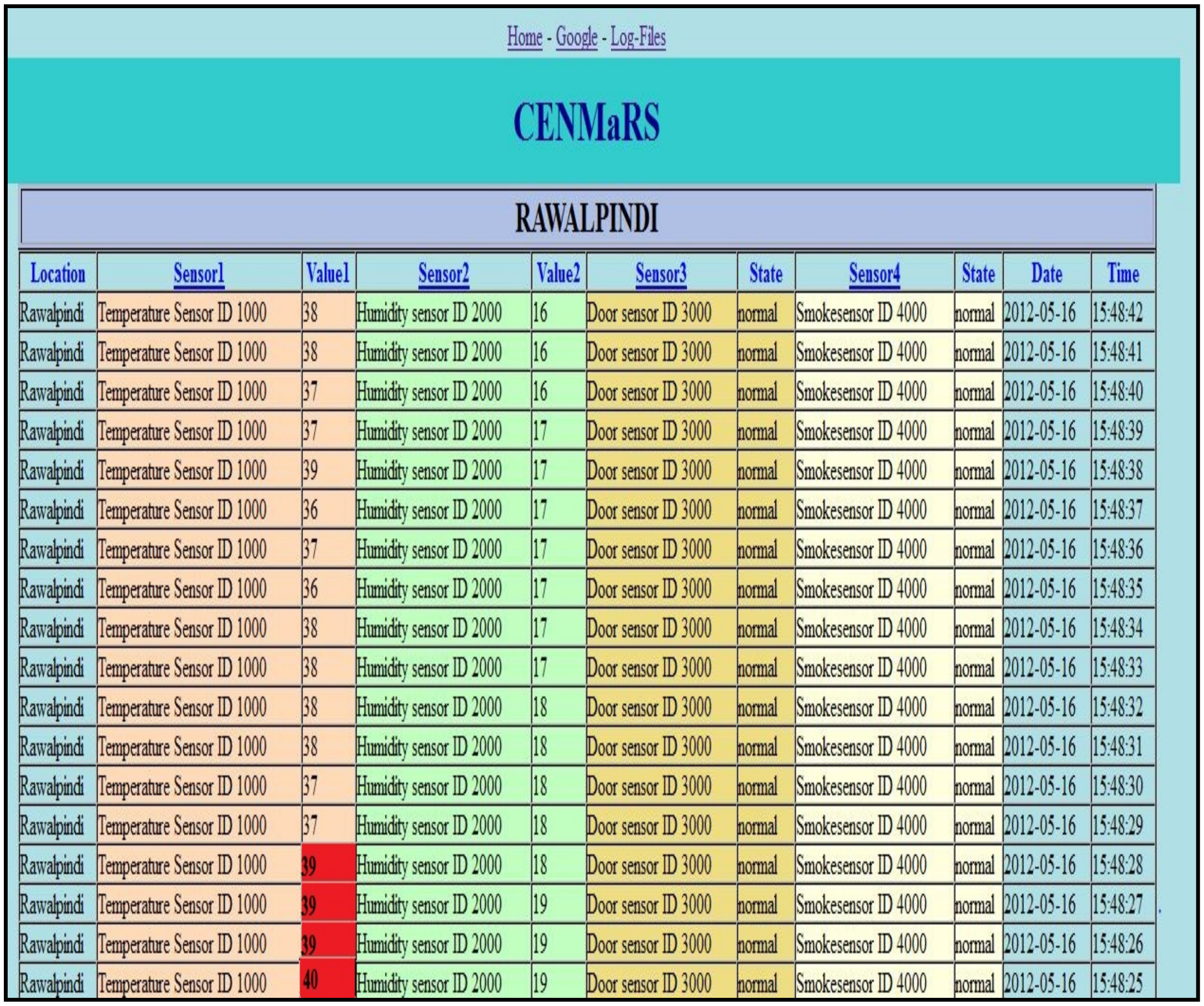


Figure 3-9 Home Page Viewed on Login

Readings of Sensors at Site Rawalpindi

Figure 3-10 is all the data from the four sensors from a particular site. This page displays the Site location, sensor ID which is specific for every sensor and was defined earlier, date and time of each reading is also displayed. If any environmental variable exceeds the set threshold then it is highlighted making it clearly visible. The sensor ID differentiates similar types of sensors within a particular location. Another page displays the sensor IDs with the corresponding sensor type for ease of the user.



Home - Google - Log-Files

CENMaRS

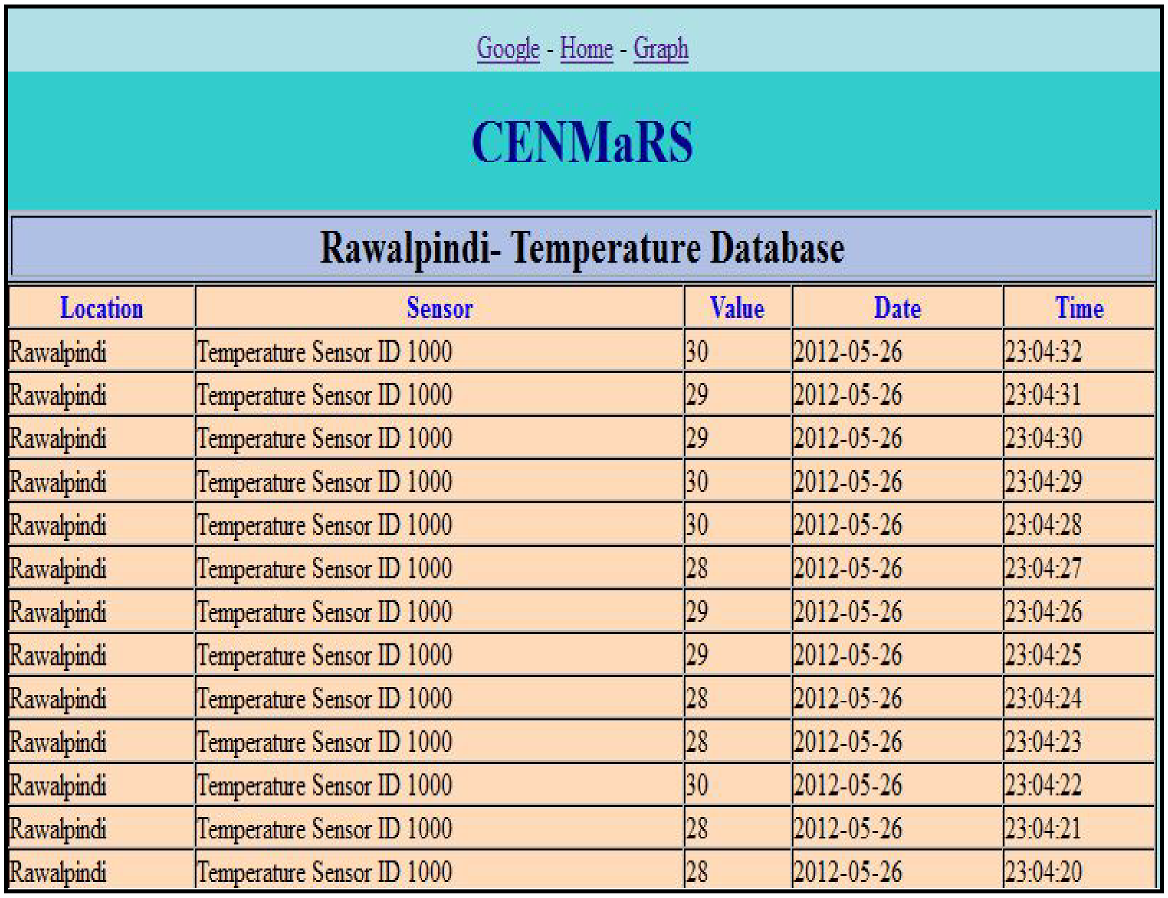
RAWALPINDI

Location	Sensor1	Value1	Sensor2	Value2	Sensor3	State	Sensor4	State	Date	Time
Rawalpindi	Temperature Sensor ID 1000	38	Humidity sensor ID 2000	16	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:42
Rawalpindi	Temperature Sensor ID 1000	38	Humidity sensor ID 2000	16	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:41
Rawalpindi	Temperature Sensor ID 1000	37	Humidity sensor ID 2000	16	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:40
Rawalpindi	Temperature Sensor ID 1000	37	Humidity sensor ID 2000	17	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:39
Rawalpindi	Temperature Sensor ID 1000	39	Humidity sensor ID 2000	17	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:38
Rawalpindi	Temperature Sensor ID 1000	36	Humidity sensor ID 2000	17	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:37
Rawalpindi	Temperature Sensor ID 1000	37	Humidity sensor ID 2000	17	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:36
Rawalpindi	Temperature Sensor ID 1000	36	Humidity sensor ID 2000	17	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:35
Rawalpindi	Temperature Sensor ID 1000	38	Humidity sensor ID 2000	17	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:34
Rawalpindi	Temperature Sensor ID 1000	38	Humidity sensor ID 2000	17	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:33
Rawalpindi	Temperature Sensor ID 1000	38	Humidity sensor ID 2000	18	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:32
Rawalpindi	Temperature Sensor ID 1000	38	Humidity sensor ID 2000	18	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:31
Rawalpindi	Temperature Sensor ID 1000	37	Humidity sensor ID 2000	18	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:30
Rawalpindi	Temperature Sensor ID 1000	37	Humidity sensor ID 2000	18	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:29
Rawalpindi	Temperature Sensor ID 1000	39	Humidity sensor ID 2000	18	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:28
Rawalpindi	Temperature Sensor ID 1000	39	Humidity sensor ID 2000	19	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:27
Rawalpindi	Temperature Sensor ID 1000	39	Humidity sensor ID 2000	19	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:26
Rawalpindi	Temperature Sensor ID 1000	40	Humidity sensor ID 2000	19	Door sensor ID 3000	normal	Smokesensor ID 4000	normal	2012-05-16	15:48:25

