**CONTENTS**

1. ABSTRACT……………………………………………………………………………....1
2. INTRODUCTION………………………………………………………………………..2

2.1. ORGANISATION PROFILE…………………………………………………….3

3. LITERATURE SURVEY……………………………………………………………….....4

4. INTRODUCTION

4.1. OVERVIEW ABOUT TOOL………………………………………………...….5

4.2. MAJOR COMPONENTS………………………………………………………...6

4.3. FEATURES………………………………………………………………………7

4.4. PLATFORM……………………………………………………………………....8

5. HARWARE AND SOFTWARE IMPLEMENTATION ……………………………..…...9

6. SOFTWARE REQUIREMENTS SPECIFICATION……………………………….........10

* 1. GENERAL INTRODUCTION……………………………………………………...11
  2. PROBLEM STATEMENT………………………………………………………….12
  3. OBJECTIVE OF STUDY……………………………………………………………13
  4. SCOPE……………………………………………………………………………….14
  5. METHODOLOGY…………………………………………………………………...15

1. REQUIREMENT ANALYSIS…………………………………………………………..16
   1. USE CASE DIAGRAM ………………………………………………………….....17
   2. ACTIVITY DIAGRAM……………………………………………………………..18
   3. FUNCTIONAL AND NON FUNCTIONAL REQUIREMENTS…………………..19
2. DETAILED DESIGN…………………………………………………………………....20
   1. HIGH LEVEL DESIGN………………………………………………………….....21
   2. SEQUENCE DIAGRAM…………………………………………………………...22
   3. COLLABORATION DIAGRAM………………………………………………..…23
3. IMPLEMENTATION……………………...……………………………………..……...24
   1. INTRODUCTION…………………………………………………………….…..….25
   2. PROJECT ESTIMATION……………………………………………………………26
   3. CODING STANDARDS………………………………………………………….....27
4. TESTING………………………………………………………………………………...28
   1. INTRODUCTION…………………………………………………………...29
   2. TESTING STRATEGIES…………………………………………………...30
   3. TEST CASES…………………………………………………………..…….31
5. CONCLUSION……………………………………………………………………….…32
6. FUTURE ENHANCEMENT………………………………………………………..…..33
7. SNAPSHOTS………………...…………………………………………………………..34
8. BIBLIOGRAPHY……………………………………………………………………..…35

**CHAPTER 2**

**INTRODUCTION**

Indian Space Research Organization (ISRO), over the years, established a comprehensive network of ground stations to provide Telemetry, Tracking and Command (TTC) support to Satellite and Launch vehicle missions. These facilities are grouped under ISRO Telemetry, Tracking And Command Network (ISTRAC) with its headquarters at Bangalore, Karnataka State, INDIA.

**2.1 ORGANISATION PROFILE**

ISTRAC is responsible for providing Space Operation services that include spacecraft control, TTC support services and other related projects and services, for the launch vehicle and low earth orbiting spacecraft and deep space missions of ISRO and other space agencies around the world. Development of RADAR systems for tracking & atmospheric applications and Establishment of Ground Segment Network for Indian Regional Navigation Satellite System of ISRO are the additional responsibilities of ISTRAC.

* Ground facilities

India has established a strong infrastructure for executing its space programme. They include facilities for the development of satellites and launch vehicles and their testing; launch infrastructure for sounding rockets and satellite launch vehicles; telemetry, tracking and command network; data reception and processing systems for remote sensing.

A number of academic and research institutions as well as industries participate in the Indian Space Programme. Several Indian industries have the expertise to undertake sophisticated jobs required for space systems.

* Launch Facility

SDSC SHAR has the necessary infrastructure for launching satellite into low earth orbit, polar orbit and geostationary transfer orbit. The launch complexes provide complete support for vehicle assembly, fuelling, checkout and launch operations. Apart from these, it has facilities for launching sounding rockets meant for studying the earth's atmosphere.

* Tracking Facility

ISTRAC activities are organized into network operations, network augmentation, mission operation and spacecraft health monitoring, communications and computers and control centre facilities and development projects. Programme planning and reliability groups support ISTRAC activities.

* Data reception & dissemination

National Remote Sensing Centre (NRSC) is responsible for remote sensing satellite data acquisition and processing, data dissemination, aerial remote sensing and decision support for disaster management.

NRSC has archived a wealth of satellite images from Indian and foreign satellites since 1983. NRSC has its data reception facility at Shadnagar, 65 km from Hyderabad city. The station has four state of the art antenna systems for data reception and archival.  
  
The Satellite data processing chain has a user friendly web mechanism to enable users to order data of their requirement. It can facilitate to acquire data pertaining to any part of the globe on user request.

* Data Analysis

Remote sensing data are being used to map/monitor/survey/manage various natural resources of the country under National Natural Resources Management System (NNRMS) programmes. Funded by various user ministries and ISRO/DOS, these programmes have been generating valuable spatial data assets and information solutions.

**Indian Regional Navigational Satellite System (IRNSS)**

Indian Regional Navigational Satellite System is ISRO’s initiative to build an independent satellite navigation system based on a constellation of Geo Stationary satellites (GEO) and Geo Synchronous satellite (GSO). The navigation related ground elements at various locations could be effectively monitored and controlled from a central location.

**IRNSS Monitoring and Control System (IRMNC)**

IRNSS network is a complex and geographically distributed network with requirements of high availability server with adequate display monitors for observing the status of the system. The ground segment of IRNSS constellation would consist of a Master Control Center (MCC), ground stations to track and estimate the satellites' orbits and ensure the integrity of the network (IRIM), and additional ground stations to monitor the health of the satellites with the capability of issuing radio commands to the satellites (TT&C stations). The MCC would estimate and predict the position of all IRNSS satellites, calculate integrity, makes necessary ionospheric and clock corrections and run the navigation software. In pursuit of a highly independent system, an Indian standard time infrastructure would also be established. The IRNSS Monitoring and Control Processor (IMCP) will do the monitoring and control operations of the entire ground segment through sufficient number of Equipment Interface Processors (EIP).

Each equipment interface processor may remotely communicate with up to twenty equipments. The communication with the equipment will be through SNMP protocol and that with the equipment will be through TCP/IP or any other protocols like GBIP or RS-232 as and when required. The data acquired from the equipments will then be transmitted to the clients who requested for the data, in a timely manner on a 24X7 basis. The System is intended to provide an absolute position accuracy of better than 20 meters throughout India and within a region extending approximately 2,000 km around it.

ISTRAC is one of the important units of ISRO that has the key task of monitoring and commanding various low earth orbiting satellites. ISTRAC has a network of ground stations to track satellites and all these stations are remotely monitored and controlled from a centralized Network Control Centre (NCC). Monitoring of stations involve interfacing with each and every equipment of every ground station, to monitor their health and to configure them periodically. These equipments can be interfaced with a computer on various protocols, as per the designed support.

ISTRAC has already established this remote monitoring and control facility and all the status data from stations flow to the control center on satellite links. It becomes necessary to monitor the link status so that any link failure can be alerted to provide 24 X 7 service in an uninterrupted way. Also some of the latest equipments are supplied with SNMP protocol for its monitoring and control.

**CHAPTER 3**

**LITERATURE SURVEY AND REVIEW**

**3.1 Literature Survey**

* **SNMP**

The Simple Network Management Protocol, also known as SNMP, is a vital protocol for Network Administrators. The SNMP Protocol uses IP with UDP or TCP. The SNMP protocol allows an Administrator to request information about one or more network devices hardware, software or configuration from a single point of management.

The SNMP Protocol has two sides, the agent and the management stations. The agent sends data about itself to the management station. The management station collects data from all the agents on the network. The protocol is not identical on each side. The agent sends alerts called traps and answers requests that were sent by the management station. The management station catches and decodes the traps. The management station also requests specific information from the agent.

The agent is a server, router, printer, bridge or workstations. Any network device that can use the IP protocol can be an SNMP agent.

The SNMP agent will monitor hardware and send a trap is any problems occur. For example, if a network printer runs low on toner or a server has a hard drive go bad, a trap will be sent to the management station. The management station catches the trap and acts on the trap as configured by the administrator. The administrator could configure the management station to send an e-mail or have the administrator paged.

The management station may also send a request to the agent. In this case the SNMP agent will reply to the management station with the specific data. For example the management station may wish to view the agents hardware event log and see the current BIOS level. The agent will reply with SNMP packets containing the hardware’s events and BIOS level. The information is passed through requests and replies with the use of the MIB or the Management Information Base.

**SNMP Messages**

SNMP messages may be initiated by either the network management system (NMS) or by the network element.

An SNMP TRAP is a message which is initiated by a network element and sent to the network management system. For example, a router could send a message if one of it’s redundant power supplies fails or a printer could send an SNMP trap when it is out of paper.

An SNMP GET is a message which is initiated by the network management system when it wants to retrieve some data from a network element. For example, the network management system might query a router for the utilization on a WAN link every 5 minutes. It could then create charts and graphs from that data, or it could warn the operator when the link was over utilized.

An SNMP SET is a message which is initiated by the NMS when it wants to change data on a network element. For example, the NMS may wish to alter a static route on a router.

**The SNMP MIB**

The SNMP MIB, or Management Information Base, is a collection of variables which is shared between the NMS and the network element (NE).

The MIB is extensible, which means that hardware and software manufacturers can add new variables to the MIB. These new MIB definitions must be added both to the network element and to the network management system.

SNMP lets TCP/IP-based network management clients use a TCP/IP-based internetwork to exchange information about the configuration and status of nodes. The information available is defined by a set of managed objects referred to as the SNMP Management Information Base (MIB).

Each SNMP element manages specific objects with each object having specific characteristics. Each object / characteristic has a unique object identifier (OID) consisting of numbers separated by decimal points (i.e., 1.3.6.1.4.1.2682.1). These object identifiers naturally form a tree as shown below. The SNMP MIB associates each OID with a readable label (i.e., dps RTUA State) and various other parameters related to the object. The MIB then serves as a data dictionary or code book that is used to assemble and interpret SNMP messages.

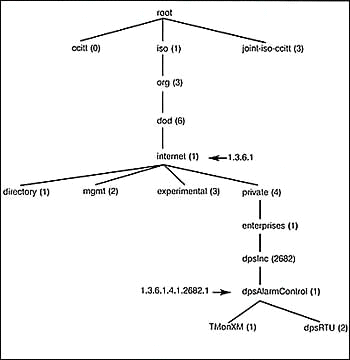


Fig 3.1.1 Sample SNMP tree structure

When an SNMP manager wants to know the value of an object / characteristic, such as the state of an alarm point, the system name, or the element uptime, it will assemble a GET packet that includes the OID for each object / characteristic of interest. The element receives the request and looks up each OID in its code book (MIB). If the OID is found (the SNMP object is managed by the element), a response packet is assembled and sent with the current value of the object / characteristic included. If the OID is not found, a special error response is sent that identifies the unmanaged object.

When an element sends an SNMP TRAP packet, it can include OID and value information (bindings) to clarify the event. DPS remote units send a comprehensive set of bindings with each TRAP to maintain traditional telemetry event visibility. Well-designed SNMP managers can use the bindings to correlate and manage the events. SNMP managers will also generally display the readable labels to facilitate user understanding and decision-making.

* **TCP/IP**

TCP/IP is most often studied in terms of its layer-based architecture and the protocols that it provides at those different layers. And we’re certainly going to do that, don’t worry. These protocols, however, represent the technical details of how TCP/IP works. They are of interest to us as students of technology, but are normally hidden from users who do not need to see the “guts” of how TCP/IP works to know that it works. Before proceeding to these details, I think it might be instructive to take a “bigger picture” look at what TCP/IP does.

**TCP/IP Services**

TCP/IP covers many layers of the OSI model, and so it collectively provides services of this sort as well in many ways. Conceptually, we can divide TCP/IP services into two groups: services provided to other protocols and services provided to end users directly.

**Services Provided to Other Protocols**

The first group of services consists of the core functions implemented by the main TCP/IP protocols such as IP, TCP and UDP. These services are designed to actually accomplish the internetworking functions of the protocol suite connections between [devices](http://www.tcpipguide.com/free/t_TCPIPServicesandClientServerOperation.htm#). Other protocols provide routing and management functionality. Higher-layer protocols use these services, allowing them to concentrate on what they are intended to accomplish.