Figure 3-10 Real-Time Sensors Reading from a Single Site

Sensor Data

The temperature sensor data page Figure 3-11 only shows the temperature values along with the location and sensor ID. Similar pages have been made for Humidity sensor, Door sensor and Smoke sensor. Purpose of these pages is to view a specific parameter at a particular site. Date and time with each reading makes it easy to track when a certain parameter exceeds threshold.


Google - Home - Graph

CENMaRS

Rawalpindi- Temperature Database

Location	Sensor	Value	Date	Time
Rawalpindi	Temperature Sensor ID 1000	30	2012-05-26	23:04:32
Rawalpindi	Temperature Sensor ID 1000	29	2012-05-26	23:04:31
Rawalpindi	Temperature Sensor ID 1000	29	2012-05-26	23:04:30
Rawalpindi	Temperature Sensor ID 1000	30	2012-05-26	23:04:29
Rawalpindi	Temperature Sensor ID 1000	30	2012-05-26	23:04:28
Rawalpindi	Temperature Sensor ID 1000	28	2012-05-26	23:04:27
Rawalpindi	Temperature Sensor ID 1000	29	2012-05-26	23:04:26
Rawalpindi	Temperature Sensor ID 1000	29	2012-05-26	23:04:25
Rawalpindi	Temperature Sensor ID 1000	28	2012-05-26	23:04:24
Rawalpindi	Temperature Sensor ID 1000	28	2012-05-26	23:04:23
Rawalpindi	Temperature Sensor ID 1000	30	2012-05-26	23:04:22
Rawalpindi	Temperature Sensor ID 1000	28	2012-05-26	23:04:21
Rawalpindi	Temperature Sensor ID 1000	28	2012-05-26	23:04:20

Figure 3-11.Real-time Temperature Sensor Readings

Real-time Graph

There is also the option to view graphs for the different parameters. Graphical representation makes changes in any of the environmental variables quickly noticeable. The website developed shows real time graphs for all four sensors. Figure3-12 is a display of the real time graph of the temperature sensor reading. The purpose of developing this page is quick viewing of prevailing conditions at a particular site for swift counteracton.

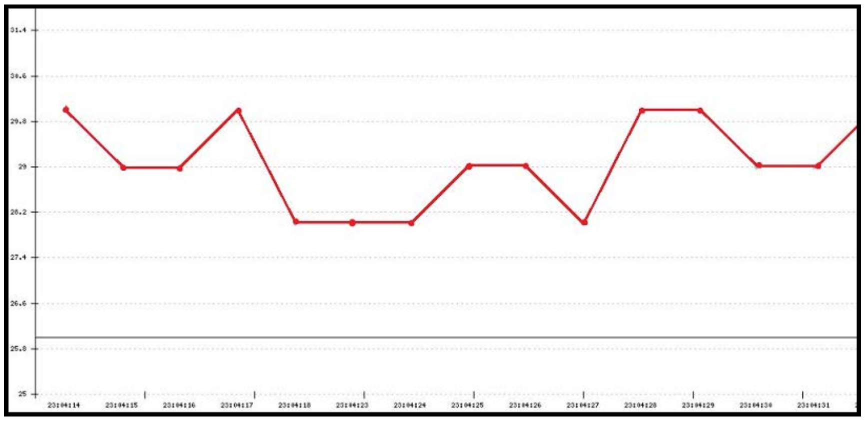
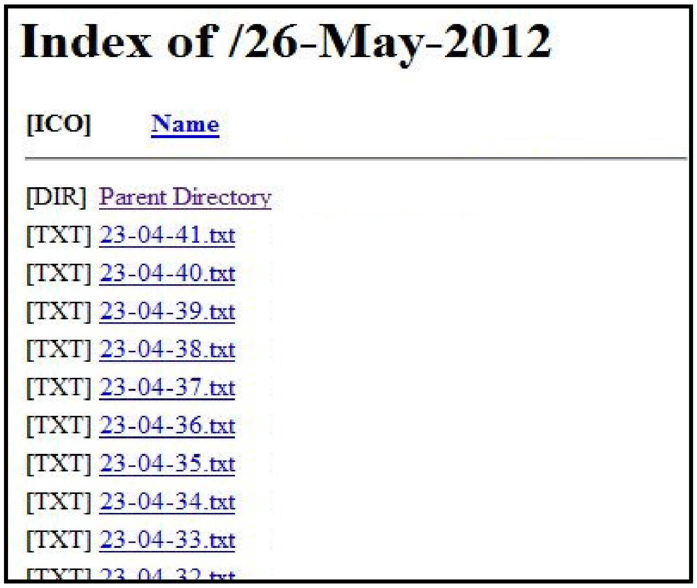


Figure 3-12Temperature Sensor reading vs Time Graph

Log Files

The log files page displays the log files according to the date. In Figure3-13 log files are being made for every second .This is for the ease of the administrator so that any log file can be accessed to check the fault that occurred at any date and time. This is how history of the site is maintained.



Index of /26-May-2012

[ICO] Name

[DIR] Parent Directory
[TXT] 23-04-41.txt
[TXT] 23-04-40.txt
[TXT] 23-04-39.txt
[TXT] 23-04-38.txt
[TXT] 23-04-37.txt
[TXT] 23-04-36.txt
[TXT] 23-04-35.txt
[TXT] 23-04-34.txt
[TXT] 23-04-33.txt
[TXT] 23-04-32.txt

Figure 3-13Log File

CHAPTER 4

EQUIPMENT USED AND CIRCUIT MODIFICATIONS

4.1 Hardware

4.1.1 Sensors

Temperature sensor

Figure 4-1 is LM35DZ temperature sensor.It is a precision integrated-circuit temperaturesensor, whose output voltage is linearly proportional to the Celsius temperature. It iscalibrated directly in ° Celsius (Centigrade). It provides 0.5°C accuracy guaranteed at +25°C. This sensor is suitable for remote applications and operates from 4 to 30 Volts; it has less than 60 μA current drain and a low impedance output, 0.1 W for 1 mA load. It uses a supply of 5V. [10].

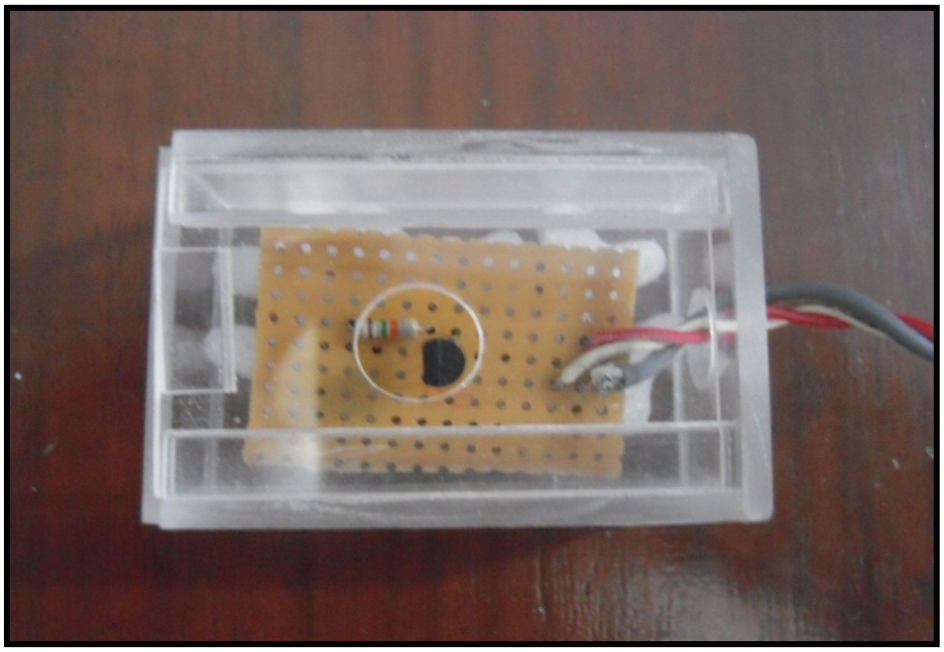


Figure 4-1 Temperature Sensor

Humidity sensor

Figure 4-2 is the humidity sensor HSU-04A1. It measures relative humidity (%RH) anddelivers it as an analog output voltage. It has a low power design; typical current draw ofonly 200μA .It hasenhanced accuracy and fast response time. Supply voltage is 5V. [8]

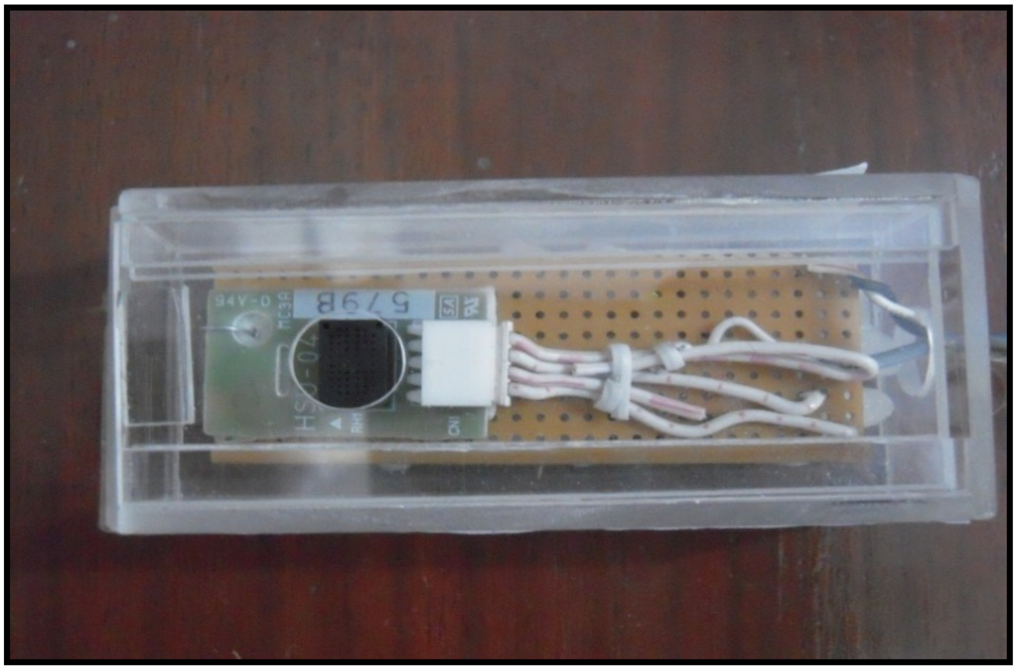


Figure 4-2 Humidity Sensor

Smoke Sensor

Figure 4-3 isthe smoke sensor MQ-2.The circuitry developed to make this sensor was obtained from the MQ-2 datasheet. When the internal heating element of the sensor is activated its resistance decreases in proportion to the amount of gas present in the air. [11].MQ-2 element was chosen due to its precision and high sensitivity. It is capable of detecting minute amounts of combustible gases in the environment including methane, smoke alcohol and gas.

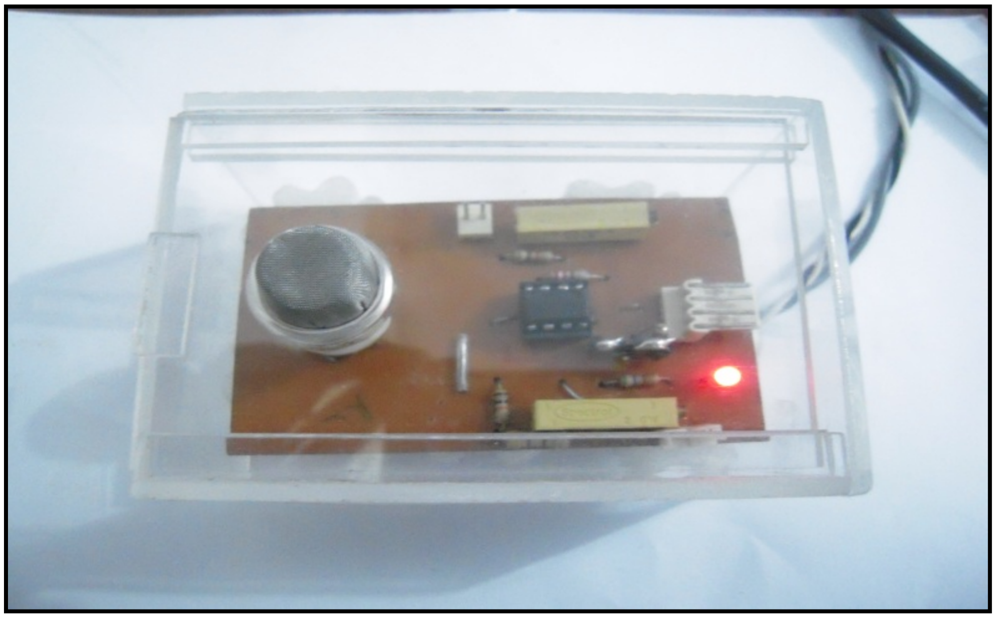


Figure 4-3 Smoke Sensor

Intrusion Sensor

Figure 4-4 is a simple touch switch circuit used for intrusion detection.The most common and simple example of a digital sensor is the touch switch. A typical touch switch is an open circuit with infinite resistance when it is not pressed and a short circuit with zero resistance when it is pressed. The door sensor is employed at the inside of the cabinet whenever the door opens the switch is no longer in pressed state causing the circuit to become an open circuit hence infinite resistance. This results in no output at the controller board input pin of the sensor.Low output state is translated as open through the programming done in C language.

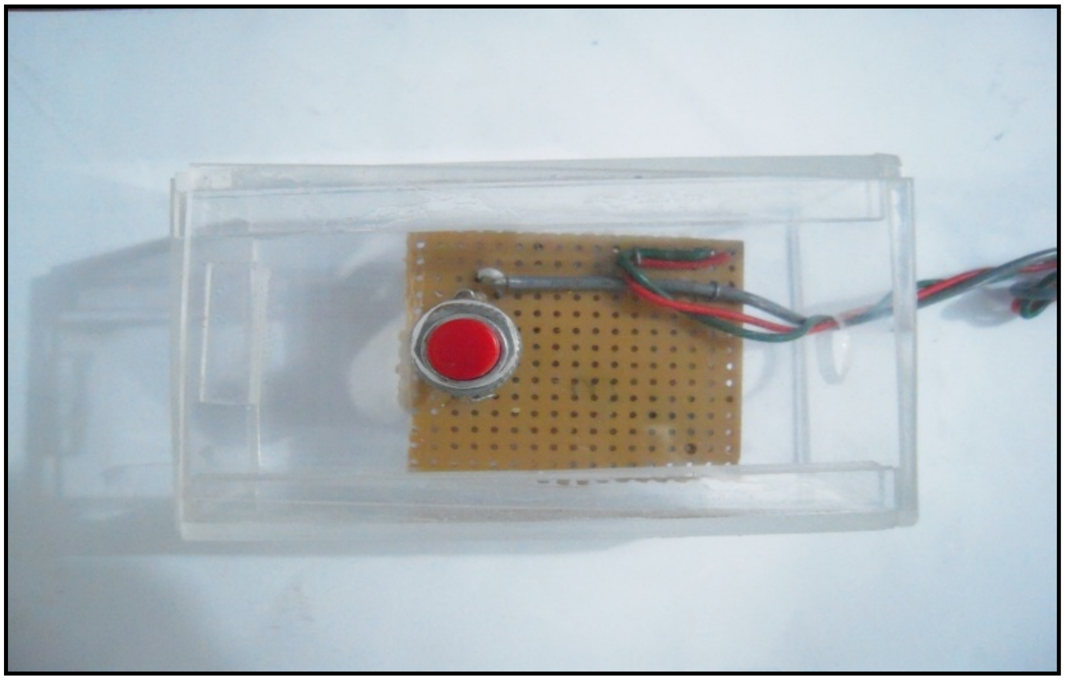


Figure 4-4 Intrusion Sensor

4.1.2 Controller

The controller board used in the project is Olimex SAM9-L9260 development board. It was selected due to its fast processing. It is a low cost development platform with ARM9 microcontroller, 64MB SDRAM and 512MB NAND Flash sufficient for storing data from the sensors. The requirement of wired connectivity between the controller, at the remote site and server is fulfilled by the on board Ethernet 100Mbit controller. The board also providesscalability for sensors as it has a 40 pin extension port with all unused SAM9260 ports available for add-on boards.One 4-channel 10-bit Analog-to-Digital Converter allows conversion of analog output from the sensors.96 Programmable I/O Lines Multiplexed with up to Two Peripheral I/Os support integration with multiple sensors. The controller board has pre-installed Linux which made running C codes on thedevelopment board easier.