**End-User Services**

The other general types of service provided by TCP/IP are end-user services. These facilitate the operation of the [applications](http://www.tcpipguide.com/free/t_TCPIPServicesandClientServerOperation.htm#) that users run to make use of the power of the Internet and other TCP/[IP networks](http://www.tcpipguide.com/free/t_TCPIPServicesandClientServerOperation.htm#). For example, the [World Wide Web (WWW)](http://www.tcpipguide.com/free/t_TCPIPWorldWideWebWWWTheWebandtheHypertextTransferP.htm) is arguably the most important Internet application. WWW services are provided through the [Hypertext Transfer Protocol (HTTP)](http://www.tcpipguide.com/free/t_TCPIPHypertextTransferProtocolHTTP.htm), a TCP/IP application layer protocol. [HTTP](http://www.tcpipguide.com/free/t_TCPIPServicesandClientServerOperation.htm#) in turn uses services provided by lower-level protocols

**The TCP/IP Client/Server Structural Model**

An important defining characteristic of TCP/IP services is that they primarily operate in the client/server structural model. This term refers to a system where a relatively small number of (usually powerful) server machines is dedicated to providing services to a much larger number of client hosts. Just as client/server networking applies to hardware, this same concept can be applied to software and protocols, and this is exactly what was done in the design of TCP/IP protocols and applications.

TCP/IP protocols are not set up so that two machines that want to communicate use identical software. Instead, a conscious decision was made to make [communication function](http://www.tcpipguide.com/free/t_TCPIPServicesandClientServerOperation-2.htm#) using matched, complementary pairs of client and server software. The client initiates communication by sending a request to a server for data or other information. The server then responds with a reply to the client, giving the client what it requested, or else an alternative response such as an error message or information about where else it might find the data. Most (but not all) TCP/IP functions work in this manner.

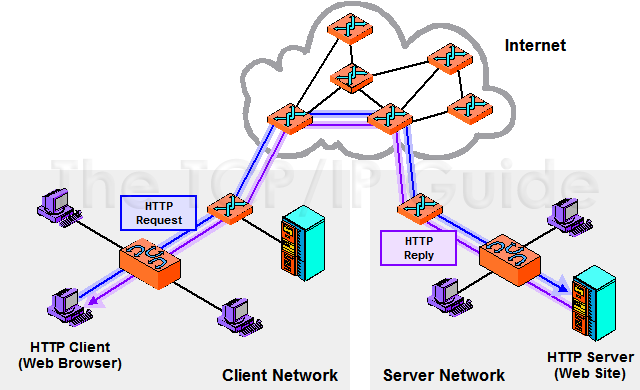


Fig 3.1.2 TCP/IP client/server operation

There are numerous advantages to client/server operation in TCP/IP. Just as client hardware and server hardware can be tailored to their very different jobs, client software and the server software can also be optimized to perform their jobs as efficiently as possible. Let’s take again the WWW as another example. To get information from the Web, Web client software (usually called a browser) sends requests to a Web server. The Web server then responds with the requested content. (There’s more to it than that, of course, but that’s how it appears to the user.) The Web browser is created to provide the interface to the user and to talk to Web servers; the Web server software is very different, generally consisting only of high-powered software that receives and responds to requests.

**3.2 Literature Summary**

Thus based on the literature survey we are using the SNMP protocol for monitoring and control of the ground station equipments. Thus this application will provide an additional interface support to the existing network monitoring & control system (NMS) which interfaces equipments on TCP/IP, GPIB, RS 232, etc. We are using the connection oriented TCP/IP protocol for reliable communication between the client and the manager. We are using the National Instruments LabWindows for building the software because it is a strict organizational requirement as their existing systems were built using the same tool.

**CHAPTER 4**

**INTRODUCTION**

* 1. **OVERVIEW ABOUT NI CVI Lab Windows**

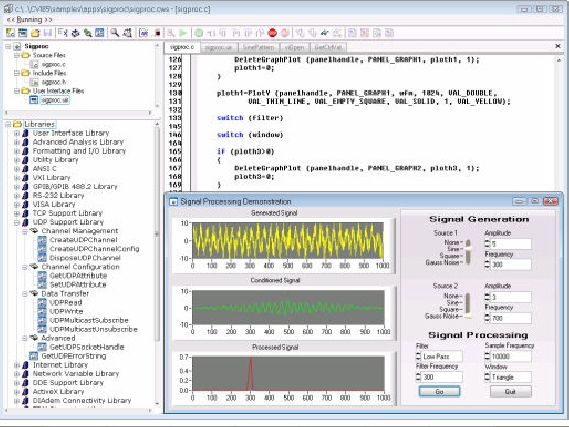
LabWindows/CVI is a proven ANSI C development environment for test and measurement that greatly increases the productivity of engineers and scientists.

National Instruments LabWindows/CVI streamlines development with hardware configuration assistants, comprehensive debugging tools, and interactive execution to run functions at design time. With the built-in ANSI C measurement libraries, you can rapidly develop complex applications such as multithreading and ActiveX server/client programs. The flexibility of NI LabWindows/CVI optimizes the data acquisition, analysis, and presentation of all test and measurement applications.

NI LabWindows/CVI is a proven ANSI C development environment for test and measurement that  
greatly increases the productivity of engineers and scientists. For more than 20 years, C  
developers have used LabWindows/CVI to develop high-performance, stable applications in the  
manufacturing test, military and aerospace, telecommunications, design validation, and  
automotive industries. LabWindows/CVI streamlines development in these areas with hardware  
configuration assistants, built-in measurement libraries, comprehensive debugging tools,  
interactive execution capabilities that developers can use to run functions at design time, and  
advanced analysis and scientific user interface tools.

* 1. MAJOR COMPONENTS
     1. INTEGRATED ENVIRONMENT

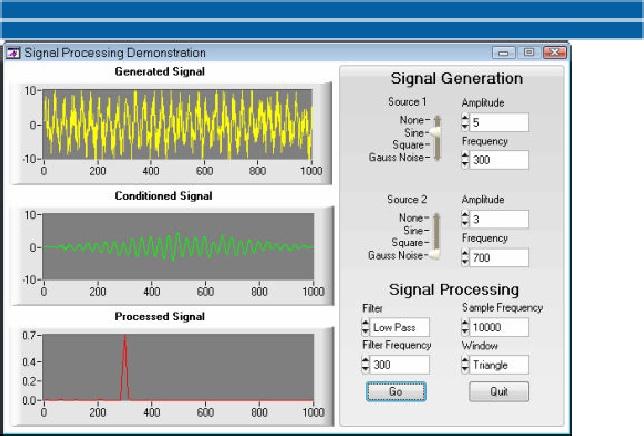
The LabWindows/CVI development environment, as shown in Figure, includes a workspace window that is divided into five main areas – the Project Tree, the Library Tree, the Window Confinement Region, the Output Region, and the Debugging Region.

The integrated workspace provides an intuitive, convenient interface for creating and managing large projects. You can also customize each of the areas to fit your specific development standard preferences, and integrate with source code control, requirements management, and data management systems.

4.2.2. STATE-OF-THE-ART HARDWARE CONNECTIVITY

Often the most critical process in a measurement application is connecting to an instrument and taking measurements. LabWindows/CVI simplifies the task of connecting to and communicating with a wide variety of instruments. With the LabWindows/CVI built-in measurement libraries, you can quickly acquire data from GPIB, USB, serial, Ethernet, PXI, and VXI instruments using the built-in instrument I/O libraries or using built-in instrument drivers.

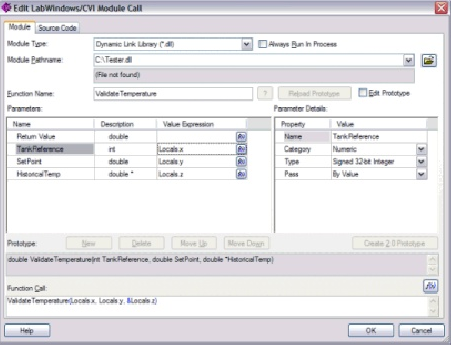


* + 1. **scientific advanced analysis**

LabWindows/CVI provides powerful algorithms and functions designed specifically for measurement analysis and signal processing.

* + 1. productive dev elopement tools

LabWindows/CVI also includes key productivity features with which you can efficiently create  
end-to-end solutions. Some additional productivity tools include new project and file templates for  
easily implementing consistent programming style while reducing the amount of redundant tasks  
performed between multiple applications.

****

cientific Advanced Analysis

LabWindows/CVI provides powerful algorithms and functions designed specifically for measurement analysis and signal processing

* 1. FEATURES
* **Multithreaded Utility Functions**  
  LabWindows/CVI 5.5 provides many utility functions to make multithreaded programming easier. You can create and manage threads with the new thread pool API. You also can protect your data in a multithreaded program with LabWindows/CVI's thread-safe queues, thread-local variables, and thread-safe variables.
* **New DataSocket Library**  
  DataSocket simplifies live data exchange between different applications on one computer or between computers connected through a network. DataSocket implements an easy-to-use, high-performance programming tool that is designed specifically for sharing and publishing live data in measurement and automation applications.
* **New RS-232 Library Functions**  
  GetComLineStatus  
  GetSystemComHandle
* **New Advanced Analysis Functions**  
  FastHilbertTransform  
  InvFastHilbertTransform  
  Erf (Error Function)  
  Erfc (Complementary Error Function)
* **Report Generation**  
  CVI provides two instrument drivers to help you generate reports. You can easily create and print formatted text reports with the NIReports instrument driver. You also can use the WordReport instrument driver to create and manipulate reports in Microsoft Word.
* **Console Applications Support**  
  A console application uses the host system's Standard I/O window to display standard output and to read standard input, instead of using the LabWindows/CVI Standard I/O window. You can use a console window for standard I/O, even when your application is not a console application, by passing HOST\_SYSTEM\_STDIO to the Utility Library SetStdioPort function.
* **Run-Time Engine and Windows Shutdown**  
  Applications using the LabWindows/CVI Run-time Engine no longer interfere with the Windows shutdown process. In previous versions of the LabWindows/CVI Run-time Engine, if you attempt to shut down, restart, or logoff Windows while an application that uses the LabWindows/CVI Run-time Engine is running, the system might not shut down immediately and might always shut down regardless of the shutdown command issued. The LabWindows/CVI 5.5 Run-time Engine no longer interferes with the Windows shutdown process.
* **Updated Windows SDK**  
  The Windows SDK has been updated to include the functions from the Windows 2000 Platform SDK. This includes all of the Windows 98 functions as well. Only a subset of the Windows SDK is included with the LabWindows/CVI beta versions.
* **Formatting and I/O Library Fmt/FmtOut/FmtFile Functions**  
  The Fmt, FmtFile, and FmtOut functions now support the 'j' modifier, which allows you to specify characters that are inserted between each element when you use the repeat code to format multiple items.

**1.2 Problem Statement**

Some of the latest equipments are supplied with SNMP protocol for its monitoring and control. This application is developed to provide monitoring and control support on SNMP interface for a given set of equipments. The application is to be built in such a way that any other equipment with SNMP interface could be integrated at any point of time. This application would Provide an additional interface support to the existing network monitoring & control system (NMS) which interfaces equipments on TCP/IP, GPIB,RS 232, etc.

**1.3 Importance**

Building this interface will enhance existing monitoring and control system at ISTRAC by

1. Providing additional interface support
2. Adding any new equipment in future
3. Monitor communication link by monitoring routers, switches etc., that typically support SNMP for monitoring, thus avoiding commercial Network Monitoring Systems.

**1.4 Organization of Report**

The report is organized in such a way that it contains all the important details pertaining to the project. This report contains the activity diagram, data flow diagrams, use case diagram, high level design and the low level design. This report also contains all the test cases and their status and the results of unit and integration testing. This report also contains the requirements and the implementation of the application. Only the code is missing in the report because the organization permissions do not allow that but sample screen shots are included. This report also contains the scope for future work on this project.

**CHAPTER 2**

**OVERVIEW AND PLANNING**

**2.1 Proposed System Overview:**

ISTRAC is one of the important units of ISRO that has the key task of monitoring and commanding various low earth orbiting satellites. ISTRAC has a network of ground stations to track satellites and all these stations are remotely monitored and controlled from a centralized Network Control Centre (NCC). Monitoring of stations involve interfacing with each and every equipment of every ground station, to monitor their health and to configure them periodically. There are various equipments involved in the function of a ground station. Some Ground Station Equipments are

* Frequency up-down converter
* RF spectrum analyzers
* Power meters
* Signal generators
* Antennae controller
* Amplifiers

These equipments can be interfaced with a computer on various protocols, as per the designed support. ISTRAC has already established this remote monitoring and control facility and all the status data from stations flow to the control center on satellite links. It becomes necessary to monitor the link status so that any link failure can be alerted to provide 24X7 service in an uninterrupted way. Also some of the latest equipments are supplied with SNMP protocol for its monitoring and control. Common devices that typically support SNMP include routers, switches, Servers, workstations, printers.

This application is developed to provide monitoring and control support on SNMP interface for a given set of equipments. The application is to be built in such a way that any other equipment with SNMP interface could be integrated at any point of time. Building this interface should enhance existing monitoring and control system at ISTRAC by

1. Providing additional interface support
2. Adding any new equipment in future
3. Monitor communication link by monitoring routers, switches etc., that typically support SNMP for monitoring, thus avoiding commercial Network Monitoring Systems.

The key components are

* **Managed device**- routers, switches, hosts
* **Manager** (Network Management System)-executes applications that monitor and control managed devices. A *manager* is an SNMP application that generates queries to SNMP agent applications and receives traps from SNMP agent applications.
* **Agent**- software that runs in the managed device that communicates with the manager. The SNMP agent is responsible for retrieving and updating local management information based on the requests of the SNMP manager. The agent also notifies registered managers when significant events or *traps* occur

The management data may be Modes of operation, Faults, Simple on-off status, Locks, Power supply etc. These variables can be read-only, write-only or read-write. The Manager can modify the variables accordingly. The management variables are stored in a database called Management Information Base (MIB).

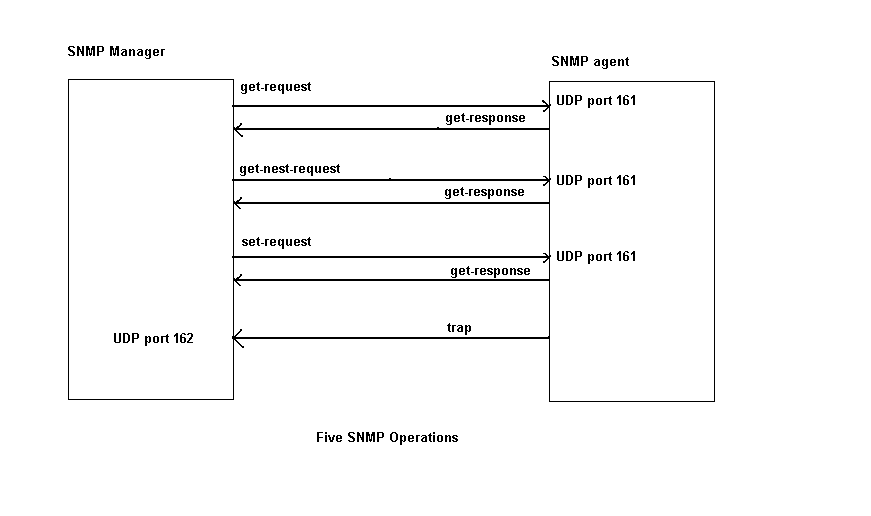
This database is present in the managed device. The agent running in the managed device interacts with the database. The Simple Network Management Protocol uses messages to communicate and exchange information between remote SNMP entities. SNMP messages contain protocol data units (PDUs) plus additional message header elements defined by the relevant RFC. A PDU is a data packet that contains SNMP data components (or fields). So when a manager sends a query message as a get message then the message is interpreted by the agent and the agent communicates with the MIB in the device and sends the response message with the status of the requested variables. When the manager requests for setting of any variable then the message is received by the agent and interprets the OID and value of the variable to set. It communicates with the MIB and then sends the response message with the new value.



Fig: 2.1.1 Managed Devices

SNMP defines only five types of messages that are exchanged between the manager and agent.

1. Fetch the value of one or more variables: the get-request operator.
2. Fetch the next variable after one or more specified variables: the get-next-request operator.
3. Set the value of one or more variables: the set-request operator.
4. Return the value of one or more variables: the get-response operator. This is the message returned by the agent to the manager in response to the get-request, get-next-request, and set-request operators.
5. Notify the manager when something happens on the agent: the trap operator.

 Fig 2.1.2 Five SNMP Operations

**2.2 Challenges**

* The computer has to be interfaced with the ground station equipment.
* Both the programming computer and the ground station equipments should be able to communicate.
* The ground station equipments should reply to the manager appropriately.
* The agent in the ground station equipments should query the MIB regularly and send the odd status variables to the manager for further action.
* The equipments should effectively send traps to the manager if it requires the manager to look at the situation.

**2.3 Assumptions**

* Our main assumption is that the ground station equipment should support SNMP so that the SNMP manager can be interfaced with the equipments.
* The ground station equipment should send the message in the appropriate manner that the can interpret the message.

**2.4 Architecture Specifications**

The Architecture diagram depicts the overall architecture of the project. It gives us the overview of the project. The Architecture diagram is of two types, high level and low level architecture diagrams. The high level architecture diagram consists of only the overview of the architecture of the project without going into the particular details of the modules of the project.

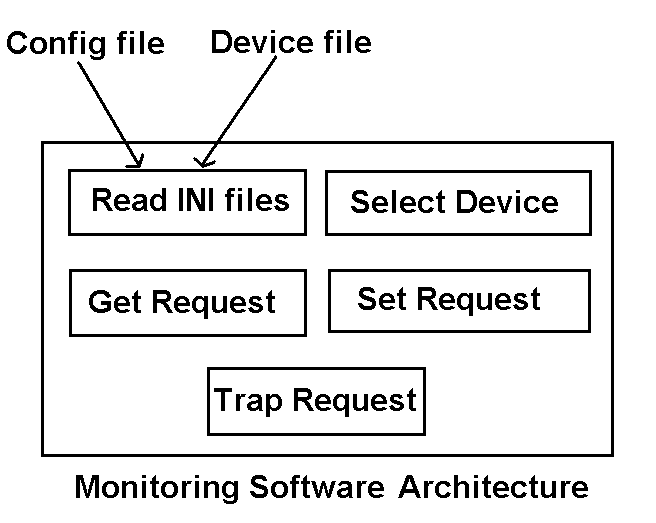


Fig 2.4.1 High Level Architecture Diagram

The low level architecture diagram gives us the each and every detail regarding the modules of the project. It gives us a view of what all the modules are used in the project and again what all the subparts are contained in each module. It is just the step prior to coding. The only step remaining after drawing the low level architecture diagram is the coding of the project.

In the current application the high level architecture diagram gives us the view of the major modules of the project. All these major modules are expanded into subparts in the low level architecture diagram. The Read files module takes the INI files as input and contains the modules for reading the INI files when required. Similarly the Select Device module consists of methods for connecting to the client using the TCP/IP protocol and retrieving the device selected and the refresh rate from the client. The Get Request module consists of the methods for getting the OIDS of the device selected by the user and getting its values from the Device. The Set Request consists of the methods for retrieving the OID and the value to be set from user and setting the corresponding value in the MIB of the device. The Trap Request module consists of the methods for registering the manager with the device for receiving messages about any peculiar conditions or errors.

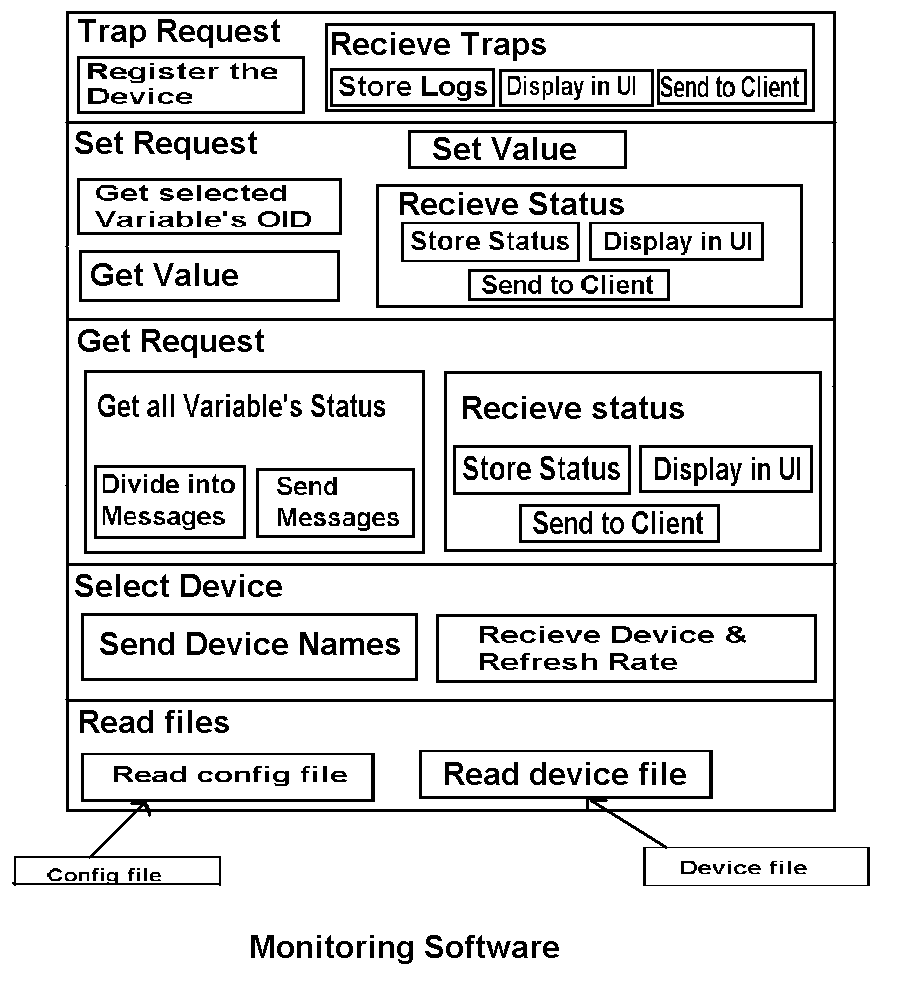


Fig 2.4.2 Low level Architecture Diagram

**UML Diagrams**

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML is appropriate for modeling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real time embedded systems. It is a very expressive language, addressing all the views needed to develop and then deploy such systems.

The vocabulary of the UML encompasses three kinds of building blocks

* Things
* Relationships
* Diagrams

Things are the abstractions that are first-class citizens in a model, Relationships tie these things together, and diagrams group interesting collections of things.

Adiagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). Diagrams projects a system from different angles and perspectives. The UML has nine diagrams, classified into two categories

**Use Case Diagram**

A use case diagram shows a set of use cases and actors and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system.

Use case diagrams commonly contain

* Use cases
* Actors
* Dependency, generalization and association relationships

Use case

A use case describes a set of sequences, in which each sequence represents the interaction of the things outside the system ( its actors) with the system itself. A use case represents a functional requirement of a system as a whole. A use case involves the interaction of actors and the system.



Actor

An actor represents a coherent set of roles that users of use cases play when interacting with these use cases. Actors can be human or they can be automated systems.

Dependency

A dependency is a semantic relationship between two things in which a change to one thing (the independent thing) may affect the semantics of the other thing (the dependent thing).

A dependency is rendered as a dashed line, possibly directed.

Generalization

A generalization is a specialization/generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent). With this the child shares the structure and the behavior of the parent.



Association

An association is a structural relationship that describes a set of links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts. Graphically, an association is rendered as a solid line.



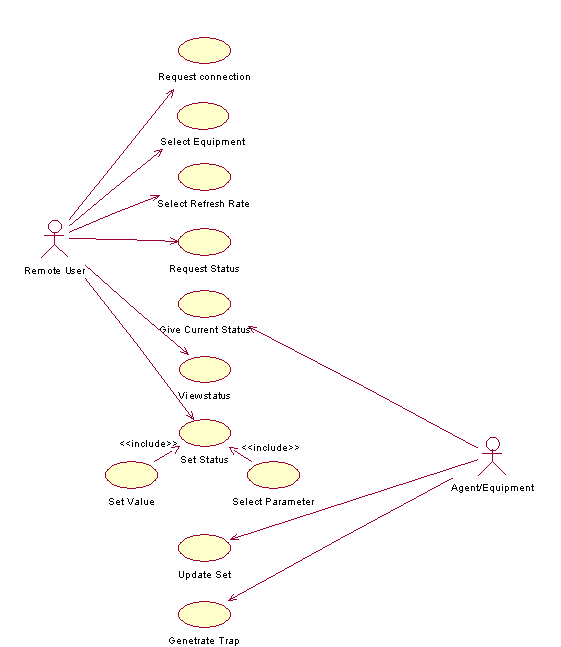
****

Fig: 2.4.3 Use Case Diagram

**Sequence Diagram**

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram is a table that shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis.

Sequence diagrams commonly contain

* Objects
* Links
* Messages

Sequence diagrams have two features. First, there is a object life line. An object lifeline is the vertical dashed line that represents the existence of an object over a period of time. Most objects that appear in an interaction diagram will be in existence for the duration of the interaction, so the objects are all aligned at the top of the diagram, with their life lines drawn from the top of the diagram to the bottom. Objects may be created during the interaction, their lifelines start with the receipt of the message stereotyped as create. Objects may be destroyed during the interaction, their lifelines start with the receipt of the message stereotyped as destroy.

Second, there is a focus of control. The focus of control is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure. The top of the rectangle is aligned with the start of the action; the bottom is aligned with its completion.