4.1.3. USB GSM modem

Figure 4-5 is SIM900D GSM/GPRS wireless modem. It is a complete Quad-band GSM/GPRS module. It delivers GSM/GPRS 850/900/1800/1900MHz[7]. The modem is suitable for the application as it has low power consumption and a supply voltage of 5V.It is compatiblewith AT commands and supports PDU as well as Text SMS mode. The port settings done for the GSM modem are Bits per second 119200, databits 8, parity none, stop bits 1, flow control Xon/off[6].

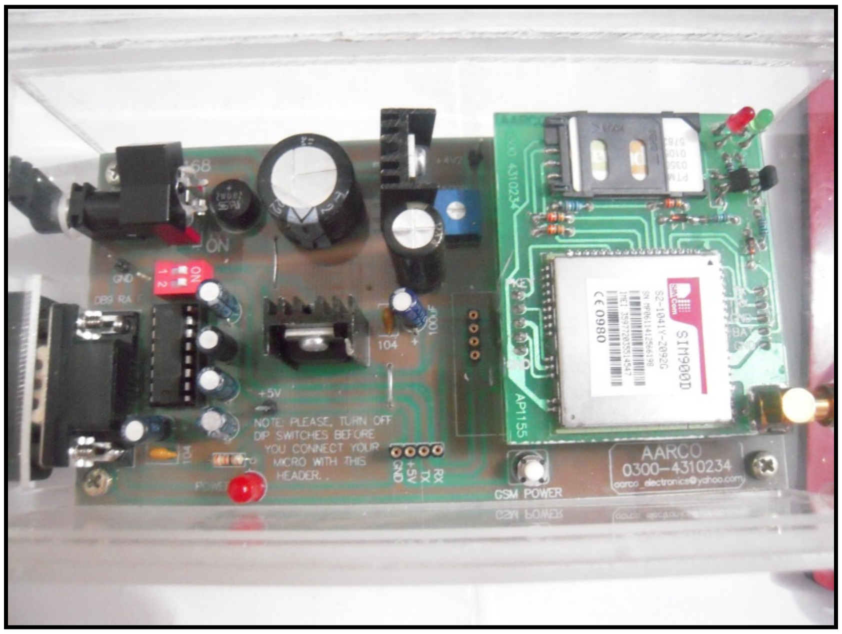


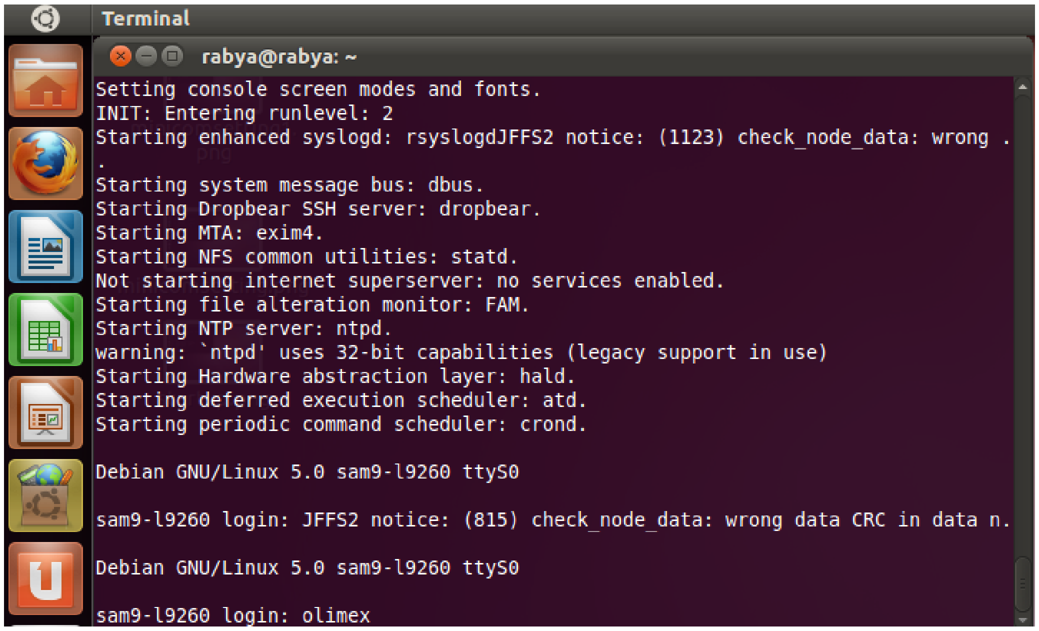
Figure 4-5GSM modem

4.2. Soft wares Used

Multiple softwares have been used in the project for accomplishment of various tasks.

4.2.1. Minicom

Figure 4-6 is Minicomwhich is used as a serial communication program. It is used under Linux/UNIX. It resembles DOS/Window TELIX. Minicom is a menu-driven communication program. It is a text based modem control and terminal emulation program for Unix-like operating systems.



```
Setting console screen modes and fonts.
INIT: Entering runlevel: 2
Starting enhanced syslogd: rsyslogdJFFS2 notice: (1123) check_node_data: wrong .
.
Starting system message bus: dbus.
Starting Dropbear SSH server: dropbear.
Starting MTA: exim4.
Starting NFS common utilities: statd.
Not starting internet superserver: no services enabled.
Starting file alteration monitor: FAM.
Starting NTP server: ntpd.
warning: `ntpd' uses 32-bit capabilities (legacy support in use)
Starting Hardware abstraction layer: hald.
Starting deferred execution scheduler: atd.
Starting periodic command scheduler: crond.

Debian GNU/Linux 5.0 sam9-l9260 ttyS0

sam9-l9260 login: JFFS2 notice: (815) check_node_data: wrong data CRC in data n.

Debian GNU/Linux 5.0 sam9-l9260 ttyS0

sam9-l9260 login: olimex
```

Figure 4-6 Minicom

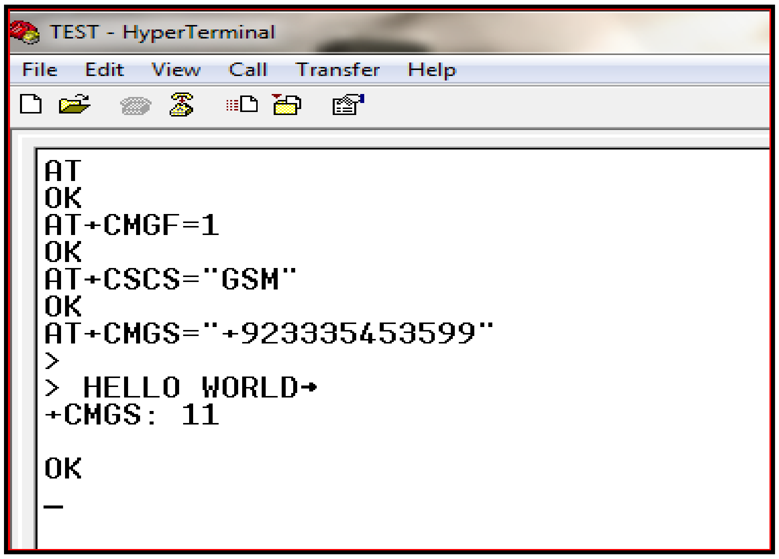
4.2.2. Cygwin

Cygwin is a Unix-like environment that is used forcommand-line interface for Microsoft Windows. Cygwin provides means for integration of Windows-based applications, data, and other system resources with applications, software tools, and data of the Unix-like environment. Thus it is possible to launch Windows applications from the Cygwin environment, as well as to use Cygwin tools and applications for UNIX applications within the Windows operating context.

4.2.3. HyperTerminal

HyperTerminal is a software program that is intended to carry out the functions of communication and terminal emulation. It uses the serial port to establish communication with the external devices. These devices can vary and include such options as radio communications equipment, robots, and instruments used for scientific measurements and similar endeavors. The HyperTerminal makes it easy to access these devices and also operate them while using the computer.