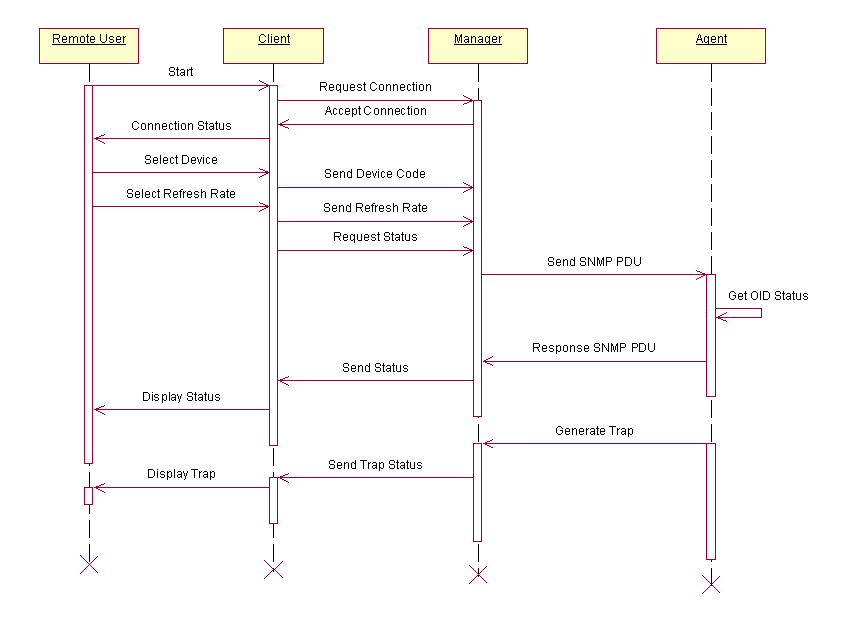


Fig: 2.4.4 Sequence Diagram for Get Request and Trap

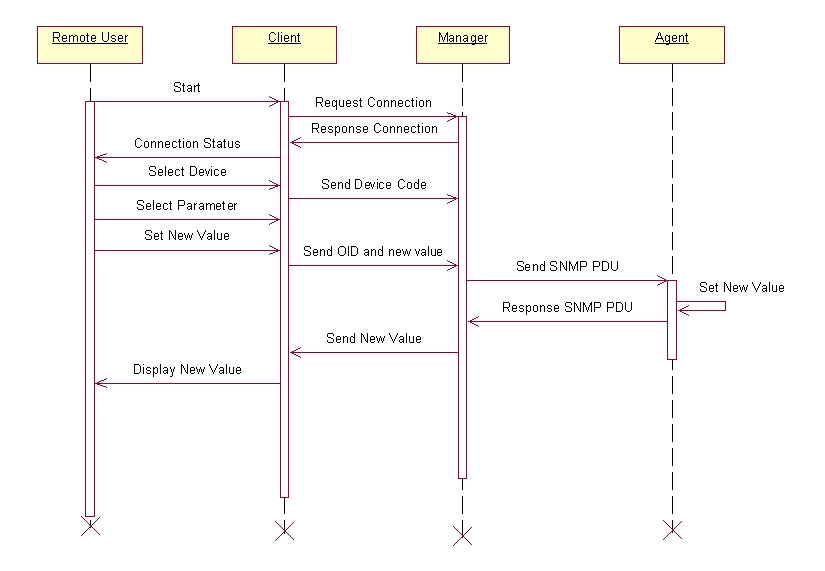
****

Fig: 2.4.5 Sequence Diagram for Get Request and Trap

**Activity Diagram**

Activity diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of systems. An activity diagram is essentially a flowchart, showing flow of control from activity to activity. Whereas interaction diagrams emphasize the flow of control from object to object, activity diagrams emphasize the flow of control from activity to activity.

An activity is an ongoing non-atomic execution within a state machine. Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value.

Activity diagrams are not only important for modeling the dynamic aspects of a system, but also for constructing executable systems through forward and reverse engineering. Graphically an activity diagram is a collection of vertices and arcs

Activity diagram commonly contain

* Activity states and action states
* Transitions
* Objects

Activity states and action states

Activity states can be further decomposed, their activity being represented by other activity diagrams. Activity states are not atomic, meaning that they can be interrupted and, in general, are considered to take some duration to complete.

The executable atomic components like call an operation on an object, end a signal to an object, or even create or destroy an object are called action states because they are states of the system, each representing the execution of an action. Action states cannot be decomposed.



There is no notational distinction between action state and activity state. We can represent by using a lozenge shape (a symbol with horizontal top and bottom and convex sides).

Transitions

When the action or activity of a state completes, flow of control passes immediately to the next action or activity state. This flow can be specified by using transitions to show the path from one action or activity state to the next action or activity state.

Objects

Every building block in the UML may be modeled in terms of their essence or in terms of their instances. An instance is a concrete manifestation of an abstraction to which a set of operations can be applied and which has a state that stores the effects of the operations. Every object has a state and name.

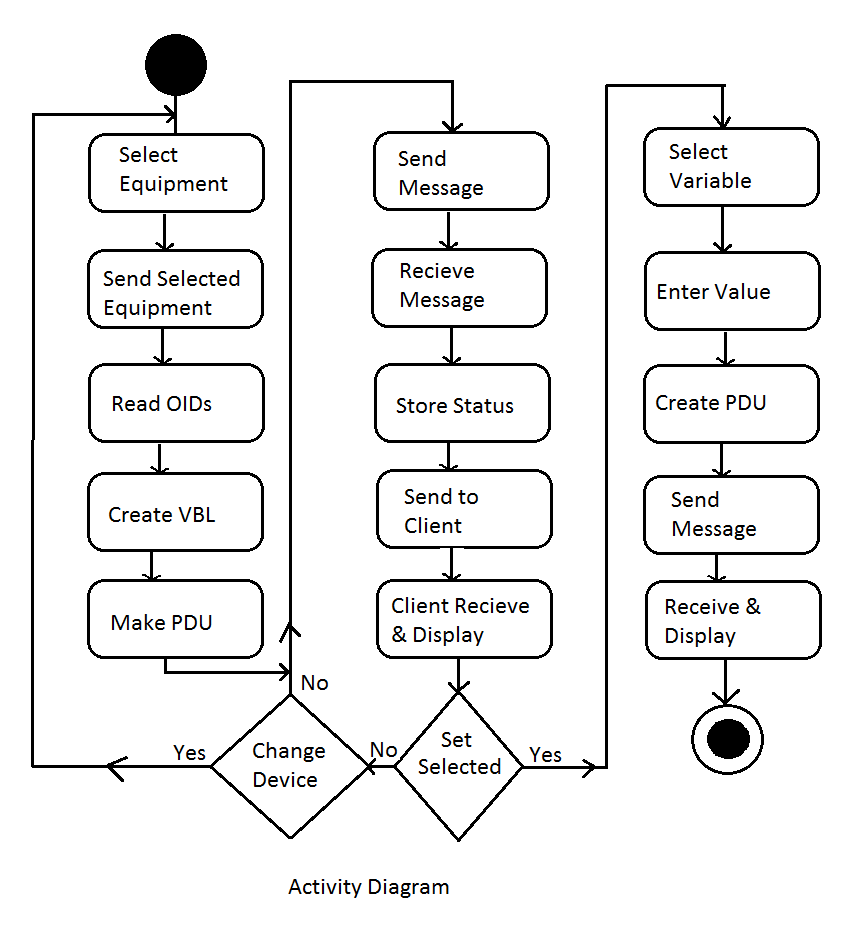


Fig: 2.4.6 Activity Diagram

**Data Flow Diagram**

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

**Level 0**

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are “owned” by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entities.

****

Fig: 2.4.7 Data Flow Diagram Level 0

**Level 1**

The purpose of this level is to show the major and high-level processes of the system and their model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level 1, example the “enquiry” data flow could be split into “enquiry request” and “enquiry results” and still be valid.

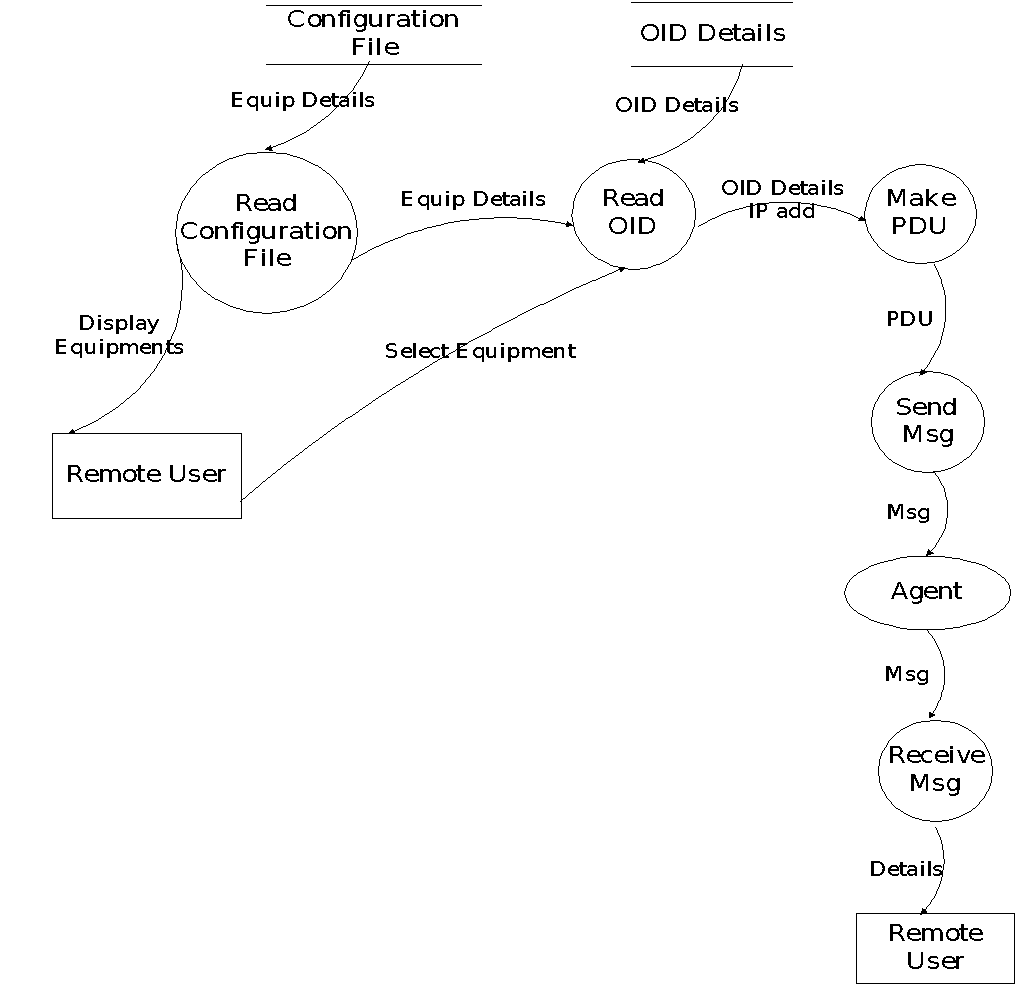
****

Fig: 2.4.8 Data Flow Diagram Level 1

**Level 2**

This level is a decomposition of a process shown in a level-1 diagram, as such there should be a level-2 diagram for each and every process shown in a level-1 diagram. Together they wholly and completely describe process 1, and combined must perform the full capacity of this parent process. As before, a level-2 diagram must be balanced with its parent level-1 diagram.

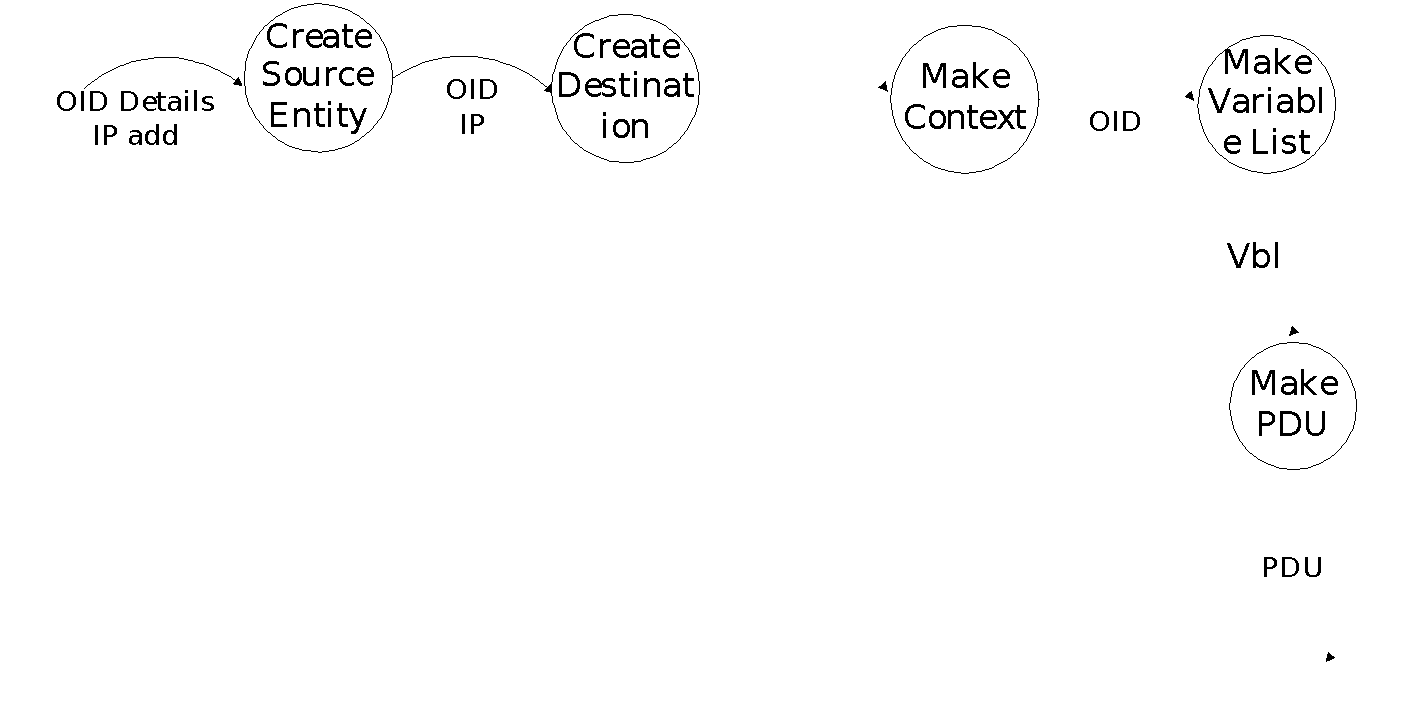
****

Fig: 2.4.9 Data Flow Diagram Level 2-Step1

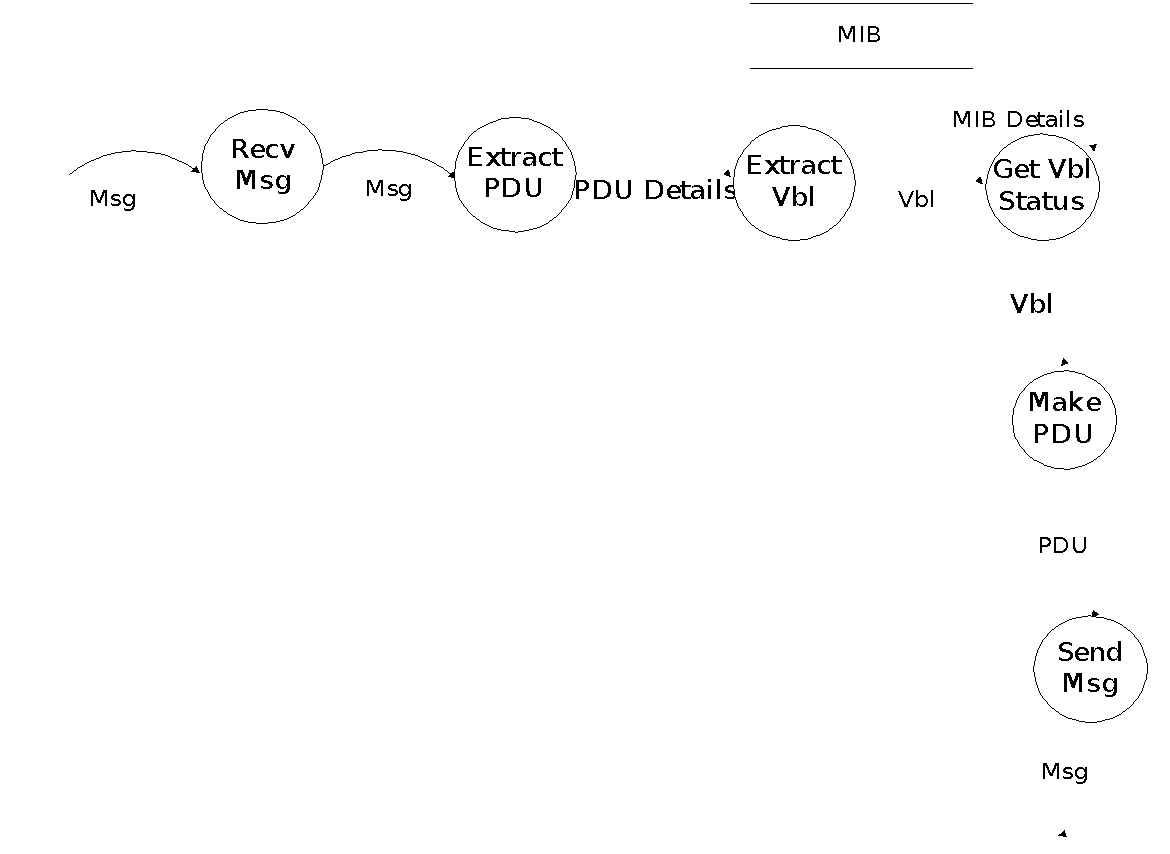
****

Fig: 2.4.10 Data Flow Diagram Level 2-Step2

**2.5 Hardware Requirements**

PROCESSOR : PENTIUM IV 2.6 GHz

RAM : 512 MB DD RAM

MONITOR : 15” COLOR

HARD DISK : 20 GB

CDDRIVE : LG 52X

KEYBOARD : STANDARD102 KEYS

MOUSE : STANDARD

**2.6 Software Requirements**

SOFTWARE USED : Lab Windows/CVI, WinSNMP API

OPERATING SYSYEM : Windows 2000

**2.7 Project Schedule (GANTT CHART)**

A **Gantt chart** is a type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Some Gantt charts also show the dependency (i.e., precedence network) relationships between activities.

| Task Symbol | Task Name | Dependencies | Duration |
| --- | --- | --- | --- |
| T1 | Connection Establishment |  | 18 Days |
| T2 | Client Communication | T1,T3,T4 | 35 Days |
| T3 | Read INI files | T1 | 7 Days |
| T4 | Get Request | T1,T3 | 28 Days |
| T5 | Set Request | T1,T3,T4 | 28 Days |
| T6 | Trap Request | T1,T3 | 28 Days |

Table 2.7.1 Gantt Chart Table

Gantt charts have become a common technique for representing the phases and activities of a project work breakdown structure (WBS), so they can be understood by a wide audience all over the world. Gantt charts only represent part of the triple constraints (cost, time and scope) of projects, because they focus primarily on schedule management. Moreover, Gantt charts do not represent the size of a project or the relative size of work elements, therefore the magnitude of a behind-schedule condition is easily miscommunicated. If two projects are the same number of days behind schedule, the larger project has a larger impact on resource utilization, yet the Gantt does not represent this difference.

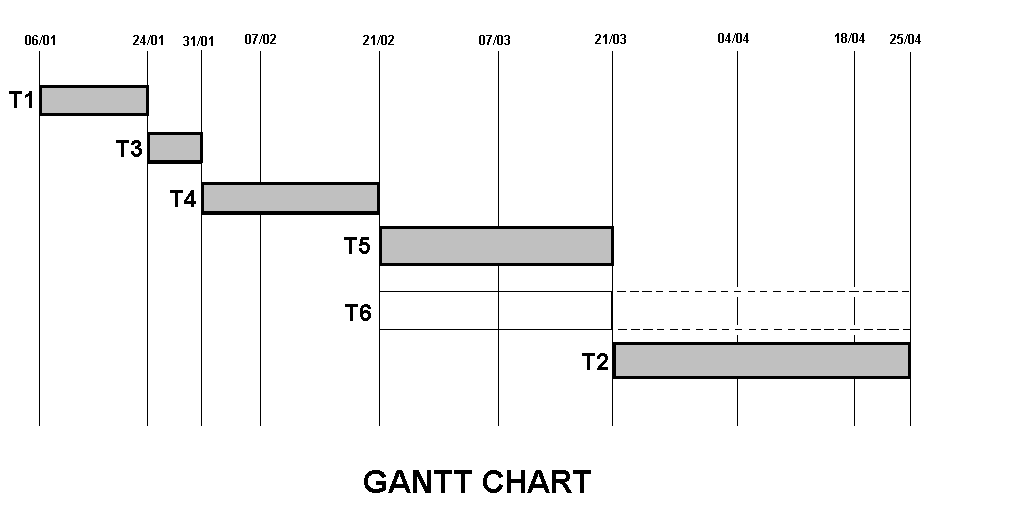


Fig: 2.7.1 Gantt chart

**2.8 Work Breakdown Structure**

A **work breakdown structure (WBS)** in project management and systems engineering, is a tool used to define and group a project’s discrete work elements in a way that helps organize and define the total work scope of the project. A work breakdown structure element may be a product, data, a service, or any combination. A WBS also provides the necessary framework for detailed cost estimating and control along with providing guidance for schedule development and control. Additionally the WBS is a dynamic tool and can be revised and updated as needed by the project manager.

The Work Breakdown Structure is a tree structure, which shows a subdivision of effort required to achieve an objective. In a project or contract, the WBS is developed by starting with the end objective and successively subdividing it into manageable components in terms of size, duration, and responsibility (e.g., systems, subsystems, components, tasks, subtasks, and work packages) which include all steps necessary to achieve the objective. The Work Breakdown Structure provides a common framework for the natural development of the overall planning and control of a contract and is the basis for dividing work into definable increments from which the statement of work can be developed and technical, schedule, cost, and labor hour reporting can be established. A work breakdown structure permits summing of subordinate costs for tasks, materials, etc., into their successively higher level “parent” tasks, materials, etc. For each element of the work breakdown structure, a description of the task to be performed is generated. This technique is used to define and organize the total scope of a project.

The WBS is organized around the primary products of the project (or planned outcomes) instead of the work needed to produce the products (planned actions). Since the planned outcomes are the desired ends of the project, they form a relatively stable set of categories in which the costs of the planned actions needed to achieve them can be collected. A well-designed WBS makes it easy to assign each project activity to one and only one terminal element of the WBS. In addition to its function in cost accounting, the WBS also helps map requirements from one level of system specification to another. The work is break down into modules and functions as in the use case diagram.

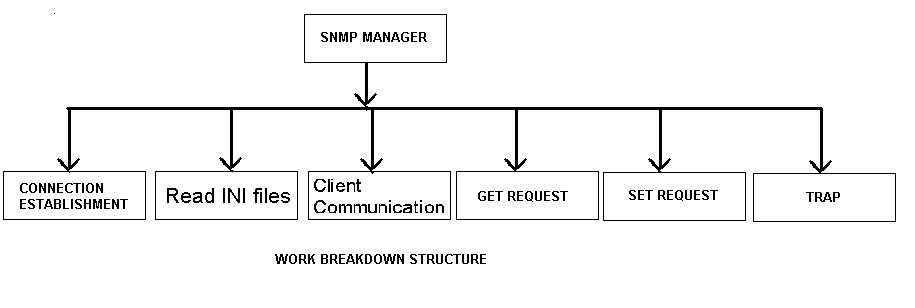


Fig 2.8.1 Work Breakdown Structure

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 High Level Design**

A **high-level design** provides an overview of a solution, platform, system, product, service, or process. The highest level solution design should briefly describe all platforms, systems, products, services and processes that it depends upon and include any important changes that need to be made to them. A high-level design document will usually include a high-level architecture diagram depicting the components, interfaces and networks that need to be further specified or developed. In addition, there should be brief consideration of all significant commercial, legal, environmental, security, safety and technical risks, issues and assumptions. The idea is to mention every work area briefly, clearly delegating the ownership of more detailed design activity whilst also encouraging effective collaboration between the various project teams. Finally, every type of end-user should be identified in the high-level design and each contributing design should give due consideration to customer experience.