GSM modem was testedusing HyperTerminal and various AT commands were explored and experimented with.The commands in Figure 4-7 were used in order to check if the modem’s settings were configured correctly or if the modem is connected properly.



```
AT
OK
AT+CMGF=1
OK
AT+CSCS="GSM"
OK
AT+CMGS="+923335453599"
>
> HELLO WORLD→
+CMGS: 11

OK
_
```

Figure 4-7 Testing of GSM Modem in HyperTerminal

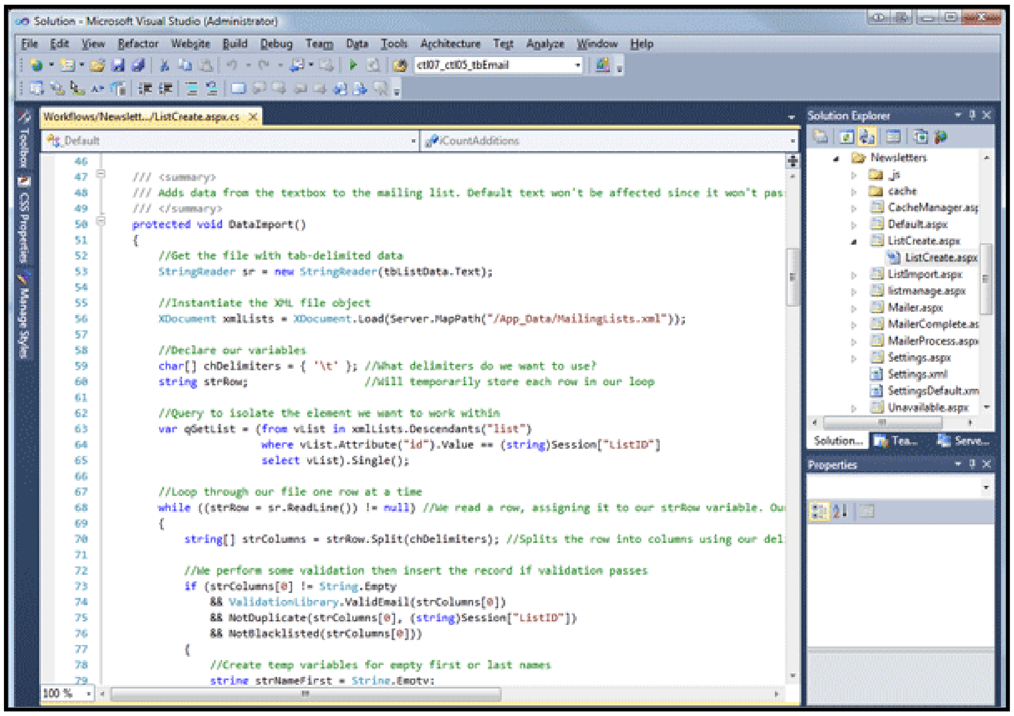
4.2.4. MySQL

MySQL client version: 5.0.22, which is an open source relational database management system, was used to maintain the record for all sensors and to maintain their history. The database was developed using phpMyAdmin version 3.2.4 which is a free Web-based front end widely installed by Web hosts worldwide, since it is developed in PHP. Various tables in the database are accessed by the ASP.Net service or PHP code when required using standard SQL queries of**Insert**-to add sensor’s reading to the “cenmars1” table,

Update-to update sensor record,**Delete**- to delete sensor record from the table and **Select**- to retrieve data from the table.

4.2.5. Visual Studio 2010, C# .NET

A windows Form application was developed in Visual Studio, C#.Net. C# was chosen because of its versatility and its importance in today's research environment. A code was formulated to receive SMS from the communication port of the designated server system. The database maintained in MySQL was linked to the website. The link to the database was formed using the .dll library MySQL. Data.. The C#.Net code is used to actually submit the information into the MySQL database. Visual Studio was not only used for database update but also for sending SMS through GSM modem when thresholds exceed. Figure 4-8 is the Visual Studio interface used to create the Windows application.



```
    /// <summary>
    /// Adds data from the textbox to the mailing list. Default text won't be affected since it won't pas
    /// </summary>
    protected void DataImport()
    {
        //Get the file with tab-delimited data
        StringReader sr = new StringReader(tbListData.Text);

        //Instantiate the XML file object
        XDocument xmlLists = XDocument.Load(Server.MapPath("/App_Data/MailingLists.xml"));

        //Declare our variables
        char[] chDelimiters = { '\t' }; //What delimiters do we want to use?
        string strRow;                  //Will temporarily store each row in our loop

        //Query to isolate the element we want to work within
        var qGetList = (from vList in xmlLists.Descendants("list")
                        where vList.Attribute("id").Value == (string)Session["ListID"]
                        select vList).Single();

        //Loop through our file one row at a time
        while ((strRow = sr.ReadLine()) != null) //We read a row, assigning it to our strRow variable. Ou
        {
            string[] strColumns = strRow.Split(chDelimiters); //Splits the row into columns using our del

            //We perform some validation then insert the record if validation passes
            if (strColumns[0] != String.Empty
                && ValidationLibrary.ValidEmail(strColumns[0])
                && NotDuplicate(strColumns[0], (string)Session["ListID"])
                && NotBlacklisted(strColumns[0]))
            {
                //Create temp variables for empty first or last names
                string strNameFirst = String.Empty;
```

Figure 4-8Visual Studio 2010

4.2.6. PHP

PHP (Hypertext Preprocessor), version 5.3.1.0., was chosen as the scripting language to create the web pages for the web interface that is the website. The website may be accessed by the admin at the server... Illustrative sensor profiles were generated and information was displayed when desired using the data from the MySQL database. PHP code, embedded into the HTML source document used standard SQL queries to attain data from the database once a connection was established with the local host at the web server.

CHAPTER 5

CONCLUSION

The project paves the way to automatic monitoring of un-manned sites along with timely reporting. A complete working prototype of Centralized Network Monitoring and Reporting System has been built to demonstrate an automatic environmental variables reading system using Client-Server network with reporting using GSM. The prototype includes simple and easy installation of ARM board with integrated sensors, which needs to be placed at the un-manned site, and the reporting system would then send SMS on the basis of alarm thresholds defined by the server end operator. The prototype developed, provides effective, reliable and efficient real time data readings from the environment using Ethernet connection and notification through the use of GSM network. Not forgetting the system devised is not only confined to the Telecommunications field but can also be extended to be used in homes as well as offices for constant monitoring.

5.1. Analysis

In the project, a working prototype of a remote site monitoring and reporting system was built with effective utilization of a few simple sensors and wired Ethernet cable infrastructure.

With the increase in un-manned sites, computerized monitoring is becoming many folds times a necessity. Computerized monitoring and reporting systems offer easier convenience to implement and provide automated site management. Various methods and technologies were established and developed already to provide and demonstrate the solution of efficiency, reliability and effectiveness of real-time monitoring. But all the methods are either too expensive to implement and operate, require complex setup of infrastructure, short operating distance and still require field intervention of human operators at the un-manned site or prone to error and have reliability issues due to delay in transmission. The developed project utilizes the reliability of Ethernet network. The reporting feature of SMS allows timely action if required.

5.1.1. Simple and Cost Effective Solution

The prototype makes efficient usage of the ARM-9 board which is an easy to use, cost effective and versatile controller board. A single controller board is utilized as a sensor interface unit and processing module per un-manned site. Many sensors can be integrated with the board simultaneously with very low overall processing.

5.1.2. No Data Loss

Since it is a wired connection link, the prototype module at the site sends real-time data to the server without any loss. The data is sent every second from the site with a minimal delay of few seconds, to reach the front-end GUI at the server. The reporting message for faulty condition is sent as soon as it occurs at the site.

5.1.3. Compact Design

After experimenting with the various linked modules, the compact design was developed for the final prototype that includes a controller board, four sensors and an Ethernet connection for the un-manned site. On the server side, a plug-in GSM Modem was used for the reporting purpose. The compact design of the prototype makes it easy to place into the existing un-manned site cabinets, meters, cupboards etc. This lays off the cost for redesigning the existing site architecture and saves both time and cost for the integration of additional hardware components.

5.1.4. Automated System

At the un-manned site, the controller board is fully operational with Linux as the OS and runs a C-language server code to capture data from the sensors as well as to process it before sending it to the server end. At the server end, the data is received and automatically uploaded to a database by running the server code. This removes the possibility of human error which is a major cause of discrepancy in the monitoring and reporting procedures.

5.1.5. Efficient Monitoring

An efficient mechanism for monitoring of environmental variables has been ensured by

using four basic sensors i.e. temperature, humidity, intrusion and smoke. The values are captured per second and then processed to be sent to the server end.

5.1.6. Ubiquitous Graphical User Interface

An illustrative and interactive user interface not only makes accessing information more easier, it also makes it ubiquitous in the sense that the officer at the server end may access their desired information of any site at a single interface. The data displayed is both elaborate and current, such that it is updated in real time with every received value.The user may view the complete information of a particular siteor the information of a particular sensor of a particular site. The data is complete with proper date and time stamp for accuracy.