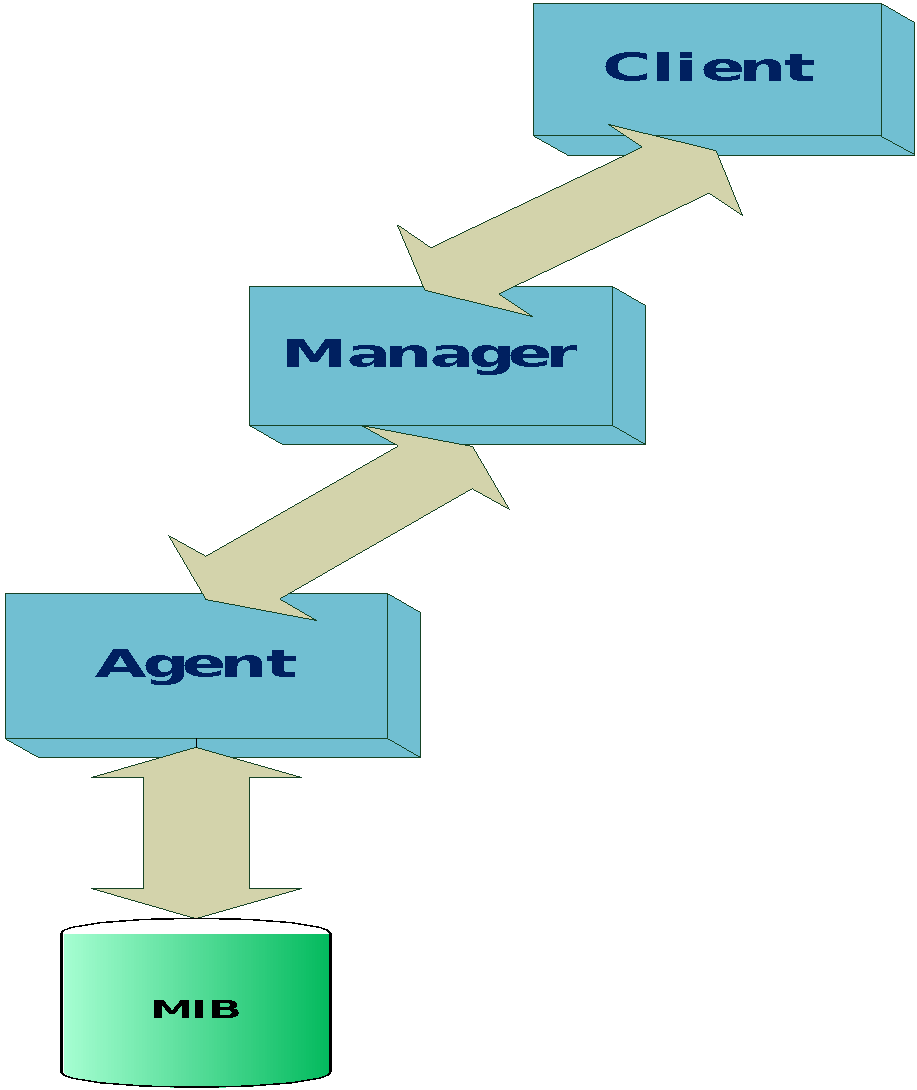


Fig 4.1.1 High Level Design

**4.2 Low Level Design**

Low Level Design (LLD) is like detailing the HLD. It defines the actual logic for each and every component of the system. Class diagrams with all the methods and relation between classes comes under LLD. Programs specs are covered under LLD.

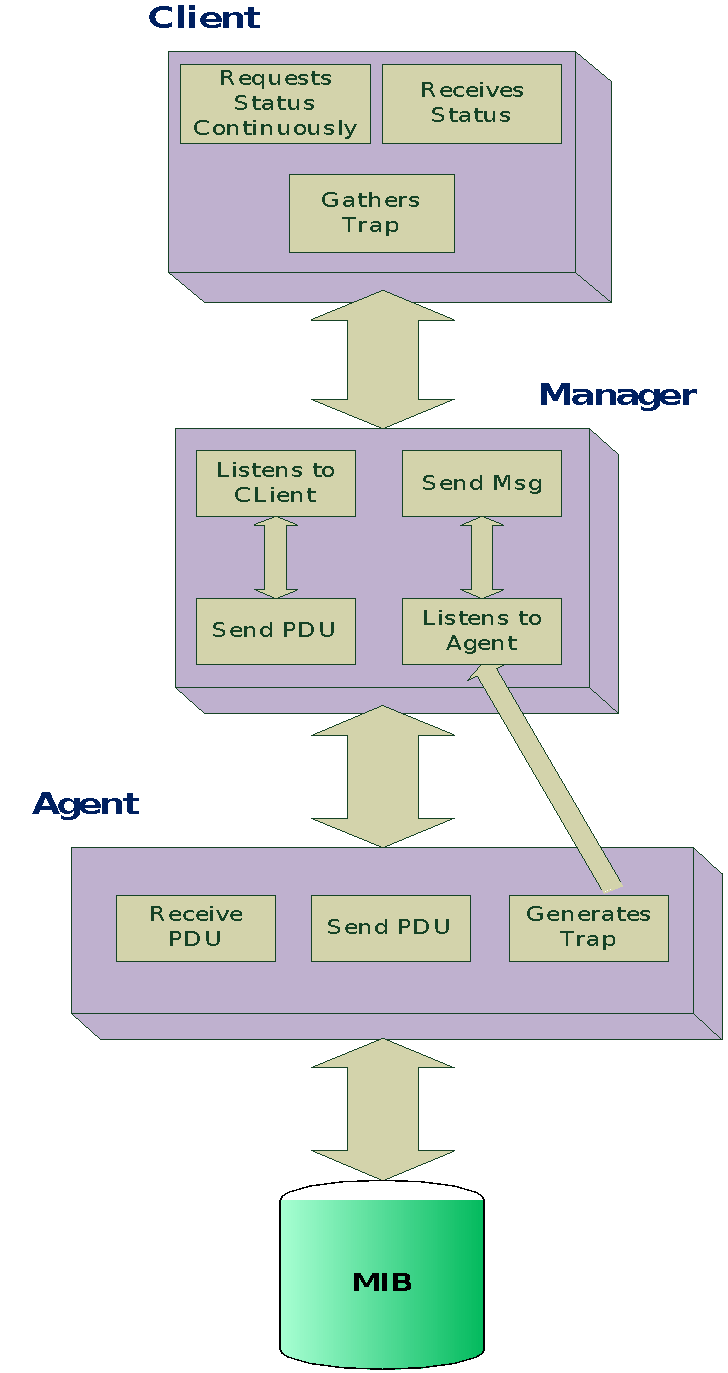


Fig 4.2.1 Low Level Design

LLD describes each and every module in an elaborate manner so that the programmer can directly code the program based on this. There will be at least 1 document for each module and there may be more for a module. The LLD will contain: - detailed functional logic of the module in pseudo code - database tables with all elements including their type and size - all interface details with complete API references (both requests and responses) - all dependency issues -error message listings - complete input and outputs for a module.

**4.3 Test Cases**

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is functioning correctly. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites.

**Formal test cases**

In order to fully test that all the requirements of an application are met, there must be at least two test cases for each requirement: one positive test and one negative test; unless a requirement has sub-requirements. In that situation, each sub-requirement must have at least two test cases. Keeping track of the link between the requirement and the test is frequently done using a traceability matrix. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted.

A formal written test-case is characterized by a known input and by an expected output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post condition.

**Informal test cases**

For applications or systems without formal requirements, test cases can be written based on the accepted normal operation of programs of a similar class. In some schools of testing, test cases are not written at all but the activities and results are reported after the tests have been run.

In scenario testing, hypothetical stories are used to help the tester think through a complex problem or system. These scenarios are usually not written down in any detail. They can be as simple as a diagram for a testing environment or they could be a description written in prose. The ideal scenario test is a story that is motivating, credible, complex, and easy to evaluate. They are usually different from test cases in that test cases are single steps while scenarios cover a number of steps.

The following are the Test Cases with the result

| Test Case ID | Test Case | Approach | Input | Output | Result |
| --- | --- | --- | --- | --- | --- |
| 1 | Periodic check of management data of the device. | Functional | Sending Set Request | Getting Response | Passed |
| 2 | Efficient communication with the agent. | Functional | Client connection and requests | Status display at Client | Passed |
| 3 | Display the status of all the data variables. | Functional | Response PDU | Display in UI | Passed |
| 4 | Display the trap messages from all devices. | Functional | Registering for Trap | Receiving Trap PDU | Passed |
| 5 | Should maintain multiple agents concurrently. | Functional | Many agents connection request | Status display on multiple agents | Passed |
| 6 | User should select any device at a given instance | Functional | Select device from UI | New status display in UI | Passed |
| 7 | Application has to alert the manager on any deviation of expected performance of device. | Functional | Receiving the trap Messages | Display of Trap messages in UI textbox | Passed |
| 8 | Product should be Scalable to add any number of equipments. | Functional | Connecting many devices | Devices names display in UI | Passed |
| 9 | Application has to alert the user of any network failure. | Functional | Disconnecting the Network | Application Disconnect | Passed |
| 10 | Allow the manager to control the equipments at any time. | Functional | Value to be Set | Set Response reply | Passed |
| 11 | Software should be available all the time. | Functional | Operating the application | 24\*7 operation | Passed |
| 12 | Subsequent packet received must be reported | Functional | Sending multiple PDU’s | Receiving response and display of all. | Passed |
| 13 | Health parameter that has write access mode must be configurable | Functional | INI file fields | Button activation for settable Vbl’s | Passed |
| 14 | User should be able to set the rate for periodic update of the status | Functional | Entering rate value in UI | Periodic update rate change | Passed |

Table 4.3.1 Test Cases

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

**5.1 Architecture Development**

The Architecture diagram depicts the overall architecture of the project. It gives us the overview of the project. The Architecture diagram is of two types, high level and low level architecture diagrams. The high level architecture diagram consists of only the overview of the architecture of the project without going into the particular details of the modules of the project.

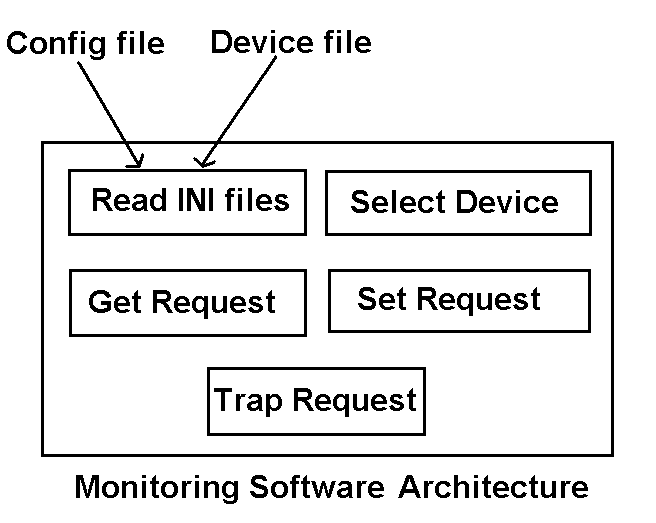


Fig 5.1.1 High Level Architecture Diagram

The low level architecture diagram gives us the each and every detail regarding the modules of the project. It gives us a view of what all the modules are used in the project and again what all the subparts are contained in each module. It is just the step prior to coding. The only step remaining after drawing the low level architecture diagram is the coding of the project.

In the current application the high level architecture diagram gives us the view of the major modules of the project. All these major modules are expanded into subparts in the low level architecture diagram. The Read files module takes the INI files as input and contains the modules for reading the INI files when required. Similarly the Select Device module consists of methods for connecting to the client using the TCP/IP protocol and retrieving the device selected and the refresh rate from the client. The Get Request module consists of the methods for getting the OIDS of the device selected by the user and getting its values from the Device. The Set Request consists of the methods for retrieving the OID and the value to be set from user and setting the corresponding value in the MIB of the device. The Trap Request module consists of the methods for registering the manager with the device for receiving messages about any peculiar conditions or errors.