5.2. Future Work

This project was taken up with the idea to computerize the un-manned sites, which would result in real time monitoring and timely response action system. It aims at providing some relief from the current financial crisis, because the implementation of such a system would result in transparency and cost efficiency. Moreover, it provides a means of keeping all the sites under full time monitoring from a central server. The future work in this field would require making the system more proficient by adding more sensors, as required by the company or location of their site. The constraints of the system need to be targeted and overcome, so the system is capable enough to be implemented commercially.

5.2.1Constraints

The objective of developing a computerized monitoring and reporting system has been successfully achieved, with features that add to the efficiency of the system. Nevertheless, the devised prototype has some constraints, which if overcome would lead to even better and improved performance by the system.

5.2.1.1. Currently for Wired Locations

The system devised is wire based thus offering a limited distance between the site location and the server. So, commercial systems having far off un-manned sites, would obviously suffer a great lag between the actual occurrence of the event at the site and its update on the GUI at the server end.

5.2.1.2. False alarm detection

Any telecom network might have thousands of sites deployed on the ground, e.g. BTS. These sites generally operate in a dumb manner being controlled by other entities, e.g. BSC. The increase in un-manned sites demands incorporation of maximum sites to the monitoring system. Increase in the number of sites, directly affects the network behavior, resource requirements, flexibility, database management, frontend management and overall credibility of the system. The most crucial point in managing such a vast network involves security from deliberate physical tampering, which will lead to false alarms.

5.2.1.3. Reporting System

The most important end goal of the entire system is the timely response to the situation. This goal is achieved through immediate alarm generation, and timely initial and subsequent reporting of the problem to relevant chain of individuals. Any delay in alarmgeneration or towards subsequent reactions may lead to huge financial and credibilityloss, making it a critical challenge.

5.2.2. Improvements

Owing to the constraints present in the prototype, the future work would require improvements that would overcome the flaws that might be faced when the system is considered for commercial implementation. These improvements will include usage of features that would add to the efficiency of the system.

5.2.2.1. Optimum use of Controller Board

The underuse of controller board needs to be countered, by introducing such an approach of data acquisition that would collect not only the environmental variables, but also other useful information like the sensor manufacturer, type, etc. Also, the processing should be done at the board in such a way to send only the changed values rather than every second data. This approach would prove beneficial, and would lead to optimum use of the advanced controller processing, offered by the controller board.

5.2.2.2. Sensor Status Detection

An effective tampering detection system can also be incorporated in the already devised prototype. This can be achieved by enabling SNMP between the sensors and the controller board, which would tell the sensor state whether it is working or has been tampered with. The state of the sensors from all the sites would be then displayed on the server GUI.

5.2.2.3. Wireless Transmission

An improvement in the system can be brought by using wireless media for transmission from the un-manned sites to the server rather than using the wired link. This can be done by either using GSM/GPRS as the transmission medium or Wi-Fi (depending upon the distance or usage). For best results, GPRS can be used for dedicated transmission between the sites and the server.

5.2.2.4. Multiple SIMs for Reporting

Mobile Networks have the tendency to get down for some time, which can be due to a number of unforeseen reasons. Such a collapse in the system can be overcome by the use of multiple SIMs, from different mobile service provider companies. So that when the network of one mobile service Provider Company is down, the Modem can switch to another SIM, so that no loss of timely reporting occurs at the server end.

5.2.2.5. System Expansion

The monitoring and reporting system can be expanded to include multiple sites from different locations, as well. Only the means of transmitting values has to be changed to wireless media so that it could work without being constrained by distance. The system would include the same modules of the prototype at each un-manned site, and the remaining server end system would remain the same.

5.2.2.6. Reporting via GPRS

Another improvement in the system can be to use GPRS as the transmission medium of reporting instead of GSM at the server end. The reporting module needs to be configured/changed for the GPRS support. A key advantage of GPRS over GSM is that GPRS has a higher data transmission speed. If SMS over GPRS is used, an SMS transmission speed of about 30 SMS messages per minute may be achieved.

5.2.2.7. Actuators Integration

A major improvement can be brought upon by introducing actuators at the un-manned site. This would enable immediate action without any consent from the server end. For example, if smoke alarm goes on, an automated actuator response of starting fire extinguisher can be programmed.

5.2.3. Commercial Implementation

The project developed is a prototype, which can be implemented commercially after making some required modifications. These considerations have been listed, which would be encountered in the process of effectuation.

5.2.3.1. Transmission Media

A major percentage of the companies have far away locations that need to be monitored. For commercial implementation, the medium of transmission from the site to the server needs to be made wireless, either by using GSM or GPRS. The main requirement is the addition of a GSM/GPRS module at the un-manned site to allow this modification to be made.

5.2.3.2. Database Expansion

With the expansion of the system, the need to define more fields would arise. Separate databases for different sites would be required, so that different standards and thresholds can be maintained for each site. Hence, an even more elaborate database would be required.

6. Glossary

Monitoring System

A system that acquires data from the environment through sensors and passes to the controller for processing

Global Positioning System (GPS)

A space-basedglobal navigation satellite system (GNSS) that provides location and time information in all weather, anywhere on or near the Earth

Global System for Mobiles (GSM)

A standard set developed by theEuropean Telecommunications Standards Institute(ETSI) to describe technologies for second generation (or "2G") digitalcellular networks

MySQL

It is arelational database management system(RDBMS)that runs as a server providing multi-user access to a number of databases.TheSQLphrase stands for Structured Query Language.

Hypertext Preprocessor (PHP)

A general purposescripting languageoriginally designed forweb developmentto producedynamic web pages.

Real time system

A*real-time system*is one that processes information and produce a response within a specified time

Short Message Service (SMS)

Text communication service component of mobile communication systems, using standardized communications protocols that allow the exchange of short text messages between fixed line or mobile phone devices