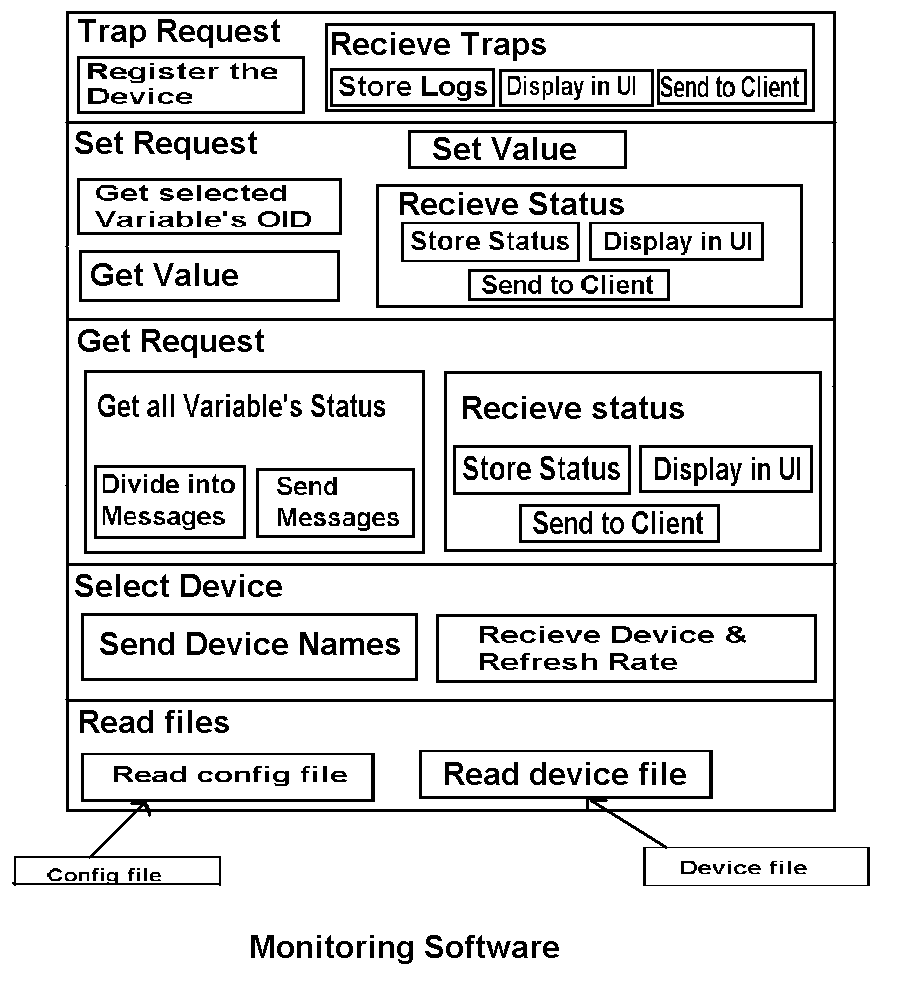


Fig 5.1.2 Low level Architecture Diagram

**Hardware Implementation:**

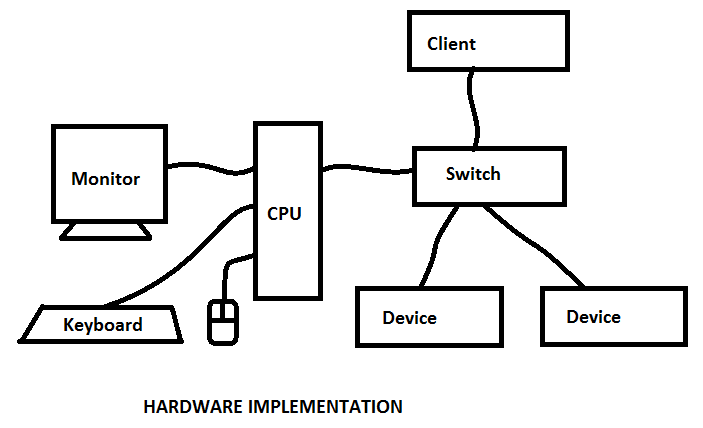


Fig 5.1.3 Hardware Implementation

**Software Implementation:**

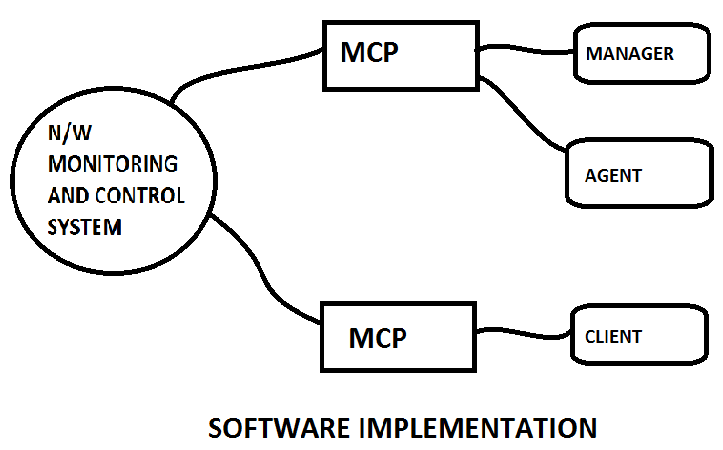


Fig 5.1.4 Software Implementation

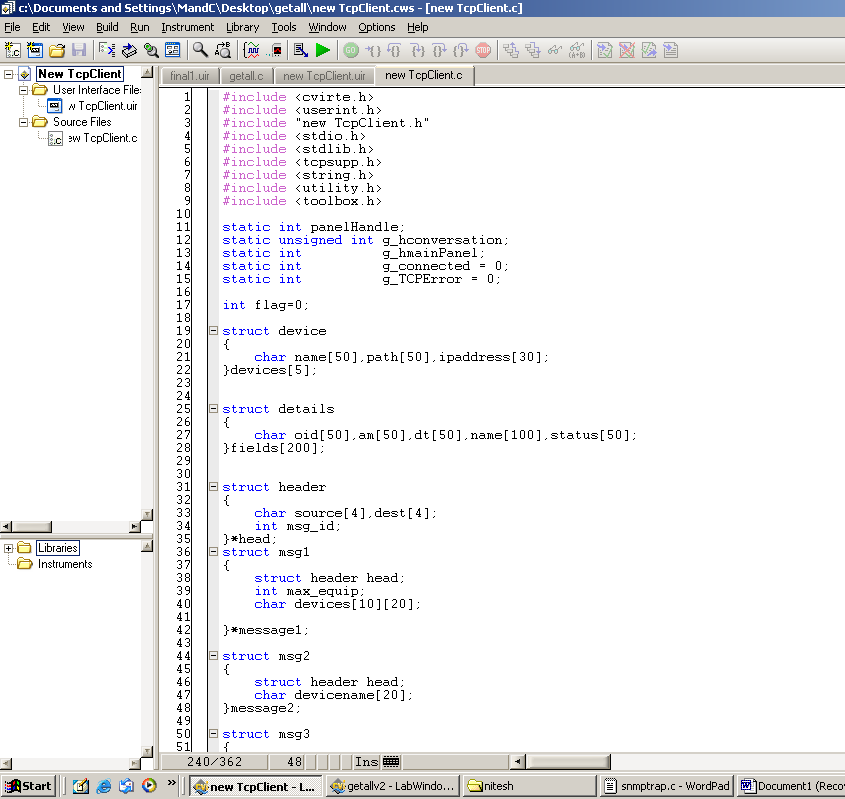
**Code:**

Fig 5.1.5 Code Screenshot 1

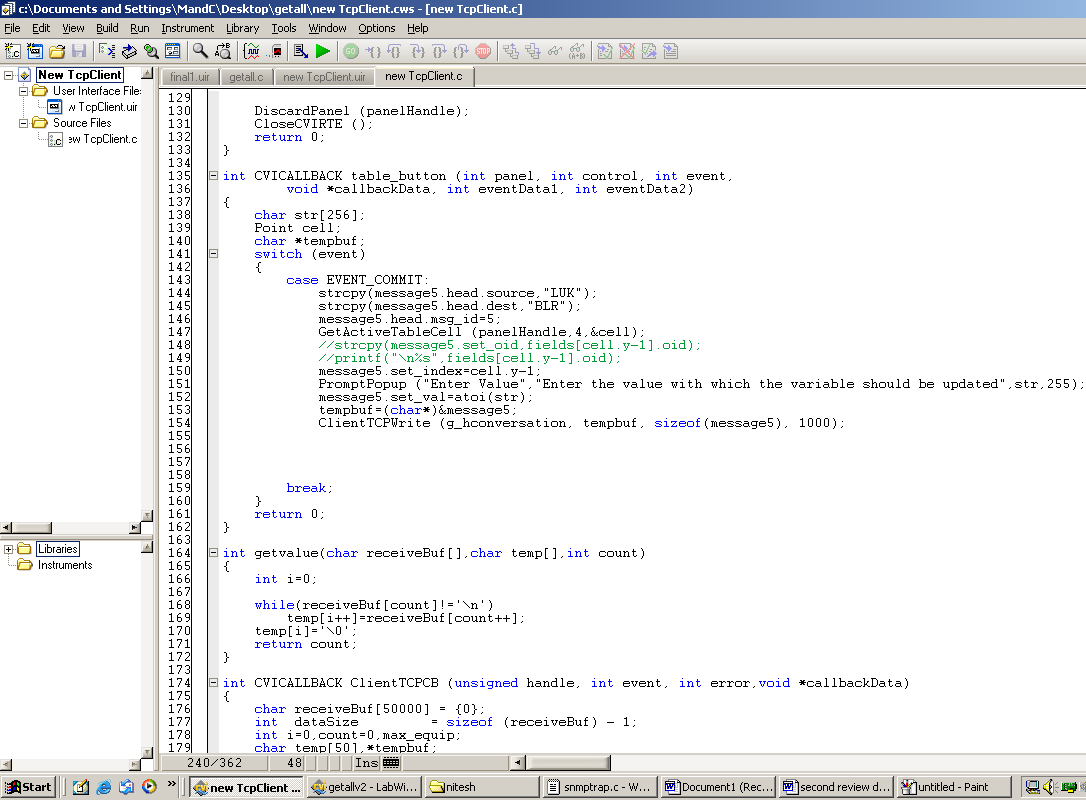
****

Fig 5.1.6 Code Screenshot 2

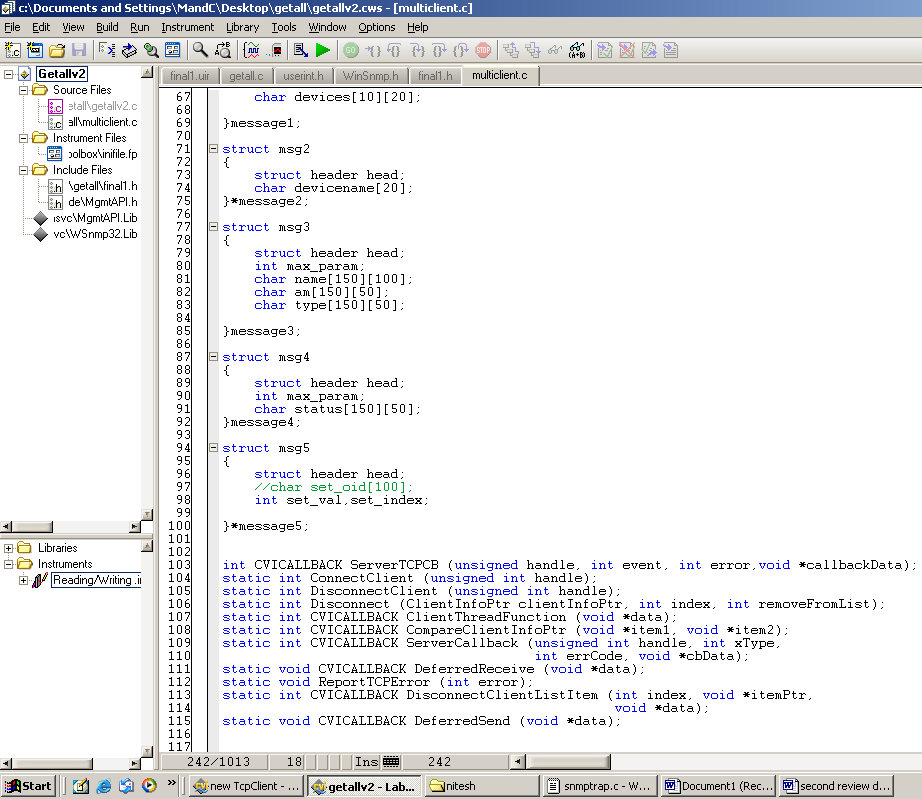
****

Fig 5.1.7 Code Screenshot 3

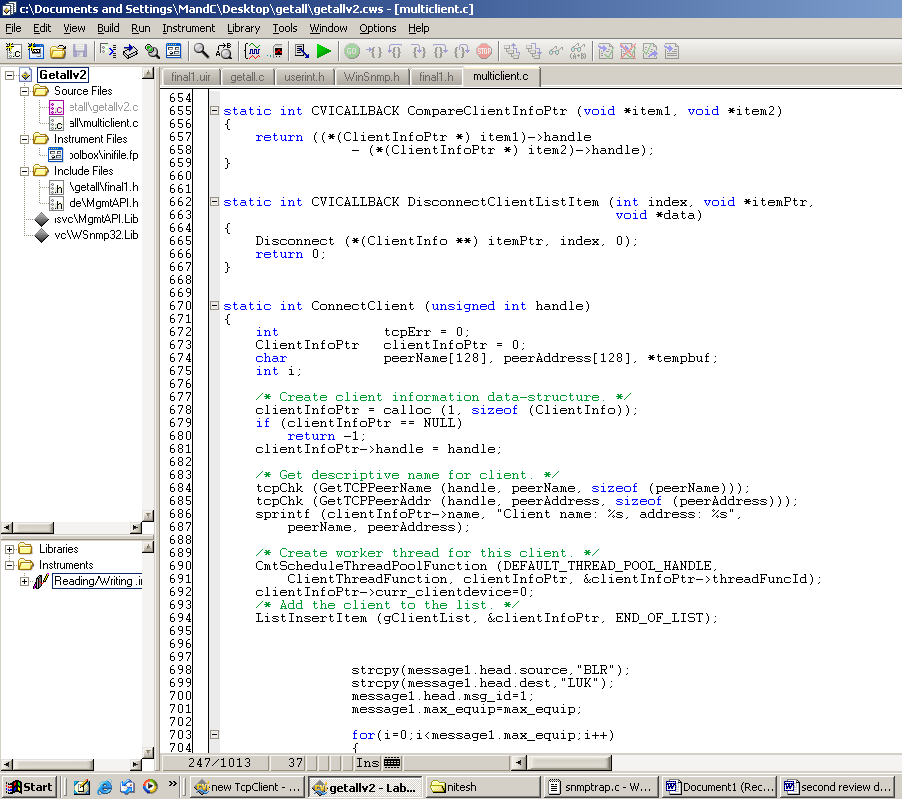
****

Fig 5.1.8 Code Screenshot 4

**5.2 Unit Testing**

Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. In object-oriented programming a unit is usually a method. Unit tests are created by programmers or occasionally by white box testers.