Wi-Fi

A wireless standard for connecting electronic devices

APPENDIX

CODES

Server Code

```
#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <sys/ioctl.h>

#include <sys/mman.h>

#include <sys/time.h>

#include <unistd.h>

#include "time.h"

#include  "unistd.h"

#include "sys/types.h"

#include "sys/socket.h"

#include "netinet/in.h"

#include "netdb.h"

#include "string.h"

#include "strings.h"

#include "sys/wait.h"

#include<arpa/inet.h>

#include<sys/stat.h>

#include<sys/sendfile.h>

#include<fcntl.h>

#include "AT91SAM9260.h"


/* ************************** LOCAL DEFINES ****************************** */

#define ADC_MAX4

/* PC0 = ADC0

 * PC0 is routed to header SV2 pin 6
 * PC1 is routed to header SV2 pin 4

 * PC2 is routed to header SV2 pin 5

 * PC3 is routed to header SV2 pin 3

 *

 * SV2 pin 1 = +3.3VDC

 * SV2 pin 2 = GND

 *

 * The ADC signals range is 0 (GND) to +3.3V

/* They are NOT buffered on-board, so pay attention to the input impendence of the chip and the
output impedance of the driving source.  In many cases, this is not an issue. Also, follow good
anti-alias filtering principles to ensure good results.

 */

static unsigned adcSignalMapping[] ={

AT91C_PIO_PC0,

AT91C_PIO_PC1};

#define OUTPUT_PIN  AT91C_PIO_PC7

#define INPUT_PIN      AT91C_PIO_PC6

#define OUTPUT_PIN1AT91C_PIO_PC10

#define INPUT_PIN1    AT91C_PIO_PC9

#define ADC_MASK    (AT91C_PIO_PC0 | AT91C_PIO_PC1)

staticintmemMapFile;

static AT91PS_PIOMAP  at91PioCtrlr;

static AT91PS_ADC  at91ADC;

FILE *fpointer;


/* ************* TEMPERATURE  and  HUMIDITY FUNCTIONS******************* */

/ * Get the ADC channel reading

static unsigned GetADCValue(unsigned index){

unsigned            retVal[ADC_MAX];
unsigned        retry;
unsigned        value;
if (index >= ADC_MAX)
/* invalid index */
return (0);
/* wait for all conversions to complete*/
/* busy loop!*/
for (retry = 0; (retry < 1000) && \
(((at91ADC->ADC_SR) & ADC_MASK) != ADC_MASK); ++retry)
usleep(1000);
if (retry >= 1000){
printf ("ERROR: timeout waiting for ADC conversions complete\n");
return (0);}
/* get all channel raw values */
retVal[0] = (at91ADC->ADC_CDR0 & 0x3FF);
/* use ADC_LCRD to clear EOCx and DRDY */
value = at91ADC->ADC_LCDR;
/* start next conversion */
at91ADC->ADC_CR = AT91C_ADC_START;
return (retVal[0]);}
static unsigned GetADCValueHumidity(unsigned index)}
unsignedretVal[ADC_MAX];
unsigned retry;
unsigned value;
if (index >= ADC_MAX)
/* invalid index */
/* wait for all conversions to complete    */
/* busy loop!*/
```

```
for (retry = 0; (retry < 1000) && (((at91ADC->ADC_SR) & ADC_MASK) != ADC_MASK);
++retry)

usleep(1000);

if (retry >= 1000){

printf ("ERROR: timeout waiting for ADC conversions complete\n");

return (0);}

/* get all channel raw values */

retVal[1] = (at91ADC->ADC_CDR1 & 0x3FF);

/* use ADC_LCRD to clear EOCx and DRDY */

value = at91ADC->ADC_LCDR;

/* start next conversion */

at91ADC->ADC_CR = AT91C_ADC_START;

return (retVal[1]);}

/***************************Intrusion sensor************************************/

static void SetOutput(unsigned high){

if (high)

{/* set high */

at91PioCtrlr->PIOC_SODR = (OUTPUT_PIN);}

else{

/* set low aka "clear" */

at91PioCtrlr->PIOC_CODR = (OUTPUT_PIN);}

/* do not worry about sync */}

/** Read the input pin*/

Static unsigned GetInput(void){

if ((at91PioCtrlr->PIOC_PDSR) & (INPUT_PIN))

{return (1);}

return (0);}
```

```
/****************** HARDWARE SUPPORT FUNCTIONS ********************** */
/** Access the hardware*/
staticintOpenSystemController(void){
if ((memMapFile = open("/dev/mem", O_RDWR | O_SYNC)) < 0){
printf("ERROR: Unable to open /dev/mem\n");
return (errno);}
at91PioCtrlr = mmap(NULL, 4096, PROT_READ | PROT_WRITE,
MAP_SHARED, memMapFile, (unsigned)AT91C_BASE_AIC);
if (at91PioCtrlr == MAP_FAILED){
printf("ERROR: Unable to mmap the system controller\n");
close (memMapFile);
memMapFile = -1;
return (errno);}
at91ADC = mmap(NULL, 4096, PROT_READ | PROT_WRITE,
MAP_SHARED, memMapFile, (unsigned)AT91C_BASE_ADC);
if (at91ADC == MAP_FAILED)
{printf("ERROR: Unable to mmap the ADC\n");
close (memMapFile);
memMapFile = -1;
return (errno);}
/* set ADC input signals */
at91PioCtrlr->PMC_PCER= (1 << AT91C_ID_ADC);
at91PioCtrlr->PIOC_PDR          = ADC_MASK;
at91PioCtrlr->PIOC_ODR          = ADC_MASK;
at91PioCtrlr->PIOC_PPUDR    = ADC_MASK;
```

```
at91PioCtrlr->PIOC_ASR                    = ADC_MASK;
at91ADC->ADC_CR                                   = AT91C_ADC_SWRST;
usleep(1000);
at91ADC->ADC_MR= ((9 << 24) | (4 << 16) | (19 << 8));
at91ADC->ADC_CHER= ADC_MASK;
usleep(500);
at91ADC->ADC_CR= AT91C_ADC_START;
/* set digital input ports */
at91PioCtrlr->PIOC_ODR= INPUT_PIN;
at91PioCtrlr->PIOC_IFE= INPUT_PIN;
at91PioCtrlr->PIOC_PPUDR= INPUT_PIN;
at91PioCtrlr->PIOC_PER        = INPUT_PIN;
/* set digital output ports init value = 0 */
at91PioCtrlr->PIOC_CODR= OUTPUT_PIN;
at91PioCtrlr->PIOC_PPUDR= OUTPUT_PIN;
at91PioCtrlr->PIOC_PER        = OUTPUT_PIN;
at91PioCtrlr->PIOC_OER= OUTPUT_PIN;
/* sync memfile */
return (0);}
/******************************smokesensor***********************************/
static void SetOutput1(unsigned high){
if (high){
/* set high */
at91PioCtrlr->PIOC_SODR = (OUTPUT_PIN1);}
else{
/* set low aka "clear" */
at91PioCtrlr->PIOC_CODR = (OUTPUT_PIN1);}/* do not worry about sync */}
static unsigned GetInput1(void){if ((at91PioCtrlr->PIOC_PDSR) & (INPUT_PIN1)){
return (1);}
return (0);}
/* ****************** HARDWARE SUPPORT FUNCTIONS ********************** */
staticint OpenSystemController1(void)
{if ((memMapFile = open("/dev/mem", O_RDWR | O_SYNC)) < 0){
printf("ERROR: Unable to open /dev/mem\n");
return (errno);}
at91PioCtrlr = mmap(NULL, 4096, PROT_READ | PROT_WRITE,
MAP_SHARED, memMapFile, (unsigned)AT91C_BASE_AIC);
if (at91PioCtrlr == MAP_FAILED){
printf("ERROR: Unable to mmap the system controller\n");
close (memMapFile);
memMapFile = -1;
return (errno);}
/* set digital input ports */
at91PioCtrlr->PIOC_ODR          = INPUT_PIN1;
at91PioCtrlr->PIOC_IFER         = INPUT_PIN1;
at91PioCtrlr->PIOC_PPUDR    = INPUT_PIN1;
at91PioCtrlr->PIOC_PER          = INPUT_PIN1;
/* set digital output ports init value = 0 */
at91PioCtrlr->PIOC_CODR         = OUTPUT_PIN1;
at91PioCtrlr->PIOC_PPUDR    = OUTPUT_PIN1;
at91PioCtrlr->PIOC_PER          = OUTPUT_PIN1;
at91PioCtrlr->PIOC_OER          = OUTPUT_PIN1;
```

```
            return (0);}


            /* ****************************** MAIN LOOPS
            *********************************** */

            int main(intargc, char **argv){

            unsigned        adcIndexT;

            unsigned        adcValueT;

            unsigned        adcConversionT;

            unsigned        adctemperature;

            unsigned        adcIndexH;

            unsigned        adcValueH;

            unsigned        adcConversionH;

            unsigned        adcHumidity;

            struct tm {

            inttm_sec; /* seconds after the minute - [0,59] */

            inttm_min; /* minutes after the hour - [0,59] */

            inttm_hour; /* hours since midnight - [0,23] */

            inttm_mday; /* day of the month - [1,31] */

            inttm_mon; /* months since January - [0,11] */

            inttm_year; /* years since 1900 */

            Inttm_wday; /* days since Sunday - [0,6] */

            inttm_yday; /* days since January 1 - [0,365] */

            inttm_isdst; /* daylight savings time flag */};

            voidmyabort(char * msg) {

            printf("Error!:  %s" ,  msg); exit(1);}

int m;

intserverport = 3000;
```

```
int len1;

char * eptr = NULL;

        int sock_desc,sock_desc2;

        socklen_tlen;

        struct stat obj;

        char length[10];

        intfilehandle;

        intk,i;

        char filename[30];

        charfileline[4096];

        structsockaddr_inclient,server;

        memset(&client,0,sizeof(client));

        memset(&server,0,sizeof(server));

        sock_desc=socket(AF_INET,SOCK_STREAM,0);

        if(sock_desc==-1){

        puts("Error in socket 1");

        exit(1);}

        if (argc> 2){

        myabort("Usage: server "); }

        if (argc == 2){

        serverport =  (int) strtol(argv[1], &eptr, 10);

        if (*eptr != '\0') myabort("Invalid Port Number!") ;  }

        bzero(&server, sizeof(server));

        server.sin_family=AF_INET;

        server.sin_port = htons(serverport);

        server.sin_addr.s_addr = INADDR_ANY;
        k=bind(sock_desc,(structsockaddr *)&server,sizeof(server));

        if(k==-1)

        {

        puts("Error in binding");

        exit(1);}

        k=listen(sock_desc,5);

        if(k==-1){

        puts("Error in listening");

        exit(1);}

        len=sizeof(client);

        sock_desc2=accept(sock_desc,(structsockaddr *)&client,&len);

        if(sock_desc2==-1){

        puts("Error in socket2");

        exit(1);}

        if (OpenSystemController()){

        printf("Unable to map hardware resources\n");

        return (-1);}

        adcIndexT = 0;

struct tm *ptr;

time_tlt;

charstr[80];

char str1[80];

char door[10];

char smoke[10];

        while (1) {

time_t now; //time_t should be declared in time.h as long
```

```
struct tm *current; //pointer to array holding the current time
now = time(0); //current time in C representation
current =(struct tm*) localtime(&now);
sprintf(str,"%i:%02i:%02i",current->tm_hour,current->tm_min,current->tm_sec);
sprintf(str1,"2012/5/%i",current->tm_mday);
memset(&fileline,0,sizeof(fileline));
memset(&door,0,sizeof(door));
memset(&smoke,0,sizeof(smoke));
adcValueT = GetADCValue(adcIndexT);
adcConversionT = adcValueT * 3300000;
adcConversionT /= (1023 * 1000);
adctemperature = adcConversionT * 0.1;
adcValueH = GetADCValueHumidity(adcIndexH);
adcConversionH = adcValueH * 3300000;
adcConversionH /= (1023 * 1000);
adcHumidity = ((adcConversionH*0.00018148)-0.16)*161.29;
SetOutput(0);
SetOutput1(0);
if (GetInput()==1){
sprintf(door,"smoke");}
else{
sprintf(door,"normal");}
if (GetInput1()==1){
sprintf(smoke,"normal");}
    else{
    sprintf(smoke,"open");}
    sprintf(fileline,"Rawalpindi,Temperature Sensor ID 1000,%d,Humidity sensor ID 2000,%d,Door
    sensor ID 3000,%s,Smokesensor ID
    4000,%s,%s,%s\n",adctemperature,adcHumidity,smoke,door,str1,str);

    k=send(sock_desc2,fileline,strlen(fileline),0);

    len1=strlen(fileline);

    memset(&str,0,sizeof(str));

    memset(&str1,0,sizeof(str1));

    printf("%s",fileline);

    if(k==-1){

    puts("Error in sending");

    exit(1);}

    sleep(1);}

    return (0); /* never get here */}
```

Client Code

```
    #include<stdio.h>
    #include<stdlib.h>
    #include <errno.h>
    #include <sys/shm.h>
    #include <sys/types.h>
    #include <sys/ioctl.h>
    #include <sys/mman.h>
    #include <sys/time.h>
    #include <unistd.h>
    #include "time.h"
    #include  "unistd.h"
    #include "sys/socket.h"
    #include "netinet/in.h"
    #include "netdb.h"
    #include "string.h"
    #include "strings.h"
    #include "sys/wait.h"
    #include<arpa/inet.h>
    #include<sys/stat.h>
    #include<fcntl.h>
    #include <sys/uio.h>
int main(intargc,char **argv){
intk,choice,n;
    FILE *fpointer;
int len1,l;
int counter;
intsock_desc,filehandle;
structsockaddr_in client;
char length[4025];
char temp1[255];
charfoldername[255];
charbuf[255];
struct tm *ptr;
time_tlt;
charstr[80];
char f1[50];
memset(&client,0,sizeof(client));
sock_desc=socket(AF_INET,SOCK_STREAM,0);
if(sock_desc==-1){
puts("Unable to create socket");
exit(1);}
if(argc != 3) {
printf("Usage: hostname, port\n");
    exit(0);}
    client.sin_family=AF_INET;
    client.sin_port = htons(atoi(argv[2]));
client.sin_addr.s_addr = inet_addr(argv[1]);
bzero(&client.sin_zero, 8);
     k=connect(sock_desc,(structsockaddr *)&client,sizeof(client));
    if(k==-1){
    puts("Error in connect");
    exit(1);}
int s=1;
int i;
while(1){
lt = time(NULL);
ptr = localtime(&lt);
sprintf(temp1,"%d.txt",s);
fpointer = fopen(temp1, "a+");
if ( fpointer == NULL ) {
printf("There has been an error opening your file");
exit(1);}
memset(&length,0,sizeof(length));
memset(&foldername,0,sizeof(foldername));
     k=recv(sock_desc,length,255,0);
if (k==-1){
    puts("Error in receiving file");
exit(1);}
     len1=strlen(length);
fprintf(fpointer,length);
fclose(fpointer);
fflush(stdout);
memset(&temp1,0,sizeof(temp1));
    s++;}
    close(sock_desc);
    exit(1);}
```

Windows Application

```
 using System;
 usingSystem.Collections.Generic;
 usingSystem.Collections;
 usingSystem.ComponentModel;
 usingSystem.Text;
 usingSystem.Windows.Forms;
 usingSystem.Diagnostics;
 using System.IO;
 //Add MySql Library
 usingMySql.Data.MySqlClient;
 usingSystem.Data;
 //Add sms references
 usingSystem.Text.RegularExpressions;
 usingSystem.Drawing;
 namespaceConnectCsharpToMysql{
 public partial class Form1 : Form {
classDBConnect{
privateMySqlConnection connection;
private string server;
private string database;
private string uid;
private string password;
privateEmant.SMS sms1;
privateSystem.ComponentModel.IContainer components;
```

```
                //Constructor
    publicDBConnect(){
    Initialize();}
                //Initialize values
    private void Initialize(){
    server = "localhost";
database = "connectcsharptomysql";
uid = "root";
password = "javaria";
stringconnectionString;
            connectionString = "SERVER=" + server + ";" + "DATABASE=" + database + ";" +
            "UID=" + uid + ";" + "PASSWORD=" + password + ";";
            connection = new MySqlConnection(connectionString);
            this.components = new System.ComponentModel.Container();
             this.sms1 = new Emant.SMS(this.components);}
             //open connection to database
privateboolOpenConnection(){
try{
connection.Open();
return true;}
catch (MySqlException ex){
switch (ex.Number){
case 0:
MessageBox.Show("Cannot connect to server.  Contact administrator");
break;
```

```
        case 1045:
        MessageBox.Show("Invalid username/password, please try again");
        break;}
                    return false;}}
                    //Close connection
         privateboolCloseConnection(){
                    try{
     connection.Close();
     return true;}
                    catch (MySqlException ex){
       MessageBox.Show(ex.Message);
       return false;}}
                    //Insert statement
                    public void Insert()
                    { sms1.Open("COM3");
                    do{
                    string[] array1 = Directory.GetFiles(@"C:\\\\cygwin\\\\home\\\\Javaria\\\\ty\\\\","*.txt");
                    foreach (string name in array1) {
string[] allLines = File.ReadAllLines(name);
string[] smokealarm = new string[allLines.Length];
string[] doorstate = new string[allLines.Length];
int[] humidityvalue = new int[allLines.Length];
int[] temperaturevalue = new int[allLines.Length];
for (int i = 0; i <allLines.Length; i++){
string[] lineSplit = allLines[i].Split(',');
```

```
                    temperaturevalue[i] = Convert.ToInt32(lineSplit[2]);
if (temperaturevalue[i] > 45 || temperaturevalue[i] < 10){
sms1.Send("temperaturealert", "+923438589673");}
humidityvalue[i] = Convert.ToInt32(lineSplit[4]);
if (humidityvalue[i] > 40) {
sms1.Send("humidityalert", "+923438589673");}
smokealarm[i]= lineSplit[8];
if (smokealarm[i] == "normal")
{ sms1.Send("smokealert", "+923438589673"); }
doorstate[i] = lineSplit[6];
if (doorstate[i] == "open")
                    {sms1.Send("intrusionalert", "+923438589673");}}
                    string query = "LOAD DATA LOCAL INFILE \"" + name + "\" INTO TABLE
                    cenmars1 FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
                    (location,temperaturesensorID,temperaturevalue,humiditysensorID,humidityvalue,doorse
                    nsorID,doorstate,smokesensorID,smokealarm,date,time)";
                    //open connection
 if (this.OpenConnection() == true) {
                    //create command and assign the query and connection from the constructor
 MySqlCommandcmd = new MySqlCommand(query, connection);
                    //Execute command
 cmd.ExecuteNonQuery();
                    //close connection
 this.CloseConnection();
 File.Delete(name);
                    System.Threading.Thread.Sleep(919);}}} while (true);}}}}
```

BIBLOIGRAPHY

[1] Dr.B.Ramamurthy, S.Bhargavi, Dr.R.Shashi Kumar, "DEVELOPMENT OF A LOW-COST GSM SMS-BASED HUMIDITY REMOTE MONITORING AND CONTROL SYSTEM FOR INDUSTRIAL APPLICATIONS", (Ijacsa) International Journal of Advanced Computer Science and Applications, Vol. 1, No. 4, October 2010

[2] Kirianaki, Yurish, Shpak, Deynega, "DATA ACQUISITION AND SIGNAL PROCESSING FOR SMART SENSORS", 2002

[3] Brian Beej Jorgensen Hall, "BEEJ'S GUIDE TO NETWORK PROGRAMMING", Version 2.4.5, 2007

[4] Alex Berson, "CLIENT/SERVER ARCHITECTURE", McGraw-Hill, 1996

[5] Robin Nixon, "LEARNING PHP, MySQL & JAVASCRIPT", O'Reilly, 2009

[6] Chen PeijiangXueHua, "DESIGN AND IMPLEMENTATION OF REMOTE MONITORING SYSTEM BASED ON GSM", IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application 2008

[7] Hardware Design SIM900D_HD_V1.02, SIMCOM, A company SIM TECH

[8] HSU-04A1, Humidity Sensor Unit, datasheet

[9] SAM9-L9260 Development board user's manual, Olimex

[10] LM35DZ Precision Centigrade Temperature Sensors, datasheet

[11]]MQ-2 Semiconductor Sensor for Combustible Gas, datasheet