Ideally, each test case is independent from the others: substitutes like method stubs, mock objects,fakes and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits. Unit tests find problems early in the development cycle.

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer’s requirements.

The following are the areas of the project must be unit-tested and signed-off before being passed on to integration Testing:

* Reading INI files
* Callback functions for UI Controls.
* Get Request module at Manager
* Set Request module at Manager
* Trap Request module at Manager
* Multiple Client connection
* Communication with Client via TCP/IP
* Displaying Received Messages on UI at the Client
* Displaying Received Messages on UI at the Manager

Each module can be tested using the following two strategies

5.2.1 Black Box testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been used to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external data access
* Performance errors
* Initialization and termination errors

5.2.2 White Box testing

In this test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been used to generate the test cases in the following cases.

**5.3 Integration Testing**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

During the repeated cycles of identifying bugs and taking receipt of new builds (containing bug fix code changes), there are several processes which are common to this phase across all projects. These include the various types of tests: functionality, performance, stress, configuration, etc. The project should plan for a minimum of 2-3 cycles of testing.

At each iteration, a debriefing should be held. Specifically, the report must show that to the best degree achievable during the iteration testing phase, all identified severity 1 and severity 2 bugs have been communicated and addressed.

The modules which are to be integrated for the manager program are

* + Reading INI files
  + Callback functions for UI Controls.
  + Get Request module at Manager
  + Set Request module at Manager
  + Trap Request module at Manager
  + TCP/IP Manager Sender
  + TCP/IP Manager Receiver
  + Displaying Received Messages on UI at the Manager

The modules which are to be integrated for the Client program are

* + Displaying Received Messages on UI at the Client
  + TCP/IP Client Sender
  + TCP/IP Client Receiver
  + Callback functions for UI Controls.

**5.4 Test Results**

Unit Testing is conducted by the Developer during code development process to ensure that proper functionality and code coverage have been achieved by each developer both during coding and in preparation for acceptance into iterations testing.

The following are the modules which are successfully tested:

* Reading INI files
* Callback functions for UI Controls.
* Get Request module at Manager
* Communication with Client via TCP/IP
* Displaying Received Messages on UI at the Client
* Displaying Received Messages on UI at the Manager
* Set Request module at Manager
* Trap Request module at Manager
* Multiple Client connection

**CHAPTER 6**

**RESULTS AND DISCUSSION**

**6.1. Output/Results**

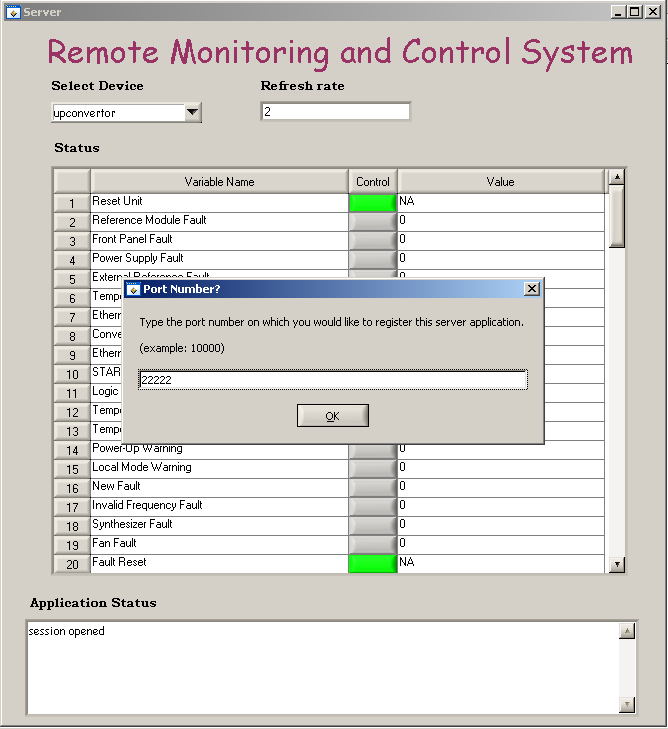
****

Fig 6.1.1 Server Initialization

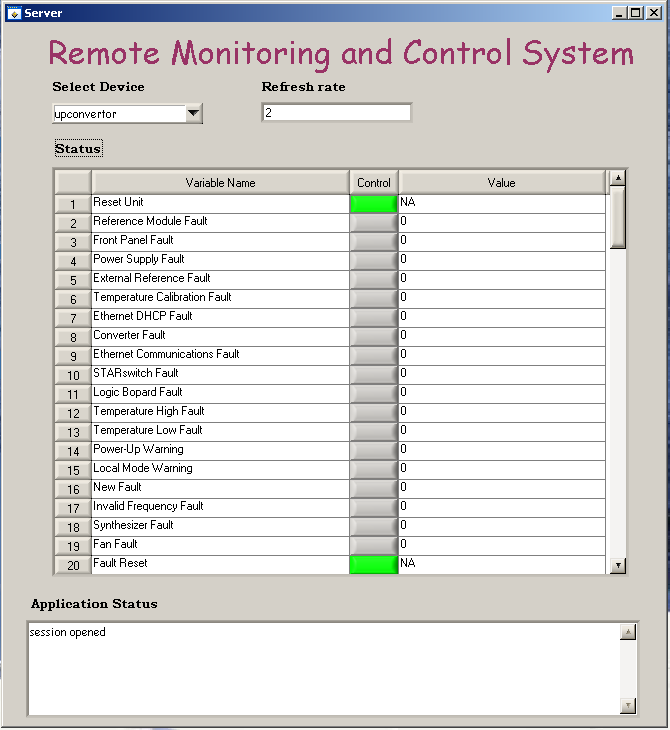
****

Fig 6.1.2 Server UI

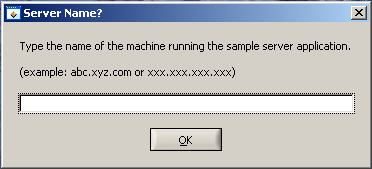
****

Fig 6.1.3 Client Initialization with sever IP

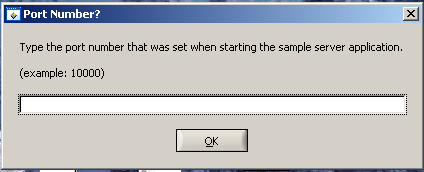
****

Fig 6.1.4 Client Initialization with sever Port Number

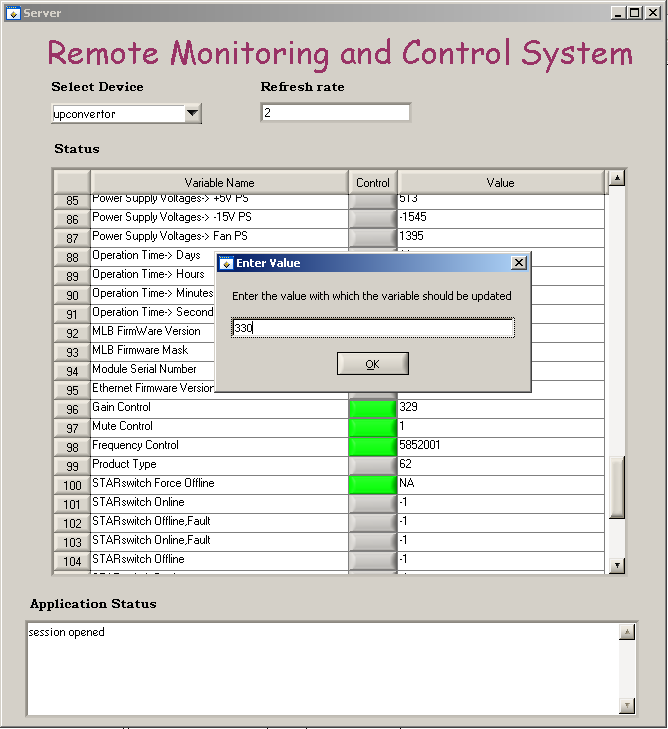
****

Fig 6.1.5 Set Operation at Sever

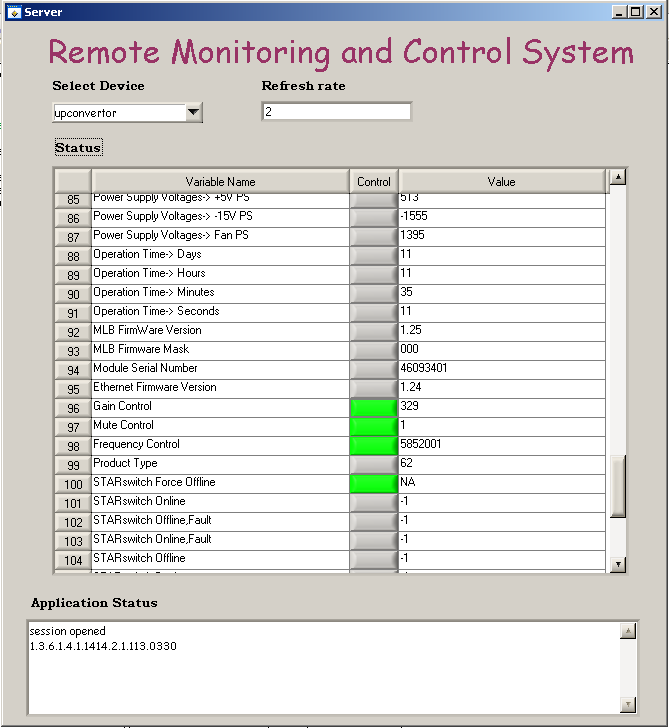
****

Fig 6.1.6 Result of Set Operation at Server

****

Fig 6.1.7 Set Operation at Client

****

Fig 6.1.8 Output Client UI

**6.2. Result Analysis**

The results are shown as screen shots in the document and all the results are par with the Performance and Functional requirements. All the test cases were checked and the application has passed in all test cases. All the Unit testing and Integration testing results were accurate with the expected results. The application can run 24\*7 without any interruption. The application will be available all the time and perform very well.

**6.3 Discussion**

Thus an application has been created for the Remote monitoring and control of the ground station equipments on SNMP interface. All the equipments which support SNMP interfacing can be remotely monitored and controlled successfully. The software is successful and any number of equipments can be added without any performance degradation. This application will act as a basis for communication link monitoring between the satellite and the ground station equipments. Building this application will add an additional feature of SNMP interface to the existing system of monitoring them with RS232, GPIB, TCP/IP interfaces.

**CHAPTER 7**

**CONCLUSION AND FUTURE WORK**

**7.1 Conclusion**

Some of the latest equipments are supplied with SNMP protocol for its monitoring and control. This application is developed to provide monitoring and control support on SNMP interface for a given set of equipments. The application is built in such a way that any other equipment with SNMP interface could be integrated at any point of time. Building this interface enhanced existing monitoring and control system at ISTRAC by

1. Providing additional interface support
2. Adding any new equipment in future
3. Monitor communication link by monitoring routers, switches etc., that typically support SNMP for monitoring, thus avoiding commercial Network Monitoring Systems.

Thus a new application has been built for interfacing with the SNMP supporting devices with great flexibility so as to avoid commercial software packages thus contributing to the organization and the country

**7.2 Scope of Future Work**

This application of Remote Monitoring and Control System has been built for the monitoring of any device that supports SNMP interfacing. This application can be smartly enhanced to integrate all the interfacing protocols so that all the ground station equipments can be monitored and controlled with the same software with the interfacing protocol it supports. This application can also be modified for the communication link monitoring between the satellite and the ground station equipments. Further the security of the application can be enhanced by authenticating the client connection with the server.

**CHAPTER 8**

**REFERENCES**

**Books**

1. Vertex RSI, Synthesized Converter System Operation and Maintenance Manual
2. Douglas, R., Mauro, Essential SNMP
3. Paul Simoneau, Hands on SNMP

**Important Links**

1. http://www.cisco.com/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ap1.htm
2. [http://msdn.microsoft.com/enus/library/aa379100%28v=vs.85%29.aspx](http://msdn.microsoft.com/enus/library/aa379100(v=vs.85).aspx)
3. [www.ni.com/analysis/lwtools\_analysis.htm](http://www.ni.com/analysis/lwtools_analysis.htm)